Lab I: Complex Numbers

At this point, you must download and install the SPFirst toolbox. This toolbox is found on the author's website:

http://spfirst.gatech.edu/matlab/

For students who have purchased official copies of MATLAB, or have obtained illegally downloaded copies, installation instructions are found on the above link. For view.ece.ufl.edu users of MATLAB, those same instructions should be followed, but the SPFirst folder should be downloaded and unzipped in the appropriate location on the view.ece.ufl.edu machine. Users of UF Web Apps, a university service to help students avoid having to pay for expensive software such as MATLAB, should utilize the guide that we have created for this purpose.

To test your installation, run the complex numbers program called `zdrill` by typing `zdrill` into your command window. If SPFirst has been successfully installed, you should be able to demo this program. If it doesn't work, then your SPFirst toolbox has not been correctly installed, and you should ask your TA and/or your classmates for installation help.

These are common MATLAB complex number operators dealing with complex numbers:

Complex conjugate:	`>>conj()`
Magnitude:	`>>abs()`
Phase in radians:	`>>angle()`
Real part:	`>> real()`
Imaginary part:	`>>imag()`
$\sqrt{-1}$:	`>>i, j`
Defining a complex number:	`>> x = 3 + 4i`
Defining a complex number:	`>> x = 3 + 4j`
$e^{j\theta}$:	`>>exp(j*theta)`

Lab Part One

1.1 Complex Numbers in MATLAB

Before proceeding in this section, make sure that you have watched the lecture video on Euler's formula, or at least have read on the Internet about Euler's formula: $e^{j\theta} = \cos\theta + j\sin\theta$

(a) Complex numbers are easy to implement in MATLAB, using the letter `i` or `j`., so long as you have not overridden MATLAB's assignment of those letters. It makes complex operations easy; **try the following** to see:

```
>> z = 3 + 4i, w = -3 +4j  % Declaring complex numbers
>>real(z), imag(z) % Real and imaginary components
>>abs([z,w])    % Computes magnitudes
>>conj(z+w)     % Computes complex conjugate of the sum
>>angle(z) % Computes the phase of the complex number
% Also written as arctan(imag(z)/real(z))
```

```
>>exp(j*pi)     % Using Euler's formula, cos(pi)+j*sin(pi)
>>exp(j*[pi/4, 0, -pi/4])
```

(b) MATLAB can compute complex valued formulas and display the results as phasor diagrams. Use the following `zvect` function to plot five vectors all on one graph. **Include this plot in your report:**

```
>>zvect([1 + j, j, 3 - 4*j, exp(j*pi), exp(2j*pi/3)])
```

If `zvect()` doesn't work, there is likely a problem with the installation of your SPFirst toolbox.See Lab 1 for installation details, or ask your TA.

(c)Use $z_1 = 10e^{-j2\pi/3}$ and $z_2 = -5 + 5j$ for all parts of this section (hint: save them in your workspace).

1. Enter z_1 and z_2 into MATLAB. **Plot** them using `zvect()` and **print** them with `zprint()`.
Superimpose an x-y axis and the unit circle on your plot by using the commands
```
>>hold on, zcoords, ucplot, hold off
```
2. **Execute** `zcat([j, -1, -2j, 1]);` . **What does `zcat()` do** with a vector of complex numbers?

3. **Compute $z_1 + z_2$**, and plot the sum using `zvect()`. Use `zcat()` to plot z_1 and z_2 as two vectors head to tail. Using `hold on`, **plot all three vectors (z1, z2, $z_1 + z_2$) on the same plot**. Using `zprint()`, **display the numerical values** of each of these.

4. **Compute** the product $z_1 z_2$ and **display the numerical result**. Plot z1, z2, and z1z2 on the same graph using `zvect()`. **What is the relationship** between the two initial angles and the angle of the product?

5. **Compute** the quotient $\frac{z_2}{z_1}$, **display** the numerical result, and **plot** it on a plot with z1 and z2.

6. **Compute** the conjugates $z_1{}^*$ and $z_2{}^*$, **display** the numerical results, and **plot** them.

7. **Compute** $\frac{1}{z_1}$ and $\frac{1}{z_2}$, **display** the numerical results, and **plot** them.

(d) Use the MATLAB editor to **create a script file** called `mylab1.m` containing the following code:

```
tt = -1:0.01:1;
xx = cos(5*pi*tt);
zz = 1.4*exp(j*pi/2)*exp(j*5*pi*tt);
plot(tt, xx, 'b-', tt, real(zz), 'r--'), grid on
title('TEST PLOT OF A SINUSOID')
xlabel('TIME (sec)')
```

Explain why the plot of `real(zz)` is a sinusoid. What is the phase and amplitude of it? Calculate the phase based on a time-shift measured from the plot.

(e) To run your script, simply type `mylab1` into the command window. **Take a screenshot of `mylab1` running in the command window.**

1.3 MATLAB Sounds

Run the MATLAB sound demo by typing `xpsound`. If you don't hear anything, check your speakers and volume settings. If running this through UF Apps, you will not be able to hear this demo.

Sounds are represented by sinusoids, so to create sound using MATLAB, you need to create a sinusoid. Your script `mylab1.m`creates a sinusoid with frequency 2.5 Hz.**Modify your script so that it produces a 2000 Hz sound, with sampling frequency 11025 Hz, which is 0.9 seconds long. The appropriate time vector is therefore:**

```
tt = 1/11025:1/11025:0.9;
```

Now use `soundsc()`(type `help soundsc`) in order to listen to the sound. If you are using UF Apps, you will need to save the audio file in order to hear it. Type `help audiowrite` to find out how. Play your generated sound for your TA, who will give you the password showing that you passed this section.
What is the password?

What is the length of your `tt`vector?

Lab Part Two

2.1 Generating Sinusoids

Write a script to do steps (a)-(d) below. Include this script in your submission.

(a) Create a time vector (`tt`) that is two cycles of the 4000Hz sinusoid defined in (b). Define `tt` similar to in part 1.3. Set `T` equal to the period of the sinusoid, and define the start of `tt`to be `-T`, and the end to be `+T`. **Make sure that the sine wave has at least 25 samples per period.** When you use the colon operator, make sure the increment (the 2nd of the 3 numbers in colon notation) is small enough to generate 25 samples per period.

(b) Generate two 4000Hz sinusoids:

$$x_1(t) = A_1\cos(2\pi(4000)(t - t_{m1})) \qquad x_2(t) = A_2\cos(2\pi(4000)(t - t_{m2}))$$

Let A_1 be equal to your age and set $A_2 = 1.2A_1$. If D and M are the day and month of your birthday and T is the period of the sinusoid, set$t_{m1} = \frac{37.2}{M}T$and $t_{m2} = \frac{-41.3}{D}T$.

(c) Create a third sinusoid that is the sum: $x_3(t) = x_1(t) + x_2(t)$.

(d) Plot all three sinusoids over the range $-T \leq t \leq T$. Use `subplot(3,1,1)`, `subplot(3,1,2)`, and `subplot(3,1,3)` (see `help subplot`). Put a title on each subplot, and include your name in one of the titles (see `help title`, `help print`, and `help orient`)

(e) **Write a function to generate a sinusoid**, $x(t) = Acos(\omega t + \varphi)$, by using four inputs: amplitude, frequency, phase, and duration. The function should have two outputs: the values of the signal and the corresponding times at which values are known. The function should generate 20 values of the sinusoid per period. Call the function `one_cos()`. Use `goodcos()`, written as part of Lab 1 last week, as a starting point. **Submit the commented code for this function, and plot the output for the following:** $A = 95, \omega = 200\pi \frac{rad}{s}$, $\varphi = \frac{\pi}{5} rad$, $dur = 0.025s$. **What is the expected period in milliseconds?**

2.2 Analyzing Sinusoids

Remember from lecture that the phase of a sinusoid is a measure of the current location within a period. 2πshift in phase is akin to a full period of movement, which is related through a frequency to the passage of some value of time. Given two out of three values - the phase, the frequency, and the time location of a peak -the third value can be calculated.

Given a frequency, the phase of a sinusoid may be calculated by measuring the time location of a positive peak.

(a) Measure the time-location of a positive peak and the amplitude of the plots of $x_1(t)$and $x_2(t)$. Then calculate by hand the phases of the two signals by converting each time shift, $t_{mi}$to phase, using the known frequencies of $x_1(t)$ and $x_2(t)$.

(b) Measure the amplitude A_2 and the time-shift $t_{m3}$of $x_3(t)$from the plot and calculate the phase by hand. Annotate the plot to show how the time-shift and amplitude were measured, and how the phase was calculated (use something like MS Paint).

(c) Use phasor addition of the complex amplitudes for $x_1(t)$ and $x_2(t)$ to determine the complex amplitude of $x_3(t)$. **Use this answer to verify that your previous calculations of A_3 and φ_3 were correct. State your percent error.**

2.3 Complex Amplitude

Write one line of MATLAB code that will generate $x_1(t)$above by using the complex-amplitude representation:

$$x_1(t) = Re\{Xe^{j\omega t}\}$$

2.4 Concatenation

In MATLAB, two vectors can be concatenated through the `cat` command, or through the notation:

```
zz = [xx, yy];
```

Run a script or function for each example for your lab doc.

(a) **Run** `soundsc()` **on an input signal/sinusoid**: Using $f_s = 11025\ \frac{samples}{s}$, instantiate a time vector of 0.5 second duration. Create a vector `x1` of samples of a sinusoid with amplitude $A = 100$, angular frequency $\omega = 2\pi(800)$, and phase $\varphi = -\frac{\pi}{3}$. Use `soundsc(x1)` to play the vector and listen to the output. Note: `soundsc()` may not work for you, in this case use `audiowrite()` to create a *.wav* file, and play outside MATLAB.

(b) **Run** `soundsc()` **on a second input signal/sinusoid:** Create a vector `x2` of samples of a sinusoid with $A = 100$, $\omega = 2\pi(1200)$, and $\varphi = \frac{\pi}{4}$. Use $f_s = 11025\ \frac{samples}{s}$and obtain a number of samples equivalent to a 0.8 second duration. Use `soundsc()` (or equivalent) to play the vector and listen to the output. **How does this sound compare to the output of (a)?**

(c) **Concatenate the input signals together:** Concatenate `x1` and `x2` into a vector `xx`, leaving a duration of 0.1 seconds of silence in between (hint: use `zeros()`). Listen to this new signal to verify that it is correct. MATLAB uses brackets, `[]`, in order to concatenate things together, this just means putting signals together to hear them one at a time.

(d) **Plot and identify the two input signals:** Execute the following commands:

```
>> tt = (1/11025)*(1:length(xx));
>> plot(tt,xx);
```

This will plot a huge number of points, but you should be able to discern the "envelope" of the signal. Verify that the amplitude changes from 100 to 0, and then from 0 to 80 at the correct times.

(e) **Different sampling frequencies affecting pitch**: Call `soundsc()` on `xx` again (or use `wavwrite()/audiowrite()` and play the sound), but this time use $f_s = 22050\ \frac{samples}{s}$. **Describe how the duration and pitch of the signal were affected and explain.**

(f) Using the `tt` vector from the MATLAB sound portion of the warmup, create sounds for the following frequencies:

```
[523, 587, 659, 698, 784, 880, 988, 1047]
```

Concatenate the sounds together, and **submit the concatenated sound** (a musical scale) as a single .wav file using `audiowrite()` or `wavwrite()`. Alternately, play the sound for your TA, **ask for a password, and write that password here.**