Electric Circuits I
Laboratory 1 – Introduction to MATLAB

To be completed independently during the 2nd week of class and submitted at the beginning of the next laboratory meeting

Objective:

- In this laboratory, you will explore the basics of MATLAB, a tool for computation and visualization.
- You will do simple calculations and plot sinusoidal and exponential signals.
- You will learn how to represent and use vectors and matrices in MATLAB.
- You will learn how to use the symbolic toolbox to solve a system of linear equations.

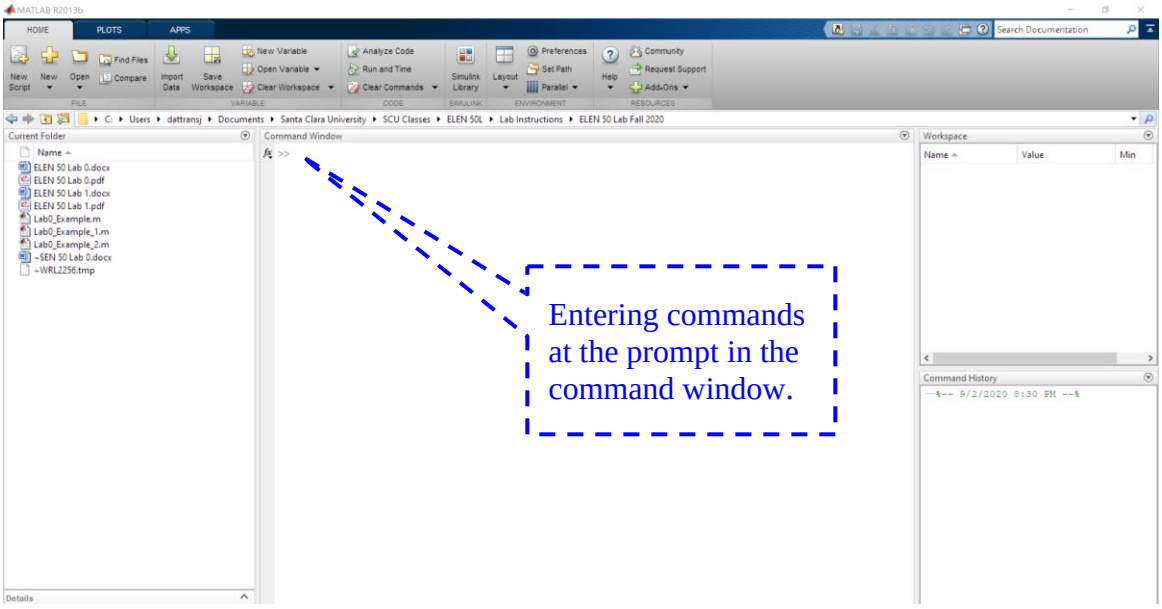
A. Background notes:

MATLAB is a versatile tool that can be used as a simple graphing calculator or a more sophisticated tool for computation and comprehensive graphical displays. We will use MATLAB for homework assignments and laboratory work in this lab.

Note that MATLAB is licensed software. An alternative to MATLAB is Octave or Python with SciPy/NumPy package. Octave is free software for various operating systems such as Linux, Windows, or macOS. Octave uses the same commands as those in MATLAB. Check the public version of Octave (https://www.gnu.org/software/octave).

B. Starting MATLAB:

In the design center, log on to a system that supports MATLAB (the teaching assistant should be able to direct you) and start MATLAB. You should see a screen that has a Command Window and possibly several other windows, such as Command History and Workspace. In the dropdown menu under "Desktop," you can select windows and layouts.



The command window allows you to type in mathematical expressions and calculate results similar to a calculator. Under the "Help" dropdown menu, select "Using the Command window." A new screen will provide a list of topics under Running Functions — Command Window and History." Select "The Command Window" and read it.

Use "Help" when you have other questions about MATLAB.

C. Simple calculator operations:

In the Command Window, type the following after the >> prompt:

```
>> 1.5*7
```

You should see the product printed as

```
>> ans = 10.5000
```

and in the workspace window, you should see that a variable named "ans" has been created with a value of 10.5.

Parentheses can be used for more complicated computations. Try typing the following expression following the prompt.

```
>>  (3*15+8*9)/11
```

You should get an answer of 10.6364.

Suppose you mistyped the expression, and the 9 should have been 19. It is not necessary to retype everything to get the correct result. In the command window, use the "up arrow" key on your keyboard to reprint the previous line, which is the incorrect expression. You can now use the left and right arrows to move the cursor position and insert or delete parts of what you typed previously. Use the "left arrow" key on your keyboard to move the cursor to where the "1" should have been. Enter the "1" in the expression and hit "enter." Now you should see an answer of 17.9091.

Work to be submitted 1:

- Using MATLAB, calculate the term grade point average of a student who took three courses and received an A- (worth 3.7) in a 4 unit course, a B (worth 3.0) in a 5 unit course, and a C+ (worth 2.3) in a 3 unit course.
- Copy the expression on the screen and the results, and paste that into the document for your lab submission.

D. Some MATLAB functions:

In this class, we will be using sinusoidal and exponential functions.

Exponential function: For the exponential function e^t use the MATLAB function exp. After the prompt, type:

```
>> exp(0)
```

You should see the response

```
>> ans = 1
```

Type exp(-1) and then exp(-5). You should see answers of 0.3679 and 0.0067.

It is possible to compute several values of the exponential function simultaneously by using a list of values for the function argument. The list, or vector of values, must be typed inside a set of square brackets. Type the following after the prompt:

```
>> exp ([0, -1, -2, -3, -4, -5])
```

You should see the following result.

```
>> ans = 1.0000    0.3679 0.1353   0.0498    0.0183 0.0067
```

In the workspace window, you should see that "ans' is now the list of the 6 values you have just computed. Alternatively, you can create a vector of values and assign a variable name to it so that a variable name can be used instead of the definitive list of vector values. Type the following and note the result in the command window and the workspace window. The computed result should be the same as the previously calculated list.

```
>> t= [0, -1, -2, -3, -4, -5];
>> exp(t)
```

Cosine function: Type the following to create a vector of angle values:

```
>> a = pi*[0:5]
```

You should see

```
>> a = 0    3.1416 6.2832 9.4248  12.5664  15.7080
```

MATLAB knows that π (pi) is 3.14159…, so you can use the name pi instead of explicitly entering the value of π. Also, note that the expression 0:5 creates a list in integers from 0 to 5, and the vector a has elements with values [0 π 2π 3π 4π 5π]. In addition, note that when you define a vector without ending the statement with a semicolon, MATLAB lists all the values in the vector. The cosine function is defined for argument values in radians. Type the following:

```
>> cos(a)
```

Now type the following:

```
>> a=pi*0.5*[0:5];
>> cos(a)
```

The result should be six values of the cosine function evaluated from a = 0 to a = 2.5pi at intervals of pi/2.

Work to be submitted 2:

- Using MATLAB, calculate 20 values of the cosine function evenly distributed over one cycle or period of the function.
- Copy the expression on the screen and the results, and paste that into the document for your lab submission.

E. Simple plots:

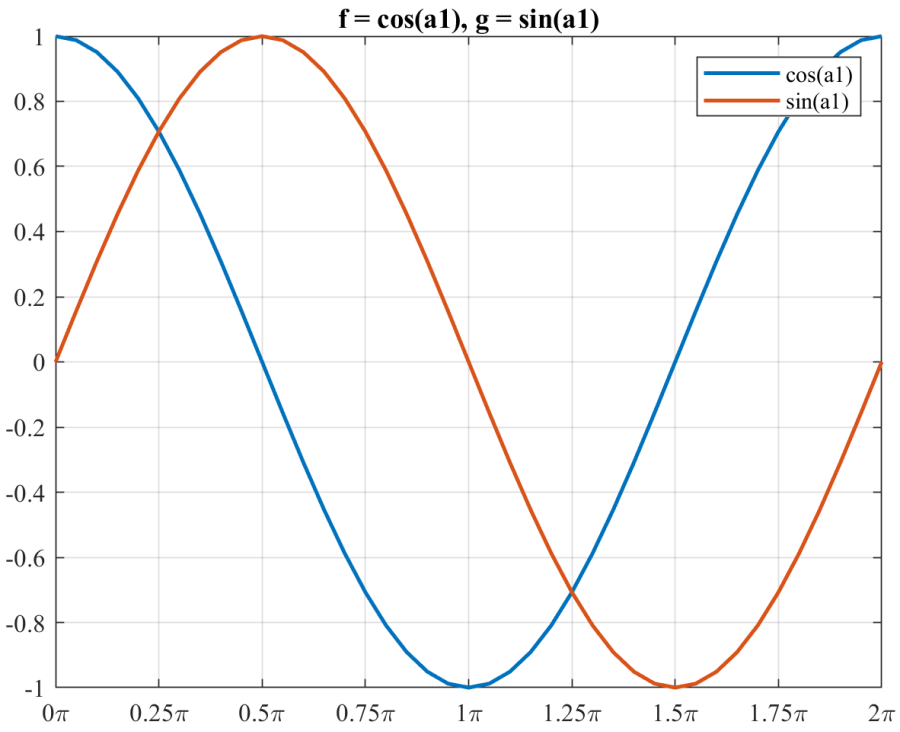
MATLAB can create plots. When you ask for a plot, a figure window is created showing your plot. Type the following instructions. Don't forget the semicolon.

```
>> a1=pi*[0:0.01:2];
>> plot (a1, cos(a1), a1,sin(a1), linewidth=1.5);
```

In the figure window, you should see a plot below the cosine and the sine in angle values from

0 to 2π with a timestep of 0.01π. The following commands format the plots and label the a-axis with values of π:

```
>> title('f = cos(a1), g = sin(a1)');
>> xlim([0, 2*pi]);
>> xticks(pi*[0:0.25:2]);
>> Labels = string([0:0.25:2]) + "\pi";
>> xticklabels(Labels);
>> set(findall(gca, '-Property', 'FontName'), 'FontName', 'Times New Roman');
>> legend("cos(a1)", "sin(a1)");
```



Under "file" in the figure window, select "save as," and then, for "save as type," select .png or .jpg. Choose a name for the plot and save it. You can then insert this .png image file into a document, such as a lab report. In the command window, type "help plot" after the >> prompt. You will see a description of many other plot options.

Work to be submitted 3:

- Given the following functions:

$$f(t) = e^{-t} \text{ and } g(a) = \cos(a)$$

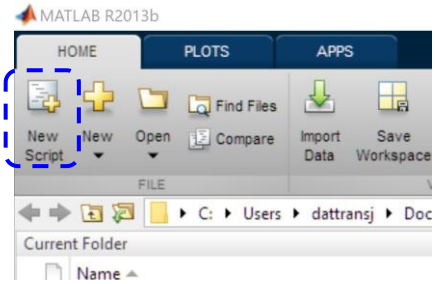
Note: e^{-t} *is encoded as* $exp(-t)$.

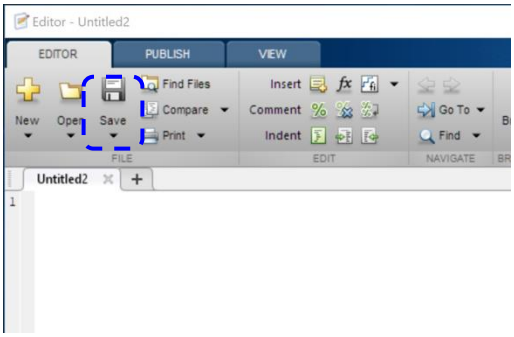
- Plot *f*(*t*) from *t* = 0 to *t* =100 using time steps of 0.01. Save the plot and insert the screen instruction, and plot into the document for your lab submission.
- Plot g(*a*) from *a* = 0 to *a* = 12π using time steps of 0.05π. Save the plot and insert the screen instruction, and plot into the document for your lab submission.

F. MATLAB Command Scripts

Instead of entering a command each time at the prompt >>, MATLAB (or Octave) offers a command script, in which commands are saved into a command script file (.m), to run many

commands sequentially.
Select “New Script” from the “HOME” tab, and a new script editor will pop up:

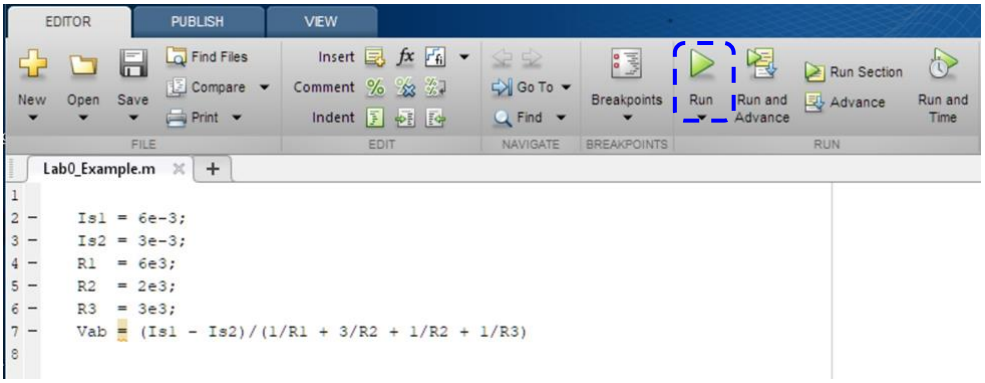




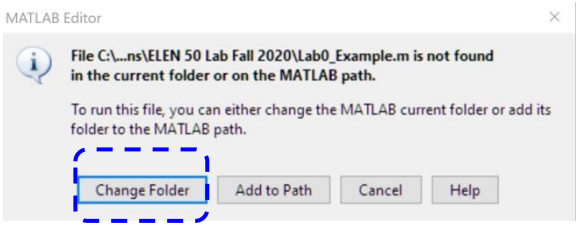
Click on “Save” and save the command script file (e.g., Lab1_Example.m). Once the script file is saved, the editor window’s title will change to the name of the script file. Enter the following commands into the script file and save it.

```
clear all;
Is1 = 6e-3;
Is2 = 3e-3;
R1  = 6e3;
R2  = 2e3;
R3  = 3e3;
Vab = (Is1 - Is2)/(1/R1 + 3/R2 + 1/R2 + 1/R3)
```

To run the script file, click on “Run”:



Note: *if the script is run the first time, a small window will pop up and select “Change Folder)*



If there is no error, the result will be displayed on the command window:

```
>> Lab1_Example_1

Vab =

    1.2000

fx >>
```

Create another script file and enter the following commands:

```
Vs1 = 6;
Is  = 2e-3;
Vo  = 4;
R1  = 3e3;
R2  = 2e3;
R3  = 3e3;
R4  = 12e3;
R5  = 1e3;
 % ************************************************************
% system of equations
% ************************************************************
IR5 = Vo/R5;
IR1 = -Is + IR5;
Vad = -Vs1 + R1*IR1 + R2*IR5 + Vo;
IR3 = -Vad/R4 + Is - IR5;
Vs  = -R3*IR3 + Vad
```

If there is no error in the command script, the result will be:

```
>> Lab0_Example_2

Vs =

    21

fx >>
```

G. Calculus Functions:

MATLAB offers many useful calculus functions, such as integration and derivative. Supposed that we have a function *f*():

$$f(x) = \frac{e^{2x^2} - 8\sin(x+3) + 5^{x+1}}{20\cos(5x+2)}$$

1. Derivative Function

- Create a new script. Be sure to add the clear and close commands to clear all variables and close all display figures at the beginning of the script.

```
clear all;
```

```
close all;
```

- Enter the function into the MATLAB script:

```
syms x
f = (exp(2*x^2) - 8*sin(x+3) + 5^(x+1))/(20 * cos(5*x+2))
```

Note: make sure that the expression has matching parentheses.

- Find the derivative of the function type in the command and click on "Run."

```
Df1 = diff(f, x)
```

- If the express is entered correctly, the result will show:

```
Df1 = (5^(x + 1)*log(5) - 8*cos(x + 3) + 4*x*exp(2*x^2))/(20*cos(5*x + 2)) +
(sin(5*x + 2)*(exp(2*x^2) - 8*sin(x + 3) + 5^(x + 1)))/(4*cos(5*x + 2)^2)
```

- If we want to express it in a more readable format, type in:

```
pretty(Df1)
```

2. Integration Function

Supposedly, we have a function *f*():

$$g(t) = e^{-4t}.$$

We want to integrat3 the function *g*() for an interval of *t* [0, 10].

- Create a new script. Be sure to add the clear and close commands to clear all variables and close all display figures at the beginning of the script:

```
clear all;
close all;
```

- Type the command and hit "Run":

```
syms t
g = exp(-4*t)
If1 = int(g, t, [0, 10])
```

- The answer will show: If1 = 1/4 - exp(-40)/4
- To calculate out a value, change the command: If1 = double (int(g, t, [0, 10]))
- The answer will be: If1 = 0.2500

Work to be submitted 4:

$$f(x) = \frac{e^{-3x} + 10\sin(x+1)}{\cos(x+1)} \text{ and } g(x) = \frac{e^{-2x}}{1+e^{-2x}}$$

- Find the derivatives of functions *f*() and *g*()
- Find the integration of the following function *h*(x) with x in the interval [-2, 2]:

$$h(x) = -4x^2$$

H. Matrix Operations

- In MATLAB, an **M**×**N** matrix is a rectangular array of numbers with **M** rows and **N** columns. It can be defined by typing the matrix elements between square brackets. Commas separate elements along a row and rows are separated by semicolons.

Work to be submitted 5:

- Create 2 x 2 matrices A and B as follows:

  A = [2,1; 3,2]
  B = [3,1; 2,2]

- Print A' and B', the transposes of these two matrices.

Matrix Multiplication:

- Compute the following 4 matrix products and print them. Are any the same? Which ones?

  A1 = A * B,
  A2 = B * A
  A3 = (A'* B')'
  A4 = (B' * A')'

Matrix Inverses:

- Use "inv" to compute the following matrix inverses:

  A5 = inv (A * B)
  A6 = inv(A) * inv(B)
  A7 = inv (B * A)
  A8 = inv(B) * inv(A)

  Check the inverse values (*checking the inverse values means using one of the variables in A1 – A4 to multiply by their corresponding inverted matrix in A5 – A8. You should expect the result to be the "Identity Matrix"*).

- What are the two products?

  A1*(A*B)
  (A*B)*A1

I. Solving Circuits with MATLAB:

The result of a KVL/KCL analysis of a circuit is the set of simultaneous equations:

$$\begin{aligned} V_1 \quad\quad\quad + V_3 &= 10 \\ 3V_1 + 3V_2 + 4V_3 &= 12 \\ 2V_1 + 2V_2 + 3V_3 &= 5 \end{aligned}$$

Which can be written using matrices as follows:

$$C * V = S \Leftrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 3 & 3 & 4 \\ 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 12 \\ 5 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 1 \\ 3 & 3 & 4 \\ 2 & 2 & 3 \end{bmatrix}, \quad V = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}, \quad S = \begin{bmatrix} 10 \\ 12 \\ 5 \end{bmatrix}$$

In solving this 3x3 system of equations, we invert the coefficient matrix ***C*** and multiply it by the source matrix ***S***.

Work to be submitted 6:

- Invert the matrix ***C*** and solve the system for the voltage matrix ***V*** = **inv(*C*)*** ***S***.
- Then verify that the result is correct by multiplying ***C********V*** and comparing the result with ***S***.

J. Solving a System of Linear Equations with Symbolic Toolbox

MATLAB or Octave has a toolbox to simplify symbolically equations. As shown in the previous section, solving a linear equation system requires that the equations' coefficients are simplified and arranged in the order of the unknown (V_1, V_2, V_3, …, V_4). Once equations are simplified and set in order of the unknown, we can generate coefficient matrix ***C*** and source vector ***S*** to solve for the unknown. The simplifying process of coefficients, however, can be tedious, with many chances of errors.

Here, the symbolic toolbox performs the simplifying process and generate automatically coefficient matrix ***C*** and source vector ***S***. Consider the following equations:

$$\frac{V_a - V_b}{R_1} - I_{s1} + \frac{V_a - V_d}{R_2} - I_{s3} = 0,$$

$$\frac{V_b - V_a}{R_1} + I_{s1} + \frac{V_d - V_a}{R_2} + \frac{V_b}{R_3} + I_{s2} + \frac{V_d}{R_4} = 0,$$

$$V_b - V_c = V_{s1},$$

$$V_d - V_c = V_{s2},$$

where v_a, v_b, v_c and v_d are unknown variables and others are known with values: V_{s1}= 6V, V_{s2} = 12V, I_{s1} = 2mA, I_{s2} = 4mA, I_{s3} = 6mA, R_1 = 1kΩ, R_2 = 1kΩ, R_3 = 2kΩ, and R_4 = 1kΩ

- Create a new command script file (.m). Enter the values for known variables:

```
clear all;
close all;

Vs1 = 6;
Vs2 = 12;
Is1 = 2e-3;
…
R1 = 1e3;
…;
```

- Define unknown symbolic variables:

```
syms Va Vb Vc Vd
```

- Enter equations above:

```
eq1 = (Va-Vb)/R1 – Is1 + (Va-Vd)/R2 – Is3 == 0;
eq2 = … == 0;
eq3 = … == Vs1;
eq4 = … == Vs2;
```

- Enter equations and variables:

```
eqns = [eq1, eq2, eq3, eq4];
vars = [Va, Vb, Vc, Vd];
```

- Use MATLAB or Octave functions to simplify coefficients and generate coefficient matrix ***C*** and source vector ***S****:*

```
[C, S] = equationsToMatrix(eqns, vars)
```

- Use the coefficient matrix ***C*** and the source vector ***S*** to solve for the unknown variables:

```
V = inv(C) * S
```

- Save the command script file and run it. If there is no error in the command script, the final results will be as follows. Compare your results with the results below

```
C =                                          S =       V =

[  1/500, -1/1000,   0, -1/1000]    1/125     13/3
[ -1/500,  3/2000,   0,  1/500]    -3/500     -8/3
[      0,       1, -1,       0]         6    -26/3
[      0,       0, -1,       1]        12     10/3
```

According to the order of the vars = [Va, Vb, Vc, Vd],

$$V_a = V(1) = \frac{13}{3};\ V_b = V(2) = -\frac{8}{3};\ V_c = V(3) = -\frac{26}{3};\ V_d = V(4) = \frac{10}{3}$$

- Create another command script file (.m) and use similar commands to solve a system of equations below:

$$-I_{s1} + \frac{V_a - V_b}{R_1} + \frac{V_d - V_c}{R_2} + \frac{V_d}{R_5} + \frac{V_d}{R_3 + R_6} = 0,$$
$$\frac{V_b - V_a}{R_1} + I_{s2} + \frac{V_c}{R_4} + \frac{V_c - V_d}{R_2} = 0,$$
$$V_c - V_b = V_s,$$
$$V_d - V_a = 2V_c,$$

where v_a, v_b, v_c, and v_d are unknown variables and others are known with values: V_s = 12V, I_{s1} = 4mA, I_{s2} = 2mA, R_1 = 1kΩ, R_2 = 1kΩ, R_3 = 1kΩ, R_4 = 1kΩ, R_5 = 1kΩ, and R_6 = 1kΩ.

Work to be submitted 7:

- Copy the MATLAB script (.m) for the system of equations above.
- Compare your results with the expected results:

```
C =                                                  S =       V =

[  1/1000, -1/1000, -1/1000,   1/400]    1/250      -4
[ -1/1000,  1/1000,  1/500, -1/1000]    -1/500     -10
[       0,      -1,      1,       0]        12       2
[      -1,       0,     -2,       1]         0       0
```

Make sure to check off with the TA before leaving the lab section.

K. Laboratory Report:

For your laboratory report, submit the document with the requested computations and plots. The template of the laboratory section is below:

Name: …
Lab Number: …
Class: ELEN 50L
Date: …
Section time: …

C. Simple calculator operations:

…

D. Some MATLAB functions:

…