Scilab Textbook Companion for Signals And Systems by I. J. Nagrath, S. N. Sharan And R. Ranjan[1]

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

May 16, 2016

[1]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

List of Figures

Chapter 1

Introduction to Signal and Systems

Scilab code Exa 1.6 Shifting Scaling Time Reversal

```
//shifting and scaling
//Ex 1.6
clear;
clc;
close;
t = 0:1/100:2;
for i = 1:length(t)
    if t(i)<1 then
      x(i) = (2)*t(i) ;
        else
        x(i)= 2;
    end
end
for i = length(t)+1:2*length(t)
x(i) = 0;
end
t1=0:1/100:4.01;
figure
subplot(3,1,1),plot2d(t1,x)
```

```
a=gca();
t2=t1-1;
subplot(3,1,2),plot2d(t2,x)
xtitle('x(t+1)')
a.y_location='origin'
a=gca();
subplot(3,1,3),plot2d(t1/1.5,x)
xtitle('x(1.5t)')
figure
a=gca();
subplot(2,1,1),plot2d((t2/1.5),x)
xtitle('x(1.5t+1)')
a.y_location='origin'
a=gca();
t3=-3:1/100:1.01;
subplot(2,1,2),plot2d(t3,x($:-1:1))
xtitle('x(-t+1)')
a.y_location = 'origin';
```

Scilab code Exa 1.9 Check for Periodicity

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

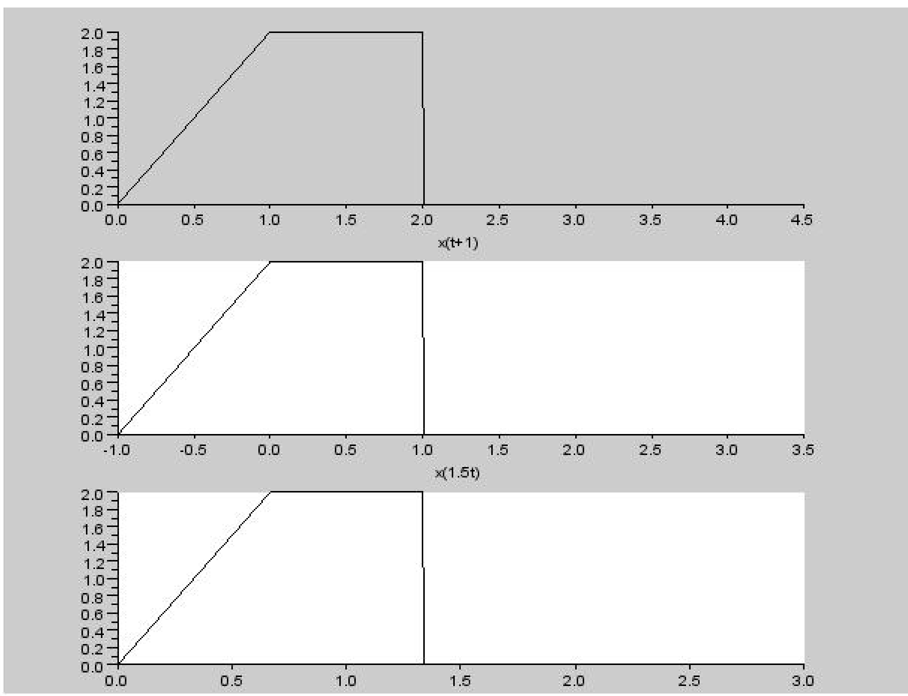

Figure 1.1: Shifting Scaling Time Reversal

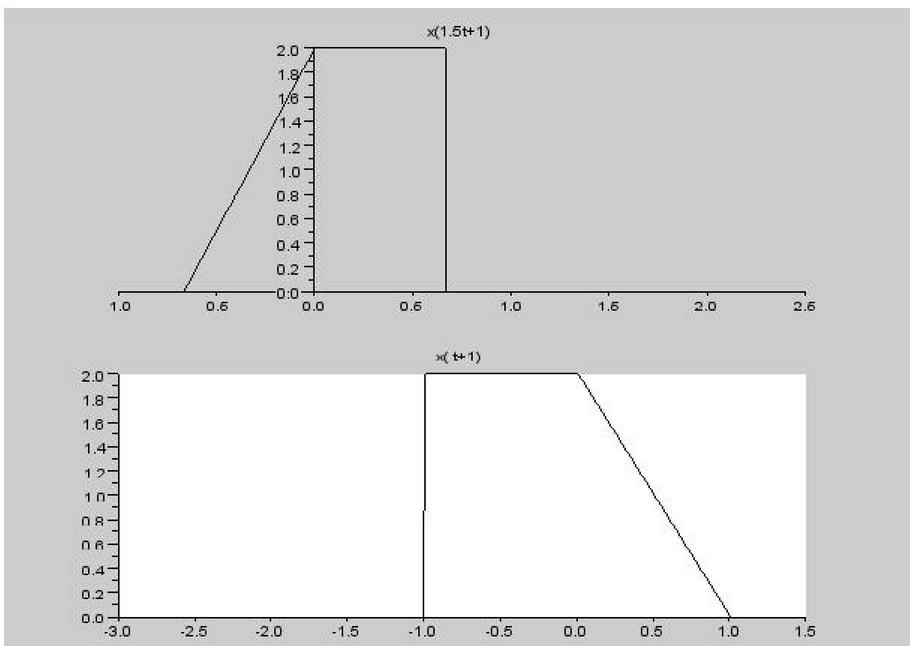

Figure 1.2: Shifting Scaling Time Reversal

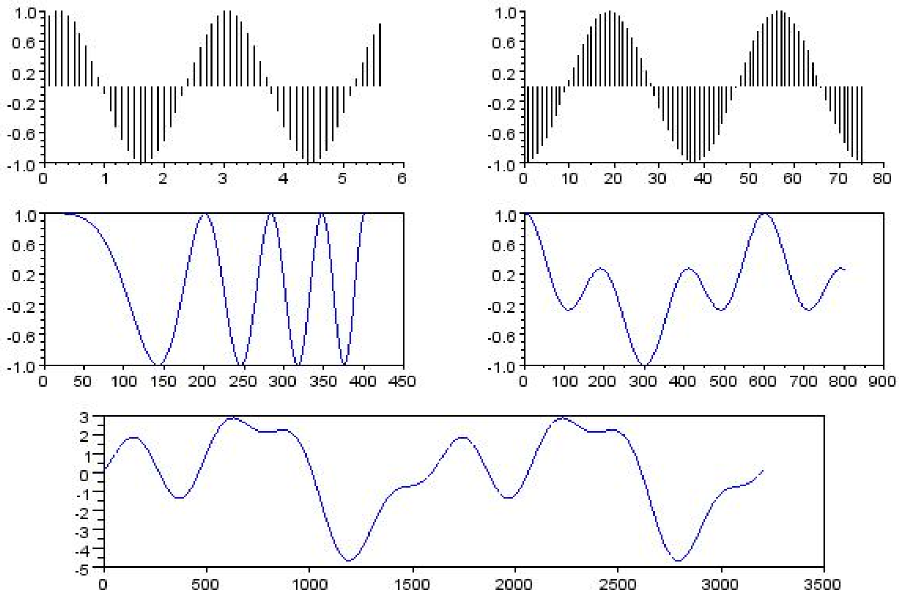

Figure 1.3: Check for Periodicity

```
x=sin((5*%pi*n/7)+1); //sinusoid function
subplot(3,2,1),plot2d3(n,x) //ploting the sinusoid showing it as periodic 2pik=5piN/7
disp('the plot shows the above signal is periodic');

k=1;
N=2*%pi*k*6;
z=2*N;
n=0:1:z //Defining discrete time
x=cos((n/6)-%pi);
subplot(3,2,2),plot2d3(n,x);//the plot shows the above signal is periodic
disp('the plot shows the above signal is periodic');

k=1;
N=sqrt(2*k*2);
z=2*N;
n=0:1/100:z //Defining discrete time
x=cos(((n.^2)*%pi)/2);
```

```
subplot(3,2,3),plot(x);//the plot shows that the above signal is not periodic
disp('the plot shows the above signal is not periodic');

k=1;
N=2*k;
z=2*N;
n=0:1/100:2*z //Defining discrete time
x=cos(n*%pi/3).*cos(2*%pi*n/3);
subplot(3,2,4),plot(x);//the plot shows that the above signal is periodic
disp('the plot shows the above signal is periodic');

k=1;
N=2*k*8;
z=2*N;
n=0:1/100:z //Defining discrete time
x=2*cos(n*%pi/4)+sin((n*%pi/2)-(%pi/3))-2*cos((n*%pi/8)+(%pi/3));
subplot(3,1,3),plot(x);//the plot shows that the above signal is periodic
disp('the plot shows the above signal is periodic');
```

Scilab code Exa 1.11 Check for Periodicity

```
//Ex 1.11
//Check for peridicity
clc;
T=2*%pi/8;
z=2*T;
t=0:0.001:z
x=%i*exp(%i*8*t);
```

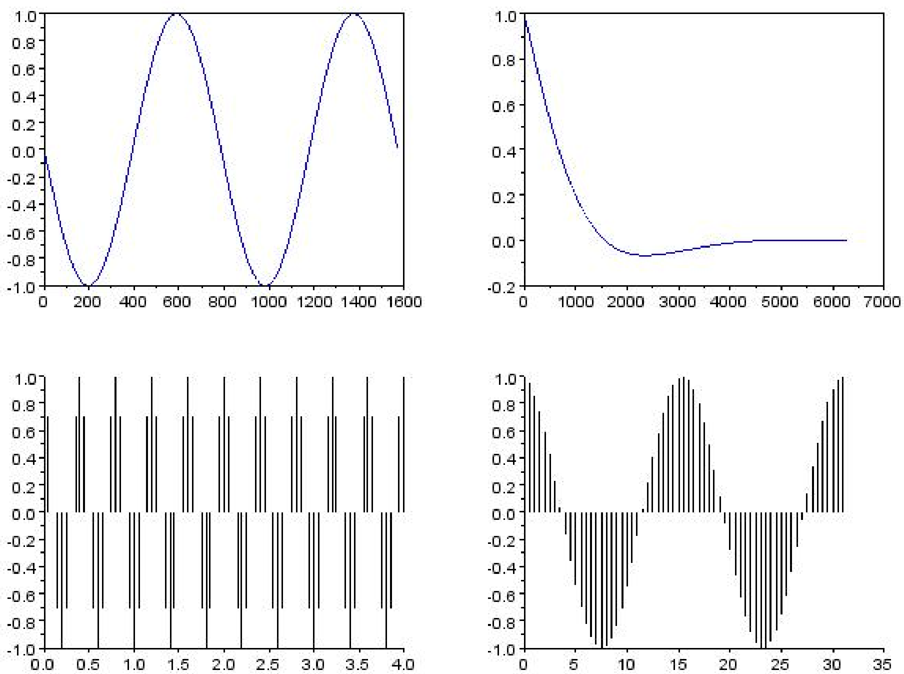

Figure 1.4: Check for Periodicity

```
subplot(2,2,1),plot(x);//the plot shows that the above signal is periodic

T=2*%pi/(-1+%i);
z=2*T;
disp('T cannot be complex so non periodic');
t=0:-0.001:z
x=exp((-1+%i)*abs(t));
subplot(2,2,2),plot(x);//the plot shows that the above signal is not periodic

N=2*%pi/%pi;
z=2*N;
n=0:0.05:z
x=exp(%i*5*%pi*n);//exp(i*(4*pi+pi)*n)=exp(i*pi*n)
subplot(2,2,3),plot2d3(n,x);//the plot shows that the above signal is periodic

k=1;
N=2*%pi*5/2;
z=2*N;
n=0:0.5:z
x=exp((%i*2/5)*(n+(1/3)));
subplot(2,2,4),plot2d3(n,x);//the plot shows that the above signal is periodic
```

Scilab code Exa 1.13 Check for Periodicity

```
//Ex 1.13
//Check for periodicity
clc;
T=2*%pi/6;
t=0:0.001:T*2
```

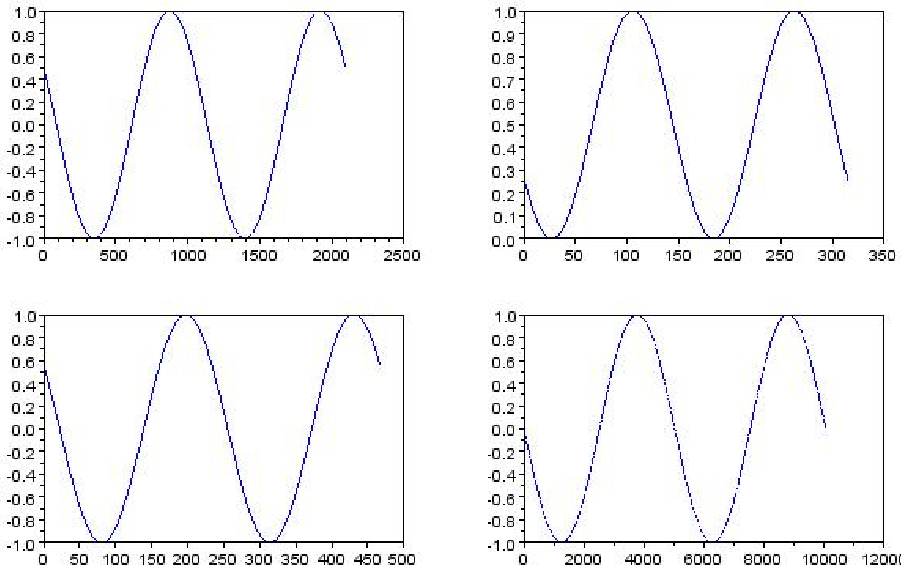

Figure 1.5: Check for Periodicity

```
x=cos((6*t)+%pi/3);
subplot(3,2,1),plot(x);
disp('the plot shows that the above signal is
   periodic');

T=2*%pi/(%i*%pi);
t=0:0.001:T*2
x=exp(%i*(%pi*abs(t-1)));//exp(%i*(%pi*t-1))=exp(%i*
   %pi*t)/exp(%i)
//since the period is a complex no so non periodic
disp('T cannot be complex so non periodic T=2*%pi/(
   %i*%pi)');

//pi=22/7
T=2*%pi/4;//calc the fundamental period
z=2*T;
t=0:1/100:z
x=(cos(2*t+%pi/3))^2; //sinusoid function
subplot(3,2,2),plot(x)
```

```
disp('the plot shows that the above signal is periodic');

k=1;
N=2*k*7/6;
z=2*N;
n=0:1/100:z
x=cos((6*%pi*n/7)+1);
subplot(3,2,3),plot(x);//the plot shows that the above signal is periodic
disp('the plot shows that the above signal is periodic');

k=1;
N=2*%pi*k*8;
z=2*N;
n=0:1/100:z
x=sin((n/8)-%pi);
subplot(3,2,4),plot(x);//the plot shows that the above signal is periodic
disp('the plot shows that the above signal is periodic');

k=1;
N=2*k*12;//2*cos(n*%pi/4).*cos(n*%pi/3)=cos(7*n*%pi/12)-cos(n*%pi/12)
z=2*N;
n=0:1/100:z
x=2*cos(n*%pi/4).*cos(n*%pi/3);
subplot(3,1,3),plot(x);//the plot shows that the above signal is periodic
disp('the plot shows the above signal is periodic');
```

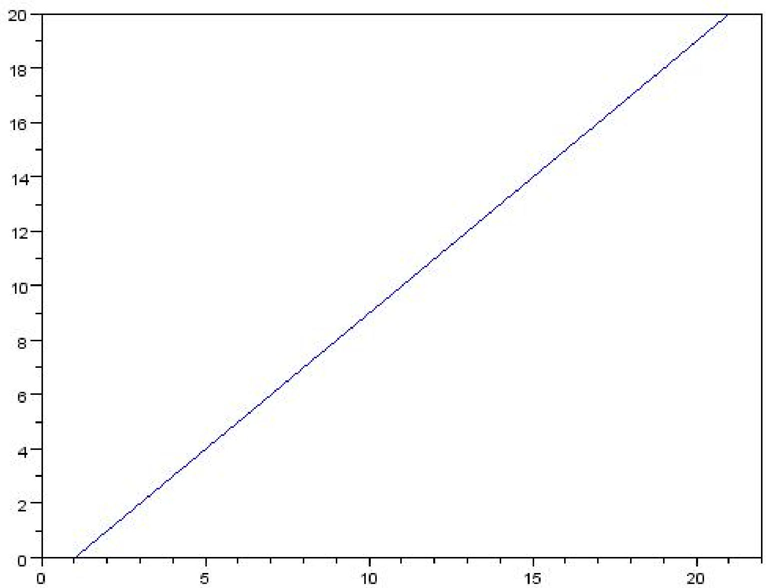

Figure 1.6: Integral of Unit step function

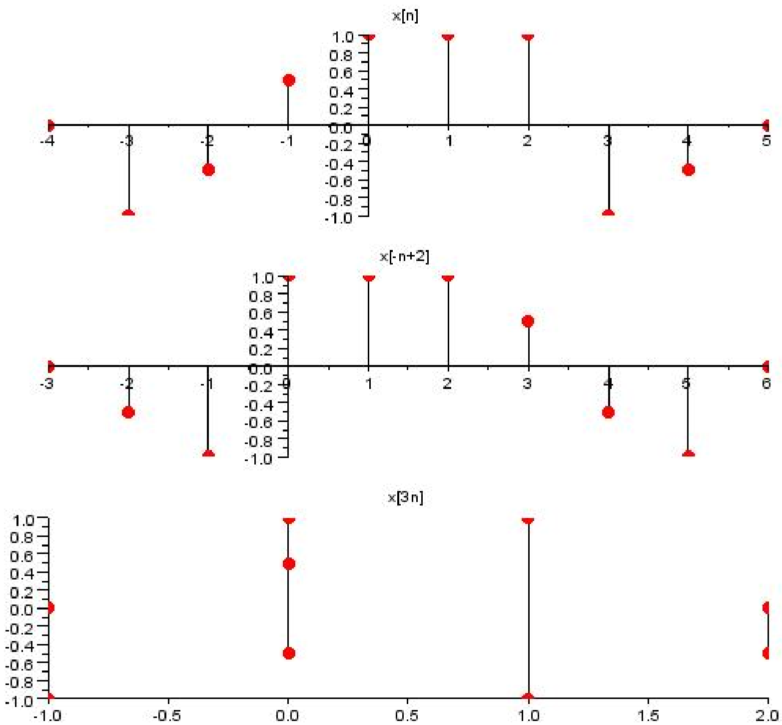

Figure 1.7: Shifting Time Reversal of discrete time signals

Scilab code Exa 1.30 Integral of Unit step function

```
//Ex 1.30
//Integral of unit step function
clc;
x0=0;//lower bound
x1=0:20;//upper bound vector
X=integrate('1','x',x0,x1);
//integration of unt step seq
//resulting in ramp seq
plot(X)
```

Scilab code Exa 1.37 Shifting Time Reversal of discrete time signals

```
//Ex 1.37
//shifting and scaling discrete signals
clear ;
clc;
close;
t=-4:5;
x=[0 -1 -.5 .5 1 1 1 -1 -.5 0]
a=gca();
subplot(3,1,1),plot2d3(t,x)
subplot(3,1,1),plot(t,x,'r.')
xtitle('x[n]')
a.x_location='origin';
a.y_location='origin';

t1=-5:4;
t2=t1+2;
a=gca();
subplot(3,1,2),plot2d3(t2,x($:-1:1))
subplot(3,1,2),plot(t2,x($:-1:1),'r.')
xtitle('x[-n+2]')

a=gca()
subplot(3,1,3),plot2d3(ceil(t/3),x)
subplot(3,1,3),plot(ceil(t/3),x,'r.')
xtitle('x[3n]')
a.x_location='origin';
a.y_location='origin';
```

Chapter 2

Analysis of LTI continous time system

Scilab code Exa 2.2 Convolution

```
//Ex 2.2
clc;
h=ones(1,10);
N2=0:length(h)-1;
a=0.5;//constant a>0
for t=1:10
    x(t)=exp(-a*(t-1));
end
N1=0:length(x)-1;
y=convol(x,h)-1;
N=0:length(x)+length(h)-2;

subplot(3,1,1),plot2d(N2,h)
xtitle('Impulse Response','t','h(t)');

subplot(3,1,2),plot2d(N1,x)
xtitle('Input Response','t','x(t)');
```

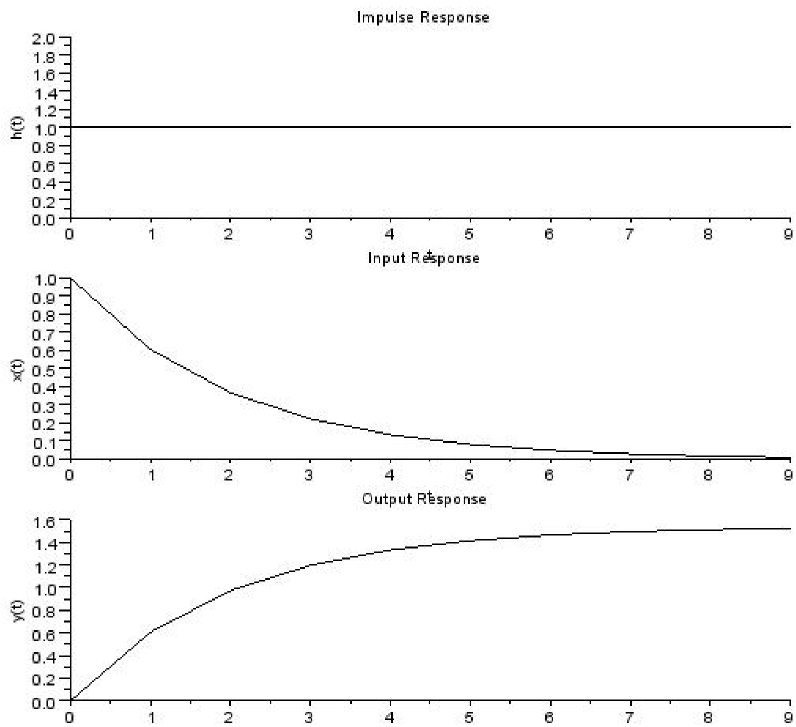

Figure 2.1: Convolution

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

Scilab code Exa 2.11.a Fourier transform of impulse function

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

Scilab code Exa 2.11.b Fourier transform of exponential function

```
clc;
A=1;
Dt=0.005;
t=-4.5:Dt:4.5;
xt=exp(-A*abs (t));
Wmax=2*%pi*1;
K=4;
k=0:(K/1000):K;
W=k*Wmax/K;
XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
XW = real (XW);
W=[-mtlb_fliplr(W),W(2:1001)];
XW=[mtlb_fliplr(XW),XW(2:1001)];
subplot(1,1,1)
subplot(2,1,1);
a=gca();
a.y_location="origin";
```

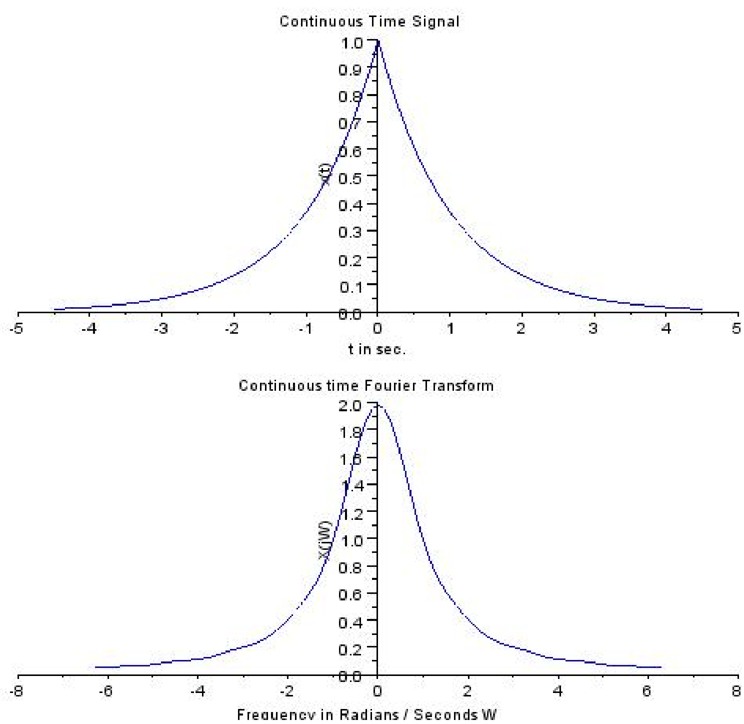

Figure 2.2: Fourier transform of exponential function

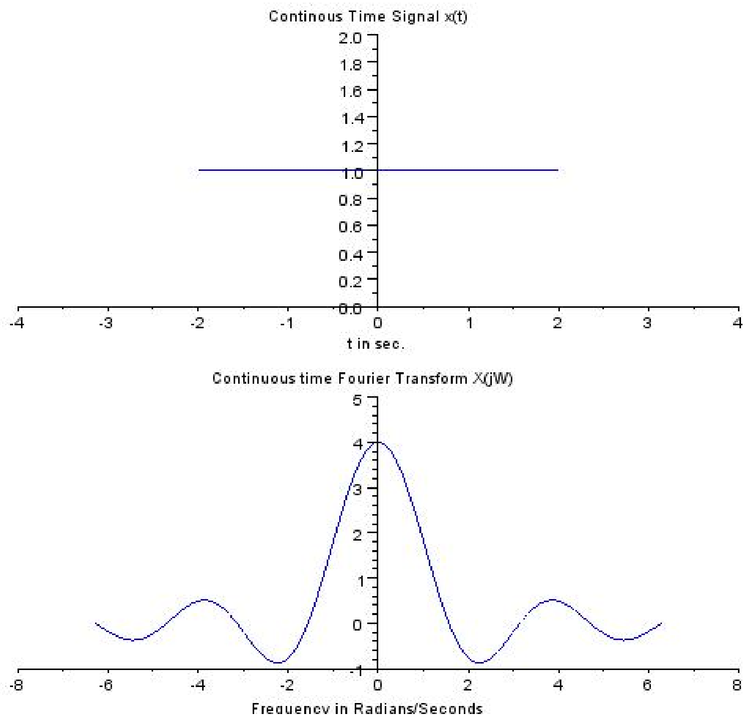

Figure 2.3: Fourier transform of Gate function

```
plot(t,xt);
xlabel(' t in sec. ');
ylabel(' x(t) ')
title(' Continuous Time Signal ')
subplot(2,1,2);
a=gca();
a.y_location="origin";
plot(W,XW);
xlabel(' Frequency in Radians / Seconds W' );
ylabel( 'X(jW) ' )
title(' Continuous time Fourier Transform ' )
```

Scilab code Exa 2.12 Fourier transform of Gate function

```
//Ex 2.12
clc;
clear;
A=1;
Dt=0.005;
T1=4;
t=-T1/2:Dt:T1/2;
for i=1:length(t)
xt(i)=A;
end
Wmax=2*%pi*1;
K=4;
k=0:(K/1000):K;
W=k*Wmax/K;
xt=xt';
XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
XW_Mag=real(XW);
W=[-mtlb_fliplr(W),W(2:1001)];
XW_Mag=[mtlb_fliplr(XW_Mag),XW_Mag(2:1001)];
subplot(2,1,1);
a=gca();
a.data_bounds=[-4,0;4,2];
a.y_location="origin";
plot(t,xt);
xlabel('t in sec.');
title('Continous Time Signal x(t)');
subplot(2,1,2);
a=gca();
a.y_location="origin";
plot(W,XW_Mag);
xlabel('Frequency in Radians/Seconds');
title('Continuous time Fourier Transform X(jW)');
```

Scilab code Exa 2.14 Fourier transform one sided exponential function

```
//Ex 2.14
//Fourier Transform
clc ;
clear;
A=1;
Dt=0.005;
t=0:Dt:10;
xt=exp(-A*t);
Wmax=2*%pi*1;
K=4;
k=0:(K/1000):K;
W=k*Wmax/K;
XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
XW_Mag=abs(XW);
W=[-mtlb_fliplr(W),W(2:1001)];
XW_Mag=[mtlb_fliplr(XW_Mag),XW_Mag(2:1001)];
[XW_Phase,db]=phasemag(XW);
XW_Phase=[-mtlb_fliplr(XW_Phase),XW_Phase(2:1001)];
figure
a=gca();
a.y_location="origin";
plot(t,xt);
xlabel('t in sec.');
ylabel('x(t)');
title('Continuous Time Signal');
figure
subplot(2,1,1);
a=gca();
a.y_location="origin";
plot(W,XW_Mag);
xlabel('Frequency in Radians/Seconds>W');
ylabel('abs(X(jW))');
title('Magnitude Response (CTFT)');
subplot(2,1,2);
a=gca();
a.y_location="origin";
```

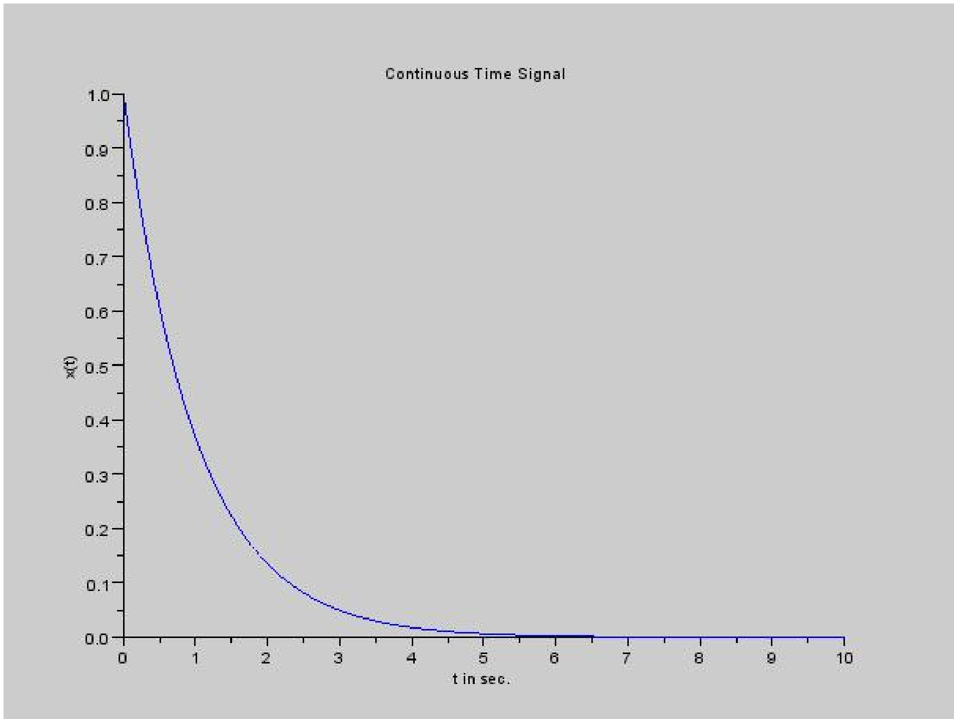

Figure 2.4: Fourier transform one sided exponential function

```
a.x_location="origin";
plot(W,XW_Phase*%pi/180) ;
xlabel(' Frequency in Radians/Seconds                >W
    ');
ylabel('<X(jW)')
title('Phase Response (CTFT) in Radians');
```

Scilab code Exa 2.20.a Laplace transform of unit impulse

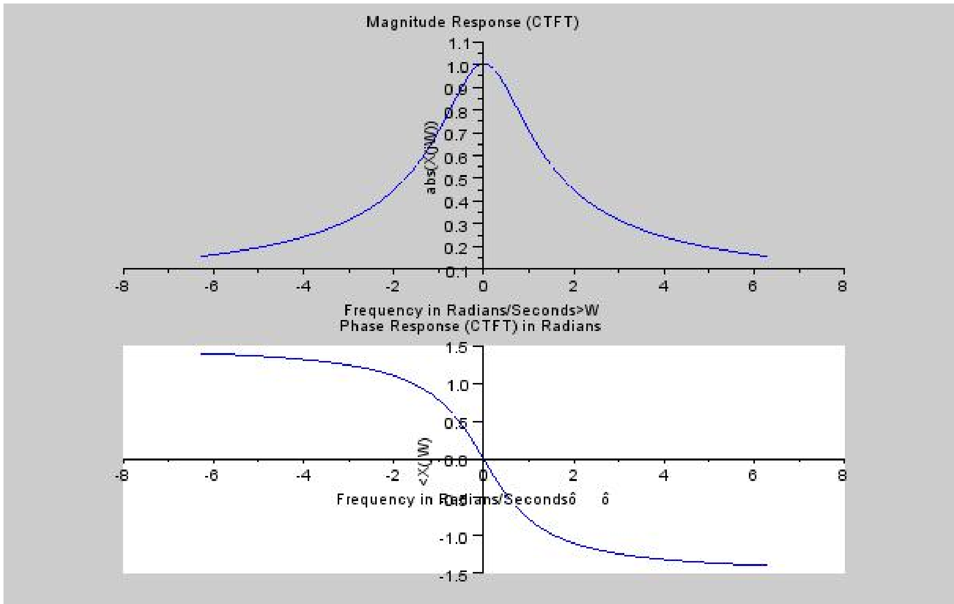

Figure 2.5: Fourier transform one sided exponential function

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

Scilab code Exa 2.23 Laplace Transform

```
syms a t;
x=exp(-a*t);
y=diff(x,t)
X=laplace(y);
```

Scilab code Exa 2.26.a Inv Laplace

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

Scilab code Exa 2.26.b Inv Laplace

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

Scilab code Exa 2.26.c Inv Laplace

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

Scilab code Exa 2.28 Laplace Transform

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

Scilab code Exa 2.33 Plot the spectrum

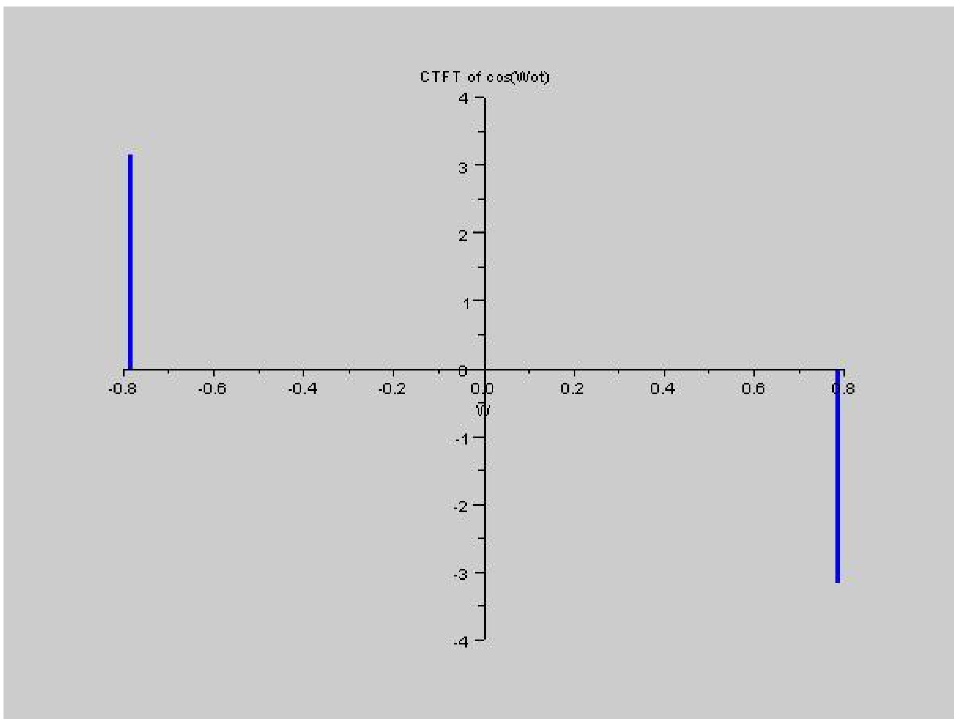

Figure 2.6: Plot the spectrum

```
//Ex 2.33
clc;
clear;
T1=2;
T=4*T1;
Wo=2*%pi/T;
W=[-Wo,0,Wo];
ak=(2*%pi*Wo*T1/%pi)/sqrt(-1);
XW=[-ak,0,ak];
ak1=-(2*%pi*Wo*T1/%pi);
XW1=[-ak1,0,ak1];
figure
a=gca();
a.y_location="origin";
a.x_location="origin";
plot2d3('gnn',W,XW1,2);
poly1=a.children(1).children(1);
poly1.thickness=3;
xlabel('W');
title('CTFT of cos(Wot)');
```

Scilab code Exa 2.39 Convolution of two signals

```
//Ex 2.39
clc;
clear;
A=1;
Dt=0.005;
T1=1;
t=-T1:Dt:T1;
for i=1:length(t)
xt(i)=A;
end
```

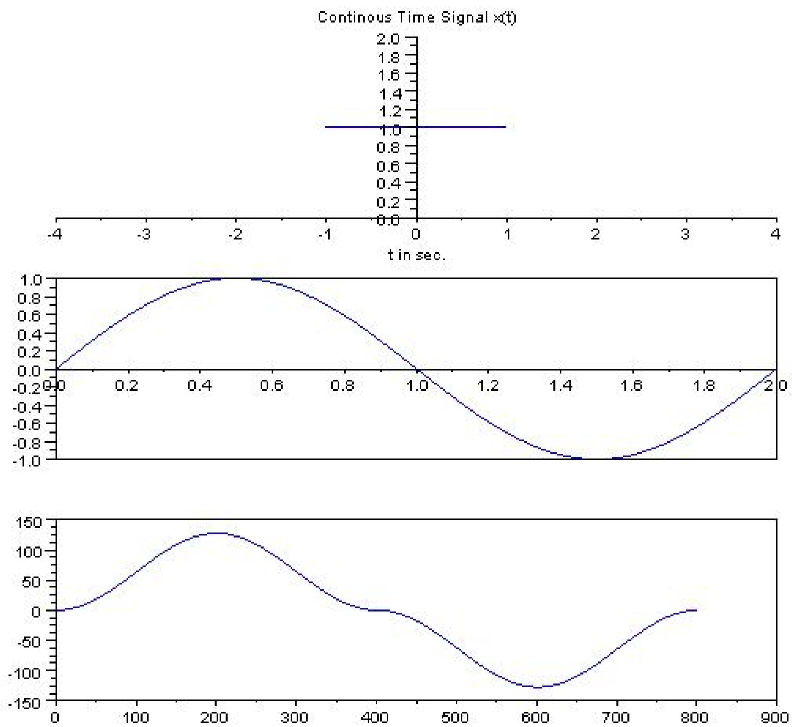

Figure 2.7: Convolution of two signals

```
xt=xt';
t1=0:0.005:2;
for j=1:length(t1)
  x1(j)=sin(%pi*t1(j));
end
subplot(3,1,1);
a=gca();
a.data_bounds=[-4,0;4,2];
a.y_location="origin";
plot(t,xt);
xlabel('t in sec.');
title('Continous Time Signal x(t)');
subplot(3,1,2);
a=gca();
plot(t1,x1);
y=convol(x1,xt);
subplot(3,1,3);
a.y_location="origin";
a.x_location="origin";
plot(y);
```

Chapter 3

Analysis of LTI Discrete time system

Scilab code Exa 3.1 Convolution sum method

```
clc;
//k=2;
r=1;
k=0:1:2;
h=0.5^(k);
l1=convol(h,r);
disp(l1);
//k=3;
r=1;
k=0:1:3;
h=0.5^(k);
l2=convol(h,r);
disp(l2);
```

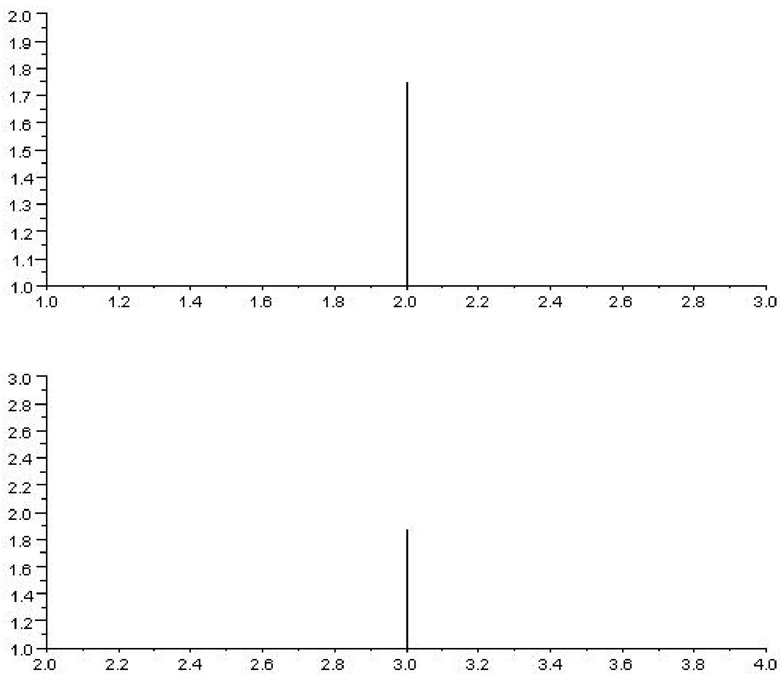

Figure 3.1: Graphical Convolution

Scilab code Exa 3.2 Graphical Convolution

```
clc;
//k=2;
r=1;
k=0:1:2;
h=0.5^(k);
l1=convol(h,r);
s1=sum(l1);
disp(s1);
n=2;
subplot(2,1,1),plot2d3(n,s1);
//k=3;
r=1;
k=0:1:3;
h=0.5^(k);
l2=convol(h,r);
s2=sum(l2);
disp(s2);
m=3;
subplot(2,1,2),plot2d3(m,s2);
```

Scilab code Exa 3.17.b Z Transform

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

Scilab code Exa 3.17.c Z Transform

```
//Ex 3.17c
clc;
syms z n B k;
x=exp(B*k);
X=symsum(x*z^(-n),n,0,%inf);
disp(X);
```

Scilab code Exa 3.17a Z Transform

```
//Ex 3.17a
clc;
syms z n;
x=1;
X=symsum(x*z^(-n),n,0,0);
disp(X);
```

Scilab code Exa 3.19 Final Value Theorem

```
//Ex 3.19
clc;
syms z;
c=(z-1)/(1+0.5*z^(-1))+(z^(-1)*(z-1))/((1+0.5*z^(-1)
    )*(1-z^(-1)));
l=limit(c,z,1);
disp(l);
```

Scilab code Exa 3.21 Inverse Ztransform

```
//Ex 3.21
clc;
```

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

Scilab code Exa 3.22 Z inverse

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

Chapter 4

Discrete Fourier Transform And Fast Fourier Transform

Scilab code Exa 4.4 DFT computation

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

Scilab code Exa 4.5 DFT computation

```
clc;
clear;
N1=8;
n=-N1:N1;
for i=1:length(n)
    x(i)=1;
end
```

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

Scilab code Exa 4.6 DFT computation

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

Scilab code Exa 4.9 DFT computation

```
clc;
clear;
a=0.5;
b=0.25;
n=-10:10;
for i=1:length(n)
    if n(i)<0 then
        x(i)=0;
        h(i)=0;
    else
        x(i)=b^n(i);
        h(i)=a^n(i);
    end
end
y=convol(x,h);
n1=-20:20;
plot2d3(n1,y)
```

Scilab code Exa 4.28 Circular convolution

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

Scilab code Exa 4.29 Record Length

```
//Ex 4.29
clc;
fm=500;
fs=2*fm;
T=1/fs;
df=10;
N=2*fm/df;
rl=N*T;
disp(rl,'Record Length');
sl=0.05;
zp=rl-sl;
disp(zp,'Zero padding required');
```

Scilab code Exa 4.31 Sampling rate and DFT size

```
//Ex 4.31
```

```
clc;
fm=10*10^3;
df=100;
N=2*fm/df;
l=log2(N);
l1=ceil(l);
N1=2^l1;
fs=N1*df;
disp(fs,'Required Sampling Freq');
```

Scilab code Exa 4.32 DFT computation

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

Chapter 5

Sampling

Scilab code Exa 5.1 Minimum Sampling Frequency

```
//Ex 5.1
clc;
fmax=30;
fmin=20;
BW=fmax-fmin;
disp(BW,'Bandwidth (kHz)');
mFs=2*BW;
disp(mFs,'Minimum Sampling Frequency (kHz)');
```

Scilab code Exa 5.2.a Minimum Sampling Interval

```
//Ex 5.2a
clc;
syms T;
disp('x(t)=1+cos(20%pi*t)');
w=20*%pi;
f=w/(2*%pi);
T=1/(2*f);
disp(T,'minimun sampling interval');
```

Scilab code Exa 5.2.b Minimum Sampling Interval

```
//Ex 5.2b
clc;
disp('x(t)=(1/2%pi*t)[sin(31%pi*t)-sin(29%pi*t)]');
w1=31*%pi;
f1=w1/(2*%pi);
w2=29*%pi;
f2=w2/(2*%pi);
if f1<f2 then
    T=1/(2*f2);
else
    T=1/(2*f1);
end
disp(T,'minimun sampling interval');
```

Scilab code Exa 5.6 Continuous time Frequency

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

Scilab code Exa 5.9 Aliasing

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

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

Scilab code Exa 5.12 Find the frequency

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

Scilab code Exa 5.14 Sampling period

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

Chapter 6

Transformed Networks

Scilab code Exa 6.1 Current flowing in a network

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

Scilab code Exa 6.2 voltage across inductor

```
//Ex 6.2
clc;
syms s;
i=4;
```

```
R=13/4;
L=1;
C=1/13;
I=laplace(i)+1;
Y=1/R+s/13+1/s;
V=I/Y;
v=ilaplace(V);
disp(v,'v(t)=');
```

Scilab code Exa 6.3 voltage across capacitor

```
//Ex 6.3
clc;
syms s;
i=5;
R=2;
L=1;
C=1/2;
Z=((R+L*s)*(1/(C*s)))/((R+L*s)+(1/(C*s)));
V=Z*i;
v=ilaplace(V);
disp(v,'v(t)=');
```

Scilab code Exa 6.9 Bode Plot

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

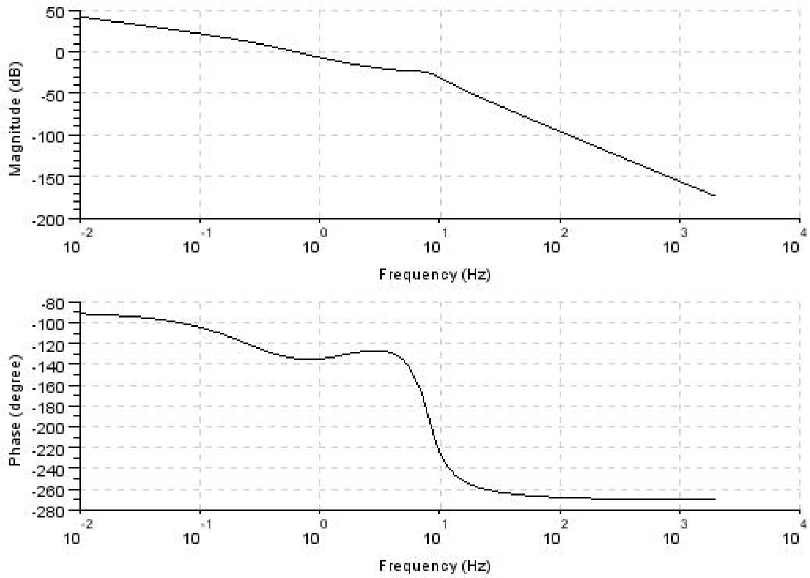

Figure 6.1: Bode Plot

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

Scilab code Exa 6.10 Bode Plot

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

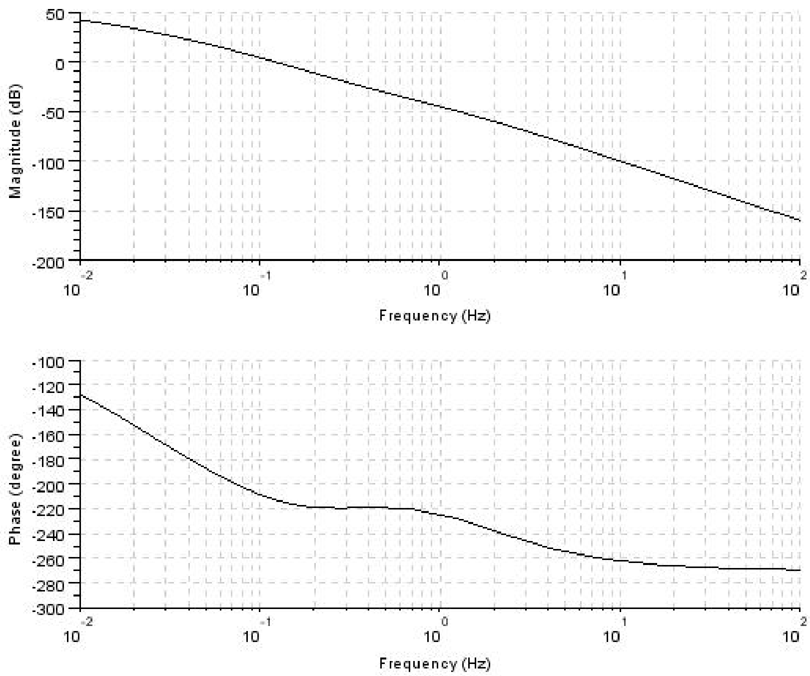

Figure 6.2: Bode Plot

Chapter 7

State Space Analysis

Scilab code Exa 7.1 State Space Representation

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

Scilab code Exa 7.2 State Space Representation

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

Chapter 8

Stability Analysis of LTI Systems

Scilab code Exa 8.1.a check for HURWITZ

```
clc;
// Define the polynomial
s=poly(0,"s");
p=10+9*s+4*s^2+s^3;
[C]=coeff(p);
l1=length(C);
x=0;
for i1=1:l1
    r1=C(1,i1);
    r1=real(a1);
    if r1>0 then
        x=x+1;
    end
end
if(x==l1) then
    [S]=roots(p);
    disp(S,"Roots=");
    l=length(S);
    c=0;
```

```
        for i=1:l
            a=S(i,1);
            r=real(a);
            if r<0 then
                c=c+1;
            end
        end
        if(c==l) then
            printf("Polynomial is Hurwitz");
        else
            printf("Polynomial is non-Hurwitz");
        end
    else
        printf("Polynomial is non-Hurwitz");
    end
```

Scilab code Exa 8.1.b check for HURWITZ

```
    clc;
    // Define the polynomial
    s=poly(0,"s");
    p=30+4*s+s^2+s^3;
    [C]=coeff(p);
    l1=length(C);
    x=0;
    for i1=1:l1
        a1=C(1,i1);
        r1=real(a1);
        if r1>0
            x=x+1;
        end
    end
    if(x==l1) then
        [S]=roots(p);
        disp(S,"Roots=");
```

```
    l=length(S);
    c=0;
    for i=1:l
        a=S(i,1);
        r=real(a);
        if r<0 then
            c=c+1;
        end
    end
    if(c==l) then
        printf("Polynomial is Hurwitz");
    else
        printf("Polynomial is non-Hurwitz");
    end
else
    printf("Polynomial is non-Hurwitz");
end
```

Scilab code Exa 8.2 checking stability

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

```
    if(c>=1) then
        printf("system is unstable");
        else
        printf("system is stable");
    end
```

Scilab code Exa 8.3 checking stability

```
    clc;
    // Define the polynomial
    s=poly(0,"s");
    p=2+2*s+5*s^2+4*s^3+4*s^4+2*s^5+s^6;
    // Calculate the routh of above polynomial
    r=routh_t(p);
    S=roots(p);
    disp(r,"Routh array=");
    disp(S,"Roots=");
    A=r(:,1);
    c=0;
    x=0;
    for i=1:6
        x=A(i,1);
        if x<0
            c=c+1;
        end
    end
    if(c>=1) then
        printf("system is unstable");
    else
        l=length(S);
        c=0;
        for i=1:l
            a=S(i,1);
            r=real(a);
            if r<0 then
```

```
                c=c+1;
            end
        end
        if c==0 then
           printf("system is stable");
        else
            printf("system is unstable");
        end

    end
```

Scilab code Exa 8.4 checking stability

```
clc;
// Define the polynomial
s=poly(0,"s");
p=1+2*s+5*s^2+5*s^3+2*s^4;
// Calculate the routh of above polynomial
r=routh_t(p);
disp(r,"Routh array=");
A=r(:,1);
c=0;
x1=0;
eps=0;
for i=1:5
    x1=A(i,1);
    if x1<0
        c=c+1;
    end
end
if(c>=1) then
    printf("system is unstable");
else
    printf("system is stable");
end
```

```
x=roots(p);
```

Scilab code Exa 8.5 checking stability

```
clc;
// Define the polynomial
s=poly(0,"s");
p=36+12*s+12*s^2+3*s^3+s^4;
// Calculate the routh of above polynomial
r=routh_t(p);
disp(r,"Routh array=");
A=r(:,1);
c=0;
x=0;
for i=1:5
    x=A(i,1);
    if x<0
        c=c+1;
        end
end
if(c>=1) then
    printf("system is unstable");
    else
    printf("system is stable");
end
x=roots(p);
```

Scilab code Exa 8.9 checking stability

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

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

Scilab code Exa 8.11 Stability of discrete time system

```
clc;
// Define the polynomial
//z=poly(0,'z');
//z^3 - 0.3*z^2 + .1*z -.1
//Put z=(1+r)/(1-r);
//Bilinear transformation
s=poly(0,'s');
p=.7+2.9*s+2.9*s^2+1.7*s^3;
// Calculate the routh of above polynomial
r=routh_t(p);
disp(r,"Routh array=");
A=r(:,1);
    c=0;
    x=0;
    for i=1:3
        x=A(i,1);
        if x<0
            c=c+1;
        end
    end
    if(c>=1) then
        printf("system is unstable");
    else
        l=length(S);
        c=0;
        for i=1:l
            a=S(i,1);
            r=real(a);
            if r<0 then
                c=c+1;
```

```
            end
        end
        if c==0 then
           printf("system is stable");
        else
            printf("system is unstable");
        end

    end
```

Chapter 9

Analog and Digital Filter Design

Scilab code Exa 9.1 BPF

```
//Ex 9.1
//Band pass Filter Design
//a
clc;
w1=10*10^3;
w2=15*10^3;
wot=sqrt(w1*w2);
wbt=w2-w1;
s=%s;
w=poly(0,'w');
wt=poly(0,'wt');
wx=(wt^2-wot^2)/(wbt*wt);
disp(w);
//b
//Let n=2
hb2=1/(s^2+sqrt(2)*s+1);
hb21=horner(hb2,(%i*w));
disp(hb21);
hb22=horner(hb21,wx);
```

```
disp(hb22);
```

Scilab code Exa 9.2 Band Stop Filter

```
//Ex 9.2
//Band Stop Filter Design
//a
clc;
w1=1200;
w2=2000;
s=%s;
w=poly(0,'w');
St=poly(0,'St');
wc=1; //For normalised Prototype
wd1t=poly(0,'wd1t');
wt1=2500;
wx1=(wt1*(w2-w1)*wd1t)/(-wt1^2+w2*w1);
wt2=400;
wx2=(wt2*(w2-w1)*wd1t)/(-wt2^2+w2*w1);
disp(w);
wx=wx1; // required attenuation to less
        // than -3dB for wx > -0.5195wd1t
wd1t=wc/0.5195;
//b
//Let n=4
hb2=1/((s^2+.7654*s+1).*(s^2+1.8478*s+1));
hb21=horner(hb2,(%i*w));
disp(hb21);
hb22=horner(hb21,wx);
disp(hb22);
hb23=horner(hb22,-%i*St);
disp(hb23);
```

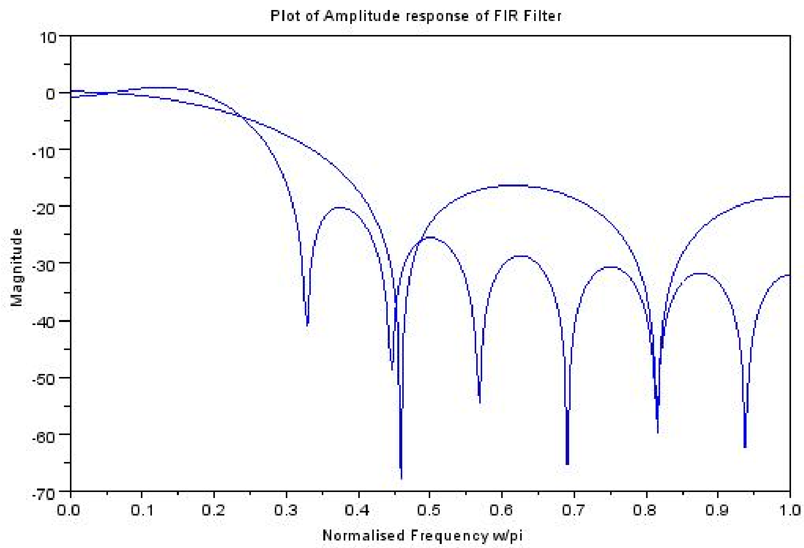

Figure 9.1: FIR Filter

Scilab code Exa 9.3 FIR Filter

```
//Ex 9.3
//FIR Filter
close;
clc;
//Fourier series method
//for N=5
N=5;
U=3;
for n=-2+U:1:2+U
if n==3
hd(n)=0.25;
else
hd(n)=sin(%pi*(n-U)/4)/(%pi*(n-U));
```

```
end
end
[hzm ,fr ]= frmag (hd ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
plot (2*fr , hzm_dB )

//for N=15
N=15;
U=8;
for n=-7+U:1:7+U
    if n==8
        hd(n)=0.25;
    else
        hd(n)=sin(%pi*(n-U)/4)/(%pi*(n-U));
    end
end
[hzm ,fr ]= frmag (hd ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
plot (2*fr , hzm_dB )
xlabel('Normalised Frequency w/pi')
ylabel('Magnitude')
title('Plot of Amplitude response of FIR Filter')
```

Scilab code Exa 9.5 Low Pass Filter

```
//Ex 9.5
clc;
N=21;
U=11;
for n=-10+U:1:10+U
if n==11
hd(n)=0.25;
else
```

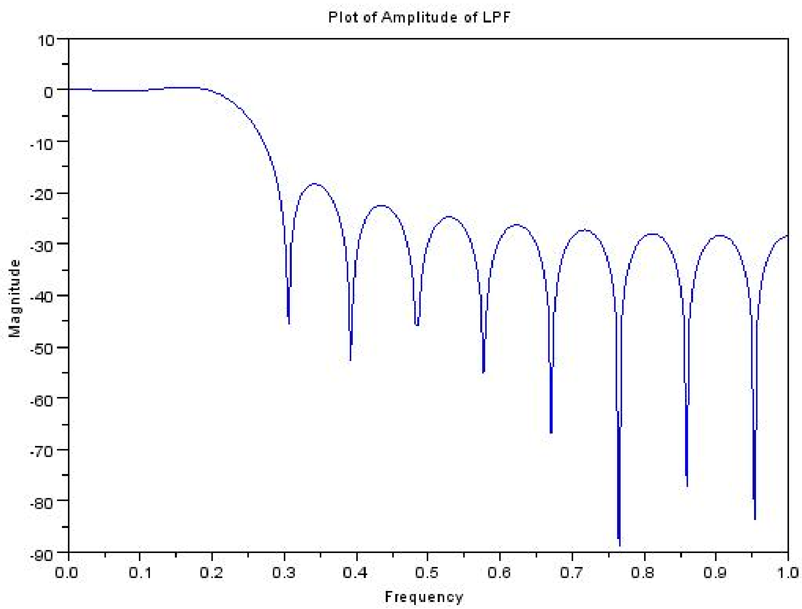

Figure 9.2: Low Pass Filter

```
hd(n)=(sin(%pi*(n-U)/4))/(%pi*(n-U));
end
end
[hzm ,fr ]= frmag (hd ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
plot (2*fr , hzm_dB );
xlabel ('Frequency');
ylabel ('Magnitude');
title('Plot of Amplitude of LPF')
```

Scilab code Exa 9.7 2nd order Digital Butterworth Filter

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

Chapter 10

MATLAB Tools for Analysis of Signals and systems

Scilab code Exa 10.1 Solving Linear Equation

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

Scilab code Exa 10.3 Find the step and impulse responce

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

Scilab code Exa 10.9 Find the impulse responce

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

Scilab code Exa 10.12 Find convolution and plot the result

```
//Ex 1012
//Convolution of two sequences
//[1 1 1 1 1]
//[1 2 3]
clc;
n1=0:1:4;
n2=0:1:2;
x=[1 1 1 1 1]
h=[1 2 3]
y=convol(x,h);
l=length(y);
n3=0:1:l-1;
figure
title('Sequence x')
plot2d3(n1,x);
figure
title('Seequence h')
plot2d3(n2,h);
figure
title('Sequence y')
plot2d3(n3,y);
```

Scilab code Exa 10.13 Find the step response

```
//Ex 1013
//Convolution of two sequences
//To plot the step response
//[.25 .25 .25 .25]
//[.25 -.25 .25 -.25]
clc;
n1=0:1:8;
n2=0:1:8;
h1=[0 0 0 .25 .25 .25 .25 0 0]
h2=[0 0 0 .25 -.25 .25 -.25 0 0]
y=convol(h1,h2);
l=length(y);
n3=0:1:l-1;
figure
title('Sequence h1')
plot2d3(n1,h1);
figure
title('Seequence h2')
plot2d3(n2,h2);
figure
title('Sequence y')
plot2d3(n3,y);
```