Lab S-0: Complex Exponentials – Adding Sinusoids

Pre-Lab: Read the Pre-Lab and do all the exercises in the Pre-Lab section ***prior to attending lab.***

Verification: The Exercise section of each lab should be completed **during your assigned Lab time** and the steps marked *Instructor Verification* 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, demonstrate the step to your instructor. Turn in the completed verification sheet before you leave the lab.

Lab Homework Questions: The Lab-Homework Sheet has a few lab related questions that can be answered at your own pace. The completed Lab-HW sheet should be turned in at the beginning of the next lab.

1 Introduction and Overview

The goal of this laboratory is to gain familiarity with complex numbers and their use in representing sinusoidal signals such as $x(t) = A\cos(\omega t + \varphi)$ as complex exponentials $z(t) = Ae^{j\varphi}e^{j\omega t}$. The key is to use the complex amplitude, $X = Ae^{j\varphi}$, and then apply the real part operator to Euler's formula to obtain

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

Furthermore, manipulating sinusoidal functions using complex exponentials turns trigonometric problems into simple arithmetic and algebra.

1.1 Complex Numbers in MATLAB

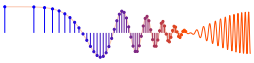
MATLAB can be used to compute complex-valued formulas and also to display the results as vector or "phasor" diagrams. For this purpose several new MATLAB functions have been written and are available on the *DSP First companion website* in the *DSP First* MATLAB *Toolbox*. Make sure that this toolbox has been installed[1] 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 and observe that it will plot five vectors all on one graph:

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

Here are some of MATLAB's 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 $\sqrt{-1}$
`x = 3 + 4i`	`i` suffix defines imaginary constant (same for `j` suffix)
`exp(j*theta)`	Function for the complex exponential $e^{j\theta}$

[1] Correct installation means that the `dspfirst` directory will be on the MATLAB path. Follow the instructions on the companion website for setting the path. Try `help path` if you need more information.

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]

Finally, there is a complex numbers drill program called `zdrill`, which uses a GUI to generate complex number problems and check your answers. *Please spend some time with this drill since it is very useful in helping you to get a feel for complex arithmetic.*

When unsure about a command, use `help` or `doc`.

1.2 Sinusoid Addition Using Complex Exponentials

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

$$x(t) = A\cos(2\pi ft + \varphi) = \Re\left\{Ae^{j\varphi}e^{j2\pi ft}\right\} \tag{1}$$

The *Phasor Addition Rule* presented in Chapter 2 of the text shows how to add several sinusoids:

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

assuming that each sinusoid in the sum has the *same* frequency, f. 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*

$$X_k = A_k e^{j\varphi_k} \tag{3}$$

Then the complex amplitude of the sum is

$$X_s = \sum_{k=1}^{N} X_k = A_s e^{j\varphi_s} \tag{4}$$

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

$$x(t) = A_s\cos(2\pi ft + \varphi_s) \tag{5}$$

Notice that the sum signal $x(t)$ in (2) and (5) is a single sinusoid that still has the *same* frequency, f.

1.3 Harmonic Sinusoids

There is an important case where $x(t)$ is the sum of N cosine waves whose frequencies (f_k) are *different,* but the frequencies (f_k) are all multiples of one basic frequency F_0, i.e.,

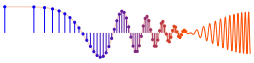
$$f_k = kF_0 \qquad \text{(HARMONIC FREQUENCIES)}$$

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

$$x_h(t) = \sum_{k=1}^{N} A_k\cos(2\pi kF_0\,t + \varphi_k) = \Re\left\{\sum_{k=1}^{N} X_k\, e^{j2\pi kF_0\,t}\right\} \tag{6}$$

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. When T_0 is the shortest such repetition period, 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 to combine the harmonic sinusoids.

[2] In the latest release of MATLAB a function called `phase()` is defined in a seldom 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 Pre-Lab

Please do the exercises in this section prior to meeting with your instructor at lab time.

2.1 Complex Numbers

This section will test your understanding of complex numbers when plotted as vectors. Use $z_1 = 2e^{j\pi/4}$ and $z_2 = -\sqrt{3} + j$ for all parts of this section.

(a) Enter the complex numbers z_1 and z_2 in MATLAB, then plot them with `zvect()`, and also print them with `zprint()`.

When unsure about a command, use `help` or `doc`.

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) Compute the conjugate z^* and the inverse $1/z$ for both z_1 and z_2 and plot the results as vectors. In MATLAB, see `help conj`. Display the results numerically with `zprint`.

(c) The function `zcat()` can be used to plot vectors in a "head-to-tail" format. Execute the statement `zcat([1+j,-2+j,1-2j]);` to see how `zcat()` works when its input is a vector of complex numbers.

(d) Compute $z_1 + z_2$ and plot the sum using `zvect()`. Then use `zcat()` to plot z_1 and z_2 as two vectors head-to-tail, thus illustrating the vector sum. Use `hold on` to put all three vectors on the same plot. If you want to see the numerical value of the sum, use `zprint()` to display it.

(e) Compute $z_1 z_2$ and z_2/z_1 and plot the answers using `zvect()` to show how the angles of z_1 and z_2 determine the angles of the product and quotient. Use `zprint()` to display the results numerically.

(f) Make a 2×2 subplot that displays four plots in one window, similar to the four operations done previously: (i) z_1, z_2, and the sum $z_1 + z_2$ on a single plot; (ii) z_2 and z_2^* on the same plot; (iii) z_1 and $1/z_1$ on the same plot; and (iv) $z_1 z_2$. Add a unit circle and x-y axis to each plot for reference.

2.2 Z-Drill

Work a few problems generated by the complex number drill program. To start the program simply type `zdrill`. Use the buttons on the graphical user interface (GUI) to produce different problems.

2.3 Publishing MATLAB Code

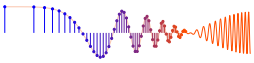
When documenting MATLAB code for lab reports, the "publish" feature in MATLAB provide an easy way to produce an `.html` file directly from an M-file. This publish feature is provided with one of the tabs in MATLAB's edit window. The following help describes the process:

`http://www.mathworks.com/help/matlab/matlab_prog/publishing-matlab-code.html`

The basic idea is to write comments according to some simple formatting rules. The following example from the MATLAB documentation illustrates the process with an example that comes from DSP, i.e., summation of harmonic sinusoids in a Fourier series.

```
edit(fullfile(matlabroot,'help','techdoc','matlab_env','examples','fourier_demo2.m'))
```

- Use the command above to open the M-file `fourier_demo2.m` in the MATLAB editor

- Run the `fourier_demo2.m` M-file to see the plots that it creates.
- Publish `fourier_demo2.m` to create `fourier_demo2.html`, and then open `fourier_demo2.html` in a web browser and view the plots that show the sums of harmonic sinusoids.

2.4 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 the signal in the vector `yy`,

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

then the `for` loop can be replaced with one line to get the same result with three lines of code:

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

Run these two code blocks to see that they give identical results. An important point to keep in mind is that the vectorized version runs much faster, which is noticeable for larger values of `M`. Use MATLAB's "publish" feature to produce an `.html` file directly from the code snippets.

2.5 Functions

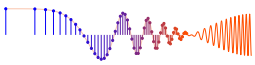
Functions are a special type of M-file that can accept inputs (matrices and vectors) and also return outputs. The keyword `function` must appear as the first word in the M-file that defines the function, and the first line of the M-file defines how the function will pass input and output arguments. The file extension must be lower case "m" as in `my_func.m`. See Section B-6 in Appendix B of the text for more discussion.

The following function has several mistakes (there are at least four). Before looking at the correct one below, try to find all the mistake(s):

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

The corrected function (without the comment lines) should look something like:

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

Notice the word `function` at the beginning of the first line. Also, the exponential needs to have an imaginary exponent `2i`, and the variable `freeq` has not been defined before being used. Finally, the function has `xx` as an output, so the variable `xx` must appear on the left-hand side of at least one assignment line within the function body. In other words, the function name is *not* used to hold values produced in the function.

3 Lab Exercise: Complex Exponentials

In the Pre-Lab part of this lab, you learned how to write function M-files. In this section, you will write two functions that can generate sinusoids, or sums of sinusoids. Use MATLAB's publish capability to put the code and plots into an `.html` file and use a web browser when doing the Instructor Verifications.

3.1 Vectorization

Use the vectorization idea to write two or three lines of code that will perform the same task as the following MATLAB script without using a `for` loop.

```
%--- make a plot of a weird signal
N = 200;
for k=1:N
   xk(k) = k/60;
   rk(k) = sqrt( xk(k)*xk(k) - 1 );
   sig(k) = exp(j*2*pi*rk(k));
end
plot( xk, real(sig), 'mo-', xk, imag(sig), 'go-' )
```

Note: there is a difference between the two multiply operations `rr*rr` and `rr.*rr` when `rr` is a vector.

Instructor Verification (separate page)

3.2 M-file to Generate One Sinusoid

Write a function that will generate a **single** sinusoid, $x(t) = A\cos(2\pi ft + \varphi)$, by using input arguments for frequency (f), complex amplitude ($X = Ae^{j\varphi}$), duration `(dur)` and starting time `(tstart)`. The function should return two output vectors: the values of the sinusoidal signal (x) and corresponding times (t) at which the sinusoid values are known. Make sure that the function generates exactly 32 values of the sinusoid per period. Call this function `onecos()`. *Hint: use* `goodcos()` *from the Pre-Lab as a starting point.* Plot the result from the following call to test your function.

```
[xx0,tt0] = onecos([2], [5*exp(j*pi/4)], 2, -1);      % (freq in Hz)
```

Use MATLAB's publish capability to put the plots into an `.html` file for the Instructor Verification.

Instructor Verification (separate page)

3.3 Sinusoidal Synthesis with an M-file: Different Frequencies

Since we often generate signals that are a "sum of sinusoids," it is convenient to have a MATLAB function for this operation. To be general, we want to allow the frequency of each component (f_k) to be different. The

following expressions are equivalent if we define the complex amplitude X_k as $X_k = A_k e^{j\varphi_k}$.

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

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

3.3.1 Write the Sum of Sinusoids M-file

Write an M-file called `add_sines.m` that will synthesize a waveform in the form of (7) using X_k defined in (3). Although `for` loops are rather inefficient in MATLAB, *you must write the function with one outer loop in this lab.* The inner loop should be vectorized. The first few statements of the M-file are the comment lines—they should look like:

```
function    [xx,tt] = add_sines(freqs, Camps, dur, tstart)
%ADD_SINES  Synthesize a signal from sum of complex exponentials
%  usage:
%    [xx,tt] = add_sines(freqs, Camps, dur, tstart)
%   freqs = vector of frequencies   (usually none are negative)
%   Camps = vector of COMPLEX amplitudes
%     dur = total time duration of the signal
%  tstart = starting time
%      xx = vector of sinusoidal values
%      tt = vector of times, for the time axis
%
%    Note: freqs and Camps must be the same length.
%          Camps(1) corresponds to frequency freqs(1),
%          Camps(2) corresponds to frequency freqs(2), etc.
%    The tt vector should be generated with a small time increment that
%         creates 32 samples for the shortest period, i.e., use the period
%         corresponding to the highest frequency in the freqs vector.
```

The MATLAB syntax `length(freqs)` returns the number of elements in the vector `freqs`, so we do not need a separate input argument for the number of frequencies. On the other hand, it is good programming practice to provide error checking to make sure that the lengths of the vectors `freqs` and `Camps` are the same. See `help error` for generating an error condition in MATLAB.

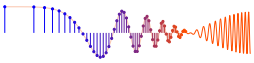
3.3.2 Testing

In order to verify that this M-file can synthesize sinusoids, try the following test and include the plot in your published `.html` file.

```
[xx0,tt0] = add_sines([12,9,0], [exp(j*pi/4),2i,-4], 1, -0.5); %-Period = ?
```

Measure the period of `xx0` on the plot, and then the DC value (which is also the average value). Write an explanation on the verification sheet of why the measured values are correct. Notice that when this M-file is used to synthesize harmonic waveforms, you must choose the entries in the frequency vector to be integer multiples of a fundamental frequency. What is the fundamental frequency for `xx0` ?

Instructor Verification (separate page)

Lab: Adding Sinusoids
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your instructor before the end of your scheduled Lab time.

Name: _____ UserID: _____ Date: _____

Part 3.1 Replace the inner `for` loop with only 1 or 2 lines of vectorized MATLAB code. Write the MATLAB code in the space below:

Verified:_____ Date/Time:_____

Part 3.2 and other places: Use MATLAB's publish capability to create and save the plots when doing the Instructor Verifications. Use a browser to show the `.html` document to your TA.

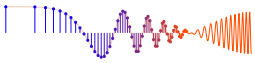
Verified:_____ Date/Time:_____

Part 3.3.2 Show your completed `add_sines.m` function to the instructor.

Verified:_____ Date/Time:_____

Part 3.3.2 Show that your `add_sines.m` function is correct by running the test in Section 3.3.2 and plotting the result. Measure the DC value and the period of `xx0` and explain why the measured values are correct. Determine the fundamental frequency for `xx0`. Write your explanations in the space below.

Verified:_____ Date/Time:_____

Lab: Adding Sinusoids
LAB HOMEWORK QUESTION

Turn this page in to your instructor at the very beginning of your next scheduled Lab time.

Name: _____ UserID: _____ Date: _____

In Chapter 3 and Appendix C the Fourier Series is studied. One result is to show that a square wave can be synthesized with a sum of sinusoids. With a finite number of sinusoids the synthesis is approximate. Figure 3-16 in the text shows an example of a square wave being approximated.

• Use your `add_sines.m` function to create a sum of sinusoids with the following complex amplitudes

$$X_k = A_k e^{j\varphi_k} = \begin{cases} j/k & k \text{ odd, and } |k| \le 9 \\ 0 & \text{elsewhere} \end{cases}$$

Make the fundamental frequency f_0 equal to 10 Hz, and plot 0.4 s of the signal over the time range from −0.2 s to 0.2 s, i.e., four periods.

• Show the code and the plot, and verify that the plot approximates a square wave.

• A square wave alternates between two values. Make a sketch of the square wave being approximated, showing those two values