Lab 2: More Sinusoids

Lab Part One

1.1 syn_sin()

Since we will generate many sums of sinusoids in this course, it will be convenient to have a function for this operation, allowing the frequency of each component to be different.

In this section, you will write a function called `syn_sin.m` that will synthesize a waveform from the vectors of frequencies and complex amplitudes. You must write the function with one loop. Below is a template that you should base your solution on:

```
function [xx, tt] = syn_sin(fk, Xk, fs, dur, tstart)
%{
    syn_sin - Function to synthesize a sum of cosine waves.

usage:
        [xx, tt] = syn_sin(fk, Xk, fs, dur, tstart)
        fk = vector of frequencies (could be negative or positive)
        Xk = vector of complex amplitudes: A*e^(j*phi) for each fk
        fs = the number of samples per second for the time axis
        dur = total time duration of the signal
        tstart = starting time (default is 0, if you make this input
                 optional)
        xx = vector of sinusoidal values
        tt = vector of times, for the time axis

        Note: fk and Xk must be the same length:
        Xk(1) corresponds to frequency fk(1),
        Xk(2) corresponds to frequency fk(2), etc.
%}
        \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
                   ADD YOUR FUNCTION BODY HERE
        \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

end
```

Here's a description in words: your function will take in two vectors of the same length: a vector of **frequencies** and a vector of **phasors**, along with numbers for sampling frequency `fs`, duration `dur`, and start time `tstart`. Adapt your code, from the `one_cos` function in your last lab, to now process these arguments, generate a sinusoid for ***each*** frequency and phasor in the two vectors and add them together.

Hints:: The MATLAB command `length(fk)` returns the number of elements in `fk`, so we do not need an input for the number of frequencies. However, you should provide error checking to make sure the lengths of `fk` and `Xk` are the same. We are no longer constrained to 20 samples per period; in this function, `fs` defines the number of samples per second, and should be used in the construction of your time vector, tt.

More Hints (cont).): Below are some relevant help commands:):

`help exp`, `help real`, `help imag`.

In Matlab, you can check the number of arguments users give a function with "nargin." So, the following line might be helpful:

```
if nargin < 5, tstart = 0, end
```

For a more mathematical basis for adding sinusoid, please see the **Appendix** at the end of this document. If you have any further questions, please see a TA, but only after giving it some thought. Remember, you can do this!

Testing:

Now that you've completed your function, you need to test it!! You must choose the entries in the frequency vector to be integer multiples of some desired fundamental frequency. Try the following test and plot the resulting sinusoid:

```
[xx0,tt0] = syn_sin([0,100,250], [10,14*exp(-j*pi/3), 8*j],10000,0.1,0);
```

(a) Measure the period `xx0` by hand.
(b) Compare the period of `xx0` to the periods of the three individual signals that make up `xx0`.
(c) Why is the period of `xx0` longer than the two sinusoids?

Lab Part Two

2.1 Sinusoid Addition

Given the signal $x(t) = Re\{2e^{j\pi t} + 2e^{j\pi(t-1.25)} + (1-j)e^{j\pi t}\}$

(a) Plot $x(t)$ against t. Use your `syn_sin()` function and a range for $t$that will cover three periods, starting at $t = -\frac{1}{2}s$.

(b) From the plot, measure the frequency, phase, and amplitude by hand. Show annotations on the plot to show how these measurements were made and what the values are. Compare to your calculation in part (c).

(c) Use the phasor addition theorem and MATLAB to determine the magnitude and phase of $x(t)$.

2.2 Fourier Synthesis

Before we get to our interesting practical application of adding sinusoids (part 2.3), let's first look at one way to perform Fourier synthesis in Matlab (if you haven't watched the lecture videos on Fourier series yet, do so now or else none of this will make sense!)

In this project we are going to use the symbolic features of MATLAB to determine Fourier series representations for periodic waveforms, synthesize the signals, and then plot them. In general, the limits on the sum in Eq. (1) are infinite, but for our computational purposes, we must restrict them to be a finite number N obtaining the 2N + 1 term approximation:

$$x_N(t) = \sum_{k=-N}^{N} a_k e^{jk\omega_0 t}.$$

To illustrate how we'll use such an equation in Matlab, we'll use the below fouriersynth.m:

```
function xt = fouriersynth( ak, N, T0 )
%FOURIERSYNTH synthesize Fourier Series formula SYMBOLICALLY
%
% usage: xt = fouriersynth( ak, N, T0 )
%
% ak = (symbolic) formula for the Fourier Series coefficients
% N = (numeric) use the Fourier coeffs from -N to +N
% T0 = (numeric) Period in secs
% xt = (symbolic) signal synthesized
%
% example:
% syms ck k xt
% ck = sin(k)/k
% xt = fouriersynth( ck, 4, pi );
syms t k wwk
kk = -N:N;
try
ak_num = subs( ak, k, kk+((kk==0)+sign(kk))*(1e-9))
catch
error('FOURIERSYNTH: ak must use k as its variable')
end
wwk = exp(j*(2*pi*kk'/T0)*t);
xt = ak_num*wwk; %-- inner product
end
```

So, what is the above function doing? Well, the important Matlab command to notice here is **subs**; while you don't need to know exactly what all the arguments mean, you should note that it first takes in "ak" and then "k". What this means is that Matlab expects ak - the first argument of this function, to be a ***symbolic expression,*** with respect to the variable "k." (When I say "symbolic expression," all I mean is a math equation; there's no special Matlab magic here).

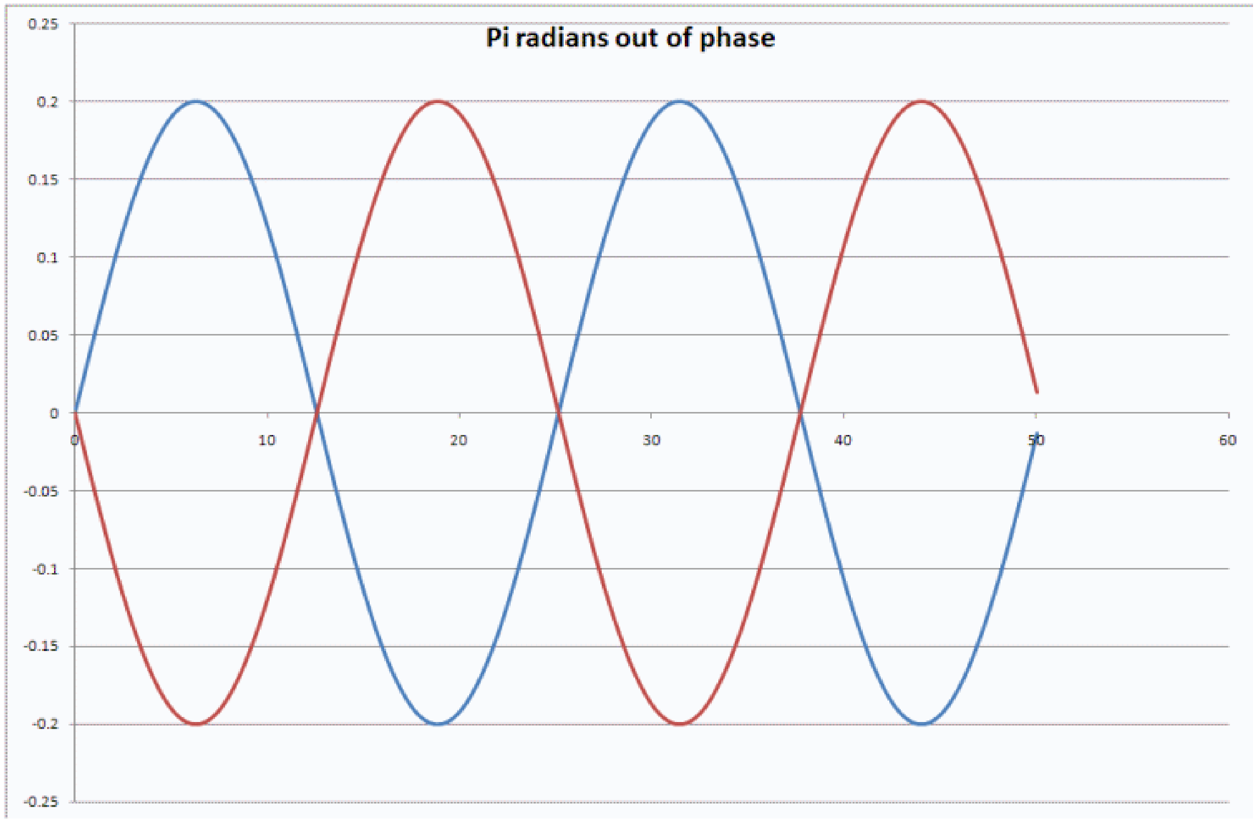
(a) Here's some code that shows an example of how this works for a periodic function whose Fourier series coefficients are given by the equation "sin(k)/k", and whose period is 5. After saving fouriersynth.m in your copy of Matlab, Run this code, and show the plot.

```
>> syms wt t ak k
>> N=12; T0=5;
>> ak = sin(k)/k;
>> wt = fouriersynth(ak, N, T0);
>> ezplot(wt, [-T0,T0]);
>> axis tight
```

(b) Now, run the code again, but this time increase the value of N. Describe how the plot is different. Can you describe how the graph would be if N was infinity?

2.3 Multipath Interference

When a signal travels through some distance, time elapses. The signal arriving at the end of a transmission line will be a *time delayed* version of the original signal. In a mobile communication system (e.g., cell phones or AM radio), signals travel through air from a transmitter to a receiver. But signals can travel through multiple paths from transmitter to receiver, and these paths can have different distances. This distance translates to a time delay, which translates to a change in the phase of the signal. Multipath interference is a destructive interference between out-of-phase reflections of the same signal.



Consider the scenario diagrammed in Fig. 1 where a vehicle traveling on the roadway receives signals from two sources: the 'direct path', directly from the transmitter, and the 'reflected path', reflected by another object such as a large building. The reflected path and direct path will have different lengths; therefore, the two signals will undergo different time delays.

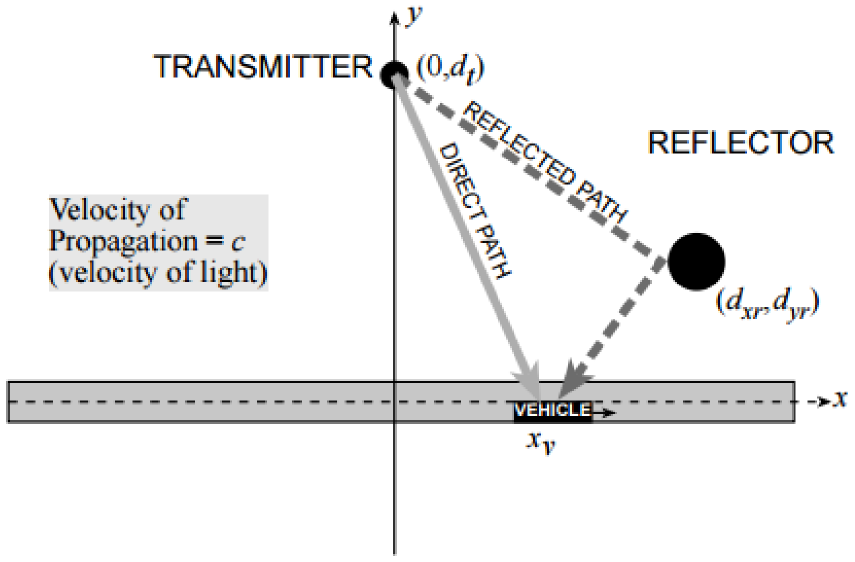


Figure 1: Scenario for multipath in mobile radio. A vehicle traveling on the roadway (to the right) receives signals from two sources: the transmitter and a reflector located at (d_{xr}, d_{yr}).

The length of the direct path is found using the Pythagorean Theorem:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

So the distance between the transmitter and the vehicle is $\sqrt{x_v^2 + d_t^2}$. Note that the transmitter location is constant, while x_v is not; the direct distance will be a function of x_v. The reflector location is also constant. The transmitter is located at $(0, 1500)$ meters and the reflector at $(100, 900)$ meters. What is the length of the reflected path as a function of x_v?

The amount of delay (in seconds) can be computed for both paths, because the time delay is the distance divided by the speed of the signal. In this case, the signal travels at the speed of light, which is $3x10^8 \frac{m}{s}$.

What is the time delay of the direct path as a function of x_v?
What is the time delay of the reflected path as a function of x_v?
Subtract the above two expressions from each other to express the difference in time delay between the two paths as a function of x_v.
Remember that time and phase are related by the following equation:

$$t = -\frac{\varphi}{\omega}$$

The difference in distance between the direct and reflected paths results in a phase difference between the two paths. Suppose the signal is a sinusoid of cyclic frequency $f_0 = 150\ MHz$. Express the phase difference between the two paths as a function of x_v.
A phase difference of π radians, or 180 degrees, results in signal cancellation, called destructive interference. Find at least one value of x_v that will result in signal cancellation.

This multipath problem can be modeled using MATLAB. The total received signal at the vehicle is the sum of two signals which are themselves delayed versions of the transmitted signal, $s(t)$.

$$r_v(t) = s(t - t_1) + s(t - t_2)$$

where $s(t)$ is the transmitted signal, which has phase 0, amplitude 1, and cyclic frequency $150\ MHz$.

Using MATLAB, use complex amplitudes to plot three periods of $r_v(t)$ when the vehicle position is $x_v = 0$ meters. Plot 3 periods and then measure the maximum amplitude.
How can a single complex addition followed by a magnitude operation be used to find the amplitude of $r_v(t)$?

We now want to make a plot of signal strength versus vehicle position. Derive an expression for complex amplitudes of each delayed sinusoid as a function of position x_v. Write a MATLAB program that will generate the time delays, create the complex amplitudes, and then add the complex amplitudes. Write this code using vectors instead of loops. You can generate a vector of all the vehicle positions, which you would use to calculate a vector of time delays. Then you would create a vector of complex amplitudes. Since you need two vectors of complex amplitudes, this process is performed twice.

Plot signal strength (amplitude) versus vehicle position over the interval from 0 meters to 300 meters. Assume that the signal strength is the peak value of the received sinusoid, $r_v(t)$.
Explain how you get the peak value from the complex sinusoid.

What are the largest and smallest values of received signal strength? Why do we get those values? Are there vehicle positions where we get complete signal cancellation, that is, where the received signal is zero? If so, determine those vehicle positions.

Is one of these vehicle positions the one you had calculated by hand above?

Lab 2 Appendix

A.1: Sinusoid Addition

We know that a sinusoid $x(t)$ can be expressed in either of the following two forms:

$$x(t) = Acos(\omega t + \varphi) = Re\{Ae^{j\varphi}e^{j\omega t}\}$$

It is intuitive that adding two sinusoids of the same frequency f_0 will return a third sinusoid of the same frequency. The Phasor Addition Rule illustrates the result:

The sum of sinusoids

$$x(t) = \sum_{k=1}^{N} A_k \cos(2\pi f_0 t + \varphi_k)$$

can be expressed as a single sinusoid with a phasor X_s that is a sum of phasors X_k:

$$X_s = \sum_{k=1}^{N} X_k = \sum_{k=1}^{N} A_k e^{j\varphi_k}$$

where

$$X_k = A_k e^{j\varphi_k}$$

and then written as one sinusoid:

$$x(t) = A_s \cos(2\pi f_0 t + \varphi_s)$$

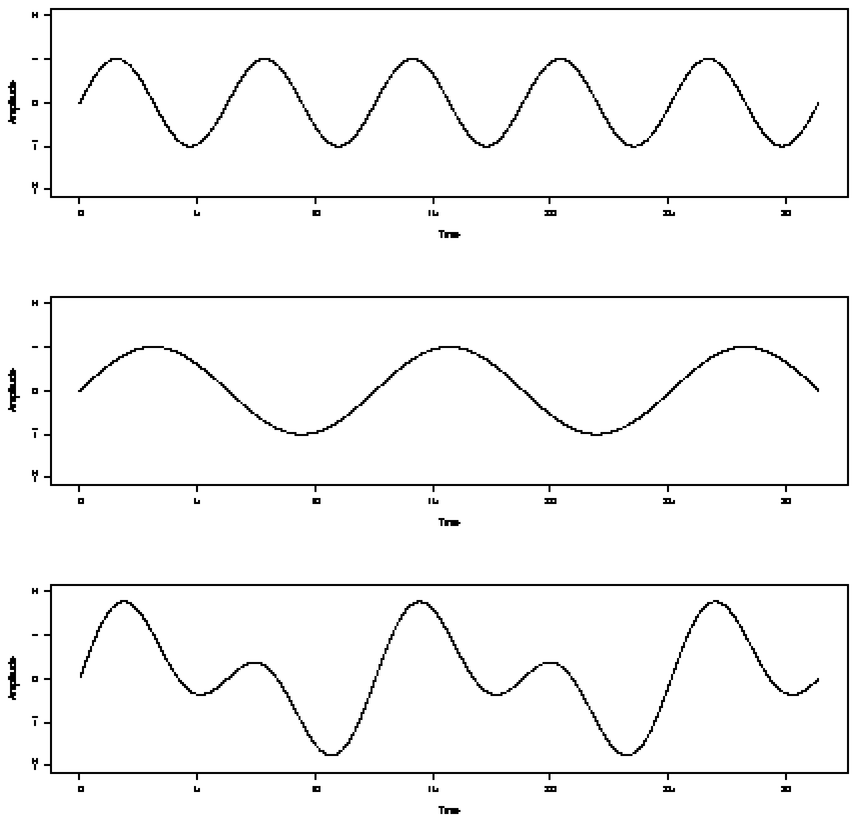
Where A_s is the amplitude of the complex phasor, and φ_s is the phase of the complex phasor.

If $x(t)$ is a sum of cosine waves whose frequencies are different, but all multiples of one base frequency f_0, then the sum is represented by:

$$x_h(t) = \sum_{k=1}^{N} A_k \cos(2\pi k f_0 t + \varphi_k) = Re\left\{\sum_{k=1}^{N} X_k e^{j2\pi k f_0 t}\right\}$$

The signal $x_h(t)$ is periodic with $T_0 = \frac{1}{f_0}$. The frequency f_0 is called the *fundamental frequency*, and T_0 is the *fundamental period*. In this section, you will write a function to generate a signal that is a summation of sinusoids.

A long time ago, French scientist and mathematician Jean Baptiste Fourier (1768–1830) proved the mathematical fact that any periodic waveform can be expressed as the sum of an infinite set of sine waves. Adding two sinusoids of the same frequency results in a sinusoid of the same frequency but with an amplitude and phase (together expressed as a '*phasor*') determined by adding the phasors of the original sinusoids. When the two sinusoids have different frequencies, the result is more complicated. We see the higher frequency sinusoid as a 'ripple' superimposed upon the lower frequency sinusoid (see figure on the following page). The fundamental frequency of the resulting signal will be the greatest common factor of the two original frequencies.



(This figure shows the sum of two sinusoids on the third line - notice how its frequency is the greatest common factor of the frequencies of its component sinusoids. Your function for part 1 of this lab should be able to create the third sinusoid given the first two.)