ECE 1050: Lab M2
Arrays, Plots, and the Voltage Divider

Learning Objectives:

- Finish the MATLAB Onramp tutorial.
- Use MATLAB to model a simple voltage divider.
- Create arrays and plots in MATLAB.

Hayden Walpole

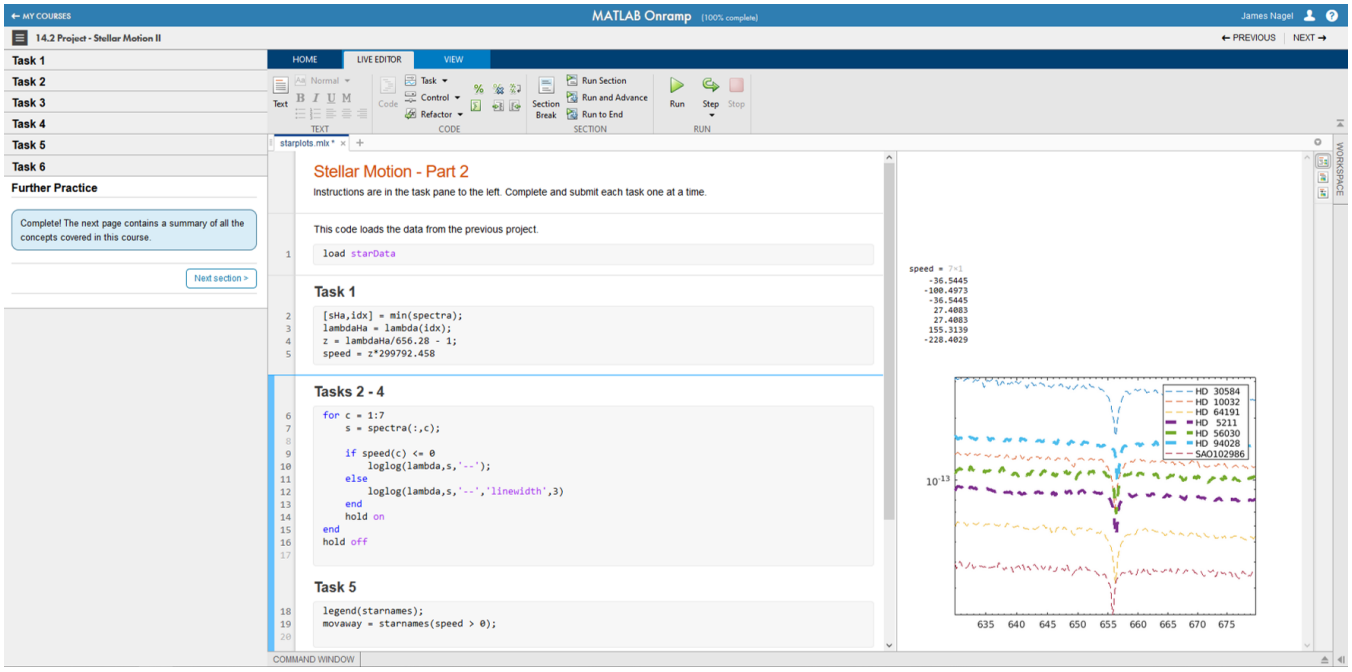
Name: _____

Sept 29, 2023

Date: _____

Part 1: MATLAB Onramp (20 Points)

In the previous assignment, you should have completed Parts 1–5 of the MATLAB Onramp tutorial. For this session, simply complete Parts 6–9. Upon completion, take a screen shot of the webpage. It should look similar to the image below. Include this screen shot in your submission to verify completion of the course.



Name: Hayden Walpole Date: Sept 29, 2023

Part 2: The Voltage Divider (10 points)

When designing circuits, it is common to find yourself in need of a fixed voltage that is some fraction of the main supply voltage. One of the simplest ways to create such a voltage is through the *voltage divider* circuit, which is depicted in Fig. 1. There are also many practical situations where a circuit behaves just like a voltage divider, and so it is important to understand its behavior.

Recall from last time that the equivalent series resistance between R_1 and R_2 is simply

$$R_s = R_1 + R_2 \ .$$

If we now apply Ohm's law, the total series current I_s that flows through the resistors is

$$I_s = V_s/R_s \ .$$

Thus, the output voltage V_{out} satisfies

$$V_{out} = I_s R_2 \ .$$

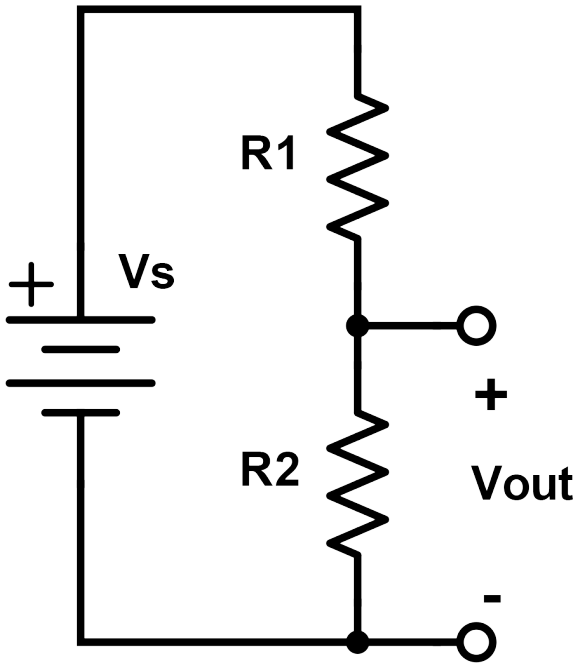


Figure 1: *Basic voltage divider circuit.*

If we now substitute from the previous expressions, we arrive at the voltage divider equation,

$$V_{out} = V_s \frac{R_2}{R_1 + R_2} \ .$$

Create a new script file called *VoltageDivider.m* that models the voltage divider circuit. Remember to include a proper header and to comment your code. Assume that $V_s = 5.0$ V, $R_1 = 1.0$ kΩ, and $R_2 = 1.5$ kΩ. Have your script calculate the series resistance R_s, the series current I_s, and the output voltage V_{out}.

$R_s =$ 2.5 Ω .

$I_s =$ 2 mA .

$V_{out} =$ 3 V .

Name: Hayden Walpole Date: Sept 29, 2023

Part 3: Arrays and Plots (20 points)

We shall now practice creating arrays and then plotting data over a large number of samples. Continuing with the same script you created in Part 2, we shall model the output voltage of a potentiometer (See Lab 1 of ECE 1245), which is an excellent tool for doing voltage division. Start by assuming that the total series resistance is a fixed value of $R_s = 15$ kΩ (you may need to overwrite the Rs variable from the previous section). The resistor R_1 can now vary continuously from zero to R_s, with $R_2 = R_s - R_1$.

A very useful tool in MATLAB is the *linspace* function, which creates a linearly-spaced array of samples between two extremes. To demonstrate, let us model R_1 as an array of 11 samples between $[0, R_s]$ by adding the following command to our script:

```
R1 = linspace(0,Rs,11);
```

As you go through these labs, it is important to bear in mind that MATLAB is a very well documented program. If you ever get happen to forget what *linspace* does or how it works, you can always type "help linspace" into the command line. Try it now and read the documentation. What happens if we leave off the number of samples and only tell MATLAB the start and stop values? How many samples N will MATLAB assume?

N = 100

Next, use your array of R_1 samples to calculate a corresponding array of samples in R_2. To accomplish this, all you need to do is subtract R_1 from R_s by typing

```
R2 = Rs - R1;
```

Notice that Rs is only a single value (i.e., *scalar*), but R1 is an array of 11 elements (i.e., a *vector*). Any time you add a scalar to a vector, MATLAB simply applies the calculation over every entry in the array.

Now let us calculate V_{out}. The simplest method is to type

```
Vout = Vs*R2/Rs;
```

Again, you may notice that R2 is an array while Rs and Vs are scalars. The multiplication and division is therefore performed over every entry in R2.

What if we instead tried the following command?

```
Vout = Vs*R2/(R1 + R2);
```

Try for yourself and write down the value for Vout that MATLAB calculated:

Vout = 2.5000

You may have noticed that Vout is only a single, scalar value rather than an 11-element array. This is **not** the desired outcome. What we want is for MATLAB to perform the calculation across every element in the array.

Name: Hayden Walpole Date: Sept 29, 2023

The problem with this instruction is that both "R2" and the quantity "R1 + R2" are arrays. Thus, the instruction "R2/(R1 + R2)" is asking MATLAB to divide an array by another array. This tends to be very confusing for beginners because array division is not something we are generally used to in ordinary arithmetic. Even so, MATLAB will still attempt to perform the calculation in accordance with the rules of linear algebra. In this situation, however, all we want to do is perform the division along each element in the arrays (element-by-element division). To that end, MATLAB gives us the "./" operator to make things explicitly clear.

Comment out the bad line of code and replace it with the following correct instruction:

```
Vout = Vs*R2./(R1 + R2);
```

What information is stored in Vout, now? Did you get an 11-element array this time?

Yes, Vout is now
[5
4.50000000000000
4
3.50000000000000
3
2.50000000000000
2
1.50000000000000
1
0.500000000000000
0]

Hayden Walpole

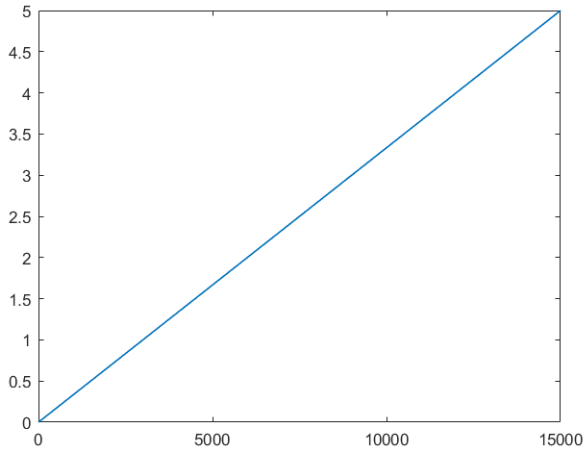
Name: _____

Sept 29, 2023

Date: _____

Let us now plot V_{out} as a function of R_2. The simplest method is to just use the *plot* command like the following:

```
plot(R2,Vout);
```



Include the above command in your script file and run it. You should now see a figure appear like the one at right. As expected, the output voltage varies linearly from 0–5 V as R2 varies from 0–15 kΩ.

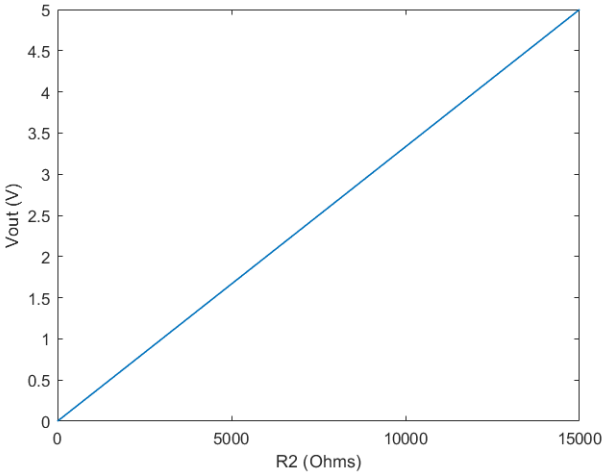
While the above code certainly works, it is generally bad practice to just start plotting data so abruptly. Often times, we may have multiple figures of data that need to get drawn, or maybe we are just modifying the same figure over and over. For this reason, it is usually better to initialize the figure before rendering any actual data. Thus, a cleaner implementation would actually look like this:

```
figure(1);
clf;
plot(R2,Vout);
```

The first command calls out the figure window and then brings it to the forefront. The "clf" line is short for "clear figure," and it simply clears any plot data that may or may not already exist.

Next, we need to add labels to our axes. To help us with this, MATLAB has given us the *xlabel* and *ylabel* commands. To see how they work, simply add the following commands after the plot:

```
xlabel('R2 (Ohms)');
ylabel('Vout (V)');
```



You should now see something similar to the image at right. In particular, take notice of the units in the axis labels. One of the most common mistakes that students make is to neglect units, which can easily lead to all kinds of confusion for the reader (not to mention loss of credit on assignments!).

Next, let us take a moment to fix the scale of our graph. Values like 5000 and 15000 are generally considered messy because of all the needless zeros. Instead, it is much neater to just scale down our resistance to units of kilo-ohms. To accomplish this, simply modify your plot command so that R2 is divided by 1000:

```
plot(R2/1e3,Vout);
```

Next, we need to update our units in the axis labels. Although we could simply write "kOhms," MATLAB is also capable of rendering Greek letters. The capital Omega, for example, is rendered through the command "\Omega." Try modifying your xlabel to read:

```
xlabel('R2 (k\Omega)');
```

Name: Hayden Walpole Date: Sept 29, 2023

Let us now experiment with the various line styles. By default, you may have noticed that MATLAB draws graphs with a solid blue line. In practice, however, you may hate this style and wish to use something different. Alternatively, you may wish to draw multiple graphs onto the same image, and so different lines will need to be rendered in different styles.

To help us with this goal, MATLAB offers a number of different line styles and colors. For example, to change the color to black, simply add another argument to the *plot* command with a 'k' character like so:

```
plot(R2/1e3,Vout,'k');
```

To change the style to a dashed line, replace the 'k' with '--k':

```
plot(R2/1e3,Vout,'--k');
```

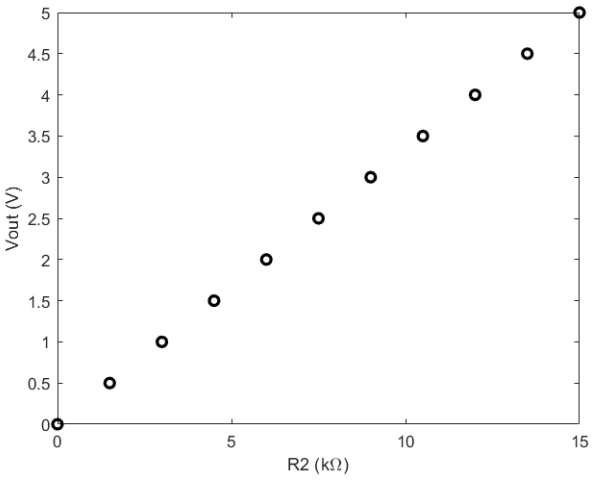
Often times, you may not like the default thickness of the line. In this case, you can add two more input commands with the form:

```
plot(R2/1e3,Vout,'--k','linewidth',2);
```

In many situations, the connected line is not the best way to display data. Instead, you may wish to simply draw a collection of dots. Such a graph is called a *scatter plot*, which you can create by using '.k' or 'ok':

```
plot(R2/1e3,Vout,'ok');
```

Try it now and see if your graph looks like the image at right.



To finish, type "help plot" into the command line and then explore the many style options available. Create a new figure window ("figure(2)") and then plot the power dissipated in R_2. You will need to use the formula

$$P_2 = I_s^2 R_2 \ .$$

Pick whatever line/plot style you think looks nice while also being legible. Convert the x-axis to units of kΩ and the y-axis to units of mW while adding the appropriate labels. When finished, save your plot as an image by selecting the "File" tab in the figure window and then "Save As." Use either .jpg or .png as your output format and include the image with your lab report.

What is the maximum power dissipated in the resistor?

$$P_{max} = \underline{\quad 1.67 \quad} \text{ mW}$$

Name: Hayden Walpole Date: Sept 29, 2023