Homework 9 **ECE 302H: Introduction to Electrical Engineering**

Problem 9.1 – Python Tutorial (1.5h)

a) Please visit https://www.anaconda.com/products/individual and download and install *Anaconda*, a python distribution which includes a variety of Python libraries as well as useful GUI tools, including *Spyder*.

b) Open *Spyder*. *Spyder* is an integrated development environment (IDE), a graphical application that allows you to write code, run code, and perform a variety of other tasks

c) The left-hand window is the Editing window where you will type code. The bottom-right window is the Console, or the output window. This is where you will see calls to Python code and the output of that code. The top-right window gives you useful insights into what your program thinks it's doing. Click on the "Variable Explorer" button to make this window display the variables in your program and their current values.

d) Your code is currently blank except for some comments at the top. Please add the code `x = 2` and press the green Play button at the top of the screen to run your code. Notice that the Console (bottom-right) indicates that your code has been run, but otherwise there is no output. This is because we haven't told the program to output anything. Notice also that the Variable Explorer (top-right) indicates some information about the program. What values does the Variable Explorer give?

e)

Variable		This indicates the name(s) of the variable(s)
Type		This indicates the type of variable, distinguishing between characters, integers, floating-point numbers, etc.
Size		Spyder expects that you will use a lot of *nxm* arrays (matrices), so it includes this in the Variable Explorer. Vectors will be nx1 or 1xm and scalars will just have a value of 1.
Value		The value of the variable

We will use a lot of arrays and vectors in this course and throughout your engineering education. To create arrays and vectors in Python, we need to use the Numpy package. To enable this package, place the following line at the top of your code (under the comments): `import numpy as np`. Next, do not modify your existing code but add at the bottom the following line: `x = np.array([[5,6,7]])` and run the code again. Notice that the Console again says that the program has been run but produces no other outputs. Notice also that the Variable Explorer now shows the variable `x` differently. As the code runs, your first command (`x = 2`) still exists, but the next command (`x = np.array([[5,6,7]])`) overrides the variable x and replaces it with a new type, a new size, and a new value. Not all programming languages can do this, and in some languages (like C) trying to write a variable twice with different types will result in an error. Python allows you to change variable types on the fly, for better or worse. What values does the Variable Explorer give now?

f)

Variable		This indicates the name(s) of the variable(s)
Type		This indicates the type of variable, distinguishing between characters, integers, floating-point numbers, etc.
Size		Spyder expects that you will use a lot of nxm arrays (matrices), so it includes this in the Variable Explorer. Vectors will be nx1 or 1xm and scalars will just have a value of 1.
Value		The value of the variable

Variable x is now a nxm array, where n is the number of rows and m is the number of columns. When either n or m is 1, we call the array a vector. When n is 1, the vector is specifically called a "row vector" because its elements are written horizontally as a row. When m is 1, the vector is specifically called a "column vector" because its elements are written vertically as a column. Is **x** a row vector or a column vector?

g) Notice that that we feed a value into the np.array() command that uses two sets of square brackets [[5,6,7]]. The outer set of brackets encompasses the entire array. The inner brackets represent one row. If we want to create an nxm matrix, we need to add more rows. Add the following code to the end of your program: `x = np.array([[1,2,3],[4,5,6],[7,8,9]])` . When you run the program now, the code will first produce **x** as a scalar (2), then replace it with a vector ([5,6,7]), then replace it with a 3x3 matrix/array. What code would you use to write **x** as a column vector with entries 5,6,7?

h) Vectors are useful for creating graphs. Think about what a graph represents – each point on the graph represents an x value that corresponds to a certain y value. Therefore, if we create a vector of x values and a vector of y values, then we can use the first x and y values to represent one point on the graph, the second x and y values to represent another point, and so on. Let's create a vector of evenly-spaced values to represent the x axis. Please add the following line to your program: `x = np.arange(-10,10,1)` , where the first argument is the starting point, the second argument is the end point, and the third argument is the spacing between entries . Run the code and look at the Variable Explorer. What is the last entry in the vector **x**? Is it what you expect based on your code? What conclusion might you reach about how np.arange interprets its inputs?

i) Please type a new command to make **x** a vector of evenly-spaced values starting from -20 to +20 (inclusive) with a spacing of 4 between values. What is this command?

j) Suppose we would like to plot y = x^2. We have a vector of values for x – we just need to take each of those values and produce a corresponding value for y. In other words, we want to do an *element-wise* operation on x, where we separately do the same operation on each entry of x. Fortunately, this is the default type of operation for the Numpy "array" object. Add a new command `y = x*x` and run the program again. Check some of the entries of y in the Variable Explorer. Check that y has the number of entries you expect and the values of the entries are what you expect.

k) We now have all of the data points we need – we just need a command to produce a visual graph. We get commands of this kind from the Python package called "matplotlib." We need to include "matplotlib" the same way we did with "Numpy." Please add the following command to the beginning of your code, right after where you import "Numpy": `import matplotlib.pyplot as plt` .

l) Now copy the following command at the end of your code: `plt.plot(x,y,label='Quadratic - Rough')` . Run the program, and notice that you don't see any output. This is because Spyder by default puts plots in a special place to keep things organized. In the top-right window of Spyder where you've been using the "Variable Explorer," click on "Plots" to see your plot.

m) This way of interacting with plots can be annoying. To get a better view, go to Preferences > Preferences > IPython Console > Graphics and change the Graphics Backend from Inline to Automatic. You must restart Spyder to make this change take effect – don't forget to save your program! After restarting Spyder, run your code again and you should see your graph in a new window, with some GUI buttons for zooming and panning and so forth.

n) Let's add a second curve to the same plot. Add the following line to your code after your other calculations but before the plotting commands: `y2 = 200 - 3*x`. Note that y2 is a new variable and x is the same independent vector as before. Plot this curve by including `plt.plot(x,y2,label='Linear')` *after* the line of code that plots (x,y).

o) Notice that the plot for y=x^2 looks kind of rough. Why doesn't the plot for y2=200-3x look bad? How would you create a smoother plot for y=x^2? Write your answer in words then implement this using the variable name y3 and the label name 'Quadratic - Smooth'.

p) It's important to always add explanatory information to any plot that you want to use to communicate with others (and often for yourself, especially if you're going to come back to a bit of code at a later date). *You may lose points on any graph submitted for EE302 missing a title, axis labels with units, and some kind of legend*.

```
plt.title('ECE302 HW1 Problem 1 - Exploring Python')  #Always
ECE302 HWX Problem Y - Brief description
plt.xlabel('X Axis (Units)')
plt.ylabel('Y Axis (Units)')
plt.legend()
```

q) Please submit a screenshot of your Spyder window including all of the code and the final graph. If your settings produce graphs in separate windows, submit one screenshot that includes both the code and the graph.

Problem 9.2 – Fourier Series Plotting (1.5h)

Consider the sawtooth wave. One cycle of this wave looks like a straight line $f(\omega t)=\omega t$ from $-\pi$ to π. This cycle repeats in the positive and negative ω directions forever and is therefore periodic. The Fourier series for this wave is given by

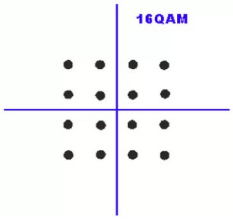
$$f(\omega t)=2\sum \frac{(-1)^{n+1}}{n}\sin(n\omega t)$$

Do the following in a single Python script and submit both your script and the final plots. Be sure to include a title, legend, x label, and y label. Note that you may not have been explicitly shown how to do each and every step here – a big part of learning to program is learning to research the commands for the tasks you want to perform.

a) Plot the sawtooth wave as a function of ωt from -3π to 3π.
b) Create a vector of ωt and use it to calculate a vector that represents the fundamental of the Fourier series. Plot it on top of the original sawtooth wave in a different color and style.
c) Calculate the sum of the first 3 harmonics of the sawtooth wave and plot it on top of the previous two plots in a different color and style.
d) Do the same again for the sum of the first 10 harmonics.
e) On a separate plot with two subplots, use the stem command (if you imported matplotlib as plt, then the stem plot command is plt.stem) to plot the magnitude and phase of the first 10 harmonics of the Fourier series of the sawtooth wave. Plot the magnitude on a log-log plot and the phase on a semilog-x plot.

Problem 9.3 – Quadrature Amplitude Modulation for Wireless Communication (1.5h)

Wireless communication systems send signals using cosine waves at high frequencies (anywhere from MHz to several GHz). These cosine waves are modulated, or changed slightly over time, to represent different digital values. Older modulation schemes would modulate frequency (FM) or amplitude (AM) to convey information. Modern systems modulate both the amplitude and the phase to convey more information in a single wave. As we've learned in class, a cosine wave with an amplitude and a phase can also be represented as a cosine wave plus a sine wave at the same frequency, each with their own amplitude but no phase. Wireless communications people usually think about their wave in the cosine sine form. The cosine part is called the "in-phase" component, and the sine part is called the "quadrature" component. Modulating both parts is called "Quadrature Amplitude Modulation" (QAM).



a) There are two variables that can be adjusted, the cosine amplitude and the sine amplitude (equivalent to the amplitude and phase of a single wave). These two variables can be plotted on a graph such as the Figure. Redraw the graph and label the x-axis "I" to represent the in-phase (cosine) amplitude. Label the y-axis "Q" to represent the quadrature (sin) amplitude.

b) The Figure represents a 16-QAM system. Each dot represents a digital value. How many bits are needed to represent all of the possible x coordinates? How many bits are needed to represent all of the possible y coordinates? Therefore, how many bits of information can be contained in a single incoming wave?

c) One way to assign digital values to the dots is to use "Gray Coding." Gray coding is not standard binary counting – it's an alternative in which increasing the number by one only flips a single bit from 1 to 0 or 0 to 1. Look up Gray Code in Wikipedia and create a table that shows the numbers 0 through 15, their Gray code representation, and also their ordinary binary code representation.

d) Use Gray coding to assign digital values to the dots in your figure. Use the two MSBs to represent the in-phase coordinate from most-negative to most-positive. Use the two LSBs to represent the quadrature component from most-negative to most-positive. A system that receives a cosine wave with a certain amplitude and phase will decompose that wave into its cosine (in-phase) and sine (quadrature) parts, look at their amplitudes, and find the closest dot on the graph – the system will then interpret the incoming cosine wave as the digital value assigned to that dot.

e) Each dot has an I and Q amplitude of either 1 or 3, so they are evenly spaced. Calculate the amplitude-phase representation of the [1011], [1111], [1010], and [1110] dots. Do you notice any correlation between the amplitude and phase of your answers and the locations of the dots on the graph?

f) If you wanted to have 64 dots instead of 16 without making the graph any larger (meaning, without having to design a circuit that can deal with any larger voltages), how many times more accurate would the transmitter need to be in creating I and Q components? How many times faster could your bitrate be?

Problem 9.4 – Complex Arithmetic Boot Camp (30m)

Complex numbers can be expressed as a real part plus an imaginary part, $z = r + qj$, where $j = \sqrt{-1}$. This is called rectangular form. They can also be expressed as a complex exponential, $z = M e^{j\phi}$. This is called polar form. In polar form, ¿ M ∨ ¿ is called the magnitude, i.e. the distance from (0,0) to z on the complex plane, and ϕ is called the angle or the argument, i.e. the angle from the real (+x) axis to z on the complex plane. It is easier to add/subtract complex numbers in rectangular form, while it is easier to multiply/divide complex numbers in polar form.

a) Express $z = 3 + 4\,j$ as a complex exponential (polar form) and plot in on the complex plane
b) Express $z = -6 + 8\,j$ as a complex exponential (polar form) and plot in on the complex plane
c) Express $z = -\,j - 4\,j$ as a complex exponential (polar form) and plot in on the complex plane
d) Express $z = 2\,j$ as a complex exponential (polar form) and plot in on the complex plane
e) Express $z = \sqrt{-1 + j}$ as a complex exponential (polar form) and plot in on the complex plane
f) Express $z = 2e^{j\pi/6}$ in rectangular form and plot in on the complex plane
g) Express $z = -3e^{-j\pi/4}$ in rectangular form and plot in on the complex plane
h) Express $z = \sqrt{3}e^{-j3\pi/4}$ in rectangular form and plot in on the complex plane
i) Express $z = -\,j^3$ in rectangular form
j) Express $z = -\,j^{-4}$ in rectangular form
k) Express $z = (2 + j)^2$ in rectangular form and polar form
l) Express $z = (3 - 2\,j)^3$ in rectangular form and polar form
m) Calculate the magnitude and phase of $T = \dfrac{1 + 3\,j}{2 - 7\,j}$
n) Calculate the magnitude and phase of $T = \dfrac{1}{3 + 4\,j}$