GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Spring 2011
Lab #2: Using Complex Exponentials

Date: 31 Jan. – 3 Feb. 2011

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.

ITS: When you come to the lab, you **must** answer the online ITS questions. You can use MATLAB or any notes you might have but you cannot discuss the exercises with any other students.

Verification: 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**. 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 *before leaving the lab.*

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 **`t-square`** under the **INFO** link. Please **label** the axes of your plots and include a title for every plot. In order to reduce "orphan" plots, include each 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 **INFO** link on **`t-square`**, or ask your TA for details.

Electronic Submission: Please submit all M-files and electronic documents in a .zip file via **`t-square`**.

Peer Evaluation: Along with the lab report, each member of the team must fill out an on-line Peer Evaluation form that will be found within **`t-square`**.

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, but you cannot give or receive any written material or electronic files. In addition, you are not allowed to use or copy material from old lab reports from previous semesters. Your submitted work must be your own original work.

Due Date: The lab report will be **due during the period 7–10 Feb. at the start of your lab.**

---

1 Introduction and Overview

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

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

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:

CD-ROM
SP First MATLAB Toolbox

```
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]

CD-ROM
Z Drill

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:

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

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

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

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*

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

[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.

Then the complex amplitude of the sum X_s 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 (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 becomes

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

This 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 to combine the harmonic sinusoids.)

2 Pre-Lab

Please do the exercises in this section prior to coming to lab.

2.1 Complex Numbers

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

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

When unsure about a command, use `help`.

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 Cell Mode in MATLAB

MATLAB has a formatting syntax that allows you to produce documentation at the same time that you make an M-file. A quick summary is that double percent signs followed by a space (`%%␣`) are interpreted as sections in cell mode so that your code is broken into natural blocks that can be run individually. In addition, the M-file can be "published" to an HTML file and then viewed as a nicely formatted web page. There are several sources with information about cell mode:

1. Videos:
   `http://www.mathworks.com/support/2009b/matlab/7.9/demos/PublishingfromtheEditor_viewlet_swf.html`
   `http://www.youtube.com/watch?v=fZDEQCWK2OM`
   `http://www.youtube.com/watch?v=sVoO7X5dIoQ`
   and `http://www.mathworks.com/support/2009b/matlab/7.9/demos/RapidCodeIterationUsingCells_viewlet_swf.html` (Rapid Code Iteration)

2. MATLAB documentation:
   `http://www.mathworks.com/help/techdoc/matlab_env/f6-_22451.html`
   and `http://www.mathworks.com/help/techdoc/matlab_env/brqxeeu-_259.html`

Please watch the videos, especially the first one, and read some of the documentation.

The MATLAB code in the following section uses cell mode. It can be published to HTML, and displayed in a web browser (the default is MATLAB's internal browser).

2.4 Structures in MATLAB

MATLAB can do structures. Structures are convenient for grouping information together. For example, we can group all the information about a sinusoid into a single structure with fields for amplitude, frequency and phase. We could also add fields for other attributes such as a signal name, the signal values, and so on. To see how a structure might be used, publish (i.e., run) the following M-file which plots a sinusoid:

```
%% Structure Example
%
%% Generate the Sinusoid
%
%%
% Here is the general formula:
%
% $x(t) = A\cos(2\pi f t + \varphi)$
%
x.Amp = 7;
x.phase = -pi/2;
x.freq = 100;
x.fs = 11025;
%-- sampling rate controls the spacing of values on the time grid
x.times = 0:(1/x.fs):0.05;
x.values = x.Amp*cos(2*pi*(x.freq)*(x.times) + x.phase);
x.name = 'My Signal';
%% Display the Stucture
% echo the contents of the structure "x"
x
%% Make the plot
plot( x.times, x.values )
title( x.name )
%% The end
```

Notice that the fields in the structure can contain different types of variables: scalars, vectors or strings.

You can also have arrays of structures. For example, if `xx` is array of sinusoid-structures with the same fields as above, you would reference one of the sinusoids via:

```
 xx(3).name, xx(3).Amp, xx(3).freq, xx(3).phase
%
 plot( xx(3).times, xx(3).values )
 title( [xx(3).name,' Amp=',num2str(xx(3).Amp),' Phase=',num2str(xx(3).phase)] )
```

Notice that the array name is `xx`, so the array index, 3, is associated with `xx`, e.g., `xx(3)`.

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.,

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

where `vv` is an N-element row vector. Vectorization can be used to simplify your code. If you have the following code that plots the signal in the vector `yy`,

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

then you can replace the `for` loop with one line and get the same result with four lines of code:

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

Run these two programs to see that they give identical results, but note that 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

You can also vectorize 2D functions. 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, vectors, structures, etc.) and also return outputs. The keyword `function` must appear as the first non-comment word in the M-file that defines the function, and that line of the M-file defines how the function will pass input and output arguments. The file extension must be lower case "m" as in `my_func.m`. See Section B-6 in Appendix B of the text for more discussion.

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

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

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

2.8 The Notebook Option (Information for Windows only)

There is an alternate way to document MATLAB programs which is useful in creating lab reports. The purpose of this section is to introduce the notebook capability that links MATLAB and Microsoft Word under Windows.

`http://www.mathworks.com/help/techdoc/matlab_env/brgbdb8.html`

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.

2.9 Notebook setup

1. In Klaus-2440 (the ECE-2025 lab), MATLAB's notebook feature is already installed. Typing `notebook` at the MATLAB prompt, or `notebook Z:\filename.doc`, should work.
   Note: If you get an error message, check **`t-square`** for files that have to be moved to your `Z:` drive.

2. If you have "Administrator privileges" on your Windows machine, then this *easy* setup works: From inside MATLAB, type `notebook -setup`, and it should be configured automatically; if not, follow the messages given by MATLAB.[4] Consult `help notebook` for usage.

3. It is possible to setup Notebook for accounts without administrative privileges, but it's much harder.

- ***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.**

2.9.1 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) using the **Notebook** menu to open a new **M-book** in MS-Word.

- 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 displayed 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 commands at the beginning of the MS-Word document
  `set(0,'DefaultFigureColor',[1 1 1]);set(0,'DefaultAxesColor',[1 1 1]);`

[3] In MATLAB 7 the cell-mode feature allows you to publish MATLAB code to a Word document. An introduction to this new MATLAB feature has been written and is available in the PDF file `mfile_to_doc.pdf` under the **INFO** link on **`t-square`**.

[4] Once 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.

3 Warm-Up: Complex Exponentials

In the Pre-Lab section of this lab, you saw examples of function M-files. In this section, you will write functions that can generate sinusoids, or sums of sinusoids. Use MATLAB's cell-mode capability to put the MATLAB code and plots into an HTML document when doing the Instructor Verifications. Demonstrate that you have run cell mode by showing the example from Section 2.4.

Instructor Verification (separate page)

For the instructor verification, you will have to demonstrate that you understand everything in a given subsection. It is not necessary to do everything in the subsections; so you can skip parts that you already know. The Instructor Verification is usually placed close to the most important item, i.e., the most likely one to generate questions from the TAs.

3.1 Vectorization

Use the vectorization idea to write one or two lines of code that will perform the *same task as the inner loop* of the following MATLAB script without using a `for` loop. If you are ambitious, try to replace both loops with some vectorized code.

```
%% Adding Complex Exponentials to Add Sinusoids
%--- make a plot of sum of cosines
dt = 1/800;
XX = rand(1,4).*exp(2i*pi*rand(1,4));  %--Random amplitude and phases
freq = 8;
cmplxsum = zeros(1,500);
for kx = 1:length(XX)
    for kt = 1:500
        t = kt*dt;
        cmplxsum(kt) = cmplxsum(kt) + XX(kx)*exp(2i*pi*freq*t);
        tt(kt) = t;
    end
end
xxsum = real(cmplxsum);
plot(tt,xxsum)   %-- Plot the sum sinusoid
grid on, zoom on, shg
```

Instructor Verification (separate page)

3.2 M-file to Generate One Sinusoid

Write a function that will generate a **single** sinusoid, $x(t) = A\cos(2\pi f t + \varphi)$. Call this function `mySinusGen()`. *Hint: use* `goodcos()` *from the Pre-Lab part as a starting point.* Here is a suggested template that needs to be completed for the M-file:

```
function sinusOut = mySinusGen( sinusIn, dur, tstart, dt )
%
freq = sinusIn.freq;
X = sinusIn.complexAmp;
%
%...(Fill in several lines of code)...
%
tt = tstart: dt : ???;    %-- Create the vector of times
xx = real(X*exp(...???;        %-- Vectorize the cosine evaluation
sinusOut.times = ???;       %-- Put vector of times into the output structure
sinusOut.values = ???;      %-- Put values into the output structure
```

The function should have the following input arguments: a sinusoid-structure with two fields for the frequency (f) in Hz, the complex amplitude ($X = Ae^{j\varphi}$), and then three other arguments: a duration argument (`dur`), followed by an argument for the starting time (`tstart`), and then a final argument which is the spacing between times, Δt. The function should return a structure having both of the fields of the input structure plus two new fields: the vector of values of the sinusoidal signal (x) along with the corresponding vector of times (t) at which the sinusoid values are known.

When you call `mySinusGen()`, make sure that the constant spacing between times in the time-vector, Δt, is small enough so that there are at least 50 time points per period of the sinusoid. Plot the result from the following call to test your function.

```
%% Testing |mySinusGen.m|
mySig.freq = 5;    %-- (in hertz)
mySig.complexAmp = 90*exp(-j*2*pi/3);
dur = 1.1;
start = -0.1;
dt = 1/(50*mySig.freq);
%myOutputSig = mySinusGen(mySig, ...?? );   %<--- Fix this line
%% Listing of |mySinusGen.m|
type mySinusGen
%% Plot the signal in myOutputSig
%plot( ?? )       %<----------- Fix this line
```

Use the cell-mode feature to publish this plot to HTML for a browser. Note: the function itself cannot be with the same M-file as the other cell-mode statements, but a listing can be made and included.

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 should allow the frequency of each component (f_k) to be different. The following expressions are equivalent if we define the complex amplitude X_k as $X_k = A_k e^{j\varphi_k}$.

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

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

3.3.1 Write a Sum of Sinusoids M-file

Write an M-file called `addManySines.m` that will synthesize a waveform in the form of (7) using X_k defined in (3). The result is not a sinusoid unless all the frequencies are the same, so the output signal has to be

represented by its values over some (finite) time interval.

Even though `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    sigOut = addManySines( sinesIn, dur, tstart, dt )
%ADDCOSVALS  Synthesize a signal from sum of sinusoids
%               (do not assume all the frequencies are the same)
%
%  usage:   sigOut = addManySines( sinesIn, dur, tstart, dt )
%
%   sinesIn = vector of structures; each one has the following fields:
%      sinesIn.freq = frequency (n Hz), usually none should be negative
%      sinesIn.complexAmp = COMPLEX amplitude of the cosine
%
%   dur = total time duration of all the cosines
%   start = starting time of all the cosines
%   dt = time increment for the time vector
% The output structure has only signal values because it is not necessarily a sinusoid
%      sigOut.values = vector of signal values at t = sigOut.times
%      sigOut.times  = vector of times, for the time axis
%
%    The sigOut.times vector should be generated with a small time increment that
%         creates 20 samples for the shortest period, i.e., use the period
%         corresponding to the highest frequency cosine in the input array of structures.
```

In order to verify that this M-file can synthesize ***harmonic*** sinusoids, try the following test:

```
%% Demonstrate Addition of Harmonic Sinusoids
ss(1).freq = 10;  ss(1).complexAmp = exp(-j*pi/4);
ss(2).freq =  4;  ss(2).complexAmp = -2i;
ss(3).freq =  0;  ss(3).complexAmp = -4;
%
dur = 2;
tstart = -1;
dt = 1/(10*20);   %-- use the highest frequency to define delta_t
%
ssOut = addManySines( ss, dur, tstart, dt );
%% Listing of |addManySines|
type addManySines
%% Plot of the Harmonic Signal
plot( ssOut.????, ssOut.???? )
%
```

Use the cell-mode feature when making the plot of `ssOut`. Notice that the waveform is periodic. Measure its period and state how the period is related to the fundamental frequency which is 2 Hz in this case.

Instructor Verification (separate page)

4 Lab Exercise: Simultaneous Equations in Circuits

One highly succesful application of the *phasor method* is the analysis of electric circuits operating in what is called the *sinusoidal steady-state*. In this condition, the circuit is known to operate at a single frequency, so that all the voltages and currents in the circuit are sinusoids at that one frequency. The electric power grid is a good example, where all the sinusoids have a frequency of 60 Hz.

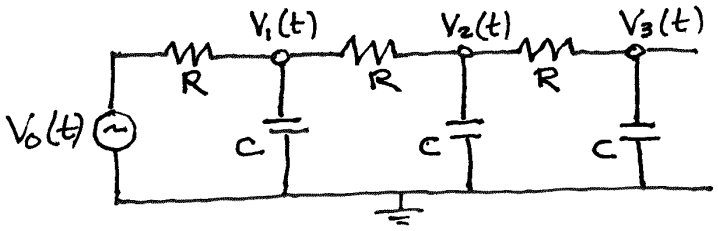

Figure 1: Ladder circuit with a known input voltage, $V_0(t)$, and three nodes where the voltages must be determined.

For this lab we consider circuits that are interconnections of passive elements: resistors, capacitors and inductors. In fact, we will study a circuit configuration called an R-C ladder which is composed of resistors and capacitors only. Figure 1 shows the circuit diagram where there are three ladder sections. The equations that completely describe the circuit are differential equations which can be derived from Kirchoff's Laws. We use what is called the *Node Voltage Method* by assigning unknown voltages $V_1(t)$, $V_2(t)$, and $V_3(t)$ to nodes in the circuit diagram as shown in Fig. 1. Then we invoke Kirchoff's law that says the sum of the currents at a node must be zero, and write the following three equations:

$$\text{First Node:} \qquad 0 = \frac{V_1(t) - V_0(t)}{R} + C\frac{d}{dt}V_1(t) + \frac{V_1(t) - V_2(t)}{R} \tag{9}$$

$$\text{Second Node:} \qquad 0 = \frac{V_2(t) - V_1(t)}{R} + C\frac{d}{dt}V_2(t) + \frac{V_2(t) - V_3(t)}{R} \tag{10}$$

$$\text{Third Node:} \qquad 0 = \frac{V_3(t) - V_2(t)}{R} + C\frac{d}{dt}V_3(t) \tag{11}$$

where $V_0(t)$ is a known voltage which is called the input voltage. The voltage $V_3(t)$ at the third node is usually called the output voltage.

If the input voltage is a sinusoid, then the circuit will operate in the sinusoidal steady-state after sufficient time has elapsed so that any transient voltages have died down. In the sinusoidal steady-state, all the voltages (and currents) in the circuit will be sinusoids.

$$V_k(t) = A_k \cos(\omega t + \varphi_k) \tag{12}$$

This is where complex amplitudes come into play: the differential equations can be solved by the *algebraic method of phasors.* All the sinusoids can be converted to complex amplitudes, and the three differential equations can be transformed into three simultaneous linear equations. Recall that the operation of taking the time-derivative of $z(t) = Ae^{j\varphi}e^{j\omega t}$ is easy: the result is a change in the complex amplitude of $z(t)$.

(a) Before writing any MATLAB code, do some mathematics: Transform the differential equations into the complex amplitude (phasor) form, and rearrange the equations as simultaneous linear equations with the unknown node voltages on the lefthand side (LHS), and the known voltage V_0 on the righthand side (RHS).[5] In this step retain the symbols R and C for the resistance and capacitance values.

[5] It might be convenient to multiply all equations by R to clear the denominators.

(b) Write a MATLAB function called `VV=solveRC(R,C,omega,Vinput)` which will return the complex amplitudes of the three unknown node voltages in the vector `VV`. The inputs will be the resistance value in Ohms (Ω), the capacitance value in Farads (F), the frequency in rad/s, and the input voltage expressed as a complex amplitude.

(c) Choose the parameters to be $R = 25,000\,\Omega$, $C = 0.1\,\mu$F, and $\omega = 2\pi(60)$ rad/s, and solve the equations for the three voltages. Assume that the input voltage is $V_0(t) = 100\cos(\omega t)$. Since the node voltages are sinusoids, express your answers as three sinusoids, i.e., write down the mathematical formulas.

(d) Make a plot of all four voltages (from the previous part) together, using either one plot or four subplots. Plot three periods of the waveforms, centered at $t = 0$.

(e) In the plot, observe that there is an amplitude and phase change when comparing $V_1(t)$ to $V_0(t)$, $V_2(t)$ to $V_1(t)$, and $V_3(t)$ to $V_2(t)$. If you concentrate on the phase changes, then you can say that the voltage is being delayed by a small amount of time from one ladder section to the next (each R-C pair is one section in the ladder). Measure these three *relative* time delays directly from the plots.

(f) For comparison to the measured time delays, calculate (mathematically) the expected time delays from the phase changes. Comment on the comparison.

(g) An expert in circuits would be able to say that at very low frequencies, the output voltage should be nearly equal to the input voltage (because the capacitors act like open circuits). To test this idea, use the same parameters for R and C, i.e., $R = 25,000\,\Omega$, and $C = 0.1\,\mu$F, but lower the frequency of the input voltage, i.e., change ω in $V_0(t) = 100\cos(\omega t)$. Run two more cases with $\omega = 2\pi(10)$ rad/s and $\omega = 2\pi(2)$ rad/s, solving the equations for the three voltages. Make a table that summarizes the complex amplitudes found for these two cases, as well as result from part (c). Comment on how the output voltage, $V_3(t)$, is changing as the frequency (ω) is lowered. Compare the output voltage to the input voltage. If necessary, run other cases with extremely low frequencies to support your discussion, e.g., $\omega = 2\pi(0.0001)$ rad/s.

Lab #2
ECE-2025 Spring-2011
INSTRUCTOR VERIFICATION SHEET

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

Name: _____ Date of Lab: _____

Part 3 Show that you can run cell mode using the code in Section 2.4:

Verified:_____ Date/Time:_____

Part 3.1 Replace the inner `for` loop with only one or two 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 cell-mode capability to create and save the plots when doing the Instructor Verifications. Show the web pages to your TA. For example, show the plot made by `addManySines.m`.

Verified:_____ Date/Time:_____

Part 3.3.1 Show that your `addManySines.m` function is correct by running the test in Section 3.3.1 and plotting the result. Measure the period of signal in the structure `ssOut`, and explain its relationship to the fundamental frequency.

Verified:_____ Date/Time:_____