

Scilab Textbook Companion for  
Signals And Systems  
by I. J. Nagrath, S. N. Sharan And R. Ranjan<sup>1</sup>

Created by  
Husnain Amjad Ali  
B.Tech (pursuing)  
Electronics Engineering  
Jamia Millia Islamia, New Delhi  
College Teacher  
Prof. Mubashshir Husain, JAMIA MILLIA ISLAMIA, NEW  
Cross-Checked by  
Sonanya tatikola

July 30, 2019

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website <http://scilab.in>

# Book Description

**Title:** Signals And Systems

**Author:** I. J. Nagrath, S. N. Sharan And R. Ranjan

**Publisher:** Tata McGraw - Hill Education Pvt. Ltd., New Delhi

**Edition:** 2

**Year:** 2010

**ISBN:** 13-97-8007014

Scilab numbering policy used in this document and the relation to the above book.

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

**AP** Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

# Contents

List of Scilab Codes	4
1 Introduction to Signal and Systems	6
2 Analysis of LTI continuous time system	19
3 Analysis of LTI Discrete time system	33
4 Discrete Fourier Transform And Fast Fourier Transform	38
5 Sampling	42
6 Transformed Networks	45
7 State Space Analysis	49
8 Stability Analysis of LTI Systems	50
9 Analog and Digital Filter Design	58
10 MATLAB Tools for Analysis of Signals and systems	64

# List of Scilab Codes

Exa 1.6	Shifting Scaling Time Reversal . . . . .	6
Exa 1.9	Check for Periodicity . . . . .	7
Exa 1.11	Check for Periodicity . . . . .	11
Exa 1.13	Check for Periodicity . . . . .	13
Exa 1.30	Integral of Unit step function . . . . .	15
Exa 1.37	Shifting Time Reversal of discrete time signals	18
Exa 2.2	Convolution . . . . .	19
Exa 2.11.a	Fourier transform of impulse function . . . . .	21
Exa 2.11.b	Fourier transform of exponential function . . . . .	21
Exa 2.12	Fourier transform of Gate function . . . . .	24
Exa 2.14	Fourier transform one sided exponential function . . . . .	24
Exa 2.20.a	Laplace transform of unit impulse . . . . .	26
Exa 2.23	Laplace Transform . . . . .	27
Exa 2.26.a	Inv Laplace . . . . .	27
Exa 2.26.b	Inv Laplace . . . . .	28
Exa 2.26.c	Inv Laplace . . . . .	28
Exa 2.28	Laplace Transform . . . . .	28
Exa 2.33	Plot the spectrum . . . . .	28
Exa 2.39	Convolution of two signals . . . . .	30
Exa 3.1	Convolution sum method . . . . .	33
Exa 3.2	Graphical Convolution . . . . .	33
Exa 3.17.b	Z Transform . . . . .	35
Exa 3.17.c	Z Transform . . . . .	35
Exa 3.17a	Z Transform . . . . .	36
Exa 3.19	Final Value Theorem . . . . .	36
Exa 3.21	Inverse Ztransform . . . . .	36
Exa 3.22	Z inverse . . . . .	37

Exa 4.4	DFT computation . . . . .	38
Exa 4.5	DFT computation . . . . .	38
Exa 4.6	DFT computation . . . . .	39
Exa 4.9	DFT computation . . . . .	39
Exa 4.28	Circular convolution . . . . .	40
Exa 4.29	Record Length . . . . .	40
Exa 4.31	Sampling rate and DFT size . . . . .	40
Exa 4.32	DFT computation . . . . .	41
Exa 5.1	Minimum Sampling Frequency . . . . .	42
Exa 5.2.a	Minimum Sampling Interval . . . . .	42
Exa 5.2.b	Minimum Sampling Interval . . . . .	43
Exa 5.6	Continuous time Frequency . . . . .	43
Exa 5.9	Aliasing . . . . .	43
Exa 5.12	Find the frequency . . . . .	44
Exa 5.14	Sampling period . . . . .	44
Exa 6.1	Current flowing in a network . . . . .	45
Exa 6.2	voltage across inductor . . . . .	45
Exa 6.3	voltage across capacitor . . . . .	46
Exa 6.9	Bode Plot . . . . .	46
Exa 6.10	Bode Plot . . . . .	47
Exa 7.1	State Space Representation . . . . .	49
Exa 7.2	State Space Representation . . . . .	49
Exa 8.1.a	check for HURWITZ . . . . .	50
Exa 8.1.b	check for HURWITZ . . . . .	51
Exa 8.2	checking stability . . . . .	52
Exa 8.3	checking stability . . . . .	53
Exa 8.4	checking stability . . . . .	54
Exa 8.5	checking stability . . . . .	55
Exa 8.9	checking stability . . . . .	55
Exa 8.11	Stability of discrete time system . . . . .	56
Exa 9.1	BPF . . . . .	58
Exa 9.2	Band Stop Filter . . . . .	59
Exa 9.3	FIR Filter . . . . .	60
Exa 9.5	Low Pass Filter . . . . .	61
Exa 9.7	2nd order Digital Butterworth Filter . . . . .	63
Exa 10.1	Solving Linear Equation . . . . .	64
Exa 10.3	Find the step and impulse response . . . . .	64
Exa 10.9	Find the impulse response . . . . .	64

Exa 10.12	Find convolution and plot the result . . . . .	65
Exa 10.13	Find the step response . . . . .	66

# List of Figures

1.1	Shifting Scaling Time Reversal . . . . .	8
1.2	Shifting Scaling Time Reversal . . . . .	9
1.3	Check for Periodicity . . . . .	10
1.4	Check for Periodicity . . . . .	12
1.5	Check for Periodicity . . . . .	14
1.6	Integral of Unit step function . . . . .	16
1.7	Shifting Time Reversal of discrete time signals . . . . .	17
2.1	Convolution . . . . .	20
2.2	Fourier transform of exponential function . . . . .	22
2.3	Fourier transform of Gate function . . . . .	23
2.4	Fourier transform one sided exponential function . . . . .	26
2.5	Fourier transform one sided exponential function . . . . .	27
2.6	Plot the spectrum . . . . .	29
2.7	Convolution of two signals . . . . .	31
3.1	Graphical Convolution . . . . .	34
6.1	Bode Plot . . . . .	47
6.2	Bode Plot . . . . .	48
9.1	FIR Filter . . . . .	60
9.2	Low Pass Filter . . . . .	62



# Chapter 1

## Introduction to Signal and Systems

Scilab code Exa 1.6 Shifting Scaling Time Reversal

```
1 //shifting and scaling
2 //Ex 1.6
3 clear;
4 clc;
5 close;
6 t = 0:1/100:2;
7 for i = 1:length(t)
8     if t(i)<1 then
9         x(i) = (2)*t(i) ;
10        else
11            x(i)= 2;
12        end
13 end
14 for i = length(t)+1:2*length(t)
15 x(i) = 0;
16 end
17 t1=0:1/100:4.01;
18 figure
19 subplot(3,1,1),plot2d(t1,x)
```

```

20 a=gca();
21 t2=t1-1;
22 subplot(3,1,2),plot2d(t2,x)
23 xtitle('x(t+1)')
24 a.y_location='origin'
25 a=gca();
26 subplot(3,1,3),plot2d(t1/1.5,x)
27 xtitle('x(1.5t)')
28 figure
29 a=gca();
30 subplot(2,1,1),plot2d((t2/1.5),x)
31 xtitle('x(1.5t+1)')
32 a.y_location='origin'
33 a=gca();
34 t3=-3:1/100:1.01;
35 subplot(2,1,2),plot2d(t3,x($:-1:1))
36 xtitle('x(-t+1)')
37 a.y_location = 'origin';

```

---

### Scilab code Exa 1.9 Check for Periodicity

```

1 //Ex 1.9
2 //Check For periodicity
3 clc;
4 //pi=22/7
5 k=1;
6 N=2*7*k/5;
7 z=2*N;
8 n=0::1:z //Defining discrete time

```

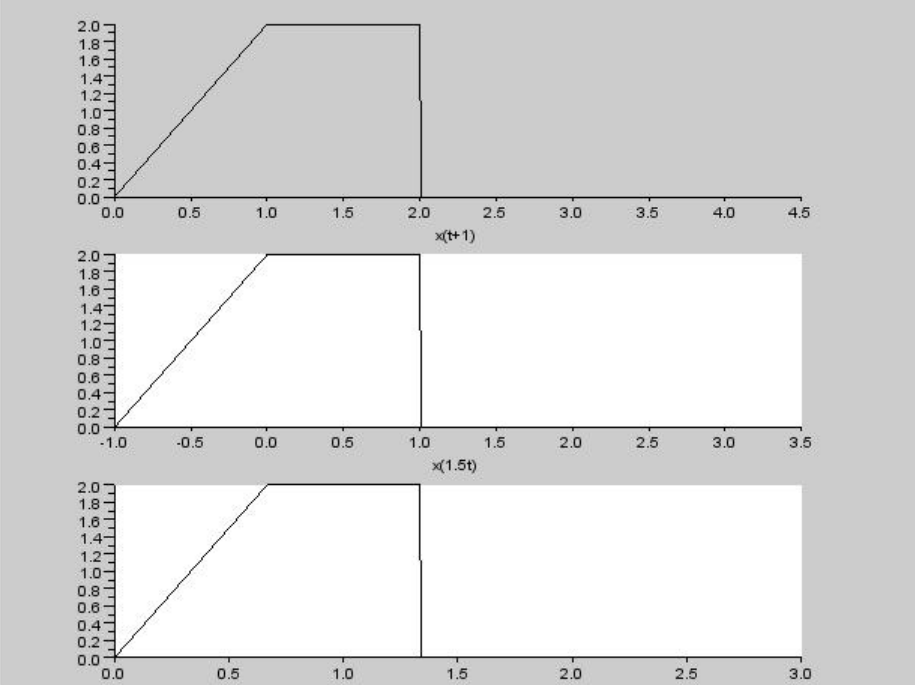


Figure 1.1: Shifting Scaling Time Reversal

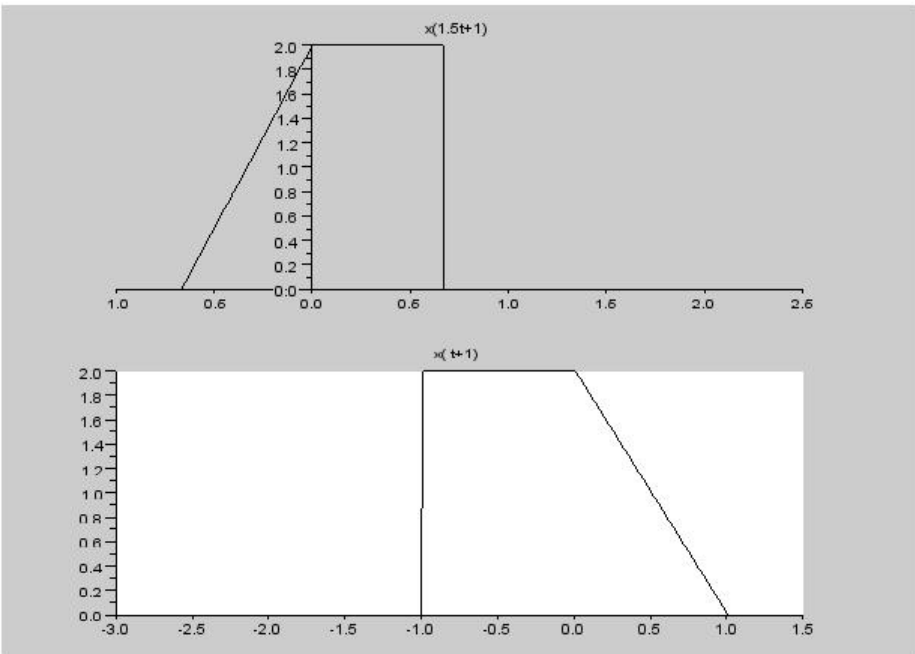


Figure 1.2: Shifting Scaling Time Reversal

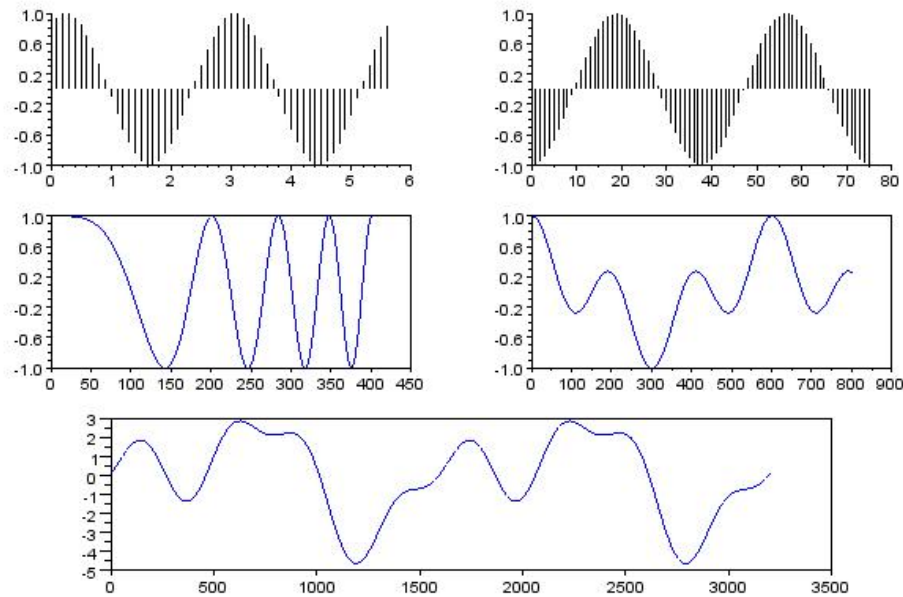


Figure 1.3: Check for Periodicity

```

9 x=sin((5*pi*n/7)+1); //sinusoid function
10 subplot(3,2,1),plot2d3(n,x) //plotting the sinusoid
    showing it as periodic 2piN=5piN/7
11 disp('the plot shows the above signal is periodic');
12
13 k=1;
14 N=2*pi*k*6;
15 z=2*N;
16 n=0:1:z //Defining discrete time
17 x=cos((n/6)-pi);
18 subplot(3,2,2),plot2d3(n,x); //the plot shows the
    above signal is periodic
19 disp('the plot shows the above signal is periodic');
20
21 k=1;
22 N=sqrt(2*k*2);
23 z=2*N;
24 n=0:1/100:z //Defining discrete time
25 x=cos(((n.^2)*pi)/2);

```

```

26 subplot(3,2,3),plot(x);//the plot shows that the
    above signal is not periodic
27 disp('the plot shows the above signal is not
    periodic');
28
29 k=1;
30 N=2*k;
31 z=2*N;
32 n=0:1/100:2*z //Defining discrete time
33 x=cos(n*pi/3).*cos(2*pi*n/3);
34 subplot(3,2,4),plot(x);//the plot shows that the
    above signal is periodic
35 disp('the plot shows the above signal is periodic');
36
37 k=1;
38 N=2*k*8;
39 z=2*N;
40 n=0:1/100:z //Defining discrete time
41 x=2*cos(n*pi/4)+sin((n*pi/2)-(pi/3))-2*cos((n*pi
    /8)+(pi/3));
42 subplot(3,1,3),plot(x);//the plot shows that the
    above signal is periodic
43 disp('the plot shows the above signal is periodic');

```

---

### Scilab code Exa 1.11 Check for Periodicity

```

1 //Ex 1.11
2 //Check for peridicity
3 clc;
4 T=2*pi/8;
5 z=2*T;
6 t=0:0.001:z
7 x=%i*exp(%i*8*t);

```

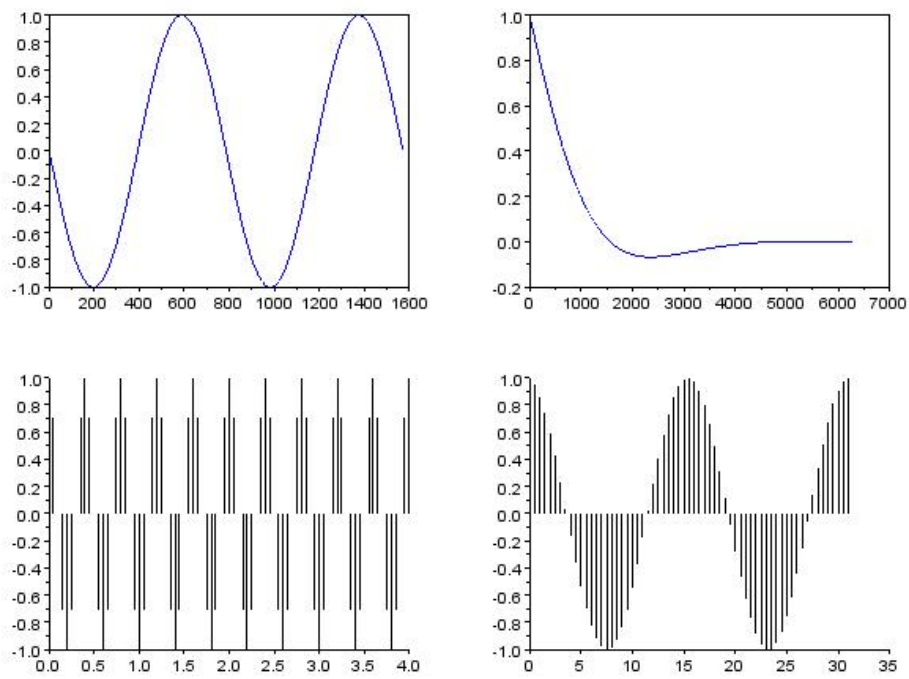


Figure 1.4: Check for Periodicity

```

8 subplot(2,2,1),plot(x);//the plot shows that the
   above signal is periodic
9
10 T=2*%pi/(-1+%i);
11 z=2*T;
12 disp('T cannot be complex so non periodic');
13 t=0:-0.001:z
14 x=exp((-1+%i)*abs(t));
15 subplot(2,2,2),plot(x);//the plot shows that the
   above signal is not periodic
16
17 N=2*%pi/%pi;
18 z=2*N;
19 n=0:0.05:z
20 x=exp(%i*5*%pi*n);//exp(i*(4*pi+pi)*n)=exp(i*pi*n)
21 subplot(2,2,3),plot2d3(n,x);//the plot shows that
   the above signal is periodic
22
23 k=1;
24 N=2*%pi*5/2;
25 z=2*N;
26 n=0:0.5:z
27 x=exp((%i*2/5)*(n+(1/3)));
28 subplot(2,2,4),plot2d3(n,x);//the plot shows that
   the above signal is periodic

```

---

### Scilab code Exa 1.13 Check for Periodicity

```

1 //Ex 1.13
2 //Check for periodicity
3 clc;
4 T=2*%pi/6;
5 t=0:0.001:T*2

```



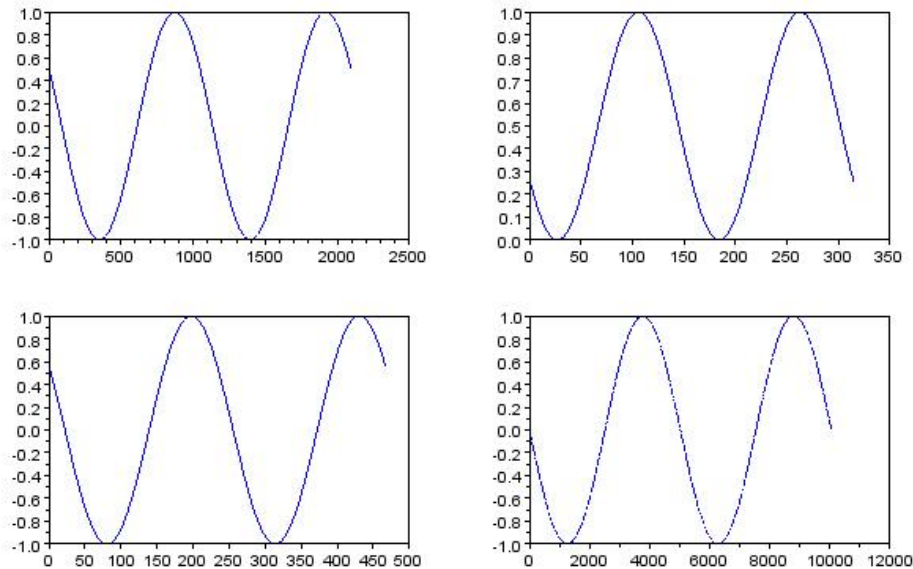


Figure 1.5: Check for Periodicity

```

6 x=cos((6*t)+%pi/3);
7 subplot(3,2,1),plot(x);
8 disp('the plot shows that the above signal is
      periodic ');
9
10 T=2*%pi/(%i*%pi);
11 t=0:0.001:T*2
12 x=exp(%i*(%pi*abs(t-1))); //exp(%i*(%pi*t-1))=exp(%i*
      %pi*t)/exp(%i)
13 //since the period is a complex no so non periodic
14 disp('T cannot be complex so non periodic T=2*%pi/(
      %i*%pi) ');
15
16 //pi=22/7
17 T=2*%pi/4; //calc the fundamental period
18 z=2*T;
19 t=0:1/100:z
20 x=(cos(2*t+%pi/3))^2; //sinusoid function
21 subplot(3,2,2),plot(x)

```

```

22 disp('the plot shows that the above signal is
    periodic ');
23
24 k=1;
25 N=2*k*7/6;
26 z=2*N;
27 n=0:1/100:z
28 x=cos((6*pi*n/7)+1);
29 subplot(3,2,3),plot(x);//the plot shows that the
    above signal is periodic
30 disp('the plot shows that the above signal is
    periodic ');
31
32 k=1;
33 N=2*pi*k*8;
34 z=2*N;
35 n=0:1/100:z
36 x=sin((n/8)-pi);
37 subplot(3,2,4),plot(x);//the plot shows that the
    above signal is periodic
38 disp('the plot shows that the above signal is
    periodic ');
39
40 k=1;
41 N=2*k*12; //2*cos(n*pi/4).*cos(n*pi/3)=cos(7*n*pi
    /12)-cos(n*pi/12)
42 z=2*N;
43 n=0:1/100:z
44 x=2*cos(n*pi/4).*cos(n*pi/3);
45 subplot(3,1,3),plot(x);//the plot shows that the
    above signal is periodic
46 disp('the plot shows the above signal is periodic ');

```

---

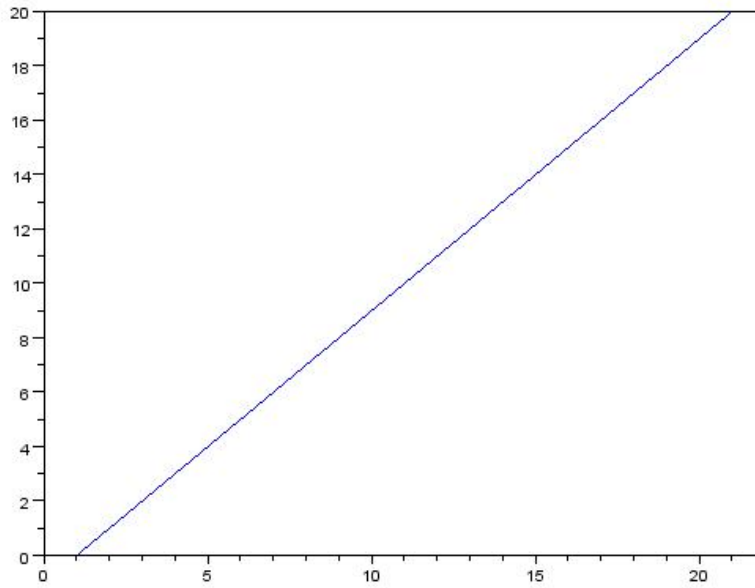


Figure 1.6: Integral of Unit step function

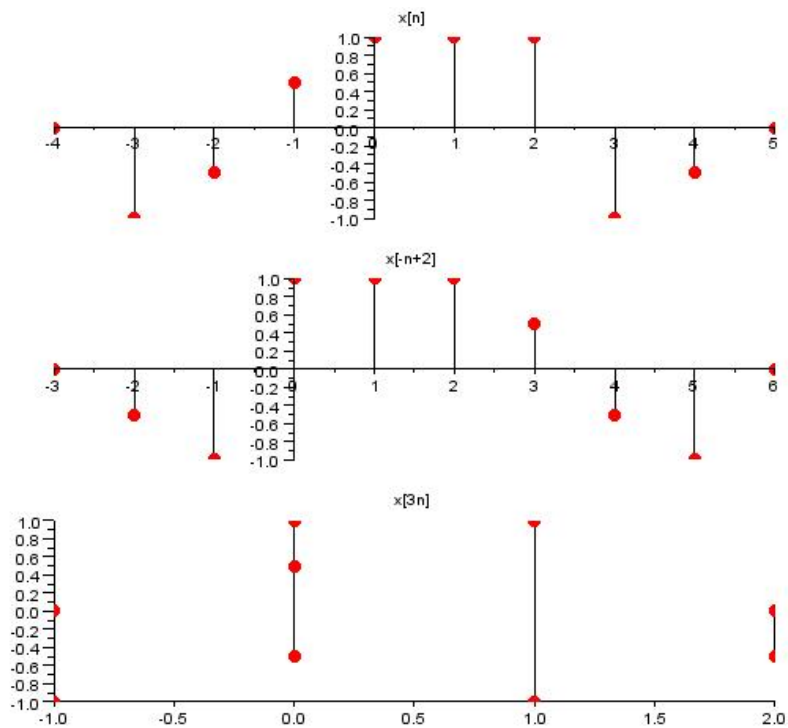


Figure 1.7: Shifting Time Reversal of discrete time signals

### Scilab code Exa 1.30 Integral of Unit step function

```

1 //Ex 1.30
2 //Integral of unit step function
3 clc;
4 x0=0; //lower bound
5 x1=0:20; //upper bound vector
6 X=integrate('1', 'x', x0, x1);
7 //integration of unt step seq
8 //resulting in ramp seq
9 plot(X)

```

---

Scilab code Exa 1.37 Shifting Time Reversal of discrete time signals

```
1 //Ex 1.37
2 //shifting and scaling discrete signals
3 clear ;
4 clc;
5 close;
6 t=-4:5;
7 x=[0 -1 -.5 .5 1 1 1 -1 -.5 0]
8 a=gca();
9 subplot(3,1,1),plot2d3(t,x)
10 subplot(3,1,1),plot(t,x,'r.')
11 xtitle('x[n]')
12 a.x_location='origin';
13 a.y_location='origin';
14
15 t1=-5:4;
16 t2=t1+2;
17 a=gca();
18 subplot(3,1,2),plot2d3(t2,x($:-1:1))
19 subplot(3,1,2),plot(t2,x($:-1:1),'r.')
20 xtitle('x[-n+2]')
21
22 a=gca()
23 subplot(3,1,3),plot2d3(ceil(t/3),x)
24 subplot(3,1,3),plot(ceil(t/3),x,'r.')
25 xtitle('x[3n]')
26 a.x_location='origin';
27 a.y_location='origin';
```

---

## Chapter 2

# Analysis of LTI continuous time system

Scilab code Exa 2.2 Convolution

```
1 //Ex 2.2
2 clc;
3 h=ones(1,10);
4 N2=0:length(h)-1;
5 a=0.5;//constant a>0
6 for t=1:10
7     x(t)=exp(-a*(t-1));
8 end
9 N1=0:length(x)-1;
10 y=convol(x,h)-1;
11 N=0:length(x)+length(h)-2;
12
13 subplot(3,1,1),plot2d(N2,h)
14 xtitle('Impulse Response','t','h(t)');
15
16 subplot(3,1,2),plot2d(N1,x)
17 xtitle('Input Response','t','x(t)');
```

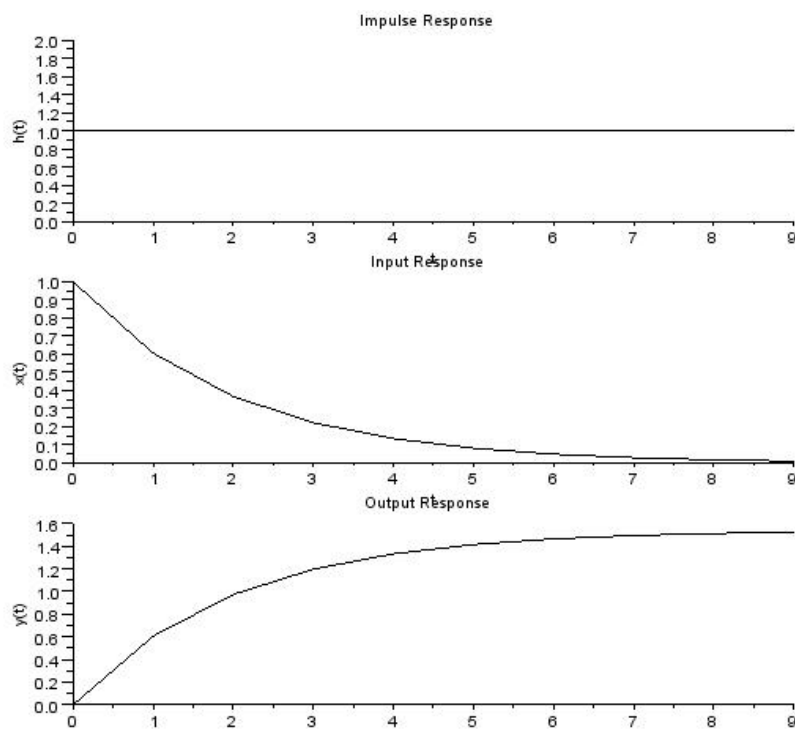


Figure 2.1: Convolution

```
18
19
20 subplot(3,1,3),plot2d(N(1:10),y(1:10))
21 xtitle('Output Response','t','y(t)');
```

---

Scilab code Exa 2.11.a Fourier transform of impulse function

```
1 //Fourier Transform of unit impulse
2 clc;
3 syms s t
4 X=symsum(1*%e^(-s*t),t,0,0);
```

---

Scilab code Exa 2.11.b Fourier transform of exponential function

```
1 clc;
2 A=1;
3 Dt=0.005;
4 t=-4.5:Dt:4.5;
5 xt=exp(-A*abs(t));
6 Wmax=2*%pi*1;
7 K=4;
8 k=0:(K/1000):K;
9 W=k*Wmax/K;
10 XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
11 XW = real(XW);
12 W=[-mtlb_fliplr(W),W(2:1001)];
13 XW=[mtlb_fliplr(XW),XW(2:1001)];
14 subplot(1,1,1)
15 subplot(2,1,1);
16 a=gca();
17 a.y_location="origin";
```



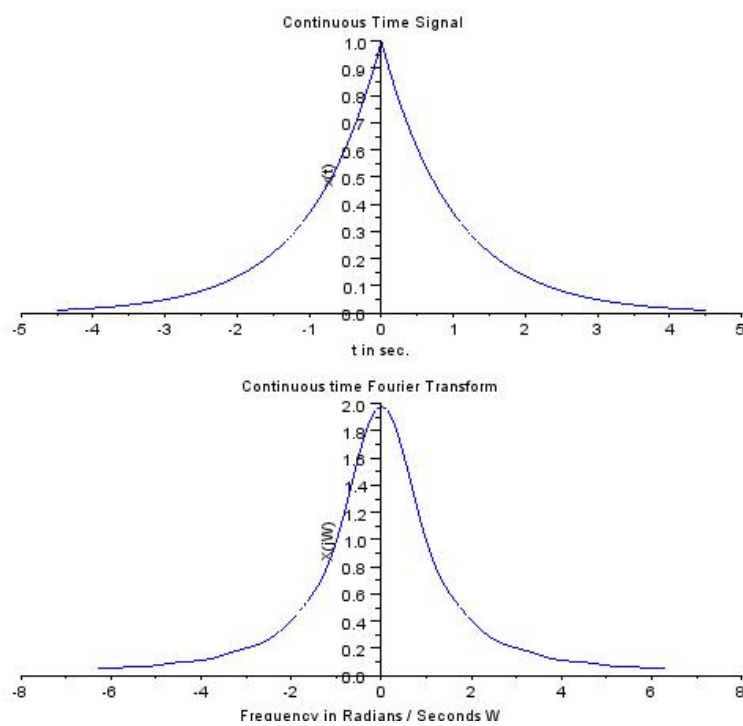


Figure 2.2: Fourier transform of exponential function

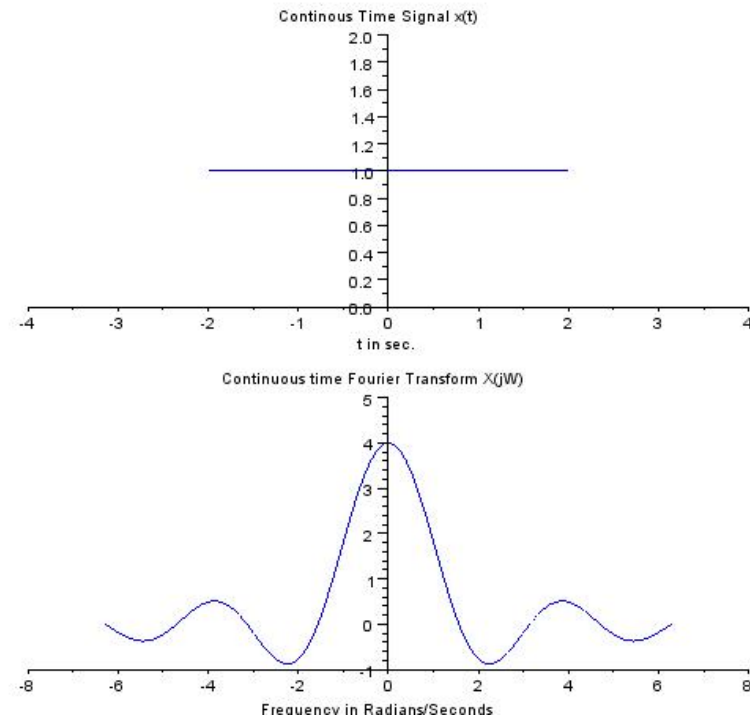


Figure 2.3: Fourier transform of Gate function

```

18 plot(t,xt);
19 xlabel( ' t in sec. ' );
20 ylabel( ' x(t) ' )
21 title( ' Continuous Time Signal ' )
22 subplot(2,1,2);
23 a=gca();
24 a.y_location="origin";
25 plot(W,XW);
26 xlabel( ' Frequency in Radians / Seconds W' );
27 ylabel( 'X(jW) ' )
28 title( ' Continuous time Fourier Transform ' )

```

---

### Scilab code Exa 2.12 Fourier transform of Gate function

```
1 //Ex 2.12
2 clc;
3 clear;
4 A=1;
5 Dt=0.005;
6 T1=4;
7 t=-T1/2:Dt:T1/2;
8 for i=1:length(t)
9 xt(i)=A;
10 end
11 Wmax=2*%pi*1;
12 K=4;
13 k=0:(K/1000):K;
14 W=k*Wmax/K;
15 xt=xt';
16 XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
17 XW_Mag=real(XW);
18 W=[-mtlbfliplr(W),W(2:1001)];
19 XW_Mag=[mtlbfliplr(XW_Mag),XW_Mag(2:1001)];
20 subplot(2,1,1);
21 a=gca();
22 a.data_bounds=[-4,0;4,2];
23 a.y_location="origin";
24 plot(t,xt);
25 xlabel('t in sec. ');
26 title('Continuous Time Signal x(t)');
27 subplot(2,1,2);
28 a=gca();
29 a.y_location="origin";
30 plot(W,XW_Mag);
31 xlabel('Frequency in Radians/Seconds');
32 title('Continuous time Fourier Transform X(jW)');
```

---

## Scilab code Exa 2.14 Fourier transform one sided exponential function

```
1 //Ex 2.14
2 //Fourier Transform
3 clc ;
4 clear;
5 A=1;
6 Dt=0.005;
7 t=0:Dt:10;
8 xt=exp(-A*t);
9 Wmax=2*%pi*1;
10 K=4;
11 k=0:(K/1000):K;
12 W=k*Wmax/K;
13 XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
14 XW_Mag=abs(XW);
15 W=[-mtlbfliplr(W),W(2:1001)];
16 XW_Mag=[mtlbfliplr(XW_Mag),XW_Mag(2:1001)];
17 [XW_Phase,db]=phasemag(XW);
18 XW_Phase=[-mtlbfliplr(XW_Phase),XW_Phase(2:1001)];
19 figure
20 a=gca();
21 a.y_location="origin";
22 plot(t,xt);
23 xlabel('t in sec. ');
24 ylabel('x(t) ');
25 title('Continuous Time Signal');
26 figure
27 subplot(2,1,1);
28 a=gca();
29 a.y_location="origin";
30 plot(W,XW_Mag);
31 xlabel('Frequency in Radians/Seconds>W');
32 ylabel('abs(X(jW)) ');
33 title('Magnitude Response (CTFT) ');
34 subplot(2,1,2);
35 a=gca();
36 a.y_location="origin";
```

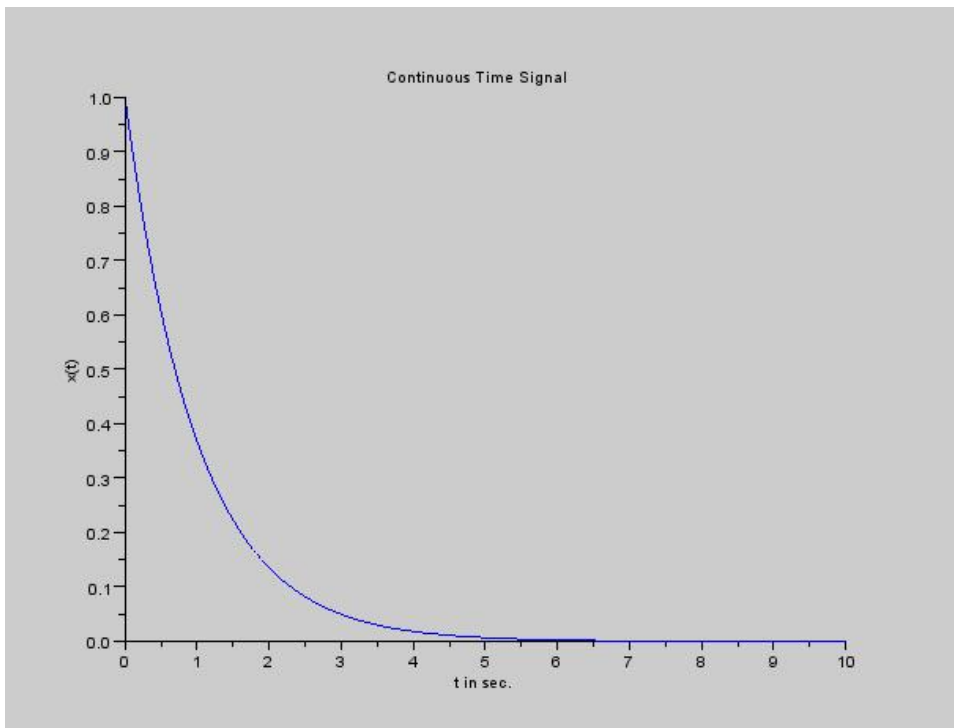


Figure 2.4: Fourier transform one sided exponential function

```

37 a.x_location="origin";
38 plot(W,XW_Phase*%pi/180) ;
39 xlabel(' Frequency in Radians/ S e c o n d s           > W
      ' );
40 ylabel('<X(jW)')
41 title('Phase Response (CTFT) in Radians');

```

---

Scilab code Exa 2.20.a Laplace transform of unit impulse

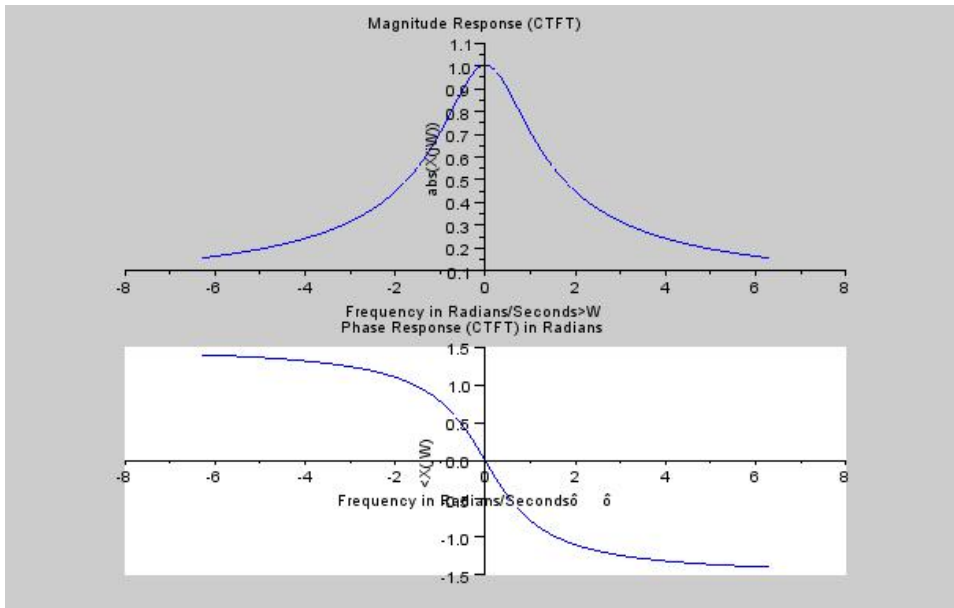


Figure 2.5: Fourier transform one sided exponential function

```

1 //laplace of unit impulse
2 clc;
3 syms s;
4 X=symsum(1*%e^(-s*t),t,0,0);

```

---

#### Scilab code Exa 2.23 Laplace Transform

```

1 syms a t;
2 x=exp(-a*t);
3 y=diff(x,t)
4 X=laplace(y);

```

---

#### Scilab code Exa 2.26.a Inv Laplace

```
1 syms s;  
2 x=ilaplace((s^2+2*s+1)/(s*(s+2)*(s+1)));  
3 disp (x);
```

---

#### Scilab code Exa 2.26.b Inv Laplace

```
1 syms s;  
2 x=ilaplace((2*s+3)/((s^2+4*s+5)*(s+1)));  
3 disp (x);
```

---

#### Scilab code Exa 2.26.c Inv Laplace

```
1 syms s;  
2 x=ilaplace(1/((s^2)*(s+1)));  
3 disp (x);
```

---

#### Scilab code Exa 2.28 Laplace Transform

```
1 clc;  
2 syms s ;  
3 X=5/((s+2)*(s-3));  
4 x=ilaplace(X);
```

---

#### Scilab code Exa 2.33 Plot the spectrum

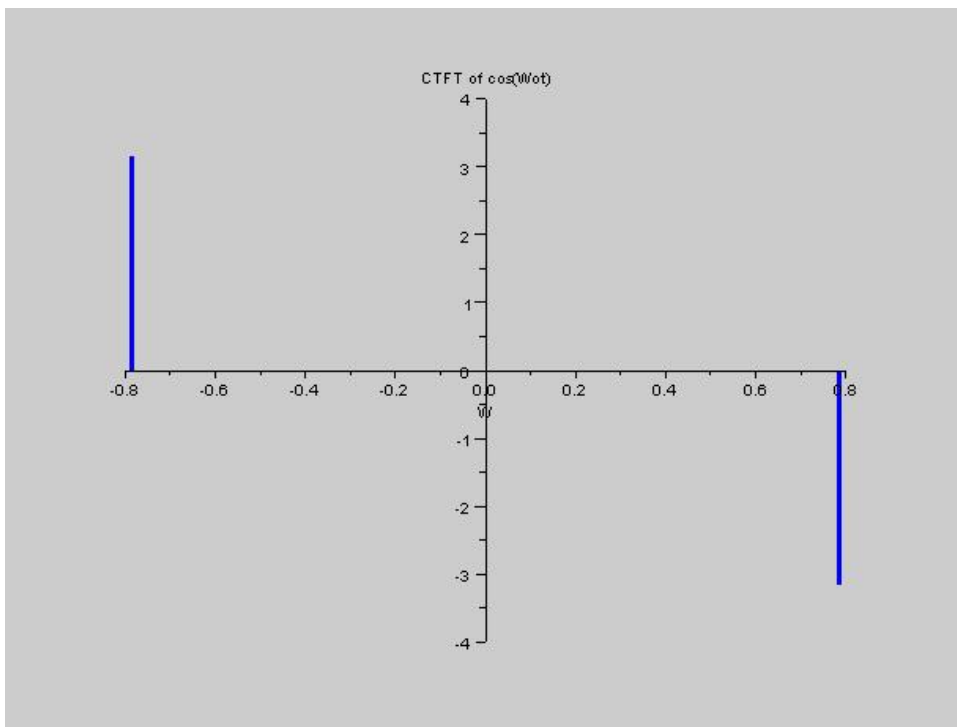


Figure 2.6: Plot the spectrum



```

1 //Ex 2.33
2 clc;
3 clear;
4 T1=2;
5 T=4*T1;
6 Wo=2*%pi/T;
7 W=[-Wo,0,Wo];
8 ak=(2*%pi*Wo*T1/%pi)/sqrt(-1);
9 XW=[-ak,0,ak];
10 ak1=- (2*%pi*Wo*T1/%pi);
11 XW1=[-ak1,0,ak1];
12 figure
13 a=gca();
14 a.y_location="origin";
15 a.x_location="origin";
16 plot2d3('gnn',W,XW1,2);
17 poly1=a.children(1).children(1);
18 poly1.thickness=3;
19 xlabel('W');
20 title('CTFT of cos(Wot)');

```

---

### Scilab code Exa 2.39 Convolution of two signals

```

1 //Ex 2.39
2 clc;
3 clear;
4 A=1;
5 Dt=0.005;
6 T1=1;
7 t=-T1:Dt:T1;
8 for i=1:length(t)
9 xt(i)=A;
10 end

```

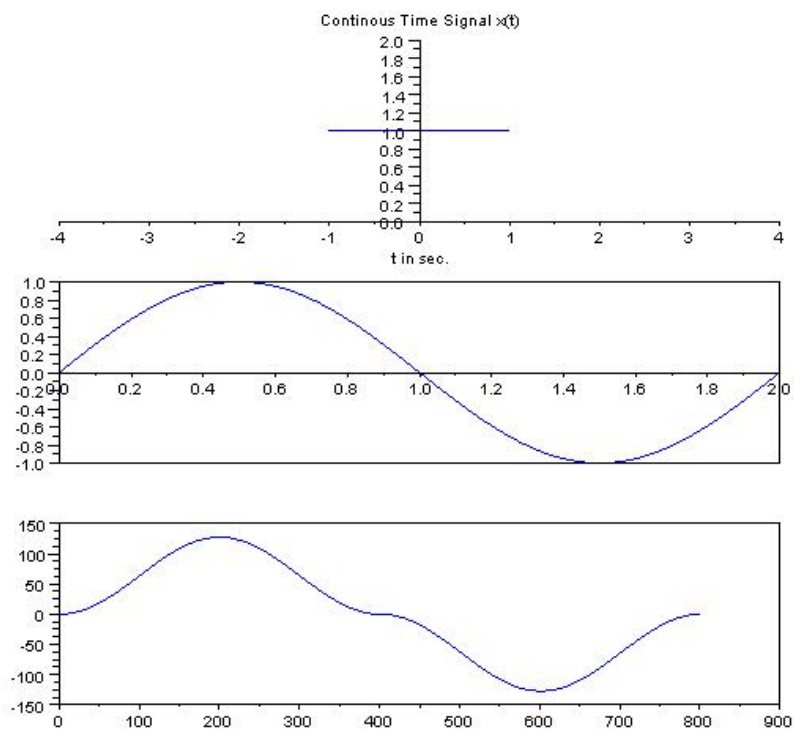


Figure 2.7: Convolution of two signals

```
11 xt=xt';
12 t1=0:0.005:2;
13 for j=1:length(t1)
14     x1(j)=sin(%pi*t1(j));
15 end
16 subplot(3,1,1);
17 a=gca();
18 a.data_bounds=[-4,0;4,2];
19 a.y_location="origin";
20 plot(t,xt);
21 xlabel('t in sec. ');
22 title('Continuous Time Signal x(t)');
23 subplot(3,1,2);
24 a=gca();
25 plot(t1,x1);
26 y=convol(x1,xt);
27 subplot(3,1,3);
28 a.y_location="origin";
29 a.x_location="origin";
30 plot(y);
```

---

# Chapter 3

## Analysis of LTI Discrete time system

Scilab code Exa 3.1 Convolution sum method

```
1  clc;
2  //k=2;
3  r=1;
4  k=0:1:2;
5  h=0.5^(k);
6  l1=convol(h,r);
7  disp(l1);
8  //k=3;
9  r=1;
10 k=0:1:3;
11 h=0.5^(k);
12 l2=convol(h,r);
13 disp(l2);
```

---

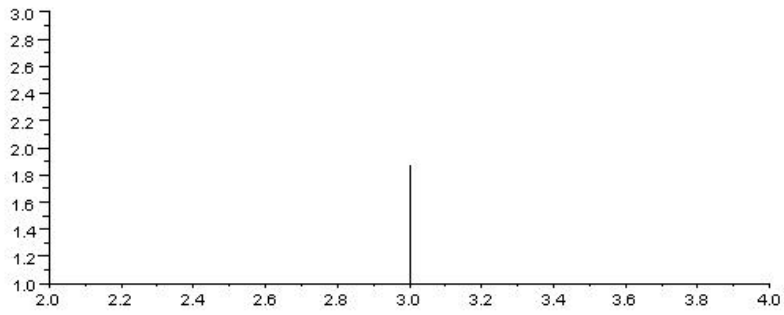
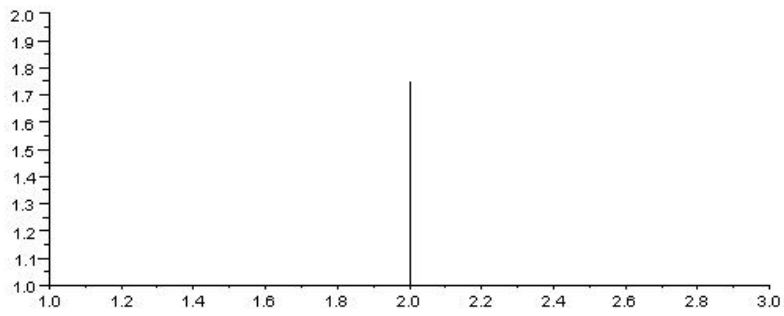


Figure 3.1: Graphical Convolution

### Scilab code Exa 3.2 Graphical Convolution

```
1  clc ;
2  //k=2;
3  r=1;
4  k=0:1:2;
5  h=0.5^(k);
6  l1=convol(h,r);
7  s1=sum(l1);
8  disp(s1);
9  n=2;
10 subplot(2,1,1),plot2d3(n,s1);
11 //k=3;
12 r=1;
13 k=0:1:3;
14 h=0.5^(k);
15 l2=convol(h,r);
16 s2=sum(l2);
17 disp(s2);
18 m=3;
19 subplot(2,1,2),plot2d3(m,s2);
```

---

### Scilab code Exa 3.17.b Z Transform

```
1  //Ex 3.17b
2  clc ;
3  syms z n;
4  x=1;
5  X=symsum(x*z^(-n),n,0,%inf);
6  disp(X);
```

---

### Scilab code Exa 3.17.c Z Transform

```

1 //Ex 3.17c
2 clc;
3 syms z n B k;
4 x=exp(B*k);
5 X=symsum(x*z^(-n),n,0,%inf);
6 disp(X);

```

---

#### Scilab code Exa 3.17a Z Transform

```

1 //Ex 3.17a
2 clc;
3 syms z n;
4 x=1;
5 X=symsum(x*z^(-n),n,0,0);
6 disp(X);

```

---

#### Scilab code Exa 3.19 Final Value Theorem

```

1 //Ex 3.19
2 clc;
3 syms z;
4 c=(z-1)/(1+0.5*z^(-1))+(z^(-1)*(z-1))/((1+0.5*z^(-1)
    )*(1-z^(-1)));
5 l=limit(c,z,1);
6 disp(l);

```

---

#### Scilab code Exa 3.21 Inverse Ztransform

```

1 //Ex 3.21
2 clc;

```

```
3 z=%z;
4 x=ldiv(z^2,(z^3)-(1.7*(z^2))+0.8*z+0.1,3);
5 dims=1;
6 xo=[0];
7 y=cat(dims,xo,x);
8 //degree of Num polynomial And Den Polynomial must
   be same
9 // else Zeros are padded accordingly on the basis of
   Std Eq.
10 disp(y);
```

---

#### Scilab code Exa 3.22 Z inverse

```
1 //Ex 2.22 a
2 clc;
3 z=%z;
4 x=ldiv(z,z-0.4,4);
5 disp(x);
```

---



## Chapter 4

# Discrete Fourier Transform And Fast Fourier Transform

Scilab code Exa 4.4 DFT computation

```
1 clc;  
2 a=0.5;  
3 for n=1:10  
4     x(n)=a^(n-1);  
5 end  
6 X=fft(x,-1);
```

---

Scilab code Exa 4.5 DFT computation

```
1 clc;  
2 clear;  
3 N1=8;  
4 n=-N1:N1;  
5 for i=1:length(n)  
6     x(i)=1;  
7 end
```

```
8 X=fft(x,-1);
```

---

#### Scilab code Exa 4.6 DFT computation

```
1 clc;
2 clear;
3 Wo=2*%pi/3;
4 n=-8:8;
5 for i=1:length(n)
6     x(i)=cos(Wo*n(i));
7 end
8 X=fft(x,-1);
```

---

#### Scilab code Exa 4.9 DFT computation

```
1 clc;
2 clear;
3 a=0.5;
4 b=0.25;
5 n=-10:10;
6 for i=1:length(n)
7     if n(i)<0 then
8         x(i)=0;
9         h(i)=0;
10    else
11        x(i)=b^n(i);
12        h(i)=a^n(i);
13    end
14 end
15 y=convol(x,h);
16 n1=-20:20;
17 plot2d3(n1,y)
```

---

#### Scilab code Exa 4.28 Circular convolution

```
1 //Ex 4.28
2 clc;
3 f1=[2,1,2,1];
4 f2=[1,2,3,4];
5 F1=fft(f1,-1);
6 F2=fft(f2,-1);
7 F=F1.*F2;
8 f=fft(F,1);
```

---

#### Scilab code Exa 4.29 Record Length

```
1 //Ex 4.29
2 clc;
3 fm=500;
4 fs=2*fm;
5 T=1/fs;
6 df=10;
7 N=2*fm/df;
8 rl=N*T;
9 disp(rl,'Record Length');
10 sl=0.05;
11 zp=rl-sl;
12 disp(zp,'Zero padding required');
```

---

#### Scilab code Exa 4.31 Sampling rate and DFT size

```
1 //Ex 4.31
```

```
2  clc;  
3  fm=10*103;  
4  df=100;  
5  N=2*fm/df;  
6  l=log2(N);  
7  l1=ceil(l);  
8  N1=2l1;  
9  fs=N1*df;  
10 disp(fs, 'Required Sampling Freq');
```

---

#### Scilab code Exa 4.32 DFT computation

```
1  //Ex 4.32  
2  clc;  
3  clear;  
4  f=[0 1 1 1 0 0];  
5  F=fft(f,-1);
```

---

# Chapter 5

## Sampling

Scilab code Exa 5.1 Minimum Sampling Frequency

```
1 //Ex 5.1
2 clc;
3 fmax=30;
4 fmin=20;
5 BW=fmax-fmin;
6 disp(BW, 'Bandwidth (kHz)');
7 mFs=2*BW;
8 disp(mFs, 'Minimum Sampling Frequency (kHz)');
```

---

Scilab code Exa 5.2.a Minimum Sampling Interval

```
1 //Ex 5.2 a
2 clc;
3 syms T;
4 disp('x(t)=1+cos(20*pi*t)');
5 w=20*pi;
6 f=w/(2*pi);
7 T=1/(2*f);
8 disp(T, 'minimum sampling interval');
```

---

**Scilab code Exa 5.2.b Minimum Sampling Interval**

```
1 //Ex 5.2b
2 clc;
3 disp('x(t)=(1/2*pi*t) [ sin(31*pi*t)-sin(29*pi*t) ] ');
4 w1=31*pi;
5 f1=w1/(2*pi);
6 w2=29*pi;
7 f2=w2/(2*pi);
8 if f1<f2 then
9     T=1/(2*f2);
10 else
11     T=1/(2*f1);
12 end
13 disp(T, 'minimun sampling interval');
```

---

**Scilab code Exa 5.6 Continuous time Frequency**

```
1 //Ex 5.6
2 clc;
3 ws=1000*pi;
4 w=ws/2;
5 disp(w, 'Cont. time Frequency w ( rad/seconed )');
```

---

**Scilab code Exa 5.9 Aliasing**

```
1 //Ex 5.9
2 clc;
3 disp('x(t)=cos(200*pi*t+theta)');
```

```
4 w=200*%pi;
5 f=w/(2*%pi);
6 nyquist_rate=2*f;
7 disp(nyquist_rate,'Minimum sampling frequency to
    avoid aliasing is');
```

---

**Scilab code Exa 5.12** Find the frequency

```
1 omega=%pi/4;
2 T=10^(-3)
3 w=omega/T;
4 disp(w)
```

---

**Scilab code Exa 5.14** Sampling period

```
1 [N,kerA]=linsolve(5*%pi/9,-2*%pi);
2 N1=floor(N);
3 disp(N1);
```

---

# Chapter 6

## Transformed Networks

Scilab code Exa 6.1 Current flowing in a network

```
1 //Ex 6.1
2 clc;
3 syms s;
4 v=10;
5 R=4;
6 L=2;
7 C=0.125;
8 V=laplace(v);
9 I=V/(R+L*s+(1/(C*s)));
10 i=ilaplace(I);
11 disp(i, 'i(t)=');
```

---

Scilab code Exa 6.2 voltage across inductor

```
1 //Ex 6.2
2 clc;
3 syms s;
4 i=4;
```



```

5 R=13/4;
6 L=1;
7 C=1/13;
8 I=laplace(i)+1;
9 Y=1/R+s/13+1/s;
10 V=I/Y;
11 v=ilaplace(V);
12 disp(v, 'v(t)=');

```

---

### Scilab code Exa 6.3 voltage across capacitor

```

1 //Ex 6.3
2 clc;
3 syms s;
4 i=5;
5 R=2;
6 L=1;
7 C=1/2;
8 Z=((R+L*s)*(1/(C*s)))/((R+L*s)+(1/(C*s)));
9 V=Z*i;
10 v=ilaplace(V);
11 disp(v, 'v(t)=');

```

---

### Scilab code Exa 6.9 Bode Plot

```

1 //Ex 6.9
2 //Obtain the Bode plot
3 clc;
4 s=poly(0, 's');
5 H=symlin('c', 8*(1+.1*s)/(s*(1+.5*s)*(1+.6*s/50+(s
    /50)^2)));

```

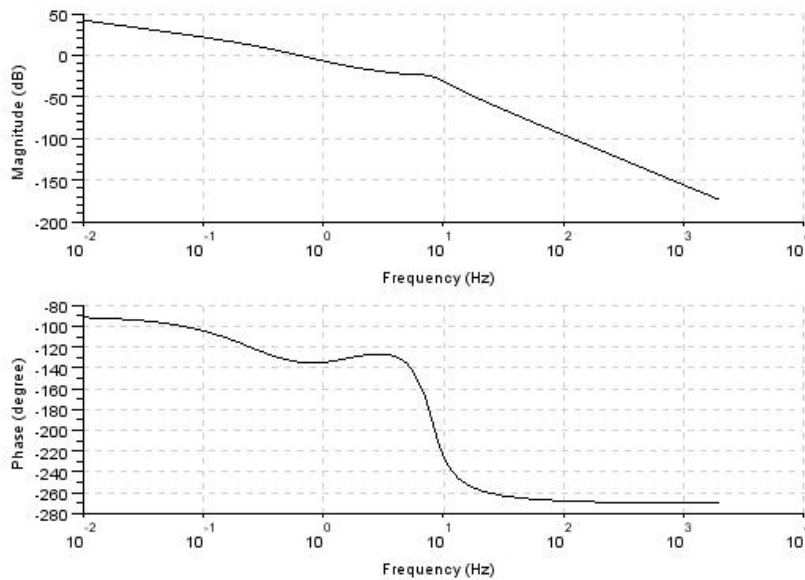


Figure 6.1: Bode Plot

6 `bode(H,0.01,2000);`

---

#### Scilab code Exa 6.10 Bode Plot

```

1 //Ex 6.10
2 //Obtain the Bode plot
3 clc;
4 H=syslin('c',10*(1+%s/2)/(%s*(1+%s/.1)*(1+%s/.5)*(1+
   %s/10)));
5 bode(H,0.01,100);

```

---

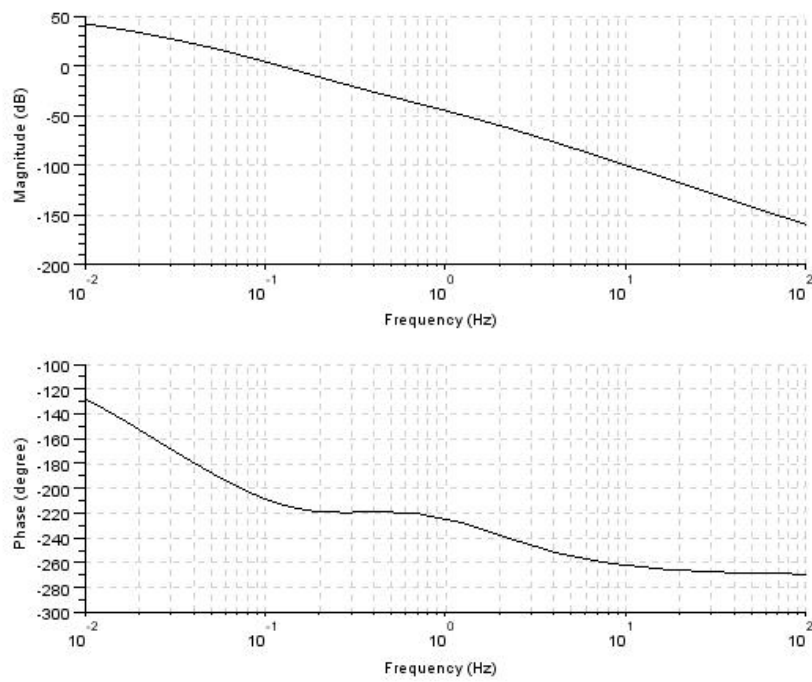


Figure 6.2: Bode Plot

# Chapter 7

## State Space Analysis

Scilab code Exa 7.1 State Space Representation

```
1 clc;  
2 close  
3 clear;  
4 s=%s;  
5 tf=syslin('c',((s+3)/(s^3+5*s^2+8*s+3)));  
6 ss=tf2ss(tf);  
7 disp(ss)
```

---

Scilab code Exa 7.2 State Space Representation

```
1 clc;  
2 close  
3 clear;  
4 s=%s;  
5 tf=syslin('c',((s+3)/((s+2)^2*(s+1))));  
6 ss=tf2ss(tf);  
7 disp(ss)
```

---

# Chapter 8

## Stability Analysis of LTI Systems

Scilab code Exa 8.1.a check for HURWITZ

```
1  clc;
2  // Define the polynomial
3  s=poly(0,"s");
4  p=10+9*s+4*s^2+s^3;
5  [C]=coeff(p);
6  l1=length(C);
7  x=0;
8  for i1=1:l1
9      a1=C(1,i1);
10     r1=real(a1);
11     if r1>0 then
12         x=x+1;
13     end
14 end
15 if(x==l1) then
16     [S]=roots(p);
17     disp(S,"Roots=");
18     l=length(S);
19     c=0;
```

```

20     for i=1:l
21         a=S(i,1);
22         r=real(a);
23         if r<0 then
24             c=c+1;
25         end
26     end
27     if(c==l) then
28         printf("Polynomial is Hurwitz");
29     else
30         printf("Polynomial is non-Hurwitz");
31     end
32 else
33     printf("Polynomial is non-Hurwitz");
34 end

```

---

#### Scilab code Exa 8.1.b check for HURWITZ

```

1     clc;
2     // Define the polynomial
3     s=poly(0,"s");
4     p=30+4*s+s^2+s^3;
5     [C]=coeff(p);
6     l1=length(C);
7     x=0;
8     for i1=1:l1
9         a1=C(1,i1);
10        r1=real(a1);
11        if r1>0
12            x=x+1;
13        end
14    end
15    if(x==l1) then
16        [S]=roots(p);
17        disp(S,"Roots");

```

```

18     l=length(S);
19     c=0;
20     for i=1:l
21         a=S(i,1);
22         r=real(a);
23         if r<0 then
24             c=c+1;
25         end
26     end
27     if(c==l) then
28         printf("Polynomial is Hurwitz");
29     else
30         printf("Polynomial is non-Hurwitz");
31     end
32 else
33     printf("Polynomial is non-Hurwitz");
34 end

```

---

### Scilab code Exa 8.2 checking stability

```

1     clc;
2     // Define the polynomial
3     s=poly(0,"s");
4     p=5+3*s+2*s^2+2*s^3+s^4+s^5;
5     // Calculate the routh of above polynomial
6     r=routh_t(p);
7     A=r(:,1);
8     c=0;
9     x=0;
10    for i=1:6
11        x=A(i,1);
12        if x<>0
13            c=c+1;
14        end
15    end

```

```

16     if(c>=1) then
17         printf("system is unstable");
18     else
19         printf("system is stable");
20     end

```

---

### Scilab code Exa 8.3 checking stability

```

1     clc;
2     // Define the polynomial
3     s=poly(0,"s");
4     p=2+2*s+5*s^2+4*s^3+4*s^4+2*s^5+s^6;
5     // Calculate the routh of above polynomial
6     r=routh_t(p);
7     S=roots(p);
8     disp(r,"Routh array=");
9     disp(S,"Roots=");
10    A=r(:,1);
11    c=0;
12    x=0;
13    for i=1:6
14        x=A(i,1);
15        if x<0
16            c=c+1;
17        end
18    end
19    if(c>=1) then
20        printf("system is unstable");
21    else
22        l=length(S);
23        c=0;
24        for i=1:l
25            a=S(i,1);
26            r=real(a);
27            if r<0 then

```



```

28             c=c+1;
29         end
30     end
31     if c==0 then
32         printf("system is stable");
33     else
34         printf("system is unstable");
35     end
36
37 end

```

---

#### Scilab code Exa 8.4 checking stability

```

1     clc;
2     // Define the polynomial
3     s=poly(0,"s");
4     p=1+2*s+5*s^2+5*s^3+2*s^4;
5     // Calculate the routh of above polynomial
6     r=routh_t(p);
7     disp(r,"Routh array=");
8     A=r(:,1);
9     c=0;
10    x1=0;
11    eps=0;
12    for i=1:5
13        x1=A(i,1);
14        if x1<0
15            c=c+1;
16        end
17    end
18    if(c>=1) then
19        printf("system is unstable");
20    else
21        printf("system is stable");
22    end

```

23      `x=roots(p);`

---

#### Scilab code Exa 8.5 checking stability

```
1      clc;
2      // Define the polynomial
3      s=poly(0,"s");
4      p=36+12*s+12*s^2+3*s^3+s^4;
5      // Calculate the routh of above polynomial
6      r=routh_t(p);
7      disp(r,"Routh array=");
8      A=r(:,1);
9      c=0;
10     x=0;
11     for i=1:5
12         x=A(i,1);
13         if x<0
14             c=c+1;
15             end
16     end
17     if(c>=1) then
18         printf("system is unstable");
19         else
20             printf("system is stable");
21         end
22     x=roots(p);
```

---

#### Scilab code Exa 8.9 checking stability

```
1      clc;
2      // Define the polynomial
3      z=poly(0,"z");
4      p=5+3*z+2*z^2+2*z^3+z^4+z^5;
```

```
5 // Calculate the routh of above polynomial
6 r=routh_t(p);
```

---

### Scilab code Exa 8.11 Stability of discrete time system

```
1 clc;
2 // Define the polynomial
3 //z=poly(0,'z');
4 //z^3 - 0.3*z^2 + .1*z - .1
5 //Put z=(1+r)/(1-r);
6 //Bilinear transformation
7 s=poly(0,'s');
8 p=.7+2.9*s+2.9*s^2+1.7*s^3;
9 // Calculate the routh of above polynomial
10 r=routh_t(p);
11 disp(r,"Routh array=");
12 A=r(:,1);
13     c=0;
14     x=0;
15     for i=1:3
16         x=A(i,1);
17         if x<0
18             c=c+1;
19         end
20     end
21     if(c>=1) then
22         printf("system is unstable");
23     else
24         l=length(A);
25         c=0;
26         for i=1:l
27             a=A(i,1);
28             r=real(a);
29             if r<0 then
30                 c=c+1;
```

```
31         end
32     end
33     if c==0 then
34         printf("system is stable");
35     else
36         printf("system is unstable");
37     end
38
39 end
```

---

# Chapter 9

## Analog and Digital Filter Design

Scilab code Exa 9.1 BPF

```
1 //Ex 9.1
2 //Band pass Filter Design
3 //a
4 clc;
5 w1=10*103;
6 w2=15*103;
7 wot=sqrt(w1*w2);
8 wbt=w2-w1;
9 s=%s;
10 w=poly(0, 'w');
11 wt=poly(0, 'wt');
12 wx=(wt2-wot2)/(wbt*wt);
13 disp(w);
14 //b
15 //Let n=2
16 hb2=1/(s2+sqrt(2)*s+1);
17 hb21=horner(hb2, (%i*w));
18 disp(hb21);
19 hb22=horner(hb21, wx);
```

```
20 disp(hb22);
```

---

### Scilab code Exa 9.2 Band Stop Filter

```
1 //Ex 9.2
2 //Band Stop Filter Design
3 //a
4 clc;
5 w1=1200;
6 w2=2000;
7 s=%s;
8 w=poly(0, 'w');
9 St=poly(0, 'St');
10 wc=1; //For normalised Prototype
11 wd1t=poly(0, 'wd1t');
12 wt1=2500;
13 wx1=(wt1*(w2-w1)*wd1t)/(-wt1^2+w2*w1);
14 wt2=400;
15 wx2=(wt2*(w2-w1)*wd1t)/(-wt2^2+w2*w1);
16 disp(w);
17 wx=wx1; // required attenuation to less
18 // than -3dB for wx > -0.5195wd1t
19 wd1t=wc/0.5195;
20 //b
21 //Let n=4
22 hb2=1/((s^2+.7654*s+1).*(s^2+1.8478*s+1));
23 hb21=horner(hb2, (%i*w));
24 disp(hb21);
25 hb22=horner(hb21, wx);
26 disp(hb22);
27 hb23=horner(hb22, -%i*St);
28 disp(hb23);
```

---

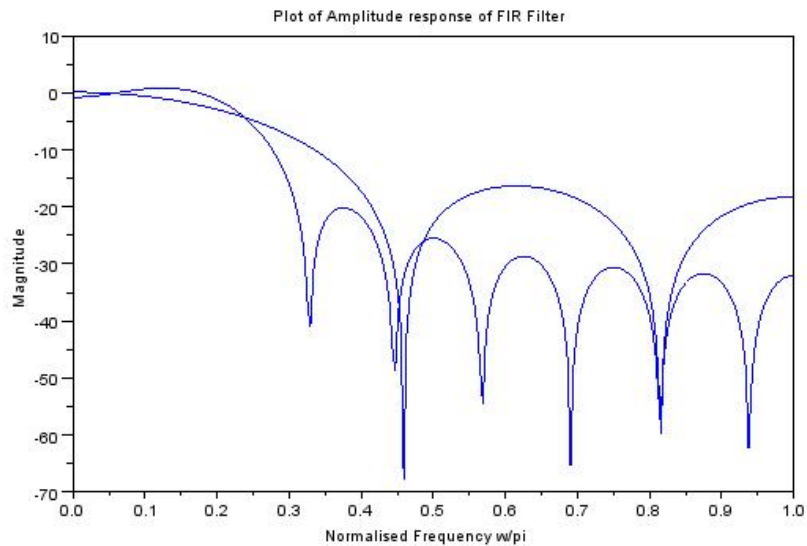


Figure 9.1: FIR Filter

### Scilab code Exa 9.3 FIR Filter

```

1 //Ex 9.3
2 //FIR Filter
3 close;
4 clc;
5 //Fourier series method
6 //for N=5
7 N=5;
8 U=3;
9 for n=-2+U:1:2+U
10 if n==3
11 hd(n)=0.25;
12 else
13 hd(n)=sin(%pi*(n-U)/4)/(%pi*(n-U));

```

```

14 end
15 end
16 [hzm ,fr ]= frmag (hd ,256) ;
17 hzm_dB = 20* log10 (hzm)./ max ( hzm );
18 plot (2*fr , hzm_dB )
19
20 //for N=15
21 N=15;
22 U=8;
23 for n=-7+U:1:7+U
24     if n==8
25         hd(n)=0.25;
26     else
27         hd(n)=sin(%pi*(n-U)/4)/(%pi*(n-U));
28     end
29 end
30 [hzm ,fr ]= frmag (hd ,256) ;
31 hzm_dB = 20* log10 (hzm)./ max ( hzm );
32 plot (2*fr , hzm_dB )
33 xlabel('Normalised Frequency w/pi')
34 ylabel('Magnitude')
35 title('Plot of Amplitude response of FIR Filter')

```

---

### Scilab code Exa 9.5 Low Pass Filter

```

1 //Ex 9.5
2 clc;
3 N=21;
4 U=11;
5 for n=-10+U:1:10+U
6     if n==11
7         hd(n)=0.25;
8     else

```



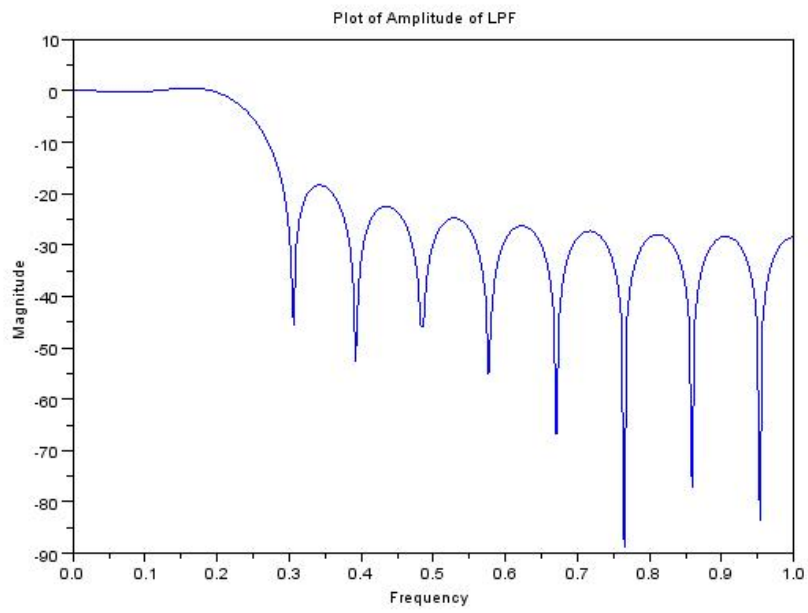


Figure 9.2: Low Pass Filter

```

9  hd(n)=(sin(%pi*(n-U)/4))/(%pi*(n-U));
10 end
11 end
12 [hzm ,fr ]= frmag (hd ,256) ;
13 hzm_dB = 20* log10 (hzm)./ max ( hzm );
14 plot (2*fr , hzm_dB );
15 xlabel ( 'Frequency' );
16 ylabel ( 'Magnitude' );
17 title('Plot of Amplitude of LPF')

```

---

#### Scilab code Exa 9.7 2nd order Digital Butterworth Filter

```

1 //Ex 9.7
2 clc;
3 s=%s;
4 T=10^(-4);
5 wdc=2*%pi*10^3;
6 wac=2/T*tan(wdc*T/2);
7 HS=1/(s^2+sqrt(2)*s+1) // Transfer Function for N=1
8 HS1=horner(HS ,s/wac);
9 disp(HS1 , 'Normalized Transfer Function , H(s) =');
10 z=%z;
11 HZ=horner(HS1 ,(2/T)*(z-1)/(z+1));
12 disp(HZ , 'H(z) =');

```

---

# Chapter 10

## MATLAB Tools for Analysis of Signals and systems

Scilab code Exa 10.1 Solving Linear Equation

```
1 clc;  
2 A=[2 1; 1 2]  
3 B=[1; 0]  
4 A=inv(A);  
5 X=A*B;
```

---

Scilab code Exa 10.3 Find the step and impulse response

```
1 clc;  
2 s=poly(0, 's');  
3 x=syslin('c', %s/(%s+2));
```

---

Scilab code Exa 10.9 Find the impulse response

```
1 //Ex 10.9
2 //Impulse response
3 num=[1 3];
4 den=[1 3 2];
5 n=0:1:20;
6 x=[1 zeros(1,20)];
7 y=filter(num,den,x);
8 plot(n,y);
```

---

Scilab code Exa 10.12 Find convolution and plot the result

```
1 //Ex 1012
2 //Convolution of two sequences
3 //[1 1 1 1 1]
4 //[1 2 3]
5 clc;
6 n1=0:1:4;
7 n2=0:1:2;
8 x=[1 1 1 1 1]
9 h=[1 2 3]
10 y=convol(x,h);
11 l=length(y);
12 n3=0:1:l-1;
13 figure
14 title('Sequence x')
15 plot2d3(n1,x);
16 figure
17 title('Sequence h')
18 plot2d3(n2,h);
19 figure
20 title('Sequence y')
21 plot2d3(n3,y);
```

---

Scilab code Exa 10.13 Find the step response

```
1 //Ex 1013
2 //Convolution of two sequences
3 //To plot the step response
4 // [.25 .25 .25 .25]
5 // [.25 -.25 .25 -.25]
6 clc;
7 n1=0:1:8;
8 n2=0:1:8;
9 h1=[0 0 0 .25 .25 .25 .25 0 0]
10 h2=[0 0 0 .25 -.25 .25 -.25 0 0]
11 y=convol(h1,h2);
12 l=length(y);
13 n3=0:1:l-1;
14 figure
15 title('Sequence h1')
16 plot2d3(n1,h1);
17 figure
18 title('Seequence h2')
19 plot2d3(n2,h2);
20 figure
21 title('Sequence y')
22 plot2d3(n3,y);
```

---