*Heaven’s light is our guide.*

**Rajshahi University of Engineering and Technology**

**(RUET)**

**Department of Electrical & Electronic Engineering**

**Course no.** EEE3110

**Course title:** Computational Methods in Electrical Engineering Sessional

**Experiment no.** 01

**Experiment name:** To get familiar with MATLAB

**Submitted to:**

Dr. Md. Masud Rana

Professor

Dept. of Electrical & Electronic Engineering,

Rajshahi University of Engineering and Technology.

**Submitted by:**

**Ashraf Al- Khalique**

Roll: 1801171; Session: 2018-2019

Dept. of Electrical & Electronic Engineering,

Rajshahi University of Engineering and Technology.

**Date of experiment:** February 27, 2022.

**Date of submission:** March 06,2022.

**Experiment No. 01**

* 1. **Experiment Name**

To get familiar with MATLAB

* 1. **Objectives**
* To become accustomed with the simulation of power electronic circuits in the MATLAB environment
* Learn how to use MATLAB to create a simple system
* Learn how to run big and complex calculations for projects and sessional through MATLAB
* To get familiar with simple plot command
  1. **Theory**
     1. **MATLAB**

MATLAB is a high-performance programming language used in technical computing. It combines computing, visualization, and programming in a user-friendly environment in which they express problems and answers in common mathematical notation. MATLAB is an abbreviation for matrix laboratory. MATLAB made it simple to access matrix data.

* + 1. **MATLAB Desktop Applications**
* Command Window: The Command Window is where you enter variables and run functions and M-Files.
* Command History: This saves statements typed into the Command Window in the Command History. We can inspect previously run statements in the Command History, as well as copy and execute chosen statements.
* Current Directory Browser: MATLAB file operations use the current directory reference point. Any file you wish to execute must be in the current directory or on the search path.
* Workspace: A MATLAB workspace is a collection of variables (called arrays) that are created and saved in memory throughout a MATLAB session.
* Editor/Debugger Window: The Editor/Debugger window is used to create and debug M-Files.
  + 1. **Common Block Libraries**

The most common used block libraries in communication system models.

1. Commonly Used Block
2. Continuous
3. Math Operation
4. Ports and Subsystem
5. Signal Routing
6. Sinks
7. Sources
   1. **Apparatus**

* MATLAB Software
  1. **Algorithm**
     1. **Set up the matrix  . Interchange the 2nd and 3rd rows using one line of**

**code.Solution:**

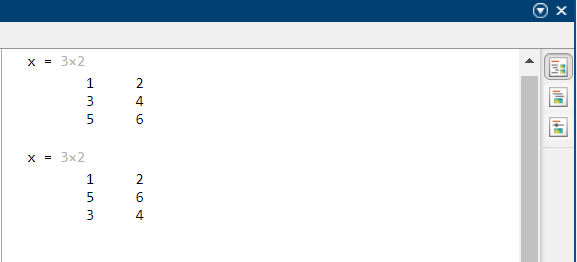
clc;

close all;

x =[1 2;3 4;5 6]

x ([2 3],:)=x([3 2],:)

**Output:**

****

* + 1. **Create a 10x10 matrix with random numbers between 0 and 10. Now, make all elements in the first row and first column equal to 1.**

**Solution:**

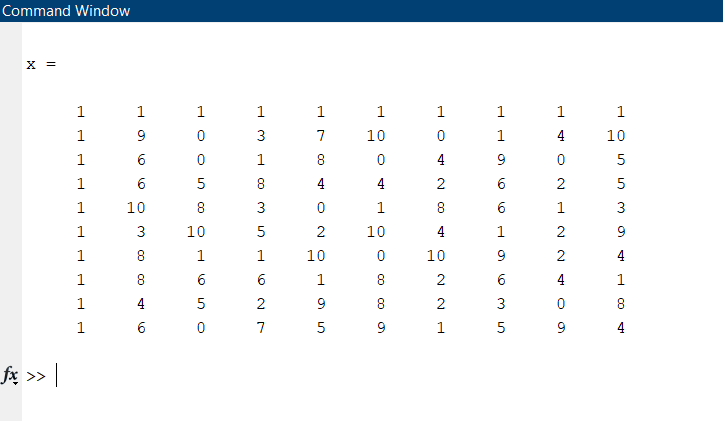
clc;

close all;

x=ones(10);

x(2:end,2:end)=randi([0 10],9)

**Output:**

****

* + 1. **We would like to create the row vector [8 6 4 2 0 K 0 0 10] with a total number of elements equal to 200 (that means there are 195 zeros in the vector). Think of two ways to create this variable without typing in all the numbers.**

**Solution:**

clc;

close all;

x = linspace(0,0,195);

A=[8 6 4 2 x 10];

Output\_1=A

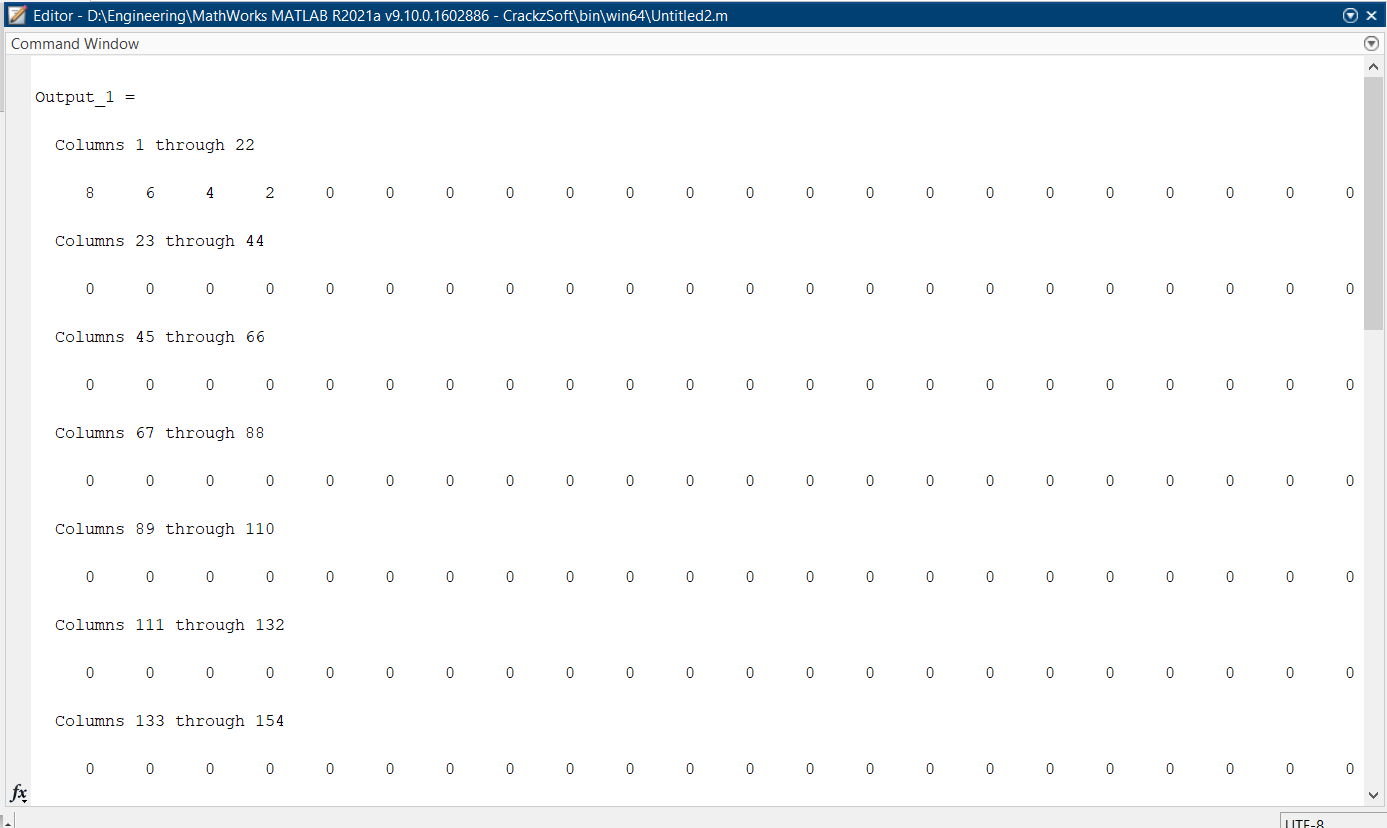
%%

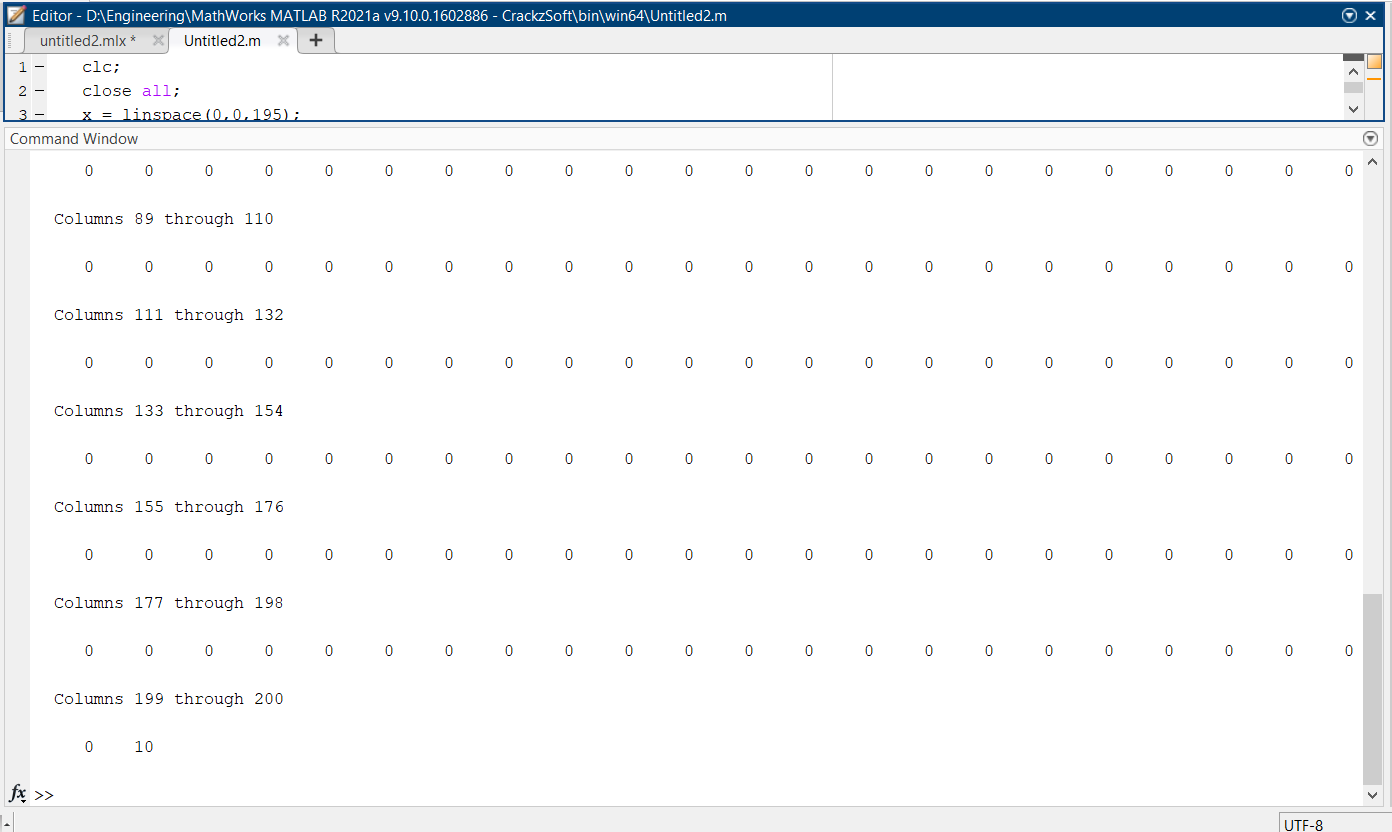
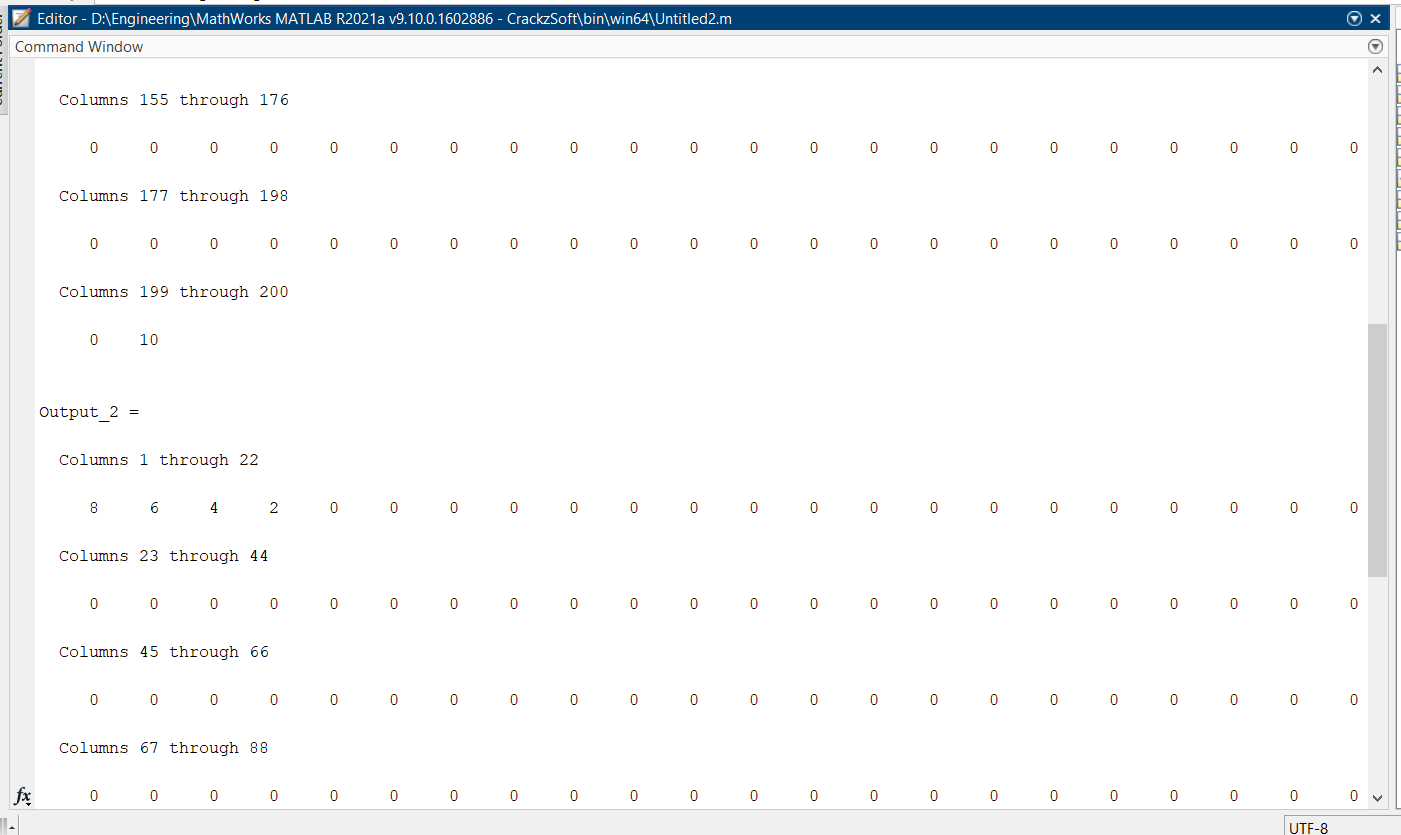
y=zeros(1,195);

B=[8 6 4 2 y 10];

Output\_2=B

**Output:**

****

****

* + 1. **Create two random vectors x and y, each with 5 elements. Write a for-loop to add x (1) to y (1), x (2) to y (2), etc. Each time, store the computed value in a variable called sumelements.**

**Solution:**

clc;

close all;

x=rand(1,5)

y=rand(1,5)

for i=1:1:5

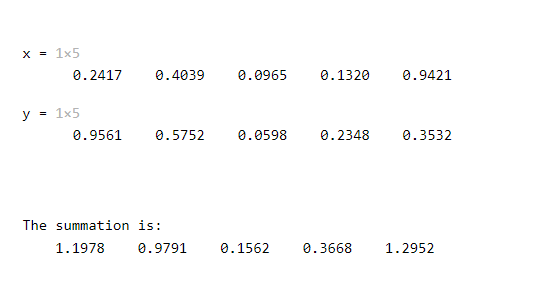
sumelements(i)=x(i)+y(i);

end

disp('The summation is: ')

disp(sumelements)

**Output:**

****

* + 1. **Write a while-loop that finds the index of the first element of x that is larger than the corresponding element of y. For example, if x = [0.1, 0.11, 0.05, 0.8, 0.91] and y = [0.83, 0.64, 0.09, 0.42, 0.5], you should find that the required index is 4, because x (4) > y (4) but the first three elements of x are all smaller than the corresponding elements of y.**

**Solution:**

clc;

close all;

x=rand(1,5)

y=rand(1,5)

z=6;

disp('Index are: ')

while z>1;

z=z-1;

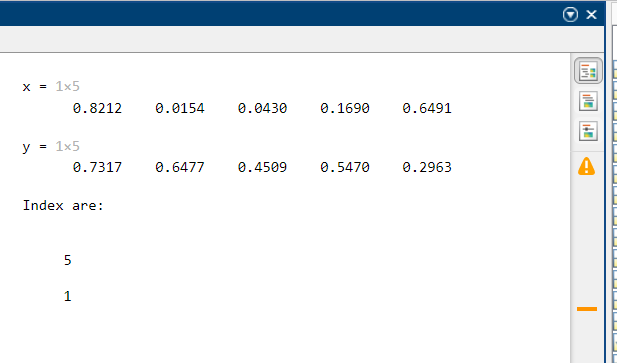
if x(z)>y(z)

disp(z)

end

end

**Output:**

****

* + 1. **Create two random numbers x and y. Write an if-statement that displays x if x is smaller than y. Now, write an if-else statement to display the minimum of x and y. Create a third random number z. Write an if-statement that displays the value of z if z is larger than x or y.**

**Solution:**

clc;

close all;

x=input('The value of x: ')

y=input('The value of y: ')

z=input('The value of z: ');

if x < y

disp('x is greater than y')

else

disp('y is greater than x')

end

if z>x||z>y

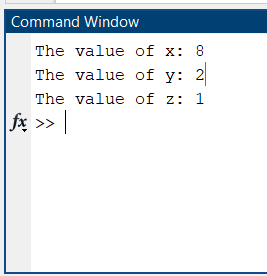
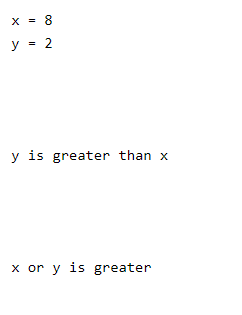
disp('z is greater than x and y')

else

disp('x or y is greater')

end

**Output:**

****

* + 1. **The height h(t) and speed S(t) of a projectile (such as a kicked soccer ball, or a rocket) launched with an initial speed S0 at an angle A to the horizontal are given by**

**h(t) = S0tsin(A)-0.5gt2**

**S(t)=√ (S0 2 − 2S0gtsin(A) + g2 t 2)**

**where g is the acceleration due to gravity. The projectile will strike the ground when h(t)=0 which gives the time to hit, thit =2(S0 / g) sin(A).**

**• Write a MATLAB program that computes the hit time for a given initial speed and angle, and displays the hit time and the speed of the rocket when it hits the ground.**

**• Follow the programming steps as discussed in the lecture.**

**• Store the program in a script file called rocket.m and run it**

**Solution:**

clc;

close all;

S=input('Speed is ')

t=input('Time is ')

A=input('Angle is ')

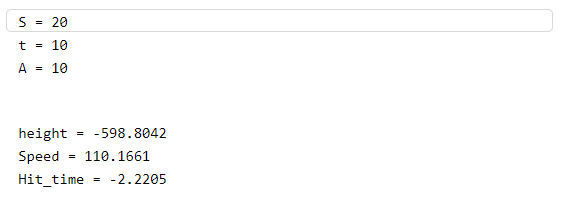
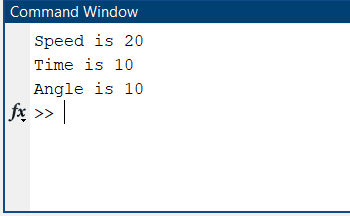
g=9.8;

height=S\*t\*sin(A)-0.5\*g\*t\*t

Speed=sqrt(S\*S-2\*S\*g\*t\*sin(A)+g\*g\*t\*t)

Hit\_time=2\*S\*sin(A)/g

**Output:**

****

* + 1. Create a function that computes the volume of a sphere with a certain radius. Call the function 'volsphere'. Now compute the volumes of the spheres with radii 0.1, 0.2, 0.3, through to 1.2 using a for-loop. Display the computed volumes. Store the program in a script file called volumes.m

**Solution:**

function volsphere(r)

v = (4/3)\*pi\*r^3;

return;

for r=0.1:0.1:1.2

volsphere(r)

end

**Output:**

** **

* + 1. Change the function and script file of task 1 so that you also compute and display the surface areas of the spheres.

**Solution:**

function s=surface(r)

s=(4\*pi\*r^2);

return;

clc;

close all;

for r=0.1:0.1:1.2

vol=volsphere(r);

disp('Volume')

disp(vol)

Area=surface(r);

disp('Area')

disp(Area)

end

**Output:**

** **

****

* 1. **Discussion & Conclusion**

In this experiment, we learned some MATLAB functions, resolved them in MATLAB, and displayed the results. Then worked with built-in functions and also generated ones as needed. In addition, learned to make of matrices, add them, and compare them. We also learned how to use MATLAB to solve mathematical difficulties. We also wrote functions and use them to solve math problems. All the programs run without a hitch.

In the end, the experiment was a success.