GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Spring 2006
Lab #2: Introduction to Complex Exponentials

Date: 24–30 Jan. 2006

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. You can use MATLAB or any notes you might have, but you cannot discuss the exercises with any other students. You will have approximately 15 minutes at the beginning of your lab session to complete the online Pre-Lab exercise. The Pre-Lab exercise for this this lab also includes some questions about concepts from the *previous* report.

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 4 as the lab report for this lab. 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 as a figure *embedded* within your report. For more information on how to include figures and plots from MATLAB in 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 you cannot give or receive written material or electronic files. Your 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.

Due Date: The lab report will be **due during the period 31-Jan. through 6-Feb. 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 and Overview

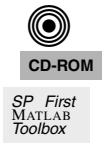
The goal of this laboratory is to gain familiarity with complex numbers and their use in representing sinusoidal signals such as $x(t) = A\cos(\omega t + \varphi)$ as complex exponentials $z(t) = Ae^{j\varphi}e^{j\omega t}$. The key is to use the complex amplitude and then the real part operator applied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

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

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.

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 *SP 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`.

1.2 Sinusoid Addition Using Complex Exponentials

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

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

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

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

[1] Correct installation means that the `spfirst` 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 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.

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*

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

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

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

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 $T_0 = 1/f_0$.

1.3 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 frequencies (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

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

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.)

2 Pre-Lab

You should do all exercises in this section to prepare for the on-line pre-lab exercise.

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 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 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 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.

2.2 Z-Drill

Work a few problems generated by the complex number drill program. To start the program simply type `zdrill` (if necessary, install the GUI and add `zdrill` to MATLAB's path). Use the buttons on the graphical user interface (GUI) to produce different problems.

2.3 The Notebook Option

The purpose of this section is to introduce you to the notebook capability that links MATLAB and Microsoft Word under Windows. This option allows you to execute MATLAB commands, scripts and functions from inside a Microsoft Word document and then have the results (including graphs) displayed automatically inside that same MS-Word document. This should make it easy to create lab reports by reducing the problems (and pain) associated with importing figures and MATLAB results into MS-Word. You are not required to use this feature to write your lab reports, but many students find it useful.[3]

Note that GUI items such as `zdrill` will not work from inside MS-Word. Also note that MATLAB functions cannot be defined from inside a MS-Word document.

2.4 Notebook setup

Depending on whether or not you have "Administrator privileges" on your Windows machine, there are two ways to set up the `notebook` functionality (which works in versions 5.x, 6.x and 7.x of MATLAB).

1. This *easy* setup works if your Windows account has administrator privileges: From inside MATLAB, type `notebook -setup`, and follow the instructions given.[4]

2. This *harder* setup works even for accounts with non-administrative privileges.

   (a) Open MS-Word and then go to the menu `Tools->Macro->Security` and select `medium` instead of `high`. Click the `OK` button.

[3] In MATLAB 7 there is a new feature that allows you to publish MATLAB code to a Word document. We will not use this during the lab, but an introduction to this new MATLAB feature has been written and is available as a link on Web-CT.

[4] Once the the setup is complete, click on the start menu and find the `New Office Document` option. Under the `General tab` you should find `m-book.dot`. Click on it to start MS-Word. This might also start up MATLAB if it is not already running.

(b) Now go to the `File` menu in MS-Word, and open the file `m-book.dot` which is in the folder[5] `C:\Appl\MatlabR14\Notebook\PC\`. *Note:* If a "Security Warning" window pops up, click "Enable Macros" to proceed. Please ignore the Microsoft Visual Basic error "Command Failed", and press "End" to continue.

(c) MATLAB will start up and once it has loaded up, return to MS-Word and select the `File` menu and pick `New M-Book`.

- ***Once MS-Word has opened a notebook document, there will be a new menu in MS-Word called* Notebook *which enables a variety of operations related to* MATLAB.**

2.4.1 Using MATLAB from MS-Word

Once the `notebook` capability has been installed you can start it by either: (1) typing `notebook` at the MATLAB command prompt, or (2) clicking on the `Start Menu -> Microsoft Office Tools` and finding the `New Office Document` option, under which you select `m-book.dot` from the `General` tab.

- To verify that the `notebook` capability works, from inside MS-Word, type

  `varA = 1+1i - 6`, and then press the `Ctrl-Enter` keys together.

  This should give you MATLAB's computed answer in the MS-Word document.

- To create a plot inside MS-Word, type the the line below (followed by `Ctrl-Enter`)
  `t=0:0.1:pi; y=sin(3*t+0.05); plot(t,y); title('My First Plot')`

- To call one of your functions or MATLAB script files, just type in the name followed by `Ctrl-Enter`. Built-in MATLAB functions are called in the same way, for example
  `[V,lam] = eig([1 2 3; 3 3 3; -1 2 3])` followed by `Ctrl-Enter`

- Take some time to investigate the other items and more advanced features under the `Notebook` menu that was installed in MS-Word. Additional help can be found in MATLAB by typing `helpdesk` and searching for Notebook.

- If you dislike the way that figures get inserted inside MS-Word with a grey background, then you can change to a white background by resetting one of MATLAB's defaults. One way is to execute the following at the beginning of the MS-Word document
  `set(0,'DefaultFigureColor',[1 1 1]);set(0,'DefaultAxesColor',[1 1 1]);`

2.5 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,

[5] The path to MATLAB might be different on your personal machine depending on where you installed MATLAB.

```
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 with one line 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-' )
```

Run these two programs to see that they give identical results, but the vectorized version runs much faster. Use the `notebook` capability to put the plots into a MS-Word document.

2.6 Vectorizing a 2-D Evaluation

Suppose that you want to plot $f(u, v) = u^2 + v^2$ versus (u, v) over the domain $[-20, 20] \times [-20, 20]$. The result should be a parabolic surface. To avoid having nested `for` loops, we can use `meshgrid` instead:

```
u = -20:0.5:20;
v = -20:0.5:20;
[uu,vv] = meshgrid(u,v);
mesh(u,v,uu.*uu + vv.*vv)
```

The `meshgrid` function generates all the pairs (u, v) for the domain.

2.7 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-6 in Appendix B of the text for more discussion.

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

```
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 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` in the first line, and the exponential needs to have an imaginary exponent. Also, the variable `freeq` has not been defined before being used. Finally, the function has `xx` as an output and hence the variable `xx` should appear in 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 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. Use MATLAB's `notebook` capability to put the plots into a MS-Word document when doing the Instructor Verifications.

3.1 Vectorization

Use the 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.

```
%--- 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 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 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 25 values of the sinusoid per period. Call this function `onecos()`. *Hint: use* `goodcos()` *from the Pre-Lab part 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 the `notebook` feature to make this plot inside a MS-Word document.

Instructor Verification (separate page)

3.3 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 MATLAB 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 $X_k = A_k e^{j\varphi_k}$.

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

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

3.3.1 Write the Sum of Sinusoids M-file

Write an M-file called `multi_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] = multi_sines(fk, Xk, dur, tstart)
%MULTI_SINES  Synthesize a signal from sum of complex exponentials
%  usage:
%    [xx,tt] = multi_sines(fk, Xk, dur, tstart)
%      fk = vector of frequencies   (usually none are negative)
%      Xk = 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: fk and Xk must be the same length.
%             Xk(1) corresponds to frequency fk(1),
%             Xk(2) corresponds to frequency fk(2), etc.
%    The tt vector should be generated with a small time increment that
%         creates 25 samples for the shortest period, i.e., 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` and `Xk` are all the same. See `help error`.

3.3.2 Testing

In order to verify that this M-file can synthesize sinusoids, try the following test and plot the result in MS-Word via the `notebook` capability.

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

Measure the DC value (which is also the average value) and also the period of `xx0` on the plot. Then 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 the fundamental frequency.

Instructor Verification (separate page)

4 Lab Exercises: Direction Finding

Why do mammals have two ears? One answer is that a brain can process acoustic signals received at the two ears and determine the direction to the source of the acoustic energy. It might be tempting to think that the ears can distinguish amplitude changes, but it's actually phase that matters. Using sinusoids, we can describe and analyze a simple scenario that demonstrates "direction finding" in terms of phase differences. This same principle is used in many other applications including radars that locate and track airplanes.

4.1 Directional Nulling 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. Since time shift corresponds to phase, we say that the signals arrive "out of phase."

The received signal at a microphone, called $r(t)$, is a delayed copy of the transmitter signal $s(t)$. Thus we could write

$$r(t) = s(t - t_d)$$

where t_d is the amount of time it takes for the signal to travel from the source to the receiver and $s(\cdot)$ is the transmitted (sinusoidal) signal.[6] The travel time t_d can be computed easily once we know the speed of sound and the locations of the source and receiver.

We can combine the signals from two receivers in many different ways. Consider the case of one source transmitting the signal $s(t)$ to two receivers. The received signals could be labeled as

$$\begin{aligned} \text{Receiver \#1:} \quad & r_1(t) = s(t - t_1) \\ \text{Receiver \#2:} \quad & r_2(t) = s(t - t_2) \end{aligned}$$

where t_1 is the propagation time from the source to Receiver #1, and t_2 the propagation time from the source to Receiver #2. If we combine the two signals by delaying one and then adding them, i.e.,

$$y(t) = r_1(t) + r_2(t - t_d) \tag{9}$$

then we can demonstrate that the parameter t_d will be able to "steer" the maximum (or minimum) response to different directions. When considering the minimum response, t_d would be adjusted to null out signals from a particular direction.

(a) The amount of the delay (in seconds) can be computed for both propagation paths. First of all, consider the path from the transmitter at (x_t, y_t) to Receiver #1. The time delay is the distance from the transmitter location (x_t, y_t) to the receiver at $(0, y_1)$, divided by the speed of sound, for which we will use the approximate value of $c = 330$ m/s. Write a mathematical expression for this time delay, which will be called t_1.

(b) Now write a mathematical formula for the time delay of the signal that travels the path from the transmitter at (x_t, y_t) to Receiver #2 at $(0, y_2)$. Call this delay time t_2 and write a mathematical expression for t_2 in terms of d_2, the distance from transmitter to receiver.

(c) The signals at the receivers, called $r_1(t)$ and $r_2(t)$, are delayed copies of the transmitter signal. Assume that the source signal $s(t)$ is a sinusoid with a frequency of $f_0 = 660$ Hz; and also assume that the

[6] For simplicity we will ignore propagation losses. Usually, the amplitude of an acoustic signal that propagates over a distance R will be reduced by an amount that is inversely proportional to R.

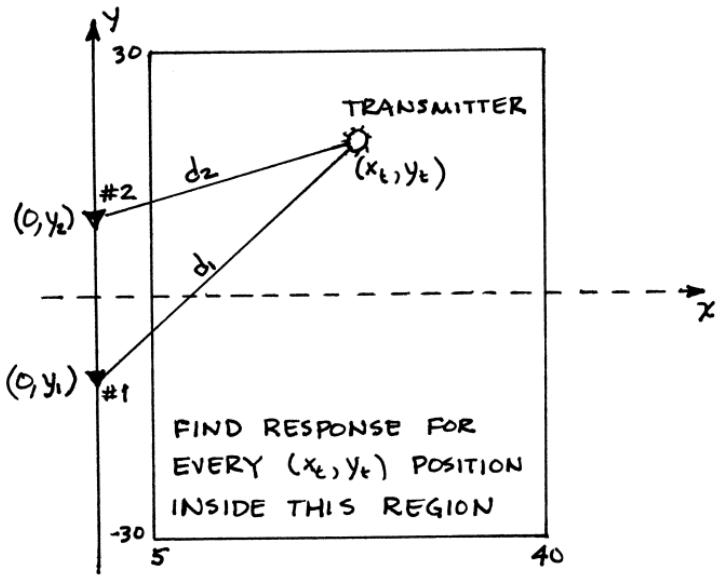


Figure 1: Region where the response should be calculated.

complex amplitude of the transmitted signal is $5e^{j\pi/2}$. Make a plot of $r_1(t)$ and $r_2(t)$ when $x_t = 17.83$ m, and $y_t = 10$ m. Assume that the receivers are located at $y_1 = -0.1$ meters and $y_2 = 0.1$ meters. Use `subplot` to put both signals on the same figure. Plot only 3 periods of the received sinusoids and then measure the *relative time-shift* between the two received signals by comparing peak locations.

(d) Continue the previous part, but now form the output signal, $y(t)$. Produce a plot of the output signal $y(t)$ for the case where $t_d = 1$ msec. On the plot, measure the peak amplitude of the output sinusoid.

(e) Describe how the calculation of peak amplitude in the previous part could have been done with a single phasor addition by examining (9) for the case where $s(t)$ is a sinusoid. Give the explicit value of the complex amplitudes needed when doing the calculation with phasors. Explain how you get the same peak amplitude value as you measured on the plot in the previous part.
Note: The complex amplitudes at the receivers $R_1 = A_1 e^{j\phi_1}$ and $R_2 = A_2 e^{j\phi_2}$ can be found by calculating the distances from the transmitter to each receiver, and using the speed of sound (c) to calculate time delay which can then be converted to phase.

(f) The next step in this lab is to write a MATLAB function that will determine the peak amplitude of the output for all possible transmitter locations (x_t, y_t) within a certain region. To do this, the received signals should be manipulated as complex amplitudes, not as sinusoidal waveforms.

The MATLAB function that you write should exploit vectorization to avoid `for` loops, like the vectorization example in Section 3.1 in the warmup. The result of your function should be a 3-D mesh plot, or a 2-D image plot, that shows the peak output amplitude for (x_t, y_t) in the region:

$$0.5 \le x_t \le 25 \qquad \text{and} \qquad -25 \le y_t \le 25 \text{ meters}$$

Show the result of your function for the case where $t_d = 1$ msec. In addition, find the places where the output is zero. Notice that transmitter positions along a line give zero output—call this the *nulling line*. That line emanates from the origin. Determine the angle it makes with the horizontal axis—call this the *nulling direction.*

(g) *Vectorization:* It is likely that your previous programming skills would lead you to write a double loop to do this implementation. The loop would run over all possible locations of the transmitter (x_t, y_t), and would do the combination of the two receivers for each transmitter position, one at a time. However, it is possible to vectorize this program by replacing one loop with a vectorized statement as was done in Section 3.1. Eliminating both loops is a bit harder, but can be done with the MATLAB function `meshgrid` which provides a way to generate all the (x_t, y_t) positions at once.

(h) Now consider a design problem: we want to select t_d so that the nulling direction is 30°. Make the appropriate calculations to find t_d to null 30° and then show a region plot to verify your result.

4.1.1 Miscellaneous Comments

The objective is to make a 3-D plot of peak amplitude versus transmitter location (x_t, y_t). One way to make a 3-D plot of a function over (x_t, y_t) is to use the MATLAB function `mesh`. Another way is to make an intensity image versus (x_t, y_t) with `imagesc`. A third way would be to use `contour`.

In order to generate a good plot of the peak amplitude versus (x_t, y_t), it will be necessary to have a dense grid over the region of interest. A dense grid will make it easy to find the locations where the response is zero. However, as the grid density goes up, the running time of the program will also increase. Doubling the grid density in both x and y would quadruple the running time.

Lab #2
ECE-2025
Spring-2006
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your TA before the end of your lab period.

Name: _____ Date of Lab: _____

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 `notebook` capability to create and save the plots when doing the Instructor Verifications. Show the MS-Word document to your TA.

Verified:_____ Date/Time:_____

Part 3.3.2 Show that your `multi_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. Write your explanations in the space below.

Verified:_____ Date/Time:_____