York University
Department of Electrical Engineering and Computer Science
EECS 4215

Lab #2 Wireless Channel Models and Radio Performance

1 Objectives

Our aim is to apply what we have learned about statistical wireless channel modelling and their impact on a basic communicator.

2 Discussion

Large-scale and small-scale wireless channel models give radio designers an idea of the average power that they can expect in a communication environment as well as the variation in received power that they may incur. Typically, these models are statistical in nature.

The data from which these models are constructed can be arrived in any of a number of ways. Of course all approaches employ a signal transmission source and a receiver that collects the wireless signals at different locations in the environment. Typically, the person making these measurement walks or drives around their environment of interest collecting signals.

A common procedure involves transmitting a known spread-spectrum sequence centred on some RF carrier frequency. The sequence repeats every time period, T. The receiver picks up this signal in whatever way that the wireless channel facilitates it, downconverts it and correlates whatever it received with the knowns spreading sequence every, T. In this way the receiver is able to effectively sample the response of the channel to an impulse of bandwidth on the order of $1/T$.

To gather these measurements all you really need is an RF receiver capable of downconverting the RF signal to a low enough frequency for an oscilloscope (or some custom baseband electronics you designed or purchased) to capture the spread-spectrum data and forward it to a computer for signal-processing. Another approach is to use a vector network analyzer (VNA), a very handy (and usually very expensive) machine capable of capturing the amplitude and phase response of a system. Obviously, if you are using a big set-up consisting of oscilloscopes and VNAs you have to place this equipment on a cart or moving vehicle to carry out the measurements.

When collecting large-scale data, the receiver is brought to a particular point, its location noted, and a number of measurements recorded of the signal power at that spot. These can later be converted to anything that you want (say the effective loss). A basic large-signal model of the loss in dB between a transmitter and a receiver separated by a distance, d is

$$L(d)|_{dB} = L(d_0) + 10n\log(d) + X_{sh} \text{ [dB]} \tag{1}$$

where d_0 is some reference separation, d is the actual separation between the transmitter and the receiver and n is the path loss coefficient, a measure of effective power density reduction as the signal propagates between the two radios. Of course there is a statistical component to this metric. Over the circular locus of points defined by the radial distance, d, we may encounter a variation in the received power due to the variation in objects capable of blocking the signal between transmitter and receiver.

This effect is accounted for by the shadowing term X_{sh} in (1). X_{sh} is a statistical term that models the loss in dB (not the loss in power directly) as a *normal* (i.e. Gaussian, or "bell curve") distribution. If you are playing around with this idea using MATLAB you can specify X_{sh} with a command like

```
Xsh = 5*randn(1);
```

which asks the language to create a random value from a Gaussian distribution with a standard deviation of 5. If you are thinking of this as your shadowing term you would say that it creates a random sample from a shadowing characteristics with 5-dB standard deviation.

For small-scale fading things get more interesting. As you know, in a realistic terrestrial wireless channel a radio signal arrives at the receiver through more than one path. That is, copies of a signal arrive with different amplitude and phase creating all sorts of opportunities for constructive and destructive interference. What's more these constructive/destructive scenarios can vary substantial with small changes in the receiver's location (hence the "small-scale" monicker) and thus induce large fluctuations (fading) in the received signal strength on top of the average established by the large-scale mechanism.

A statistical analysis of this phenomena has resulted in a couple (among many) popular characterization of small-scale fading. An especially popular probability density function (PDF) for characterizing the received signal power in a small-scale fading environment is the Rayleigh PDF which is

$$f_{ray}(a) = \frac{a}{\sigma^2}\exp\left(\frac{-a^2}{2\sigma^2}\right). \tag{2}$$

To be clear this expression outlines the distribution in the *amplitude* of a sinusoidal signal. If you want to generate Rayleigh statistics that conform to such a distribution in MATLAB you can try the command

```
raystats = sigma*(randn(1) + i*randn(1));
```

Note that the expression above retains the complex nature of this variable which effectively keeps track of the randomly varying amplitude and phase characteristics of this random variable. More exactly it represents the variable in terms of orthogonal (sin and cos carriers). Often in communication systems you need to keep track of both of these since you are sending signals on orthogonal carries (e.g. QPSK) and you want to know how each of these is being affected by the noise. If you just want the envelope of course then all you need to go (in MATLAB) is `abs(raystats}`.

Another common radio signal amplitude envelope distribution to consider it the Ricean PDF given by

$$f_{rice}(a) = \frac{a}{\sigma^2}\mathrm{I}_0\left(\frac{am}{\sigma^2}\right)\exp\left[\frac{-(a^2+m^2)}{2\sigma^2}\right] \tag{3}$$

where $\mathrm{I}_0(x)$ is the zeroth-order modified Bessel function, which is handily taken care of my MATLAB with the function `besseli(0,x)`. Similar to the the Rayleigh, the Ricean random variable can be created in MATLAB using

```
ricestats = m + sigma*(randn(1) + i*randn(1));
```

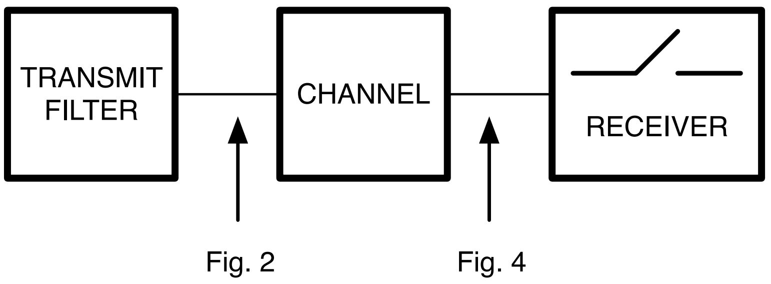

Figure 1: A basic baseband communications model.

This underscores the nature of this statistical entity, it is just the Rayleigh distribution with some average offset m. In terms of wireless communications this means that, among all the random signal arrivals (i.e. rays), there is a prevalent line-of-sight (LOS) link between a transmitter and receiver.

The questions below ask you to ponder some of the ideas expressed above in a little more quantitative detail. They also ask you to consider their impact on a very basic communication system. A block diagram of a very basic communication system is pictured in Fig. 1. This is a baseband equivalent of a passband (i.e. carrier modulated) radio. Any high frequency modulation will be handled implicitly in the simulation, but not explicitly. As you can probably imagine this is often necessary as the signals in a typical bandpass modulated communicator operate on vastly different time scales, the carrier at time steps on the order of a nanosecond and the signal on the order of milliseconds for established radio technologies. Asking any simulation to simultaneously handle signals separated by six-orders of magnitude in their time scale is quite challenging.

It consists of the basic transmitter, channel, receiver components making up all communications systems. The transmitter begins with a digital data source, random 1's and 0's entering our communicator. The 1's and 0's then go through a transmit filter that turns these into waveforms suitable for transmission across an analog channel. The simplest transmit filter just turns the discrete 1's and 0's into rectangular waveforms allowing some period of time T_b for each level.

Fig. 2 is a simple example of what the output of a transmit filter may look like in a simulation. In this case the 10-bit data sequence [1001010001] is being sent. Note that each bit is represented as 10 samples (that is "ten" samples, hence we have 100 samples total in the figure) and 1's are mapped into $+1$ and 0's are mapped to -1. These new mappings are called *symbols*. In digital communications all sorts of *symbols* can be sent, some better than others. Note that I have not actually defined what T_b is in generating the data in Fig. 2, the absolute time expanse of the 10 samples constituting a symbol has not yet been pinned down in any way, everything is relative. By taking 10 samples per symbol I have made the bandwidth of my simulation $10\times$ greater than the highest fundamental harmonic of my input data. Of course there are higher-order harmonics present in randomly switching square wave signals and increasing my sampling rate will capture these entities more accurately, but usually a $10\times$ oversampling is ok for non-critical work. So, if our T_b were 1 ms (i.e. I'm operating at a bit rate of 1-kbps) then I'm sampling harmonic content up to 5 kHz if I'm taking 10 samples per T_b.

Once you have the data ready for transmission you are ready to put it in the channel. This is where we can apply any of our wireless channel models. For illustration purposes imagine that the channel takes on a characteristic that can be described with a butterworth low-pass filter. A FIR filter equivalent can be constructed with a command such as

```
[Bchan,Achan] = butter( 6,1/10, 'low' );
```

In the command above the 6 implies that I want a sixth order butterworth low pass filter and that I want its bandwdith (3-dB bandwidth) to be at the symbol rate (i.e. at $1/T_b$, even though we don't have an actual T_b

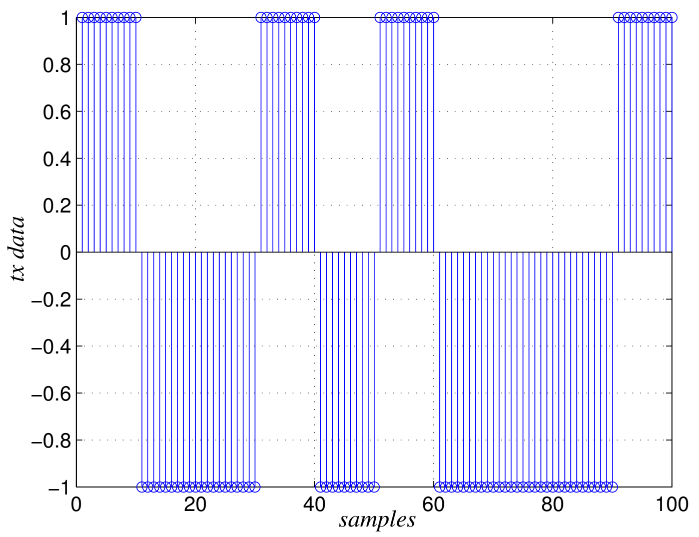

Figure 2: Data out of the transmit filter.

defined, everything is just treated in a relative manner), which is why I set the second term to 1/10, that is one-tenth the simulation bandwidth. If we had used 100 samples per symbol, then I would set this value to 1/100 if I want a BW at the symbol rate. The higher I make this value the more of my signal that I pass.

In MATLAB, I can pass the signal out of my transmit filter, say we call it `txdata` using the command

```
[chandata,filterstatenew] = filter(Bchan,Achan,txdata,filterstateold);
```

Notice how we define the filter setting using the `Bchan` and `Achan` parameters calculated with the previous command and then used these along with the channel input, `txdata`, to calculate `chandata`, what the data looks like coming out of the channel. The parameters `filterstateold` and `filterstatenew` hold and record the state of the filter from previous inputs, in case you want the filter to start of in some known non-zero state. Take a look at Fig. 3 for an example of `chandata`, it shows what the bit sequence [1001100000] looks like coming out of a couple different channels.

Next we should add noise to the data to model the noise it would be joined by at the receiver's antenna. As always the key metric is not how much noise is added, but how big the noise is *relative* to the information signal, that is the signal to noise ratio (SNR). In high-level simulations we typically define noise magnitude based on the SNR levels we wish to study. That is, we assume that we have achieved a power amplifier and encountered channel attenuation of such a nature that at the antenna, the signal to noise ratio is some value in dB, SNR_{dB}. We first change this to an expression in absolute terms

$$SNR = 10^{SNR_{dB}/10}. \tag{4}$$

A detailed discussion can show that for a signal varying between $+A$ and $-A$ on top of a carrier the average power is $A^2/2$. The energy in a symbol of duration T_b is then $A^2T_b/2$ and the SNR can be expressed as

$$SNR = \frac{A^2T_b}{2E_n} \tag{5}$$

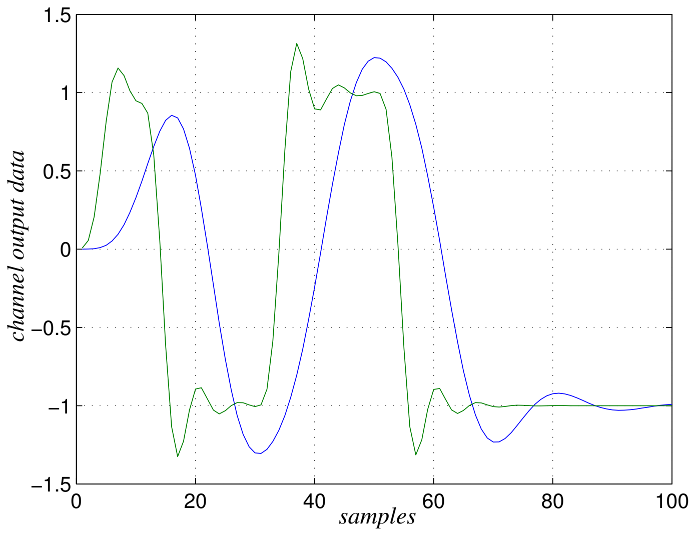

Figure 3: Low pass filtered channel data, one through a channel filter bandwidth at the symbol rate, the other with a channel filter bandwidth at three times the symbol rate. Guess which is which.

Where E_n is the noise energy over the symbol period. Thus,

$$E_n = \frac{A^2 T_b}{2 \cdot SNR}. \tag{6}$$

And converting this back to a voltage or current equivalent we have the average noise amplitude

$$A_n = \sqrt{\frac{A^2 T_b}{2 \cdot SNR}}. \tag{7}$$

The last thing is to convert this back into our simulation structure where the symbol duration is effectively defined in terms of the number of samples per symbol, N_s (10 in the examples above) thus we can express the noise samples like so in MATLAB

```
An = sqrt(A^2 * Ns/(2*SNR)*(randn(1) + i*randn(1));
```

Obviously if we were to apply this to the system considered above `A` would be set to 1. Hopefully the fact that the noise is expressed in terms of complex numbers does not cause you too much worry. When signals data is being sent on quadrature (i.e. sin and cos) carriers, the parts of the information on either carrier is effected by different noise disturbances which are readily modelled by expressing `noisesample` as a complex value. If you are not using quadrature modulation you can just use `real(noisesample` as your disturbance.

With the channel filtered signal and noise disturbance in place we can express the received signal as

```
rxdata = chandata + An;
```

an example of the transmit filter and received data (through a 5th order butterworth filter with a bandwidth of three times the symbol rate) at a 10-dB SNR is shown in Fig. 4.

Now it is the job of the receiver to recover the input as best it can. In the simple receiver of Fig. 1 we imply that the data is sampled at some opportune time, perhaps the centre of the anticipated symbol interval. A

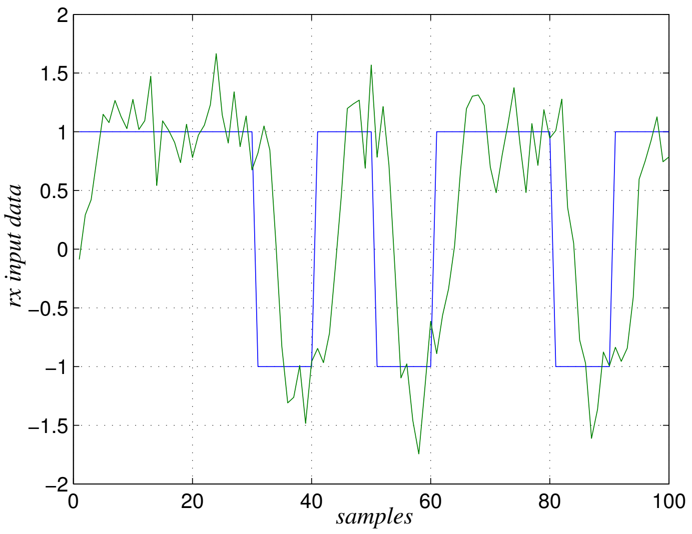

Figure 4: Transmitted and received data at 10-dB SNR.

smarter thing to do however is to run the input through a component that integrates the input over a symbol period, T_b. For square pulses, if the input is a $+A$ the output of the integrator is $+A \cdot T_b$ if the input is $-A$ the output of the integrator is $-A \cdot T_b$. This is good, because integrating the noise component over the same period raises its rms amplitude by $A_n\sqrt{T_b}$, a consequence of its random nature. A more sophisticated justification for this component can be outlined and it is typically referred to as the *matched filter*, but for our purposes this description satisfies. In any case, when we sample after T_b we have a better chance of correctly recognizing the symbol that arrived.

In MATLAB the effect of such the integrator can be easily realized with one line

```
intdata = filter(Bint,Aint,rxdata);
```

where `Aint` = 1 and where

```
Bint = ones(1,Ns)/Ns;
```

That is, just use the MATLAB `filter` function to sum up N_s, $1/N_s$ sized chunks of the input over a N_s samples (just like you'd expect an integrator to do).

Then, every N_s, a threshold detector (a *slicer*) can check whether `intdata` is greater than zero (in which case we assume that 1 arrived) or less than 0 (in which case we assume that a 0 arrived).

To gauge the quality of your communication system under various channels just check whether your receiver made the right decision and increment an error variable whenever it did not. At the end of the simulation divide the number of errors by the total number of bits processes to approximate the bit-error-rate (BER).

To get a good sense of the performance you have to run the simulation long enough to make enough errors such that you get a statistically useful result. To give you a sense of how many data points to run you can refer to the theoretical BER, P_e (i.e. the probability of error) for a particular modulation scheme under the same SNR as you are considering in your simulation (but with an otherwise ideal channel, not the case in your simulation). For example for simple BPSK the probability of error is

$$P_e|_{BPSK} = Q\left(\sqrt{2 \cdot SNR}\right) \tag{8}$$

where $Q(\cdot)$ is the complementary cumulative distribution function. In MATLAB this is also pretty easy to implement wherein the output of $Q(x)$ can be realized with

```
out=0.5*erfc(x/sqrt(2));
```

You should try for at least 100 errors in which case you should aim to simulate $100/P_e|_{BPSK}$ data points through your communicator.

One last thing, when trying out some random channel whose filter equivalent consists of more than one tap (e.g. a `Bchan` consisting of more than one term) you may not just look at the output of your integrator every kN_s samples (where k is an integer). Instead there may be an optimal delay, k_d for you to wait such that you don't sample until after $N_s + k_d$ samples on your first symbol and then look at the integrator output every N_s samples thereafter. In such cases, when seeking the BER of your receiver, first find the optimal delay, k_d. You find this in a straightforward manner, just try different k_d for some SNR until you find a minimum BER. Then stick with that k_d for the rest of your simulations.

3 Questions

1) Engineers have characterized the large-scale fading characteristics of an outdoor wireless channel by recording the path loss at a variety of points. At each point they made 30 measurements. Their results are summarized in `large_scale.dat`. Using these results provide the best model of the channel that you can.

2) Compute and show the histogram for the electric field amplitude envelope of a signal received after going through a channel exhibiting small-scale Rayleigh fading. Assume that the normalized variance of the received signal is 3. The histogram should summarize 100,000 random points and consist of 50 bars. For easy comparison, overlap your histogram plot with a curve of the closed form calculation of the Rayleigh probability density function (PDF) for the same parameters.

3) Repeat the computation for the problem above, but this time for a Ricean distribution with the same variance, but with an average of 2.

4) Engineers have also characterized the small-scale fading characteristics of the channel. They do this by moving their channel measurement equipment over very small distances around some central point. The channel loss data are summarized in the file `small_scale.dat`. Estimate as best you can the small-scale fading statistical parameters of this channel.

5) The delay-power profile of a GSM channel is included in `delay_power.dat`. What is the rms delay spread and coherence bandwidth of this channel?

6) Plot the BER curve as a function of SNR for a BPSK communicator operating over a completely ideal channel with additive noise present. Use 20 samples per period and try it for SNRs between -2 dB and 30 dB in steps of 2 dB. For easy checking also put your SNRs and the corresponding BER values in a table.

7) Plot the BER curves as a function of SNR for a BPSK communicator operating over a Rayleigh flat fading channel. Update the Rayleigh channel every symbol (not every sample).

8) Plot the BER curve as a function of SNR for a BPSK communicator operating over a Ricean flat fading channel.

9) Plot the BER curve as a function of SNR for a BPSK communicator operating over a frequency selective Rayleigh channel consisting of two ray with a separation of half a symbol period.

10) Repeat the previous 4 questions but instead consider a DBPSK communicator.