**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**EE-232 Signals and Systems**

**Lab #4 Introduction to Complex Exponentials**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. no.** | **Report Marks / 10** | **Viva Marks / 5** | **Total/15** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Lab4: Introduction to Complex Exponentials**

**Objectives**

The goal of this laboratory is to gain familiarity with complex numbers and their use in representing sinusoidal signals such as  as complex exponentials. The key is to use the appropriate complex amplitude together with the real part operator as follows:



* How to work with Complex NumbersinMATLAB
* Familiarization with Matalb Function and commands for Complex Exponentials
* Sinusoid Addition Using Complex Exponentials

**Lab Report Instructions**

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* MATLAB codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# Introduction to Complex Exponentials

## Pre-Lab

### Introduction

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.

### 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 this toolbox has been installed by using 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:



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

*conj*  Complex conjugate

*abs*  Magnitude

*angle* Angle (or phase) in radians

*real* Real part

*imag* Imaginary part

*i,j* pre-defined as 

x = 3 + 4i i suffix defines imaginary constant (same for j suffix)

exp(j\*theta) Function for the complex exponential 

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.

### Sinusoid Addition Using Complex Exponentials

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



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



Assuming that each sinusoid in the sum has the *same* frequency, *f0*. 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*



Then the complex amplitude of the sum is



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



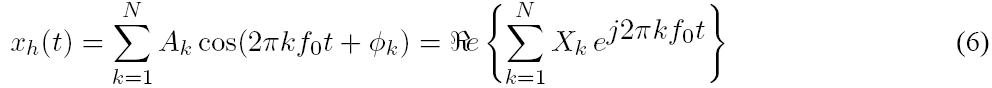
We see that the sum signal x(t) in (2) and (5) is a single sinusoid that still has the same frequency, *f0*, and it is periodic with period *T0* = 1/*f0*.

### Harmonic Sinusoids

There is an important extension where x(t) is the sum of N cosine waves whose frequencies (fk) are *different.* If we concentrate on the case where the fk are all multiples of one basic frequency *f0*.

fk = k f0 (HARMONIC FREQUENCIES),

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



This particular signal *xh*(t) has the property that it is also periodic with period T0 = 1/f0, because each of the cosines in the sum repeats with period T0. The frequency f0 is called the *fundamental frequency*, and T0 is called the *fundamental period*. (Unlike the single frequency case, there is no phasor addition theorem here to combine the harmonic sinusoids). Do all the exercises in this section before attending the regularly scheduled lab section.

### Complex Numbers

This section will test your understanding of complex numbers. Use for

all parts of this section.

**(a)** Enter the complex numbers z1 and z2 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 z1 + z2 and plot the sum using zvect(). Then use zcat() to plot z1 and z2 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 z1z2 and plot the answer using zvect() to show how the angles of z1 and z2 determine the angle of the product. Use zprint() to display the result numerically.

**(e)** Compute z2/z1 and plot the answers using zvect() to show how the angles of z1 and z2 determine the angle of the quotient. Use zprint() to display the result numerically.

**(f)** Compute the conjugate z\* for both z1 and z2 and plot the results. In MATLAB, see help conj. Display the results numerically with zprint.

**(g)** Compute the inverse 1/z for both z1 and z2 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) z1 and z2 (ii) z1\* and z2\* on the same plot; (iii) 1/z1 and 1/z2 on the same plot; and (iv) z1z2.

### ZDrill

There is a complex numbers drill program called:

*zdrill*

This 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.*

### 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(vv) = [cos(vv(1)), cos(vv(2)), cos(vv(3)), ... cos(vv(N))]*

Where 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-’)*

### 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. 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*

*% 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.

## Lab Tasks

### 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.

### Lab Task 1:

#### M-file to generate a Sinusoid

Write a function that will generate a single sinusoid, , by using four input arguments: amplitude (A), frequency (ω), phase (*ɸ*) 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 par (a) as a starting point.* Demonstrate that your one\_cos()function works by plotting the output for the following parameters: A = 95, ω = 200  *ɸ* rad/sec, *ɸ* = π/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 millisec?

#### 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 (fk) to be different. The following expressions are equivalent if we define the complex amplitudes: .





### Lab task 2:

#### 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 but youmust 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\*eˆ(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 define 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.*

#### 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*

#### Testing

In order to use this M-file to synthesize harmonic waveforms, you must choose the entries in the frequency vector tobbe 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.

### Lab Task 3:

#### Representation of Sinusoids with Complex Exponentials

In MATLAB consult help on exp, real and imag.

(a) Generate the signal 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 = −0.5 secs. *Include the* MATLAB *code with your report.*

http://users.ece.gatech.edu/mcclella/SPFirst/Updates/SPFirstMATLAB.html