Lab 1
Intro to MATLAB and Octave Online

Objectives

- *concepts*

1. Variables, vectors, and arrays
2. Plotting data
3. Script files

- *skills*

1. Use MATLAB to solve homework problems
2. Plot lab data and mathematical models
3. Write script files to save, edit and debug MATLAB commands
4. Solve systems of equations

Key Prerequisites

- None

Required Resources

- Access to MATLAB (buy here) or Octave Online (access at octave-online.net)

Circuit analysis is often a computationally challenging activity. Even powerful programmable calculators can be inadequate to the task. In this lab we develop some basic proficiency with MATLAB and its open-source cousin Octave Online, so that you can use these tools for your homework and future lab activities.

You are welcome to use MATLAB or Octave Online for this lab. Since most people will probably be using Octave Online on their home computers, this lab will illustrate concepts with Octave Online. For what we will be using them for, however, the two are almost identical.

Vocabulary

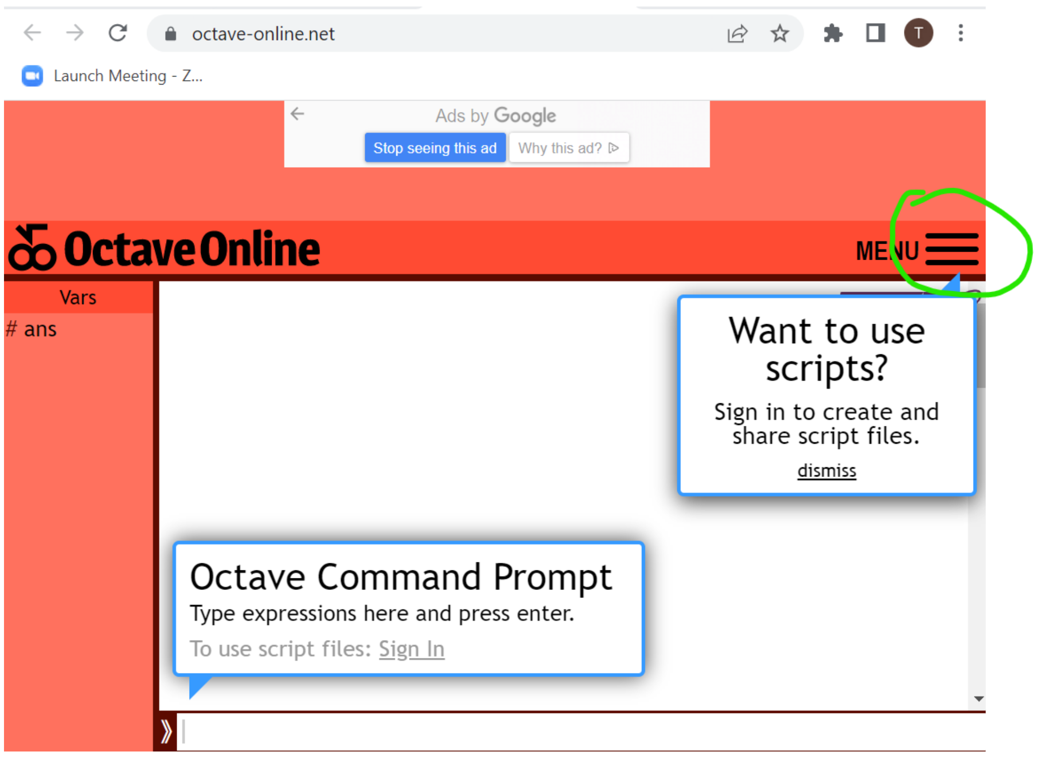
All key vocabulary used in this lab are listed below, with closely related words listed together:

variable, function
vector, array
plot, axes, legend
script file

Discussion and Procedure

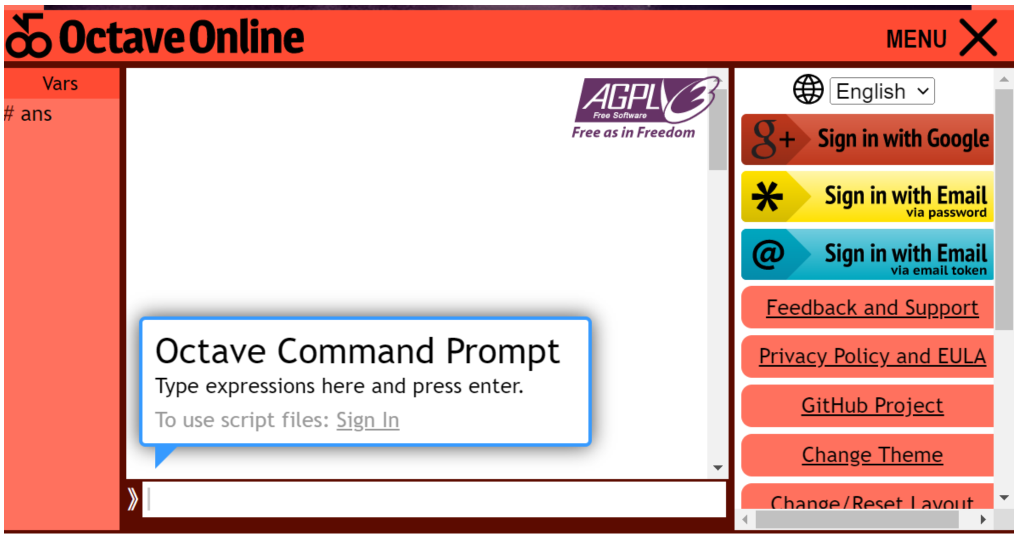
Part 1. Basics and Simple Arithmetic

Before we begin, we’ll need to set up an account on Octave Online. Visit the site at octave-online.net. You'll see a window that looks like this. There will be ads running across the top or the sidebar at right. If you donate a small amount of money ($3/month), the ads will go away once you log in:



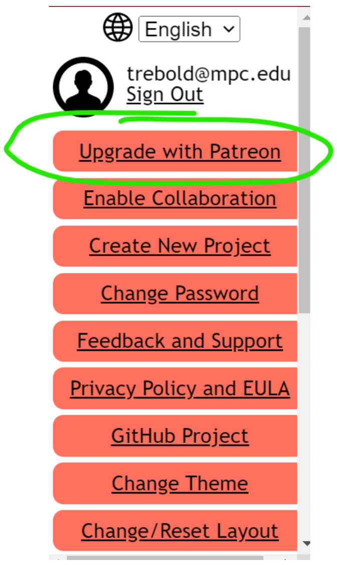
Although you can do a lot of work just using the Octave Command Prompt, we are going to use scripts (programs we write and store online), so we'll need to setup a login account.

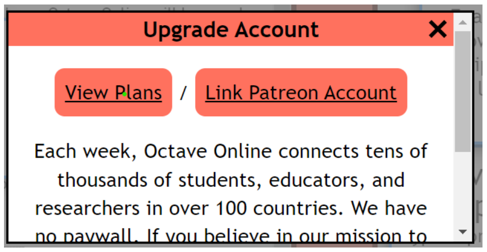
Click the "hamburger" (three horizontal lines circled in green next to MENU) and choose how you'd like to sign in. You can have multiple accounts on Octave Online, which can get confusing, so you should remember which way you use and always login the same way.



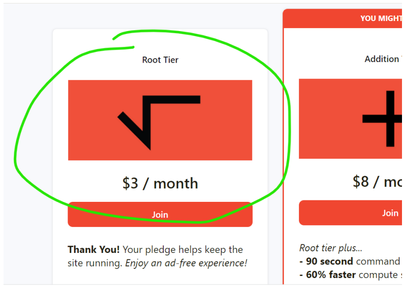
Octave Command Prompt
Type expressions here and press enter.
To use script files: Sign In

You can, if you wish, donate a small monthly amount to the Free Software Foundation through Patreon, which will eliminate the ads that show up in Octave Online. To do this, click the hamburger again after you log in. You'll see the upgrade option at the top of the right sidebar. When you click on that you'll be given a chance to connect to Patreon and subscribe. You can skip this for now and come back to it later if you like.





Each week, Octave Online connects tens of thousands of students, educators, and researchers in over 100 countries. We have no paywall. If you believe in our mission to

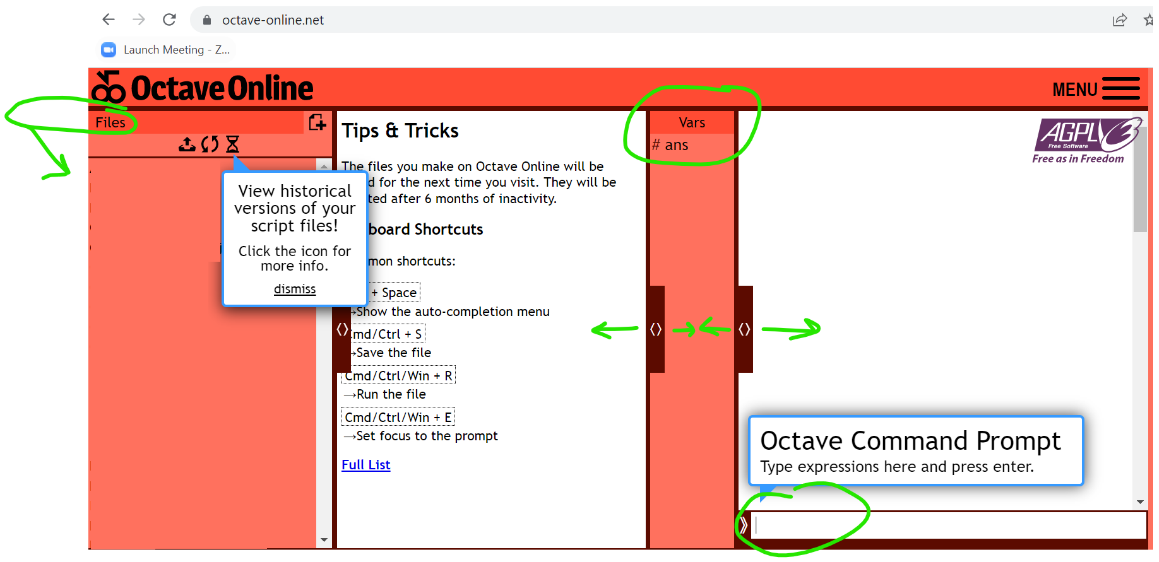


Thank You! Your pledge helps keep the site running. Enjoy an ad-free experience!

Root tier plus...
- 90 second command
- 60% faster compute

Now we will identify the main components of the Octave Online environment, shown in figure 1 below. Octave Online divides your window into 4 sections. From left to right, these are the **Files**, **Tips&Tricks**, **Vars**, and **Command Line** window. These windows roughly correlate with the standard MATLAB windows, but they are reorganized for convenience working in a browser.

Note that you can easily resize the windows by dragging the vertical separator bars at the middle indicated by green left/right arrows below.



View historical versions of your script files!

Click the icon for more info.

The files you make on Octave Online will be
d for the next time you visit. They will be
ted after 6 months of inactivity.

board Shortcuts

mon shortcuts:

+ Space
→Show the auto-completion menu

md/Ctrl + S
→Save the file

Cmd/Ctrl/Win + R
→Run the file

Cmd/Ctrl/Win + E
→Set focus to the prompt

Octave Command Prompt

Type expressions here and press enter.

Figure 1. The Octave Online Environment

The first window at the left is the **Files** window where the scripts (programs) you write will be shown. The **Tips&Tricks** window is self-explanatory.

The **Command Window** is the furthest to the right, and is where you can type commands directly into the Octave processor and view the results. We spend much of our time in Octave Online working in the **Command Window**, at right, which is where we type instructions.

Any variables we create are displayed in the **Vars** window, which lists all the variables we've created. You can view the value stored inside a variable by clicking on it in the Vars window. Programming is largely an activity in moving information around to different variables.

The simplest command in MATLAB/Octave is one that simply processes numeric information. If you enter a number at the command prompt:

```
>> 1.5
```

and hit enter, MATLAB evaluates the number and echoes it back as the answer (ans).

```
ans = 1.5000
```

You can write more complex expressions using the arithmetic operators +, --, * , / , and ^ (* is multiplication and ^ is exponentiation), for example (try these out yourself):

```
>> 1.5 + 2
ans = 3.5000

>> 1.5 + 2 * 3
ans = 6.5000

>> (1.5 + 2 ) * 3
ans = 10.500

>> 3^2 + 4^2
ans = 25
```

Note how MATLAB follows the order of operations (multiplication before addition) for arithmentic, and how parentheses can change the order of operations.

Part 2. Variables, calculation, and functions

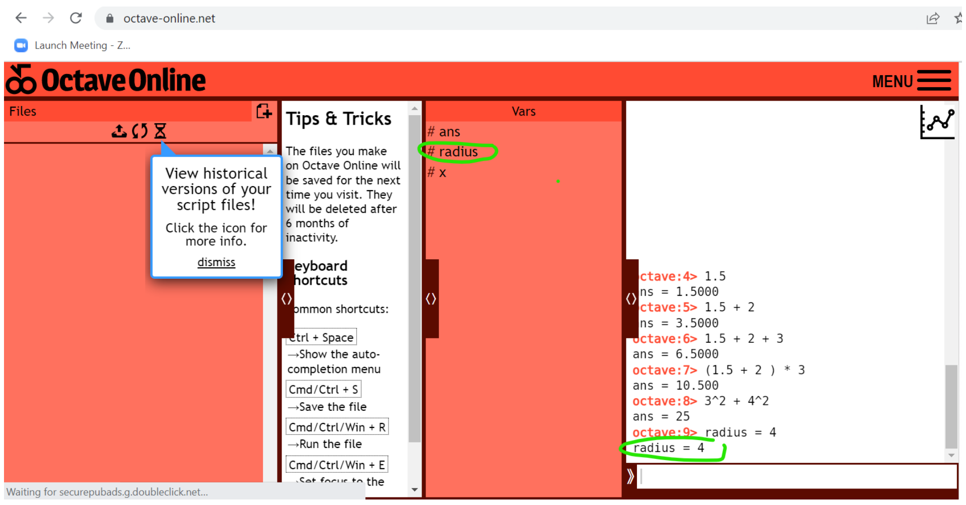
When computations get more complicated, we need a way to keep track of where information is stored. A **variable** is a memory location that we name, so we can put values in it and go back later and retrieve the values from it. For example, if we type (TRY THIS):

```
>> radius = 4
```

MATLAB will echo back the new variable name, and the value it contains.

```
radius = 4
```

Also, notice that the Variable Workspace window and the Command History windows also show the new variable we created (radius), as seen in Figure 2 below.



View historical versions of your script files!

Click the icon for more info.

The files you make on Octave Online will be saved for the next time you visit. They will be deleted after 6 months of inactivity.

eyboard hortcuts

ommon shortcuts:

trl + Space →Show the auto-completion menu

Cmd/Ctrl + S →Save the file

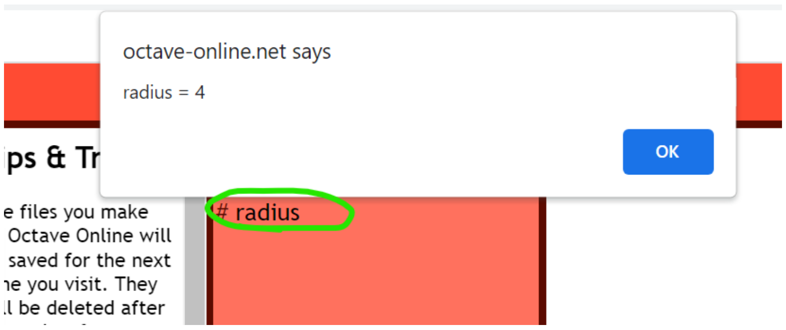
Cmd/Ctrl/Win + R →Run the file

Cmd/Ctrl/Win + E →Set focus to the

```
ctave:4> 1.5
ns = 1.5000
ctave:5> 1.5 + 2
ns = 3.5000
octave:6> 1.5 + 2 + 3
ans = 6.5000
octave:7> (1.5 + 2 ) * 3
ans = 10.500
octave:8> 3^2 + 4^2
ans = 25
octave:9> radius = 4
radius = 4
```

Figure 2. Notice the Octave Online environment after the variable `radius` is created, showing also in the Vars window.

Also, if you click on the variable radius in the Vars window, you can view its value:

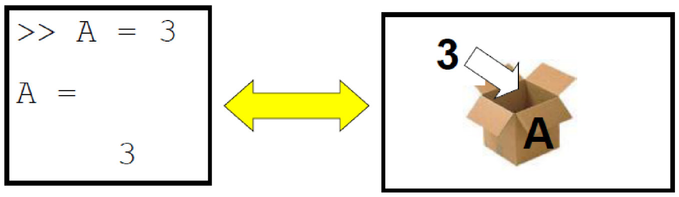


octave-online.net says

radius = 4

e files you make Octave Online will saved for the next ne you visit. They l be deleted after

A good way to visualize internally what happens when we create a variable is to picture the cardboard box on the right in Figure 3, where we've stored a value of 3 in the box (variable) A.



```
>> A = 3

A =

     3
```

Figure 3. A variable is like a box that holds a value. The name of the box is the name of the varible, also called the identifier. The contents of the box is the value it holds.

The rules for naming variables in MATLAB are similar to other programming languages: they must start with a letter, and otherwise may be composed of any letters, numbers, or the underscore character " _." For example, the following statements use acceptable variable names:

```
experiment8temp = 54
one_two_three = 456
residualData = experiment8temp - 45
```

Note that in the last line we are pulling out the value of the `experiment8temp` variable to use in an arithmetic expression. In this case, `residualData` is assigned the value 9.

We can now create a new variable for the area of a circle using the value in `radius` as well as the built in variable `pi`. Simply type (TRY THIS):

```
>> area = pi * radius^2
area = 50.2655
```

If we want or need to do more complicated calculations, MATLAB provides a large number of **functions**. These work very similarly to the functions in your math class. They take an input value and give back an output value.

For example, if you are trying to find the hypotenuse of a right triangle with a base of 3 and a height of 1.5, you can use the `sqrt` function (TRY THIS)

```
>> hypotenuse = sqrt(3^2 + 1.5^2)
hypotenuse = 3.3541
```

Other functions you may want to use are listed in Table 1:

Table 1: Common MATLAB/Octave Online mathematical functions

Function	Description	Function	Description	Function	Description
`sin`	sine	`cos`	cosine	`tan`	tangent
`asin`	arcsine	`acos`	arccosine	`atan`	arctangent
`log`	log to base e	`log2`	log to base 2	`log10`	log to base 10
`sqrt`	square root	`exp`	e raised to		

(TRY THIS) **Now type the up arrow**. Notice how the last statement has been recovered? You can edit the statement and put a semicolon (;) after it so it now reads:

```
>> hypotenuse = sqrt(3^2 + 4^2);
>>
```

Notice that nothing is printed. The variable `hypotenuse` was calculated, but the semicolon instructs MATLAB not to print its value to the screen. You can see the value of `hypotenuse` by entering the command:

```
>> hypotenuse
ans = 5
```

Question 1 ***(Write in the Lab1 Datasheet)*** *Summarize the use of the semicolon in MATLAB.*

Part 3. Using Script files to Solve Analysis Problems

To further reinforce what we've just learned, and to bring in an additional technique, we will now work an extended example taken from lecture. Please follow along by typing commands where indicated.

Example 1 A conductor has a constant current of 5 A. How many electrons pass a fixed point on the conductor in one minute?

To solve this problem we'll first consider three things:

a) what are the INPUTs to the problem?
b) what are the relevant FORMULAs we can use to solve the problem?

c) what is the OUTPUT or result we are calculating?

For this example, the answer to a) is current = 5A. For b) we must round up the formulae for how many charges are in a coulomb, how many coulombs per second it takes to give 1 amp of current, and how to change units from charges per second to charges per minute. For c) we realize the result we are looking for is charges per minute.

Now that we have our basic material together, we can start forming a solution in Octave Online. Before we get started though, it's a good idea to clear out the Variable Workspace by typing **clear**. This wipes out all variables as you can see in the workspace window, so that you don't accidently use a value stored from a previous calculation. If you want to erase the command window, just type **clc**.

Question 2 ***(Write in the Lab1 Datasheet)*** *Why is it a good idea to type* `clear` *before developing a new solution in MATLAB?*

First we'll define the variables for current and the number of charges in a coulomb (TRY THIS):

```
>> current = 5
>> coulomb = 6.25e18
```

Note that we use the letter e in the above expression to indicate x10 to the power. Now we can compute the number of charges per second using the statement

```
>> charges_per_sec = current * coulomb
```

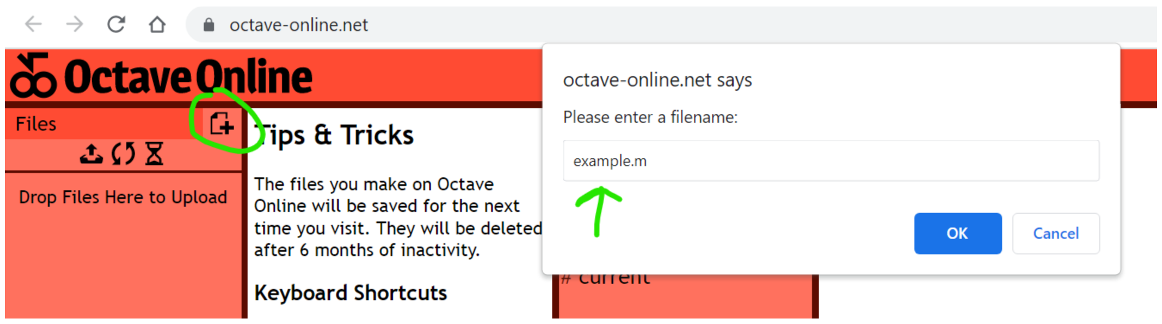
Finally, we can compute the number of charges per minute using the statement:

```
>> charges_per_min = charges_per_sec * 60
```

And we get our final answer, 1.8750e+021.

Although this gets us the answer, it isn't the most effective way of using MATLAB. The best way is to put the commands for the solution in a separate file, called a **script** file. That way if we have any errors, or would like to repeat the calculation with different input values, we can easily run the entire solution again.

To create our first script file (TRY THIS), click the +Folder option in the Files window, then type example.m in the text box that pops up. as shown in figure 4:



The files you make on Octave Online will be saved for the next time you visit. They will be deleted after 6 months of inactivity.

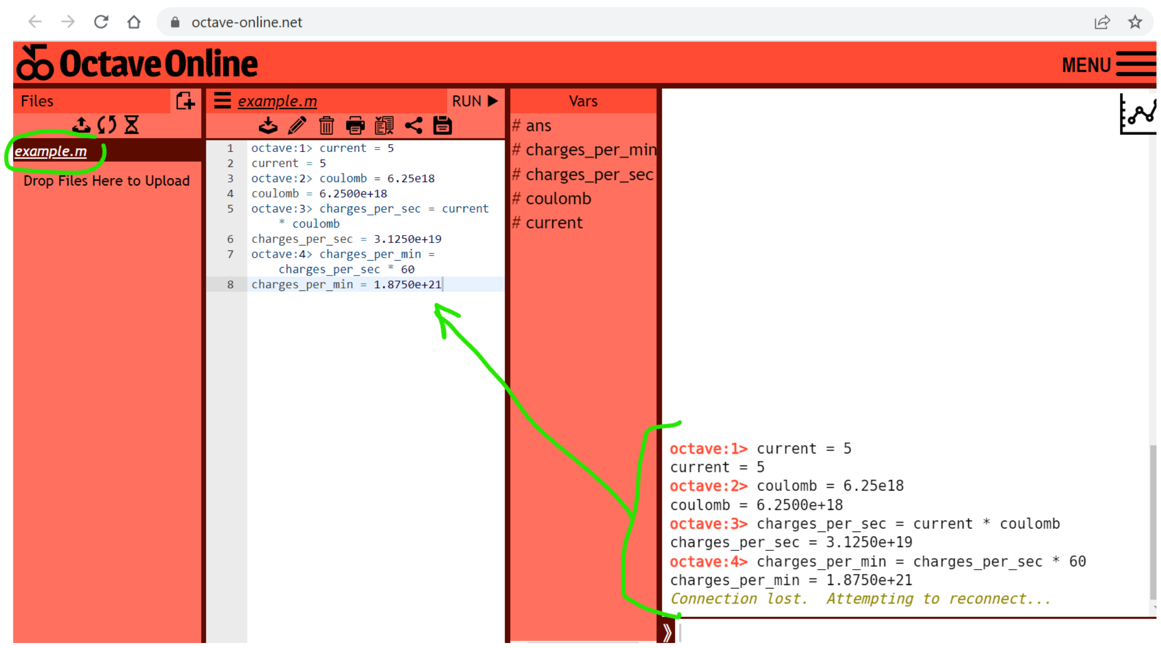
octave-online.net says

Please enter a filename:

example.m

Figure 4. Creating a script in the Files Window

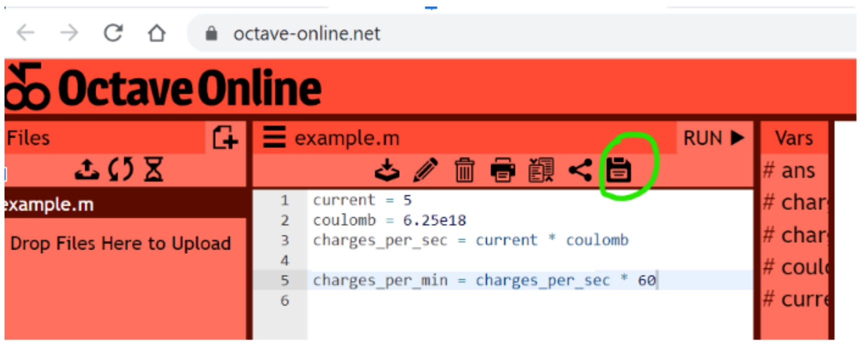
Now click on example.m in the File window and you'll see the code that's in the file in the window that used to say Tips&Tricks, currently a simple display message. Next, copy your statements from the command window and paste them into the examples.m window as shown below:



```
octave:1> current = 5
current = 5
octave:2> coulomb = 6.25e18
coulomb = 6.2500e+18
octave:3> charges_per_sec = current * coulomb
charges_per_sec = 3.1250e+19
octave:4> charges_per_min = charges_per_sec * 60
charges_per_min = 1.8750e+21
```

```
octave:1> current = 5
current = 5
octave:2> coulomb = 6.25e18
coulomb = 6.2500e+18
octave:3> charges_per_sec = current * coulomb
charges_per_sec = 3.1250e+19
octave:4> charges_per_min = charges_per_sec * 60
charges_per_min = 1.8750e+21
Connection lost.  Attempting to reconnect...
```

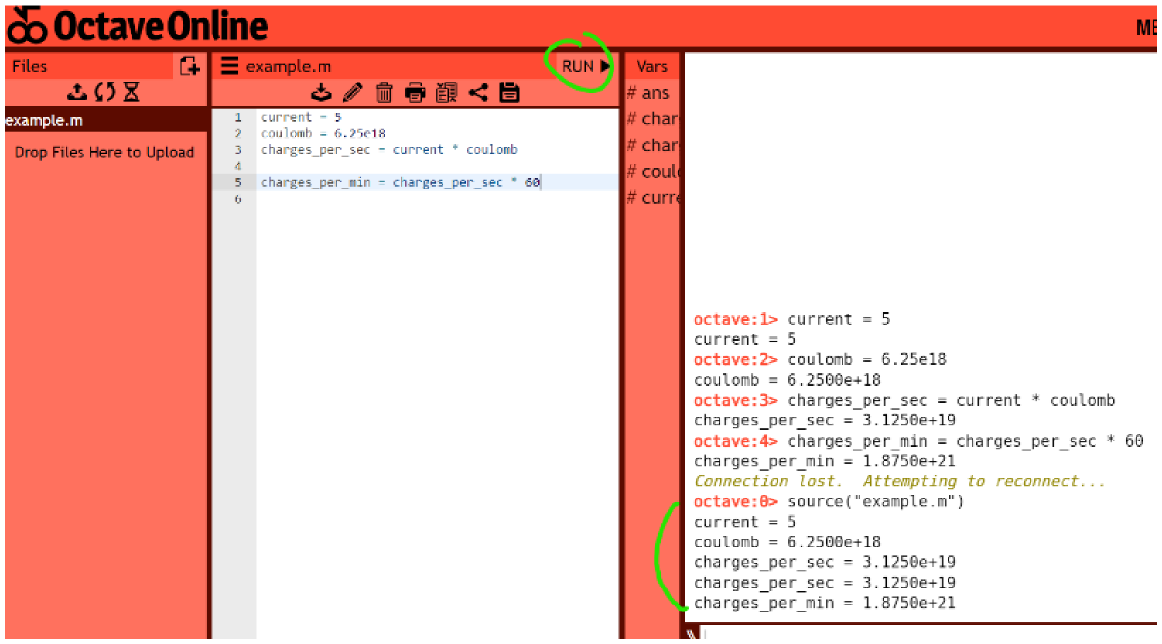
Now edit out the extraneous info in the file (all the octave prompts and the results) and click SAVE, so that it looks like this:



```
current = 5
coulomb = 6.25e18
charges_per_sec = current * coulomb

charges_per_min = charges_per_sec * 60
```

That's it! Now whenever you wish to run the program, instead of typing the statements at the command line, you can just click the RUN button in the examples.m editor window. And you will see the results in the command window. Like this:



```
current = 5
coulomb = 6.25e18
charges_per_sec = current * coulomb

charges_per_min = charges_per_sec * 60
```

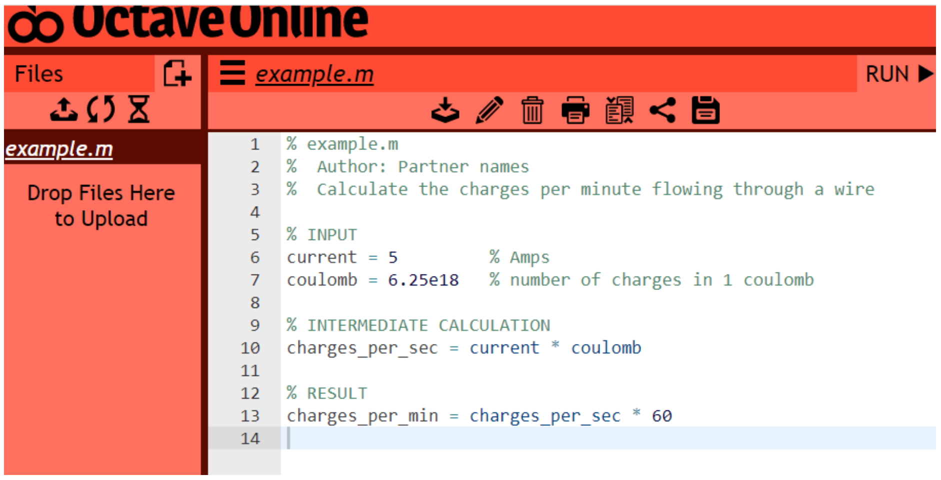
```
octave:1> current = 5
current = 5
octave:2> coulomb = 6.25e18
coulomb = 6.2500e+18
octave:3> charges_per_sec = current * coulomb
charges_per_sec = 3.1250e+19
octave:4> charges_per_min = charges_per_sec * 60
charges_per_min = 1.8750e+21
Connection lost.  Attempting to reconnect...
octave:0> source("example.m")
current = 5
coulomb = 6.2500e+18
charges_per_sec = 3.1250e+19
charges_per_sec = 3.1250e+19
charges_per_min = 1.8750e+21
```

Note that Octave Online and MATLAB script files always have a **.m** at the end of the file name.

(TRY THIS) In the example.m editor window, change the value of current to 10, click SAVE and then RUN to repeat the calculation. Notice the new result that you get.

Finally, your solution would be more readable if you have some comments explaining the purpose of the program, and what the statements are doing. MATLAB uses the % sign as a comment symbol. Anything appearing after the % on a line is ignored by the interpreter and not processed. So you can put notes in your program without affecting its execution.

Edit your example1 script so it looks like this (you may need to slide the vertical bar of the editor window to the right to make room for the longer lines of text):



```
% example.m
%  Author: Partner names
%  Calculate the charges per minute flowing through a wire

% INPUT
current = 5          % Amps
coulomb = 6.25e18    % number of charges in 1 coulomb

% INTERMEDIATE CALCULATION
charges_per_sec = current * coulomb

% RESULT
charges_per_min = charges_per_sec * 60

```

Note that semicolons are added after each intermediate step to reduce the output display. If we suspected an error, we could remove the semicolons and inspect all the parts leading up to the final answer for errors.

Question 3 *Copy the code in your example1 script into the Lab1 datasheet.*

If you are working with a partner, now is a good time to switch "drivers" on the computer. For further practice, create a new script file by clicking the + symbol in the Files window. Save the new script as Question4.m. Inside the script, develop and debug a solution to the following problem.

Question 4 *Calculate the mass of water in a cylindrical tank of height 1.5m and radius 0.3m, given that the density of water is 1000 kg/m^3.Your answer should be 424.115 kg.*

When it works, copy the code in your Question4 script into the Lab1 datasheet.

Part 4. Vectors and Arrays

MATLAB variables can also store arrays or vectors. A two dimensional array is a table of values, with **m** rows and **n** colums. A vector is an array with only 1 row (a row vector) or 1 column (a column vector). Type in the following commands and notice the results (TRY THIS):

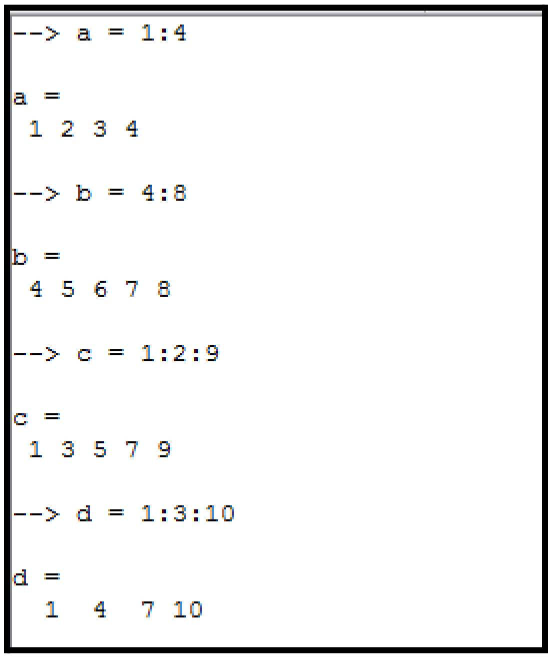
```
>> clear

>> a = 1:4

>> b = 4:8
>> c = 1:2:9

>> d = 1:3:10
```

Check your results with Figure 5.



```
--> a = 1:4

a =
 1 2 3 4

--> b = 4:8

b =
 4 5 6 7 8

--> c = 1:2:9

c =
 1 3 5 7 9

--> d = 1:3:10

d =
   1   4   7 10
```

Figure 4. Four vectors created using the colon (:) operator

Question 5 ***(Write in the Lab1 datasheett)*** *Summarize the use of the colon in MATLAB.*

There are other ways to create array variables. (TRY THESE)

```
>> m1 = [1, 8, 4, 6, 2]

>> m2 = [1; 8; 4]

>> m3 = [2 4 6 22]


>> m4 = [1 2 3; 4 5 6]

>> m5 = [a; d]
```

Calculations

MATLAB can also perform math operations on array/vector variables. The following commands use the previously defined values for a, b, c, and d. (TRY THESE):

```
w = a + d

x = b + c

y = a + c

z = sqrt(w)
```

Question 6 ***(Write it the Lab1 datasheet)*** *Summarize the results of the above for your report. The third command gives you an error. Why?*

Part 5. Plotting Data

One of the most useful features of MATLAB is its plotting capability.

Question 7 Enter the following commands in the Octave Online command window, and write what happens as each command is executed in your datasheet.

```
t = [0:0.1:10];

x = exp(-t);

plot(t,x);

xlabel('time (s)');

ylabel('x(t)');

title('plot of e^(-t)');

grid on
```

The plot output will show in the command window, similar to that in Figure 5 if you are using Octave Online. Note that sometimes the title or axis label gets cut off in Octave Online. You can usually recover them by expanding the image to fill the whole screen.

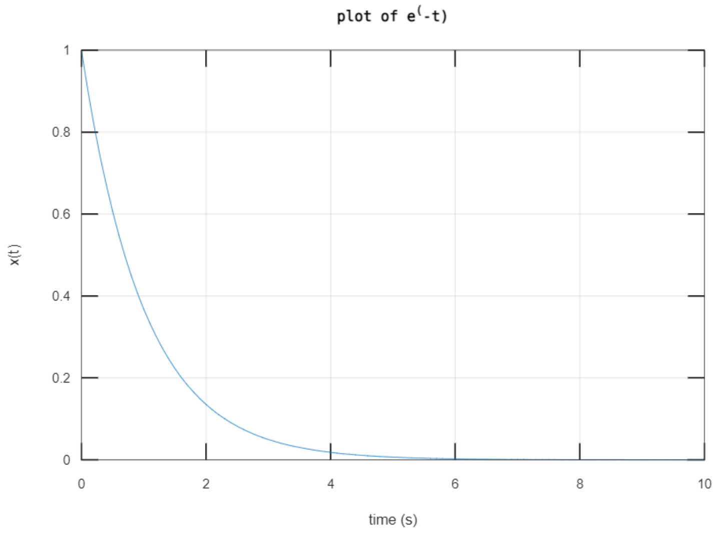
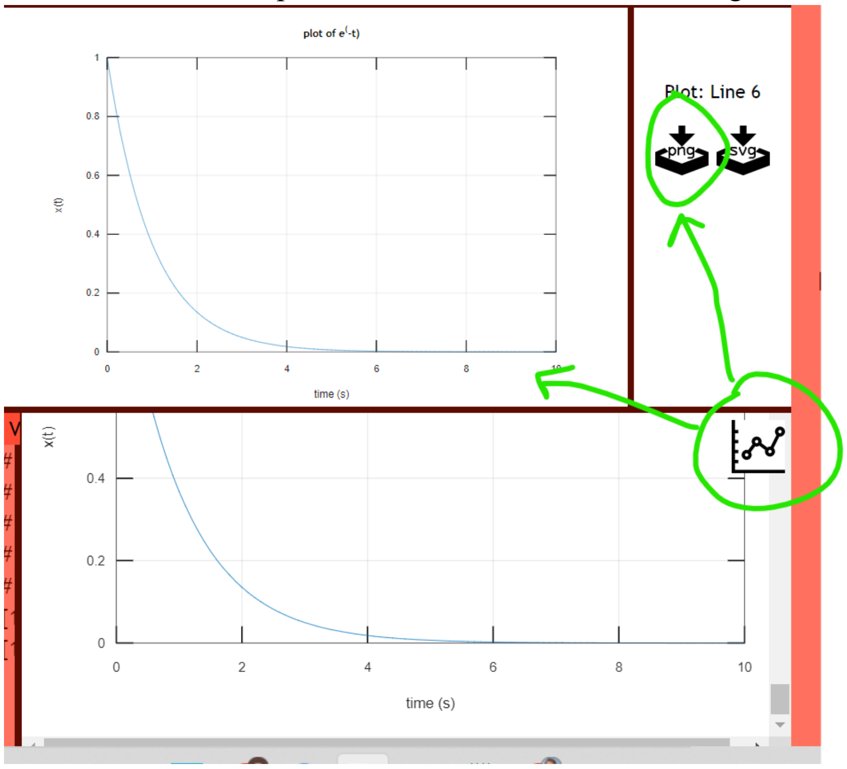


Figure 5. Results of plotting two vectors, x vs t.

You can also download a .png format image of the plot if you click on the zig-zag line in the top right of the plot image. Clicking on it a second time will restore the normal command window version of the plot view, as shown in the next image:



You can also plot more than one signal at a time. Enter the following commands in the Octave command window (TRY THIS):

```
y = sin(t);
plot(t,x,'-', t,y, '-.');
legend('x(t)=exp(-t)', 'y(t) = sin(t)');
grid on
xlabel('time (s)');
title('two plots at once')
```

Notice that each plot component has three arguments: the *x* axis values, the *y* axis values, and a string for the line style. You should now be able to see the two signals in one plot.

Question 8) Suppose we performed a test of Ohm's law in the lab. We applied five different voltages across a resistor and at the same time measured the current passing through the resistor at each voltage setting, resulting in the following table of data:

Measurement	1	2	3	4	5
V, Volts	0.5	1	2	3	10
I, Amps	0.0052	0.009	0.021	0.03	0.105

Make a new script file called Question 8. In this file, write code to create a plot of the data in the previous table. Define the vector V = [*list the values separated by commas*] and define the vector I the same way. Then use the plot command to plot the data. Make sure I is on the x-axis and V is on the y-axis. Add labels and a title to your plot and copy (Select Tools > Copy) and paste it in your report as well as the code that produced it.

Part 6. Solving Systems of Equations

Consider a common occurrence in Engineering, the need to solve a system of *simultaneous* equations:

$$3x + 4y + 5z = 32$$

$$21x + 5y + 2z = 20$$

$$x - 2y + 10z = 120$$

In this case we are looking for a solution set—a value of x, y, and z—that satisfies all 3 equations. In general, these 3 equations could have 1 solution, many solutions, or NO solutions. For now, we will look for a single solution using matrix algebra.

The method involves setting up the equation in matrix/vector form, in other words,

$$\begin{vmatrix} x \\ y \\ z \end{vmatrix}$$

$$\begin{vmatrix} \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{21} & \mathbf{5} & \mathbf{2} \\ \mathbf{1} & \mathbf{-2} & \\ \mathbf{10} & & \\ & & \\ & & \end{vmatrix} = \begin{vmatrix} \\ \mathbf{32} \\ \\ \mathbf{20} \\ \mathbf{12} \\ \mathbf{0} \end{vmatrix}$$

By separating the coefficients from the unknowns, we can rewrite the above system of equations as a single matrix algebra equation:

$$\mathbf{A\ u = b,}$$

$$\text{where } \mathbf{A}= \begin{vmatrix} \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{21} & \mathbf{5} & \mathbf{2} \\ \mathbf{1} & \mathbf{-2} & \\ \mathbf{10} & & \\ & & \\ & & \end{vmatrix} \quad , \quad \mathbf{b} = \begin{vmatrix} \\ \mathbf{32} \\ \\ \mathbf{20} \\ \mathbf{12} \\ \mathbf{0} \end{vmatrix} \quad , \quad \text{and } \mathbf{u} = \begin{vmatrix} x \\ y \\ z \\ \\ \\ \\ \end{vmatrix}$$

There are many ways of solving the system as we've currently defined it. MATLAB is particularly effective, since it was initially designed to process large matrix algebra systems with thousands of degrees of freedom (unknowns).

In Matlab/Octave, we simply define the **A** matrix and the **b** vector by typing (TRY THIS):

```
>> A = [3 4 5; 21 5 2; 1 -2 10]

>> b  = [ 32 ;  20 ; 120]
```

The matrix equation **A u = b** is solved by pre-multiplying both sides by the inverse of **A** to get

u = I/A * b (not a Matlab command)

where I is the identity matrix. Note that since we are using matrices here, the order of operation is critical. It would be wrong to say **u = b/A**, even though that would work in basic algebra. We have to **<u>pre</u>**-multiply by the inverse of A.

Matlab provides two ways to do this…using "Left division" (the backslash symbol), we can say **`u = A \ b`** (This is the recommended approach for solving systems of equations, since it uses the more stable Gauss Elimination method). Or we can use the matrix inverse function inv( ): **`u = inv(A) * b`**

```
>> u = A\b
u =
     1.4497       ( value of x)
    -6.3249       ( value of y)
    10.5901       ( value of z)
```

Finally, to prove you have the correct answer, you can plug these values in the original equation **`check = A * u`** and see if you get the b vector you started with. If you don't, there may be something wrong with the specification of the system resulting in numeric instability.

First type in the previous example to make sure you get the correct answer. Then solve the following sets of equations:

Question 9. Find the solution to the following system and paste your code and answer into the Lab1 datasheet.

```
  x  +   y  +   z   =  6
 2x  +  5y  +   z   = 15
-3x  +   y  +  5z   = 14
```

Question 10. Solve the following word problem. Copy the script file and answer into the datasheet.

The admission fee at a small fair is \$1.50 for children and \$4.00 for adults. On a certain day, 2200 people enter the fair and \$5050 is collected. How many children and how many adults attended?