GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

EE 2200 Winter 1999
Lab #2: Introduction to Complex Exponentials

Date: week of 18 Jan 1999

A short Lab Quiz will be held during the first 10-15 minutes of lab. It will cover basic MATLAB issues.

Lab is held in College of Computing Building, room 309.

This is *the official* Lab #2 description; it is *similar* to the one in Appendix C.2 of the text.

The Warm-up section of each lab must be completed in Lab and the steps marked *Instructor Verification* must also be signed off **during your scheduled lab time**. One of the laboratory instructors must verify the appropriate steps by initialing on the **Instructor Verification** line. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the instructor.

Lab Report: It is only necessary to turn in Section 4 with explanations as this week's lab report. Staple the **Instructor Verification** sheet to the end of your lab report as evidence that the appropriate steps were witnessed by the instructor.

The report will **due during the week of 25-Jan at the start of your lab**.

---

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 real part operator is the key:

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

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 combining 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 *DSP First CD-ROM*. 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 will plot five vectors all on one graph:


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

[1] Correct installation means that the `dspfirst` directory will be on the MATLAB path. Try `help path` is you need more information.

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
`exp(j*theta)`	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.

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`.

2.2 Sinusoid Addition Using Complex Exponentials

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

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

The *Phasor Addition Rule* presented in Section 2.6.2 of the text (page 33) shows how to add several sinusoids:

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

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*

$$Z_k = A_k e^{j\phi_k} \tag{3}$$

Then the complex amplitude of the sum is

$$Z_s = \sum_{k=1}^{N} Z_k = A_s e^{j\phi_s} \tag{4}$$

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

$$x(t) = A_s \cos(2\pi f_0 t + \phi_s) \tag{5}$$

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 $T_0 = 1/f_0$.

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.,

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

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

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

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. The frequency f_0 is called the *fundamental frequency*, and T_0 is called the *fundamental period*.

3 Warm-up

The instructor verification sheet is at the end of this lab.

3.1 Complex Numbers

To exercise your understanding of complex numbers, do the following:

(a) Define $z_1 = -2 - j2$ and $z_2 = 3 - j\sqrt{3}$. Enter these complex numbers in MATLAB and plot them with `zvect()`, and print them with `zprint()`.

When unsure about a command, use `help`.

(b) 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 plot by doing the following:
`hold on, zcoords, ucplot, hold off`

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

(d) 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 it works.

(e) Compute $z_1 + z_2$ and plot the sum using `zcat()` to show z_1, z_2, and the sum as 3 vectors head-to-tail. Use `zprint()` to display the sum numerically.

(f) 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.

(g) Make a 2×2 subplot that displays four operations: (i) z_1, z_2, and $z_1 + z_2$ via `zcat`; (ii) z_2 and z_2^* on the same plot; (iii) z_1 and $1/z_1$ on the same plot; and (iv) $z_1 z_2$.

Instructor Verification (separate page)

(h) Work a few problems on 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.

3.2 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. See Section B.5 in Appendix B for more discussion.

The following exercises are designed to help you write functions in MATLAB.

(a) Find the mistake(s) in the following function:

```
 [xx,tt] = cosgen(f,dur)
%COSGEN  Function to generate a cosine wave
%  usage:
%        xx = cosgen(f,dur)
%         f = desired frequency
%       dur = duration of the waveform in seconds
%
tt = 0:1/(20*f):dur;  % gives 20 samples per period
cosgen = cos(2*pi*freq*tt);
```

(b) Explain what the statement `tq1 = linspace(0,1.0,20)` produces. Do you get the same thing using `tq2 = 0:(1/20):1.0`?

When unsure about a command, use `help`.

(c) Write a function that will generate a single sinusoid by using four arguments: amplitude (A), frequency (ω), phase (ϕ) and duration `(dur)`. The function should return two outputs: the values of the sinusoidal signal and corresponding times at which the sinusoid values are known. Make sure that the function generates 20 values of the sinusoid per period.
Call this function `makesine`. *Hint: use part (a) as a starting point.*

Demonstrate that your `makesine()` function works by plotting the output for the following parameters: $A = 3.2$, $\omega = 880\pi$ rad/sec, $\phi = -0.5\pi$ radians, and `dur = 0.01` seconds. What is the length of the period in milliseconds?

Instructor Verification (separate page)

4 Exercises: Complex Exponentials

4.1 Sinusoidal Synthesis with an M-file

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 amplitude Z_k as $Z_k = A_k e^{j\phi_k}$.

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

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

4.1.1 Write the Function M-file

Write an M-file called `makecos` 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 part.* The first few statements of the M-file are the comment lines—they should look like:

```
function        [xx,tt] = makecos(ff, ZZ, fs, dur)
%MAKECOS   Function to synthesize a sum of cosine waves
%  usage:
%      xx = makecos(ff, Z, fs, dur)
%      ff = vector of frequencies
%              (these could be negative or positive)
%      ZZ = vector of complex exponentials: Amp*e^(j*phase)
%      fs = the number of samples per second for the time axis
%     dur = total time duration of signal
%      xx = vector of sinusoidal values
%      tt = vector of times, for the time axis
%
%    Note: ff and ZZ must be the same length.
%             ZZ(1) corresponds to frequency ff(1),
%             ZZ(2) corresponds to frequency ff(2), etc.
```

The MATLAB syntax `length(ff)` returns the number of elements in the vector `ff`, 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 `ff` and `ZZ` 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 using 20 samples per period.
Include a copy of the MATLAB *code with your lab report.*

4.1.2 Testing

In order to use this M-file to synthesize periodic (harmonic) waveforms, you would simply choose the entries in the frequency vector to be integer multiples of the desired fundamental frequency. Try the following tests and plot the results (if the function does not return an error).
Make a subplot with all the working test cases. Measure the period of `xx4` (by hand).

```
xx1 = makecos([10],   [j],    150, 0.3);
xx2 = makecos([5],   [1,j],   150, 0.3);
xx3 = makecos([10,10,10], [j,-1,j], 150, 0.3);
xx4 = makecos([4,6,8], [j,1,0.3*j], 20, 1);   %--Notice the Period
```

4.1.3 Testing with Sounds

Summation: Use your `makecos()` function to generate the **sum** of three sinusoids with frequencies that correspond to notes in a piano chord: $f_1 = 349$ Hz, $f_2 = 440$ Hz, and $f_3 = 523$ Hz. If these 3 tones are played at the same time, the result should be the F-major chord. Make all the amplitudes equal, pick the phases to be random, and make the length of the signal 1.2 seconds. Use a sampling rate of 11025 Hz. Play the signal with `soundsc()` to verify that it makes a pleasant sounding chord.

Concatenation: Utilize your `makecos()` function to generate a **sequence** of tones with frequencies at $f_1 = 349$ Hz, $f_2 = 440$ Hz, and $f_3 = 523$ Hz. These notes should be played individually, one after another. Make the duration of each note equal to 0.6 seconds. Note: in MATLAB row vectors can be concatenated by writing `xxx = [xx1,xx2,xx3]`. In this case, determine the total length of the `xxx` vector.
For your lab report, include the MATLAB *code used to generate the sounds in both cases.*

4.2 Representation of Sinusoids with Complex Exponentials

In MATLAB consult `help` on `exp`, `real` and `imag`.

(a) Generate the signal $x(t) = Ae^{j(\omega_0 t + \phi)}$ for $A = 3$, $\phi = 0.1\pi$, and $\omega_0 = 2\pi(1250)$. Take a range for t that will cover 2 or 3 periods. *Include the* MATLAB *code with your report.*

(b) Plot the real part versus t and the imaginary part versus t. Use `subplot(2,1,i)` to put both plots in the same window. *Include the* MATLAB *plot with your report.*

(c) Measure the frequency, phase and amplitude of the real and imaginary parts by hand. Show annotations on the plots to indicate how these measurements were made and what the values are.

4.3 Verify Addition of Sinusoids Using Complex Exponentials

Use the `makecos` function to generate three sinusoids with the following amplitudes and phases:

$$\begin{aligned} x_1(t) &= \tfrac{1}{2}\cos(2\pi(7)t + 0.1\pi) \\ x_2(t) &= \sqrt{3}\cos(2\pi(7)t - 0.5\pi) \\ x_3(t) &= \tfrac{1}{2}\cos(2\pi(7)t - 0.1\pi) \end{aligned}$$

(a) Make a plot of all three signals over a range of t that will exhibit approximately 4 cycles. Make sure the plot includes negative time so that the phase at $t = 0$ can be measured. Use `subplot(3,1,i)` to make a 3-panel subplot that puts all of these plots on the same page. In addition, try the command `axis tight` to get the maximum usage of the plotting area. *In order to get a smooth plot make sure that your have at least 20 samples per period of the wave.*[2]

(b) Verify that each sinusoidal signal in part (a) has the correct maximum amplitude, and also verify that the phase of all three signals is correct by measuring a "peak time" and showing how the phase is related to the "peak time."

(c) Use the `makecos` function to create the sum sinusoid via: $x_0(t) = x_1(t) + x_2(t) + x_3(t)$. Make a plot of $x_0(t)$ over the same range of time as used in the previous plots.

Measure the magnitude and phase of $x_0(t)$ directly from the plot. In your lab report, include this plot with sufficient annotation to show how the magnitude and phase were measured.

(d) Now do some complex arithmetic; create the complex amplitudes corresponding to the sinusoids $x_i(t)$:

$$z_i = A_i e^{j\phi_i} \qquad\qquad i = 0, 1, 2, 3$$

Give the numerical values of z_i in polar *and* Cartesian form. Relate the magnitude and phase of z_0 to the plot of $x_0(t)$.

(e) Verify that $z_0 = z_1 + z_2 + z_3$. Show a plot of these four complex numbers as vectors. Use the MATLAB functions `zvect`, `zcat` and `zprint` to make the "head-to-tail" plot of the sum.

[2] When we study sampling in Chapter 4, then this requirement of 20 samples per period will be recognized as oversampling.

Lab #2
EE-2200
Winter-1999
INSTRUCTOR VERIFICATION SHEET

Staple this page to the end of your Lab Report.

Name: _____ Date of Lab: _____

Part 3.1(g) Show 2×2 subplot of various complex number operations:

Verified:_____ Date/Time:_____

Part 3.2(c)
Write the function `makesine()` and demonstrate that it works by plotting the output for the following parameters: $A = 3.2$, $\omega = 880\pi$ rad/sec, $\phi = -0.5\pi$ radians, and `dur = 0.01` seconds:

Verified:_____ Date/Time:_____