In the script window start by writing the commands ”clear all”, ”close all”, and ”clc”. Including this in scripts is a good practice to clean up the workspace before writing a script.

Question 1: What do these functions do?

Answer:

“clear all” - removes all variables and functions from RAM,

“close all” - closes all figures,

“clc” - Acts like “clear” in unix terminals, clears the command window.

Write ”y = 2*3 ˆ 2*exp(2)”

Question 2: What is the output in the command window?

Answer: 133.0030

Now add a semicolon behind the equation in the script.

Question 3: What does the semicolon do to the output of the script?

Answer: Nothing because the semicolon suppresses output to console.

Define a new variable x = 2 in the line above the previous equation. And then change the definition of y to y = x*3 ˆx*exp(x). Now the variable x can be changed and y will change automatically when the script is run. This removes the need for manually changing the input to the equation.

Use the .m file from above and create a vector called v1 with the command v1 = [1 2 3 4]

Question 4: Is this a row or a column vector?

Answer: v1 is a row vector

Now create v2 = [1; 2; 3; 4].

Question 5: What do semicolons do in this case?

Answer: v2 is a column vector because semicolons denote the end of a row in Matlab

---

Multiply v1 and v2 together.

Question 6: Does order matter? Ie v1*v2 vs v2*v1, explain why.

Answer: Order matters because matrices must have appropriate dimensions. In math terms, matrices (particularly those of vectors R^n) form a non-abelian group.

Other useful ways of creating these vectors would be v1 = 1:1:4 or v1 = linspace(1,4,4). These functions enable the creation of large vectors without having to manually populate them

Question 7: What are the input parameters for these functions, how do they differ between the two methods?

Answer: The input parameters for the first line is start:step:end where start is the number to start counting at, step is the increment value, and 4 is the number to end on.

The linspace function is of the form linspace(start,end,elements) where start is the number to start counting at, end is the number to stop at, and elements is the number of elements you want to take to get there.

Define a voltage v as v = linspace(0,12,121).

Question 8: What is the purpose of feeding 121 into this equation (as opposed to any other number)?

Answer: Because we started counting at 0, we must add an extra element to account for the element containing 0

Now define a 4.7 kΩ resistance with R =4.7e3 (note: the powers of ten are represented by e, not exp). To find the current we need to solve Ohm's law: i = v/R. Create i and run the function

Question 9: What happens when you run the script?

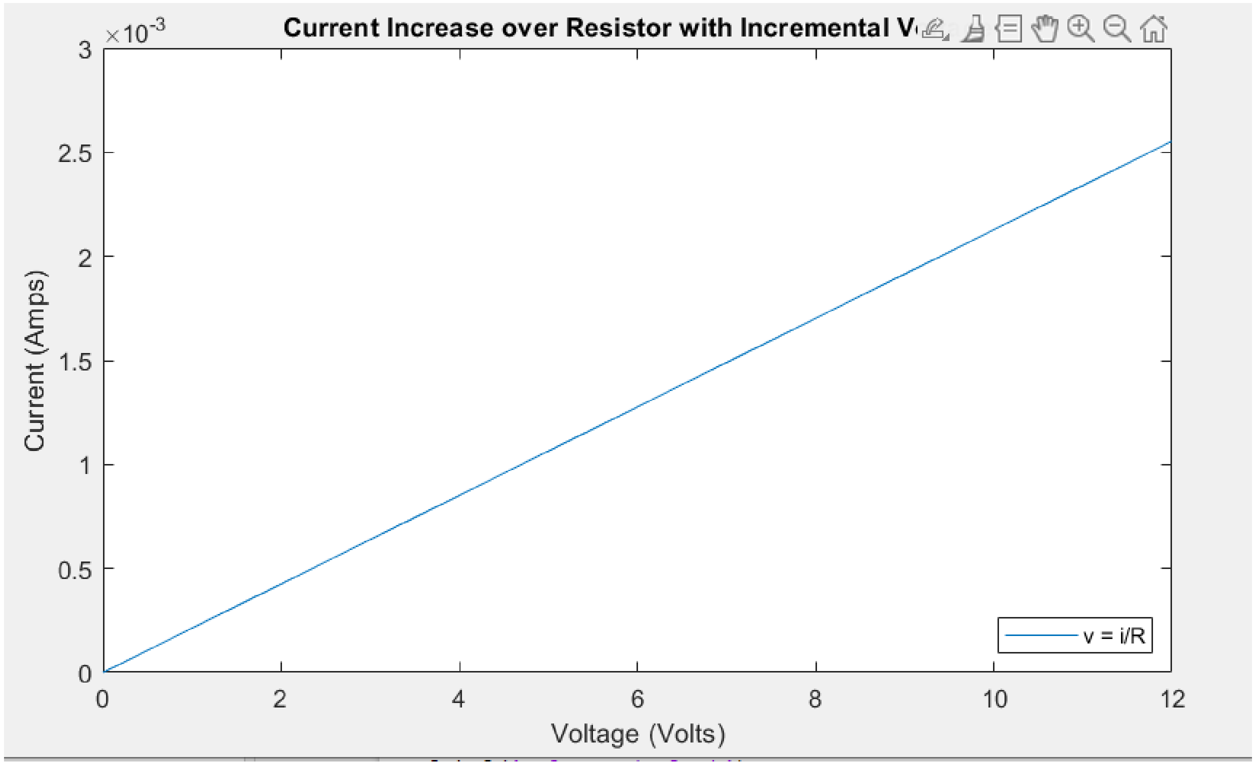
Answer: It outputs the result of matrix division of the row vector of size 121 and the 1x1 vector R

MATLAB®does by default use matrix operations. To suppress those, a period is used in front of the operator. So redefine i = v./R and run it. To access specific element n in a vector in MATLAB®, the command is i(n)

Question 10: What is the voltage and current when n=89?

Answer: (v,i) = (8.8 volts, 1.9mA)

To plot a function the command "plot" can be used. Create a new line in the .m file with the command "plot(v,i)". Then add labels on the x and y axis. Add a legend with the equation. Save the plot and include in the report



Create a new .m file.
• Define the variables x = 10 and y = 5.
• Enter the following expressions:
u = x > y
v = x < y
w = x == y
ww = x >= 2 *y
Question 11: What are the outputs and what are the meaning of the values?
Answer:
Output:

```
u =
  logical
   1
v =
  logical
   0
w =
  logical
   0
ww =
  logical
   1
```

The output is a logical matrix of the result of the logical comparisons of each element of matrices x and y. In this case x and y are 1x1 matrices so the logical matrices that result are also 1x1.

Look up the syntax for if statements. Then create an if statement that checks whether x is greater than y and sets y = x, else it should set x = y. **Show the code segment.**

```
function maxima = max(x,y)
    if x>y
          y=x;
    else
          x=y;
    end
    maxima = x
end
```

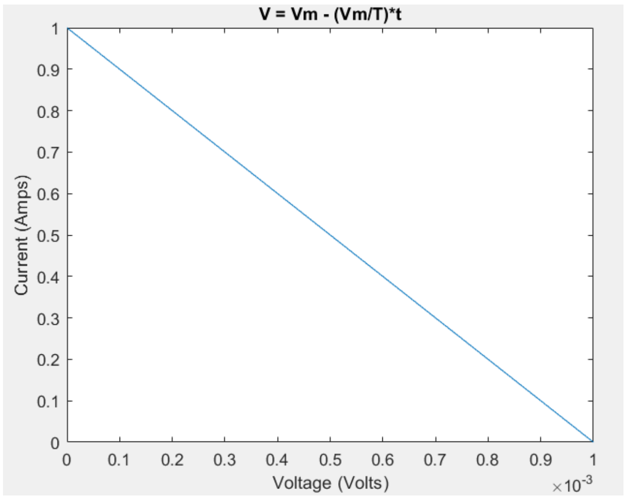
Create a new .m file or add to the one created above.
• Define the period T = 1 millisecond.
• Define the voltage amplitude Vm = 1 volt.
• Define the time vector t = linspace(0,T,1001).
Question 12: What is the incremental value of t?

Answer: 10^{-6}

Look up the syntax for for loops. Then use a for loop to create the vector v which will be based on
v(t) = Vm − (Vm/T) * t.
Be careful to use the loop counter as the vector t index and also use the same index for storing the value of
v. Whenever you are plotting something in MATLAB (or any software), you should have a good idea of what
it should look like. In other words, you can always plot by hand to get a comparison. This is a very simple sanity check to verify whether the code is outputting the desired plot(s)

Question 13: Plot v vs. t

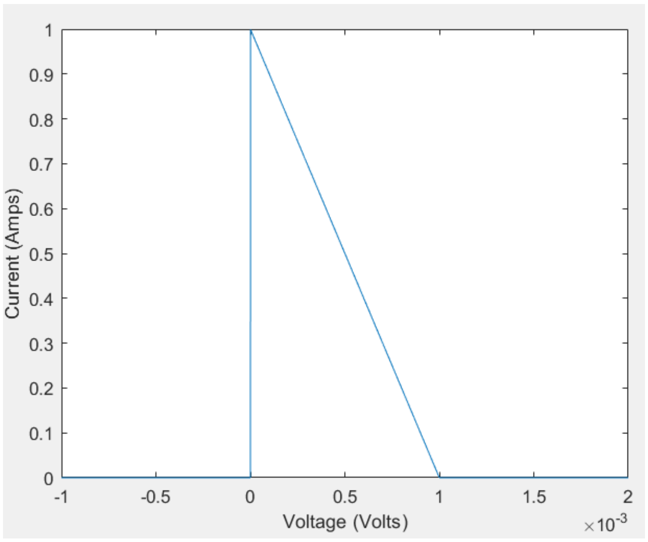
Answer:



Now redefine the time vector t = linspace(-T,2*T,3001). And create the vector v using (see project pdf)

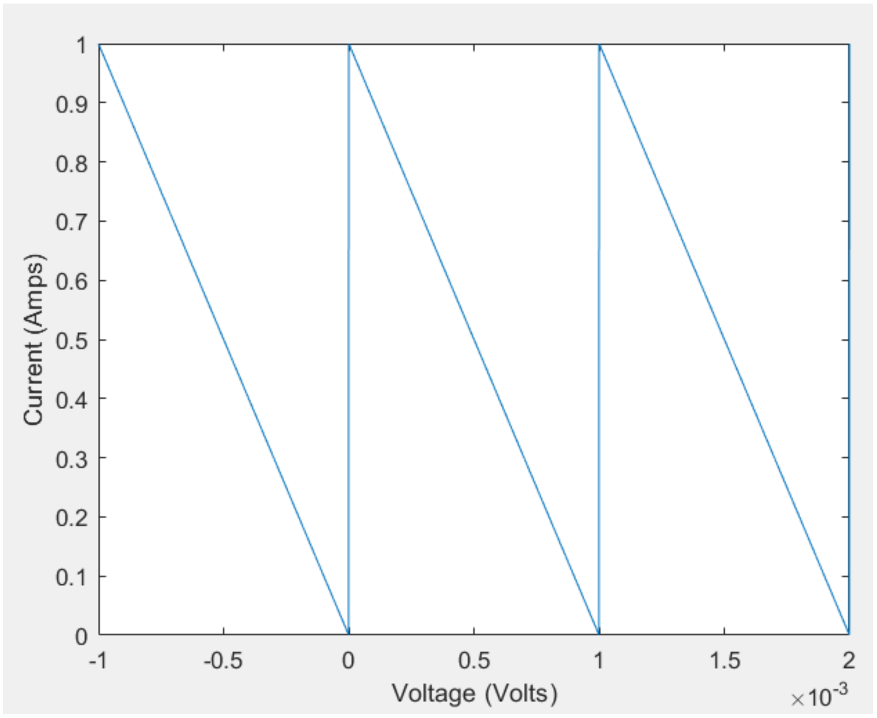
Question 14: Plot v vs. t over the range of -T to 2T

Answer:



Now modify the vector v so that is represents a periodic function with a period T over this time period (hint: it should look like a sawtooth)

Question 15: After modification, plot v vs. t over -T to 2T again.
Answer:


Current (Amps)
Voltage (Volts)
$\times 10^{-3}$