**Date: 04-09-2021**

**Experiment 4**

**Aim:** To work with MATLAB loops, logical arrays and vectorization.

**Apparatus:** MATLAB Software

**Objective:**

1. To learn how to execute a sequence of statements more than once using different loops like for loop, while loop.
2. To learn the different applications of logical arrays.
3. To learn the benefits of vectorization by comparing the same logical code using loop and vectorization.

**Problems:**

**Q-1.** Write an M-file to evaluate the equation y(x)=x2-3x+2 for all values of *x* between -1 and 3, in steps of 0.1. Do this twice, once with a for loop and once with vectors. Plot the resulting function using a 3-pointthick dashed red line.

**Code:**

clc;

clear all;

close all;

x = -1:0.1:3;

for i = -1:0.1:3

y = x.\*x - 3.\*x + 2;

end

disp(y);

plot(x,y,"LineStyle","--","Color","red","LineWidth",3)

figure

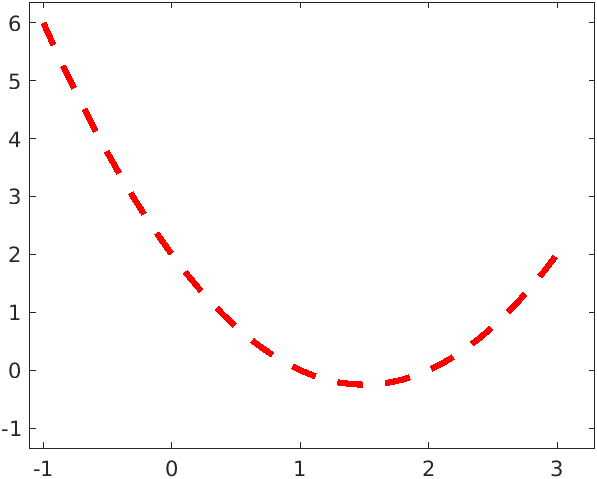
x1 = -1:0.1:3;

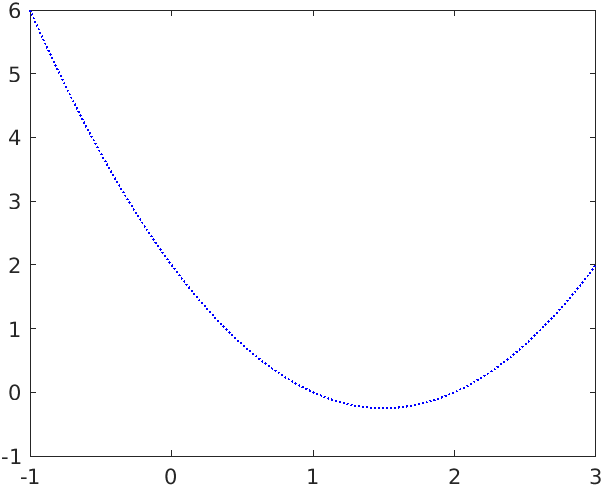
y1 = x.\*x - 3.\*x + 2;

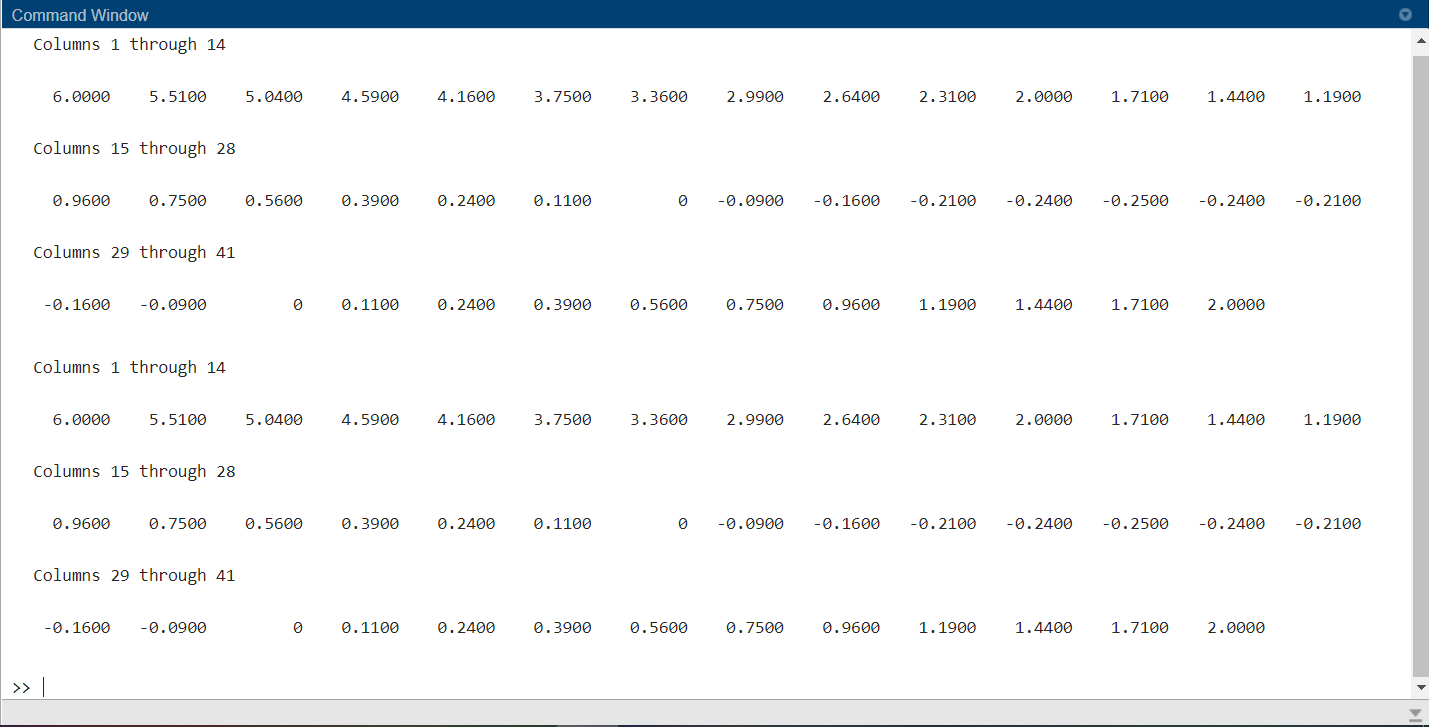
disp(y1);

plot(x1,y1,"LineStyle",":","Color","blue","LineWidth",1)

**Output:**

****

****



**Q-2.** Examine the following for statements and determine how many times each loop will be executed in MATLAB.

(*a*) for ii = -32768:32767

(*b*) for ii = 32768:32767

(*c*) for kk = 2:4:3

(*d*) for jj = ones(5,5)

**Code:**

clc;

clear all;

close all;

c = 0;

for i = -32768:32767

c = c +1;

end

fprintf("\n1. %d",c);

c = 0;

for i = 32768:32767

c = c +1;

end

fprintf("\n2. %d",c);

c = 0;

for i = 2:4:3

c = c +1;

end

fprintf("\n3. %d",c);

c = 0;

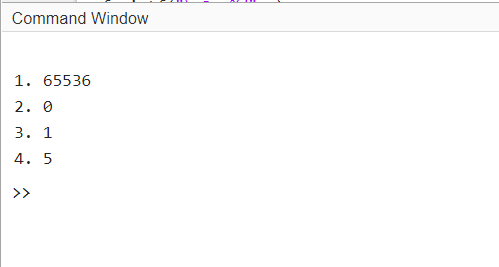
for i = ones(5,5)

c = c +1;

end

fprintf("\n4. %d",c);

**Output:**



**Q-3.** Examine the following for loops and determine the value of ires at the end of each of the loops, and also the number of times each loop executes.

(*a*) ires = 0;

for index = -10:10

ires = ires + 1;

end

(*b*) ires = 0;

for index = 10:-2:4

if index == 6

continue;

end

ires = ires + index;

end

(*c*) ires = 0;

for index = 10:-2:4

if index == 6

break;

end

ires = ires + index;

end

(*d*) ires = 0;

for index1 = 10:-2:4

for index2 = 2:2:index1

if index2 == 6

break

end

ires = ires + index2;

end

end

**Code:**

clc;

clear all;

close all;

fprintf("Sr.\tIres\tCount1\tCount2")

c = 0;

ires = 0;

for i = -10:10

ires = ires + 1;

c = c +1;

end

fprintf("\n1.\t%d\t%d",ires,c);

c = 0;

ires = 0;

for i = 10:-2:4

if i == 6

c = c +1;

continue;

end

ires = ires + i;

c = c +1;

end

fprintf("\n2.\t%d\t%d",ires,c);

c = 0;

ires = 0;

for i = 10:-2:4

if i == 6

c = c +1;

break;

end

ires = ires + i;

c = c +1;

end

fprintf("\n3.\t%d\t%d",ires,c);

c1 = 0; c2 = 0;

ires = 0;

for i1 = 10:-2:4

for i2 = 2:2:i1

if i2 == 6

break

end

ires = ires + i2;

c2 = c2 +1;

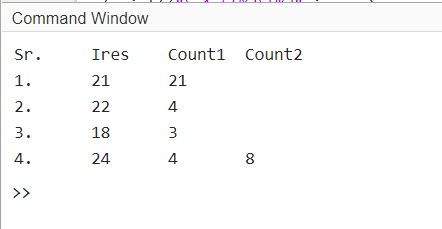
end

c1 = c1 +1;

end

fprintf("\n4.\t%d\t%d\t%d",ires,c1,c2);

**Output:**



**Q-4.** Examine the following while loops and determine the value of ires at the end of each of the loops and the number of times each loop executes.

(*a*) ires = 1;

while mod(ires,10) ~= 0

ires = ires + 1;

end

(*b*) ires = 2;

while ires <= 200

ires = ires^2;

end

(*c*) ires = 2;

while ires > 200

ires = ires^2;

end

**Code:**

clc;

clear all;

close all;

fprintf("Sr.\tIres\tCount")

c = 0;

ires = 1;

while mod(ires,10) ~= 0

ires = ires + 1;

c = c +1;

end

fprintf("\n1.\t%d\t%d",ires,c);

c = 0;

ires = 2;

while ires <= 200

ires = ires^2;

c = c +1;

end

fprintf("\n2.\t%d\t%d",ires,c);

c = 0;

ires = 2;

while ires > 200

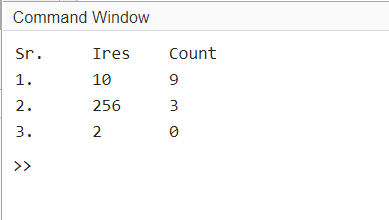
ires = ires^2;

c = c +1;

end

fprintf("\n3.\t%d\t%d",ires,c);

**Output:**



**Q-5.** What is contained in array arr1 after each of the following sets of statements have been executed in MATLAB?

(*a*) arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];

mask = mod(arr1,2) == 0;

arr1(mask) = -arr1(mask);

(*b*) arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];

arr2 = arr1 <= 5;

arr1(arr2) = 0;

arr1(~arr2) = arr1(~arr2).^2;

**Code:**

clc;

clear all;

close all;

arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];

mask = mod(arr1,2) == 0;

arr1(mask) = -arr1(mask);

fprintf("1.----------arr1---------- \n")

disp(arr1);

arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];

arr2 = arr1 <= 5;

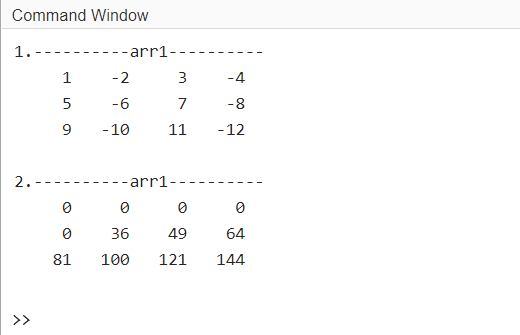
arr1(arr2) = 0;

arr1(~arr2) = arr1(~arr2).^2;

fprintf("2.----------arr1---------- \n")

disp(arr1);

**Output:**



**Q-6.** Write a MATLAB program to evaluate the function

for any user-specified value of *x*, where ln is the natural logarithm (logarithm to the base *e*). Write the program with a while loop, so that the program repeats the calculation for each legal value of *x* entered into the program. When an illegal value of *x* is entered, terminate the program. (Any x>=1 is considered an illegal value.)

**Code:**

clc;

clear all;

close all;

while (true)

s = "\n\nEnter Your X value: ";

x = input(s);

if x>=1

fprintf("\nInvalid X value (Illegal)");

break;

else

y = log(1/(1-x));

fprintf("\nY is: %f",y);

end

end

**Output:**



**Q-7.** The *n*th Fibonacci number is defined by the following recursive equations:

f(1)=1;

f(2)=2;

f(n)=f(n-1)+f(n-2)

Therefore, f(3)=f(2)+f(1)=2+1=3, and so forth for higher numbers. Write an M-file to calculate and write out the *n*th Fibonacci number for n>2, where *n* is input by the user. Use a while loop to perform the calculation.

**Code:**

clc;

clear all;

close all;

s = "Enter Your N value: ";

n = input(s);

a = 0;

b = 1;

fprintf("%d %d ",a,b);

i = 3;

sum = 0;

while (i<=n)

c = a+b; %performs add operation on previous two values

fprintf("%d ",c); % It prints from third value to given length

a=b;

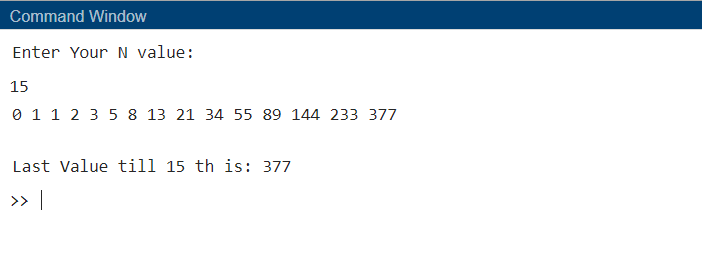
b=c;

i=i+1;

end

fprintf("\n\nLast Value till %d th is: %d ",n,c);

**Output:**



**Q-8.** The current flowing through the semiconductor diode shown in Figure 4.7 is given by the equation

Where, voltage across the diode, in volts

current flow through the diode, in amps

leakage current of the diode, in amps

q = charge on an electron, 1.602 x 10-19 coulombs

k = Boltzmann’s constant, 1.38 x 10-23 joule/K

T = temperature, in kelvins (K)

The leakage current of the diode is 2.0 µA. Write a program to calculate the current flowing through this diode for all voltages from -1.0 V to +0.6 V, in 0.1 V steps. Repeat this process for the following temperatures: 75°F and 100°F, and 125°F. Create a plot of the current as a function of applied voltage, with the curves for the three different temperatures appearing as different colors.

**Code:**

clc;

clear all;

close all;

t=[75,100,125];

format long;

color=['r','g','k'];

c=1;

I0=2 \* 10^-6;

V=-1:0.1:0.6;

k=1.38 \* 10^-23;

q=1.602 \* 10^-19;

for i=t

i=(i - 32) \* 5/9 + 273.15;

temp=(q/(k\*i))\*V;

I=I0\*(exp(temp)-1);

plot(V,I,color(c));

hold on

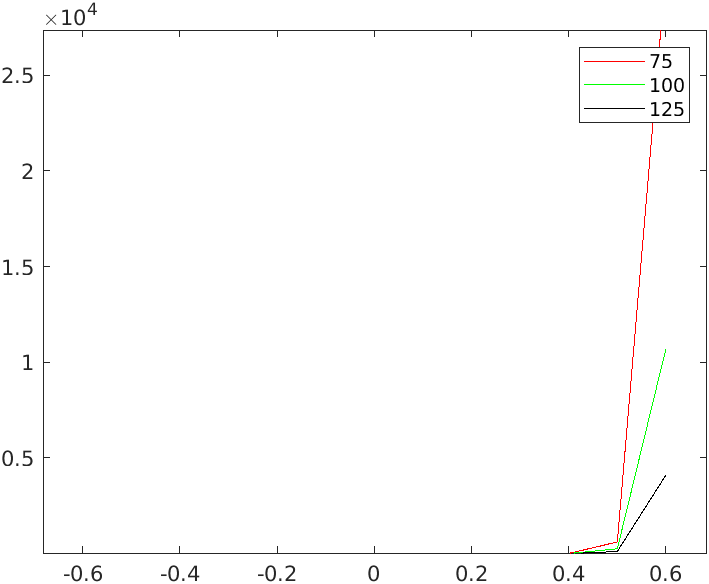
c=c+1;

I=[];

end

legend('75','100','125');

**Output:**



**Q-9.** Engineers often measure the ratio of two power measurements in

*decibels*, or dB. The equation for the ratio of two power measurements

in decibels is

where is the power level being measured and is some reference power level. Assume that the reference power level is 1 watt, and write a program that calculates the decibel level corresponding to power levels between 1 and 20 watts, in 0.5 W steps. Plot the dB-versus-power curve on a log-linear scale.

**Code:**

clc;

clear all;

close all;

p1 = 1;

p2 = 1:0.5:20;

dB = 10 .\* log10(p2./p1)

figure(1)

plot(p2,dB);

ylabel("Decibal");

xlabel("Power");

title("Normal Graph");

figure(2)

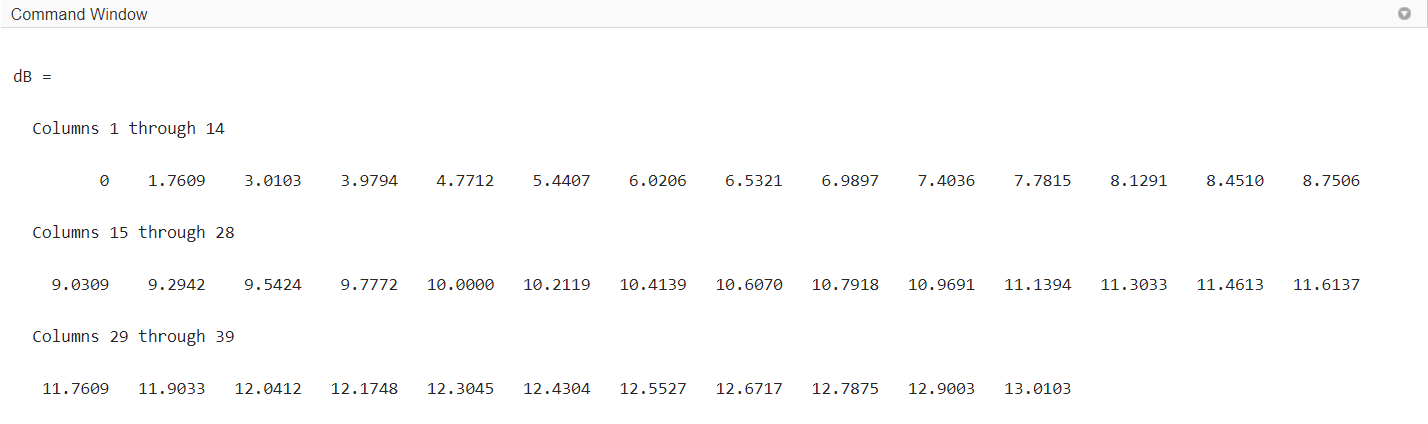
semilogx(p2,dB);

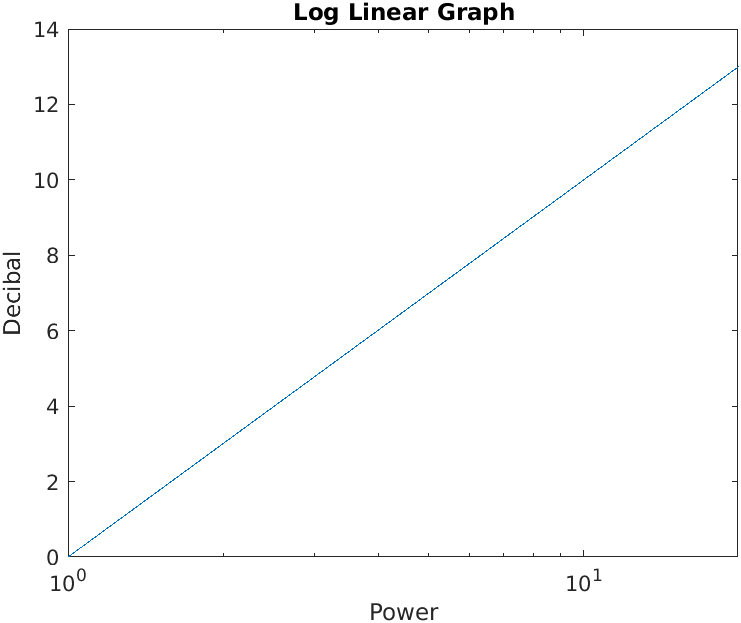
ylabel("Decibal");

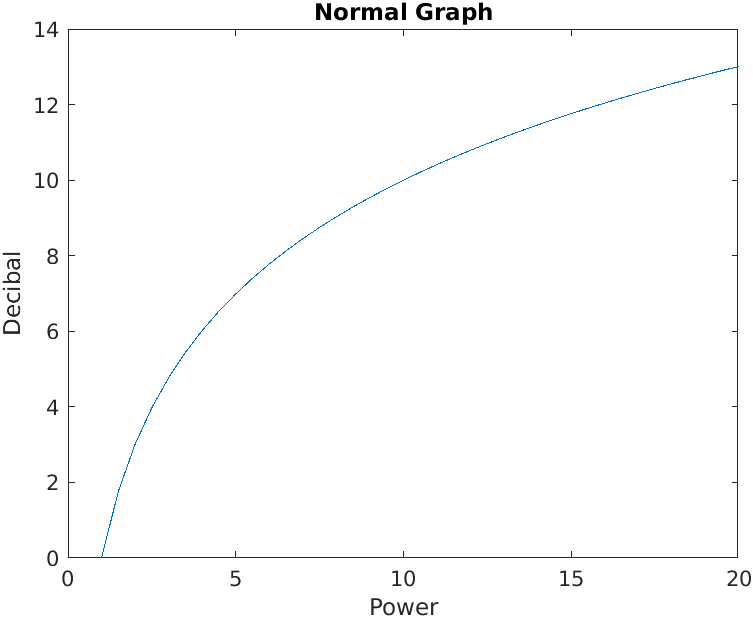
xlabel("Power");

title("Log Linear Graph");

**Output:**







**Conclusion:**

From this experiment we came to learn different types and ways to plot graphs, use for and while loops. As using for loops we came to understand how we have to provide different types of conditions in for loop. We even got acquainted with break and continue statements. We learnt about the while iterative loops also.