ECE 4680 DSP Laboratory 1: Introduction to Python for Basic Signal Representation

This lab is to be worked by each student individually. Each student will submit a lab report, that in this case is in the form of a worked homework assignment. Show all Python code/command line input and plotted results.

Introduction

MATLAB is likely the most popular tool in use today by DSP developers, but Python with the *scipy stack* is gaining in popularity, and is freely available[1]. As refresher to past signals and system course work and as an introduction to Python, you will in this first lab perform some basic signal representation exercises. I highly encourage you to download and install scientific Python (version 3.8x or higher) on your own machine and then use Jupyter Lab to perform these exercises. For added practice in learning Python, you may want to first work the problems in MATLAB and then work them in Python. Note in the following I assume that `scikit-dsp-comm` is installed on your system.

Laboratory Exercises

Jupyter Notebook Quick Start

```
%pylab inline
#%matplotlib qt
import sk_dsp_comm.sigsys as ss
import scipy.signal as signal
from IPython.display import Audio, display
from IPython.display import Image, SVG
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
pylab.rcParams['savefig.dpi'] = 100 # default 72
#pylab.rcParams['figure.figsize'] = (6.0, 4.0) # default (6,4)
#%config InlineBackend.figure_formats=['png'] # default for inline viewing
%config InlineBackend.figure_formats=['svg'] # SVG inline viewing
#%config InlineBackend.figure_formats=['pdf'] # render pdf figs for LaTeX
```

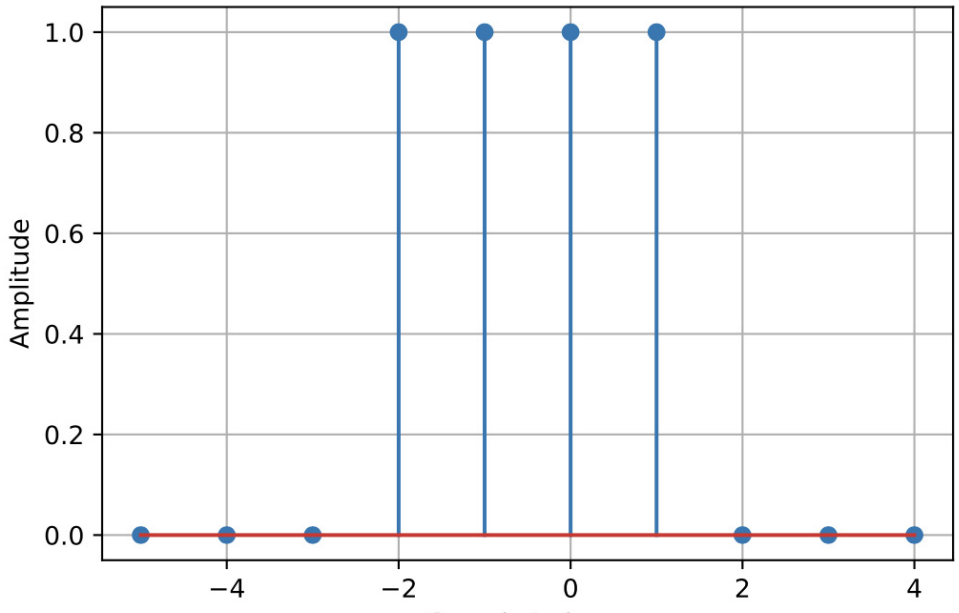
1. As of fall 2021 it is safe to say moving beyond Python there is now Julia which is very fast and has a broad-based developer community/eco system.

```
n = arange(-5,5)
w = ss.drect(n+2,4)
stem(n,w)
xlabel(r'Sample Index n')
ylabel(r'Amplitude')
ylim([-0.05, 1.05])
grid();
```

shift-tab in function argument will bring up context help

With the newest matplotlib graphics need to include a third argument:

```
stem(n,w,use_line_collection=True)
```



Discrete-time Signal Representation

1. Using the `stem()` function in Python (~~MATLAB~~) compute and plot the following functions

$$x_1[n] = \sin\left(\frac{\pi}{5}n\right), 0 \le n \le 15$$

$$x_2[n] = \cos\left(\frac{3\pi}{5}n\right), 0 \le n \le 15 \quad (1)$$

$$x_3[n] = \sin\left(\frac{\pi}{4}n\right)\cos\left(\frac{\pi}{4}n\right), 0 \le n \le 32$$

   What is the fundamental period of each signal?

2. Using the `stem()` function compute and plot the function

$$x[n] = \begin{cases} 2n+1, & -3 \le n \le 3 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

   on the interval $-8 \le n \le 8$. **Note**: No `for` loop is needed. Note the module `ss.py` has the function `ss.drect()` which can serve as a window function over the interval $n \in [-3, 3]$.

3. Using the `stem()` function compute and plot the function

$$x[n] = u[n]u[8-n] + u[n-2] - u[n-5] \tag{3}$$

on the interval $-2 \le n \le 16$. Note the module `sigsys` (aliased to `ss`) has the function `ss.dstep()` defined, which makes this problem very easy.

Transformation of the Time Index

4. Consider the signal

$$y[n] = \begin{cases} 2, & n = 0 \\ 1, & n = 1 \\ -1, & n = 3 \\ 3, & n = 4 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Hint: The intent of this problem is that you a simple function:

```
def y(n):
    """
    Use the sigsys function dimpulse to directly define
    the waveform function y[n]
    """

    # Write one line of code
    return yy
```

Now you can answer the below using a shifted index array `n` to drive your function.

a) Using the `stem()` function plot $z_1[n] = y[n-2]$

b) Using the `stem()` function plot $z_2[n] = y[n+2]$

c) Using the `stem()` function plot $z_3[n] = y[-n]$