GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2003
Lab #2: Introduction to Complex Exponentials

Date: 2–8 Sept. 2003

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 20 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 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 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.

The lab report will be **due during the period 9-Sept. through 15-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 and Overview

The goal of this laboratory is to gain familiarity with complex numbers and their use in representing sinusoidal signals such as $x(t) = A\cos(\omega t + \phi)$ 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:

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

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:

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

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

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

[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*

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

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 $T_0 = 1/f_0$.

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

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

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 the exercises in this section to prepare for the on-line pre-lab questions.

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 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 results into MS-Word. You are not required to use this feature to write your lab reports, but many students find it useful.

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 and 6.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.[3]

2 This *harder* setup that works even for accounts with non-administrative privileges (i.e., it works in ECE Labs such as BH-216).

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

[3] 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 `C:\Appl\MatlabR13\Notebook\PC\`.

(c) A dialog box comes up where you type the path to the `matlab.exe` file.
Type in `C:\Appl\MatlabR13\Bin\win32\`.

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

- A new MS-Word window will come up and the setup is complete. MS-Word will have a new menu called `Notebook` which contains a number of new commands for communicating with 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) by selecting `New M-Book` in the `File` menu of MS-Word.

- To verify that the `notebook` capability works, from inside MS-Word, type `var_A=1+1i-6`, and then press the *Ctrl-Enter* keys together. This should give you the 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,

```
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. Use the `notebook` capability to put the plots into a MS-Word document.

2.6 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 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, 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, $x(t) = A\cos(\omega t + \phi)$, by using five input arguments: amplitude (A), phase (ϕ), frequency (ω), 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 32 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( [5], [-pi/3], [2], 2, -1);
```

Use the `notebook` feature to make this plot inside a MS-Word document.

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

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

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

3.3.1 Write the Sum of Sinusoids M-file

Write an M-file called `addcos.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 outer loop in this lab.* The inner loop can be vectorized. The first few statements of the M-file are the comment lines—they should look like:

```
function    [xx,tt] = addcos(Ak, phik, fk, dur, tstart)
%ADDCOS  Synthesize a sum of cosine waves
%  usage:
%    [xx,tt] = addcos(Ak, phik, fk, dur, tstart)
%      Ak = vector of amplitudes
%    phik = vector of phases
%      fk = vector of frequencies   (all positive)
%     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, 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), etc.
```

```
%    The tt vector should be generated with a small time increment that
%         creates 32 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`.

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] = addcos( [5,2,3], [-pi/3,pi/4,pi], [6,4,0], 2, -1); %-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: Fading in Mobile Radio

4.1 Representation of Sinusoids with Complex Exponentials

In MATLAB consult `help` on `exp`, `real` and `imag`. Be aware that you can also use the SP First function `zprint` to print the polar and rectangular forms of any vector of complex numbers.

(a) Generate the signal $x(t) = \Re e\{-2e^{j75\pi t} + 5e^{j75\pi(t+0.02)} + (-4-j3)e^{j75\pi t}\}$ and make a plot versus t. Use the `addcos` function and take a range for the time axis t that will cover 3 periods. *Include the* MATLAB *code and the plot with your report.*

(b) From the plot of $x(t)$ versus t, measure the frequency, phase and amplitude of the sinusoidal signal by hand. Show annotations on the plots to indicate how these measurements were made and what the values are. Compare to the calculation in the next part.

(c) Use the phasor addition theorem along with MATLAB to determine the magnitude and phase of $x(t)$ from the complex amplitudes of the three terms.

4.2 Multipath Fading

In a mobile radio system (e.g., cell phones or AM radio), there is one type of degradation that can be modeled easily with sinusoids. This is the case of *multipath fading* caused by the sum of reflected radio waves that combine constructively or destructively at various locations. Consider the scenario diagrammed in Fig. 1 where a vehicle traveling on the roadway receives signals from two sources: directly from the transmitter and a reflection from another object such as a large building. The total received signal at the vehicle is the sum of the two signals which are themselves delayed versions of the transmitted signal, $s(t)$.

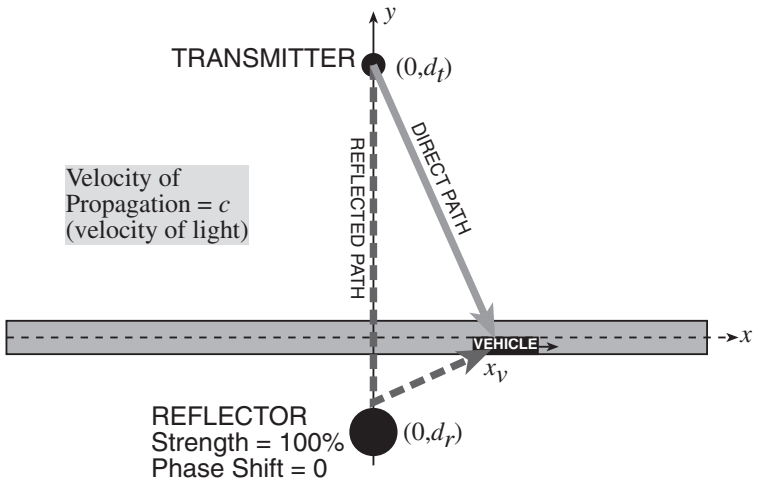

Figure 1: Scenario for multipath in mobile radio. A vehicle traveling on the roadway receives signals from two sources: the transmitter and a reflector. The reflector might be a large structure such as a building.

(a) The amount of the delay (in seconds) can be computed for both of the propagation paths. First of all, consider the direct path. The time delay is the distance divided by the speed of light (3×10^8 m/s). Write a mathematical expression for the time delay in terms of the vehicle position x_v and the transmitter location $(0, d_t)$. Call this delay time t_1 and make sure that you express it as a function of x_v, i.e., $t_1(x_v)$.
Notice that the y-axis has been chosen conveniently so that both the transmitter and the reflector have an x coordinate equal to zero.

(b) Now write a mathematical formula for the time delay of the signal that travels the reflected path from the transmitter at $(0, d_t)$ to the reflector at $(0, d_r)$ and then to the vehicle at $(x_v, 0)$. Call this delay time t_2 and make sure that you also express it as a function of x_v, i.e., $t_2(x_v)$. In this case, you must add together two delays: transmitter to reflector and then reflector to vehicle.

(c) The received signal at the vehicle, $r_v(t)$, is the sum of the two delayed copies of the transmitter signal

$$r_v(t) = s(t - t_1) + s(t - t_2)$$

where $s(\cdot)$ is the transmitted signal, and the reflection is assumed to be perfect.[4]

Assume that the source signal $s(t)$ is a zero-phase unit-amplitude sinusoid at $f_0 = 120$ MHz; and also assume that the transmitter is located at (0, 1500) meters and the reflector at (0, −300.3) meters. Then the received signal, $r_v(t)$, is the sum of two sinusoids. Make a plot of $r_v(t)$ when the vehicle position is $x_v = 50$ meters. Plot 3 periods of $r_v(t)$ and then measure its maximum amplitude.

(d) Repeat the work in part (c), but use the complex amplitudes (of the sinusoids) instead of the time signals. Explain how a single complex addition, followed by a magnitude operation can be used to find the amplitude of $r_v(t)$.

(e) The objective in the rest of this lab is to make a plot of signal strength versus vehicle position (x_v). One approach would be to repeat the process in part (c) for every position, i.e., generate the resultant

[4]For simplicity we are also ignoring propagation losses: When a radio signal propagates over a distance R, its amplitude will be reduced by an amount that is inversely proportional to R^2.

sinusoid (in time) and measure its amplitude. However, that would be a huge waste of computation. Instead, the complex amplitude approach in part (d) is a much more efficient method.

Derive a general mathematical expression for the complex amplitudes of both delayed sinusoids as a function of position x_v. Then write a MATLAB program that will generate the time delays, form the complex amplitudes, and finally add together the complex amplitudes. Have the MATLAB program loop through the entire set of vehicle positions specified in part (f) below. Include this MATLAB code in your report and explain how it works.

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 add the two complex amplitudes calculated at each x_v.
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 distance and time delay calculations to generate a vector of time delays; next, a vector of complex amplitudes would be formed. In each of these calculations, only one line of code is needed and *no loops.* You need two vectors of complex amplitudes (one for the direct path and the other for the reflected path), and then you can perform a vector add of the two complex amplitudes.

(f) Utilize the MATLAB program from the previous part to generate a plot of maximum signal strength versus vehicle position, x_v, over the interval from 0 meters to +150 meters.[5] Assume that *maximum signal strength* is defined to be the peak value of the received sinusoid, $r_v(t)$. State how you get the peak value from the complex amplitude.

(g) Explain your results from the previous part. In particular, what are the largest and smallest values of received signal strength? Why do we get those values? Are there vehicle positions where we get complete signal cancellation (i.e., zero signal strength)? If so, determine those vehicle positions.

[5] MATLAB works on vectors so you will have to produce a vector of positions starting at 0 and ending at $+150$ with a spacing that is *small enough* to capture all the variations in signal strength.

Lab #2
ECE-2025
Fall-2003
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 `for` loop with a couple of lines of vectorized MATLAB code. Write the MATLAB code in the space below:

Verified:_____ Date/Time:_____

Part 3.3.2 Show that your `addcos.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:_____

General: 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:_____