THE NEOTIA UNIVERSITY

LAB MANUAL

SIGNAL & SYSTEM LAB

2020-2021

CONTENT

Serial no	Name of the experiment	Experiment number
1	Generation of basic signals using MATLAB	**PC-EEP601/01**
2	Performing different operations on various signals using MATLAB	**PC-EEP601/02**
3	Simulation of even and odd component of a signal using MATLAB	**PC-EEP601/03**
4	Generation of various signals using MATLAB	**PC-EEP601/04**
5	Simulation of convolution of two signals starting from different indices using MATLAB	**PC-EEP601/05**
6	Simulation of Laplace transform and inverse Laplace transform	**PC-EEP601/06**
7	Simulation of Fourier transform and inverse Fourier transform	**PC-EEP601/07**
8	Simulation of Z	**PC-EEP601/08**

	transform and inverse Z transform	

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/01

2.0 NAME OF JOB/EXPERIMENT: Generation of basic signals using MATLAB

3.0 OBJECTIVE: To generate basic signals using MATLAB

4.0 PRINCIPLE:

SINUSOIDAL CURVE

The **sine wave** or **sinusoid** is a mathematical function that describes a smooth repetitive oscillation. A continuous time sinusoidal signal is given by $x(t) = A\sin(\Omega t+\theta)$, where A is the amplitude, Ω is the frequency in radians per second and θ is the phase angle in radians.

EXPONENTIAL CURVE

Y = exp(X) returns the exponential for each element of X. exp operates element-wise on arrays. A real exponential signal is defined as $x(t) = A e^{at}$, where A and a both are real. a can have both positive and negative values. For positive value of a, exponential will be growing exponential. For negative values of a, it will be decaying exponential.

Unit sample sequences:

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} = \{\ldots,0,0,1,0,0,\ldots.\}$$

Type equation here.

In MATAB the function **zeros** (1, N) a row vector of N zeros, which can be used to implement (n) over a finite interval. However, the logical relations n==0 is an elegant way of implementing (n).For example, to implement

$$\delta(n-n0)=\begin{cases}1, & n=0\\ 0, & n\neq 0\end{cases}$$

Over the $n_1 \leq n_0 \geq n_2$ interval, we will use the following MATLAB function.

Unit step sequences:

$u(n)=\begin{cases}1, & n\geq 0\\ 0, & n<0\end{cases}$ ={....0,0,1,1,1, ...}

↑

In MATLAB the function **ones (1, N)** generates a row vector of N ones. It can be used to generate u(n) over a finite interval. Once again an elegant approach is to use the logical relations n>=0.To implement

$$u(n-n0)=\begin{cases}1, & n\geq 0\\ 0, & n<0\end{cases}$$

Over the n1≤n0≤n2 interval, we will use the following MATLAB function.

RAMP SIGNAL

The ramp signal is defined as r(t) = t for t>=0

= 0 for t<0

Real- valued exponential sequences:

$x(n)=a^n \forall n; a\epsilon\mathbb{R}$

Complex-valued exponential signal:

$x(n)=e^{(\sigma+j\omega_0)}$n, $\forall n$

Where σ produces an attenuation (if<0) or amplification (if>0) and ω_0 is the frequency in radians. A MATLAB function exp is used to generate exponential sequences.

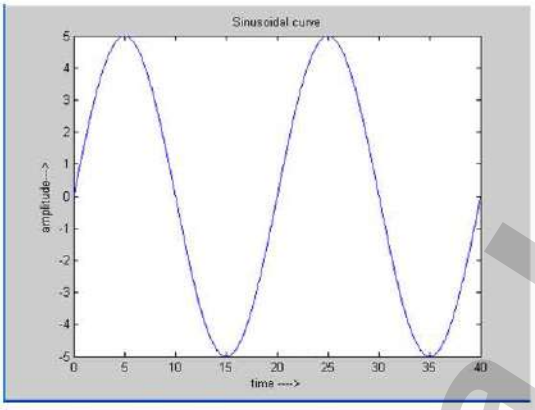
Program code:

SINUSOIDAL CURVE

```
t=0:0.05:40;
```

```
f=0.05;
a=5;
x=a*sin(2*pi*f*t);
plot(t,x);
title('Sinusoidal curve');
xlabel('time ---->');
ylabel('amplitude--->');
```

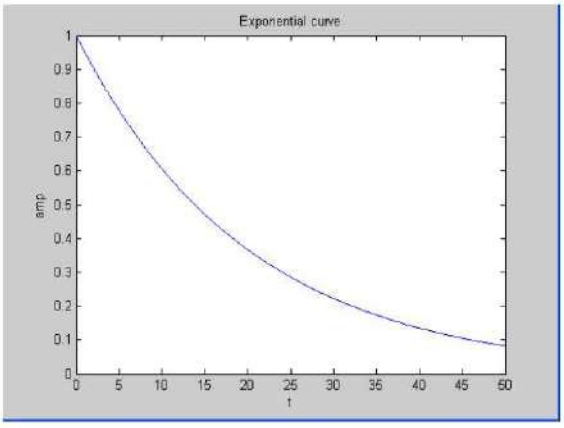
Output



EXPONENTIAL CURVE

```
t=0:0.005:50;
amp=0.05;
x=exp(-amp*t);
plot(t,x),
title( 'Exponential curve')
xlabel('t')
ylabel('amp')
```

Output



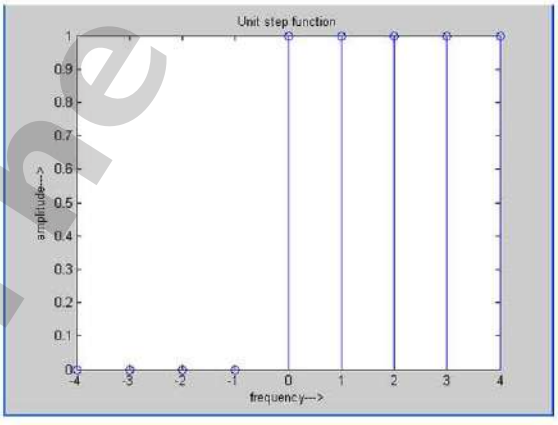
Unit step sequences:

```
function [x, n] = stepseq (n0, n1, n2)
% generates x (n) = u (n-n0; n1<=n<=n2)
[x, n]=stepseq (n0, n1, n2);
n= [n1:n2] ;
x= [(n-n0)>=0];
end;
```

test file:

```
n1=-4;
n2=4;
n0=0;
[x, n]=stepseq (n0, n1, n2);
stem(n,x)
```

output



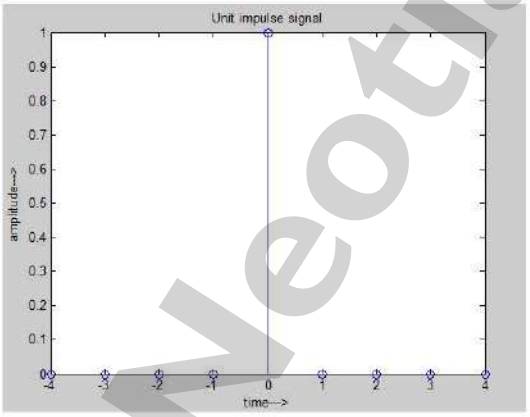
Unit sample or unit impulse sequences:

```
function [x, n] = impseq (n0, n1, n2)
%generates x(n) = delta (n-n0; n1<=n<=n2);
[x, n]= impseq (n0, n1, n2);
n= [n1:n2] ;
x= [(n-n0) ==0];
end;
```

test file:

```
n1=-4;
n2=4;
n0=0;
[x, n]=impseq (n0, n1, n2);
stem(n,x);
title('Unit impulse signal');
xlabel('time--->');
ylabel('amplitude--->');
```

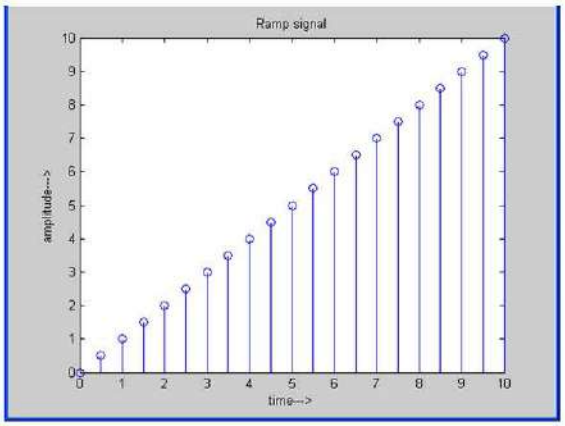
Output



Ramp signal

```
n=0:0.5:10;
x1=n;
stem(n,x1);
title('Ramp signal');
xlabel('time--->');
ylabel('amplitude--->');
```

Output

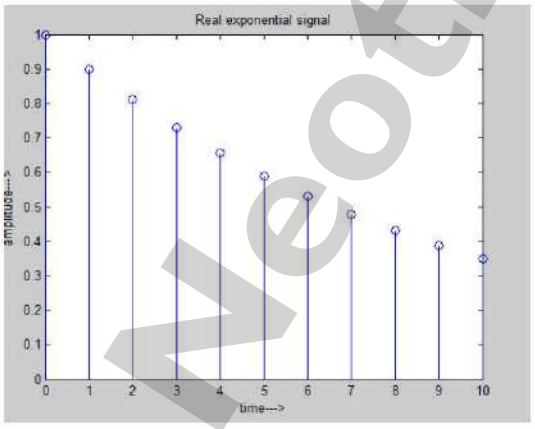


Real- valued exponential sequences

In MATLAB an array operator "• ^" is required to implement a real exoponential sequences.For example, to generate x(n)=$(0.9)^n$,$0 \leq n \leq 10$,we will need the following MATLAB script.

```
>> n = [0:10]; x = (0.9).^n;
```

Output



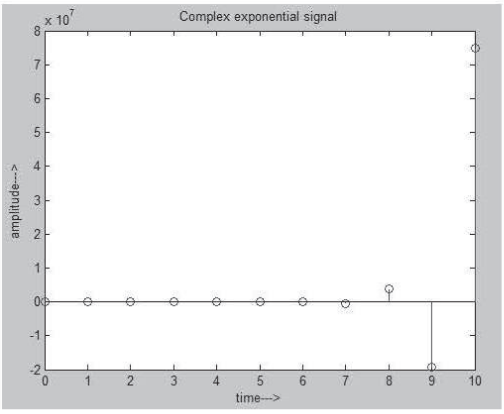
Complex-valued exponential signal:

For example, to generate x (n) =exp [(2+j3) n], $0 \leq n \leq 10$, we will need the following MATLAB script.

Programme code:

```
n= [0:10];

x=exp ((2+3j)*n);
```

Output



5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 SAFETY:

NOT APPLICABLE

7.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/02

2.0 NAME OF JOB/EXPERIMENT: Performing differentoperations on various signals using MATLAB

3.0 OBJECTIVE: To perform different operations on various signals using MATLAB

4.0 PRINCIPLE:

Signal addition: This is sample-by-sample addition given by

$$\{x1(n)\} + \{x2(n)\} = \{x1(n) + x2(n)\}$$

Signal multiplication: This is sample-by-sample (or "dot") multiplication given by

$$\{x1(n)\}.\ \{x2(n)\} = \{x1(n)\ x2(n)\}$$

Scaling: In this operation each sample is multiplied by a scalar a.

$$a\{x(n)\} = \{ax(n)\}$$

Shifting (Time-shifting): In this operation each sample of x(n) is shifted an amount k to obtain a shift sequence, given as

$y(n)=x(n-k)$

If we let $m=n-k$, then $n=m+k$ and the above operations is given by

$y(m+k) = \{x(m)\}$

Hence the operations has no effect on the vector x, but the vector n is changed by adding k to each element. This is shown in the below function **sigshift.**

Folding a signal (Time – reversal)

In this operation each sample of x(n) is folded or time reversed to the other side of the time axis , given as x(-n).

5.0 Program code

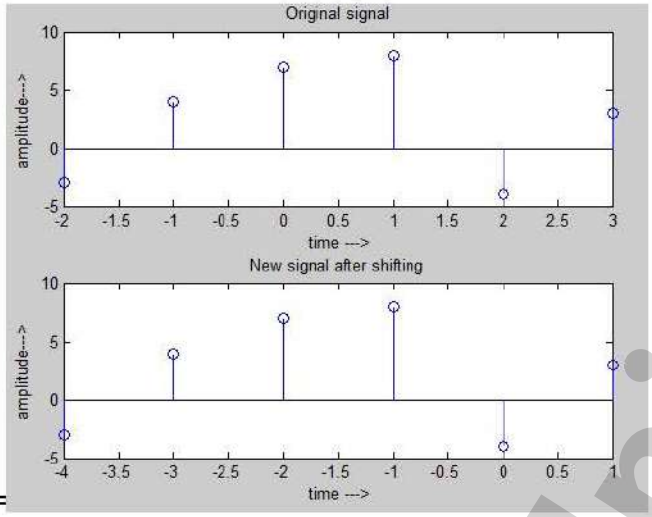
Time-shifting

```
function [y, n ] = sigshift(x,m,k)
  %implements y(n)= x(n-k)
 [y,n]= sigshift(x,m,k)
   n=m+k;
   y=x;
end
```

TEST PROGRAM:

```
clc;
m=-2:3;
x=[-3,4,7,8,-4,3];
n0=-2;
[y,n]=sigshift(x,m,n0);
subplot(211);
stem(m,x);
title('Original signal');
xlabel(' time --->');
ylabel('amplitude--->');
subplot(212);
stem(n,y);
title('New signal after shifting');
xlabel(' time --->');
ylabel('amplitude--->');
```

Output



=

Time-reversal

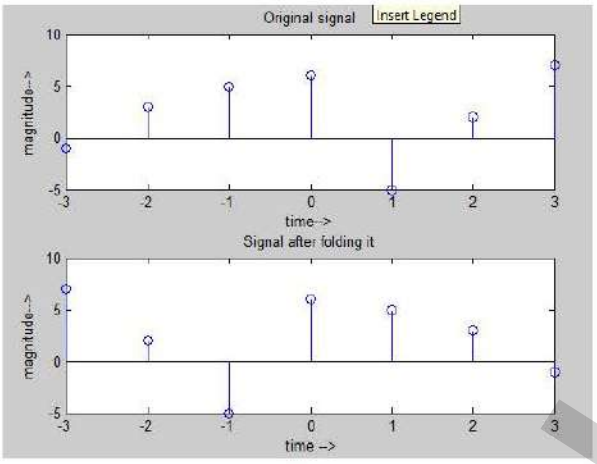
```
function[y n]=sigfold(x,n);
y=fliplr(x);
m=-fliplr(n);
end
```

TEST PROGRAM:

```
clc;
n=-3:3;
x=[-1,3,5,6,-5,2,7];
[y,m]=sigfold(x,n);
subplot(211);
stem(n,x);
title('Original signal');
xlabel('time-->');
ylabel('magnitude-->');
subplot(212);
stem(m,y);
title('Signal after folding it');
```

```
xlabel(' time -->');
ylabel('magnitude-->');
```

Result



Amplitude scaling

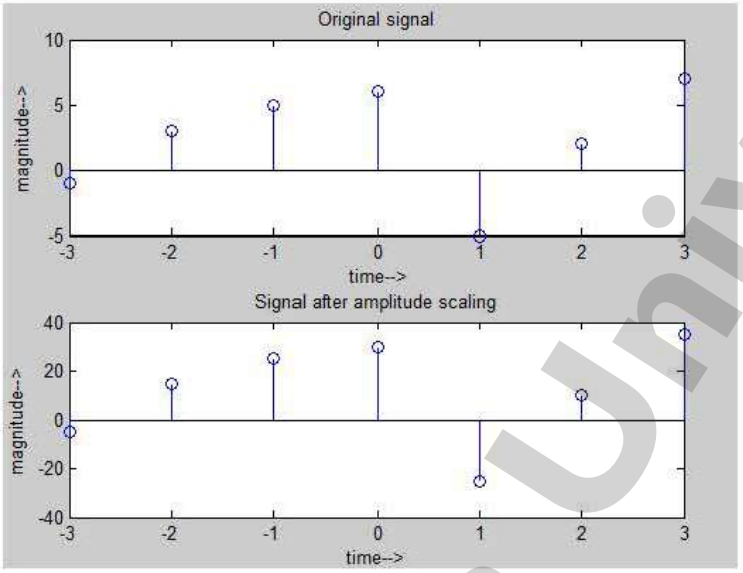
```
Function[y n]=sigamp(x,n,a);
y=a*x;
end
```

TEST PROGRAM:

```
clc;
n=-3:3;
x=[-1,3,5,6,-5,2,7];
a=5;
m=n;
[y,m]=sigamp(x,n,a);
subplot(211);
stem(n,x);
title('Original signal');
xlabel(' time-->');
ylabel('magnitude-->');
subplot(212);
stem(m,y);
```

```
title('Signal after amplitude scaling');
xlabel('time-->');
ylabel('magnitude-->');
```

Result



Addition of two signals

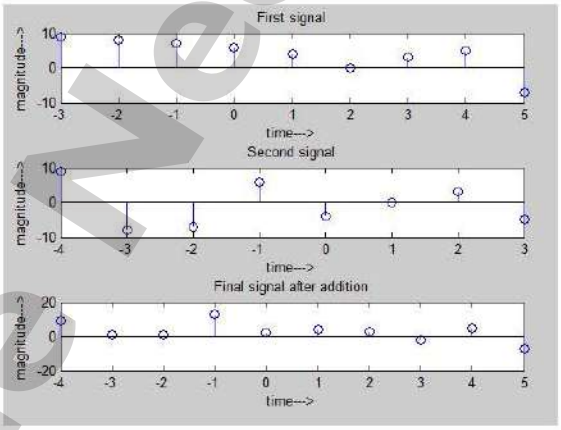
```
function[y n]=sigadd(x1,n1,x2,n2);
n=min(min(n1),min(n2)):max(max(n1),max(n2));
y1=zeros(1,length(n));
y2=y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y=y1+y2;
end
```

TEST PROGRAM :

```
clc;
n1=-3:5;
x1=[9,8,7,6,4,0,3,5,-7];
subplot(311);
stem(x1);
```

```
title('Addition of two signals');
title('First signal');
xlabel('time--->');
ylabel('magnitude--->');
n2=-3:5;
x2=[9,8,7,6,4,0,3,5,-7];
subplot(312);
stem(x2);
title('Second signal');
xlabel('time--->');
ylabel('magnitude--->');
[y n]=sigadd(x1,n1,x2,n2);
subplot(313);
stem(n,y);
title('Final signal after addition');
xlabel('time--->');
ylabel('magnitude--->');
```

Output



Multiplication of two signals

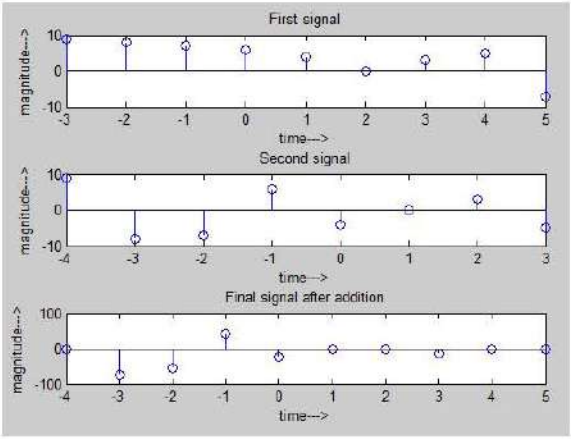
```
function[y n]=sigmult(x1,n1,x2,n2);
n=min(min(n1),min(n2)):max(max(n1),max(n2));
y1=zeros(1,length(n));
y2=y1;
y1(find((n>=min(n1))&(n<=max(n1)==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y=y1.*y2;
end
```

TEST PROGRAM :

```
clc;
n1=-3:5;
x1=[9,8,7,6,4,0,3,5,-7];
subplot(311);
stem(x1);
title('Addition of two signals');
title('First signal');
xlabel('time--->');
ylabel('magnitude--->');
n2=-3:5;
x2=[9,8,7,6,4,0,3,5,-7];
subplot(312);
stem(x2);
title('Second signal');
xlabel('time--->');
ylabel('magnitude--->');
[y n]=sigmult(x1,n1,x2,n2);
subplot(313);
stem(n,y);
title('Final signal after addition');
```

```
xlabel('time--->');
ylabel('magnitude--->');
```

Output



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/03

2.0 NAME OF JOB/EXPERIMENT: Simulation of even and odd component of a signal using MATLAB

3.0 OBJECTIVE: To simulate the even and odd component of a signal using MATLAB

4.0 PRINCIPLE:

EVEN AND ODD SYNTHESIS: A real- valued sequences $x_e(n)$ is called even(symmetric) if

$x_e(-n) = x_e(n)$

Similarly, a real-valued sequence $x_o(n)$ is called odd (asymmetric) if
$x_0(-n) = -x_o(n)$

Then any arbitrary real-valued sequences x (n) can be decomposed into its even and odd components

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

where the even and odd parts are given by

$x_e(n)=(1/2)*[x(n) +x(-n)]$ and $x_o(n) =(1/2)*[x(n)-x(-n)]$ respectively.

5.0 Program code:

1.Let x (n) =[-3,4,7,2,-5,-4].Decompose x (n) into even and odd components.

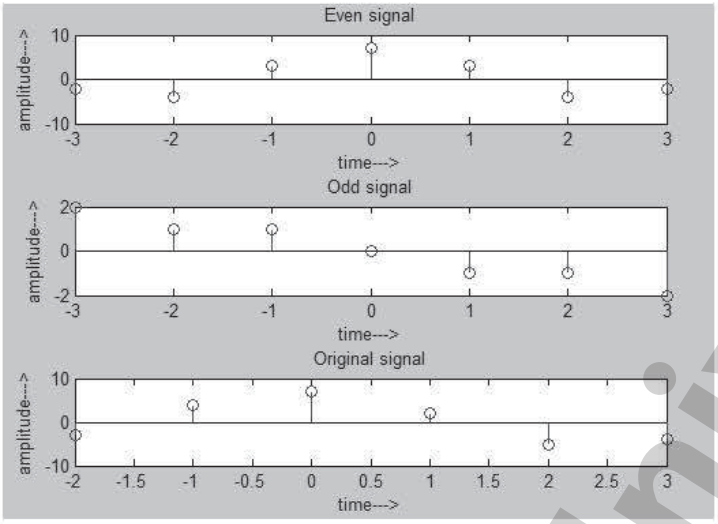
```
function[xe,xo,m]=evenodd(x,n);
m=-fliplr(n);
m1=min([m,n]);
m2=max([m,n]);
m=m1:m2;
nm=n(1)-m(1);
```

```
n1=1:length(n);
x1=zeros(1,length(m));
x1(n1+nm)=x;
xe=0.5*(x1+fliplr(x1));
xo=0.5*(x1-fliplr(x1));
end
```

TEST PROGRAM:

```
n=-2:3;
x=[-3,4,7,2,-5,-4];
[xe,xo,m]=evenodd(x,n);
subplot(311);
stem(m,xe);
title('Even signal');
xlabel('time--->');
ylabel('amplitude--->');
subplot(312);
stem(m,xo);
title('Odd signal');
xlabel('time--->');
ylabel('amplitude--->');
subplot(313);
stem(n,x);
title('Original signal');
xlabel('time--->');
ylabel('amplitude--->');
```

Output

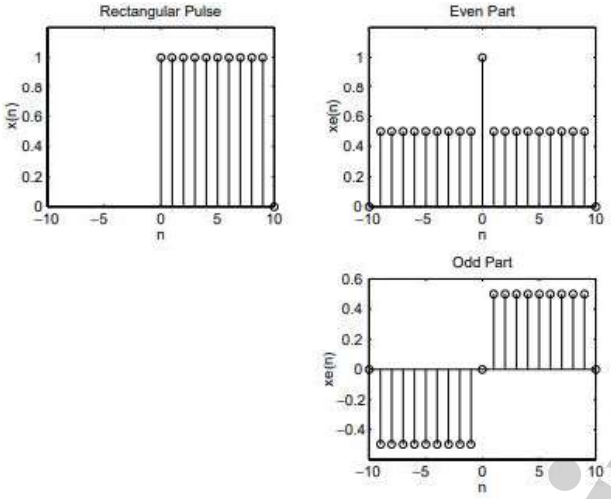


2. Let x(n) =u(n)-u(n-10).Decompose x (n) into even and odd components.

The sequence x (n), which is nonzero over $0 \leq n \leq 9$ is called a rectangular pulse. We will use MATLAB to determine and plots its even and odd parts.

```
n= [0:10]; x= stepseq (0, 0, 10)-stepseq (10, 0, 10);
[xe, xo, m] =evenodd(x, n);
subplot (2, 2, 1); stem (n, x); title ('rectangular pulse);
xlabel ('n'); ylabel ('x (n)'); axis ([-10, 10, 0, 1.2])
subplot (2, 2, 2); stem (m, Xe); title ('even parts');
xlabel ('n'); ylabel ('xe (n)'); axis ( [-10, 10, 0, 1.2])
subplot (2, 2, 4); stem (m, Xo); title ('odd parts');
xlabel ('n'); ylabel ('xo (n)'); axis( [-10, 10, 0.6, 0.6])
```

Output



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/04

2.0 NAME OF JOB/EXPERIMENT: Generation of various signals using MATLAB

3.0 OBJECTIVE: To generate various signals using MATLAB

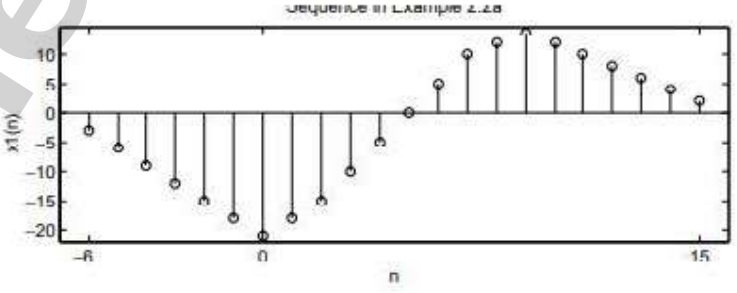
4.0 Program code

Simulate the signaly(n) = 2x(n-5) – 3x(n+4) wherex(n)=[1,2,3,9,4,10] where the starting index is -3.

Solution

```
clc;
n=-3:2;
x=[1,2,3,9,4,10];
[x1 n1]=sigshift(x,n,5);
[x2 n2]=sigshift(x,n,-4);
[y1,n]=sigadd(2*x1,n1,-3*x2,n2);
stem(n,y1);
title('Signal x(n)=2x(n-5)–3x(n+4)');
xlabel('n-->');
ylabel('y1(n)-->');
```

Output



Generate and plot the sequence over the indicated interval.

$x(n)=2\delta(n+2)-\delta(n-4), \quad -5 \leq n \leq 5$

Solution

```
n= [-5:5];
x= 2*impseq (-2,-5, 5)-impseq (4,-5, 5);
stem (n, x);
xlabel ('n');
ylabel ('x (n)');
```

Output

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 SAFETY:

NOT APPLICABLE

7.0 DISPOSAL:

NOT APPLICABLE

8.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/05

2.0 NAME OF JOB/EXPERIMENT: Simulation of convolution of two signals starting from different indices using MATLAB

3.0 OBJECTIVE: To simulate the convolution of two signals starting from different indices using MATLAB

4.0 PRINCIPLE

If arbitrary sequences are of infinite duration, then MATLAB cannot be used to directly compute to convolution. MATLAB does provide a built-in function called **conv** that computes the convolution between two finite-duration sequences. The **conv** function assumes that the two sequences begin at n=0 and is invoked by y=conv(x, h); to obtain the correct y(n) values.

However the **conv** neither provides nor accepts any timing information if the sequences have arbitrary support. What is needed is a beginning point and an end point of y(n).

Given finite duration of x(n) and h(n), it is easy to determine these points.

Let $\{X(n); nxb \leq n \leq nxe\}$ and $\{h(n); nhb \leq n \leq nhe\}$ be two finite duration sequences. Then referring to the above example we serve that the beginning and end points of y(n) are

$nyb = nxb + nhb$ And $nye = nxe + nhe$

respectively.

A simple modification of the conv function, called conv_m which perform the convolution of arbitrary support sequences can now be designed.

4.0 Programme code:

x(n) and h(n) are two sequences without having any timing information. For these two sequences, convolution can be done using conv function.

Solution

```
x= [3, 11, 7, 0, -1, 4, 2];

h=[2, 3, 0, -5, 2, 1];

y=conv(x, h);
```

Output

y=6 31 47 6 -51 -5 41 18 -22 -3 8 2

Finding convolution of two sequences with timing information. The signals are x=[3,11,7,0,-1,4,2] with starting index nx=-3, and h=[2,3,0,-5,2,1] with starting index nh=-1;

Solution

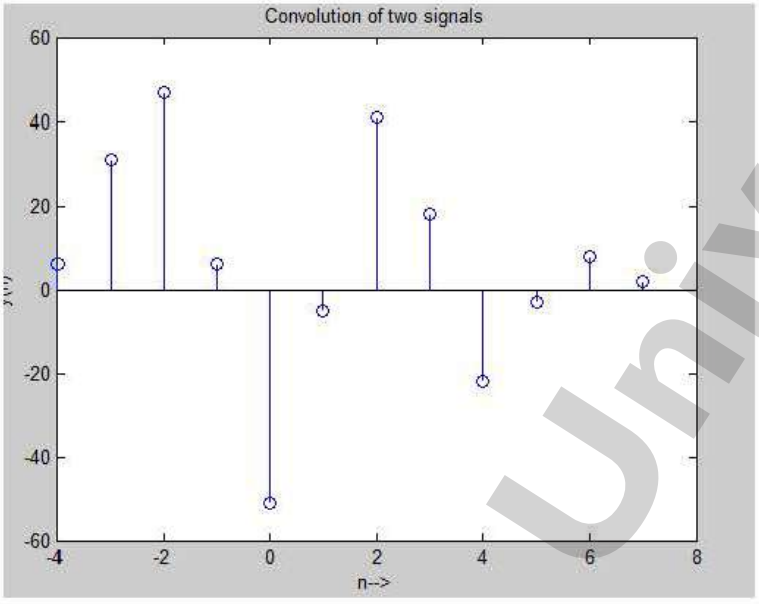
```
function [y, ny] = conv_m(x, nx, h, nh)
% modified convolution routine for signal processing
% [y, ny] = conv_m(x, nx, h, nh)
% [y, ny] = convolution result
% [x, nx] = first signal
% [h, nh] = second signal
nyb=nx(1)+nh(1);
nye=nx(length(x))+nh(length(h));
ny= [nyb; nye];
y=conv(x, h);
end
```

Test program

```
clc;
clear all;
close all;
x=[3,11,7,0,-1,4,2];
nx=-3:3;
h=[2,3,0,-5,2,1];
nh=-1:4;
[y,ny]=conv_m(x,nx,h,nh);
stem(ny,y);
title('Convolution of two signals');
xlabel('frequency-->');
```

```
ylabel('magnitude');
```

Output



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/06

2.0 NAME OF JOB/EXPERIMENT: Simulation of Laplace transform and inverse Laplace transform

3.0 OBJECTIVE: To simulateLaplace transform and inverse Laplace transform

4.0 PRINCIPLE:

DEFINITION OF THE LAPLACE TRANSFORM

There are two types of Laplace transforms:

The two-sided (or bilateral) Laplace transform:

The two-sided Laplace transform allows time functions to be nonzero for negative time.

$$L[x(t)] = X(s) = \int_{-\infty}^{\infty} x(t)\, e^{-st} dt$$

where s is the complex frequency given by $s = \sigma + j\omega$

The one-sided Laplace transform allows time functions to be zero for negative time.

$$L[x(t)] = X(s) = \int_{0}^{\infty} x(t)\, e^{-st} dt$$

where s is the complex frequency given by $s = \sigma + j\omega$

where σ (in Np/s) and ω (in rad/s) are the real and imaginary parts of s, respectively. The advantage of the bilateral Laplace transform is that it can handle both causal and noncausal signals over $-\infty$ to ∞.

In inverse Laplace transform the s-domain $X(s)$ is changed back to t-domain $x(t)$ and is given as

$$L^{-1}[X(s)] = x(t) = \int_{\sigma 1-\infty}^{\sigma 1+j\alpha} X(s)\, e^{st} ds$$

where the integration is performed along a straight line ($\sigma 1 + j\omega$, $-\infty < \omega < \infty$) .

5. 0 PROCEDURE:

Use MATLAB to find the Laplace transform of

$$\mathbf{X(t) = 2\partial(t) + e^{-3t}}$$

Matlab program:

```
syms x t;
x = 2*dirac(t) + exp(-3*t);
X = laplace(x);
```

Results:

X =1/(s+3) + 2

Use MATLAB to obtain the Laplace transform of

$$\mathbf{X(t) = u(t-1) - 2}e^{-t}$$

```
syms x t;
x = heaviside(t-1) + 2*exp(-t);
X = laplace(x);
```

Results:

$$\mathbf{X=}\frac{1}{s} * \boldsymbol{exp}(-\boldsymbol{s})\mathbf{-}\frac{2}{(s+1)}$$

Use MATLAB to find the inverse Laplace transform of

$$\mathbf{V(s)=}\frac{10s^2+4}{s(s+1)(s+2)^2}$$

```
syms s v V
V =(10*s^2 +4)/( s*(s+1)*(s+2)^2 );
v = ilaplace(V);
```

Results:

v =1-14*exp(-t)+(13+22*t)*exp(-2*t)

Use MATLAB to obtain *g*(*t*) if

$$\mathbf{G(s)=}\frac{s^3+2s+6}{s(s+3)(s+1)^2}$$

```
syms s v V
V =(s^3 +2*s+6)/( s*(s+3)*(s+1)^2 );
v = ilaplace(V);
```

Results:

(2 −3.25*exp(-t) − 1.5*t* exp(-t) + 2.25* exp(-t))*heaviside(t)

Use the residue command to find the Laplace inverse of

$$X(s)=\frac{4s^5+20s^4+16s^3+10s^2-12}{s^4+5s^3+8s^2+4s}$$

This is an indirect way of finding the inverse Laplace transform. We specify the numerator (num) and the denominator (den) of the transfer function $X(s)$. We findthe residues of $X(s)$ using the following code.

num = [4 20 16 10 0 −12]; % numerator coefficients in descendingpowers of s

den = [1 5 8 4 0]; % denominator coefficients in descendingpowers of s

[r,p,k] = residue(num,den); % call residue

Results

r =

-15.0000

46.0000

2.0000

-3.0000

p =

-2.0000

-2.0000

-1.0000

0

k =

4 0

This produces a vector *r* that has the residues and a vector *p* that has the correspondingpoles.

Use MATLAB to find the zeros and poles of

$$H(s)=\frac{s^2+3s+1}{s^3+4s^2+3s}$$

We use the command **roots** to find the roots of the numerator to get the zeros, and denominator to get the poles.

```
num = [1 3 1];
den = [1 4 3 0];
z=roots(num);
p=roots(den);
```

Results:

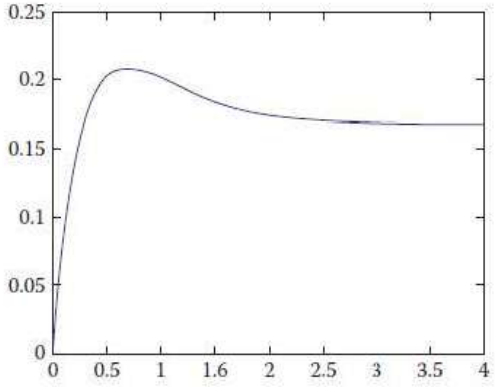
```
z =
-2.6180
-0.3820

p =
0
-3
-1
```

Use MATLAB to plot the step response of a system whose transfer function is

$$H(s)=\frac{s+1}{s^2+5s+6}$$

```
num = [1 1];
den = [1 5 6];
t = 0: 0.1: 4;
y=step(num,den,t);
plot(t,y)
```

Results:

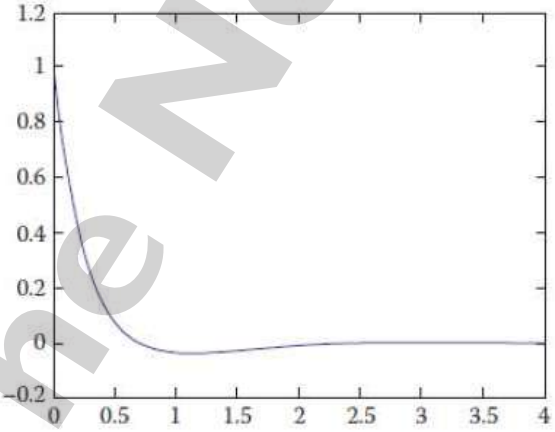


Use MATLAB to plot the impulse response of a system whose transfer function is

$$\mathbf{H(s)}=\frac{s+1}{s^2+5s+6}$$

```
num = [1 1];
den = [1 5 6];
t = 0: 0.1: 4;
y=impulse(num,den,t);
plot(t,y)
```

Results:



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/07

2.0 NAME OF JOB/EXPERIMENT: Simulation of Fouriertransform and inverse Fourier transform

3.0 OBJECTIVE: To simulateFourier transform and inverse Fouriertransform

4.0 PRINCIPLE:

DEFINITION OF FOURIER TRANSFORM

The mathematical representation of Fourier transform is given as

$$X(\omega) = F[x(t)] = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$

where F is the Fourier transform operator.

The **Fourier transform** of a signal $x(t)$ is the integration of the product of $x(t)$ and $e^{-j\omega t}$over the interval from $-\infty$ to $+\infty$.

$X(\omega)$ is an integral transformation of $x(t)$ from the time-domain to the frequency- domain and is generally a complex function. $X(\omega)$ is known as the spectrum of $x(t)$. Theplotofitsmagnitude$|X(\omega)|$versusωiscalledtheamplitudespectrum,whilethat ofitsphase$\angle X(\omega)$versusωiscalledthephasespectrum.Ifasignalisreal-valued,its magnitude spectrum is even, that is, $|X(\omega)| = |X(-\omega)|$ and its phase spectrum is odd, thatis,$\angle X(\omega) = -\angle X(-\omega)$.Boththeamplitudeandphasespectraprovidethephysical interpretation of the Fourier transform.

We can write x(t) in terms of $X(\omega)$ and we obtain the inverse Fourier transform as

$$x(t) = F^{-1}[X(\omega)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega$$

We say that the signal $x(t)$ and its transform $X(\omega)$ form a Fourier transform pair and denote their relationship by

$$x(t) \square X(\square)$$

Wewilldenotesignalsbylowercaseletters andtheirtransformsbyuppercaseletters.

Not all functions have Fourier transforms. The necessary conditions required for a function
to be transformable are called the **Dirichlet conditions**. These are listed as:

1. $x(t)$ isbounded.
2. $x(t)$ has a finite number of maxima and minima within any finiteinterval.
3. $x(t)$ has a finite number of discontinuities within any finite interval. Furthermore, each of these discontinuities must befinite.
4. $x(t)$ is absolutely integrable, thatis,

$$\int_{-\infty}^{\infty} |x(t)|dt \leq \infty$$

Therefore, absolutely integrable signals that are continuous or that have a finite number of discontinuities have Fourier transforms.

5.0 PROCEDURE:

COMPUTING WITHMATLAB

MATLAB® function **fourier** can be used to find the Fourier transform of a signal $x(t)$, while the function **ifourier** can be used to find the inverse Fourier transform of $X(\omega)$. How to use them can be displayed with the **help Fourier** and **help ifourier** commands.Sincetransferfunctionsarealreadyfunctionsoffrequency,itisstraight-forward and easy to plot them using MATLAB.

Use MATLAB to find the Fourier transform of

$$x(t) = u(t-2) - e^{-2t}u(t)$$

The MATLAB script for finding the Fourier transform of $x(t)$ is as follows:

```
syms x t;
 x=heaviside(t-2)-exp(-2*t)*Heaviside(t) ; %define the signal
 X = fourier(x);
pretty(X);  % simplify
```

Results:

```
                /                  i \      1
exp(-2 w i) | pi dirac(w) - - | - -------
                \                  w /   2 + w i
 '
```

Let $y(t) = 4\delta(t) + 2e^{-t}u(t)$. Use MATLAB to find $Y(\omega)$.

```
syms x t;
 x=4*dirac(t)+2*exp(-t)*heaviside(t); %define the signal
 X = fourier(x);
pretty(X);  % simplify
```

Results:

```
2
------- + 4
1 + w i
```

Use MATLAB to find the inverse Fourier transform of

$$X(\omega) = \frac{2\,j\omega}{1+j\omega}$$

Solution

The following MATLAB script is used to find the inverse of $X(\omega)$.

```
syms X xw;
X = 2*j*w/( 1 + j*w);
x = ifourier(X) ;
pretty(x);
```

Results:

```
4 pi exp(-x) dirac(x) - 4 pi exp(-x) heaviside(x)
-------------------------------------------------
                      2 pi
```

Use MATLAB to find the inverse Fourier transform of

$$X(\omega) = \frac{\delta(\omega)}{1+j2\omega}$$

Solution

The following MATLAB script is used to find the inverse of $X(\omega)$.

```
syms X xw;
X = dirac(w)/( 1 + 2*j*w);
x = ifourier(X) ;
pretty(x);
```

Results:

1

---

2 pi

Consider a system with transfer function

$$H(s)=\frac{100s^2}{s^4+25s^3+50s^2+400s+6000}\ ,\ s=jw$$

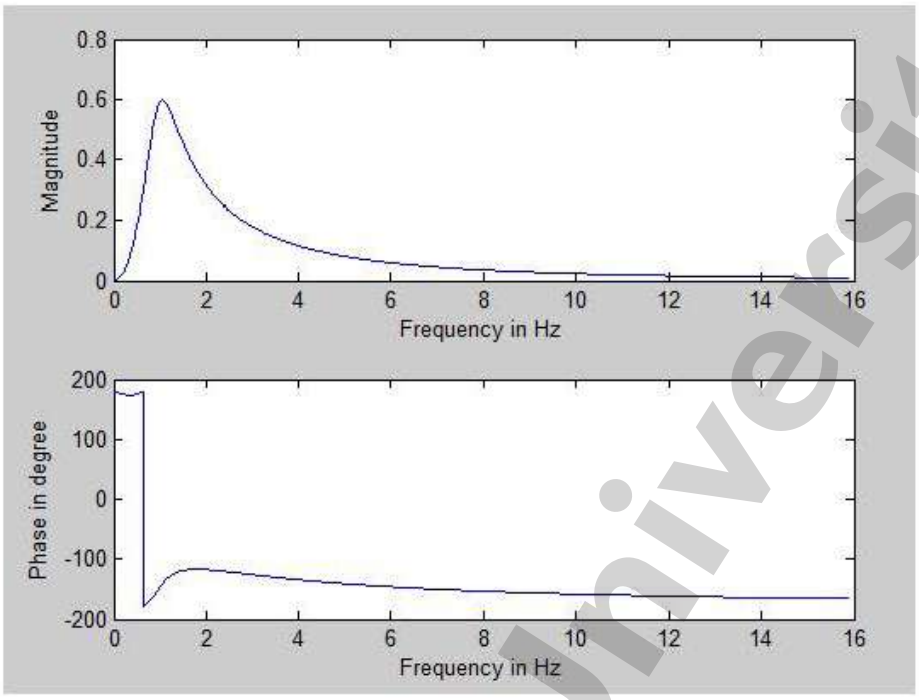
Use MATLAB to obtain the plot of $H(\omega)$.

Solution

The MATLAB script is as follows.

```
num = [100 0 0];
den = [ 1 25 50 4006000];
w = 0.1:0.1:100;
H = freqs(num, den,w);
 mag =abs(H);
phase=angle(H)*180/pi;      % converts phase todegrees
subplot(2,1,1);
plot(w/(2*pi), mag) ;
%xlabel('Frequency inHz') ;
%ylabel('Magnitude');
subplot(2,1,2) ;
plot(w/(2*pi),phase);
%xlabel('Frequency in Hz');
%ylabel('Phasein degree');
```

Results



UseMATLABtoplotthefrequencyresponsetothetransferfunction

$$H(s)=\frac{1}{s^3+7s^2+14s+8}\ ,\ s=jw$$

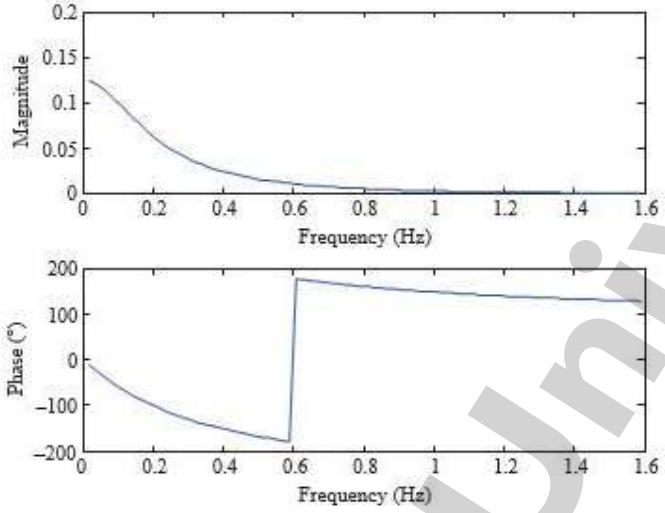
Solution

The MATLAB script is as follows.

```
num = [1];
den = [ 1 7148];
w = 0.1:0.1:100;
H = freqs(num, den,w);
 mag =abs(H);
phase=angle(H)*180/pi;      % converts phase todegrees
subplot(2,1,1);
plot(w/(2*pi), mag) ;
xlabel('Frequency(Hz)') ;
ylabel('Magnitude');
subplot(2,1,2) ;
plot(w/(2*pi),phase);
```

xlabel('Frequency(Hz)');

ylabel('Phase()');



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.:PC-EEP601/08

2.0 NAME OF JOB/EXPERIMENT: Simulation of Ztransform and inverse Ztransform

3.0 OBJECTIVE: To simulateZ transform and inverse Ztransform

4.0 PRINCIPLE:

The z-transform is the generalization of the discrete-time Fourier transform (DTFT). We recall that the DTFT of a signal $x[n]$ is given by

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\Omega n}$$

Inserting a factor ρ^{-n} in the above equation leads to

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x[n] \rho^{-n} e^{-j\Omega n} = \sum_{n=-\infty}^{\infty} x[n] (\rho e^{j\Omega})^{-n}$$

If we let $z = \rho e^{j\Omega}$. Then, above equation becomes

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

This is the two-sided (or bilateral) z-transform. The one-sided (or unilateral) z-transform is

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n}$$

Only the one-sided z-transform will be discussed in this chapter. $X(z)$ and $x[n]$ constitute a z-transform pair.

ote the following:
$x[n] \xrightarrow{Z} X(z)$

INVERSEz-TRANSFORM

The problem of finding the inverse z-transform is the problem of finding $x[n]$ for a given $X(z)$. The direct way to do this is to evaluate the contour integral:

$$x(n) = \frac{1}{2\Pi j}\oint_{\Gamma} X(z)z^{n-1}\,dz$$

where Γ is a closed contour that encloses the origin and includes all poles of $X(z)$. Γ is usually determined by the region of convergence. Itencirclesallsingularities in the ROC.

5.0 PROCEDURE:

COMPUTING WITHMATLAB®

In this section, we discuss how MATLAB can be used to do the following:

- Find the z-transform of a discrete-time signal using the MATLAB com- mand **ztrans**.
- Determine the inverse z-transform using MATLAB command**iztrans**.
- Determinethepolesofatransferfunctionusing MATLAB command**roots**

  and evaluate the stability of thesystem.
- Use the MATLAB commands **dstep** and **filter** to find the step response of discrete-time linear system.

Using MATLAB, find the z-transform of$x[n] = n\,u[n]$

Solution

The MATLAB script for doing this is

```
syms  X x nz;
x = n*heaviside(n);
X =ztrans(x)
```

Results:

```
X = z/(z^2 - 2*z + 1)
```

Use MATLAB to find the z-transform of $y[n] = n^2u[n]$

Solution

The MATLAB script for doing this is

```
syms  X x nz;
x = (n^2)*heaviside(n);
X =ztrans(x)
```

Results:

```
X =(z*(z + 1))/(z - 1)^3
```

Given the function

$$X(z) = \frac{6z+4}{z^2-3.5z+3}$$

Find the inverse z-transform $x[n]$.

Solution

We can do this in two ways.

***Method* 1:** Using the **iztrans** command, we have MATLAB script as follows.

```
syms X x n z
X = (6*z+4)/(z^2 -3.5*z + 3);
x = iztrans(X)
```

Result

```
x =
16*2^n - (52*(3/2)^n)/3 + (4*kroneckerDelta(n, 0))/3
```

Method 2: We can also use the **residue** or **residuez** command, which carries out partial-fraction expansion. We illustrate it in the following script.

```
num = [ 64]
den = [1 -3.53]
[r, p, k] = residuez(num, den)
```

MATLAB responds as follows (r = residues, r = poles, and k = direct terms).

```
r =
     32
    -26
p =
    2.0000
    1.5000
k =
     []
```

Use MATLAB to find the inverse z-transform of

$$X(z) = \frac{z-2}{z^2+3z+2}$$

Solution

Using the **iztrans** command, we have MATLAB script as follows.

```
syms X x n z
X = (z -2)/(z^2 +3*z +2);
x = iztrans(X)
```

Results:

$$x = 3*(-1)^n - 2*(-2)^n - \text{Kronecker Delta}\,(n, 0)$$

The transfer function of a system is given by

$$H(z)=\frac{2z-3}{z^3-2z^3-z-0.5}$$

Determine whether the system is stable.

Solution

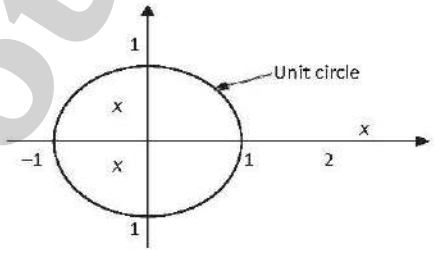
We only need to find the roots of the denominator to determine the system stability.

```
den = [1 -2 -1-0.5];
r = roots(den)
```

MATLAB returns

```
r = 2.4837
  -0.2418 +0.3779i
  -0.2418 -0.3779i
```

The poles are located on the complex plane. We notice that the magnitude of the first pole (2.4847) is greater than one and therefore lies outside the unit circle. (The other poles lie within the unit circle.) Hence, the system is unstable.



Pole locations for the above problem

Obtain the unit step response of a discrete-time system whose transform is

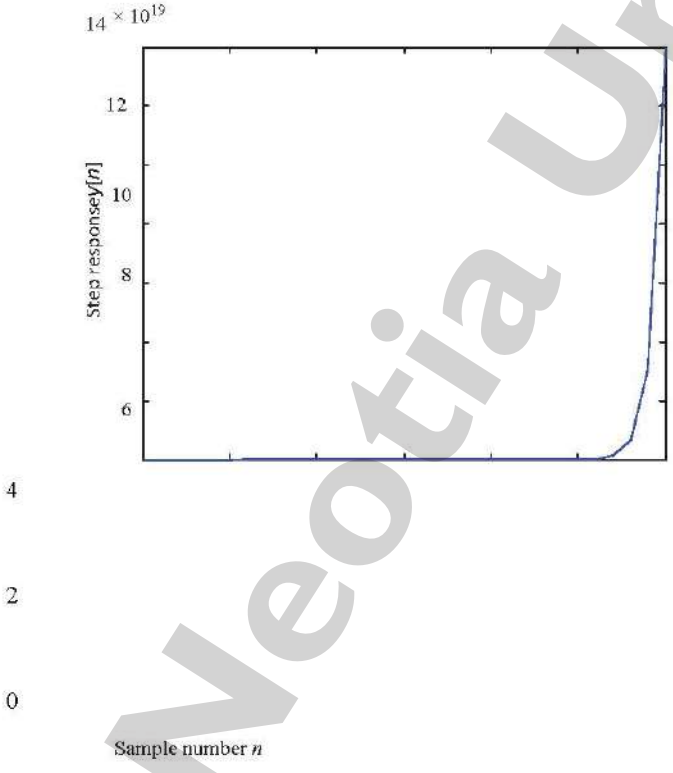
$$H(z)=\frac{3+z}{z^2-5z+2}$$

Solution

The MATLAB script is as follows. We use the MATLAB command **filter** in this problem. Given the transfer function, the input and initial conditions, filter returns the system output.

```
num = [13];
den = [ 1 -52];
n = 0:1:30;
x = [1*ones(size(n))]; % unit step input y = filter(num, den, x);
% d = length(y);
% n = 0:1:d-1
plot(n,y);
xlabel('Sample number n');
ylabel('step response y[n]')
```

Result



6.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

7.0SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 Write your final report as per the Work Instruction