NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL
INTRODUCTION TO MATLAB/SCILAB/WEB DESIGN LAB (IT-406)

INFORMATION TECHNOLOGY (IT)

EXPERIMENT NO-1

OBJECTIVE: To write a MATLAB program to perform some basic operation on matrices such as addition, subtraction, multiplication.

SOFTWARE REQURIED:-

1.MATLAB R2010a.
2.Windows XP SP2.

THEORY:-

MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Built in Functions:

1. Scalar Functions:

Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector). They are summarized below.

1. sin - trigonometric sine
2. cos - trigonometric cosine

3. tan - trigonometric tangent
4. asin - trigonometric inverse sine (arcsine)
5. acos - trigonometric inverse cosine (arccosine)
6. atan - trigonometric inverse tangent (arctangent)
7. exp - exponential
8. log - natural logarithm
9. abs - absolute value
10. sqrt - square root
11. rem - remainder
12. round - round towards nearest integer
13. floor - round towards negative infinity
14. ceil - round towards positive infinity

2. Vector Functions:
Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these functions are given below.
1. max largest component : get the row in which the maximum element lies
2. min smallest component
3. lengthlength of a vector
4. sortsort in ascending order
5. sumsum of elements
6. prod product of elements
7. medianmedian value
8. meanmean value std standard deviation

3. Matrix Functions:
Much of MATLAB‟s power comes from its matrix functions. These can be further separated into two sub-categories.
The first one consists of convenient matrix building functions, some of which are given below.
1. eye - identity matrix
2. zeros - matrix of zeros
3. ones - matrix of ones
4. diag - extract diagonal of a matrix or create diagonal matrices
5. triu - upper triangular part of a matrix
6. tril - lower triangular part of a matrix
7. rand - randomly generated matrix

commands in the second sub-category of matrix functions are
1. sizesize of a matrix
2. det determinant of a square matrix
3. inv inverse of a matrix
4. rankrank of a matrix
5. rref reduced row echelon form
6. eig eigenvalues and eigenvectors
7. poly characteristic polynomial

PROCEDURE:-
- Open MATLAB
- Open new M-file

Type the program
Save in current directory
Compile and Run the program
For the output see command window\ Figure window

PROGRAM:-

```
clc;
close all;
clear all;
a=[1 2 -9 ; 2 -1 2; 3 -4 3];
b=[1 2 3; 4 5 6; 7 8 9];
disp('The matrix a= ');
a
disp('The matrix b= ');
b
% to find sum of a and b
c=a+b;
disp('The sum of a and b is ');
c
% to find difference of a and b
d=a-b;
disp('The difference of a and b is ');
d
%to find multiplication of a and b
e=a*b;
disp('The product of a and b is ');
e
```

OUTPUT:-

```
The matrix a=
a =
1 2 -9
2 -1 2
3 -4 3
The matrix b=
b =
1 2 3
4 5 6
7 8 9
The sum of a and b is
c =
2 4 -6
6 4 8
10 4 12
The difference of a and b is
d =
0 0 -12
-2 -6 -4
-4 -12 -6
The product of a and b is
```

e =
-54 -60 -66
12 15 18
8 10 12

RESULT:-
Finding addition, subtraction, multiplication using MATLAB was Successfully completed.

POSSIBLE VIVA QUESTIONS:-
1.Expand MATLAB?And importance of MATLAB?
2.What is clear all and close all will do?
3.What is disp() and input()?
4.What is the syntax to find the eigen values and eigenvectors of the matrix?
5. What is the syntax to find the rank of the matrix?

EXERCISE:
1. Enter the matrix
M = [1,-2,8,0] and N = [1 5 6 8; 2 5 6 9]
Perform addition on M and N and see how matlab reacts.
2. Find the transpose of null matrix using matlab
3. write a MATLAB program to perform the division operation on the following matrix
A = [24,-30, 64,-81], b= [6,5,8,9] and verify the result.
4. Write a matlab program to perfom addition operation using 2x3 matrix. Assume any numbers
5. Enter the matrix
A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7], B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9],
C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4]
Find [(A+B)+C]T
6. Enter the matrix
A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7], B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9],
C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4]
Find $[(A-B)+C]^{-1}$
7. Write a matlab program to perfom addition operation using 3x2 matrix. Assume any numbers
8 write a MATLAB program to perform the division operation on the following matrix A = [25,-35, 121,-21], b= [5,5,11,3] and perform the transpose function on the answer
9.Find the addition of null matrix and unity matrix of order 3x3.
10. Enter the Matrix the following Matrices and multiply M and N using M*N. Observe the output in the command window.

$$M = \begin{matrix} -1 & 2 & 4 \\ 2 & -1 & -1 \\ 4 & 2 & 0 \end{matrix} \quad N = \begin{matrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{matrix}$$

Experiment No-2

OBJECTIVE: To write a "MATLAB" Program to generate various signals and sequences, such as unit impulse, unit step, unit ramp, sinusoidal, square, sawtooth, triangular, sinc signals.

SOFTWARE REQURIED:-

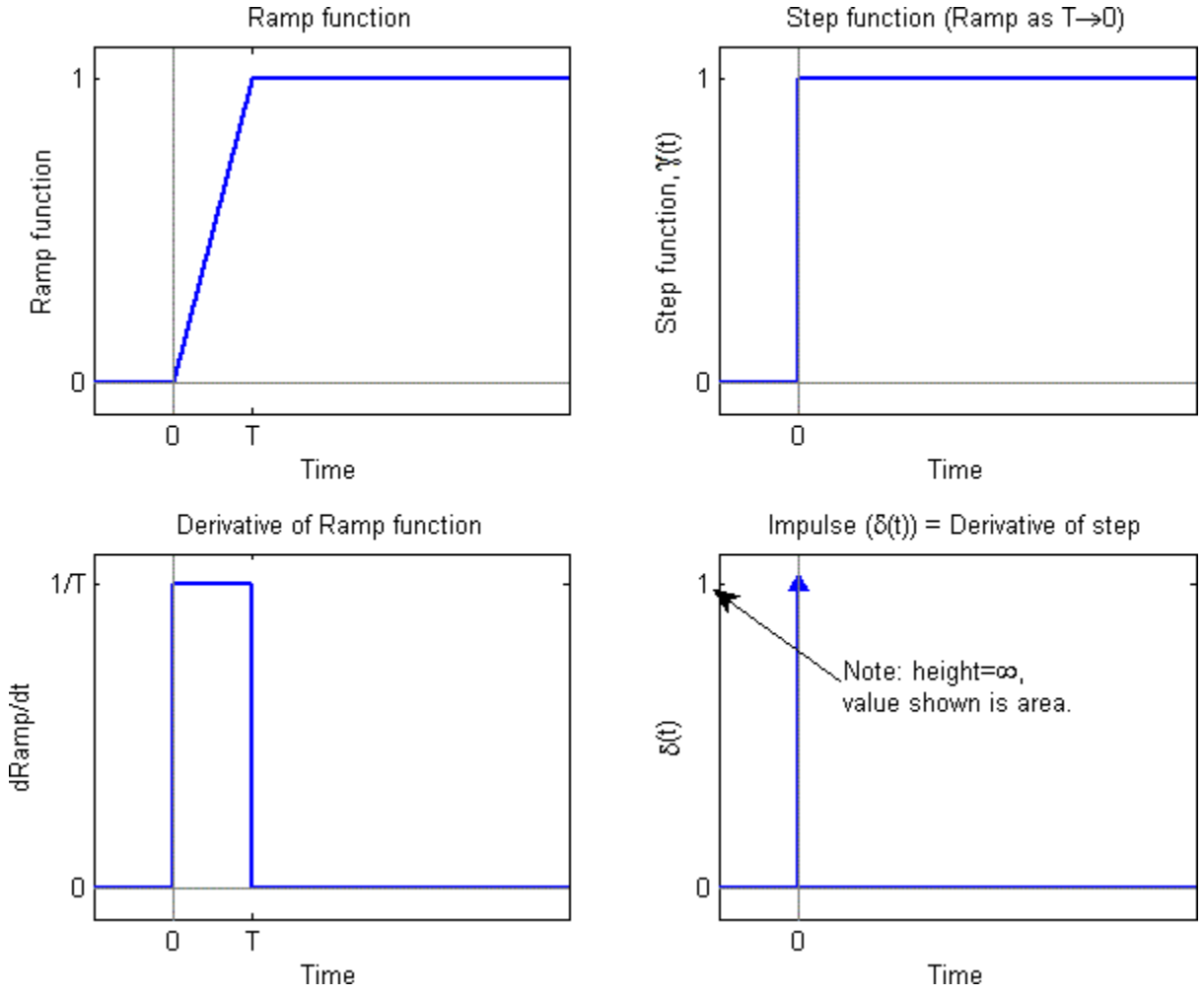
1.MATLAB R2010a.

2. Windows XP SP2.

THEORY:-

UNIT IMPULSE FUNCTION:

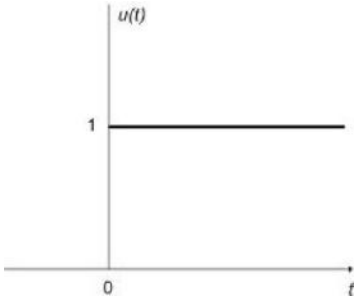
One of the more useful functions in the study of linear systems is the "unit impulse function."An ideal impulse function is a function that is zero everywhere but at the origin, where it is infinitely high. However, the *area* of the impulse is finite. This is, at first hard to visualize butwe can do so by using the graphs shown below.



UNIT STEP FUNCTION

The unit step function and the impulse function are considered to be fundamental functions in engineering, and it is strongly recommended that the reader becomes very familiar with both of these functions.The unit step function, also known as the Heaviside function, is defined as such:

$$u(t) = \begin{cases} 0, & \text{if } t < 0 \\ 1, & \text{if } t > 0 \\ \frac{1}{2}, & \text{if } t = 0 \end{cases}$$



Sinc Function

There is a particular form that appears so frequently in communications engineering, that wegive it its own name. This function is called the "Sinc function and discussed below

The Sinc function is defined in the following manner:

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \text{ if } x \neq 0$$

And Sinc(0)=1

The value of sinc(x) is defined as 1 at x = 0, since

$$\lim_{x \to 0} \operatorname{sinc}(x) = 1$$

***Rect* Function**

The Rect Function is a function which produces a rectangularcentered at t = 0. The Rect function pulse also has a height of 1. The Sinc function and therectangular function form a Fourier transform pair.A *Rect* function can be written in the form:

$$\operatorname{rect}\left(\frac{t - X}{Y}\right)$$

where the pulse is centered at X and has widthY. We can define the impulse function above interms of the rectangle function by centering the pulse at zero (X = 0), setting it's height to 1/Aand setting the pulse width to A, which approaches zero:

$$\delta(t) = \lim_{A \to 0} \frac{1}{A} \operatorname{rect}\left(\frac{t - 0}{A}\right)$$

We can also construct a *Rect* function out of a pair of unit step functions

$$\operatorname{rect}\left(\frac{t - X}{Y}\right) = u(t - X + Y/2) - u(t - X - Y/2)$$

Here,both unit step functions are set a distance of Y/2 away from the center point of (t - X).

SAW TOOTH:-

The sawtooth wave (or saw wave) is a kind of non-sinusoidal waveform. It is named a sawtooth based on its resemblance to the teeth on the blade of a saw. The convention is that a sawtooth wave ramps upward and then sharply drops. However, there are also sawtooth waves in which the wave ramps downward and then sharply rises. The latter type of sawtooth wave is called a 'reverse sawtooth wave' or 'inverse sawtooth wave'. As audio signals, the two orientations of sawtooth wave sound identical. The piecewise linear function based on the floor function of time t, is an example of a sawtooth wave with period 1.

$$x(t) = 2\left(\frac{t}{a} - \operatorname{floor}\left(\frac{t}{a} + \frac{1}{2}\right)\right)$$

TRIANGLE WAVE

A triangle wave is a non-sinusoidal waveform named for its triangular shape.Abandlimited triangle wave pictured in the time domain (top) and frequency domain (bottom). The fundamental is at 220 Hz (A2).Like a square wave, the triangle wave contains only odd harmonics. However, the higher harmonics roll off much faster than in a square wave (proportional to the inverse square of the harmonic number as opposed to just the inverse).It is possible to approximate a triangle wave with additive synthesis by adding odd harmonics of the fundamental, multiplying every (4n..1)th harmonic by 1 (or changing its phase by), and rolling off the harmonics by the inverse square of their relative frequency to the fundamental. This infinite Fourier series converges to the triangle wave:

$$\begin{aligned} x_{\mathbf{triangle}}(t) &= \frac{8}{\pi^2}\sum_{k=0}^{\infty}(-1)^k\frac{\sin\left((2k+1)\omega t\right)}{(2k+1)^2} \\ &= \frac{8}{\pi^2}\left(\sin(\omega t) - \frac{1}{9}\sin(3\omega t) + \frac{1}{25}\sin(5\omega t) - \cdots\right) \end{aligned}$$

where ω is the angular frequency.

Sinusoidal Signal Generation

The sine wave or sinusoid is a mathematical function that describes a smooth repetitiveoscillation. It occurs often in pure mathematics, as well as physics, signal processing, electrical engineering and many other fields. Its most basic form as a function of time (t)is:where:

- A, the amplitude, is the peak deviation of the function from its center position.
- the angular frequency, specifies how many oscillations occur in a unit time interval, in radians per second
- the phase, specifies where in its cycle the oscillation begins at t = 0.

A sampled sinusoid may be written as:

$$x\ n\ = A\sin(2\pi\frac{f}{f_s}n + \theta)$$

where f is the signal frequency, fs is the sampling frequency, θ is the phase and A is theamplitude of the signal.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory

- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:-

%unit impulse function%

```
clc;
clearall;
closeall;
t=-10:1:10;
x=(t==0);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit impulse function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit impulse discreat function');
```

%unit step function%

```
clc;
clearall;
closeall;
N=100;
t=1:100;
x=ones(1,N);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit step function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit step discreat function');
```

%unit ramp function%

```
clc;
clearall;
closeall;
t=0:20;
x=t;
```

```
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit ramp function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit ramp discreat function');
```

%sinusoidal function%

```
clc;
clearall;
closeall;
t=0:0.01:2;
x=sin(2*pi*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sinusoidal signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sinusoidal sequence');
```

%square function%

```
clc;
clearall;
closeall;
t=0:0.01:2;
x=square(2*pi*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('square signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('square sequence');
```

%sawtooth function%

```
clc;
clearall;
```

```
closeall;
t=0:0.01:2;
x=sawtooth(2*pi*5*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sawtooth signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sawtooth sequence');
```

%trianguler function%

```
clc;
clearall;
closeall;
t=0:0.01:2;
x=sawtooth(2*pi*5*t,0.5);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('trianguler signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('trianguler sequence');
```

%sinc function%

```
clc;
clearall;
closeall;
t=linspace(-5,5);
x=sinc(t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sinc signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sinc sequence');
```

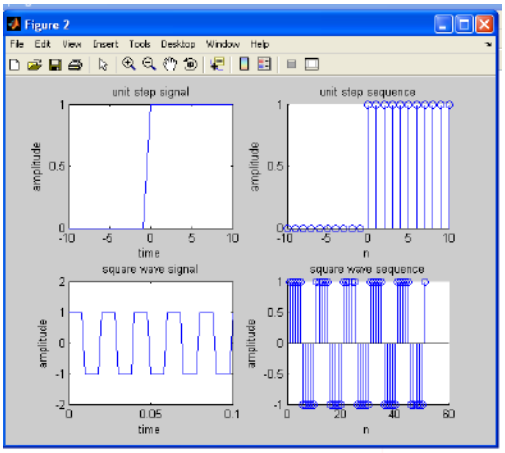


Figure 2
File Edit View Insert Tools Desktop Window Help
unit step signal
unit step sequence
square wave signal
square wave sequence
amplitude
time
n

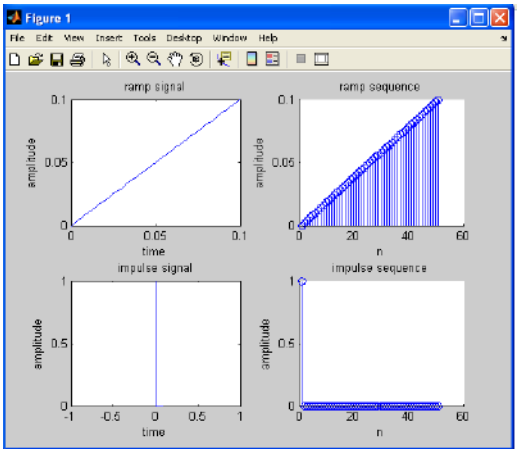


Figure 1
File Edit View Insert Tools Desktop Window Help
ramp signal
ramp sequence
impulse signal
impulse sequence
amplitude
time
n

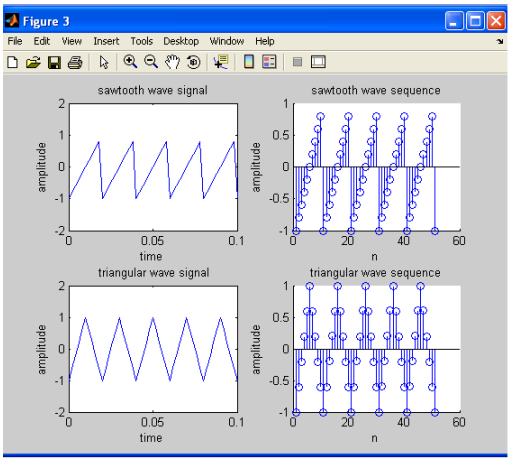


Figure 3
File Edit View Insert Tools Desktop Window Help
sawtooth wave signal
sawtooth wave sequence
triangular wave signal
triangular wave sequence
amplitude
time
n

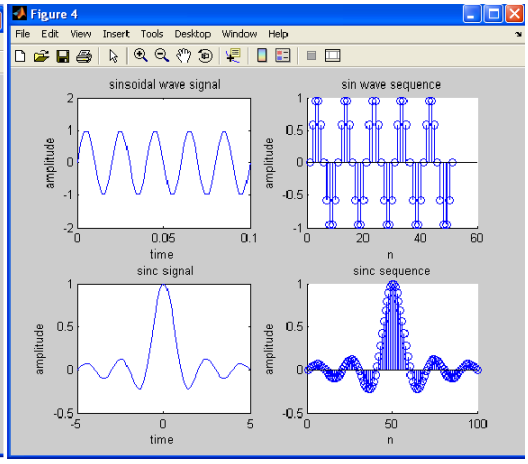
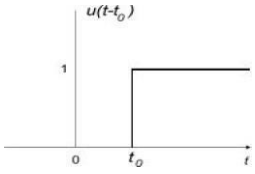
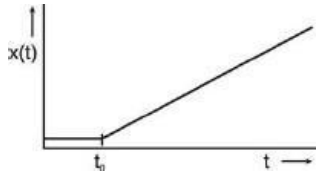


Figure 4
File Edit View Insert Tools Desktop Window Help
sinsoidal wave signal
sin wave sequence
sinc signal
sinc sequence
amplitude
time
n


u(t-t0)
1
0
t0
t


x(t)
t0
t

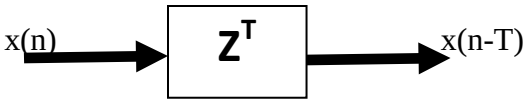
EXPERIMENT No-3

OBJECTIVE: To performs operations on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and average power.

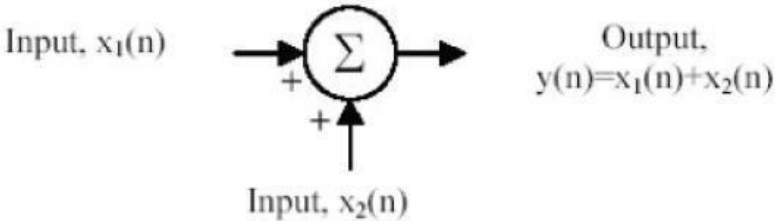
SOFTWARE REQURIED:-
1.MATLAB R2010a.
2.Windows XP SP2.

THEORY:-
Basic Operation on Signals:
Time shifting: y(t)=x(t-T)The effect that a time shift has on the appearance of a signal If T is a positive number, the time shifted signal, x(t -T) gets shifted to the right, otherwise it gets shifted left.

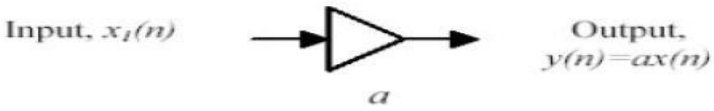
Signal Shifting and Delay:
Shifting : y(n)={x(n-k)} ; m=n-k; y=x;
Time reversal: Y(t)=y(-t) Time reversal flips the signal about t = 0



Signal Addition and Subtraction:
Addition: any two signals can be added to form a third signal,z (t) = x (t) + y (t)

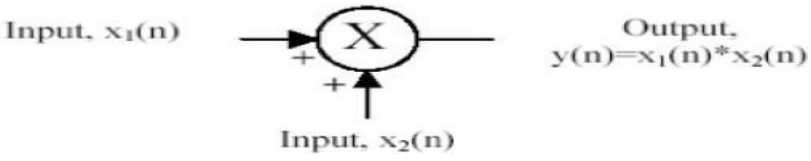


Signal Amplification/Attuation :



Multiplication/Divition :

of two signals, their product is also a signal.
z (t) = x (t) y (t)



folding:

y(n)={x(-n)} ; y=fliplr(x); n=-fliplr(n);

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:-

```
clear   all;
close   all;
t=0:.01:1;
% generating two input signals
x1=sin(2*pi*4*t);
x2=sin(2*pi*8*t);
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('signal1:sine wave of frequency 4Hz');
subplot(2,2,2);
plot(t,x2);
xlabel('time');
subplot(4,1,3);
ylabel('amplitude');
title('signal2:sine wave of frequency 8Hz');
% addition of signals
y1=x1+x2;
subplot(2,2,3);
plot(t,y1);
xlabel('time');
ylabel('amplitude');
title('resultant signal:signal1+signal2');
% multiplication of signals
y2=x1.*x2;
subplot(2,2,4);
plot(t,y2);
xlabel('time');
ylabel('amplitude');
title('resultant signal:dot product of signal1 and signal2');
% scaling of a signal1
A=10;
y3=A*x1;
figure;
```

```
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
subplot(2,2,2);
plot(t,y3);
xlabel('time');
ylabel('amplitude');
title('amplified input signal1 ');
% folding of a signal1
l1=length(x1);
nx=0:l1-1;
subplot(2,2,3);
plot(nx,x1);
xlabel('nx');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
y4=fliplr(x1);
nf=-fliplr(nx);
subplot(2,2,4);
plot(nf,y4);
xlabel('nf');
ylabel('amplitude');
title('folded signal');
%shifting of a signal
figure;
t1=0:.01:pi;
x3=8*sin(2*pi*2*t1);
subplot(3,1,1);
plot(t1,x3);
xlabel('time t1');
ylabel('amplitude');
title('sine wave of frequency 2Hz');
subplot(3,1,2);
plot(t1+10,x3);
xlabel('t1+10');
ylabel('amplitude');
title('right shifted signal');
subplot(3,1,3);
plot(t1-10,x3);
xlabel('t1-10');
ylabel('amplitude');
title('left shifted signal');
%operations on sequences
n1=1:1:9;
s1=[1 2 3 0 5 8 0 2 4];
figure;
subplot(2,2,1);
stem(n1,s1);
```

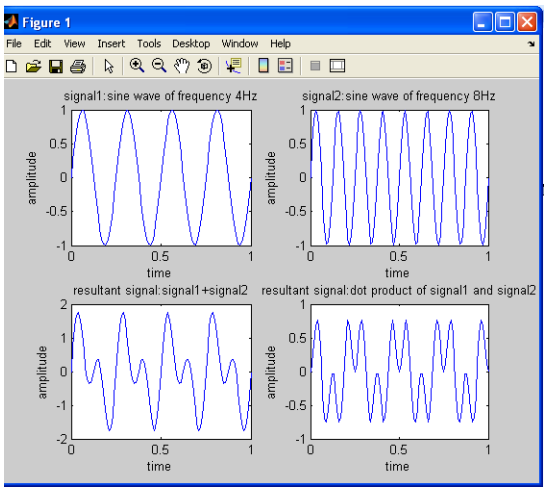
```
xlabel('n1');
ylabel('amplitude');
title('input sequence1');

n2=-2:1:6;
s2=[1 1 2 4 6 0 5 3 6];
subplot(2,2,2);
stem(n2,s2);
xlabel('n2');
ylabel('amplitude');
title('input sequence2');
% addition of sequences
s3=s1+s2;
subplot(2,2,3);
stem(n1,s3);
xlabel('n1');
ylabel('amplitude');
title('sum of two sequences');
% multiplication of sequences
s4=s1.*s2;
subplot(2,2,4);
stem(n1,s4);
xlabel('n1');
ylabel('amplitude');
title('product of two sequences');
% scaling of a sequence
figure;
subplot(2,2,1);
stem(n1,s1);
xlabel('n1');
ylabel('amplitude');
title('input sequence1');
s5=4*s1;
subplot(2,2,2);
stem(n1,s5);
xlabel('n1');
ylabel('amplitude');
title('scaled sequence1');

subplot(2,2,3);
stem(n1-2,s1);
xlabel('n1');
ylabel('amplitude');
title('left shifted sequence1');
subplot(2,2,4);
stem(n1+2,s1);
xlabel('n1');
ylabel('amplitude');
title('right shifted sequence1');
% folding of a sequence
```

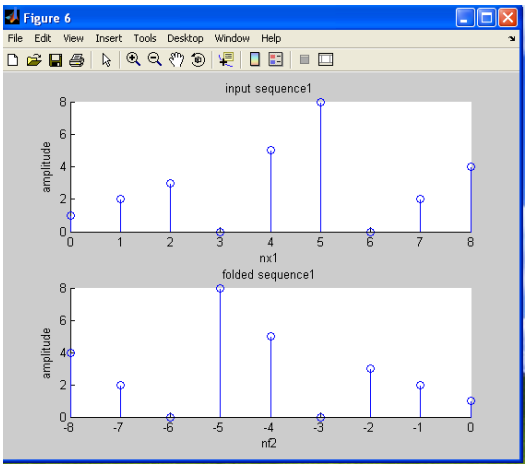
```
l2=length(s1);
nx1=0:l2-1;
figure;
subplot(2,1,1);
stem(nx1,s1);
xlabel('nx1');
ylabel('amplitude');
title('input sequence1');
s6=fliplr(s1);
nf2=-fliplr(nx1);
subplot(2,1,2);
stem(nf2,s6);
xlabel('nf2');
ylabel('amplitude');
title('folded sequence1');
% program for energy of a sequence

e1=sum(abs(z1).^2);
e1
% program for energy of a signal
t=0:pi:10*pi;
z2=cos(2*pi*50*t).^2;
e2=sum(abs(z2).^2);
e2
% program for power of a saequence
p1= (sum(abs(z1).^2))/length(z1);
p1
% program for power of a signal
p2=(sum(abs(z2).^2))/length(z2);
p2
```

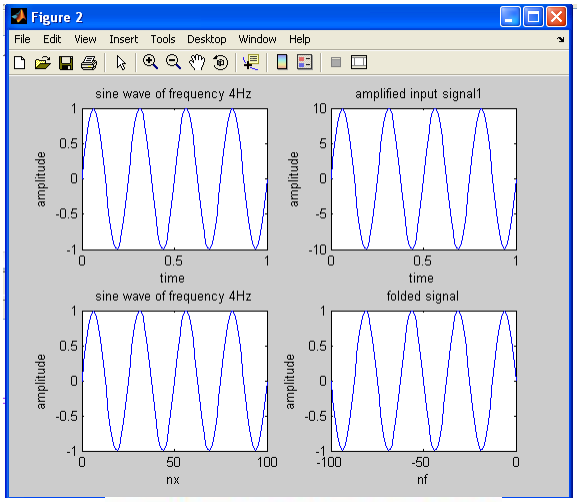
OUTPUT:

```
enter the input sequence [1 3 5 6]
e1 =
71
e2 =
4.0388
p1 =
17.7500
p2 =
0.3672
```

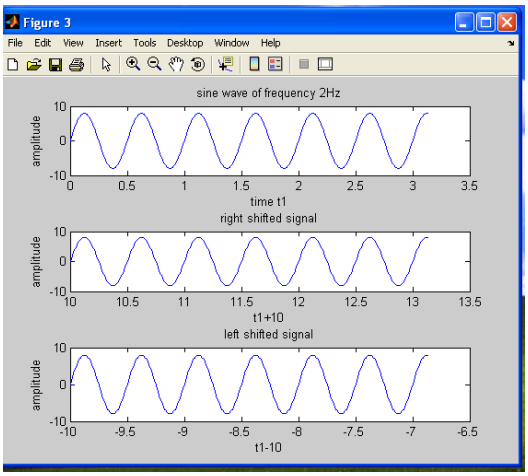
Result: Various operations on signals and sequences are performed.



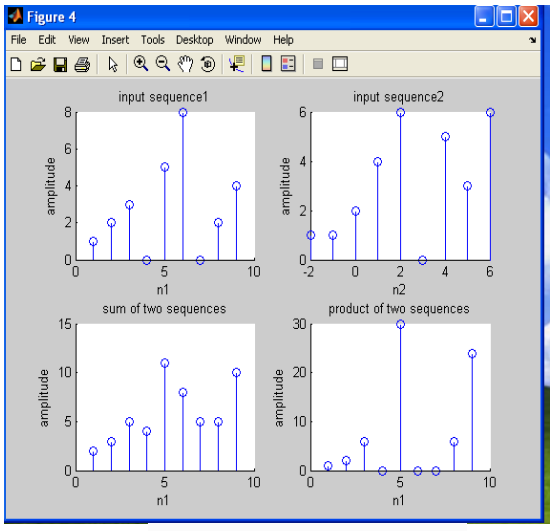




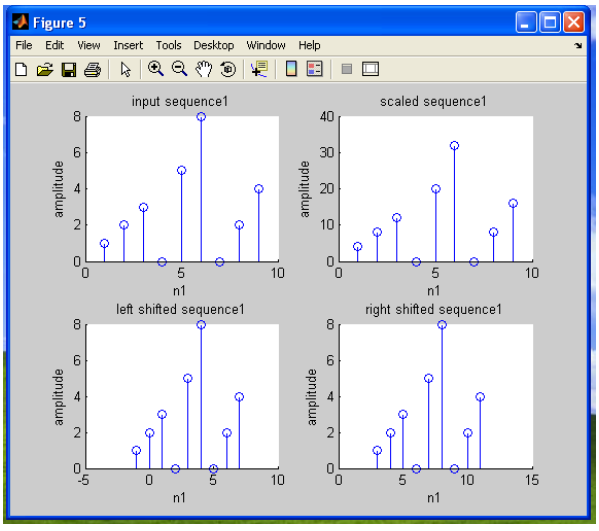
Amplitude scaling for signals



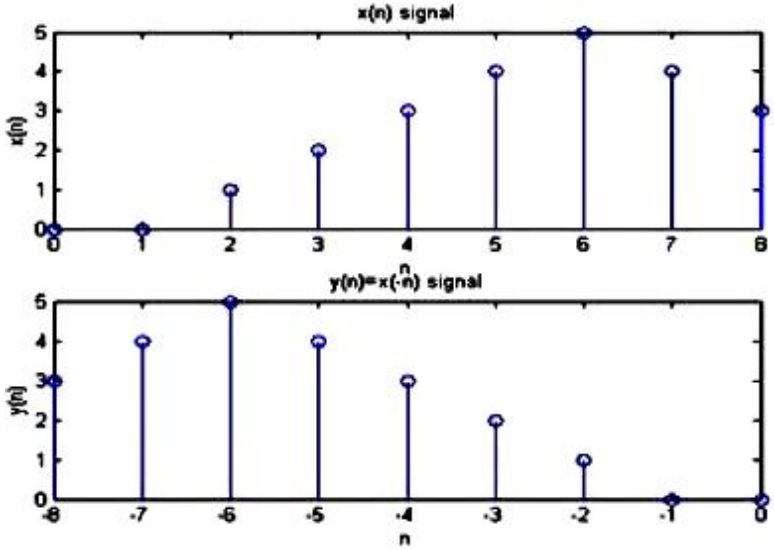
Time scaling for signals



Time shifting of a signal



Time folding of a signal


x(n) signal
x(n)
n
y(n)=x(-n) signal
y(n)
n

EXPERIMENTY No-4

OBJECTIVE: Finding even and odd part of the signal and sequence and also findreal and imaginary parts of signal.

Software Required:
Matlab software 7.0 and above.

Theory:
EVEN AND ODD PART OF A SIGNAL:
Any signal x(t) can be expressed as sum of even and odd components I e
$X(t)=x_e(t)+x_o(t)$

$$x_e(t) = \frac{1}{2}\{x(t) + x(-t)\}, \quad ; \quad x_o(t) = \frac{1}{2}\{x(t) - x(-t)\}$$

$$x(t) = x_e(t) + x_o(t)$$

$$= \frac{1}{2}\{x(t) + x(-t)\} + \frac{1}{2}\{x(t) - x(-t)\}$$

Program:

```
Clc;
close all;
clear all;
% Even and odd parts of a signal
t=0:.005:4*pi;
x=sin(t)+cos(t); % x(t)=sint(t)+cos(t)
subplot(2,2,1)
plot(t,x)
xlabel('t');
ylabel('amplitude')
title('input signal')
y=sin(-t)+cos(-t) % y=x(-t)
subplot(2,2,2)
plot(t,y)
xlabel('t');
ylabel('amplitude')
title('input signal with t=-t')
z=x+y
subplot(2,2,3)
plot(t,z/2)
xlabel('t');
ylabel('amplitude')
title('even part of the signal')%assigning a name to the plot
p=x-y
subplot(2,2,4)
plot(t,p/2)
xlabel('t');
```

```
ylabel('amplitude');
title('odd part of the signal');
% Even and odd parts of a sequence
z=[0,2+j*4,-3+j*2,5-j*1,-2-j*4,-j*3,0];
n=-3:3
% plotting real and imginary parts of the sequence
figure;
subplot( 2,1,1);
stem(n,real(z));
xlabel('n');
ylabel('amplitude');
title('real part of the complex sequence');
subplot( 2,1,2);
stem(n,imag(z));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the complex sequence');
zc=conj(z);
zc_folded= fliplr(zc);
zc_even=.5*(z+zc_folded);
zc_odd=.5*(z-zc_folded);
% plotting even and odd parts of the sequence
figure;
subplot( 2,2,1);
stem(n,real(zc_even));
xlabel('n');
ylabel('amplitude');
title('real part of the even sequence');
subplot( 2,2,2);
stem(n,imag(zc_even));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the even sequence');
subplot( 2,2,3);
stem(n,real(zc_odd));
xlabel('n');
ylabel('amplitude');
title('real part of the odd sequence');
subplot( 2,2,4);
stem(n,imag(zc_odd));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the odd sequence');
```

RESULT: Even and odd part of the signal and sequence is computed.

OUTPUT:

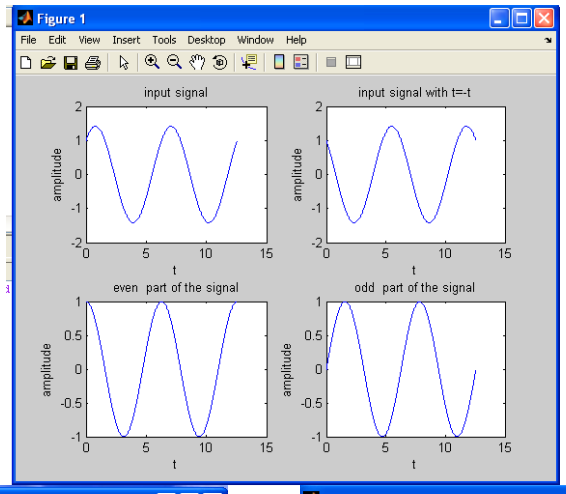


Figure 1
File Edit View Insert Tools Desktop Window Help
input signal
input signal with t=-t
even part of the signal
odd part of the signal
amplitude
t

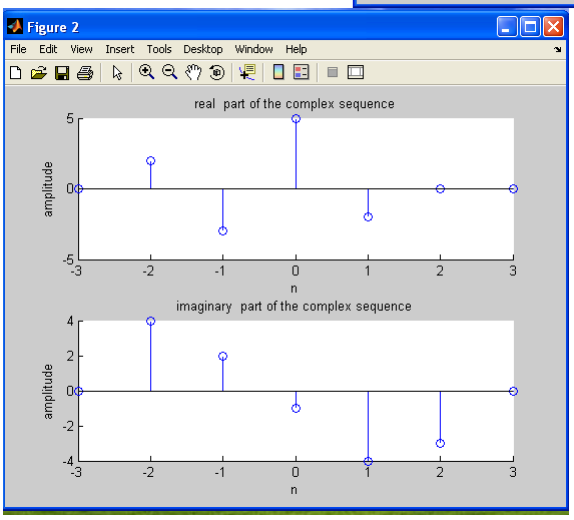


Figure 2
File Edit View Insert Tools Desktop Window Help
real part of the complex sequence
imaginary part of the complex sequence
amplitude
n

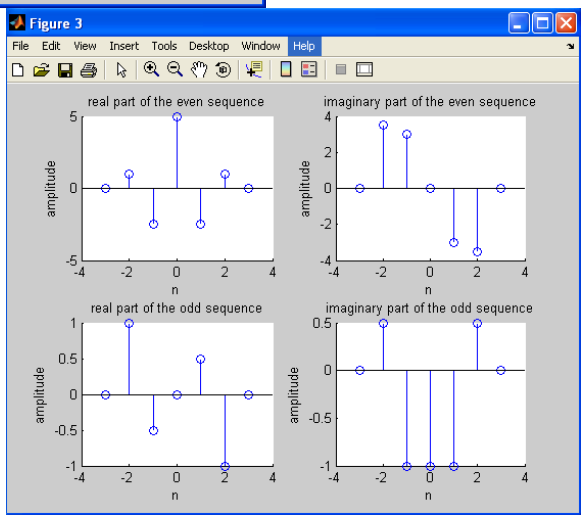


Figure 3
File Edit View Insert Tools Desktop Window Help
real part of the even sequence
imaginary part of the even sequence
real part of the odd sequence
imaginary part of the odd sequence
amplitude
n

EXPERIMENTY No-5

OBJECTIVE: To find the output with linear convolution operation Using MATLAB Software.

SOFTWARE REQURIED:-

1.MATLAB7.2(2006b) / MATLAB 8.6(2015b)/MATLAB 7.6 2008a(Trial version)/MATLAB 7.9(2009b)(Trial Version)/MATLAB 7.10(2010a) Trial version.

2.Windows XP SP2.

THEORY:-

Linear Convolution involves the following operations.

1. Folding
2. Multiplication
3. Addition
4. Shifting

These operations can be represented by a Mathematical Expression as follows:

$$y[n] = \sum_{k=-\infty} x[k]h[n-k]$$

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

%**program for convolution of two sequences**

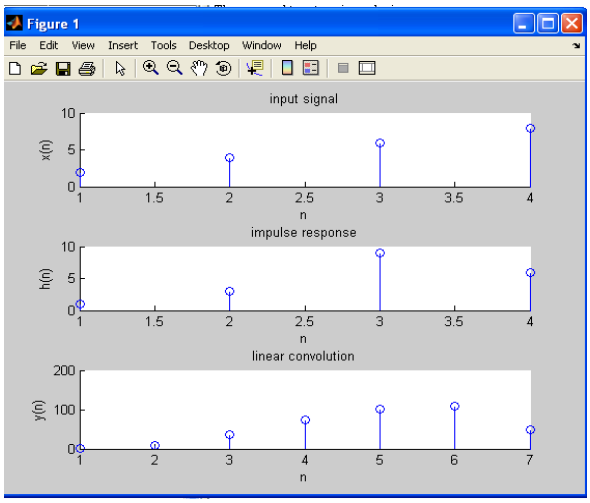
```
clc;
close all;
clear all;
%program for convolution of two sequences
x=input('enter input sequence');
h=input('enter impulse response');
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input signal')
subplot(3,1,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse response')
subplot(3,1,3);
stem(y);
```

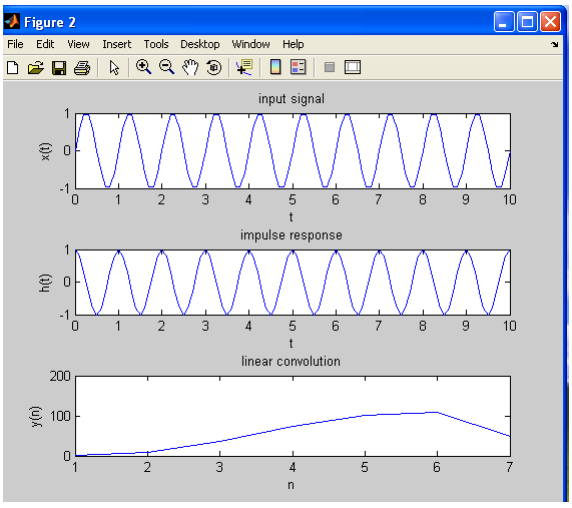
```
xlabel('n');
ylabel('y(n)');
title('linear convolution')
disp('The resultant signal is');
disp(y)
```

%program for signal convolution

```
t=0:0.1:10;
x1=sin(2*pi*t);
h1=cos(2*pi*t);
y1=conv(x1,h1);
figure;
subplot(3,1,1);
plot(t,x1);
xlabel('t');
ylabel('x(t)');
title('input signal')
subplot(3,1,2);
plot(t,h1);
xlabel('t');
ylabel('h(t)');
title('impulse response')
subplot(3,1,3);
plot(y1);
xlabel('n');
ylabel('y(n)');
title('linear convolution');
```

OUTPUT:-





RESULT: convolution between signals and sequences is computed.

Output:

```
enter input sequence[2 4 6 8]
enter impulse response[1 3 9 6]
The resultant signal is
2 10 36 74 102 108 48
```

Experiment No-6

i. OBJECTIVE: Practicing MATLAB environment with simple exercises to familiarize Command Window, History, Workspace, Current Directory, Figure window, Edit window, Shortcuts, Help files.

ii. SOURCE CODE :

Working on general commands on scilab environment

Help – detailed help menu for scilab commands
Who – list all the variables from the variable browser window
Whos - list all the variables with byte size, variable type etc.
Clc – Clears command window
Clear – Removes the variable from memory
Quit – to close the session
Pwd – present working directory
Ls – list the files
Ls -ltr – list the detailed view on the files
Cd - to change the directory
Mkdir – to create a new directory

To work on Special variables / Pre-Defined Variables
%pi – 3.14
Ans
% e = 2.718
%eps – epsilon
%inf – infinity

Basic Scalar & Vector Operations

Creation of Scalar Elements
Y = [1 4 6] ➔ Declares a row vetor
yT = [1; 4; 6] ➔ Declares a Column vector
Creation of Vector Elements
Y = [1 4 6; 2 7 3; 4 1 1]
➔ Creates a 3*3 matrix

To determine the size / order of the vectors.
Size(y)

To change the elements in the given vector
vector(i,j) = value

Performing element by element operations using dot operator

.*, ./, .^,

Linspace[a,b,N] ➔ Vector can be created by evenly spaced points. From a to b the vector is created by 'N' evenly spaced points

Eg: linspace[0,1,5] → 0 0.25 0.5 0.75 1

Transpose of a matrix – y'

ans =

1. 2. 4.
4. 7. 1.
6. 3. 1.

RESULT

Study of basic SCILab commands are worked and executed

Experiment No-7

i. **OBJECTIVE:** **To study on basic Matrix Constructors and Operations.**

ii. **SOURCE CODE :**

Zeros(m,n) – creates m rows with n cols

```
zeros(3,2)
 ans  =
    0.   0.
    0.   0.
    0.   0.
```

Eye(m,n) – creates identity matrix

```
eye(2,3)
 ans  =
   1.   0.   0.
   0.   1.   0.
```

Ones(m,n) – creates matrix with all 1's for all m rows and n cols

```
ones(2,2)
 ans  =
1.   1.
1.   1.
```

rand(m,n) – creates matrix with random numbers

```
rand(4,4)
 ans  =
   0.2113249   0.6653811   0.8782165   0.7263507
   0.7560439   0.6283918   0.0683740   0.1985144
   0.0002211   0.8497452   0.5608486   0.5442573
   0.3303271   0.6857310   0.6623569   0.2320748
```

Max(z) -- returns the largest element in a vector.

```
Y  =
   1.   4.   6.
```

```
    2.    7.    3.
    4.    1.    1.
max(Y)
 ans =
    7.
```

Min(z) - returns the smallest element in a vector.

```
min(Y)
 ans =
    1.
```

Sum(z) – returns the sum of elements in a vector

```
sum(Y)
 ans =
    29.
```

prod(z) – returns the product of all elements in a vector.

```
prod(Y)
 ans =
   4032.
```

Mathematical operations

Sin(z) – Retrieve the sine value for the given matrix / vector

```
sin(Y)
 ans =

    0.8414710 - 0.7568025   - 0.2794155
    0.9092974    0.6569866    0.1411200
  - 0.7568025    0.8414710    0.8414710
```

Similar operation can be obtained for Cos, tan, sec, csc, cot. The hyperbolic for sine, cosine etc can be retrieved using sinh, cosh etc.

Inverse of cosine, sine can be obtained using the acos, asin etc

Exp(z) – Returns the exponential, element wise

```
exp(Y)
 ans  =

    2.7182818    54.59815   403.42879
    7.3890561   1096.6332   20.085537
    54.59815    2.7182818   2.7182818
```

Log10(z) and log(z) provides the base 10 & natural logarithm for the given vector and matix

```
log(Y)
 ans  =
         0.        1.3862944   1.7917595
    0.6931472   1.9459101   1.0986123
    1.3862944         0.             0.
```

Sqrt(z) provides the square root for the matrix elements.

```
sqrt(Y)
 ans  =
         1.             2.          2.4494897
    1.4142136   2.6457513      1.7320508
         2.             1.              1.
```

Performing addition, subtraction, multiplication and division for array vectors or matrix elements

Floating Point operations

Working on floating point operations like ceil, floor, fix, round for both vectors and matrices

nthroot(x,n) -- Is the real vector/matrix of the nth root of the x elements

```
nthroot(Y,4)
 ans  =
    1.          1.4142136   1.5650846
```

1.1892071 1.6265766 1.316074

1.4142136 1. 1.

Sign(z) -- returns the matrix made of the signs of z(i,j)

sign(Y)

ans = 1. 1. 1.

1. 1. 1.

1. 1. 1.

Modulo(n,m) – computes n\m and gives the remained.

pmodulo(n,m) – positive arithmetic remainder modulo .

Cat -- Concatenate several arrays

Cat(1,y,z) – concatenates the array / vector ‘y’ with ‘z’ row wise

Y = 1. 4. 6.

2. 7. 3.

4. 1. 1.

Z =

9. 8. 7.

5. 6. 4.

3. 2. 1.

cat(1,Y,Z)

ans =

1. 4. 6.

2. 7. 3.

4. 1. 1.

9. 8. 7.

5. 6. 4.

3. 2. 1.

Cat(2,y,z) – concatenates the array / vector ‘y’ with ‘z’ column wise

Matrix Analysis Commands

It helps in finding determinant, Rank and sum of eigen values

y = [1 -2; -2 0]

det(y) = -4

rank(y) = 2

trace(y) = 1 [sum(diag(x))]

spec(y) ➔ will provide the eigen values of matrix

= -1.5615528

2.5615528

RESULT :

Thus the Matrix constructors and operations are successfully executed

Experiment No-8

i. **OBJECTIVE: To study on Matrix Bitwise operations, Relational Operations and Logical Operations.**

ii. **SOURCE CODE :**

Relational operators: < <= > >= == ~=

```
X=5;        % X=5*ones(3,3);

X >=[1 2 3;4 5 6;7 8 9];
Output:
T  T  T
T  T  F
F F  F


x<=[1 2 3; 4 5 6; 7 8 9];
Output:

F  F  F
F T T
T T T

x<[1 2 3; 4 5 6; 7 8 9];
Output:

F  F  F
F  F  T
T T T

x~=[1 2 3; 4 5 6; 7 8 9];
 Output:

T T T
T F T
T T T
```

LOGICAL OPERATORS:

```
a=0;b=10;
if a and b
   disp("Condition is true");
else
   disp("Condition is false");
end
```

```
if a or b
   disp("Condition is true");
end
if (~a)
   disp("Condition is true");
end

exec('C:\Users\admin\Documents\relational.sce', -1)
```

Condition is false

Condition is true

BITWISE OPERATORS:

```
U = [0 0 1 1 0 1];
V= [0 1 1 0 0 1];
>> U |V
Output:

Ans =
F T T T F T

a = 60; % 0011 1100
b = 13; % 0000 1101

c=bitand(a,b);        % 12 = 0000 1100
ans  =

   12.

d=bitor(a,b); % 61 = 00111101
ans  =

   61.

e= bitxor(a,b); % 49 = 00110001
ans  =

   49.
```

RESULT :

The study on Relational, logical and bitwise operations on matrices is performed.

Experiment No-9

i. **OBJECTIVE: To write and execute programs that demonstrate on Control Structures (If-Else, If-elseif –else, Select) using SCI Notes.**

ii. **ALGORITHM:**

STEP 1: Start the program

STEP 2: Get the input from user using input() method.

STEP 3: pmodulo()_gives the positive remainder of the number.

STEP 4: pmodulo(number,2) tells if the number is divisible by 2. By which the the given number odd or even is determined.

STEP 5: Using select statement multiple cases are executed.

STEP 6: The dayNum returns the number for the given system date ie. Sun is considered 1, Mon = 2 , Tue = 2 etc.

STEP 7: The dayString returns the day ie. Mon,Tue etc.

STEP 8: Using dayString, the different cases are dealt and the statements are dealt accordingly.

STEP 9: A number is taken as input and checked for positive or negative or zero using the if-elseif – else condition.

iii. **SOURCE CODE :**

If- Else Program

To find whether a number is an even number or not

```
a=input("Enter a number:");
if pmodulo(a,2)==0
disp("Number is even");
else
disp("Number is odd");
end
```

SELECT STATEMENTS:

To print on what day we are in a week

```
[dayNum,dayString]=weekday(datenum());
 select dayString
        case "Mon" then
              disp("Start of work week");
        case "Tue" then
              disp("Day2");
        case "Wed" then
              disp("Day3");
        case "Thu" then
              disp("Day4");
       case "Fri" then
              disp("Last day of work week");
 else
       disp("Weekend");
 end
```

If-elseif –else condition:

To determine whether a number is +ve or –ve or zero

```
number = input("Enter a number:");
if number >0
   disp("positive");
elseif number< 0
   disp("negative");
else
   disp("Zero");
end
```

iv. SAMPLE INPUTS & OUTPUTS:

If- Else Program

```
exec('C:\Users\admin\Documents\if.sce', -1)
Enter a number:5

 Number is odd

-->exec('C:\Users\admin\Documents\if.sce', -1)
Enter a number:6

 Number is even
```

SELECT STATEMENTS:

```
 -->exec('C:\Users\admin\Documents\select.sce', -1)

  Start of work week
```

If-elseif –else condition:

```
exec('C:\Users\admin\Documents\nestedif.sce', -1)
Enter a number:4

 positive

-->exec('C:\Users\admin\Documents\nestedif.sce', -1)
Enter a number:0

 Zero

-->exec('C:\Users\admin\Documents\nestedif.sce', -1)
Enter a number:-7

 negative
```

RESULT :

The programs are executed using if-else, select, if-elseif-else statements.

Experiment No-10

OBJECTIVE: To write and execute programs that demonstrate on Control Structures (for, while, break and continue) using SCI Notes.

i. ALGORITHM:

STEP 1: Start the program.

STEP 2: For the given user input number, the factorial is to be found.

STEP 3: The variable fact is initialized to 1 and the input is stored in variable n.

STEP 4: The for loop is executed till and is multiplied with fact variable repeatedly.

STEP 5: The same functionality is executed using while loop.

STEP 6: For break and continue statement execution, the input is got from the user.

STEP 7: If the input is a positive number, the sum is calculated repeatedly, else it prompts for input if a negative number is entered.

STEP 8: If the input is zero, the program is stopped by printing the final sum.

ii. SOURCE CODE :

FOR LOOP:

To find factorial of given number

```
function fact = factorial(n)
fact =1;
for i=1:n
        fact=fact*i;
end
fprintf('Factorial of %d is %d\n',n,fact);
end
```

WHILE LOOP:

To find factorial of given number

```
function fact = factorial(n)
fact =1;
i=1;
while i<=n
   fact=fact*i;
   i=i+1;
end
fprintf('Factorial of %d is %d\n',n,fact);
end
```

BREAK AND CONTINUE STATEMENTS:

To find sum of all positive numbers entered by user (enter '0' to terminate)

```
a=1;
sum=0;
while a
   n=input('Enter a
   number:'); if n>0
      sum=s
   um+n;
   elseif
   n<0
      disp('Enter a positive
      number.'); continue;
   else
   br
   ea
   k;
end
printf('Sum of all positive numbers is %d',sum);
```

iii. SAMPLE INPUTS & OUTPUTS:

For loop:

```
>>
factorial(
5) ans  =
    120.
```

While loop:

```
>>
factorial(
5) ans  =
    120.
```

BREAK AND CONTINUE STATEMENTS:

```
Enter a
number:2 Enter
a number:-2
 Enter a positive
```

number. Enter a

number:1

Enter a number:0

Sum of all positive numbers is 3

RESULT:

The programs are executed using for, while and break-continue statements.