University of Washington – ECE Department
EE235 Lab 5 – Time Domain to Frequency Domain
Background Material

In this lab, we will learn how to transform signals from the time domain to the frequency domain and identify the frequencies of $e^{j\omega t}$ that comprise a periodic signal. The concepts that we'll focus on are finding the Fourier Series coefficients a_k of x(t), identifying frequency components of x(t), and understanding the relationship between Fourier Series coefficient index k and frequency ω.

Concepts TO REVIEW	Concepts You Will Learn
• Generating a signal in time domain • Playing audio signal • Subplots – usage, labeling, titles • Extracting a list of elements in a vector • Vector concatenation • Loops, functions	• Using fft and fftshift functions • Finding magnitude and phase • A few other useful functions

1.0 Frequency Analysis on a Computer

In this lab, because we are working on a computer, the signals are discrete-time signals with a finite length. As a result, the Fourier transform of those signals are discrete and finite in length. That is, we actually use a discrete Fourier transform (DFT), which differs from the continuous-time Fourier transform that you in class in two ways: i) the frequency domain is discrete, and ii) the frequency domain is over the window [-fs/2, fs/2) where fs is the sampling frequency (in Hz).

In order to compute the DFT, we use the Fast Fourier Transform (FFT) using numpy in Python (i.e., import numpy as np).

```
xf = np.fft.fft(x,nfft)          # FFT of time signal x nfft frequency samples
xfs = np.fft.fftshift(xf)        # shift FFT of xf to be centered around 0
yt = np.fft.ifft(yf,nt)          # inverse FFT of yf, optional nt specifies length of yt
```

Function numpy.fft.fft() returns a vector whose frequency samples correspond to the range [0, fs). In order to make the frequency range be [fs/2, fs), we use function numpy.fft.fftshift. If your DFT is length N and the range is [0, fs), then the interval between frequency samples corresponds to $\Delta f = fs/N$. So, if you are interested in X(f), then you need to use the index that is the nearest integer to $f/\Delta f$. In this lab, we plot the spectrum of x(t), which is positive. Hence, you take the absolute value of the Fourier transformed signal using function numpy.abs().

2.0 Other Needed Functions

(1) Generating sinusoidal signals

```
numpy.pi        # the value of π
numpy.sin(x)    # sin(x), where x could be a single time or an array of time samples
numpy.cos(x)    # cos(x), where x could be a single time or an array of time samples
```

Here is a sample code to generate a sinusoidal signal with frequencies 941Hz and 1336Hz. ‘numpy’ is imported as ‘np’.

```
fs=8000
t=np.arange(0,0.25,1/fs)
d0=np.sin(2*np.pi*941*t)+np.sin(2*np.pi*1336*t)
```

(2) Finding the index whose element is larger than a threshold

Consider an array a = [3, 4, 1, -6, 6]. We want to make an array ‘b’ that contains indices of the elements of the array a, which is larger than 3. Then, we expect that b = [1, 4] because ‘4’ and ‘6’ are larger than 3 and their indices are 1 and 4, respectively. Here is a sample code.

```
threshold = 3
a = np.array([3, 4, 1, -6, 6])
ind = np.arange(len(a))
b = ind[a[ind] > threshold]
```

When finding the frequency component of a signal from the spectrum, we find the indices of elements in an array that represents the spectrum, which has the value larger than a predetermined threshold. (i.e., arrays ‘a’, ‘b’, and threshold denote the array representing a spectrum, array containing the indices, and predetermined threshold.)