# Virtuoso Spectre Circuir Simulator RF Analysis User Guide

**Product Version 7.2**
**December 2009**

# Contents

# Preface

Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) provides functionality designed for the needs of RF designers. Spectre RF

- Supports the efficient calculation of the operating point, transfer function, noise, and distortion of common RF and communication circuits, such as mixers, oscillators, sample and holds, and switched capacitor filters.
- Supports a multi-technology simulation (MTS) mode that enables the simulation of a system consisting of blocks designed with different processes, such as RF System-in-Package (SIP).

This user guide assumes that you are familiar with:

- RF circuit design.
- SPICE simulation.
- The Virtuoso® analog design environment (ADE).

Spectre and Spectre RF features are available in different tiers. The L tier offers basic design creation and implementation capabilities. The XL tier introduces new technologies and advancements in automation. The following table provides an overview of the features supported by each tier of products.

**Features Supported in Spectre L and Spectre XL Tiers**

| Features | L Tier | XL Tier | GXL Tier |
|---|---|---|---|
| DC, AC, and transient analysis. | X | X | X |
| Noise, transfer function, and sensitivity analysis. | X | X | X |
| Transient noise analysis. | X | X | X |
| Monte Carlo and parametric statistical support. | X | X | X |
| Built-in measurement description language (MDL). However, MDL does not support RF analysis. | X | X | X |
| Parametric sweep. | X | X | X |

**Features Supported in Spectre L and Spectre XL Tiers,** *continued*

| Features | L Tier | XL Tier | GXL Tier |
|---|---|---|---|
| Multi-threading. (L Tier is limited to 4 CPUs; XL Tier to 8 CPUs; GXL Tier has no limit.) | X | X | X |
| Periodic and quasi-periodic steady state analysis (PSS and QPSS) based on harmonic balance and shooting Newton. | | X | X |
| Periodic and quasi-periodic noise analysis (PNoise, QPNoise). | | X | X |
| Periodic and quasi-periodic small signal analysis (PAC, PXF, PSP, QPAC, QPXF, QPSP). | | X | X |
| Periodic stability analysis (PSTB). | | x | x |
| Time-domain and frequency-domain envelope analysis. | | X | X |
| Perturbation-based rapid IP2 and IP3. | | X | X |
| Cosimulation with Simulink® from The MathWorks. | | X | X |
| MMSIM Toolbox for MATLAB® from The MathWorks. | | X | X |
| Spectre Turbo. | | X | X |
| Spectre Parasitics. | | | X |

# Licensing for Spectre RF

To run the G, GX, and GXL tier features of Spectre RF, you must have access to a corresponding license or combination of licenses. The order in which these licenses are used is determined either by default or by using the `+lorder` option.

The `+lorder` option lets you specify a custom license checkout order for simulation. Spectre checks for a license in the specified order.

```
+lorder licenseList
```

| | |
|---|---|
| `licenseList` | A list of licenses. Use `:` between the license names when defining the order. For example,<br>`+lorder Virtuoso_Multi_mode_Simulation:Virtuoso_Spectre`<br>specifies that the token license is checked before the `Virtuoso_Spectre` license. |

The default license checkout order for Spectre RF is

`Virtuoso_Multi_mode_Simulation:Virtuoso_Spectre_XL:Virtuoso_Spectre_GXL`

where the `Virtuoso_Multi_mode_Simulation` license takes 1 token.

If the `Virtuoso_Spectre_XL` license or the `Virtuoso_Spectre_GXL` license is used, no other license is required to run Spectre RF. Otherwise, one of the following add-on licenses is required and the default checkout order for them is

`Virtuoso_Spectre_RF:Virtuoso_Multi_mode_Simulation`

where the `Virtuoso_Multi_mode_Simulation` license takes one additional token.

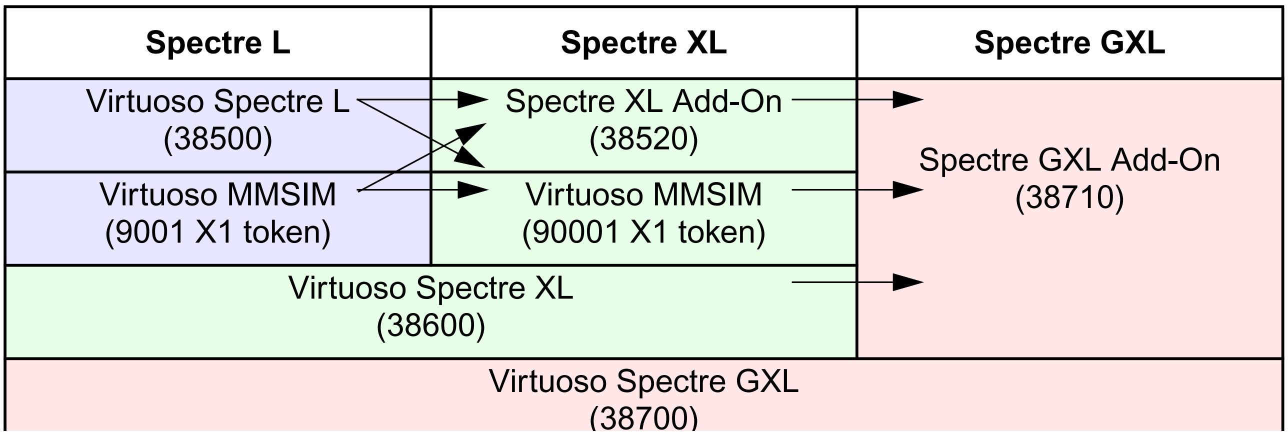
To summarize these rules, Spectre RF can run with two types of licenses, either a Spectre XL license or 2 MMSIM tokens. The Spectre XL license is available as the `Virtuoso_Spectre_XL` license or as the `Virtuoso_Spectre_RF` add-on license.

The following table lists the licenses for Spectre RF.

| **License Name** | **Feature** |
|---|---|
| `Virtuoso_Spectre_RF` | License associated with *Virtuoso Spectre Circuit Simulator XL Add on Option to 38500* (38520). |
| `Virtuoso_Multi_mode_Simulation` | Token license associated with *Virtuoso Multi-mode Simulation Product* (90001). Two tokens are checked out simultaneously for Spectre Turbo, MathWorks, or RF analysis. |
| `Virtuoso_Spectre` | License associated with the *Virtuoso Spectre Circuit Simulator L* (38500). |
| `Virtuoso_Spectre_XL` | License associated with *Virtuoso Spectre Circuit Simulator XL* (38600). This license is a superset of `Virtuoso_Spectre` and `Virtuoso_Spectre_RF`. |
| `Virtuoso_Spectre_GXL` | License associated with *Virtuoso Spectre Circuit Simulator GXL* (38700). This license is a superset of `Virtuoso_Spectre` and `Virtuoso_Spectre_RF`. |

| License Name | Feature |
|---|---|
| `Virtuoso_Spectre_GXL_Option` | License associated with *Virtuoso Spectre Simulator GXL ADD On Option to 38600* (38710) |

The following illustration summarizes the possible licensing combinations.




The following licenses are no longer sold but might be available at some sites.

| License Name | Feature |
|---|---|
| `32500` | License associated with Virtuoso Spectre Circuit Simulator (32500). |
| `SpectreRF` | License associated with Virtuoso Spectre RF Simulation Option (32520). |

# Related Documents for Spectre RF

This user guide contains information about the RF functionality. The following documents provide more information about Spectre RF and related products.

- For a complete description of Spectre RF functionality you also need to refer to the Virtuoso Spectre Circuit Simulator documentation set. See
    - *Virtuoso Spectre Circuit Simulator Reference*

- *Virtuoso Spectre Circuit Simulator User Guide*

- For in-depth information and detailed examples of Spectre RF usage, see the documents listed in Appendix M, "Documents That Ship in the Software Hierarchy." So that the listed documents can be as up-to-date as possible, they are shipped with the MMSIM hierarchy rather than with the standard Cadence document set.

- To learn more about the Analog Circuit Design Environment, consult the *Virtuoso® Analog Design Environment User Guide*.

- To learn more about Spectre RF, see the reference information and theoretical concepts in *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

## Third Party Tools

To view any `.swf` multimedia files, you need:

- Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the Adobe website.

- Speakers and a sound card installed on your computer for videos with audio.

## Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

| | |
|---|---|
| `text` | Indicates text you must type exactly as it is presented. |
| *`argument`* | Indicates text that you must replace with an appropriate argument. |
| `[ ]` | Denotes an optional argument. When used with vertical bars, they enclose a list of choices from which you can choose one. |
| `{ }` | Used with vertical bars, they denote a list of choices from which you must choose one. |
| `\|` | Separates a choice of options. |
| *text* | Indicates names of manuals, menu commands, form buttons, and form fields. |

# 1

# Spectre RF Simulation Form Reference

The Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) simulation forms include the Choosing Analyses form, the Options forms, and the Results forms. The simulation forms change to display only the fields relevant for the currently selected analysis.

The field description topics are presented in the following sections.



Within each section, the form field descriptions are arranged alphabetically according to the top-level headings on the forms. The top-level headings are usually found at the leftmost margin of the form.

The Spectre RF wizards are described in



The chapter concludes with a consolidated reference to the Spectre RF forms.

# Choosing Analyses Form

Use the Choosing Analyses form in the Analog Circuit Design Environment (the ADE window) to select and set up RF simulations.

To open the Choosing Analyses form,

➤ In the ADE window, choose the *Analyses – Choose* command to open the Choosing Analyses form.

Figure 1-1 on page 2 shows the Choosing Analyses form for the Envelope analysis. The content of the Choosing Analyses form changes depending on the analysis selected.

**Figure 1-1 Choosing Analyses Form**


The Analysis section at the top of the Choosing Analyses form displays the available analyses, including the Spectre RF analyses (ENVLP, PSS, PAC, PSTB, Pnoise, PXF, PSP, QPSS, QPAC, QPnoise, QPXF, and QPSP). When you highlight an analysis in this section, the form changes to display the title of the analysis (directly below the analysis section), and below the title, relevant parameters for the selected simulation.

At the bottom of the form, highlight *Enabled* to select and run the analysis with the next simulation. Click *Options* to display the Options form for the selected analysis. Each Options form displays only the parameters relevant for that particular analysis.

The Spectre RF analyses are

- The periodic large-signal PSS, periodic steady state analysis
- The periodic large-signal QPSS, quasi-periodic steady state analysis
- The periodic small-signal analyses: PAC, PSTB, PSP, Pnoise, and PXF
- The quasi-periodic small-signal analyses: QPAC, QPSP, QPnoise, and QPXF
- The envelope ENVLP analysis

When you highlight an analysis type, the Choosing Analyses form changes to allow you to specify information for that simulation.

When your simulation requires that two analyses be run (for example, a PSS large-signal analysis followed by a Pnoise small-signal analysis), the Choosing Analyses form maintains the simulation set-up data for the two simulations interactively. For example, when you highlight *pnoise*, the values displayed in the Choosing Analyses form reflect the information you entered for the Pnoise analysis. When you highlight *pss*, the values displayed in the Choosing Analyses form reflect the information you entered for the PSS analysis.

Run the periodic small-signal PAC, PSP, Pnoise and PXF analyses after a large-signal PSS analysis. Run the quasi-periodic small-signal QPAC, QPSP, QPnoise and QPXF analyses after a large-signal QPSS analysis.

# Field Descriptions for the Choosing Analyses Form

The following sections describe the panes and fields that can appear on the Choosing Analyses form, independently of the analysis that is selected. The descriptions are arranged alphabetically, according to the labels that are usually found along the left side of the form. If you are looking for descriptions of the Choosing Analyses form as they appear for a particular analysis, see “Choosing Analyses Form” on page 151.

The following sections are:

## Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)

Quickly adjusts the simulator parameters.


The `errpreset` parameter quickly adjusts the simulator accuracy parameters to fit your needs. In most cases, `errpreset` should be the only parameter you need to adjust.

For a fast simulation with reasonable accuracy, set `errpreset` to *`liberal`*.

For greater accuracy, set `errpreset` to *`moderate`*.

For maximum accuracy, set `errpreset` to *`conservative`*.

The effect of `errpreset` on other parameters varies depending on the type of analysis to which you are applying it.

For details see the following sections in *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

- The errpreset Parameter in PSS Analysis
- The errpreset Parameter in QPSS Analysis
- The errpreset Parameter in Envlp Analysis

## Add Specific Points (HBAC, HBNOISE)

Specifies specific points to be added to the set of swept values.


If more than one value is entered, separate the values with spaces. For example, you might type

```
3 13
```

into the field to add two points to the sweep defined by `start` = 0, `stop` = 20, `step` =5, so that the swept values are 0, 3, 5, 10, 13, 15, 20.

## Additional Time for Stabilization (tstab) (PSS, QPSS)

For the shooting engine, specifies an amount of additional time to allow for the circuit to settle.


Use `tstab` if the circuit exhibits more than one periodic solution and you want only one. A long `tstab` can also improve convergence.

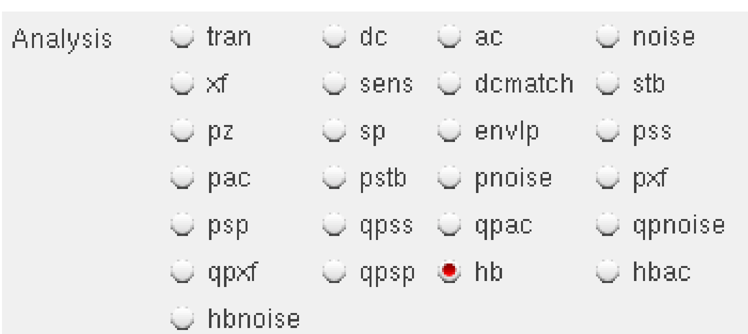
## Additional Time for Transient-Aided HB (tstab) (PSS, QPSS, HB)

For the harmonic balance engine, specifies an amount of additional time to allow for the circuit to settle.


Use `tstab` if the circuit exhibits more than one periodic solution and you want only one. A long `tstab` can also improve convergence.

## Analysis

Selects the Spectre RF analysis type and controls whether instantiated measurement and analysis cells are used.




The RF analyses are

- The periodic large-signal PSS analysis
- The quasi-periodic large-signal QPSS analysis

- The periodic small-signal analyses: PAC, PSTB, PSP, Pnoise, and PXF
- The quasi-periodic small-signal analyses: QPAC, QPSP, QPnoise, and QPXF
- The Envelope analysis, ENVLP
- The harmonic balance analyses: HB, HBAC. and HBnoise

When you highlight an analysis type, the Choosing Analyses form changes to allow you to specify information for that simulation. Below the Analysis buttons, the analysis title changes to the name of the analysis you select. Highlighting *measure* displays the Measurement Analysis pane of the Choosing Analyses form.

When your simulation requires that two analyses be run (for example, a PSS analysis followed by a Pnoise small-signal analysis), the Choosing Analyses form maintains the simulation set-up data for both simulations. You can edit the data for both simulations interactively. For example, when you highlight *pnoise*, the values displayed in the Pnoise Choosing Analyses form reflect the information you entered for the Pnoise analysis.

Run the periodic small-signal analyses, PAC, PSTB, PSP, Pnoise, and PXF, after a large-signal PSS analysis. Run the quasi-periodic small-signal QPnoise analysis after a QPSS analysis. Run the HBnoise analysis after an HB analysis.

## Beat Frequency, Beat Period, and Auto Calculate (PSS)

Determines whether the PSS analysis uses beat frequency or beat period and supplies an initial value.


- Highlight either *Beat Frequency* or *Beat Period*.
- The *Beat Frequency* (or *Beat Period*) field is initially empty.
- Enter a *Beat Frequency* value in one of two ways:
  - Type a frequency value for which all the tone frequencies are integer multiples of the value.
  - Select *Auto Calculate* to automatically calculate a value. The field disappears.

    The *Beat Frequency* value is calculated based on the tones present in the Fundamental Tones list box. It is the greatest common multiple of all the tone frequencies that are not small-signals.

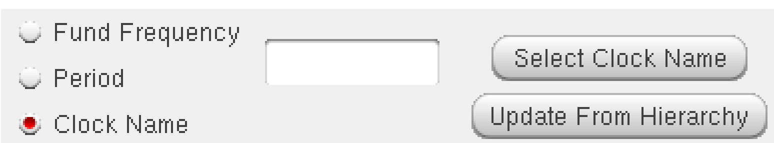
- Enter a *Beat Period* value in one of two ways:
  - Type a period value for which all the tone frequencies are integer multiples of the inverse of the value.
  - Click the *Auto Calculate* button to automatically calculate a value. The field disappears.

    The *Beat Period* value is calculated based on the tones present in the Fundamental Tones list box. The *Beat Period* value is set to the inverse of the frequency.

  The *Beat Frequency* (or *Period*) value is displayed in a read-only field at the top of the periodic small-signal analysis forms.

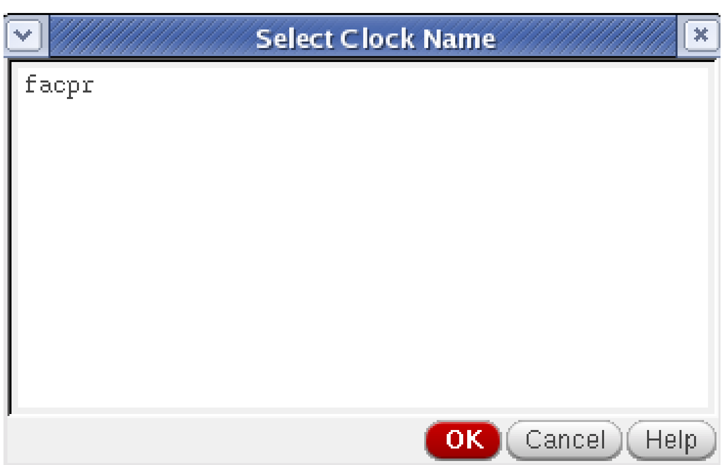
### Clock Name and Select Clock Name Button (ENVLP)

Identifies the clock signal for an ENVLP analysis. The simulator automatically determines the clock period by looking through all the sources with the specified name.




Enter a clock signal name in the *Clock Name* field in one of two ways:

- By typing the clock signal name in the *Clock Name* field.
- By clicking the *Select Clock Name* button to display a list of clock signals.



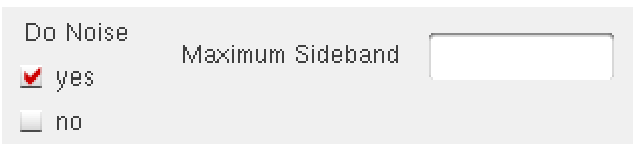

**a.** Click to highlight a clock signal from the list.

**b.** Click *OK* to select the signal and display it in the *Clock Name* field.

See “Stop Time (ENVLP)” on page 74 for related information.

## Do Noise (PSP, QPSP)

Performs noise measurements during the PSP or QPSP analysis.




Highlight *yes* for *Do Noise* to compute noise figure, equivalent noise sources, and noise parameters during the analysis.

To include relevant noise folding effects, specify a maximum sideband value in the *MaximumSideband* field. A sideband array of the form

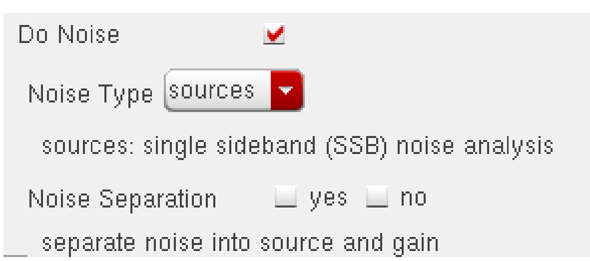
```
[-max. sideband . . . 0 . . . + max. sideband]
```

is automatically generated.

## Do Noise (HBNOISE)

Calculates transfer functions and performs noise measurements during the HBnoise analysis.

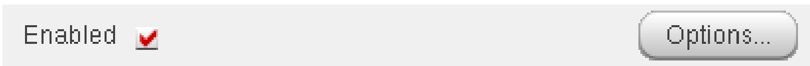
The only supported noise type for this release is *sources*.




Highlight *yes* for *Noise Separation* to calculate the contributions that noise sources make to the output. In addition to the noise contributions, the simulator also determines the transfer functions. When you highlight *no* for *Noise Separation*, the transfer functions are determined but the noise contributions are not.

## Enabled

Includes the analysis in the next simulation.




Select *Enabled* to perform the analysis in the next simulation. Enabled analyses are listed in the ADE window.

Click *Options* to display the Options Form for that analysis.

## Engine (ENVLP, PSS, QPSS)

Specifies the method to be used for the simulation.


**Shooting** engine combined with a time-varying small-signal analyses is efficient for circuits that respond in a strongly nonlinear manner to the LO or the clock. Consequently, you can use the Spectre RF simulations with the shooting engine to simulate strongly nonlinear circuits, such as switched-capacitor filters, switching mixers, chopper-stabilized amplifiers, PLL-based frequency multipliers, sample-and-holds, and samplers.

**Harmonic Balance** engine supports frequency domain harmonic balance analyses. It provides efficient and robust simulation for linear and weakly nonlinear circuits.

**Multi Carrier** engine extends single-carrier HB envelope analysis to better handle cases such as down-conversion mixers and circuits with multi-carriers. This choice appears only for the ENVLP analysis.

## Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)

Defines the analysis sweep range, sweep type, and any additional sweep points for a small-signal analysis. The PAC, Pnoise, PXF, PSP, and PSTB periodic small-signal analyses follow a PSS large-signal analysis. The QPAC, QPnoise, QPXF, and QPSP quasi-periodic small-signal analyses follow a QPSS analysis.

### Frequency Sweep Range (Hz)

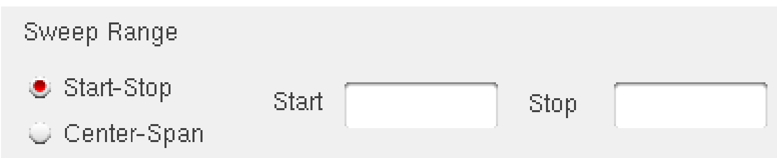
Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

For small-signal analyses following a swept PSS or QPSS analysis, *Single-Point* and *Freq* are the only *Frequency Sweep Range (Hz)* options for the small-signal analyses.

When you make a selection from the *Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

#### *Start - Stop*

Defines the beginning and ending points for the sweep.



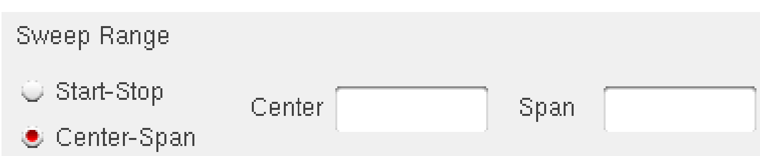

1. Highlight *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

#### *Center - Span*

Defines the center point for the sweep and its span.



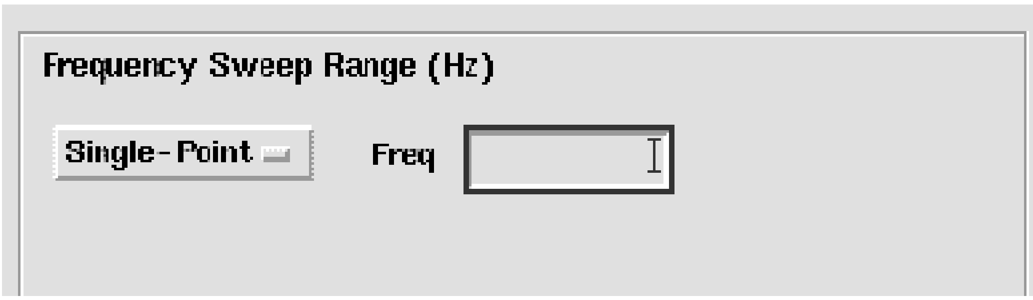

1. Highlight *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

### *Single - Point and Freq*

Defines the frequency range as a single point and prompts you for the point value.



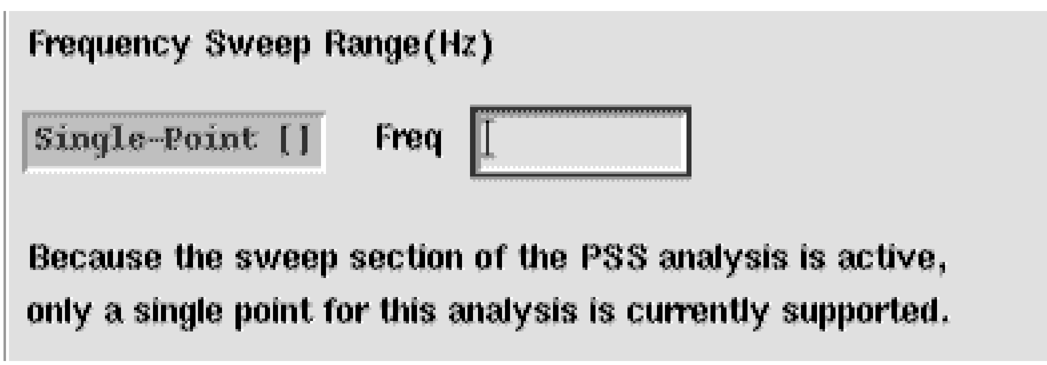

1. Highlight *Single-Point*.

   The form changes to let you type the frequency.

2. Type the specific frequency for the small-signal analysis.

### *Single - Point and Freq Following Swept PSS or Swept QPSS Analysis*

When small-signal analyses follows a swept PSS or QPSS analysis, the only *Frequency Sweep Range (Hz)* option for the small-signal analyses is *Single-Point* and *Freq*.




➤ In the *Freq* field, type the specific frequency for the small-signal analysis that follows each pass of a PSS or QPSS sweep.

### Sweep Type

Specifies whether the small-signal sweep is linear, logarithmic, or automatic.

### *Linear*

Specifies a linear sweep.


1. Select *Linear*.

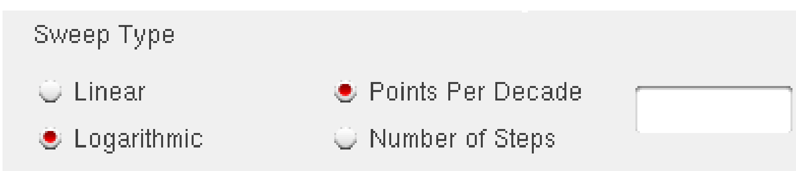
   The form changes to let you type either the step size or the total number of points (steps).

2. Do one of the following:

   - Highlight *Step Size* and type the size of each step in the field.
   - Highlight *Number of Steps* and type the number of steps in the field.

### *Logarithmic*

Specifies a logarithmic sweep.




1. Select *Logarithmic*.

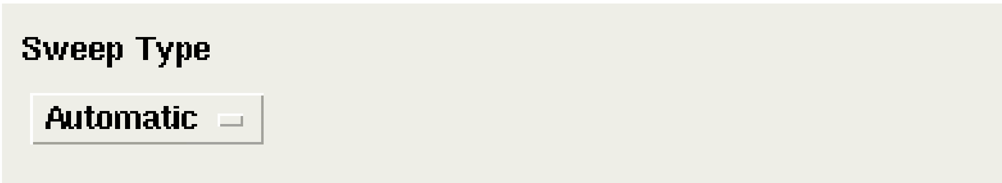
   The form changes to let you type either the number of points per decade or the total number of points (steps).

2. Do one of the following:

   - Highlight *Points per Decade* and type the number of points per decade in the field.
   - Highlight *Number of Steps* and type the number of steps in the field.

### *Automatic*

Lets the simulator determine whether the *Sweep Type* is *Linear* or *Logarithmic*.




➤ Select *Automatic*.

The *Sweep Type* is *Linear* if the ratio of *Start* to *Stop* values is less than 10 or *Logarithmic* if the ratio of *Start* to *Stop* values is10 or higher.

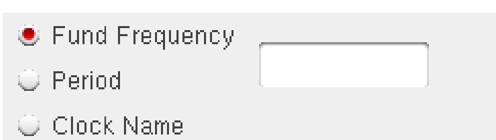
### Add Specific Points

Specifies additional sweep points for the small-signal analysis.


Highlight *Add Specific Points* and type the additional sweep point values into the field. Separate them with spaces.

## Fund Frequency (ENVLP)

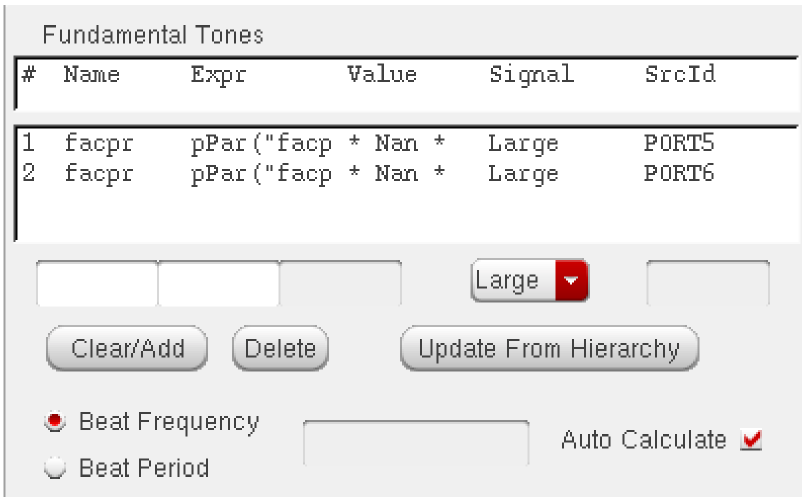
Specifies the frequency of the clock fundamental.




Type the frequency into the field.

## Fundamental Tones (PSS, QPSS)

The *Fundamental Tones* fields include the following list box, data entry fields, and data entry buttons. In addition, the PSS analysis includes the related *Beat Frequency*, *Beat Period*, and *Auto Calculate* buttons.




### Fundamental Tones List Box

The *Fundamental Tones* list box displays information about every top-level tone in the circuit that has both a non-zero frequency or period value, and a non-zero amplitude value (absolute). The tones in the list box are arranged alphabetically by name.

For QPSS analyses, you can edit

- The *Signal* level designation
- The *Harms* (Harmonic Range) value

To edit values for a tone, highlight the tone in the list box then edit in the data entry fields.

For tones that are not at the top level of the schematic, you can manually create a tone entry by typing the pertinent information in the data entry fields.

For non-small-signal tones, each tone name and its correlated frequency value are used in the *Select from range* and *Array of coefficients* fields in the *Output Harmonics* and *Sidebands* sections of the PSS, PAC, Pnoise, PXF, and QPnoise small-signal analysis forms.

### Fundamental Tones List Box Terms and Data Entry Fields

- **Name** – Displays the name assigned to the tone. This tone name must be entered into the pertinent Component Description Format (CDF) fields of each source in the schematic with a tone. The current CDF name field prompts are “First frequency name”, “Second frequency name”, “Frequency name”, and “Frequency name for 1/period”.

- **Expr** – Displays the value or expression representing the frequency of a particular tone. The expression can also be a user variable or it can contain user variables. If the frequency for the tone is specified as a variable, the *Expr* field displays the name of the variable. Otherwise, the field displays the numerical value of the frequency.

- **Value** – Displays the evaluated value of the *Expr* field using the current values of the user variables.

- **Signal** – This cyclic field displays one of two values: *Large* or *Moderate*.

  For QPSS analysis, you must select one *Large* tone. This is the only time you use the *Large* specification. Specify *Moderate* for all additional tones you want to include in the simulation.

  For PSS analysis, *Moderate* appears for all tones in the simulation.

- **SrcId** – Displays the instance name of the source in the schematic where the tone is declared.

- **Harms** – Used *only* with QPSS analysis.

  Specifies the range of harmonics, which in turn determines the maximum number of harmonics of the tone to be used during the simulation.

  The *Harms* value must be 1 or higher for the tone to be included in the simulation. The default value is 1.

  Cannot deal with AHDL sources unless they are done with inlined Spectre sources.

### Setting Up Tones for a QPSS Analysis

When you set up tones for a QPSS analysis,

- Designate one signal as the *Large* tone. Designate all other tones as *Moderate*.

  By choosing *Large* as the *Signal* value for a tone, you specify that tone to be the *Large* or clock signal. Each QPSS analysis must have one tone set to be the *Large* signal.

  Select your *Large* signal to be the signal that

  - ❑ Causes the largest response in the circuit.

- ❑ Is the least sinusoidal signal in the circuit.
- ❑ Causes the most nonlinearity in the circuit.

■ Designate a non-zero harmonic range (*Harms*) value for each tone.

Never set the harmonic range value to zero. If you do, the simulation does not run properly and you might get incorrect results.

Choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with an harmonic range (*Harms*) value of 0 is ignored by the simulation. In general, when selecting a *Harms* value:

For the *Large* signal, in most cases, set the *Harms* value to be equal to or greater than 5. An harmonic range of 5 gives 11 harmonics (-5, ..., 0, ... +5) for the *Large* tone.

*Important*

Be aware of the following information about the *Harms* value.

- ❑ The *Harms* value you use for the *Large* signal varies with the circuit you are analyzing. For example, for a down converting mixer, a *Harms* value of 1 is sufficient. Entering a large *Harms* value for the large tone does not affect the simulation run time.
- ❑ For *Moderate* tones, set the *Harms* value to be approximately 2 or 3. Setting the *Harms* value to 2 gives you up to the 3rd order intermodulation terms. Setting the *Harms* value to 3 gives you up to the 5th order intermodulation terms.

  To obtain higher order intermodulation terms, you can increase the *Harms* value accordingly. However, for *Moderate* tones, increasing the *Harms* value increases the simulation run time. For example, when you specify *Harms* values of 5 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [5,2]` which gives you 11 harmonics for the *Large* tone and 5 harmonics (-2, -1, 0, 1, 2) for the *Moderate* tone. As a result, you get noise from 11 x 5 = 55 frequency sidebands.
- ❑ The *Harms* value you select depends on the degree of nonlinearity the signal causes. If the signal is not nonlinear, then a *Harms* value of 1 is sufficient. For a moderate signal, a few harmonics should be sufficient to accurately capture the nonlinearity. However, it is hard to determine before running the analysis what is sufficient. Generally, for a given *Harms* value, if increasing it does not significantly change the spectrum results, then the *Harms* value is high enough. In some situations, you could use a *Harms* value as high as 9. For a high *Harms* value, the algorithm still works, but not as efficiently.

### Fundamental Tones Data Entry Buttons

- **Clear/Add** – Clears the data entry fields for the purpose of manually adding a new tone to the list box. This button also resets the list box so that no line is currently selected.
- **Delete** – Deletes a tone selected in the *Fundamental Tones* list box.

  You cannot use the *Delete* button to delete tones that are specified in the schematic. Such tones are deleted by setting the CDF frequency or period field value to zero or blank and CDF amplitude value(s) to zero (absolute) or blank.
- **Update From Hierarchy** – Updates the values in the *Fundamental Tones* list box by searching all the levels of the hierarchy. If you do not click *Update From Hierarchy*, only the top level of the hierarchy is searched.

### Using the Fundamental Tones Data Entry Fields and Buttons

Use the data entry fields below the list box to edit the information in the *Fundamental Tones* list box or to add new tones.

To specify a new tone,

1. Make sure there are no selected tones in the list box. If a there is a selected tone, click the *Clear/Add* button.
2. Type a value in any of the data entry fields (typically starting with the *Name* field).
3. As you advance to a second data entry field, a new tone line is added to the list box and is selected.
4. If required, select a value from the *Signal* cyclic field.
5. Continue editing each data entry field until all the pertinent information is complete.
6. The last value in the data entry fields is recorded in the list box when the next operation is performed in the analysis form or when you click the *Clear/Add* button. (Other operations that terminate the editing operation include moving the cursor off the Choosing Analyses form, clicking either the *OK* or *Apply* button, or changing a value in a non-related field.)

To edit an existing tone,

1. Select the tone in the list box.
2. The tone data appears in the data entry fields where you can edit it.

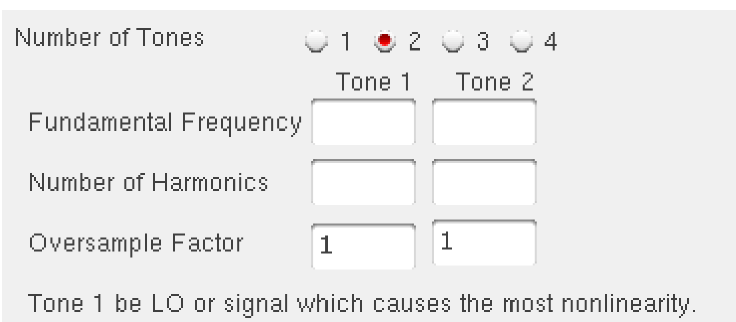
3. Modify a value in any one of the data entry fields. Values you cannot edit in the data entry fields (such as values specified in the schematic), are grayed out. Values originally specified in the schematic must be edited in the schematic.

4. The last modified value in the data entry fields is recorded in the list box when the next operation is performed in the analysis form or when you click the *Clear/Add* button.

## Tones (HB)

The Tones selection field provides two options for users to define the fundamental frequencies used by the HB analysis. Frequencies is selected by default. However, if you select the Names option, the same procedure must be used as explained in the Fundamental Tones (PSS, QPSS) section on page 48.


If Frequencies is selected, you can directly input the fundamental frequencies to be used by HB simulation.




Up to four independent tones (fundamental frequencies) can be specified in the analysis form. For each of the specified tone, you can provide values for the frequency, number of harmonics and oversample value in the fields along side. To set harmonics consider the expected waveshape. If the waveform is more square wave, start with about 15 harmonics and an oversample factor of 2 If the waveform in nearly sinusoidal start with 5 harmonics and oversample=1 Re-run the simulkation with about 5% more harmonics and the next higher oversample number. If the results change, you need more harmonics and/or a higher oversample number. In the testbench, the same value of frequencies or the linear combination of those fundamental frequencies can be defined in the port setup.

The Frequencies selection is recommended for composite triple beat (CTB) simulation. If the frequency is specified after divider as one of the tones, the freqdivide option is not required to be specified in the Harmonic Balance Options form.

## Harmonic Balance Homotopy Method (HB)

Determines the method used to pursue convergence in a circuit.


The *Harmonic Balance Homotopy Method* choices are:

- *tstab* – Runs a transient analysis to determine initial conditions for the circuit. This value is appropriate when the circuit contains devices, such as digital frequency dividers, that display strong non-linear behavior.
- *source* – Determines the initial conditions by decreasing the nonlinearity of the circuit and then gradually stepping up the nonlinearity to its original value. Each time the nonlinearity is stepped, the solution from the previous iteration is used as the starting condition. This value is appropriate when the power level of the large tone or of the RF tones is high.

  The *source* value must not be used for autonomous circuits.
- *tone* – First generates a single tone solution by turning off all the tones except the first one. The multi-tone circuit is then solved by restoring all the tones and using the single-tone solution as the initial guess. This value is appropriate for multitone designs with a strong first tone.
- *gsweep* – Determines the initial conditions by reducing and then incrementing the dynamic range of the circuit. The *gsweep* value is appropriate for situations where convergence fails for both the *source* and *tstab* values.

## Harmonics (QPSS)

The *Harmonics* area enables you to select the important moderate tone harmonics for QPSS multitone simulation.This makes the QPSS analysis more efficient.

Choices for *Harmonics* are: *Default*, *View All*, and *Select*.

### Default

Select harmonics by choosing *auto*, *even*, *odd*, or *all* in the *Harm selection for each moderate tone* field. The *Harm selection for each moderate tone* field only appears when *Engine* is set to *Shooting*.

### View All


Lets you first enter a frequency range and then select harmonics from within this range of harmonic values.

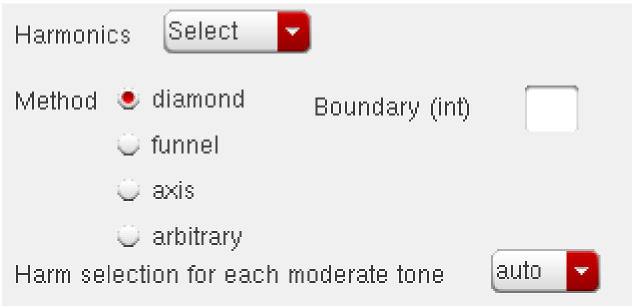
1. In the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range.

   The form changes to display all input frequencies, their harmonics, and the intermodulations of the frequencies and harmonics for named, top-level, large and moderate input signals in the circuit whose frequencies fall within the specified frequency range. The first column is the frequency, and the remaining columns specify tone coefficients for each fundamental tone that contributed to the listed harmonic.

2. Select from the listed harmonics.

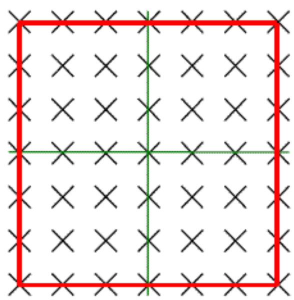
   Click a harmonic in the list box to select it. Select adjacent harmonics by clicking and dragging with the mouse over the harmonics to select. Select harmonics that are not adjacent by holding the `Control` key down while you click the individual harmonics. (Deselect a harmonic by clicking a selected harmonic while you hold the `Control` key down.)

### Select

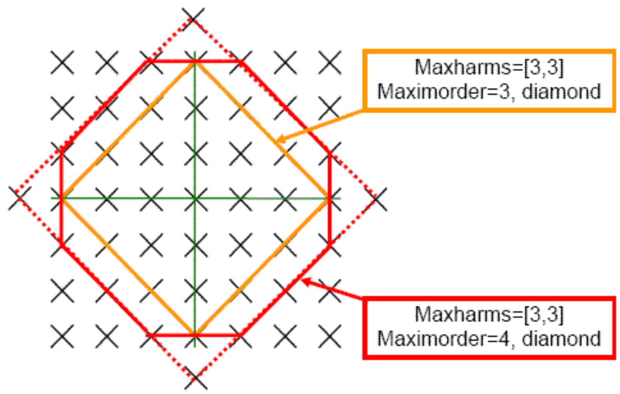



**Method** specifies how the harmonics are selected: *diamond*, *funnel*, *axis*, or *arbitrary*. The harmonics selected by each method depend on the `maxharms` and *MaxImOrder* or *Boundary* values.

- If the *Harmonics* value is something other than *Select*, the *Method* field does not appear in the form and the box selection method is the default.

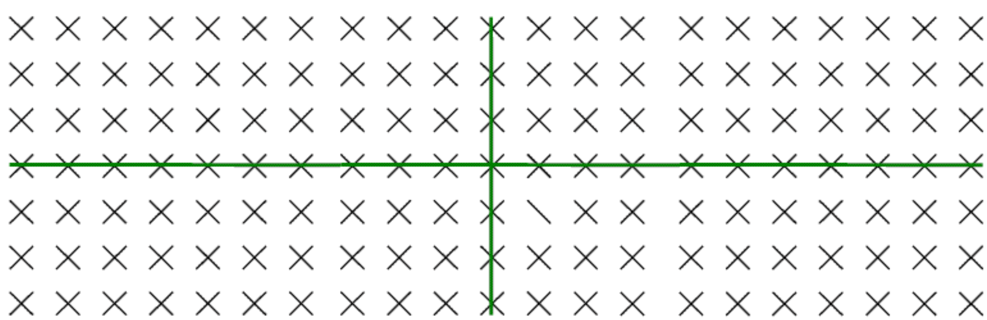


- The diamond method is designed to keep the important on-axis harmonics but to reduce the number of off-axis (intermodulation) harmonics. The evaluated harmonics consist of the intersection of the universe of harmonics (specified by the `maxharms` values) and of the superimposed diamond (whose order or size is specified by either the *MaxImOrder* or *Boundary* values). In the following illustration, for example, the evaluated harmonics lie on or within either the solid red line or the solid orange line.

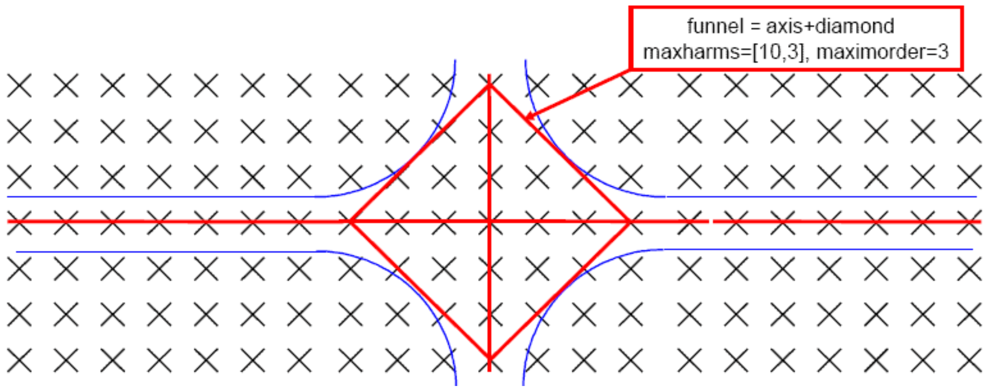



**Note:** The diamond method must not be used in a QPSS analysis that is followed by a QPAC, QPXD, QPNOISE, or QPSP analysis because some sidebands are not included in the small signal calculation.

- The *axis* method selects harmonics only along the axes; that is, there are no intermodulation harmonics. In the following illustration, for example, the evaluated harmonics include only those indicated by the green lines.



- The *funnel* method is a combination of the *diamond* method and *axis* methods. It selects harmonics along the axes and intermodulation harmonics of a specified order or less.




- The *arbitrary* method allows you to specify indices of interest for each of the moderate fundamental tones. Guidance on the format to use appears in a message when you select this method.

**Boundary (int)** specifies the harmonic selection boundary.

**Harm selection for each moderate tone**

When *Engine* is set to *Harmonic Balance*, the *Harm selection for each moderate tone* and *Boundary (int)* options are replaced by an option called *MaxImOrder*.

**MaxImOrder** specifies the maximum intermodulation order. This value interacts with the chosen *Method* to determine which harmonics are selected.

## Input Frequency Sweep Range (HBAC)

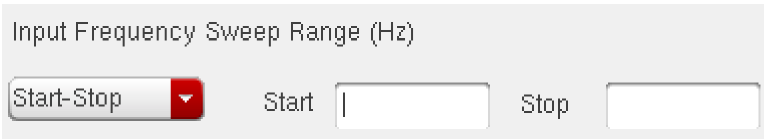
Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

For small-signal analyses following a swept PSS or QPSS analysis, *Single-Point* and *Freq* are the only *Frequency Sweep Range (Hz)* options for the small-signal analyses.

When you make a selection from the *Input Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

### *Start - Stop*

Defines the beginning and ending points for the sweep.



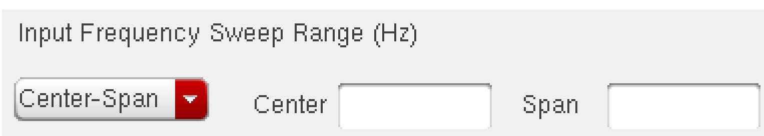

1. Select *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

### *Center - Span*

Defines the center point for the sweep and its span.




1. Select *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

### *Single - Point and Freq*

Defines the frequency range as a single point and prompts you for the point value.


1. Select *Single-Point*.

   The form changes to let you type the frequency.

2. Type the specific frequency for the small-signal analysis.

## Input Source and Reference Side-Band (Pnoise, HBnoise)

Identifies the noise generator and the reference sidebands to use for the Pnoise simulation.

The Reference Side-Band field specifies which conversion gain to use when the Spectre RF simulation computes the input-referred noise, noise factor, and noise figure.

The designated reference sidebands, as well as the sideband zero, are included in the pool of sidebands used in noise calculations. For example, for the *ne600* test schematic,

- If `refsideband=-1` and `sidebands=[-2]`, then `Pnoise` computes contributions from sidebands -2, -1, and 0.
- If `refsideband=-2` and `sidebands=[-2]`, then Spectre RF computes contributions from sidebands -2 and 0.
- See "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 57 for information on selecting sidebands.

The output total noise is different for the two simulation setups. The input-referred noise, noise factor, and noise figure are also different. Pnoise analysis internally includes the refsideband as a contribution to the total noise. This inclusion is not reflected in the netlist.

### Input Source

Choices for Input Source are: *voltage*, *current*, *port*, or *none*.

#### *Voltage*


1. Select *voltage* from the *Input Source* cyclic field.

   The *Reference side-band* fields appear.

2. Either type the name of the noise voltage generator in the *Input Voltage Source* field or click *Select* and then click the generator in the schematic.

#### *Current*


1. Select *current* from the *Input Source* cyclic field.

   The *Reference side-band* fields appear.

2. Either type the name of a noise current generator in the *Input Current Source* field or click *Select* and then click the generator in the schematic.

#### *Port*


1. Select *port* from the *Input Source* cyclic field.

   The *Reference side-band* fields appear.

2. Either type the name of a port in the *Input Port Source* field or click *Select* and then click the port in the schematic.

When you select *port*, the analysis computes the noise voltage across the port and subtracts the contribution of this port in noise figure calculations.

#### *None*


➤ Select *none* from the *Input Source* cyclic field.

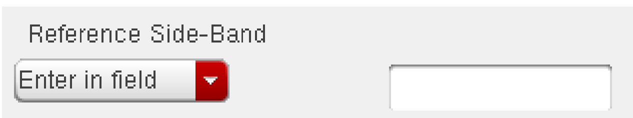
When you select none, there are certain calculations you cannot perform. For example, input referred noise.

### Reference Side-Band

Reference Side-Band specifies which conversion gain to use when the Spectre RF simulation computes the input-referred noise, noise factor, and noise figure.

Choices are *Enter in field* or *Select from list*.

#### *Enter in Field*

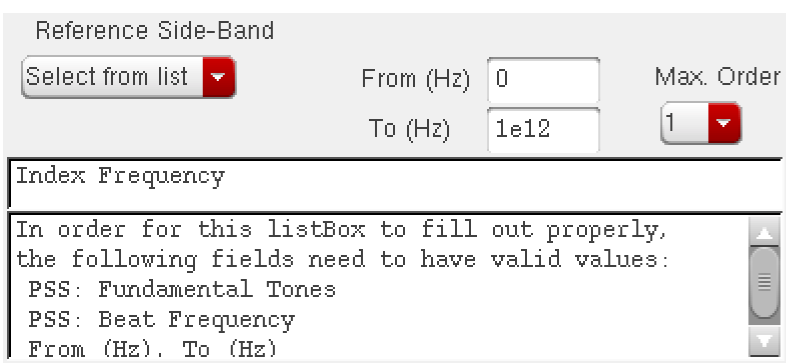



- Select *Enter in field* from the *Reference side-band* cyclic field.
- Type an integer value in the field.
  - `0` for amplifiers and filters
  - `-K` for down converters
  - `+K` for up converters

  where `K` is the mixing harmonic.

#### *Select From List*

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.




Depending on the selection in the ***Sweeptype*** cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands,

1. In the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range.

   The sideband frequencies displayed in the list box are within this range of frequencies.

2. From the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands.

   If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

To select from the listed sidebands,

➤ Click a sideband in the list box to select it.

  Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the `Control` key down while you click the individual sidebands. (Deselect a sideband by clicking a selected sideband while you hold the `Control` key down.)

## Input Source and Reference Side-Band (QPnoise)

Identifies the noise generator and the reference sidebands vector for the QPnoise simulation.

### Input Source

Choices for *Input Source* are: *voltage, current, port,* or *none*.

#### *Voltage*


1. Select *voltage* from the *Input Source* cyclic field.

2. Either type the name of the noise voltage generator in the *Input Voltage Source* field or click *Select* and then click the generator in the schematic.

#### *Current*


1. Select *current* from the *Input Source* cyclic field.
2. Either type the name of a noise current generator in the *Input Current Source* field or click *Select* and then click the generator in the schematic.

#### *Port*


1. Select *port* from the *Input Source* cyclic field.
2. Either type the name of a port in the *Input Port Source* field or click *Select* and then click the port in the schematic.

#### *None*


➤ Select *none* from the *Input Source* cyclic field.

When you select none, there are certain calculations you cannot perform. For example, input referred noise. *Reference Side-Band* is not available when you select *none* for *Input Source.*

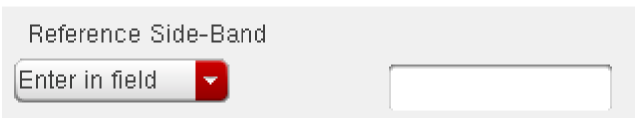
### Reference Side-Band

*Reference side-band* specifies which conversion gain to use when the Spectre RF simulation computes the input-referred noise, noise factor, and noise figure. *Reference side-band* is available with all *Input Source* choices except *none.*

For QPnoise analysis, the *Reference Side-Band* field value is a vector. For example, `1 0 0`.

Choices are *Enter in field* or *Select from list*.

#### *Enter in Field*




Select *Enter in field* from the *Reference side-band* cyclic field and type a vector value into the field.

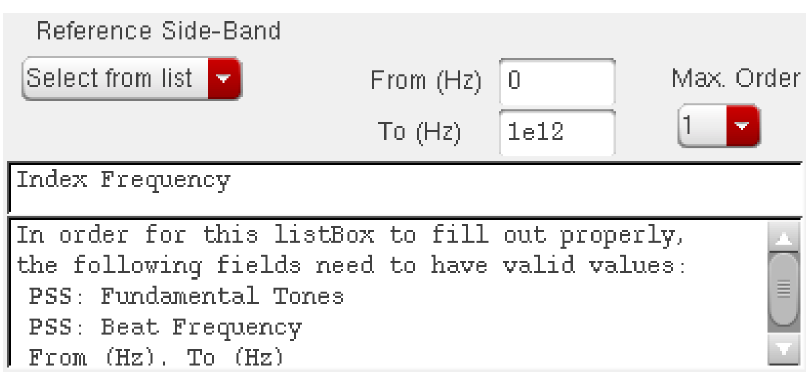
When the input and output are at the same frequency, use the zero vector

```
[0 0 ...]
```

- When you do not use the zero vector, the single sideband noise figure is computed.

#### *Select From List*

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.




Depending on the selection in the *Sweeptype* cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands,

1. First, in the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range.

   The sideband frequencies displayed in the list box are within this range of frequencies.

2. Then, from the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands.

   If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

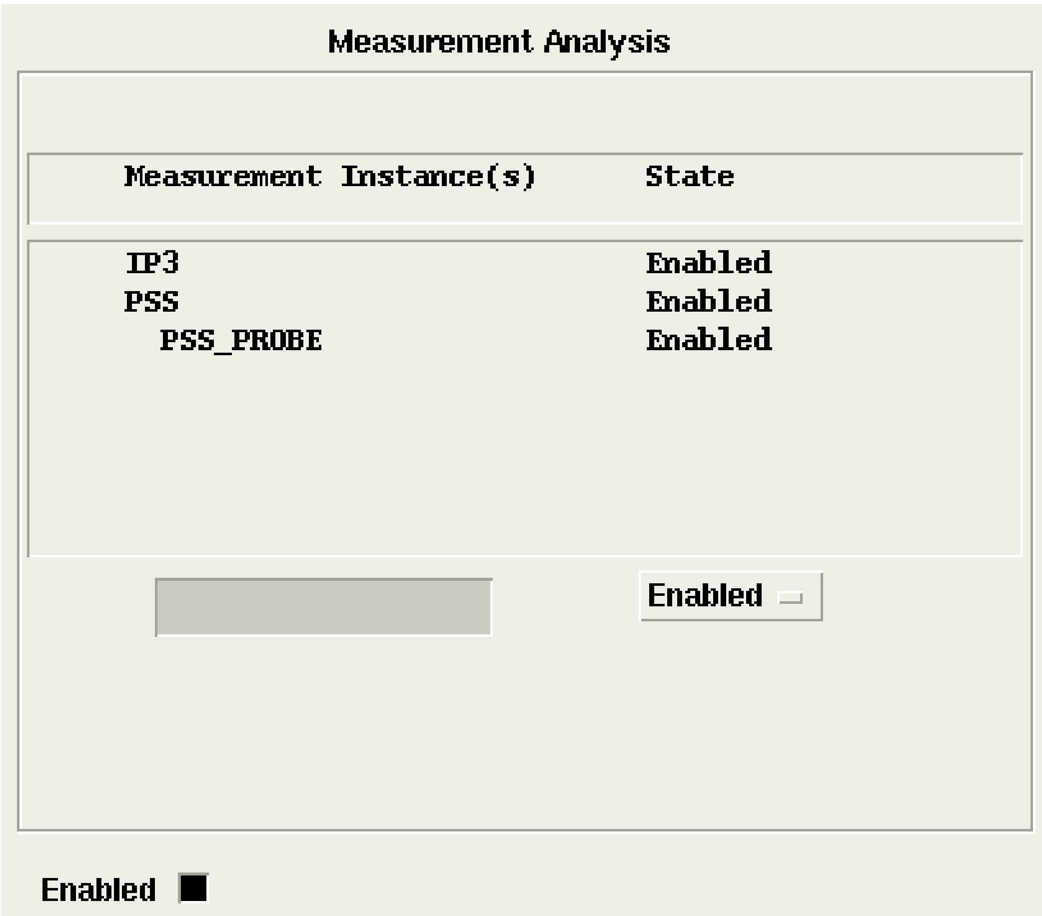
To select from the listed sidebands,

➤ Click a sideband in the list box to select it.

  Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the `Control` key down while you click the individual sidebands. (Deselect a sideband by clicking a selected sideband while you hold the `Control` key down.)

## Measurement Analysis (measure)

Controls which instantiated analyses run during simulation. The items in this list are defined in measurement and analysis cells, such as the components of the `measureLib` library, that are instantiated in the schematic.

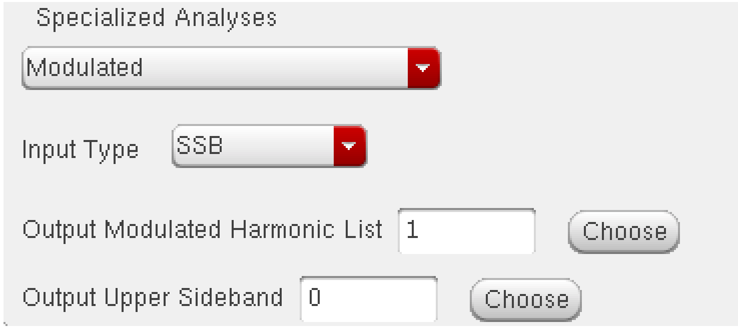



To turn a measurement on or off, highlight the measurement and then choose the desired setting with the *Enabled* drop-down menu.

## Modulated Analysis (PAC, PXF)

Measures AM and PM small-signal effects for the PAC and PXF analyses. When you select *Modulated* in the *Specialized Analyses* field, the Choosing Analyses form changes to let you enter more information.

The *Modulated Analysis* fields for the PXF analysis include the following.

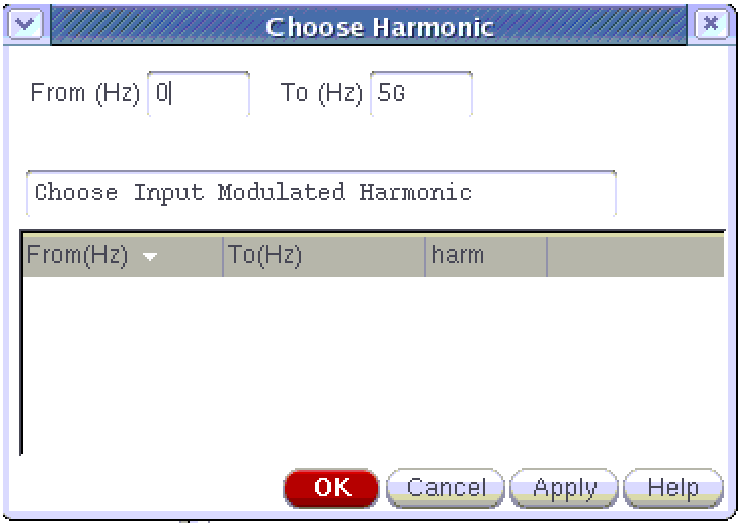



### Modulated Analysis for PXF Terms and Data Entry Fields

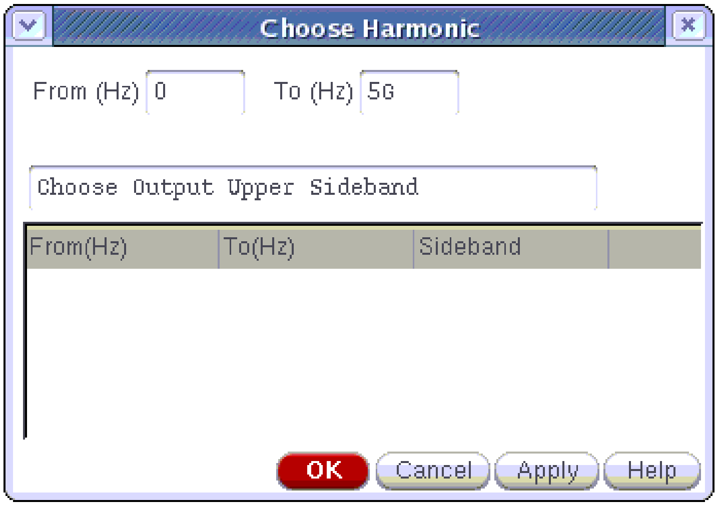
- **Output Type** – This cyclic field displays one of two values: *SSB* and *SSB/AM/PM.*
- **Input Modulated Harmonic List** – Lists the harmonic indexes.
- **Input Upper Sideband** – (Displays when you choose *SSB* for *Output Type*.) Click *Choose* to display the Choose Harmonic pop up.
- **Output Modulated Harmonic** – (Displays when you choose the *Output Type SSB/AM/PM*.) Click *Choose* to display the Choose Harmonic pop up.

### Choose Harmonic Pop Up

Selects harmonics or sidebands for the analysis. Display the Choose Harmonic pop up from *Modulated Analysis* with the *Choose* button.



or




#### *Choose Output Modulated Harmonic List Box and Data Entry Fields*

The *Choose Output Modulated Harmonic* list box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To (Hz)* fields, changes the harmonic indexes and frequency ranges displayed in the list box.

**From (Hz) and To (Hz) fields -** The upper and lower bounds for the frequency range.

**Harm** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency.

For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.

**From (Hz) -** The lower bound for the frequency range associated with the harmonic.

**To (Hz) -** The upper bound for the frequency range associated with the harmonic.

#### *Choose Input Upper Sideband List Box and Data Entry Fields*

The *Choose Input Upper Sideband* list box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To (Hz)* fields, changes the harmonic indexes and frequency ranges displayed in the list box.

**From (Hz) and To (Hz) fields -** The upper and lower bounds for the frequency range.

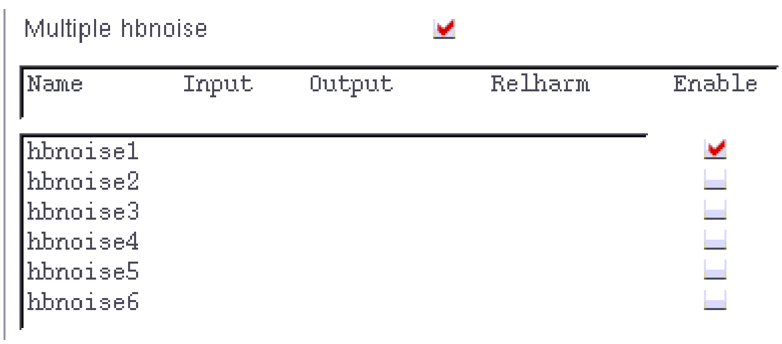
**Sideband** – Displays the sidebands, the integers which are the periodic small-signal output frequencies of interest.

**From (Hz) -** The lower bound for the frequency range associated with the sideband.

**To (Hz) -** The upper bound for the frequency range associated with the sideband.

#### Multiple hbnoise

##### *Enable the multiple hbnoise simulation in one ADE session*




When the selection is enabled, the pop up window allows the user to specify up to four hbnoise simulations. By default, the first simulation is highlighted. The values of Output, Input Source and Relative Harmonic are updated in the simulation summary field. To add one more hbnoise simulation, move the curser to highlight the hbnoise name, for example, hbnoise2. Each hbnoise simulation has its own *Enable* button to allow the user to enable or disable the simulation.

**Note:** The above description is also applicable to pnoise and qpnoise.

### New Initial Value For Each Point (HB)

Specifies whether to restart the DC/PSS solution from scratch or to reuse the previous solution as the initial guess.

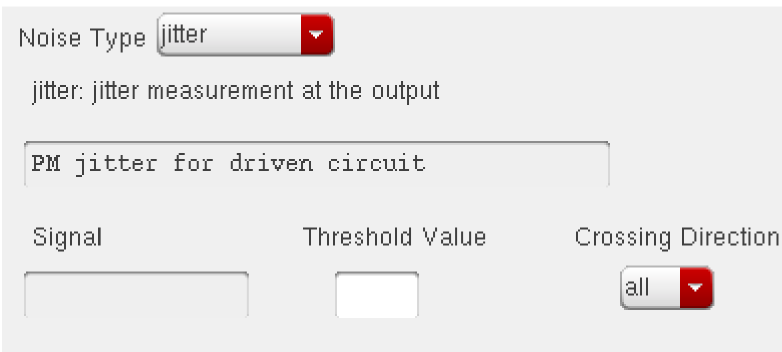
.

## Noise Type (Pnoise)

Specifies the type of noise to compute. Choices are: *jitter, modulated, sources*, and *timedomain*.

### Jitter (for a driven circuit)

Measures PM jitter at the output for a driven circuit.




1. Select *jitter* from the *Noise Type* cyclic field.

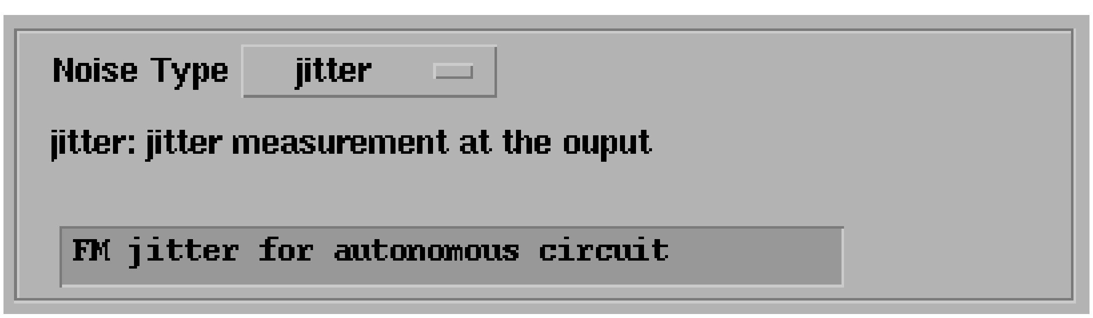
   The *Signal* field displays the signal to measure.

2. In the *Threshold Value* field, type the value where PM jitter is measured when the signal crosses this value.

3. From the *Crossing Direction* cyclic field, select the transition where jitter is measured.

### Jitter (for an autonomous circuit)

Measures FM jitter at the output for an autonomous circuit.



### Modulated

Separates noise into AM and PM components. It also calculates USB (upper sidebands) and LSB (lower sidebands) noise.

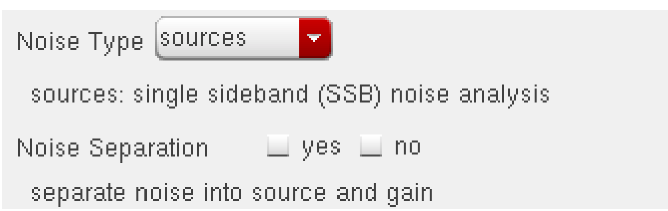
1. Select *modulated* from the *Noise Type* cyclic field.

   The following message displays:

   ```
   When Noise Type is “modulated,“ Sweeptype must be “relative.“
   ```

2. Choose *Close* to dismiss the message box and select relative for Sweeptype. See “Sweeptype (Pnoise)” on page 82 for more information.


### Sources




1. Select *sources* from the *Noise Type* cyclic field.

   When you select *sources,* the simulation computes the total time-average noise at an output over a given frequency range. It computes the contribution of each noise source to the total noise at each frequency.

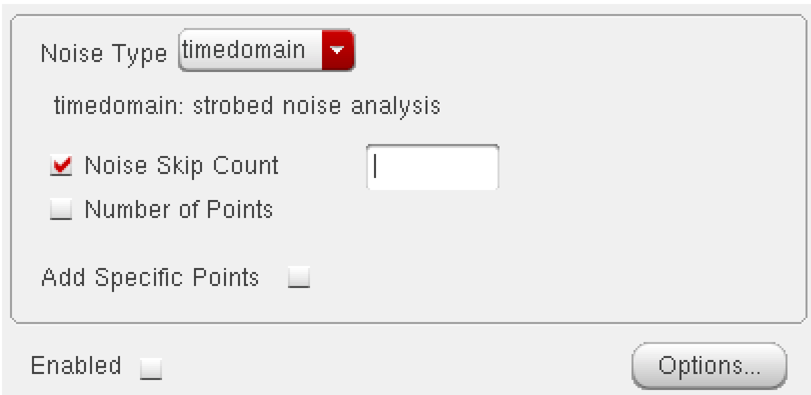
2. Choose whether *Noise Separation* runs or not.

   This option determines whether the noise separation feature runs during the simulation. The possible values are *yes* and *no*.

| | |
|---|---|
| *yes* | Noise separation runs and the results are saved. |
| *no* | Noise separation does not run. |

### Timedomain

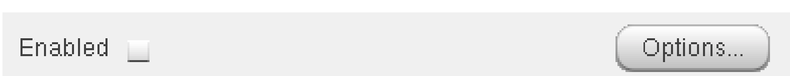
Computes the time-varying instantaneous noise power in a circuit with periodically driven components.




1. Select *timedomain* from the *Noise Type* cyclic field.
2. Type the noise skip count in the *Noise Skip Count* field.
3. (Optional) Highlight *Number of Points* and type the number of points in the *Number of points* field.
4. (Optional) Highlight *Add Specific Points* and type one or more points.

## Options

Opens the Options form.

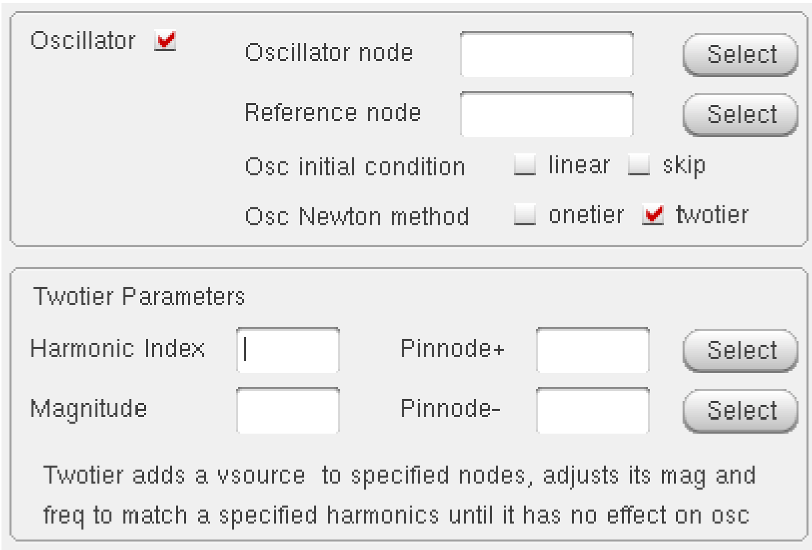



➤ Click *Options* to open the Options form for the selected analysis.

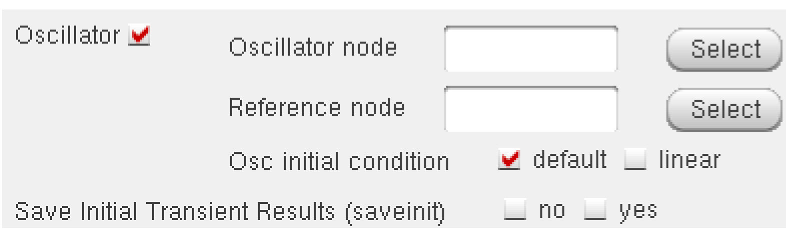
See "Options Forms" on page 165 for information on the Options forms.

## Oscillator (PSS, ENVLP, HB)

Specifies that the circuit is an oscillator. When you highlight *Oscillator*, the form changes to let you enter additional information. For PSS and HB, the Oscillator pane looks like this.




For ENVLP, the Oscillator pane looks like this.




➤ Either

- ❑ Type the *Oscillator node* and *Reference node* names.
- ❑ Click the corresponding *Select* and then click the node in the schematic.

If you leave the *Reference node* field empty, the name defaults to `gnd`.

The *OSC initial condition* option determines how the starting values for the oscillator are determined. The possible values are *default* and *linear*.

| | |
|---|---|
| *default* | This is the default value. A transient analysis controlled by the `tstab` setting is used to generate the initial guess of the solution. |

| | |
|---|---|
| *linear* | A linear analysis at the DC solution is used to estimate the oscillation frequency and the amplitude. Then a transient analysis controlled by the `tstab` setting is performed using the estimated linear solution as a starting point.<br><br>With this approach, the large `tstab` value that would otherwise be required for high-Q oscillators can be reduced or eliminated.<br><br>The *linear* value is suited for linear oscillators such as LC and crystal oscillators. |

The *Osc Newton method* option specifies that only the *onetier* or only the *twotier* method, but not both, are to be used. By default, Spectre RF runs *onetier.*

| | |
|---|---|
| *onetier* | The frequency and voltage spectrum are solved simultaneously in one single set of nonlinear equations. The initial guess of onetier method is generated by running transient analysis or by linear IC, or by the combination of linear IC and transient analysis. |

*twotier* The nonlinear equations are split into two sets: the inner set of nonlinear equations solves the spectrum of node voltage equations; the outer set of nonlinear equations solves the oscillation frequency. To use twotier method, a voltage probe must be added at a circuit node in the oscillator. This probe is used to generate the initial guess and control the two level solving process mentioned above. It is recommended to add this probe at a node on the tank of the oscillator core.

This probe is specified by four parameters:

- Pinnode+ and Pinnode- : These two nodes specify where the voltage probe should be placed. Default setting of Pinnode- is “0” or “gnd”. If differential nodes exist in the oscillator core, Pinnode+ and Pinnode- should be set to the differential ones.
- Magnitude: The user can provide an estimate value of the voltage probe. This will speed up the initial guess generation process. If not provided, a default value will be used.
- Harmonic index: The harmonic index of the voltage probe. If no divider in the oscillator, this parameter should always be set to “1”. Default value is “1”

Twotier method needs a initial guess frequency to start. There are two ways to generate the guess frequency. One way is by using the “linear” method. If linear” is used, SpectreRF will use the linear IC method to get the guess frequency. “linear” is the default setting.

The other way is by using the “skip” method. If “skip” is used, SpectreRF will directly use the user provided frequency as the guess frequency. The user should provide a frequency as close as the true oscillating frequency. For oscillator with dividers, the user is not recommended to use twotier method .

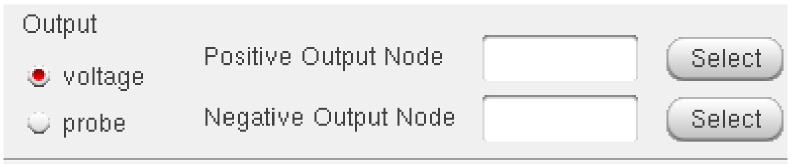
The *Save Initial Transient Results (saveinit)* option saves the initial results of the transient analysis.

## Output (PXF, QPXF)

Specifies the output transfer function for the PXF and QPXF analyses. Choices are *voltage* or *probe*.

### Voltage

In PXF analysis, the voltage across the two nodes that you specify is the output for each transfer function.




1. Highlight *voltage*.

   The form changes to let you specify a *Positive Output Node* and a *Negative Output Node*.

2. Specify the node names by either

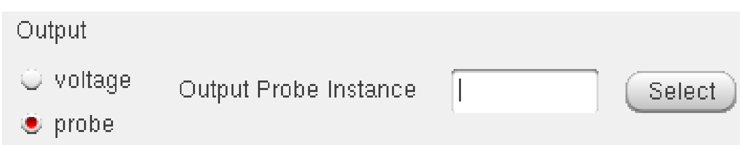
   ❑ Typing the node name into the *Positive Output Node* or *Negative Output Node* fields.

   ❑ Clicking the adjacent *Select* button and then clicking the node in the schematic.

   If you leave the *Negative Output Node* field empty, it defaults to `gnd`.

### Probe

In PXF analysis, the current through the point that you select is the output of each transfer function.




1. Highlight *Probe*.

   The form changes to let you specify an output voltage source.

2. Specify the output voltage source by either

   ❑ Typing the instance name into the *Output Probe Instance* field.

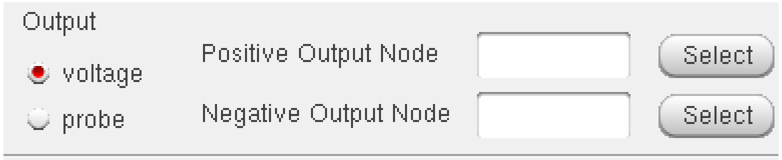
   ❑ Clicking the adjacent *Select* button and then clicking the node in the schematic.

## Output (Pnoise, QPnoise, HBnoise)

The *Output* cyclic field lets you specify the output for the Pnoise, QPnoise, and HBnoise analyses. Choices are *voltage* or *probe*.

### Voltage

The analysis computes the noise voltage across the two nodes.




1. Select *Voltage*.

   The form changes to let you specify a *Positive Output Node* and a *Negative Output Node*.

2. Specify the nodes by either

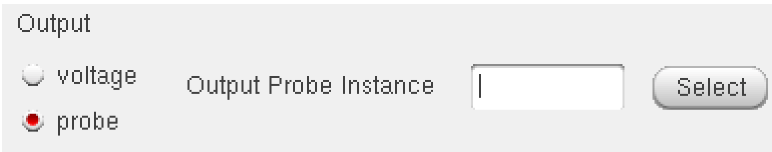
   ❑ Typing a node name into the type-in field.

   ❑ Clicking the adjacent *Select* button and then clicking the appropriate node in the schematic.

   If you leave the *Negative Output Node* field empty, it defaults to gnd.

### Probe

The analysis computes the noise voltage across the port. The noise contribution of the port is subtracted during the noise figure calculation.




1. Select *Probe*.

   The form changes to let you specify the *Output Probe Instance*.

2. Specify the node either by

   ❑ Typing the node name into the type-in field.

- Clicking the adjacent *Select* button and then clicking the appropriate node in the schematic.

## Output Frequency Sweep Range (HBnoise)

Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

For small-signal analyses following a swept PSS or QPSS analysis, *Single-Point* and *Freq* are the only *Frequency Sweep Range (Hz)* options for the small-signal analyses.

When you make a selection from the *Output Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

### *Start - Stop*

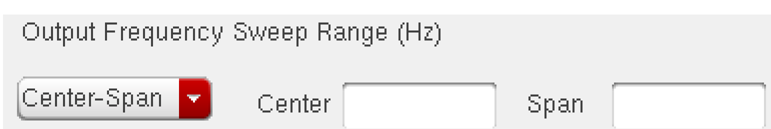
Defines the beginning and ending points for the sweep.


1. Select *Start-Stop*.

   The form changes to let you type the start and stop points.
2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

### *Center - Span*

Defines the center point for the sweep and its span.




1. Select *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

#### *Single - Point and Freq*

Defines the frequency range as a single point and prompts you for the point value.


1. Select *Single-Point*.

   The form changes to let you type the frequency.
2. Type the specific frequency for the small-signal analysis.

## Output Harmonics (PSS, ENVLP)

Lets you select output harmonics. When you select from the *Output Harmonics* cyclic field, the form changes to let you specify appropriate data.

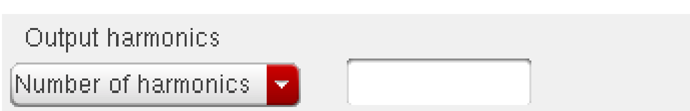
For PSS analysis, the choices are: *Number of Harmonics*, *Select from Range*, *Array of Coefficients*, *Array of Indices*.

For ENVLP analysis, the choices are: *Number of Harmonics*, and *Array of Indices*.

You can use the separate choices in the *Output Harmonics* cyclic field in combination. For example, if you add a harmonic using *Array of Coefficients*, the value you add appears in the *Select from Range* list box and as a currently active index field for the *Array of Indices*.

### Number of Harmonics (PSS and ENVLP)

Specifies a single value for the number of output harmonics.




1. Select *Number of Harmonics*.

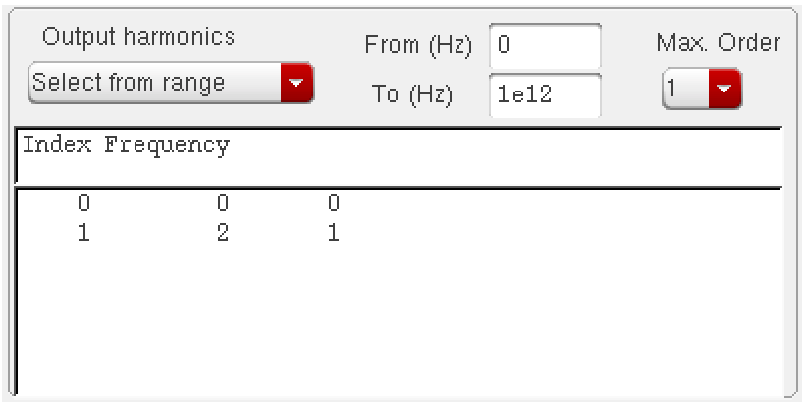
   The form changes to let you specify a single integer value.

2. Type the number of harmonics into the *Number of harmonics* field.

   Harmonics in this field are relative to the value of the fundamental. Type `0` into the *Number of harmonics* field to specify no harmonics.

### Select from Range (PSS)

Lets you enter a frequency range and select harmonics from within this range.




1. Select *Select from Range*.

   The form changes to display a cyclic field and two data entry fields.

2. In the *From (Hz)* and *To (Hz)* fields, type the lower and upper values for the frequency range.

3. From the *Max. Order* cyclic field, select the maximum order of harmonics that contribute to the output harmonics.

   The form changes to display harmonics matching these specifications in the list box.The first column is the index of a harmonic, the second column specifies its frequency, and the remaining columns specify tone coefficients for each fundamental tone that contributed to the listed harmonic.

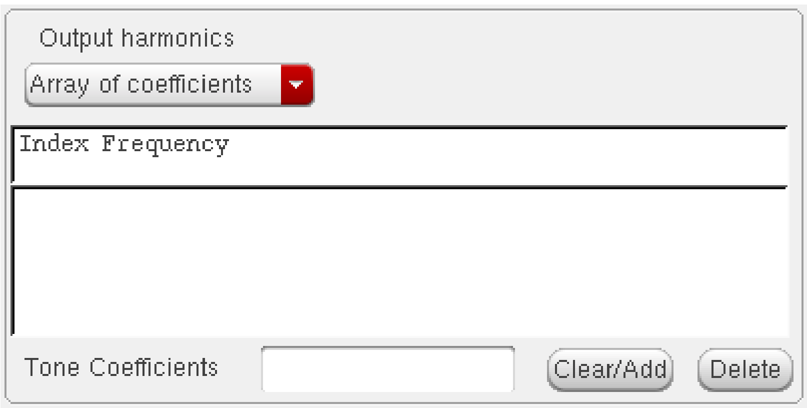
   If, for example, you select 5 in the *Max. Order* cyclic field, the sum of the absolute values of the tone coefficients contributing to the listed output harmonics is less than or equal to five. Negative integers displayed in the list box represent the tone coefficients of harmonics below the fundamental and positive integers represent the tone coefficients of harmonics above the fundamental.

4. Click a harmonic in the list box to select it.

   Select adjacent harmonics by clicking and dragging with the mouse over the harmonics to select. Select harmonics that are not adjacent by holding the `Control` key down while you click the individual harmonics. (Deselect a harmonic by holding the `Control` key down and clicking the harmonic.)

### Array of Coefficients (PSS)

Lets you specify output harmonics by entering their tone coefficients.




1. Select *Array of Coefficients*.

   The form changes to display a data entry field.

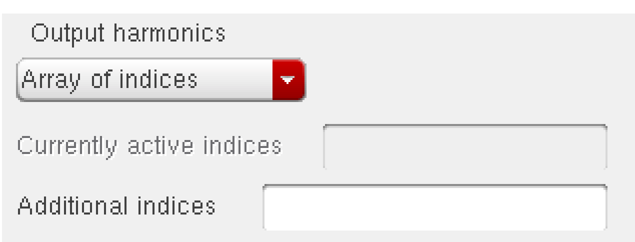
2. Type tone coefficients separated by spaces in the *Tone Coefficients* field and click *Clear/Add*.

   Values that appear in the list box are absolute values or relative to the fundamental, depending on the selection in the *Sweeptype* cyclic field.

   To delete a harmonic, select it in the list box and click *Delete*.

### Array of Indices (PSS and ENVLP)

Lets you specify an array of harmonics by entering their indices.



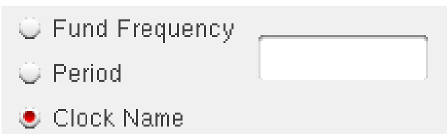

1. Select *Array of Indices*.

The form changes to display a data entry field. Any currently selected indices appear in the *Currently active indices* field.

2. Type the additional harmonic indices separated by spaces and in any order in the *Additional Indices* type-in field.

## Period (ENVLP)

Specifies the period (or, for autonomous circuits, the estimated period) of the clock fundamental.

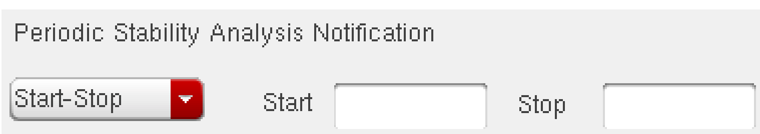



## Periodic Stab Analysis Notification (PSTB)

Defines the beginning and end points or a single point to be used for the analysis. The choices are: *Start-Stop*, *Center-Span*, *Single-Point*.

### Start – Stop

Defines the beginning and ending points for the sweep.



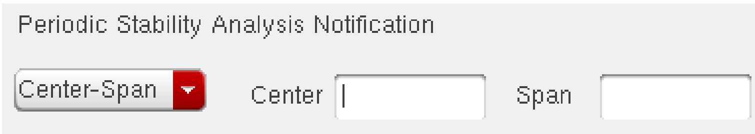

1. Select *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

### Center – Span

Defines the center point for the sweep and its span.




1. Select *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

### Single-Point

Defines the frequency range as a single point.

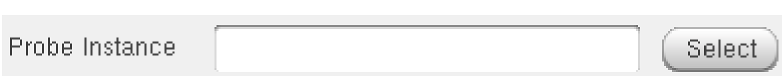

1. Select *Single-Point*.

   The form changes to let you type the frequency.

2. Type the specific frequency for the small-signal analysis.

## Probe Instance (PSTB)

In PSTB analysis, a `probe` component is placed in the feedback loop to identify and characterize the particular loop of interest. Introducing the `probe` component must not change the circuit characteristics.

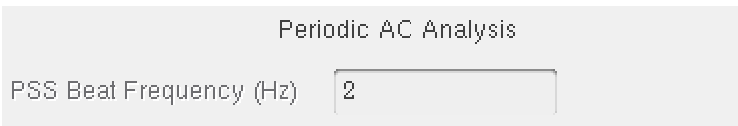



➤ Specify the probe instance by either

- Typing it into the *Probe Instance* field.
- Clicking the adjacent *Select* button and then clicking the node in the schematic.

## PSS Beat Frequency (PAC, PSTB, Pnoise, PXF)

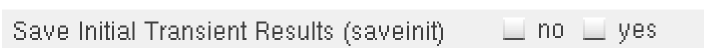
Displays the *Beat Frequency* for the initial PSS analysis.




This is a display-only field. You cannot edit the information here.

## Save Initial Transient Results (PSS, QPSS, HB)

Saves the initial transient waveforms.

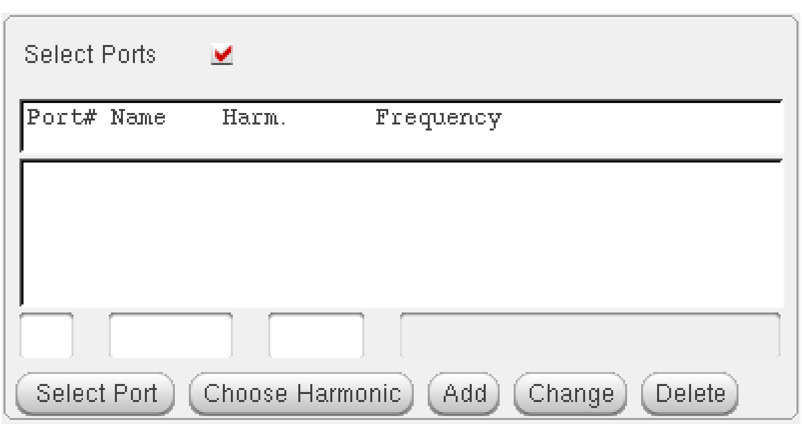



Highlight *yes* to save the initial transient waveforms.The default is *no*.

## Select Ports (PSP, QPSP)

Specifies the active ports for PSP and QPSP analyses.

The *Select Ports* fields include the following list box, data entry fields, and data entry buttons.



### Select Ports List Box

The *Select Ports* list box displays information about all selected ports. Ports in the list box are arranged numerically by the port numbers (*Port#*) that you assign.

#### Select Ports List Box Terms and Data Entry Fields

The list box displays the ports that have already been specified for PSP or QPSP analysis. Enter data for new ports using the data entry fields and buttons below the list box. You can enter new ports or edit values for existing ports. Notice that the frequency field is grayed out and you cannot edit frequency values.

**Port#** – Displays the port number assigned to the port. In the list box, ports are numbered sequentially.

**Name** – Displays the name of the port in the schematic. Use the *Select Port* button to select a port from the schematic.The name of the selected port in the schematic displays in the editable field.

**Harm.** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency. Use the *Choose Harmonic* button to display a list of harmonics and frequencies to select from. The selected harmonic index displays in the editable field. The associated frequency range also displays.

> For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.
>
> **Frequency** – Displays a range of frequency values associated with each harmonic index. Use the *Choose Harmonic* button to display a list of harmonics and frequencies to select from. The frequency range associated with the selected harmonic index value displays in the field. Notice that the Frequency field is grayed out and you cannot edit frequency range values.

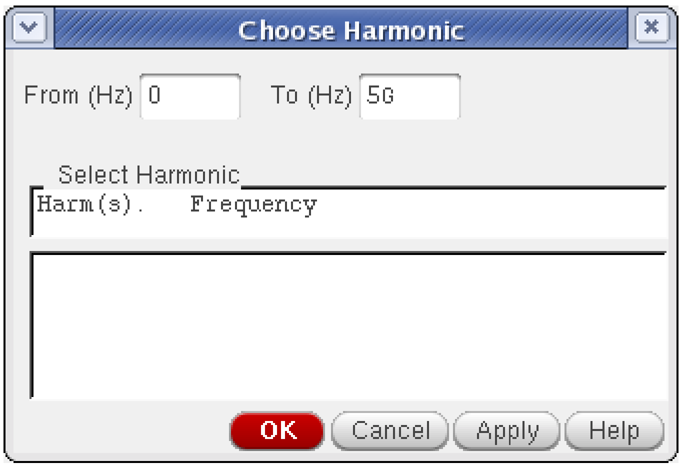
#### Select Ports Data Entry Buttons

- **Select Port** – Prompts you to select a port from the schematic. Displays the port’s name from the schematic in the editable field.
- **Choose Harmonic** – Opens the Choose Harmonic Pop UP where you can select a harmonic for the port.
- **Add** – Adds a port to the list box using the information in the data entry fields.

- **Change** – Adds a modified port to the list box using the modified information in the data entry fields. Highlight a port in the list box to move it's information to the data entry fields.
- **Delete** – Deletes a highlighted port from the list box.

### Choose Harmonic Pop Up (PSP and QPSP)

Selects a harmonic for a port. Open the Choose Harmonic pop up from *Select Ports*.




#### *Choose Harmonic List Box and Data Entry Fields*

The *Select Harmonic* list box displays the harmonic indexes and associated frequency ranges to select from for the ports that have already been specified for PSP or QPSP analysis. Changing the values in the *From (Hz)*, *To (Hz)* and *Max. Order* fields, changes the harmonic indexes and frequency ranges displayed in the *Select Harmonic* list box.

**From (Hz) and To (Hz) fields** - The upper and lower bounds for the frequency range.

**Max. Order** - For QPSP analysis only. Displays the maximum order of harmonics that contribute to the harmonics.

**Harm(s).** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency.

For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.

**Frequency** – Displays a range of frequency values associated with each harmonic index.

### Using the Select Ports Data Entry Fields and Buttons

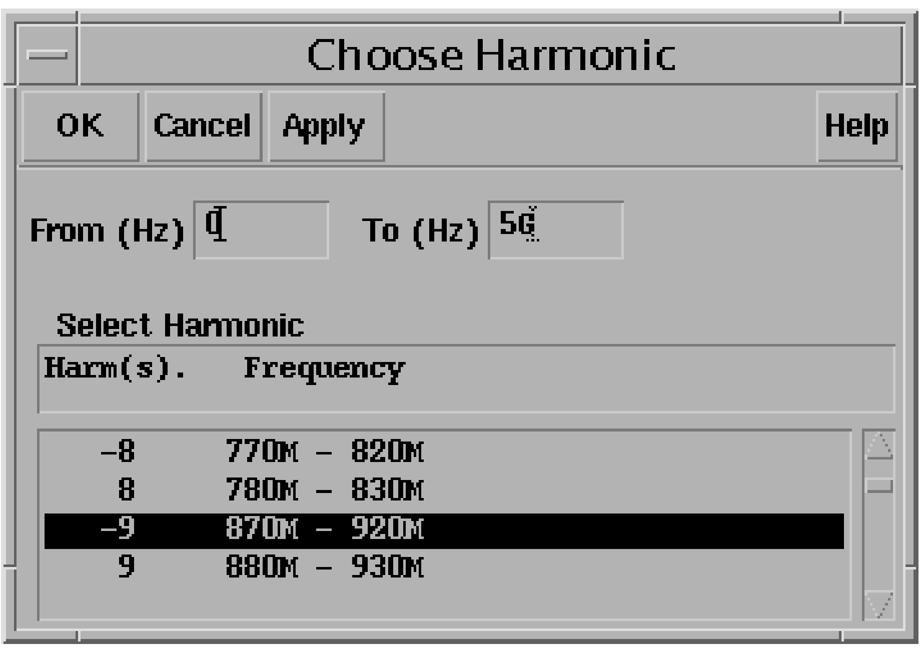
Use the data entry fields below the list box to edit the information in the *Select Ports* list box or to add new ports.

To specify a new port,

1. Type an integer in the first field, the *Port#* field. It is directly above the *Select Port* button.
2. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

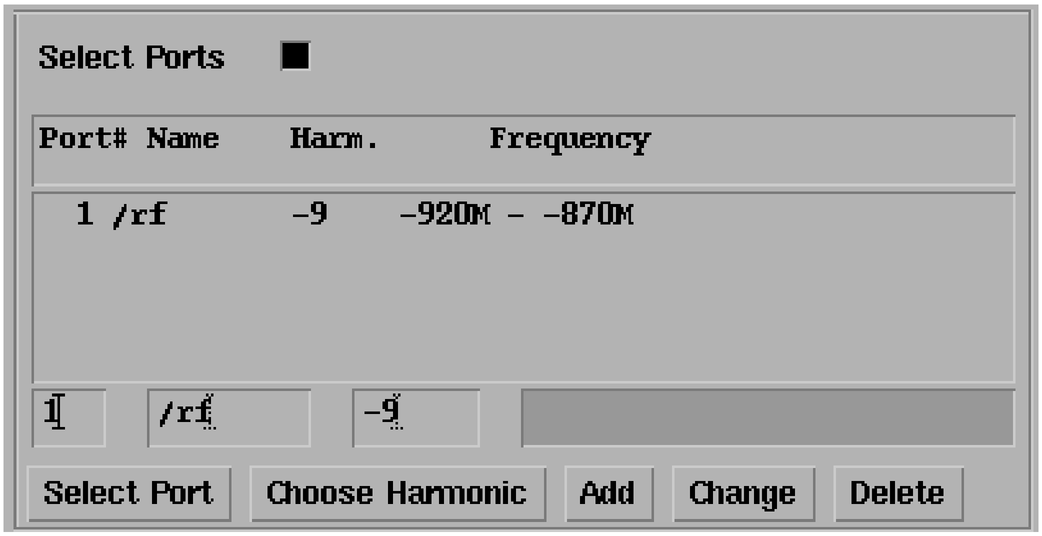
   `Select source...`
3. In the Schematic window, click an appropriate port, for example `/rf`.

   `/rf` appears in the *Name* field.
4. Click *Choose Harmonic*.
5. The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rf* port.




6. In the Choose Harmonic form, scroll through the list and highlight a harmonic to select it.
7. Click *OK*.

8. The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, the *Harm(s)* value of the selected harmonic displays in the *Harm*. field.
9. In the *Select Ports* area, click *Add*.
10. Information for the input port displays in the *Select Ports* list box.




To edit an existing port,

1. Highlight the port in the list box.
2. The port data appears in the data entry fields where you can edit it.
3. Modify a value in any one of the data entry fields. Values you cannot edit in the data entry fields (such as frequency range values), are grayed out.
4. Click *Change*.

   The modified port data replaces the port in the list box.

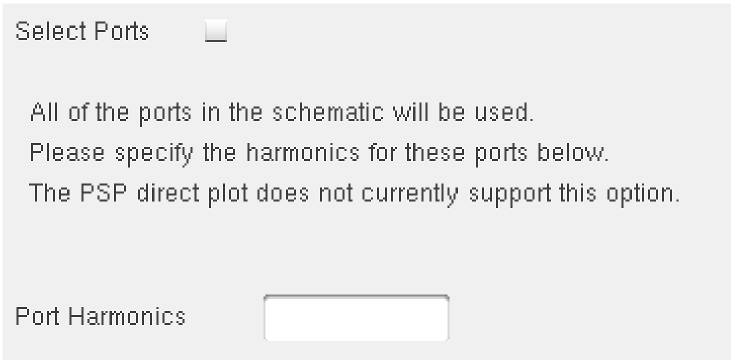
To delete an existing port,

1. Highlight the port in the list box.
2. The port data appears in the data entry fields.
3. Click *Delete*.

   The port is removed from the list box.

To use all ports in the schematic,

**1.** Deselect *Select Ports*.




**2.** In the *Port Harmonics* field, type the harmonics for these ports separated by spaces. (PSP direct plot does not currently support this option.)

## Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)

Lets you select the set of periodic small-signal output frequencies of interest. When you select from the *Sidebands* cyclic field, the form changes to let you specify appropriate data.

For HBAC, the choices are: *Maximum sideband*, *Select from range*, *Array of coefficients*. For the other analyses, the choices also include *Array of indices*

You can use the choices in the *Sidebands* cyclic field in combination. For example, if you add a sideband using *Array of coefficients*, the sideband value you added appears in the *Select from range* list box and as a *Currently active index* for the *Array of indices*.

### Maximum sideband

Specifies the number of frequency conversion terms (values of k) to take into account.


Prompts you for a *Maximum sideband* value and automatically generates a sideband array of the form:

```
[–maximum sideband . . . 0 . . . +maximum sideband]
```

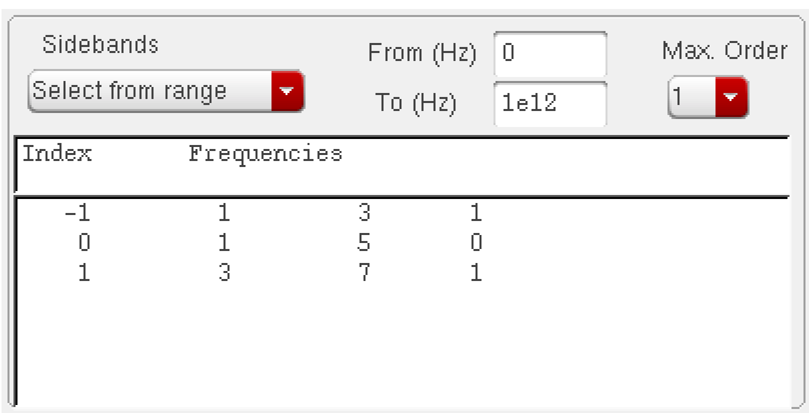
A typical analysis strategy is to begin by setting *Maximum sideband* to 7, the default value. Then increase the *Maximum sideband* value while observing the effect on output noise. If output noise changes, there is significant frequency conversion for values of k greater than 7. Continue to increase the *Maximum sideband* value until the output noise stops changing.

When you set *Maximum sideband* to zero, the reported output noise does not include any frequency conversion terms. For a fundamental oscillator, *Maximum sideband* must be at least 1 to see flicker noise upconversion. In general, small values for *Maximum sideband* are not recommended.

The *Maximum sideband* is ignored in HB small signal analysis when the value is larger than the *harms* or *maxharms* value of the large signal analysis.

### Select from range

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.




Depending on the selection in the *Sweeptype* cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands,

1. In the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range.

   The sideband frequencies displayed in the list box are within this range of frequencies.

2. From the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands.

If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

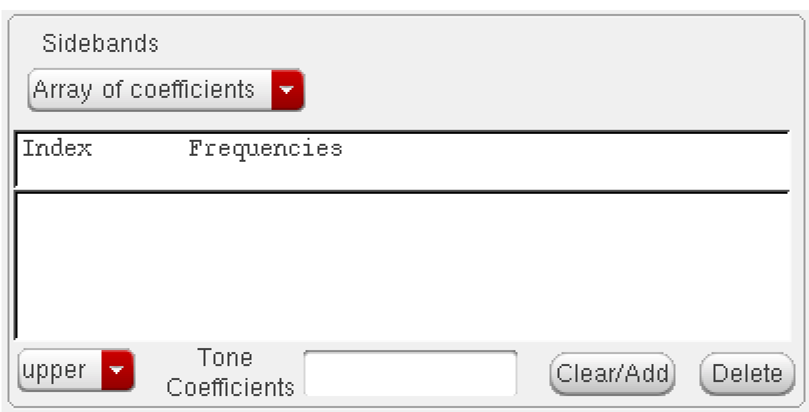
To select from the listed sidebands,

➤ Click a sideband in the list box to select it.

Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the `Control` key down while you click the individual sidebands. (Deselect a sideband by clicking a selected sideband while you hold the `Control` key down.)

### Array of Coefficients

Lets you specify sidebands by typing their tone coefficients.




To specify sidebands by using tone coefficients,

1. Type the tone coefficients, separated by spaces, into the *Tone Coefficients* type-in field.
2. Click *Clear/Add* to add the tone coefficient to the list box.

   Values that appear in the list box are *absolute* values or are *relative* to the fundamental, depending on the selection in the *Sweeptype* cyclic field.
3. Place the specified sideband above or below the fundamental using the *upper* cyclic field.

To delete a sideband,

➤ Select the sideband in the list box and click *Delete*.

### Array of Indices

Lets you specify an array of sidebands by entering their indices.


To specify an array of sidebands by using indices,

1. Choose *Array of Indices*.

   Currently selected indices appear in the *Currently active indices* field.

2. Type the indices of the sidebands you want to specify into the *Additional indices* field.

   You can type the additional sideband indices in any order. Separate the indices with spaces.

## Sidebands (QPAC, QPnoise, QPXF)

Lets you select the set of quasi-periodic small-signal output frequencies of interest for the QPAC, QPnoise, or QPXF analyses.

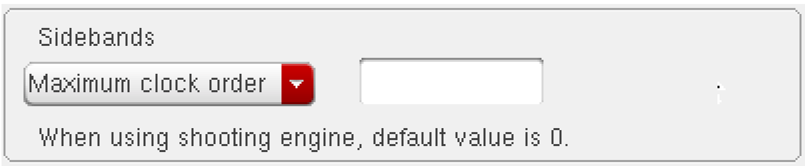
For the quasi-periodic analyses, sidebands are vectors specified with the `sidevec` and `clockmaxharm` parameters.

When you select from the *Sidebands* cyclic field, the form changes to let you specify appropriate data. Choices are: *Maximum clock order*, *Select from range*, and *Array of coefficients*.

You can use the choices in the *Sidebands* cyclic field in combination. For example, if you add a sideband using *Array of Coefficients*, the sideband value you added appears in the *Select from Range* list box.

### Maximum clock order

Specifies the largest sideband value and generates the sideband array.



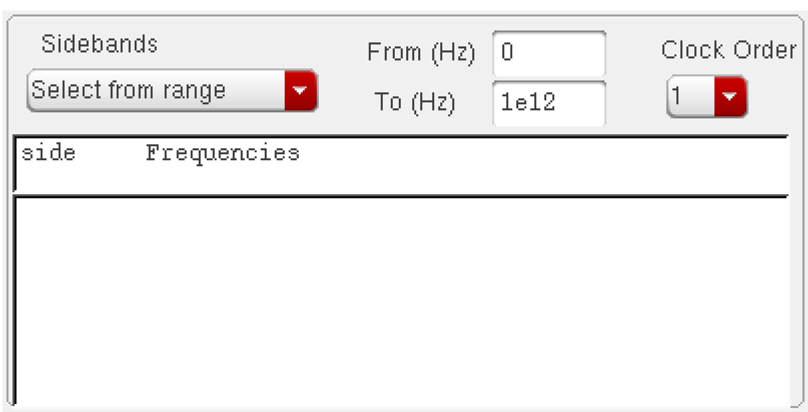

Prompts you for a maximum sideband value and automatically generates a sideband array of the form:

```
[-max. sideband . . . 0 . . . + max. sideband]
```

If the sidebands field is left blank, the default value is used. For shooting, this is 7 sidebands, and for harmonic balance it defaults to the number of harmonics specified in the harmonic balance setup.

### Select from range

Lets you specify a frequency range and select sidebands from within this range of sideband values.




Depending on the selection in the [Sweeptype](#) cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands,

1. In the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range.

   The sideband frequencies displayed in the list box are within this range of frequencies.

2. From the *Max. Order* cyclic field, select the maximum order of harmonics that contribute to the sidebands.

   If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

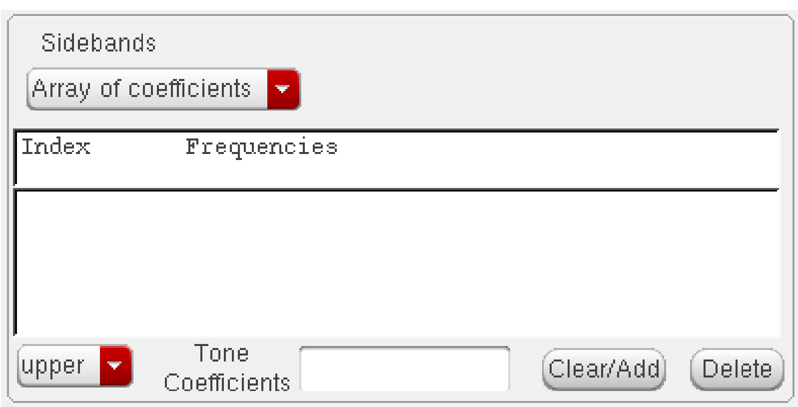
To select from the listed sidebands,

➤ Click a sideband in the list box to select it.

   Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the `Control` key down while you click the individual sidebands. (Deselect a sideband by clicking a selected sideband while you hold the `Control` key down.)

### Array of Coefficients

Lets you specify sidebands by typing their tone coefficients.




To specify sidebands using tone coefficients,

1. Type the tone coefficients, separated by spaces, into the *Tone Coefficients* type-in field.
2. Click *Clear/Add* to add the tone coefficients to the list box.

   Values that appear in the list box are *absolute* values or are *relative* to the fundamental, depending on the selection in the *Sweeptype* cyclic field.

To delete a sideband,

➤ Select it in the list box and click *Delete*.

## Multi-rate Harmonic Balance (HB)

Lets you divide a circuit into partitions that share common fundamental tones and harmonics. The division can improve simulation performance by allowing the simulator to use different harmonic sets in different partitions rather than assuming the worst-case scenario for the design as a whole.

**Note:** At present, multi-rate harmonic balance is only supported for driven circuits. Autonomous circuits will however be supported in future.

To define partitions for a multitone circuit,

1. Highlight *Multi-rate Harmonic Balance (Signal Partition)*.

   The Signal Partition form appears.

2. Specify the instances to be included in each partition. Either

   - ❑ Type the instance names into the type-in fields at the bottom of the Signal Partition pane. Use the format

     `inst1 ; inst2 ; ...`

     For example, you might enter a list such as

     `/I1;V4;V5`

   - ❑ Click *Select Port Instances*; select, in the Schematic Editing window, the instances to be included in the partition; and press *Esc* when you are done.

3. In the type-in field for the *Harms* column, type the harmonic values to be used for the partition. Use the format

   `val1 val2 ...`

4. In the type-in field for the *Fundratios* column, type the integer ratios to be used to calculate the effective fundamental tone for the partition. Use the format

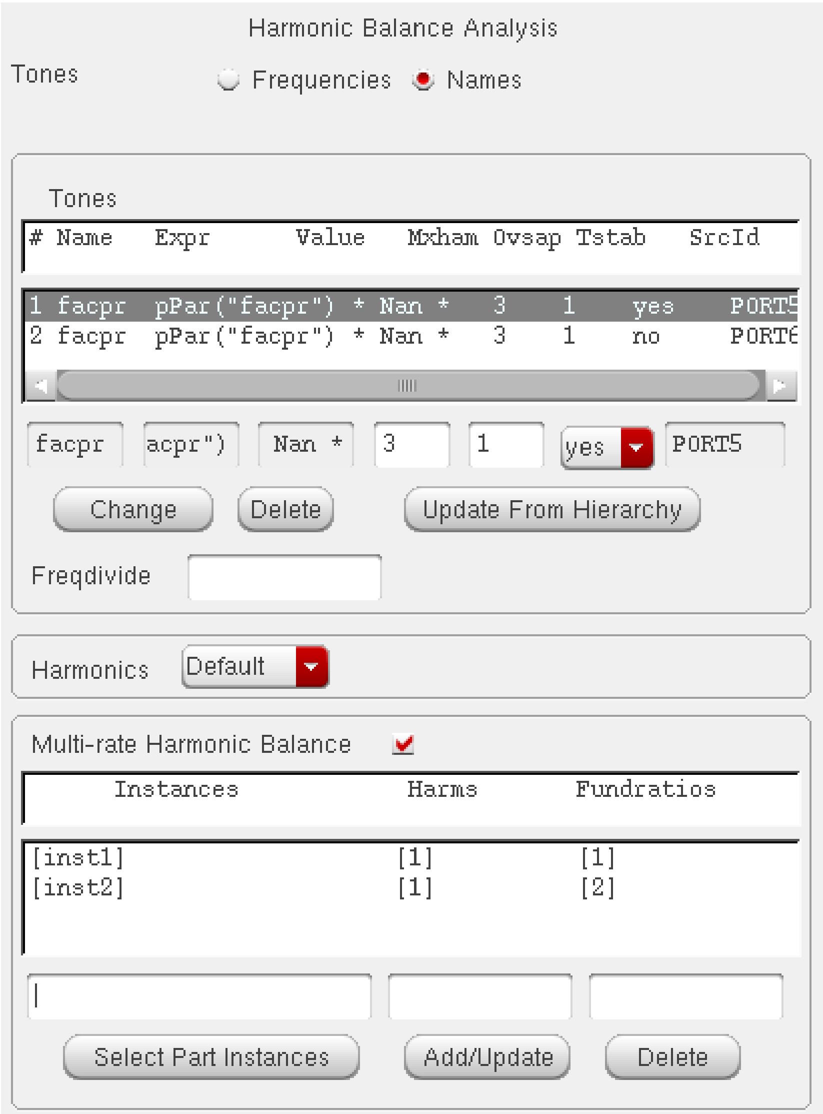
   `val1 val2 ...`

5. Click *Add/Update*.

   The new information appears in the table of partition information.

6. If you want to specify another partition, repeat the process from step 2.

The information in the *Fundratios* column of the Signal Partition pane interacts with the information in the *Tones* pane to characterize the signal partitions. For example, with the form set up as follows




there are 2 entries in each of the *Fundratios* specifications, one for each of the unique tones listed in the *Tones* pane (`vco` and `rf`). The simulator determines the fundamental frequencies of the tones in a signal partition by multiplying the first value in the *Fundratios* column with the frequency of the first (lowest numbered) tone listed in the *Tones* pane. In this example, for the partition that includes v1 and v2, the fundamental frequency for the `vco` tone is 2 * 4.96G/freqdivide. Here, freqdivide is a large signal frequency division.

**Note:** freqdivide is a parameter for hb analysis. When it is not equal to 1, the fundamental frequency of the large tone is equal to the frequency of the source (vco divided by freqdivide).

Similarly, the fundamental frequency for the second tone in the partition (`rf`) is determined by multiplying the second *Fundratios* value with the frequency of the second tone listed in the *Tones* pane (in this example, the second tone is numbered as 3 and 4). In this example the result is 1 * 2.479G.

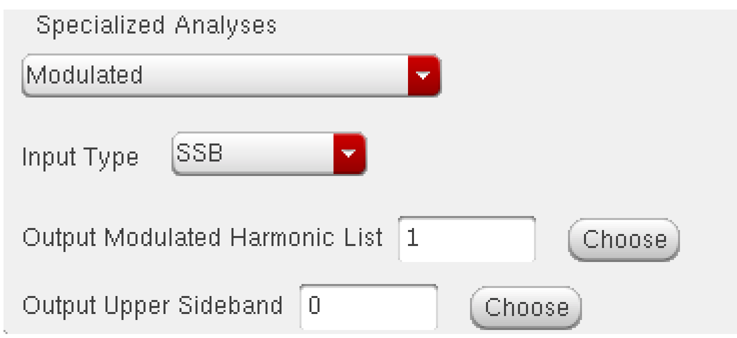
## Specialized Analyses (PAC, HBAC)

Measures AM and PM small-signal effects for the PAC and HBAC analyses.

### Modulated

When you select *Modulated* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *Modulated* PAC analysis include the following.

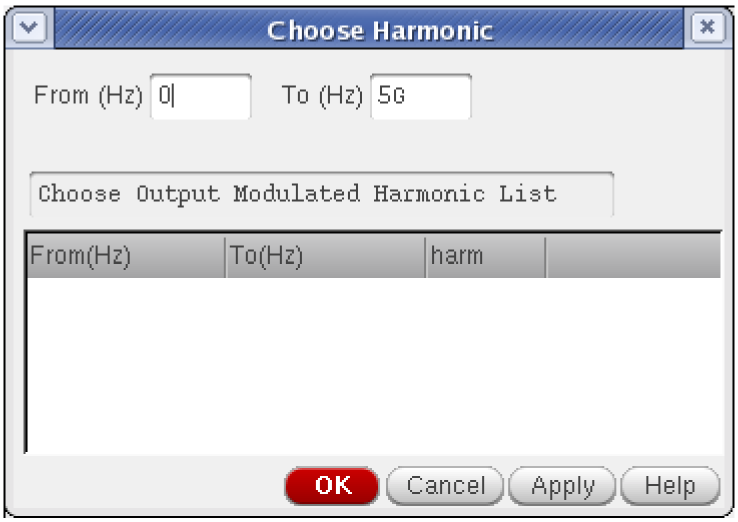



#### *Modulated Analysis Terms and Data Entry Fields*

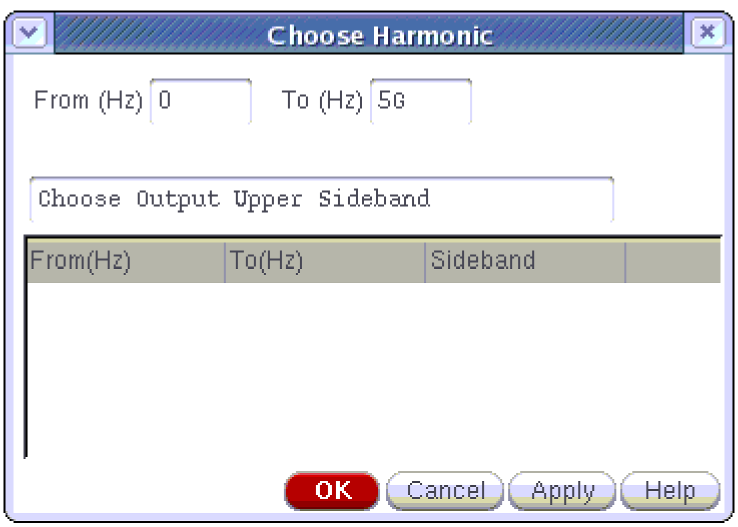
- **Input Type** – This cyclic field displays one of two values: *SSB* and *SSB/AM/PM.*
- **Output Modulated Harmonic List** – Lists the harmonic indexes.
- **Output Upper Sideband** – (Displays when you choose the *Input Type SSB*.) Click *Choose* to display the Choose Harmonic pop up.
- **Input Modulated Harmonic** – (Displays when you choose the *Input Type SSB/AM/PM*.) Click *Choose* to display the Choose Harmonic pop up.

### *Choose Harmonic Pop Up (PAC)*

Selects harmonics or sidebands for the analysis. Display the Choose Harmonics pop up from *Modulated* Analysis with the *Choose* button.




or




### *Choose Input/Output Modulated Harmonic List Box and Data Entry Fields*

The *Choose Input* (or *Output*) *Modulated Harmonic List* box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To (Hz)* fields changes the harmonic indexes and frequency ranges displayed in the list box.

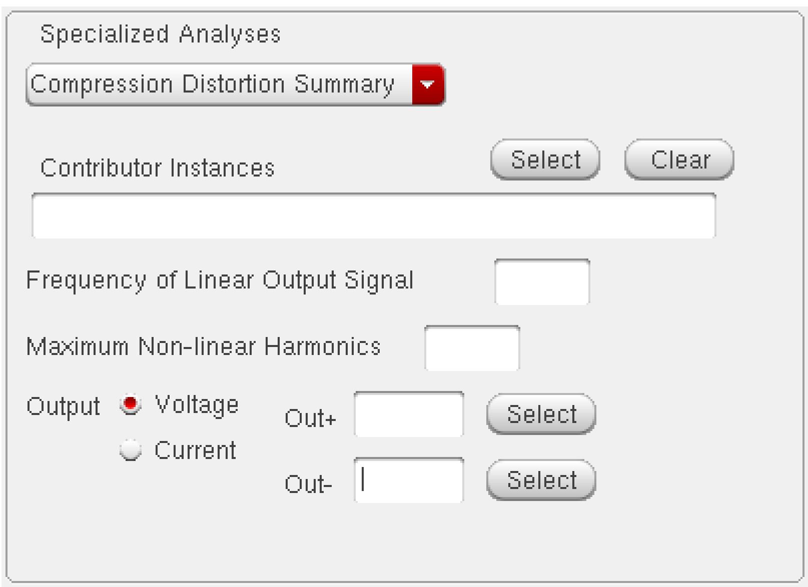
**From (Hz) and To (Hz) fields -** The upper and lower bounds for the frequency range.

**Harm** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency. For example, you set up a PSS analysis of port1 named RF with an harmonic index of 1. Given a PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.

### Compression Distortion Summary Specialized PAC Analysis

When you select *Compression Distortion Summary* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *Compression Distortion Summary* PAC analysis include the following.




#### *Compression Distortion Summary Terms and Data Entry Fields*

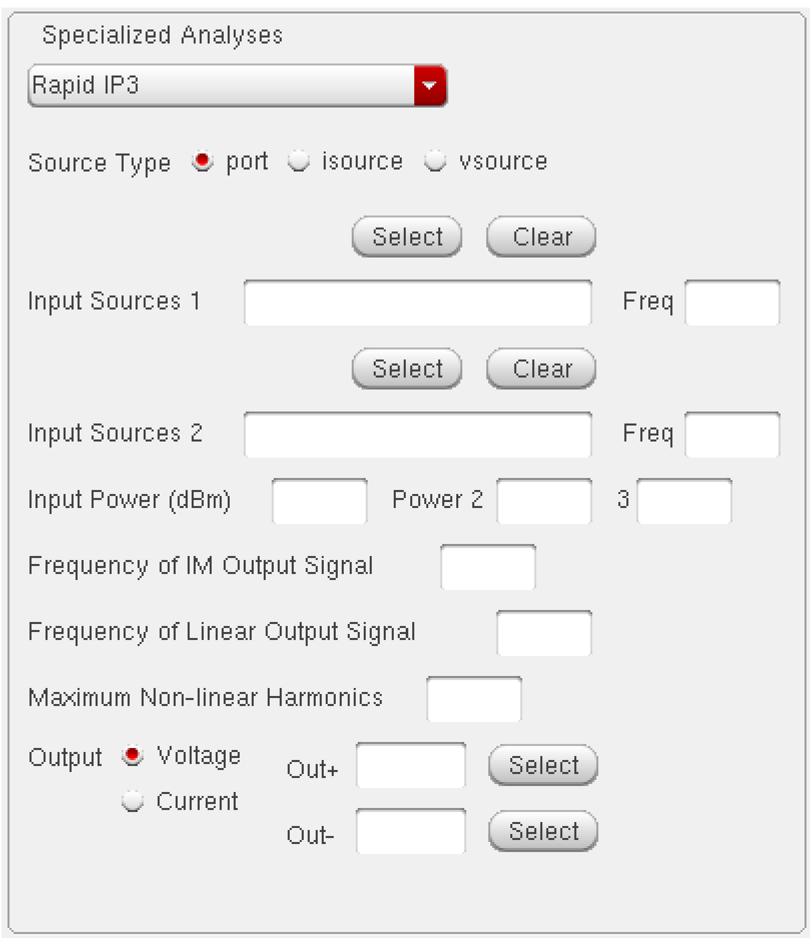
- **Contributor Instances** – Selects and displays an array of device names for the distortion summary. When the field is empty, calculates distortion from each non-linear device.
- **Frequency of Linear Output Signal** – Frequency of the linear output signal. Default is 0.
- **Maximum Non-linear Harmonics** – Frequency of the IM output signal. Default is 0.

- **Output** – **Voltage** Displays output signals.
- **Out+** Displays the positive output signal.
- **Out-** Default is ground.
- **Output** – **Current** Displays the output terminal.
- **Term -** Selects and displays the name of the output terminal.

### Rapid IP3 Specialized PAC Analysis

When you select *Rapid IP3* in the *Specialized Analyses* cyclic field, the PAC form changes to let you enter more information.

The *Specialized Analysis* fields for the *Rapid IP3* PSS and PAC analysis include the following.



### *Rapid IP3 Analysis Terms and Data Entry Fields*

- **Source Type** – Specifies whether the source is a current, voltage, or port. Select *isource*, *vsource*, or *port*.
- **Input Sources 1** and **Freq** fields – The RF source magnitude and frequency. Click *Select* and select the RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Start* field for the PAC analysis.
- **Input Sources 2** and **Freq** fields – A second RF source magnitude and frequency. Click *Select* and select the second RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Stop* field for the PAC analysis.

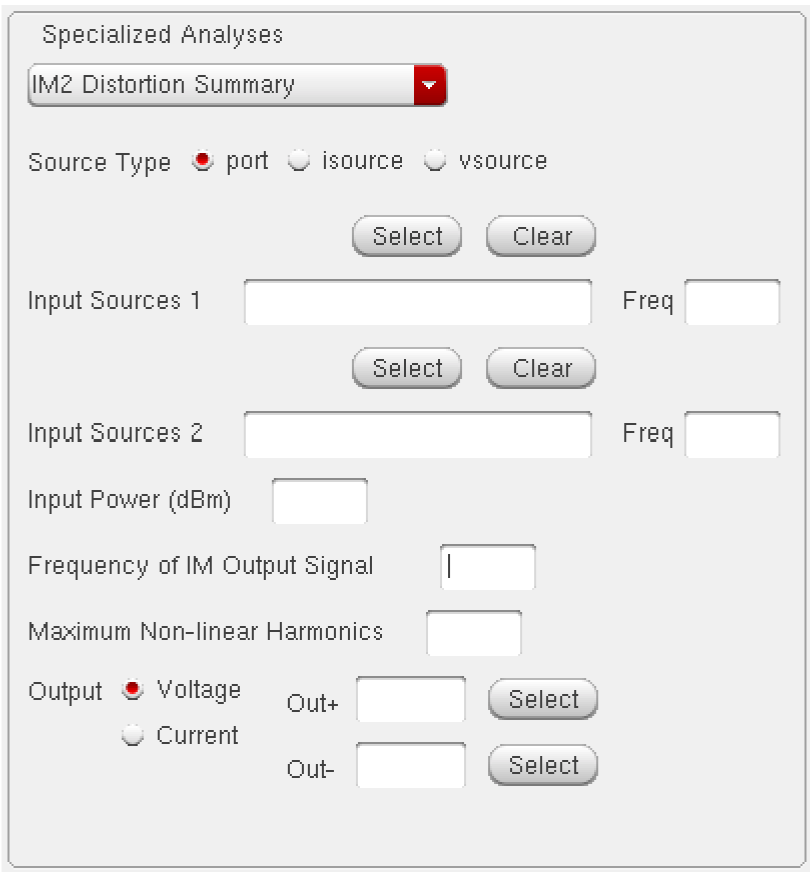
  The RF1 source and the RF2 source can be the same. The associated frequency values must be different.
- **Input Power (dBm)** – RF source power.
- **Frequency of IM Output Signal** – IM3 frequency at the output.
- **Frequency of Linear Output Signal** – IM1 frequency at the output.
- **Maximum Non-linear Harmonics** – Number of harmonics used for the RF signals. Default is 4.
- **Output** – **Voltage** Output node 1 where IP3 is measured.
- **Out+** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a net in the schematic.
- **Out-** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a second net in the schematic. The default is *gnd!*.
- **Output** – **Current** Selects a terminal in the schematic.
- **Term -** (Displays when you choose *Current* for *Output*.) For output current in a source. Specify it as `<source_name>:p`.

  When the output is current in a port, you must use the ADE *Outputs - To be saved - Select in schematic* menu pick and select the terminal in the schematic where current is to be computed and saved.

### IM2 Distortion Summary

When you select *IM2 Distortion Summary* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *IM2 Distortion Summary* include the following.




#### *IM2 Distortion Summary Terms and Data Entry Fields*

- **Source Type** – Specifies whether the source is a port, current, or voltage. Select *port, isource*, or *vsource*.
- **Input Sources 1** and **Freq** fields – The RF source magnitude and frequency. Click *Select* and select the RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Start* field for the PAC analysis.

- **Input Sources 2** and **Freq** fields – A second RF source magnitude and frequency. Click *Select* and select the second RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Stop* field for the PAC analysis.

  The RF1 source and the RF2 source can be the same. The associated frequency values must be different.

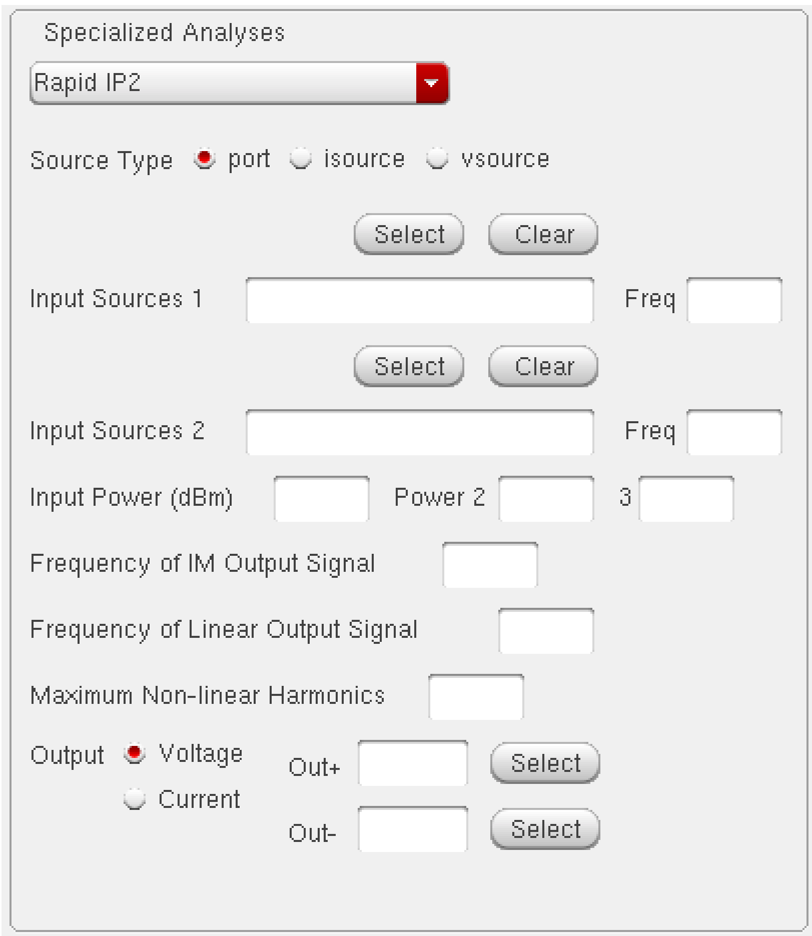
- **Input Power (dBm)** – RF source power.
- **Frequency of IM Output Signal** – IM3 frequency at the output.
- **Maximum Non-linear Harmonics** – Number of harmonics used for the RF signals. Default is 4.
- **Output** – **Voltage** Output node 1 where IP3 is measured.
- **Out+** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a net in the schematic.
- **Out-** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a second net in the schematic. The default is *gnd!*.
- **Output** – **Current** Selects a terminal in the schematic.
- **Term -** (Displays when you choose *Current* for *Output*.) For output current in a source. Specify it as `<source_name>:p`.

  When the output is current in a port, you must use the ADE *Outputs - To be saved - Select in schematic* menu pick and select the terminal in the schematic where current is to be computed and saved.

### Rapid IP2

When you select *Rapid IP2* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *Rapid IP2* analysis include the following.




#### *Rapid IP2 Terms and Data Entry Fields*

- **Source Type** – Specifies whether the source is a port, current, or voltage. Select *port, isource*, or *vsource*.
- **Input Sources 1** and **Freq** fields – The RF source magnitude and frequency. Click *Select* and select the RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Start* field for the PAC analysis.

- **Input Sources 2** and **Freq** fields – A second RF source magnitude and frequency. Click *Select* and select the second RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Stop* field for the PAC analysis.

  The RF1 source and the RF2 source can be the same. The associated frequency values must be different.

- **Input Power (dBm)** – RF source power.
- **Frequency of IM Output Signal** – IM3 frequency at the output.
- **Frequency of Linear Output Signal** – IM1 frequency at the output.
- **Maximum Non-linear Harmonics** – Number of harmonics used for the RF signals. Default is 4.
- **Output** – **Voltage** Output node 1 where IP3 is measured.
- **Out+** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a net in the schematic.
- **Out-** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a second net in the schematic. The default is *gnd!*.
- **Output** – **Current** Selects a terminal in the schematic.
- **Term -** (Displays when you choose *Current* for *Output*.) For output current in a source. Specify it as `<source_name>:p`.

  When the output is current in a port, you must use the ADE *Outputs - To be saved - Select in schematic* menu pick and select the terminal in the schematic where current is to be computed and saved.

## Start ACPR Wizard (ENVLP)

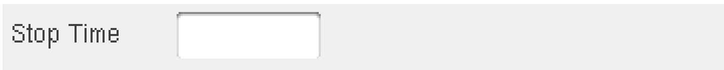
Opens the ACPR Wizard form. The ACPR Wizard helps you through the complex process of measuring ACPR (Adjacent Channel Power Ratio) and PSD (Power Spectral Density).


See "ACPR Wizard" on page 134 for related information.

You can also open the ACPR Wizard from the ADE window by choosing *Tools – RF – Wizards – ACPR.*

## Stop Time (ENVLP)

Specifies the end point in an ENVLP analysis.




➤ Type a time value in the *Stop Time* field.

Make the time interval long enough to let slow signals complete at least one cycle. See “Clock Name and Select Clock Name Button (ENVLP)” on page 9 for related information.

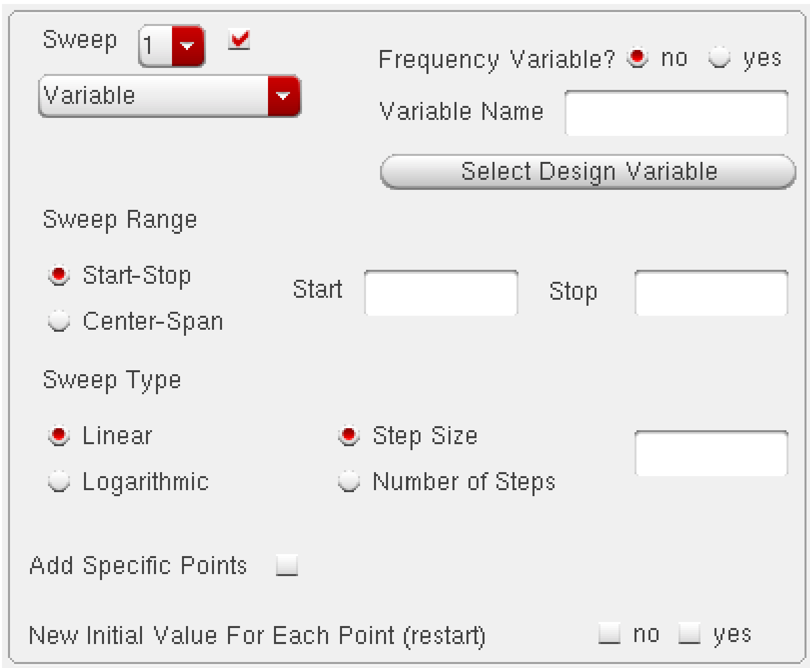
## Sweep (PSS, QPSS, HB)

Specifies how a sweep is performed. Choices are: *Variable*, *Temperature*, *Component Param*, and *Model Param*.

When you activate *Sweep* on the PSS or QPSS analysis form, the *Frequency Sweep Range* on the small-signal analysis forms can be used to sweep a single point or a small signal.

### Variable

Sweeps a design variable.

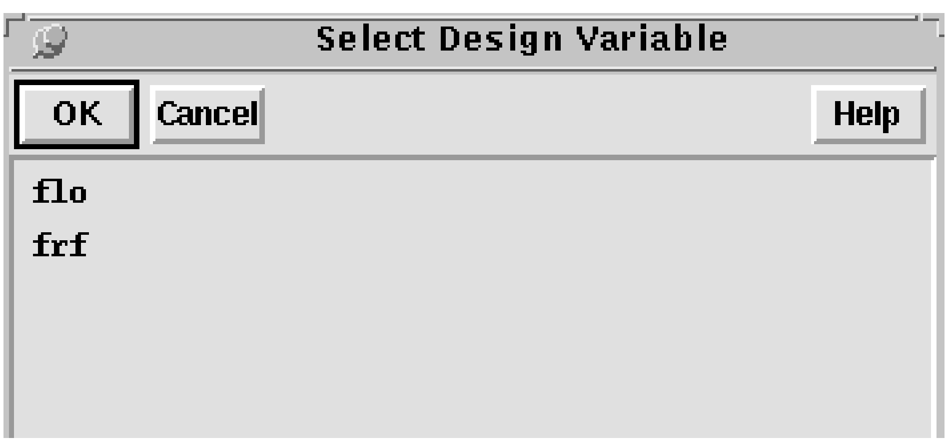



To sweep a design variable,

1. Select *Variable* from the cyclic field.

   The form changes to accept data for the variable sweep.

2. Click *Select Design Variable* to display the Select Design Variable form.



3. In the Select Design Variable form, select a variable and click *OK*.

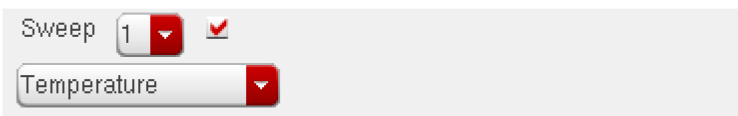
   The variable name appears in the *Variable Name* field. (You can also simply type the name in the *Variable Name* field.)

4. If the selected variable is a frequency variable, highlight *yes* for *Frequency Variable*.

5. You can sweep up to three variables at the same time. The first level of sweep is enabled by default. For second and third level(s), the *Enable* button must be clicked manually to include that level in the sweep.

#### Temperature

Collects temperature data during the sweep.




#### Component Param

Sweeps a component parameter.

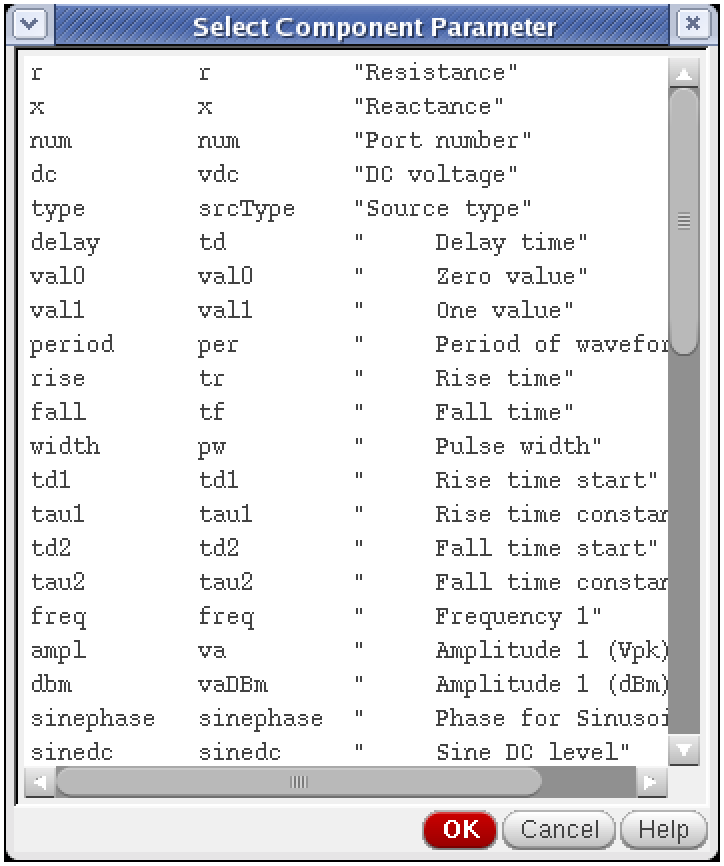

1. Select *Component Param.*

   The form changes to accept data for the component parameter sweep.

2. Click *Select Component*.

3. Click the component in the schematic.

The Select Component Parameter form appears.




4. Select a parameter in this form and then click *OK*.

   Both the component and parameter names appear in their respective fields. (You can also simply type the component and parameter names in their respective fields.)

### Model Param

Sweeps a model parameter.


1. Select *Model Param*.

   The form changes to accept data for the model parameter sweep.

2. Type the model name and the parameter name in their respective type-in fields.

## Sweep Range, Sweep Type, and Add Specific Points (PSS, QPSS)

Defines the analysis sweep range, the type of sweep, and any additional individual sweep points for the large-signal PSS analysis or the medium-signal QPSS analysis that precedes a small-signal analysis.

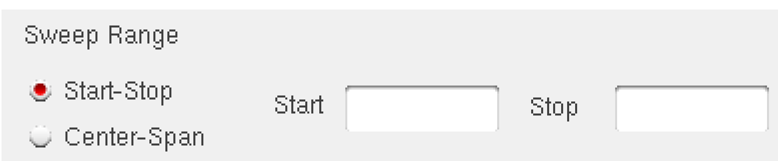
When you highlight *Sweep*, the *Sweep Range*, *Sweep Type*, and *Add Specific Points* fields open below the *Sweep* fields.

### Sweep Range

Specifies the bounds for the sweep either as beginning and ending points or as a center point and a span.

#### *Start – Stop*

Defines the beginning and ending points for the sweep.




1. Highlight *Start-Stop*.

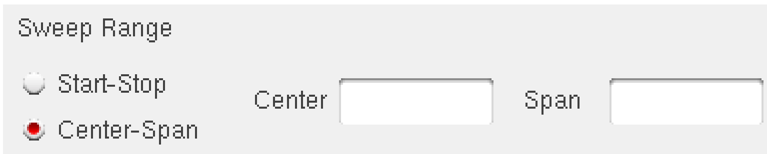
   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

- The *Start* and *Stop* sweep values can be frequencies, periods, or design variable values that correspond to your selection in the *Sweep* cyclic field.

### *Center – Span*

Defines the center point for the sweep and its span.




1. Highlight *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

## Sweep Type

Specifies whether the sweep is linear or logarithmic.

### *Linear*

Specifies a linear sweep.


1. Highlight *Linear*.

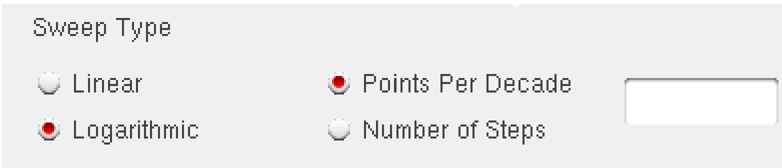
   The form changes to let you type either the step size or the total number of points (steps).

2. Either

   - ❑ Highlight *Step Size* and type the size of each step in the field.
   - ❑ Highlight *Number of Steps* and type the number of steps in the field.

#### *Logarithmic*

Specifies a logarithmic sweep.




1. Highlight *Logarithmic*.

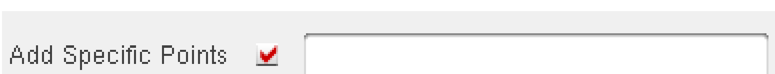
   The form changes to let you type either the number of points per decade or the total number of points (steps).

2. Either

   - Highlight *Points per Decade* and type the number of points per decade in the field.
   - Highlight *Number of Steps* and type the number of steps in the field.

#### Add Specific Points

Specifies additional sweep points for the analysis.




1. Highlight *Add Specific Points.*
2. Type the additional sweep point values into the field.

   Separate the sweep points with spaces.

### Sweep Type (PSTB, HBAC, HBNOISE)

Specifies whether the sweep is linear, logarithmic, or chosen automatically.

### Linear

Specifies a linear sweep.


1. Select *Linear*.

   The form changes to let you specify the step size or the total number of points (steps).

2. Either

   - Highlight *Step Size* and type the size of each step in the field.
   - Highlight *Number of Steps* and type the number of steps in the field.

### Logarithmic

Specifies a logarithmic sweep.


1. Select *Logarithmic*.

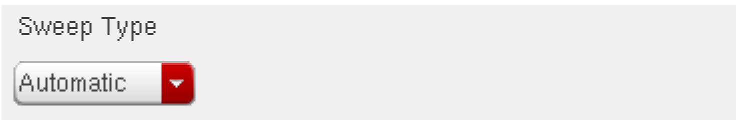
   The form changes to let you specify the number of points per decade or the total number of points (steps).

2. Either

   - Highlight *Points per Decade* and type the number of points per decade in the field.
   - Highlight *Number of Steps* and type the number of steps in the field.

### Automatic

Automatically chooses either the linear or logarithmic sweep types. The sweep is linear when the ratio of stop to start values is less than 10 and the sweep is logarithmic when the ratio is 10 or greater.




➤ Select *Automatic*.

## Sweeptype (Pnoise)

Controls the inclusion of the *sweeptype* parameter in the netlist. Choices are: *absolute*, *relative*, and *blank*, with *blank* selecting the appropriate Spectre RF default.

The results vary depending on whether you are simulating an autonomous circuit (an oscillator) or a driven circuit (a mixer) as determined by the *Oscillator* button selection on the PSS Choosing Analyses form.

In general,

- When you simulate an autonomous circuit (the *Oscillator* section of the PSS Choosing Analyses form is active), you can select *relative* for *Sweeptype*. If you leave *Sweeptype* blank, it defaults to *relative*.
- When you simulate a driven circuit (the Oscillator section of the PSS Choosing Analyses form is not active), you can either select either *relative* or *absolute* for *Sweeptype*. If you leave *Sweeptype* blank, it defaults to *absolute*.

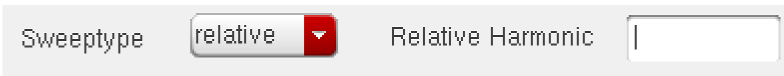
### Absolute

Puts the *Sweeptype* as *absolute* in the Spectre netlist.


In the output for the `Pnoise` sweep, the x axis corresponds to the *Start* and *Stop* values. There is no indication on the plot that you selected *Sweeptype* as *absolute.*

#### Relative

Puts the *Sweeptype* as *relative* in the Spectre netlist.




When you select *Sweeptype* as *relative*, you must also enter a value for *Relative Harmonic*. If you enter a `1`, it appears in the netlist as `relharmnum=1`.

In the output for a Pnoise analysis, the x axis corresponds to the *Start* and *Stop* values in the `Pnoise` sweep. The plot label indicates the selected relative harmonic (for example, `relharmnum=1`). In prior versions of the software, there is no indication.

#### Default

Does not put a *Sweeptype* parameter in the Spectre netlist. Sets the appropriate *Sweeptype* value depending on the circuit type.


Sets the appropriate *Sweeptype* value according to whether the circuit is autonomous or driven.

- For autonomous circuits, Spectre RF automatically sets *Sweeptype* to *relative* and the *Relative Harmonic* to `1`.
- For driven circuits, Spectre RF automatically sets *Sweeptype* to *absolute* and displays the message

  `Sweep is Currently Absolute`

In the output, the x axis corresponds to the *Start* and *Stop* values in the `Pnoise` sweep.

- For autonomous circuits, the plot label indicates the default *Relative Harmonic* (`relharmnum=1`). In prior versions of the software, there is no indication.
- For driven circuits, Spectre RF automatically sets *Sweeptype* to *absolute*. There is no indication in the output that you selected *Sweeptype* as *default*.

## Sweeptype (PAC, PXF, HBAC, HBNOISE)

Controls the inclusion of the *sweeptype* newlink parameter in the Spectre netlist. Choices are: *absolute*, *relative*, and *default*, with *default* selecting the appropriate Spectre RF default.

The results vary depending on whether you are simulating an autonomous circuit (an oscillator) or a driven circuit (a mixer) as determined by the *Oscillator* button selection on the PSS or HB Choose Analyses form.

In general,

- When you simulate an autonomous circuit (the *Oscillator* section of the PSS or HB Choosing Analyses form is active), you can select *relative* for *Sweeptype*. If *Sweeptype* is set to *default*, the effective value is *relative*.
- When you simulate a driven circuit (the *Oscillator* section of the PSS or HB Choosing Analyses form is not active), you can select either *relative* or *absolute* for *Sweeptype*. If *Sweeptype* is set to *default*, the defective value is *absolute*.

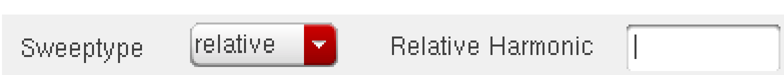
### Absolute

Sets the *Sweeptype* as *absolute* in the Spectre netlist.


There is no indication on the plot that you selected *Sweeptype* as *absolute.*

### Relative

Sets the *Sweeptype* as *relative* in the Spectre netlist.




When you select the *relative* value for *Sweeptype*, you must also enter a value for *Relative Harmonic*. If you type `1`, it appears in the netlist as `relharmnum=1`.

For PXF analysis with *sweeptype* set to *relative* and freqaxis=in, the simulation output is shifted frequency with the x axis labeled

```
relative frequency (offset from xx HZ)
```

For PAC analysis with *sweeptype* set to *relative* and freqaxis=out, the simulation output is shifted frequency with the x axis labeled

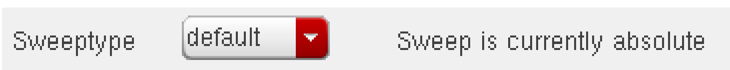
`relative frequency (offset from xx HZ)`

For PXF analysis with *sweeptype* set to *relative* and freqaxis=absin, the simulation output is absolute frequency. For PAC analysis with *sweeptype* set to *relative* and freqaxis=absout, the simulation output is absolute frequency.

At the bottom of the direct plot form for the analysis, the `freqaxis` value displays along with one of the following messages-`relative frequency (offset xxx)` or `relative freq`. The plot label also indicates the selected relative harmonic (for example, `relharmnum=1`).

#### Default

Does not put a *Sweeptype* parameter in the Spectre netlist.




Sets the appropriate Spectre RF default depending on whether the circuit is autonomous or driven.

For autonomous circuits, Spectre RF automatically sets the *Sweeptype* to *relative* and the *Relative Harmonic* to `1`.

For driven circuits, Spectre RF automatically sets the *Sweeptype* to *absolute* and displays the message `Sweep is Currently Absolute.`

In the output, the x axis corresponds to the *Start* and *Stop* values in the `Pnoise` sweep.

- For autonomous circuits, the plot label indicates the default *Relative Harmonic* (`relharmnum=1`). In prior versions of the software, there is no indication.
- For driven circuits, Spectre RF automatically sets the *Sweeptype* to *absolute*. There is no indication you selected *Sweeptype* as *blank*.

### Sweeptype (PSP, QPSP)

Controls the inclusion of the *sweeptype* parameter in the Spectre netlist. Choices are: *absolute, relative*, and *blank,* with *blank* selecting the appropriate Spectre RF default.

Because the computations for PSP analysis involve inputs and outputs at frequencies that are relative to multiple harmonics, *Sweeptype* behaves differently in PSP and QPSP analysis than it does in PAC, Pnoise, and PXF analyses. With PSP and QPSP analysis, the frequency of the input and the frequency of the response are usually different.

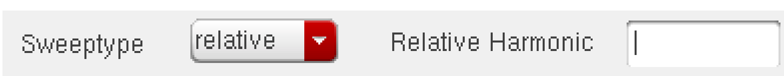
### Absolute

Puts the *Sweeptype* as *absolute* in the Spectre netlist.


Specifying *Sweeptype* as *absolute*, sweeps the absolute input source frequency. There is no indication on the plot that you selected *Sweeptype* as *absolute.*

### Relative

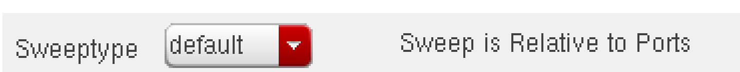
Puts the *Sweeptype* as *relative* in the Spectre netlist.




Specifying *Sweeptype* as *relative*, indicates to sweep relative to the analysis harmonics (rather than the PSS or QPSS fundamental).

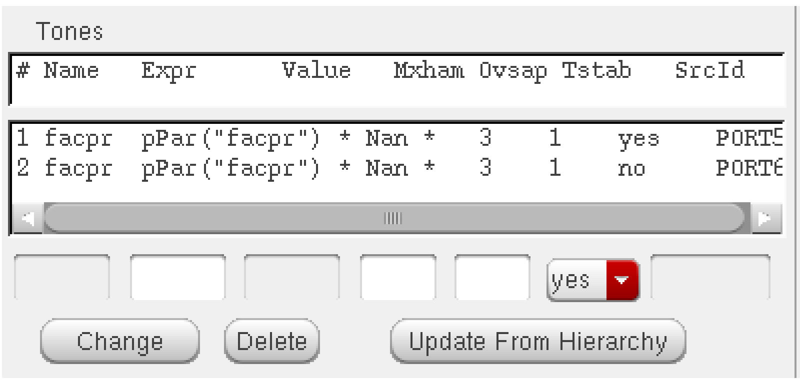
### Default

Does not put a *Sweeptype* parameter in the Spectre netlist. Sets the appropriate Spectre RF default depending on the circuit type.



## Tones (HB)

The *Tones* fields include the following list box, data entry fields, and data entry buttons.




### Tones List Box

The *Tones* list box displays information about every top-level tone in the circuit that has both a non-zero frequency or period value, and a non-zero amplitude value (absolute). The tones in the list box are arranged alphabetically by name.

To edit values for a tone, highlight the tone in the list box, then edit in the data entry fields.

For tones that are not at the top level of the schematic, you can manually create a tone entry by typing the pertinent information in the data entry fields.

For non-small-signal tones, each tone name and its correlated frequency value are used in the *Select from range* and *Array of coefficients* choices of the *Sidebands* field in the HBAC and HBnoise small-signal analysis forms.

### Fundamental Tones List Box Terms and Data Entry Fields

- **Name** – Displays the name assigned to the tone. This tone name must be entered into the pertinent Component Description Format (CDF) fields of each source in the schematic that has a tone. The CDF name field prompts are *First frequency name*, *Second frequency name, Frequency name*, and *Frequency name for 1/period*.
- **Expr** – Displays the value or expression representing the frequency of a particular tone. The expression can also be a user variable or it can contain user variables. If the frequency for the tone is specified as a variable, the *Expr* field displays the name of the variable. Otherwise, the field displays the numerical value of the frequency.

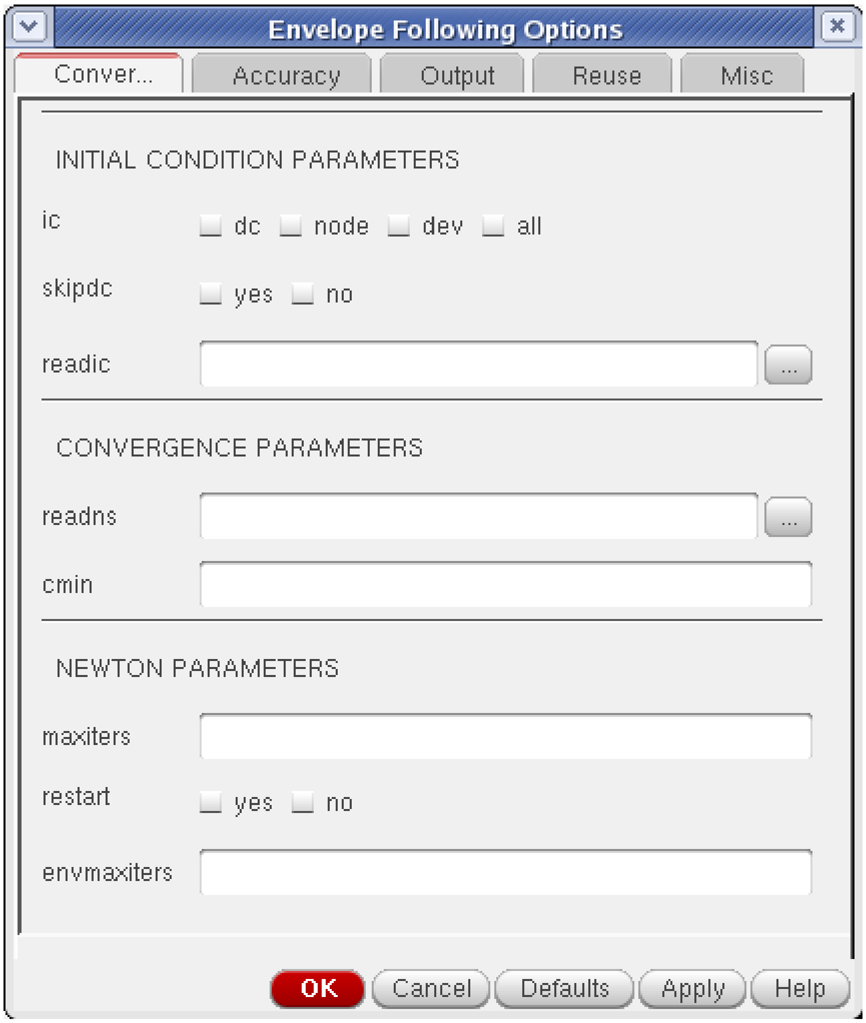
- **Value** – Displays the evaluated value of the *Expr* field using the current values of the user variables.
- **Mxham** – The maximum harmonic for the tone.
- **Ovsap** – A factor that determines the amount by which the tone is oversampled. Used to decrease aliasing effects.
- **Tstab** – A large signal (that could be used to run tstab).
- **SrcId** – Displays the instance name of the source in the schematic where the tone is declared.

## Options Forms

Each Options form lets you specify parameter values for a Spectre RF analysis. Options that are not relevant for a particular analysis do not appear on its Options form.

➤ On the Choosing Analyses form, click the *Options* button to open the Options form corresponding to the analysis type that is currently highlighted on the Choosing Analyses form.

For example, the following figure shows the top portion of the Options form for the Envelope analysis.




Use the Options form for an analysis to define its parameter values. Only those options that are relevant for a particular analysis are available on its Options form.

# Field Descriptions for the Options Forms

The following sections describe all the simulation parameters whose values you specify on Options forms. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the form.

## Accuracy Parameters (PSS, QPSS, ENVLP)

*Important*

In most cases, the `errpreset` parameter is the only accuracy parameter you should set.

Use the following links to locate detailed descriptions of how the `errpreset` parameter works to set accuracy parameters.

- The errpreset Parameter in PSS Analysis
- The errpreset Parameter in QPSS Analysis
- The errpreset Parameter in Envlp Analysis

**envlteratio** (ENVLP only) the ratio the simulator uses to compute envelope LTE tolerances for Envelope analysis. The default value is based on the accuracy default.

**fdharms** (PSS only) sets the number of harmonics considered for distributed (frequency-domain) components such as nport, delay, mtline, and delayed controlled sources. This parameter is supported only for the shooting engine.

**finitediff** (PSS and QPSS)

- For PSS analysis, uses the finite difference (FD) refinement method after the shooting method for driven circuits. The *finitediff* parameter refines the simulation results and is only meaningful when *highorder* is set to no. The possible settings are *no*, *yes* or *refine*.
- For QPSS analysis, uses the finite difference (FD) refinement method to refine the simulation results after the quasi-periodic shooting method. The possible settings are *no*, *yes* or *refine*.

  **no** turns off the finite difference refinement method.

  **yes** applies the finite difference refinement method to the PSS or QPSS analysis. The finite difference method tries to improve the initial small time steps if necessary.

  **refine** applies the finite difference refinement method to the PSS or QPSS analysis.

  For the PSS analysis the finite difference method tries to refine the time steps. When the simulation uses the *gear2* method, uniform 2nd order gear is used.

  When you use *readpss* and *writepss* to re-use the PSS analysis results for a driven circuit, *finitediff* automatically changes from *no* to *yes*. If you are concerned that the accuracy of your simulation might be affected by a loose *steadyratio*, you might want to try *finitediff*.

You might also set *finitediff* to *yes* to get more uniform time steps and reduce the noise floor. This does not always work. At lower power levels, the finite difference method might sometimes reduce the noise floor. We recommend that you reduce the noise floor by setting *highorder* to *yes*.

*Important*

Be aware of the following information for the finite difference method.

- The finite difference method works for driven circuits only. If you use the finite difference method with an oscillator circuit, it serves only as a loading routine. Even when a circuit or analysis parameter has changed, the old solution is loaded and may be inaccurate.
- For the QPSS analysis the finite difference method tries to refine the solution after the quasi-periodic shooting method.
- When you use *readqpss* and *writeqpss* to re-use the QPSS analysis results, *finitediff* automatically changes from *no* to *samegrid*.

**fullpssvec** (PSS only) Uses the full vector containing solutions at all PSS time steps in the linear solver. Default behavior is derived from the size of the equation and the property of the PSS time steps. Possible values are *no* or *yes*.

**highorder** (PSS only) executes high-order refinement after low order convergence when `errpreset` is either `moderate` or `conservative`. Uses the Multi-Interval Chebyshev (MIC) polynomial spectral algorithm. The *highorder* parameter works for both driven and autonomous circuits. The possible settings are *no* or *yes*. *yes* turns on the MIC method and tries harder to converge. *no* turns off the MIC method.

When you set `errpreset` to either `moderate` or `conservative` and you have not set *highorder*, MIC is used but it does not aggressively try to converge. MIC does try harder to converge when *highorder* is explicitly set to *yes*.

**inexactNewton** (PSS and QPSS) determines whether the inexact Newton method is used. The possible settings are *no* or *yes*.

**itres=1e$^{-4}$** (for the shooting engine), **itres=0.9** (for the HB engine) (PSS only) sets the relative tolerance for the linear solver.

**lnsolver** (PSS and QPSS) specifies the linear solver to be used.

**bicgstab** specifies that the biconjugate gradient stabilized (bicgstab) variant of the conjugate gradient (cg) solver is to be used. The bicgstab solver is formulated for nonsymmetric linear systems. The bicgstab solver uses less memory than either gmres or qmr but is the least robust of the solvers.

**gmres** specifies that the general minimum residual (gmres) linear solver is to be used. This is the default. If memory issues arise when using this solver, consider using either bicgstab or qmr, which use less memory.

**qmr** specifies that the quasi minimal residual (qmr) linear solver is to be used. The qmr solver uses less memory than gmres but is not as robust.

**resgmres** specifies that the restarted GMRES solver (resgmres) linear solver is to be used. The resgmres solver can reduce memory cost and simulation time.

**lteratio** is the ratio the simulator uses to compute LTE tolerances from the Newton tolerance. The default value is based on the accuracy default.

**maxacfreq** (not QPSS) is the maximum frequency used in a subsequent periodic small-signal analysis. This parameter automatically adjusts *maxstep* to reduce errors due to aliasing in frequency-domain results. The default is based on *maxstep* and *harms* values. See "Virtuoso Spectre Circuit Simulator RF Analysis Theory" for more information about specifying this parameter.

**maxorder** is the maximum order of the MIC polynomials used during waveform approximation. Values range from 2 to 16. The default value is 16 for driven circuits and 12 for autonomous circuits.

**maxperiods** is the maximum number of simulated periods allowed for the simulation to reach steady-state. Default is `50`.

**psaratio=1** is the ratio used to compute the MIC accuracy from the Newton tolerance.

**relref** is the reference used for relative convergence criteria. Your **Accuracy Defaults** choice sets the default values.

**allglobal** is the same as sigglobal except that it also compares the residues for each node to the historical maximum.

**alllocal** compares the relative error at each node to the largest values ever found for that node.

**pointlocal** compares the relative errors in quantities at each node to that node alone.

**sigglobal** compares relative errors in each signal to the maximum value for all signals.

**resgmrescycle** specifies the length of the computation cycle and determines whether the recycling feature of the resgmres solver is used.

- The *instant*, *short*, and *long* values are most appropriate when memory is the primary concern.

- The *recycleinstant*, *recycleshort*, and *recyclelong* values are most appropriate when simulation time is the major concern. Circuits where neighboring frequency points are close benefit most from using the three recycle values. The recycle values are not supported for large signal analyses.
- For large signal analyses, Cadence recommends using the resgmres solver only for large-signal single-tone analyses such as PSS. For the best performance in large signal analyses, try the gmres solver first and then, if necessary, try the resgmres solver.

Possible values are

**instant** uses the shortest computation cycle, which means the least amount of memory is used. Use this value if the circuit is almost linear or is only weakly non-linear.

**short** uses a mid-length computation cycle, which requires more memory than the shortest computation cycle. This is the default value.

**long** uses the longest computation cycle, which requires the most memory. Use this value if the circuit is strongly non-linear.

**recycleinstant** turns on the recycling features of the solver, uses the shortest computation cycle, and uses the least amount of memory. Use this value if the circuit is almost linear or is only weakly non-linear. This value is not supported for large signal analyses.

**recycleshort** turns on the recycling features of the solver, uses a mid-length computation cycle, and uses more memory than the shortest computation cycle. This value is not supported for large signal analyses.

**recyclelong** turns on the recycling features of the solver, uses the longest computation cycle, and requires the most memory. Use this value if the circuit is strongly non-linear. This value is not supported for large signal analyses.

**steadyratio** is the ratio the simulator uses to compute steady-state tolerances from the LTE tolerance. This parameter adjusts the maximum allowed mismatch in node voltages and current branches during the steady-state period. The default is based on the accuracy default.

## Additional Parameters (All)

**additionalParams** provides a place where you can enter parameters that are not supported in the graphical user interface.

## Annotation Parameters (All)

**annotate** lets you specify what information is printed at the beginning of the output to identify the results. Default is `status`. Choices are no, `title`, `sweep`, `status` and `steps`.

**stats** tells the simulator to generate analysis statistics. Default is no.

## Convergence Parameters (All)

**augmented** specifies if the frequency aware PPV method is used to calculate the total noise of the oscillator. Possible values are no, yes, pmonly and amonly.

**cmin** is the minimum capacitance from each node to ground. Default is 0.

**gear_order** is the order used for Gear-type interpolations. Default is 2 (second order).

**lnsolver** (PAC) specifies the linear solver to be used.

> **bicgstab** specifies that the biconjugate gradient stabilized (bicgstab) variant of the conjugate gradient (cg) solver is to be used. The bicgstab solver is formulated for nonsymmetric linear systems. The bicgstab solver uses less memory than either gmres or qmr but is the least robust of the solvers.
>
> **gmres** specifies that the general minimum residual (gmres) linear solver is to be used. This is the default. If memory issues arise when using this solver, consider using either bicgstab or qmr, which use less memory.
>
> **qmr** specifies that the quasi minimal residual (qmr) linear solver is to be used. The qmr solver uses less memory than gmres but is not as robust.
>
> **resgmres** specifies that the restarted GMRES solver (resgmres) linear solver is to be used. The resgmres solver can reduce memory cost and simulation time.

**oscsolver** lets you specify the type of solver to use for an oscillator circuit. Possible values are `std`, `turbo` or `ira`. The default is `turbo`.

> **ira** uses the implicitly restarted Arnoldi algorithm to calculate the dominant eigenvalue and the corresponding eigenvector for small-signal analysis of oscillator circuits. `Ira` uses less memory than `turbo`.
>
> **std** uses a full eigen analysis to calculate the dominant eigenvalue and the corresponding eigenvector. Using the `std` solver for small-signal analysis of oscillator circuits is slower but more robust than simulation using the `turbo` setting. Use the `std` setting if simulation with the `turbo` setting is unsuccessful.

**turbo** uses the Arnoldi algorithm based on the Krylov subspace to calculate the dominant eigenvalue and the corresponding eigenvector for oscillator small-signal analysis. `Turbo` is significantly faster than `std` and uses significantly less memory. In rare situations `turbo` might be *less* accurate.

**readns** lets you specify the name of a file that contains an estimate of the initial transient solution. Enter the complete path to the file. No default.

**resgmrescycle** specifies the length of the computation cycle and determines whether the recycling feature of the resgmres solver is used. For more information, see resgmrescycle.

**solver** lets you specify the type of solver to use for a linear system. Possible values are `std` or `turbo`. The default is `turbo`.

**std** for each frequency value, solves the linear system using the full GMRES algorithm. Simulations using the `std` solver are slower but more robust than simulations using the turbo setting. Use the `std` setting if simulation with the `turbo` setting is unsuccessful.

**turbo** for each frequency value, solves the linear system using the recycled Krylov subspace algorithm. Using `turbo` is significantly faster than using `std`. In rare situations `turbo` might be *less* accurate.

**tolerance** is the relative tolerance for the linear solver when solving for convergence. Default is $10^{-9}$.

## Harmonic Balance Parameters (HB)

**maxperiods** is the maximum number of simulated periods allowed for the simulation to reach steady-state. Default is `50`.

**itres** sets the relative tolerance for the linear solver. Default is 0.9.

**freqdivide** sets the frequency division ratio of a large signal. The default is 1.

## Initial Condition Parameters (PSS, QPSS, ENVLP, HB)

**ic** specifies the methods used to set the initial condition (`dc`, `node`, `dev` and `all`). For an explanation of each of these methods, consult the *Virtuoso Spectre Reference* manual. Default is all.

**readic** lets you specify the name of the file that contains the initial conditions.

**skipdc**

**no calculates** the initial solution using the normal DC analysis. This is the default.

**yes** omits the DC analysis from the transient analysis and gives the Initial solution either in the file specified by the *readic* parameter or by the values specified on the *ic* statements.

**sigrampup** independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. Enables waveform production in the time-varying independent source after the rampup phase.

- The rampup simulation is from *tstart* to time=0 s.
- The main simulation is from time=0 s to *tstab*.
- If you do not specify the *tstart* parameter, the default *tstart* time is set to -0.1 multiplied by *tstab*.

## Integration Method Parameters (PSS, QPSS, ENVLP, HB)

**envmethod** (ENVLP) specifies the integration method for the ENVLP analysis. Your accuracy default choice sets the default value. The possible settings are

**euler** is backward Euler.

**gear2** is the backward Euler and second-order Gear methods.

**gear2only** is Gear's second-order backward difference method only.

**trap** is the backward Euler and trapezoidal methods.

**trapgear2** is the backward Euler, trapezoidal and second-order Gear methods.

**traponly** is the trapezoidal rule only.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

**method** (PSS, QPSS and ENVLP) specifies the integration method. Your accuracy default choice sets the default value. The possible settings are

**euler** is backward Euler.

**gear2** is the backward Euler and second-order Gear methods.

**gear2only** is Gear's second-order backward difference method only.

**trap** is the backward Euler and trapezoidal methods.

**traponly** is the trapezoidal rule only.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

Use the *gear_order* option on the small-signal analyses Options form to set the order of a Gear-type interpolation.

**tstabmethod** (PSS) specifies the tstab integration method. Your accuracy default choice sets the default value. The possible settings are

**euler** is backward Euler.

**gear2** is the backward Euler and second-order Gear methods.

**gear2only** is Gear's second-order backward difference method only.

**trap** is the backward Euler and trapezoidal methods.

**traponly** is the trapezoidal rule only.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

Use the *gear_order* option on the small-signal analyses Options form to set the order of a Gear-type interpolation.

## Multitone Stabilization Parameter (QPSS)

**stabcycles** specifies the number of stabilization cycles to perform when both large and moderate sources are enabled. The default is 2.

## Newton Parameters (PSS, QPSS, ENVLP)

**envmaxiters** specifies the maximum number of Newton iterations per envelope step for the ENVLP analysis. The default is 3.

**maxiters** is the maximum number of iterations per time step.

**restart** tells the simulator not to use the previous DC solution as an initial guess. Default is yes (means do not use).

## Output Parameters (All)

**compression** directs the simulator to perform data compression on the output. Default is *no.*

**cyclo2txtfile** outputs cyclo-stationary noise to a text file as an input source for a succeeding stage.

**enable osc ppv** is equivalent to *save osc ppv*.

**freqaxis** specifies what version of the frequency to plot the output against in spectral plots.

- For the PAC and QPAC analysis

  *absout* is the absolute value of the output frequency.
  *in* is the input frequency.
  *out* is the output frequency.

- For the PXF and QPXF analysis

  *absin* is the absolute value of the input frequency.
  *in* is the input frequency.
  *out* is the output frequency.

- For the PSP and QPSP analysis

  *absin* is the absolute value of the frequency swept at the input.
  *in* is the scattered frequency at the input.
  *out* is the scattered frequency at the output.

**nestlvl** specifies the levels of subcircuits to output. The field is activated by choosing the *lvl* or *lvlpub* values for the *save* field.

**oppoint** specifies whether the simulator outputs the operating point information. You can send the information to a rawfile, the logfile, or the screen. Default is *no*.

**outputperiod** lets you specify the time-domain output period. The time-domain small-signal response is computed for the period specified, rounded to the nearest integer multiple of the PSS analysis period.

**outputtype** lets you specify the output type for ENVLP Analysis. Possible values are *both*, *envelope*, or *spectrum*. The default is *both*.

**save** tells the simulator what signals to save. You have the following choices:

**all** saves all signals.

**allpub** saves only signals that are normally useful. Normally useful signals include shared node voltages and currents through voltage sources and iprobes.

**lvl** saves all signals up to *nestlvl* deep in the subcircuit hierarchy. *lvl* is relevant for subcircuits. Click *lvl* to activate the *nestlvl* type-in field. Then enter a value for the *nestlvl* parameter.

**lvlpub** saves all normally useful signals up to *nestlvl* deep in the subcircuit hierarchy. *lvlpub* is equivalent to *allpub* for subcircuits. Normally useful signals include shared node voltages and currents through voltage sources and iprobes. Click *lvlpub* to activate the *nestlvl* type-in field. Then enter a value for the *nestlvl* parameter.

**selected** saves only the signals you request on the *Outputs* menu in the ADE window. This is the default setting.

Use *lvl* or *all* (instead of *lvlpub* or *allpub*) to include internal node voltages and currents through other components that compute current.

Use *lvlpub* or *allpub* to exclude signals at internal nodes on devices (the internal collector, base, emitter on a BJT, the internal drain and source on a FET, and so on). *lvlpub* and *allpub* also exclude the currents through inductors, controlled sources, transmission lines, transformers, etc.

**saveallsidebands** lets you save all sidebands for Pnoise and Qpnoise analyses.

**save osc ppv** lets you save a perturbation projection vector (PPV) file. A PPV can be thought of as representing the oscillator's phase sensitivity to perturbations in the voltage or current at the nodes of the oscillator. The saved PPV file is used in phase noise calculations, such as when you prepare to generate a VCO macromodel.

The PPV file is saved into the raw directory after simulation. For example, after running an autonomous circuit named `vco.ckt` with the *save osc ppv* parameter set to *yes*, the PPV file is placed in the `vco.raw` directory. The PPV file is named *`analysisID`*`.td.ppv.pss`.

If the *save osc ppv* parameter is set to *yes* and the PNOISE analysis *enable osc ppv* option is set to *yes* also, only the PPV file for the PNOISE analysis is saved.

**skipcount** specifies how many points to skip before saving a point. No default.

**skipstart** specifies when the simulator starts skipping output data. Default is *starttime s* (seconds).

**skipstop** specifies when the simulator stops skipping output data. Default is *stoptime s* (seconds).

**stimuli** specifies what *PXF* and *QPXF* uses for inputs to the transfer functions.

> **nodes_and_terminals** specifies that all possible transfer functions are computed.
>
> **sources** specifies that the sources present in the circuit are used as the inputs to the transfer functions.

**strobedelay** is the delay (phase shift) between the skipstart time and the first strobe point. Default is 0.

**strobeperiod** is the output strobe interval in seconds. The actual strobe interval is rounded to an integer multiple of the clock period.

Strobing is crucial to getting the very low noise floors required for ACPR measurements. Some users require noise floors 70 to 80 dB below the peak, which is not possible if the Fourier analysis has to interpolate between unevenly spaced data points.The *strobeperiod* option forces the output envelope to have evenly spaced data points. The subsequent Fourier analysis can proceed without interpolation.

The PPV file is saved into the raw directory after simulation. For example, after running an autonomous circuit named `vco.ckt` with the *Save osc PPV* option set to *yes*, the PPV file is placed in the `vco.raw` directory. The PPV file is named *`analysisID`*`.td.ppv.pss`.

If the *Save osc PPV* option is set to *yes* and the PNOISE analysis *use ppv for osc* option is set to *yes* also, only the PPV file for the PNOISE analysis is saved.

## Simulation Bandwidth Parameters (ENVLP)

**modulationbw** specifies the modulation bandwidth.

## Simulation Interval Parameters (ENVLP, PSS, QPSS)

**outputstart** (ENVLP) specifies the timepoint when the simulator starts to save output.

**start** (ENVLP) specifies the analysis start time.

**tstab** (ENVLP) specifies the initial stabilization time. Default is `0`.

**tstart** (PSS and QPSS) is the start time you specify for transient analysis. It can be negative or positive. Default is `0`.

## State File Parameters (ENVLP, PSS, QPSS)

**checkpss** Options are yes and no.

**readpss** specifies the file from which the steady-state solution is read. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

**readqpss** specifies the file from which the steady-state solution is read. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

**recover** (PSS) specifies the tstab analysis states file to be restored. The saveperiod, savetime, savefile, and recover parameters are used to save and restart the tstab part of the PSS, QPSS, and ENVLP analyses. This feature provides functionality similar to that of the save and restart feature of transient analysis.

**saveclock** (PSS) saves the tran analysis periodically at specified wall clock times.

**savefile** (PSS) saves the tstab part of tran analysis states into the specified file.

**saveperiod** (PSS) saves the tstab part of tran analysis states periodically at specified simulation times.

**savetime** (PSS) saves the tstab part of tran analysis states into files at the specified time points. The savetime parameter takes precedence over the saveperiod parameter when both are specified.

**swapfile** is a temporary file that holds steady-state information. If you enter a filename, the simulator stores the operating point in that file rather than in virtual memory. Use this option if you receive a warning about not having enough memory to complete the analysis. Enter the complete path to the file. This parameter is supported for only the shooting engine.

**write** lets you specify the name of the file to which the Spectre RF simulation writes the initial transient solution.

**writefinal** lets you specify the name of the file to which the Spectre RF simulation writes the final transient solution.

**writepss** specifies the file to which the steady-state solution is written. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

**writeqpss** specifies the file to which the steady-state solution is written. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

## Time Step Parameters (ENVLP, PSS, QPSS)

**envmaxstep** specifies the maximum outer envelope size. The default is set by the accuracy default.

**fixstepsize** (ENVLP) fixes the envelope step size for speeding up an ENVLP analysis. The default value is no. If you select Yes, you can fix either the Step Size or Step Period.

**liberal** = 0.1/max AC frequency

**moderate** = 2 x liberal

**conservative** = 4 x liberal

**maxacfreq** (QPSS shooting engine only) specifies the maximum frequency requested in a subsequent periodic small-signal analysis. The default is derived from *Accuracy Defaults (errpreset)* and *Harms*.

**maxstep** is the largest allowable time step. The default is set by the *Accuracy Defaults (errpreset)*.

**step** is the smallest simulator time step used to improve the look of the results. Default is 0.001 x fundamental period seconds.

**stepsize** (ENVLP) specifies the number of cycles skipped for each step when *fixstepsize* is set to *yes*.

The interval (in seconds of envelope following time) between two steps, when *fixstepsize* is set to yes, must be greater than the clock period. For autonomous, fm or shooting envelop, this is rounded to the nearest integer multiple of the clock period.

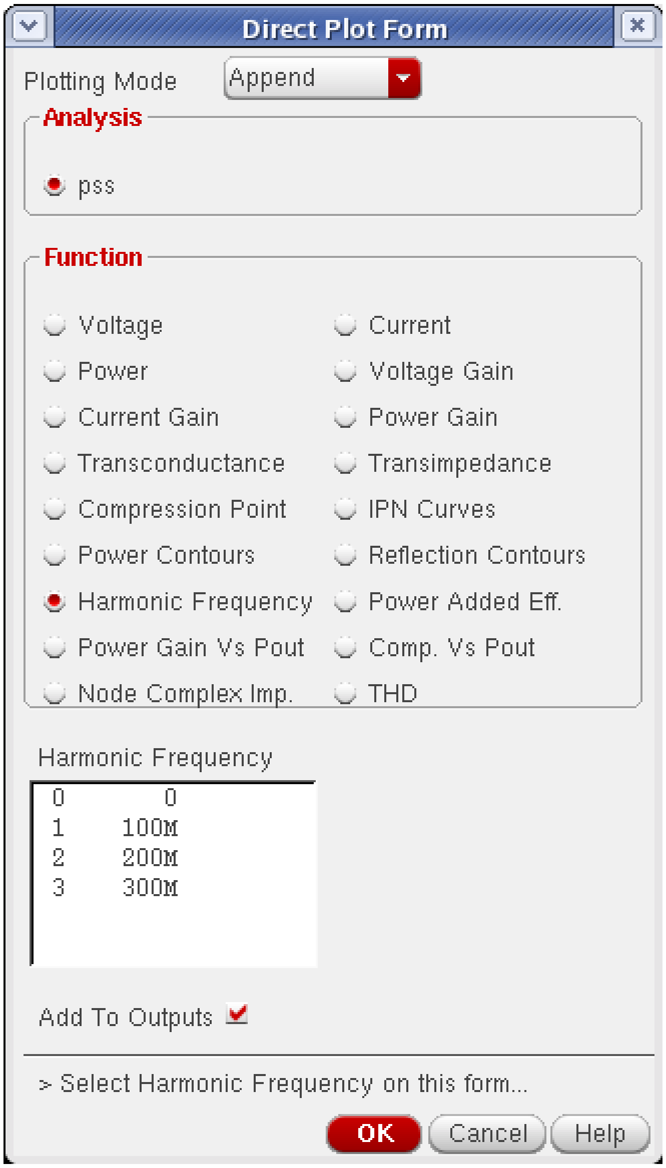
# Direct Plot Form

Use the *Direct Plot* command in the ADE window to plot most RF simulation results.

## Opening the Direct Plot Form

Choose the *Results – Direct Plot – Main Form* command in the ADE window, to access results for the most recently performed analyses.The Direct Plot form is similar to the one shown in the figure below.

#### Figure 1-2 Direct Plot Form




The *Analysis* section lists one or more analyses with available data.

For periodic large and small-signal analyses, choose *pss* to access the results from the initial PSS analysis. Choose *pac*, *pnoise*, *pxf*, or *psp* to access the results of any available periodic small-signal analyses that ran after the PSS analysis.

For quasi-periodic large and small-signal analyses, choose *qpss* to access the results from the initial QPSS analysis. Choose *qpac*, *qpnoise*, *qpxf*, or *qpsp* to access the results of any available quasi-periodic small-signal analyses that ran after the QPSS analysis.

Choose *Envlp* to access the results from an envelope analysis.

## Defining Measurements in a Plot Form

You define the RF simulation measurements to display by entering and selecting items in the Direct Plot form such as functions, plots, and modifiers and selecting nets, terminals, or other objects on the schematic.

While making selections in the Direct Plot form, follow the messages at the bottom of the form for instructions and prompts. For example,

```
> Select Net on schematic...
```

and

```
> To plot, press Sij-button on this form...
```

When you click the plot button (or perform another specified action), a simulation plot appears, by default, in the current waveform window. If the waveform window or Schematic window is not open, selecting a direct plot function automatically opens both windows.

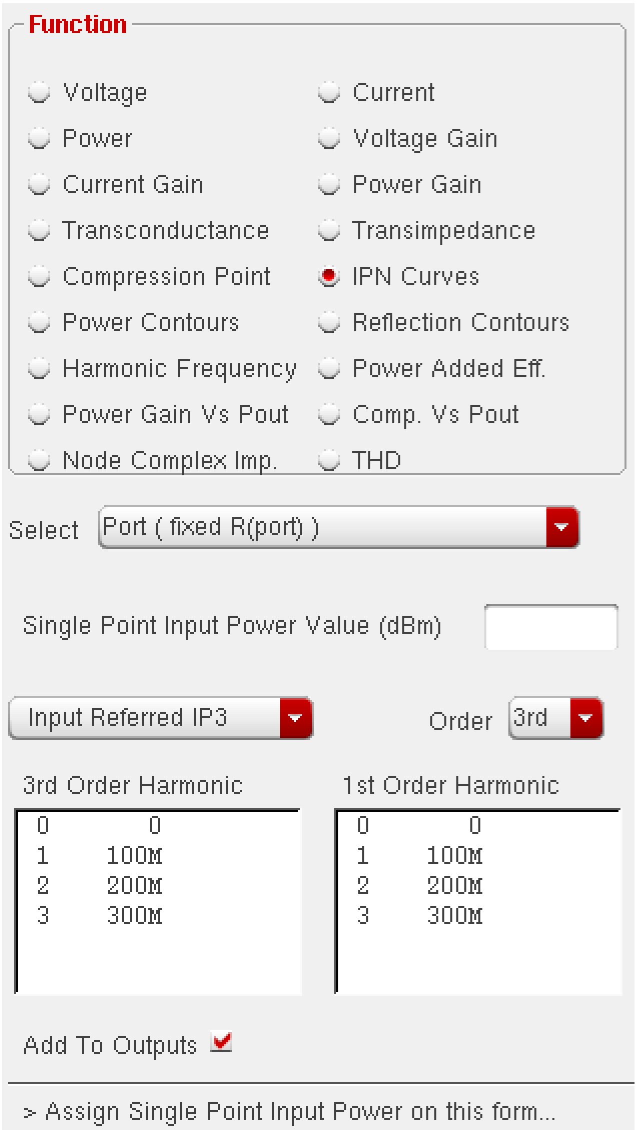
Informative messages often also display at the bottom of the waveform and Schematic windows.

## Plotting Data for Swept Simulations

For swept analyses, the last section in the Direct Plot form includes one or more list boxes that display different values of the design variable you selected when setting up the swept analysis. Figure 1-3 on page 106, the Direct Plot form for an IP3 measurement, has two such list boxes. Following the messages at the bottom of the Direct Plot form, in the list box, click the design variable values you want to plot. The name and type of the variable are displayed at the top of the list box.

For some measurements, for example, second and third order intercept functions, the Direct Plot form displays a second list of variable values for the reference variable. The following example shows the Direct Plot form for 3rd order IPN curves.

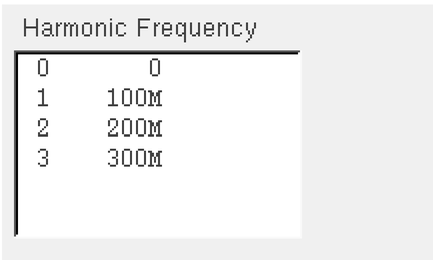
### Figure 1-3 Direct Plot form for IP3 Curve




Follow the messages at the bottom of the Direct Plot form and click the design variable values you want to plot.

## Selecting Sidebands and Harmonics

When Sidebands or Harmonics are available, they are displayed in one or more list boxes at the bottom of the Direct Plot form.




As before, follow the messages at the bottom of the Direct Plot form and click sideband or harmonic values in the list box to select them.

To select one value, simply click it.

- To select two or more adjacent values, click and drag with the mouse over the values you want to select.
- To select two or more values that are not adjacent, hold the `Control` key down while you click the individual values.
- To deselect a value, hold the `Control` key down while you click a selected value.

## Generating a Spectral Plot

Spectral plots show the function level at each frequency component of a single analysis point (one step in a sweep of frequency, input power, or design variable).

1. In the ADE window, choose *Results – Direct Plot – Main Form* to open the Direct Plot form, the waveform window, and the design in the Schematic window.
2. Follow the prompts displayed at the bottom of the Direct Plot form for instructions. Sometimes additional information is also displayed elsewhere in the form.
3. In the Direct Plot form, click a *Plot Mode* button to specify whether the curves you plot are appended to or replace any existing curves in the waveform window.
4. Click an *Analysis* button to select the simulation for which you want to plot results.
5. Click a *Function* button to select the function or measurement you want to plot.

   The *Function* values displayed vary with the simulations run. If you need a function that is not included on the Direct Plot form, use the RF version of the analog design environment calculator. See *Virtuoso Visualization and Analysis Tool User Guide* for more information.

6. In the *Select* cyclic field, determine what to select in the Schematic window.

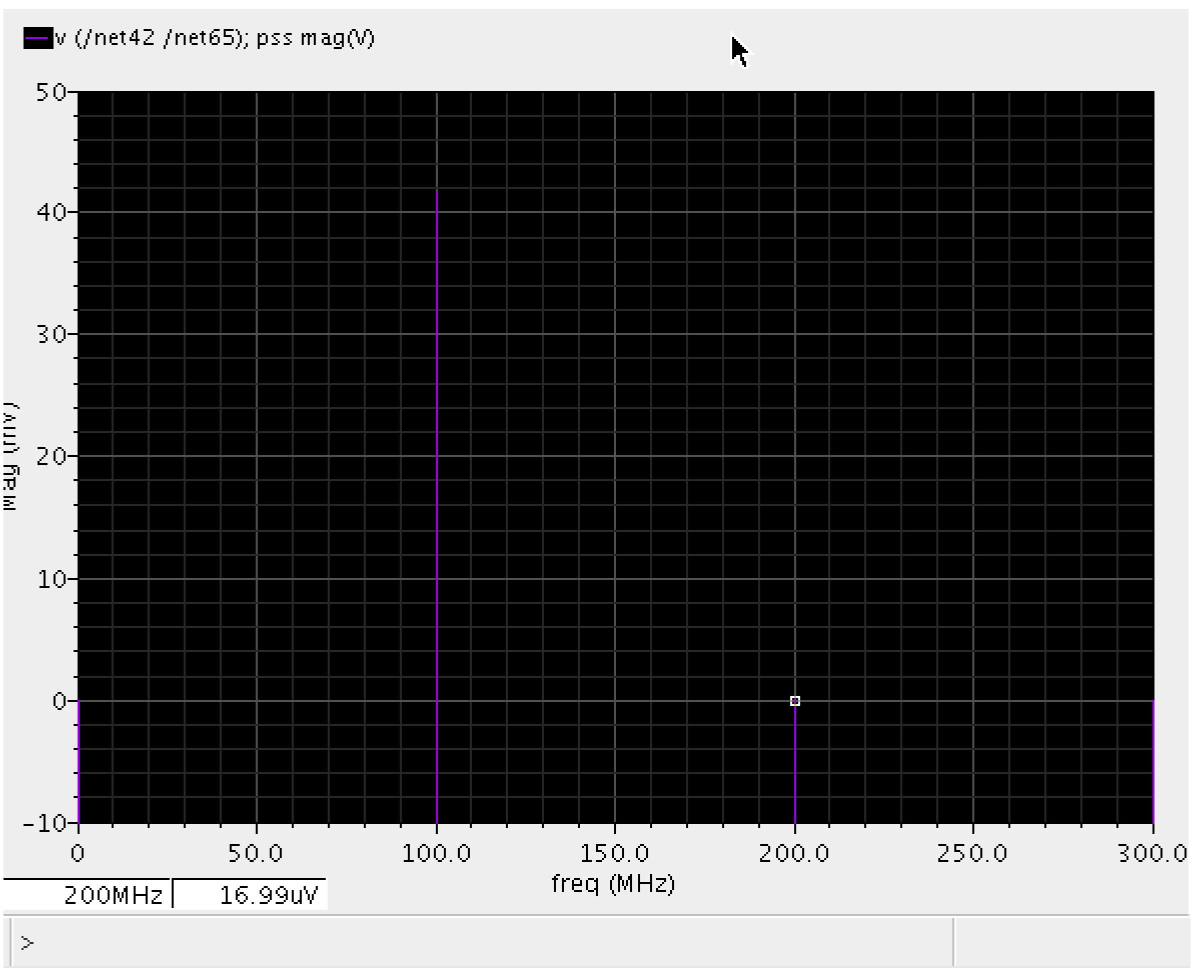
   All *Function* values have several *Select* options. For example, for the *Voltage* function, you can now choose between *Net*, *Differential Nets*, and *Instance with 2 Terminals*. Follow the prompts displayed at the bottom of the Direct Plot form for more information.

7. For PSS sweeps, set *Sweep* to *spectrum*.

8. For some RF functions, set *Signal Level*. The choices are *peak* or *rms*.

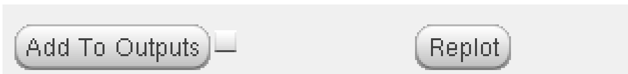
9. For some RF functions, set a plot *Modifier*. The choices are *Magnitude*, *Phase*, *dB20*, *Real*, or *Imaginary.*

10. Follow the prompts at the bottom of the Direct Plot form to draw the curve in the waveform window.



## Saving a Displayed Output and Displaying Saved Outputs.

In the Plot form, first click the *Add to Outputs* button and then click the *Replot* button to add a plot displayed in the waveform window to the *Outputs* list in the ADE window.




- When you select a curve to plot from the list of curves in the *Outputs* section of the ADE window, the new expression is *always* plotted as the *Magnitude* of the signal, not the plot with your original *Modifier* selection. If you want a specific scale, for example *dB20*, you have two choices. You can either
  - ❑ Modify the expression using *Curve – Edit – Scale* in the waveform window.
  - ❑ Paste the expression from the *Outputs Setup* form into the calculator buffer and edit it.

## Changing the Noise Floor of a Spectral Plot

➤ Choose the *Axes – Y axis* command in the waveform window to change the noise floor of a spectral plot. In the Axes – Y axis form, set *Range* to *Min-Max* and specify a *Min* value.

## Generating a Time Waveform

Time waveforms plot voltage and current against time.

1. Choose *Results – Direct Plot – Main Form* to open the waveform window, the design in the Schematic window, and the Direct Plot form.
2. Follow the prompts displayed at the bottom of the Direct Plot form for information on what to do next. Sometimes additional information is also displayed elsewhere in the form.
3. In the Direct Plot form, click a *Plot Mode* button to specify whether the curves you plot are appended to or replace any existing curves in the waveform window.
4. Click an *Analysis* button to select the simulation for which you want to plot results.
5. Click a *Function* button to select the function or measurement you want to plot.

   The *Function* values displayed vary with the simulations run. If you need a function that is not included on the Direct Plot form, use the RF version of the analog design environment calculator. See *Virtuoso Visualization and Analysis Tool User Guide* for more information.

6. In the *Select* cyclic field, determine what to select in the Schematic window.

   All *Function* values have several *Select* options. For example, for the *Voltage* function, you can now choose between *Net*, *Differential Nets* and *Instance with 2 Terminals*. Follow the prompts displayed at the bottom of the Direct Plot form for more information.

7. For PSS sweeps, set *Sweep* to *time*.

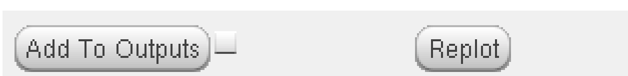
8. For some RF functions, set *Signal Level.* The choices are *peak* or *rms*.

9. For some RF functions, set a plot *Modifier*. The choices are *Magnitude*, *Phase*, *dB20*, *Real*, or *Imaginary*.

10. Follow the prompts at the bottom of the Direct Plot form to draw the curve in the waveform window.

## Saving a Displayed Output and Displaying Saved Outputs.

In the Plot form, first click the *Add to Outputs* button and then click the *Replot* button to add a plot displayed in the waveform window to the *Outputs* list in the ADE window.




When you select a curve to plot from the list of curves in the *Outputs* section of the ADE window, the new expression is *always* plotted as the *Magnitude* of the signal, not the plot with your original *Modifier* selection. If you want a specific scale, for example *dB20*, you have two choices. You can either

- Modify the expression using *Curve – Edit – Scale* in the waveform window.
- Paste the expression from the *Outputs Setup* form into the calculator buffer and edit it.

## Plotting Complex Impedance

Spectre RF does not provide a GUI to plot the complex impedance of a node in a Smith Chart. Follow this procedure to create a node complex impedance Smith Chart plot.

To produce the data used to create the transimpedance expression used in step , perform a swept PSS analysis and set the Direct Plot Sweep to `variable`. Otherwise, the harmonic call in the numerator of the transimpedance expression does not appear and you have to manually add this call in step 1.

Select the net that connects to the terminal of interest. Then set the input and output harmonic on the Direct Plot form to the same index.

Use the Direct Plot form to send the transimpedance expression to the ADE Outputs window.

Your transimpedance expression should look similar to the following.

```
(harmonic(v("/net1" ?result "pss_fd") '(1)) /
 :harmonic(i("/I8/out" ?result "pss_fd") '(1)))
```

1. Use a `vi` window (or the calculator buffer) along with cutting and pasting from the ADE Setting Outputs form to construct the final gamma expression.

2. (
   transimpedance

   ```
   - 50.) / (
   transimpedance
   + 50.)
   ```

   Your final gamma expression looks like this.

   ```
   (
    (harmonic(v("/net1" ?result "pss_fd") '(1)) /
    harmonic(i("/I8/out" ?result "pss_fd") '(1)))
    - 50.) / (
    (harmonic(v("/net1" ?result "pss_fd") '(1)) /
    harmonic(i("/I8/out" ?result "pss_fd") '(1)))
    + 50.)
   ```

3. Add this new expression to the ADE Outputs form.

4. Click *Plot Outputs* to plot the outputs.

5. In the waveform window, choose *Axes – To Smith – Impedance* to create the desired node complex impedance Smith Chart plot.

6. Use ADE *Save State* to save your expression.

# Field Descriptions for the Direct Plot Form

The following sections describe the fields on the Direct Plot form. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the forms.

## 1st Order Harmonic

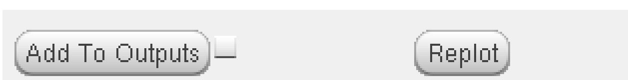
See First-Order Harmonic.

### 2nd-7th Order Harmonic

See Nth Order Harmonic.

## Add To Outputs

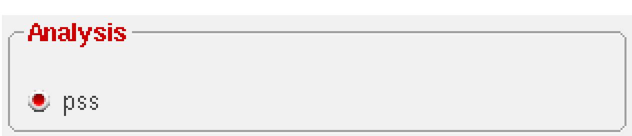
Adds the expression plotted in the waveform window to the *Outputs* list in the ADE window.




- Highlight *Add To Outputs* to automatically add each new plot to the *Outputs* list whenever you click *Replot*.
- With *Add To Outputs* dehighlighted, click the *Add To Outputs* button to add only the current plot to the *Outputs* list.

## Analysis

Selects the analysis to plot results for.



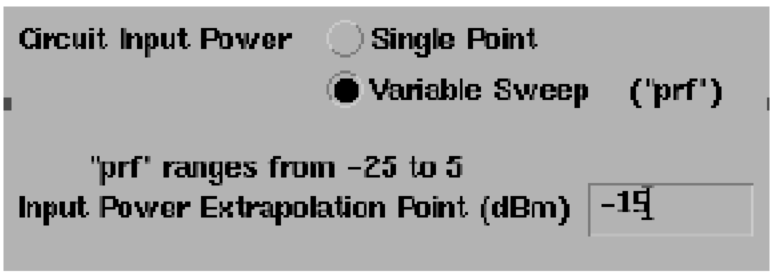

The *Analysis* area lists analyses for which results are available. If you performed one or more small-signal analysis, each small-signal analysis has its own button separate from the button for the large-signal analysis performed as a prerequisite.

When noise separation information has been generated, a *pnoise separation* choice becomes available. This choice turns on the ability to plot the noise contributions of selected sidebands and objects. For more information, see “Function” on page 116.

Highlighting an analysis changes the Direct Plot form to display fields for that analysis.

## Circuit Input Power (QPSS, PAC)

Selects between *Single Point* and an appropriate *Sweep*.



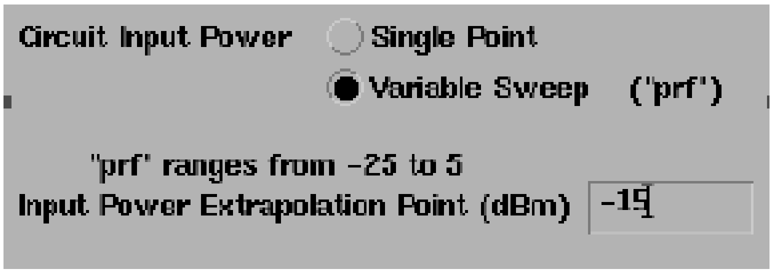

- If you choose *Single Point*, the *Input Power Value (dBm)* field appears, and you must type a value into it.
- If you choose a *Sweep*, the *Extrapolation Point* field appears, and you can type a value. Placing a value in the *Extrapolation Point* field is optional.

## Close Contours (PSS, ENVLP)

See “Maximum Reflection Magnitude (ENVLP)” on page 122.

## Extrapolation Point (PSS, QPSS)

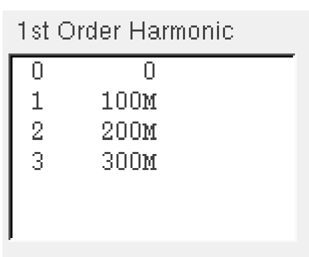
Specifies the value of the design variable from which the straight line approximations of first- and third-order harmonics are produced. See "Circuit Input Power (QPSS, PAC)" on page 113 for more information.




The *Extrapolation Point* field is optional. It specifies the value of the design variable from which the straight line approximations of first- and third-order harmonics are produced. If you do not specify a value, the default value is the smallest x axis sweep value.

## First-Order Harmonic (PSS)

Lists available first-order harmonics when you select the *IPN Curves Function* for a PSS analysis.

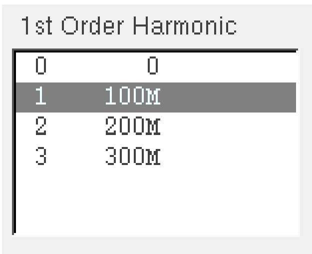



Lists the harmonics available for plotting by number and associated frequency values.

Select one harmonic from the list box and then select the appropriate net on the schematic.

## First Order Harmonic

Lists available first-order harmonics when you select the *IPN Curves Function* for an analysis.

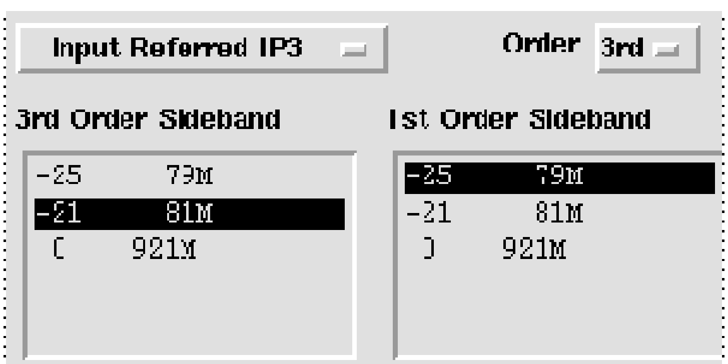



In the list box, the first column lists the frequency value of a harmonic, the following columns list the tone coefficient for each fundamental tone that contributed to the harmonic.

Select one harmonic from the list box and then select the appropriate net on the schematic.

## First Order Sideband (PAC)

Lists available first-order sidebands when you select the *IPN Curves Function* for a PAC analysis.

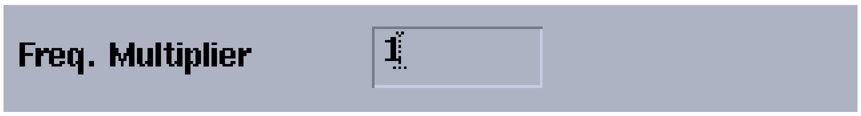



Lists the first-order sidebands available for plotting by number and associated frequency values.

Select one sideband from the list box and then select the appropriate net on the schematic.

## Freq. Multiplier (Pnoise)

Specifies the ratio of the jitter output signal frequency to the PSS fundamental frequency.

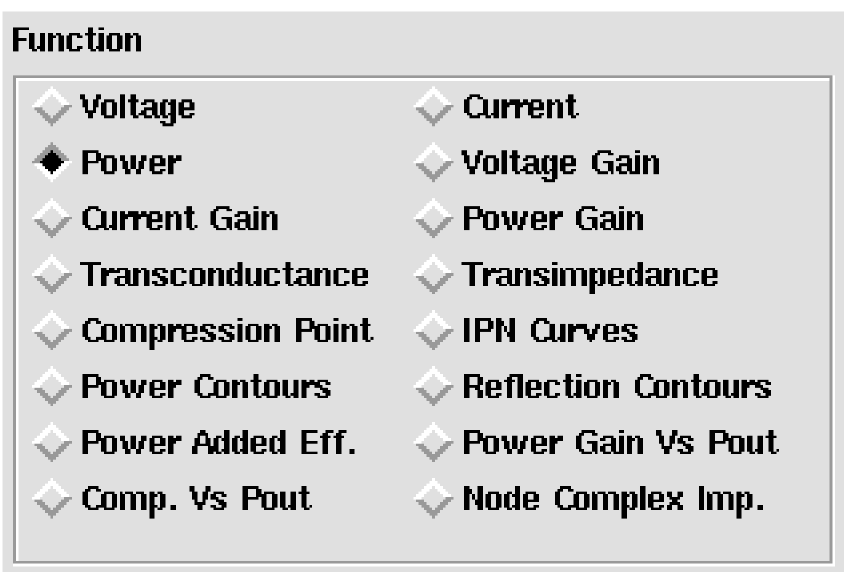



Set this parameter to an integer greater than 1 only when the selected output frequency for measuring jitter is not the same as the PSS fundamental frequency.

For example, if you have an oscillator with a counter and you measure the jitter at the output of the oscillator where the frequency is a harmonic of the fundamental (rather than measuring at the output of the counter where the frequency is the same as the PSS fundamental), then set the *Freq. Multiplier* value to the number of the harmonic. So, if the measured output is the second harmonic of the fundamental, you set *Freq. Multiplier* to 2.

## Function

Specifies a quantity to plot.



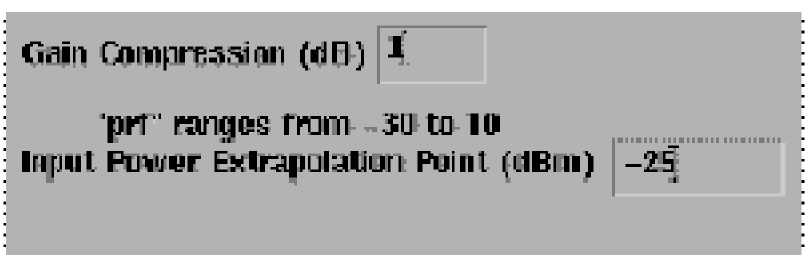

Each *Function* button specifies a different quantity that you can plot. The available *Functions* vary depending on the *Analysis* you select. For some functions, you must select one or two objects on the schematic after selecting the *Function* button.

When the chosen *Analysis* value is *pnoise separation*, the following become available in the Function pane:

| | |
|---|---|
| *Sideband Output* | Plots the noise contribution of selected sidebands. |
| *Source Output* | Plots the noise contribution of primary noise sources such as re and rb in a BJT to the output at one selected sideband. |
| *Primary Source* | Plots the primary noise sources such as re and rb in a BJT at one selected sideband. |
| *Instance Output* | Plots the noise contribution to the output of instances, such as MOS and BJT, of a selected sideband. |
| *Instance Source* | Plots the noise sources of some instances at one selected sideband. |
| *Src. Noise Gain* | Plots the noise gains of primary noise sources such as re and rb in a BJT from source to output at one selected sideband. |

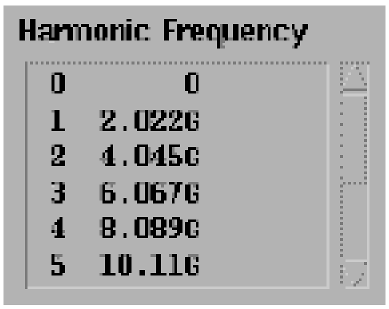
## Gain Compression (PSS, QPSS)

Specifies the *Gain Compression* when you plot the *Compression Point*.



## Harmonic Frequency (PSS)

Lists available harmonic frequencies s by number when you select the *Harmonic Frequency* function.

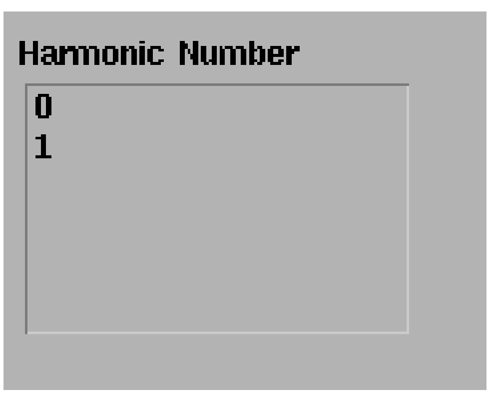



Lists the harmonics available for plotting by number and associated frequency values.

Select one harmonic from the list box and then select the appropriate net on the schematic.

## Harmonic Number (ENVLP)

Lists available harmonics by number when you select the *Power Function* for an ENVLP analysis.

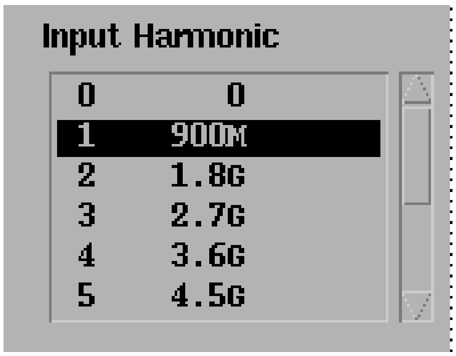



Lists the harmonics available for plotting by number and associated frequency values.

Select one harmonic from the list box and then select the appropriate net on the schematic.

## Input Harmonic (PSS)

Lists available input harmonics by number.

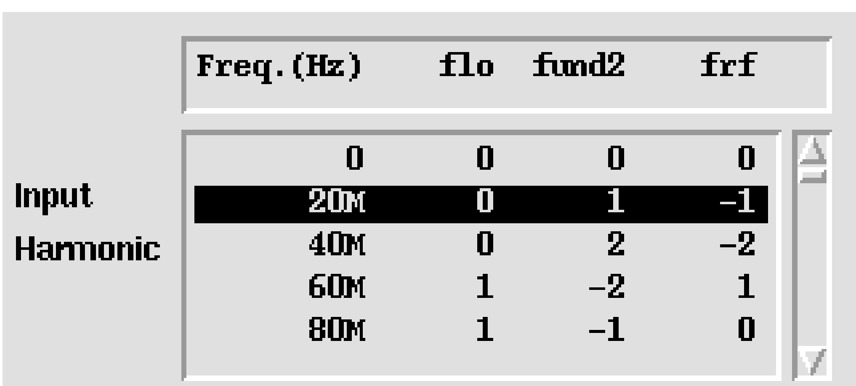



Lists available input harmonics by number. the list box appears on the PSS Plot form when you select one of the following functions: *Voltage Gain*, *Current Gain*, *Power Gain*, *Transconductance*, or *Transimpedance Functions*.

The values in the list box are those you requested in the *Output Harmonics* specification in the Choosing Analyses form.

## Input Harmonic (QPSS)

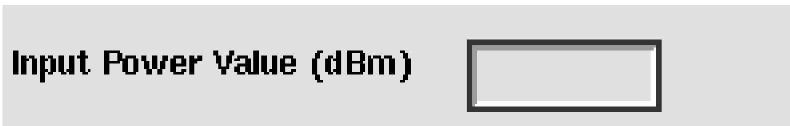
Lists available input harmonics.




In the list box, the first column lists the frequency value of a harmonic. The following columns list the tone coefficients for each fundamental tone that contributed to the harmonic.

## Input Power Value (dBm) (PSS, QPSS, PAC)

Specifies an input power value.

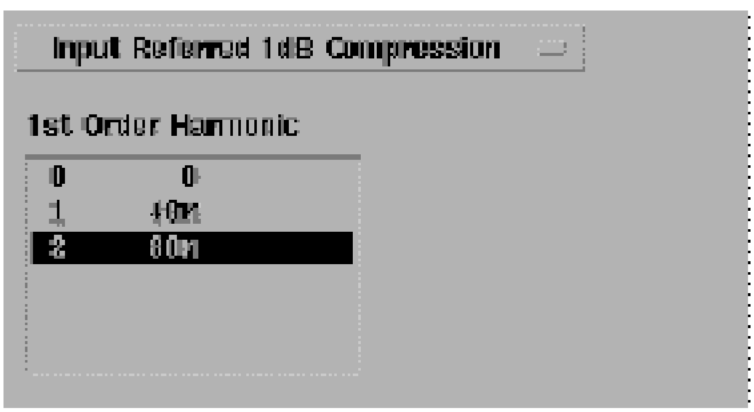



This field appears

- When you select *Single Point* for *Circuit Input Power*.
- When you select the IPN Curves Function.
- When you want information about a single point after running a power sweep.
- Assign an input power value.

## Input or Output Referred 1dB Compression (PSS, QPSS)

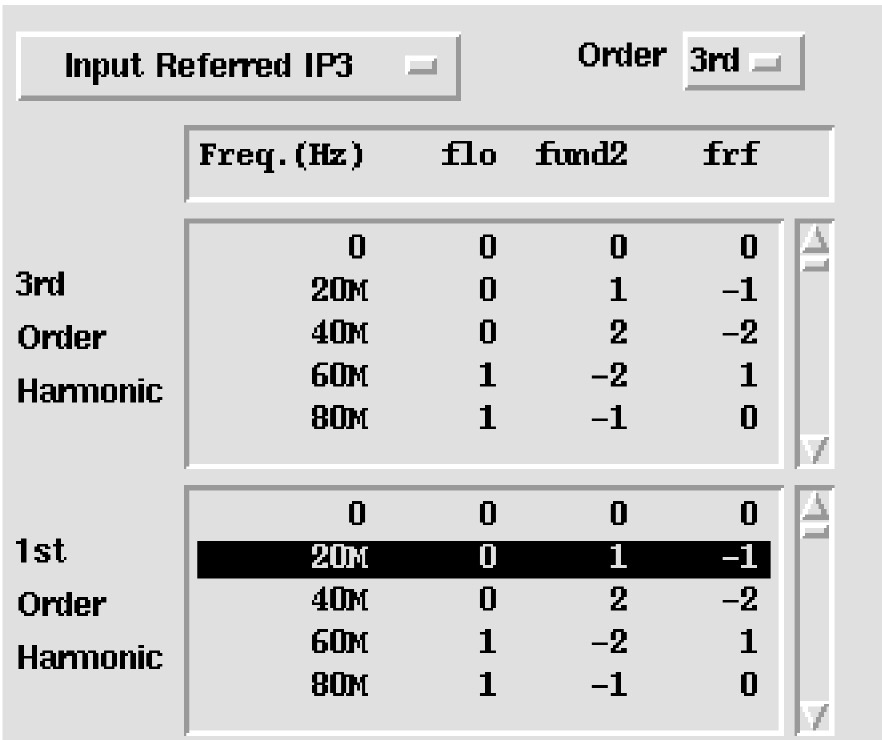
Selects input or output referred 1 dB compression.




- Output referred compression is referred to the y axis.
- Input referred compression is referred to the x axis.

## Input or Output Referred IPN and Order (PSS, QPSS, PAC)

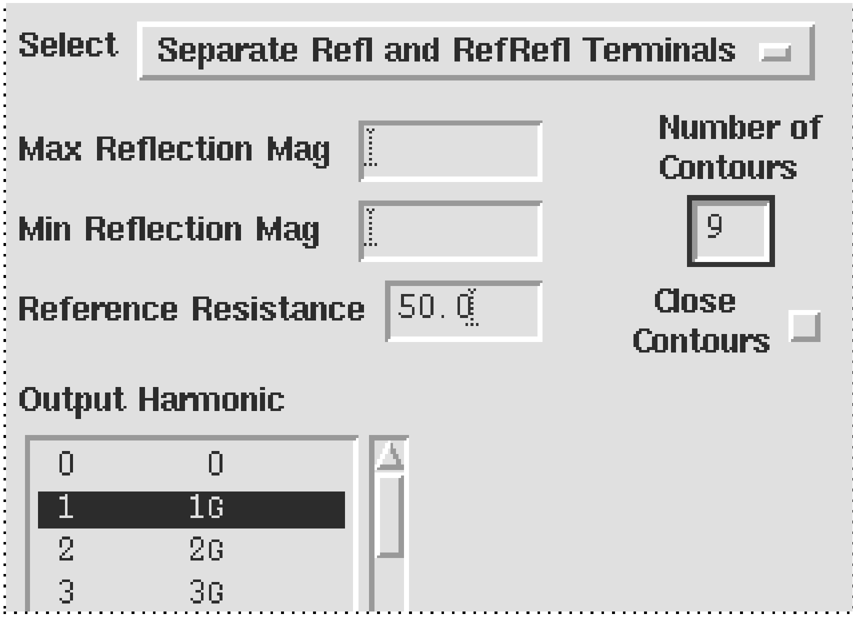
Selects Input or Output Referred Nth-Order Intercept point.




- Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.
- Select *Input Referred IPN* or *Output Referred IPN* in the Input/Output Referred IPN cyclic field.
- Output referred IPN is referred to the y axis.
- Input referred IP3 is referred to the x axis.

## Maximum Reflection Magnitude (ENVLP)

When you select the *Reflection Contours Function* in the envelope analysis:



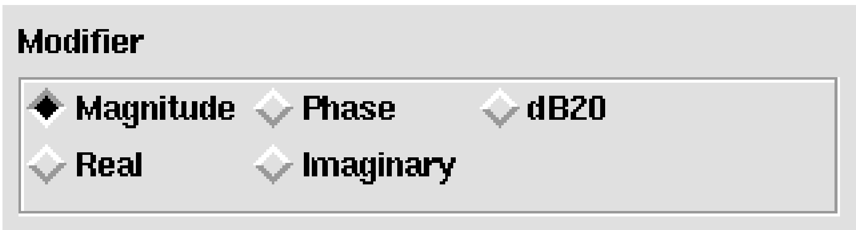

- Specifies a maximum reflection magnitude.
- Specifies a minimum reflection magnitude.
- Sets the resistance of the port adapter when you plot *Power* or *Reflection Contours*.
- Specifies the number of *Power* or *Reflection Contours* to plot.
- Sets open or closed contours for *Power* and *Reflection Contours*.
- When you select *Close Contours*, the plot appears as a closed figure. If *Close Contours* is not selected, the plot appears as an open figure. The default is to leave the two most distant points in the plot unconnected.
- Selects the Output harmonic.

## Min Reflection Mag

See “Maximum Reflection Magnitude (ENVLP)” on page 122.

## Modifier

Sets the units for the y axis of the plot.




Choices vary depending on the *Function* highlighted.

**Magnitude** is the raw value, in volts, amps, or no units at all.

**Phase** sets the y axis to degrees.

**dB20** sets the y axis to decibels with tick marks every 20 dB.

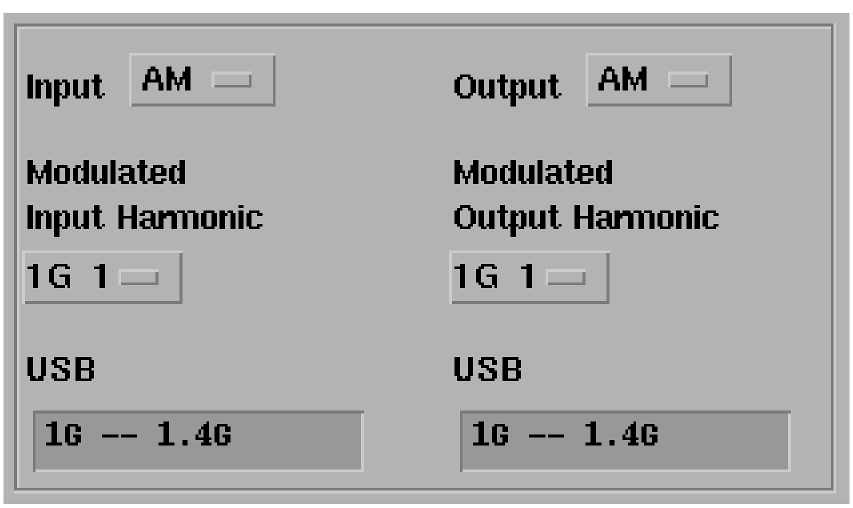
**dBm** sets the y axis to dB 10 plus 30.

**dB10** sets the y axis to decibels with tick marks every 10 dB.

**Real** and **Imaginary** restrict plots to only the real or imaginary range of the curve.

## Modulated Input/Output (PAC, PXF)

Displays information so you can plot PAC and PXF modulated input and output curves.



The values displayed in the *USB* fields are always with respect to the offset frequency. For example, if the *Output USB* results are for the 2nd or 3rd modulated harmonics, the frequency range is `2*fund_pss+f_offset` or `3*fund_pss+f_offset`. The Input USB values are also modulated. For example, for the 1st harmonic, the USB value is `1*fund_pss+f_offset`.

## Noise Type

Lists the types of noise calculated following a Pnoise analysis with *modulated Noise Type* selected.


Choices may vary.

*USB* is the upper sideband noise.

*LSB* is the lower sideband noise.

*AM* is the amplitude modulated noise.

*PM* is the phase modulated noise.

## Number of Contours

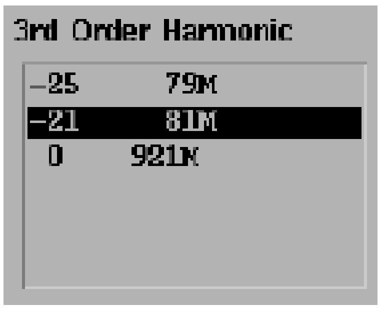
See “Maximum Reflection Magnitude (ENVLP)” on page 122.

Lists the harmonics available for plotting by number and associated frequency values. (Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.)

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

## Nth Order Harmonic (QPSS)

Lists available Nth Order Harmonics when you select the *IPN Curves Function* for the QPSS analysis.

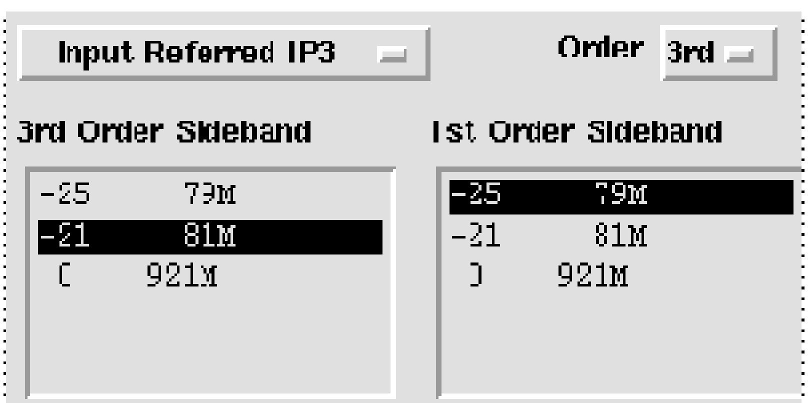



In the list box, the first column lists the frequency value of a harmonic. The following columns list the tone coefficient for each fundamental tone that contributed to the harmonic.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

## Nth Order Sideband (PAC)

Lists available Nth Order sidebands when you select the *IPN Curves Function* for a PAC analysis.




Lists the sidebands available for plotting by number and associated frequency values. (Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.)

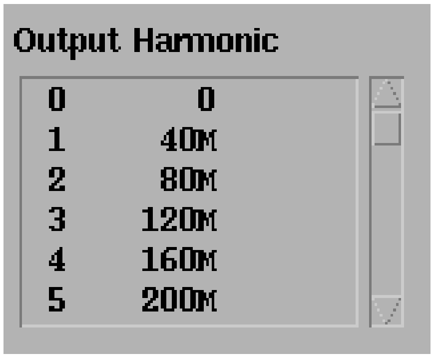
➤ Select one sideband from the list box and then select the appropriate net on the schematic.

## Order

See “Input or Output Referred IPN and Order (PSS, QPSS, PAC)” on page 121.

## Output Harmonic (PSS)

Lists available Output Harmonics for PSS analysis.




Lists the harmonics available for plotting by number and associated frequency.

Click harmonics in the list box to select them.

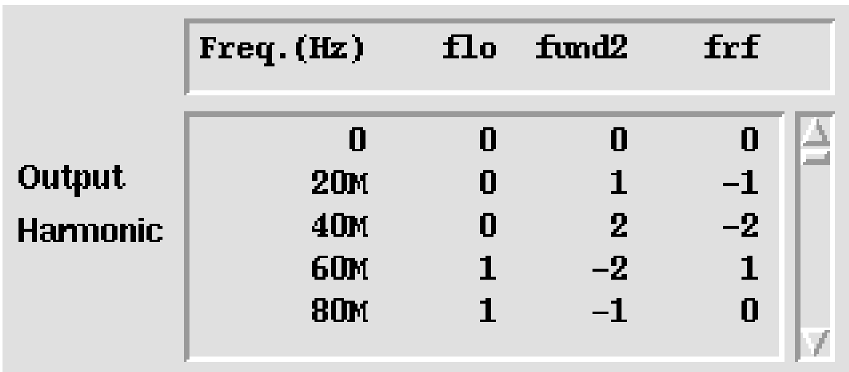
Select adjacent harmonics by clicking and dragging with the mouse over the harmonics you want to select.

Select harmonics that are not adjacent by holding the `Control` key down while you click the individual sidebands.

Deselect harmonics by holding the `Control` key down while you click a selected harmonic.

## Output Harmonic (For QPSS)

Lists available Output Harmonics for QPSS analysis.




In the list box, the first column is the frequency of a harmonic. The second and third columns specify the tone coefficients for each fundamental tone that contributed to the listed harmonic.

Click harmonics in the list box to select them.

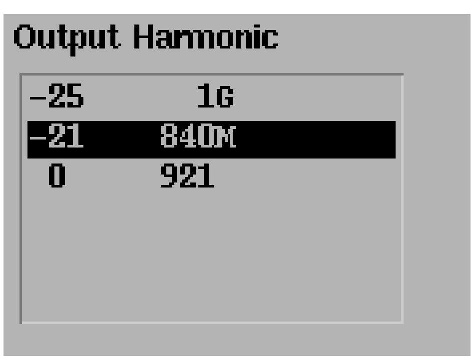
Select adjacent harmonics by clicking and dragging with the mouse over the harmonics you want to select.

Select harmonics that are not adjacent by holding the `Control` key down while you click the individual sidebands.

Deselect harmonics by holding the `Control` key down while you click a selected harmonic.

## Output Sideband (PAC, PXF)

Lists the sidebands you requested on the small-signal Choosing Analyses form.



This list box appears when you choose variable for *Sweep* on a small-signal Direct Plot form. It lists all the sidebands you requested on the small-signal Choosing Analyses form.

Click sidebands in the list box to select them.

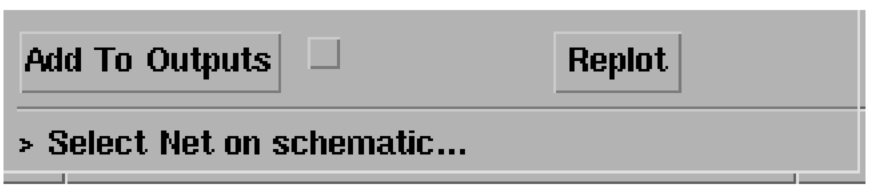
Select adjacent sidebands by clicking and dragging with the mouse over the sidebands you want to select.

Select sidebands that are not adjacent by holding the `Control` key down while you click the individual sidebands.

Deselect sidebands by holding the `Control` key down while you click a selected sideband.

## Plot and Replot

Displays a plot in the waveform window.



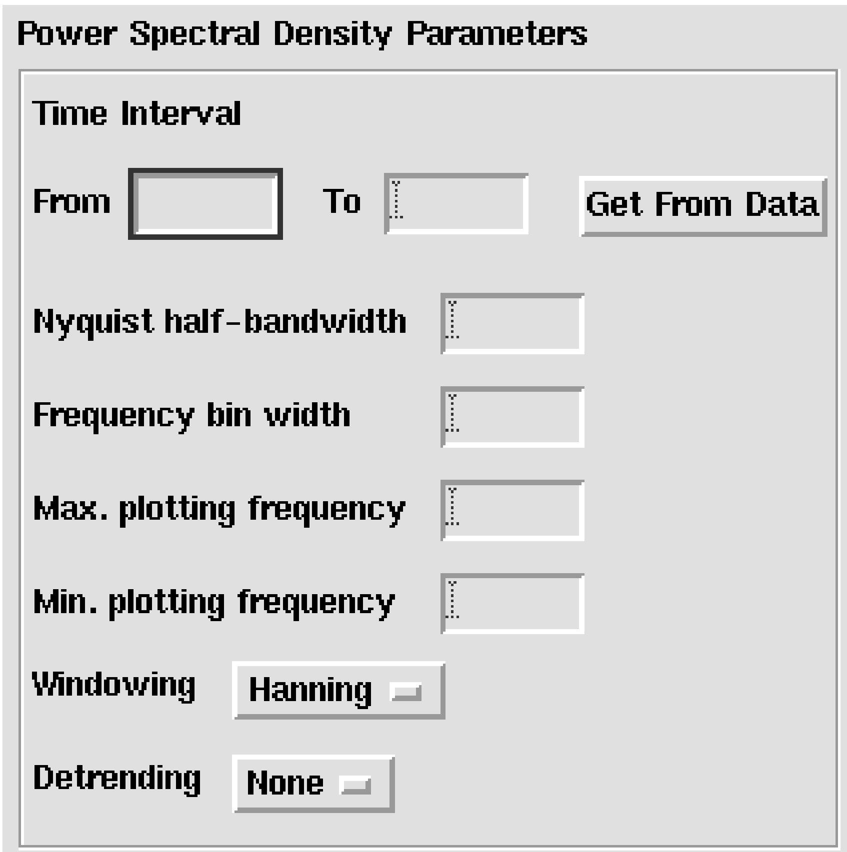

## Plot Mode

Determines whether to add the next plot to those currently displayed in the waveform window or to clear the window and display only the next plot.


- *Append* combines the next plot with other curves already plotted in the waveform window.
- *Replace* clears the waveform window just prior to displaying the next plot.

## Power Spectral Density Parameters (ENVLP)

Determines how the Power Spectral Density is calculated.




- **Time Interval** – The starting and ending times for the spectral analysis interval. They are usually the start and stop times for the simulator.
- **Get From Data** -- Sets the *From* and *To* values to match the values recorded in the results data.
- **Nyquist half-bandwidth** – The maximum frequency at which there are signals of interest. This is usually three to five times the maximum band frequency.
- **Frequency bin width** – The frequency resolution, such as the width of the frequency bins.
- **Max. plotting frequency** – Sets the maximum x axis value for the waveform you want to plot.
- **Min. plotting frequency** – Sets the minimum x axis value for the waveform you want to plot.

- **Windowing** – A preset list of available windowing functions used during the spectrum calculation.
- **Detrending** – Removes trends from the data before the spectral analysis.

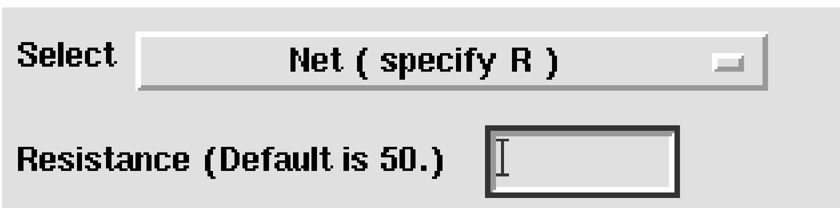
## Reference Resistance (ENVLP)

See “Maximum Reflection Magnitude (ENVLP)” on page 122.

## Resistance

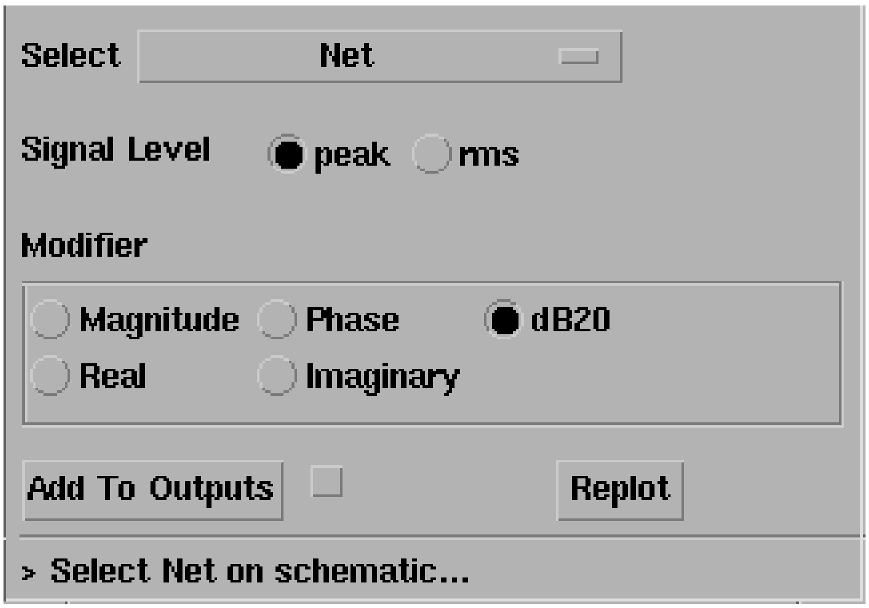
Sets the resistance of the port adapter when you plot *Power* or *Reflection Contours*.

Sets the resistance of the port adapter when you select Net (specify R) and plot Power or Reflection Contours.



## Select

Determines the type and number of objects to select on the schematic. There is a prompt at the bottom of the form describing how to make the selection.




Choices available in the *Select* cyclic field vary depending on the highlighted *Function*. The message at the bottom of the form prompts you to make an appropriate selection.

**Differential Nets** -- Select differential nets on the schematic.

**Differential Nets (dB, 1ohm reference)** -- Select differential nets on the schematic.

**Differential Terminal** -- Select a differential terminal on schematic.

**Instance with 2 Terminals** -- Select an instance with two terminals on the schematic.

**Net** -- Select a net on the schematic.

**Net (dB, 1ohm reference)** -- Select a net on the schematic.

**Out. and In. Ports (fixed R(OutPort))** -- Select output and input ports on the schematic.

- **Out. and In. Instances with 2 Terminals** -- Select output and input instances with two terminals on the schematic.
- **Output and Input Nets** -- Select output and input nets on the schematic.
- **+- Output and +- Input Nets** -- Select output and input nets on the schematic.

- **Output and Input Terminals** -- Select output and input terminals on the schematic.
- **+- Output and +- Input Terminals** -- Select output and input terminals on the schematic.
- **Output Net and Input Terminal** -- Select an output net and an input terminal on the schematic.
- **+- Output Net and Input Terminal** -- Select an output net and an input terminal on the schematic.
- **Port (fixed R(port))** -- Select a port on the schematic.
- **+- Power and +- Refl Terminals** -- Select power and reflection terminals on the schematic.
- **Separate Power and Refl Terminals** -- Select separate power and reflection terminals on the schematic.
- **Single Power/Refl Terminal** -- Select one power or reflection terminal on the schematic.
- **Single Power/Refl Term and ref Term** -- Select one power or reflection terminal and a reference terminal on the schematic.
- **Terminal** -- Select a terminal on the schematic.
- **Terminal and V-Reference Terminal** -- Select a terminal and a V-Reference terminal on the schematic.

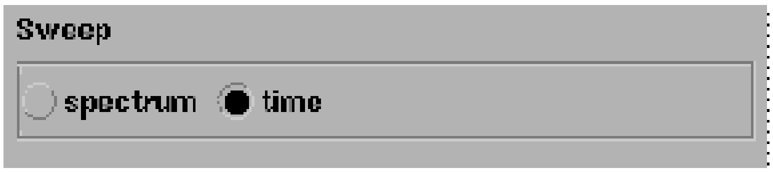
## Signal Level (PSS, QPSS)

Determines the signal value to plot.


- **Peak** -- Plots the maximum value of the signal.
- **rms** -- Plots the root-mean-square value, or effective value, of the signal.

## Sweep (PSS, PXF, ENVLP)

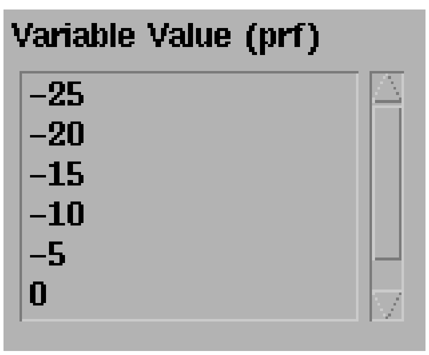
Sets the units for the x axis of the plot. Choices vary depending on the *Function* highlighted.




- *spectrum*, *sideband*, and *frequency* set the x axis to display frequency.
- *time* sets the x axis to display seconds.
- *variable* sets the x axis to display the value of a design variable.

## Variable Value (PSS, QPSS, PAC)

Lists the swept variable values that you can plot for a PSS, QPSS, or PAC analysis.




The PSS, QPSS, and PAC Results forms display this list of sweep values that you can specify for a specific variable, temperature, component parameter, or model parameter. the variable name is included in the title.

The range of values is determined by the *Sweep Range* specification in the Choosing Analyses form. The number of values is determined by the *Sweep Type* specification in the Choosing Analyses form.

In the sample figure, which shows the PSS version, the values are listed for the design variable `prf`. The QPSS version of the form is formatted slightly differently but gives the same information.

# ACPR Wizard

The ACPR Wizard simplifies the procedure for measuring ACPR and PSD.

Open the ACPR wizard in one of two ways.

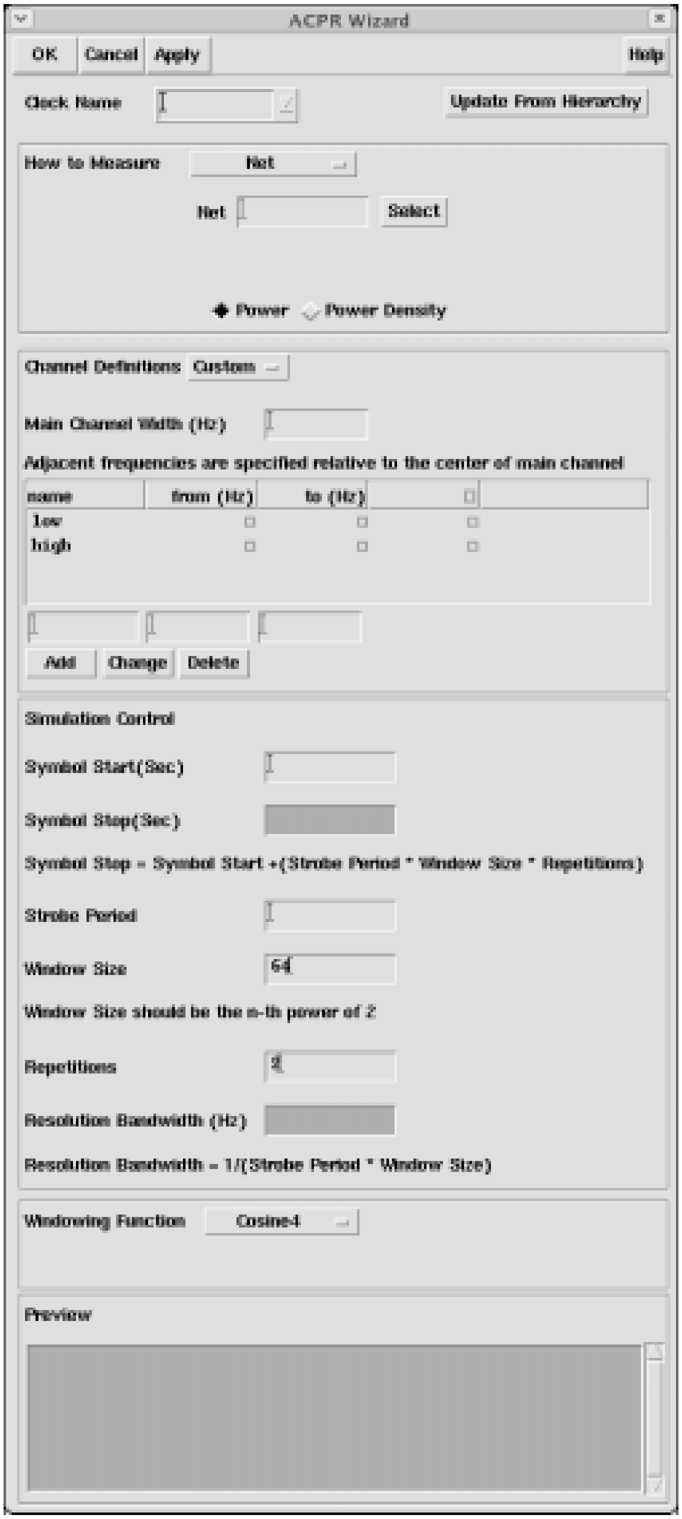
In the ADE window,

- Choose *Tools – RF – Wizards – ACPR*

  or

- In the *ENVLP* Choosing Analyses form, click *Start ACPR Wizard*.

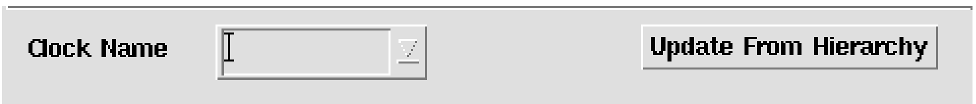
The ACPR wizard form opens.



The following sections describe the fields on the ACPR Wizard form.

## Clock Name

In the *Clock Name* cyclic field, select the clock signal from those listed in the cyclic field.



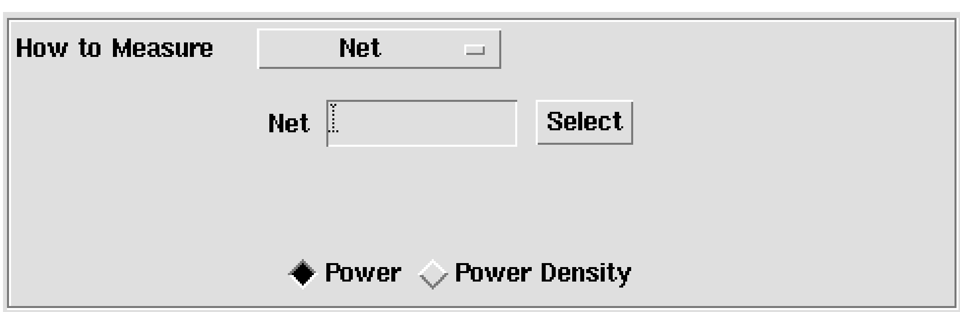

The *Clock Name* identifies the source of the modulated signal.

## How to Measure

The *How to Measure* section allows you to choose whether to measure ACPR for a single *Net* or between *Differential Nets*. Use the *How to Measure* cyclic field to choose a single *Net* or *Differential Nets*.

To measure ACPR for a single net,

1. Select *Net* in the *How to Measure* cyclic field.
2. Click *Select*. Then select the output net in the Schematic window.
3. *RFOUT* displays in the *Net* field.



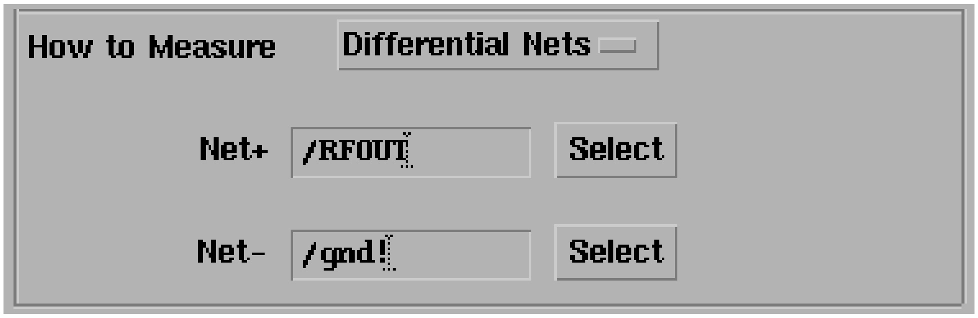

To measure ACPR for differential nets,

1. Select *Differential Nets* in the *How to Measure* cyclic field.
2. Click *Select* next to the *Net+* field. Then select the positive net in the Schematic window.
3. The signal name displays in the *Net+* field.
4. Click *Select* next to the *Net-* field. Then select the negative net in the Schematic window.

**5.** The signal name displays in the *Net-* field.

**6.** Select the method to calculate ACPR. Power selection uses the power-ratio method which compares the power in the specified adjacent-channel bandwidth to the total power of the carrier across the entire carrier bandwidth.

Power Density selection uses the power density method which compares the power density at the offset frequency to the power is within an average bandwidth of the same width in the carrier-channel bandwidth.



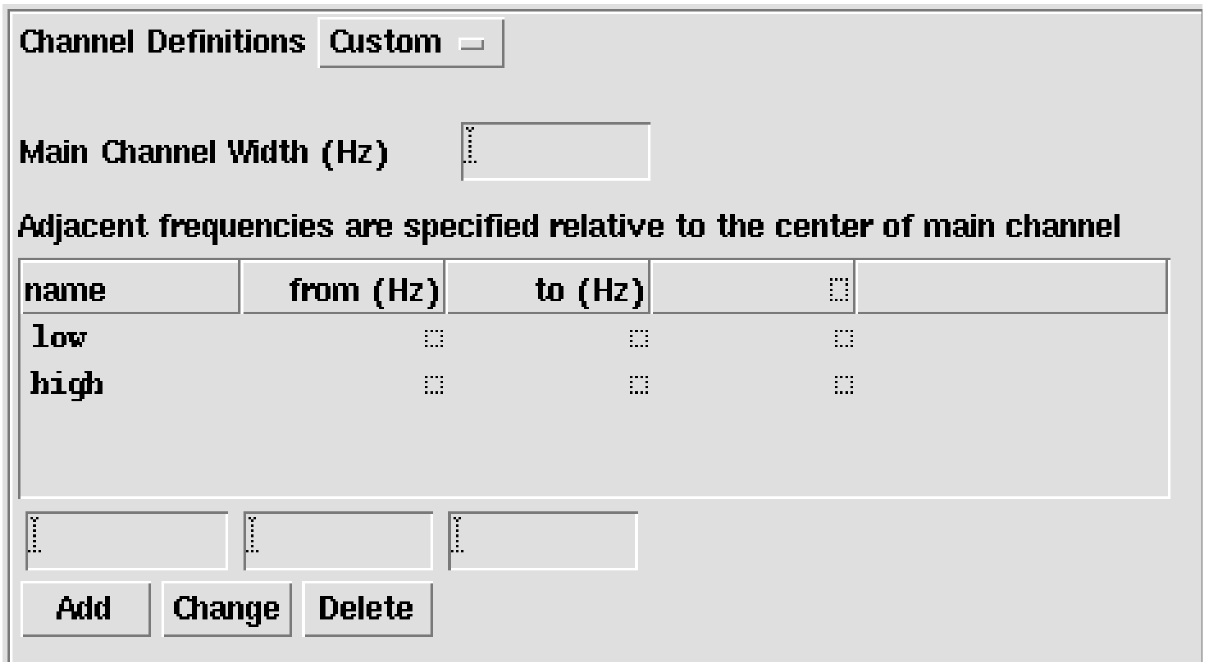

## Channel Definitions

In the *Channel Definitions* cyclic field, select a preset channel definition. Choices include *Custom*, *IS-95*, and *W-CDMA*.

When you select *IS-95*

- The *Main Channel Width (Hz)* field is calculated. This is the width of the main channel in Hz. Enter a number greater than zero. When you click *OK* or *Apply*, the content of this field is verified.
- The *adjacent frequencies* are determined and display in the list box. Note that adjacent frequencies are specified relative to the center of the main channel.
- Use the edit fields and the *Add*, *Change*, and *Delete* buttons to enter or modify channel definitions in the list box. You can hand edit or enter adjacent frequency names and

upper and lower boundaries. Specify adjacent frequencies relative to the center of the main channel. All channel widths must be greater than zero.




## Simulation Control

*Symbol Start (sec)* specifies the starting point of the time domain waveform to be used for DFT analysis.

*Symbol Stop (sec)* specifies the ending point of the time domain waveform to be used for DFT analysis. The wizard will calculate this value automatically based on the other fields using the following equation:

Symbol Stop = Symbol Start + (Strobe Period* Window Size* Repetitions)

*Strobe Period* specifies the output strobe interval. To achieve better accuracy, set strobe period to be half of baseband signal sample rate. It can be the value of period in second or in the format of 1/(strobe frequency).

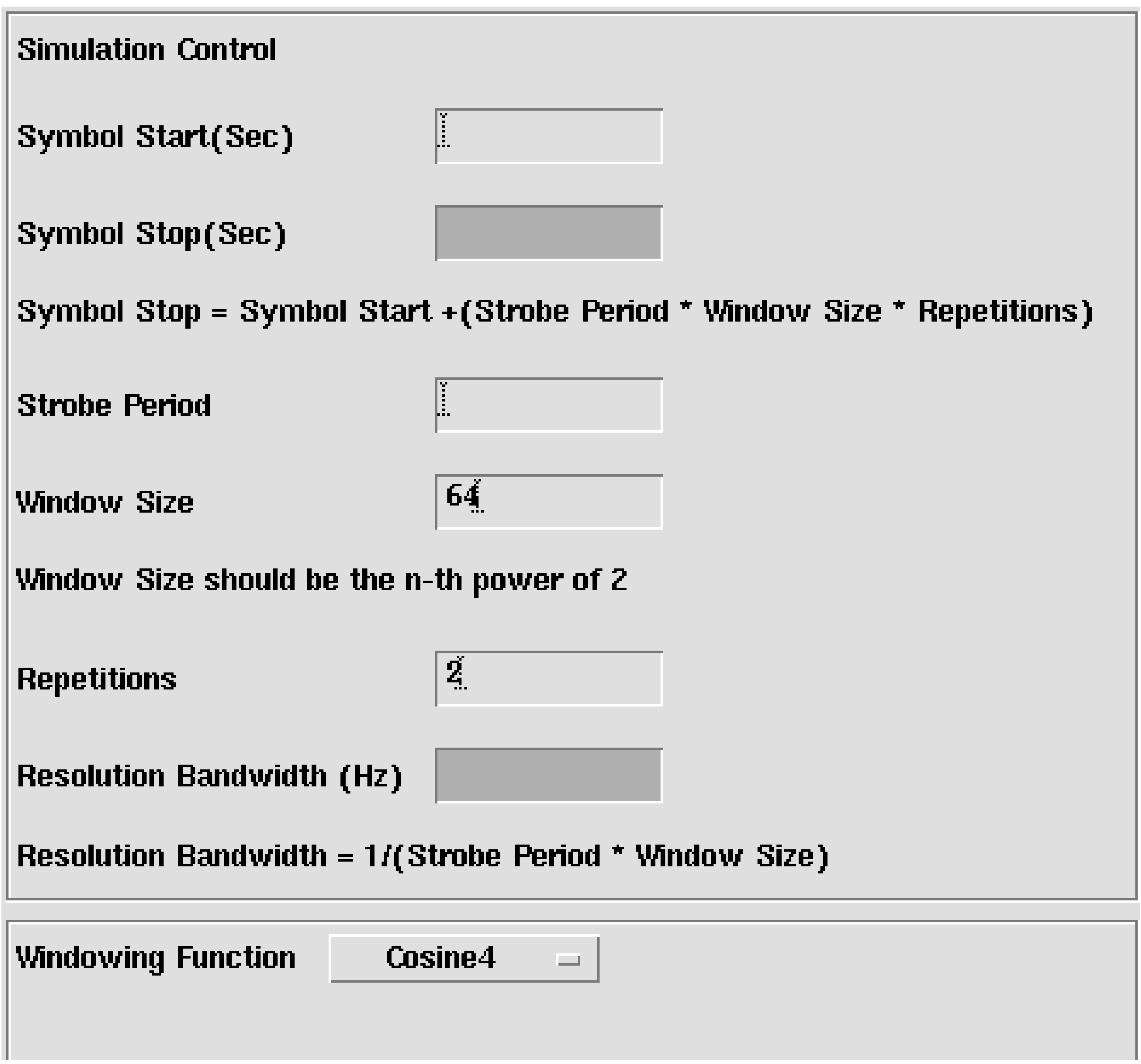
*Repetitions* specifies the number of times to repeat the DFT for averaging. When you increase the number of repetitions, the power density curve is smoother, simulation time is longer and the data file is larger.

*Resolution Bandwidth (Hz)* specifies the spacing of data points on the on the power density curve, in Hz.When you decrease the resolution bandwidth, simulation time is longer and the data file is larger. The wizard will calculate the resolution bandwidth based on the following relation:

Resolution Bandwidth = 1/ (Strobe Period * Window Size)

In the *Windowing Function* cyclic field, select a preset windowing function. Choices include: *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular*.

A *Windowing Function* tapers the signal before performing the DFT to reduce the effect of any edge discontinuities.

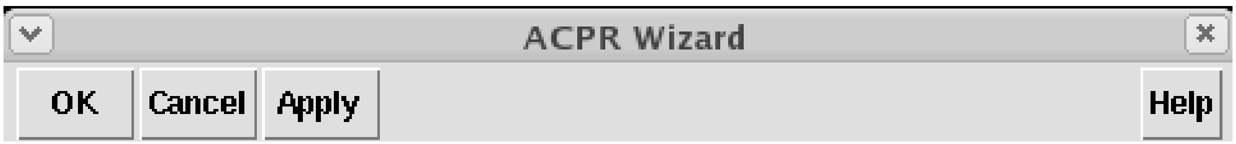


## Preview

Displays the equation used to calculate the ACPR. Based on this equation, you can adjust the Simulation Control parameters to achieve accuracy and simulation speed.

## OK and Apply

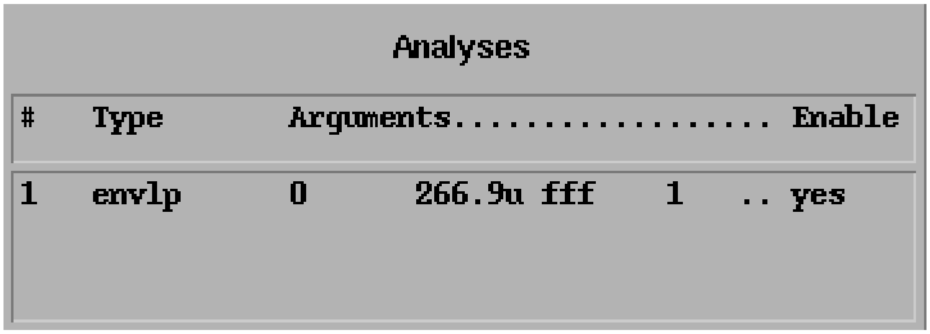
When you click *OK* or *Apply*, the values you entered in the ACPR wizard are used to determine values for the required ENVLP analysis and the ENVLP choosing analyses form is filled in.



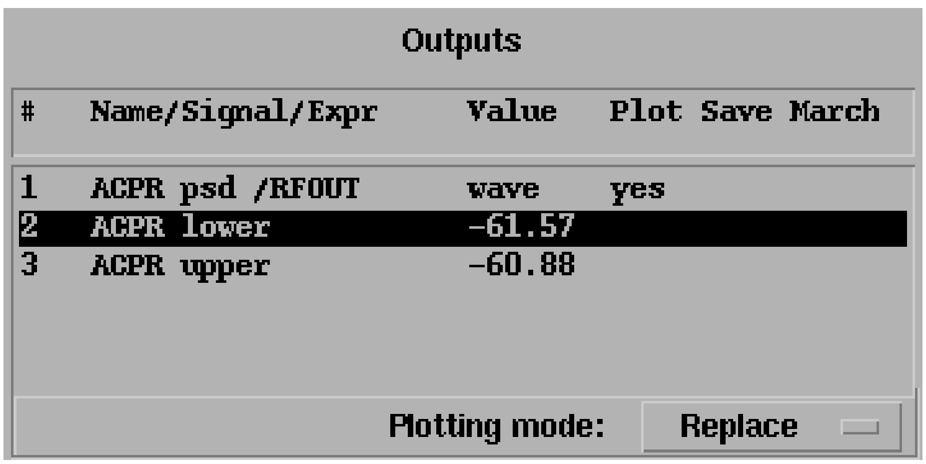

In the ENVLP choosing analyses form, values are calculated as follows.

- The *Clock Name* field in the ENVLP form is the same as the *Clock Name* field on the ACPR wizard.
- The *Stop Time* value in the ENVLP form is calculated.
- The *Output Harmonics* field in the ENVLP form is set to 1.
- The ENVLP choosing analyses form is enabled.
- The *Start* field in the ENVLP Options form is left blank.
- The *modulationbw* value for the ENVLP Options form is calculated.
- The *strobeperiod* value for the ENVLP Options form is calculated.
- You can modify these values entered on the ENVLP analysis form, but your changes are not propagated back to the ACPR wizard.

- The *Analyses* area in the ADE window reflects the ENVLP analysis.




When the ENVLP analysis completes, the ACPR values for each channel and the PSD waveform display in the *Outputs* area in the ADE window.



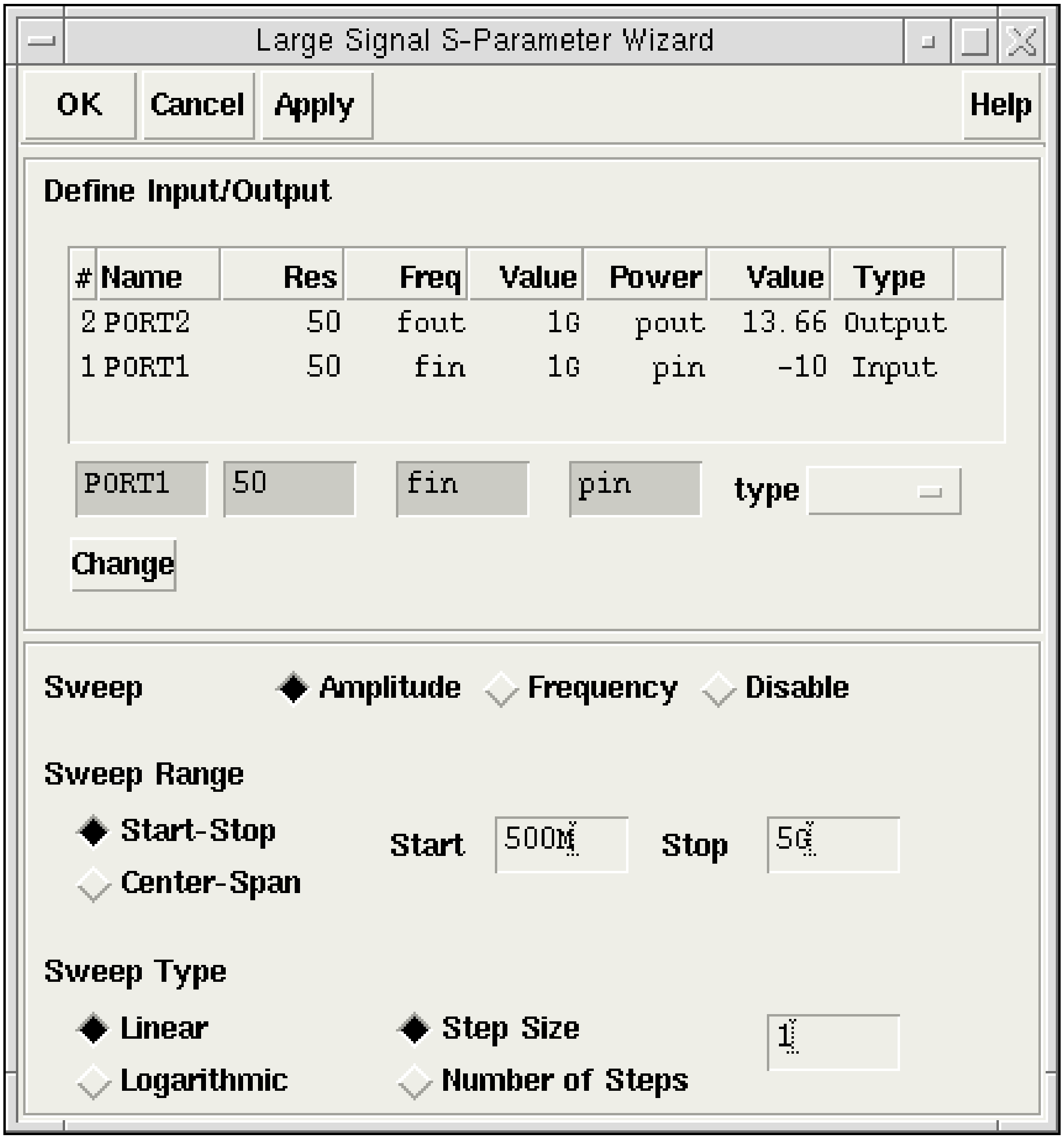

# Large Signal S-Parameter Wizard

The Large Signal S-Parameter (LSSP) Wizard simplifies the procedure for measuring large signal S-parameters with the Spectre RF simulator.

To open the LSSP Wizard form,

➤ In the ADE window, choose *Tools – RF – Wizards – LSSP*

The LSSP Wizard form opens.

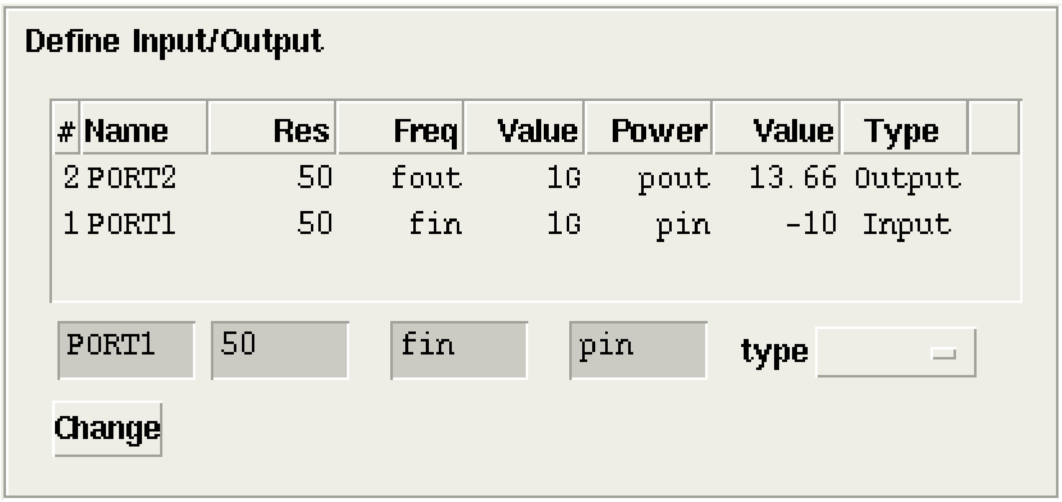



The following sections describe the fields on the LSSP Wizard form.

## Define Input/Output

In the *Define Input/Output* table, specify both the input and output ports. Ensure that the frequency and the amplitude of the ports are set to variables because it must be possible to sweep both the frequency and the power for both the input and output.

Two PSS sweeps are required for LSSP. The two sweeps require different source statuses. For the input power sweep, `port2` acts as a resistance when `port1` is a normal sine source. For the output power sweep, `port1` acts as a resistance when `port2` is a normal sine source.

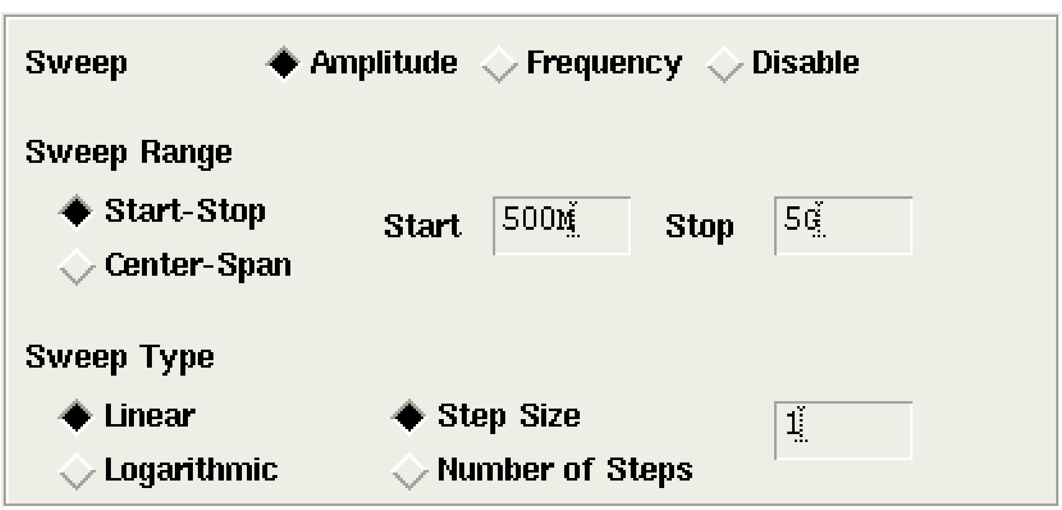



Use the *type* field to specify whether a port is *Input* or *Output*.

Use the *Change* button to modify port definitions.

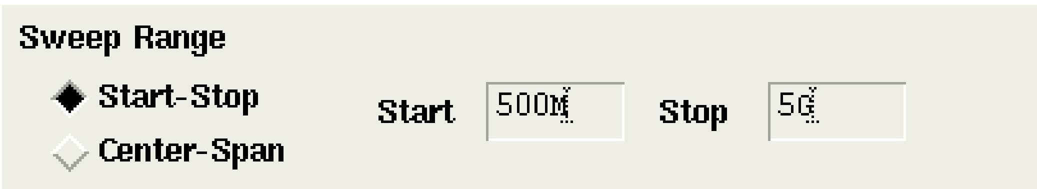
## Sweep

The *Sweep* setting affects only the input port. The sweep for the output port is set up by the simulator after the input sweep finishes.



## Sweep Range

Defines the sweep range for the analysis. Choices are: *Start-Stop*, and *Center-Span*. When you select one of these choices, the adjacent fields change to let you specify appropriate data.




### *Start - Stop*

Defines the beginning and ending points for the sweep.

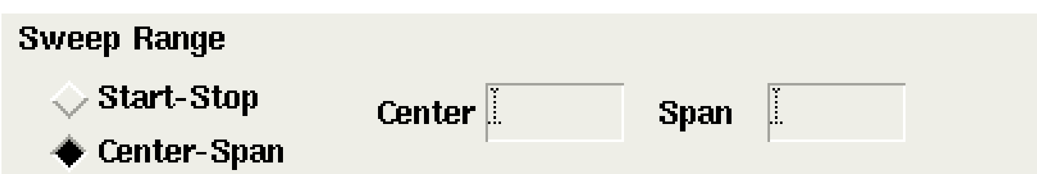

1. Highlight *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

### *Center - Span*

Defines the center point for the sweep and its span.




1. Highlight *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

## Sweep Type

Specifies whether the sweep is linear or logarithmic.

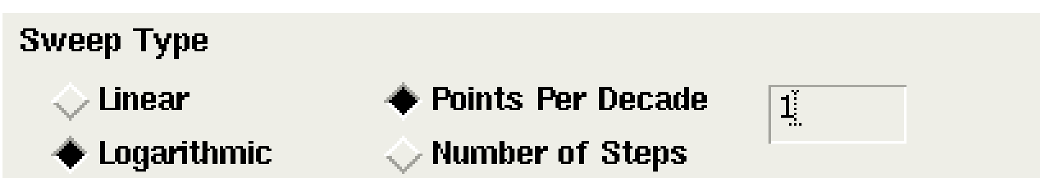
### *Linear*

Specifies a linear sweep.


1. Select *Linear*.
2. Do one of the following:
   - ❑ Highlight *Step Size* and type the size of each step in the field.
   - ❑ Highlight *Number of Steps* and type the number of steps (points) in the field.

### *Logarithmic*

Specifies a logarithmic sweep.




1. Select *Logarithmic*.
2. Do one of the following:
   - ❑ Highlight *Points per Decade* and type the number of points per decade in the field.
   - ❑ Highlight *Number of Steps* and type the number of steps (steps) in the field.

### OK and Apply

When you click *OK* or *Apply*, the values you entered in the LSSP wizard are used.

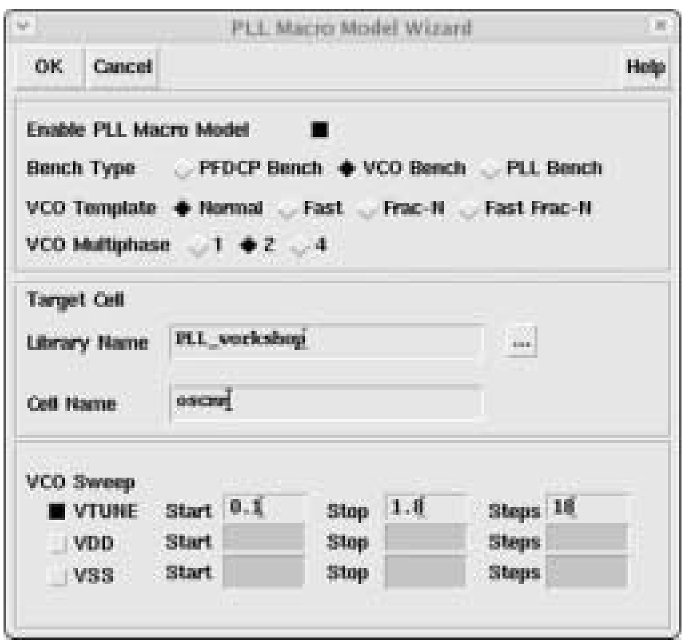

## PLL Macro Model Wizard

The PLL Macro Model Wizard simplifies the procedures involved in using a macro-model based simulation methodology.for PLL flows.

To open the PLL Macro Model Wizard form,

➤ In the ADE window, choose *Tools – RF – Wizards – PLL*.

The PLL Macro Model Wizard form opens.




For more information about using the PLL Macro Model Wizard, see *“Using the Noise-Aware PLL Flow in ADE,”* in appendix G of *Virtuoso Spectre Circuit Simulator RF Analysis User Guide (Volume 2)*.

# Instantiate Measurement Wizard

The Instantiate Measurement Wizard helps you choose and instantiate appropriate measurement or analysis cells.

To open and use the Instantiate Measurement Wizard form,

1. In the ADE window, choose *Tools – RF – Wizards – Instantiate Measurement*.

   The Instantiate Measurement Wizard opens with the *Circuit Type* pane displayed.

2. Select the type of the circuit in which you are instantiating the measurement component.

   The *Measurement* pane replaces the *Circuit Type* pane.

3. In the *Measurement* pane, select the type of measurement you want to instantiate.

   The *Method* pane replaces the *Measurement* pane.

4. In the *Method* pane, select the method that the instantiated measurement component must support.

   The Instantiate Measurement Wizard closes and the Add Instance form appears displaying values for one of the `measureLib` instances.

5. In the Add Instance form, set the parameters so that they are appropriate for your purpose.

6. Move the cursor over the Schematic Editing window and place the instance into the schematic.

7. When you finish adding instances, check and save the schematic.

   That completes the process of adding the component to the schematic. To enable the measurement, use the Choosing Analyses form with *measure* selected. For more information, see "Measurement Analysis (measure)" on page 34.

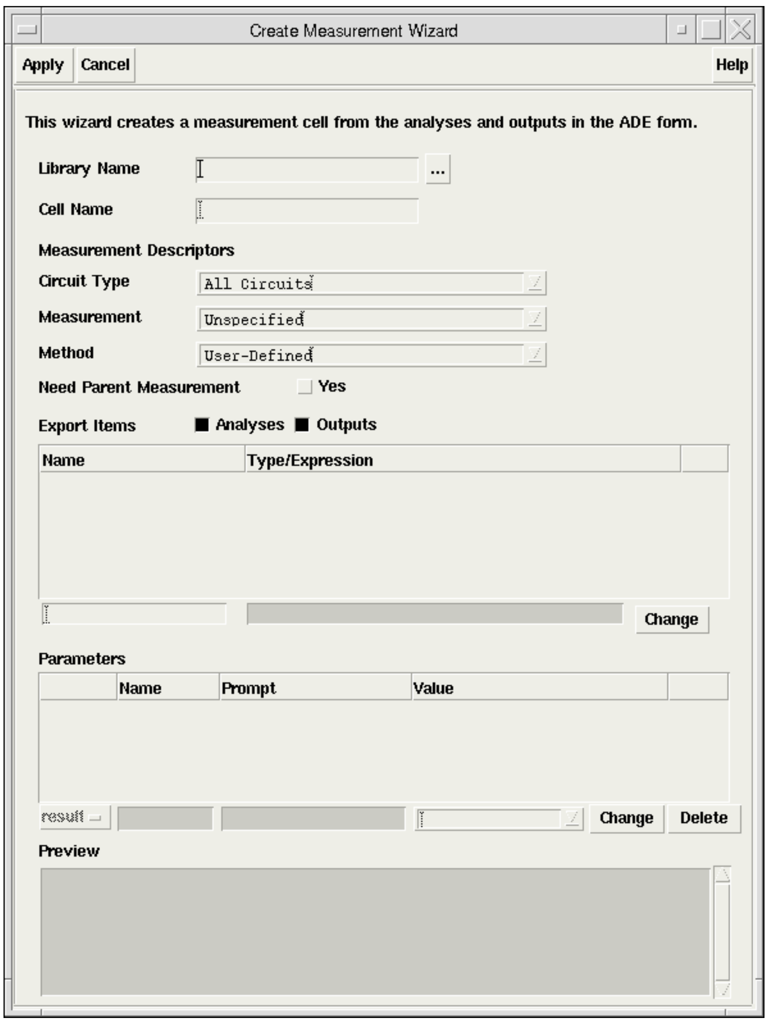
# Create Measurement Wizard

The Create Measurement Wizard helps you create new measurements or analyses cells based on the analysis and output setups in the ADE form.

To open and use the Create Measurement Wizard,

1. Set up the analyses and outputs that you want to include in the new measurement and verify that they work correctly.
2. In the ADE window, choose *Tools – RF – Wizards – Create Measurement*.

   The Create Measurement Wizard opens.




3. In the *Library Name* field, specify the library to hold the to-be-generated component.
4. In the *Cell Name* field, specify the cell name for the new component.

**5.** In the *Circuit Type* field, specify the circuit type or testbench to which the measurement applies.

| | |
|---|---|
| *All Circuits* | The measurement can be used on all kinds of circuits. |
| *Voltage Controlled Oscillator* | The measurement is intended for use on voltage-controlled oscillators. |

**6.** In the *Measurement* field, specify the kind of measurement that is implemented by the new component. If none of the predefined descriptors is appropriate, you can type in the field to specify something different. These descriptors are used by the Instantiate Measurement Wizard as an aid in selecting the correct measurement.

| |
|---|
| *Unspecified* |
| *Frequency and Power of VCO* |
| *Frequency Pulling of VCO* |
| *Jitter of VCO* |
| *Noise of OSC* |
| *Phase noise of VCO* |

**7.** In the *Method* field, indicate the analyses that are set up and used by the measurement.

| | |
|---|---|
| *User-Defined* | Sets up unspecified analyses, presumably analyses different than the kinds listed in the other rows of this table. |
| *PNOISE* | |
| *PSS + Pnoise* | Sets up a PSS analysis followed by a Pnoise analysis, such as might be used to compute a noise figure. |
| *PSS + modulated Pnoise* | |
| *Sweep PSS* | Sets up an IP3 analysis that uses swept PSS analysis. |

**8.** In the *Need Parent Measurement* field, select Yes if ...???

9. (Optional) In the *Export Items* pane, indicate whether the defined analyses, the defined outputs, or both are included in the CDF of the measurement cell being created.???

10. (Optional) In the *Parameters* pane, change the prompts for the parameters that are used in the analyses and in the output expressions. The *Parameters* pane lists all the parameters extracted from analyses and outputs. The values in the Parameters table are used as CDF parameters of the measurement cell instance.

| | |
|---|---|
| First column | Analyses name or output index of the parameter. When the first column contains `ana`, the parameter applies to all analyses. A number in the first column is the index of a parameter derived from outputs. |
| *Name* | Name of the analysis. |
| *Prompt* | Prompt for the parameter. |
| *Value* | Default value and possible choices. |

You can delete parameters if you know there is no need to change their values when the created component is instantiated.

11. When you are done making changes to the form, click *Apply*.

The measurement or analysis cell is created and added to the specified library. For information on instantiating the cell, see “Instantiate Measurement Wizard” on page 147. For information on running a simulation that uses the measurement, see “Measurement Analysis (measure)” on page 34.

# The Spectre RF Simulation Forms Quick Reference

The Spectre RF simulation forms include the following:


- "ACPR Wizard" on page 134
- "Choosing Analyses Form" on page 151
- "Create Measurement Wizard" on page 148
- "Direct Plot Form" on page 168
- "Instantiate Measurement Wizard" on page 147
- "Large Signal S-Parameter Wizard" on page 141
- "Options Forms" on page 165 (One for each analysis)
- "PLL Macro Model Wizard" on page 146


The simulation forms change to display only the fields relevant for the currently selected analysis. The field description topics for each analysis form are briefly described here and linked to the detailed description in this chapter.

## Choosing Analyses Form

The Choosing Analyses form changes depending on which analysis is selected. For guidance on what the form contains for a particular analysis, choose the appropriate link here.


- "Envelope (ENVLP) Choosing Analyses Form" on page 154
- "Harmonic Balance (HB) Choosing Analyses Form" on page 162
- "Harmonic Balance AC (HBAC) Choosing Analyses Form" on page 164
- "Harmonic Balance Noise (HBnoise) Choosing Analyses Form" on page 164
- "Measurement Analysis (measure) Choosing Analyses Form" on page 162
- "Periodic AC (PAC) Choosing Analyses Form" on page 155
- "Periodic Noise (Pnoise) Choosing Analyses Form" on page 156
- "Periodic S-Parameter (PSP) Choosing Analyses Form" on page 158
- "Periodic Stability (PSTB) Choosing Analyses Form" on page 156
- "Periodic Steady-State (PSS) Choosing Analyses Form" on page 152

- "Periodic Transfer Function (PXF) Choosing Analyses Form" on page 158
- "Quasi-Periodic AC (QPAC) Choosing Analyses Form" on page 160
- "Quasi-Periodic Noise (QPnoise) Choosing Analyses Form" on page 159
- "Quasi-Periodic S-Parameter (QPSP) Choosing Analyses Form" on page 161
- "Quasi-Periodic Steady State (QPSS) Choosing Analyses Form" on page 153
- "Quasi-Periodic Transfer Function (QPXF) Choosing Analyses Form" on page 161

### Periodic Steady-State (PSS) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *Engine* specifies the method used to analyze the design. | "Engine (ENVLP, PSS, QPSS)" on page 11 |
| *Fundamental Tones* displays and edits information for top level tones in the circuit. | "Fundamental Tones (PSS, QPSS)" on page 16 |
| *Beat Frequency*, *Beat Period*, *Auto Calculate* determine whether the PSS analysis uses *Beat Frequency* or *Beat Period*. | "Beat Frequency, Beat Period, and Auto Calculate (PSS)" on page 8 |
| *Output Harmonics* selects and defines output harmonics. | "Output Harmonics (PSS, ENVLP)" on page 47 |
| *Accuracy Defaults* quickly adjusts simulation parameters. | "Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)" on page 6 |
| *Additional Time for Stabilization* allows time for stabilization. | "Additional Time for Stabilization (tstab) (PSS, QPSS)" on page 7 |
| *Save Initial Transient Results* saves the initial transient solution. | "Save Initial Transient Results (PSS, QPSS, HB)" on page 52 |
| *Oscillator* defines the simulation for an oscillator circuit. (Displays additional fields to specify oscillator analysis.) | "Oscillator (PSS, ENVLP, HB)" on page 40 |
| *Sweep* selects swept analysis. (Displays additional fields to specify sweep.) | "Sweep (PSS, QPSS, HB)" on page 74 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Enabled* includes this analysis in the next simulation. | “Enabled” on page 11 |
| *Options* displays the Options form for this analysis. | “Options” on page 40 |

#### *Modifications to PSS Form for Oscillator Analysis*

Oscillator Node and Reference Node specify how the PSS oscillator analysis is performed.

#### *Modifications to PSS Form for Swept PSS Analysis*

Sweep, Sweep Range, Sweep Type, and Add Specific Points specify how the PSS sweep is performed.

#### Quasi-Periodic Steady State (QPSS) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | “Analysis” on page 7 |
| *Engine* specifies the method used to analyze the design. | “Engine (ENVLP, PSS, QPSS)” on page 11 |
| *Fundamental Tones* displays and edits information for top level tones in the circuit. | “Fundamental Tones (PSS, QPSS)” on page 16 |
| *Harmonics* displays fields used to specify harmonics. | “Harmonics (QPSS)” on page 21 |
| *Accuracy Defaults* quickly adjusts simulation parameters. | “Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)” on page 6 |
| *Additional Time for Stabilization* allows time for stabilization. | “Additional Time for Stabilization (tstab) (PSS, QPSS)” on page 7 |
| *Save Initial Transient Results* saves the initial transient solution. | “Save Initial Transient Results (PSS, QPSS, HB)” on page 52 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Sweep* selects swept analysis. (Displays additional fields to specify sweep.) | “Sweep (PSS, QPSS, HB)” on page 74 |
| *Enabled* includes this analysis in the next simulation. | “Enabled” on page 11 |
| *Options* displays the Options form for this analysis. | “Options” on page 40 |

#### *Modifications to QPSS Form for Swept QPSS Analysis*

Sweep, Sweep Range, Sweep Type, and Add Specific Points specifies how the QPSS sweep is performed.

### Envelope (ENVLP) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | “Analysis” on page 7 |
| *Engine* specifies the method used to analyze the design. | “Engine (ENVLP, PSS, QPSS)” on page 11 |
| *Fund Frequency* specifies the frequency of the clock fundamental. | “Fundamental Tones (PSS, QPSS)” on page 16 |
| *Period* specifies the period of the clock fundamental. For autonomous circuits, Period specifies the estimated period. | “Period (ENVLP)” on page 50 |
| *Clock Name* and *Select Clock Name* select the clock signal for the analysis. | “Clock Name and Select Clock Name Button (ENVLP)” on page 9 |
| *Stop Time* specifies the end time for the analysis. | “Stop Time (ENVLP)” on page 74 |
| *Output Harmonics* selects and defines output harmonics. | “Output Harmonics (PSS, ENVLP)” on page 47 |
| *Oscillator* defines the simulation for an oscillator circuit. (Displays additional fields to specify oscillator analysis.) | “Oscillator (PSS, ENVLP, HB)” on page 40 |

| Field or Pane | User Interface Help (page) |
| --- | --- |
| *Accuracy Defaults* quickly adjusts simulation parameters. | "Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)" on page 6 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

**Periodic AC (PAC) Choosing Analyses Form**

| Field or Pane | User Interface Help (page) |
| --- | --- |
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *PSS Beat Frequency (Hz)* displays the Beat Frequency for the associated PSS analysis. | "PSS Beat Frequency (PAC, PSTB, Pnoise, PXF)" on page 52 |
| *Sweeptype*, *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 57 |
| *Specialized Analyses (PAC)* specifies Modulated analysis. | "Specialized Analyses (PAC, HBAC)" on page 65 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

### *Modifications to PAC Form for Swept PSS Analysis*

Sweeptype**,** Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept PSS analysis.

### Periodic Stability (PSTB) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *PSS Beat Frequency (Hz)* displays the *Beat Frequency* for the associated PSS analysis. | "PSS Beat Frequency (PAC, PSTB, Pnoise, PXF)" on page 52 |
| *Periodic Stab Analysis Notification (PSTB)* sets up the beginning and end points of the sweep. | "Periodic Stab Analysis Notification (PSTB)" on page 50 |
| *Sweep Type (PSTB)* determines whether the sweep is linear, logarithmic, or chosen automatically. | "Sweep Type (PSTB, HBAC, HBNOISE)" on page 80 |
| *Add Specific Points* helps set up the sweep for the analysis. | "Probe Instance (PSTB)" on page 51 |
| *Probe Instance (PSTB)* specifies a probe that is placed in the feedback loop to identify and characterize the particular loop of interest. Introducing the probe component must not change the circuit characteristics. he probe must be a gain instance, such as a bjt transistor or a mos transistor. | "Probe Instance (PSTB)" on page 51 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

### Periodic Noise (Pnoise) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *PSS Beat Frequency (Hz)* displays the *Beat Frequency* for the associated PSS analysis. | "PSS Beat Frequency (PAC, PSTB, Pnoise, PXF)" on page 52 |
| *Sweeptype*, *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 57 |
| *Output*, *Input Source*, and *Reference Side-Band* selects the output, noise generator, and reference sidebands for the Pnoise analysis. (Displays additional fields. | "Output (Pnoise, QPnoise, HBnoise)" on page 45, "Input Source and Reference Side-Band (Pnoise, HBnoise)" on page 27 |
| *Noise Type* selects the type of noise to compute. (Active only when PSS analysis is not swept.) | "Noise Type (Pnoise)" on page 38 |
| *Noise Separation* tells the simulator to calculate how noise sources contribute noise to the output. | "Do Noise (HBNOISE)" on page 10 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

### *Modifications to Pnoise Form for Swept PSS analysis*

Sweeptype**,** Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept PSS analysis.

(Noise Type is not active when PSS analysis is swept.)

### Periodic Transfer Function (PXF) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *PSS Beat Frequency (Hz)* displays the *Beat Frequency* for the associated PSS analysis. | "PSS Beat Frequency (PAC, PSTB, Pnoise, PXF)" on page 52 |
| *Sweeptype*, *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 57 |
| *Output* selects the output. | "Output (PXF, QPXF)" on page 43 |
| *Modulated Analysis (PAC, PXF)* — One of the Specialized Analyses, specifies Modulated analysis. | "Modulated Analysis (PAC, PXF)" on page 34 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to PXF Form for Swept PSS Analysis*

Sweeptype**,** Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept PSS analysis.

### Periodic S-Parameter (PSP) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |

| **Field or Pane** | **User Interface Help (page)** |
|---|---|
| *Sweeptype*, *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Select Ports* selects the active ports for the PSP analysis. | "Select Ports (PSP, QPSP)" on page 52 |
| *Do Noise* selects whether or not to measure noise during the PSP analysis. | "Do Noise (PSP, QPSP)" on page 10 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to PSP Form for Swept PSS Analysis*

Sweeptype**,** Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept PSS analysis.

### Quasi-Periodic Noise (QPnoise) Choosing Analyses Form

| **Field or Pane** | **User Interface Help (page)** |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (QPAC, QPnoise, QPXF)" on page 60 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Output*, *Input Source*, *Reference Side-Band*, and *refsidebandoption* selects the output, noise generator, reference sidebands, and refsidebandoption for the QPnoise analysis. (Displays additional fields.) | "Output (Pnoise, QPnoise, HBnoise)" on page 45, "Input Source and Reference Side-Band (QPnoise)" on page 30 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to QPnoise Form for Swept QPSS Analysis*

Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept QPSS analysis.

### Quasi-Periodic AC (QPAC) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (QPAC, QPnoise, QPXF)" on page 60 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to QPAC Form for Swept QPSS Analysis*

Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept QPSS analysis.

### Quasi-Periodic Transfer Function (QPXF) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
| --- | --- |
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (QPAC, QPnoise, QPXF)" on page 60 |
| *Output* selects the output. | "Output (PXF, QPXF)" on page 43 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to QPXF Form for Swept QPSS Analysis*

Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept QPSS analysis.

### Quasi-Periodic S-Parameter (QPSP) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
| --- | --- |
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Sweeptype*, *Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 84, "Frequency Sweep Range, Sweep Type, Add Specific Points (Small-Signal)" on page 11 |
| *Select Ports* selects the active ports for the QPSP analysis. | "Select Ports (PSP, QPSP)" on page 52 |
| *Do Noise* selects whether or not to measure noise during the QPSP analysis. | "Do Noise (PSP, QPSP)" on page 10 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to QPSP Form for Swept QPSS Analysis*

Sweeptype**,** Frequency Sweep Range, Single-Point, and Freq specify the frequency for the small-signal analysis that follows each swept QPSS analysis.

### Measurement Analysis (measure) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Measurement Analysis* lists the measurement and analysis cells that are instantiated in the schematic. | "Measurement Analysis (measure)" on page 34 |

### Harmonic Balance (HB) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Analysis" on page 7 |
| *Tones* displays and edits information for top level tones in the circuit. | "Tones (HB)" on page 87 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Signal Partition* defines partitions, fundamentals, and harmonics that can be used to improve the performance of the harmonic balance engine in multi-tone circuits such as RF mixers. | "Multi-rate Harmonic Balance (HB)" on page 63 |
| *Accuracy Defaults* quickly adjusts simulation accuracy parameters. | "Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)" on page 6 |
| *Additional Time for Transient-Aided HB* allows time for stabilization. | "Additional Time for Transient-Aided HB (tstab) (PSS, QPSS, HB)" on page 7 |
| *Save Initial Transient Results* saves the initial transient solution. | "Save Initial Transient Results (PSS, QPSS, HB)" on page 52 |
| *Harmonic Balance Homotopy Method* determines the method used to pursue convergence in a circuit. | "Harmonic Balance Homotopy Method (HB)" on page 21 |
| *Oscillator* defines the simulation for an oscillator circuit. (Displays additional fields to specify oscillator analysis.) | "Oscillator (PSS, ENVLP, HB)" on page 40 |
| *Sweep* selects swept analysis. (Displays additional fields to specify sweep.) | "Sweep (PSS, QPSS, HB)" on page 74 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

### *Modifications to HB Form for Oscillator Analysis*

Oscillator Node and Reference Node specify how the HB oscillator analysis is performed. Osc initial condition determines how the starting values for the oscillator are determined.

### *Modifications to HB Form for Swept Analysis*

Sweep, Sweep Range, Sweep Type, Add Specific Points, and New Initial Value For Each Point (HB) specify how the HB sweep is performed.

### Harmonic Balance AC (HBAC) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | “Analysis” on page 7 |
| *Sweeptype*, *Input Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | “Sweeptype (PAC, PXF, HBAC, HBNOISE)” on page 84, “Input Frequency Sweep Range (HBAC)” on page 25, “Sweep Type (PSTB, HBAC, HBNOISE)” on page 80, “Add Specific Points (HBAC, HBNOISE)” on page 6 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | “Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)” on page 57 |
| *Specialized Analyses* measure AM and PM small-signal effects. | “Specialized Analyses (PAC, HBAC)” on page 65 |
| *Enabled* includes this analysis in the next simulation. | “Enabled” on page 11 |
| *Options* displays the Options form for this analysis. | “Options” on page 40 |

### Harmonic Balance Noise (HBnoise) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | “Analysis” on page 7 |
| *Sweeptype*, *Output Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | “Sweeptype (PAC, PXF, HBAC, HBNOISE)” on page 84, “Output Frequency Sweep Range (HBnoise)” on page 46, “Sweep Type (PSTB, HBAC, HBNOISE)” on page 80, “Add Specific Points (HBAC, HBNOISE)” on page 6 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | “Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)” on page 57 |

| Field or Pane | User Interface Help (page) |
|---|---|
| *Output*, *Input Source*, and *Reference Side-Band* selects the output, noise generator, and reference sidebands for the Pnoise analysis. (Displays additional fields. | “Output (Pnoise, QPnoise, HBnoise)” on page 45, “Input Source and Reference Side-Band (Pnoise, HBnoise)” on page 27 |
| *Do Noise* s(Displays additional fields. | “Do Noise (HBNOISE)” on page 10 |
| *Enabled* includes this analysis in the next simulation. | “Enabled” on page 11 |
| *Options* displays the Options form for this analysis. | “Options” on page 40 |

#### *Modifications to HBnoise Form for Do Noise*

Noise Type and Noise Separation (described in Do Noise (HBNOISE)) calculate transfer functions and perform noise measurements

## Options Forms

The Options form contains the parameters that are less frequently used for each analysis. For guidance on what the form contains for a particular analysis, choose the appropriate link here.

- “Envelope Following Analysis Options Form” on page 167
- “Harmonic Balance AC Options Form” on page 168
- “Harmonic Balance Noise Options Form” on page 168
- “Harmonic Balance Options Form” on page 168
- “Periodic Small-Signal Analyses Options Forms” on page 167
- “PSS Analysis Options Form” on page 165
- “QPSS Analysis Options Form” on page 166
- “Quasi-Periodic Small-Signal Analyses Options Forms” on page 167

### PSS Analysis Options Form

Time Step Parameters define the time step used for the PSS analysis.

Initial Condition Parameters define the initial conditions for the PSS analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the PSS analysis.

State File Parameters provides the locations of files associated with the PSS analysis.

Integration Method Parameters define the integration method used for the PSS analysis.

Accuracy Parameters define the level of accuracy to use for the PSS analysis. Enable/disable/refine use of the Finite difference method of the PSS analysis.

Annotation Parameters define statistics and other information recorded and displayed for the PSS analysis.

Output Parameters defines information related to the results of the PSS analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the PSS analysis.

Simulation Interval Parameters define the start time for the initial transient analysis for this PSS analysis.

### QPSS Analysis Options Form

Time Step Parameters define the time step used for the QPSS analysis.

Initial Condition Parameters define the initial conditions for the QPSS analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the QPSS analysis.

Multitone Stabilization Parameter (QPSS) specifies the number of stabilization cycles to be performed.

State File Parameters provides the locations of files associated with the QPSS analysis.

Integration Method Parameters define the integration method used for the QPSS analysis.

Accuracy Parameters define the level of accuracy to use for the QPSS analysis.

Annotation Parameters define statistics and other information recorded and displayed for the QPSS analysis.

Output Parameters defines information related to the results of the QPSS analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the QPSS analysis.

Simulation Interval Parameters define the start time for the initial transient analysis for this QPSS analysis.

### Envelope Following Analysis Options Form

Simulation Interval Parameters defines the start time, output start time, and the stabilization time period for the ENVLP analysis.

Simulation Bandwidth Parameters define the modulation bandwidth for the ENVLP analysis.

Time Step Parameters define the time step and the outer envelope size used for the ENVLP analysis.

Initial Condition Parameters define the initial conditions for the ENVLP analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the ENVLP analysis.

State File Parameters provides the locations of files associated with the ENVLP analysis.

Integration Method Parameters define the integration method used for the ENVLP analysis.

Accuracy Parameters define the level of accuracy to use for the ENVLP analysis.

Annotation Parameters define statistics and other information recorded and displayed for the ENVLP analysis.

Output Parameters defines information related to the results of the ENVLP analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the ENVLP analysis.

### Periodic Small-Signal Analyses Options Forms

Convergence Parameters provide convergence information for the small-signal analysis.

Annotation Parameters define statistics and other information recorded and displayed for the small-signal analysis.

Output Parameters defines information related to the results of the small-signal analysis.

### Quasi-Periodic Small-Signal Analyses Options Forms

Convergence Parameters provide convergence information for the small-signal analysis.

Annotation Parameters define statistics and other information recorded and displayed for the small-signal analysis.

Output Parameters defines information related to the results of the small-signal analysis.

### Harmonic Balance Options Form

Harmonic Balance Parameters specify accuracy settings for the HB analysis.

Integration Method Parameters define the integration method used for the HB analysis.

Initial Condition Parameters define the initial conditions for the HB analysis.

Convergence Parameters provide convergence information for the HB analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HB analysis.

Output Parameters defines information related to the results of the HB analysis.

### Harmonic Balance AC Options Form

Convergence Parameters provide convergence information for the HBAC analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HBAC analysis.

Output Parameters defines information related to the results of the HBAC analysis.

### Harmonic Balance Noise Options Form

Convergence Parameters provide convergence information for the HBAC analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HBAC analysis.

Output Parameters defines information related to the results of the HBAC analysis.

## Direct Plot Form

See Direct Plot Form for information on using the Direct Plot Form.

See Field Descriptions for the Direct Plot Form for descriptions of fields on the Direct Plot Form.

## ACPR Wizard

Clock Name Specifies the clock signal for the ENVLP analysis

How to Measure Selects whether to measure ACPR for a single net or between two differential nets.

Channel Definitions Select one of three channel definitions: Custom, IS-95 or W-CDMA.

Main Channel Width Specify channel width in Hz.

Adjacent frequencies list box Specifies the adjacent channel frequencies. Us the Add, Change and Delete buttons and the editing fields to modify adjacent frequencies.

Simulation Control Enter the Stabilization Time and Repetitions in the adjacent fields.

Resolution Bandwidth (Hz) and Calculate Click *Calculate* to determine the Resolution Bandwidth.

Windowing Function Selects the windowing function to use.

# 2

# Spectre RF Analyses

The Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) analyses add capabilities to the Virtuoso Spectre circuit simulator, such as direct, efficient computation of steady-state solutions and simulation of circuits that translate frequency. You use the Spectre RF analyses in combination with the Fourier analysis capability of the Spectre circuit simulator and with the Verilog®-A behavioral modeling language.

## Periodic Analyses

Spectre RF adds periodic large and small-signal analyses to Spectre simulation.

- Periodic Steady-State Analysis, PSS (Large-Signal)
- Periodic AC Analysis, PAC (Small-Signal)
- Periodic S-Parameter Analysis, PSP (Small-Signal)
- Periodic Transfer Function Analysis, PXF (Small-Signal)
- Periodic Noise Analysis, Pnoise (Small-Signal)

For details on the periodic analyses, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

Periodic Steady-State (PSS) analysis is a large-signal analysis that directly computes the periodic steady-state response of a circuit. With PSS, simulation times are independent of the time constants of the circuit, so PSS can quickly compute the steady-state response of circuits with long time constants, such as high-Q filters and oscillators. You can also sweep frequency or other variables using PSS.

After completing a PSS analysis, the Spectre RF simulator can model frequency conversion effects by performing one or more of the periodic small-signal analyses, Periodic AC analysis (PAC), Periodic S-Parameter analysis (PSP), Periodic Transfer Function analysis (PXF) and Periodic Noise analysis (Pnoise). The periodic small-signal analyses are similar to the Spectre L AC, SP, XF, and Noise analyses, but you can apply the periodic small-signal analyses to periodically driven circuits that exhibit frequency conversion. Examples of important frequency conversion effects include conversion gain in mixers, noise in oscillators, and filtering using switched-capacitors.

Therefore, with periodic small-signal analyses you apply a small signal at a frequency that may be noncommensurate (not harmonically related) to the small signal fundamental. This small signal is assumed to be small enough so that it is not distorted by the circuit.

# Quasi-Periodic Analyses

Spectre RF adds quasi-periodic large and small-signal analyses to Spectre L simulation.

- Quasi-Periodic Steady-State Analysis, QPSS (Large-Signal)
- Quasi-Periodic AC Analysis, QPAC (Small-Signal)
- Quasi-Periodic S-Parameter Analysis, QPSP (Small-Signal)
- Quasi-Periodic Transfer Function Analysis, QPXF (Small-Signal)
- Quasi-Periodic Noise Analysis, QPnoise (Small-Signal)

For details on the quasi-periodic analyses, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

Quasi-Periodic Steady-State (QPSS) analysis, a large-signal analysis, is used for circuits with multiple large tones. With QPSS, you can model periodic distortion and include harmonic effects. (Periodic small-signal analyses assume the small signal you specify generates no harmonics).

QPSS computes both a large signal, the periodic steady-state response of the circuit, and also the distortion effects of a specified number of moderate signals, including the distortion effects of the number of harmonics that you choose. In shooting, one signal must be designated large. This signal should be the signal that causes the largest amount of distortion in the system, or the signal that is non-sinusoidal. In harmonic balance, tstab should be set to yes for this signal. With QPSS, you can apply one or more additional signals at frequencies not harmonically related to the large signal, and these signals can be large enough to create distortion. Shooting requires that these signals be sinusoidal, but harmonic balance allows one additional signal to be a pulse signal.

Quasi-Periodic Noise (QPnoise) analysis is similar to a transient noise analysis, except that it includes frequency conversion and intermodulation effects. QPnoise analysis is useful for predicting the noise behavior of mixers, switched-capacitor filters and other periodically or quasi-periodically driven circuits. QPnoise analysis linearizes the circuit about the quasi-periodic operating point computed in the prerequisite QPSS analysis. It is the quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion and intermodulation.

The Quasi-Periodic AC (QPAC), Quasi-Periodic S-Parameter (QPSP) and Quasi-Periodic Transfer Function (QPXF) analyses all work in a similar way as the Spectre L AC, SP and XF analyses.

# Envelope Analysis

Envelope analysis allows RF circuit designers to efficiently and accurately predict the envelope transient response of the RF circuits used in communication systems.

For details on the Envelope analysis, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory.*

# Cosimulation with Virtuoso AMS Designer

Spectre RF cosimulation with Virtuoso AMS Designer enables the verification of full chip RF transceivers with mixed-signal baseband and digital control circuits. Using Spectre RF envelope analysis in AMS Designer, you can calculate the transient envelope response of RF circuits while simulating the rest of the system using digital event-driven simulation or mixed-signal behavioral models. Cosimulation allows fast verification of communication systems in which the system architecture is highly dependant on the performance of the individual RF blocks. Using Spectre RF periodic and quasi-periodic steady state analysis in AMS Designer enables fast and accurate simulation of digitally calibrated RF blocks. The technology makes it possible to analyze the effect of digital control on the performance of RF circuits.

For more information about cosimulating with AMS Designer, see *Virtuoso AMS Environment User Guide*.

# The Harmonic Balance and Shooting Method Simulation Engines

Spectre RF provides a choice of simulation engines between the traditional *shooting method* and the new *harmonic balance method* (HB) with most analyses. The harmonic balance engine complements the capabilities of the shooting method.

The combination of a PSS or QPSS analysis using the shooting method with a time-varying small-signal analyses is efficient for circuits that respond in a strongly nonlinear manner to the LO or the clock. Consequently, you can use the Spectre RF simulations with the shooting method to simulate strongly nonlinear circuits, such as switched-capacitor filters, switching mixers, chopper-stabilized amplifiers, PLL-based frequency multipliers, sample-and-holds, and samplers. You can use the Spectre RF simulations with the harmonic balance method to simulate weakly nonlinear circuits as well.

### Harmonic Balance Method

The harmonic balance engine supports frequency domain harmonic balance analyses. It provides efficient and robust simulation for linear and weakly nonlinear circuits. The harmonic balance engine is supported on the Solaris, Linux, HP and IBM platforms for both 32 and 64 bit architectures. See *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on the harmonic balance engine.

### Shooting Method

Spectre RF has traditionally used an engine known as the *shooting method* [kundert90] to implement periodic and quasi-periodic analyses and the envelope analysis. The shooting method is a time domain method and it is used in most descriptions and examples in this manual.

## Large vs. Small Signal Analysis

Spectre RF provides a variety of time-varying small signal analysis for both periodic and quasi-periodic circuits. These small-signal analyses accurately model the frequency translation effects of time-varying circuits. Rather than using traditional small-signal analyses for circuits that exhibit frequency translation, such as amplifiers and filters, you can simulate these circuits using time-varying small-signal analyses.

Circuits designed to translate from one frequency to another include mixers, detectors, samplers, frequency multipliers, phase-locked loops and parametric oscillators. Such circuits are commonly found in wireless communication systems.

Other circuits that translate energy between frequencies as a side effect include oscillators, switched-capacitor and switched-current filters, chopper-stabilized and parametric amplifiers, and sample-and-hold circuits. These circuits are found in both analog and RF circuits.

The quasi-periodic small-signal analyses accurately model the small signal characteristics of circuits with a quasi-periodic operating point, such as mixers with multiple LO frequencies or large RF inputs. The periodic small-signal analyses are more useful for circuits with a single fundamental frequency.

Applying a time-varying small-signal analysis is a two-step process.

First, the simulator ignores the small input or noise signals while performing a PSS or QPSS analysis to compute the steady-state response to the remaining large-signals, such as the LO or the clock. The initial PSS or QPSS analysis, linearizes the circuit about the time-varying large-signal operating point.

For each subsequent small-signal analysis, the simulator uses the time-varying operating point computed by the PSS or QPSS analysis to predict the circuit response to a small sinusoid at an arbitrary frequency. You can perform any number of small-signal analyses after calculating the time-varying large-signal operating point.

The input signals for the small-signal analyses must be sufficiently small that the circuit does not respond to them in a significantly nonlinear fashion. You should use input signals that are at least 10 dB smaller than the 1 dB compression point. This restriction does not apply to the signals you apply in the large-signal analysis.

This two-step process is widely applicable because most circuits that translate frequency react in a strongly nonlinear manner to one stimulus, usually either the LO or the clock, while they react in a weakly nonlinear manner to other stimuli such as the inputs. A mixer is a typical example. Its noise and conversion characteristics improve if it is discontinuously switched between two states by the LO, yet it must respond linearly to the input signal over a wide dynamic range.

To analyze a mixer with a small RF input and a single LO, you should use a PSS large-signal analysis followed by one or more of the PAC, PNoise, PSP or PXF small-signal analyses.

If the mixer has a small RF input and a large blocker as well as the LO, then a QPSS analysis would be the more appropriate large-signal analysis. Follow the QPSS analysis with one or more of the QPAC, QPNoise, QPSP or QPXF small-signal analyses for the RF input.

Some circuits, such as frequency dividers, generate subharmonics. PSS can simulate the large-signal behavior of such circuits if you specify the period *T* to be that of the subharmonic. For other circuits, such as delta-sigma modulators, the periodically driven circuits respond chaotically, and you must use transient analysis rather than the PSS or QPSS analyses.

With the time-varying small-signal analyses such as QPAC or PXF, unlike traditional small-signal analyses such as AC or XF, there are many transfer functions between any single input and output due to harmonics. Usually, however, only one or two harmonics provide useful information. For example, when you analyze the down-conversion mixers found in receivers, you want to know about the transfer function that maps the input signal at the RF to the output signal at the IF, which is usually the LO minus the RF.

# 3

# New HB Analyses

- HB (large-signal analysis). See “Harmonic Balance Steady State Analysis (HB)” on page 178.
- HBAC (small-signal analysis). See “Harmonic Balance AC Analysis (HBAC)” on page 188.
- HBnoise (small-signal analysis). See “Harmonic Balance Noise Analysis (HBnoise)” on page 193.

# Harmonic Balance Steady State Analysis (HB)

This analysis uses harmonic balance (in the frequency domain) to compute the response of circuits that have either one fundamental frequency (periodic steady-state, PSS) or that have multiple fundamental frequencies (Quasi-Periodic Steady State, QPSS). The simulation time required for an HB analysis is independent of the time-constants of the circuit. This analysis also determines the circuit's periodic or quasi-periodic operating point, which can then be used during a periodic or quasi-periodic time-varying small-signal analysis, such as HBAC or HBnoise.

Usually, harmonic balance (HB) analysis is a very efficient way to simulate weakly nonlinear circuits. Also, HB analysis works better than shooting analysis (in the time domain) for frequency dependent components, such as delay, transmission line, and S-parameter data.

An HB analysis consists of two phases. The first phase calculates an initial solution, which the second phase then uses to compute the periodic or quasi-periodic steady-state solution, using the Newton method.

The two most important parameters for HB analysis are `funds` and `maxharms`. The funds parameter accepts a list of names of fundamentals that are present in the sources. These names are specified in the sources by the `fundname` parameter. When only one name appears, the analysis is an HB PSS analysis. When more than one name appears, the analysis is an HB QPSS analysis. The `maxharms` parameter accepts a list of numbers of the harmonics that are required to adequately model the responses due to the different fundamentals.

## HB Synopsis

*Name* ( [*p*] [*n*] ) hb <*parameter=value*> ...

## HB Parameters

### HB Fundamental Parameters

| | |
|---|---|
| `evenodd=[...]` | Array of `even`, `odd`, or `all` strings for moderate tones to select harmonics. |
| `freqdivide` | Large signal frequency division. |
| `funds=[...]` | Array of fundamental frequency names for fundamentals to use in analysis. |
| `fundfreqs=[...]` | Array of fundamental frequencys to use in analysis. |

| | |
|---|---|
| `maxharms=[...]` | Array of number of harmonics of each fundamental to consider for each fundamental. |
| `maximorder` | Maximum intermodulation order (same parameter as `boundary'). |
| `selectharm` | Name of harmonics selection methods. Possible values are box, diamond, funnel or axis. Default is diamond when `maximorder' or `boundary' is set; otherwise, default is box. |

### Simulation Interval Parameter

| | |
|---|---|
| `tstab=0.0 s` | Extra stabilization time after the onset of periodicity for independent sources. |

### Time-Step Parameter

| | |
|---|---|
| `maxstep (s)` | Maximum time step. Default derived from `errpreset'. |

### Initial Conditions Parameters

`ic=all`

What should be used to set initial conditions. Possible values are `dc`, `node`, `dev`, or `all`.

- `ic=dc`: Any initial condition specifiers are ignored, and the DC solution is used.
- `ic=node`: The `ic` statements are used, and the `ic` parameters on the capacitors and inductors are ignored.
- `ic=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.
- `ic=all`: Both the `ic` statements and the `ic` parameters are used, and the `ic` parameters override the `ic` statements.

| | |
|---|---|
| `oscic=default` | Oscillator IC method. It determines how the starting values for the oscillator are calculated. `oscic=lin' gives you an accurate initial value, but it takes some time; `fastic' is very fast, but it is less accurate. Possible values are default, lin or fastic. |
| `readic` | File that contains initial conditions. |

| | |
|---|---|
| `skipdc=no` | Determines whether DC analysis is used for the initial transient.<br>■ `skipdc=no`: Initial solution is calculated using the normal DC analysis (default).<br>■ `skipdc=yes`: Initial solution is given in the file specified by the `readic` parameter or by the values specified on the `ic` statements.<br>■ `skipdc=sigrampup`: Independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. The waveform production in the time-varying independent source is enabled after the rampup phase. The rampup simulation is from `tstart` to time=0 s, and the main simulation is from time=0 s to `tstab`. If the `tstart` parameter is not specified, the default `tstart` time is set to 0.1 * tstab. |

### Convergence Parameters

| | |
|---|---|
| `cmin=0 F` | Minimum capacitance from each node to ground. |
| `readns` | File that contains an estimate of the initial transient solution. |

### Output Parameters

| | |
|---|---|
| `compression=no` | Do data compression on output. See full description below. Possible values are `no` or `yes`. |
| `nestlvl` | Levels of subcircuits to output. |
| `save` | Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, `none`, or `nooutput`. |
| `saveinit=no` | If `set`, saves the waveforms for the initial transient before the steady state is reached. Possible values are `no` or `yes`. |

### Integration Method Parameter

| | |
|---|---|
| `tstabmethod` | Integration method used in stabilization time. Default is `traponly` for autonomous circuits, or is derived from `errpreset` for driven circuits. Possible values are `euler`, `trap`, `traponly`, `gear2`, or `gear2only`. |

### Accuracy Parameters

| | |
|---|---|
| `errpreset` | Selects a reasonable collection of parameter settings. Possible values are `liberal`, `moderate`, or `conservative`.<br><br>The `errpreset` parameter lets you adjust the simulator parameters to fit your needs quickly. In most cases, it should also be the only parameter you need to adjust. |
| `glog=5` | Number of steps, log sweep for hbhomotopy of gsweep. |
| `gstart=1.e-7` | Start conductance for hbhomotopy of gsweep. |
| `gstop=1.e-12` | Stop conductance for hbhomotopy of gsweep. |
| `hbhomotopy=tstab` | Name of the HB homotopy selection method. Possible values are `tstab`, `source`, or `gsweep`. The value `source` is not applicable for autonomous circuits.<br><br>The homotopy method is not applicable for autonomous circuits. |
| `hbpartition_defs=[...]` | Define HB partitions. |
| `hbpartition_fundratios=[...]` | Specify HB partition fundamental frequency ratios. |
| `hbpartition_harms=[...]` | Specify HB partition harmonics. |
| `itres=1e-4` for shooting, `0.9` for HB | Rel3ative tolerance for linear solver. The value is between [0,1]. |
| `maxperiods` | Maximum number of simulated periods to reach steady-state.<br><br>The parameter `maxperiods` default value is set to 50 for HB. |
| `oscmethod=onetier` | Osc Newton method for autonomous HB. Possible values: `onetier` or `twotier`. |
| `oversample=[...]` | Array of oversample factors for each tone. It overrides oversamplefactor. |
| `pinnode` | Node to pin during autonomous HB simulation. |
| `pinnodemag` | This parameter gives an estimate of the magnitude of the pin node voltage. Defaule value is 0.01. |
| `pinnodeminus` | Second node to pin during autonomous HB simulation. Only needed when differential nodes exist in oscillator. |

| | |
|---|---|
| `pinnoderank` | Harmonic rank to pin during autonomous HB simulation. |
| `sweepic` | IC extroplation method in sweep hb analysis. Possible values are , linear, log, or none. Default value is none. |

#### Annotation Parameters

| | |
|---|---|
| `annotate=sweep` | Degree of annotation. Possible values are no, `title`, `sweep`, `status`, `steps`, `iters`, `detailed`, `rejects`, or `alliters`. |
| `stats=no` | `Stats` parameter is not supported. Use `annotate` instead. |
| `title` | Analysis title. |

#### Newton Parameter

| | |
|---|---|
| `restart=no` | If set to `yes`, restart the DC/PSS/QPSS solution from scratch. If set to `no`, reuse the previous solution as the initial guess. Possible values are `no` or `yes`. |

#### Circuit Age Parameter

| | |
|---|---|
| `circuitage (Years)` | Age or stress time of the circuit, used to simulate hot-electron degradation of MOSFET and BSIM circuits. |
| `readhb` | File from which final harmonic steady-state solution must be read. Small signal analyses such as hbac and hbnoise can read in the steady-state solution from this file directly instead of running the hb analysis again. |
| `writehb` | File to which final harmonic balance steady-state solution is to be written. Small signal analyses such as hbac and hbnoise can read in the steady-state solution from this file directly instead of running the hb analysis again. |

#### Annotation Parameters

| | |
|---|---|
| `recover` | Specifies the file to be restored. |
| `saveclock (s)` | Save the tran analysis periodically on the wall clock time. Default is 1800s for spectre. The feature is disabled in turbo mode by default. |
| `savefile` | Save the analysis states into the specified file. |

| | |
|---|---|
| `saveperiod` | Save the tran analysis periodically on the simulation time. |
| `savetime=[...]` | Save the analysis states into files on the specified time points. |

## Details about Using HB Analysis Parameters

The initial transient analysis provides a flexible mechanism to direct the circuit to a particular steady-state solution of interest and to avoid undesired solutions. The initial transient simulation also helps convergence by eliminating the large, but fast decaying, modes that are present in many circuits.

In some circuits, the linearity of the relationship between the initial and final states depends on when HB analysis begins. In practice, starting analysis at a good point can improve convergence, and starting at a bad point can degrade convergence and slow the analysis.

### cmin Parameter

If the circuit you are simulating can have infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), the simulator might not converge well. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves convergence.

### compression Parameter

The default value for `compression` is `no`. The output file stores data for every signal at every time point for which Spectre calculates a solution. Spectre saves the x axis data only once, because every signal has the same x value. If `compression=yes`, Spectre writes data to the output file only if the signal value changes by at least 2 * the convergence criteria. To save data for each signal independently, x axis information corresponding to each signal must be saved. If the signals stay at constant values for large periods of the simulation time, setting `compression=yes` results in a smaller output data file. If the signals move around a lot, setting `compression=yes` results in a larger output data file.

### funds Parameter

For the `funds` parameter, the frequencies associated with fundamentals are figured out automatically by the simulator. An important feature is that each input signal can be a composition of more than one source. However, these sources must have the same fundamental name. For each fundamental name, the fundamental frequency is the greatest common factor of all frequencies associated with the name. Omitting a fundamental name in the `funds` parameter is an error that stops the simulation. If `maxharms` is not given, a warning message is issued, and the number of harmonics defaults to 1 for each of the fundamentals.

#### hbhomotopy Parameter

The convergence rate of large signal analyses is mainly determined by two factors: the initial condition and the nonlinearity of the circuit. When the initial condition is close to the true solution and the nonlinearity is small, convergence is typically fast. One way to calculate the initial condition is by running a transient analysis but with this method it is hard to determine how long the transient analysis needs to run.

The homotopy method uses a different approach. Given a circuit with strong nonlinearity, the homotopy method obtains a solution by starting from the same circuit but with an altered, much lower nonlinearity. Then the method increases the nonlinearity, using the solution from the lower nonlinearity as the initial condition. When the nonlinearity of the circuit reaches the original nonlinearity, a good initial condition is available from the previous solution so convergence is fast. Source stepping is used to change the nonlinearity of the circuit.

A third approach is available when neither of the preceding methods is successful. In this method, a resistor and a parallel capacitor are added between each circuit node and the ground. These components reduce the dynamic range of the circuit and help with convergence. Initially, this method obtains a solution by assuming a small resistor value. Then the method increases the resistance and obtains a new solution, using as the initial value the solution obtained on the previous iteration. When the resistance becomes large enough, the inserted components become negligible and the iteratively derived solution provides a good guess of the true solution.

The parameter used to control HB homotopy is called `hbhomotopy`. The parameter has three possible values: `source, tstab` (which is the default), and `gsweep`.

- For the `source` value, the initial condition is generated by stepping the RF source level. This value is appropriate when the power level of the large tone or of the RF tones is high. Because there is no source available in autonomous circuits, this value is not applicable to oscillators.
- For the `tstab` value, the initial condition is generated by the usual transient analysis described above. This value is appropriate when the circuit contains devices, such as digital frequency dividers, that display strong non-linear behavior.
- The `gsweep` value is appropriate for situations where convergence fails for both the `source` and `tstab` values.

### ic Parameter

You can specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

When you specify initial conditions, the simulator computes the actual initial state of the circuit by performing a DC analysis. During this analysis, the simulator forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see options).

With the `ic` statement, it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground or setting the current through a branch that is not an inductor. If you initialize the simulator inconsistently, its solution jumps; that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes because the simulator can have convergence problems while trying to make the jump.

Initial conditions and nodesets have similar implementations but produce different effects. Initial conditions actually define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. However, differential equations have an infinite number of solutions, and a complete set of initial conditions must be specified to identify the desired solution. Any initial conditions you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator toward finding the solution closest to the nodeset values.

### oscic Parameter

When HB analysis is used for oscillators, initialization is performed to obtain an initial guess of the steady state solution and of the oscillating frequency. Two initialization methods are implemented, based on transient and linear analysis.

- When `oscic=default` is specified, transient initialization is used and the length of the transient is specified by `tstab`. You must start the oscillator using initial conditions or using a brief impulsive stimulus, just as you would if you were simulating the turn-on transient of the oscillator using transient analysis. Initial conditions would be provided for the components of the oscillator’s resonator. If an impulsive stimulus is used, it should be applied so as to couple strongly into the oscillatory mode of the circuit and poorly into any other long-lasting modes, such as those associated with bias circuitry. The Designer's Guide to Spice and Spectre [K. S. Kundert, Kluwer Academic Publishers, 1995] describes in depth some techniques for starting oscillators.

- When `oscic=lin` is specified, linear initialization is used. In this method, both oscillation frequency and amplitude are estimated based on linear analysis at the DC solution. No impulsive stimulus or initial conditions are needed. Linear initialization is suitable for linear oscillators such as LC and crystal oscillators. Note that `tstab` transient is still performed after linear initialization, though it can be significantly shortened or skipped. Either way, specifying a non-zero `tstab` parameter can improve convergence.

#### readns Parameter

Nodesets help the simulator find the DC or initial transient solution. You can supply nodesets in the circuit description file with `nodeset` statements, or in a separate file using the `readns` parameter. When nodesets are given, Spectre computes an initial guess of the solution by performing a DC analysis while forcing the specified values onto nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the true solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator toward computing the desired one. Second, they are a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or dramatically speed convergence.

Nodesets and initial conditions have similar implementations but produce different effects. Initial conditions actually define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. However, differential equations have an infinite number of solutions, and a complete set of initial conditions must be specified to identify the desired solution. Any initial conditions you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator toward finding the solution closest to the nodeset values.

#### skipdc Parameter

You can skip the DC analysis entirely by using the `skipdc` parameter. If the DC analysis is skipped, the initial solution will be either trivial or given in the file you specified by the `readic` parameter, or, if the `readic` parameter is not given, the values specified on the `ic` statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

# Harmonic Balance AC Analysis (HBAC)

The harmonic balance AC (HBAC) analysis computes transfer functions for circuits that exhibit single or multi-tone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like. HBAC is a small-signal analysis like AC analysis, except the circuit is first linearized about a periodically or quasi-periodically varying operating point rather than about a simple DC operating point. Linearizing about a periodically or quasi-periodically time-varying operating point allows transfer-functions that include frequency translation, whereas simply linearizing about a DC operating point cannot because linear time-invariant circuits do not exhibit frequency translation. Also, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response of a periodically or quasi-periodically varying circuit is a two step process. First, the small stimulus is ignored and the periodic or quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed using HB analysis. As a normal part of the HB analysis, the periodically or quasi-periodically time-varying representation of the circuit is computed and saved for later use. Second, the small stimulus is applied to the periodically varying linear representation to compute the small signal response. This is done using the HBAC analysis. A HBAC analysis cannot be used alone, it must follow a HB analysis. However, any number of periodic or quasi-periodic small-signal analyses, such as HBAC or HBnoise, can follow a HB analysis.

Modulated small signal measurements are possible using the Analog Design Environment (ADE). The `modulated` option for HBAC and other modulated parameters are set by ADE. HBAC analyses with this option produce results that can have limited use outside of ADE. Direct Plot is configured to analyze these results and combine several wave forms to measure AM and PM response due to single sideband or modulated stimuli. For details, see the *Virtuoso Spectre Circuit Simulator RF Analysis User Guide*.

Unlike other analyses in Spectre, the HBAC analysis can sweep only frequency.

## HBAC Synopsis

*Name* `hbac` <*parameter=value*> ...

## HBAC Parameters

### Sweep Interval Parameters

| | |
|---|---|
| `center` | Center of sweep. |
| `dec` | Points per decade. |

| | |
|---|---|
| `lin=50` | Number of steps, linear sweep. |
| `log=50` | Number of steps, log sweep. |
| `relharmvec=[...]` | Sideband - vector of QPSS harmonics - to which a relative frequency sweep should be referenced. |
| `span=0` | Sweep limit span. |
| `start=0` | Start sweep limit. |
| `step` | Step size, linear sweep. |
| `stop` | Stop sweep limit. |
| `sweeptype= unspecified` | Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. When the `unspecified` value is used, Spectre RF sweeps the absolute input source for non-PSP-driven cases; for other cases, Spectre RF sweeps relative to the port harmonics. Possible values are `absolute`, `relative`, or `unspecified`. |
| `values=[...]` | Array of sweep values, which specifies particular values that the sweep parameter should take. If you give both a set of values specified with the parameter and a set specified using a sweep range, the two sets are merged and collated before being used. |

#### Output Parameters

| | |
|---|---|
| `freqaxis=absout` | Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the output frequency. Possible values are `absout`, `out`, or `in`. |
| `maxsideband=0` | An alternative to the sidebands array specification, which automatically generates the array: [ -maxsideband ... 0 ... +maxsideband ]. The `maxsideband` parameter is ignored in HB small signal when the value is larger than the `harms` or `maxharms` of large signal. |
| `nestlvl` | Levels of subcircuits to output. |
| `save` | Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, `none`, or `nooutput`. |
| `sidevec=[...]` | Array of relevant sidebands for the analysis. |

### Convergence Parameters

| | |
|---|---|
| `lnsolver=gmres` | Linear solver. Possible values are `gmres`, `qmr`, `bicgstab`, or `resgmres`. |
| `resgmrescycle= short` | Restarted gmres cycle. Possible values are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, or `recyclelong`. |
| `tolerance` | Relative tolerance for the linear solver. The default value is 1.0e-9 for shooting-based solver; 1.0e-6 for driven, and 1.0e-4 for autonomous for flexbalance-based solver. |

### Annotation Parameters

| | |
|---|---|
| `annotate=sweep` | Degree of annotation. Possible values are no, `title`, `sweep`, `status`, or `steps`. |
| `stats=no` | `Stats` parameter is not supported. Use `annotate` instead. |
| `title` | Analysis title. |

### Modulation Conversion Parameters

| | |
|---|---|
| `contriblist="NULL"` | Array of device names for distortion summary. When `contriblist=[""]`, the simulator calculates the distortion from each non-linear device. |
| `fim_out=0 Hz` | Frequency of IM output signal. |
| `flin_out=0 Hz` | Frequency of linear output signal. |
| `inmodharmnum=1` | Harmonic for the PAC input source modulation. |
| `maxharm_nonlin=4` | Maximum harmonics of input signal frequency induced by non-linear effect. |
| `modsource` | Refer the output noise to this component. |
| `modulated=no` | Compute transfer functions/conversion between modulated sources and outputs. Possible values are `single`, `first`, `second`, or no. |
| `moduppersideband=1` | Index of the upper sideband included in the modulation of an output for PAC or an input for PXF. |
| `out1="NULL"` | Output signal 1. |
| `out2="NULL"` | Output signal 2. |

| | |
|---|---|
| `outmodharmvec=[...]` | Harmonic list for the PAC output modulations. |
| `perturbation=linear` | The type of PAC analysis. Default is `linear` for normal PAC analysis, `im2ds` for im2 distortion summary, and `ds` for distortion summary. Possible values are `linear`, `ds`, `ip3`, `ip2`, or `im2ds`. |
| `rf1_src="NULL"` | Array of RF1 source names for IP3/IP2/IM2. |
| `rf2_src="NULL"` | Array of RF2 source names for IP3/IP2/IM2. |
| `rfdbm=0` | RF source dBm. |
| `rfmag=0` | RF source magnitude. |

## Details about Using HBAC Analysis Parameters

### freqaxis Parameter

With HBAC, the frequency of the stimulus and of the response are usually different—this is one important way that HBAC differs from AC. The `freqaxis` parameter is used to specify whether the results should be output versus the input frequency (`in`), the output frequency (`out`), or the absolute value of the output frequency (`absout`).

### maxsideband and sidevec Parameters

You can select the set of periodic small-signal output frequencies of interest by setting either the `maxsideband` or the `sidevec` parameters.

When there is only **one tone in HB analysis**, sidebands are n integer numbers, K1, K2, ..., Kn, and the output frequency at each sideband is computed as

```
f(out)= f(in) + Ki * fund(hb)
```

where `f(in)` represents the (possibly swept) input frequency, and `fund(hb)` represents the fundamental frequency used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is Ki= -1. When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is Ki= 1. By setting the `maxsideband` value to Kmax, all 2 * Kmax + 1 sidebands from -Kmax to +Kmax are generated.

When there are **multiple tones in HB analysis**, sidebands are vectors. Assume we have one large tone and one moderate tone in HB. A sideband, K1, is represented as [K1_1 K1_2]. The corresponding frequency is

```
K1_1 * fund(large tone of HB) + K1_2 * fund(moderate tone of HB)
```

We assume that there are L large and moderate tones in HB analysis and a given set of n integer vectors representing the sidebands, K1 = { K1_1, ..., K1_j, ..., K1_L}, K2, ..., Kn. The output frequency at each sideband is computed as

```
f(out)= f(in) + SUM_j=1_to_L{Ki_j * fund_j(hb)}
```

where `f(in)` represents the (possibly swept) input frequency, and `fund_j(hb)` represents the fundamental frequency used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is {-1, 0}. When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is {1, 0}. You enter `sidevec` as a sequence of integer numbers, separated by spaces. The set of vectors {1 1 0} {1 -1 0} {1 1 1} becomes `sidevec=[1 1 0 1 -1 0 1 1 1]`. For `maxsideband`, only the large tone, the first fundamental, is affected by this entry. All the other tones, the moderate tones, are limited by `maxharms`, specified for a HB analysis. Given maxharms=[k1max k2max ... knmax] in HB and maxsideband=Kmax, all

```
(2*Kmax + 1)*(2*k2max+1)*(2*k3max+1) *...*(2*knmax+1)
```

sidebands are generated.

The number of requested sidebands has a significant impact on the simulation time.

### Sweep Interval Parameters

You can specify sweep limits by giving the end points or by providing the center value and the span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can give a step size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step size parameter, the sweep is linear when the ratio of stop to start values is less than 10 and logarithmic when this ratio is 10 or greater. All frequencies are in Hertz.

# Harmonic Balance Noise Analysis (HBnoise)

The harmonic balance periodic or quasi-periodic noise (HBnoise) analysis is similar to conventional noise analysis, except that HBnoise analysis includes frequency conversion effects. As a consequence, it is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically driven circuits. It is particularly useful for predicting the phase noise of autonomous circuits, such as oscillators.

HBnoise analysis linearizes the circuit about the periodic or quasi-periodic operating point computed in the prerequisite HB analysis. It is the periodically or quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion. In addition, the effect of a periodically or quasi-periodically time-varying bias point on the noise generated by the various components in the circuit is also included.

The time-average of the noise at the output of the circuit is computed in the form of a spectral density versus frequency. The output of the circuit is specified with either a pair of nodes or a probe component. To specify the output of a circuit with a probe, specify it using the `oprobe` parameter. If the output is voltage (or potential), choose a resistor or a port as the output probe. If the output is current (or flow), choose a vsource or iprobe as the output probe.

The noise analysis always computes the total noise at the output, including contributions from the input source and the output load. The amount of the output noise that is attributable to each noise source in the circuit is also computed and output individually. If the input source is identified (using `iprobe`) and is a vsource or isource, the input-referred noise, which includes the noise from the input source itself, is computed. Finally, if the input source is identified (using `iprobe`) and is noisy, as is the case with ports, the noise factor and noise figure are computed. Thus if

`No` = total output noise
`Ns` = noise at the output due to the input probe (the source)
`Nsi` = noise at the output due to the image harmonic at the source
`Nso` = noise at the output due to harmonics other than input at the source
`Nl` = noise at the output due to the output probe (the load)
`IRN` = input referred noise
`G` = gain of the circuit
`F` = noise factor
`NF` = noise figure
`Fdsb` = double sideband noise factor
`NFdsb` = double sideband noise figure
`Fieee` = IEEE single sideband noise factor
`NFieee` = IEEE single sideband noise figure

then,

```
IRN = sqrt(No^2/G^2)
F = (No^2 - Nl^2)/Ns^2
NF = 10*log10(F)
Fdsb = (No^2 - Nl^2)/(Ns^2+Nsi^2)
NFdsb = 10*log10(Fdsb)
Fieee = (No^2 - Nl^2 - Nso^2)/Ns^2
NFieee = 10*log10(Fieee)
```

When the results are output, `No` is named `out`, `IRN` is named `in`, `G` is named `gain`, `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee` are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee` respectively.

The computation of gain and IRN for quasi-periodic noise in HBnoise assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the gain and IRN computation.

An HBnoise analysis must follow an HB analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

## HBnoise Synopsis

`Name ( [p] [n] ... ) hbnoise <parameter=value> ...`

The optional terminals (*p* and *n*) specify the output of the circuit. If you do not give the terminals, then you must specify the output with a probe component.

## HBnoise Parameters

### Sweep Interval Parameters

| | |
|---|---|
| `start=0` | Start sweep limit. |
| `stop` | Stop sweep limit. |
| `center` | Center of sweep. |
| `span=0` | Sweep limit span. |
| `step` | Step size, linear sweep. |
| `lin=50` | Number of steps, linear sweep. |
| `dec` | Points per decade. |
| `log=50` | Number of steps, log sweep. |

| | |
|---|---|
| `values=[...]` | Array of sweep values, which specifies the particular values that the sweep parameter should take. If you give both a set of values specified with this parameter and a set specified using a sweep range, the two sets are merged and collated before being used. |
| `sweeptype= unspecified` | Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps the absolute input source for non-PSP-driven cases; for other cases, Spectre RF sweeps relative to the port harmonics. Possible values are `absolute`, `relative`, or `unspecified`. |
| `relharmvec=[...]` | Sideband - vector of QPSS harmonics - to which relative frequency sweep should be referenced. |

### Probe Parameters

| | |
|---|---|
| `oprobe` | Compute total noise at the output defined by this component. |
| `iprobe` | Refer the output noise to this component. |
| `refsideband=[...]` | Conversion gain associated with this sideband is used when computing input-referred noise or noise figure. |
| `refsidebandoption= individual` | Whether to consider the input at the frequency or the input at the individual quasi-periodic sideband specified. Possible values are `freq` or `individual`.<br><br>Note that different sidebands can lead to the same frequency. |

### Output Parameters

| | |
|---|---|
| `noisetype=sources` | Specifies if the pnoise analysis should output cross-power densities or noise source information. Possible values are `sources`, `correlations`, `timedomain`, or `pmjitter`. |
| `maxsideband=7` | Maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal. It is ignored in HB small signal when it's larger than the harms/maxharms of large signal. |
| `save` | Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, `none`, or `nooutput`. |
| `nestlvl` | Levels of subcircuits to output. |

| | |
|---|---|
| `cycles=[...]` | Array of relevant cycle frequencies. Valid only if `noisetype=correlations`. |
| `saveallsidebands=no` | Save noise contributors by sideband. Possible values are no or `yes`. |
| `xfonly=no` | Do XF analysis only. Possible values are no or `yes`. |
| `stimuli=sources` | Stimuli used for xf analysis in hbnoise. Possible values are `sources` or `nodes_and_terminals`. |
| `separatenoise=no` | Separate noise into sources and transfer functions. Possible values are no or `yes`. |
| `cyclo2txtfile=no` | Output cyclo-stationary noise to text file as input source of next stage. Possible values are no or `yes`. |

### Convergence Parameters

| | |
|---|---|
| `tolerance` | Relative tolerance for linear solver, default value is 1.0e-9 for shooting-based solver; 1.0e-6 for driven and 1.0e-4 for autonomous for flexbalance-based solver. |
| `lnsolver=gmres` | Linear solver. Possible values are `gmres`, `qmr`, `bicgstab`, or `resgmres`. |
| `resgmrescycle=short` | Restarted gmres cycle. Possible values are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, or `recyclelong`. |
| `ppv=no` | If `yes`, save the oscillator PPV after doing noise analysis. Possible values are no or `yes`. |

### Annotation Parameters

| | |
|---|---|
| `annotate=sweep` | Degree of annotation. Possible values are no, `title`, `sweep`, `status`, or `steps`. |
| `stats=no` | `Stats` parameter is not supported. Use `annotate` instead. |
| `title` | Analysis title. |

### iprobe Parameter

If you want the input-referred noise or noise figure, specify the input source using the `iprobe` parameter. For input-referred noise, use either a vsource or isource as the input probe; for noise figure, use a port as the probe. Currently, only a vsource, an isource, or a port can be used as an input probe. If the input source is noisy, as is a port, the noise analysis computes the noise factor (F) and noise figure (NF). To match the IEEE definition of noise figure, the input probe must be a port with no excess noise and its noisetemp must be set to 16.85C (290K). In addition, the output load must be a resistor or port and must be identified as the oprobe.

If port is specified as the input probe, then both input-referred noise and gain are referred back to the equivalent voltage source inside the port. S-parameter analysis calculates those values.

#### maxsideband Parameter

In practice, noise can mix with each of the harmonics of the periodic drive signal applied in the HB analysis and end up at the output frequency. However, the HBnoise analysis only includes the noise that mixes with a finite set of harmonics that are typically specified using the `maxsideband` parameter.

If Ki represents sideband i, then for periodic noise,

```
f(noise_source) = f(out) + Ki * fund(hb)
```

For quasi-periodic noise with multiple tones in HB analysis, assuming there are one large tone and one moderate tone, Ki is represented as [Ki_1 Ki_2]. The corresponding frequency shift is

```
Ki_1 * fund(large tone of HB) + Ki_2 * fund(moderate tone of HB)
```

Assuming that there are L large and moderate tones in HB analysis and a set of n integer vectors representing the sidebands

```
K1 = { K1_1,...K1_j..., K1_L}, K2, ..., Kn.
```

Then

```
f(noise_source) = f(out) +  SUM_j=1_to_L{ Ki_j * fund_j(hb) }
```

The `maxsideband` parameter specifies the maximum |Ki| included in the HBnoise calculation. For quasi-periodic noise, only the large tone, the first fundamental, is affected by this entry. All the other tones, the moderate tones, are limited by `maxharms`, specified for a HB analysis.

The number of requested sidebands changes the simulation time substantially.

When HBnoise analysis does only an xf analysis (`xfonly=yes`), the variable of interest at the output can be voltage or current, and its frequency is not constrained by the period of the large periodic solution. While sweeping the selected output frequency, you can select the periodic small-signal input frequencies of interest by setting the `maxsideband` parameter.

With this analysis, the frequency of the stimulus and of the response are usually different (this is an important way that this analysis differs from XF).

#### refsideband Parameter

The reference sideband (`refsideband`) specifies which conversion gain is used when computing input-referred noise, noise factor, and noise figure. The reference sideband specifies the input frequency relative to the output frequency with:

```
|f(input)| = |f(out) + refsideband frequency shift|
```

For periodic noise (only **one tone in HB analysis**), the `refsideband` is a number. Use `refsideband=0` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters). When `refsideband` differs from 0, the single side-band noise figure is computed.

While for quasi-periodic noise (**multiple tones in HB analysis**), reference sidebands are vectors. Assume we have one large tone and one moderate tone in HB. A sideband Ki will be a vector [Ki_1 Ki_2]. It gives the frequency at

```
Ki_1 * fund(large tone of HB) + Ki_2 * fund(moderate tone of HB)
```

Use `refsideband=[0 0 ...]` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters).

#### stimuli Parameter

You can use the `stimuli` parameter to specify what serves as the inputs for the transfer functions.

- `stimuli=sources` indicates that the sources present in the circuit are to be used. You can use the `xfmag` parameters provided by the sources to adjust the computed gain to compensate for gains or losses in a test fixture. You can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters.
- `stimuli=nodes_and_terminals` indicates that all possible transfer functions are to be computed. This is useful when you do not know in advance which transfer functions are interesting. Transfer functions for nodes are computed assuming that a unit magnitude flow (current) source is connected from the node to ground. Transfer functions for terminals are computed assuming that a unit magnitude value (voltage) source is connected in series with the terminal. By default, the transfer functions from a small set of terminals are computed. If you want transfer functions from specific terminals, specify the terminals in the `save` statement. You must use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the `options` statement. If you want

transfer functions from all terminals, specify `currents=all` and `useprobes=yes` on the `options` statement.

With HBnoise analysis, transfer function outputs are always available except when `noisetype=correlations`.

### Sweep Interval Parameters

You can specify sweep limits by giving the end points or by providing the center value and the span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can give a step size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step size parameter, the sweep is linear when the ratio of stop to start values is less than 10, and logarithmic when this ratio is 10 or greater. All frequencies are in Hertz.

### xfonly Parameter

When option `xfonly` is set to `yes`, HBnoise analysis does only a conventional transfer function (xf) analysis, which computes the transfer function from every source in the circuit to a single output. This analysis differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit. HBnoise computes the transfer functions from any source at any frequency to a single output at a single frequency. Thus, like HBAC analysis, it includes frequency conversion effects. It directly computes such useful quantities as conversion efficiency (transfer function from input to output at desired frequency), image and sideband rejection (input to output at undesired frequency), and LO feed-through and power supply rejection (undesired input to output at all frequencies).

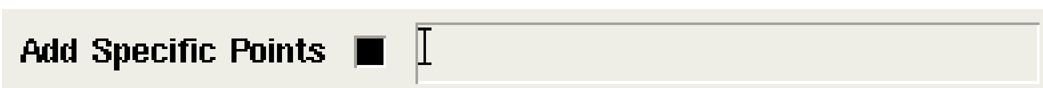
## Field Descriptions for the Choosing Analyses Form

The following sections describe the panes and fields that ever appear on the Choosing Analyses form, independently of the analysis that is selected. The descriptions are arranged alphabetically, according to the labels that are usually found along the left side of the form. If you are looking for descriptions of the Choosing Analyses form as they appear for a particular analysis, see “Choosing Analyses Form” on page 215.

The following sections are:

## Add Specific Points (HBAC, HBNOISE)

Specifies specific points to be added to the set of swept values.




If more than one value is entered, separate the values with spaces. For example, you might type

```
3 13
```

into the field to add two points to the sweep defined by `start` = 0, `stop` = 20, `step` =5, so that the swept values are 0, 3, 5, 10, 13, 15, 20.

## Additional Time for Transient-Aided HB (tstab) (PSS, QPSS, HB)

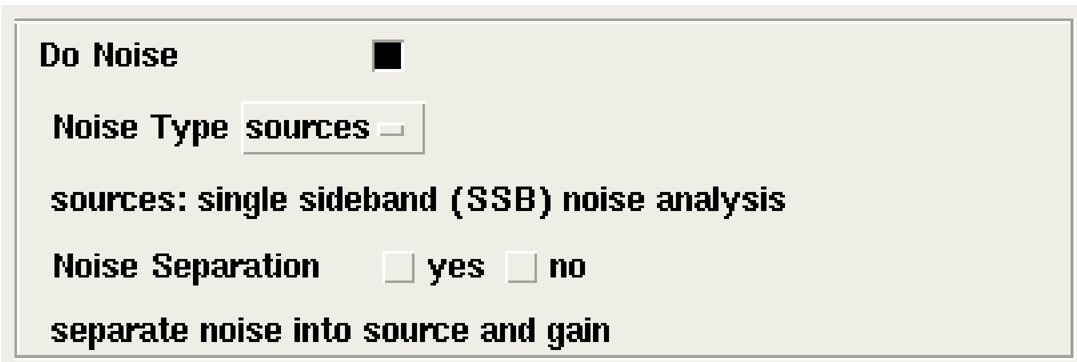
For the harmonic balance engine, specifies an amount of additional time to allow for the circuit to settle.


Use `tstab` if the circuit exhibits more than one periodic solution and you want only one. A long `tstab` can also improve convergence.

## Do Noise (HBNOISE)

Calculates transfer functions and performs noise measurements during the HBnoise analysis.

The only supported noise type for this release is *sources*.




Highlight *yes* for *Noise Separation* to calculate the contributions that noise sources make to the output. In addition to the noise contributions, the simulator also determines the transfer functions. When you highlight *no* for *Noise Separation*, the transfer functions are determined but the noise contributions are not.

## Harmonic Balance Homotopy Method (HB)

Determines the method used to pursue convergence in a circuit.


The *Harmonic Balance Homotopy Method* choices are:

- *tstab* – Runs a transient analysis to determine initial conditions for the circuit. This value is appropriate when the circuit contains devices, such as digital frequency dividers, that display strong non-linear behavior.
- *source* – Determines the initial conditions by decreasing the nonlinearity of the circuit and then gradually stepping up the nonlinearity to its original value. Each time the nonlinearity is stepped, the solution from the previous iteration is used as the starting condition. This value is appropriate when the power level of the large tone or of the RF tones is high.

  The *source* value must not be used for autonomous circuits.

- *gsweep* – Determines the initial conditions by reducing and then incrementing the dynamic range of the circuit. The *gsweep* value is appropriate for situations where convergence fails for both the *source* and *tstab* values.

## Input Frequency Sweep Range (HBAC)

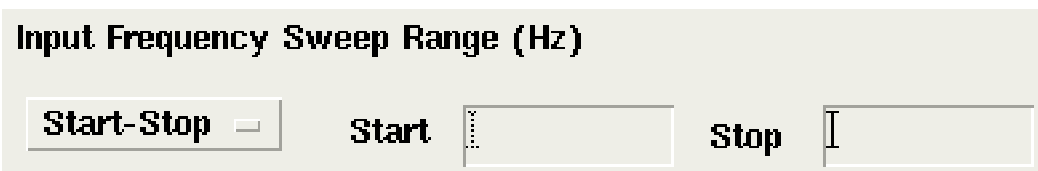
Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

For small-signal analyses following a swept PSS or QPSS analysis, *Single-Point* and *Freq* are the only *Frequency Sweep Range (Hz)* options for the small-signal analyses.

When you make a selection from the *Input Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

### *Start - Stop*

Defines the beginning and ending points for the sweep.



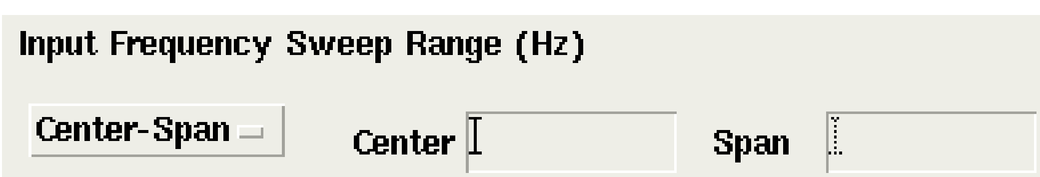

1. Select *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

### *Center - Span*

Defines the center point for the sweep and its span.



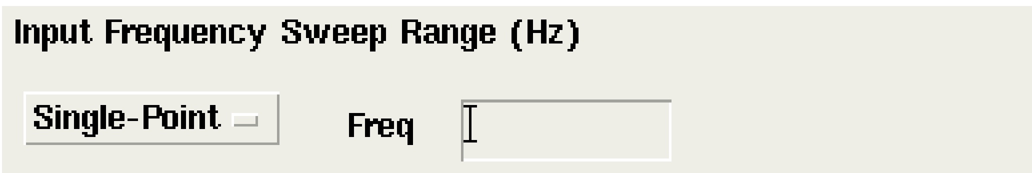

1. Select *Center-Span*.

The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

#### *Single - Point and Freq*

Defines the frequency range as a single point and prompts you for the point value.




1. Select *Single-Point*.

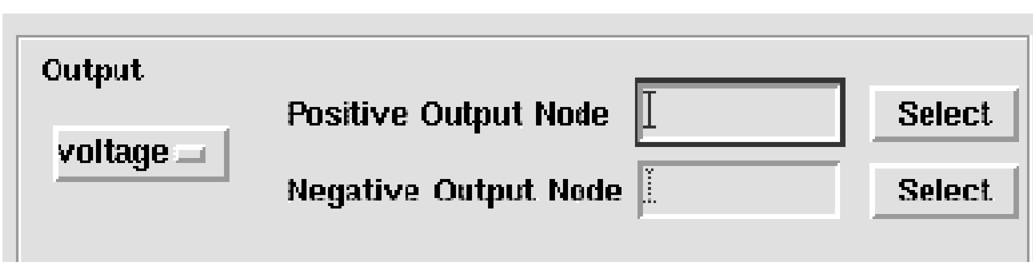
   The form changes to let you type the frequency.
2. Type the specific frequency for the small-signal analysis.

## Output (Pnoise, QPnoise, HBnoise)

The *Output* cyclic field lets you specify the output for the Pnoise, QPnoise, and HBnoise analyses. Choices are *voltage* or *probe*.

#### Voltage

The analysis computes the noise voltage across the two nodes.




1. Select *Voltage*.

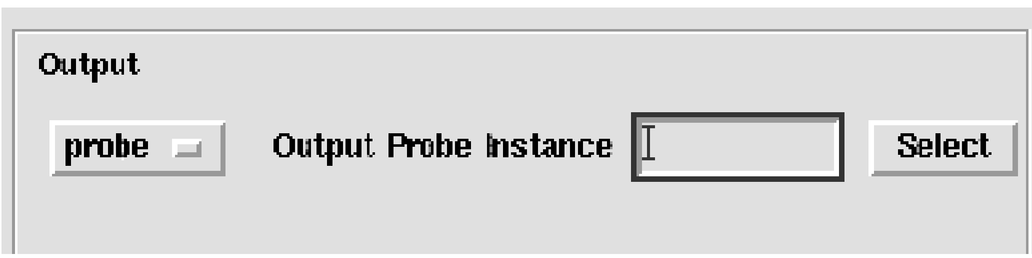
   The form changes to let you specify a *Positive Output Node* and a *Negative Output Node*.

2. Specify the nodes by either
   - ❑ Typing a node name into the type-in field.
   - ❑ Clicking the adjacent *Select* button and then clicking the appropriate node in the schematic.

   If you leave the *Negative Output Node* field empty, it defaults to gnd.

#### Probe

The analysis computes the noise voltage across the port. The noise contribution of the port is subtracted during the noise figure calculation.




1. Select *Probe*.

   The form changes to let you specify the *Output Probe Instance*.
2. Specify the node either by
   - ❑ Typing the node name into the type-in field.
   - ❑ Clicking the adjacent *Select* button and then clicking the appropriate node in the schematic.

### Output Frequency Sweep Range (HBnoise)

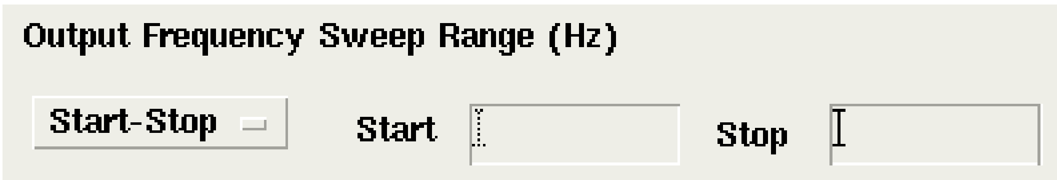
Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

For small-signal analyses following a swept PSS or QPSS analysis, *Single-Point* and *Freq* are the only *Frequency Sweep Range (Hz)* options for the small-signal analyses.

When you make a selection from the *Output Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

#### *Start - Stop*

Defines the beginning and ending points for the sweep.



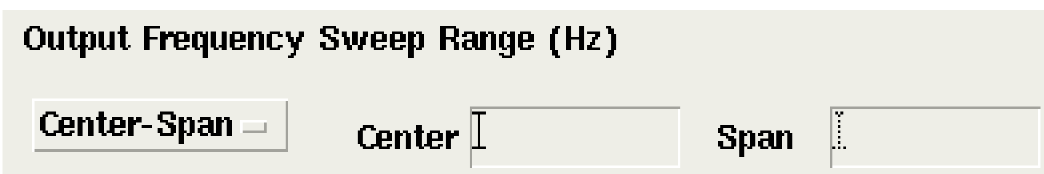

1. Select *Start-Stop*.

   The form changes to let you type the start and stop points.

2. Type the initial point for the sweep in the *Start* field.
3. Type the final point in the *Stop* field.

#### *Center - Span*

Defines the center point for the sweep and its span.



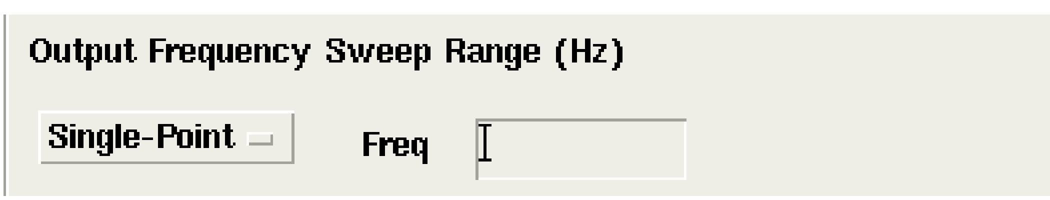

1. Select *Center-Span*.

   The form changes to let you type the center point and span.

2. Type the midpoint for the sweep in the *Center* field.
3. Type the span in the *Span* field.

#### *Single - Point and Freq*

Defines the frequency range as a single point and prompts you for the point value.



1. Select *Single-Point*.

   The form changes to let you type the frequency.

2. Type the specific frequency for the small-signal analysis.

## Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)

Lets you select the set of periodic small-signal output frequencies of interest. When you select from the *Sidebands* cyclic field, the form changes to let you specify appropriate data.

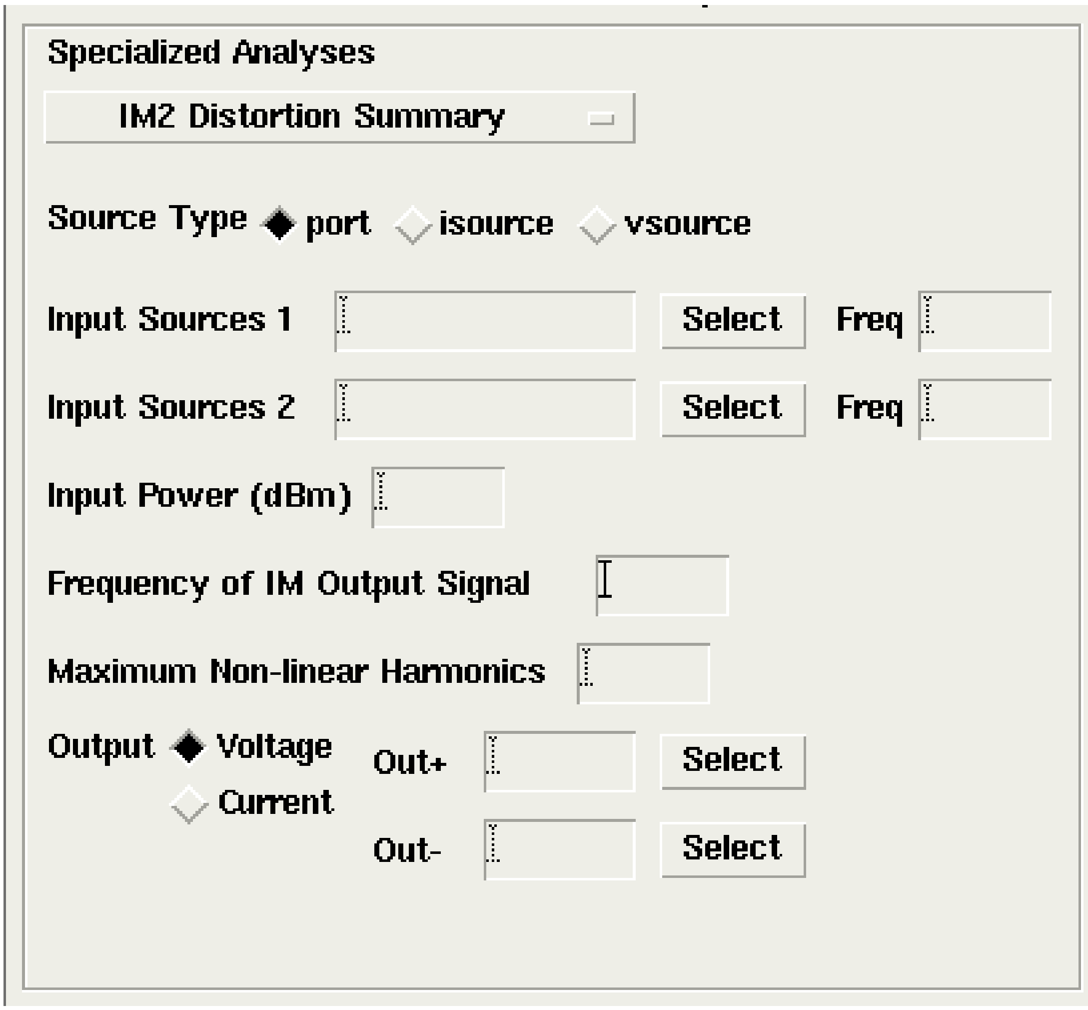
For HBAC, the choices are: *Maximum sideband*, *Select from range*, *Array of coefficients*. For the other analyses, the choices also include *Array of indices*

You can use the choices in the *Sidebands* cyclic field in combination. For example, if you add a sideband using *Array of coefficients*, the sideband value you added appears in the *Select from range* list box and as a *Currently active index* for the *Array of indices*.

### IM2 Distortion Summary

When you select *IM2 Distortion Summary* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *IM2 Distortion Summary* include the following.




#### *IM2 Distortion Summary Terms and Data Entry Fields*

- **Source Type** – Specifies whether the source is a port, current, or voltage. Select *port, isource*, or *vsource*.
- **Input Sources 1** and **Freq** fields – The RF source magnitude and frequency. Click *Select* and select the RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Start* field for the PAC analysis.

- **Input Sources 2** and **Freq** fields – A second RF source magnitude and frequency. Click *Select* and select the second RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Stop* field for the PAC analysis.

  The RF1 source and the RF2 source can be the same. The associated frequency values must be different.

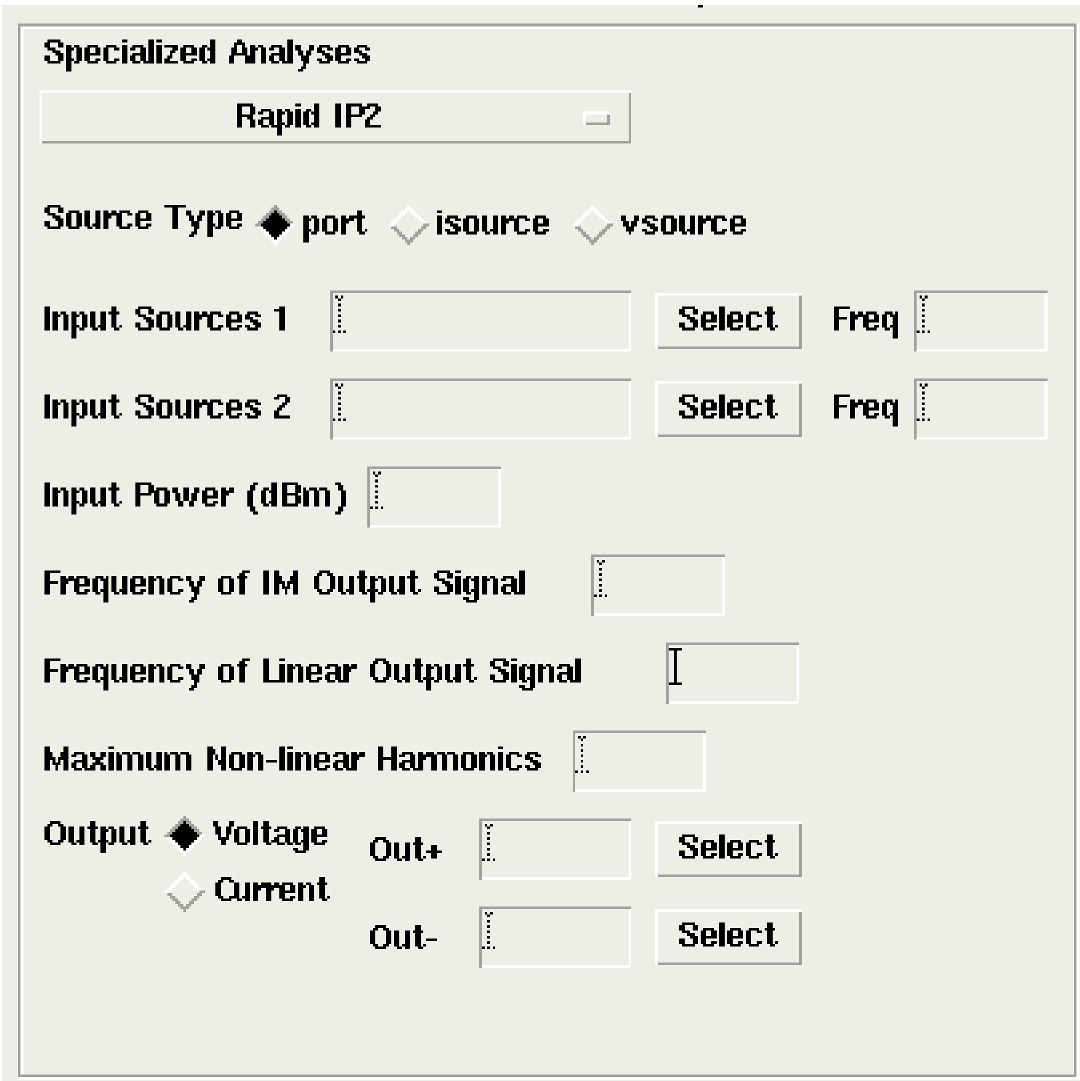
- **Input Power (dBm)** – RF source power.
- **Frequency of IM Output Signal** – IM3 frequency at the output.
- **Maximum Non-linear Harmonics** – Number of harmonics used for the RF signals. Default is 4.
- **Output** – **Voltage** Output node 1 where IP3 is measured.
- **Out+** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a net in the schematic.
- **Out-** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a second net in the schematic. The default is *gnd!*.
- **Output** – **Current** Selects a terminal in the schematic.
- **Term -** (Displays when you choose *Current* for *Output*.) For output current in a source. Specify it as `<source_name>:p`.

  When the output is current in a port, you must use the ADE *Outputs - To be saved - Select in schematic* menu pick and select the terminal in the schematic where current is to be computed and saved.

### Rapid IP2

When you select *Rapid IP2* in the *Specialized Analyses* cyclic field, the form changes to let you enter more information.

The *Specialized Analyses* fields for the *Rapid IP2* analysis include the following.




#### *Rapid IP2 Terms and Data Entry Fields*

- **Source Type** – Specifies whether the source is a port, current, or voltage. Select *port, isource*, or *vsource*.

- **Input Sources 1** and **Freq** fields – The RF source magnitude and frequency. Click *Select* and select the RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Start* field for the PAC analysis.
- **Input Sources 2** and **Freq** fields – A second RF source magnitude and frequency. Click *Select* and select the second RF source in the schematic. In the *Freq* field, type the frequency for the source. The *Freq* value is copied to the *Stop* field for the PAC analysis.

  The RF1 source and the RF2 source can be the same. The associated frequency values must be different.
- **Input Power (dBm)** – RF source power.
- **Frequency of IM Output Signal** – IM3 frequency at the output.
- **Frequency of Linear Output Signal** – IM1 frequency at the output.
- **Maximum Non-linear Harmonics** – Number of harmonics used for the RF signals. Default is 4.
- **Output** – **Voltage** Output node 1 where IP3 is measured.
- **Out+** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a net in the schematic.
- **Out-** (Displays when you choose *Voltage* for *Output*.) Click *Select* and select a second net in the schematic. The default is *gnd!*.
- **Output** – **Current** Selects a terminal in the schematic.
- **Term -** (Displays when you choose *Current* for *Output*.) For output current in a source. Specify it as `<source_name>:p`.

  When the output is current in a port, you must use the ADE *Outputs - To be saved - Select in schematic* menu pick and select the terminal in the schematic where current is to be computed and saved.

## Sweep (PSS, QPSS, HB)

Specifies how a sweep is performed. Choices are: *Variable*, *Temperature*, *Component Param*, and *Model Param*.

When you activate *Sweep* on the PSS or QPSS analysis form, the *Frequency Sweep Range* on the small-signal analysis forms is restricted to a single point.

## Sweeptype (PAC, PXF, HBAC, HBNOISE)

Controls the inclusion of the *sweeptype* newlink parameter in the Spectre netlist. Choices are: *absolute*, *relative*, and *default*, with *default* selecting the appropriate Spectre RF default.

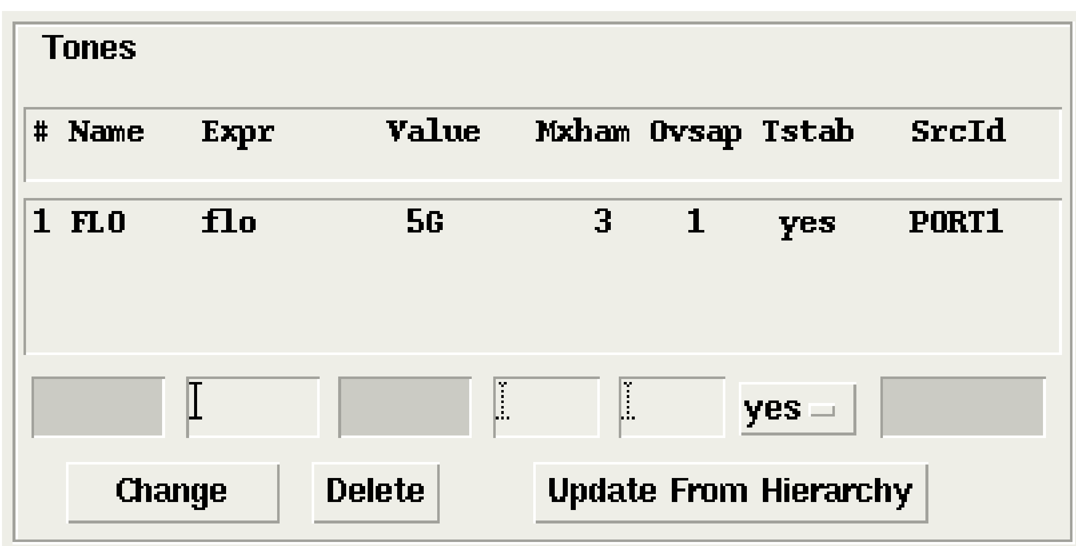
The results vary depending on whether you are simulating an autonomous circuit (an oscillator) or a driven circuit (a mixer) as determined by the *Oscillator* button selection on the PSS or HB Choose Analyses form.

In general,

- When you simulate an autonomous circuit (the *Oscillator* section of the PSS or HB Choosing Analyses form is active), you can select *relative* for *Sweeptype*. If *Sweeptype* is set to *default*, the effective value is *relative*.
- When you simulate a driven circuit (the *Oscillator* section of the PSS or HB Choosing Analyses form is not active), you can select either *relative* or *absolute* for *Sweeptype*. If *Sweeptype* is set to *default*, the defective valuse is *absolute*.

## Tones (HB)

The *Tones* fields include the following list box, data entry fields, and data entry buttons.



### Tones List Box

The *Tones* list box displays information about every top-level tone in the circuit that has both a non-zero frequency or period value, and a non-zero amplitude value (absolute). The tones in the list box are arranged alphabetically by name.

To edit values for a tone, highlight the tone in the list box, then edit in the data entry fields.

For tones that are not at the top level of the schematic, you can manually create a tone entry by typing the pertinent information in the data entry fields.

For non-small-signal tones, each tone name and its correlated frequency value are used in the *Select from range* and *Array of coefficients* choices of the *Sidebands* field in the HBAC and HBnoise small-signal analysis forms.

### Fundamental Tones List Box Terms and Data Entry Fields

- **Name** – Displays the name assigned to the tone. This tone name must be entered into the pertinent Component Description Format (CDF) fields of each source in the schematic that has a tone. The CDF name field prompts are *First frequency name*, *Second frequency name, Frequency name*, and *Frequency name for 1/period*.
- **Expr** – Displays the value or expression representing the frequency of a particular tone. The expression can also be a user variable or it can contain user variables. If the frequency for the tone is specified as a variable, the *Expr* field displays the name of the variable. Otherwise, the field displays the numerical value of the frequency.
- **Value** – Displays the evaluated value of the *Expr* field using the current values of the user variables.
- **Mxham** – The maximum harmonic for the tone.
- **Ovsap** – A factor that determines the amount by which the tone is oversampled. Used to decrease aliasing effects. This parameter defaults to 1 and can be a positivie integer. It is typically used for circuits that have sharp edges or sudden transitions. Values range from 1 for mostly sinusoidal circuits to 4 (or rarely larger) for circuits with vewry sharp edges in them.
- **Tstab** – ???
- **SrcId** – Displays the instance name of the source in the schematic where the tone is declared.

# Field Descriptions for the Options Forms

The following sections describe all the simulation parameters whose values you specify on Options forms. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the form.

## Convergence Parameters (All)

**cmin** is the minimum capacitance from each node to ground. Default is 0.

**gear_order** is the order used for Gear-type interpolations. Default is 2 (second order).

**hbhighq** activates the harmonic balance algorithm for high Q oscillators, providing a robust way of finding sound initial conditions for high Q oscillators. Default is no. This parameter is available only when *Osc Newton method* is set to *twotier*.

**lnsolver** (PAC) specifies the linear solver to be used.

> **bicgstab** specifies that the biconjugate gradient stabilized (bicgstab) variant of the conjugate gradient (cg) solver is to be used. The bicgstab solver is formulated for nonsymmetric linear systems. The bicgstab solver uses less memory than either gmres or qmr but is the least robust of the solvers.
>
> **gmres** specifies that the general minimum residual (gmres) linear solver is to be used. This is the default. If memory issues arise when using this solver, consider using either bicgstab or qmr, which use less memory.
>
> **qmr** specifies that the quasi minimal residual (qmr) linear solver is to be used. The qmr solver uses less memory than gmres but is not as robust.
>
> **resgmres** specifies that the restarted GMRES solver (resgmres) linear solver is to be used. The resgmres solver can reduce memory cost and simulation time.

**oscsolver** lets you specify the type of solver to use for an oscillator circuit. Possible values are `std`, `turbo` or `ira`. The default is `turbo`.

> **ira** uses the implicitly restarted Arnoldi algorithm to calculate the dominant eigenvalue and the corresponding eigenvector for small-signal analysis of oscillator circuits. `Ira` uses less memory than `turbo`.
>
> **std** uses a full eigen analysis to calculate the dominant eigenvalue and the corresponding eigenvector. Using the `std` solver for small-signal analysis of oscillator circuits is slower but more robust than simulation using the `turbo` setting. Use the `std` setting if simulation with the `turbo` setting is unsuccessful.

**turbo** uses the Arnoldi algorithm based on the Krylov subspace to calculate the dominant eigenvalue and the corresponding eigenvector for oscillator small-signal analysis. `Turbo` is significantly faster than `std` and uses significantly less memory. In rare situations `turbo` might be *less* accurate.

**readns** lets you specify the name of a file that contains an estimate of the initial transient solution. Enter the complete path to the file. No default.

**resgmrescycle** ??? restarted gmres cycle.

**instant**

**short**

**long**

**recycleinstant**

**recycleshort**

**recyclelong**

**solver** lets you specify the type of solver to use for a linear system. Possible values are `std` or `turbo`. The default is `turbo`.

**std** for each frequency value, solves the linear system using the full GMRES algorithm. Simulations using the `std` solver are slower but more robust than simulations using the turbo setting. Use the `std` setting if simulation with the `turbo` setting is unsuccessful.

**turbo** for each frequency value, solves the linear system using the recycled Krylov subspace algorithm. Using `turbo` is significantly faster than using `std`. In rare situations `turbo` might be *less* accurate.

**tolerance** is the relative tolerance for the linear solver when solving for convergence. Default is $10^{-9}$.

## Harmonic Balance Parameters (HB)

**maxperiods** is the maximum number of simulated periods allowed for the simulation to reach steady-state. Default is `50`.

**itres** sets the relative tolerance for the linear solver. Default is 0.9.

**freqdivide** sets the frequency division ratio of a large signal. The default is 1.

### Output Parameters (All)

**compression** directs the simulator to perform data compression on the output. Default is *no.*

**cyclo2txtfile** outputs cyclo-stationary noise to a text file as an input source for a succeeding stage.

**enable osc ppv** is equivalent to *save osc ppv*.

**freqaxis** specifies what version of the frequency to plot the output against in spectral plots.

- ❑ For the PAC and QPAC analysis

  *absout* is the absolute value of the output frequency.
  *in* is the input frequency.
  *out* is the output frequency.

- ❑ For the PXF and QPXF analysis

  *absin* is the absolute value of the input frequency.
  *in* is the input frequency.
  *out* is the output frequency.

- ❑ For the PSP and QPSP analysis

  *absin* is the absolute value of the frequency swept at the input.
  *in* is the scattered frequency at the input.
  *out* is the scattered frequency at the output.

**nestlvl** specifies the levels of subcircuits to output. The field is activated by choosing the *lvl* or *lvlpub* values for the *save* field.

## Field Descriptions for the Direct Plot Form

The following sections describe the fields on the Direct Plot form. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the forms.

### Choosing Analyses Form

The Choosing Analyses form changes depending on which analysis is selected. For guidance on what the form contains for a particular analysis, choose the appropriate link here.

- “Harmonic Balance (HB) Choosing Analyses Form” on page 216

- "Harmonic Balance AC (HBAC) Choosing Analyses Form" on page 217
- "Harmonic Balance Noise (HBnoise) Choosing Analyses Form" on page 217

#### Harmonic Balance (HB) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | Analysis on page 7 |
| *Tones* displays and edits information for top level tones in the circuit. | "Tones (HB)" on page 211 |
| *Accuracy Defaults* quickly adjusts simulation parameters. | "Accuracy Defaults (errpreset) (PSS, QPSS, ENVLP, and HB)" on page 6 |
| *Additional Time for Transient-Aided HB* allows time for stabilization. | "Additional Time for Transient-Aided HB (tstab) (PSS, QPSS, HB)" on page 200 |
| *Save Initial Transient Results* saves the initial transient solution. | "Save Initial Transient Results (PSS, QPSS, HB)" on page 52 |
| *Harmonic Balance Homotopy Method* determines the method used to pursue convergence in a circuit. | "Harmonic Balance Homotopy Method (HB)" on page 201 |
| *Oscillator* defines the simulation for an oscillator circuit. (Displays additional fields to specify oscillator analysis.) | "Oscillator (PSS, ENVLP, HB)" on page 40 |
| *Sweep* selects swept analysis. (Displays additional fields to specify sweep.) | "Sweep (PSS, QPSS, HB)" on page 210 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to HB Form for Oscillator Analysis*

Oscillator Node and Reference Node specify how the HB oscillator analysis is performed. Osc initial condition determines how the starting values for the oscillator are determined.

#### *Modifications to HB Form for Swept Analysis*

Sweep, Sweep Range, Sweep Type, Add Specific Points, and New Initial Value For Each Point (HB) specify how the HB sweep is performed.

### Harmonic Balance AC (HBAC) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | Analysis on page 7 |
| *Sweeptype*, *Input Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 211, "Input Frequency Sweep Range (HBAC)" on page 202, "Sweep Type (PSTB, HBAC, HBNOISE)" on page 80, "Add Specific Points (HBAC, HBNOISE)" on page 200 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 206 |
| *Specialized Analyses* measure AM and PM small-signal effects. | "Specialized Analyses (PAC, HBAC)" on page 65 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

### Harmonic Balance Noise (HBnoise) Choosing Analyses Form

| Field or Pane | User Interface Help (page) |
|---|---|
| *Analysis* selects the type of analysis to set up. | "Options" on page 40 |

| **Field or Pane** | **User Interface Help (page)** |
|---|---|
| *Sweeptype*, *Output Frequency Sweep Range*, *Sweep Type*, and *Add Specific Points* set up the sweep for the small-signal analysis. | "Sweeptype (PAC, PXF, HBAC, HBNOISE)" on page 211, "Output Frequency Sweep Range (HBnoise)" on page 204, "Sweep Type (PSTB, HBAC, HBNOISE)" on page 80, "Add Specific Points (HBAC, HBNOISE)" on page 200 |
| *Sidebands* selects the set of periodic small-signal output frequencies of interest. | "Sidebands (PAC, Pnoise, PXF, HBAC, HBNOISE)" on page 206 |
| *Output*, *Input Source*, and *Reference Side-Band* selects the output, noise generator, and reference sidebands for the Pnoise analysis. (Displays additional fields. | "Output (Pnoise, QPnoise, HBnoise)" on page 203, "Input Source and Reference Side-Band (Pnoise, HBnoise)" on page 27 |
| *Do Noise* s(Displays additional fields. | "Do Noise (HBNOISE)" on page 201 |
| *Enabled* includes this analysis in the next simulation. | "Enabled" on page 11 |
| *Options* displays the Options form for this analysis. | "Options" on page 40 |

#### *Modifications to HBnoise Form for Do Noise*

Noise Type and Noise Separation calculate transfer functions and perform noise measurements

## Options Forms

The Options form contains the parameters that are less frequently used for each analysis. For guidance on what the form contains for a particular analysis, choose the appropriate link here.

- "QPSS Analysis Options Form" on page 166
- "QPSS Analysis Options Form" on page 166
- "Envelope (ENVLP) Choosing Analyses Form" on page 154
- "Periodic Small-Signal Analyses Options Forms" on page 167
- "Quasi-Periodic Small-Signal Analyses Options Forms" on page 167
- "Harmonic Balance Options Form" on page 219

### Harmonic Balance Options Form

Harmonic Balance Parameters specify accuracy settings for the HB analysis.

Integration Method Parameters define the integration method used for the HB analysis.

Initial Condition Parameters define the initial conditions for the HB analysis.

Convergence Parameters provide convergence information for the HB analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HB analysis.

Output Parameters defines information related to the results of the HB analysis.

### Harmonic Balance AC Options Form

Convergence Parameters provide convergence information for the HBAC analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HBAC analysis.

Output Parameters defines information related to the results of the HBAC analysis.

### Harmonic Balance Noise Options Form

Convergence Parameters provide convergence information for the HBAC analysis.

Annotation Parameters define statistics and other information recorded and displayed for the HBAC analysis.

Output Parameters defines information related to the results of the HBAC analysis.

# 4

# Setting Up for the Examples

This chapter explains how to set up your software and environment to run the examples in this user guide. This chapter describes the procedure for accessing the latest version of Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF), which ships in the MMSIM release stream.

Before you perform the Spectre RF analyses, you need to set up the component files and start the Cadence® software.

## Setting Up Environment Variables and the Path Statement

Set the following environment variables for Spectre RF.

```
setenv CDS_rfExamples ‘cds_root icms‘
setenv CDS_Netlisting_Mode Analog
```

`CDS_rfExamples` defines the path to the piece-wise linear (PWL) model files in the Cadence software installation hierarchy.

## Using Spectre RF from the MMSIM Hierarchy

Starting with the 5.1.41 USR1 release, you have the option to obtain the Virtuoso Spectre Circuit Simulator (Spectre), Spectre RF, and other simulators, from the MMSIM release stream. While the version of Spectre and Spectre RF that shipped with 5.1.41 USR1 continue to ship with the remaining 5.1.41 releases, documentation for new features and most bug fixes are provided exclusively with the MMSIM release stream. The first MMSIM release, MMSIM6.0, was released at the same time as the 5.1.41 USR1 update.

You must download and install the MMSIM simulators in a separate simulation installation hierarchy than the hierarchy you use for the Cadence software.

The Spectre RF examples use the following environment variables to point to the two installation hierarchies,

| | |
|---|---|
| `MMSIMHOME` | Path to the installation hierarchy for the MMSIM simulators |
| `CDSHOME` | Path to the installation hierarchy for the Cadence software |

To use the MMSIM simulators, put all paths to the MMSIM simulators such as

`<path_to_MMSIM_simulators>/tools/bin`

before any paths to the dfII software in your $PATH statement.

The Spectre RF examples use the following path

```
path = ( $MMSIMHOME/tools/bin \
$CDSHOME/tools/bin  \
$CDSHOME/tools/dfII/bin \
$CDSHOME/tools/java \
$CDSHOME/tools/java/bin \
$path )
```

## Accessing the Most Current Spectre RF Documentation

The documentation for the latest features of Spectre RF is always found in the MMSIM hierarchy. If you are using the MMSIM version of Spectre RF, access the Spectre RF documentation from the MMSIM hierarchy.

*Important*

Note that the help buttons on the forms lead you to the IC version of the documentation.

## Creating a Local Editable Copy of the rfExamples Library

Make a copy of the *rfExamples* library and save it in a directory in your account. Change the name and access mode of your local copy so you have write access to your local copy of the library. This permits you to edit the schematics and other files in this library as you follow the examples.

The *rfExamples* library is located at

`<CDSHOME>/tools/dfII/samples/artist/rfExamples`

where `CDSHOME` is the installation directory for your Cadence software.

# Setting Up the Cadence Libraries

The Cadence Libraries are defined in the UNIX text file *`cds.lib`*. You can edit this file by using the library path editor or by using a UNIX shell window.

## Using the Library Path Editor

To access the Library Path Editor, use the following procedure.

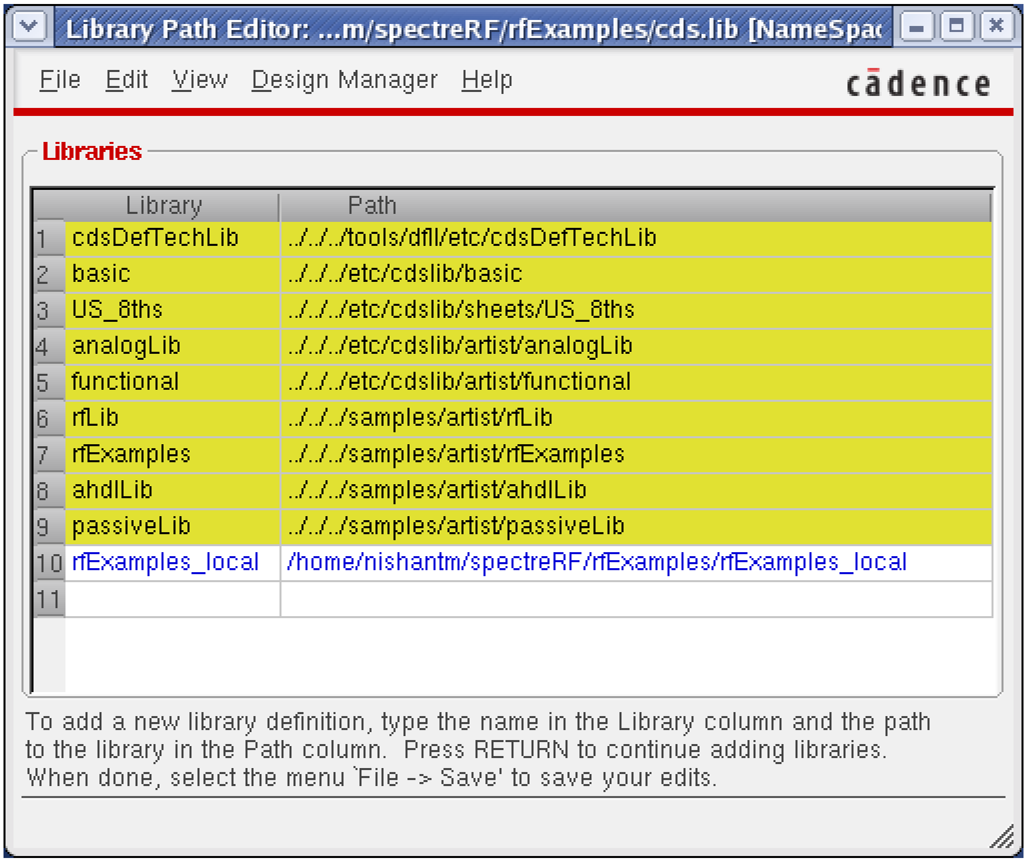
1. In a UNIX window, type `virtuoso &` to start the Cadence software.

   The Command Interpreter Window (CIW) appears.


2. In the CIW, choose *Tools – Library Path Editor*.

   The Library Path Editor appears.

### Library Path Editor Window




In the Library Path Editor, follow the instructions at the bottom of the form.

3. Type a name for each required library and the associated path to the library in the software installation hierarchy.You need the libraries listed in “Library Path Editor Window” on page 224.

4. Choose *File – Save* to save your definitions in the `cds.lib` file.

   When you are using Open Access (OA), in addition to saving the `cds.lib` file an OA `lib.defs` file is also saved.

5. Exit the Library Path Editor.

## Using a UNIX Shell Window

To set up the libraries in a UNIX shell window, use the following procedure:

1. In a UNIX shell window, open the `cds.lib` file for editing using *vi*, *emacs* or a similar text editor.

The `cds.lib` file is in your installation directory.

2. In the `cds.lib` file, define the Cadence provided libraries.
3. Define a user library where the sample circuits can be tested.

   You can label your test library any name you choose. This example assumes that you have called the library *`my_rfExamples`*. The name *`my_dir`* represents the directory into which you copied the *rfExamples* library.

   You must use the names *`basic`*, *`sample`* and *`analogLib`*. You cannot rename these libraries.

   After these steps, the definitions in the *`cds.lib`* file look similar to the following:

```
DEFINE rfExamples $CDSHOME/tools/dfII/samples/artist/rfExamples
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
DEFINE ahdlLib $CDSHOME/tools/dfII/samples/artist/ahdlLib
DEFINE rfLib $CDSHOME/tools/dfII/samples/artist/rfLib
DEFINE my_rfExamples /home/belinda/my_libs/rfExamples
DEFINE sample $CDSHOME/tools/dfII/samples/cdslib/sample
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
DEFINE passiveLib $CDSHOME/tools/dfII/samples/artist/passiveLib
DEFINE pllLib $CDSHOME/tools/dfII/samples/artist/pllLib
```

# Setting Up For Simulation

## Opening a Circuit in the Schematic Window

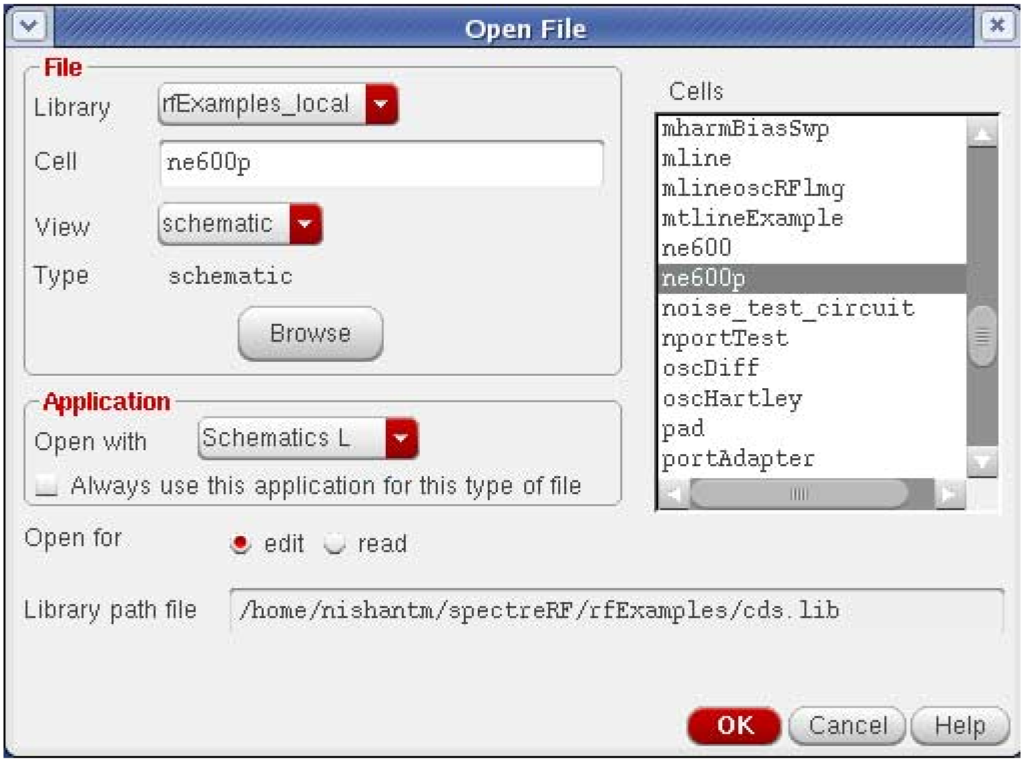
To open a circuit in the schematic window,

1. In the CIW, choose *File – Open*.

   The Open File form appears.

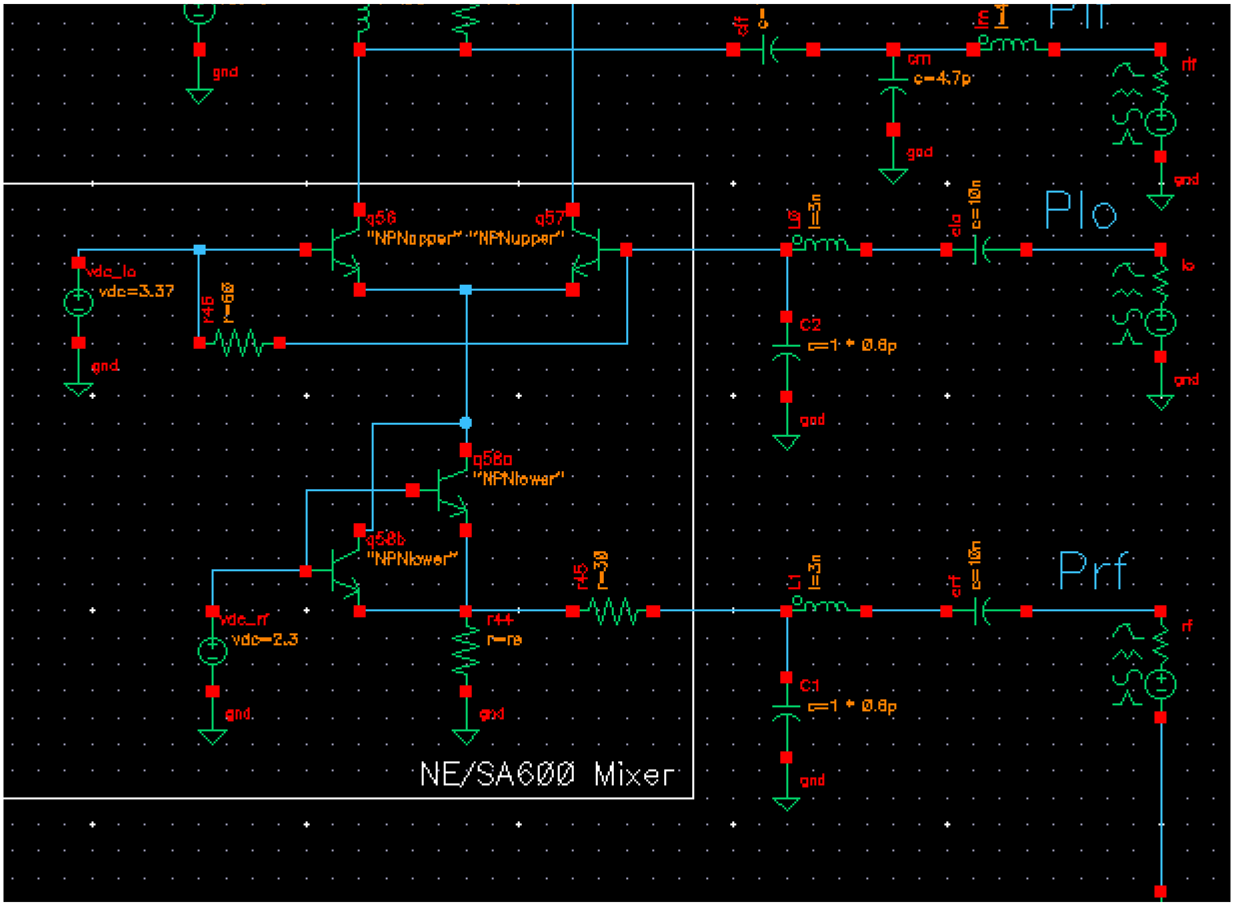
2. In the Open File form, choose *my_rfExamples* in the *Library Name* menu.
3. Choose the circuit you want to work with from the list in the *Cell Names* list box. the *ne600p* circuit is used in this example.

The completed Open File form appears like the one below.




**4.** Click *OK*.

The Schematic window opens to display the circuit. In this case, the *ne600p* mixer appears.

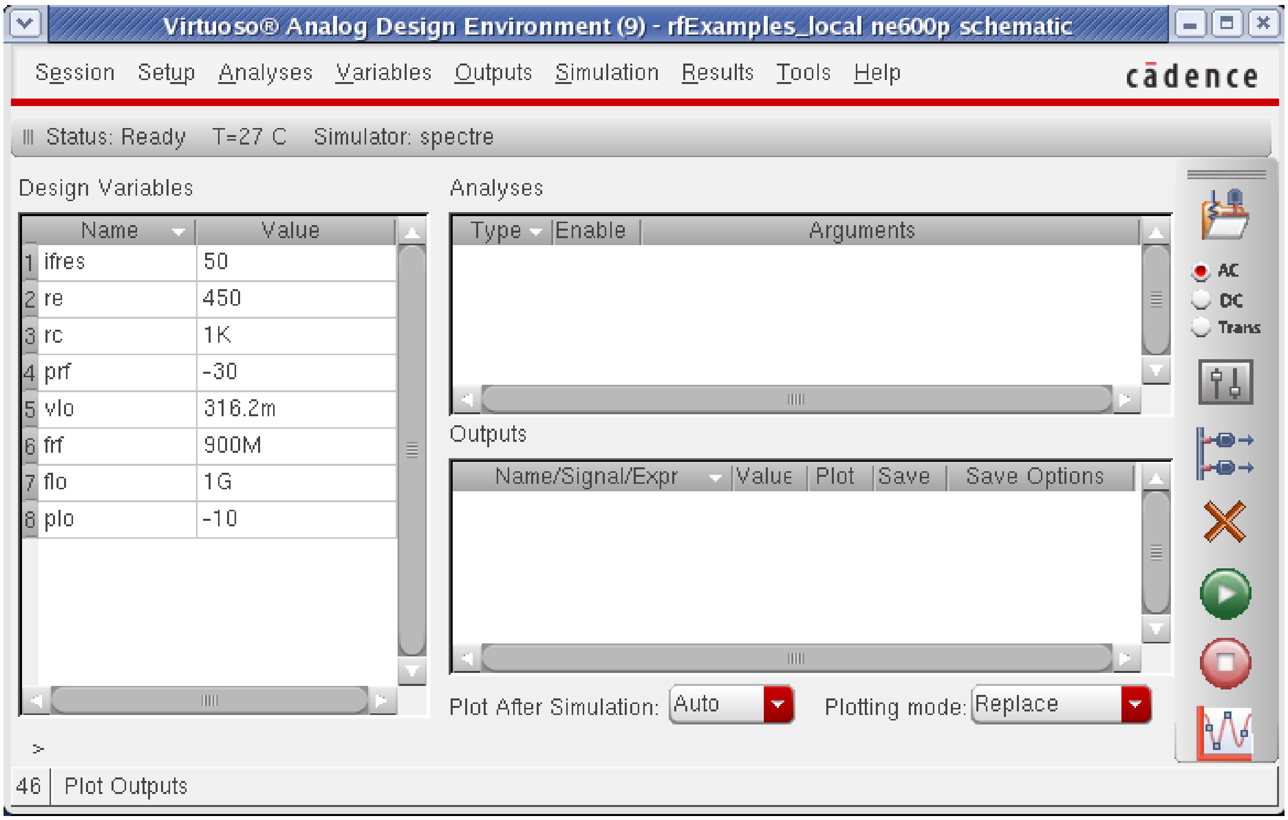



## Opening the Simulator Window

To open the Simulator window,

1. In the Schematic window, choose *Tools– Analog Environment*.

The ADE window opens. This window is also called the Cadence® Analog Circuit Design Environment.




**Note:** You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and then choosing *ne600p* in the Choosing Design form.

## Choosing Simulator Options

To set up the simulator options,

1. Choose *Setup – Simulator/Directory/Host* in the ADE window.

   The Simulator/Directory/Host form appears.

2. In the Simulator/Directory/Host form, specify the following:

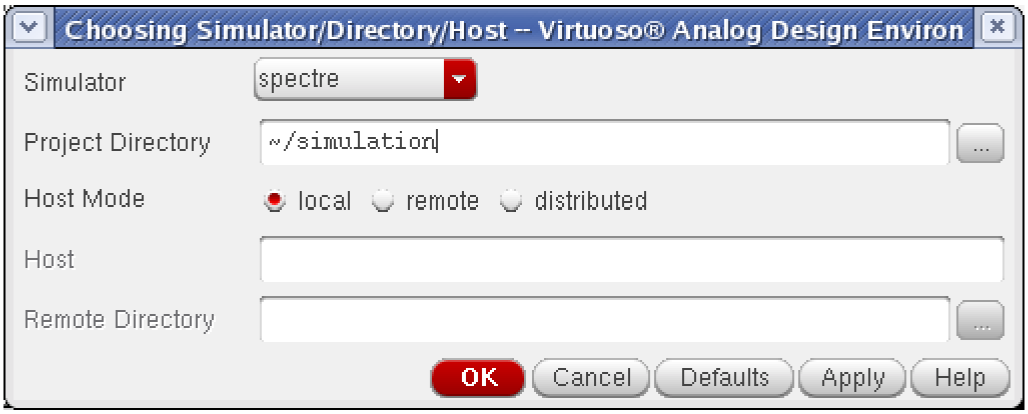
   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

**c.** Highlight the *local* or the *remote* button to specify the *Host Mode*.

**d.** (Optional) For remote simulation, type the name of the host machine and the remote directory in the appropriate fields.

The completed form appears like the one below.



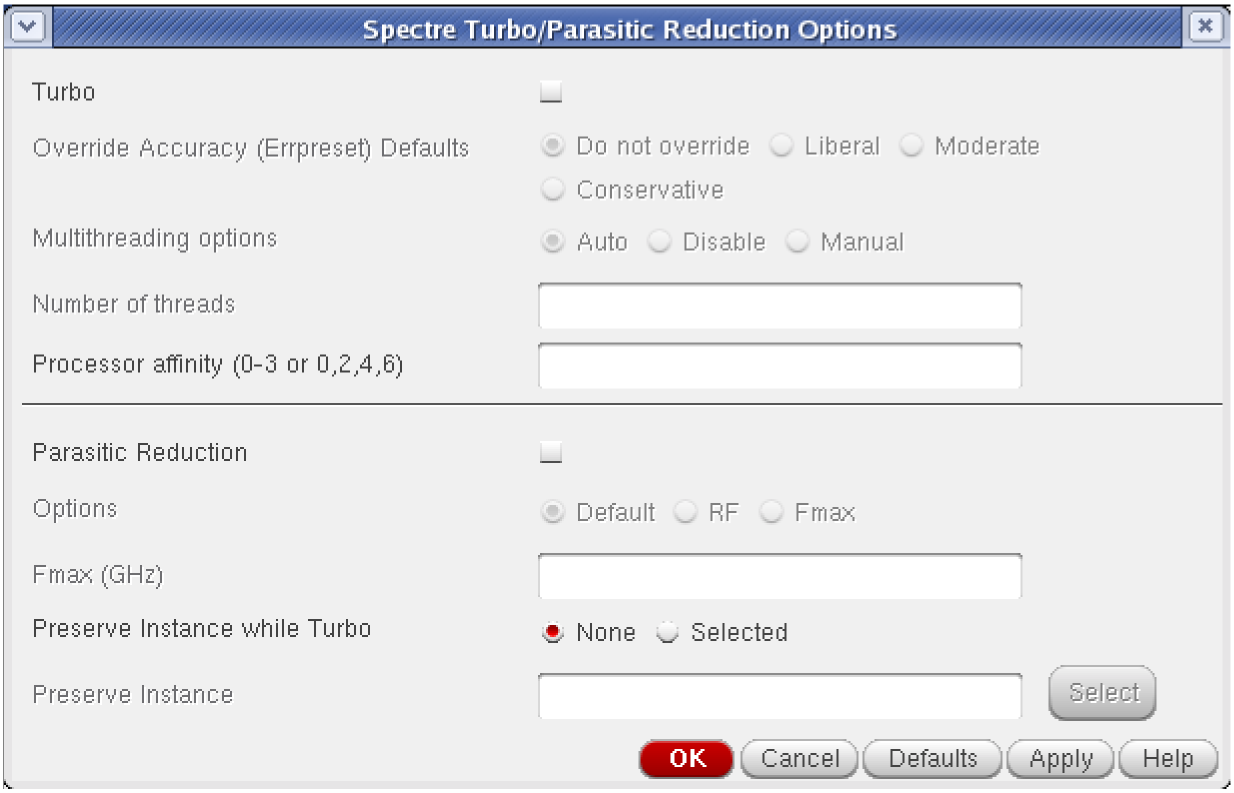

**3.** In the Simulator/Directory/Host form, click *OK*.

## Choosing Turbo and Parasitic Options

Spectre RF Turbo mode provides significant performance gains over the baseline Spectre RFproduct without degrading accuracy. Other than turning on and setting up for the mode, theuse model is the same as with baseline Spectre RF. You can use both Turbo and the parasiticreduction modes simultaneously or use either mode separately.To turn on and set up the Turbo and parasitic options,

**1.** From the Virtuoso Analog Design Environment window, choose *Setup – High Performance Simulation*.

The Spectre *High Performance Simulation Options* form appears.




Turbo mode provides additional simulation speed for CMOS circuits by using more efficient device models and parallelizing device currrent evaluation. APS adds full parallelization of the matrix solution for harmonic balance to the turbo features. It uses a faster algorithm for the small-singnal analyses that follow a PSS, QPSS, or HB analysis. In addition it uses the APS algorithm for the DC analysis at the start of the tstab interval and during the tstab interval For shooting, APS only accelerates the DC and tstab portions of the simulation. The shooting interval and small-signal analyses use turbo at the current time. This limits the speedup of shooting.

2. Select the option that best suits your need.

3. In the *Override Accuracy (Errpreset) Defaults* field, specify whether the errpreset value from the netlist is to be overridden, and, if so, what the new value should be.

4. Choose a suitable value from the *Multithreading options* radio buttons.

   These values apply only to multi-cpu or multi-core machines. Auto detects how many cores are present on the machine and spawns that many threads.

5. If you choose the Manual value for *Multithreading options*, specify the number of threads to be used in the *Number of threads* field.

6. The *Processor affinity* field allows you to specify which processor cores you must use on your machine. If you have 8 cores available and want to run 2 4-threaded jobs at the same time, specify 0-3 for one job and 4-7 for the second job. This simplifies the task for the operating system and speeds the overall simulation time.
7. Select the *Parasitic Reduction* check box if you want to enable parasitic reduction for post-layout circuits that are dominated by parasitics. Note that when parasitic reduction is selected, some nodes may disappear in the simulation result. If this happens, set the preserve Instance in High Performance field to selected and define the instances you want to preserve in the preserve instances field.
8. In the *Options* field, select the option for parasitic reduction.

   Default sets the minumum reduced pole frequency to 1GHz.

   RF Cadence recommends this value because it preserves the level of accuracy needed by RF analyses by setting the minimum pole frequency to 30GHz.

   *Fmax* turns on the *Fmax (GHz)* field, allowing you to specify minimum pole frequency in the reduced netlist. Lower pole frequencies allow for more reduction and faster simulation times but with more error if your operating frequency is above the frequency you set for Fmax.
9. Click *OK*.

The High Performance Simulation Options form closes.

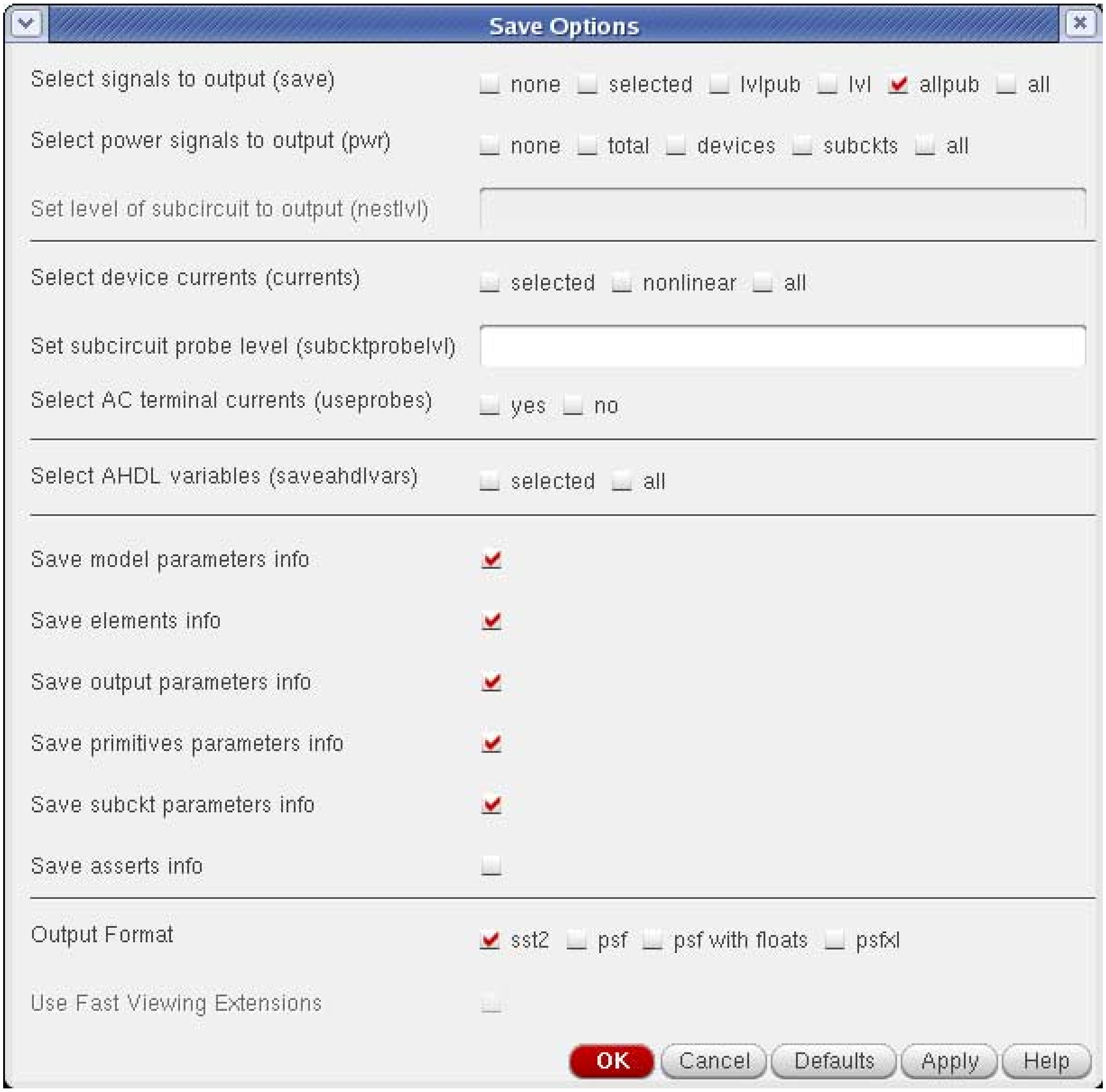
## Specifying Outputs to Save

To specify the outputs that you want to save,

1. In the ADE window, choose *Outputs – Save All.*

   The Save Options form appears.

**2.** In the *Select signals to output section*, be sure *allpub* is highlighted.




## Setting Up Model Libraries

To set up the model libraries,

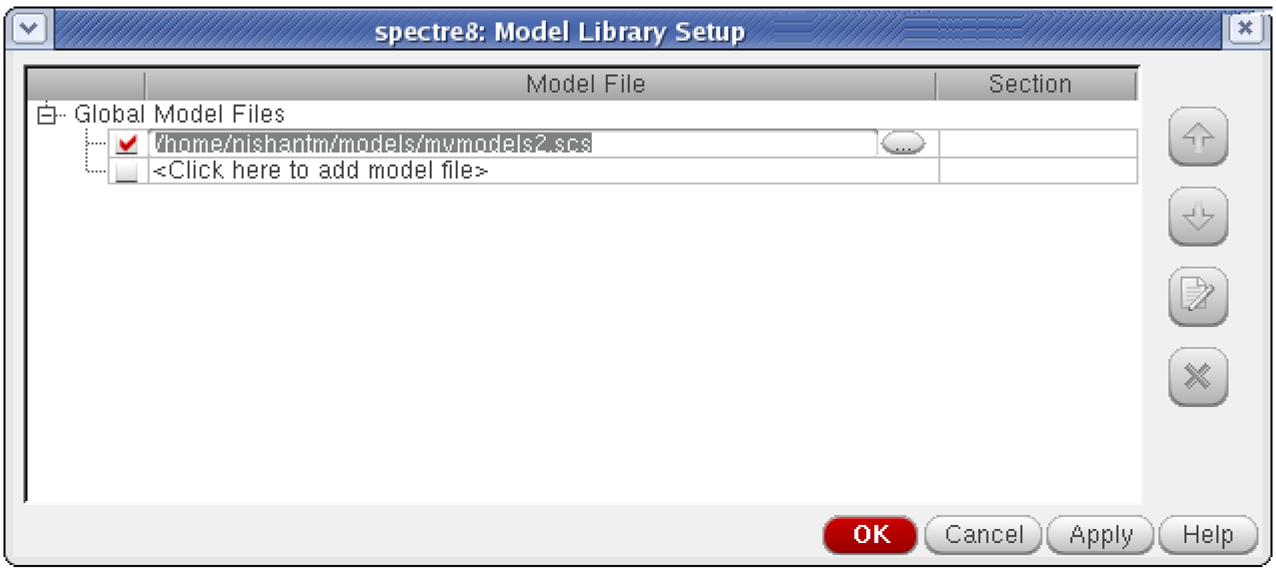
**1.** In the ADE window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs`.

   When you type the path to the model library, you must expand `<CDSHOME>` and type in the actual path to your software installation hierarchy.

   The completed form appears like the one below.

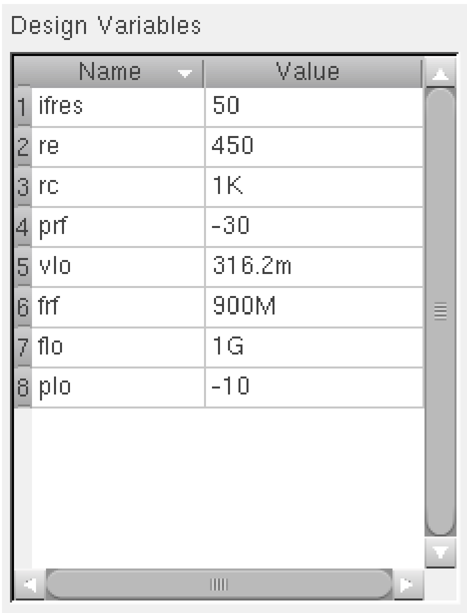



3. In the Model Library Setup form, click *Add*.
4. Click *OK*.

## Editing Design Variable Values

To edit the design variable values,

**1.** Verify the current variables and their values in the Design Variables area in the ADE window.


Design Variables

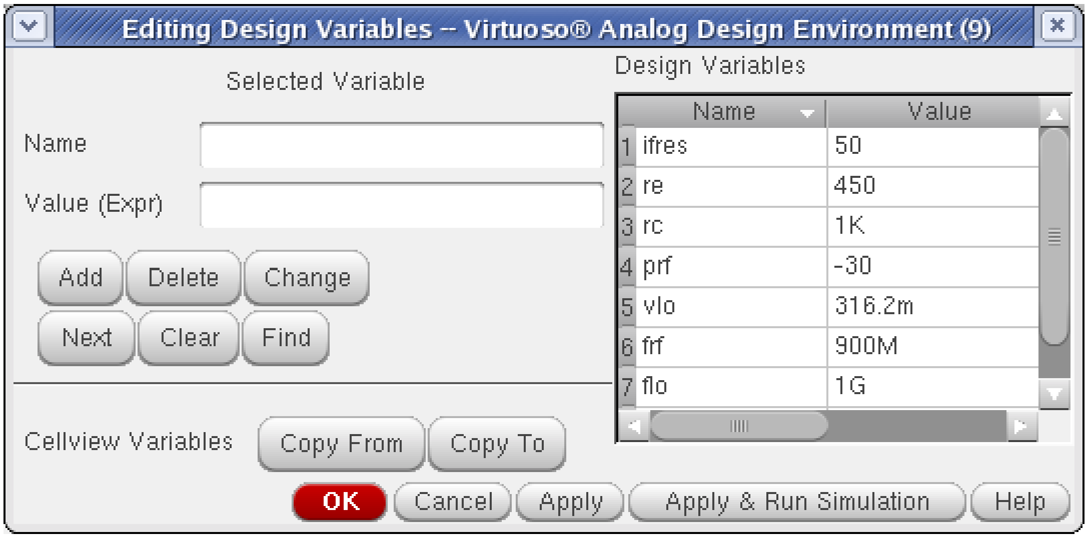
| | Name | Value |
|---|---|---|
| 1 | ifres | 50 |
| 2 | re | 450 |
| 3 | rc | 1K |
| 4 | prf | -30 |
| 5 | vlo | 316.2m |
| 6 | frf | 900M |
| 7 | flo | 1G |
| 8 | plo | -10 |

**2.** In the ADE window, use the following procedure to set the design variables to the values required for the simulation.

(See the description of each simulation for the required variable names and values.) This example changes the value of the variable `frf`.

**a.** In the ADE window, choose *Variables – Edit*.

The Editing Design Variables form appears.




**b.** In the *Table of Design Variables*, select a variable to edit.

**c.** The variable name and value display in the *Name* and *Value (Expr)* fields.

**d.** In the *Value (Expr)* field, type `900M` (or `920M`)for the value of `frf` and click *Change*.

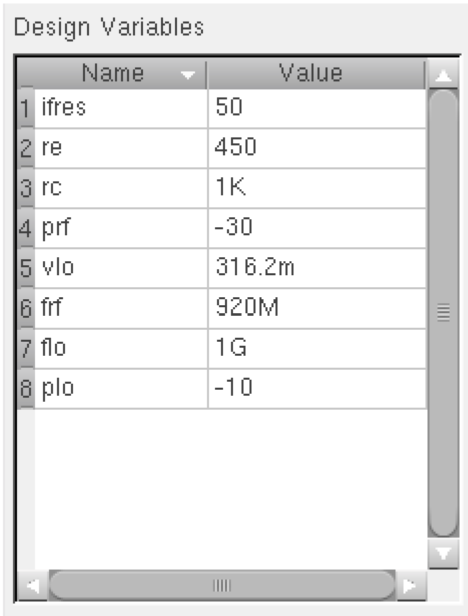
**e.** Edit the variable's value in the *Value (Expr)* field. Then click *Change*.

**f.** The variable value changes in the *Table of Design Variables.*

**g.** Repeat to edit more variables.

**h.** In the Editing Design Variables form, click *OK*.

**i.** View the new variable value in the *Design Variables* section of the ADE window.



# 5

# Simulating Mixers

The Spectre RF simulator can simulate circuits, such as mixers, that show frequency conversion effects. This chapter uses a commercially available integrated circuit mixer, the ne600p, configured as a down converter to illustrate how the Spectre RF simulator can determine the characteristics of your design. The ne600p circuit examples illustrate both the capabilities and the requirements of the Spectre RF simulator. The examples show you how to perform a simulation with an RF to IF ratio of almost 12 to 1. The examples also show you how to be sure that all time-varying independent signal sources have a common integral multiple (40 MHz in this case).

In the mixer examples, you plot the following nonlinear characteristics of the `ne600p` mixer.

| Measurements | Analyses |
|---|---|
| Total Harmonic Distortion Measurement with PSS | PSS |
| Compression Distortion Summary with PSS and PAC | PSS and PAC |
| Rapid IP3 Measurement with PSS and PAC | PSS and PAC |
| Noise Figure Measurement with PSS and Pnoise | PSS and Pnoise |
| Noise Figure Measurement and Periodic S-Parameter Plots with PSS and PSP | PSS and PSP |
| Conversion Gain and Power Supply Rejection with PSS and PXF | PSS and PXF |
| Calculating the 1 dB Compression Point with Swept PSS | Swept PSS |
| Third-Order Intercept Measurement with Swept PSS and PAC | Swept PSS and PAC |
| Intermodulation Distortion Measurement with QPSS | QPSS |
| Noise Figure with QPSS and QPnoise | QPSS and QPnoise |
| IP3 and Compression Distortion Summary Measurements for a Mixer | PSS and Specialized PAC |

| Measurements | Analyses |
|---|---|
| Rapid IP2 and IM2 Distortion Measurements for a Gilbert Direct Conversion Mixer | PSS and Specialized PAC |
| Rapid IP3 and Compression Distortion Summary Measurements for a Power Amplifier | Specialized AC |

To use this chapter, you must be familiar with the Spectre RF simulator analyses as well as know about mixer design. For more information about the Spectre RF simulator analyses, see *Spectre RF Simulation Option Theory*.

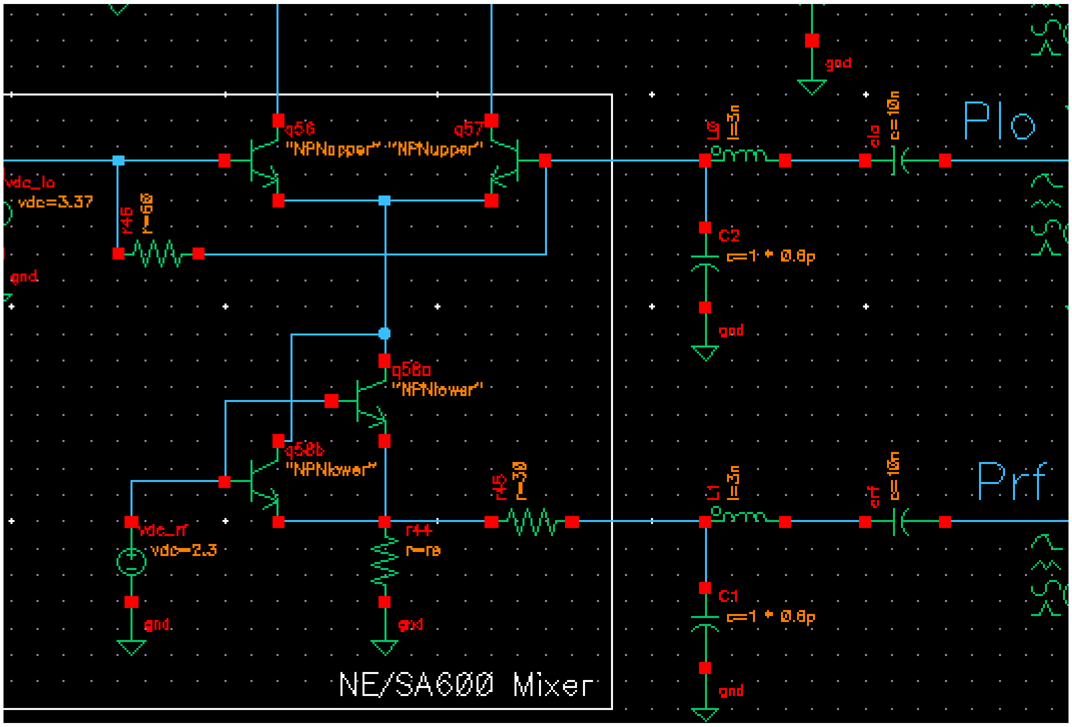
To make these measurements, you run one of the following combination of analyses.

- A Periodic Steady-State analysis (PSS), and a periodic small-signal analysis: Periodic AC (PAC) analysis, Periodic Transfer Function (PXF) analysis, Periodic Noise (Pnoise) analysis, or Periodic S-Parameter (PSP) analysis.
- A Quasi-Periodic Steady-State (QPSS) Analysis and the small-signal Quasi-Periodic Noise (QPnoise) analysis.

# The ne600p Mixer Circuit

The ne600p integrated circuit is commercially available and is designed for low power communication systems from 800-1200 MHz. The receiver is a single balanced mixer containing four bipolar transistors. The schematic for the ne600p mixer circuit is shown in Figure 5-1.

**Figure 5-1 Schematic for the ne600p Mixer Circuit**




The high-level local oscillator (LO) signal (~0 dBm at 50 Ohms) enters through the emitter follower q57. The transistor q56 is in common base configuration. The q56 emitter follows the q57 emitter. When the LO signal goes high, both emitters also go high and the B-E voltage on q56 decreases. As a result, the two BJTs form a balanced pair.

The collector currents of q56 and q57 are 180 degrees out of phase. The q58 pair feed the low-level (-30 to -20 dBm) RF signal to the balanced pair. The IF signal is drawn from the open collector output of q56 through a bandpass passive network.

The following tables list some measured values for different aspects of the ne600p mixer.

| Measurement | Measured |
|---|---|
| LO frequency (Hz) | 1 GHz |
| RF frequency (Hz) | 920 MHz |
| IF frequency (Hz) | 80 MHz |
| LO power | 0 dBm |
| RF power | -30 dBm |

| Measurement | Measured |
|---|---|
| Conversion gain | -2.6 dB |
| Noise figure | 16 dB |
| Input 1dB compression point | -4 dBm |
| Input IP3 (from swept power) | 6 dBm |
| Input IP3 (from PAC analysis) | 6 dBm |

| Design Variable | Default Value |
|---|---|
| prf (RF power) | -30 dBm |
| vlo (LO magnitude) | 316.2 m |
| frf (RF frequency) | 920 MHz |
| flo (LO frequency) | 1 GHz |

Before you use the ne600p circuit from the sample library, be sure that the *frf* design variable is set to the appropriate value, either 920 MHz or 900 MHz, depending on the simulation. If you adjust the RF frequency to the center of the IF tuned circuit, approximately 100 MHz, the conversion gain becomes -2.57dB and the other values change slightly.

# Setting Up to Simulate the ne600p Mixer

Before you start, perform the setup procedures described in Chapter 4, "Setting Up for the Examples."

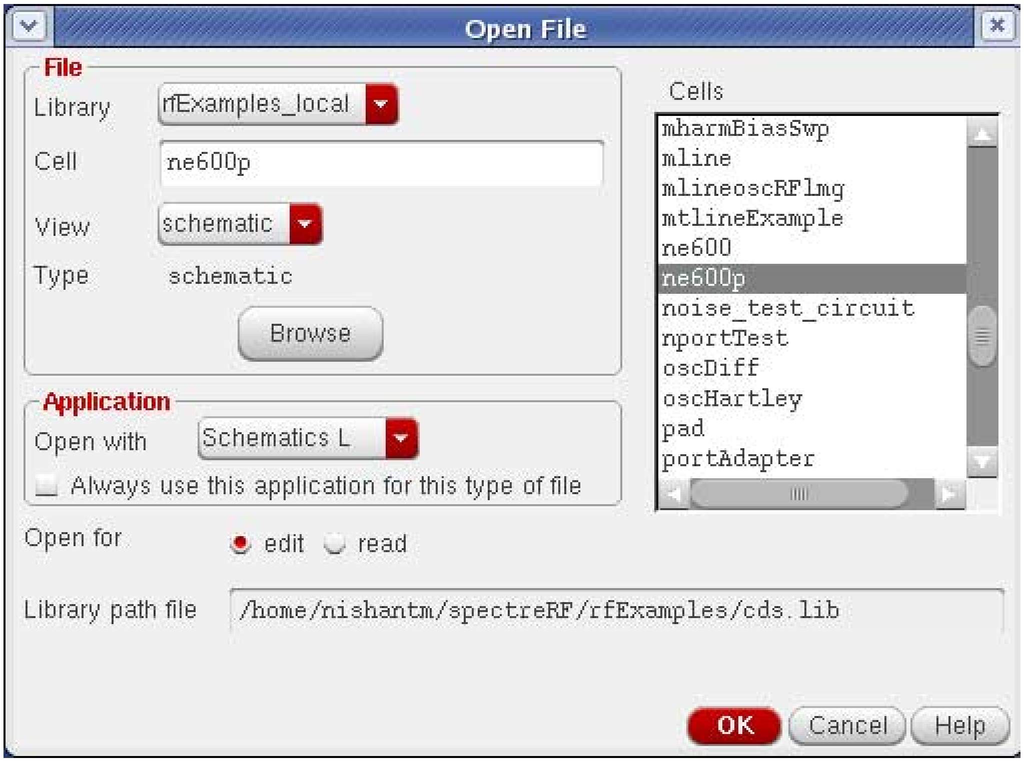
## Opening the ne600p Mixer Circuit in the Schematic Window

In the CIW, choose *File – Open*.

The Open File form appears.

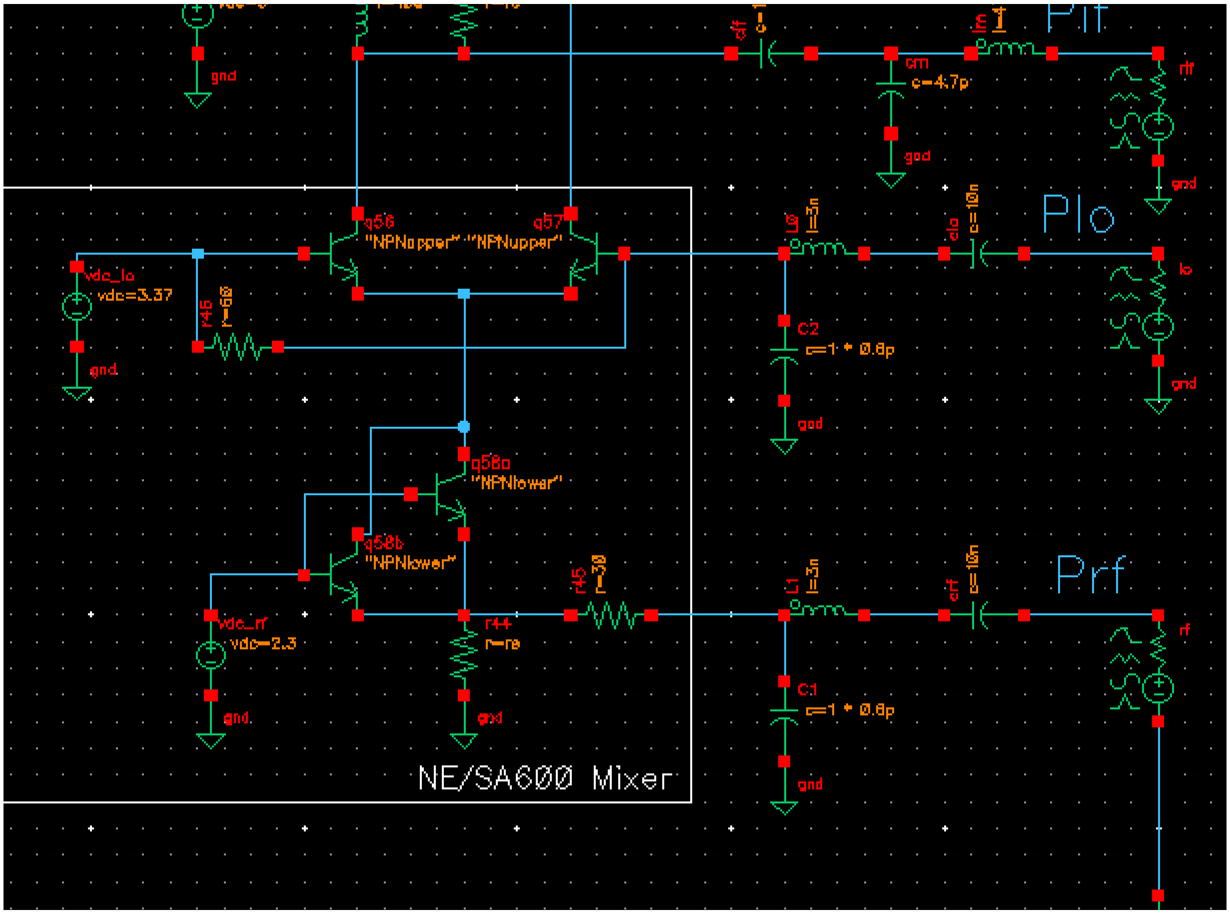
1. In the Open File form, choose *my_rfExamples* in the *Library Name* menu. Choose the editable copy of the rfExamples library you created as described in Chapter 4, "Setting Up for the Examples."
2. Choose *ne600p* in the *Cell Name* list box.

The completed Open File form appears like the one below.



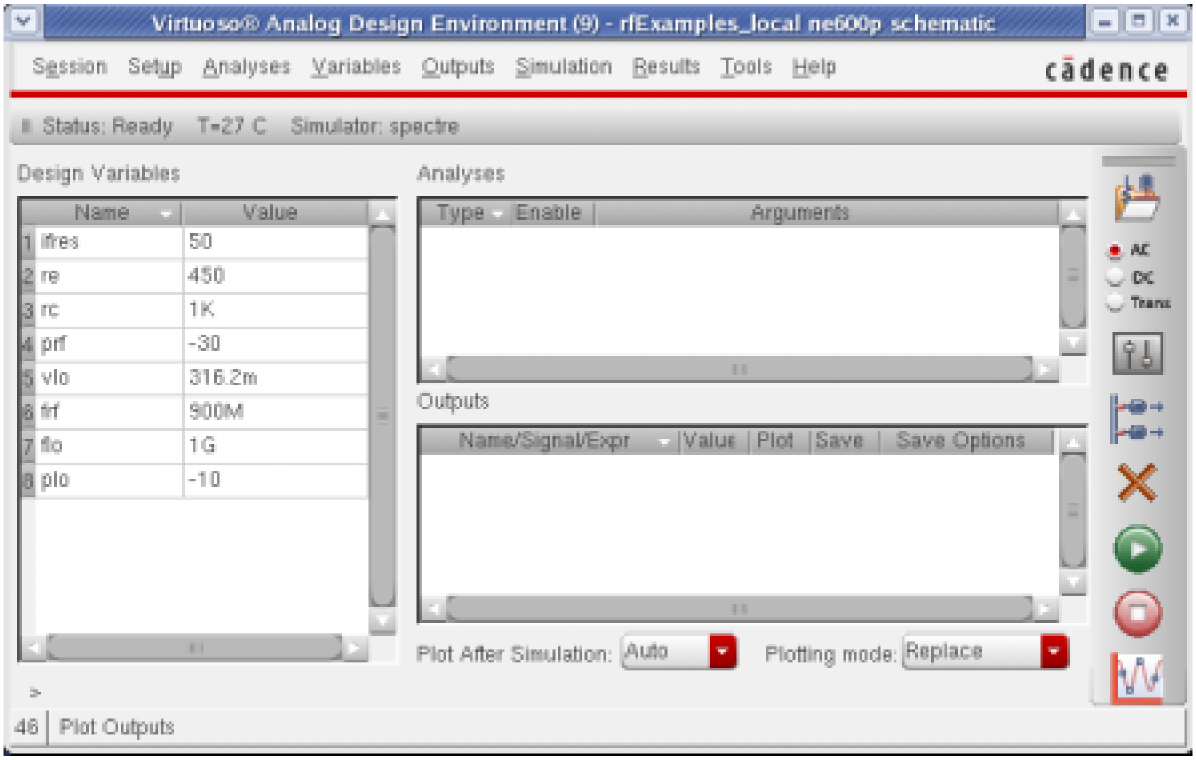

**3.** Click *OK*.

The Schematic window for the *ne600p* mixer appears.




**4.** In the Schematic window, choose *Tools– Analog Environment*.

The Virtuoso Analog Design Environment window opens.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the Virtuoso Analog Design Environment window without opening the design. You can open the design later by choosing *Setup – Design* in the Virtuoso Analog Design Environment window and choosing the *ne600p* in the Choosing Design form.

## Choosing Simulator Options

1. Choose *Setup – Simulator/Directory/Host* in the Virtuoso Analog Design Environment window.

   The Choosing Simulator/Directory/Host form appears.

2. Specify the following:

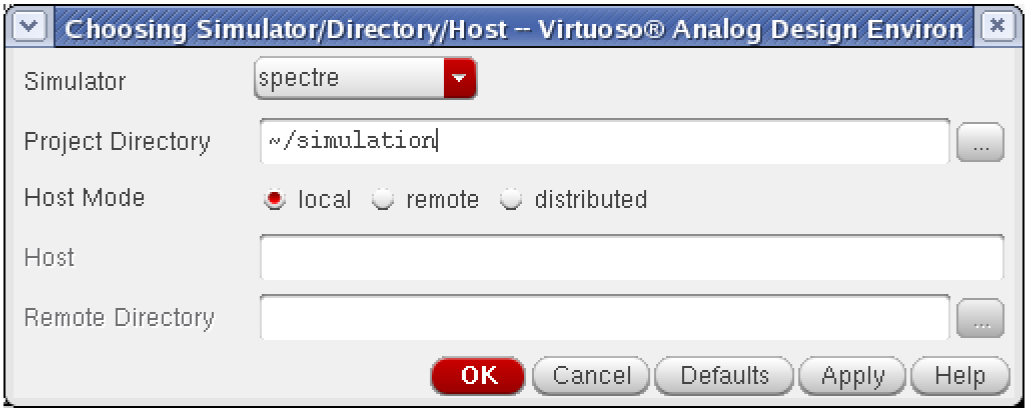
   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

   c. Highlight the *Host Mode* that corresponds to your situation.

   d. If you choose *remote* or *distributed*, fill in the additional fields as necessary.

The completed form appears similar to the one below.



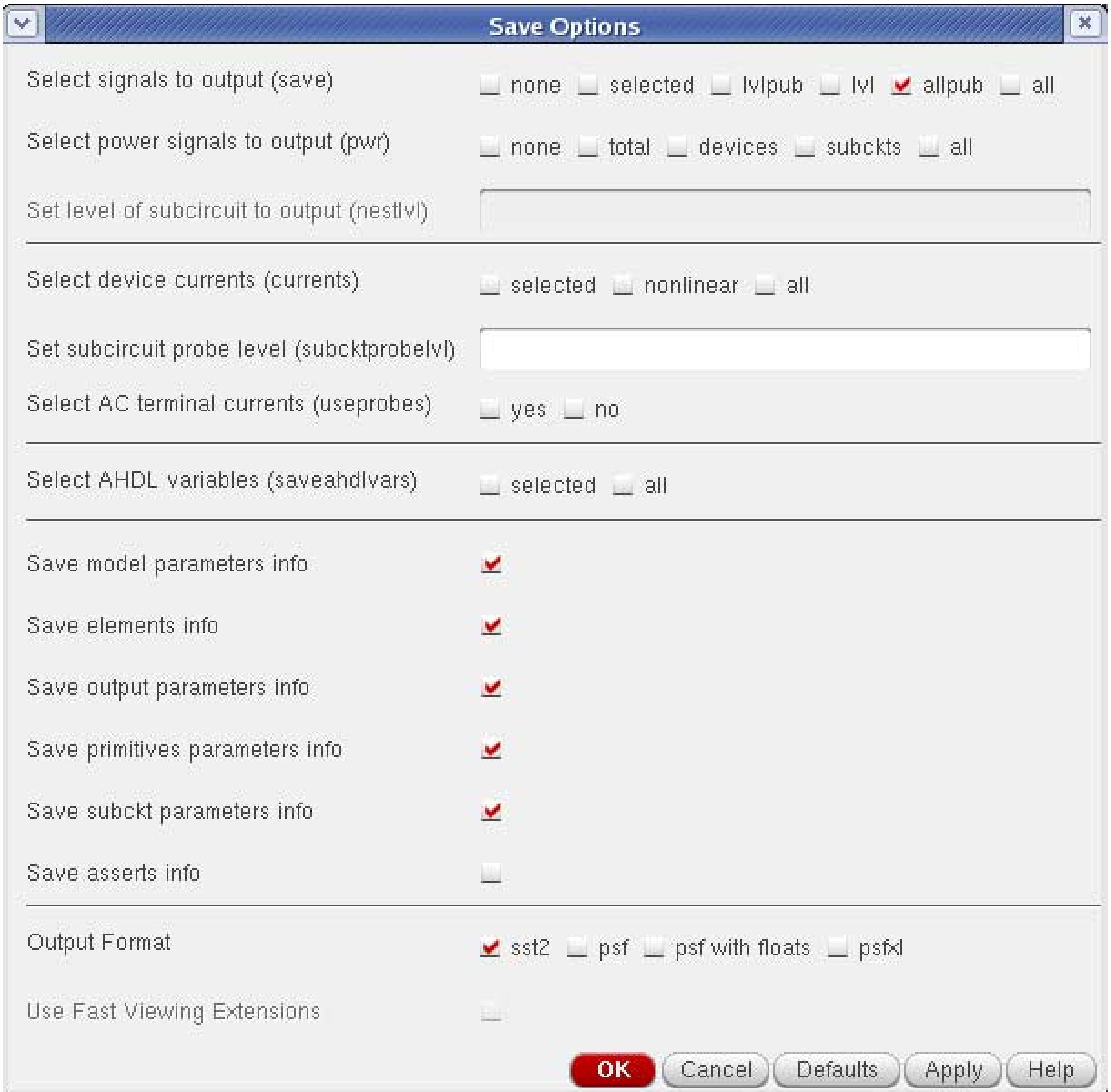

3. Click *OK*.

   The Choosing Simulator/Directory/Host form closes.

4. In the Virtuoso Analog Design Environment window, choose *Outputs – Save All*.

   The Save Options form appears.

**5.** In the *Select signals to output* section, be sure *allpub* is highlighted.




**6.** Click *OK*.

The Save Options form closes.

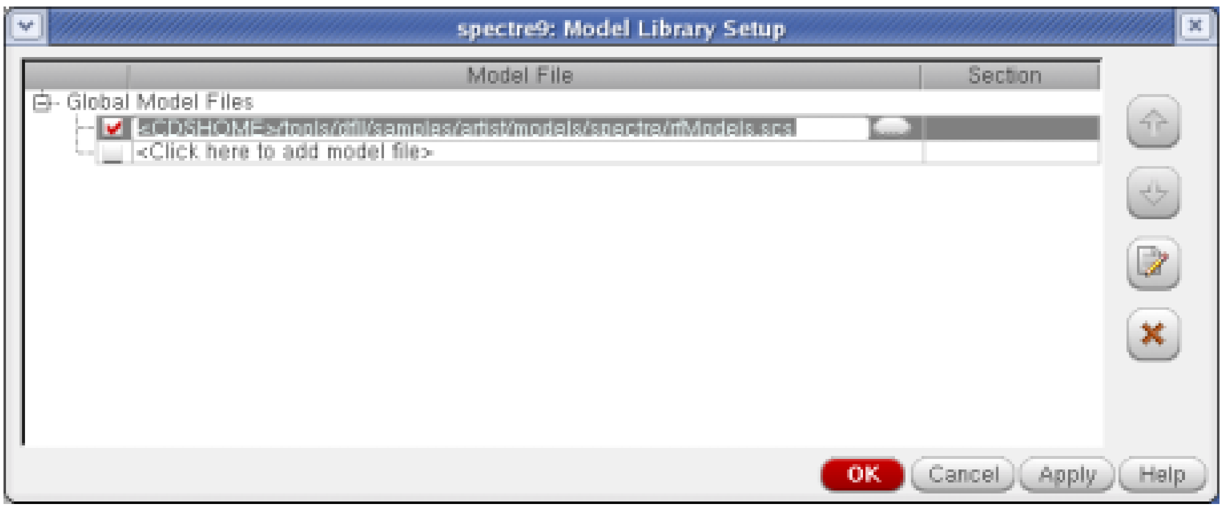
## Setting Up Model Libraries

**1.** In the Virtuoso Analog Design Environment window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs`.
3. Click *Add*.

   The Model Library Setup form looks like the following.




4. Click *OK*.

   The Model Library Setup form closes.

5. In the Virtuoso Analog Design Environment window, disable any analyses you ran previously.

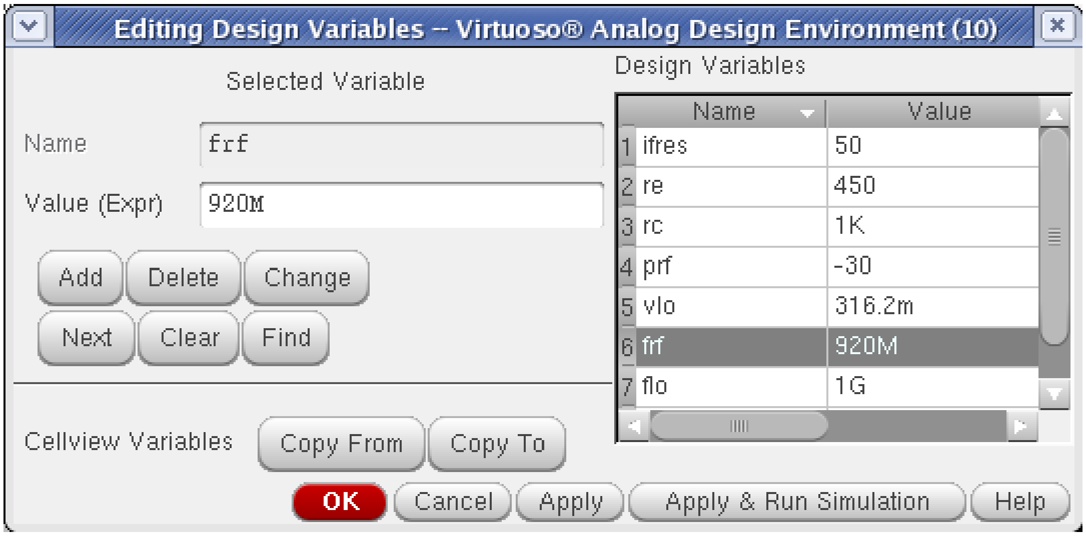
## Setting Design Variables

Use the following procedure to set the design variables to the values required for each simulation. (See the description of each simulation for the required value.)

This example sets values for `frf` and `flo`.

1. In the Virtuoso Analog Design Environment window, highlight `frf` in the *Design Variable* list box.
2. Choose *Variables – Edit*.

3. The Editing Design Variables form appears.




4. Highlight `frf` in the *Table of Design Variables* list box.
5. In the *Value (Expr)* field, type `920M` for the value of `frf` and click *Change*.
6. Highlight `flo` in the *Table of Design Variables* list box.
7. In the *Value (Expr)* field, type `1G` for the value of `flo` and then click *Change*.
8. In the Editing Design Variables form, click *OK*.

   The form closes.

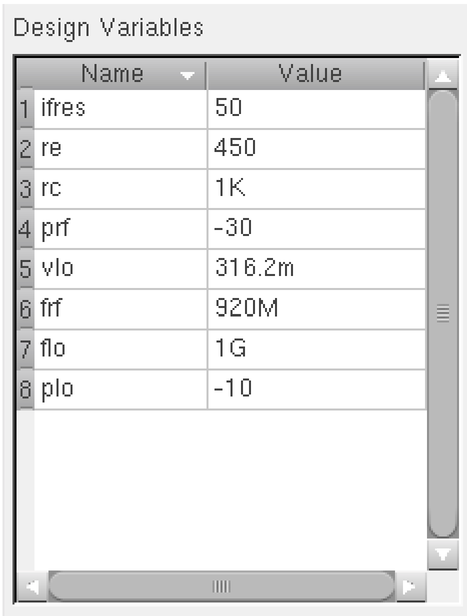
# Total Harmonic Distortion Measurement with PSS

In this example, a single PSS analysis determines the Total Harmonic Distortion (THD) for the mixer.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

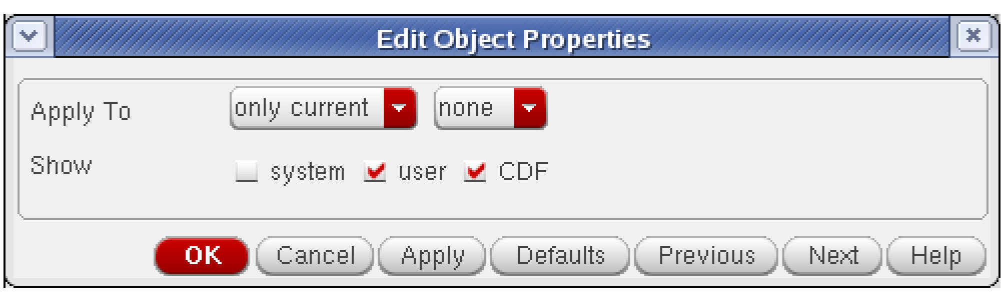



## Editing the Schematic

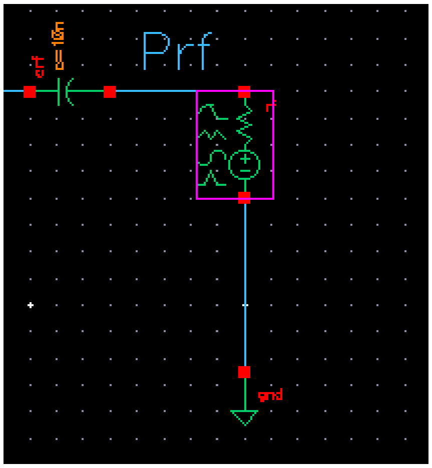
A critical part of this analysis is the correct use of the programmable voltage sources in the `ne600p` circuit. The RF voltage source is based on the *port* sample component. You must change the behavior of this component for each analysis.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the *rf* voltage source and modify the schematic for this simulation.



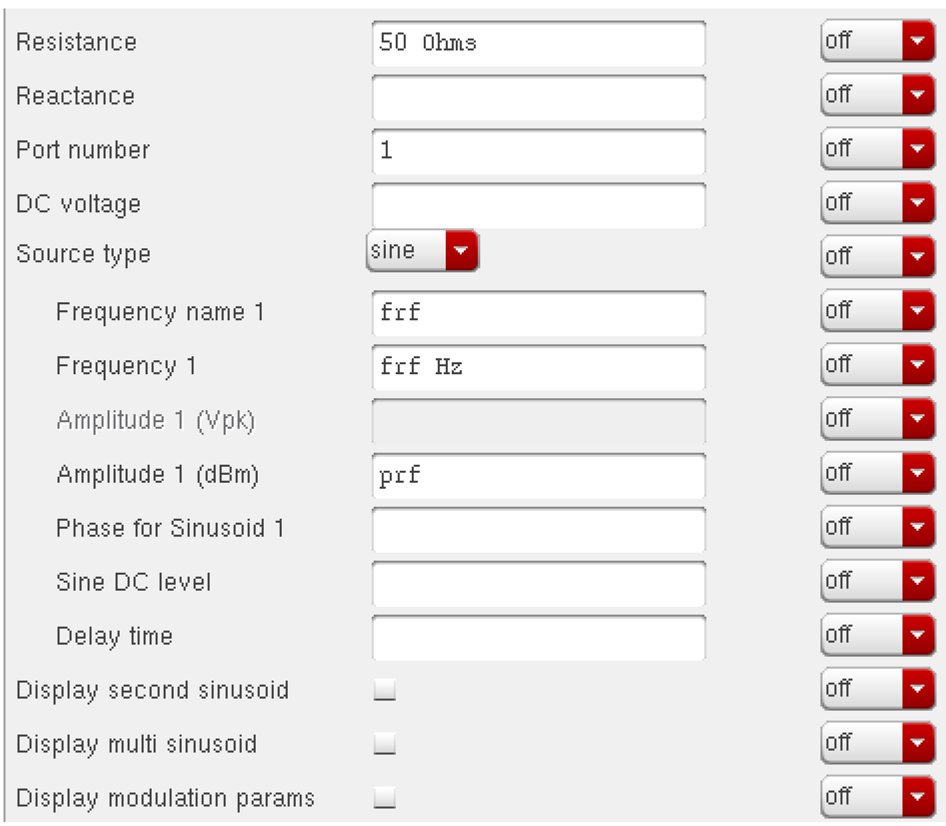
**2.** In the Schematic window, click the *rf* voltage source.




The Edit Object Properties form changes to display information for the voltage source.

**3.** In the Edit Object Properties form, be sure that *sine* is specified as the *Source type*.

A portion of the completed Edit Object Properties form for the rf port appears like the one below:



| | | |
|---|---|---|
| Resistance | 50 Ohms | off |
| Reactance | | off |
| Port number | 1 | off |
| DC voltage | | off |
| Source type | sine | off |
| Frequency name 1 | frf | off |
| Frequency 1 | frf Hz | off |
| Amplitude 1 (Vpk) | | off |
| Amplitude 1 (dBm) | prf | off |
| Phase for Sinusoid 1 | | off |
| Sine DC level | | off |
| Delay time | | off |
| Display second sinusoid | | off |
| Display multi sinusoid | | off |
| Display modulation params | | off |

**4.** In the Edit Object Properties form, click *OK*.

**5.** In the Schematic window, choose *Design – Check and Save*.

## Setting Up the PSS Analysis

**1.** Choose *Analyses – Choose* in the Virtuoso Analog Design Environment window.

The Choosing Analyses form appears.

**2.** In the Choosing Analyses form, click *pss*.

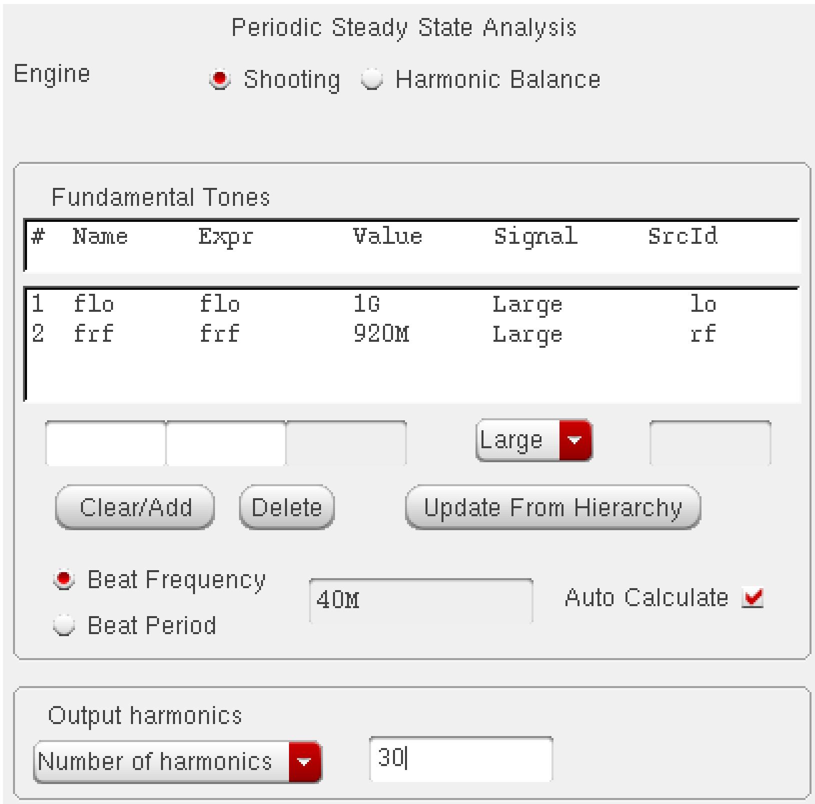
**3.** In the *Fundamental Tones* list box, *Beat Frequency* is highlighted by default. Be sure *Auto Calculate* is also highlighted.

The *Beat Frequency* is calculated. The `40M` value is the minimum period for which both RF at `920 MHz` and LO at `1 GHz` are periodic (or for which RF and LO are integer multiples of the fundamental frequency).

**4.** In the *Output harmonics* cyclic field, choose *Number of harmonics*. Type `30` in the *number of harmonics* field.

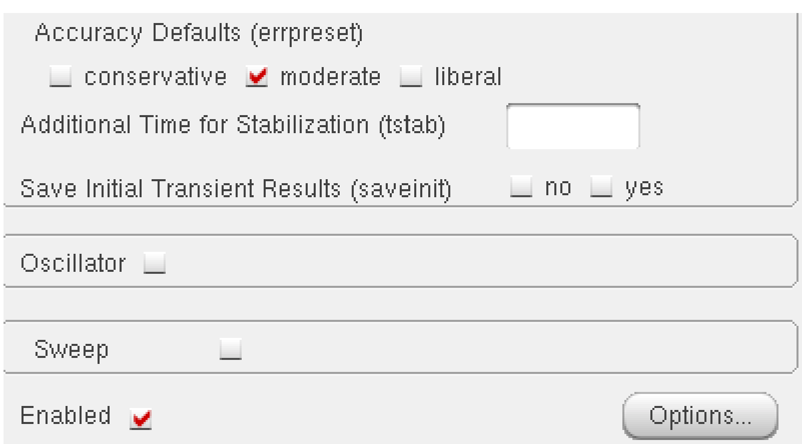
This entry expands the plotted frequency range to include the areas of interest around both `40 MHz` and `1 GHz`. (`30` x `40 MHz` equals `1.2 GHz`, or `0` to `1.2 GHz`, and includes all important frequencies.)

The top of the PSS Choosing Analyses form is as follows.




5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Verify that *Enabled* is highlighted.

   The bottom of the PSS Choosing Analyses form is as follows.




7. Click *Apply*.

**8.** Click *OK.*

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

**2.** Look in the CIW for a message that says the simulation completed successfully.

## Plotting and Calculating Total Harmonic Distortion

**1.** Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

The Direct Plot form appears.

**2.** In the Direct Plot form, select *Append* in the *Plotting Mode* cyclic field.

**3.** Highlight *pss* for *Analysis*.

**4.** Highlight *Voltage* for *Function*.

**5.** In the *Select* cyclic field, select *Differential Nets* to plot voltage against frequency.

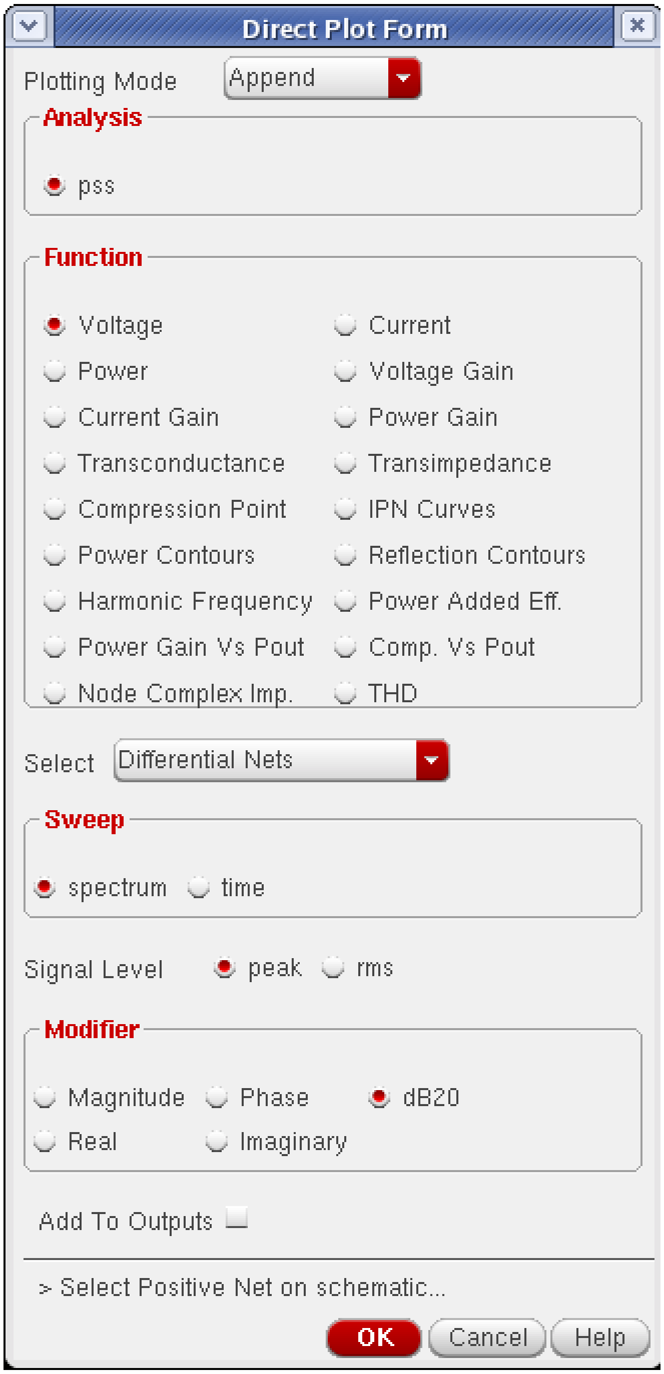
The message at the bottom of the form changes.

**6.** For *Sweep*, highlight *spectrum*.

**7.** For *Signal Level*, highlight *peak*.

**8.** For *Modifier*, highlight *dB20*.

The Direct Plot form appears as follows.




**9.** To plot the voltage against frequency, click the *Pif* and *Prf* wires in the Schematic Window.

**a.** Following the instructions at the bottom of the Direct Plot Form, `Select Positive Net On Schematic...`

Select the positive net, *Pif*, on the schematic.

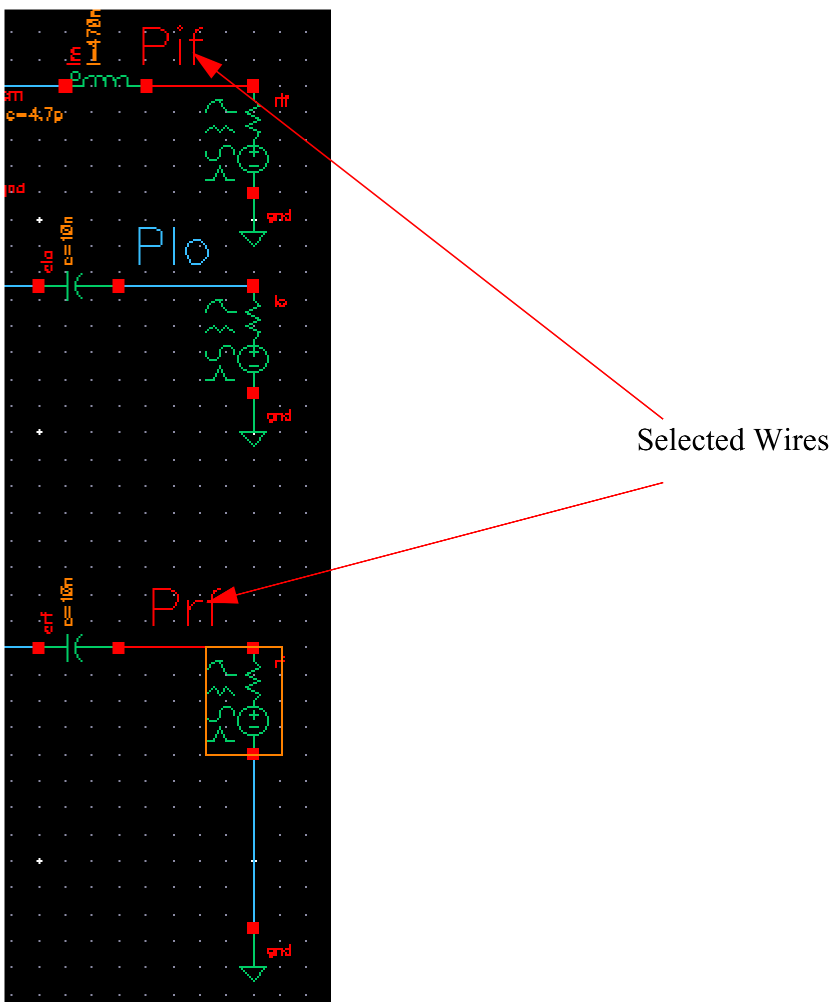
*Pif* is highlighted in the schematic window and the instructions at the bottom of the Direct Plot form change.

**b.** Following the instructions at the bottom of the Direct Plot Form,

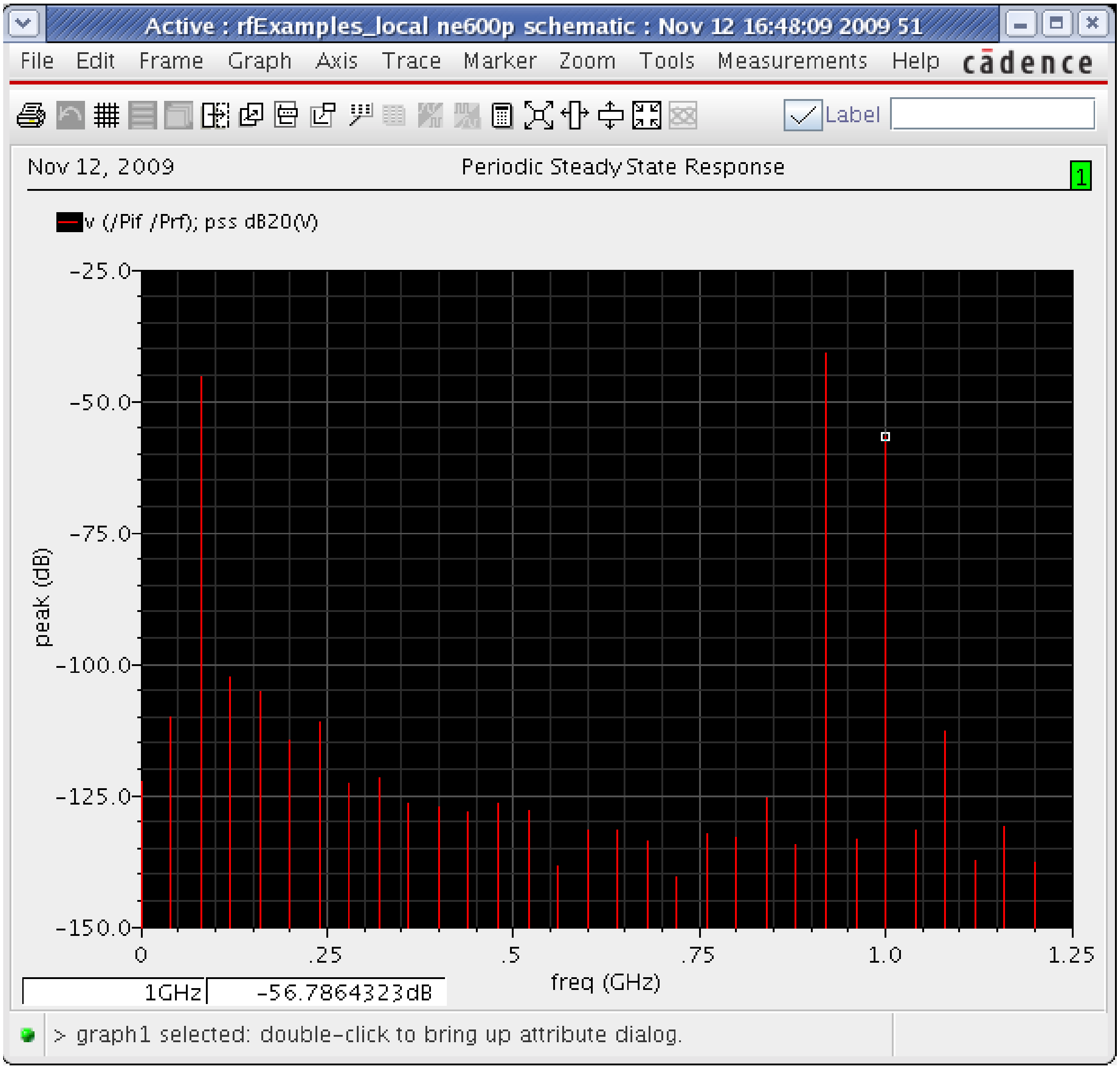
`Select Negative Net on schematic...`

Select the negative net, *Prf*, on the schematic.

*Prf* is highlighted in the schematic window.



The waveform window display appears like the one below.




**10.** Measure the `Vif` values at the second, fourth, and sixth harmonics. That is take measurements at `80 MHz`, `160 MHz`, and `240 MHz`.

To measure `Vif` at each harmonic, place the cursor at the top of the second, forth, and sixth harmonic in turn. For the selected harmonic, the X value in MHz and Y value in dB appears at the bottom left corner of the waveform window.

Measurements for the three selected harmonics are as follows.

| Harmonic Number | Frequency Value | Vif Value |
|---|---|---|
| *h*2 — the 2nd harmonic | 80 MHz | -45.48 dB |
| *h*4 — the 4th harmonic | 160 MHz | -111.1 dB |
| *h*6 — the 6th harmonic | 240 MHz | -116.2 dB |

**11.** Calculate the total harmonic distortion (THD) with the following formula:

$$THD = dB10[(h4)^2 + (h6)^2] - dB20(h2)$$

where

*h*2 is the value of `Vif` at the 2nd harmonic: -45.48 dB

*h*4 is the value of `Vif` at the 4th harmonic: -111.1 dB

*h*6 is the value of `Vif` at the 6th harmonic: -116.2 dB

The value of `THD` is 10.93768 `dB`.

# Compression Distortion Summary with PSS and PAC

You can combine a PSS analysis with a small-signal PAC analysis to measure and print the compression distortion summary for selected components in the ne600p circuit.

PAC analysis calculates the distortion contributed by each selected component in the schematic.

The distortion summary is a quantitative measurement of how much distortion a device adds to the output signal. Distortion is measured as

$$dB = 10 \log\left(\frac{V_{out}}{c_1 V_{RF}}\right)$$

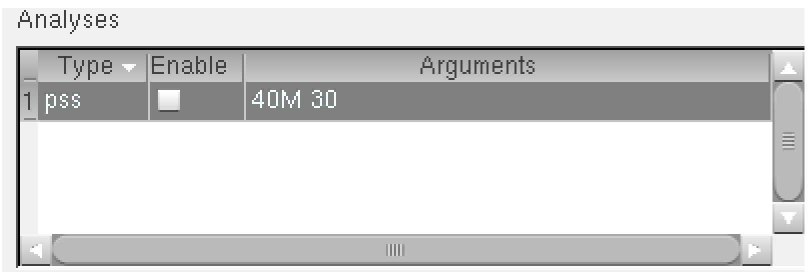
Where

$V_{RF}$ and $V_{out}$ are amplitudes of the input and output signals, respectively.

$C_1$ is the ideal linear conversion gain.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

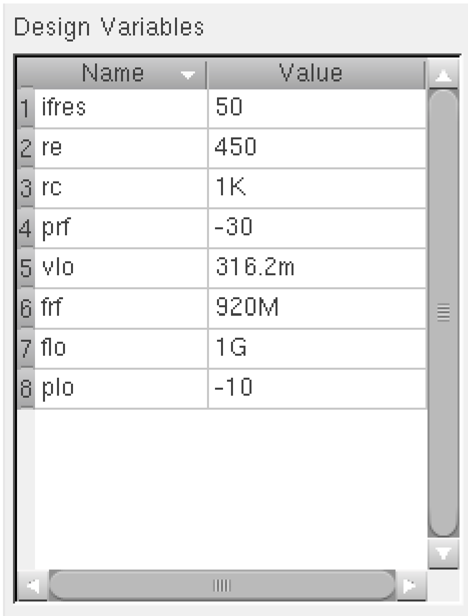



In this figure, the PSS analysis is disabled.

4. If necessary, set the design variables.

| | |
|---|---|
| `flo` | 1G |
| `prf` | -30 |
| `frf` | 920M |

5. Check the Design Variables section to verify the current design variable values.




## Editing the Schematic

Modify properties of the rf voltage source in the ne600p mixer schematic to be sure the PSS analysis response is limited to the LO signal only.

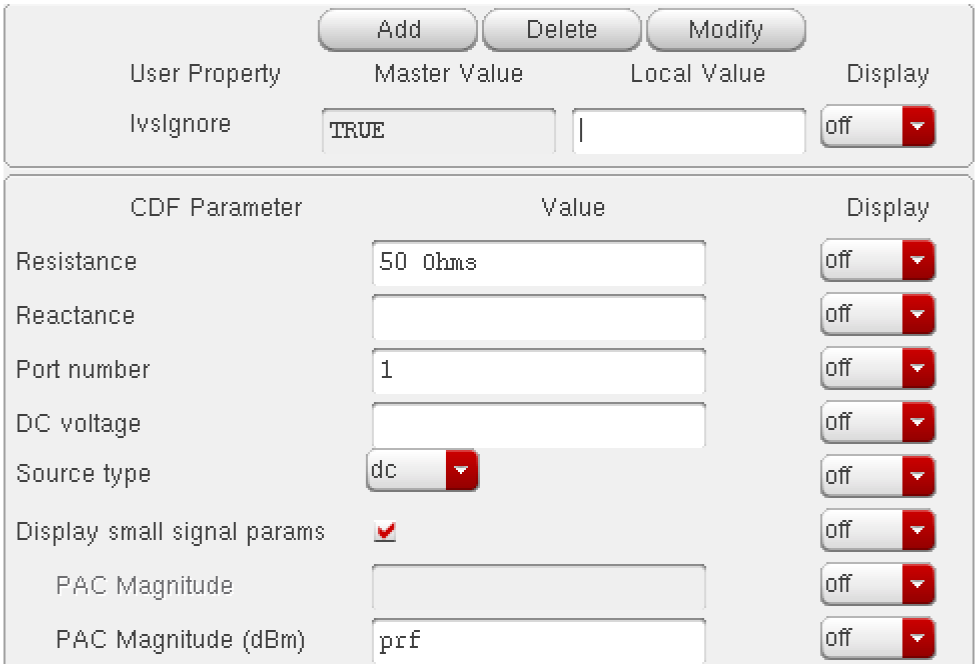
1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, make the following changes and click *OK*.

**4.** Choose *dc* as the *Source type*, if necessary.




**5.** Click *Display small signal parameters* to display small signal parameters associated with the rf source.

**6.** Verify that the *PAC Magnitude (dBm)* small signal parameter is set and that the value is `prf`. The variable prf appears in the Variables section of the Virtuoso Analog Design Environment window.

The *PAC magnitude (dBm)* parameter measures the magnitude of the RF input at the RF source.

For non-RF sources, you should not use the *PAC magnitude (dBm)* parameter or you should set it to 0.

**7.** Click *OK*.

The Edit Object Properties form closes.

**8.** In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

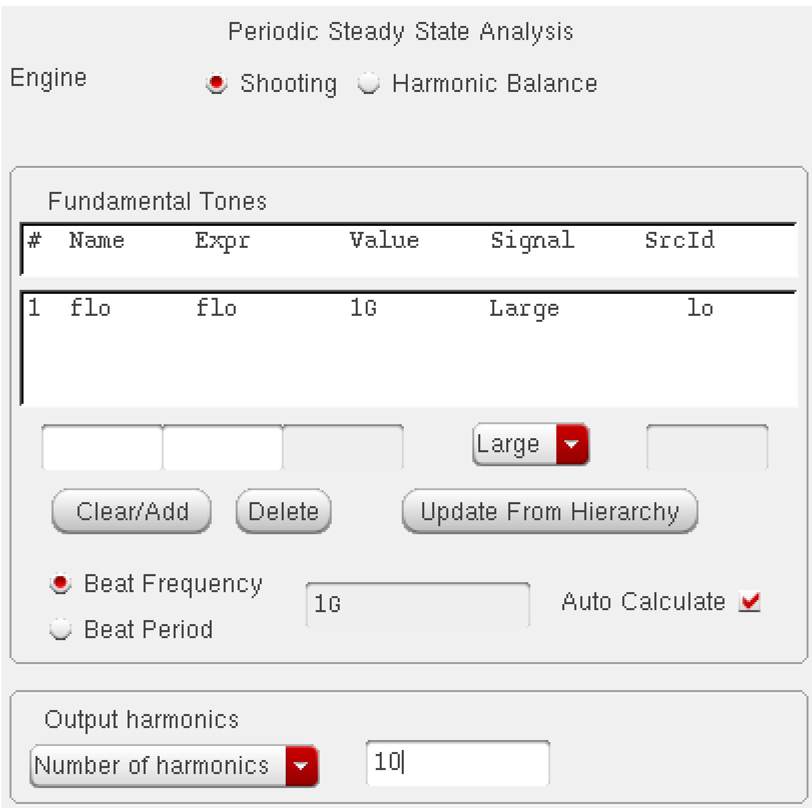
### Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the *Fundamental Tones* section, highlight *Auto Calculate*.

   The *Beat Frequency* is calculated as `1G`.

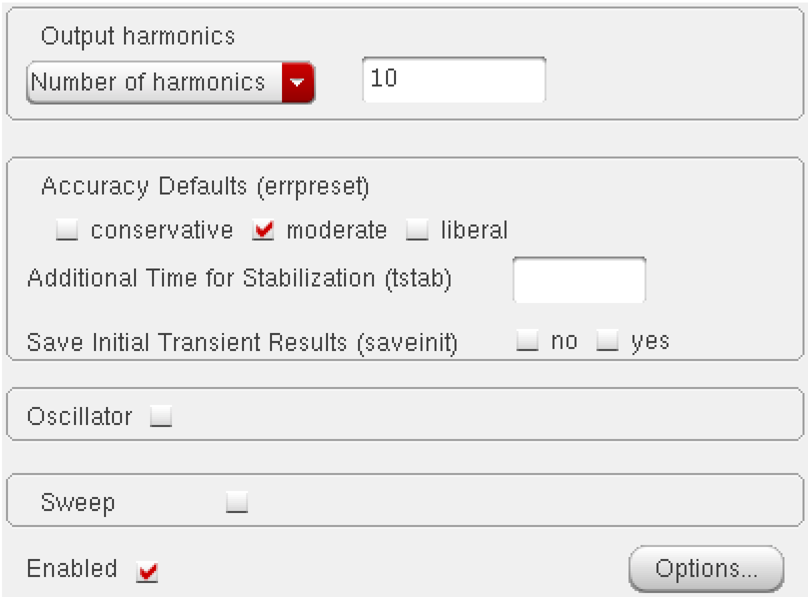
   For *Output Harmonics*, select *Number of harmonics* and enter 10.

   The top of the PSS Choosing Analyses form appears as follows.




3. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
4. Highlight *Enabled*.

5. The bottom of the PSS Choosing Analyses form appears as follows.




6. Click *Apply* in the PSS Choosing Analyses form.

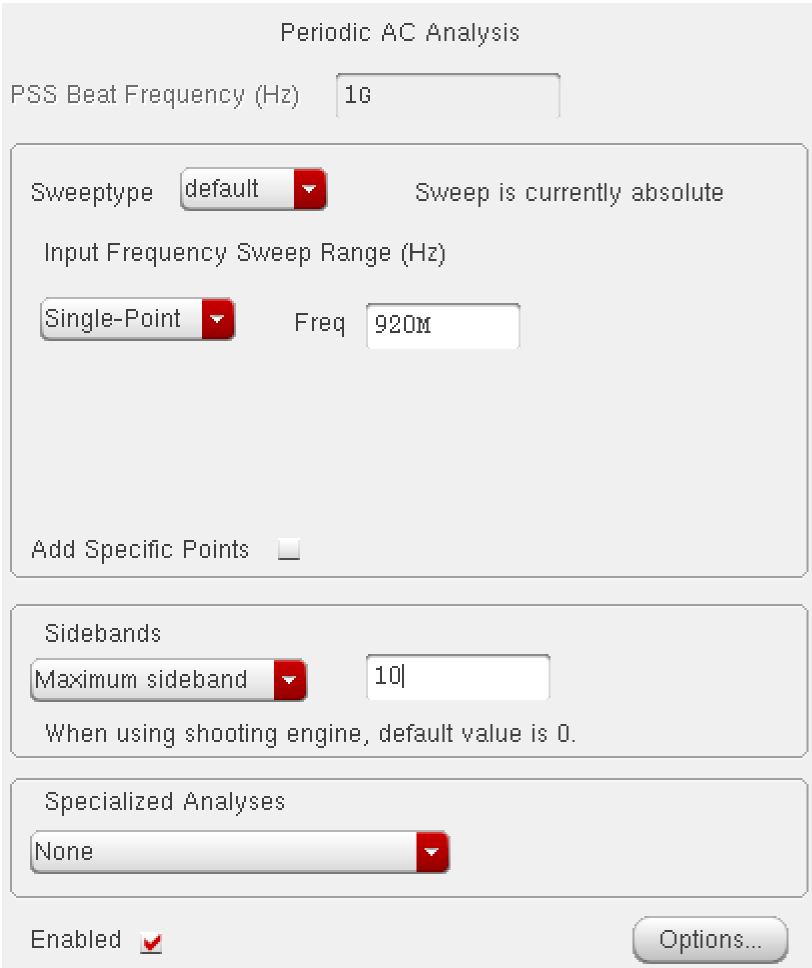
## Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a PAC analysis.

2. In the *Input Frequency Sweep Range (Hz)* cyclic field, choose *Single-Point*.
3. Type `920M` in the *Freq* field.
4. In the Sidebands field, select Maximum Sideband and enter 10.

The top of the PAC Choosing Analyses form looks as follows.




**5.** In the *Specialized Analyses* cyclic field, choose *Compression Distortion Summary*.

The PAC Choosing Analyses form changes to display fields for setting up the Compression Distortion Summary.

A message also displays reminding you to set the pacmag parameter on the RF source in the schematic.

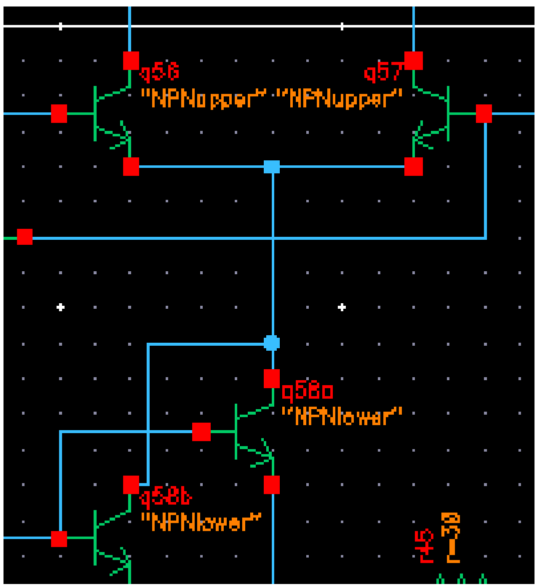
**6.** Click *Close* to dismiss the message and display the *Compression Distortion Summary* fields for the PAC analysis.

**7.** Click *Select*, just above the *Contributor Instances* field.

A message appears in the Virtuoso Schematic Editing window, telling you to

`Select contributor instances. Press Esc when done.`

**8.** Select the transistors in the mixer as the components to measure for the distortion summary.




**a.** Select *q56 "NPNupper"* in the schematic.

**b.** Select *q57 "NPNupper"* in the schematic.

**c.** Select *q58a "NPNlower"* in the schematic.

**d.** Select *q58b "NPNlower"* in the schematic.

**9.** Press *Esc*, when you are done selecting the transistors.

The four selected components display in the *Contributor Instances* field.

**10.** In the *Frequency of Linear Output Signal* field type 80M.

**11.** In the *Maximum Non-linear Harmonics* field, type 4. This value specifies the maximum harmonic used to represent the nonlinear response to RF signals.

**12.** Next to *Output*, click *Voltage*.

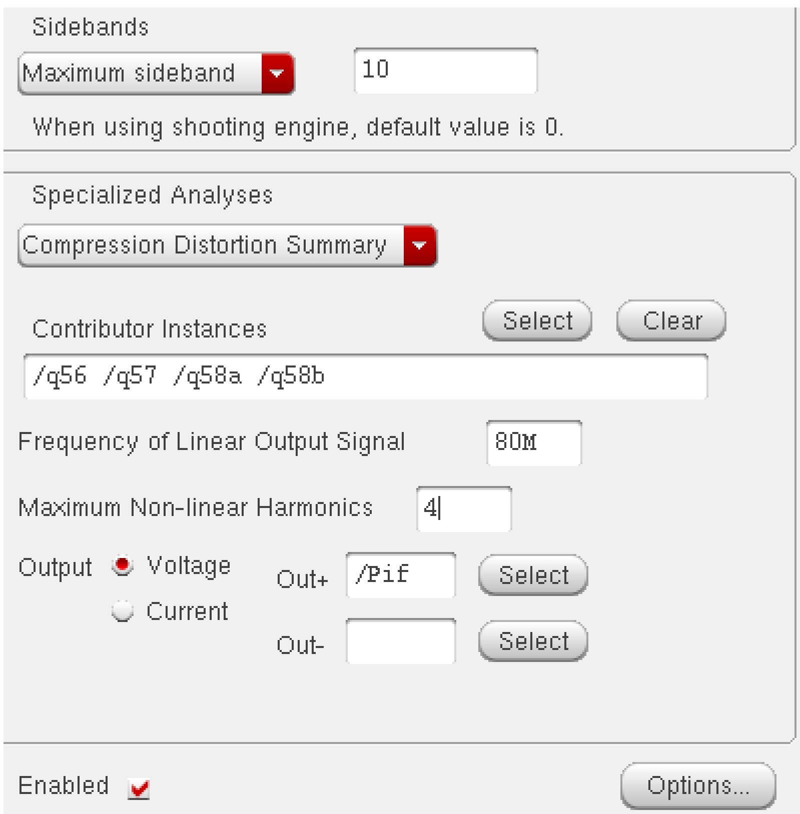
**13.** Next to *Out+* click *Select*. Then click the appropriate net in the Schematic window to choose *Pif*.

`/Pif` appears in the *Out+* field. this is the output node for measurement.

Leave the *Out-* field empty.

The *Out-* field defaults to `/gnd!` You can set *Out-* to a different value by clicking on the *Select* button and then choosing an output node in the schematic.

The bottom of the PAC Choosing Analyses form looks as follows.




**14.** Highlight *Enabled*.

**15.** Click *Apply*.

**16.** Click *OK*.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

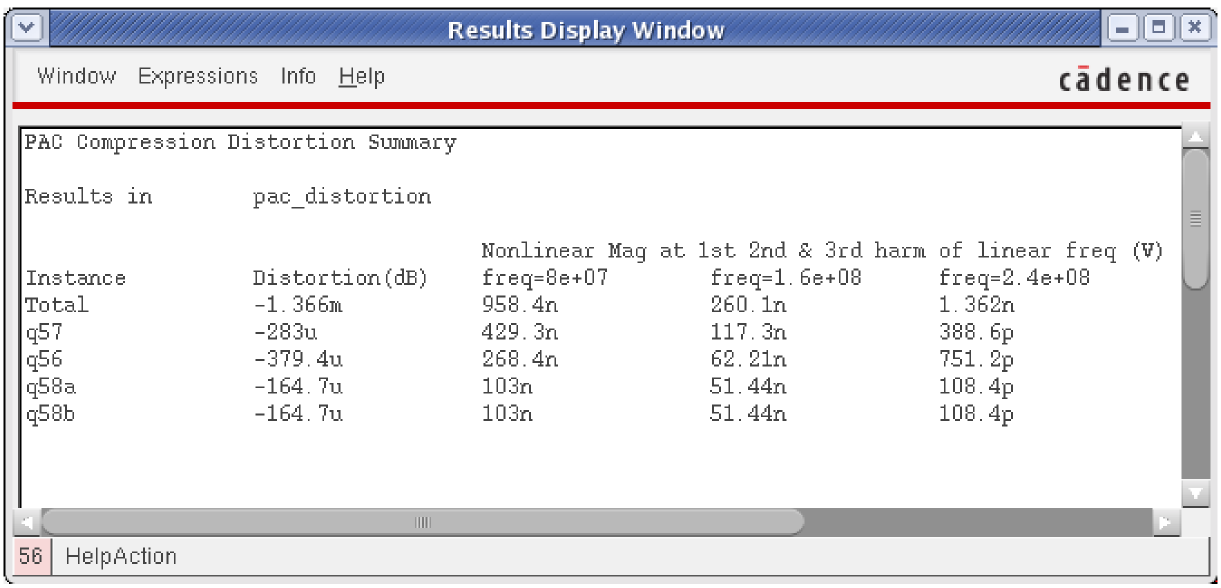
The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

### Printing the Compression Distortion Summary

➤ In the Virtuoso Analog Design Environment window, choose *Results - Print - PAC Distortion Summary*.

The Results Display Window opens and displays the Distortion Summary.




The PAC Compression Distortion Summary report helps you determine the amount of distortion generated by each selected component from your schematic.

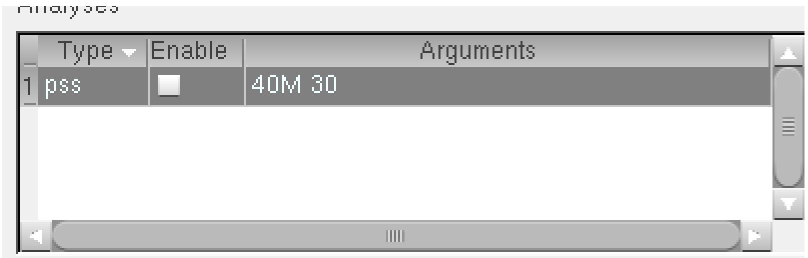
## Rapid IP3 Measurement with PSS and PAC

In this example, IP3 is quickly computed for the ne600p circuit using PSS and PAC analyses. This method uses extra function evaluations and small signal PAC analysis to evaluate higher order terms.

The rapid IP3 measurement computes the nonlinear response to RF signals around the periodically time-varying operating point. This IP3 measurement is much faster than the IP3 measurement procedure described in “Third-Order Intercept Measurement with Swept PSS and PAC” on page 318.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

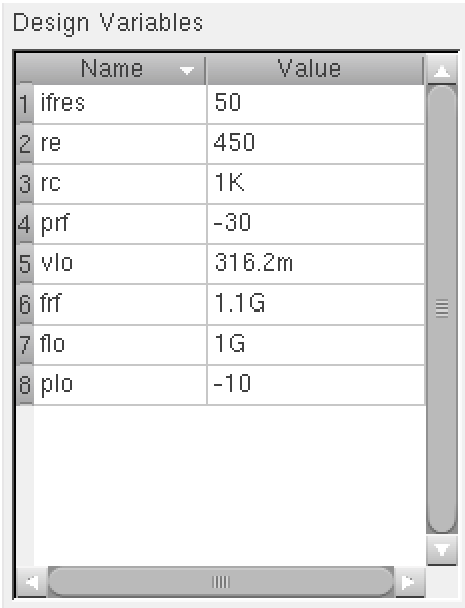



In this figure, the PSS analysis is disabled.

4. If necessary, set the design variables.

| | |
|---|---|
| flo | 1G |
| prf | -30 |
| frf | 1.1G |

5. Check the Design Variables section to verify the current design variable values.


Design Variables

| | Name | Value |
|---|---|---|
| 1 | ifres | 50 |
| 2 | re | 450 |
| 3 | rc | 1K |
| 4 | prf | -30 |
| 5 | vlo | 316.2m |
| 6 | frf | 1.1G |
| 7 | flo | 1G |
| 8 | plo | -10 |

## Editing the Schematic

Modify properties of the rf voltage source in the ne600p mixer schematic to be sure the PSS analysis response is limited to the LO signal only.

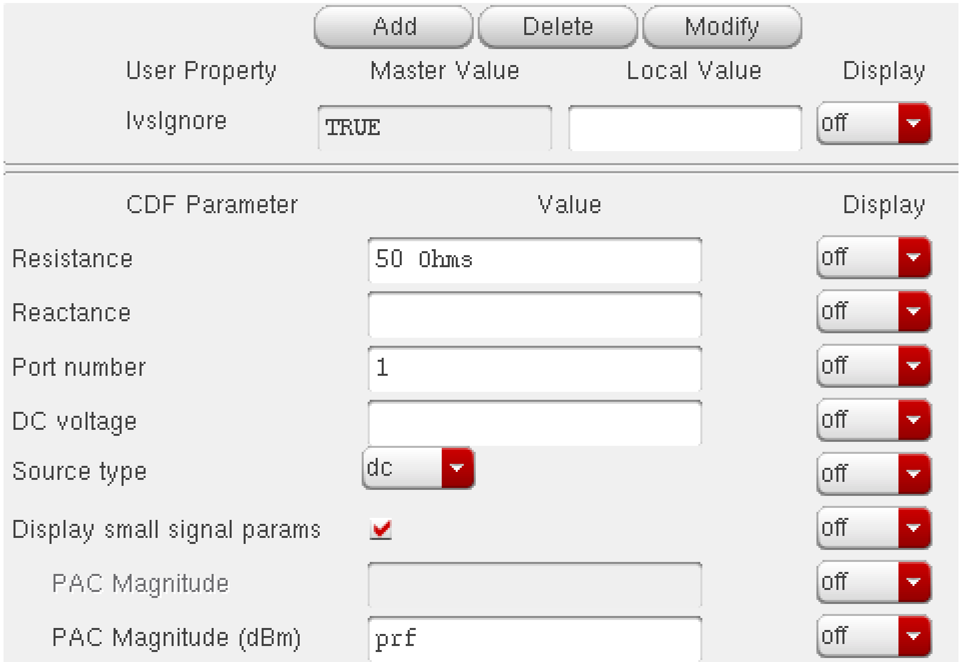
1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, make the following changes and click *OK*.

4. Choose *dc* as the *Source type*, if necessary.




5. In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

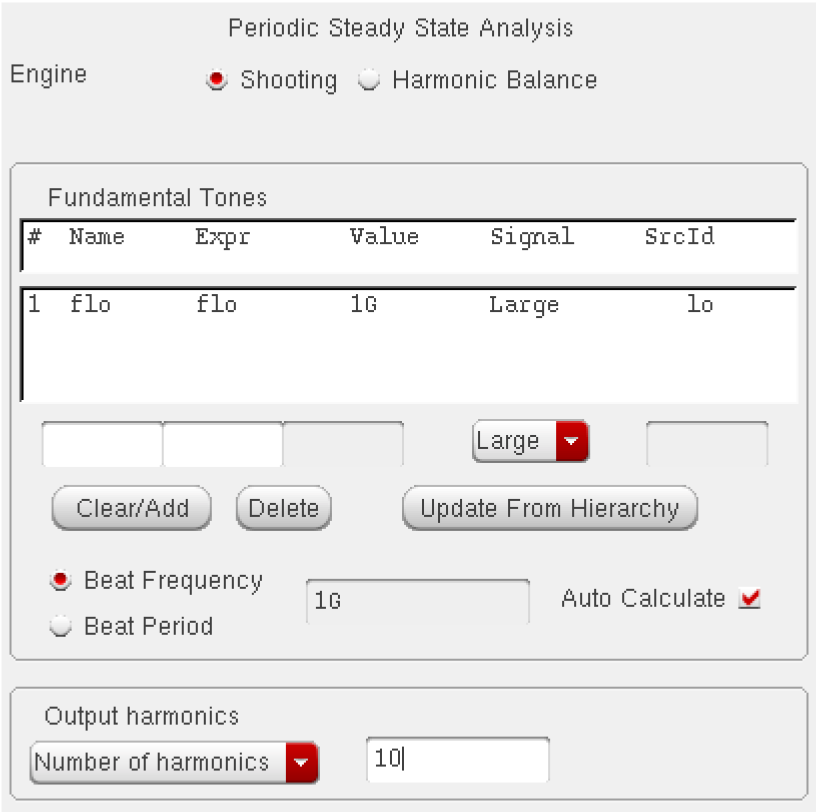
The Choosing Analyses form appears.

### Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the *Fundamental Tones* section, highlight *Auto Calculate*.

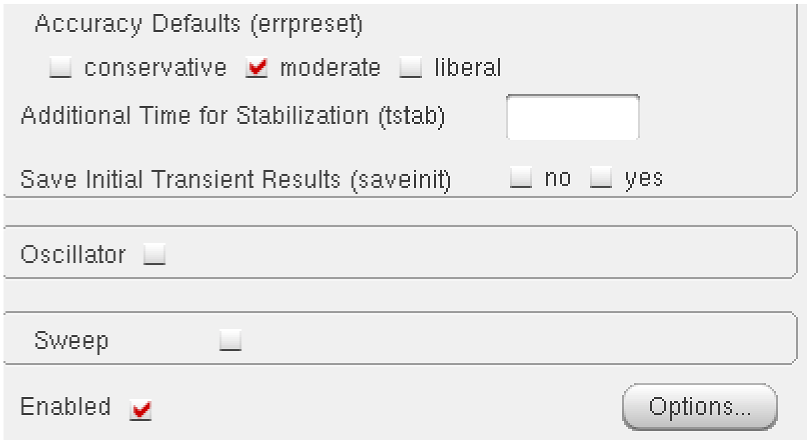
   The *Beat Frequency* is calculated as `1G`.
3. For *Output Harmonics*, select *Number of harmonics* and enter 10.

The top of the PSS Choosing Analyses form appears as follows.




4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

   The bottom of the PSS Choosing Analyses form appears as follows.




6. Click *Apply* in the PSS Choosing Analyses form.

### Setting Up the PAC Analysis

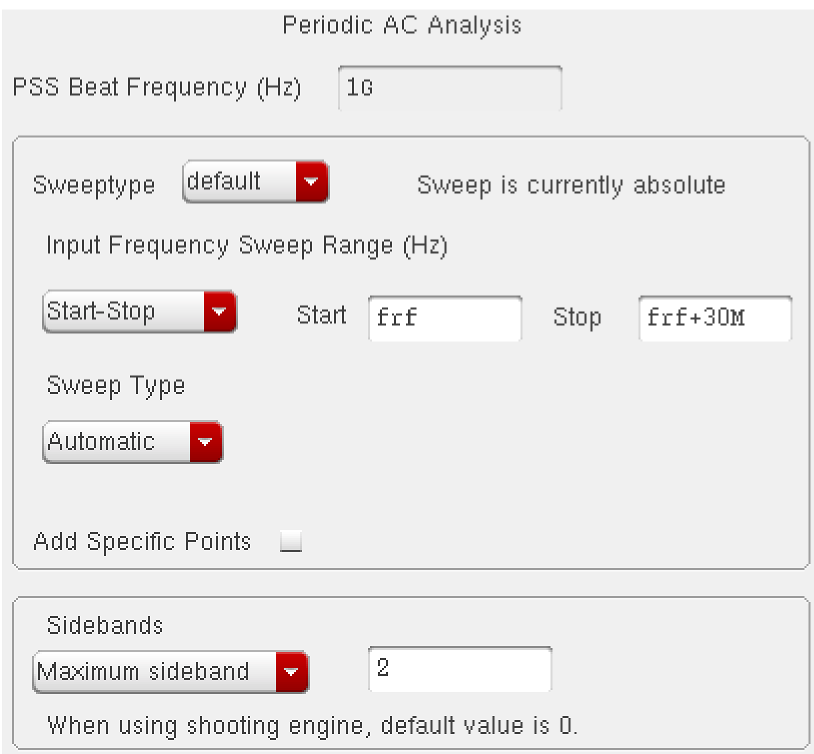
1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a *PAC* analysis.

2. In the *Input Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*.
3. Type `frf` in the *Start* field.
4. Type `frf+30M` in the *Stop* field.

   If the *Stop* field is grayed out, set *Specialized Analyses* to *None* before typing in the field.

5. In the *Sidebands* field, select *Maximum Sideband* and enter 2.
6. The top of the PAC Choosing Analyses form looks as follows.




7. In the *Specialized Analyses* cyclic field, choose *Rapid IP3*.

   The PAC Choosing Analyses form changes to display fields for setting up the Rapid IP3 calculation.

8. For the *Source Type* select *port*.

**9.** Select the RF sources and set up their frequencies.

The second RF port can be the same as the first.

**a.** To the right of *Input Sources 1* click *Select*. Then click the rf source in the Schematic window.

**b.** Press *esc* to display `/rf` in the *Input Sources 1* field.

**c.** In the *Freq* field, enter the input frequency `frf`.

This is the same as the *Start* value.

**d.** To the right of *Input Sources 2* click *Select*. Then click the same rf source in the Schematic window.

**e.** Press *esc* to display `/rf` in the *Input Sources 2* field.

**f.** In the *Freq* field, enter the input frequency `frf+30M.`

This is the same as the *Stop* value.

**10.** In the *Input Power (dBm)* field, enter `prf`. This input power value applies to both sources.

**11.** In the *Frequency of IM Output Signal* field, type `160M`, which is the IM3 frequency.

**12.** In the *Frequency of Linear Output Signal* field, type `100M`, which is the IM1 frequency.

**13.** In the *Maximum Non-linear Harmonics* field, type `4`. This number specifies the maximum harmonic used to represent nonlinear response to RF signals.

**14.** Specify the output nodes for IP3 measurement.

**a.** Next to *Output* highlight *Voltage*.

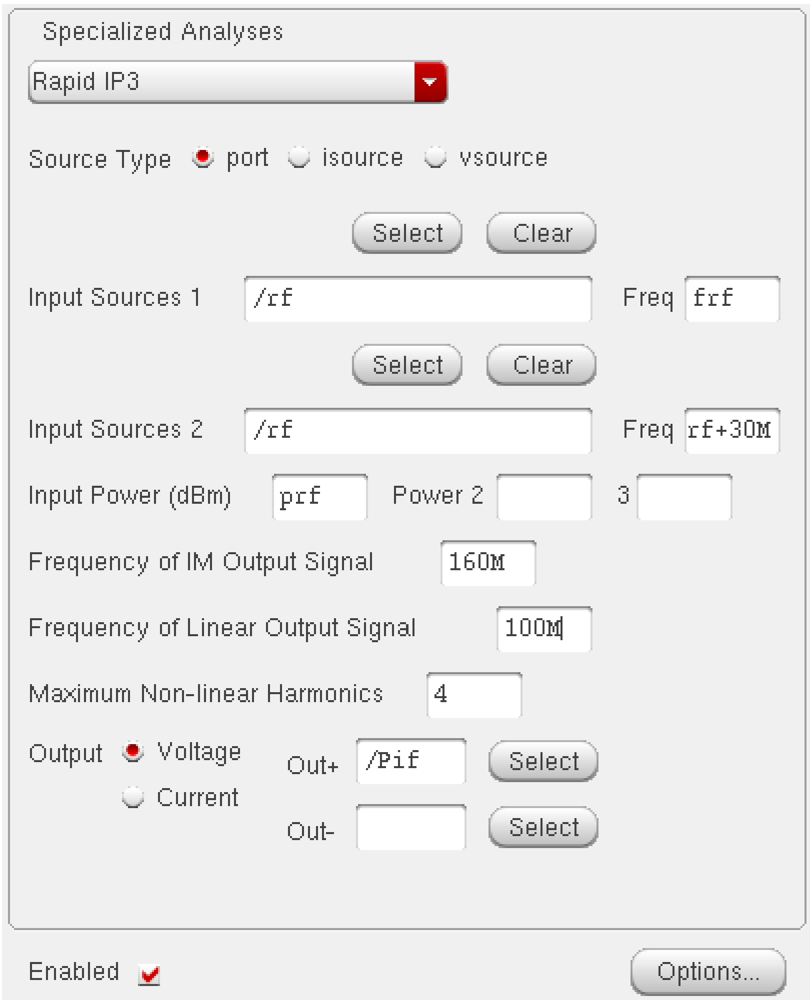
**b.** To the right of the *Out+* field, click *Select* and select the appropriate net in the Schematic window to choose *Pif*.

`/Pif` appears in the *Out+* field. This is the output node for measurement.

Leave the *Out-* field empty.

This field defaults to `/gnd!` You can set *Out-* to a different value by clicking on the *Select* button and then choosing an output node in the schematic

The bottom of the PAC Choosing Analyses form looks as follows.




**15.** Highlight *Enabled*.

**16.** Click *Apply*.

**17.** Click *OK*.

### Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

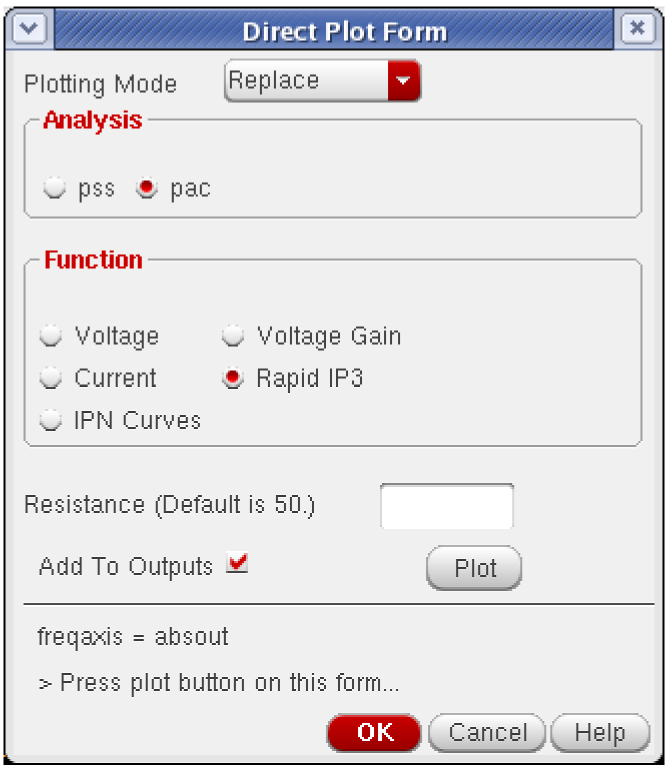
The output log file appears and displays information about the simulation as it runs.

**2.** Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Rapid IP3 Curve

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.




2. Select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pac* for *Analysis*.

   The form changes to display information for the PAC analysis.

4. Highlight *Rapid IP3* for *Function*.

   The form changes again.

5. Highlight *Add to Outputs*.

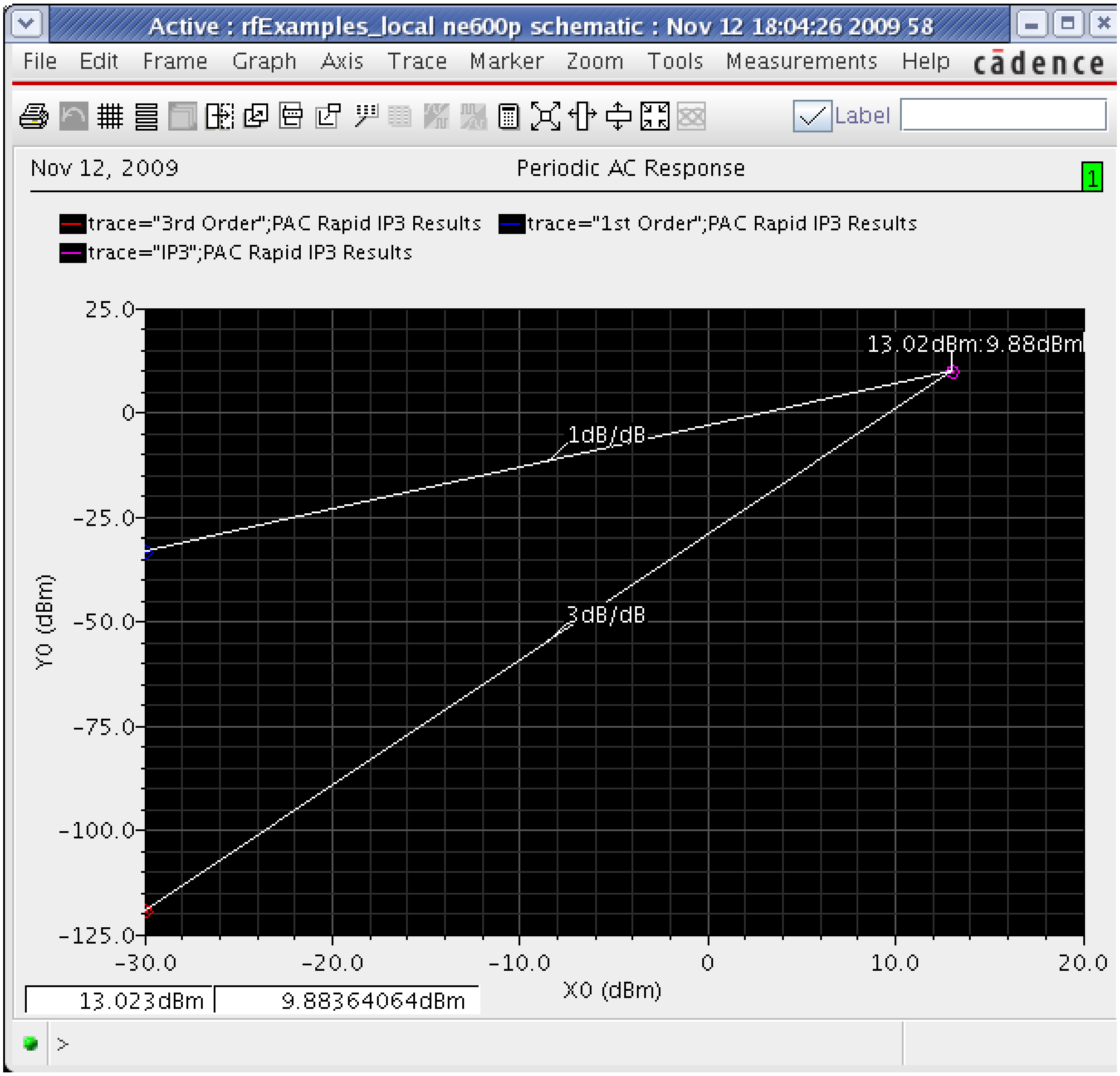
   Notice that `freqaxis = absout` for this plot.

6. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

7. Press *Plot*.

The waveform window display appears as shown below.



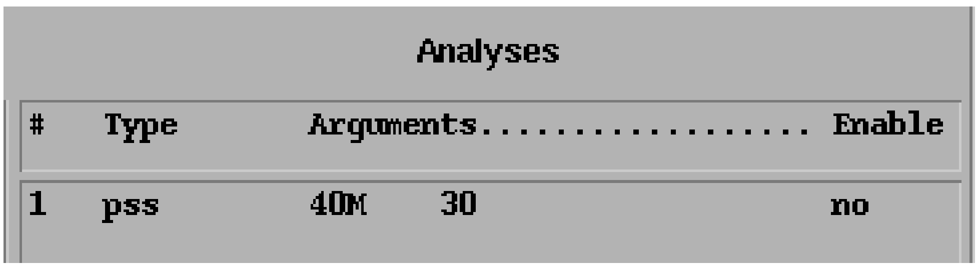

# Noise Figure Measurement with PSS and Pnoise

You can combine a PSS analysis with a small-signal Pnoise analysis to determine the noise figure for the ne600p circuit.

Pnoise analysis calculates the total noise at the output of the circuit. The equation for the noise figure is given in the Spectre RF Theory document. The Spectre RF Pnoise analysis computes the single sideband noise figure (`-1` in this case). The total noise can vary with the number of harmonics you choose because each harmonic contributes a noise component.

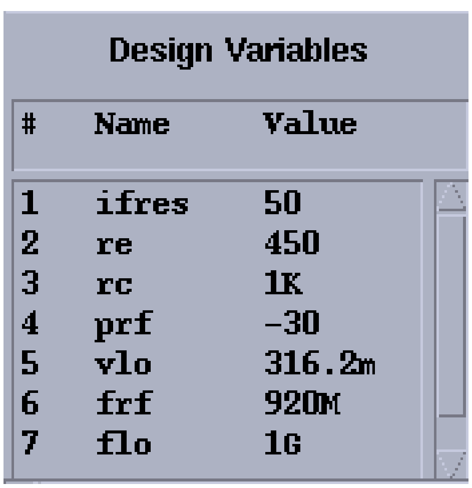
## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)




In this figure, the analysis is disabled.

If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)



## Editing the Schematic

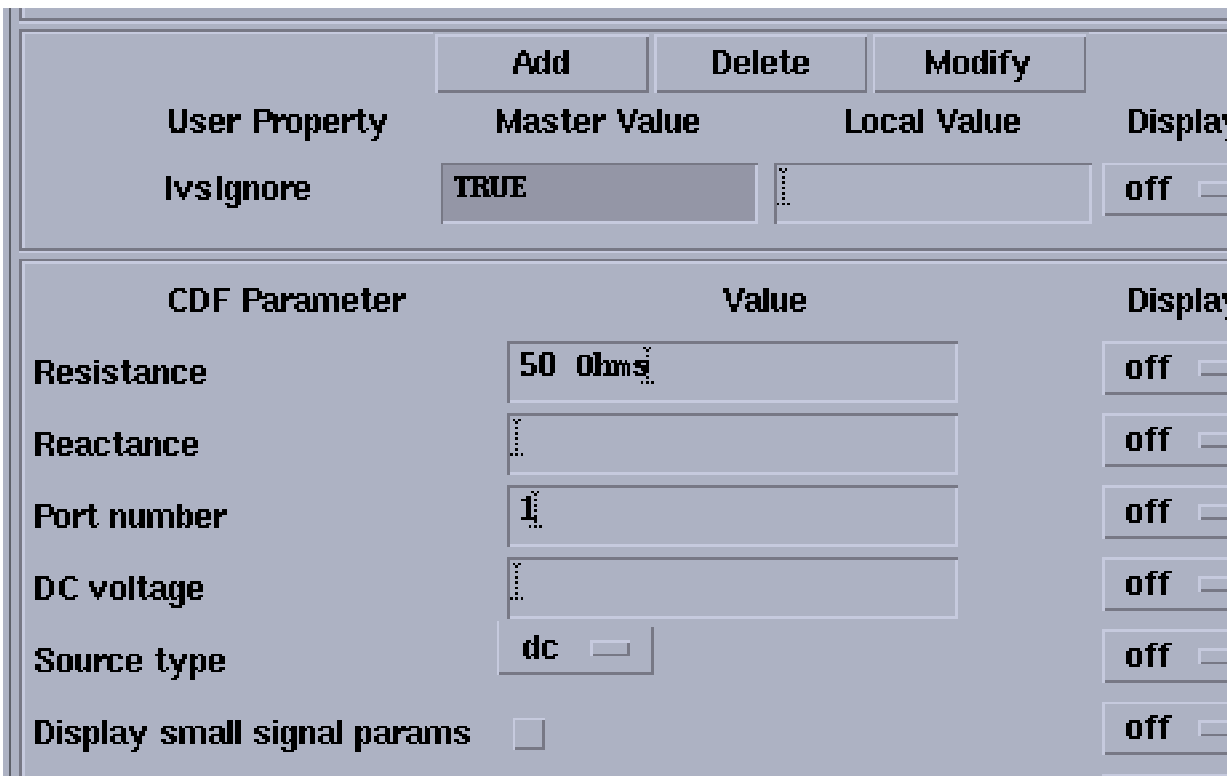
Modify the schematic to be sure the PSS analysis is the response of the ne600p mixer to *only* the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.




4. In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS and Pnoise Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

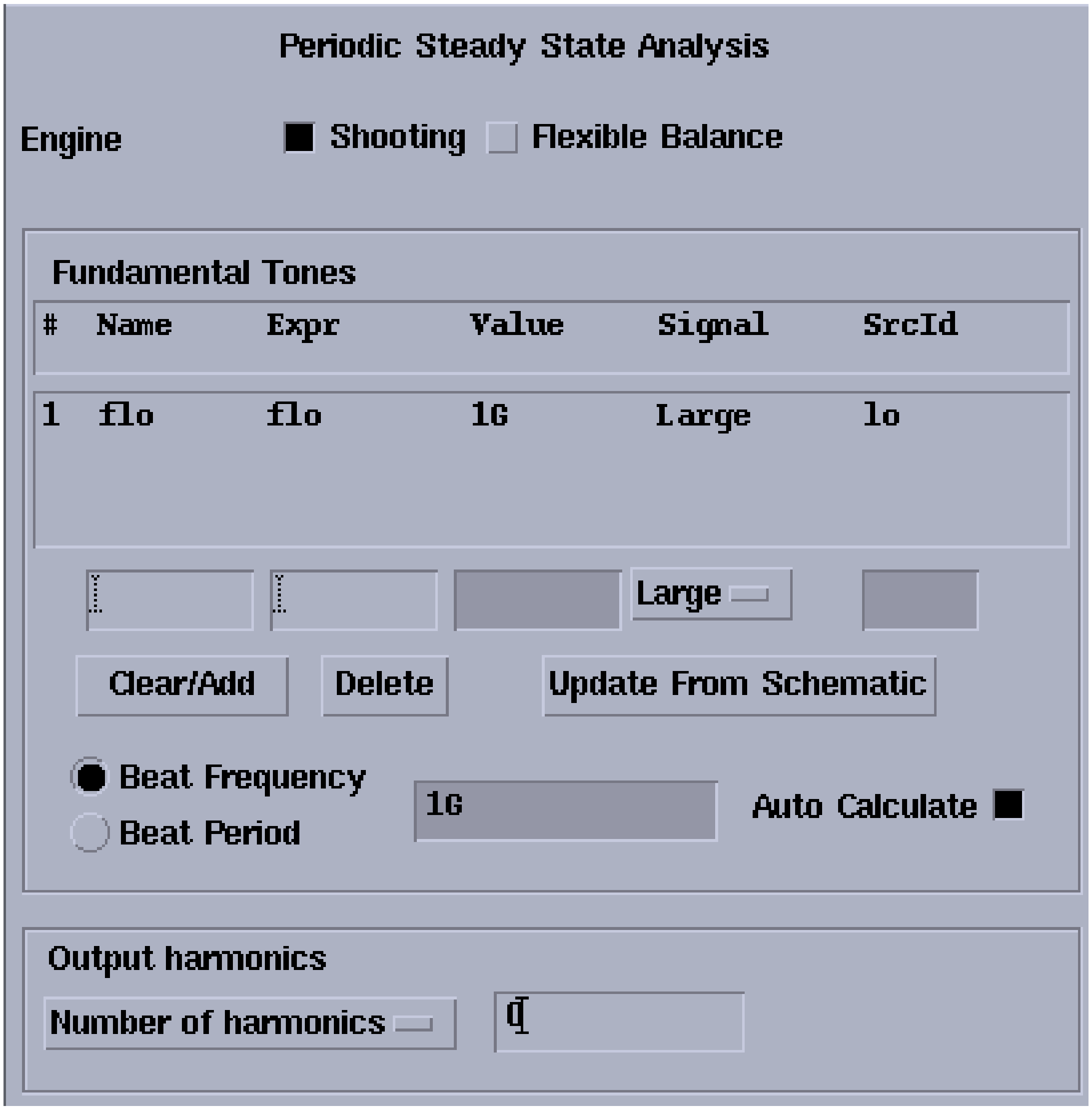
#### Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the Fundamental Tones section, highlight *Auto Calculate*.

   The *Beat Frequency* is now displayed as `1G`. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 0 in the field.

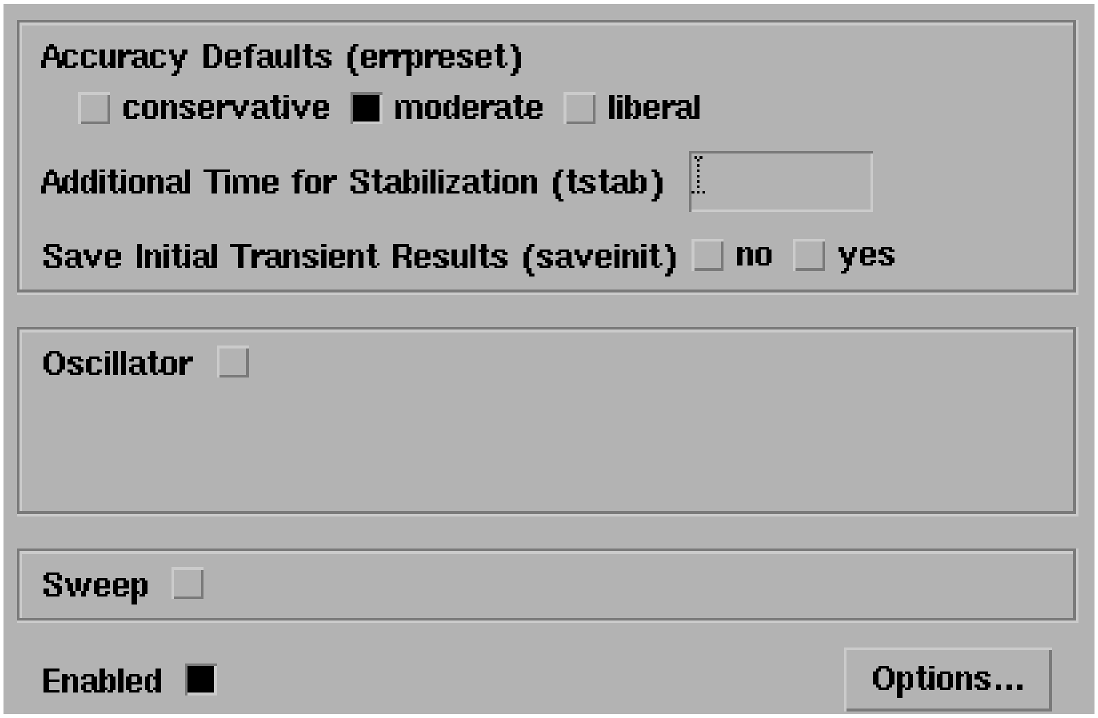
   **Note:** Pnoise requires that you set the *Number of harmonics* value to `0` to determine the circuit's response to LO only.

The top of the PSS Choosing Analyses form appears as follows.




4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

The bottom of the PSS Choosing Analyses form appears as follows.




6. Click *Apply*.

### Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, highlight *pnoise*.

   The Choosing Analyses form changes to let you specify data for a *Pnoise* analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*. Type `1K` in the *Start field* and `2G` in the *Stop* field.

   This frequency range covers the frequencies of interest, but avoids the value 0. You cannot include the value 0 in a logarithmic sweep.

3. In the *Sweep Type* cyclic field, choose *Logarithmic* for the sweep type, highlight *Points Per Decade* and type `10` in the *Points Per Decade* field.

4. In the *Sidebands* section, choose *Maximum sideband* and type `30` for the number of sidebands.

   With this setting, you specify that 30 sidebands contribute noise to the output. The value is taken from the first example (*PSS analysis only*) in this chapter.

   The top of the Pnoise Choosing Analyses form looks as follows.

5. In the *Output* cyclic field, choose *voltage* for the *Output* value.

6. Click the *Select* button for *Positive Output Node*, then click the appropriate wire in the Schematic window to choose *Pif*.

   `/Pif` appears in the *Positive Output Node* field.

7. Leave the *Negative Output Node* field empty.

   This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

8. In the *Input Source* cyclic field, choose *voltage*.

9. Click the *Select* button for *Input Voltage Source*, then click the *rf* port in the Schematic window.

   `/rf` appears in the *Input Voltage Source* field.

10. In the *Reference Side-band* cyclic field, choose *Enter in field*.

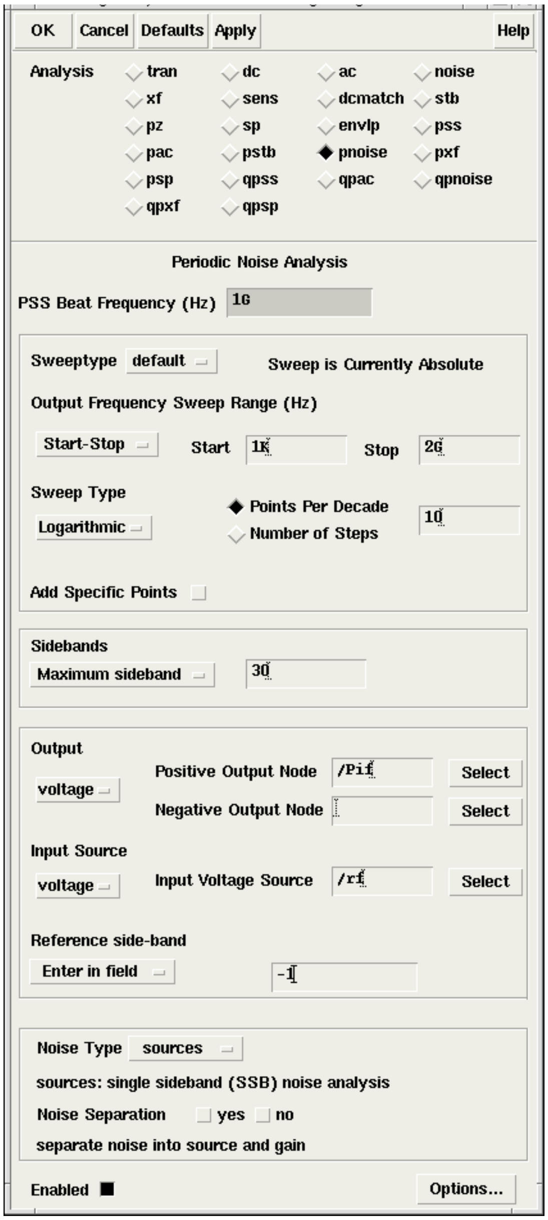
11. Type `-1` in the *Reference Sideband* field.

    In this field, you specify the difference between the input and output frequencies in the whole *frf*. The *Reference Sideband* must be `-1` because this is a down converter. Other possible choices are `0` and `+1`.

12. Leave *sources* in the *Noise Type* cyclic field to plot single sideband noise.

13. Highlight *Enabled*.

The Pnoise Choosing Analyses form looks as follows.




**14.** Click *Apply*.

**15.** Click *OK*.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

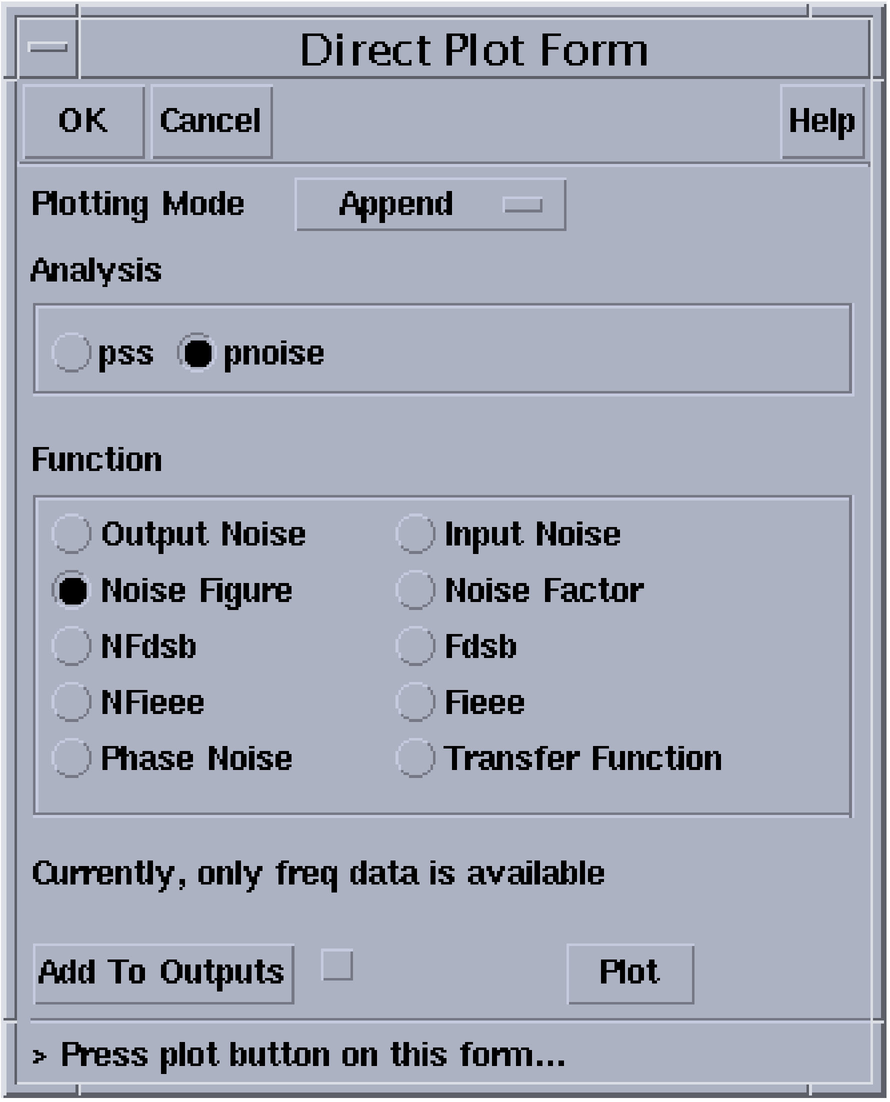
2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Noise Figure

1. To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.
2. In the Direct Plot form, select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pnoise* for *Analysis*.

   The Direct Plot form changes to display choices for results of the Pnoise analysis.
4. Highlight *Noise Figure* for *Function*.

The completed Direct Plot form appears like the one below.



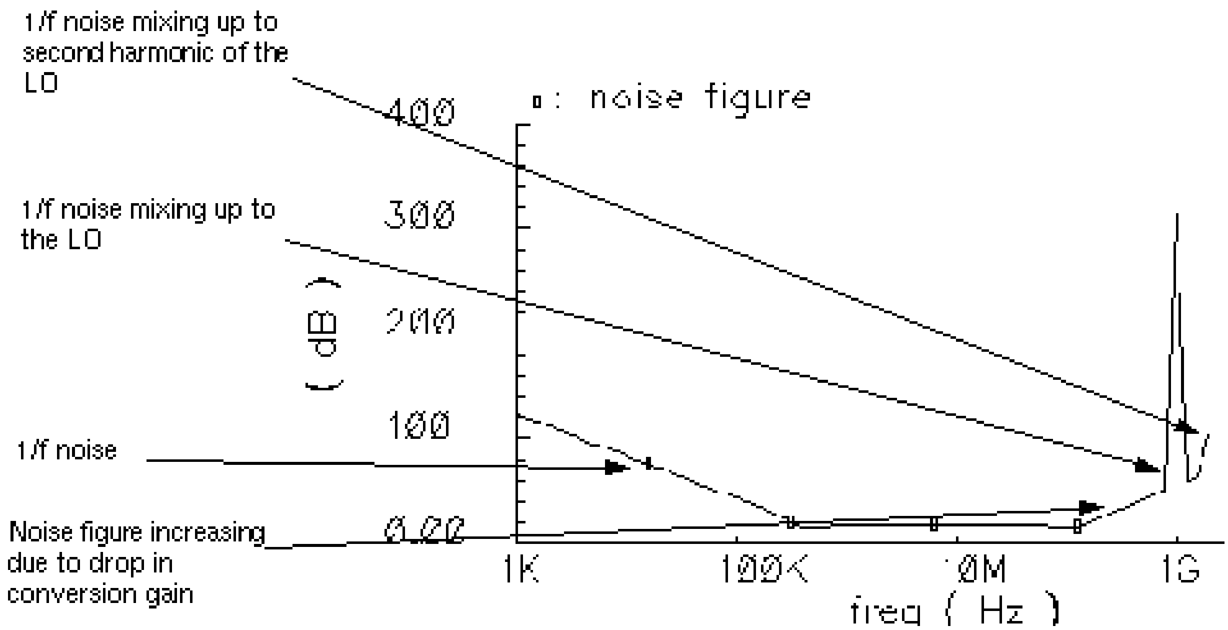

5. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click *Plot* in the Direct Plot form.

   The plot appears in the waveform window.

The waveform window displays the noise figure:




6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the waveform window. In the above plot, the noise figure is around `16.31 dB` at `80 MHz`.

## Noise Figure Measurement and Periodic S-Parameter Plots with PSS and PSP

The periodic S-Parameter (PSP) analysis computes scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and similar circuits.

In this example, you follow a large-signal PSS analysis with a small-signal PSP analysis to analyze noise folding terms induced by the RF input for the ne600p mixer circuit.

This PSP analysis computes

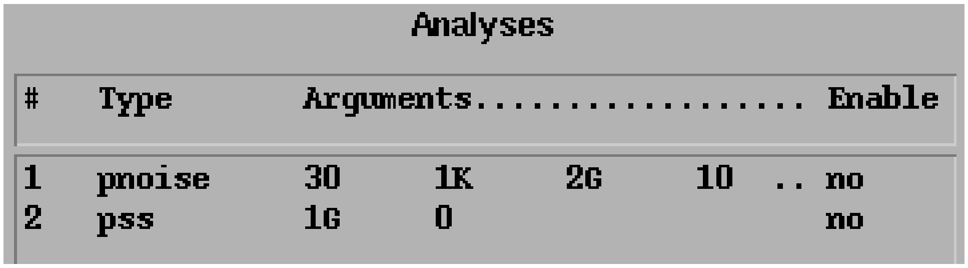
- Periodic S-parameters
- Periodic noise correlation matrices
- Noise figure
- Equivalent noise parameters

The PSP analysis is a small-signal analysis like the SP analysis, except that, as is true for PAC and PXF analysis, the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing the circuit about a periodically time-varying operating point allows for the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

The PSP analysis can also calculate noise parameters in frequency-converting circuits. PSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. The noise features of the PSP analysis include noise folding effects due to the periodically time-varying nature of the circuit. This is also true for the Pnoise analysis, but not for the SP analysis.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)


Analyses

| # | Type | Arguments.................. | | | | | Enable |
|---|---|---|---|---|---|---|---|
| 1 | pnoise | 30 | 1K | 2G | 10 | .. | no |
| 2 | pss | 1G | 0 | | | | no |

4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

**Design Variables**

| # | Name | Value |
|---|---|---|
| 2 | re | 450 |
| 3 | rc | 1K |
| 4 | prf | -30 |
| 5 | vlo | 316.2m |
| 6 | frf | 900M |
| 7 | flo | 1G |

## Editing the Schematic

The rf source used in the ne600p mixer circuit has a *Source type* property that you can set to either *sine* or *dc* depending on your application. Suppose that your RF input signal is at 900 MHz, the LO is at 1 GHz, and the IF is at 100 MHz. Before you perform a PSP analysis, you must first perform a PSS analysis.

In many cases for small-signal analysis, it is sufficient to treat the RF input as a small-signal. For example, most gain measurements are performed under small-signal conditions in which case you set the *Source type* to *dc*.

When it is important to analyze additional noise folding terms induced by the rf input, you treat the rf source as a large signal and set the *Source type* to *sine*.

See Appendix M, “Using PSP and Pnoise Analyses” for more information.

In this example, you assume the rf source is a large signal so you edit the rf port on the schematic and set the *Source type* parameter to *sine*.

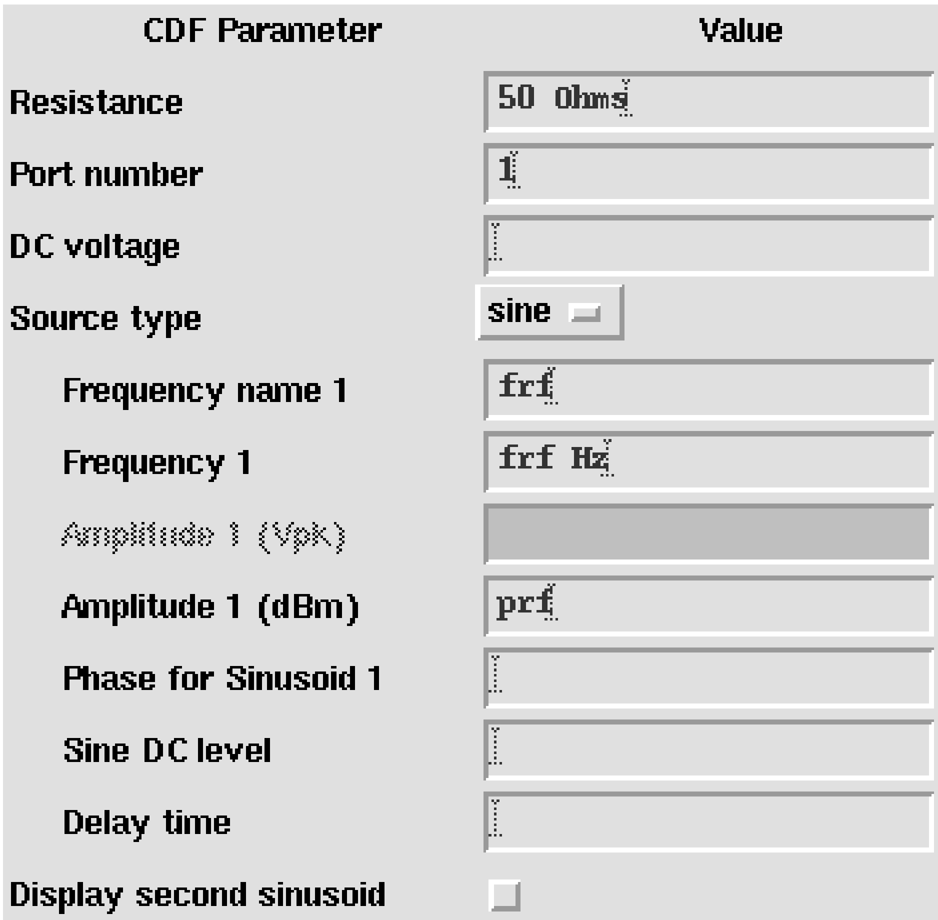
1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click the *rf* voltage source.

   The Edit Object Properties form changes to display information for the voltage source.

**3.** In the Edit Object Properties form, choose *sine* as the *Source type*, if necessary, and click *OK*.



| CDF Parameter | Value |
|---|---|
| Resistance | 50 Ohms |
| Port number | 1 |
| DC voltage | |
| Source type | sine |
| Frequency name 1 | frf |
| Frequency 1 | frf Hz |
| Amplitude 1 (Vpk) | |
| Amplitude 1 (dBm) | prf |
| Phase for Sinusoid 1 | |
| Sine DC level | |
| Delay time | |
| Display second sinusoid | |

**4.** In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS and PSP Analyses

Use the Choosing Analyses form, to set up two analyses, first PSS then PSP.

➤ Choose *Analyses – Choose* in the Virtuoso Analog Design Environment window.

The Choosing Analyses form appears.

### Setting Up the PSS Analysis

**1.** At the top of the Choosing Analyses form, highlight *pss*.

2. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed. (The *Beat Frequency* button is highlighted by default.) The 100 M value for beat frequency is the minimum period for which both RF at 900 MHz and LO at 1 GHz are periodic (or for which RF and LO are integer multiples of the fundamental frequency).

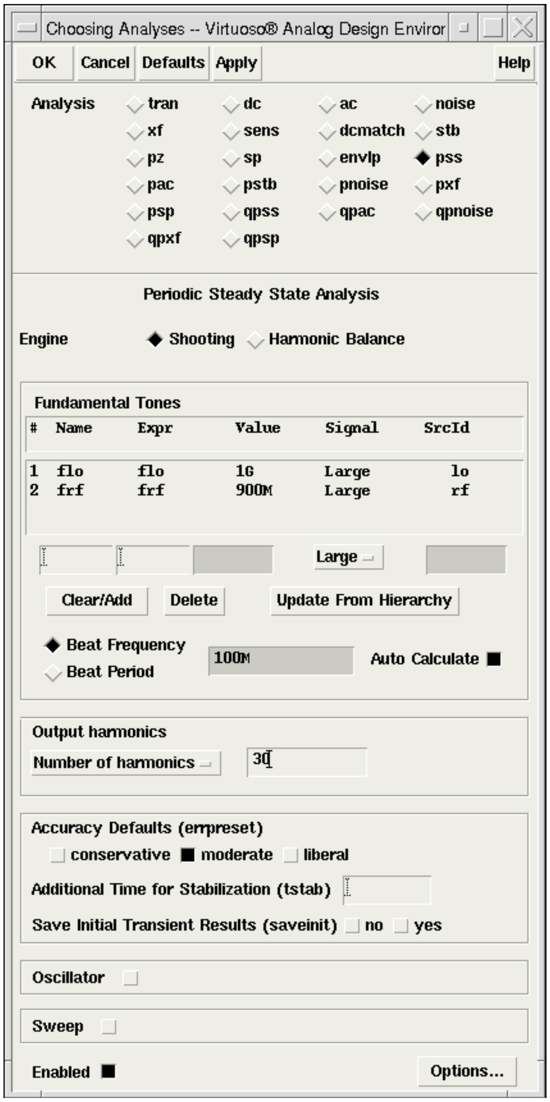
3. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `30` in the *Number of harmonics* field.

   This entry expands the plotted frequency range to include the areas of interest around both 100 MHz and 1 GHz. (30 $\times$ 100 MHz equals 3GHz, or 0 to 3GHz, and includes all important frequencies.)

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

5. Be sure the *Enabled* button is highlighted.

The correctly filled out form looks like this.




6. Click *Apply* to check the information you entered in the Choosing Analyses form for pss.

## Setting Up the PSP Analysis

1. In the *Analysis* section at the top of the Choosing Analyses form, highlight *psp*.

   The Choosing Analyses form changes to let you specify data for the *PSP* analysis.

2. In the *Sweeptype* cyclic field, choose *relative*.

The *Sweeptype* parameter controls the way that frequencies are swept in the PSP analysis. When you specify a *relative* sweep, the sweep is relative to the analysis harmonics (not the PSS fundamental frequency).

For PSP analysis, the frequencies of the input and response are usually different. This is an important difference between the PSP and SP analyses. Because the PSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the `freqaxis` and `sweeptype` parameters behave differently in PSP analysis than they behave in PAC and PXF analyses.

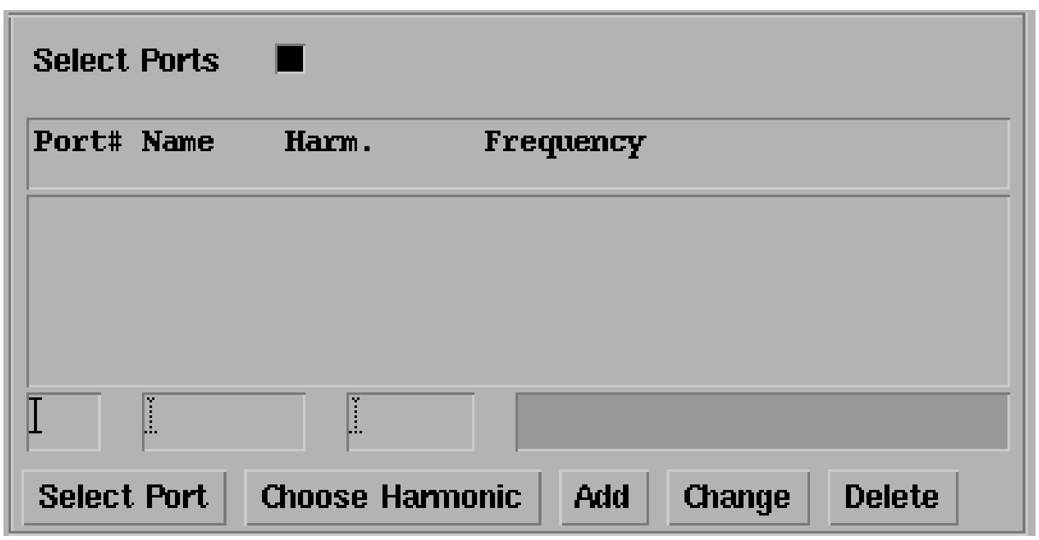
3. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop* and type `-20M` in the *Start* field and `30M` in the *Stop* field.

   The PSP analysis is performed from 20 MHz below the RF center frequency to 30 MHz above it. This frequency range accounts for inputs on the rf port in the range of -920 MHz to -870 MHz. Noise parameters (such as noise figure) are computed in a 50 MHz band around the frequency specified by the output harmonic.

4. Leave the *Sweep Type* cyclic field set to *Automatic*.
5. Highlight *Select Ports* to select the input and output ports on the schematic.

   The first port, the input port, is the *rf* port.

   The second port, the output port, is the *rif* port.




6. Select the input port.
   a. Type `1` in the first field, the *Port#* field. It is directly above the *Select Port* button.
   b. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

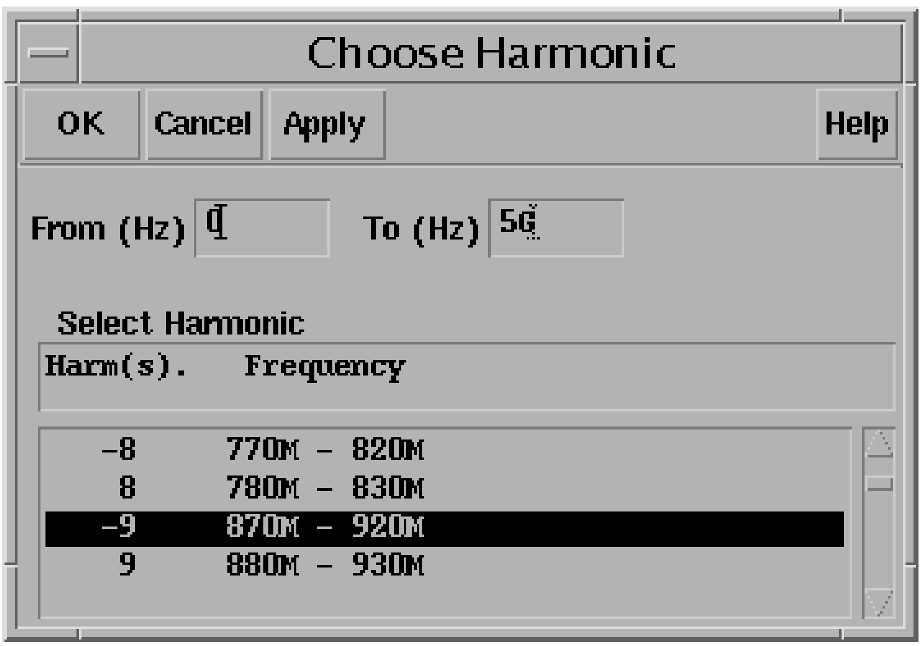
`Select source...`

**c.** In the Schematic window, click the appropriate port to choose `/rf`.

`/rf` appears in the *Name* field.

**7.** Click *Choose Harmonic*.

The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rf* port.



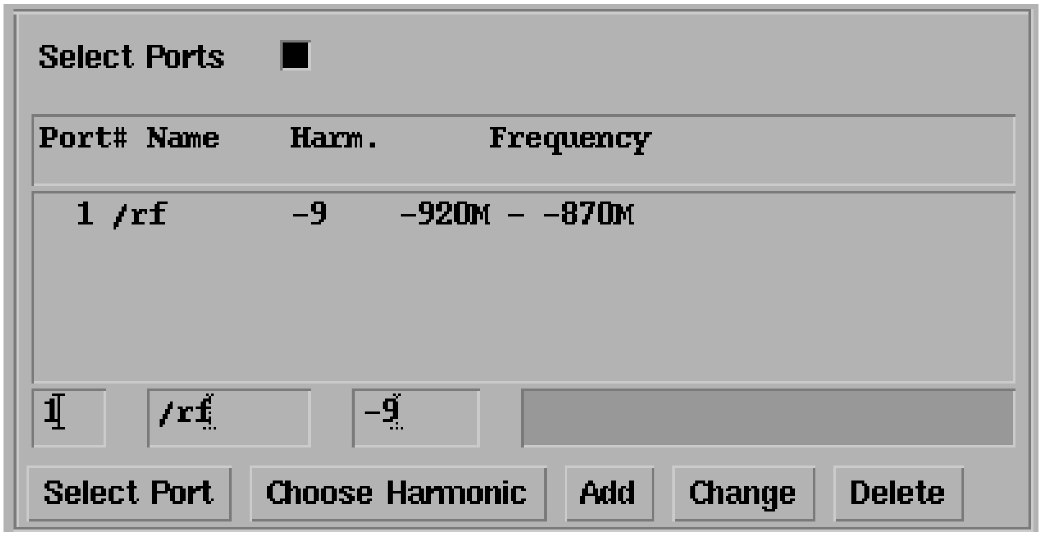

**8.** In the Choose Harmonic form, scroll through the list and highlight the harmonic with index `-9`.

**9.** Click *OK*.

The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, `-9` displays in the *Harm*. field.

**10.** In the *Select Ports* area, click *Add*.

Information for the input port displays in the *Select Ports* list box.




**11.** Select the output port.

**a.** Type 2 in the first field, the *Port#* field. It is directly above the *Select Port* button.

**b.** Click *Select Port* and follow the prompt at the bottom of the Schematic window.

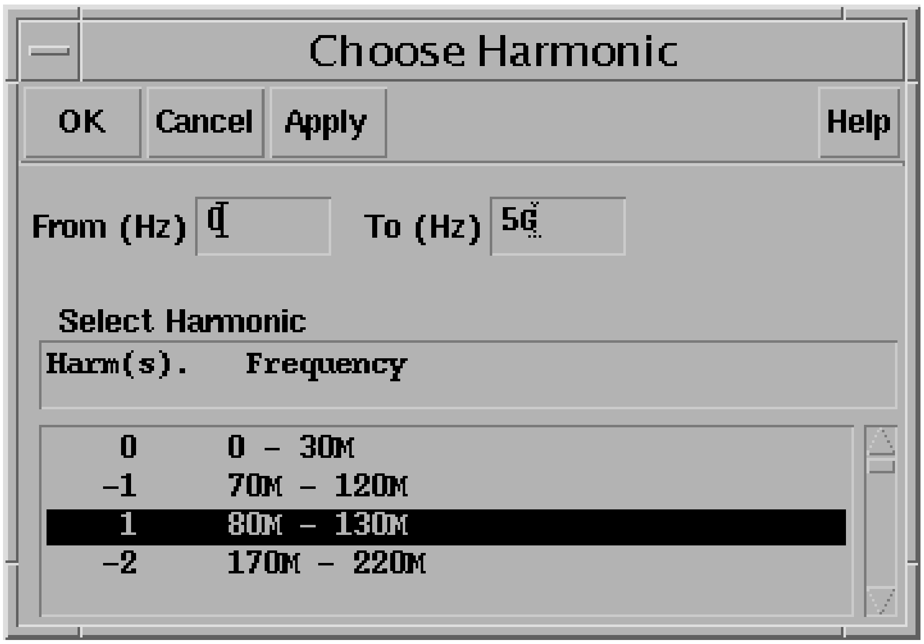
`Select source...`

**c.** In the Schematic window, click the appropriate port to choose `/rif`.

`/rif` appears in the *Name* field.

**12.** Click *Choose Harmonic*.

The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rif* port.



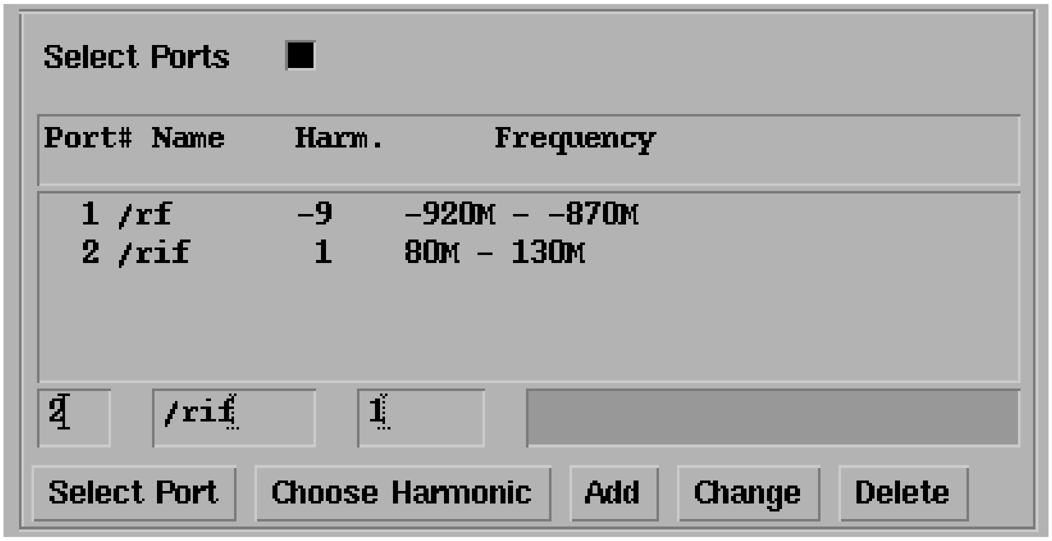

**13.** In the Choose Harmonic form, scroll through the list and highlight the harmonic with index 1.

**14.** In the Choose Harmonic form, click *OK*.

The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, 1 displays in the *Harm*. field.

**15.** In the *Select Ports* area, click *Add*.

Information for the output port is added to the *Select Ports* list box.




With the 1 GHz LO, small-signal inputs around 900 MHz (harmonics +/-9 of the PSS fundamental) appear at around 100 MHz (or harmonics -/+1 of the fundamental). You select harmonic 1 as the output harmonic, and harmonic -9 as the input harmonic because a single complex-exponential input

$$e^{i\omega_s t}$$

on the lower side of 900 MHz (harmonic -9) appears as the upper sideband of harmonic 1 at around 100 MHz.

Harmonics -9 and 1 are separated by 10 fundamental periods, which corresponds to the LO frequency of 1 GHz.

**16.** Below *Do Noise* near the bottom of the form, highlight *yes* to calculate noise parameters as part of the PSP analysis.

**17.** Type `50` in the *Maximum Sideband* field.

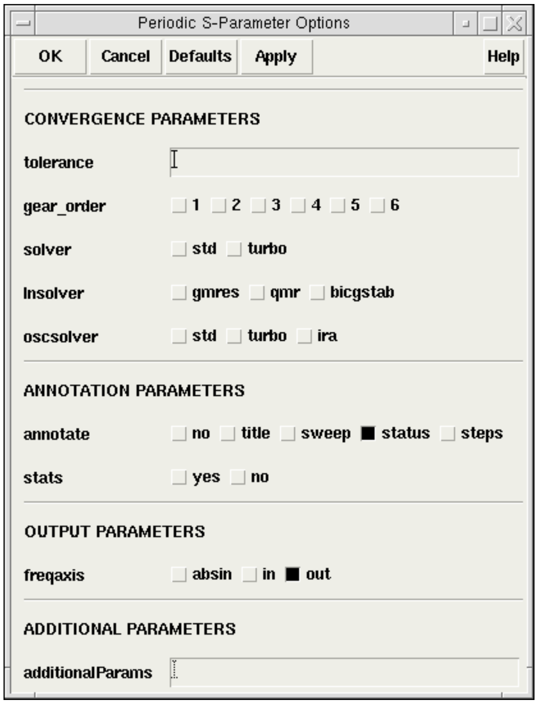
Setting *Maximum Sideband* to `50` accounts for noise folding up to 5 GHz in frequency.

All noise computations in PSP analysis involve noise folding effects. The *maxsideband* parameter specifies the maximum sideband to include for summing noise contributions either up-converted or down-converted to the output at the frequency of interest.

**18.** Highlight *Enabled*.

**19.** Click the *Options* button to display the PSP Options form.

**20.** Highlight *out* for the *freqaxis* parameter.

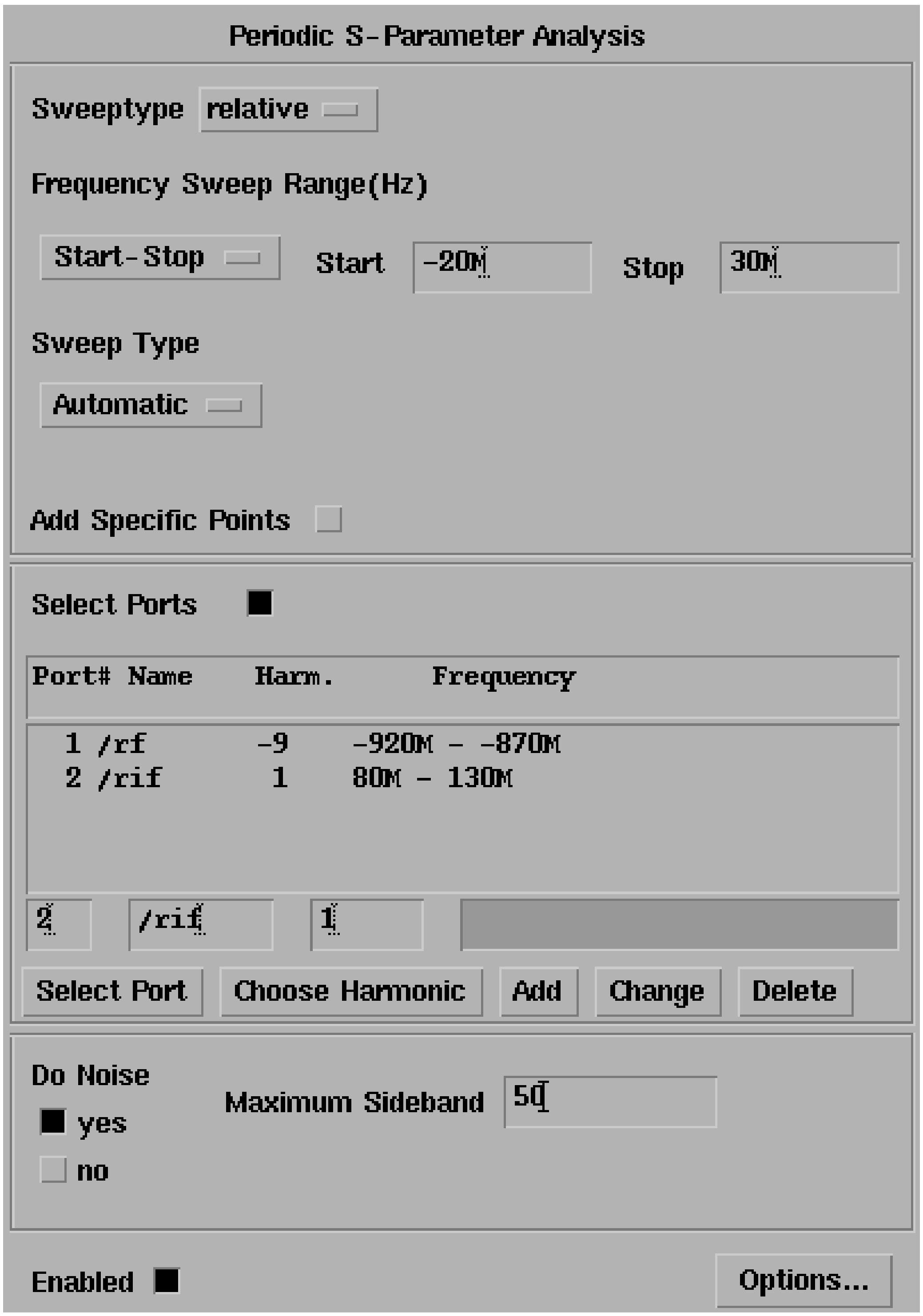



Selecting *out* changes the scale used for the output axis so that it runs from 80 MHz to 130 MHz.

The *freqaxis* parameter specifies whether the results should be output versus the absolute value of the input frequency (*absout*), the input frequency (*in*), or the output frequency (*out*).

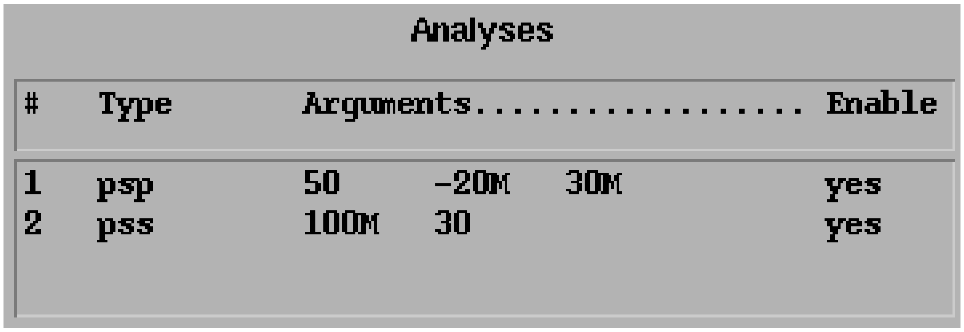
**21.** Click *OK* in the Periodic S-Parameter Options form.

The completed Periodic S-Parameter Analysis section of the Choosing Analyses form appears like the one below.



**22.** In the Choosing Analyses form, click *OK*.

Information about the `pss` and `psp` analyses appears in the Analyses section of the Virtuoso Analog Design Environment window.



Analyses

| # | Type | Arguments.................. | | | Enable |
|---|---|---|---|---|---|
| 1 | psp | 50 | -20M | 30M | yes |
| 2 | pss | 100M | 30 | | yes |

## Running the Simulation

You can now run the simulation. In addition to computing the periodic S-parameters, the periodic noise correlation matrixes are also computed, as are the noise figure and the equivalent noise parameters.

To run the simulation,

1. Choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Noise Figure

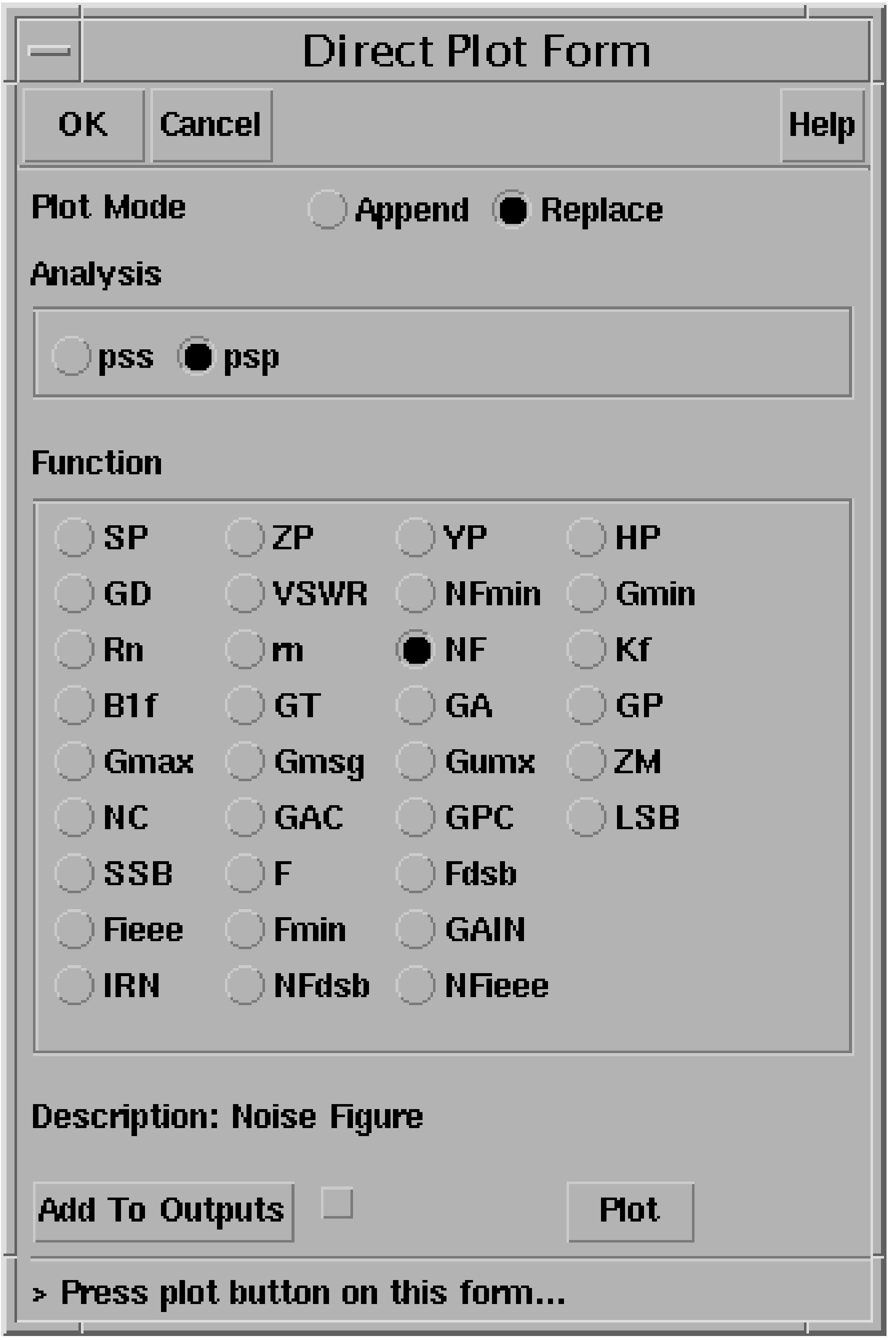
1. To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *psp* for *Analysis*.

   The form changes to display choices appropriate for PSP analysis results.

4. Highlight *NF* for *Function* to display the Noise Figure.

   The form changes to display choices appropriate for noise figure.

The completed Direct Plot form appears like the one below.

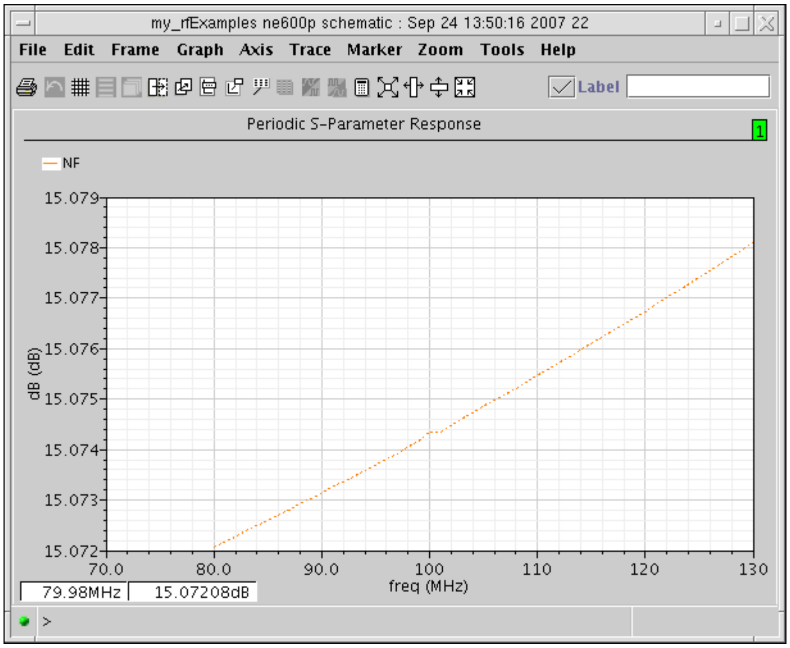



5. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click *Plot* to display the Noise Figure.

The plot appears in the waveform window.




6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve and observe the x and y values displayed at the top of the waveform window.

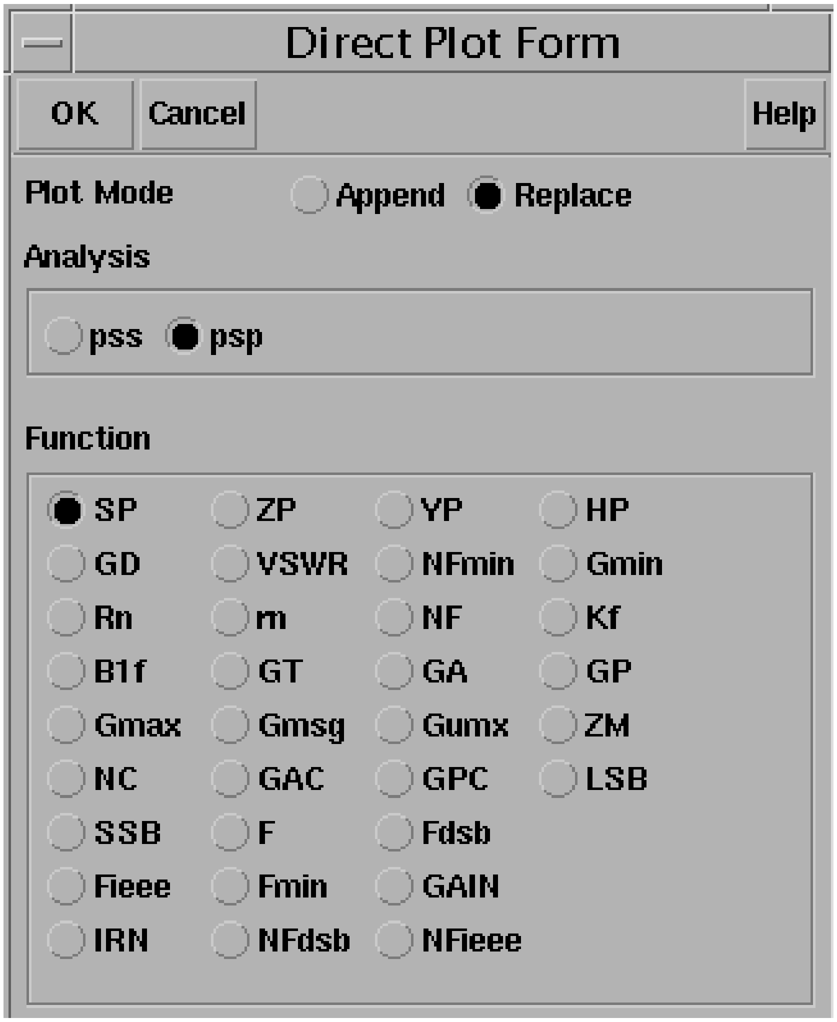
## Plotting Periodic S-Parameters

1. If necessary, open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   In the Direct Plot form, do the following:

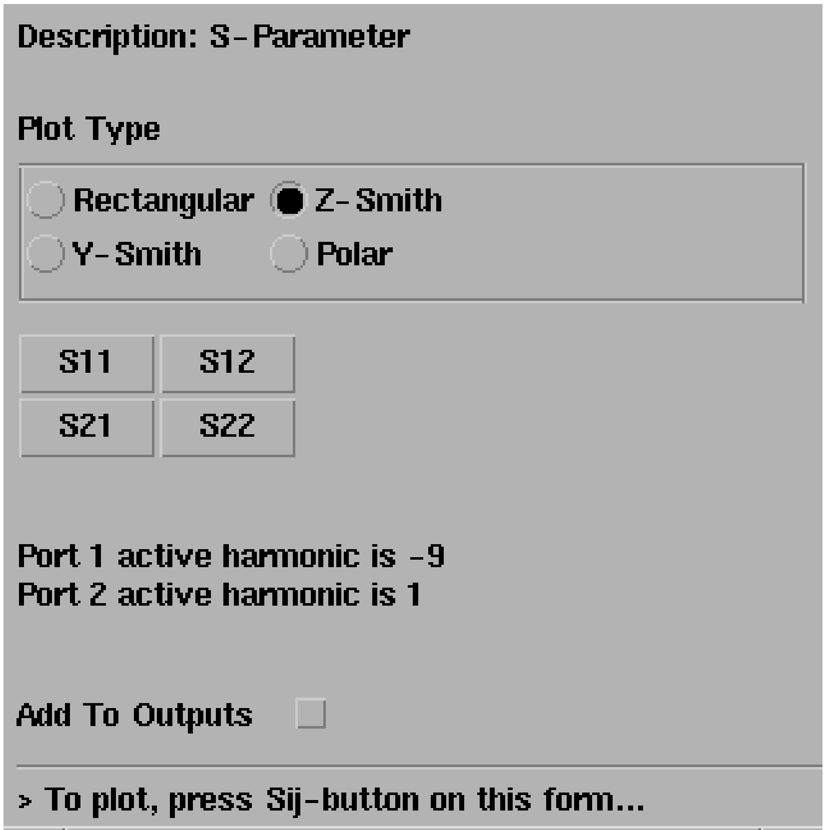
2. Highlight *Replace* for *Plot Mode*.
3. Highlight *psp* for *Analysis*.
4. Highlight *SP* for *Function* to display periodic S-parameters.

The top of the completed Direct Plot form appears like the one below.




**5.** Highlight *Z-Smith* for *Plot Type* to display periodic S-parameters.

The bottom of the completed Direct Plot form appears like the one below.




**6.** Follow the prompt at the bottom of the form

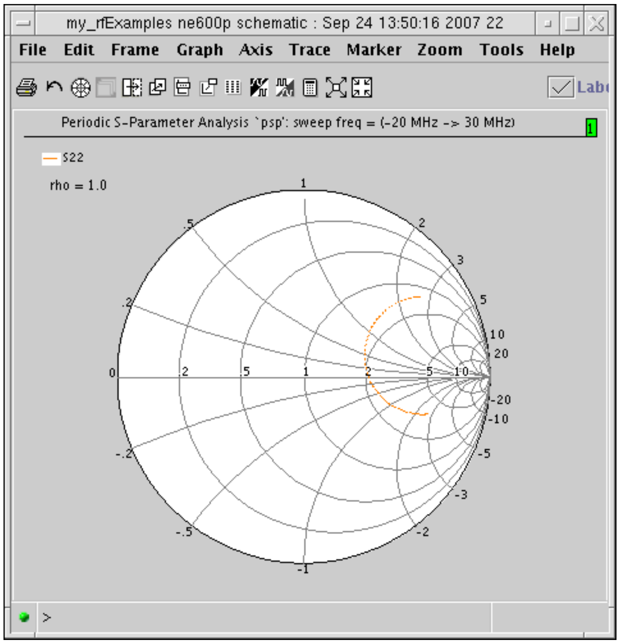
```
To plot, press Sij-button on this form...
```

Click *S22* to plot output match.

Recall from basic S-parameter theory that for a 2-port circuit S22 represents the match at port 2. In this example S22 represents the match at the `if` port.

The plot appears in the waveform window.

The waveform window display appears like the one below:




At the center of the Smith chart (at the1), the match is perfect. The match is better as the curve gets closer to the center of the Smith chart. Notice that frequency changes with match. The displayed frequency and position on the Smith chart change as you move the cursor along the curve.

The frequency where the output match is the best is approximately `105.4`MHz, and the reflection coefficient at that frequency is `341.2m` at a phase of `9.017` degrees

# Conversion Gain and Power Supply Rejection with PSS and PXF

You can combine a PSS analysis with a periodic small-signal transfer function PXF analysis to determine the conversion gain of the down converter. With changes to the plotting form, you can also calculate the power supply rejection and local oscillator feedthrough.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.

3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

## Editing the Schematic

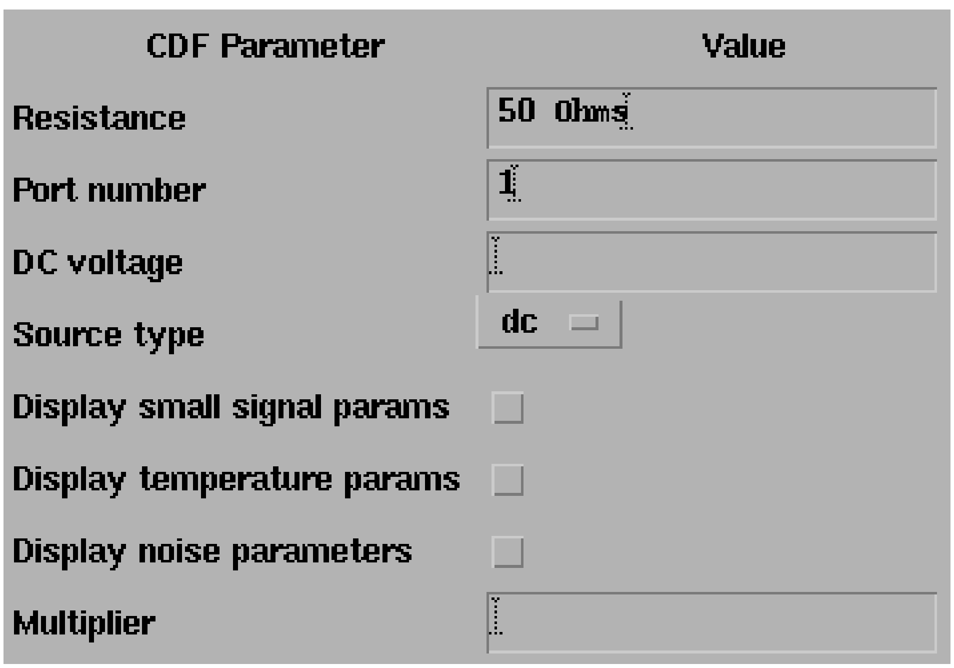
You must modify the schematic to set the RF source to a DC source to be sure the PSS analysis is the response of the `ne600p` mixer to only the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.




4. In the Schematic window, choose *Design – Check and Save*.

## Setting Up the PSS and PXF Analyses

1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

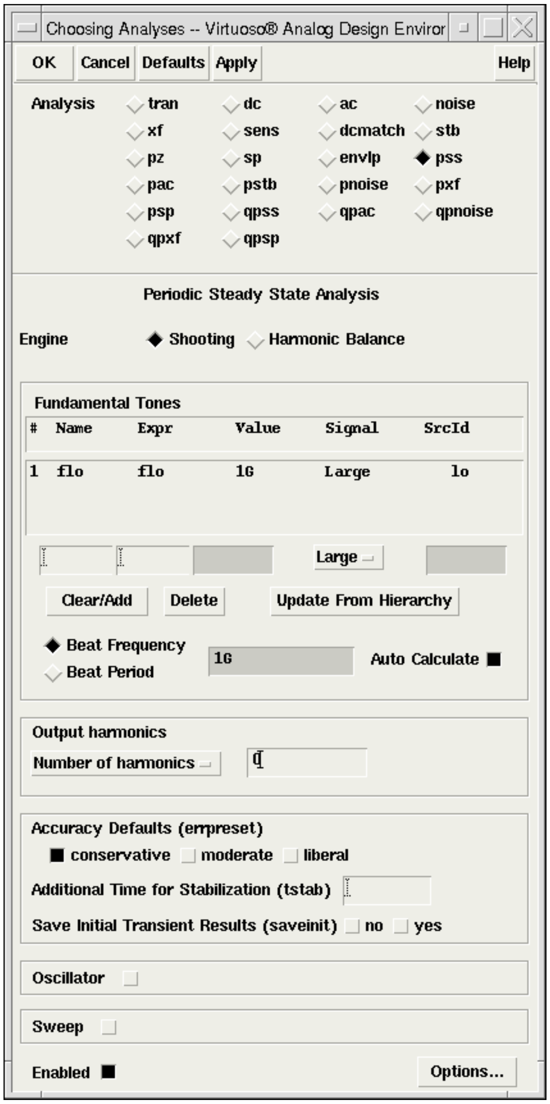
### Setting Up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed as `1G`. LO is the only time-varying signal in the circuit, so it becomes the fundamental frequency. The *Beat Frequency* button is highlighted by default.
3. In the *Output Harmonics* cyclic field, choose *Number of Harmonics* and type `0` in the *Number of Harmonics* field.

   You must turn off harmonic generation because it is required to set up the subsequent PXF analysis.
4. Highlight *conservative* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

6. The PSS Choosing Analyses form looks like this.




7. Click *Apply*.

## Setting Up the PXF Analysis

1. At the top of the Choosing Analyses form, click *pxf*.

2. The Choosing Analyses form changes to let you specify data for the PXF analysis.

3. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*, and type `1M` in the *Start* field and `300M` in the *Stop* field.

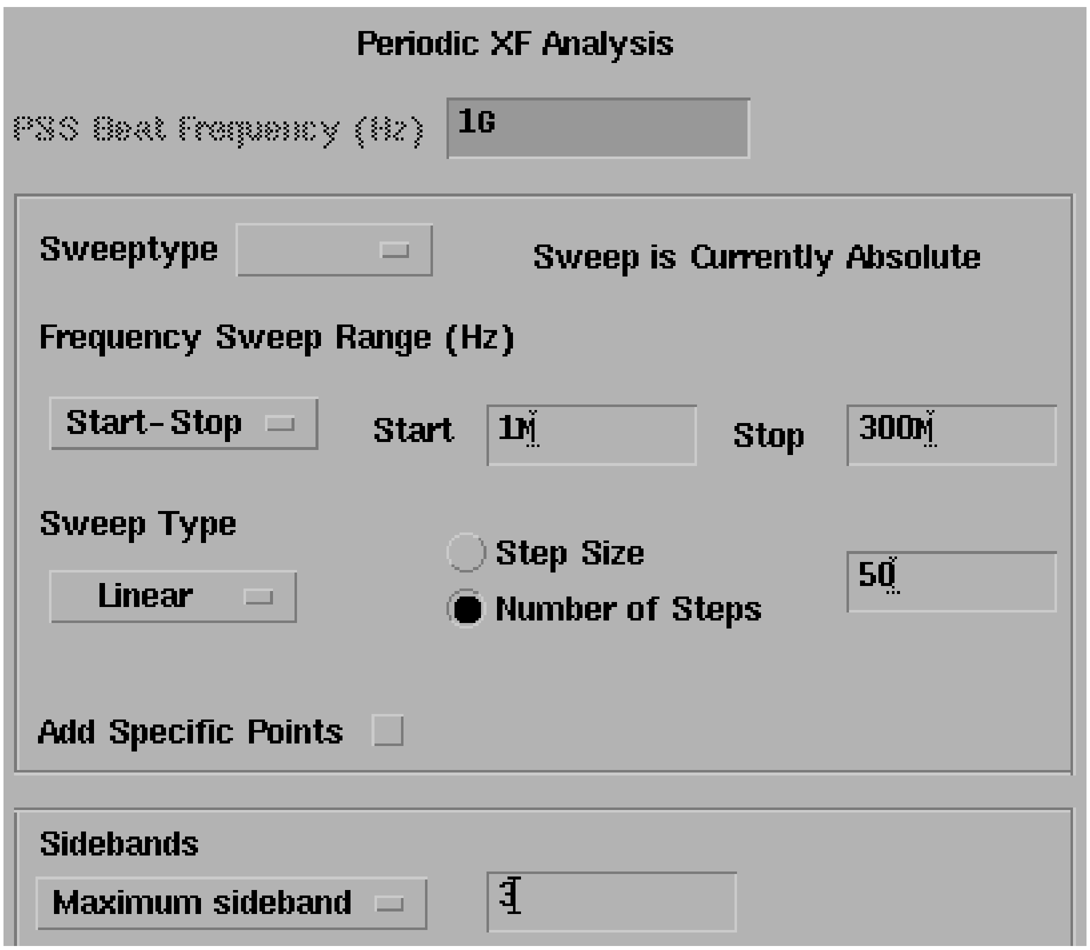
This range specification leaves space between the plots, but still shows the dramatic gain changes in the first 100 Mhz of each plot.

**4.** In the *Sweep Type* cyclic field, choose *Linear* and highlight *Number of Steps*. Type `50` for the *Number of Steps*.

Larger numbers of total points increase the resolution of the plot but also require a longer simulation time.

**5.** In the *Sidebands* cyclic field, choose *Maximum sideband* and type 3 as the value.

The top of the PXF Choosing Analyses form looks like this.




**6.** In the *Output* section, highlight *voltage*.

**7.** Click the *Select* button for *Positive Output Node*. Then click the appropriate wire in the Schematic window to choose *Pif*.

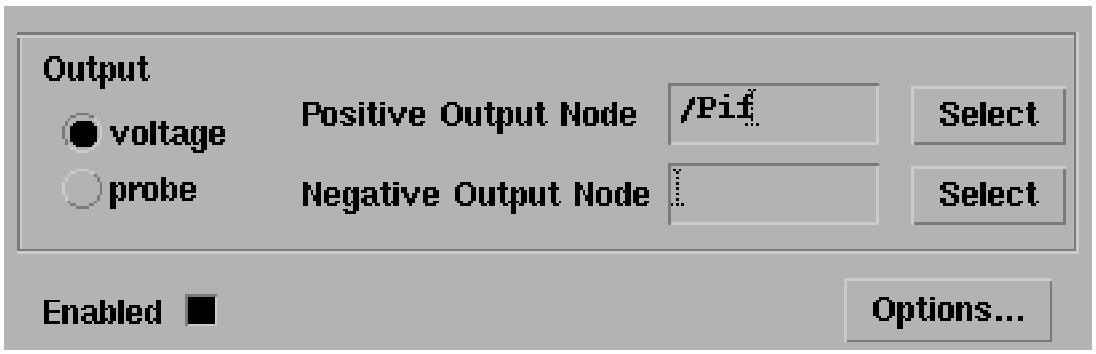
`/Pif` appears in the *Positive Output Node* field.

8. Leave the *Negative Output Node* field empty.

   This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

9. Highlight *Enabled*.

10. The bottom of the PXF Choosing Analyses form looks like this.




11. In the Choosing Analyses form, click *OK*.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation while it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Conversion Gain

1. Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   The Direct Plot form appears.

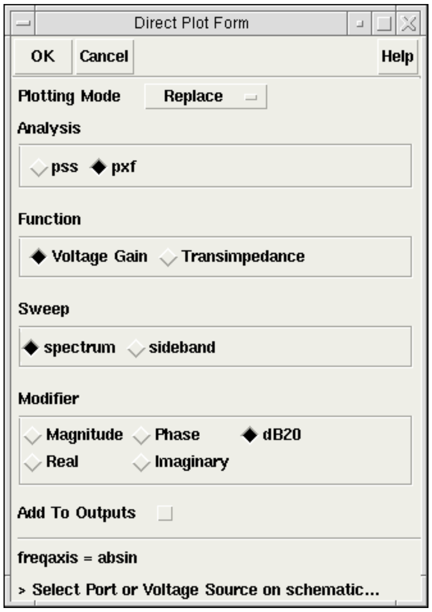
2. Highlight *Replace* for *Plot Mode*.

3. Highlight *pxf* for *Analysis*.

4. Highlight *Voltage Gain* for *Function*.

5. Highlight *dB20* for *Modifier*.

The completed PXF Direct Plot form looks like this.

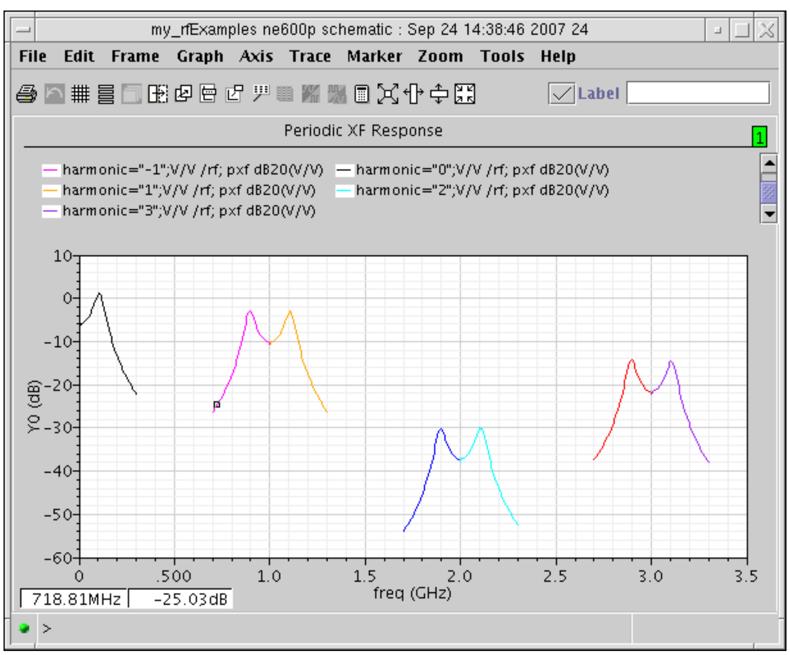



6. Follow the prompt at the bottom of the form

   `Select Port or Voltage Source on schematic...`

   Click the *Prf* source on the schematic.

   The plot appears in the waveform window.



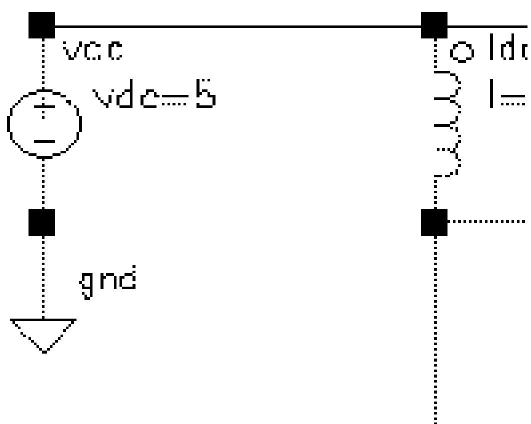
7. To determine the gain at different frequencies, move the cursor along the curve and use the readout at the top of the waveform window.

   For example, the gain is about `-5.46 dB` at `920 MHz`.

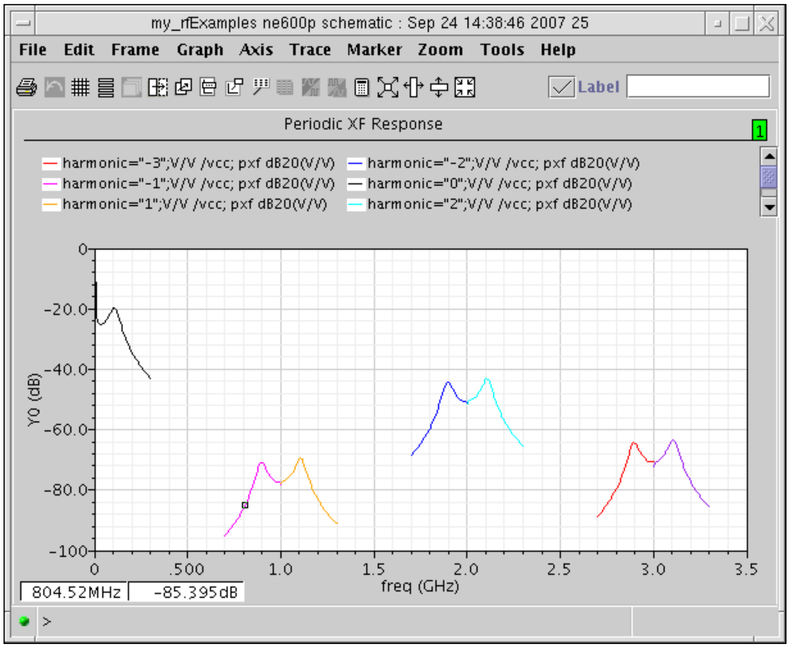
## Plotting the Power Supply Rejection

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pxf* for *Analysis*.
3. In the Schematic window, click the *vcc* power supply at the top left of the schematic.



The waveform window appears as follows.




4. To determine the rejection at different frequencies, move the cursor along the curve and use the readout in the lower-left corner of the waveform window.

# Calculating the 1 dB Compression Point with Swept PSS

In this example, a swept PSS analysis determines the `1dB` compression point for the `ne600p` mixer configured as a down converter.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

## Editing the Schematic

1. In the Schematic window, click the *rf* voltage source.
2. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.
3. Choose *sine* for *Source type*.
4. Type `prf` in the *Amplitude 1 (dBm)* field.
5. Click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

## Setting Up the Swept PSS Analysis

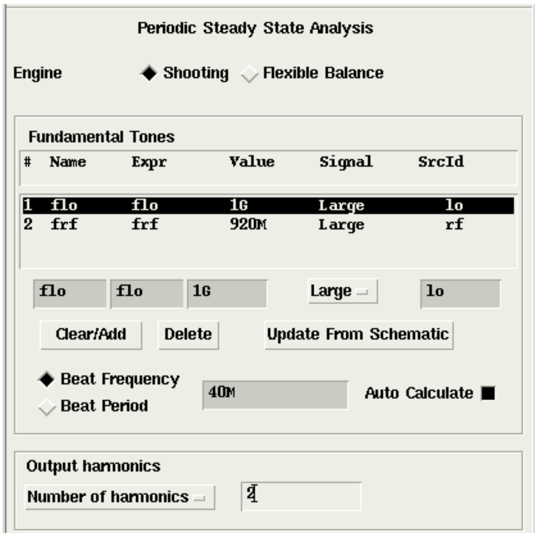
1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, choose *pss* for the *Analysis*.
3. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed as `40M`. Now that the RF signal is active again, the fundamental frequency of the circuit goes back to `40 MHz`. The *Beat Frequency* button is highlighted by default.
4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 2 in the *Number of harmonics* field.

   Only two harmonics are required to determine the 1 dB compression point.

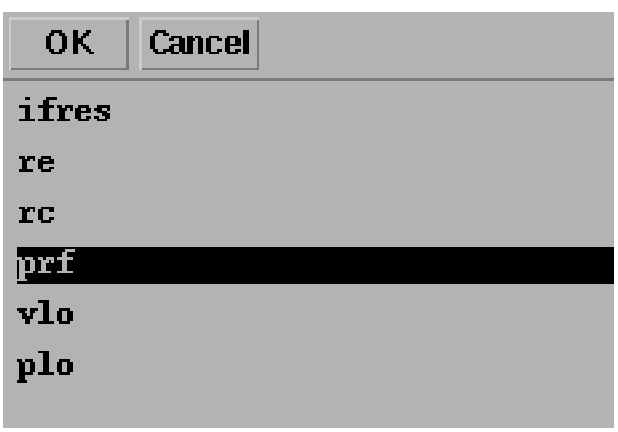
The top of the Swept PSS Choosing Analyses form looks like the following.




5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
7. In the *Sweep* cyclic field, choose *Variable*.
8. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
9. In the Select Design Variable form, highlight *prf* and click *OK*.




The *Variable Name* `prf` appears in the Choosing Analyses form. Selecting the `prf` variable sweeps RF input.

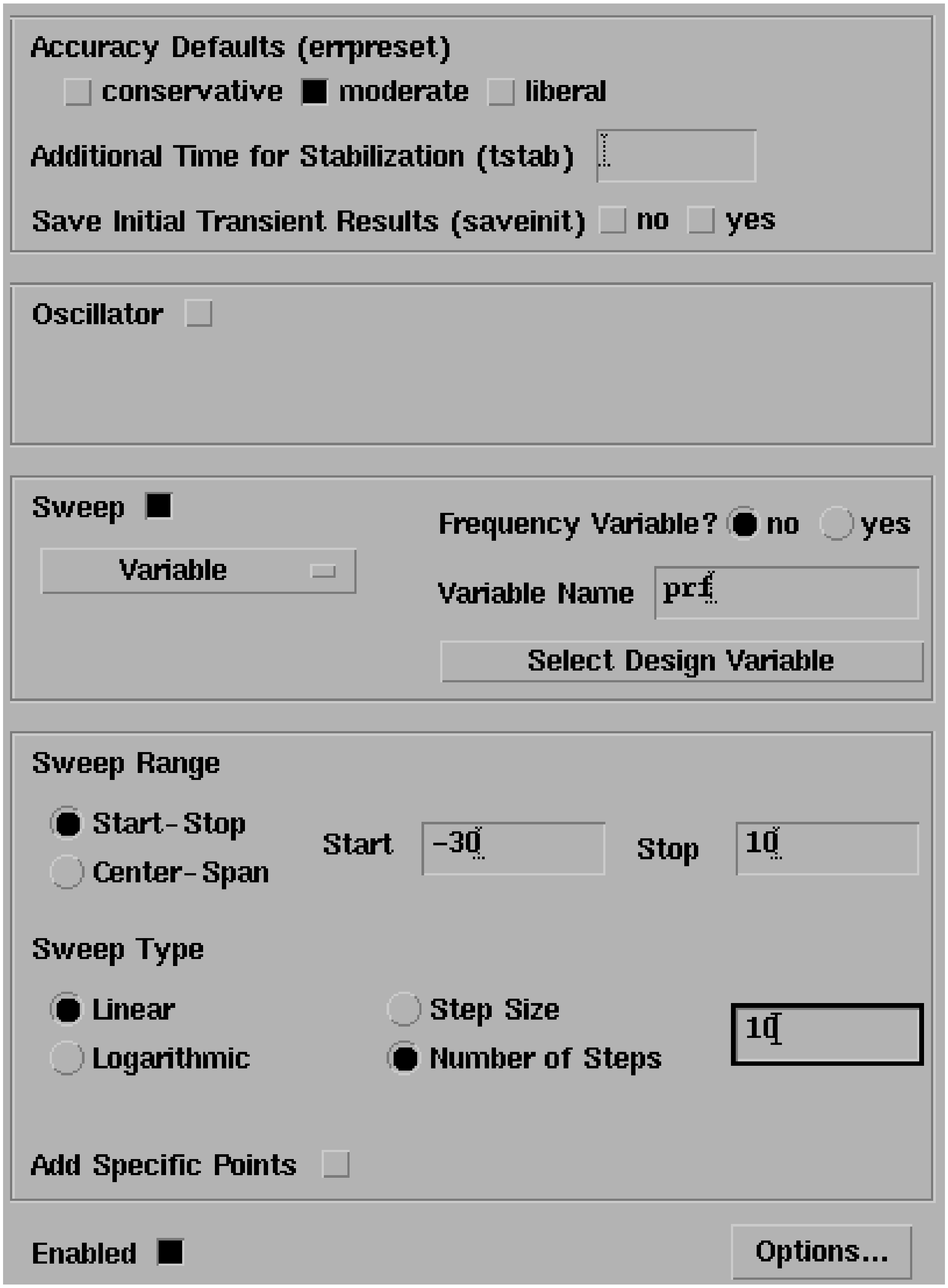
Choose *Start-Stop* for the *Sweep Range*, and then type `-30` in the *Start* field and `10` in the *Stop* field.

Both the signal source and the sweep are done in dBm. You learn where to set the sweep limits and how many points to include in the sweep with experience.

**10.** Choose *Linear* for the *Sweep Type*, and specify `10` for the *Number of Steps*.

**11.** Highlight *Enabled*.

The top of the Swept PSS Choosing Analyses form looks like the following.




**12.** In the Choosing Analyses form, first click *Apply* then click *OK*.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the 1 dB Compression Point

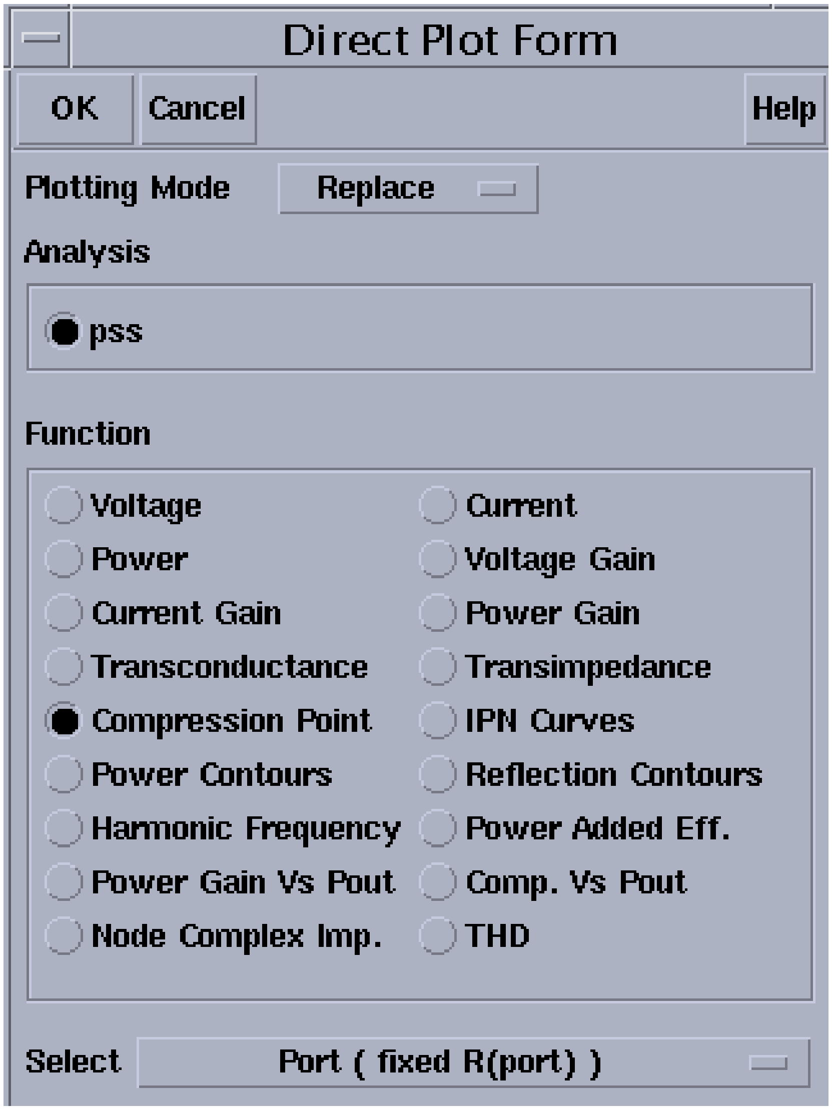
The first sideband of the IF signal is the second harmonic of the 40 MHz fundamental. To plot the 1 dB compression point, perform the following:

1. Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plotting Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Compression Point* for *Function*.

5. The top of the Direct Plot form looks like this.




The form changes to display fields for compression point calculation.

Type `1` for *Gain Compression (dB)*.

6. Type `-25` for *Input Power Extrapolation Point (dBm)*.

   This value specifies the point where the ideal amplification curve intersects the output curve. You must estimate where this point is located. If you do not specify a value, the plot defaults to the minimum variable value.

7. In the cyclic field, choose *Input Referred 1dB Compression*.

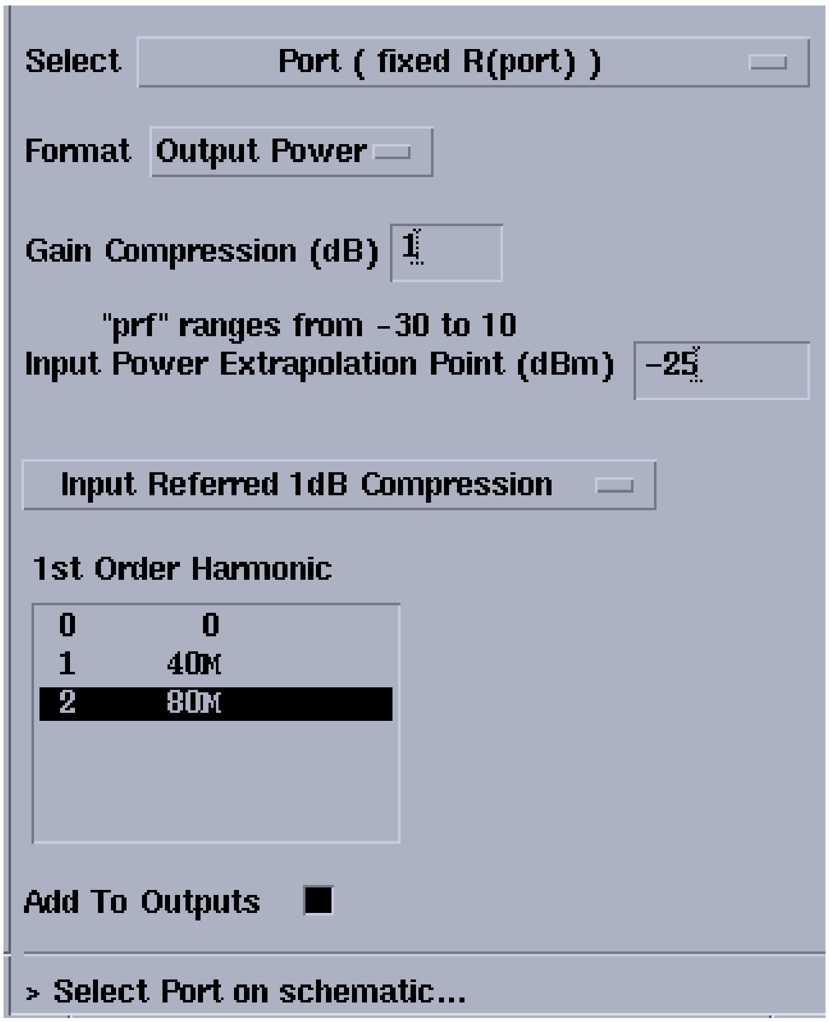
8. Follow the prompt at the bottom of the form

`Select 1st Order Harmonic on this form...`

**9.** In the *1st Order Harmonic* list box, highlight harmonic *2 (80 MHz)*.

This is the second harmonic of the 40 MHz fundamental frequency, which is the IF frequency.

The bottom of the Direct Plot form looks like this.

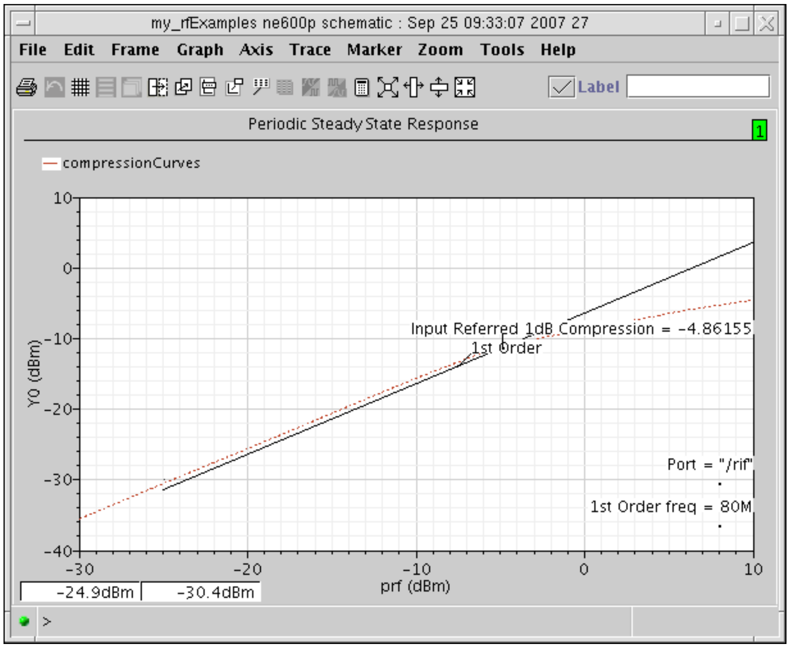



**10.** Follow the new prompt at the bottom of the form

`Select Port on schematic...`

In the Schematic window, click the *if* port.

The waveform window display looks like this.




The Spectre RF simulation plots the signal curve and an ideal linear slope and automatically calculates a value of `-4.86155 dBm` for the 1 dB compression point.

If you have trouble reading the label attached to the 1 dB point, you can use the cursor to drag it to another location.

# Third-Order Intercept Measurement with Swept PSS and PAC

In this example, you combine a swept PSS analysis with a Periodic AC (PAC) analysis to produce data for an IP3 plot.

To calculate an IP3 measurement, you could use settings similar to those used for the 1 dB compression point, but with another tone for the second RF signal. That would be a swept PSS with three tones. The method described here, however, runs more quickly because

- It processes the second RF tone only during the PAC analysis
- It considers only two of the second RF tone sidebands

  The swept PSS analysis for the 1 dB compression point example considered all sidebands for all signals.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

## Editing the Schematic

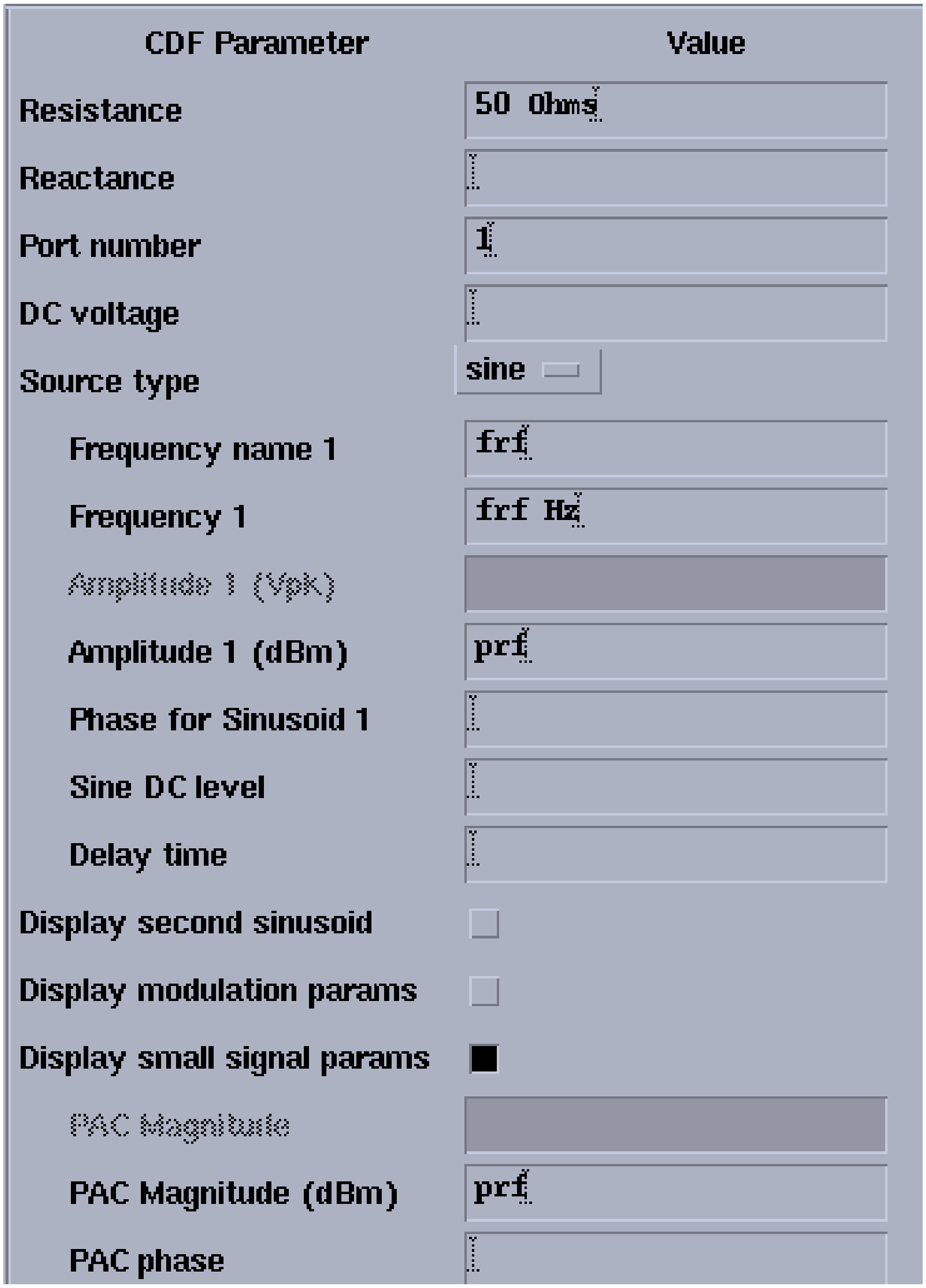
1. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears. You use this form to modify the schematic by changing the list of CDF properties.
2. To modify the schematic for this simulation, click the rf voltage source in the Schematic window.
3. Highlight *Display second sinusoid*, remove any values that are set from previous analyses, then remove the highlight.
4. Choose *sine* for *Source type*.
5. Type `frf` for *Frequency 1*.
6. Type `prf` in the *Amplitude 1 (dBm)* field.
7. Highlight *Display small signal params*.

   The form changes to let you specify small signal parameters.
8. Type `prf` for the *PAC Magnitude (dBm)* value.

   This simulation uses dBm values rather than magnitude.

The completed Edit Object Properties form appears like the one below.




**9.** Click *OK*.

**10.** In the Schematic window, choose *Design – Check and Save*.

## Setting Up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

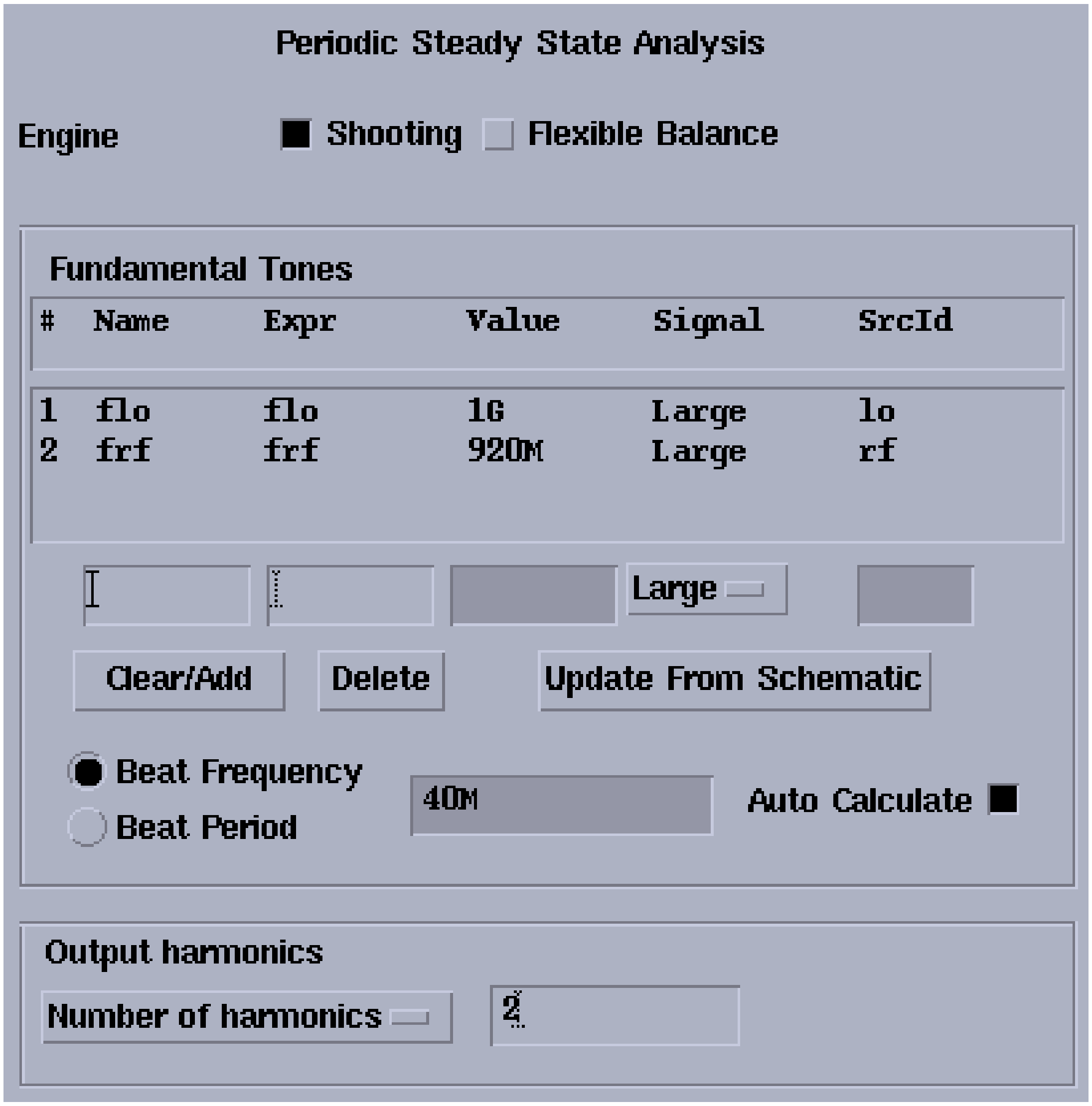
The Choosing Analyses form appears.

### Setting Up the PSS Analysis

1. In the Choosing Analyses form, choose *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The value 40M is specified as the *Beat Frequency*. The PAC analysis is responsible for the two tones, and the PSS analysis is now a single signal analysis. Consequently, the fundamental frequency is set to the original 40 MHz. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 2 in the field.

The top of the swept PSS Choosing Analyses form looks like this.



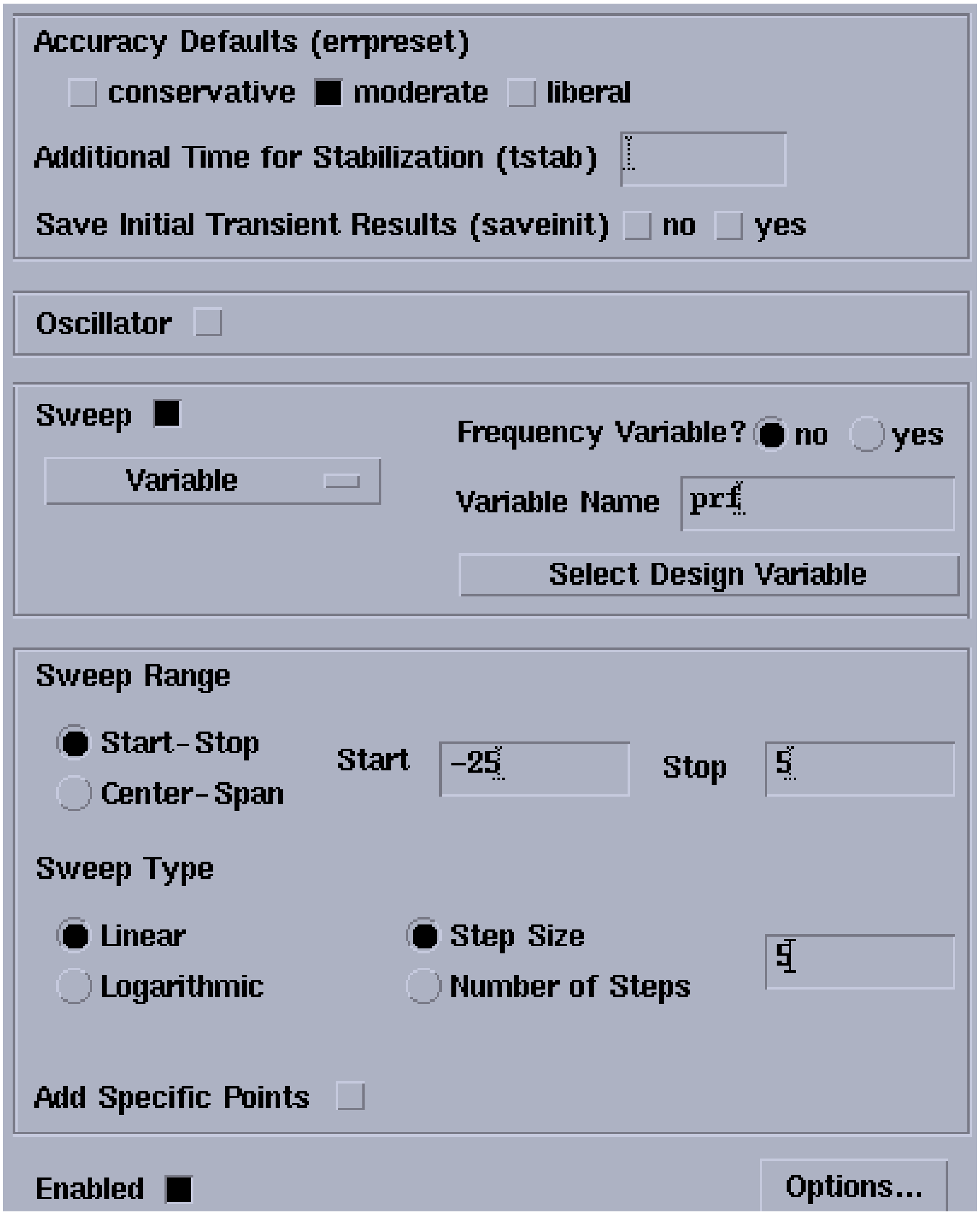

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
8. Highlight `prf` in this form and click *OK*.

9. Choose *Start-Stop* for the *Sweep Range* value, and then type -25 in the *Start field* and 5 in the *Stop* field.
10. Highlight *Linear* for the *Sweep Type*, highlight *Step Size,* and then type 5 in the *Step Size* field.
11. Highlight *Enabled*.

The bottom of the swept PSS Choosing Analyses form looks like this.




**12.** Click *Apply*.

### Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a *pac* analysis.

2. In the *Specialized Analyses* cyclic field, choose *None*.

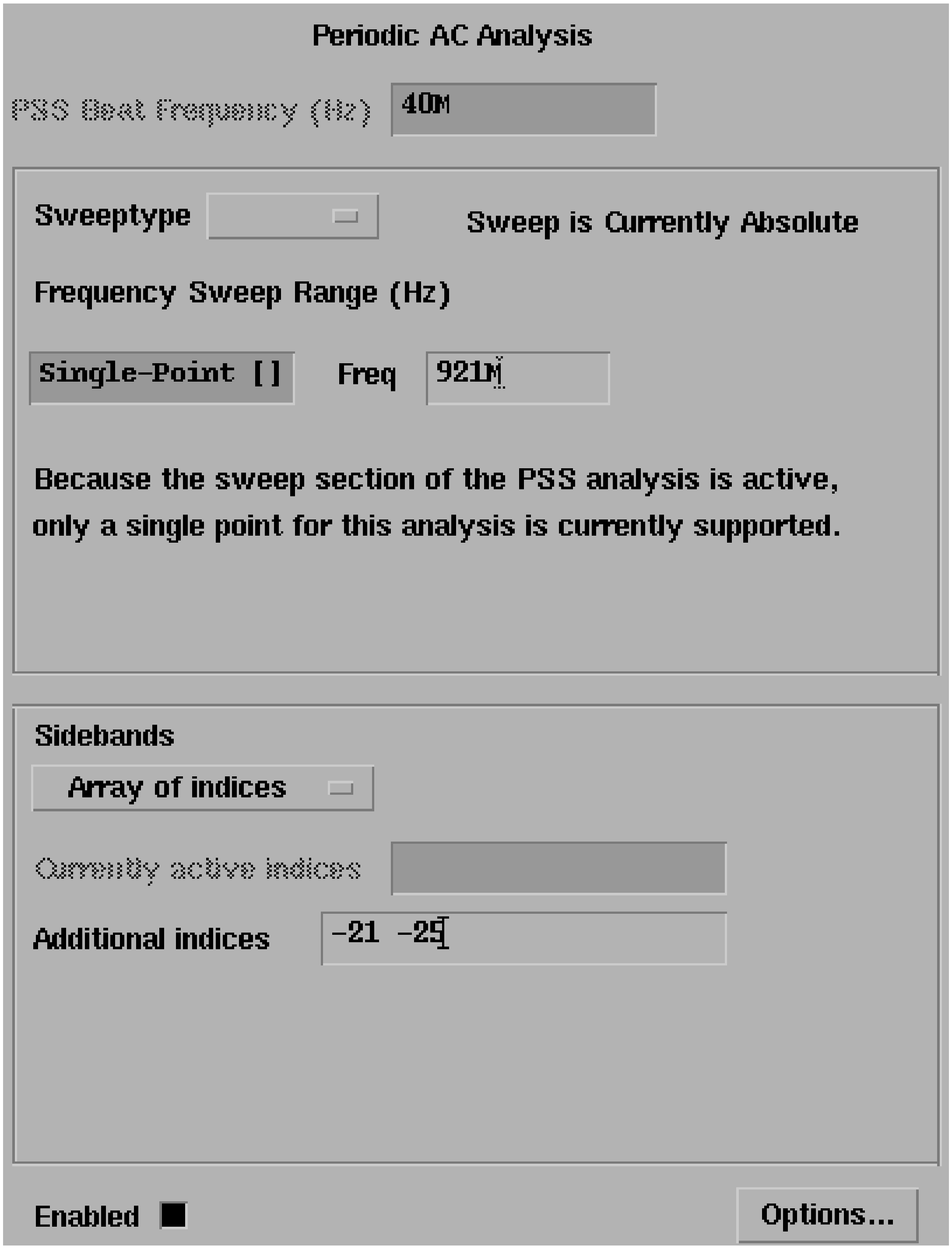
3. Type `921M` for the *Freq* value.

4. In the *Sidebands* cyclic field, choose *Array of indices* and in the *Additional indices* field type `-21` and `-25` with a space between them.

   Given a fundamental tone of 40 MHz, the LO at 1 GHz, and two RF tones at 920 MHz and 921 MHz, the sidebands of `-25` and `-21` represent respectively the first-order harmonic of the IF output at 79 MHz (921 -25*40 = 79) and the third-order harmonic at 81 MHz (921 - 21*40 = 81).

5. Highlight the *Enabled* button.

The PAC Choosing Analyses form looks like this.




6. Click *Apply*.
7. Click *OK*.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   **Note:** This example compares two signals only 1MHz apart. An analysis of two signals so close together would have taken much longer with the previous analysis.

   The output log file appears and displays information about the simulation as it runs.

2. Check the output log file to be sure the simulation completes successfully.

## Plotting the IP3 Curve

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *Replace* for *Plotting Mode*.
3. Highlight *pac* for *Analysis*.

   The form changes to display information for the PAC analysis.

4. Highlight *IPN Curves* for *Function*.

   The form changes again.

5. For *Circuit Input Power*, choose *Variable Sweep (“prf”)*.
6. Type `-15` for *Input Power Extrapolation Point (dBm)*.

   This value is the intercept point for the ideal amplification extrapolation. If you do not specify a value, the plot defaults to the minimum variable value.

7. In the cyclic field, choose *Input Referred IP3*.
8. Follow the prompt at the bottom of the form

   `Select 3rd Order Harmonic on this form...`

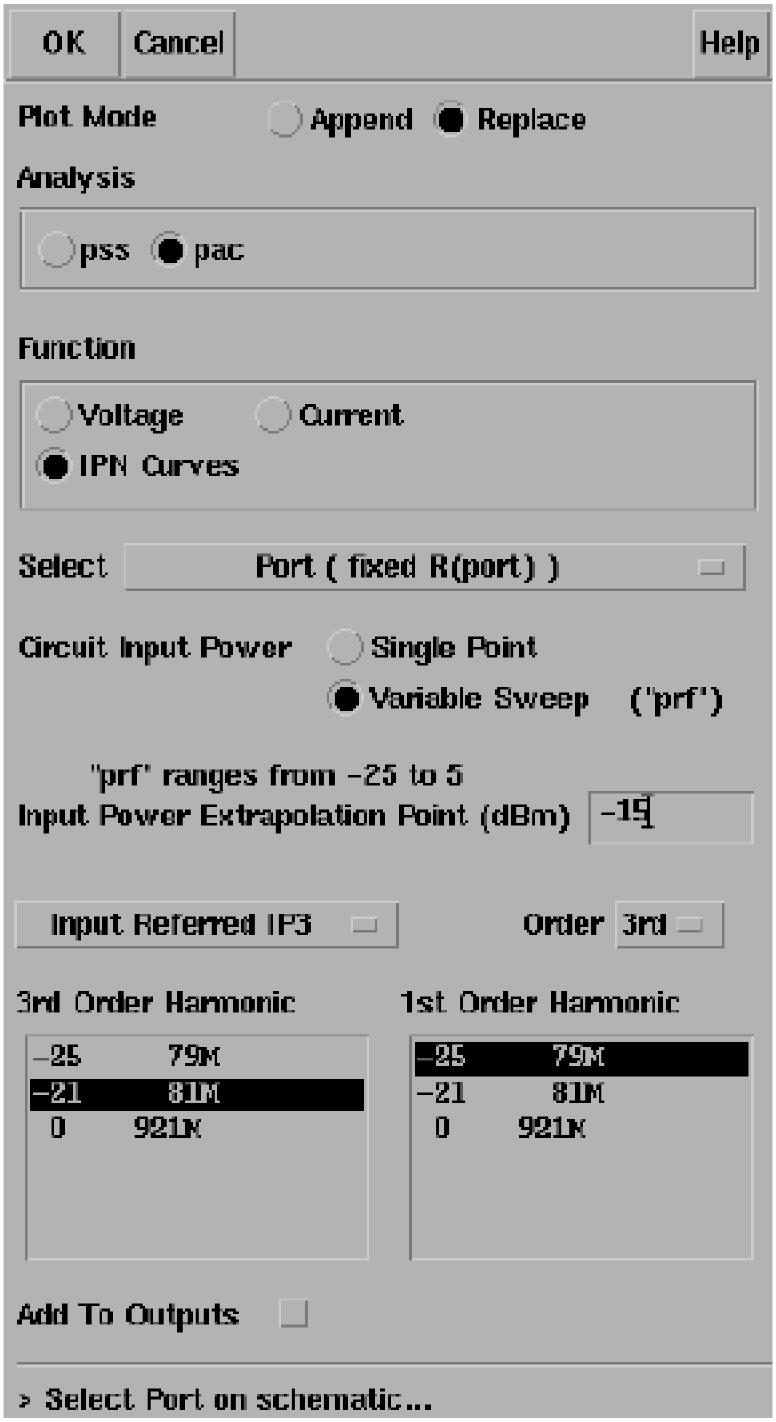
   Highlight *-21 81M* in the *3rd Order Harmonic* list box.

9. Follow the new prompt at the bottom of the form

   `Select 1st Order Harmonic on this form...`

   Highlight *-25 79M* in the *1st Order Harmonic* list box.

The completed form looks like this.



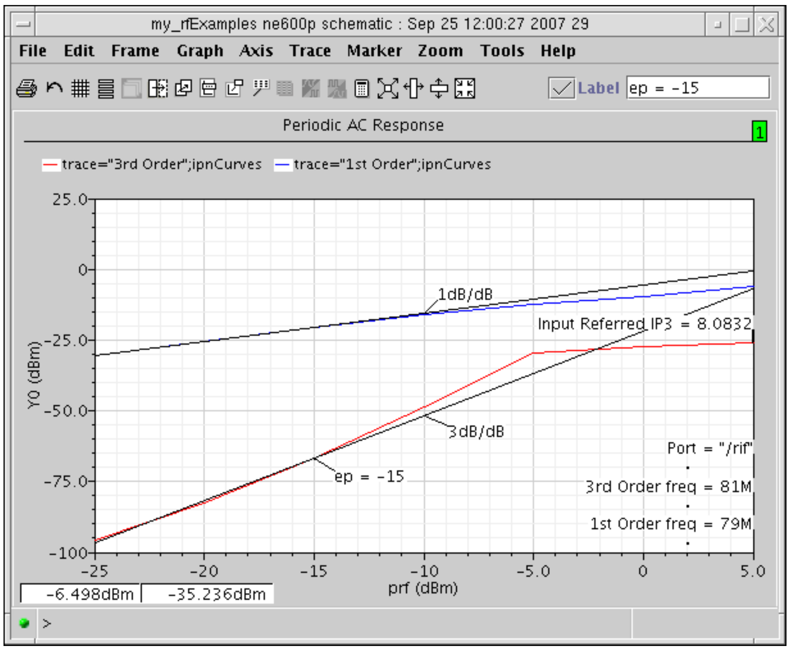

Notice that *freqaxis = absout* near the bottom of the form.

Follow the new prompt at the bottom of the form

```
Select Port on schematic...
```

In the Schematic window, click the *if* port.

The waveform window display appears as shown below. (You can click and drag to move the two labels with arrows, *ep* and *Input Referred IP3*, to improve visibility.)




The general equation used to compute the input-referred third-order intercept point is

$$IIP3 = V_{rf} + \frac{(V_{if(-25)} - V_{if(-21)})}{2}$$

where $V_{rf}$ is the value for the input power extrapolation point (dBm).

# Intermodulation Distortion Measurement with QPSS

The QPSS analysis lets you consider the effects of a few harmonics in intermodulation distortion calculations. This example describes how to run a QPSS analysis with the ne600p mixer circuit.

With the QPSS analysis, you can compute the distortion of moderately sized signals, as opposed to the small signals you investigated with the swept PSS analysis with PAC approach where you assumed there were no significant distortion effects from the small signal harmonics. With QPSS, you specify a large, or clock, signal for the fundamental and one or more moderately sized signals whose distortion effects you want to measure.

## Setting Up the Simulation

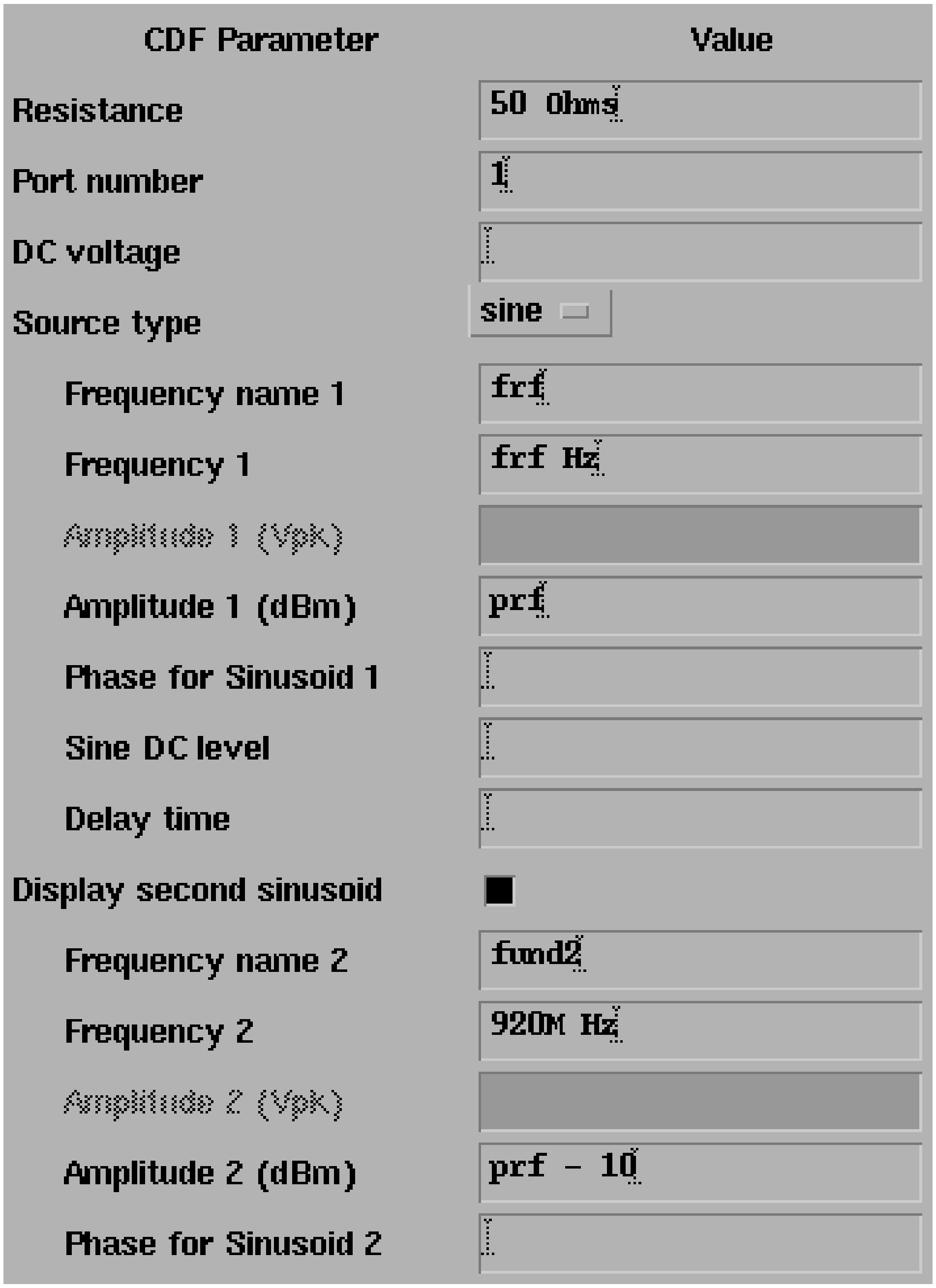
1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

## Editing the Schematic

1. In the Schematic window, choose the *rf* input port and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *rf* port appears.
2. Choose *Display small signal params* and remove any remaining values from any previous analysis.
3. Choose *sine* for *Source type*.
4. Choose *Display second sinusoid* and specify the following when the form changes to display new fields:
   a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.
   b. Type `920M` for the *Frequency 2* value.
   c. Type `prf-10` for the *Amplitude 2 (dBm)* value.

The completed form looks like this.



| CDF Parameter | Value |
| --- | --- |
| Resistance | 50 Ohms |
| Port number | 1 |
| DC voltage | |
| Source type | sine |
| Frequency name 1 | frf |
| Frequency 1 | frf Hz |
| Amplitude 1 (Vpk) | |
| Amplitude 1 (dBm) | prf |
| Phase for Sinusoid 1 | |
| Sine DC level | |
| Delay time | |
| Display second sinusoid | |
| Frequency name 2 | fund2 |
| Frequency 2 | 920M Hz |
| Amplitude 2 (Vpk) | |
| Amplitude 2 (dBm) | prf - 10 |
| Phase for Sinusoid 2 | |

5. In the Edit Object Properties form, click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

## Setting Up the QPSS Analysis

1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis. There are three tones in the *Fundamental Tones* list box, two that were present in the original specifications and one that you added.

   To update the fundamental tones, perform the following steps:

3. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

4. If necessary, choose *Moderate* from the *Signal* cyclic field.

5. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `frf` tone.

   An increase in the harmonic range for a moderate tone, such as `frf`, directly increases the simulation run time.

6. In the list box, highlight the tone named `fund2` (or whatever name you gave it).

   The new values for `frf` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

7. If necessary, choose *Moderate* from the *Signal* cyclic field.

8. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `fund2` tone.

9. In the Fundamental Tones list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

10. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and then type `1` to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

    The new values for `flo` display in the list box. the harmonic range of 1 gives 3 harmonics (-1, 0,1) for the large tone.

    By choosing *Large* as the *Signal* value, you specify `flo` to be the *Large* or clock signal. Each QPSS analysis must have one tone set to be the *Large* signal. When you choose a signal for the large signal, choose a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with an harmonic range (*Harms*) value of 0 is ignored by the simulation. In general,

- ❑ The harmonic range (*Harms)* value for the *Large* tone should generally be at least 5 which guarantees11 harmonics. In some cases, for example in a down converting situation such as this one, *Harms* can be as low as 1.
- ❑ The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

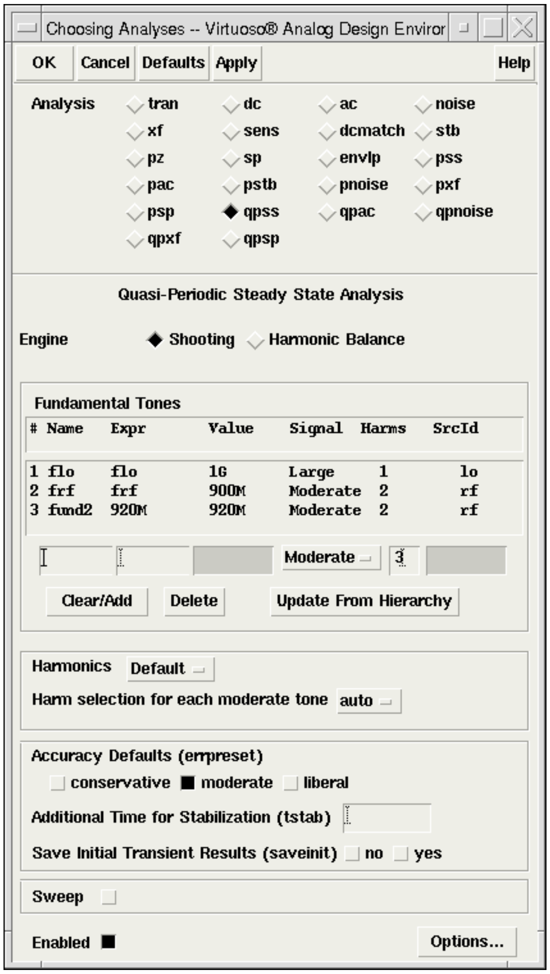
  Setting the *Harms* value to 2, yields up to the 3rd order intermodulation terms; setting the *Harms* value to 3, yields up to the 5th intermodulation terms. For higher order intermodulation terms, increase the *Harms* value accordingly. However, you should avoid setting the *Harms* value unnecessarily high because it increases the simulation time.

When, as in this example, you specify *Harms* values of 1 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [1,2,2]` which gives you 3 harmonics (-1, 0, 1) for the *Large* tone and 5 harmonics (-2, -1, 0, 1, 2) for each *Moderate* tone. As a result, you get noise from 3 x 5 x 5 = 75 frequency sidebands.

**11.** Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

**12.** Be sure the *Enabled* button is highlighted.

The completed form appears like the one below.




**13.** Click *Apply*.

**14.** Click *OK*.

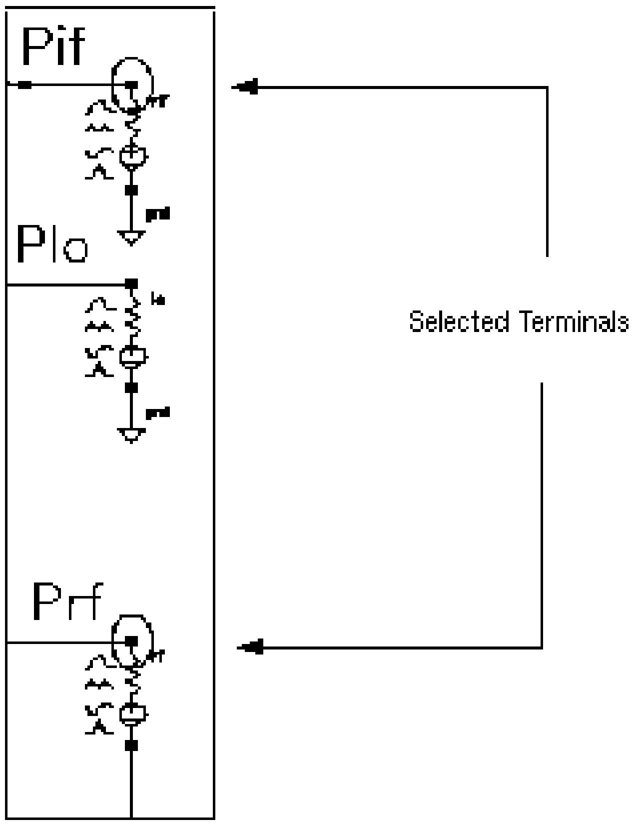
## Selecting Simulation Outputs

1. In the Virtuoso Analog Design Environment window, choose *Outputs – To Be Saved – Select on Schematic*.

   If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to choose specific currents on the schematic.

2. In the Schematic window, click the appropriate terminals to choose *rf* and *rif*.

The terminals are circled in the Schematic window after you choose them.




The selected terminals also display in the Virtuoso Analog Design Environment window.

**Outputs**

| # | Name/Signal/Expr | Value | Plot | Save | March |
|---|---|---|---|---|---|
| 1 | rf/PLUS | | no | yes | no |
| 2 | rif/PLUS | | no | yes | no |

3. Click the *Esc* key when you are finished selecting terminals.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

## Plotting the Voltage and Power

➤ In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

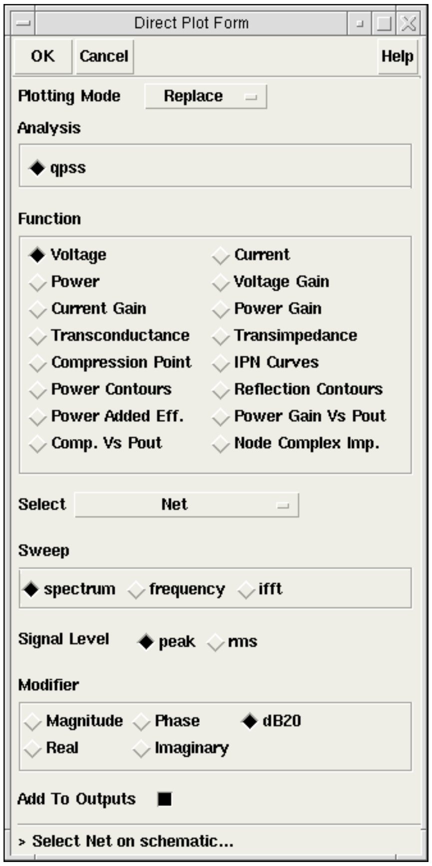
The Direct Plot form appears.

### Plotting Voltage

To plot voltage do the following in the Direct Plot form:

1. Select *Replace* for *Plotting Mode*.
2. Highlight *qpss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Ensure that the *Select* cyclic field displays *Net*.
5. Highlight *peak* for *Signal Level*.
6. Highlight *dB20* for *Modifier*.

The completed form looks like this.

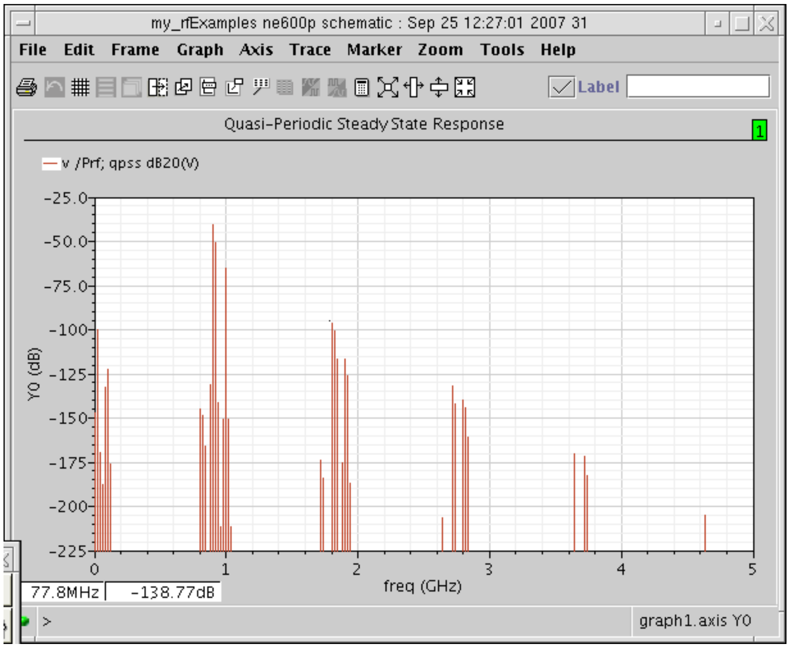



7. Follow the prompt at the bottom of the form

   `Select Net on schematic...`

   Click the net connecting to the *rf* terminal.

The voltage plot for the *rf* terminal displays in the waveform window.




Because this is a down-converting mixer, the cluster of frequencies from 20 M to 120 M near the Y axis of the plot are of interest. To find a frequency and its associated voltage, place the cursor at the tip of a line. The X and Y values that represent the frequency and the voltage, respectively, appear at the lower left corner of the window. If you place the cursor at the points shown above, you display the following voltage and frequency values:

| Difference Translation | Frequency | Voltage |
|---|---|---|
| 920M - 900M | 20M | -100.2dB20(V) |
| 1G - 920M | 80M | -132.5db20(V) |
| 1G - 900M | 100M | -122.6db20(V) |

### Plotting Power

To plot the power, do the following in the Direct Plot form:

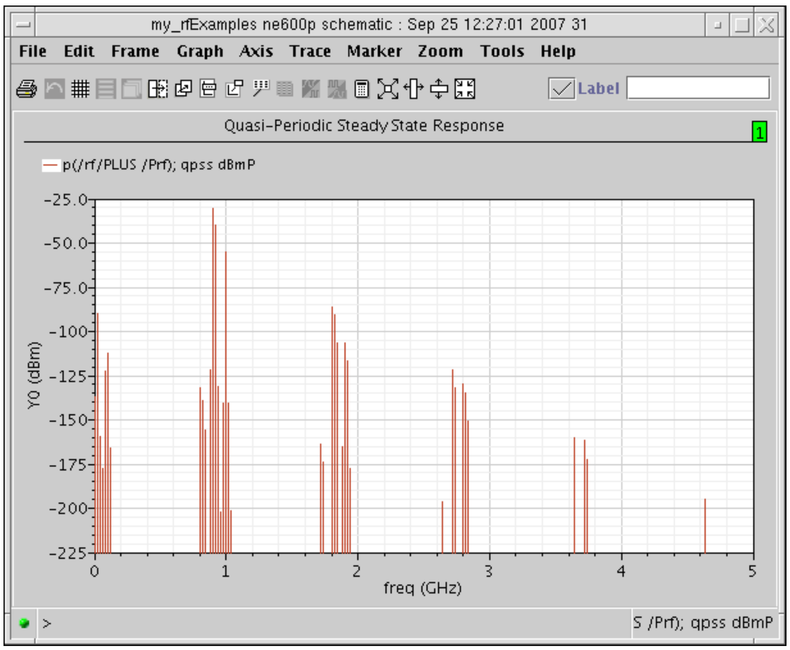
1. Select *Replace* for *Plotting Mode*.
2. Highlight *Power* for *Function*.
3. Ensure that the *Select* cyclic field displays *Terminal*.

4. Highlight *dBm* for *Modifier*.
5. Follow the message at the bottom of the form.

   `Select Instance Terminal on schematic...`

   In the Schematic window, click the *rf* terminal.

   The plot for the power appears in the waveform window.




6. If you want to see the voltage and power for the *rif* terminal, repeat the steps to create the plots for the *rif* terminal, but choose the *rif* terminal in the schematic.

   **Note:** If you want to run other simulations in this chapter, reset the design variable *frf* to `920 M`.

# Noise Figure with QPSS and QPnoise

You can combine a QPSS analysis with a small-signal QPnoise analysis to determine the degradation of the noise figure for the ne600p mixer circuit caused by the two interfering signals present at the input of the mixer. The noise figure for the ne600p mixer without the interferer signals is determined in "Noise Figure Measurement with PSS and Pnoise" on page 274. The QPSS and QPnoise analyses described in this example include the interferer signals which allow more noise foldings to occur into the output IF band.

The Quasi-Periodic Noise, or QPnoise analysis, is similar to the conventional noise analysis, except that it includes frequency conversion and intermodulation effects. Hence is it useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits.

## Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

## Editing the Schematic

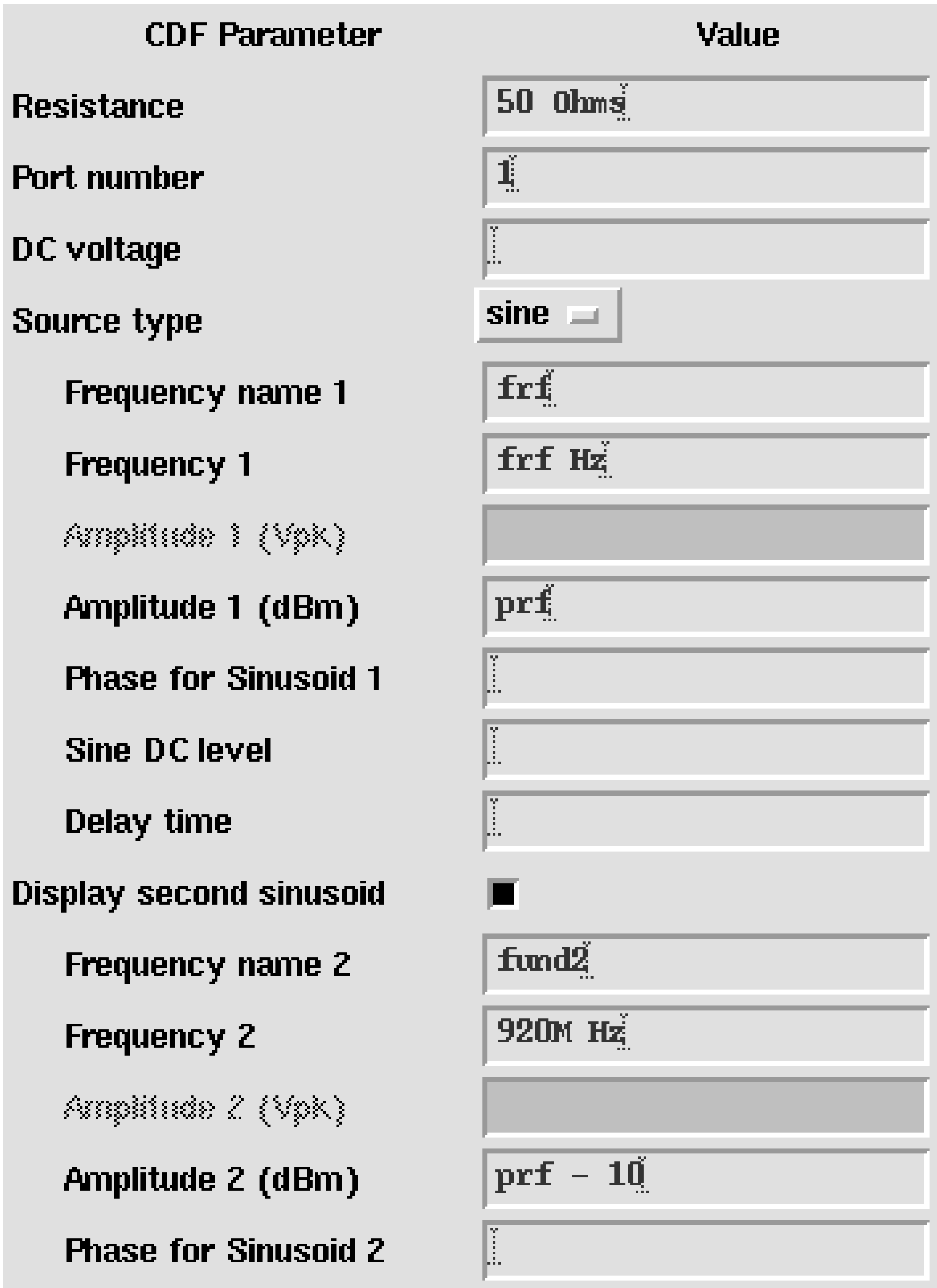
1. In the Schematic window, click the *rf* input port and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *rf* input port appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

In the Edit Object Properties form, do the following:

1. Choose *Display small signal params* and remove any remaining values from the previous analysis.
2. Because the `rf` signal is a moderate signal, choose *sine* for *Source type*.
3. Choose *Display second sinusoid* and specify the following when the form changes to display new fields:
   a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.
   b. Type `920M` for the *Frequency 2* value.
   c. Type `prf - 10` for the *Amplitude 2 (dBm)* value.

The completed form looks like this.



| CDF Parameter | Value |
|---|---|
| Resistance | 50 Ohms |
| Port number | 1 |
| DC voltage | |
| Source type | sine |
| Frequency name 1 | frf |
| Frequency 1 | frf Hz |
| Amplitude 1 (Vpk) | |
| Amplitude 1 (dBm) | prf |
| Phase for Sinusoid 1 | |
| Sine DC level | |
| Delay time | |
| Display second sinusoid | ■ |
| Frequency name 2 | fund2 |
| Frequency 2 | 920M Hz |
| Amplitude 2 (Vpk) | |
| Amplitude 2 (dBm) | prf - 10 |
| Phase for Sinusoid 2 | |

**4.** In the Edit Object Properties form, click *OK*.

**5.** In the Schematic window, choose *Design – Check and Save*.

## Setting Up the QPSS and QPnoise Analyses

In the following sections you set up two analyses, first *qpss* then *qpnoise*.

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

### Setting Up the QPSS Analysis

1. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis.

2. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

3. If necessary, choose *Moderate* from the *Signal* cyclic field.

4. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `frf` tone.

   An increase in the harmonic range for a moderate tone directly increases the simulation run time.

5. In the list box, highlight the tone named `fund2`.

   The new values for `fund2` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

6. If necessary, choose *Moderate* from the *Signal* cyclic field.

7. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `fund2` tone.

8. In the list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

9. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and type 5 to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

   The new values for `flo` display in the list box. The harmonic range of 5 gives 11 harmonics (-5, ... ,0, ... +5) for the large tone. In this case where a QPnoise analysis is following the QPSS analysis, the *Harms* value must be sufficiently large to guarantee an accurate noise analysis. Generally, 5 is a sufficiently large *Harms* value. Entering a large *Harms* value for the large tone does not affect the simulation run time.

By choosing *Large* as the *Signal* value, you specify `flo` to be the large or clock signal. Each QPSS analysis must have one tone set to be the large signal. When you choose a large signal, select a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with a harmonic range value of 0 is ignored by the simulation. In general,

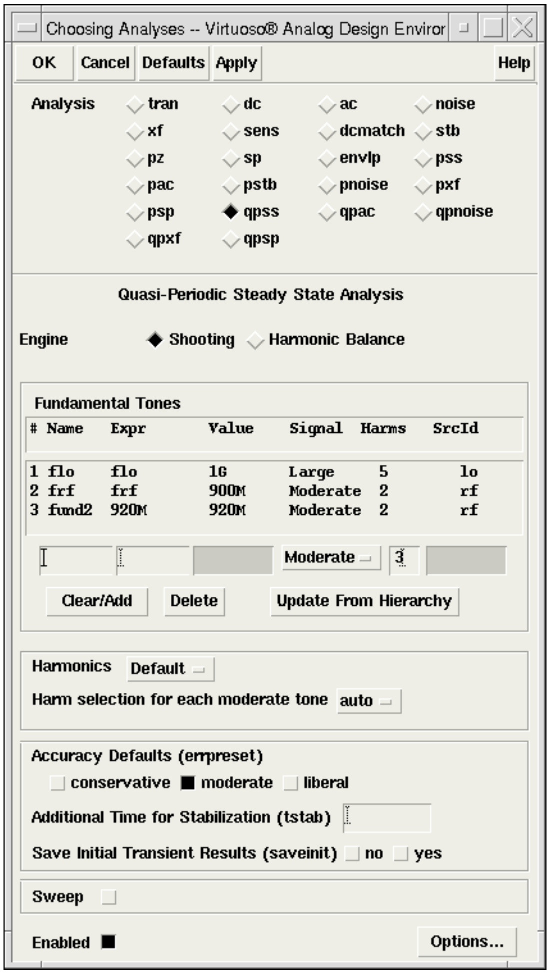
- The harmonic range (*Harms)* value for the *Large* tone should be at least 5, guaranteeing 11 harmonics.
- The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

  When, as in this example, you specify *Harms* values of 5 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [5,2]` which gives you 11 harmonics for the Large tone and 5 harmonics (-2, -1, 0, 1, 2) for the Moderate tone. As a result, you get noise from 11 x 5 = 55 frequency sidebands. Spectre RF considers all these combinations when performing QPnoise calculations.

**10.** Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

**11.** Be sure the *Enabled* button is highlighted.

The completed form appears like the one below.




**12.** Click *Apply*.

## Setting Up the QPnoise Analysis

1. In the Analysis section at the top of the Choosing Analyses form, highlight *qpnoise*.

   The Choosing Analyses form changes to let you specify data for a Quasi-Periodic Noise Analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*.

3. Type `50M` in the *Start* field and `150M` in the *Stop* field.

   This frequency range covers the frequencies of interest.

4. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `20` in the *Number of Steps* field.
5. In the *Sidebands* cyclic field, choose *Maximum clock order* and type `5`.

   Maximum clock order is related to large signal harmonics range. A *Maximum clock order* of 5 insures that sufficient large-signal harmonics are processed. You can increase this value.
6. In the *Output* cyclic field, choose *probe* for the *Output* value.
7. Click the *Select* button for *Output Probe Instance*. Then click the appropriate component in the Schematic window to choose *Pif*.

   `/rif` appears in the *Output Probe Instance* field.
8. In the *Input Source* cyclic field, choose *port*.
9. Click the *Select* button for *Input Port Source*. Then click the appropriate component in the Schematic window to choose *rf*.

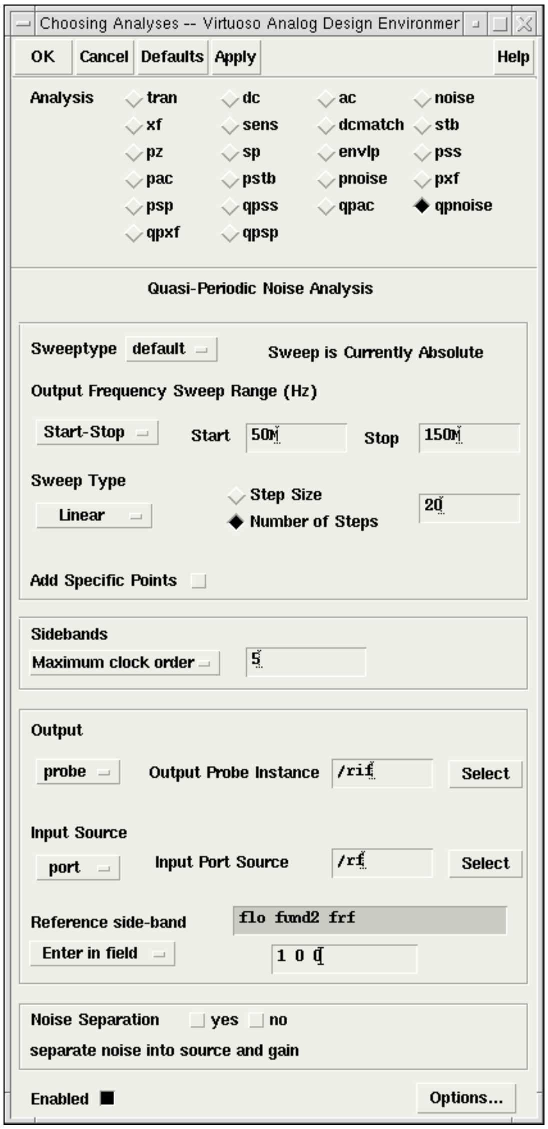
   `/rf` appears in the *Input Port Source* field.
10. Type `1 0 0` in the *Reference side-band* field. One entry for each tone separated by spaces.

    These numbers define a vector whose indices have a one-to-one correspondence with the fundamental tones. For this example, the correspondence is illustrated in the following table.

| | | | |
|---|---|---|---|
| **Ref Sideband Vector** | 1 | 0 | 0 |
| **Fundamental Tones** | LO | FUND2 | RF |
| **Frequencies** | 1 GHz | 920 MHz | 900 MHz |

    The reference sideband vector `1 0 0` mixes with the LO tone only to get the output.

The completed *Quasi Periodic Noise Analysis* section of the Choosing Analyses form appears like the one below.




**11.** In the Choosing Analyses form, click *OK*.

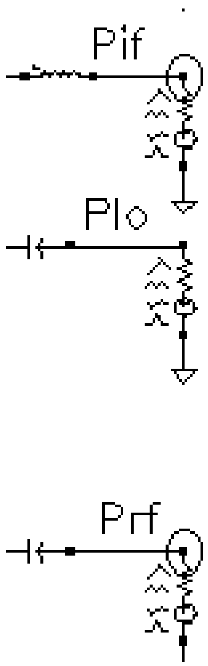
## Selecting Simulation Outputs

**1.** In the Virtuoso Analog Design Environment window, choose *Outputs – To Be Saved – Select on Schematic*.

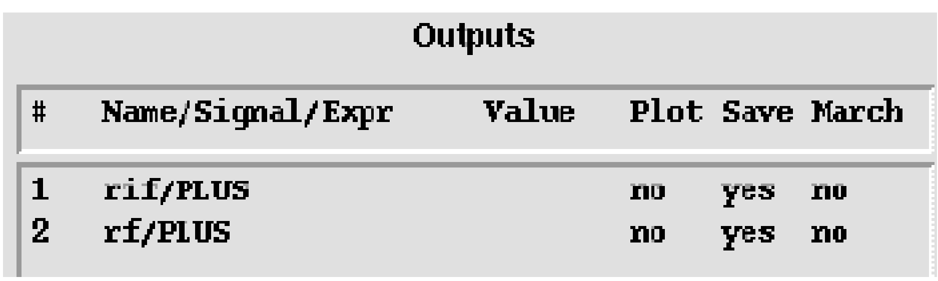
If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to select the specific currents to save on the schematic.

**2.** In the Schematic window, click the appropriate terminals to choose *rf* and *rif*.

The terminals are circled in the Schematic window after you choose them.




The selected terminals also appear in the Virtuoso Analog Design Environment window.



**Outputs**

| # | Name/Signal/Expr | Value | Plot | Save | March |
|---|---|---|---|---|---|
| 1 | rif/PLUS | | no | yes | no |
| 2 | rf/PLUS | | no | yes | no |

**3.** Click the *Esc* key when you are done selecting terminals.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

## Plotting the Noise Figure

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

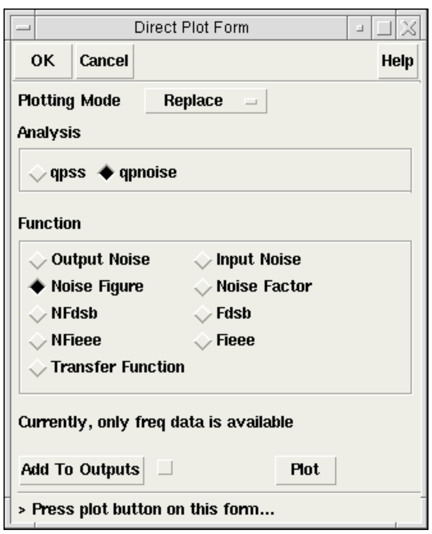
   The Direct Plot form window appears.

2. Highlight *Replace* for *Plot Mode*.

3. Highlight *qpnoise* for *Analysis*.

4. Highlight *Noise Figure* for *Function*.

   The completed Direct Plot form for Qpnoise appears like the one below.



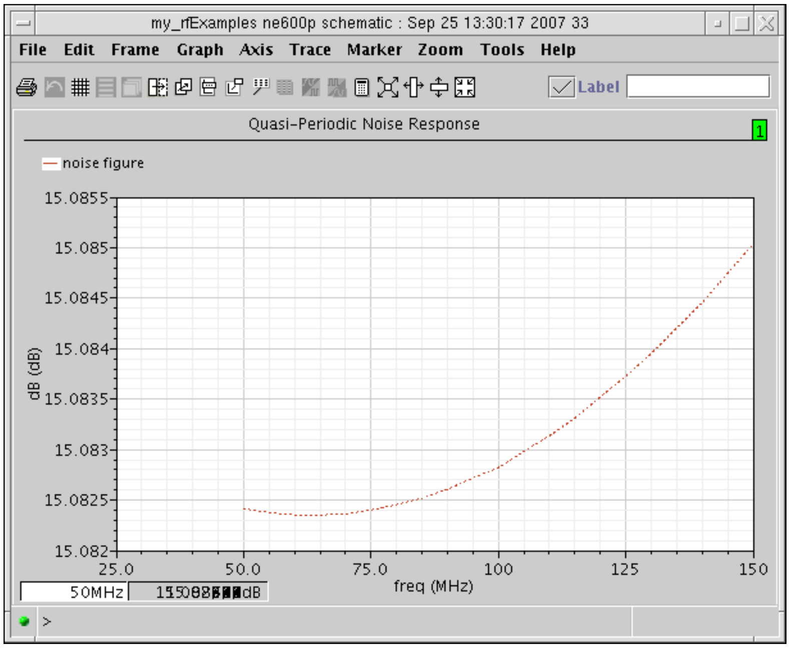

5. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click *Plot* in the Direct Plot form.

   The plot appears in the waveform window.

The waveform window appears like the one below:




**6.** To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the waveform window. In the above plot, the noise figure is about `73.24dB` at `80 MHz`.

## Plotting the Output Noise

In the Direct Plot form, do the following:

**1.** Highlight *Replace* for *Plot Mode*.

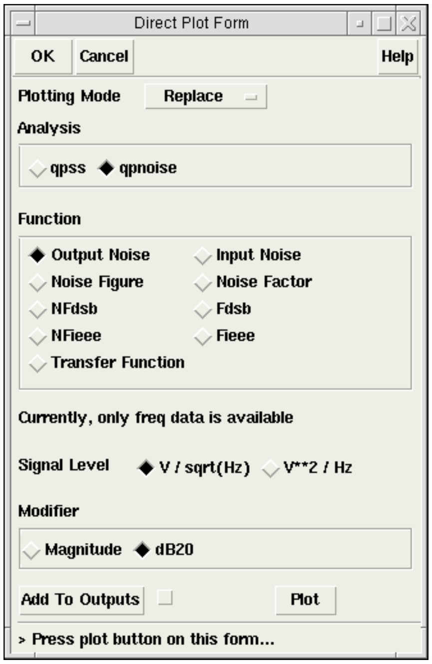
**2.** Highlight *qpnoise* for *Analysis*.

**3.** Highlight *Output Noise* for *Function*.

**4.** Highlight V/sqrt(Hz) for Signal Level.

**5.** Highlight *dB20* for *Modifier*.

The completed Direct Plot form appears like the one below.



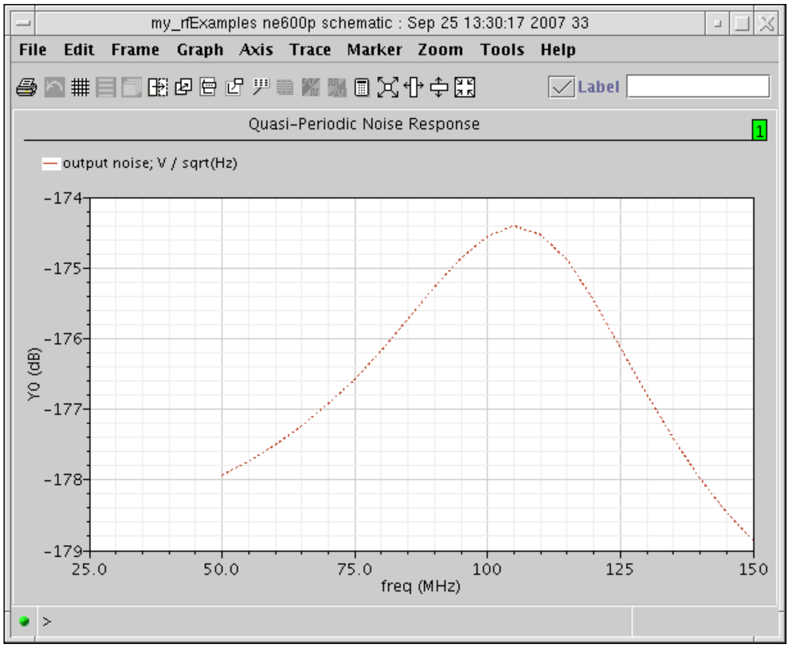

**6.** Follow the prompt at the bottom of the form

`Press plot button on this form...`

Click *Plot* in the Direct Plot form.

The plot appears in the waveform window.

The waveform window appears like the one below:




7. To determine the output noise at different frequencies, move the cursor along the output noise curve in the waveform window. In the above plot, the output noise is about `-119.2 dB` at `80 MHz`.

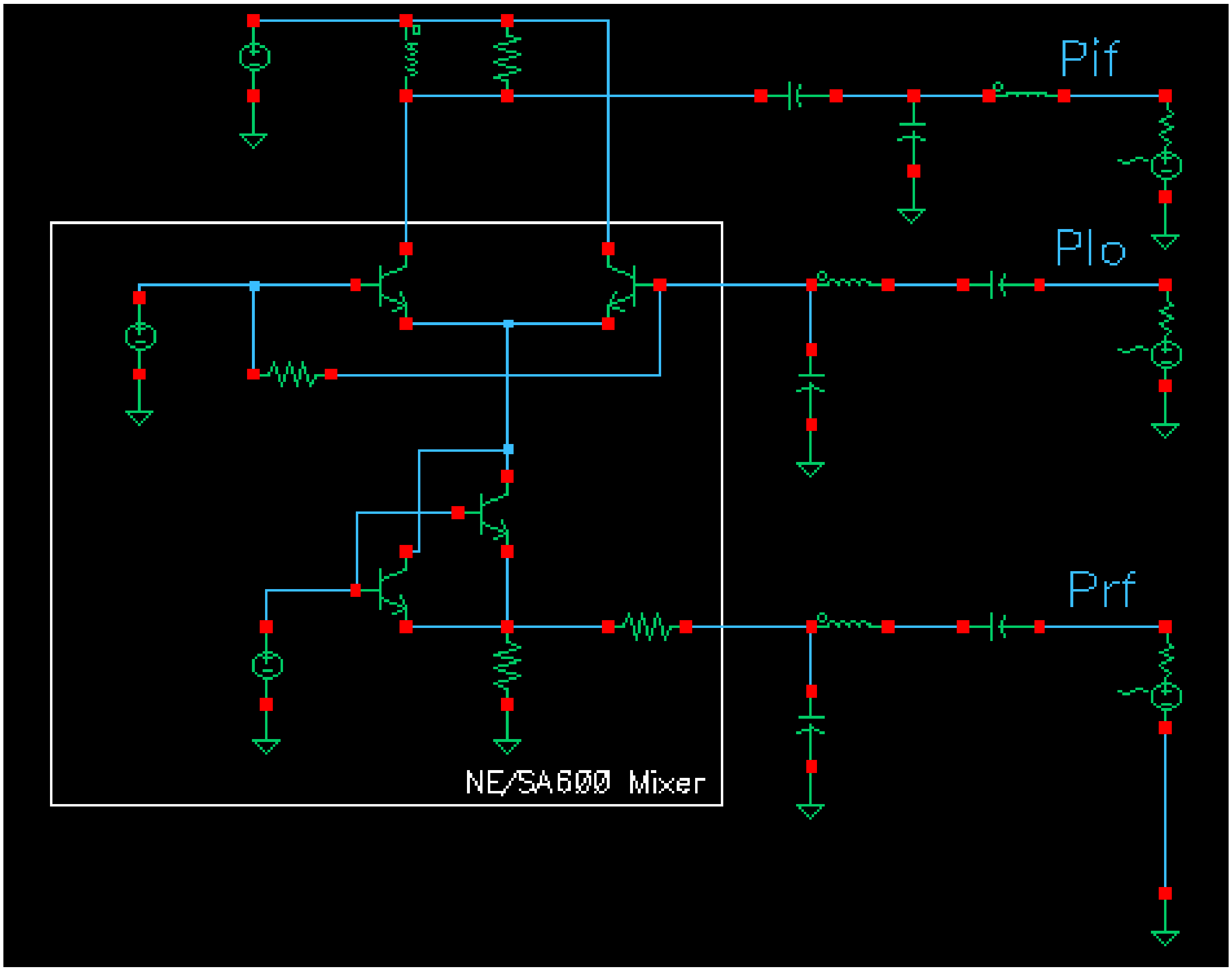
**Note:** Before you run most other simulation examples in this chapter, be sure to reset the *Design Variable* `frf` to `920M`.

# IP3 and Compression Distortion Summary Measurements for a Mixer

Mixer distortion limits the sensitivity of a receiver if there is a large interfering signal present within the bandwidth of the RF input filter (a characteristic known as *selectivity*).

Consider measuring Rapid IP3 and Compression Distortion Summary for the *NE600* using perturbation-based PAC analysis. The *NE600* mixer schematic from the *rfExamples* library is shown in Figure 5-2.

**Figure 5-2 The NE600 Mixer Schematic**




When choosing the RF input signals *frf1* and *frf2*, you can perform a PAC analysis to determine the bandwidth of the circuit. Then you can place the RF input signals in the middle of the bandwidth. The RF input signals *frf1* and *frf2* should be close enough in frequency so that their intermodulation terms are also within the bandwidth.

Suppose the RF input signal is at *frf1* = 900 MHz, the LO signal is at 1 GHz, and the IFsignal is at 100 MHz. Another RF signal is at *frf2* = 925 M, so one of the $3^{rd}$ intermodulation products is at 2*frf1* - *frf2* = 875 M and the corresponding IF signal is at 125 M. Before you perform the Rapid IP3 measurement, you must run a PSS analysis. For small-signal analysis, it is sufficient to treat the RF input as a small-signal (for example, by setting the *Source Type* to *dc*).

## Measuring Rapid IP3

The Rapid IP3 mixer measurement is calculated using Spectre RF in the Analog Design Environment (ADE).

**Stimuli:**

| | |
|---|---|
| Apply a large sinusoidal signal at the LO port (*PORT2)* | *flo* |
| Apply a moderate *dc* source at the RF port (*PORT1*) | *frf1* |

**Parameters:**

Set the parameters

| | |
|---|---|
| *flo* | `1G` |
| *prf* | `-30` |
| *frf1* | `900M` |
| *frf2* | `925M` |

**Simulation/Analyses:**

1. Set up a PSS analysis.

   **Beat frequency:** *flo*

   **Output harmonics:** *number of harmonics 30*.

   **Note:** There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in the PSS analysis affect the accuracy of the IP3 result.

2. Set up a PAC analysis.

   **Specialized Analyses:** *Rapid IP3*

   **Source Type:** *port*

   **Input Sources 1:** */PORT1* **freq:** *frf1*

   **Input Sources 2:** */PORT1* **freq:** *frf2*

***Input Power (dBm):*** *prf*

**Frequency of IM Output Signal:** *flo-(2*frf1-frf2)*

**Frequency of Linear Output Signal:** *flo-frf1*

**Maximum Non-Linear Harmonics:** *4*

**Output:** *Voltage*

**Out+:** */pif*

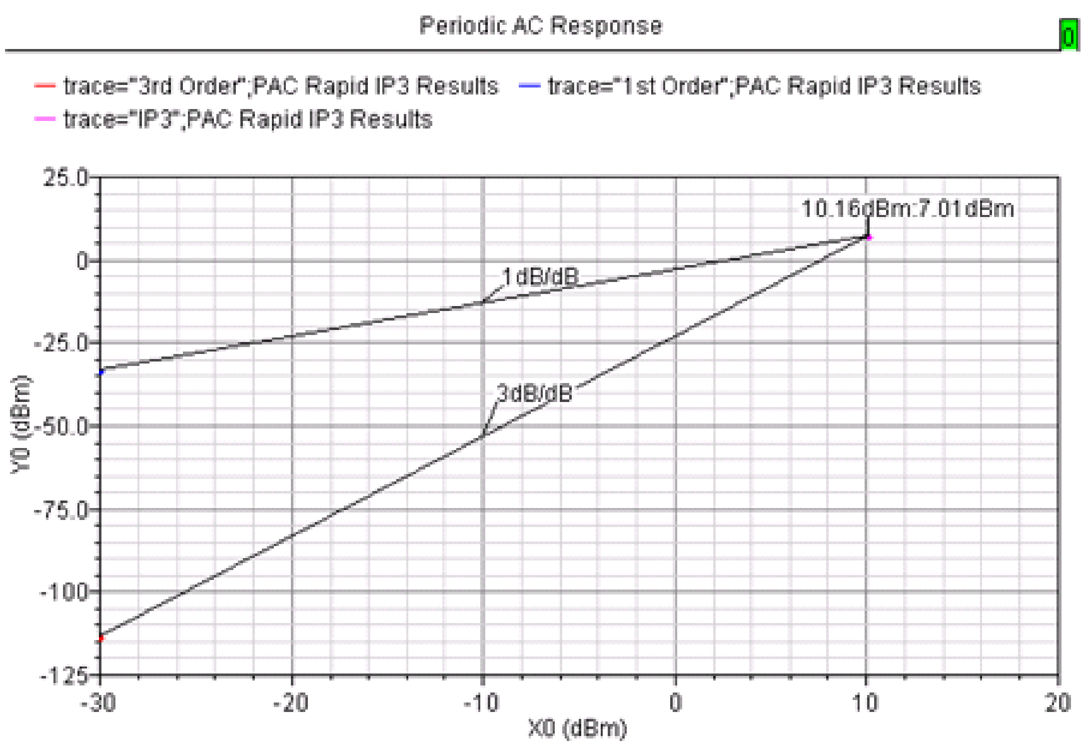
**Out-:** /gnd!.

**Note:** If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

3. Run the PSS and the PAC simulations:
4. Display the results with the Direct Plot form.

   To plot Rapid IP3, click *PAC* analysis. Select *Rapid IP3*. IP3 is plotted as in Figure 5-3.

**Figure 5-3 Rapid IP3 Measurement**



## Measuring Compression Distortion Summary

The Compression Distortion Summary mixer measurement is calculated using SpectreRF in the Analog Design Environment (ADE).

#### Stimuli:

Apply a large sinusoidal signal at the LO port (*PORT2)*. Use the RF port (*PORT1*) for a moderate DC source, set the parameter for *PAC magnitude (dBm)* in the *Small Signal Parameters to -30 dBm*. This value will be used as the input power in the following small signal analysis.

#### Parameters:

Set the parameters

| | |
|---|---|
| *flo* | `1G` |
| *frf* | `900M` |

#### Simulation/Analyses:

1. Set up a PSS analysis.

   **Beat frequency:** *flo*

   **Output harmonics:** *number of harmonics 30*.

   **Note:** There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the Compression Distortion Summary result.

2. Set up a PAC analysis.

   **Start:** *frf*

   **Specialized Analyses:** *Compression Distortion Summary*

   **Contributor Instances:**

   **Frequency of Linear Output Signal:** *flo-frf*

   **Maximum Non-Linear Harmonics:** *4*

   **Output:** *Voltage*

**Out+:** */pif*

**Out-:** */gnd!*

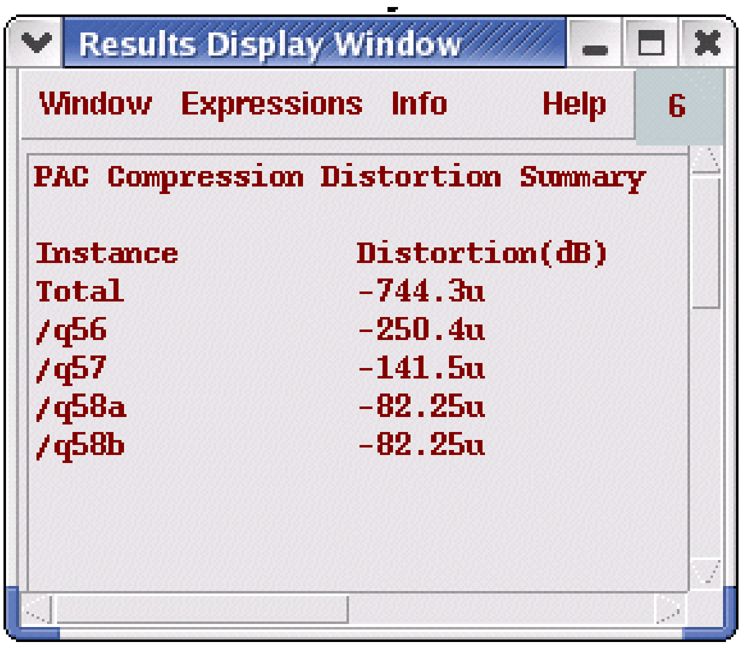
**Note:** If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

When no devices are selected, all the nonlinear devices will be calculated. Because PAC needs to be run for each device specified, computation could be time-consuming if the device list is large. You should select only the most important devices to the output. We do not recommended running the distortion summary for all nonlinear devices in the circuit.

3. Run the PSS and the PAC simulations.
4. Display/Data Analysis.

   You can find the information in the *PAC Distortion Summary* which you can access by choosing *Results – Print.* The result is shown in Figure 5-4.

**Figure 5-4 PAC Compression Distortion Summary**

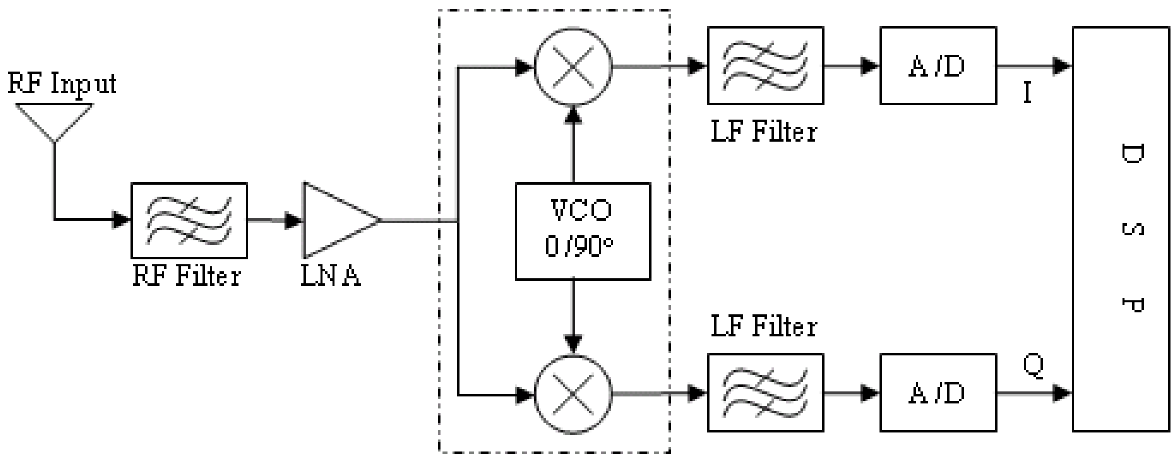



This summary lists the various distortion contributors and how much distortion each contributes to the output.

If no devices are selected, all the nonlinear devices will be calculated. You can quickly identify top distortion contributors, and you can look up the distortion contribution for a particular device.

# Rapid IP2 and IM2 Distortion Measurements for a Gilbert Direct Conversion Mixer

**Figure 5-5 Scheme of a Direct Conversion Receiver**



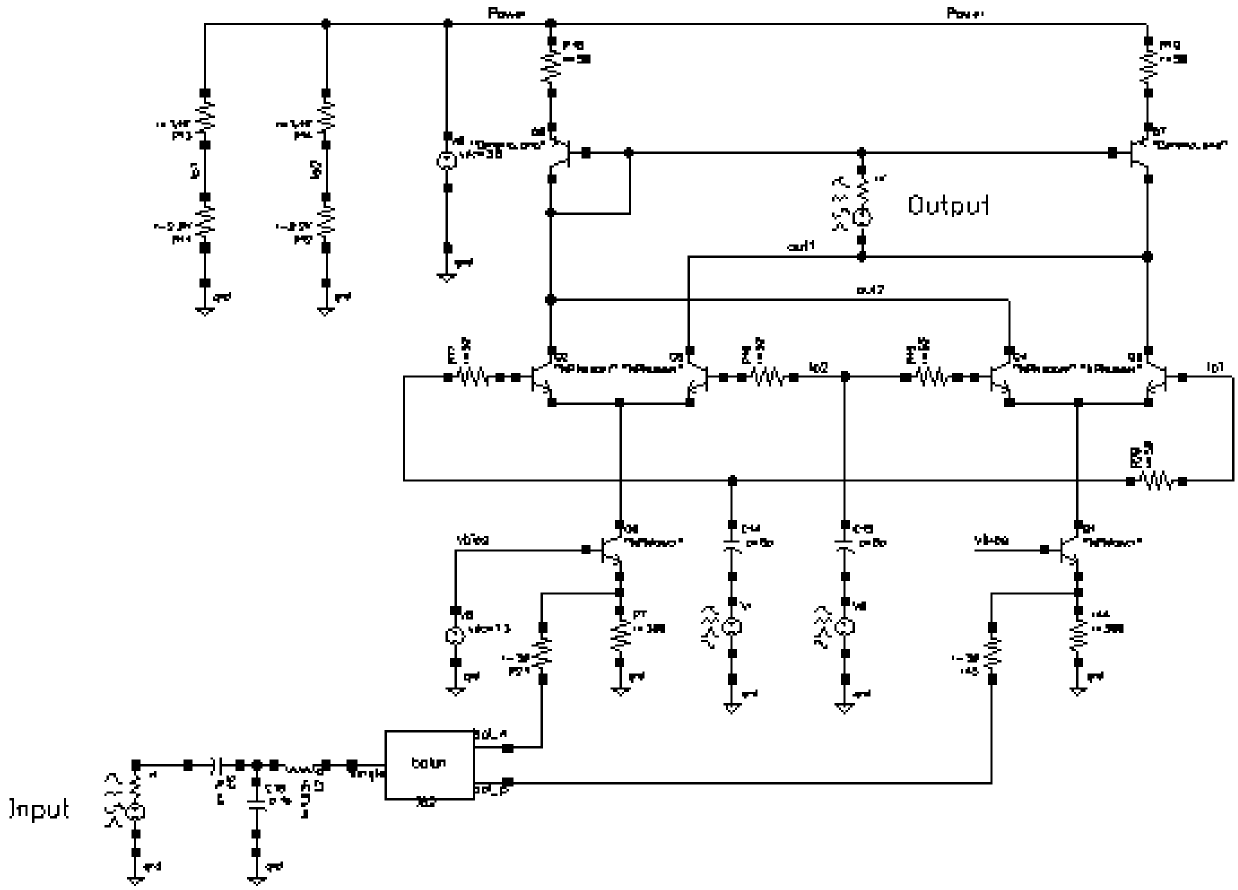

Direct conversion is an alternative architecture to the highly integrated and low-power super-heterodyne receiver. Its fundamental advantage is that the received signal is amplified and filtered at baseband rather than at some high intermediate frequency. A typical front-end frame for a direct conversion receiver is shown in Figure 5-5.

The direct conversion mixer (outlined by the dashed rectangle in Figure 5-5) is the key component in this receiver. If the local oscillator frequency (LO) equals to (zero IF) or approaches (low IF) the carrier frequency (RF), the mixer translates signals from the RF band to baseband, generally resulting in two mutually exclusive groups of signals.

In an ideal situation, the mixer output would be an exact replica of the input signal. In reality, mixer output is distorted due to non-linearity in the mixer, caused by BJT, MOS etc. Linearity is important to the receiver's performance. The most important nonlinearities of a direct conversion mixer are IP2 and IM2, which you can efficiently measure using Spectre RF perturbation technology.

The mixer measurements described here are calculated using Spectre RF in ADE. The design investigated is a Gilbert double-balanced mixer shown in Figure 5-6.

### Figure 5-6 Gilbert Double-Balanced Mixer




The core of the mixer is the Gilbert cell, constructed from NPN Q2, Q3, Q4 and Q5. The NPN Q0 and Q1 are connected as common-base, and act to import RF signals.

The input RF signal enters a Balun transmission line block (implemented by AHDL). Two output difference signals feed into the emitters of Q0 and Q1 respectively.

The PNP Q6 and Q7 act as current mirrors supplying the Gilbert cell.

The circuit runs with a local oscillator at *flo=1.9 G*. Input RF signal at *frf1=1.9012 G* and the corresponding IF at *frf1-flo=1.2 M*. Another RF signal is at *frf2=1.901 G*. One of the IM2 is at *frf1-frf2=200 K*.

## Measuring Rapid IP2

The rapid IP2 mixer measurement is described in the following sections. It is calculated using Spectre RF in ADE.

### Stimuli:

The circuit has two ports providing large sinusoidal LO signals. They have same frequency *flo=1.9G* and amplitude *ampl=316.2m* with different phase. One is *0*° and the other is *180*°.

### Parameters:

Set the parameters

| | |
|---|---|
| *flo* | `1.9G` |
| *frf1* | `1.9012G` |
| *frf2* | `1.901G` |
| *prf* | `-30` |

### Simulation/Analyses:

1. Set up a PSS analysis.

   **Beat frequency:** *flo*

   **Output harmonics:** *number of harmonics 20*.

   **Note:** There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the small signal analyses.

2. Set up a PAC analysis.

   **Specialized Analyses:** *Rapid IP2*

   **Source Type:** *port*

   **Input Sources 1:** */rf* **freq:** *frf1*

   **Input Sources 2:** */rf* **freq:** *frf2*

**Input Power (dBm):** *prf*

**Frequency of IM Output Signal:** *frf1-frf2*

**Frequency of Linear Output Signal:** *frf1-flo*

**Maximum Non-Linear Harmonics:** *4*

**Output:** *Voltage*

**Out+:** *out2*

**Out-:** *out1*.

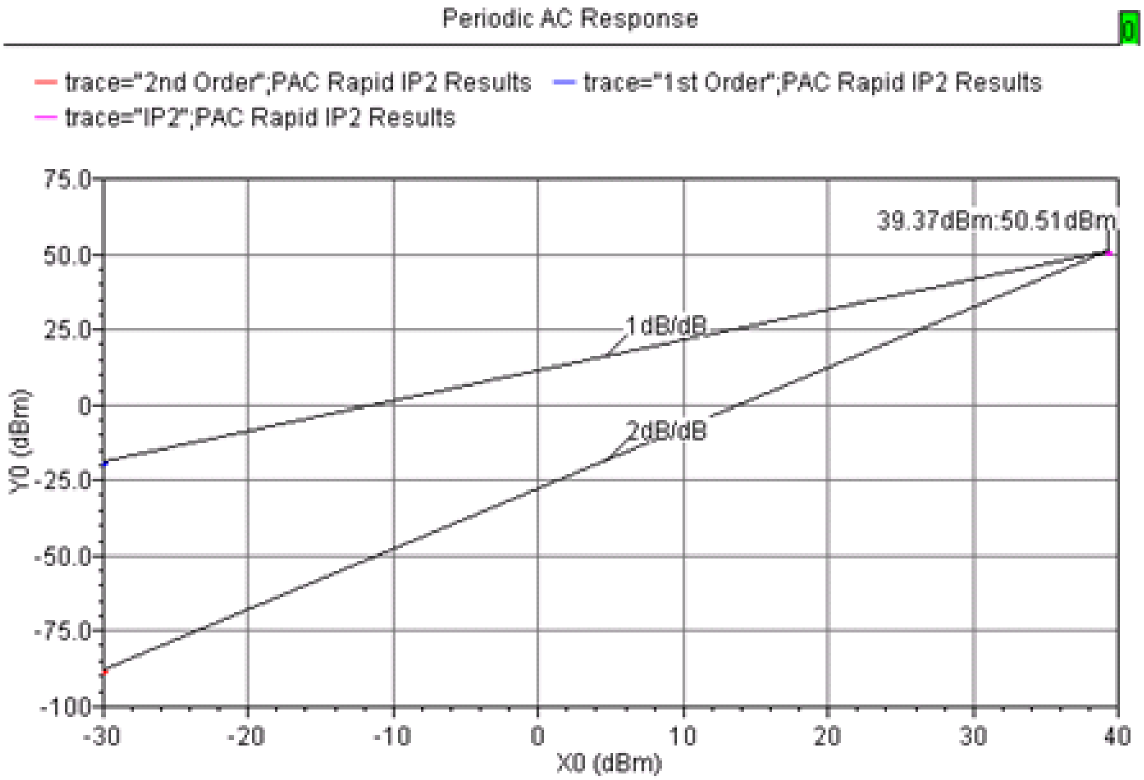
**Note:** If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre does not calculate it.

3. Run the PSS and PAC simulations.
4. Display/Data Analyses.

   Display results using *Results-Direct Plot*.

   To plot Rapid IP2, click *PAC* analysis. Select *Rapid IP2*. The IP2 for the mixer is plotted in Figure 5-7.

**Figure 5-7  PAC Rapid IP2 Plot for the Mixer**




## Measuring IM2 Distortion Summary

The IM2 distortion summary mixer measurement is described in the following section. It is calculated using Spectre RF in ADE.

### Stimuli:

The circuit has two ports providing large sinusoidal LO signals. They have the same frequency *flo=1.9G* and amplitude *ampl=316.2m* with different phase. One is *0*° and the other is *180*°.

**Parameters:**

Set the parameters

| | |
|---|---|
| *flo* | `1.9G` |
| *vlo* | `316.2m` |
| *frf1* | `1.901G` |
| *frf2* | *frf1*`+200K` |
| *prf* | `-30` |

**Simulation/Analyses:**

1. Set up a PSS analysis.

   **Beat frequency:** *flo*

   **Output harmonics:** *number of harmonics 20*.

   **Note:** There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the small signal analyses.

2. Set up a PAC analysis.

   **Specialized Analyses:** *IM2 Distortion Summary*

   **Source Type:** *port*

   **Input Sources 1:** */rf* **freq:** *frf1*

   **Input Sources 2:** */rf* **freq:** *frf2*

   **Input Power (dBm):** *prf*

   **Frequency of IM Output Signal:** *frf1-frf2*

   **Maximum Non-Linear Harmonics:** *4*

   **Output:** *Voltage*

   **Out+:** *out2*

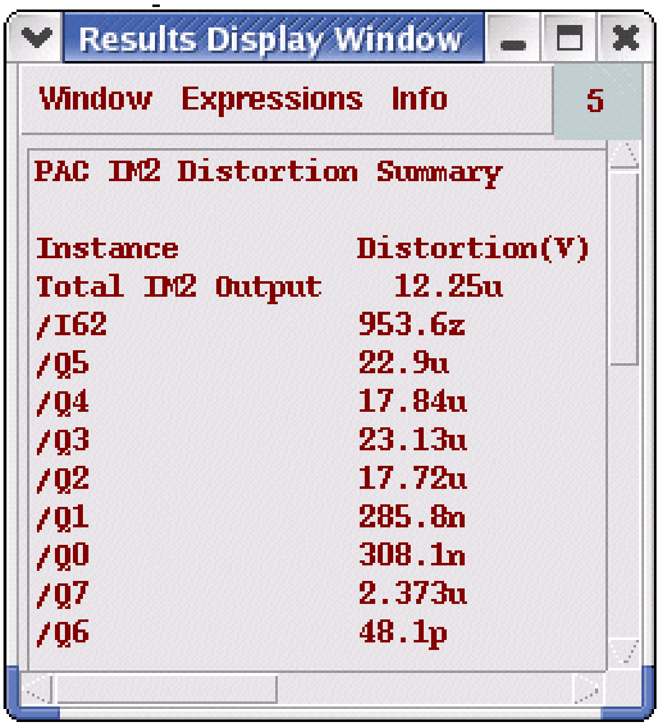
   **Out-:** *out1*.

   **Note:** If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre does not calculate it.

3. Run the PSS and the PAC simulations.
4. Display/Data Analysis

   Choose *Results-Print* to display the *IM2 Distortion Summary* as shown in Figure 5-8.

**Figure 5-8  PAC IM2 Distortion Summary for the Mixer**




# Rapid IP3 and Compression Distortion Summary Measurements for a Power Amplifier

One of the main concerns in power amplifier (PA) design is the nonlinearity effect. The nonlinearity of a PA can degrade the quality of the Tx signal and interfere with adjacent channels. Typically, IP3 and adjacent channel power ratio (ACPR) are used to characterize PA nonlinearity.

In this example,

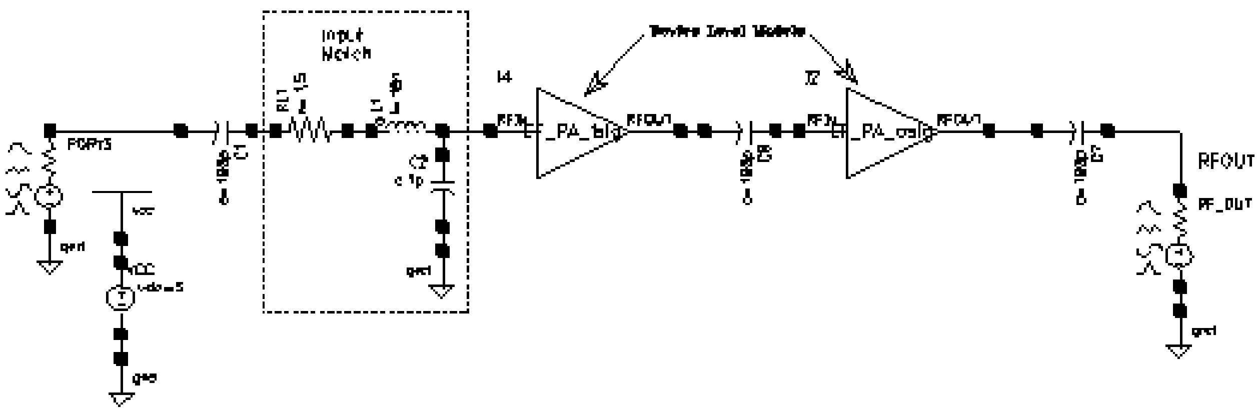
- Rapid IP3 is used to measure the PA's IP3.

- The contribution of every nonlinear device to the PA's total nonlinearity is evaluated using the Compression Distortion Summary.

Compared with measurements made for the mixer, IP3 measurement for a PA or LNA can use AC-based Rapid IP3 and Compression Distortion Summary because their operating point is time-independent. In this case, AC-based measurement is more efficient than PAC-based Rapid IP3 and Compression Distortion Summary.

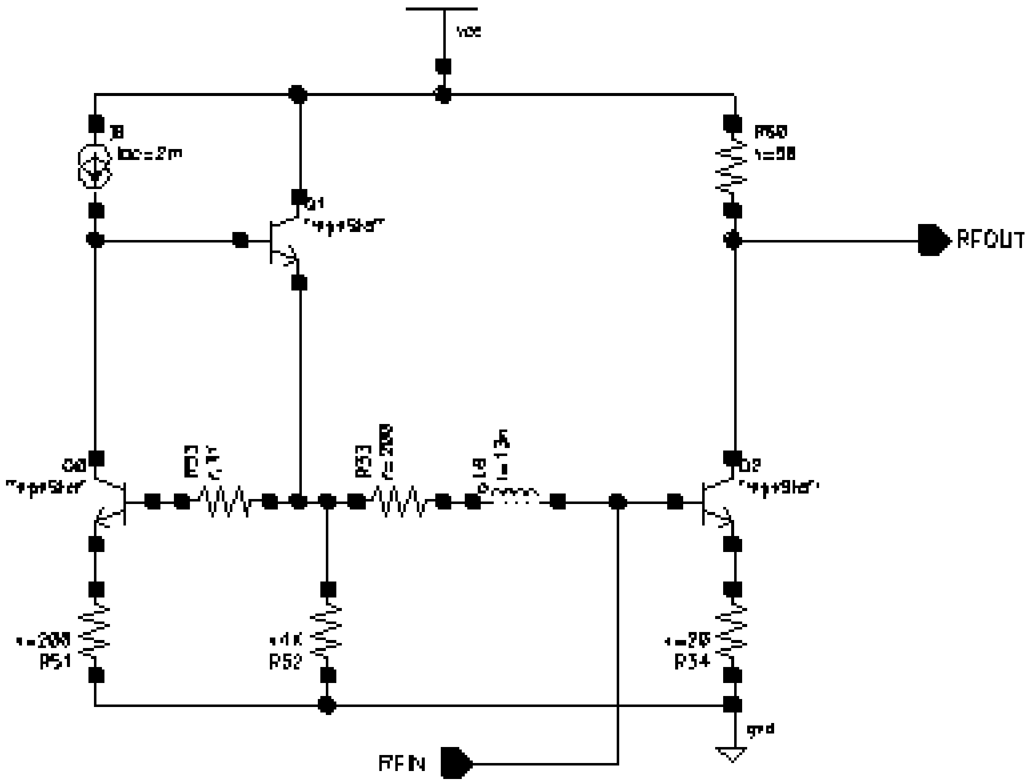
The PA used in the following measurements is shown in Figure 5-9. This PA is an edited version of the *EF_example* schematic in the *rfExamples* library. It consists of three blocks: input match, the input amplifier (I4) and the output amplifier (I2).

**Figure 5-9  The Power Amplifier Schematic (EF_example)**



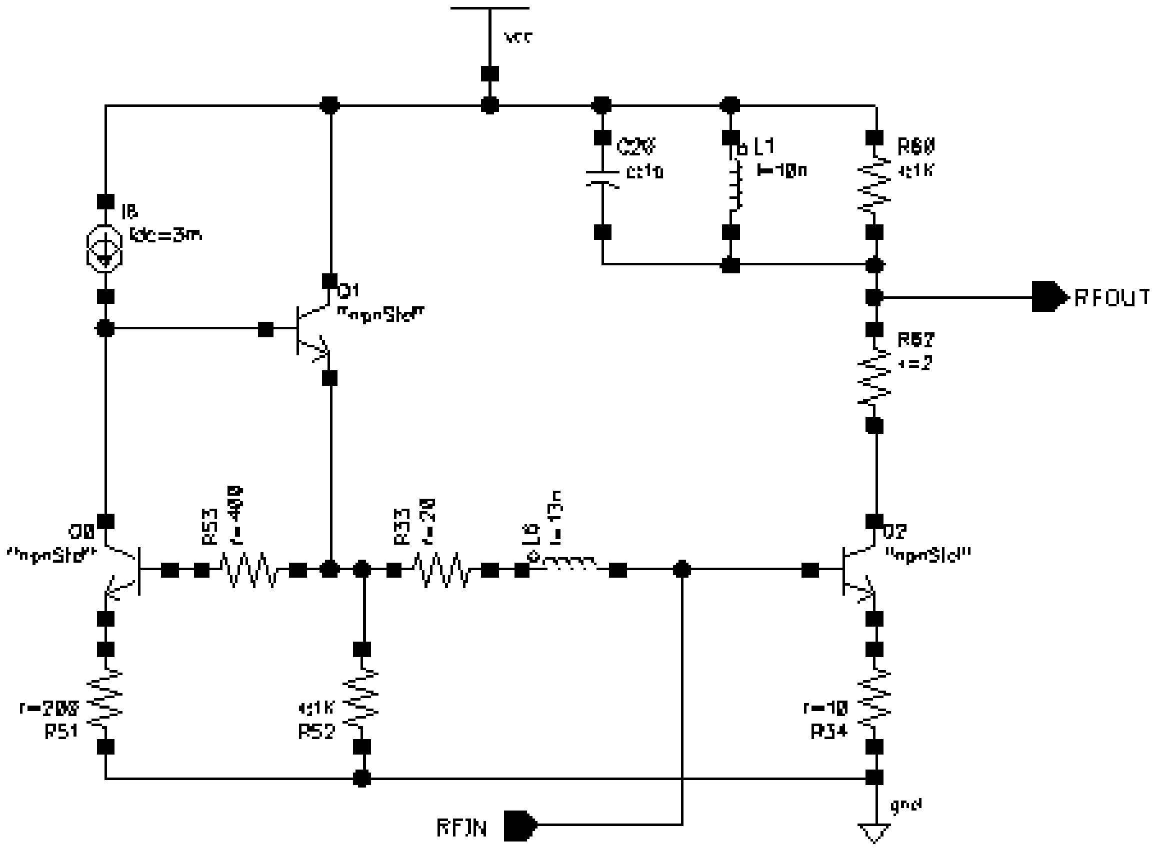

Subcircuit I4 is shown in Figure 5-10. This input amplifier subcircuit is called *EF_PA_istg* and it is in the *rfExamples* library.

**Figure 5-10 Schematic for Subcircuit I4 (the Input Amplifier Subcircuit)**




Subcircuit 12 is shown in Figure 5-11. This output amplifier subcircuit is called *EF_PA_ostg* and it is in the *rfExamples* library.

**Figure 5-11  Schematic for Subcircuit 12 (the Output Amplifier Subcircuit)**




## The Rapid IP3 Measurement

The Rapid IP3 PA measurement is calculated using Spectre RF in ADE.

### Stimuli:

For the RF port (PORT3), set *dc* as the *Source Type*. Set the parameter for *AC magnitude* to `7m` Volts in the *small signal params* section. This value will be used as the input magnitude in the AC small signal analysis.

**Parameters:**

Set the parameters

| | |
|---|---|
| *frf1* | *1G* |
| *frf2* | *frf1 + 0.002G* |
| *prf* | *-30* |

**Simulation/Analyses:**

1. Set up an AC analysis.

   **Specialized Analyses:** *Rapid IP3*

   **Source Type:** *port*

   **Input Sources 1:** */PORT3* **freq:** *frf1*

   **Input Sources 2:** */PORT3* **freq:** *frf2*

   ***Input Power (dBm):*** *prf*

   **Frequency of IM Output Signal:** *2frf1-frf2*

   **Frequency of Linear Output Signal:** *frf2*

   **Maximum Non-Linear Harmonics:** *4*

   **Output:** *Voltage*

   **Out+:** */RFOUT*

   **Out-:** */gnd!*

   **Note:** If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

2. Run the AC simulation.

3. Display/Data Analyses.

   To display the Rapid IP3 results,

   a. Open the Direct Plot form.

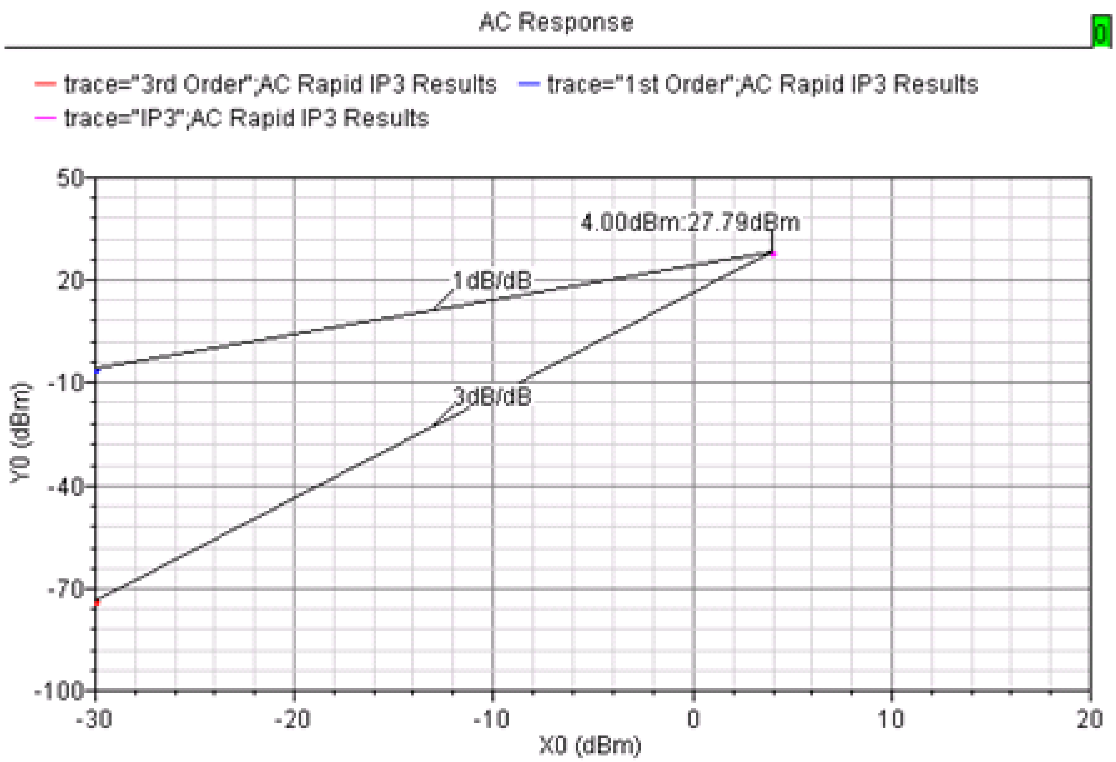
   b. Click *AC* analysis.

**c.** Select *Rapid IP3*.

**d.** Click *plot*.

Rapid IP3 for the PA is plotted as shown in Figure 5-12.

**Figure 5-12 AC Rapid IP3 Plot for the PA**




## Measuring Compression Distortion Summary

The Compression Distortion Summary PA measurement is calculated using SpectreRF in ADE.

### Stimuli:

For the RF port (PORT3), set *dc* as the *Source Type*. Set the parameter for *AC magnitude* to `7m` Volts in the *small signal params* section. This value will be used as the input magnitude in the AC small signal analysis.

**Parameters:**

Set the parameters:

*frf* *1.002G*

*prf* *-30*

**Simulation/Analyses:**

1. Set up an AC analysis.

   **Start:** *frf*

   **Specialized Analyses:** *Compression Distortion Summary*

   **Contributor Instances:**

   **Frequency of Linear Output Signal:** *frf*

   **Maximum Non-Linear Harmonics:** *4*

   **Output:** *Voltage*

   **Out+:** */RFOUT*

   **Out-:** */gnd!*

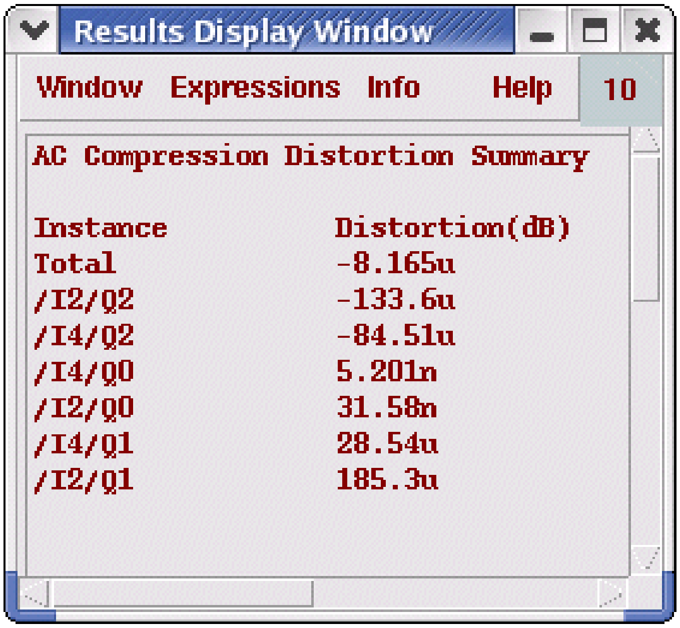
   **Note:** If output is current in a port, user must use the save statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

**Display/Data Analysis:**

➤ Display Rapid IP3 results by selecting *Results-Print-AC Distortion Summary*.

The result is shown in Figure 5-13. It can be seen that the distortion from subcircuit I2 devices is more significant than that from the subcircuit I4 devices. This may be because subcircuit I2 is the second stage amplifier in the PA and it performs the main power amplifier function. It also should be noticed that the total distortion is lessened for the interaction between devices.

### Figure 5-13 AC Compression Distortion Summary for the PA



# 6

# Simulating Oscillators

Autonomous circuits, such as oscillators, are time-invariant circuits with time-varying responses. Thus, autonomous circuits generate non-constant waveforms even though they are not driven by a time-varying stimulus.

You cannot specify the analysis period for autonomous circuits because you do not know the precise oscillation period in advance of the simulation. Instead, you specify an estimate of the oscillation period for the simulation. The PSS analysis uses your estimate to compute the precise period and the periodic solution waveforms.

## Phases of Autonomous PSS Analysis

A PSS analysis has two phases:

- A transient analysis phase to initialize the circuit

  The transient analysis phase is divided into three intervals:

  - A beginning interval that starts at `tstart`, which is normally 0, and continues through the onset of periodicity for the independent sources
  - A stabilization interval of length `tstab`

    For driven circuits, the stabilization interval is optional. For autonomous circuits, `tstab` needs to be set to an interval that gives the circuit a good initial condition.
  - A final interval that is four times the estimated oscillation period specified in the PSS Analysis form

    During the final interval, the PSS analysis monitors the waveforms in the circuit and improves the estimate of the oscillation period.

- A shooting phase to compute the periodic steady state solution

  During the shooting phase, the circuit is simulated repeatedly over one period. The length of the period and the initial conditions are modified to find the periodic steady state solution.

See *Autonomous PSS Analysis* in *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on the autonomous PSS analysis algorithm.

## Phase Noise and Oscillators

Oscillators tend to amplify any noise present near the oscillation frequency. The closer the noise frequency is to the oscillation frequency, the greater the noise amplification. Noise amplified by the oscillator in this manner is called *phase noise*. Phase noise is the most significant source of noise in oscillators, and because phase noise is centered about the oscillation frequency, filtering can never completely remove it.

You can understand phase noise if you recognize that the phase of an oscillator is arbitrary because there is no drive signal to lock to. Any waveform that is a solution to an oscillator can be shifted in time and still be a solution. If a perturbation disturbs the phase, nothing restores the phase, so it drifts without bound. If the perturbation is random noise, the drift is a random walk. Furthermore, the closer the perturbation frequency is to the oscillation frequency, the better it couples to the phase and the greater the drift. The perturbation need not come from random noise. Noise might also couple into the oscillator from other sources, such as the power supplies.

The first two examples in this chapter both examine phase noise.

## Starting and Stabilizing Oscillators

To simulate an oscillator using PSS analysis, you must first start the oscillator by supplying either:

- A brief impulse stimulus

  The stimulus should couple strongly into the oscillatory mode of the circuit and poorly into other long-lasting modes such as bias circuitry.

- A set of initial conditions for the components of the oscillator's resonator

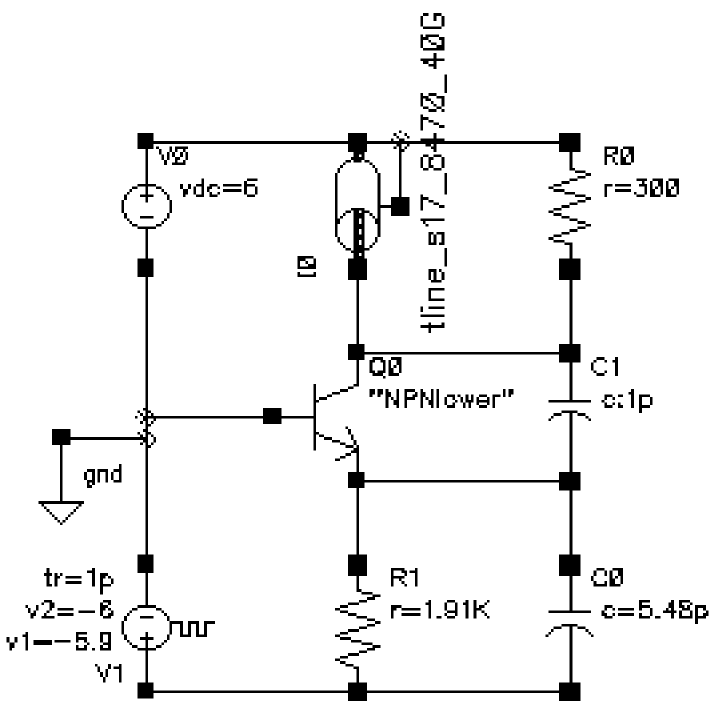
Regardless of which technique you use to start the oscillator, allow the oscillator to run long enough to stabilize before you start the shooting phase and compute the steady state solution. Adjust the *tstab* parameter to supply the additional stabilization period.

See *Oscillators and Autonomous PSS Analysis* in the *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on oscillator simulation. For details about the autonomous PSS analysis see *Autonomous PSS Analysis* in the *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

# The tline3oscRF Oscillator Circuit

This example computes the periodic steady state solution and the phase noise for the *tline3oscRF* circuit shown in Figure 6-1.

#### Figure 6-1 Schematic for the tline3oscRF Oscillator Circuit




This circuit supplies the brief impulsive stimulus needed to start the oscillator, so you do not need to set initial conditions. You supply the data necessary for PSS analysis to estimate the period when you specify the fundamental frequency.

## Simulating the tline3oscRF Oscillator Circuit

Before you begin, perform the setup procedures described in Chapter 4, "Setting Up for the Examples."

### Opening the tline3oscRF Circuit in the Schematic Window

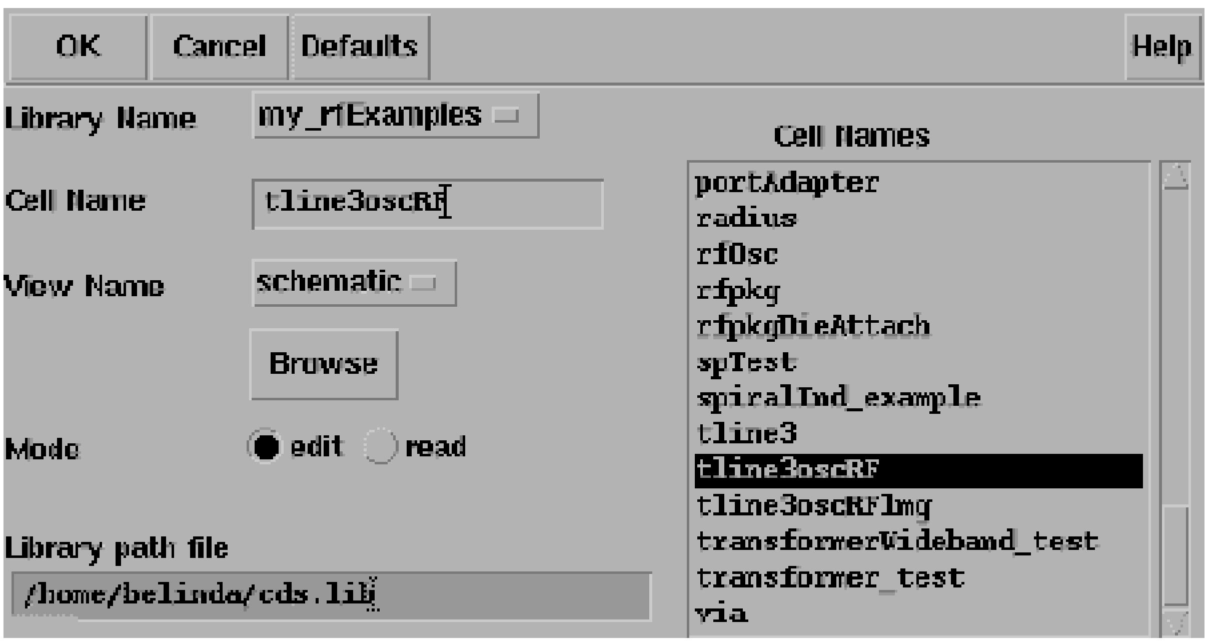
1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created. SeeChapter 4, “Setting Up for the Examples” for more information.
3. Choose *schematic* for *View Name*.
4. In the *Cell Names* list box, highlight *tline3oscRF*.

   *tline3oscRF* appears in the *Cell Name* field.
5. Highlight *edit* to choose the *Mode*.

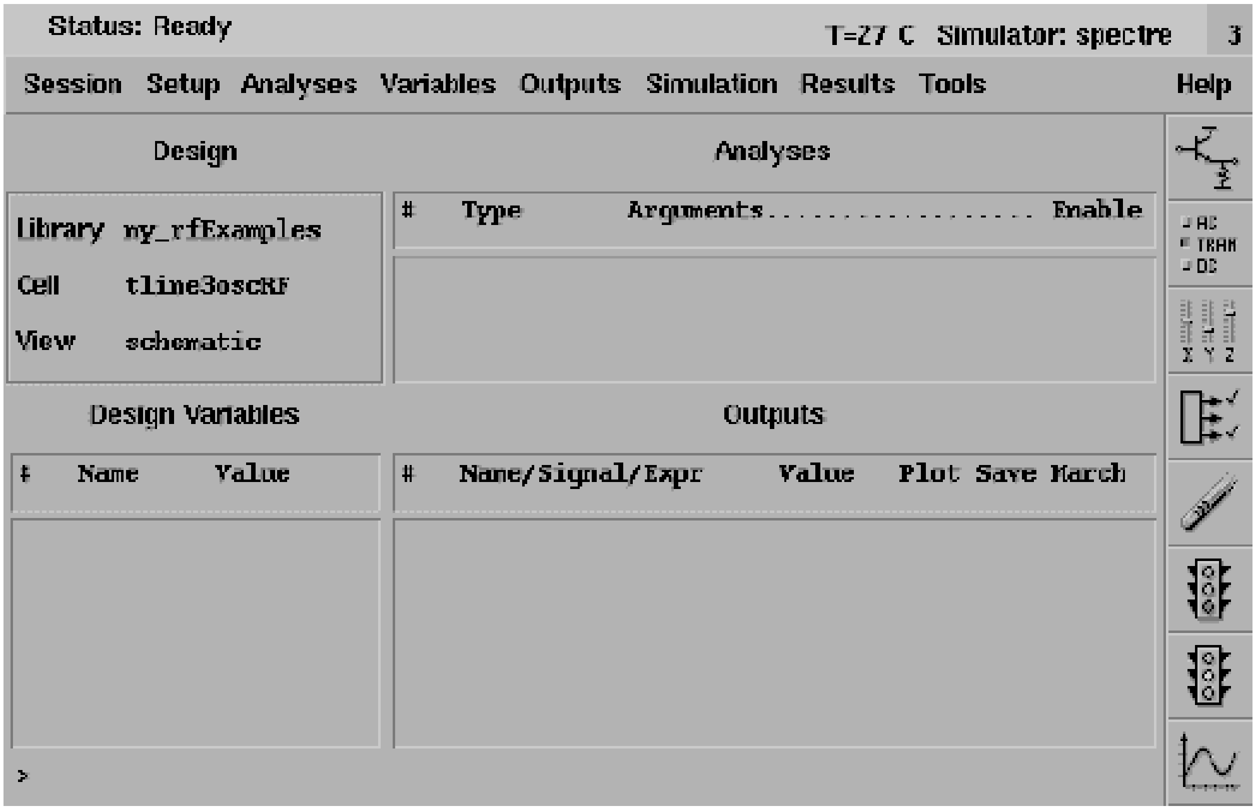
   The completed Open File form appears like the one below.




6. Click *OK*.

   The Schematic window for the *tline3oscRF* oscillator opens.
7. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *tline_3oscRF* in the Choosing Design form.

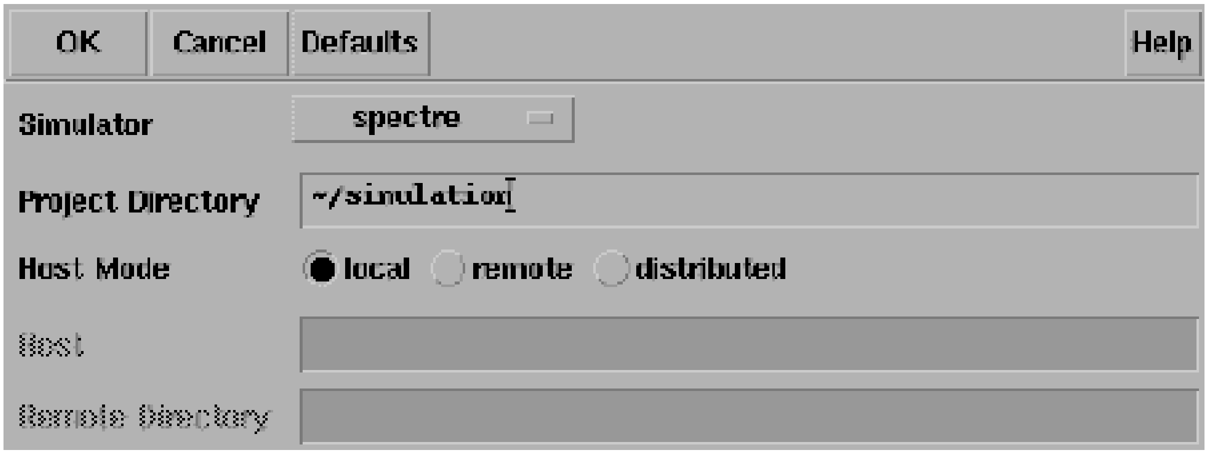
### Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator*/*Directory*/*Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, choose *spectre* for *Simulator*.
3. Type in the name of the project directory, if necessary.
4. Highlight the appropriate mode for the *Host Mode*.
5. For remote or distributed simulations, type the required information into the fields that become available for each choice.

The completed form looks like this.




6. Click *OK*.

### Setting Up Model Libraries

1. In the ADE window, choose *Setup – Model Libraries*.

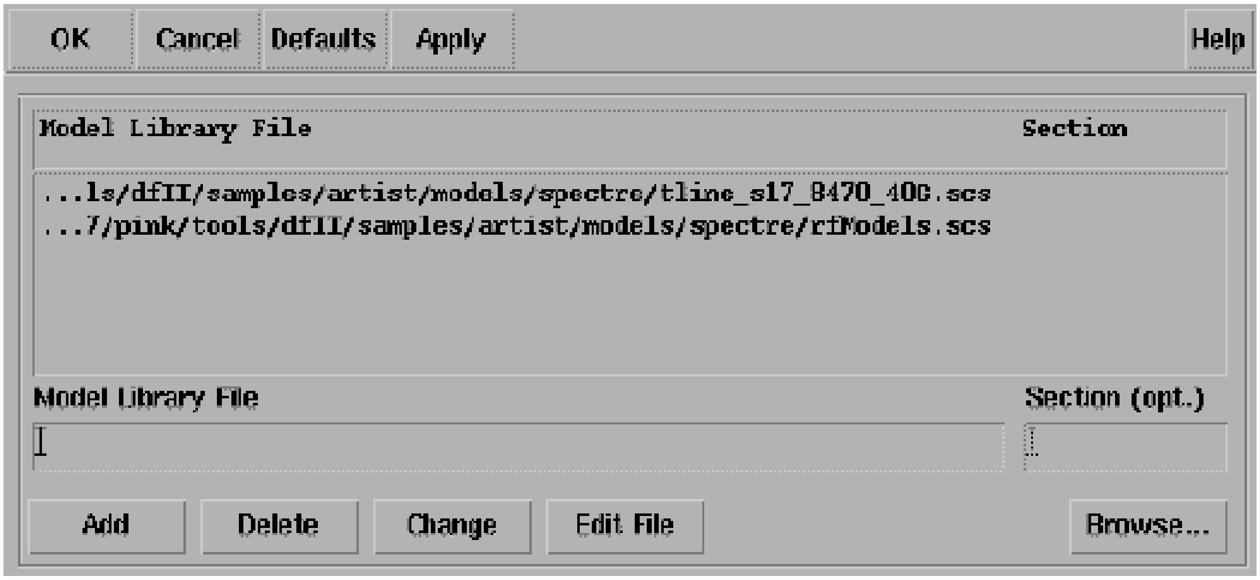
   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the `rfModels.scs` model file including the file name, `rfModels.scs`.

3. Click *Add*.

4. In the *Model Library File* field, type the full path to the `tline_s17_8470_40G.scs` model file including the file name, `tline_s17_8470_40G.scs` and click *Add*.

The completed form looks like this.




**5.** In the Model Library Setup form, click *OK*.

# Periodic Steady State and Phase Noise with PSS and Pnoise

This example computes the periodic steady state solution and phase noise for the *tline3oscRF* oscillator circuit. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

## Setting Up the Simulation

**1.** If necessary, open the tline3oscRF circuit.

**2.** If necessary, specify the full path to the model files.

**3.** If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

## Setting Up the PSS Analysis

**Note:** If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a transient analysis before you run this PSS analysis. From the transient analysis you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click *pss*.

   The form changes to display options needed for `PSS` analysis.

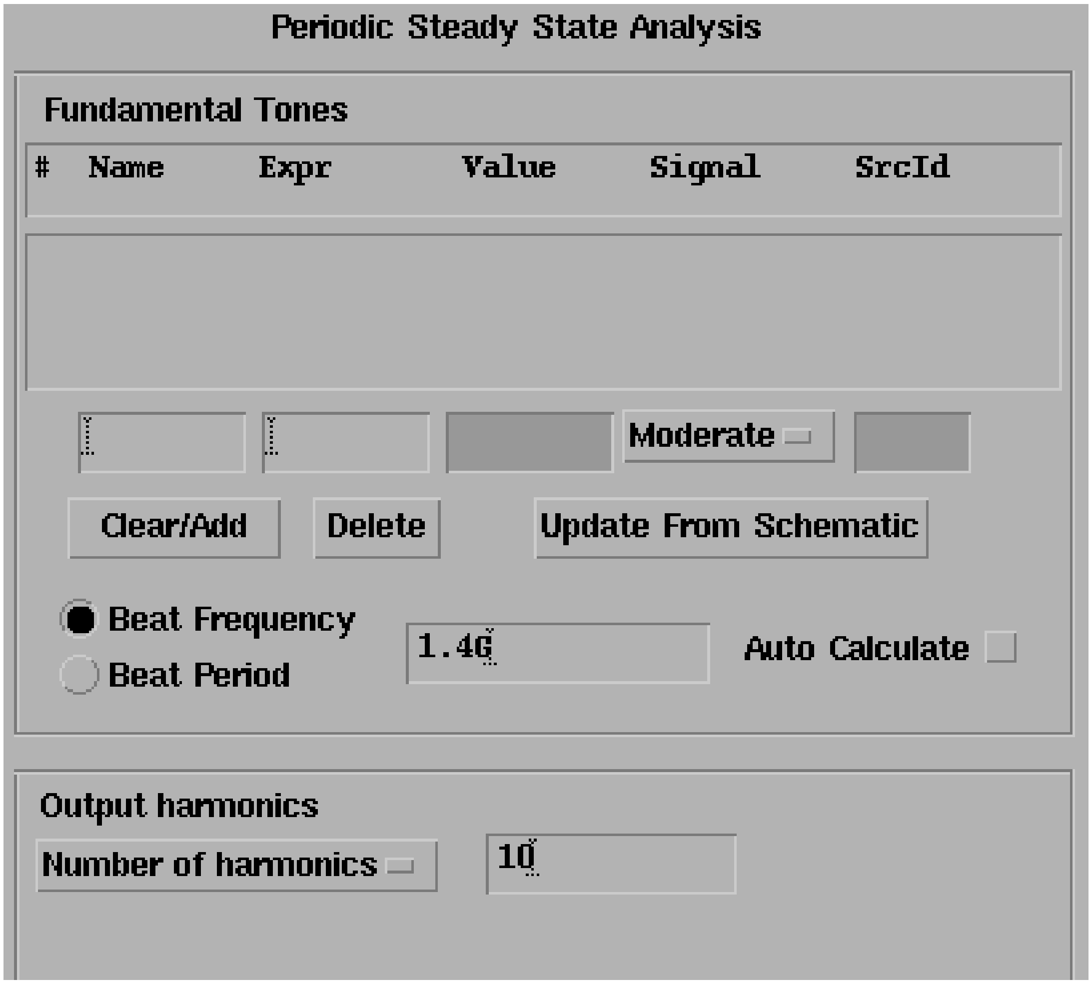
   In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

   In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   Type `1.4G` in this example.

3. In the *Output harmonics* cyclic field, choose Number of harmonics and type in a reasonable number, such as `10`.

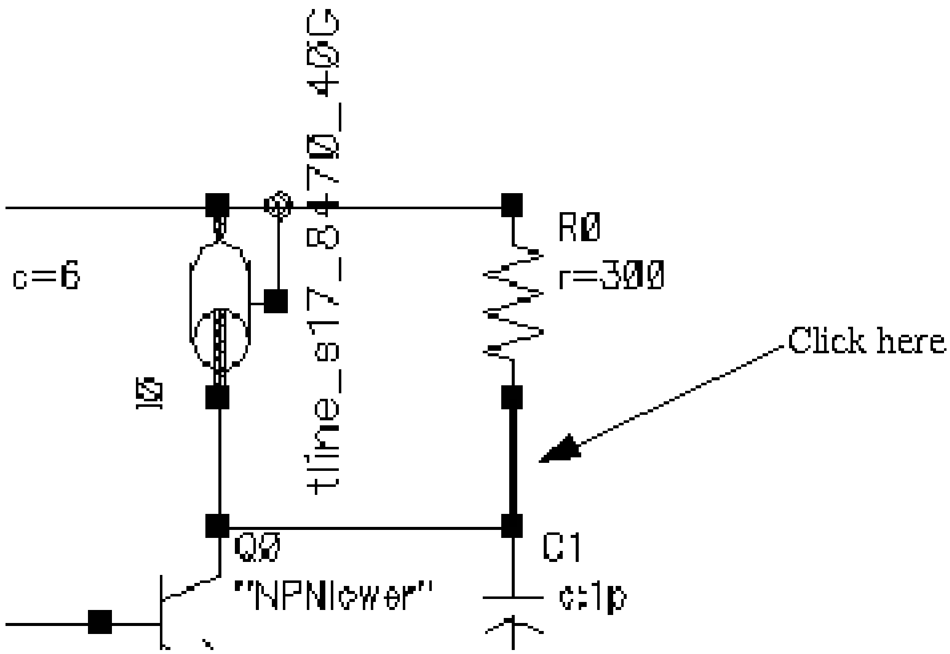
The top of the PSS Choosing Analyses form appears below.




4. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
5. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.

**6.** Click the *Select* button next to the *Oscillator node* field. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose *net5*.



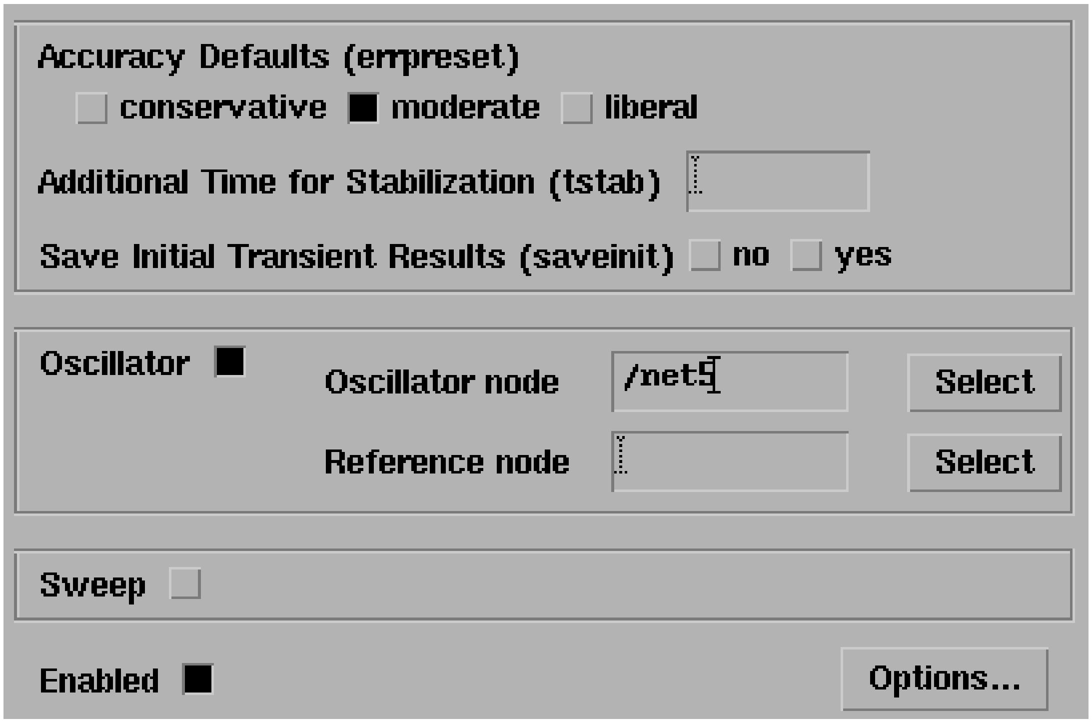

`/net5` appears in the *Oscillator node* field.

**7.** Leave the *Reference node* field empty.

The *Reference node* field defaults to `/gnd!`. You can set *Reference node* to a different value by clicking on the *Reference node Select* button and then choosing the appropriate node in the schematic.

**8.** Verify that *Enabled* is highlighted.

The bottom of the PSS Choosing Analyses form appears below.




## Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the Pnoise analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value `2K` away from the fundamental frequency.

3. Choose *Start-Stop* for the type of *Frequency Sweep Range (Hz)*. Type in reasonable values, such as `1K` and `100M` as the *Start* and *Stop* values.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `201` in the field.

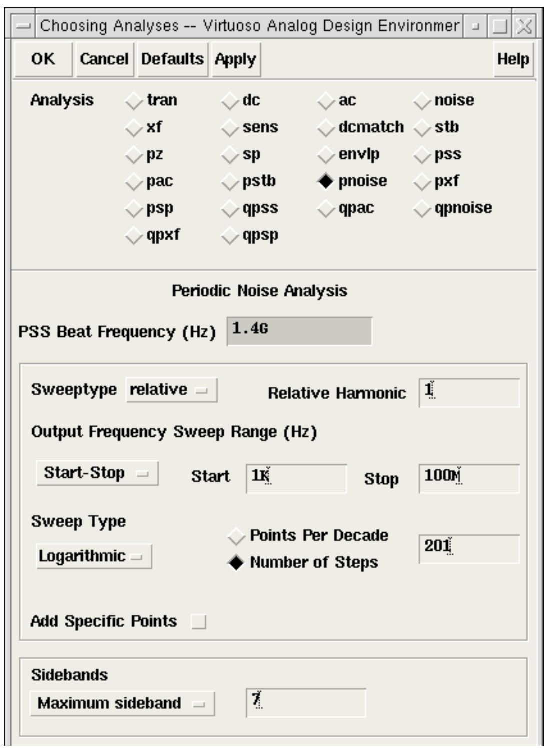
   **Note:** Be sure to use a nonzero starting value for logarithmic sweeps.

6. In the *Sidebands* cyclic field, choose [*Maximum sideband*]() and type `7` in the *Maximum sideband* field. The default parameter value is `maxsideband=7`.
7. Begin with a *Maximum sideband* value of 7. Then, in subsequent simulations, increase the value to see whether the output noise changes. Continue to increase the *Maximum sideband* value until the output noise stops changing.

   For a fundamental oscillator, *Maximum sideband* must be at least `1` to see any flicker noise up conversion. In general, small values for *Maximum sideband* are not recommended.

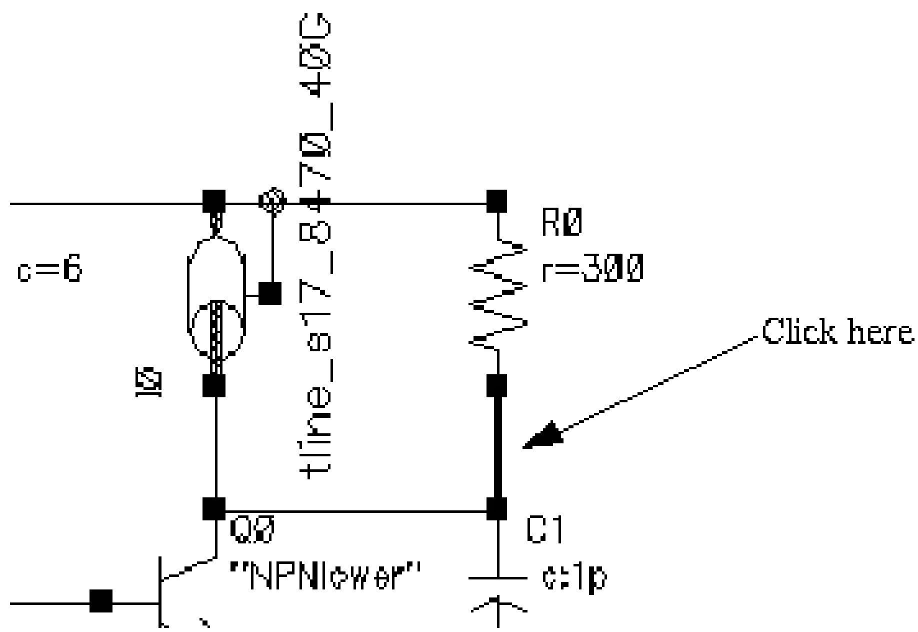
   Be sure to use a nonzero *Maximum sideband* value.

   The top of the Pnoise Choosing Analyses form appears below.




8. In the *Output* cyclic field, choose *voltage*.

**9.** Click the *Select* button next to the *Positive Output Node* field.Then click the appropriate wire in the Schematic window to choose net 5.



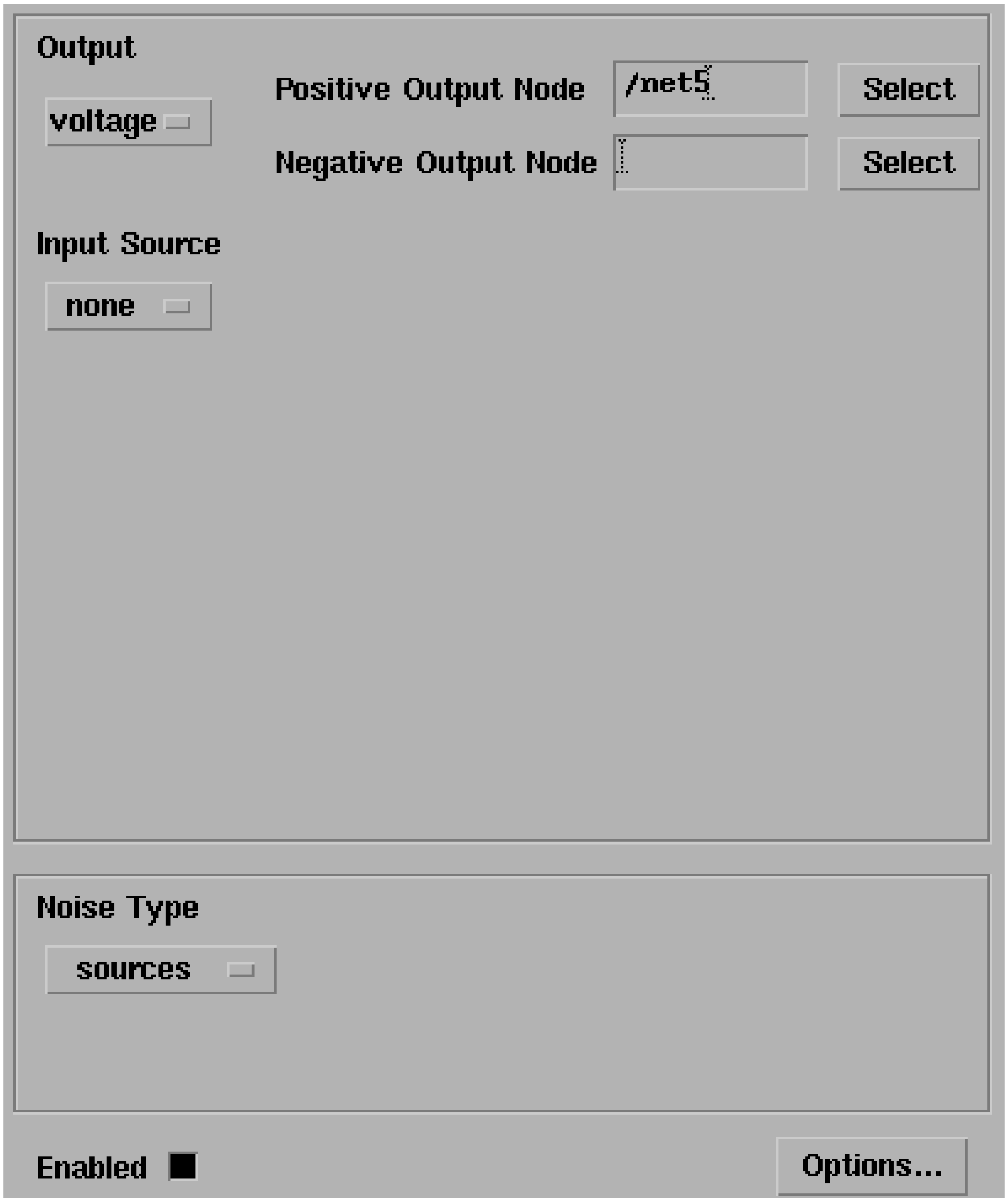

`/net5` appears in the *Positive Output Node* field.

**10.** Leave the *Negative Output Node* field empty.

This field defaults to `/gnd!`. You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

**11.** In the *Input Source* cyclic field, choose *none*.

The bottom of the Pnoise Choosing Analyses form appears below.




**12.** Verify that the Pnoise analysis is Enabled. Then click *OK* in the Choosing Analyses form.

The PSS and `Pnoise` options you choose appear in the *Analyses* list box in the ADE window.

Analyses

| # | Type | Arguments.................. | | | | Enable |
|---|---|---|---|---|---|---|
| 1 | pnoise | 7 | 1K | 100M | 201 .. | yes |
| 2 | pss | 1.4G | 10 | /net5 | | yes |

## Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Fundamental Frequency

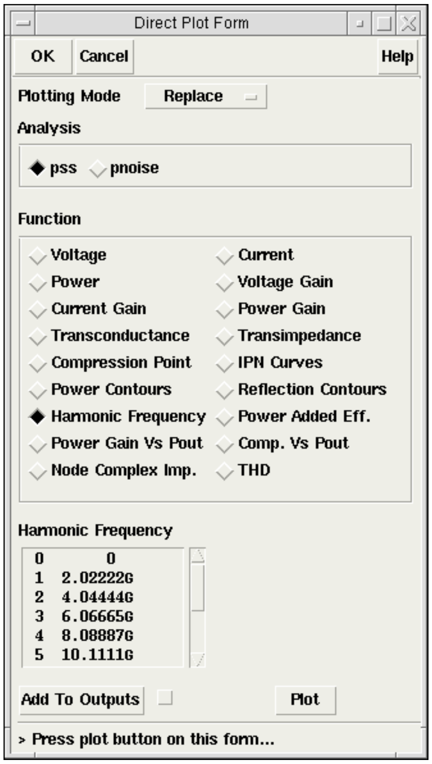
These steps describe how to plot the fundamental frequency.

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

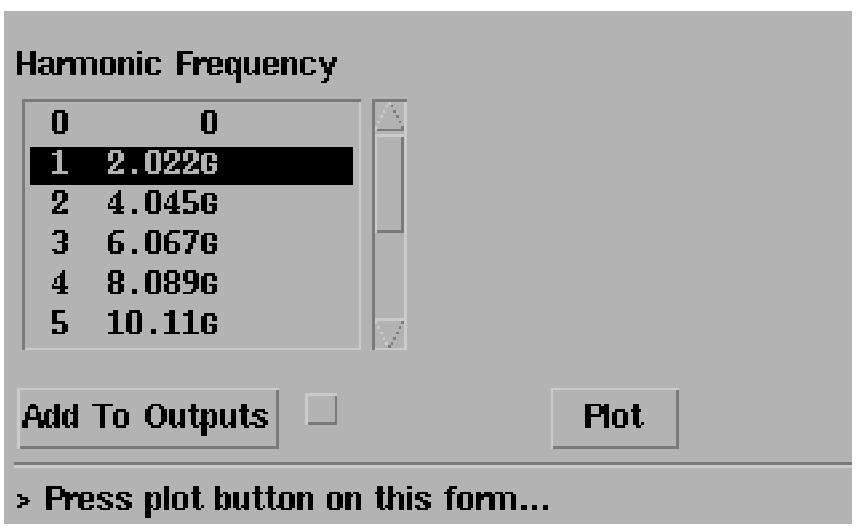
   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis.*
4. Highlight *Harmonic Frequency* for *Function*.

The form changes to display the *Harmonic Frequency* list box.

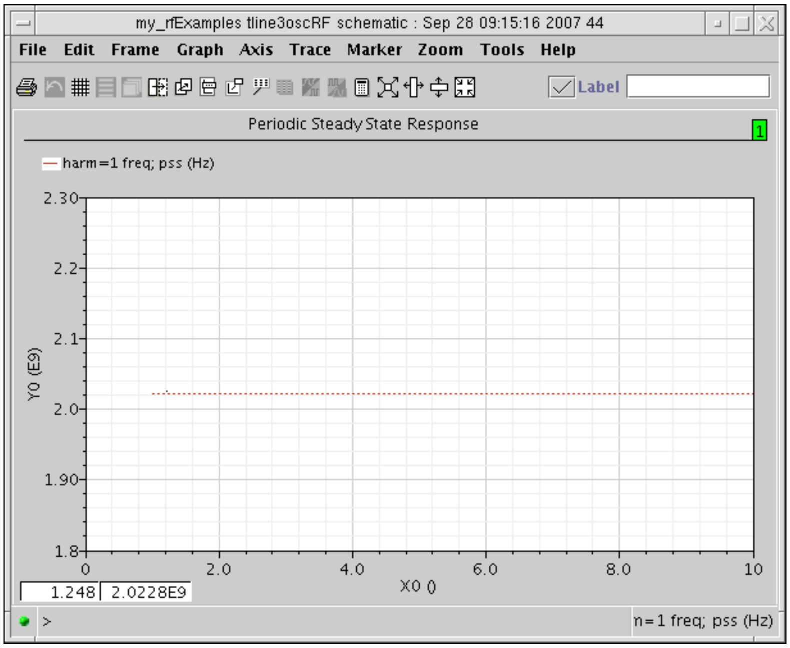



5. As the prompt at the bottom of the form instructs, highlight the first harmonic in the *Harmonic Frequency* list box.




6. Click *Plot*.

The waveform window displays the fundamental frequency.




Notice that the fundamental frequency value of `2.022G` is also displayed in the *Harmonic Frequency* list box in the Direct Plot form.

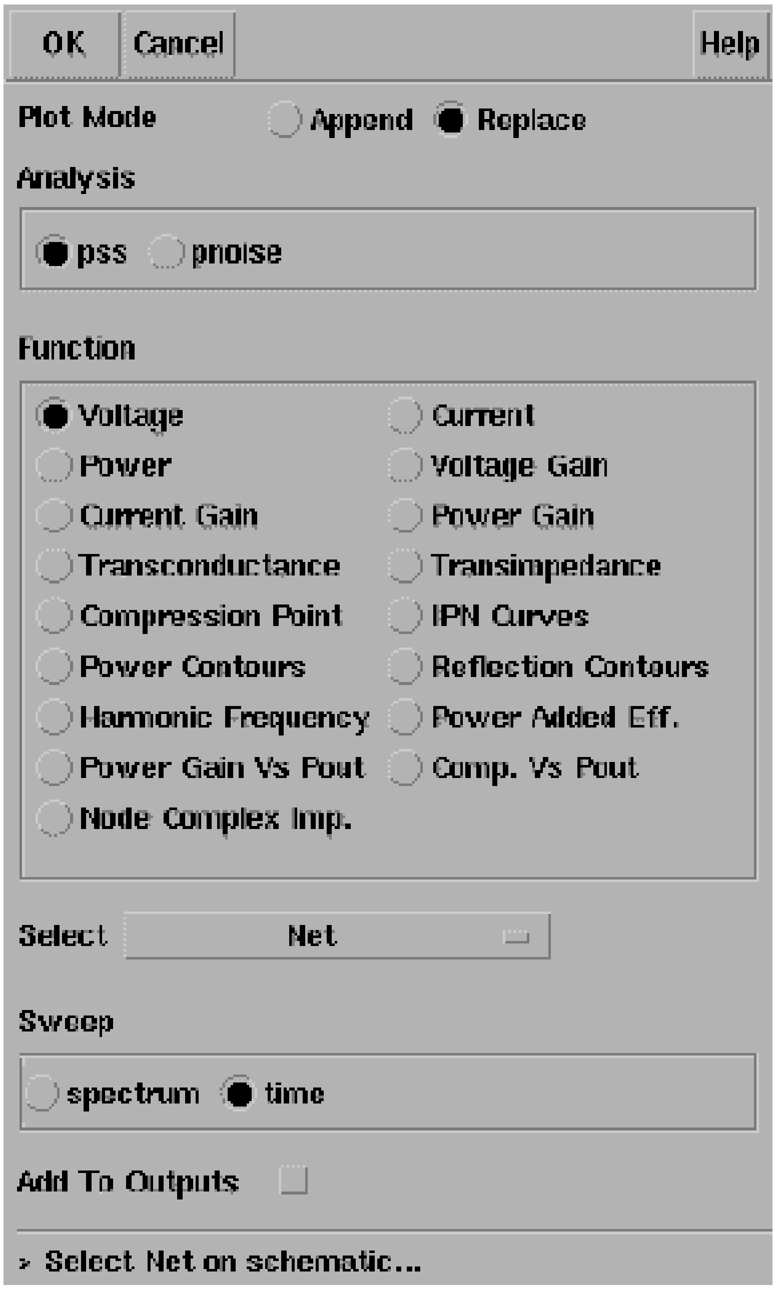
## Plotting the Periodic Steady State Solution

If necessary, in the ADE window choose *Results – Direct Plot – Main Form* to open the Direct Plot form.

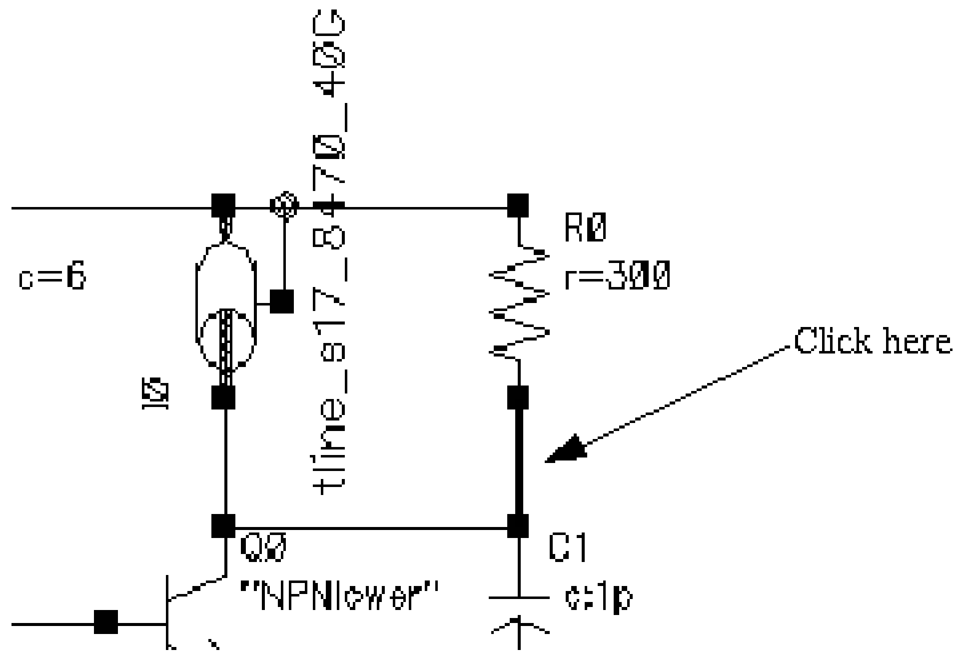
To plot the periodic steady state solution waveform for distortion, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Highlight *Time* for *Sweep*.

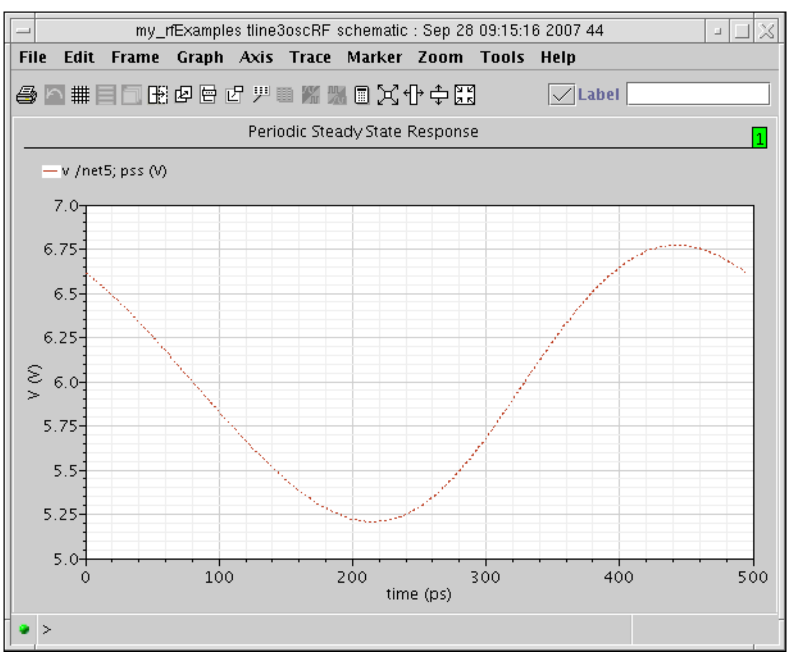
5. Notice that the default choice in the *Select* cyclic field is *Net*. Also notice the *Select net on schematic* prompt at the bottom of the form.



**6.** In the Schematic window, click the appropriate wire to choose net 5.




The waveform window displays the sweep of voltage versus time—the periodic steady state solution.



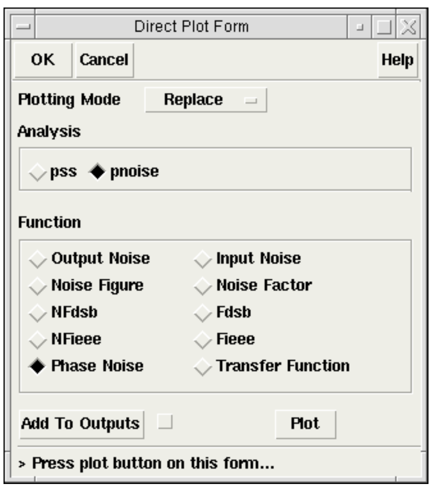
## Plotting the Phase Noise

If necessary, in the ADE window choose *Results – Direct Plot – Main Form* to open the Direct Plot form.

To plot the phase noise waveform, do the following:

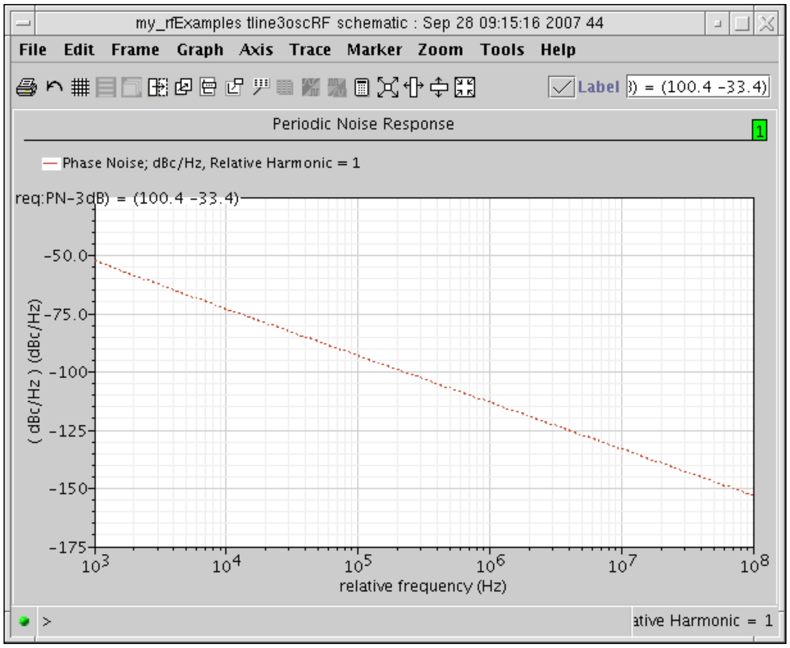
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise* for *Analysis*.

   The form changes to display fields relevant for the `Pnoise` analysis.
3. Highlight *Phase Noise* for *Function*.




4. Click *Plot*.

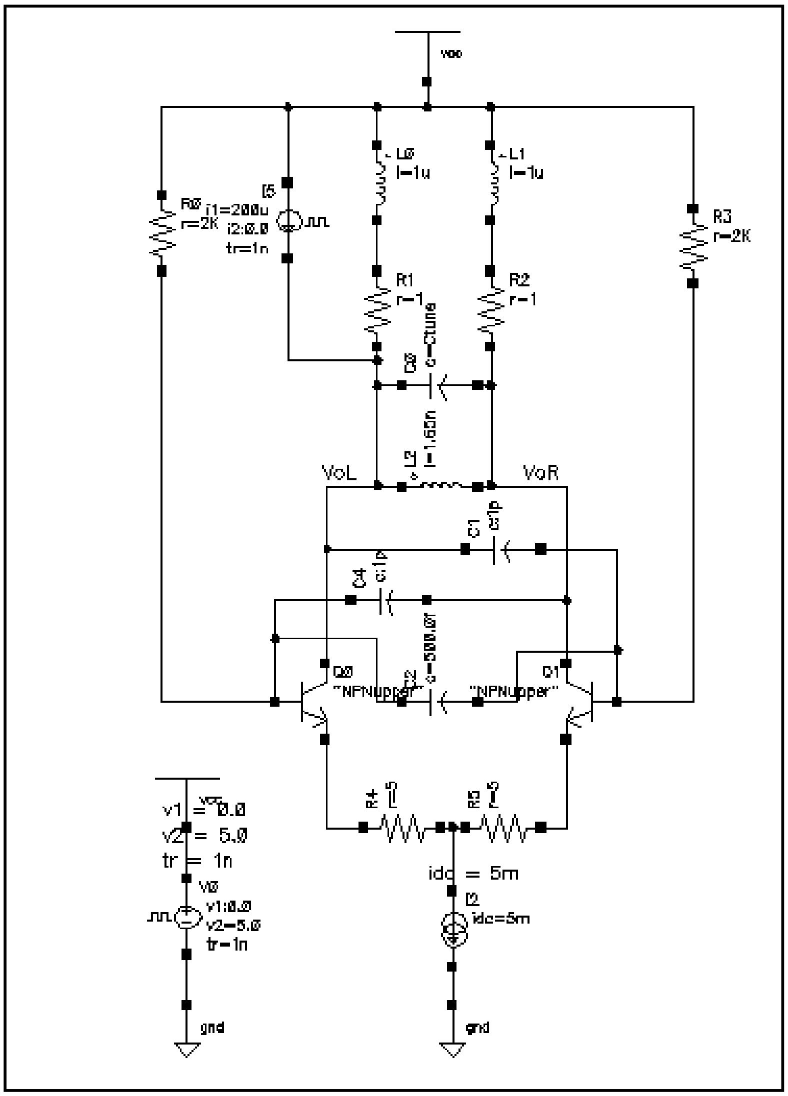
The waveform window displays the phase noise versus frequency with the frequency in logarithmic scale. When you plot phase noise, you get more useful information if you use the logarithmic scale.



# The oscDiff Circuit: A Balanced, Tunable Differential Oscillator

This example computes the fundamental frequency, output noise, and phase noise for the *oscDiff* circuit, the balanced, tunable differential oscillator shown in Figure 6-2.

**Figure 6-2 Schematic for the oscDiff Oscillator Circuit**




The *oscDiff* circuit generates sinusoids in antiphase for differential nodes *VoL* and *VoR*. The oscillation frequency is primarily governed by inductor *L0* and capacitor *C3*. The *C3* capacitance value is voltage-dependent and obeys Equation 6-1, the standard Schottky barrier capacitance equation.

**(6-1)**

$$Ctune = \frac{Cj0}{\left(1 + \frac{Vcntrl}{phi}\right)^{gamma}}$$

where

| | |
|---|---|
| *Ctune* | *C3* capacitance value |
| *Cj0* | Zero-bias junction capacitance |
| *Vcntrl* | Applied varactor voltage |
| *phi* | Barrier potential |
| *gamma* | Junction grading coefficient |

# Simulating the oscDiff Circuit

Before you begin, be sure that you have performed the setup procedures described in Chapter 4, "Setting Up for the Examples."

## Opening the oscDiff Circuit in the Simulation Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created. See Chapter 4, "Setting Up for the Examples"for more information.

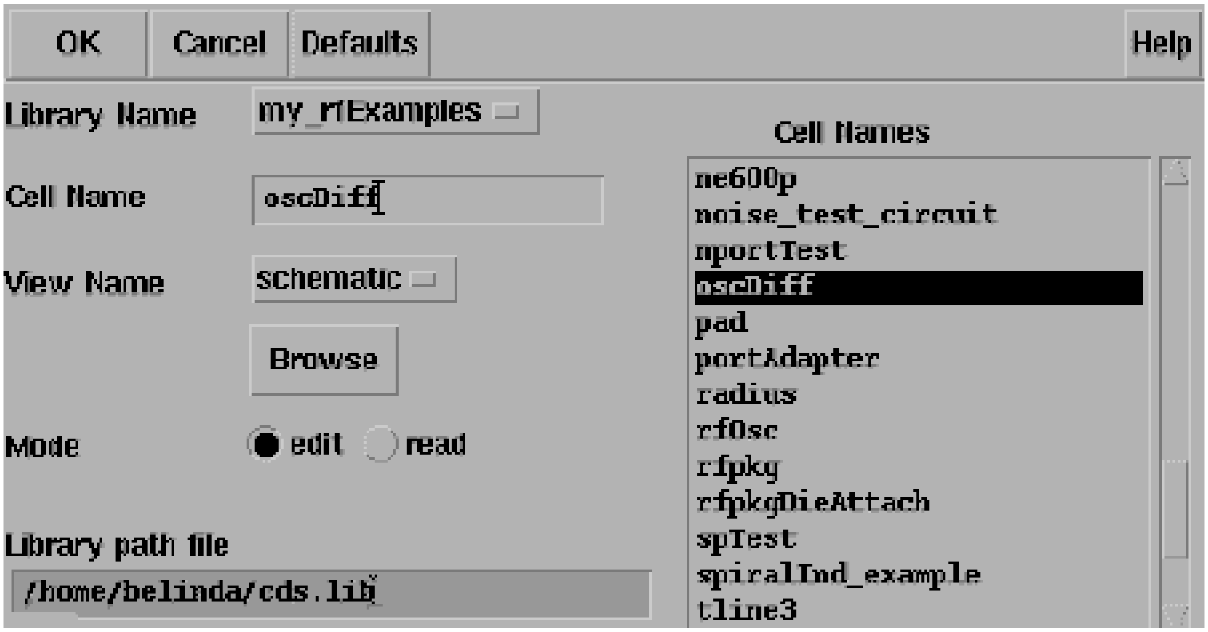
3. Choose *schematic* for *View Name*.

4. In the *CellNames* list box, highlight *oscDiff*.

   *oscDiff* appears in the Cell Name field.

5. Highlight *edit* for *Mode*.

The completed Open File form appears like the one below.

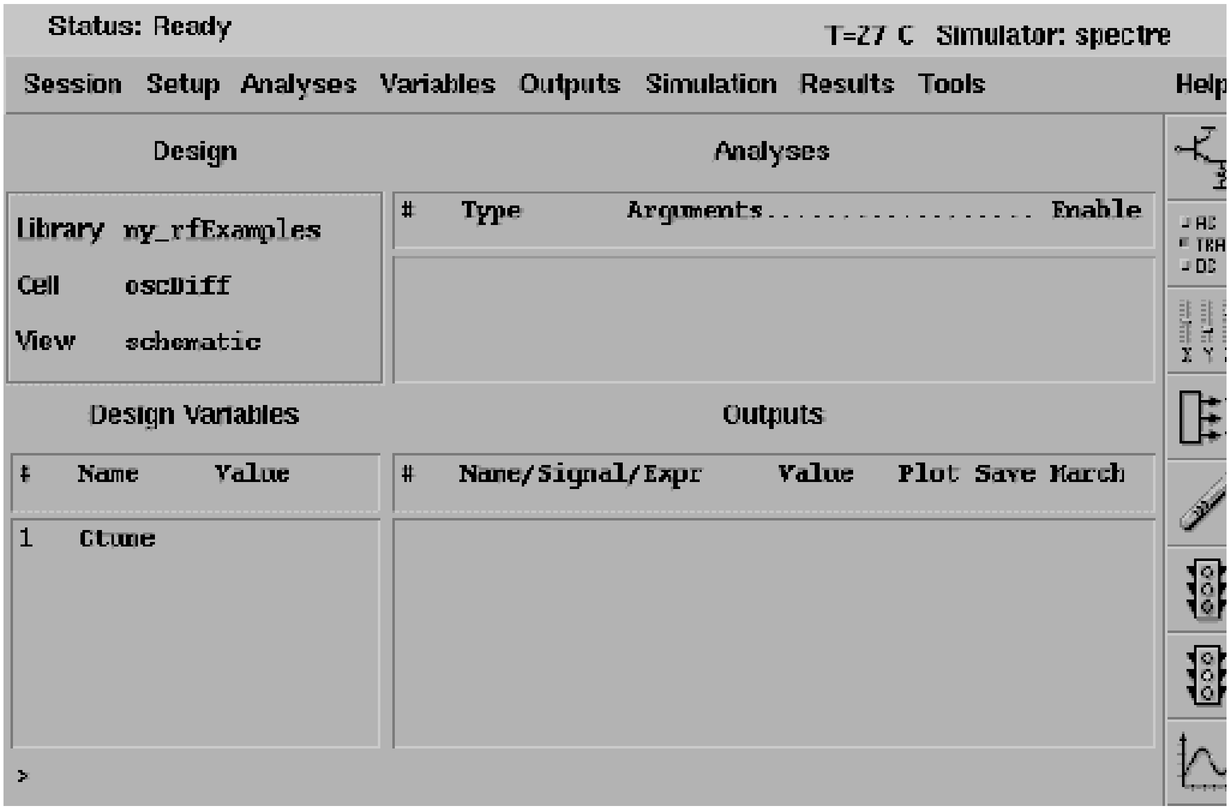



**6.** Click *OK*.

The Schematic window for the oscDiff oscillator opens as shown in Figure 6-2 on page 392.

**7.** In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *oscDiff* in the Choosing Design form.

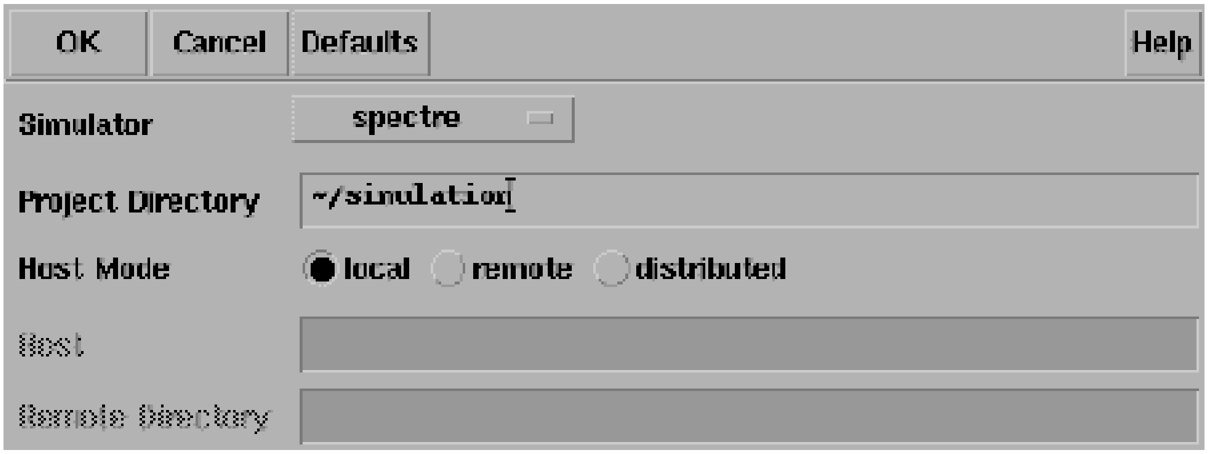
## Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, choose *spectre* for *Simulator*.
3. Type in the name of the project directory, if necessary.
4. Highlight the appropriate button to specify the *Host Mode*.
5. For remote or distributed simulation, fill in the additional fields that become available.

The completed form appears like the one below.




**6.** In the Choosing Simulator/Directory/Host form, click *OK*.

## Setting Up Model Libraries

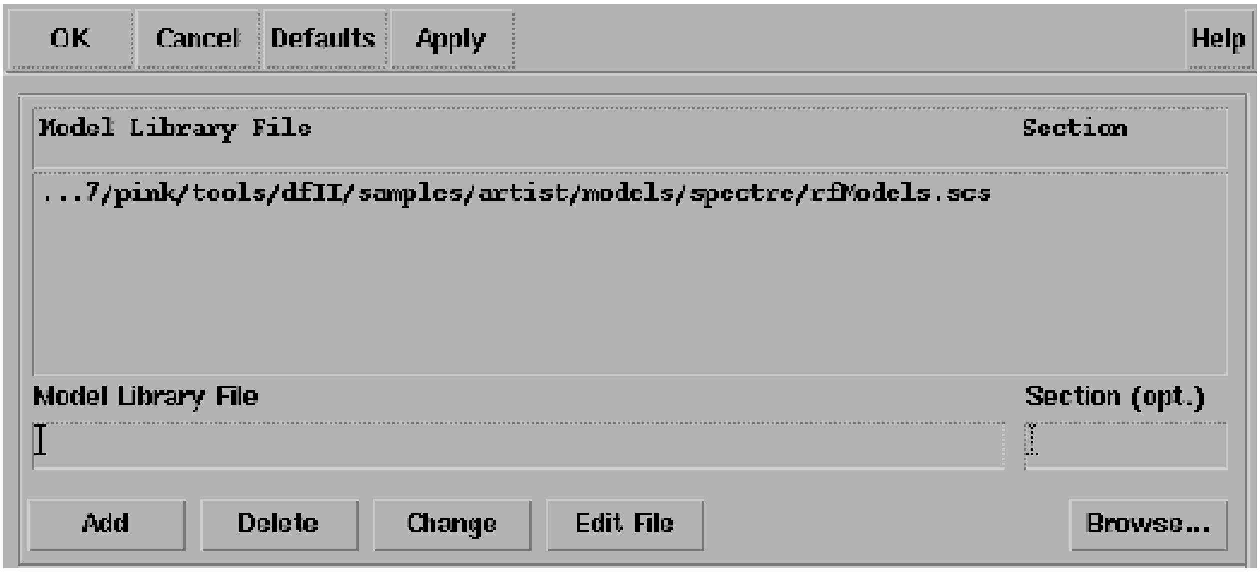
**1.** In the ADE window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

**2.** In the Model Library File field, type the full path to the model file including the file name, *rfModels.scs*.

**3.** Click *Add*.

The completed form looks like this.




4. Click *OK*.

# Fundamental Frequency, Output Noise, and Phase Noise with PSS and Pnoise

This example computes the fundamental frequency, output noise, and phase noise for the oscDiff differential oscillator circuit. You perform a PSS analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the fundamental frequency, output noise, and phase noise.

## Setting Up the Simulation

1. If necessary, open the *oscDiff* circuit.
2. If necessary, specify the full path to the model files.
3. If necessary, in the ADE window choose *Analyses – Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

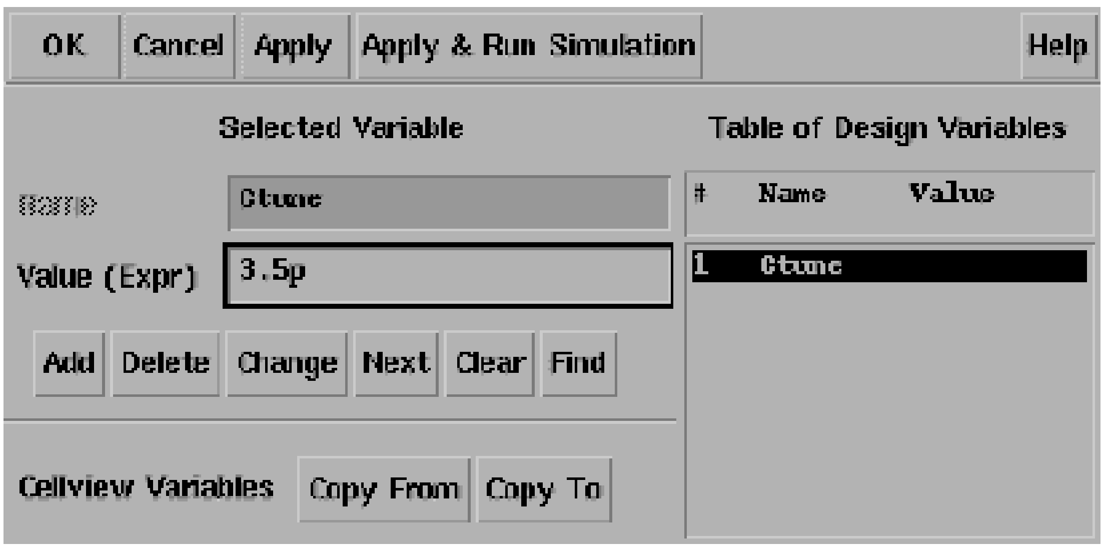
## Editing Design Variables

1. In the ADE window, choose *Variables – Edit*.

The Editing Design Variables form appears. For this example, you use this form to set the value of the variable *Ctune*.

2. In the Table of Design Variables list box, highlight Ctune.
3. Type 3.5p in the Value (Expr) field.
4. Click *Change*.

   The completed form looks like this.




5. Click *OK*.

   *Ctune* appears in the *Design Variables* section of the ADE window with its modified value

## Setting Up the PSS Analysis

**Note:** If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this PSS analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the analysis.

The form changes to display options needed for the PSS analysis. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

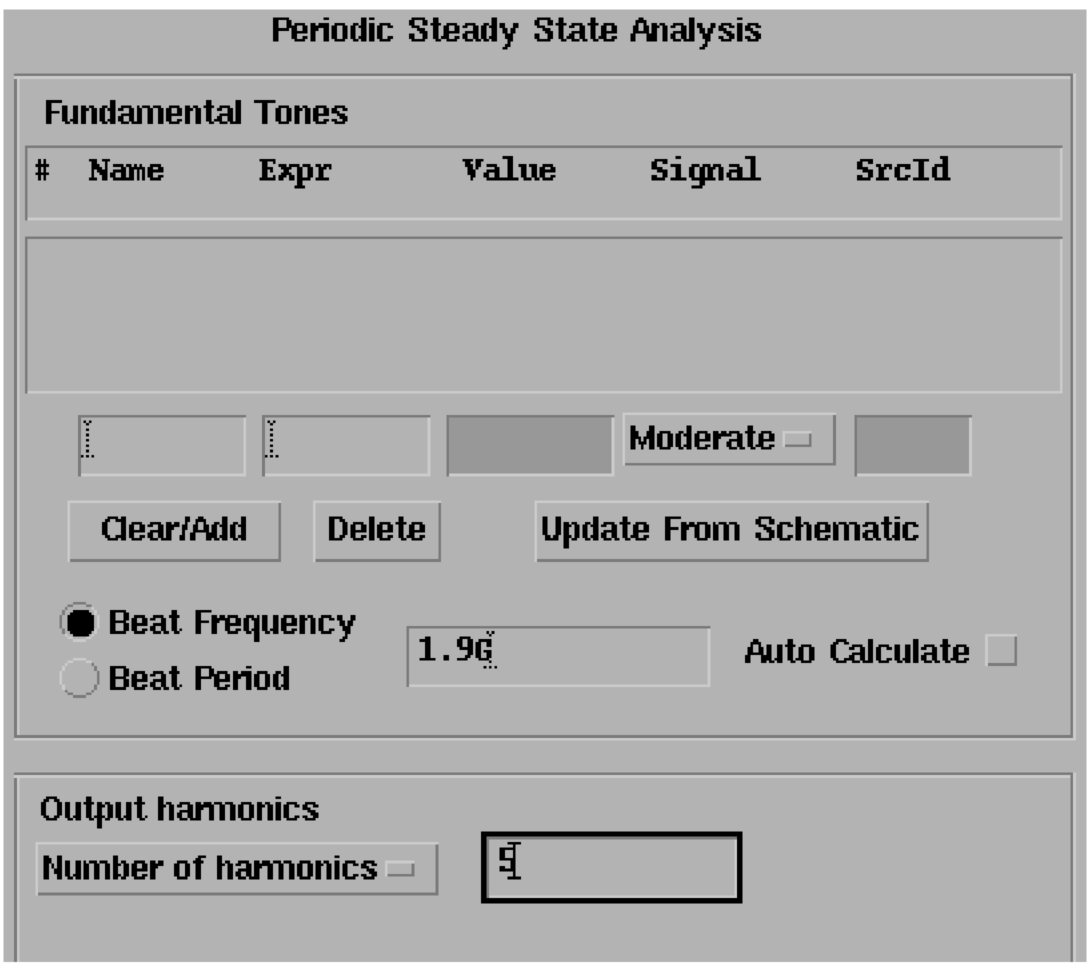
3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

4. In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   Type `1.9G` in this example.

5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `5` in the field.

6. The top of the Periodic Steady State Analyses form appears below.




7. Highlight *moderate* for the *Accuracy Default (errpreset)*.

8. Type `100.5n` in the *Additional Time for Stabilization (tstab)* field.

   You can use a slightly different value as long as it is above 100n.

9. If you want to save the results of the initial transient analysis, set *Save Initial Transient Results (saveinit)* to *yes*.

10. Highlight *Oscillator*.

    The form changes to let you specify the two nodes needed for oscillator simulation.

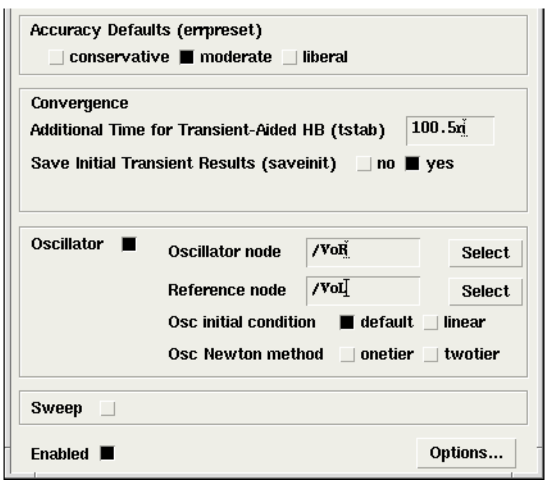
11. Click the *Select* button next to the *Oscillator node* field. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled *VoR*.

    `/VoR` appears in the *Oscillator node* field.

12. Click the *Select* button next to the *Reference node* field. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node labeled *VoL*.

    `/VoL` appears in the *Reference node* field. Because this is a differential oscillator, the reference node is not `/gnd!` as it was in the *tline3oscRF* example.

    The bottom of the Choosing Analyses form for PSS looks like the following.




Verify that *Enabled* is highlighted for the PSS analysis.

## Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter 1 in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value 2K away from the fundamental frequency.

3. In the *Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `1` and `100M` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

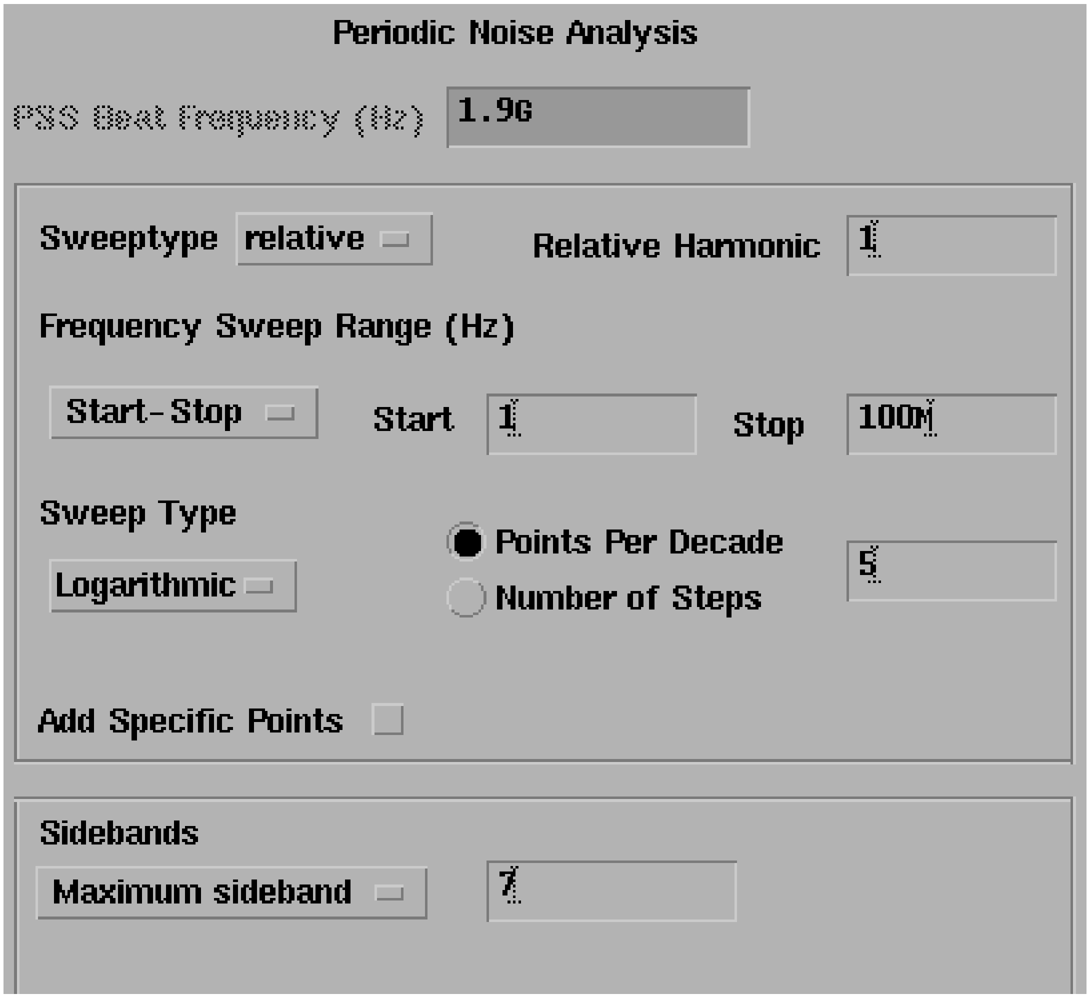
5. Highlight *Points Per Decade* and type `5` in the field.

   **Note:** Be sure to use a nonzero starting value for logarithmic sweeps.

6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `7` in the *Sidebands* field. The default value is 7.

   Be sure to use a nonzero *Maximum sideband* value.

The top of the Choosing Analyses form for the `Pnoise` analysis looks like this.

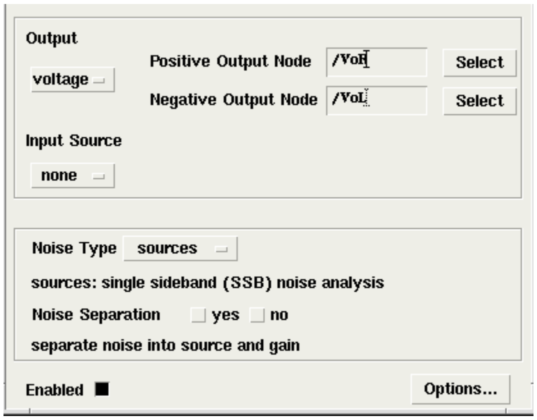



7. In the *Output* cyclic field, choose *voltage*.
8. Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoR*.

   `/VoR` appears in the *Positive Output Node* field.
9. Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoL*.

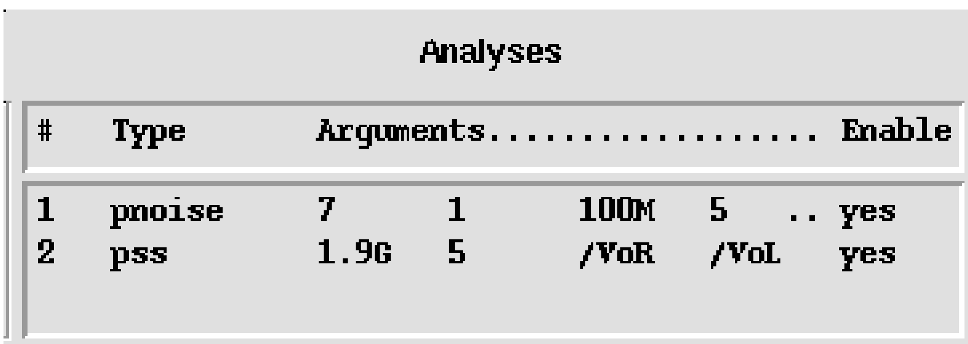
   `/VoL` appears in the *Negative Output Node* field. Because this is a differential oscillator, the negative output node is not `/gnd!` as it was for the *tline3oscRF* oscillator.
10. In the *Input Source* cyclic field, choose *none*.

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.




**11.** In the Choosing Analyses form, verify that the *Enabled* field at the bottom of the Pnoise form is highlighted. Then click *OK* at the top of the Choosing Analyses form.

The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.




## Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Fundamental Frequency

To plot the fundamental frequency, do the following steps.

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Harmonic Frequency* for *Function*.

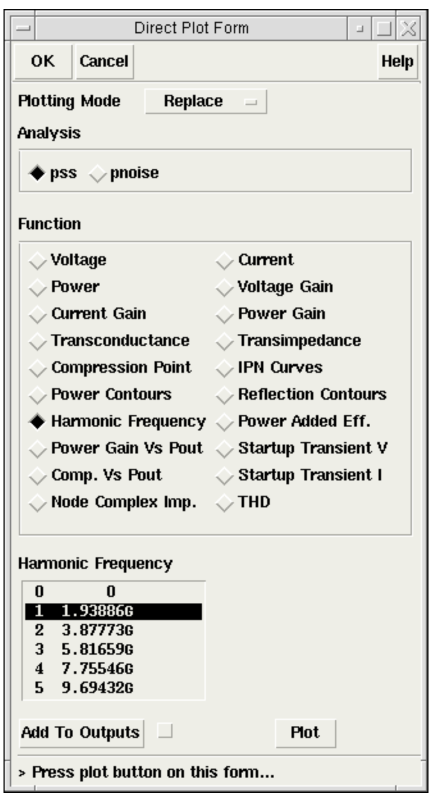
   The Direct Plot form changes to display available frequencies in the *Harmonic Frequency* list box.

5. Follow the prompt at the bottom of the Direct Plot form,

   *Select Harmonic Frequency on this form...*

   then highlight the first harmonic in the *Harmonic Frequency* list box.

   The Direct Plot form appears as follows.

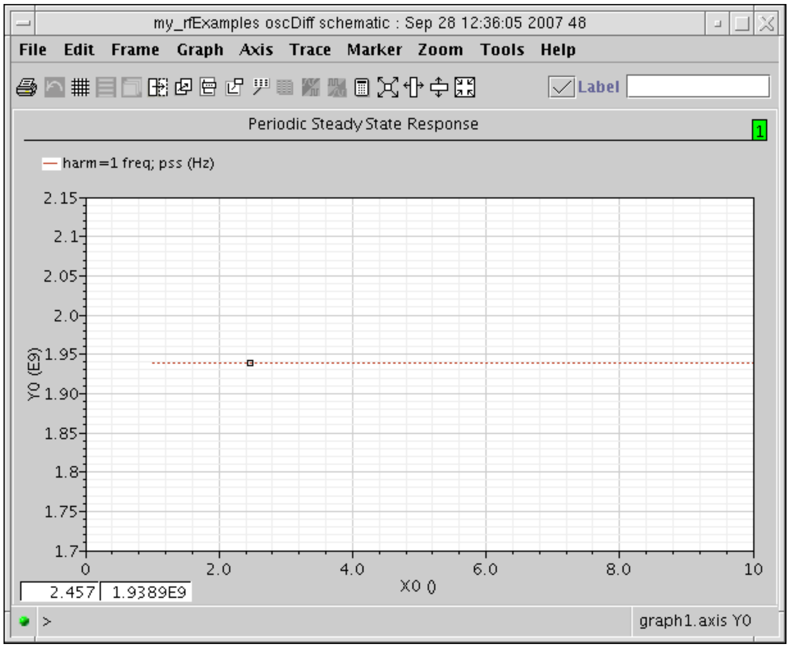



6. Follow the prompt at the bottom of the Direct Plot form,

   *Press plot button on this form...*

   then click *Plot*.

The waveform window displays the fundamental frequency.




Notice that the fundamental frequency value for the first harmonic of `1.939 G` is also displayed in the Harmonic Frequency list box in the Direct Plot form.

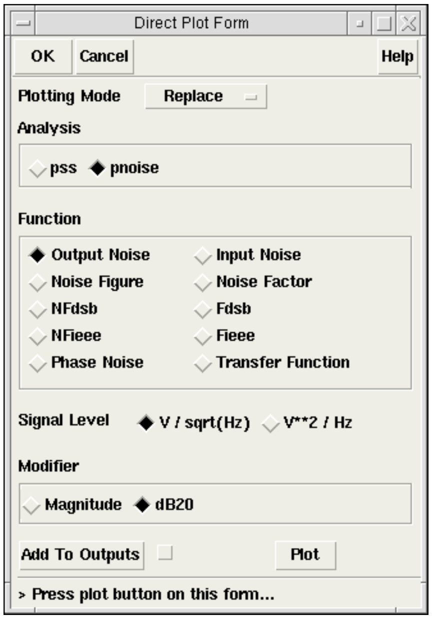
## Plotting the Output Noise and Phase Noise

If necessary, in the ADE window choose *Results - Direct Plot - Main Form* to open the Direct Plot form.

To plot the output noise, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pnoise* for *Analysis*.
3. Highlight *Output Noise* for *Function*.
4. Highlight *V/sqrt(Hz)* for *Signal Level*.

**5.** Highlight *dB20* for *Modifier*.

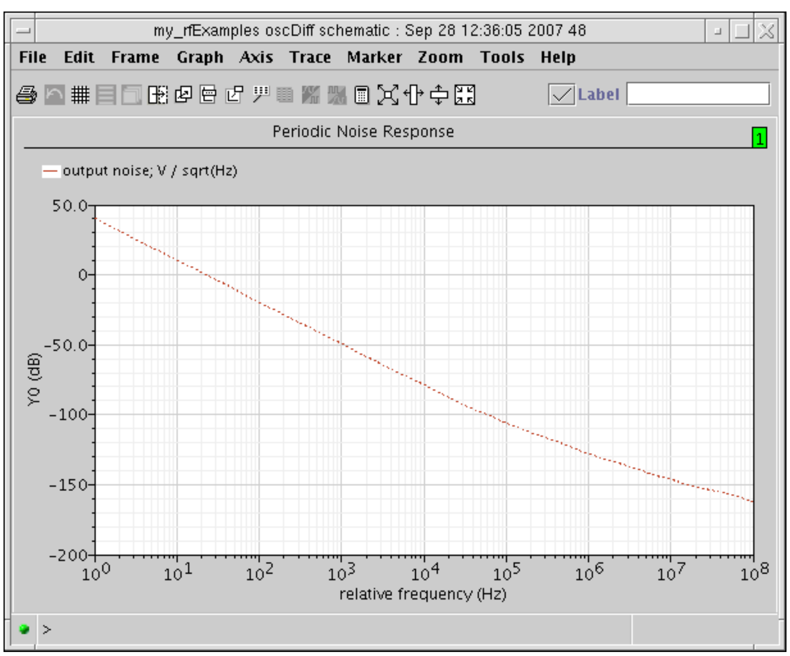



**6.** Follow the prompt at the bottom of the Direct Plot form,

*Press plot button on this form...*

then click *Plot*.

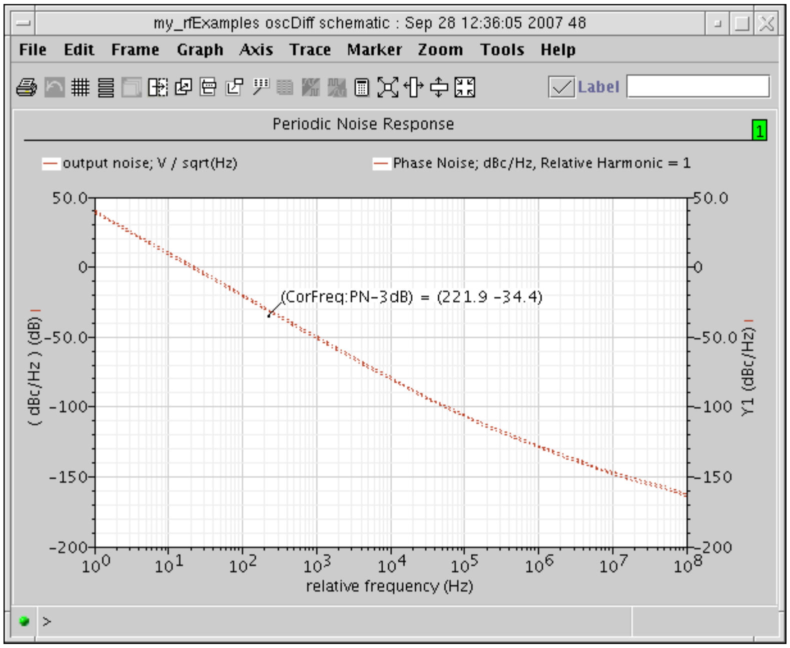
The output noise is plotted in the waveform window.



To plot phase noise on the same graph with output noise, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Phase Noise* for *Function*.
3. Click *Plot*.

   The phase noise plot is added to the output noise plot.

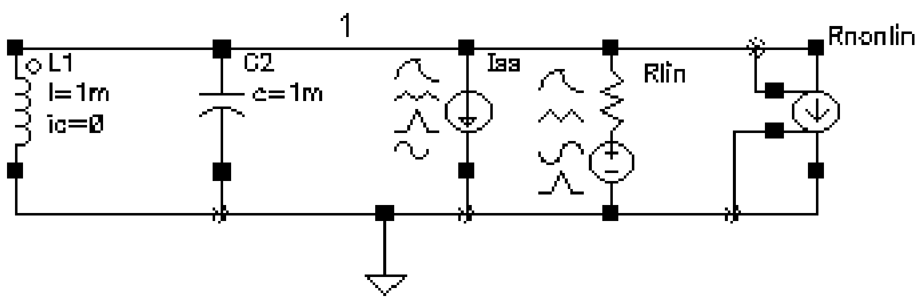



By plotting the phase noise on the same graph as the output noise, you can see the linear relationship between the output noise and the phase noise. The output noise is scaled by the carrier amplitude to produce the phase noise value.

# The Van der Pol Circuit: Measuring AM and PM Noise Separation

As a simple example, consider the `vdp_osc` circuit, a feedback amplifier circuit shown in Figure 6-3. This example separates oscillator noise into AM and PM components. It also calculates USB and LSB noise for the oscillator.

**Figure 6-3 Schematic for the vdp_osc Feedback Amplifier Circuit**




The `vdp_osc` is a parallel RLC circuit with a nonlinear transconductance represented by a polynomial voltage controlled current source. At small capacitor voltages, the transconductance is negative; that is, it is an active device which creates positive feedback that seeks to increase the voltage on the capacitor. At larger capacitor voltages, where the transconductance term goes into compression, it effectively acts as a positive resistor (that is, it creates negative feedback) and limits the voltage. You use a cubic polynomial model for the circuit.

# Simulating the vdp_osc Circuit

Before you begin, be sure that you have performed the setup procedures described inChapter 4, "Setting Up for the Examples."

## Opening the vdp_osc Circuit

1. In the Command Interpreter Window (CIW), choose *File – Open*.

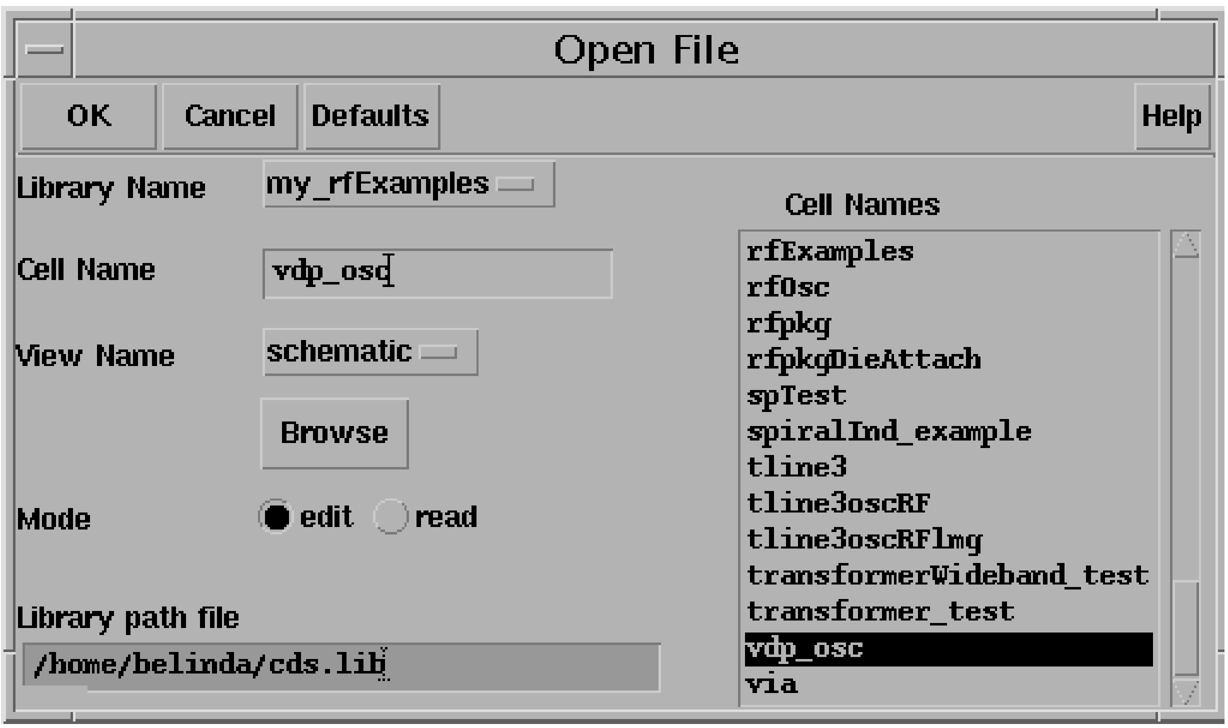
   The Open File form appears. Filling in the Open File form opens the vdp_osc schematic.

2. For *Library Name*, choose *my_rfExamples* or whatever you called the editable copy of the *rfExamples* library. See Chapter 4, "Setting Up for the Examples" for more information.

3. Choose *schematic* for *View Name*.
4. In the *CellNames* list box, highlight *vdp_osc*.

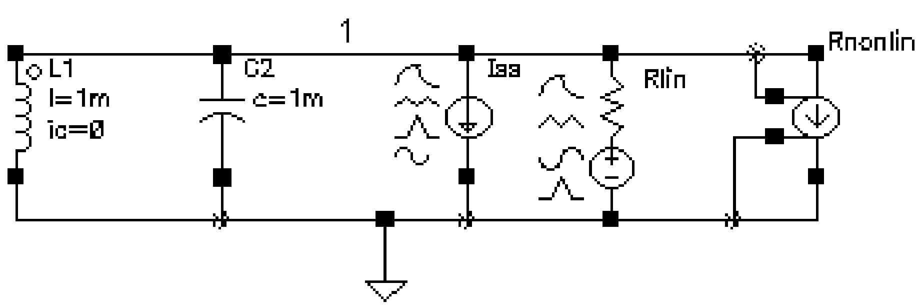
   *vdp_osc* appears in the Cell Name field.
5. Highlight *edit* for *Mode*.

   The completed Open File form appears like the one below.




6. Click *OK*.

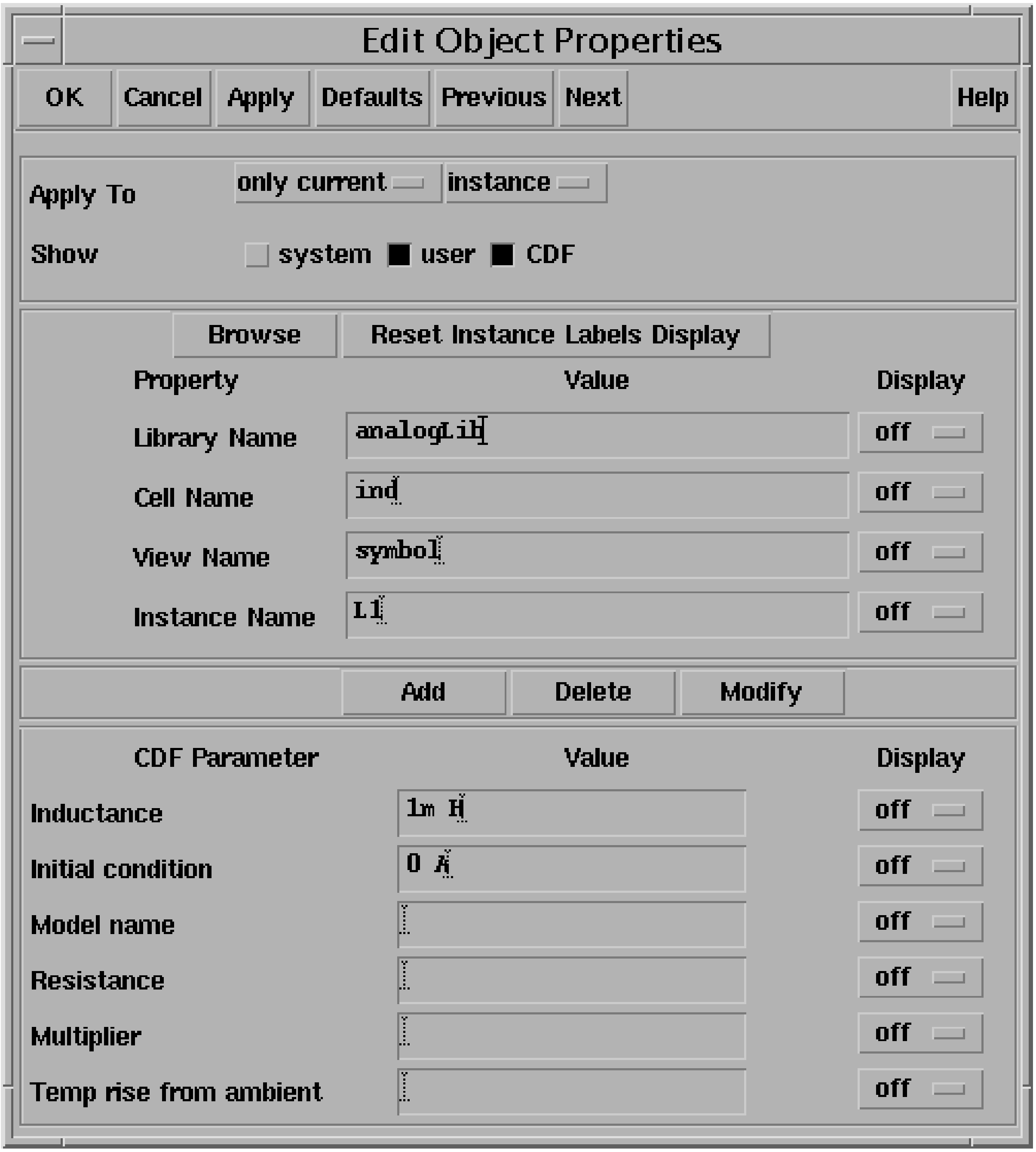
   The Schematic window for the vdp_osc oscillator opens.



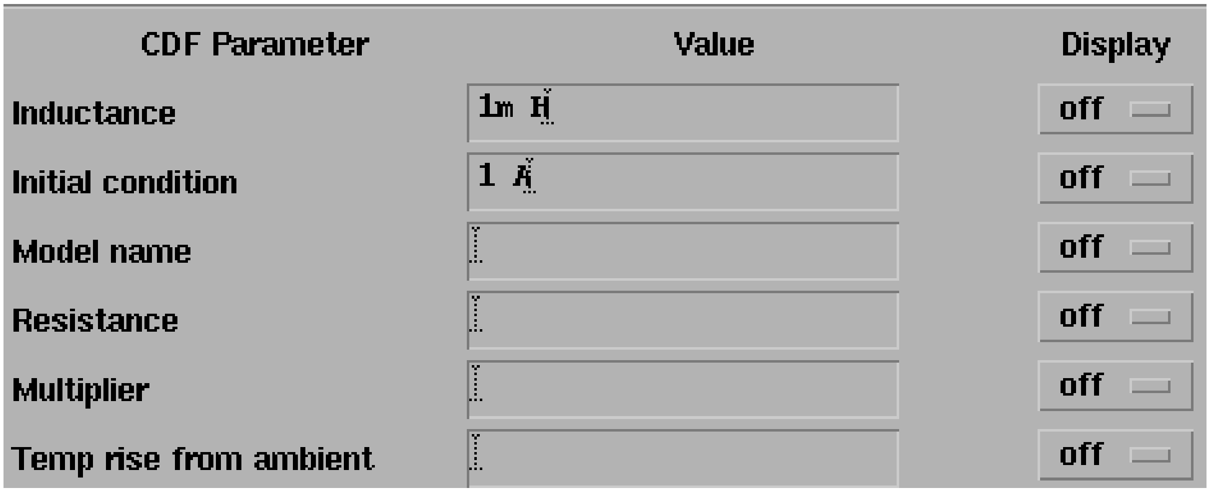
## Editing Properties for the Inductor

1. In the Schematic window, select the L1 inductor.
2. With the L1 inductor selected, choose *Edit-Properties-Objects*.

   The Edit Object Properties form displays for the L1 inductor.



3. Edit the *Initial condition* value by replacing the `0` with `1`.

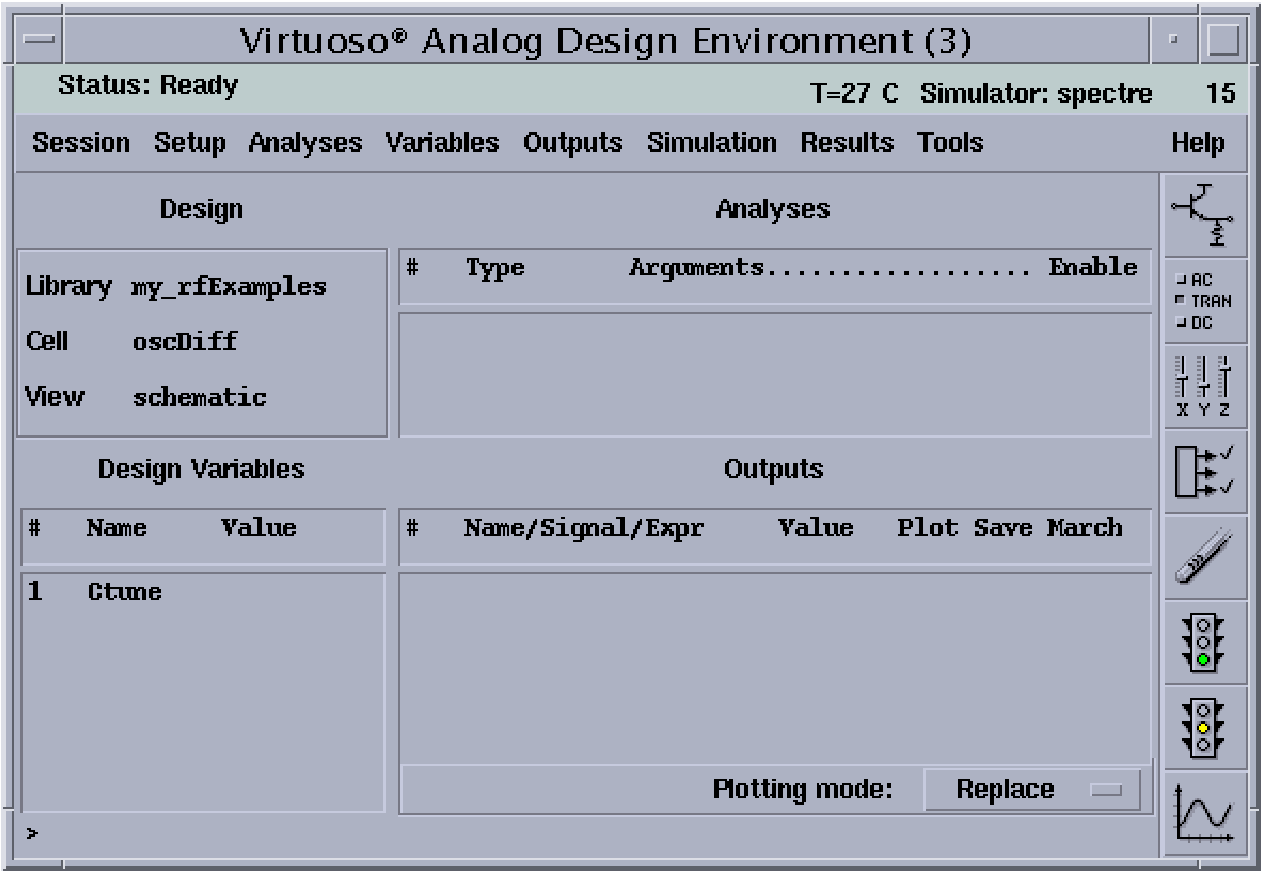



4. Click *OK* in the Edit Object Properties form.
5. Choose *Design – Check and Save* in the Schematic window.

## Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *vdp_osc* in the Choosing Design form.

# Measuring AM and PM Conversion with PSS and Pnoise

This example computes AM and PM conversion for the `vdp_osc` oscillator circuit. You perform a PSS analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the AM and PM noise components from the results of the PSS and `Pnoise` analyses.

## Setting Up the Simulation

**1.** If necessary, open the vdp_osc circuit.

2. If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

## Setting Up the PSS Analysis

**Note:** If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this PSS analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis (see step 3).

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the *Analysis*.

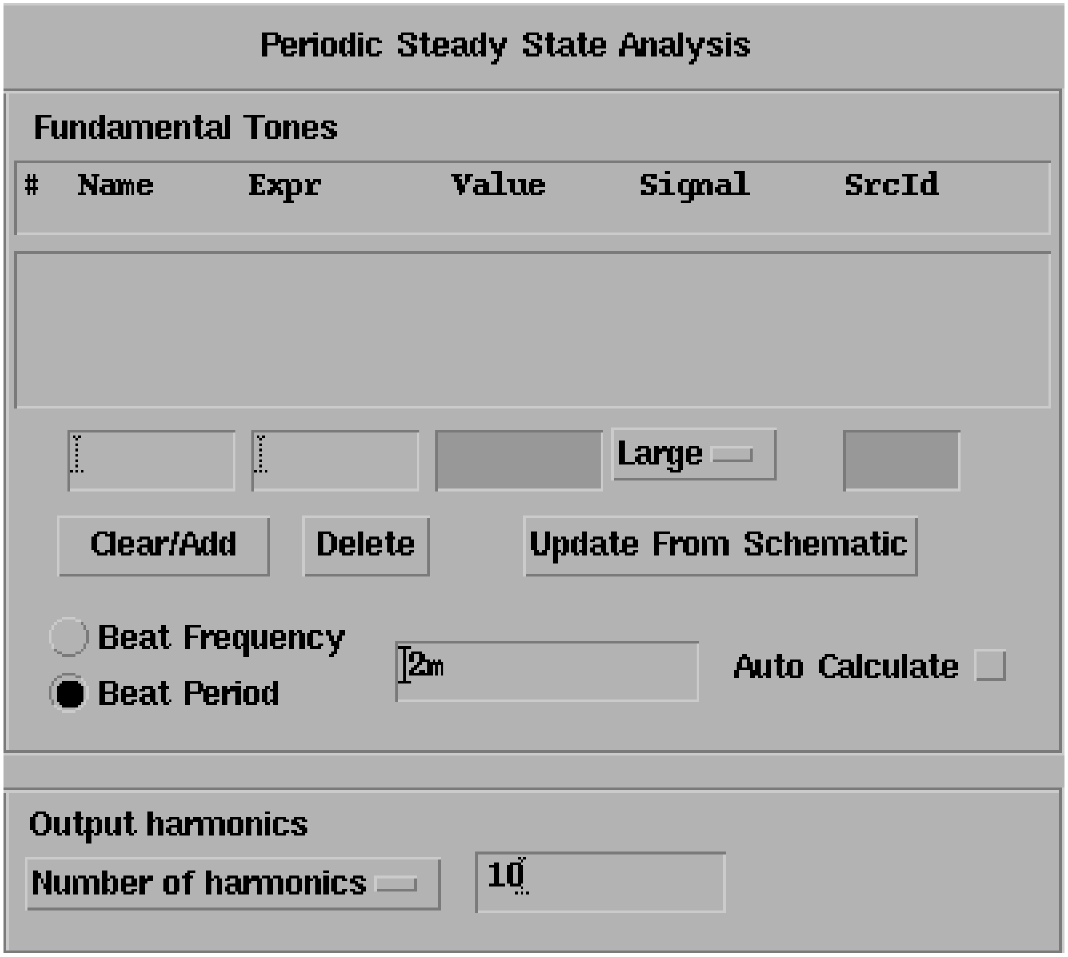
   The form changes to display options needed for the PSS analysis. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

3. In the *Fundamental Tones* area, highlight *Beat Period.* Be sure the *Auto Calculate* button is *not* highlighted.

4. In the field next to *Beat Period*, type `2m`. This is an estimate based either on the results of a prior transient analysis or quick manual calculations from LC oscillator equations.

5. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `10` in the *Number of harmonics* field.

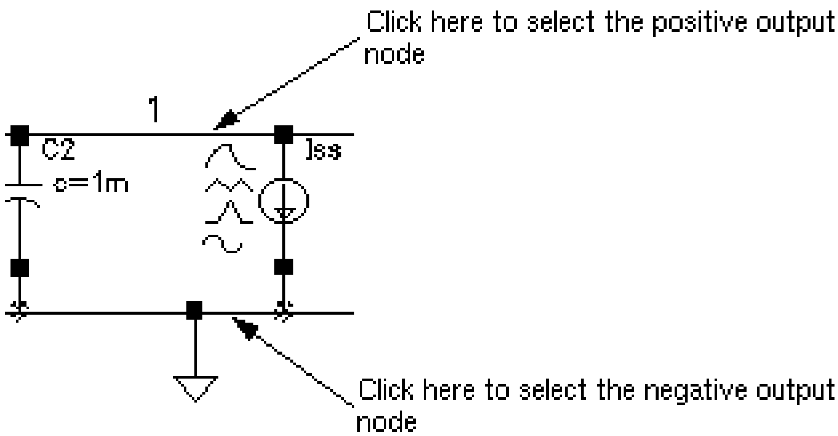
The top of the PSS Choosing Analyses form appears below.




6. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
7. In the *Additional Time for Stabilization field (tstab)*, type `700n`.
8. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.

**9.** Click the *Oscillator node Select* button. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled `1`.



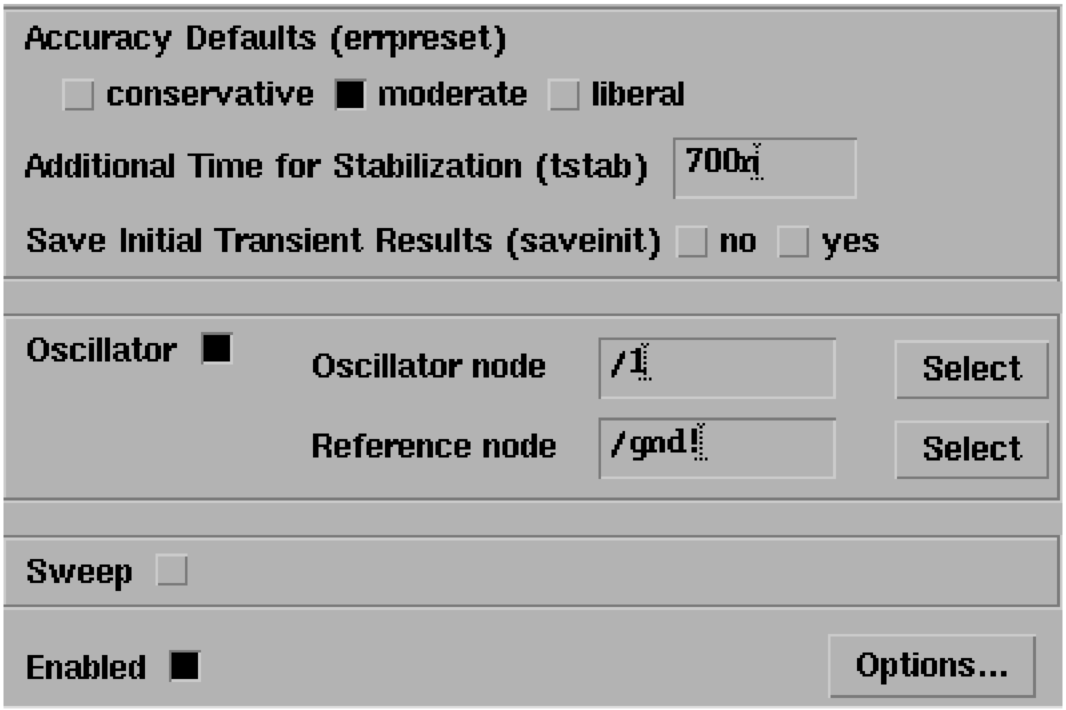

`/1` appears in the *Oscillator node* field.

**10.** Click the *Reference node Select* button. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node where the ground is located.

**11.** `/gnd!` appears in the *Reference node* field.

You can also leave the *Reference node* field empty as the *Reference node* field defaults to `/gnd!`.

The bottom of the Choosing Analyses form for PSS looks like the following.




**12.** Verify that *Enabled* is highlighted for the PSS analysis and click *Apply*.

Correct any errors reported for the PSS analysis after you click *Apply*.

## Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify 2K, you choose a value 2K away from the fundamental frequency. This example uses 1, the first harmonic, because this is the best way to present phase noise for an oscillator which shows up next to the oscillator's fundamental frequency.

3. In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `5` and `10k` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `501` in the *Number of Steps* field.

   **Note:** Be sure to use a nonzero starting value for a logarithmic sweep.

   This example uses 501 steps, which allows you to see all the small features and get very accurate noise calculations. Because this is a very small circuit, we can use any number of steps. Your circuit might require a different number of steps.

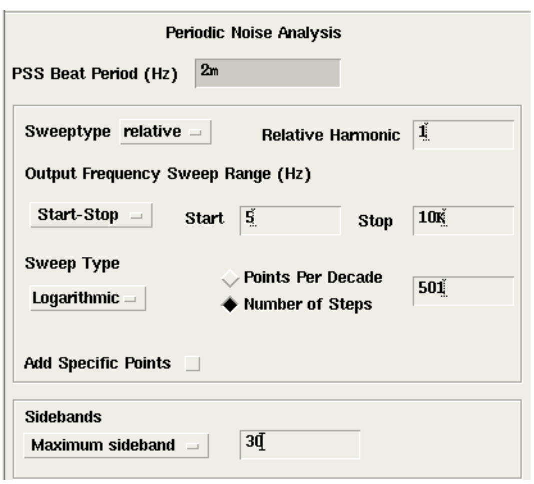
   In an actual circuit, it is typically more efficient to start off by setting *Number of Steps* to 2 or 3, as you iterate through the simulations needed to determine an appropriate value for the *Maximum sideband*. After that value is determined, you can set the *Number of Steps* to whatever value you need to obtain an accurate noise calculation.

6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `30` in the *Sidebands* field.

   Be sure you always use a nonzero *Maximum sideband* value. The default value is `7`. A large *Maximum sideband* value provides more accurate noise calculations by allowing for more harmonic folding. (In this small circuit, however, the large *Maximum sideband* value provides no advantage.)

   In an actual circuit, you determine an appropriate value for *Maximum sideband* by experimentation, gradually increasing the value in a series of simulations until the change in additional accuracy diminishes to an amount that satisfies your requirements. As noted in step 5, you can run this series of simulations with a reduced *Number of Steps* and, after determining the *Maximum sideband*, increase the *Number of Steps* to obtain an accurate noise calculation.

   The top of the Choosing Analyses form for the Pnoise analysis looks like this.

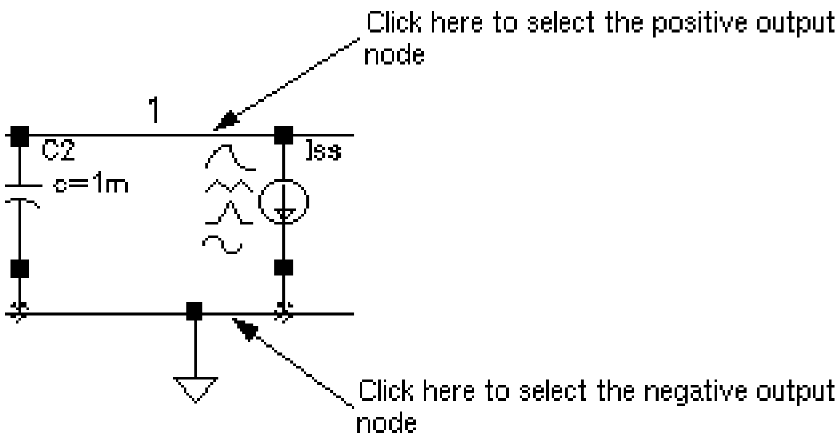



7. In the *Output* cyclic field, choose *voltage*.

8. Notice the informational message that displays in the CIW.

```
*info*  Since voltage was selected as the output measurement
        technique for the pnoise analysis, spectre will NOT
        subtract any load resistance contribution from the Noise
        Figure calculation. If this is undesirable, please select
        probe as the output measurement technique and select a port.
```

**9.** Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose */1*.




`/1` appears in the *Positive Output Node* field.

**10.** Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose the ground.

`/gnd!` appears in the *Negative Output Node* field.

**11.** In the *Input Source* cyclic field, choose *none*.

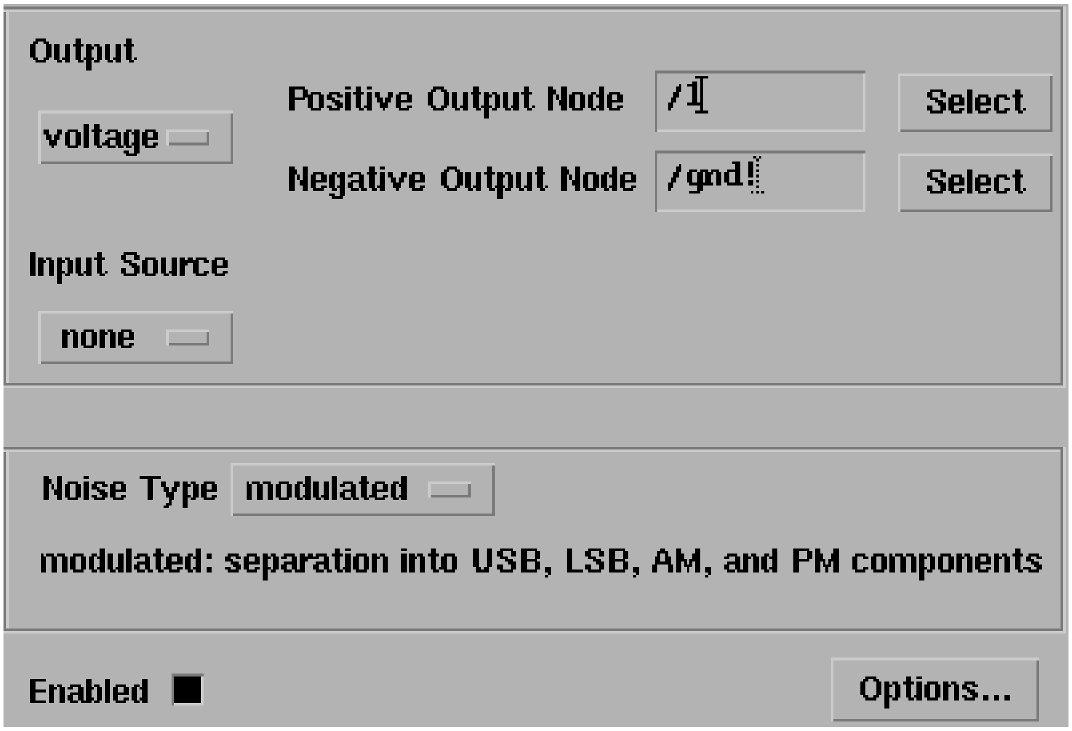
In this case, because we are calculating Total Noise and its AM and PM components, the Input Source is not needed. (Using an Input Source generates NF, IRN, and other parameters.) Input Source is not usually used in oscillator simulations.

**12.** In the *Noise Type* cyclic field, choose *modulated*.

The following text appears below the *Noise Type* cyclic field.

*modulated: separation into USB, LSB, AM, and PM components*

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.



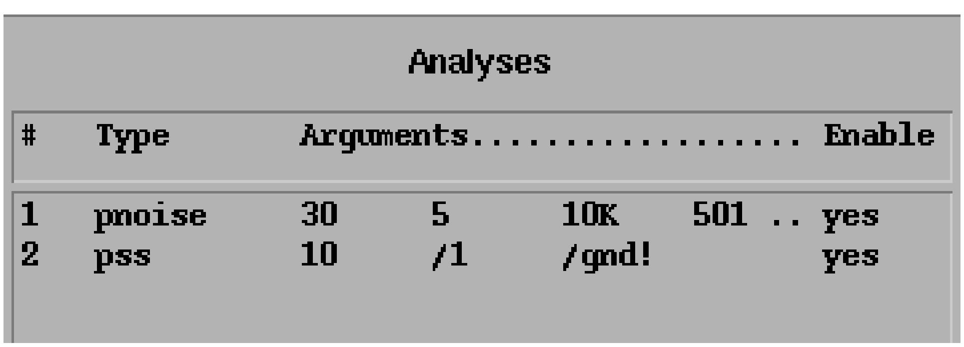

**13.** Verify that the *Enabled* field for the `Pnoise` analysis is highlighted. Then click *Apply* at the top of the Choosing Analyses form.

**14.** Correct any errors reported for the `Pnoise` analysis after you click *Apply*.

**15.** Click *OK.*

The Choosing Analyses form closes.

The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.



## Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

### Plotting Modulated Pnoise with the dBc Modifier

The dBc modifier on the Direct Plot form plots the noise power in decibels, relative to the carrier, on the y axis.

To plot the modulated pnoise, you start by plotting USB noise and then plot AM and PM noise in the same waveform window.

➤ If necessary, open the Direct Plot form. by choosing *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

### Plotting USB Noise

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise modulated* for *Analysis*.

   This selects the results of the modulated pnoise analysis for display.

3. Highlight *USB* for *Noise Type*.

   This selects to plot the standard pnoise analysis results.

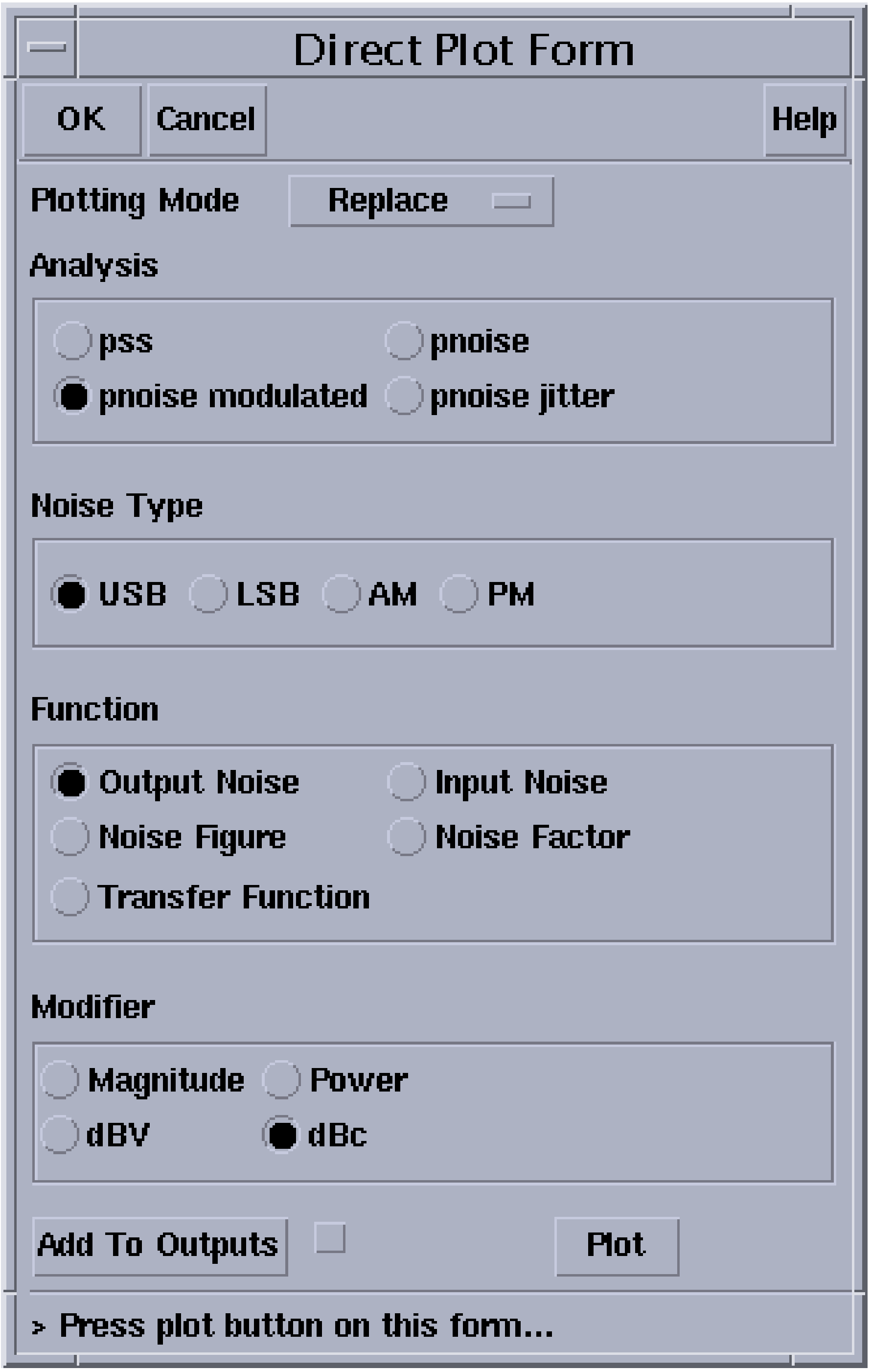
4. Highlight *Output Noise* for *Function*.

   This selects to plot Output noise.

5. Highlight *dBc* for *Modifier*.

   Selecting dBc plots the noise power in decibels, relative to the carrier, on the y axis.

The Direct Plot form appears as follows.

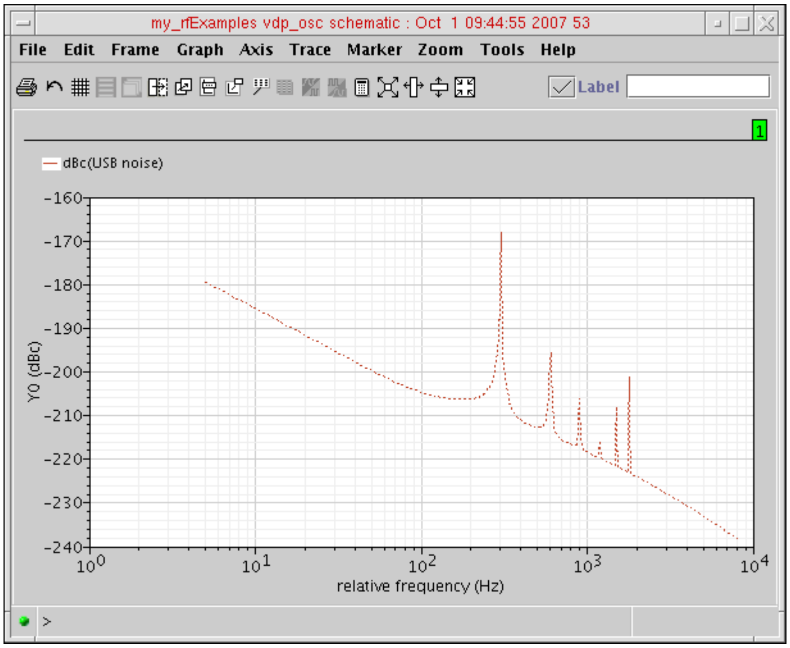



**6.** Follow the prompt at the bottom of the Direct Plot form

*Press plot button on this form...*

and click *Plot*.

The waveform window displays the USB noise function.




The upper sideband noise appears in the Direct Plot form.

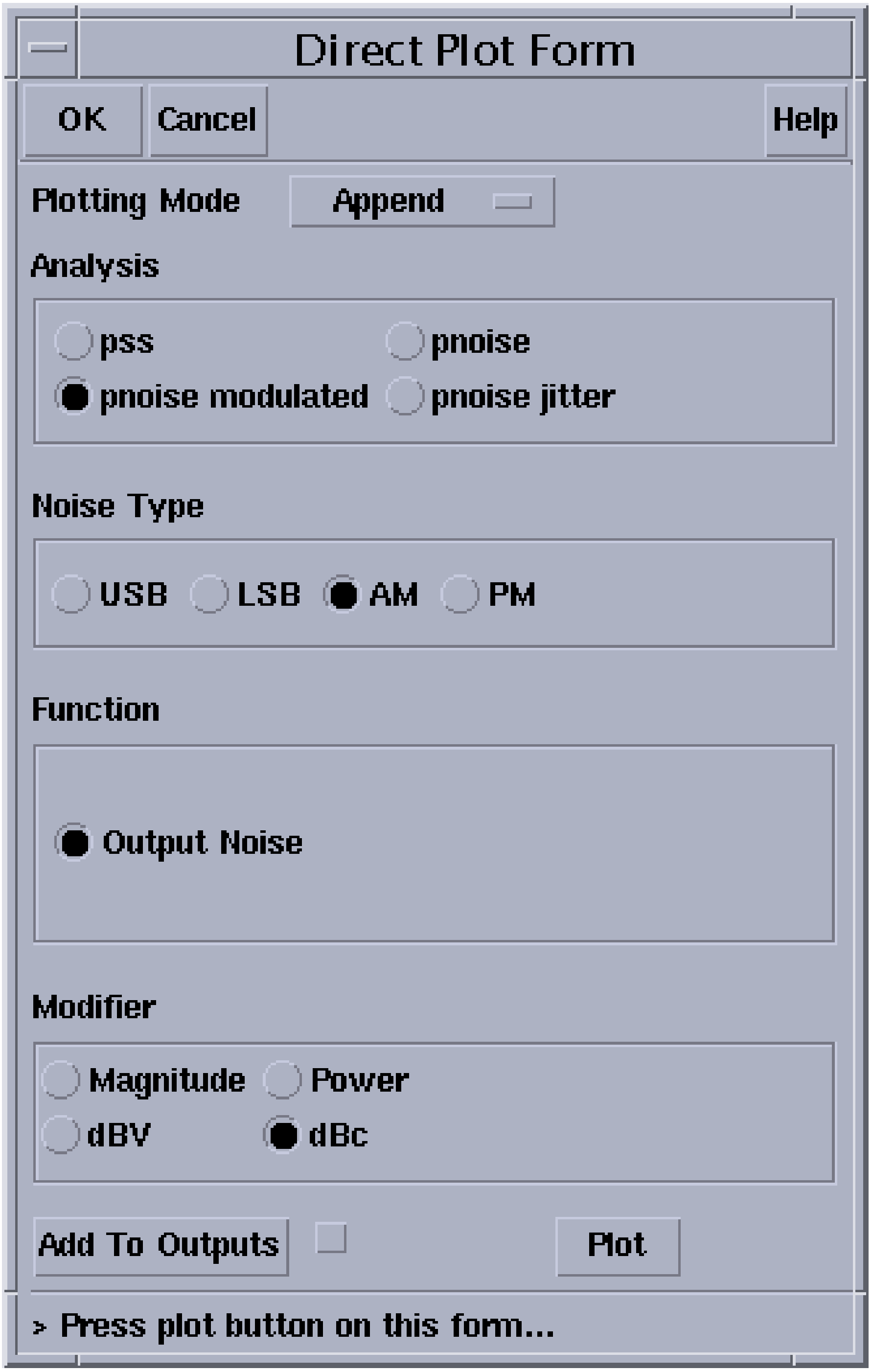
### Plotting AM Noise

Plot the AM noise in the same waveform window from the same Direct Plot form.

1. Highlight *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *AM* for *Noise Type*.

   When you highlight *AM* for *Noise Type*, all *Function* selections other than *Output Noise* go away.
4. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

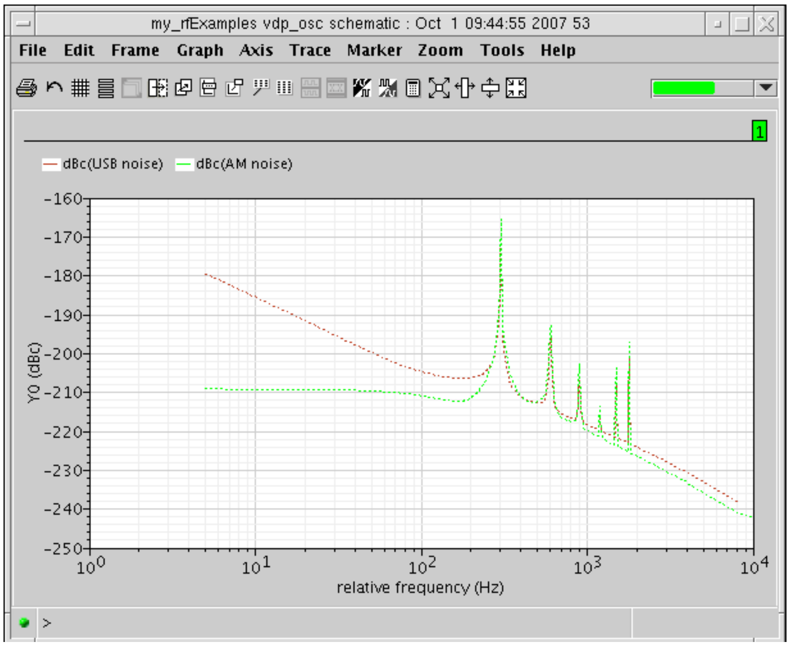



**5.** Follow the prompt at the bottom of the Direct Plot form

*Press plot button on this form...*

and click *Plot*.

The waveform window displays the USB and AM noise functions.

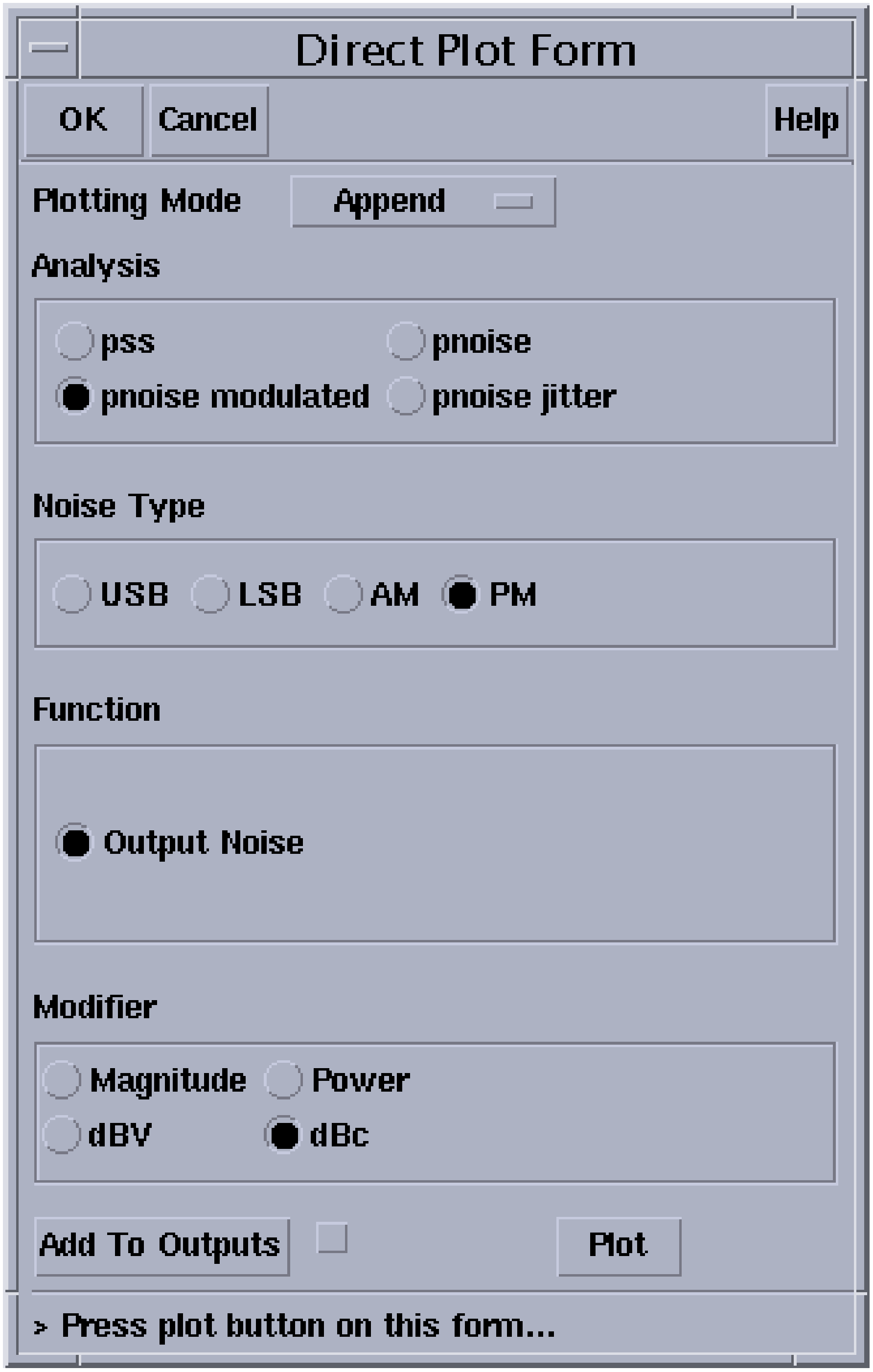



## Plotting PM Noise

Finally, plot the PM noise in the same waveform window from the same Direct Plot form.

1. Keep *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *PM* for *Noise Type*.
4. Keep *Output Noise* for *Function*.
5. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

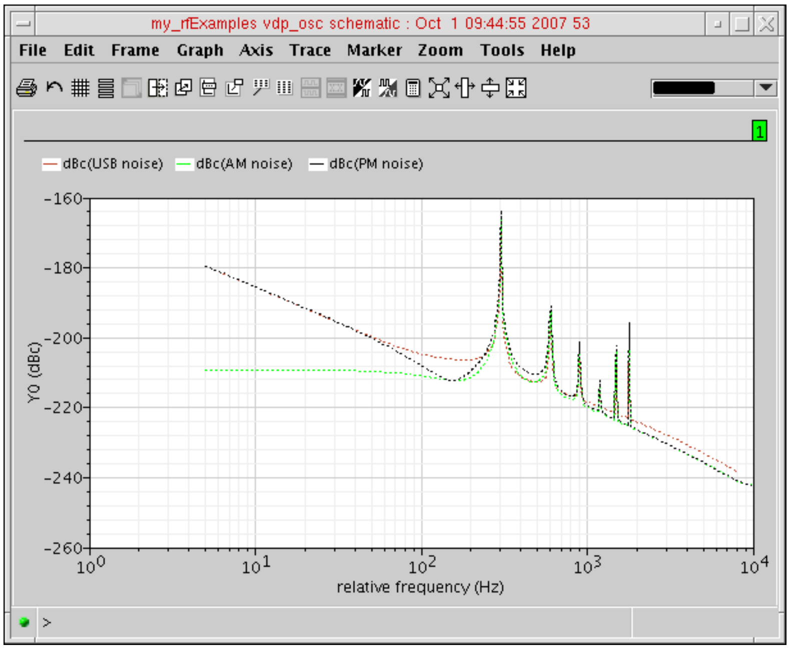



**6.** Follow the prompt at the bottom of the Direct Plot form

*Press plot button on this form...*

and click *Plot*.

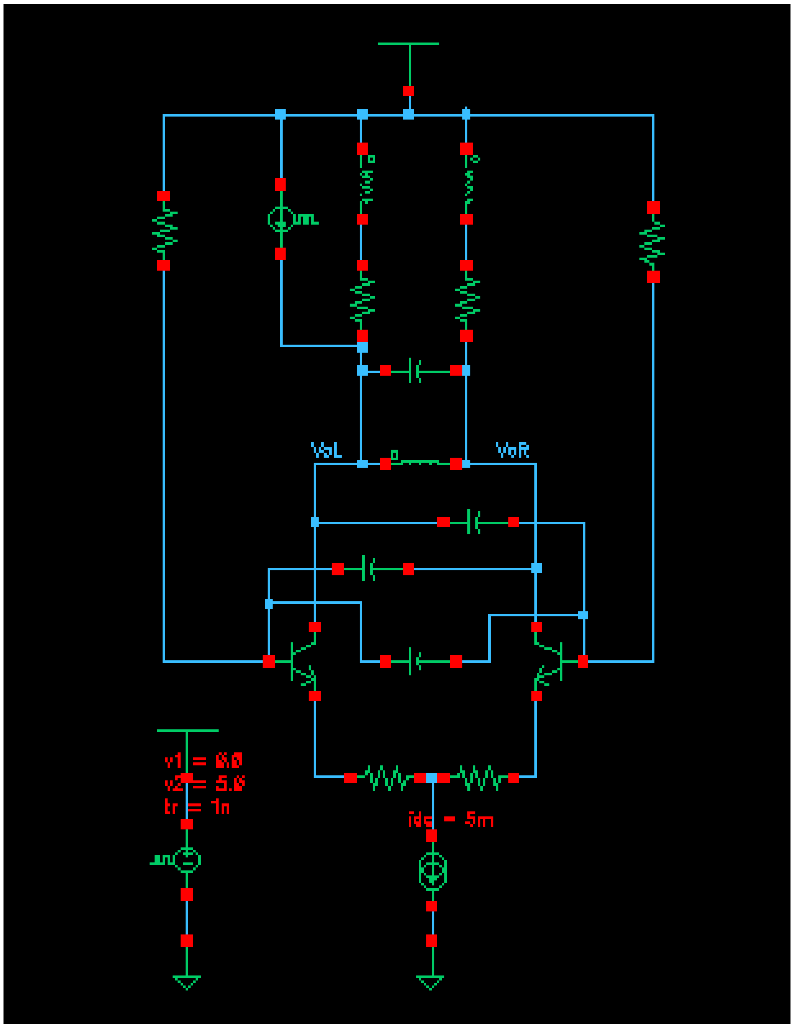
The waveform window displays the USB, AM, and PM noise functions.




# Measuring Jitter with PSS and Pnoise Jitter Analyses

As an example, consider *oscDiff*, the feedback amplifier circuit shown in Figure 6-4 on page 427.

**Figure 6-4 Schematic for the OscDiff Circuit**




Before you begin, be sure that you have performed the set-up procedures described in Chapter 4, "Setting Up for the Examples."

## Opening the oscDiff Circuit

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the oscDiff schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created. See Chapter 4, "Setting Up for the Examples" for more information.

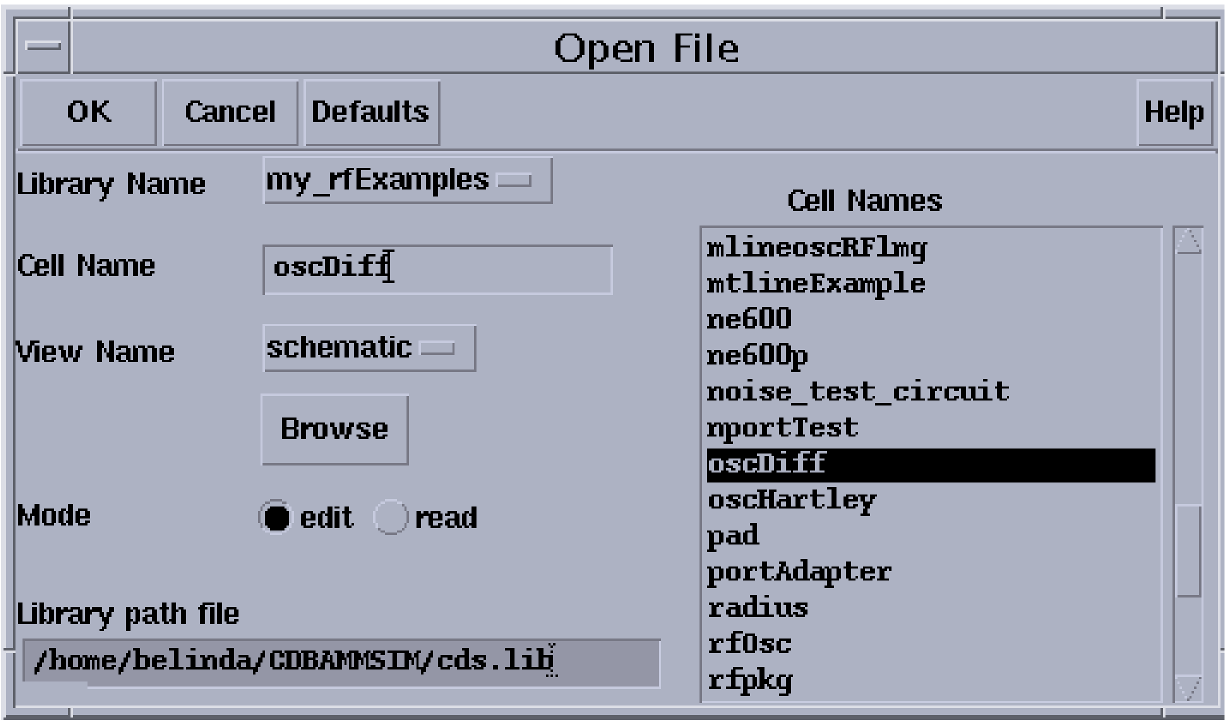
**3.** Choose *schematic* for *View Name*.

**4.** In the *CellNames* list box, highlight *oscDiff*.

*oscDiff* appears in the *Cell Name* field.

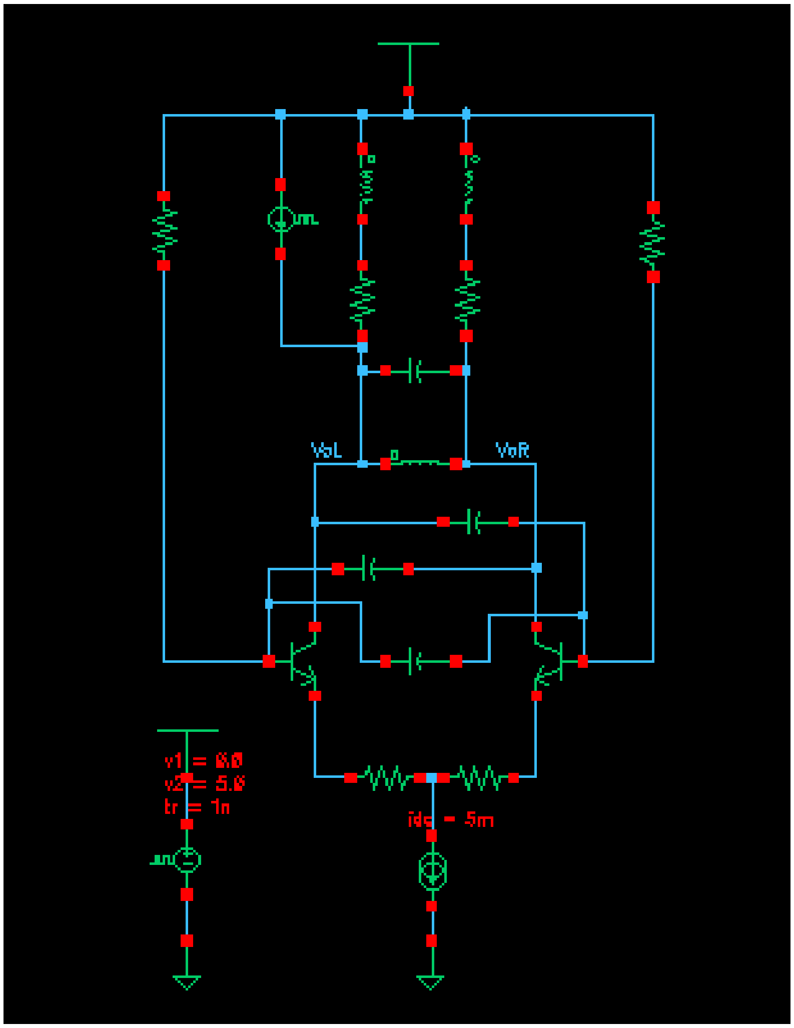
**5.** Highlight *edit* for *Mode*.

The completed Open File form appears like the one below.




**6.** Click *OK*.

The Schematic window for the `oscDiff` oscillator opens.

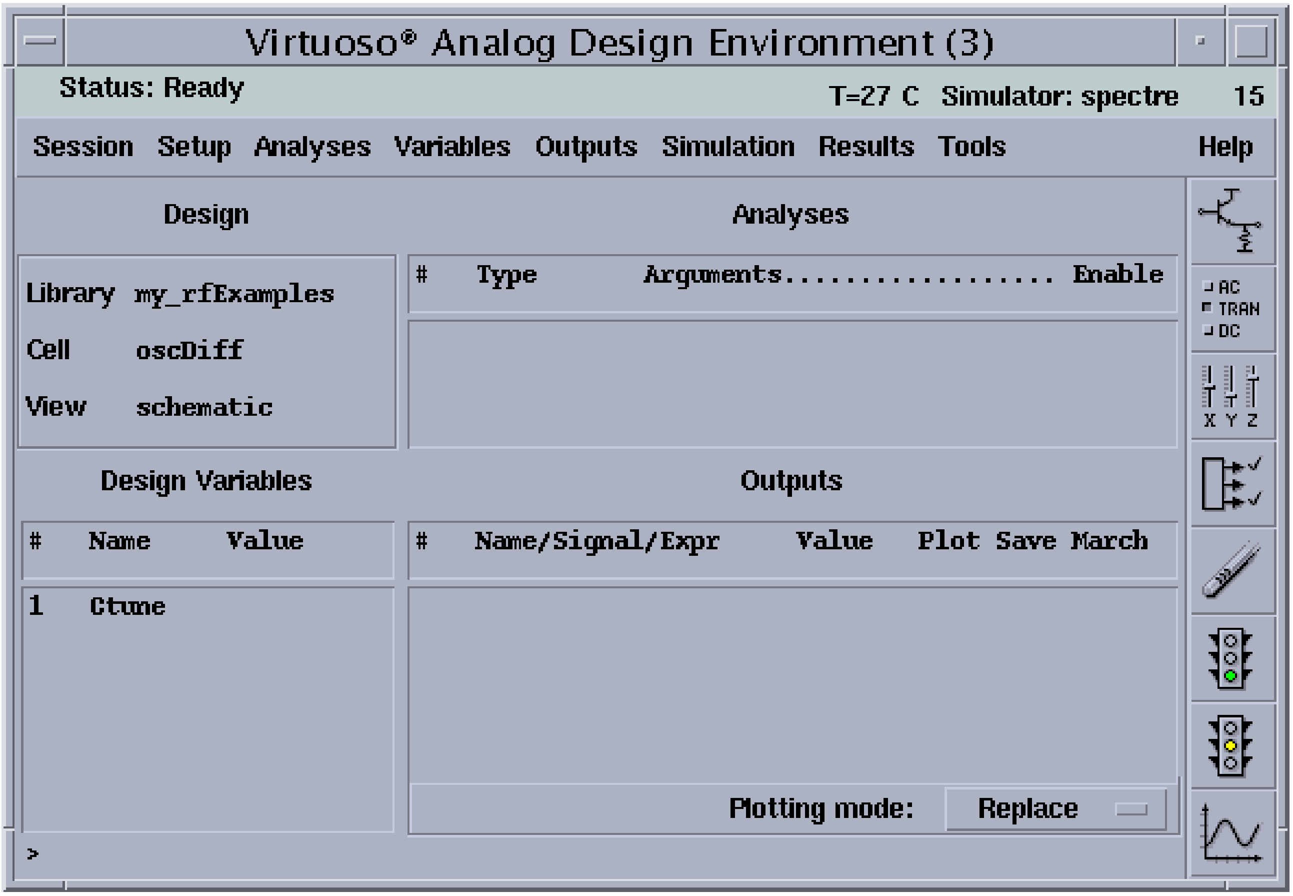



See “The oscDiff Circuit: A Balanced, Tunable Differential Oscillator” on page 392 for information about this circuit.

### Opening the Simulation Window

**1.** In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *oscDiff* in the Choosing Design form.

### Editing Design Variables

1. In the ADE window, choose *Variables – Edit*.

   The Editing Design Variables form appears. For this example, you use this form to set the value of the variable *Ctune*.

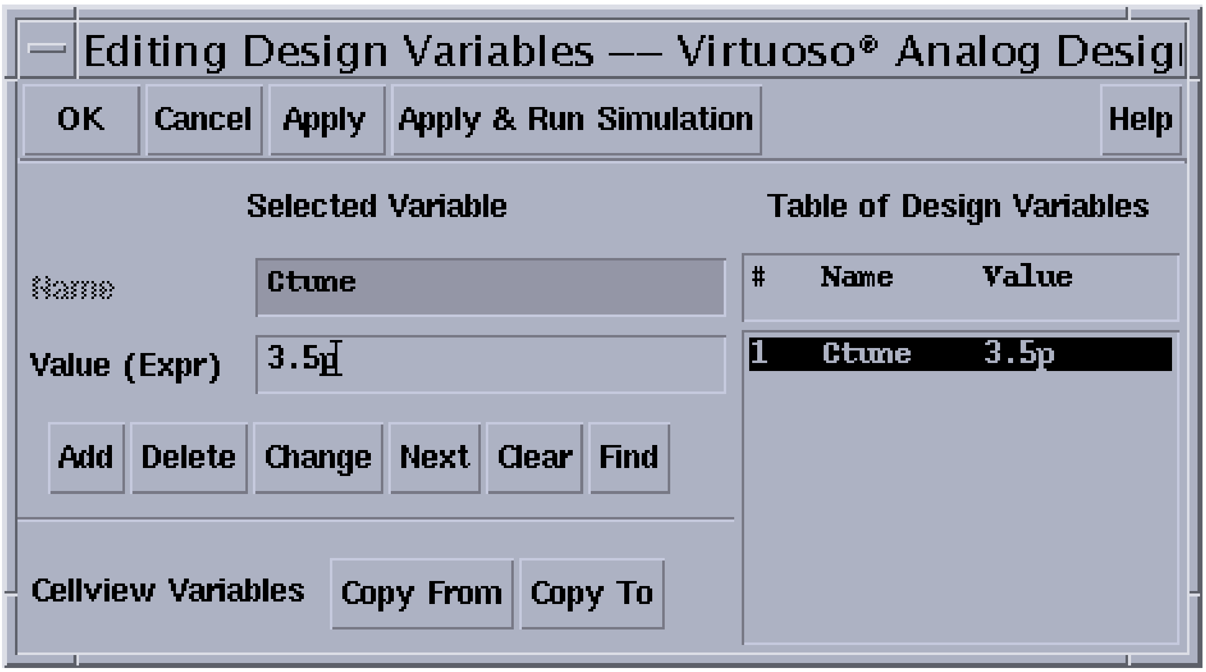
2. In the Editing Design Variables form, in the *Table of Design Variables* list box, highlight *Ctune*.

   *Ctune* displays in the *Name* field.

3. Type 3.5p in the *Value (Expr)* field.

4. Click *Change*.

   The completed form looks like this.




5. Click *OK*.

   *Ctune* appears in the *Design Variables* section of the ADE window with its modified value

## Setting Up the Simulation

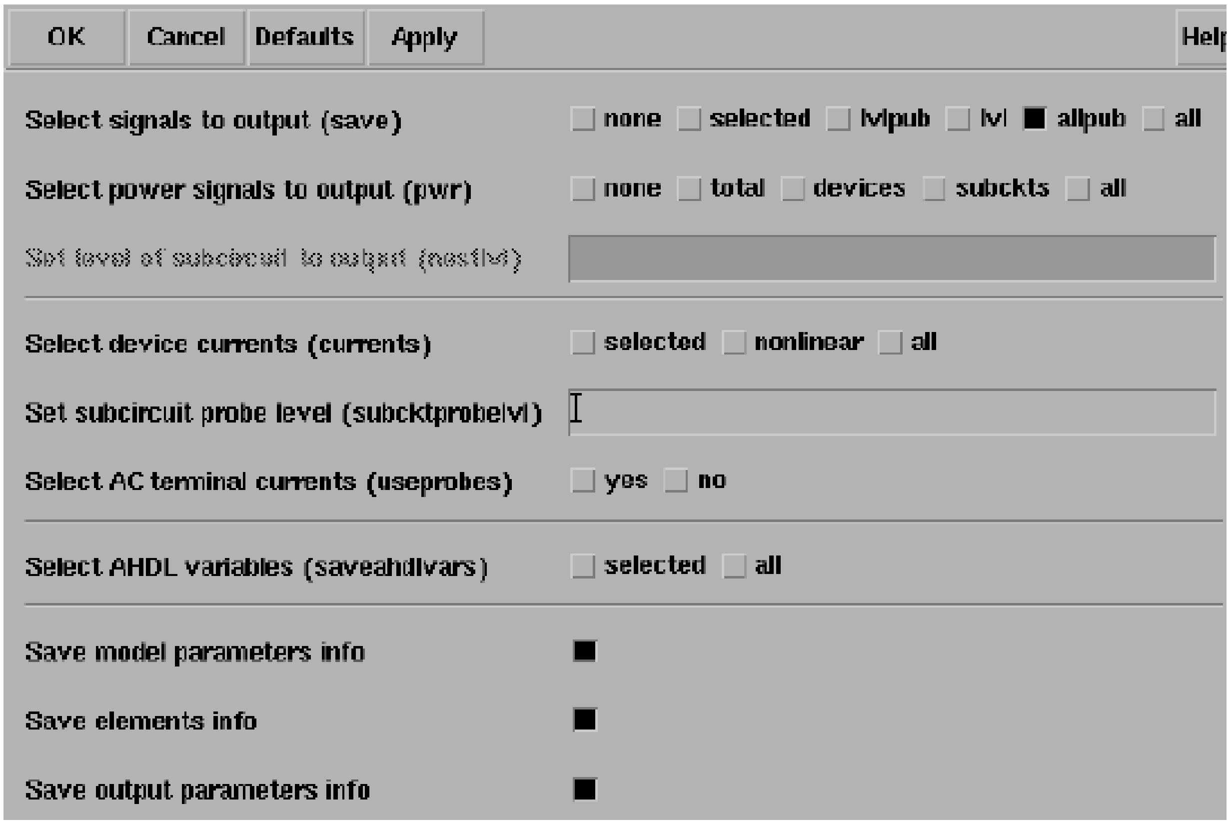
1. If necessary, see “Opening the oscDiff Circuit in the Simulation Window” on page 393.
2. If necessary, see “Choosing Simulator Options” on page 395.
3. If necessary, see “Setting Up Model Libraries” on page 396.
4. If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

## Specifying Outputs to Save

1. In the ADE window, choose *Outputs – Save All.*

The Save Options form appears.

2. In the *Select signals to output section*, be sure *allpub* is highlighted.




3. Click *OK*.

## Setting Up the PSS Analysis

**Note:** If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this PSS analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the analysis.

   The form changes to display options needed for the PSS analysis. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

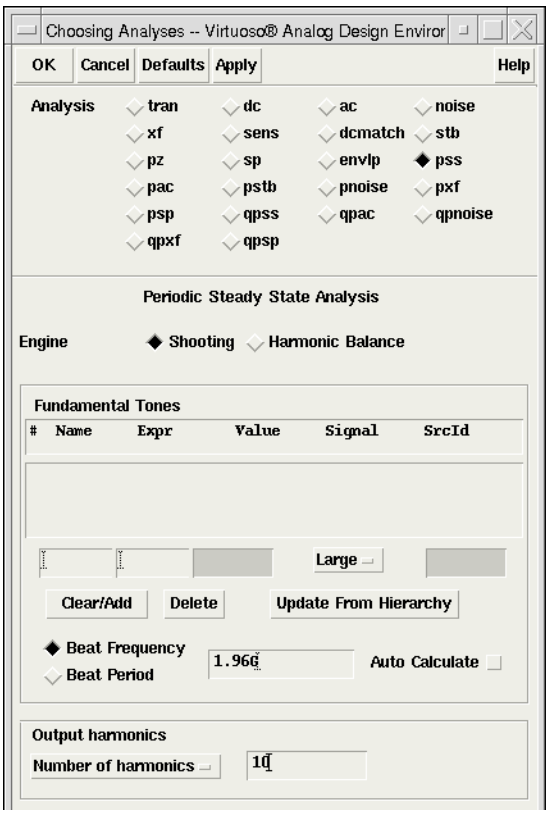
3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

4. In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   For this example, type `1.9G`.

5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `10` in the field.

   The top of the PSS Choosing Analyses form appears below.




6. Highlight *conservative* for the *Accuracy Default (errpreset)*.

7. Type `100.6n` in the *Additional Time for Stabilization (tstab)* field.

   You can use a slightly different value as long as it is above 100.

8. If you want to save the results of the initial transient analysis, set *Save Initial Transient Results (saveinit)* to *yes*.

9. Highlight *Oscillator*.

The form changes to let you specify the two nodes needed for oscillator simulation.

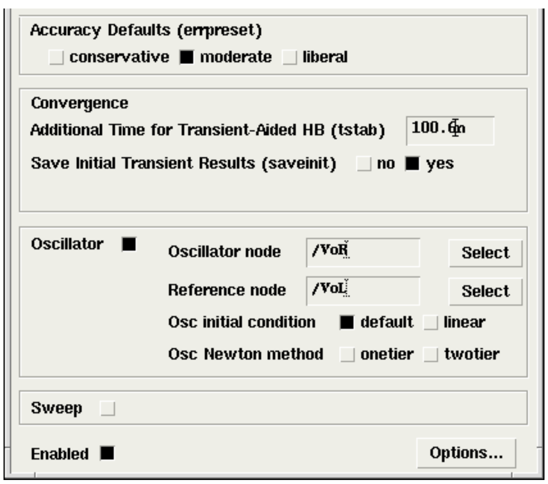
**10.** Click the *Oscillator node Select* button. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled *VoR*.

`/VoR` appears in the *Oscillator node* field.

**11.** Click the *Reference node Select* button. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node labeled *VoL*.

`/VoL` appears in the *Reference node* field. Because this is a differential oscillator, the reference node is not `/gnd!` as it was in the *tline3oscRF* example.

The bottom of the Choosing Analyses form for PSS looks like the following.




**12.** Verify that *Enabled* is highlighted for the PSS analysis and click *Apply*.

## Setting Up the Pnoise Analysis

**1.** At the top of the Choosing Analyses form, click *pnoise*.

The form changes to let you specify data for the `Pnoise` analysis.

**2.** In the *Sweeptype* cyclic field, choose *relative*. Enter 1 in the *Relative Harmonic* field.

This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify 2K, you choose a value 2K away from the fundamental frequency.

**3.** In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `100` and `200M` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

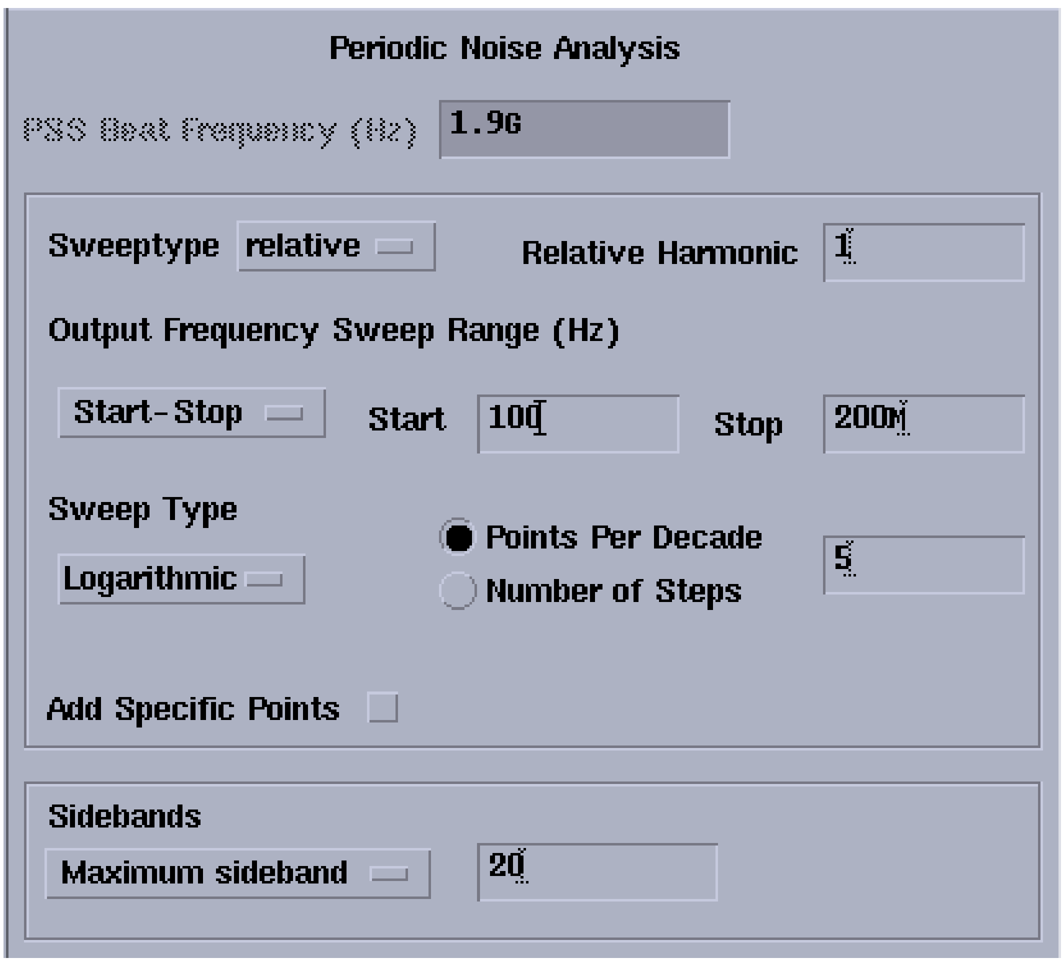
5. Highlight *Points Per Decade* and type `5` in the field.

   Be sure to use a nonzero starting value for logarithmic sweeps.

6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `20` in the *Sidebands* field. The default value is 20.

   Be sure to use a nonzero *Maximum sideband* value.

   The top of the Choosing Analyses form for the `Pnoise` analysis looks like this.




7. In the *Output* cyclic field, choose *voltage*.

8. Notice the informational message that displays in the CIW.

```
*info*  Since voltage was selected as the output measurement
        technique for the pnoise analysis, spectre will NOT
        subtract any load resistance contribution from the Noise
```

```
Figure calculation. If this is undesirable, please select
probe as the output measurement technique and select a port.
```

**9.** Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoR*.

`/VoR` appears in the *Positive Output Node* field.

**10.** Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoL*.

`/VoL` appears in the *Negative Output Node* field. Because this is a differential oscillator, the negative output node is not `/gnd!` as it was for the *tline3oscRF* oscillator.

**11.** In the *Input Source* cyclic field, choose *current*.

**12.** Click the *Input Current Source Select* button. Then click the appropriate component in the Schematic window to choose /I*2*.

`/I2` appears in the *Input Current Source* field.

**13.** In the *Reference side-band* cyclic field, choose *Enter in Field* and type `-1` in the *Reference side-band* field.

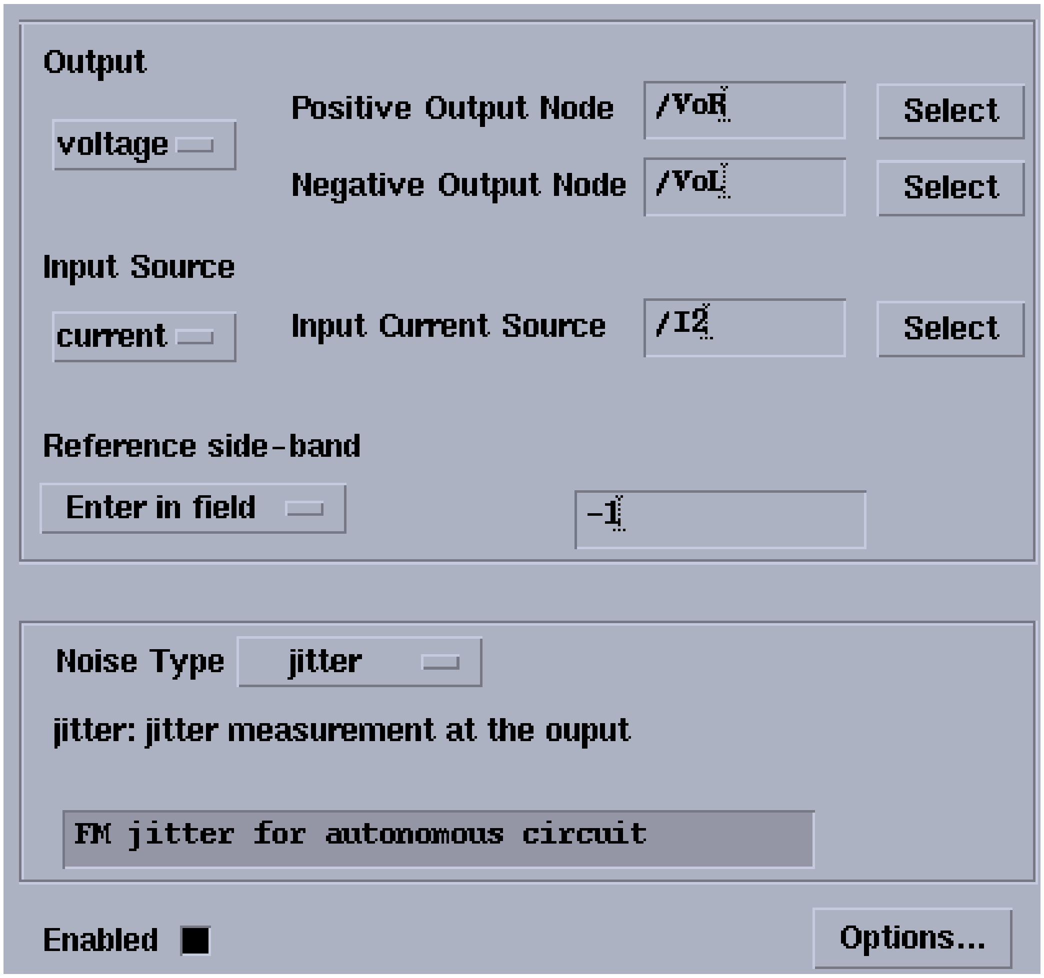
**14.** In the *Noise Type* cyclic field, choose *Jitter.*

**15.** The following information displays.

*jitter: jitter measurement at the output*

*FM jitter for autonomous circuit*

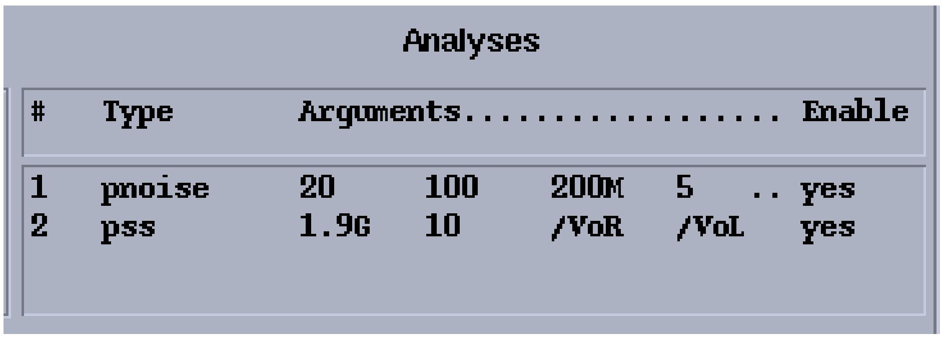
The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.




**16.** Verify that the *Enabled* field at the bottom of the Pnoise form is highlighted. Then click *Apply* at the top of the Choosing Analyses form.

**17.** Click *OK*.

The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.




## Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Measuring Jitter

### Plotting Phase Noise

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

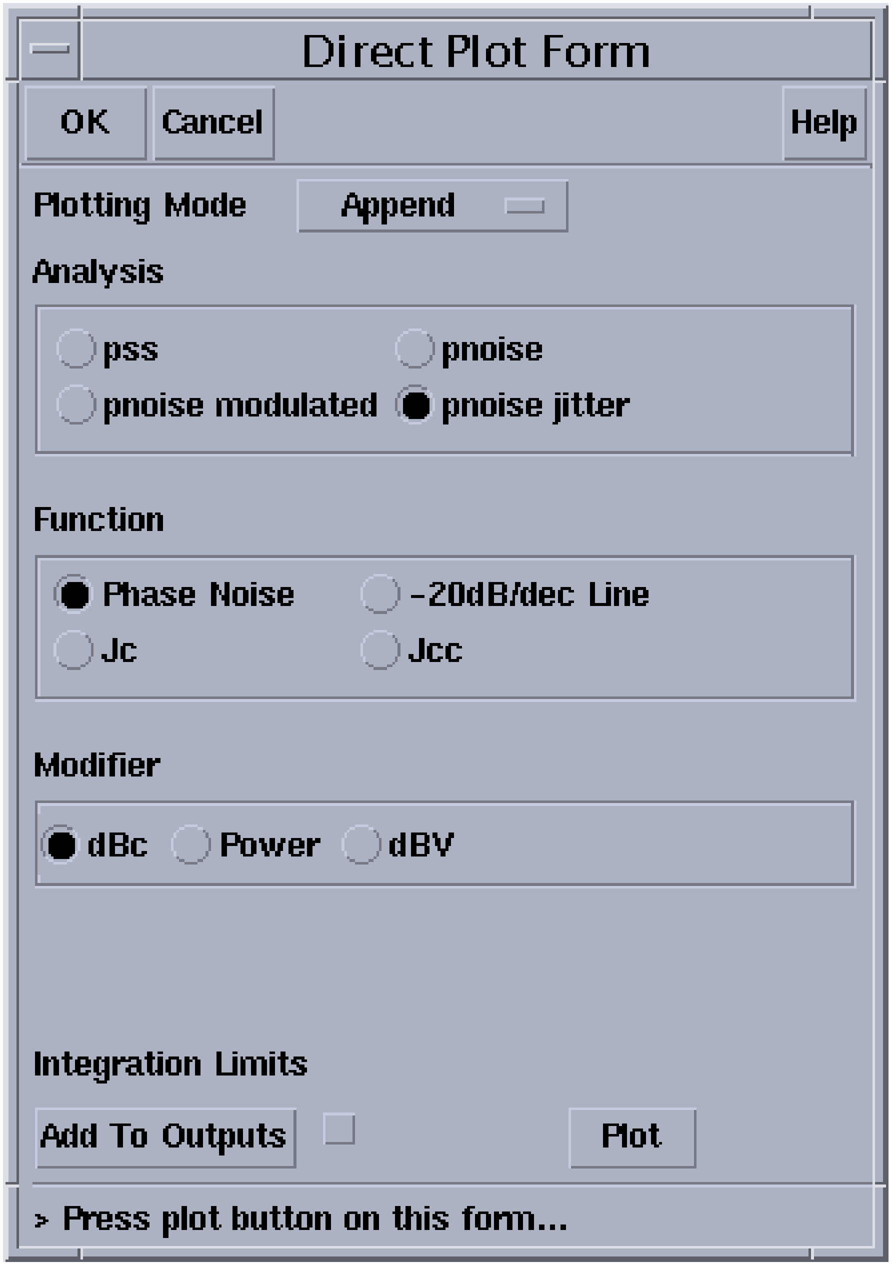
   The Direct Plot form appears.

2. Highlight *Replace* for *Plotting Mode*.
3. Highlight *pnoise jitter* for *Analysis*.

   This selects the results of the pnoise jitter analysis for display and changes the appearance of the Direct Plot form.

4. Highlight *Phase Noise* for *Function*.
5. Highlight *dBc* for *Modifier*.

The Direct Plot form looks like this.



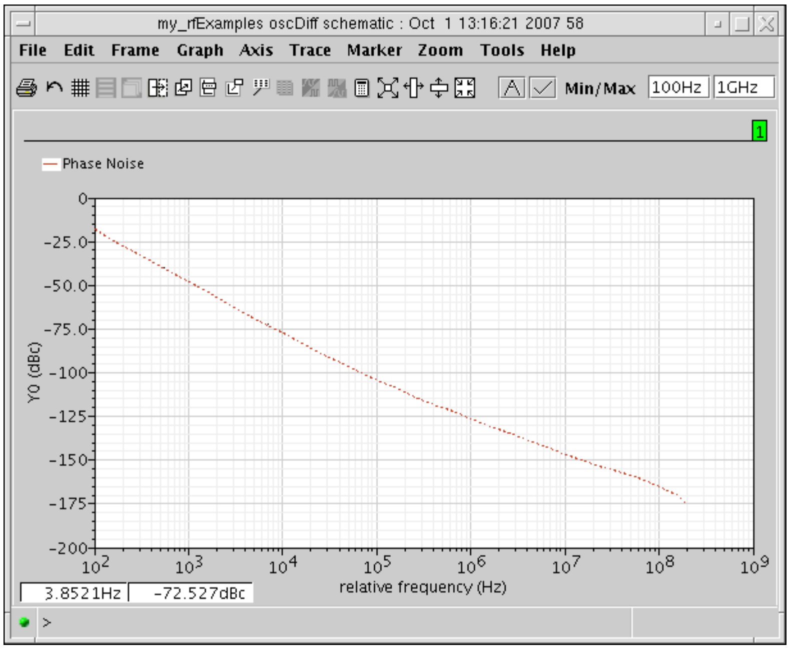

6. In the Direct Plot form, click *Plot*.
7. In the waveform window, set the x axis to a logarithmic scale.

   Using a logarithmic scale on the x axis makes it easier for you to figure out the range of frequencies over which to measure simple jitter.

   a. In the waveform window, select the x axis.

**b.** Choose *Axis – Log* to choose a logarithmic scale for the x axis.

The waveform window displays the phase noise curve.



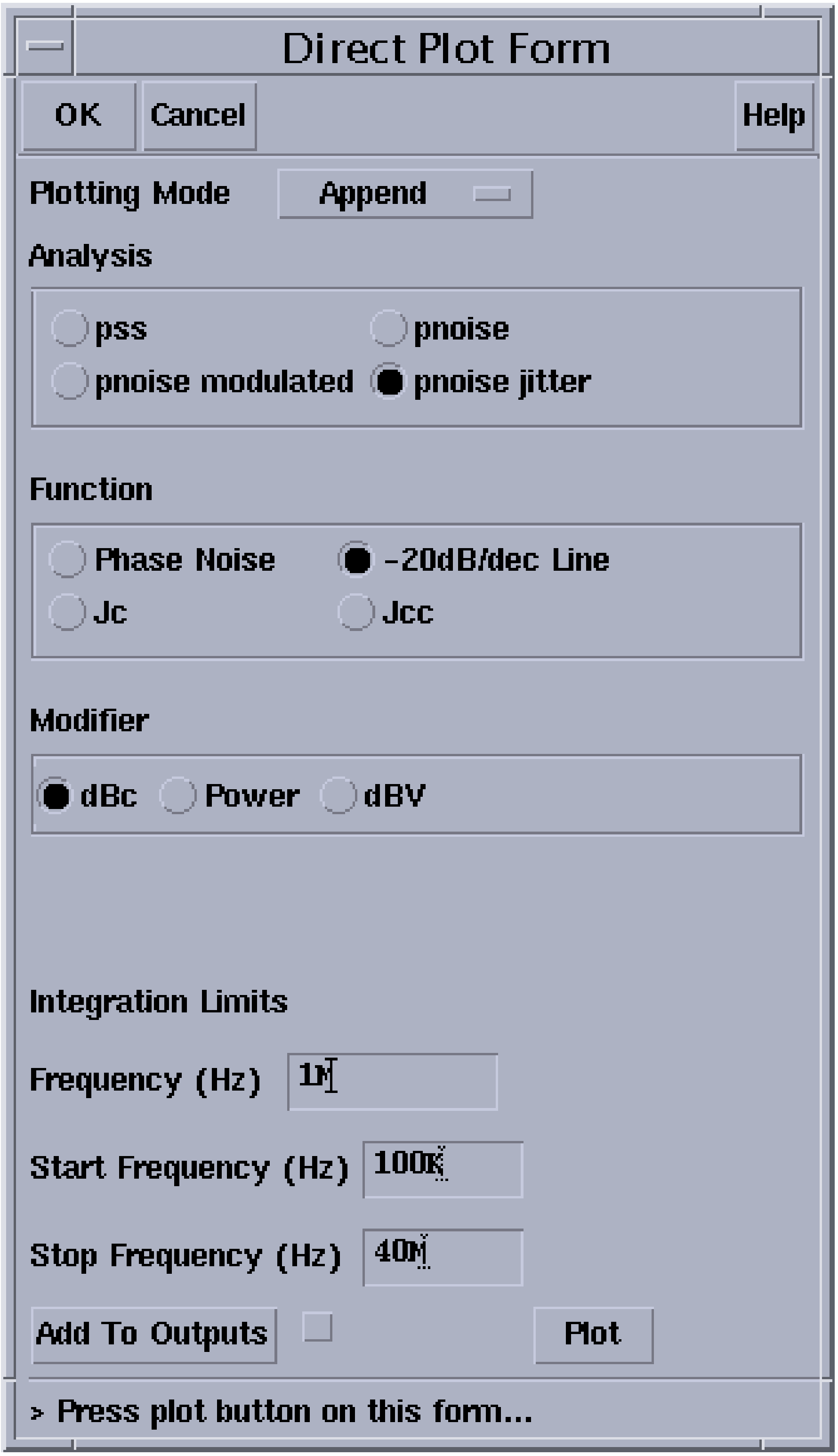

### Find the White FM Noise or the 1/f$^2$ Range of the Phase Noise

The `-20dB/decade` function provides a way to find the frequency range where white FM noise is significant. The function also makes it also possible to see the presence of the flicker noise in the system.

1. In the Direct Plot Form, select *Append* in the *Plotting Mode* cyclic field.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *dBc* for *Modifier*.
4. Highlight *-20dB/dec Line* for *Function*.

   The form changes to display the *Integration Limits* section.
5. Set the *Frequency (Hz)* to 1M., a value in the range where the shape of the phase noise curve is changing from `1/f` to `1/f`$^2$ to `1/f`$^3$.
6. Ignore the *Start Frequency (Hz)* and *Stop Frequency (Hz)* fields. They are not used in the plotting step that follows.

The Direct Plot form appears as follows.

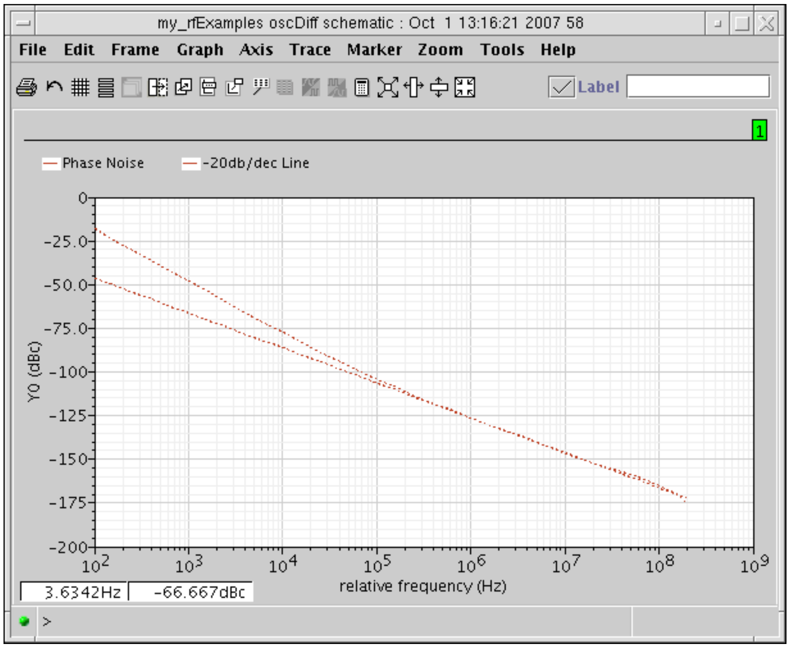


7. Follow the prompt at the bottom of the Direct Plot form

   *Press plot button on this form...*

   and click *Plot*.

   The waveform window displays both the phase noise curve and the -20dB/decade line.




8. Estimate the `1/f`$^2$ section of the phase noise curve. You can see the FM flicker noise at the frequencies below the `1/f`$^2$ range (200-300 kHz).

### Plotting Cycle-to-Cycle Jitter

1. In the Direct Plot Form, select *Replace* in the *Plotting Mode* cyclic field.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jcc* for *Function*.

   The form changes to add fields where you can enter more data.
4. The *Number of Cycles [k]* defaults to 1.
5. Highlight *rms* for *Signal Level*.
6. Highlight *Second* for *Modifier*.

   Select the *Modifier* value based on the units you are using in your design specifications.

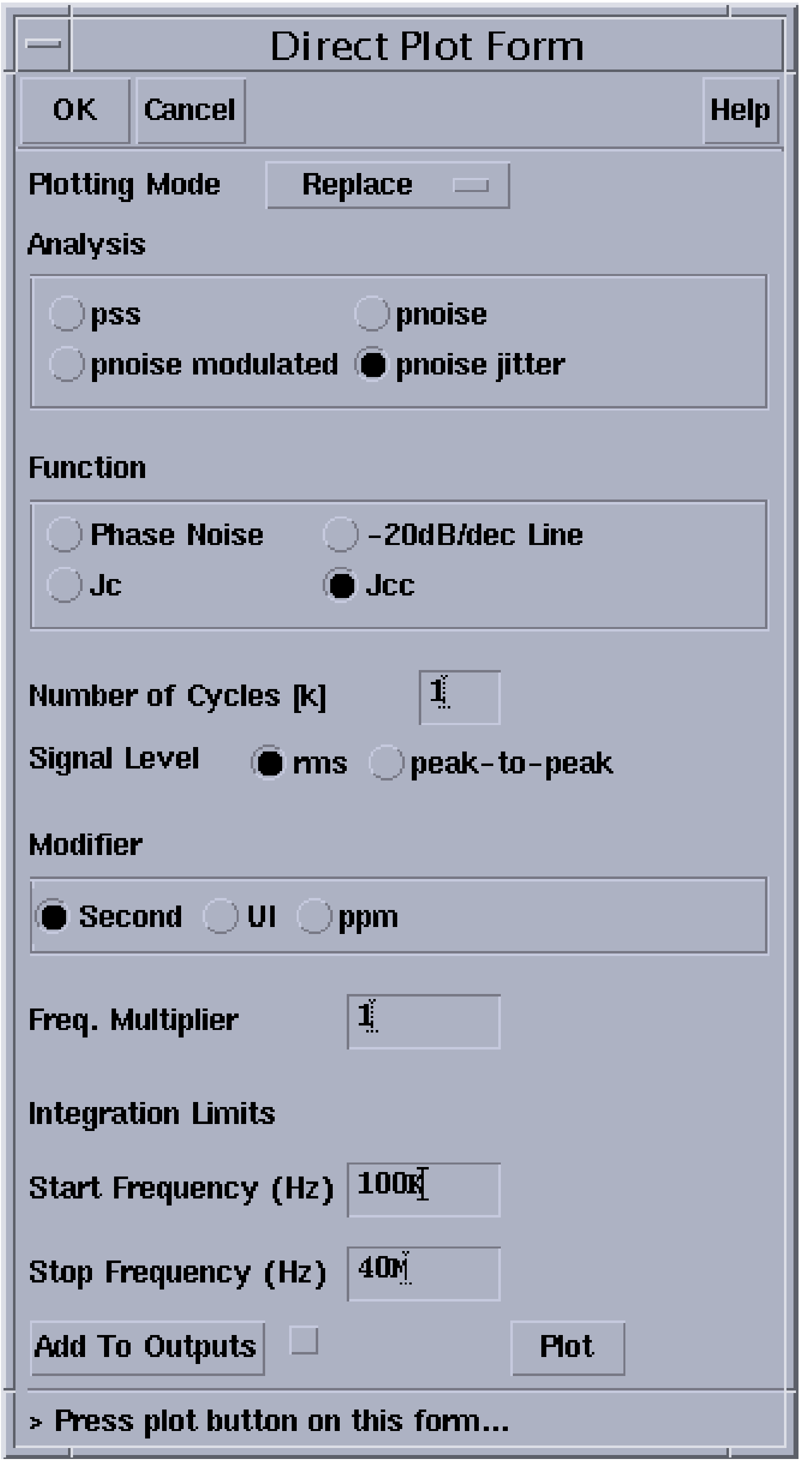
7. The *Freq. Multiplier* field defaults to 1.

8. Fill in the *Start Frequency (Hz)* and *Stop Frequency (Hz)* fields.

   The appropriate values depend on the methodology. Sometimes appropriate values are included in the jitter specifications or in the verification methodology. The best values to use also depend on the circuit properties. For a small *Number of Cycles [k]*, such as a short observation time of one or a few periods, the lower frequency noise does not contribute much to either Jc or Jcc jitter. For a larger *Number of Cycles [k]*, the lower frequency noise and the lower limit of integration are important. See *Application Note. Long Term jitter measurements* for information on how to set up the lower limit of the integration.

   To continue the example, type 100K into the *Start Frequency (Hz)* field and type 40M into the *Stop Frequency (Hz)* field.

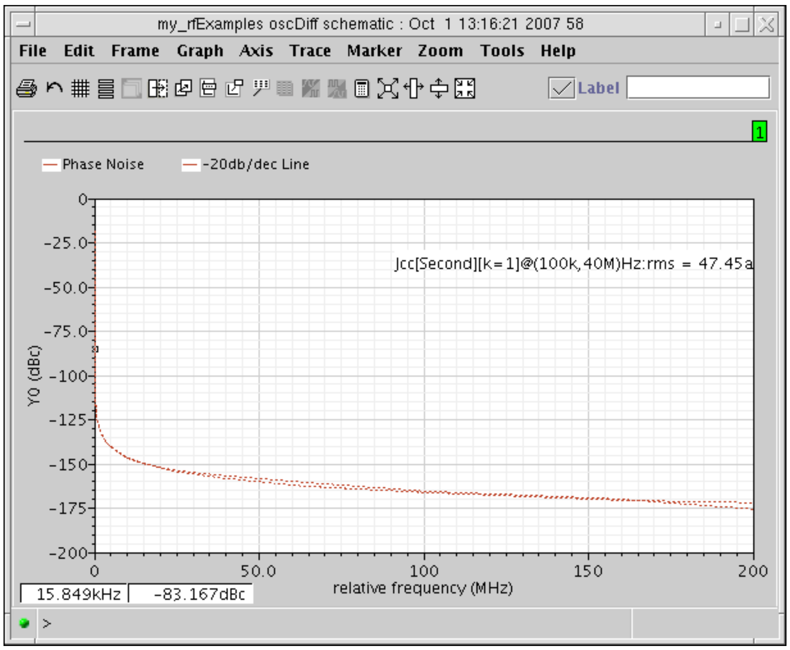
The Direct Plot form looks like this.




**9.** Click *Plot*.

**10.** The waveform window displays the Jcc value.


The Jcc value is added to the waveform.




The phase noise is numerically integrated during the jitter computations. It accounts for all noise contributors that exist in the schematic.

There is a significant difference between RMS phase jitter and cycle or cycle-to-cycle jitter.

- RMS jitter is easily generated by integrating the phase noise over the required frequency range. The contributions from different frequency ranges correspond to the PSD of the phase noise.
- The situation is different for Jc and Jcc. Regardless of the phase noise PSD, the lower frequency contributions are mostly negligible for the small K cycle, short term jitter computations and become important only as the observation time increases.

## Troubleshooting for Oscillator Circuits

If your oscillator simulation fails to converge, try the following techniques to help the Spectre RF simulator find the solution. You can find more information about each of these techniques in the Spectre RF Theory document available in the online documentation library.

- Increase the value of the `tstab` parameter.
- Be sure that you successfully started the oscillator.

  PSS fails to converge if the circuit cannot sustain an autonomous oscillation. Start the oscillator so that it responds with a signal level between 25 percent and 100 percent of the expected final level. Also, do not "kick" the oscillator so hard that its response is unnecessarily nonlinear. If possible, avoid exciting response modes that are unrelated to the oscillation, especially those associated with longtime constants.

- Improve your estimate of the period.

  Be careful that the period you specify is not too short. Overestimating the period is better than underestimating the period.

- When possible, use *gear2only* as the value of the `method` parameter.
- Increase the value of the `maxperiods` parameter to increase the maximum number of iterations for the shooting methods.
- If the shooting iteration approaches convergence and then fails, increase the value of the `steadyratio` parameter.

  The `steadyratio` parameter controls how much the final solution can deviate from being periodic.

- Change the value of the `tolerance` parameter for periodic small-signal analyses.

  Changing the value of the `tolerance` parameter does not change the accuracy of the final solution, but it does increase the chance that the periodic small-signal analyses converge.

- Change the value of the `itres` parameter for the large-signal PSS analyses.

  The `itres` parameter adjusts the level of accuracy used when solving nonlinear systems of equations. Loosen the `itres` parameter but make sure you get convergence for the nonlinear problem. If your PSS simulation converges, the final result is accurate.

  The default value for the `itres` parameter is $1e^{-4}$ which can sometimes be too tight for particular circuits. You can the increase the `itres` parameter value as long as it remains less than 1.

Tighten the normal simulation tolerances by changing the values of the `maxstep`, `reltol`, `iteratio`, and `errpreset` parameters. Avoid setting `errpreset` to `liberal`.

# 7

# Simulating Low-Noise Amplifiers

To use this chapter, you must be familiar with the Spectre RF simulator analyses and have a working knowledge of low-noise amplifier design. For more information about the Spectre RF simulator analyses, see Chapter 2, “Spectre RF Analyses.”

## Analyses and Measurement Examples in this Chapter

This chapter shows you how to simulate and plot the following:

- Voltage Gain
- Output Voltage Distribution
- S-Parameter Analysis, including the Voltage Standing Wave Ratio
- S-Parameter Noise Analysis, including
    - Noise Figure and Minimum Noise Figure
    - Equivalent Noise Resistance
    - Load and Source Stability Circles
    - Noise Circles
- Noise Figure Calculations with the Pnoise Analysis
- The 1dB Compression Point
- The Third-Order Intercept Point
- Conversion Gain and Power Supply Rejection with the Periodic Transfer Function Analysis

# Simulating the lnaSimple Example

Before you start, perform the setup procedures described in Chapter 4, "Setting Up for the Examples."

## Opening the lnaSimple Circuit in the Schematic Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

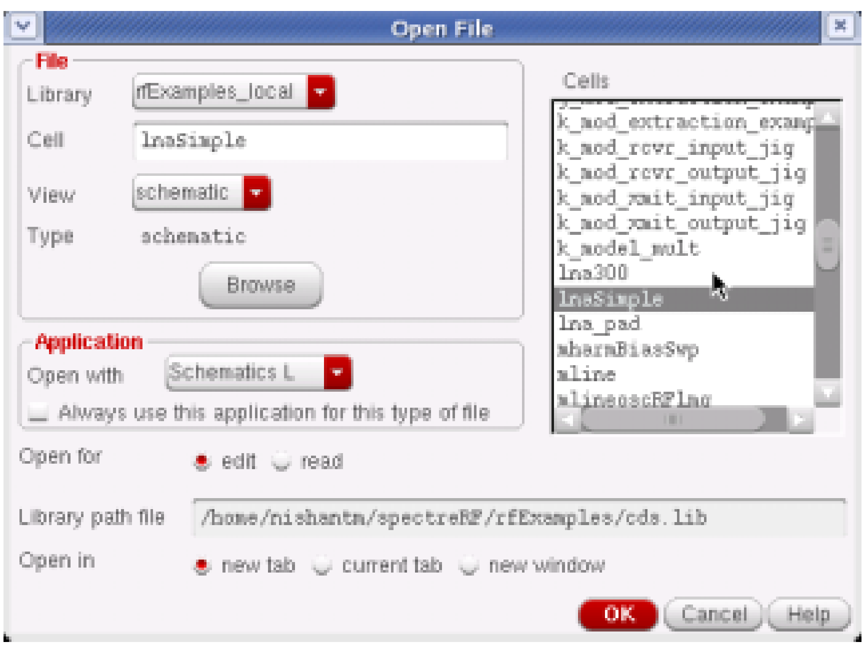
   The Open File form appears.

2. Choose *my_rfExamples* for *Library Name*. This is the editable copy of the *rfExamples* library you created following the directions given in Chapter 4, "Setting Up for the Examples."

3. Choose *lnaSimple* in the *Cell Names* list box.

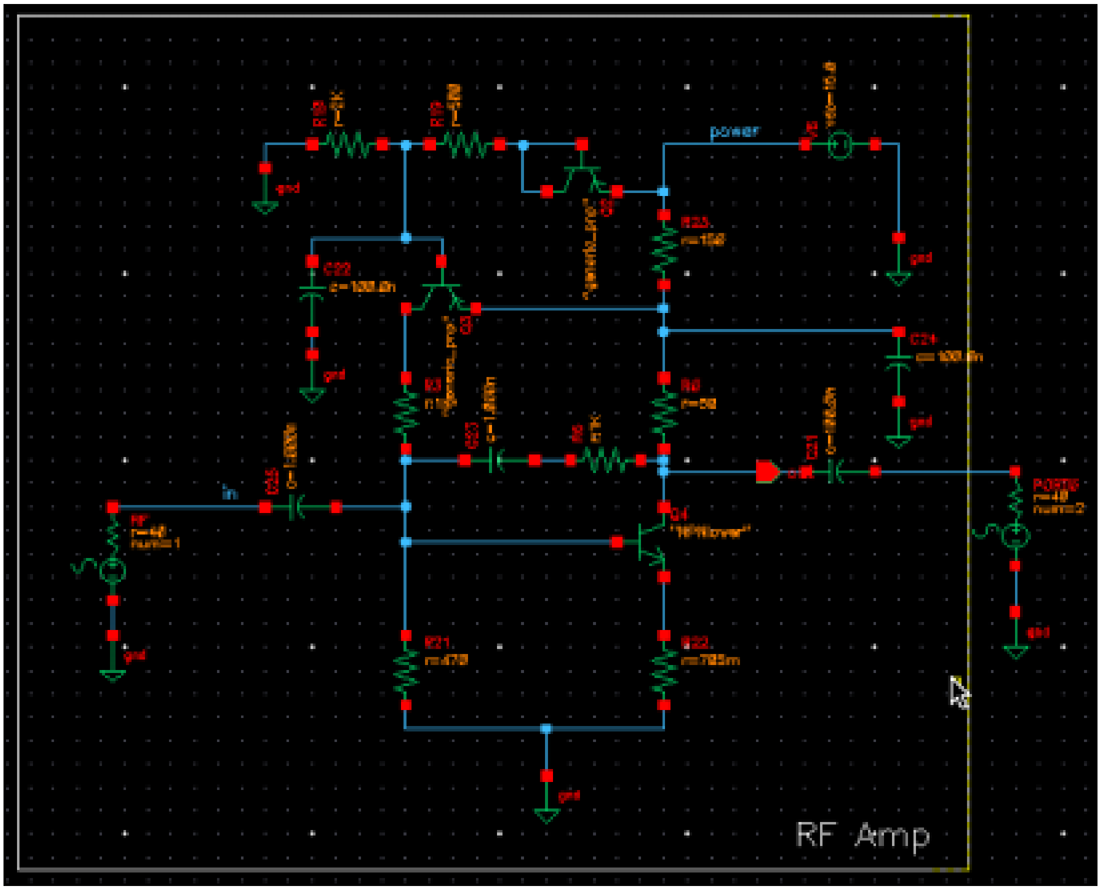
4. Choose *schematic* for *View Name*.

   The filled-in form appears like the one below.



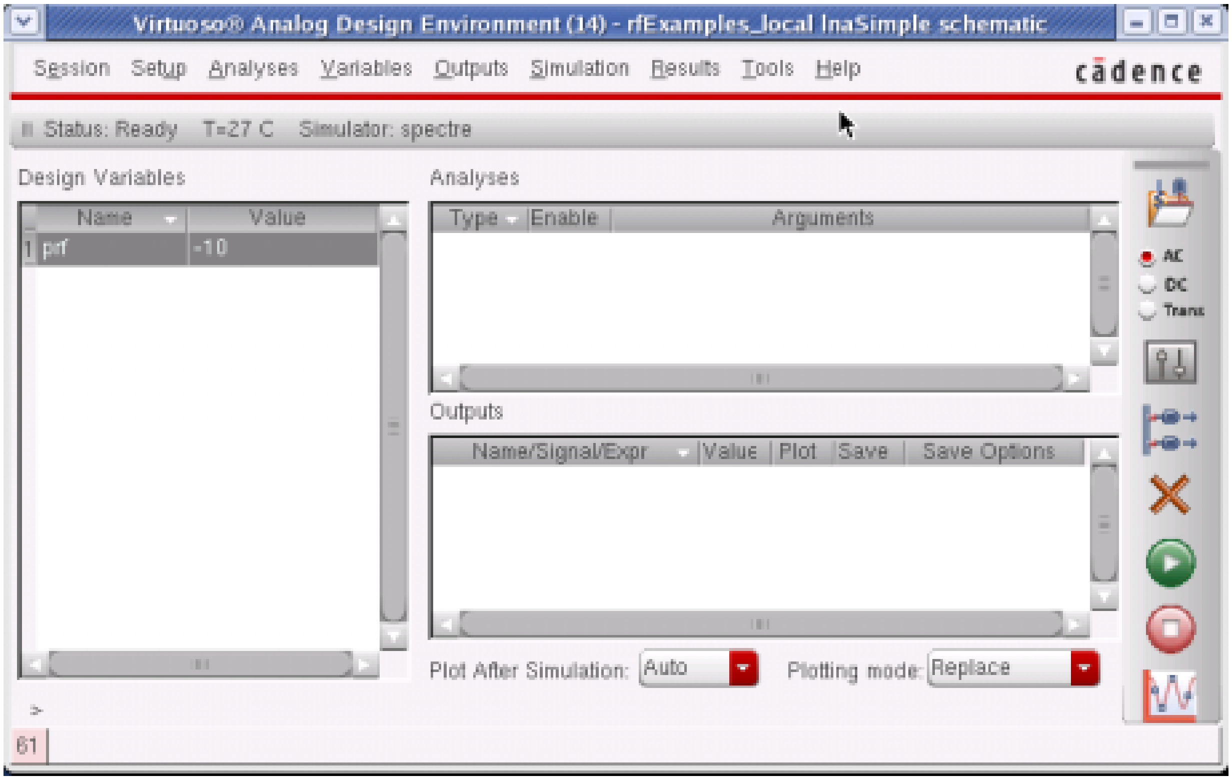

5. In the Open File form, click *OK.*

The Schematic window now shows the *lnaSimple* schematic.




**6.** In the Schematic window, choose *Tools – Analog Environment*.

The Virtuoso Analog Design Environment (ADE) window appears.



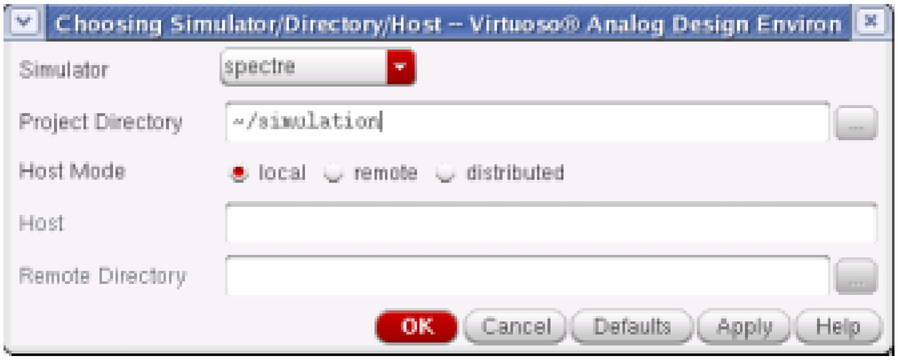

## Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

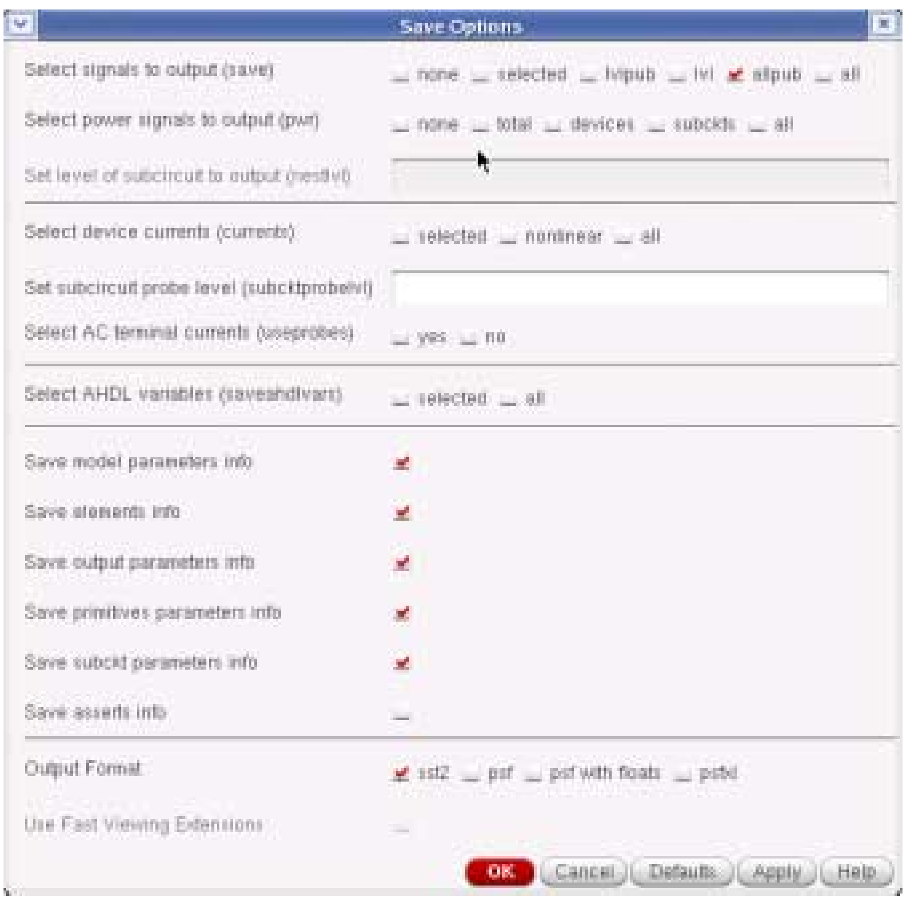
2. Choose *spectre* for *Simulator*.
3. Type the name of the project directory, if necessary.
4. Highlight the *Host Mode* that corresponds to your situation.
5. If you choose *remote* or *distributed*, fill in the additional fields as necessary.

The completed form looks like this.




6. In the Choosing Simulator/Directory/Host form, click *OK*.
7. In the ADE window, choose *Outputs – Save All*.

   The Save Options form appears.
8. In the *Select signals to output (save)* section, be sure *allpub* is highlighted.



9. Click *OK*.

   The Save Options form closes.

### Setting Up Model Libraries

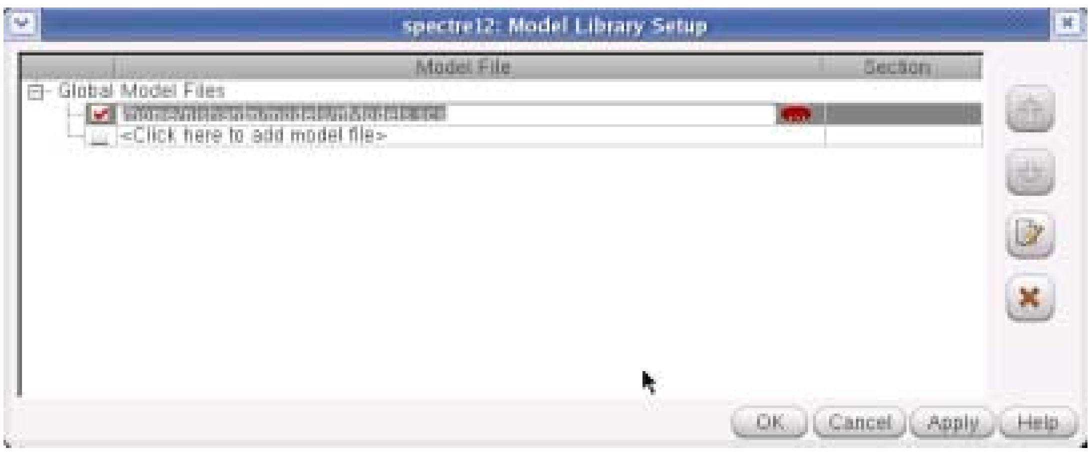
1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs`.

3. In the Model Library Setup form, click *Add*.

   The completed form looks like this.




4. In the Model Library Setup form, click *OK*.

## Calculating Voltage Gain with PSS

The most important characteristic of an amplifier is its gain. In the following example, a PSS analysis determines the voltage gain of the low noise amplifier.

### Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.

2. If necessary, set up the simulator options.

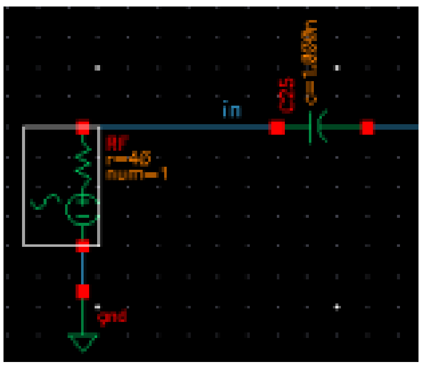
3. If necessary, specify the full path to the model files in the Model File Set-up form.

For this simulation you need the `rfModels.scs` model file.

4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

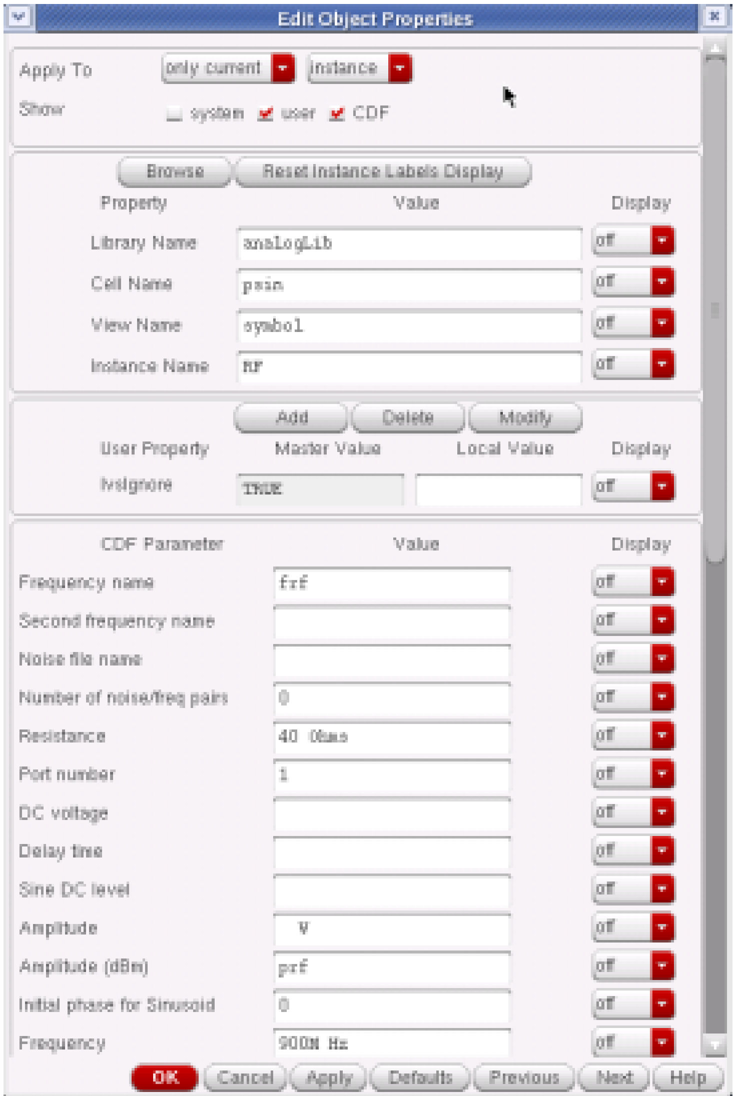
## Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.




2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form for the port appears.




3. If necessary, in the *Source Type* field type `sine` and click *OK* in the Edit Object Properties form.

4. In the Schematic window, choose *Design – Check and Save*.

## Setting Up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

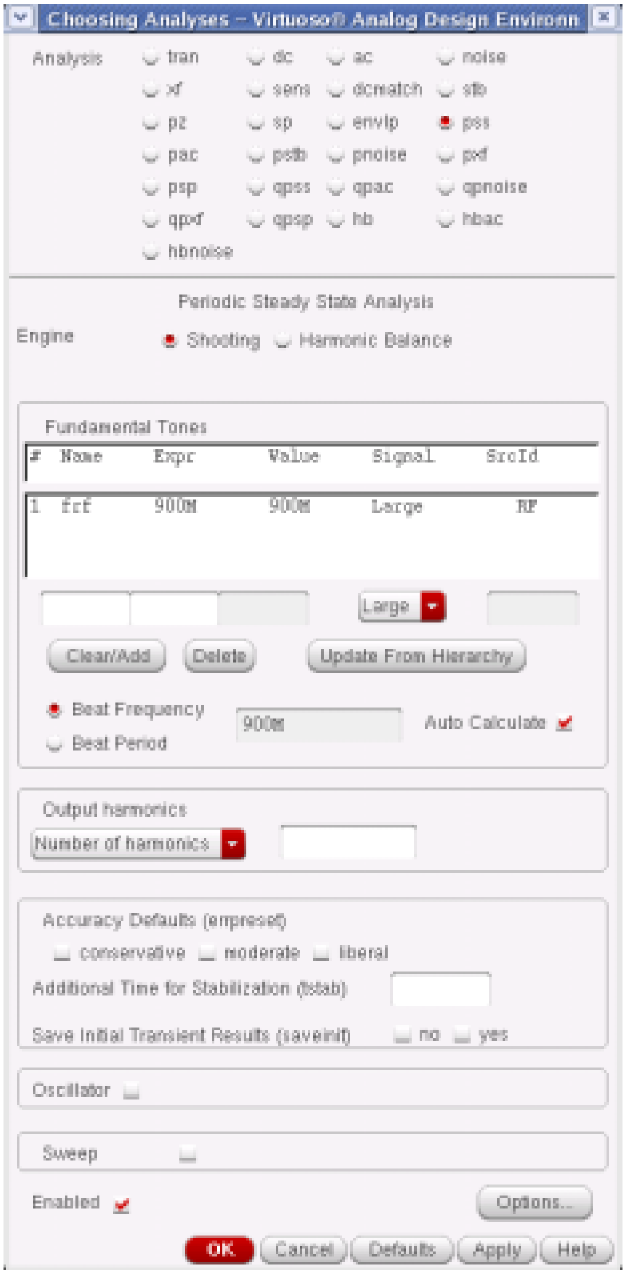
The form changes to let you specify data for the PSS analysis.

3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Highlight the *Auto Calculate* button.

   The fundamental frequency for this analysis, which is `900M`, appears in the *Beat Frequency* field.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type a reasonable value, such as 10, in the *Number of harmonics* field. (You can choose a different number.)

The PSS Choosing Analyses form looks like this.




5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
6. Highlight *Enabled*, if necessary.
7. In the Choosing Analyses form, click *OK*.

## Running the Simulation

1. To run the PSS analysis, choose *Simulation – Netlist and Run* in the ADE window.

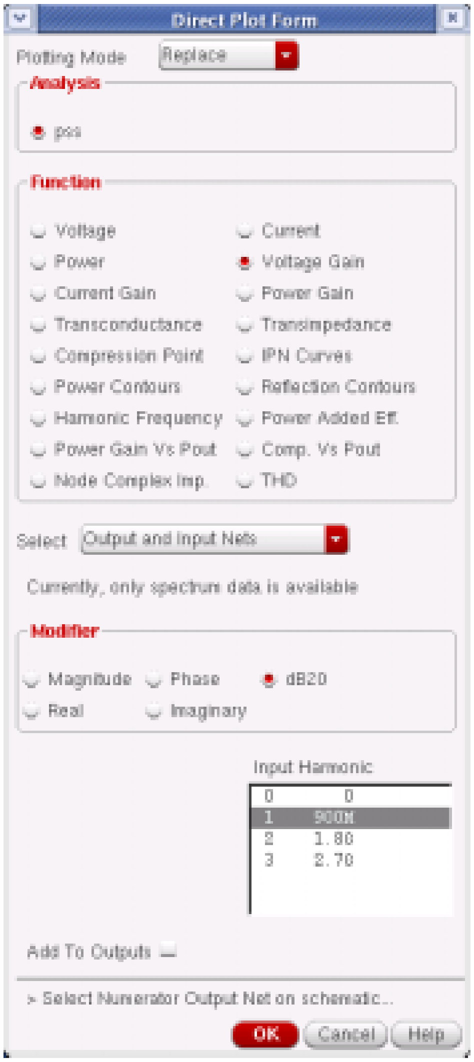
   The output log file appears and displays information about the simulation as it runs.
2. Look in the output log file to be sure the simulation completes successfully.

## Plotting Voltage Gain

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, highlight *Replace* for *Plotting Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Voltage Gain* for *Function*.

The Direct Plot form appears below.




The Direct Plot form changes to reflect your choice.

Notice that the *Select* cyclic field changes to display *Output and Input nets*.

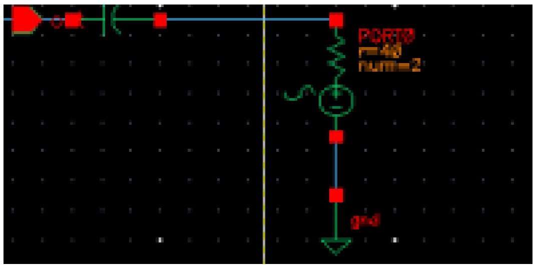
Notice the message at the bottom of the form *Select Input Harmonic on this form…*.

5. Highlight *dB20* for *Modifier*.
6. Highlight *900M* in the *Input Harmonic* list box.
7. Calculate the voltage gain as *Vout*/*Vin* by following the instructions at the bottom of the Direct Plot form as illustrated in the next two steps—click first on the output net then click the input net.

**8.** Following the message at the bottom of the form,

*Select Numerator Output Net on schematic...*

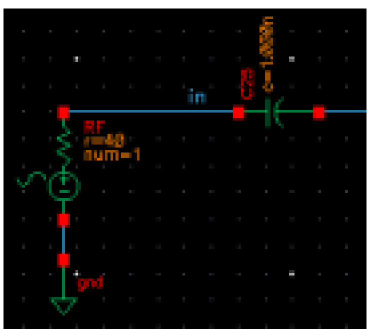
Select the `Vout` output net.




**9.** Following the message at the bottom of the form,

*Select Denominator Input Net on schematic...*

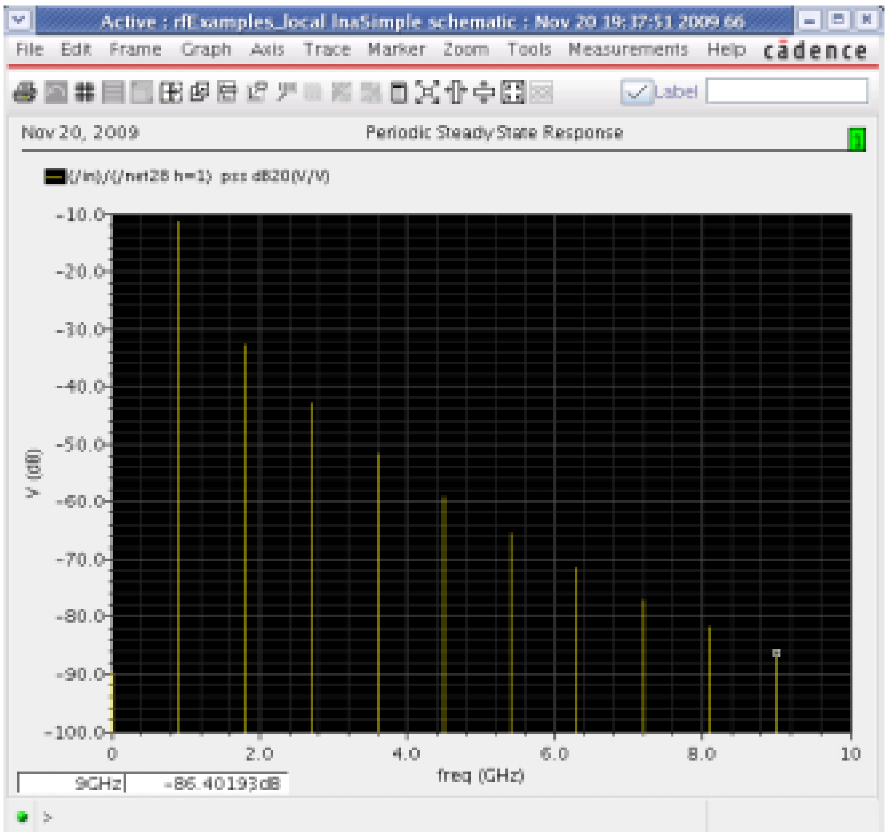
Select the `Vin` input net at the RF source.




The voltage gain plot appears in waveform window.

**10.** To display the voltage gain value, in the waveform window, place your cursor at the top of the line representing the 900 MHz fundamental (the tallest, leftmost line).

The x and y axis values displayed in the lower-left corner of the waveform window show that the voltage gain at 900 MHz is 11.45 dB.




# Calculating Output Voltage Distribution with PSS

The output voltage distribution of an amplifier is important in analyzing intermodulation distortion when the two RF input signals are close together. To model this situation, you modify a parameter to make the voltage source component generate two separate frequencies.

## Setting Up the Simulation

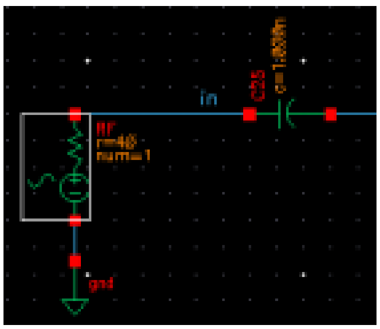
1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file

**4.** In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

## Editing the Schematic

**1.** In the Schematic window, click the RF voltage source.



**2.** In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.


3. In the *Second frequency name* field, type `fund2,` or any name that you choose.
4. In the *Frequency 2* field, type `920M`.
5. In the *Amplitude 2 (dBm)* field, type `prf`.

   This sets the amplitude of the second tone equal to that of the first tone.
6. In the Edit Object Properties form, click *OK*.
7. In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

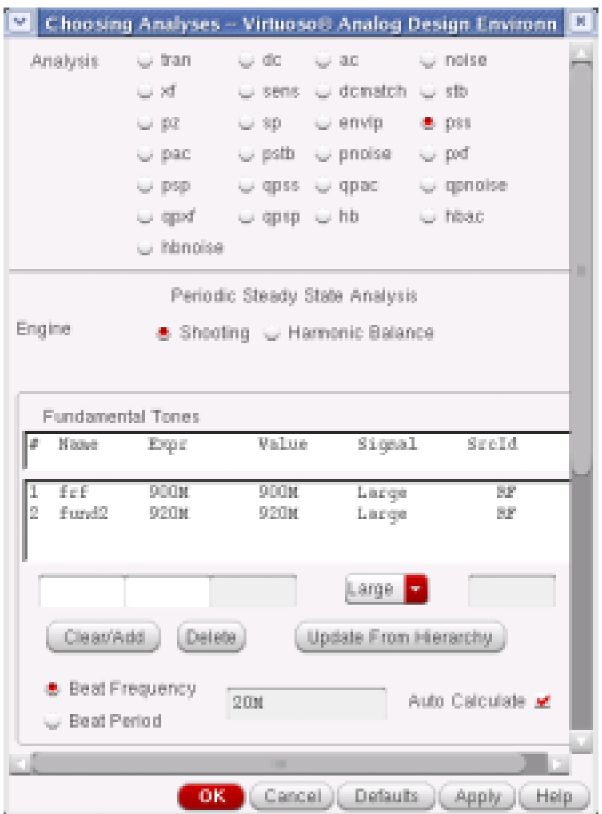
2. Click *pss* for the *Analysis* choice.

   The Choosing Analyses form changes to display fields for the PSS analysis.

3. In the *Fundamental Tones* area, highlight the *Auto Calculate* button. The *Beat Frequency* button is highlighted by default.

   The *Beat Frequency* field displays `20 M` because there are two RF tones—one at 900 M and the second at 920 M.

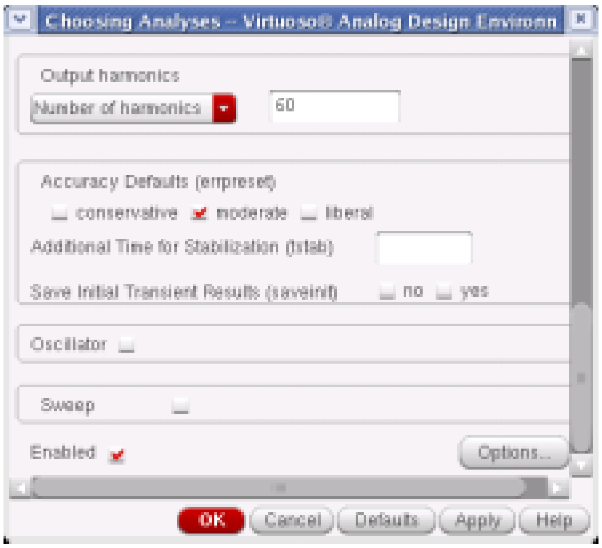
   The top of the Choosing Analyses form appears as follows.




4. In the *Output harmonics* cyclic field, choose *Number of Harmonics* and type `60` in the *Number of Harmonics* field.

   The largest common multiple of 900 Mhz and 920 MHz is 20MHz. Using 60 harmonics extends the results out to 1.2 GHz, but you can choose another number if you want.

5. Highlight *moderate* as the *Accuracy Defaults* value.
6. Make sure that *Enabled* is highlighted.
7. The bottom of the Choosing Analyses form appears as follows.




8. In the Choosing Analyses form, click *OK*.

## Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the PSS analysis.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Output Voltage Distribution

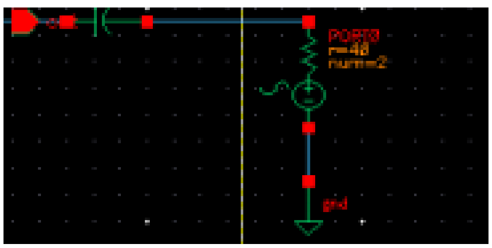
1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Voltage* for *Function*.
5. Notice that the *Select* cyclic field displays *Net*.

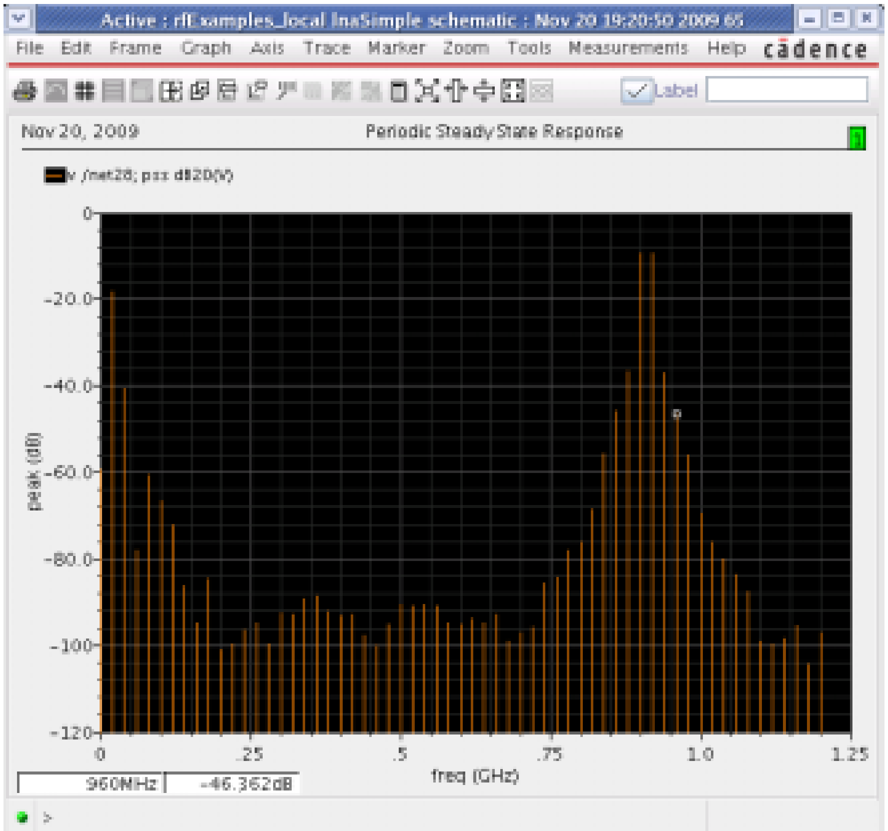
6. Highlight *spectrum* for *Sweep* value.
7. Highlight *peak* for *Signal Level*.
8. Highlight *dB20* for *Modifier*.
9. Following the message at the bottom of the form,

   *Select Net on schematic...*

   Select the `Vout` amplifier output net.



In the waveform window, you can see spikes at 900 MHz, 920 MHz (-9.3 dB) and a third spike at 20 Mhz. The next strongest harmonics are lower than -36 dB.




# S-Parameter Analysis for Low Noise Amplifiers

Before you can perform S-parameter analyses of the low noise amplifier, you need to set up for the example.

## Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.

4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

## Editing the Schematic

1. In the Schematic window, select the `RF` voltage source.
2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
3. In the Edit Object Properties form, be sure the *Source type* is set to `sine`, delete the *Frequency 2* value, if there is one, and click *OK*.

   The input voltage source now has only one frequency at `900MHz`. *Frequency 2* is zero.
4. In the Schematic window, choose *Design – Check and Save*.

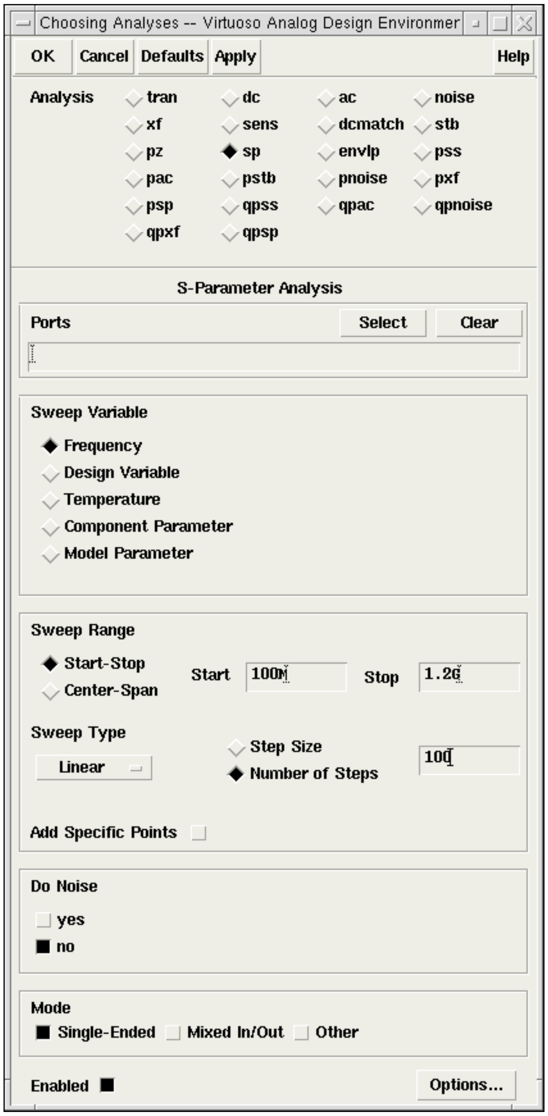
## Setting up the S-Parameter Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click *sp* for *Analysis*.

   The form changes to let you specify data for an S-parameter analysis.
3. Highlight *Frequency* for *Sweep Variable*.
4. Highlight *Start-Stop* for *Sweep Range* and type `100M` for *Start* and `1.2G` for *Stop*.
5. In the *Sweep Type* cyclic field, choose *Linear.*
6. Click the *Number of Steps* button and type `100` in the *Number of Steps* field.
7. Highlight *Enabled*.

**8.** The filled-in form looks like this.




**9.** In the Choosing Analyses form, click *OK*.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

The output log file appears and displays information about the simulation as it runs.

**2.** Look in the CIW for a message that says the simulation completed successfully.

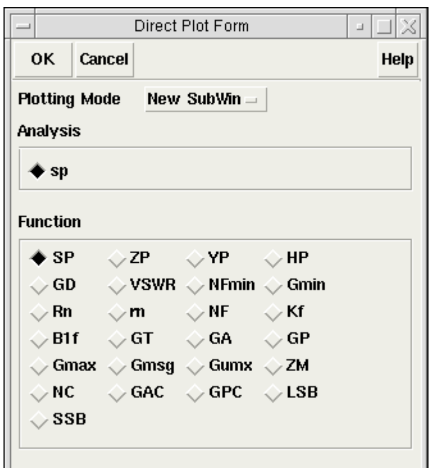
## Plotting S-Parameters

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

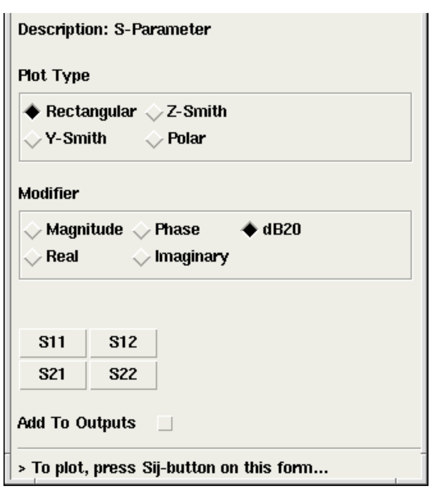
2. Select *New SubWin* for *Plotting Mode*.
3. Highlight *sp* for *Analysis*.
4. Highlight *SP* for *Function*.

   The top of the Direct Plot form looks like this.




5. Highlight *Rectangular* for *Plot Type*.
6. Highlight *dB20* for *Modifier*.

   The bottom of the SP Direct Plot form looks like this.



**7.** Following the message at the bottom of the form,

*To plot, press Sij-button on this form...*

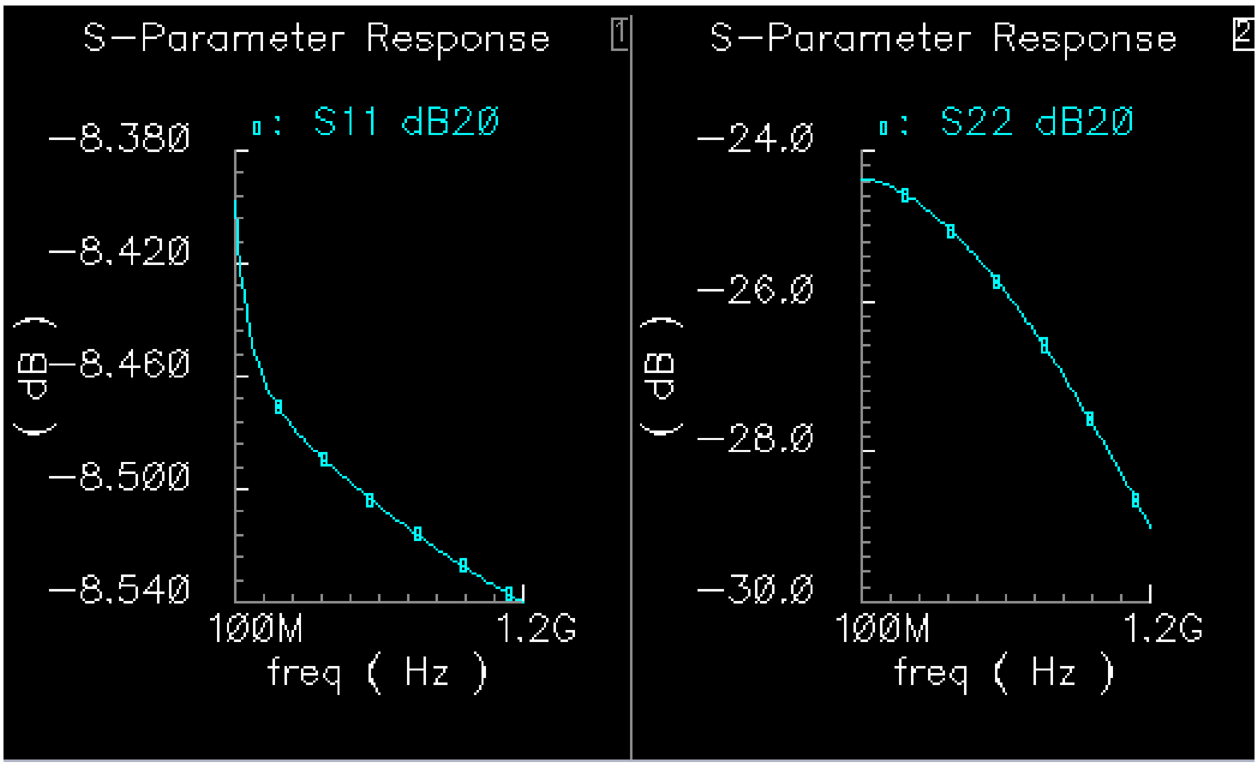
Select the S-parameters to plot:

**a.** Click *S11*.

S11 is plotted in the waveform window.

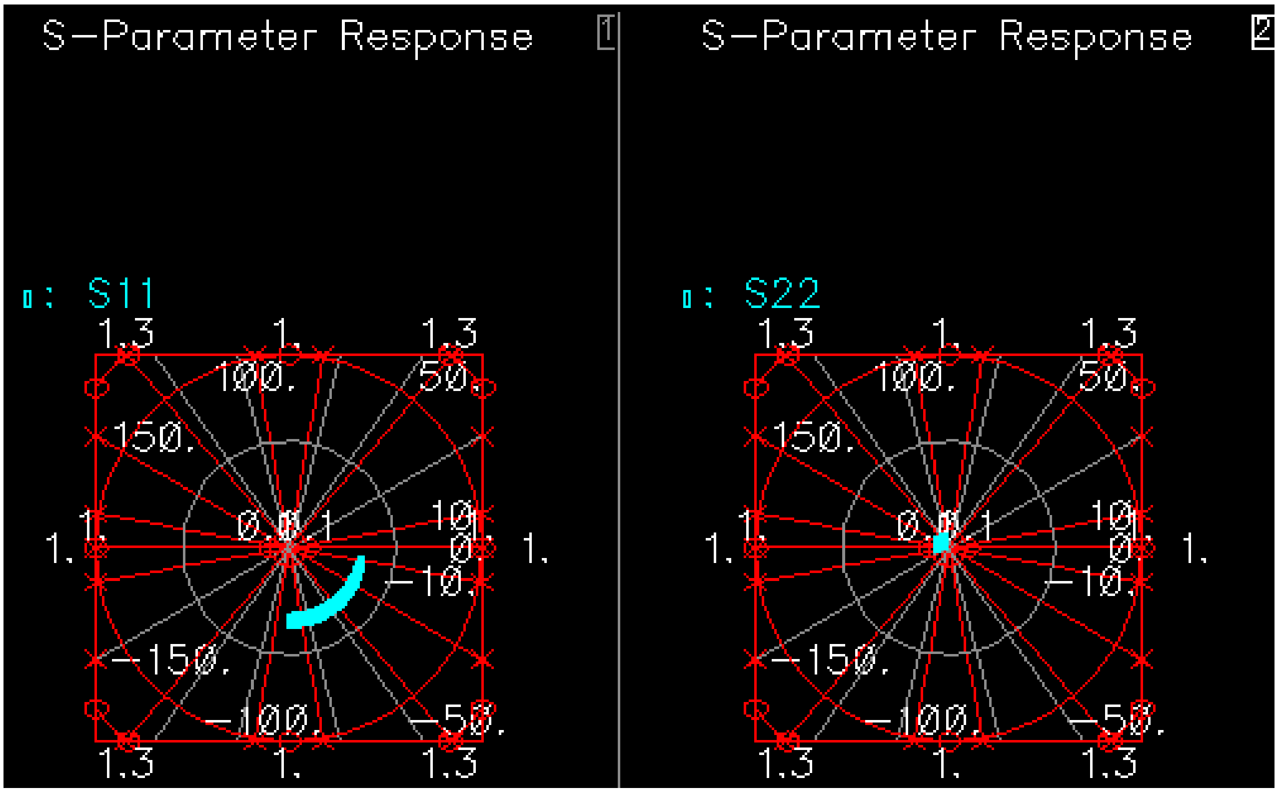
**b.** Click *S22*.

S22 is plotted in the right half of the waveform window.

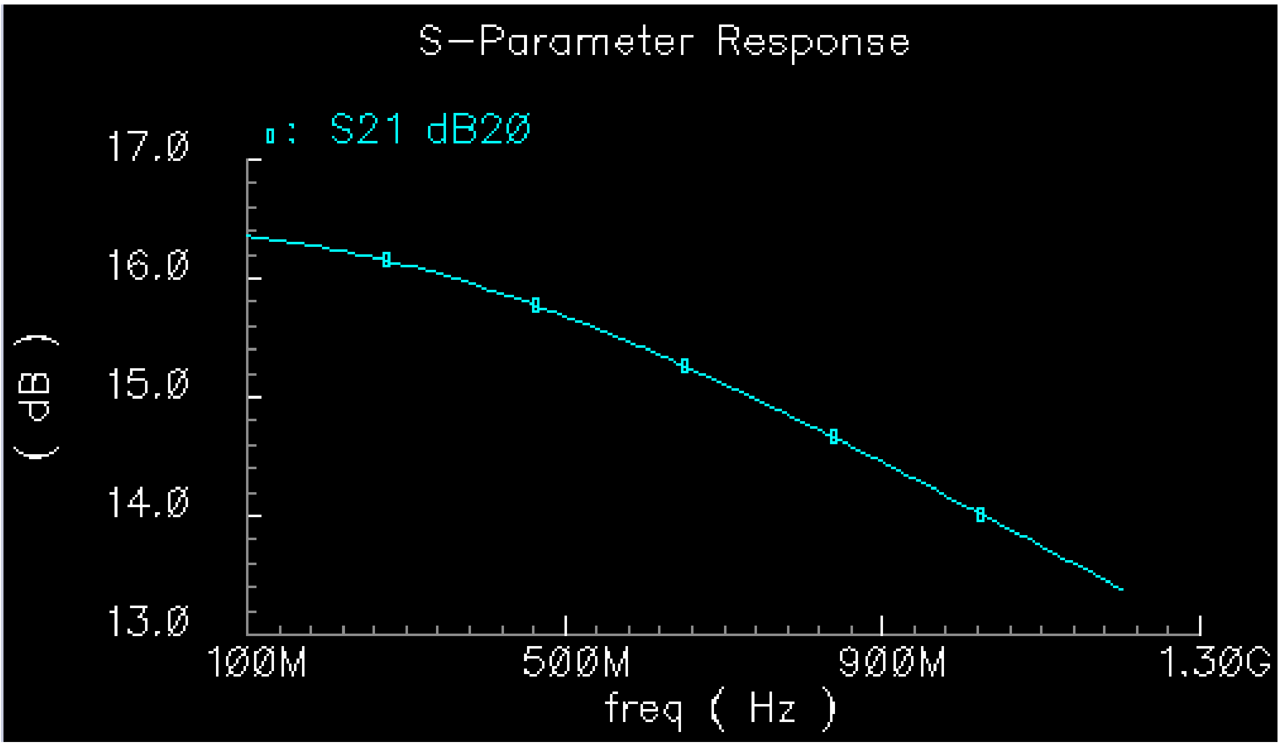



The values of S11 and S22 measure whether input and output are matched.

You can choose among several different plot types in the Direct Plot form. The following plots show S11 and S22 with the *Plot Type* set to *Polar*.



You can also choose S12 and S21 for plotting in the Direct Plot form. S12 is the reverse gain and a measure of isolation.




## Plotting the Voltage Standing Wave Ratio

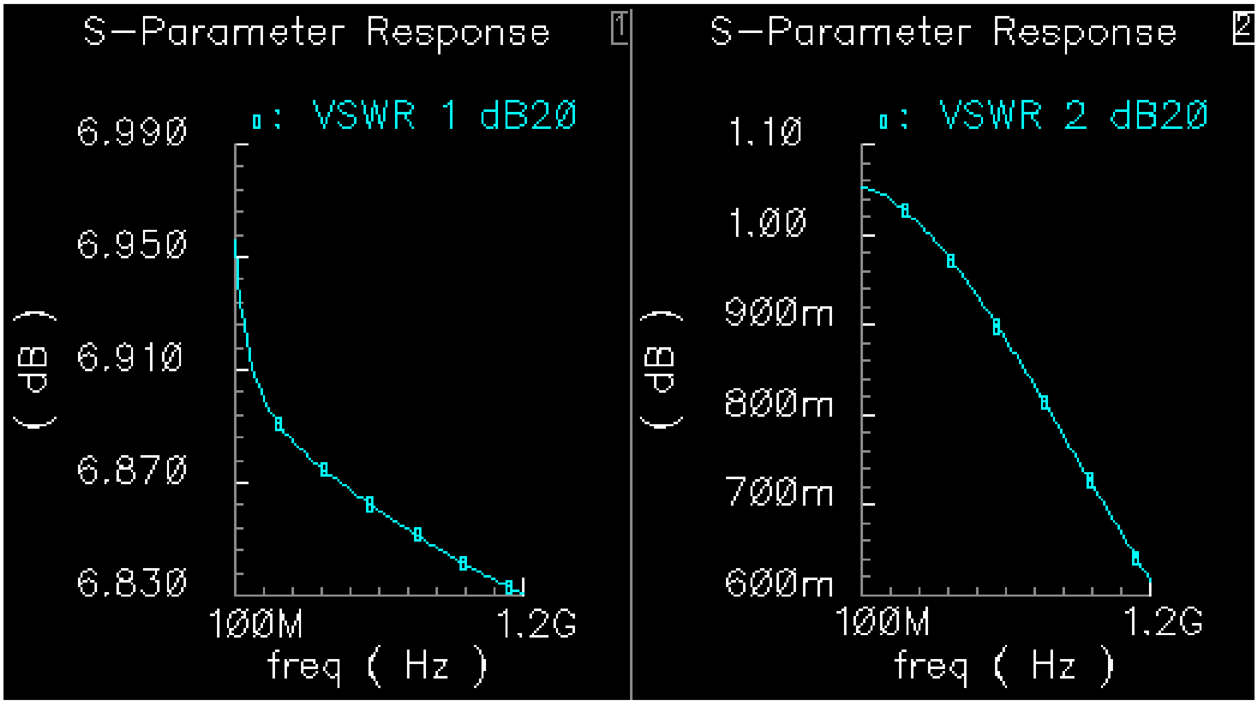
In the Direct Plot form do the following.

1. Highlight *New SubWin* for *Plotting Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *VSWR* for *Function*.
4. Click *VSWR1*.

   *VSWR1* is plotted in the waveform window.
5. Click *VSWR2*.

   *VSWR2* is plotted in the right half of the waveform window.

The two voltage standing wave ratio plots display in the waveform window.




# Linear Two-Port Noise Analysis with S-Parameters

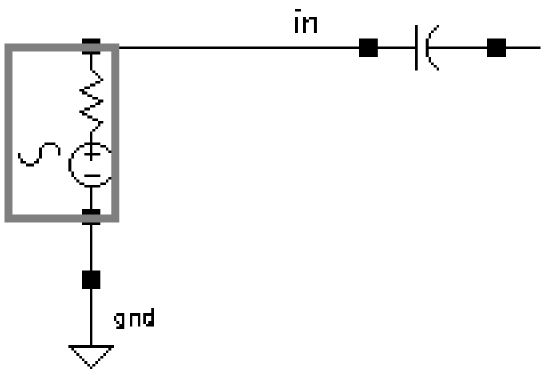
## Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

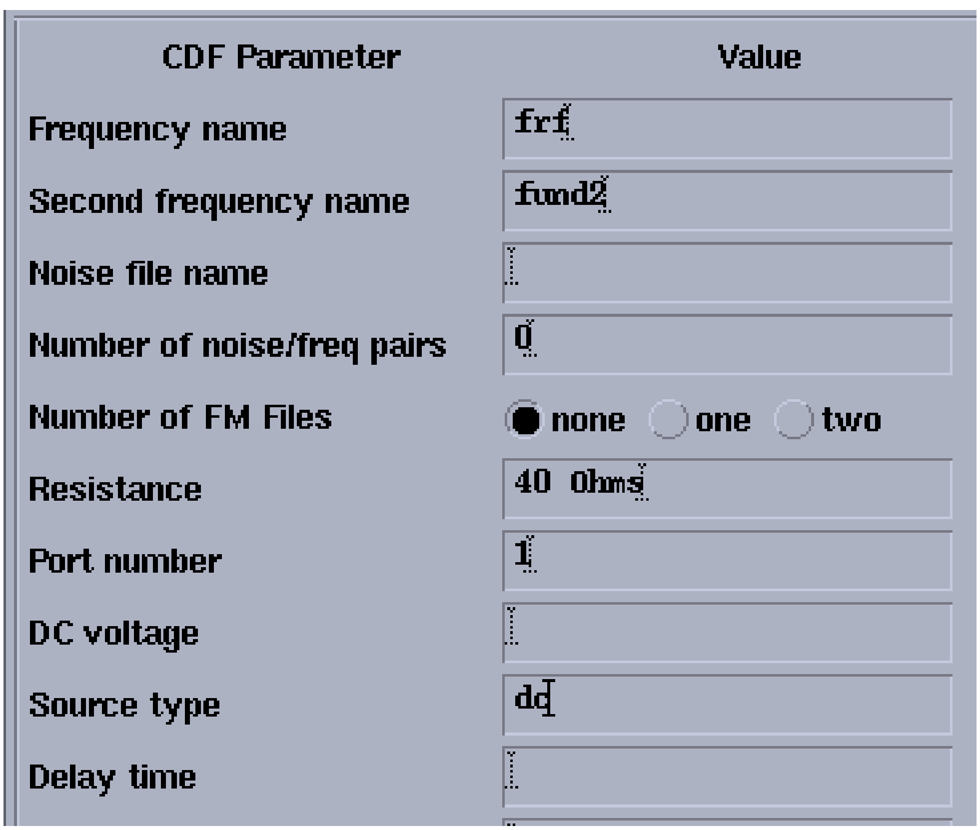
## Editing the Schematic

1. In the Schematic window, click the RF voltage source.




2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.



| CDF Parameter | Value |
|---|---|
| Frequency name | frf |
| Second frequency name | fund2 |
| Noise file name | |
| Number of noise/freq pairs | 0 |
| Number of FM Files | none / one / two |
| Resistance | 40 Ohms |
| Port number | 1 |
| DC voltage | |
| Source type | dc |
| Delay time | |

3. In the *Source type* field, type dc.

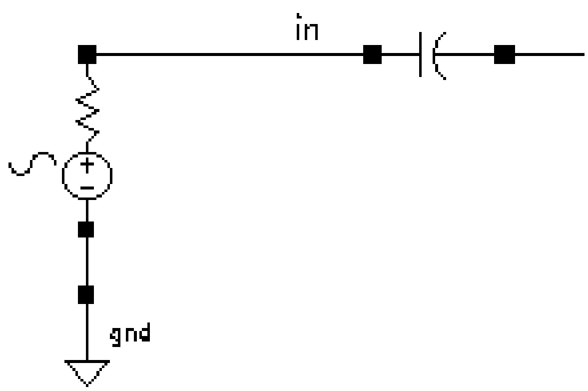
4. Click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

### Note on the isnoisy Parameter

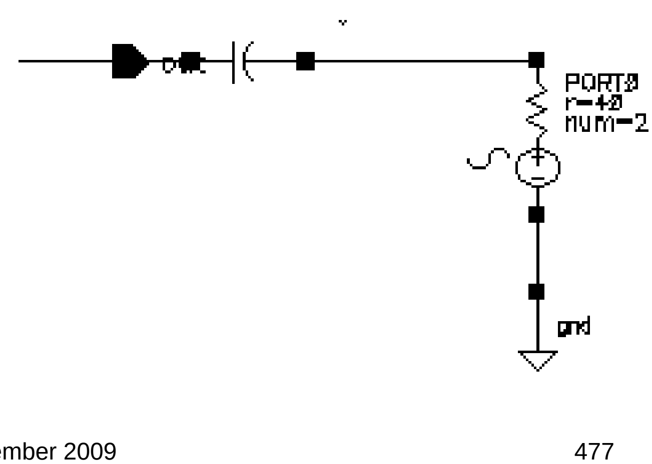
The *isnoisy* parameter, which you can set on the Edit Object Properties form by using the *Generate noise?* field, specifies whether a resistor generates noise. If you set the parameter to *yes* or *no*, the parameter setting appears in the netlist. If you leave the parameter unspecified by using the blank value in the cyclic field, the default setting is *yes* but the parameter does not appear in the netlist.

## Setting up the S-Parameter Analysis

1. In the ADE window, choose *Analyses – Choose*.
2. In the Choosing Analyses form, click *sp* for the *Analysis* choice.
3. Click *Select* in the *Ports* pane. Then, in the schematic, click on the input port (`RF`)

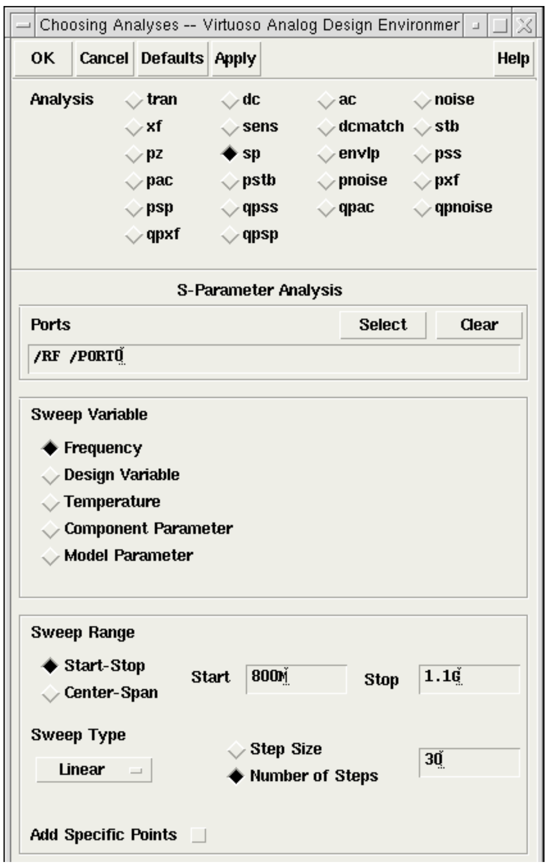



and then on the output port (`PORT0`)



In response, `/RF /PORT0` appears in the Ports field.

4. Highlight *Frequency* for the *Sweep Variable*.
5. Highlight *Start-Stop* for the *Sweep Range*. Type `800M` in the *Start* field and `1.1G` in the *Stop* field.
6. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `30` in the *Number of Steps* field.
7. The top of the SP Choosing Analyses form appears as follows.




8. Highlight *yes* for *Do Noise*.

   The form changes to let you specify values for the *Output port* and *Input port* fields.

9. To select the Output port,

   a. Click the *Select* button next to the *Output port* field.

   b. In the Schematic window, select the output port.

      `/PORT0` displays in the *Output port* field.

**10.** To select the Input port,

  **a.** Click the *Select* button next to the *Input port* field.

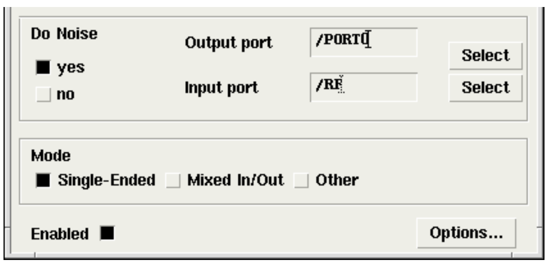
  **b.** In the Schematic window, select the RF port.

  `/RF` displays in the *Input port* field.

**11.** For the *Mode*, select *Single-Ended*.

**12.** Make sure that *Enabled* is highlighted.

The bottom of the SP Choosing Analyses form appears as follows.




**13.** In the Choosing Analyses form, click *OK*.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

The output log file appears and displays information about the simulation as it runs.

**2.** Look in the CIW for a message that says the simulation is completed successfully.

## Plotting the Noise Figure and Minimum Noise Figure

**1.** In the ADE window, choose *Results – Direct Plot – Main Form*.

The Direct Plot form appears.

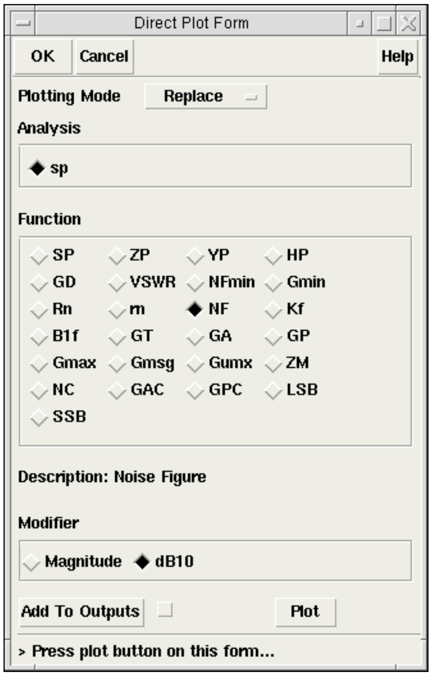
**2.** In the Direct Plot form, highlight *Replace* for *Plotting Mode*.

**3.** Highlight *sp* for *Analysis*.

**4.** Highlight *NF* for *Function*.

**5.** Highlight *dB10* for *Modifier*.

The completed form looks like this.

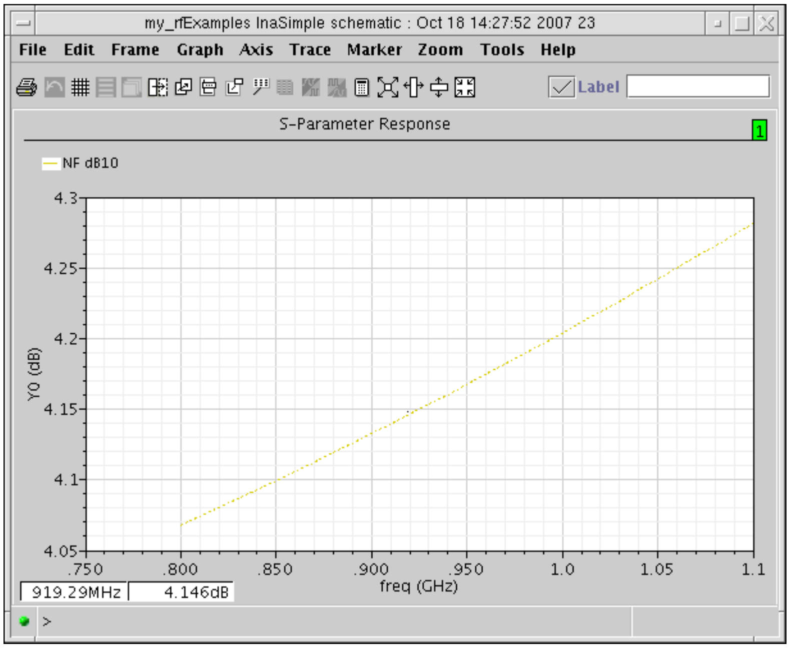



**6.** Following the message at the bottom of the form,

*Press Plot button on this form...*

Click *Plot*.

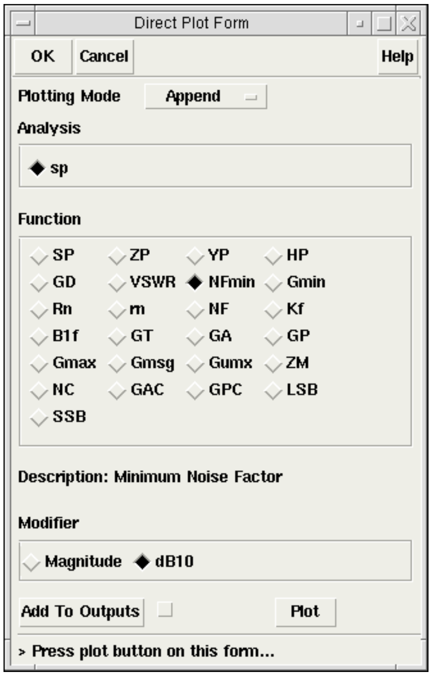
The plot for the noise figure appears in the waveform window.




To add the minimum noise figure plot to the waveform window, in the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *NFmin* for *Function*.
4. Highlight *dB10* for *Modifier*.

The completed form looks like this.

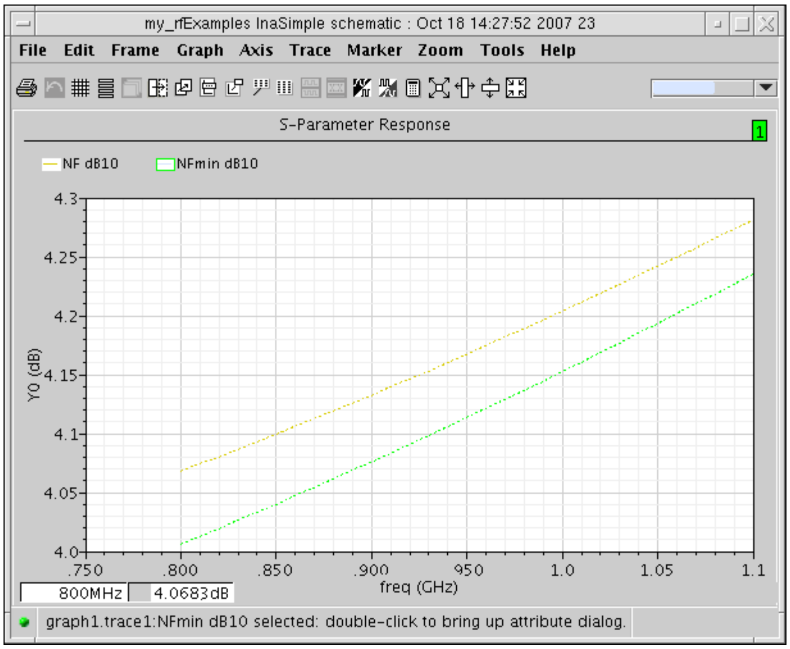



**5.** Following the message at the bottom of the form,

*Press Plot button on this form...*

Click *Plot*.

The plot for the minimum noise figure is added to the noise figure plot in the waveform window.

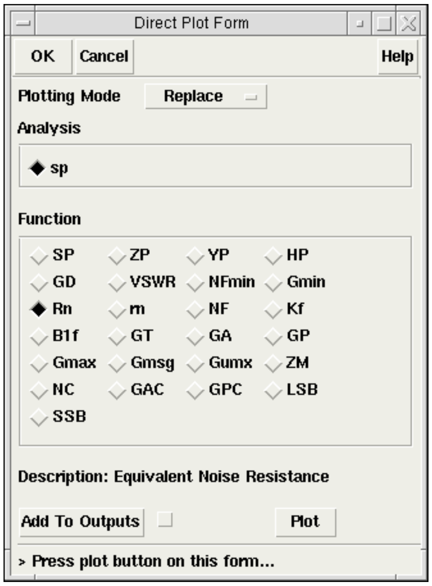



## Plotting the Equivalent Noise Resistance

In the Direct Plot form, do the following:

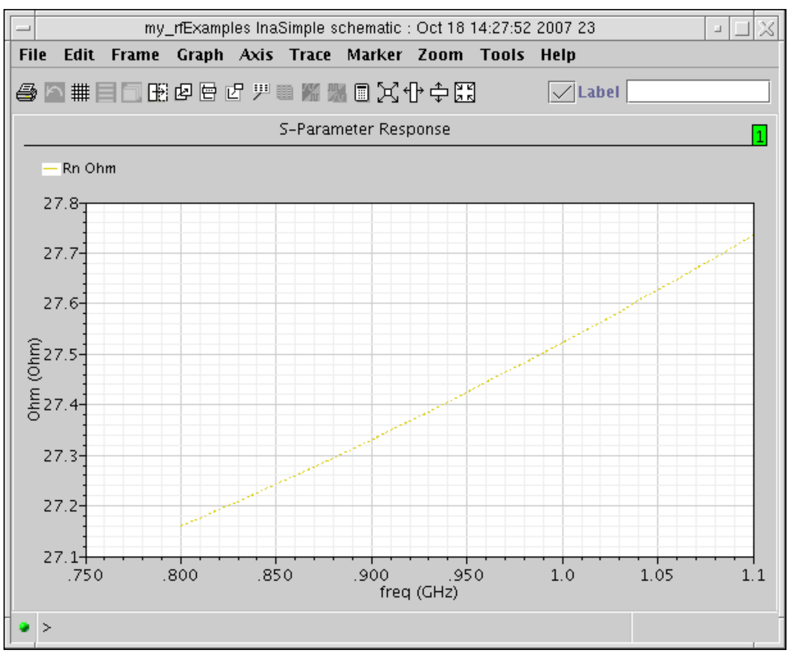
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *Rn* for *Function*.

The completed form looks like this.




4. In the Direct Plot form, click *Plot*.

   The equivalent noise resistance is plotted in the waveform window.



## Plotting Load and Source Stability Circles

Load and source stability circles show the boundaries between values of load and source impedance that cause instability and values that do not. Either the inside or the outside of the circles might represent the stable region, depending on other values. If, as in the example below, the values of S-parameters $S_{11}$ and $S_{22}$ are less than one for a 50 Ohm impedance system, the center of the normalized Smith chart falls within the stable region.

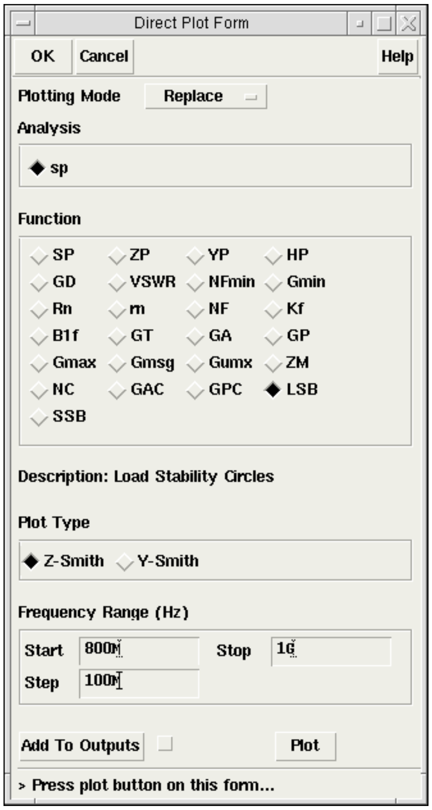
### To Plot the Load Stability Circles

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode.*
2. Highlight *sp* for *Analysis*.
3. Highlight *LSB* for *Function*.
4. Highlight *Z-Smith* for *Plot Type*.

   The Direct Plot form changes to accept *Frequency Range (Hz)* values.
5. For the *Frequency Range (Hz)* values
   a. Type `800M` in the *Start* field.
   b. Type `1G` in the *Stop* field.
   c. Type `100M` in the *Step* field.

The Direct Plot form looks like this.

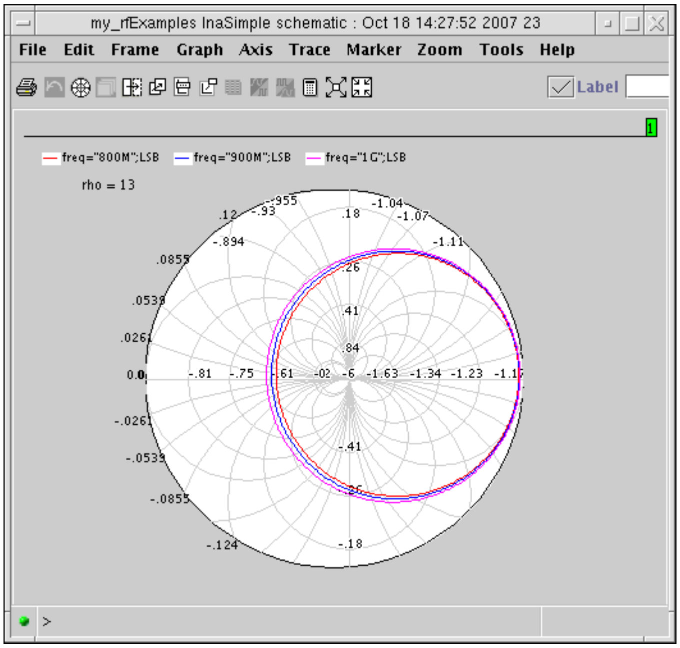



**6.** Following the message at the bottom of the form,

*Press Plot button on this form...*

Click *Plot*.

The load stability circles are plotted.

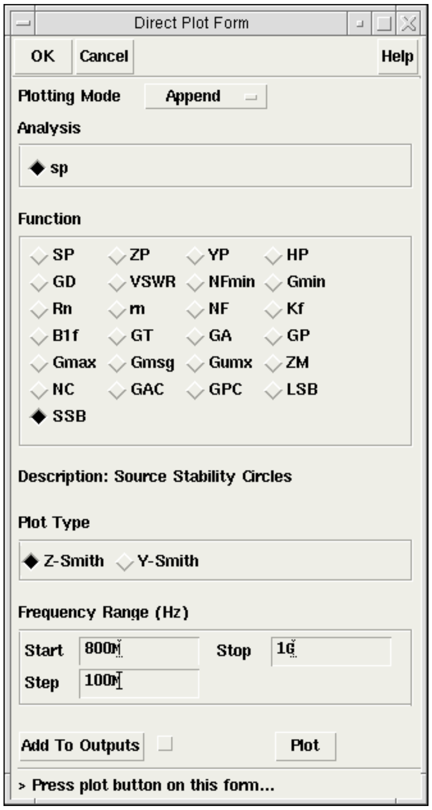



### To Plot the Source Stability Circles

In the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *SSB* for *Function*.
4. Highlight *Z-Smith* for *Plot Type*.
5. For the *Frequency Range (Hz)* values
   a. Type `800M` in the *Start* field.
   b. Type `1G` in the *Stop* field.
   c. Type `100M` in the *Step* field.

The Direct Plot form looks like this.



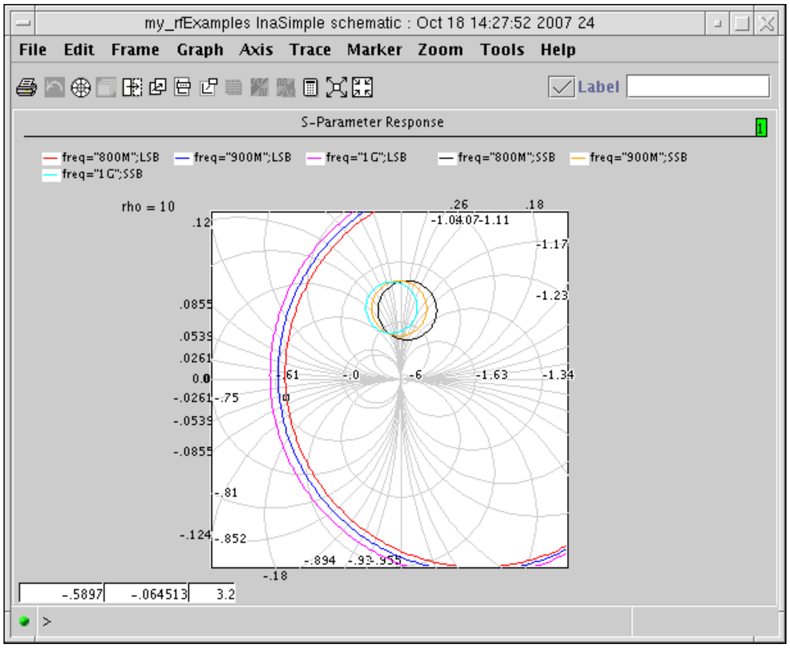

6. Following the message at the bottom of the form,

   *Press Plot button on this form...*

   Click *Plot*.

   The source stability circles are plotted in the waveform window along with the load stability circles.

**7.** To see the plot as it appears below, in the waveform window, choose *Zoom – Fast Zoom Out* three times consecutively.




In the center of the Smith Chart, the values of S11 and S22 are greater than 1, so the center of the Smith chart is part of the unstable region. Therefore, all points inside the load stability circle and all points outside the source stability circle belong to the unstable region.

## Plotting the Noise Circles

In the Direct Plot form, do the following:

**1.** Highlight *Replace* for *Plot Mode*.

**2.** Highlight *sp* for *Analysis*.

**3.** Highlight *NC* for *Function*.

**4.** Highlight *Z-Smith* for *Plot Type*.

**5.** Highlight *Noise Level (dB)* for *Sweep*.

**6.** Type `900M` for *Frequency (Hz)*.

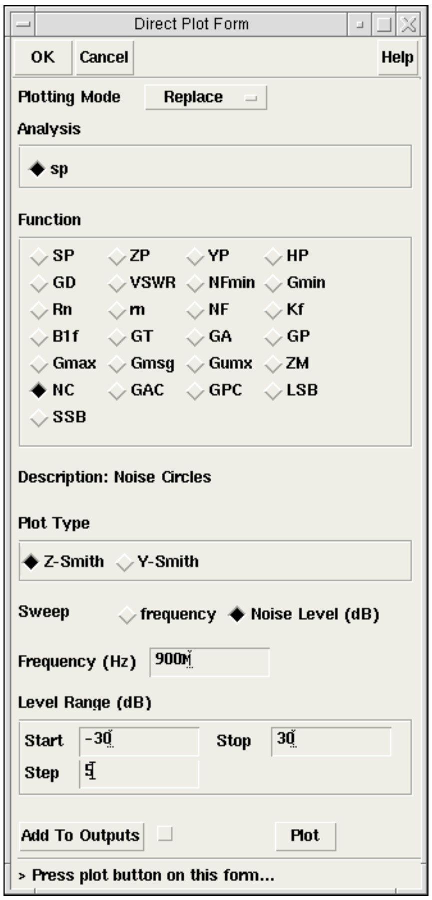
**7.** For the *Level Range (dB)* values

  **a.** Type `-30` in the *Start* field.

**b.** Type `30` in the *Stop* field.

**c.** Type `5` in the *Step* field.

The Direct Plot form looks like this.

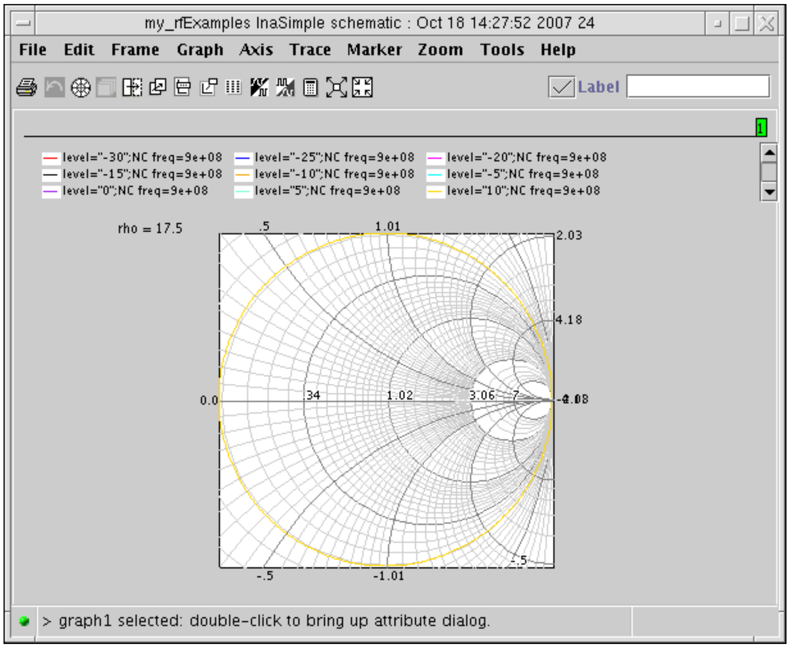



**8.** Following the message at the bottom of the form,

*Press Plot button on this form...*

Click *Plot*.

The noise circles are plotted in the waveform window.




# Noise Calculations with PSS and Pnoise

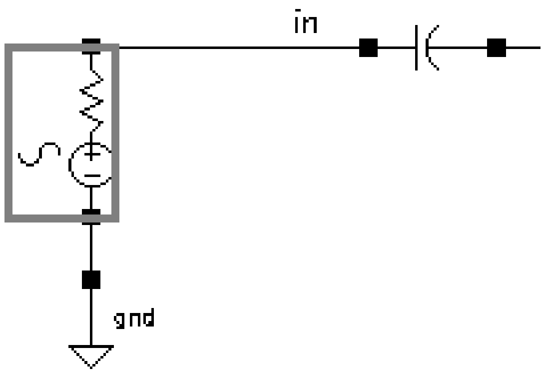
## Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

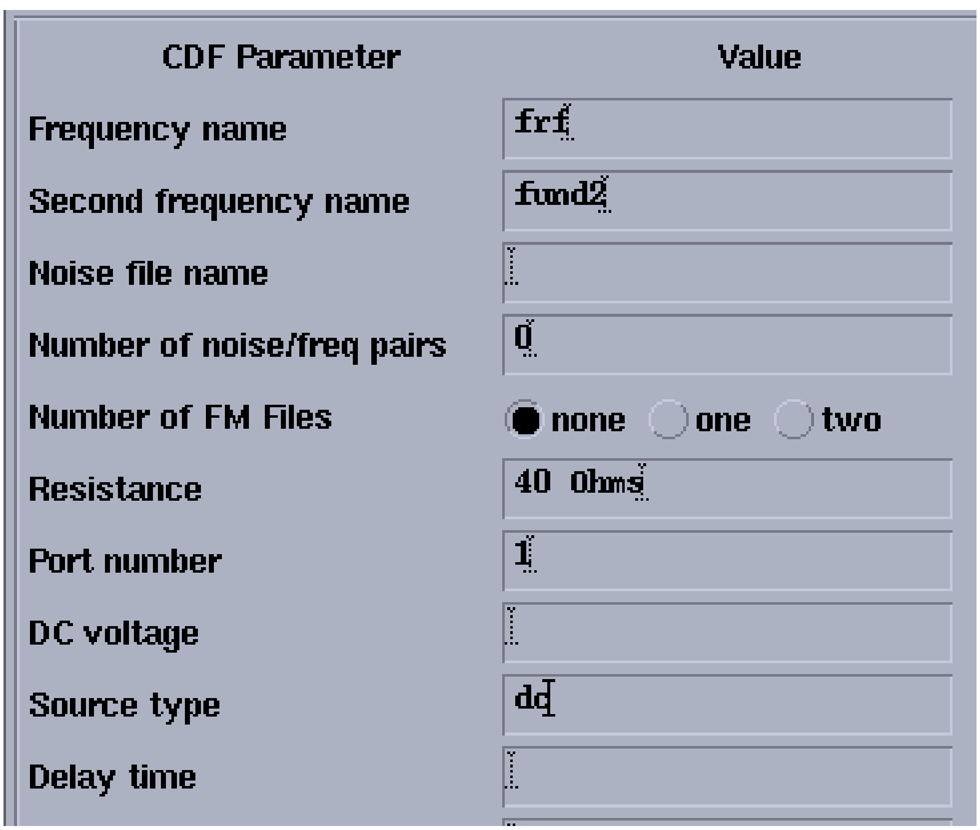
## Editing the Schematic

1. In the Schematic window, click the RF voltage source.




2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.



| CDF Parameter | Value |
|---|---|
| Frequency name | frf |
| Second frequency name | fund2 |
| Noise file name | |
| Number of noise/freq pairs | 0 |
| Number of FM Files | none / one / two |
| Resistance | 40 Ohms |
| Port number | 1 |
| DC voltage | |
| Source type | dc |
| Delay time | |

3. In the *Source type* field, type dc.

**4.** Click *OK*.

**5.** In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS and Pnoise Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

### Setting up the PSS Analysis

**1.** In the Choosing Analyses form, click *pss*.

**2.** The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

When the *Source type* is `dc`, you must enter the fundamental frequency manually.

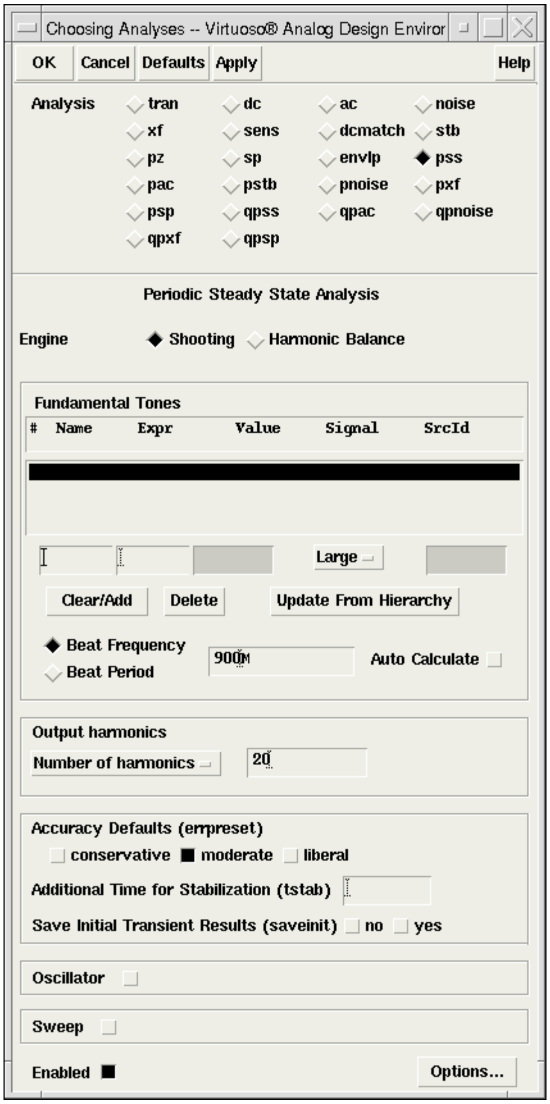
**3.** Be sure the *Auto Calculate* button is *not* highlighted.

**4.** In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `20`, in the *Number of harmonics* field.

**5.** Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

**6.** Highlight *Enabled*.

The Choosing Analyses form for PSS appears as follows.




## Setting up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the periodic noise (Pnoise) analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*. Type `800M` in the *Start* field and `1.2G` in the *Stop* field.

3. In the *Sweep Type* cyclic field, choose *Linear* and click the *Number of Steps* button. Type `201` in the *Number of Steps* field.

**4.** In the *Sidebands* cyclic field choose *Maximum sideband* and type `20` in the *Maximum sideband* field.

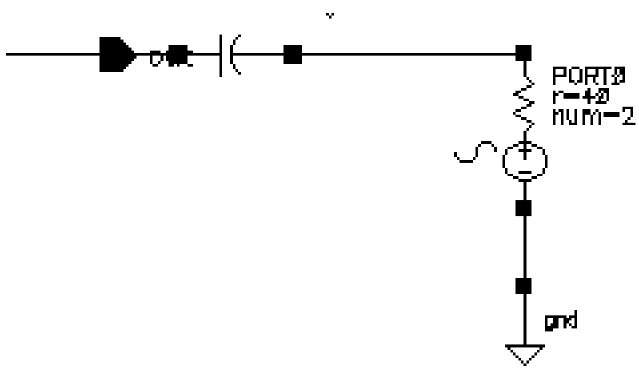
**5.** In the Output cyclic field, choose *voltage*.

The value is appropriate because the reference sideband is an LNA. The form changes to display positive and negative output node fields.

**6.** To select the *Positive Output Node*

**a.** Click the *Select* button next to the *Positive Output Node* field.

**b.** In the Schematic window, select the amplifier output net.




`/net28` displays in the *Positive Output Node* field.

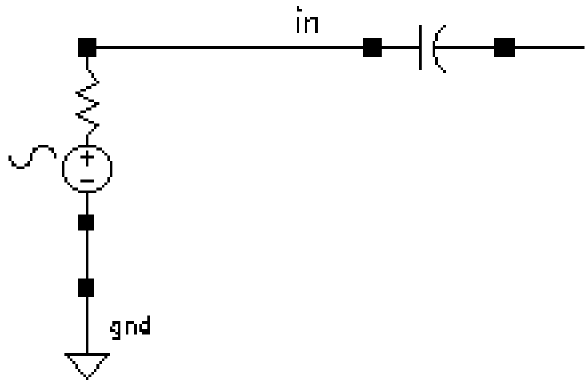
**c.** Leave the *Negative Output Node* field empty. It defaults to `/gnd!`.

You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

**7.** To select the *Input Voltage Source*

**a.** Click the *Select* button next to the *Input Voltage Source* field.

**b.** In the Schematic window, select the RF port.



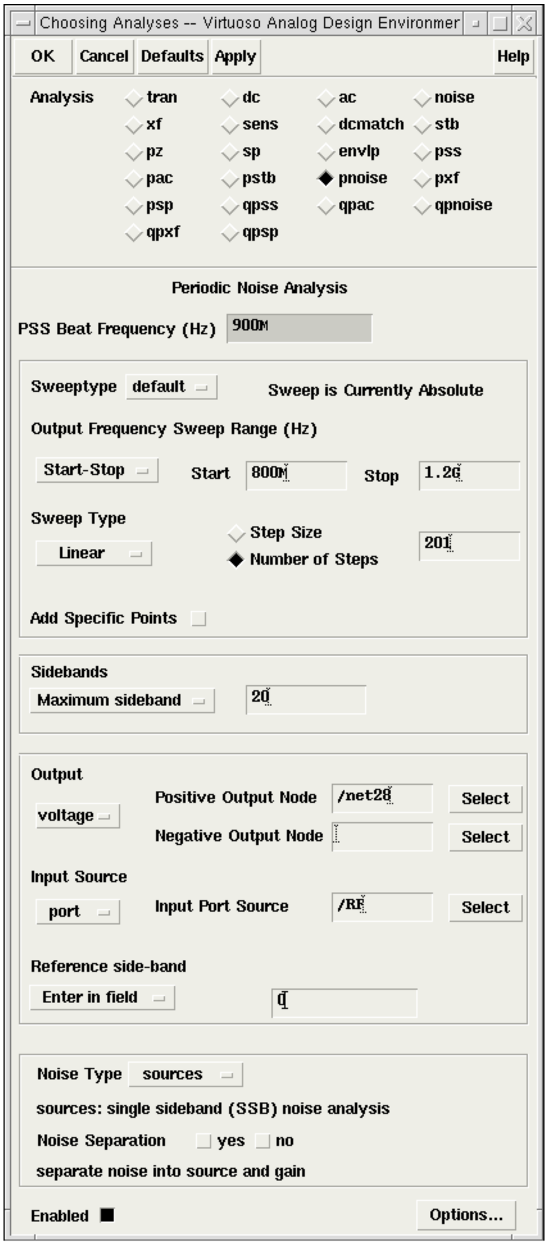

`/RF` displays in the *Input Voltage Source* field.

**8.** In the *Reference side-band* field, type `0`.

You specify the reference sideband as `0` for an LNA because an LNA has no frequency conversion from input to output.

**9.** Highlight *Enabled*.

The Choosing Analyses form for Pnoise appears as follows.




**10.** In the Choosing Analyses form, click *OK*.

## Running the Simulation

**1.** To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation is completed successfully.

## Plotting the Noise Calculations

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

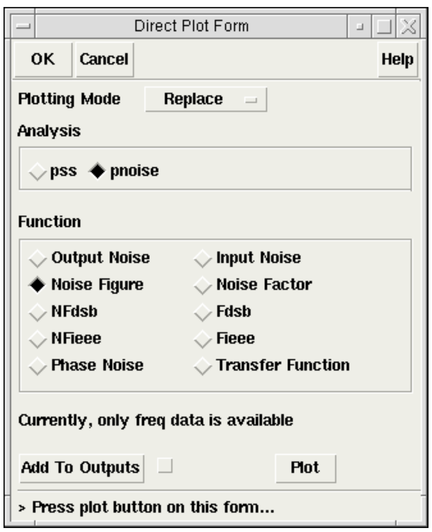
   The Direct Plot form appears.

2. In the Direct Plot form, select *Replace* for *Plotting Mode*.
3. Highlight *pnoise* for *Analysis*.

   The form changes to display fields for the Pnoise analysis.

4. Highlight *Noise Figure* for *Function*.

   The completed form looks like this.

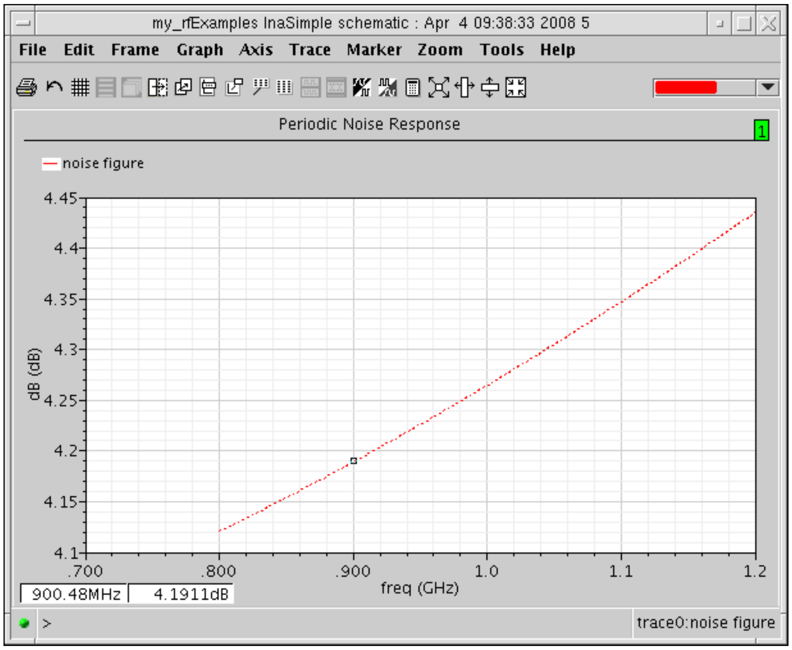



5. Follow the message at the bottom of the form

   *Press plot button on this form...*

   Click *Plot*.

The plot appears in the waveform window. The Noise Figure at 900MHz is about 4.19 dB.




6. Close the Direct Plot form and the waveform window.

## Printing the Noise Summary

1. In the ADE window, choose *Results – Print – Noise Summary*.

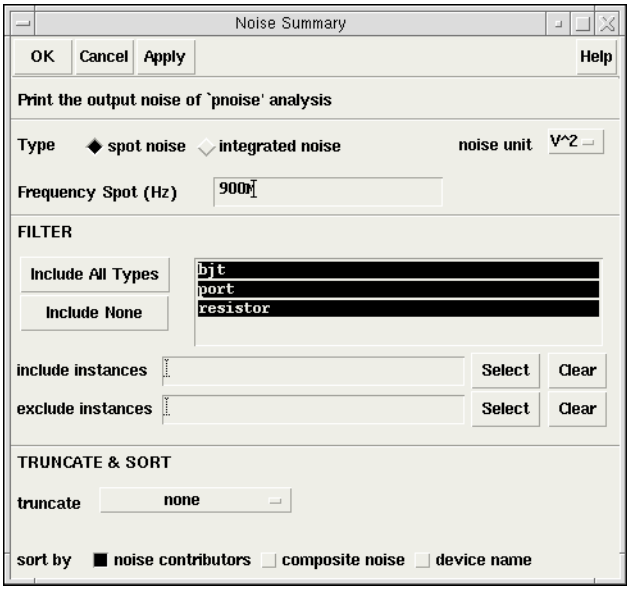
   The Results Display Window and the Noise Summary form appear.

2. In the Noise Summary form, highlight *spot noise* for *Type*.
3. Type `900M` in the *Frequency Spot (Hz)* field.
4. Click *Include All Types* in the *FILTER* list box.

   The *Include All Types* is the *FILTER* list box default, however, you must click the button to activate the choice.

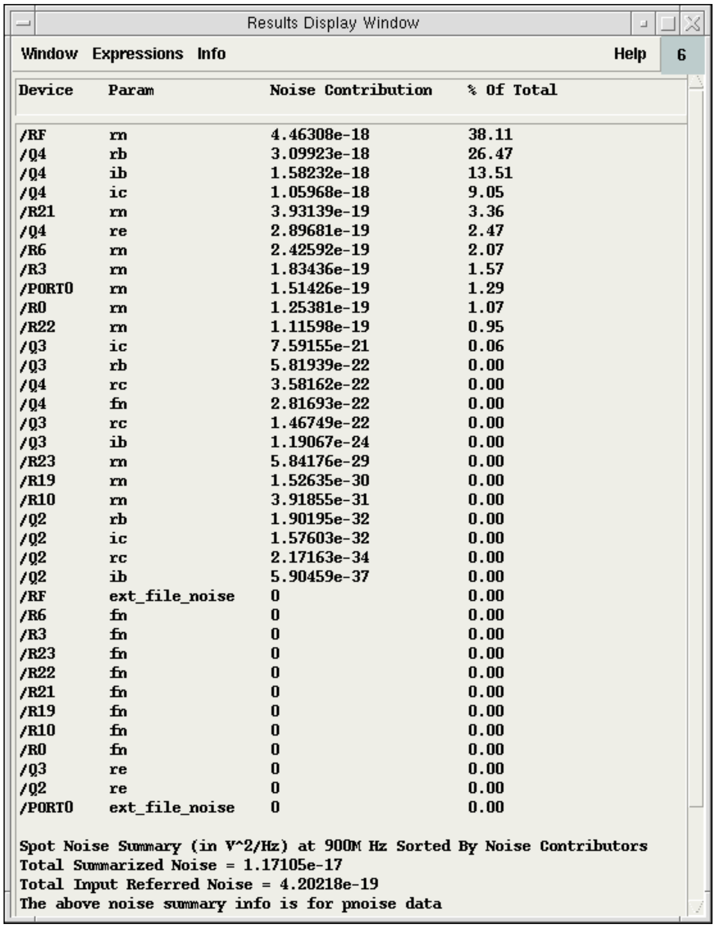
5. In the *truncate* cyclic field, select *none* in the *TRUNCATE & SORT* section.

The filled-in Noise Summary form looks like this.




**6.** In the Noise Summary form, click *OK*.

The noise contributors now appear in the Results Display window.




| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /RF | rn | 4.46308e-18 | 38.11 |
| /Q4 | rb | 3.09923e-18 | 26.47 |
| /Q4 | ib | 1.58232e-18 | 13.51 |
| /Q4 | ic | 1.05968e-18 | 9.05 |
| /R21 | rn | 3.93139e-19 | 3.36 |
| /Q4 | re | 2.89681e-19 | 2.47 |
| /R6 | rn | 2.42592e-19 | 2.07 |
| /R3 | rn | 1.83436e-19 | 1.57 |
| /PORT0 | rn | 1.51426e-19 | 1.29 |
| /R0 | rn | 1.25381e-19 | 1.07 |
| /R22 | rn | 1.11598e-19 | 0.95 |
| /Q3 | ic | 7.59155e-21 | 0.06 |
| /Q3 | rb | 5.81939e-22 | 0.00 |
| /Q4 | rc | 3.58162e-22 | 0.00 |
| /Q4 | fn | 2.81693e-22 | 0.00 |
| /Q3 | rc | 1.46749e-22 | 0.00 |
| /Q3 | ib | 1.19067e-24 | 0.00 |
| /R23 | rn | 5.84176e-29 | 0.00 |
| /R19 | rn | 1.52635e-30 | 0.00 |
| /R10 | rn | 3.91855e-31 | 0.00 |
| /Q2 | rb | 1.90195e-32 | 0.00 |
| /Q2 | ic | 1.57603e-32 | 0.00 |
| /Q2 | rc | 2.17163e-34 | 0.00 |
| /Q2 | ib | 5.90459e-37 | 0.00 |
| /RF | ext_file_noise | 0 | 0.00 |
| /R6 | fn | 0 | 0.00 |
| /R3 | fn | 0 | 0.00 |
| /R23 | fn | 0 | 0.00 |
| /R22 | fn | 0 | 0.00 |
| /R21 | fn | 0 | 0.00 |
| /R19 | fn | 0 | 0.00 |
| /R10 | fn | 0 | 0.00 |
| /R0 | fn | 0 | 0.00 |
| /Q3 | re | 0 | 0.00 |
| /Q2 | re | 0 | 0.00 |
| /PORT0 | ext_file_noise | 0 | 0.00 |

Spot Noise Summary (in V^2/Hz) at 900M Hz Sorted By Noise Contributors
Total Summarized Noise = 1.17105e-17
Total Input Referred Noise = 4.20218e-19
The above noise summary info is for pnoise data

This summary helps you determine the percentage of noise contributed by the different devices in the circuit.

# Plotting the 1dB Compression Point

## Setting Up the Simulation

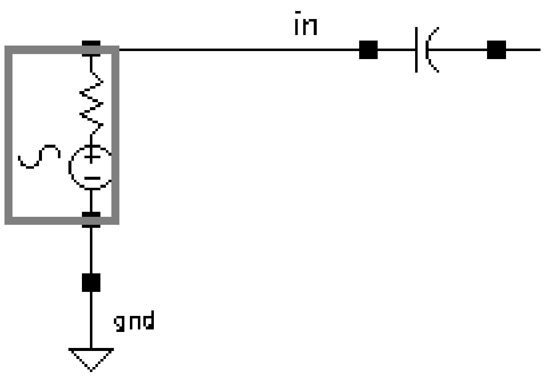
1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.

**4.** In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

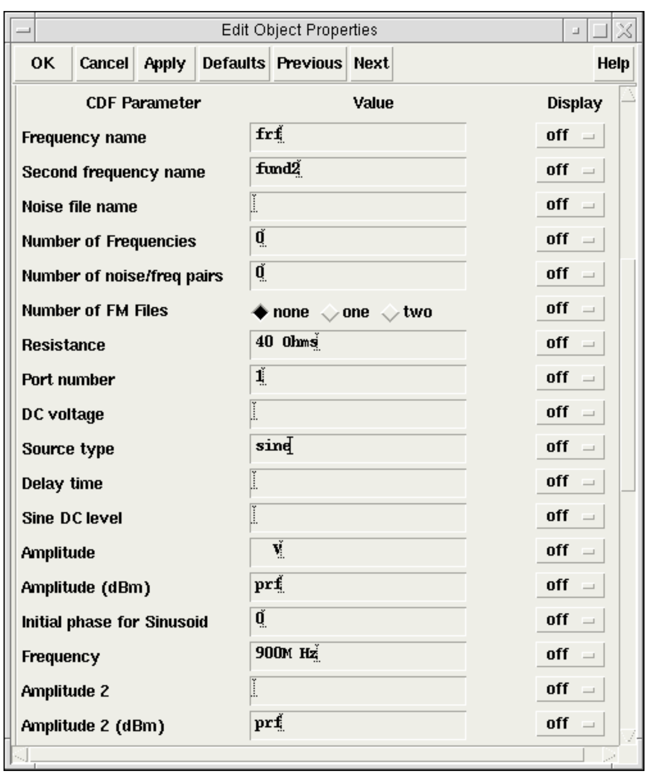
## Editing the Schematic

**1.** In the Schematic window, click the RF voltage source.




**2.** In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



3. In the *Source type* field, type `sine`.
4. Type `900M` for the *Frequency* value.
5. Type `prf` for the *Amplitude (dBm)* value.
6. Be sure the *Amplitude* field is empty.
7. In the Edit Objects Properties form, click *OK.*
8. In the Schematic window, choose *Design – Check and Save*.

## Setting up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click *pss*.

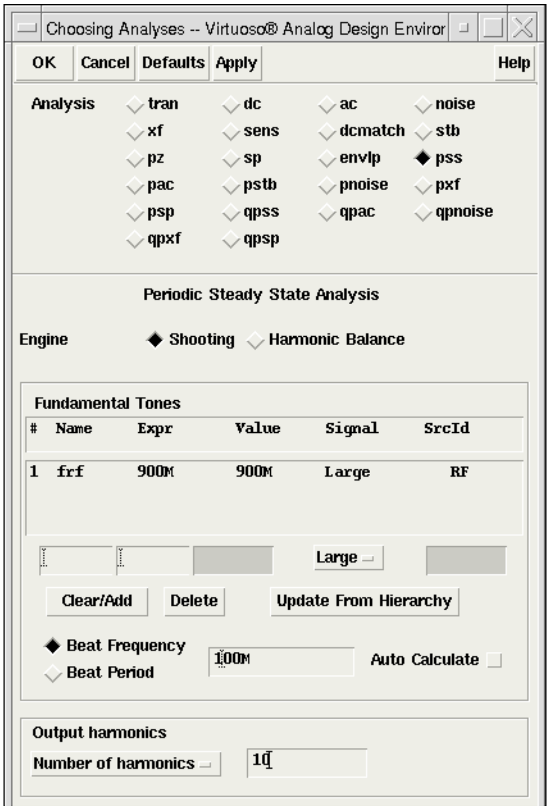
   The form changes to let you specify data for the PSS analysis.

3. Type `100M` for the *Fundamental Frequency*.

   The *Beat Frequency* button is highlighted by default.

4. Be sure the *Auto Calculate* button is *not* highlighted.
5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `10`, in the *Number of harmonics* field.

The top of the PSS form appears like the one below.




6. Highlight *moderate* for the *Accuracy Default (errpreset).*
7. Highlight the *Sweep* button.

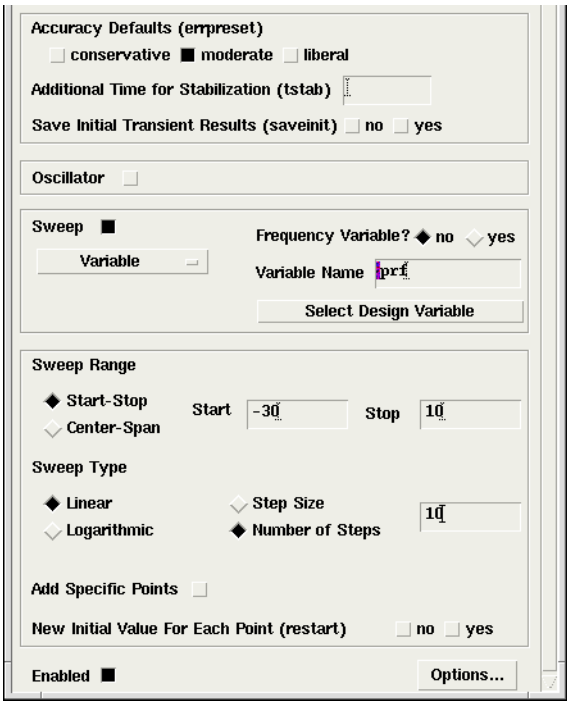
   The form changes to let you specify data for the sweep.
8. In the *Sweep* cyclic field, choose *Variable*.
9. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
10. In the Select Design variable form, highlight *prf* and click *OK*.

    The *prf* variable is the amplitude, in dBm, of the RF input source. If you are not sure of this, use the Edit Object Properties form to examine the *Amplitude (dBm)* field for the RF voltage source.
11. Choose *Start-Stop* for the *Sweep range* and type `-30` in the *Start* field and `10` in the *Stop* field.

**12.** Choose *Linear* for the *Sweep Type* and click the *Number of Steps* button. Type `10` as the number of steps in the field.

The bottom of the PSS form appears like the one below.




**13.** In the Choosing Analyses form, click *OK*.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the 1dB Compression Point

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *pss* for *Analysis*.
3. Highlight *Compression Point* for *Function*.

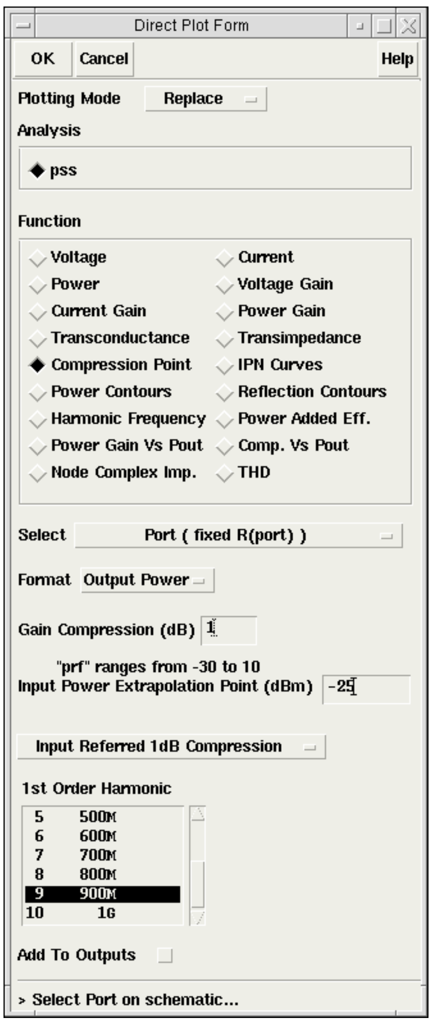
4. Type -25 for the *Input Power Extrapolation Point (dBm)*.

   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.

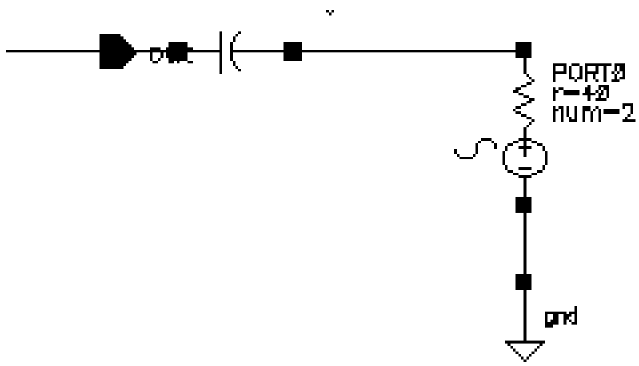
5. Choose *Input Referred 1dB Compression*.

6. Click *900M* in the *1st Order Harmonic* list box.

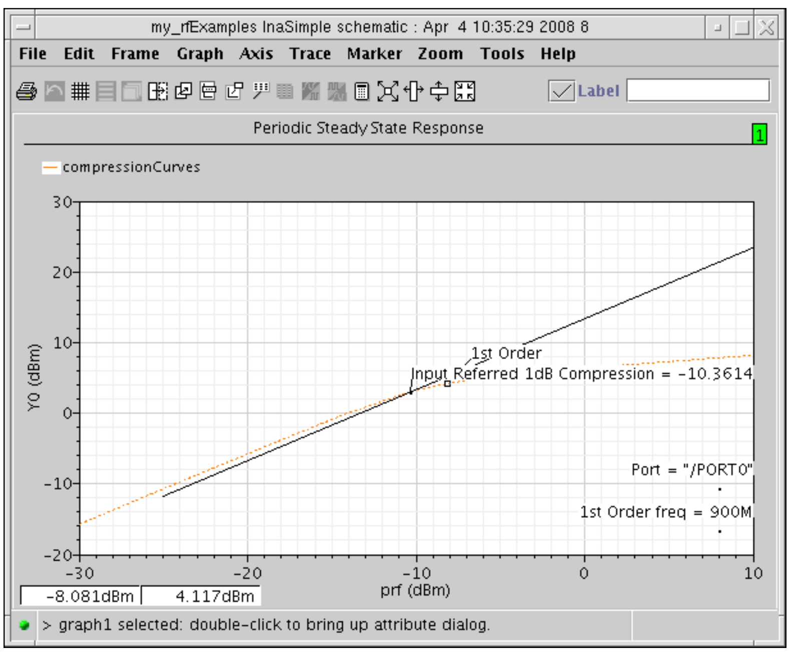
   The filled-in form looks like this.



7. In the Schematic Window, click the `PORT0` output port.




The plot appears in the waveform window.




The Spectre RF simulator calculates the 1dBCompressionPoint as -10.3614.

# Calculating the Third-Order Intercept Point with Swept PSS

The method illustrated here uses only swept PSS analysis. An alternative method, which runs more quickly if the input signals are very close together, uses both swept PSS and PAC analyses and is described in .Third-Order Intercept Measurement with Swept PSS and PAC on page 318

## Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

## Editing the Schematic

1. In the Schematic window, click the input voltage source and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
2. In the Edit Object Properties form, choose *sine* for the *Source Type*.
3. Type `fund2`, or any name you choose, in the *Second frequency name* field.
4. Type `prf` for the *Amplitude (dBm)* and *Amplitude 2 (dBm)* values.
5. Type `900M` for the *Frequency* value.
6. Type `920M` for the *Frequency 2* value.
7. Be sure the *Amplitude* and *Amplitude2* fields are empty.
8. In the Edit Object Properties form, click *OK*.
9. In the Schematic window, choose *Design – Check and Save*.

## Setting up the Swept PSS Analysis

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

In the Choosing Analyses form, do the following and then click *OK*:

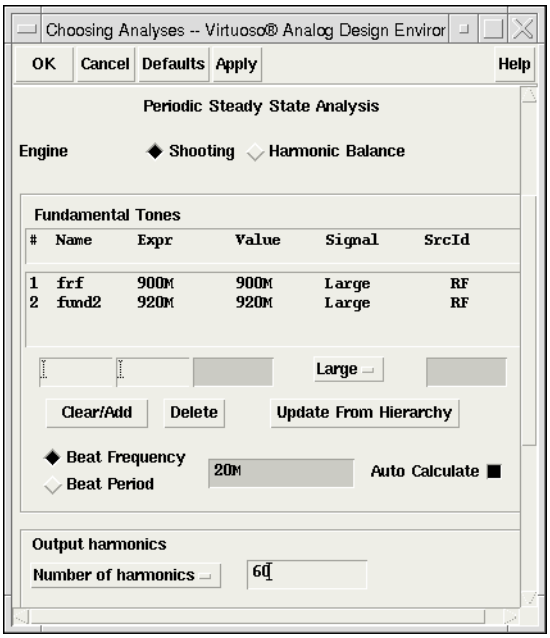
1. In the Choosing Analyses form, click *pss*.
2. In the *Fundamental Tones* list box, be sure the *Auto calculate* button is highlighted.

The fundamental frequency, 20M, is displayed.

The *Beat Frequency* button is highlighted by default.

3. In the *Output harmonics* cyclic field, choose *Number of harmonics* for the *Output harmonics* choice and type in a reasonable value, such as `60`, in the field.

   The top of the PSS form looks like this.

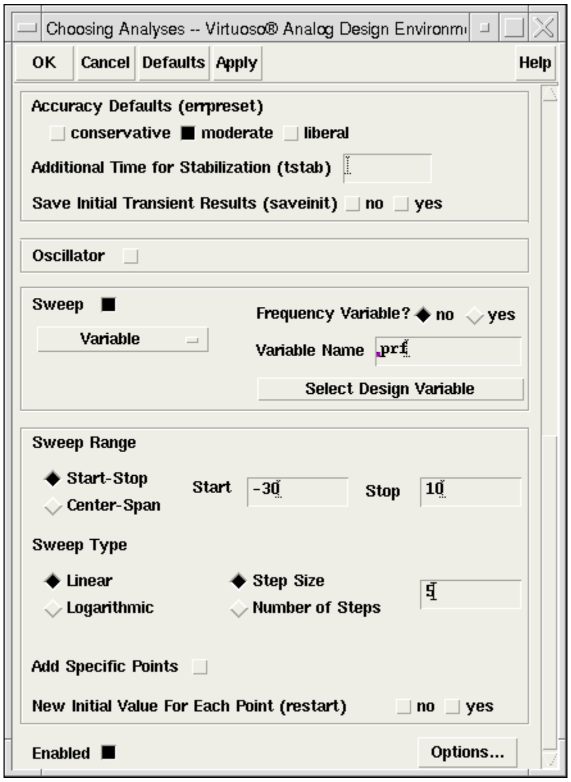



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
8. In the Select Design Variable form, highlight *prf* and click *OK*.
9. Choose *Start-Stop* for the *Sweep Range*, and type `-30` as the *Start* value and type 10 as the *Stop* value.
10. Choose *Linear* for the *Sweep Type* and click the *Step Size* button. Type 5 as the step size.

The bottom of the PSS form looks like this.




**11.** Click *OK*.

The Choosing Analyses form closes.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Third-Order Intercept Point

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

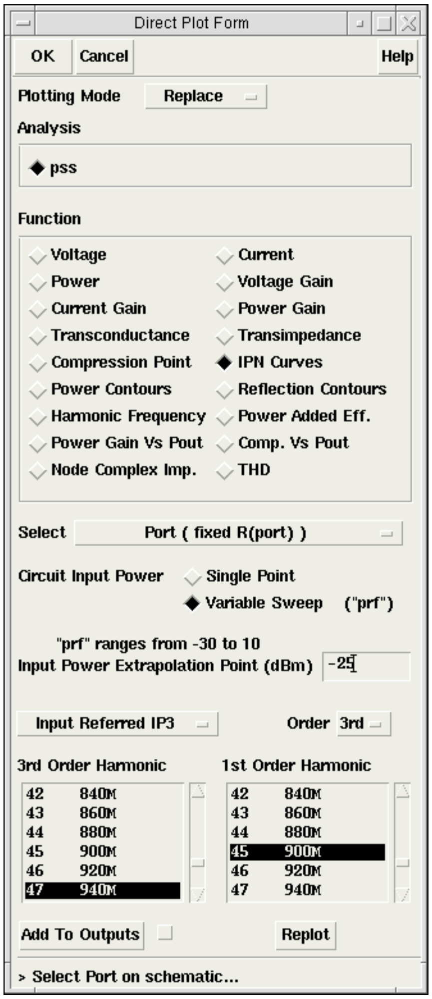
2. Highlight *Replace* for *Plotting Mode*.
3. Highlight *pss* for *Analysis*.

4. Highlight *IPN Curves* for *Function*.
5. Highlight *Variable Sweep (“prf”)* for *Circuit Input Power*.
6. Choose *Input Referred IP3*.
7. Type `-25` for the *Input Power Extrapolation Point (dBm)*.

   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.
8. In the *order* cyclic field, choose *order 3rd*.
9. In the *3rd Order Harmonic* list box, click *940M*.
10. In the *1st Order Harmonic* list box, click *900M*.

    Because the two input frequencies are 900MHz and 920MHz, the two-tone, third-order harmonics are 880MHz and 940MHz. These correspond to the 45th and 47th harmonics.

The filled-in form looks like this.

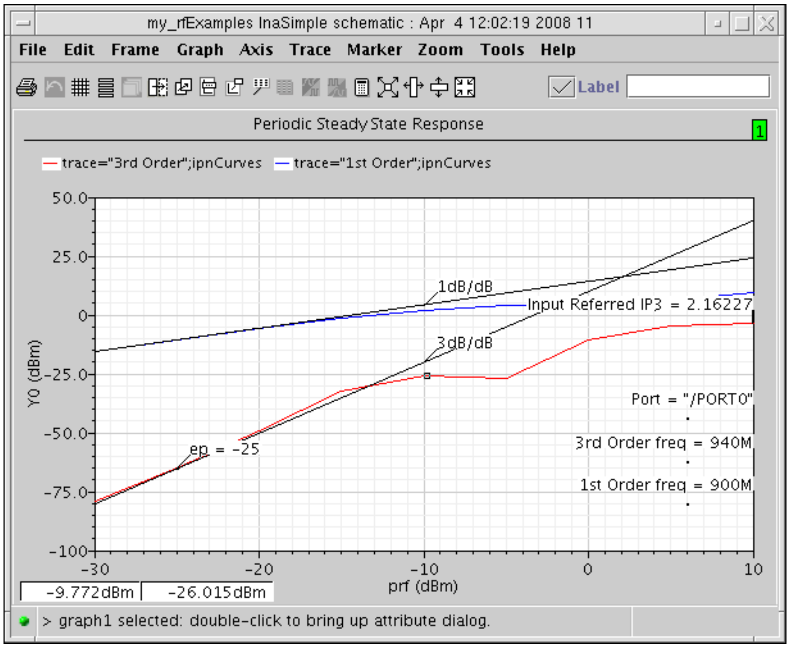



**11.** In the Schematic window, click the PORT 0 output port.

The IPN point is plotted in the waveform window.

**12.** Click *Replot.*

The Spectre RF simulator plots the first-order and third-order curves and identifies the intersection of their slopes.




The third-order interpolation point for this example is 2.16227 dB.

# Calculating Conversion Gain and Power Supply Rejection with PSS and PXF

## Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

## Editing the Schematic

1. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

2. In the Schematic window, click the RFinput voltage source.

   The Edit Object Properties form changes to let you specify data for the input voltage source.

3. In the Edit Object Properties form, if necessary, type `dc` for the *Source Type*.

4. Click *OK*.

5. in the schematic editing window, choose *Design – Check and Save*.

## Setting up the PSS and PXF Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

### Setting up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss*.

2. The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

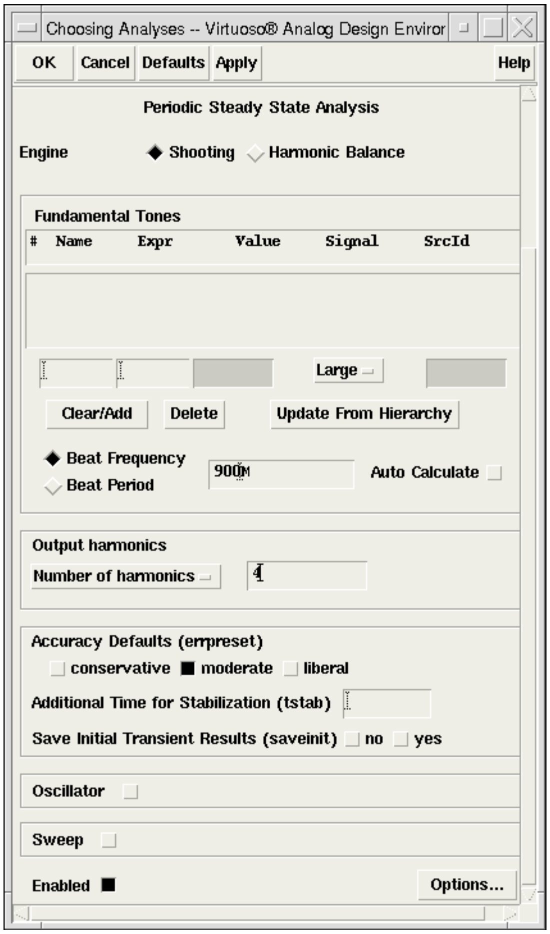
   Because the source type is dc, you must manually enter the fundamental frequency.

3. Be sure the *Auto Calculate* button is not highlighted.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `4`, in the field.

5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

The Choosing Analyses form looks like this.




## Setting up the PXF Analysis

1. At the top of the Choosing Analyses form, click *pxf*.

   The form changes to let you specify data for the PXF analysis.

2. In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop*, and type `10M` and `1.2G` for the *Start* and *Stop* values, respectively in the fields.

3. In the *Sweep Type* cyclic field, choose *Linear* and highlight the *Number of Steps* button.

4. Type `101` for *Number of Steps* in the field.

5. In the *Sidebands cyclic field*, choose *Maximum sideband* for the *Sidebands* choice. Type `0` in the field.

6. Choose *voltage* for the *Output* choice.

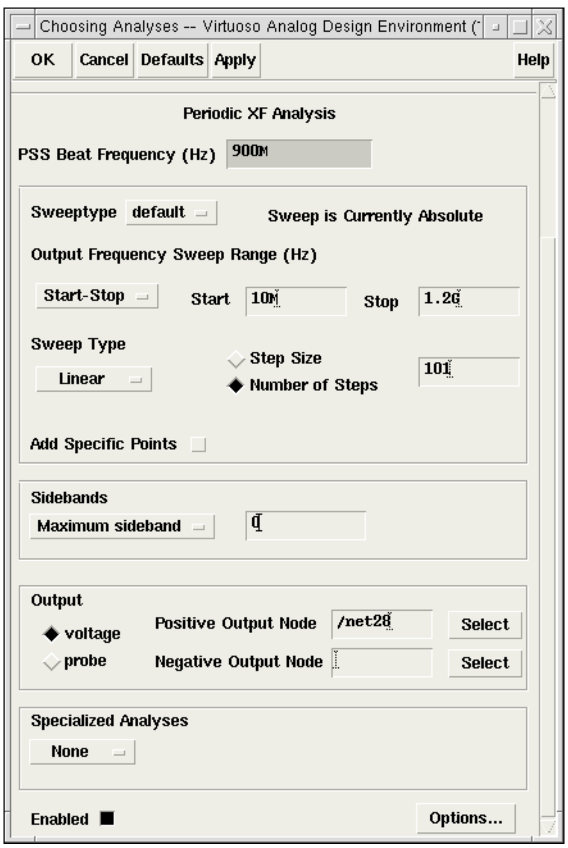
7. Click the *Select* button for the *Positive Output Node*, and then click the output signal in the Schematic window.

8. Leave the *Negative Output Node* field empty.

   This field defaults to */gnd!* You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

9. Highlight the *Enabled* button for the small-signal analysis, if necessary.

   The pxf section of the form looks like this.




10. In the Choosing Analyses form, click *OK*.

## Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

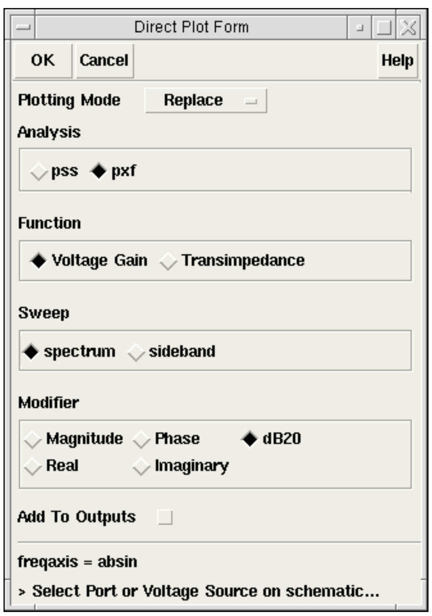
## Plotting Conversion Gain and Power Supply Rejection

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *Replace* for *Plotting mode*.
3. Highlight *pxf* for *Analysis*.
4. Highlight *Voltage Gain* for *Function*.
5. Highlight *dB20* for *Modifier*.

   The filled-in form looks like this.



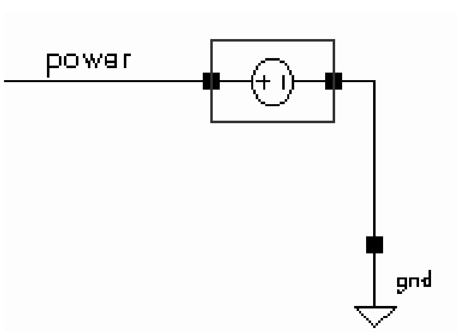

6. In the Schematic window, click the RF source component.

The transfer function for the LNA is plotted in the waveform window.

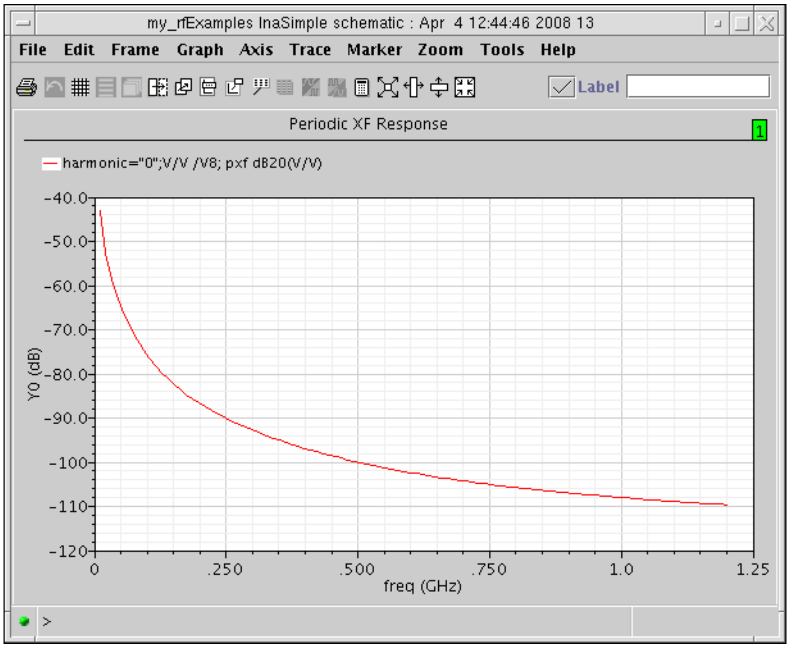

The conversion gain at 900M is approximately 14.4 dBV.

**7.** In the Direct Plot form, choose *Replace* for *Plotting Mode*.

**8.** To see the power supply rejection, click the vdb DC voltage source in the schematic.



Be sure to click the components, not the wires.




The plot shows that the power supply rejection at 900M is approximately -106 dBV.

# 8

# Modeling Transmitters

This chapter tells you how to use several Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) transmitter design features. It emphasizes

- Setting Up an Envelope analysis (Envlp)

  The Envelope analysis example explains

  - How to interpret Envlp analysis results
  - How to visually detect distortion quickly

- Measuring ACPR and PSD. PSD (Transmitted Power Spectral Density) is characterized by the ACPR number (Adjacent Channel Power Ratio).

  The ACPR and PSD example explains

  - How to use the ACPR wizard.
  - How to estimate PSD.

- Measuring Load Pull Contours and Reflection Coefficients

  The Load-Pull example explains

  - How to select an optimal power amplifier load
  - How to determine whether the input matching network needs to be re-designed for the optimal load

- Using S-parameter Input Files

  The S-parameter example explains

  - How to generate tabulated S-parameters
  - How to include tabulated S-parameters as input in a Spectre RF analysis

- Measuring AM and PM Conversion with Modulated PAC, AC and PXF Analyses

The AM/PM conversion example explains how to determine the amplitude and phase modulation effects in RF circuits using the Modulated PAC and PXF analyses.

- ❑ How to create the EX_AMP example.
- ❑ How to set up the modulated PAC and PXF analyses

- Measuring Jitter with PSS and Pnoise Analyses

  This example explains how to measure different jitter measurements using the Pnoise Jitter measurement.

  - ❑ How to set up or create the EX_AMP circuit.
  - ❑ How to set up the Pnoise jitter analyses

# Envelope Analysis

This example tells you how to set up and run the Envelope analysis then explains why the time results look somewhat unusual.

Before you start, perform the setup procedures described in Chapter 3.

## Opening the EF_example Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

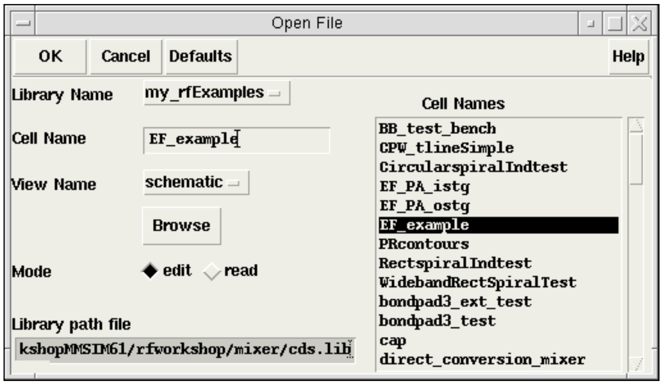
      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

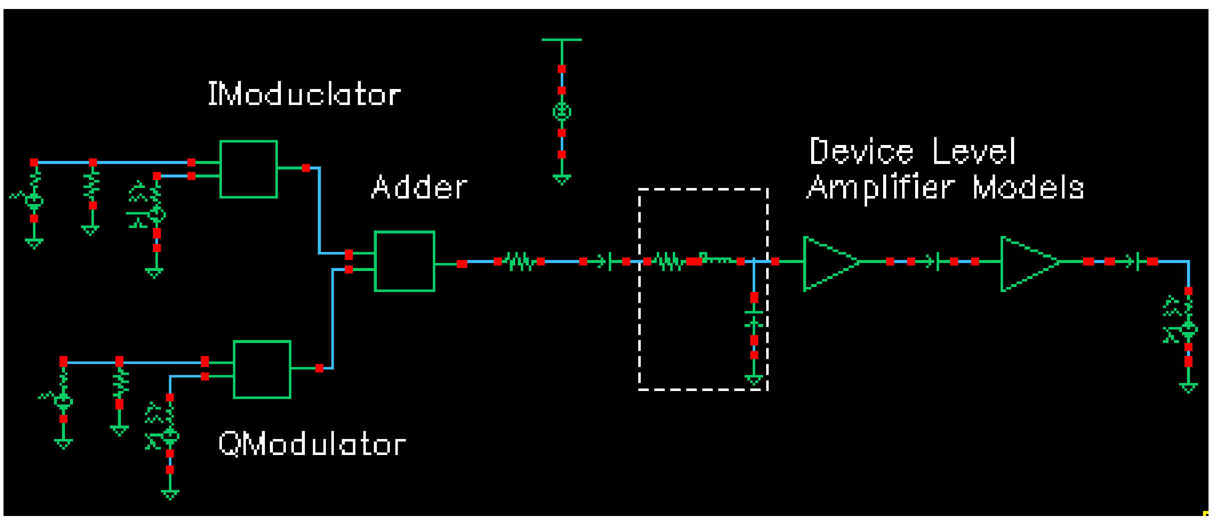
   d. Highlight *edit* for *Mode*.

After these steps the Open File form looks like this.




**e.** Click *OK*.

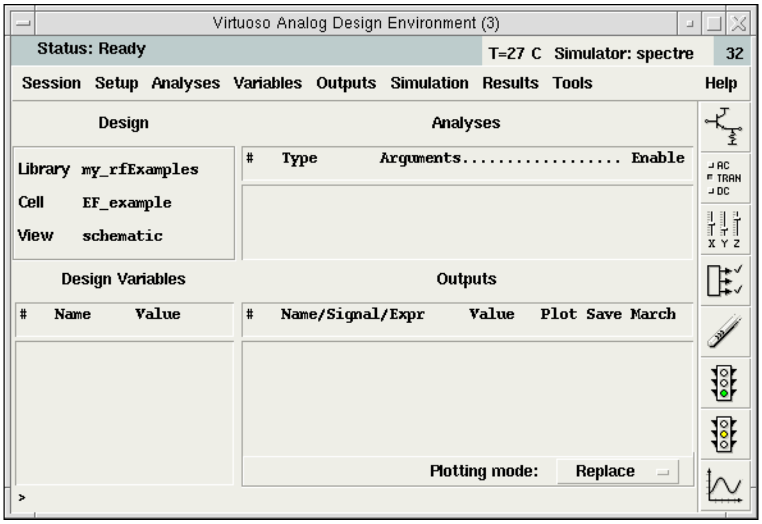
The Schematic window appears with the *EF_example* schematic. This is a simple direct-conversion transmitter with ideal I/Q modulators.




## Opening the Simulation Window

**1.** In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




**Note:** You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *EF_example* in the Choosing Design form.

## Setting Up the Model Libraries

1. In the ADE window, choose *Setup - Model Libraries.*

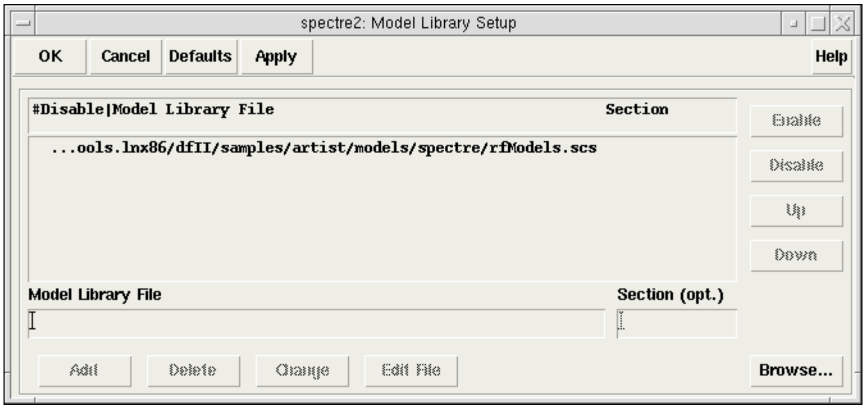
   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, where `CDSHOME` is the installation directory for the Cadence software.

   `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs.`

3. Click *Add*.

The Model Library Setup form looks like the following.



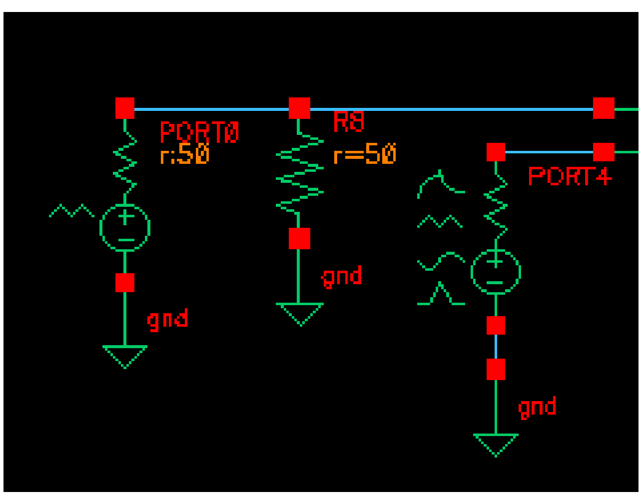

**4.** Click *OK*.

## Editing PORT0 and PORT1 in the EF_example Schematic

The *EF_example* circuit uses the programmable voltage source, *port*. The RF voltage source is based on the *port* sample component. You can easily change the behavior of this programmable component.

In the example, you edit `PORT0` and `PORT1` on the left side of the schematic.

**1.** In the Schematic window, select `PORT0`. (The port in the top, left corner of the *EF_example* schematic.)



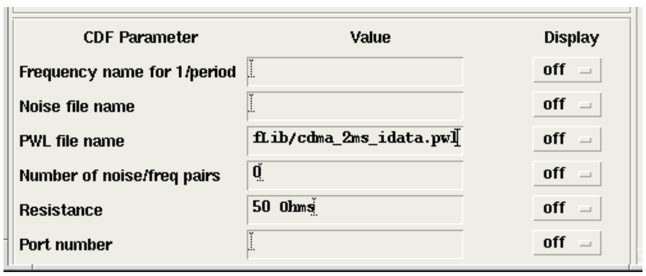

**2.** In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears and changes to display information for `PORT0`. You use this form to change the list of CDF (component description format) properties for the `PORT0` and modify the schematic for this simulation.

**3.** In the Edit Object Properties form, edit the *PWL filename* of `PORT0`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click *Apply*.

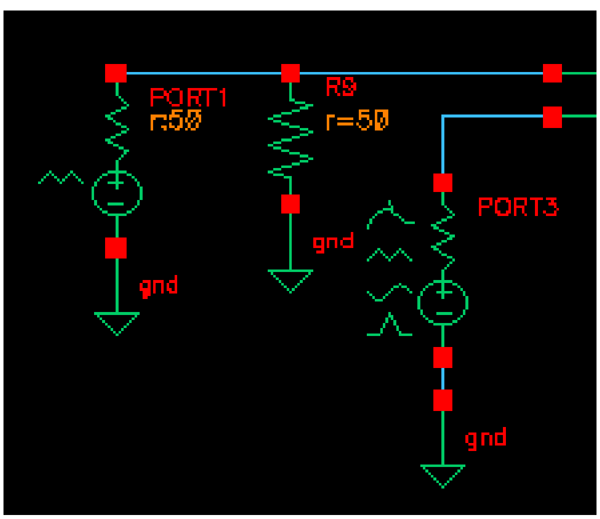
```
<CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_idata.pwl
```

The section of the Edit Object Properties form that includes the *PWL file name* field looks like the following.



| CDF Parameter | Value | Display |
|---|---|---|
| Frequency name for 1/period | | off |
| Noise file name | | off |
| PWL file name | fLib/cdma_2ms_idata.pwl | off |
| Number of noise/freq pairs | 0 | off |
| Resistance | 50 Ohms | off |
| Port number | | off |

**4.** In the Schematic window, select `PORT1`. (The port in the bottom, left corner of the *EF_example* schematic.)




The Edit Object Properties form changes to display information for `PORT1`.

**5.** In the Edit Object Properties form, edit the *PWL filename* of `PORT1`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click *Apply*.

```
<CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_qdata.pwl
```

6. Click *OK*.
7. Choose *Design - Check and Save* in the Schematic window.

## Setting Up an Envelope Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *envlp*.

   The Choosing Analyses form changes to let you specify values for the Envelope analysis.

3. In the Choosing Analyses form, do the following.

   a. Click *Clock Name*.

   b. Click *Select Clock Name.*

      The Select Clock Name form appears.

   c. In the Select Clock Name form, click `fff` then click *OK.*

      In the *Clock Name* field, `fff` appears.

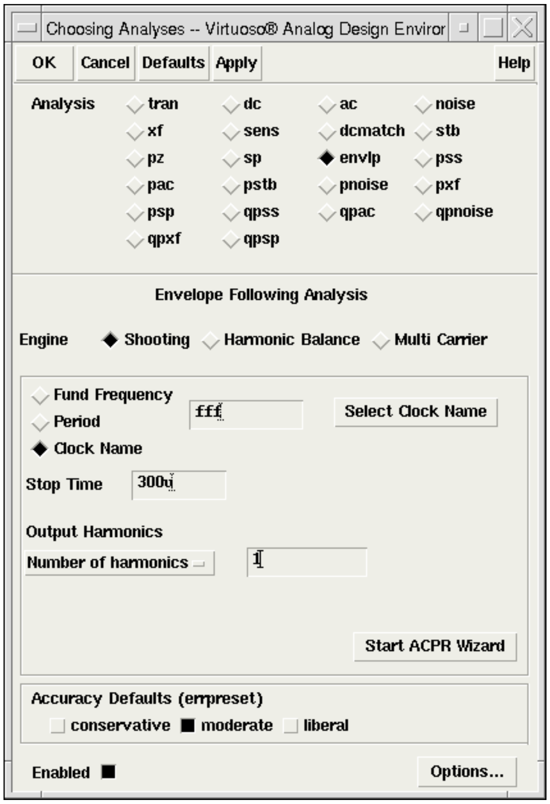
   d. Type `300u` for *Stop Time*.

   e. In the for *Output Harmonics* cyclic field, choose *Number of harmonics* and type 1 in the adjacent field.

   f. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

   g. Verify that *Enabled* is highlighted.

The completed form looks like this.




**h.** Click *OK*.

**4.** In the ADE window, choose *Simulation – Netlist and Run*.

Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

## Looking at the Envelope results

**1.** In the ADE window, choose *Results – Direct Plot – Main Form*.

The Direct Plot form and the waveform window appear.

**2.** In the Direct Plot form, do the following:

**a.** Choose *Replace* for *Plotting Mode*.

**b.** Highlight *envlp* for *Analysis*.

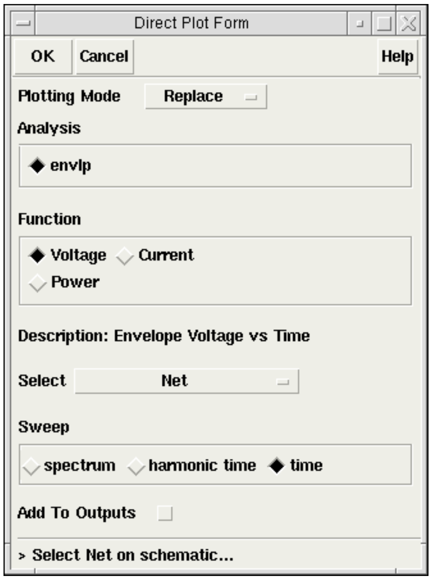
**c.** Highlight *Voltage* for *Function*.

**d.** Highlight *time* for *Sweep*.

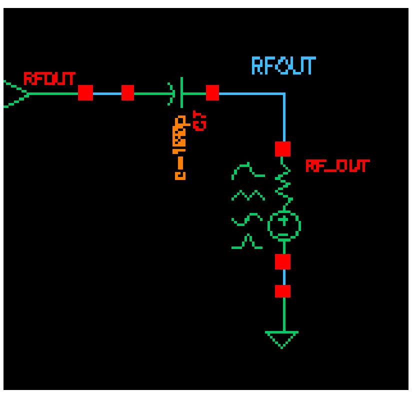
**e.** The *Select* cyclic field displays *Net* and a prompt displays at the bottom of the form.

*> Select Net on Schematic....*

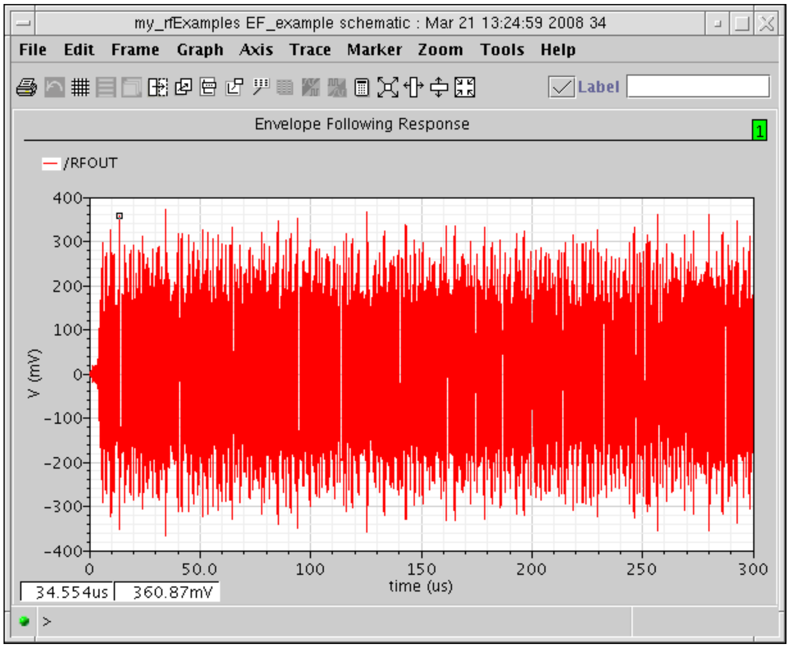
The completed form looks like this.




**3.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (/RFOUT)

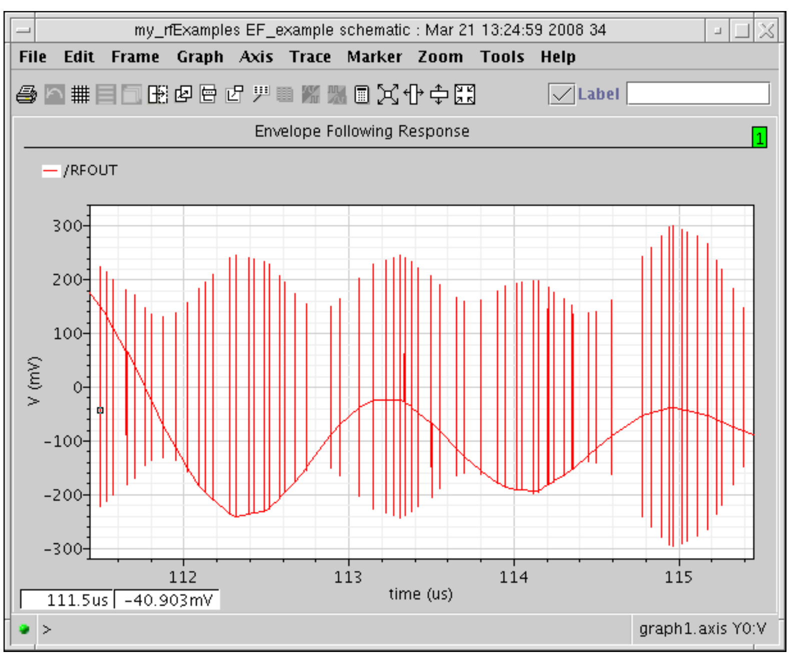


The voltage waveform for *RFOUT* appears in the waveform window.




4. To get a closer look at a small section of the waveform, in the waveform window, right click and drag to select a small section of the waveform. (The area you select covers a narrow vertical rectangle.)

You might have to do this several times to get a plot similar to the one shown.



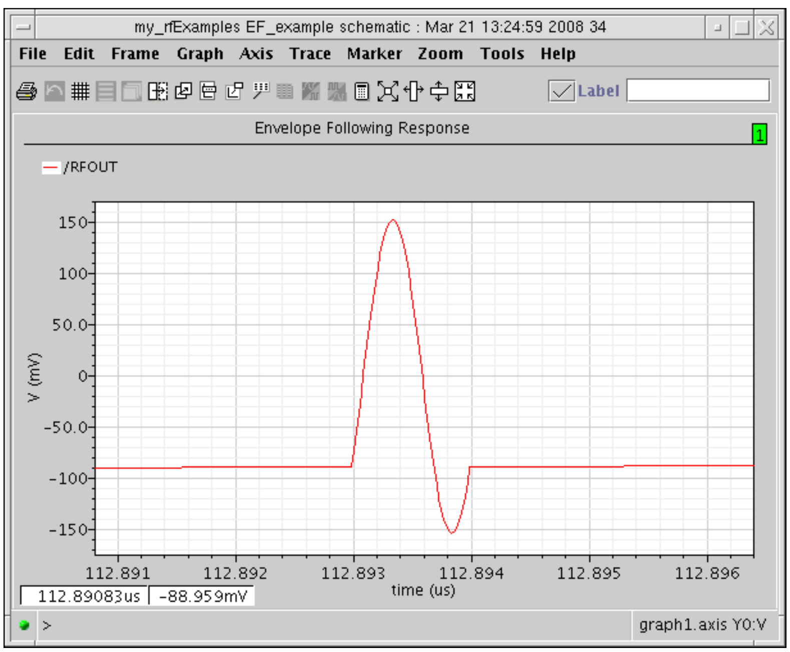
The plot changes to display a number of vertical lines with a wavy line running through them.

- ❑ Each vertical line is a time point where a detailed calculation was performed.
- ❑ The wavy line connects these points

An Envelope analysis runs much faster than a Spectre transient analysis because Envelope analysis skips carrier cycles when it can do so while still satisfying numerical tolerances.

**5.** To get a closer look, right click and drag to select one of the vertical lines. You might have to do this several times.

After several selections, you are able to see the detailed simulation plot for one complete cycle. What you see should be similar to the following waveform depending on the areas you selected.




## Following the Baseband Signal Changes Through an Ideal Circuit

The modulation riding on the RF carrier is the baseband signal, the information to be transmitted. The baseband signal determines the amplitude and phase of the RF carrier. In transmitter design, it is important to determine how the transmitter might alter the baseband signal.

This section illustrates how to extract and plot the baseband signal at several points in the circuit.

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the waveform window appear.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plotting Mode*.

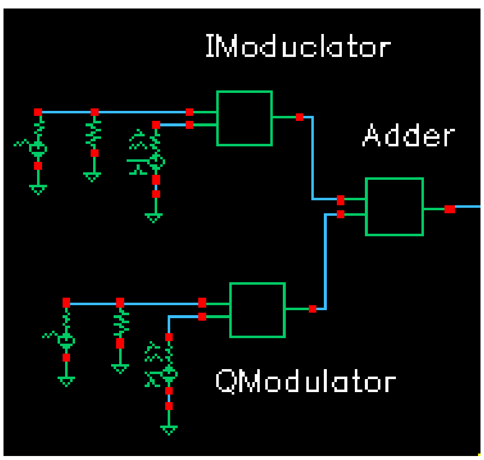
   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form. displays
   `> Select Net on Schematic.`

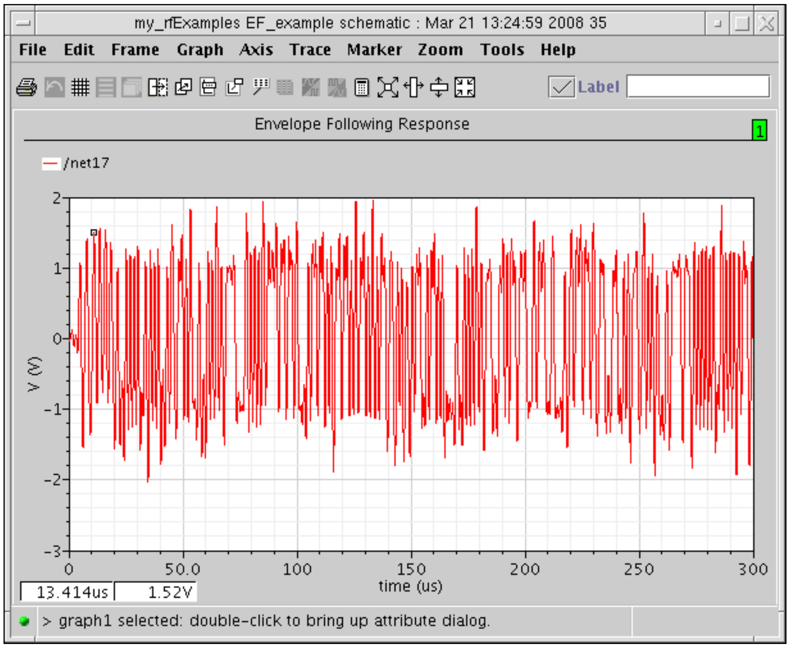
   e. Highlight *time* for *Sweep*.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator source net (the I-modulator input net.




**Note:** You might have to click *Zoom – Fit* in the waveform window before you are able to see the waveform.

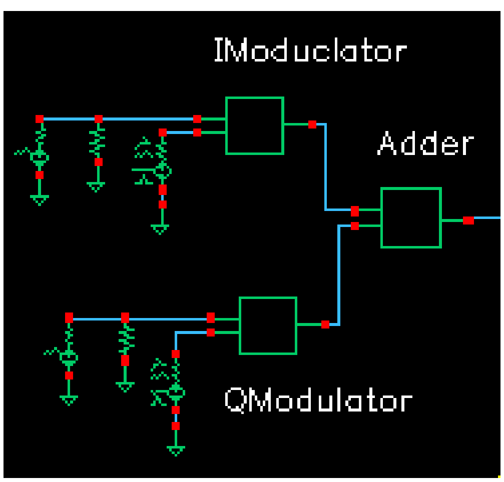
The plot for the modulation source waveform appears.




**4.** In the Direct Plot form, highlight *Append* for *Plotting Mode*.

This adds new waveforms to any waveforms currently displayed in the waveform window.

**5.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator output net as shown in the following schematic.



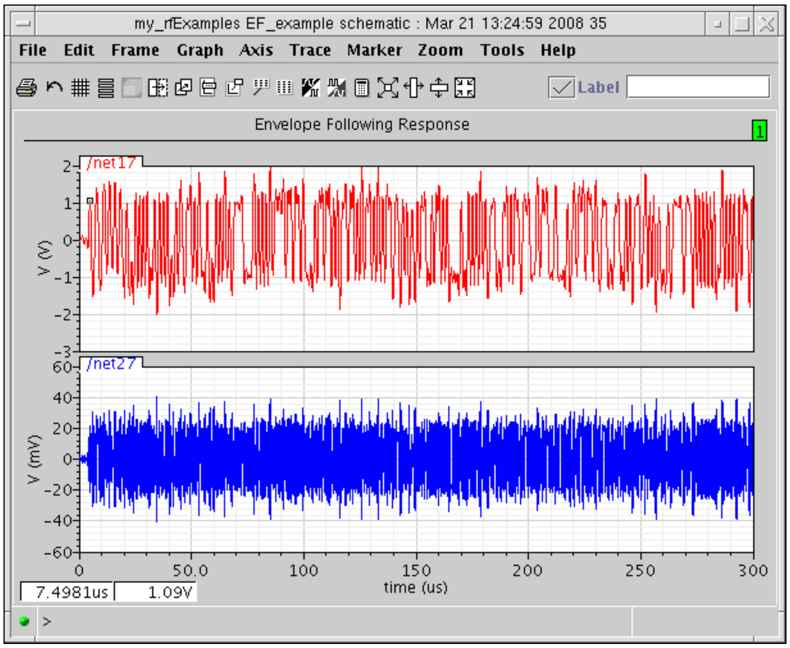

This is the in-phase carrier modulated by the I-component of the baseband signal. In this example,

- The in-phase carrier is `cos(`$\omega_c$`t)`, where $\omega_c$ is the carrier frequency in radians per second.
- The quadrature carrier is `sin(`$\omega_c$`t)`.

The I-modulator output waveform is added to the waveform window.

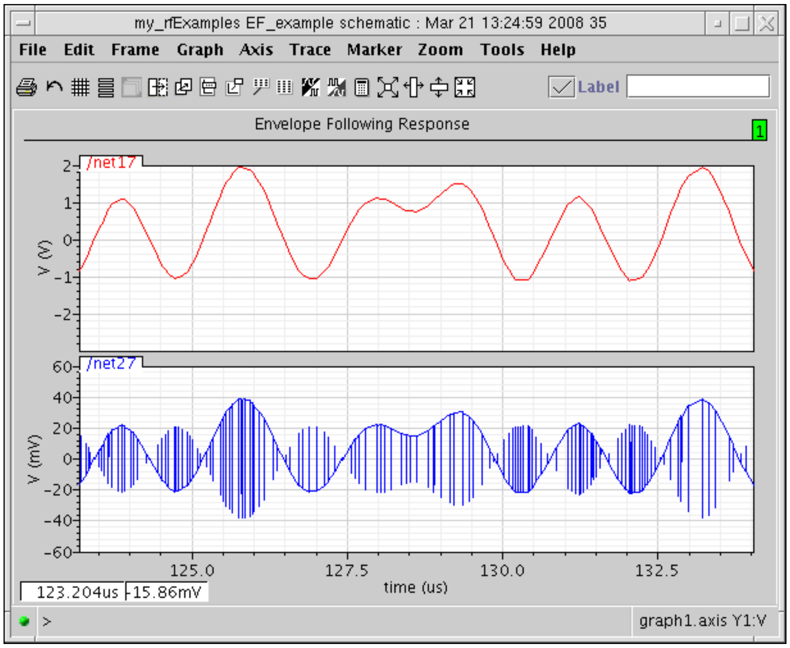
**6.** In the waveform window, choose *Axis – Strips*.

This changes the waveform window to display multiple waveforms in strips, one above the other.




**7.** In the waveform window, right click and drag over a narrow section of both waveforms.

You can see the modulation wave within the output wave. It is coincidental that the top waveform and smooth curve in the bottom waveform look alike. If the individual cycles in the bottom waveform were sampled at a different phase, the smooth curve in the bottom waveform might look different.




**Note:** The appearance of this plot, as well as the appearance of subsequent plots, depends on the section of the waveform window in which you perform the click and drag.

**8.** In the Direct Plot form, do the following:

**a.** Highlight *harmonic time* for *Sweep*.

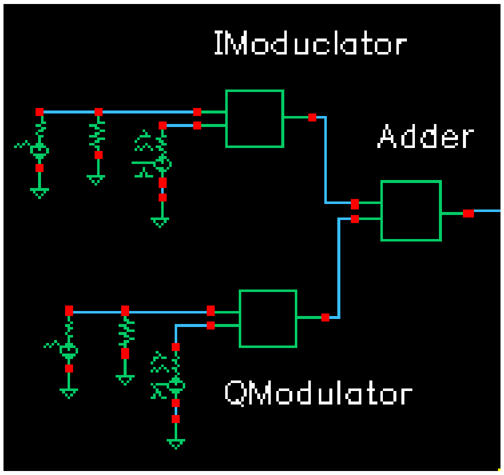
**b.** Highlight *Real* for *Modifier*.

**c.** Highlight *1* for *Harmonic Number*.

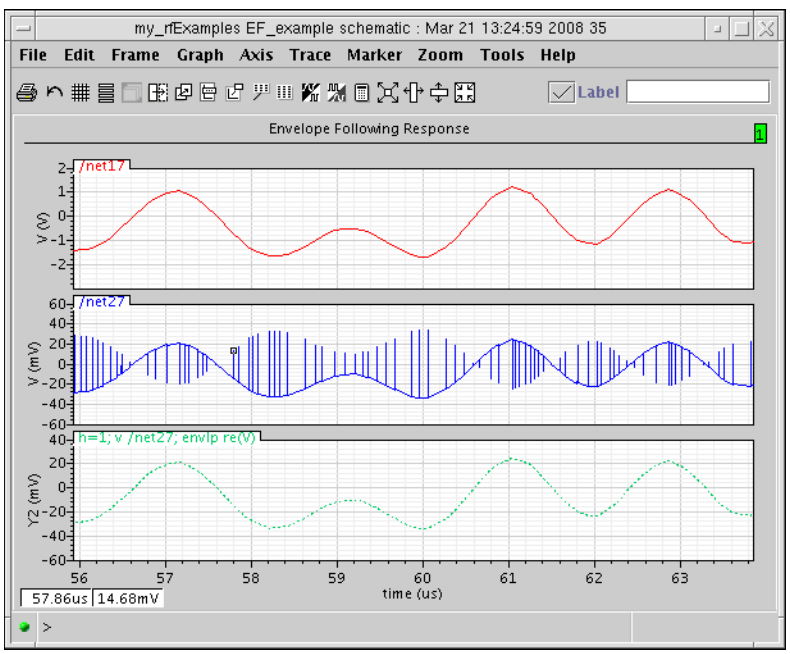
**d.** The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

*> Select Net on Schematic....*

Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator output net.




**9.** The bottom plot in the waveform window is the baseband waveform recovered from the modulated RF carrier. Aside from a linear scale factor, It looks exactly like the top waveform because the modulator is ideal.

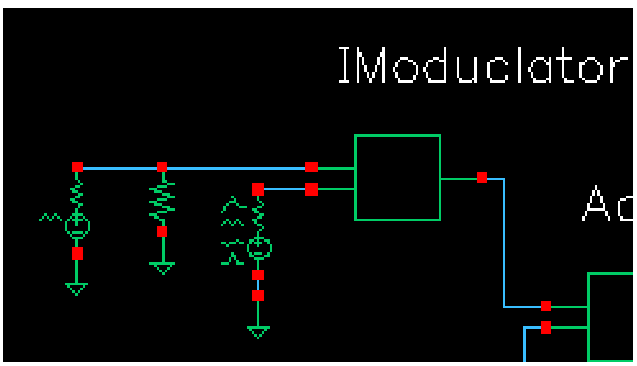



## Following the Baseband Signal Changes Through a Non-Ideal Circuit

To see the change in the signal as it passes through the entire length of the circuit, which is not ideal, follow the instructions in this section.

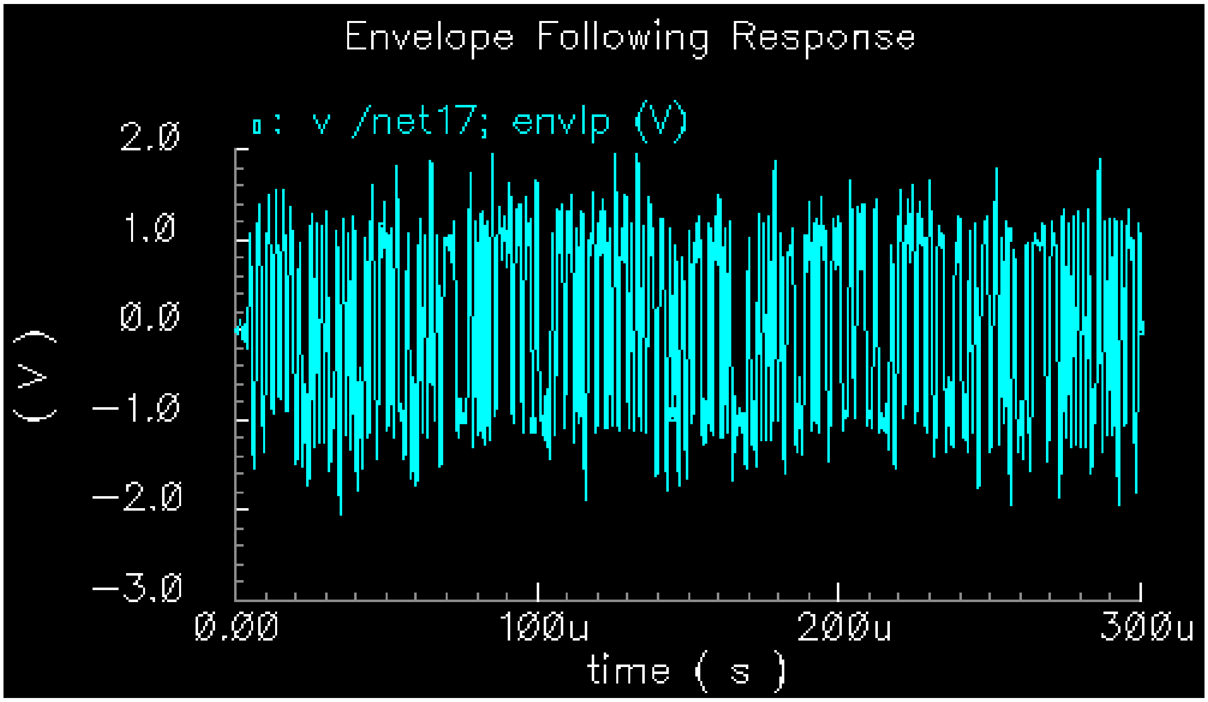
1. In the waveform window, choose *File - Close* to close the waveform window.
2. In the Direct Plot form, do the following:
   - **a.** Choose *Replace* for *Plotting Mode*.
   - **b.** Highlight *envlp* for *Analysis*.
   - **c.** Highlight *Voltage* for *Function*.
   - **d.** Highlight *time* for *Sweep*.
   - **e.** The *Select* cyclic field displays *Net* and a prompt at the bottom of the form

     `> Select Net on Schematic....`
3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator source net.




The plot for the modulation source waveform appears.

You might have to choose *Zoom – Fit* in the waveform window before you are able to see the waveform.



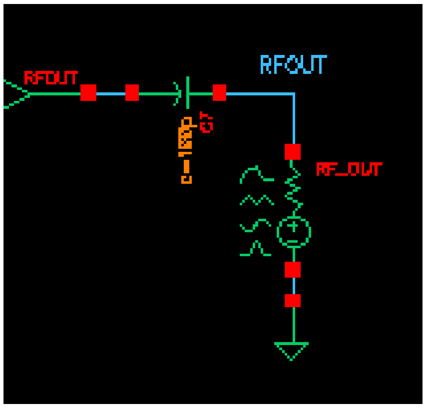

4. In the waveform window, choose *Axes – To Strip*.

   This changes the waveform window to display multiple waveforms in strips, one above the other.

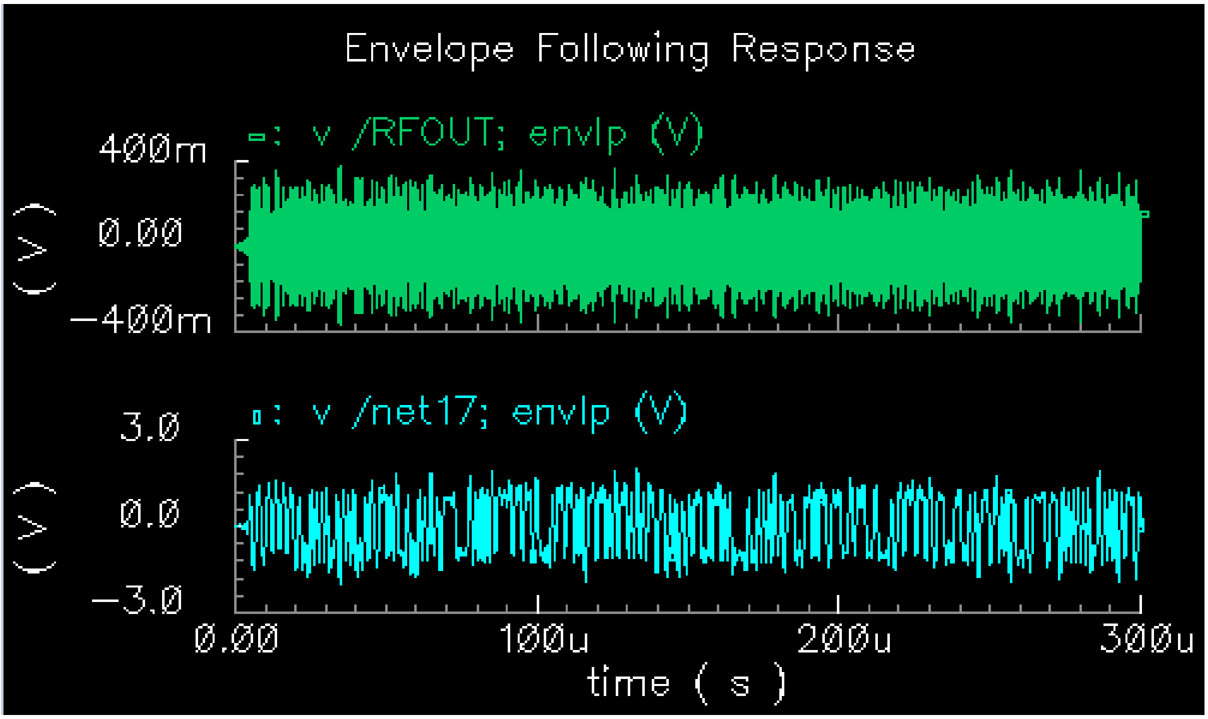
5. In the Direct Plot form, highlight *Append* for *Plotting Mode*.

   This adds new waveforms to any waveforms currently displayed in the waveform window.

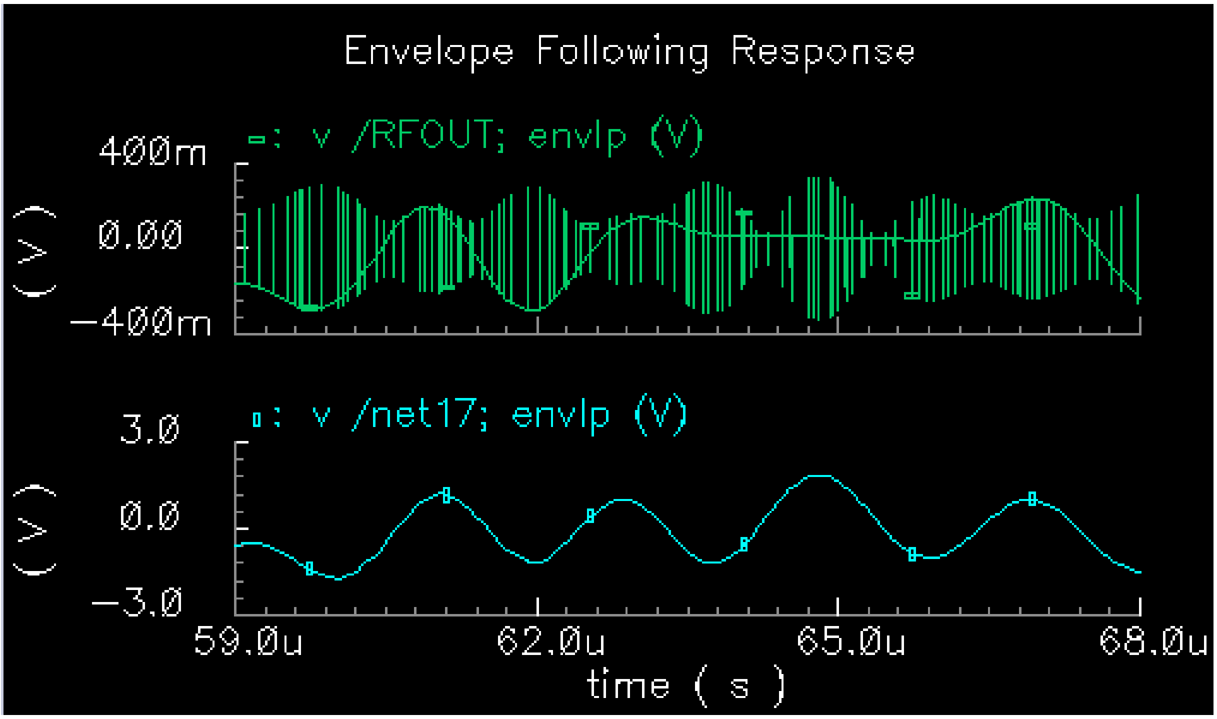
**6.** Following the prompt at the bottom of the Direct Plot form, in the Schematic window, click the transmitter output net (RFOUT).




The transmitter output waveform is added to the waveform window.



**7.** In the waveform window, right click and drag over a narrow section of both waveforms.




You can see the modulation wave within the output wave. It is coincidental that the bottom waveform and smooth curve in the top waveform look alike. If the individual cycles in the top waveform were sampled at a different phase, the smooth curve in the top waveform might look different.

**Note:** The appearance of this plot and of the subsequent plots depends on the section of the waveform window in which you perform the click and drag.

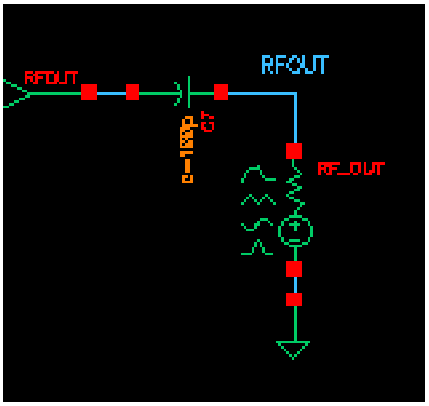
**8.** In the Direct Plot form, do the following:

  **a.** Highlight *harmonic time* for *Sweep*.

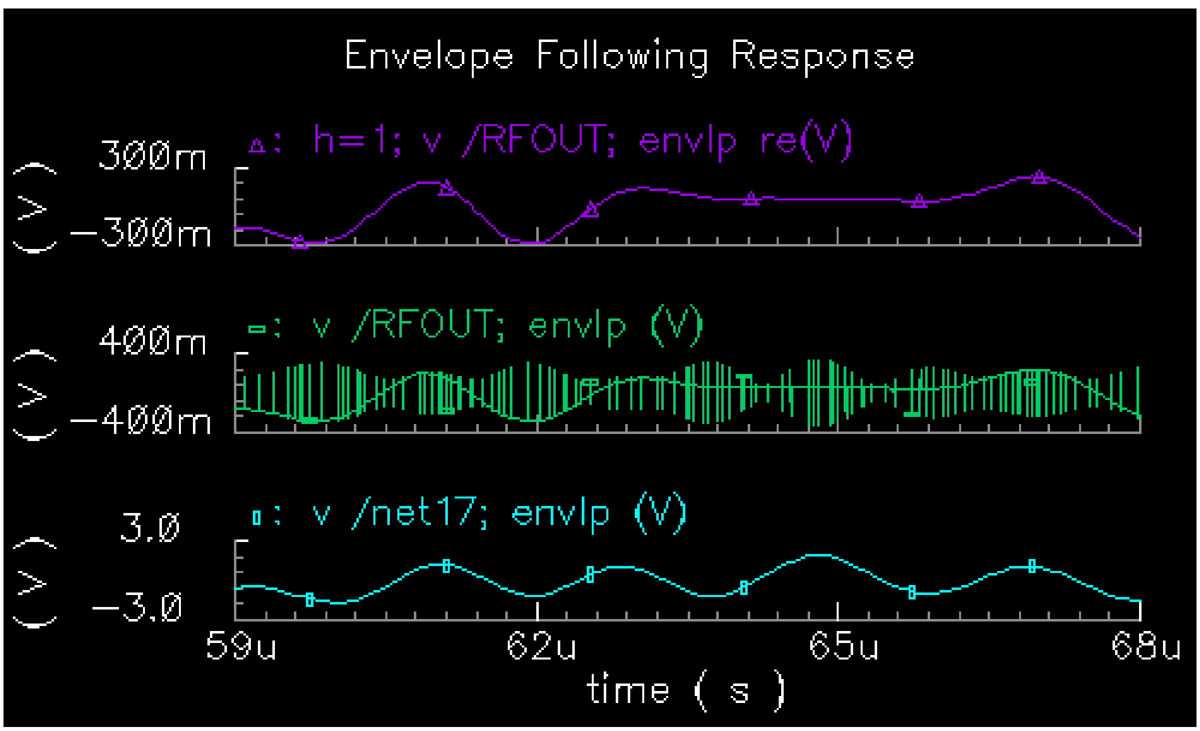
  **b.** Highlight *Real* for *Modifier*.

  **c.** Highlight *1* for *Harmonic Number*.

**9.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (RFOUT).




The waveform at the transmitter output appears at the top of the waveform window. It shows a greater change in the signal.




## Plotting the Complete Baseband Signal

**1.** In the Direct Plot form, do the following:

a. Choose *Replace* for *Plotting Mode*.

b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

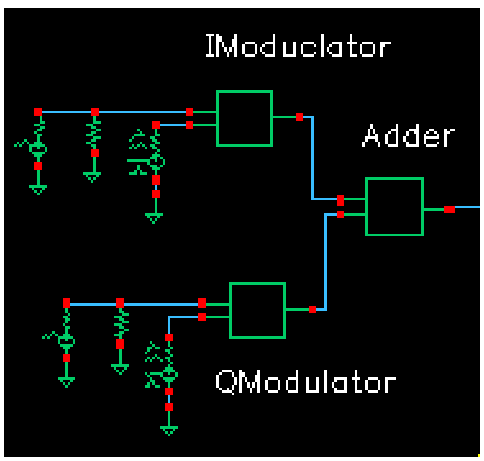
```
> Select Net on Schematic.
```

e. Highlight *harmonic time* for *Sweep*.

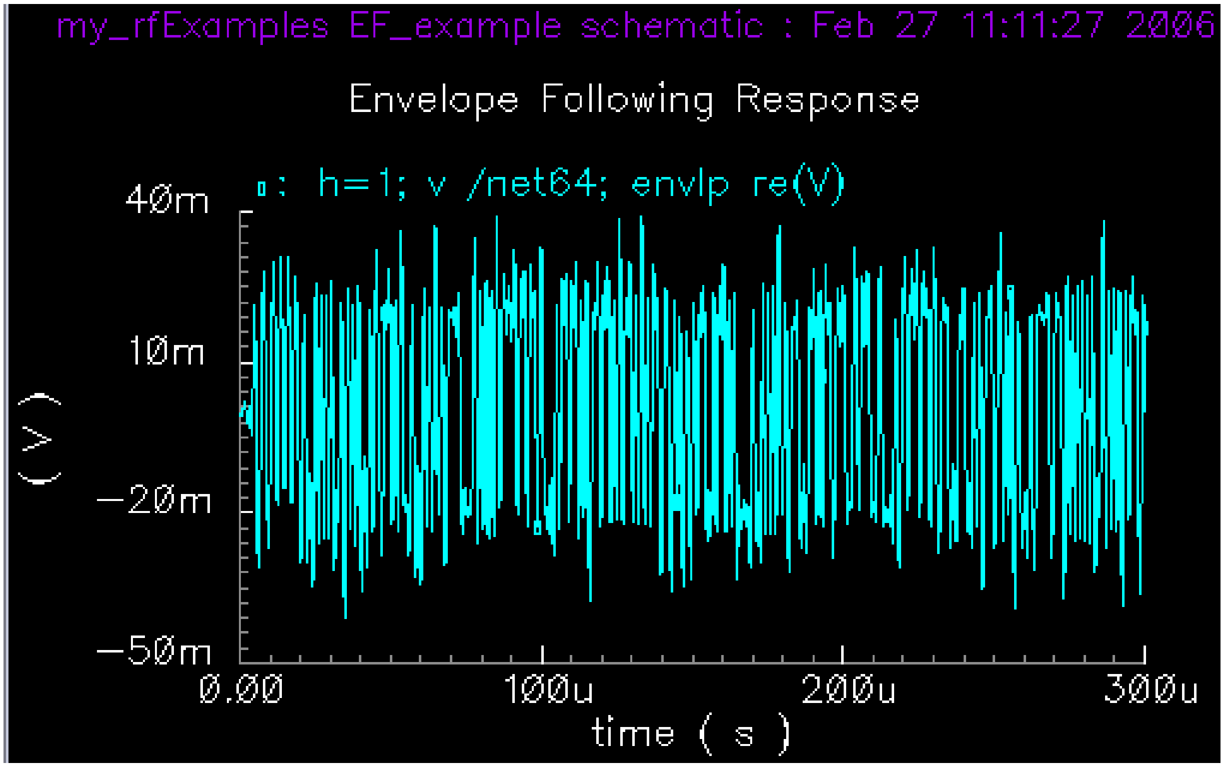
f. Highlight *Real* for *Modifier*.

g. Highlight *1* for *Harmonic Number*.

2. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.



A plot for the real portion of the adder output waveform appears in the waveform window.

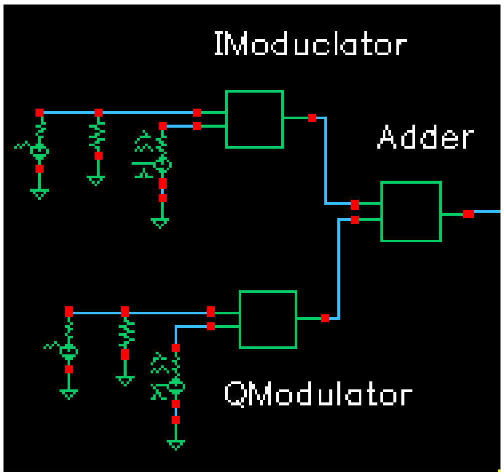



**3.** In the Direct Plot form, make the following changes:

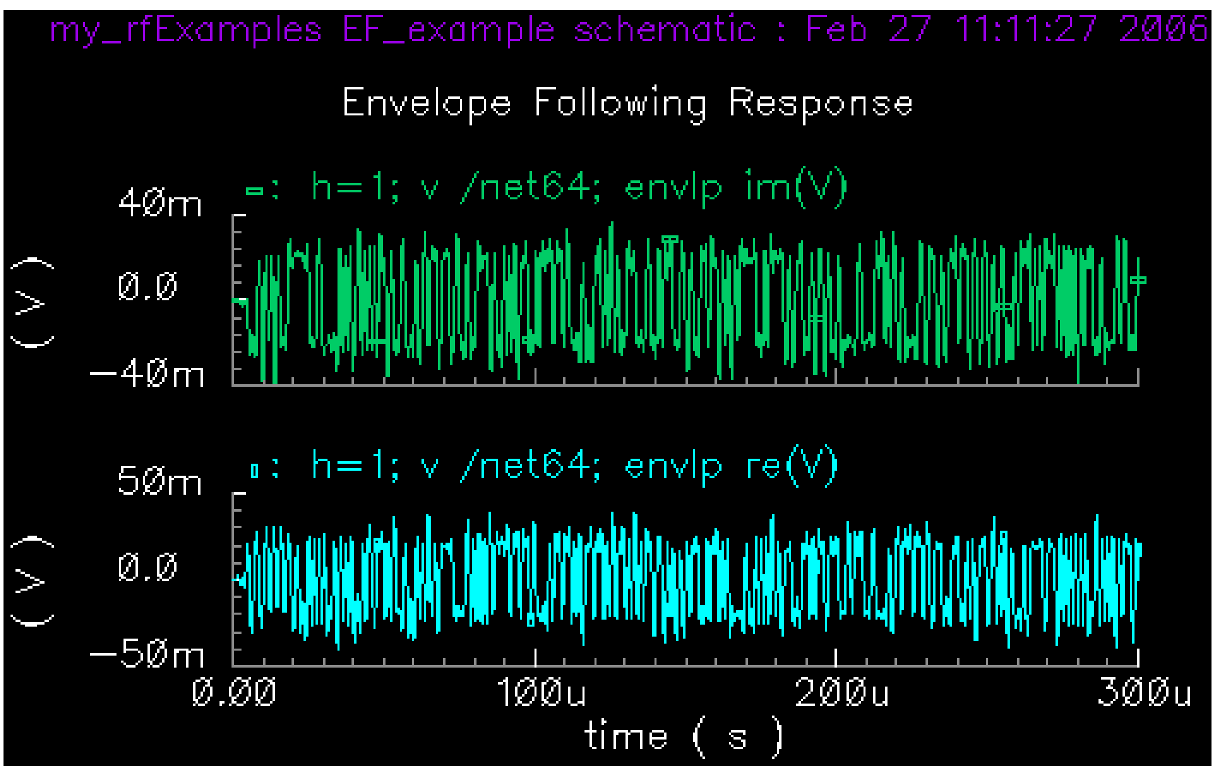
**a.** Choose *Append* for *Plotting Mode*.

**b.** Highlight *Imaginary* for *Modifier*.

4. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.

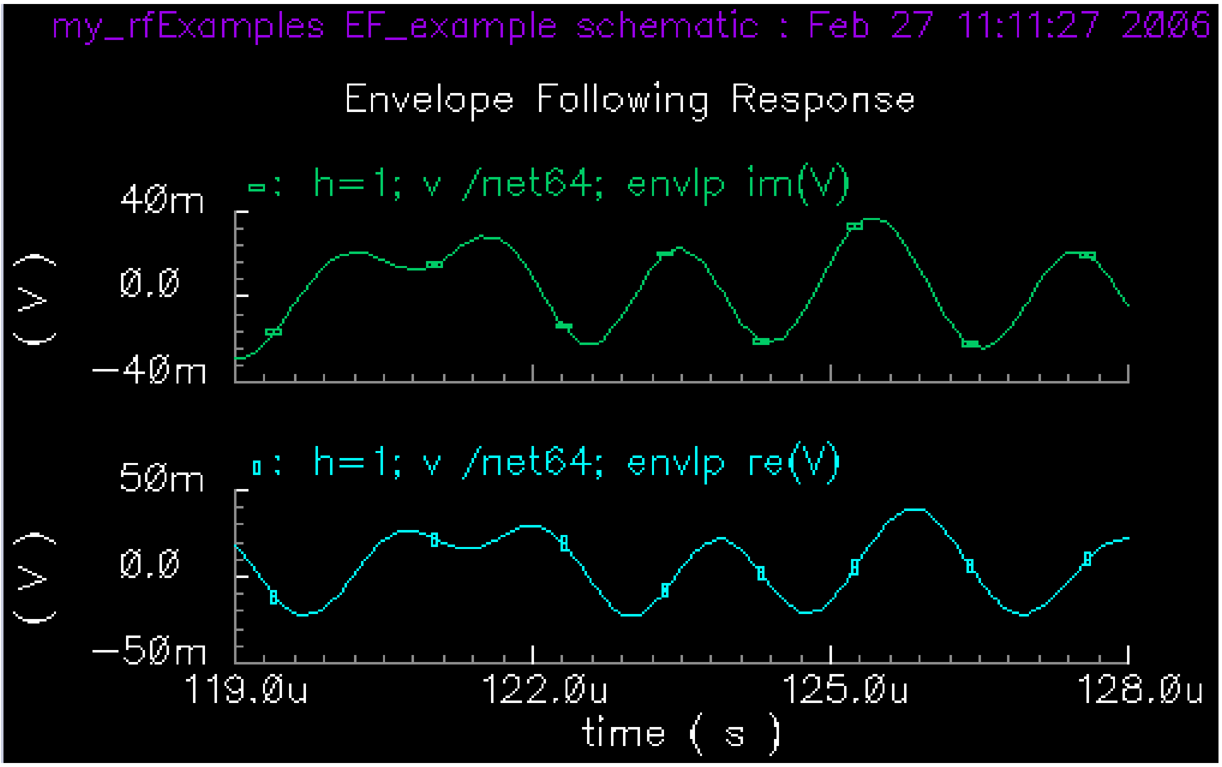



A plot for the imaginary portion of the adder output waveform is added to the waveform window.




5. In the waveform window, right click and drag to select a portion of the waveform window over a narrow portion of the x-axis.

You can now clearly see both the real part and the imaginary part of the baseband signal.




## Plotting the Baseband Trajectory

This section describes how to display the baseband trajectory, the plot of one waveform against the other. The baseband waveforms recovered from the modulated RF carrier, as displayed in the previous section, do not directly reveal much information about how the transmitter affects them.

The baseband trajectory reveals much more information about the kind of distortion the transmitter introduces. The procedure described in this section displays first the input baseband trajectory followed by the output baseband trajectory. A comparison of the two trajectories reveals whether the power amplifiers in this example introduce any phase shift.

### Displaying the Input Baseband Trajectory

1. In the waveform window, choose *Window - Reset*.
2. In the Direct Plot form, do the following:
   a. Choose *Replace* for *Plotting Mode*.

**b.** Highlight *envlp* for *Analysis*.

**c.** Highlight *Voltage* for *Function*.

**d.** The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

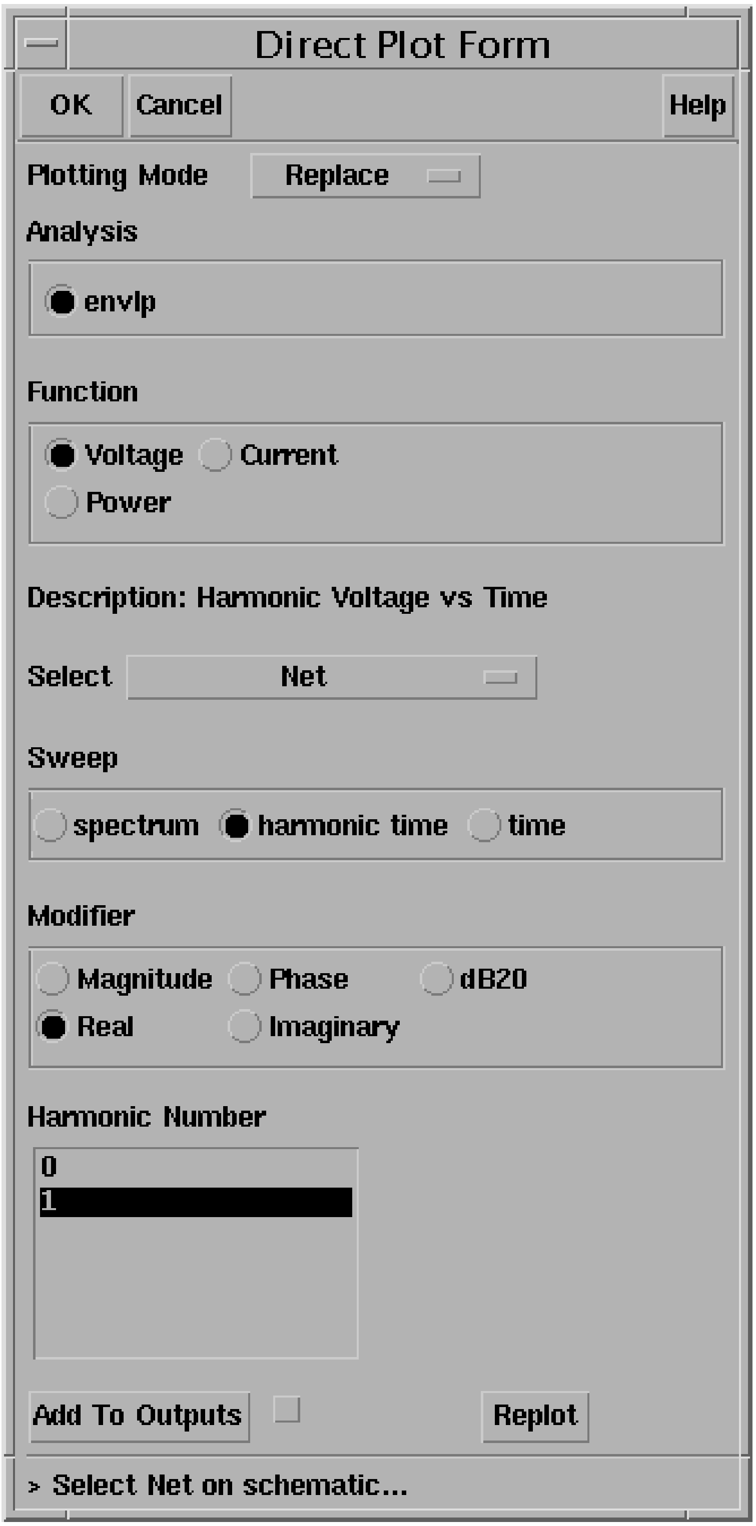
```
> Select Net on Schematic.
```

**e.** Highlight *harmonic time* for *Sweep*.

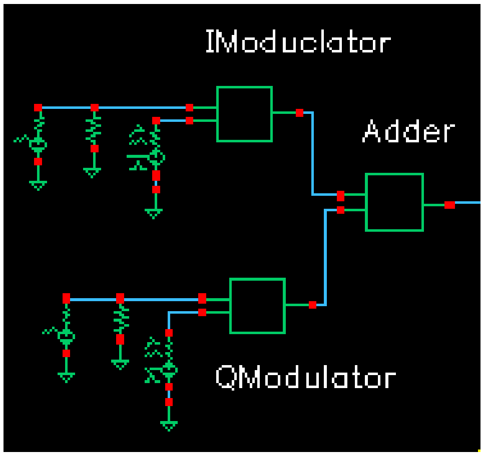
**f.** Highlight *Real* for *Modifier*.

**g.** Highlight *1* for *Harmonic Number*.

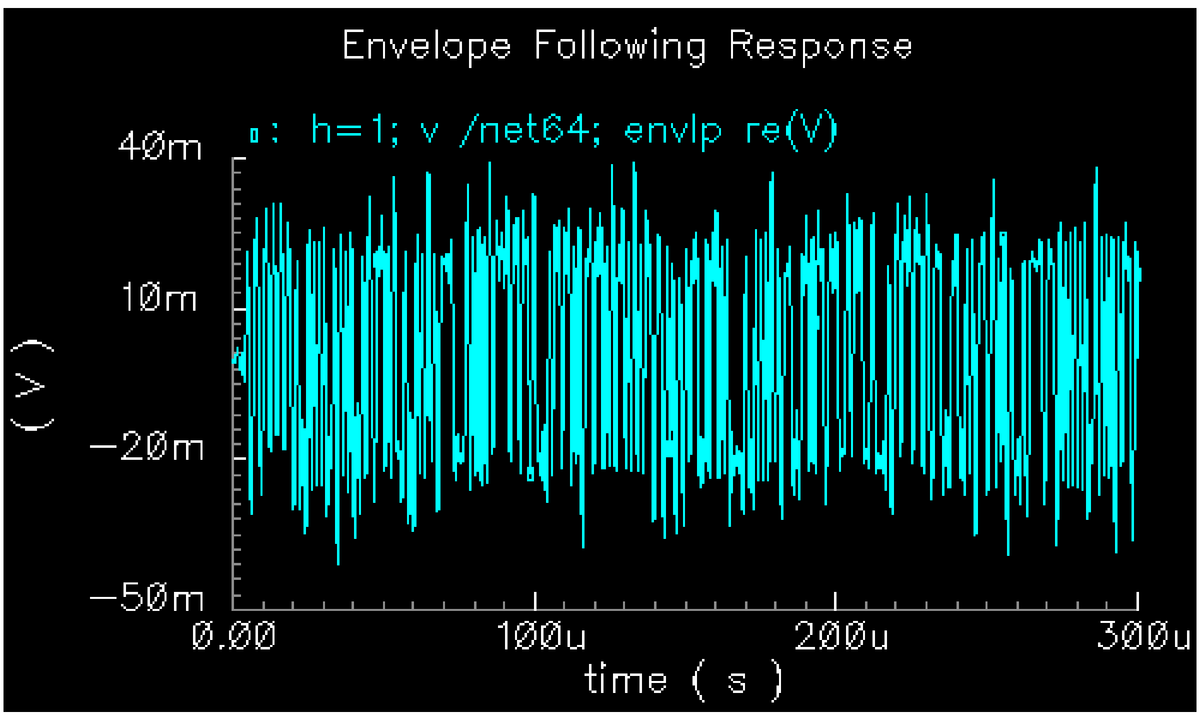
The completed form looks like this.



**3.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.




A plot for the real portion of the adder output waveform appears in the waveform window.

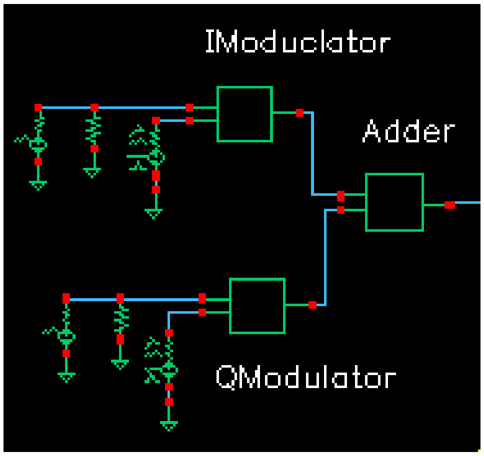



**4.** In the Direct Plot form, change the following:

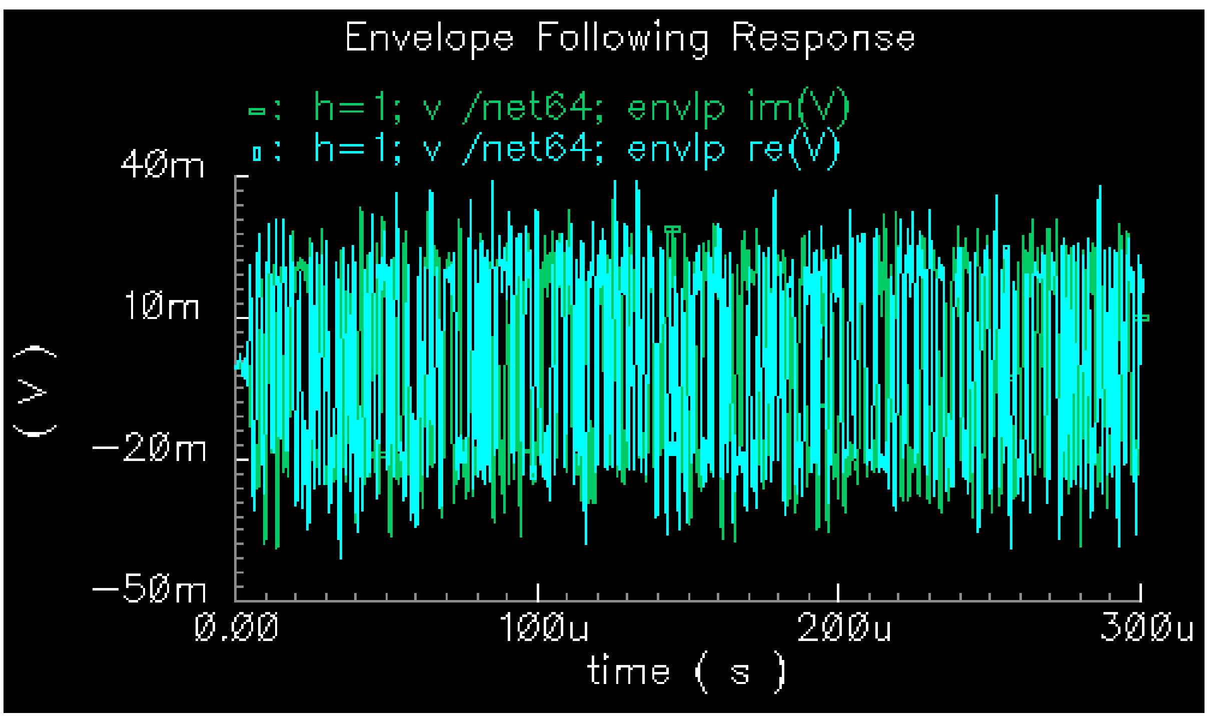
**a.** Choose *Append* for *Plotting Mode*.

**b.** Highlight *Imaginary* for *Modifier*.

**5.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.




A plot for the imaginary portion of the adder output waveform is added to the real portion in the waveform window.

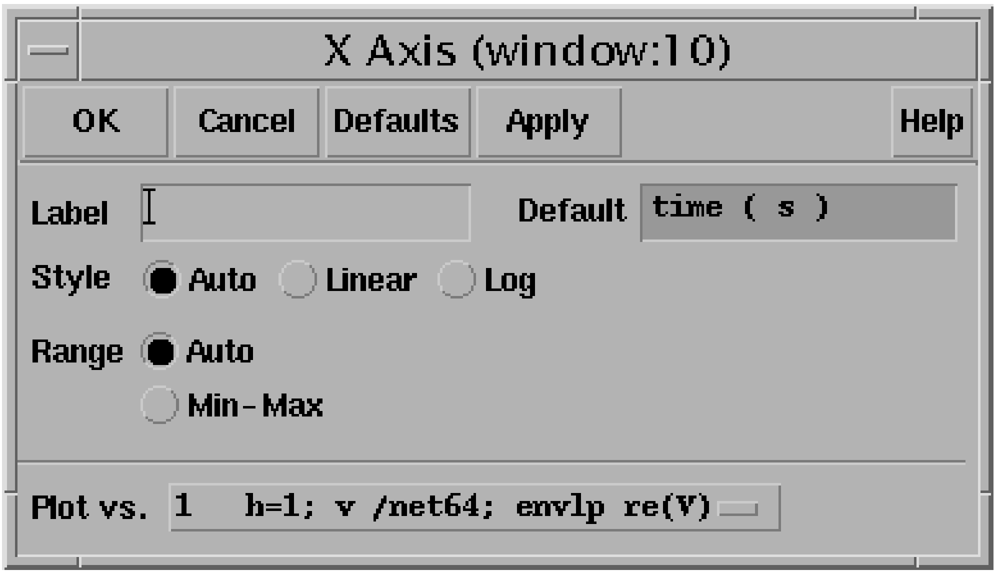



**6.** In the waveform window, choose *Axes – X Axis*.

The X Axis form appears.

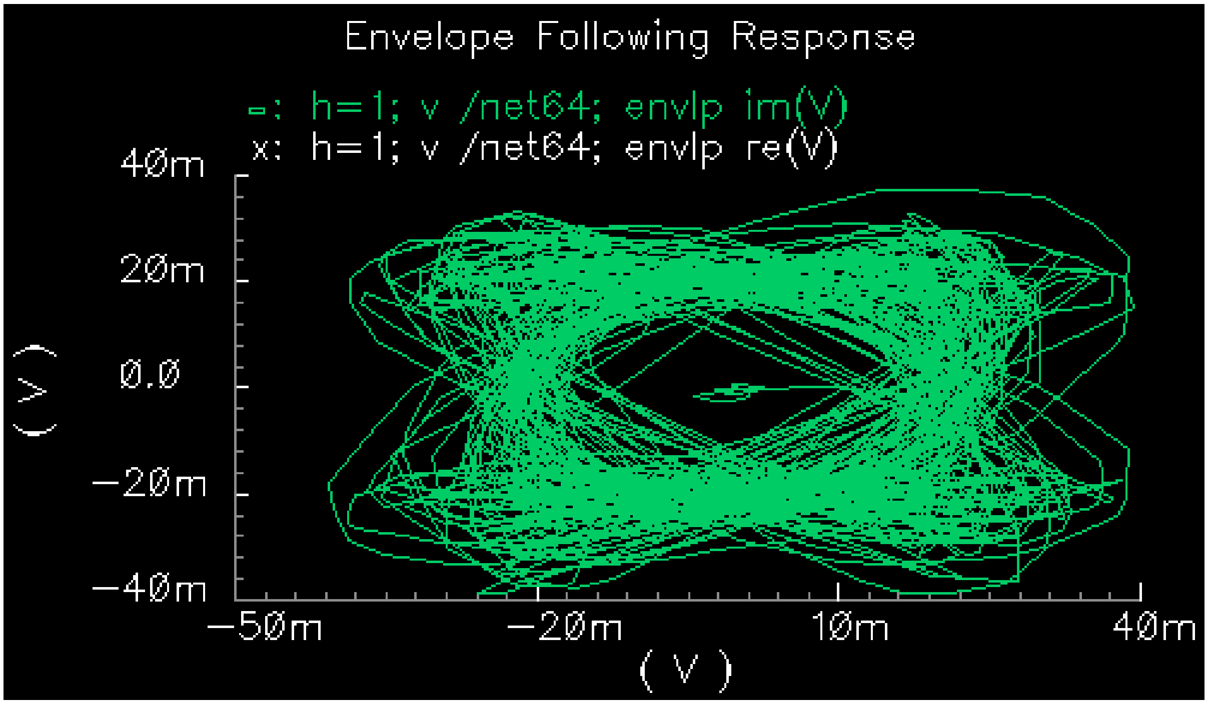
**7.** In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose

```
1    h-1;   v    /net64;  envlp  re(V)
```




**8.** Click *OK*.

The input baseband trajectory, undistorted by the power amplifiers, appears in the waveform window.




### Displaying the Output Baseband Trajectory

1. In the ADE window, choose *Tools – Waveform*.

   A second waveform window appears.

2. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plotting Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

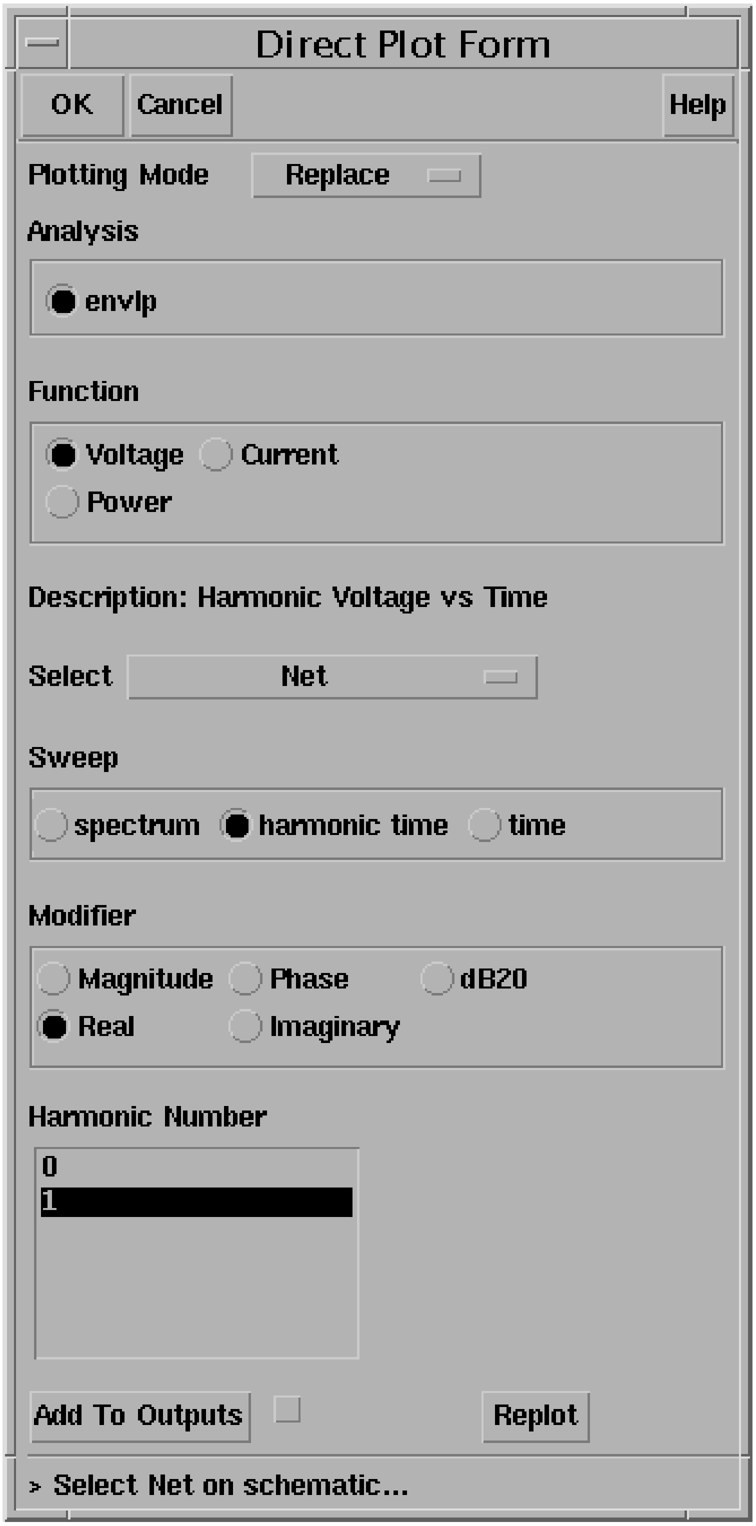
   ```
   > Select Net on Schematic.
   ```

   e. Highlight *harmonic time* for *Sweep*.

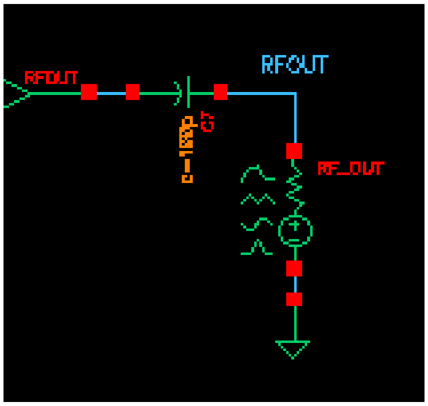
   f. Highlight *Real* for *Modifier*.

   g. Highlight *1* for *Harmonic Number*.

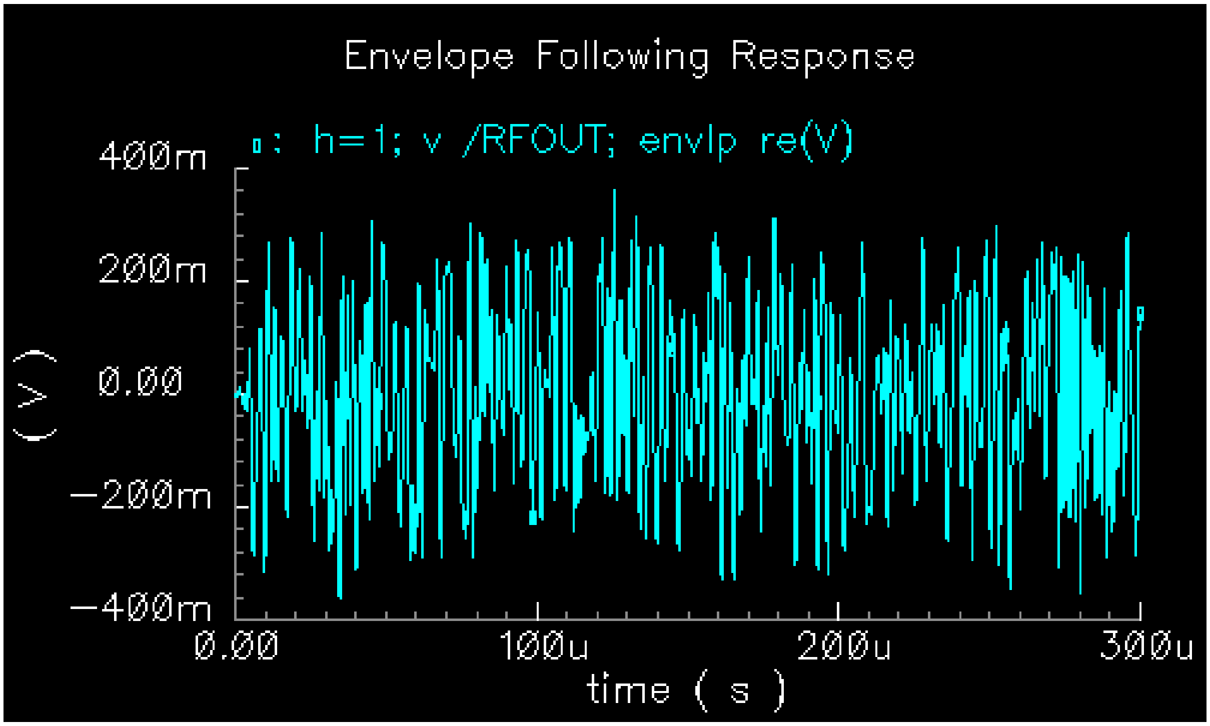
The completed form looks like this.



**3.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (/RFOUT).




A plot of the real portion of the transmitter output waveform appears in the new waveform window.



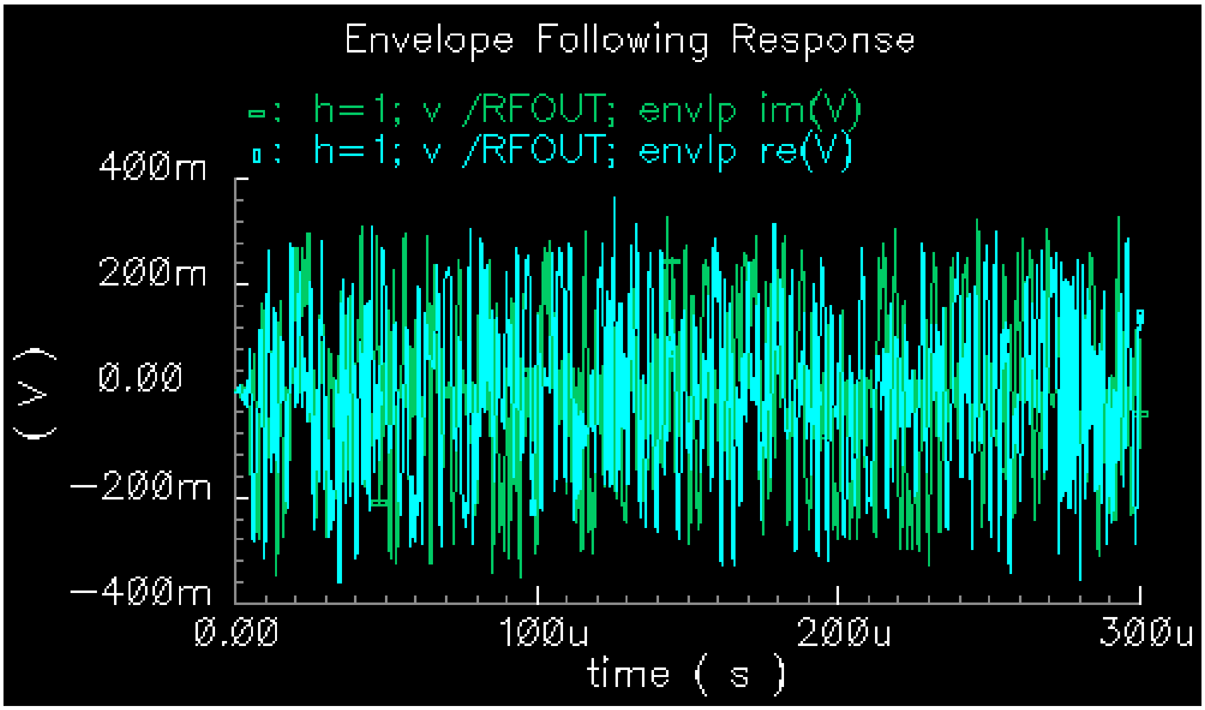

**4.** In the Direct Plot form, change the following:

**a.** Highlight *Append* for *Plotting Mode*.

**b.** Highlight *Imaginary* for *Modifier*.

**5.** Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (RFOUT).

A plot for the imaginary portion of the transmitter output waveform is added to the real portion in the waveform window.

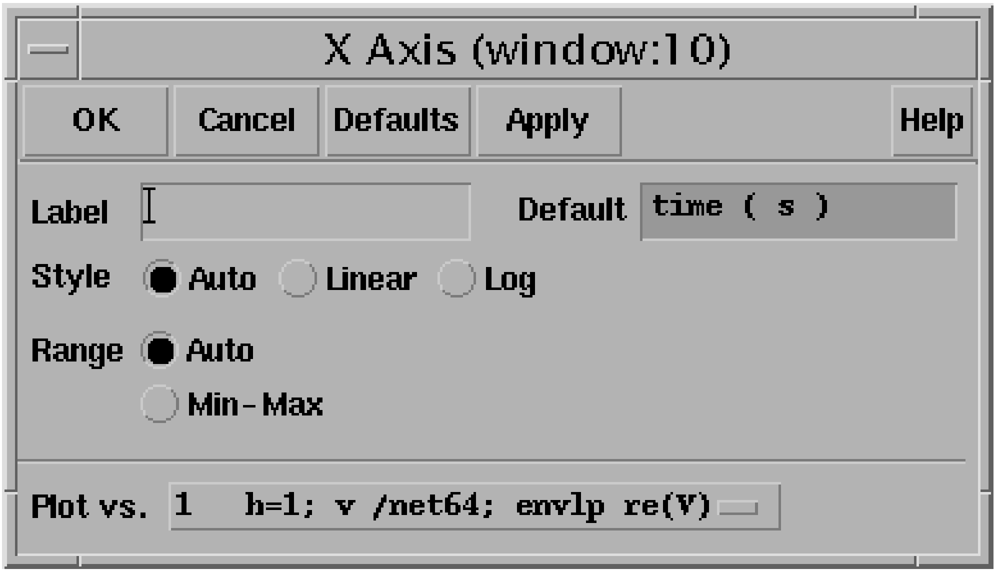



**6.** In the waveform window, choose *Axes – X Axis*.

The X Axis form appears.

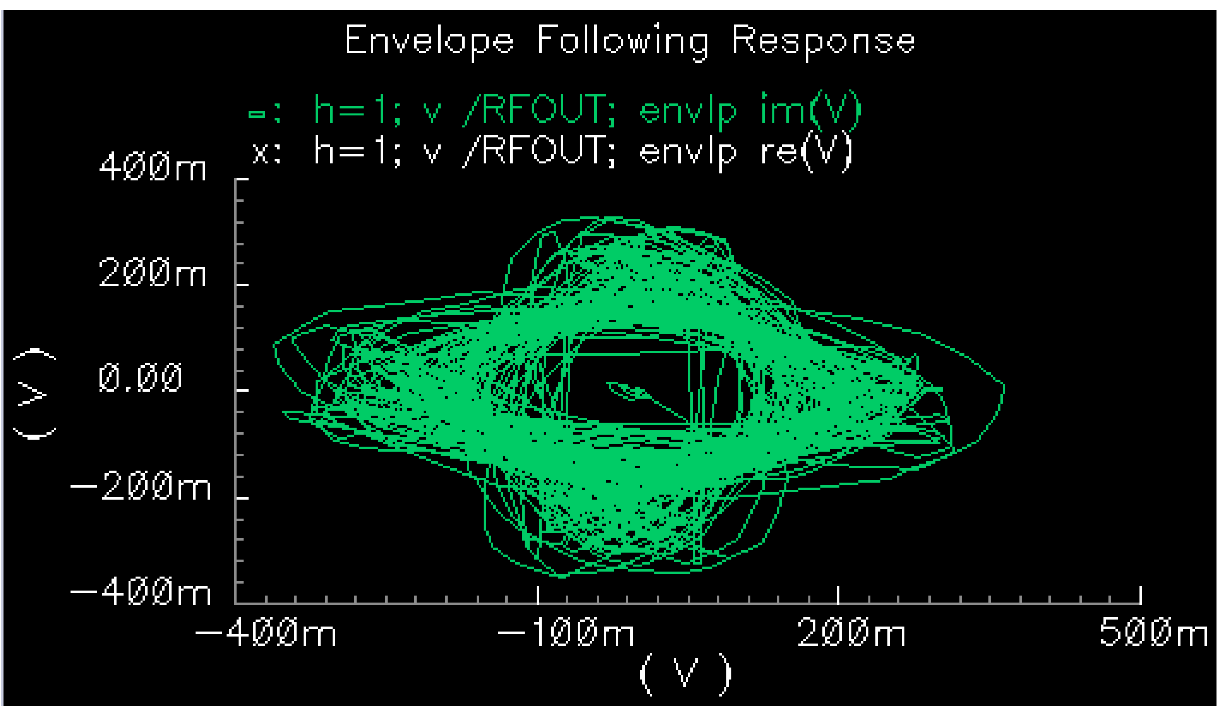
**7.** In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose

```
1    h-1;   v    /RFOUT;  envlp  re(V)
```




**8.** Click *OK*.

The output baseband trajectory appears in the second waveform window.



As shown in Figure 8-1 on page 556, the output baseband trajectory (on the right) is the input baseband trajectory (on the left) scaled linearly and rotated.

**Figure 8-1 Input and Output Baseband Trajectories**

In other words, the output baseband signal is the input baseband signal multiplied by a complex constant. The input and output waveforms look different because of the rotation, not because of some non-linear distortion. A common non-linear distortion, such as saturation, would make the outer edges of the trajectory lie on a circle.

# Measuring ACPR and PSD

Adjacent Channel Power Ratio (ACPR) is a common measure of the power a transmitter emits outside its allotted frequency band. ACPR is the ratio of the power in an adjacent band to the power in the allotted band.

$$ACPR = \frac{\text{Power in an adjacent band}}{\text{Power in the allotted band}}$$

Regardless of how you choose the frequencies and bands for the ACPR measurement, ACPR is always extracted from the power spectral density (PSD) of the transmitted signal. PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *envlp* analysis.

## The ACPR Wizard

The ACPR Wizard simplifies the complicated calculations needed to measure ACPR. It determines the appropriate *envlp* simulation parameters and `psdbb` function arguments from the information you supply in the ACPR wizard form. The ACPR wizard form contains a minimum number of clearly labeled fields so you can easily supply the required information.

After you provide the information and press *Apply* or *OK* in the ACPR wizard, the calculated values are used to fill in the *envlp* Choosing Analyses form and the *envlp* Options form. You can then run the simulation and analyze the PSD and ACPR data.

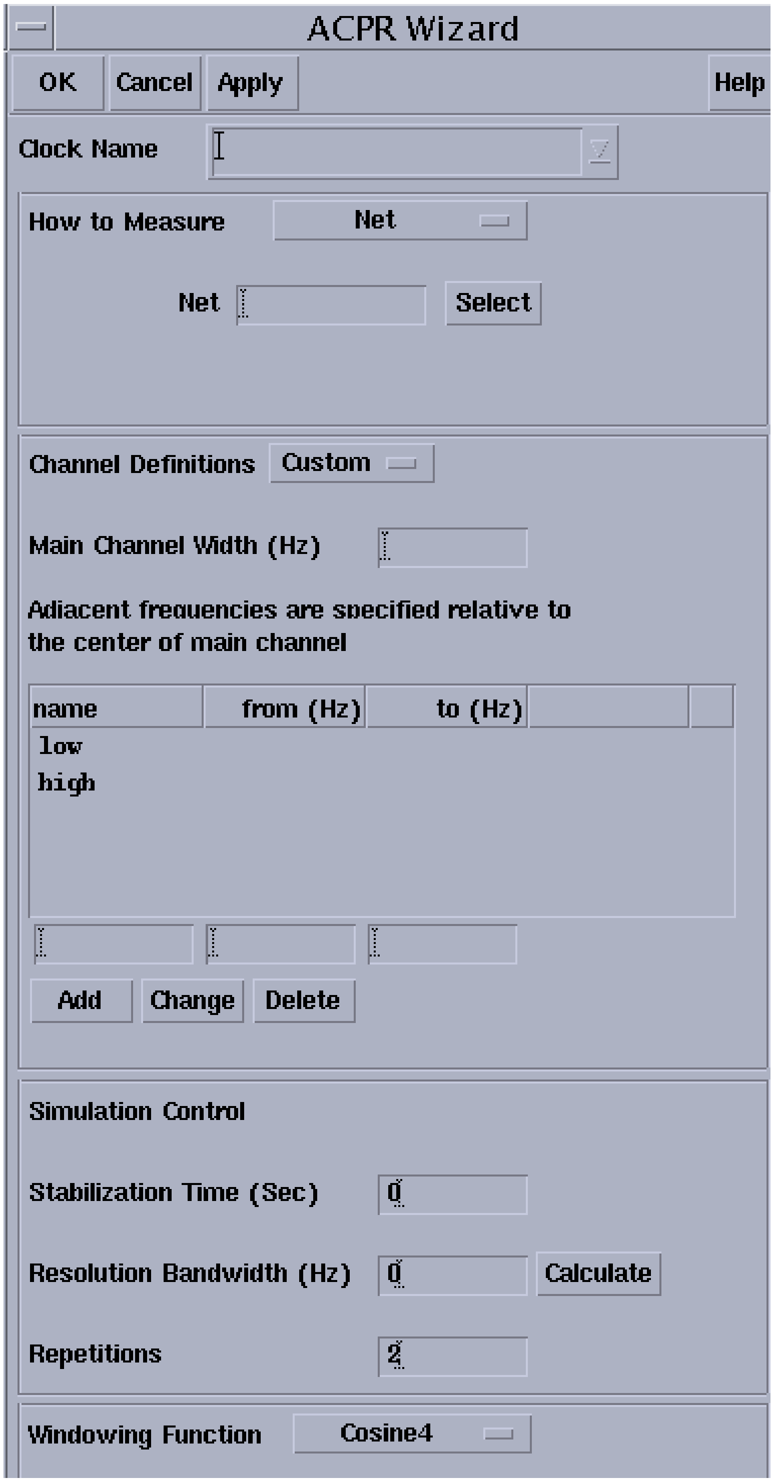
Open the ACPR wizard in one of two ways.

- In the ADE window, choose *Tools – RF – Wizards – ACPR*

  or

- In the *envlp* Choosing Analyses form, press *Start ACPR Wizard*.

In either case the ACPR Wizard displays.



## Measuring ACPR

The ACPR wizard minimizes the number of items you need to supply. Its fields are easy to understand and fill in. After you provide the ACPR wizard field values, these values are used to calculate and fill in the field values on both the *envlp* Choosing Analyses and Options forms. The ACPR wizard computes the appropriate simulation parameters and `psdbb` arguments from the information you enter in the ACPR wizard.

To use the ACPR wizard,

1. Fill in the ACPR wizard form.
2. Click *OK* or *Apply* in the ACPR wizard to calculate field values for both the *envlp* Choosing Analyses and *envlp* Options forms.
3. Run the *envlp* analysis. When the simulation completes, the *envlp* analysis results are plotted.

This example illustrates how to use the ACPR wizard and to set up and run the Envelope analysis. This example uses the *EF_example* circuit from your writable copy of the *rfExamples* library.

Before you start, perform the setup procedures described in Chapter 3 if you have not yet set up the writable copy of the *rfExamples* library.

Set up the *EF_example* schematic and the simulation environment.

1. Open the *EF_example* schematic as described in “Opening the EF_example Circuit in the Schematic Window” on page 522.
2. Open the ADE window as described in “Opening the Simulation Window” on page 523.
3. Set Up the Model Libraries as described in “Setting Up the Model Libraries” on page 524.
4. Edit the *PWL file names* for *PORT0* and *PORT1* as described in “Editing PORT0 and PORT1 in the EF_example Schematic” on page 525.

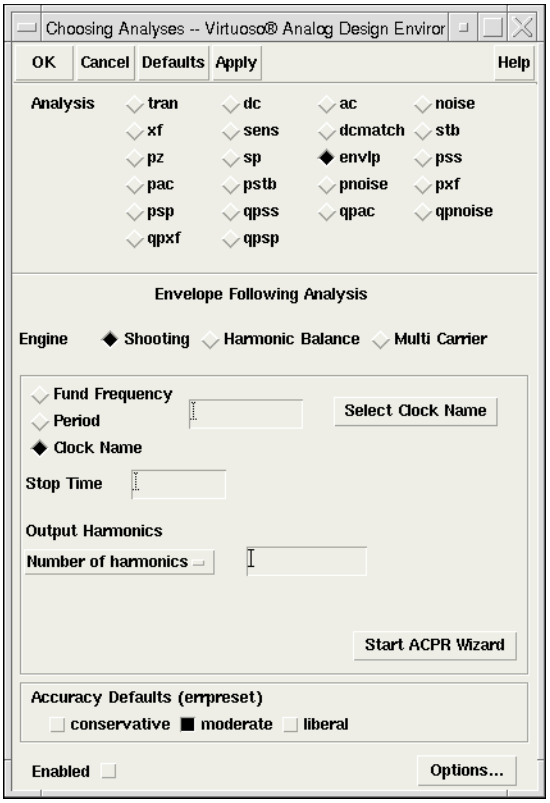
### Setting Up the ACPR Wizard and the envlp Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

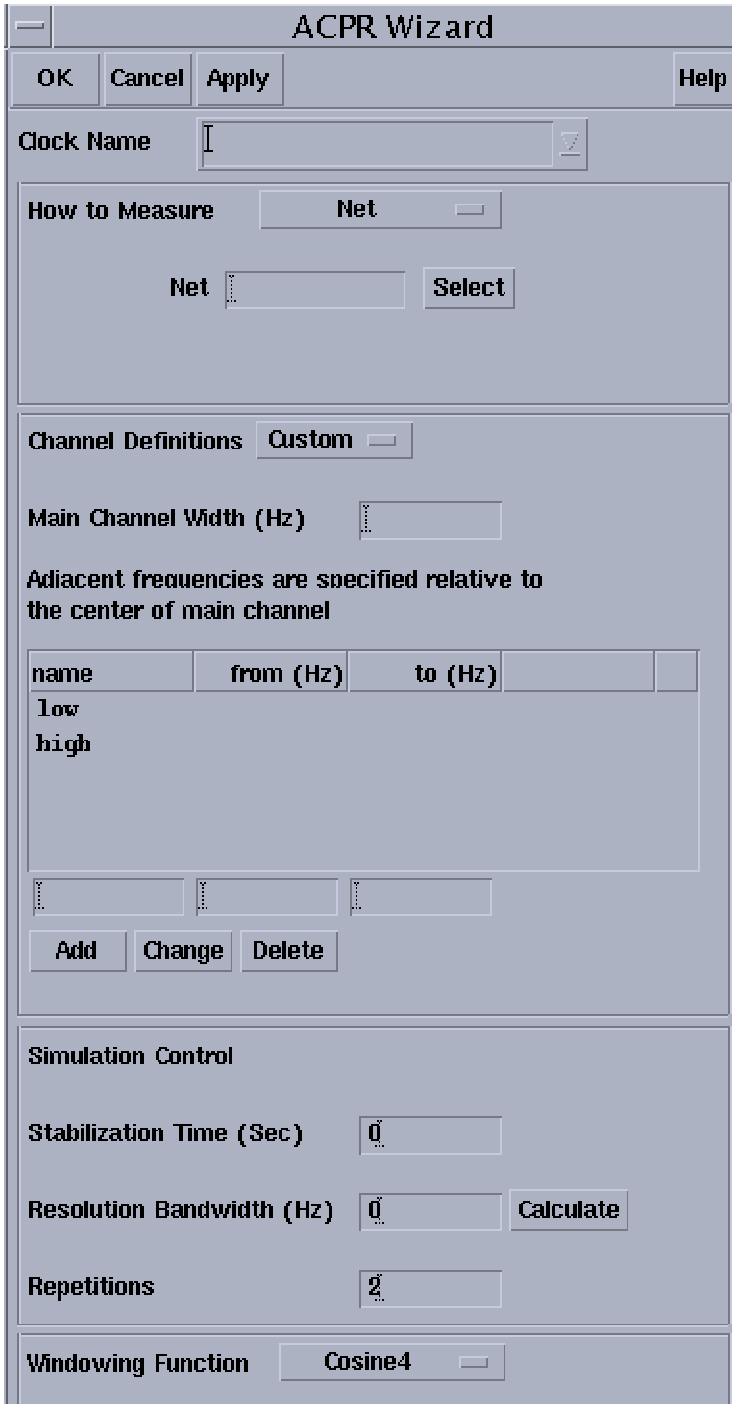
2. In the Choosing Analyses form, highlight *envlp*.

The Choosing Analyses form changes to let you specify values for the Envelope analysis.




3. In the *envlp* Choosing Analyses form, click *Start ACPR Wizard*.
4. Leave the *envlp* Choosing Analyses form open in the background.

The ACPR wizard appears.




**5.** In the ACPR wizard, do the following.

**a.** In the *Clock Name* cyclic field, select *fff*.

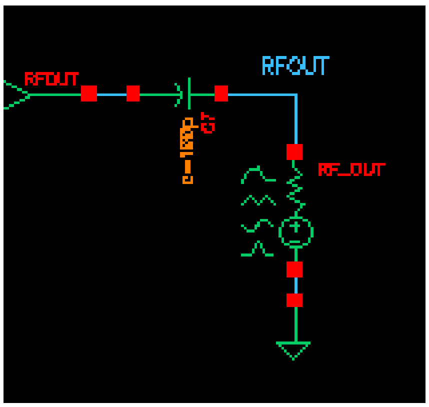
The clock name identifies the source of the modulated signal.

**b.** In the *How to Measure* cyclic field, select *Net*.

You can measure ACPR

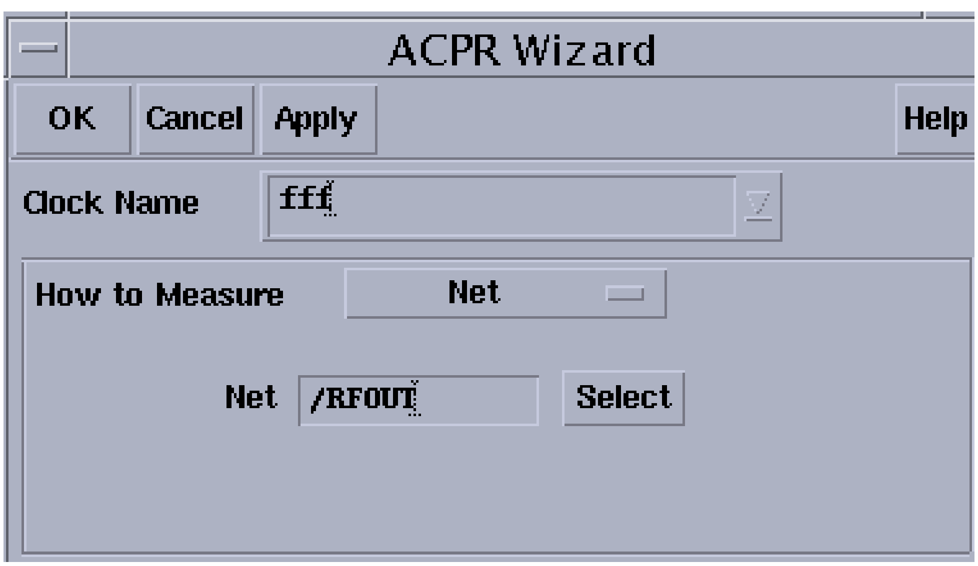
- For a single *Net* (Select one net, *Net*)
- Between *Differential Nets* (Select two nets, *Net+* and *Net-*)

**c.** To select a single output net in the Schematic window, click *Select* to the right of the *Net* field. In the Schematic window, select the transmitter output net, *RFOUT*.




The output net name, */RFOUT*, displays in the *Net* field.

The top of the ACPR wizard appears below.



**d.** In the *Channel Definitions* cyclic field, select the *IS-95* standard.

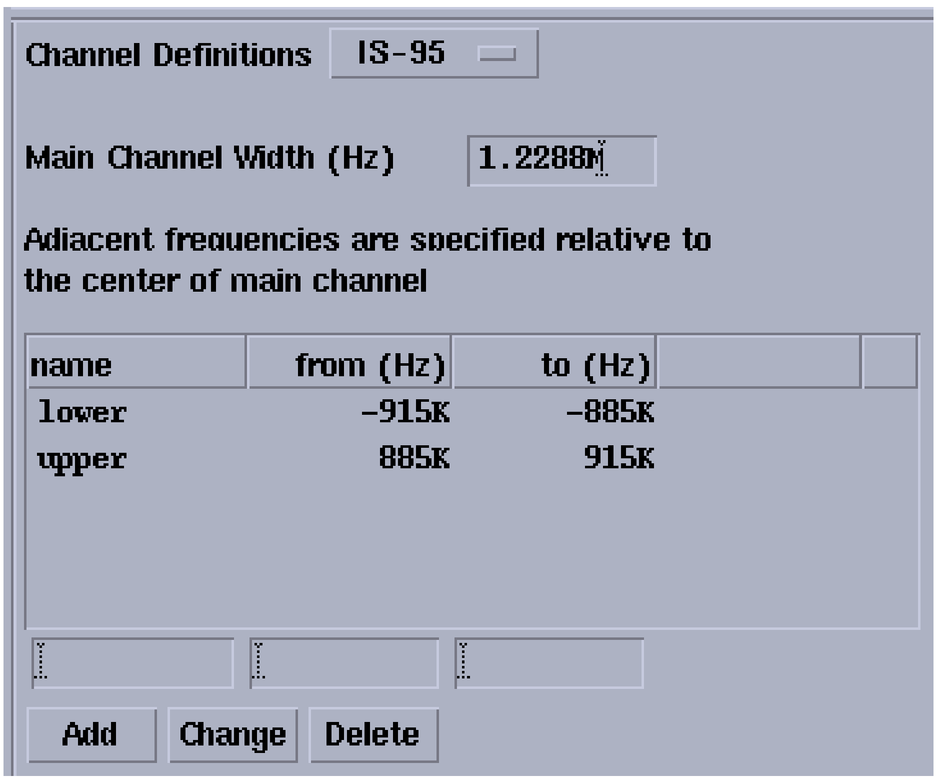
- ❍ The *Main Channel Width (Hz)* field is calculated as `1.2288M`.
- ❍ The *adjacent frequencies* are determined and display in the list box.

**e.** Note the text on the ACPR Wizard form,

*Adjacent frequencies are specified relative to*
*the center of the main channel*

- ❍ *Channel Definitions* presets include *Custom*, *IS-95*, and *W-CDMA*
- ❍ *Main Channel Width (Hz)* specifies a frequency band in Hz

The middle of the ACPR wizard appears below.




Use the edit fields below the list box and the *Add*, *Change* and *Delete* buttons to modify channel definitions in the list box.

You can hand edit or enter adjacent frequency names and *upper* and *lower* boundaries. Specify adjacent frequencies relative to the center of the main channel. All channel widths must be greater than zero.

**f.** In the *Stabilization Time (Sec)* field, enter `72n`.

`7.2e`$^{-08}$ displays in the *Stabilization Time (Sec)* field and 2 displays in the *Repetitions* field.

The *Stabilization Time* is the length of time in seconds to wait before using the data for analysis. *Repetitions* is the number of times to repeat the discrete Fourier transform for averaging.

Increasing the number of repetitions makes the power density curve smoother, but at the expense of longer simulation time and increased data file size.

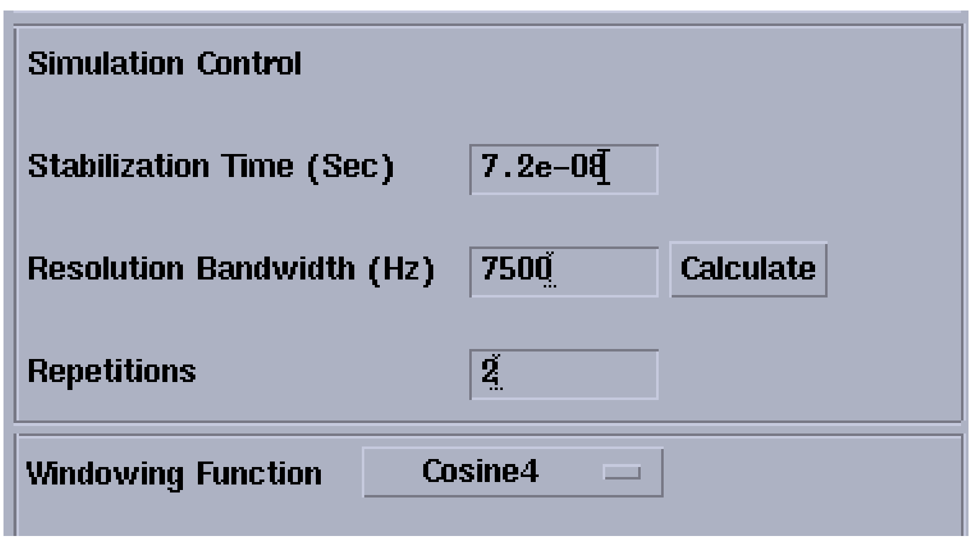
**g.** Click *Calculate* to determine the *Resolution Bandwidth (Hz)*.

`7500` displays in the *Resolution Bandwidth (Hz)* field. *Resolution Bandwidth* specifies the spacing of data points on the resulting power density curve, in Hz. Reducing *Resolution Bandwidth* increases simulation time and data file size.

**h.** In the *Windowing Function* cyclic field, select *Cosine4*.

*Windowing Function* presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.*

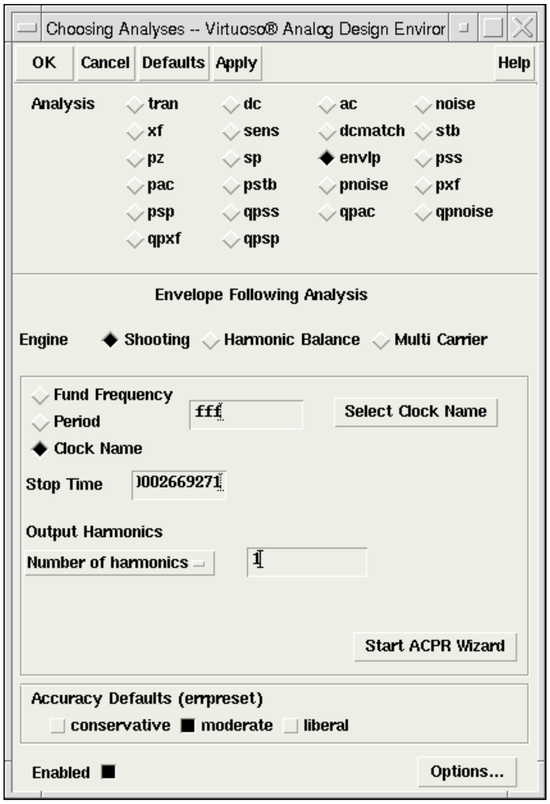
The *Simulation Control* section at the bottom of the ACPR Wizard looks like the following.




**6.** In the ACPR Wizard, click *Apply*

When you click *OK* or *Apply* in the ACPR wizard, values are calculated and appear in the *envlp* Choosing Analyses form.

If necessary choose *Choose – Analyses* in the ADE window to display the envlp Choosing Analyses form.




Values in the *envlp* Choosing Analyses form are as follows.

- ❑ The *Clock Name* is the same in both forms. In this case `fff`.
- ❑ *Stop Time* for *envlp* is calculated as `0.0002669271`.
- ❑ For the *envlp* analysis O*utput Harmonics, Number of Harmonics* is selected in the cyclic field and the *Number of Harmonics* is set to `1`.
- ❑ The *envlp* analysis is *Enabled*
- ❑ The *envlp* analysis *errpreset* is *moderate*.

Values in the *envlp* Options form are as follows.

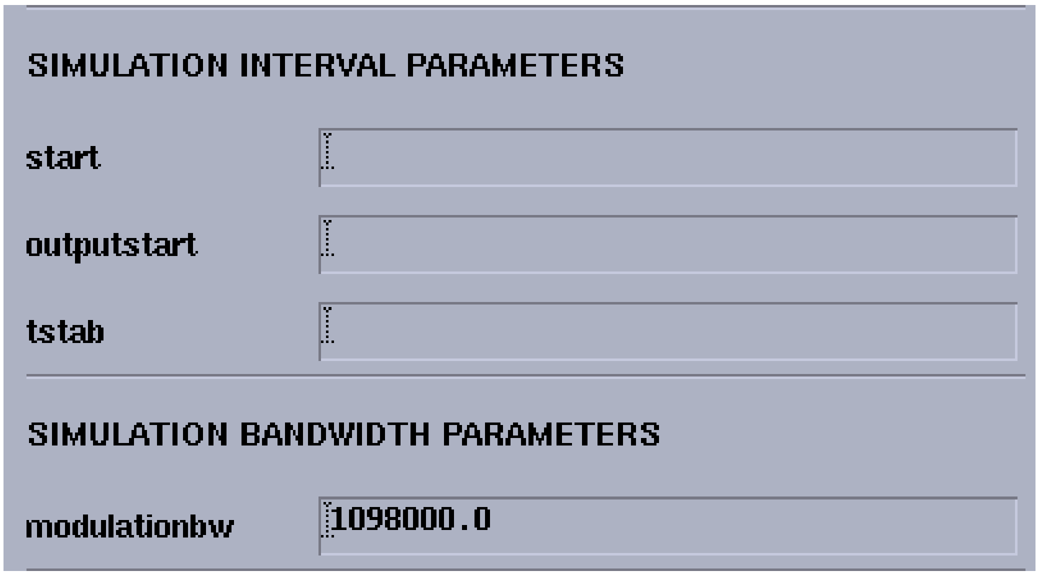
- ❑ The *envlp* option *start* is set to blank.
- ❑ The *envlp* option *modulationbw* is calculated as `1098000.0`
- ❑ The *envlp* option *strobeperiod* is calculated as `2.604167e-07`.

**Note:** You can modify values on the *envlp* Choosing Analyses and Options forms but your changes are not propagated back to the ACPR wizard.

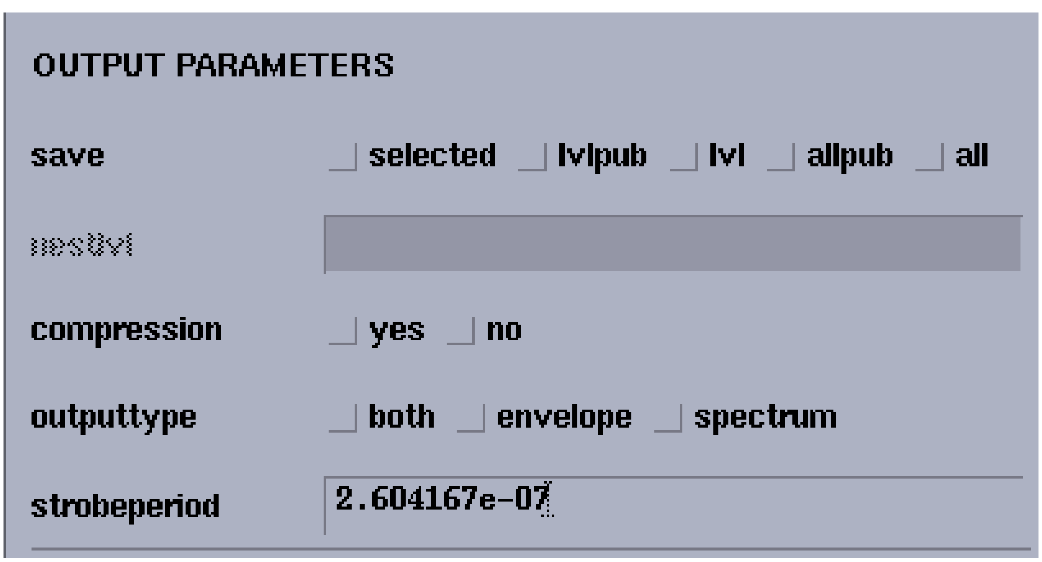
**7.** In the *envlp* Choosing Analyses form, click *Options* to open the *envlp* Options form.

Notice that values for the *modulationbw* and *strobeperiod* parameters are calculated and the *start* parameter is blank.

The section for the *start* and *modulationbw* parameters looks like this.




The section for the *strobeperiod* parameter looks like this.




**8.** Click *OK* in the Envelope Following Options form.

**9.** Verify that *Enabled* is highlighted and click *Apply* in the *envlp* Choosing Analyses form.

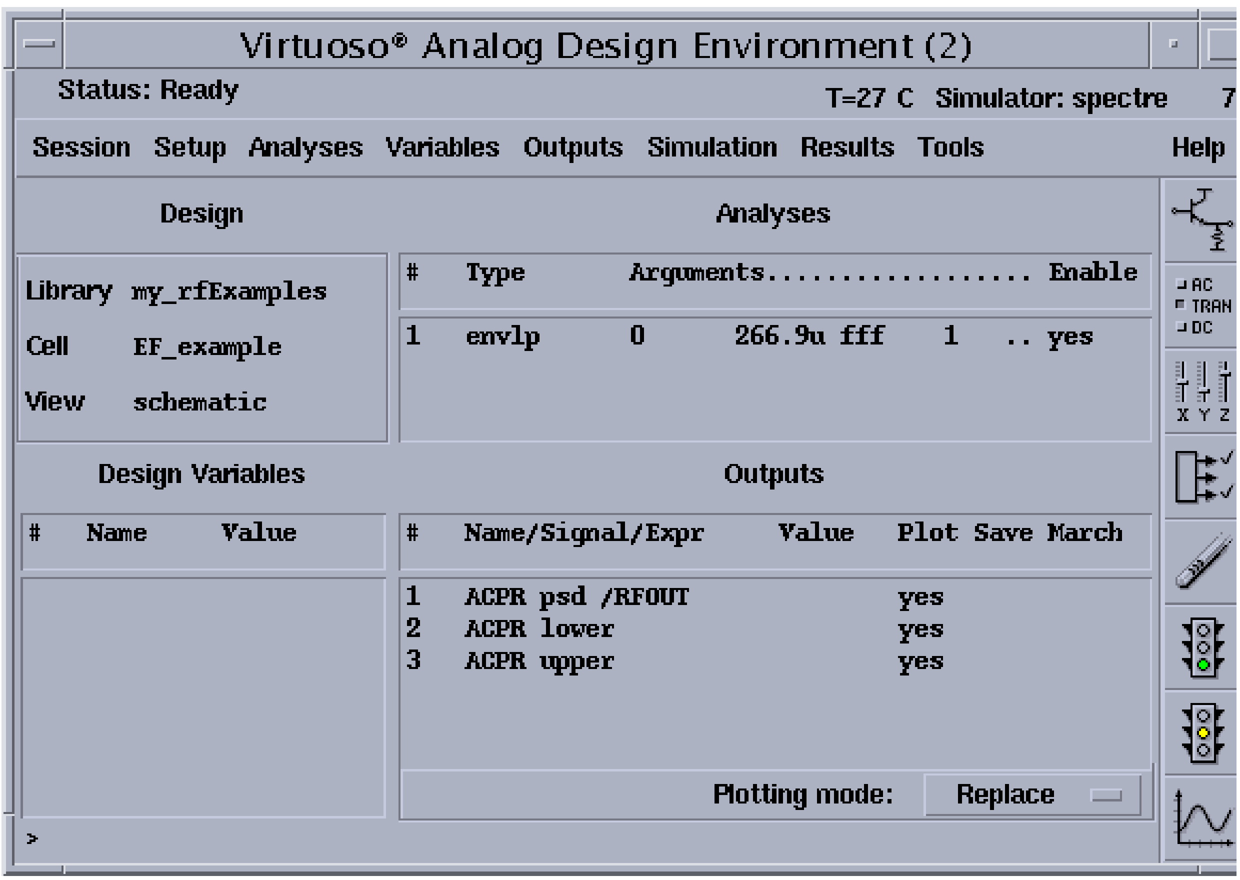
**10.** Click *OK* in the *envlp* Choosing Analyses form.

The Choosing Analyses form closes.

**11.** Click *OK* in the ACPR Wizard.

The ACPR Wizard closes.

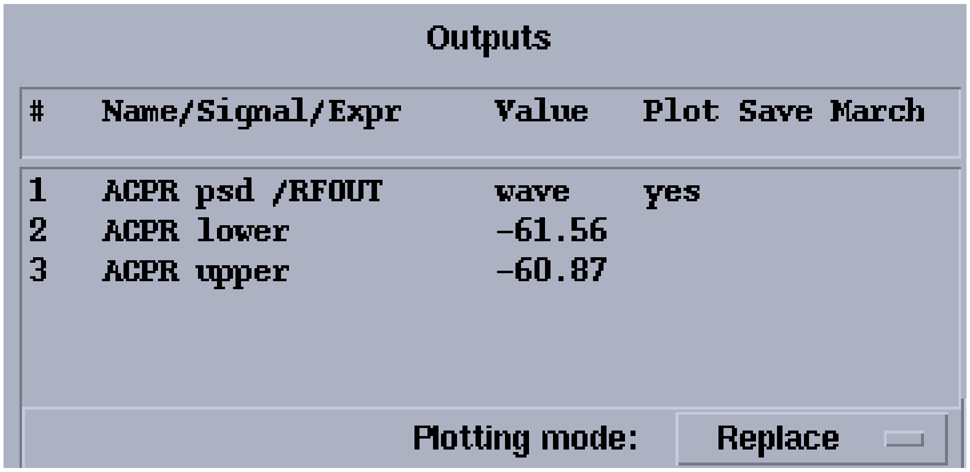
The ADE window reflects the *Analysis* and *Outputs* information from the ACPR wizard, the *envlp* Choosing Analyses form and the calculations that resulted.




**12.** In the ADE window, choose *Simulation – Netlist and Run*.

Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

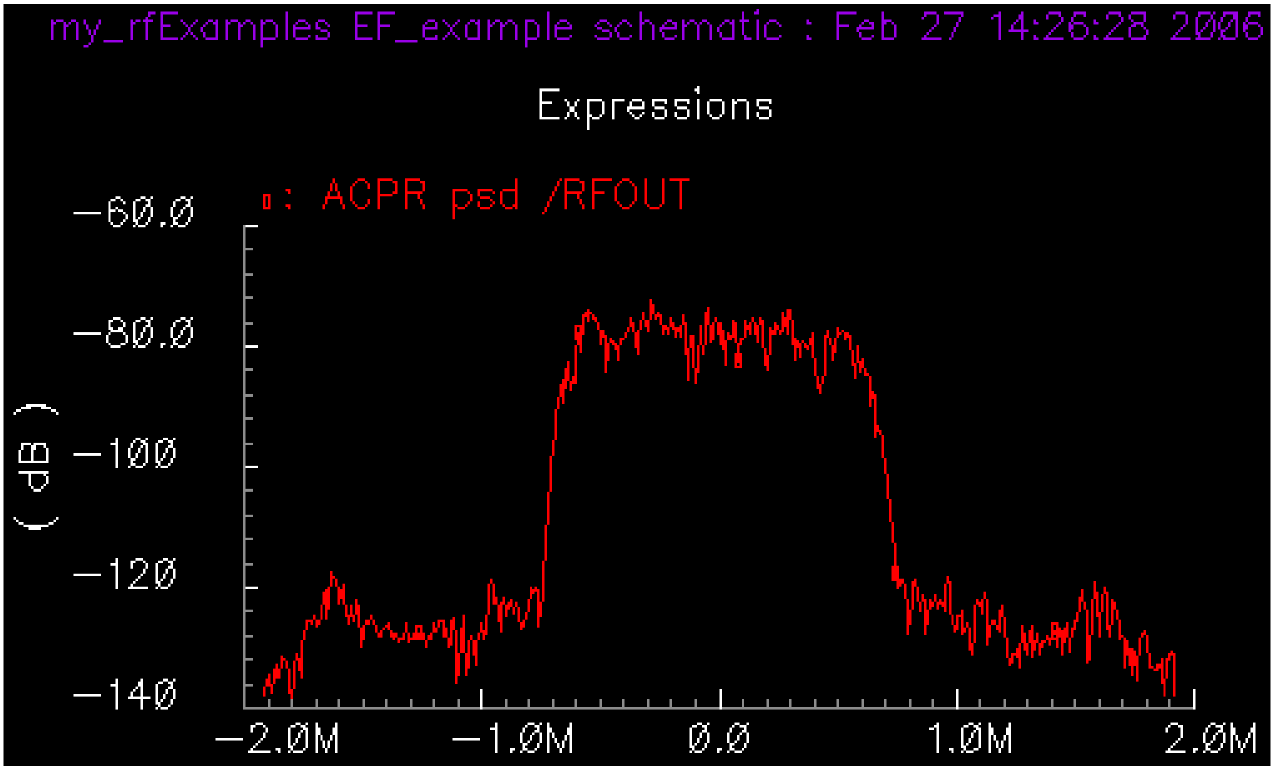
When the simulation successfully completes, an ACPR value for each channel appears in the *Value* column in the ADE window *Output* section.




- The *ACPR* `psd` */RFOUT Value* is a *wave* which is plotted.
- The *ACPR lower* channel value is `-61.56`.
- The *ACPR upper* channel value is `-60.07`.

When the simulation finishes, the PSD plot displays as in Figure 8-2.

**Figure 8-2 PSD Plot Generated by the ACPR Wizard**




## Estimating PSD From the Direct Plot Form

The power spectral density (PSD) is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *envlp* analysis.

This example shows how to estimate PSD given the results of the *envlp* analysis you just performed.

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

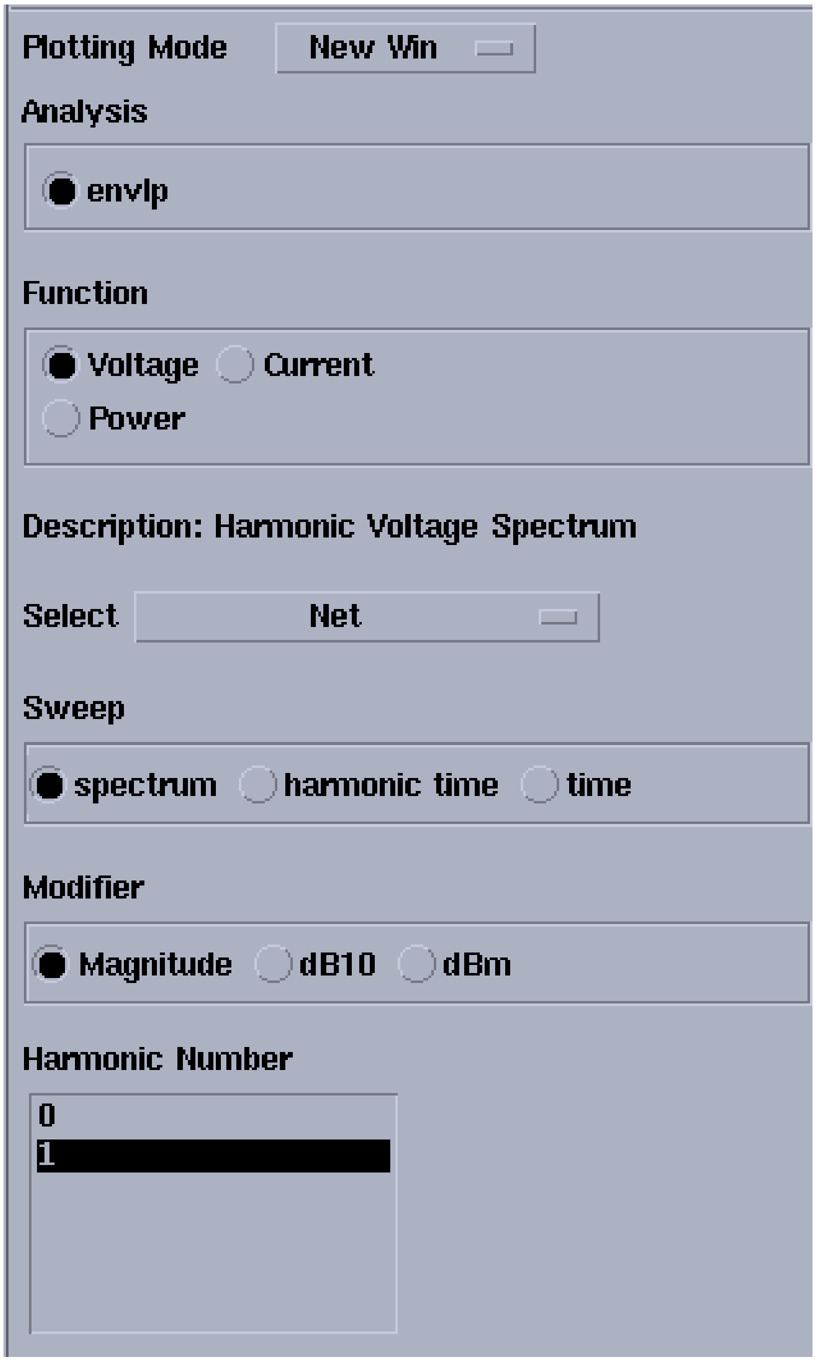
2. In the Direct Plot form, choose *New Win* for *Plotting Mode*.

This plots PSD in a new waveform window.

3. Highlight *envlp* for *analysis.*
4. Highlight *Voltage* for *Function*.
5. Highlight *spectrum* for *Sweep*.

   The *Power Spectral Density Parameters* section opens at the bottom of the form.
6. Highlight *Magnitude* for *Modifier*.
7. Choose *1* for *Harmonic Number*.

The top of the Direct Plot form looks like the following.




**8.** In the Bottom of the Direct Plot form, enter information for the *Power Spectral Density Parameters*

**a.** Press *Get From Data* to calculate the values for the *Time Interval* values, *From* is `0.0` and *To* is `0.0002669271.`

**b.** Type `5M` for *Nyquist half-bandwidth*.

**c.** Type `.1M` for *Frequency bin width*.

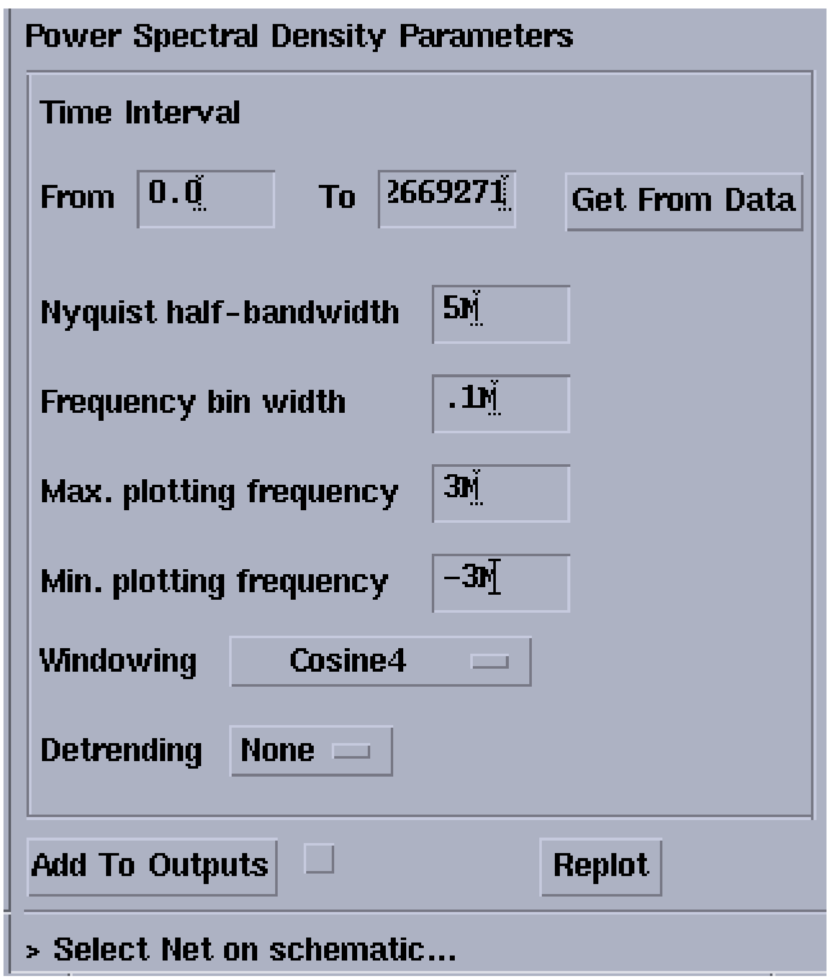
**d.** Type `3M` for *Max. plotting frequency*.

**e.** Type `-3M` for *Min. plotting frequency*.

**f.** In the *Windowing* cyclic field, select *Cosine4*.

**g.** In the *Detrending* cyclic field, select *None*.

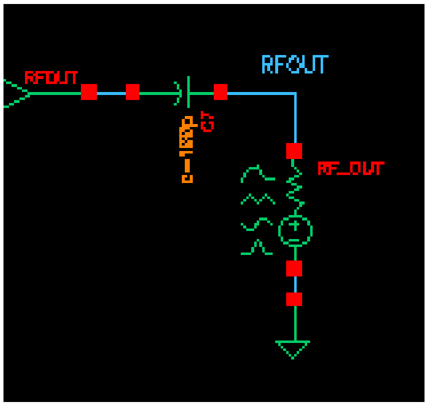
The *Power Spectral Density Parameters* area at the bottom of the *envlp* Direct Plot form is as follows.



**h.** Following the prompt at the bottom of the form,

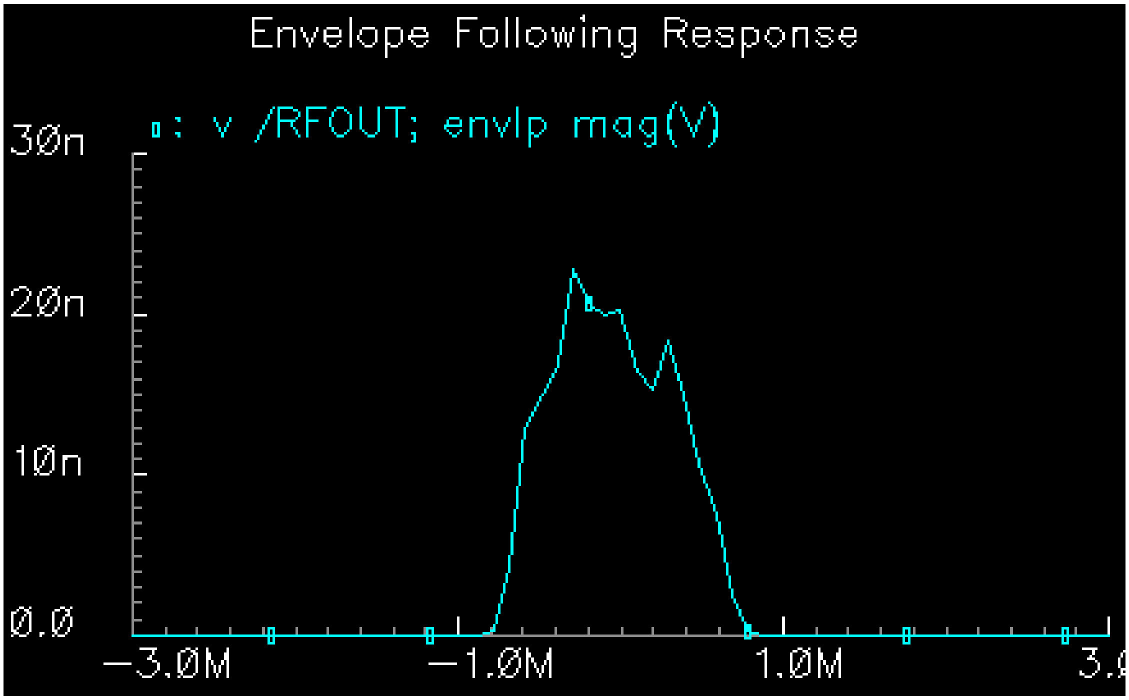
```
> Select Net on schematic.
```

Click the transmitter output net (/*RFOUT*).




The PSD plot displays.

**Figure 8-3 PSD mag (V) from the envlp Direct Plot Form**



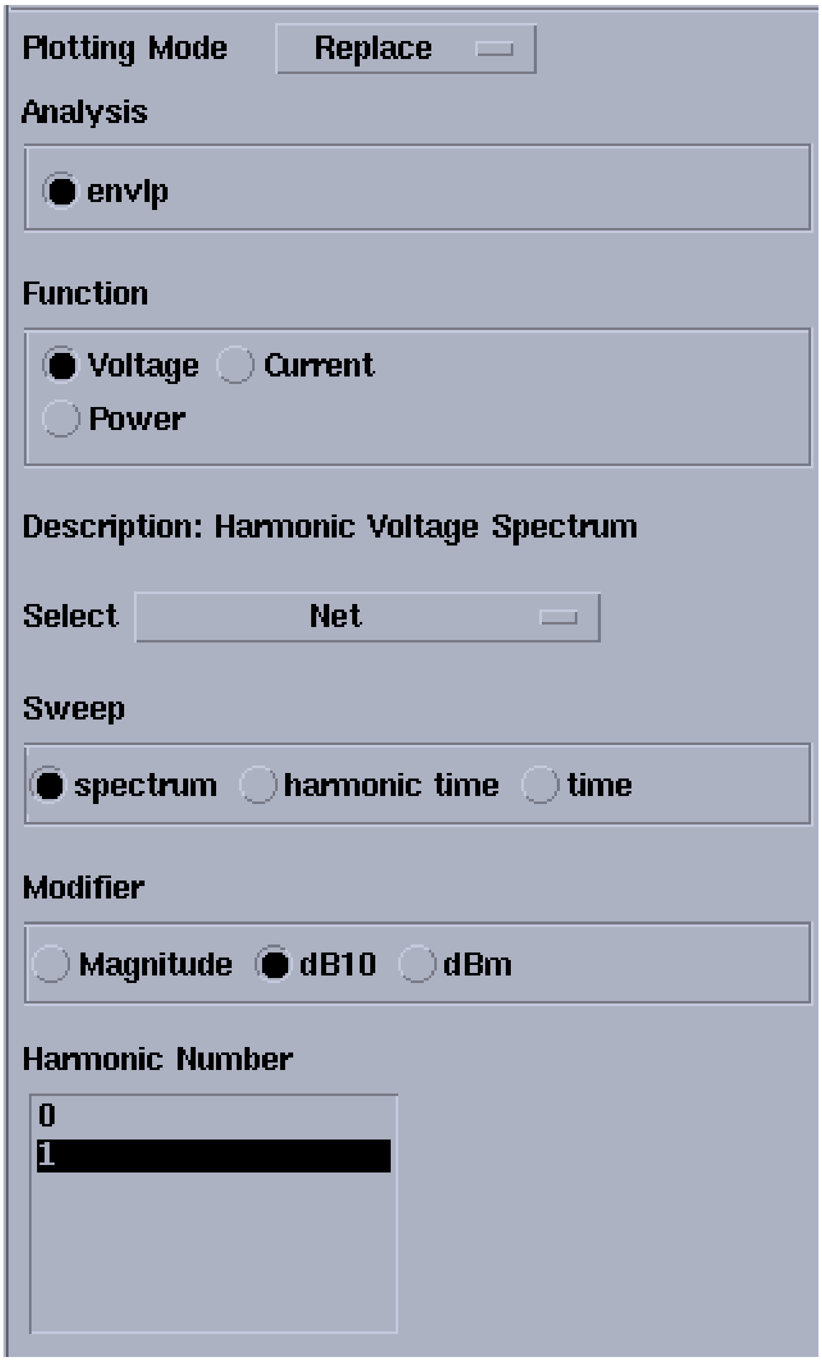

To plot the modified PSD plot,

**1.** In the Direct Plot form, make the following changes:

**a.** Choose *New Win* for *Plotting Mode*.

**b.** Highlight *dB10* for *Modifier*.

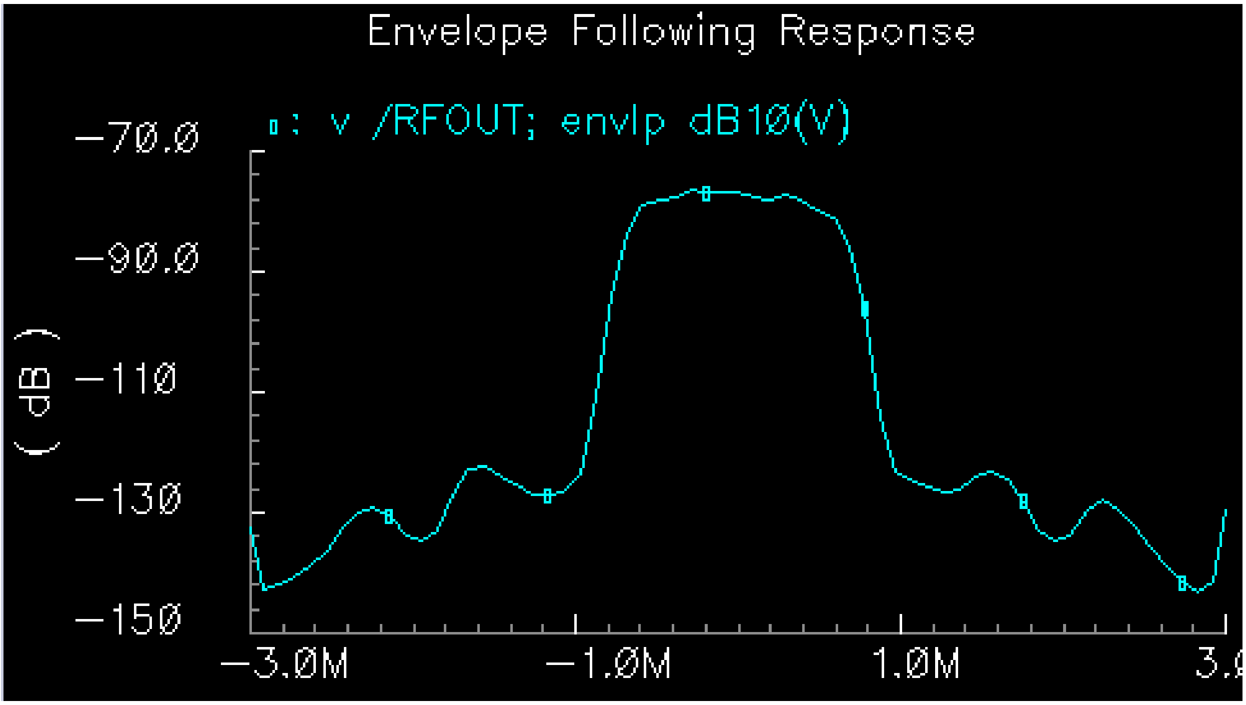
The top of the Direct Plot form looks like this.



**c.** Select the RFOUT net in the Schematic window.

The modified psd plot displays, as shown in Figure 8-4.

**Figure 8-4 Estimated PSD from the envlp Direct Plot Form**




Compare Figure 8-4 with the PSD plot shown in Figure 8-3 on page 573.

## Reference Information for ACPR and PSD Calculations

The process outlined in this section is complex largely because the *envlp* analysis parameters are not directly related to the calculator `psdbb` function arguments.

- The *envlp* parameters include nyquist half-bandwidth, frequency bin width and time interval.
- The *psdbb* function arguments are the total number of samples, the window size and the bin-width. All three parameters are in terms of the number of DFT time samples.

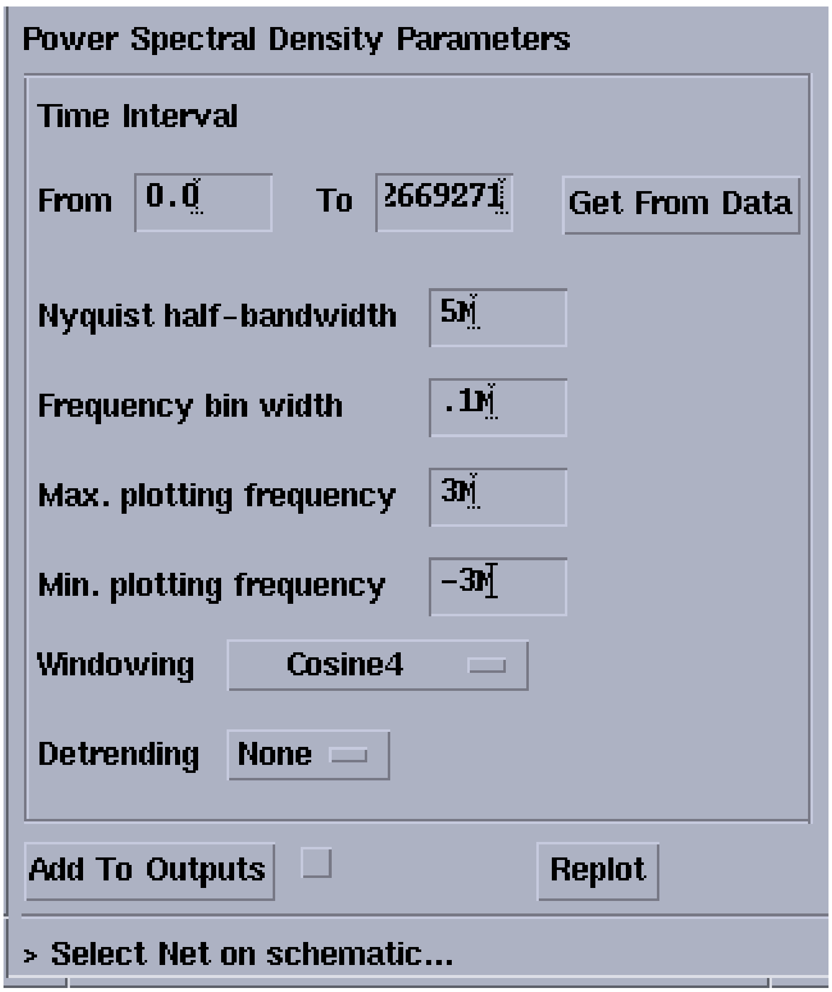
However, to make optimum use of the simulation data, it is necessary that the simulation parameters and the `psdbb` function arguments be compatible.

### The Power Spectral Density (PSD) Parameters

When you select *spectrum* for *Sweep* in the Direct Plot form, the *Power Spectral Density Parameters* section opens at the bottom of the Direct Plot form. The *Power Spectral Density Parameters* section displays the analysis parameters that control the PSD estimate.

The *Power Spectral Density Parameters* section is shown in Figure 8-4.

### PSD Parameters on the envlp Direct Plot Form



The PSD Parameters are described in Table 8-1.

**Table 8-1 Power Spectral Density Parameters From the Direct Plot Form**

| PSD Parameter | Description |
|---|---|
| *Time Interval* | The *Time Interval* parameters specify the time record to analyze in the frequency domain. The time interval should be long enough to support the required frequency resolution.<br><br>Use *Get From Data* to calculate the *From* and *To* values from the ACPR wizard data. |
| *Nyquist half-bandwidth* | The *Nyquist half-bandwidth* parameter is half the sampling frequency used in the DFTs. Make the nyquist half-bandwidth large enough to prevent aliasing. The true spectrum of the baseband signal should be negligible at this frequency and beyond it. |
| *Frequency bin width* | The *Frequency bin width* parameter specifies the required frequency resolution.<br><br>■ When this value is too small; the resulting PSD looks noisy and the PSD has a jagged appearance.<br><br>■ When this value is too large; the resulting PSD might be softened when the spectrum should have sharp edges.<br><br>*Windowing Function* presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.* |
| *Windowing* | The *Windowing* parameter specifies which windowing function to apply before performing the DFTs. |
| *Detrending* | The *Detrending* parameter has one of three values: *None*, *Mean*, or *Linear*. |

If necessary, the waveform is first interpolated to generate evenly spaced data points in time. The data point spacing is the inverse of the DFT sampling frequency. The PSD is computed by

- Breaking the time interval up into overlapping segments
- Multiplying each segment, time point by time point, by the specified *Windowing* function.

Windowing reduces errors caused by a finite time record. It is impossible to work with an infinite time record. Direct use of an unwindowed finite time record is equivalent to multiplying the infinite record by a rectangular pulse that lasts as long as the data record. Multiplication in the time domain corresponds to convolution in the frequency domain. The Fourier transform of a rectangular pulse is a *sinc* function. Considering the frequency domain convolution, the side lobes of the sinc function cause parts of the true spectrum to leak into the frequency of interest, that is, the frequency of the main lobe. Ideally, the sinc function would be a Dirac delta function but that requires an infinite time record. Good window functions have smaller side lobes than the sinc function.

The DFT is performed on each windowed segment of the baseband waveform. At each frequency, the DFTs from all segments are averaged together. Fewer segments means fewer data points in the average at a particular frequency. The length of each segment is inversely proportional to the *Frequency bin width*, which is why a small *Frequency bin width* produces a jagged PSD. A smaller *Frequency bin width* means a longer time segment.

Fewer long segments fit into the given time interval so there are fewer DFTs to average together. In the extreme, there is only one segment and no averaging. Without averaging, the PSD is the square of the magnitude of the DFT of a stochastic process. At the other extreme, large *Frequency bin widths* produce lots of points to average at each frequency but there are fewer frequencies at which to average because fewer large bins fit into the Nyquist frequency. The PSD is smoother but it does not have as much resolution.

PSD is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

The calculator *psdbb* function performs a discrete Fourier transform (DFT) on the voltage curve which produces the power spectral density (PSD) curve. Integrate the PSD curve to calculate power in the channel.

### The Calculator psdbb Function

The calculator *psdbb* function, which estimates PSD, derives the function parameters it requires from the PSD parameters and values you supply in the envlp Direct Plot form.

- *Nyquist half-bandwidth*
- *Frequency bin width*
- *Time Interval*

See Table 8-1 for more information about the PSD parameters on the Direct Plot form.

The calculator *psdbb* function parameters are

- The total number of samples
- The window size
- The bin-width

All three parameters are in terms of the number of DFT time samples.

The calculations defined in Table 8-2 generate the `psdbb` parameters. The calculated *psdbb* parameters are printed in the CIW window.

**Table 8-2 Parameter Values Calculated by the psdbb Calculator Function**

| psdbb Calculation | Source of the Data |
|---|---|
| `L = ` *`To – From`* | *`To`* and *`From`* are the values from the *Time Interval* *`To`* and *`From`* fields on the *envlp* Direct Plot form. |
| $f_{max}$ | *Nyquist half-bandwidth* and *Frequency bin width* are values from these fields on the *envlp* Direct Plot form. |
| `#bins = floor(L*binwidth)` | `#bins >= 1`. Here, *floor* means *to take the integer part of*, i.e. truncate to the nearest integer. |
| $2^m * (\#bins) > 2 * L * f_{max}$ | Compute the smallest `m`. |
| $windowsize=2^m$ | |
| `#bins * windowsize` | Compute the number of samples. |

You might want to use the `psdbb` function directly when strobing time-harmonic results to eliminate interpolation error. Use of the `psdbb` function is described in the calculator documentation.

### Calculating ACPR

This section describes how to calculate the ACPR (Adjacent Channel Power Ratio) by hand. The ACPR Wizard performs these calculations for you.

ACPR is measured with respect to any $x_1$ and $x_2$, as $y_1$ - $y_2$. You can calculate the ACPR for any two x-axis values by subtracting their associated y-axis values.

For example, given the following two spectral parameters,

- The adjacent channel is 2.5 MHz from the carrier
- You choose to define ACPR in terms of power at just two frequencies,
    - 2.5 MHz
    - 0 MHz

Use the cursor to determine the RF output power at 2.5 MHz and 0 MHz frequencies. As you slide the cursor along the curve, you read the X and Y values off the top of the waveform window:

```
X:2.5M    Y:-115.8        v /RFOUT;envlp mag(V)
```

and

```
X:0    Y:-77.33        v /RFOUT;envlp mag(V)
```

So

- The power at 2.5 MHz equals -115.8 dB
- The power at 0 MHz equals -77.33 dB

Remember that the horizontal scale is frequency offset from the carrier fundamental. The difference between these measurements equals -38.47 dB.

$$ACPR = -115.8 - (-77.33) = -38.47$$

For this definition of ACPR, ACPR = -38.47 dB.

Other definitions of ACPR are possible. You can compute them by adjusting the spectral parameters and applying the calculator to the spectral plot.

### Calculating PSD

For a constant baseband signal, calculate power spectral density as shown in Table 8-3.

**Table 8-3 Power Spectral Density for a Constant Baseband Signal**

| | |
|---|---|
| Baseband signal | $1 + j \times q$<br>where<br>$j = \sqrt{-1}$ |

**Table 8-3 Power Spectral Density for a Constant Baseband Signal**

| | |
|---|---|
| Passband signal | $i \times \cos(w \times t) - q \times \sin(w \times t)$ |
| Baseband signal power | $i \times i + q \times q \qquad volts \times volts$ |
| Passband signal power | $i \times \frac{i}{2} + q \times \frac{q}{2} = (1/2) \times (baseband""power)$ |

Envelope Spectral analysis computes these values per Hz. You can also express these values as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts}) \;/ (Hz)$$

Envelope Time-Harmonic analysis computes peak volts because they can be directly compared with the modulating signals.

For this example

- You see a waveform displayed in the waveform window as $V^2$ / ($Hz$) versus frequency.
- You can think of this as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts}) \;/ (Hz)$$

*Important*

The displayed spectrum is the estimated PSD of the complex envelope divided by two. The division by two is included because the envelope is expressed in units of peak carrier volts, but power in the carrier equals the square of the peak divided by two. It is convenient to express the envelope in peak units because you can then directly compare it against an input baseband signal.

### PSD and the Transmitted Baseband Signal

When you measure ACPR, it is crucial that you drive the transmitter with the proper baseband signals. The baseband signals driving the transmitter dominate the transmitted PSD. In most cases, the baseband signals are produced by digital filters so the digital filters constrain the spectrum of the input baseband signal. Distortion in the transmitter causes the spectrum to grow where it should not, hence the need for an ACPR measurement.

It is not practical to model digital filters in Spectre RF because Spectre RF cannot simulate state variables inside Verilog®-A modules. Consequently, for now, you must pre-compute and store the baseband inputs and then read them into the Spectre RF analysis through *ppwlf* sources found in the *analogLib*, as shown in "Computing the Spectrum at the Adder Output" on page 582. The *ppwlf* sources also read SPW format, so you can also generate and record the input baseband waveforms using SPW.

The *rfLib* contains three sets of stored baseband waveforms, *cdma*, *dqpsk*, and *gsm*. These waveforms were created with the baseband signal generators in the *measurement* category of the *rfLib*.

If you want to measure ACPR with the noise floor much more than 40 dB below the peak of the output power spectral density, you must create baseband drive signals with a noise floor at or below the required noise floor. If you use a DSP tool such as SPW to create the signals, the filters in the baseband signal generator must operate perhaps hundreds of times faster than those in the actual generator.

Sometimes an ACPR specification exceeds the ACPR of the baseband drive signals. To see if the transmitter meets specifications in that event, it must be driven with an unrealistic baseband signal. Otherwise the signals do not have enough resolution. The noise floor depends heavily on interpolation error.

The Fourier analysis used to compute the power spectral density uses evenly spaced time points. If data does not exist at one of the Fourier time points, the Fourier algorithm must interpolate to create the missing Fourier time point.

You can strobe the harmonic time results to eliminate interpolation of the output but you cannot eliminate interpolation of the baseband drive signals. The only way you can reduce interpolation errors at the input is to use ultra-high-resolution drive signals so that no matter where the interpolation occurs, the error is small. For now, you must generate the ultra-high-resolution drive signals yourself.

### Computing the Spectrum at the Adder Output

The following results were obtained by

1. Generating high resolution baseband signals in SPW
2. Storing the high resolution baseband signals
3. Reading the high resolution baseband signals into an *envlp* analysis through the *pwl* sources

The SPW FIR filters operate at 300 times the chip rate.

Figure 8-5 shows the spectrum at the output of the adder. In this circuit, the RF signal is undistorted at the adder output. The spectrum was computed three times. Each time with a different value for *number of samples*.

The envlp analysis was run with the following parameters:

| | |
|---|---|
| *reltol* | $1e^{-5}$ |
| *strobeperiod* | `100n` |

**Note:** If the *strobeperiod* does not equal an integer number of clock cycles, it is internally truncated to an integer number of clock cycles.

For the `psdbb` function

| | |
|---|---|
| *Time interval* | `100u` to `1m` |
| *Window size* | `1024` |
| Windowing | *Hanning* |

**Note:** *Window size* must be a power of 2. If you enter a value that is not a power of 2, the `psdbb` function truncates the value to the nearest power of 2.

With *strobeperiod* set to `100` ns (100 clock cycles), the simulation produced 9000 evenly spaced samples inside the time interval. The spectra were calculated using 8500, 8750, 9000, 9250, and 9500 points for the *number of samples*. Figure 8-5 clearly shows interpolation errors when the number of samples used in the `psdbb` function does not equal the actual number of samples.

**Figure 8-5 Adder Output Spectrum Computed with Three Different Numbers of Samples**

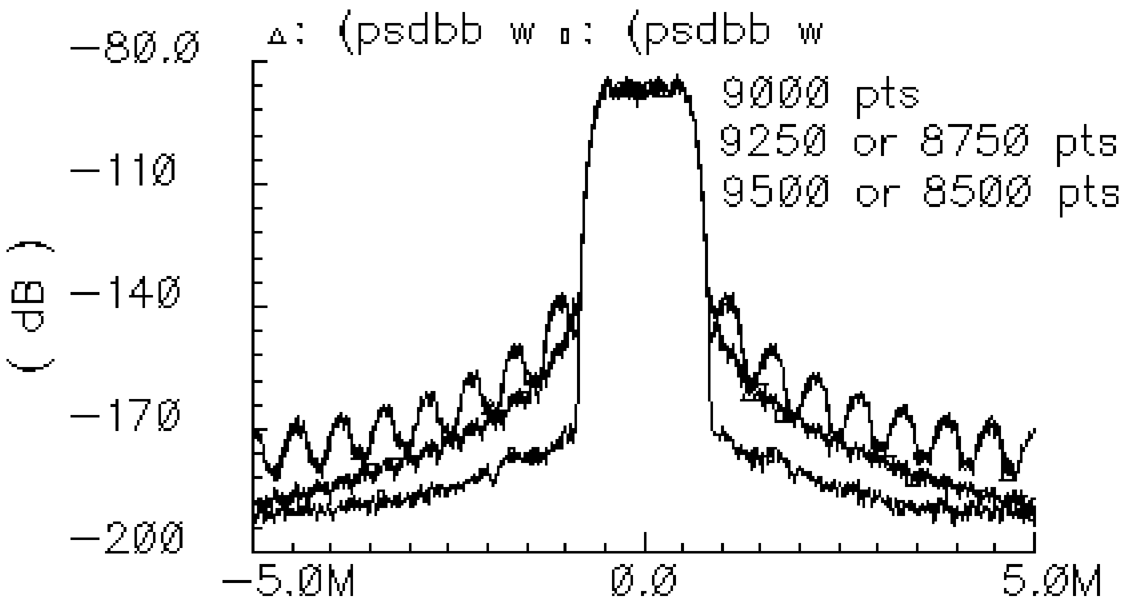
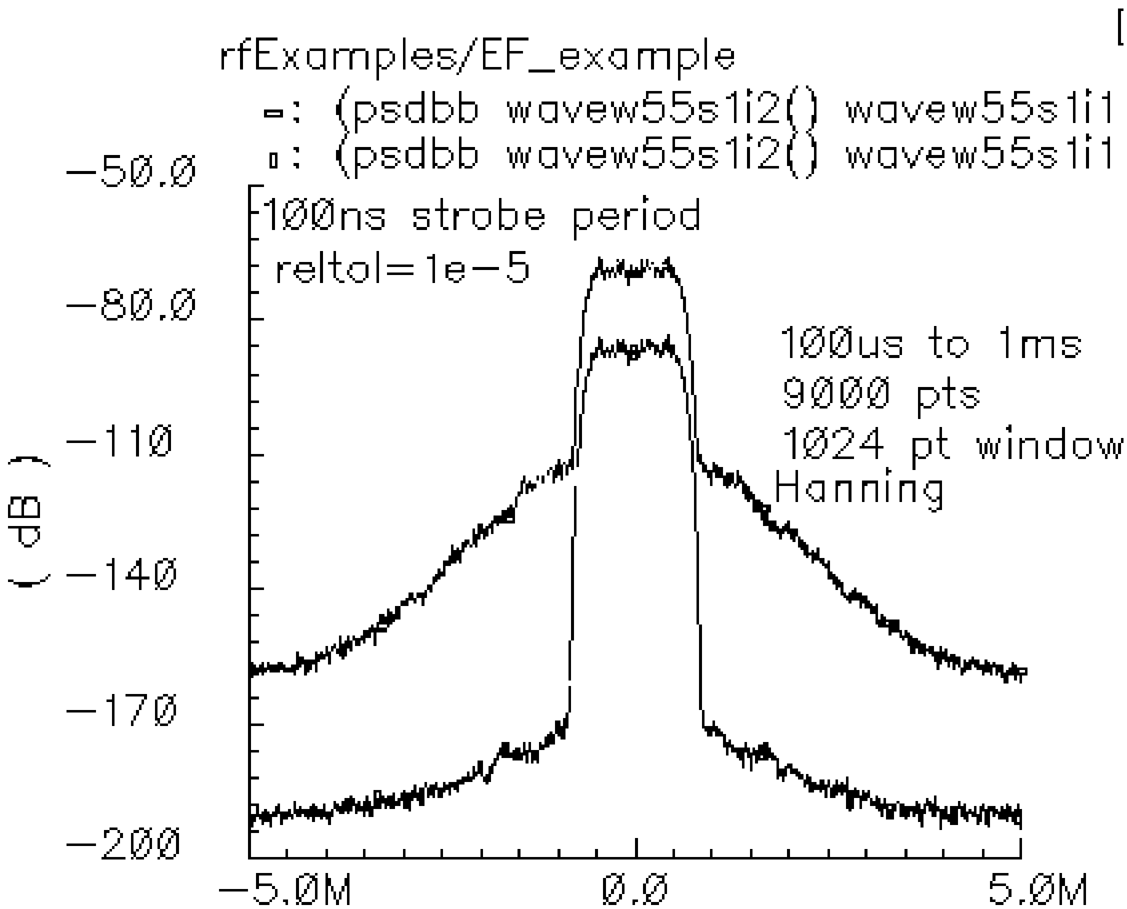



Figure 8-6 compares the input and output PSD plots using the high resolution drive signals and the following set of `psdbb` parameters:

| | |
|---|---|
| *From* | `100u` |
| *To* | `1m` |
| Number of Samples | 9000 |
| Window Size | 1024 |
| Window Type | Hanning |

In Figure 8-6, the upper PSD plot is the power amplifier output and clearly shows spectral regrowth when compared to the input PSD plot.

**Figure 8-6 Comparison of Input and Output PSD plots**



# Measuring Load-Pull Contours and Load Reflection Coefficients

This section describes how to generate load pull contours and how to determine whether you must redesign the input matching network for the optimal load.

A *load-pull contour* is a set of points on a Smith chart representing all the loads that dissipate a given amount of power. Load-pull contours help you match a load to a power amplifier for maximum power transfer. Just as a topographical map shows a hiker where the mountain peak lies and how steep the climb is, load-pull contours show which load dissipates the most power and how sensitive that power is to small load perturbations.

To properly use Load-Pull results, you must understand how the Load-Pull feature defines load reflection coefficients. Load reflection coefficients are computed using the PSS analysis.

The *load reflection coefficients* are computed from the load impedance, which is computed as the ratio of the voltage across the load to the current flowing into the load at the RF carrier frequency. Suppose the RF carrier is 1 GHz and the load waveforms are distorted. The impedance is not computed from small-signal perturbations about an operating point. Rather, the impedance is computed as the ratio of the 1 GHz components of the load voltage and current waveforms simulated by a PSS analysis. Because the load is passive and linear, the load reflection coefficient computed in this manner equals the small-signal, or incrementally computed, load impedance.

This is not necessarily true for the *input reflection coefficient* because the input circuitry can be non-linear and it can also contain input offset voltages and currents. However, because matching networks are usually designed only for the RF fundamental, defining the reflection coefficient as the ratio of fundamental Fourier components of the large signals is often justified.

## Creating and Setting Up the Modified Circuit (EF_LoadPull)

This example tells you how to

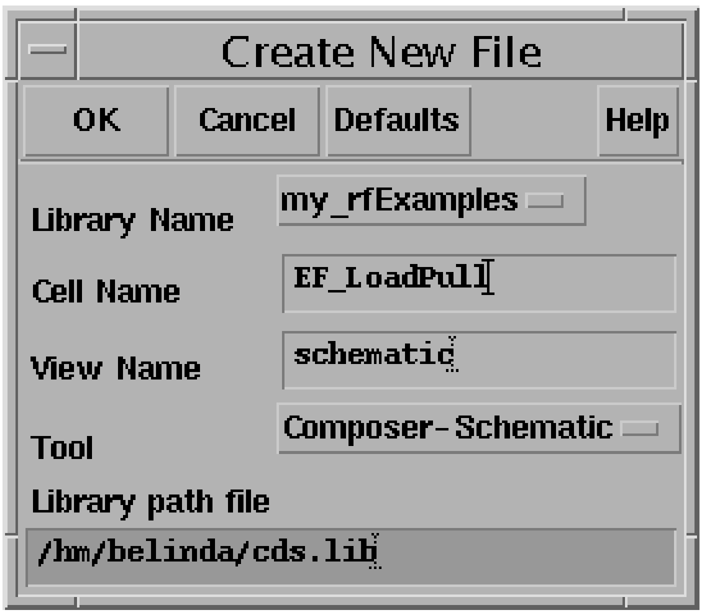
- Create the *EF_LoadPull* schematic for this example. The *EF_LoadPull* schematic is a modified copy of the *EF_example* schematic from the *rfExamples* library.
- Set up and run the necessary PSS and Parametric analyses
- Measure load-pull contours for the modified *EF_LoadPull* schematic
- Measure load reflection coefficients for the modified *EF_LoadPull* schematic

Before you start, make sure you have performed the setup procedures for the writable *rfExamples* library, as described in Chapter 3.

### Creating a New Empty Schematic Window

1. In the CIW choose *File – New – Cellview*.

   The Create New File form appears.




2. In the Create New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

      In the *Cell Name* field, enter *EF_LoadPull*.

   b. In the *View Name* field, enter *schematic*.

   c. In the *Tool* cyclic field, select *Composer-Schematic*.

   d. Click *OK*.

   A new, empty Schematic window named *EF_LoadPull* opens.

### Opening and Copying the EF_example Circuit

Now open another schematic window containing the *EF_example* circuit. Then copy the *EF_example* circuit into this empty *EF_LoadPull* Schematic window.

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_example*.

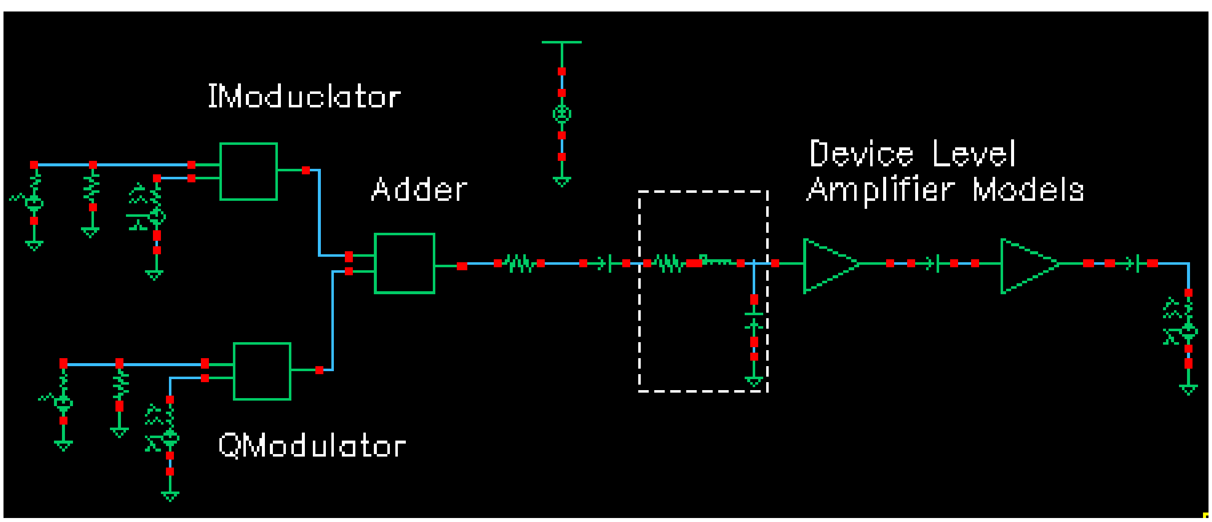
   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic as shown in Figure 8-6. This is a simple direct-conversion transmitter with ideal I/Q modulators.

### The EF_example Schematic




3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

4. Following the prompt,

   `> point at object to copy`

   left click and drag to create a box around the entire *EF_example* circuit.

   The *EF_example* components are highlighted in yellow.

5. Following the prompt,

   `> point at reference point for copy`

   click inside the outlined elements.

6. Following the prompt,

   `> point at destination point for copy`

   move the cursor to the empty *EF_LoadPull* Schematic window and click there. This drags a copy of the *EF_example* circuit into the empty Schematic window.

   The *EF_LoadPull* Schematic window now contains a copy of the *EF_example* schematic.

7. If necessary, choose *Window - Fit* to center the *EF_LoadPull* circuit in the Schematic window.

8. In the *EF_example* Schematic window, choose *Window - Close*.

   The *EF_example* Schematic window closes.

### Opening the Simulation Window for the EF_LoadPull Schematic

➤ Open the ADE window from the *EF_LoadPull* Schematic window as described for the *EF_example* schematic in "Opening the Simulation Window" on page 523.

### Setting up the Model Libraries for the EF_LoadPull Schematic

➤ Set up the Model Libraries for the *EF_LoadPull* Schematic as described for the *EF_example* schematic in "Setting Up the Model Libraries" on page 524.

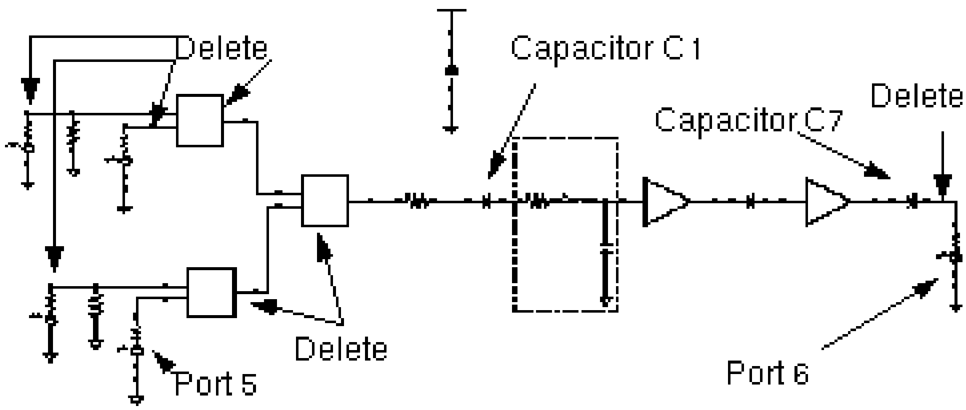
### Editing the EF_LoadPull Schematic

To prepare the schematic, follow the instructions in the following sections.

### *Delete Components and Wires from the EF_LoadPull Schematic*

In the *EF_LoadPull* Schematic window, delete components and their associated connecting wires as shown in Figure 8-7.

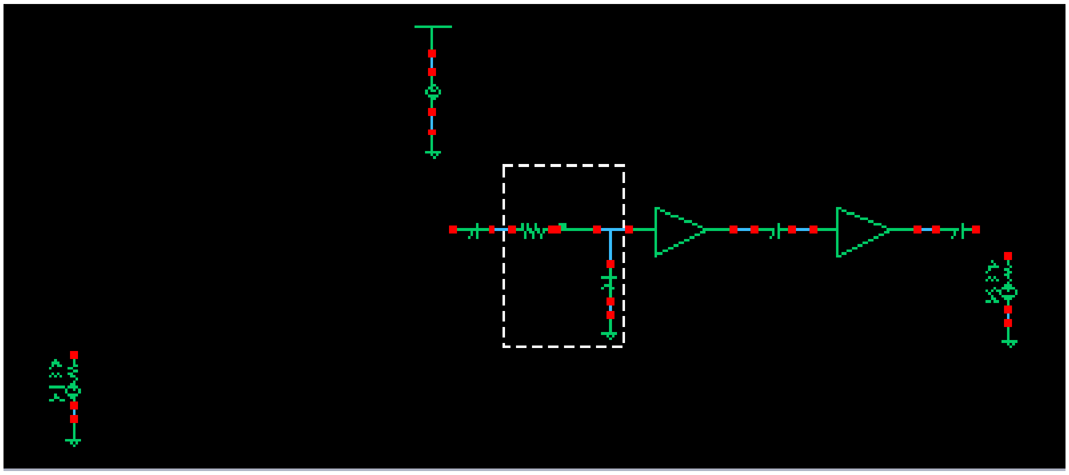
**Figure 8-7 Components and Wires to Delete for the EF_LoadPull Schematic**




If you need assistance with methods for editing the schematic, see the *Virtuoso® Schematic Composer™ User Guide*.

1. In the *EF_LoadPull* Schematic window, choose *Edit – Delete.*
2. Click a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. After deleting the indicated components and wires, the *EF_LoadPull* Schematic should look like Figure 8-8 on page 591.

**Figure 8-8  EF_LoadPull Schematic After Deleting Components**



5. In the *EF_LoadPull* Schematic window, choose *Edit – Move* to move *PORT5* closer to capacitor *C1* and to move *PORT6* away from capacitor *C7* to make room for the *PortAdaptor* component you add later.

    a. Left click and drag to create a rectangle around *PORT5* and its *GND*.

        *PORT5* and the *Gnd* are highlighted in yellow.

    b. Click inside the yellow line and drag the outlined components to move them close to capacitor *C1* as shown in Figure 8-9 on page 593.

    c. Left click and drag to create a rectangle around *PORT6* and its *GND*.

        *PORT6* and the *Gnd* are highlighted in yellow.

    d. Click inside the yellow line and drag the components to move them away from capacitor *C7* as shown in Figure 8-9 on page 593.

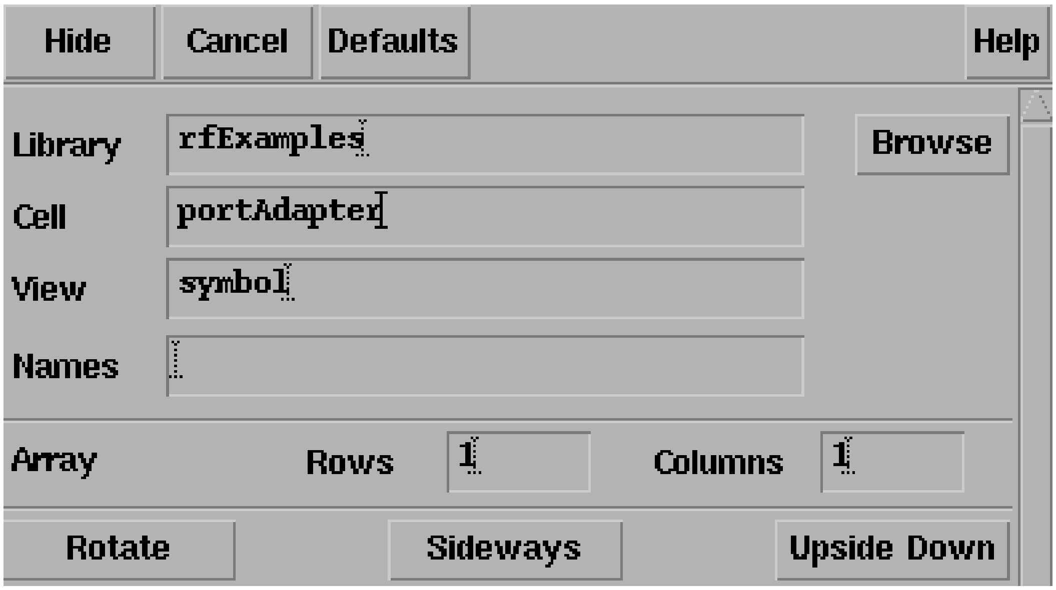
        This makes room for the *PortAdapter*.

6. Choose *Window - Fit* to center the edited schematic in the window.

### *Place the PortAdaptor*

1. In the *EF_LoadPull* Schematic window, choose *Add – Instance*.

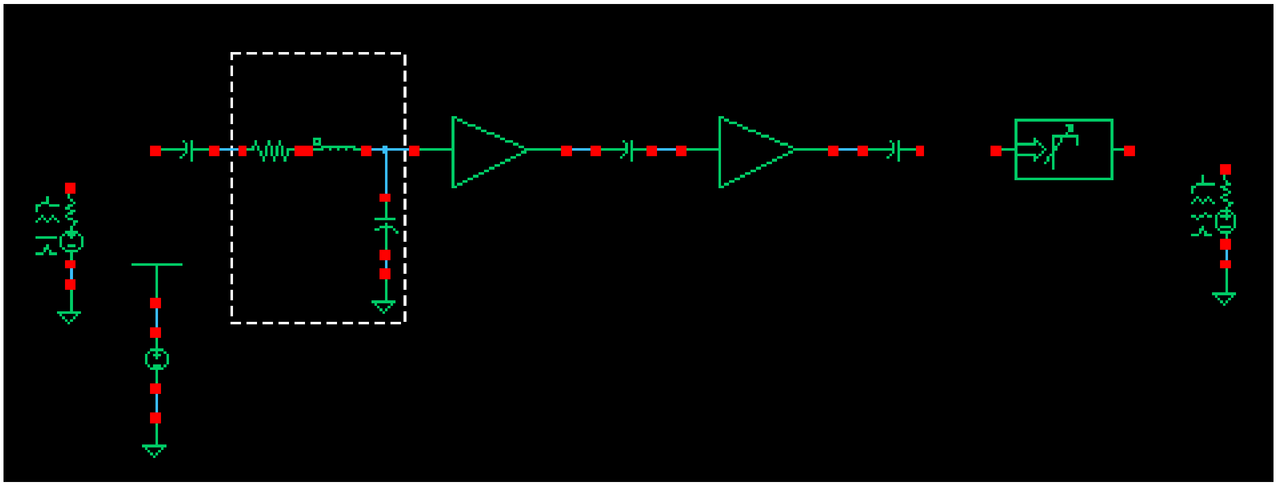
The Add Instance form appears.




2. In the Add Instance form *Library* field, type `rfExamples`.
3. In the *Cell* field, type `portAdapter`.
4. In the *View* field, type `Symbol`.

   As you move your cursor from the form to the *EF_LoadPull* Schematic window, a copy of the *portAdapter* moves with the cursor.
5. Left click to place the *portAdapter* as shown in Figure 8-9 on page 593.
6. After you place the *portAdapter* in the schematic, press the *Esc* key to remove the *portAdapter* symbol from your cursor and close the Add Instance form.

**Figure 8-9  The EF_LoadPull Schematic**



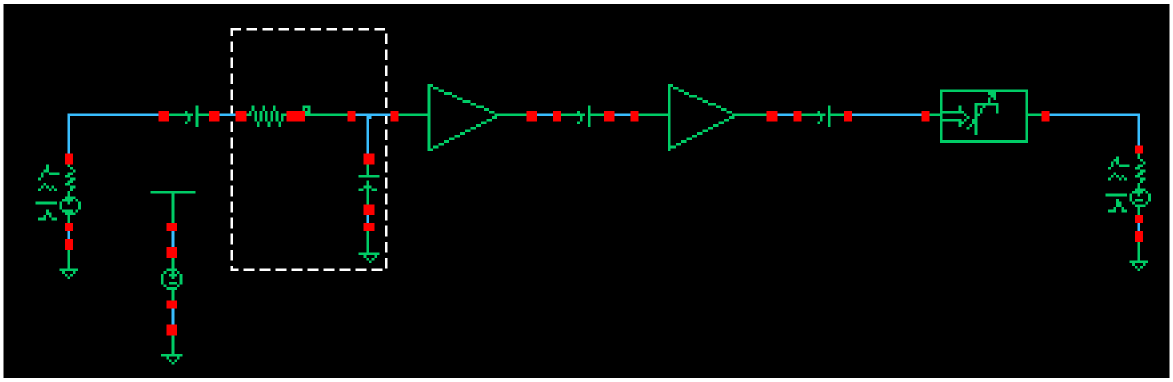
### *Wire the Schematic*

In the *EF_LoadPull* Schematic window, you need to connect *PORT5* to Capacitor *C1* and connect the *portAdapter* between Capacitor *C7* and *PORT6*. To do so,

1. In the Schematic window, choose *Add - Wire (Narrow)*.
2. Click the terminal on *PORT5* then click the terminal on *C1*.
3. Click the terminal on *C7* then click the terminal on the *PortAdaptor.*
4. Click the terminal on the *PortAdaptor* then click the terminal on *PORT6.*
5. Press the Esc key to stop wiring.

   The edited schematic looks like the one in Figure 8-9.

### The EF_LoadPull Schematic



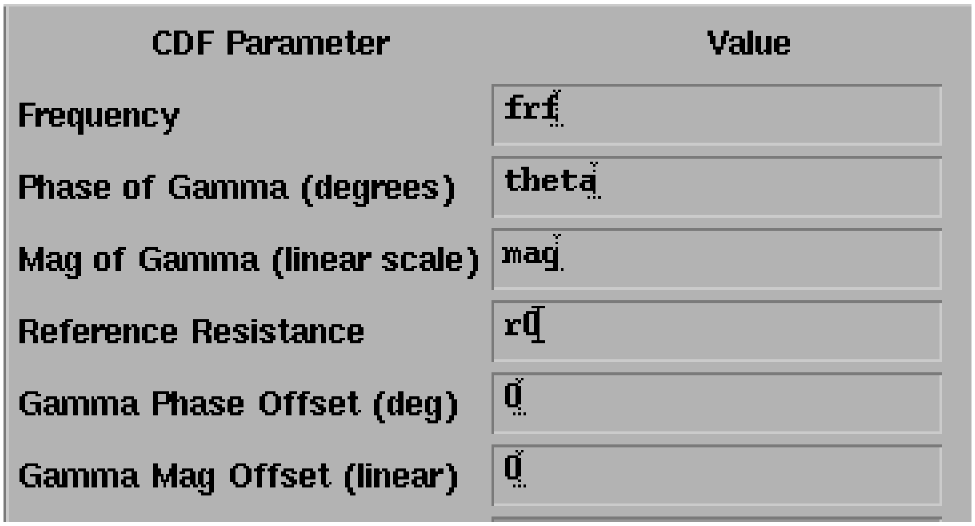
#### *Edit CDF Properties for both the PortAdaptor and Port6*

In the *EF_LoadPull* Schematic, the *portAdapter* and its terminating port, *Port6*, must have the same reference resistance.

1. In the *EF_LoadPull* Schematic window, select the *PortAdapter*.
2. In the *EF_LoadPull* Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears with information for the *Port Adapter* displayed.
3. In the Edit Object Properties form, do the following and click *Apply*.
   a. Type *frf* for *Frequency*.
   b. Type *theta* for *Phase of Gamma (degrees)*.
   c. Type *mag* for *Mag of Gamma (linear scale)*.
   d. Type *r0* for *Reference Resistance*.

The completed form looks like this.



| CDF Parameter | Value |
|---|---|
| Frequency | frf |
| Phase of Gamma (degrees) | theta |
| Mag of Gamma (linear scale) | mag |
| Reference Resistance | r0 |
| Gamma Phase Offset (deg) | 0 |
| Gamma Mag Offset (linear) | 0 |

4. In the *EF_LoadPull* Schematic window, select *Port6*.

   The Edit Object Properties form changes to display data for *Port6*.

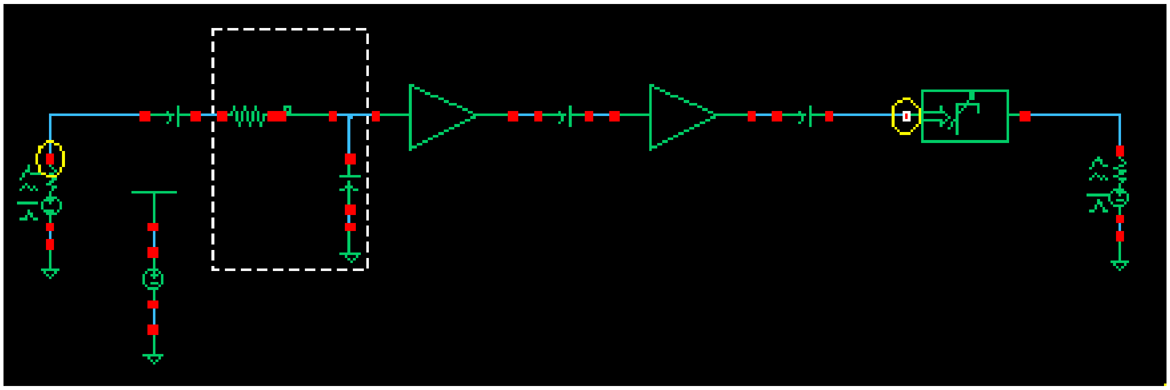
5. In the Edit Object Properties form for *Port6*, type *r0* for *Resistance* and click *OK*.

### *Select Outputs To Save*

1. In the ADE window for the *EF_LoadPull* Schematic, choose *Outputs – To Be Saved – Select on Schematic*.

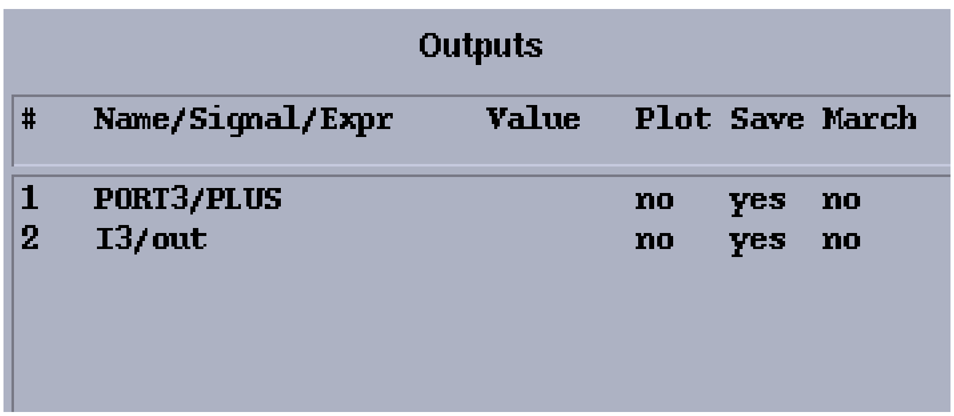
2. In the *EF_LoadPull* Schematic window, click each terminals that is circled in Figure 8-10.

**Figure 8-10 Outputs to Save in the EF_LoadPull Schematic**



After you click a terminal, it is circled in the schematic as shown in Figure 8-10. The selected outputs are also displayed in the ADE window Outputs area as shown in Figure 8-11.

**Figure 8-11 Outputs Area in the Simulation Window**



Outputs

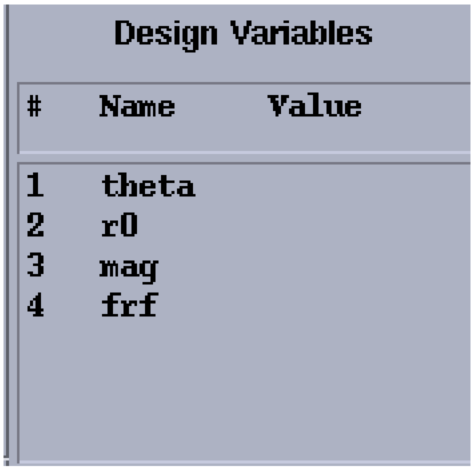
| # | Name/Signal/Expr | Value | Plot | Save | March |
|---|---|---|---|---|---|
| 1 | PORT3/PLUS | | no | yes | no |
| 2 | I3/out | | no | yes | no |

### *Editing Variables*

1. In the ADE window, choose *Variables – Copy from Cellview*.

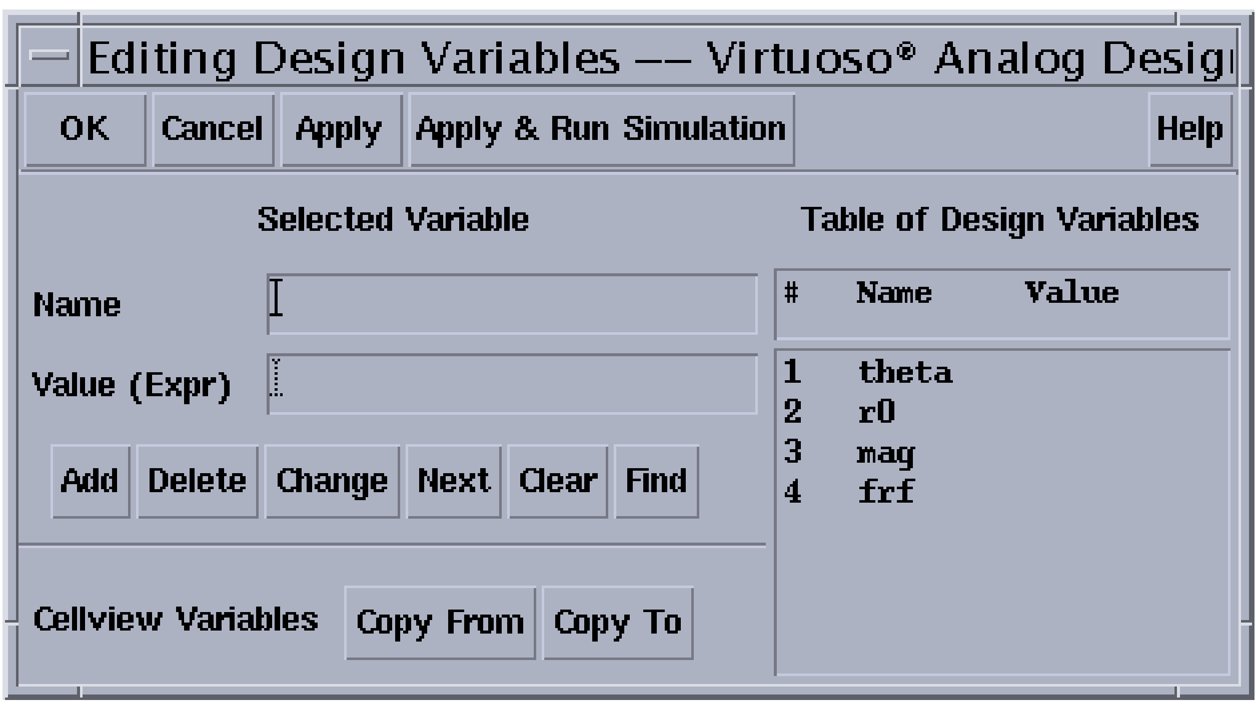
   The *Design Variables* list box in the ADE window changes to reflect the copied variables from the schematic as shown in Figure 8-12.

**Figure 8-12 Design Variables Area Showing Variable Names**




2. In the ADE window, choose *Variables - Edit*.

   The Editing Design variables form appears.




3. In the Editing Design Variables form, do the following.

   a. Highlight one of the variables In the *Table of Design Variables* list box.

      The variable name displays in the *Name* field.

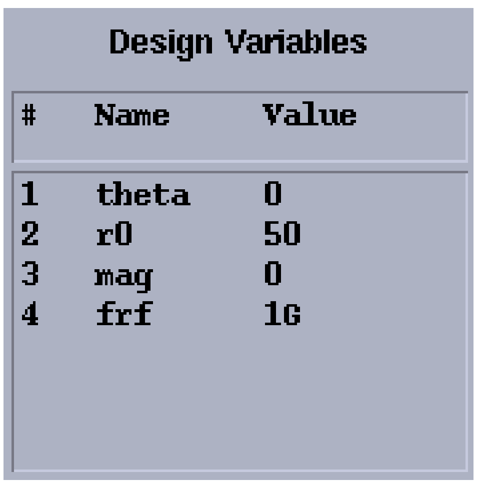
**b.** Type the variable's associated value from the following list into the *Value (Expr)* field.

- `theta 0`
- `r0 50`
- `mag 0`
- `frf 1G`

**c.** Click *Change*.

**d.** When you have given values to all four variables, click *OK*.

The Design Variables list box in the ADE window now displays both variable names and the value associated with each name.




#### *Save the Changes to the EF_LoadPull Schematic*

➤ In the Schematic window, choose *Design – Check and Save* to save the current state of the *EF_LoadPull* schematic.

## Setting Up and Running the PSS and Parametric Analyses

This example tells you how to

- Set up the swept PSS analysis to sweep the design variable `theta`.
- Set up the Parametric analysis to sweep the design variable `mag`.
- Run the parametric and swept PSS analyses.

■ Plot the results.

### Setting Up the PSS Simulation

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

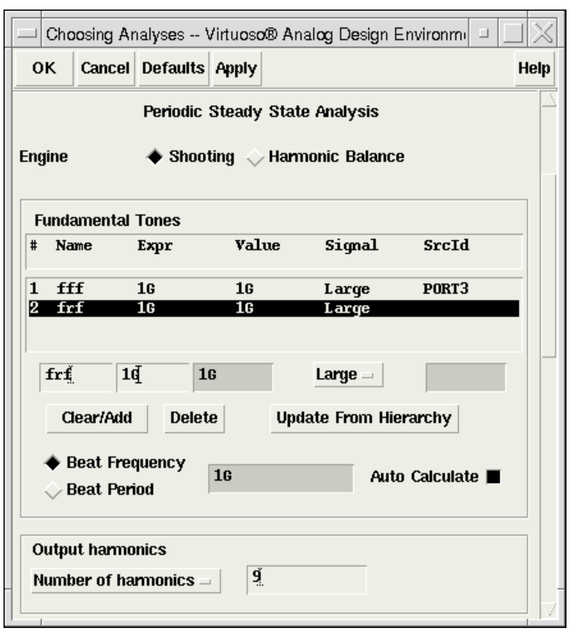
3. In the Choosing Analyses form, if information for *frf* is not displayed in the *Fundamental Tones* list box, type `frf` in the *Name* field below the list box, type `1G` in the *Expr* field below the list box and click *Clear/Add*.

4. Click *Update From Hierarchy* to update the values in the *Fundamental Tones* list box by searching all levels of the hierarchy. If you do not click *Update From Hierarchy*, only the top level of the hierarchy is searched.

5. Click *Auto Calculate* to automatically calculate and enter a value in the *Beat Frequency* field. 1G displays in the *Beat Frequency* field.

6. Choose *Number of harmonics* for *Output harmonics* and type 9 in the adjacent field.

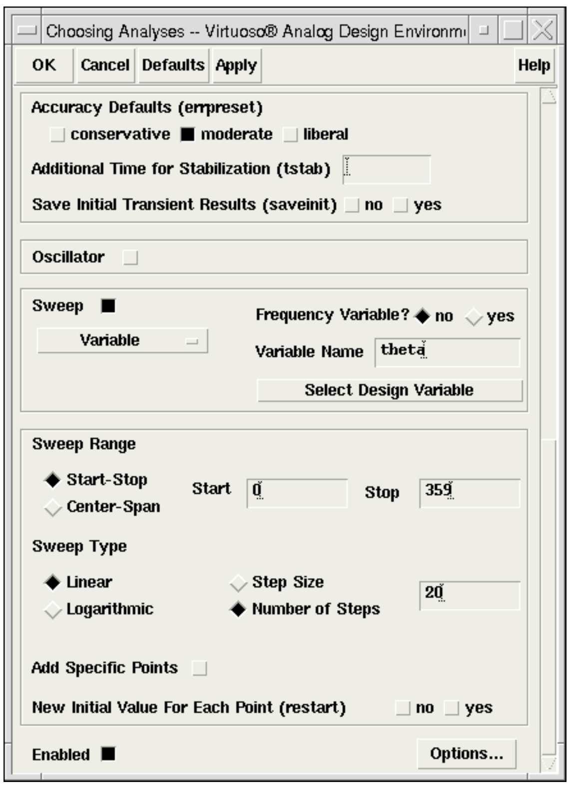
   The top of the PSS analysis form looks like this.




7. Highlight *moderate* for *Accuracy Defaults (errpreset).*

8. Highlight the *Sweep* button and choose *Variable* in the associated cyclic field.
9. Type `theta` for *Variable Name* (or click *Select Design Variable*, highlight *theta* in the form that appears and click *OK*).
10. Choose *Start – Stop* for *Sweep Range*.
11. Type `0` for the *Start* value and `359` for the *Stop* value.
12. Choose *Linear* for *Sweep Type*.
13. Highlight *Number of Steps* and type `20` in the adjacent field.
14. Verify that *Enabled* is highlighted.

    The bottom of the PSS analysis form looks like this.




   a. Click *Apply*. Then click *OK*.

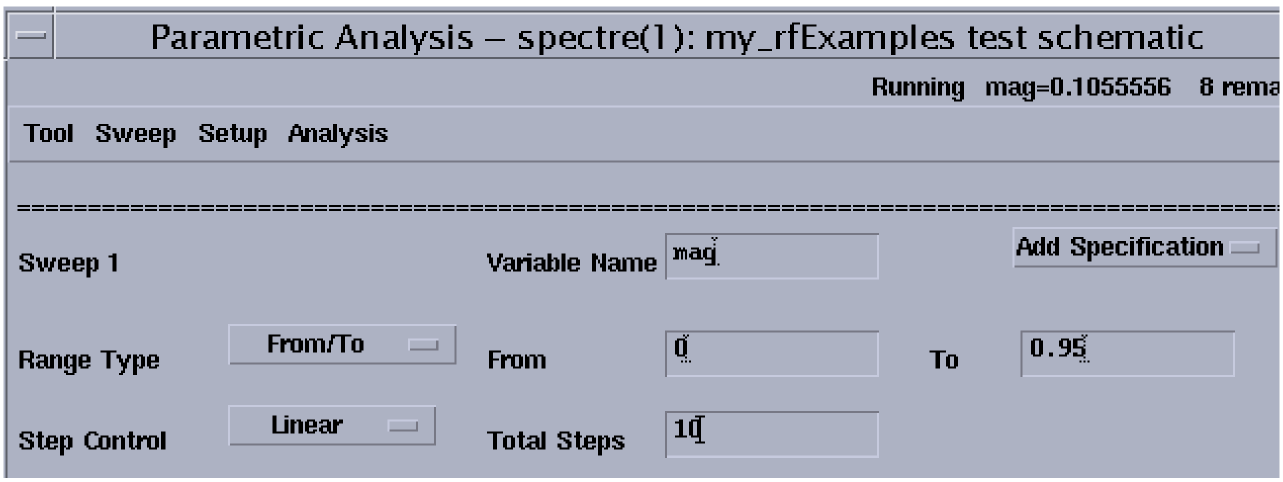
### Setting Up the Parametric Analysis and Performing the Analyses

1. In the ADE window, choose *Tools – Parametric Analysis*.

   The Parametric Analysis form appears.

2. In the Parametric Analysis form, type *mag* for *Variable Name*.
3. Choose *From/To* in the *Range Type* cyclic field.
4. Type `0` for *From* and `0.95` for *To* in the adjacent fields.
5. Choose *Linear* in the *Step Control* cyclic field.
6. Type `10` in the adjacent *Total Steps* field.

   The completed form looks like this.



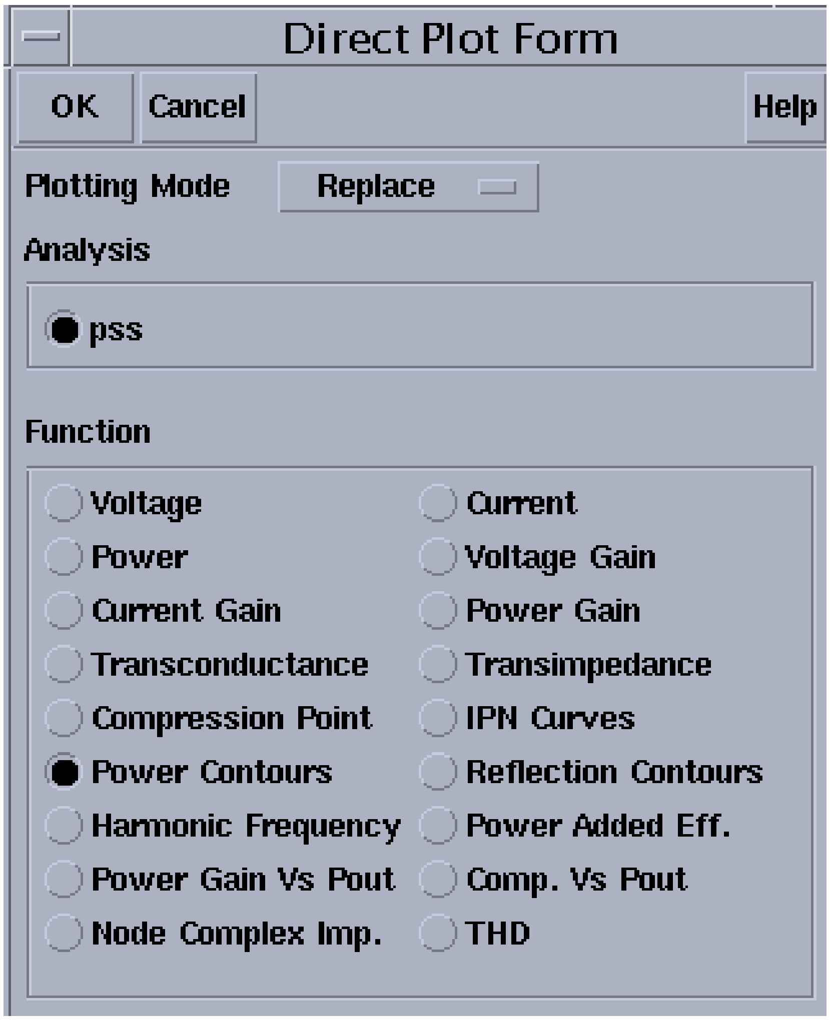

7. In the Parametric Analysis form, choose *Analysis – Start*.
8. Look in the CIW for a message that says the simulation has completed successfully.

## Displaying Load Contours

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pss* for *Analysis*.
4. Highlight *Power Contours* for *Function*.

The top of the Direct Plot form looks like the following.




**5.** Highlight *Magnitude* for *Power Modifier*.

**6.** Leave *Maximum Power* and *Minimum Power* blank.

**7.** Type `9` for *Number of Contours*, if necessary.

**8.** Type `50.0` for *Reference Resistance*, if necessary.

**9.** Following the prompt at the bottom of the form,

`> Select Output Harmonic on this form...,`

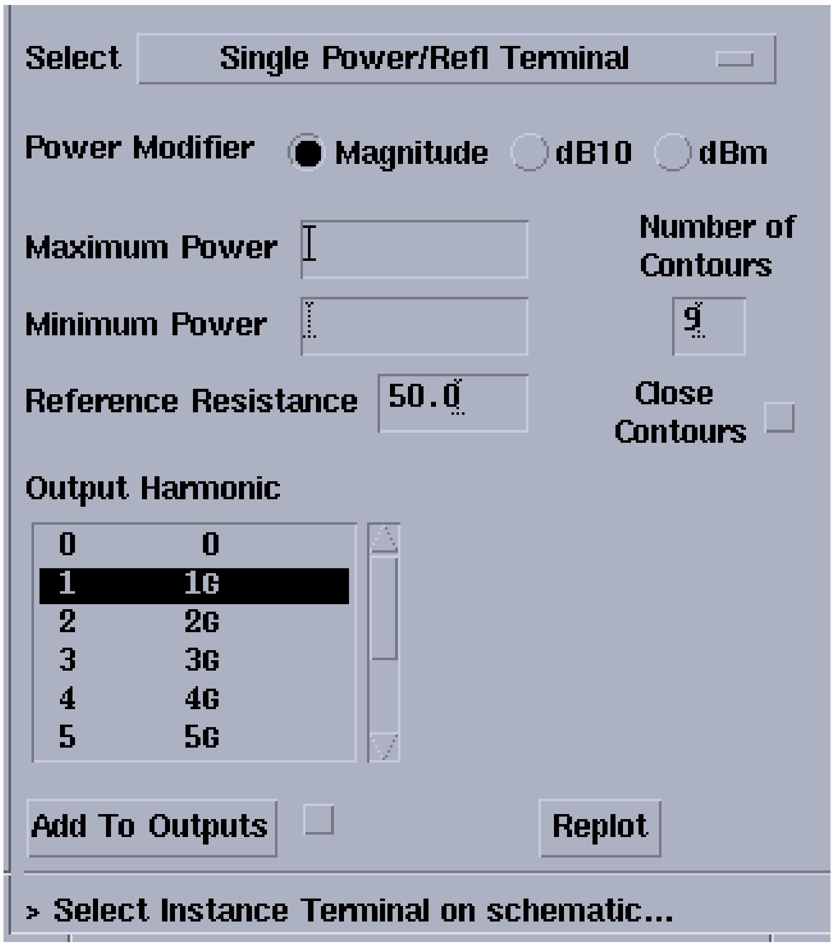
highlight *1 1G* for *Output Harmonic*.

You would select a different harmonic if the PSS fundamental frequency were smaller than `1 GHz`. For example, if another part of the circuit is driven at `1.5 Hz`, the fundamental is `500 MHz`. But because the part of the circuit where you want to do load-pull analysis operates at `1 GHz`, in order to plot the correct contours, the contours associated with `1 GHz`, you must specify the second harmonic of the fundamental, `2*500 MHz = 1 GHz`.

The *Select* cyclic field displays *Single Power/Refl Terminal* and the prompt at the bottom of the form displays

`> Select Instance Terminal on Schematic.`

The bottom of the Direct Plot form looks like the following.

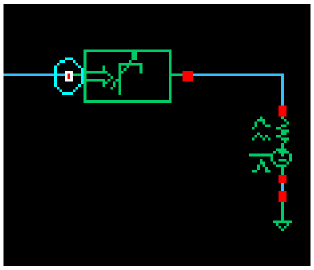



**10.** Follow the prompt at the bottom of the Direct Plot form,

`> Select Instance Terminal on Schematic`

by clicking, in the schematic, the terminal at the port adapter as shown in Figure 8-13.

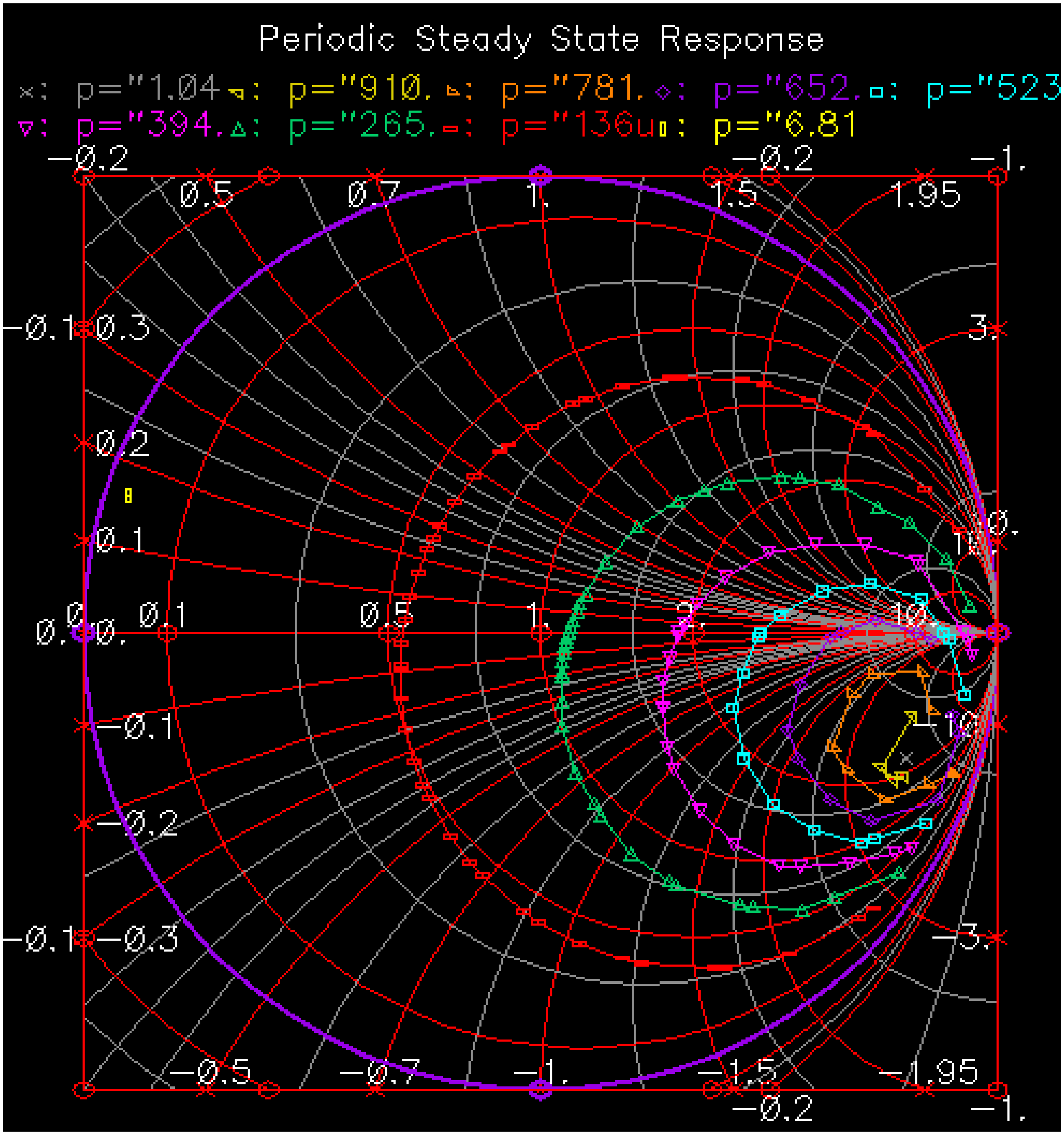
**Figure 8-13 Terminal at Port Adapter**



**Note:** Subsequent instructions call this terminal the *terminal at the port adapter*.

After you select the terminal, the plot for the load contours appears in the waveform window. Figure 8-13 shows the entire plot.

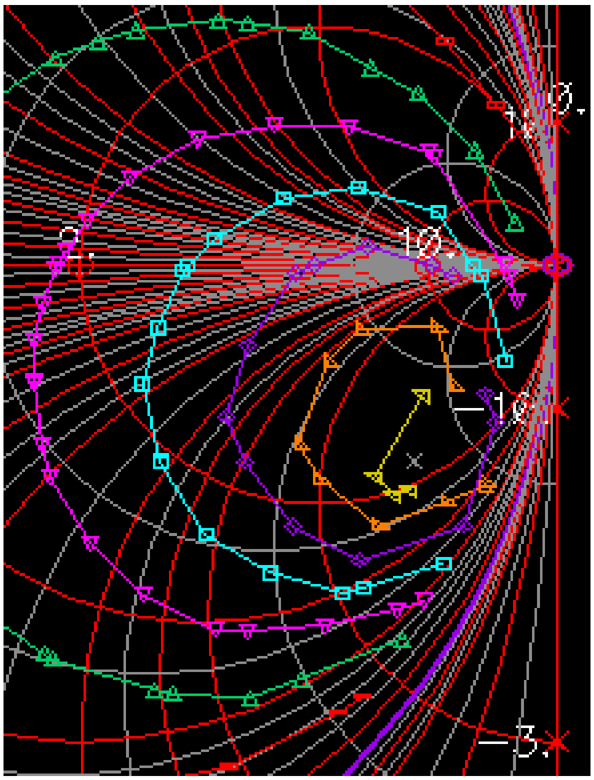
### Complete Load Contour Plot




A small X appears at the maximum power point, which in this case lies near the center of the smallest (yellow) constant power contour. See Figure 8-14 on page 606 for a detailed view.

Figure 8-14, an enlarged area taken from Figure 8-13, which more clearly shows the X that marks the maximum power point (inside the yellow contour).

**Figure 8-14 Enlarged View Showing Max Power Point (X)**




If you place the cursor on the X, you can read the following information across the top of the waveform window.

```
Real:2.479      Imag:-4.737      Freq:-360     p="1.04m";Constant Power Contours
```

This indicates that a normalized load impedance of about `2.48 - j4.737` dissipates the most power. The X appears at the maximum power point.

### Adding the Reflection Contours to the Plot

You might want to maximize load power subject to a constraint on the magnitude of the amplifier's input reflection coefficient. Such a constraint can prevent unstable interactions with the preceding stage.

You can overlay load-pull contours with contours of constant input reflection coefficient magnitude. The optimal load corresponds to the reflection coefficient that lies on both the *largest power* load-pull contour and on a constant input reflection coefficient contour that is within the constraint. Here *largest power* means the contour corresponding to the largest amount of power delivered to the load.

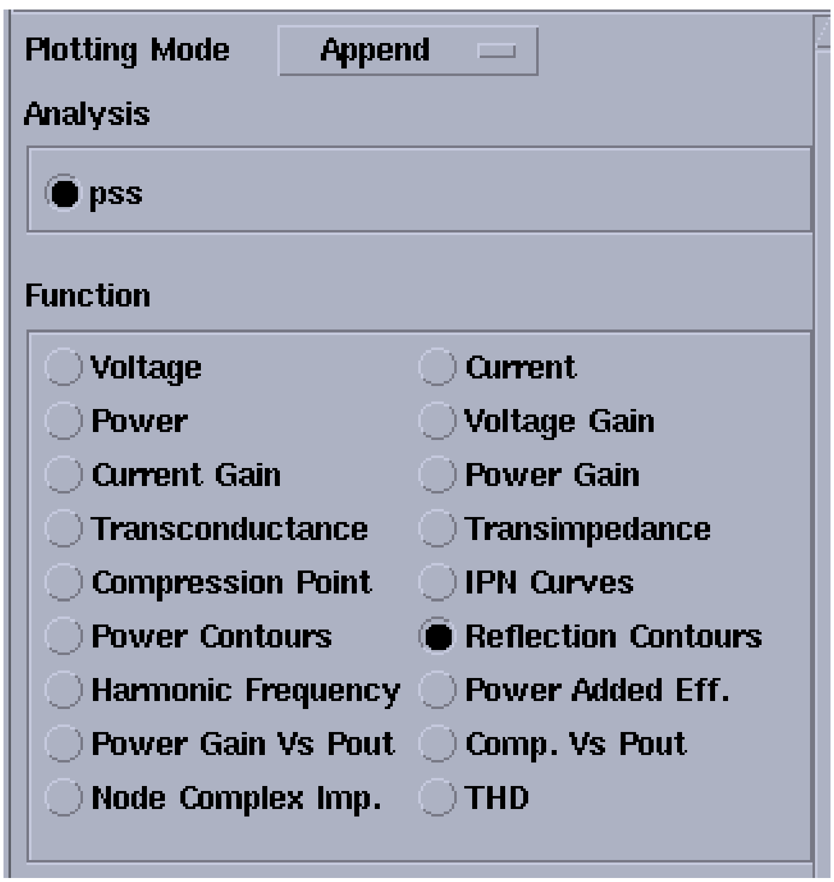
To use the topographical map analogy again, this is like overlaying constant elevation contours with constant temperature contours. The optimum objective is similar to trying to find the highest point on the mountain where the temperature is above 60 degrees.

1. If necessary, in the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *Append* in the *Plotting Mode* cyclic field.
3. Highlight *pss* for *Analysis*.
4. Highlight *Reflection Contours* for *Function*.

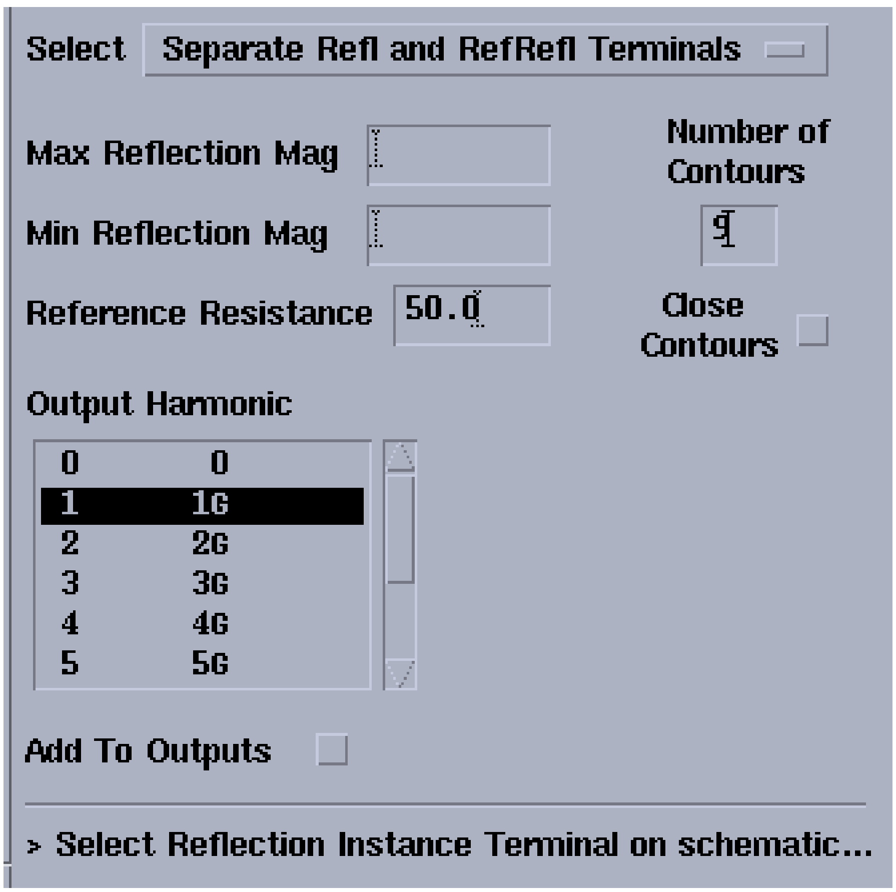
   The top of the completed form looks like this.



5. Type `9` for *Number of Contours*, if necessary.
6. Type `50.0` for *Reference Resistance*, if necessary.
7. Highlight *1 1G* for *Output Harmonic*.
8. In the *Select* cyclic field, select *Separate Refl and RefRefl Terminals*. The prompt at the bottom of the form changes to

```
> Select Reflection Instance Terminal on Schematic
```

The bottom of the completed form looks like this.

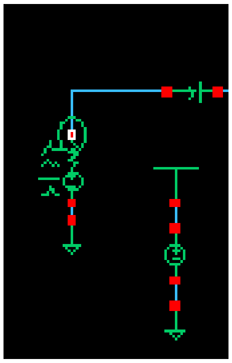



9. Following the prompt at the bottom of the Direct Plot form,

```
> Select Reflection Instance Terminal on Schematic
```

in the Schematic window, click the terminal at port 5, as shown in Figure 8-15.

**Figure 8-15  Terminal at Port 5**



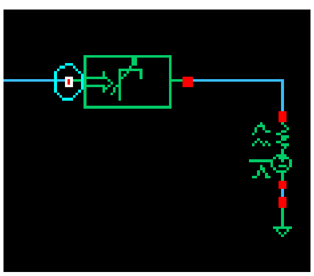
**Note:** Subsequent instructions call this terminal the *terminal at Port 5*.

The prompt on the bottom of the Direct Plot form is

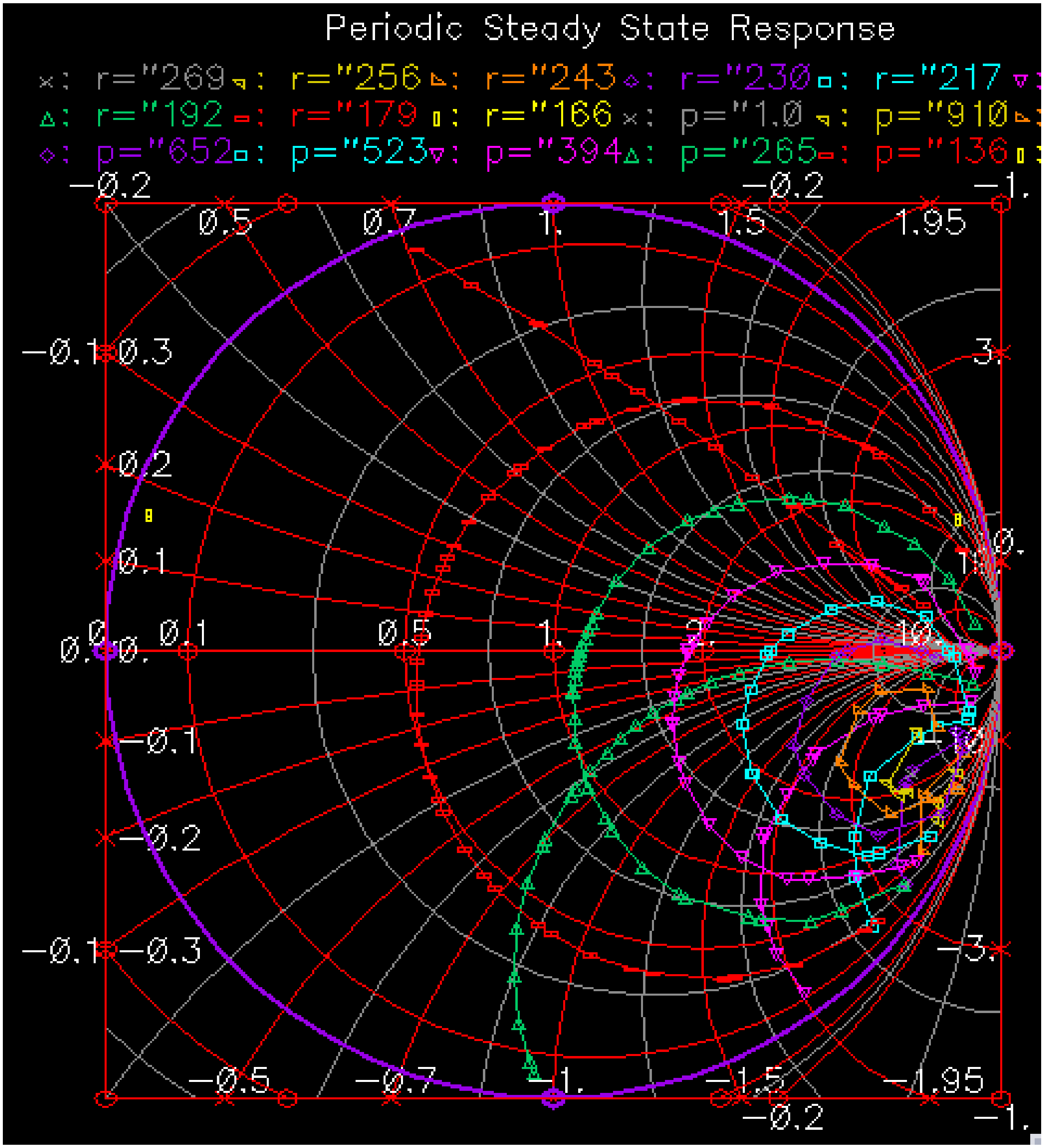
```
> Select Reflection Instance Terminal on Schematic....
```

Follow the prompt. In the Schematic window, click the *terminal at the port adapter* (See 8-16).

**Figure 8-16  Terminal at Port Adapter**



The plot for reflection contours is added to the waveform window.




Assume that, for stability, the input reflection coefficient should be less than or equal to 0.218. For the curves shown above, the optimal normalized load impedance is approximately

3.908 -j5.073. Assuming you were free to draw smaller constant power contours, the true constrained optimal power point would occur when the constant power contour is just tangent to the constant input reflection coefficient contour of 0.218.

In the Direct Plot Form for *Reflection Contours*, *Separate Refl & RefRefl Terminals* is selected to measure *Reflection Contours* as contours of constant reflection (magnitude) as seen from the power amplifier's input (measured at *port 5,* the input port) with respect to the reflection of the power amplifier's load (measured at the input to the port adapter.)

In this case,

- In response to the first prompt,

  ```
  > Select Reflection Instance Terminal on Schematic
  ```

  Select the input to the power amplifier, the terminal of *port 5* in the schematic. (See Figure 8-15 on page 609)

- In response to the second prompt

  ```
  > Select Reflection Instance Terminal on Schematic
  ```

  Select the output of the power amplifier, the input terminal to the *port adapter* in the schematic. (See Figure 8-16 on page 609)

So the *port adapter's* reflection (both `mag` and `phase`) is swept, which is the same as sweeping the power amplifier's load. In the Smith chart you see input reflection plots as seen by *port 5*.

The reflection contour plot shows the effect of the power amplifier's load on the input reflection while the power contour plot shows the effect of the power amplifiers's load on the power gain.

For reflection contours, the following terminal selections apply.

1. For *Single Refl/RefRefl Terminal* select one terminal.

   One reflection coefficient of that terminal is computed as

   $$gamma1 = \frac{Z - Z0}{Z + Z0}$$

   where *Z* is the large signal impedance at the fundamental.

   The resulting Smith chart plots contours where `gamma1` is constant.

2. For *Single Refl/RefRefl Term and ref Term* select two terminals.

   This is the differential case for number 1. The different voltages of the two nets are used in the large signal impedance calculation.

3. For *Separate Refl and RefRefl Terminals* select two terminals

   Two reflection coefficients for each terminal are computed as

$$gamma1 = \frac{Z1 - Z0}{Z1 + Z0}$$

$$gamma2 = \frac{Z2 - Z0}{Z2 + Z0}$$

   where *Z1* and *Z2* are the large signal impedance for each terminal at the fundamental.

   The resulting Smith chart plots gamma2 and contours where `gamma1` is constant.

4. For *+ - Refl and + - Ref Refl Terminals* select 4 terminals

   This is the differential case for number 3.

For power contours, the following terminal selections apply.

1. For *Single Power/Ref Terminals* select one terminal.

   The power (*P1*) and the reflection coefficient (`gamma1`) of that terminal are computed. The resulting Smith chart plots `gamma1`and contours where `P1` is constant.

2. For *Single Power/Refl Term and ref Term*

   This is the differential case for number 1.

3. For *Separate Power and Refl Terminals* select 2 terminals

   The power of the first terminal (*P1*) and the reflection coefficients of the second terminal (`gamma2`) are computed. The resulting Smith chart plots `gamma2` and contours where *P1* is constant.

4. For *+ - Power and + - Refl Terminals*

   This is the differential case for the third case, *Separate Refl and RefRefl Terminals*.

### Are Constant Power Contours the Same as Constant Gain Contours?

In general, *constant power contours* are not the same as *constant power gain contours*. The simulator does not maintain any input impedance match while it sweeps the load to generate load-pull contours. The resulting contours are not constant power gain contours, but are only constant power contours. However, if the input reflection coefficient and input power do not change much over the load sweep range, the generated contours are good approximations of constant gain contours. By plotting constant power contours in the input reflection coefficient plane, you can see how both change when you sweep load reflection coefficient. If the input power does not vary with load, the constant power contours are also constant power gain contours. If the input power does vary with the load, the constant power contours tell you how much you have to change the input matching network during the load sweep to maintain a constant input power.

1. In the Direct Plot form, do the following:

    a. Highlight *Replace* for *Plot Mode*.

    b. Highlight *Power Contours for Function*.

    c. Highlight *Single Power/Ref Terminal*.

    d. Highlight *Magnitude* for *Power Modifier*.

    e. Type `9` for *Number of Contours*, if necessary.

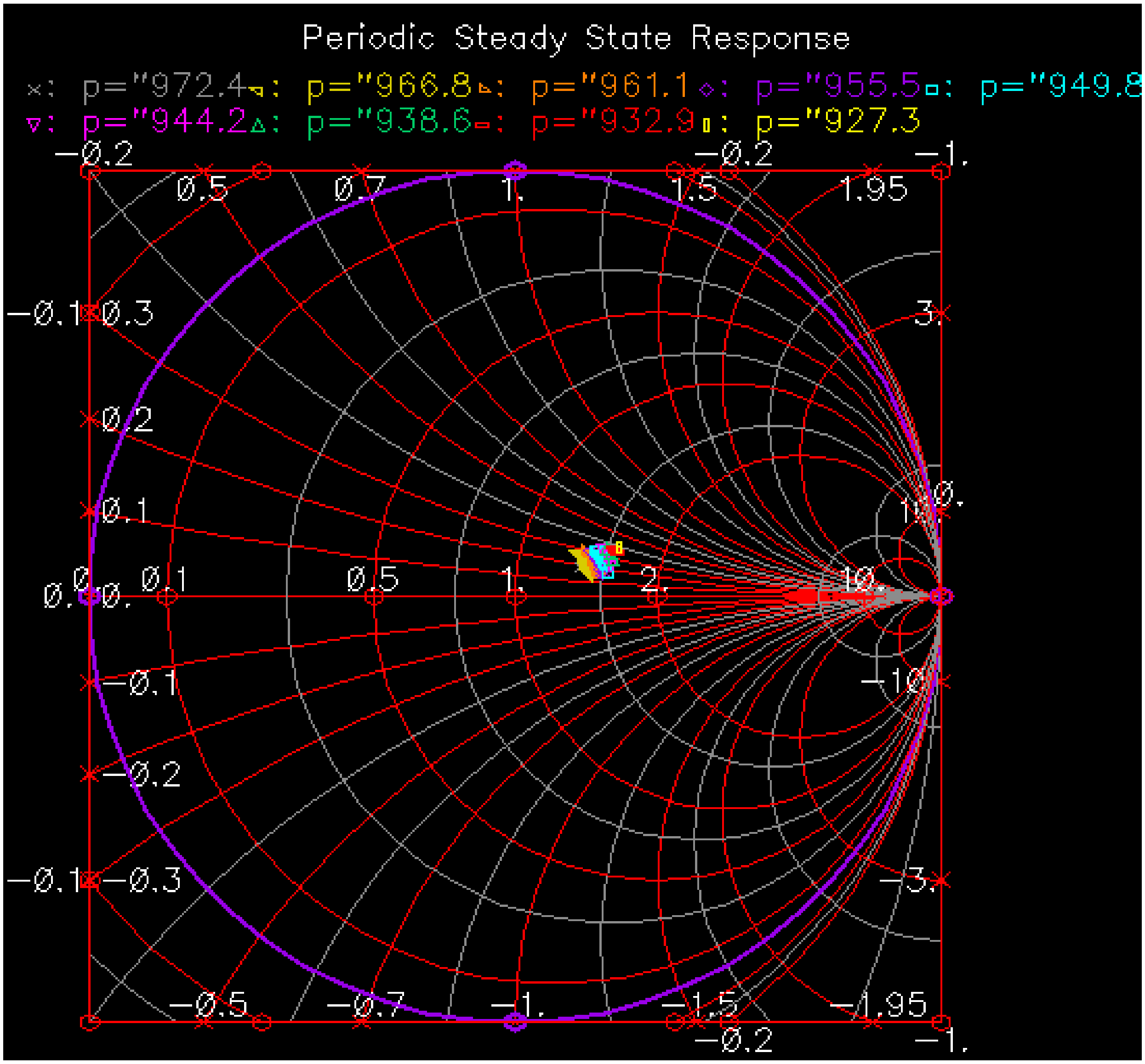
    f. Type `50.0` for *Reference Resistance*, if necessary.

    g. Highlight 1 G for *Output Harmonic*.

2. In the Schematic window, click the terminal at Port 5 (See Figure 8-15 on page 609).

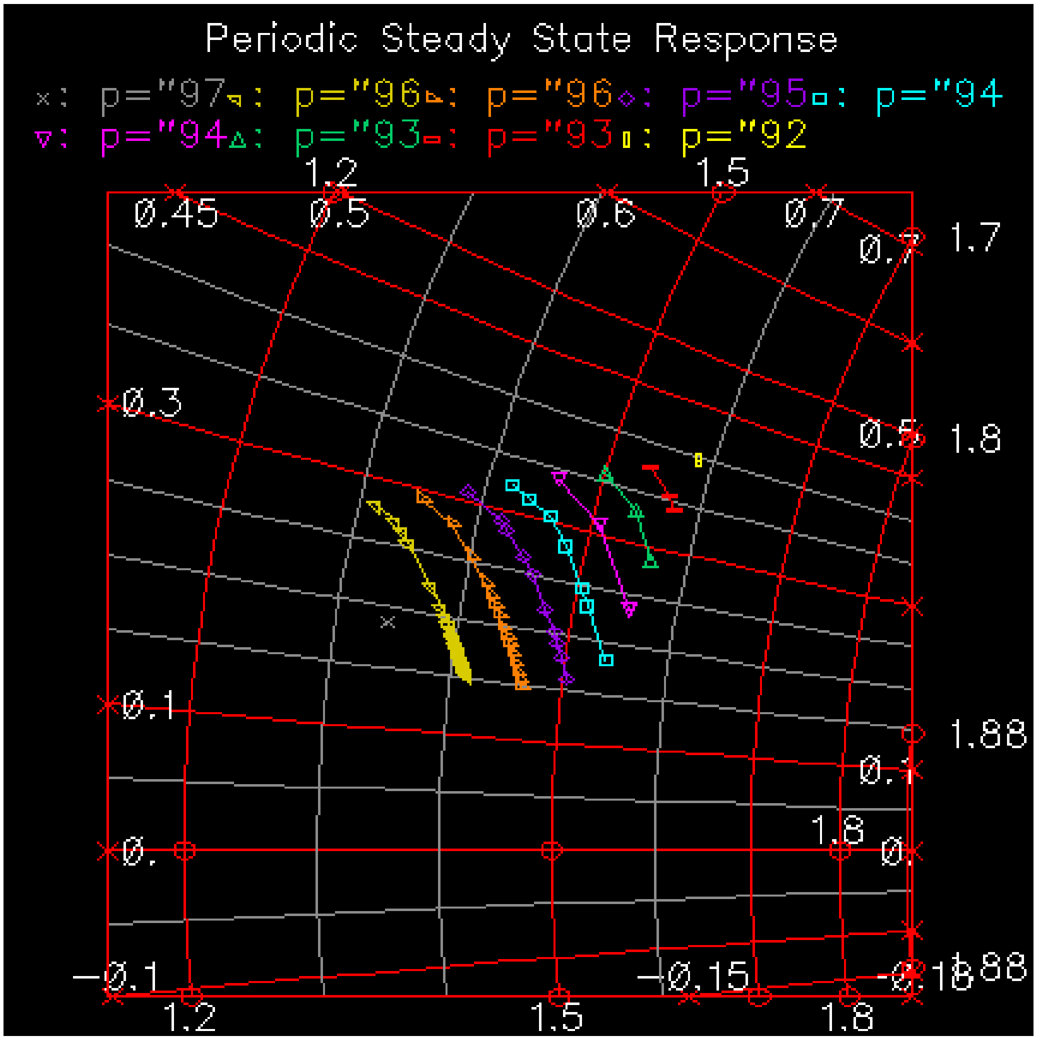
3. The plot of input reflection coefficients appears as in Figure Figure 8-17 on page 614.

**Figure 8-17 Input Reflection Coefficients Generated by Sweeping Load**




4. In the waveform window, choose *Zoom – ZoomIn* and then click and drag with the mouse to form a rectangle that includes the area you want to see.The magnified view is shown in Figure 8-17.

### Magnified View of Input Reflection Coefficients



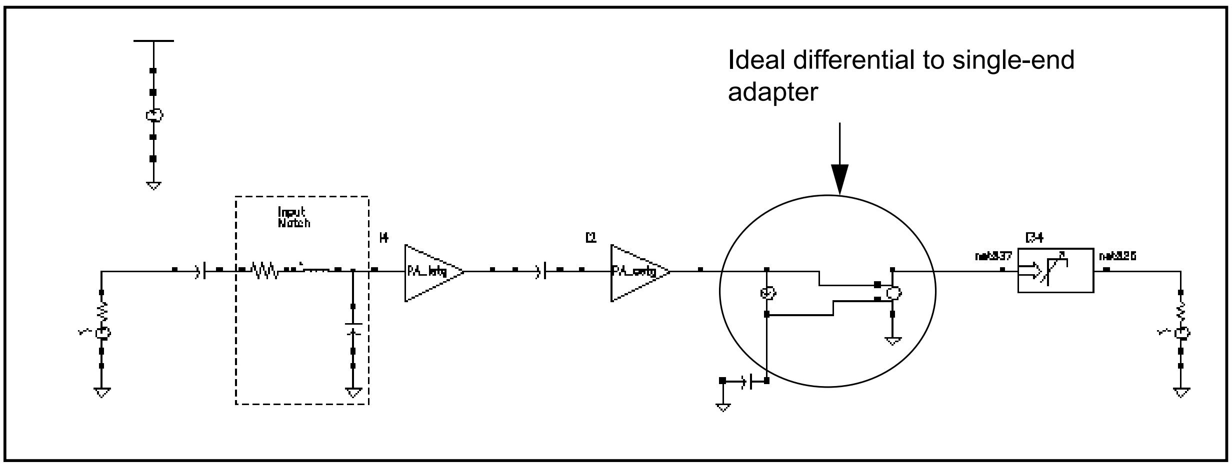

The real part of the input impedance varies from about 1.3 to 1.6 in normalized units and the imaginary part varies from 0.15 to 0.4 in normalized units. The input power varies from 927 nW to 967 nW. You find these numbers by placing the cursor on the end points of the outside contours. You read the impedance and power/reflection for that cursor location at the top of the window.

The numbers vary slightly as the load is swept. The objective here is to determine how much the input reflection coefficient and the input power vary as the load varies. If the input quantities vary significantly, then the generated load pull contours do not coincide with the constant power gain contours because the input power does not remain constant. The amount of variation in the input reflection coefficient tells you by how much you would have to re-tune the input matching network to maintain constant input power as you sweep the load. You have to decide whether this much variation matters.

### Moving to Differential Mode

Up until now, the examples used in this chapter have been single-ended. Because many circuits are differential, Figure 8-18 shows a simple way to modify the example circuit to make it differential.

#### Figure 8-18 Simple Differential Circuit




If your circuit has one or more differential ports, you can translate between single-ended and differential circuitry using linear-dependent controlled sources. The extra circuitry added to Figure 8-18 shows a modified version of the circuit where the output has been modified to make it differential. The transition from single-ended to differential output was made using the controlled sources *F0* and *E1* (circled in Figure 8-18).

The controlled sources, *cccs* and *vcvs*, are available in *analogLib*. The gain of *cccs* is -1 to be consistent with the polarity of the *vcvs*. The *vcvs* (*E1*) is the *Name of voltage source* parameter in the *cccs*.

## Using S-Parameter Input Files

In this example, you create an S-parameter data file then you enter it back into the original circuit. The S-parameter data file replaces that portion of the circuit originally used to create the S-parameters. To enter tabulated S-parameters from some other source, record the data in the format shown in the S-parameter data file and select the appropriate numerical options in the *nport*.

This example shows you how to

- Set up and simulate the *sparamfirst* circuit and create an S-parameter data file (`sparam.practice`) for the *sparamfirst* schematic
- Add an *n2port* device in the frequency domain to a modified copy of the *sparamfirst* schematic.
- Run another simulation of the modified *sparamfirst* circuit where the *n2port* component reads in the S-parameter data file
- Plot and compare the two sets of data produced.
  - The output of the original branch of the circuit
  - The output of the modified branch of the circuit where the *n2port* component replaced the components in the original branch. The *n2port* reads in the `sparam.practice` S-parameter file you created.

  When you superimpose the two plots, it is clear that the S-parameter file produces the same plot as the original components.

## Setting Up the EF_example Schematic for the First Simulation

This example requires that you

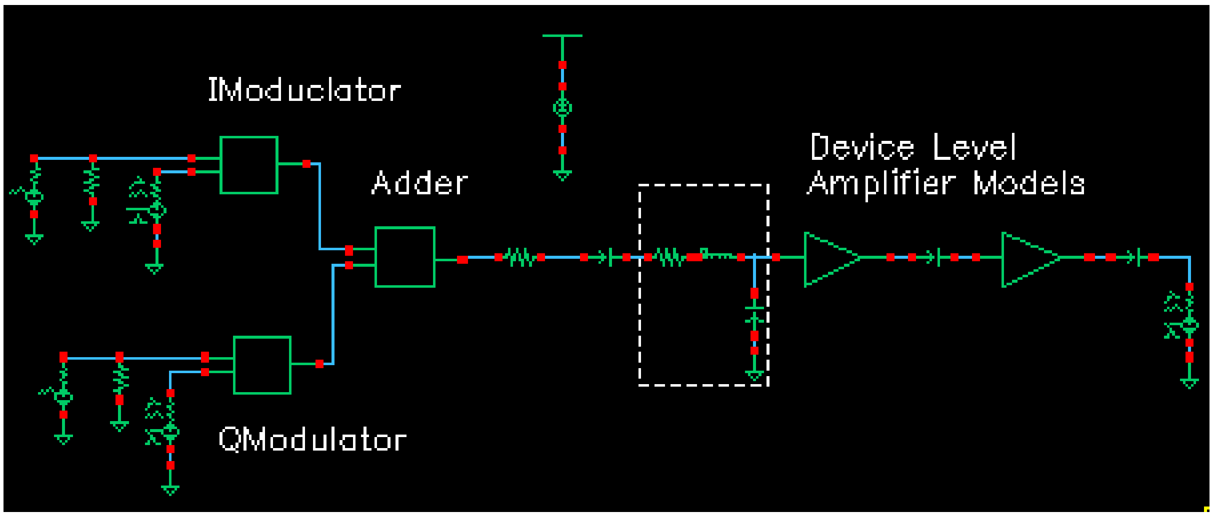
- Open the *EF_example* schematic
- Create a new, empty cellview named *sparamfirst*
- Copy part of the *EF_example* schematic and paste it in the *sparamfirst* schematic
- Simulate the new schematic.

  This first simulation generates the S-parameters of a linear time-invariant part of the original circuit (*EF_example*). The S-parameter file is stored so you can use it in the second simulation to import S-parameters.

### Opening the EF_example Schematic

1. Open the *EF_example* schematic as described in “Opening the EF_example Circuit in the Schematic Window” on page 522.

The *EF_example* schematic is the original schematic used in the first Envelope example in this chapter. Leave the *EF_example* schematic open in the background.




### Creating and Editing the New Schematic

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

2. In the New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

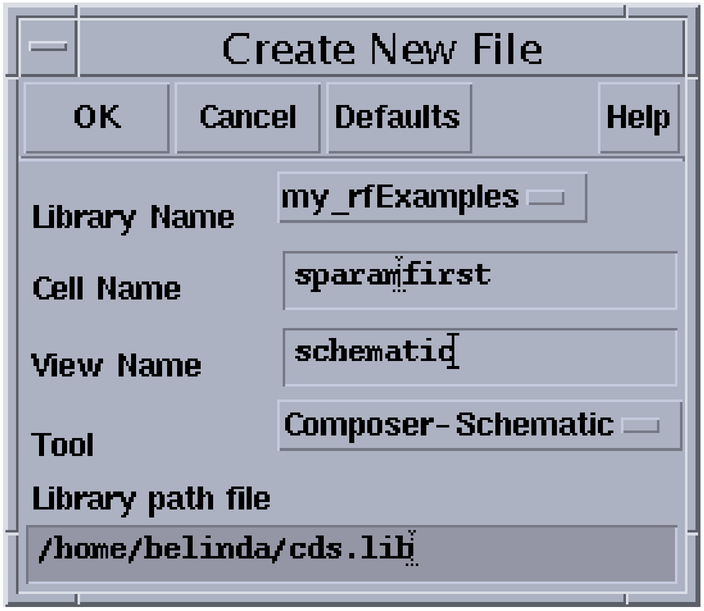
      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* field, type *sparamfirst*, a name for the new view.

   c. In the *View Name* field, type `schematic`.

   d. Choose *Composer-Schematic* for *Tool*.

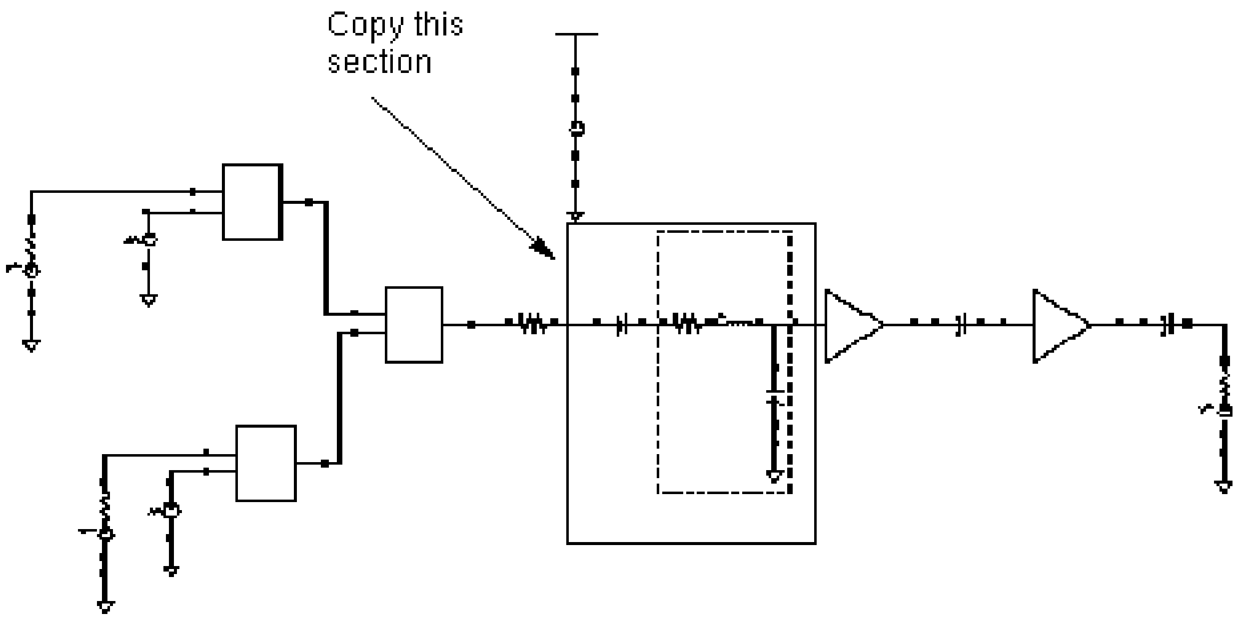
**e.** Click *OK*.




After you click *OK,* the new empty Schematic window for *sparamfirst* appears.

**3.** In the *EF_example* Schematic window, copy the part of the schematic shown in Figure 8-18 and paste it into the new Schematic window.

### Portion of EF_example Schematic to Copy




**a.** In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

**b.** Following the prompt,

```
> point at object to copy
```

left click and drag to create a box around the part of the *EF_example* circuit indicated in 8-18.

The selected *EF_example* components are highlighted in yellow.

**c.** Following the prompt,

```
> point at reference point for copy
```

click inside the outlined elements.

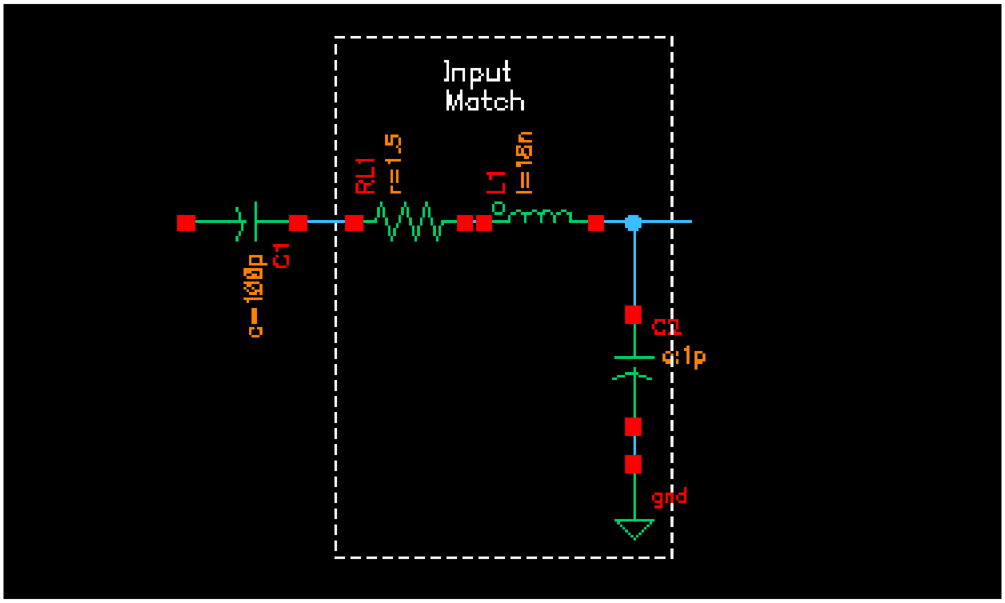
**d.** Following the prompt,

```
> point at destination point for copy
```

move the cursor to the *sparamfirst* schematic window and click there. This copies the selected part of the *EF_example* circuit into the empty Schematic window.

The empty Schematic window now contains the portion of the *EF_example* schematic shown in 8-18.

**e.** If necessary, choose *Window - Fit* to center the copied section of the *EF_example* circuit in the *sparamfirst* Schematic window.



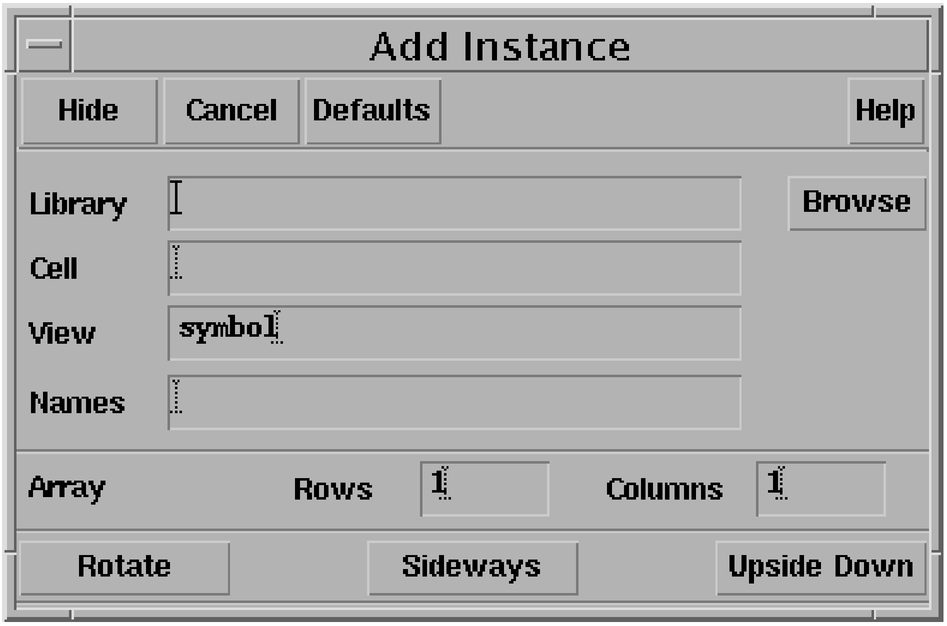

**4.** In the original *EF_example* Schematic window, choose *Window - Close*.

**5.** The *EF_example* Schematic window closes.

## Adding Components to the Schematic

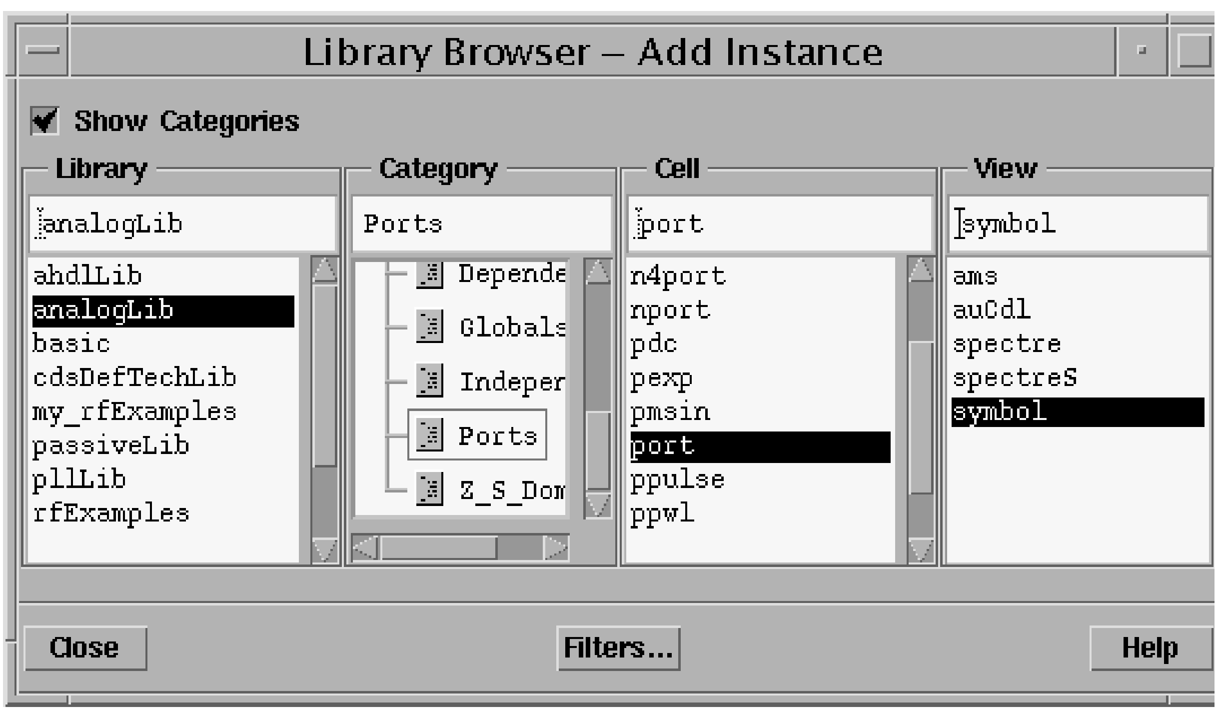
**1.** In the *sparamfirst* Schematic window, choose *Add – Instance.*

The Add Instance form appears.




**2.** In the Add Instance form, click *Browse*.

The Library Browser – Add Instance form appears.



**3.** In the Library Browser – Add Instance form, do the following:

**a.** Select the *analogLib* and highlight *Show Categories*.

**b.** Scroll down and select *Sources* in the *Category* list box.

The *Category* list box expands to display several subcategories under *Sources*.

**c.** Scroll down and select the *Ports* subcategory under *Sources*.

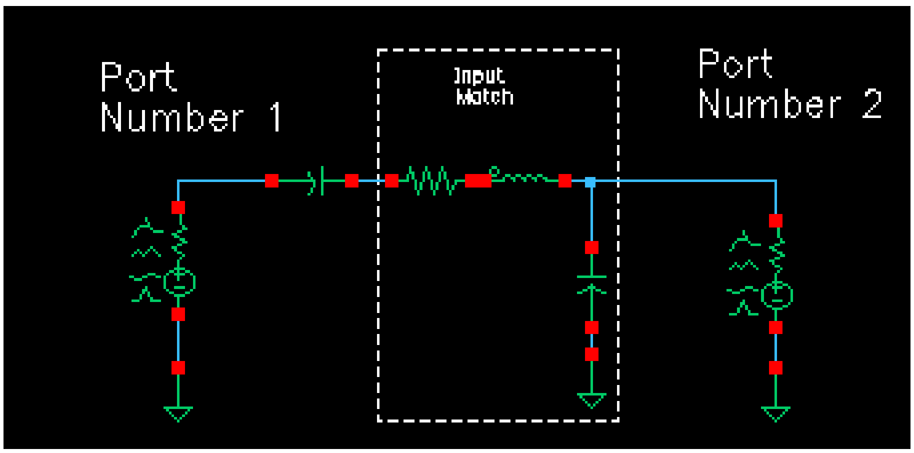
**d.** Select *port* for *Cell*.

**e.** Select *symbol* for *View*.

A copy of the port component moves with the cursor into the new Schematic window.

**f.** Place two *port* symbols in the *sparamfirst* Schematic window as shown in Figure 8-18.

**Placement and Port Numbering in the sparamfirst Schematic**




**4.** In the Library Browser – Add Instance form, do the following:

**a.** Select *analogLib* for *Library*.

**b.** Select *Everything* for *Category*.

**c.** Select *gnd* for *Cell*.

**d.** Select *symbol* for *View*.

**e.** Place two *gnd* components in the Schematic window as shown in Figure 8-18. Press *Esc* when you are done.

5. In the Schematic window, choose *Add – Wire (narrow)* and wire up the new components in the *sparamfirst* Schematic window. Press *Esc* when you are done.
6. In the new Schematic window, choose *Edit – Properties – Objects*.
7. For each of the *port* components in the new schematic, do the following in the Edit Object Properties form:
   a. Select the *port*.
   b. Type the appropriate number in the *Port number* field.

      Figure 8-18 shows the appropriate number to type. Leave the Port resistance at its default of 50 Ohms.
   c. Click *Apply*.
8. In the new Schematic window, choose *Design – Check and Save*.

## Setting Up the sparamfirst Schematic

1. In the *sparamfirst* Schematic window, open the ADE window as described in "Opening the Simulation Window" on page 523.
2. In the ADE window, set up the model libraries as described in "Setting Up the Model Libraries" on page 524.

## Running the SP Simulation

Run an SP analysis to write out an S-parameter file, `sparam.practice`, that describes the *sparamfirst* circuit.

1. In the ADE window, choose *Analyses – Choose*.

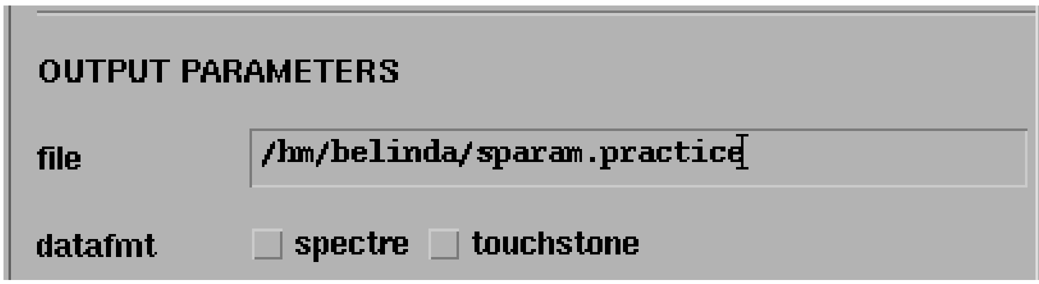
   The Choosing Analyses form appears.
2. In the Choosing Analyses form, do the following:
   a. Highlight *sp* for *Analysis*.
   b. Highlight *Frequency* for *Sweep Variable*.
   c. Highlight *Start – Stop* for *Sweep Range*.
   d. Type `100K` for *Start* and `10G` for *Stop*.
   e. Choose *Automatic* for *Sweep Type*.

   **f.** Highlight *no* for *Do Noise*.

3. In the Choosing Analyses form, click *Options.*

   The S-parameter Options form appears.

4. In the OUTPUT PARAMETERS section of the S-parameter Options form, type the path for the output S-parameter file in the *file* field. Click *Apply*. Click *OK*.




5. In the Choosing Analyses form, click *Apply*. Click *OK*

6. In the ADE window, choose *Simulation – Netlist and Run*.

7. Check the CIW for a message that says the simulation completed successfully.

   The sp analysis wrote an S-parameter file, `sparam.practice`, to the directory you indicated.

### The S-Parameter File

Use the S-parameter output file, `sparam.practice`, you created as input for the next simulation. You can use a text editor to open and examine the format of the S-parameter output file. You can use this format to create S-parameter files from other sources.

The top of the `sparam.practice` S-parameter file looks like this.

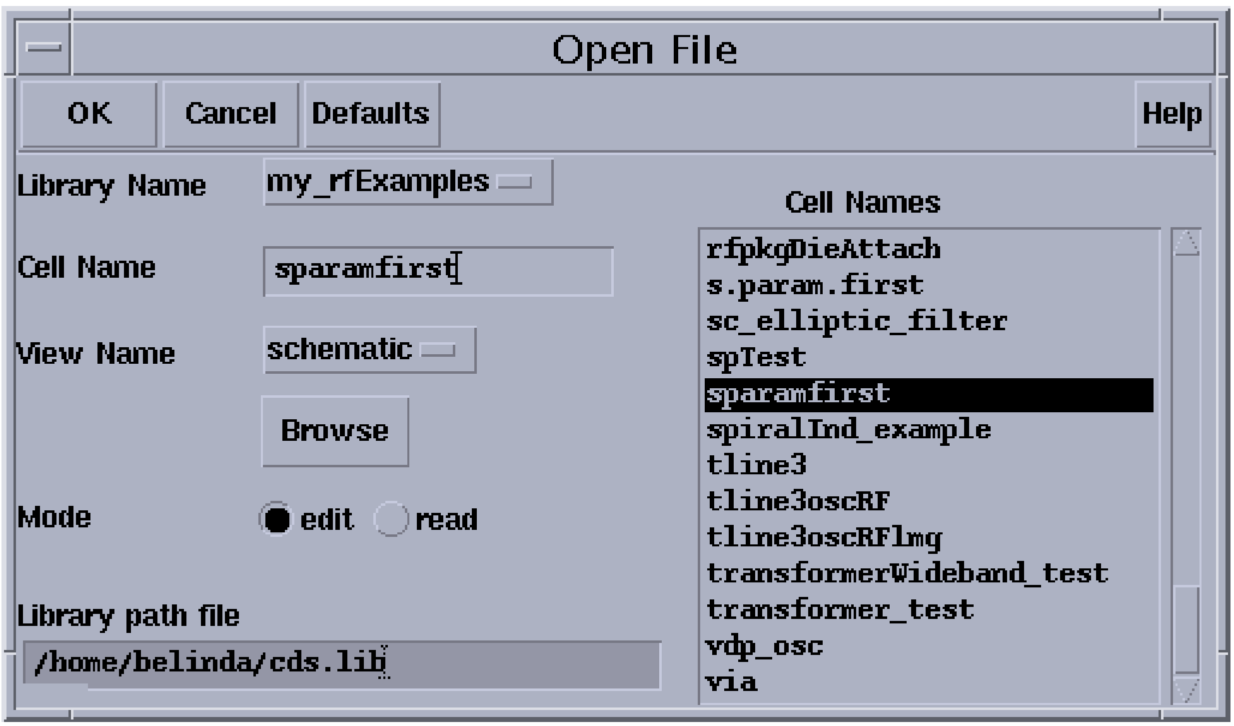
```
; S-parameter data file /hm/belinda/sparam.practice.
; Tue Feb 28 18:10:27 2006
; Number of ports is 2

reference resistance
                 port2 = 50.000000
                 port1 = 50.000000

format freq:  s1:1(real,imag)   s2:1(real,imag)
              s1:2(real,imag)    s2:2(real,imag)

1.00000000e+05:    0.99996,      -0.00628293      4.02664e-05,      0.00628293
                4.02664e-05,      0.00628293         0.99996,      -0.00634576
1.25892541e+05:    0.999936,     -0.00790956      6.38165e-05,      0.00790956
```

```
                6.38165e-05,       0.00790956          0.999936,      -0.00798866
1.58489319e+05:    0.999899,      -0.00995718       0.000101139,       0.00995717
                0.000101139,        0.00995717            0.999898,      -0.0100567
```

## Setting Up and Running the Second sp Simulation

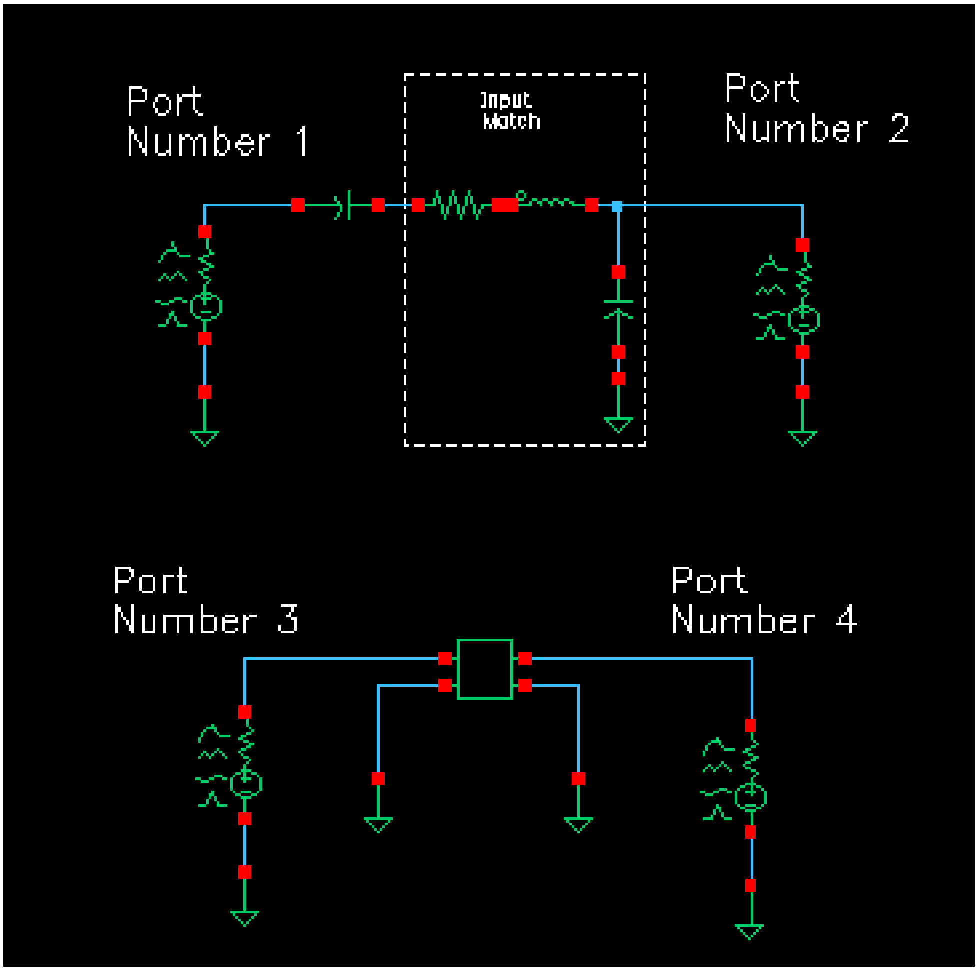
In this simulation, use the S-parameter output file, `sparam.practice`, from the first simulation as input to the second simulation. The S-Parameter file models the components in the `sparamfirst` Schematic.

1. If necessary, open the `sparamfirst` schematic.




2. Add components to the *sparamfirst* schematic you created for the SP analysis as shown in Figure 8-19.

**Figure 8-19 Wired sparamfirst Schematic**




The components to add are described in the following table. See “Adding Components to the Schematic” on page 621 for information on adding components to a schematic.

**Table 8-4 Components to Add to the sparamfirst Schematic**

| Library | Category | Subcategory | Cell | View | Number to Add |
|---|---|---|---|---|---|
| analogLib | Sources | Ports | n2port | symbol | 1 |
| analogLib | Sources | Ports | port | symbol | 2 |
| analogLib | Everything | None | gnd | symbol | 4 |

**3.** Wire the components as shown in Figure 8-19.

**4.** Edit the properties on the two new ports to assign port numbers 3 and 4 as shown in Figure 8-19. Then edit the properties on the new *n2port*.

**5.** In the Schematic window select the *port* on the lower left.

**6.** In the Schematic window, choose *Edit – Properties – Objects.*

**7.** In the Edit Object Properties form, do the following:

   **a.** Type 3 in the *Port number* field.

   Figure 8-19 on page 627 shows the appropriate number to type. Leave the *Port resistance* at the default of 50 Ohms.

   **b.** Click *Apply*.

   **c.** In the Schematic window, select the *port* on the lower right.

   **d.** Type 4 in the *Port number* field.

   **e.** Leave the *Port resistance* at the default of 50 Ohms.

   **f.** Click *Apply*.

**8.** In the Schematic window, select the *n2port*.

The Edit Object Properties form changes to display information for the *n2port*.

**9.** In the Edit Object Properties form, do the following and click *OK*:

   **a.** In the *S-parameter data file* field, type the absolute path to the S-parameter file you created in the first simulation.

   `/hm/belinda/sparam.practice`

   **b.** Choose *rational* for *Interpolation method*.

   The form changes to let you add additional information.

   **c.** Type `.001` for *Relative error*.

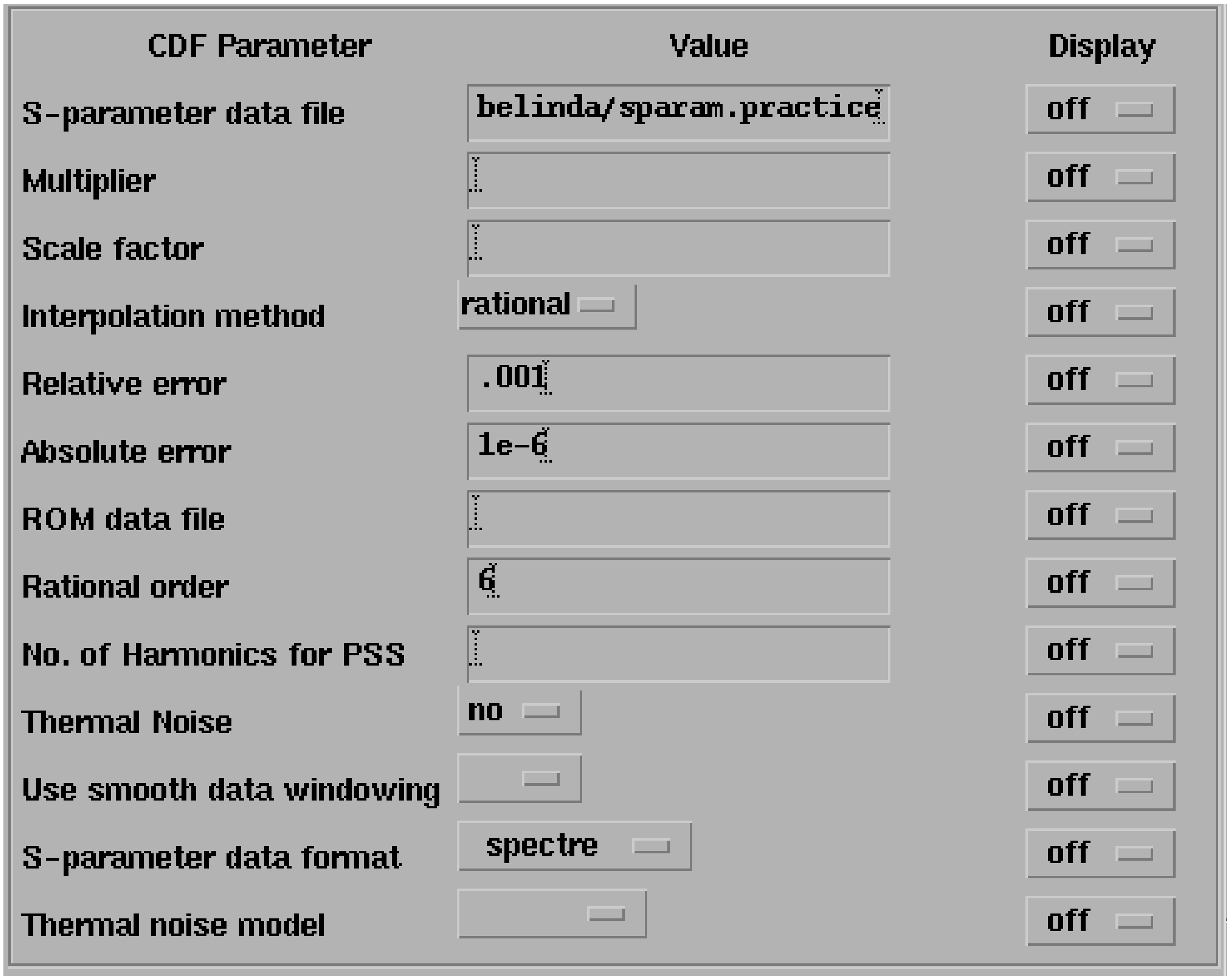
   **d.** Type `1e`$^{-6}$ for *Absolute error*.

   **e.** Type 6 for *Rational order*.

   **f.** Select *no* for *Thermal Noise*.

   **g.** Select *spectre* for *S-parameter data format*.

The completed Edit Object Properties form for the *n2port* looks like this.




**10.** In the Schematic window, choose *Design – Check and Save*.

**11.** In the ADE window, choose *Simulation – Netlist and Run*.

### Plotting Results

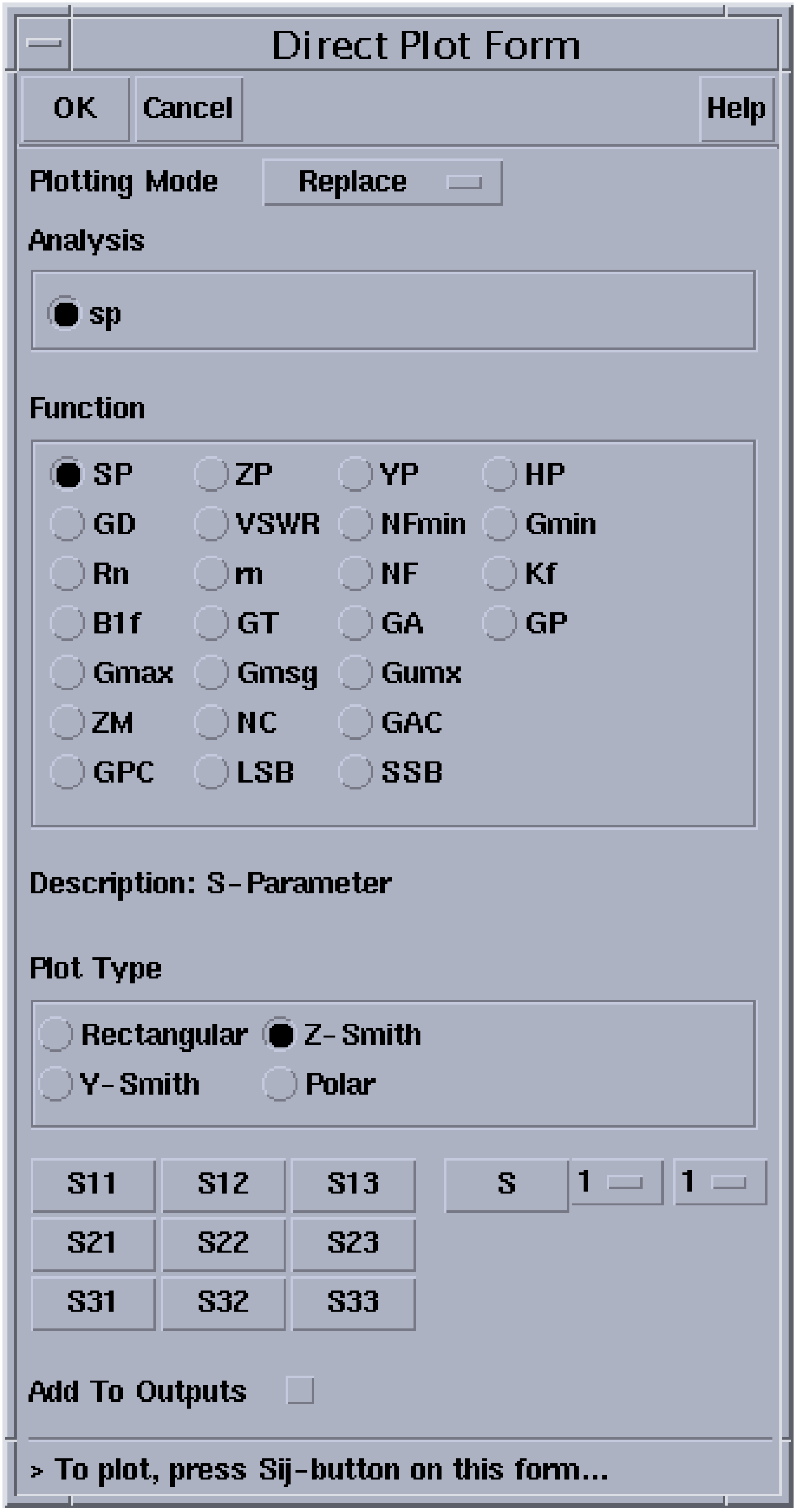
1. In the ADE window, choose *Results – Direct Plot – Main Form.*

   The Direct Plot form appears.

2. In the Direct Plot form, choose *Replace* for *Plotting Mode*.
3. Highlight *sp* for *Analysis*.
4. Highlight *SP* for *Function*.

5. Highlight *Z-Smith* for *Plot Type*.

The Direct Plot form looks like the following.



**a.** Click *S11*.

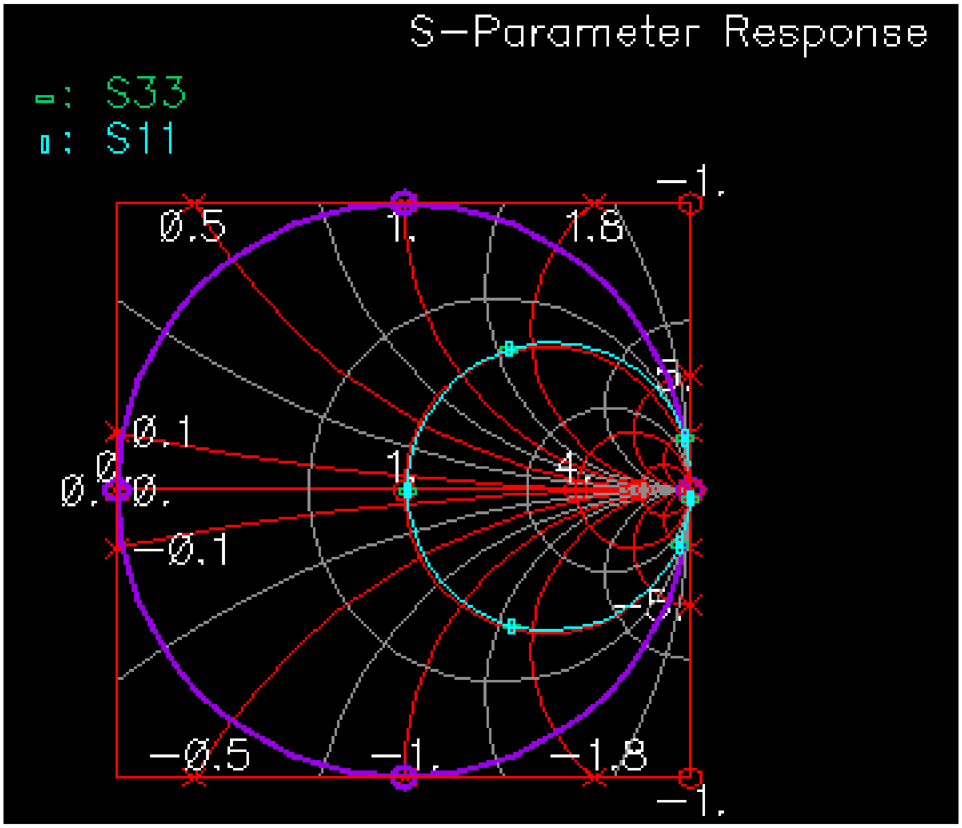
The plot for S11 appears in the waveform window.

**6.** In the Direct Plot form, do the following:

**a.** Choose *Append* for *Plot Mode*.

**b.** Click *S33*.

The plot for S33 is appended to the S11 plot. The two plots lie one on top of the other which shows that the two plots are identical. Thus the results produced by the first simulation are the same as those produced by the second simulation which used the S-parameter input file.




You can also check other plots for equivalency. For example, you can do the following to plot S21 and S43.

**1.** Choose *Replace* for *Plotting Mode.*

**2.** Plot S21 as described in "Plotting Results" on page 629.

**3.** Plot S43 as follows.

**a.** Choose *Append* for *Plotting Mode*.

**b.** Select *4* in the first cyclic field to the right of *S*.

**c.** Select *3* in the second cyclic field to the right of *S*.

**d.** Click *S* to create the plot.

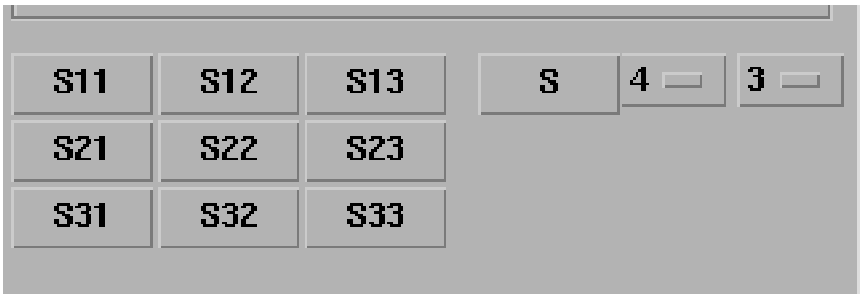
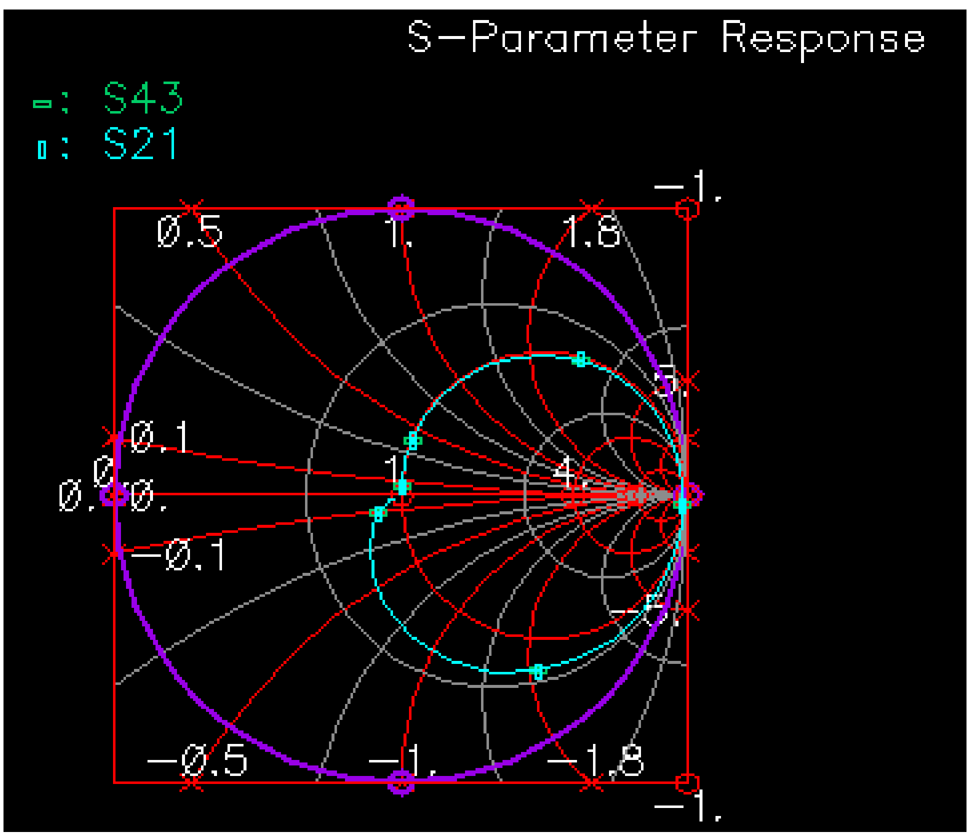



Figure 8-4 is the plot produced by appending `S431` to the `S21` plot.

### Plot Showing S21 and S43



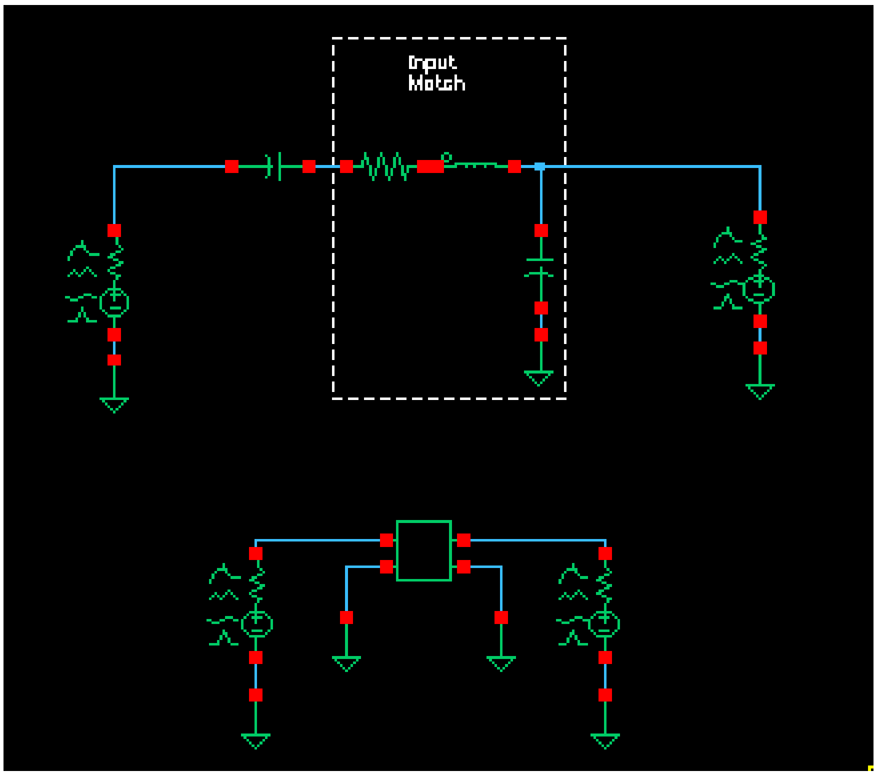
## Using the S-Parameter Input File with a Spectre RF Envlp Analysis

Use the `sparam.practice` S-parameter format data file as input to an Envelope analysis. Again, it compares results of an *envlp* analysis using the S-parameter file to results of an *envlp* analysis using the original components.

### Setting Up the Schematic

1. Open the *EF_example* schematic. Choose *File – Open* in the CIW.
2. Open the *sparamfirst* schematic and move it to the background.

**Figure 8-20 The sparamfirst Schematic**



3. In the *EF_example* schematic. choose *Design – Save As* in the Schematic window.

   The Save As form appears.

4. In the Save As form, do the following:

**a.** In the *Library Name* field, type the name of your local, editable copy of the *rfExamples* library.

**b.** In the *Cell Name* field, type *EF_example_copy,* the name for the copy of the *EF_example* schematic.

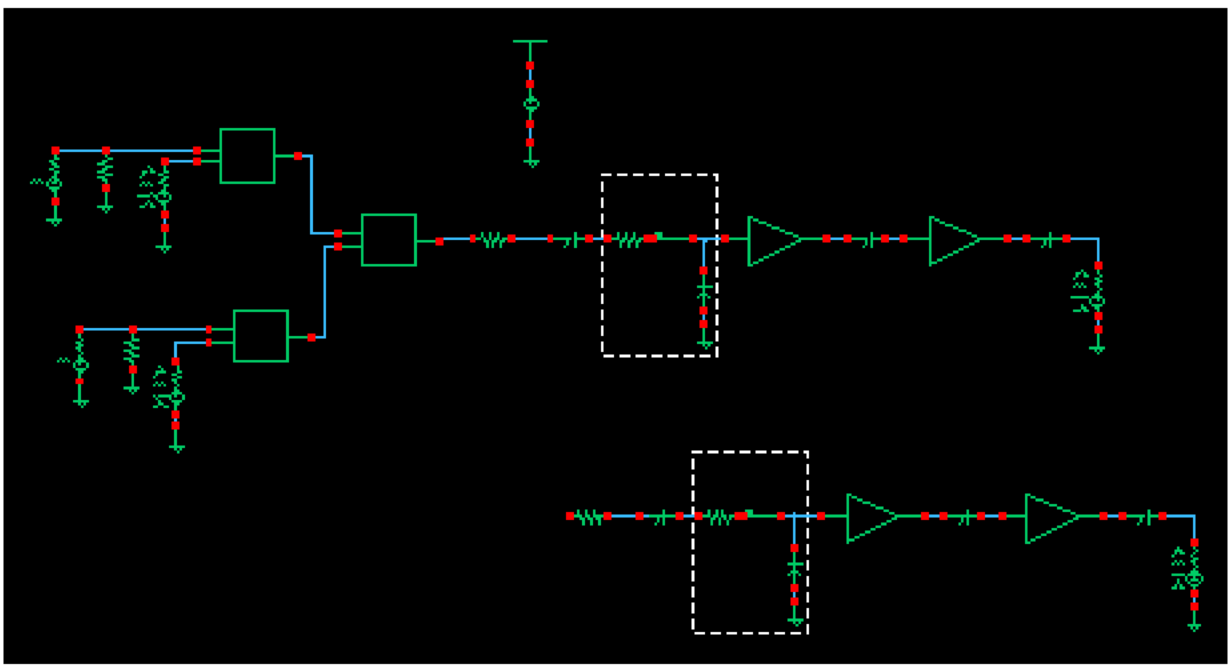
**c.** Click *OK*.

You now have a copy of the *EF_example* schematic called *EF_example_copy*.

In the *EF_example* Schematic window., choose *Window – Close* to close the *EF_example* schematic.

**5.** Open the *EF_example_copy* schematic.

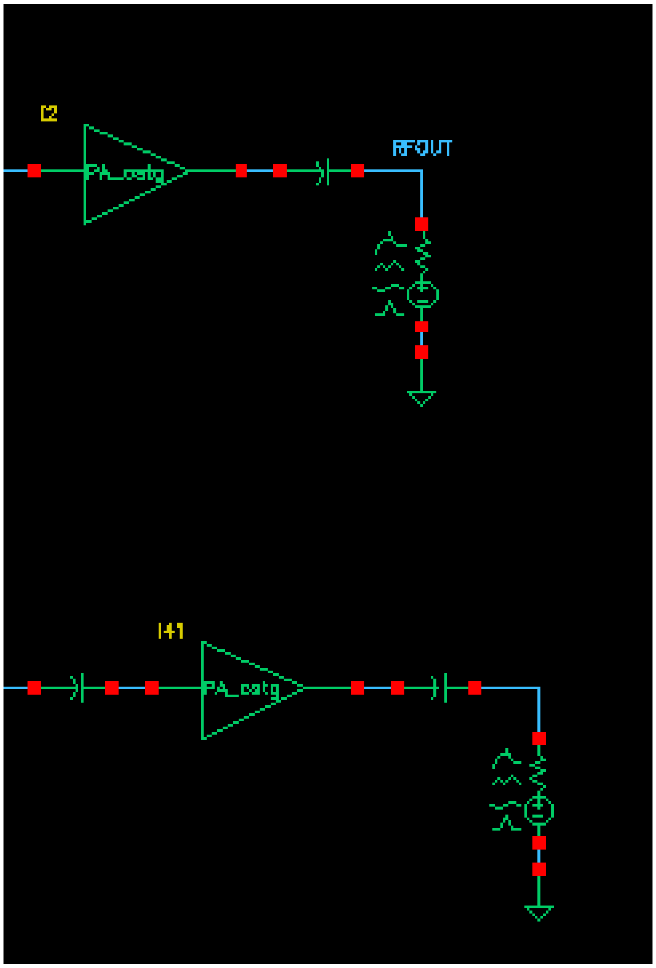
**6.** In the *EF_example_copy* Schematic, copy everything to the right of the adder and paste the copy below the original.



*Important*

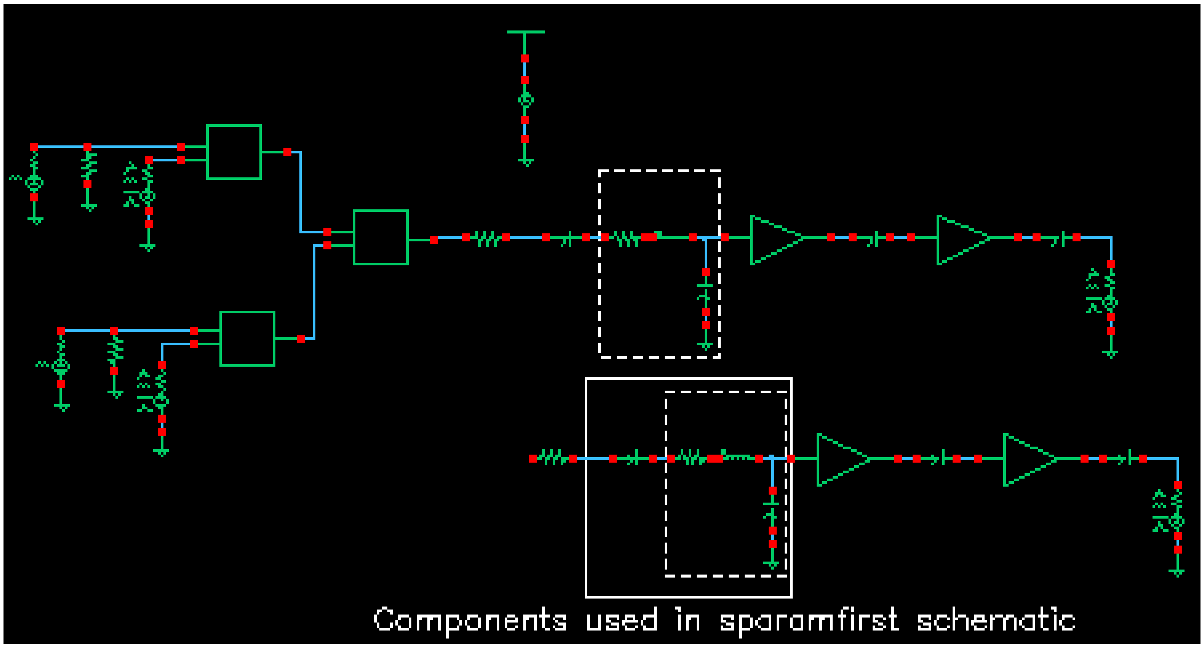
Be sure to remove the *RFOUT* label from the duplicate branch or the two branches are shorted together at their outputs.

**7.** To delete RFOUT from the bottom branch, choose *Edit – Delete,* highlight *RFOUT* and then press *Esc*.




**8.** In the lower branch of the *EF_example_copy* schematic, delete the circuitry you used to create the S-parameter file in the *sparamfirst* schematic, see Figure 8-21.

**Figure 8-21 Components to Delete from the EF_example_copy Schematic**




9. Choose *Edit – Delete* and follow the prompts at the bottom of the *EF_example_copy* schematic window.

10. Following the prompt,

    ```
    > point at object to delete
    ```

    left click and drag to create a box around the indicated part of the *EF_example_copy* circuit.

    The selected *EF_example_copy* components are deleted.

11. Press *Esc* to stop deleting.

12. Replace the deleted components in the *EF_example_copy* schematic with the *n2port* component and the two attached *gnd* (ground) components from the *sparamfirst* schematic.The *n2port* and *gnd* components are shown in Figure 8-20 on page 634.

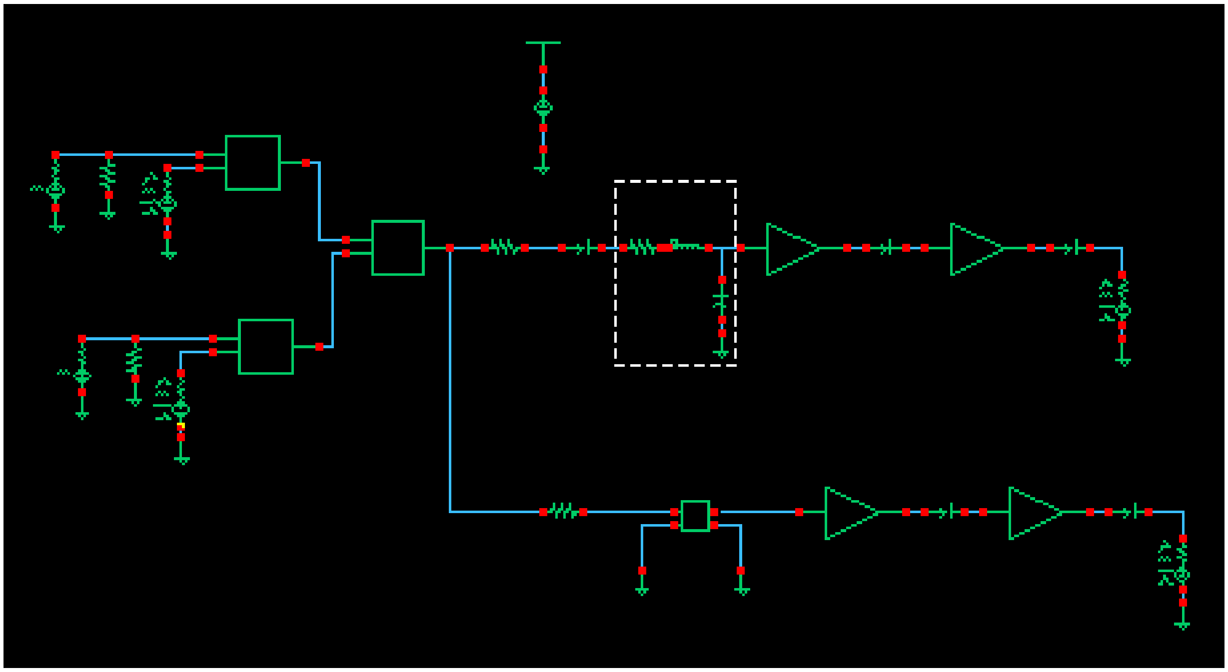
    a. In the *sparamfirst* schematic.window, choose *Edit – Copy* to copy the *n2port* and the two attached *gnd* components. Follow the prompts at the bottom of the *sparamfirst* schematic window.

       **Note:** (Close the *sparamfirst* schematic window when you are done.)

    b. As you move the cursor into the *EF_example_copy* Schematic window, a copy of the *n2port* and *gnd* components moves with the cursor. Place the components in the *EF_example_copy* Schematic window as shown in Figure 8-21

**13.** In the Schematic window, choose *Add – Wire (narrow)* and wire up the new components in the *EF_example_copy* Schematic window. Press *Esc* when you are done.

**The EF_example_copy Schematic with n2port Component in Place**



**14.** In the Schematic window, select the *n2port* component and choose *Edit – Properties – Object*. Then verify that the *n2port* component has the following properties.

**a.** The absolute path to the S-parameter data file displays in the S-parameter data file field. In this example,

`/hm/belinda/sparam.practice`

**b.** *Interpolation method* is *Rational*.

**c.** *Relative error* is `.001`.

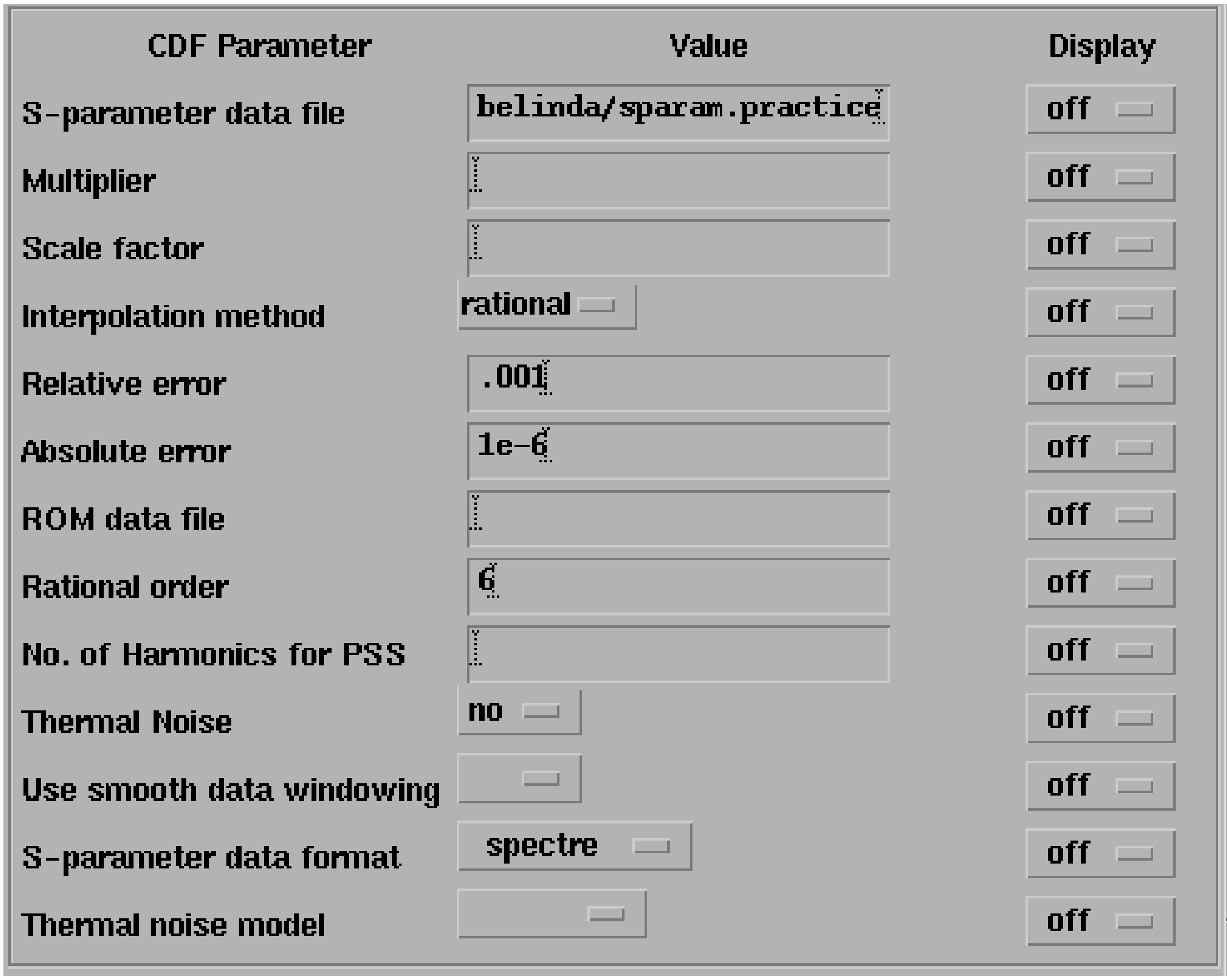
**d.** *Absolute error* is `1e`$^{-6}$.

**e.** *Rational order* is `6`.

**f.** *Thermal Noise* is *no*.

**g.** *S-parameter data format* is *spectre*.

The completed Edit Object Properties form for the *n2port* looks like this.




**15.** In the Schematic window, choose *Design – Check and Save*.

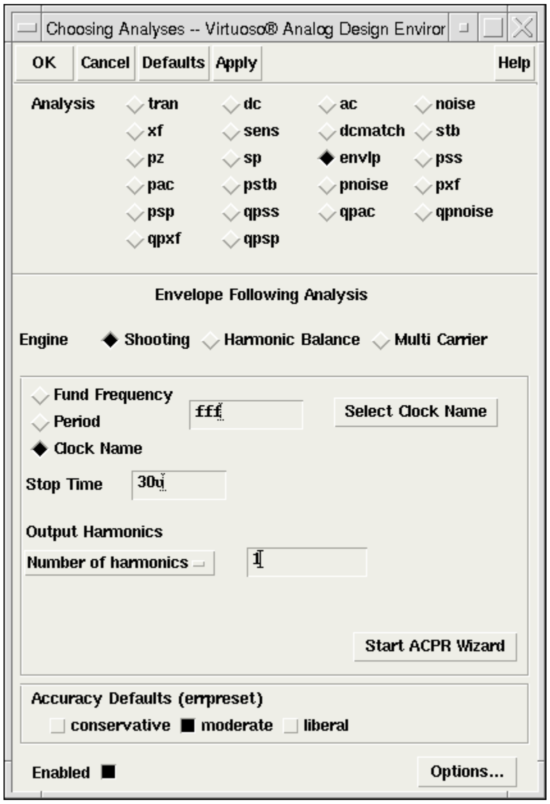
### Setting Up and Running the Simulation

Set up and run an *envlp* analysis as described in the sections listed here.

- "Opening the Simulation Window" on page 523
- "Setting Up the Model Libraries" on page 524
- "Editing PORT0 and PORT1 in the EF_example Schematic" on page 525
- "Setting Up an Envelope Analysis" on page 527.

When you set up the *envlp* analysis, use 30u for the *Stop Time* value.

The completed *envlp* Choosing Analyses form looks like this.




➤ In the ADE window, choose *Simulation – Netlist and Run*.

Check the CIW for a message that says the simulation completed successfully.

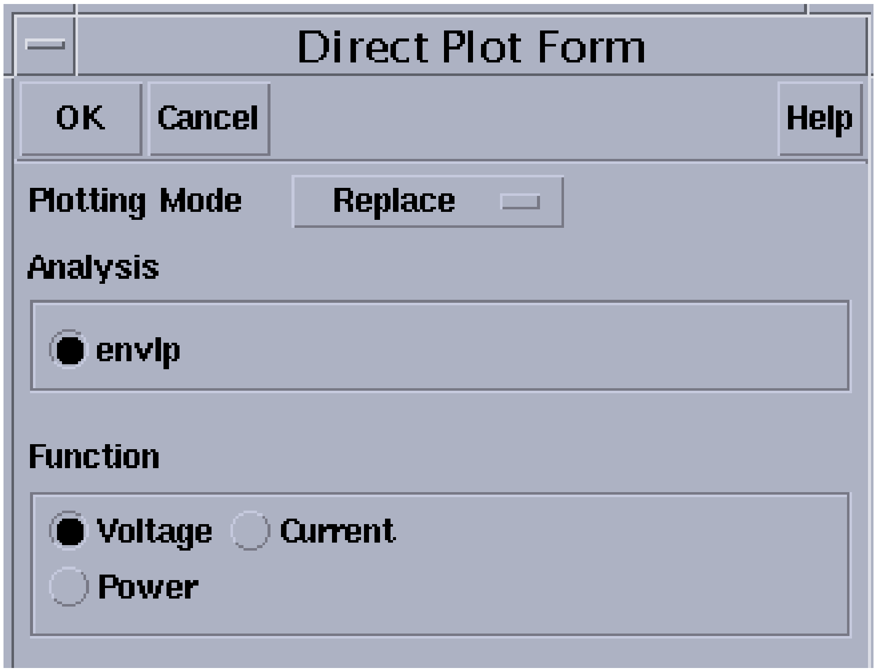
### Displaying the envlp Following Results

1. In the ADE window, choose *Results – Direct Plot – envlp*.

   The Direct Plot form appears.

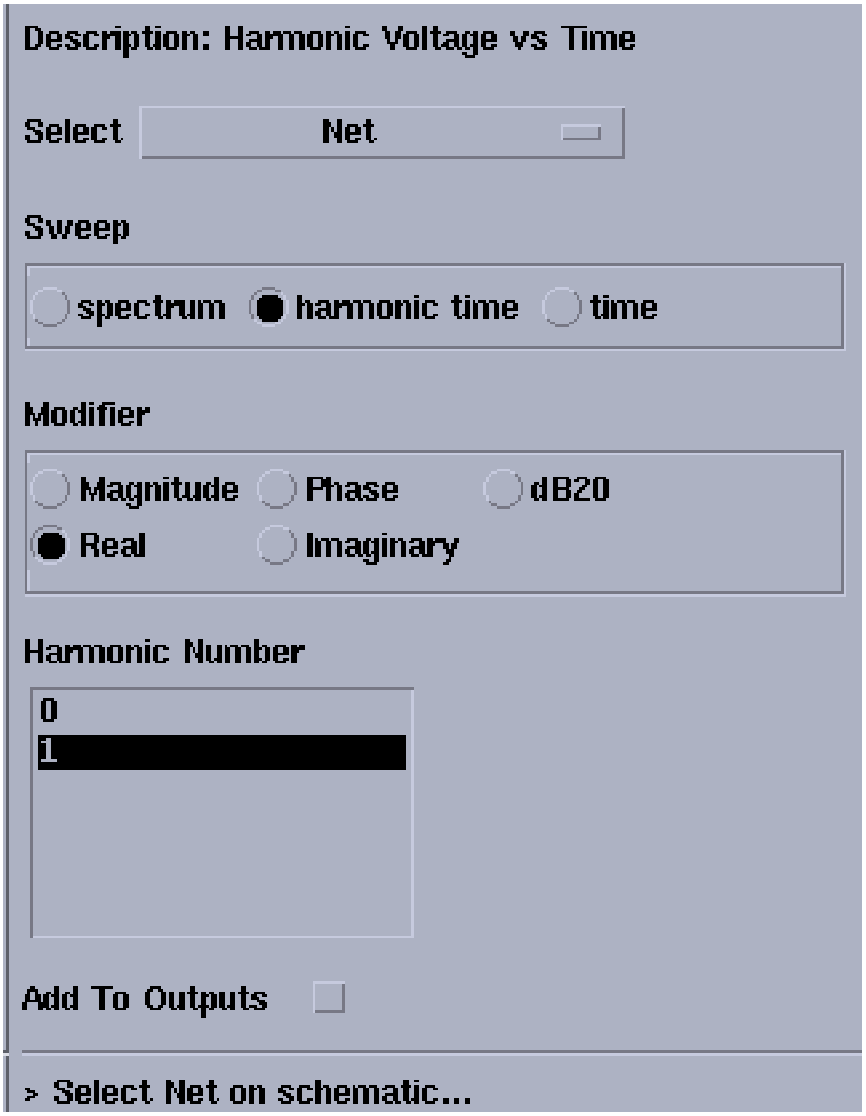
2. In the Direct Plot form, choose *Replace* for *Plotting Mode*.
3. Highlight *envlp* for *analysis.*
4. Highlight *Voltage* for *Function*.

The top of the form looks like this.




5. Highlight *Harmonic Time* for *Sweep*.
6. Highlight *Real* for *Modifier*.
7. Choose *1* for *Harmonic Number*.

The bottom of the form looks like this.




The prompt on the bottom of the Direct Plot form is

```
> Select Net on Schematic....
```

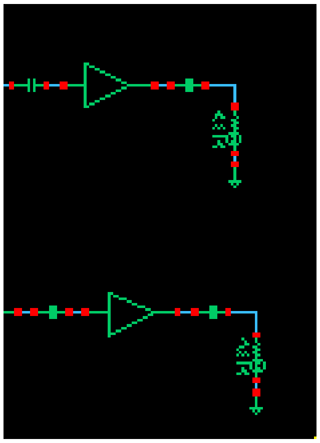
**8.** Follow the prompt. In the Schematic window, click one of the output nodes as shown in "Output Nodes" on page 643.

**9.** Change the *Plot Mode* to *Append.*

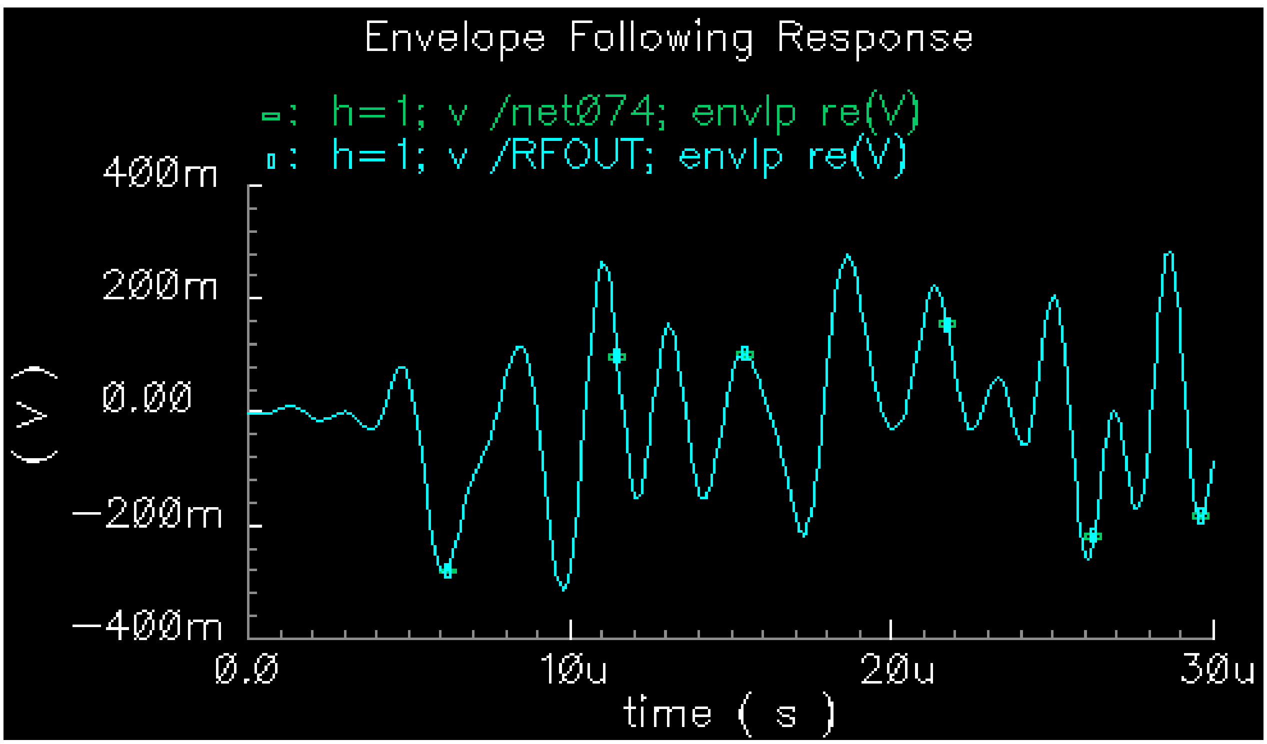
**10.** Click the other output node as shown in "Output Nodes" on page 643.

You can also confirm that the imaginary parts of the waveforms match.

#### Figure 8-22 Output Nodes



The coincident plots in the waveform window show that the output is the same at both nodes. When the S-parameters are correct, the *n2port* device accurately simulates the circuitry represented by the S-parameters.



# Measuring AM and PM Conversion with Modulated PAC, AC and PXF Analyses

This section describes AM and PM small signal characterization in Spectre RF. AM/PM conversion measurements allow you to investigate the amplitude and phase characterization of RF circuits. The modulated PAC, AC and PXF analyses calculate conversion gain and other characteristics between AM, PM, and SSB sources and AM, PM, and SSB outputs.

AM/PM conversion computes transfer functions and gain measurements involving AM and PM inputs and outputs. In general, there are three possible types of inputs and outputs for which you might want to compute transfer functions.

- Unmodulated or single sideband (SSB)
- Amplitude modulated sinusoids (AM)
- Phase modulated sinusoids (PM)

## The Modulated Analysis Settings

In total, a modulated analysis can calculate 9 cross conversion metrics from 3 types of inputs (AM, PM and SSB) to 3 types of outputs (AM, PM and SSB). Use the *Modulated Analysis* section in the PAC and PXF Choosing Analyses forms to set up your analyses.

- *Input Type* (for PAC analysis) and *Output Type* (for PXF analysis) allow you to choose whether to measure all 9 modulated conversions (choose *SSB*/*AM*/*PM*) or only 3 conversions (choose *SSB*).
- *Output Modulated Harmonic List* (for PAC output modulations) and *Input Modulated Harmonic List* (for PXF modulated sources) specify a vector of harmonic indexes. You can type the indexes separated by spaces or you can choose them from a scrolling list.
- *Input Modulated Harmonic* (for PAC) and *Output Modulated Harmonic* (for PXF) specify a single harmonic index for PAC input source modulation or for PXF output modulation. You can type the index or you can choose it from a scrolling list. This choice appears when you select the *SSB*/*AM*/*PM Input* or *Output Type*.

  For *SSB Input Type* (PAC) or *Output Type* (PXF), specify an *Output Upper Sideband* or *Input Upper Sideband.* You can type the sideband index or you can choose it from a scrolling list.
- For modulated PXF analysis, select the *Output Probe Instance* or the *Positive* and *Negative voltage Output Nodes* from the schematic.

This example illustrates how to

- Create the *EF_AMP* schematic, a modified copy of the *EF_example* schematic.
- Save a copy of the *EF_AMP* schematic in the *my_rfExamples* library. You use this schematic for other examples later.
- Set up and run the necessary PSS, modulated PAC and modulated PXF analyses.
- Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

After you run the simulation, use the Direct Plot form to plot the modulated analysis results.

## Creating the EF_AMP Circuit

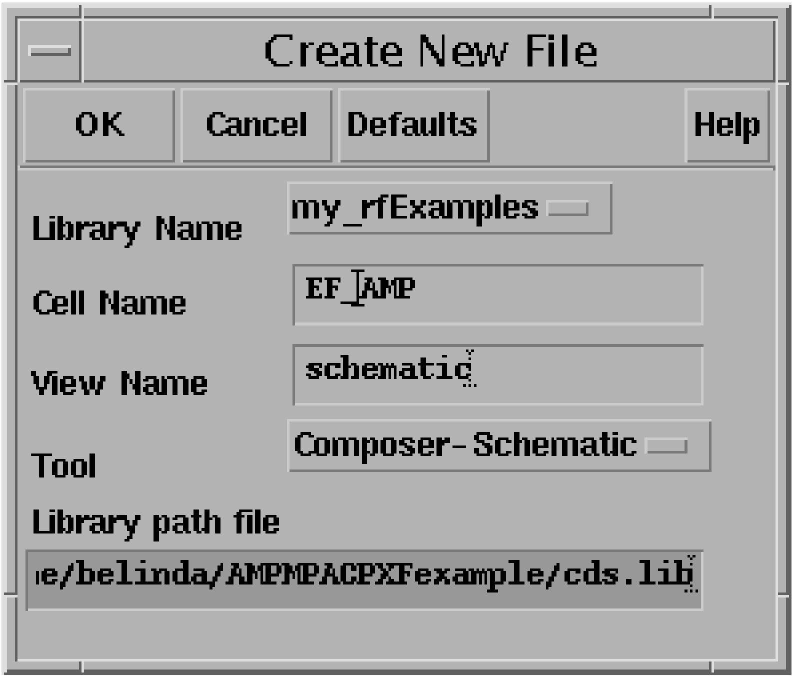
Create a modified copy of the *EF_example* schematic, *EF_AMP* and save the *EF_AMP* schematic in the *my_rfExamples* library you created. You use the *EF_AMP* schematic in the Jitter example.

### Create a New Empty Schematic Window

Open the *EF_example* schematic in a schematic window. Open another empty schematic window, name it EF_AMP and copy the *EF_example* schematic into the EF_AMP schematic window.

1. In the CIW, choose *File – New – Cellview*.

The Create New File form appears.




2. In the Create New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

      In the *Cell Name* field, enter *EF_AMP*.

   b. In the *View Name* field, enter *schematic*.

   c. In the *Tool* cyclic field, select *Composer-Schematic*.

   d. Click *OK*.

A new, empty Schematic window named *EF_AMP* opens.

### Open and Copy the EF_example Schematic

Open the *EF_example* schematic and copy it into the empty *EF_AMP* Schematic window. Then edit *EF_AMP* before simulating.

1. In the CIW, choose *File – Open*.

The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

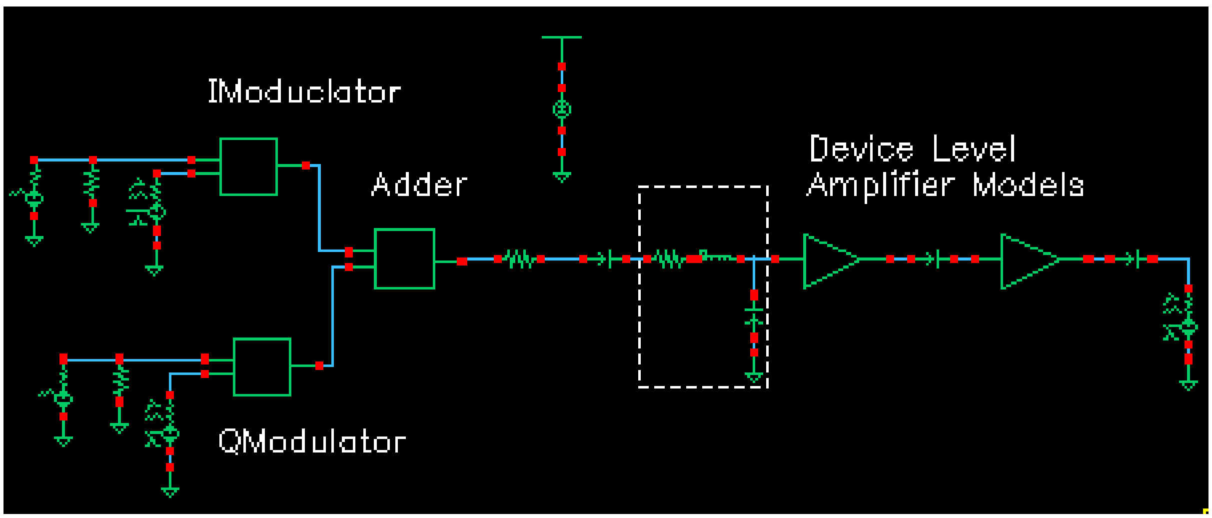
   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic.




3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts.

4. Following the prompt at the bottom of the Schematic window,

   `> point at object to copy`

   left click and drag to create a box around the entire *EF_example* circuit.

   The *EF_example* components are highlighted.

5. Following the prompt,

   `> point at reference point for copy`

   click inside the outlined elements.

6. Following the prompt,

   `> point at destination point for copy,`

   move the cursor to drag a copy of the entire circuit into the *EF_AMP* Schematic window and click there.

   The *EF_AMP* Schematic window now contains a copy of the *EF_example* schematic.

7. In the *EF_AMP* Schematic window, if necessary, choose *Window - Fit* to scale and center the circuit in the Schematic window.

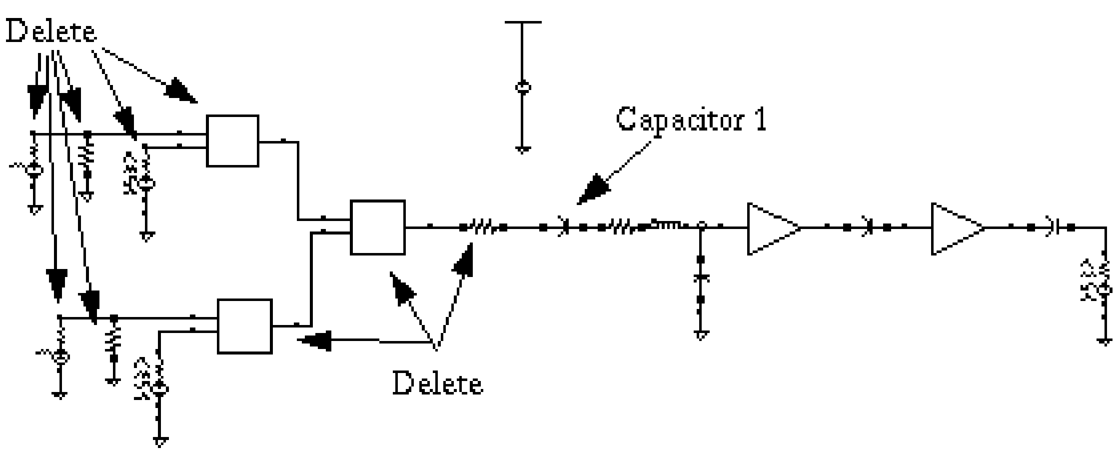
8. In the *EF_example* Schematic window, choose *Window - Close*.

   The *EF_example* Schematic window closes.

### Editing the EF_AMP Schematic

In the *EF_AMP* Schematic window, delete components and their associated connecting wires as shown in Figure 8-23. The final *EF_AMP* Schematic should look like Figure 8-25 on page 650.

**Figure 8-23 Components and Wires to Delete from the EF_LoadPull Schematic**

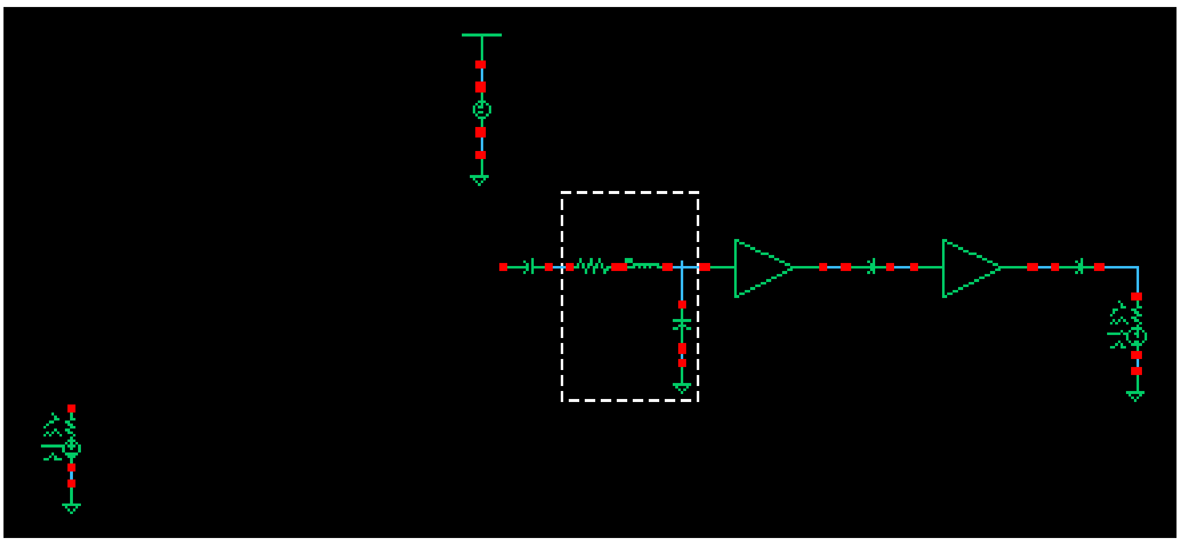



**Note:** If you need assistance with methods for editing the schematic, see the *Virtuoso® Schematic Composer™ User Guide*.

### *Delete Components and Wires*

1. In the *EF_AMP* Schematic window, choose *Edit – Delete.*
2. Click a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. After you delete the components and wires, the EF_AMP schematic looks like Figure 8-24.

**Figure 8-24 EF_AMP Schematic After Deleting Components**



5. In the Schematic window, choose *Edit – Move* to move both *PORT 3* and *VCC* as shown in “The Edited EF_AMP Schematic” on page 650.
6. To move *PORT 3* closer to capacitor *C!,* click and drag to create a rectangle around *PORT 3* and its ground.

   *PORT 3* and the *Gnd* are highlighted.

7. Click and drag the components to move them close to capacitor `C1` as shown in “The Edited EF_AMP Schematic” on page 650.
8. To move *VCC* below and between the two device-level amplifier models, click and drag to create a rectangle around *VCC*.

   *VCC* is highlighted in yellow.

9. Click and drag the components to move them below and between the two device-level amplifier models as shown in “The Edited EF_AMP Schematic” on page 650.

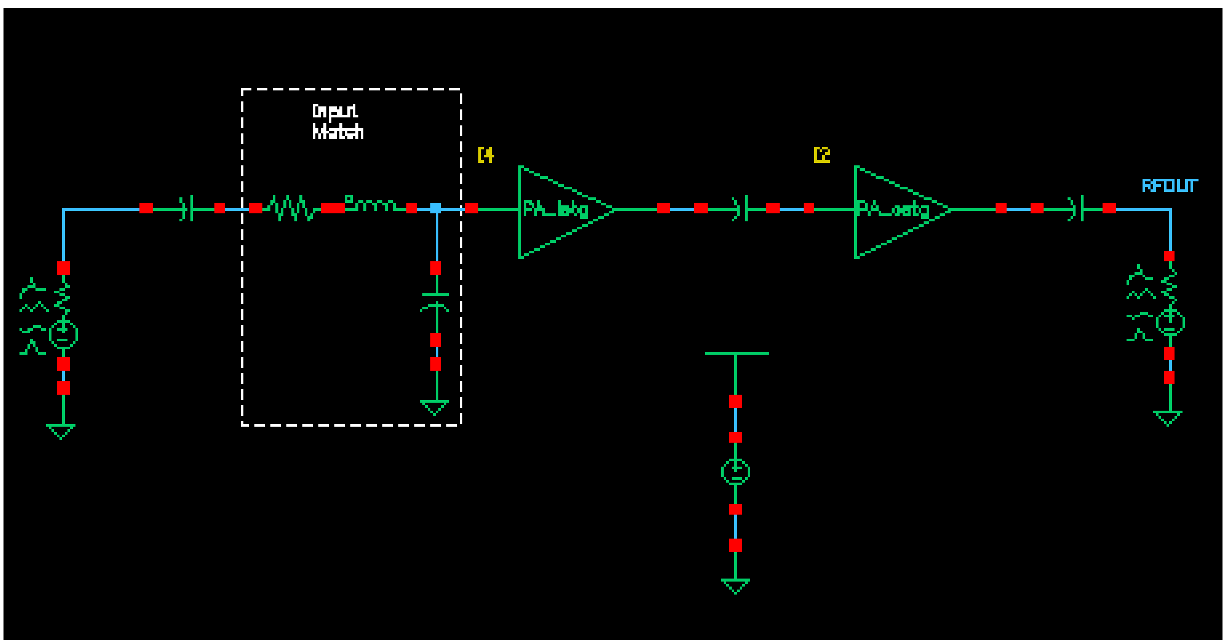
**10.** Choose *Window - Fit* to center the edited schematic in the window.

***Wire the Schematic.***

1. In the Schematic window, connect *PORT3* to Capacitor *C1*.
   a. In the Schematic window, choose *Add - Wire (Narrow)*.
   b. Click the terminal on *PORT 3* then click the terminal on *C1*.
   c. Press the Esc key to stop wiring.
2. Choose *Window -- Fit* to center the edited schematic in the window.

   The edited schematic looks like the one in Figure 8-25.

**Figure 8-25 The Edited EF_AMP Schematic**




***Edit CDF Properties for PORT3 and RFOUT***

In this section, you edit CDF properties for both *PORT 3* and *RF_OUT*.

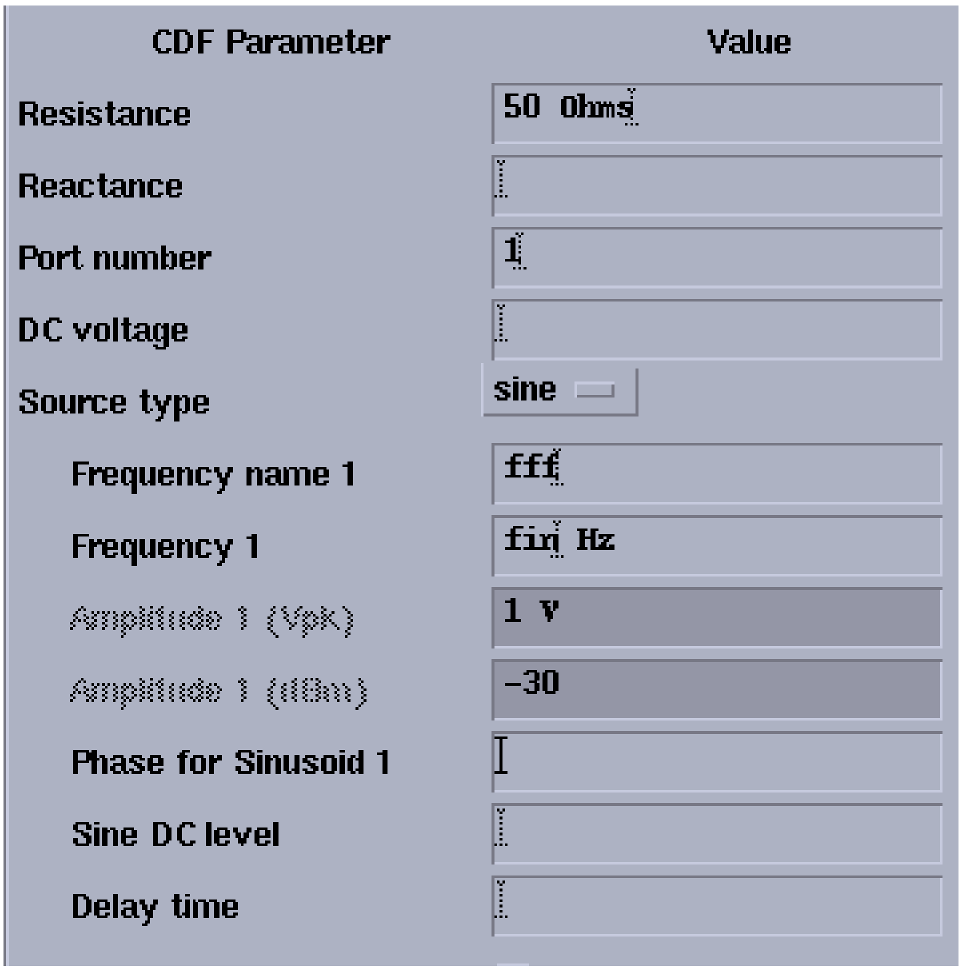
1. In the Schematic window, select *PORT 3*.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties form appears with information for *PORT 3* displayed.

**3.** In the Edit Object Properties form, do the following.

**a.** Type *fin* for *Frequency 1*.

The completed form looks like this.

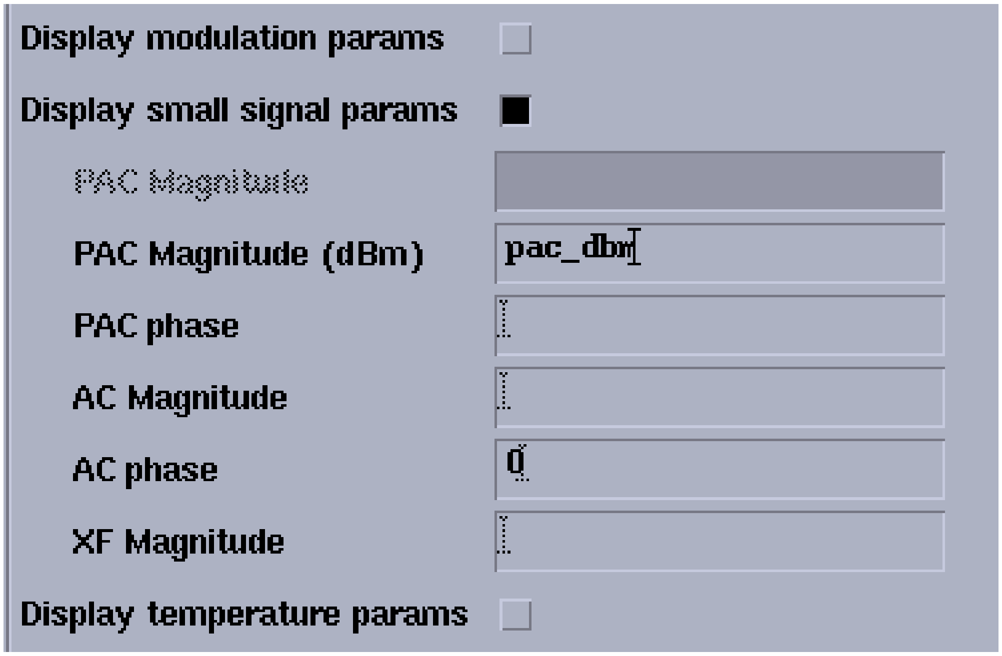



**b.** Highlight *Display small signal params*.

The small signal parameters section of the form opens up.

**c.** Type *pac_dbm* for *PAC Magnitude (dBm)*.

The completed form looks like this.

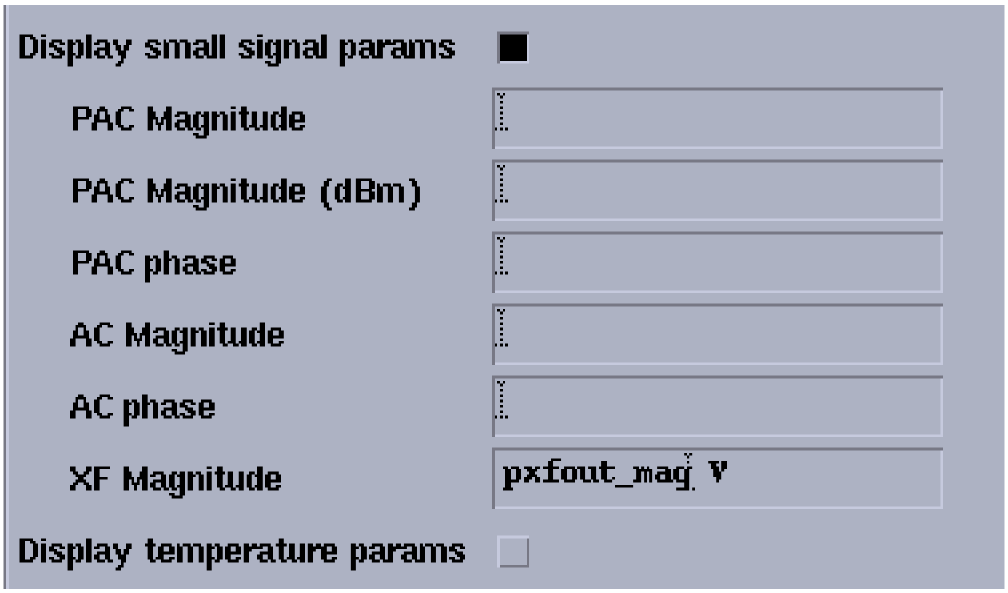



4. Click *Apply* in the Edit Object Properties form.
5. In the Schematic window, select *RF_OUT*.

   The Edit Object Properties form changes to display data for *RF_OUT*.
6. In the Edit Object Properties form, do the following.
   a. Highlight *Display small signal params*.

      The small signal parameters section of the form opens up.
   b. Type *pxfout_mag* for *XF Magnitude*.

The completed form looks like this.




**7.** Click *OK* in the Edit Object Properties form.

**8.** In the Schematic window, choose *Design - Check and Save*.

## Setting Up the EF_AMP Circuit

Before you start, perform the setup procedures described in Chapter 3.

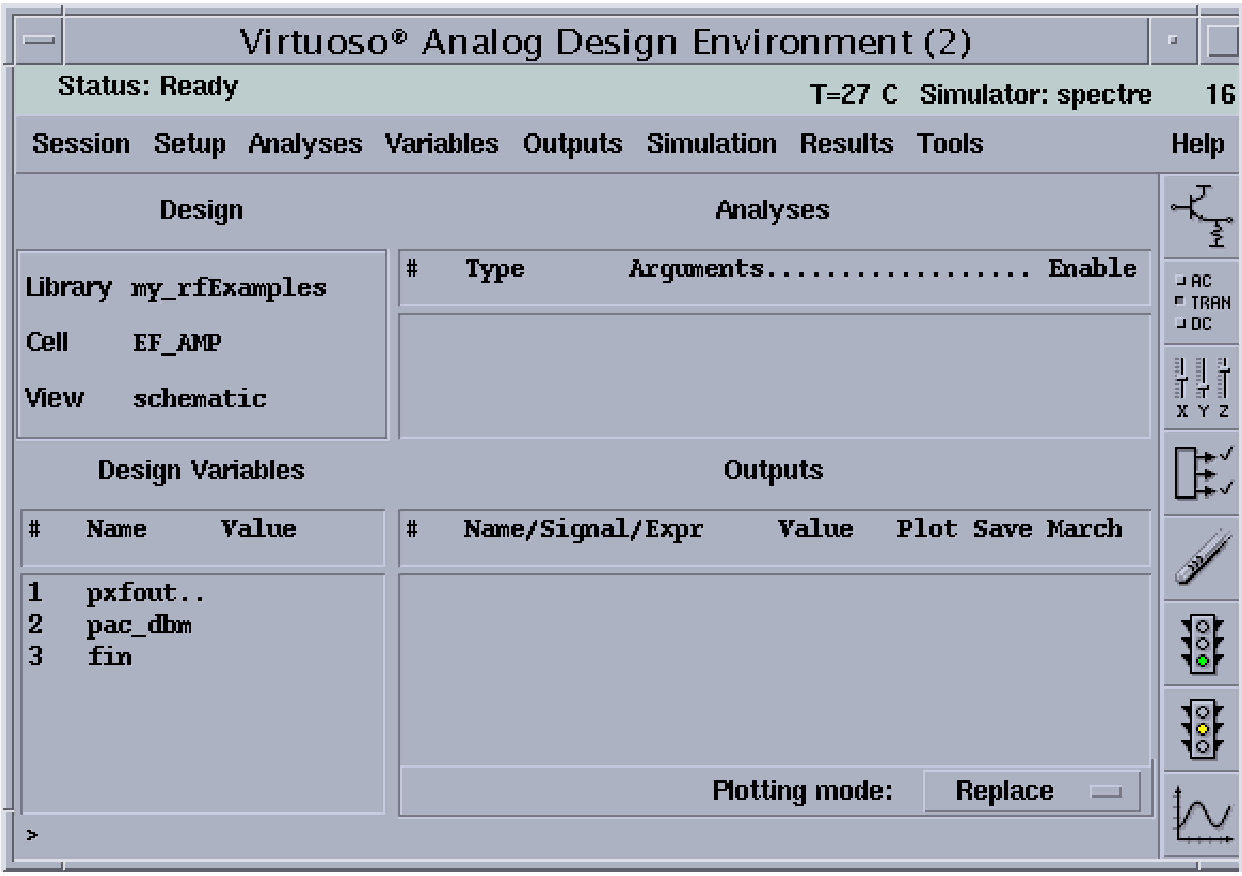
**1.** Set up and run the necessary PSS, modulated PAC and modulated PXF analyses on the *EF_AMP* schematic.

**2.** Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

### Opening the Simulation Window for the EF_AMP Circuit

**1.** In the EF_AMP Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




2. In the ADE window, choose *Variables – Copy From Cellview*.

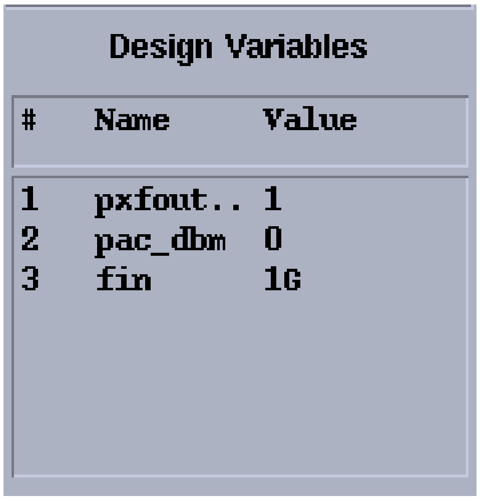
   The variable names display in the Design Variables area of the ADE window.

## Edit the Variable Values

➤ Following the directions in Chapter 4, “Setting Up for the Examples” edit the variables to have the following values.

| | |
|---|---|
| *pxfout_mag* | `1` |
| *pac_dbm* | `0` |
| *fin* | `1G` |

Check the Design Variables area in the ADE window to display the variable values.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *EF_example* in the Choosing Design form.

### Setting up the Model Libraries for the EF_AMP Circuit

1. In the ADE window, choose *Setup - Model Libraries.*

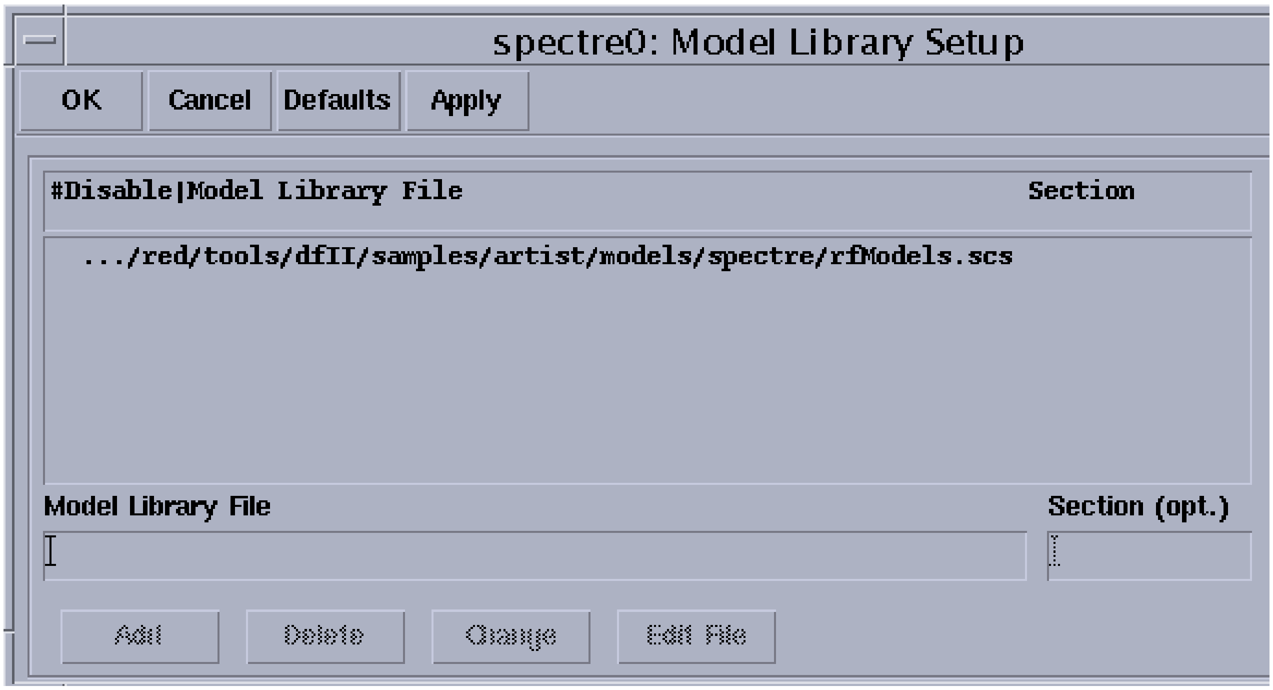
   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name. *`CDSHOME`* is the installation directory for the Cadence software.

   ```
   <CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs
   ```

3. Click *Add*.

The Model Library Setup form looks like the following.

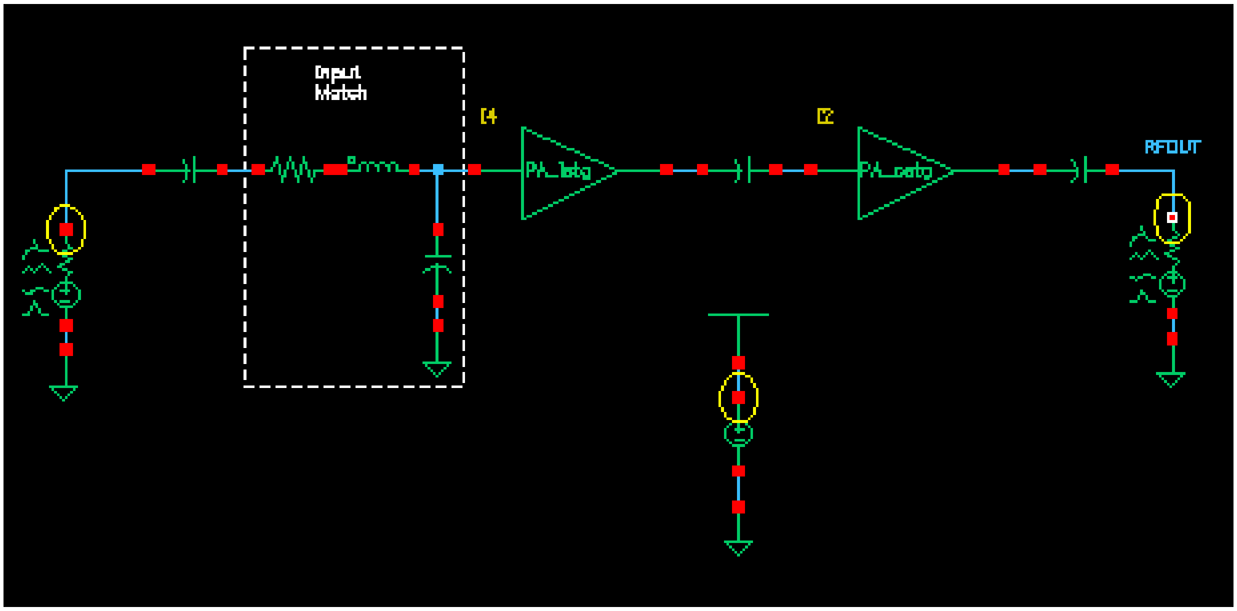



4. Click *OK*.

## Selecting Outputs To Save

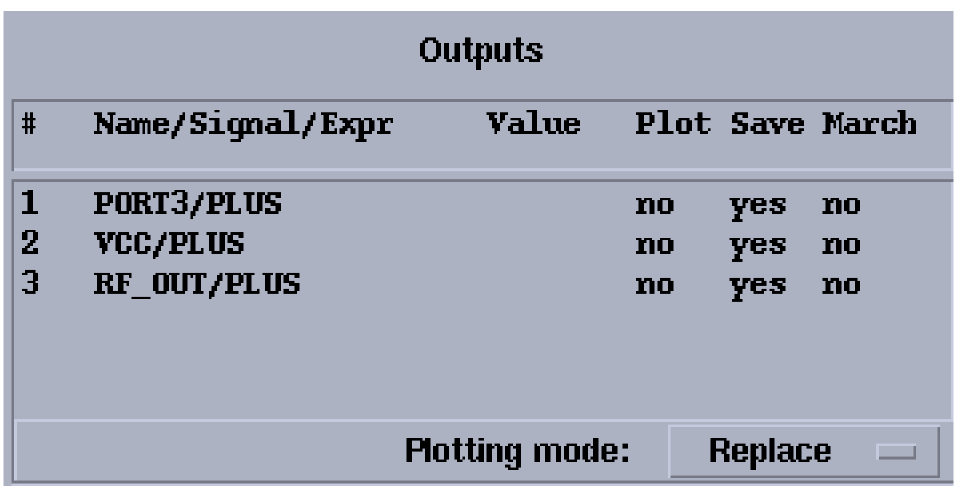
1. In the ADE window, choose *Outputs – To Be Saved – Select on Schematic*.
2. In the Schematic window, click the terminals that are circled in Figure 8-26.

**Figure 8-26  Outputs to Save in the EF_AMP Schematic**



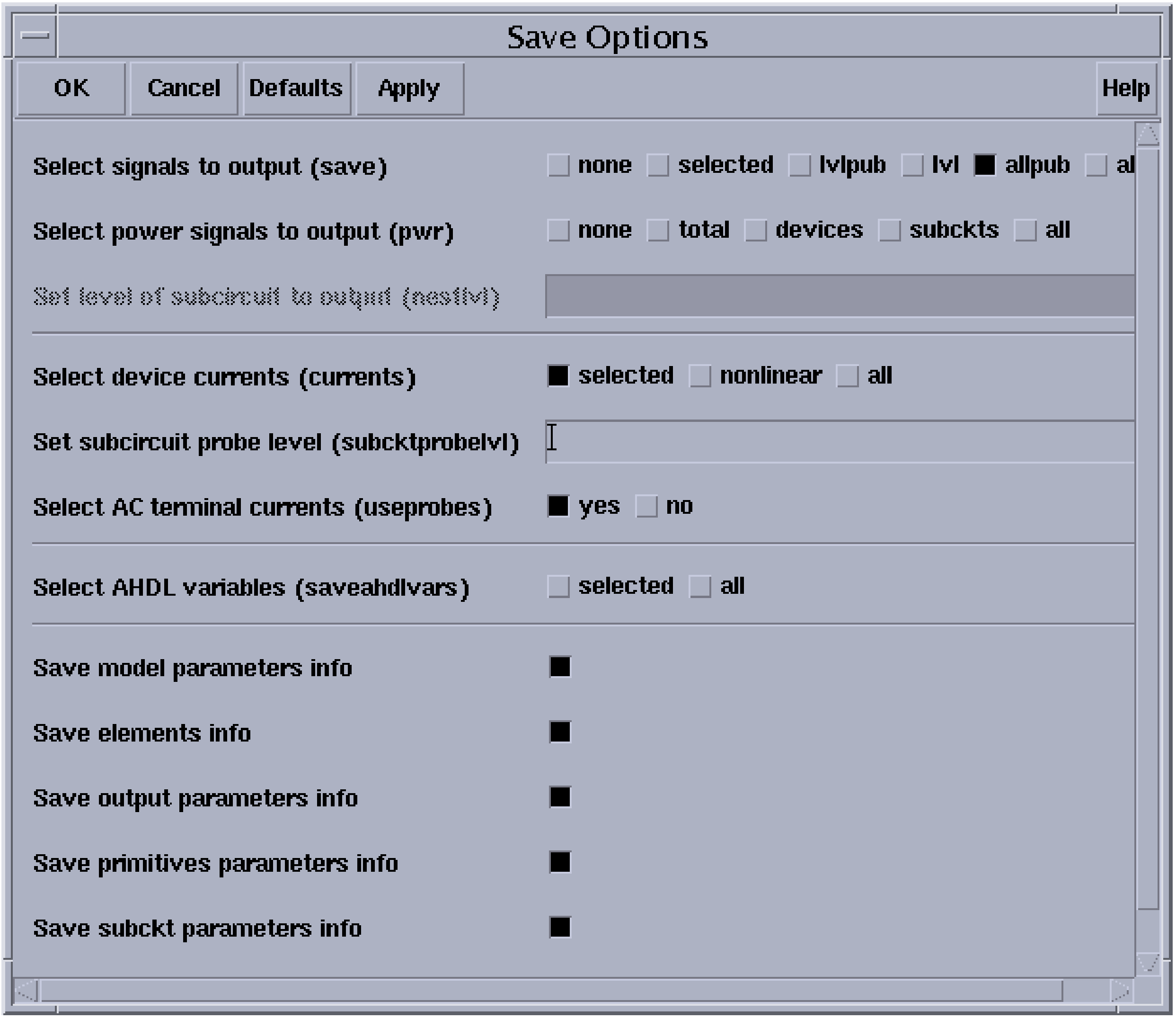

After you click a terminal, it is circled in the schematic. The selected outputs are also displayed in the Schematic window Outputs area as shown in Figure 8-27.

**Figure 8-27  Outputs Area in Simulation Window**




**3.** In the ADE window, choose *Outputs-Save All*.

**4.** The Save Options form displays.




**5.** Set the following values

| | |
|---|---|
| *save* | *allpub* |
| *currents* | *selected* |
| *useprobes* | *yes* |

**6.** Click *OK*.

## Setting Up and Running the PSS, PAC Modulated, and PXF Modulated Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

### Setting up the PSS Analysis

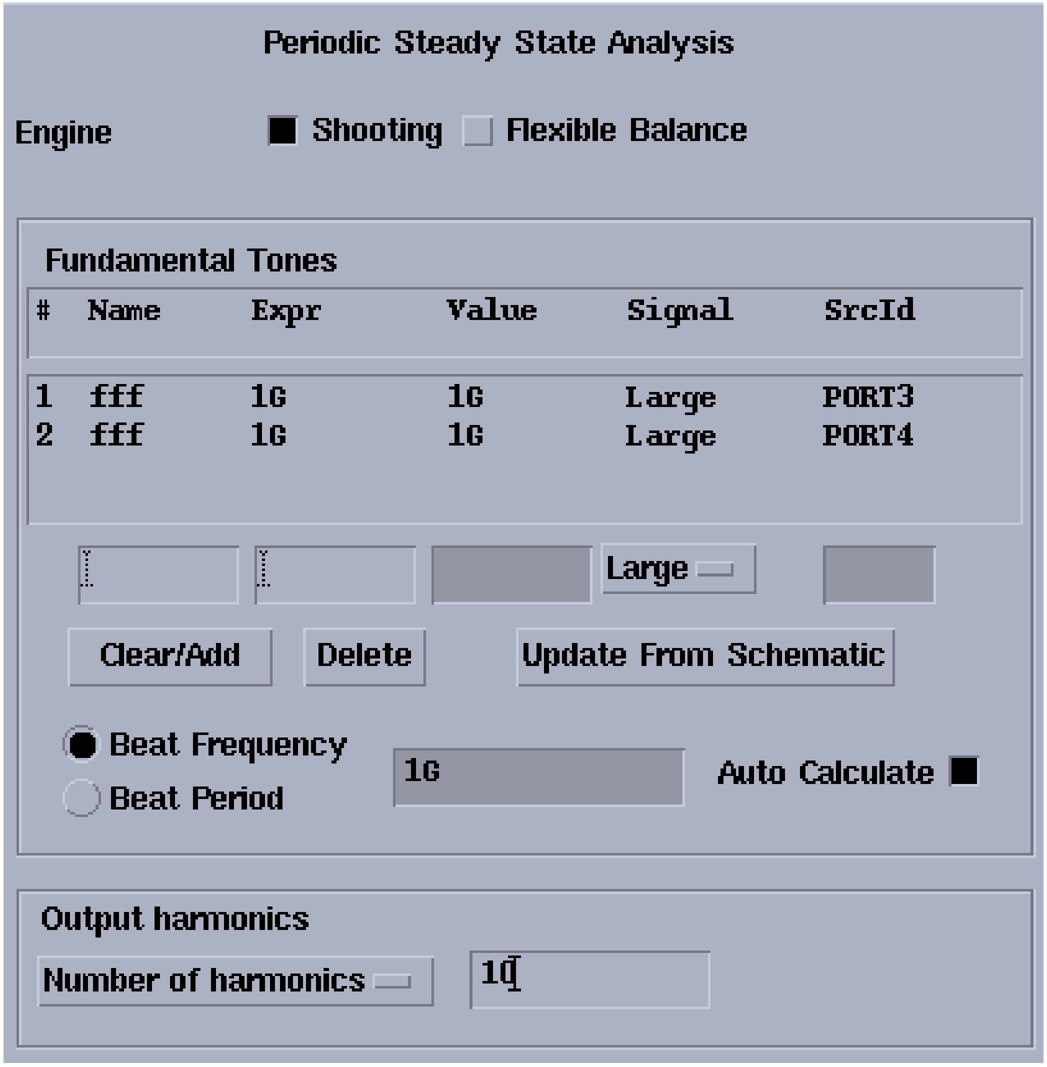
1. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

2. Highlight the *Auto Calculate* button to automatically calculate and enter the value `1G` in the *Beat Frequency* field.

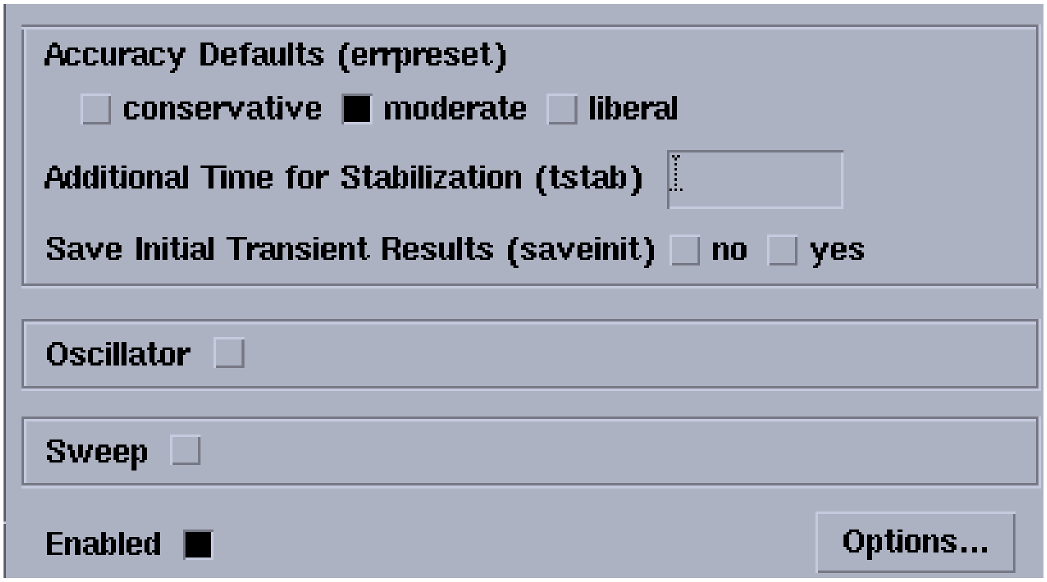
3. Choose *Number of harmonics* for *Output harmonics* and type `10` in the adjacent field.

The top of the PSS analysis form looks like this.

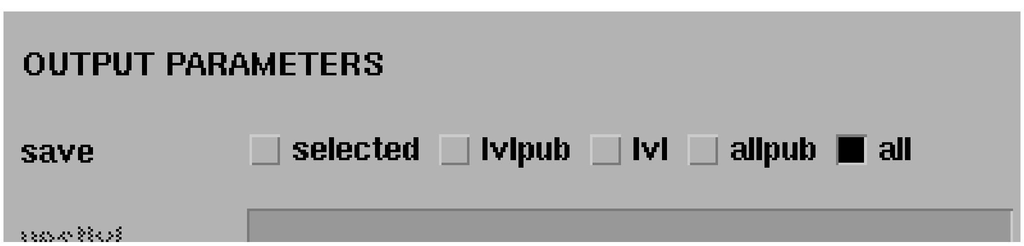



4. Highlight *moderate* for Accuracy Defaults *(errpreset)*.
5. Verify that *Enabled* is highlighted in the PSS Choosing Analyses form.

The bottom of the PSS Choosing Analyses form looks like this.




6. Click the *Options* button to display the Options form for the PSS analysis.
7. In the Options form, scroll to the *Output Parameters* section, highlight *all* for *save.* The *Output Parameters* section of the Options form looks like the following.




8. In the PSS Options form, click *OK*.
9. In the PSS Analysis form, click *Apply*.

### Setting up the PAC Modulated Analysis

1. In the Choosing Analyses form, highlight *pac*.

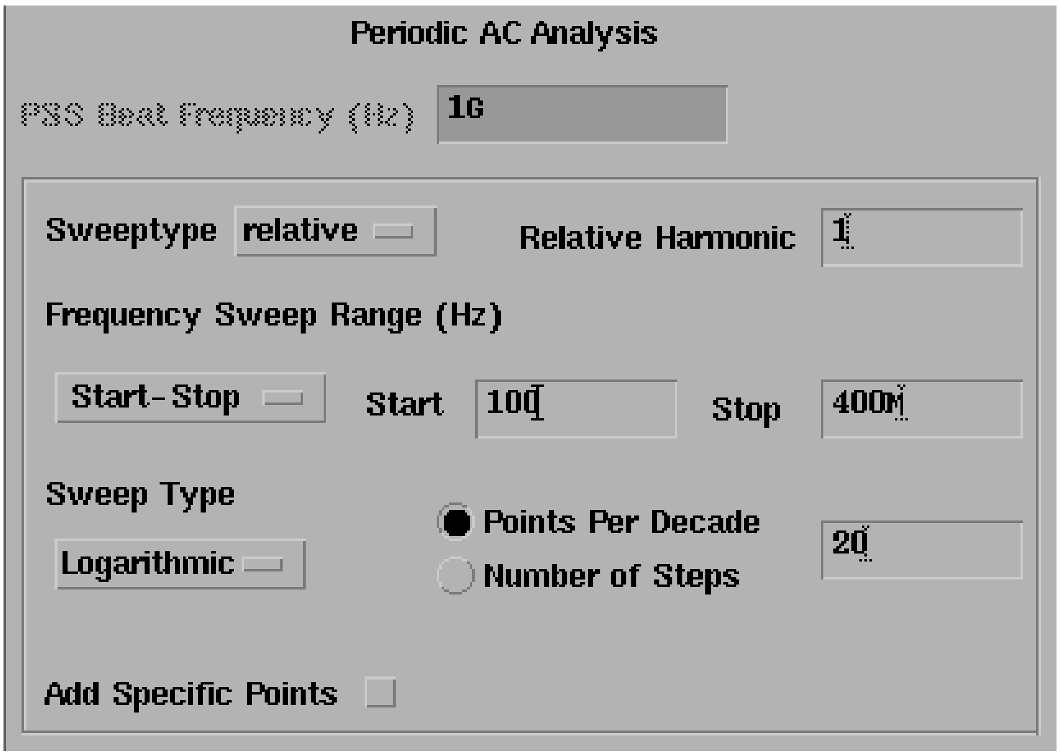
   The form changes to display options for PAC simulation.

2. For *Sweeptype*, select *Relative* in the cyclic field and type 1 in the *Relative Harmonic* field.

3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

   The top of the PAC analysis form looks like this.

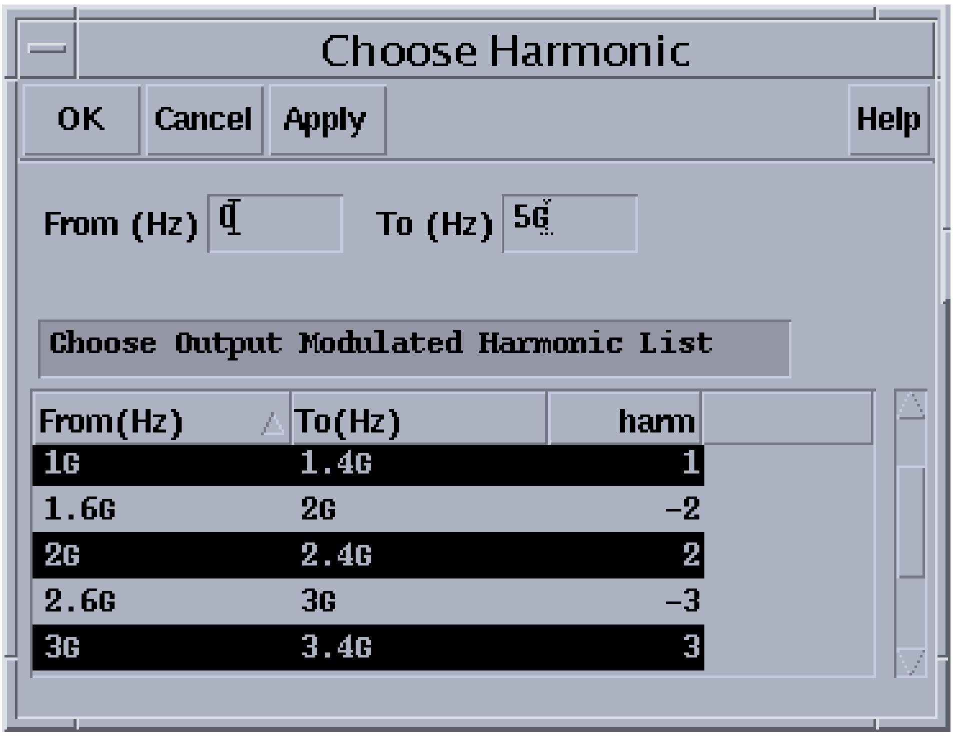



5. For *Sidebands*, select *Maximum sideband* in the cyclic field. Type `3` in the field.

6. In the *Specialized Analyses* cyclic field select *Modulated* to open the modulated analysis section of the PAC Choosing Analyses form.

7. For *Input Type* choose *SSB/AM/PM* in the cyclic field.

**8.** For *Output Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

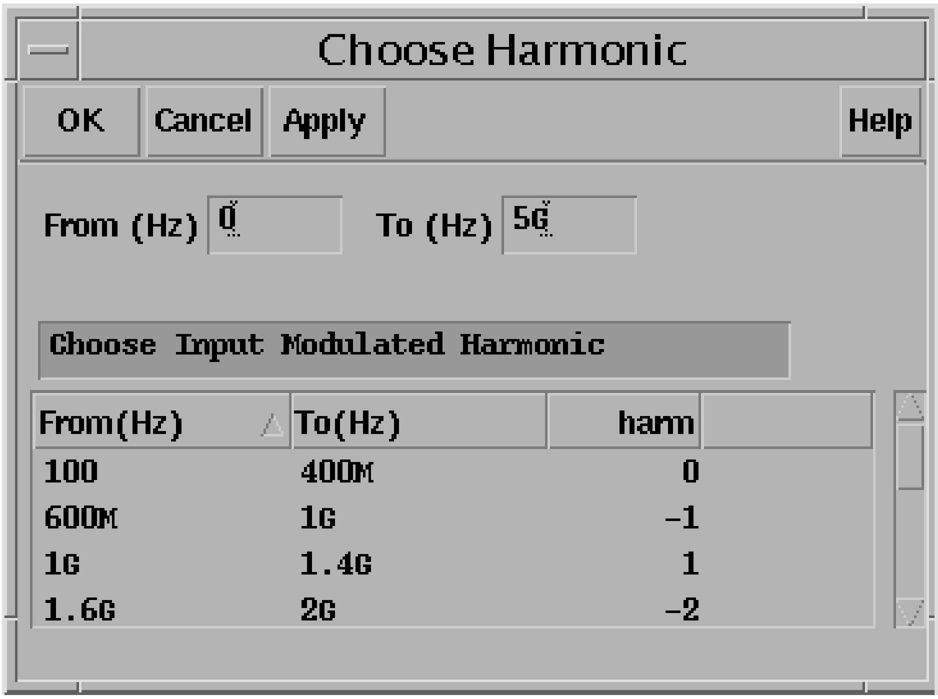



**9.** Scroll the list of harmonics and select harmonics with indexes 1, 2 and 3. Click to highlight the first harmonic. Press and hold down the *Control* key while you click to select harmonics 2 and 3. Then click OK.

`1`, `2`, and `3` display in the *Output Modulated Harmonic List* field.

You can also simply type the values, separated by spaces, in the *Output Modulated Harmonic List* field.

**10.** For *Input Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.



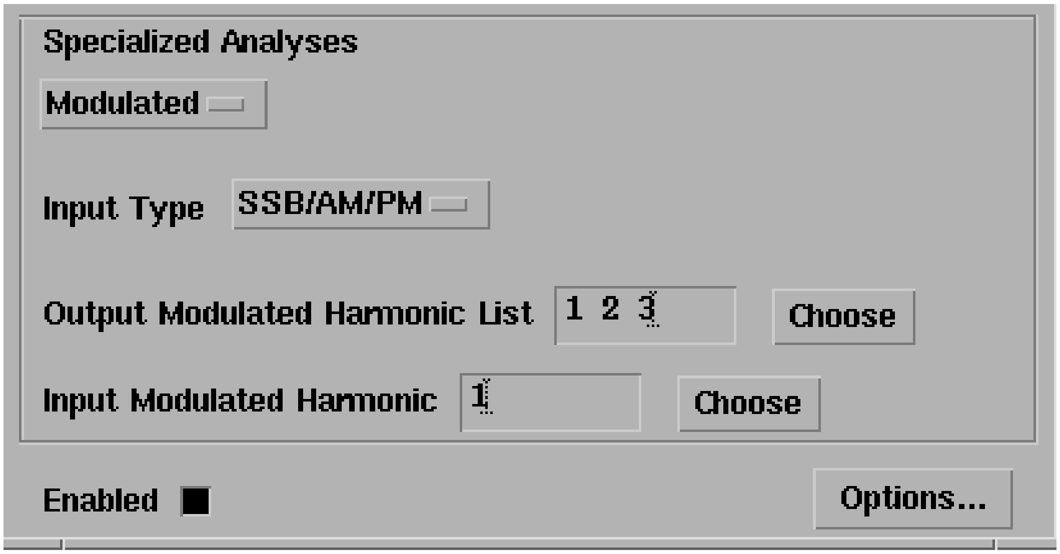

**11.** Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click *OK*.

`1` displays in the *Input Modulated Harmonic* field.

You can also simply type the value in the *Input Modulated Harmonic* field.

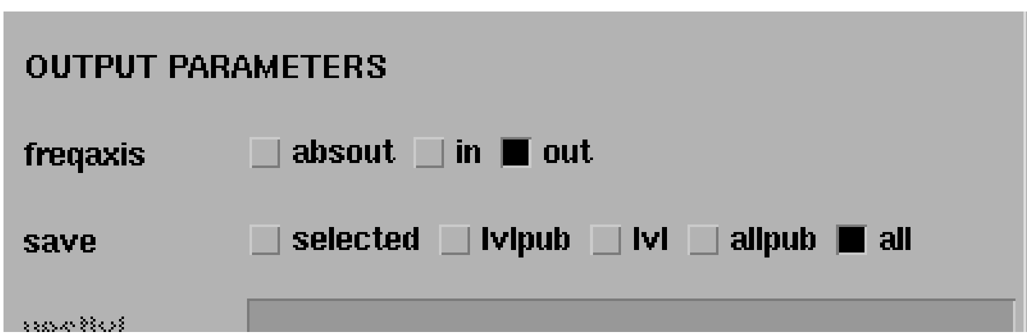
**12.** Verify that *Enabled* is highlighted in the PAC Choosing Analyses form and click *Apply.*

The bottom of the PAC analysis form looks like this.




13. Click the *Options* button to display the Options form for the PAC analysis.
14. In the *Output Parameters* section of the Options form, highlight, *out* for *freqaxis* and *all* for *save.*

    The Output *Parameters* section of the Options form looks like the following.




15. In the PAC Options form, click *OK*.
16. In the PAC Analysis form, click *Apply*.

### *Setting up the PXF Modulated Analysis*

1. In the Choosing Analyses form, highlight *pxf*.

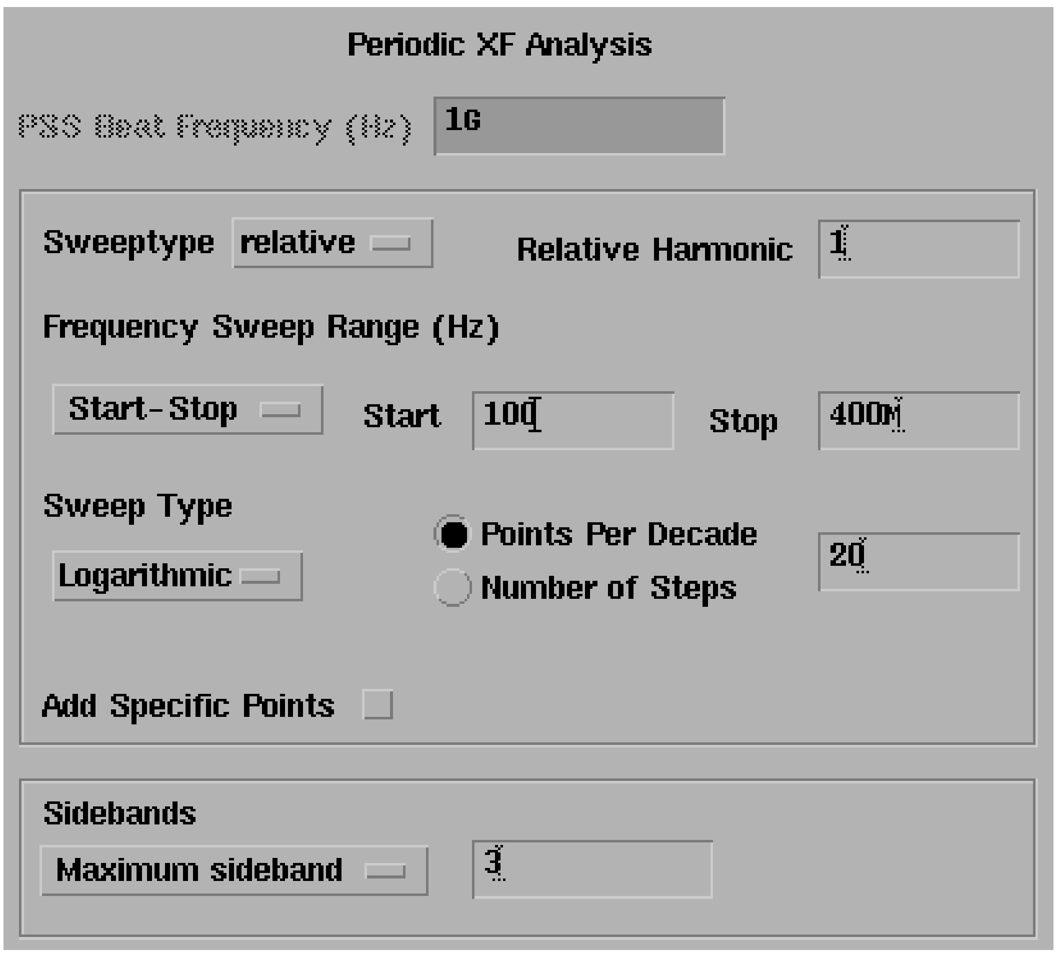
   The form changes to display options for PXF analysis.

2. For *Sweeptype*, select *relative* in the cyclic field and type `1` in the *Relative Harmonic* field.

3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

   The top of the PXF analysis form looks like this.




5. For *Sidebands*, select *Maximum sideband* in the cyclic field and type 3 in the field.

6. For *Output*, highlight *probe*. Click *Select* and follow the prompt at the bottom of the Schematic window

   ```
   Select output probe instance...
   ```

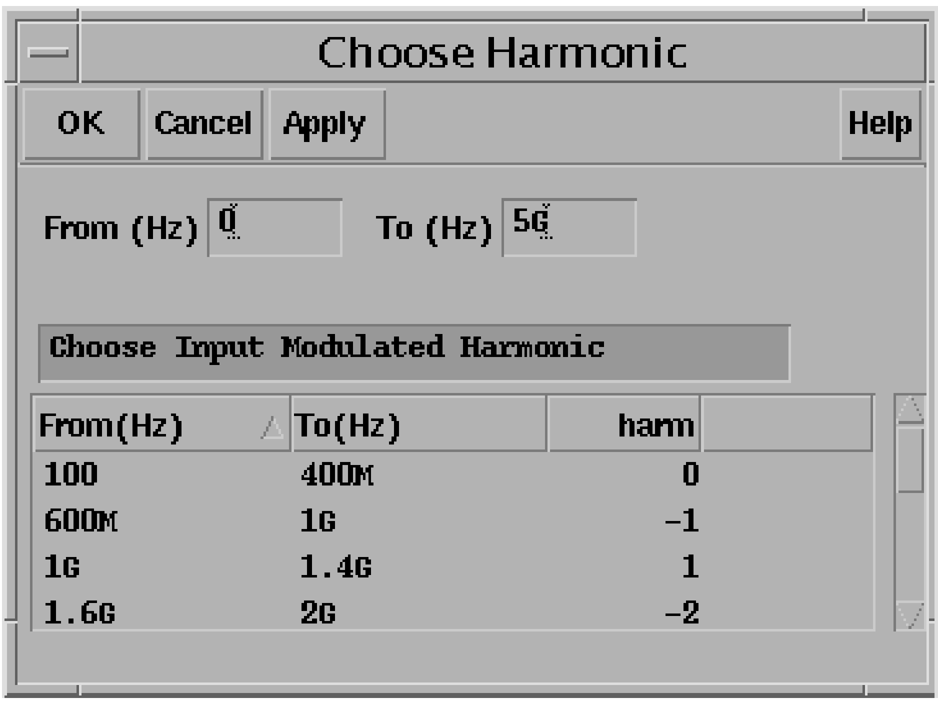
   and select the RF port on the right side of the schematic.

`/RF_OUT` displays in the *Output Probe Instance* field.

**7.** Highlight *Modulated Analysis* to open the modulated analysis section of the PXF Choosing Analyses form.

**8.** For *Output Type* choose *SSB/AM/PM* in the cyclic field.

**9.** For *Input Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

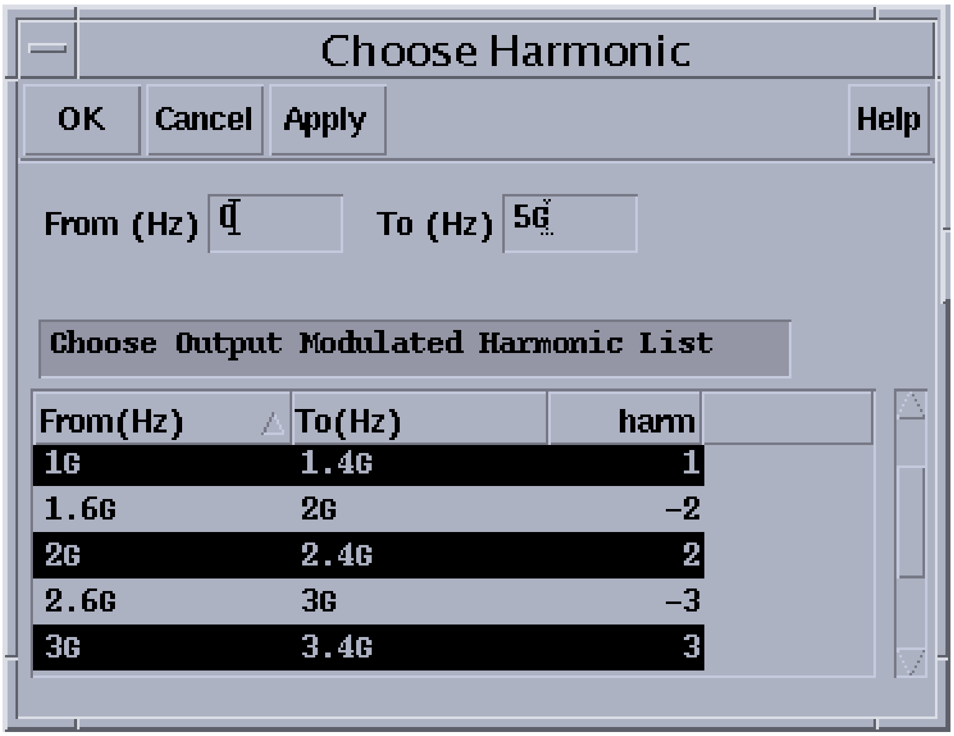



**10.** Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

`1` displays in the *Input Modulated Harmonic List* field.

You can also simply type the value in the *Input Modulated Harmonic List* field.

**11.** For *Output Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.

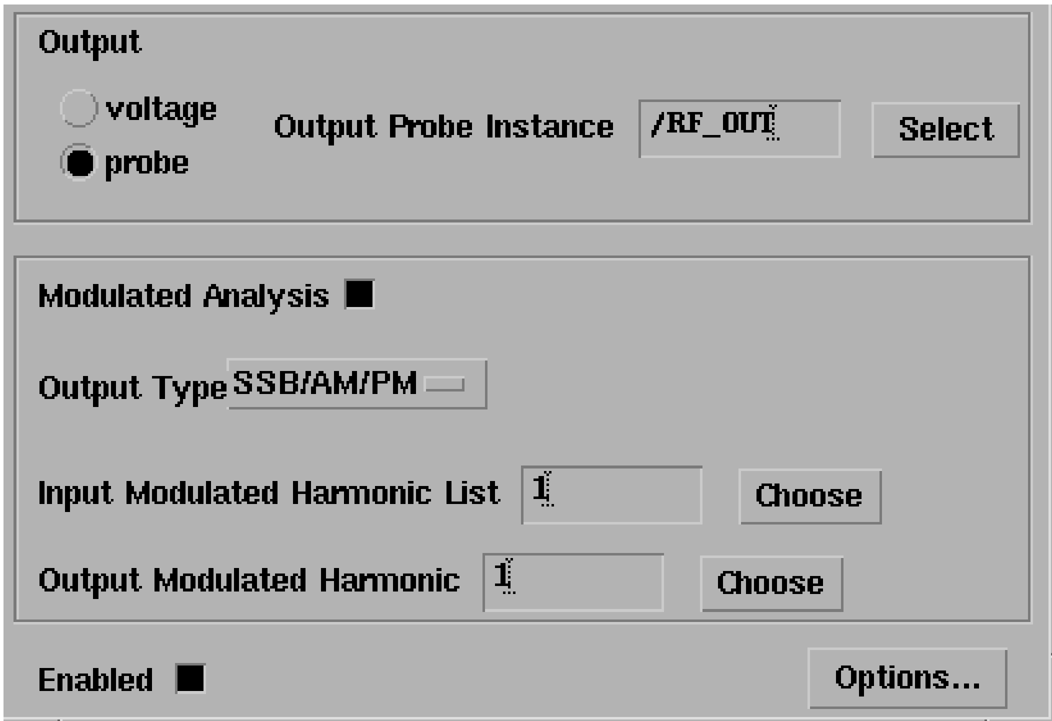



**12.** Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

`1` displays in the *Output Modulated Harmonic* field.

You can also simply type the value in the *Input Modulated Harmonic* field.

The bottom of the PXF analysis form looks like this.

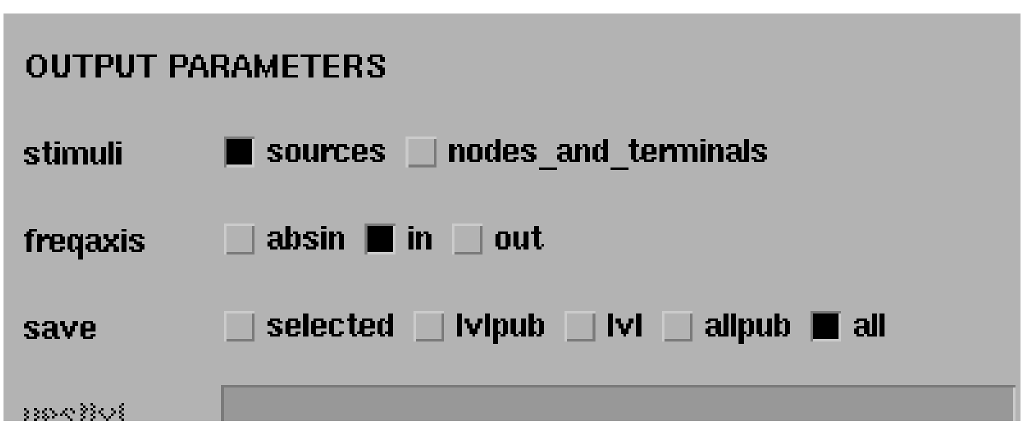



**13.** Verify that *Enabled* is highlighted in the PXF Choosing Analyses form.

**14.** Click the *Options* button to display the Options form for the PXF analysis.

**15.** In the *Output Parameters* section of the Options form *in* for *freqaxis* and *all* for *save. sources* is already highlighted for *stimuli* because it is the default.

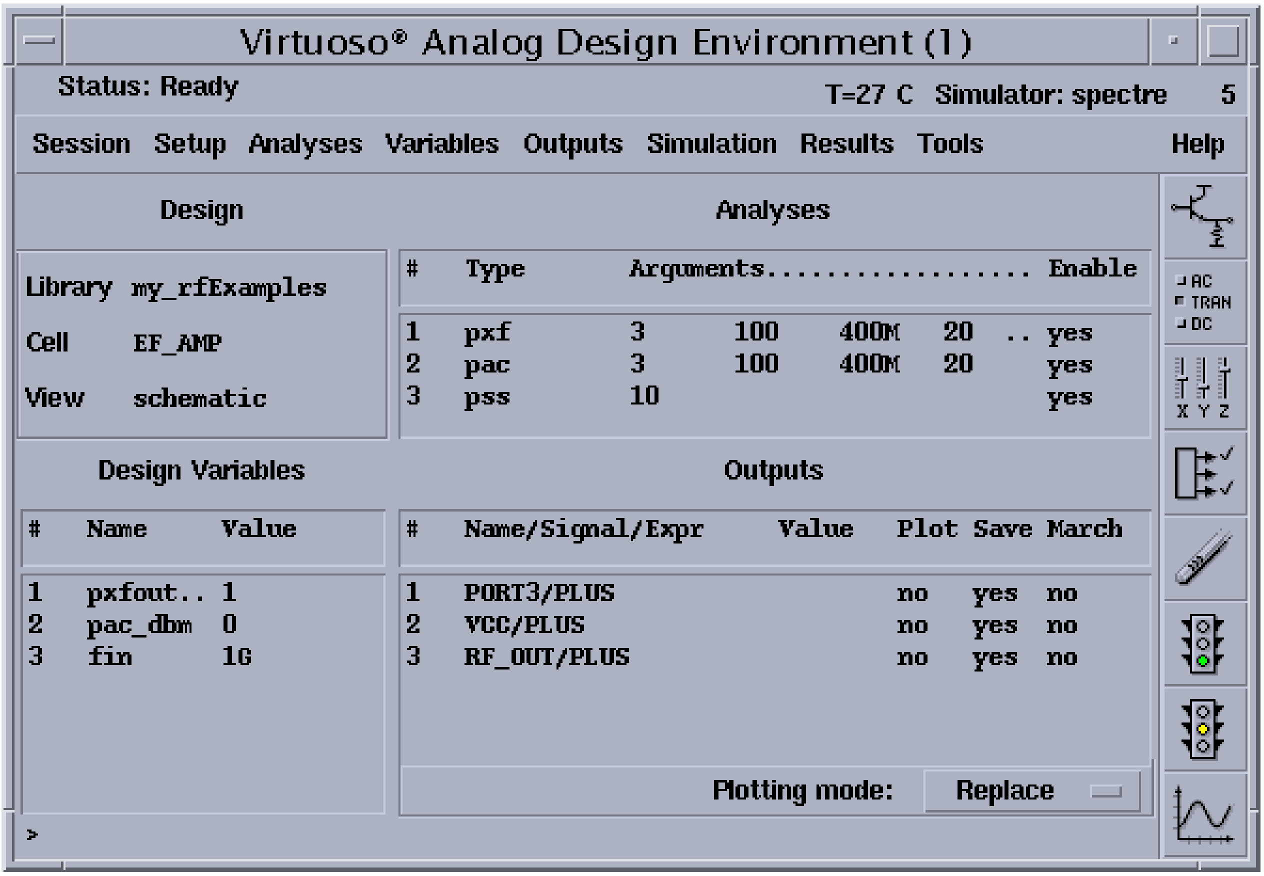
The Output *Parameters* section of the Options form looks like the following.




**16.** Click OK in the PXF Options form.

**17.** Click OK in the PXF Choosing Analyses form.

The Choosing Analyses form closes. The ADE window displays the analysis information you have set up.




## Running the Simulations

1. To run the simulations, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting and Calculating PAC Modulated Results

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears

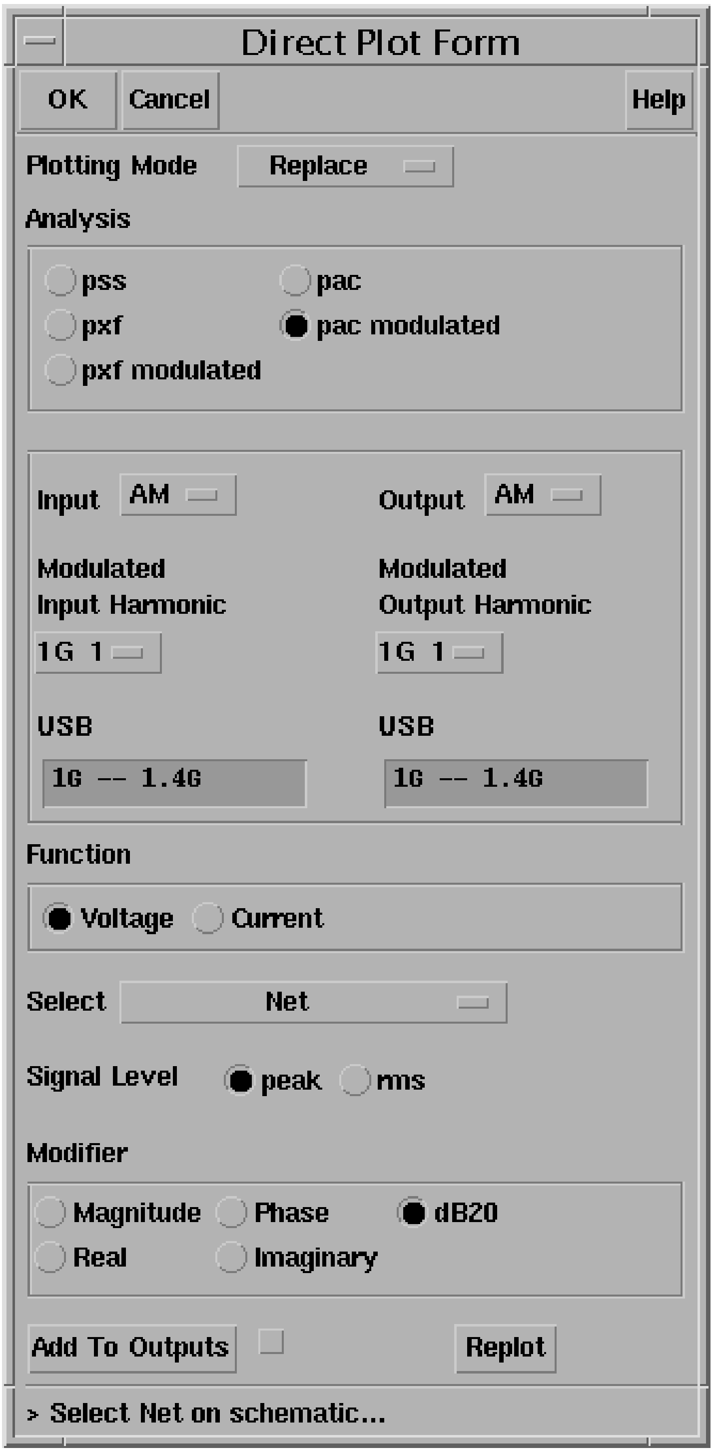
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *pac modulated* for *Analysis*.

   The form changes to reflect the modulated PAC analysis.
4. In the *Input* cyclic field, select *AM*.
5. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

   The *USB* field displays `1G -- 1.4G`.
6. In the *Output* cyclic field, select *AM*.
7. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.
8. Highlight *Voltage* for *Function*.
9. Highlight *peak* for *Signal Level*.
10. Highlight *dB20* for *Modifier*.

**11.** The Direct Plot form displays as follows.

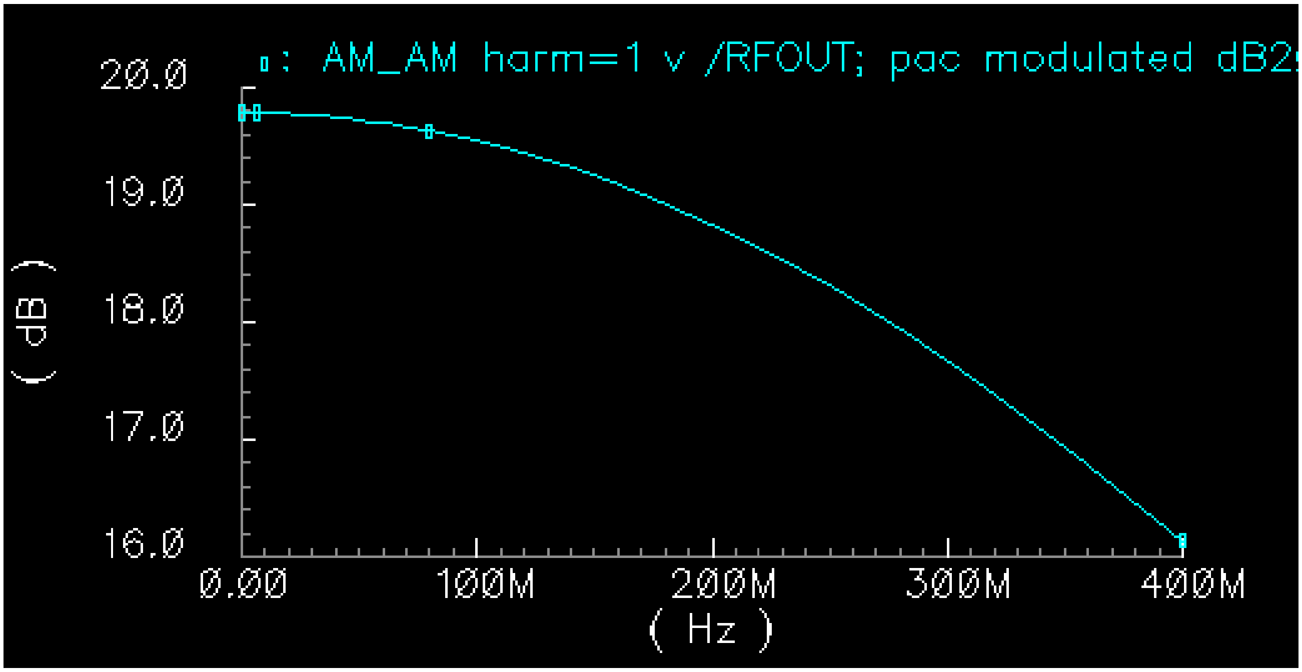



**12.** Select *Net* in the *Select* cyclic field. Follow the prompt at the bottom of the form

`Select Net on schematic`

Select the net labeled *RFOUT* on the right side of the schematic.

The waveform looks as follows.

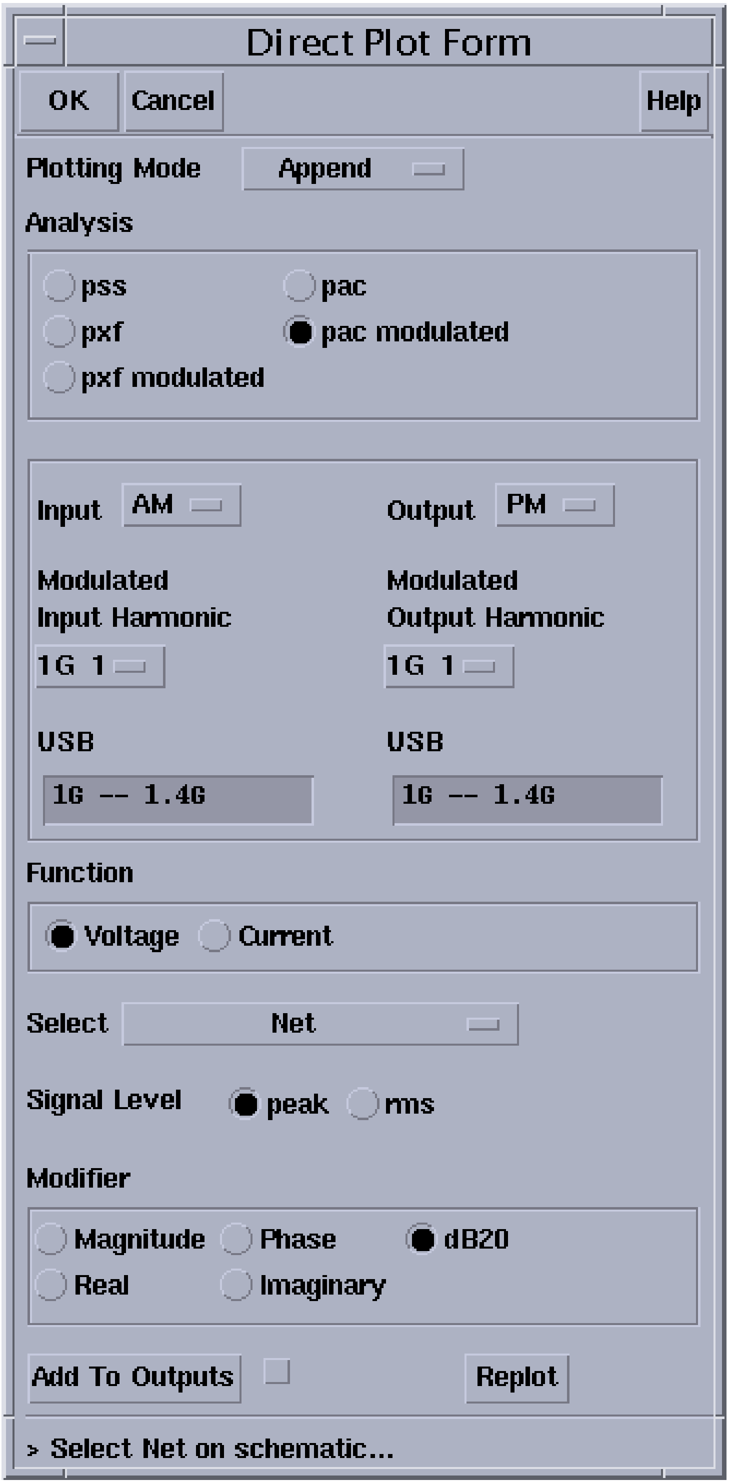



To add AM/PM to the plot, make the following changes in the Direct Plot form.

1. Change Plot Mode to Append.
2. Change the *Output* cyclic field to *PM*.
3. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.

The Direct Plot form changes as follows.

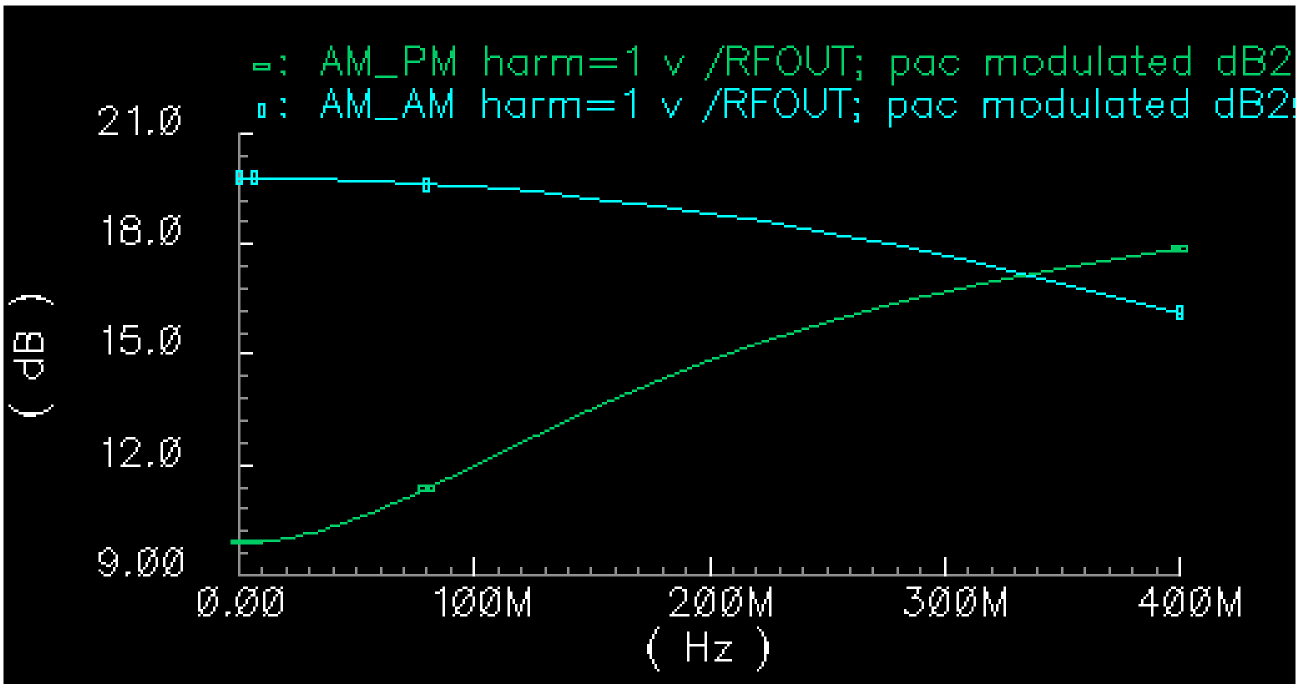


4. Follow the prompt at the bottom of the form

   `Select Net on schematic`

   Select the net labeled *RFOUT* again.

   The waveform looks as follows.




## Plotting and Calculating PXF Modulated Results

1. If necessary, choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *pxf modulated* for *Analysis*.
4. In the *Input* cyclic field, select *AM*.
5. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

   The *USB* field displays `1G -- 1.4G`.

6. In the *Output* cyclic field, select *AM*.

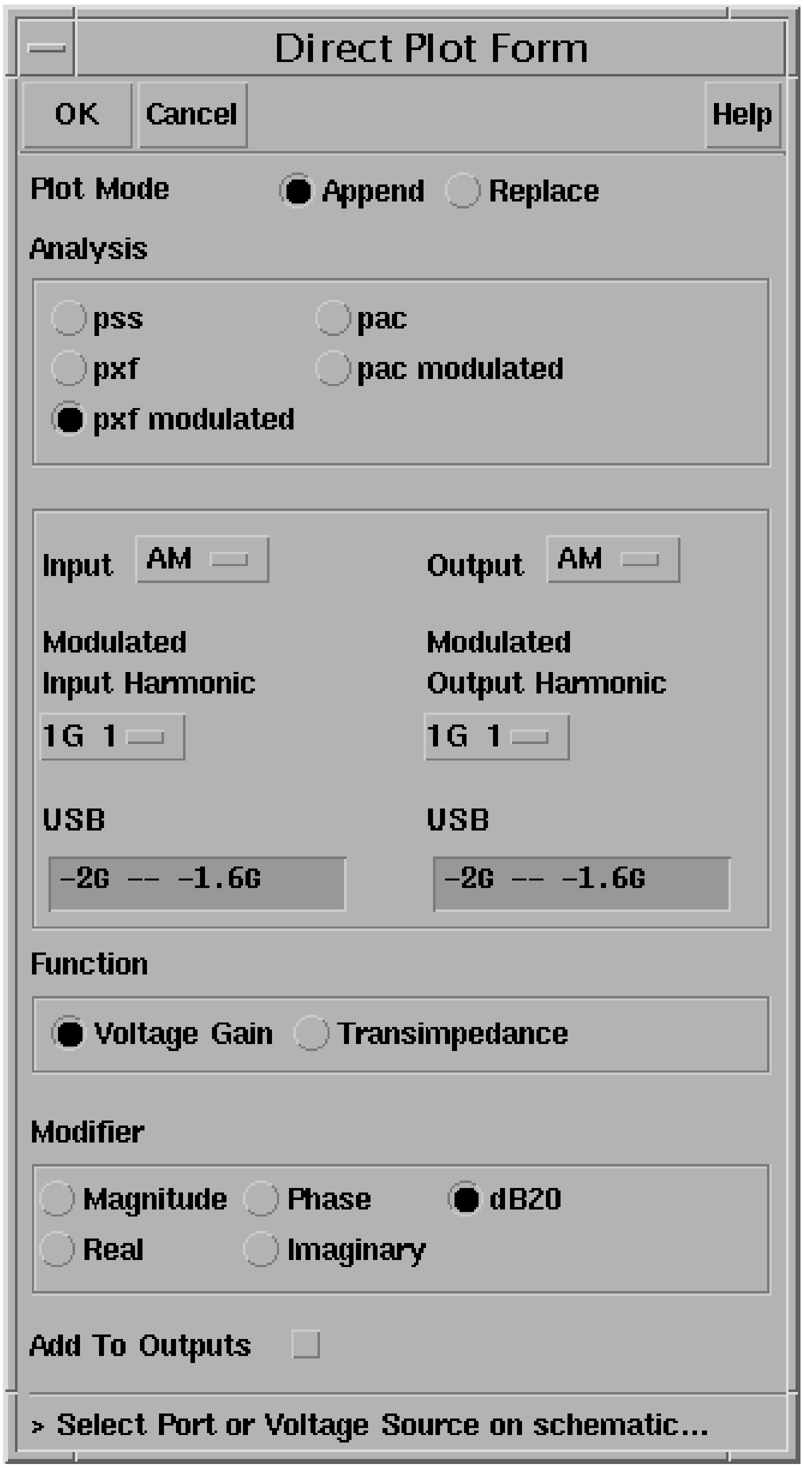
7. In the *Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.

8. Highlight *Voltage Gain* for *Function*.

9. Highlight *dB20* for *Modifier*.

**10.** The Direct Plot form displays as follows.

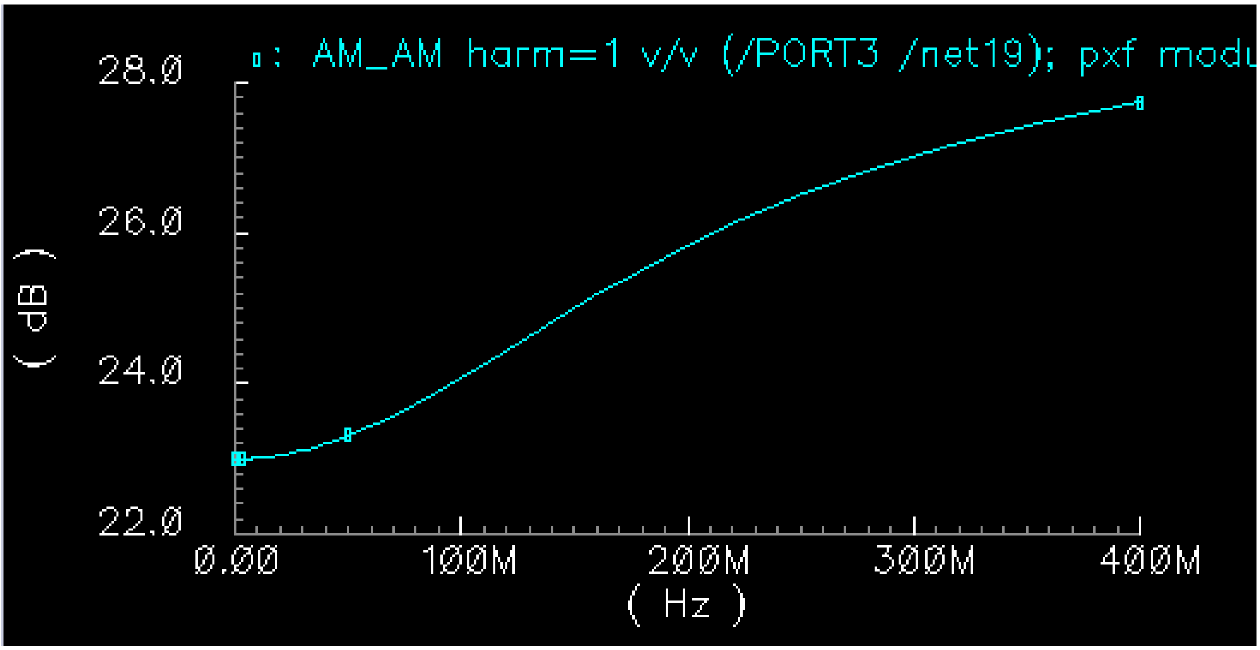



**11.** Follow the prompt at the bottom of the form

```
Select Port or voltage source on schematic
```

Select the input port labeled *PORT3* on the left side of the schematic.

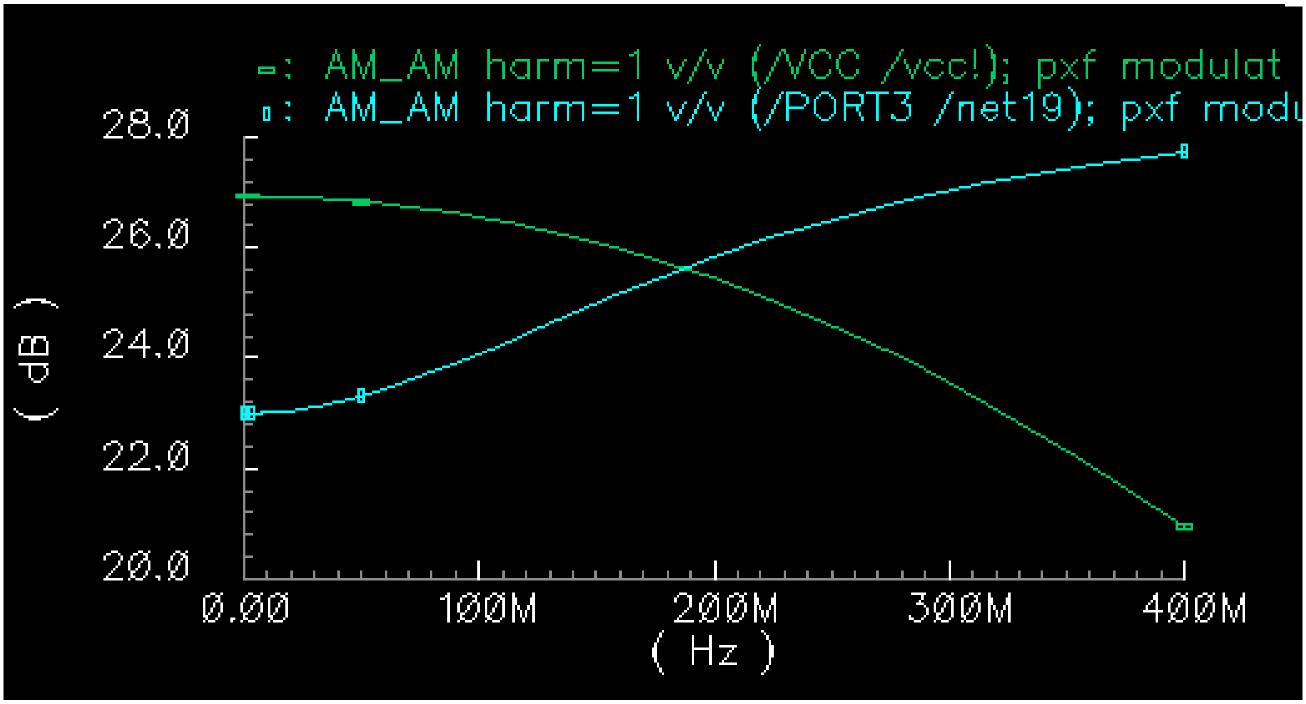
The waveform looks as follows.




To add the VCC voltage source to the plot,

1. Change Plot Mode to Append.
2. Select the voltage source labeled *VCC* in the lower center of the schematic.

The VCC voltage source is added to the plot.




This plot compares two different voltage sources from the schematic.

# Measuring Jitter with PSS and Pnoise Analyses

This section describes jitter measurement.

## Setting Up the EF_AMP Circuit

Before you start, perform the setup procedures described in Chapter 3.

- If you have a copy of the *EF_AMP* schematic in your *my_rfExamples* library, simply open it. If you do not have a copy of the *EF_AMP* schematic, See “Creating the EF_AMP Circuit” on page 645 for information on creating the EF_AMP circuit.
- Open the simulation window for the EF_AMP schematic as described in “Opening the Simulation Window for the EF_AMP Circuit” on page 653.
- Set up the model libraries as described in “Setting up the Model Libraries for the EF_AMP Circuit” on page 655.

- Select Outputs to save as described in “Selecting Outputs To Save” on page 656.
- Set up and run the necessary PSS and modulated Pnoise analyses on the *EF_AMP* schematic.
- Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

## Opening the EF_AMP Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

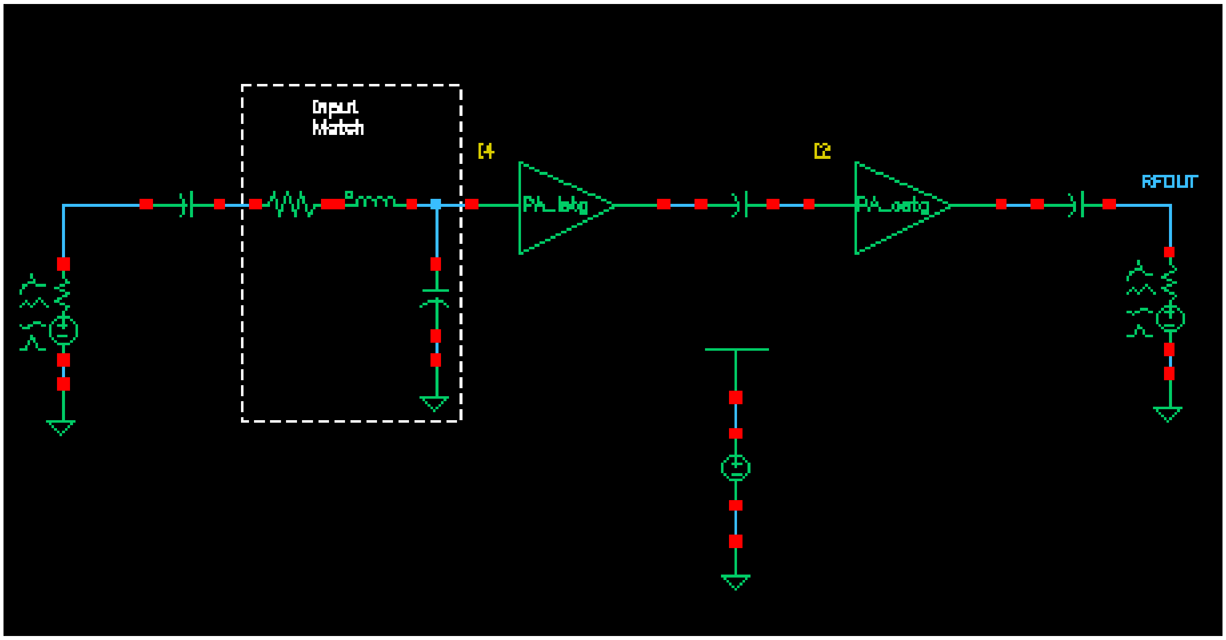
   b. In the *Cell Name* list box, highlight *EF_AMP*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

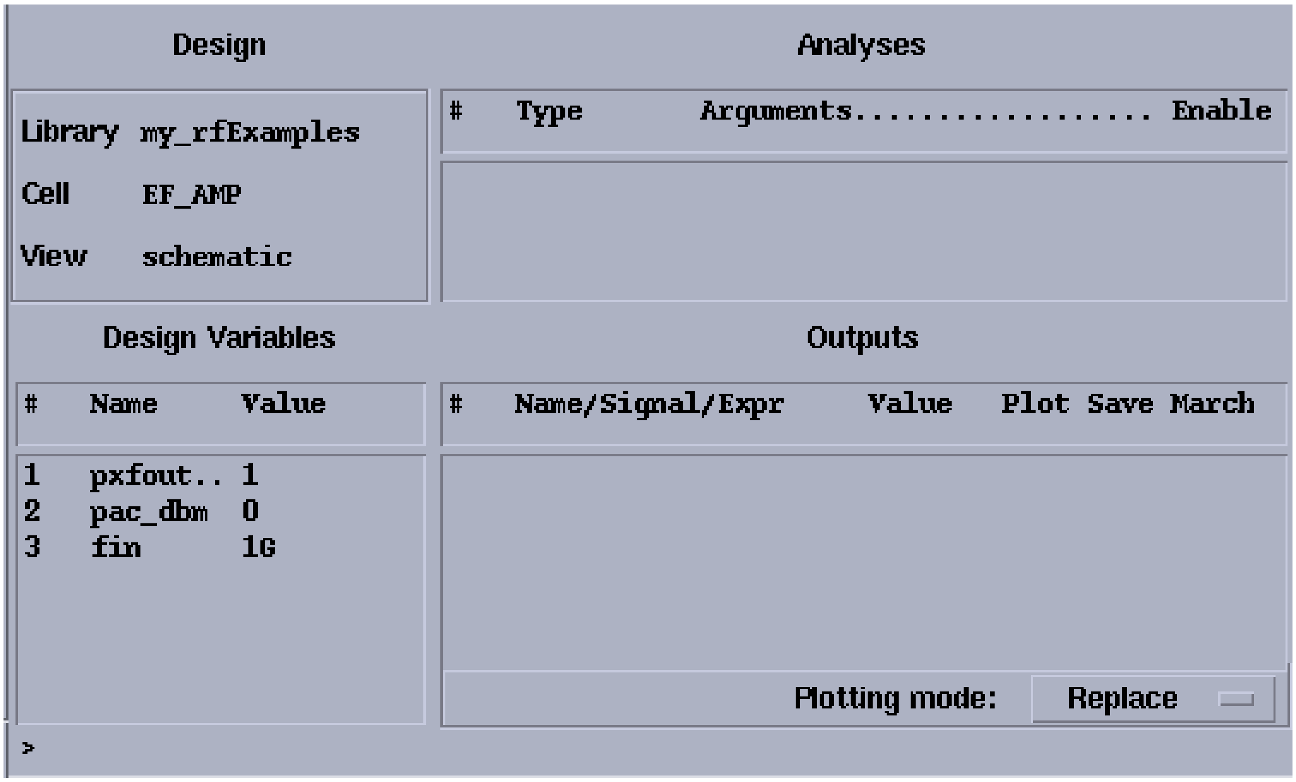
The Schematic window appears with the *EF_AMP* schematic.




## Opening the Simulation Window for the EF_AMP Circuit

**1.** In the EF_AMP Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




## Setting Up and Running the PSS and Pnoise Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

### Setting up the PSS Analysis

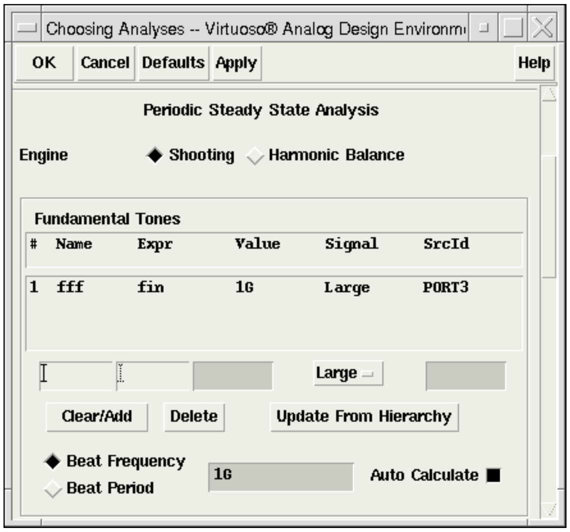
1. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

2. Highlight the *Auto Calculate* button to automatically calculate and enter the value `1G` in the *Beat Frequency* field.

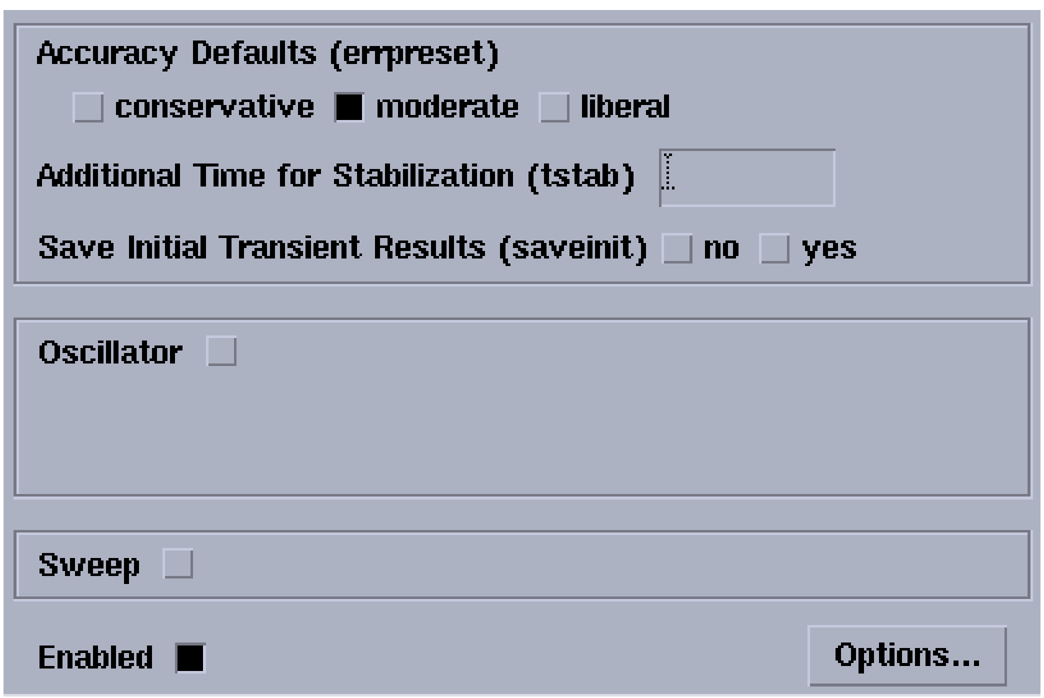
3. Choose *Number of harmonics* for *Output harmonics* and type `10` in the adjacent field. Output Harmonics are not critical for this analysis.

The top of the PSS analysis form looks like this.




4. Highlight *moderate* for Accuracy Defaults *(errpreset)*.
5. Verify that *Enabled* is highlighted in the PSS Choosing Analyses form.

   The bottom of the PSS Choosing Analyses form looks like this.




6. Click the *Options* button to display the Options form for the PSS analysis.

7. In the Options form, scroll to the *Output Parameters* section, highlight *all* for *save.* The *Output Parameters* section of the Options form looks like the following.


8. In the PSS Options form, click *OK*.
9. In the PSS Analysis form, click *Apply*.

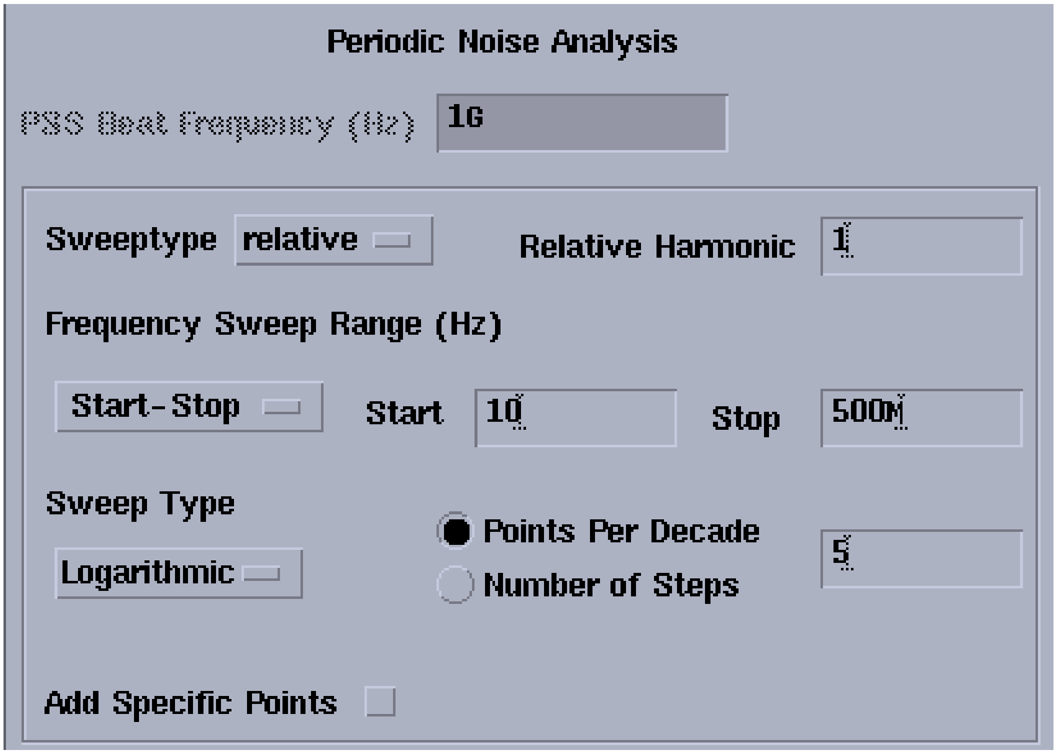
### Setting up the Pnoise Jitter Analysis

1. In the Choosing Analyses form, highlight *pnoise*.

   The form changes to display options for pnoise simulation.

2. For *Sweeptype*, select *Relative* in the cyclic field and type `1` in the *Relative Harmonic* field.
3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `10` in the *Start* field and `500M` in the *Stop* field.
4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `5` in the field.

The top of the pnoise analysis form looks like this.




5. For *Sidebands*, select *Maximum sideband* in the cyclic field. Type 7 in the field.
6. Choose *voltage* in the *Output* cyclic field.

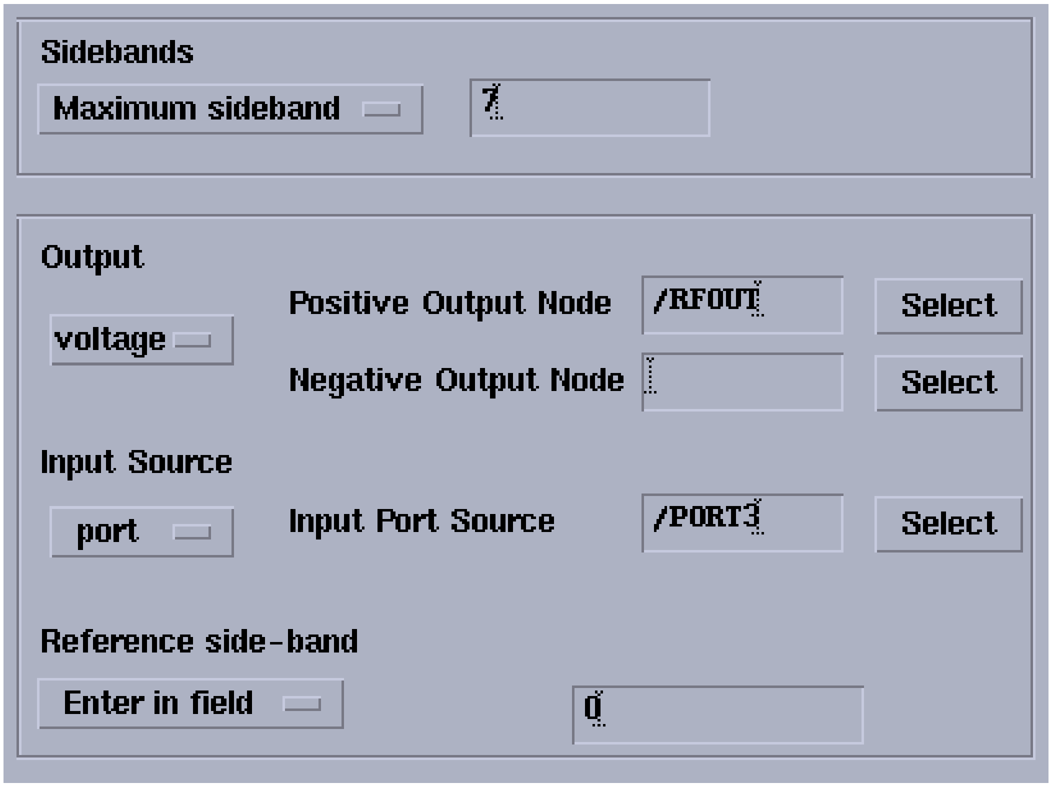
   The following message displays in the CIW.

```
Since voltage was selected as the output measurement
technique for the pnoise analysis, spectre will NOT
subtract any load resistance contribution from the Noise
Figure calculation. If this is undesirable, please select
probe as the output measurement technique and select a port.
```

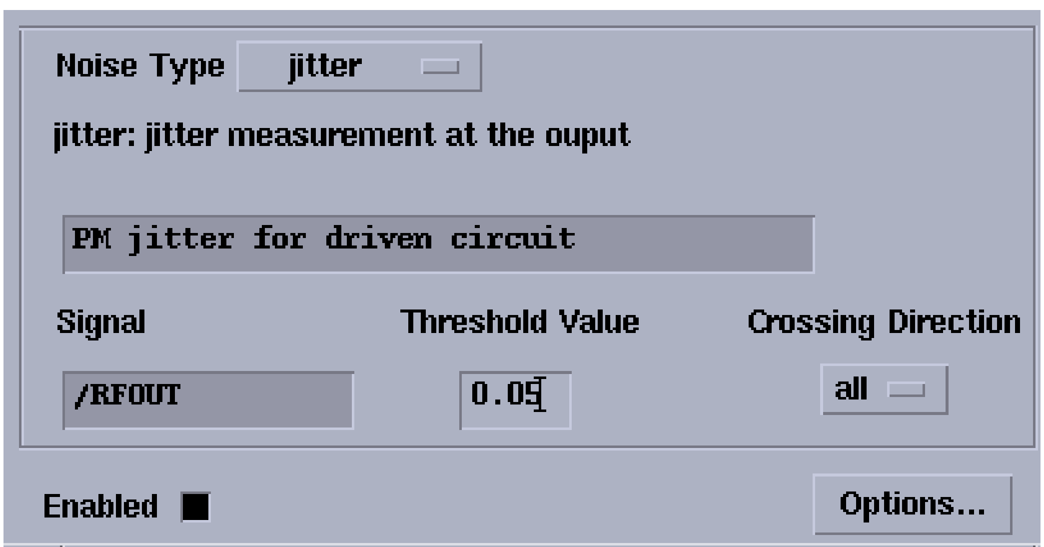
7. Click *Select* for the *Positive Output Node*, and then click the *RFOUT* net in the Schematic window.
8. Leave the *Negative Output Node* field empty.

   This field defaults to */gnd!* You can set the *Negative Output Node* to a different value by clicking *Select* and then clicking on the output node in the schematic.
9. Choose *port* in the *Input Source* cyclic field.
10. Click *Select* for the *Input Port Source*, and then click *port 3* in the Schematic window.

**11.** In the *Reference side-band* cyclic field, select *Enter in field* and type `0` in the field.




**12.** In the *Noise Type* cyclic field select *Jitter* to open the jitter section of the Pnoise Choosing Analyses form.



The form indicates that you are measuring PM jitter at the output for a driven circuit and that the output signal is */RFOUT*.

**13.** Set the *Threshold Value* to `0.05`. Select the threshold value based on knowledge of the results of and earlier run of the PSS analysis. Base your selection on knowledge of your circuit.

**14.** Select *All* in the *Crossing Direction* cyclic field. This captures both up and down crossing points.

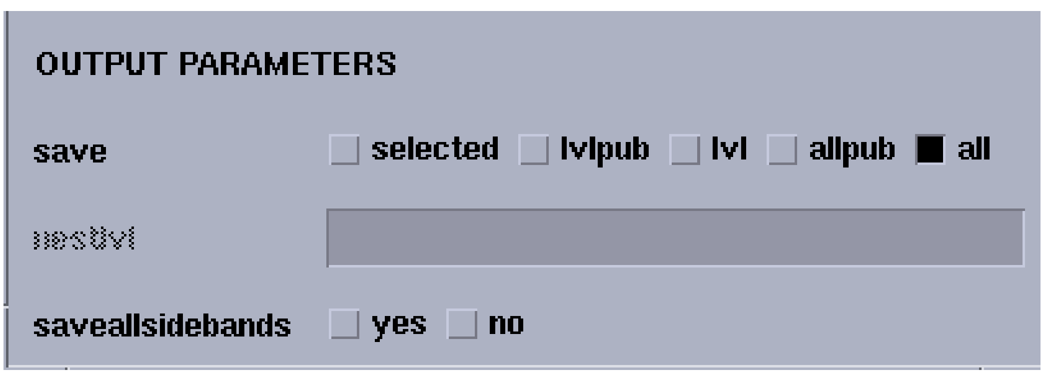
Select *Crossing Direction* equal to *all* cautiously because it directly affects the simulation speed. If the output signal has many transitions, the Pnoise analysis is slowed.

**15.** Verify that *Enabled* is highlighted in the *Pnoise* Choosing Analyses form and click *Apply.*

**16.** Click the *Options* button to display the Options form for the *Pnoise* analysis.

**17.** In the *Output Parameters* section of the Options form, highlight *all* for *save*.

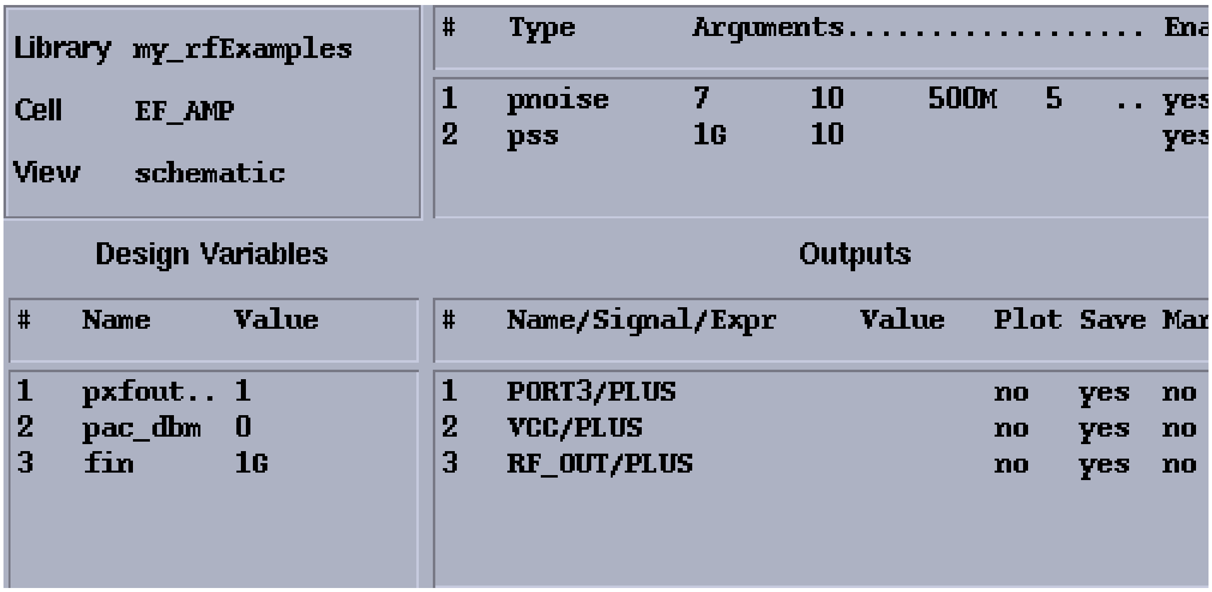
The Output *Parameters* section of the Options form looks like the following.




**18.** In the Pnoise Options form, click *OK*.

**19.** In the Pnoise Analysis form, click *OK*.

The Choosing Analyses form.closes. The ADE window displays the analysis information you have set up.




## Running the Simulations

1. To run the simulations, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Jitter Measurement

➤ Choose *Results – Direct Plot – Main Form* in the ADE window.

The Direct Plot form appears.

### Plot the Upward Transition and Event Time

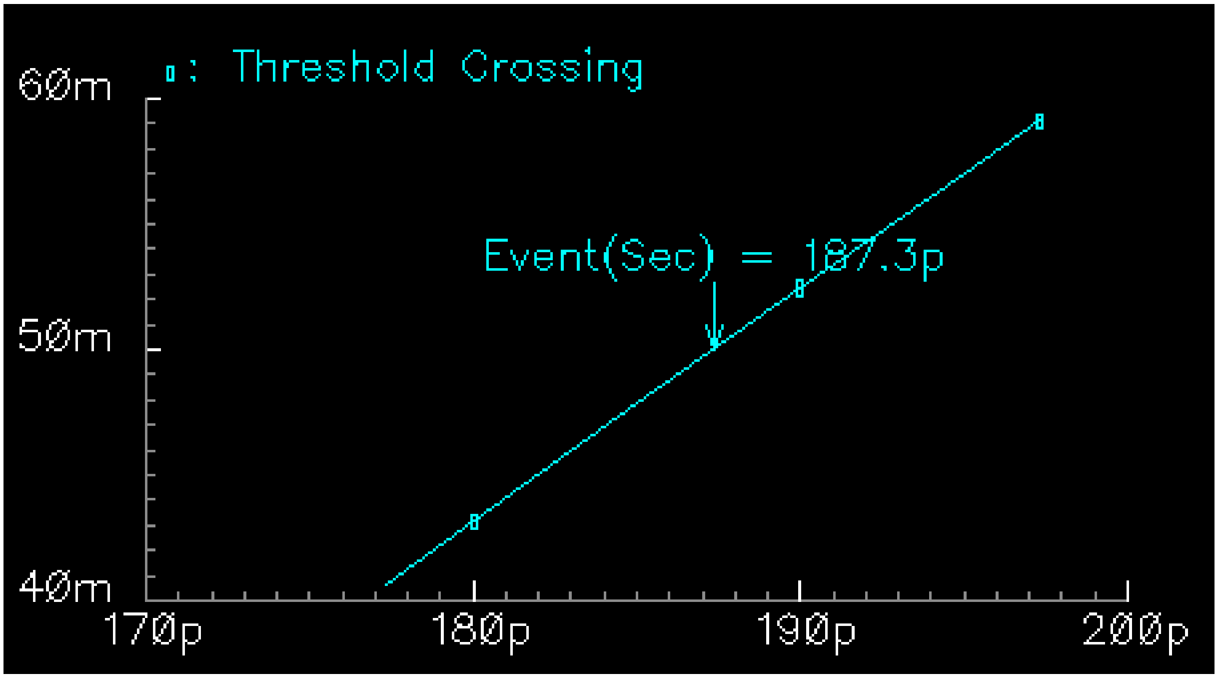
In the Direct Plot form, plot the upward transition.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.

   The form changes to reflect the pnoise jitter analysis.

3. Highlight *Threshold Xing* for *Function*.
4. In the *Signal Level* cyclic field, select *187.3p.*
5. Click *Plot.*

   The up transition and its location display.



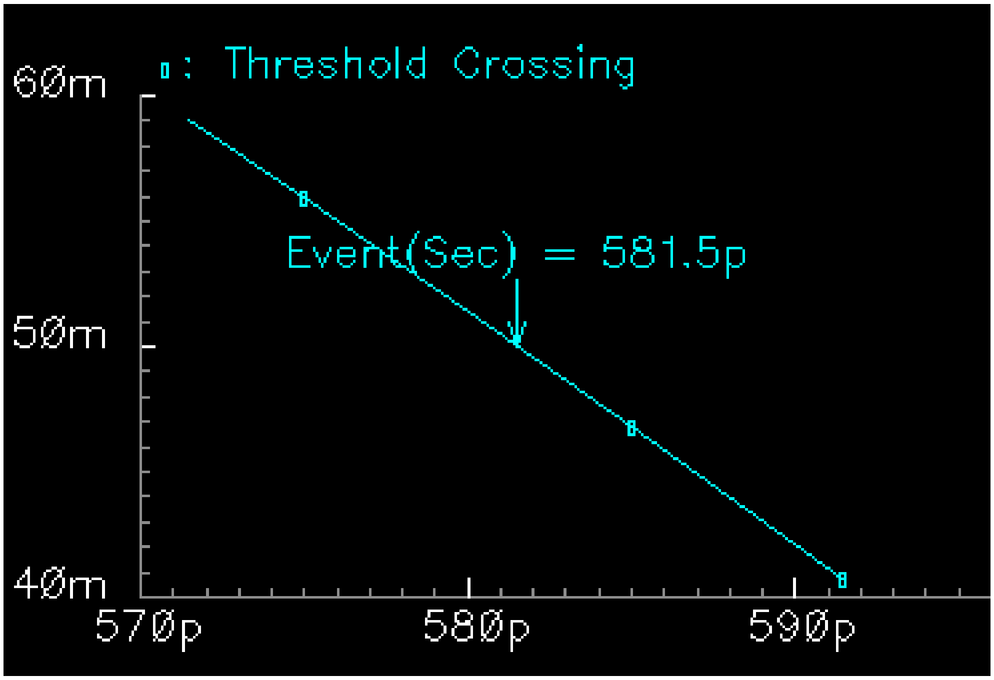

### Plot the Downward Transition and Event Time

In the Direct Plot form, plot the downward transition.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.

   The form changes to reflect the pnoise jitter analysis.
3. Highlight *Threshold Xing* for *Function*.
4. In the *Signal Level* cyclic field select *581.5p.*
5. Click *Plot.*

The down transition and its location display.



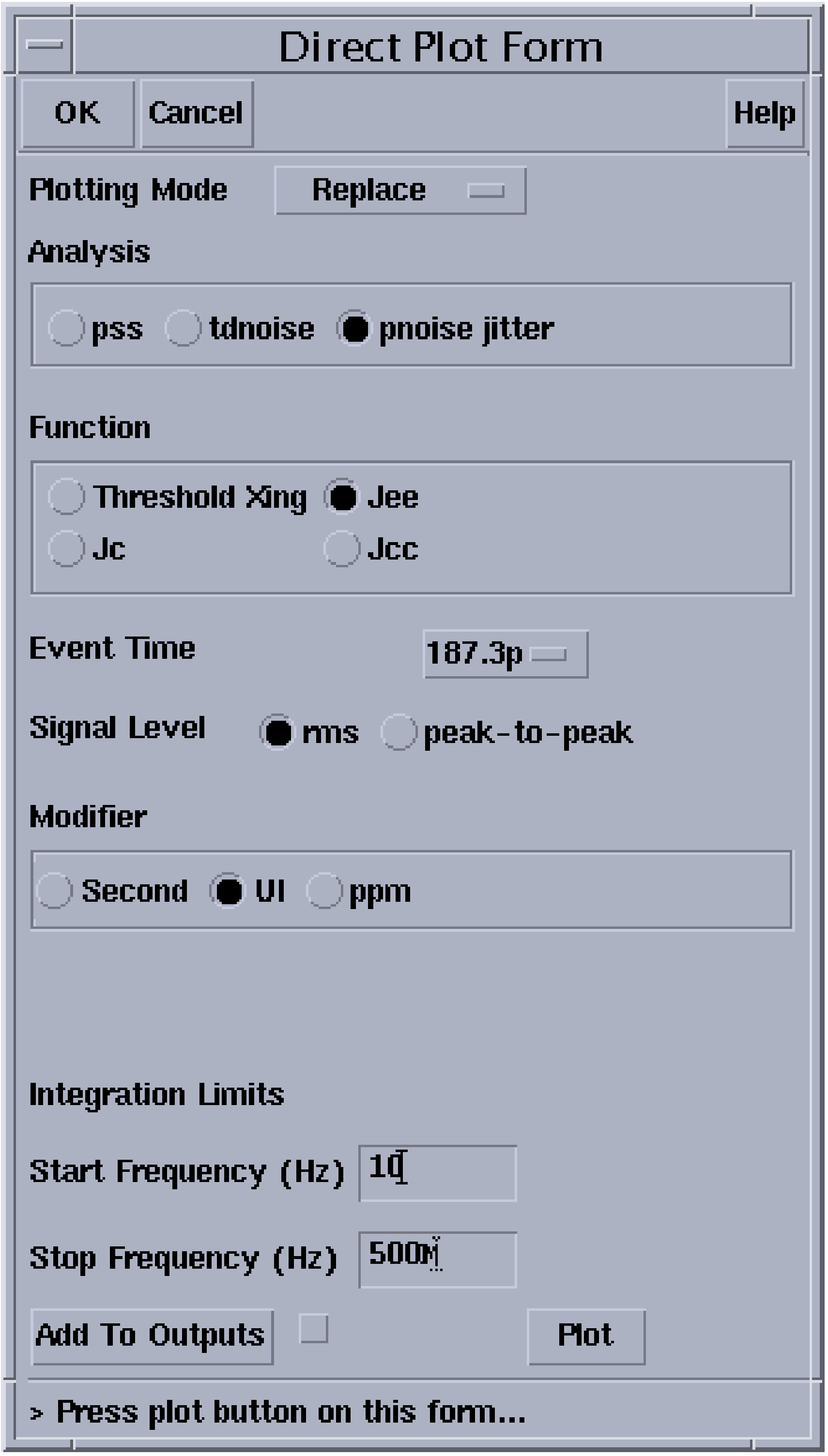

### Plot Edge to Edge Jitter

In the Direct Plot form, plot the edge to edge jitter.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jee* for *Function*.

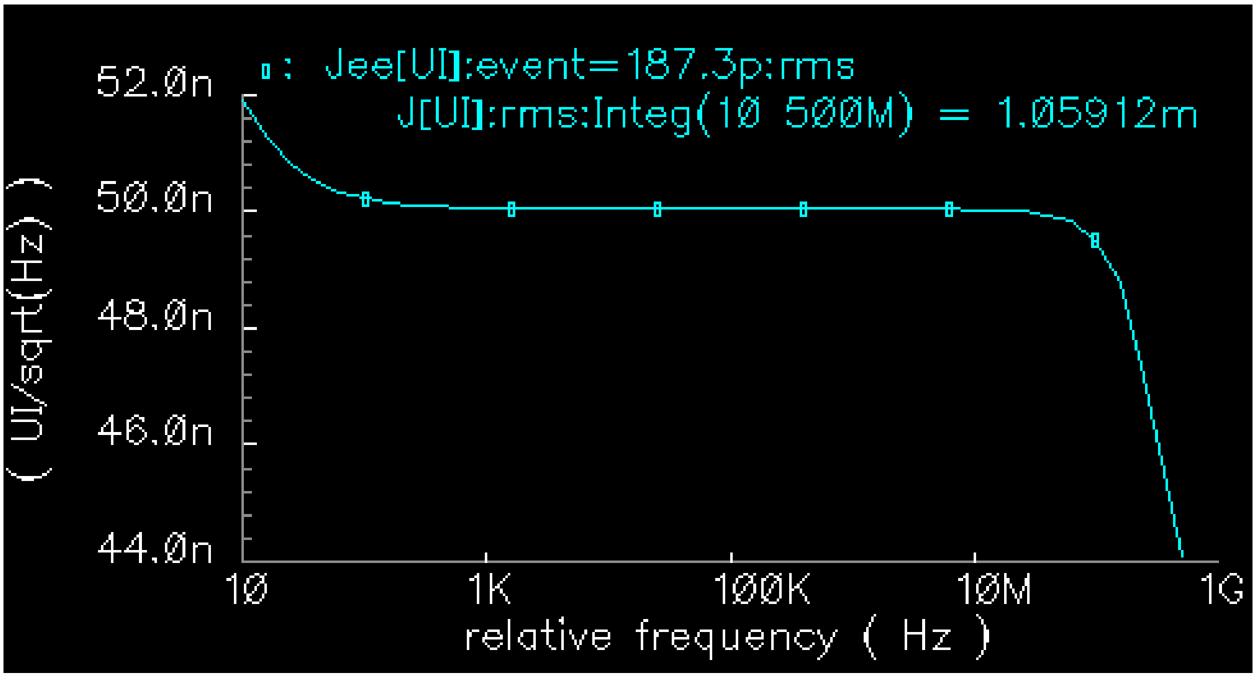
   The form changes to reflect the Function.
4. In the *Event Time* cyclic field, select *187.3p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *UI* for *Modifier*.

UI, the Unit Interval, plots time jitter scaled by the period.



**7.** Click *Plot.*

Edge to edge jitter displays.




Selecting the unit interval, *UI*, selects time jitter scaled by the period. This plots the frequency dependent jitter power spectral density. The plot also indicates the integrated, or total, jitter as a number scaled by the selected modifier, UI in this case.

Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

### Plot Cycle to Cycle Jitter for Both Event Times

In the Direct Plot form, plot cycle to cycle jitter.

**1.** Highlight *Replace* for *Plot Mode*.

**2.** Highlight *pnoise jitter* for *Analysis*.

**3.** Highlight *Jcc* for *Function*.

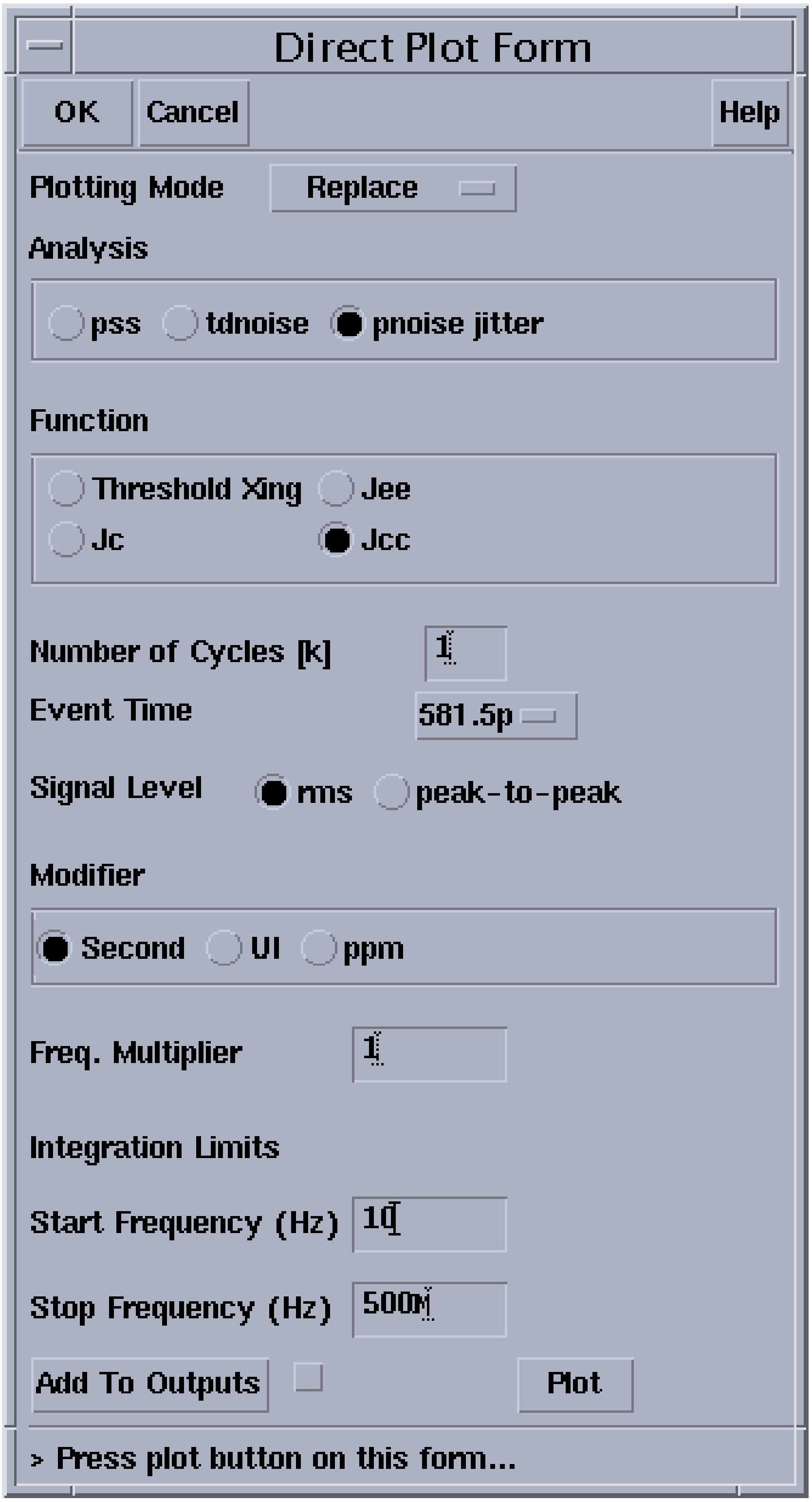
The form changes to reflect the cycle to cycle jitter analysis.

The *Number of cycles [k]* field displays *1.*

4. In the *Event Time* cyclic field select *581.5p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *Second* for *Modifier*.

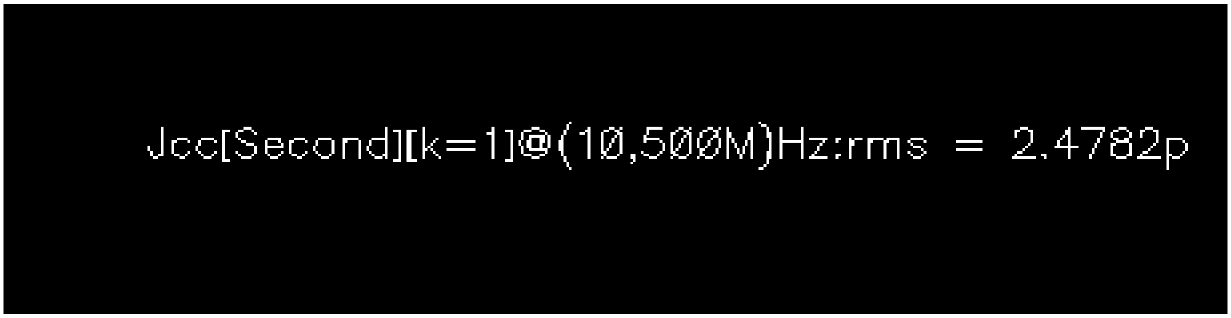
   The *Freq. Multiplier* displays as `1` and the *Integration Limits* display as `10` and `500M` Hz.

The Direct Plot form displays as follows.



7. Click *Plot.*

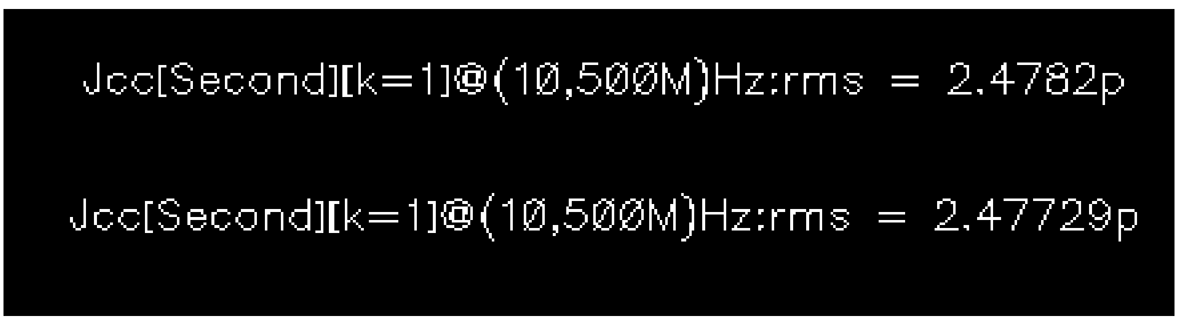
   Cycle to cycle jitter for the 581.5p event time displays.




In the Direct Plot form, make the following changes.

1. Highlight *Append* for *Plot Mode*.
2. In the *Event Time* cyclic field select *187.3p.*
3. Click *Plot.*

   Cycle to cycle jitter displays for both event times.




Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

### Plot Jc for Both Event Times

In the Direct Plot form, plot Jc jitter.

1. Highlight *Replace* for *Plot Mode*.

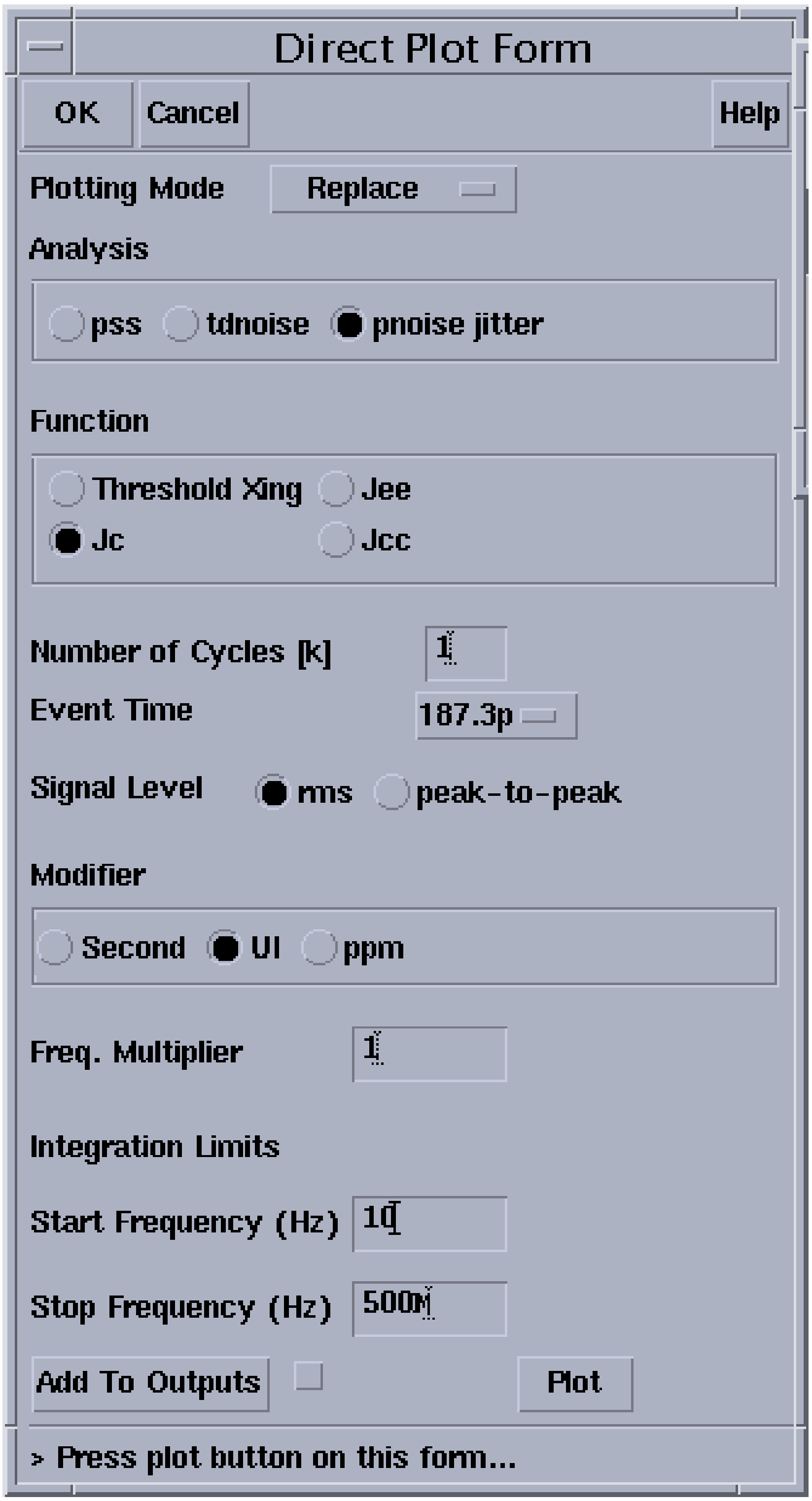
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jc* for *Function*.

   The form changes to reflect the Jc jitter analysis.

   The *Number of cycles [k]* field displays *1.*
4. In the *Event Time* cyclic field select *187.3p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *UI* for *Modifier*.

   The *Freq. Multiplier* displays as `1` and the *Integration Limits* display as `10` and `500M` Hz.

The Direct Plot form displays as follows.



7. Click *Plot.*

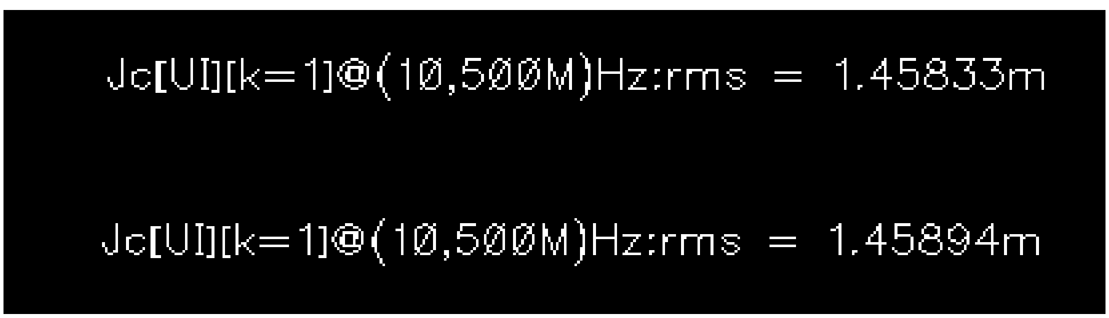
   Jc displays.


### Plot Jc for the 581.5p Event Time

In the Direct Plot form,

1. Highlight *Append* for *Plot Mode*.
2. Change the *Event Time* cyclic field selection to *581.5p.*
3. Click *Plot.*

   Jc display for the second event time is appended to the plot.




Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

# 9

# Methods for Top-Down RF System Design

## Methods for Top Down RF System Design

This chapter describes a methodology for designing analog RF subsystems that fit into larger DSP systems. In particular, this chapter describes how to use a canonical set of top-down behavioral baseband models for exploring RF architectures in the analog design environment. These models come from the following categories in *rfLib*

- Category *top_dwnBB* contains models of common RF function blocks.
  - The default view of each model is the baseband view (called *veriloga*).
  - All models in this category also have a differential passband view (called *veriloga_PB*).

  The only exceptions are the BB_loss and VGA_BB models. They are meant only for baseband analysis and have no passband view.
- Category *top_dwnPB* contains single-ended passband versions of the baseband models.
- Category *measurement* contains the instrumentation block and baseband signal generator models used to make RF measurements. These elements are not part of an RF architecture. They simply facilitate RF measurements and diagnostics.
- Category *testbenches* contains the test circuits used in this chapter to define the model specifications in the *rfLib*. Where possible, the models are specified in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbenches category serve that purpose

See Appendix D, “The RF Library” for more information about the *rfLib* and detailed descriptions of the models it contains.

These models provide RF designers with a fast method to map RF system specifications into detailed RF designs. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs of individual RF system components.

Baseband models are behavioral models and all behavioral models sacrifice some accuracy for increased simulation speed. Such sacrifices are usually acceptable in architectural studies because many implementation-dependent details do not affect high level decisions. The modeling approach taken in top-down design is to simulate only those effects that drive the decisions at hand.

Baseband modeling in no way replaces passband modeling. Some effects missed by equivalent baseband models can affect high level decisions. However, the application of baseband models early and passband models later minimizes the number of slow simulations needed at the lower levels of design abstraction. Baseband models help you to quickly weed out designs that would surely fail tests simulated with passband models.

The success of a modeling approach to top-down design hinges on knowing how the models fit into the design flow and knowing exactly what each modeling parameter means. This chapter has two goals:

- To describe the top-down design flow, from a modeling perspective, for baseband modeling
- To define, as clearly and concisely as possible, the parameters that specify the models

# Top-Down Design of RF Systems

Ideally, the digital signal processing, or DSP, team specifies an RF subsystem that fits snugly into the DSP system. A *snug fit* means that

- The specified RF subsystem does exactly what it needs to do at the lowest possible cost
- A functional specification exists that describes requirements for the RF subsystem

In a top down design flow like the one shown in Figure 9-1 on page 702, the DSP team writes a functional specification for an RF subsystem that has not yet been designed. The functional specification describes what the RF subsystem should and should not do without describing how to build the RF subsystem.

The functional specification supplied by the DSP team describes the RF subsystem at the highest possible level of abstraction. At this point behavioral models can be specified rather than measured. This early in the design cycle, the functional specification might well be incomplete or inconsistent. A good top-down design flow can detect problems, such as omissions and inconsistencies in the design, early in the design cycle when they are easier and less expensive to fix. Problems detected later in the design cycle can be much more costly and very difficult to resolve.

Using the functional specification supplied by the DSP team and the behavioral baseband models from *rfLib*, the RF system designers can easily explore RF architectures in the analog design environment. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. By switching to the passband views of the baseband models, the RF design team maps DSP measurements to RF measurements. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs for individual system components.

Using the functional specification and exploring and testing with the baseband and passband models, the RF team can efficiently create a detailed design specification that fully describes the RF subsystem. The design specification can include detailed instructions for building the RF subsystem. At this stage of the design cycle, everything that is known about the design is described at the lowest level of abstraction.

You can now extract behavioral models of a detailed design from simulated measurements. The problem remains that detailed designs usually do not exist until the project is complete. To jump directly to a detailed design implies that the design flow is bottom up. Bottom up flows are important in many projects, but not in all.

DSP and RF designers sometimes have trouble communicating through specifications because the two groups deal with different metrics. For example, DSP designers deal with *bit error rates* and *error vector magnitude statistics* whereas RF designers deal with *intercept points* and *noise figures*.

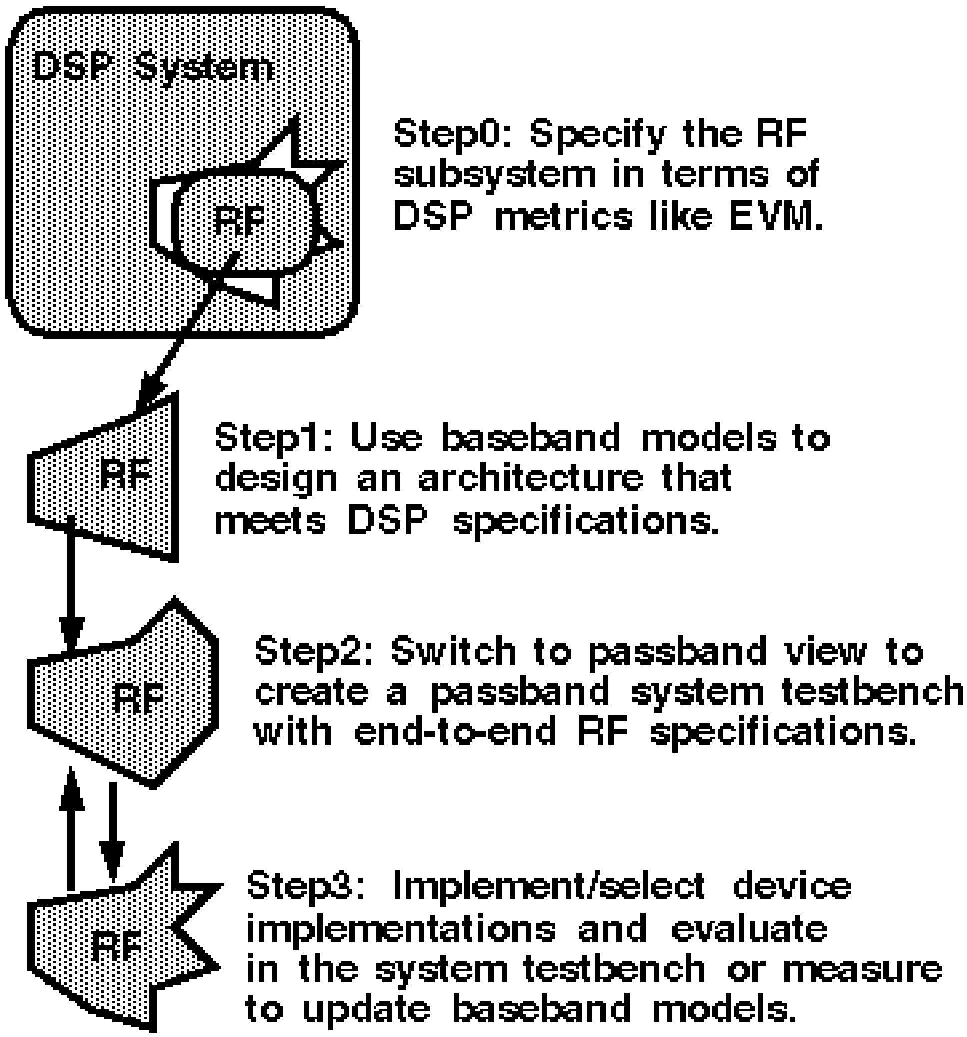
The new models described here are designed to help RF system designers in two ways.

- First, the baseband models enable RF system designers to quickly explore the RF architectural space, as specified by the DSP metrics, while letting the RF engineers specify the RF system components with RF metrics. The circuit implementations of the RF system components are easier to design and test when the components are specified with standard RF metrics.
- Second, the baseband models can be switched quickly to a passband views where the RF system model can generate end-to-end RF metrics. With end-to-end metrics, the new view can quickly simulate how the detailed design of a particular RF system component affects end-to-end performance.

## Use Model for Top Down Design

The following steps outline the RF design process with focus on the early phases of the design as illustrated in Figure 9-1 on page 702.

**Figure 9-1 The Top Down Design Flow and Use Model**




### Specify the RF Subsystem in Terms of DSP Metrics

Before you begin the RF subsystem top-down design flow, the DSP design team should completely specify the RF subsystem in terms of DSP metrics. This preliminary step distinguishes the end of the DSP design flow from the beginning of the RF top down design flow and formally hands-off the RF subsystem design specifications to the RF design team.

### Explore Candidate Architectures with Baseband Models

The first step in top down RF design is to select a candidate RF architecture. An RF architecture is a set of interconnected RF function blocks that, taken together, describe how a receiver or transmitter operates. You specify each function block in terms of standard RF metrics such as IP3, gain, bandwidth, and noise figure.

The models you use early in the design cycle as you explore candidate RF architectures must run fast. Each simulations can span hundreds of symbols and each symbol can easily span thousands of RF carrier cycles. The space defined by the function block specifications in each candidate architecture is far too vast to explore with slow, highly precise models. Models used for architectural exploration must quickly reduce the design space down to a size that can be explored with more precision.

The most efficient models for architectural exploration suppress the RF (IF) carrier and are called *baseband* models. In contrast, the *passband* models (introduced in the next step) do not suppress carriers.

You can use the Circuit Optimizer during architectural exploration to help balance the function block specifications for a candidate architecture. For example, you can use the Circuit Optimizer to minimize RMS EVM while ensuring that other measurements stay within acceptable limits.

When you have determined the nominal specifications for each function block, you must put tolerances around them. In the analog world *specifications without tolerances are meaningless*. The tolerance space is usually explored with some mix of experience, feasibility, a variety of analyses, and outright arbitrary decisions.

There are several ways that you can use the baseband models to test candidate tolerances as well as to determine some tolerances analytically.

One way to test a candidate set of tolerances is to run a Monte Carlo analysis on the metric of interest, like RMS, EVM, or signal-to-noise ratio (SNR).

Another approach is to use the Circuit Optimizer *in reverse*, as a de-optimizer, to determine worst case performance.

Yet another approach is to compute each tolerance separately from a parametric plot. When you have determined all but 2 or 3 tolerances, you can use a multidimensional parametric analysis to map out the performance space and easily identify the remaining tolerances.

### Switch to Passband Models and Create an RF System Testbench

The second step in top down RF design is to create a passband view of the system model.

The passband system model performs two functions:

- Confirms that the filters perform as expected.

- Creates an end-to-end testbench that you can use to design the individual function blocks.

For computational efficiency during system passband testing, at any one time, model one or two selected function blocks at the device level. Model all other blocks in the system behaviorally using passband models.

Derive the tolerances by performing the same Monte Carlo analysis or Circuit Optimizer analysis you used to test the function block tolerances in the first step, but this time replace the DSP metrics with end-to-end RF metrics. After you know how far the end-to-end RF metrics can vary, you can insert a device-level model of a function block into the testbench to see how close it drives the system toward violating a derived end-to-end RF specification.

#### Implement the Function Blocks with Active and Passive Devices

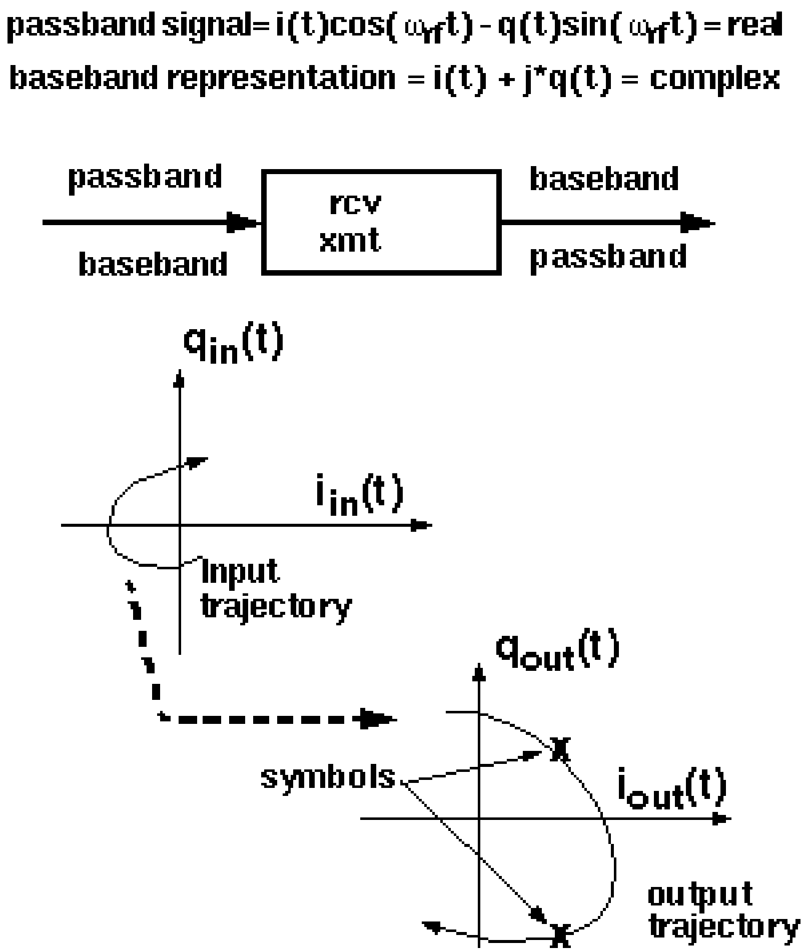
The last step in the top down design process is to implement the function blocks with device models. Because the function blocks are specified in terms of standard RF metrics, you can easily measure the modeling parameters to make sure they fall within the specified tolerances. You can also insert the measured parameters back into the baseband model of the system to check the DSP metrics, or insert the device-level model directly into the passband testbench to check the derived end-to-end RF specifications.

## Baseband Modeling

A baseband model for an RF function block simulates what happens to the baseband representation of a signal as it passes through the block. A baseband model maps input baseband signal trajectories into output baseband signal trajectories. If you sample a baseband signal periodically in time and plot the samples in the complex plane, the resulting scatter plot shows the symbol constellation.

Figure 9-2 on page 705 mathematically defines a baseband representation of a passband signal. The *i* and *q* signals are the real and imaginary parts of a complex signal that rides on the two phases of an RF carrier.

#### Figure 9-2 Baseband Representation of a Passband Signal



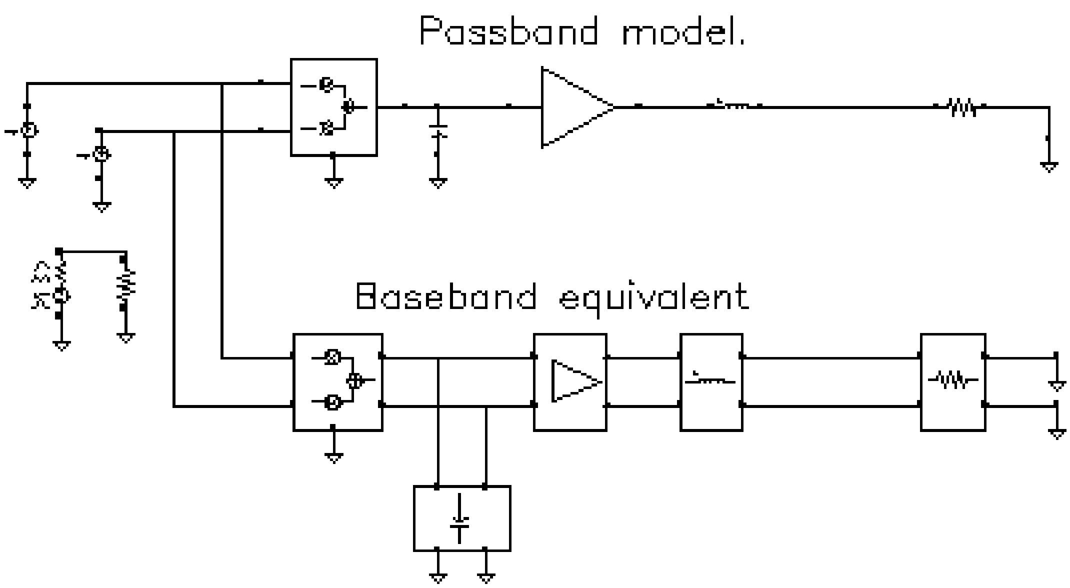

Baseband models simulate only what happens to the carrier fundamental. Consequently, they only account for non-linearities with odd symmetry. Non-linearities with even symmetry produce no output at the carrier fundamental; they affect the carrier fundamental only when cascaded. For example, a second order non-linearity in one block can create a DC offset at it’s output. Upon passing through a subsequent block with another second order non-linearity, the DC offset can mix with the carrier to affect the output carrier fundamental. You should model cascaded blocks producing unfiltered even harmonics as a single baseband model rather than as separate baseband models cascaded together. The non-linearities that most often dominate performance have odd symmetry.

## Example Comparing Baseband and Passband Models

The example in this section walks you through an Envelope analysis that illustrates the relationship between baseband and passband models. Following the simulation, you plot the baseband equivalent output signals as computed by the baseband and passband circuits.

The *BB_test_bench* schematic shown in Figure 9-3 on page 706 illustrates the difference between passband and baseband modeling. This circuit is located in the *rfExamples* library.

**Figure 9-3 The BB_test_bench Schematic**




The *BB_test_bench* circuit shows a passband circuit (across the top of the schematic) and its baseband equivalent circuit (across the bottom of the schematic). The same baseband signals drive both circuits but only the passband circuit mixes the baseband signals up to RF. The power amplifier is not matched to either input or output impedances and both impedances are reactive.

Before you start, perform the setup procedures described in Chapter 3.

### Opening the Baseband Test Bench Circuit

1. In the CIW, choose *File – Open*.

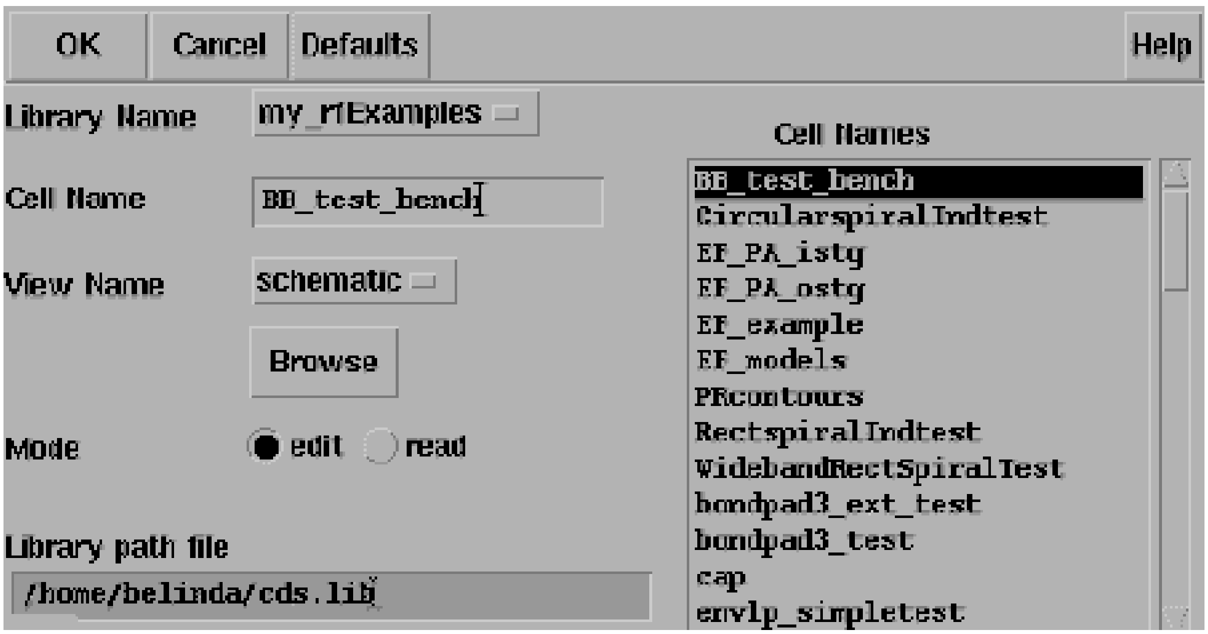
   The Open File form appears.

2. In the Open File form,

**a.** Choose *rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

**b.** Choose *BB_test_bench* in the *Cell Names* list box. Note that the *View Name* cyclic field displays *Schematic*.

**c.** The completed Open File form appears like the one below.

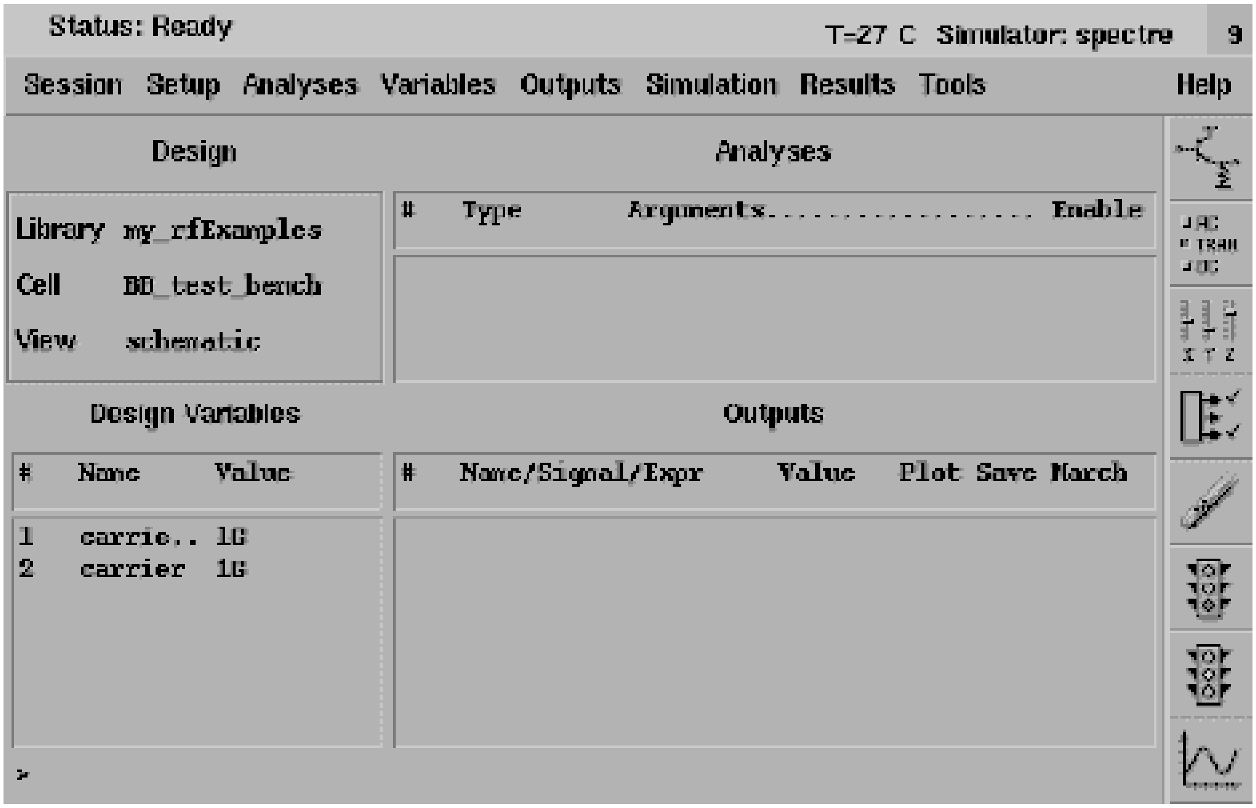



**3.** Click *OK*.

The Schematic window for the *BB_test_bench* appears.

**4.** In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *BB_test_bench* in the Choosing Design form.

### Choosing Simulator Options

**1.** Choose *Setup – Simulator/Directory/Host* in the ADE window.

The Simulator/Directory/Host form appears.

**2.** In the Simulator/Directory/Host form, specify the following:

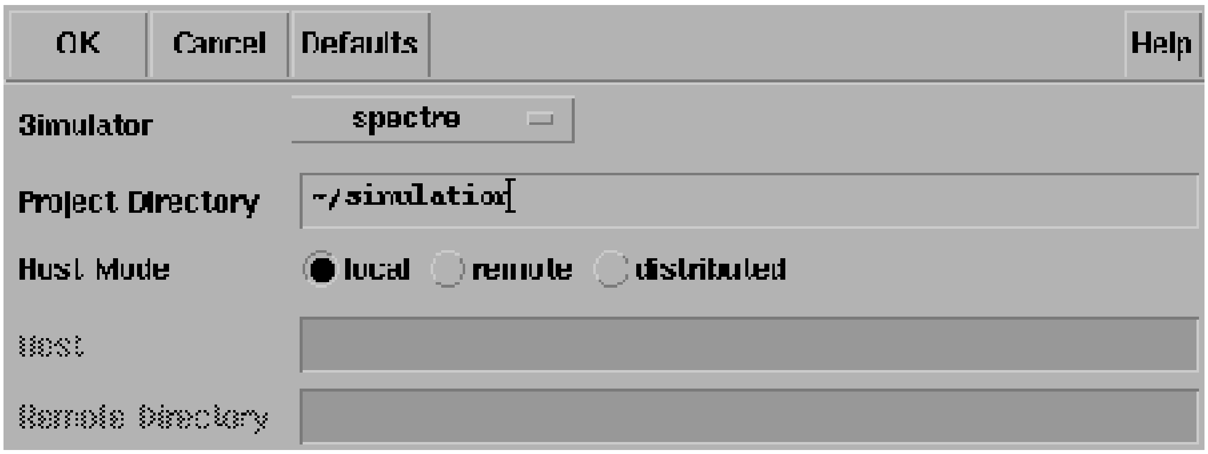
**a.** Choose *spectre* for the *Simulator*.

**b.** Type the name of the project directory, if necessary.

**c.** Highlight the *local*, *remote*, or *distributed* button to specify the *Host Mode*.

For remote simulation, type the name of the host machine and the remote directory in the appropriate fields. For distributed simulation, fill out the other fields that appear.

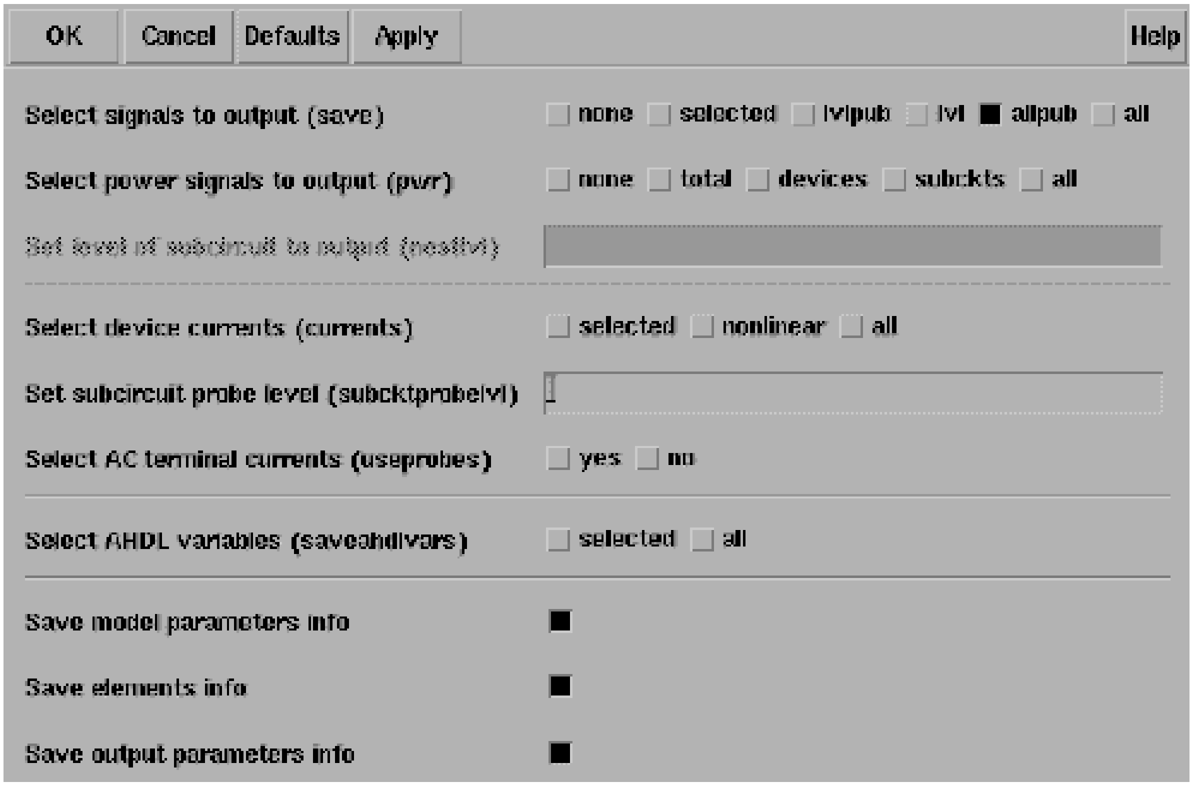
The completed form appears like the one below.




3. In the Simulator/Directory/Host form, click *OK*.
4. In the ADE window, choose *Outputs – Save All.*

   The Save Options form appears.

5. In the *Select signals to output (save)* section, be sure *allpub* is highlighted.




6. In the Save Options form, click *OK*.

### Setting Up Model Libraries

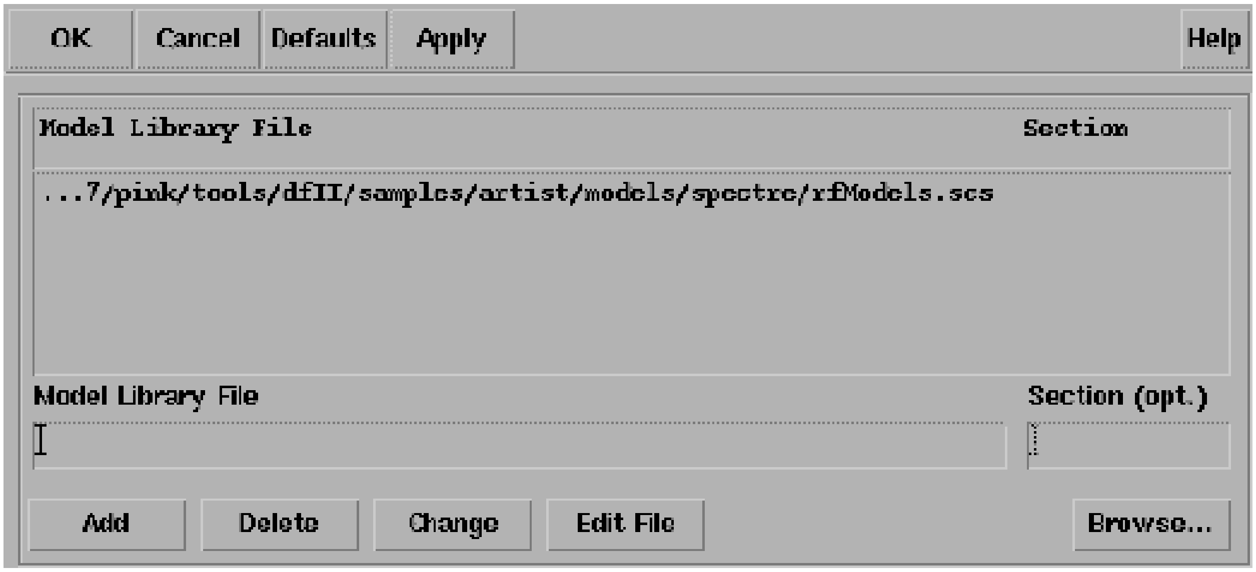
1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs.`

3. In the Model Library Setup form, click *Add*.

The completed form appears like the one below.




**4.** In the Model Library Setup form, click *OK*.

**5.** In the ADE window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

### Setting Up the Envelope Analysis

**1.** Choose *Analyses – Choose* in the ADE window.

The Choosing Analyses form appears.

**2.** In the Choosing Analyses form, click *envlp.*

**a.** Enter *ff* in the *Clock Name* field.

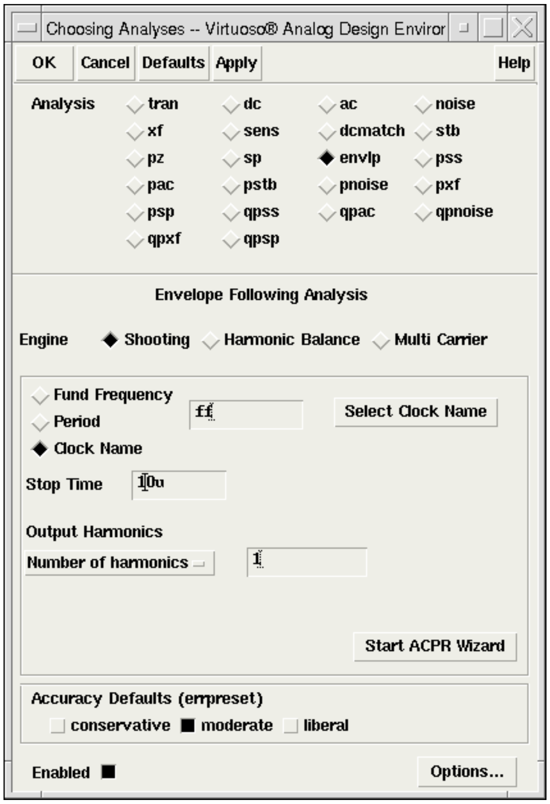
**b.** Enter `10u` in the *Stop Time* field.

**c.** In the *Output Harmonics* cyclic field, select *Number of harmonics*.

**d.** Enter `1` in the *Number of harmonics* field.

**e.** Select *moderate* for the Accuracy Defaults.

The correctly filled out form appears below.




**3.** In the Choosing Analyses form, click *OK*.

## Running the Simulation

**1.** In the ADE window, choose *Simulation – Netlist and Run*.

The output log file appears and displays information about the simulation as it runs.

Look in the CIW for a message that says the simulation completed successfully.

## Plotting the Baseband Equivalent Output Signals

**1.** In the ADE window, choose *Results-Direct Plot-Main Form*.

The Direct Plot form appears.

**2.** In the Direct Plot form, do the following:

**a.** Highlight *Replace* for *Plotting Mode*.

**b.** Highlight *envlp* for *Analysis*.

**c.** Highlight *Voltage* for *Function*.

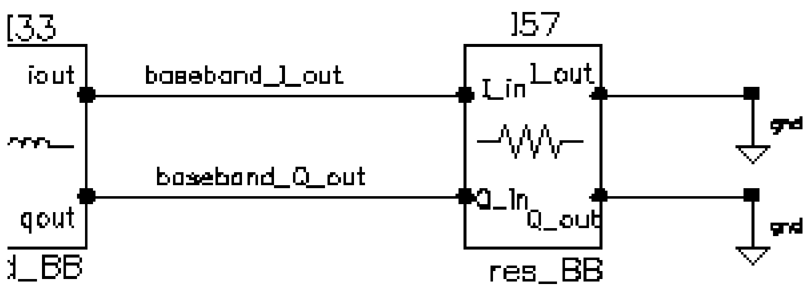
**d.** Highlight *time* for *Sweep*.

The *Net* selection appears in the *Select* cyclic field and the label *Description: Envelope Voltage vs Time* appears.

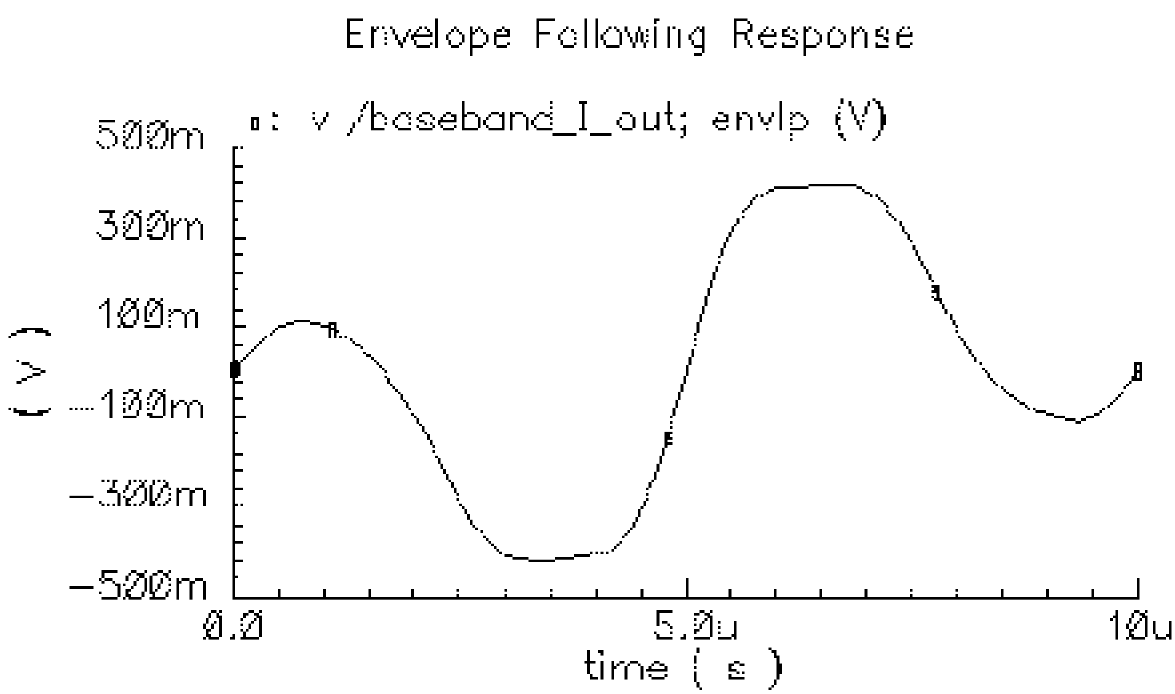
**e.** Following the message at the bottom of the Direct Plot form

*Select Net on schematic...*

Click the `baseband_I_out` net.




The first trace appears in the waveform window.



3. In the Direct Plot form, do the following:

   a. Highlight *Append* for *Plotting Mode*.

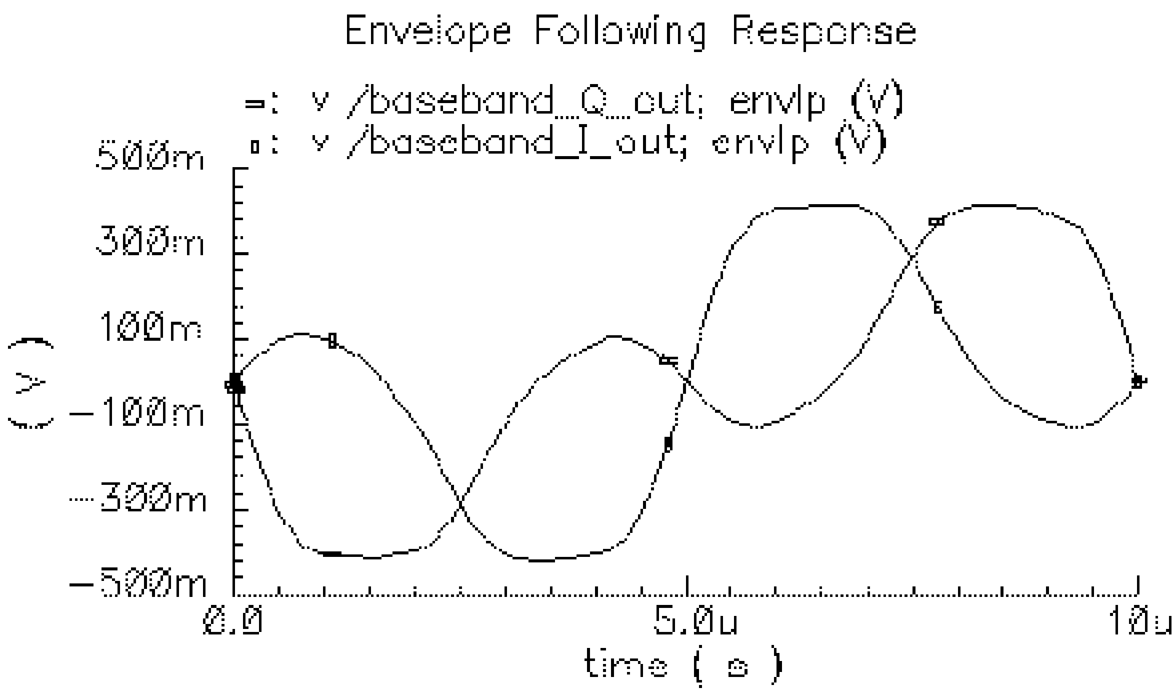
   b. Leave *Voltage* set for *Function* and *time* set for *Sweep*.

   c. Following the message at the bottom of the Direct Plot form

      *Select Net on schematic...*

      Click the `baseband_Q_out` net.

      The second trace is added to the waveform window. Both baseband equivalent output signals for the baseband model are plotted.




4. In the Direct Plot form, do the following:

   a. Leave *Append* for *Plotting Mode* and *Voltage* for *Function.*

   b. Highlight *harmonic time* for *Sweep*.

   c. Highlight *Real* for *Modifier*.

   d. Following the message at the bottom of the form,

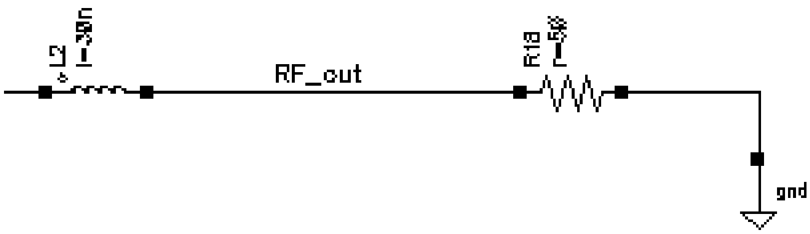
      `Select Harmonic Number on this form...`

      Select 1 for harmonic number.

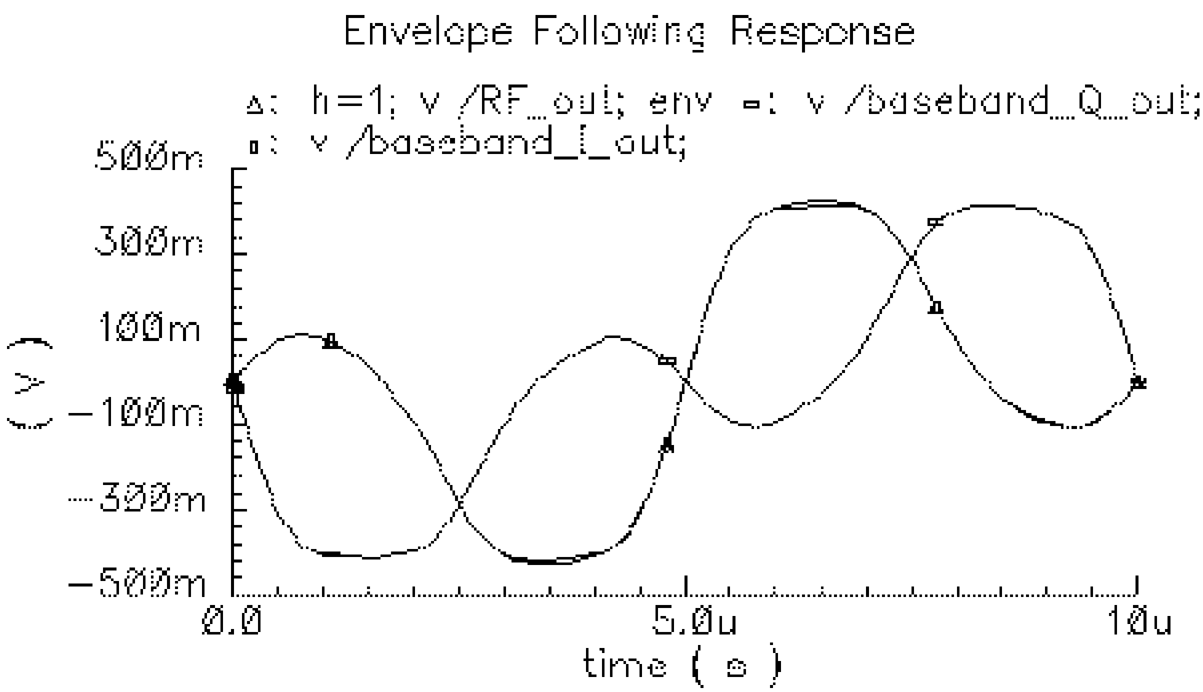
**e.** Following the next message at the bottom of the form

`Select Net on schematic...`

Click the `RF_out` net.




A third trace is added to the waveform window.




**5.** In the Direct Plot form, do the following:

**a.** Leave *Append* for *Plot Mode*, *Voltage* for *Function*, *harmonic time* for *Sweep*, and 1 for *Harmonic Number*.

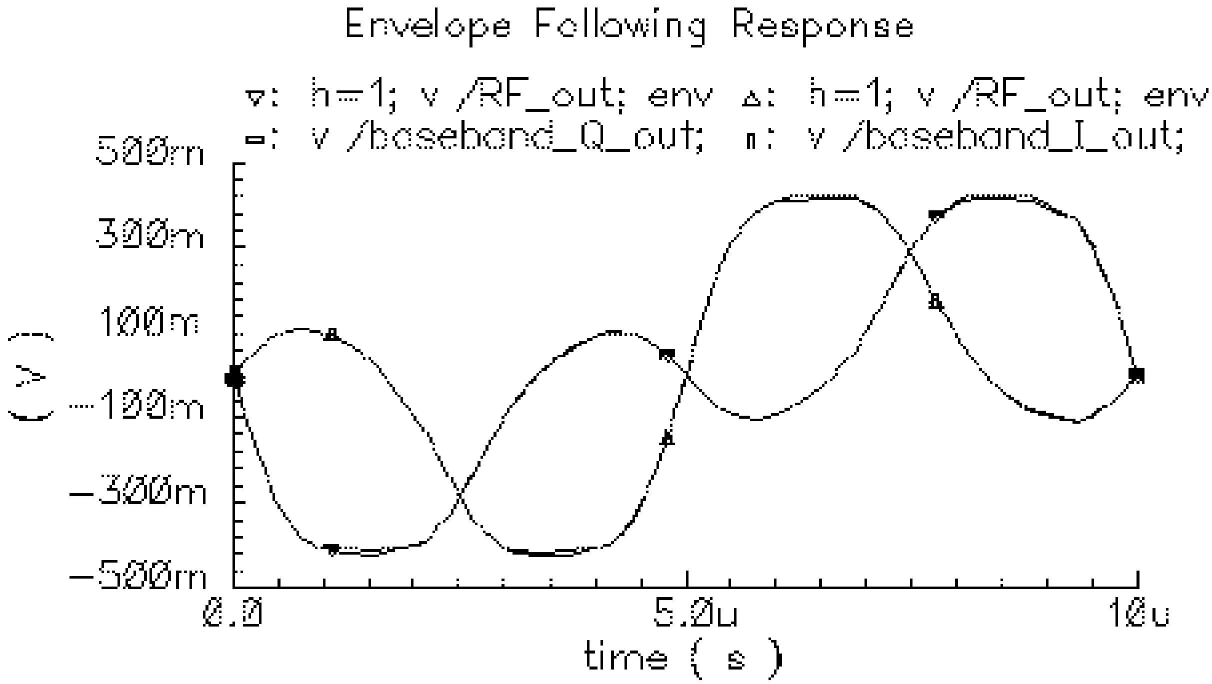
**b.** Highlight *Imaginary* for *Modifier*.

**6.** Following the message at the bottom of the form

`Select Net on schematic...`

Click the `RF_out` net again.

A fourth trace is added to the waveform window. Both baseband equivalent output signals for the passband model are added to the plot.




In the waveform display window you should now see what at first appears to be two traces. When you look more closely, you should see that each trace is actually two traces, one nearly on top of the other, making a total of four traces.

The plot resulting from this example illustrates how well baseband modeling corresponds to the time-varying fundamental Fourier component computed by Envelope analysis and raises two questions:

- Why use baseband models when Envelope analysis gives the same results?
- Why not use baseband models all the time?

Running a transient analysis with only the baseband models answers the first question. If from the ADE window you deactivate the passband circuit by setting the *`carrier_pb`* variable to zero, disable the Envelope analysis, and set up and run a 10µs transient analysis, you observe the same baseband results, but the transient simulation runs over 100 times faster.

Examining the Envelope results answers the second question. If you look closely at both waveforms you notice that the baseband waveforms clip at a slightly lower level than the Envelope waveforms. This is because hard limiting of the carrier generates higher-than-third-order harmonics and the behavioral baseband model only simulates third order non-linearities.

# rfLib Library Overview

The *rfLib* include three kinds of models to support baseband modeling:

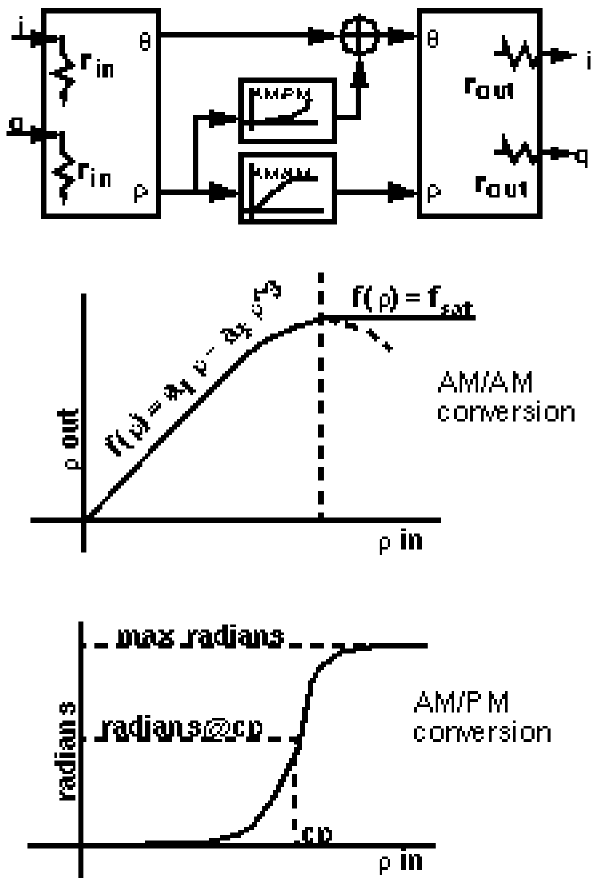
- Instrumentation models
- Non-linear memoryless models
- Linear models with memory

The instrumentation models provide stimuli, diagnostics, and performance metrics relevant to the DSP system.

Both the linear models with memory and the non-linear memoryless models simulate the function blocks in an RF architecture and are specified in terms of common RF metrics. The RF function block models include input referenced white Gaussian noise as specified by noise figure. The *rfLib* includes models for the following RF function blocks—amplifiers, mixers, filters, and phase shifters; where *filters* includes single resistors, capacitors, and inductors.

The non-linear models simulate AM/AM conversion [1] with a third-order polynomial that saturates at the peak of the transfer curve. The polynomial is specified by the gain and either the input-referred IP3 or the output-referred 1 dB compression point. Only the non-linear baseband models simulate AM/PM conversion. AM/PM conversion [1] is an important effect that is hard, if not impossible, to simulate with passband behavioral models. Figure 9-4 on page 718 shows the basic baseband non-linearity.

**Figure 9-4 Basic, Baseband Non-Linearity**




The linear models are the key to simulating loading effects at baseband. In RF integrated circuits, loading effects are important because it is often hard to integrate impedance matching networks. The baseband models of reactive elements differentiate our approach from the spreadsheet-based approaches to RF system design. The baseband capacitor and inductor models (*cap_BB* and *ind_BB* in *top_dwnBB*) let you simulate reactive loading effects in the time domain, where non-linearities are more naturally modeled.

The baseband models of reactive elements also play a key role in modeling filters. Most digital communications text books [1,2] explain that you can model a passband transfer function at baseband by simply frequency-shifting the transfer function. What these books do not describe is how to implement the resulting transfer function in a general circuit simulator such as Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF). The shifted transfer function usually lacks complex conjugate symmetry about zero frequency and therefore has a complex impulse response.

The first consequence of modeling RF function blocks at baseband is that all equivalent baseband models have four terminals instead of two:

- One set of terminals represents the in-phase signals, $i_{in}(t)$ and $i_{out}(t)$
- The other set of terminals represents the quadrature signals, $q_{in}(t)$ and $q_{out}(t)$

  Both sets of terminals are illustrated in Figure 9-2 on page 705.

The mathematics illustrated in Figure 9-5 on page 720 and Figure 9-6 on page 721 summarize the ideas behind a time-varying coordinate transformation that models reactive elements at baseband. The mathematics apply to capacitors as well as inductors.

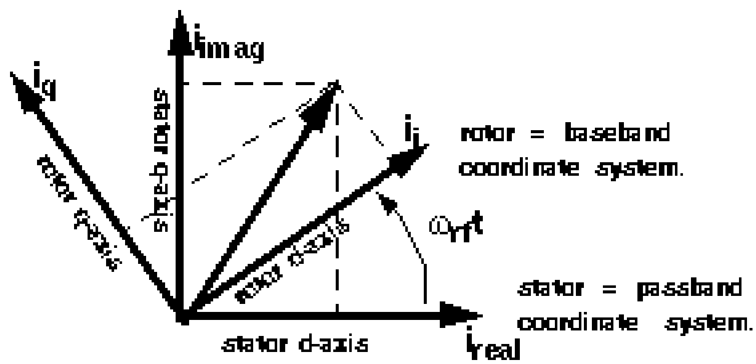
There is a well-documented but little-known electro-mechanical analogy for the derivation of the inductor baseband equivalent model. The four inductor terminals resemble the stator windings of a two-phase rotating machine with shaft speed equal to the RF carrier frequency. Modulation is mathematically analogous to the flux linking a stator winding due to currents in orthogonal rotor windings. The flux depends on the shaft angle just as a modulated signal depends on the carrier phase. Transforming the vectorial equation for $v=Ldi/dt$ to the rotor reference frame suppresses the RF carrier and introduces a *speed voltage* [3,4,5,6,7,8,9], or back electro-motive force (back EMF), that couples the differential equations.

An expression for the real current (i.e. the passband current) appears in Figure 9-6 on page 721. The real current is modeled as the projection of a two-dimensional rotating vector onto a stationary axis, the *real* axis. The vector rotates with an angular velocity equal to the RF carrier frequency.

**Figure 9-5 Passband Current for an Inductor**

$$i_{real} = i_i(t)\cos(\omega_{rf}t) - i_q(t)\sin(\omega_{rf}t) = Real[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$

$$i_{imag} = i_i(t)\sin(\omega_{rf}t) + i_q(t)\cos(\omega_{rf}t) = Imag[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$




$$i=(i_i+j*i_q)*e^{j\omega_{rf}t}$$

$$v=(v_i+j*v_q)*e^{j\omega_{rf}t}$$

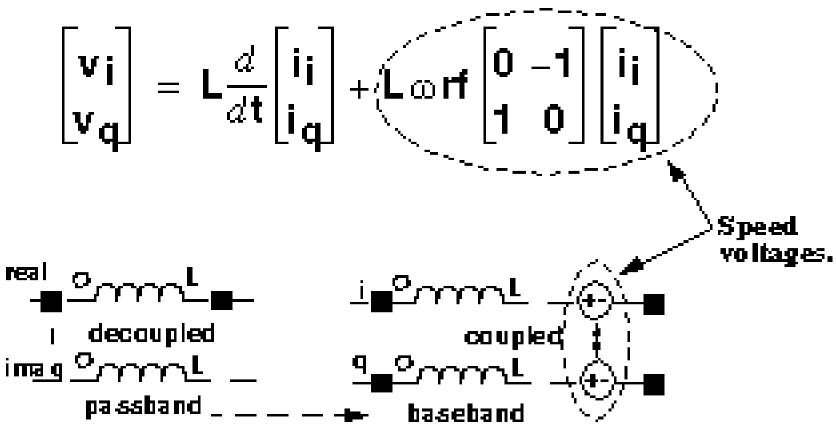
$$v=L\,di/dt$$

$$(v_i+j*v_q)\cancel{e^{j\omega_{rf}t}}=L*[d(i_i+j*i_q)/dt + j*\omega_{rf}*(i_i+j*i_q)]\cancel{e^{j\omega_{rf}t}}$$

The rotating vector also has a projection onto another stationary axis orthogonal to the real axis. In the baseband literature, the orthogonal projection is the Hilbert transform of the real signal. The constitutive relationship of the inductor, *v=Ldi/dt*, is expressed in terms of coordinates in a reference frame that rotates with the vector.

Figure 9-6 on page 721 shows the constitutive inductor relationship between voltage and current in the rotating reference frame. Note that the trigonometric terms, the terms that slow simulation speed, are gone and the two projections are now coupled through *speed voltages*. The term speed voltage comes from the fact that the voltages depend on the angular speed of the rotating reference frame. In motor theory, that speed is the shaft speed. Speed voltage is similar to the back EMF in a motor. Because of speed voltages, baseband models of filters and reactive elements must have their carrier frequency specified. The carrier frequency is the frequency for which the baseband signals are referenced. For example, the carrier frequency for an RF filter would be the RF frequency while the carrier frequency for an IF filter would be the IF frequency.

**Figure 9-6  Relationship Between Voltage and Current for an Inductor**



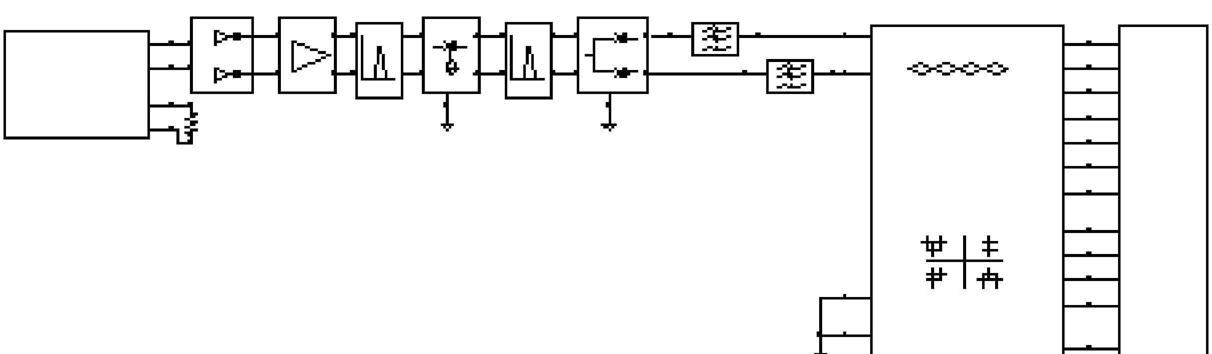

The baseband counterparts of the passband filter models are built up from inductors and capacitors modeled in the rotating reference frame.

In the complex expression for $v=Ldi/dt$, if you replace $d/dt$ with $j\omega$, you find that the impedance of the inductor changes from $jL\omega$ to $jL(\omega+\omega_{rf})$. The same holds for capacitors, which means a filter transfer function, $H(\omega)$, has a baseband equivalent equal to $H(\omega+\omega_{rf})$. This is simply the original passband transfer function shifted to the left by an amount equal to the carrier frequency. Our time domain baseband models are consistent with the text book frequency domain explanation of baseband modeling.

# Use Model and Design Example

This section describes how to use the baseband models during the architectural design phase. The following examples show you how to

- Construct a baseband model for a simple receiver



- Use the Circuit Optimizer to balance specifications among the function blocks
- Create a passband testbench for the receiver

The design goals were chosen arbitrarily. The example is meant simply to illustrate how to use the library and is a derivative of the design found in [10]. If you find that some parameters are not specified, leave them as default values. You construct the receiver from left to right, from input to output.

## Opening a New Schematic Window

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

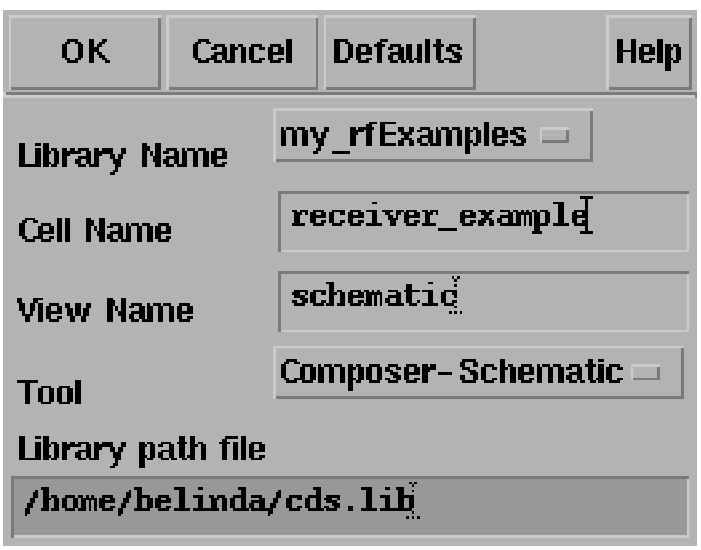
2. In the Create New File form,

   a. Choose *my_rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

   b. Enter `receiver_example` in the *Cell Name* field.

   c. Select *Composer-Schematic* in the *Tool* cyclic field. `schematic` appears in the *View Name* field.

   d. The completed form appears like the one below.



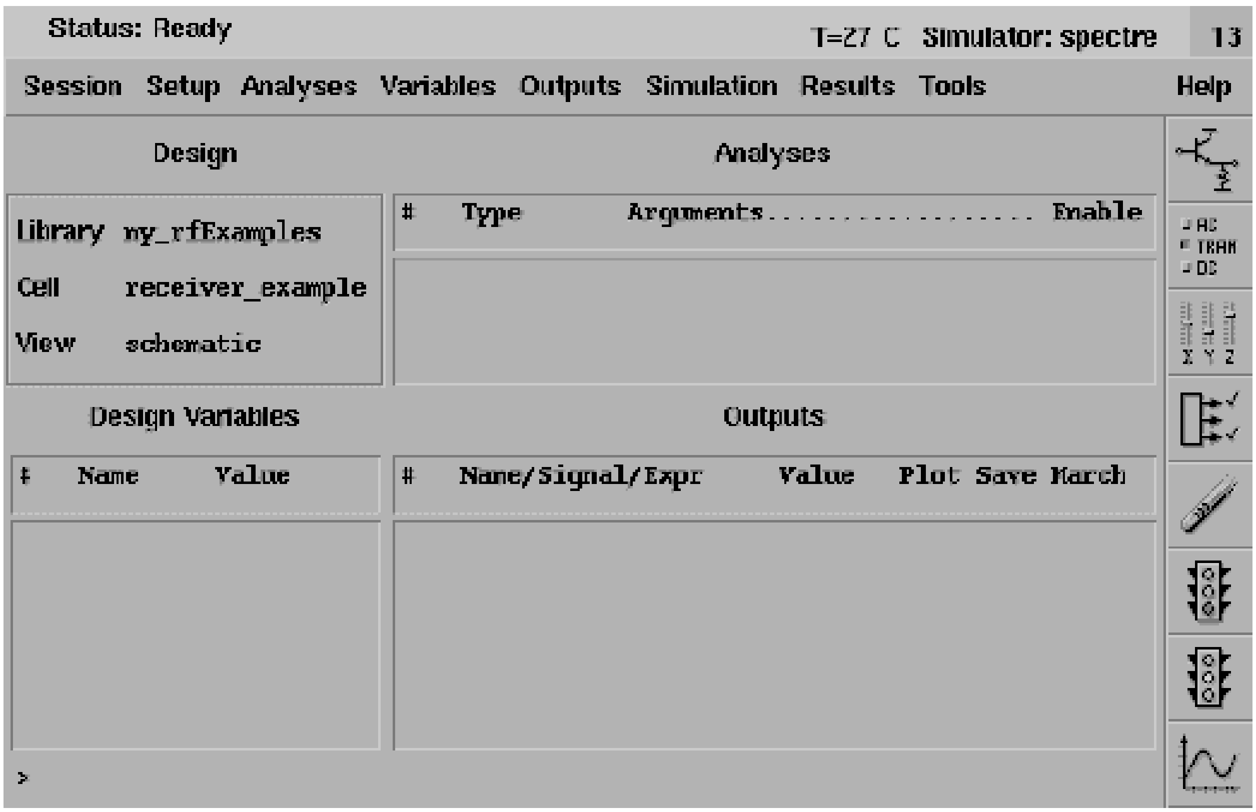

3. Click *OK*.

   An empty Schematic window for the *receiver_example* appears.

## Opening the Analog Environment

**1.** In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.




The *Library*, *Cell*, and *View* names appear in the *Design* section of the ADE window.

**2.** Set the simulator options from the Simulator window as described in “Choosing Simulator Options” on page 708.

**3.** Set up the model libraries from the Simulator window as described in “Setting Up Model Libraries” on page 710.

## Constructing the Baseband Model for the Receiver

Construct the receiver in the Schematic window by adding blocks from left to right, from input to output, as listed in Table 9-1.

Except for the resistor, ground, and port models (which come from the *analogLib*), all blocks come from the *rfLib*. Unless otherwise instructed, leave the port resistances at their default value of 50 Ohms.

**Table 9-1 Blocks Used to Create the Receiver**

| Block Name and Reference | Element Name | Library and Category |
|---|---|---|
| CDMA signal source — See Adding the CDMA Signal Source | `CDMA_reverse_xmit` | From the *measurement* category in *rfLib*. |
| Resistor—attach to CDMA signal source | `res` | From *analogLib* |
| Driver — See Adding the Driver | `BB_driver` | From the *measurement* category in *rfLib*. |
| Low noise amplifier — See Adding the Low Noise Amplifier | `LNA_BB` | From the *top_dwnBB* category in *rfLib*. |
| Butterworth bandpass filter — See Adding a Butterworth Band Pass Filter | `BB_butterworth_bp` | From the *top_dwnBB* category in *rfLib* |
| RF-to-IF mixer — See Adding an RF-to-IF Mixer | `dwn_cnvrt` | From the *top_dwnBB* category in *rfLib* |
| Butterworth bandpass filter — See Adding Another Butterworth Bandpass Filter | `BB_butterworth_bp` | From the *top_dwnBB* category in *rfLib* |
| IQ demodulator — See Adding an IQ Demodulator | `IQ_demod_BB` | From the *top_dwnBB* category in *rfLib* |
| Butterworth lowpass filters (create two) — See Adding Two Butterworth Lowpass Filters | `butterworth_lp` | From the *top_dwnPB* category in *rfLib* |
| Instrumentation model — See Adding an Instrumentation Block | `offset_comms_instr` | From the *measurement* category in *rfLib*. |
| Terminator — See Adding an Instrumentation Terminator | `instr_term` | From the *measurement* category in *rfLib* |
| Grounds—attach to RF-to-IF mixer, IQ demodulator, and Instrumentation model | `gnd` | From *analogLib* |

### Adding the CDMA Signal Source

Add the first receiver block, a CDMA signal source (`CDMA_reverse_xmit`), to the schematic.

1. In the Schematic window, choose *Add – Instance*.

   The Add Instance form appears. It may be empty or it may display information for a previously added element. The default for the *View* field is `symbol`.

2. In the Add Instance form, click *Browse*.

   The Library Browser - Add Instance form appears.

3. In the Library Browser - Add Instance form,

   a. If necessary, click *Show Categories* to display the *Category* column so you can view the elements (or cells) in the *rfLib* by category.

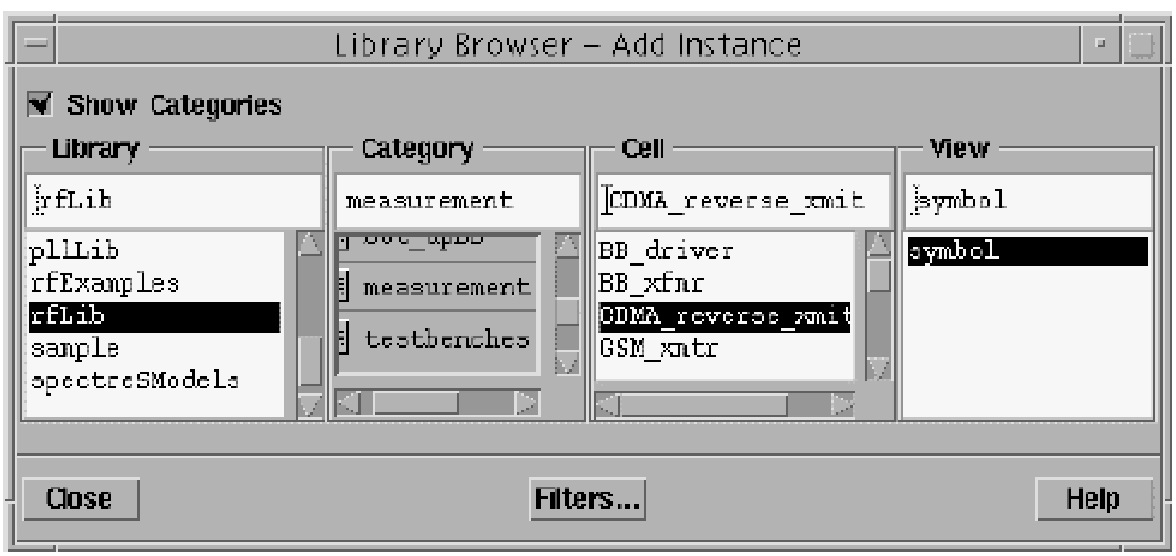
   b. In the *Libraries* column, click *rfLib* to display categories of elements in *rfLib*.

      The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column. (In the Add Instance form, *rfLib* displays in the *Library* field.)

   c. In the *Category* column, click *measurement* to list only the cells in the *measurement* category.

   d. In the *Cell* column, click *CDMA_reverse_xmit*.

   In the Library Browser, cell *CDMA_reverse_xmit* and it's default view *symbol* are both selected.

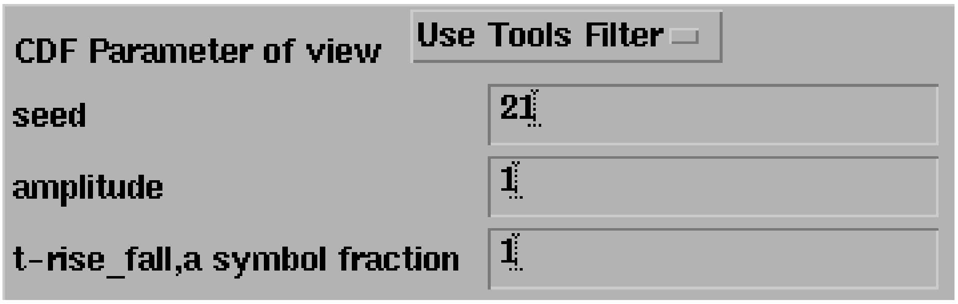



In the Add Instance form,

- ❑ `rfLib` appears in the *Library* field
- ❑ `CDMA_reverse_xmit` displays in the *Cell* field

- `symbol` displays in the *View* field

The CDF parameters for the element and their default values appear at the bottom of the form.




**4.** To place a *CDMA_reverse_xmit* block in the schematic,

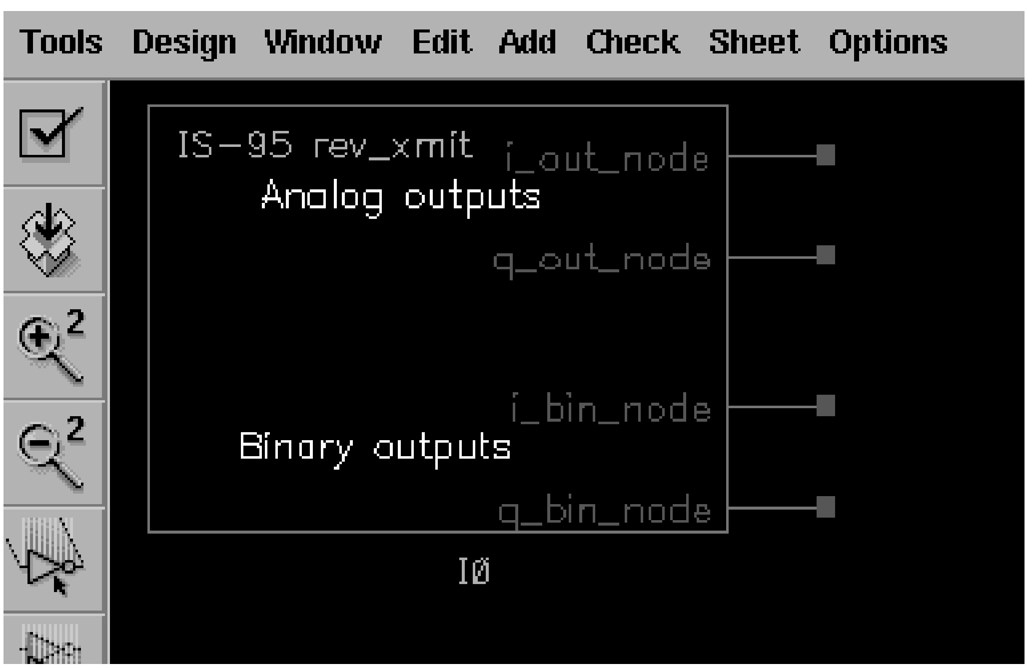
**a.** Move the cursor over the Schematic window.

The outline for the *CDMA_reverse_xmit* symbol is attached to the cursor.

**b.** Move the cursor near the top left corner of the schematic and click to place the *CDMA_reverse_xmit* block.

This block models a CDMA signal source.

**c.** Click *Esc* to remove the symbol from the cursor.



### Adding a Resistor to the CDMA Signal Source

Because this example does not use the binary output nodes (`i_bin_node` and `q_bin_node`) on the CDMA signal source, connect a resistor between these nodes to avoid unused pin warnings.

1. In the *Libraries* column of the Library Browser - Add Instance form, click *analogLib* to display elements in *analogLib*.

   If *Show Categories* is selected, the *Everything* category is displayed by default and all cells in *analogLib* are listed in the *Cells* column.

2. Scroll through the list of cells in *analogLib* to locate the resistor cell, *res*.

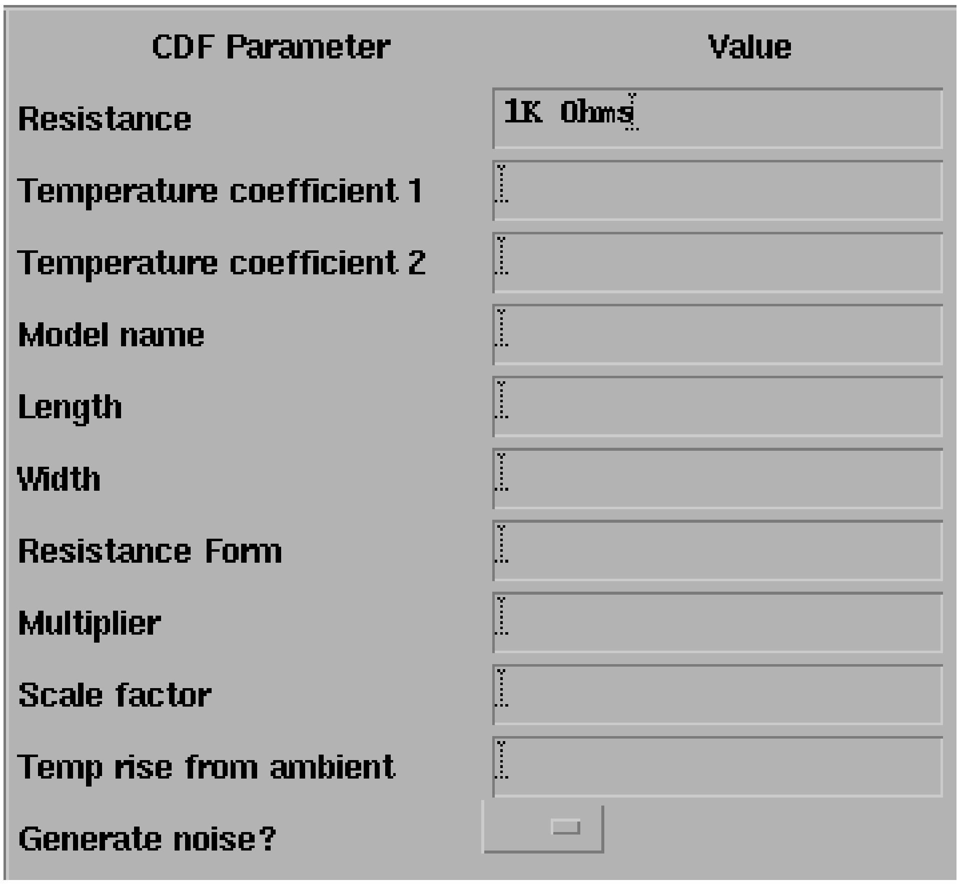
3. Click *res* in the *Cell* column.

   The cell *res* and it's default view *symbol* are both selected.


In the Add Instance form,

- ❑ `analogLib` displays in the *Library* field
- ❑ `res` displays in the *Cell* field
- ❑ and `symbol` displays in the *View* field

The CDF parameters for the element and their default values appear at the bottom of the form.




4. Move the cursor over the Schematic window.
5. Click to place the top resistor terminal in line with the top binary output node (`i_bin_out`) on the lower right side of the *CDMA_reverse_xmit* block
6. Click *Esc* to remove the symbol from the cursor.

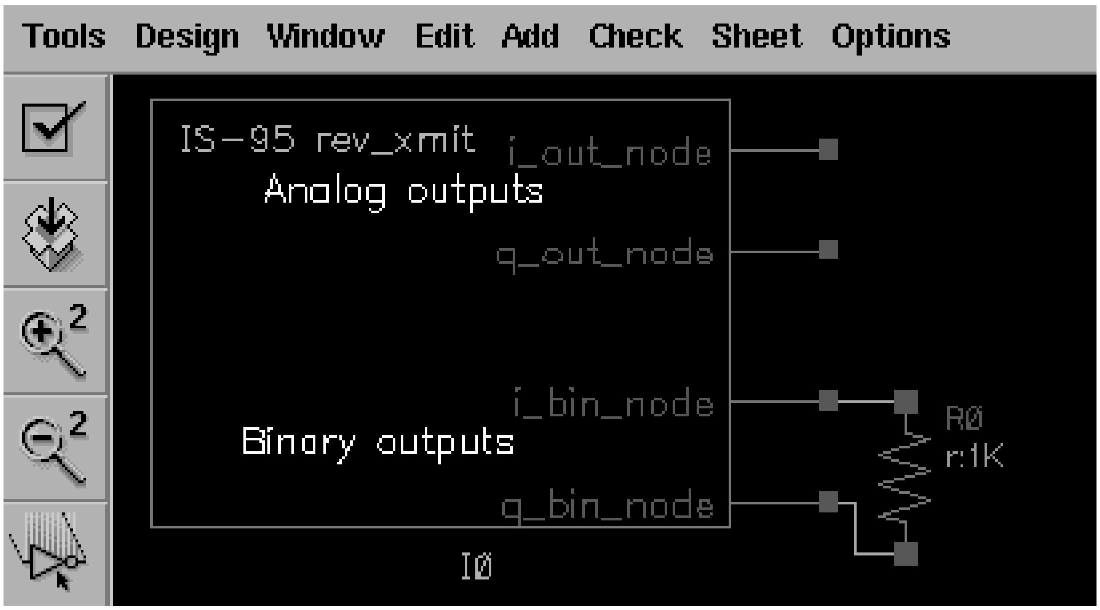
### Wiring the Resistor to the CDMA Signal Source

Wire the resistor to the binary outputs, `i_bin_node` and `q_bin_node`, of the CDMA signal source.

1. To wire the resistor to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `i_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the top node of the resistor.

3. Click `q_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the bottom node of the resistor.
4. Click *Esc* to stop wiring.

   The CDMA signal source and resistor wired together appear as follows.




### Adding the Driver

Add a driver block to the right of the CDMA signal source block.

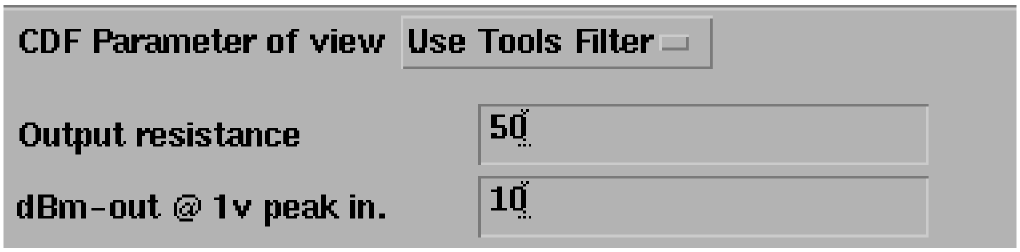
1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the following selections.

| Library | Category | Cell | View |
|---|---|---|---|
| *rfLib* | *measurement* | *BB_driver* | *symbol* |

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field,

- `BB_driver` displays in the *Cell* field and
- `symbol` displays in the *View* field.
- The CDF parameters and their default values appear at the bottom of the Add Instance form.




4. Move the cursor over the Schematic window.

5. Click to place the *BB_driver* to the right of the *CDMA_reverse_xmit* block. Align the input pins of the driver with the analog output pins of the CDMA signal source.

6. Click *Esc* to remove the symbol from the cursor.

#### *Wiring the Signal Source to the Driver*

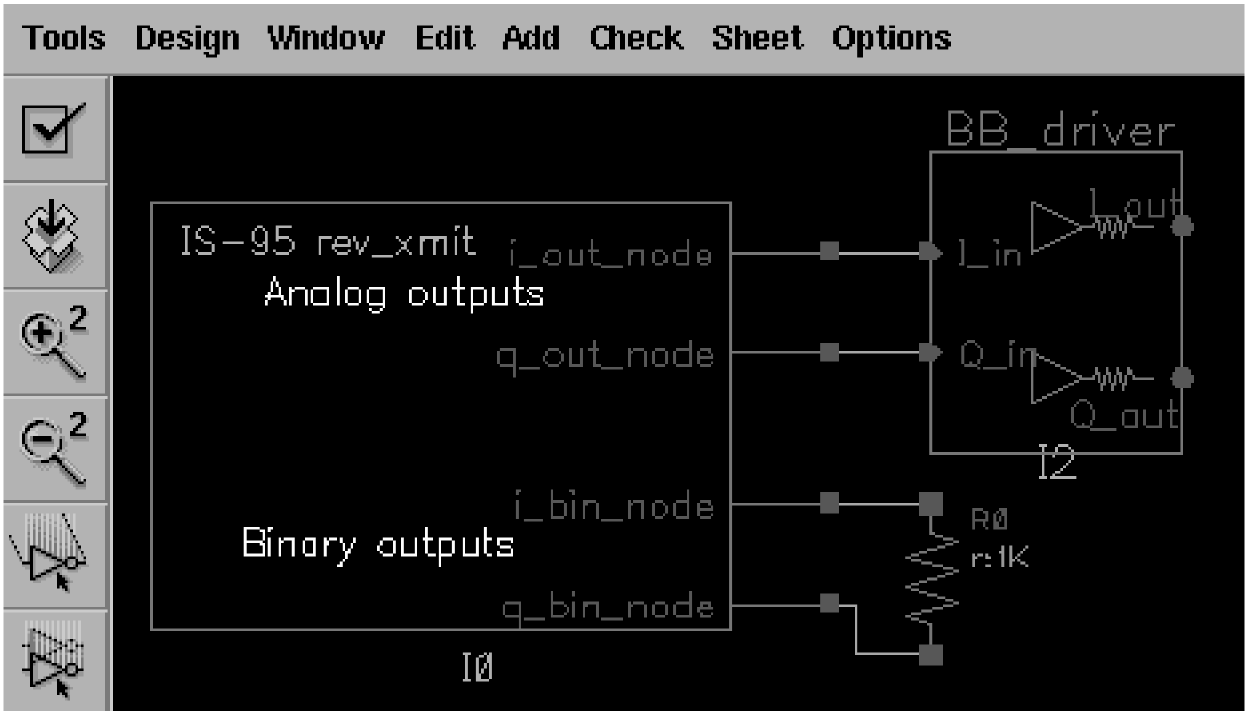
1. To wire the *BB_driver* block to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add – Wire (narrow)*.

2. Click `i_out_node` on the *CDMA_reverse_xmit* block then click `I_in` on the *BB_driver* block.

3. Click `q_out_node` on the *CDMA_reverse_xmit* block then click `Q_in` on the *BB_driver block*.

4. Click *Esc* to stop wiring.

The schematic now appears as follows.




***Modifying Parameter Values for the Driver***

Edit the value of the *BB_driver* CDF parameter *dBm-out@1v peak in driver* as follows.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the values of CDF (component description format) properties for the driver and modify the schematic for this simulation.

2. In the Schematic window, click the *BB_driver* block.
3. The Edit Object Properties form changes to display information for the *BB_driver* block
4. Change the *dBm-out@1v peak in* parameter value as follows.

| Parameter Name | Value |
|---|---|
| *dBm-out@1v peak in* | *-16* |

The driver converts 1 peak volt from the CDMA signal source to –16 dBm referenced to the output resistance of the driver.

5. Click *OK* in the Edit Object Properties form.

   The form closes.

### Adding the Low Noise Amplifier

Add a low noise amplifier to the right of the driver.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click *Show Categories* at the top of the Library Browser, to display the *Category* column.)

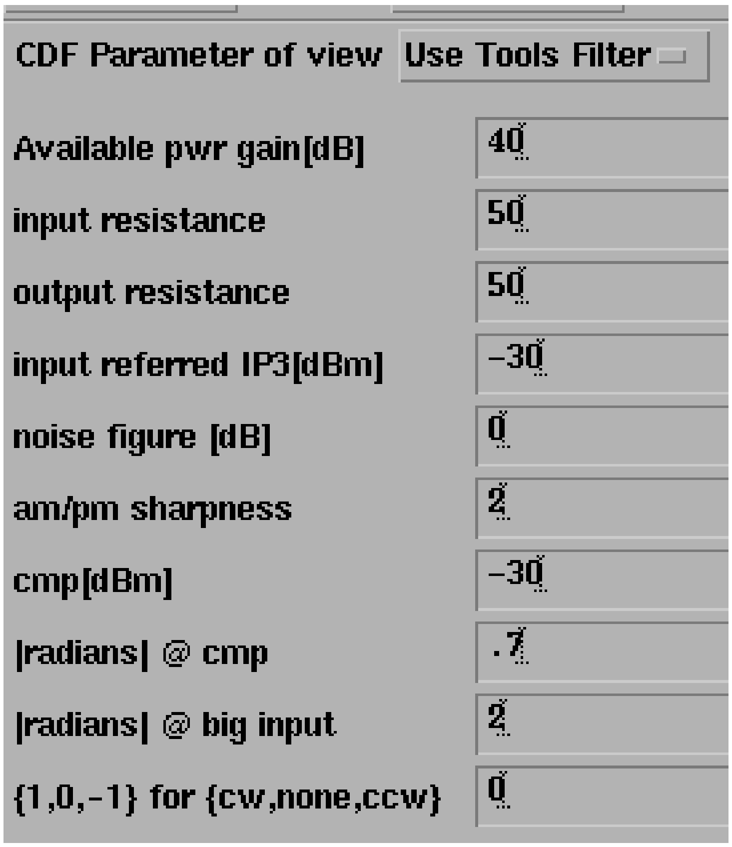
| Library | Category | Cell | View |
|---|---|---|---|
| rfLib | top_dwnBB | LNA_BB | symbol |

In the Library Browser, cell *LNA_BB* and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `LNA_BB` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field

The CDF parameters for the element and their default values display at the bottom of the LNA_BB Add Instance form.

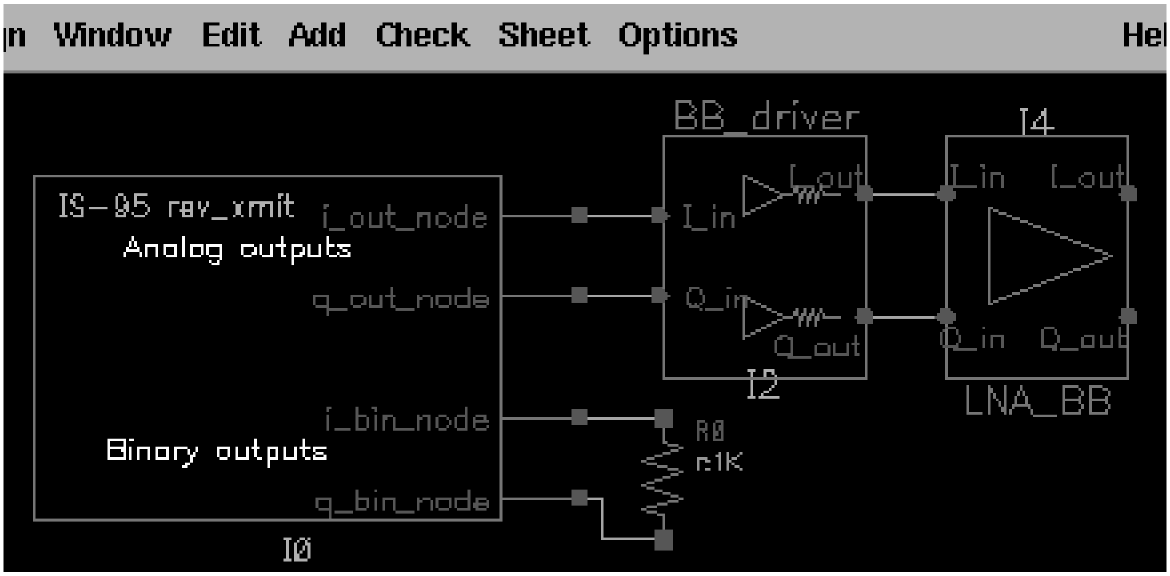



4. Move the cursor over the Schematic window. The outline for the LNA symbol is attached to the cursor. Align the input pins of the LNA with the output pins of the driver.
5. Click to place the *LNA_BB* block to the right of the *BB_driver* block.
6. Click *Esc* to remove the symbol from the cursor.

### Wiring the Driver to the LNA

1. To wire the *LNA_BB* block to the *BB_driver* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *BB_driver* block then click `I_in` on the *LNA_BB* block.
3. Click `Q_out` on the *BB_driver* block then click `Q_in` on the *LNA_BB* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.




### *Modifying Parameter Values for the LNA*

**1.** Edit the CDF parameter values for the LNA.

**a.** Choose *Edit – Properties – Objects* in the Schematic window.

The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the LNA and modify the schematic for this simulation.

**b.** In the Schematic window, click the *LNA*.

The Edit Object Properties form changes to display information for the LNA.

**c.** Change the CDF parameter values for the LNA as follows.

| Parameter Name | Value |
|---|---|
| *Available pwr gain [dB]* | `lna_gain` |
| *Input resistance* | `50` |
| *Output resistance* | `300` |
| *Input referred IP3 [dBm]* | `lna_ip3` |
| *Noise figure [dB]* | `10` |

| Parameter Name | Value |
|---|---|
| *am/pm sharpness* | `2` |
| *cmp [dBm]* | `lna_ip3` |
| *\|radians\| @ cmp* | `.05` |
| *\|radians\| @ big input* | `.7` |
| *{1, 0, -1} for {cw, none, ccw}* | `1` |

**2.** Click *OK* in the Edit Object Properties form.

The form closes.

### Adding a Butterworth Band Pass Filter

Add a Butterworth band pass filter to the right of the low noise amplifier.

**1.** In the Schematic window, choose *Add – Instance* to display the Add Instance form.

Information for the LNA is still displayed in the form.

**2.** In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

**3.** In the Library Browser - Add Instance form, make the following selection.

| Library | Category | Cell | View |
|---|---|---|---|
| *rfLib* | *top_dwnBB* | *BB_butterworth_bp* | *symbol* |

In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

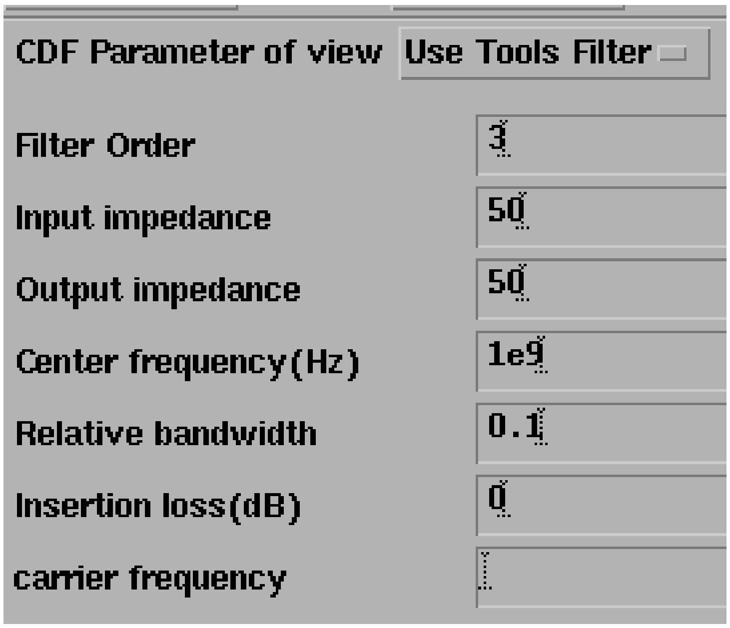
At the top of the Add Instance form,

❑ `rfLib` displays in the *Library* field

❑ `BB_butterworth_bp` displays in the *Cell* field

❑ `symbol` displays in the *View* field.

❑ The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.




**4.** Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.

**5.** Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.

**6.** Click *Esc* to remove the symbol from the cursor.

### Wiring the LNA to the Filter

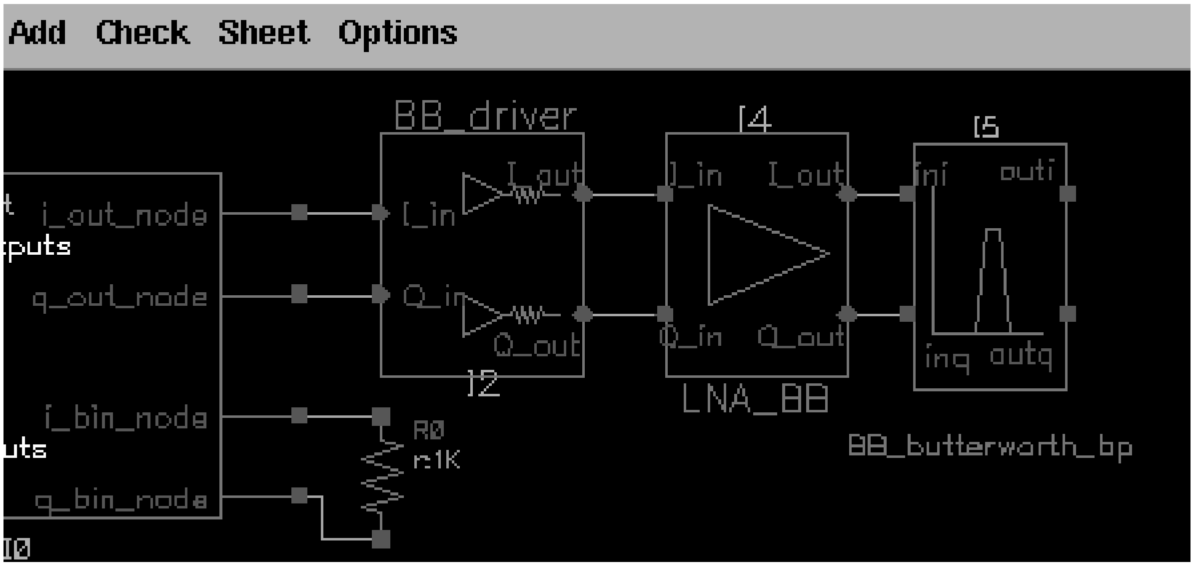
**1.** To wire the *BB_butterworth_bp* block to the *LNA_BB* block, in the Schematic window choose *Add – Wire (narrow)*.

**2.** Click `I_out` on the *LNA_BB* block then click `ini` on the *BB_butterworth_bp* block.

**3.** Click `Q_out` on the *LNA_BB* block then click `inq` on the *BB_butterworth_bp* block.

**4.** Click *Esc* to stop wiring.

The schematic now appears as follows.




### *Modifying Parameter Values for the Band Pass Filter*

Edit the CDF parameter values for the Butterworth band pass filter.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

2. In the Schematic window, click the filter.

   The Edit Object Properties form changes to display information for the filter.

3. Edit the parameter values to match those in Table 9-2.

4. Click *OK*.

**Table 9-2 CDF Parameter Values for the Butterworth Filter**

| Parameter Name | Value |
|---|---|
| *Filter order* | `3` |
| *Input impedance* | `50` |
| *Output impedance* | `50` |
| *Center frequency (Hz)* | `frf` |

**Table 9-2 CDF Parameter Values for the Butterworth Filter**

| Parameter Name | Value |
|---|---|
| *Relative bandwidth* | `rf_rbw` |
| *Insertion loss (dB)* | `3` |
| *Carrier frequency* | `frf` |

Specify the *Carrier frequency* parameter value for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure 9-6 on page 721, the carrier frequency is used to compute speed voltages. Because filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its *Carrier frequency* parameter value is the IF frequency.

- The *Carrier frequency* is the frequency value to which the baseband signals are referenced.
- The *Center frequency* is the frequency for which a filter is designed. The *Center frequency* parameter value for a bandpass filter does not have to equal the *Carrier frequency* parameter value.

### Adding an RF-to-IF Mixer

Add an RF-to-IF mixer (*dwn_cnvrt*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the selections indicated in Table 9-3.

**Table 9-3 Library Browser selections for the RF to IF Mixer**

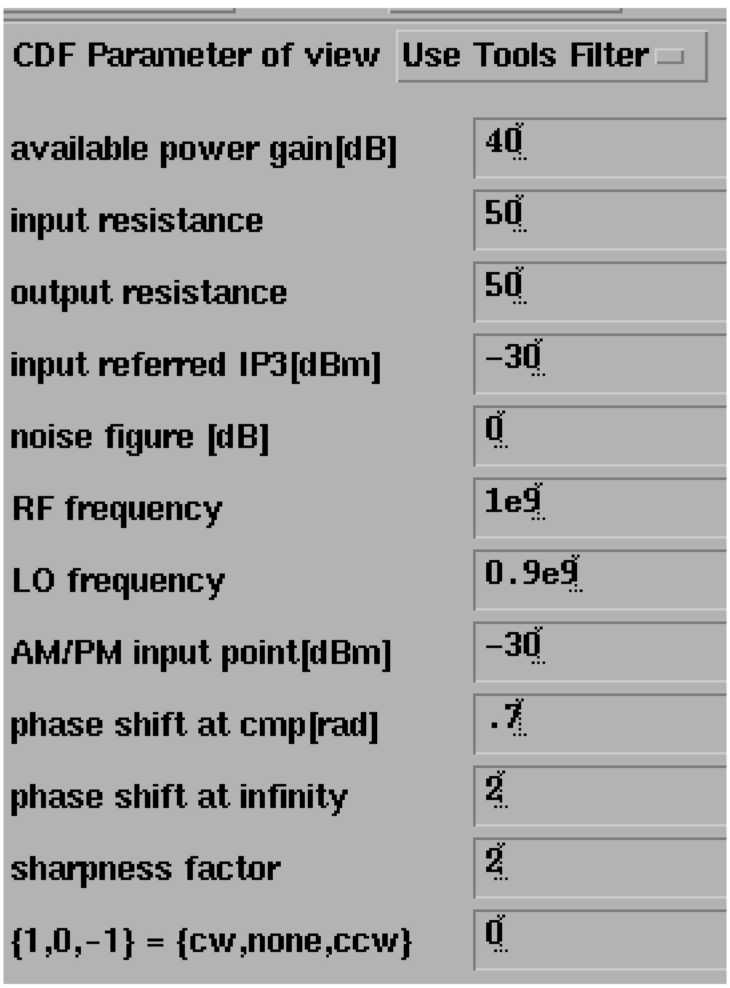
| Library | Category | Cell | View |
|---|---|---|---|
| *rfLib* | *top_dwnBB* | *dwn_cnvrt* | *symbol* |

In the Library Browser, cell *dwn_cnvrt* (the RF-to-IF mixer) and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `dwn_cnvrt` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field.

The CDF parameters for the down converter and their default values display at the bottom of the form.



| | |
|---|---|
| CDF Parameter of view | Use Tools Filter |
| available power gain[dB] | 40 |
| input resistance | 50 |
| output resistance | 50 |
| input referred IP3[dBm] | -30 |
| noise figure [dB] | 0 |
| RF frequency | 1e9 |
| LO frequency | 0.9e9 |
| AM/PM input point[dBm] | -30 |
| phase shift at cmp[rad] | .7 |
| phase shift at infinity | 2 |
| sharpness factor | 2 |
| {1,0,-1} = {cw,none,ccw} | 0 |

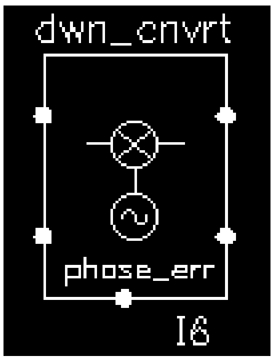
4. Move the cursor over the Schematic window. The outline for the RF-to-IF Mixer symbol is attached to the cursor. Align the input pins of the mixer with the output pins of the filter.

5. Click to place the *dwn_cnvrt* block to the right of the *BB_butterworth_bp* block.

6. Click *Esc* to remove the symbol from the cursor.

### *Grounding the phase_err Pin on the Mixer*

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the RF-to-IF mixer.




1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

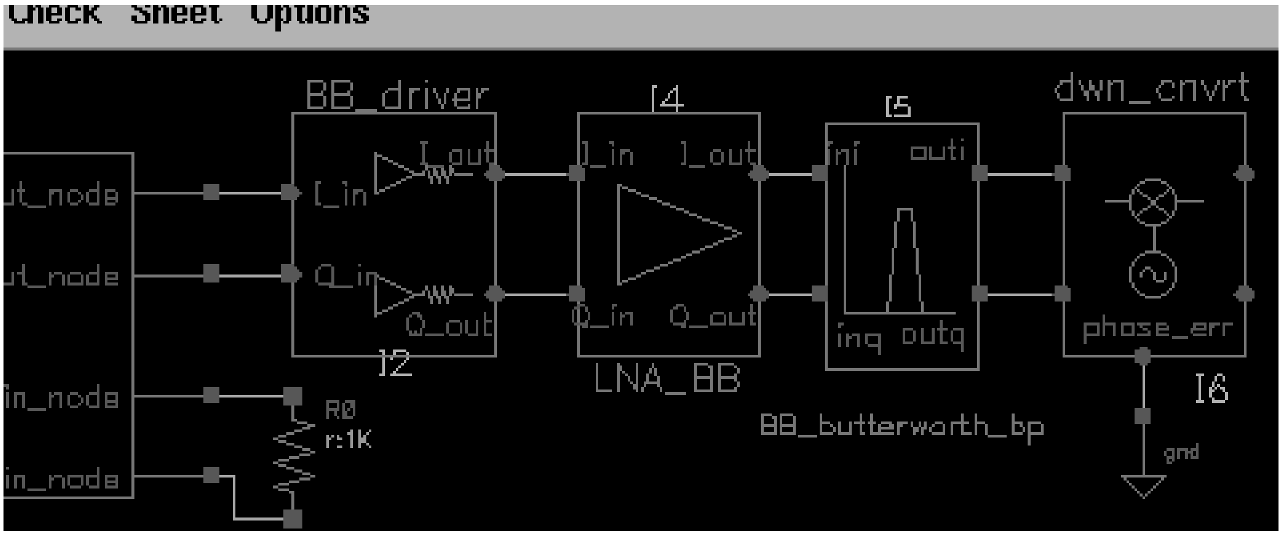
2. In the Add Instance form, type
   - `analogLib` in the *Library* field
   - `gnd` in the *Cell* field
   - `symbol` in the *View* field.
3. Move the cursor over the Schematic window.
4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *dwn_cnvt* block.
5. Click *Esc* to remove the symbol from the cursor.

### Wiring the Filter and Ground to the Mixer

1. To wire the *dwn_cnvrt* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *dwn_cnvrt* block.
3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *dwn_cnvrt* block.
4. Click the port on the *gnd* block then click `phase_err` on the *dwn_cnvrt* block.

**5.** Click *Esc* to stop wiring.

The schematic now appears as follows.




### *Modifying Parameter Values for the RF-to-IF Mixer*

**1.** Edit the CDF parameter values for the RF-to-IF mixer (*dwn_cnvrt*) as listed in Table 9-4 on page 741.

**a.** Choose *Edit – Properties – Objects* in the Schematic window.

The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *dwn_cnvrt* and modify the schematic for this simulation.

**b.** In the Schematic window, click *dwn_cnvrt*.

The Edit Object Properties form changes to display information for *dwn_cnvrt*.

**c.** Change the parameter values to match those in Table 9-4.

**d.** Click OK.

**Table 9-4 CDF Parameter Values for the RF-to-IF Mixer**

| Parameter Name | Value |
|---|---|
| available power gain[dB] | `if_mx_gain` |
| Input resistance | `50` |

**Table 9-4 CDF Parameter Values for the RF-to-IF Mixer**

| Parameter Name | Value |
|---|---|
| output resistance | `50` |
| input referred ip3[dBm] | `if_mx_ip` |
| noise figure [dB] | `10` |
| RF frequency | `frf` |
| LO frequency | `flo1` |
| AM/PM input point[dBm] | `-30` |
| phase shift at cmp[rad] | `.7` |
| phase shift at infinity | 2 |
| sharpness factor | 2 |
| {1,0,–1} = {cw,none,ccw} | `0` |

### Adding Another Butterworth Bandpass Filter

Add another Butterworth band pass filter block to the right of the RF-to-IF Mixer block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the following selection.

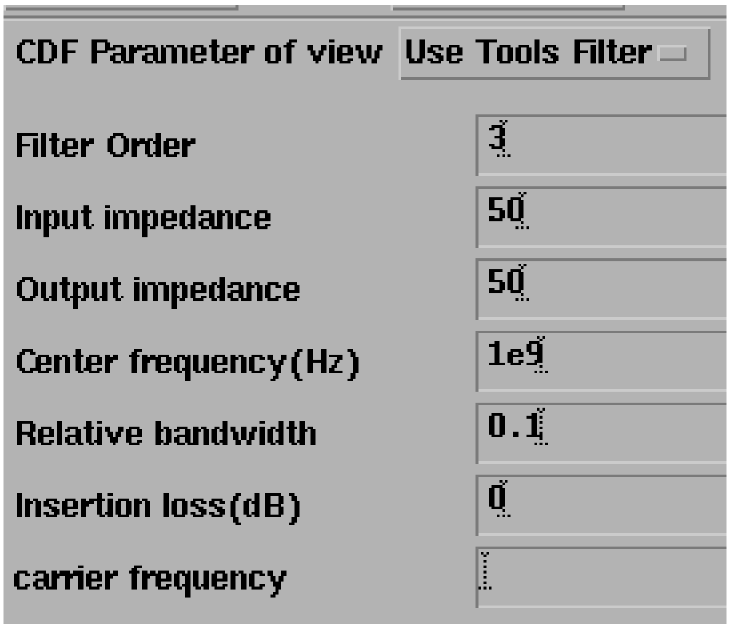
| Library | Category | Cell | View |
|---|---|---|---|
| *rfLib* | *top_dwnBB* | *BB_butterworth_bp* | *symbol* |

In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `BB_butterworth_bp` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field.

The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.

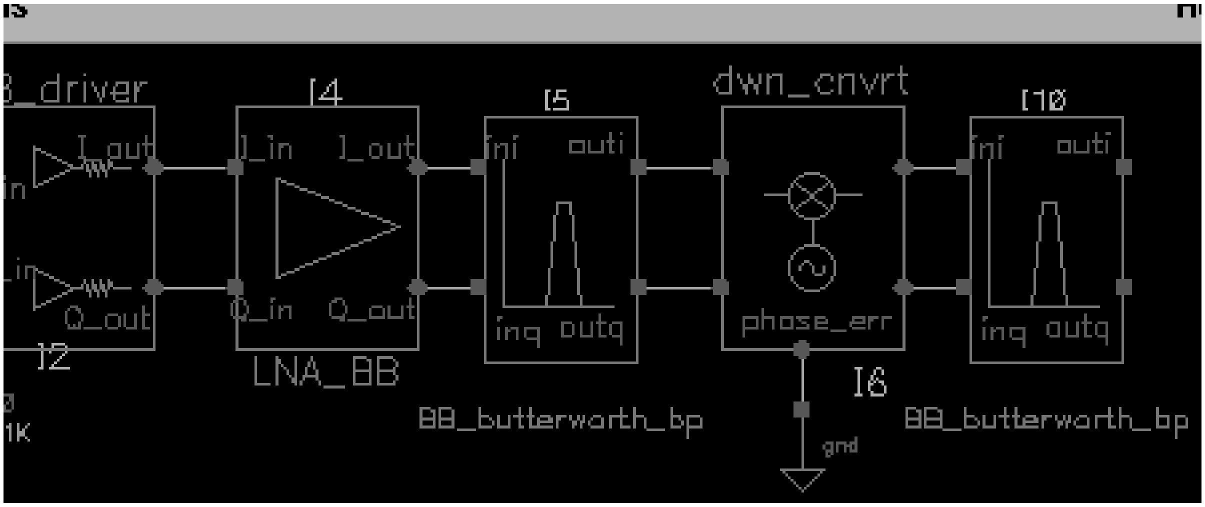



4. Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.
5. Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.
6. Click *Esc* to remove the symbol from the cursor.

### Wiring the Mixer to the Filter

1. To wire the *BB_butterworth_bp* block to the *dwn_cnvrt* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *dwn_cnvrt* block then click `ini` on the *BB_butterworth_bp* block.
3. Click `Q_out` on the *dwn_cnvrt* block then click `inq` on the *BB_butterworth_bp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.




### *Modifying Parameter Values for the Band Pass Filter*

1. Edit the CDF parameter values for the Butterworth band pass filter as listed in Table 9-5 on page 744.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

   b. In the Schematic window, click the filter.

      The Edit Object Properties form changes to display information for the filter.

   c. Change the parameter values to match those in Table 9-5.

   d. Click OK.

**Table 9-5 CDF Parameter Values for the Second Butterworth Filter**

| **Parameter Name** | **Value** |
|---|---|
| Filter Order | `3` |
| Input impedance | `50` |
| Output impedance | `50` |
| Center frequency (Hz) | `-frf+flo1` |

**Table 9-5 CDF Parameter Values for the Second Butterworth Filter**

| Parameter Name | Value |
|---|---|
| Relative bandwidth | `if_rbw` |
| Insertion loss (dB) | `1` |
| Carrier frequency | `–frf+flo1` |

As for the first band pass filter, specify the carrier frequency for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure 9-6 on page 721, the carrier frequency is used to compute speed voltages. Because filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its carrier frequency is the IF frequency. The carrier frequency is the frequency to which the baseband signals are referenced. The center frequency of the bandpass filter does not have to equal the carrier frequency. The center frequency is the frequency for which a filter is designed.

### Adding an IQ Demodulator

Add an IQ Demodulator (*IQ_demod_BB*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form,

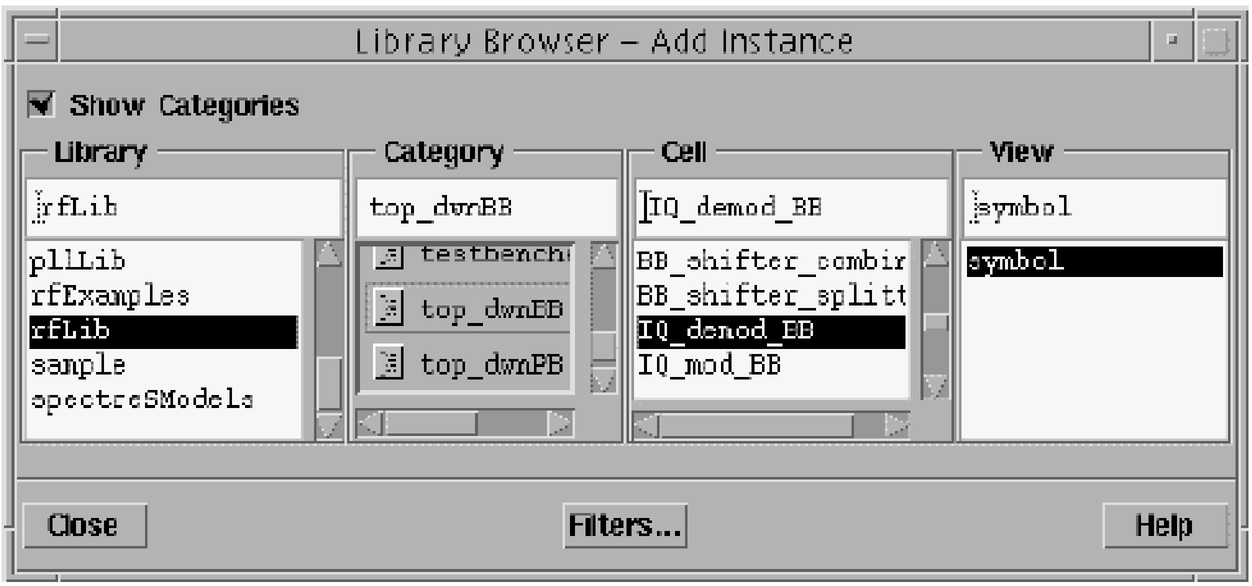
   a. If necessary, click *Show Categories* to display the *Category* column.

   b. Click *rfLib* to display elements in *rfLib*. The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column.

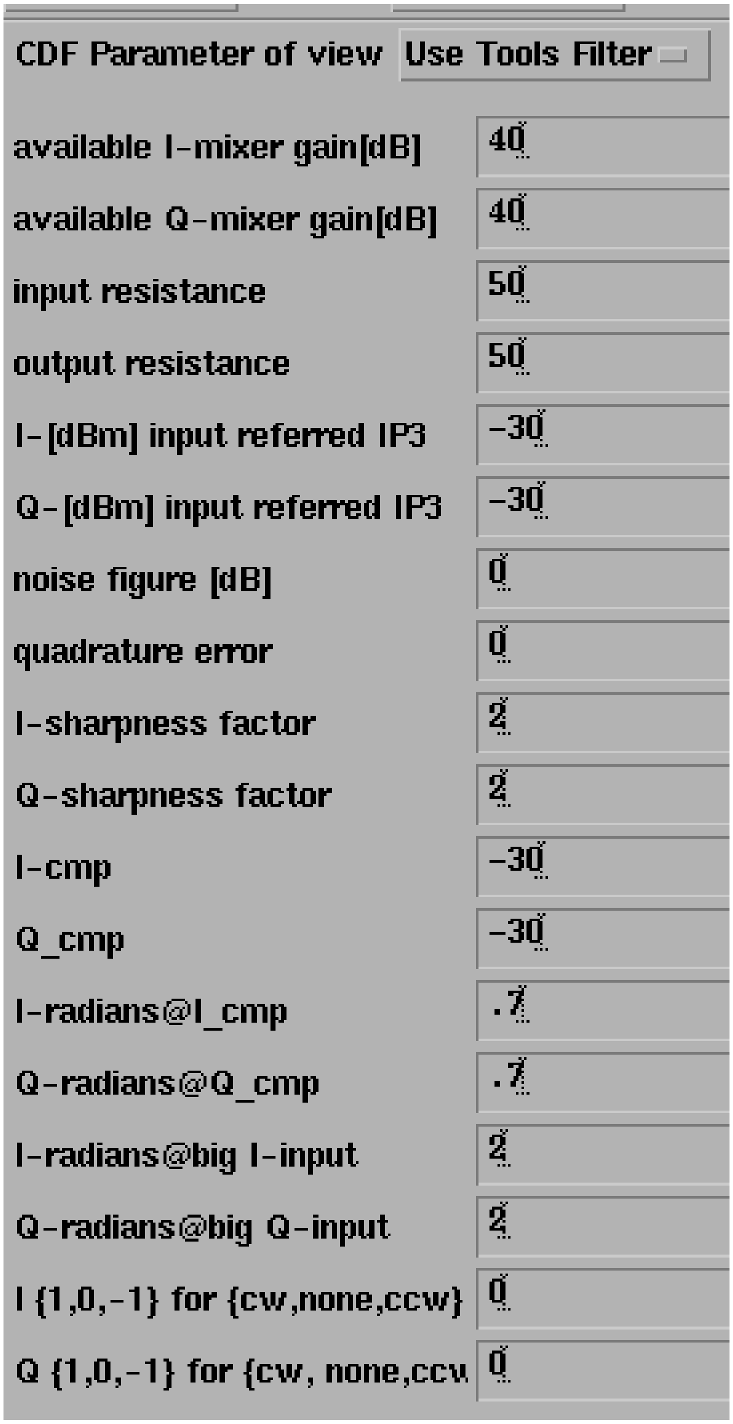
   c. In the *Category* column, click *top_dwnBB* to display cells in the *top_dwnBB* category.

   d. In the *Cell* column, click *IQ_demod_BB*.

In the Library Browser, cell *IQ_demod_BB* (the IQ demodulator) and it's default view *symbol* are both selected.



At the top of the Add Instance form, `rfLib` displays in the *Library* field, `IQ_demod_BB` displays in the *Cell* field and `symbol` displays in the *View* field. The CDF parameters for the IQ demodulator and their default values display at the bottom of the form.



CDF Parameter of view: Use Tools Filter

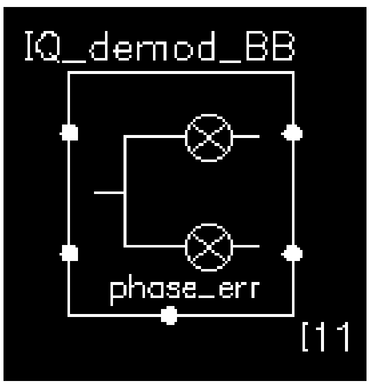
| | |
|---|---|
| available I-mixer gain[dB] | 40 |
| available Q-mixer gain[dB] | 40 |
| input resistance | 50 |
| output resistance | 50 |
| I-[dBm] input referred IP3 | -30 |
| Q-[dBm] input referred IP3 | -30 |
| noise figure [dB] | 0 |
| quadrature error | 0 |
| I-sharpness factor | 2 |
| Q-sharpness factor | 2 |
| I-cmp | -30 |
| Q_cmp | -30 |
| I-radians@I_cmp | .7 |
| Q-radians@Q_cmp | .7 |
| I-radians@big I-input | 2 |
| Q-radians@big Q-input | 2 |
| I {1,0,-1} for {cw,none,ccw} | 0 |
| Q {1,0,-1} for {cw, none,ccv | 0 |

4. Move the cursor over the Schematic window. The outline for the IQ demodulator symbol is attached to the cursor. Align the input pins of the IQ demodulator with the output pins of the filter and click to place the *IQ_demod_BB* block to the right of the second *BB_butterworth_bp* block.

5. Click *Esc* to remove the symbol from the cursor.

#### *Grounding the phase_err Pin on the IQ Demodulator*

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the IQ demodulator.




1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type
   - ❑ `analogLib` in the *Library* field
   - ❑ `gnd` in the *Cell* field
   - ❑ `symbol` in the *View* field.

3. Move the cursor over the Schematic window.

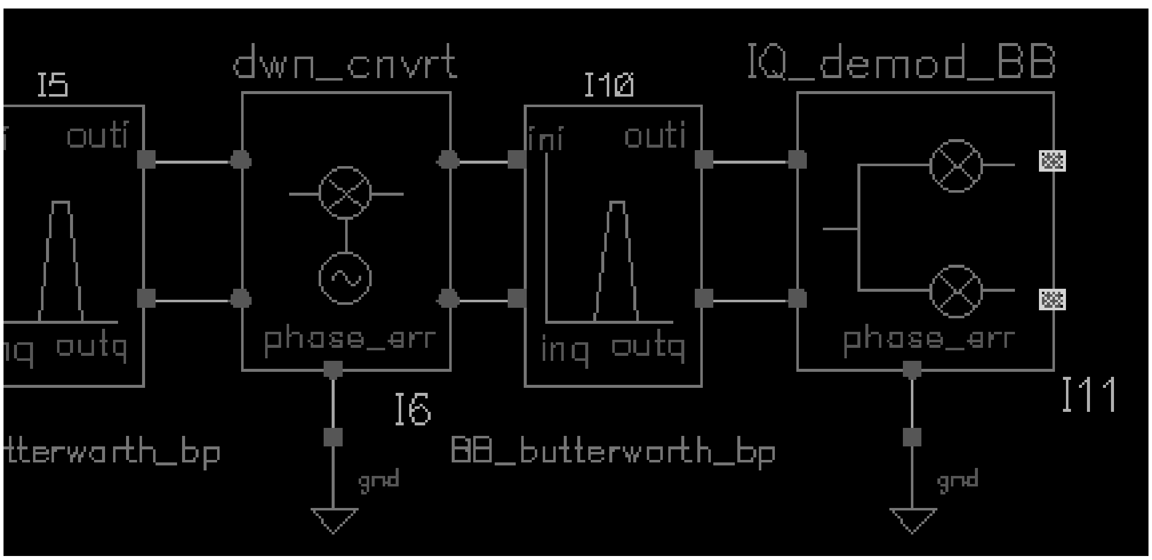
4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *IQ_demod_BB* block.

5. Click *Esc* to remove the symbol from the cursor.

### Wiring the Filter and Ground to the IQ Demodulator

1. To wire the *IQ_demod_BB* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *IQ_demod_BB* block.
3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *IQ_demod_BB* block.
4. Click the port on the *gnd* block then click `phase_err` on the *IQ_demod_BB* block.
5. Click *Esc* to stop wiring.

   The schematic now appears as follows.




### *Modifying Parameter Values for the IQ Demodulator*

1. Edit the CDF parameter values for the IQ Demodulator (*IQ_demod_BB*) as listed in Table 9-6 on page 750.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *IQ_demod_BB* and modify the schematic for this simulation.

   b. In the Schematic window, click *IQ_demod_BB*.

      The Edit Object Properties form changes to display information for *IQ_demod_BB*.

c. Change the parameter values to match those in Table 9-6.

d. Click OK.

**Table 9-6 CDF Parameter Values for the IQ Demodulator**

| Parameter Name | Value |
|---|---|
| available I-mixer gain[dB] | `0` |
| available Q-mixer gain[dB] | `0` |
| Input resistance | `50` |
| output resistance | `50` |
| I–[dBm] input referred IP3 | `40` |
| Q–[dBm] input referred IP3 | `40` |
| noise figure [dB] | `2` |
| quadrature error | `0` |
| I–sharpness factor | `2` |
| Q–sharpness factor | `2` |
| I_cmp | `–30` |
| Q_cmp | `–30` |
| I–radians@I_cmp | `.7` |
| Q–radians@Q_cmp | `.7` |
| I–radians@big I-input | `2` |
| Q–radians@big Q-input | `2` |
| I {1,0,–1} for {cw,none,ccw} | `0` |
| Q {1,0,–1} for {cw,none,ccw} | `0` |

### Adding Two Butterworth Lowpass Filters

1. Add two Butterworth low pass filters to the right of the IQ demodulator block.
   - ❑ Align one filter with the demodulator's `i_out` pin.
   - ❑ Align the other filter with its `q_out` pin.
2. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

3. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

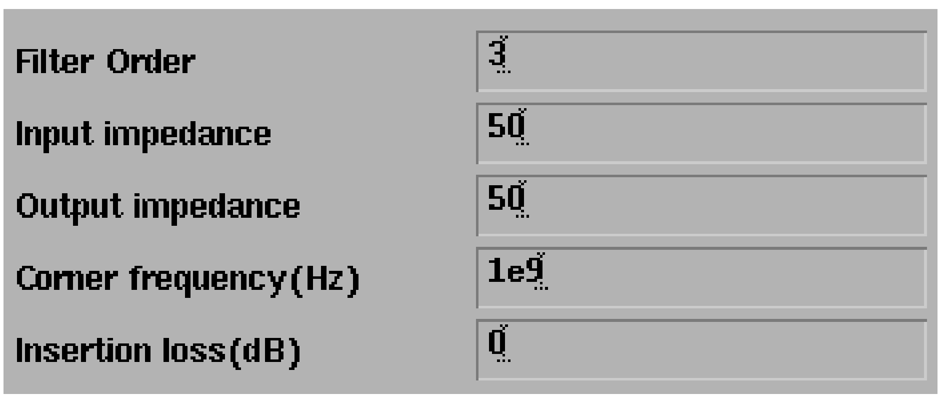
4. In the Library Browser - Add Instance form, make the following selections.

| Library | Category | Cell | View |
|---|---|---|---|
| *rfLib* | *top_dwnPB* | *butterworth_lp* | *symbol* |

In the Library Browser, cell *butterworth_lp* and it's default view are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `butterworth_lp` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field.
- ❑ The CDF parameters for the Butterworth low pass filter and their default values display at the bottom of the Add Instance form.




5. Move the cursor over the Schematic window. The outline for the butterworth low pass filter symbol is attached to the cursor.

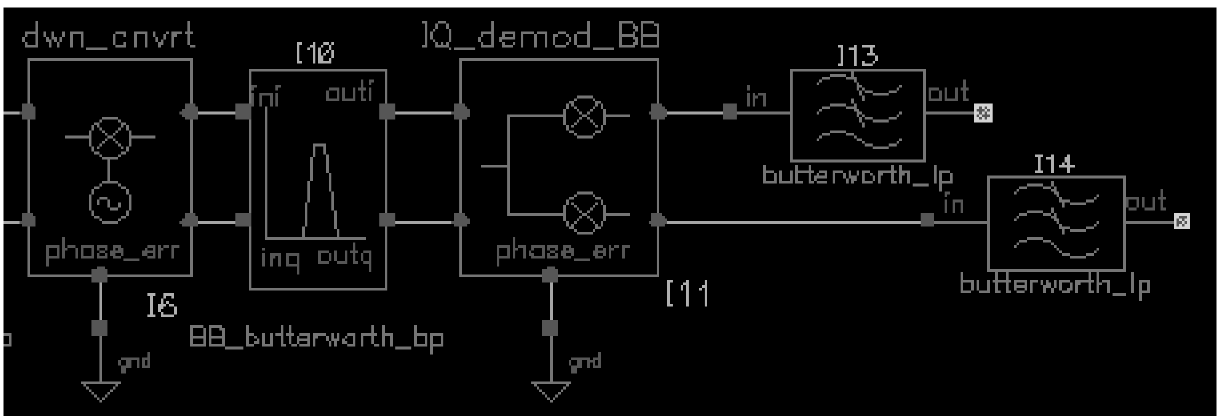
   Align the *in* pin of the first butterworth low pass filter with the *I_out* pin (the top pin) of the IQ demodulator and click to place the filter close to the demodulator. Align the *in* pin of the second butterworth low pass filter with the *Q_out* pin (the bottom pin) of the IQ demodulator. You have to place it further from the demodulator to align it with the *Q_out* pin.

6. Click *Esc* to remove the symbol from the cursor.

### Wiring the IQ Demodulator to the Filters

1. To wire the *IQ_demod_BB* block to the *butterworth_lp* blocks, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *IQ_demod_BB* block then click `in` on the first *butterworth_lp* block.
3. Click `Q_out` on the *IQ_demod_BB* block then click `in` on the second *butterworth_lp* block.
4. Click *Esc* to stop wiring.

   The schematic now appears as follows.




### *Modifying Parameter Values for Both Low Pass Filters*

1. Edit the CDF parameter values for the Butterworth low pass filters as listed in Table 9-7 on page 753.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for each filter and modify the schematic for this simulation.

   b. In the Schematic window, click the first low pass filter.

      The Edit Object Properties form changes to display information for the filter.

**c.** Change the parameter values to match those in Table 9-7.

**Table 9-7 CDF Parameter Values for the Butterworth Low Pass Filters**

| **Parameter Name** | **Value** |
|---|---|
| Filter Order | 3 |
| Input impedance | 50 |
| Output impedance | 50 |
| Corner frequency (Hz) | 10M |
| Insertion loss (dB) | 0 |

**d.** Click Apply.

**e.** In the Schematic window, click the second low pass filter.

The Edit Object Properties form displays the information you entered for the filter as shown in Table 9-7.

**f.** Click OK.

### Adding an Instrumentation Block

Add an instrumentation block (*offset_comms_instr*) block to the right of the low pass filter blocks.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

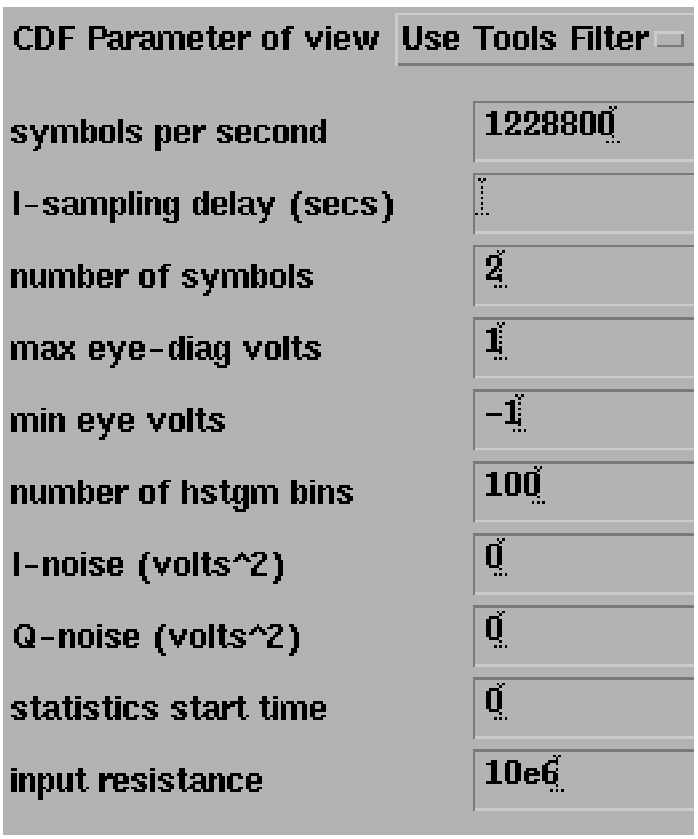
| **Library** | **Category** | **Cell** | **View** |
|---|---|---|---|
| *rfLib* | *measurement* | *offset_comms_instr* | *symbol* |

In the Library Browser, cell *offset_comms_instr* and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library*
- ❑ `offset_comms_instr` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field

The CDF parameters for the *offset_comms_instr* and their default values display at the bottom of the form.

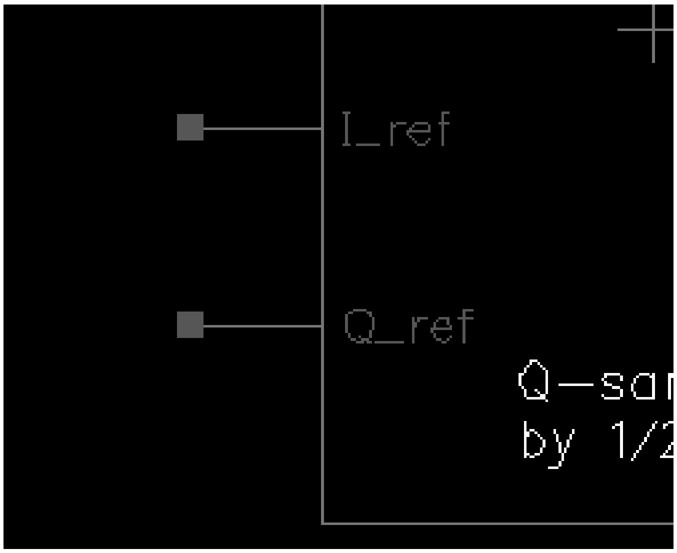



4. Move the cursor over the Schematic window. The outline for the *offset_comms_instr* block symbol is attached to the cursor. Align the *I_in* and *Q_in* pins of the *offset_comms_instr* block with the *out* pins of the butterworth low pass filters and click to place the *offset_comms_instr* block to the right of the low pass filter blocks.

5. Click *Esc* to remove the symbol from the cursor.

#### *Grounding the Reference Pins on the Instrumentation Block*

It is necessary to ground the reference pins (*I_ref* and *Q_ref*) pin near the lower left corner of the instrumentation block.




#### *Grounding the phase_err Pin on the Mixer*

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type

   - `analogLib` in the *Library* field
   - `gnd` in the *Cell* field
   - `symbol` in the *View* field.

3. Move the cursor over the Schematic window.

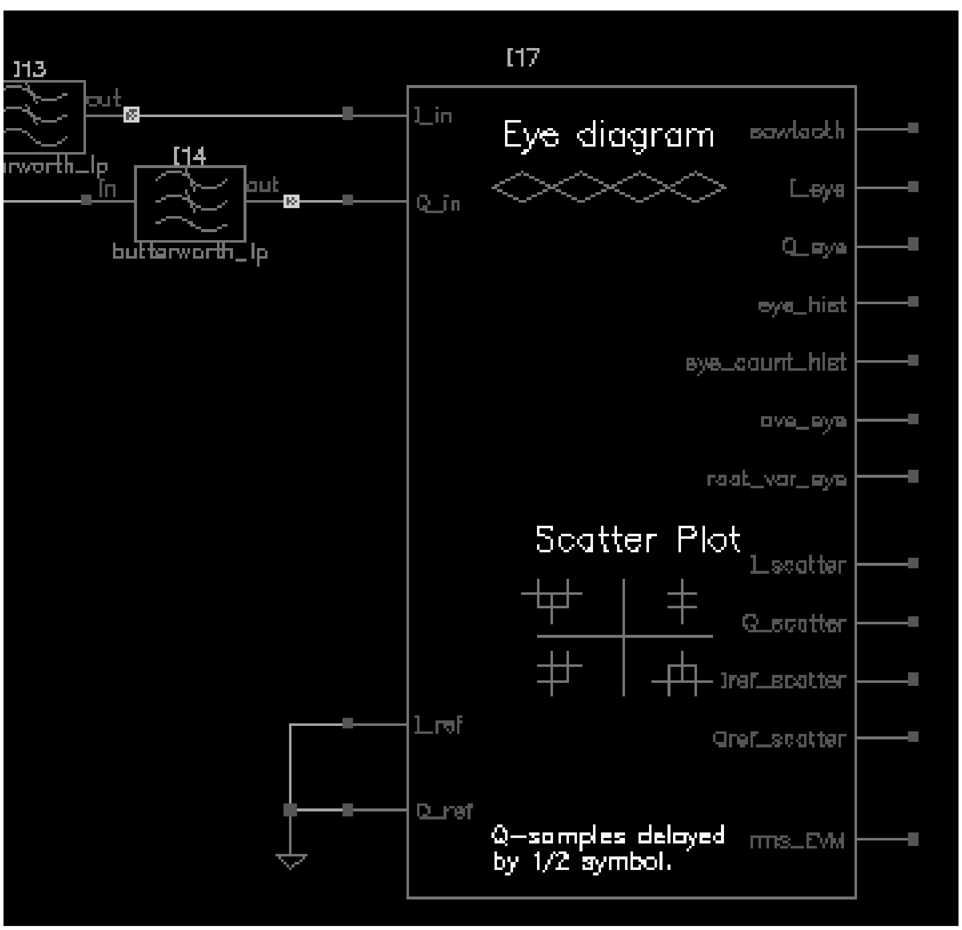
4. Click to place the ground terminal in line with the `Q_ref` node on the bottom of the *offset_comms_instr* block.

5. Click *Esc* to remove the symbol from the cursor.

### Wiring the Filter and Ground to the Instrumentation Block

1. To wire the low pass filters to the *offset_comms_instr* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `out` on the upper *butterworth_lp* block (the low pass filter connected to the `I_out` node on the IQ demodulator) then click `I_in` on the *offset_comms_instr* block.
3. Click `out` on the lower *butterworth_lp* block (the low pass filter connected to the `Q_out` node on the IQ demodulator) then click `Q_in` on the *offset_comms_instr* block.
4. Click the port on the *gnd* block then click the `Q_ref` node on the *offset_comms_instr* block.
5. Click the port on the *gnd* block then click the `I_ref` node on the *offset_comms_instr* block.
6. Click *Esc* to stop wiring.

   The schematic now appears as follows.



#### *Modifying Parameter Values for the Instrumentation Block*

1. Edit the CDF parameter values for the *offset_comms_instr* as listed in Table 9-8 on page 757.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *offset_comms_instr* and modify the schematic for this simulation.

   b. In the Schematic window, click the *offset_comms_instr* block.

      The Edit Object Properties form changes to display information for *offset_comms_instr* block.

   c. Change the parameter values to match those in Table 9-8.

   d. Click OK.

**Table 9-8 CDF Parameter Values for the Instrumentation Block**

| Parameter Name | Value |
|---|---|
| symbols per second | `1228800` |
| I-sampling delay (secs) | `134n` |
| number of symbols | `2` |
| max eye-diag volts | `1` |
| min eye volts | `–1` |
| number of hstgm bins | `100` |
| I-noise (volts$^2$) | `0` |
| Q-noise (volts$^2$) | `0` |
| statistics start time | `30u` |
| input resistance | `50` |

#### Adding an Instrumentation Terminator

Add an instrumentation termination (*instr_term*) block to the right of the instrumentation block. The *instr_term* block terminates the outputs on the instrumentation block and prevents unused pin warnings.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

   | Library | Category | Cell | View |
   |---|---|---|---|
   | *rfLib* | *measurement* | *instr_term* | *symbol* |

   In the Library Browser, cell *instr_term* and it's default view *symbol* are both selected.

   At the top of the Add Instance form,

   ❑ `rfLib` displays in the *Library*

   ❑ `instr_term` displays in the *Cell* field

   ❑ `symbol` displays in the *View* field

   There are no CDF parameters for the *instr_term* cell.

4. Move the cursor over the Schematic window. The outline for the *instr_term* symbol is attached to the cursor. Move the *instr_term* block to the right of the instrumentation block and align the input pins of the instrumentation termination block with the output pins of the instrumentation block.

5. Click to place the *instr_term* block.

6. Click *Esc* to remove the symbol from the cursor.

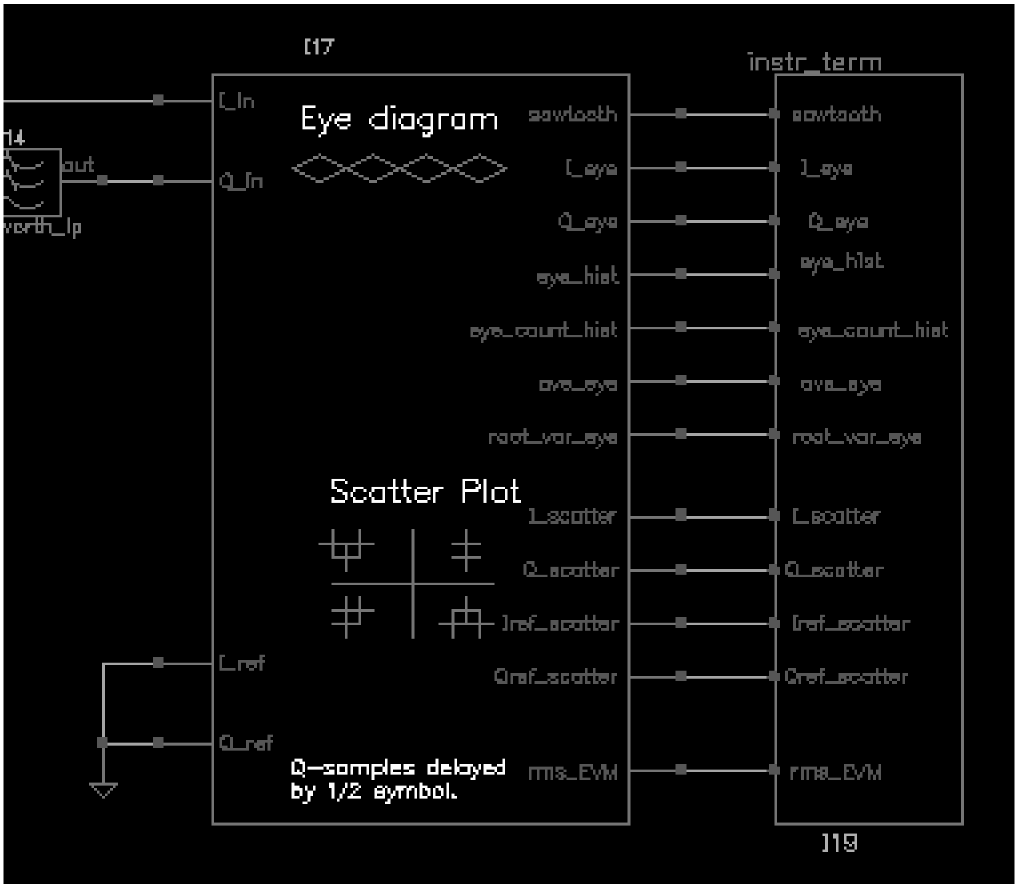
### Wiring the Termination Block to the Instrumentation Block

1. To wire the instrumentation (*offset_comms_instr)* block to the Instrumentation terminator (*instr_term)* block, in the Schematic window choose *Add - Wire (narrow)*.

2. Wire the aligned pins straight across.

3. Click *Esc* to stop wiring.

The schematic should look as follows.

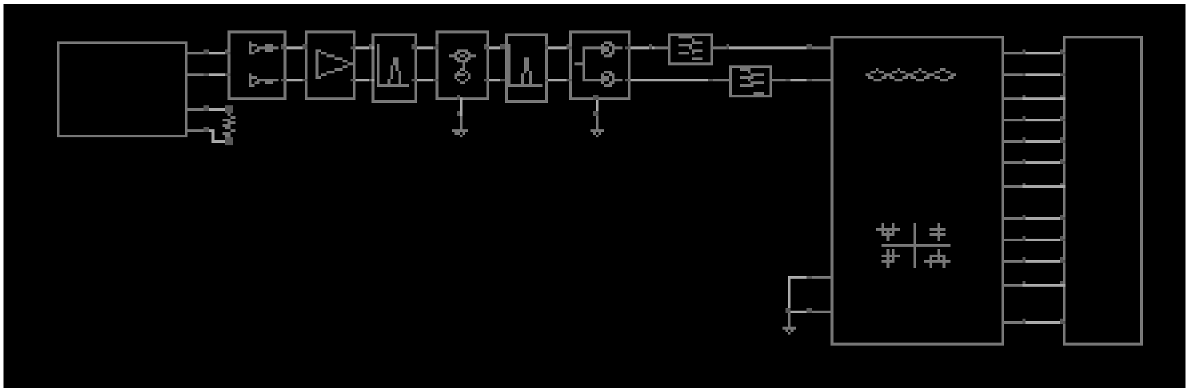



**4.** In the Schematic window, choose *Design – Design Check and Save*.

The completed schematic is verified and saved.

The schematic for the complete receiver model should look like the one in Figure 9-7 on page 760.

**Figure 9-7 Completed receiver model**

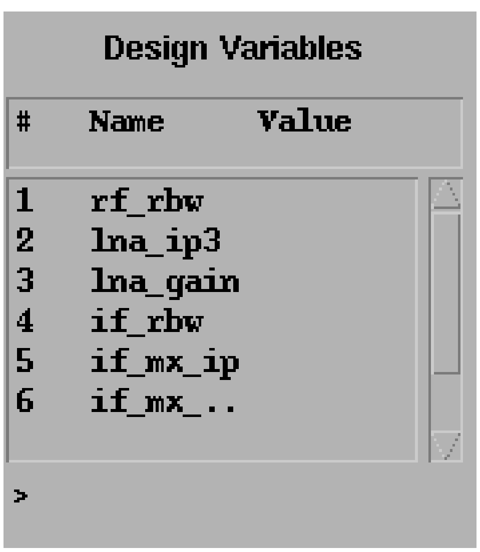


## Setting Variable Values for the Receiver Schematic

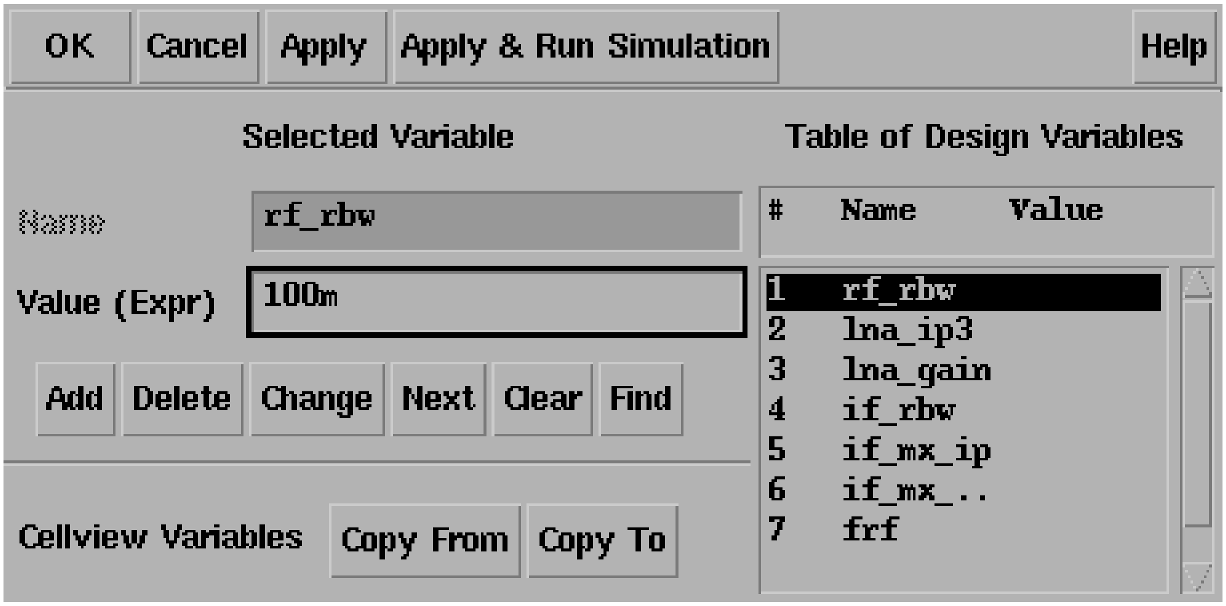
Copy the variables you entered as CDF parameters for the individual blocks from the receiver schematic to the ADE window. Then edit each variable to give it the value specified in Table 9-9 on page 761.

1. In the ADE window, choose *Variables – Copy From Cellview* to copy the variables from the receiver schematic to the Design Variables area on the ADE window.

   The copied variables display in the *Design Variables* area in the ADE window.



**2.** In the ADE window, choose *Variables – Edit* to open the Editing Design Variables form.




### Adding the Values to the Copied Variables

In the Editing Design Variables form, one by one, select each variable in the *Table of Design Variables* and associate with each one, the value listed in Table 9-9.

**Table 9-9 Values for Receiver Variables**

| Variable | Value |
|---|---|
| `lna_gain` | 15 |
| `lna_ip3` | –5 |
| `if_mx_gain` | 10 |
| `if_mx_ip` | 35 |
| `frf` | 2.14G |
| `flo1` | 2.354G |
| `if_rbw` | 200m |
| `rf_rbw` | 100m |

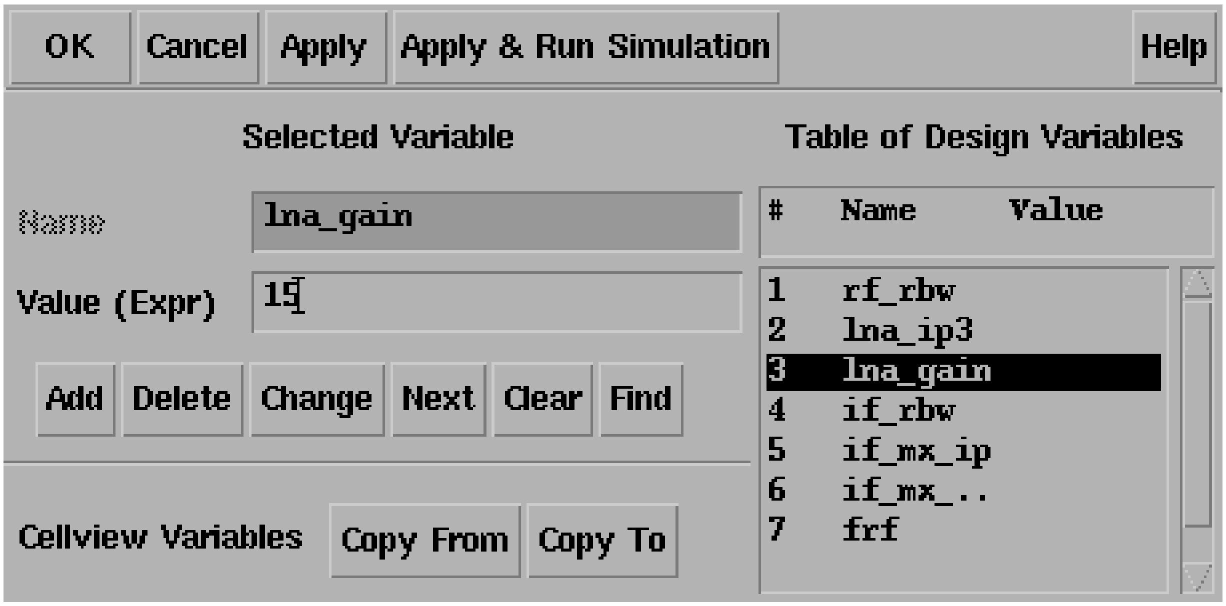
To associate a value with a design variable

1. In the *Table of Design Variables*, click `lna_gain`.

   `lna_gain` displays in the *Name* field.

2. In the *Value (Expr)* field, enter the number `15,` the value from Table 9-9 on page 761.

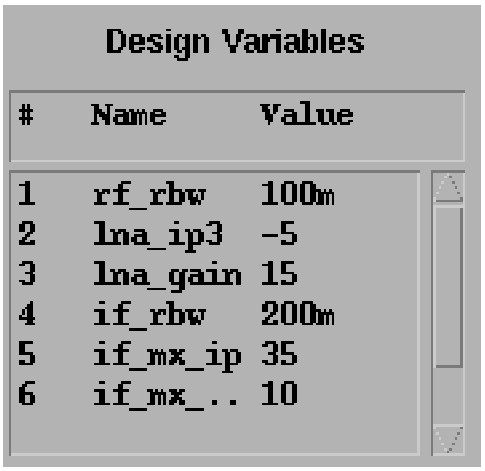
3. Click *Change* to list the variable name and its value from the *Table of Design Variables.*




4. Repeat these steps for the remaining variables listed in the *Table of Design Variables* to associate the values from Table 9-9 on page 761 with the variable names.

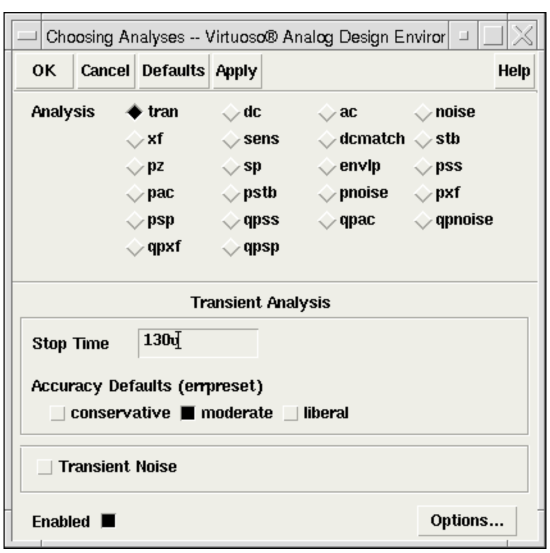
5. Click *OK* in the Editing Design Variables form after you have added all the variable values.

The table of *Design Variables* in the ADE window is updated and the Editing Design Variables form is closed.



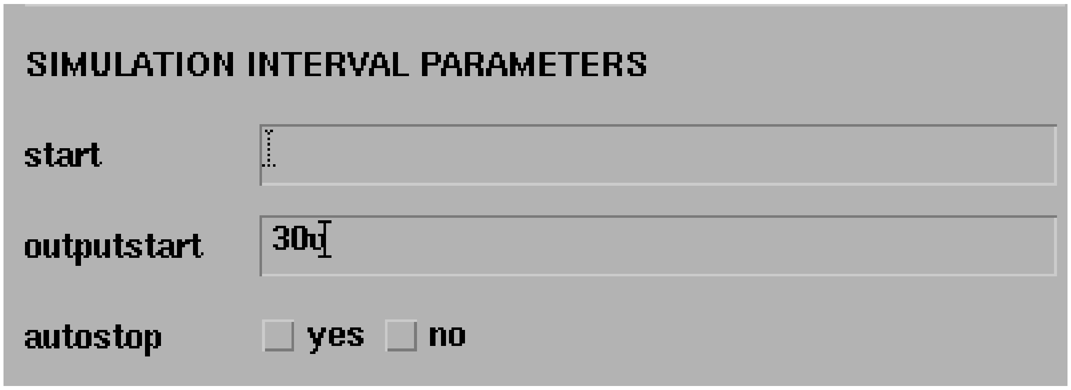

## Setting Up and Running a Transient Analysis

1. In the ADE window, choose *Analyses – Choose* to display the Choosing Analyses form.
2. In the Choosing Analyses form, if necessary, click *tran* to select a transient analysis.
3. In the Choosing Analyses form, enter `130u` in the *Stop Time* field.
4. Highlight *moderate* for *Accuracy Defaults (errpreset)*.




5. In the Choosing Analyses form, click *Options* to display the Transient Options form.

**6.** In the Transient Options form, enter `30u` in the *outputstart* field.




By delaying the output start, you remove start-up transients from the eye-diagrams and scatter plots.

**7.** Click *OK* in the Transient Options form.

**8.** Click *OK* in the Choosing Analyses form.

**9.** If you have not already done so, set up the simulator and model libraries with the following steps.

   **a.** Set the simulator options from the Simulator window as described in "Choosing Simulator Options" on page 708.

   **b.** Set up the model libraries from the Simulator window as described in "Setting Up Model Libraries" on page 710.

**10.** In the ADE window, choose *Simulation – Netlist and Run*.

Messages display in the CIW. The simulation log window opens. Watch for messages stating that the simulation has completed successfully.

Watch the CIW for messages stating the simulation is running and that it has completed successfully.

## Examining the Results: Eye Diagram, Histogram, and Scatter Plot

In this section we examine the results of the transient analysis of the receiver.

### Plotting the Eye Diagram (and Transient Response)

First plot an eye diagram.

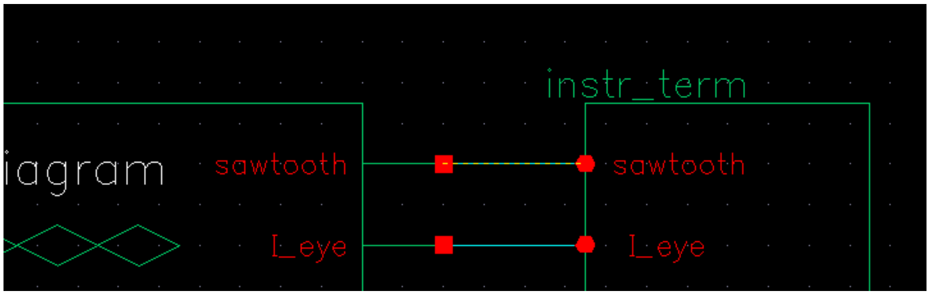
1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   The Virtuoso Schematic Editing window appears.

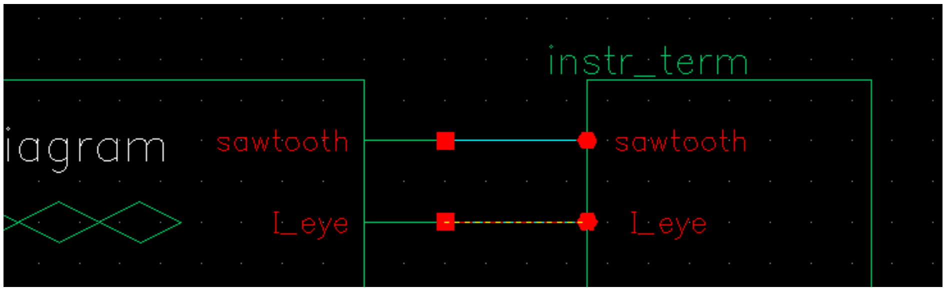
2. Following the prompts at the bottom of the window,

   *> Select nodes or terminals, press <esc> to finish selection*

   a. In the Schematic window, click the *sawtooth* net from the instrumentation (*offset_comms_instr*) block.

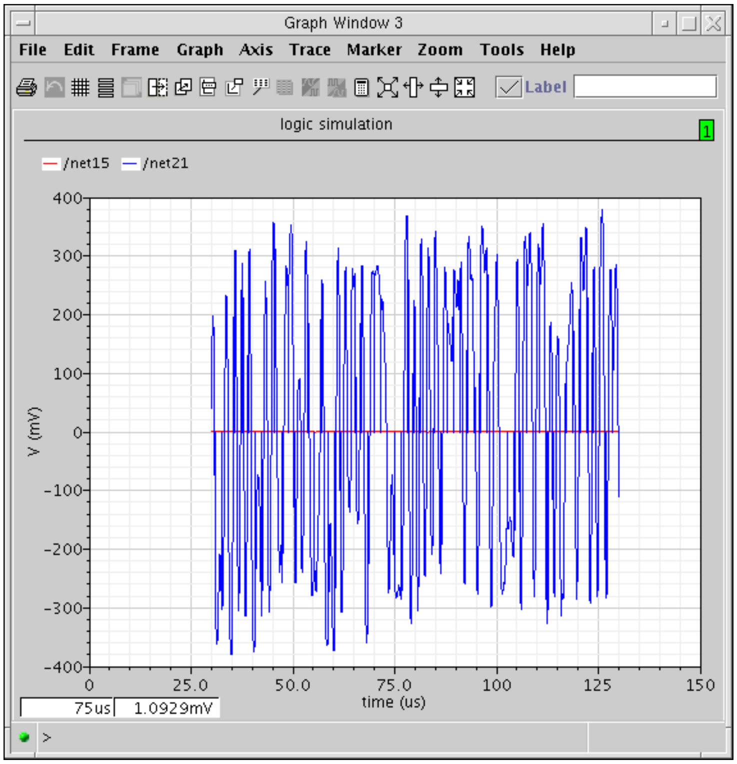



   b. Click the *I_eye* net from the instrumentation (*offset_comms_instr*) block.




   c. Press *Esc* to indicate that you have finished selecting outputs.

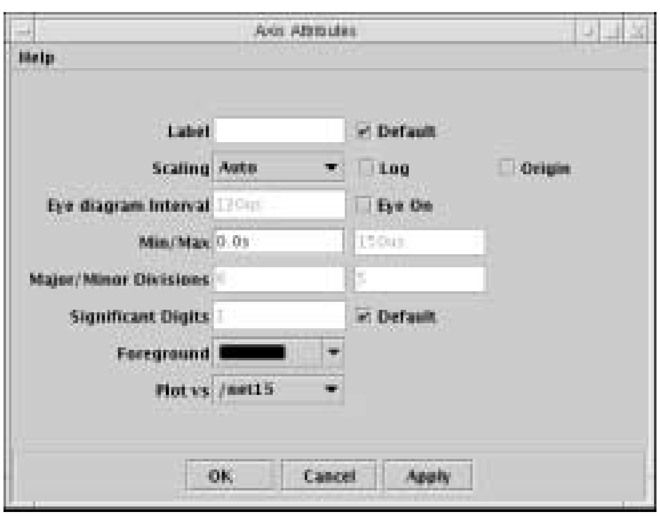
This opens a waveform window and creates a plot of the Transient Response of the net.




**3.** In the waveform window, double-click the X axis.

The Axis Attributes form displays:

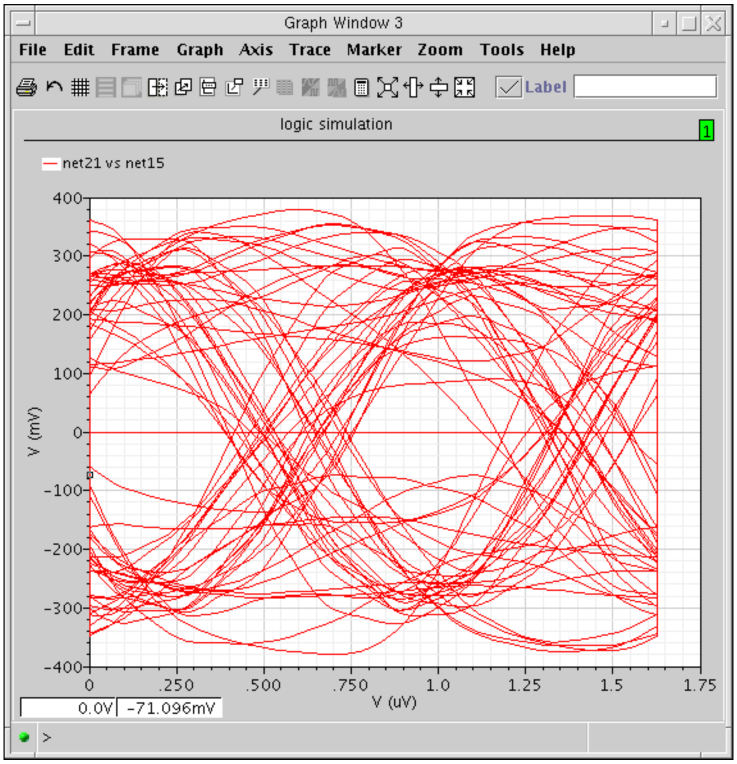
**a.** In the *Plot vs* field, select the `sawtooth` net, such as `/net15`.



**b.** Click *OK*.

You see the eye-diagram shown in Figure 9-8 on page 767.

**Figure 9-8 Eye-diagram**




The I-sampling delay parameter in the instrumentation block (*I_del*) is chosen with respect to this eye diagram. The delay is the time when the eye opens the widest.

The instrumentation block samples the input waveforms with this delay to compute all statistics and to produce scatter plots.

**4.** In the waveform window, choose *File – Close*.

The waveform window closes.

### Generate the Histogram

Now generate a histogram of the I-voltage at the sampling times.

**1.** In the ADE window, choose *Results – Direct Plot – Transient Signal*.

The Virtuoso Schematic Editing window appears.

2. Following the prompts at the bottom of the window,

   a. Click the *eye_hist* net from the instrumentation (*offset_comms_instr)* block.

   b. Click the *eye_count_hist* net from *offset_comms_instr*.

   c. Press *Esc* to indicate that you have finished selecting outputs.

      This creates a plot in the waveform window.

3. In the waveform window, double-click the X axis.

   The Axis Attributes form displays:

   a. In the *Plot vs* field, select the *eye_hist* output, such as `/net17`.

   b. Click *OK*.

   This creates an unintelligible intermediate plot in the waveform window.

4. In the waveform window, double-click the trace in the graph.

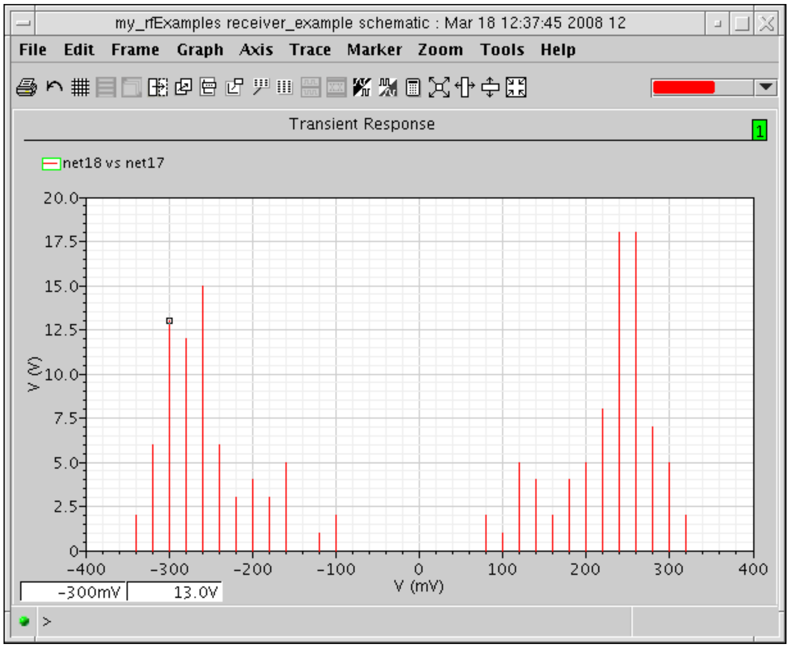
   The Trace Attributes form displays.

   a. In the *Type*/*Style* cyclic field, select *Bars*.

   b. Click *OK*.

   You see a plot like Figure 9-9 on page 769.

#### Figure 9-9 Histogram




5. In the waveform window, choose *File – Close*.

   The waveform window closes.

### Generating the Scatter Plot

Generate a scatter plot of the received symbols.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   This displays the waveform window.

2. Following the prompts at the bottom of the waveform window,

   In the Schematic window:

   a. Click the *I_scatter* output from the instrumentation (*offset_comms_instr)* block.

   b. Click the *Q_scatter* output from *offset_comms_instr*.

   c. Click *Esc* to indicate that you have finished selecting outputs.

      This creates a plot in the waveform window.

3. In the waveform window, double-click the X axis.

The Axis Attributes form displays:

**a.** In the *Plot vs* field, select the *I_scatter* output, such as `/net22`.

**b.** Click *OK*.

**4.** In the waveform window, double-click the trace in the graph.

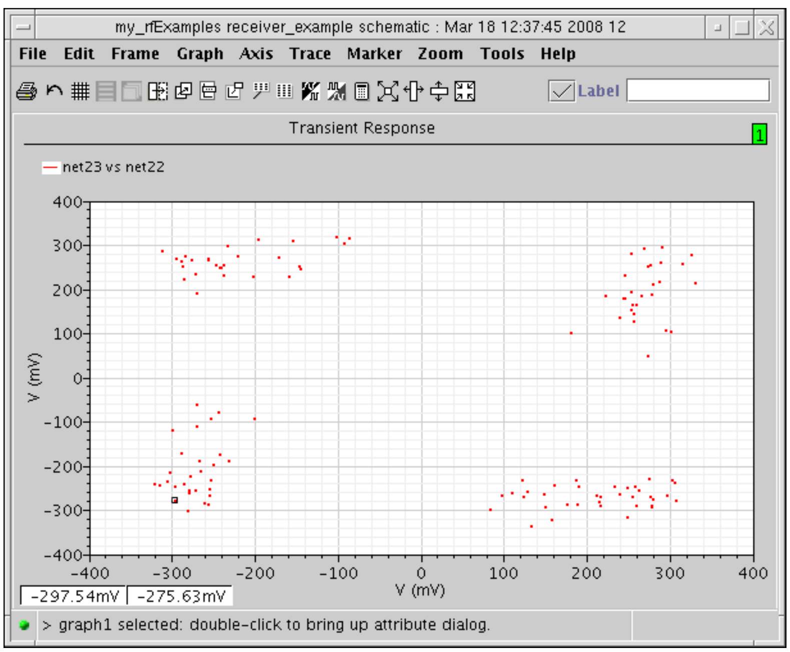
The Trace Attributes form displays.

**a.** In the *Type/Style* cyclic field, select *Points*.

**b.** Click *OK*.

You see a scatter plot like Figure 9-10 on page 770.

**Figure 9-10 Scatter Plot**




**5.** In the waveform window, choose *File – Close*.

The waveform window closes.

## The Various Instrumentation Blocks

The CDMA source (*CDMA_reverse_xmit)* produced offset QPSK symbols. Offset QPSK modulation avoids traversing the origin by staggering the digital changes in the I and Q signals. Running the baseband trajectory through the origin increases spectral regrowth in the transmitters.

The instrumentation block (*offset_comms_instr*) samples the I and Q signals at different times then plots the two staggered samples against each other. The resulting scatter plot shows the received symbols. A scatter plot of the unstaggered samples reveals only what is happening in one dimension, either the I or Q dimension.

For non-offset QPSK and QAM modulation schemes, use the *comms_instr* instrumentation block instead of the *offset_comms_instr* block.

### Measuring RMS EVM

You can use the same instrumentation block (*offset_comms_instr*) to compute root-mean-squared error vector magnitude (RMS EVM). The error vector is the vectorial difference between the ideal received symbol and the actual received symbol.

- EVM (error vector magnitude) is the magnitude of the error vector.
- RMS EVM is the root-mean-squared value of a sequence of EVMs.

RMS EVM is one measure of a receiver's quality. RMS EVM can account for as much or as little distortion and noise as you like. The trick is to figure out where the ideal received symbol lies. You can do this using the *I_ref* and *Q_ref* inputs to the *offset_comms_instr* instrumentation block.

To calculate RMS EVM, you

- Create a duplicate copy of the receiver chain from the *BB_driver* to the *IQ_demod_BB* including these two blocks
- Place the duplicate copy below the original receiver chain in the Schematic window
- Modify the duplicate receiver chain to make it as *ideal* as you like by changing parameter values for the individual function blocks.

  For example, to see the effect of just the LNA's IP3 value on RMS EVM, in the duplicate receiver chain make the LNA's IP3 absurdly large.

### Constructing the Ideal Receiver Chain

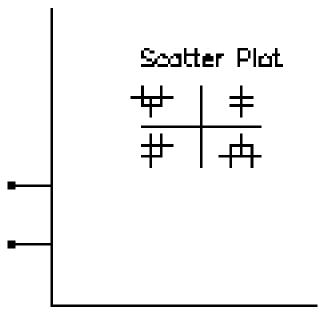
The *ideal* receiver chain (the duplicated and modified receiver chain) is driven from the same input, the *CDMA_reverse_xmit* block, as the original receiver chain. The output of the ideal receiver chain drives the instrumentation block's *I_ref* and *Q_ref* inputs.

In the ideal receiver chain, you copy the first receiver chain and make every block ideal.

1. Remove the *gnd* from the *I_ref* and *Q_ref* pins on the instrumentation block.

   a. In the Schematic window, choose *Edit - Delete*.

   b. In the Schematic window, click each wire and the *Gnd* symbol attached to the *I_ref* and *Q_ref* pins.




   c. Click *esc* to stop deleting.

2. Duplicate the receiver chain from the *BB_driver* to the *IQ_demod_BB* inclusive. Do not duplicate the filters. Follow the prompts at the bottom of the Schematic window.

   a. In the Schematic window, choose *Edit – Copy*.

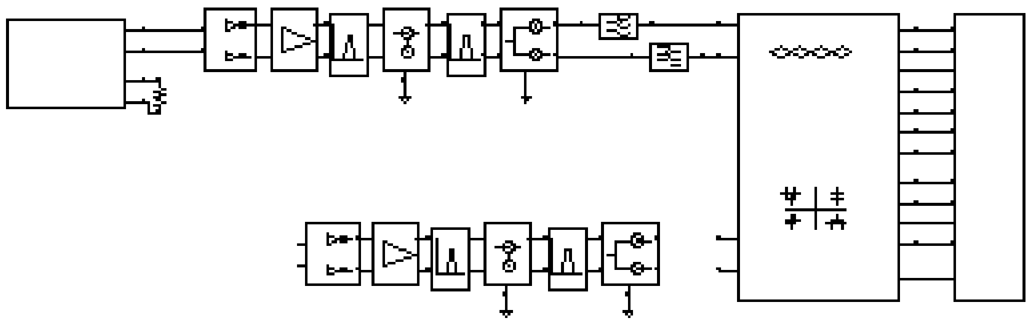
   b. In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *IQ_demod_BB* block. Click again to complete the box.

   The blocks within the box are highlighted.

   c. Click within the highlighted area.

   A copy of the highlighted blocks in the receiver chain now moves with the cursor.

**d.** Place the duplicate receiver chain so that the output pins on the *IQ_demod_BB* are in line with the *I_ref* and *Q_ref* pins on the instrumentation block.



**3.** Wire the duplicate receiver chain to the CDMA signal source (*CDMA_reverse_xmit)* and the instrumentation block (*offset_comms_instr*).

**a.** In the Schematic window, choose *Add - Wire (narrow).*

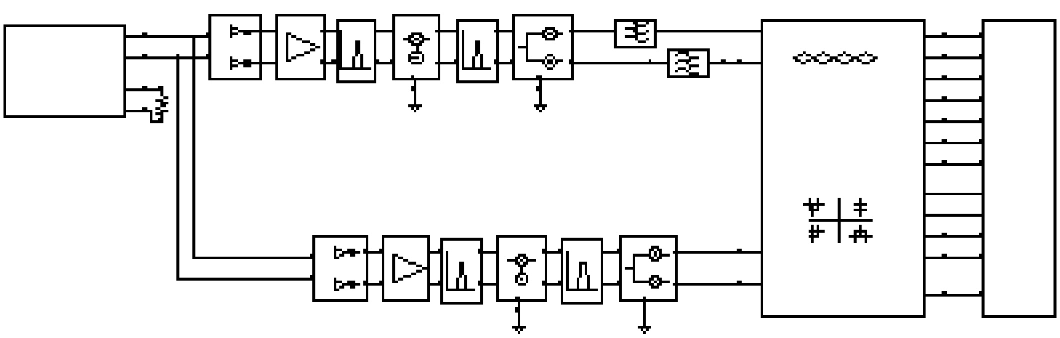
**b.** Connect the *IQ_demod_BB* outputs on the duplicate receiver to the *I_ref* and *Q_ref* pins on the instrumentation block.

**c.** Connect the output pins on *CDMA_reverse_xmit* to the input pins of the duplicate *BB_driver*. This drives the duplicate receiver from the CDMA signal source.

**d.** Click *esc* to stop wiring.

The schematic with the duplicate receiver chain wired up looks like Figure 9-11.

**Figure 9-11 Receiver Model with Duplicated Receiver Chain**



### Modifying Parameter Values to Make the Blocks Ideal

Now modify the parameter values for each block in the duplicate receiver chain to create ideal blocks. Block names, parameter names, and parameter values are given in Table 9-10.

**Table 9-10 Parameter Values to Create an Ideal Receiver**

| Block Names | Parameter Names | New Parameter Values |
|---|---|---|
| `LNA_BB` | *Input referred IP3 [dBm]*<br>*{1,0,-1} for {cw, none, ccw}* | `100`<br>`0` |
| `BB_butterworth_bp` | *Cell Name* | `BB_loss` |
| `dwn_cnvrt` | *Input referred IP3 [dBm]* | `100` |
| `BB_butterworth_bp` | *Cell Name* | `BB_loss` |
| `IQ_demod_BB` | *I-[dBm] Input referred IP3*<br>*Q-[dBm] Input referred IP3* | `100`<br>`100` |

1. In the Schematic window, choose *Edit – Properties – Objects* to open the Edit Object Properties form.
2. In the Schematic window, select the *LNA_BB* block.

   The Edit Object Properties form changes to display properties for the *LNA_BB* block.

   a. Set *Input referred IP3 [dBm]* to 100.

   b. Set {*1, 0,-1*} *for* {*cw, none, ccw*} to 0. (This eliminates AM/PM conversion.)

   c. Click *Apply*.
3. In the Schematic window, select the first RF *BB_butterworth_bp* block.

   The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block.

   a. Change the *Cell Name* to *BB_loss*.

   b. Click *Apply*.

   The properties and symbol change to those for the *BB_loss* block. The sole purpose of the *BB_loss* model is to replace a filter in an RMS EVM analysis.

The *Reference impedance* for the *BB_loss* block should equal the *Output impedance* of the *BB_butterworth_bp* bandpass filter block it replaces. The value should be 50 ohms for both blocks and you should not have to change it.

The *BB_loss* model retains the filter's loss but eliminates the filter's dynamics so you can see what, if any, affect the filter has on EVM through inter-symbol interference. To eliminate the loss as well as the dynamics, you might even replace the filter with straight wires. This example uses the *BB_loss* block instead.

**4.** In the Schematic window, select the *dwn_cnvrt* block.

The Edit Object Properties form changes to display properties for the *dwn_cnvrt* block

**a.** Set *Input referred IP3 [dBm]* to 100.

**b.** Click *Apply*.

**5.** In the Schematic window, select the second *BB_butterworth_bp* block.

The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block

**a.** Change the *Cell Name* to *BB_loss*.

**b.** Click *Apply*.

**6.** In the Schematic window, select the *IQ_demod_BB* block.

The Edit Object Properties form changes to display properties for the *IQ_demod_BB* block

**a.** Set *I-[dBm] input referred IP3* to 100.

**b.** Set *Q-[dBm] input referred IP3* to 100.

**c.** Click *Apply*.

**7.** In the Edit Object Properties form, click *OK* to close the form.

**8.** In the Schematic window, choose *Design – Check and Save* to check and save your modifications to the circuit.

### Set Up and Run a Transient Analysis

Set up and run a transient analysis as described in "Setting Up and Running a Transient Analysis" on page 763. Set the *Stop Time* to `130u` and the *outputstart* option to `30u`. Click *OK* in both the Transient Options and Choosing Analyses forms. Choose *Simulation – Netlist and Run* to run the transient analysis.

Look for messages in the CIW stating that the simulation is starting. Watch the simulation log window for messages that the simulation has completed successfully.

### Plot the RMS EVM Output

After the simulation, plot the RMS_EVM output of the instrumentation block.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   This displays the waveform window.

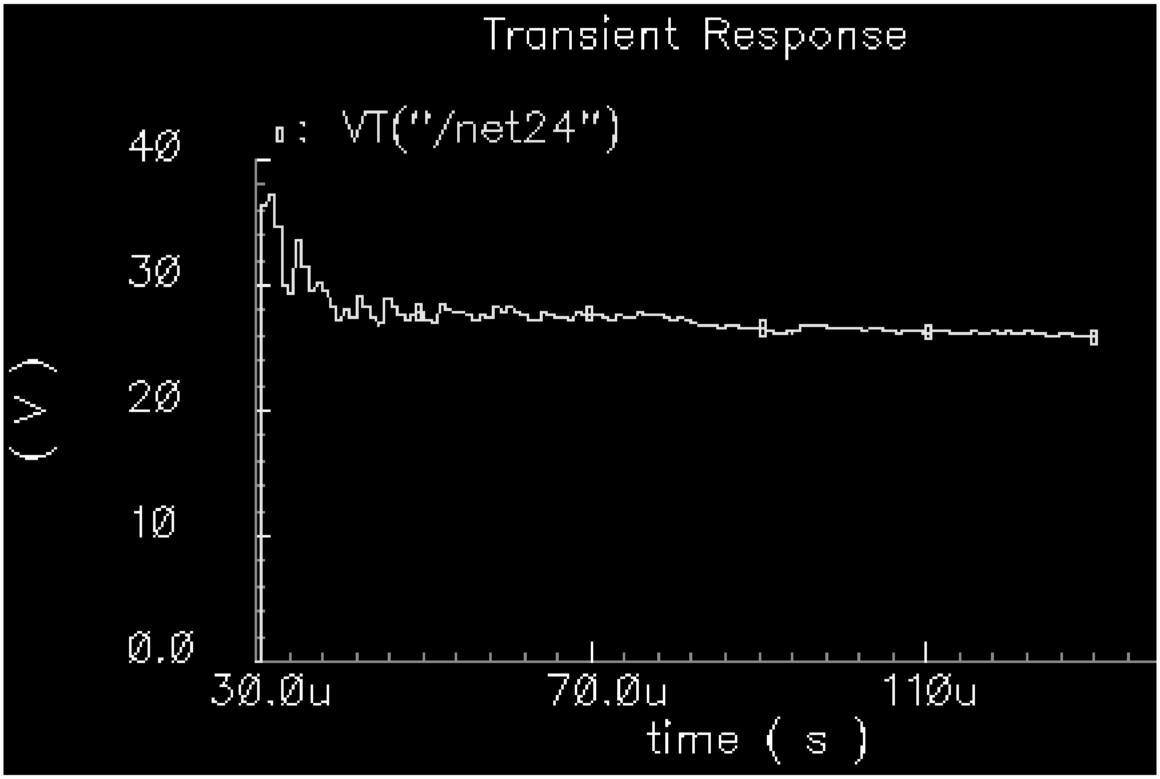
2. Following the prompts at the bottom of the waveform window,

   a. In the schematic, click the *rms_EVM* output net from the instrumentation (*offset_comms_instr)* block.

   b. Click *Esc* to indicate that you have finished selecting outputs.

      This creates the RMS EVM plot in the waveform window as shown in Figure 9-12 on page 776.

**Figure 9-12  RMS EVM**



The RMS EVM trace starts at 30us, which is the `statistics start time` parameter of the instrumentation block. The `statistics start time` parameter keeps start-up transients out of the statistics.

The trace settles out at 25.84 Volts. This means that after 130us of data is collected, and ignoring the first 30us, the RMS EVM is 25.84%. The EVM measurement is normalized to the RMS magnitude of the ideal symbol then multiplied by 100 to express the measurement as a percentage.

**3.** In the waveform window, choose *File – Close*.

## Computing Minimized RMS Noise Using the Optimizer

There is one more construction step before proceeding to the Circuit Optimizer application. You set up the Circuit Optimizer to minimize RMS noise subject to performance constraints. This step replicates the receiver chain yet one more time to generate the noise measurement.

**1.** Duplicate the original receiver chain from the *BB_driver* up to and including both low pass filters (*butterworth_lp*). Follow the prompts at the bottom of the Schematic window.

**a.** In the Schematic window, choose *Edit – Copy*.

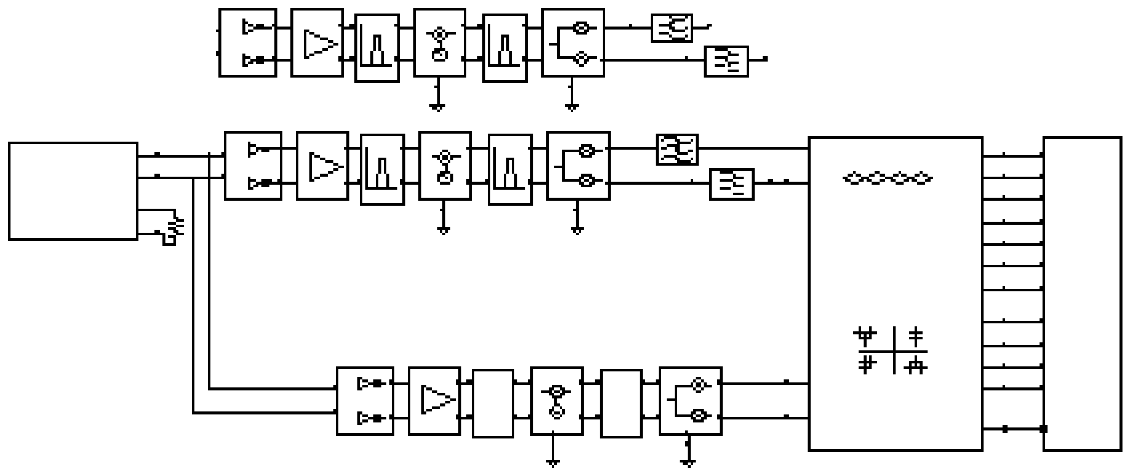
**b.** In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *butterworth_lp* filter blocks. Click again to complete the box.

The blocks within the box are highlighted.

**c.** Click within the highlighted area.

A copy of the highlighted blocks in the receiver chain now moves with the cursor.

**d.** Place the duplicate receiver chain above the original receiver chain.



**2.** In the duplicate receiver chain, ground the *Q_in* pin on the *BB_driver* block.

**a.** In the Schematic window, choose *Add – Wire.*

**b.** Click the *Q_in* pin and run the wire to the *Gnd* symbol below the *dwn_cnvrt* block.

**c.** Click the *Gnd* symbol below the *dwn_cnvrt* block.

**d.** Click *esc* to stop wiring.

**3.** Add a 50mV DC voltage source to the left of the *BB_driver* block to drive the *I_in* pin on the *BB_driver* block. At the same time add a *gnd* symbol below the *port* in the schematic.

**a.** In the Schematic window, choose *Add – Instance* to display the Add Instance form.

**b.** In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

**c.** In the Library Browser - Add Instance form, click *analogLib*.

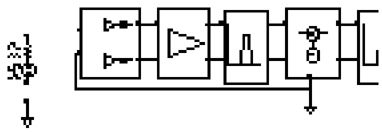
**d.** Scroll the elements in the *Cell* column and click *port*.

**e.** The outline for the *port* symbol is attached to the cursor. Move the *port* symbol to the left of the *BB_driver* block and click to place the *port* symbol.

**f.** Return to the Library Browser - Add Instance form and scroll the elements in the *Cell* column and click *gnd*.

**g.** The outline for the *gnd* symbol is attached to the cursor. Move the *gnd* symbol below the *port* symbol and click to place it there.

**h.** Click *Esc* to remove the *gnd* symbol from the cursor.



**4.** In the Schematic window, choose *Edit – Object – Properties* to modify the *port* symbol using the Edit Object Properties form.

**a.** In the Schematic window, click the *port* symbol.

The Edit Object Properties form changes to display information for the *port* symbol.

**b.** In the *Source Type* cyclic field, select *dc*.

**c.** In the *DC Voltage* field, enter 50m.

**d.** Highlight *Display small signal params* to display small signal parameters.

**e.** In the *AC Magnitude* field type `1V`.

**f.** In the *AC Phase* field type `0`.

**g.** Click *OK* in the Edit Object Properties form.

**5.** Load the low pass filters with a *res_BB* model from the *top_dwnBB* category of *rfLib*. Use the default parameters and ground the output pins.

**a.** In the Schematic window, choose *Add – Instance* to display the Add Instance form.

**b.** In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

**c.** In the Library Browser - Add Instance form, click *rfLib*.

**d.** Scroll the elements in the *Cell* column and click *res_BB*.

The outline for the *res_BB* symbol is attached to the cursor.

**e.** Move the *res_BB* symbol to the right of the two low pass filters (*butterworth_lp*) and click to place the *res_BB* symbol.

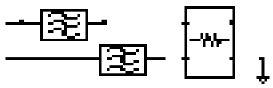
**f.** Return to the Library Browser - Add Instance form and click *analogLib.*

**g.** Scroll the elements in the *Cell* column and click *gnd*.

The outline for the *gnd* symbol is attached to the cursor.

h. Move the *gnd* symbol to the right of the *res_BB* symbol and click to place it there.

i. Click *Esc* to remove the symbol from the cursor.



6. In the duplicate receiver chain, wire the *port*, the *res_BB* block, and their *gnd* blocks.

   a. In the Schematic window, choose *Add – Wire.*

   b. Click the *I_in* pin on the *BB_driver*, then click the top pin on the *port*.

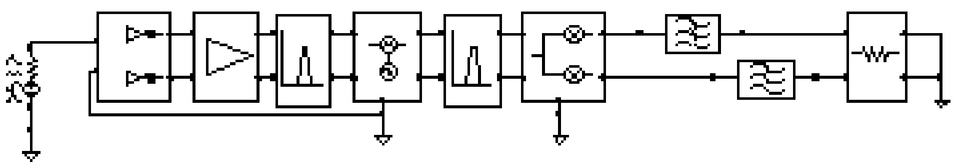
   c. Click the bottom pin on the *port*, then click the *gnd* pin just below it.

   d. Click the *out* pin on the top *butterworth_lp* filter, then click the *I_in* pin on the *res_BB* block.

   e. Click the *out* pin on the lower *butterworth_lp* filter, then click the Q_*in* pin on the *res_BB* block.

   f. Click the *I_out* pin on the *res_BB* block. Then click the top pin on the *gnd* located to it's right.

   g. Click the Q_*out* pin on the *res_BB* block. Then click the top pin on the same *gnd*.

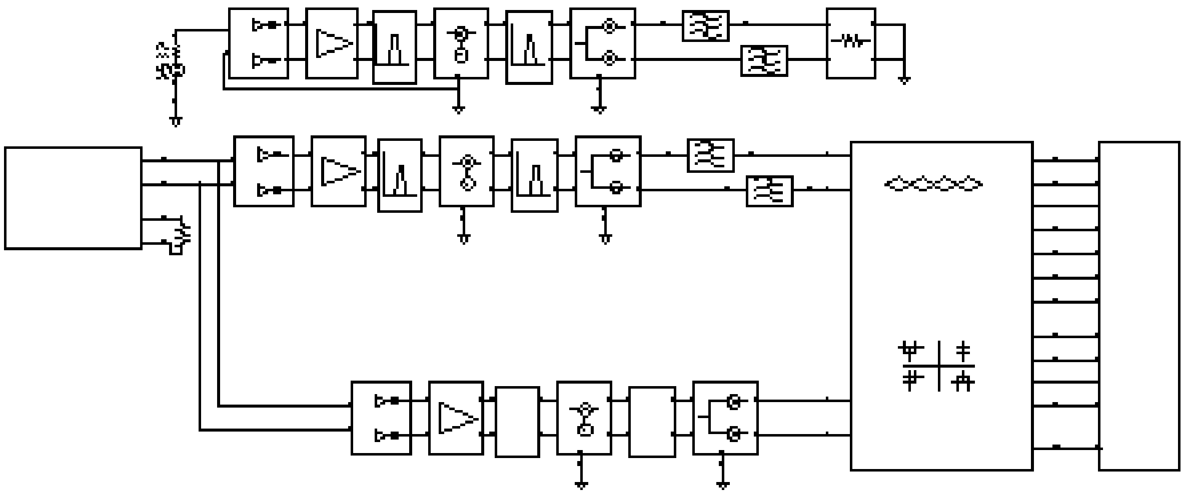


   h. Click *esc* to stop wiring.

7. In the Schematic window, choose *Design – Check and Save* to check and save the schematic.

The schematic with the third receiver chain is shown in Figure 9-13.

**Figure 9-13 Schematic with Noise Generating Receiver**



### Set Up and Run Transient and Noise Analyses

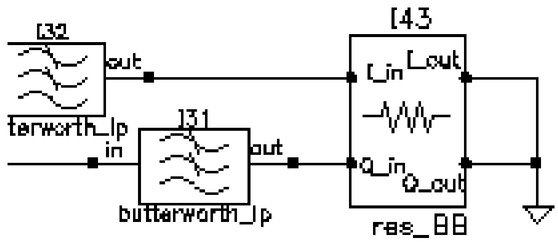
Set up a transient analysis as described in "Setting Up and Running a Transient Analysis" on page 763. Set the *Stop Time* to `130u` and the *outputstart* option to `30u` and make sure that the transient analysis is enabled.

Set up a noise analysis as follows:

1. In the Choosing Analyses form, click *noise* to select a noise analysis.
2. For *Sweep Variable*, click *Frequency*.
3. For *Sweep Range*, click *Start-Stop*.
4. Set up the analysis to sweep frequency from 0 to100 MHz.
   a. For the starting frequency, in the *Start* field enter `0`.
   b. For the stop frequency, in the *Stop* field enter `100M`.
5. Set up the *Output Noise* source.

**a.** In the *Output Noise* cyclic field, select *voltage.*

**b.** To select the *Positive Output Node*, click *Select* next to the *Positive Output Node* field. Then, in the Schematic window, click the net next to the *I_in* pin on the *res_BB* block.

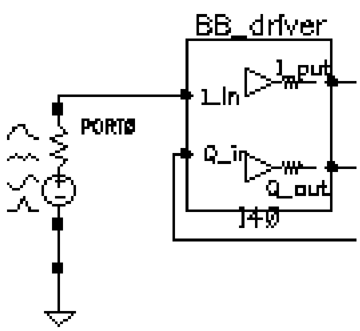



**c.** To select the *Negative Output Node*, click *Select* next to the *Negative Output Node* field. Then, in the Schematic window, click the net next to the *I_out* pin on the *res_BB* block.

**6.** Set up the *Input Noise* source.

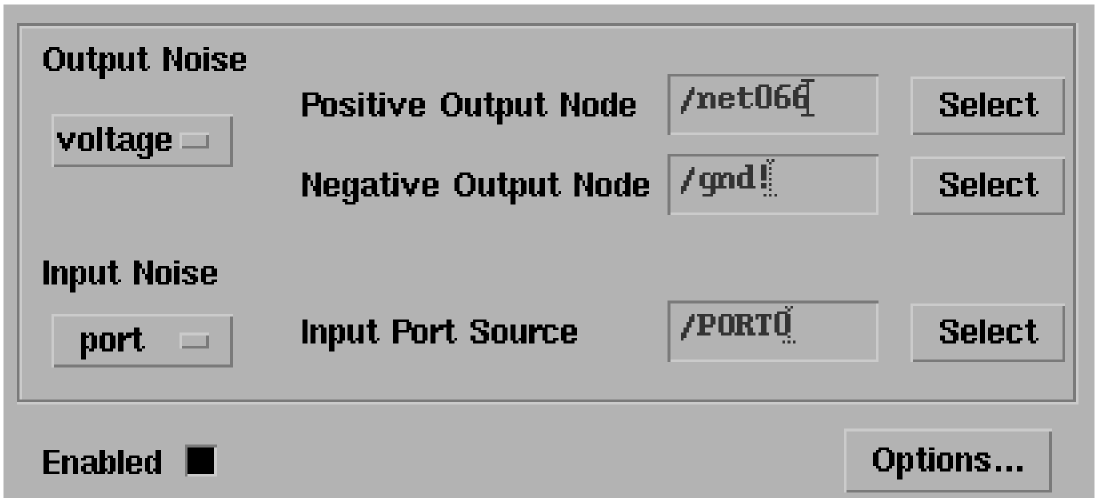
**a.** In the *Input Noise* cyclic field, select *port.*

**b.** To select the *Input Port Source*, click *Select* next to the *Input Port Source* field. Then, in the Schematic window, click the *DC input port* model.



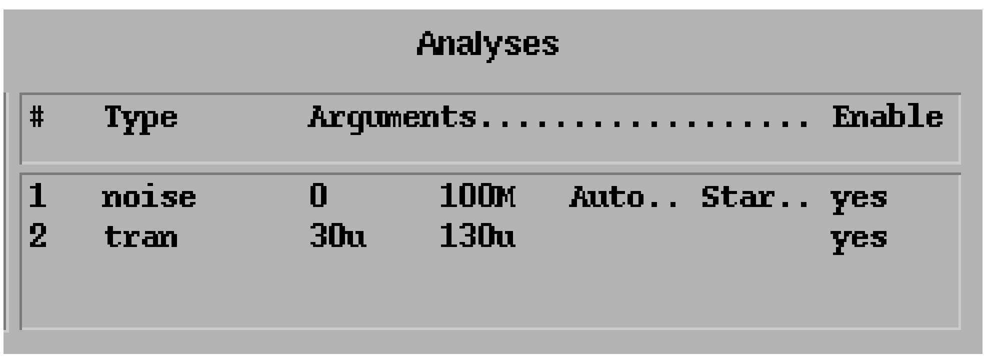

**7.** Click *Enabled*.

The *Output Noise* and *Input Noise* sections of the Noise analysis form look as follows. (The net instance numbers might be different, however.)




8. Verify that *Enabled* is highlighted and click *OK* in the Choosing Analyses form.

9. In the ADE window, check the Analysis area to verify that both the transient and noise analyses are set up properly and that they are both enabled.

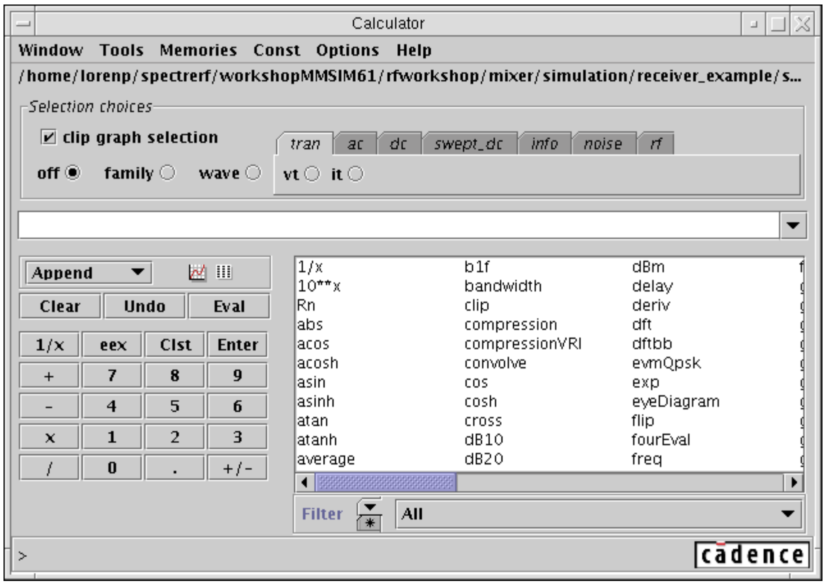



10. In the ADE window, choose *Simulation – Netlist and Run* to start the simulations.

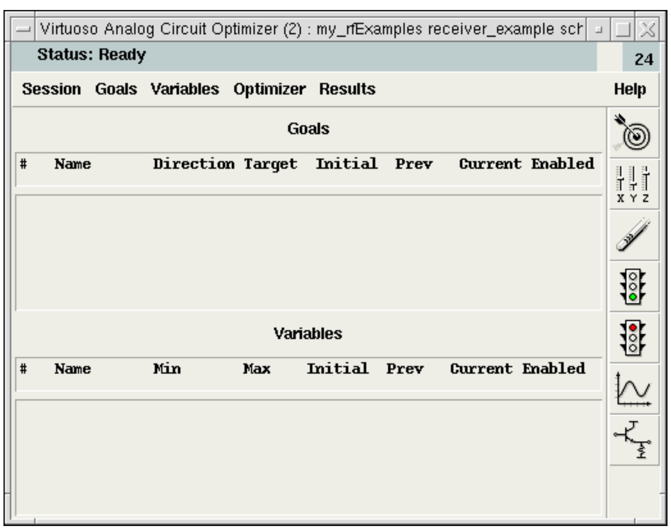
    Watch the messages in the CIW to verify that everything is set up properly and that the simulations start. Check the simulation log window to see that the simulations run and complete properly.

### Set Up to Run the Circuit Optimizer

1. In the ADE window, choose *Tools – Calculator* to open the calculator window.




2. In the ADE window, choose *Tools – Optimization* to open the Circuit Optimizer window.




### *Add the First Goal*

1. In the calculator window's function panel, select *rmsNoise*.
2. When the RMS Noise panel opens,
   a. In the *From* field, enter `0`.

b. In the *To* field, enter `100M`.

c. Click *OK*.

The expression

`rmsNoise(0,100M)`

displays in the buffer of the calculator window.

3. In the Circuit Optimizer window, choose *Goals – Add* to add the first goal.

   The Adding Goals form opens.

4. Fill in the Adding Goals form.

   a. In the *Name* field, type `rmsNoise`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

      The expression `rmsNoise(0 100000000)` displays in the *Expression* field.

   c. Ensure that the *Direction* is minimize.

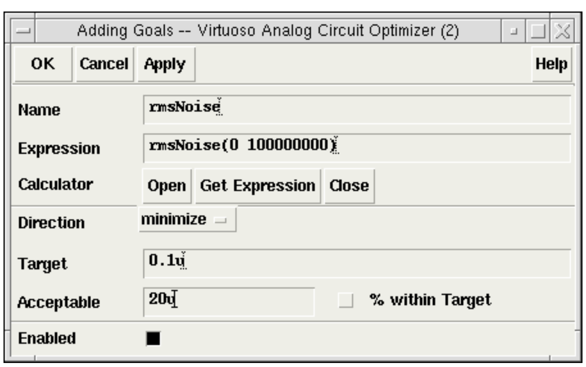
      The default for the Optimizer is to minimize goals.

   d. In the *Target* field, enter `0.1u`.

   e. In the *Acceptable* field, enter `20u`.

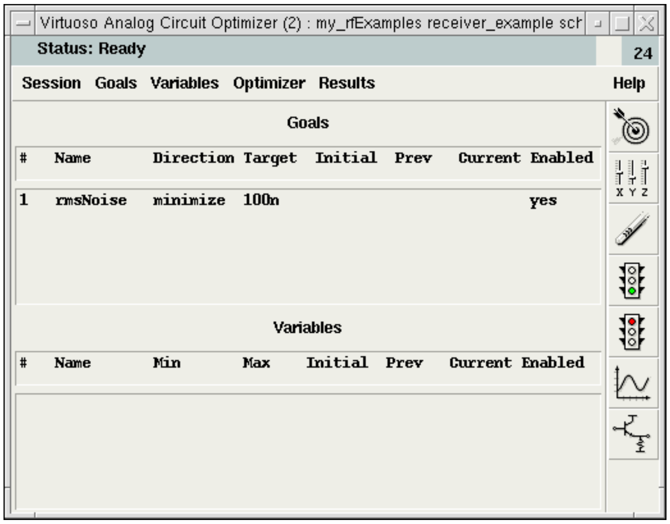
   f. Verify that the form is enabled.

      The Adding Goals form looks like this.




   g. Click *OK*.

The first goal is added to the Circuit Optimizer window.




### *Add the Second Goal*

1. In the calculator window, under the *tran* tab, click the *vt* button.

   The Schematic Editing window becomes active.

2. In the Schematic window, click the *rms_EVM* output net of the instrumentation block.

   An expression similar to `VT("/net24")` displays in the calculator buffer.

3. In the calculator window's function panel, select *value*.

   When the Value panel appears

   a. Enter `130u` in the *Interpolate At* field.

   b. Click *OK*.

      The expanded expression `value(VT("/net24") 130u )` displays in the buffer of the calculator window.

4. In the Circuit Optimizer, choose *Goals – Add*.

   When the Adding Goals form opens

   a. In the *Name* field, enter `evm`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

      An expression similar to

`value(VT("/net24") 0.00013)`

displays in the *Expression* field.

**c.** In the *Direction* cyclic field, select <=

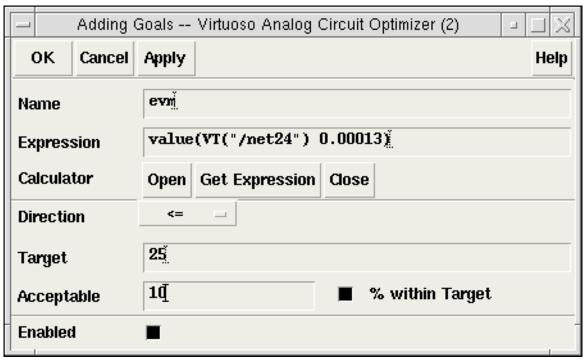
**d.** In the *Target* field, enter `25`.

**e.** In the *Acceptable* field, enter `10`.

**f.** Click the *% within Target* button.

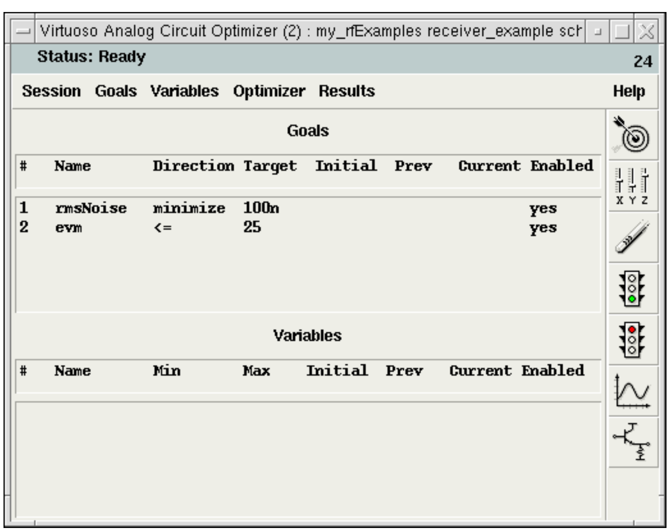
**g.** Verify that *Enabled* is active.

The Adding Goals form looks like this.




**h.** Click *OK*.

The second goal is added to the Circuit Optimizer window.



### *Add the Third Goal*

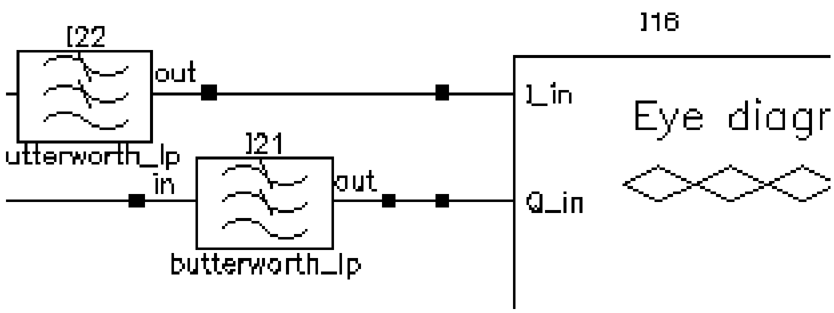
1. In the calculator window, click *Clear*.

   The buffer is cleared.

2. Under the *tran* tab, click the *vt* button.

   The Schematic Editing window becomes active.

3. In the Schematic window, click the *I_in* pin of the instrumentation block.




   An expression similar to `VT("/net14")` displays in the buffer of the calculator window.

4. In the calculator window's function panel, select *rms*.

   An expanded expression similar to

   `rms(VT("/net14"))`

   displays in the buffer of the calculator window.

   The objective is to keep the *rms* value of this signal level above 300 mV. Note that all goals must be scalars.

5. In the Circuit Optimizer, choose *Goals - Add*.

   The Adding Goals form opens.

6. Fill in the fields of the Adding Goals form.

   a. In the *Name* field, enter `sig_level`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

      And expression similar to

      `rms(VT("/net14"))`

      displays in the *Expression* field.

**c.** In the *Direction* cyclic field, select >=

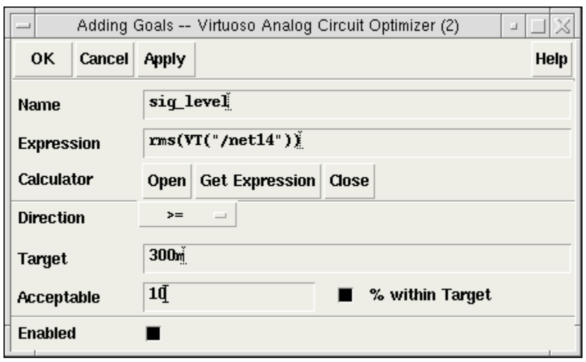
**d.** In the *Target* field, enter `300m`.

**e.** In the *Acceptable* field, enter `10`.

**f.** Click the *% within Target* button.

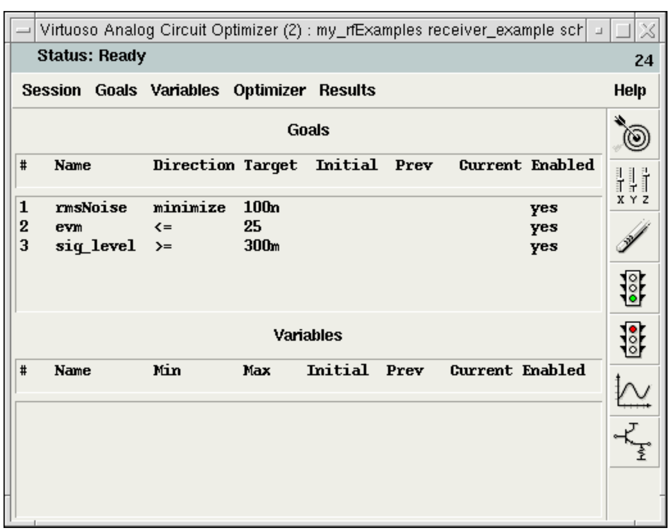
**g.** Verify that the *Enabled* button is active.

The Adding Goals form looks like this.




**h.** Click *OK*.

The third goal is added to the Circuit Optimizer window.



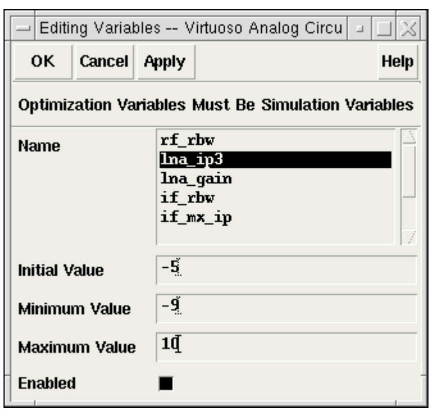

### *Add the Circuit Variables to the Optimizer*

Add the variables to the Circuit Optimizer window.

1. In the Circuit Optimizer window, choose *Variables – Add/Edit*.
2. When the Editing Variables form opens
   a. In the *Name* list box, click the *lna_ip3* variable.

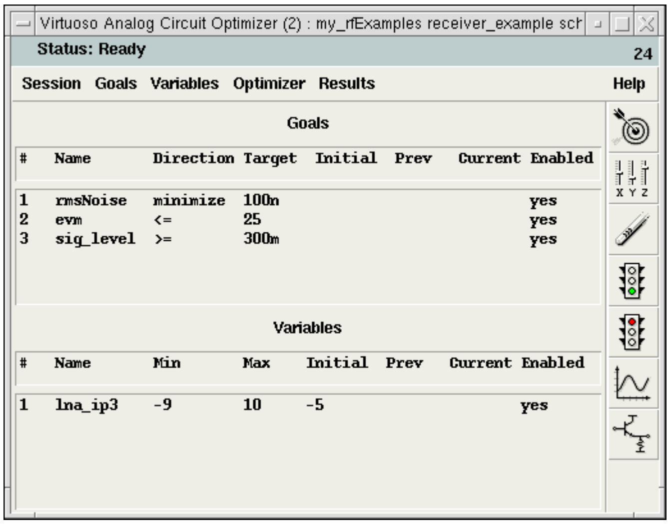
      The *lna_ip3* variable is highlighted in the list box and it's current value `-5` displays in the *Initial Value* field.
   b. In the *Minimum Value* field, enter `-9`.
   c. In the *Maximum Value* field, enter `10`.
   d. If necessary, click *Enabled*.

      The Editing Variables form appears as follows.




   e. Click *Apply*.

Information for the *lna_ip3* variable displays in the *Variables* section of the Circuit Optimizer.




3. Repeat this procedure to add all the variables and values listed in Table 9-11

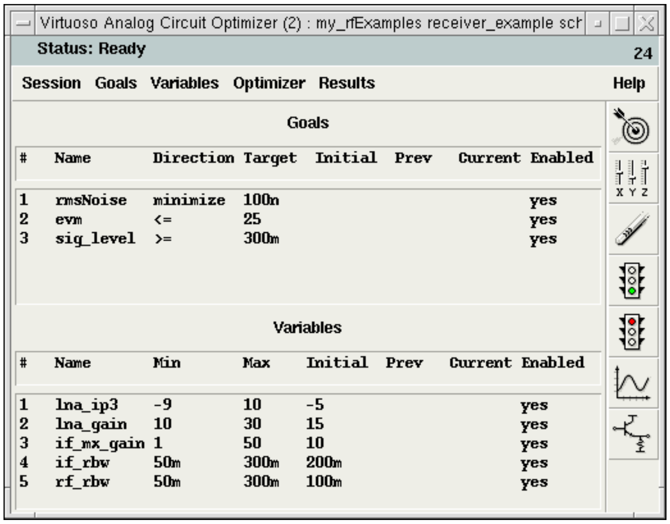
**Table 9-11 Variables and Values for the Optimizer**

| Variable Name | Initial Value | Minimum Value | Maximum Value |
|---|---|---|---|
| `lna_ip3` | -5 | -9 | 10 |
| `lna_gain` | 15 | 10 | 30 |
| `if_mx_gain` | 10 | 1 | 50 |
| `if_rbw` | 200m | 50m | 300m |
| `rf_rbw` | 100m | 50m | 300m |

4. Click *OK* to close the Editing Variables form.

   In the Circuit Optimizer window, the variables and optimization goals appear as shown in Figure 9-14 on page 792.

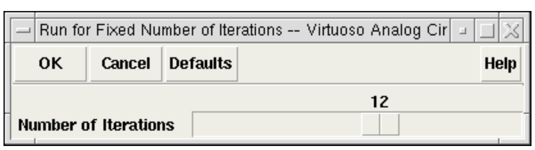
### Figure 9-14 Circuit Optimizer Setup




### Run the Circuit Optimizer

**Note:** This Circuit Optimizer analysis might take several hours to complete.

1. In the Circuit Optimizer, choose *Optimizer – Run n* to display the *Run for Fixed Number of Iterations* form.
2. In the *Number of Iterations* field, move the slider to the right until *12* displays.




3. Click *OK.*

   The *Run for Fixed Number of Iterations* form closes and the Circuit Optimizer starts running.

   Watch for simulator startup messages in the CIW. Monitor the progress of the analysis in the log window.

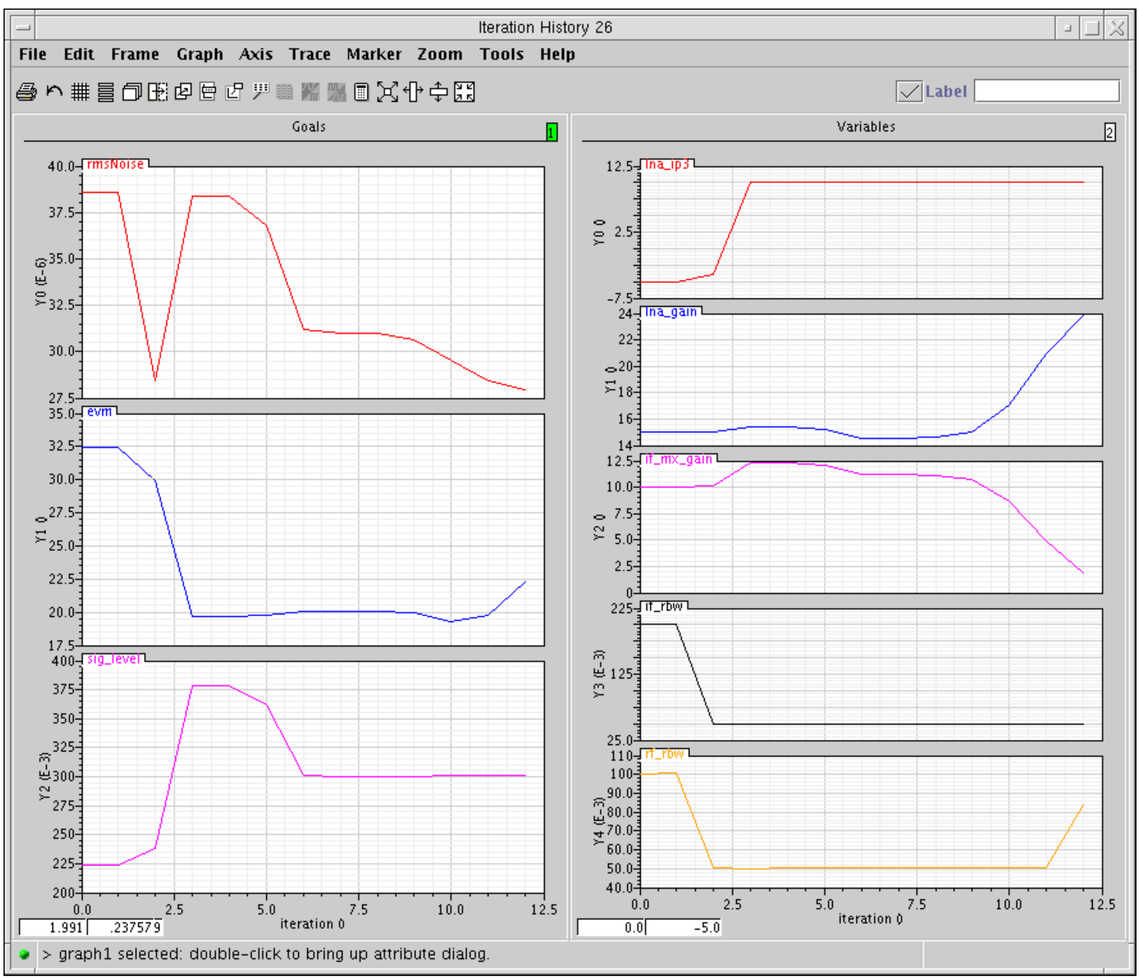
   The waveform window opens when the first simulation completes. As simulations complete, the results are added to the plots in the waveform window.

### Viewing the Circuit Optimizer Output

The Circuit Optimizer results displayed in the waveform window are shown in Figure 9-15 on page 793.

- The traces on the left show the EVM and RMS output signal level, and the RMS noise at each Circuit Optimizer iteration.
- The traces on the right show the variable function block specifications at each Circuit Optimizer iteration.

### Figure 9-15  Circuit Optimizer Results After 12 Iterations




Although the example is contrived, it illustrates the use model. After the Circuit Optimizer met the constraints it tried to minimize RMS noise.

1. Save the initial state of the Analog Design Environment in case you want to start over.

2. Then in the Circuit Optimizer window, click *Results – Update Design*. The last click updates the variables in the Analog Design Environment window with the last set of variables found by the Circuit Optimizer. You use these states in the passband view.

## Summarizing the Design Procedure

To summarize, the semi-automated design procedure consists of

1. Setting up the measurements
2. Placing tolerances on the block parameters
3. Constraining the system performance
4. Identifying a quantity to minimize (or maximize)
5. Running the Optimizer
6. Evaluating the results

This is why the process is called *semi*-automated. After evaluating results for the first or second time you probably need to

1. Refine tolerances
2. Refine goals
3. Add or delete constraints
4. Add or delete variables

Each simulation covers 100s, or about 80 CDMA symbols. The suppressed carriers are an RF carrier at 2.14 GHz, an LO carrier at 2.354GHz, and an IF carrier at 214 MHz. The symbol rate is 1.2288 Mega-symbols per second.

## Creating a Passband View of the Architectural Model

After you have designed an architecture, you can quickly create a passband view of the architectural model. (Currently, the passband behavioral models in the *top_dwnPB* category and in the passband view do not introduce any specifications that are not in the baseband models.)

The passband view checks for problems that might have escaped detection in the baseband view. For example, although the baseband view quickly assesses what filters do to the baseband signal, baseband models do not indicate whether the filters are indeed removing undesired carrier harmonics.

Baseband modeling is also not the best way to evaluate image rejection. Although the baseband model accurately simulates how the desired signal propagates through an image rejection receiver, it does not accurately simulate how much of the image signal propagates to the receiver output.

The passband view also creates a system testbench as mentioned in “Top-Down Design of RF Systems” on page 700.

### Procedures for Creating the Passband Model of the Receiver

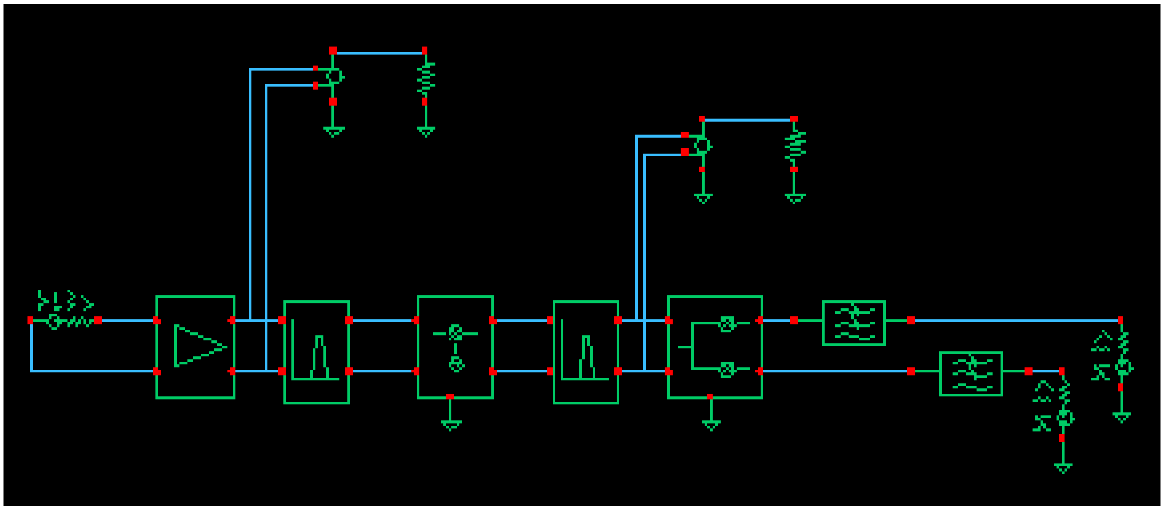
The procedures described in this section illustrate how to

- Switch from a baseband to a passband view
- Make an end-to-end RF measurement
- Measure the one dB compression point

  The one dB compression point is usually a transmitter specification but it is used to demonstrate this flow because it is easier to set up.

1. Copy the original receiver model from the Circuit Optimizer analysis to a new schematic window. Copy everything from the LNA to the low pass filters.
2. Edit the properties of the IQ_demodulator (`IQ_demod_BB`) to set the last parameter, *flo*, to *-frf+flo1*. The baseband view does not need the local oscillator frequency but the passband view does.
3. Load the low pass filters with ports.
4. Connect a port across the LNA_BB inputs. Set the Frequency name to `fin`, the frequency to `frf`, and the amplitude to `power`. (Do not abbreviate `power` to `pwr` because `pwr` is a reserved variable and you do not get any warning. Spectre RF may complain about a mysterious indexed undefined variable that increments from run to run.)
5. Add loaded voltage-controlled-voltage sources as shown in Figure 9-16 on page 796 to observe intermediate differential voltages.

**Figure 9-16 Passband View**



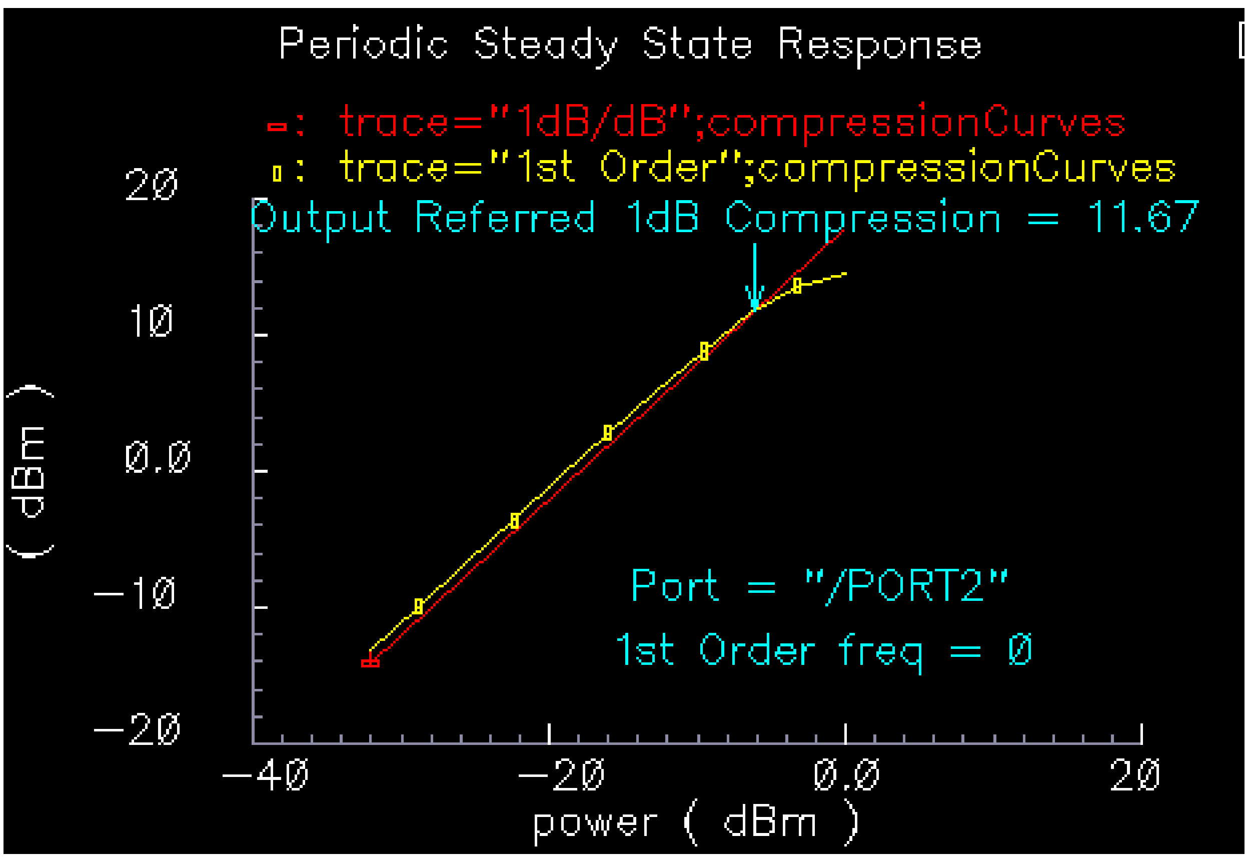
6. Check and save the schematic
7. Close the schematic window.
8. Bring up a Library Manager and select but do not open the schematic you just created.
9. In the Library Manager window click *File – New – Cell View*.
10. In the window that comes up, select Hierarchy Editor for the Tool. Click *OK*.
11. Type in `schematic` for the View then click *Use Template*.
12. Set the *Name* to `Spectre`.
13. Enter `veriloga_PB` as the first item in the View List, then click *OK*.
14. In the Hierarchy Editor window, click *File – Save*. This is important.
15. Click the *Open* button to bring up the schematic.
16. Bring up an Analog Design Environment tool and use the states from the last Circuit Optimizer iteration.
17. Recall the states you saved from the last Circuit Optimizer iteration. Add the `power` variable and set it to -16.
18. Delete the previous analyses and set up a PSS analysis. Enter a new Fundamental Tone named `LO` and make it 2.354GHz. *Auto Calculate* the *Beat Frequency* and type `1` for the *Number of harmonics*.
19. Run the analysis and observe the intermediate differential voltages. The model is indeed now a passband model. At the higher power levels the LNA output contains odd

harmonics of the RF carrier. The filter reflects the odd harmonics back to the LNA and does not let them propagate forward. The baseband model does not simulate the odd harmonics but it does simulate the intermodulation term between the second harmonic and fundamental that falls at the fundamental. One reason to simulate the passband view is to check for peak voltage levels that might exceed voltage ratings. The baseband models only simulate peak voltage at the carrier fundamental, not the absolute peak.

**20.** Set up a swept PSS analysis. Sweep *power* from -32 to 0 in 10 steps.

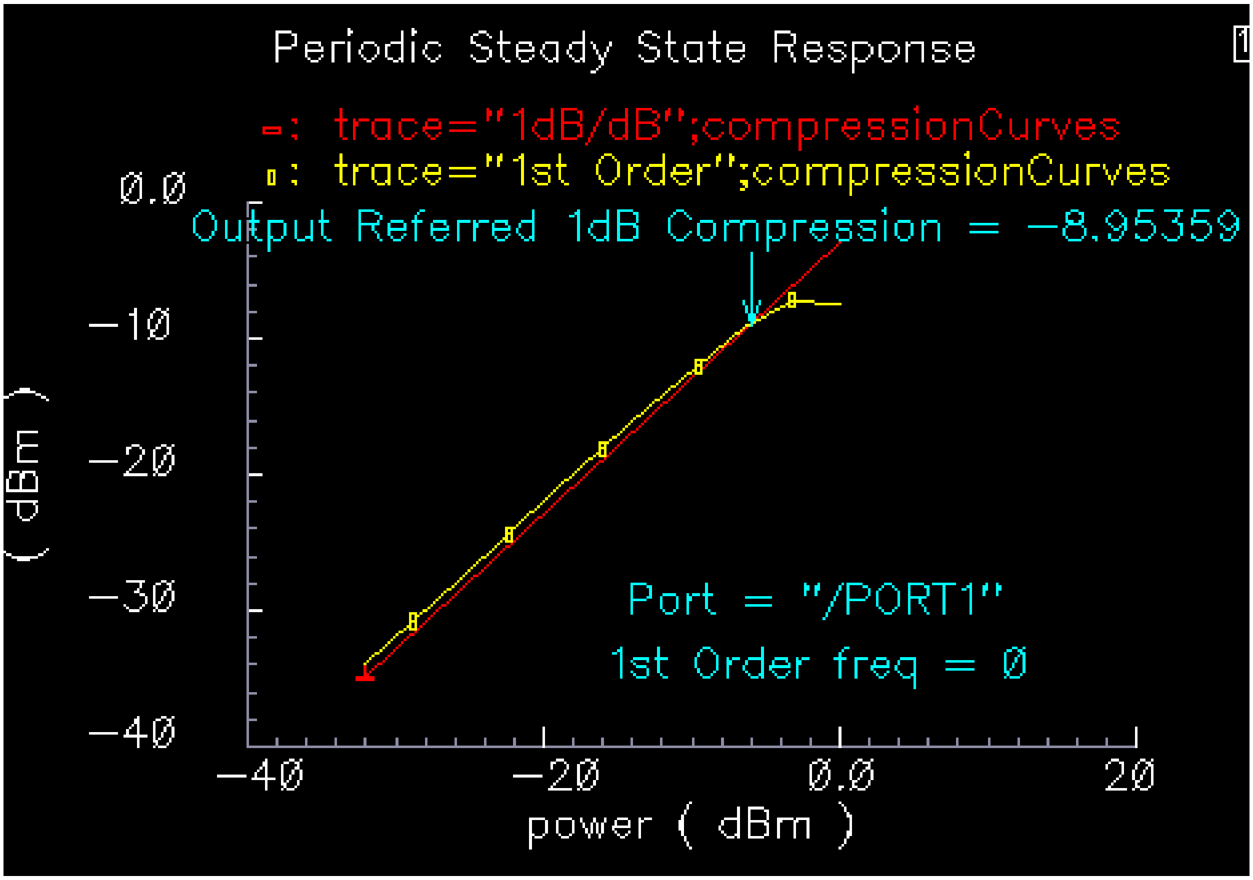
**21.** After the sweep finishes, click *Results – Direct Plot* in the Environment window, select *Compression point*, *1dB*, *Output Referred*. Select the 0 harmonic because the end-to-end system produces a baseband output. Then click the port loading the top output low pass filter. You should see the compression point plot shown in Figure 9-17 on page 797.

**Figure 9-17 End-to-end RF measurement, one dB compression point at the I-baseband output.**




**22.** Repeat the last step but this time click the lower output port. You should see the compression plot in Figure 9-18 on page 798.

**Figure 9-18 End-to-end RF measurement, one dB compression point at the Q-baseband output.**




## Comparing Baseband and Passband Models

This section illustrates how to compare baseband and passband models by:

1. Setting up a Transient analysis with the passband view
2. Setting up a Transient analysis with the baseband view
3. Directly comparing the baseband and passband models.

You run one analysis of the baseband view and two analyses of the passband view. You perform the second passband analysis with tightened tolerances.

1. Save the passband schematic under a different name. You use the new copy.
2. Repeat steps 9 through 17 from the last recipe for the new copy but do not enter the `veriloga_PB` view in the *View List* yet. You do a baseband analysis first.

3. Delete the port driving the LNA.

4. Delete the loaded voltage-controlled-voltage sources.

5. You need to synthesize an antenna signal. Add an `IQ_mod_BB` from the `top_dwnBB` category. Set the I and Q gains to 0 dB. Set the 1dB compression points to 1000 so that the modulator is ideal. Instantiate it in front of the LNA with the pins aligned, then wire the pins straight across. Ground the `phase_err` pin.

6. Drive the `I_in` pin of the IQ modulator with a port. Set the port frequency to 2MHz and name the frequency BB1. Set the amplitude to -16dBm.

7. Do the same for the `Q_in` modulator input.

8. Load or duplicate the states from the 12-iteration Circuit Optimizer analysis but delete the Noise and Transient analyses.

9. Remove AM/PM conversion from the LNA by setting the last parameter in the properties list to zero.

   It is not fair to compare passband and baseband views with AM/PM conversion because the passband view does not capture it.

10. Set up a 1us Transient analysis with default options.

11. Run the analysis and plot the filtered baseband outputs, the outputs of the low pass filters.

    Note how fast the simulation runs. Save the results so you can plot them again later. [1]

12. Switch to the passband view by entering “veriloga_PB” in the View List in the Hierarchy Editor. Click the update button in the hierarchy editor.

13. After you switch to the `veriloga_PB` view, edit the IQ modulator properties to set *flo* to `frf`. Edit the demodulator properties and set its *flo* to `flo1-frf.`

14. Click *Results – Printing Plotting Options* then click the *Overlay Plots* button.

15. Overlay the passband results with the baseband results. You see the waveforms in Figure 9-19 on page 800. The comparison is not very good.

16. Rerun the analysis with conservative options and set reltol to 1e-6. This run takes longer.

---

1. Note that the baseband outputs are out of phase with each other, even though the baseband inputs are in phase. In the baseband model, changing the RF-IF mixer LO from “flo1” to “-flo1” fixes the sign problem. In the passband model, the IQ_demodulator flo should be frf-flo1. To maintain the convention, in the baseband model the IF filter’s carrier frequency should be frf-flo1.

**17.** Plot the results.

**18.** Recall the saved baseband results and overlay them with those from the last simulation. You see the waveforms in Figure 9-20 on page 801. The passband results now lay right on top of the baseband results but took much longer to compute! It was not obvious without the baseband results that the first passband simulation did not run with tight enough numerical tolerances.

**Figure 9-19 Passband and baseband results with default options in the passband analysis**

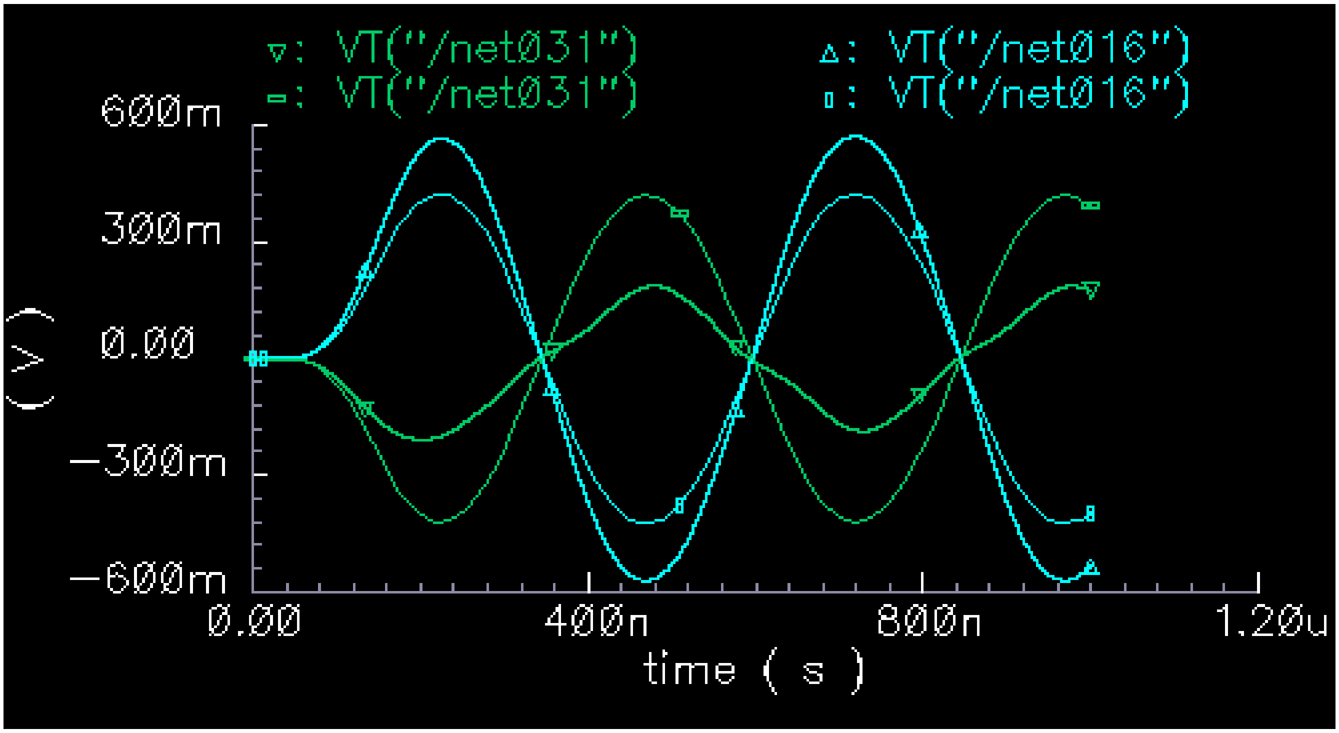
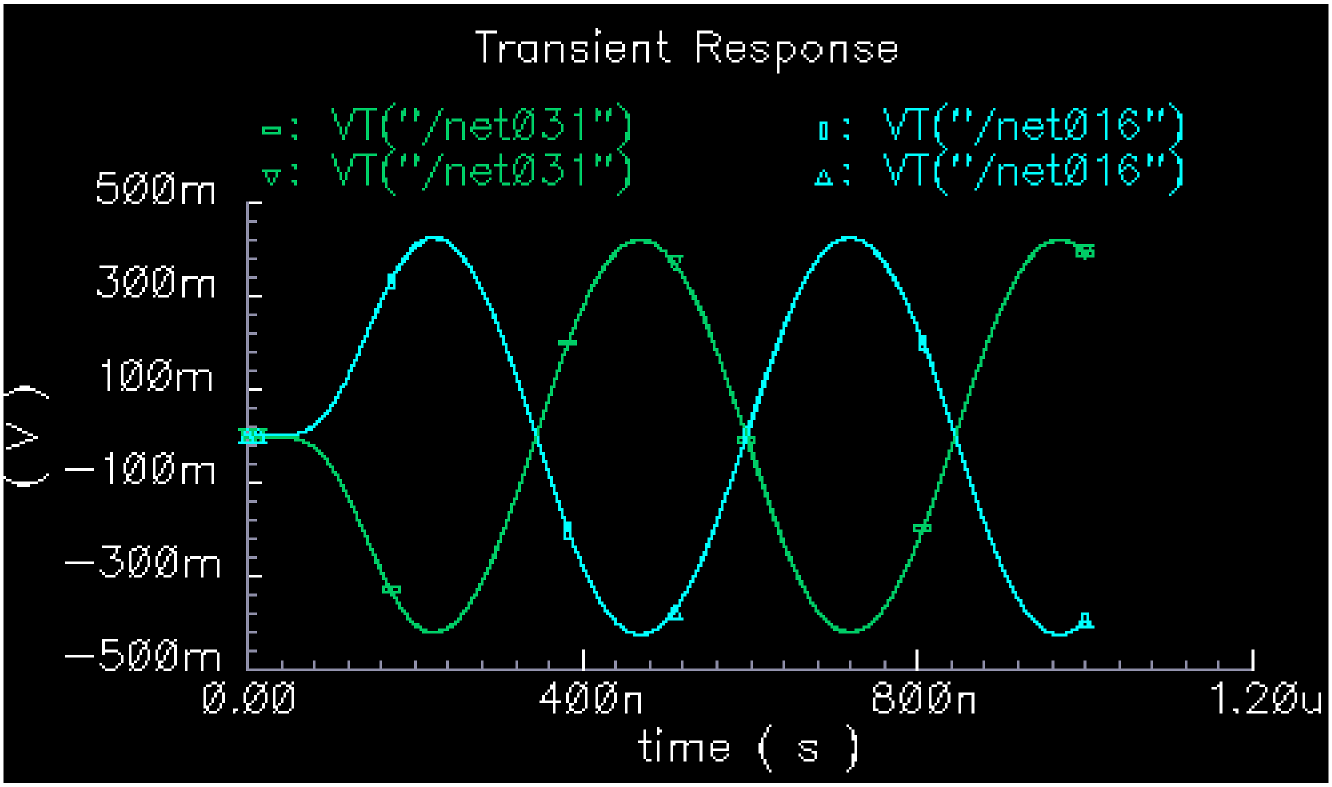


**Figure 9-20  Passband and baseband results with tighter options in the passband analysis**




# Relationship Between Baseband and Passband Noise

Noise analysis at baseband can be confusing because factors of two appear in a number of places throughout the calculations. For example:

- Each passband node becomes two equivalent baseband nodes.
    - Does noise injected at a passband node split between the two equivalent baseband nodes?
    - If so, does the noise split evenly?
- As shown in the `BB_test_bench` example, baseband models simulate peak in-phase and peak quadrature components of the carrier.

  When analyzing signal-to-noise ratios, does that mean you have to use half the square of the baseband signals?
- The analog design environment displays single sided power spectral densities.

Because the baseband power spectral density is the two-sided passband density shifted down, is there another factor of two because we can only see the baseband density for positive frequencies?

- White noise at the input of a mixer mixes up to the carrier from DC but there is also noise at twice the carrier that mixers down to the carrier.

  Does the baseband model account for this?

These questions are answered in this section.

Before sorting out the factors of two, note that baseband noise analysis is valid only for small signals. If any element in the architecture operates in a non-linear fashion, the noise analysis might be inaccurate. This is due to the fact that a baseband noise analysis follows a DC operating point analysis, rather than a PSS analysis.

Instantaneous incremental gain in a passband static non-linear model dithers at the carrier frequency.

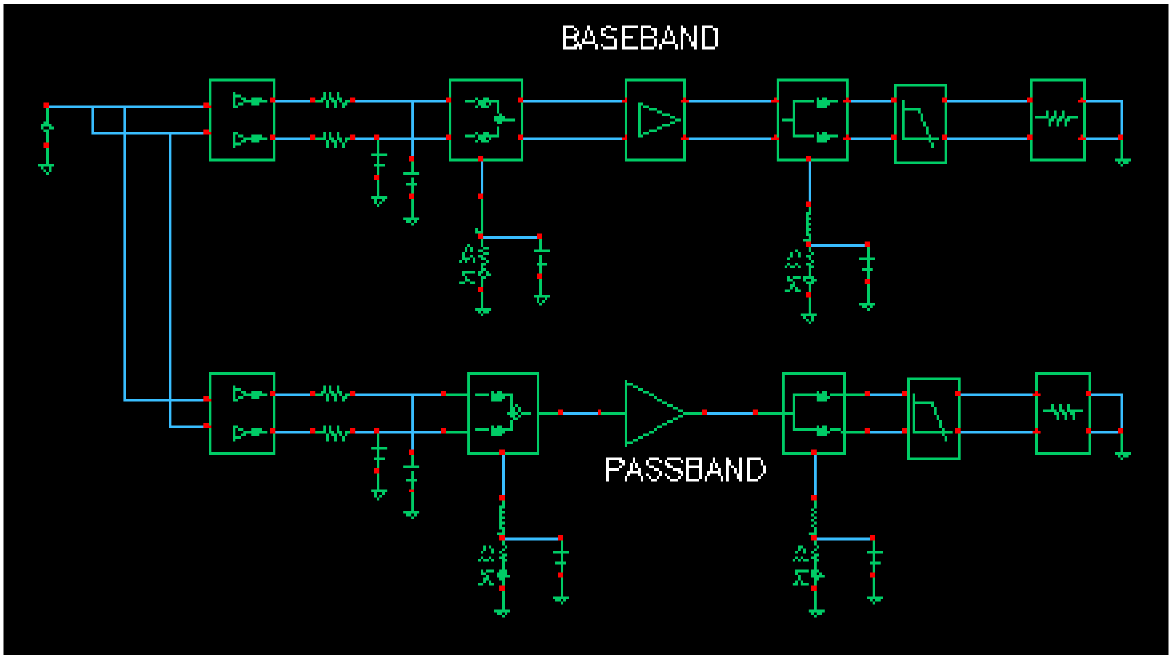
- When the carrier swings through zero, the incremental gain is large and noise at the input is amplified.
- When the carrier reaches its peak and drives the circuit into saturation, the incremental gain is smaller.

The average gain is greater than the minimum gain. The baseband model remains in the non-linear region because it only simulates peak voltages. Consequently, the incremental gain is always a minimum and the baseband model under-estimates the amount of noise propagating to the output. If the peak input signal drives the model into saturation, be sure to scale the baseband noise results accordingly.

## Introduction to Analysis

The circuit discussed here is called `noise_test_circuit` and you can find it in the `rfExamples` library. The circuit looks as shown in Figure 9-21 on page 803.

**Figure 9-21 The noise_test_circuit in the rfExamples Library**




The `noise_test_circuit` shows the relationship between baseband and passband noise. One branch consists of passband models. The other branch is a baseband equivalent of the first. You can assess noise at each of three observation points located in each branch of the circuit. At each observation point, you can examine both the noise and pnoise summaries.

The I and Q inputs are both driven by the same DC source so that you only have to view one baseband output, the other baseband output is identical by symmetry. Noise parameters in the passband and baseband models are identical. Aside from the behavioral blocks at the end of each branch, each behavioral block has noise injected at its input.

## Preparation Steps for Analyses

1. Set up a PSS analysis.

   Because the local oscillator is inside the passband mixer models you have to manually enter the frequency (1GHz) into the PSS analysis form. Let the beat frequency be *autocalculated* and use 1 harmonic.

2. Set up a pnoise analysis.

   Set the start frequency equal to 0 Hz, the stop frequency equal to 100MHz, use a linear sweep with 100 steps. Set the Maximum sideband to 1. For the input source select *none* and for the output use *voltage*. For the positive output node, select the I-input of the IQ_modulator in the passband branch. Use ground for the negative node.

**3.** Set up a noise analysis.

Sweep the frequency from 0 to 100MHz linearly in 100 steps. Use voltage for the output noise and select the I-input of the *IQ_mod_BB* component in the baseband branch for the positive node. Select ground for the negative node. Set the input noise port to any one of the ports in the circuit. Noise from that port does not affect either passband or baseband branches.

**4.** Run the analysis.

**5.** When the analysis finishes, go to the analog design environment simulation window and click *results – print – pss noise summary*.

**6.** In the form that appears, include All types, set *Type* to *integrated noise*, look at noise from 0 to 100MHz, select *truncate by number*, and view the top 2 noise contributors.

There are only 2 noise contributors at this point, noise at the I/Q inputs and noise from the low pass resistors.

**7.** Print the noise summary (which is different from the pnoise summary).

Again, print integrated noise (from 0 to 100MHz). Select All types and print noise from the top 2 noise contributors.

Figure 9-22 on page 805 shows the noise summaries. The two summaries agree because at this point in the circuit, both nodes are really baseband nodes.

**Figure 9-22 Noise Summaries for the First PSS PNoise and Noise Analyses**

```
Device     Param              Noise Contribution     % Of Total

/I2        IQ_modulator_i     1.56676e-12            70.06
/R20       rn                 6.69678e-13            29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
No input referred noise available


Device     Param          Noise Contribution     % Of Total
/I22       IQ_mod_BB_i    1.56676e-12            70.06
/R22       rn             6.69679e-13            29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
Total Input Referred Noise = Inf
```

**8.** Repeat the analyses.

This time select the `outi` pins on the low pass filters as outputs in the appropriate noise analyses. Again print the same noise summaries but this time look at the top 12 noise contributors in each summary.

Figure 9-23 on page 806 shows the noise summaries for the second analysis. The top 7 noise contributors in the baseband and passband branches agree. The remaining noise contributors are negligible, and should be negligible for the circuit because it has no AM/PM conversion.

**Figure 9-23  Noise Summaries for the Second PSS PNoise Analyses**

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I77 | PA_BB | 7.49879e-10 | 82.29 |
| /I2 | IQ_modulator_i | 9.69448e-11 | 10.64 |
| /R20 | rn | 4.1437e-11 | 4.55 |
| /I87 | butterworth_lp_i_noise | 1.52928e-11 | 1.68 |
| /I84 | IQ_demodulator | 7.64648e-12 | 0.84 |
| /PORT13 | rn | 3.77571e-14 | 0.00 |
| /PORT14 | rn | 3.77571e-14 | 0.00 |
| /I2 | IQ_modulator_q | 2.33139e-15 | 0.00 |
| /R21 | rn | 9.96506e-16 | 0.00 |
| /R22 | rn | 0 | 0.00 |
| /R23 | fn | 0 | 0.00 |
| /R23 | rn | 0 | 0.00 |

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
No input referred noise available
```

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I76 | PA_BB_i | 7.49861e-10 | 82.29 |
| /I22 | IQ_mod_BB_i | 9.69448e-11 | 10.64 |
| /R22 | rn | 4.14371e-11 | 4.55 |
| /I86 | butterworth_lp_i_noise | 1.52928e-11 | 1.68 |
| /I82 | IQ_demod_BB_i | 7.64642e-12 | 0.84 |
| /PORT15 | rn | 3.77571e-14 | 0.00 |
| /PORT9 | rn | 3.77571e-14 | 0.00 |
| /I76 | PA_BB_q | 1.80329e-14 | 0.00 |
| /I22 | IQ_mod_BB_q | 2.33136e-15 | 0.00 |
| /R23 | rn | 9.96493e-16 | 0.00 |
| /I82 | IQ_demod_BB_q | 1.54184e-31 | 0.00 |
| /R23 | fn | 0 | 0.00 |

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
Total Input Referred Noise = 0.0151541
```

**9.** Repeat the analysis again.

This time change the Pnoise sweep to run from 900MHz to 1.1GHz in 200 linear steps and select the power amplifier output as the noise output node. For the noise analysis, leave the sweep at zero to 100MHz, use 100 steps as before, and change the output node to be the I-output of the power amplifier.

Look at the top 7 noise contributors in each analysis. This time, integrate noise from 900MHz to 1.1GHz for the pnoise run and integrate noise from 0 to 100MHz for the noise run.

Figure 9-24 on page 807 shows the new summaries. Although the noise analyses agree at the ends of the branches, the noise analyses appear to disagree at a point where the baseband node is only a baseband equivalent, not a true baseband node.

**Figure 9-24 Noise Summaries for the Third PSS PNoise Analyses**

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I77 | PA_PB | 2.3769e-09 | 83.70 |
| /I2 | IQ_modulator_i | 1.53663e-10 | 5.41 |
| /I2 | IQ_modulator_q | 1.53663e-10 | 5.41 |
| /R20 | rn | 6.568e-11 | 2.31 |
| /R21 | rn | 6.568e-11 | 2.31 |
| /I84 | IQ_demodulator | 2.42376e-11 | 0.85 |
| /PORT13 | rn | 1.19682e-13 | 0.00 |

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83995e-09
No input referred noise available

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I76 | PA_BB_i | 2.3769e-09 | 83.69 |
| /I22 | IQ_mod_BB_i | 3.07294e-10 | 10.82 |
| /R22 | rn | 1.31347e-10 | 4.62 |
| /I82 | IQ_demod_BB_i | 2.42376e-11 | 0.85 |
| /PORT9 | rn | 1.19682e-13 | 0.00 |
| /I76 | PA_BB_q | 5.71606e-14 | 0.00 |
| /I22 | IQ_mod_BB_q | 7.38993e-15 | 0.00 |

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83997e-09
Total Input Referred Noise = 0.0148587

The *apparent* disagreement shown in Figure 9-24 on page 807 requires an explanation. Let us examine the noise contributors and try to answer some the questions we posed earlier.

### Noise at the power amplifier output due to noise injected at a passband node

- Passband model contributors: *PA_PB* and *IQ_demodulator*, port 9.
- Baseband model contributors: *PA_BB_i*, *IQ_demod_BB_i*, port 13.

Passband and baseband counterparts contribute the same amount of noise. However, in the baseband model, from symmetry you see the same numbers if you look at the Q-node. This means the baseband model predicts twice as much total noise due to noise injected between the modulators.

This factor of two is intentionally introduced to maintain the correct signal-to-noise ratio. The baseband model simulates peak signals; the carrier is suppressed. Without the carrier, signal power equals the square of the peak rather than one half of the square. This factor of two is not as arbitrary as it seems. The baseband model predicts the correct noise after demodulation because the passband demodulator model includes an extra factor of two to offset the factor of two inherent in the demodulation process.

Let the modulated carrier be `i(t)*cos(wc*t) -q(t)*sin(wc*t) + noise(t)`.

Now consider the I-output. To generate the I-output, the demodulator multiplies the signal by `cos(wc*t)`. The only part that propagates through the subsequent filter is `(1/2)i(t) + noise(t)*cos(wc*t)`. To recover `i(t)`, the passband demodulator model must scale this sum by two. (The baseband demodulator does not need to scale by two to extract the baseband signal because the carrier is suppressed.)

Thus, noise at passband demodulator model output equals `2*noise(t)*cos(wc*t)`. The filtered noise power density is then `4*(`*`input noise density`*`)/2`. The factor of 1/2 comes from the cosine. The filtered output noise density is twice the input noise density.

In the baseband model, doubling the noise injected at the passband nodes was not simply a matter of convenience.

So, to answer questions 1 and 2,

- Yes, noise injected at a passband node splits evenly between the two equivalent baseband nodes

  but

- Because each split is doubled, the ratio of *peak* signal to total noise equals the true signal-to-noise ratio.

Question 3 is rendered moot by using the noise summary and integrating over the proper band. If the pnoise analysis only integrated from 1GHz to 1.1GHz, (instead of from 900MHz to 1.1GHz), there would be a mysterious factor of two error.

In the baseband model, phase noise entering on the phase error pin propagates to both the I and Q outputs. In the baseband model, the same noise power appears on just the one output. Again, the total noise in the baseband model is twice that of the passband model to maintain the correct signal-to-noise ratio.

#### Noise injected at the modulator input (resistors and modulator noise)

Total noise in the baseband model due to modulator and input resistor noise is twice what it is from just one phase. Thus, the total noise due to sources on the input sides of the modulators differ by a factor of two. This occurs because the passband model is a real multiplier, which modulates the noise. If the peak signal voltage agrees with the baseband model, the passband modulator model attenuates input noise most of the time. The important thing is that the signal-to-noise ratios in the passband and baseband models agree anywhere in the system.

Now, copy the circuit, remove the capacitors at the modulator inputs, and repeat the last set of noise analyses. You see that in the passband model, the input resistors, R20 and R21, together contribute twice as much as the baseband counterpart, R22. With the capacitors, R20 and R21 together contribute just as much as R22. Without the capacitors, the input noise is truly white over the frequencies of interest. The same thing happens to the modulator noise itself. Figure 9-25 on page 810 shows the results.

In particular, the modulator now also has noise at twice the carrier frequency and that noise mixes down to the carrier frequency. The baseband model is just that, a baseband model. The answer to question 4 is *no*. The baseband models do not account for noise, or signals, at carrier harmonics. The baseband equivalent noise analysis is valid only if noise injected into the modulators has no power beyond the local oscillator frequency. Phase noise injected at the phase noise pins should also be band limited.

**Figure 9-25  Noise Summaries with the input capacitors removed.**

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I2 | IQ_modulator_q | 4.22571e-09 | 29.21 |
| /I2 | IQ_modulator_i | 4.22571e-09 | 29.21 |
| /I77 | PA_PB | 2.3769e-09 | 16.43 |
| /R21 | rn | 1.80619e-09 | 12.49 |
| /R20 | rn | 1.80619e-09 | 12.49 |
| /I84 | IQ_demodulator | 2.42376e-11 | 0.17 |

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 1.44651e-08
No input referred noise available
```

| Device | Param | Noise Contribution | % Of Total |
|---|---|---|---|
| /I22 | IQ_mod_BB_i | 4.22561e-09 | 50.11 |
| /I76 | PA_BB_i | 2.3769e-09 | 28.19 |
| /R22 | rn | 1.80615e-09 | 21.42 |
| /I82 | IQ_demod_BB_i | 2.42376e-11 | 0.29 |
| /PORT9 | rn | 1.19682e-13 | 0.00 |
| /I22 | IQ_mod_BB_q | 1.01619e-13 | 0.00 |

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 8.43322e-09
Total Input Referred Noise = 9.76997e-11
```

# 10

# Cosimulation with MATLAB and Simulink

This chapter describes how to set up and use a cosimulation link between the MATLAB® and Simulink® system-level simulation environment and Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF). The sections in this chapter are:

# Introduction to Cosimulation with MATLAB

Cosimulation combines the best of system-level simulation with lower-level analog and RF simulation. Simulink provides large libraries of DSP algorithms for generating complicated signals and post processing while Spectre RF supports transient and envelope analysis of common RF and communication circuits such as mixers, oscillators, sample and holds, and switched capacitor filters at both the transistor and behavioral levels.

Cosimulation makes it easier

- To detect concept errors early
- To detect design flaws before tape-out
- To quickly correct issues and resimulate

The system-level design in Simulink serves as a golden reference. System designers can use the Simulink system-level design as a testbench for implementing and simulating the design. Unfortunately, when system-level designs are simulated by themselves, the effects originating from subsystems are often not considered. With cosimulation, system designers can create low-level models of critical analog blocks and use these models one at a time to analyze the performance impact of individual blocks on the system-level simulation.

# Software Requirements

Cadence recommends that you use Cadence software version MMSIM 6.1 USR1 or later. With MMSIM 6.1 USR1 and MMSIM 6.2, use MATLAB R14 or MATLAB 2006. With MMSIM 6.2 USR1, use MATLAB 2007B or later.

# Setting Up and Running a Cosimulation

To prepare for and run a cosimulation, you

1. Connect the coupler block into the system-level Simulink schematic.
2. Determine how you want to start and run the cosimulation.
3. Generate a netlist for the lower-level block that reflects how you want to start and run the cosimulation.
4. Run the cosimulation.

These steps are described in greater detail in the following sections.

Before you continue, however, be sure that the programs are ready to run.

➤ Add the Spectre RF and MATLAB/Simulink engine install paths to the `MATLABPATH` environment variable.

You can automate this step by adding the appropriate command to your `.cshrc` file. For example, if you are using a C shell, you can add the command

```
setenv MATLABPATH `cds_root spectre`/tools/spectre/simulink:${MATLABPATH}
```

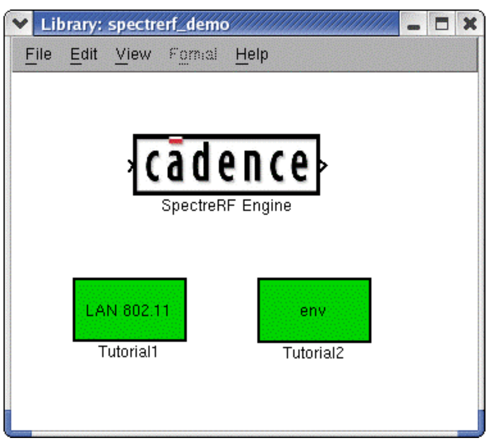
# Connecting the Coupler Block Into the System-Level Simulink Schematic

To prepare the Simulink part of the design for cosimulation,

1. Start MATLAB, by typing

```
matlab &
```

The Simulink library opens.




This library contains the coupler module (distinguished by the Cadence logo and labeled `SpectreRF Engine`). You can insert the coupler module in any Simulink design by dragging and dropping it from this library.

2. Open your testbench or high-level design.

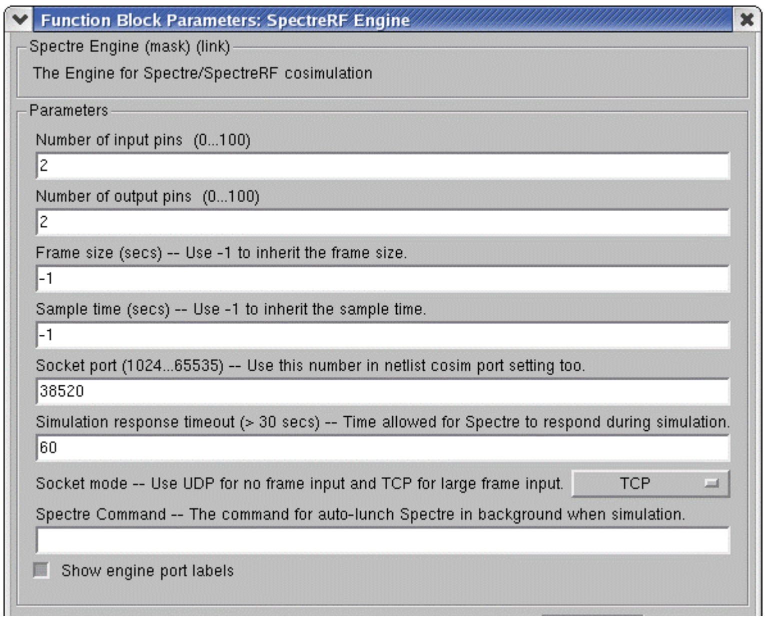
3. Drag and drop the `SpectreRF Engine` coupler block into the testbench and place it in the design.

4. Connect the `SpectreRF Engine` block into the design.

To make a signal connection, move the mouse pointer over the module pin. The pointer changes to a cross. Press the left mouse button, move the cursor to the destination pin and release the button.

**5.** Double-click the `SpectreRF Engine` block.

This opens the Function Block Parameters: SpectreRF Engine window where you can edit the coupler block's parameter values.




The fields have the following meanings:

| **Field** | **Meaning** |
|---|---|
| *Number of input pins* | The number of input pins to the `SpectreRF Engine` block (0…100). The input to the block is the input signal from the Simulink design. |
| *Number of output pins* | The number of output pins from the `SpectreRF Engine` block (0…100). The output from the block is the signal generated by the Spectre RF simulator. |
| *Frame size* | The number of samples per frame (any positive integer or -1; default: -1). Allows you to use the coupler in a loop where frame size inheritance does not work. Simulink automatically detects conflicting frame sizes. Set this value to -1 to inherit the frame size from the input. |

| | |
|---|---|
| *Sample time* | Sample period of the block, in seconds (default: -1). The *Sample time* is the frame period, not the time between the samples inside the frame. Allows the coupler to have only outputs, where the Simulink coupler acts as a source and defines a *Sample time*. Also allows the coupler to have only inputs. Set this value to -1 to inherit the sample period of the connected blocks. See the MATLAB and Simulink *Help* menu for more information on *Sample time* propagation. |
| *Socket port* | The number of the service that is identified by TCP/IP. Normally, the system reverses port numbers less than 1024, so this value can range from 1024 to 65535 (default: 38520). |
| *Simulation response timeout* | Maximum time, in seconds, to wait for an answer from the Spectre RF simulator during simulation. Increase this time if the simulator requires a long calculation time for each sample or frame (default: *120*). |
| *Socket mode* | TCP or UDP. Choose UDP if frame size is less 50. TCP and UDP are two different translate protocols in TCP/IP. Without CRC, UDP is usually faster in a good net environment with small data packages and TCP is usually better for large data packages. |
| *Spectre command* | Used to run a Spectre RF simulation. This parameter enables a use mode where Spectre RF is called internally by the Simulink simulation. |
| *Show engine port labels* | If checked, the `SpectreRF Engine` shows label information. |

**6.** Set the number of input pins and output pins as well as any other parameters you need to set.

**7.** Note the Spectre command field but do not change the value now.

Depending on how you choose to run the cosimulation, you might need to return to this field and type in a `spectre` command.

**8.** Click *OK*.

The form closes and the `SpectreRF Engine` block is updated with the correct number of pins.

**9.** (Optional) Make other necessary changes in the Simulink testbench window.

For example,

- Choose *Format – Port/Signal Displays – Signal Dimensions*.

  This switches the signal dimension display on so you can see details of framed signals.

- Choose *View – Simulink library* to display the library browser. Double-click *Math Operations*.

  The library appears. Here you can select Simulink converters, such as a `Complex to Real-Imag` block. Drag and drop blocks into the schematic and connect them as necessary.

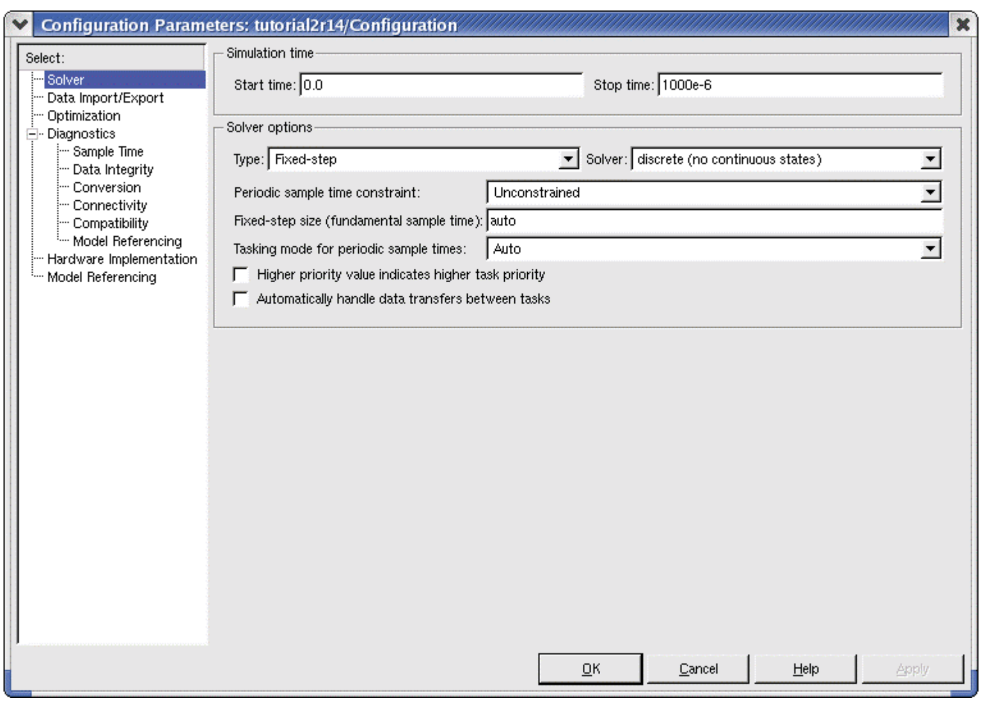
Double-click any placed blocks and set appropriate values.

**10.** Choose *Simulation – Configuration Parameters*.

The Configuration Parameters form opens.

**11.** In the *Stop time* field, set the stop time.

The Configuration Parameters form looks something like this:




**12.** Click *OK*.

The Configuration Parameters form closes.

**13.** Save the modified design by choosing *file – save as*.

This completes the necessary modifications to the Simulink testbench.

# Determining How You Want to Start and Run the Cosimulation

There are three ways to run a cosimulation, after all the setup is finished.

1. You can start the two applications (Spectre RF and MATLAB) separately.

   This method is appropriate if you need to be able to modify both the system-level Simulink design and the analog circuit.

2. You can start ADE and arrange to have MATLAB start automatically.

   This method is appropriate if you are an analog design who needs to validate a circuit with system-level design input and output.

3. You can start MATLAB and arrange to have Spectre RF start automatically.

   This method is appropriate if you are a system-level designer, because, after set up, you can use the SpectreRF Engine just like another block in Simulink.

The setup differs for each of these approaches so it is useful to decide which is most appropriate for your design before continuing.

# Generating a Netlist for the Lower-Level Block

The previous section describes how to insert a coupler block into the system-level design. That defines one end of the connection but you must still establish a connection with the lower-level analog block that is simulated by Spectre RF. To do that, you insert a `cosim` statement into the netlist, either by hand or by using the ADE environment.

## Preparing the Netlist When Using ADE

This section describes how to prepare a netlist for cosimulation using the Analog Design Environment (ADE).

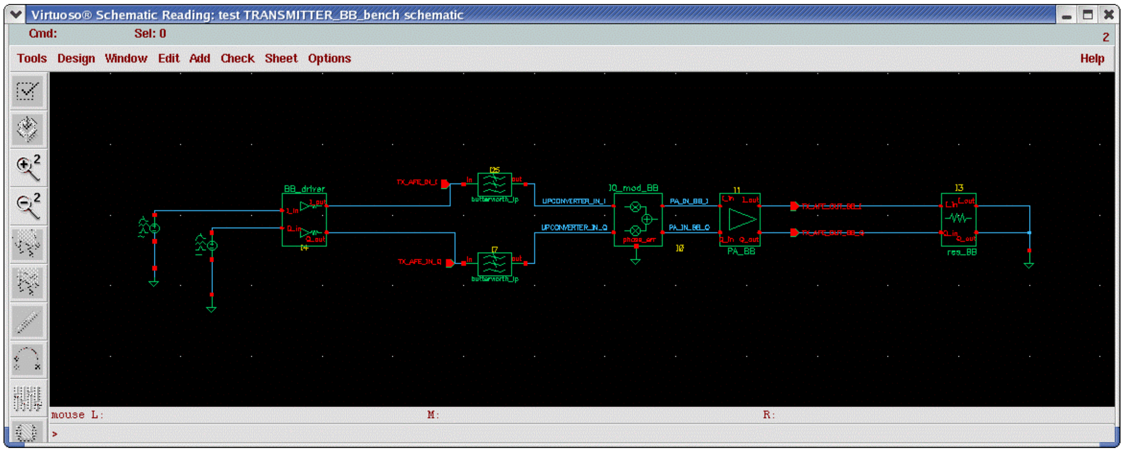
1. Start the icms tool.

```
icms &
```

The version of `icms` must be release IC5.1.4.1 USR4 ISR62 or later.

2. Open the schematic view of the cell.

   For example,



3. Open the Virtuoso Analog Design Environment from the schematic editor by choosing *Tools – Analog Environment*.

4. Choose *Setup – Matlab/Simulink – Setting*.

   The Cosimulation Options form appears.

5. In the Cosimulation Options form, click the *Select* button located beside *Cosimulation inputs*. Switch to the schematic viewer, where you see the following information below the schematic.

   ```
   Select source instance as cosimulation inputs. Press Esc when done.
   ```

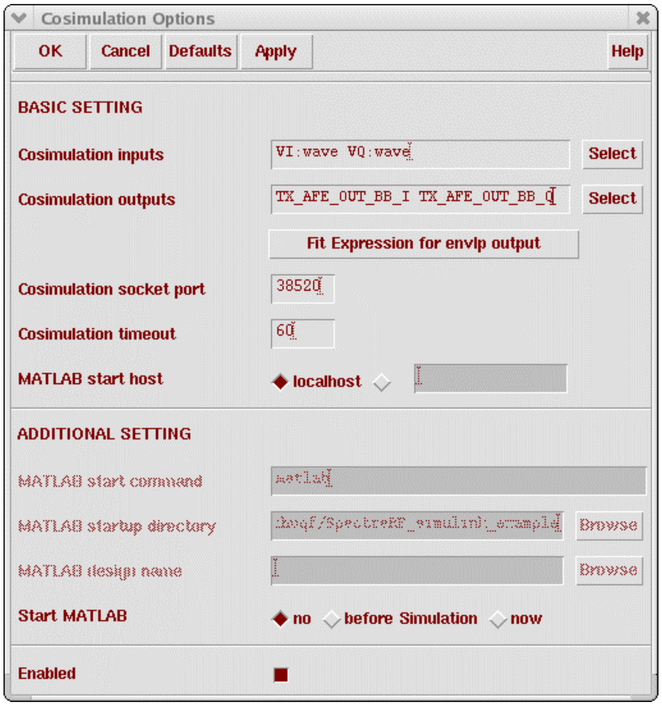
   Select the sources (which are connected to the outputs from the Simulink level of the design), then press the *Esc* key.

6. In the Cosimulation Options form, click the *Select* button beside *Cosimulation outputs*. Switch to the schematic viewer, where you see the following information below the schematic.

   ```
   Select Net/Terminal as cosimulation outputs. Press Esc when done
   ```

   Select the outputs (which are connected to the inputs at the Simulink level of the design), then press the *Esc* key.

The Cosimulation Options form looks something like this:




7. Ensure that the value of *Cosimulation socket port* is the same as the port value of the `SpectreRF Engine` block defined in "Connecting the Coupler Block Into the System-Level Simulink Schematic" on page 813.

8. In the Cosimulation Options form, turn on *Enabled*.

9. Examine the possible values for the *Start MATLAB* field.

| **Value** | **Behavior** |
|---|---|
| *now* | ADE launches a MATLAB session and opens a MATLAB desktop that is no different than it would be if you launched it manually. When you use the *now* value, you must start the simulation in the Simulink design before starting the Spectre RF simulation.<br><br>Using the *now* value is a good way to test whether ADE can start MATLAB and open the design successfully. |

| Value | Behavior |
|---|---|
| *before Simulation* | ADE launches an internal MATLAB session and no MATLAB desktop appears. ADE opens the specified MATLAB design and initiates the MATLAB simulation before Spectre simulation starts. ADE starts MATLAB once but cannot start it again before you close the first session.<br><br>Cadence suggests using the *now* value as a test before you run with the *before Simulation* value.<br><br>The *now* and *before Simulation* values share one log file, which can be opened by choosing *Setup – Matlab/ Simulink – Log file*. You can use the log file to monitor the simulation when no MATLAB desktop is visible. |
| *no* | The *MATLAB start command*, *MATLAB startup directory*, and *MATLAB design name* fields become active. When you use the *no* value, you must start the simulation in the Simulink design before starting the Spectre RF simulation. |

**10.** Continue the process of preparing the netlist according to how you want to start the applications that run the cosimulation.

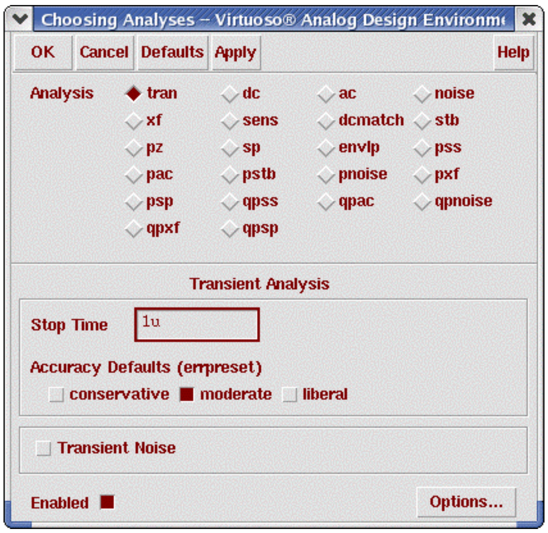
| To use this starting method... | Follow the guidance in this section... |
|---|---|
| Start the two applications (Spectre RF and MATLAB) separately. | "Preparing to Start the Two Applications Separately" on page 820 |
| Start ADE and arrange to have MATLAB start automatically. | "Preparing to Start ADE Manually and MATLAB Automatically" on page 821 |
| Start MATLAB and arrange to have Spectre RF start automatically. | "Preparing to Start MATLAB Manually and Spectre RF Automatically" on page 822 |

### Preparing to Start the Two Applications Separately

1. Set the *start MATLAB* value to *no*.
2. Click *OK* to close the Cosimulation Options form.
3. Use the Choosing Analyses form to set up the analysis.

The *Stop Time* can be any value. The Spectre RF simulator synchronizes the stop time with Simulink.

The Choosing Analyses window looks something like this:




4. Create a netlist and make sure the Simulink `cosim` statement appears in it.

### Preparing to Start ADE Manually and MATLAB Automatically

1. Set the value of the *Start MATLAB* field to *before Simulation*.
2. Type the name of the MATLAB design into the *MATLAB design name* field.

   Such designs have an extension of `.mdl`. For example, `env_d.mdl`. The MATLAB design must be in a location that is included in the `MATLABPATH` environment variable.
3. Set the value of the *Start MATLAB* field to *now*.

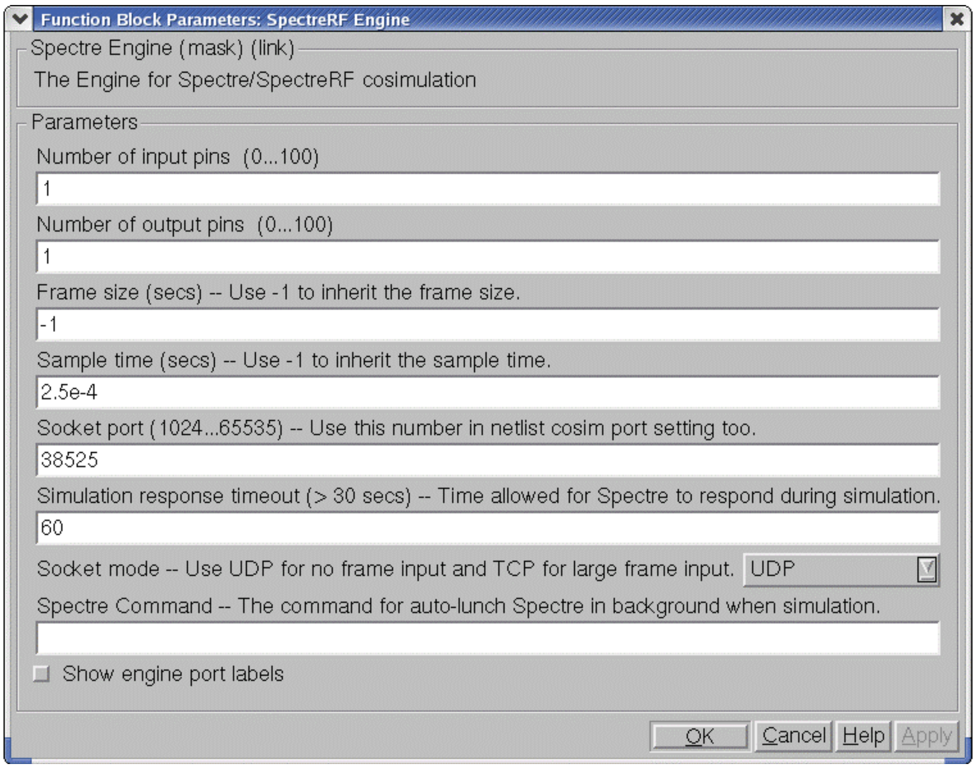
   A new MATLAB application starts and helps you open the Simulink design.
4. Double-click the `SpectreRF Engine` in the design and set the parameters.

   a. Set the *Sample time*. The Sample time can be set to -1, if you are uncertain about the appropriate time to use.

   b. Set *Socket Port* to the value you set in the Cosimulation Options form.

   c. Leave the *Spectre Command* field empty.

This field is used when you wish to start Spectre RF automatically after starting MATLAB manually.




**d.** Click *OK* to close the SpectreRF Engine form.

**5.** In the Cosimulation Options form, set *Start MATLAB* to *before Simulation*.

**6.** Click *OK* to close the Cosimulation Options form.

**7.** Use the Choosing Analyses form to set up the analysis.

**8.** In the ADE form, choose *File – Save* to save the design. Then exit from the MATLAB that was opened by ADE.

### Preparing to Start MATLAB Manually and Spectre RF Automatically

**1.** Set the value of the *Start MATLAB* field to *now*.

A new MATLAB application starts and helps you open the Simulink design.

**2.** Double-click the `SpectreRF Engine` in the design and set the parameters.

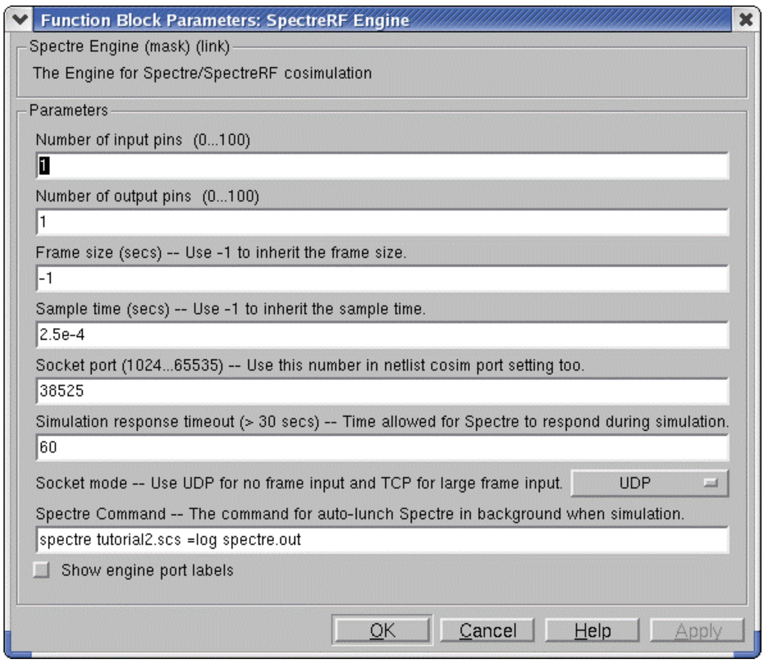
**a.** Set the *Sample time*. The Sample time can be set to -1, if you are uncertain about the appropriate time to use.

**b.** Set *Socket Port* to the value you set in the Cosimulation Options form.

**c.** Type a command similar to the following into the *Spectre Command* field.

`spectre` *`netlist_file`* `=log spectre.out`

The Function Block Parameters: SpectreRF Engine window looks something like this:




**d.** Click *OK* to close the Function Block Parameters: SpectreRF Engine form.

**3.** In the Cosimulation Options form, set *Start MATLAB* to *no*.

**4.** Click *OK* to close the Cosimulation Options form.

**5.** Use the Choosing Analyses form to set up the analysis.

**6.** In the ADE form, choose *File – Save* to save the design. Then exit from the MATLAB that was opened by ADE.

## Preparing the Netlist Without Using a Graphical User Interface

The steps in this section are necessary only if Spectre RF needs to start MATLAB automatically. If you are using the “Start MATLAB and arrange to have Spectre RF start automatically” starting method, ensure that there is no `cosim` statement in the netlist.

**1.** Edit the netlist file.

For example,

```
vi tutorial1/tutorial1.scs.
```

2. Add a `cosim` statement to the netlist.

   For example,

```
matlab cosim server="bj2lnx20"  port=38525 inputs=["VI:wave" "VQ:wave"]
outputs=[TX_AFE_OUT_BB_I TX_AFE_OUT_BB_Q]
```

The parameters for the `cosim` statement are described below:

**Table 10-1 Parameters of the cosim Statement**

| Parameter | Meaning |
|---|---|
| `design` | Refers to the file Simulink associates with the netlist.<br>`design="env_d.mdl"`<br>means the testbench whose file is `env_d.mdl`. |
| `inputs` | Input vector to identify the flow from MATLAB to `SpectreRF Engine`. The format is<br>`[“instance_name:wave” “instance_name1:wave” …]`<br>where `wave`, is a key word.<br>**Note:** The sequence of instances in square brackets follows the same one as the `SpectreRF Engine` port labels. This label information can be found by double-clicking the `SpectreRF Engine` and then by checking the *Show engine port labels* at the bottom in Function Block Parameters of the `SpectreRF Engine`.<br>After the inputs are set in the netlist, the sources associated with the inputs are meaningless and their parameters do not affect the simulation. |
| `outputs` | Output vector from `SpectreRF Engine` to MATLAB. This vector can be the net name or terminal name of interest. If net name is used, only voltage is given; if terminal name is used, only current is output.<br>To get current in an envlp analysis, you must add probes in the topology. |
| `port` | The port on the server running MATLAB. Its value should be the same as the value in the `SpectreRF Engine` in your current MATLAB design. |

**Table 10-1 Parameters of the cosim Statement,** *continued*

| | |
|---|---|
| `server` | The name of the machine that MATLAB starts. It can be set with the machine name or with the IP address of that machine. The accepted form is<br>`server = ”155.110.110.110”`<br>and<br>`server = “bj2lnx20”`<br>`server=localhost` means that Spectre and MATLAB use the same machine. |
| `silent` | Tells MATLAB whether to open the window during simulation. If `silent=yes`, the testbench window is opened. |
| `timeout` | The period of time MATLAB spends waiting for a response from the Spectre simulator.<br>`timeout=60`<br>stops MATLAB if it does not receive any response. |

**3.** Close and save the netlist.

After the above steps, the netlist is ready for simulation.

# Running the Cosimulation

With the coupler connected into the MATLAB design and with an appropriate netlist for the low-level design, you are ready to run the cosimulation. Note that these starting methods work only when the design is prepared as described earlier in this chapter.

| To use this starting method... | Follow the guidance in this section... |
|---|---|
| Start the two applications (Spectre RF and MATLAB) separately. | “Starting the Two Applications Separately” on page 826 |
| Start Spectre RF and arrange to have MATLAB start automatically. | “Starting Spectre RF Manually and MATLAB Automatically” on page 826 |
| Start MATLAB and arrange to have Spectre RF start automatically. | “Starting MATLAB Manually and Spectre RF Automatically” on page 827 |

## Starting the Two Applications Separately

1. Open the high-level design or testbench in the MATLAB design window.
2. In the MATLAB design window, choose *Simulation – Start*.

   The MATLAB desktop issues the following message.

   ```
   block 'modified/SpectreRF Engine': (COSIM_OK) Waiting for incoming connection
   on port 38520, timeout: 60 sec ...
   ```

   Then quickly do step 3.

   **Note:** The time interval between step 2 and step 3 must be within the *Simulation response timeout* defined as Function Block Parameters of the `SpectreRF Engine` (by double-clicking the `SpectreRF Engine`).

3. In the ADE window, click *Netlist and Run* or enter a spectre command at the command line. For example,

   ```
   spectre tutorial1/tutorial1.scs
   ```

   The cosimulation begins. The MATLAB desktop issues a message similar to the following when simulation ends.

   ```
   block 'modified/SpectreRF Engine': (COSIM_OK) Simulation finished
   ```

## Starting Spectre RF Manually and MATLAB Automatically

To run the cosimulation by starting Spectre RF,

- If you are using ADE, open the low-level design in the ADE window.

  a. Click *Netlist and Run*.

     The cosimulation begins. The MATLAB desktop issues a message similar to the following when simulation ends.

     ```
     block 'modified/SpectreRF Engine': (COSIM_OK) Simulation finished
     ```

- If you are using standalone Spectre RF,

  a. At the command line, enter a command similar to the following.

     ```
     spectre tutorial1/tutorial1.scs
     ```

     The cosimulation begins.

## Starting MATLAB Manually and Spectre RF Automatically

**a.** Start MATLAB.

```
matlab&
```

**b.** Choose *Simulation – Start*.

The MATLAB desktop issues a message similar to the following.

```
block 'env_d/SpectreRF Engine': (COSIM_OK) Waiting for incoming connection
on port 38525, timeout: 60 sec ...
block 'env_d/SpectreRF Engine': (COSIM_OK) Launch Spectre with commad
'spectre tutorial2.scs =log spectre.out'”
```

The cosimulation runs.

**c.** After the cosimulation finishes, review the MATLAB/Simulink output, close the MATLAB desktop, and exit from MATLAB.

# MATLAB Cosimulation Support Metrics for MMSIM7.1

| **MATLAB Release** | | **MATLAB Platform** | |
|---|---|---|---|
| | ***Linux 32 / 64*** | ***Solaris 32 / 64*** | ***Windows 32*** |
| R14SP1 | NOT Supported (1) | NOT Supported | Supported |
| R14SP2 | NOT Supported (1) | Supported | Supported |
| R14SP3 | NOT Supported (1) | Supported | Supported |
| R2006a | Supported | Supported | Supported |
| R2006b (2) | Supported | Supported (3) | Supported |
| R2007a (2) | Supported | Supported | Supported |
| R2007b (2) | Supported | Supported | Supported |
| R2008a (2) | Supported | Supported | Supported |
| R2008b (2) | Supported (4) | Supported (4) | Supported (4) |

- MATLAB Linux_32, Linux_64, Solaris_64, Solaris_32 platform testing is cosimulated with the corresponding MMSIM7.1 platform stream.
- MATLAB Windows_32 platform testing is cosimulated with MMSIM7.1 Linux stream.
- From MATLAB R2007a and forward there is only Solaris_64 platform MATLAB.
- From MATLAB R2006b, and backward, there is only Solaris_32 platofrm MATLAB.
- No ibmrs and sol86 platforms MATLAB from R2008b back to R14SP1.

**Note:** (1) Gcc library / OS Image compatability.

**Note:** (2) CCMPR00611059: Adjust Cosimulation information broadcasting order in MATLAB R2006b and forward.

**Note:** (3) Initialization failure in icfb, correlated with above issue (2).

(4) CCMPR00622034: Modify SpectreRF/MATLAB cosimulation tutorial case for MATLAB R2008b and later releases.

# 11

# Using PSS Analysis Effectively

Periodic steady-state (PSS) analysis is a prerequisite for all periodic small-signal analyses such as the Periodic AC (PAC), Periodic Transfer Function (PXF), Periodic S-Parameter (PSP) and Periodic Noise (Pnoise) analyses provided by Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF). PSS provides a rich set of parameters to help you adapt it to your own applications. For most circuits, PSS converges with the default parameter values. However, for some difficult circuits, changing the values of some parameters is necessary to achieve convergence.

This appendix describes methods you can use to remedy nonconvergence. This appendix also tells you how to improve convergence and efficiency using hierarchical PSS runs.

This appendix is divided into the following three main sections:

"General Convergence Aids" on page 829 describes techniques you can use to resolve PSS nonconvergence with both driven and autonomous circuits

"Convergence Aids for Oscillators" on page 831 describes techniques you use only with autonomous circuits such as oscillators

"Running PSS Analysis Hierarchically" on page 832 describes how to run a sequence of PSS analyses to improve the convergence, efficiency, and quality of the PSS solution used in subsequent periodic small-signal analyses

## General Convergence Aids

You can use the convergence aids described in this section to remedy PSS nonconvergence with driven as well as autonomous circuits. Autonomous circuits are usually harder to converge than driven circuits.

Adjusting the steadyratio and tstab Parameters

You can converge most difficult circuits by manipulating the `steadyratio` and `tstab` parameters.

The `steadyratio` parameter guards against false convergence. However, in unusual situations, the default value for `steadyratio` might be too conservative (1.0e$^{-3}$ is the default).

The PSS convergence criteria (for voltage-valued variables) is roughly

$$|\Delta v| < (reltol \times |vsig| + vabstol) \times steadyratio \times lteratio$$

To solve the periodic steady-state problem, Spectre RF simulation replaces the time derivatives in the one-period time interval with discrete differences. This turns the nonlinear, continuous differential equations into a set of discrete nonlinear equations that the Spectre RF simulator can solve.

The `steadyratio` parameter specifies the accuracy requirements for the discrete system. The discrete system might have its own solution (a steady state of the discrete difference equations) independent of what is happening in the continuous limit. In other words, the discrete system might be solved to zero tolerance. This happens frequently, particularly in driven circuits which is why setting a conservative `steadyratio` value generally works quite well.

In some cases, however, the solution to zero tolerance might not occur oscillators seem to be especially problematic. Avoid setting the convergence tolerance too tightly. For example, if `reltol` **=** 1.0e$^{-6}$ and `steadyratio` **=** 1.0e$^{-3}$, the relative tolerance for solving the discrete system is approximately `reltol` $\times$ `steadyratio` **=** 1.0e$^{-9}$. This tolerance level approaches the limit of precision the simulator can provide. In this situation, loosen `steadyratio` to 1.0 or 0.1 and reduce the precautions against false convergence.

Providing a larger value for `tstab` usually improves convergence. Occasionally, you must set `tstab` to a value equal to or greater than the time needed for the circuit to reach approximate steady state.

## Additional Convergence Aids

Below is a list of additional suggestions that sometimes help convergence.

- Carefully evaluate and resolve any notice, warning, or error messages.
- While trapezoidal rule ringing is simply annoying in transient analysis, in PSS analysis it can cause the shooting iteration to stall before convergence is achieved. You can remedy this problem by changing the PSS options `method` parameter from `traponly` to either `trap`, `gear2`, or `gear2only`.

- Help convergence by increasing the `maxperiods` parameter to increase the maximum iterations for shooting method to use. Sometimes a PSS analysis might simply need more than the default number of iterations to converge. However, in some situations convergence does not occur regardless of the number of iterations. In this case, increasing the iteration limit simply causes the simulation to take longer to fail.
- Decrease the maximum allowed time step to help convergence. To adjust the time step, either decrease the `maxstep` parameter or increase the `maxacfreq` parameter.
- Use the `errpreset` parameter properly. Use `liberal` for digital circuits; use `moderate` for typical analog circuits; use `conservative` for sensitive analog circuits (for example, charge storage circuits).

# Convergence Aids for Oscillators

Oscillator circuits are usually harder to converge than their driven counterparts. In addition to manipulating the `steadyratio` parameter, as discussed in “General Convergence Aids” on page 829, set the `tstab` parameter large enough so that the oscillation amplitude increases almost to its steady-state value and most other transients die out. You can estimate the required value of `tstab` by performing a transient analysis, or in the PSS analysis itself, set `saveinit = yes`.

For some circuits, the oscillation might die out before the oscillator builds up a final value, or the circuit might oscillate temporarily but then return to a zero state. Setting `saveinit = yes` lets you view the initial transient waveforms to identify the problem. This problem might be due to difficulty in starting the oscillator, or it might be caused by artificial numerical losses introduced by very large time steps. The latter is particularly likely if you set the `method` parameter to `gear2only`, `gear2`, or `euler`. In this case, you might try using `method = traponly`. If the problem persists, force the simulator to use smaller step sizes by decreasing `reltol` or by setting the `maxstep` parameter.

With autonomous PSS analysis, exclusive use of the trapezoidal rule can lead to ringing that spans the length of the oscillation period and causes convergence problems. When you set `method = trap`, the Spectre RF simulator occasionally takes a backward Euler step, which acts to damp the ringing. The `gear2` and `gear2only` methods use Gear’s backward difference method, which is not subject to ringing. Each of these alternatives to `traponly` avoids trapezoidal rule ringing and the attendant convergence problems at the expense of adding a small amount of artificial numerical damping. This damping slightly reduces the computed Q of the oscillator.

Be sure that the method you choose to start your oscillator is effective. It must *kick* the oscillator hard enough to start the oscillation and make the oscillator respond with a signal level that is between 25 and 100 percent of the expected final level. Avoid kicking the oscillator so hard that it responds in an unnecessarily nonlinear fashion. Also try to avoid exciting response modes in the circuit that are unrelated to the oscillation, especially those associated with long time constants.

Try to improve your estimate of the period. The simulator uses your estimate of the period to determine the length of the initial transient analysis interval. This interval is used to measure the oscillation period. If you specify a period that is too short, the estimate of the oscillation period is not accurate, and the PSS analysis might fail. Overestimation of the period is not a serious problem because the only disadvantage is a longer initial transient interval. However, significant overestimation can result in an excessively long simulation time.

Sometimes the analysis might need more than the default number of iterations (`maxperiods = 50`) to converge. This is more likely to occur with high-Q circuits. You can increase the maximum iterations for shooting methods using the `maxperiods` parameter.

# Running PSS Analysis Hierarchically

For most circuits, a single PSS analysis run is sufficient to find the periodic steady-state solution. However, for some difficult circuits it is preferable, or even necessary, to run PSS analysis multiple times to find the steady-state solution with the parameter settings you want (for example, with a tight `reltol`).

The biggest obstacle to PSS convergence is poorly chosen initial conditions. The backbone of PSS analysis is Newton's method. Theoretically, when the initial guess is close enough to the solution and the problem is not ill-conditioned, Newton's method is guaranteed to converge because of its contraction property. Consequently, it is very important to provide the best initial conditions that you can to ensure rapid convergence.

To run PSS hierarchically, you

1. Start by running a minimally accurate PSS analysis to obtain, with high likelihood, a coarse-grid solution that converges in an acceptably short simulation time.

2. Use the coarse-grid solution as the initial condition for another PSS run to achieve a fine-grid solution.

3. This practice significantly increases the chance that the fine-grid PSS analysis converges promptly. As you might expect, it is also more efficient. Running PSS hierarchically often reduces the total simulation time needed to find a periodic steady-state solution because it reduces the number of time-consuming fine-grid PSS iterations.

The `writefinal` and `readic` parameters serve as threads between hierarchical PSS runs.

- When you set `writefinal` to *SomeFileName*, both the associated time and period information (for autonomous PSS) and the final transient solution at PSS steady-state are saved to the file, *SomeFileName*.
- When you run another similar PSS analysis and you set `readic` to *SomeFileName* and `skipdc` to `yes`, setting `skipdc` to `yes` forces the simulator to use the initial conditions in the file *SomeFileName* as the initial transient solution for the first PSS iteration.

For example, if you want to find the periodic steady-state solution with a tight `reltol` such as $1.0e^{-5}$, you might

- Run an initial PSS analysis with a looser tolerance; for example, `reltol =` $1.0e^{-3}$
- Use the `writefinal` parameter in the initial PSS analysis to write out the final results to a file. A PSS analysis runs faster with the looser tolerance because fewer time points are generated during each transient integration performed during each PSS iteration.
- Run a second PSS analysis with the tighter tolerance, `reltol` **=** $1.0e^{-5}$
- Use the *readic* and *skipdc* in the second PSS analysis to read in the final results of the first PSS analysis as the initial conditions.
- After the second PSS analysis, you can run small-signal analyses such as PAC.

```
set1 set reltol=1.0E-3
pss1 pss ... writefinal="SomeFile"
set2 set reltol=1.0E-5
pss2 pss ... readic="SomeFile" skipdc=yes
...  pac ... ...
```

Always use a sequence of PSS runs when a you need a tight tolerance PSS solution. If necessary, you can run more than two PSS analyses in the hierarchical process. You choose the tolerance sequence for the continuation. The multiple PSS approach usually produces a better periodic steady-state solution for subsequent small-signal analyses.

The above procedure is often called a continuation on the simulation parameter `reltol`. However, you can use continuation with many other simulation parameters. For example, in order to achieve PSS convergence at a high input power that causes nonconvergence, you might gradually increase the input power at an RF port.

# 12

# Using QPSS Analysis Effectively

Quasi-Periodic Steady-State (QPSS) analysis computes the quasi-periodic steady-state responses of circuits with multiple periodic inputs of differing frequencies. The QPSS analysis is a prerequisite for all quasi-periodic small-signal analyses such as the quasi-periodic AC (QPAC), quasi-periodic transfer function (QPXF), quasi-periodic S-Parameter (QPSP), and quasi-periodic noise (QPnoise) analyses provided by Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF).

A quasi-periodic signal has multiple fundamental frequencies. Closely spaced or incommensurate fundamental frequencies cannot be efficiently resolved by PSS. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) QPSS allows you to compute responses to several moderately large input signals in addition to a strongly nonlinear tone which represents the LO or clock signal. The circuit is assumed to respond in a strongly nonlinear fashion to the large tone and in a weakly nonlinear fashion to the moderate tones. You might use a QPSS analysis, for example, to model intermodulation distortion with two moderate input signals. QPSS treats one of the input signals (usually the one that causes the most nonlinearity or the largest response) as the large signal, and the others as moderate signals.

The QPSS analysis employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. When input signals are stiff, that is, when they have large time constants, the QPSS analysis with its MFT algorithm is more efficient than transient circuit simulation. MFT also performs better than traditional algorithms such as harmonic balance when solving highly nonlinear problems.

A typical application of QPSS analysis is to predict the harmonic distortion of switched-capacitor filters that operate under widely separated fundamentals. See an example in “Switched Capacitor Filter Example” on page 845.

QPSS can also predict intermodulation distortion of a narrowband circuit that is driven by a local oscillator (LO) and two high-frequency input fundamentals that are closely spaced in frequency, such as in the down conversion stage of a receiver. See an example in “High-Performance Receiver Example” on page 846.

This appendix contains the following:

- A discussion of when to use the QPSS analysis
- A brief description of the MFT method
- A comparison of the QPSS and PSS analyses
- A comparison of the QPSS and PAC analyses
- Two examples of typical use: a switched capacitor filter and a high-performance receiver
- Procedures for setting up and running QPSS analysis

## When Should You Use QPSS Analysis

The increasing demand for low-cost, mobile communication systems increases the need for efficient and accurate simulation algorithms for RF communication circuits. These circuits are difficult to simulate because they process modulated carrier signals that consist of a high-frequency carrier and a low-frequency modulation signal. Typically, the carrier frequency ranges from 1-5 GHz while the modulation frequency ranges from 10 kHz to 1 MHz. If you were to apply a standard transient analysis to such a circuit, it might require simulating the detailed response of the circuit over hundreds of thousands of clock cycles (or millions of timepoints), a generally impractical approach.

Fortunately, many RF circuits of interest operate near a time-varying, but periodic, operating point. You can analyze some of these circuits if you assume that one of the circuit inputs produces a periodic response that you can calculate using a PSS analysis. You can treat other time-varying circuit inputs as small signals and linearize the circuit around the periodic operating point. You can then analyze small signals efficiently using the Spectre RF Periodic AC (PAC) analysis [8]. This approach lets you avoid long simulation times with transient analysis [7]. See “QPSS and PSS/PAC Analyses Compared” on page 844 for more information.

However, many RF circuits cannot be analyzed efficiently with the periodic-operating-point plus small-signal approach. For example, predicting the intermodulation distortion of a narrowband circuit, such as a receiver down converter (a mixer followed by a filter), requires calculating the nonlinear response of the mixer circuit, driven by a LO, to two closely spaced high-frequency inputs. The response to both inputs is within the band width of the filter. The steady-state response of such a circuit is quasi-periodic.

As a further complication, many multi-timescale circuits, such as mixers and switched-capacitor filters, have a highly nonlinear response with respect to one or more of the exciting inputs. Consequently, steady-state approaches such as multi frequency harmonic balance do not perform well.

The mixed frequency-time (MFT) approach used for QPSS analysis avoids these difficulties. MFT methods assume that many circuits of engineering interest have a strongly nonlinear response to only one input, such as the clock in a switched-capacitor circuit or the LO in a mixer, and respond in a weakly nonlinear manner to other inputs.

Compared to previous MFT methods [2, 4], the MFT algorithm used for QPSS analysis has the following advantages.

The MFT algorithm used for QPSS analysis

- Avoids the ill-conditioning caused by poorly chosen boundary conditions found in previous algorithms
- Uses a multi-dimensional discrete Fourier transform (DFT) scheme for cycle placement
- Uses a continuation method to enhance the global convergence of Newton's method
- Uses a matrix-implicit, Krylov-subspace-based iterative scheme that enables MFT methods to solve large problems
- Uses a preconditioning strategy that permits the iterative solver to converge rapidly

If you are unfamiliar with terminology such as *matrix-implicit*, *Krylov-subspace*, and *preconditioning,* you can find detailed descriptions in reference [6]. You can find an introduction to *continuation methods* in reference [1].

## Essentials of the MFT Method

Circuit behavior is usually described by a set of nonlinear differential-algebraic equations (DAEs) that can be written as,

**(12-1)** $$\frac{d}{dt}Q(v(t)) + I(v(t)) + u(t) = 0$$

Where

- $Q(v(t)) \in \Re^N$ is typically the vector of sums of capacitor charges at each node
- $I(v(t)) \in \Re^N$ is the vector of sums of resistive currents at each node
- $u(t) \in \Re^N$ is the vector of inputs
- $v(t) \in \Re^N$ is the vector of node voltages

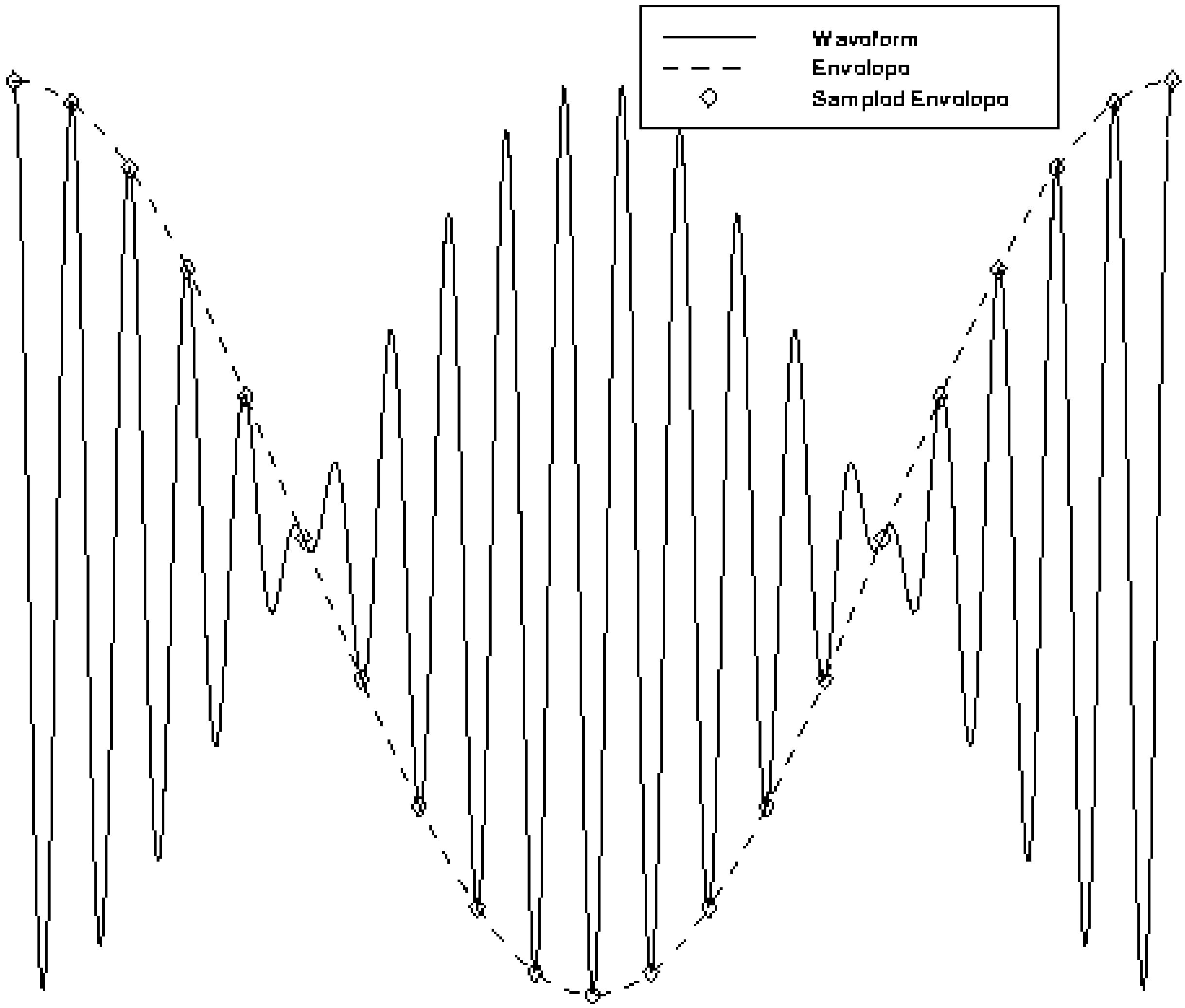
- $N$ is the number of circuit nodes

The MFT algorithm assumes that the circuit is in quasi-periodic steady-state; that is, that the signals can be represented as,

**(12-2)** $$v(t) = \sum_k \sum_l V_{kl} e^{j2\pi(l f_0 + k f_1)t}$$

where, for simplicity, there are only two fundamental frequencies, $f_0$ and $f_1$. The signal $v(t)$ is then sampled at one of the fundamental frequencies, $f_0$, which is called the *clock* signal. This is shown in Figure 12-1 on page 839, where sampling a two-fundamental quasi-periodic signal at one of the fundamental frequencies creates a sampled waveform that is one-fundamental quasi-periodic, or simply-periodic. MFT directly finds the solution that, when sampled at $f_0$, is periodic in $f_1$.

**Figure 12-1  Sample Envelope Shown Sampled at the Waveform Peaks**




The sample envelope shown in Figure 12-1 is the waveform traced out when the signal is sampled with the clock period. The envelope is shown sampled at the peaks but this is not necessary.

The sampled waveform is,

**(12-3)** $$\bar{v}_n = v(nT_0) = \sum_{k=-\infty}^{\infty} \bar{V}_k e^{j2\pi k f_1 t}$$

Where

$T_0 = 1/f_0$

Alternatively, you can also write,

**(12-4)** $\bar{v}_n = F^{-1}\bar{V}$

which states that

$\bar{v}$

is the inverse Fourier transform of

$\bar{V}$

Recall that $v$ is a solution of the circuit equations and that

$\bar{v}$

is simply $v$ uniformly sampled, so given

$\bar{v}_n$

you can compute a subsequent sample point

$\bar{v}_{n+1}$

using Equation 12-5,

**(12-5)** $\bar{v}_{n+1} = \phi(\bar{v}_n, nT_0, (n+1)T_0)$

In Equation 12-5,

- $\phi$ is the state transition function for the circuit

- $\phi(v_{0,}\, t_{0,}\, t_1)$ is the solution for the circuit equations at $t_1$, given that it starts from the initial condition $v_0$ at $t_0$.
- Consider the $n^{\text{th}}$ sample interval. Then let

$$x_n = \bar{v}_n$$

be the solution at the start of the interval and let

$$y_n = \bar{v}_{n+1} = x_{n+1}$$

be the solution at the end of the interval. Equation 12-5 uses the circuit equations to relate the solution at both ends of the interval as shown in Equation 12-6.

**(12-6)** $$y_n = \phi(\bar{v}_n, nT_0, (n+1)T_0)$$

Let $X = F_{\mathrm{x}}$ and $Y = F_{\mathrm{y}}$ ($X$ and $Y$ are the Fourier transforms of $x$ and $y$). Then, from Equation 12-3 and because $y_{\mathrm{n}} = x_{\mathrm{n+1}}$,

**(12-7)** $$X_k = e^{-j2\pi k f_1 T_0} Y_k$$

Or

**(12-8)** $$X = D_{T_0} Y$$

Where

$$D_{T_0}$$

is a diagonal delay matrix with

$$e^{-j2\pi k f_1 T_0}$$

being the $k^{\text{th}}$ diagonal element.

Together, Equations 12-6 and 12-8 make up the MFT method.

In practice,

$$\bar{V} = X$$

is band-limited, so only a finite number of harmonics are needed. In addition, if the circuit is driven with one large high-frequency signal at $f_0$, which is called the *clock* signal, and one moderately-sized sinusoid at $f_1$, only $K$ harmonics are needed and the method is efficient. With only $K$ harmonics, Equation 12-6 on page 841 is evaluated over $2K + 1$ distinct intervals that are spread evenly over one period of the lowest beat tone. Therefore, the total simulation time is proportional to the number of harmonics needed to represent the sampled waveform. The simulation time is independent of the period of the lowest-frequency beat tone or the harmonics needed to represent the clock signal.

Equation 12-8 on page 841, along with the associated Fourier transforms, relate the starting and ending points of the solution of the circuit equations over each interval, and consequently represent a boundary-value constraint on Equation 12-6 on page 841.

Shooting methods are the most common method for solving boundary-value problems. Their use of transient analysis to solve the circuit equations over an interval brings two important benefits.

Transient analysis handles abruptly discontinuous signals efficiently because the timestep shrinks to follow rapid transitions.

Transient analysis easily handles the strongly nonlinear behavior of the circuit as it responds to the large clock signal.

# QPSS and PSS Analyses Compared

Like PSS analysis, QPSS analysis uses a shooting Newton method. However, instead of doing a single transient integration, each Newton iteration does transient integrations over a number of nonadjacent clock periods. Each of the integrations differs by a phase-shift in each moderate input signal. The number of integrations is determined by the number of harmonics of moderate fundamentals that you specify. Given `maxharms` = [`k1`, `k2`, ..., `kS`], the total number of integrations is

$$\prod_{s=1}^{S} (2k_s + 1)$$

Consequently, the efficiency of the algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals.

Fortunately, the number of harmonics of the clock does not significantly affect the efficiency of the shooting algorithm. The boundary conditions of a shooting interval are such that the time-domain integrations are consistent with a frequency-domain transformation with a shift of one large-signal period.

As a Spectre RF user, you might need to run a PSS analysis to calculate the steady-state operating point of circuits with multiple input frequencies. You can do this by making the PSS beat frequency the greatest common factor of all the input frequencies.

When the greatest common factor is relatively close to the clock frequency; for example, within a factor of `10`, PSS analysis might be preferable to QPSS analysis for two reasons.

Each nonlinear iteration would not require excessive integrations of clock period.

1. PSS analysis solves a much smaller linear system in each nonlinear iteration.
2. The size of the linear system resulting from QPSS analysis is

$$K = \prod_{s=1}^{S} (2k_S + 1)$$

times as big as that resulting from PSS analysis.

In general,

- If the ratio of *clock frequency/beat frequency* for a PSS analysis is smaller than $K$, a PSS analysis might be preferable to a QPSS analysis.
- For circuits such as switched-capacitor filters that operate on wide timescales, where the ratio of *clock frequency/beat frequency* for PSS analysis can be greater than 1000 — QPSS is clearly the analysis of choice.

Additional differences between the PSS and QPSS analyses are that PSS analysis does not have any restrictions on input sources, whereas QPSS analysis requires that all nonclock inputs must be sinusoidal. Also the QPSS analysis cannot be applied to autonomous circuits.

Like the PSS analysis and the periodic small-signal analyses, you must run the QPSS analysis to determine the quasi-periodic operating point before you can run the quasi-periodic small-signal analyses.

# QPSS and PSS/PAC Analyses Compared

Like PAC analysis, the QPSS analysis calculates responses of a circuit that exhibits frequency translations. However, instead of having small-signal linear behavior, QPSS models the response as having components of a few harmonics of input-signal frequencies. This permits computing responses to moderately large input signals.

PAC analysis assumes that only the clock is generating harmonics. For example, for a clock frequency $f_c$, and a small-signal frequency $f_s$, amplitudes of circuit response are generated at $f_s \quad k_c f_c$ where $k_c$ is bonded by the parameter `harms` in PSS analysis. In contrast, QPSS also permits nonclock fundamentals to generate harmonics. In the same situation, a spectrum at frequencies $k_s f_s \quad k_c f_c$ is generated, where $k_s$ and $k_c$ are bonded by the QPSS parameter `maxharms`.

# QPSS Analysis Parameters

While QPSS analysis inherits most PSS parameters directly, the QPSS analysis adds two new parameters and extends the meaning of a few parameters. The two new parameters are the most important QPSS parameters.

- `funds`
- `maxharms`

They replace the PSS parameters, `fund` (or `period`) and `harms`, respectively.

The `funds` parameter accepts a list of names of fundamentals that are present in the sources. You specify these names in the sources using the `fundname` parameter.The simulator figures out the frequencies associated with the fundamental names.

An important feature of the `funds` parameter is that each input signal can be composed of more than one source. However, these sources must all have the same fundamental name. For each fundamental name, its fundamental frequency is the greatest common factor of all frequencies associated with the name.

The first fundamental is considered the large signal. You can use a few heuristics to pick the large fundamental.

- Pick the fundamental that is not sinusoidal.
- Pick the fundamental that causes the most nonlinearity.
- Pick the fundamental that causes the largest response.

The `maxharms` parameter accepts a list of numbers of harmonics needed for each fundamental.

If you do not list all the fundamental names using the `funds` parameter, the current analysis is skipped. However, if you do not specify `maxharms`, a warning message is issued, and the number of harmonics defaults to 1 for each fundamentals.

QPSS analysis expands the role of two PSS parameters.

- `maxperiods`
- `tstab`

The maxperiods parameter that controls the maximum number of shooting iterations for PSS analysis also controls the maximum number of shooting iterations for QPSS analysis.

The tstab parameter controls both the length of the initial transient integration, while only the clock tone is active, and the number of stabilizing iterations, while both the clock tone and the moderate tones are active. The stabilizing iterations run before the Newton iterations begin.

The remaining QPSS analysis parameters are inherited directly from PSS analysis, and their meanings remain essentially unchanged.

The `errpreset` parameter quickly adjusts several simulation parameters. In most cases, `errpreset` should be the only parameter you need to adjust. See The errpreset Parameter in QPSS Analysis in *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for information about the `errpreset` parameter.

This is demonstrated by the following two examples

- A switched capacitor filter
- A high-performance receiver

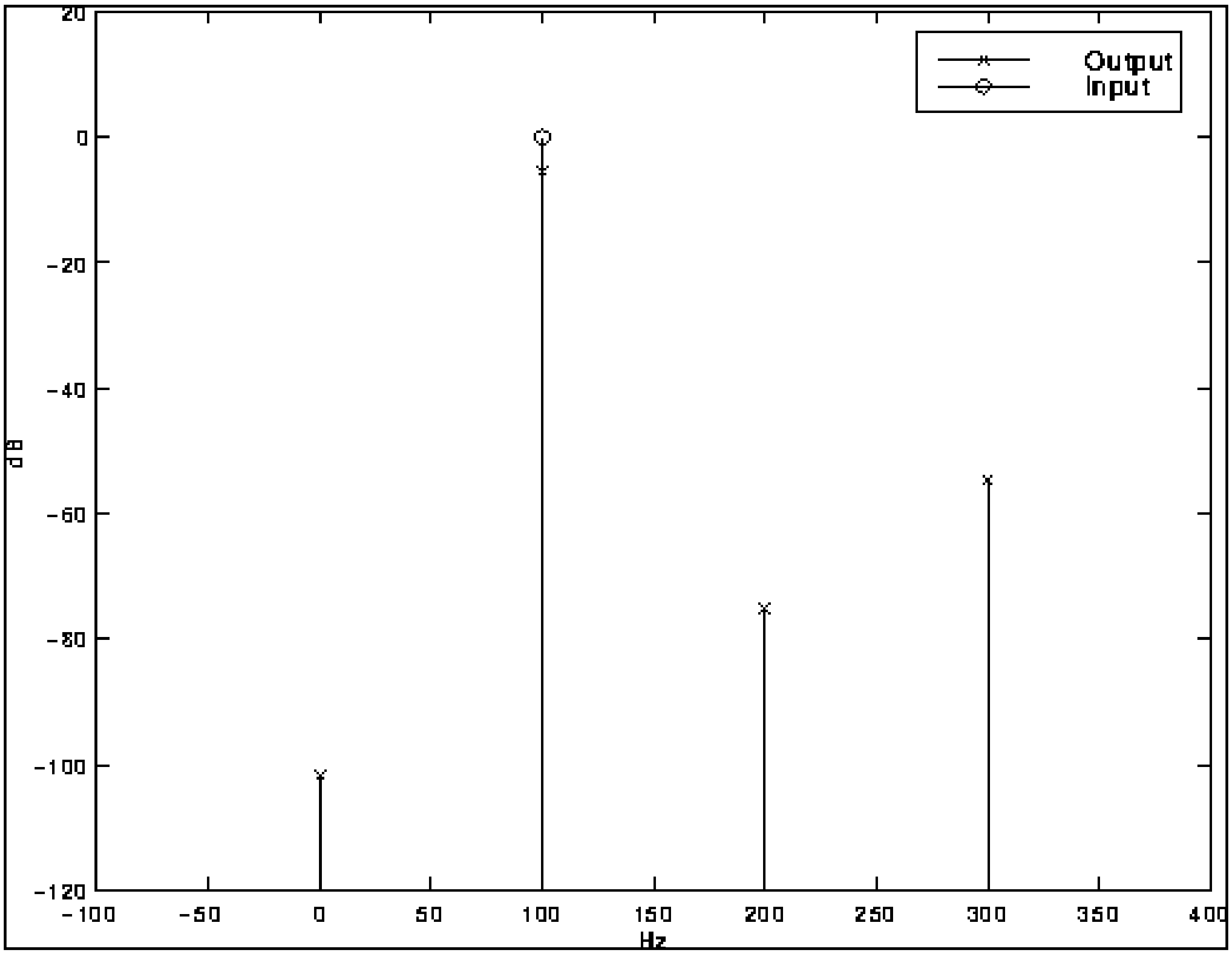
## Switched Capacitor Filter Example

The low-pass switched-capacitor filter example has a 4 kHz bandwidth and 238 nodes, resulting in 337 equations. To analyze this circuit, the QPSS analysis was performed with an 8-phase 100 kHz clock and a 1*V* sinusoidal input at 100 Hz.

The 1000-to-1 clock-to-signal ratio makes this circuit difficult for traditional circuit simulators to analyze. Three harmonics were used to model the input signal. The eight-phase clock required about 1250 timepoints for each transient integration. The total number of variables solved by the analysis is 337 X (2 X 3 +1) X 1250 = 2,948,750, slightly less than three million. The simulation completes in less than 20 minutes on a Sun UltraSparc1 workstation with 128

Megabyte memory and a 167 MHz CPU clock. A swap file is used because the analysis cannot be finished in core. For more information, see "Memory Management" on page 853. Figure 12-2 on page 846 shows the harmonic distortion.

**Figure 12-2 Harmonic Distortion of a Switched Capacitor Filter**




## High-Performance Receiver Example

The high-performance image rejection receiver example consists of a low-noise amplifier, a splitting network, two double-balanced mixers, and two broad-band Hilbert transform output filters combined with a summing network that suppresses the unwanted sideband. A limiter in the LO path controls the amplitude of the LO. It is a rather large RF circuit that contains 167 bipolar transistors and uses 378 nodes. This circuit generated 987 equations in the simulator.

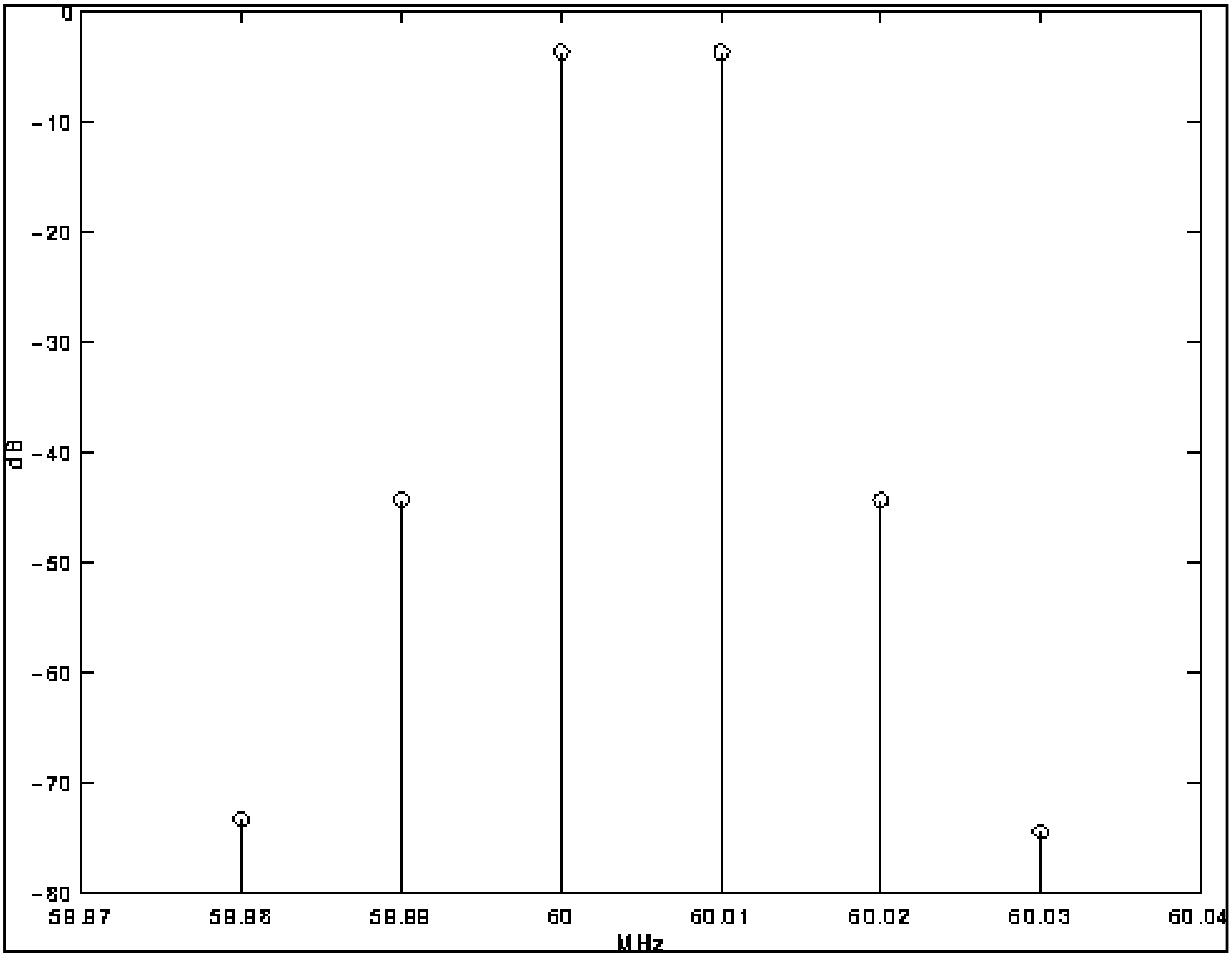
To determine the intermodulation distortion characteristics, the circuit is driven by a 780 MHz LO and two 50 mV closely placed RF inputs, at 840 MHz and 840 MHz+10 KHz, respectively. Three harmonics are used to model each of the RF signals, and 200 time points are used in each transient clock-cycle integration, considered to be a conservative accuracy specification for this circuit. As a consequence, about 10 million unknowns are generated

$$987 \times (2 \times 3 + 1)^2 \times 200 = 9{,}672{,}600$$

The simulation requires 55 CPU minutes on a Sun UltraSparc10 workstation with 128 megabytes of physical memory and a 300 MHz CPU clock. A swap file is used because the analysis cannot be finished in core. For more information, see "Memory Management" on page 853. Figure 12-3 on page 848 shows the 3rd and 5th order distortion products.

To appreciate the efficiency of the MFT method, consider that traditional transient analysis needs at least 80,000 cycles of the LO to compute the distortion, a simulation time of over two days. Additionally, the results might be inaccurate because of the large numerical error accumulated by integrating over so many cycles. In contrast, the MFT method is able to resolve very small signal levels, such as the 5th order distortion products shown in Figure 12-3.

**Figure 12-3 Intermodulation Distortion of a High-Performance Receiver**




# Running a QPSS Analysis

The following sections describe how to set up and run a QPSS analysis. They also present ways to promote convergence.

## Picking the Large Fundamental

Your first task is to select the large fundamental, called the clock or the LO. Below are a few guidelines for selecting the large fundamental.

- Choose the one that is not sinusoidal.

- Choose the one that causes the most nonlinearity.
- Choose the one that causes the largest response.
- Choose the one that has the highest frequency.

## Setting Up Sources

You can specify the clock input using any type of source. However, other fundamentals can only be sinusoidal sources.

In addition to specifying the waveform parameters such as `type`, and `ampl`, you must also use the parameter `fundname` to specify a name for each non-DC source. Each fundamental can be a composition of several input sources with the same name. This is a difference between QPSS and other analyses.

Example 12-1 shows how to set up the sources for the “Switched Capacitor Filter Example” on page 845.

An eight-phase clock:

### Example 12-1 Setting Up the Sources for the Switched Capacitor Filter

```
//*****************BEGIN NETLIST**********************************
             .
             .
             .
// Clocks
Phi1  (phi1  gnd)  vsource type=pulse period=2.5us delay=0.25us\ width=1us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi2  (phi2  gnd)  vsource type=pulse period=2.5us delay=1.5us\ width=1us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi8  (phi8  gnd)  vsource type=pulse period=5.0us delay=1.5us\ width=2.25us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi9  (phi9  gnd)  vsource type=pulse period=5.0us delay=1.25us\ width=2.75us
val0=VDD val1=-VDD rise=10ns fundname="Clock"
Phi10 (phi10 gnd)  vsource type=pulse period=10.0us delay=3.75us\ width=5.25us
val0=VDD val1=-VDD rise=10ns fundname="Clock"
Phi11 (phi11 gnd)  vsource type=pulse period=10.0us delay=4.0us\ width=4.75us
val0=-VDD val1=VDD rise=10ns fundname="Clock"

// Input source
Vin   (pin   gnd)  vsource type=sine freq=100_Hz ampl=1 sinephase=0\
fundname="Input"
// QPSS Analysis
harmDisto QPSS funds=["Clock" "Input"] maxharms=[3 3] +swapfile="SomeFileName"
//*****************END NETLIST************************************
```

Example 12-2 shows how to set up the sources and analysis for the “High-Performance Receiver Example” on page 846.

### Example 12-2 Setting Up the Sources for the High Performance Receiver

```
//*****************BEGIN NETLIST**********************************
             .
             .
             .
// LO input
Vlo (8 7) vsource type=sine ampl=400mV freq=780MegHz fundname="LO"
//RF inputs
Prf1 (15 44) port  type=sine ampl=50mV freq=840MegHz fundname="F1"
+ ampl2=50mV freq2=840.01MegHz fundname2="F2"
//Analysis
InterModDisto QPSS funds=["LO" "F1" "F2"] maxharms=[5 3 3]
+ swapfile="SomeFileName"
//*****************END NETLIST**********************************
```

## Sweeping a QPSS Analysis

You can combine a QPSS analysis with a Spectre circuit simulator Sweep analysis to create a powerful tool for a wide variety of applications, such as IP3 and IP5 calculations. For example, you might want to calculate the distortion for input power ranging from -60 dBm to 0 dBm. The netlist for this task is shown in Example 12-3.

### Example 12-3 Calculating the Input Power Distortion from -60 to 0 dBm

```
//*****************BEGIN NETLIST*********************************
             .
             .
             .
parameters inputpower=-60
// Clock

Vlo (clock 0) vsource type=pulse val0=-1 val1=1 period=1n delay=0\ rise=10p
fall=10p width=460.00p fundname="carrier"
//  Input port
Prf (input 0)  port  type=sine r=50.0 num=1.0
+freq=+9.0E+08  dbm=inputpower fundname="RF1"
+freq2=+9.05E+08 dbm2=inputpower fundname2="RF2"
//  Analysis

swp sweep param=inputpower start=-60 stop=0 lin=6 {
distortion QPSS funds=["LO" "RF" "RF2"] maxharms=[5 3 3]
}
//*****************END NETLIST**********************************
```

Always arrange the sweep values so that analyses that converge more easily are performed first. When you sweep QPSS, it automatically uses the converged steady-state solution of the previous QPSS analysis. As discussed in "Convergence Aids" on page 851, this practice can also aid convergence.

# Convergence Aids

Normally QPSS analysis converges with default parameter settings, but occasionally you might need to adjust some parameter settings in order to achieve convergence.

Normally, giving a sufficiently large `tstab` parameter value or a looser `steadyratio` value resolves convergence problems during the initial QPSS stages. For convergence problems during the QPSS iterations, try the following procedures.

- Increase the tstab parameter value.
- The tstab parameter controls both the length of the initial transient integration, with only the clock tone activated, and the number of stabilizing iterations, with the moderate tones activated. The stable iterations are run before Newton iterations begin.
- Increase the `steadyratio` parameter value.
- The `steadyratio` parameter guards against false convergence. Its default values, `1.0` for `liberal`, `0.1` for `moderate`, and `0.01` for `conservative`, are derived from the `errpreset` parameter. Tighten `steadyratio` only if you suspect false convergence.
- Sometimes `steadyratio` must be loosened (for example to `steadyratio` =1) particularly with a tight `reltol` setting. Also loosen `steadyratio` when convergence stagnates. An indication of stagnation is that the `convNorm` value, which you can see on the screen, fluctuates within a certain range and never decreases further.
- The convergence tolerance of QPSS is determined by the product of `steadyratio` and `reltol`. Normally, you do not set `steadyratio` to a value higher than `10`.
- Severe trapezoidal rule ringing can prevent convergence.
- If you suspect trapezoidal rule ringing, use `method` = `gear2` or `method` = `gear2only`.
- Avoid using unnecessarily tight `reltol` settings.
- Excessively tight `reltol` significantly reduces efficiency besides causing convergence problems.
- Do a continuation on a parameter, such as input power, of moderate fundamentals.
- When the circuit is behaving in a highly nonlinear manner at a certain input power level, the PSS plus PAC approach might not compute a good enough estimate of the initial condition. One effective strategy is to ramp up the input power gradually by carefully arranging a sweep as described in "Sweeping a QPSS Analysis" on page 850. Because it is usually much easier to achieve convergence at a low power input level where the

circuit behaves in a more linear manner, start with a low input power. After the simulation converges, save the steady-state solution as the initial condition for the next input power level. This process repeats automatically as QPSS is swept. You can achieve convergence at the desired input power level if the sweep steps are sufficiently small.

- In general, avoid using an excessively high number of harmonics for the moderate fundamentals.
- Using too many harmonics lengthens the simulation time and uses a lot of memory. However, if the number of harmonics you use is not high enough for a particular fundamental to adequately model its nonlinear effects, convergence problems might also occur.

An important indication of convergence or divergence is the `Conv` value printed to the screen. There are a few typical scenarios shown in Figure 12-4 on page 853.

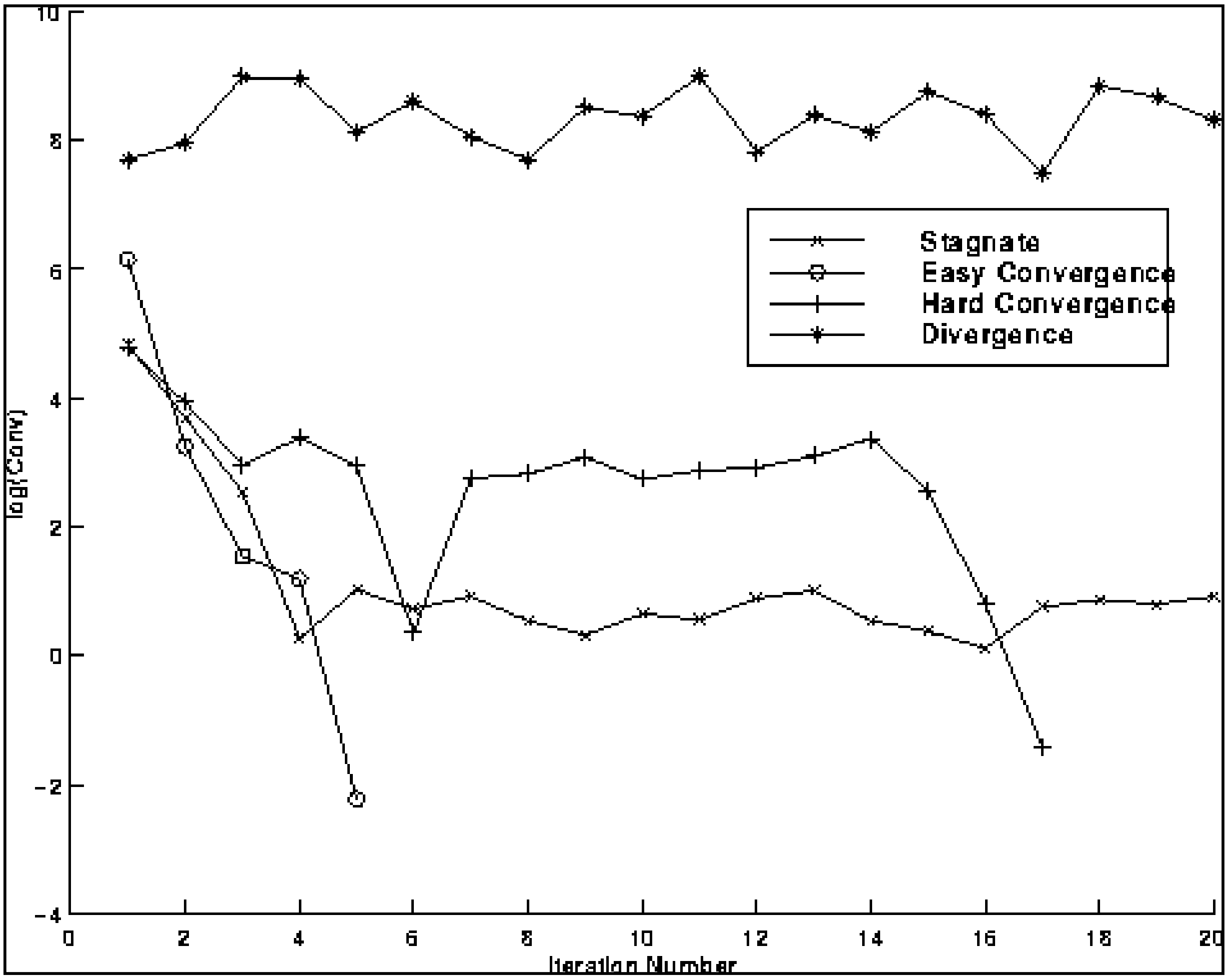
For most QPSS runs, the *Easy Convergence* scenario occurs.

A few simulations follow the *Hard Convergence* scenario.

If QPSS iterations *Stagnate* (the `Conv` value fluctuates close to but above `1.0`), loosen `steadyratio`. Loosening `steadyratio` solves the stagnation problem.

If the iterations show *Divergence*, you usually must improve the initial condition. Do a continuation on input power level as described within the preceding list.

**Figure 12-4 QPSS Convergence Scenarios**




# Memory Management

QPSS is a memory-intensive analysis. If QPSS cannot be finished in core with real physical memory, use a swap file residing on a local disk. The simulator manages swapping much more efficiently than the operating system. The examples found in “Setting Up Sources” on page 849 depict how to set up the `swapfile` parameter. Typically, 80% CPU utilization is achieved.

**Note:** The `swapfile` parameter is supported for only the shooting engine.

# Dealing with Sub-harmonics

One advantage of QPSS analysis over PSS is that you need only provide a name for each fundamental frequency. The actual beat frequency value associated with the name is calculated automatically by the simulator. For each unique fundamental name, the simulator first finds all the frequencies associated with it. Then the greatest common factor is calculated among these frequencies, which is used as the beat frequency associated with the fundamental name.

However, if the fundamental frequency has sub-harmonics (circuit responses at some fraction of a driven frequency, typically 1/2 or 1/3), as with a divider, for example, the simulator currently cannot detect them. As a workaround, add a dummy source that tells the simulator of the existence of a sub-harmonic associated with a fundamental frequency.

# Understanding the Narration from the QPSS Analysis

The examples in this section describe information that is typically printed to the screen during a QPSS analysis run.

After initialization, the Spectre RF simulator confirms the fundamental tone names that were read and their beat frequencies. For this example, this circuit has

- A 1 GHz fundamental tone named `flo` as the large or clock signal.
- Two fundamental tones named `frf` and `fund2`, as moderate input signals. The frequency for `frf` is 900 MHz and the frequency for `fund2` is 920 MHz.

```
     .
     .
     .
Fundamental flo:  period = 1 ns, freq = 1 GHz.
Fundamental fund2:  period = 1.08696 ns, freq = 920 MHz.
Fundamental frf:  period = 1.11111 ns, freq = 900 MHz.
**************************************************************
Quasi-Periodic Steady State Analysis ‘qpss’: largefund = 1 GHz
****************************************************************
=============================
‘qpss’: time = (1 ns -> 2 ns)
================================
     .
     .
     .
```

QPSS prints the following message and begins the initial transient iteration.

```
Starting qpss analysis iterations.
```

The initial transient iteration runs with only the clock tone (the large signal) active and all moderate input signals suppressed. Unlike PSS analysis (where each iteration performs a single transient integration), each QPSS iteration performs a number of transient integrations. In this example, 25 transient integrations are performed. The `tstab` parameter controls the length of the initial transient integration.

For the 1st transient integration, QPSS prints data as it steps through the integration. For the 2nd through 25th integrations, QPSS prints a countdown timer for each integration.

At the end of the 25th transient iteration, QPSS prints out a summary that includes `tstab`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)

2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
'tstab' iter = 1, convNorm = 64.3, maximum dI(rif:p) = 51.5933 uA, took 5.49 s.
     .
     .
     .
```

One or more stabilizing transient iterations run with all signals active; the clock tone (the large signal) and all moderate input signals. As for the first transient iteration, each QPSS iteration performs a number of transient integrations. This example performs 25 transient integrations.

The `tstab` parameter controls the number of stabilizing iterations run when all tones are active.

At the end of each stabilizing transient iteration, QPSS prints out a summary that includes `tstab`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)

2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
'tstab' iter = 2, convNorm = 15.2, maximum dI(rif:p) = -43.8125 uA, took 5.41 s.
     .
     .
     .
```

The Newton iterations run after the stabilizing iterations.

The QPSS analysis employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. The large tone is resolved in the time domain and the moderate tones are resolved in the frequency domain (hence the name mixed frequency time algorithm). The QPSS analysis uses the shooting Newton method as its backbone. However, unlike PSS analysis where each Newton iteration performs a single transient integration, for each Newton iteration the QPSS analysis performs a number of transient integrations.

When you set up a QPSS analysis, you determine the number of integrations performed by the number of moderate fundamental harmonics you select. The efficiency of the shooting Newton algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals. The number of harmonics of the large fundamental does not significantly affect the efficiency of the Newton algorithm. The boundary conditions of a shooting Newton interval are such that the time domain integrations are consistent with a frequency domain transformation with a shift of one large signal period.

The shooting Newton iterations run with all signals active. As for the stabilizing transient iterations, each shooting Newton iteration performs a number of transient integrations. This example performs 25 transient integrations.

At the end of each shooting Newton iteration, QPSS prints out a summary that includes `Newton iter`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)
2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 1, convNorm = 28.3, maximum dI(rif:p) = 9.16928 uA, took 7.58 s.
     .
     .
     .
```

In this example, four Newton iterations were performed to reach the steady-state solution. Information about the QPSS analysis including the steady-state solution print at the end.

```
     .
     .
     .
1st_Transient_Integration:
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 2, convNorm = 1.35, maximum dI(L1:1) = 927.427 nA, took 8.29 s.
1st_Transient_Integration:
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 3, convNorm = 4.21e-03, maximum dI(q56:i_extra) = -20.204 nA,
took 8.24 s.
1st_Transient_Integration:
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 4, convNorm = 1.05e-06, maximum dI(q56:i_excess) = -478.108 fA,
took 5.78 s.
qpss: The steady-state solution was achieved in 6 iterations.
Number of accepted qpss steps = 360 each in 25 time intervals.
Starting spectrum calculation.
Total time required for qpss analysis ‘qpss’ was 42.83 s.
     .
     .
     .
```

Occasionally, you might see warning messages such as the following

```
Minimum time step used.  Solution might be in error.
```

or

```
Junction current exceeds ‘imelt’.  The results computed by Spectre are now incorrect
because the junction current model has been linearized.
```

You can ignore these warning messages if they appear in the early stage of QPSS iterations. They might be caused by bad starting integration conditions and do not affect the final solution. However, if they appear in the final iteration, the solution might be in error.

# References

[1] A. Allgower and K. Georg, *Numerical Continuation Methods*, Springer-Verlag, New York, 1990.

[2] L. O. Chua and A. Ushida, “Algorithms for computing almost periodic steady-state response of nonlinear systems to multiple input frequencies,” *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 953-971, 1981.

[3] D. Feng, J. Phillips, K. Nabors, K. Kundert, and J. White, “Efficient computation of quasi-periodic circuit operating conditions via a mixed frequency/time approach,” Submitted to *Proceedings of the 36th Design Automation Conference*, June 1999.

[4] K. Kundert, J. White, and A. SangiovanniI-Vincentelli, “A mixed frequency-time approach for distortion analysis of switching filter circuits*,” IEEE Journal of Solid State Circuits*, vol 24, pp. 443-451, 1989.

[5] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, second ed., 1985.

[6] Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.

[7] R. Telichevesky, J. White, and K. Kundert, “Efficient steady-state analysis based on matrix-free krylov-subspace methods,” in *Proceedings of 32rd Design Automation Conference*, June 1995.

[8] _____, “Efficient AC and noise analysis of two-tone RF circuits,” in *Proceedings of the 1996 Design Automation Conference*, June 1996.

# 13

# Using PSP and Pnoise Analyses

## Overview of PSP and Pnoise Analyses

This appendix describes how to calculate small-signal quantities such as noise, noise figure, periodic scattering parameters, and gain in periodically-driven circuits. The appendix explains

- The concepts of periodic S-parameters,
- The concepts of noise correlation parameters
- The various definitions of noise figure and gain

Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) provides four small-signal analyses for circuits with a DC operating point: AC, XF, Noise, and SP. The Spectre RF simulator also provides four small-signal analyses for circuits with a a periodically time-varying operating point: PAC, PXF, Pnoise, and PSP. Because the periodic small-signal analyses linearize the circuit about the time-varying operating point that is obtained using the PSS analysis, they can analyze frequency conversion effects.

- PAC analysis computes the small-signal response at all outputs to the small stimulus of a single group of sources.
- PXF analysis computes the transfer function from every source in the circuit to a single output.
- Pnoise analysis computes noise parameters such as noise figure as well as detailing noise contributions by devices.
- PSP analysis contains some of the capabilities of PAC, PXF, and Pnoise analyses. PSP analysis can compute periodic scattering parameters that describe the small-signal relations between several different ports in a circuit. It can also compute noise parameters, such as noise correlation matrices, equivalent noise sources, and noise figure.

# Periodic S-parameters

## Linear Time-Invariant S-Parameters

Designers of microwave and RF circuits typically characterize the frequency-dependent behavior of linear networks through sets of *scattering* or *S*-parameters. The notion of scattering parameters is rooted in transmission line concepts where the scattering parameter matrix relates the magnitude and phase of incident and reflected waves.

Consider an arbitrary *N*-port linear time-invariant (LTI) network, such as the two-port shown in Figure 13-1 on page 863. Each port is driven by a source of reference impedance $Z_i$, where the index $i$ runs from 1 to *N*. For the remainder of this document we assume a real valued reference impedance, $R_i$, for each port. In terms of the port currents $I_i$ and voltages $V_i$, the *incident* and *reflected* quantities, $a_i$ and $b_i$ respectively, are defined for each port.

The *incident* quantity, $a_i$ as

$$a_i = \frac{v_i}{2\sqrt{R_i}} + \frac{\sqrt{R_i}}{2} I_i$$

The *reflected* quantity, $b_i$ as

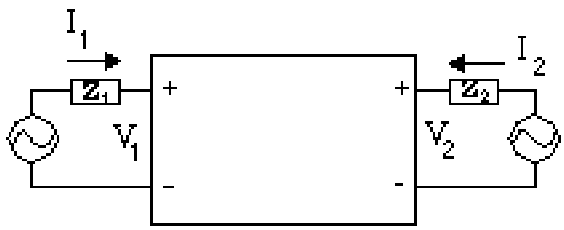
$$b_i = \frac{v_i}{2\sqrt{R_i}} - \frac{\sqrt{R_i}}{2} I_i$$

The frequency-dependent S-parameter matrix relates $a$ and $b$ by

$$b(\omega) = S(\omega)a(\omega)$$

These definitions are possible because for LTI systems, sources at a frequency $\omega$ generate steady-state responses, and therefore outputs, at the same frequency.

**Figure 13-1 Two-Port Linear Network**




## Frequency Translating S-Parameters

The Spectre RF small-signal analyses treat the circuit as linear time-varying (LTV). The primary difference between linear time-varying networks, those that come from circuits with a time-varying operating point, and LTI networks, those that come from circuits with a DC operating point, is that LTV networks shift signals in frequency.

For periodically linear time-varying (PLTV) systems, inputs at a frequency $\omega$ may generate circuit responses, and therefore outputs, at the frequencies $\omega + n\omega_o$, where $\omega_o$ is the fundamental frequency and $n$ is a (signed) integer. We can adopt the S-parameter concept to PLTV systems by considering the inputs and outputs generated at the sidebands of each harmonic to be *virtual ports* of a generalized linear system.

That is, we can define $a_{i,n}$ by

$$a_{i,n}(\omega) = \frac{v_i(\omega + n\omega_o)}{2\sqrt{R_i}} + \frac{\sqrt{R_i}}{2} I_i(\omega + n\omega_o)$$

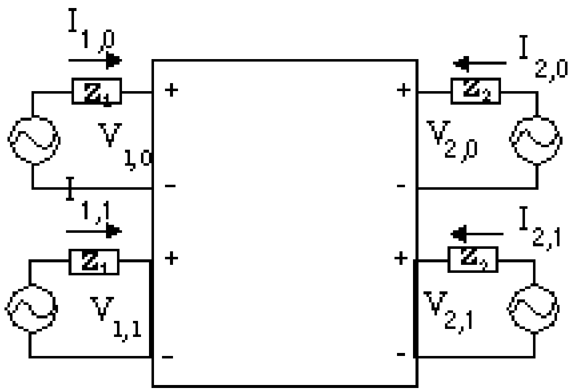
And we can define $b_{i,n}$ by

$$b_{i,n}(\omega) = \frac{v_i(\omega + n\omega_o)}{2\sqrt{R_i}} - \frac{\sqrt{R_i}}{2} I_i(\omega + n\omega_o)$$

where the integer $n$ represents an harmonic index.

For example, as shown in Figure 13-2 on page 864, an ideal mixer may be represented as a four-port.

**Figure 13-2 Ideal Mixer Represented as a Four-Port**




Note that each *virtual port* of a given *physical port* has the same reference impedance. The periodic S-parameter matrix is the $4 \times 4$ matrix $\tilde{s}$ that relates the extended vectors

$$\tilde{b} = \begin{bmatrix} b_{1,0} & b_{1,1} & b_{2,0} & b_{2,1} \end{bmatrix}^T, \tilde{a} = \begin{bmatrix} a_{1,0} & a_{1,1} & a_{2,0} & a_{2,1} \end{bmatrix}^T$$

by

$$\tilde{b}(\omega) = \tilde{S}(\omega)\tilde{a}(\omega)$$

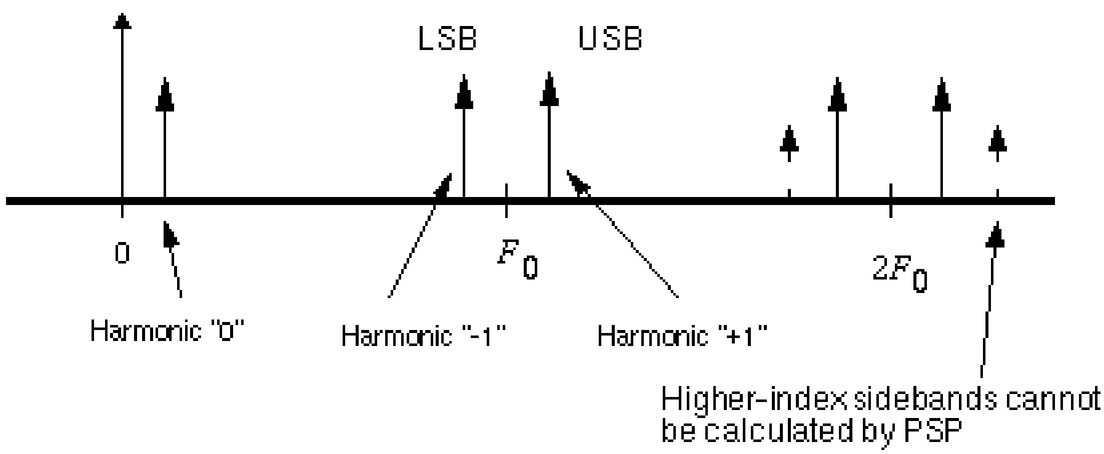
For example, consider an upconverting mixer. You might write $S(2,1|1,0)$ to represent the signal generated at port #2 (typically the output) on the upper sideband of harmonic $+1$ by an incident signal at port #1 (typically the input) at harmonic zero (baseband). $[S(2,1|1,0)\ (\omega)]^2$ would represent the power gain from baseband to RF at the baseband-referenced frequency $\omega$. In the PSP results generated by Spectre, $S(2,1|1,0)$ is accessible as S21~1:0.

Note that because *multiple virtual* ports are used as both inputs and outputs in PSP analysis, PSP analysis must follow an absolute indexing scheme for the small-signal responses. This is different from the relative indexing scheme used in PXF, PAC, and PSP analyses in releases 4.4.5 and earlier. See “Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise Analyses” on page 898 for a discussion of the differences.

### Upper and Lower Sidebands

Each harmonic may have an input, or response, at both the upper and lower sidebands of each harmonic. In PSP analysis, the upper sideband is denoted by a positive integer, and the lower sideband as a negative integer. See Figure 13-3 on page 865.

**Figure 13-3 Harmonics, Sidebands, and Virtual Ports**



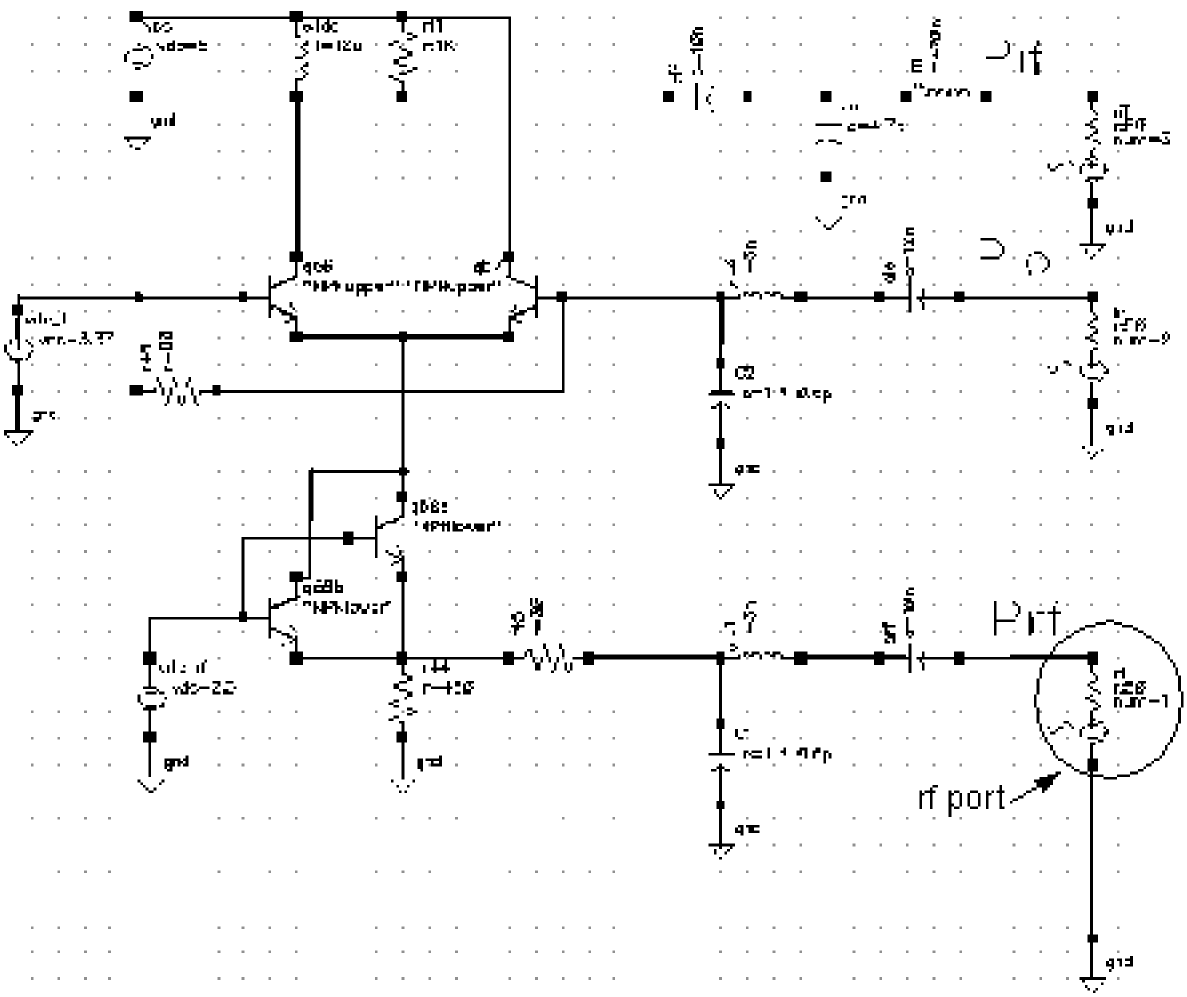

For a small-signal frequency $\Delta F$, the upper sideband of the $k^{th}$ harmonic is at the frequency $/k/\ F_0+\Delta F$ and the lower sideband is at $|k|\ F_0 - \Delta F$.

In general, an input (perhaps at baseband at the frequency $\Delta F$ in an upconversion mixer) can generate responses at all the frequencies $|k\ F_0+l\Delta F|$, for $k$ and $l$ integers. However, because small-signal analyses are linear, they can only calculate the signals at the first sideband. Thus we only need notation for the $l$ =1 terms in PSP analysis.

## PSP Analysis Example

Consider performing a PSP analysis on the *NE600* mixer schematic from the *rfExamples* library. The schematic is shown in Figure 13-4 on page 866.

**Figure 13-4 NE600 Mixer**




Suppose the RF input signal is at 900 MHz, the LO at 1 GHz, and the IF at 100 MHz. Before the PSP analysis is performed, a PSS analysis must be run. For small-signal analysis, in many cases it would be sufficient to treat the RF input as small-signal (for example, by setting the *source type* to *DC*). However, sometimes it is important to analyze additional noise folding terms induced by the RF input, so in this example we assume the RF source is a large signal (e.g., *source type = sine*). The PSS fundamental need to be set to 100 MHz.

Now for the sake of demonstration suppose we wish to perform the small-signal analysis from 20 MHz below the RF center frequency to 30 MHz above. To set up the analysis, we first select a *frequency sweep*. We select *sweeptype=relative*, with a range of -20 MHz to 30 MHz. This accounts for inputs on the RF port in the range of 880 MHz to 930 MHz. Noise parameters such as *noise figure* are computed in a 50 MHz band around the frequency specified by the output harmonic.

Next we select the ports and harmonics. The *input* and *output ports* are selected from the schematic as always. Selecting the harmonics is somewhat trickier.

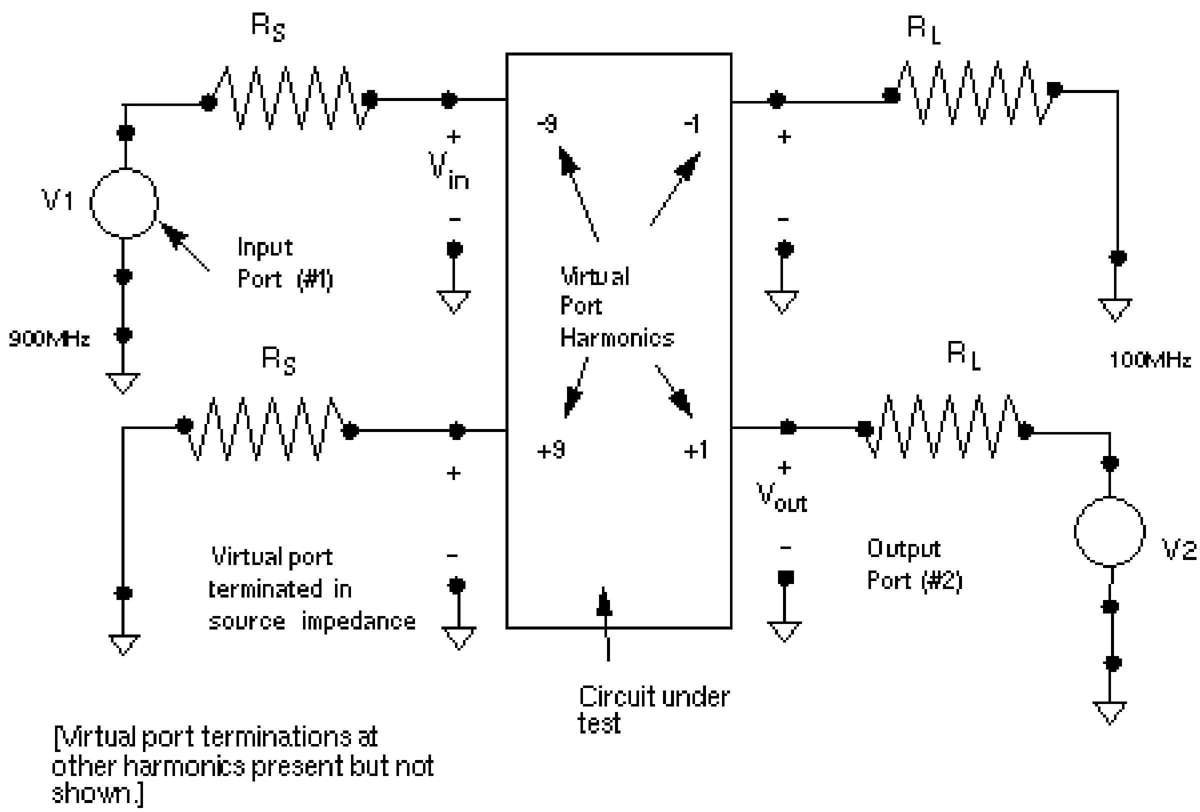
With the 1 GHz LO, small-signal inputs at around 900 MHz, or harmonics +/–9 of the PSS fundamental, appear at around 100 MHz, or harmonics –/+1 of the fundamental. We may select harmonic 1 as the output harmonic and harmonic -9 as the input harmonic because a single complex-exponential input

$$e^{i\omega_s t}$$

input on the lower side of 900 MHz (harmonic -9) appears as the upper sideband of harmonic 1 (around 100 MHz). -9 and 1 are separated by 10 fundamental periods, which corresponds to the LO frequency of 1 GHz. Figure 13-5 on page 868 conceptually illustrates this setup.

After the simulation is complete, a limited set of data is accessible through direct plot, and the full data set is accessible with the results browser. The default is to output all quantities versus the input frequency, which in this case would be a sweep from -920 MHz to -870 MHz, because the input harmonic is -9 and the sweep ran from -20 Mhz to 30 MHz. The *freqaxis* parameter on the options form may be used to change the axes that are output by Spectre RF. In 4.4.5 the *freqaxis* parameter and *frequency sweep* specifications are solely responsible for the data's axis generation. Setting *freqaxis=out* in this example would produce an axis running from 80 Mhz to 130 Mhz.

**Figure 13-5 Measuring Periodic S-Parameters**




An additional port/harmonic pair can be included in the PSP analysis by using the *auxiliary port* fields.

If it is desirable to include more than three harmonics in the PSP analysis, they can be added to the list in the form below the *input.*/*output*/*auxiliary*. For example, to examine additional images in the PSP analysis, +9 and -1 could be additional harmonics. S21~9:-1 would represent the transducer gain from RF-USB to IF-LSB. Note that S21:9:1 is likely to be small, because this term represents a frequency shift of 800 MHz. There are no elements in the circuit that vary at 800 MHz, so significant 800 MHz frequency translations are not present.

## Noise and Noise Parameters

### Calculating Noise in Linear Time-Invariant (DC Bias) Circuits

The standard Noise analysis has two parts.

First, the circuit is analyzed without the noise sources present in order to find a DC operating point.

Next, the circuit is linearized around that operating point and the noise sources are turned on.

The linearized circuit is used to compute a set of transfer functions that represent the gain from each noise source to the node pair or probe that is identified as the output for the Noise analysis. All noise generators present in the circuit are automatically included in the Noise analysis, as are the noise sources of the source and load.

### Calculating Noise in Time-Varying (Periodic Bias) Circuits

Noise analysis in RF circuits, where the circuit operating point is time-varying, is computationally involved, but conceptually similar to Noise analysis for circuits with a DC bias.

Find the circuit operating point using the PSS analysis.

Linearize the circuit around that operating point.

Use the PXF analysis, based on the linearized circuit, to compute a set of transfer functions that relate the noise sources to the noise at the circuit output.

The treatment of the sources and the transfer functions is more complicated for RF circuits because of the time-varying operating point.

For noise sources that are bias dependent, such as shot noise sources, the time-varying operating point acts to modulate the noise sources. Active elements with a time-varying bias point can convert noise from one frequency to another, a process known as *noise folding*, regardless of the origin of the noise.

Because of these effects, noise generated in RF circuits usually has *cyclostationary* properties. Cyclostationary random processes are processes whose statistical properties are periodically time-varying. In the frequency domain, a simple way to think of cyclostationarity is as frequency-correlation. For example, noise at the input of a mixer appears on the output, but shifted in frequency. Thus the noise at the mixer output at a given frequency is correlated with noise at the mixer input at a frequency separated by the frequency of the local oscillator. In contrast, the noise generated by a circuit with a DC bias point is usually modeled as being uncorrelated with noise at any other frequency. Spectre RF correctly accounts for cyclostationary statistics when calculating noise, noise figure, noise correlation parameters, and equivalent noise sources.

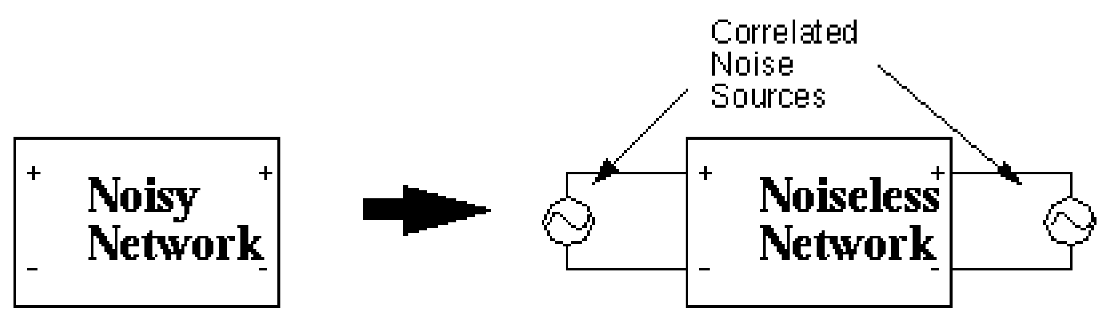
### The maxsideband Parameter

All the noise computations in PSP involve noise folding effects. The *maxsideband* parameter specifies the maximum sideband included for summing noise contributions either up-converted or down-converted to the output at the frequency of interest. The contribution of the noise source to the output is modulated by the periodic transfer function. Modulation with a periodic transfer function is convolution with the discrete spectrum of the transfer function. *Maxsideband* specifies the number of sidebands to be involved in this calculation.

## Noise Correlation Matrices and Equivalent Noise Sources

Noise correlation matrices represent a decomposition of a linear circuit into a noiseless linear network and correlated noise sources. For example in Figure 13-6 on page 870, a noisy linear network is decomposed into a noiseless network and equivalent noise current sources, one for each circuit port, that represent the effect of all the noise generators internal to the original network. Note that in general the equivalent noise sources are correlated. The equivalent sources can be completely described by a (frequency-dependent) noise correlation matrix. Note that after a noise correlation matrix along with an admittance, impedance, or S-parameter matrix are known, then the properties of the noisy linear network as seen from the I/O ports is completely specified. All circuit input/output properties—gain, noise figure, etc.—can be calculated from the S-parameter and noise correlation matrices.

**Figure 13-6 Decomposing a Network Into Noiseless and Noisy Elements.**



The SP analysis computes noise correlation parameters in the admittance representation, i.e., the sources in Figure 13-6 on page 870 are current sources. If we let $I_N$ denote the vector of equivalent noise currents as

$$I_N = \begin{bmatrix} I_{N1} \\ I_{N2} \end{bmatrix}$$

Then the noise correlation matrixes $C_y11$ and $C_y12$ are defined as

$$C_Y1x = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{I_N I_N^H\right\}$$

And the noise correlation matrixes $C_y21$ and $C_y22$ are defined as

$$C_Y2x = \left[\frac{1}{4k \times T0\_2 \times df}\right] \times E\left\{I_N I_N^H\right\}$$

where

$k$ is Boltzmann's constant

$T0\_1$ is the noise temperature of the input port

$T0\_2$ is the noise temperature of the output port

$df$ is noise bandwidth

superscript $H$ ($I^H$) denotes the Hermitian transpose

$E\{°\}$ denotes statistical expectation.

For example, for a two-port,

$$C_Y = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{\begin{bmatrix} I_{N1}\overline{I_{N1}} & I_{N1}\overline{I_{N2}} \\ I_{N2}\overline{I_{N1}} & I_{N2}\overline{I_{N2}} \end{bmatrix}\right\}$$

where the overbar signifies complex-conjugate. Note that this matrix must have real diagonal elements, because the diagonals represent a total noise power, but the off-diagonals, which represent the correlations, may be complex.

In the periodic case, we define the periodic admittance noise correlation matrix in precisely the same way. The situation is slightly more complicated because the *virtual ports* may lie at different frequencies in RF systems.

Letting

$$\tilde{I}_N$$

denote the extended vectors of noise currents at each of the virtual ports (each virtual port consisting of a physical port. harmonic pair), for example

$$\tilde{I_N} = \left[I_{1,\,0}\ I_{1,\,1}\ I_{2,\,0}\ I_{2,\,1}\right]^T$$

where the first index indexes the physical port, and the second index specifies the harmonics.

The periodic admittance noise correlation matrix

$$\tilde{C}_Y$$

is defined as

$$\tilde{C}_Y = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{\tilde{I}_N \tilde{I}_N^H\right\}$$

When you specify the PSP analysis option *donoise=yes*, then the complex noise correlation matrix of order (#active ports X #active sidebands) is computed.

## Two-Port Noise Parameters

As an alternative to the noise correlation matrices that define the equivalent sources, Spectre RF calculates the values of the equivalent noise parameters $F_{min}$, $R_n$, $G_{opt}$, $B_{opt,}$ and $NF_{min}$. These are calculated from the two-port admittance ($Y_{11}$, $Y_{12}$, $Y_{21}$, $Y_{22}$) and noise correlation admittance ($CY_{11}$, $CY_{12}$, $CY_{21}$, $CY_{22}$) parameters. These calculations are done as part of the SP and PSP analysis. In terms of the admittance and noise correlation matrices, the parameters are

```
Fmin = 1 + 2 CY22/|Y21| 2 (Gopt + Re{Y11 - Y21(CY12/CY22)})
Gopt = sqrt[|Y21|2 (CY11/CY22) - |Y21|2(|CY12|2/CY222) + (Re{Y11 - Y21(C12/C22)})2]
Bopt = -Im{Y11 - Y21(CY12/CY22)}
Rn = CY22/|Y21|2
Yopt = Gopt + jBopt
NFmin = 10log(Fmin)
```

$Y_{opt}$ is the source admittance that gives the minimal noise factor $F_{min}$ (corresponding to the source reflection coefficient Gamma opt, or $\Gamma_{\text{opt}}$) and $R_n$ is the equivalent noise resistance. Finally, $NF_{\min}$, the minimum noise figure, is $F_{\min}$, the minimum noise factor, in dB.

For more information, see Janusz A. Dobrowolski, *Introduction to Computer Methods for Microwave Circuit Analysis and Design, Artech House, Boston, 1991*, page 193.

## Noise Circles

Noise factor is a function of the source admittance $Y_s = G_s + jB_s$. For a given $Y_s$ the noise factor is

$$F = F_{min} + \frac{R_n}{G_s}(Y_s - Y_{opt})^2$$

Varying $Y_s$ traces out circles of constant noise factor $F$. In the 4.4.5 release, noise circles are only available in direct plot for the SP analysis.

# Noise Figure

## Performing Noise Figure Computations

The Noise, SP, Pnoise and PSP analyses all provide the ability to calculate various types of noise figure. Noise and SP analyses are used for circuits with a DC bias. Pnoise and PSP analyses are used for circuits with a periodic bias. The *generic* way Spectre RF calculates the noise figure is by first computing the noise factor $F$,

$$F = \frac{totalOutputNoise - outputNoiseFromLoad}{outputNoiseFromSource}$$

where the noise is specified in units of power (e.g., $V^2/Hz$). Noise figure is then $NF=10log_{10}F$.

The various definitions of noise figure differ in the following ways

- How the contributions to the total output noise are calculated (e.g. Pnoise analysis has noise folding effects, Noise analysis does not)
- What noise is considered to be due to the load
- What noise is considered to be due to the source

All the analyses share some common rules that must be followed to obtain correct answers

- You must specify a *port* (not a *vsource*, *isource*, or *ahdl* source) as an circuit *input* or *iprobe*.
- You must specify the *load* as an *output* or *oprobe*. The *load* may be a *resistor* or a *port*. Note that all noise from the source is included in the denominator of the noise factor fraction, including excess noise, so do not specify excess noise on the input port. (Excess noise is specified with the `noisefile` or `noisevec` option.)

In rare cases there may be no load, in which case you can specify the output using a pair of nodes. Be warned, however, that if there is a load in the circuit, and a pair of nodes is specified as an output, then you obtain different results than if a load was specified as output. This is because the load contributes some noise to the total output noise that must be subtracted out before using the equation above to compute the noise figure. If only a pair of nodes is used, Spectre RF has no way of determining which of the elements in the circuit is the load (there could be multiple resistors connected to the output nodes, for example) and so cannot determine the amount of output noise due to the load.

Note that these requirements are automatically enforced in SP and PSP analyses, because the input and output sources are always ports that must be identified to the analyses.

## Noise Figure From Noise and SP Analyses

The Noise and SP analyses perform noise figure computations on circuits with a DC operating point. The above prescription for noise figure computation is straightforward: the output noise, contribution from source, and contribution from load must be computed.

Mathematically, if we let $X_L$ denote the transfer function from output load to output, (at the same frequency) and $X_S$ denote the transfer function from input source to output then

$$F(f) = \frac{N_o(f) - |X_L|^2 n_L(f)}{|X_s|^2 n_s(f)}$$

In this equation, $X_S$ plays a role similar to transducer gain in traditional treatments of noise figure.

## Pnoise (SSB) Noise Figure

Because noise in an RF circuit can originate at many different frequencies, the denominator in the noise factor computation is in a sense ill-defined. See the book *Microwave Mixers* by S. Maas for a discussion of various possible noise figure definitions.

There are three common noise factor definitions in use. The Spectre RF Pnoise and PSP analyses compute as $F$ or $NF$ what is referred to as conventional single-sideband (SSB) noise figure. The conventional SSB noise figure is typically useful for heterodyne receivers. To compute the conventional SSB noise figure, a reference sideband must be specified that identifies the input noise used in the denominator of the noise factor computation. Only the contribution of the noise from the input source, generated at the frequency specified by the reference sideband, is included in the noise factor denominator.

In the Pnoise context, the numerator contains the total output noise, except the noise from the output load that was generated at the output frequency. Note in particular that noise from the input source folded from all sidebands, and noise from the output load folded from the non-zero sidebands, is included in the noise factor numerator. Mathematically, Pnoise computes conventional single-sideband noise factor as

$$F_{ssb}(f_{out}) = \frac{N_o(f_{out}) - \left|X_L^{(0)}\right|^2 n_L(f_{out})}{\left|X_S^{(K_{ref})}\right|^2 n_S(f_{out} + K_{ref} f_0)}$$

where

- $f_{out}$ is the output frequency swept by Pnoise
- $X_L^{(0)}$ is the transfer function associated with the zero sideband, from load to output

- $X_s^{(K_{ref})}$ is the transfer function associated with the reference sideband, from source to the output
- $n_L(f_{out})$ is noise generated by load at the output frequency
- $n_S\,(f_{out}+K_{ref}f_0)$ is the noise generated by the source at the input frequency.

In the PSP analysis context, the noise factor denominator includes only noise from the input harmonic. The numerator contains all output noise, except noise from the output load at the output harmonic. (Refer to "Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise Analyses" on page 898 for a discussion of differences in frequency indexing in PXF and PSP analyses.)

To be mathematically precise in what follows, let

- $X_L^{(K)}$ denote the transfer function from output load, $k^{th}$ harmonic to output
- $X_S^{(K)}$ denote the transfer function from input source, $k^{th}$ harmonic to output
- $K_o$ denote the output harmonic
- $K_i$ denote the input harmonic

The conventional single-sideband noise factor is computed by PSP as

$$F_{ssb}(f_{out}) = \frac{N_o(K_o f_0 + f) - \left|X_L^{(K_o)}\right|^2 n_L(K_o f_0 + f)}{\left|X_S^{(K_i)}\right|^2 n_S(K_i f_0 + f)}$$

where

- $f$ is the PSP relative sweep frequency
- $K_o f_0 + f$ is the output frequency
- $K_i f_0 + f$ is the input frequency
- $n_L(K_o f_0 + f)$ is the noise generated by the load at the output frequency
- $n_S(K_i f_0 + f)$ is the noise generated by the source at the input frequency

The conventional SSB noise figures computed by PSP and Pnoise analyses are the same and are computed in the same way internally, it is only the notation above that is different.

## DSB Noise Figure

In some applications, such as direct conversion receivers, it is more appropriate to compute what is called double-sideband (DSB) noise figure. Double-sideband noise factor is obtained by ratioing the same numerator as for SSB to the noise from the input at the input harmonic as well as its primary image.

Double-sideband noise figure is usually 3 dB below single-sideband noise figure, except when the input signal band is converted to baseband output with DSB noise figure equal to SSB noise figure.

In double-sideband computation, the input signal band is assumed to be either down-converted or up-converted to the output signal band. Hence you should associate the appropriate harmonic number to the input and the output port in the *portharmsvec* parameter. For a mixer, their difference is the LO band. The image sideband is the sideband on the other side of the LO band. The distance from the image band to the LO band is the same as that from the LO band to the input band. For example, if the fundamental frequency is100 MHz, the LO frequency is 1 GHz, and the RF input frequency is 900 MHz, then the LO band is 10, the RF input band is 9 and the image band is 11.

Double-sideband noise figure is computed by as

$$F_{dsb}(f_{out}) = \frac{N_o(K_o f_0 + f) - \left|X_L^{(K_o)}\right|^2 n_L(K_o f_0 + f)}{\left|X_S^{(K_i)}\right|^2 n_S(K_i f_0 + f) + \left|X_S^{(K_{\mathrm{image}})}\right|^2 n_S(K_{\mathrm{image}} f_0 + f)}$$

where

$n_S(K_{\mathrm{image}} f_0 + f)$ is the noise generated by source at the image input frequency obtained according to the above description.

All other quantities are the same as those in $F_{ssb}$. Note that both the PSP and Pnoise analyses compute double-sideband noise figure.

## IEEE Noise Figure

Sometimes it is desirable to define noise figure quantities where we assume that the noise from input images that are potentially filtered is not present. The IEEE definition of noise figure in mixers differs from the conventional definition in that it does not include the contribution to the output noise from the image sideband in the numerator of the noise factor.

Spectre RF eliminates all image harmonics/sidebands from the output noise in the noise factor numerator when computing *Fieee*. Using the above notation, *Fieee* is

$$F_{ieee} = \frac{N_o(K_o f_0 + f) - \left|X_L{}^{(K_o)}\right|^2 n_L(K_o f_0 + f) - \sum_{K \neq K_i} \left|X_S^{(K)}\right|^2 n_S(K f_0 + f)}{\left|X_S{}^{(K_i)}\right|^2 n_S(K_i\, f_0 + f)}$$

Note that both the PSP and Pnoise analyses compute *Fieee*.

Figure 13-7 on page 879, Figure 13-8 on page 880, and Figure 13-9 on page 881 summarize the treatment of the input source and output load for the various noise figure definitions.

**Figure 13-7 Input Source Treatment for Denominator in Noise Factor Computations**

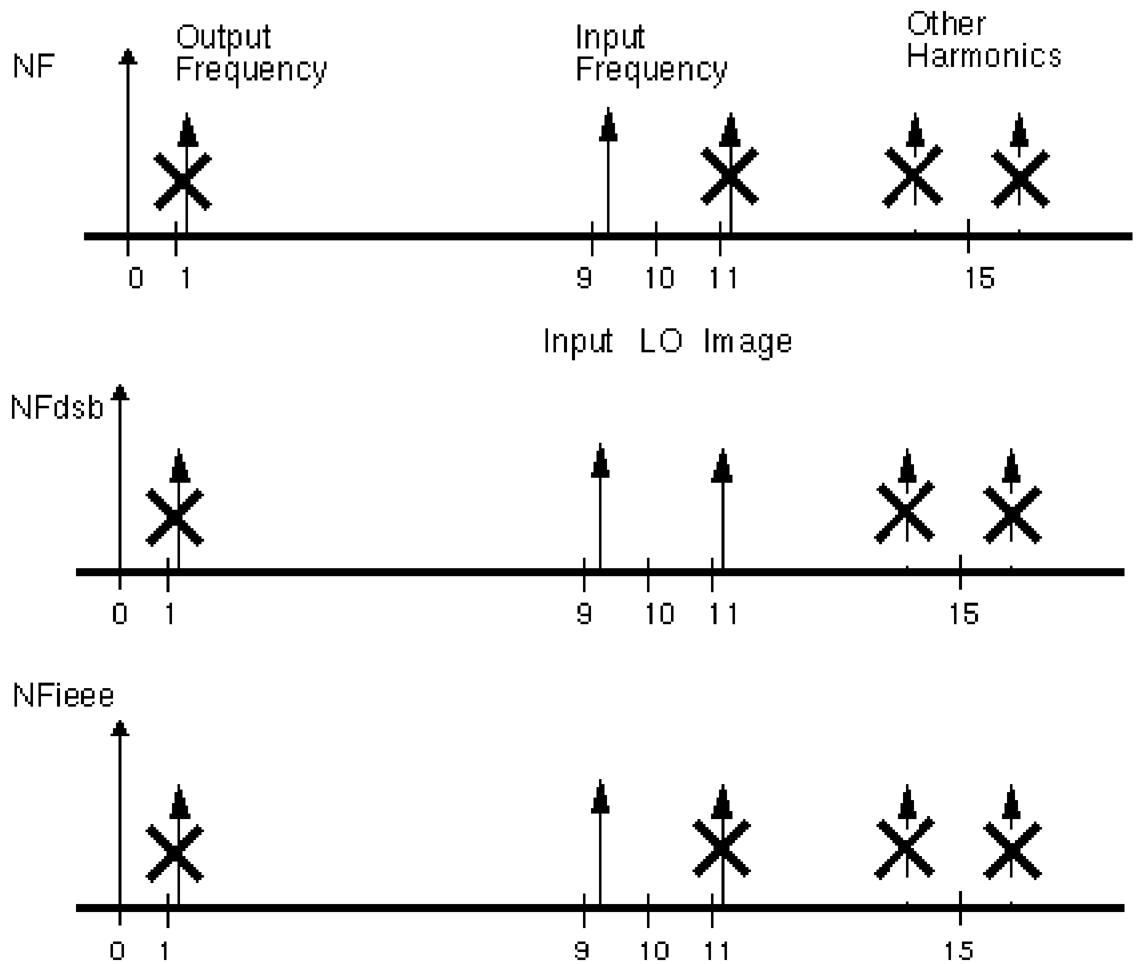


**Figure 13-8 Output Load Treatment for Numerator in Noise Factor and Noise Figure Computations**

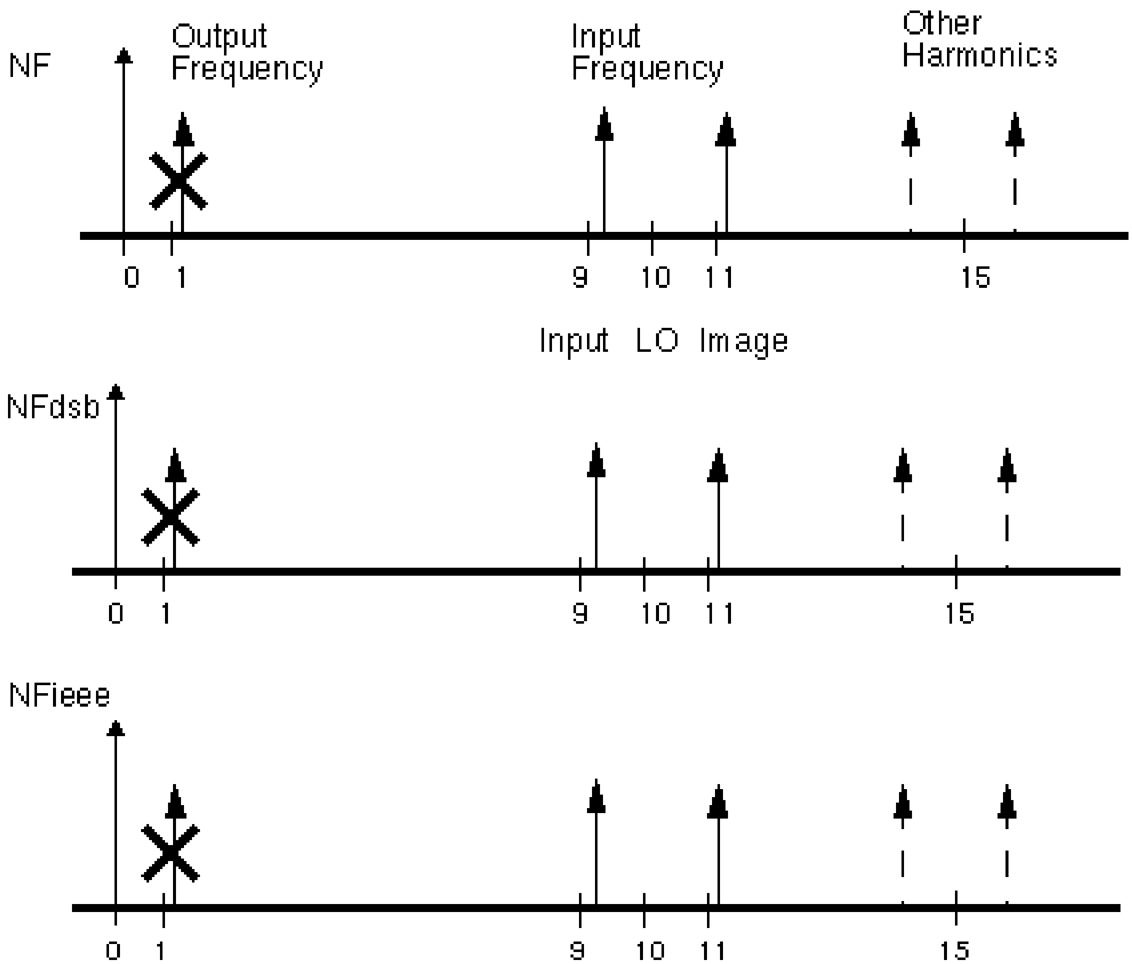
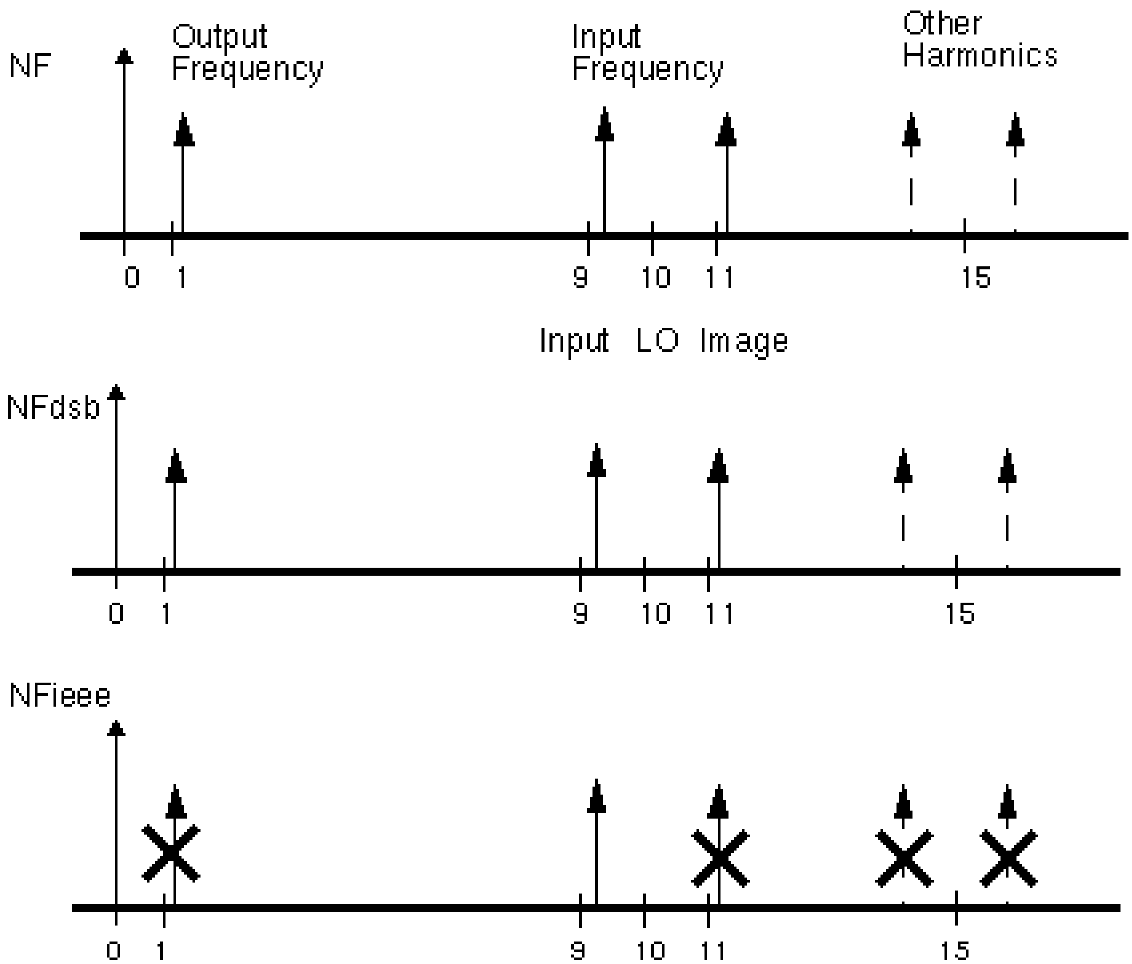


**Figure 13-9  Input Source Treatment for Numerator in Noise Factor Computations**




# Noise Computation Example

Now consider performing noise computations as part of the PSP analysis of the *ne600* mixer example presented earlier (See [“PSP Analysis Example”](#) on page 865).

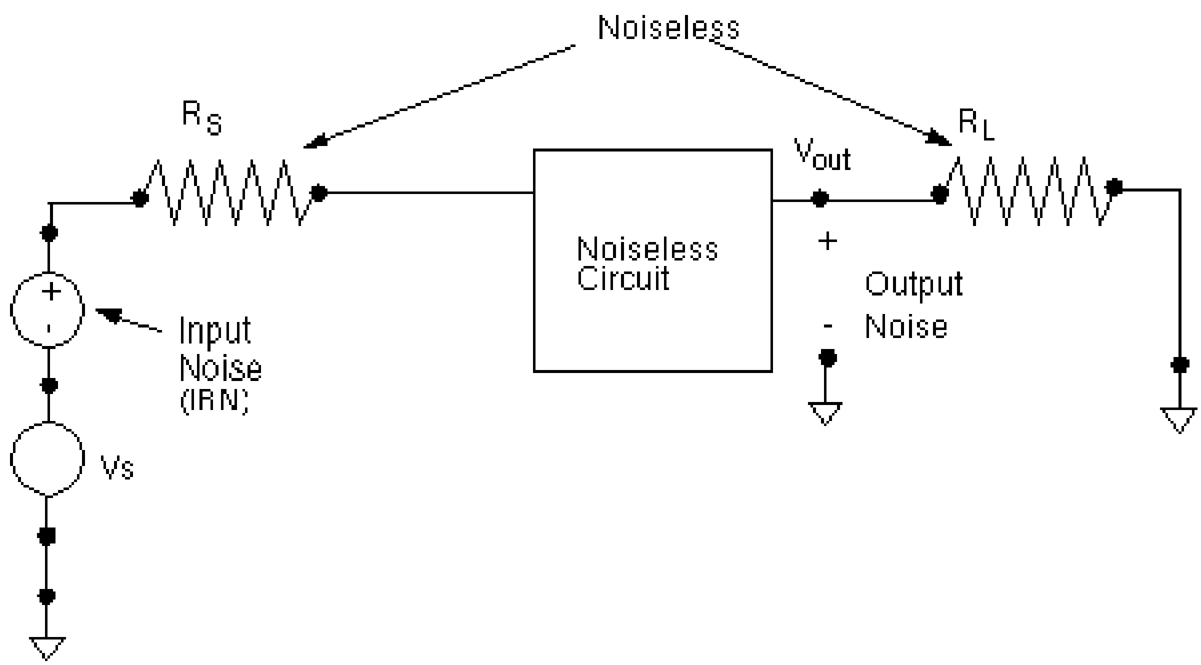
To compute noise parameters, set *donoise=yes* on the PSP analysis form and select a reasonable number for *maxsideband*. In this case, *maxsideband* should certainly be greater than 10 and probably greater than 50 or so. Setting *maxsideband=50* would account for noise folding from up to 5 GHz in frequency. The simulation can now be run as before. In addition to computing the periodic S-parameters, the periodic noise correlation matrices are also computed, as are noise figure and the equivalent noise parameters.

Recall that the output axes for PSP analysis are set by the `freqaxis` option and are shared by all quantities calculated by PSP analysis. This can seem odd in the case of noise figure, where the output noise is actually analyzed for the frequency range of 80 Mhz to 130 MHz. This is because the output harmonic is 1, representing a center frequency of 100 Mhz for the relative sweep, and the sweep ran from -20 MHz to 30 MHz. Setting `freqaxis=out` in this example would produce an axis running from 80 Mhz to 130 Mhz.

## Input Referred Noise

For a given output noise spectrum, the equivalent input noise, *input referred noise* or IRN, is the noise that, if it were generated by the circuit element specified as input, would produce the same output noise. This assumes that the circuit loading conditions, etc., are unchanged. Note that the equivalent input noise includes noise from the source and load if they are noisy in the original circuit. For example, consider a circuit with a voltage source as input, as in Figure 13-10 on page 883. If a voltage source inserted in series with the input source generates the input referred noise, and the rest of the circuit is noiseless, the noise observed at the output is the same as in the original noisy circuit.

**Figure 13-10 Equivalent Input Noise**




## Using Input Referred Noise

Input referred noise, or IRN, gives a direct estimate of how much the noise in a circuit corrupts signals passing through, because the amplitude of the noise can be directly compared to the amplitude of signals on the input. In principle, it can also be used to build macromodels of the circuit. If a source of the same type (vsource, isource, or port) has the *`noisefile`* argument set to a file to which the input referred noise has been written, and all the noise elements in the circuit turned off (perhaps as may happen when replacing the circuit with an S-parameter macromodel) then the noise that appears as the circuit output is the output noise of the original circuit. Note that Spectre RF *`noisefile`* arguments are usually given in terms of power, such as $V^2/Hz$, whereas the simulator usually outputs the noise in units of signal amplitude (such as $V/sqrt(Hz)$).

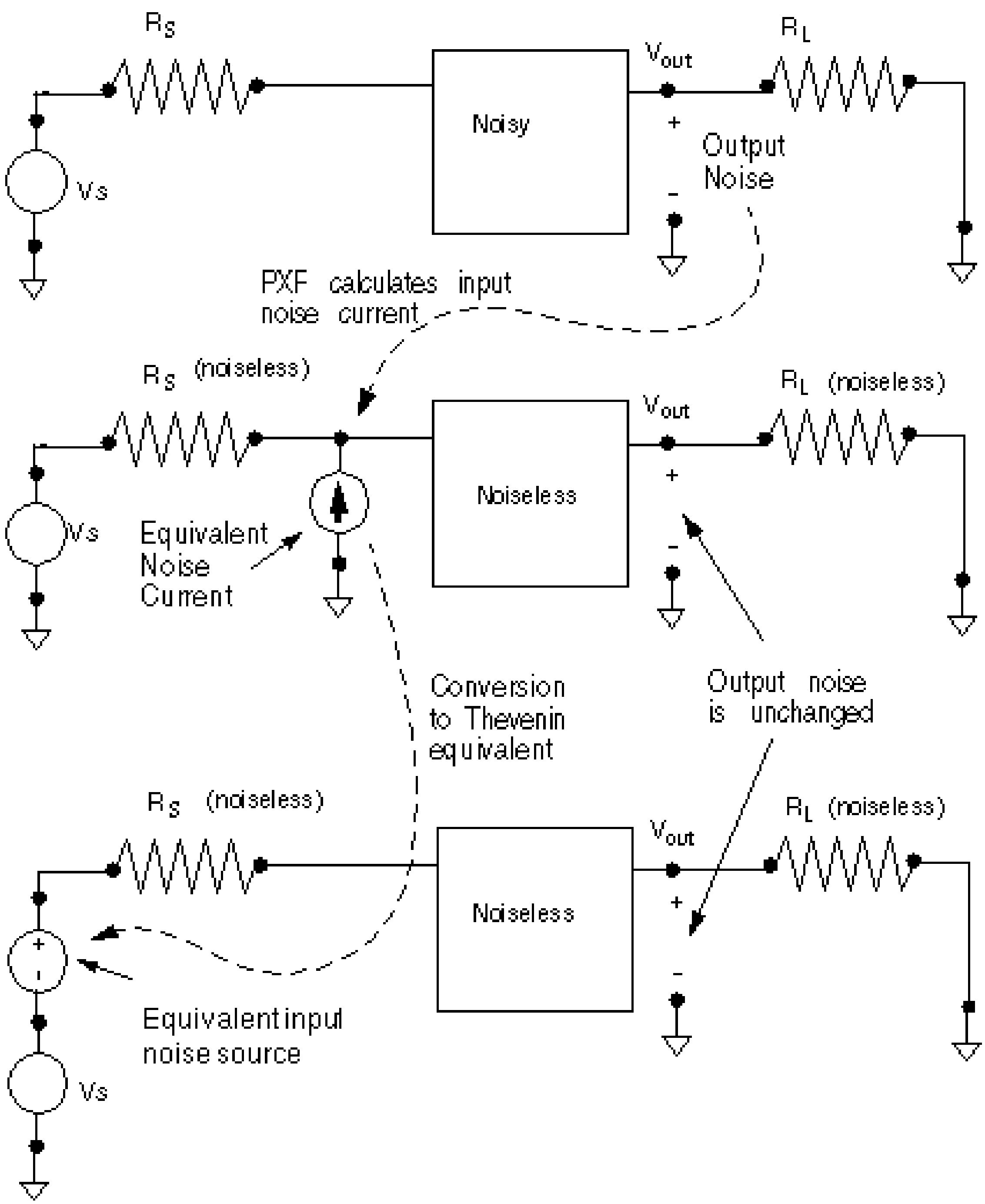
It is also possible to calculate noise figure from input referred noise, however, it is essential to be sure that any noise contribution from the load, etc., has first been properly treated. For this reason, it is recommended to use the Spectre RF built-in noise figure calculation, because the contribution of load noise, noise from other images, etc., is automatically accounted for. (See "The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong." on page 894 for more information.)

## How IRN is Calculated

Spectre RF calculates input referred noise from the results of the Noise or Pnoise analysis and an XF or PXF analysis. The XF or PXF analysis needed is fortunately the same analysis needed to compute the noise.

As an example of an input noise calculation, consider Figure 13-11 on page 885, which shows the calculation of the equivalent input noise when a port component drives the circuit. First the total output noise is calculated. Note that this noise generally includes contributions from the circuit load. Next the results of the PXF analysis are used to express this noise as an equivalent current source attached to the circuit input. The amplitude of the current source is the amplitude of the output noise divided by the transfer function from an imaginary current source connected to the circuit input to the output node. Finally, the equivalent current source is converted into the equivalent input voltage noise by converting to Thevenin form.

**Figure 13-11 IRN Calculation**



## Relation to Gain

It is common to think of the input noise as the output noise divided by the gain,

$$IRN = \frac{OutputNoise}{G}$$

where the output noise and input noise are measured in volts/sqrt(Hz). This is a true statement if the proper gain is used. The gain reported by Pnoise analysis is in fact the ratio between the output noise and the equivalent input noise as computed in the previous section. However, the gain reported by Pnoise analysis is not necessarily the gain useful for any other purposes (see “Gain Calculations in Pnoise” on page 891).

## Referring Noise to Ports

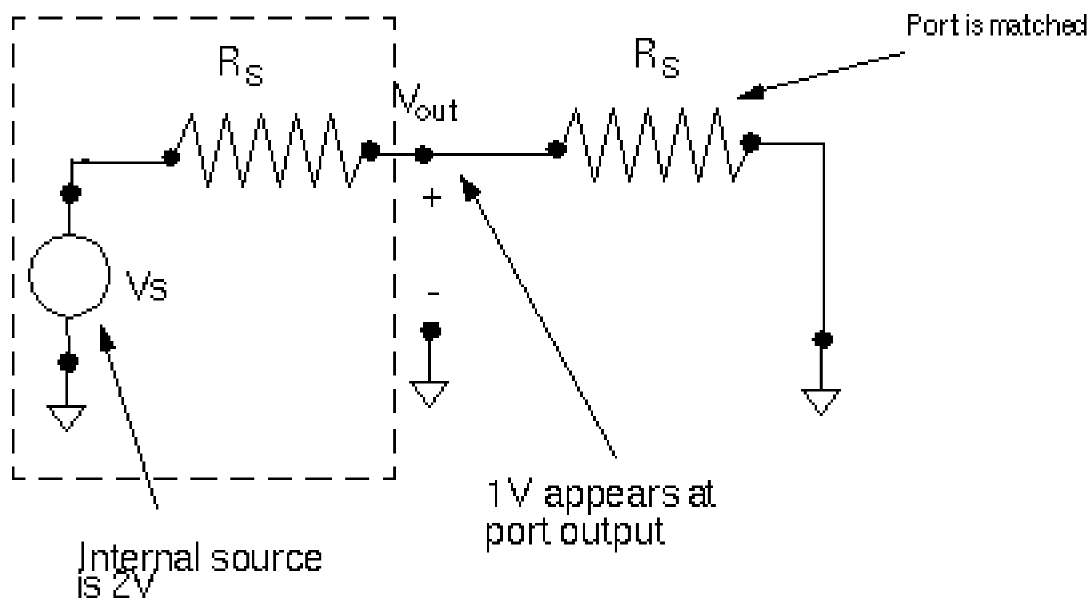
A port component in Spectre RF is a voltage source combined with a resistive impedance. Ports exist to enable easy generation of test fixtures, accurate noise figure calculations, and S-parameter calculations. Noise and S-parameter calculations in Spectre RF can (usually) only be properly performed when ports are used to drive the input and output. While the port is electrically equivalent to a source + resistor, the excitations are specified in a somewhat different way.

The port component is designed so that when a 1 V source is specified, 1 V appears at the port output when the circuit impedance is matched to the port reference impedance. The port output, however, is not the internal node that connects the voltage source and resistor, but rather the exterior resistor terminal that actually connects to the circuit under test. Thus, as shown in Figure 13-12 on page 887, the voltage on the internal voltage source must be twice the voltage specified to appear at the port output.

One implication of this convention is that when a port is specified as an input in Pnoise analysis it is the noise referred to the port that is computed, not the noise referred to the internal voltage source. In other words, the input equivalent noise reported is the noise that would be fed back to spectre as a *noisefile* argument on the port to produce the same output noise. The noise referred to the port has amplitude half of the noise referred to the input voltage source.

Likewise, the gain reported by Spectre RF is the gain that would produce the equivalent input noise, referred to the port. It is half the gain from the input voltage source.

**Figure 13-12  Port Driving Matched Circuitry**




# Gain Calculations

## Definitions of Gain

To understand gain calculations in Spectre RF it is necessary to specify which of several possible gains are being calculated.

**Figure 13-13 Equivalent Input Noise**

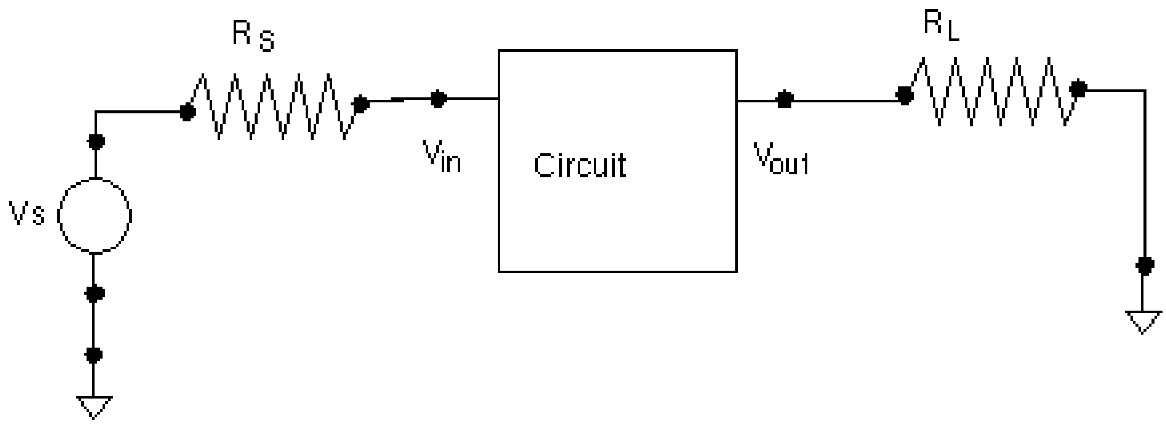



Figure 13-13 on page 888 shows a circuit configured for a gain calculation. Source and load impedances are present. These loads might represent other circuit components or *testbench* circuitry. The circuit is driven with the source *Vs* and the output voltage *Vout* measured. One possible gain is the gain *Gs*.

$$Gs = \frac{Vout}{Vs}$$

from the voltage source to the circuit output.

However, Gs is not the actual gain of the circuit, it is the gain of the circuit and the testbench put together. For high frequency circuits, this gain in fact cannot be measured on the bench, only in the artificial world of a circuit simulator. For these reasons, the proper way to calculate gain is the use a port component, which allows the user to specify the impedance of the driving circuitry. The Spectre RF PSP analysis computes the gain,

$$G = \frac{Vout}{Vin}$$

which is the voltage gain from the circuit input to the circuit output.

G is calculated for historical reasons, and is not necessarily a useful number, but after an SP or PSP analysis is performed, several other gains can be defined and computed. Varying source and load impedance can also be considered. In the 4.4.5 release, the SP direct plot form displays various gains. The PSP direct plot form does not automatically compute the various gains but they can be computed from the basic definitions using the calculator.

Some gains of interest in two-port circuits are:

- $G_A$ (Available Gain)
- $G_P$ (Power Gain)
- $G_T$ (Transducer Gain)
- $G_{umx}$ (Maximum Unilateral Transducer Power Gain)
- $G_{max}$ (Maximum Available Gain)

**GA (Available Gain)**, the power gain obtained by optimally (conjugately) matching the output of the network.

$$G_A = \frac{|S_{21}|^2(1-|\Gamma_S|^2)}{|1-S_{11}\Gamma_S|^2(1-|\Gamma_2|^2)}$$

where

$$\Gamma_2 = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1-S_{11}\Gamma_S}$$

and $\Gamma_S$ is the source reflection coefficient,

$$\Gamma_S = \frac{Z_S - Z_{S,\,ref}}{Z_S + Z_{S,\,ref}}$$

with $Z_S$ is the source impedance and $Z_{S,\,ref}$ is the reference impedance for the input port.

**GP (Power Gain)**, the power gain obtained by optimally (conjugately) matching the input of the network.

$$G_P = \frac{|S_{21}|^2(1-|\Gamma_L|^2)}{|1-S_{22}\Gamma_L|^2(1-|\Gamma_1|^2)}$$

where

$$\Gamma_1 = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1 - S_{22}\Gamma_L}$$

and $\Gamma_L$ is the load reflection coefficient.

**GT (Transducer Gain)**, the ratio of the power dissipated in the load to the power available from the source,

$$G_T = \frac{(1 - |\Gamma_S|^2)|S_{21}|^2(1 - |\Gamma_L|^2)}{|(1 - S_{11}\Gamma_S)(1 - S_{22}\Gamma_L) - S_{12}S_{21}\Gamma_S\Gamma_L|^2}$$

**Gumx (Maximum Unilateral Transducer Power Gain)**

$$G_{umx} = \frac{|S_{21}|^2}{(1 - |S11|^2)(1 - |S22|^2)}$$

**Gmax (Maximum Available Gain),** the transducer power gain when there exists a simultaneous conjugate match at both ports.

For $Kf > 1$

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}[Kf - SQRT((Kf)^2 - 1)]$$

For $Kf < 1$

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}$$

Where $K_f$ is the stability factor, $K_f$

$$K_f = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |D|^2}{2|S_{22}||S_{12}|}$$

and

$$D = S_{11}S_{22} - S_{21}S_{12}$$

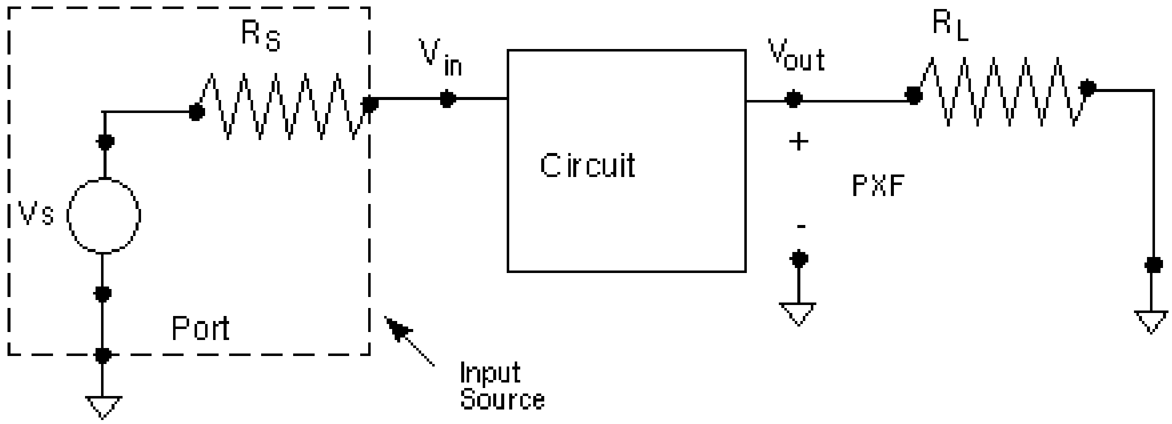
## Gain Calculations in Pnoise

When ports are used to drive the circuit input and a Pnoise analysis is performed, Pnoise analysis reports a quantity called `gain` that can be misleading. Consider the circuit in Figure 13-14 on page 892 where a port component is used as the input to the circuit, and specified as such by using the iprobe option to Pnoise.

The goal of Pnoise analysis is usually to calculate the noise at the circuit output, as shown in Figure 13-14 on page 892. Pnoise analysis also calculates an approximation to the gain $G$ as a by-product of the input-referred noise calculation. It turns out that if it is assumed that the circuit input impedance is matched, i.e., the impedance seen by the port component shown as the dashed box in Figure 13-14 on page 892, is the same as the source impedance, then the gain needed for the input-referred noise calculation is also the gain $G$.

However, the Pnoise analysis is based on a series of PXF analyses and when the input is not matched, then a single PXF analysis is not sufficient to calculate $G$. PSP analysis is needed to accurately calculate $G$ independent of match conditions.

**Figure 13-14  Pnoise with Port Component**




Coincidentally, because of the matched-input assumption made both in Pnoise analysis and in the definition of the port, the gain G from circuit input to circuit output that is reported by Pnoise analysis happens to be exactly twice the gain from the source to the output, Gs. This can be seen by noting that if the circuit input is matched, then the circuit acts as a voltage divider, and the voltage at Vin is

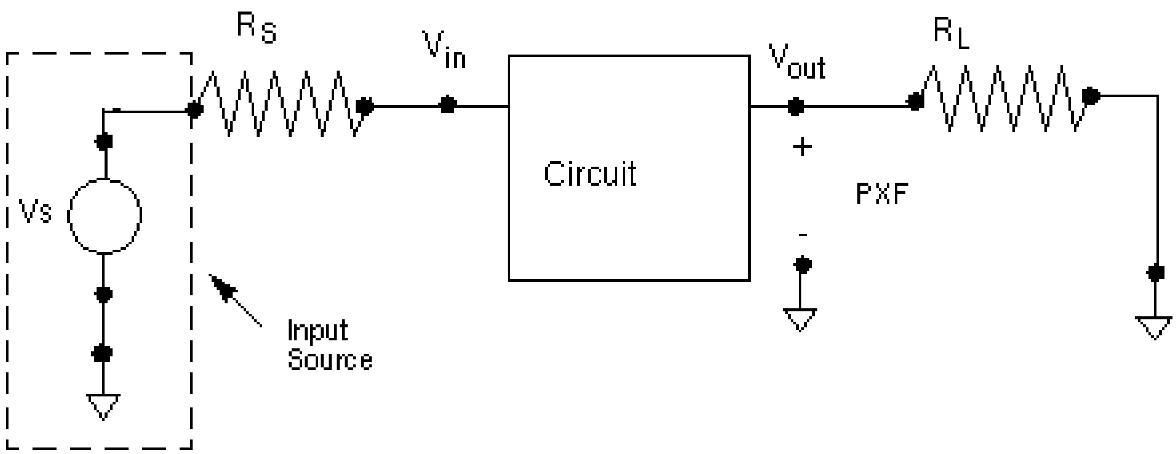
$$Vin = \frac{Zin}{Zin + Rs} Vs = \frac{Rs}{Rs + Rs} Vs = \frac{Vs}{2}$$

Thus in this special case, Vout/Vin is twice Vout/Vs. The `gain` reported by Pnoise is twice Gs regardless of the match conditions.

To alleviate this confusion, in future releases the `gain` number will not be reported by Pnoise analysis when the input is driven by a port.

Now consider when a separate vsource and resistor are used to drive the circuit, as shown in Figure 13-15 on page 893. In this case, the voltage source would be specified as the input probe. Spectre RF computes the gain from the input probe to the output, which in this case is the gain from Vs to the output, Gs, because Vs was specified as the input source.

**Figure 13-15  Pnoise With vsrc Component**




## Phase Noise

The Direct Plot form in the analog design environment plots *output noise* and *phase noise*. The phase noise form is designed for use in oscillator noise computations.

The *output noise* plots are the Total Output Noise (*out*) data, taken directly from the psf files with no modification.

The *phase noise* plots show the noise power relative to the carrier power. Technically the plot is not phase noise, simply normalized output noise. However, for oscillators, close to the fundamental frequency, the noise is mostly phase noise.

The normalization is done relative to the power in the fundamental component of the noise-free oscillation as calculated by the PSS analysis. If you look at the PSS analysis results in the frequency domain, the fundamental has a particular amplitude $V1$, which means that the fundamental component of the oscillation is $V1\cos(2pf_c + j)$, for some $\varphi$, where $f_c$ is the fundamental frequency.

Defining

$\text{noise power} = \text{out}^2$,

where $\text{out}$ is the *Total Output Noise*

$\text{carrier fundamental power} = (V_1)^2/2$

The normalized power is $\text{normalized power} = (\text{out2})/[(V_1)^2/2]$

Thus the formula used by Direct Plot in Artist 4.4.3 and later to plot the normalized power (*phase noise*) in `dB10` is:

$$\text{normalized power in dB} = 10\log 10\{(\text{out}^2)/[(V_1)^2/2]\}$$

For more information on phase noise, see Chapter 14, "Oscillator Noise Analysis" which summarizes how to get good phase noise calculations.

# Frequently Asked Questions

**The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong.**

The key to noise figure computation is to remember that three pieces of information are needed:

- The total noise at the output
- How much of that noise is due to the output load
- How much of that noise is due to the input source

Noise figure is the log of noise factor, and noise factor is fundamentally the total output noise less the noise due to the load ratioed to the noise due to the input. The easiest way to be sure of getting the correct noise figure is to use ports as input and output sources and loads and to have the Spectre RF simulator perform the noise figure computation.

Spectre RF correctly accounts for the noise due to the load, as well as the different possible treatments of noise from the input source that result in different types of noise figure that are used in RF circuits (customary SSB, IEEE SSB, DSB, as discussed in "Noise Figure" on page 873). See "Performing Noise Figure Computations" on page 873 for information on setting up noise figure computation.

Most mistakes in noise figure calculations arise when improperly accounting for items two and three in the list above. Hand computations of noise figure require that you correctly account for noise from the load and source, which can be difficult. For example, some users have tried to use the information from the noise summary table to calculate noise figure. The noise summary table reports total noise contributions, from all sidebands present in the Pnoise analysis, sorted by contributor. If the total noise reported in the noise summary is used as the numerator for the noise factor computation the wrong answer is obtained because the noise due to the load has not been subtracted. Likewise, the noise summary table does not sort information by sideband. To compute SSB noise figure, the denominator of the noise factor fraction must be only the noise from the reference sideband, not the total from all sidebands as reported in the noise summary. Noise figure calculations based on the reported input-equivalent noise, or on output noise and gain, suffer similar problems.

You can compute the noise figure by hand, but you must take care. For example, use the following steps to compute the customary single-sideband noise figure from raw data.

Perform a Pnoise analysis with the *saveallsidebands* parameter set to *yes*.

Using the results browser or direct plot, obtain the total noise at the output. Call this *OutputNoise*.

Now use the results browser to obtain the amount of total noise due to the load contributed from the zero sideband. Call this *LoadNoise*.

Finally, again using the results browser, obtain the amount of noise due to the source contributed from the reference sideband. Call this *sourcenoise*.

The noise factor is then

$$F = \frac{OutputNoise - LoadNoise}{SourceNoise}$$

Be sure to use the same units on all quantities. Note that, because in this procedure the noise due to the source and load are obtained by hand inspection, the procedure can be used to compute noise figure even if ports are not used to drive the circuit. Hand calculations are not recommended because they are tedious and prone to error.

In versions of the Analog Circuit Design Environment prior to 4.4.5, it was possible to perform noise figure computations in ways that incorrectly or incompletely specified the sources/input or load/output. Because, at that time, there was no way for the simulation environment to track down which component was contributing, for example, the *LoadNoise*, this part of the noise factor might have been neglected.

The rule of thumb is: *if you have not specified your sources and loads to the simulator, the simulator does not know what the loads and sources are*, so the simulator may not be able to compute noise figure properly. Generally, the input to the circuit must be a *port*, and *not* some other electrically equivalent combination of components. The output may be a resistor probe, or it may be a port. In up-to-date versions of software, 4.4.2 and later, using ports (or possibly a resistor as the output probe) will insure accurate computations. Inspect the netlist if you are unsure about the identity of the components presented to the simulator.

Finally, if you ever have any doubt about the propriety of the procedure you are using, use the results browser to access the noise figure directly from the simulation data. This data is correct in all software versions.

#### Why is the gain reported by Pnoise twice the gain that I expect?

- First, unlike some circuit simulators, Pnoise and PSP compute the gain of the circuit from its input to its output, instead of from an external source to the output. See "Definitions of Gain" on page 887 to see if the gain you expect is actually the gain that you want.
- Second, Pnoise makes certain assumptions about input matching in order to calculate this gain. See "Gain Calculations in Pnoise" on page 891 for further explanations of the way Pnoise calculates gain.
- In general, you should not use the gain calculated by Pnoise unless you know your circuit is input-matched. Use PSP instead.
- Does Spectre RF compute single-sideband (SSB) or double-sideband (DSB) noise figure?

The Spectre RF Pnoise analysis computes SSB noise figure, (except in certain special cases). PSP can compute both SSB and DSB noise figure. See "Noise Figure" on page 873.

#### Why does the axis for noise or noise figure have negative frequencies?

The axis labeling in Spectre RF is fairly independent of the actual computation. In all cases, Spectre RF computes single-sided noise (that is, there is no need to add noise from *negative* frequencies to get the total noise power or noise figure).

Recall that PSP analysis performs many computations at once—periodic S-parameters, noise correlation matrices, noise figure, etc. In 4.4.5, for historical reasons, all these quantities are stored as a sweep with a single axis. An axis appropriate for visualizing s11 may not be good to display NF.

To obtain strictly positive axis labels for noise quantities in PSP, choose a positive integer for the output harmonic, use *sweeptype=relative*, and set *freqaxis=out* on the PSP analysis options form.

#### How does match affect the noise and gain calculations in Spectre RF?

If the input impedance is not matched to the port impedance, the gain and input referred noise, or IRN, reported by Pnoise analysis is probably not the gain you want. out, F, NF, and in (input-referred noise) are still correct.

PSP analysis computes the correct G, *i*n (input-referred noise), F and NF regardless of match.

### Does Spectre RF include noise parameter data from S-parameter files?

Spectre RF supports writing and reading S-parameter files. Noise parameter data can be written into a file, can be read from a file, and can be included in a simulation.

# Known Problems and Limitations

## Dubious AC-Noise Analysis Features

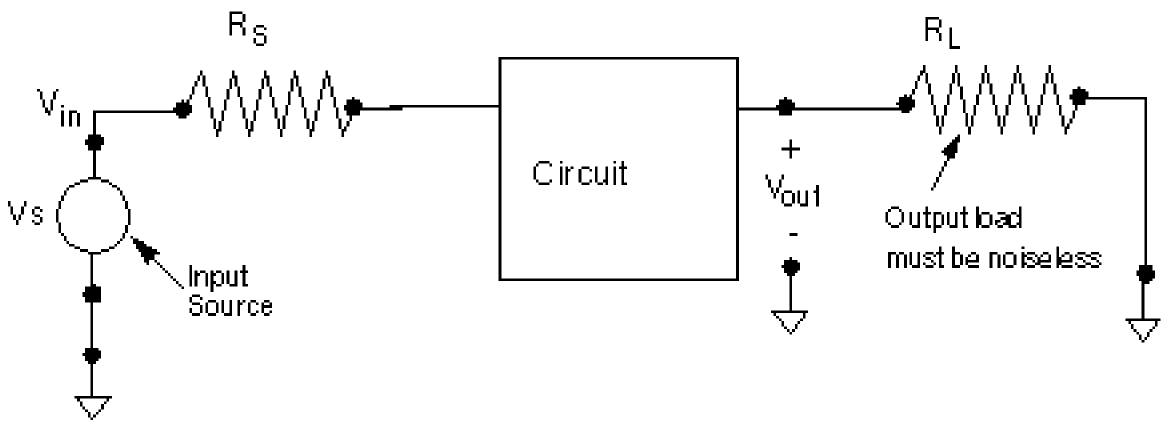
When computing noise figure in circuits with a DC bias point in releases prior to 4.4.5, the noise figure available from the direct plot form was obtained from knowledge of the total output noise computed with a Noise analysis, the source impedance as entered on a form, and circuit gain data obtained in an auxiliary AC analysis. Simulators other than Spectre RF still require this procedure.

In 4.4.5, if you use ports as the output load and input source, the noise figure is obtained directly from the Spectre RF simulator data. An artifact of the previous use model is that the noise figure direct plot still requires an AC analysis to run, but this analysis has no effect on the computed noise figure if ports are used as sources and loads.

If ports are not used as the input source and load (for example, using a *vsin* component as the input and a pair of nodes as the output) it is still possible to compute noise figure in the Analog Circuit Design Environment, but the procedure is not recommended as it can be tricky to obtain the desired result (see "Performing Noise Figure Computations" on page 873 and "The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong." on page 894).

To obtain reasonable noise figure calculations using this (unsupported) procedure, you must configure the circuit as shown in Figure 13-16 on page 898. Be sure to select the proper nodes as input and output (note that a *port* component cannot be used in this topology because the *Vin* node cannot be accessed). The contribution of the output load to the total noise must be eliminated by making the node noiseless, for example by setting `Generate noise?` in a resistor component to no.

**Figure 13-16  Alternative (Undesirable) Noise Figure Computation Setup**




Note that you cannot use this topology to compute noise figure in Pnoise, SP or PSP analyses.

## Gain in Pnoise and PSP Analyses Inconsistent

The `gain` computed by the Pnoise analysis is really a number that relates equivalent input and output noise. PSP analysis computes a somewhat different quantity that can be interpreted as a circuit gain. If the input impedance of the circuit is matched, the PSP and Pnoise gains match. See the “Gain Calculations” on page 887 for details on computing circuit gains.

## Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise Analyses

Frequent users of Spectre RF may already be familiar with the concepts of *harmonics* and *sidebands*.

Typically, the term *harmonic* is used to refer to a signal at a multiple of the carrier or local oscillator frequency.

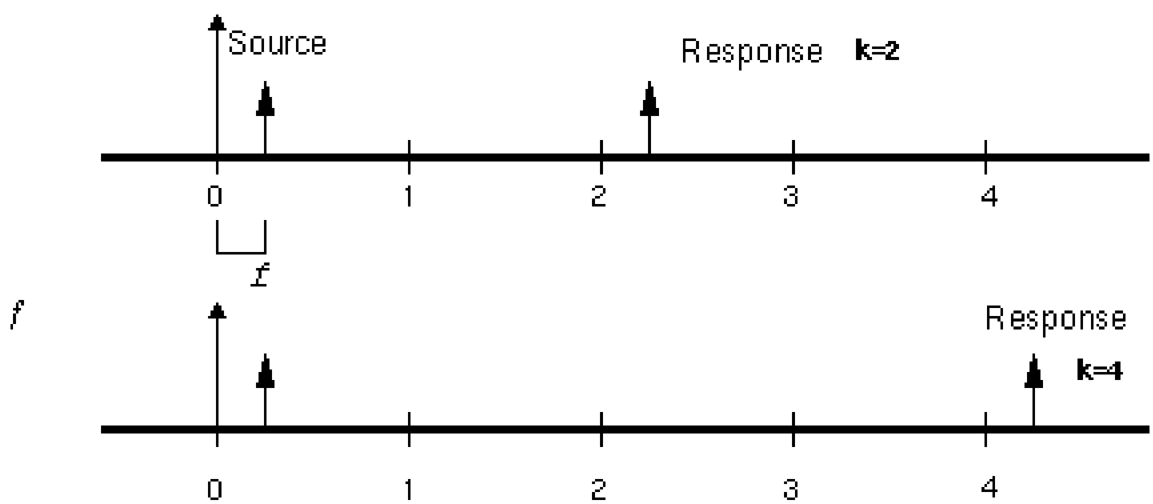
If an additional input is applied, such as for RF data, additional responses appear as *sidebands*.

In small-signal analysis, only the first-order sidebands are computed.

In the context of small-signal analysis, a frequency is specified by an index $k$ and a frequency offset $f$. These numbers specify a frequency $kf_0 + f$. In an absolute indexing scheme, such as used in PSP analysis, if the frequency $f$ is restricted to lie in an interval of size $f_0$, then for any given frequency, the numbers $k$ and $f$ are unique.

For example, consider an upconversion mixer, as shown in Figure 13-17. Two possible computations are shown. In both cases, the source is at k=0, baseband, and relative offset $f$. In the top half of the figure, the response is at k=+2, frequency $f$, and in the bottom half, k=4, offset $f$.

**Figure 13-17  Absolute Indexing Schemes in Spectre RF Small-Signal Analyses**




In contrast, PAC, PXF, and Pnoise analyses use a relative indexing scheme. In a relative indexing scheme, a single frequency may be referred to in different ways depending on context.

**Figure 13-18 Relative Indexing in Spectre RF**

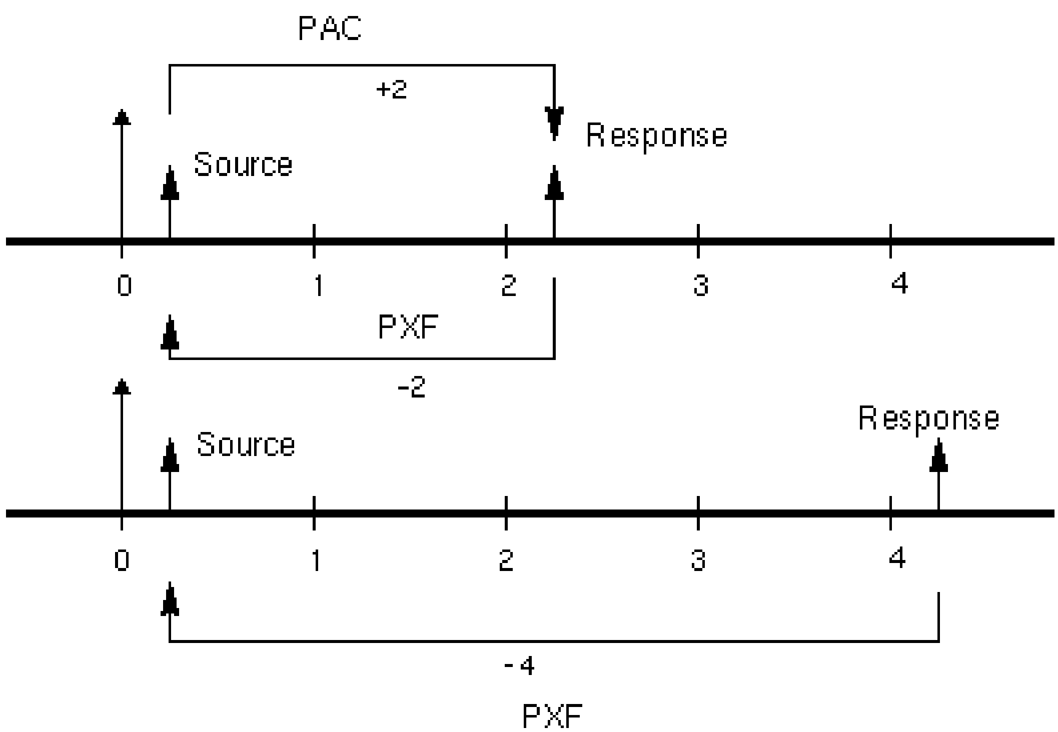



Figure 13-18 on page 900 shows how relative indexing may change depending on context. In the top half of the figure, a source may be specified in PAC analysis using either `sweeptype=absolute` and frequency $-f_0 + f$, or `sweeptype=relative`, `relharmnum=-1`, and frequency $f$. The response is shown at frequency $f_0 + f$, which is $k=2$, because $f_0 + f = -f_0 + f + 2f_0$.

If instead a PXF analysis was used to analyze the circuit, then the location of the *response* is specified with either `sweeptype=absolute` and frequency $f_0 + f$, or `sweeptype=relative` and `relharmnum=+1`. The source is at $k=-2$. Now consider the bottom half of the figure. The response is shown at the frequency $3f_0 + f$. If PXF analysis is used to analyze the circuit, the source lies at $k=-4$.

Note that if all the frequencies involved in the above examples were shifted upwards by $f_0$, then the indexing in PSP analysis would change by $+1$, but the indexing of the transfer functions in PXF and PAC analyses would remain unchanged, even though different frequencies, and thus physical different transfer functions, would be computed.

The names used for the responses and transfer functions in the Spectre RF environment are not entirely consistent in release 4.4.5 and earlier. The Spectre RF interface asks for *sidebands* in order to specify which of the transfer functions associated with the relative index $k$ are wanted. This is reasonable because the small-signal responses appear as sidebands to the harmonics of the fundamental frequency of the PSS steady-state solution. However, the data is output and displayed with the label *harmonic*. Thus the index naming is not consistent. However, because the transfer function associated with the relative index $k$ results from the small-signal being convolved with the $k^{\text{th}}$ harmonic of the steady-state solution, the *harmonic* label is not entirely inappropriate either.

# 14

# Oscillator Noise Analysis

In RF systems, local oscillator phase noise can limit the final system performance. Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) lets you rigorously characterize the noise performance of oscillator elements. This appendix explains phase noise, tells how it occurs, and shows how to calculate phase noise using the Spectre RF simulator.

"Phase Noise Primer" on page 904 discusses how phase noise occurs and provides a simple illustrative example.

"Models for Phase Noise" on page 907 contains mathematical details about how the Spectre RF simulator calculates noise and how these calculations are related to other possible phase noise models. You can skip this section without any loss of continuity, but this section can help you better understand how Spectre RF calculates phase noise and better appreciate the drawbacks and pitfalls of other simple phase noise models. This section can also help in debugging difficult circuit simulations.

"Calculating Phase Noise" on page 918 provides some suggestions for successful and efficient analysis of oscillators and discusses the limitations of the simulator.

"Troubleshooting Phase Noise Calculations" on page 920 explains troubleshooting methods for difficult simulations.

"Frequently Asked Questions" on page 925 answers some commonly asked questions about phase noise and the Spectre RF simulator.

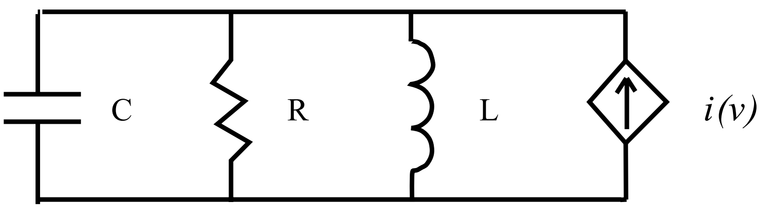
"Further Reading" on page 930 and "References" on page 930 list additional sources of information on oscillator noise analysis.

The procedures included in this appendix are intended for Spectre RF users who analyze oscillator noise. You must have a working familiarity with Spectre RF simulation and its operating principles. In particular, you must understand the Spectre RF PSS and Pnoise analyses. For information, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory.*

# Phase Noise Primer

Consider the simple resonant circuit with a feedback amplifier shown in Figure 14-1, a parallel LC circuit with nonlinear transconductance. At small capacitor voltages, the transconductance is negative, and the amplifier is an active device that creates positive feedback to increase the voltage on the capacitor. At larger voltages, where the transconductance term goes into compression, the amplifier effectively acts as a positive resistor (with negative feedback) and limits the capacitor voltage.

**Figure 14-1 A Simple Resonant Oscillator**




A simple model for the nonlinear transconductance is a cubic polynomial. We hypothesize a nonlinear resistor with a current-voltage relation given by

$$i(v) = -\left(\frac{v}{R}\right)(1 - \alpha v^2)$$

The effect of the resistor in parallel with the inductor and the capacitor can be lumped into this transconductance term. The parameter is a measure of the strength of the nonlinearity in the transconductance relative to the linear part of the total transconductance. Because the signal amplitude grows until the nonlinearity becomes significant, the value of this parameter does not affect the qualitative operation of the circuit.

For simplicity, for the remainder of this appendix

$$\alpha = 1/3$$

After some renormalization of variables, where time is scaled by

$$1/\omega_0$$

with

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

and current is scaled by

$$\sqrt{C/L}$$

You can write the differential equations describing the oscillator in the following form

$$\frac{dv}{dt} = -i + \frac{1}{Q}(1 - \alpha v^2)v + \xi(t)$$

and

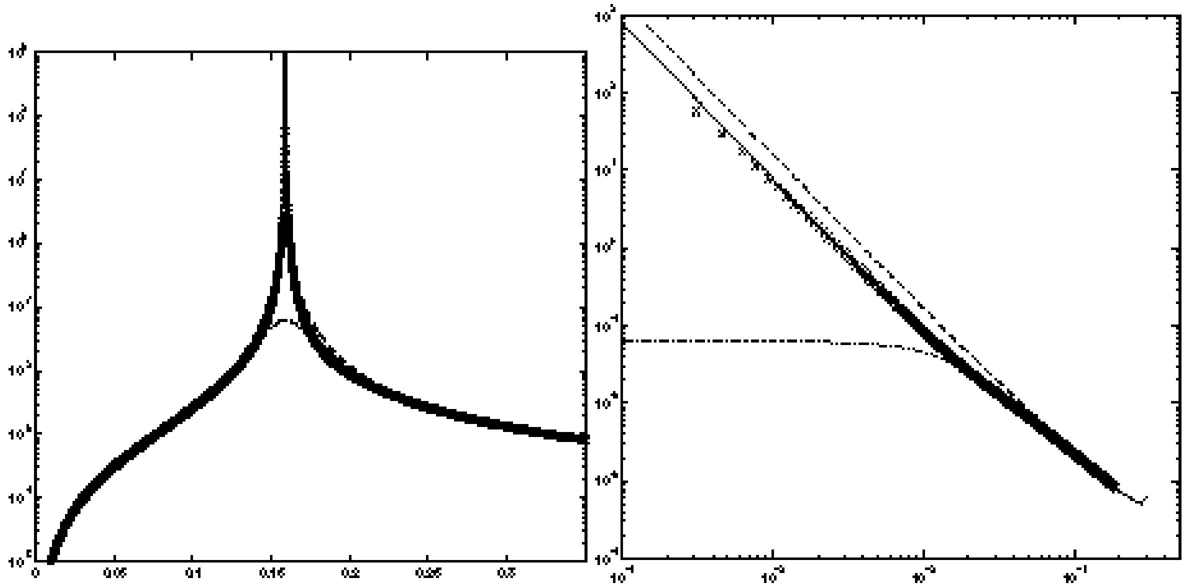
$$\frac{di}{dt} = v$$

In these equations, $v$ and $i$ represent the normalized capacitor voltage and inductor current, respectively, and $\xi(t)$ is a small-signal excitation such as white Gaussian noise, $Q = R/\omega_0 L$ is the quality factor of an RLC circuit made by replacing the nonlinear transconductance by a positive resistance $R$.

The equations just discussed describe the familiar Van der Pol oscillator system. This model includes many of the qualitative aspects of oscillator dynamics, yet it is simple enough to analyze in detail. Many more complicated oscillators that operate in a weakly nonlinear mode can be approximated with this model by using the first few terms in the Taylor series expansion of the relevant transconductances.

As a brute-force method of calculating the noise properties of this circuit, the nonlinear stochastic differential equations that describe the current and voltage processes were numerically integrated [1], and the noise power was obtained using a standard FFT-periodiagram technique. This technique requires several hundred simulations of the oscillator over many thousands of periods. Consequently, it is not a feasible approach for practical circuits, but it is rigorously correct in its statistical description even though it requires no knowledge of the properties of oscillators, noise, periodicity, or signal amplitudes. Figure 14-2 shows the total time-averaged noise in the voltage variable.

**Figure 14-2 Noise in a Simple Van der Pol System**



By plotting *Power Spectral Density* against *Normalized Noise Frequency Offset* for a Q = 5 system, Figure 14-2 shows noise in a simple Van der Pol system.

The left half of Figure 14-2 shows noise as a function of absolute frequency.

The right half of Figure 14-2 shows noise as a function of frequency offset from the oscillator fundamental frequency.

The dashed line is LC-filtered white noise, the dash-dot line is RLC-filtered white noise, the solid line is Spectre RF phase noise, and (x) marks are noise power from a full nonlinear stochastic differential equation solution.

The resulting noise power spectral density looks much like the voltage versus current response of a parallel LC circuit. The oscillator in steady-state, however, does not look like an LC circuit. As you see in the following paragraphs, this noise characteristic similarity occurs because both systems have an infinite number of steady-state solutions.

The characteristic shape of the small-signal response of an LC circuit results because an excitation at the precise resonant frequency can introduce a drift in the amplitude or phase of the oscillation. The magnitude of this drift grows with time and is potentially unbounded. In the frequency domain, this drift appears as a pole on the imaginary axis at the resonant frequency. The response is unbounded because no restoring force acts to return the amplitude or phase of the oscillation to any previous value, and perturbations can therefore accumulate indefinitely.

Similarly, phase noise exists in a nonlinear oscillator because an autonomous oscillator has no time reference. A solution to the oscillator equations that is shifted in time is still a solution. Noise can induce a time shift in the solution, and this time shift looks like a phase change in the signal (hence the term *phase noise*). Because there is no *resistance* to change in phase, applying a constant white noise source to the signal causes the phase to become increasingly uncertain relative to the original phase. In the frequency domain, this corresponds to the increase of the noise power around the fundamental frequency.

If the noise perturbs the signal in a direction that does not correspond to a time shift, the nonlinear transconductance works to put the oscillator back on the original trajectory. This is similar to AM noise. The signal uncertainty created by the amplitude noise remains bounded and small because of the action of the nonlinear amplifier that created the oscillation. The LC circuit operates differently. It lacks both a time (or phase) reference and an amplitude reference and therefore can exhibit large AM noise.

Another explanation of the similarity between the oscillator and the LC circuit is that both are linear systems that have poles on the imaginary axis at the fundamental frequency, $\omega_0$. That is, at the complex frequencies $s = i\omega_0$. However, the associated transfer functions are not the same. In fact, because of the time-varying nature of the oscillator circuit, multiple transfer functions must be considered in the linear time-varying analysis.

Understanding the qualitative behavior of linear and nonlinear oscillators is the first step towards a complete understanding of oscillator noise behavior. Further understanding requires more quantitative comparisons that are presented in Models for Phase Noise. If you are not interested in these mathematical details, you might skip ahead to “Calculating Phase Noise” on page 918.

# Models for Phase Noise

This section considers several possible models for noise in oscillators. In the engineering literature, the most widespread model for phase noise is the Leeson model [2]. This heuristic model is based on qualitative arguments about the nature of noise processes in oscillators. It shares some properties with the LC circuit models presented in the previous section. These models fit well with an intuitive understanding of oscillators as resonant RLC circuits with a feedback amplifier. In the simplest treatment, the amplifier is considered to be a negative conductance whose value is chosen to cancel any positive real impedance in the resonant tank circuit. The resulting linear time-invariant noise model is easy to analyze.

## Linear Time-Invariant (LTI) Models

To calculate the noise in a parallel RLC configuration, the noise of the resistor is modeled as a parallel current source of power density

$$S(\omega) = \frac{4k_B T}{R}$$

Where $k_B$ is Boltzman's constant. In general, if current noise excites a linear time-invariant system, then the noise power density produced in a voltage variable is given by [3] as follows

$$S_v(\omega) = |H(\omega)|^2 S_i(\omega)$$

where $H(\omega)$ is the transfer function of the LTI transformation from the noise current source *input* to the voltage *output*. The transfer function is defined in the standard way to be

$$H(\omega) = \frac{v_0(\omega)}{i_s(\omega)}$$

where $i_s$ is a (deterministic) current source and $v_0$ is the measured voltage between the nodes of interest.

It follows that the noise power spectral density of the capacitor voltage in the RLC circuit is, at noise frequency $\omega = \omega_0 + \omega'$ with $\omega' << \omega_0$.

$$S_v(\omega') = \frac{4k_B TR}{1 + 4(\omega'/\omega_0)^2 Q^2}$$

where the quality factor of the circuit is

$$Q = \frac{R}{\omega_0 L}$$

The parallel resistance is $R$ (the source of the thermal noise), and $\omega_0$ is the resonant frequency.

If a noiseless negative conductance is added to precisely cancel the resistor loss, the noise power for small $\omega' / \omega_0$ becomes

$$S_v(\omega') = \frac{k_B TR}{(\omega'/\omega_0)^2 Q^2}$$

This linear time-invariant viewpoint explains some qualitative aspects of phase noise, especially the $(\omega_0 / Q\omega')^2$ dependencies. However, even for this simple system, a set of complicating arguments is needed to extract approximately correct noise from the LTI model. In particular, we must explain the 3 dB of excess amplitude noise inside the resonant bandwidth generated by an LC model but not by an oscillator (see “Amplitude Noise and Phase Noise in the Linear Model” on page 913). Furthermore, many oscillators, such as relaxation and ring oscillators, do not naturally fit this linear time-invariant model. Most oscillators are better described as time-varying (LTV) circuits because many phenomena, such as upconversion of $1/f$ noise, can only be explained by time-varying models.

## Linear Time-Varying (LTV) Models

For linear time-invariant systems, the noise at a frequency $\omega$ is directly due to noise sources at that frequency. The relative amplitudes of the noise at the system outputs and the source noise are given by the transfer functions from noise sources to the observation point. Time-varying systems exhibit frequency conversion, however, and each harmonic $k\omega_0$ in the oscillation can transfer noise from a frequency $\omega \pm k\omega_0$ to the observation frequency $\omega$. In general, for a stationary noise source $\xi(t)$, the total observed noise voltage is [3]

$$S_v(\omega) = \sum_k \left|H_k(\omega)\right|^2 S_\xi(\omega + k\omega_0)$$

Each term in the series represents conversion of current power density at frequency $\omega + k\omega_0$ to voltage power density at frequency $\omega$ with gain $|H_k(\omega)|^2$. As an example, return again to the Van der Pol oscillator with $\alpha = 1/3$ and notice how a simple time-varying linear analysis of noise proceeds.

The first analysis step for the Van der Pol oscillator is to obtain a large-signal solution, so you set $\xi(t) = 0$. In the large-Q limit, the oscillation is nearly sinusoidal and so it is a good approximation to assume the following

$$v(t) = a \sin \omega_0 t$$

The amplitude, $a$, and oscillation frequency can be determined from the differential equations that describe the oscillator. Recognizing that

$$i(t) = \left(\frac{a}{\omega_0}\right) \cos(\omega_0 t)$$

and substituting into the equation for $dv/dt$, $a$ and $\omega_0$ are determined by the following

$$a\omega_0 \cos(\omega_0 t) - \left(\frac{1}{Q}\right)\left(a \sin(\omega_0 t) - \frac{a^3}{3} \sin^3(\omega_0 t)\right) - \left(\frac{a}{\omega_0}\right) \cos(\omega_0 t) = 0$$

Substituting

$$\sin^3(\omega_0 t) = \frac{3 \sin(\omega_0 t) - \sin(3\omega_0 t)}{4}$$

and using the orthogonality of the sine and cosine functions, it follows that

$$a - \left(\frac{a^3}{4}\right) = 0$$

and

$$\omega_0 - \left(\frac{1}{\omega_0}\right) = 0$$

(The $\sin(3\omega_0 t)$ term is relevant only when we consider higher-order harmonics of the oscillation.) Therefore, to the lowest order of approximation, $a = 2$ and $\omega_0 = 1$.

The only nonlinear term in the Van der Pol equations is the current-voltage term, $v^3/3$. This term differentiates the Van der Pol oscillator from the LC circuit. The small-signal conductance is the derivative with respect to voltage of the nonlinear current

$$\frac{-(1-v^2)}{Q}$$

With

$$v(t) = 2\sin t$$

the small-signal conductance as a function of time is

$$(1/Q)(1-\cos 2t)$$

Because there is a nonzero, time-varying, small-signal conductance, the Periodic Time Varying Linear (PTVL) model is different from the LTI LC circuit model. In fact, the time-average conductance is not even zero. However, the time-average power dissipated by the nonlinear current source is zero, a necessary condition for stable, sustained oscillation.

Oscillators are intrinsically time-varying elements because they trade off excessive gain during the low-amplitude part of the cycle with compressive effects during the remainder of the cycle. This effect is therefore a generic property not unique to this example.

To complete the noise analysis, write the differential equations that the small-signal solution $i_{s(t)}$, $v_{s(t)}$ must satisfy,

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(1-3\alpha v^2(t))v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

From the large signal analysis, $v(t) = 2\sin t$, and so

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(2\cos 2t - 1)v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

The time-varying conductance can mix voltages from a frequency $\omega$ to $\omega$ - $2$. For small $\omega'$, if an excitation is applied at a frequency $\omega = 1 + \omega'$, $i_s$ and $v_s$ are expected to have components at $1 + \omega'$ and $-1 + \omega'$ for the equations to balance. (Higher-order terms are again presumed to be small.) Writing

$$i_{s(t)} = i_{e^{(i\,(1 + \omega')\,t)}} + i_{-e^{(i\,(-1 + \omega')\,t)}}$$

and substituting into the small-signal equations with

$$\xi(t) = c_{e^{i\,(1 + \omega')\,t}}$$

leads to the following system of equations for $i_+$ and $i_-$

$$\begin{bmatrix} 1 - (1 + \omega')^2 + \left(\frac{i}{Q}\right)(1 + \omega') & -\left(\frac{i}{Q}\right)(-1 + \omega') \\ -\left(\frac{i}{Q}\right)(1 + \omega') & 1 - (-1 + \omega')^2 + \left(\frac{i}{Q}\right)(-1 + \omega') \end{bmatrix} \begin{bmatrix} i_+ \\ i_- \end{bmatrix} = \begin{bmatrix} c_+ \\ 0 \end{bmatrix}$$

Solving these equations gives the transfer function from an excitation at frequency $1 + \omega'$ to the small-signal at frequency $1 + \omega'$ that we call $H_0(\omega')$. A similar analysis gives the other significant transfer function, from noise at frequency $-1 + \omega'$ of amplitude C_ to the small-signal response at frequency $1 + \omega'$, that we call $H_{-2}(\omega')$. In the present case, for small $\omega'$,

$$H_0^2 \cong H_{-2}^2 \cong \frac{R^2}{16Q^2\left(\frac{\omega'}{\omega_0}\right)^2}$$

For a general Van der Pol circuit with a parallel resistor $R$ that generates white current noise, $\xi(t)$, with $S_\xi(\omega) = 4\,k_B\,T/R$,

$$S_v(\omega') = \frac{k_B TR}{2\left(\frac{\omega'}{\omega}\right)^2 Q^2}$$

Note that this is precisely one-half the noise predicted by the LC model.

You can gain additional insight about phase noise by analyzing the time-domain small-signal response. The small-signal current response is.

$$i_s(t) = \frac{ie^{i\omega' t}(c_- + c_+)}{2\omega'} \sin t$$

Notice that $c_+$ and $c_-$ are complex random variables that represent the relative contribution of white noise at separate frequencies. As white noise has no frequency correlations, they have uncorrelated random phase, and thus zero amplitude expectation, and unit variance in amplitude. Because the large-signal current is $i(t) = 2\cos t$, and the sine and cosine functions are orthogonal, the total noise for small $\omega'$ that we computed is essentially all phase noise.

## Amplitude Noise and Phase Noise in the Linear Model

Occasional claims are made that in oscillators, “Half the noise is phase noise and half the noise is amplitude noise.” However, as the simple time-varying analysis in the previous section shows, in a physical oscillator the noise process is mostly phase noise for frequencies near the fundamental. It is true that in an LC-circuit half the total noise power corresponds to AM-like modulation and the other half to phase modulation. In the literature, the AM part of the noise is sometimes disregarded when quoting the oscillator noise although this is not always the case. (The Spectre RF simulator computes the total noise generated by the circuit; see “Details of the Spectre RF Calculation” on page 914).

However, a *linear* oscillator does not really exist. Physical oscillators operate with a tradeoff of gain that causes growing signal strength and nonlinear compressive effects that act to limit the signal amplitude. For noise calculation, the oscillator cannot be considered a linear time-invariant system because there are intrinsic nonlinear effects that produce large phase noise but limited amplitude noise. Oscillators are time-varying, and they therefore require a time-varying small-signal analysis.

Arguments that start with stationary white noise and pass it through a linear model in a forward-analysis fashion produce incorrect answers. This is true because they neglect the time-variation of the conductances (and possibly the capacitances) in the circuit. In the simple cases considered here, the conductances vary in time in a special way so as to produce no amplitude noise, only phase noise.

They have that special variation because they result from linearization about an oscillator limit cycle. An oscillator in a limit cycle has a large response to phase perturbations, but not to amplitude perturbations. The amplitude perturbations are limited by the properties of the nonlinear amplifier, but the phase perturbations can persist. The Spectre RF simulator calculates the correct phase noise because it *knows* about the oscillator properties.

Similarly, arguments [13] that start with noise power and derive phase noise in a backwards fashion also usually produce incorrect results because they cannot correctly account for frequency correlations in the noise of the oscillator. These frequency correlations are introduced by the time-varying nature of the circuit.

Occasionally, a netlist appears in which a negative resistance precisely cancels a positive resistance to create a pure LC circuit. Because such a circuit has an infinite number of oscillation modes, the Spectre RF simulator cannot correctly calculate the noise because it assumes a unique oscillation. Such a circuit is not physically realizable because adding or subtracting a microscopically small amount of conductance makes the circuit either go into nonlinear operation (amplifier saturation) or become a damped LC circuit that has a unique final equilibrium point. This equilibrium point is the zero-state solution. Trying to create the negative resistance oscillator is like trying to bias a circuit on a metastable point. Any amplitude oscillation can exist, depending on the initial conditions, as long as the amplitude is less than the amplifier saturation point.

## Details of the Spectre RF Calculation

This section contains the mathematical details of how the Spectre RF simulator computes noise in oscillators. Understanding the material in this section can help you troubleshoot and understand difficult oscillator problems.

The analysis the Spectre RF simulator performs is similar to the simple analysis in the section "Linear Time-Varying (LTV) Models" on page 909. During analysis, the Spectre RF simulator

1. First finds the periodic steady state of the oscillator using the PSS analysis.

2. Then linearizes around this trajectory.

3. The resulting time-varying linear system is used to calculate the noise power density. The primary difference between the Spectre RF calculation and the previous analysis is that the basis functions used for the Spectre RF calculation are not just a few sinusoids, but rather a collection of many piecewise polynomials. The use of piecewise polynomials allows the Spectre RF simulator to solve circuits with arbitrary waveforms, including circuits with highly nonlinear behavior.

Noise computations are usually performed with a small-signal assumption, but a rigorous small-signal characterization of phase noise is complicated because the variance in the phase of the oscillation grows unbounded over time. From a mathematical viewpoint, an oscillator is an autonomous system of differential equations with a stable limit cycle. An oscillator has phase noise because it is neutrally stable with respect to noise perturbations that move the oscillator in the direction of the limit cycle. Such *phase* perturbations persist with time, whereas transverse fluctuations are damped with a characteristic time inversely proportional to the quality factor of the oscillator.

Further care is necessary because, in general, the two types of excitations (those that create phase slippage and those responsible for time-damped fluctuations) are not strictly those that are parallel or perpendicular, respectively, to the oscillator trajectory, as is sometimes claimed (for example, in [4]).

However, one must realize that the noise powers at frequencies near the fundamental frequency correspond to correlations between points that are widely separated on the oscillator envelope. In other words, they are long-time signal effects. In fact, asymptotically (at long times), the ratio of the variance of any state variable to its power at the fundamental frequency is unity for any magnitude of the noise excitation. Therefore, in practical cases, you can consider only small deviations in the state variables when describing the phase noise.

The first step in the noise analysis is to determine the oscillator steady-state solution. This is done in the time domain using shooting methods [5]. After the periodic steady-state is obtained, the circuit equations are linearized around that waveform in order to perform the small-signal analysis.

The time-varying linear system describing the small-signal response $\mathrm{vs}(t)$ of the oscillator to a signal $w(t)$ can be written in general form as [6, 7]

$$\left[C(t)\frac{d}{dt} + G(t)\right]v_S \equiv L(t)v_S(t) \ = \ w(t)$$

where $\mathrm{C}(t)$ and $\mathrm{G}(t)$ represent the linear, small-signal, time-varying capacitance and conductance matrixes, respectively. These matrixes are obtained by linearization about the periodic steady-state solution (the limit cycle). To understand the nature of time-varying linear analysis, the concept of Floquet multipliers is introduced.

Suppose $x(t)$ is a solution to the oscillator circuit equations that is periodic with period $T$. If $x(0)$ is a point on the periodic solution $x_L(t)$, then $x(T) = x(0)$. If $x(0)$ is perturbed slightly off the periodic trajectory, $x(0) = x_L(0) + \delta x$, then $x(T)$ is also perturbed, and in general for small $\delta x$,

$$x(T) - x_L(T) \approx \frac{\partial x(T)}{\partial x(0)} \delta x$$

The Jacobian matrix

$$\frac{\partial x(T)}{\partial x(0)}$$

is called the sensitivity matrix. The Spectre RF simulator uses an implicit representation of this matrix both in the shooting method that calculates the steady-state and in the small-signal analyses. To see how the sensitivity matrix relates to oscillator noise analysis, consider the effect of a perturbation at time $t = 0$ several periods later, at $t = nT$. From the above equation,

$$x(nT) - x_L(nT) \approx \left[\frac{\partial x(T)}{\partial x(0)}\right]^n \delta x$$

so

$$x(nT) - x_L(nT) \approx \sum_i C_i \lambda_i^n \phi_i$$

where $\phi_i$ is an eigenvector of the sensitivity matrix. The $C_i$ are the expansion coefficients of $\delta x$ in the basis of $\phi_i$. If $\psi_i$ is a left eigenvector (an eigenvector of its transpose) of the sensitivity matrix, then

$$C_i = \psi_i^T \delta x$$

Let $\lambda$ be an eigenvalue of the sensitivity matrix. In the context of linear time-varying systems, the eigenvalues $\lambda$ are called *Floquet multipliers*. If all the $\lambda$ have magnitude less than one (corresponding to left-half-plane poles), the perturbation decays with time and the periodic trajectory is stable. If any $\lambda$ has a magnitude greater than one, the oscillation cannot be linearly stable because small perturbations soon force the system away from the periodic trajectory $x_L(t)$.

A stable nonlinear physical oscillator, however, must be neutrally stable with respect to perturbations that move it in the direction of the orbit. These are not necessarily perturbation *in* the direction of the orbit because, in general,

$$\Psi \neq \phi_1$$

This is true because a time-shifted version of the oscillator periodic trajectory still satisfies the oscillator equations. In other words, one of the Floquet multipliers must be equal to unity. This Floquet multiplier is responsible for phase noise in the oscillator. The associated eigenvector determines the nature of the noise.

If $\lambda = e^{\eta}$ is a Floquet multiplier, then $\eta + ik\omega_0$ is a pole of the time-varying linear system for any integer $k$. Therefore, because of the unity Floquet multiplier, the time-varying linear system has poles on the imaginary axis at $k\omega_0$. This is very similar to what occurs in a pure LC resonator, and it explains the identical shape of the noise profiles.

Because operator $L(t)$ has poles at the harmonics of the oscillation frequency, numerical calculations of the noise at nearby frequencies become inaccurate if treated in a naive manner [8, 9]. To correctly account for the phase noise, the Spectre RF simulator finds and extracts the eigenvector that corresponds to the unity Floquet multiplier. To correctly extract the phase noise component, both the right and left eigenvectors are required. After these vectors are obtained, the singular (phase noise) contribution to the noise can be extracted. The remaining part of the noise can be obtained using the usual iterative solution techniques [6] in a numerically well-conditioned operation.

In Figure 14-2 on page 906, you can see that the Spectre RF PTVL analysis correctly predicts the total noise, including the onset of 3 dB amplitude noise outside the bandwidth of the resonator. Note that this simulation was conducted at

$$E\left\{\xi^2(t)\right\} = 10^{-3}$$

which represents a very high noise level that is several orders of magnitude higher than in actual circuits. The good match of the PTVL models to the full nonlinear simulation shows the validity of the PTVL approximation.

# Calculating Phase Noise

The following sections suggest simulation parameters, give you tips for using these parameters, and advise you about checking for accuracy.

## Setting Simulator Options

The Spectre RF time-varying small-signal analyses are more powerful than the standard large-signal analyses (DC, TRAN) but, like any precision instrument, they also have greater sensitivity to numerical errors. For many circuits, particularly oscillators, more simulator precision is needed to get good results from the PAC, PXF, and Pnoise calculations than is needed to get good DC or TRAN results.

The small-signal analyses operate by linearizing around the periodic steady state solution. Consequently, the oscillator noise analysis, and the periodic small-signal analyses in general, inherit most of their accuracy properties from the previous PSS simulation. You must be sure the PSS simulation generates a sufficiently accurate linearization. See "What Can Go Wrong" on page 921 for a discussion.

Table 14-1 recommends simulator options for various classes of circuits.

**Table 14-1 Recommended Spectre RF Parameter Values**

| Circuit | errpreset | reltol | vabstol | iabstol |
|---|---|---|---|---|
| Easy | moderate | $1.0e^{-4}$ | default | default |
| Hard-I | conservative | $1.0e^{-5}$ | 10n | 1p |
| Hard-II | | $1.0e^{-6}$ | 1n | 0.1p |
| Hard-III | | $1.0e^{-7}$ | 0.1n | 0.1p |

- *Easy* circuits are low-$Q$ (about $Q < 10$) resonant oscillators, ring oscillators, and weakly-nonlinear relaxation oscillators. Most textbook circuits are in this category. This is the default setting when you set `errpreset=moderate`.
- *Hard-I* circuits are most high-$Q$ resonant oscillators; circuits with complicated AGC, load, or bias circuitry; and relaxation or ring oscillators that exhibit moderate to strong nonlinear or *stiff* effects. This is the best general-purpose set of options and it is the default when you set `errpreset=conservative`.
- *Hard-II* and *Hard-III* circuits include a few particularly difficult circuits.

Usually these options are used only in a convergence study or for circuits that previously failed a conversion study using less strict options. Circuits in this category often exhibit some form of unusual behavior (see "What Can Go Wrong" on page 921). Sometimes this behavior results from circuit properties, for example, some very high-$Q$ crystal oscillators and some very stiff relaxation oscillator circuits. Occasionally, the behavior reflects a design flaw.

Usually setting `method=gear2only` is recommended for the PSS simulation (but see "What Can Go Wrong" on page 921).

The parameters in Table A-1 are used in error control at two places.

At the local truncation error (LTE) at each transients integration. The LTE control formula is

$$v_n(t) - v_{n,\,pred}(t) < lteratio \times [reltol \times v_{n,\,max} + vabstol]$$

- To control the periodicity error over the period. The periodicity error control formula is

$$v_n(0) - v_n(T) < lteratio \times steadyratio \times [reltol \times v_{n,\,max} + vabstol]$$

The default value for `vabstol` is `1.0e`$^{-6}$, The default value for `iabstol` is `1.0e`$^{-12}$ which is accurate enough for most cases. Tighten `iabstol` when necessary. Refer to section 4.3 in [17] for detailed discussion about transient integration.

To speed up transients integration during the `tstab` stage, the default values at the `tstab` stage are `reltol=1.0e-3`, `lteratio=3.5`, `relref=sigglobal`, `maxstep=T/25`. `method=traponly`. After the `tstab` stage, those parameters are set back according to `errpreset`.

For circuits, such as extremely high Q oscillators, that need very high accuracy, you can gain further accuracy by turning on the highorder refinement `highorder=yes` and `errpreset=conservative.` This runs multi-interval Chebyshev PSS refinement after the shooting phase of the PSS analysis.

An effective and fast method to start the oscillator is to run the new autonomous envelope analysis and to save the simulation results. Use the results of the autonomous envelope analysis as the initial condition for the PSS analysis.

A longer `tstab` stage helps with PSS convergence. However a longer `tstab` stage can slow the simulation. Using autonomous envelope analysis to establish `tstab` is considerably faster than using the transient analysis for `tstab.` See *Virtuoso Spectre Circuit Simulator RF Analysis Theory.* for information on the autonomous envelope analysis.

For releases before MMSIM60 USR1, an effective method to start the oscillation is to add a kicker to the circuit. The kicker can be either a voltage or current source. To effectively start the oscillation, the kicker has to be placed at the most sensitive place which usually is close to the oscillating transistor. The kicker can be either a PWL source or a damped sinusoidal source with frequency set to the oscillation frequency. The kicker must die down and remain stable after oscillation is established to avoid affecting the PSS analysis.

# Troubleshooting Phase Noise Calculations

The Spectre RF simulator calculates noise effectively for most oscillators. However, circuits that are very stiff, very nonlinear, or just poorly designed can occasionally cause problems for the simulator. Stiff circuits exhibit dynamics with two or more very different time scales; for example, a relaxation oscillator with a square-wave-like periodic oscillation. Over most of the cycle, the voltages change very slowly, but occasional rapid transitions are present. This section describes some of the reasons for the problems, what goes wrong, how to identify problems, and how to fix them.

See “Details of the Spectre RF Calculation” on page 914 for help troubleshooting particularly difficult circuits.

## Known Limitations of the Simulator

Any circuit that does not have a stable periodic steady-state cannot be analyzed by the Spectre RF simulator because oscillator noise analysis is performed by linearizing around a waveform that is assumed to be strictly periodic.

For example, oscillators based on IMPATT diodes generate strong subharmonic responses and cannot be properly analyzed with the Spectre RF simulator. As another example, Colpitts oscillators, properly constructed, can be made to exhibit chaotic as well as subharmonic behavior.

Similarly, any circuit with significant large-signal response at tones other than the fundamental and its harmonics might create problems for the simulator. Some types of varactor-diode circuits might fit this category. In addition, some types of AGC circuitry and bias circuitry can create these effects.

The Spectre RF simulator cannot simulate these circuits because simulation of an autonomous circuit with subharmonic or other aperiodic components in the large signal response essentially requires foreknowledge of which frequency components are important. Such foreknowledge requires Fourier analysis of very long transient simulations and cannot be easily automated. Such simulations can be very expensive.

## What Can Go Wrong

The Spectre RF simulator can have problems in the following situations.

### Generic PSS Simulation Problems

Any difficulties in the underlying PSS analysis affect the phase noise computation. For example, underestimating the oscillator period or failing to start the oscillator properly can cause PSS convergence problems that make running a subsequent Pnoise analysis impossible.

### Hypersensitive Circuits

Occasionally, you might see circuits that are extremely sensitive to small parameter changes. Such a circuit was a varactor-tuned VCO that had the varactor bias current, and therefore the oscillation frequency, set by a $1\ \mathrm{T\Omega}$ resistor. Changing to a $2\ \mathrm{T\Omega}$ resistor, which is a $1e^{-12}$ relative perturbation in the circuit matrixes, changed the oscillation frequency from $125$ MHz to $101$ Mhz. Such extreme circuit sensitivity results in very imprecise PSS simulations. In particular, the calculated periods have relatively large variations. If precise PSS simulations are impossible, precise noise calculations are also impossible. In such a case, you must fix the circuit.

### Subharmonics or Parametric Oscillator Modulation

Sometimes bias and AGC circuitry might create small-amplitude parasitic oscillations in the large signal waveform. You can identify these oscillations by performing a transient simulation to steady-state and then looking for modulation of the envelope of the oscillation waveform. For high-$Q$ circuits and/or low-frequency parasitics, this transient simulation might be very long.

In this case, because the oscillator waveform is not actually periodic, the PSS simulation can only converge to within approximately the amplitude of the parasitic oscillation. If the waveform possesses a parasitic oscillation that changes amplitude, over one period, around $10^{-5}$ relative to the oscillator envelope, then convergence with `reltol` < $10^{-5}$ is probably not possible (assuming `steadyratio` is one or less).

These effects might also appear as a parametric sideband amplification phenomenon.

See “Frequently Asked Questions” on page 925 for more information.

### Small-Signal Frequency is Much Higher than the Fundamental Frequency

The same timesteps are used for both the small-signal analysis and the PSS analysis. If the small-signal frequency is much higher than the fundamental frequency, much smaller timesteps might be required to accurately resolve the small-signal than are needed for the large signal. To force the Spectre RF simulator to take sufficiently small timesteps in the PSS simulation, be sure the `maxacfreq` parameter is set correctly.

### Wide Timestep Variation

Occasionally, in simulations that generate PSS waveforms with timesteps that vary over several orders of magnitude, the linear systems of equations that determine the small-signal response become ill-conditioned. As a result, the noise analysis is inaccurate. Usually this occurs because you have requested excessive simulator precision; for example, nine-digit precision. You can sometimes eliminate this problem using `method = traponly` in the PSS solution. You might also set `maxstep` to a very small value in the PSS analysis or you might specify a very large `maxacfreq` value.

### Problems with Device Models

When the device models leave their physically meaningful operating range during the large-signal PSS solution, the noise calculations are usually inaccurate. Similarly, when the models are discontinuous, or have discontinuous derivatives, the small-signal analysis might be inaccurate.

### Problems Resolving Floquet Multipliers in Stiff Relaxation Oscillators

Sometimes in very stiff relaxation oscillators, the PSS solution rapidly and easily converges; but the numerically calculated Floquet multiplier associated with the PSS solution is far from unity. Typically, this multiplier is real and has a magnitude much larger than unity. The Spectre RF simulator prints a warning (see “Message III” on page 924). It is interesting that sometimes the phase noise is quite accurate even with low simulation tolerances. If you have this problem, perform a convergence study.

### Problems Resolving Floquet Multipliers in High-Q Resonant Circuits

In a physical oscillator, there is one Floquet multiplier equal to unity. In an infinite-$Q$ linear resonator, however, the multipliers occur in complex conjugate pairs. A very high-$Q$ nonlinear oscillator has another Floquet multiplier on the real axis nearly equal to, but slightly less than, one. In this presence of numerical error, however, these two real Floquet multipliers can appear to the simulator as a complex-conjugate pair. The phase noise is computed using the

Floquet vector associated with the unity Floquet multiplier. When the two multipliers appear as a complex pair, the relevant vector is undefined. When the Spectre RF simulator correctly identifies this situation, it prints a warning (see “Message III” on page 924). The solution is usually to simulate using the next higher accuracy step (see Table 14-1 on page 918). Sometimes varying `tstab` can also help with this problem.

If the circuit is really an infinite-$Q$ resonator (for example, a pure parallel LC circuit), the multipliers always appear as complex conjugate pairs and the noise computations are not accurate close to the fundamental frequency. Such circuits are not physical oscillators, and the Spectre RF simulator is not designed to deal with them; see “Amplitude Noise and Phase Noise in the Linear Model” on page 913 and “Frequently Asked Questions” on page 925.

## Phase Noise Error Messages

Spectre RF displays error messages when it encounters several types of known numerical difficulty. To interpret the error messages produced by the phase noise analysis, you must know the material in “Details of the Spectre RF Calculation” on page 914.

### Message I

**The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol and method=gear2only.**

The eigenvector responsible for phase noise was inaccurately computed and the PSS simulation tolerances might be too loose. Try simulating the circuit at the next higher accuracy setting (see Table 14-1 on page 918) and then compare the calculated noise in the two simulations.

### Message II

**The Floquet eigenspace computed by spectre PSS analysis appears to be ill-defined. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only. Check the circuit for unusual components.**

This can be an accuracy problem, or it can result from an unusual circuit topology or sensitivity. Tighten the accuracy requirements as much as possible (see Table 14-1 on page 918). If this message appears in all simulations, the noise might be incorrect even if the simulations agree.

**Message III**

**The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate and/or the oscillator possesses more than one stable mode of oscillation. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only.**

All the real Floquet multipliers were well-separated from unity, suggesting that the PSS simulation tolerances might be too loose. Simulate the circuit at the next higher accuracy setting (see Table 14-1 on page 918) and then compare the calculated noise in the two simulations. If the calculated noise does not change, it is probably correct even if this message appears in both simulations.

## The tstab Parameter

Because Spectre RF performs the PSS calculation in the time domain by using a *shooting method*, an infinite number of possible PSS solutions exist, depending on where the first timepoint of the PSS solution is placed relative to the oscillator phase.

The placement of the first timepoint is determined by the length of the initial transient simulation, which you can control using the `tstab` parameter. If the `tstab` value causes the edges of the periodic window to fall on a point where the periodic oscillator waveform is making very rapid transitions, it is very difficult for PSS to converge. Similarly, the results of the small-signal analyses are probably not very accurate. Avoid such situations. If the start of the PSS waveform falls on a very fast signal transition, you usually need to view the results of further small-signal analyses with some skepticism.

Although a poor choice of the `tstab` parameter value can degrade convergence and accuracy, appropriate use of `tstab` can help to identify problem circuits and to estimate the reliability of their noise computations.

If you perform several PSS and Pnoise computations that differ only in their `tstab` parameter values, the results should be fairly similar, within a relative deviation of the same order of magnitude as the simulator parameter `reltol`. If this is not the case, you might not have set the simulator accuracy parameters sufficiently tight to achieve an accurate solution; and you need to reset one or more of the parameters `reltol`, `vabstol`, or `iabstol`. The circuit might also be poorly designed and very sensitive to perturbations in its parameters.

If the calculated fundamental period of the oscillator varies with `tstab` even when you set `reltol`, `iabstol`, and `vabstol` to very small (but not vanishingly small) values, the circuit is probably poorly designed, exhibiting anomalous behavior, or both. (see “Known Limitations of the Simulator” on page 920).

# Frequently Asked Questions

The following questions are similar to those commonly asked about oscillator noise analysis with the Spectre RF simulator.

### Does Spectre RF simulation calculate phase noise, amplitude noise, or both?

Spectre RF simulation computes the total noise of the circuit, both amplitude and phase noise. What the analog circuit design environment plots as *phase noise* is really the total noise scaled by the power in the fundamental oscillation mode. Close enough to the fundamental frequency, the noise is all phase noise, so what the analog circuit design environment plots of *phase noise* is really the phase noise as long as it is a good ways above the noise floor.

Some discussions of oscillator noise based on a simple resonator/amplifier description describe the total noise, at small frequency offsets from the fundamental, as being half amplitude noise and half phase noise. In reality, for physical oscillators, near the fundamental nearly all the noise is phase noise. Therefore, these simple models overestimate the total noise by 3 dB. For a detailed explanation, see the phase noise theory described in “Details of the Spectre RF Calculation” on page 914 and the detailed discussion of the Van der Pol oscillator “Linear Time-Varying (LTV) Models” on page 909.

### I have a circuit that contains an oscillator. Can I simulate the oscillator separately and use the phase noise Spectre RF calculates as input for a second PSS/PNOISE simulation?

No. Oscillators generate noise with correlated spectral sidebands. Currently, Spectre RF simulation output represents only the time-average noise power, not the correlation information, so the noise cannot be input to a simulation that contains time-varying elements that might mix together noise from separate frequencies.

If the second circuit is a linear filter (purely lumped linear time-invariant elements, such as resistors, capacitors, inductors, or a linearization of a nonlinear circuit around a DC operating point) that generates no frequency mixing, then you can use the output of the Spectre RF Pnoise analysis as a *noisefile* for a subsequent NOISE (not Pnoise) analysis.

### How accurate are the phase noise calculations? What affects the errors?

Initially, it is important to distinguish between modeling error and simulation (numerical) error. If the device models are only good to 10% the simulation is only good to 10% (or worse). So, for the rest of this appendix, we discuss numerical error introduced by the approximations in the algorithms.

You must also distinguish between absolute and relative signal frequencies in the noise analysis. When the noise frequency is plotted on an absolute scale, the error is primarily a function of the variance in the calculated fundamental period. This is true because of the singular behavior, in these regions, of the phase noise near a harmonic of the fundamental. To see this behavior, note that for the simple oscillator driven by white noise, the noise power is proportional to the offset from the fundamental frequency,

$$S_v(\omega) \propto \frac{1}{(\omega - \omega_0)^2}$$

If you make a small error in the calculation of $\omega_0$, the error $\Delta S_v$ in the noise is proportional to $S / \omega_0$

$$\Delta S_v(\omega) \propto \frac{\Delta\omega_0}{(\omega - \omega_0)^3}$$

This error can be very large even if $\Delta\omega_0$, the error in $\omega_0$, is small. However, because of the way Spectre RF simulation extracts out the phase noise, the calculated phase noise, as a function of offset from the fundamental frequency, can be quite accurate even for very small offsets.

Now consider how much error is present in the calculated fundamental frequency. Because the numerical error is related to many simulation variables, it is difficult to quantify, without examination, how much is present. However, as a rough approximation, if we define the quantity

$$r = min\left\{reltol, \frac{iabstol}{max(i)}, \frac{vabstol}{max(v)}\right\}$$

where $max(i)$ and $max(v)$ are the maximum values of current and voltage over the PSS period, then, under some assumptions, $\Delta\ \omega_0$, the error in the fundamental $\omega_{0,}$ probably satisfies

$$r\omega_0 < \delta\omega_0 < Mr\omega_0$$

where $M$ is the number of timesteps taken for the PSS solution. This analysis assumes that `steadyratio` is sufficiently tight, not much more than one, and also that `iabstol` and `vabstol` are sufficiently small.

If you require a good estimate of the accuracy in the fundamental, run the PSS simulation with many different accuracy settings, initial conditions and `tstab` values (See "The tstab Parameter" on page 924). For example, to estimate how much numerical error remains in the calculated fundamental frequency for a given simulation, run the simulation; reduce `reltol`, `iabstol`, and `vabstol` by a factor of 10 to100; rerun the simulation; and then compare the calculated fundamental frequencies. For the sorts of parameters we recommend for oscillator simulations, four to five digits of precision seems typical. Past that point, round off error and anomalous effects introduced by vastly varying timesteps offset any gains from tightening the various accuracy parameters.

For phase noise calculations, again it is unrealistic to expect relative precision of better than the order of `reltol`. That is, if `reltol` is $10^{-5}$ and the oscillator fundamental is about $1$ GHz, the Spectre RF numerical fuzz for the calculated period is probably about $10$ KHz. Therefore, when plotted on an absolute frequency scale, the phase noise calculation exhibits substantial variance within about $10$ KHz of the fundamental.

However, when plotted on a frequency scale *relative* to the fundamental, the phase noise calculation might be more precise for many oscillators. If the circuit is strongly dissipative (that is, low-$Q$, such as ring oscillators and relaxation oscillators), the phase noise calculation is probably fairly accurate up to very close to the fundamental frequency even with loose simulation tolerance settings. High-$Q$ circuits are more demanding of the simulator and require more stringent simulation tolerances to produce good results. In particular, circuits that use varactor diodes as tuning elements in a high-$Q$ tank circuit appear to cause occasional problems. Small modifications to the netlist (runs with different `tstab` values and minor topology changes) can usually tell you whether (and where) the simulator results are reliable.

Simulation accuracy is determined by how precisely Spectre RF simulation can solve the augmented nonlinear boundary value problem that determines the periodic steady-state. The accuracy of the BVP solution is controlled primarily by the simulation variables `reltol`, `iabstol`, `vabstol`, `steadyratio`, and `lteratio`. Typically, `steadyratio` and `lteratio` are fixed, so `reltol` is usually the variable of interest.

Occasionally accuracy might be somewhat affected by other variables such as `relref`, `method`, the number of timesteps, and `tstab`. Again, the physical properties of the circuit might limit the accuracy.

#### I have a circuit with an oscillator and a sinusoidal source. Can I simulate this circuit with Spectre RF simulation?

In general, Spectre RF simulation is not intended to analyze circuits that contain autonomous oscillators and independent periodic sources.

If the circuit contains components that could potentially oscillate autonomously and also independent large-signal sinusoidal sources, Spectre RF simulation works properly only if two conditions are fulfilled. The system must be treated as a driven system, and the coupling from the sinusoidal sources to the oscillator components must be strong enough to lock the oscillator to the independent source frequency. (In different contexts, this is known as *oscillator entrainment* or *phase-locking*) The normal (nonautonomous) PSS and small-signal analyses function normally in these conditions.

If the autonomous and driven portions of the circuit are weakly coupled, the circuit waveform might be more complicated; for example, a two-tone (quasi-periodic) signal with incommensurate frequencies. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) Even if PSS converges, further small-signal analyses (PAC, PXF, Pnoise) almost certainly give the wrong answers.

### What is the significance of total noise power?

First, you must understand that Spectre RF simulation calculates and measures noise in voltages and currents. The total power in the phase process is unbounded, but the power in the actual state variables is bounded.

Oscillator phase noise is usually characterized by the quantity

$$d(f) = \frac{Sv(f)}{P_1}$$

where $P_1$ is the power in the fundamental component of the steady state solution and $Sv(f)$ is the power spectral density of a state variable $V$.

For an oscillator with only white-noise sources, $L(f)$ has a Lorentzian line shape,

$$L(f) = \frac{1}{\pi}\frac{a}{a^2 + f^2}$$

where $a$ is dependent on the circuit and noise sources, and thus the total phase noise power

$$\int L(f)df = 1$$

Because

$$varv(t) = R_v(t,t) = \int_{-\infty}^{\infty} S_v(f)df$$

we are led to the uncomfortable, but correct, conclusion that the variance in any variable is 100 percent of the RMS value of the variable, *irrespective of circuit properties or the amplitude of the noise sources*.

Physically, this means that if a noise source has been active, because $t = -\infty$, then the voltage variable in question is randomly distributed over its whole trajectory. Therefore, the relative variance is one. Clearly, the variance is not a physically useful characterization of the noise, and the total noise power must be interpreted carefully. What is actually needed is the variance as a function of time, given a fixed reference for the signal in question; or, more often, the rate at which the variance increases from a zero point; or, sometimes, the increment in the variance from cycle to cycle. That is, we want to specify the phase of the oscillator signal at a given time point and to find a statistical characterization of the variances relative to that time. But because of the non-causal nature of the Fourier integral, quantities like the total noise power give us information about the statistical properties of the signal over all time.

### What's the story with pure linear oscillators (LC circuits)?

Oddly enough, Spectre RF simulation is not set up to do Pnoise analysis on pure LC circuits.

Pure LC circuits are not physically realizable oscillators, and the mathematics that describes them is different from the mathematics that describes physical oscillators. A special option must be added to the code in order for Pnoise to handle *linear oscillators*. See "Models for Phase Noise" on page 907, and, in particular, "Amplitude Noise and Phase Noise in the Linear Model" on page 913. Because the normal NOISE analysis is satisfactory for these circuits and also much faster, it is unlikely that Pnoise is modified.

### Why doesn't the Spectre RF model match my linear model?

As is discussed in "Amplitude Noise and Phase Noise in the Linear Model" on page 913, the difference between the Spectre RF model (the correct answer) and the linear oscillator model is that in the linear oscillator, both the amplitude and the phase fluctuations can become large. However, in a nonlinear oscillator, the amplitude fluctuations are always bounded, so the noise is half as much, asymptotically.

We emphasize that computing the correct total noise power requires using the time-varying small signal analysis. An oscillator is, after all, a time-varying circuit by definition. Time-invariant analyses, like the *linear oscillator model*, can sometimes be useful, but they can also be misleading and should be avoided.

**There are funny sidebands/spikes in the oscillator noise analysis. Is this a bug?**

Very possibly this is parametric small-signal amplification, a real effect. This sometimes occurs when there is an AGC circuit with a very long time constant modulating the parameters of circuit elements in the oscillator loop. Sidebands in the noise power appear at frequencies offset from the oscillator fundamental by the AGC characteristic frequency.

Similarly, any elements that can create a low-frequency parasitic oscillation, such as a bias inductor resonating with a capacitor in the oscillator loop, can create these sorts of sidebands.

# Further Reading

The best references on the subject of phase noise are by Alper Demir and Franz Kaertner. Alper Demir's thesis [10], now a Kluwer book, is a collection of useful thinking about noise. Kaertner's papers [11, 12, 9] contain a reasonably rigorous and fairly mathematical treatment of phase noise calculations.

The book by W. P. Robins [13] has a lot of engineering-oriented thinking. However, it makes heavy use of LTI models, and much of the discussion about noise cannot be strictly applied to oscillators. As a consequence, you must interpret the results in this book with care.

Hajimiri and Lee's paper [4] is worth reading, but their analysis is superseded by Kaertner's.

Other references include [8, 14, 15, 16].

# References

[1] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1995.

[2] D. Leeson, "A simple model of feedback oscillator noise spectrum," *Proc. IEEE*, vol. 54, pp. 329–330, 1966.

[3] W. A. Gardner, *Introduction to random processes*. McGraw Hill, 1990.

[4] A. Hajimiri and T. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid. State Circuits*, vol. 33, pp. 179–193, 1998.

[5] R. Telichevesky, J. White, and K. Kundert, "Efficient steady-state analysis based on matrix-free krylov-subspace methods," in *Proceedings of 32rd Design Automation Conference*, June 1995.

[6] R. Telichevesky, J. White, and K. Kundert, “Efficient AC and noise analysis of two-tone RF circuits,” in *Proceedings of 33rd Design Automation Conference*, June 1996.

[7] M. Okumura, T. Sugaware, and H. Tanimoto, “An efficient small-signal frequency analysis method of nonlinear circuits with two frequency excitations,” *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 225–235, 1990.

[8] W. Anzill and P. Russer, “A general method to simulate noise in oscillators based on frequency domain techniques,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 41, pp. 2256–2263, 1993.

[9] F. X. Kärtner, “Noise in oscillating systems,” in *Proceedings of the Integrated Nonlinear Microwave and Millimeter Wave Circuits Conference*, 1992.

[10] A. Demir, *Analysis and simulation of noise in nonlinear electronic circuits and systems*. PhD thesis, University of California, Berkeley, 1997.

[11] F. X. Kaertner, “Determination of the correlation spectrum of oscillators with low noise,” *IEEE Trans. Microwave Theory and Techniques*, vol. 37, pp. 90–101, 1989.

[12] F. X. Kaertner, “Analysis of white and f-a noise in oscillators,” *Int. J. Circuit Theory and Applications*, vol. 18, pp. 485–519, 1990.

[13] W. P. Robins, *Phase Noise in Signal Sources*. Institution of Electrical Engineers, 1982.

[14] A. A. Abidi and R. G. Meyer, “Noise in relaxation oscillators,” *IEEE J. Sol. State Circuits*, vol. 18, pp. 794–802, 1983.

[15] B. Razavi, “A study of phase noise in cmos oscillators,” *IEEE J. Sol. State Circuits*, vol. 31, pp. 331–343, 1996.

[16] K. Kurokawa, “Noise in synchronized oscillators,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 16, pp. 234–240, 1968.

[17] K. Kundert, “The Designer’s Guide to Spice & Spectre,” *Kluwer Academic Publishers*, 1995.

# 15

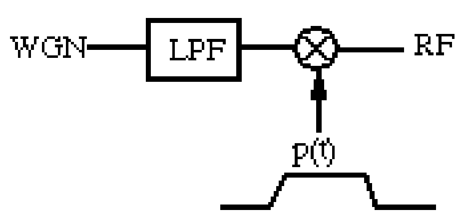
# Analyzing Time-Varying Noise

RF circuits are usually driven by periodic inputs. The noise in RF circuits is generated by sources that can therefore typically be modeled as periodically time-varying. Noise that has periodically time-varying properties is said to be cyclostationary.

## Characterizing Time-Domain Noise

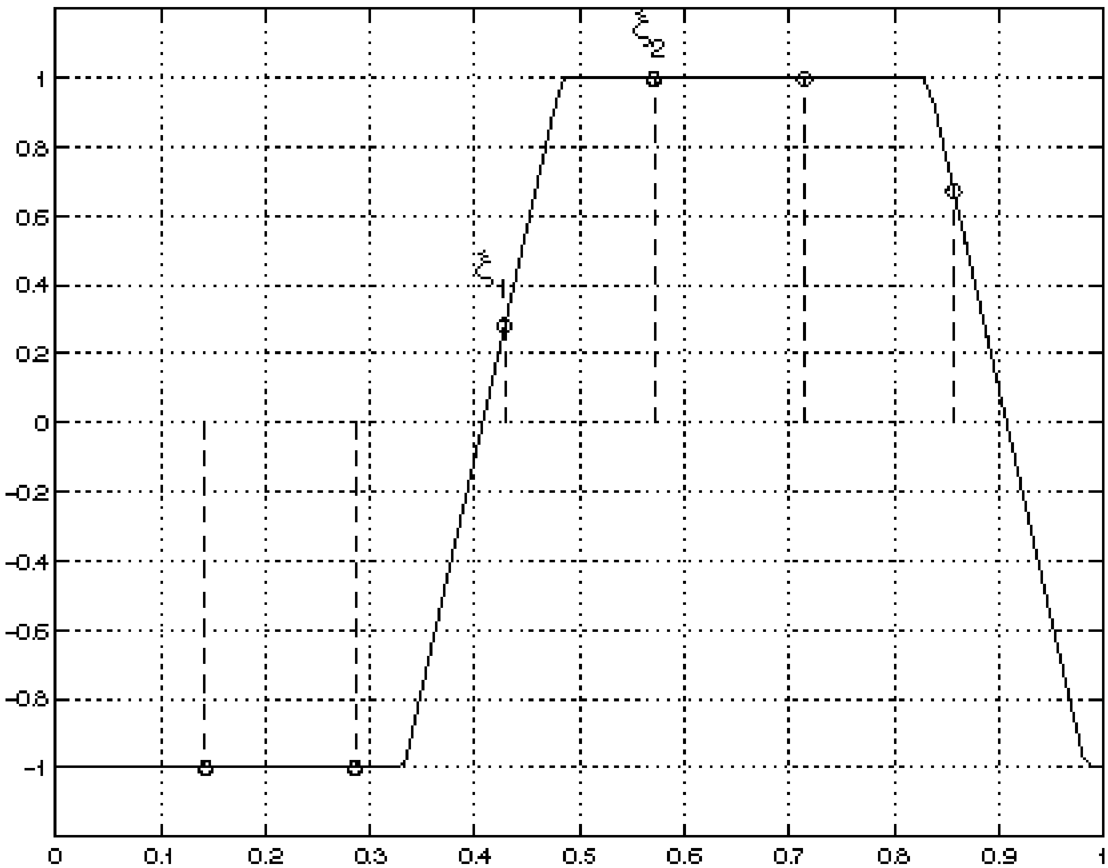
Noise in a circuit that is periodically driven, say with period $T$, exhibits statistical properties that also vary periodically.To understand time-domain characterization of noise, consider the simple circuit shown in Figure 15-1.

**Figure 15-1 Very Simple Mixer Schematic**




The amplitude of the noise measured at the RF output shown in Figure 15-1 periodically varies depending on the magnitude of the modulating signal $p(t)$, as shown by the sample points in Figure 15-2.

**Figure 15-2 Time-Varying Noise Process Analyzed at $\xi_1$ and $\xi_2$**




In Figure 15-2

- The solid line shows the envelope $p(t)$ that modulates the noise process.
- The circles show possible phase points on the envelope where you might calculate the time-varying noise power.
- The circles marked $\xi_1$ and $\xi_2$ indicate the two phase points on the envelope where time-varying noise power is calculated.
- Noise in circuits that are periodically driven, say with period $T$, exhibits statistical properties that also vary periodically. To understand time-domain characterization of noise, consider the simple circuit shown in Figure 15-1 on page 933. The amplitude of the noise measured at the RF output periodically varies depending on the magnitude of the modulating signal $p(t)$, as shown by the sample points (or circles on the signal envelope) in Figure 15-2 on page 934.

- Figure 15-2 is a representation of periodically-modulated noise. It shows noise processes for two different phases in the periodic interval. Each process is stationary.

Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) can calculate the time-varying noise power at any point in the fundamental period. In fact, Spectre RF can calculate the full auto correlation function

$$R^{\xi}(p,q) = < x^{\xi}(p)x^{\xi}(p+q) > = R^{\xi}(q)$$

and its spectrum for the discrete-time processes $x^{\xi}$ obtained by periodically sampling the time-domain noise process at the same point in phase.

Figures 15-3 and 15-4 show two such noise processes for two different phases in the periodic interval. Each process is stationary. Figure 15-3 shows the noise process for the phase marked $\xi_1$ in Figure 15-2 on page 934.

#### Figure 15-3  Noise Process for Phase $\xi_1$

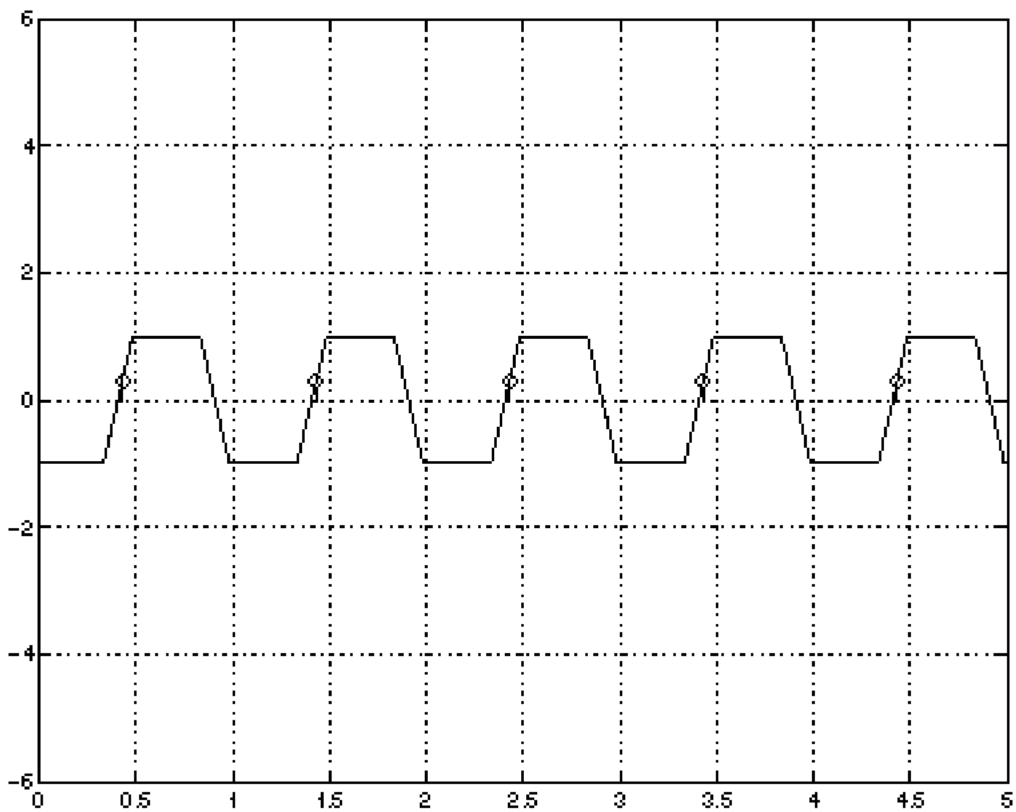
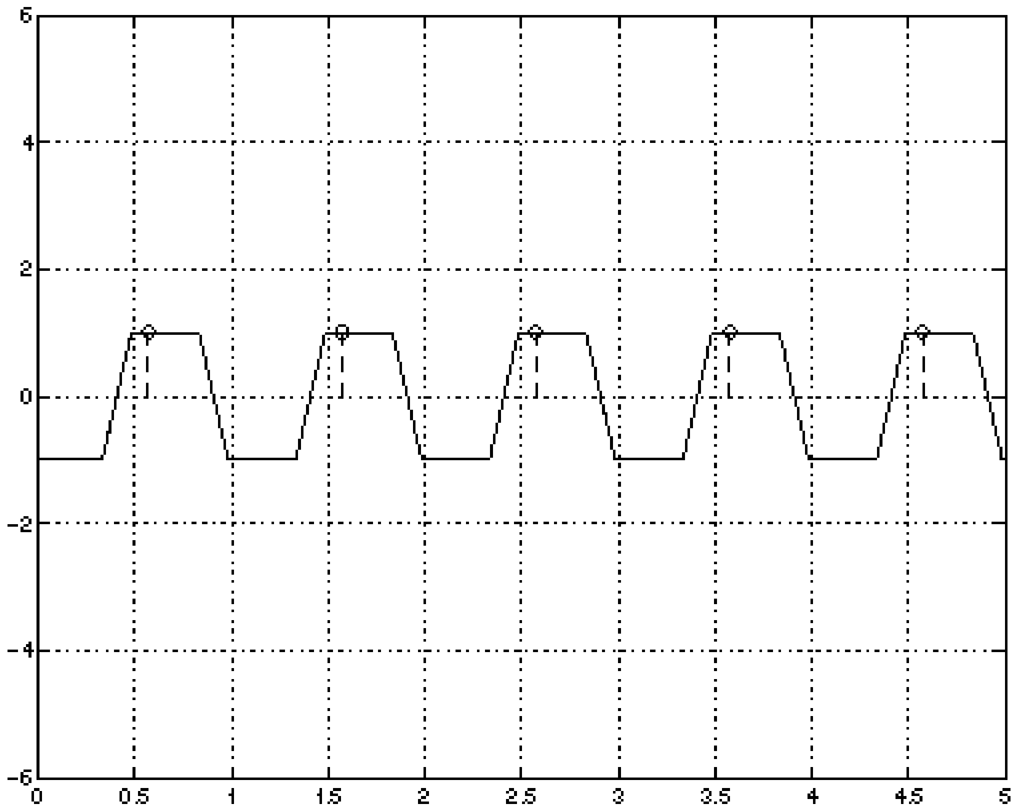


Figure 15-4 shows the noise process for the phase marked $\xi_2$ in Figure 15-2 on page 934.

**Figure 15-4 Noise Process for Phase $\xi_2$**




See the "Reference Information on Time-Varying Noise" on page 948 for a more detailed introduction to noise in periodically time-varying systems.

# Calculating Time Domain Noise

The following steps tell you how to calculate time-domain noise using Spectre RF.

1. In a terminal window, type `icms` to start the environment.
2. In the ADE window, select *Analyses – Choose.*

   The Choosing Analyses form appears.
3. In the Choosing Analyses form, highlight pss and perform the PSS analysis setup.
4. In the Choosing Analyses form, highlight *pnoise.*

**5.** The Choosing Analyses form changes to let you specify information for a Pnoise analysis.

**6.** In the Choosing Analyses form, perform the following:

**a.** Choose Noise Type *timedomain*.

**b.** Specify an appropriate frequency range and sweep for the analysis.

You might, for example, perform a linear sweep up to the fundamental frequency. Because each time point in the calculation is a separate frequency sweep, use the minimum number of frequency points possible to resolve the spectrum. This step minimizes computation time.

**c.** Specify a *noiseskipcount* value or specify additional explicit time points with *noisetimepoints.*

**d.** Specify an appropriate set of time points for the time-domain noise analysis.

**e.** Use *noiseskipcount* to calculate time-domain noise for one of every *noiseskipcount* time points.

If you set *noiseskipcount* to a value greater than or equal to zero, the simulator uses the *noiseskipcount* parameter value and ignores any *numberofpoints* parameter value. When *noiseskipcount* is less than zero, the simulator ignores the *noiseskipcount* parameter. The default is *noiseskipcount = -1.*

You can add specific points by specifying a time relative to the start of the PSS simulation interval. *noiseskipcount = 5* performs noise calculations for about 30 time points in the PSS interval.

If you only need a few time points, add them explicitly with the *noisetimepoints* parameter and set *noiseskipcount* to a large value like 1000.

**7.** In the ADE window, choose *Simulation – Netlist and Run.*

**8.** The simulation runs.

**9.** In the ADE window, choose *Results – Direct Plot – PSS*.

The PSS Results form appears.

**10.** To calculate time-varying noise power, perform the following steps in the PSS Results form:

**a.** Click *tdnoise* and then select *Integrated noise power*.

**b.** Type `0` as the start frequency and the PSS fundamental frequency as the stop period.

For example, type `1G` if the PSS period is 1ns.

A periodic waveform appears that represents the expected noise power at each point in the fundamental period.

**11.** To display the spectrum of the sampled processes, perform the following steps in the PSS Results form:

   **a.** Highlight *Output Noise*.

   **b.** Highlight *Spectrum* for the type of sweep.

   **c.** Clicking on *Plot*.

A set of curves appears, one for each sample phase in the fundamental period.

**12.** To calculate the autocorrelation function for one of the sampled processes, perform the following steps:

   **a.** Display the spectrum using instructions from step 11.

   **b.** In the ADE window, choose *Tools – Calculator*.

   **c.** The calculator appears.

   **d.** Click *wave* in the calculator and the select the appropriate frequency-domain spectrum.

   One of the sample waveforms is brought into the calculator

   **e.** Choose *DFT* from the list of special functions in the calculator. Then set 0 as the *From* and the PSS fundamental as the *To* value.

   **f.** Choose an appropriate window (e.g., Cosine2) and number of samples (around the number of frequency points in the interval [0,1/T]),

   **g.** Apply the *DFT* and plot the results.

Harmonic $q$ of the DFT results gives the value of the discrete autocorrelation for this sample phase, $R(q)$.

Be sure the noise is in the correct units of power (e.g., $V^2$/Hz), not V/square root of Hz) before performing the DFT to obtain the autocorrelation.

# Calculating Noise Correlation Coefficients

To characterize the noise in multi-input/multi-output systems, it is necessary to calculate both the noise power at each port and the correlation between the noise at various ports. The situation is complicated in RF systems because the ports may be at different frequencies. For example, in a mixer, the input port may be at the RF frequency and the output port at the IF frequency.

Denote the power spectrum of a signal $x$ by $S_{XX}(\omega)$, that is

$$S_{XX}(\omega) = X^*(\omega)\, X(\omega)$$

where $X(\omega)$ is the Fourier transform of the signal $x(t)$. For random signals like noise, calculate the expected value of the power spectrum $S_{XX}(\omega)$. To characterize the relationship between two separate signals $x(t)$ and $y(t)$, you also need the cross-power spectrum

$$S_{XY}(\omega) = X^*(\omega)\, Y(\omega)$$

For random signals, the degree to which $x$ and $y$ are related is given by the cross-power spectrum. You can define a correlation coefficient $\rho xy(\omega)$ by

$$\rho XY(\omega) = \frac{S_{XY}(\omega)}{\sqrt{S_{XX}(\omega) S_{YY}(\omega)}}$$

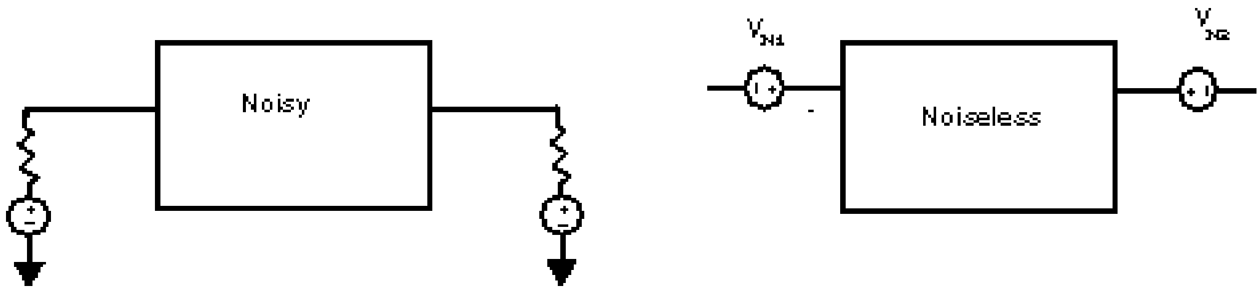
- A correlation coefficient ($\rho xy(\omega)$) of `0` indicates the signals are completely uncorrelated.
- A correlation coefficient of `1` indicates the signals are perfectly correlated. For example, a signal is always perfectly correlated with itself.
- You might also want to consider correlations between noise at different frequencies. The following quantity

$$S^{\alpha}_{XY}(\omega)$$

expresses the correlation of a signal $x$ at frequency $\omega$ with the signal $y$ at frequency $\omega + \alpha$. For example, white Gaussian noise is completely uncorrelated with itself for $a \neq 0$. Noise in an RF system generally has $S^{\alpha}(\omega)$ non-zero when $\alpha$ is the fundamental frequency, for example, the LO frequency in a mixer.

After you have measured the noise properties of a circuit, you can represent the circuit as a noiseless multiport with equivalent noise sources. For example, in Figure 15-5, first you measure the noise voltage appearing at the excitation ports of the circuit on the left in Figure 15-5. Then, you can express the noise properties of the circuit as two equivalent frequency-dependent noise voltages $V_{N1}$ and $V_{N2}$, and a complex correlation coefficient $\rho 12$.

**Figure 15-5 Calculating Noise Correlations and Equivalent Noise Parameters**

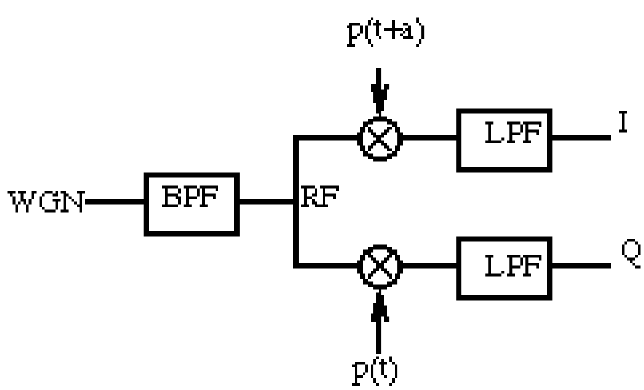



When you know the noise at each port and its correlation, you can obtain any of various sets of equivalent noise parameters. For example, you can express noise in an impedance representation as the equivalent correlated noise voltage sources shown in Figure 15-5, as equivalent noise resistances and the correlation parameters, and as $F_{min}$, $R_N$, $G_{opt}$, and $B_{opt}$.

# Cyclostationary Noise Example

As an example which illustrates the various aspects of cyclostationary noise, consider the simple mixer circuit shown in Figure 15-6.

**Figure 15-6 Simple Mixer Circuit**




In this simple mixer circuit, white Gaussian noise passes through a high-order band-pass filter with center frequency $\omega_0$. Then it is multiplied by two square-waves which have a phase shift $a$ with respect to each other. Finally the output of the ideal multipliers is put through a one-pole low-pass filter to produce $I$ and $Q$ outputs.

The time-domain behavior of the noise is examined first. The most dramatic effect can be seen by looking directly at the mixer outputs in Figure 15-7 on page 941. This figure shows the contributions to the time-varying noise power made by three separate source frequencies. Two of the source frequencies were selected around $\omega_0$, the third source frequency was selected away from $\omega_0$, slightly into the stop band of the band-pass filter. The sharp change in noise power over the simulation interval occurs because the mixers were driven with square-wave LO signals.

#### Figure 15-7  Time-Varying Noise Power Before Low-Pass Filter

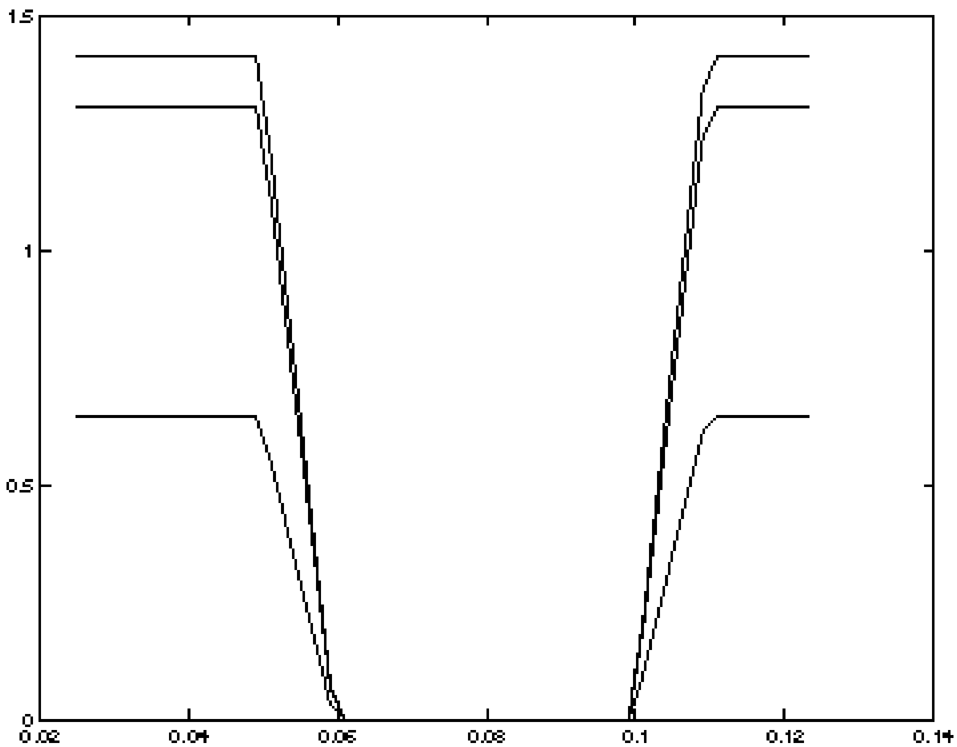
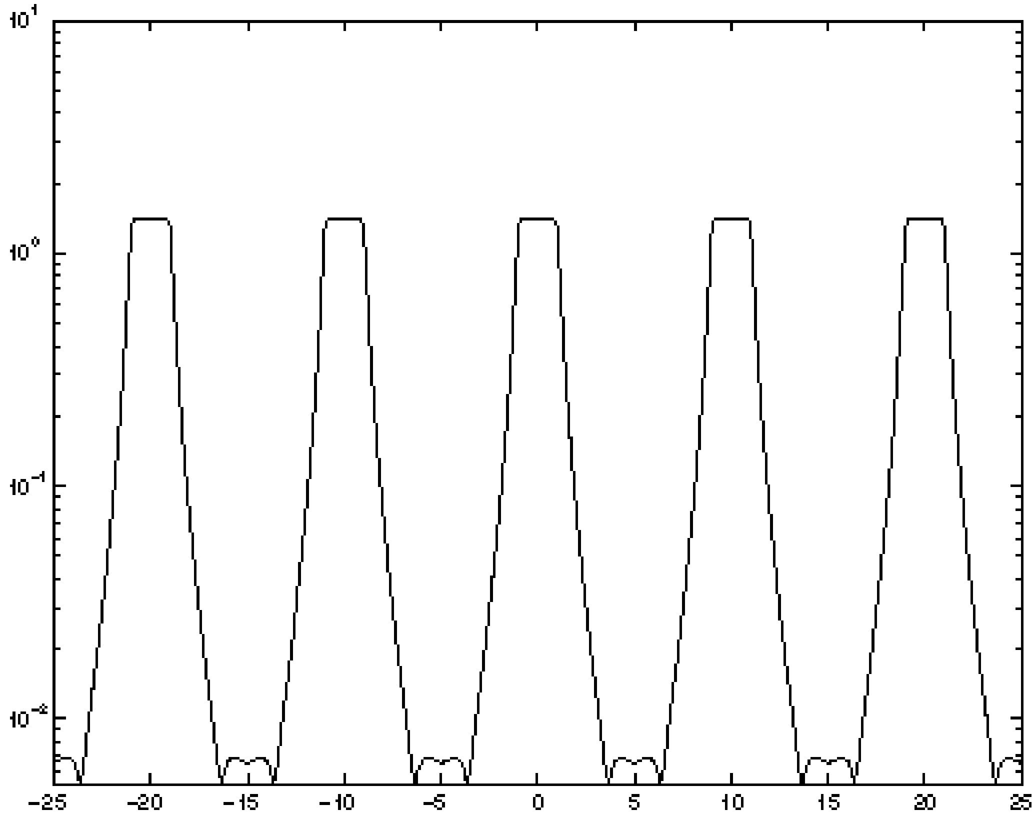


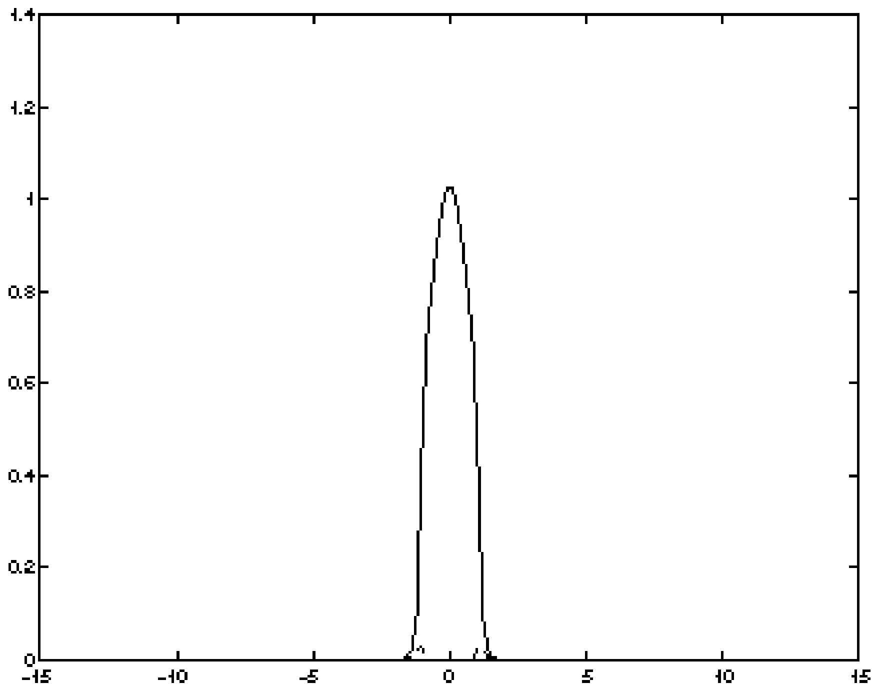
Figure 15-8 shows the spectrum of a sampled noise process. Note the periodically replicated spectrum.

**Figure 15-8  Spectrum of a Sampled Noise Process**




The noise behavior at the output ports is examined next. The output spectra at the $I$ and $Q$ outputs are shown in Figures 15-9 and 15-10. The noise density at $I$ is concentrated around zero because the noise at the RF input to the mixers (band-limited around $\omega_0$) is shifted down to zero and up to $2\omega_0$, but components not around zero are eliminated by the low-pass filter.

### Figure 15-9 Power Spectra With LO Tones $90^{deg}$ Out of Phase




More interesting is the cross-correlation spectrum of the $I$ and $Q$ outputs, shown as the dashed line in Figures 15-9 and 15-10. When the signals applied to the mixers are 90 degrees out of phase (as in Figures 15-9), the cross-power spectral density of the noise at the separate $I$ and $Q$ outputs is small, indicating little noise correlation. If the tones are not quite out of phase (as in Figures 15-10), the correlation is much more pronounced, though in neither case is it completely zero.

In Figures 15-9 and 15-10, the solid and dashed lines represent the following

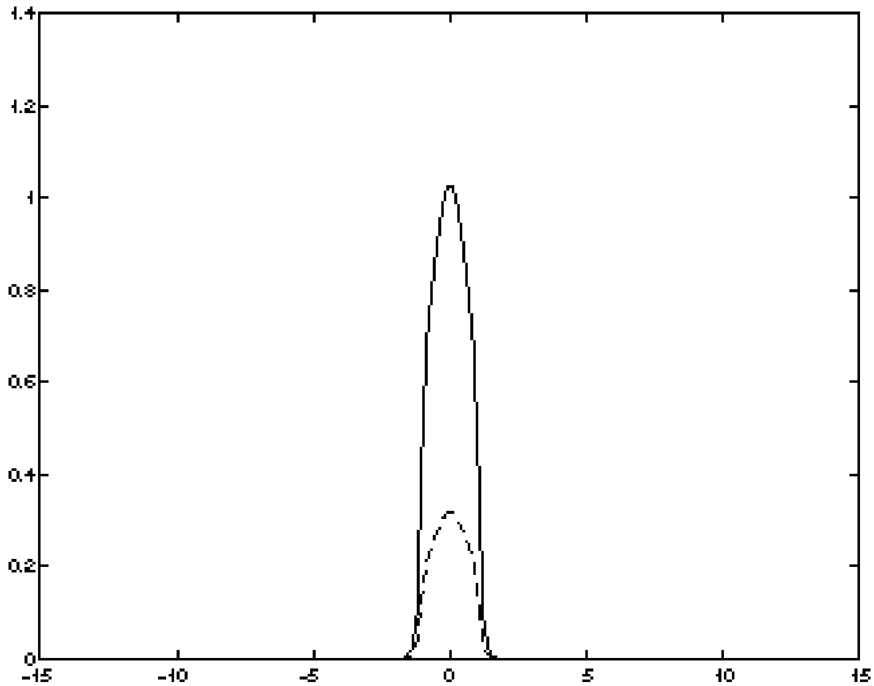
- The solid line represents the power spectrum for the $I$ output with the function

$$S_{II}(\omega)^0$$

- The dashed line represents the cross-spectral density for $I$ and $Q$ with the function

$$S_{IQ}^{0}(\omega)$$

**Figure 15-10 Power Spectra With LO Tones 72$^{deg}$ Out of Phase**

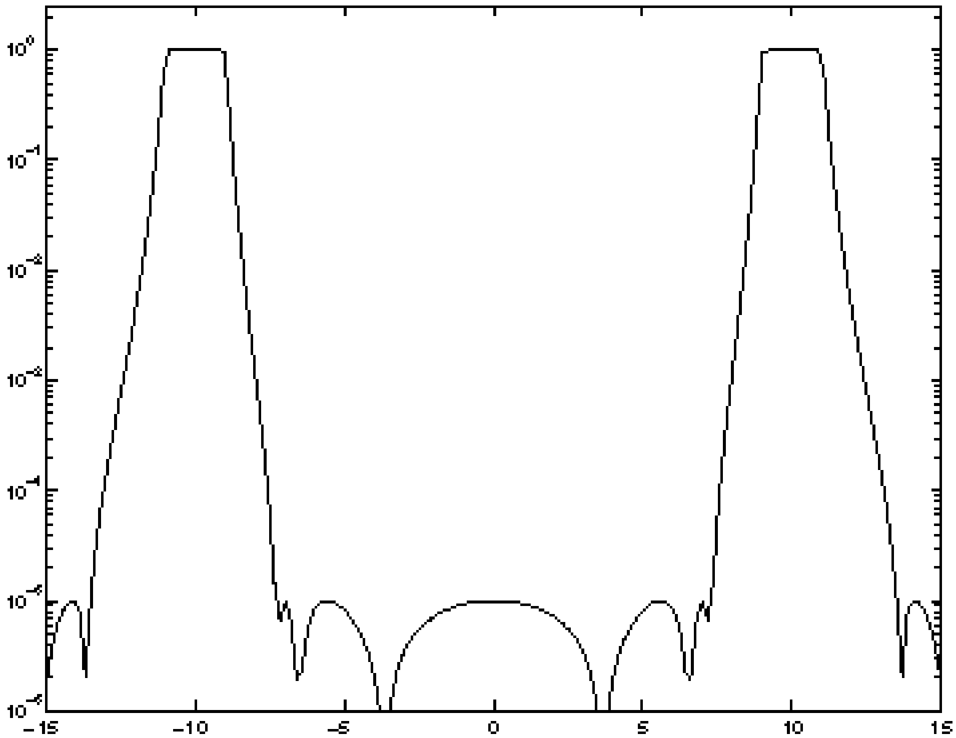



A more interesting example comes from examining the correlation between the noise at the $I$ output and the noise at the RF input. The density function as given by

$$S_{IR}^{(1)}(\omega)$$

is significant because it represents the correlation between the noise at the $I$ output around the baseband frequency with the noise at the RF input, $\omega_0$ higher in frequency. The correlation is high because the noise at the RF input is centered around $\omega_0$ and converted to zero-centered noise by the mixer.

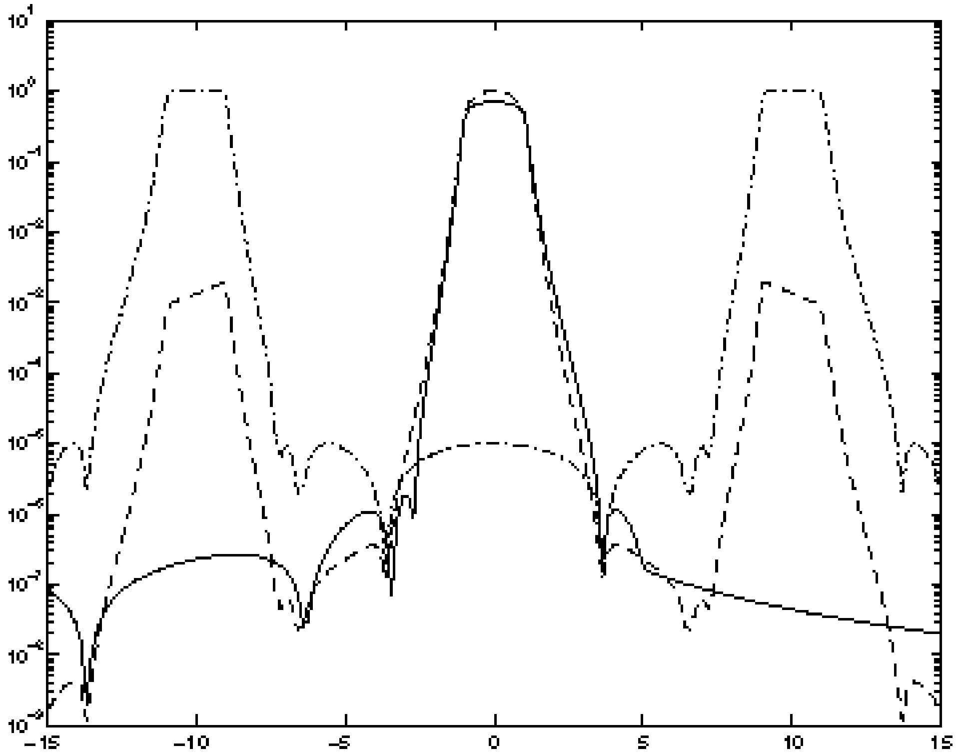
**Figure 15-11  Noise Spectrum at the RF Input**




In Figure 15-11, the noise spectrum at the RF input is given by the following function

$$S_{RR}^{0}(\omega)$$

**Figure 15-12 Noise Spectrum at Various Power Densities**




In Figure 15-12, the solid, dashed, and dashed-dot lines represent the following

- The solid line represents the cross power spectrum which indicates correlation between output noise power at the $I$ output versus noise at the RF input that is one harmonic higher in frequency. This is represented by the following function

$$S_{IR}^{(1)}(\omega)$$

- The dashed line represents the noise spectrum at the $I$ output with the following function

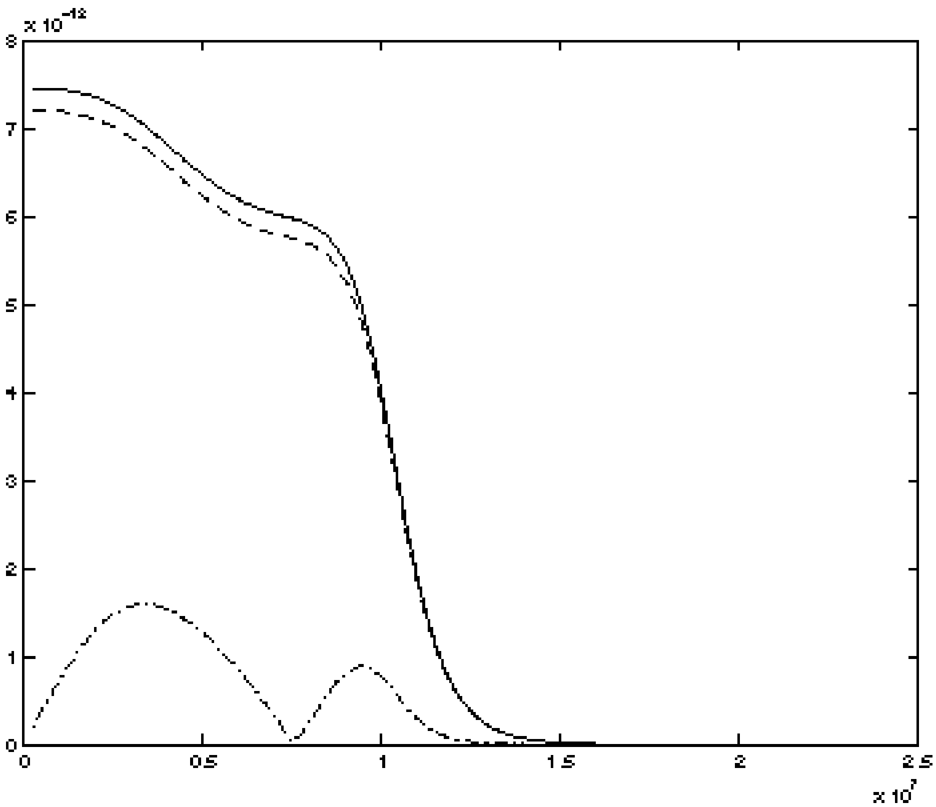
$$S_{II}^{(0)}(\omega)$$

- The dashed-dot line represents the noise spectrum at the RF input with the following function

$S_{RR}^{(0)}$

Finally a detailed circuit example was considered. A transistor-level image-reject receiver with $I$ and $Q$ outputs was analyzed. The noise spectra at the $I$ and $Q$ outputs were found to be very similar, as shown in Figure 15-13.

**Figure 15-13  Power Spectral Densities of an Image-Reject Receiver**

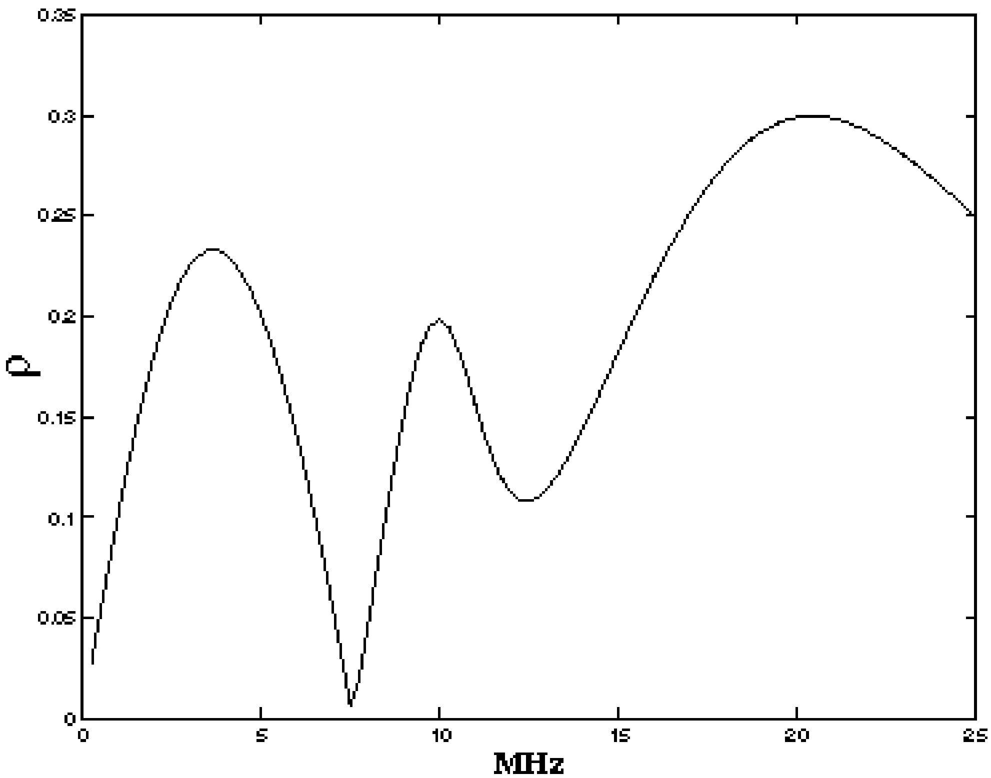



In the image-reject receiver example shown in Figure 15-13, the power spectral densities are represented as follows

- $I$ output is a solid line
- $Q$ output is a dashed line
- $IQ$ cross-power density is a dash-dot line

The $IQ$ cross-power density was smaller, but not negligible, indicating that the noise at the two outputs is partially correlated. The correlation coefficient between noise at the $I$ and $Q$ outputs of the image-reject receiver is shown in Figure 15-14.

**Figure 15-14  Correlation Coefficient for Output Noise of Image-Reject Receiver**




# Reference Information on Time-Varying Noise

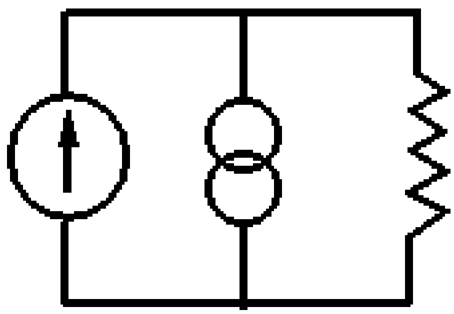
The following sections provide background and reference information on the following noise-related topics

- Thermal Noise
- Linear Systems and Noise
- Time-Varying Systems and the Autocorrelation Function
- Time-Varying Systems and Frequency Correlations
- Time-Varying Noise Power and Sampled Systems

## Thermal Noise

The term *noise* is commonly used to refer to any unwanted signal. In the context of analog circuit simulation, noise is distinguished from such phenomena as distortion in the sense that it is non-deterministic, being generated from *random* events at a microscopic scale. For example, suppose a time-dependent current $i(t)$ is driven through a linear resistor, as shown in Figure 15-15.

### Figure 15-15  Deterministic Current Source Driving a Noisy Linear Resistor

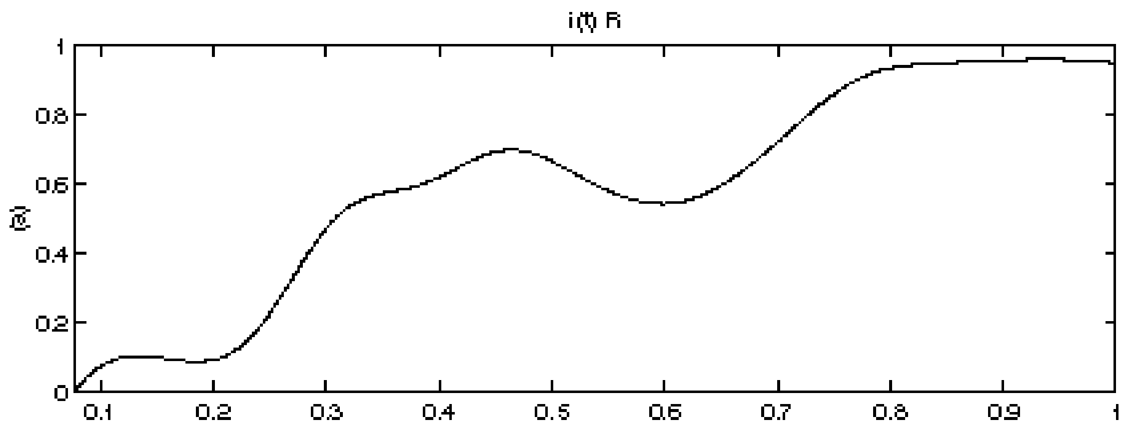


The voltage that appears across the resistor is

$$v(t) = i(t)R + n(t)$$

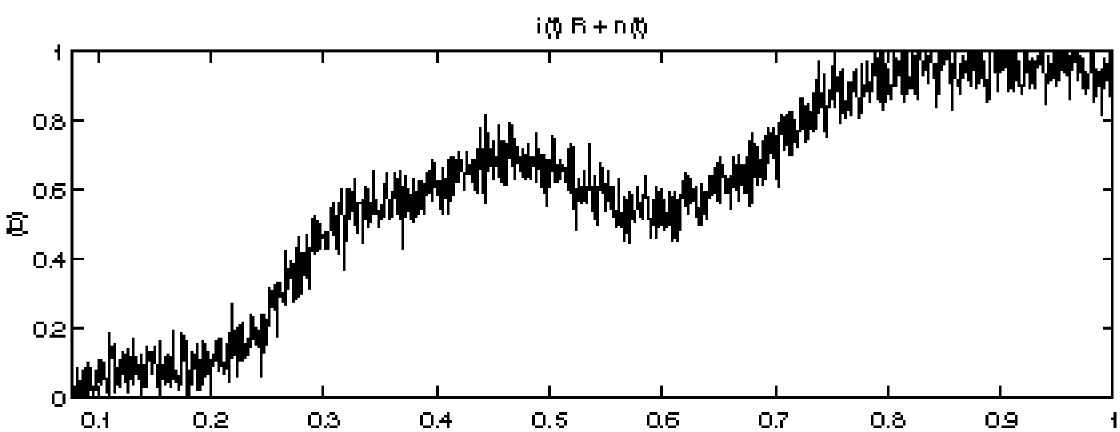
The desired signal $i(t)R$, shown in Figure15-16, is corrupted by an added noise voltage $n(t)$ that is due to resistive thermal noise. The thermal noise of the resistor is modelled by a current source in parallel with the resistor.

**Figure 15-16 The Desired Signal i(t)R**



The total measured voltage is shown in Figure 15-17.

**Figure 15-17 The Total Measured Voltage**

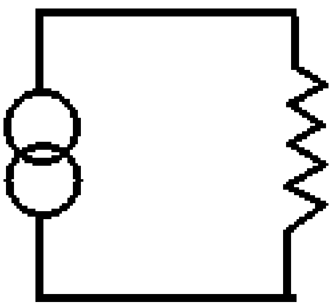


The added noise process alone, $n(t)$, is a random process and so it must be characterized in ways that are different than for deterministic signals. That is, at a time $t_0$ the voltage produced by the driven current can be exactly specified—it is $i_0 sin t_0 R$. Just by inspecting Figure 15-16 we can predict this part of the measured signal.

On the other hand, the exact value of the noise signal cannot be predicted in advance, although it can be measured to be a particular value $n(t_0)$. However, if another measurement is performed, the noise signal $n(t)$ we obtain is different and Figure 15-17 changes. Due to its innate randomness, we must use a statistical means to characterize $n(t)$.

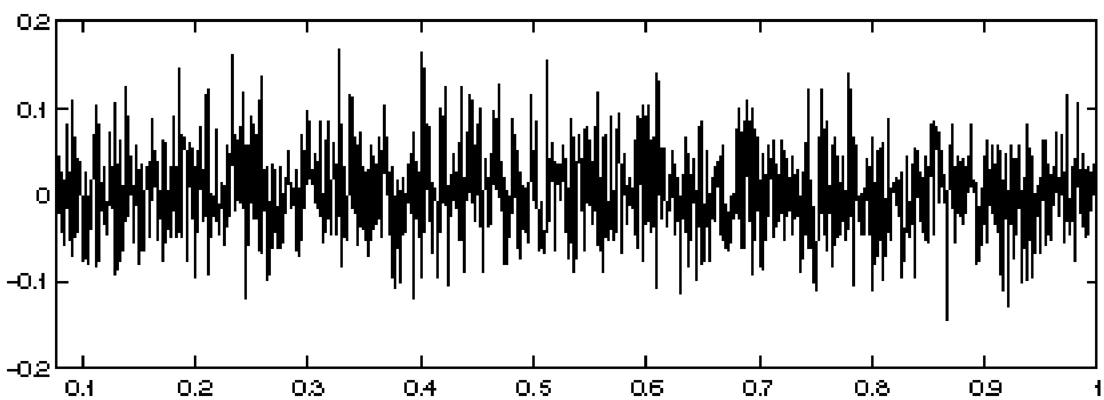
Now consider the circuit in Figure 15-18, where we restrict attention to the noise source/ resistor pair alone.

**Figure 15-18 Resistor Modeled as a Noiseless Resistance with an Equivalent Noise Current Source**



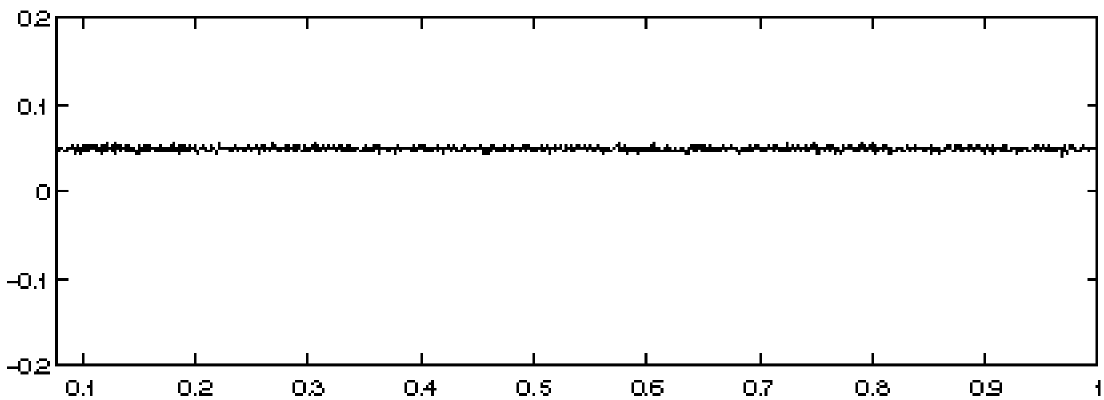
A typical measured noise current/voltage is shown in Figure 15-19.

**Figure 15-19 Typical Measured Noise Current/Voltage**



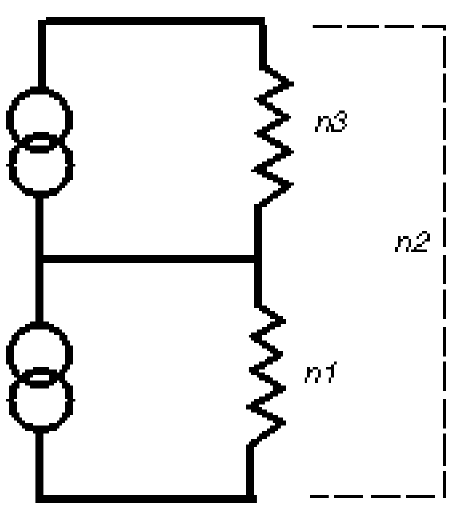

Because we cannot predict the specific value of $n(t)$ at any point, we might instead try to predict what its value would be on average, or what we might *expect* the noise to be. For example, if we measure many noise voltage curves in the time domain, $n(t)$, and average over many different curves, we obtain an approximation to the expected value of $n(t)$ which we denote by $E\{n(t)\}$. For thermal noise, we find that $E\{n(t)\} = 0$. Therefore, instead of computing $E\{n(t)\}$, let us instead compute $E\{n(t)^2\}$, the expected noise power. An example of this sort of measurement is shown in Figure 15-20. 250 measurements were needed to compute this curve.

**Figure 15-20 Expected Noise Power**




Now suppose that we wish to tap the circuit at multiple points. Each point has its own noise characteristics, but they are not necessarily independent. Consider the circuit shown in Figure 15-21.

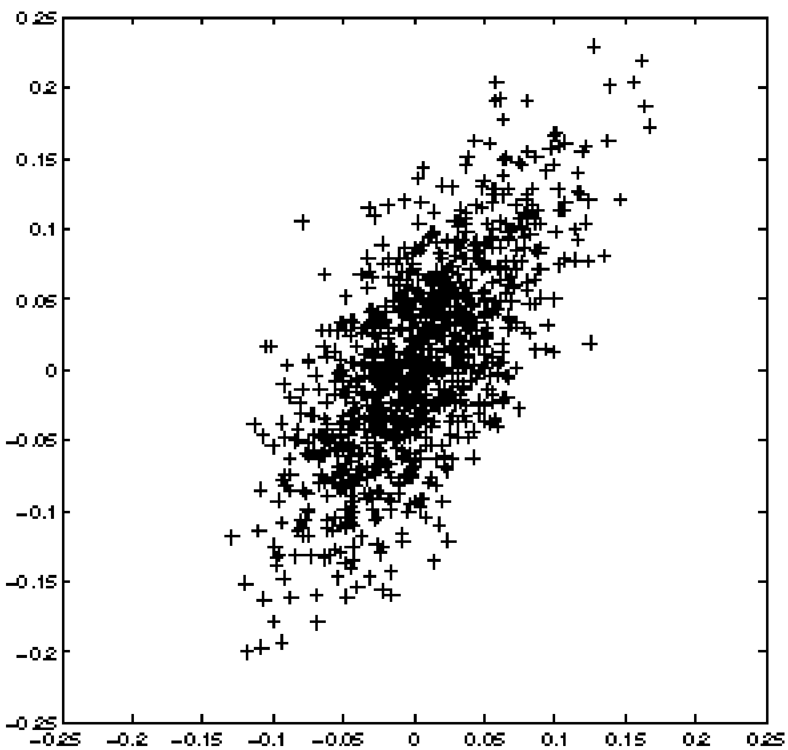
**Figure 15-21 Circuit Illustrating Correlated Noise**



The signals $n_1(t)$ and $n_2(t)$ are obtained by measuring the voltage across a single resistor ($n_1(t)$), and across both resistors ($n_2(t)$), respectively. Just measuring $E\{n_1(t)^2\}$ and $E\{n_2(t)^2\}$ is not enough to predict the behavior of this system, because $n_1(t)$ and $n_2(t)$ are not independent.

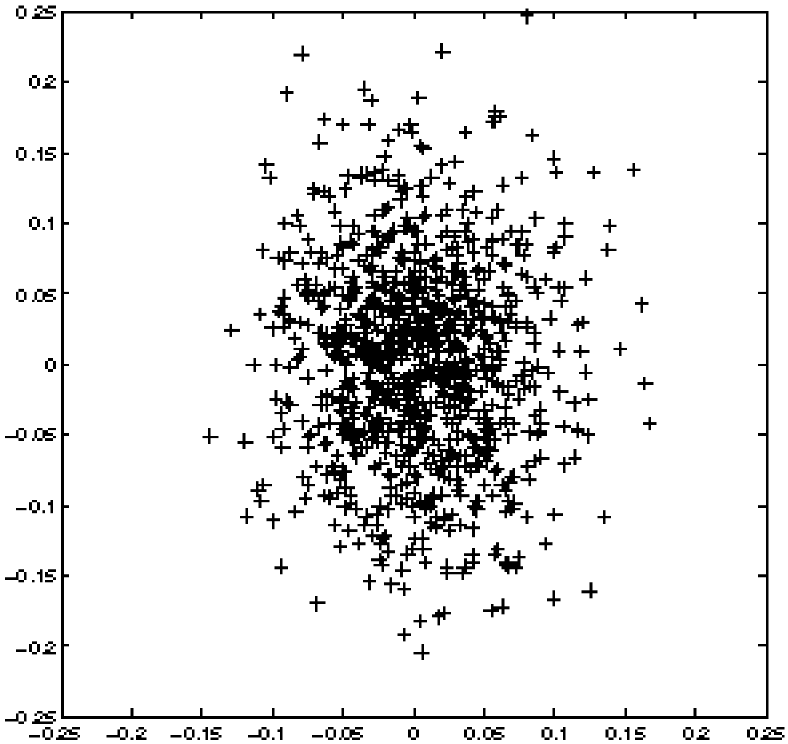
To see $n_1(t)$ and $n_2(t)$ are not independent, consider Figures 15-22 and 15-23. Samples of each of the processes are taken and plotted on an X-Y graph.

#### Figure 15-22 Samples of $n_1(t)$ Plotted Versus $n_2(t)$




Because $n_1(t)$ composes part of $n_2(t)$, $n_1(t)$ and $n_2(t)$ are correlated so in Figure 15-22, the X-Y plot has a characteristic skew along the $X=Y$ line, relative to the $n_1(t)$, $n_3(t)$ plot in Figure 15-23,

**Figure 15-23 Samples of $n_1$(t) Plotted Versus $n_3$(t)**




The signals $n_1(t)$ and $n_3(t)$ are uncorrelated because they represent thermal noise from different sources. The additional measurement needed to describe the random processes is the measurement of the correlation between the two processes, $E\{n_1(t)n_2(t)\}$. We can also define a time-varying correlation coefficient ρ, with $\rho \in [0,1]$, as

$$\rho(t) = \frac{E\{n_1(t)n_2(t)\}}{\sqrt{E\left\{n_1(t)^2\right\}E\left\{n_2(t)^2\right\}}}$$

A value of $\rho = 0$ indicated completely uncorrelated signals, and a value near one indicates a high degree of correlation. In this example we would find that $\rho(t) = 1/2$, representing the fact that each of the two noise sources contributes half of the process $n_2(t)$.

When there are multiple variables of interest in the system, it is convenient to use matrix notation. We write all the random processes of interest in a vector, for example

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

and then we can write the correlations as the expected value of a vector outer product, $E\{x(t)x^H(t)\}$, where the $H$ superscript indicates Hermitian transpose.

For example, we might write a time-varying correlation matrix as

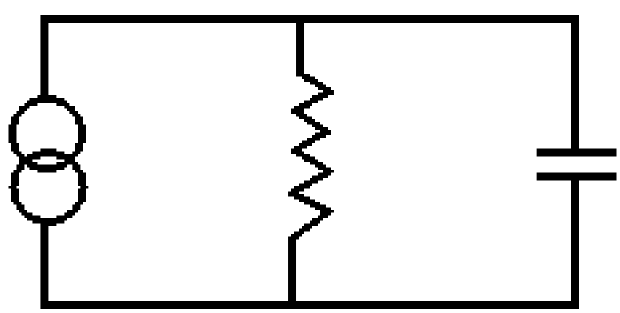
$$R_{xx}(t, t) \equiv E\{x(t)x^H(t)\} = \begin{bmatrix} E\{x_1(t)x_1(t)\} & E\{x_1(t)x_2(t)\} \\ E\{x_2(t)x_1(t)\} & E\{x_2(t)x_2(t)\} \end{bmatrix}$$

## Linear Systems and Noise

The examples in the preceding sections describe how to characterize purely static systems. Now we need to add some elements with memory, such as inductors and capacitors.

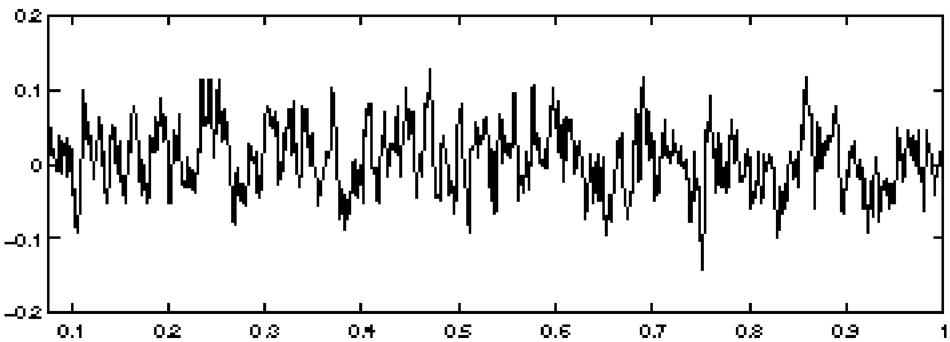
As a first example, consider adding a capacitor in parallel to the simple resistor, as shown in Figure 15-24.

**Figure 15-24 A Simple RC Circuit**



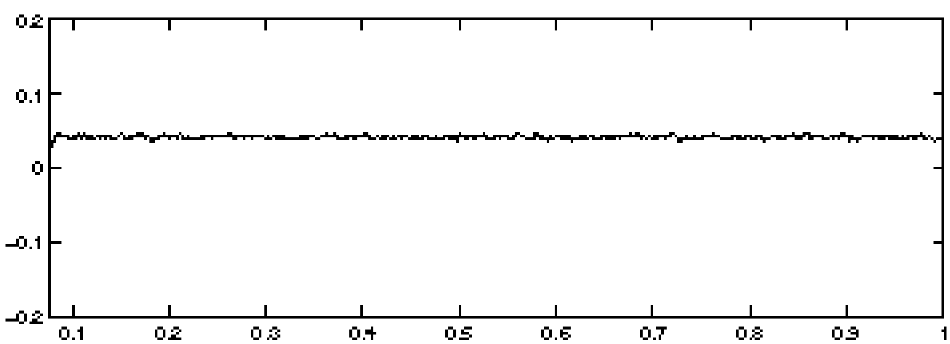
A sample of the noise process in shown in Figure 15-25.

**Figure 15-25 Noise Process for a Simple RC Circuit**




The noise looks different than the noise of the resistor alone, because the low-pass filter action of the RC circuit eliminates very high frequencies in the noise. However, we cannot see this effect simply by measuring $E\{n(t)^2\}$ as shown in Figure 15-26.

**Figure 15-26 Expected Noise Power for an RC Circuit**




The measurement of $E\{n(t)^2\}$ is independent of time for an RC circuit, just as it was for the for the resistor circuit.

## Spectral Densities in Two Simple Circuits

Instead of expected noise power, let us look at the expected power density in the frequency domain. Let $n(\omega)$ denote the Fourier transform of one sample of $n(t)$. Then, $E\{n(\omega)n(\omega)^*\}$ is the expected power spectral density, which we denote by $S_n(\omega)$.

In the present case, the capacitor has a pronounced effect on the spectral density. Figure 15-26 shows a computed power spectral density for the resistor thermal noise previously considered. The spectrum is essentially flat (some deviations occur because a finite number of samples was taken to perform the calculation). The flat spectrum represents the fact that in the resistor's noise, all frequencies are, in some statistical sense, equally present. We call such a process *white noise*.

**Power Spectral Density for Resistor Thermal Noise**

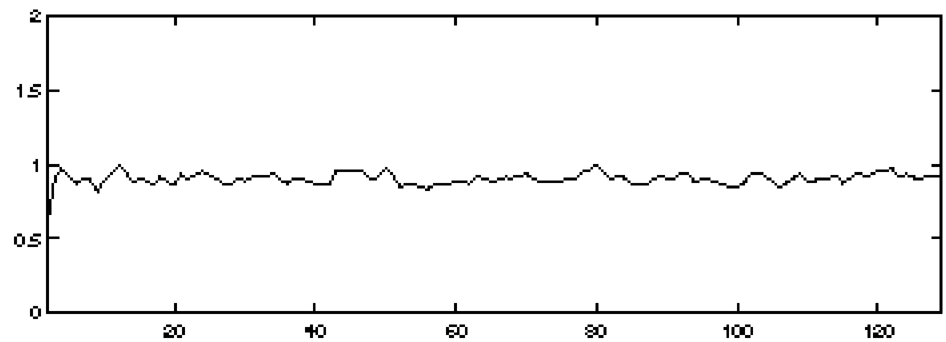
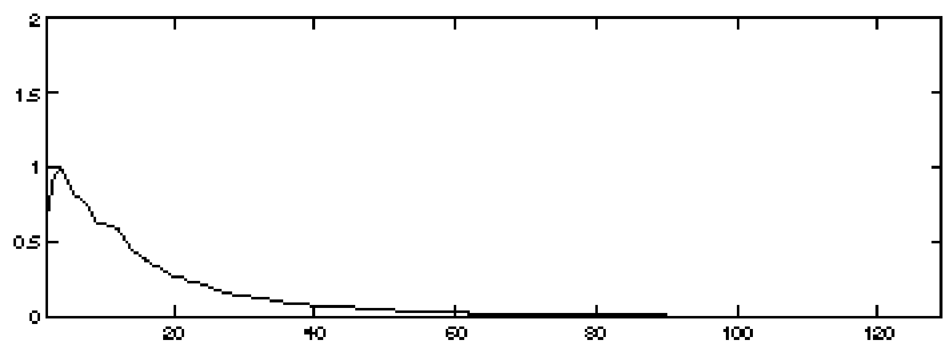



Figure 15-27 shows the spectrum of the noise process after filtering by the resistor-capacitor system.

**Figure 15-27 Resistor-Capacitor Filtered Spectral Noise Process**




It is easy to rigorously account for the effect of the RC-filter on the power spectrum of the noise signal. Suppose a random signal $x$ is passed through a time-invariant linear filter with frequency-domain transfer function $h(\omega)$. Then the output is $y(\omega)=h(\omega)x(\omega)$.

Because expectation is a linear operator, we can easily relate the power spectral density of $y$, $S_y(\omega)$ to $S_x(\omega)$, the power spectral density of $x$, by using the definitions of $y$ and power density. Specifically,

$$S_y(\omega) = E\{y(\omega)y(\omega)^*\} = E\{h(\omega)x(\omega)x(\omega)^*h(\omega)^*\} = |h(\omega)|^2 S_x(\omega)$$

The noise from the resistor can be considered to be generated by a noise current source $i$, with power density

$$S_i(\omega) = \frac{4k_BT}{R}$$

placed in parallel with the resistor. With the capacitor in parallel, the transfer function from the current source to the resistor voltage is just the impedance $Z(\omega)$,

$$h(\omega) = Z(\omega) = \frac{(1/C)}{j\omega + \frac{1}{RC}}$$

and so the noise voltage power density is

$$S_n(\omega) = \frac{\frac{4k_BT}{RC}}{\omega^2 + \left(\frac{1}{RC}\right)^2}$$

Clearly the spectrum is attenuated at high frequencies and reaches a maximum near zero.

For a vector process, we may define a matrix of power-spectral densities,

$$S_{xx}(\omega) \equiv E\left\{x(\omega)x^H(\omega)\right\}$$

The diagonal terms are simple real-valued power densities, and the off-diagonal terms are generally complex-valued cross-power densities between two variables. The cross-power density gives a measure of the correlation between the noise in two separate signals at a specific frequency. We may define a correlation coefficient as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{[S_{x_i}(\omega) S_{x_j}(\omega)]^{1/2}}$$

It is often more useful to examine the correlation coefficient because the cross-power density may be small. As an example, consider a noiseless amplifier. The noise at the input is simply a scaled version of the noise at the output leading to a $\rho=1$, but the cross-power density is much smaller than the output total noise power density if the amplifier has small gain.

In a numerical simulation it is important to compute *only* the correlation coefficient when the diagonal spectral densities are sufficiently large. If one of the power densities in the denominator of the correlation-coefficient definition is very small, then a small numerical error could lead to large errors in the computed coefficient, because of division by a number close to zero.

In the vector case, the transfer function is also a matrix $H(\omega)$, such that $y(\omega)=H(\omega)x(\omega)$ and so the spectral densities at the input and output are related by

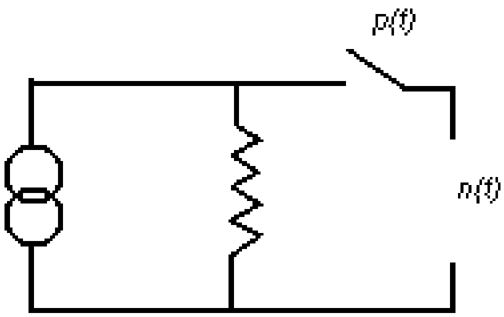
$$S_{yy}(\omega) = E\left\{H(\omega)x(\omega)x^H(\omega)H^H(\omega)\right\} = H(\omega)S_{xx}(\omega)H^H(\omega)$$

## Time-Varying Systems and the Autocorrelation Function

If all the sources of noise in a system are resistors, and the circuit consists strictly of linear time-invariant elements, then the matrix of spectral densities $S_{xx}(\omega)$ is sufficient to describe the noise. However, most interesting RF circuits contain nonlinear elements driven by time-varying signals. This introduces time-varying noise sources as well as time-varying filtering. Because most noise sources are small, and generate small perturbations to the circuit behavior, for purposed of noise analysis, most RF circuits can be effectively modeled as linear time-varying systems. The simple matrix of power spectra is not sufficient to describe these systems.

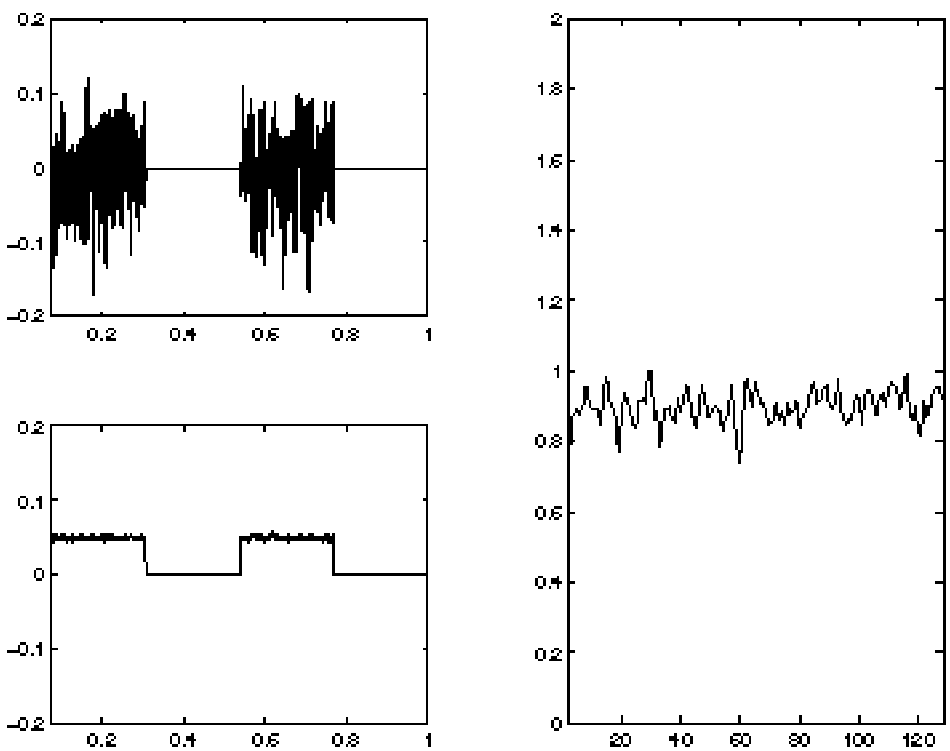
To see this, return to the simple resistor example. Suppose that a switch is connected between the resistor and the voltage measuring device, as shown in Figure 15-28.

**Figure 15-28 SimpleTime-Varying Circuit with Switch**




Further suppose that the switch is periodically opened and closed. When the switch is open, there is no noise measured. When the switch is closed, the thermal noise is seen at the voltage output. A typical noise waveform is shown on the bottom left in Figure 15-29.

**Figure 15-29 Typical Waveforms for Noise, Noise Power and White Noise**



The time-varying noise power $E\{n(t)^2\}$ can be computed and is shown in Figure 15-29 on the top left, above the time-varying noise waveform. The expected power periodically switches between zero and the value expected from the resistor noise. This is different than the resistor-only and resistor-capacitor systems considered previously. Indeed, no linear time-invariant system could create this behavior. However, if we examine the power spectrum on the right in Figure 15-29, we again find that it is flat, corresponding to *white* noise.

### The Autocorrelation Function

At this point it is clear that $E\{n(t)\}$ and $E\{n(t)^2\}$ do not completely specify the random process $n(t)$, nor does the power spectral density. To obtain a complete characterization, consider measuring $n(t)$ at two different timepoints, $t_1$ and $t_2$. $n(t_1)$ and $n(t_2)$ are two separate random variables. They may be independent of each other, but in general they have some correlation. Therefore, to completely specify the statistical characteristics of $n(t_1)$ and $n(t_2)$ together, we must specify not only the variances $E\{n(t_1)^2\}$ and $E\{n(t_2)^2\}$, but also the covariance $E\{n(t_1)n(t_2)\}$. In fact because $n(t)$ has infinite dimension, an infinite number of these correlations must be specified to characterize the entire random process. The usual way of doing this is by defining the autocorrelation function $R_n(t,t+\tau) = E\{n(t)n(t+\tau)\}$.

If $x(t)$ is a vector process,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

then we define the autocorrelation matrix as

$$R_{xx}(t, t+\tau) \equiv E\left\{ x(t)x^H(t+\tau) \right\} = \begin{bmatrix} E\{x_1(t)x_1(t+\tau)\} & E\{x_1(t)x_2(t+\tau)\} \\ E\{x_2(t)x_1(t+\tau)\} & E\{x_2(t)x_2(t+\tau)\} \end{bmatrix}$$

where superscript $H$ indicates Hermitian transpose.

The diagonal term gives the autocorrelation function for a single entry of the vector, e.g, $E\{x_1(t)x_t(t+\tau)\}$. For $\tau=0$, this is the time-varying power in the single process, e.g. $E\{x_1(t)^2\}$. If the process $x(t)$ is Gaussian, it is completely characterized by its autocorrelation function $R_x(t, t+\tau)$ because all the variances and co-variances are now specified.

We can also precisely define what it means for a process to be *time-independent*, or *stationary*—A stationary process is one whose autocorrelation function is a function of $\tau$

only, not of $t$.  This means that not only is the *noise power* $E\{n(t)^2\}$ independent of $t$, but the correlation of the signal at a time point with the signal at another timepoint is only dependent on the difference between the timepoints, $\tau$. The white noise generated by the resistor, and the RC-filtered noise, are both stationary processes.

### Connecting Autocorrelation and Spectral Densities

At different points in the discussion above it was claimed that the expected time-varying power $E\{n(t)^2\}$ of the resistor voltage is constant in time, and also the power density $S_n(\omega)$ is constant in frequency. At first this seems odd because a quantity that is *broad* in time should be *concentrated* in frequency, and vice versa.

The answer comes in the precise relation of the spectral density to the autocorrelation function. Indeed, it turns out that the spectral density is the Fourier transform of the autocorrelation function, but with respect to the variable $\tau$, not with respect to $t$. In other words, the measured spectral density is related to the correlation of a random process with time-shifted versions of itself. Formally, for a stationary process $R_n(t,t+\tau) = R_n(t))$ we write

$$S_n(f) = \int_{-\infty}^{\infty} e^{i\omega\tau} R_n(\tau) d\tau$$

For example, in the resistor-capacitor system considered above, we can calculate the autocorrelation function $R_n(\tau)$ by an inverse Fourier transform of the power spectral density, with the result

$$R_n(\tau) = \left(\frac{4k_BT}{C}\right)e^{-|\tau|/(RC)}$$

From inspecting this expression we can see that what is happening is that adding a capacitor to the system creates memory. The random current process generated by the thermal noise of the resistor has no memory of itself so the currents at separate time-instants are not correlated.  However, if the current source adds a small amount of charge to the capacitor, the charge takes a finite amount of time to discharge through the resistor creating voltage. Thus voltage at a time-instant is correlated with the voltage at some time later, because part of the voltage at the two separated time instants is due to the same bit of added charge. From inspecting the autocorrelation function it is clear that the correlation effects last only as long as the time it takes any particular bit of charge to decay, in other words, a few times the $RC$ time constant of the resistor-capacitor system.

Note that the process is still stationary because this memory effect depends only on how long has elapsed since the bit of charge has been added, or rather how much time the bit of charge has had to dissipate, not the absolute time at which the charge is added. Charge added at separate times is not correlated because arbitrary independent amounts can be added at a given instant. In particular, the time-varying noise power,

$$E\left\{n(t)^2\right\} = \int_{-\infty}^{\infty} S_n(\omega)d\omega$$

## Time-Varying Systems and Frequency Correlations

Now we have seen that the variation of the spectrum in frequency is related to the correlations of the process, in time. We might logically expect that, conversely, variation of the process in time (that is, non-stationarity) might have something to due with correlations of the process in frequency. To see why this might be the case, suppose we could write a random process $x$ as a sum of complex exponentials with random coefficients,

$$x = \sum_{k=-K}^{K} c_k e^{i\omega t}$$

Noting that $c_{-k}=c_k*$, the time-varying power in the process is

$$E\left\{x^2(t)\right\} = \sum_{k=-K}^{K}\sum_{l=-K}^{K} E\{c_k c*_l\}e^{i(w_k - w_l)t}$$

and it is clear that $E\{x(t)^2\}$ is constant in time if and only if

$$E\{c_k c*_L\} = |c_k|^2 \varsigma_{kl}$$

In other words, the coefficients of expansion of sinusoids of different frequencies must be uncorrelated. In general, a stationary process is one whose frequency-domain representation contains no correlations across different frequencies.

To see how frequency correlations might come about, let us return to the resistor-switch example. Let $n(t)$ denote the voltage noise on the resistor, and $h(t)$ the action of the switch, so that the measure voltage is given by $v(t) = h(t)n(t)$, where $h(t)$ is periodic with period $T$ and frequency

$$\omega_0 = \frac{2\pi}{T}$$

The time-domain multiplication of the switch becomes a convolution in the frequency domain,

$$v(\omega) = h(\omega) \otimes n(\omega)$$

where ⊗ denotes convolution.

Because $h(t)$ is periodic, its frequency-domain representation is a series of Dirac deltas,

$$h(\omega) = \sum_k h_k \varsigma(\omega - k\omega_0)$$

and so

$$v(\omega) = \sum_k h_k n(\omega - k\omega_0)$$

and the spectral power density is simply

$$S_v(\omega) = E\{\langle v(\omega)v(\omega)^*\rangle\} = \sum_k \sum_l h_k h_l^* E\{n(\omega - k\omega_0)n(\omega - l\omega_0)^*\}$$

Because the process $n$ is stationary, this reduces to

$$S_v(\omega) = \sum_k |h_k|^2 S_n(\omega - k\omega_0)$$

Because $S_n(\omega)$ is constant in frequency, $S_v(\omega)$ is also.

However, the process $v$ is no longer stationary because frequencies separated by multiples of $\omega_0$ have been correlated by the action of the time-varying switch. We may see this effect in the time-variation of the noise power, as in Figure 15-29 on page 960, or we may examine the correlations directly in the frequency domain.

To do this, we introduce the *cycle spectra*

$$S^{\alpha}_{xx}(\omega)$$

that are defined by

$$S^{\alpha}_{xx}(\omega) = E\left\{x(\omega)x^{H}(\omega+\alpha)\right\}$$

and are a sort of cross-spectral density, taken between two separate frequencies. $S_0(\omega)$ is just the power spectral density we have previously discussed. In fact we can define a frequency-correlation coefficient as

$$\rho^{\alpha}_{n}(\omega) \equiv \frac{S_{n}(\omega)^{\alpha}}{\sqrt{S_{n}(\omega)S_{n}(\omega+\alpha)}}$$

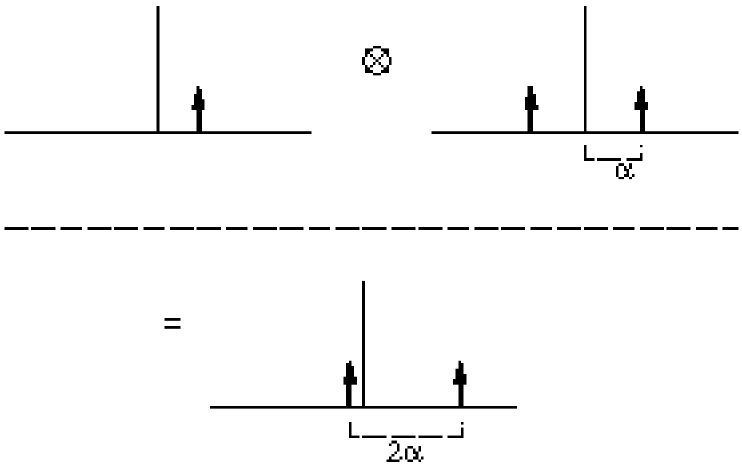
and if

$$\rho^{\alpha}_{n}(\omega) = 1$$

then the process $n$ has frequency content at $\omega$ and $\omega + \alpha$ that is perfectly correlated.

Consider separating out a single frequency component of a random process and multiplying by a sinusoidal waveform of frequency $\alpha$, as shown in Figure 15-30. The component at $\omega$ is shifted to re-appear at $\omega + \alpha$ and $\omega - \alpha$. The new process' frequency components at $\omega - \alpha$ and $\omega + \alpha$ are deterministically related to the components of the old process located at $\omega$. Therefore, they are correlated, and $S^{2a}(\omega)$ is non-zero.

**Figure 15-30 Time-Variation Introduces Frequency Correlation**




Physically, what happens is that to form a waveform with a defined *shape* in time, the different frequency components of the signal must add in a coherent, or correlated fashion. In a process like thermal noise, the Fourier coefficients at different frequencies have phase that is randomly distributed with respect to each other, and the Fourier components can only add incoherently. Their powers add, rather than their amplitudes. Frequency correlation and time-variation of statistics are thus seen to be equivalent concepts.

Another way of viewing the cycle spectra is that they represent, in a sense, the two-dimensional Fourier transform of the autocorrelation function, and are therefore just another way of expressing the statistics of the process.

### Time-Varying Noise Power and Sampled Systems

Again supposing the signal $n$ to be cyclostationary with period $T$, for each sample phase $\xi \in [0,T)$, we may define the discrete-time autocorrelation function

$$R_n^{\xi}(p, q)$$

to be

$$R_n^{\xi}(p, p+q) = R_n(\xi + pT, \xi + (p+q)T)$$

Because the cyclostationary process $R_n$ is periodic, by inspection

$$R_n^{\xi}(p, p+q)$$

is independent of $p$ and thus stationary, that is

$$R_n^{\xi}(p, p+q) = R_n^{\xi}(q)$$

Note that

$$R_n^{\xi}(p, p) = R^{\xi}(0)$$

gives the expected noise power, $R_n(\xi,\xi)$, for the signal at phase $\xi$. Plotting $R^{\xi}(0)$ versus $\xi$ shows how the noise power varies periodically with time.

The discrete-time process

$$R_n^{\xi}(p, p+q) = R_n^{\xi}(q)$$

can be described in the frequency-domain by its discrete Fourier transform,

$$R_n^{\xi}(\phi) = \sum_{q=-\infty}^{\infty} R^{\xi}(q)e^{iq2\pi\phi T}$$

Note that the spectrum of the discrete (sampled) process

$$R_n^{\xi}(\phi)$$

is periodic in frequency with period $1/T$.

All noise power is aliased into the Nyquist interval $[-1/2T, 1/2T]$ (or, equivalently, the interval $[0,1/T]$). Generally it is the noise spectrum which is available from the circuit simulator. To obtain the autocorrelation function or time-varying noise power, an inverse Fourier integral must be calculated by

$$R_n^{\xi}(q) = \int_0^{1/T} R_n^{\xi}(\phi) e^{iq2\pi\phi} d\phi$$

# Summary

- All useful noise metrics can be interpreted in terms of correlations. Physically these can be interpreted as the expected value of two-term products. In the case of random vectors these are expected values of vector outer products.
- The power spectral density of a variable indexed $i$ is

  $$S_{x_i x_i}(\omega) = E\{x_i(\omega)x_i(\omega)^*\}$$

  This is what the current Spectre RF pnoise analysis computes.

  $S_{xx}(\omega)$ is constant if and only if $x$ is a white noise process. In that case $R_{xx}(\tau) = R\delta(\tau)$ if there are no correlations in time for the process.

  The cross-power densities of two variables $x_i$ and $x_j$ are

  $$S_{x_i x_j}(\omega) = E\left\{x_i(\omega)x_j(\omega)^H\right\}$$

  If and only if the two variables have zero correlation at that frequency, then

  $$S_{x_i x_j} = 0$$

A correlation coefficient may be defined as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{\sqrt{S_{x_i}(\omega) S_{x_j}(\omega)}}$$

and $\rho_{ij}(f) \in [0,1]$.

The cycle-spectra

$$S_{xx}^{\alpha}(f)$$

represent correlations between frequencies separated by the cycle-frequency $\alpha$

$$S_{xx}^{\alpha}(f) = E\left\{ x(\omega) x^{H}(\omega + \alpha) \right\}$$

For a single process $x_i$, a correlation coefficient may be defined as

$$\rho_{x_i}^{\alpha}(\omega) \equiv \frac{S_{x_i x_i}(\omega)^{\alpha}}{\sqrt{S_{x_i}(\omega) S_{x_i}(\omega + \alpha)}}$$

and

$$\rho_{x_i}^{\alpha}(f) \in [0, 1]$$

- A process is stationary if and only if

$$S_{xx}^{\alpha}(\omega) = 0$$

for all $\omega$ and all $\alpha \neq 0$, that is, if there are no correlations in frequency for the process.

In other words,

$$S_{xx}^{\alpha}(\omega) = S_{xx}(\omega)\delta(\alpha)$$

- A process is cyclostationary if

$$S_{xx}^{\alpha} = 0$$

for all $\alpha \neq m\omega_0$ for some $\omega_0$ and integer $m$. Frequencies separated by $m\omega_0$ are correlated. A stationary process passed through a periodically linear-time varying filter in general is cyclostationary with $\omega_0$ the fundamental harmonic of the filter.

We might also compute correlations between different nodes at different frequencies, with the obvious interpretation and generalization of the correlation coefficients.

# 16

# Noise-Aware PLL Flow

This chapter includes the following sections.

# Introduction

Phase-locked loops (PLL) are at the core of some important RF designs such as frequency synthesis, clock and data recovery, and clock de-skew. Some of the PLL characteristics, such as phase noise, jitter, power supply and substrate noise interference, step response, acquisition time, and static phase offset are of interest to most designers, but to measure these characteristics, the simulator must overcome the following obstacles:

- The large difference in the time-constant between the VCO and other PLL blocks

  The VCO can operate in the range of GHz; the PFD and CP in the range of MHz; the output of LPF and control of VCO in the range of KHz. Transient analysis forces all the PLL blocks to use the same small time-steps required by the VCO frequency, which makes the analysis very time-consuming.

- The VCO is an autonomous circuit while the other blocks are driven circuits

  PSS and QPSS analyses cannot simulate this kind of circuit.

- The PLL generates repetitive switching events as an essential part of its operation, and the noise performance must be evaluated in the presence of this large-signal behavior

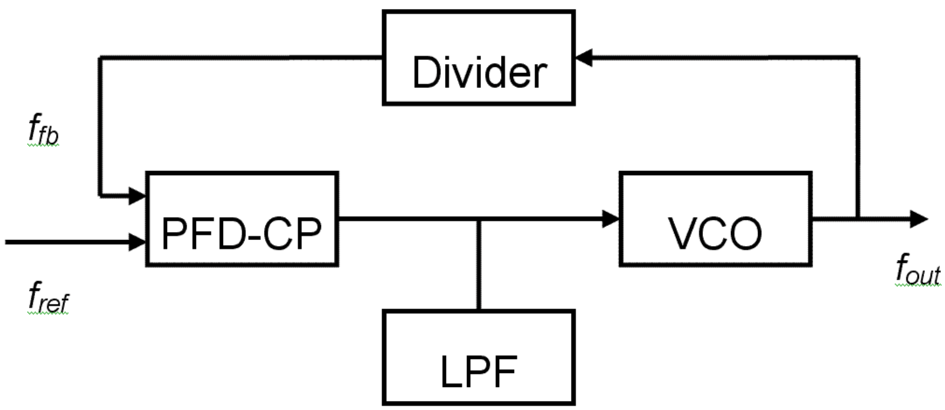
  Transient noise analysis is very time-consuming and neither PNOISE nor QPNOISE analyses can be used.

The Spectre RF noise-aware PLL flow provides a solution to these challenges. This flow uses a macro-model based simulation methodology, in which the PFD-CP, VCO, and divider in the PLL are replaced by automatically generated macro-models that characterize the behavior of the original blocks. The VCO and the divider are integrated into one macro-model, greatly improving the simulation efficiency. In addition, both integer N and fractional-N PLLs can be simulated.

# Preparation

When a PLL is used as a frequency synthesizer, the implementation usually consists of four main blocks: PFD-CP, LPF, VCO, and divider. Figure 16-1 on page 973 is an example of a synthesizer schematic that includes these blocks.

**Figure 16-1 Block diagram of a frequency synthesizer**




The primary steps in setting up and using the noise-aware PLL flow are:

1. Preparing the PFD-CP, VCO, divider, and LPF subcircuits.
2. Inserting the PFD-CP into the PFD-CP test bench and running PSS and PNOISE with the PFD-CP extraction plugin, which automatically generates a Verilog-A model.
3. Inserting the VCO into the VCO test bench and running PSS and PNOISE with the VCO extraction plugin, which automatically generates a `.mat` file defining a VCO CMI macro-model.
4. Simulating the divider with PSS and PNOISE, which generates a Verilog-A model for the divider jitter.
5. Inserting the VCO CMI model, the PFD-CP Verilog-A module, the divider jitter Verilog-A model, and the LPF subcircuit into the PLL macro-model test bench and running a TRAN analysis with the phase noise PSD plugin.

**Note:** Be aware of the following information.

- The test benches mentioned above are included in `pllMMLib`.
- The divider's large signal behavior is merged into the VCO.
- To use the plugin, ADE needs to run in batch mode. This option can be set in the `.cdsinit` file by using

```
envSetVal("spectre.envOpts" "controlMode" 'string "batch")
```

- The command for running the Spectre simulator with a plugin is

```
spectre  -plugin  *.so
```

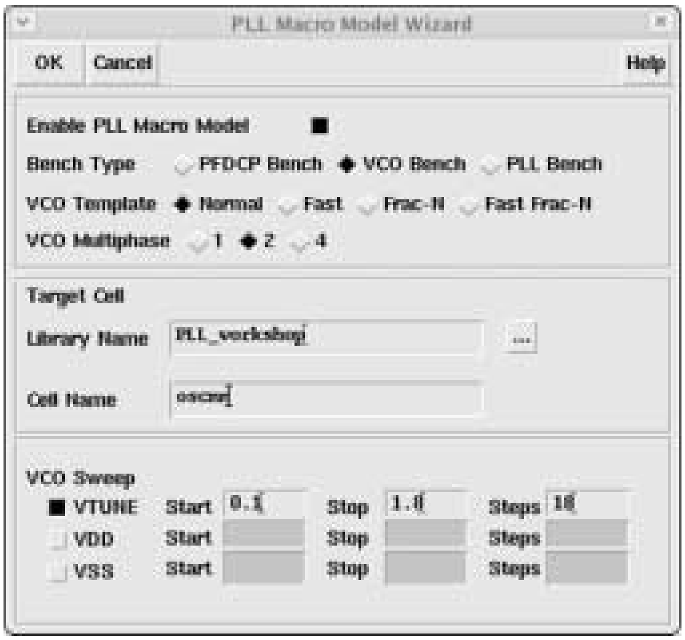
The following plugin files (`.so`) and CMI lib file are needed:

- `libpllPPVoscModel_sh.so` (for VCO macro-model extraction)
- `libpllTTpfd_cpModel_sh.so` (for PFD-CP macro-model extraction)
- `libpllDivider_sh.so` ( for divider macro-model extraction )
- `libpllMMpsd_sh.so` (for phase noise PSD calculation)
- `libpll_sh.so` (VCO macro-model CMI lib)

# Using the Noise-Aware PLL flow in ADE

The above mentioned step 2 (PFD-CP extraction), step 3 (VCO extraction), and step 5 (PLL simulation with macro-models) are integrated into ADE as a wizard. To open the wizard, in the Analog Design Environment (ADE), choose *Tools – RF – Wizard – PLL*.

**Figure 16-2 PLL Macro Model Wizard**



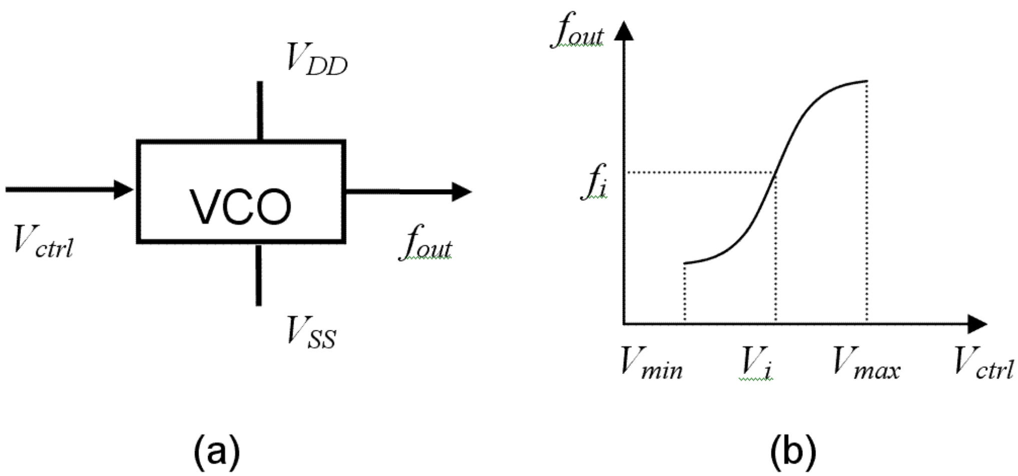

To use the flow, the *Enable PLL Macro Model* option must be selected. For PFD-CP extraction, divider extraction or VCO extraction, a library name and cell name must be specified to indicate where the corresponding macro-model is placed after it is generated. For PLL simulation, *Bench Type* should be *PLL Bench*.

# VCO Extraction

### Character of the Large Signal

A VCO is an oscillator whose output frequency ($f_{out}$) can be tuned by an input control voltage ($V_{ctrl}$). The relationship between the output frequency and the input voltage is known as the tuning curve. A simple VCO schematic (a) and its tuning curve (b) are shown in Figure 16-3 on page 976. In the tuning curve beyond $V_{min}$ and $V_{max}$, the VCO output frequency changes little with $V_{ctrl}$ change. $V_{DD}$ and $V_{SS}$ variation can also have an impact on the output frequency.

**Figure 16-3 Simple VCO and tuning curve**




Based on the VCO tuning curve, a linear table VCO model can be built, which is

**(16-1)** $$f_{out} = K_i \cdot v(t) + f_i$$

where $K_i$ is the VCO gain around the control voltage $V_i$ and $v(t)$ is the control voltage deviation from $V_i$. From Equation 16-1 on page 976, the VCO output phase can be obtained as

**(16-2)** $$\phi_{out}(t) = 2\pi f_i t + \int K_i v(t) dt_i$$

In the Spectre RF noise-aware PLL flow, the VCO macro-model is a nonlinear table model based on perturbation projection vector (PPV) theory. In this model, the VCO output phase is determined by

**(16-3)** $$\phi_{out}(t) = 2\pi f_i t + 2\pi f_i a(t)$$

**(16-4)** $$\frac{d\alpha(t)}{dt} = ppv^T \cdot n(t)$$

where $n(t)$ represents noise sources at every node and branch. The PPV can be output using Spectre RF PSS or PNOISE analysis. The nonlinear model can be used to characterize the following VCO effects that are difficult challenges for the linear model.

- Injection locking
- Phase noise due to supply and substrate noise

### VCO Internal Noise

Generally the internal noise sources consist of white and flicker noise and have the form

**(16-5)** $$S_u(\Delta f) \sim a + \frac{b}{\Delta f}$$

where $\Delta f$ is the offset frequency, $a$ is white noise and another item represents flicker noise. Under noise perturbation, the VCO output phase noise takes the form

**(16-6)** $$S_\phi(\Delta f) \sim \frac{a}{\Delta f^2} + \frac{b}{\Delta f^3}$$

### VCO Macro-Model

The VCO macro-model is a table model that is obtained by sweeping the input control voltage $V_{ctrl}$, $V_{DD}$ and $V_{SS}$. Its main content is shown in Table 16-1 on page 978. $V_{out}$ is the VCO output waveform. The VCO macro-model looks up values in the table according to the input control voltage, $V_{DD}$ and $V_{SS}$ and calculates the output based on Equation 16-3 on page 977 and Equation 16-6 on page 977.

**Table 16-1**

| | | | | |
|---|---|---|---|---|
| $V_{ctrl}[1]$ | $f_{out}[1]$ | $V_{out}[1]$ | PPV[1] | $S_{\phi}[1]$ |
| ... | ... | ... | ... | ... |
| $V_{ctrl}[i]$ | $f_{out}[i]$ | $V_{out}[i]$ | PPV[*i*] | $S_{\phi}[i]$ |
| ... | ... | ... | ... | ... |
| $V_{ctrl}[n]$ | $f_{out}[n]$ | $V_{out}[n]$ | PPV[*n*] | $S_{\phi}[n]$ |

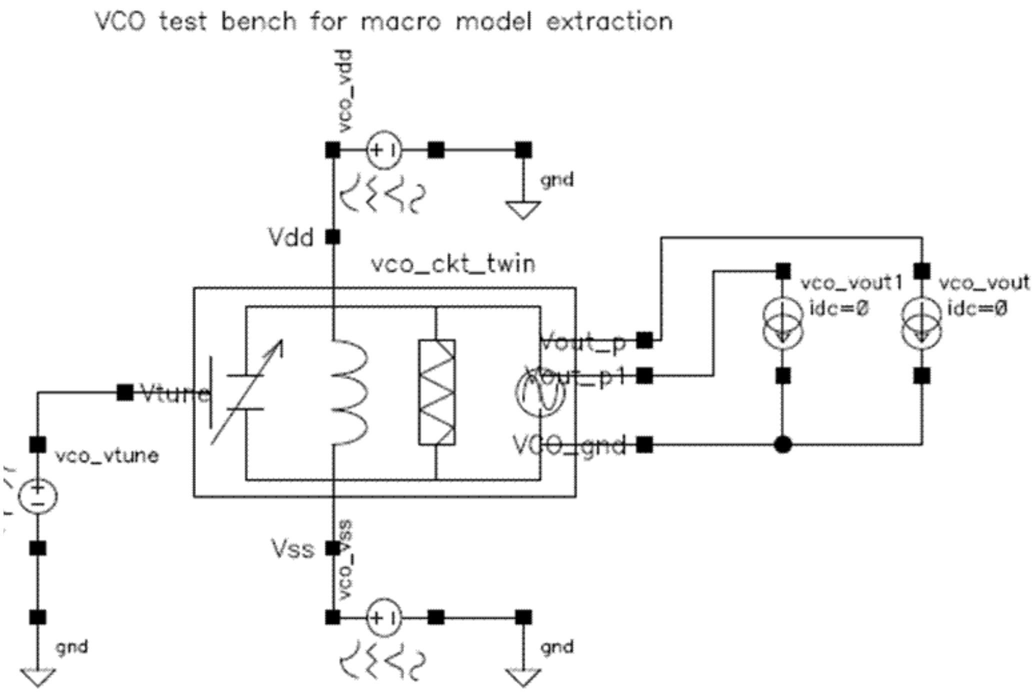
## Extracting the VCO Macro-Model

To extract the VCO macro-model,

1. Insert the VCO transistor-level circuit into a VCO extraction test bench from the `pllMMLib` library.
   - For a VCO with one output terminal, use `vco_bench`.
   - For a VCO with two outputs, use `vco_twin_bench`.
   - For a VCO with four outputs, use `vco_tet_bench`.

   Figure 16-3 on page 979 is the schematic of the test bench for a VCO with two outputs.

**Figure 16-4 VCO test bench for macro-model extraction**



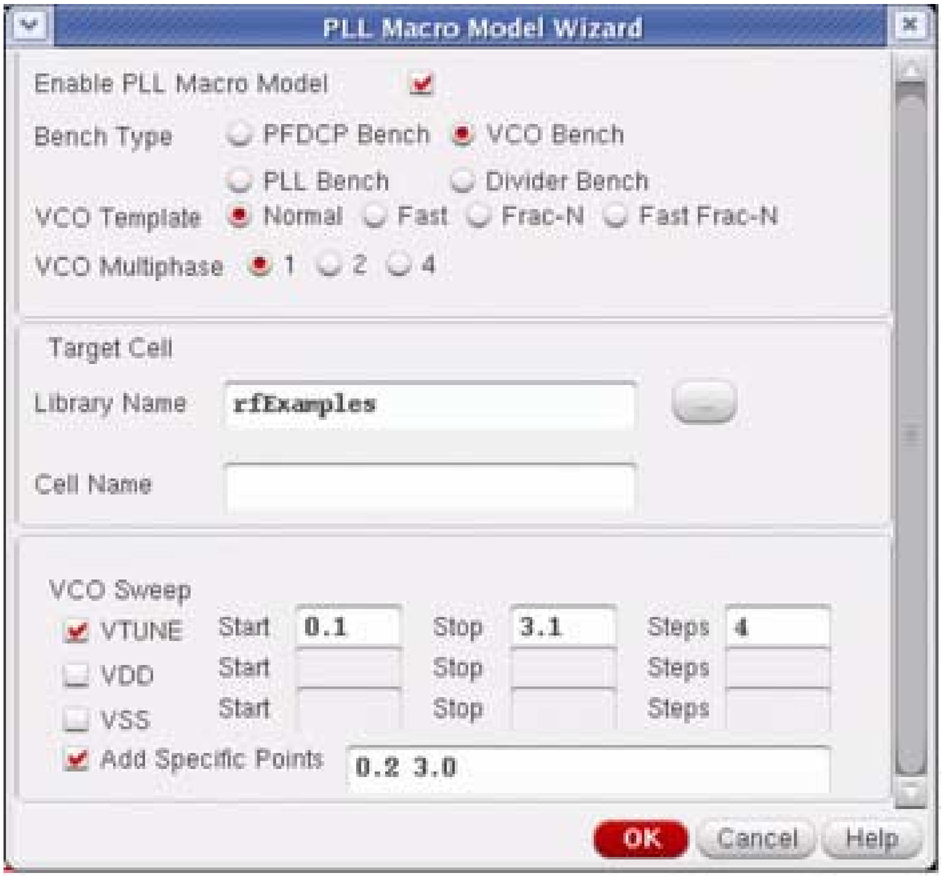

2. In the PLL Macro Model Wizard, select *VCO Bench*.

3. Select the kind of VCO template to be used.

   There are four VCO templates that can be selected: *Normal*, *Fast*, *fractional-N*, and *Fast Frac-N*, which correspond to four symbols: `oscmm`, `oscmm_fast`, `oscmm_frac`, and `oscmm_frac_fast`. (See "Using VCO Macro-Models" on page 982 for information about the four symbols).

4. In the *VCO Multiphase* field, select the number of outputs used by the VCO.

**Figure 16-5 PLL Macro Model Wizard**




5. Specify a library name and a cell name to contain the generated VCO macro-model.

6. Select the kind of VCO sweep.

   - ❑ If the *VTUNE* value is chosen for VCO Sweep, the three parameters `vtunestart`, `vtunestop`, and `vtunesteps` are set in the netlist

   - ❑ If the *VDD* value is chosen for *VCO Sweep*, the three parameters `vddstart`, `vddstop`, and `vddsteps` are set in the netlist. For example,

     ```
     parameters  vddstart=3.2 vddstop=3.4 vddsteps=3
     ```

   - ❑ If the *VSS* value is chosen for *VCO Sweep*, the three parameters `vssstart`, `vssstop`, and `vsssteps` are set in the netlist.

7. Set up a PSS + PNOISE analysis. In the PNOISE analysis, set `oprobe=vco_vout` rather than setting two output nets.

8. Run the simulation with the VCO extraction plugin.

   Using the plugin for VCO extraction, the command line is

```
spectre –plugin libpllPPVoscModel_sh.so
```

After extraction, a VCO macro-model is generated in the specified library and an associated file `oscPPVmodel.mat` is generated. The content of the file is similar to that illustrated in Table 16-1 on page 978.

**Note:** Be aware of the following information.

- In PLL designs, there are often buffers and prescalers after the VCO that could have a loading effect. These circuits can be included in the VCO extraction. However, including too many circuits or circuits that are large can affect the accuracy of the VCO macro-model.
- The plugin searches the source instances `vco_vtune`, `vco_vdd`, `vco_vss`, and `vco_vout` to get the related nets and terminals measurement results. These four names must not be changed.
- The $V_{ctrl}$ sweep range should be between $V_{min}$ and $V_{max}$ (see Figure 16-3 on page 976). A sweep grid that uses small steps increases the accuracy of the macro-model at the cost of increased extraction time. A good strategy is to use small steps in the grid near the locked $V_{ctrl}$ value and a coarse grid farther away.
- The $V_{DD}$ and $V_{SS}$ sweeps are optional. They simulate the impact of the large supply variation impact on the PLL. If the supply noise can be seen as a small signal, the $V_{DD}$ and $V_{SS}$ sweeps are not necessary. If both are selected, the total number of PSS + PNOISE runs are `N1*N2*N3`, where `N1` is the number of sweep points for $V_{ctrl}$ , `N2` is the number for $V_{DD}$ , and `N3` is the number for $V_{SS}$. So, the simulation can be very time-consuming.
- If no PNOISE analysis is set, the generated VCO macro-model is noiseless.
- VCO noise includes both white noise and flicker noise. The frequency sweep range should be larger than a decade. The noise below the start frequency (and beyond the stop frequency) point is treated as white noise with power the same as that of the start frequency (or stop frequency). So, if a one frequency point sweep is set, the VCO noise is white.
- From Equation 16-4 on page 977, `vco_vdd` and `vco_vss` are equivalent for VCO extraction. From this point, `vco_vdd` and `vco_vss` can be any voltage source whose noise can affect the VCO output, not only the drain or source supply for the transistor. For example, if the analog supply AVDD noise and the digital supply DVDD noise are considered in the VCO at the same time, you can extract the VCO by connecting `vco_vdd` to AVDD and `vco_vss` to DVDD.

### Using VCO Macro-Models

There are 12 symbols (4 templates times 3 multiphase choices) that are supported for the oscmm model.


- oscmm Symbol on page 983
- oscmm_fast Symbol on page 984
- oscmm_frac Symbol on page 986
- oscmm_frac_fast Symbol on page 989

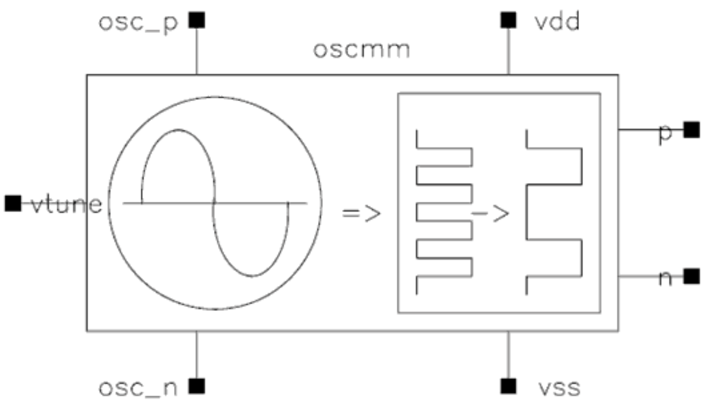

The oscmm symbols with `twin` in their names have two output terminals so that the symbols can be used to simulate differential VCOs.


- oscmm_twin Symbol on page 990
- oscmm_fast_twin Symbol on page 992
- oscmm_frac_twin Symbol on page 993
- oscmm_frac_fast_twin Symbol on page 995


The oscmm symbols that have `tet` in their names have four output terminals so that the symbols can be used to simulate VCOs with four output phases.


- oscmm_tet Symbol on page 996
- oscmm_fast_tet Symbol on page 998
- oscmm_frac_tet Symbol on page 999
- oscmm_frac_fast_tet Symbol on page 1001

### *oscmm Symbol*




Pins:

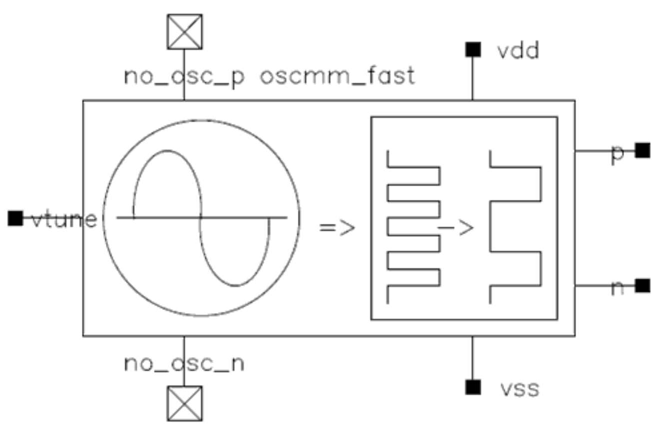
| pin | use |
|---|---|
| `div_p` | Divider output. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `osc_p` | VCO output. |
| `osc_gnd` | VCO ground. |

Parameters:

| parameter | use |
|---|---|
| `model_file` | Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`. |
| `divider_ratio` | The VCO output divided by this ratio is the divider output. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |

| parameter | use |
|---|---|
| bound | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| vcojitter | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| jitterstart | Determines when the VCO noise (or jitter) takes effect in the transient. |
| noiseseed | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |
| osctrtf | Risetime and falltime of VCO output waveform. |
| divtrtf | Risetime and falltime of divider output waveform. |

#### *oscmm_fast Symbol*




Pins:

| pin | use |
|---|---|
| div_p | Divider output. |
| div_gnd | Divider ground. |

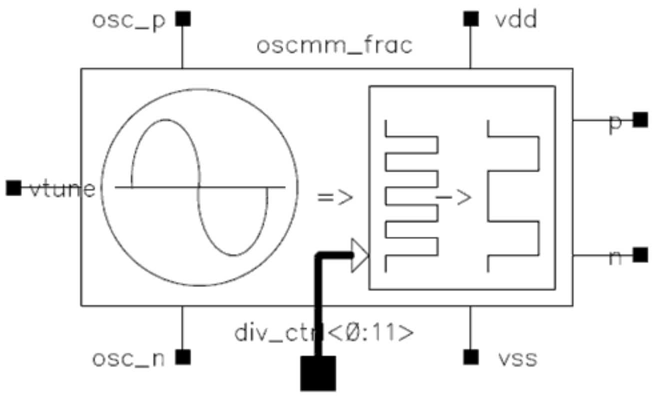
| pin | use |
|---|---|
| vtune | VCO control voltage input. |
| vdd | VCO power voltage input. |
| vss | VCO substrate voltage input. |

Parameters:

| parameter | use |
|---|---|
| model_file | Name of the file containing the VCO macro-model. Default value is oscPPVmodel.mat. |
| divider_ratio | The VCO output divided by this ratio is the divider output. |
| vlo | VCO and divider output low level voltage. |
| vhi | VCO and divider output high level voltage. |
| bound | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| vcojitter | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless. |
| jitterstart | Determines when the VCO noise (or jitter) takes effect in the transient. |
| noiseseed | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

**Note:** The oscmm_fast symbol does not include VCO output, which makes the simulation very fast. However, this lack of VCO output disables phase noise measurement at the VCO output.

### *oscmm_frac Symbol*




Pins:

| pin | use |
|---|---|
| div_p | Divider output. |
| div_gnd | Divider ground. |
| vtune | VCO control voltage input. |
| vdd | VCO power voltage input. |
| vss | VCO substrate voltage input. |
| osc_p | VCO output. |
| osc_gnd | VCO ground. |
| div_ctl<0:11> | Control bus. Controls the divider ratio instantly according to the signal on it. |

Parameters:

| parameter | use |
|---|---|
| model_file | Name of file containing the VCO macromodel. Default value is oscPPVmodel.mat. |
| vlo | VCO and divider output low level voltage. |
| vhi | VCO and divider output high level voltage. |

| parameter | use |
|---|---|
| bound | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| vcojitter | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| jitterstart | Determines when the VCO noise (or jitter) takes effect in the transient. |
| div_ctl_vlo | Low level of the control signal on the control bus. |
| div_ctl_vhi | High level of the control signal on the control bus. |
| div_ratios | Array of divide ratios. For example, [2477 2478 2479 2480]. |
| noiseseed | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |
| mappingmode | Mapping mode between divider control bus signal and divider ratio. Possible values are parammapping, filemapping and signalmapping. Default value is parammapping. |
| mappingfile | The file which contains the mapping between divider control bus signal and divider ratio. |
| osctrtf | Risetime and falltime of VCO output waveform. |
| divtrtf | Risetime and falltime of divider output waveform. |

**Note:** Be aware of the following information.

- The `oscmm_frac` symbol allows you to change the divider ratio instantly, which can be used to simulate a fractional-N PLL or PLL channel selection.
- The signals on the `div_ctl<0:11>` bus are seen as digital signals. The high level is specified by parameter `div_ctl_vhi` and the low level by parameter `div_ctl_vlo`. The threshold voltage is determined by `(div_ctl_vhi+div_ctl_vlo)/2`. At any time, the signals form a control byte `CTLBYTE` with a value range of 0~4095. The divider's instant divide ratio is determined by `CTLBYTE` and the mapping mode. If mapping mode is “parammapping”, the parameter `div_ratios`. `div_ratios` is an array and its maximum length is 4096. Suppose `div_ratios(n)` represents the nth value in the

`div_ratios` array, the mapping between `CTLBYTE` and the divider's instant divide ratio is shown in Table 16-2 on page 988.

**Table 16-2**

| CTLBYTE | divider ratio |
|---|---|
| 0x000 | div_ratios(0) |
| … | … |
| 0x064 | div_ratios(100) |
| … | … |
| 0xFFF | div_ratios(4095) |

If mapping mode is filemapping, the parameter mappingfile gives the mapping between CTLBYTE and the divider ratio. The file content is two columns ASCII, the first column is CTLBYTE (which must be binary) and the second column is divider ratio. If an instant CTLBYTE is not found in the file, the previous divider ratio will be used.

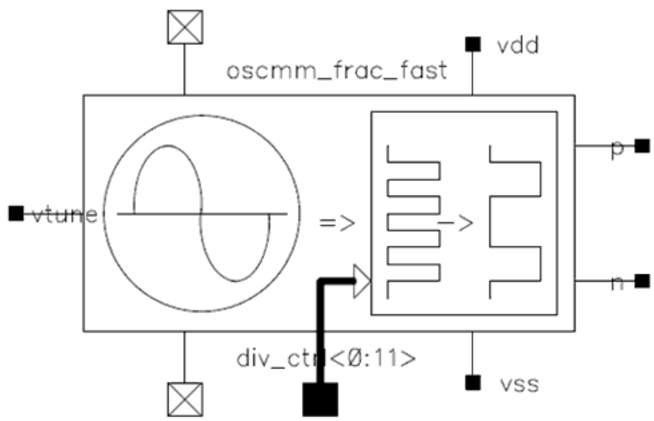
For example, if a file has the following columns:

010110101111 120
000101010011 121
000111010001 122

which shows that the divider ratio has changed from 120 to 122, according to CTLBYTE.

If mapping mode is signalmapping, CTLBYTE is directly interpreted as the divider ratio which is in the range of 1 ~ 4095, such as 0x064 representing a divider ratio of 100.

### *oscmm_frac_fast Symbol*




Pins:

| pin | use |
|---|---|
| `div_p` | Divider output. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `div_ctl<0:11>` | Control bus. |

Parameters:

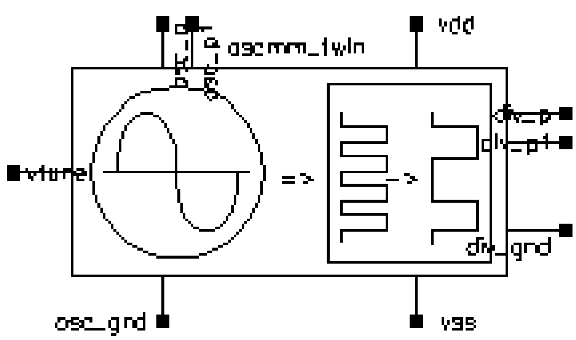
| parameter | use |
|---|---|
| `model_file` | Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |

| parameter | use |
|---|---|
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |
| `div_ratios` | Array of divide ratios. Such as [2477 2478 2479 2480]. |

**Note:** Be aware of the following information.

- The `oscmm_frac_fast` symbol does not include VCO output, which makes simulation very fast. However, this lack of VCO output disables phase noise measurement at the VCO output.

### *oscmm_twin Symbol*




Pins:

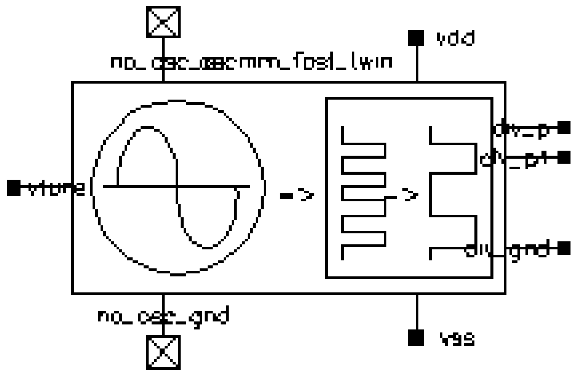
| pin | use |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |

| pin | use |
| --- | --- |
| `osc_p` | VCO output 0. |
| `osc_p1` | VCO output 1. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `osc_gnd` | VCO ground. |

Parameters:

| parameter | use |
| --- | --- |
| `model_file` | Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`. |
| `divider_ratio` | The VCO output divided by this ratio is the divider output. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `noiseseed` | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

### *oscmm_fast_twin Symbol*




Pins:

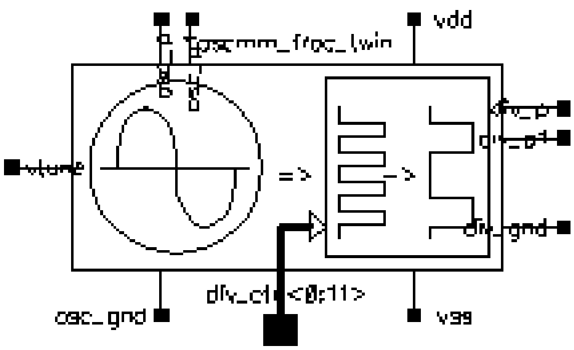
| **pin** | **use** |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |

Parameters:

| **parameter** | **use** |
|---|---|
| `model_file` | Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`. |
| `divider_ratio` | The VCO output divided by this ratio is the divider output. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |

| parameter | use |
|---|---|
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `noiseseed` | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

### *oscmm_frac_twin Symbol*




Pins:

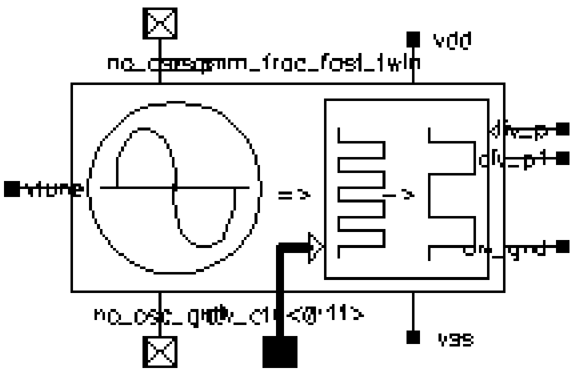
| pin | use |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `osc_p` | VCO output 0. |
| `osc_p1` | VCO output 1. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |

| **pin** | **use** |
|---|---|
| `vss` | VCO substrate voltage input. |
| `osc_gnd` | VCO ground. |
| `div_ctl<0:11>` | Control bus. |

Parameters:

| **parameter** | **use** |
|---|---|
| `model_file` | Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |
| `div_ratios` | Array of divide ratios. For example, [2477 2478 2479 2480]. |
| `noiseseed` | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

### *oscmm_frac_fast_twin Symbol*




Pins:

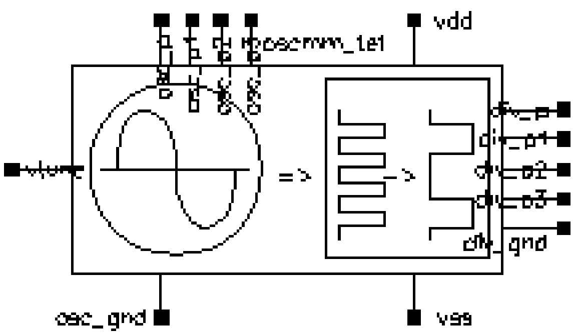
| pin | use |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `div_ctl<0:11>` | Control bus. |

Parameters:

| parameter | use |
|---|---|
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |

| **parameter** | **use** |
|---|---|
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |
| `div_ratios` | Array of divide ratios. Such as, [2477 2478 2479 2480]. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |

### *oscmm_tet Symbol*




Pins:

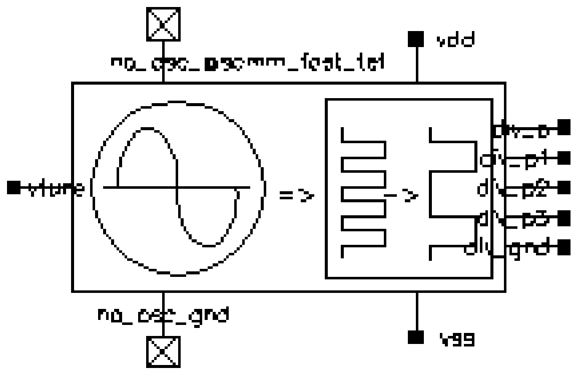
| **pin** | **use** |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `div_p2` | Divider output 2. |
| `div_p3` | Divider output 3. |
| `osc_p` | VCO output 0. |
| `osc_p1` | VCO output 1. |

| **pin** | **use** |
|---|---|
| `osc_p2` | VCO output 2. |
| `osc_p3` | VCO output 3. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `osc_gnd` | VCO ground. |

Parameters:

| **parameter** | **use** |
|---|---|
| `model_file` | Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |

### *oscmm_fast_tet Symbol*




Pins:

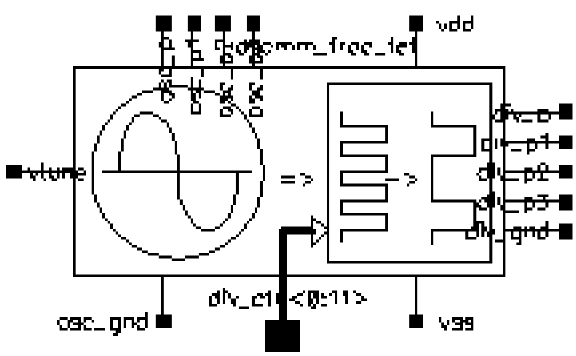
| pin | use |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `div_p2` | Divider output 2. |
| `div_p3` | Divider output 3. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |

Parameters:

| parameter | use |
|---|---|
| `model_file` | Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`. |
| `divider_ratio` | The VCO output divided by this ratio is the divider output. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |

| **parameter** | **use** |
|---|---|
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `noiseseed` | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

### *oscmm_frac_tet Symbol*




Pins:

| **pin** | **use** |
|---|---|
| `div_p` | Divider output 0. |
| `div_p1` | Divider output 1. |
| `div_p2` | Divider output 2. |
| `div_p3` | Divider output 3. |
| `osc_p` | VCO output 0. |

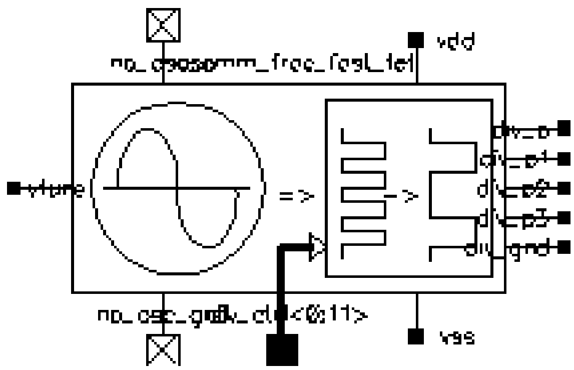
| pin | use |
|---|---|
| `osc_p1` | VCO output 1. |
| `osc_p2` | VCO output 2. |
| `osc_p3` | VCO output 3. |
| `div_gnd` | Divider ground. |
| `vtune` | VCO control voltage input. |
| `vdd` | VCO power voltage input. |
| `vss` | VCO substrate voltage input. |
| `osc_gnd` | VCO ground. |
| `div_ctl<0:11>` | Control bus. |

Parameters:

| parameter | use |
|---|---|
| `model_file` | Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |
| `div_ratios` | Array of divide ratios. For example, [2477 2478 2479 2480]. |

| parameter | use |
|---|---|
| noiseseed | A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment. |

### *oscmm_frac_fast_tet Symbol*




Pins:

| pin | use |
|---|---|
| div_p | Divider output 0. |
| div_p1 | Divider output 1. |
| div_p2 | Divider output 2. |
| div_p3 | Divider output 3. |
| div_gnd | Divider ground. |
| vtune | VCO control voltage input. |
| vdd | VCO power voltage input. |
| vss | VCO substrate voltage input. |
| div_ctl<0:11> | Control bus. |

Parameters:

| **parameter** | **use** |
|---|---|
| `model_file` | Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`. |
| `vlo` | VCO and divider output low level voltage. |
| `vhi` | VCO and divider output high level voltage. |
| `bound` | Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement. |
| `vcojitter` | This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless. |
| `jitterstart` | Determines when the VCO noise (or jitter) takes effect in the transient. |
| `div_ctl_vlo` | Low level of the control signal on the control bus. |
| `div_ctl_vhi` | High level of the control signal on the control bus. |
| `div_ratios` | Array of divide ratios. Such as, [2477 2478 2479 2480]. |

### oscmm CMI Command Line

Using oscmm CMI, the command line is

```
spectre –cmiconfig myconfig.cfg
```

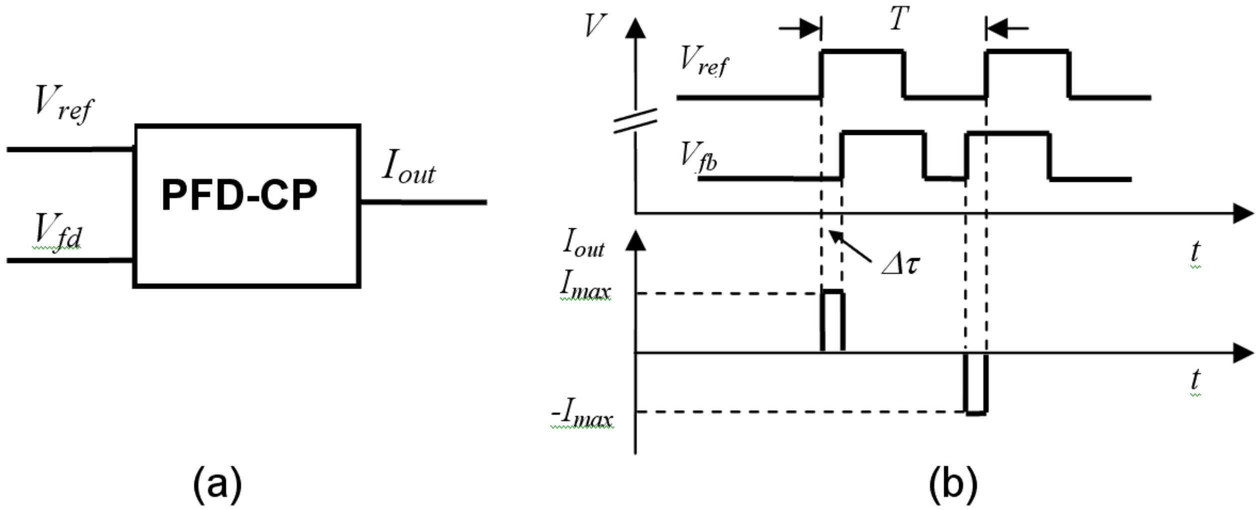
`myconfig.cfg` is a config file, its content is as follows.

```
append   ... /tools.lnx86/spectre/lib
load libpll_sh.so
```

## Extracting the PFD-CP Macro-Model

A simple three state PFD-CP schematic (a) and its input and output relationships (b) are shown in Figure 16-6 on page 1003.

**Figure 16-6  PFD-CP input and output**




It receives two input signals, the reference clock ($V_{ref}$) and the feedback clock ($V_{fd}$). The CP output current signal ($I_{out}$) changes according to the phase difference ($\Delta\tau$ or $\Delta\phi$) of the two PFD input signals. If only the average output current $\langle I_{out} \rangle$ is considered, then

**(16-7)** $$\langle I_{out} \rangle = I_{max} \cdot (\Delta t)/T$$

where $T$ is the period of the two input signals and $I_{max}$ is the CP maximum output current. Generally, $\langle I_{out} \rangle$ is noisy for PFD-CP internal noise. Assuming that the output current noise variance is $(\sigma_{\langle Iout \rangle})^2$, the variance can be equivalent to the input phase noise (or input edge jitter), as follows,

**(16-8)** $$J_{ee} = \sigma_{\langle Iout \rangle} \cdot T/I_{max}$$

where $J_{ee}$ is the input signal edge jitter. $\sigma_{\langle Iout \rangle}$ can be calculated as follows,

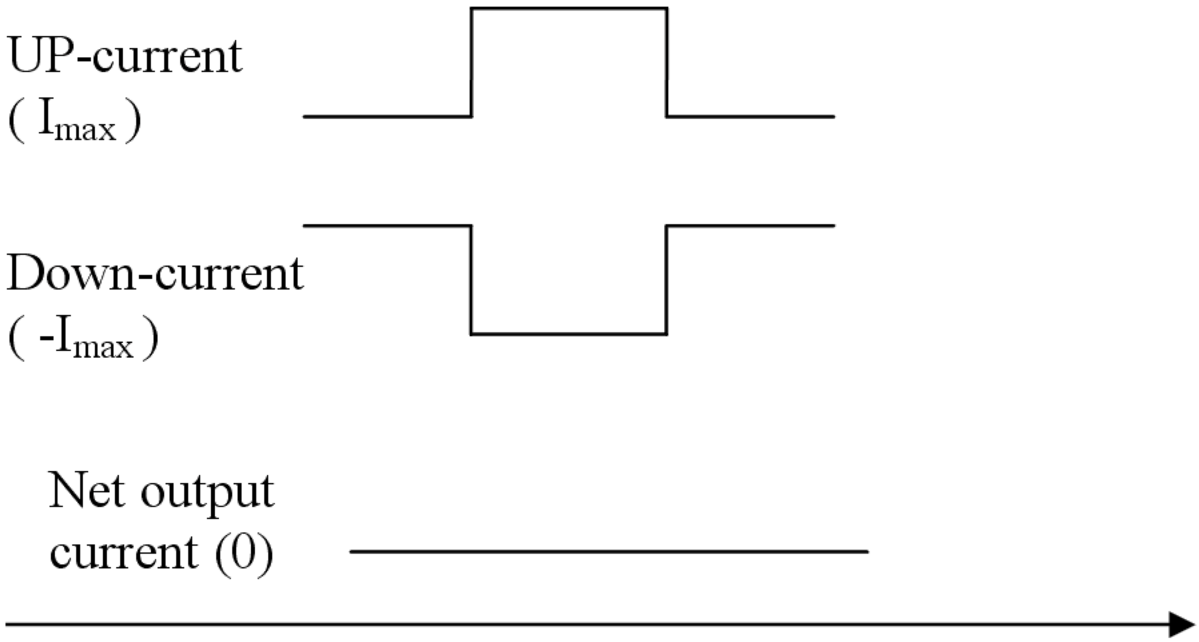
**(16-9)** $$\sigma_{\langle Iout \rangle} = \sqrt{\frac{1}{2}\int_0^{\infty} S_n(f)df}$$

where $S_n(f)$ is the PFD-CP total output noise power spectral density, which can be obtained using PNOISE analysis.

There are several nonideal effects that impact PFD-CP performance.

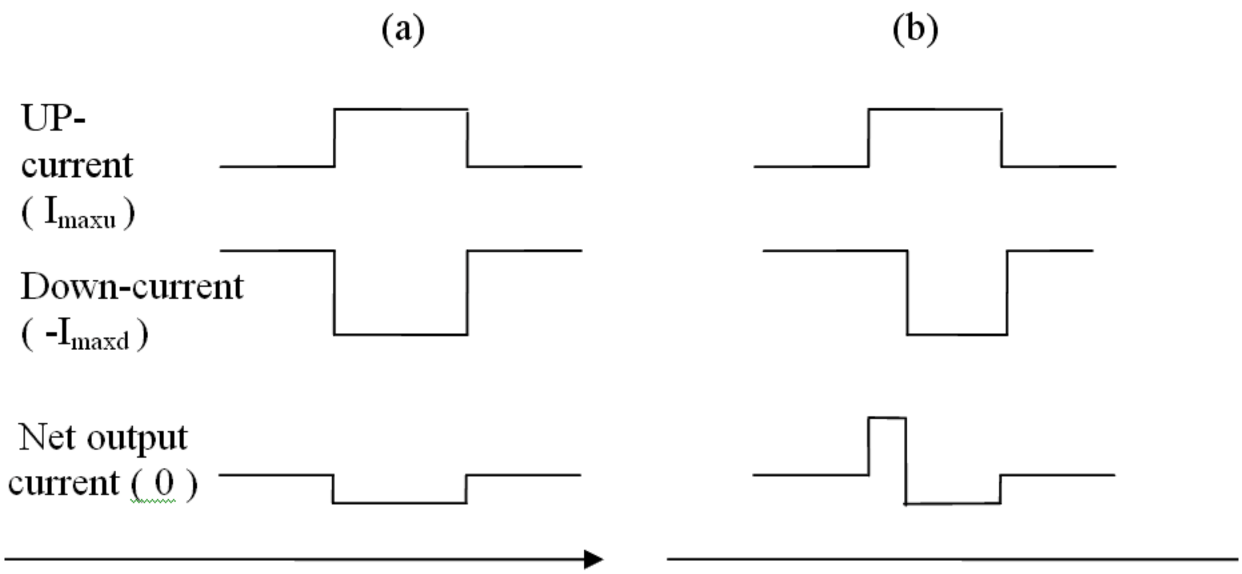
- PFD-CP current mismatch

**Figure 16-7 Ideal PFD_CP output when the PLL is locked**




The `UP` current and `DOWN` current open at the same time, so the net output should be zero. Generally, for a real circuit, the `UP` current and `DOWN` current are mismatched, which leads to a nonzero net output, although the input reference and feedback clock phases are the same (see (a) in Figure 16-6 on page 1005). To remain locked, the reference and feedback clock must have a small offset so that the net output current in one cycle is zero (see (b) in Figure 16-6 on page 1005). This output has periodic ripples that can modulate `Vtune` and lead to a closed-loop VCO output that has spurs with offset frequency $N*f_{ref}$, where N is a positive integer and $f_{ref}$ is a reference frequency.

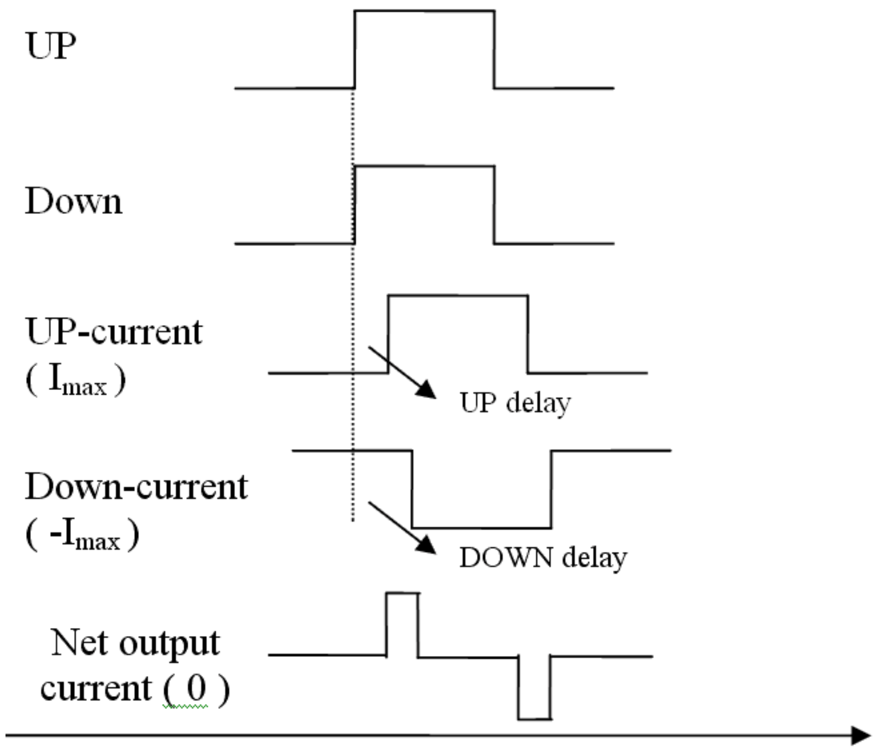
**Figure 16-8  Effect of UP and DOWN current mismatch on PFD-CP**




- PFD-CP delay mismatch

  Figure 16-6 on page 1006 shows the delay mismatch. Suppose the PLL is locked and the `UP` and `DOWN` signals arrive at the same time, which will open the `UP` and `DOWN` current for a short interval. If the delay from the `UP` signal to the `UP` current is different from that from the `DOWN` signal to the `DOWN` current, the net output current in a cycle is not constant but instead has ripples or spurs.

**Figure 16-9 Effect of UP and DOWN current delay mismatch on PFD-CP**



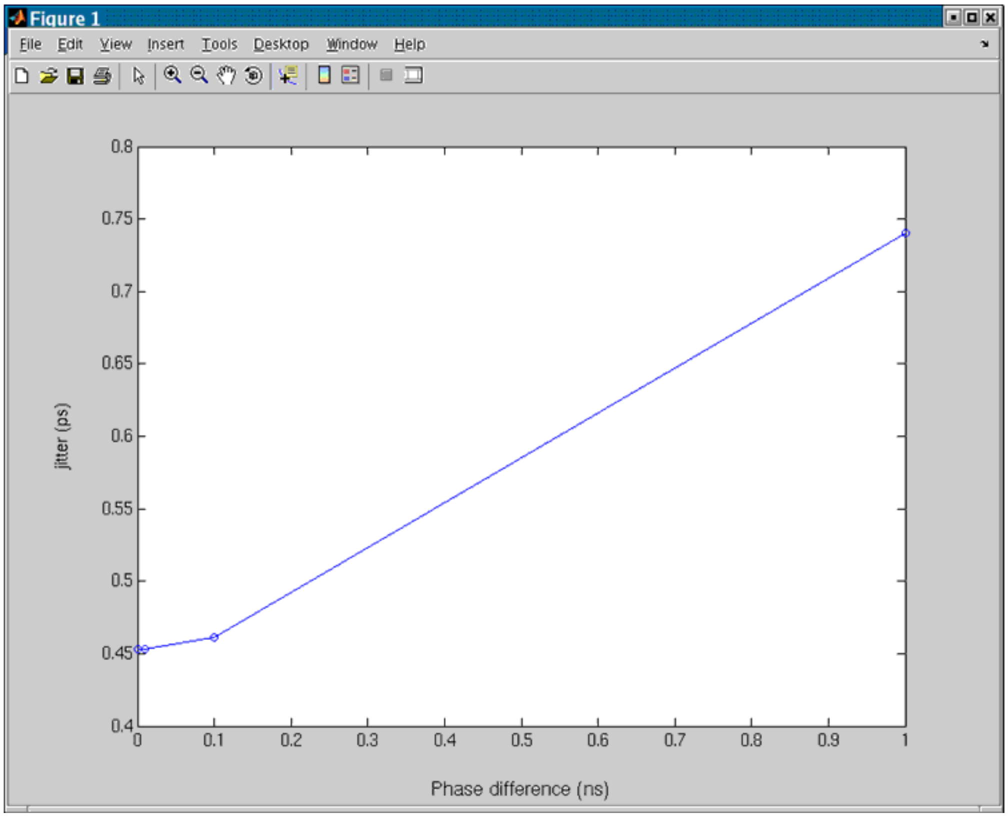

These irregularities are injected into the `Vtune` and impact the PLL output.

Other nonideal effects, such as rising time mismatch, falling time mismatch, and output current offset, can have a similar effect on the performance of the PFD-CP.

- PFD-CP jitter variation with phase difference

  The jitter and output current variation of the PFD-CP is dependent on the phase difference between the input reference clock, feedback clock and the load voltage. Simulation performed on a sample design shows the jitter dependency on phase difference in Figure 16-6 on page 1007.

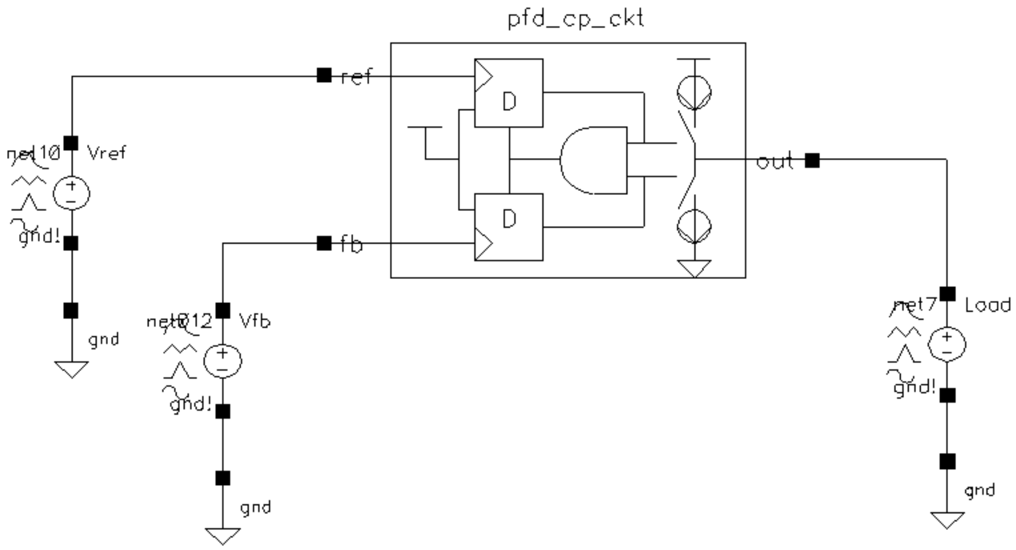
**Figure 16-10 PFD-CP jitter related to phase difference**




## PFD-CP macro-model extraction

For PFD-CP macro-model extraction, the PFD-CP transistor level circuit is instantiated in the PFD-CP extraction test bench `pfd_cp_bench`, which is provided in the `pllMMLib` library. [Figure 16-11](#) on page 1008 is the schematic of the test bench.

**Figure 16-11 PFD-CP test bench for macro-model extraction**




To run the extraction,

1. Set up a PSS + PNOISE analysis. Set the reference clock and feedback clock to square waves with the same reference frequency. Set the correct DC value for *Load source* so that the CP can work normally under the bias.
2. In the PLL Macro Model Wizard, select *PFDCP Bench*.
3. Specify a library name and a cell name to contain the PFD-CP macro-model.

**Figure 16-12 PLL Macro Model Wizard**


**4.** The optional *PFDCP Sweep* item is for simulating the jitter and output current dependency on phase difference. The three supported kinds of sweep are *Disable, fine*, and *coarse*.

- If *Disable* is selected, jitter and the current is measured at 0.1*T and -0.1*T
- Selecting a *fine* sweep requires the most extraction time but the created macro-model is the most accurate. If *fine* is selected, the parameter `jittersweep=fine` is set in the netlist. This selection measures jitter and output current values at phase difference equals 0.1*T, 0.01*T, 0.001*T, 0.000*T, 0, -0.0001*T, -0.001*T, -0.01*T, -0.1*T respectively, where T is the reference clock period.
- If *coarse* is selected, jitter and the current is measured at 0.1*T, 0 and -0.1*T.

**5.** The optional Load Voltage Sweep option can be used for simulating the load voltage effect on the jitter and output current.

**6.** Run the simulation with the PFD-CP extraction plugin. After PFD-CP extraction finishes, the PFD-CP Verilog-A macro-model is generated.

**Note:** Be aware of the following information.

- The Spectre simulator command for using the plugin for PFD-CP extraction is

  `spectre –plugin libpllTTpfd_cpModel_sh.so`

- The plugin searches the `Vref`, `Vfb`, and `Load` source instances to get the related nets and terminals measurement results. These three names must not be changed.

- Usually, a tri-state PFD-CP is a strongly nonlinear circuit, so it is best to use shooting PSS + PNOISE to extract the PFD-CP model. Also, shooting is usually more accurate than harmonic balance for obtaining the rising time and delay, which are time domain parameters.

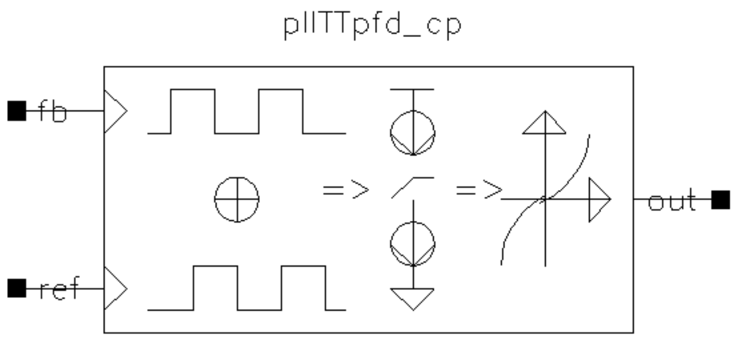
- Set the two parameters `dir` and `vth`. The first parameter specifies the PFD input trigger edge (a `dir` value >= 0 sets a rise edge trigger; a `dir` value <0 sets a fall edge trigger); the second parameter specifies the plugin trigger threshold voltage.

- Use the vsource `delay` parameter to set the phase difference between $V_{ref}$ and $V_{fb}$. The delay difference of $V_{fb}$ and $V_{ref}$ can be arbitrary.

- Choose the upper limit of the sweep frequency range of the PNOISE analysis so that the total noise at frequencies outside the range is negligible. The noise should be at least 40 dB down and dropping at the highest frequency simulated. Choose the lower limit of integration by considering what value is below the system bandwidth and the amount of validity in the flicker noise model, which can grow unbounded as the frequency decreases.

## Using PFD-CP macro-model

### pllTTpfd_cp Symbol




pins:

| pin | use |
|---|---|
| `out` | CP current output. |
| `ref` | Reference clock input. |
| `fb` | Feedback clock input. |

Parameters:

| parameter | use |
|---|---|
| `Imax` | CP maximum output current. |
| `Vtrans` | Input trigger threshold. |
| `vth` | pfd_cp trigger threshold voltage. |
| `uptr` | Up current rise or fall time. |
| `downtr` | Down current rise or fall time. |
| `refdelay` | Delay time from reference clock input to up current output. |
| `fbdelay` | Delay time from feedback clock input to down current output. |

| parameter | use |
|---|---|
| `Imis` | Mismatch between up current and down current. It is defined by $$I_{mis} = \frac{I_{up} - I_{down}}{I_{max}}$$ $$I_{max} = \frac{I_{up} - I_{down}}{2}$$ |
| `Ioffset` | Output current offset. |
| `dir` | Set `dir`=1 for a rising edge trigger and `dir`=-1 for a falling edge trigger. |
| `jitter` | Equivalent input edge jitter from PFD-CP internal noise. |
| `modelfile` | The file contains PFD-CP macro-model jitter information. |
| `noisestart` | The time when jitter noise starts. |
| `noiseseed` | Jitter noise seed. |

**Note:** Be aware of the following information.

- If the `dir` parameter is not set during extraction, its default value is 1.
- If the `vth` parameter is not set during extraction, the default value of `Vtrans` is 1.0.

# Divider Extraction

## Divide Macro-Model Basis

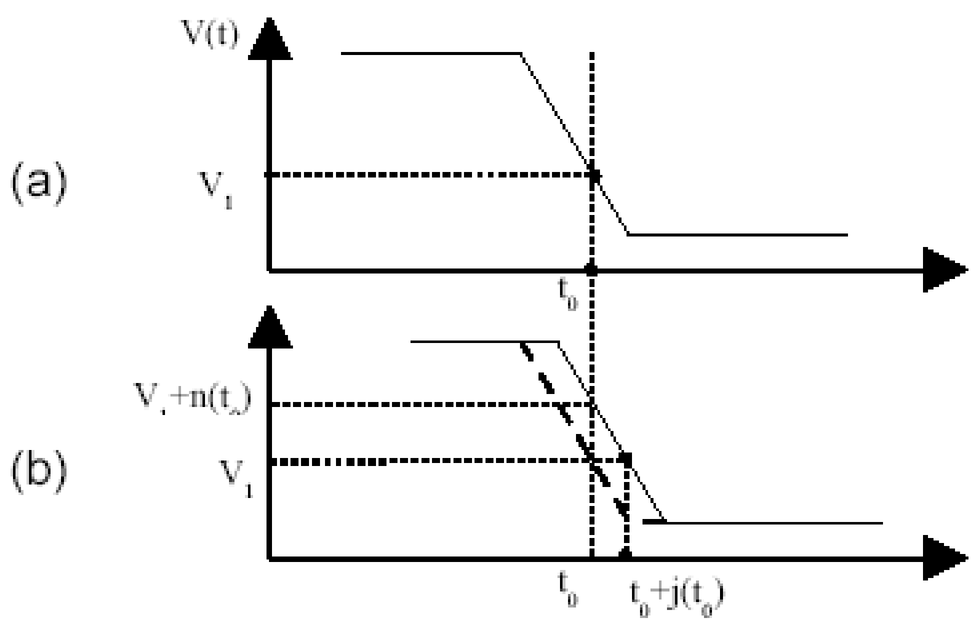
Divider's large signal behavior +N is integrated into VCO macro-model, so extracted divider s macro-model will only include its small signal (jitter) behavior, risetime, falltime and delay.

Suppose v*(t)* represents the divider's ideal periodical output, a falling edge threshold cross event happens at (t0, v1) (see figure G-13 (a)). However, when noise *n(t)* is added to the signal, $v_n(t) = v(t) + n(t)$, the cross event is displaced slightly and happens at (t0+j($t_0$), v1) (see G-13 (b)), j($t_0$) is instant jitter. The jitter is determined by n(t) and the slew rate of the signal dV(t)/dt.

Generally *n* is not stationary, but cyclostationary. It is only important to know when the noisy periodic signal *v (t)* crosses the threshold, so the statistics of *n* are only significant at the time when $v_n(t)$ crosses the threshold. In this case, the edge-edge jitter can be calculated:

$$J_{ee} = \frac{\sqrt{\mathrm{var}(n(t_0))}}{dv(t_0)/dt}$$

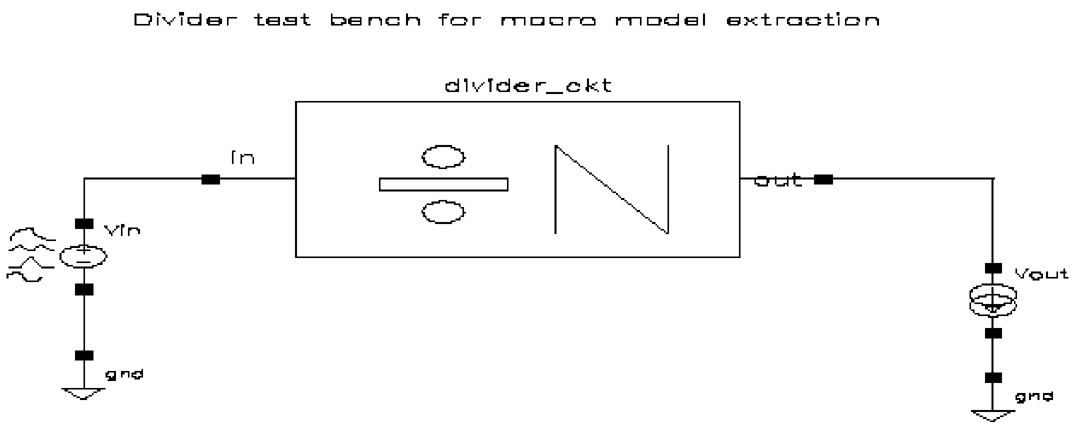
**Figure 16-13  PDF-CP input and output**



## Divider Macro-Model Extraction

For divider macro-model extraction, the divider transistor level circuit is instantiated in the divider extraction test bench `divider_bench`, which is provided in the `pllMMLib` library. Figure G-14 is the schematic of the test bench.

**Figure 16-14  Divider test bench for macro-model extraction**




To run the extraction:

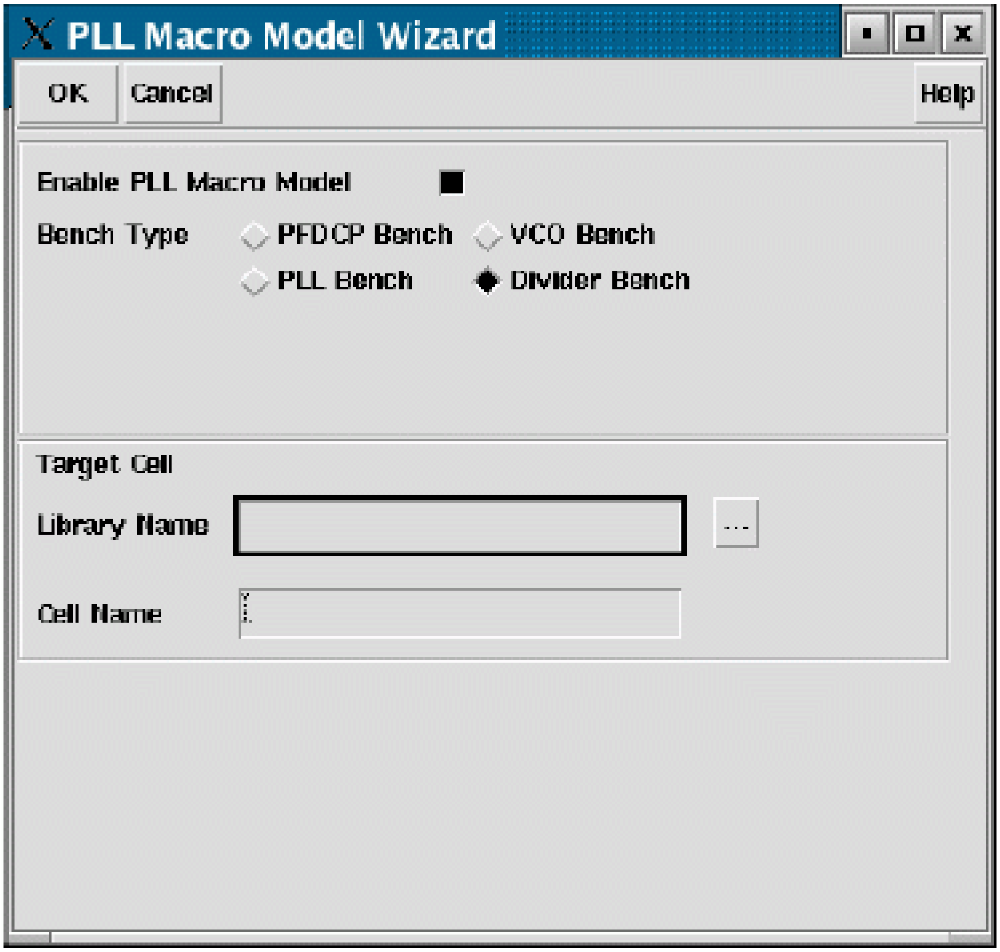
1. Set up a PSS + PNOISE analysis.

   The input signal should be set to square waves. PNOISE sweep upper frequency should be set to $f_0/2$, where $f_0$ is input signal s frequency. The noise type should be set to “jitter” in PNOISE.

2. In the ADE PLL wizard, select *Divider Bench*.

3. Specify a *Library Name* and a *Cell Name* to contain the divider macro-model.

**Figure 16-15**




4. Run the simulation with divider extraction plugin. After divider extraction finishes, a Verilog-A module file is generated, which is the divider Verilog-A macro-model.

**Note:**

- Using the plugin for divider extraction, the command line is:

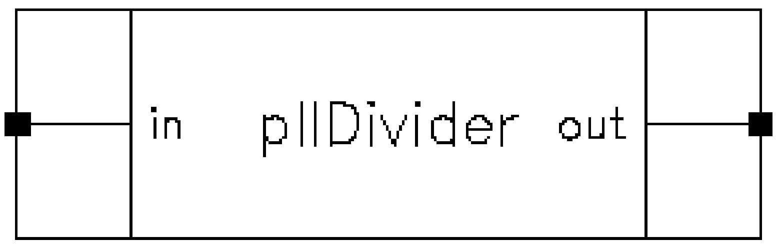
  ```
  spectre plugin libpllDivider_sh.so
  ```

- The plugin searches the Vin and Vout to get the related nets and terminals measurement results. These names cannot be changed.
- At present, only shooting PSS + PNOISE can be used to extract divider macro-model.
- When noise type is jitter, SpectreRF uses strobed PNoise analysis to compute $S_n( f )$. The sample point should be set to coincide with the point where the output signal crosses the threshold of the subsequent stage (the phase detector) in the appropriate direction.

## Using Divider Macro-Model

Model name: jee_gen

Symbol: pllDivider




pins: (out in)

| | |
|---|---|
| `out` | Divider output. |
| `In` | Divider input. |

Parameters:

| | |
|---|---|
| `Vth` | Threshold voltage of input cross event. |
| `High` | High level voltage of divider output. |
| `Low` | Low level voltage of divider output. |
| `Frequency_Divider` | Output signal’s frequency. |
| `modelFile` | File which contains divider model information. It is generated automatically during extraction. Main content in this file is jitter power spectrum. |
| `Tr` | Risetime of divider output signal. |
| `Tf` | Falltime of divider output signal. |
| `Td` | Delay time from input to output. |
| `Jitter` | Divider jitter ($J_{ee}$). The jitter has Gaussian distributions. If this parameter is provided, the macro-model s internal jitter is disabled. By default, the macro-model s internal jitter is frequency dependent. |

| | |
|---|---|
| `noiseSeed` | Noise seed of jitter random process. |
| `noiseStart` | Time when jitter is added to output signal. |

**Note:** This jee_gen model can be also used to simulate reference clock noise. That is, if you can provide reference clock's jitter power spectrum according modelFile format, the model will reproduce the jitter profile.

# PLL Simulation with Macro-Models

After PFD-CP and VCO extraction, the whole PLL simulation with these two macro-models can be performed. Figure 16-16 on page 1019 and Figure 16-17 on page 1020 show the PLL test bench in `pllMMLib`.

If the PLL noise performance is of interest, add a `freq_meter` instance to the test bench. This instance measures the periods of VCO output (`vco_p` and `vco_n`) or divider output (`div_p div_n`) in response to rise cross events and writes the periods into a file. After a transient analysis, a plugin searches this instance and the file and calculates the phase noise power spectrum density (PSD) from the periods in the file.

To prepare for and run the simulation, do the following:

1. Attach the LPF circuit into `lpf_ckt`.
2. Insert the extracted `pllTTpfd_cp` view and pllDivider view into the PLL bench. Set parameters if needed.
3. Insert the extracted oscmm view into the PLL bench. Set the oscmm parameters.
4. Set the `freq_meter` parameters. There are four parameters used in `freq_meter`:

| **parameter** | **use** |
|---|---|
| `Vthup` | Assigns the rise cross threshold voltage. |
| `ttol` | Time tolerance for cross event. |
| `outfile` | Assigns the output file name (default is `periods.txt`). |
| `outStart` | Gives the time when the measured periods will be output to the outfile. |

5. In the PLL Macro Model Wizard, select *PLL Bench*.
6. Set up the Tran analysis and run the simulation.

If the PSD plugin is used, a file `pllmmnoise.vcsv` is generated in the result directory. It can be opened directly by using the Result Browser or from the Direct Plot Form by clicking *PLL Noise PSD*.

### Figure 16-16 PLL macro-model test bench (integer-N)

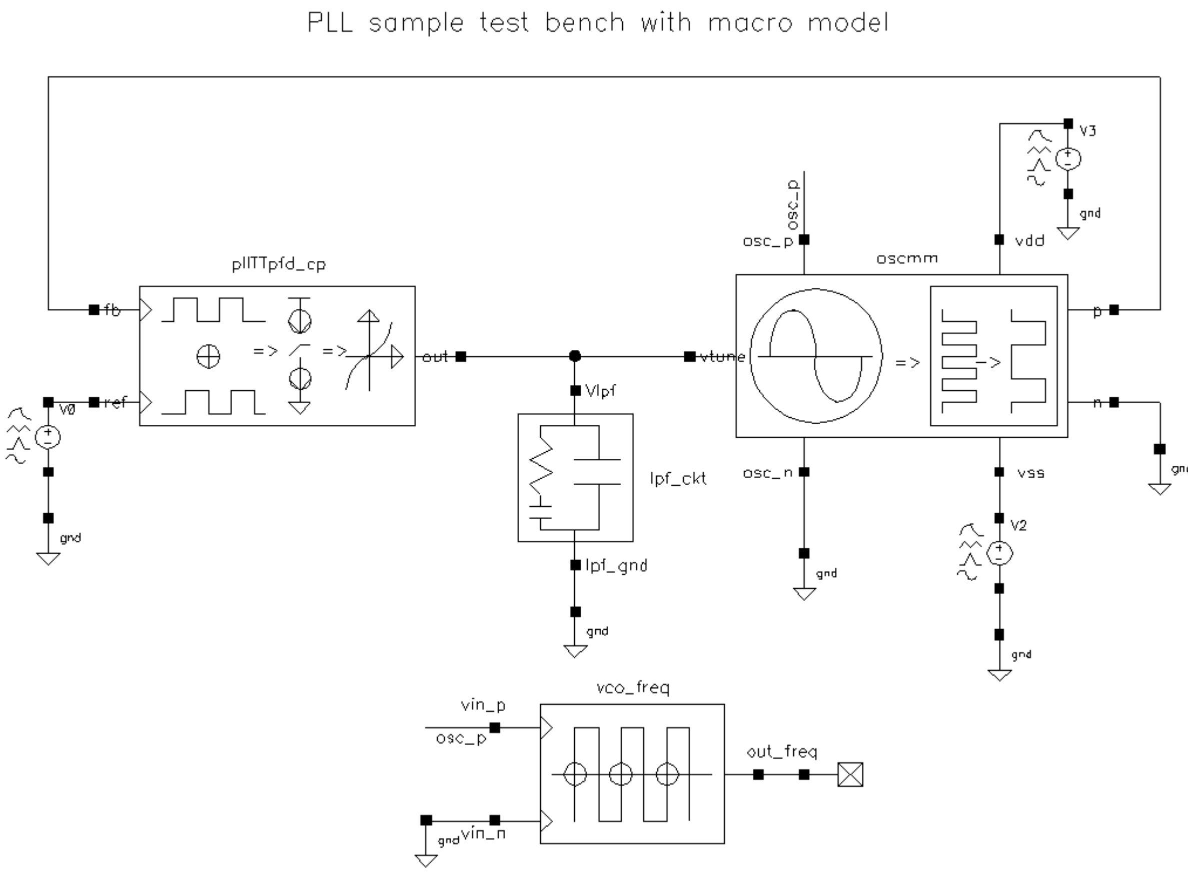
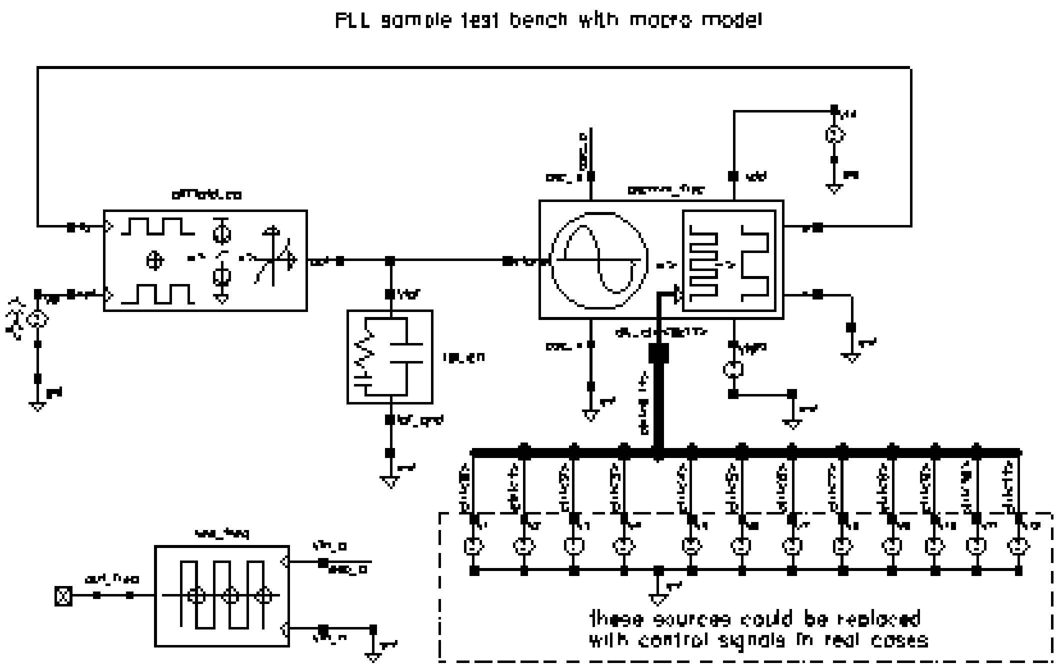


**Figure 16-17 PLL macro-model test bench (fractional-N)**




**Note:** Be aware of the following information.

- Using the PSD plugin for phase noise PSD calculation, the command line is

  `spectre –plugin libpllMMpsd_sh.so`

- To obtain accurate locked PLL noise behavior, ensure that the `freq_meter outStart` parameter is large enough so that the PLL is well locked.

- The PSD plugin searches the `vco_freq` instance to get the related period measurement results. This name must not be changed.

- The single-sided PSD output offset frequency range is $\Delta f_{min}$~$\Delta f_{max}$ with respect to $f_0$ which is the VCO locked output frequency, where $\Delta f_{max}=f_0/2$ and $\Delta f_{min}= 4/(t_{start}t_{stop})$. The $t_{stop}$ is the transient analysis stop time and the $t_{start}$ equals to 'outstart' value of the freq_meter. To obtain low offset frequency PLL noise behavior, the transient analysis stop time must be set to a large value.

- To simulate a fractional-N PLL, use either the `oscmm_frac` or `oscmm_frac_fast` symbols in the PLL test bench. The `div_ctrl<0:11>` can be connected to the delta-sigma modulator.

# Sigma-Delta Modulator Macro-Model

There are three demo sigma-delta modulators macro-models, `SDM_MASH_3rd_111_macro`, `SDM_3rd_MASH_111` and `SDM_3rd_MASH_2-1` in `pllMMLib`. The latter two are constructed using veriloga-A modules, such as `z_inv_digi` `(z`$^{-1}$`)`, `z_integrator_digi(z`$^{-1}$`/(1-z `$^{-1}$`))`, `one-z_inv(1-z `$^{-1}$`)` and `quantizer_midtread_digi (quantizer)`. You can construct other type of sigma-delta modulators with these modules. )

The function and parameters are:

Module name: `z_inv_digi_(z`$^{-1}$`)`

Function: one bit sample

pins (in out clk):

| | |
|---|---|
| `in` | Signal input |
| `out` | Signal output |
| `clk` | Clock signal input |

parameters:

| | |
|---|---|
| `Vtrans` | Clock trigger threshold voltage. |

Module name: `z_integrator_digi_(z`$^{-1}$`/ (1-z`$^{-1}$`)`

Function: one bit z integrator

pins (in out clk):

| | |
|---|---|
| `in_Signal` | input |
| `out_Signal` | output |
| `clk` | Clock signal input |

Module name: `quantizer_midtread_digi_(quantizer)`

Function: `quantizer`. The input signal will be quantized between `N*level_hi` and `N*level_lo`, the steps will be `N`. `[-N/2, N/2]` will be quantized as `0`.

pins (in out):

| | |
|---|---|
| `in` | input |
| `out` | output |

parameters:

| | |
|---|---|
| `N` | The quantizer resolution is `1/N`. `N` must be a power of 2. |
| `level_hi` | Highest quantized value is `N*level_hi`. |
| `level_lo` | Lowest quantized value is `N*level_lo`. |
| `tr` | Risetime and falltime of the output. |
| `td` | Delay time from input to output. |
| `tt` | Time tolerance of transition event. |

Module name: `SDM_MASH_3rd_111_macro`

Function: Mash 111 type 3rd SDM

pins (fset cp f f av):

| | |
|---|---|
| `fset` | Fraction part is set by this pin's voltage which should be 0~1. |
| `cp` | Clock signal input. |
| `f` | Output of the sigma-delta. It's default value is in the range of -3~4. |
| `f av` | Average value of pin f. |

parameters:

| | |
|---|---|
| `BIT` | Set the ADC bit in the module. |
| `Vtrans` | CP trigger threshold. |

| | |
|---|---|
| `outOffset` | Output value of the sigma-delta is in the range of -3+outOffSet ~ 4+outOffset. |

# 17

# Measuring AM, PM, and FM Conversion

## Derivation

Consider a sinusoid that is simultaneously both amplitude and phase modulated as in Equation 17-1.

**(17-1)** $$v_m(t) = A_c(1 + \alpha(t))\cos(\omega_c t + \phi_c + \phi(t))$$

In Equation 17-1, $A_c$, $\phi_c$, $\omega_c$, are the amplitude, phase and angular frequency of the carrier, while $\alpha(t)$, and $\phi(t)$ are the amplitude and phase modulation.

When you assume that $\phi(t)$ is small for all $t$, this allows the narrowband angle modulation approximation as in Equation 17-2. See [ziemer76].

**(17-2)** $$v_m(t) = A_c(1 + \alpha(t))[\cos(\omega_c t + \phi_c) - \phi(t)\sin(\omega_c t + \phi_c)]$$

Converting to complex exponentials gives Equation 17-3.

**(17-3)** $$v_m(t) = \frac{A_c}{2}(1 + \alpha(t)) \left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\phi(t)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Letting both the amplitude and phase modulation be complex exponentials with the same frequency, $\omega_m$, gives Equations 17-4, 17-5, 17-6, 17-7 and 17-8.

**(17-4)** $$\alpha(t) = Ae^{j\omega_m t}$$

**(17-5)** $$\phi(t) = \Phi\,\frac{e^{j\omega_m t} + e^{-j\omega_m t}}{2}$$

Where

**(17-6)** $$A = A_{Ae^{j\phi_A t}}$$

**(17-7)** $$\Phi = A_{\Phi e^{j\phi_\Phi t}}$$

**(17-8)** $$\begin{aligned} v_m(t) = {} & \frac{A_c}{2}\left(1 + Ae^{j\omega_m t}\right) \\ & \left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + \right. \\ & \left. j\frac{1}{2}\Phi\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right] \end{aligned}$$

Assuming that both A and $\Phi$ are small and neglecting cross modulation terms gives Equation 17-9.

**(17-9)** $$\begin{aligned} v_m(t) = {} & \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + \right. \\ & Ae^{j\omega_m t}e^{j(\omega_c t + \phi_c)} + Ae^{j\omega_m t}e^{-j(\omega_c t + \phi_c)} + \\ & j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)e^{j(\omega_c t + \phi_c)} - \\ & \left. j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)e^{-j(\omega_c t + \phi_c)}\right] \end{aligned}$$

Simplifying gives Equation 17-10.

**(17-10)**

$$
\begin{aligned}
v_m(t) &= \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} \\
&+ Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)} \\
&+ \frac{1}{2} j\Phi\Big[e^{j((\omega_m + \omega_c)\,t + \phi_c)} - e^{j((\omega_m - \omega_c)\,t - \phi_c)} \\
&+ e^{j((-\omega_m + \omega_c)t + \phi_c)} - e^{j((-\omega_m - \omega_c)t - \phi_c)}\Big]\Big]
\end{aligned}
$$

In Equation 17-10,

- The AM terms are

$$
Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)}
$$

- The PM terms are

$$
\begin{aligned}
&\frac{1}{2}\Big[j\Phi e^{j((\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((\omega_m - \omega_c)\,t - \phi_c)} \\
&+ j\Phi e^{j((-\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((-\omega_m - \omega_c)\,t - \phi_c)}\Big]
\end{aligned}
$$

Because the left-side term $v_m(t)$ represents a real-time signal, only the real parts of the complex terms are of interest. Dropping the imaginary parts and rearranging Equation 17-10 produces Equation 17-11,

**(17-11)**

$$
\begin{aligned}
v_m(t) &= \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} \\
&+ Ae^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} \\
&+ Ae^{j(\omega_m + \omega_c)\,t}e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)\,t}e^{j\phi_c}\Big]
\end{aligned}
$$

When you ignore the negative $\omega_m$ term in Equation 17-11, you get

- The LSB (lower sidebands) terms are

$$Ae^{j(\omega_m - \omega_c)t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)t}e^{-j\phi_c}$$

- The USB (upper sidebands) terms are

$$Ae^{j(\omega_m + \omega_c)t}e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)t}e^{j\phi_c}$$

Assume that you perform a PAC analysis, which applies a single complex exponential signal that generates responses at the upper and lower sidebands of the $\omega_c$ signal. Assume the transfer functions are $L$ and $U$, so the lower and upper sideband signals are given by Equations 17-12 and 17-13.

**(17-12)** $$l(t) = Le^{j(\omega_m - \omega_c)t}$$

**(17-13)** $$u(t) = Ue^{j(\omega_m + \omega_c)t}$$

Where

$$L = A_L e^{j\phi_L}$$

$$U = A_U e^{j\phi_U}$$

Matching common frequency terms between Equations 17-11, 17-12, and 17-13 gives Equations 17-14, 17-15, 17-16 and 17-17.

**(17-14)** $$L = \frac{A_c}{2}\left(Ae^{-j\phi_c} - j\Phi e^{-j\phi_c}\right)$$

**(17-15)** $$U = \frac{A_c}{2}\left(Ae^{j\phi_c} + j\Phi e^{j\phi_c}\right)$$

**(17-16)** $$\frac{2}{A_c}Le^{j\phi_c} = A - j\Phi$$

**(17-17)** $$\frac{2}{A_c}Ue^{-j\phi_c} = A + j\Phi$$

Solving for the modulation coefficients gives Equations 17-18 and 17-19.

**(17-18)** $$A = \frac{1}{A_c}\left(Le^{j\phi_c} + Ue^{-j\phi_c}\right)$$

**(17-19)** $$\Phi = \frac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)$$

Thus, Equation 17-18 gives the transfer function for amplitude modulation and Equation 17-19 gives the transfer function for phase modulation.

## Positive Frequencies

Notice that `L` is defined in Equation 17-12 on page 1028 to be the transfer function from the input to the sideband at $\omega_m - \omega_c$, which is a negative frequency. This is usually a natural definition for use with the Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) small signal analyses (depending on the setting of the `freqaxis` parameter). It can be cumbersome though when the only data available is at positive frequencies. Thus, the transfer function to $\omega_c - \omega_m$ is defined as

$$\tilde{L}$$

Then, as in Equation 17-20,

**(17-20)** $\tilde{l}(t) = \tilde{L}e^{j(\omega_c - \omega_m)t}$

Because the signals are real, L is a complex conjugate of

$\tilde{L}$

And the reverse is also true, as in Equation 17-21,

**(17-21)** $L = \tilde{L}^*$

Equations 17-22 and 17-23 are produced by rewriting Equation 17-18 on page 1029 and Equation 17-19 on page 1029 in terms of

$\tilde{L}$

**(17-22)** $A = \frac{1}{A_c}\left(\tilde{L}^* e^{j\phi_c} + Ue^{-j\phi_c}\right)$

**(17-23)** $\Phi = \frac{j}{A_c}\left(\tilde{L}^* e^{j\phi_c} - Ue^{-j\phi_c}\right)$

## FM Modulation

For FM modulation, the phase modulation $\phi(t)$ becomes the integral of the FM modulation signal, $\omega(t)$ as shown in Equation 17-24.

**(17-24)** $v_m(t) = A_c \cos(\omega_c t + \phi(t))$

Where

**(17-25)** $$\phi(t) = \int \omega(t) dt$$

Recall from Equation 17-5 on page 1026 and Equation 17-19 on page 1029 that

**(17-26)** $$\phi(t) = \frac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)e^{j\omega_m t}$$

Combining Equation 17-25 and Equation 17-26 and the differentiating both sides results in Equations 17-27, 17-28, and 17-29.

**(17-27)** $$\omega(t) = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)e^{j\omega_m t}$$

**(17-28)** $$\Omega = A_\Omega e^{j\phi_\Omega} = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)$$

Or

**(17-29)** $$\Omega = j\omega_m \Phi$$

# Simulation

The test circuit, represented by the two netlists shown in Example on page 1032 and Example on page 1032, was run with Spectre RF. The test circuit consists of three, linear, periodically-varying modulators that are driven with the same input. The input is constant valued in the large signal PSS analysis, and generates a single complex exponential analysis during the PAC analysis. The idea is to compute the transfer functions from this input to the upper and lower sidebands at the output of the modulators and then use the derivation just described to convert these transfer functions into transfer functions to the AM, PM, and FM modulations and then check the simulation results against the expected results.

Notice that `freqaxis=out`. This is necessary to match the derivation. If you would rather use `freqaxis=absout`, you would have to use the complex conjugate of $L$ as in Equation 17-22 on page 1030 and Equation 17-23 on page 1030.

### Netlist for the AM, PM, and FM Conversion Test Circuit

```
// AM, PM, and FM modulation test circuit

simulator lang=spectre
ahdl_include "modulators.va"

parameters MOD_FREQ=10MHz
parameters CARRIER_FREQ=1GHz

Vin (in 0) vsource pacmag=1 pacphase=0
Mod0 (unmod in) AMmodulator freq=CARRIER_FREQ mod_index=0
Mod1 (am in) AMmodulator freq=CARRIER_FREQ mod_index=1
Mod2 (pm in) PMmodulator freq=CARRIER_FREQ kp=1
Mod3 (fm in) FMmodulator freq=CARRIER_FREQ fd=MOD_FREQ

waves pss fund=CARRIER_FREQ outputtype=all tstab=2ns harms=1
xfer pac start=MOD_FREQ maxsideband=4 freqaxis=out
```

The netlist for the modulator models shown in Example 17-29, has the filename `modulators.va`.

### Netlist for the Modulator Models Written in Verilog-A

```
`include "discipline.h"
`include "constants.h"

module AMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real mod_index = 1;

    analog begin
        V(out) <+ (1+mod_index*V(in)) * cos(2*`M_PI*freq*$abstime);
        $bound_step( 0.05 / freq );
    end
endmodule

module PMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
```

```
    parameter real kp = 1 from (0:inf);

    analog begin
        V(out) <+ cos(2*`M_PI*freq*$abstime + kp*V(in));
        $bound_step( 0.05 / freq );
    end
endmodule

module FMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real fd = 1 from (0:inf);
    real phi;

    analog begin
        V(out) <+ cos(2*`M_PI*(freq*$abstime + idtmod(fd*V(in),0,1, -0.5)));
        $bound_step( 0.05 / freq );
    end
endmodule
```

# Results

The simulations were run with various values for `pacphase` on `Vin`.

Table 17-1 shows results for the output of the AM modulator with $v_{LO} = cos(\omega_c t)$.

Table 17-2 shows results for the output of the PM modulator with $v_{LO} = cos(\omega_c t)$.

**Table 17-1 Results for the AM Modulator Output**

| pacphase | L | U | A | $\Phi$ |
|---|---|---|---|---|
| 0 | 1/2 | 1/2 | 1 | 0 |
| 45 | $\frac{1+j}{2\sqrt{2}}$ | $\frac{1+j}{2\sqrt{2}}$ | $\frac{1+j}{2}$ | 0 |
| 90 | $j$/2 | $j$/2 | $j$ | 0 |
| 180 | –1/2 | –1/2 | –1 | 0 |

**Table 17-2 Results for the PM Modulator Output**

| pacphase | L | U | A | $\Phi$ |
|---|---|---|---|---|
| 0 | –1/2 | 1/2 | 0 | 1 |
| 45 | $\frac{1-j}{2\sqrt{2}}$ | $\frac{j-1}{2\sqrt{2}}$ | 0 | $\frac{1+j}{\sqrt{2}}$ |
| 90 | 1/2 | –1/2 | 0 | $j$ |
| 180 | $j$/2 | –$j$/2 | 0 | –1 |

If you repeat the simulations but replace the $cos$ function in the modulators with the $sin$ function, which is equivalent to changing the LO to $v_{LO} = sin(\omega_c t)$ or setting $\phi_c = -90$, you achieve the following results.

- Table 17-3 shows results for the output of the AM modulator with $v_{LO} = sin(\omega_c t)$.

- Table 17-4 shows results for the output of the PM modulator with $v_{LO} = sin(\omega_c t)$.

**Table 17-3 Results for the AM Modulator Output**

| pacphase | L | U | A | Φ |
|---|---|---|---|---|
| 0 | $j/2$ | $-1/2$ | 1 | 0 |
| 45 | $\frac{j-1}{2\sqrt{2}}$ | $\frac{1-j}{2\sqrt{2}}$ | $\frac{1+j}{\sqrt{2}}$ | 0 |
| 90 | $-1/2$ | $1/2$ | $j$ | 0 |
| 180 | $-j/2$ | $j/2$ | $-1$ | 0 |

**Table 17-4 Results for the PM Modulator Output**

| pacphase | L | U | A | Φ |
|---|---|---|---|---|
| 0 | $1/2$ | $1/2$ | 0 | 1 |
| 45 | $\frac{1+j}{2\sqrt{2}}$ | $\frac{1+j}{2\sqrt{2}}$ | 0 | $\frac{1+j}{\sqrt{2}}$ |
| 90 | $j/2$ | $j/2$ | 0 | $j$ |
| 180 | $-1/2$ | $-1/2$ | 0 | $-1$ |

Finally, Table 17-5 shows the results for the FM modulator with $v_{LO} = cos(\omega_c t)$. The FM modulator has a modulation coefficient of $\omega_m$ built-in, which renormalizes the results.

**Table 17-5 Results for the FM Modulator Output**

| pacphase | L | U | Ω |
|---|---|---|---|
| 0 | $-1/2$ | $1/2$ | 1 |
| 45 | $-\frac{1+j}{2\sqrt{2}}$ | $\frac{1+j}{2\sqrt{2}}$ | $\frac{1+j}{\sqrt{2}}$ |
| 90 | $-j/2$ | $j/2$ | $j$ |
| 180 | $1/2$ | $-1/2$ | $-1$ |

# Conclusion

This appendix shows that the PAC analysis can be used to determine the level of AM or PM modulation that appears on a carrier. This is done by applying a small signal and using the phase of the carrier along with the transfer function to the upper and lower sidebands of the carrier to compute an AM or PM transfer function.

# References

[Ziemer 76] R. Ziemer and W. Tranter. *Principles of Communications: Systems, Modulation, and Noise.* Houghton Miffin, 1976.

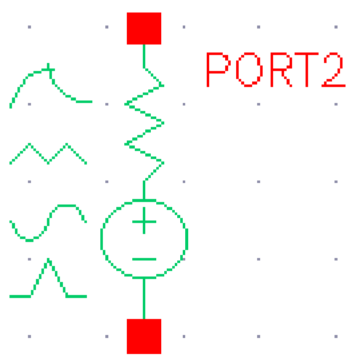
[Robins 96] W. Robins. *Phase Noise in Signal Sources (Theory and Application)*. IEE Telecommunications Series, 1996.

# 18

# Using the Port Component

You can use the *port* component, located in the *analogLib* library, in RF circuits for Virtuoso® Spectre® circuit simulator RF Analysis (Spectre RF) and Spectre S-parameter simulations.

The *port* component, located in the *analogLib* library, is similar to the existing *psin* component. The *port* component supports all the *Source types* of the Spectre *port* primitive: *pwl*, *pulse*, *sine*, *dc*, and *exp*.




The *port* component is an independent resistive source tied between positive and negative terminals. It is equivalent to a voltage source in series with a resistor, where the reference resistance of the *port* is the value of the resistor.

## Capabilities of the port Component

While the *port* component is most useful as a stimulus in high-frequency circuits, it also has the following unique capabilities.

It defines the ports of a circuit to the S-parameter analysis

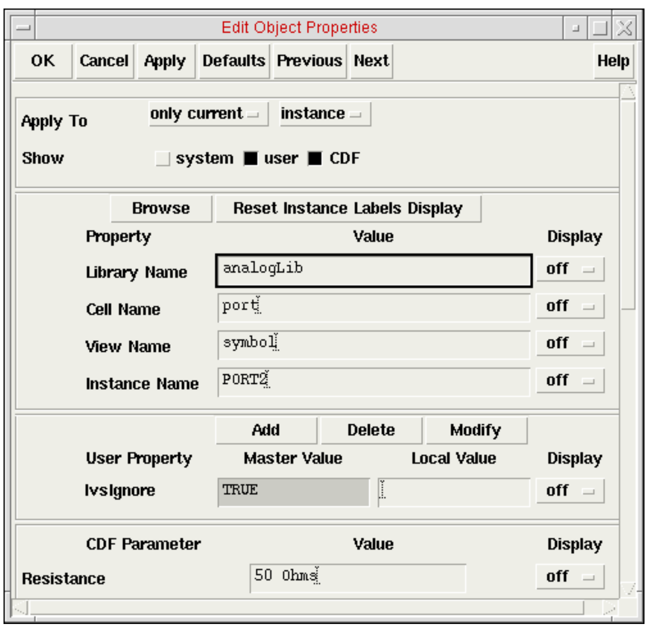
- It has an intrinsic noise source that lets the noise analysis directly compute the noise figure of the circuit
- Is the only source for which you can specify the amplitude in terms of power

# Terminating the Port

When you specify the voltage on a *port*, Spectre RF assumes that the port is properly terminated in it's reference resistance. The specified voltage value is not the voltage on the internal voltage source, which is actually set to twice the value specified on the `port`. If you use a `port` source to drive an open circuit, the voltage (for DC, transient, AC, and PAC signals) is double its specified value. However, you can alternatively specify the amplitude of the sine wave in the transient and PAC analyses as the power in dBm delivered by the *port* when terminated with the reference resistance.

The *port* component Edit Object Properties form is shown in Figure 18-1 on page 1038 and Figure 18-2 on page 1039.

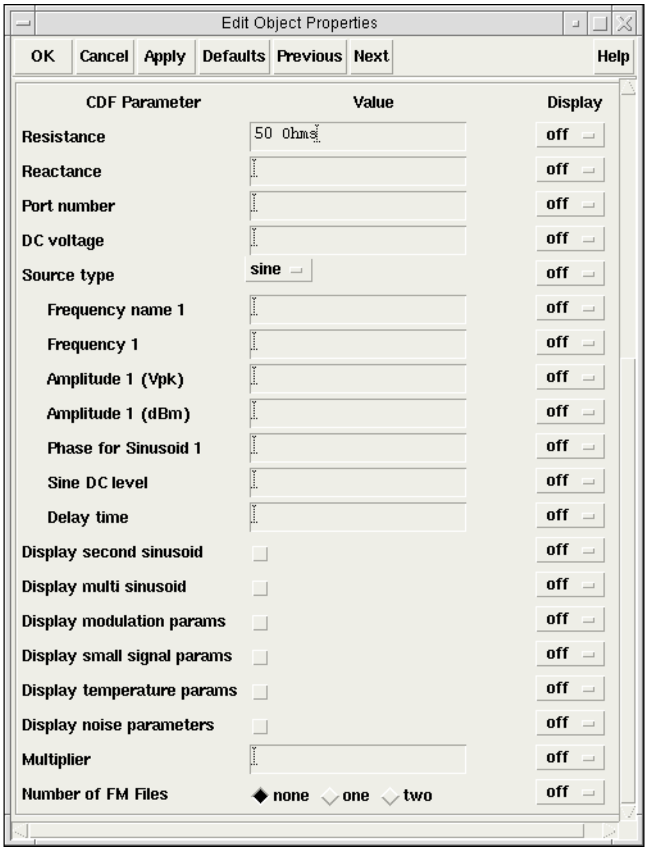
**Figure 18-1 Top of the port Component Edit Object Properties Form**




# Parameters for the Port Component

The *port* component's CDF parameters described here are grouped by parameter types, rather than in the order they appear on the *port* Edit Object Properties form.

### Figure 18-2 Bottom of the port Component Edit Object Properties Form




## Port parameters

Resistance

Reactance

Port number

Multiplier

Number of FM Files

## General waveform parameters

Source type

Delay time

## DC Waveform parameters

DC voltage

## Pulse waveform parameters

Zero value

One value

Period of waveform

Rise time

Fall time

Pulse width

## PWL waveform parameters

Waveform Entry Method

File name

Number of PWL/Time pairs

DC offset

Amplitude scale factor

Time scale factor

Breakpoints

Period

Transition Width

## Sinusoidal waveform parameters

Sine DC level

Frequency name 1

Frequency 1

Amplitude 1 (Vpk)

Amplitude 1 (dBm)

Phase for Sinusoid 1

Frequency name 2

Frequency 2

Amplitude 2 (Vpk)

Amplitude 2 (dBm)

Phase for Sinusoid 2

Sinusoid Frequency 1

Sinusoid Ampl 1 (Vpk)

Sinusoid Ampl 1 (dBm)

Sinusoid Phase 1

Sinusoid Maxharm 1

## Amplitude and Frequency modulation parameters

AM modulation index 1

AM modulation frequency 1

AM modulation phase 1

FM modulation index 1

FM modulation frequency 1

Damping factor 1

## Exponential waveform parameters

Delay time

Zero value

One value

Rise time start

Rise time constant

Fall time start

Fall time constant

## Small-signal parameters

PAC magnitude

PAC magnitude (dBm)

PAC phase

AC magnitude

AC phase

XF magnitude

## Temperature effect parameters

Linear temperature coefficient

Quadratic temperature coefficient

Nominal temperature

## Noise parameters

Noise Entry Method

Noise file name

Number of noise/freq pairs

## Port Parameters

Port parameters include *Resistance*, *Reactance*, *Port Number*, *Multiplier*, and *Number of FM Files*.

### Resistance

The reference resistance of the system. The value must be a real number, but not 0. The default value is 50 $\Omega$ (50 Ohms).

#### Reactance

The imaginary part of impedance, used for harmonic balance analyses only. The value must be a real number. Default: 0 Ω
Units: Ohms

#### Port number

The number associated with the *port*. The value must be a nonzero integer. Each *port* in a schematic must have a unique *Port number*. The *Port number* is not automatically indexed when you place a new *port* on your schematic.

#### Multiplier

The multiplicity factor specifies a number of *ports* in parallel. The value must be a nonzero real number and the default is 1. For example, if you set *Resistance* to `50` and *Multiplier* to `2`, you specify two *ports* in parallel, with an effective reference resistance of 25 Ω.

#### Number of FM Files

The number of files that contain the data for frequency modulated waveforms. FM I/Q signals can be written to one file or to two files with the I file first.

### General Waveform Parameters

The General Waveform parameters include *Source type* and *Delay time*.

#### Source type

The *Source type* parameter lets you select a wave shape for the port from the *Source type* cyclic field: *dc*, *pulse*, *exp*, *pwl*, *sine*, or blank. Each *Source type* has different parameter settings associated with it. You can define several different wave shapes and quickly switch between them without losing the wave shape settings. The wave shape settings are described in detail in the *DC*, *Pulse*, *Piecewise Linear*, *Sinusoidal*, and *Exponential* Waveform Parameters sections.

The typical *Source types* used in Spectre RF analyses are *dc*, *pulse*, and *sine*. For example, you can quickly switch from a sinusoid level (for PSS analysis) to a DC level (for PAC analysis) by changing the *Source type* from *sine* to *dc*.

When you set *Source type* to the blank value, the *port* acts as a resistive load.

#### Delay time

The *Delay time* is the amount of time that the source stays at the DC level before it starts generating waveforms (assuming the *Source type* is set to *sine*, *pulse*, *pwl*, or *exp*). The value must be a real number. The default value is 0 and the units are seconds.

### DC Waveform Parameters

To generate a dc waveform from the *port* component, select *dc* in the *Source type* cyclic field, as illustrated in Figure 18-3 on page 1045.

When the *Source type* is set to *dc*, the *dc* and *temperature effect* parameters are active.

The *dc* setting sets the DC level for all analyses. In DC analysis, this setting also determines the DC level generated by the source, regardless of what *Source type* you specify.

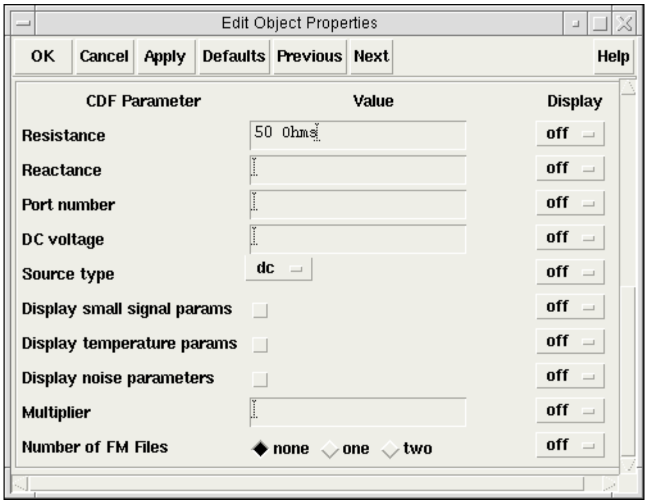
### DC voltage

The *DC voltage* parameter sets the port's DC level for DC analysis. The value must be a real number. If you do not specify the DC value, it is assumed to be the *time=0* value of the waveform. The default value is 0 and the units are Voltage.

The *DC voltage* parameter specifies the DC voltage across the *port* when it is terminated in its reference resistance. In other words, the *DC voltage* of the internal voltage source is double the specified DC value, *dc*. The same is true for the values for the *transient*, *AC*, and *PAC* signals from the `port`. However, you can alternatively specify the amplitude of the sine wave in the transient and PAC analyses as the power delivered in dBm by the *port* when it is terminated with the reference resistance.

Because all small signal analyses (AC, XF, and Noise) use DC analysis results, the *DC voltage* level also affects the small-signal analyses. Transient analysis is not affected unless you specify *Source* `type`*=dc* or use *dc* as a default for the other waveform types.

**Figure 18-3 Source type=dc in the Edit Object Properties form**



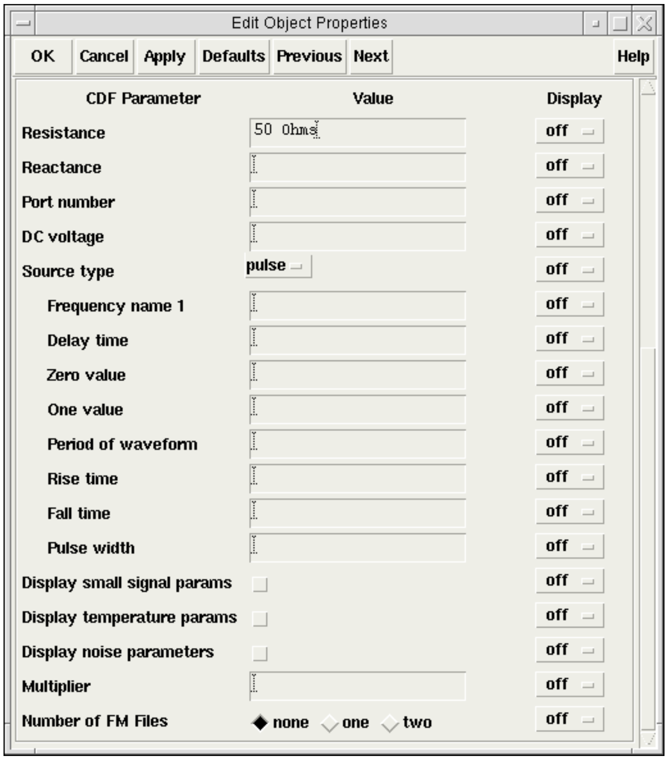

The *Display small signal params, Display temperature params,* and *Display noise parameters* fields are discussed in "Small-Signal Parameters" on page 1063, "Temperature Effect Parameters" on page 1065, and "Noise Parameters" on page 1061.

## Pulse Waveform Parameters

To generate a pulse waveform from the *port* component, select *pulse* in the *Source type* cyclic field.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

#### Figure 18-4 Source type=pulse in the Edit Object Properties form




### Frequency name 1

The *Frequency name 1* parameter is described in [“Frequency name 1”](#) on page 1051.

### Delay time

The *Delay time* parameter is described in [“Delay time”](#) on page 1044

### Zero value

The *Zero value* parameter (*val0*) is used with the *pulse* and *exp* waveforms. The default value is 0 and the units are Voltage.

### One value

The *One value* parameter (*val1*) is used with the *pulse* and *exp* waveforms. The default value is 1 and the units are Voltage.

#### Period of waveform

The *period* parameter of the *pulse* waveform. The default value is infinity and the units are seconds.

#### Rise time

The *Rise time* parameter of the *pulse* waveform is the time for the transition from the *Zero value* to the *One value*. The units are seconds.

#### Fall time

The *Fall time* parameter of the *pulse* waveform is the time for the transition from the *One value* to the *Zero value*. The units are seconds.

#### Pulse width

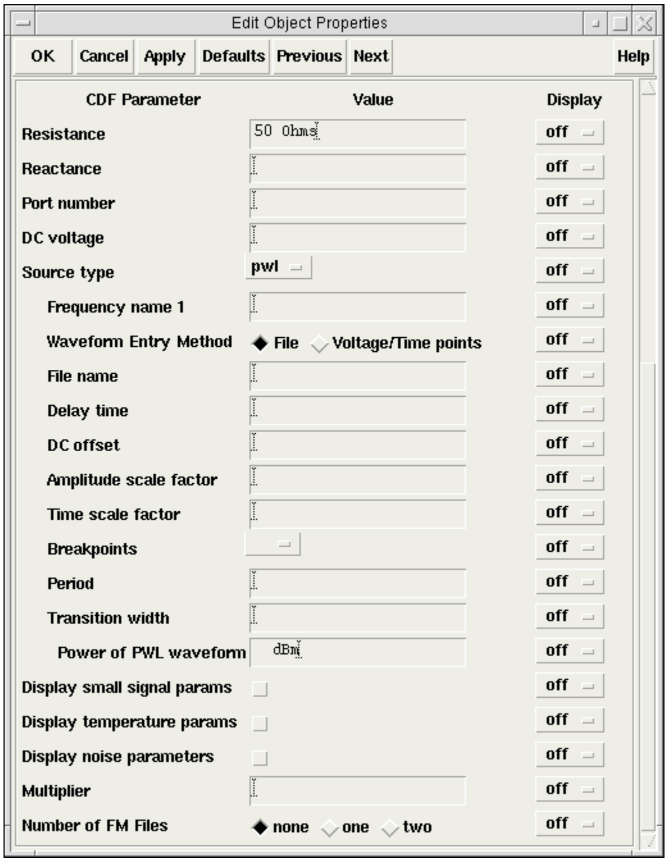
The *Pulse width* parameter of the *pulse* waveform is the width, or duration of the *One value*. The default value is infinity and the units are seconds.

### PWL Waveform Parameters

To generate a piecewise linear waveform from the *port* component, select *pwl* in the *Source type* cyclic field. This sets the *Source type* CDF parameter to *pwl* and displays additional fields for the PWL CDF parameter settings.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

**Figure 18-5 Source type=pwl in the Edit Object Properties form**




## Waveform Entry Method

With the *Waveform Entry Method* buttons, select how you enter piecewise-linear data,

- By specifying a *File name*.
- By entering a series of *Voltage/Time points*.

### File name

When you select *File* as the *Waveform Entry Method*, type the name of the file containing your piecewise-linear data in the *File name* field,. The *File name* must be a string. There is no default.

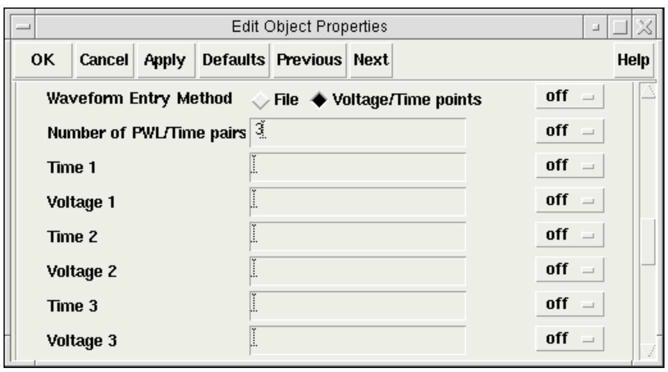
In your file, list the piecewise-linear data in the form of time-value pairs. Enter one pair per line with a space or tab between the time and voltage values. The numbers in the file must be simple numbers. You cannot use SI scale factors (p, n, u, m, k, M, G, etc.).

### Number of PWL/Time pairs

When you select *Voltage/Time points* as the *Waveform Entry Method*, the *Number of PWL-Time pairs* field (tvpairs) opens. Enter the number of time-value pairs you plan to enter. The form expands to let you enter the designated number of *Time* and *Voltage* values. Units are seconds and volts. The default is 0 and the maximum value is 50.

In this example, the number of voltage-time pairs is 3.

#### Figure 18-6 Waveform Entry Method=Voltage/Time points




The *DC offset, Amplitude scale factor,* and *Time scale factor* parameter fields let you quickly adjust the amplitude, frequency, and offset of your piecewise-linear data pairs without editing each individual time-value pair in the *pwl* waveform.

#### *DC offset*

*DC offset* (*offset*) for the *pwl* waveform. Default: 0 Units: V

#### *Amplitude scale factor*

*Amplitude scale factor* (*scale*) for the *pwl* waveform. Default: 1

#### *Time scale factor*

*Time scale factor* (*stretch*) for the time given for the *pwl* waveform. Default: 1

#### *Breakpoints*

Possible values are *no*, *yes*, or blank. If you set *Breakpoints* to *yes*, you force Spectre RF to place time points at each point specified in a *pwl* waveform during a transient analysis. This can be very expensive for waveforms with many points. If you set *Breakpoints* to *no*, Spectre RF inspects the waveform, looking for abrupt changes, and forces time points only at those changes. If you set *Source type = pwl* and set *Breakpoints* to blank, the default is *yes* if the number of points you specify is less than 20.

#### *Period*

The *pwl* waveform is periodic if you specify *Period* (*pwlperiod*). Units: seconds

If the value of the waveform you specify is not exactly the same at both its beginning and its end, then you must provide a nonzero value for *Transition Width*.

#### *Transition Width*

*Transition width* (*twidth*) is used when making *pwl* waveforms periodic. Default: PWL period/1000. Units: seconds

Before repeating, the waveform changes linearly in an interval of *Transition Width* from its value at *(Period – Transition Width)* to its value at the beginning of the waveform. Thus the *Transition Width* must always be less than the *Period*.

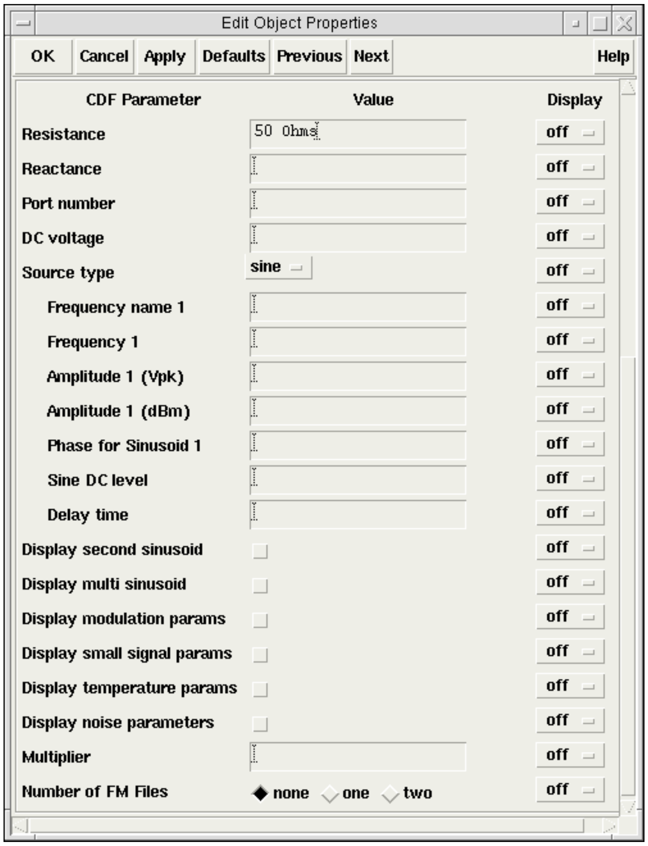
## Sinusoidal Waveform Parameters

The *port* component can generate up to two sinusoids simultaneously. They are denoted as 1 and 2. You can set the amplitude, frequency, and phase for both individually. The amplitude can be set to either a voltage or a power level. You can also specify sinusoidal AM or FM modulation of sinusoid 1.

The Edit Object Properties form for *Source type=sine* is shown in Figure 18-7 on page 1051.

The first sinusoid is described by the parameters *Frequency name 1, Frequency 1, Amplitude 1 (Vpk)*, *Amplitude 1 (dBm)*, *Phase for Sinusoid 1*, *Sine DC level*, *Damping factor 1*, and by AM or FM modulation terms.

### Figure 18-7 Source type=sine in the Edit Object Properties form




### Frequency name 1

Names the fundamental tones of sinusoid 1. After you save the schematic, the names you assign appear in the *Fundamental Tones* list box on the Choosing Analyses form.

*Frequency 1* is the frequency of the first sinusoidal waveform (carrier frequency). You typically use unmodulated signals in Spectre RF analyses. The value must be a real number. Default: 0 Units: Hz

### Amplitude 1 (Vpk)

The peak amplitude of the first sinusoidal waveform that you generate. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (Vpk)* has a value, the *Amplitude 1 (dBm)* field is grayed out. The value must be a real number. Default: 1 Units: V

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

### Amplitude 1 (dBm)

*Amplitude 1 (dBm)* is the amplitude of the first sinusoidal waveform, in dBm. The value specified is the power delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (dBm)* has a value, the *Amplitude 1 (Vpk)* field is grayed out. The value must be a real number. Units: dBm

### Phase for Sinusoid 1

The phase at the specified *Delay time*. To achieve a specified phase and still remain continuous, the sinusoidal waveform might start before the given *Delay time*. For example, if you want to generate a cosine wave, set this parameter to $90^{o}$. The value must be a real number. Default: 0  Units: degrees

### Sine DC level

Sets the DC level for sinusoidal waveforms in transient analyses. This parameter is used when the sinusoid has a different average level than the one specified for the DC analyses. If not specified, the average value of the sinusoid is the same as that of the DC level of the source. The value must be a real number. Default: dc Units: V

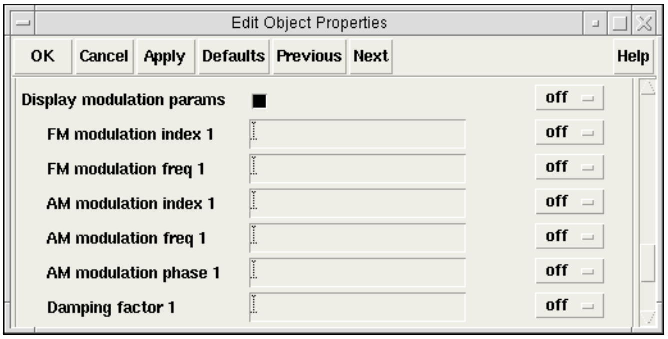
## Modulation Parameters

### Display Modulation Parameters

When selected, the form expands and the following modulation parameters are displayed: *FM modulation index 1, FM modulation freq 1, AM modulation index 1, AM modulation freq 1, AM modulation phase 1,* and *Damping factor 1*.

Only the first sinusoid can be modulated.

#### Figure 18-8 Display modulation params




### FM Modulation (Background Information)

The frequency modulation for the sinusoidal case is defined as

$v_{FM}(t) = A \sin(2\pi f_c t + \beta \sin(2\pi f_m t) + \phi)$

where

- A is the amplitude of sinusoid 1
- $\beta$ is the FM modulation index
- $\sin(2\pi f_m t)$ is the modulation signal
- $f_c$ is the carrier frequency
- $\phi$ is the *phase for sinusoid 1*

The frequency modulation parameters affect only the first sinusoid generated by *port*. They have no effect on the second sinusoid.

### FM modulation frequency 1

FM modulation frequency for the sinusoidal waveform ($f_m$ in the previous equation). The value must be a real number. Default: 0 Units: Hz

### FM modulation index 1

FM index of modulation for the sinusoidal waveform, the ratio of peak frequency deviation divided by the center frequency ($\beta$ in the above equations).

$\beta = \Delta f / f_m$

The value must be a real number. Default: 0

## Effect of Amplitude Modulation (Background Information)

The amplitude modulation (double sideband suppressed carrier, or DSB-SC) is defined as

$$v_{AM}(t) = A\ (1 + m\ \sin(2\pi f_m t + \phi))\ \sin(2\pi f_c t)$$

where

- A is the carrier amplitude (amplitude of sinusoid 1)
- m is the AM modulation index
- $f_m$ is the AM modulation frequency
- $\phi$ is the AM modulation phase
- $\sin(2\pi f_c t)$ is the carrier signal

The amplitude modulation parameters affect only the first sinusoid generated by *port*. They have no effect on the second sinusoid.

### AM modulation frequency 1

AM modulation frequency for the first sinusoidal waveform ($f_m$ in the previous equation). The value must be a real number. Default: 0 Units: Hz

### AM modulation phase 1

AM phase of modulation for the first sinusoidal waveform ($\phi$ in the previous equation). The value must be a real number. Default: 0 Units: degrees

### AM modulation index 1

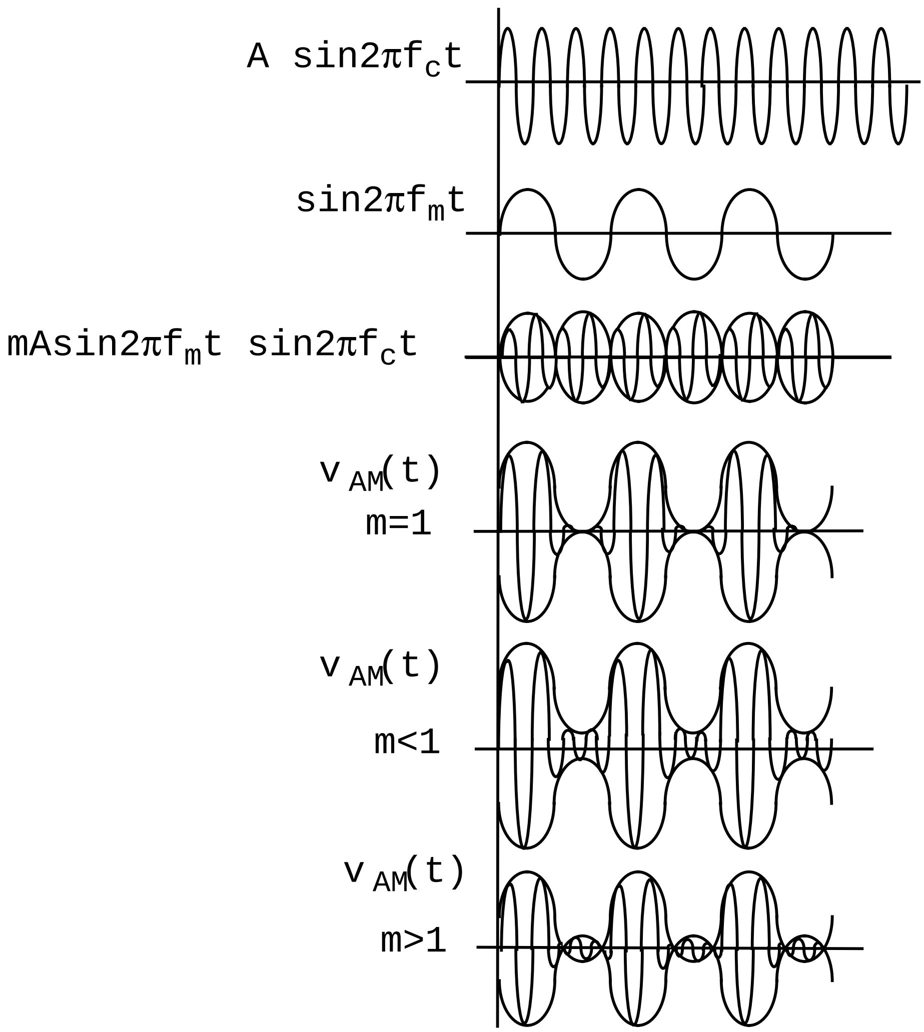
AM index of modulation for the first sinusoidal waveform (m in the previous AM equation). The *AM modulation index 1* is a dimensionless scale factor used to control the ratio of the sidebands to the carrier.

```
m = (peak_DSB-SC_amplitude)/(peak_carrier_amplitude)
```

The value must be a real number. Default: 0

The following figure shows the effect of varying modulation indexes for the following three cases: *m < 1*, *m = 1*, and *m > 1*. $f_c$ is the carrier frequency, and $f_m$ is the modulation frequency.

**Figure 18-9 Amplitude Modulation: Effects of Varying Modulation Indexes**




### Damping factor 1

Damping factor for the sinusoidal waveform. *Damping factor 1* specifies the time it takes to go from the envelope (full amplitude) at *time*=0 to 63 percent of the full amplitude. For example, consider the following damped sinusoid:

$v(t) = A\, e^{-\sigma t} \sin(2\pi f t + \phi)$

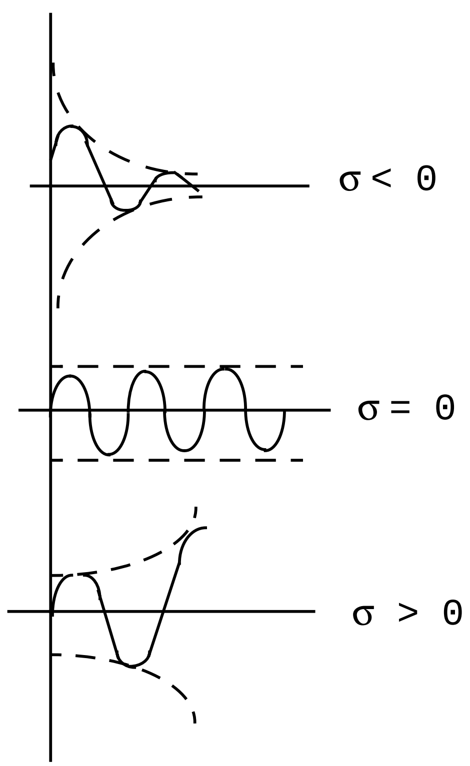
where

$\sigma$ = *Damping factor 1*, `A` is the amplitude of sinusoid 1, and $\phi$ is the *Phase for sinusoid 1*

- If $\sigma = 0$, the waveform is a pure sinusoid (steady state).
- If $\sigma < 0$, the waveform exhibits decaying oscillations.
- If $\sigma > 0$, the waveform exhibits growing oscillations.
- It takes $5\sigma$ to diminish to 1 percent of the peak amplitude. The value must be a real number. Default: 0. Units: 1/seconds

The following figure shows the effect of *Damping factor 1* on the first sinusoid for three values of $\sigma$.

**Figure 18-10 Effect of Damping Factor 1 on the First Sinusoid**



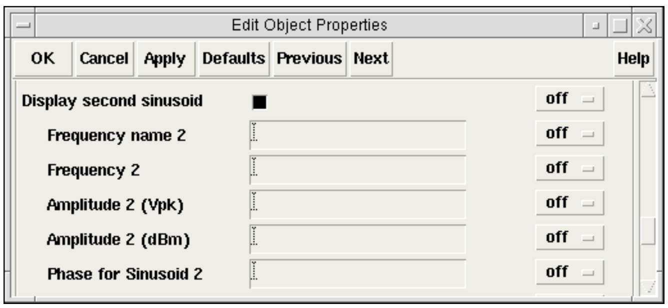

## Display second sinusoid

Displays the CDF parameters for the second sinusoid in the Edit Object Properties and Add Instance forms. When selected, the form expands to show the following CDF parameters:

*Frequency name 2, Frequency 2, Amplitude 2 (Vpk), Amplitude 2 (dBm),* and *Phase for Sinusoid 2*.

**Note:** The second sinusoid cannot be modulated.

**Figure 18-11 Display second sinusoid**




### Frequency name 2

Name for the second sinusoid. After you save the schematic, the name you assign appears in the *Fundamental Tones* list box on the Choosing Analyses form.

*Frequency 2* is the frequency of the second sinusoidal waveform. The value must be a real number. Default: 0 Units: Hz

### Amplitude 2 (Vpk)

Peak amplitude of the second sinusoidal waveform. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 2 (Vpk)* or *Amplitude 2 (dBm)*, but not both. If *Amplitude 2 (Vpk)* has a value, the *Amplitude 2 (dBm)* field is grayed out. The value must be a real number. Default: 1 Units: V

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

### Amplitude 2 (dBm)

*Amplitude 2 (dBm)* is the amplitude of the second sinusoidal waveform, in dBm. The value specified is the power delivered into a matched load. You can select either *Amplitude 2 (Vpk)* or *Amplitude 2 (dBm),* but not both. If *Amplitude 2 (dBm)* has a value, the *Amplitude 2 (Vpk)* field is grayed out. The value must be a real number. Units: dBm

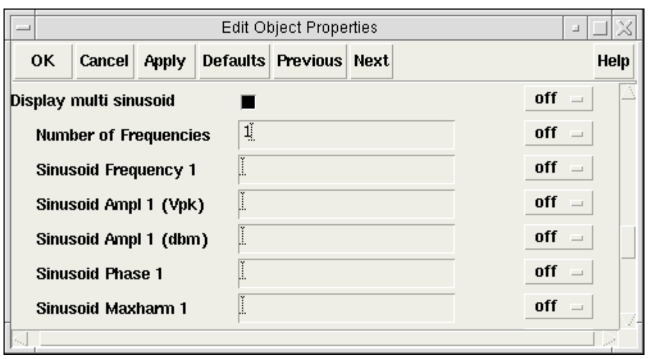
### Phase for Sinusoid 2

The phase at the specified *Delay time* for the second sinusoid. To achieve specified phase while still remaining continuous, the sinusoidal waveform might start before the given *Delay time*. The value must be a real number. Default: 0 Units: degrees

## Display multi sinusoid

Displays the CDF parameters for multiple sinusoids in the Edit Object Properties form. When selected, the form expands to show the *Number of Frequencies* field and for each one of the frequencies, a set of parameters patterned after: *Sinusoid Frequency 1, Sinusoid Ampl 1 (Vpk), Sinusoid Ampl 1 (dBm),* *Sinusoid Phase 1, Sinusoid Maxharm 1*.

**Figure 18-12 Display multi sinusoid**




### Number of Frequencies

Number of sinusoid frequencies to be specified.

### Sinusoid Frequency 1

The frequency of the first sinusoidal waveform (carrier frequency). The value must be a real number. Default: 0 Units: Hz

### Sinusoid Ampl 1 (Vpk)

The peak amplitude of the sinusoidal waveform that you generate. The value must be a real number. Default:1 Units: V

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Use either *Sinusoid Ampl 1 (Vpk)* or *Sinusoid Ampl 1 (dBm)* but do not set both.

### Sinusoid Ampl 1 (dBm)

The amplitude of the first sinusoidal waveform when specified in dBm. The value must be a real number. Units: dBm.

Use either *Sinusoid Ampl 1 (Vpk)* or *Sinusoid Ampl 1 (dBm)* but do not set both.

### Sinusoid Phase 1

The phase at the specified delay time. To achieve a specified phase and still remain continuous, the sinusoidal waveform might start before the given delay time. For example, the generate a cosine wave, set this parameter to 90 ° . The value must be a real number. Default: 0 Units: degrees.

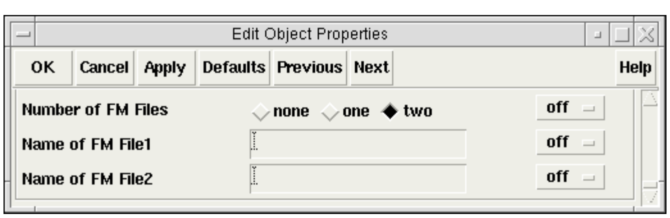
### Sinusoid Maxharm 1

An array of the number of harmonics of each fundamental to consider for each fundamental.

## Number of FM Files

Specifies the number of FM files used to hold FM waveform I/Q signals (none, one, or two) and expands the form to show the following parameters: *Name of FM File1,* and *Name of FM File2*.

**Figure 18-13 Number of FM Files with Two Selected.**



### Name of FM File1

Name of an FM file containing data for frequency modulated waveforms for a sinusoidal source.

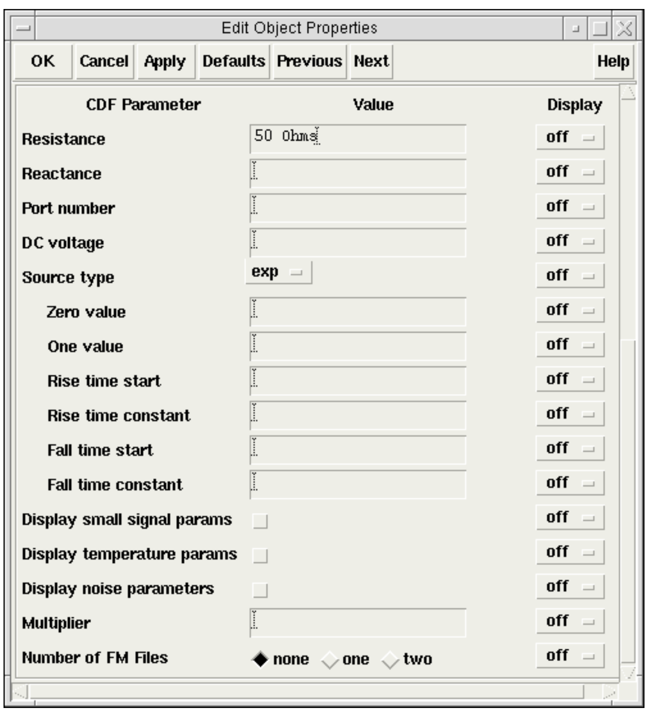
### Name of FM File2

Name of a second FM file containing data for frequency modulated waveforms for a sinusoidal source.

## Exponential Waveform Parameters

To generate an exponential waveform from the `port` component, set the CDF parameter *Source type*=*exp*, as shown in the next figure.

**Figure 18-14 Source type=exp in the Edit Object Properties form**




When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

#### Zero value

The *Zero value* (*val0*) used in *pulse* and *exp* waveforms. Default: 0 Units: V

#### One value

The *One value* (*val1*) used in *pulse* and *exp* waveforms. Default: 1 Units: V

#### Rise time start

The *Rise time start* (*td1*) for the *exp* waveform. Default: 0 Units: seconds

#### Rise time constant

The *Rise time constant* (*tau1*) for the *exp* waveform. Units: seconds

#### Fall time start

The *Fall time start* (*td2*) for the *exp* waveform. Units: seconds

#### Fall time constant

The *Fall time constant* (*tau2*) for the *exp* waveform. Units: seconds

### Noise Parameters

The noise parameters include *Noise temperature*, *Noise Entry Method*, *Noise file name*, and *Number of Noise Frequency Pairs*.

#### Noise temperature

The *Noise temperature* of the *port*. If not specified, the *Noise temperature* is assumed to be the actual temperature of the *port*. When you compute the noise figure of a circuit driven at its input by a *port*, set the *Noise temperature* of the *port* (Spectre parameter `noisetemp`) to 16.85C (290K). This setting matches the standard IEEE definition of noise figure. In addition, disable all other sources of noise in the *port*, such as the Spectre parameters `noisefile` and `noisevec`. If you want a noiseless *port*, set the *Noise temperature* to absolute zero or below, and do not specify a noise file or noise vector. Default: Actual temperature of the port. Units: $^{o}$C

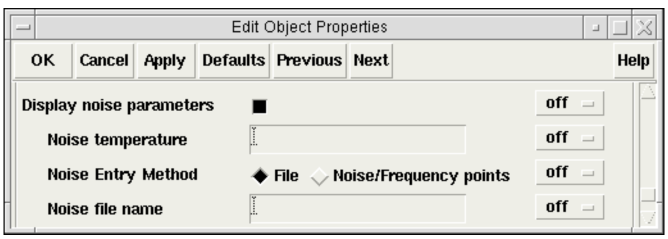
### Noise Entry Method

You can select one of two ways to enter noise data, either by specifying a *File* name or entering a series of *Noise/Frequency points*.

### Noise file name

If *Noise Entry Method = File* is selected, you enter the name of the file containing the excess spot noise data in the form of frequency-noise pairs. In your file, list the frequency-noise pairs as one pair per line with a space or tab between the frequency and noise values. The value must be a string. Default: none

**Figure 18-15 Display noise parameters: Noise Entry Method=File**

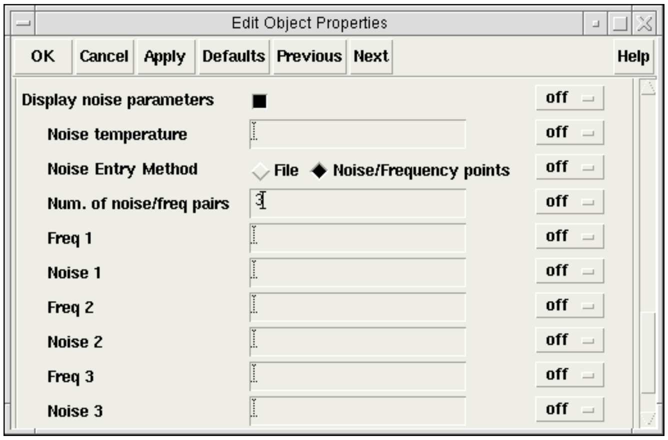



### Num. of noise/freq pairs

If the *Noise Entry Method=Noise/Frequency points*, you specify the *Number of noise/freq pairs*. The form expands to let you type in the designated *Freq* and *Noise* values. The noise values must be in $V^2$/Hz, and frequency in Hz. Default: 0 Maximum value: 10

The example in the next figure has the *Number of noise/freq pairs* set to 3.

**Figure 18-16 Display noise parameters: Noise Entry Method=Noise/Frequency Points**

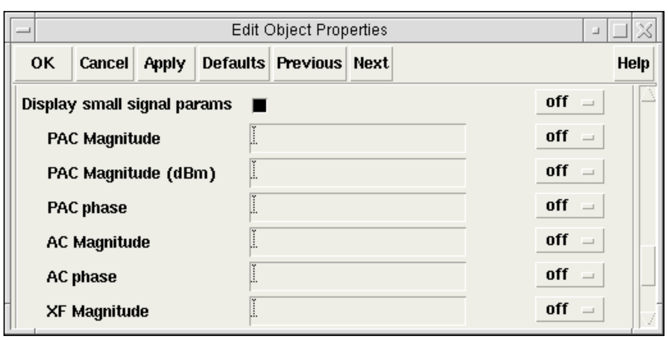



## Small-Signal Parameters

### Display small signal params

If selected, the Edit Object Properties/Add Instance form expands to show the small-signal parameters *PAC Magnitude*, *PAC Magnitude (dBm)*, *PAC phase*, *AC Magnitude*, *AC phase*, and *XF Magnitude*.

**Figure 18-17 Display small signal params**




When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*. The same is true for the values for the transient, AC, and PAC signals. However, the amplitude of the sine wave in the PAC and transient analysis can alternatively be specified as the power in dBm delivered by the `port` when terminated with the reference resistance.

#### PAC Magnitude

The peak periodic AC analysis magnitude. Setting this value to unity is a convenient way of computing the transfer function from this source to the output.

You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude* has a value, the *PAC magnitude (dBm)* field is grayed out. The value must be a real number. Default: 0 Units V

#### PAC Magnitude (dBm)

The periodic AC analysis magnitude in dBm (alternative to *PAC magnitude*). You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude (dBm)* has a value, the *PAC magnitude* field is grayed out. The value must be a real number. Units: dBm

#### PAC phase

The periodic AC analysis phase. The value must be a real number. Default: 0  Units: degrees

Typically, only one source in the circuit has a *PAC magnitude* set to a value other than zero, and usually it has a *PAC magnitude*=1 and *PAC phase*=0. However, there are situations where more than one source has a nonzero *PAC magnitude*. For example, applying a differential small-signal input could be done with two sources with the *PAC magnitude*s set to 0.5 and the *PAC phase*s set to 0 and 180.

You do not specify the PAC frequency in the *port* Edit Object Properties form. Instead, you set the PAC frequency in the PAC *Choosing Analyses form*. For example, when making an IP3 measurement, you set the PAC frequency to a variable value in the Choosing Analyses Form. Then, you enter the same variable in the *PAC Amplitude* (or *PAC Amplitude dBm*) field of the *port* Edit Object Properties form.

#### AC Magnitude

The peak small-signal voltage. The value must be a real number. Default: 0 Units: V

#### AC phase

The small-signal phase. The value must be a real number. Default: 0 Units: degrees

Typically, only one source in the circuit has *AC Magnitude* set to a value other than zero, and usually it has an *AC magnitude*=1 and *AC phase*=0. However, there are situations where more than one source has a nonzero *AC magnitude*. For example, you can apply a differential small-signal input with two sources with the *AC magnitude*s set to 0.5 and the *AC phase*s set to 0 and 180.

### XF Magnitude

The transfer function analysis magnitude. Use *XF magnitude* to compensate for gain or loss in the test fixture. The value must be a real number. Default: 1 Units: V/V

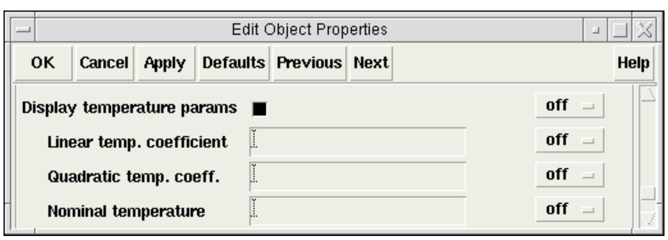
## Temperature Effect Parameters

Temperature effect parameters include the *Linear temperature coefficient*, *Quadratic temperature coefficient*, and *Nominal temperature*.

### Display temperature params

If selected, the Edit Object Properties/Add Instance form expands and the following three parameters appear in the form: *Linear temp. coefficient, Quadratic temp. coeff.,* and *Nominal temperature*.

**Figure 18-18 Display temperature params**




### Linear temp. coefficient

First order (linear) temperature coefficient of the *DC voltage* (`tc1`). The value must be a real number. Default: 0 Units: $^{o}C^{-1}$

### Quadratic temp. coeff.

Second order (quadratic) temperature coefficient of the *DC voltage* (`tc2`). The value must be a real number. Default: `0` Units: $^{o}C^{-2}$

#### Nominal temperature

The *Nominal temperature* for *DC voltage* (*tnom*). The value must be a real number. Default: Set by options specifications. Units: $^{o}$C

### How Temperature Parameters Affect the Voltage Level (Background Information)

The value of the *DC voltage* can vary as a function of the temperature if you specify *tc1* and *tc2*. The variation is given by

$V_{DC}(T)$ = dc * [1 + tc1 * ($T$ – tnom) + tc2 * ($T$ – tnom)$^2$]

where *T* is the analysis temperature specified in the analysis options, *tnom* is the *Nominal temperature* specified in the *Choosing Analyses* form, and dc is the *DC voltage*.

If the analysis temperature equals the nominal temperature, the result is the voltage amplitude that you specified, V(*T*)=dc.

If the nominal and analysis temperatures differ, the voltage amplitude is given by

$V_{DC}(T)$ = dc * [1 + tc1 * ($T$ – tnom) + tc2 * ($T$ – tnom)$^2$]

where *T* is the analysis temperature you specify in the analysis options and tnom is the nominal temperature, *tc1* and *tc2* are the *Linear* and *Quadratic temperature coefficients*, and *dc* is the *DC voltage*.

For example, if the *Nominal temperature* is 27$^{o}$C and the analysis temperature is 25$^{o}$C, there is a 2$^{o}$ difference between the nominal and analysis temperature. The voltage amplitude is

$V_{DC}(T)$ = dc * [1 + tc1 * (–2) + tc2 * (–2)$^2$]

### Additional Notes

#### Active Parameters in Analyses

In DC analyses, the only active parameters are *dc, m*, and the temperature coefficient parameters.

In AC analyses, the only active parameters are *m, mag,* and *phase*.

In transient analyses, all parameters are active except the small-signal parameters and the noise parameters.

In PAC, the only active parameters are *m*, *PAC magnitude* (amplitude or dBm), and *PAC phase*.

*XF magnitude* is active in XF and PXF analyses only.

# 19

# Using Tabulated S-parameters

Many passive component models commonly used in RF designs are available only as tables of S-parameter data. You can completely characterize any linear, time-invariant circuit network or component by specifying its S-parameter network at each frequency of interest. See "Using the nport Component" on page 1070 for information on how to use the S-parameter data tables as input for RF analyses.

Virtuoso® Spectre® circuit simulator RF analysis (Spectre RF) uses time-domain shooting methods to achieve excellent performance on large, highly nonlinear circuits. Consequently, using S-parameter data in Spectre RF is not as straightforward as it is with frequency-domain simulators such as those based on harmonic balance. Before you perform an RF simulation, such as a PSS analysis, you must first convert the frequency-domain S-parameter data to an equivalent time-domain model. For large, complicated S-parameter data sets this conversion can be time-consuming. However, you can avoid repeating the time consuming conversion process. You can convert the S-parameter data set to a time-domain model only once and then use one of the *nport* components to read in the converted time-domain data set multiple times. Thus, you avoid converting the S-parameter data set more than once. The procedure is described in "Model Reuse" on page 1076.

You might encounter three potential difficulties while converting a frequency-domain S-parameter data set into an equivalent time-domain model.

- Some frequency-domain S-parameter data sets do not have valid time-domain descriptions. Time-domain models must be stable and causal and you cannot generate a time-domain model for a frequency-domain data that lacks these properties.

- When you convert the frequency-domain data to an equivalent time-domain model, the frequencies between the tabulated data points must be interpolated. This is because the time-domain description of the frequency-domain model depends on *every* frequency not just the frequencies given at the tabulated data points. A special, robust, high-order, rational interpolation algorithm performs the interpolation. The rational interpolation process introduces some error into the final model description. See "Controlling Model Accuracy" on page 1072 to understand and control this error. This section also describes how to deal with data that contains noise that might corrupt the rational interpolation process.

- Any algorithm that converts S-parameter data for use in a time-domain simulator must extrapolate the data outside the range of tabulated frequencies. By definition, some frequencies are not included in a tabular data file. In addition, extrapolation might introduce nonphysical effects into the model, particularly when the S-parameter data is given over a very narrow frequency range. See "Troubleshooting" on page 1075 for information on how to diagnose and solve any problems. That section also describes how to interpret the warning messages that Spectre RF produces when a non-physical extrapolation might be occurring.

# Using the nport Component

Use the *nport* component to read in S-parameter data. Follow these simplified steps to prepare the component for use in Spectre RF simulations. For more complete information, see the Cadence documents *Virtuoso Simulator Circuit Components and Device Models Manual* and *Using the nport Component from the analogLib Library Application Note*.

1. Select the *nport* component from the *analogLib*.

   The *n1port*, *n2port*, *n2port*, and *n4port* components are provided only for backward compatibility.

2. Place the *nport* component in the Schematic window.

3. In the Schematic window, highlight the new component and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *nport* component appears.

4. In the *Number of Ports* field, specify the number of ports you want to use.

   For example, type 2 to create a 2 port component.

5. In the *Multiplier* field, enter the multiplicity factor. The default value is 1.

6. In the *Scale factor* field, enter the frequency scale factor. The default value is 1.

7. Set the *interpolation method* cyclic field.

   The *linear* and *spline* values are usually preferable to *rational* but before making a choice you should be familiar with the information on nports found in the Cadence documents *Virtuoso Simulator Circuit Components and Device Models Manual* and *Using the nport Component from the analogLib Library Application Note*.

   When you select *rational*, four new fields are displayed. For information about these fields, see the application note.

- The *ROM data file* field specifies an absolute pathname for the time-domain reduced order model data file (ROM) you are creating for the converted S-parameter data.
- The *Relative error* field specifies the maximum relative tolerance.
- The *Absolute error* field specifies the maximum absolute tolerance.
- The *Rational order* field specifies the order of rational function to use in fitting the S-parameter data.

**8.** In the *No. of Harmonics for PSS* field, enter the number of harmonics to consider in the PSS solution. The default value is 20.

**9.** In the *Thermal Noise* cyclic field, select *yes* or *no*.

**10.** In the *Use smooth data windowing* cyclic field, select *yes* or *no*.

Use `yes` if you are not setting the passivity check to yes for any S-parameter data files that do not have sufficient information to describe a physical system. Also use `yes` for all-pass type systems, where S-parameters are non-zero for the frequencies beyond `fmax`. Do not set *Use smooth data windowing* if you also set *Causality*.

**11.** In the *S-parameter data format cyclic* field, select the data format of your S-parameter data file if desired, or take the default. Spectre will parse the file and automatically read it if it's in spectre, Touchstone, or Citi format.

**12.** In the *Thermal noise model* cyclic field, select *internal* or *external.*

- Select *internal* to specify use of the internal thermal noise model.
- Select *external* to use the noise data in the S-parameter data file.

By default, Spectre RF uses external data whenever it is available.

**13.** In the *S-parameter data file* field, type the S-parameter data file name.

For example, type *sparam1.dat*.

**14.** The *DC extrapolation* can be set to `constant` or `unwrap`. `Constant` projects the first point down to zero frequency at exactly the same level. `Unwrap` does an estimation based on the first fiew points in the file.

**15.** The *High frequency extrapolation* field can be set to constant or linear. Constant maintains the same amplitude as the last point in the S-Paramater file to infinite frequency. Linear projects the last 2 points in the file to infinite frequency. Do not use linear if you set causality to `fmax` or `auto`.

**16.** The *Check Passivity selection* can be set to no, check, or enforce. No does not perform a passivity check. Check forces a passivity check and reports whether the network is passive in the spectre.out file. Enforce performs a passivity check and changes the transfer function in the S-Parameter file to make the transient model passive. It is recommended that yes be selected for all S-Parameter files that describe passive networks.

**17.** The *Causality* selection has 3 choices: no, `fmax`, or `auto`. `Auto` is recommended. Either use causality or smooth data windowing. Not both. Also, don't specify *High frequency extrapolation* to be linear when using causality. `No` doesn't add a causality check. `Fmax` will retain the data in the frequen cy range of the s-parameter file and then add a transfer function above the frequency range in the S-Parameter file to force the system ot be causal. `Auto` applies causality if it needs it.

**Recommendation:** Set *causality* to `auto` and if you have a passive network, set *passivity* to `enforce`.

**18.** Click *OK* in the *Edit Object Properties* form for the *nport* component.

The *Edit Object Properties* form closes.

**19.** In the Virtuoso Schematic Editing window, choose *Design – Check and Save*.

**20.** Run the simulation.

# Controlling Model Accuracy

The *nport* component has three parameters to control the accuracy of the rational interpolation process,

- *relerr* (The *Relative error* field)
- *abserr* (The *Absolute error* field)
- *ratorder* (The *Rational order* field)

You can use these parameters to trade off accuracy against model size and simulation time. In general, the more stringent the accuracy requirement, the higher the model order. Higher-order models require longer simulation time. Spectre RF can automatically generate a model that meets a specified accuracy requirement. You can also specify the model order directly to Spectre RF.

## Using the relerr and abserr Parameters

Let

$$S_{ij}\langle\omega\rangle$$

denote the *i,j* entry of the scattering parameter matrix at frequency ω and let

$$\hat{S}_{ij}\langle\omega\rangle$$

denote the corresponding rational interpolant.

The rational interpolation algorithm attempts to find an interpolant such that

$$\max_{\omega,\, i,\, j} \left|S_{ij}(\omega) - \hat{S}_{ij}(\omega)\right| < max(relerr \times \left|S_{ij}(\omega)\right|, abserr)$$

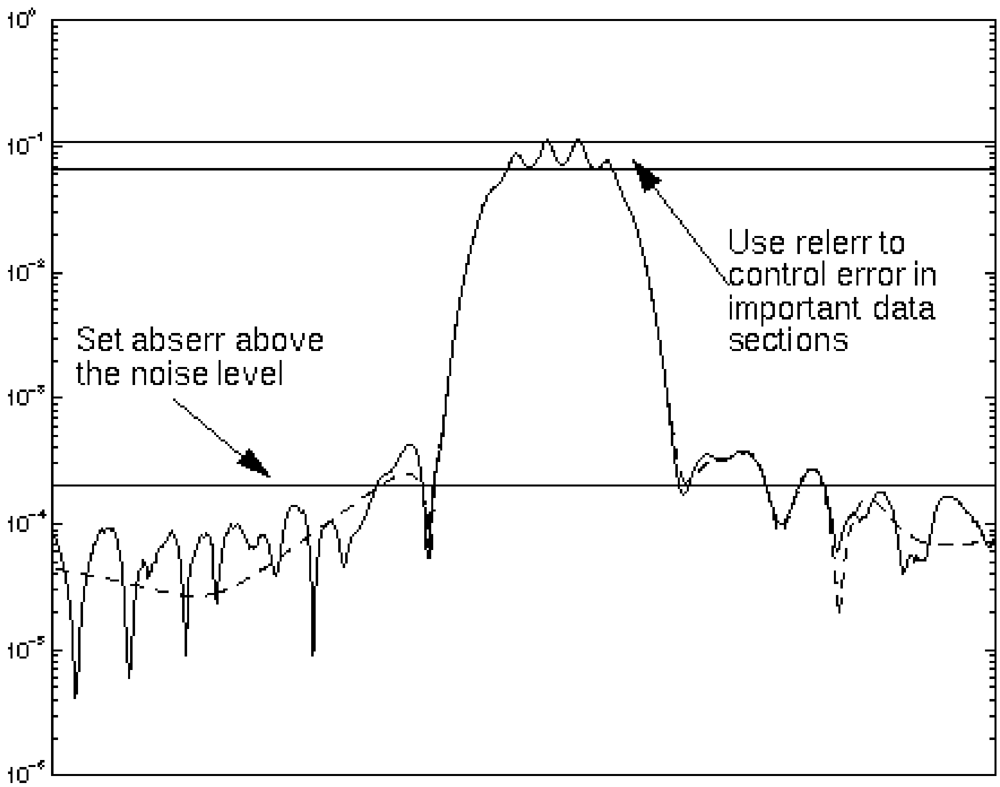
Generally, you use *relerr* for relative error control, and *abserr* for absolute error control.

Consider the data shown by the solid line in Figure 19-1. In this example

- *relerr* can control the error in the passband
- *abserr* can control the error in the stopband

If you are not interested in the details of the filter behavior in the stopband, you can set *abserr* to the level shown and these details are ignored in the interpolation.

**Figure 19-1 Using relerr and abserr**




This example has approximately a 10% ripple in the passband. To ensure adequate error control there, set *relerr* to much less than 0.1, perhaps to 0.01.

The dashed line in Figure 19-1 shows an interpolation function with the error control levels specified in Figure 19-1: *abserr*=2e$^{-4}$, and *relerr*=0.01. Details of the data below the *abserr* threshold are not resolved by the rational interpolation.

To recover the details in the stopband, set *abserr* to about 1e$^{-5}$ for this example.

If there is noise in the data, set *relerr* and *abserr* above the respective noise levels. Otherwise, Spectre RF attempts to interpolate the noise, resulting in very high order models and very long simulation times.

Remember that if you set *relerr* to zero, then, from the formula above, pure absolute error is used. Conversely, if you set *abserr* to zero, the error control is based solely on the errors relative to the magnitude of the input data.

### Using the ratorder Parameter

In general, it is usually best to specify only the accuracy parameters, *relerr* and *abserr*, and to let Spectre RF automatically select *ratorder,* the order of the rational approximation. However, if you have special information about your data set, you can direct Spectre RF to use a specific order of approximation in the rational interpolation. For example, if you know that your tabulated S-parameter data represents a sixth-order filter, you might instruct Spectre RF to use a seventh or eighth order fit. The slightly higher order gives Spectre RF flexibility to adjust for any noise or non-ideal behavior in the data.

When you specify the *ratorder* parameter, *relerr* and *abserr* are used to generate warnings when the order you selected is insufficient to meet accuracy requirements. Otherwise, the *relerr* and *abserr* parameters are not used.

## Troubleshooting

Certain S-parameter data sets might cause difficulty for the rational interpolation process. Types of data to avoid are

- Data specified only over a very narrow frequency range.

  Time-domain simulation requires time-domain models with a wider range of values; values that lie outside of this narrow frequency range. Data extrapolation might be difficult and is always risky.

- Very noisy data.

  Noisy data might lead to large, unreliable time-domain models.

- Data on a very sparse frequency grid.

  Accurate interpolation of such data might be impossible.

- Data with long ideal delays.
- Data representing idealized lossless elements, such as lossless transmission lines.

## Assessing the Quality of the Rational Interpolation

If you suspect a problem with the rational interpolation process, you can investigate it using the *sp* analysis with the following steps:

1. Construct a test schematic consisting of an *nport* component with interpolation set to rational, as discussed in “Using the nport Component” on page 1070.

2. Next add the appropriate number of *port* components to the schematic.
3. Perform an *sp* analysis on the *nport* component and look for anomalies.

   Refer to "Using S-Parameter Input Files" section of Chapter 8 in the *Virtuoso Spectre Circuit Simulator RF Analysis User Guide (Volume 1)* for an example of how to set up and run this type of simulation.

   Large swings in interpolated values and S-parameter magnitudes greater than one both suggest a problem.

   Large changes in the interpolant result from an inaccurate fit

   S-parameter values greater than one result from a non-passive (energy-generating) model that might create unstable time-domain solutions. Be particularly critical of anomalies near the zero frequency (DC).

   When the anomalous behavior occurs *within* the frequency range of data in the S-parameter data file, it usually indicates an inaccurate rational interpolation.
4. Verify that all the conditions listed at the beginning of the Troubleshooting section are met.
5. Try decreasing *relerr* or *abserr* or both.
6. Try specifying a higher-order interpolation with the *ratorder* parameter.

   Remember that measured data can contain fine details that might require a higher order than you might expect from a casual inspection of the data.

   If the anomalous behavior occurs *outside* the tabulated frequency range, try changing the *abserr*, *relerr*, or *ratorder* parameters. Sometimes anomalies can be removed by using a more accurate fit. However, there are limits to the ability to extrapolate outside the frequency interval you specify. You might need to specify additional data points to fix the problem.

# Model Reuse

The ROM data file (time-domain model) feature of the *nport* components lets you perform the conversion form a S-parameter data set to a time-domain model once and reuse the ROM data file in many designs.

For a given S-parameter data set, after you have specified a location for its ROM data file and Spectre RF has performed the conversion to a time-domain model and written the time-domain model to the ROM data file, you can reuse the model file in future simulations. To reuse the model for another *nport*, enter the time-domain model file name in the *ROM data* *file* field on the *nport* component's Edit Object Properties form in the new design. At this point, you no longer need to specify and convert a raw S-parameter data file.

When you specify both an S-parameter data file and a ROM data file,

- If the two files are consistent, Spectre RF reuses the time-domain model in the ROM data file
- If the two files are *not* consistent, Spectre RF generates a new time-domain model from the raw tabulated S-parameter data and overwrites the ROM data file with the new time-domain model data

This feature lets you specify both the ROM data file and the S-parameter data file in a design at the time when you place the *nport* component. Spectre RF then automatically generates the time-domain model during the first simulation and then reuses the time-domain model for all subsequent simulations without needing to change the *nport* component parameters.

If you require a more accurate rational interpolation, then you must regenerate the model in the ROM data file by changing the *relerr*, *abserr* or *ratorder* fields as described in "Controlling Model Accuracy" on page 1072.

# The S-Parameter File Format Translator (SPTR)

The S-parameter data file format translator (`sptr`) is a separate program from the Spectre RF simulator. You can find documentation for `sptr` in the *Virtuoso® Spectre User Guide*.

## References

To learn technical details about how Spectre RF converts S-parameter data to a time-domain description, see the article "Robust rational function approximation algorithm for model generation," by C. P. Coelho, J. R. Phillips, and L. M. Silveira. This article appeared in the proceedings of the 36th Design Automation Conference, New Orleans, LA, June 1999.

# 20

# Plotting Spectre S-Parameter Simulation Data

This chapter describes the equations used by the calculator to plot data generated by Spectre S-parameter simulations.

Using the calculator in the analog design environment, you can plot the following S-parameter data:

- Network Parameters
- Two-Port Scalar Quantities
- Two-Port Gain Quantities
- Two-Port Network Circles
- Equation for VSWR (Voltage Standing Wave Ratio)
- Equation for GD (group delay)

Use the buttons located in the *Function* area of the S-parameter Direct Plot form to specify the type of data to plot.

## Network Parameters

You can plot S, Y, Z, and H network parameters.

S-parameters, Y-parameters, and Z-parameters (denoted as SP, YP, and ZP on the Direct Plot form) are defined for circuits with any number of ports. H-parameters (denoted as HP on the Direct Plot form) are defined only for two-port circuits.

You can plot parameters on polar charts, Smith charts, or on rectangular plots after applying a *Modifier* option. The dB conversion uses $20 \log_{10} X$ because the parameters represent scalar ratios (for example, voltage).

## Equations for Network Parameters

For the ZP, YP, and HP parameters, Spectre returns S-parameters to the analog design environment. The environment converts them, as needed, to the equivalent Z, Y, and H matrixes using standard published methods. Spectre calculates S-parameter values.

SP (S-parameter) values

SP (S-parameter) values are calculated by Spectre.

ZP (Z-parameter) equation

The Z-parameter equation is as follows

$$Z_m = [Z_{ref}][I + S_m][I - S_m]^{-1}[Z_{ref}]$$

Where

- $S_m$ is the N-port S-parameter matrix
- $I$ is the N x N identity matrix
- $Z_{ref}$ is the characteristic impedance of the port
- $Z_m$ is the resulting Z-parameter matrix

#### YP (Y-parameter) equations

The Y-parameter equations are as follows

$$Y_m = [Y_{ref}][I - S_m][I + S_m]^{-1}[Y_{ref}]$$

Where

- $S_m$ is the N-port S-parameter matrix
- $I$ is the N x N identity matrix
- $Y_{ref}$ is a diagonal matrix defined as

$$Y_{ref} = \frac{1}{\sqrt{\Re e\{Z_i\}}}$$

Where

- $Z_i$ is the terminating impedance at port $i$
- $Y_m$ is the resulting Y-parameter matrix

### The HP (H-parameter) equations

The HP (H-parameter) equations only apply to two-port circuits.

$D$ is

$$D = (1 - S_{11})(1 + S_{22}) + S_{21}S_{12}$$

$H_{11}$ is

$$H_{11} = \frac{[(1 + S_{11})(1 + S_{22}) - S_{21}S_{12}]Z_{ref1}^{2}}{D}$$

$H_{21}$ is

$$H_{21} = \left(\frac{-2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

$H_{12}$ is

$$H_{12} = \left(\frac{2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

and $H_{22}$ is

$$H_{22} = \frac{(1 - S_{11})(1 - S_{22}) - S_{21}S_{12}Z_{ref}^{2}}{D}$$

Where

- $Z_{ref1}$ is the terminating impedance at port 1
- $Z_{ref2}$ is the terminating impedance at port 2

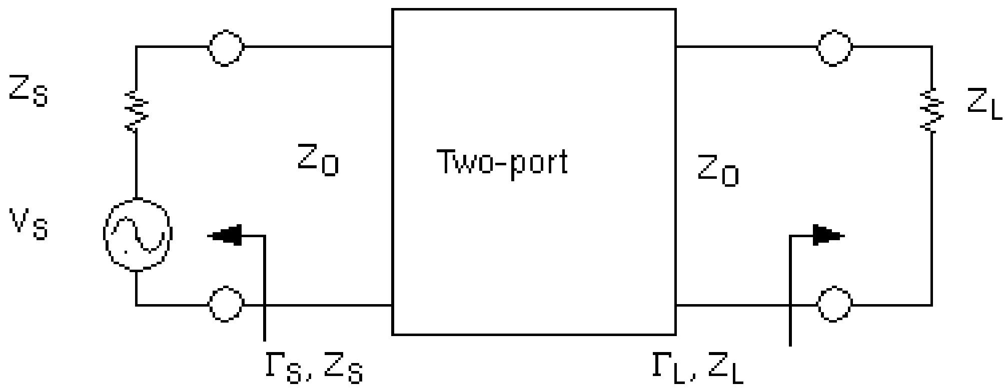
# Two-Port Scalar Quantities

The two-port scalar quantities include

- $NF_{min}$ is minimum noise figure
- $NF$ is noise figure
- $R_n$ is equivalent noise resistance
- $G_{min}$ is Optimum Noise reflection coefficient
- $B_{1f}$ is alternative stability factor
- $K_f$ is stability factor

You can plot two-port scalar quantities only against frequency. In addition, you can plot them only on rectangular charts. Figure 20-1 illustrates a generic two-port circuit that defines impedances and reflection coefficients.

**Figure 20-1 A Generic Two-Port Circuit**




In Figure 20-1

- $Z_s$ is the source impedance
- $\Gamma_s$ is the input reflection coefficient
- $Z_L$ is the load impedance
- $\Gamma_L$ is the load reflection coefficient
- $Z_O$ is the characteristic impedance

## Equations for Two-Port Scalar Quantities

### $G_{min}$ (Optimum Noise Reflection Coefficient) Equation

$G_{min}$ (also known as $\Gamma_{min}$ in the literature) is the reflection coefficient associated with minimum noise figure. You can plot $G_{min}$ on a Smith chart or, by using the `Modifier` field, on a rectangular chart.

$$G_{min} = \Gamma_{min} = \frac{Z_{on} - Z_o}{Z_{on} + Z_o}$$

Where

- $Z_o$ is the characteristic impedance
- $Z_{on}$ is the source impedance associated with minimum noise figure $(NF_{min})$

### $NF_{min}$ (Minimum Noise Figure) and NF (Noise Figure) Equations

The $NF_{min}$ and $NF$ plots are controlled by the *Modifier* option in the Direct Plot form.

Plot noise figure (NF) in dB by setting *Modifier* to *dB10*

- Plot noise factor (F) by setting *Magnitude*

Use $NF_{min}$ and $NF$ only for two-port circuits.

$$NF = 10\log_{10}F$$

Where

- $F$ is the noise factor
- $NF_{min}$ is the minimum noise figure

The $NF_{min}$ values are calculated by Spectre

The $NF$ (noise figure) equation is calculated by the analog design environment from $NF_{min}$, $G_{min}$ ($\Gamma_{min}$), and $r_n$. You can specify the optional source reflection coefficient $\Gamma_S$ as an argument if you use the analog design environment calculator. From the Direct Plot form, the analog design environment assumes $\Gamma_S$ to be 0 (input matched to reference termination).

$$NF = NF_{min} + \frac{4r_n\left|r_s - r_{min}\right|^2}{\left(1 - \left|r_s\right|^2\left|1 + r_{min}\right|^2\right)}$$

Where

$$r_n = \frac{R_n}{Z_o}$$

Here $r_n$ is the *normalized* equivalent noise resistance.

Rn (equivalent noise resistance)

$R_n$ plots equivalent noise resistance. The $R_n$ values are calculated by Spectre.

Stability Factors

$K_f$ and $B_{1f}$ plot the Rollet stability factor and its intermediate term. Use these parameters only for two-port circuits.

$D$ is

$$D = S_{11}S_{22} - S_{21}S_{12}$$

$K_f$ is

$$K_f = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |D|^2}{2|S_{21}||S_{12}|}$$

and $B_{1f}$ is

$$B_{1f} = 1 + |S11|^2 - |S22|^2 - |D|^2$$

# Two-Port Gain Quantities

The following gain quantities are valid only for two-port circuits

$G_A$ (available gain) is the power gain obtained by optimally matching the output of the network.

$G_P$ (power gain) is the power gain obtained by optimally matching the input of the network.

$G_T$ (transducer gain) shows the insertion effect of a two-port circuit. This quantity is used in amplifier design.

$G_{umx}$ (maximum unilateral transducer power gain)

$G_{max}$ (maximum available gain) shows the transducer power gain when there exists a simultaneous conjugate match at both ports.

$G_{msg}$ (maximum stable gain) shows the gain that can be achieved by resistively loading the two-port such that $k = 1$ and then simultaneously conjugately matching the input and output ports. For conditionally stable two-ports, you can approach the maximum stable gain as you reduce the input and output mismatch. If you attempt a simultaneous conjugate match and $k < 1$, the two-port oscillates.

## Equations for Two-Port Gain Calculations

### $G_A$ (Available Gain) Equations

Available gain equations for output conjugately matched.

$$G_A = \frac{|S_{21}|^2\left(1 - |\Gamma_S|^2\right)}{|1 - S_{11}\Gamma_S|^2\left(1 - |\Gamma_2|^2\right)}$$

Where

$$\Gamma_2 = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1 - S_{11}\Gamma_S}$$

**Note:** When you use the S-parameter Direct Plot form, $\Gamma_S$ is set to zero, and therefore available gain ($G_A$) is plotted as

$$G_A = \frac{|S_{21}|^2}{1 - |S_{22}|^2}$$

To plot $G_A$ for nonzero values of $\Gamma_S$, use the analog design environment calculator.

### $G_P$ (Power Gain) Equations

Power gain equations for input conjugately matched.

$$G_P = \frac{|S_{21}|^2(1 - |\Gamma_L|^2)}{|1 - S_{22}\Gamma_L|^2(1 - |\Gamma_1|^2)}$$

Where

$$\Gamma_1 = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1 - S_{22}\Gamma_L}$$

**Note:** When you use the S-parameter Direct Plot form, $\Gamma_L$ is set to zero, and therefore the power gain, $G_P$, is plotted as

$$G_P = \frac{|S_{21}|^2}{1 - |S_{11}|^2}$$

To plot $G_P$ for nonzero values of $\Gamma_L$, use the analog design environment calculator.

### $G_T$ (Transducer Gain) Equations

$$G_T = \frac{(1 - |\Gamma_S|^2)|S_{21}|^2(1 - |\Gamma_L|^2)}{|(1 - S_{11}\Gamma_S)(1 - S_{22}\Gamma_L) - S_{12}S_{21}\Gamma_S\Gamma_L|^2}$$

**Note:** When using the S-parameter Direct Plot form, the analog design environment assumes that the source ($\Gamma_S$) and load ($\Gamma_L$) reflection coefficients are zero. $G_T$, therefore, plots the insertion gain.

$$G_T = |S_{21}|^2$$

Using the calculator, you can plot $G_T$ and specify the source and load terminations.

$G_{max}$ (Maximum Available Gain) Equations

For K >1

$$G_{max} = \left|\frac{S_{21}}{S_{12}}\right|\left[K - \sqrt{K^2 - 1}\right]$$

For K<=1

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}$$

Where K is the stability factor, $K_f$

**$G_{umx}$ (Maximum Unilateral Transducer Power Gain) Equation**

$$G_{umx} = \frac{|S_{21}|^2}{(1 - |S_{11}|^2)(1 - |S_{22}|^2)}$$

**$G_{msg}$ (Maximum Stable Power Gain) Equation**

$$G_{msg} = \frac{|S_{21}|}{|S_{12}|}$$

# Two-Port Network Circles

NC plots constant noise contours at the input of a two-port circuit. GAC plots constant gain contours at the input port, and GPC plots constant gain contours at the output port. Gain contour values reflect an optimum match at the opposing port.

Noise and Gain circles can be plotted at a single dB value for a range of frequencies or at a single frequency for a range of dB values. If you do not enter values for the frequency range, a circle is plotted for every simulated frequency for which a circle with the specified value exists.

SSB plots stability circles at the input port, and LSB plots stability circles at the output port. You can also specify a limited frequency range for these contours.

## Equations for Two-Port Network Circle

### NC (Noise Circle) Equations

$$N_i = \frac{(F_i - F_{min})\left|1 + \Gamma_{min}\right|^2}{4r_n}$$

where

$$\Gamma_{min} = G_{min}$$

The center is calculated using

$$C_N = \frac{\Gamma_{min}}{1 + N_i}$$

The radius is calculated using

$$r_N = \sqrt{\frac{N_i^2 + N_i\left(1 - \left|\Gamma_{min}\right|^2\right)}{1 + N_i}}$$

Where i is the index number

### GAC (Available Gain Circle) Equations

The center is calculated using

$$C_A = \frac{g_a(S_{11}{}^* - D^* S_{22})}{1 + g_a\left(\left|S_{11}\right|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_A = \frac{\sqrt{1 - 2K_f\left|S_{21}S_{12}\right|g_a + \left|S_{12}S_{21}\right|^2 g_a^2}}{\left|1 + g_a\left(\left|S_{11}\right|^2 - |D|\right)^2\right|}$$

Where

$$G_a = \frac{G_A}{\left|S_{21}\right|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

### GPC (Power Gain Circle) Equations

The center is calculated

$$C_P = \frac{g_p(S_{22}{}^* - D^* S_{11})}{1 + g_p\left(\left|S_{22}\right|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_P = \frac{\sqrt{1 - 2K_f\left|S_{21}S_{12}\right|g_p + \left|S_{12}S_{21}\right|^2 g_p^2}}{1 + g_p\left(\left|S_{22}\right|^2 - |D|^2\right)}$$

Where

$$g_p = \frac{G_P}{\left|S_{21}\right|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

### LSB (Load Stability Circle) Equations

The center is calculated using

$$C_L = \frac{S_{11}D^* - S_{22}{}^*}{|D|^2 - \left|S_{22}\right|^2}$$

The radius is calculated using

$$r_L = \left|\frac{S_{12}S_{21}}{|D|^2 - \left|S_{22}\right|^2}\right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

### SSB (Source Stability Circle) Equations

The center is calculated using

$$C_S = \frac{S_{22}D^* - S_{11}{}^*}{|D|^2 - |S_{11}|^2}$$

The radius is calculated using

$$R_S = \left|\frac{S_{12}S_{21}}{|D|^2 - |S_{11}|^2}\right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

## Equation for VSWR (Voltage Standing Wave Ratio)

VSWR is calculated from the S-parameters. You can plot the VSWR at any port in the circuit on a rectangular chart.

$$VSWR_i = \frac{1 + |S_{ii}|}{1 - |S_{ii}|}$$

Where $i$ is the port number.

Equation for ZM (Input Impedance)

You can plot input impedance if all other ports are matched

$$Z_m = \frac{1 + S_{ii}}{1 - S_{ii}} R$$

Where R the reference impedance of the port of interest and i is the port number

## Equation for GD (group delay)

GD (group delay) approximates the derivative of the phase with respect to frequency, normalized to 360 degrees. Units for group delay are in seconds.

$$G_d \cong \frac{-d\phi}{d\omega}$$

Group delay is calculated from the phase of the corresponding S-parameter (for example, $GD_{21}$ corresponds to $S_{21}$).

# 21

# Documents That Ship in the Software Hierarchy

In addition to the Spectre RF documents visible through CDNSHelp, there are many documents that ship in the software hierarchy along with workshops. To find these documents, go to

`$MMSIMHOME/tools/spectre/examples/SpectreRF_workshop`

The structure of the `SpectreRF_workshop` directory is given in the following table. To make the best use of this information, look at the summaries of contents to determine which document contains the information you need. That document will tell you which tar ball contains the corresponding database. The tar balls are also included in the `SpectreRF_workshop` directory, but are not listed here.

| Document or directory | For a brief summary of contents, see |
|---|---|
| `SpectreRF_simulink_example.pdf` | SpectreRF_simulink_example.pdf on page 1096 |
| `SpectreRF_AN` directory | |
| `EnvelopeAN.pdf` | EnvelopeAN.pdf on page 1096 |
| `HB_AN.pdf` | HB_AN.pdf on page 1099 |
| `JitterAN.pdf` | JitterAN.pdf on page 1101 |
| `LSSP_AN.pdf` | LSSP_AN.pdf on page 1097 |
| `LTJM_AN.pdf` | LTJM_AN.pdf on page 1099 |
| `MatlabAN.pdf` | MatlabAN.pdf on page 1101 |
| `MatlabWorkshop.pdf` | MatlabWorkshop.pdf on page 1097 |
| `NS_AN.pdf` | NS_AN.pdf on page 1098 |
| `PerturbationAN.pdf` | PerturbationAN.pdf on page 1100 |

| Document or directory | For a brief summary of contents, see |
|---|---|
| `PLL_Jitter_AN.pdf` | PLL_Jitter_AN.pdf on page 1098 |
| `PSRR_Drv_AN.pdf` | PSRR_Drv_AN.pdf on page 1098 |
| `PSRR_Osc_AN.pdf` | PSRR_Osc_AN.pdf on page 1100 |
| `PstbAN.pdf` | PstbAN.pdf on page 1102 |
| `readme.txt` | readme.txt on page 1099 |
| `RF_Blocks_AN.pdf` | RF_Blocks_AN.pdf on page 1100 |

# SpectreRF_simulink_example.pdf

Title: *Co-Simulation with SpectreRF MATLAB/Simulink Tutorial*

This document is a tutorial that describes how to run a cosimulation between Spectre RF and MATLAB/Simulink.

Tutorial 1 demonstrates Simulink, Spectre RF cosimulation while the Spectre RF simulator runs a tran analysis, for the system-level module of a wireless LAN. The three steps required for the cosimulation include:

1. Inserting and configuring the Spectre RF Engine in the Simulink schematic.
2. Setting up the netlist to adopt the Spectre RF and MATLAB cosimulation.
3. Running the cosimulation.

Tutorial 2 demonstrates Simulink, Spectre RF cosimulation while the Spectre RF simulator runs an envlp analysis. This tutorial illustrates:

1. The use mode of manually running Spectre with ADE and calling MATLAB from ADE.
2. The use mode of manually running Simulink and activating Spectre through Simulink.

# EnvelopeAN.pdf

Title: *A User's Guide to Envelope Following Analysis Using SpectreRF*

This document is an application note that describes how to use envelope following analysis with Spectre RF. The topics covered include:

1. Envelope following algorithm

2. Using the envelope following analysis
3. Parameters used in the analysis
4. Plotting results
5. An automatic gain control (AGC) example
6. Comparing simulation and measurement results for a GSM power amplifier
7. Using autonomous envelope following, with examples
8. Simulating circuits with FM sources

# LSSP_AN.pdf

Title: *Measuring Large-Signal S-Parameters Using SpectreRF*

This document illustrates how to measure large signal S-parameters using Spectre and Spectre RF from within ADE. The contents include:

1. Calculating LSSP in terms of input power
2. Calculating LSSP in terms of beat frequency
3. Comparing large and small-signal S-parameters

# MatlabWorkshop.pdf

Title: *SpectreRF Workshop: Using the SpectreRF MATLAB Toolbox*

This document is a workshop that describes how to use Spectre RF and the Spectre RF MATLAB toolbox to measure parameters that are important in the design verification of low noise amplifiers (LNA). The contents include:

1. Measuring the small signal gain (SP) of an LNA
2. Conducting a large signal noise analysis (PSS and Pnoise)
3. Measuring gain compression and total harmonic distortion (Swept PSS)
4. Measuring IP3 (PSS plus PAC)
5. Measuring IP3 (QPSS analysis with shooting or the harmonic balance engine)
6. Measuring IP3 (Rapid IP3 using AC analysis)

7. Measuring IP3 (Rapid IP3 using PSS Plus PAC)

# PLL_Jitter_AN.pdf

Title: *Application Note. Pll jitter measurements*

This document illustrates how to use the Spectre and Spectre RF simulators with ADE to measure jitter characteristics of phase-locked loop (PLL) circuits. The contents include:

1. Using Spectre RF to simulate the individual blocks and measure the jitter and operating parameters of the models
2. Creating or modifying the behavioral models of the blocks to incorporate the jitter
3. Time domain simulation of the PLL using the behavioral models of the blocks.
4. Post processing of the simulation results to obtain the jitter and noise characteristics of the entire PLL.

# NS_AN.pdf

Title: *The Application Note of Noise Separation in Pnoise/Qpnoise Analysis*

This document is an application note that describes how to use noise separation features in Pnoise and Qpnoise analysis to extract noise sources. The contents include:

1. The principles of noise separation in RF circuits
2. The graphical user interface used for noise separation
3. The flow to follow to determine which noise sources add the most noise to the output
4. Understanding the noise source summary report
5. An example is provided that illustrates how to locate the noise sources that contribute the most noise to the output in an NE600 mixer.

# PSRR_Drv_AN.pdf

Title: *Power Supply Rejection Ratio Characterization Using SpectreRF Driven Circuits*

This document illustrates how to characterize the power supply rejection ratio (PSRR) of driven circuits. The contents include:

1. How to set up the PXF analysis
2. Using Direct Plot for the PXF analysis to plot the periodic transfer function

## readme.txt

This text document lists the main topic of each of the `.pdf` files in the `SpectreRF_workshop`.

## HB_AN.pdf

Title: *Harmonic Balance Release Note*

This document illustrates how to use the harmonic balance (HB) method, which is very efficient in simulating circuits with only low order harmonics, such as low-noise amplifiers. The contents include:

1. The parameters used to specify HB for driven circuit large signal simulations
2. How to use HB from within ADE
3. Discussion of when it is most appropriate to use time domain shooting and when it is better to use HB

## LTJM_AN.pdf

Title: *Application Note. Long term jitter measurements*

This document describes the procedures used to measure long term jitter characteristics from within the Analog Design Environment (ADE). Long term jitter characterizes the variation in the accumulated width of a large number of clock periods. The contents include:

1. Plotting the phase noise power spectral density, which will be used to compute the long term jitter
2. Selecting the number of cycles to use for the measurement
3. Selecting the minimum frequency of the integration. Long term jitter is described as very sensitive to the lower frequency limit that is used in the integration of the phase noise.

# PerturbationAN.pdf

Title: *Perturbation Based Measurements Using SpectreRF*

This document describes how to use perturbation based rapid IP3, rapid IP2, the IM2 distortion summary, and the compression distortion summary to measure the compression distortion and intercept points of RF circuits. Contents include:

1. A discussion of intermodulation distortion
2. A discussion of compression distortion
3. A discussion of the perturbation method used in the analyses
4. An example using IP3 and compression distortion summary measurements of a mixer
5. An example using IP2 and IM2 distortion summary measurements of a Gilbert direct conversion mixer
6. An example using IPa3 and compression distortion summary measurements of a power amplifier

# PSRR_Osc_AN.pdf

Title: *Power Supply Rejection Ratio Characterization Using SpectreRF Autonomous Circuits*

This document describes how to measure the power supply rejection ratio (PSRR) of oscillators and other autonomous circuits. Knowing PSRR values can help you protect against noise sources such as power supply ripples. The same methodology can be used to measure the effect of substrate noise. Contents include:

1. Setting up the PXF analysis to measure the PSRR of a VCO.
2. A discussion of the different definitions of PSRR
3. A discussion of confirming the results of the PXF analysis by taking quasi-static measurements of the tuning sensitivity and frequency pushing of the VCO

# RF_Blocks_AN.pdf

Title: *Using Verilog-A Baseband Models of RF Blocks Application Note*

This document describes how to create and use Verilog-A signal flow and conservative baseband models of RF blocks. Contents include:

1. Several examples exploring the correspondence between passband (PB) and baseband (BB) representations of RF signals
2. Examples illustrating baseband modeling of linear time invariant blocks
3. Examples illustrating baseband modeling of nonlinear blocks using an amplifier
4. Code examples for Verilog-A modules, including
   - An ideal baseband to passband converter
   - A behavioral model of a passband to baseband converter
   - A behavioral model of a deviation frequency meter
   - A baseband model of a passband low pass filter with transfer function `H(s) = a/(a+s)`
   - A signal flow baseband model of a nonlinear RF block based on the complex BB gain approach

# JitterAN.pdf

Title: *Jitter Measurements Using SpectreRF Application Note*

This document illustrates how to measure the jitter characteristics of typical blocks, working within ADE. Examples are presented for both driven and autonomous circuits. Contents include:

1. A discussion of how time (or phase) jitter is defined. RMS jitter and peak-to-peak jitter measurements are introduced
2. An example illustrating how to measure jitter for an autonomous circuit (differential VCO)
3. An example illustrating how to measure phase noise and jitter at the output of a divider, in a circuit that uses a VCO with divide-by-2 prescaler

# MatlabAN.pdf

Title: *Spectre/RF Matlab Toolbox Application Note*

This document describes how to use Spectre RF MATLAB toolbox to read in a simulation result and perform standard RF measurements. Contents include:

1. Instructions for obtaining and setting up the toolbox
2. A list and description of the basic functions included in the toolbox:
   - `cds_srr`
   - `cds_evalsig`
   - `cds_plotsig`
   - `cds_harmonic`
   - `cds_interpsig`
   - `cds_fft`
   - `cds_compression`
   - `cds_ipn`
3. An example illustrating how to obtain the 1dB compression point, the first order harmonic signal, and the total harmonic distortion
4. A discussion of compatibility with the Aptivia MATLAB functions

## PstbAN.pdf

Title: *Stability analysis of Linear Periodical Time-Varying Circuit using SpectreRF PSTB Analysis Application Note*

This document discusses how to use STB analysis to evaluate the stability of a linear periodical time-varying circuit. Contents include:

1. An example illustrating how to run an STB and PSTB analysis on a VCO with an inherent nonlinearity
2. An example illustrating how to examine the local stability of an injection-locked oscillator
3. An example illustrating how to examine the global stability of an injection-locked oscillator

# Index

## Numerics



## A



## B



## C

# D



# E

## F



## G



## H

## I



## J



## L



## M

# P

## T



## U



## V



## W