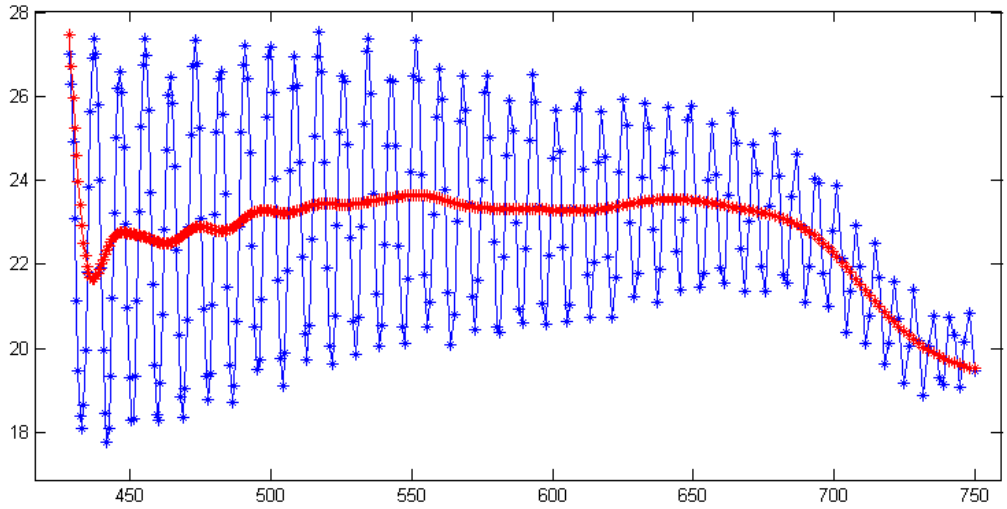
Class #15: Experiment
Introduction to Matlab



Purpose: The objective of this experiment is to begin to use Matlab in our analysis of signals, circuits, etc.

Background: Before doing this experiment, students should be able to

- Analyze simple circuits consisting of combinations of resistors, especially voltage dividers.
- Do a transient (time dependent) simulation of circuits using LTspice
- Determine and validate mathematical expressions for steady-state sine and cosine voltage and current waves including amplitude, frequency and phase.
- Use the mathematical expressions that describe experimental and simulated voltage and current signals to validate and understand the relationships between voltage and current for inductors and capacitors.
- Determine complex impedance for R, L and C
- Analyze RC and RL circuits using phasor analysis and approaches applicable to standard voltage dividers.
- Convert phasor voltages and currents back to time dependent forms.
- Review the background for the previous experiments.

Learning Outcomes: Students will be able to

- Perform basic mathematical operations with Matlab
- Plot basic functions (e.g. sinusoidal, exponential, polynomials) using Matlab
- Plot experimental or simulated data with Matlab

Resources Required:

- LTspice
- Matlab with activation for RPI students

Helpful links for this experiment can be found on the course website under Class #15.

Pre-Lab

Required Reading: Before beginning the lab, at least one team member must read over and be generally acquainted with this document and the other **required reading** materials.
Hand-Drawn Circuit Diagrams: Before beginning the lab, hand-drawn circuit diagrams must be prepared for all circuits either to be analyzed using LTspice or physically built and characterized using Analog Discovery.
Due: At the beginning of Class #17

Part A – Starting Matlab and Using Basic Math Operations

Background

To prepare for the following experiments, review the material in the videos listed under Class #15 and under Matlab in the course website

Experiment

We begin with a series of simple operations listed in the table below. Perform each operation and fill in the remaining cells in the table. Note that some operations work as written while others must follow Matlab syntax. Note that the last two rows are left blank for you to try other operations you can think of.

Operation	Example	Answer
Sum	$333+614+218$	
Difference	$333-292$	
Multiplication	333×292	
Division	$333/29$	
Raising to a Power	55^3	
Roots	$\sqrt{419}$	
Round	$round(456.79)$	
Defining a Variable	$y=67$	
Operating on Variable	$\sqrt{y}$	
What is π?	π	
Sines & Cosines	$\sin(pi/3)$	
ln Function	$\ln(4)$	
Natural Number e	e	
Inverse Tangent	$\tan^{-1}(1)$	

Table A-1

Getting Help

The easiest way to get help with some operation in Matlab is to type help. For example, to learn about the use of the tangent function, type 'help tan' and you will see something like what follows. Note that this immediately tells us that trig functions assume arguments in radians, not degrees.

```
>> help tan
 tan    Tangent of argument in radians.
    tan(X) is the tangent of the elements of X.
     See also atan, tand, atan2.
    Overloaded methods:
       codistributed/tan
    Reference page in Help browser
       doc tan
```

If you do not know the exact term that Matlab uses, you can first search online for something like 'tangent in Matlab' and you can click on the question mark symbol at the top of the Matlab screen and browse or search.

Summary

Many Matlab operations look the same as we usually see them in mathematical expressions, but others must follow specific syntax. Also, some constants are known and their symbols reserved, but others must be calculated.

Part B – What Can Matlab Do?

Before we learn how Matlab can do many useful things for us, we want to get a sense of what it can do by running some demos. There are some good demos that are no longer available from Matlab (in the version we are using – 2015). Go to the course website and click on Matlab. Then select the link to Demos. Select xpsound.m and save it to your m-file directory. This is called 'Visualizing Sound.' Briefly play with this demo without annoying nearby students. You might want to use earbuds to hear better.

Figure B-1

Another interesting demo is called penny.m or 'Viewing a Penny.' When you type the word 'penny' in the command line, pictures will begin to appear. You can cycle from one to the next by hitting any key. Note the great resolution that is possible from scanned info. This demo is discussed further at http://blogs.mathworks.com/cleve/2014/03/17/higher-resolution-penny/.

You can also reach some other examples through Matlab Help, as shown in Figure B-2 below.

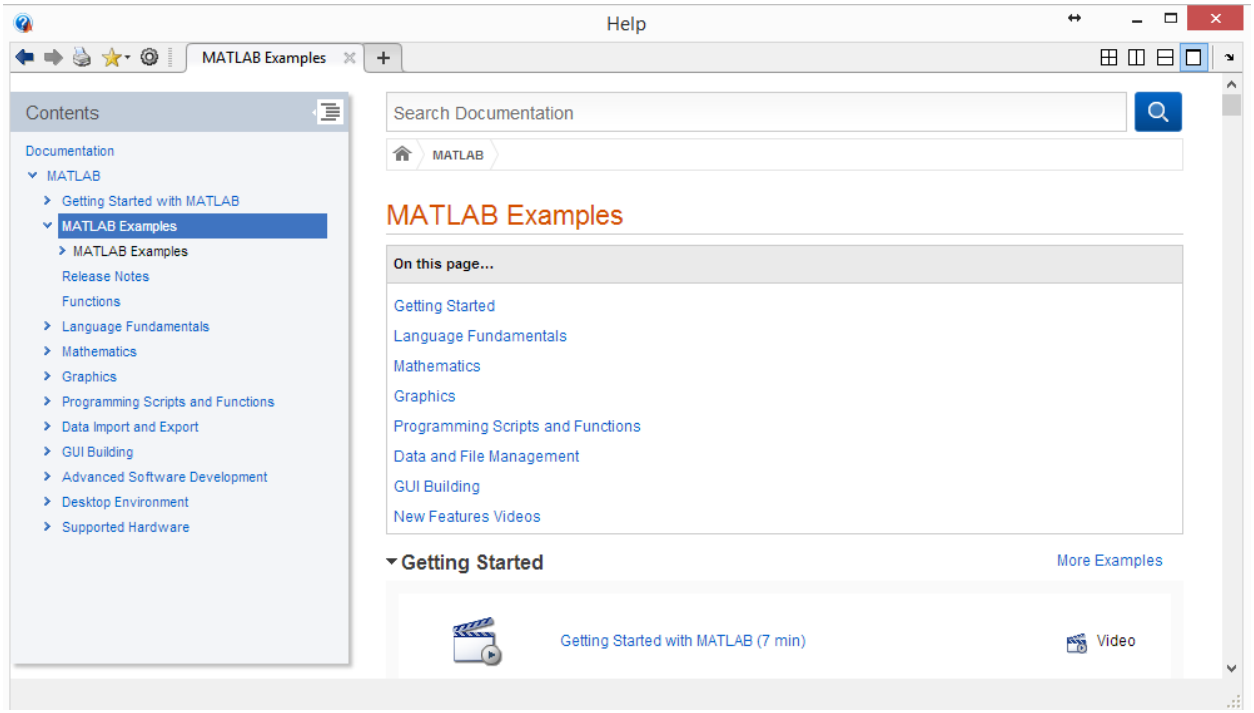


Figure B-2

Scan through the remainder of the examples and try a few more out. What were the ones you enjoyed the most? Which ones look like they would be most useful for you as an electrical or computer engineer? Be sure to ask questions if you have any problems or are just curious about what the demos is actually doing. We may or may not know the answer because the range of Matlab applications is so broad.

Part C – Vectors and Arrays

Recall that when we obtain data from measurements or simulation, we produce arrays of numbers. For example, columns can be time, voltage, current, voltage difference … and the number of rows can be huge (many thousands). The 'Mat' in Matlab stands for Matrix, so it is ideally suited to work with the arrays of numbers we produce.

Experiment
There are many ways to define a one-dimensional array, also known as a vector.

Define the vector R = (0,1,2,3,4,5,6,7,8,9,10) with the following command:

```
>>  R=[0 1 2 3 4 5 6 7 8 9 10]
```

Matlab will immediately tell you what it thinks you told it to do. If you do not want to see it repeat everything, put a semi-colon at the end of the line. You now have an array of 11 numbers called R. Do the following operations on R. Note that if Matlab thinks you did something wrong, it will give you an error message. Pay close attention to the error messages.

Operation on R	Result
sum(R)	
mean(R)	
median(R)	
min(R)	
max(R)	
S=5R	
V=R+S	
T=R^2	

Table C-1

We can also define arrays of numbers. Try the following and see what Matlab thinks you told it to do.

```
>> U=[0 1 2;3 4 5;6 7 8]
```

Helpful hint. Matlab has a great feature where you can repeat or modify previous commands without retyping them. Use the up arrow to cycle through all of the previous commands. Once you find the one you are looking for, you can modify it and use it again. If you know the beginning of the command, type the first letter or two and it will only find commands that start with what you typed. Try this and see what happens.

Much of the work we will do will involve plotting function of some variable. Variables are best written in a uniform array with lots of terms in it. To do this, we need a way to write such arrays easily. For example, to have an array of 101 time values from 0s to 1s, we can write the following:

```
>> t=[0:.01:1];
```

If we want 1001 values from 0ms to 3ms (three periods of a 1kHz sine wave), we can write:

```
>> t=1e-3*[0:.001:1];
```

Note that we have multiplied each term by 0.001 written in scientific notation. Again, it is important to make sure that Matlab understands what we are asking it to do. You can check one of the elements in the array and see if it is correct by typing t(22) for example, which should give you the 22nd time element.

Summary

We have now seen how arrays of numbers are written and operated on, at least somewhat, in Matlab. For each of the operations above, be sure you understand what is being done.

Part D – Plotting

Next, we want to try to plot some standard functions. We will use the time array you just defined.

First we will define the function $V = 10\cos(\omega t + \vartheta)$ using a frequency *f = 1kHz*. There are no Greek letters in Matlab, so we will use w for omega and Th for theta. All parameters have to be defined before they are used anywhere else.

```
>> f=1e3;
>> w=2*pi*f;
>> Th=0;
```

Next, we will do the function.

```
>> V=10*sin(w*t+Th);
```

And plot V (y axis) vs t (x axis).

```
>> plot(t,V);
```

The plot you will produce will look rather boring, with no grid or labels. Ask help to show you how to fix that. Once it gives you the info you need, add grid lines, x and y labels, a title, change the color of the cosine wave and its thickness … make it as interesting as you can.

```
>> help plot
```

Do each of the plots shown below. For the plots involving exponentials, you will need a value for the time constant τ. Assume that it is equal to the period of the cosine wave. Again we need to use something to represent τ = tau.

```
>> tau=1/f;
```

For the current (4th case) assume its amplitude is equal to the voltage so that they both are the same size on the plot. Do at least two cases with $\vartheta_I = \pm\pi/2$. You should try others to see how the phase affects the signal. Save all plots and fully annotate them in your report. For the 5th entry in the table, you decide what to plot.

Function	Plotted?
$V_o \cos(\omega t+\vartheta)$	
$V_o e^{-t/\tau}$	
$V_o e^{-t/\tau} \cos(\omega t+\vartheta)$	
$V_o \cos(\omega t+\vartheta_V)$ & $I_o \cos(\omega t+\vartheta_I)$	

Table D-1

Part E – Scripts

Since most of the things we do with Matlab are repeated over and over, it is a good idea to write scripts (m-files) to save them for the future. For any of the plots you just completed, create such a script. To open the script editor, go to the file menu and select 'New' then 'Script.'An example script that plots a simple polynomial function is shown below. Note that lots of comments are included (lines that start with %) to help understand what the script does. Invariably, you will forget the next time you need to use it, so add lots of comments. You will have to choose a place to save your scripts for this course. Once you have the directory selected, then save your script with a name

you can remember easily. Also, Matlab is case sensitive, so you will have to type the name correctly or it will not know what you mean. Once you save your script, you should see it listed in the current folder window.

Aside: What information is being provided by the various windows usually opened by Matlab?

Example Script

```
% Plotting the Function 1-x^2
% K. A. Connor 22 October 2015

% The x array is chosen so that the function will go from 1 to zero
x=[0:.001:1];

% The function (note the .^2 to square each element in the vector)
y=1-x.^2;
z=x.^2;

% Plotting
plot(x,y,x,z);
grid;
xlabel('x');
ylabel('y');
title('Plotting Example');
legend('1-x^2','x^2','Location','West');% Note that the dot is not necessary
here
axis([0 1 0 1]);
```

Add Averaging to Script

In your script, add a calculation of the average of your function and plot it along with the other signals you have plotted. What information is the average giving you? What Matlab function is used to find the average?

Part F – Task Checklist

The following list summaries the tasks from Parts A, B, C, D and E

- Fill in the table of mathematical operations
- Try out a few demos
- Define arrays and fill in the table of array operations
- Complete the plots in the plotting table.
- Write at least one script and then run it.