GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2001
Lab #2: Introduction to Complex Exponentials

Date: 4–10 Sept. 2001

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

You should read the Pre-Lab section of the lab and do all the exercises in the Pre-Lab section before your assigned lab time. You **MUST** complete the online Pre-lab exercise on Web-CT at the beginning of your lab session during the period 4-Sept to 10-Sept. You can use MATLAB or any notes you might have but you cannot discuss the exercises with any other students. You will have approximately 20 minutes at the beginning of your lab session to complete the online Pre-Lab exercise. The Pre-Lab exercise for this lab includes some questions about concepts from the Lab#1 report as well as questions on the Pre-Lab section of this lab.

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 raise your hand and demonstrate the step to the TA or instructor. After completing the warm-up section, turn in the verification sheet to your TA.

It is only necessary to turn in Section 5 as this week's lab report. More information on the lab report format can be found on Web-CT under the 'Information" link. You are asked to **label** the axes of your plots and include a title for every plot. In order to reduce missing plots, include your plot *inlined* within your report. For more information on how to include figures and plots from MATLAB to your report file, consult the "information" link on Web-CT. If you still do not know how to do so, ask your TA.

Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students and you are allowed to consult old lab reports but the submitted work should be original and it should be your own work.

The lab report and verification will be graded out of 100 points. The Pre-Post-Lab questions will be graded separately and will be worth about 20–30 points per lab.

The report will be **due during the period 11-Sept to 17-Sept at the start of your lab.**

PRINTING BUDGET: For the printers in the ECE labs, you have a quota. Please limit your printing to essential items for the labs. If you need to print lecture slides and other large documents, use the central (OIT) printing facilities.

---

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 complex amplitude and then the real part operator applied to Euler's formula:

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

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, try 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 $e^{j\theta}$

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`.

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

[2] In 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.

2.2 Sinusoid Addition Using Complex Exponentials

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

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

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*

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

Then the complex amplitude of the sum is

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

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.,

$$f_k = kf_0 \qquad \text{(HARMONIC FREQUENCIES)}$$

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

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

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.* (Unlike the single frequency case, there is no phasor addition theorem here to combine the harmonic sinusoids.)

3 Pre-Lab

You need to do all exercises in this section to be able to solve the on-line pre-lab exercise.

3.1 Complex Numbers

This section will test your understanding of complex numbers. Use $z_1 = 2e^{j\pi/3}$ and $z_2 = -\sqrt{2} + 5j$ for all parts of this section.

(a) Enter the complex numbers z_1 and z_2 in MATLAB and 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) 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`.

(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 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.

(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.

3.2 Z-Drill

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.

3.3 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.,

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

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 n=1:M
   x(i) = i;
   y(i) = cos( 0.001*pi*x(i)*x(i) );
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 few mistakes. Before looking at the correct one below, try to find these mistake(s) (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
%      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 Warm-Up: 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.

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 exactly 32 values of the sinusoid per period. Call this function `one_cos()`. *Hint: use* `goodcos()` *from the Pre-Lab part as a starting point.*

Demonstrate that your `one_cos()` function works by plotting the output for the following parameters: $A = 10^4$, $\omega = 3\pi \times 10^6$ rad/sec, $\phi = -\pi/4$ radians, and `dur` = 10^{-6} 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 microseconds?

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

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

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

4.2.1 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, Ak, phik, dur)
%SYN_SIN  Function to synthesize a sum of cosine waves
%  usage:
%    [xx,tt] = syn_sin(fk, Ak, phik, dur)
%      fk = vector of frequencies
%              (these should all be positive)
%      Ak = vector of amplitudes
%    phik = vector of phases
%     dur = total time duration of the signal
%              (starting time should be 0)
%      xx = vector of sinusoidal values
%      tt = vector of times, for the time axis
%
%    Note: fk, Ak, and phik must all be the same length.
%             Ak(1) and phik(1) corresponds to frequency fk(1),
%             Ak(2) and phik(2) corresponds to frequency fk(2),
%    The tt vector should be generated with a small time increment that
%         creates 25 samples per period. Use the period corresponding to
%         the highest frequency in the fk vector.
```

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`, `Ak` and `phik` are all the same. See `help error`.

4.2.2 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], [10,14,8], [0,-pi/3,pi/2], 0.1);   %-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 Exercises: Direction Finding

Why do humans have two ears? One answer is that the brain can process acoustic signals received at the two ears and determine the direction to the source of the acoustic energy. Using sinusoids, we can describe and analyze a simple scenario that explains this "direction finding" capability in terms of phase differences (or time-delay differences). This same principle is used in many other applications including radars that locate and track airplanes.

5.1 Direction Finding with Microphones

Consider a simple measurement system that consists of two microphones that can both hear the same source signal. If the microphones are placed some distance apart, then the sound must travel different paths from the source to the receivers. When the travel paths have different lengths, the two signals will arrive at different times. Thus a comparison of the two received signals will allow us to *measure* the relative time difference (between peaks), and from that we can *calculate* direction. If the source signal is a sinusoid, we can could measure the travel times by measuring phases.

The scenario is diagrammed in Fig. 1 where a vehicle traveling on the roadway has a siren that is "transmitting" a very loud sinusoidal waveform whose frequency is $f = 400$ Hz. The roadway forms the x-axis of a coordinate system. The two receivers are located some distance away, but are aligned parallel to the roadway. The distance from the road is $y_r = 100$ meters, and the receiver separation is $d = 0.4$ meters. The signals at the receivers must be processed to

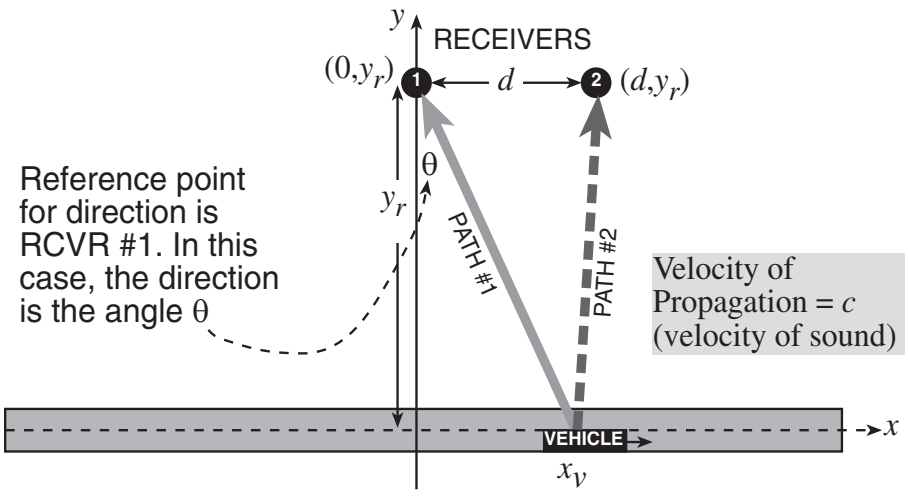

Figure 1: Scenario for multipath in mobile radio. A vehicle traveling on the roadway (to the right) receives signals from two sources: the transmitter and a reflector.

find the angle from Receiver #1 to the vehicle, which is denoted as θ in Fig. 1.

(a) The amount of the delay (in seconds) can be computed for both propagation paths. First of all, consider Path #1 from the vehicle to Receiver #1. The time delay is the distance from the vehicle location $(x_v, 0)$ to the

receiver at $(0, y_r)$, divided by the speed of sound which is approximately $c = 333\frac{1}{3}$ m/s. Write a mathematical expression for the time delay in terms of the vehicle position x_v. Call this delay time t_1 and express it as a function of x_v, i.e., $t_1(x_v)$.

(b) Now write a mathematical formula for the time delay of the signal that travels path #2 from the transmitter at $(x_v, 0)$ to Receiver #2 at (d, y_r). Call this delay time t_2 and make sure that you also express it as a function of x_v, i.e., $t_2(x_v)$.

(c) The received signals at the receivers, called $x_1(t)$ and $x_2(t)$, are delayed copies of the transmitter signal

$$x_1(t) = s(t - t_1) \qquad \text{and} \qquad x_2(t) = s(t - t_2)$$

where $s(\cdot)$ is the transmitted (sinusoidal) signal.[3]

Assume that the source signal $s(t)$ is a zero-phase sinusoid at $f_0 = 400$ Hz; and also assume that the amplitude of the transmitted signal is 1000. Make a plot of $x_1(t)$ and $x_2(t)$ when $x_v = 100$ meters. Use `subplot` to put both signals on the figure. Plot only 3 periods and then measure the *relative time-shift* between the two received signals by comparing peak locations.

(d) How do we convert *relative time-shift* into the direction θ? The answer is the following equation which can be solved for θ:

$$\frac{d}{c} \sin\theta = (t_1 - t_2) \tag{9}$$

For the *relative time-shift* obtained in the previous part, calculate θ. In addition, use geometry and the values of x_v and y_r to figure out what the "true value" of θ should be. Verify that your calculated value of θ is very close to the true value.

(e) The objective in the rest of this lab is to write a MATLAB function that will process the received signals to find direction. To do this, the received signals will be given as complex amplitudes, and a MATLAB function called `DF_gen` is supplied for generating the receiver signals.[4]

```
function [X1,X2,theta] = DF_gen(xx)
%DF_GEN  generate complex amplitudes at the two receivers
%        for the Direction Finding Lab
% usage  [X1,X2,theta] = DF_gen;
%
%      X1 = complex amplitude at Receiver #1
%      X2 = complex amplitude at Receiver #2
%   theta = the TRUE value of the "direction" in DEGREES
%
% alternate usage:
%       [X1,X2,theta] = DF_gen(xx);
%
%      xx = vector of "x positions" of the vehicle
%           then X1, X2 and theta are vectors
```

When you have the complex amplitudes, the relative time-shift cannot be measured directly so the measurement has to be done with the phases. In other words, the receivers will have complex amplitudes $X_1 = A_1 e^{j\phi_1}$ and $X_2 = A_2 e^{j\phi_2}$ and we must determine the phase difference $\Delta\phi = (\phi_1 - \phi_2)$. The phase difference can then be converted into a time difference according to the well-known relationship for sinusoids.

Show that you can compute the phase difference from X_1 and X_2 by doing the following:

$$\Delta\phi = \text{angle}\,\{X_1 X_2^*\} \tag{10}$$

where the "star" superscript denotes the complex conjugate. Use the ideas in (9) and (10) to write a MATLAB function that will compute the direction θ from the complex amplitudes.

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

[4] Download the binary file `DF_gen.p` from the Lab page on Web-CT.

(f) Debug the MATLAB program from the previous part by using the MATLAB function `DF_gen` to test individual cases. Then run your function for the vehicle moving from -400 meters to +500 meters, in steps of one meter. Compare the computed value of θ to the true value of θ (given by `DF_gen`) by plotting both on the same graph.

(g) *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 do the θ calculation for each x_v position, one at a time.
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 phase-difference and time-delay calculations to generate a vector of time delays; finally, a vector of calculated θ's would be formed. In each of these calculations, only one line of code is needed, i.e., *no loops.*

Lab #2
ECE-2025
Fall-2001
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your TA before 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 4.2.2 Show that your `syn_sin.m` function is correct by running the test in Section 4.2.2 and plotting the result. Measure the period(s) and explain why the period of `xx0` is longer that the periods of the signals used to form `xx0`. Write your explanations in the space below.

Verified:_____ Date/Time:_____