ECE90L – Instructions for Worksheets and Plotting

An Excel *workbook* is a file containing one or more *worksheets* (spreadsheets). You will be provided with an Excel workbook for each experiment. In some cases, you will be asked to plot data from a worksheet. It is recommended that you use MATLAB for plotting data.

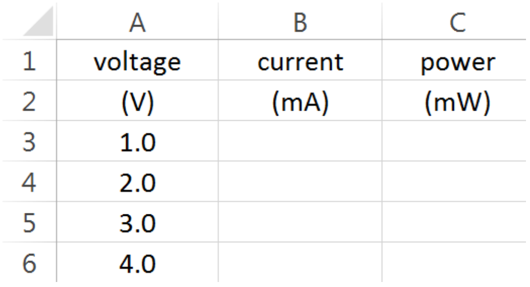
This set of notes first discusses worksheets and how to copy a worksheet into a Word document (your lab report). Then plotting in MATLAB is discussed, including how to import data from a worksheet into MATLAB.

1. Worksheets

In addition to recording your data in your lab notebook during the laboratory session, you should enter data in these worksheets (perhaps outside of the laboratory). Not all data that you take during the laboratory session (and that you record in your lab notebook) needs to be entered into a worksheet. By examining each worksheet, you will figure out what data should be entered.

Figure 1 is an example of a worksheet that might appear in the workbook for some experiment. This is how the worksheet appears before you have entered data into it. This worksheet suggests that you have set the voltage to each of the following four values: 1.0, 2.0, 3.0 and 4.0 V. For each of these voltages, you presumably measured a corresponding current in units of mA. The current data should be entered in column B. For each voltage/current pair, a power is to be calculated; the results of those calculations will appear in column C in units of mW.

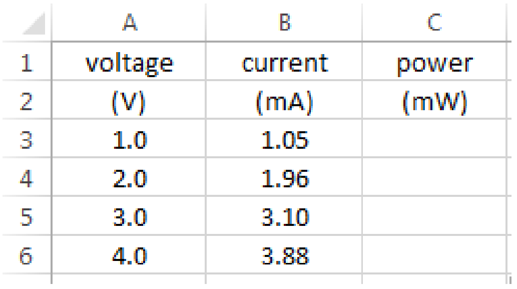
Figure 1: Example worksheet (as provided, before completion by student)



	A	B	C
1	voltage	current	power
2	(V)	(mA)	(mW)
3	1.0		
4	2.0		
5	3.0		
6	4.0		

Figure 2 shows how the worksheet of Figure 1 might look after data entry by the student. Column B contains the data for the current in mA. You will need to decide what precision should be used for representing the numbers in each column. You can change the number of decimal places as follows. Select (highlight) a column of numbers, right-click and from the menu select **`Format cells`**. In the dialog box that appears, under the **`Number`** tab, select the category **`Number`**. Here you will have the opportunity to specify the number of decimal places.

Figure 2: Current measurements added to worksheet

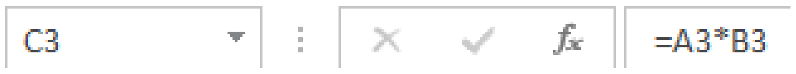


	A	B	C
1	voltage	current	power
2	(V)	(mA)	(mW)
3	1.0	1.05	
4	2.0	1.96	
5	3.0	3.10	
6	4.0	3.88	

We want column C to contain calculated powers. In the uncompleted worksheet of Figure 1, column C was given a heading, so that you know what goes there. However, the formulas associated with the cells in that column are not provided for you. It will be part of your job to determine the correct formulas and to enter those formulas into the worksheet.

Figure 3 shows the formula used for cell C3 in the example of Figure 2. If you need more information about entering formulas into Excel worksheets, see the Excel help pages with the titles "Overview of formulas in Excel" and "Understand and use cell references".

Figure 3: Formula for cell C3

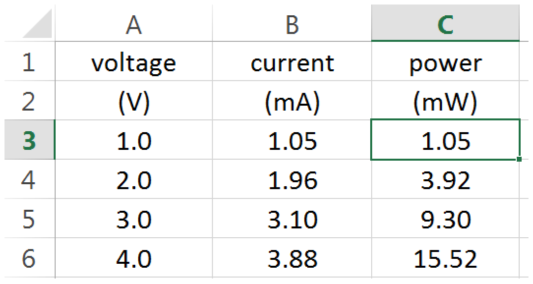


C3

=A3*B3

After formulas have been entered, the completed worksheet looks like Figure 4.

Figure 4: Completed example worksheet



	A	B	C
1	voltage	current	power
2	(V)	(mA)	(mW)
3	1.0	1.05	1.05
4	2.0	1.96	3.92
5	3.0	3.10	9.30
6	4.0	3.88	15.52

You can copy tabular data from a worksheet into a Word document as described here. This is something you will want to do for your lab report. Within the worksheet, select (highlight) a rectangular array of cells using the mouse. Then right-click and select **copy** from the menu. Then go to the Word document and **paste** what you just copied from the worksheet. Figure 5 shows the resulting table of numbers as it will appear in the Word document.

Figure 5: Table of numbers in a Word document, coming from a worksheet

voltage (V)	current (mA)	power (mW)
1.0	1.05	1.05
2.0	1.96	3.92
3.0	3.10	9.30
4.0	3.88	15.52

2. Plotting in MATLAB

Figure 6 lists a MATLAB script that imports data from an Excel spreadsheet and then plots the data, along with a theoretical curve.

Figure 6: MATLAB script **plot_0.m**

```
% data:
A = xlsread('workbook_0.xlsx','resistive circuit');
V = A(:,1);
P = A(:,3);

% theory:
x = linspace(0,4,21);
y = x.^2;

% plotting:
figure(1)
plot(x,y,'-k')
hold on
plot(V,P,'ok','MarkerFaceColor','k')
axis([0 4 0 20])
grid on
set(gca,'FontSize',14)
set(gca,'XTick',0:4)
set(gca,'YTick',0:4:20)
title('Resistive Circuit')
xlabel('voltage (V)')
ylabel('power (mW)')
set(findobj(gcf,'LineWidth',0.5),'LineWidth',2)
legend('theory','data','Location','Northwest')
saveas(1,'resistive circuit','png')
```

The individual lines of code in the script **`plot_0.m`** are described in some detail below.

`% data:`
This is a comment line because of the percent sign (`%`). This comment suggests that the next action will be to import the data.

`A = xlsread('workbook_0.xlsx','resistive circuit');`
The **`xlsread`** function imports data from an Excel worksheet into the MATLAB workspace. The first input argument is a string that names the workbook (the Excel file), and the second input argument is a string that names the worksheet within the workbook. The data are stored in the MATLAB workspace in a matrix named **`A`**.

`V = A(:,1);`
The first column of the matrix **`A`** is assigned to the new variable **`V`**. The colon means "all" in this context. When referring to an element or group of elements in a matrix, the first index refers to the row and the second to the column. So, literally, we are asking for all rows in column 1 (that is, the entire first column).

`P = A(:,3);`
The third column of the matrix **`A`** is assigned to the new variable **`P`**.

`% theory:`
This comment suggests that the next action will be to calculate the theoretical curve.

`x = linspace(0,4,21);`
A vector named **`x`** is created that begins at **`0`**, ends at **`4`**, and has **`21`** uniformly spaced (linearly spaced) values.

`y = x.^2;`
A vector named **`y`** is created. It has the same length as **`x`**. Each element of **`y`** equals the square of the corresponding element of **`x`**.

`% plotting:`
This comment suggests that the next action will be to create a plot.

`figure(1)`
Start a new figure and assign it the figure handle **`1`**. The figure handle is just a number that uniquely identifies a figure.

`plot(x,y,'-k')`
Plot a curve using the elements in the vectors **`x`** and **`y`**. (This is the theoretical curve.) The first element of **`x`** paired with the first element of **`y`** will be treated as the first point on the curve. Element *n* of **`x`** paired with element *n* of **`y`** will be treated as point *n* on the curve. The vectors **`x`** and **`y`** must be of the same length. The curve will be created by connecting straight lines between adjacent points. If there is a sufficient density of points, the curve should look smooth.

The string `'-k'` specifies the appearance of the curve. In this case, a solid (`-`), black (`k`) curve is requested. In the MATLAB documentation, there is a page named **`LineSpec`** that lists a lot more options for the appearance of plotted curves.

```
hold on
```
We want to plot more data on the same set of axes. Here we are telling MATLAB to *add* the data of the next **`plot`** function to this figure. Without this **`hold on`**, MATLAB would *replace* the first curve with the next.

```
plot(V,P,'ok','MarkerFaceColor','k')
```
This call to the **`plot`** function will create a discrete set of points, rather than a curve. The string `'ok'` specifies little circles (`o`) that are black (`k`) to mark the locations of the data points. The horizontal and vertical coordinates of the data points are to be extracted from the vectors **`V`** and **`P`**, respectively. These two vectors must have the same length. The `MarkerFaceColor` property is changed to black (`k`); this means, to put it in hockey terms, that the donuts (circles) will be changed to pucks.

```
axis([0 4 0 20])
```
This specifies the limits on the axes. There is only one input argument: a vector of length 4. The first two elements of this vector request a horizontal axis that begins at **`0`** and ends at **`4`**. The last two elements request a vertical axis that begins at **`0`** and ends at **`20`**.

```
grid on
```
Grid lines are created.

```
set(gca,'FontSize',14)
```
The `FontSize` property is changed to **`14`** points for all graphics objects associated with the current axes. The default font size is 10 points. So this line of code increases the font size of the text, making it easier to read. The first input argument, **`gca`**, asks MATLAB to consider all objects associated with the current axes. (**`gca`** means “get current axes”.)

```
set(gca,'XTick',0:4)
```
The third argument is the vector **`[0,1,2,3,4]`**. A short-hand notation in MATLAB for this vector is **`0:4`**; literally, this means a vector that starts at **`0`**, increments by **`1`**, and ends at **`4`**. (The default increment is **`1`**, so there is no need to state explicitly an increment of **`1`**.) The `XTick` property is changed to the vector **`[0,1,2,3,4]`**. This means that on the horizontal axis the tick marks (with accompanying numeric labels) will appear at the locations **`0`**, **`1`**, **`2`**, **`3`**, and **`4`**.

```
set(gca,'YTick',0:4:20)
```
The third argument is the vector **`[0,4,8,12,16,20]`**. A short-hand notation in MATLAB for this vector is **`0:4:20`**; literally, this means a vector that starts at **`0`**, increments by **`4`**, and ends at **`20`**. The `YTick` property is changed to the vector **`[0,4,8,12,16,20]`**. This means that on the vertical axis the tick marks (with accompanying numeric labels) will appear at the locations **`0`**, **`4`**, **`8`**, **`12`**, **`16`**, and **`20`**.

`title('Resistive Circuit')`
The string `'Resistive Circuit'` is to be the title for this figure.

`xlabel('voltage (V)')`
The string `'voltage (V)'` is to be the horizontal axis label for this figure.

`ylabel('power (mW)')`
The string `'power (mW)'` is to be the vertical axis label for this figure.

`set(findobj(gcf,'LineWidth',0.5),'LineWidth',2)`
This line of code thickens all lines from half-a-point to **`2`** points. Here we have one function (**`findobj`**) nested inside another (**`set`**). The function **`findobj`** returns the handles of all graphics objects within the current figure (**`gcf`** = "get current figure") whose `LineWidth` property equals **`0.5`**. Since **`0.5`** is the default line width, this normally includes all graphics objects, such as the plotted curve and the border of the plot, that have a `LineWidth` property. The set function changes the `LineWidth` property to **`2`** points for all graphics objects returned by the function **`findobj`**.

`legend('theory','data','Location','Northwest')`
The function **`legend`** creates a legend, where the first string, `'theory',` is associated with the first call to the **`plot`** function and the second string, `'data',` is associated with the second call to the **`plot`** function. The `Location` property is changed to `Northwest`. In other words, the legend is placed in the `Northwest` corner (the top-left corner) of the figure.

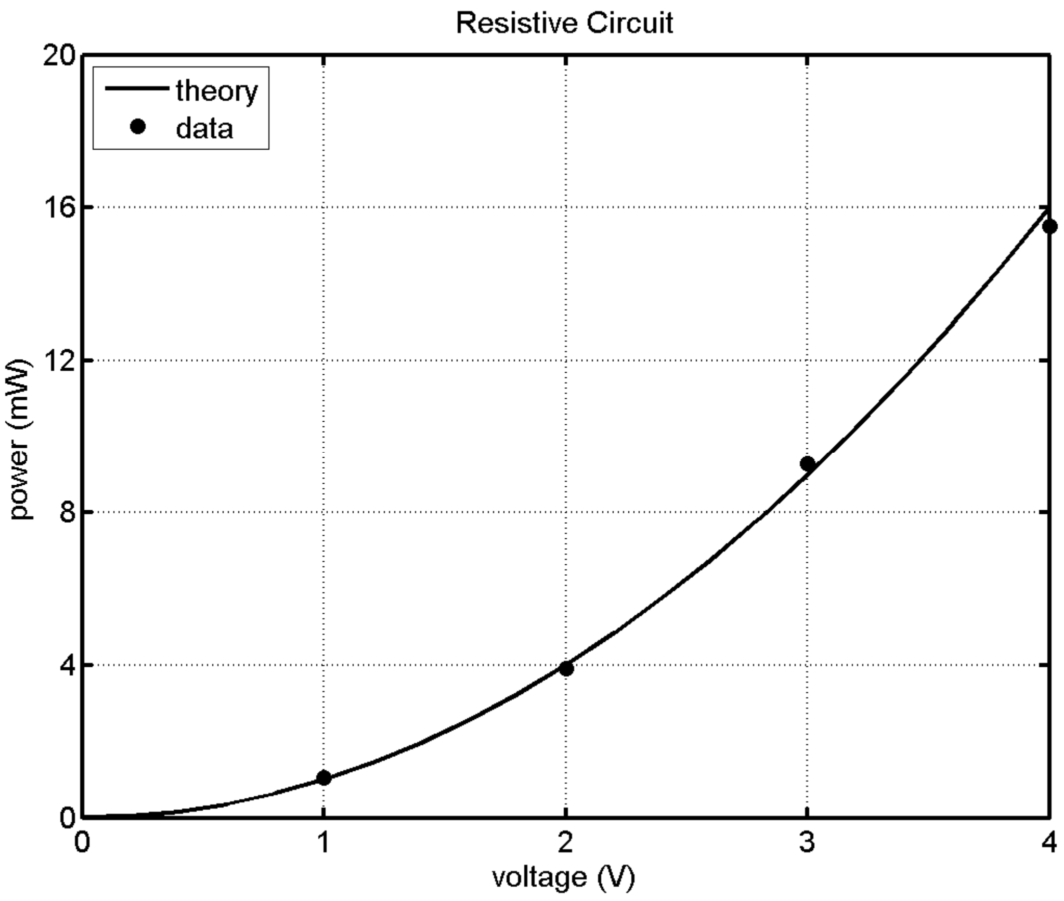
`saveas(1,'resistive circuit','png')`
The figure having figure handle **`1`** is saved to a graphics file of type `png` (Portable Network Graphics). The filename is `resistive circuit.png`. You can save the figure to a JPEG file instead by changing the string `'png'` to `'jpg'`.

Figure 7 shows the plot that is created by the script **`plot_0.m`**.

Not all of the lines of code in the script **`plot_0.m`** are essential. For example, it is not essential to increase the font size or to thicken the lines. The **`axis`** function call is unnecessary; if you omit it, MATLAB will make its own decision about the limits on the axes. It is not essential to identify where the tick marks should go; if you omit these instructions, MATLAB will make its own decisions about tick marks. But if you omit some of these lines of code, you might be unhappy with MATLAB's decisions. In general, it is good to be familiar with the **`axis`**, **`set`**, **`xlabel`**, and **`ylabel`** functions.

Once a graphics file is created, it is easy to incorporate it into a Word document (such as your lab report). In Word, under the **`INSERT`** tab, select **`Pictures`**.

Figure 7: Plot created by the script **`plot_0.m`**


Resistive Circuit
theory
data
power (mW)
voltage (V)