# 2. Lab 0 - Part 2

## Table of Contents



# 2.1 - Generating Sinusoids

1. Create a time vector t that is two cycles of the 5000 Hz sinusoid defined in (b). (Hint: define t similar to in Section 1.3) Set T equal to the period of the sinusoid and define the start of t to be -T, and the end to be +T. Make sure that the sine wave has at least 25 samples per period. In other words, when you make the time vector, make sure the step size is small enough to generate 25 samples per period.

```
omega = 2*pi*5000;
T = 2*pi/omega;
t = linspace(-T,T,50);
```

2. Generate two 5000 Hz sinusoids

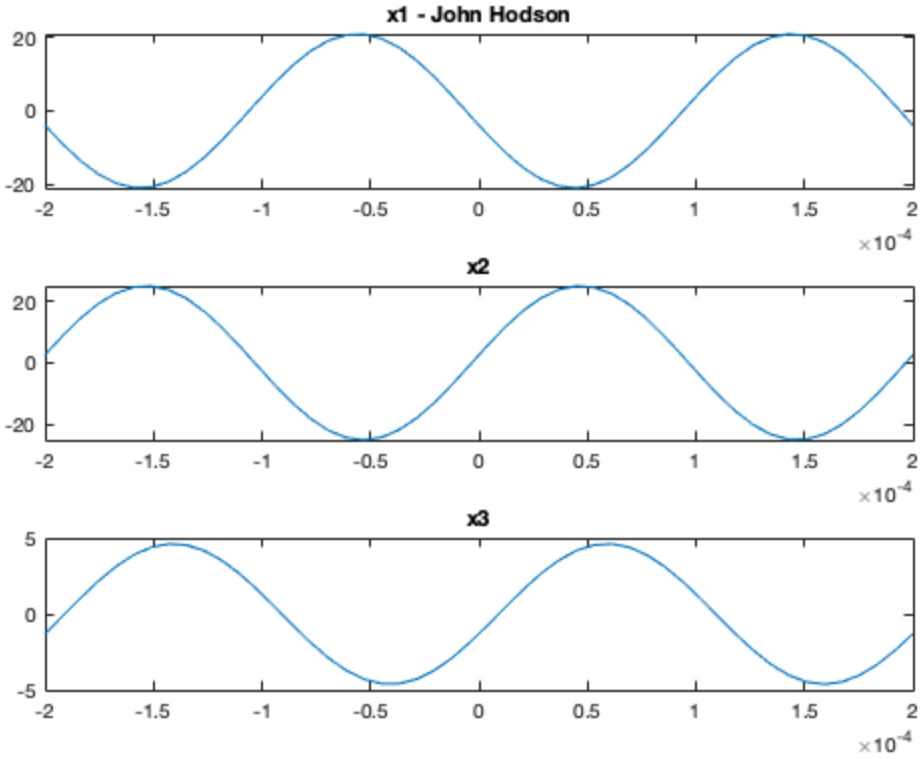
```
A1 = 21;
A2 = 1.2*A1;
D = 3;
M = 10;
tm1 = 37.2/M*T;
tm2 = -41.3/D*T;
x1 = A1 * cos(omega*(t-tm1));
x2 = A2 * cos(omega*(t-tm2));
```

3. Create a third sinusoid that is the sum:

```
x3 = x1+x2;
```

4. Plot all three sinusoids over the range -T < t < T. Use subplot(3,1,1) and 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)

```
subplot(3,1,1), plot(t, x1), title("x1 - John Hodson"),
subplot(3,1,2), plot(t, x2), title("x2"),
subplot(3,1,3), plot(t, x3), title("x3");
```




5.Write a function to generate a sinusoid, 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 exactly 25 values of the sinusoid per period. Call the function one_cos(). Use your answer to 1.3 of part 1 of this lab as a starting point.

```
[x,t] = one_cos(95,200*pi,.025,5);
```

Question 10: What is the expected period in milliseconds?

Expected period in milliseconds = 1/(angular frequency/(2*pi))*10^3 = 10ms

```
angular_freq = 200*pi;
1/(angular_freq/(2*pi))*10^3
```

```
ans =

    10
```

# 2.2 - Analyzing Sinusoids

1. Measure the time-location of a positive peak and the amplitude of the plots of x1 and x2. Calculate by hand the phases of the two signals by converting each time shift, to phase, using the known frequencies of 1 and 2.

```
tm1 = -5.306*10^-5;
```

```
phi1 = -omega*tm1

tm2 = 4.490*10^-5;
phi2 = -omega*tm2


phi1 =

    1.6669


phi2 =

   -1.4106
```

2. Measure the amplitude A2 and the time-shift tm3 of x3 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).

```
A3 = 4.610;
tm3 = 6.122*10^-5;
phi3 = -omega*tm3


phi3 =

   -1.9233
```

Refer to PNG image 2.2.2 included in the .zip file to see the annotated picture of the wave.

3. Use phasor addition of the complex amplitudes for x1 and x2 to determine the complex amplitude of x3. Use this answer to verify that your previous calculations of A3 and phi3 were correct. State and explain your percent error. Ae^(j*phi)*e^(jwt)

```
actual_sinusoid = A1*exp(-37.2/M*T*omega*j) + A2*exp(41.3/
D*T*omega*j);
actual_A3 = abs(actual_sinusoid)
A3_percent_error = (A3-actual_A3)/actual_A3*100
actual_phi3 = angle(actual_sinusoid)
phi3_percent_error = (phi3-actual_phi3)/actual_phi3*100


actual_A3 =

    4.6208


A3_percent_error =

   -0.2341
```

```
actual_phi3 =

   -1.8562


phi3_percent_error =

    3.6150
```

As seen above, the percent error was very low for both of these quantities. For the phase shift, my percentage error was 3.615%. For the amplitude, my percentage error was -0.23%.

# 2.3 - Complex Amplitude

1. Write one line of code that will generate x1(t) above by using the complex-amplitude representation:

```
x1 = real( A1*exp(j*omega*t)*exp(j*-tm1*omega) );
length(x1)


ans =

       25001
```

# 2.4 - Concatenation

1. Run soundsc() on an input signal/sinusoid. Using fs = 11025 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 w = 2*pi*800, and phase phi = ?pi/3. Play the vector and listen to the output.

```
fs = 11025;

T1 = 0.5;
t1 = linspace(0,T1,fs*T1); % generate fs*T points between 0 and T
A1 = 100;
w1 = 2*pi*800;
phi1 = -pi/3;

x1 = real( A1*exp(j*w1*t1)*exp(j*phi1) );
soundsc(x1)
```

2. Create a vector x2 of samples of a sinusoid with A = 80, w = 2*pi*1200, and phi = pi/4. Use fs = 11025 samples/s and obtain a number of samples equivalent to a 0.8 second duration. Play the vector and listen to the output.

```
T2 = 0.8;
t2 = linspace(0,T2,fs*T2); % generate fs*T points between 0 and T
A2 = 80;
w2 = 2*pi*1200;
phi2 = pi/4;
```

```
x2 = real( A2*exp(j*w2*t2)*exp(j*phi2) );
soundsc(x2)
```

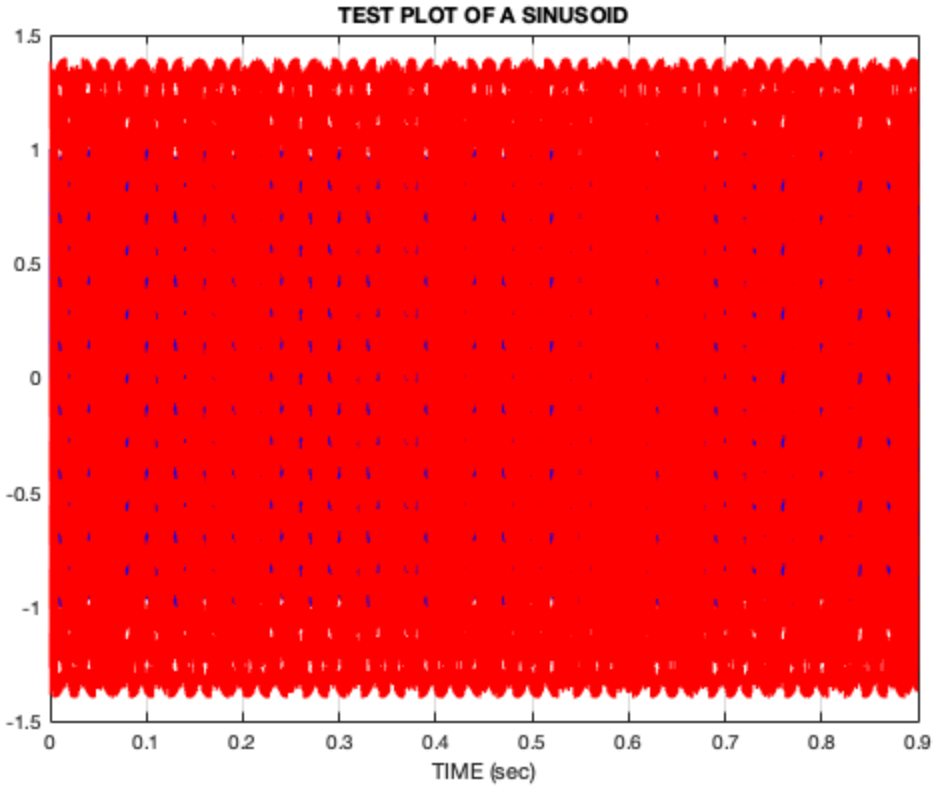
3. Concatenate the input signals together: Concatenate x1 and x2 into a vector x, leaving a duration of 0.1 seconds of silence in between.

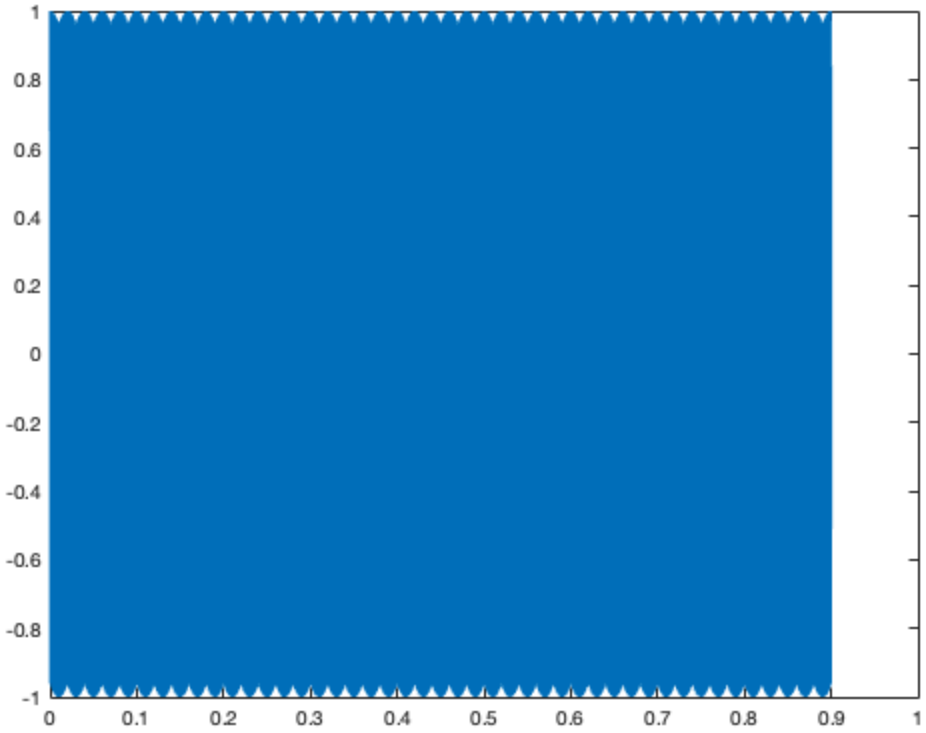
```
t3 = 0.1;
x = [x1,zeros(1,floor(t3*fs)),x2];

soundsc(x)
```

4. Plot and identify the two input signals. Execute the following commands, and plot:

```
figure(2) % plot on another figure for the purpose of publishing
mylab1;
tt = (1/11025)*(1:length(xx));
figure(3); % plot on another figure for the purpose of publishing
plot(tt,xx);
```






5. Different sampling frequencies affecting pitch: Call soundsc() on x again (or use audiowrite() and play the output file), but this time use fs = 22050 samples/s.

```
higher_freq_fs = 22050;
t1 = linspace(0,T1,higher_freq_fs*T1); % generate fs*T points between
 0 and T
higher_freq_x1 = real( A1*exp(j*w1*t1)*exp(j*phi1) );
t2 = linspace(0,T2,higher_freq_fs*T2); % generate fs*T points between
 0 and T
higher_freq_x2 = real( A2*exp(j*w2*t2)*exp(j*phi2) );
higher_freq_x = [x1,zeros(1,floor(t3*higher_freq_fs)),x2];
```

higher_freq_x is very similar to x, but it seems crisper. The duration is the same, as it was adjusted accordingly to be so.

6. Using the t vector from section 2.1.4 of this lab, create sounds for the following:

```
frequencies = [523, 587, 659, 698, 784, 880, 988, 1047];
output = [];

fs = 11025;
T = 0.5;
t = linspace(0,T,fs*T); % generate fs*T points between 0 and T

for i=1:length(frequencies)
    f = frequencies(i);
    w = 2*pi*f;
```

```
    current_wave = cos(w*t);
    output = [output, current_wave];
end
soundsc(output)
audiowrite("2.6.wav", output, 44100)
```

Question 11: How does this sound compare to the output of the previous sound?

In this sound, you can clearly hear the scale change over time and the 8 distinct parts. This wave is significantly different than the last because of this reason.

# 2.5 - Magic Square

```
mymagic(3) == magic(3)

mymagic(5) == magic(5)

mymagic(7) == magic(7)


ans =

  3×3 logical array

   1   1   1
   1   1   1
   1   1   1


ans =

  5×5 logical array

   1   1   1   1   1
   1   1   1   1   1
   1   1   1   1   1
   1   1   1   1   1
   1   1   1   1   1


ans =

  7×7 logical array

   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
   1   1   1   1   1   1   1
```

*Published with MATLAB® R2019a*