Laboratory 4 – Sampling and Discrete Fourier Transform

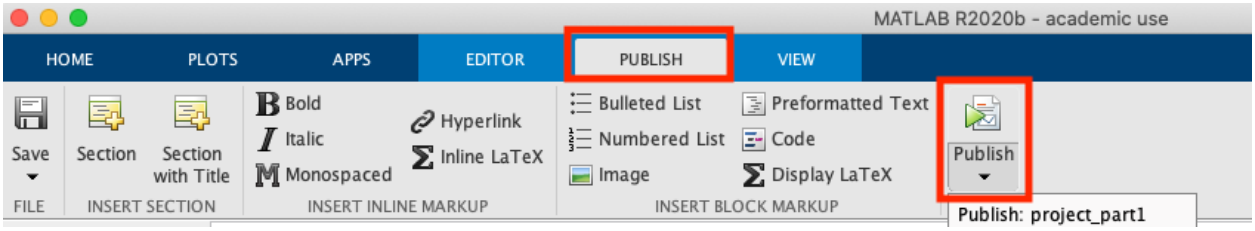
Introduction:

The objectives of this lab are to:

1. study the signal sampling and effect of sampling frequency;
2. verify the properties of DFT, and identify frequency components in a given signal.

There are **2** exercises in this lab.

You need to submit a lab report and all m files used in your experiment. You can print your code and plots using "PUBLISH" function in MATLAB, as shown below. It publishes code, results, and plots. You can cut and paste code and results to your report.



The lab report should consist of following sections.

1. A cover page including the title, name, student number, date, and location.
2. A table of content.
3. Introduction of the lab.
4. Equipment: list equipment and software used.
5. Results and Discussion: this is the main part of your lab report. Please include your codes, and plots. Note: the code must be well documented. If you were asked to compare experimental results to the calculations, or to explain parts of your experiment, it should be presented in this section.
6. Conclusion: state what you learn from this lab, lab objectives you achieved, and any difficulties you met.
7. Appendix: for MATLAB code used in lab (or you can include your code in Results and Discussion section).

The lab report should submit to course website under "Lab" section before the deadline.

Files to be submitted:

Your written lab report and MATLAB code (you can include your code in Results and Discussion section or in Appendix).

Grading details:

75% of your lab grade is for completing all lab requirements correctly. 15% is for clear writing, correct format, good presentation, and well-documented code; 10% is for extra work or analysis or going beyond the lab requirements.

Lab instructions:

Introduction to MATLAB

1. MATLAB command to read, write, and play audio files

MATLAB provides functions that allow you to read and write sound files in various formats. A list of audio file formats can be found at https://en.wikipedia.org/wiki/Audio_file_format. The commonly used sound file format on PC is WAV (https://en.wikipedia.org/wiki/WAV). The WAV file stores audio signal as pulse code modulation (PCM) data. In the WAV format, the amplitude of each sample is encoded in binary and stored in the WAV file. The amplitude of each sample can be directly read from and written to a WAV file. In this lab, we will use following MATLAB command to read, write, and play audio files.

1. Audiowrite: Write audio files
   audiowrite(FILENAME, Y, FS) is a function that writes data Y to an audio file specified by the file name FILENAME, with a sample rate FS Hz (i.e. there are FS samples in a second). The file format to use when writing the file is inferred from FILENAME's extension. So to generate a WAV file, the FILENAME should be with file extension ".wav".

   Example:
   To write data X to audio file "myfile.wav" at sampling rate of 4000Hz,
   Audiowrite('myfile.wav',X, 4000);

2. audioread: Read audio files
   [Y, FS]=audioread(FILENAME) reads an audio file specified by the string FILE, returning the sampled data in Y and the sample rate FS, in Hertz.

   Example:
   [y,fs]=audioread('myfile.wav');

3. sound: Play vector as sound.
   sound(Y,FS) sends the signal in vector Y (with sample frequency FS) out to the speaker on platforms that support sound. Values in Y are assumed to be in the range -1.0 <= y <= 1.0. Values outside that range are clipped.

4. audioinfo: Information about an audio file.
   INFO = audioinfo(FILENAME) returns a structure whose fields contain information about an audio file. FILENAME is a string that specifies the name of the audio file.

FILENAME must be in the current directory, in a directory on the MATLAB path, or a full path to a file.

Example:
Audioinfo('myfile.wav')

2. MATLAB commends for DFT

MATLAB has functions to perform DFT and inverse DFT. These functions are listed below:

5. fft(X): Discrete Fourier transform
   fft(X) is the discrete Fourier transform (DFT) of vector X. For matrices, the fft operation is applied to each column.

6. fft(X,N): N-point fft
   fft(X,N) is the N-point fft, padded with zeros if X has less than N points and truncated if it has more.

To extract frequencies from N-point FFT, you need to normalize the fft plot. This can be done as follows:

```
y=fft(x,n);
nlen=length(y);
f=(0:1/nlen:1-1/len)*fs;  %fs is the sampling frequency
plot(f,y);
```

MATLAB also provides function to plot frequency response, i.e. freqz.

freqz: Frequency response of digital filter
[H,W] = freqz(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter:

$$H(e^{j\omega}) = \frac{b_0 + b_1 e^{-j\omega} + \cdots + b_m e^{-jm\omega}}{a_0 + a_1 e^{-j\omega} + \cdots + a_n e^{-jn\omega}}$$

given numerator and denominator coefficients in vectors B and A.

Example:
Given an impulse response: h(n)={1,1,2,1,1}, the frequency response can be plotted by:

```
>>freqz(h,1);
```

Other useful commands in MATLAB:

1. abs(X), is the absolute value of the elements of X. When X is complex, abs(X) is the magnitude.
2. angle(X), returns the phase angles, in radians, of a matrix with complex elements.
3. max(X), finds the largest element in X.
4. min(X), finds the smallest element in X.

Use MATLAB "help" command to find more details of above functions.

Laboratory Exercise 1

Q1. Study the following Matlab code:

```
clear all
a=0:(1/8000):2;
f=150*a;
x=sin(2*pi*f.*a);
subplot(2,1,1), plot(a,f)
subplot(2,1,2), plot(a,x)
audiowrite('sound1.wav',x,8000);
sound(x,8000);
```

(1) Based on your understanding of the code, how many samples are there in 1 second, or what is the sampling frequency?
(2) Run the program and listen to the generated sound. Explain why it sounds in the way it does.

Q2. The keys of a piano generate musical notes at given frequencies. For a list of these frequencies, check https://en.wikipedia.org/wiki/Piano_key_frequencies.

(1) Write a Matlab code to generate a sine wave over the C major scale (key numbers: 40,42,44,45,47,49,51,52), each note lasting 0.5 seconds, with the amplitude in the range of -1 to 1 and the sampling frequency of 8000Hz. Save the data to a WAV file called Cscale.wav.
(2) Listen to the Cscale.wav to make sure you generate a correct C major scale.
(3) Read the C major scale to the vector Y. Listen to the C major scale with different sampling frequencies of 4000Hz, 8000Hz, and 16000Hz. Do they sound same? Why?
(4) Read the C major scale to Y, and write a Matlab code to generate Z[k]=Y[-k] and write Z to a WAV file, called CscaleZ.wav. Listen to CscaleZ.wav. How CscaleZ.wav differs from Cscale.wav? (***Please save Cscale.wav and CscaleZ.wav, these files are required in Lab Exercise 2***.)

Q3. In Matlab, type "help audiowrite". You will find the following explanations about the valid range of the data in Y:

The valid range for the data in Y depends on the data type of Y. Supported data types are as follows:

Data Type of Y	Valid Range for Y

---

uint8 **0 <= Y <= 255**

int16 **-32768 <= Y <= +32767**

int32 **-2^32 <= Y <= 2^32-1**

single **-1.0 <= Y <= +1.0**

double **-1.0 <= Y <= +1.0**

Data beyond the valid range is clipped. If Y is single or double, then audio data in Y should be normalized to values in the range -1.0 and 1.0, inclusive.

(1) Read in the Cscale.wav file created in Q2 to ***x*** using audioread, examine what is the valid range of data. Take the FFT of ***x*** and plot its magnitude response to verify its frequency is the same as the key number 40, play the signal ***x*** to verify that it sounds like a C major.
(2) Change the program in Q2 to generate the same C major signal with peak magnitude of 20, i.e. in the range of -20 to 20, save the data to a WAV file, Cscale2.wav. Do you receive any warning or error message?
(3) Read Cscale.wav and Cscale2.wav to vectors CY1 and CY2, respectively. Listen to both CY1 and CY2. Do they sound the same? Explain what happened to data CY2 by plotting both waveforms.
(4) Take the FFT of both CY1 and CY2 and plot the magnitude responses of CY1 and CY2. You can do this by using Matlab command: y=fft(x), and plot(abs(y)). What is your observation?
(5) Read in the sound1.wav in Q1, change the peak magnitude to 20 and save it to sound2.wav. Do you receive any warning or error message? Listen to both sound1.wav and sound2.wav. Do they sound the same? From the observations you made in (3) and (4), please explain what happened to sound2.wav.

Laboratory Exercise 2

Q4. The DFT can only tell us which frequencies are present in a signal as you did in Lab Exercise 1. But, it cannot tell you how a melody flows over time. If you want to see the frequency contents over time, what you would do is to compute the Fourier transform of several short successive sections of the original signal. A representation of this sort is called the short-time Fourier transform (STFT). This exercise is to show you how to perform the STFT.

(1) In Matlab, read two audio files CScale.wav and CScaleZ.wav. Perform the DFT on each signal and plot the magnitude of each signal. Can you tell the difference between these two signals based on the plots?
(2) Let's perform short time Fourier transform on the audio signal from CScale.wav. Note that the signal read from CScale.wav consists of 8 "keys" in one vector. Recall that the

duration of each key is 0.5 second and sampling frequency is 8000Hz when you create the signal. How many points are there in each "key"?

(3) In order for you to perform STFT, you need to split these 8 "keys" into 8 separate signals, e.g. $x1$, $x2$, …, $x8$. Write a Matlab program to split the C Scale signal into 8 vectors, each vector contains one "key". After the separation, plot the magnitude response of any one of the "keys" to verify you did it correctly. This process is referred as windowing, i.e. you cut a long sequence into several pieces, where the length of each piece is determined by the window. Note the window length is the number of points you determined in (2).

(4) Create a "for" loop, calculate the FFT of the windowed signal, i.e. $x1$, …, $x8$. The number point of DFT should be the same as the window length. Store FFT results to x1fft, …, x8fft.

(5) Every fft produces a vector of complex numbers. Reshape them as a column vector. Put all these column vectors in a matrix. Call this matrix "spect". Note that successive frames of the original signal are now represented as successive columns of the original time function (rather than rows as they were before). The rows of "spect" represent different frequency components, and the columns represent different time segments.
Add the following lines to your Matlab script for display purpose.

```
figure
spect_mag=20*log10(abs(spect));
t=(0:window_length:(length(x)-window_length))/fs;
f=(1:window_length)*fs/window_length;
imagesc(t, f, spect_mag);
axis xy
colormap(jet)
colorbar
```

Run your script to generate the frequency components of C Scale (this plot is called spectrogram). Does it match the key frequencies you generated?

Read in the audio file 'CScaleZ.wav', Repeat the steps to create a spectrogram for CScaleZ signal. What are the differences between the spectrogram of CScale and CScaleZ? Compare it with the fft you did in (1).

Q5. Matlab has a built-in spectrogram function, spectrogram. Run Matlab's spectrogram for both CScale and CScaleZ using the call:

```
>>spectrogram(x, [], 8000)
```

Compare the results with what you generated. Explain what your findings are.

Type: 'help spectrogram' in Matlab to get details of spectrogram function.