**Experiment 1**

**Creating a One-Dimensional Array (Row / Column Vector); Creating a two-dimensional array (Matrix of given size) and**

**(A) Performing Arithmetic Operations: Addition, Subtraction, Multiplication and Exponentiation.**

**(B) Performing Matrix operations -Inverse, Transpose, Rank.**

**Tool Used:** MATLAB 7.0 / Octave Online.

**Theory:** MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. It is a high-performance language for technical computing. It integrates, computation, visualisation, and programming in an easy to use environment where problems and solutions are expressed in a familiar mathematical notation. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. It is used in Math and computation, Algorithm development, Data acquisition, Modelling, simulation and prototyping, Data analytics and, exploration, and visualisation, Scientific and engineering graphics, Application development includes graphical user interface building.

Basic Theory of Experiment:

Matrix Addition and Subtraction is merely adding or subtracting the matrix elements by their corresponding elements in the other matrix.

Consider two matrixes A and B is an m x n matrix and b is a n x p matrix, they could be multiplied together to produce an m x n matrix C. Matrix multiplication is a possibly only if the number of columns n in A is equal to the number of rows and columns in the second matrix. Each element in the (i,j)thposition , in the resulting matrix C, is the summation of the products of elements in the ithrow of the first matrix with the corresponding element in the jthcolumn of the second matrix. In MATLAB, the matrix multiplication is performed by using the \* operator.

The inverse of a matrix does not always exist. If the determinant of the matrix is zero then the inverse does not exist and the matrix is singular. Inverse of matrix is given by inv(A).

Transpose operations switches the rows and columns in a matrix. It is represented by a singular quote (’).

Rank function provides an estimate of the number of linearly independent rows and columns. k=rank(A,tol) returns the number of the singular values of A that are larger than tol.

**INPUT-**

**% Algebraic operations in a matrix%**

A= [ 2 5 4; 8 7 5; 0 3 6]

B= [ 6 6 8; 1 2 2; 5 7 1]

**% Addition and Subtraction**

C=A+B

D=A-B

**%Multiplication of two matrices**

E=A\*B

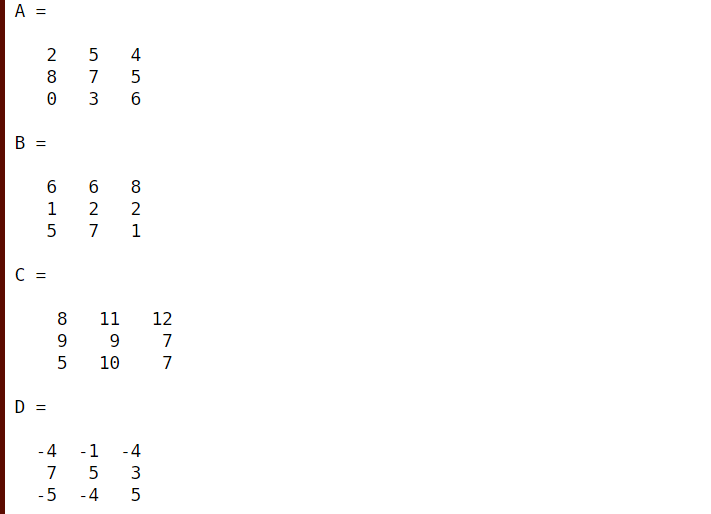
**%Division of two matrices**

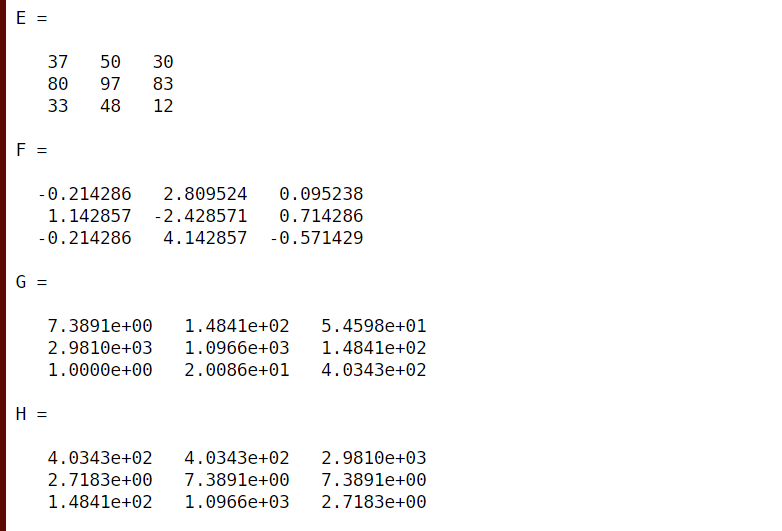
F=A/B

G=exp(A)

H=exp(B)

**OUTPUT-**

****

****

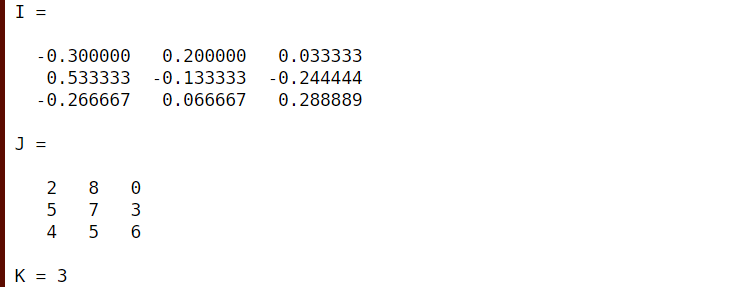
**INPUT-**

I=inv (A)

J=A’

K=rank(A)

**OUTPUT-**

****

**Experiment-2**

**(A) Performing Matrix manipulations- Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a vertical/ horizontal axis.**

**(B) Creating two arrays and performing: Relational operations: >, <, ==, <=,>= and   Logical Operations: &, |, XOR**

**Tools Used:** MATLAB 7.0/Octave Online.

**Theory:** An operator is a symbol that tells the compiler to perform certain mathematical and logical manipulations. MATLAB is designed to operate primarily o n whole matrices and arrays. MATLAB allows four types of elementary operations:

* Arithmetic Operations
* Logical Operations
* Relational Operations
* Bitwise Operations
* Set Operations

Arithmetic Operations: MATLAB allows two different types of arithmetic operations that are:

* Matrix Arithmetic Operations
* Array Arithmetic Operations

Matrix arithmetic are same as defined as in linear algebra and array operations are executed element by element, both on one dimensional and multi-dimensional array.

Matrix operators and array operators are differentiated by the period (.) symbol. However, as the addition and subtraction operation is same for matrices and arrays, the operator is same for the both cases.

|  |  |
| --- | --- |
| Operator | Description |
| + | Addition and unary plus. A+B adds A and B. A and B must have the same size, unless one is a scalar. A scalar can be added to a matrix of any size. |
| - | Subtraction or unary minus. A-B subtracts B from A. A and B must have the same size, as one is scalar. A scalar can be subtracted from the matrix of any size. |
| \* | Matrix multiplication. C=A\*B is the linear algebraic product of the matrices A and B. For a non-scalar matrix, the no of columns A must be equal to the no of rows of B. A scalar can multiply the matrix of any size. |
| .\* | Array multiplication. A.\*B is the element by the element of the product of arrays A and B. A must have the same size as one is scalar. |
| / | Backslash or matrix left division. If A and B is the square matrix then A and B as the inv(A)\*B , except it is computed in a different way. If A is a n by n matrix and Bis a column vector with n components or the matrix with such columns, then X=A/B is the solution to the equation AX=B. A warning message will display if A is badly scalar or nearly scalar. |
| ./ | Array left division. A./B is the matrix with the elements B(i,j) and A(I,j). A and B must have the same size, unless one of them is scalar. |
| ^ | Matrix power. X^p is X is the power of p, if p is the scalar. If p is an integer then power computed by repeating squaring. If the integer is negative then, X is inverted first. For other values of p, the calculation involves eigen vectors and eigen values such that if [V,D]=eig(X), then X^p=V\*D.^p/V. |

Relational Operators: It can also work for both scalar and non scalar data.  Relational operators perform element by element comparisons on the two arrays and return the logical array of the same size where the element is to be logical (1) true where it is true and else logical (0) false where the element is false. The following element shows the relational operator available in the MATLAB:

|  |  |
| --- | --- |
| Operator | Description |
| == | Equal To |
| ~= | Not Equal To |
| >= | Greater Than Equal To |
| <= | Less Than Equal To |
| > | Greater Than |
| < | Less Than |

Logical Operators: MATLAB offers two types of logical operators and functions:

* Element-wise: These operators operate on corresponding elements of logical arrays. In other words we can say, these operators operate element-by-element on logical arrays. The symbols &, |, and ~ are the logical array operators AND, OR, and NOT.
* Short-circuit: these operators operate on scalar, logical expressions. Short-circuit logical operators allow short-circuiting on scalar logical expressions and operations. The symbols && and || are the logical short-circuit operators AND and OR.

Bitwise Operator: Bitwise operator works on bits and performs bit-by-bit operation. MATLAB provides various functions for bit-wise operations like 'bitwise and', 'bitwise or' and 'bitwise not' operations, shift operation, etc.

Concatenating Matrices: In MATLAB one can concatenate two matrices to create a larger matrix. The pair of square brackets '[]' is the concatenation operator. MATLAB allows two types of concatenations −

* Horizontal concatenation
* Vertical concatenation

When you concatenate two matrices by separating those using commas, they are just appended horizontally. It is called horizontal concatenation. Alternatively, if you concatenate two matrices by separating those using semicolons, they are appended vertically. It is called vertical concatenation.

**INPUT-**

%Concatenation

A=ones (3,6)

B=zeros (3,6)

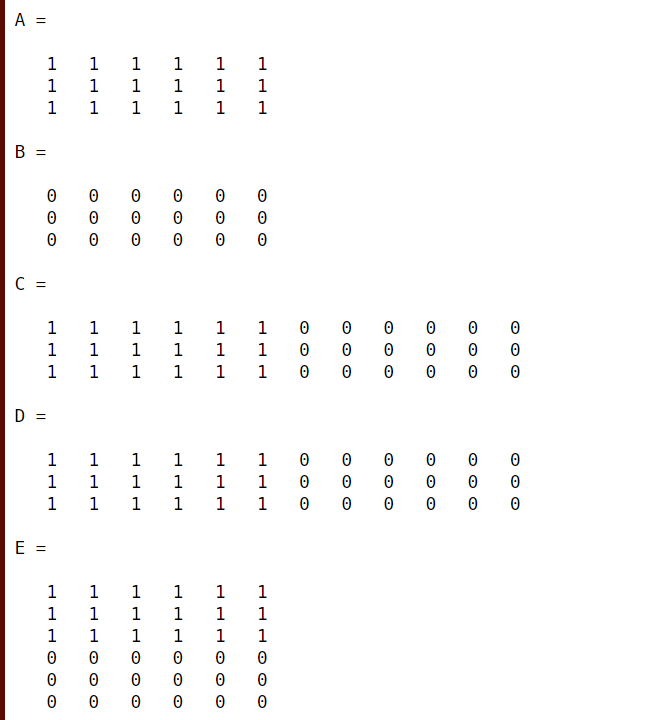
C= [A B]

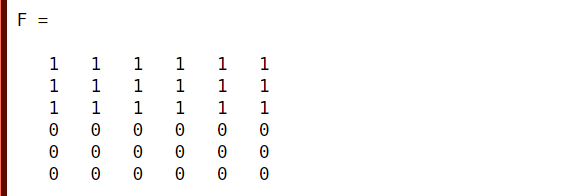
D=horzcat (A, B)

E= [A; B]

F=vertcat (A, B)

**OUTPUT-**

****

****

**INPUT-**

%Indexing

G= [16 75 87 13 22 66 62 80 94]

H=G (6)

I=G ([2 6 7])

J=G (5:8)

K=1:3:80

l=G ([5:9 1:4])

M=G (3: end-1)

N=G (1:2: end)

G ([1 5 9]) = [31 62 75]

G1= [1 1 7; 6 8 3; 4 6 1]

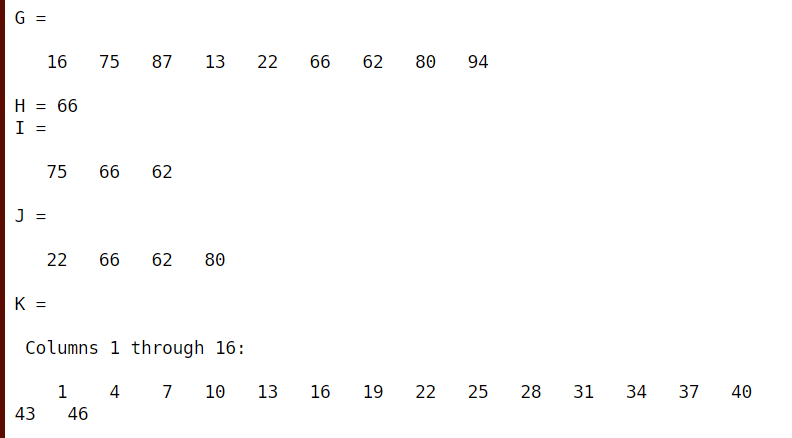
H1=G1(1,3)

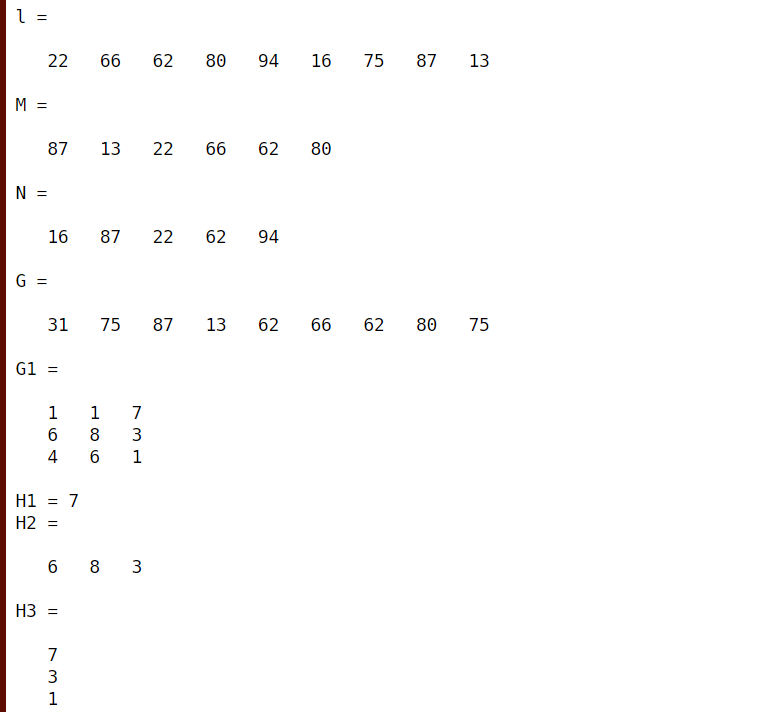
H2=G1(2, :)

H3=G1(:,3)

H4=G1(:,end)

**OUTPUT-**

****

****

****

%Sorting

A1= [45 6 98 33 5 34 79 19]

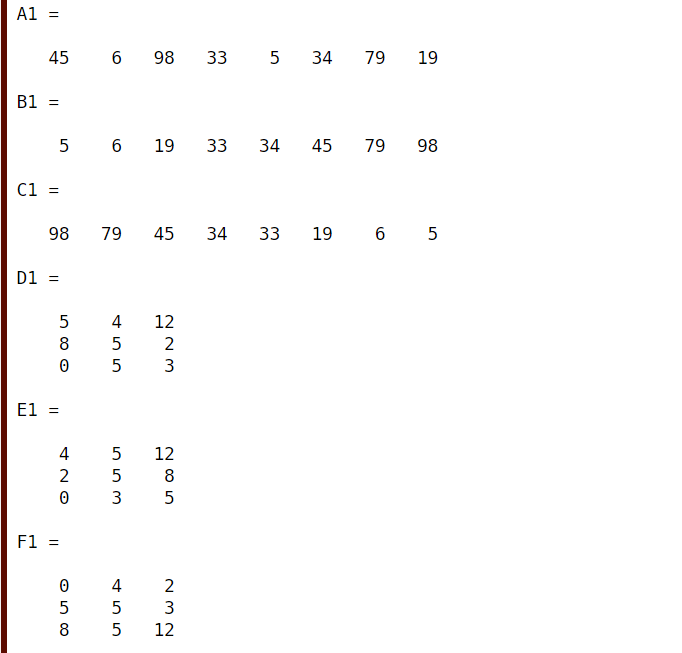
B1=sort(A1)

C1=sort(A1,'descend')

D1= [5 4 12; 8 5 2; 0 5 3]

E1=sort(D1,2)

F1=sort(D1,1)

****

%Shifting

A2=rand (3,3)

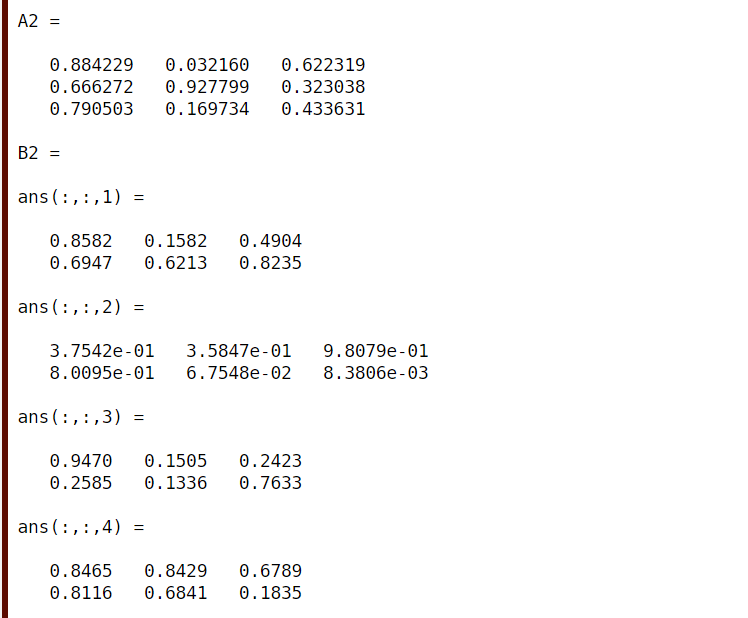
B2=rand (2,3,4)

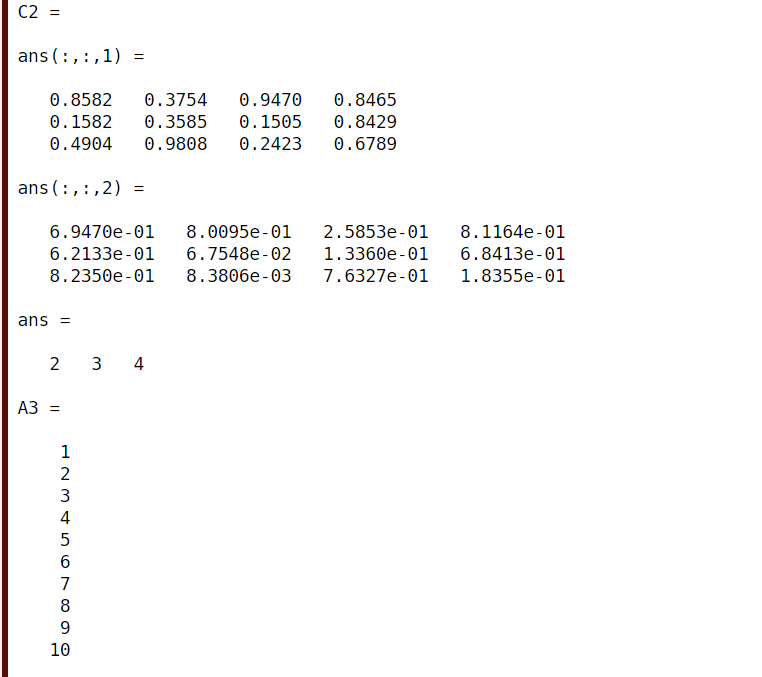
C2=shiftdim(B2,1)

size(B2)

A3= (1:10)'

B3=circshift(A3,5)

****

****

****

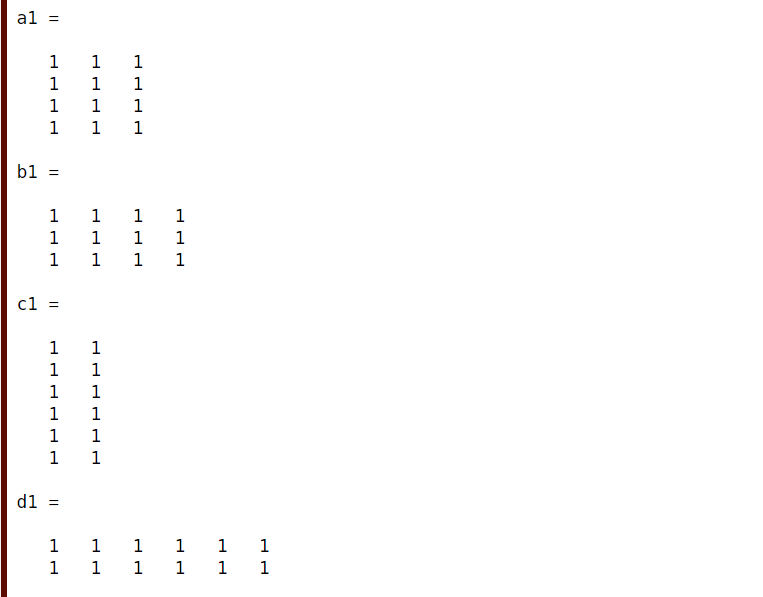
%Reshaping

a1=ones (4,3)

b1=reshape(a1,3,4)

c1=reshape(a1,6,2)

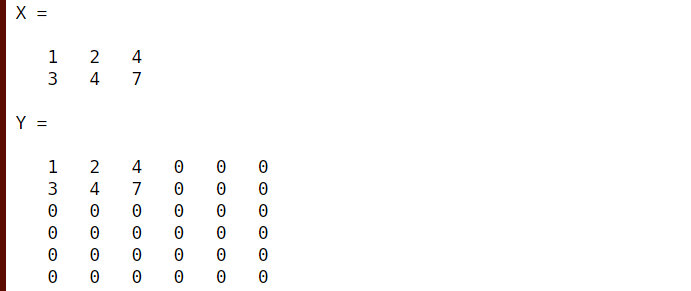
d1=reshape(a1,2,6)

****

%Resizing

X= [1 2 4; 3 4 7]

Y=resize(X,6)

****

%Flipping

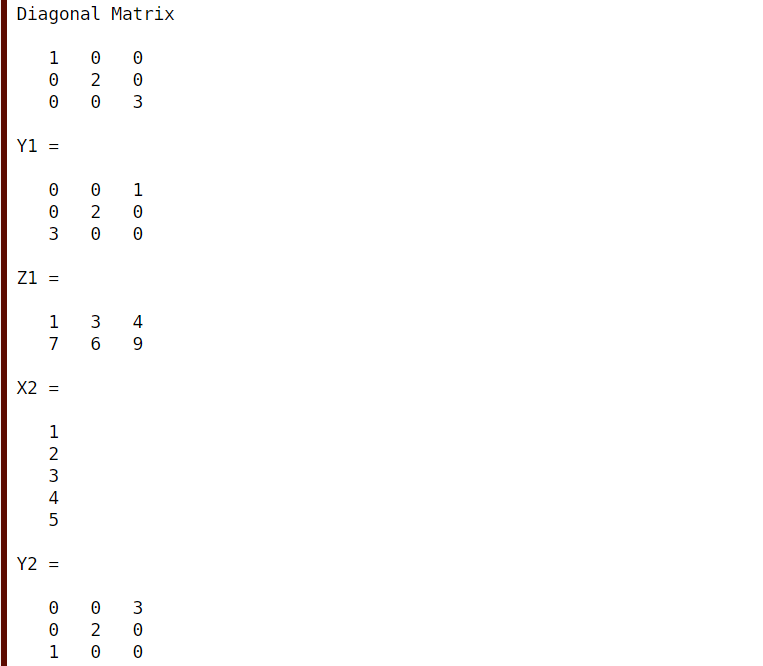
X1=diag ([1 2 3])

Y1=flip(X1,2)

Z1= [1 3 4 ;7 6 9]

X2= (1:5)'

Y2=rot90(X1)

****

%Relational operators

a1= [4 5 7 ;9 4 5]

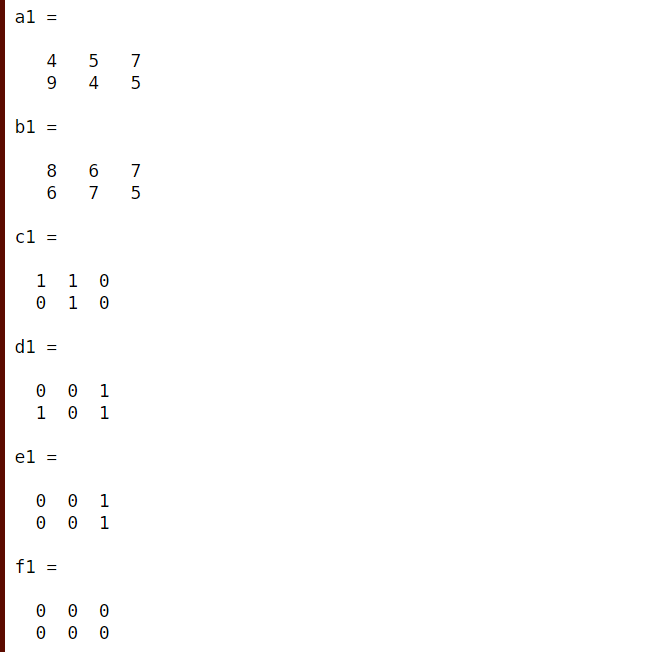
b1= [8 6 7 ;6 7 5]

c1=a1<b1

d1=a1>=b1

e1=a1==b1

f1=a1<0 & b1<0

****

%Logical operators

N1= [0 0 1;1 0 1]

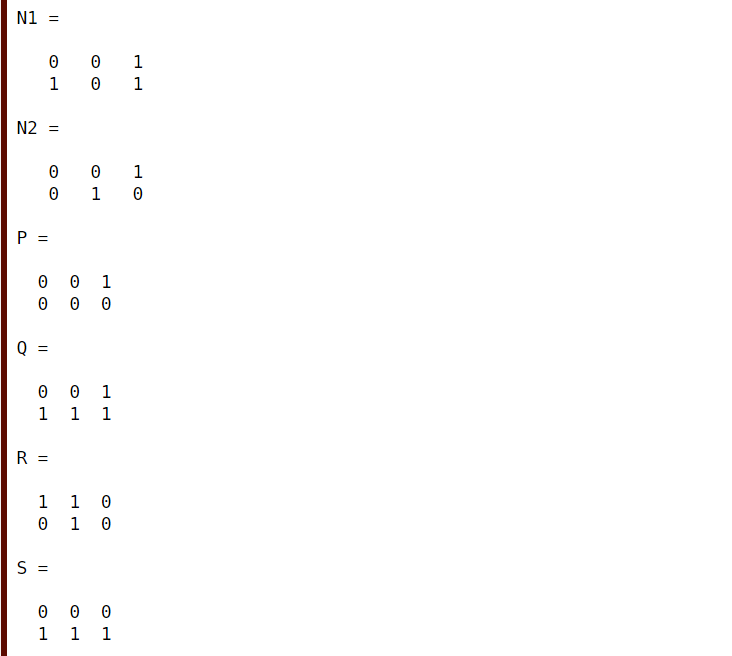
N2= [0 0 1;0 1 0]

P=N1 & N2

Q=N1 | N2

R=not(N1)

S=xor(N1, N2)

****

**Experiment-3**

**(A)Generating a set of commands on a given vector. Add up the values of the elements (check   sum)**

**(B)Compute the running sum (check with sum), where running sum for element i=the sum of the elements from 1 to i inclusive. Also, generating a random sequence using rand ()/randn () functions and plotting them.**

**Tools Used:** MATLAB7.0/Octave Online.

**Theory:** When you create random numbers using software, the results are not random in a strict, mathematical sense. However, software applications, such as MATLAB®, algorithms that make your results appear to be random and independent. The results also pass various statistical tests of randomness and independence. This apparently random and independent numbers are often described as pseudorandom and pseudo independent. You can use these as if they are truly random and independent. One benefit of using pseudorandom, pseudo independent numbers is that you can repeat you can repeat a random number calculation at any time. This can be useful in testing or diagnostic situations. All the random number functions, rand, randn draw values from a shared random number generator.  Every time you start MATLAB generator resets itself to the same state. Therefore, a command such as rand (2,2) returns the same result anytime you execute it immediately following start-up. Also, any script or function that calls a random number of functions return the same result whenever you restart.

Some More Functions:

If X= [1 2 3 4 5] and Y= [1 2 3;4 5 6;7 8 9]

•A=sum(X);

It is used to find the sum of all the elements of the matrix X.

•B=sum (X (2:4));

It is used to find the sum of elements from the 2nd element to the 4th element of the matrix X.

•C=cumsum(Y,1);

It is used to find the cumulative sum of the columns of the matrix Y.

•D=cumsum(Y,2);

It is used to find the cumulative sum of the rows of the matrix Y.

•E=sum(Y,1);

It is used to find the sum of the columns of the matrix Y.

•F=sum(Y,2);

It is used to find the sum of the rows of the matrix Y.

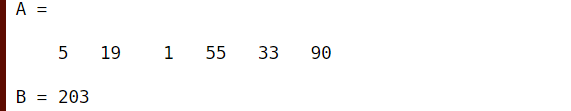
**INPUT-**

%Sum

A= [5 19 1 55 33 90]

B=sum(A)

**OUTPUT-**



**INPUT-**

%Running sum

sum1=0

    for i=1:6

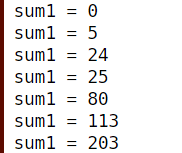
        sum1=sum1+A(i);

        i=i+1;

        sum1

    end

**OUTPUT-**



**INPUT-**

%Cumsum

C= [ 12 3 1; 0 6 2; 6 1 1]

D=cumsum(C)

E=cumsum(C,2)

**OUTPUT-**



**%plot random**

F=rand (4,5)

G=randi (4,5,3)

plot(F)

grid on;

title ('Graph of F')

w=-pi:0.01: pi

x=sin(w);

plot(w,x,'r')

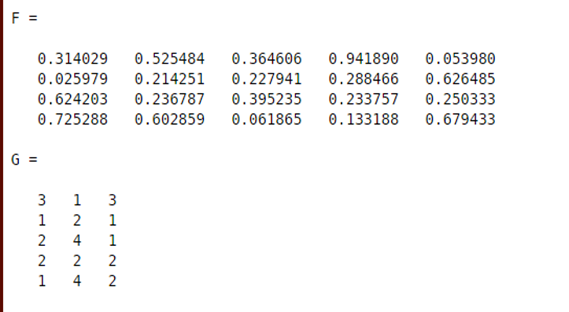
hold on

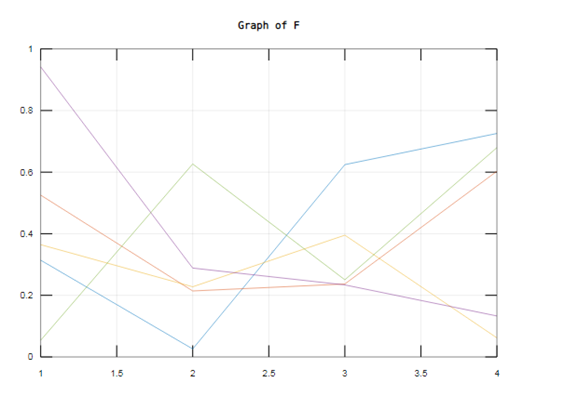
y=cos(w);

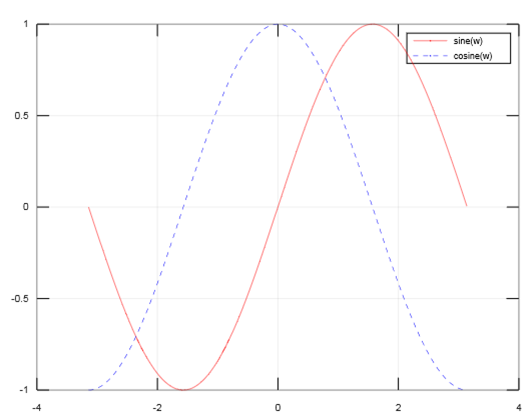
plot(w,y,'b--')

grid on

legend('sine(w)','cosine(w)')







**Experiment-4**

**(A) Evaluating a given expression and rounding it to nearest integer value using Round, Floor, Ceil and Fix functions**

**(B) Also, generating plots of**

**Trigonometric functions: sin(x), cos(x), tan(x), sec(x), cosec(x), cot(x)**

**Logarithmic functions:  log(a), log10(a), square root of ‘a’, real nth root of ‘a’.**

**APPARATUS**: Octave online, MatLab

**THEORY:**

* Round () rounds up the given number.
* Ceil () rounds to the next higher integer.
* Floor () rounds to the next lower integer.
* Fix () rounds to the integer next closest to zero.
* subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB numbers subplot positions by row. The first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on. If axes exist in the specified position, then this command makes the axes the current axes.
* The plot function plots Y versus X. If X and Y are both matrices, then they must have equal size. The plot function plots columns of Y versus columns of X. If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length.

**INPUT-**

a= [2.0 2.5 -1.9 4.6]

a1=round(a)

a2=floor(a)

a3=ceil(a)

a4=fix(a)

x=0: pi/5:3\*pi;

f1=sin(x);

subplot (3,3,1)

plot (x, f1,'r-o-')

grid on;

xlabel('x')

ylabel('sine x')

f2=cos(x);

subplot (3,3,3)

plot (x, f2,'p-+-')

grid on;

xlabel('x')

ylabel('Cosine x')

f3=tan(x);

subplot (3,3,4)

plot (x, f3,'b-+-')

grid on;

xlabel('x')

ylabel('tangent x')

f4=sec(x);

subplot (3,3,6)

plot (x, f4,'c')

grid on;

xlabel('x')

ylabel('secant x')

f5=csc(x);

subplot (3,3,7)

plot (x, f5,'r--')

grid on;

xlabel('x')

ylabel('cosecant x')

f6=cot(x);

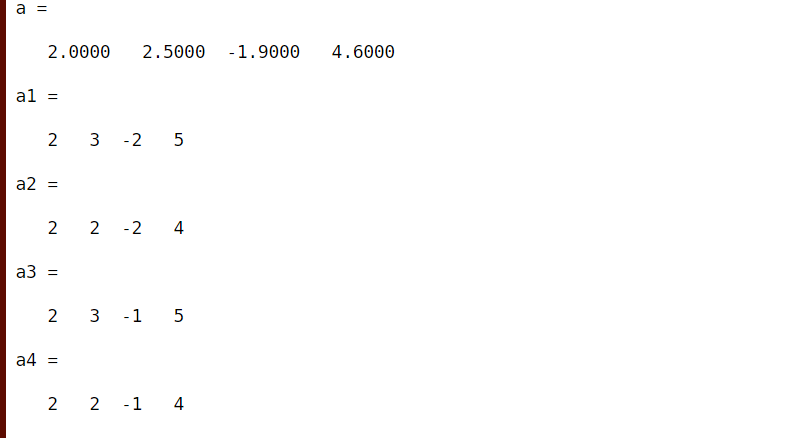
subplot (3,3,9)

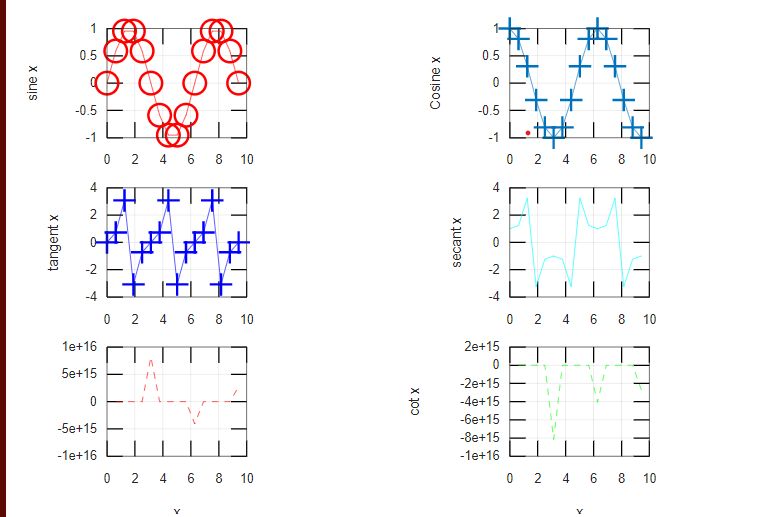
plot (x, f6,'g--')

grid on;

xlabel('x')

ylabel('cot x')

****

****

clear all;

close all;

clc;

a=[1 6 9; 5 4 2; 1 9 8]

b=log(a)

plot(b,'r--o')

hold on;

c=log10(a)

plot(c,'g+-')

d=sqrt(a)

grid on;

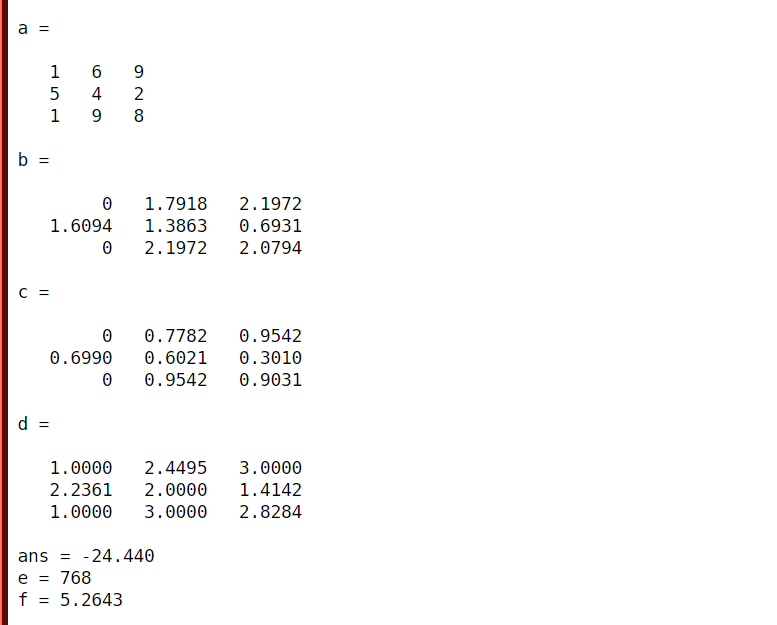
legend("log","log10")

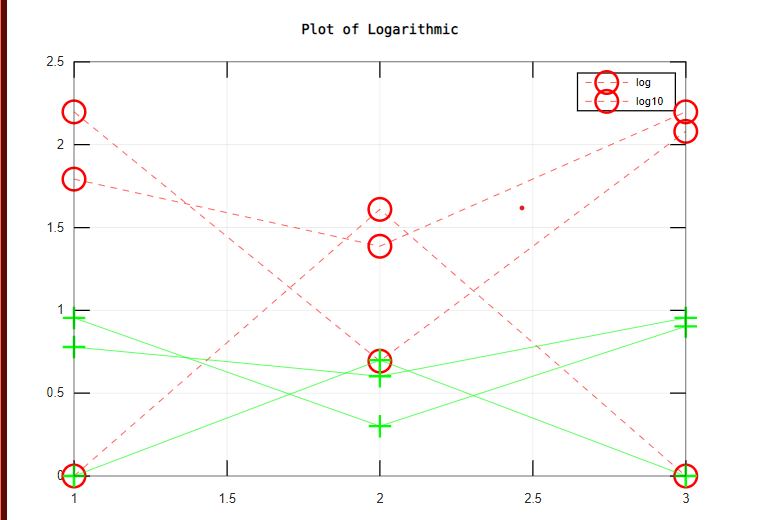
title("Plot of Logarithmic")

e=768

f=nthroot(e,4)

**OUTPUT-**

****

****

**EXPERIMENT-5**

**Aim-** Creating a vector X with elements, Xn = (-1)n+1/(2n-1) and Adding up 100 elements of the vector, X; And, plotting the functions, x, x3, ex, exp(x2) over the interval 0 < x < 4 (by choosing appropriate mesh values for x to obtain smooth curves), on A Rectangular Plot.

**APPARATUS**: Octave online, MatLab.

**THEORY:**

• For a vector, clim can be used to specify limits.

• Hold on is used to plot more than one function in the same graph.

• Subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB numbers subplot positions by row. The first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on. If axes exist in the specified position, then this command makes the axes the current axes.

• The plot function plots Y versus X. If X and Y are both matrices, then they must have equal size. The plot function plots columns of Y versus columns of X . If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length.

**PROGRAM-**

clc;

close all;

clear all;

n = [0.04:0.04:4];

X = (-1). ^(n+1). /((2\*n) +1);

figure ('name','graph of x, x^2 and x^3');

plot (n, X.+ 3, 'g-', 'markersize', 2);

hold on

plot (n, X.^2, 'm+-', 'markersize', 2)

hold on

plot (n, X.^3.-3, 'c\*-', 'markersize', 2)

hold on

legend('x','y','y1')

grid on;

title ('plot for x, x^2, x^3')

xlabel('n-->')

ylabel('X(n)')

figure ('name','graph of exp(x), exp(x^2) and exp(x^3)');

plot (n, exp(X). - 3, 'g+-', 'markersize', 2);

hold on;

plot (n, exp (X.^2). - 6, 'go-', 'markersize', 2);

hold on;

plot (n, exp (X.^3). - 9, 'b+-', 'markersize', 2);

legend('exp(x)','exp(x^2)','exp(x^3)');

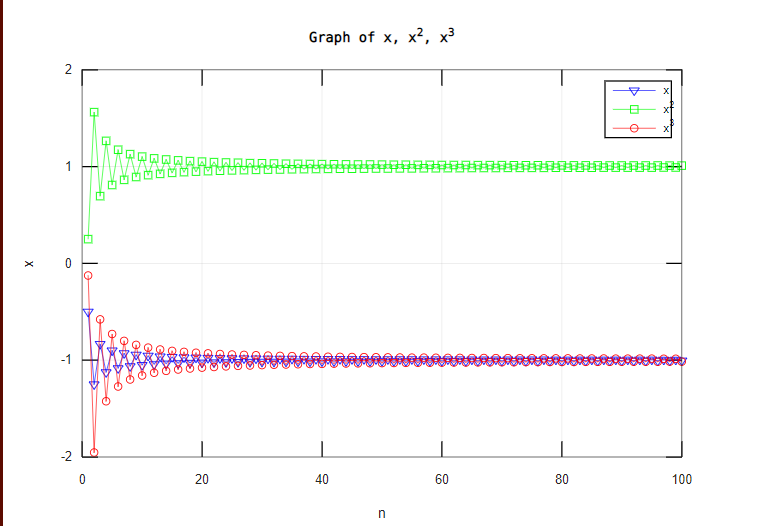
grid on;

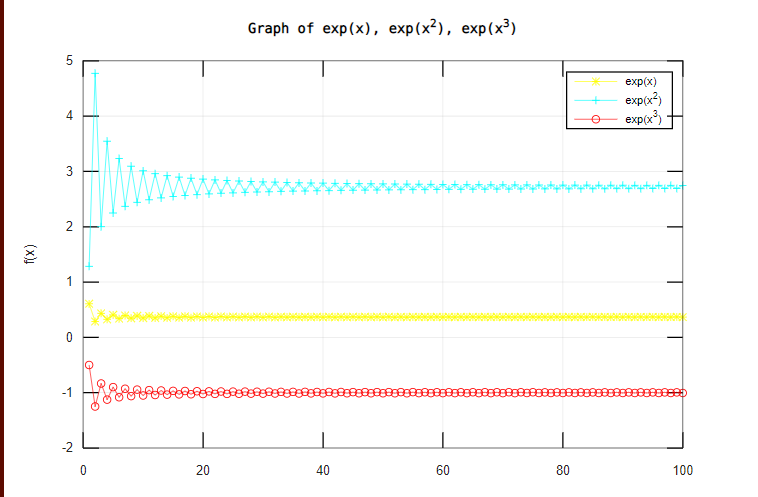
title ('plot for exp(x), exp(x^2), exp(x^3)');

xlabel('n-->');

ylabel('F(x)');

**OUTPUT-**





**EXPERIMENT 6**

**AIM:** Generating a sinusoidal signal of a given frequency (say 100Hz) and plotting with graphical enhancements- Titling, Labelling, adding text, adding legends, adding new plots to existing plot, Plotting as multiple and subplot.

**APPARATUS:** Octave online, MatLab

**THEORY:**

* **Title**: Add a title to the chart by using the **title** function.
* **Labelling:** Each axes graphics object can have one **label** for the x-, y-, and z-axis. The **label** appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot. For example: **xlabel**(' string ') labels the x-axis of the current axes.
* **Adding Text:**Text command is used to add text to a graph at a certain point.
* **Adding Legend:** The **legend** of a graph reflects the data displayed in the graph's Y-axis, also called the graph series. This is the data that comes from the columns of the corresponding grid report, and usually represents metrics. A graph legend generally appears as a box to the right or left of your graph.

**PROGRAM:**

clc;

clear all;

close all;

f=100;

T=1/f;

t= (0:200) \*T;

y=2\*pi\*f\*t;

plot(y,sin(y),'r\*-','markersize',1);

grid on;

title ('plot of sine waveform')

t1=linspace(0,2\*pi,100);

x1=0:100:2\*pi;

y1=sin(t1);

y2=t1;

y3=t1-t1. ^3/6+t1. ^5/120;

plot (t1, y1,'g+-','markersize',3, t1, y2,'c--','markersize',4, t1, y3,'b.','markersize',3)

text (4.0,0,'sin(t1)');

gtext('linear approx');

gtext('first three');

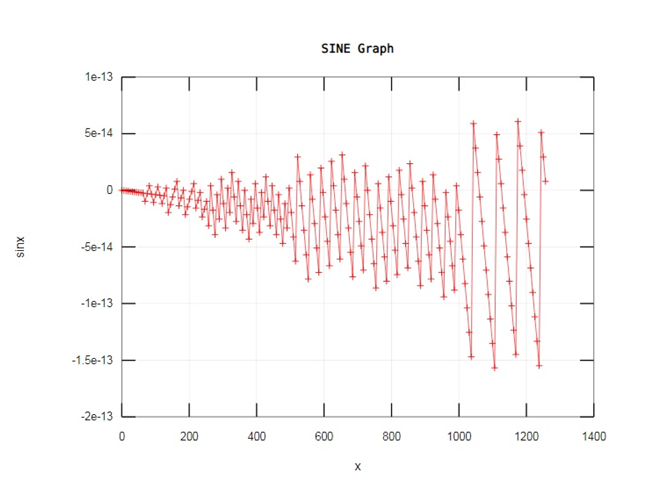
grid on;

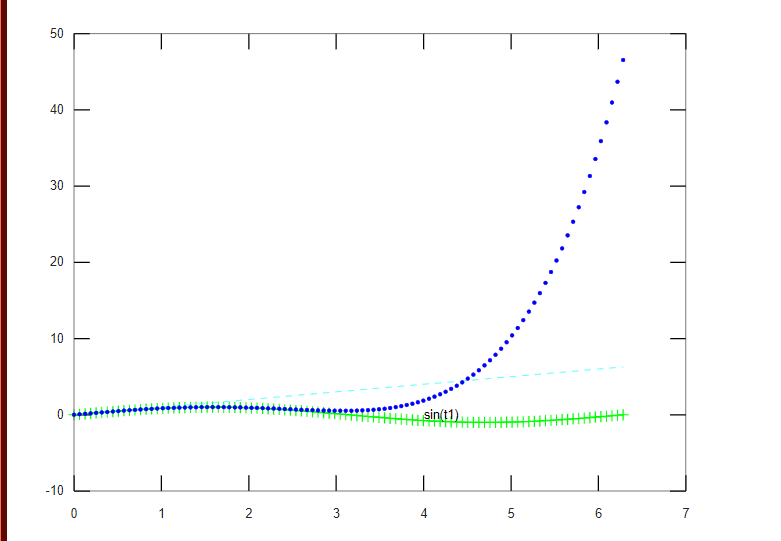
title ('plot of sine')

xlabel('----->n')

ylabel('----->x')

**OUTPUT:**

****



**EXPERIMENT 7**

AIM:

Solving First Order Ordinary Differential Equation using Built-in Functions.

APPARATUS: Octave online, MatLab

THEORY:

* Symbolic functions represent math functions. Symbolic functions are used for differentiation, integration, solving ODEs, and other math operations.
* **syms:**Creates symbolic variables and functions.

1. Creates symbolic functions with one and two arguments (syms  f(x,y)).
2. Creates a symbolic function and can specify its formula by using a symbolic matrix (syms f(x))

* Using ode23 and ode45,

**[[t,y] = ode23(odefun,tspan,y0,options)](https://www.mathworks.com/help/matlab/ref/ode45.html?s_tid=srchtitle" \t "_blank)**

**[[t,y] = ode45(odefun,tspan,y0,options)](https://www.mathworks.com/help/matlab/ref/ode45.html?s_tid=srchtitle" \t "_blank)**

PROGRAM-

clc;

clear all;

close all;

syms f(x,y)

f(x,y) = x + 5\*y

f(1,5)

xVal = 1:5;

yVal = 3:7;

f(xVal,yVal)

dfx = diff(f,x)

syms f(x)

M = [x x^3; x^2 x^4];

f(x) = M

f(3)

syms y(t) a

eqn = diff(y,t) == a\*y

cond = y(0) == 5

ySol(t) = dsolve(eqn,cond)

tspan = [0 5];

y0 = 0;

[t,y] = ode45(@(t,y) 2\*t, tspan, y0);

plot(t,y,'b-+','markersize',2)

OUTPUT-

