

Signal Processing First

Lab 02: Introduction to Complex Exponentials – Multipath

Pre-Lab and Warm-Up: You should read at least the Pre-Lab and Warm-up sections of this lab assignment and go over all exercises in the Pre-Lab section before going to your assigned lab session.

Verification: The Warm-up section of each lab must be completed **during your assigned Lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by signing on the **Instructor Verification** line. When you have completed a step that requires verification, simply demonstrate the step to the TA or instructor. Turn in the completed verification sheet to your TA when you leave the lab.

Lab Report: It is only necessary to turn in a report on Sections 5 and 6 with graphs and explanations. You are asked to **label** the axes of your plots and include a title for every plot. In order to keep track of plots, include your plot *inlined* within your report. If you are unsure about what is expected, ask the TA who will grade your report.

1 Introduction

The goal of this laboratory is to gain familiarity with complex numbers and their use in representing sinusoidal signals such as $x(t) = A \cos(\omega t + \phi)$ as complex exponentials $z(t) = Ae^{j\phi} e^{j\omega t}$. The key is to use the appropriate complex amplitude together with the real part operator as follows:

$$x(t) = A \cos(\omega t + \phi) = \Re\{Ae^{j\phi} e^{j\omega t}\}$$

2 Overview

Manipulating sinusoidal functions using complex exponentials turns trigonometric problems into simple arithmetic and algebra. In this lab, we first review the complex exponential signal and the phasor addition property needed for adding cosine waves. Then we will use MATLAB to make plots of phasor diagrams that show the vector addition needed when adding sinusoids.

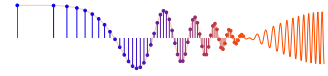
2.1 Complex Numbers in MATLAB

MATLAB can be used to compute complex-valued formulas and also to display the results as vector or “phasor” diagrams. For this purpose several new MATLAB functions have been written and are available on the *SP First CD-ROM*. Look for the “MATLAB Files” link just below the link for this lab. Make sure that this toolbox has been installed¹ by doing `help` on the new M-files: `zvect`, `zcat`, `ucplot`, `zcoords`, and `zprint`. Each of these functions can plot (or print) several complex numbers at once, when the input is formed into a vector of complex numbers. For example, the following function call will plot five vectors all on one graph:

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

¹Correct installation means that the `spfirst` directory will be on the MATLAB path. Try `help path` if you need more information.





Here are some of MATLAB's built-in complex number operators:

<code>conj</code>	Complex conjugate
<code>abs</code>	Magnitude
<code>angle</code>	Angle (or phase) in radians
<code>real</code>	Real part
<code>imag</code>	Imaginary part
<code>i, j</code>	pre-defined as $\sqrt{-1}$
<code>x = 3 + 4i</code>	<code>i</code> suffix defines imaginary constant (same for <code>j</code> suffix)
<code>exp(j*theta)</code>	Function for the complex exponential $e^{j\theta}$

Each of these functions takes a vector (or matrix) as its input argument and operates on each element of the vector. Notice that the function names `mag()` and `phase()` do not exist in MATLAB.²

When unsure about a command, use `help`.

2.2 Sinusoid Addition Using Complex Exponentials

Recall that sinusoids may be expressed as the real part of a complex exponential:

$$x(t) = A \cos(2\pi f_0 t + \phi) = \Re \{ A e^{j\phi} e^{j2\pi f_0 t} \} \quad (1)$$

The *Phasor Addition Rule* shows how to add several sinusoids:

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

assuming that each sinusoid in the sum has the *same* frequency, f_0 . This sum is difficult to simplify using trigonometric identities, but it reduces to an algebraic sum of complex numbers when solved using complex exponentials. If we represent each sinusoid with its *complex amplitude*

$$X_k = A_k e^{j\phi_k} \quad (3)$$

Then the complex amplitude of the sum is

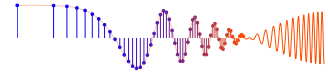
$$X_s = \sum_{k=1}^N X_k = A_s e^{j\phi_s} \quad (4)$$

Based on this complex number manipulation, the *Phasor Addition Rule* implies that the amplitude and phase of $x(t)$ in equation (2) are A_s and ϕ_s , so

$$x(t) = A_s \cos(2\pi f_0 t + \phi_s) \quad (5)$$

We see that the sum signal $x(t)$ in (2) and (5) is a single sinusoid that still has the same frequency, f_0 , and it is periodic with period $T_0 = 1/f_0$.

²In the latest release of MATLAB a function called `phase()` is defined in a rarely used toolbox; it does more or less the same thing as `angle()` but also attempts to add multiples of 2π when processing a vector.



2.3 Harmonic Sinusoids

There is an important extension where $x(t)$ is the sum of N cosine waves whose frequencies (f_k) are *different*. If we concentrate on the case where the (f_k) are all multiples of one basic frequency f_0 , i.e.,

$$f_k = k f_0 \quad (\text{HARMONIC FREQUENCIES})$$

then the sum of N cosine waves given by (2) becomes

$$x_h(t) = \sum_{k=1}^N A_k \cos(2\pi k f_0 t + \phi_k) = \Re \left\{ \sum_{k=1}^N X_k e^{j2\pi k f_0 t} \right\} \quad (6)$$

This particular signal $x_h(t)$ has the property that it is also periodic with period $T_0 = 1/f_0$, because each of the cosines in the sum repeats with period T_0 . The frequency f_0 is called the *fundamental frequency*, and T_0 is called the *fundamental period*. (Unlike the single frequency case, there is no phasor addition theorem here to combine the harmonic sinusoids.)

3 Pre-Lab

Do all the exercises in this section before attending the regularly scheduled lab section.

3.1 Complex Numbers

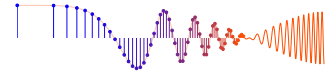
This section will test your understanding of complex numbers. Use $z_1 = 10e^{-j2\pi/3}$ and $z_2 = -5 + j5$ for all parts of this section.

- (a) Enter the complex numbers z_1 and z_2 in MATLAB. Plot them with `zvect()` and print them with `zprint()`.

When unsure about a command, use `help`.

Whenever you make a plot with `zvect()` or `zcat()`, it is helpful to provide axes for reference. An x - y axis and the unit circle can be superimposed on your `zvect()` plot by doing the following:
`hold on, zcoords, ucplot, hold off`

- (b) The function `zcat()` can be used to plot vectors in a “head-to-tail” format. Execute the statement `zcat([j, -1, -2j, 1])`; to see how `zcat()` works when its input is a vector of complex numbers.
- (c) Compute $z_1 + z_2$ and plot the sum using `zvect()`. Then use `zcat()` to plot z_1 and z_2 as 2 vectors head-to-tail, thus illustrating the vector sum. Use `hold on` to put all 3 vectors on the same plot. If you want to see the numerical value of the sum, use `zprint()` to display it.
- (d) Compute $z_1 z_2$ and plot the answer using `zvect()` to show how the angles of z_1 and z_2 determine the angle of the product. Use `zprint()` to display the result numerically.
- (e) Compute z_2/z_1 and plot the answers using `zvect()` to show how the angles of z_1 and z_2 determine the angle of the quotient. Use `zprint()` to display the result numerically.
- (f) Compute the conjugate z^* for both z_1 and z_2 and plot the results. In MATLAB, see `help conj`. Display the results numerically with `zprint`.



- (g) Compute the inverse $1/z$ for both z_1 and z_2 and plot the results. Display the results numerically with `zprint`.
- (h) Make a 2×2 subplot that displays the following four plots in one figure window: (i) z_1 and z_2 (ii) z_1^* and z_2^* on the same plot; (iii) $1/z_1$ and $1/z_2$ on the same plot; and (iv) $z_1 z_2$. Add a unit circle and x - y axis to each plot for reference.

3.2 ZDrill

There is a complex numbers drill program called:

```
zdrill
```

which uses a GUI to generate complex number problems and check your answers. Go to Appendix A demos to download and run. *Please spend some time with this drill since it is very useful in helping you to get a feel for complex arithmetic.*



CD-ROM

Appendix
A
Demos
ZDrill

3.3 Vectorization

The power of MATLAB comes from its matrix-vector syntax. In most cases, loops can be replaced with vector operations because functions such as `exp()` and `cos()` are defined for vector inputs, e.g.,

$$\cos(\mathbf{vv}) = [\cos(\mathbf{vv}(1)), \cos(\mathbf{vv}(2)), \cos(\mathbf{vv}(3)), \dots, \cos(\mathbf{vv}(N))]$$

where \mathbf{vv} is an N -element row vector. Vectorization can be used to simplify your code. If you have the following code that plots a certain signal,

```
M = 200;
for k=1:M
    x(k) = k;
    y(k) = cos( 0.001*pi*x(k)*x(k) );
end
plot( x, y, 'ro-' )
```

then you can replace the `for` loop and get the same result with 3 lines of code:

```
M = 200;
y = cos( 0.001*pi*(1:M).*(1:M) );
plot( 1:M, y, 'ro-' )
```

Use this vectorization idea to write 2 or 3 lines of code that will perform the same task as the following MATLAB script without using a `for` loop. (Note: there is a difference between the two operations `xx*xx` and `xx.*xx` when `xx` is a vector.)

```
%--- make a plot of a weird signal
N = 200;
for k=1:N
    xk(k) = k/50;
    rk(k) = sqrt( xk(k)*xk(k) + 2.25 );
    sig(k) = exp(j*2*pi*rk(k));
end
plot( xk, real(sig), 'mo-' )
```



3.4 Functions

Functions are a special type of M-file that can accept inputs (matrices and vectors) and also return outputs. The keyword `function` must appear as the first word in the ASCII file that defines the function, and the first line of the M-file defines how the function will pass input and output arguments. The file extension must be lower case “m” as in `my_func.m`. See Section B.5 in Appendix B for more discussion.

The following function has a few mistakes. Before looking at the correct one below, try to find these mistakes (there are at least three):

```
matlab mfile [xx,tt] = badcos(ff,dur)
%BADCOS Function to generate a cosine wave
% usage:
%     xx = badcos(ff,dur)
%     ff = desired frequency in Hz
%     dur = duration of the waveform in seconds
%
tt = 0:1/(100*ff):dur;    %-- gives 100 samples per period
badcos = cos(2*pi*freeq*tt);
```

The corrected function should look something like:

```
function [xx,tt] = goodcos(ff,dur)
tt = 0:1/(100*ff):dur;    %-- gives 100 samples per period
xx = cos(2*pi*ff*tt);
```

Notice the word “function” in the first line. Also, “freeq” has not been defined before being used. Finally, the function has “xx” as an output and hence “xx” should appear in the left-hand side of at least one assignment line within the function body. The function name is *not* used to hold values produced in the function.

4 Warmup: Complex Exponentials

In the Pre-Lab part of this lab, you learned how to write M-files. In this section, you will write two functions that can generate sinusoids or sums of sinusoids.

4.1 M-file to Generate a Sinusoid

Write a function that will generate a single sinusoid, $x(t) = A \cos(\omega t + \phi)$, by using four input arguments: amplitude (A), frequency (ω), phase (ϕ) and duration (`dur`). The function should return two outputs: the values of the sinusoidal signal (x) and corresponding times (t) at which the sinusoid values are known. Make sure that the function generates 20 values of the sinusoid per period. Call this function `one_cos()`. *Hint: use `goodcos()` from part (a) as a starting point.*

Demonstrate that your `one_cos()` function works by plotting the output for the following parameters: $A = 95$, $\omega = 200\pi$ rad/sec, $\phi = \pi/5$ radians, and `dur = 0.025` seconds. Be prepared to explain to the lab instructor features on the plot that indicate how the plot has the correct period and phase. What is the expected period in millisecc?

Instructor Verification (separate page)

4.2 Sinusoidal Synthesis with an M-file: Different Frequencies

Since we will generate many functions that are a “sum of sinusoids,” it will be convenient to have a function for this operation. To be general, we will allow the frequency of each component (f_k) to be different. The following expressions are equivalent if we define the complex amplitudes X_k as $X_k = A_k e^{j\phi_k}$.

$$x(t) = \Re \left\{ \sum_{k=1}^N X_k e^{j2\pi f_k t} \right\} \quad (7)$$



$$x(t) = \sum_{k=1}^N A_k \cos(2\pi f_k t + \phi_k) \quad (8)$$

4.2.1 Write the Function M-file

Write an M-file called `syn_sin.m` that will synthesize a waveform in the form of (7). Although `for` loops are rather inefficient in MATLAB, *you must write the function with one loop in this lab*. The first few statements of the M-file are the comment lines—they should look like:

```
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
% (these could be negative or positive)
% Xk = vector of complex amplitudes: Amp*e^(j*phase)
% fs = the number of samples per second for the time axis
% dur = total time duration of the signal
% tstart = starting time (default is zero, 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.
```

The MATLAB syntax `length(fk)` returns the number of elements in the vector `fk`, so we do not need a separate input argument for the number of frequencies. On the other hand, the programmer (that's you) should provide error checking to make sure that the lengths of `fk` and `Xk` are the same. See `help error`. Finally, notice that the input `fs` defines the number of samples per second for the cosine generation; in other words, we are no longer constrained to using 20 samples per period.

Include a copy of the MATLAB code with your lab report.

4.2.2 Default Inputs

You can make the last input argument(s) take on default values if you use the `nargin` operator in MATLAB. For example, `tstart` can be made optional by including the following line of code:

```
if nargin<5, tstart=0, end %--default value is zero
```

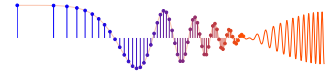
4.2.3 Testing

In order to use this M-file to synthesize harmonic waveforms, 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 result.

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

Measure the period of `xx0` by hand. Then compare the period of `xx0` to the periods of the three sinusoids that make up `xx0`, and write an explanation on the verification sheet of why the period of `xx0` is longer.

Instructor Verification (separate page)



5 Lab Exercise: Representation of Sinusoids with Complex Exponentials

In MATLAB consult `help` on `exp`, `real` and `imag`. Be aware that you can also use the DSP First function `zprint` to print the polar and rectangular forms of any vector of complex numbers.

- Generate the signal $x(t) = \Re\{2e^{j\pi t} + 2e^{j\pi(t-1.25)} + (1-j)e^{j\pi t}\}$ and make a plot versus t . Use the `syn_sin` function and take a range for t that will cover three periods starting at $t = -\frac{1}{2}$ secs. *Include the MATLAB code with your report.*
- From the plot of $x(t)$ versus t , measure the frequency, phase and amplitude of the sinusoidal signal by hand. Show annotations on the plots to indicate how these measurements were made and what the values are. Compare to the calculation in part (c).
- Use the phasor addition theorem and MATLAB to determine the magnitude and phase of $x(t)$.

6 Lab Exercise: Multipath Fading

In a mobile radio system (e.g., cell phones or AM radio), there is one type of degradation that is a common problem. This is the case of *multipath fading* caused by reflections of the radio waves which interfere destructively at some locations. Consider the scenario diagrammed in Fig. 1 where a vehicle traveling on the roadway receives signals from two sources: directly from the transmitter and reflections from another object such as a large building. This multipath problem can be modeled easily with sinusoids. The total received signal at the vehicle is the sum of two signals which

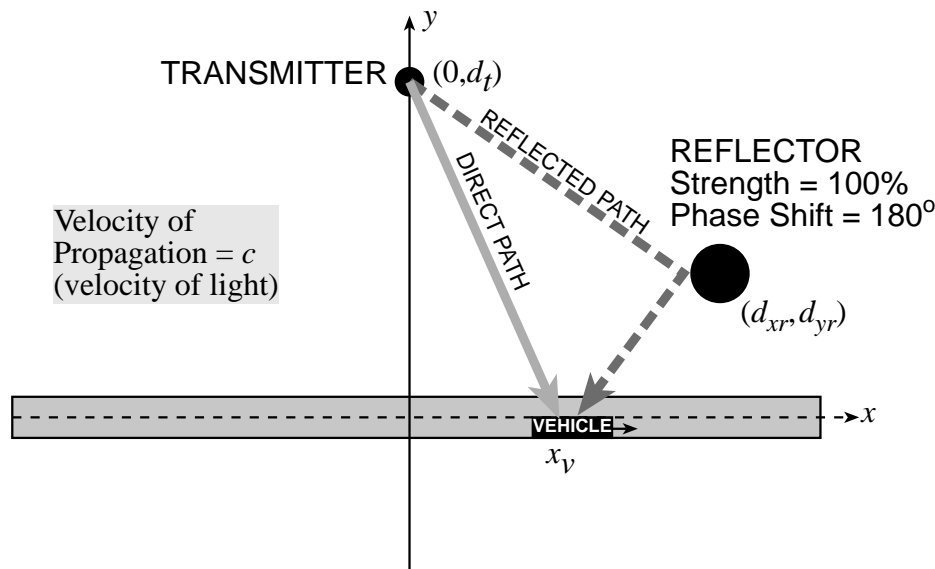


Figure 1: Scenario for multipath in mobile radio. A vehicle traveling on the roadway (to the right) receives signals from two sources: the transmitter and a reflector located at (d_{xr}, d_{yr}) .

are themselves delayed versions of the transmitted signal, $s(t)$.

- The amount of the delay (in seconds) can be computed for both propagation paths. First of all, consider the direct path. The time delay is the distance divided by the speed of light (3×10^8 m/s). Write a mathematical expression for the time delay in terms of the vehicle position x_v and the transmitter location $(0, d_t)$. Call this delay time t_1 and express it as a function of x_v , i.e., $t_1(x_v)$.
- Now write a mathematical formula for the time delay of the signal that travels the reflected path from the transmitter at $(0, d_t)$ to the reflector at (d_{xr}, d_{yr}) and then to the vehicle at $(x_v, 0)$. Call this delay time t_2 and make sure that you also express it as a function of x_v , i.e., $t_2(x_v)$. In this case, you must add together two delays: transmitter to reflector and then reflector to vehicle.



- (c) The received signal at the vehicle, $r_v(t)$, is the sum of the two delayed copies of the transmitter signal

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

where $s(\cdot)$ is the transmitted signal, and the minus sign accounts for fact that the reflection reverses the phase of the reflected signal.³

Assume that the source signal $s(t)$ is a zero-phase unit-amplitude sinusoid at frequency $f_0 = 150$ MHz; and also assume that the transmitter is located at $(0, 1500)$ meters and the reflector at $(100, 900)$ meters. Then the received signal is the sum of two sinusoids. Make a plot of $r_v(t)$ when the vehicle position is $x_v = 0$ meters. Plot 3 periods of $r_v(t)$ and then measure its maximum amplitude.

- (d) Repeat the work in part (c), but use the complex amplitudes instead. Explain how a single complex addition, followed by a magnitude operation can be used to find the amplitude of $r_v(t)$.
- (e) Our aim in the rest of this lab is to make a plot of signal strength versus vehicle position (x_v). One approach would be to repeat the process in part (c) for every position, i.e., generate the resultant sinusoid and measure its amplitude. However, that would be a huge waste of computation. Instead, the complex amplitude approach in part (d) is a much more efficient method.

Derive a mathematical expression for complex amplitudes of each delayed sinusoid as a function of position x_v . Then write a MATLAB program that will generate the time delays, form the complex amplitudes, and then add the complex amplitudes. Have the MATLAB program loop through the entire set of vehicle positions specified in part (f) below. Include this MATLAB code in your report and explain how it works.

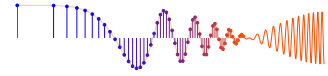
Vectorization: It is likely that your previous programming skills would lead you to write a loop to do this implementation. The loop would run over all possible values of x_v , and would add the two complex amplitudes calculated at each x_v .

However, there is a *much more efficient way in MATLAB*, if you think in terms of vectors (which are really lists of numbers). In the vector strategy, you would make a vector containing all the vehicle positions; then do the distance and time delay calculations to generate a vector of time delays; next, a vector of complex amplitudes would be formed. In each of these calculations, only one line of code is needed and *no loops*. You need two vectors of complex amplitudes (one for the direct path and the other for the reflected path), so the whole process is done twice and then you can perform a vector add of the two complex amplitudes.

- (f) Utilize the MATLAB program from the previous part to generate a plot of signal strength versus vehicle position, x_v , over the interval from 0 meters to +300 meters.⁴ Assume that *signal strength* is defined to be the peak value of the received sinusoid, $r_v(t)$. State how you get the peak value from the complex amplitude.
- (g) Explain your results from the previous part. In particular, 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? If so, determine those vehicle positions.

³For simplicity we are ignoring propagation losses: When a radio signal propagates over a distance R , its amplitude will be reduced by an amount that is inversely proportional to R^2 .

⁴MATLAB works on vectors so you will have to produce a vector of positions starting at 0 and ending at +300 with a spacing that is small enough to capture all the variations in signal strength.



Lab 02a

INSTRUCTOR VERIFICATION SHEET

For each verification, be prepared to explain your answer and respond to other related questions that the lab TA's or professors might ask. Turn this page in at the end of your lab period.

Name: _____

Date of Lab: _____

Part 4.1 Demonstrate that your `one_cos` function is correct by plotting a sinusoidal signal with the given parameters. Use the space below to calculate the period of the sinusoid.

Verified: _____

Date/Time: _____

Part 4.2.3 Show that your `syn_sin.m` function is correct by running the test in Section 4.2.3 and plotting the result. Measure the period(s) and explain why the period of `xx0` is longer than the periods of the signals used to form `xx0`. Write your explanations in the space below.

Verified: _____

Date/Time: _____