Scilab Textbook Companion for Linear Systems And Signals by B. P. Lathi[1]

Created by
Satyajit Gantayat
BTech
Electrical Engineering
NITK Surathkal
College Teacher
None
Cross-Checked by
None

April 14, 2019

[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: Linear Systems And Signals

Author: B. P. Lathi

Publisher: Oxford University Press,newyork

Edition: 2

Year: 2006

ISBN: 9780195686210

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

Chapter 1

SIGNALS AND SYSTEMS

Scilab code Exa 0.b.1 express complex numbers in polar form

```
clc
clear

//defining the given complex numbers
a=complex(2,3);
b=complex(-2,1);
c=complex(-2,-3);
d=complex(1,-3);

printf("sl \t abs_value \t angle(degree)\n");//
   printing the abs_value and phase of all complex
   numbers for polar form
printf("a \t %f \t %f   \n",abs(a),phasemag(a));
printf("b \t %f \t %f   \n",abs(b),phasemag(b));
printf("c \t %f \t %f   \n",abs(c),phasemag(c));
printf("d \t %f \t %f   \n",abs(d),phasemag(d));
```

Scilab code Exa 0.b.2 find Cartesian form of the given exponentials

```
clc
clear

//defining all the given datas
a=2*exp(%i*%pi/3);
b=4*exp(-%i*3*%pi/4);
c=2*exp(%i*%pi/2);
d=3*exp(-%i*3*%pi);
e=2*exp(%i*4*%pi);
f=2*exp(-%i*4*%pi);

printf("sl \t cartesian_form\n");//showing all the cartesian forms
disp("(a)    "+string(a));
disp("(b)    "+string(b));
disp("(c)    "+string(c));
disp("(d)    "+string(d));//scilab won't show exactly zero,so very small value can be regarded as 0 here
disp("(e)    "+string(e));
disp("(f)    "+string(f));
```

Scilab code Exa 0.b.3 `find polar and Cartesian form of complex no`

```
clc
clear

format(6)
z1=complex(3,4);
z2=complex(2,3);

mul=z1*z2;
div=z1/z2;
mag_mul=abs(mul);      //finding magnitude of z1*z2
phas_mul=phasemag(mul);//finding angle of z1*z2 in
```

```
degrees
mag_div=abs(div);       //finding magnitude of z1/z2
phas_div=phasemag(div);//finding angle of z1/z2 in degrees

printf("\nz1*z2 in cartesian form= %.1f+i%.1f, in polar form= %.1f/_%.1f degree\n",real(mul),imag(mul),mag_mul,phas_mul);//round-off error might be there
printf("\nz1/z2 in cartesian form= %.1f+i%.1f, in polar form= %.1f/_%.1f degree\n",real(div),imag(div),mag_div,phas_div);//round-off error might be there
```

Scilab code Exa 0.b.4 evaluate the given complex no operations

```
clc
clear

z1=2*exp(%i*%pi/4);
z2=8*exp(%i*%pi/3);

a=2*z1-z2;
b=1/z1;
c=z1/(z2)^2;
d=(z2)^(1/3);

disp("a= ");disp(a)
disp("b= ");disp(b)
disp("c= ");disp(c)
disp("d= ");disp(d)
```

Scilab code Exa 0.B.5 Cartesian and polar form

```
clc
clear

w=input("enter the value of omega:  ");//initialising w using value from users
X=(2+%i*w)/(3+%i*w*4);

rl_X=real(X);    //finding real part
img_X=imag(X);  //finding imaginary part
mag_X=abs(X);   //finding absolute part
phas_X=phasemag(X);//finding angle in degree

printf("\nX(w) in-\ncartesian form: real part= %f and imaginary part= %f\n\n",rl_X,img_X);
printf("polar form    : magnitude= %f  and angle= %f degrees\n",mag_X,phas_X);
```

Scilab code Exa 0.b.6 express given signals in single sinusoidal form

```
clc
clear

x1="cos(w0t)-sqrt(3)sin(w0t)";//given signal
a1=1;b1=-sqrt(3);
c1=sqrt(a1^2+b1^2);
theta_1=atan(-b1/a1);
theta_1=(180/%pi)*theta_1;//converting radian to degrees

x2="-3 cos(w0t)+4*sin(w0t)";//given signal
a2=-3;b2=4;
c2=sqrt(a2^2+b2^2);
theta_2=atan(-b2/a2);
theta_2=(180/%pi)*theta_2-180;//here 180 is reduced as the phasor lies in the 3rd quadrant because
```

```
atan gives values only in 1st quadrant

//showing the signals as single sinusoids with values with accuracy upto 2 points
printf("(a) x1(t)= %.2f*cos(w0t+%.2f(deg))\n",c1,theta_1);
printf(" (b) x2(t)= %.2f*cos(w0t%.2f(deg))\n",c2,theta_2);
```

Scilab code Exa 0.b.7 Cramers rule

```
clc
clear

A=[2,1,1;1,3,-1;1,1,1];//co-efficient matrix
b=[3;7;1];              //b matrix in the equation Ax=b

//finding modified matrices to find unknown variables
A1=[b A(1:3,2:3)];
A2=[A(1:3,1) b A(1:3,3)];
A3=[A(1:3,1:2) b];

//finding unknown variables using Cramer's rule
x1=(det(A1)/det(A))
x2=det(A2)/det(A)
x3=det(A3)/det(A)

printf("using cramers rule,we get: x1= %.0f , x2= %.0f , x3= %.0f \n",x1,x2,x3);
```

Scilab code Exa 0.b.8 Partial fraction

```
clc
clear

//for repeated roots use of pfss function is not
   efficient
//so manual calculation needed
s=%s
num=s^3+3*s^2+4*s+6;// given numerator
den1=(s+1);//given denominators
den2=(s+2);
den3=(s+3)^2;
root1=-1;root2=-2;root3=-3;

g1=num/(den2*den3);
g2=num/(den1*den3);
g3=num/(den1*den2);

k1=horner(g1,root1);           //finding co-
   efficient for (s+1)
k2=horner(g2,root2);           //finding co-
   efficient for (s+2)
k3=horner(derivat(g3),root3);  //finding co-
   efficient for (s+3)
k4=horner(g3,root3);           //finding co-
   efficient for (s+3)^2

printf("the partial fractions are as below\n");
printf("\t%.0f/(s+1)      %.0f/(s+2)         %.0f/(s
   +3)        %.0f/(s+3)^2",k1,k2,k3,k4);
```

Scilab code Exa 0.b.9 Partial fraction

```
clc
clear

```

```
//pfss is eficient for non-repeated roots

num = -11+9*%s + 2*%s^2 ;   //given numerator
den = (%s+1)*(%s-2)*(%s+3); //given denominator
h2 = syslin('c',num/den);
d = pfss(h2);
disp("the obtained partial fractions are");
disp(d)          //displaying the partial fractions
```

Scilab code Exa 0.b.10 Partial fraction

```
clc
clear

//for repeated roots use of pfss function is not
   efficient
s=%s
num=4*s^3+16*s^2+23*s+13;// given numerator
den1=(s+2);//given denominators
den2=(s+1)^3;
root1=-1;root2=-2;

g1=num/den1;
g2=num/den2;

k1=horner(g1,root1);                    //finding co
   -efficient for (s+1)^3
k2=horner(derivat(g1),root1);           //finding co
   -efficient for (s+1)^2
k3=horner(derivat(derivat(g1)),root1)/2;//finding co
   -efficient for (s+1)
k4=horner(g2,root2);                    //finding co
   -efficient for (s+2)

printf("the partial fractions are as below\n");
```

```
printf("\t%.0f/(s+1)^3        %.0f/(s+1)^2            %.0f/(s+1)            %.0f/(s+2)",k1,k2,k3,k4);
```

Scilab code Exa 0.b.11 Partial fraction

```
clc
clear

//defining transfer functions
num = -20+9*%s + 3*%s^2 ;//given numerator
den = (%s-2)*(%s+3); //given denominator
h2 = syslin('c',num/den);
d = pfss(h2,4,'c');     //finding partial fractions
disp("the obtained partal fractions are");
disp(d) //displaying the partial fractions
```

Scilab code Exa 0.b.12 Inverse of matrix

```
clc
clear

A=[2 1 1;1 2 3;3 2 1];
disp("the inverse matrix is");
disp(inv(A));
```

Scilab code Exa 0.b.13 eigen value of matrix and exponential of the matrix

```
clc
clear

```

```
//finding eigen values of the given matrix
A=[0 1;-2 -3];
egn=spec(A);
disp("the eigen values are  ");disp(egn);

//finding exponential of a matrix
[N,d]=coff(A);
b=N/d;
b=pfss(b);
disp("the laplace transform of each elements of the exponential matrix(e^(At)) are given as partial fractions below");
disp("the ILT of this will result in exponential matrix function")
;disp(b);
//let P=exponential matrix(e^(At))
//then here from b we get
//P(1)=inv_laplace(2/(1+s)-1/(s+2))=2e^(-t)-e^(-2t)
//P(2)=inv_laplace(-2/(1+s)+2/(s+2))=-2e^(-t)+2e^(-2t)
//P(3)=inv_laplace(1/(1+s)-1/(s+2))=e^(-t)-e^(-2t)
//P(4)=inv_laplace(-1/(1+s)+2/(s+2))=-e^(-t)+2e^(-2t)
```

Scilab code Exa 1.1 determine energy and power of given signals

```
clc
clear

//defining unit step function
function y=u(x)
  y=sign((sign(x)+1))
endfunction

format(12)
```

```
dt=0.0001;    //defining increment in time vector
t=-2:dt:20;
f=2.*(u(t+1)-u(t))+2.*exp(-t/2).*u(t); //defining the enrgy signal in fig(a)
g=t.*(u(t+1)-u(t-1));                   //defining one cycle of the periodoc signal from fig(b)

E=sum((f.^2)*dt);   //calculating energy of f
P=0.5*sum((g.^2)*dt); //calculating power of g
printf("\nthe energy of the signal a is %f",E);//ans won't be exactly 8 as infinity is not defined howmuch large the limit maybe
printf("\nthe power of the signal b is %f",P);
```

Scilab code Exa 1.3 show delayed and advanced signals

```
clc
clear

//defining unit step function
function y=u(x)
   y=sign((sign(x)+1))
endfunction

deff('[y]=f(x)','y=exp(-2.*x).*u(x)');//defining the signal function
t=-2:0.01:7;

subplot(3,1,1);plot(t,f(t));set(gca(),"grid",[1,1]);xtitle("original signal");//plotting original signal
subplot(3,1,2);plot(t,f(t-1));set(gca(),"grid",[1,1]);xtitle("delayed signal");//plotting
```

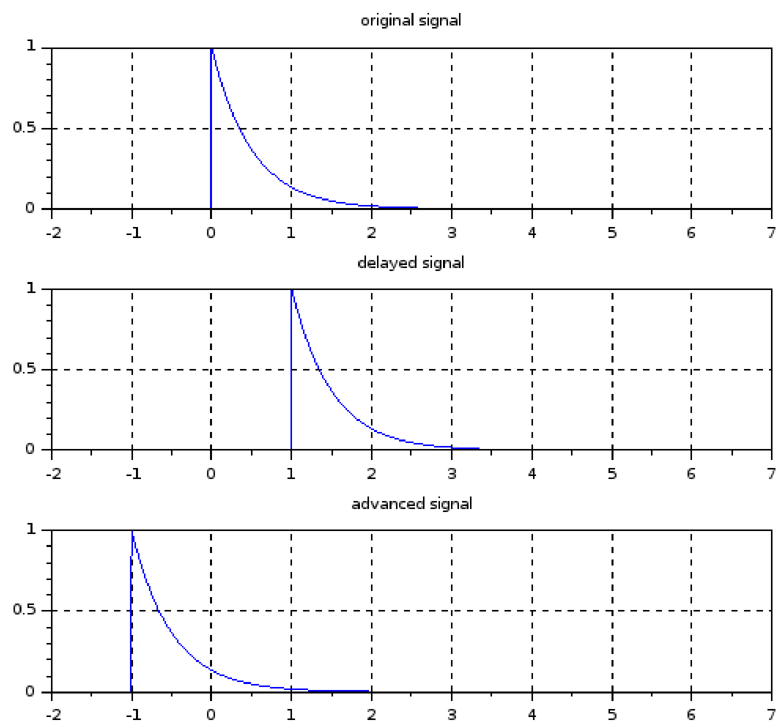

Figure 1.1: show delayed and advanced signals

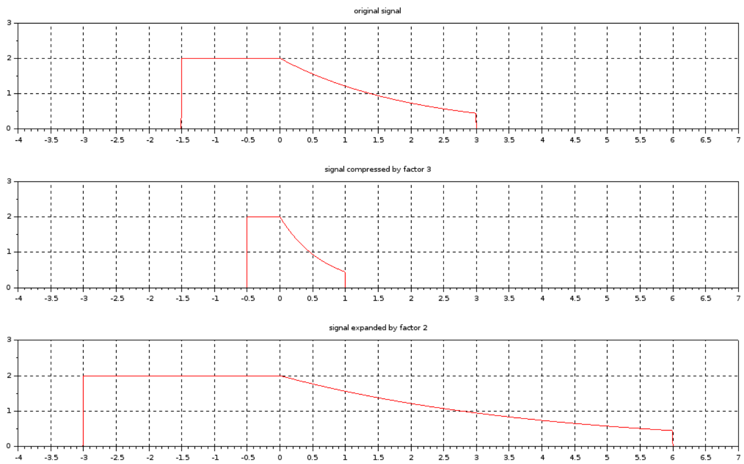

Figure 1.2: show compressed and expanded signals

```
    delayed signal
subplot(3,1,3);plot(t,f(t+1));set(gca(),"grid"
    ,[1,1]);xtitle("advanced signal");//plotting
    advanced signal
```

Scilab code Exa 1.4 `show compressed and expanded signals`

```
clc
clear

//defining unit step function
function y=u(x)
    y=sign((sign(x)+1))
endfunction

deff('[y]=f(x)','y=2.*exp(-x./2).*(u(x)-u(x-3))+2.*(
    u(x+1.5)-u(x))');//defining the signal function
t=-4:0.01:7;
```

```

subplot(3,1,1);
h=gca();
zoom_rect(h,[-4 0 7 3]);  //setting limits of the plot
plot(t,f(t),'r');               //plotting original signal
h.grid=[1,1];
xtitle("original signal");

subplot(3,1,2);
h=gca();
zoom_rect(h,[-4 0 7 3]);
plot(t,f(3.*t),'r');               //plotting compressed signal
h.grid=[1,1];
xtitle("signal compressed by factor 3");

subplot(3,1,3);
h=gca();
zoom_rect(h,[-4 0 7 3]);
plot(t,f(t./2),'r');               //plotting expanded signal
h.grid=[1,1];
xtitle("signal expanded by factor 2");
```

Scilab code Exa 1.5 plot reversed signal

```
clc
clear

//defining unit step function
function y=u(x)
```

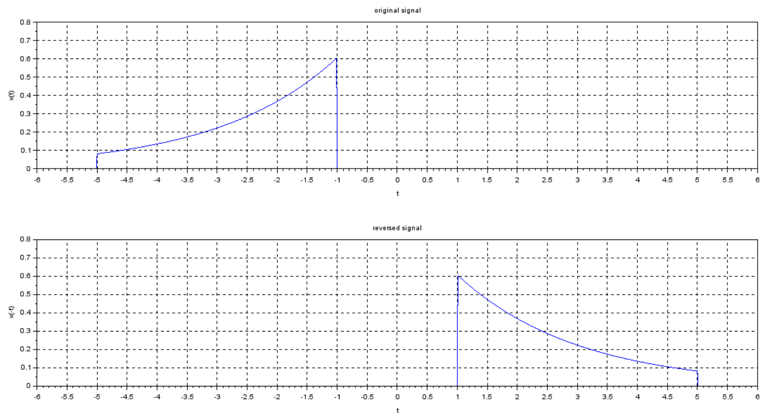

Figure 1.3: plot reversed signal

```
    y=sign((sign(x)+1))
endfunction

deff('[y]=f(x)','y=exp(x./2).*(u(x+5)-u(x+1))');//defining the signal function
t=-6:0.01:6;

subplot(2,1,1);
h=gca();
zoom_rect(h,[-6,0,6,0.8]);  //setting limits of the plot
plot(t,f(t));               //plotting original signal
h.grid=[1,1];
xtitle("original signal","t","x(t)");

subplot(2,1,2);
h=gca();
zoom_rect(h,[-6,0,6,0.8]);
plot(t,f(-t));              //plotting reversed signal
h.grid=[1,1];
xtitle("reversed signal","t","x(-t)");
```

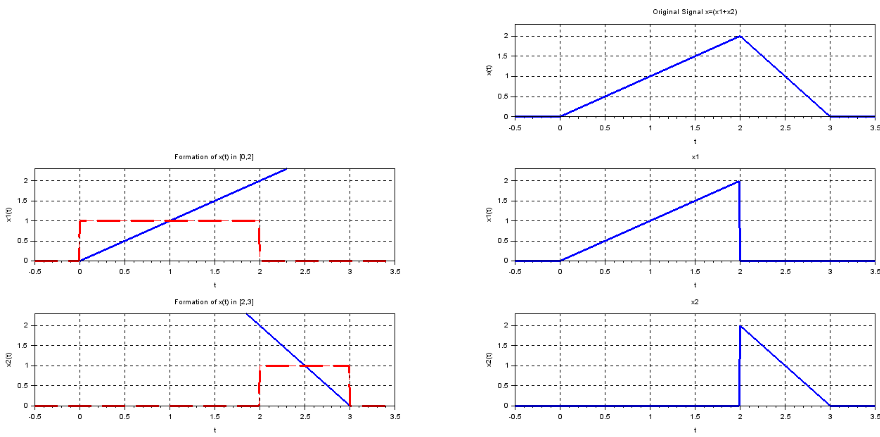

Figure 1.4: find expression for given signal and plot it

Scilab code Exa 1.6 `find expression for given signal and plot it`

```
clc
clear

//defining unit step function
function y=u(x)
   y=sign((sign(x)+1))
endfunction

deff('[y]=f(x)','y=t.*(u(t)-u(t-2))-2*(t-3).*(u(t-2)
   -u(t-3))');//finding expression for the given
   signal
t=-0.5:0.01:3.5;

subplot(3,2,2);plot(t,f(t));//plotting the signal
h=gca();
h.grid=[1,1]
```

```
xtitle("Original Signal x=(x1+x2)","t","x(t)")
zoom_rect([-0.5, 0, 3.5, 2.3]);

subplot(3,2,3);plot(t,t);plot(t,1.*(u(t)-u(t-2)),'--r');
h=gca();
h.grid=[1,1]
xtitle("Formation of x(t) in [0,2]","t","x1(t)");
zoom_rect([-0.5, 0, 3.5, 2.3]);

subplot(3,2,4);plot(t,t.*(u(t)-u(t-2)));
h=gca();
h.grid=[1,1]
xtitle("x1","t","x1(t)");
zoom_rect([-0.5, 0, 3.5, 2.3]);

subplot(3,2,5);plot(t,-2*(t-3));plot(t,1.*(u(t-2)-u(t-3)),'--r');
h=gca();
h.grid=[1,1]
xtitle("Formation of x(t) in [2,3]","t","x2(t)");
zoom_rect([-0.5, 0, 3.5, 2.3]);

subplot(3,2,6);plot(t,(-2*(t-3)).*(u(t-2)-u(t-3)));
h=gca();
h.grid=[1,1]
xtitle("x2","t","x2(t)");
zoom_rect([-0.5, 0, 3.5, 2.3]);
```

Scilab code Exa 1.7 find expression for the given signal and plot it

```
clc
clear
```

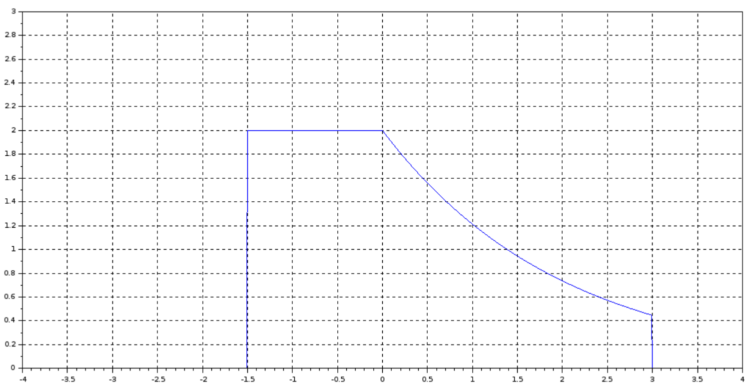

Figure 1.5: find expression for the given signal and plot it

```

//defining unit step function
function y=u(x)
   y=sign((sign(x)+1))
endfunction

deff('[y]=f(x)','y=2.*(u(t+1.5)-u(t))+2.*exp(-t./2)
   .*(u(t)-u(t-3))');//finding expression for the
   given signal
t=-4:0.01:4;

plot(t,f(t));//plotting the signal
h=gca();
h.grid=[1,1];
zoom_rect(h,[-4 0 4 3])
```

Scilab code Exa 1.8 `find even and odd component of the given signal`

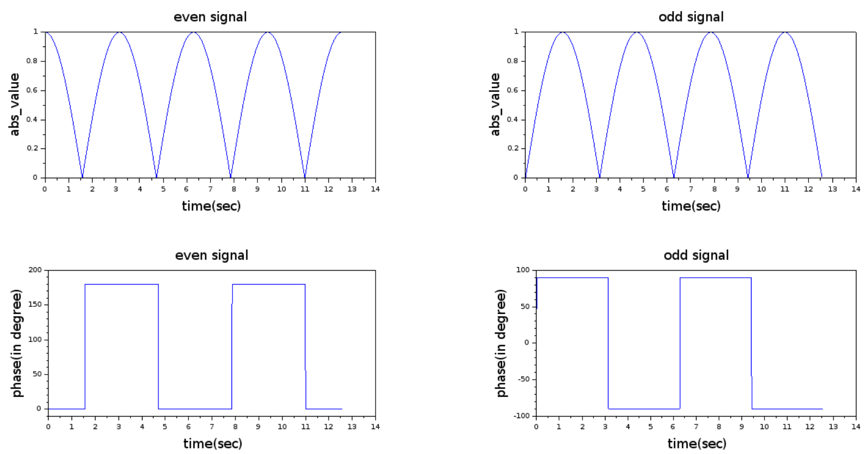

Figure 1.6: find even and odd component of the given signal

```
clc
clear

t=0:0.01:4*%pi;
odd=0.5.*(exp(%i.*t)-exp(-%i.*t));//finding odd component of given signal
even=0.5.*(exp(%i.*t)+exp(-%i.*t));//finding even component of given signal

subplot(2,2,1);plot(t,abs(even));//plotting the even part
title("even signal","fontsize",4);
xlabel("time(sec)","fontsize",4),ylabel("abs_value","fontsize",4);

subplot(2,2,3);plot(t,phasemag(even));
title("even signal","fontsize",4);
xlabel("time(sec)","fontsize",4);ylabel("phase(in degree)","fontsize",4);
h=gca();
zoom_rect(h,[0,-10,14,200]);

```

```
subplot(2,2,2);plot(t,abs(odd));//plotting the odd part
xlabel("time(sec)","fontsize",4),ylabel("abs_value","fontsize",4);
title("odd signal","fontsize",4);

subplot(2,2,4);plot(t,phasemag(odd));
xlabel("time(sec)","fontsize",4);ylabel("phase(in degree)","fontsize",4);
title("odd signal","fontsize",4);
```

Chapter 2

Time domain analysis of continuous time systems

Scilab code Exa 2.1 Zero Input Response ZIR of the given systems

```
clc
clear
close;
funcprot(0)

//********************part a(non-repeated real roots)***********************
//--------------------------------------------------
//--------------------------------------------------
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1);
endfunction

y0=[0;-5];
t0=0;
t=linspace(0,10,500);
y=ode(y0,t0,t,f);   //solving the 2nd order ode
```

```
system equation

figure(1)
subplot(1,3,1);plot(t,y(1,:));      //plotting the obtained result
title("ZIR of the system(non-repeated roots)_part_a","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(ZIR)","fontsize",4);

subplot(1,3,2);plot(t,-5.*exp(-t)+5.*exp(-2.*t));//plotting the estimated answer for comparision
title("estimated ans for comparison_part a","fontsize",4);
xlabel("time","fontsize",4);
ylabel("-5.*exp(-t)+5.*exp(-2.*t)","fontsize",4);

//____________________________________________________________________________
//____________________________________________________________________________




//**************************part b(repeated real roots)*****************************
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-6*y(2)-9*y(1);
endfunction

y0=[3;-7];
t0=0;
t=linspace(0,10,500);
y=ode(y0,t0,t,f);   //solving the 2nd order ode
```

```
    system equation

subplot(1,3,3);plot(t,y(1,:));      //plotting the obtained result
title("ZIR of the system(repeated roots)_part b","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(ZIR)","fontsize",4);

figure(2)
subplot(1,3,1);plot(t,(3+2*t).*exp(-3*t));//plotting the estimated answer for comparision
zoom_rect([0,-0.5,15,3])
title("estimated ans for comparison_part b","fontsize",4);
xlabel("time","fontsize",4);
ylabel("(3+2*t).*exp(-3*t))","fontsize",4);

//____________________________________________________________________________________
//____________________________________________________________________________________



//*******************part c(non-repeated imaginary roots)*************************8
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-4*y(2)-40*y(1);
endfunction

y0=[2;16.78];
t0=0;
t=linspace(0,10,500);
```

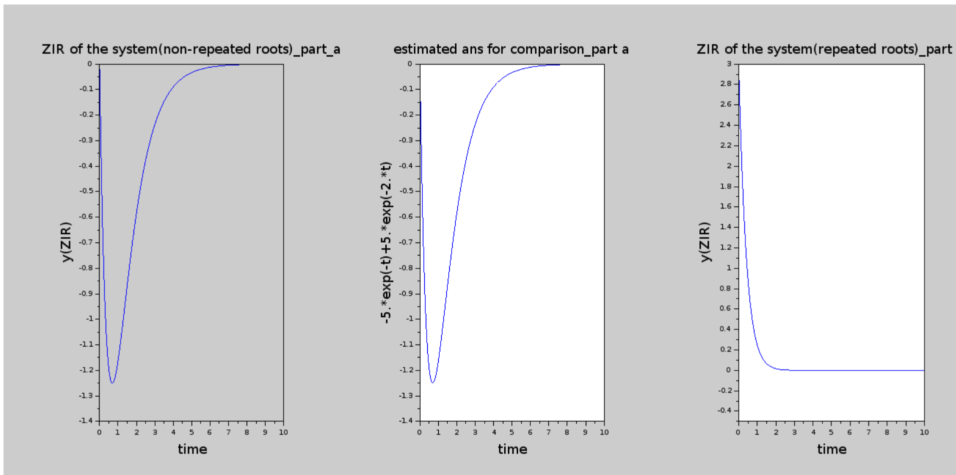

Figure 2.1: Zero Input Response ZIR of the given systems

```
y=ode(y0,t0,t,f);    //solving the 2nd order ode
    system equation

subplot(1,3,2);plot(t,y(1,:));      //plotting the
    obtained result
title("ZIR of the system(imaginary roots)_part c","
    fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(ZIR)","fontsize",4);


subplot(1,3,3);plot(t,4*exp(-2*t).*cos(6*t-%pi/3));
    //plotting the estimated answer for comparision
//zoom_rect([0,-0.5,15,3])
title("estimated ans for comparison_part c","
    fontsize",4);
xlabel("time","fontsize",4);
ylabel("4*exp(-2*t).*cos(6*t-%pi/3)","fontsize",4);
```

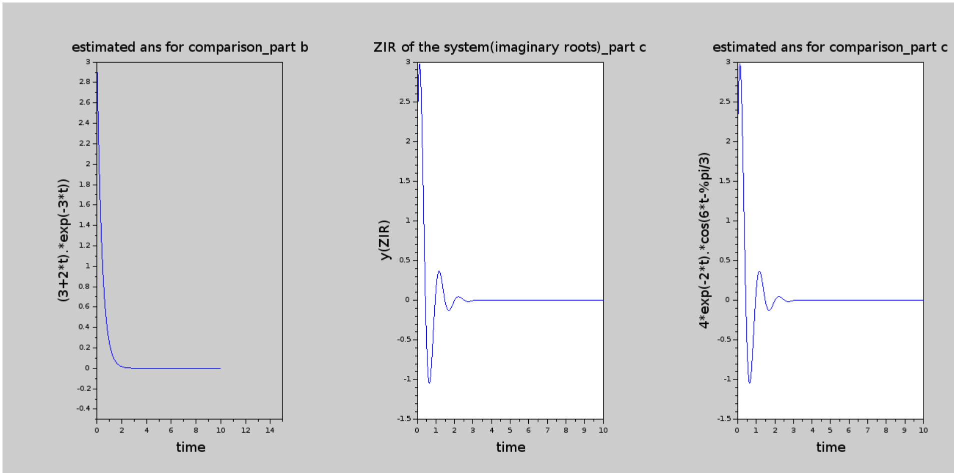

Figure 2.2: Zero Input Response ZIR of the given systems

Scilab code Exa 2.2 `Finding loop current and initial values`

```
clc
clear
close;

function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1);
endfunction

y0=[0;-5];
t0=0;
t=linspace(0,15,500);
```

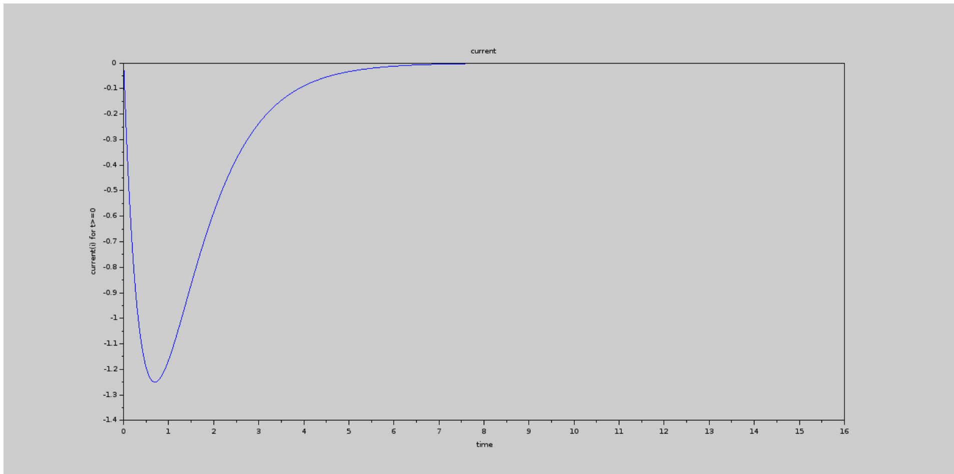

Figure 2.3: Finding loop current and initial values

```
y=ode(y0,t0,t,f);    //solving the 2nd order ode system equation
v_0=5;
y_0=y0(1);
dy_0= 10-v_0-3*y_0;//eq obtained from circuit

figure(1)
plot(t,y(1,:));      //plotting the obtained result
xtitle("current ","time","current(i) for t>=0");
printf("\nthe value of y(0+)= %d,  y''(0+)= %d\n\n",y_0,dy_0);
```

Scilab code Exa 2.3 `Finding impulse response of the system`

```
clc
clear
close;

```

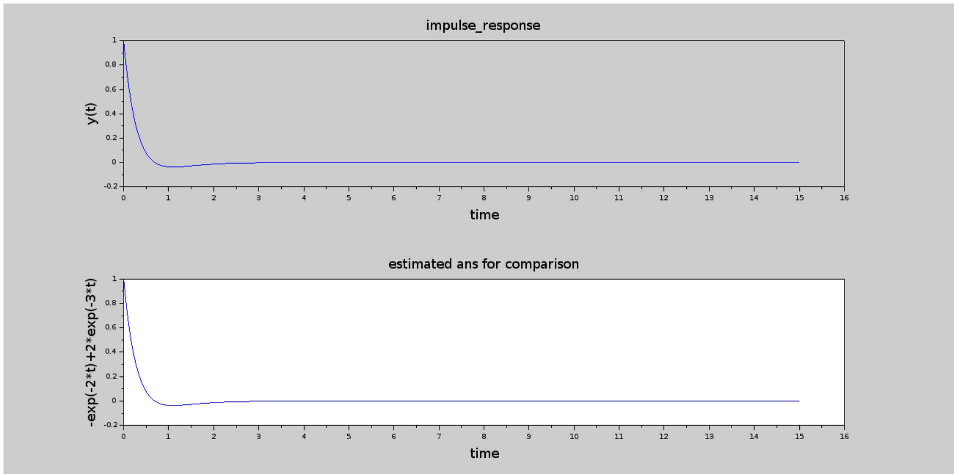

Figure 2.4: Finding impulse response of the system

```
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-5*y(2)-6*y(1);
endfunction

y0=[1;-4];//initial conditions for impulse response
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);    //solving the 2nd order ode system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));      //plotting the obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(t)","fontsize",4);

//checking if the obtained ans is correct

subplot(2,1,2);plot(t,-exp(-2*t)+2*exp(-3*t));//
```

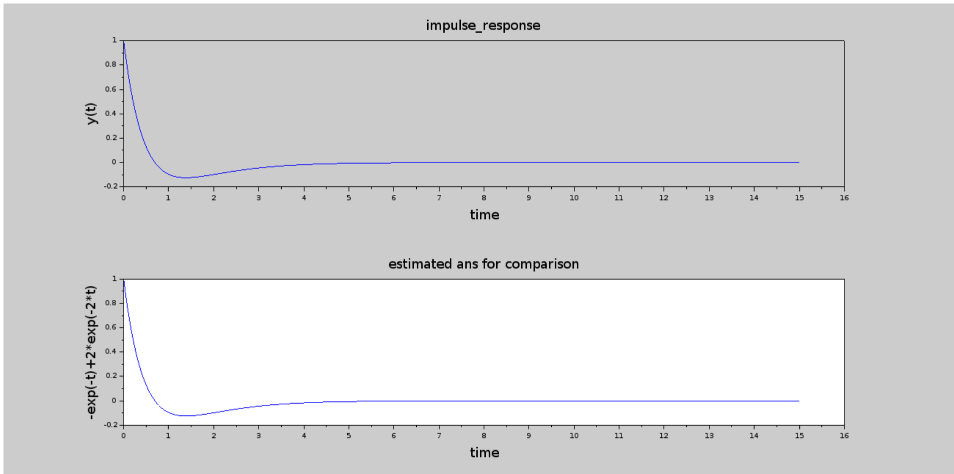

Figure 2.5: Finding impulse response of the system

```
    plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
xlabel("time","fontsize",4);
ylabel("−exp(−2∗t)+2∗exp(−3∗t)","fontsize",4);
```

Scilab code Exa 2.4 Finding impulse response of the system

```
clc
clear
close;

function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1);
endfunction

```

```
y0=[1;-3];//initial conditions for impulse response calculated from given system equation
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);    //solving the 2nd order ode system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));      //plotting the obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(t)","fontsize",4);

//checking if the obtained ans is correct

subplot(2,1,2);plot(t,-exp(-t)+2*exp(-2*t));//plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
xlabel("time","fontsize",4);
ylabel("-exp(-t)+2*exp(-2*t)","fontsize",4);
```

Scilab code Exa 2.5 `ZSR of given system`

```
clc
clear
close;

t=[0:0.1:5];
// Defining unit step function
    function y=u(x)
        y=sign((sign(x)+1))
    endfunction
```

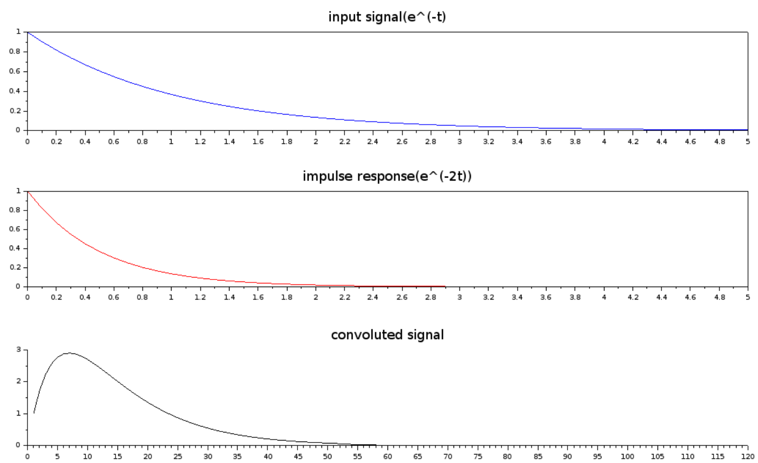

Figure 2.6: ZSR of given system

```

x=exp(-t).*u(t);//defining the iinput function
h=exp(-2*t).*u(t);//defining the impulse response

subplot(3,1,1);plot(t,x,'b');
title("input signal(e^(-t)","fontsize",4);

subplot(3,1,2);plot(t,h,'r');
title("impulse response(e^(-2t))","fontsize",4);

[z]=convol(x,h); //operating convolution operation
subplot(3,1,3);
plot2d(z);
title("convoluted signal","fontsize",4);
printf("the obtained graph should be iinterpolated back to t=0 to get completre ZSR")
```

Scilab code Exa 2.6 `Finding loop current`

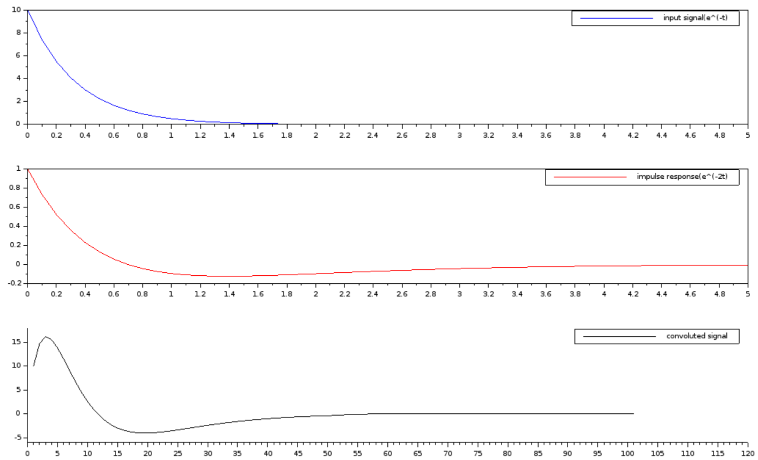

Figure 2.7: Finding loop current

```
clc
clear
close;

t=[0:0.1:5];
// Defining unit step function
    function y=u(x)
        y=sign((sign(x)+1))
    endfunction

x=10.*exp(-3*t).*u(t);//defining the iinput function
h=(2.*exp(-2*t)-exp(-t)).*u(t);//defining the
   impulse response

subplot(3,1,1);plot(t,x,'b');
legend("input signal(e^(-t)");

subplot(3,1,2);plot(t,h,'r');
legend("impulse response(e^(-2t)");

[z]=convol(x,h); //operating convolution operation
subplot(3,1,3);
```

```
plot2d(z);
legend("convoluted signal");
printf("the obtained graph should be iinterpolated
   back to t=0 to get completre ZSR")
```

check Appendix AP 3 for dependency:

conv_gui.sci

Scilab code Exa 2.7 `Graphical convolution`

```

clc
close;


//execute the dependecy file first that is needed
   for graphical convolution
//exec('/home/satyajit/Desktop/my_octave/
   scilab_project/chapter_2/conv_gui.sci', -1)//
   change your path to the location of the file "
   conv_gui.sci"

//all the inputs should be given in the form of
   strings

t='[-5:0.02:5]';
f1='exp(-t).*u(t)';     //defining x(t)
f2='exp(-2*t).*u(t)';  //defining h(t)
conv_gui(f1,f2,t);


//for analysis goto options > illustration > move
   right in the plot window
```

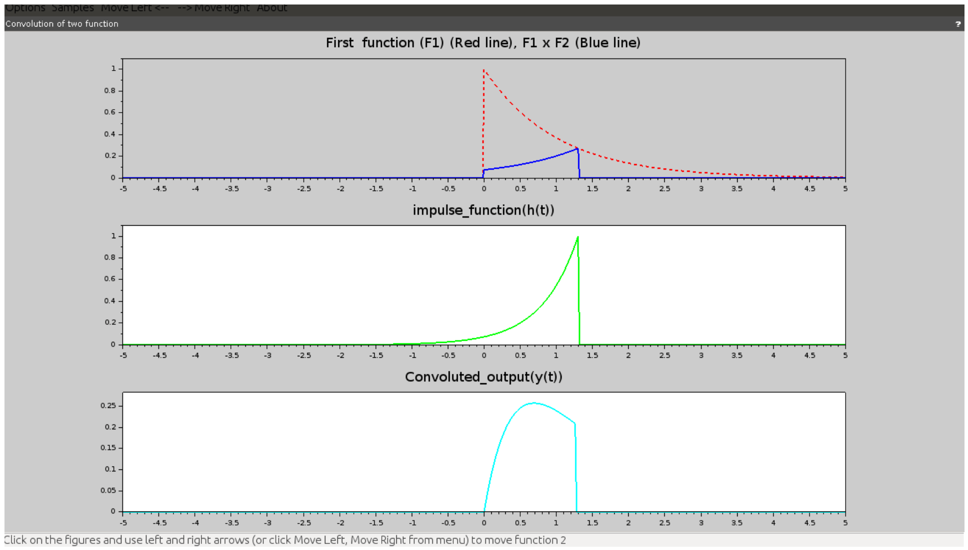

Figure 2.8: Graphical convolution

check Appendix AP 3 for dependency:

`conv_gui.sci`

Scilab code Exa 2.8 `Graphical convolution`

```

clc
close;

//execute the dependecy file first that is needed
   for graphical convolution
//exec('/home/satyajit/Desktop/my_octave/
   scilab_project/chapter_2/conv_gui.sci', -1)
```

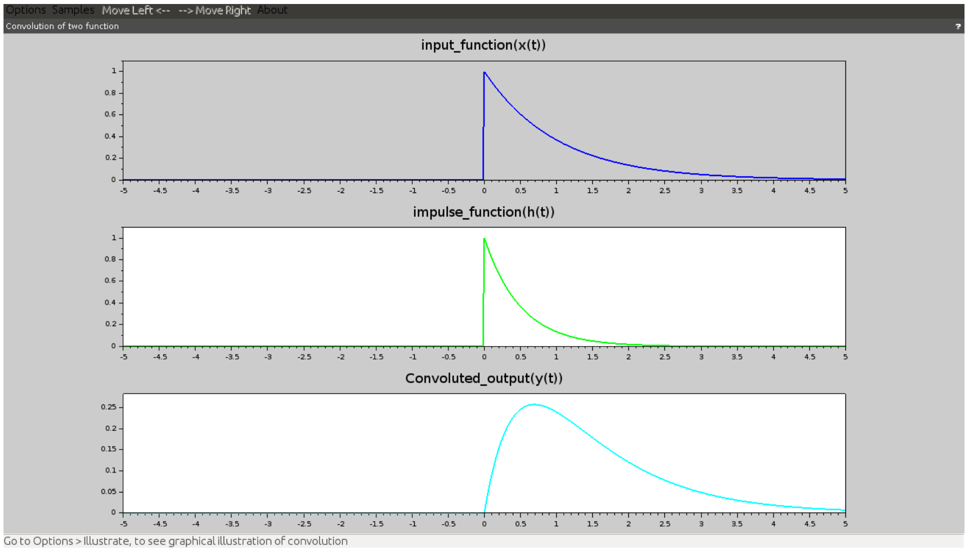

Figure 2.9: Graphical convolution

```
//change your path to the location of the file "conv_gui.sci"


//all the inputs should be given in the form of strings

t='[-5:0.02:5]';
f1='u(t)';     //defining x(t)
f2='-2*exp(2*t).*u(-t)+2*exp(-t).*u(t)'; //defining h(t)
conv_gui(f1,f2,t);


//for analysis goto options > illustration > move right in the plot window
```

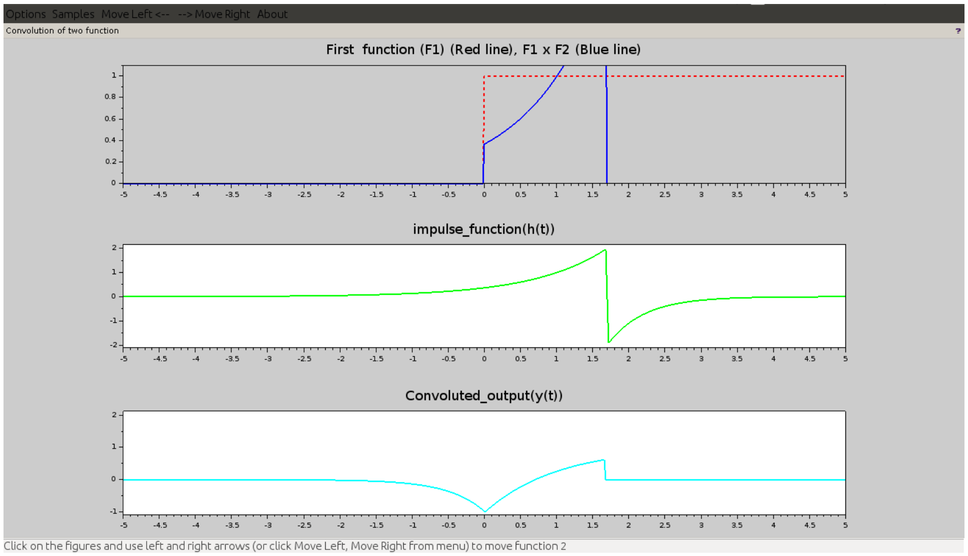

Figure 2.10: Graphical convolution

check Appendix AP 3 for dependency:

`conv_gui.sci`

Scilab code Exa 2.9 `Graphical convolution`

```

clc
close;

//execute the dependecy file first that is needed for graphical convolution
//exec('/home/satyajit/Desktop/my_octave/scilab_project/chapter_2/conv_gui.sci', -1)
```

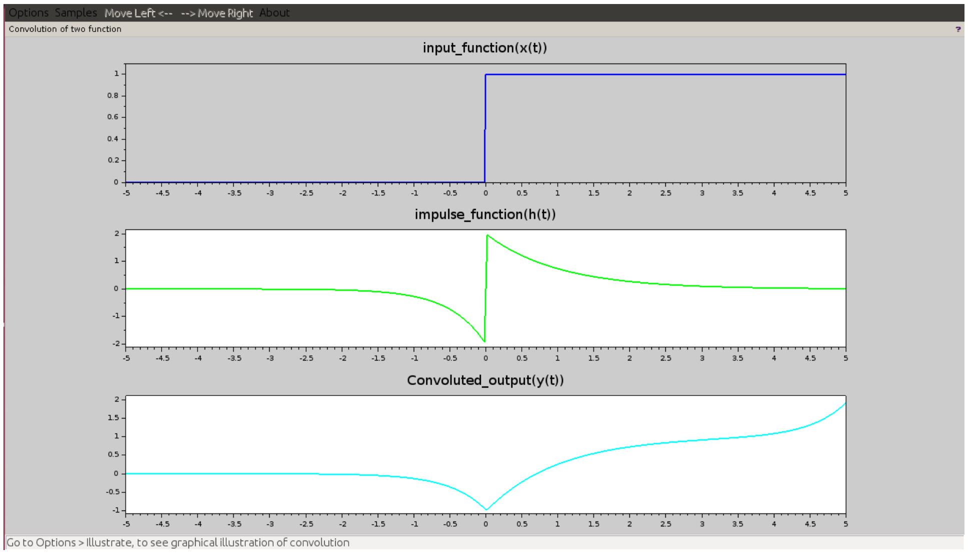

Figure 2.11: Graphical convolution

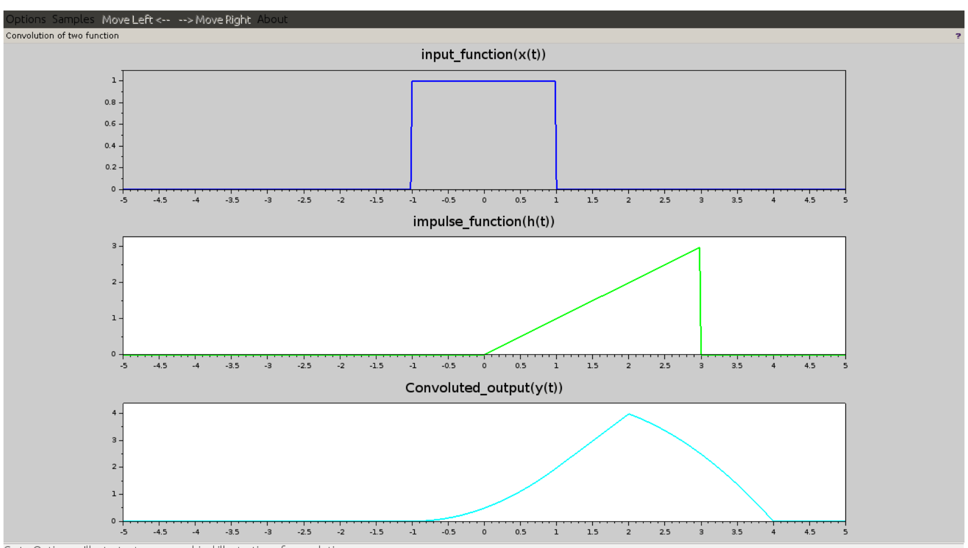

Figure 2.12: Graphical convolution

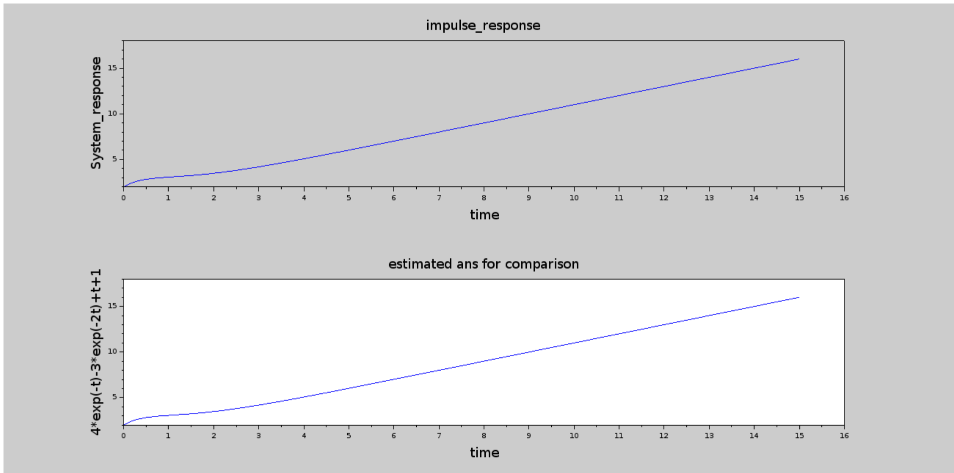

Figure 2.13: Solving ODE of given system

```
//change your path to the location of the file "
    conv_gui.sci"


//all the inputs should be given in the form of
    strings

t='[-5:0.02:5]';
f1='u(t+1)-u(t-1)';    //defining x(t)
f2='t.*(u(t)-u(t-3))'; //defining h(t)
conv_gui(f1,f2,t);


//for analysis goto options > illustration > move
    right in the plot window
```

Scilab code Exa 2.10 `Solving ODE of given system`

```
clc
clear
close;

function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1)+2*t+5;
endfunction

y0=[2;3];//initial conditions for impulse response
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);    //solving the 2nd order ode
   system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));      //plotting the
   obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("System_response","fontsize",4);


//checking if the obtained ans is correct

subplot(2,1,2);plot(t,4*exp(-t)-3*exp(-2*t)+t+1);//
   plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
xlabel("time","fontsize",4);
ylabel("4*exp(-t)-3*exp(-2t)+t+1","fontsize",4);
```

Scilab code Exa 2.11 Analysing system behaviour with different inputs

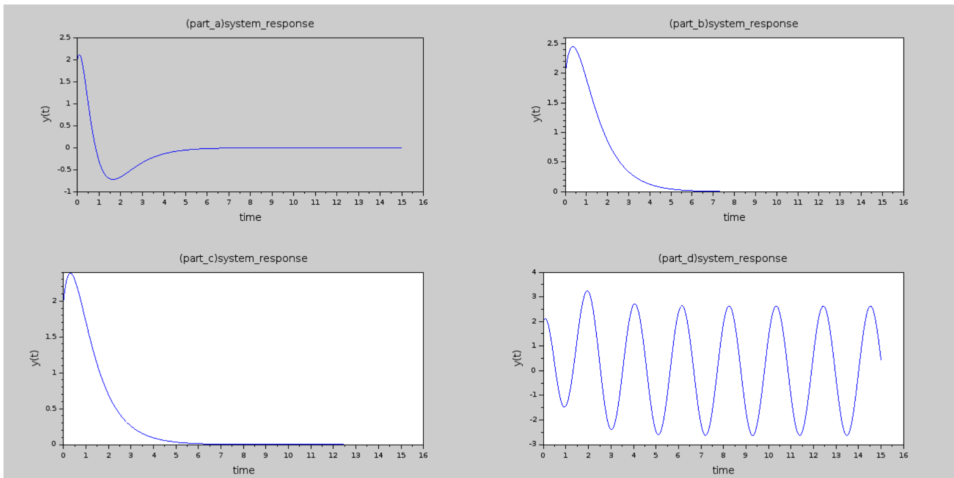

Figure 2.14: Analysing system behaviour with different inputs

```
clc
clear
close;

y0=[2;3];//initial conditions for impulse response
t0=0;
t=linspace(0,15,500);

//************************part-a(x(t)=10e^(-3t))************************
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1)-30*exp(-3*t);
endfunction

y1=ode(y0,t0,t,f);    //solving the 2nd order ode system equation

figure(1)
subplot(2,2,1);plot(t,y1(1,:));      //plotting the obtained result
```

```
title("(part_a)system_response","fontsize",3);
xlabel("time","fontsize",3);
ylabel("y(t)","fontsize",3);


//************************part-b(x(t)=5))************************
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1);
endfunction

y2=ode(y0,t0,t,f);   //solving the 2nd order ode system equation

subplot(2,2,2);plot(t,y2(1,:));     //plotting the obtained result
title("(part_b)system_response","fontsize",3);
xlabel("time","fontsize",3);
ylabel("y(t)","fontsize",3);



//************************part-c(x(t)=e^-2t)************************
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1)-2*exp(-2*t);
endfunction

y3=ode(y0,t0,t,f);   //solving the 2nd order ode system equation

subplot(2,2,3);plot(t,y3(1,:));     //plotting the obtained result
title("(part_c)system_response","fontsize",3);
```

```
xlabel("time","fontsize",3);
ylabel("y(t)","fontsize",3);



//************************part-d(x(t)=10cos(3t+pi/3)************************
function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1)-30*sin(3*t+%pi/3);
endfunction

y4=ode(y0,t0,t,f);    //solving the 2nd order ode system equation

subplot(2,2,4);plot(t,y4(1,:));     //plotting the obtained result
title("(part_d)system_response","fontsize",3);
xlabel("time","fontsize",3);
ylabel("y(t)","fontsize",3);
```

Scilab code Exa 2.12 Finding loop current

```
clc
clear
close;

function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1)-30*exp(-3*t);
endfunction
```

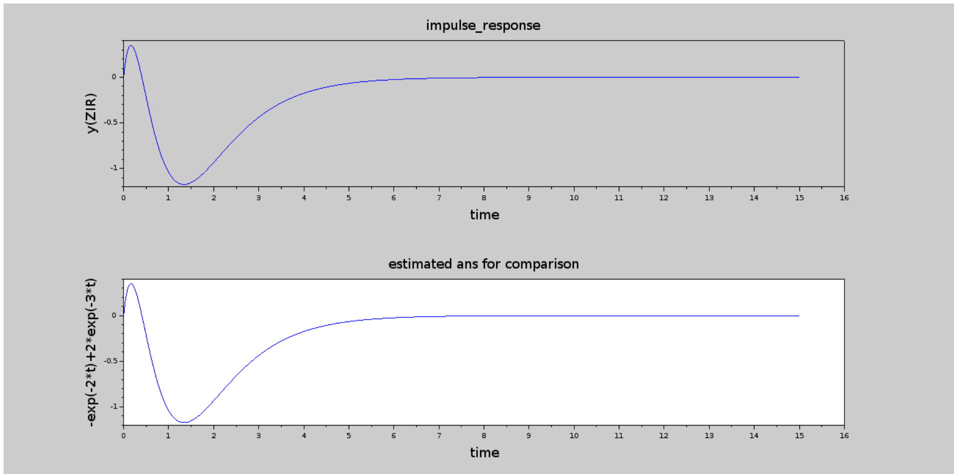

Figure 2.15: Finding loop current

```

y0=[0;5];//initial conditions for impulse response
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);    //solving the 2nd order ode system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));      //plotting the obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(ZIR)","fontsize",4);


//checking if the obtained ans is correct

subplot(2,1,2);plot(t,-10*exp(-t)+25*exp(-2*t)-15*exp(-3*t));//plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
xlabel("time","fontsize",4);
```

```
ylabel("-exp(-2*t)+2*exp(-3*t)","fontsize",4);
```

Chapter 3

Time domain analysis of discrete time systems

Scilab code Exa 3.1 energy and power of discrete signals

```
clc
clear

n=[0:1:5];
// Defining unit step function
function y=u(x)
   y=sign((sign(x)+1))
endfunction

f=n.*(u(n)-u(n-6));//defining the given function
subplot(1,2,1);plot2d3(n,f);
title("figure(a)","fontsize",4);
f2=repmat(f,1,5);
n1=0:1:length(f2)-1;
subplot(1,2,2);plot2d3(n1,f2,[5]);title("figure(b)",
   "fontsize",4);
energy=sum(f.^2);//finding energy of signal in fig(a
```

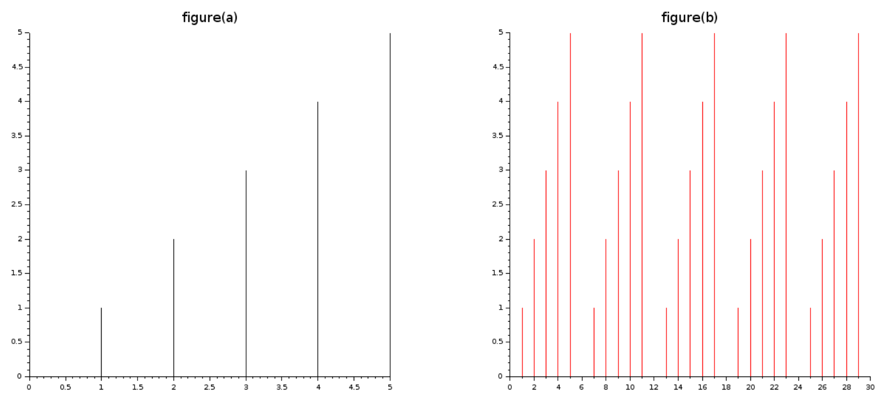

Figure 3.1: energy and power of discrete signals

```
)
power=(1/6).*sum(f.^2);//finding power of signal in fig(b)

printf("\nthe energy in fig (a) is %.2f and power in fig(b) is %.2f\n",energy,power);//rounding off the answer to two digits
```

Scilab code Exa 3.3 discrete signal representation

```
clc
clear

n=[0:1:5];
// Defining unit step function
function y=u(x)
   y=sign((sign(x)+1))
endfunction
```

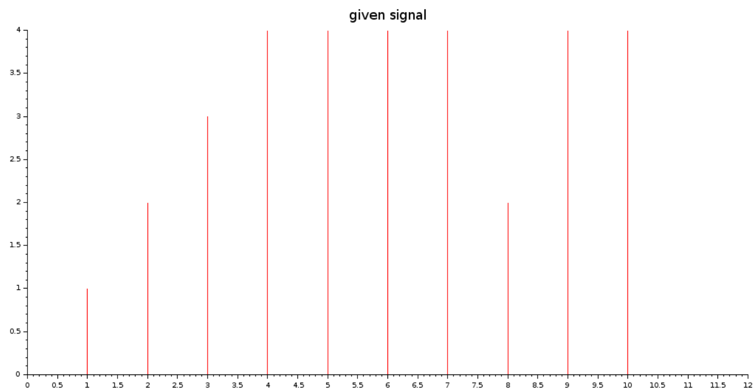

Figure 3.2: discrete signal representation

```

n=0:1:11;
f=n.*(u(n)-u(n-5))+4.*(u(n-5)-u(n-11))-2.*(n==8);//defining the given function
plot2d3(n,f,[5])
title("given signal","fontsize",4)
```

Scilab code Exa 3.8 `iterative solution of discrete time systems`

```
clc
clear
close

n=[-1:4]';//defining the discrete time
y= [16;zeros(length(n)-1,1)];

//predefining the initial conditions + zero output array
```

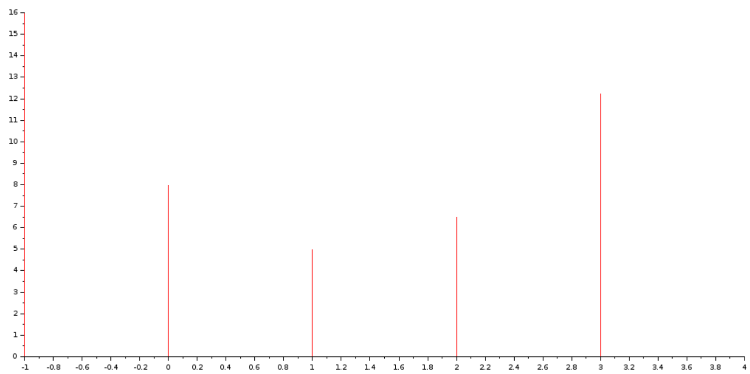

Figure 3.3: iterative solution of discrete time systems

```

x=[0;n(2:$).^2]; //defining input array with first entry corresponding to time instant −1

for k=1:length(n)-2
y(k+1) =0.5*y(k)+x(k+1);
end;

plot2d3(n,y,[5]);
```

Scilab code Exa 3.9 `iterative solution of discrete time systems`

```
clc
clear
close

n=[-2:10]';//defining the discrete time
y= [1;2;zeros(length(n)-2,1)];
```

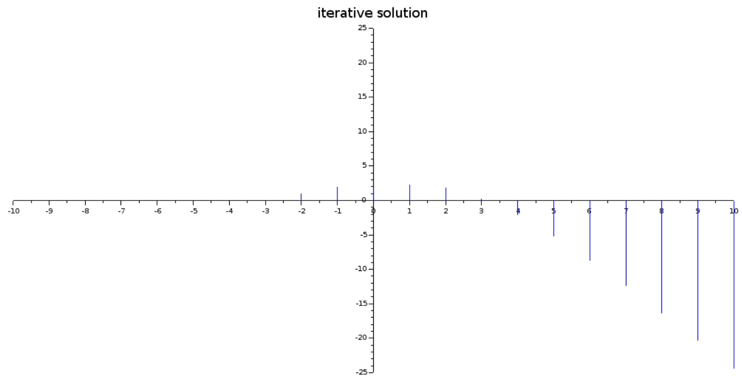

Figure 3.4: iterative solution of discrete time systems

```

//predefining the initial conditions + zero output array

x=[0;0;n(3:$)]; //defining input array with first two entries corresponding to time instants −2 and −1

for k=1:length(n)-2
y(k+2) = y(k+1) -0.24*y(k) + x(k+2) - 2*x(k+1);
end;

plot2d3(n,y,[2]);
set(gca(),"zoom_box",[-10 -25 10 25],"x_location","middle","y_location","middle");
title("iterative solution","fontsize",4)
```

Scilab code Exa 3.10 Zero Input Response of the various systems

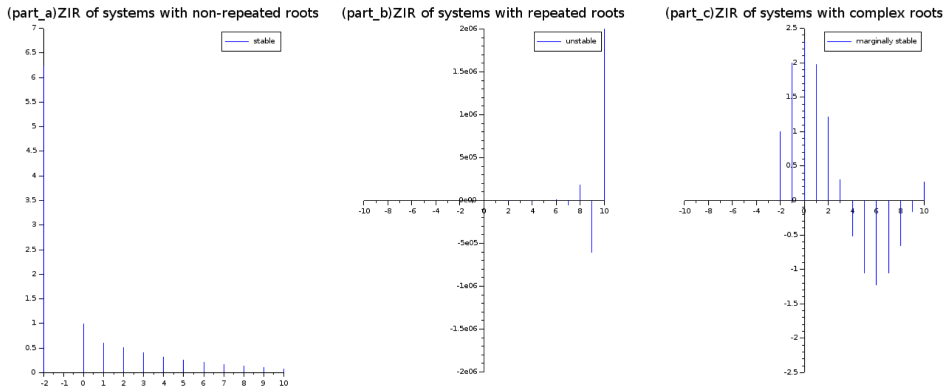

Figure 3.5: Zero Input Response of the various systems

```
clc
clear
close

n=[-2:10]';//defining the discrete time

//******************************part(a)(non-repeated roots)********************

y1= [25/4;0;zeros(length(n)-2,1)];//predefining the initial conditions + zero output array
for k=1:length(n)-2
y1(k+2) = 0.6*y1(k+1) + 0.16*y1(k);
end;
subplot(1,3,1);plot2d3(n,y1,[2]);title("(part_a)ZIR of systems with non-repeated roots","fontsize",4);
legend("stable");

//******************************part(b)(repeated roots)********************

```

```
y2= [-2/9;-1/3;zeros(length(n)-2,1)];//predefining the initial conditions + zero output array
for k=1:length(n)-2
y2(k+2) = -6*y2(k+1)-9*y2(k);
end;
subplot(1,3,2);plot2d3(n,y2,[2]);title("(part_b)ZIR of systems with repeated roots","fontsize",4);
set(gca(),"zoom_box",[-10 -max(y2) 10 max(y2)],"x_location","middle","y_location","middle");
legend("unstable");

//*****************************part(c)(complex roots)********************

y3= [1;2;zeros(length(n)-2,1)];//predefining the initial conditions + zero output array
for k=1:length(n)-2
y3(k+2) =1.56* y3(k+1) -0.81*y3(k);
end;
subplot(1,3,3);plot2d3(n,y3,[2]);title("(part_c)ZIR of systems with complex roots","fontsize",4);
set(gca(),"zoom_box",[-10 -2.5 10 2.5],"x_location","middle","y_location","middle");
legend("marginally stable");
```

Scilab code Exa 3.11 impulse response

```
clc
clear
close;

n=[-2:20]';//defining the discrete time
y= [0;0;zeros(length(n)-2,1)];//predefining the
```

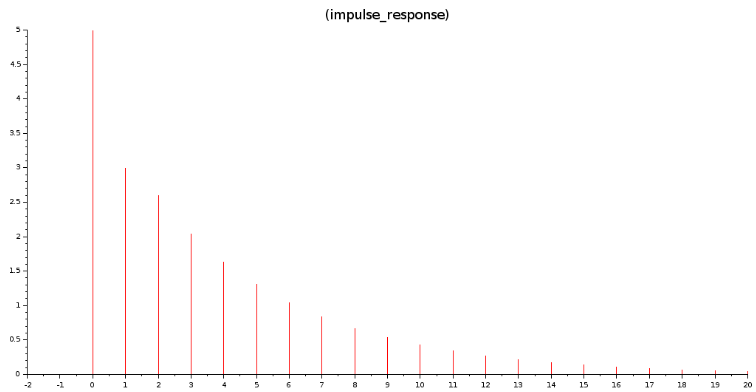

Figure 3.6: impulse response

```
    initial conditions + zero output array
x=zeros(length(n),1);x(3)=1;
for k=1:length(n)-2
  y(k+2) = 0.6*y(k+1) + 0.16*y(k)+5*x(k+2);
end;
plot2d3(n,y,[5]);title("(impulse_response)","
    fontsize",4);
```

Scilab code Exa 3.13 discrete convolution

```
clc
clear
close

function y=u(x)
     y=sign(sign(x)+1);
endfunction

```

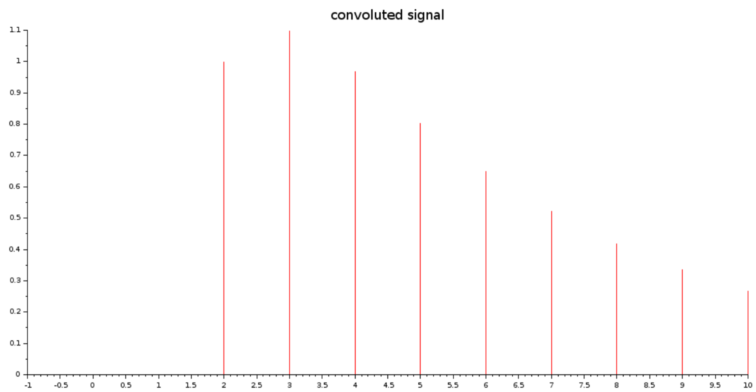

Figure 3.7: discrete convolution

```
n=-1:10;
x=((0.8).^n).*u(n);
g=((0.3).^n).*u(n);

c=convol2d(x,g);//discrete convolution

plot2d3(0:length(c)-1,c,[5]);
zoom_rect([-1 min(c) 10 max(c)]);
title("convoluted signal","fontsize",4)
```

Scilab code Exa 3.14 `Zero state response of discrete systems`

```
clc
clear
close

n=[0:20];
x=(4.^(-n));
```

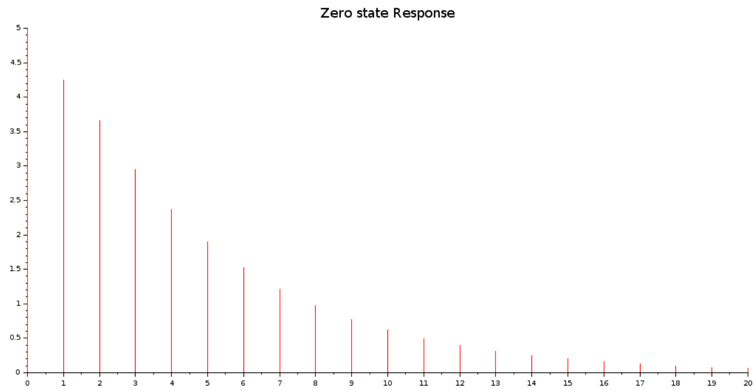

Figure 3.8: Zero state response of discrete systems

```
h=((-0.2).^n+4*(0.8).^n);
c=convol(x,h);//discrete convolution

plot2d3(n,c(1:length(n)),[5]);title("Zero state
    Response","fontsize",4);
```

Scilab code Exa 3.15 `discrete convolution`

```
clc
clear
close

function y=u(x)
    y=sign(sign(x)+1);
endfunction

n=[0:6];
x=((0.8).^n).*u(n);
```

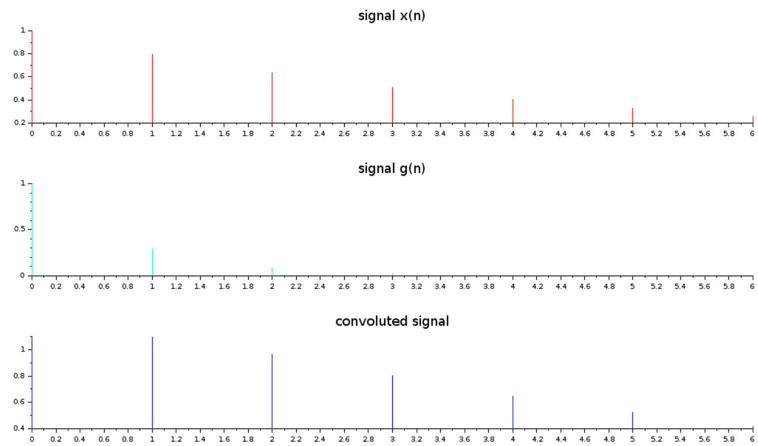

Figure 3.9: discrete convolution

```
g=((0.3).^n).*u(n);
c=convol2d(x,g);//discrete convolution

subplot(3,1,1);plot2d3(n,x,[5]);title("signal x(n)",
   "fontsize",4);
subplot(3,1,2);plot2d3(n,g,[4]);title("signal g(n)",
   "fontsize",4);
subplot(3,1,3);plot2d3(n,c(1:length(n)),[2]);title("
   convoluted signal","fontsize",4);
```

Scilab code Exa 3.16 `sliding tape discrete convolution`

```
clc
clear
close

function y=u(x)
    y=sign(sign(x)+1);
```

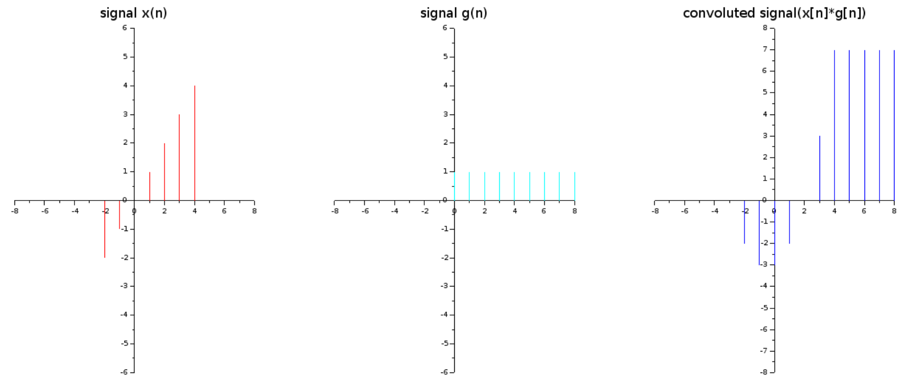

Figure 3.10: sliding tape discrete convolution

```
endfunction

n=[-2:15];
x=n.*(u(n+2)-u(n-5));
g=1.*u(n);
c=convol2d(x,g);//discrete convolution

subplot(1,3,1);plot2d3(n,x,[5]);title("signal x(n)","fontsize",4);//plotting x[n]
set(gca(),"zoom_box",[-8 -6 8 6],"x_location","middle","y_location","middle");
subplot(1,3,2);plot2d3(n,g,[4]);title("signal g(n)","fontsize",4);//plotting g[n]
set(gca(),"zoom_box",[-8 -6 8 6],"x_location","middle","y_location","middle");
subplot(1,3,3);plot2d3([-4:length(c)-5],c,[2]);title("convoluted signal(x[n]*g[n])","fontsize",4);//plotting convoluted signal
set(gca(),"zoom_box",[-8 -8 8 8],"x_location","middle","y_location","middle");
```

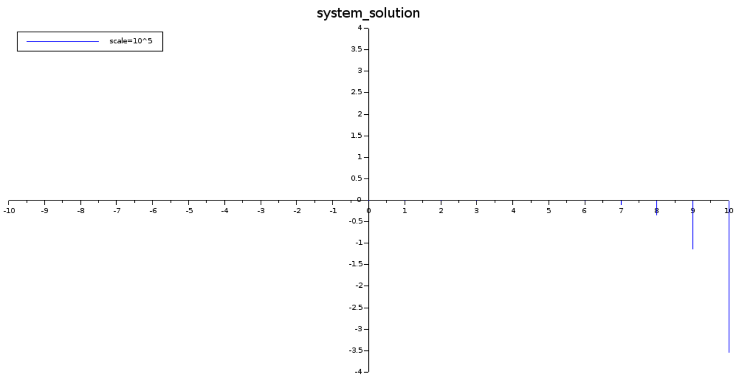

Figure 3.11: system response to auxiliary conditions

Scilab code Exa 3.17 `system response to auxiliary conditions`

```
clc
clear
close

n=[0:10]';//defining the discrete time
y= [4;13;zeros(length(n)-2,1)];//predefining the initial conditions + zero output array
x=(3*n+5);
for k=1:length(n)-2
y(k+2) = 5*y(k+1)-6*y(k)+x(k+1)-5*x(k);
end;

plot2d3(n,y/(10^5),[2]);title("system_solution","fontsize",4);
set(gca(),"zoom_box",[-10 -4 10 4],"x_location","
```

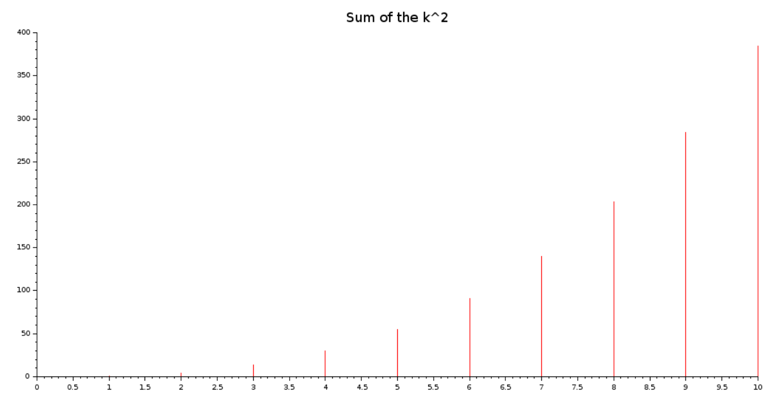

Figure 3.12: summation of given function

```
    middle","y_location","middle");
legend("scale=10^5","fontsize",2)
```

Scilab code Exa 3.18 summation of given function

```
clc
clear
close

n=[0:10]';
y= [0;zeros(length(n)-1,1)];//predefining the initial conditions + zero output array
x=(n+1).^2;
for k=1:length(n)-1
  y(k+1) =y(k)+x(k);
end;

plot2d3(n,y,[5]);title("Sum of the k^2","fontsize"
```

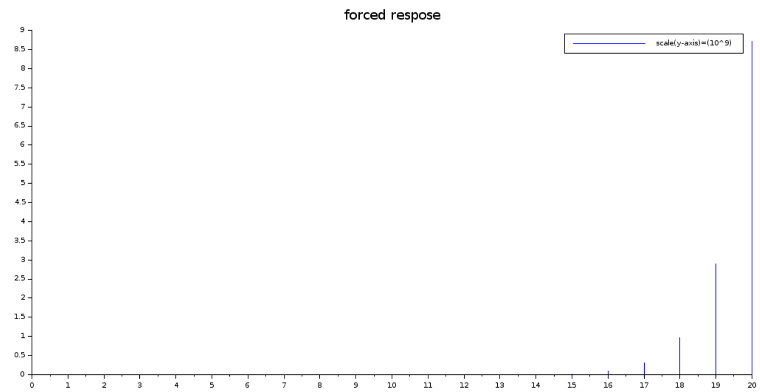

Figure 3.13: forced response

```
,4);
```

Scilab code Exa 3.19 `forced response`

```
clc ;
clear ;
close ;


n =0:20;
x=(3.^n);
a =[1 -3 2];
b =[0 1 2];
y= filter (b,a,x);
plot2d3 (n,y/(10^9),[2]);
title("forced respose","fontsize",4);
legend("scale(y-axis)=(10^9)")
```

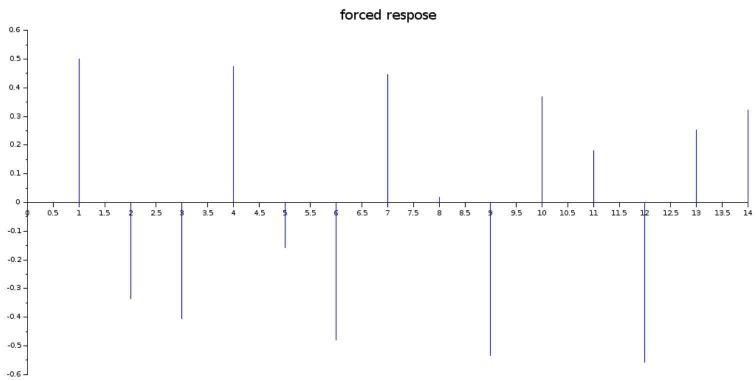

Figure 3.14: forced response

Scilab code Exa 3.20 `forced response`

```
clear ;
close ;
clc ;


n =0:14;
x= cos(2*n+%pi/3);//forcing function
a =[1 -1 0.16];
b =[0 1 0.32];
y= filter (b,a,x);
plot2d3 (n,y,[2]);
title("forced respose","fontsize",4);
set(gca(),"x_location","middle");
```

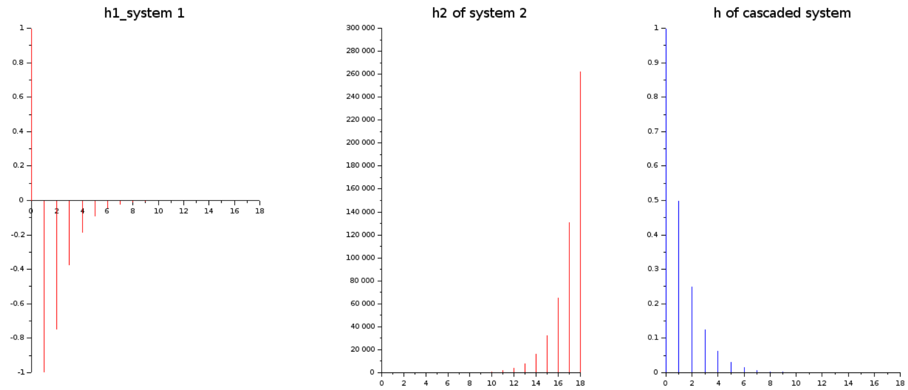

Figure 3.15: response of cascaded system

Scilab code Exa 3.21 `response of cascaded system`

```
clc
clear
close;

n=0:18;
h1=4*(n==0)-3*(0.5).^n;
h2=(2^n);
h_cas=conv(h1,h2);//response of cascaded system
subplot(1,3,1);plot2d3(n,h1,[5]);title("h1_system 1","fontsize",4);
set(gca(),"zoom_box",[0 -max(h1) 18 max(h1)],"x_location","middle");
subplot(1,3,2);plot2d3(n,h2,[5]);title("h2 of system 2","fontsize",4);
subplot(1,3,3);plot2d3(n,h_cas(1:length(n)),[2]);
```

```
title("h of cascaded system","fontsize",4);
```

Chapter 4

Laplace Transform

Scilab code Exa 4.1 Laplace transform of exponential signal

```
//scilab symbolic toolbox(scilab-scimax) is needed for this code
//symbolic toolbox should be strictly installed in Ubuntu-14.04 and Scilab-5.0.0 as it's not supported in higher versions
//use laplace.sci before this code using correct path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\laplace.sci', -1)
clc
Syms t s;
a = 3;
y =laplace('%e^(-a*t)',t,s);//finding laplace transform
t1=0:0.001:8;
plot(t1,exp(-a*t1),'r');
title("x(t)=e^-at","fontsize",4);
disp(y)
y1 = laplace('%e^(a*-t)',t,s);//finding laplace
```

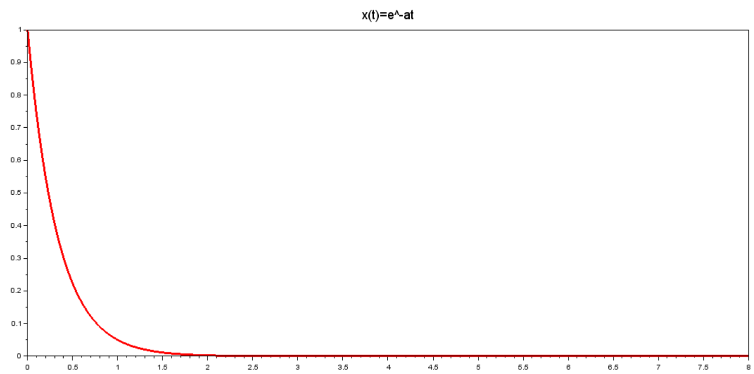

Figure 4.1: Laplace transform of exponential signal

```
   transform
disp(y1)
```

check Appendix AP 2 for dependency:

`laplace.sci`

Scilab code Exa 4.2 `Laplace transform of given signals`

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions

//use laplace.sci before this code using correct
   path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   laplace.sci', -1)

//(a) laplace transform of delta(t)
```

```
Syms t s;
y =laplace('0',t,s)
disp(y)

//(b) Laplace Transform of u(t)
y1 =laplace('1',t,s);
disp(y1)

//(c) laplace transform of cos(w0*t)u(t)
y2 =laplace('cos(w0*t)',t,s);
disp(y2)
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.3 Inverse Laplace transform of given transfer functions

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions


clc
//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
    inv_laplace.sci', -1)
//*******************************part(a)
    *********************
Syms t s;
[A]=pfss((7*s-6)/((s^2-s-6))); //partial fraction of
    F(s)
F1 = ilaplace(A(1),s,t)
```

```
F2 = ilaplace(A(2),s,t)
Fa = F1+F2;
disp(Fa,"fa(t)=")

//(***************************part(b)
   ************************
[A]=pfss((2*s^2+5)/((s^2-3*s+2))); //partial
   fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
Fb = F1+F2;
disp(Fb,"fb(t)=")

//*************************part(c)
   **************************
[A]=pfss((6*(s+34))/(s*(s^2+10*s+34))); //partial
   fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
Fc = F1+F2;
disp(Fc,"fc(t)=")
```

check Appendix AP 2 for dependency:

laplace.sci

Scilab code Exa 4.4 Laplace transform of given signals

```
clc
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions
//use laplace.sci before this code using correct
   path]
```

```
//exec('C:\Users\Satyajit\Desktop\sample_codes\laplace.sci', -1)
Syms t s;
a = 3;
T = 1;
y1 = laplace('t',t,s);
y2 = laplace('t',t,s);
y3 = laplace('1',t,s);
y=y1*(%e^(-s))+y2*(%e^(-2*s))+y3*(%e^(-4*s))
disp(y)
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.5 Inverse Laplace transform of given transfer function

```
//scilab symbolic toolbox(scilab-scimax) is needed for this code
//symbolic toolbox should be strictly installed in Ubuntu-14.04 and Scilab-5.0.0 as it's not supported in higher versions
//use inv_laplace.sci before this code using correct path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\inv_laplace.sci', -1)
clc

s1 =%s ;
Syms t s;
[A]=pfss((s1+3)/((s1+1)*(s1+2))); //partial fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
Fa = F1+F2;
disp(Fa,"f1(t)=")
```

```
[B]=pfss((5)/((s1+1)*(s1+2))); //partial fraction of
    F(s)
F1 = ilaplace(B(1),s,t)
F2 = ilaplace(B(2),s,t)
Fb = (F1+F2)*(%e^(-2*s));
disp(Fb,"f2(t)=")
disp(Fa+Fb,"f(t)=")
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.8 `Time convolution using laplace transform`

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions
//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
clc
Syms t s;
a=3;b=2;//taking random values of a and b
[A]=pfss(1/(s^2-5*s+6)); //partial fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F = F1+F2;
disp(F,"f(t)=")
```

Scilab code Exa 4.9 `Initial and final values of laplace transform`

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions
//limit can be used after installing symbolic
    toolbox only
clc

s=%s;
num =poly([30 20],'s','coeff')
den =poly([0 5 2 1],'s','coeff')
X = num/den;
disp (X,"X(s)=")
SX = s*X;
Initial_Value =limit(SX,s,%inf);
final_value =limit(SX,s,0);
disp(Initial_Value,"x(0)=")
disp(final_value,"x(inf)=")
```

---

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.10 `solving system equation`

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions

//use inv_laplace.sci before this code using correct
     path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
    inv_laplace.sci', -1)
```

```
clc
Syms t s;
[A] = pfss((2*s^2+20*s+45)/((s+2)*(s+3)*(s+4)));
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"y(t)=")
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.11 `Finding inductor current`

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions

//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
clc
Syms t s;
[A] = pfss((2*s)/(s^2+2*s+5));
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"inductor_current=")
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.12 Response of the given system

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions

//use inv_laplace.sci before this code using correct
     path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
    inv_laplace.sci', -1)
clc
Syms t s;
[A] = pfss((3*s+3)/((s+5)*(s^2+5*s+6)));
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"system_solution=")
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.15 Finding loop current

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions
```

```
//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
clc

Syms s t;
[A] = pfss((10)/(s^2+3*s+2));
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"Loop_current=");
```

check Appendix AP 2 for dependency:

`laplace.sci`

Scilab code Exa 4.16 Finding loop currents

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions

//use laplace.sci before this code using correct
   path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   laplace.sci', -1)

clc
Syms t s;
y1 =laplace('24*%e^(-3*t)+48*%e^(-4*t)',t,s);
disp(y1,"y1=")
```

```
y2 =laplace('16*%e^(-3*t)-12*%e^(-4*t)',t,s);
disp(y2,"y2")
```

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.17 Finding loop current

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions

//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
clc
Syms t s;
[A] = pfss((2*s^2+9*s+4)/((s)*(s^2+3*s+1)));
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"output")
```

Scilab code Exa 4.23 frequency response of the given system

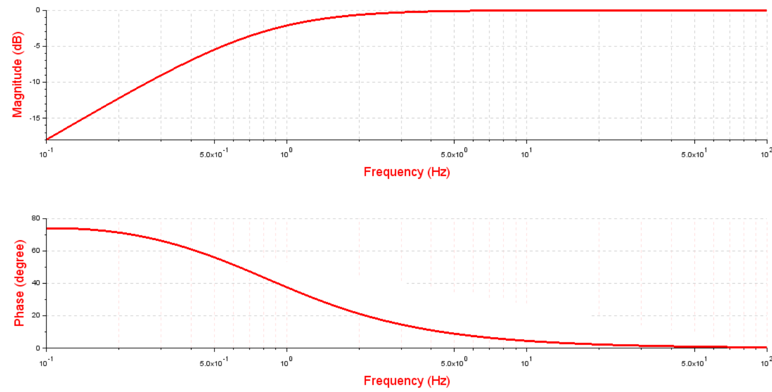

Figure 4.2: frequency response of the given system

```
clc
clear
close;

s=poly(0,'s');//polynomial definition
h=syslin('c',(s+0.1)/(s+5));//transfer function
bode(h,0.1,100);//plotting frequency response
```

Scilab code Exa 4.24 `frequency response of the given systems`

```
clc
clear
close;

s=poly(0,'s')//polynomial definition

h=syslin('c',(s^2/s))
figure(1)
bode(h,0.1,100);

h1=syslin('c',(1/s))//polynomial definition
```

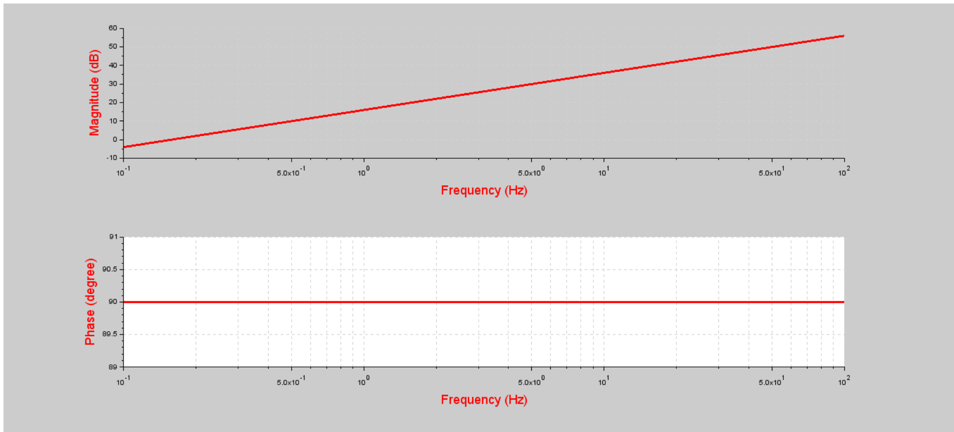

Figure 4.3: frequency response of the given systems

```
figure(2)
bode(h1,0.1,100);
```

Scilab code Exa 4.25 `bode plots of the given transfer function`

```

clc
clear
close;

s=%s;
h=syslin('c',((20*s^2+2000*s)/(s^2+12*s+20)))//transfer function
bode(h,0.1,100);
```

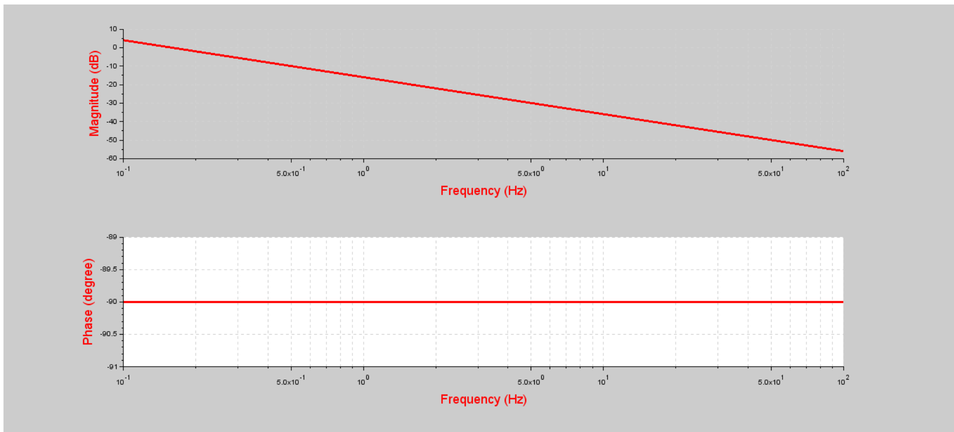

Figure 4.4: frequency response of the given systems

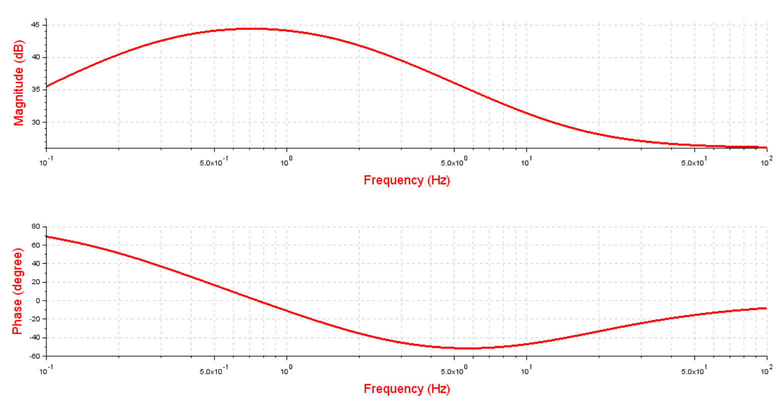

Figure 4.5: bode plots of the given transfer function

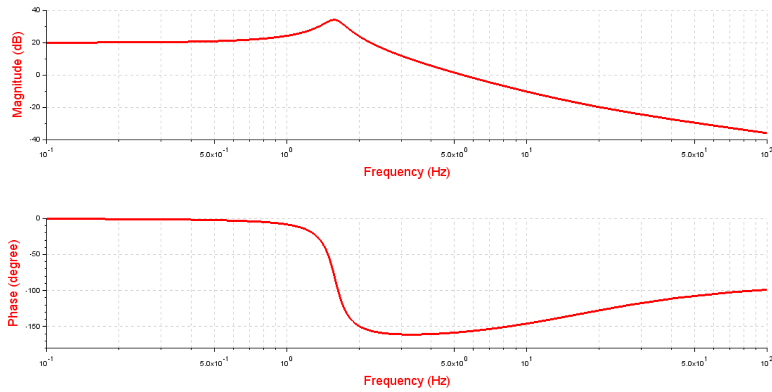

Figure 4.6: bode plots of the given transfer function

Scilab code Exa 4.26 `bode plots of the given transfer function`

```

clc
clear
close;

s=%s;
h=syslin('c',((10*s+1000)/(s^2+2*s+100)))//transfer
    function
bode(h,0.1,100);
```

Scilab code Exa 4.27 `second order notch filter analysis`

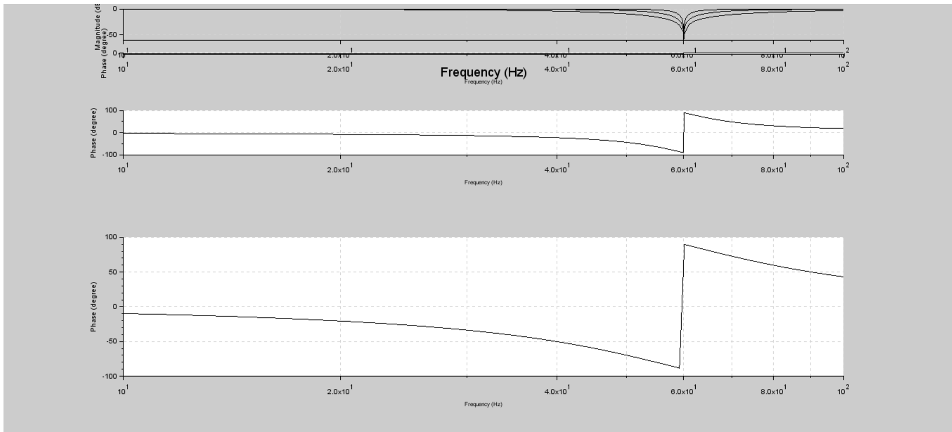

Figure 4.7: second order notch filter analysis

```
clc
clear
close

omega_0=2*%pi*60; theta = [60 80 87]*(%pi/180);
omega = (0:0.5:1000)';
mag = zeros(3,length(omega));
s=poly(0,'s')
figure(1)
for m =1:length(theta)
   H=syslin('c',((s^2+omega_0^2)/(s^2+2*omega_0*cos(
      theta(m))*s +omega_0^2)));
   bode(H,10,100);
end
```

---

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.28 `Inverse Laplace transform of given transfer function`

```
//scilab symbolic toolbox(scilab-scimax) is needed
    for this code
//symbolic toolbox should be strictly installed in
    Ubuntu-14.04 and Scilab-5.0.0 as it's not
    supported in higher versions

//use inv_laplace.sci before this code using correct
     path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
    inv_laplace.sci', -1)

clc

s=%s;
t=%t;
//*************part(a)************************
[A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F=F1+F2;
disp(F,"f(t)=")

//******************part(b)*****************
[A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F = -F1-F2;
disp(F,"f(t)=")

//****************part(c)*******************
[A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F = -F1+F2;
disp(F,"f(t)=")
```

check Appendix AP 1 for dependency:

inv_laplace.sci

Scilab code Exa 4.29 Finding loop current in RC circuit

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
//symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab -5.0.0 as it's not
   supported in higher versions

//use inv_laplace.sci before this code using correct
    path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
clc

Syms s t;
[A] = pfss((-s)/((s-1)*(s-2)*(s+1)));//partial
   fraction of transfer function
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F3 = ilaplace(A(3),s,t)
F = F1+F2+F3
disp(F,"F=");
```

check Appendix AP 1 for dependency:

inv_laplace.sci

Scilab code Exa 4.30 System response from transfer function

```
//scilab symbolic toolbox(scilab-scimax) is needed
   for this code
```

```
//symbolic toolbox should be strictly installed in Ubuntu-14.04 and Scilab-5.0.0 as it's not supported in higher versions

//use inv_laplace.sci before this code using correct path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\inv_laplace.sci', -1)
clc

Syms s t;
[A] = pfss((-1)/((s-1)*(s+2)));//partial fraction of transfer function
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F = F1+F2
disp(F)
```

---

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 4.31 System response from transfer function

```
//scilab symbolic toolbox(scilab-scimax) is needed for this code
//symbolic toolbox should be strictly installed in Ubuntu-14.04 and Scilab-5.0.0 as it's not supported in higher versions

//use inv_laplace.sci before this code using correct path]
//exec('C:\Users\Satyajit\Desktop\sample_codes\inv_laplace.sci', -1)
clc
Syms t s;
```

```
//for  Re s>-1
[A] = pfss(1/((s+1)*(s+5)));//partial fraction of
    transfer function
F1 = ilaplace(A(1),s,t)
F2 = ilaplace(A(2),s,t)
F = F1+F2
disp(F,"for Re(s)>-1")

//for -5< Re s <-2
[B] = pfss(-1/((s+2)*(s+5)));//partial fraction of
    transfer function
G1 = ilaplace(B(1),s,t)
G2 = ilaplace(B(2),s,t)
G = G1+G2
disp(G,"for -5<Re(s)<-2")
```

Chapter 5

Discrete time analysis using Z transform

Scilab code Exa 5.1 `Z transform of the given signal`

```
//This code needs symbolic tool(scilab-scimax) to be
    instaslled
//The tool should be installed properly in Ubuntu
   -14.04 and scilab -5.5.0 as it may not work in
   higher versions

clc
Syms n z;
a = 0.5;
x =(a)^n;
n1=0:10;
plot2d3(n1,a^n1); xtitle('a^n','n');
plot(n1,a^n1,'r.')
X = symsum(x*(z^(-n)),n,0,%inf)
disp(X,"ans=")
```

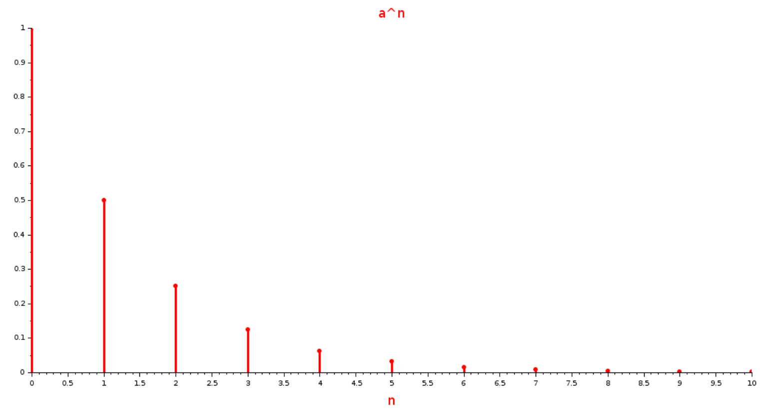

Figure 5.1: Z transform of the given signal

Scilab code Exa 5.2 Z transform of the given signals

```
//This code needs symbolic tool(scilab-scimax) to be
    instaslled
//The tool should be installed properly in Ubuntu
   -14.04 and scilab -5.5.0 as it may not work in
   higher versions

clc
Syms n z;
Wo =%pi/4;
a = (0.33)^n;
x1=%e^(sqrt(-1)*Wo*n);
X1=symsum(a*x1*(z^(-n)),n,0,%inf)
x2=%e^(-sqrt(-1)*Wo*n)
X2=symsum(a*x2*(z^(-n)),n,0,%inf)
X =(1/(2*sqrt(-1)))*(X1+X2)
disp(X,"ans1=")

```

```
//****************part(a)******************8
X=symsum(1*(z^(-n)),n,0,0);
disp(X,"ans2=")

//****************part(b)******************8
X=symsum(1*(z^(-n)),n,0,4);
disp(X,"ans3=")

//****************part(c)******************8
X=symsum(1*(z^(-n)),n,0,%inf);
disp(X,"ans4=")
```

Scilab code Exa 5.3 Inverse Z transform of the given transfer functions

```
//This code needs symbolic tool(scilab-scimax) to be instaslled
//The tool should be installed properly in Ubuntu-14.04 and scilab-5.5.0 as it may not work in higher versions
//limit function can be accessed after installing symbolic toolbox
clc

//********************************part(a)**********************
z = %z;
Syms n z1;//To find out Inverse z transform z must be linear z = z1
X  =(8*z-19)/((z-2)*(z-3))
X1 = X.den;              //don't use denom() or numer(), they'll be removed in upcoming versions
zp = roots(X1);
X1 = (8*z1-19)/((z1-2)*(z1-3))
F1 = X1*(z1^(n-1))*(z1-zp(1));
F2 = X1*(z1^(n-1))*(z1-zp(2));
```

```
h1 = limit(F1,z1,zp(1));
disp(h1,'h1[n]=')
h2 = limit(F2,z1,zp(2));
disp(h2,'h2[n]=')
h = h1+h2;
disp(h,'h[n]=')



//**********************************part(c)***********************
X  =(2*z*(3*z+17))/((z-1)*(z^2-6*z+25))
X1 = X.den;
zp = roots(X1);
X1 = 2*z1*(3*z1+17)/((z1-1)*(z1^2-6*z1+25))
F1 = X1*(z1^(n-1))*(z1-zp(1));
F2 = X1*(z1^(n-1))*(z1-zp(2));
h1 = limit(F1,z1,zp(1));
disp(h1,'h1[n]=')
h2 = limit(F2,z1,zp(2));
disp(h2,'h2[n]=')
h = h1+h2;
disp(h,'h[n]=')
```

Scilab code Exa 5.5 Solving given system equation using z transform

```
//This code needs symbolic tool(scilab-scimax) to be instaslled
//The tool should be installed properly in Ubuntu -14.04 and scilab -5.5.0 as it may not work in higher versions
//limit function can be accessed after installing symbolic toolbox

clc
```

```
Syms n z;
H1 = (26/15)/(z-(1/2));
H2 = (7/3)/(z-2);
H3 = (18/5)/(z-3);
F1 = H1*z^(n)*(z-(1/2));
F2 = H2*z^(n)*(z-2);
F3 = H3*z^(n)*(z-3);

h1 = limit(F1,z,1/2);
disp(h1,'h1[n]=')
h2 = limit(F2,z,2);
disp(h2,'h2[n]=')
h3 = limit(F3,z,3);
disp(h3,'h3[n]=')
h = h1-h2+h3;
disp(h,'h[n]=')
```

Scilab code Exa 5.6 solving system difference equation

```
//This code needs symbolic tool(scilab-scimax) to be
    instaslled
//The tool should be installed properly in Ubuntu
   -14.04 and scilab-5.5.0 as it may not work in
   higher versions
//limit function can be accessed after installing
   symbolic toolbox

clc
Syms n z;
H1 = (2/3)/(z+0.2);
H2 = (8/3)/(z+0.8);
H3 = (2)/(z+0.5);
F1 = H1*z^(n)*(z+0.2);
F2 = H2*z^(n)*(z+0.8);
F3 = H3*z^(n)*(z+0.5);
```

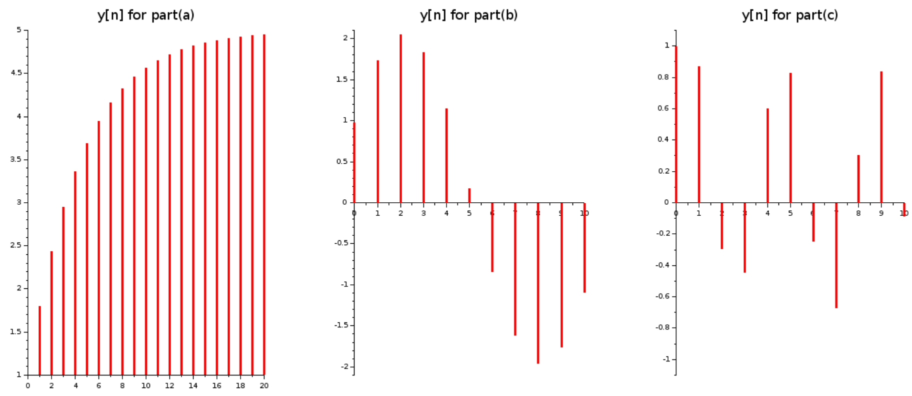

Figure 5.2: System response to different inputs

```

h1 = limit(F1,z,-0.2);
disp(h1,'h1[n]=')
h2 = limit(F2,z,-0.8);
disp(h2,'h2[n]=')
h3 = limit(F3,z,-0.5);
disp(h3,'h3[n]=')
h = h1-h2+h3;
disp(h,'h[n]=')
```

Scilab code Exa 5.10 System response to different inputs

```
clc ;
clear ;
close ;


n =0:20;
```

```
T=0.001;
x1=1.^n;
x2=cos(%pi/6*n-0.2);
x3=cos(1500*n*T);//sampled input
a =[1 -0.8];
b =[1 0];
y1= filter (b,a,x1);
y2= filter (b,a,x2);
y3= filter (b,a,x3);

subplot(1,3,1);plot2d3(n,y1,[5]);
title(”y[n] for part(a)”,”fontsize”,4);

subplot(1,3,2);plot2d3(n,y2,[5]);
title(”y[n] for part(b)”,”fontsize”,4);
set(gca(),”x_location”,”middle”,”zoom_box”,[0 -2.1
   10 2.1]);

subplot(1,3,3);plot2d3(n,y3,[5]);
title(”y[n] for part(c)”,”fontsize”,4);
set(gca(),”x_location”,”middle”,”zoom_box”,[0 -1.1
   10 1.1]);
```

Scilab code Exa 5.12 `Calculating maximum sampling interval or minimum frequency`

```
clc
clear

f=50*10^3;
T=0.5/f;
disp(1/(T*10^3),”the maximum sampling frequency in
   kHz=”)//in seconds
```

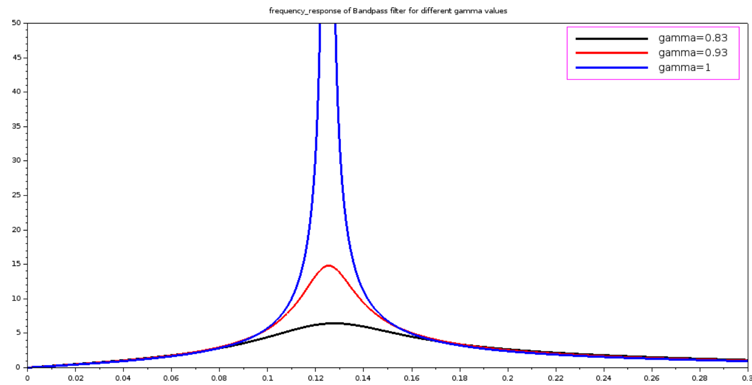

Figure 5.3: frequency response of Bandpass filter

Scilab code Exa 5.13 Calculating maximum sampling interval or minimum frequency

```
clc
clear

T=25*10^-6;
disp(1/(T*10^3),"the maximum sampling frequency in
    kHz=")//in seconds
```

Scilab code Exa 5.14 frequency response of Bandpass filter

```
clc
clear
close

gm=[0.83 0.93 1.00];
[xm1,fr1]=frmag([1 0 -1],[1 -sqrt(2)*gm(1) gm(1)
    ^2],4097);
```

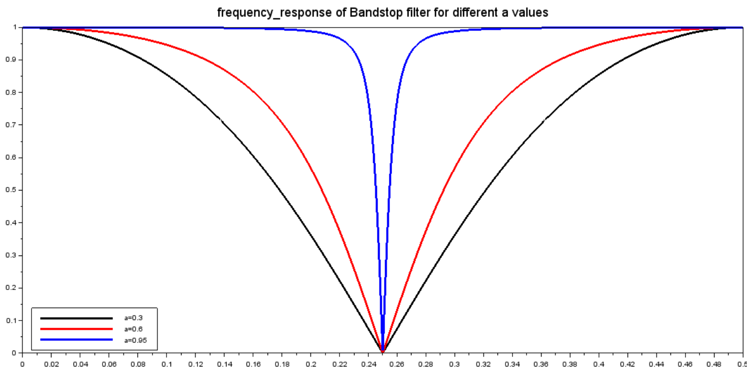

Figure 5.4: frequency response of Bandstop filter

```
[xm2,fr2]=frmag([1 0 -1],[1 -sqrt(2)*gm(2) gm(2)
    ^2],4097);
[xm3,fr3]=frmag([1 0 -1],[1 -sqrt(2)*gm(3) gm(3)
    ^2],4097);
plot(fr1,xm1,'k');
plot(fr2,xm2,'r');
plot(fr3,xm3,'b');
zoom_rect([0 0 0.3 50])
title("frequency_response of Bandpass filter for
    different gamma values","fontsize",4)
legend("gamma=0.83","gamma=0.93","gamma=1");
```

Scilab code Exa 5.15 `frequency response of Bandstop filter`

```
clc
clear
close

a=[0.3 0.6 0.95];
```

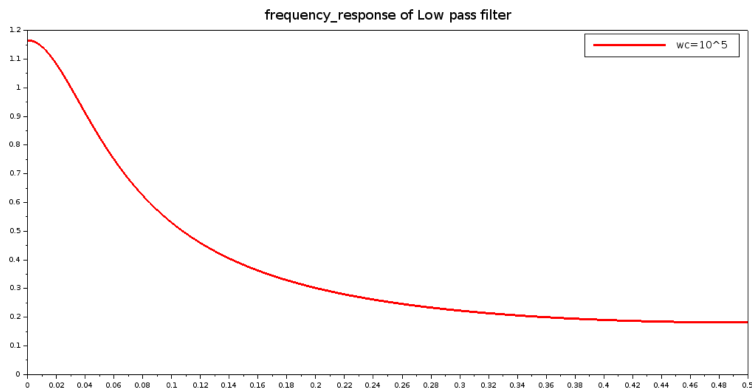

Figure 5.5: frequency response of low pass filter

```
[xm1,fr1]=frmag([1+a(1)^2 0 1+a(1)^2], [2 0 2*a(1)^2],4097);

[xm2,fr2]=frmag([1+a(2)^2 0 1+a(2)^2], [2 0 2*a(2)^2],4097);

[xm3,fr3]=frmag([1+a(3)^2 0 1+a(3)^2], [2 0 2*a(3)^2],4097);

plot(fr1,xm1,'k');
plot(fr2,xm2,'r');
plot(fr3,xm3,'b');
zoom_rect([0 0 0.5 1])
title("frequency_response of Bandstop filter for different a values","fontsize",4)
legend("a=0.3","a=0.6","a=0.95",[3]);
```

Scilab code Exa 5.16 `frequency response of low pass filter`

```
clc
clear
close

wc=10^5;
[xm1,fr1]=frmag([0.3142 0], [1 -0.7304],4097);//finding fr response from transfer functions

plot(fr1,xm1,'r');
zoom_rect([0 0 0.5 1.2])
title("frequency_response of Low pass filter","fontsize",4)
legend("wc=10^5");
```

Scilab code Exa 5.17 Z transform of the given signal

```
//This code needs symbolic tool(scilab-scimax) to be instaslled
//The tool should be installed properly in Ubuntu-14.04 and scilab-5.5.0 as it may not work in higher versions

clc
Syms n z;
a=0.9
b = 1.2;

x1=(a)^(n)
x2=(b)^(-n)
X1=symsum(x1*(z^(-n)),n,0,%inf)
X2=symsum(x2*(z^(n)),n,1,%inf)
X = X1+X2;
disp(X,"ans=")
```

Scilab code Exa 5.18 Inverse Z transform of the given function

```
//This code needs symbolic tool(scilab-scimax) to be instaslled
//The tool should be installed properly in Ubuntu-14.04 and scilab-5.5.0 as it may not work in higher versions
//limit function can be accessed after installing symbolic toolbox

clc
z = %z;
Syms n z1;


//*******************************part(a)*******************************
X  =-z*(z+0.4)/((z-0.8)*(z-2))
X1 = X.den;
zp = roots(X1);
X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2))
F1 = X1*(z1^(n-1))*(z1-zp(1));
F2 = X1*(z1^(n-1))*(z1-zp(2));
h1 = limit(F1,z1,zp(1));
disp(h1,'h1[n]=')
h2 = limit(F2,z1,zp(2));
disp(h2,'h2[n]=')
h = h1+h2;
disp(h,'h[n]=')

//*******************************part(b)*******************************
X  =-z*(z+0.4)/((z-0.8)*(z-2))
X1 = X.den;
```

```
zp = roots(X1);
X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2))
F1 = X1*(z1^(n-1))*(z1-zp(1));
F2 = X1*(z1^(n-1))*(z1-zp(2));
h1 = limit(F1,z1,zp(1));
disp(h1*'u(n)','h1[n]=')
h2 = limit(F2,z1,zp(2));
disp((h2)*'u(-n-1)','h2[n]=')
disp((h1)*'u(n)'-(h2)*'u(n-1)','h[n]=')

//******************************part(c)
    ******************************
X  =-z*(z+0.4)/((z-0.8)*(z-2))
X1 = X.den;
zp = roots(X1);
X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2))
F1 = X1*(z1^(n-1))*(z1-zp(1));
F2 = X1*(z1^(n-1))*(z1-zp(2));
h1 = limit(F1,z1,zp(1));
disp(h1*'u(-n-1)','h1[n]=')
h2 = limit(F2,z1,zp(2));
disp((h2)*'u(-n-1)','h2[n]=')
disp(-(h1)*'u(-n-1)'-(h2)*'u(-n-1)','h[n]=')
```

Scilab code Exa 5.19 ZSR of given system

```
//This code needs symbolic tool(scilab-scimax) to be
     instaslled
//The tool should be installed properly in Ubuntu
    -14.04 and scilab -5.5.0 as it may not work in
    higher versions
//limit function can be accessed after installing
    symbolic toolbox

clc
```

```
syms n z;
H1 = -z/(z-0.5);
H2 = (8/3)*z/(z-0.8);
H3=(-8/3)*z/(z-2);
F1 = H1*z^(n-1)*(z-0.5);
F2 = H2*z^(n-1)*(z-0.8);
F3 = H3*z^(n-1)*(z-2);
h1 = limit(F1,z,0.5);

disp(h1,'h1[n]=')
h2 = limit(F2,z,0.8);
disp(h2,'h2[n]=')
h3 = limit(F3,z,2);
disp(h3,'h3[n]=')
h = h1+h2+h3;
disp(h,'h[n]=')
```

Scilab code Exa 5.20 ZSR of given system

```
//This code needs symbolic tool(scilab-scimax) to be instaslled
//The tool should be installed properly in Ubuntu-14.04 and scilab-5.5.0 as it may not work in higher versions
//limit function can be accessed after installing symbolic toolbox

clc
Syms n z;
H1 = (-5/3)*z/(z-0.5);
H2 = (8/3)*z/(z-0.8);
H3=5*z/(z-0.5);
H4=-6*z/(z-0.6);
F1 = H1*z^(n-1)*(z-0.5);
F2 = H2*z^(n-1)*(z-0.8);
```

```
F3 = H3*z^(n-1)*(z-0.5);
F4 = H4*z^(n-1)*(z-0.6);
h1 = limit(F1,z,0.5);

disp(h1,'h1[n]=')
h2 = limit(F2,z,0.8);
disp(h2,'h2[n]=')
h3 = limit(F3,z,0.5);
disp(h3,'h3[n]=')
h4 = limit(F4,z,0.6);
disp(h4,'h4[n]=')
h = h1+h2+h3+h4;
disp(h,'h[n]=')
```

Chapter 6

Continuous time signal analysis The Fourier series

Scilab code Exa 6.1 Compact trigonometric Fourier series

```
clc
clear
close;

//finding fourier series
w0=2;   T0=2*%pi/w0;//period and frequency
dt=0.01;
t=0:dt:%pi-dt;
y=(exp(-t/2)).*((t>=0)&(t<%pi));
N=length(y);
c0=sum(y.*dt)/T0;
n1=[1:10]';
for n=1:10
  aa(n)=2*sum(y.*cos(n*w0*t).*dt)/T0;
  bb(n)=2*sum(y.*sin(n*w0*t).*dt)/T0;
  cn(n)=sqrt(aa(n).^2+bb(n).^2);
  thetan(n)=atan(-bb(n)/aa(n));
```

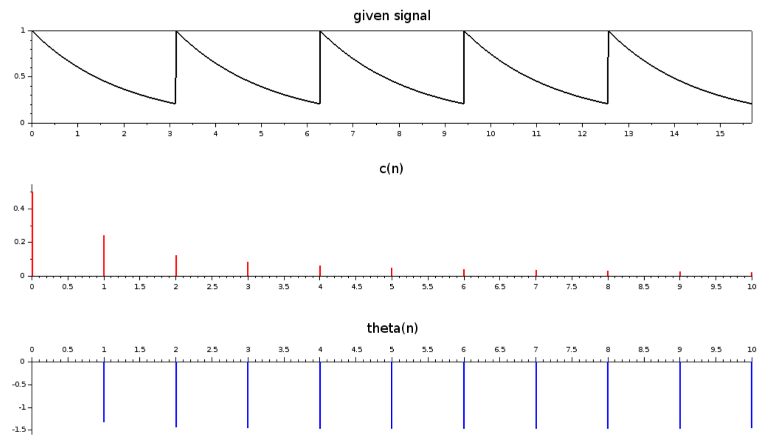

Figure 6.1: Compact trigonometric Fourier series

```
end

cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
y=repmat(y,1,5);t=0:dt:(length(y)-1).*dt;//periodic
    signal definition

subplot(3,1,1);plot(t,y,'r');title("given signal","
    fontsize",4);
set(gca(),"zoom_box",[t(1) 0 t($) 1]);
subplot(3,1,2);plot2d3(n,cn,[2]);title("c(n)","
    fontsize",4)
subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)
    ","fontsize",4)
set(gca(),"x_location","top");
```

Scilab code Exa 6.2 `Compact trigonometric Fourier series`

```
clc
```

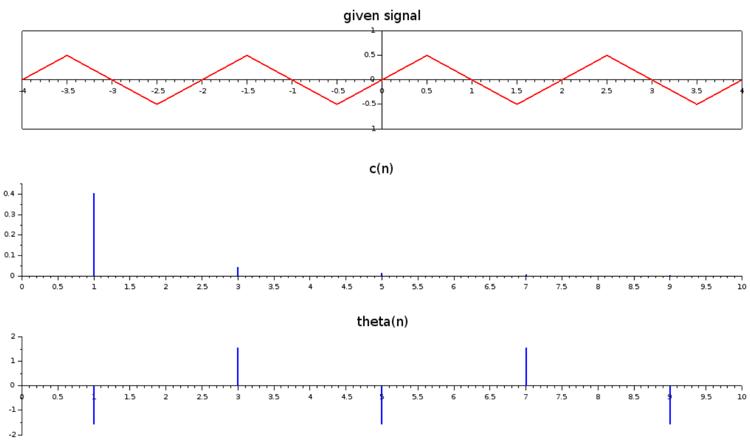

Figure 6.2: Compact trigonometric Fourier series

```
clear
close;
//error in phase might be there as atan finds angle only in 1st quadrant
//Taking A=0.5
w0=%pi;   T0=2;//period and frequency
dt=0.01;          //time increment
t=(-0.5:dt:1.5-dt);
y=(t.*((t>=-0.5)&(t<0.5)))+((-t+1).*((t>=0.5)&(t<1.5)));
N=length(y);
c0=sum(y.*dt)/T0;
n1=[1:10]';
for n=1:10
  aa(n)=2*sum(y.*cos(n*w0*t).*dt)/T0;
  bb(n)=2*sum(y.*sin(n*w0*t).*dt)/T0;
  cn(n)=sqrt(aa(n).^2+bb(n).^2);
  thetan(n)=atan(-bb(n)/aa(n));
end
thetan(7)=-thetan(7);
thetan(2*(1:5))=[0;0;0;0;0];//as even components are zero angles at even n are not needed
```

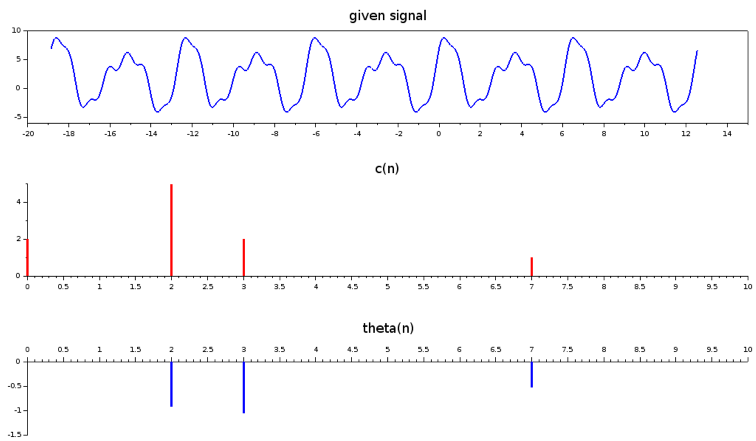

Figure 6.3: Compact trigonometric Fourier series

```
cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
y=repmat(y,1,5);t=-4.5:dt:(length(y)-1).*dt-4.5; // periodic signal definition

subplot(3,1,1);plot(t,y,'r');title("given signal","fontsize",4);
set(gca(),"zoom_box",[-4 -1 4 1],"x_location","middle","y_location","middle");
subplot(3,1,2);plot2d3(n,cn,[2]);title("c(n)","fontsize",4)
subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)","fontsize",4)
set(gca(),"x_location","middle");
```

Scilab code Exa 6.3 Compact trigonometric Fourier series

```
clc
clear
```

```
close;
//error in phase might be there as atan finds angle only in 1st quadrant

w0=1;    T0=2*%pi/w0;//period and frequency
dt=0.01;           //time increment
t=(0:dt:T0-dt);
y=2+3*cos(2*t)+4*sin(2*t)+2*sin(3*t+%pi/6)-cos(7*t+150*%pi/180);
N=length(y);
c0=sum(y.*dt)/T0;
n1=[1:10]';
for n=1:10
  aa(n)=2*sum(y.*cos(n*w0*t).*dt)/T0;
  bb(n)=2*sum(y.*sin(n*w0*t).*dt)/T0;
  cn(n)=sqrt(aa(n).^2+bb(n).^2);
  thetan(n)=atan(-bb(n)/aa(n));
end

cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
y=repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0;//periodic signal definition

subplot(3,1,1);plot(t,y,'b');title("given signal","fontsize",4);
subplot(3,1,2);plot2d3(n,cn,[5]);title("c(n)","fontsize",4)
subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)","fontsize",4)
set(gca(),"x_location","top","zoom_box",[0 -1.5 10 0]);
```

Scilab code Exa 6.4 Compact trigonometric Fourier series

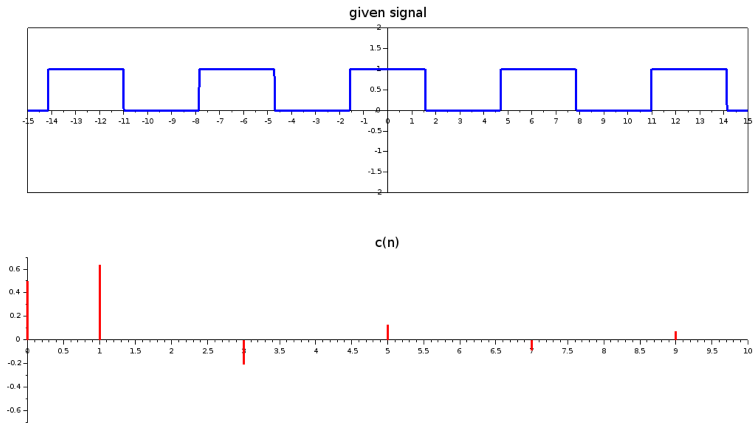

Figure 6.4: Compact trigonometric Fourier series

```
clc
clear
close;
//error in phase might be there as atan finds angle only in 1st quadrant

w0=1;T0=2*%pi/w0;//period and frequency
dt=0.01;           //time increment
t=(-%pi:dt:%pi-dt);
y=1.*((t>=-%pi/2)&(t<=%pi/2));
N=length(y);
c0=0.5;
n1=[1:10]';
for n=1:10
  dn(n)=sum(y.*exp(-%i*w0*n*t).*dt)/T0;
  cn(n)=2*dn(n);
end

cn=[c0;cn];n=[0;n1];
y=repmat(y,1,7);t=-3.5*T0:dt:(length(y)-1).*dt-3.5*T0;//periodic signal definition

```

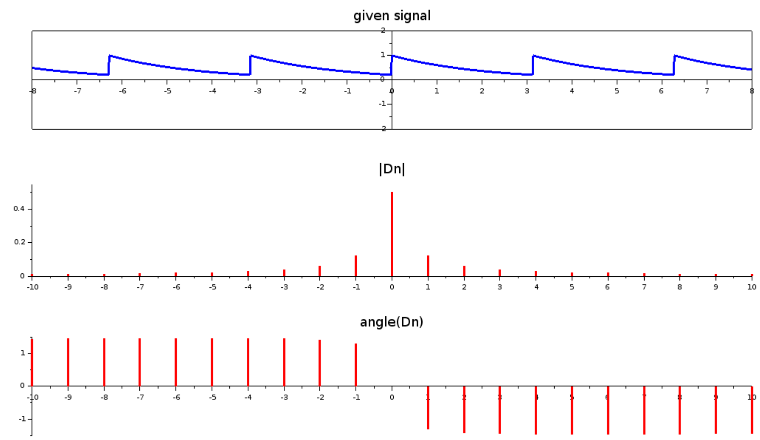

Figure 6.5: exponential Fourier series

```
subplot(2,1,1);plot(t,y,'b');
title("given signal","fontsize",4);
set(gca(),"zoom_box",[-15 -2 15 2],"x_location","middle","y_location","middle");
subplot(2,1,2);plot2d3(n,cn,[5]);title("c(n)","fontsize",4);
set(gca(),"x_location","middle","zoom_box",[0 -0.7 10 0.7]);
```

Scilab code Exa 6.5 `exponential Fourier series`

```
clc
clear
close;
//error in phase might be there as atan finds angle only in 1st quadrant

w0=2;T0=2*%pi/w0;//period and frequency
```

```
dt=0.01;          //time increment
t=(0:dt:%pi-dt);
y=exp(-t/2);
N=length(y);
p=1;
for n=-10:10
  dn(p)=sum(y.*exp(-%i*w0*n*t).*dt)/T0;
  mag_dn(p)=abs(dn(p));
  ang_dn(p)=phasemag(dn(p))*%pi/180;
  p=p+1;
end

n=-10:10;
y=repmat(y,1,7);t=-3*T0:dt:(length(y)-1).*dt-3*T0;//periodic signal definition

subplot(3,1,1);plot(t,y,'b');
title("given signal","fontsize",4);
set(gca(),"zoom_box",[-8 -2 8 2],"x_location","middle","y_location","middle");
subplot(3,1,2);plot2d3(n,mag_dn,[5]);title("     |Dn|          ","fontsize",4)
subplot(3,1,3);plot2d3(n,ang_dn,[5]);title("angle(Dn)","fontsize",4);
set(gca(),"x_location","middle");
```

Scilab code Exa 6.6 exponential Fourier series from trigonometric Fourier series

```
clc
clear
close;


n=-12:3:12;
function cn=c(n)             //defining given cn
```

```
    cn=(16-(4/3)*n).*(n>=0);
endfunction
function thetan=theta(n)   //defining given theta_n
    thetan=((-%pi/12)*n).*((n>=0)&(n<=6))+((%pi/12)*(n-12)).*(n>6);
endfunction

dn=(c(n)+c(-n))/2;
d_theta=(theta(n)-theta(-n));
figure(1)
subplot(1,2,1);plot2d3(n,c(n));title("C_n given","fontsize",4)
subplot(1,2,2);plot2d3(n,theta(n));title("Theta_n given","fontsize",4)
set(gca(),"x_location","top");
figure(2)
subplot(1,2,1);plot2d3(n,dn);title("|Dn|_calculated","fontsize",4)
subplot(1,2,2);plot2d3(n,d_theta);title("Theta(Dn)_calculated","fontsize",4)
set(gca(),"x_location","middle");
```

Scilab code Exa 6.7 trigonometric Fourier series of impulse train

```
clc
clear
close;
//error in phase might be there as atan finds angle only in 1st quadrant
```

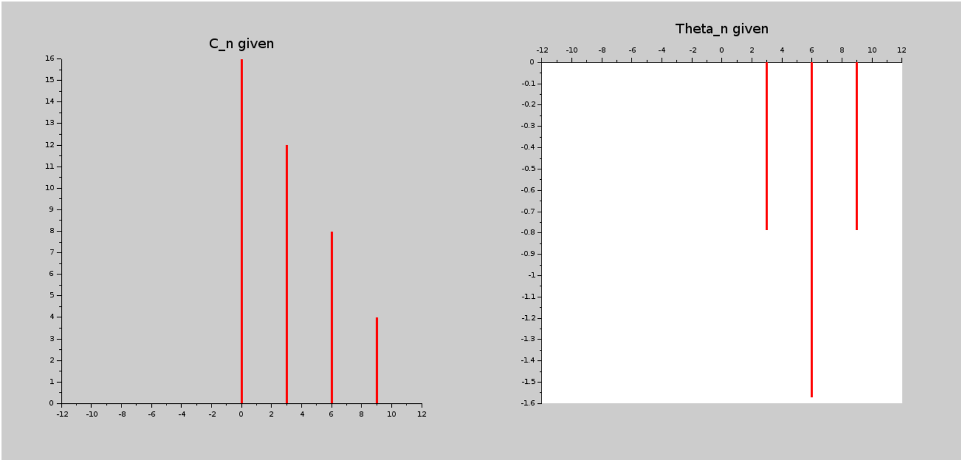

Figure 6.6: exponential Fourier series from trigonometric Fourier series

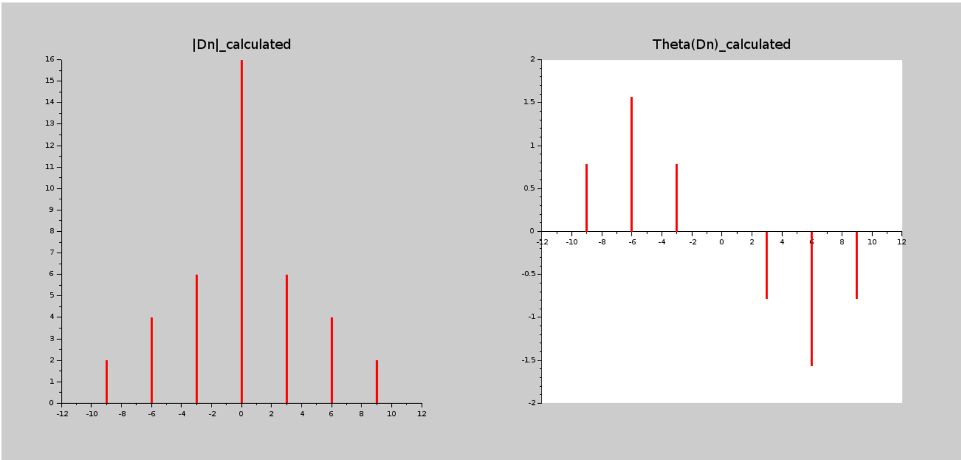

Figure 6.7: exponential Fourier series from trigonometric Fourier series

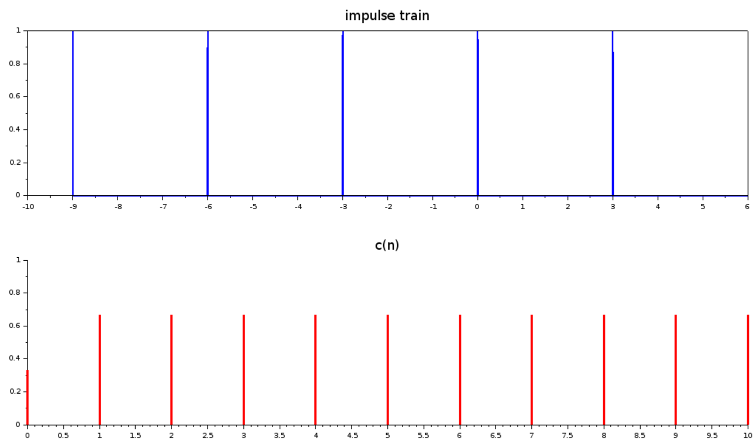

Figure 6.8: trigonometric Fourier series of impulse train

```

//taking period of 3
T0=3;
dt=0.01;          //time increment
t=(0:dt:T0-dt);
y=1.*(t==0);//definiing impulse function
c0=1/T0;
n1=[1:10]';
for n=1:10
    cn(n)=2/T0;
end

cn=[c0;cn];n=[0;n1];
y=repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0;//periodic signal definition

subplot(2,1,1);plot(t,y,'b');title("impulse train","fontsize",4);
subplot(2,1,2);plot2d3(n,cn,[5]);title("c(n)","fontsize",4);
set(gca(),"zoom_box",[0 0 10 1]);
```

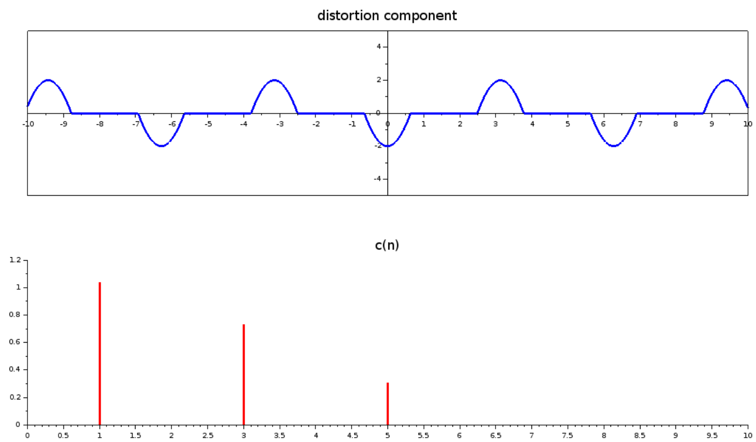

Figure 6.9: Harmonic distortion in audio amplifier

Scilab code Exa 6.8 `Harmonic distortion in audio amplifier`

```
clc
clear
close;
//error in phase might be there as atan finds angle only in 1st quadrant

w0=1;    T0=2*%pi/w0;//assuming frequency 1Hz
dt=0.01;          //time increment
t=(-T0/2:dt:T0/2-dt);
y=(10*cos(w0*t)-8).*((t>=-0.1024*T0)&(t<=0.1024*T0))+(10*cos(w0*t)+8).*((t<=(-0.5+0.1024)*T0)&(t>=-0.5*T0))+(10*cos(w0*t)+8).*((t>=(0.5-0.1024)*T0)&(t<=0.5*T0));//distortion component
N=length(y);
c0=sum(y.*dt)/T0;
```

```
n1=[1:10]';
for n=1:10
    dn(n)=sum(y.*exp(-%i*n*w0*t).*dt)/T0;
    cn(n)=2*dn(n);
end

cn=[c0;cn];n=[0;n1];
y=repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0;//periodic signal definition
subplot(2,1,1);plot(t,y,'b');title("distortion component","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -5 10 5]);
subplot(2,1,2);plot2d3(n,cn,[5]);title("c(n)","fontsize",4)
set(gca(),"zoom_box",[0 0 10 1.2]);

//distortion calculation
en=50;//energy of undistortecd signal
P=integrate("(10*cos(w0*t)-8).^2","t",0,0.1024*T0)*4/T0;//totaql harmonic power
P_3rd=((abs(cn(4))).^2)/2;            // power of third harmonic
D_tot=(P/en)*100;
D_3rd=(P_3rd/en)*100;
printf("\nTotal hatrmonic distortion is %.2f percent\n 3rd harmonic distortion is %.2f percent\n",D_tot,D_3rd);
//round-off error may be there
```

Scilab code Exa 6.9 dc value and rms value of rectified signal

```
clc
```

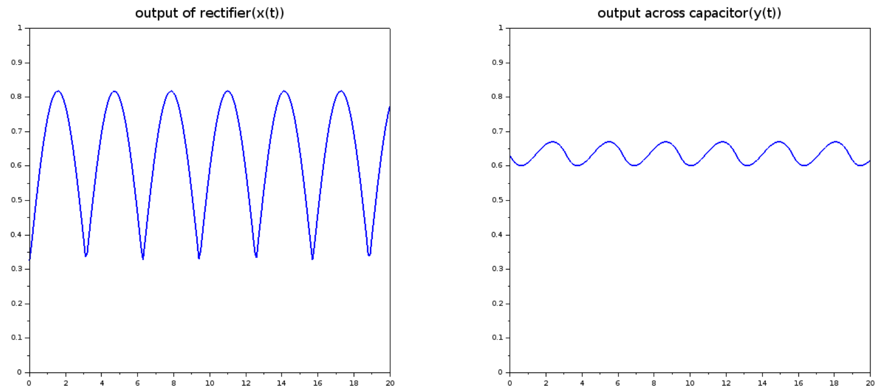

Figure 6.10: dc value and rms value of rectified signal

```
clear;
close;

t=0:0.1:20;
pi=%pi;
n=[0:20];
p=1;
//finding x(t) and y(t) from fourier series co-efficients
for t1=t
    cx=0;
    cy=0;
    for n1=n
        cx=cx+(2/(pi*(1-4*n1.^2)))*exp(%i*2*n1*t1);
        cy=cy+(2/(pi*(1-4*n1.^2)*(1+6*%i*n1)))*exp(%i*2*n1*t1);
    end
    x(p)=cx;
    y(p)=cy;
    p=p+1;
end
subplot(1,2,1);plot(t',x);title("output of rectifier
```

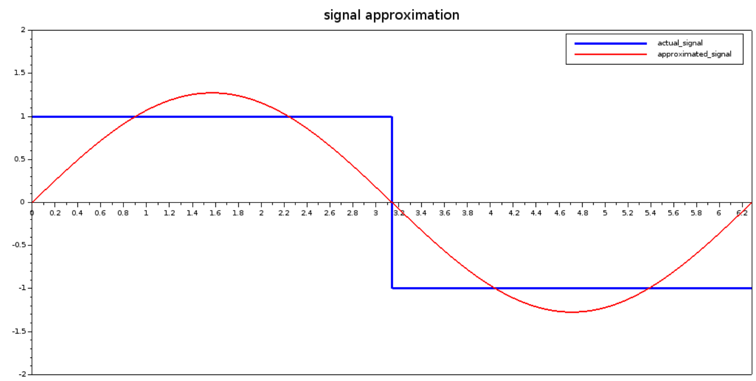

Figure 6.11: Signal approximation with minimum error signal

```
    (x(t))","font_size",4);
subplot(1,2,2);plot(t',y);title("output across
    capacitor(y(t))","font_size",4)
zoom_rect([t(1) 0 t($) 1]);
//finding ripple
DC=2/%pi;
power=0;
for n=1:50
    Dn(n)=2/(pi*(1-4*n.^2)*(1+6*%i*n));
    power=power+2*sum((abs(Dn(n))).^2);
end
rms=sqrt(power);
printf("\n DC value of output is %.2f and ripple(rms
    ) is %.2f\n",DC,rms);
```

Scilab code Exa 6.10 `Signal approximation with minimum error signal`

```
clc
```

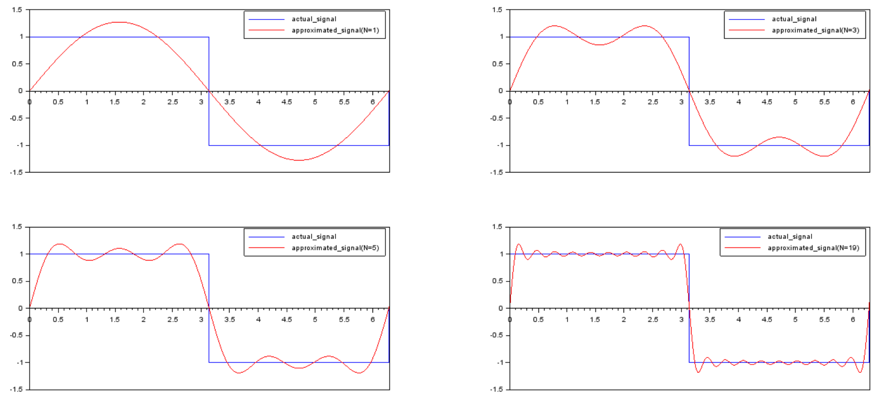

Figure 6.12: Regeneration of signal from Fourier series coefficient

```
clear
close

Ey=integrate("sin(t).^2","t",0,2*%pi);//energy of x(
    t)
C=(1/Ey).*integrate("sin(t).*(1.*((t>=0)&(t<=%pi))
    -1.*((t>%pi)&(t<=2*%pi)))","t",0,2*%pi);
printf("\napproximated signal is x(t) ~ %.3f*sin(t)\
    n",C);
t=0:0.001:2*%pi;
x=1.*((t>=0)&(t<=%pi))-1.*((t>%pi)&(t<=2*%pi));
x_approx=(4/%pi).*sin(t);
plot(t,x,'b');plot(t,x_approx,'r');
title("signal approximation","fontsize",4);
legend("actual_signal","approximated_signal");
set(gca(),"x_location","middle","zoom_box",[t(1) -2
    t($) 2]);
```

Scilab code Exa 6.11 Regeneration of signal from Fourier series coefficient

```
clc
clear all
close;

t=0:0.001:6.3;
x=1.*((t>=0)&(t<%pi))-1.*((t>=%pi)&(t<=(2*%pi)));
p=[1,3,5,19];

x_approx=0;
for n=1:2:p(1)
    x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
end
subplot(2,2,1);plot(t',x,'b');plot(t,x_approx,'r');
set(gca(),"x_location","middle","zoom_box",[t(1) -1.5 t($) 1.5]);
legend("actual_signal","approximated_signal(N=1)");

x_approx=0;
for n=1:2:p(2)
    x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
end
subplot(2,2,2);plot(t',x,'b');plot(t,x_approx,'r');
set(gca(),"x_location","middle","zoom_box",[t(1) -1.5 t($) 1.5]);
legend("actual_signal","approximated_signal(N=3)");

x_approx=0;
for n=1:2:p(3)
    x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
end
subplot(2,2,3);plot(t',x,'b');plot(t,x_approx,'r');
set(gca(),"x_location","middle","zoom_box",[t(1) -1.5 t($) 1.5]);
legend("actual_signal","approximated_signal(N=5)");

x_approx=0;
for n=1:2:p(4)
    x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
```

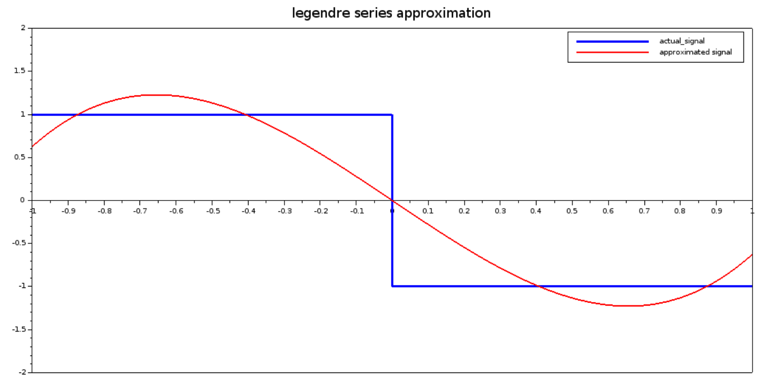

Figure 6.13: Legendre Fourier series

```
end
subplot(2,2,4);plot(t',x,'b');plot(t,x_approx,'r');
set(gca(),"x_location","middle","zoom_box",[t(1)
    -1.5 t($) 1.5]);
legend("actual_signal","approximated_signal(N=19)");

//Finding Error
E1=2*%pi-(16/%pi);//for N=1
E3=2*%pi-(16/%pi)*(1+1/9);//for N=3
print(E1)
print(E3)
```

Scilab code Exa 6.12 `Legendre Fourier series`

```
clc
clear all
close;

```

```
t=-1:0.001:1;
x=1.*((t>=-1)&(t<0))-1.*((t>=0)&(t<=1));
x_approx=(-3/2)*t+(7/8)*(2.5*t.^3-1.5*t);
plot(t',x,'b');plot(t,x_approx,'r');title("legendre series approximation","fontsize",4)
set(gca(),"x_location","middle","zoom_box",[t(1) -2 t($) 2]);
legend("actual_signal","approximated signal");
```

Chapter 7

Continuous time signal analysis The Fourier transform

Scilab code Exa 7.1 `Fourier transform of given signal`

```
clc
clear
close;
//phase error will be there due to round-off error

t=0:0.009:2;
dt=0.009;
x=exp(-2*t);
omega=-20:0.009:20;
p=0;
for om=-20:0.009:20
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
subplot(1,3,1);plot(t,x,'k');
title("given signal-(e^-2t)*u(t)","fontsize",4);
```

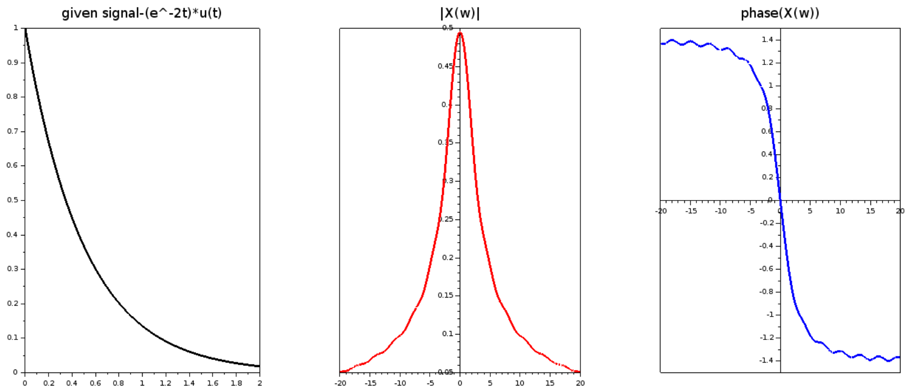

Figure 7.1: Fourier transform of given signal

```
subplot(1,3,2);plot(omega,abs(X),'r');
title("|X(w)|","fontsize",4);
set(gca(),"y_location","middle");
subplot(1,3,3);plot(omega,phasemag(X)*%pi/180,'b');
set(gca(),"x_location","middle","y_location","middle
    ");
title("phase(X(w))","fontsize",4);
```

Scilab code Exa 7.2 Fourier transform of given rect signal

```
clc
clear
close;
//taking tau=2
t=-4:0.009:4;
dt=0.009;
x=1.*((t>=-1)&(t<=1));
omega=-20:0.009:20;
p=0;
for om=-20:0.009:20
```

```
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
figure(1)
subplot(1,2,1);plot(t,x,'k');
title("given signal-rect(t/2)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-4 -2 4 2]);
subplot(1,2,2);plot(omega,X,'r');
title("X(w)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-20 -2 20 2]);

figure(2)
subplot(1,2,1);plot(omega,abs(X),'r');
title("|X(w)|","fontsize",4);
subplot(1,2,2);plot(omega,-phasemag(X)*%pi/180,'r');
title("phase(X(w))","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-20 -4 20 4]);
```

Scilab code Exa 7.3 Fourier transform of given impulse signal

```
clc
clear
close;

t=-2:0.001:2;
```

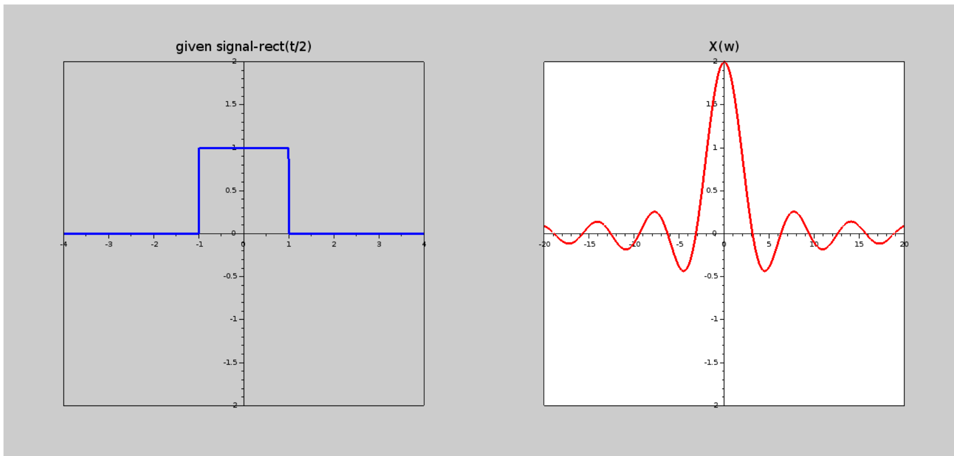

Figure 7.2: Fourier transform of given rect signal

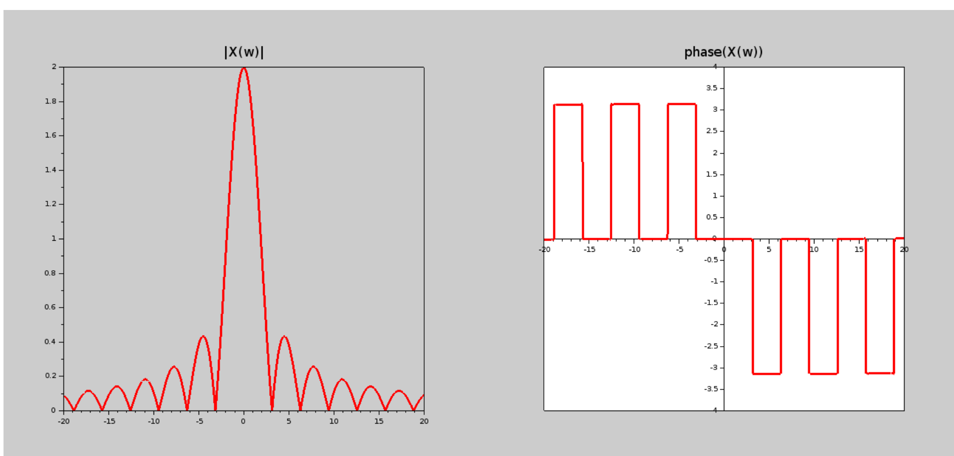

Figure 7.3: Fourier transform of given rect signal

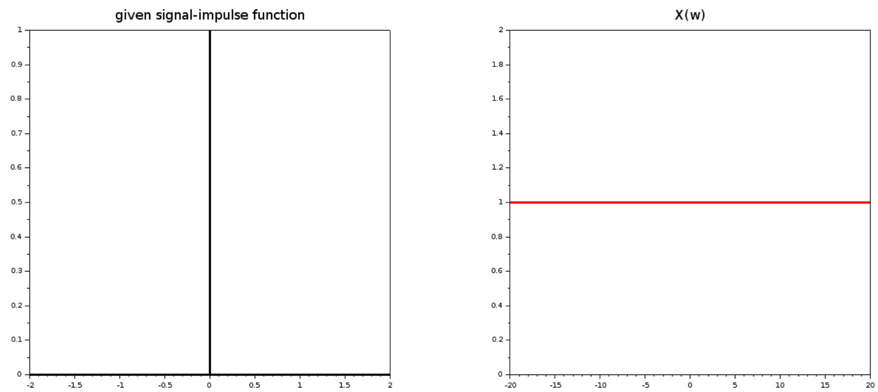

Figure 7.4: Fourier transform of given impulse signal

```
dt=0.001;
x=1.*(t==0);
omega=-20:0.001:20;
X=ones(1,length(omega));
subplot(1,2,1);plot(t,x,'k');
title("given signal-constant function","fontsize",4);
subplot(1,2,2);plot(omega,abs(X),'r');
title("X(w)","fontsize",4);
```

Scilab code Exa 7.4 `Inverse fourier transform of impulse signal`

```
clc
clear
close;

t=0:0.001:2;
dt=0.001;
```

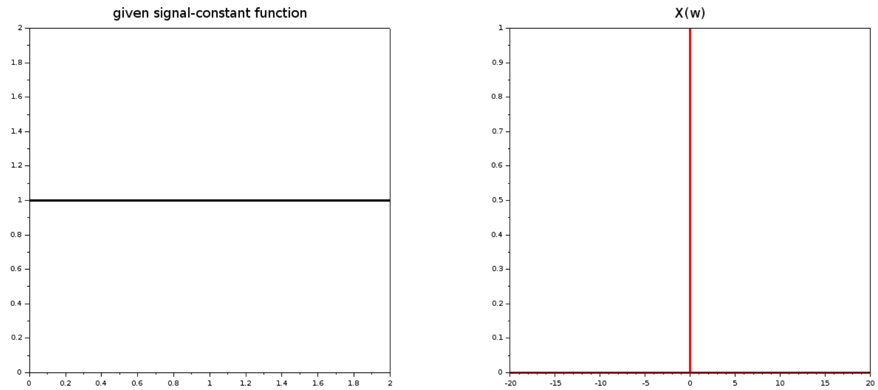

Figure 7.5: Inverse fourier transform of impulse signal

```
x=ones(1,length(t));
omega=-20:0.001:20;
X=1.*(omega==0);
subplot(1,2,1);plot(t,x,'k');
title("given signal-constant function","fontsize",4);
subplot(1,2,2);plot(omega,abs(X),'r');
title("X(w)","fontsize",4);
```

Scilab code Exa 7.6 Fourier transform of given everlasting sinusoidal signal

```
clc
clear
close;
//taking wo=2*%pi

t=-8:0.01:8;
dt=0.01;
```

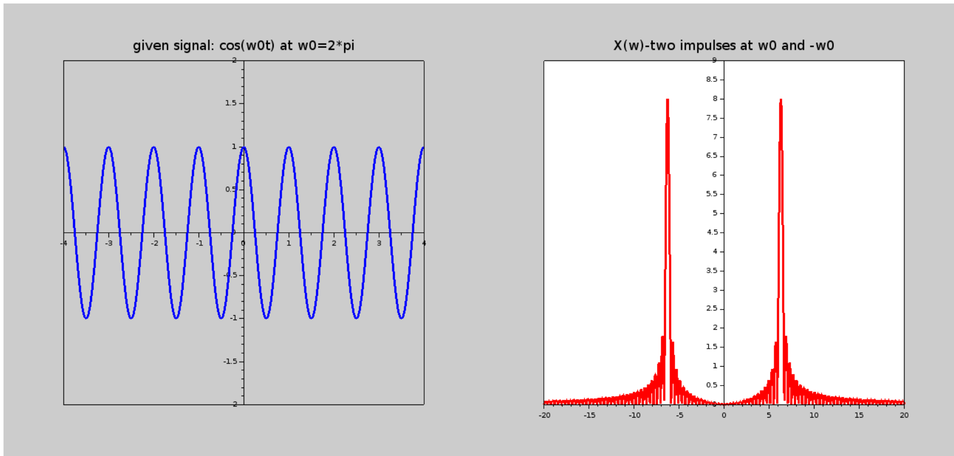

Figure 7.6: Fourier transform of given everlasting sinusoidal signal

```
x=cos(2*%pi*t);
omega=-20:0.01:20;
p=0;
for om=-20:0.01:20
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
figure(1)
subplot(1,2,1);plot(t,x,'b');
title("given signal: cos(w0t) at w0=2*pi","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-4 -2 4 2]);
subplot(1,2,2);plot(omega,abs(X),'r');
title("X(w)-two impulses at w0 and -w0","fontsize",4);
set(gca(),"y_location","middle");
```

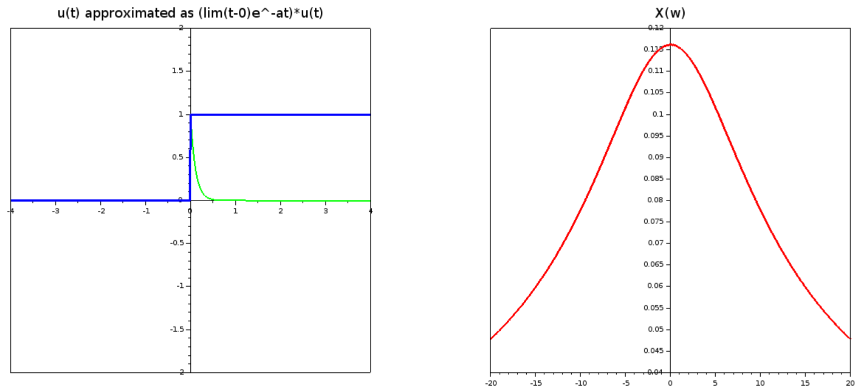

Figure 7.7: Fourier transform of given unit step signal

Scilab code Exa 7.9 `Fourier transform of given unit step signal`

```
clc
clear
close;

t=-4:0.01:4;
dt=0.01;
x=exp(-9*t).*(t>=0);
x1=1.*(t>=0);
omega=-20:0.01:20;
p=0;
for om=-20:0.01:20
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
subplot(1,2,1);plot(t,x,'b');plot(t,x1,'g');
title("u(t) approximated as (lim(t-0)e^-at)*u(t)","
```

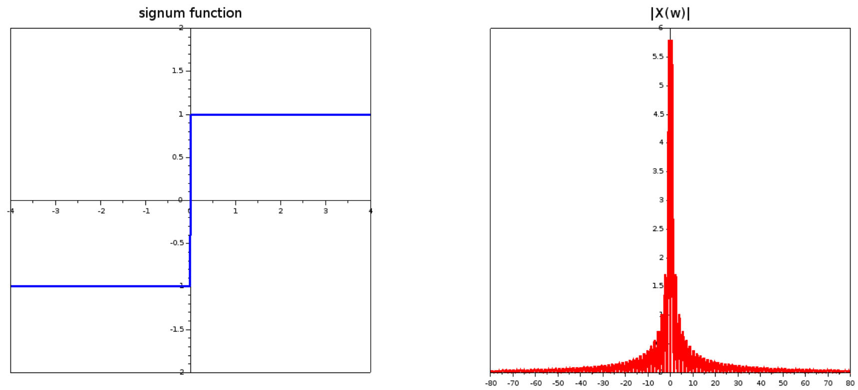

Figure 7.8: Fourier transform of given Signum function

```
   fontsize",4);
set(gca(),"x_location","middle","y_location","middle
   ","zoom_box",[-4 -2 4 2]);
subplot(1,2,2);plot(omega,abs(X),'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle");
```

Scilab code Exa 7.10 `Fourier transform of given Signum function`

```
//the result is the outer covering of the obtained
   plot
clc
clear
close;

t=-4:0.01:4;
dt=0.01;
x=sign(t);
```

```
omega=-80:0.01:80;
p=0;
for om=-80:0.01:80
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
subplot(1,2,1);plot(t,x,'b');
title("signum function","fontsize",4);
set(gca(),"x_location","middle","y_location","middle
   ","zoom_box",[-4 -2 4 2]);
subplot(1,2,2);plot(omega,abs(X),'r');
title("|X(w)|","fontsize",4);
set(gca(),"y_location","middle");
//here absolute value of X(w) is shown
//if angle is included then right side part of fig 2
    will be downwards
```

Scilab code Exa 7.11 Duality property

```
//taking tau=2
clc
clear
close;

t=-20:0.01:20;
dt=0.01;
x1=1.*((t>=-1)&(t<=1));
x2=2*sinc(t);
omega=-20:0.01:20;
p=0;
for om=-20:0.01:20
  p=p+1;
  X1(p)=sum(x1.*exp(-%i*om*t).*dt);
  X2(p)=sum(x2.*exp(-%i*om*t).*dt);
```

```
end
X1=round(X1*10000)/10000;
X2=round(X2*10000)/10000;
figure(1)
subplot(1,2,1);plot(t,x1,'b');
title("x(t): rect(t/2)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -2 10 2]);
subplot(1,2,2);plot(omega,X1,'r');
title("X(w): 2*sinc(w)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[omega(1) -2 omega($) 2]);
figure(2)
subplot(1,2,1);plot(t,x2,'b');
title("x(t): 2*sinc(t)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[t(1) -2 t($) 2]);
subplot(1,2,2);plot(omega,X2,'r');
title("X(w): 2*pi*rect(w/2)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -7 10 7]);
```

Scilab code Exa 7.12 Fourier transform of given signals

```
//taking a=2
clc
clear
close;

t=-10:0.01:10;
dt=0.01;
```

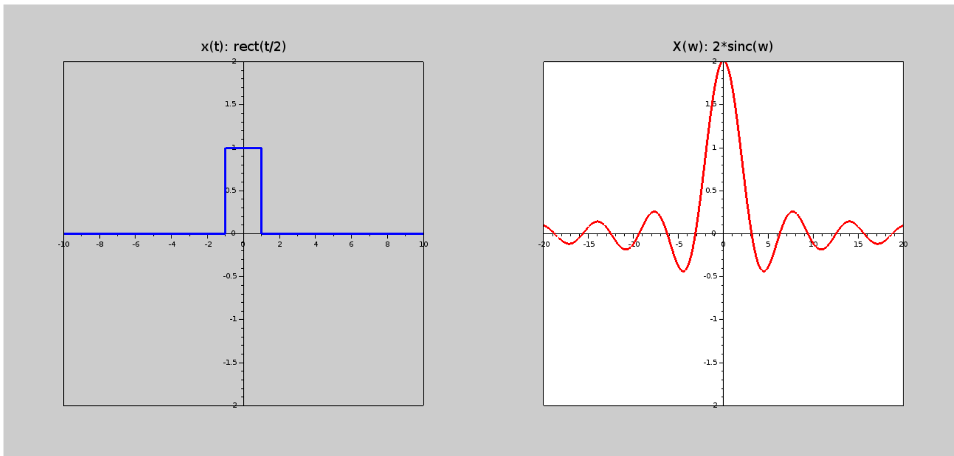

Figure 7.9: Duality property

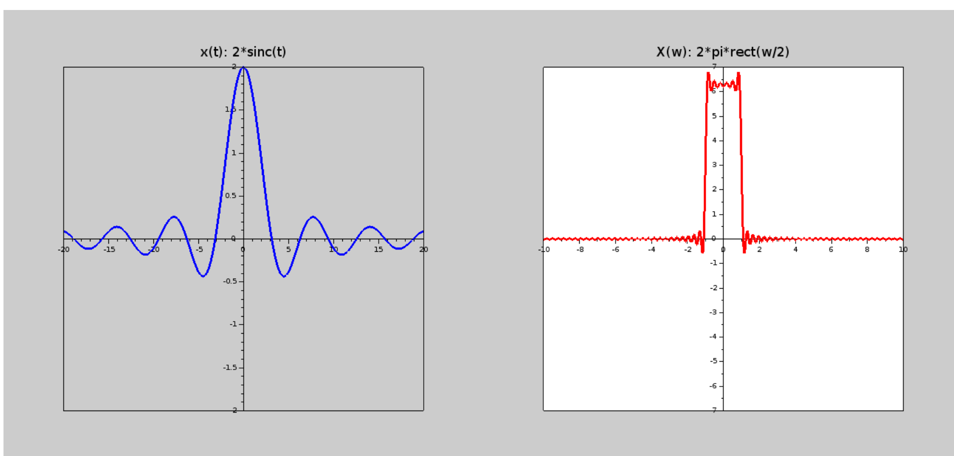

Figure 7.10: Duality property

```
x1=exp(2*t).*(t<=0);//defining e^at*u(-t)
x2=exp(-2*abs(t));//defining e^a|t|
omega=-10:0.01:10;
p=0;
for om=-10:0.01:10
  p=p+1;
  X1(p)=sum(x1.*exp(-%i*om*t).*dt);
  X2(p)=sum(x2.*exp(-%i*om*t).*dt);
end
X1=round(X1*10000)/10000;
X2=round(X2*10000)/10000;
figure(1)
subplot(1,2,1);plot(t,x1,'b');
title("x1(t): (e^at)*u(-t)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -1.2 10 1.2]);
subplot(1,2,2);plot(omega,X1,'r');
title("X1(w): 2*sinc(w)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -2 10 2]);
figure(2)
subplot(1,2,1);plot(t,x2,'b');
title("x2(t): e^-a|t|","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -1.2 10 1.2]);
subplot(1,2,2);plot(omega,X2,'r');
title("X2(w)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -2 10 2]);
```

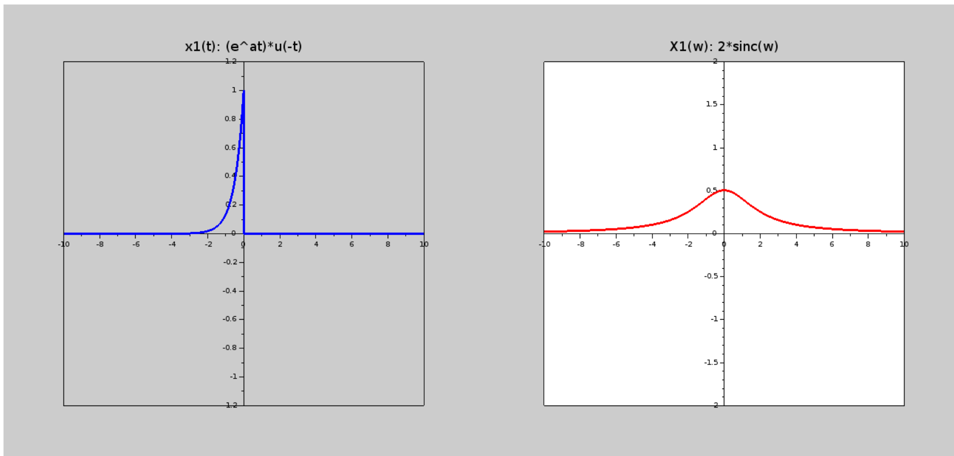

Figure 7.11: Fourier transform of given signals

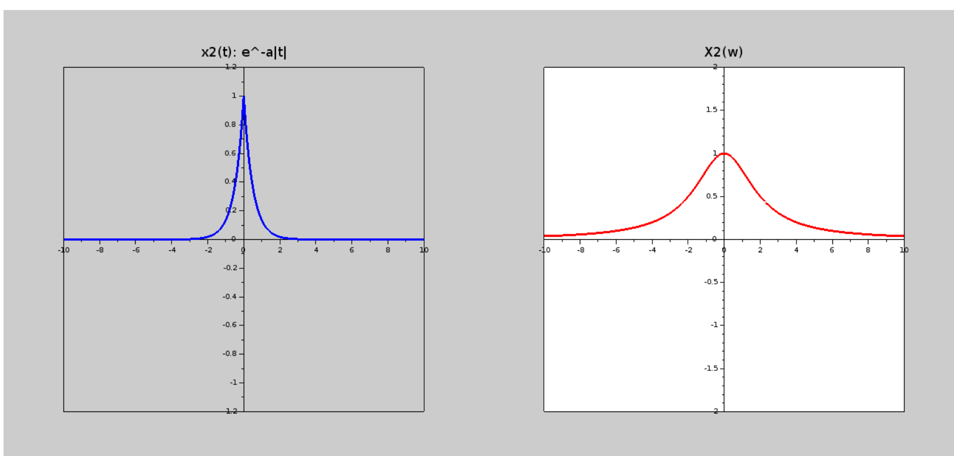

Figure 7.12: Fourier transform of given signals

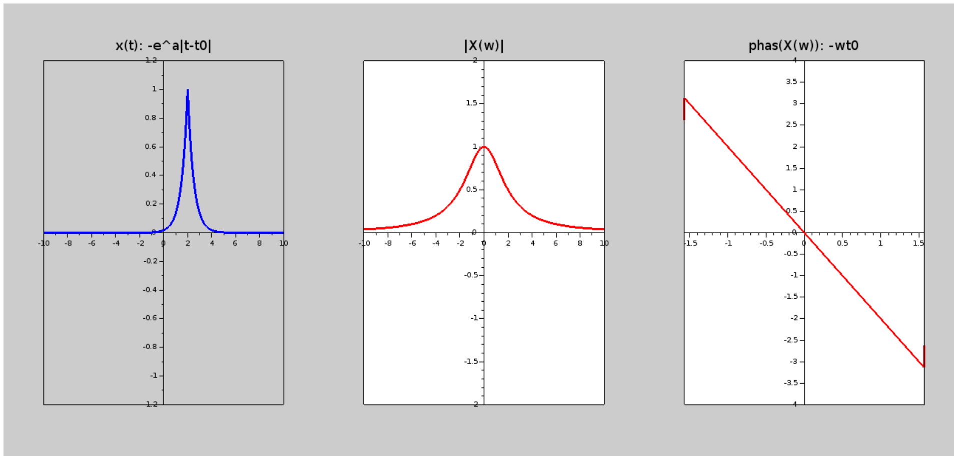

Figure 7.13: Fourier transform of time shifted signal

Scilab code Exa 7.13 `Fourier transform of time shifted signal`

```
//taking a=2,t0=2
clc
clear
close;

t=-10:0.01:10;
dt=0.01;
x=exp(-2*abs(t-2));//defining e^a|t|
omega=-10:0.01:10;
p=0;
for om=-10:0.01:10
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
figure(1)
subplot(1,3,1);plot(t,x,'b');
title("x(t): -e^a|t-t0|","fontsize",4);
set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -1.2 10 1.2]);
```

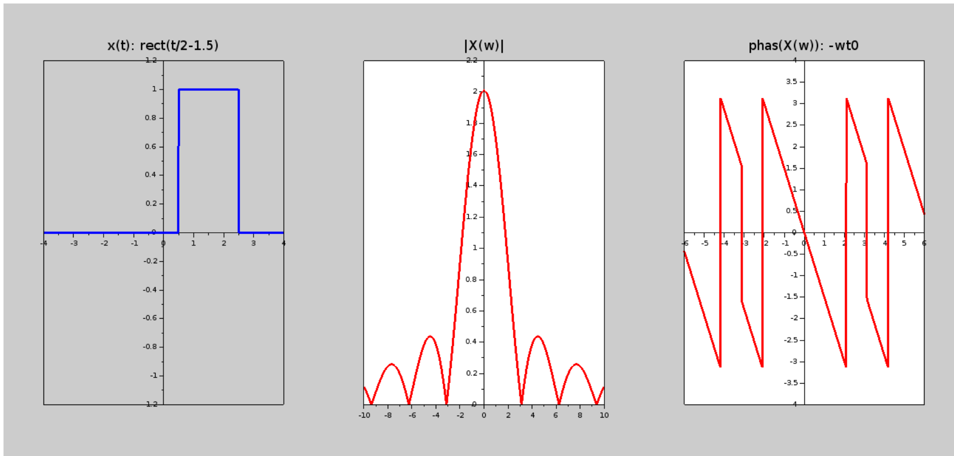

Figure 7.14: Fourier transform of time shifted rect signal

```
subplot(1,3,2);plot(omega,abs(X),'r');
title("|X(w)|","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-10 -2 10 2]);
subplot(1,3,3);plot(omega,phasemag(X)*%pi/180,'r');
title("phas(X(w)): -wt0","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-%pi/2 -4 %pi/2 4]);

//the angle is -wt0 but phasemag() will give angle in the range of [0:2*pi] only,hence the phase plot will be like -wt0 in the range[-pi/2:pi/2] and it will repeat thereafter
```

Scilab code Exa 7.14 `Fourier transform of time shifted rect signal`

```
//taking tau=2
clc
```

```
clear
close;

t=-10:0.01:10;
dt=0.01;
x=1.*((t>=0.5)&(t<=2.5));//rect(t/2) delayed by (3*tau/4)=1.5
omega=-10:0.01:10;
p=0;
for om=-10:0.01:10
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
ph=round(phasemag(X)*10000)/10000;
figure(1)
subplot(1,3,1);plot(t,x,'b');
title("x(t): rect(t/2-1.5)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-4 -1.2 4 1.2]);
subplot(1,3,2);plot(omega,abs(X),'r');
title("|X(w)|","fontsize",4);
set(gca(),"y_location","middle");
subplot(1,3,3);plot(omega,ph*%pi/180,'r');
title("phas(X(w)): -wt0","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-6 -4 6 4]);
```

Scilab code Exa 7.15 Fourier transform of modulated signal

```
clc
clear
close;
```

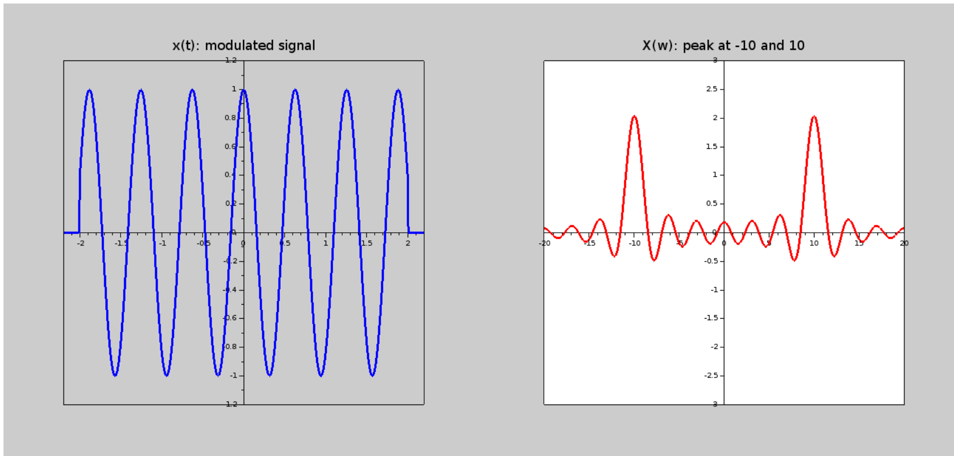

Figure 7.15: Fourier transform of modulated signal

```
t=-10:0.01:10;
dt=0.01;
x=(1.*((t>=-2)&(t<=2))).*cos(10*t);//modulated signal
omega=-30:0.01:30;
p=0;
for om=-30:0.01:30
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
figure(1)
subplot(1,2,1);plot(t,x,'b');
title("x(t): modulated signal","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-2.2 -1.2 2.2 1.2]);
subplot(1,2,2);plot(omega,X,'r');
title("X(w): peak at -10 and 10","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[-20 -3 20  3]);
```

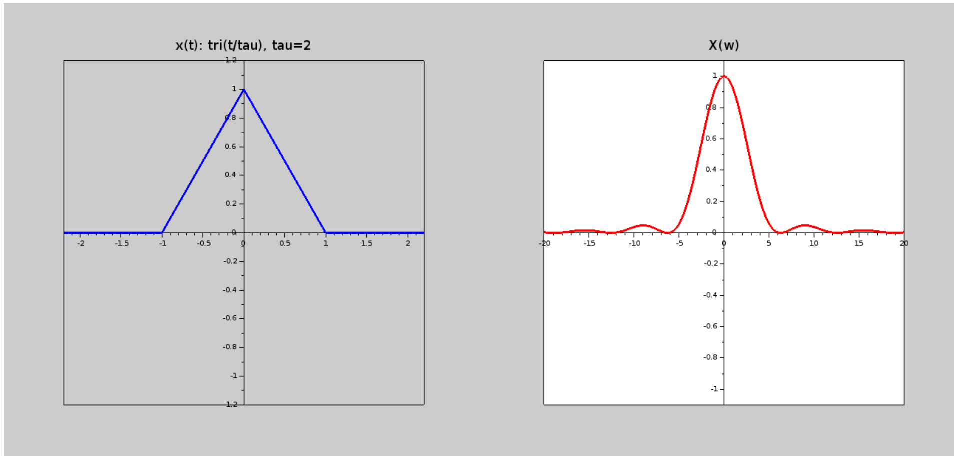

Figure 7.16: Fourier transform of given triangle pulse

Scilab code Exa 7.17 Fourier transform of given triangle pulse

```
//taking tau=2
clc
clear
close;

t=-3:0.01:5;
dt=0.01;
x=(t+1).*((t>=-1)&(t<=0))+(1-t).*((t>0)&(t<=1));
omega=-20:0.01:20;
p=0;
for om=-20:0.01:20
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
```

```
figure(1)
subplot(1,2,1);plot(t,x,'b');
title("x(t): tri(t/tau), tau=2","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-2.2 -1.2 2.2 1.2]);
subplot(1,2,2);plot(omega,X,'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[-20 -1.1 20  1.1]);
```

Scilab code Exa 7.19 Filtered signals

```
clc
clear
close;

//********************part a ******************************
//defining x(t)
function x=p(t)
    x=cos(10*%pi*(t-0.05)).*((t>=0)&(t<=0.1));
endfunction

t=-0.12:0.0009:0.12;
wc=2000*%pi;
x=p(t);
z=x.*cos(wc*t);
tg=10^-3;
y=2*p(t-tg).*cos(wc*(t-tg)-0.4*%pi);

figure(1)
subplot(1,3,1);plot(t,x,'b');
title("x(t)","fontsize",4);
set(gca(),"zoom_box",[t(1) -2 t($) 2],"x_location","middle","y_location","middle");
```

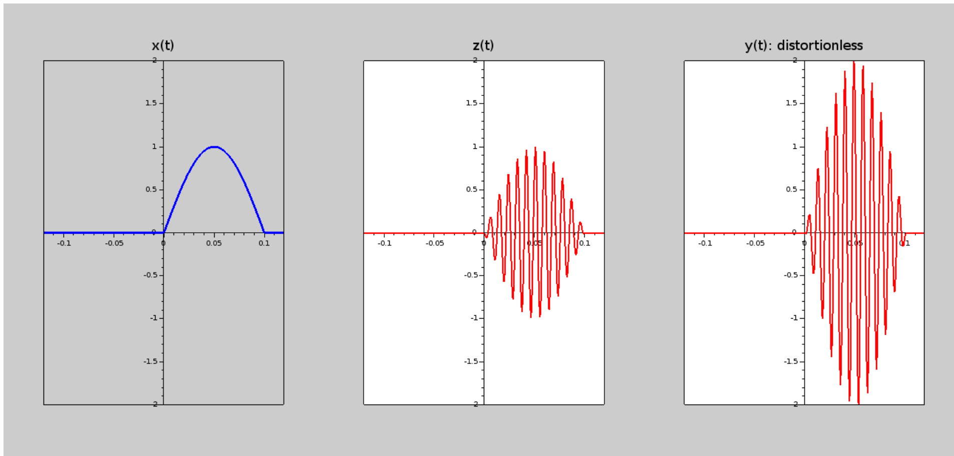

Figure 7.17: Filtered signals

```
subplot(1,3,2);plot(t,z,'r');
title("z(t)","fontsize",4);
set(gca(),"zoom_box",[t(1) -2 t($) 2],"x_location","
    middle","y_location","middle");
subplot(1,3,3);plot(t,y,'r');
title("y(t): distortionless","fontsize",4);
set(gca(),"zoom_box",[t(1) -2 t($) 2],"x_location","
    middle","y_location","middle");

//******************part b
    ********************************
wc=4000*%pi;
y2=1.5*p(t).*cos(wc*t-3.1*%pi);
figure(2)
plot(t,y2,'b');
title("(part_b) y2(t): distortionless","fontsize",4)
    ;
set(gca(),"zoom_box",[t(1) -2 t($) 2],"x_location","
    middle","y_location","middle");
```

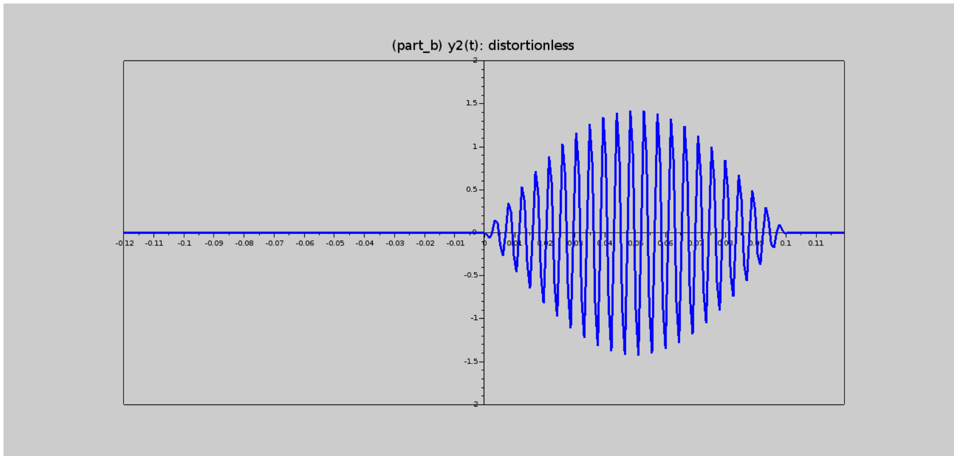

Figure 7.18: Filtered signals

Scilab code Exa 7.20 `Calculation of frequency range with signal energy`

```
clc
clear
close;

a=1;//taking a=1
E=1/(2*a);
W=a*tan(0.95*%pi*a*E);
printf("\nthe frequency upto W=%.3f rad/s will
   contain \n95 percent of the signal energy\n",W);
```

Scilab code Exa 7.23 `Toned modulated signals`

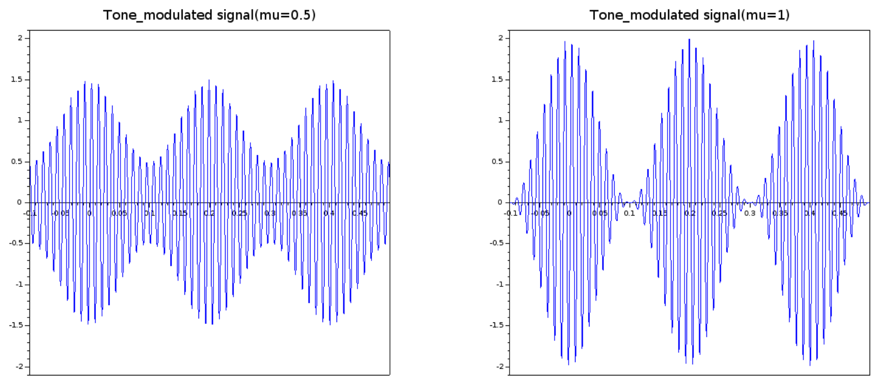

Figure 7.19: Toned modulated signals

```
clc
clear
close;
//assuming some random values
A=1;
wm=10*%pi;
wc=2000*%pi;
t=-0.1:0.00092:0.5;
m1=0.5;
m2=1;
y_am1=A*(1+m1*cos(wm*t)).*cos(wc*t);
y_am2=A*(1+m2*cos(wm*t)).*cos(wc*t);
subplot(1,2,1);plot(t,y_am1,'b');
title("Tone_modulated signal(mu=0.5)","fontsize",4);
set(gca(),"zoom_box",[t(1) -2.1 t($) 2.1],"
    x_location","middle");
subplot(1,2,2);plot(t,y_am2,'b');
title("Tone_modulated signal(mu=1)","fontsize",4);
set(gca(),"zoom_box",[t(1) -2.1 t($) 2.1],"
    x_location","middle");
```

Chapter 8

Sampling

Scilab code Exa 8.1 Effects of different sampling rates

```
clc
clear
close;
t=-5:0.01:5;
dt=0.01;
x=sinc(5*%pi*t).^2;
omega=-30:0.01:30;
p=0;
for om=-30:0.01:30
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
X=round(X*10000)/10000;
figure(1)
subplot(2,2,1);plot(t,x,'b');
title("x(t)=sinc(5pi*t).^2","fontsize",4);
set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-2 -1.2 2 1.2]);
subplot(2,2,2);plot(omega,X,'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","x_location","middle
```

```
","zoom_box",[-30 -0.4 30  0.4]);

//Under sampling(period=0.2)
p=1;
r=0.2;
for n=t(1)/r:1:t($)/r
    x1(p)=sinc(5*%pi*r*n).^2;
    p=p+1;
end
x1=x1';
t1=t(1)/r:1:t($)/r;
omega=-30:0.01:30;
p=0;
for om=-30:0.01:30
  p=p+1;
  X1(p)=sum(x1.*exp(-%i*om*t1));
end
X1=round(X1*10000)/10000;
subplot(2,2,3);plot2d3(t1,x1,[2]);
title("Under sampling(T=0.2)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-2 -1.2 2 1.2]);
subplot(2,2,4);plot(omega,abs(X1),'r');
title("X(w): signal unrecoverable","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[-20 -4 20  4]);

//Nyquist rate(period=0.1)
p=1;
r=0.1;
for n=t(1)/r:1:t($)/r
    x2(p)=sinc(5*%pi*r*n).^2;
    p=p+1;
end
x2=x2';
t2=t(1)/r:1:t($)/r;
omega=-30:0.01:30;
p=0;
```

```
for om=-30:0.01:30
  p=p+1;
  X2(p)=sum(x2.*exp(-%i*om*t2));
end
X2=round(X2*10000)/10000;
figure(2)
subplot(2,2,1);plot2d3(t2,x2,[2]);
title("sampling at Nyquist rate","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-4 -1.2 4 1.2]);
subplot(2,2,2);plot(omega,abs(X2),'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[-20 -4 20  4]);

//oversampling(period=0.05)
p=1;
r=0.05;
for n=t(1)/r:1:t($)/r
    x3(p)=sinc(5*%pi*r*n).^2;
    p=p+1;
end
x3=x3';
t3=t(1)/r:1:t($)/r;
omega=-30:0.01:30;
p=0;
for om=-30:0.01:30
  p=p+1;
  X3(p)=sum(x3.*exp(-%i*om*t3));
end
X3=round(X3*10000)/10000;
subplot(2,2,3);plot2d3(t3,x3,[2]);
title("oversampliing","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-4 -1.2 4 1.2]);
subplot(2,2,4);plot(omega,abs(X3),'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","x_location","middle
```

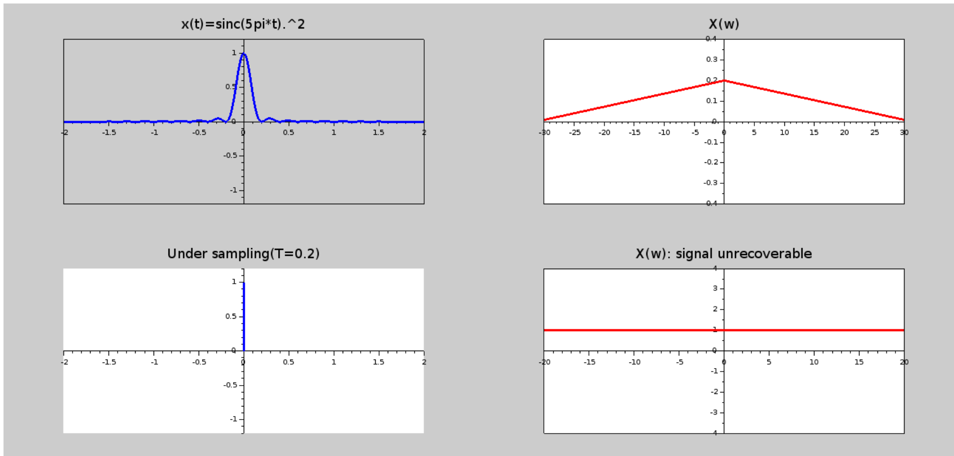

Figure 8.1: Effects of different sampling rates

```
","zoom_box",[-20 -4 20  4]);
```

Scilab code Exa 8.2 `Practical sampling`

```
//this code will take some time to give result as
   increments are very small
clc
clear
close;

t=-0.5:0.01:0.5;
dt=0.01;
x=sinc(5*%pi*t).^2;
omega=-30:0.01:30;
p=0;
for om=-30:0.01:30
```

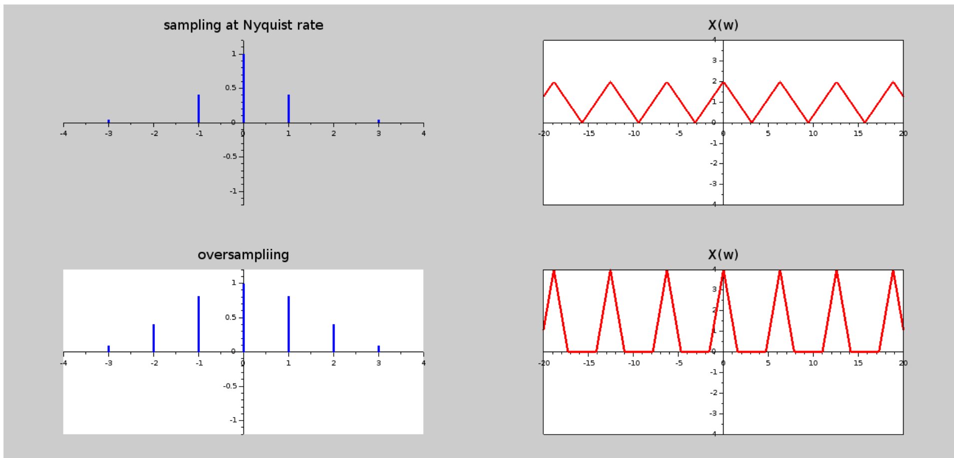

Figure 8.2: Effects of different sampling rates

```
  p=p+1;
  X(p)=sum(x.*exp(-%i*om*t).*dt);
end
figure(1)
subplot(1,2,1);plot(t,x,'b');
title("x(t)=sinc(5pi*t).^2","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-0.3 -1.2 0.3 1.2]);
subplot(1,2,2);plot(omega,X,'r');
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[-30 -0.4 30  0.4]);

//practical sampling
T=0.1;
p=1;
x1=(sinc(5*%pi*t).^2).*((abs(modulo(t,T))<0.0125)|(abs(modulo(t,T))>0.0875));
omega=-50*%pi:0.01:50*%pi;
p=0;
for om=-50*%pi:0.01:50*%pi
  p=p+1;
```

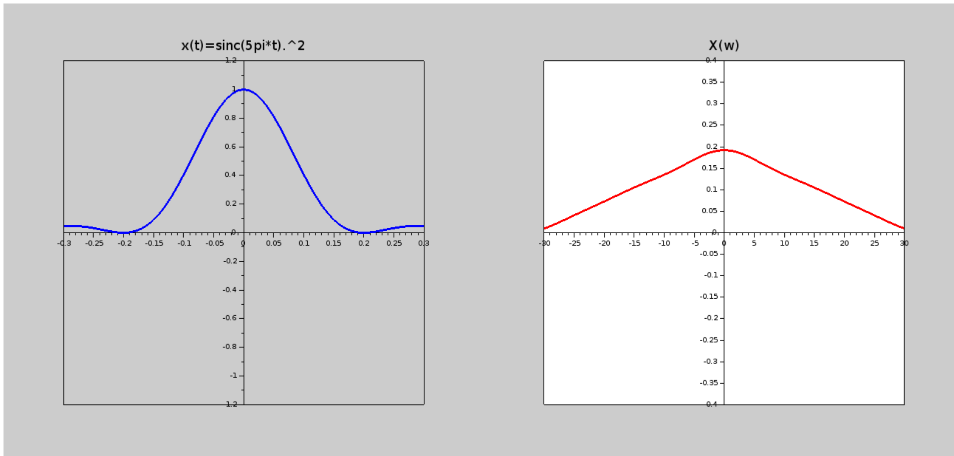

Figure 8.3: Practical sampling

```
  X1(p)=sum(x1.*exp(-%i*om*t).*0.009);
end
figure(2)
subplot(1,2,1);plot(t,x1,'b');plot(t,x,'r');
title("practical_sampling of x(t)","fontsize",4);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-0.3 -1.2 0.3 1.2]);
subplot(1,2,2);plot(omega,abs(X1),'r');
title("X(w) new","fontsize",4);
set(gca(),"y_location","middle","x_location","middle","zoom_box",[omega(1) -0.06 omega($) 0.06]);

//there will be small error in amplitude
```

Scilab code Exa 8.5 `Finding bit rate and sampling rate`

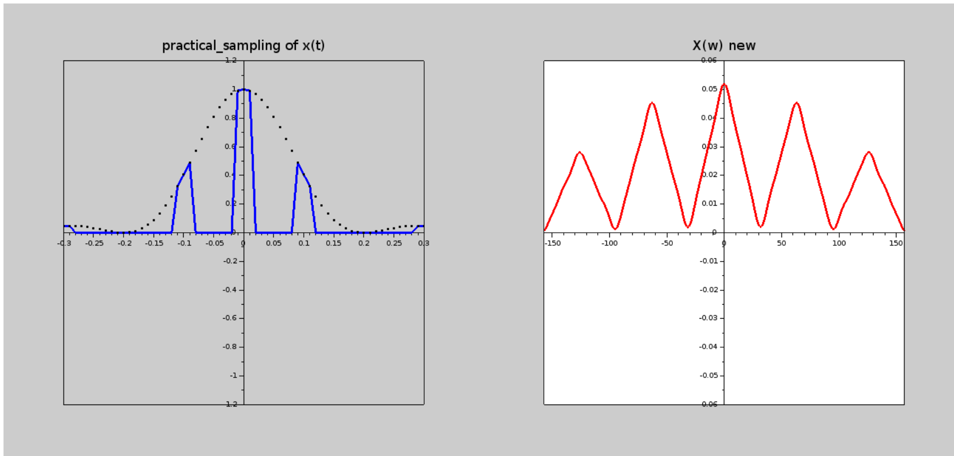

Figure 8.4: Practical sampling

```

clc
clear

fN=2*3000;//Nyquist sampling rate
fA=fN*(4/3);//actual sampling rate
L=256;//as it shouldn't be more than 0.05% and shouls be power of 2
bit=8;
bit_rate=bit*fA;
printf("\n required sampling rate= %.0f, bits required= %d, bit_rate= %d bits/s\n",fA,bit,bit_rate);
```

Scilab code Exa 8.7 `Finding N0 for DFT`

```
clc
clear

f0=100;
```

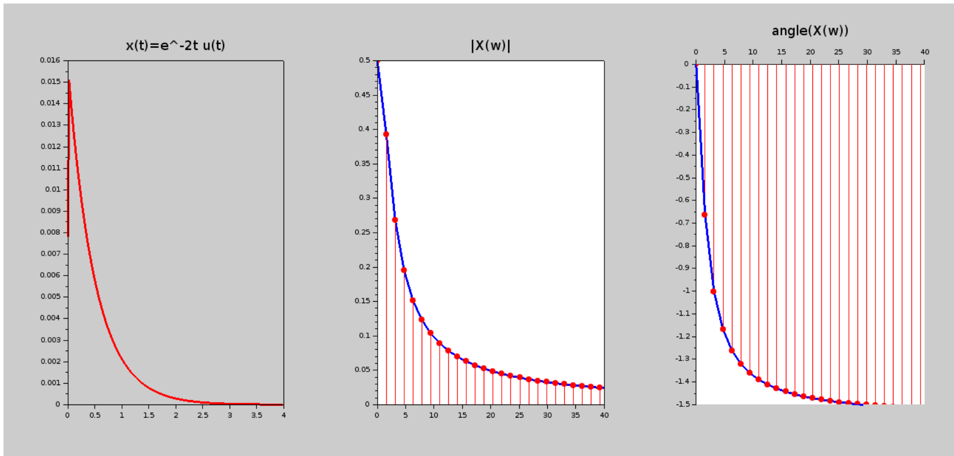

Figure 8.5: Fourier transform of given exponential signal using DFT

```
T0=1/f0;
B=10000;
fs=2*B;
T=1/fs;
N0=fs/f0;
printf("N0_calculated is %d\nfor proper DFT N0
    should be 2^n closet to %d(i.e;N0=256)",N0,N0);
```

Scilab code Exa 8.8 `Fourier transform of given exponential signal using DFT`

```
clc
clear
close;

N0=256; //samples no
T0=4;
T=T0/N0;
t=(0:T:(N0-1)*T)';
```

```
x=T*exp(-2*t);
x(1)=T*(exp(-2*T0)+1)/2;
Xr=fft(x);
r=-N0/2:N0/2-1;
omega=r*2*%pi/T0;
Xr_abs=fftshift(abs(Xr));
Xr_phase=fftshift(phasemag(Xr)*%pi/180);

figure(1)
subplot(1,3,1);plot(t,x,'r');
title("x(t)=e^-2t u(t)","fontsize",4);
subplot(1,3,2);plot(omega,Xr_abs,'b');plot2d3(omega,
    Xr_abs,[5]);
title("|X(w)|","fontsize",4);
set(gca(),"zoom_box",[0 0 40 0.5]);
subplot(1,3,3);plot2d3(omega,Xr_phase,[5]);plot(
    omega,Xr_phase,'b');
title("angle(X(w))","fontsize",4);
set(gca(),"zoom_box",[0 -1.5 40 0],"x_location","top
    ");
```

Scilab code Exa 8.9 Fourier transform of given pulse using DFT

```
clc
clear
close;

N0=32; //samples no
T0=4;
T=T0/N0;
x=[ones(1,4) 0.5 zeros(1,23) 0.5 ones(1,3)]';
Xr=fft(x);
```

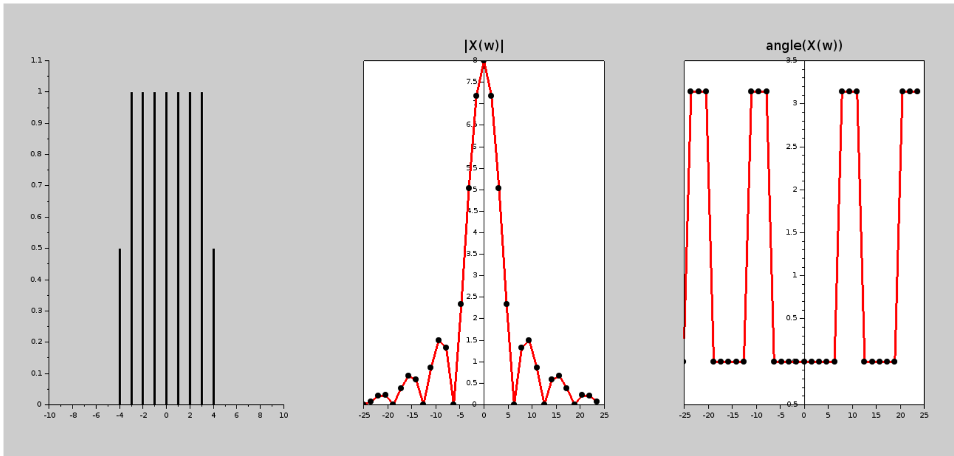

Figure 8.6: Fourier transform of given pulse using DFT

```
r=-N0/2:N0/2-1;
omega=r*2*%pi/T0;
Xr_abs=fftshift(abs(Xr));
X_phase=fftshift(phasemag(Xr)*%pi/180);

figure(1)
subplot(1,3,1);plot2d3([-32:63],repmat(x',1,3));
set(gca(),"zoom_box",[-10 0 10 1.1]);
subplot(1,3,2);plot(omega,Xr_abs,'b');
title("|X(w)|","fontsize",4);
set(gca(),"y_location","middle","zoom_box",[-25 0 25
    8]);
subplot(1,3,3);plot(omega,X_phase,'b');
title("angle(X(w))","fontsize",4);
set(gca(),"zoom_box",[-25 -0.5 25 3.5],"y_location",
   "middle");
```

Scilab code Exa 8.10 `Finding filtered output using DFT`

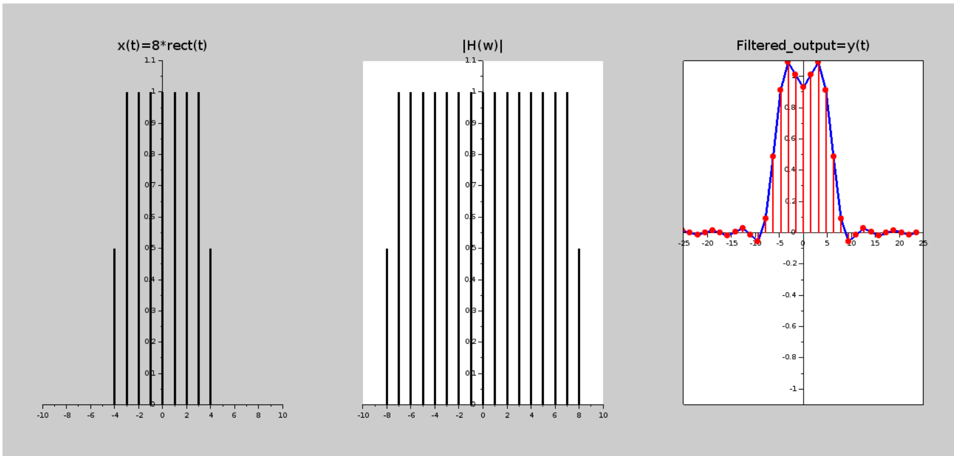

Figure 8.7: Finding filtered output using DFT

```
clc
clear
close;

N0=32; //samples no
T0=4;
T=T0/N0;
x=[ones(1,4) 0.5 zeros(1,23) 0.5 ones(1,3)]';
Xr=fft(x);
Hr=[ones(1,8) 0.5 zeros(1,15) 0.5 ones(1,7)]';
Yr=Xr.*Hr;
y=ifft(Yr);
r=-N0/2:N0/2-1;
n=r;
omega=r*2*%pi/T0;
y=fftshift(y);
figure(1)
subplot(1,3,1);plot2d3([-32:63],repmat(x',1,3));
set(gca(),"zoom_box",[-10 0 10 1.1],"y_location","
    middle");
title("x(t)=8*rect(t)","fontsize",4);
subplot(1,3,2);plot2d3([-32:63],repmat(Hr',1,3));
```

```
title("|H(w)|","fontsize",4);
set(gca(),"y_location","middle","zoom_box",[-10 0 10 1.1]);
subplot(1,3,3);plot(omega,y,'b');plot2d3(omega,y,[5]);
title("Filtered_output=y(t)","fontsize",4);
set(gca(),"zoom_box",[-25 -1.1 25 1.1],"y_location","middle","x_location","middle");
```

Chapter 9

Fourier analysis of discrete time signals

Scilab code Exa 9.1 Discrete time Fourier series

```
clc
clear
close;

 N0=20;
n=0:N0-1;
x=sin(0.1*%pi*n);
xr=fft(x)/N0;
xr=round(xr*10000)/10000;
n=-4*N0/2:4*N0/2-1;

subplot(1,3,1);plot2d3(n,repmat(x,1,4));
set(gca(),"x_location","middle","y_location","middle
   ");
title("x(n)=sin(0.1*pi*n)","fontsize",4);
subplot(1,3,2);plot2d3(n,repmat(abs(xr),1,4),[5]);
set(gca(),"y_location","middle");
```

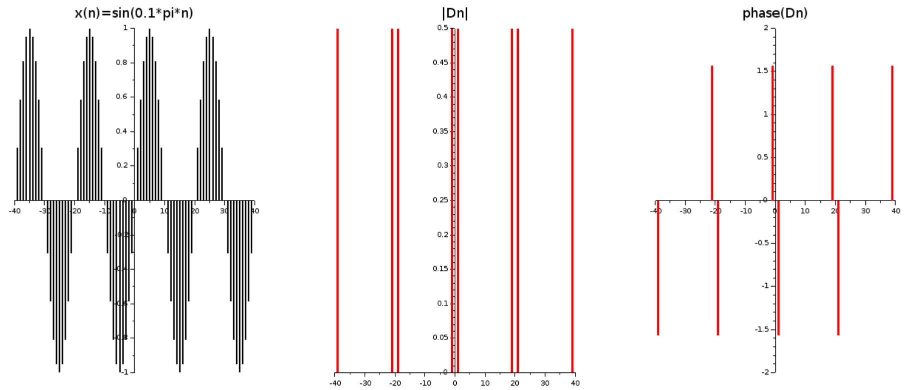

Figure 9.1: Discrete time Fourier series

```
title("|Dn|","fontsize",4);
subplot(1,3,3);plot2d3(n,repmat(phasemag(xr)*%pi/180,1,4),[5]);
set(gca(),"x_location","middle","y_location","middle");
title("phase(Dn)","fontsize",4);
```

Scilab code Exa 9.2 Discrete time Fourier series of periodic sampled gate function

```
clc
clear
close;

N0=32;
n=0:N0-1;
x=[ones(1,5) zeros(1,23) ones(1,4)];
xr=fft(x)/N0;
xr=round(xr*10000)/10000;
```

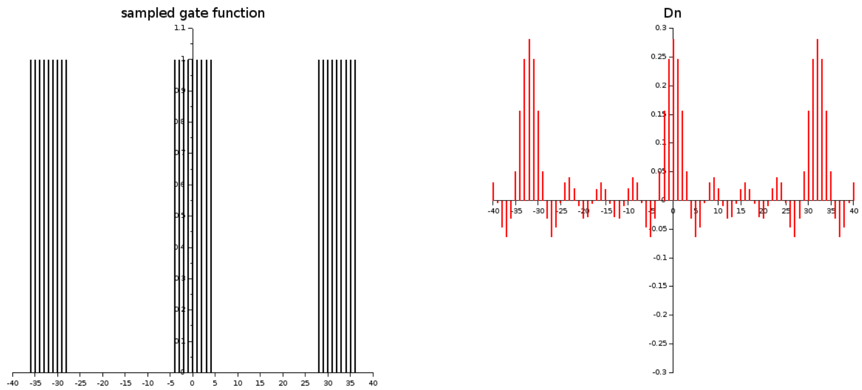

Figure 9.2: Discrete time Fourier series of periodic sampled gate function

```
n=-4*N0/2:4*N0/2-1;

subplot(1,2,1);plot2d3(n,repmat(x,1,4));
set(gca(),"y_location","middle","zoom_box",[-40 0 40 1.1]);
title("sampled gate function","fontsize",4);
subplot(1,2,2);plot2d3(n,repmat(xr,1,4),[5]);
set(gca(),"x_location","middle","y_location","middle","zoom_box",[-40 -0.3 40 0.3]);
title("Dn","fontsize",4);
```

Scilab code Exa 9.3 `Discrete time Fourier transform`

```
clc
clear
close;

N=1024; //samples number
```

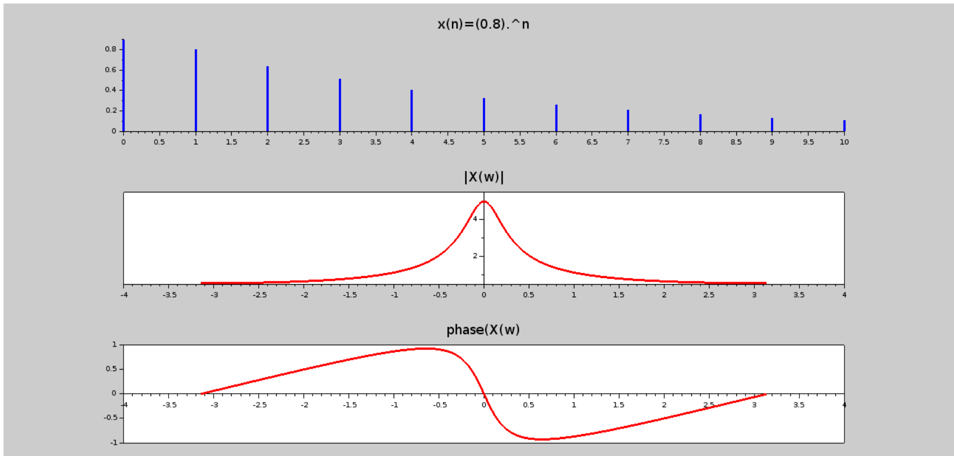

Figure 9.3: Discrete time Fourier transform

```
n=0:N-1;
x=(0.8).^n;
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);

figure(1)
subplot(3,1,1);plot2d3(n,x,[2]);
title("x(n)=(0.8).^n","fontsize",4);
set(gca(),"zoom_box",[0 0 10 0.9]);
subplot(3,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
subplot(3,1,3);plot(omega,X_angle,'r');
set(gca(),"x_location","middle");
title("phase(X(w)","fontsize",4);
```

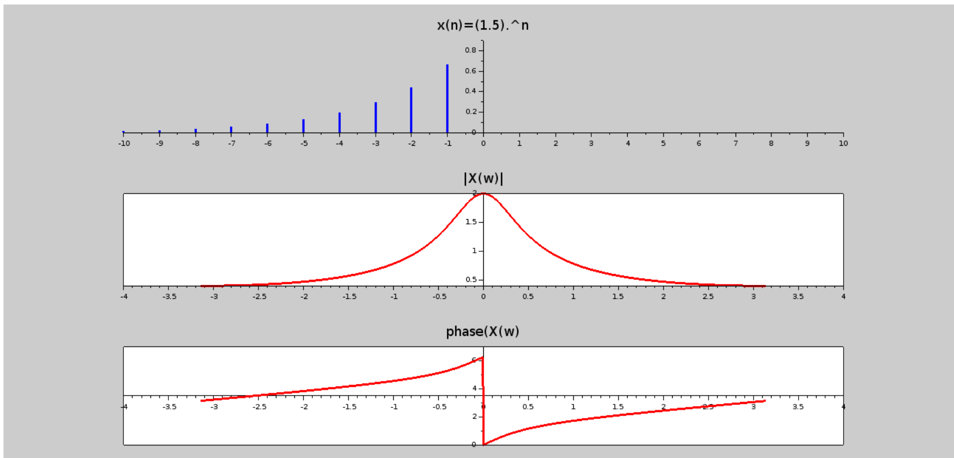

Figure 9.4: Discrete time Fourier transform

Scilab code Exa 9.4 `Discrete time Fourier transform`

```
clc
clear
close;

N=1024; //samples number
n=-N:-1;
x=(1.5).^n;
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);

figure(1)
subplot(3,1,1);plot2d3(n,x,[2]);
title("x(n)=(1.5).^n","fontsize",4);
set(gca(),"zoom_box",[-10 0 10 0.9],"y_location","
```

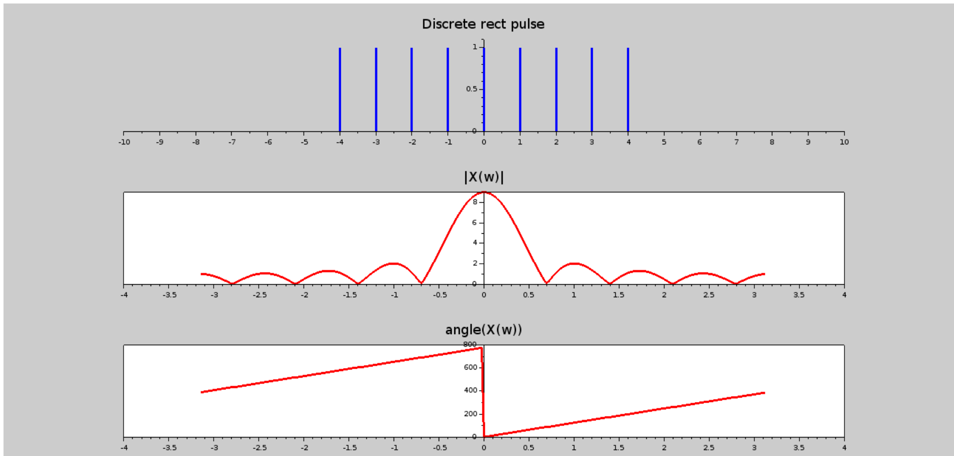

Figure 9.5: DTFT of discrete time rectangular pulse

```
    middle");
subplot(3,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
subplot(3,1,3);plot(omega,X_angle,'r');
set(gca(),"x_location","middle","y_location","middle
    ");
title("phase(X(w)","fontsize",4);
```

Scilab code Exa 9.5 `DTFT of discrete time rectangular pulse`

```
clc
clear
close;

N=256; //samples number
n=-(N/2):N/2-1;
x=1.*((n>=-4)&(n<=4));
```

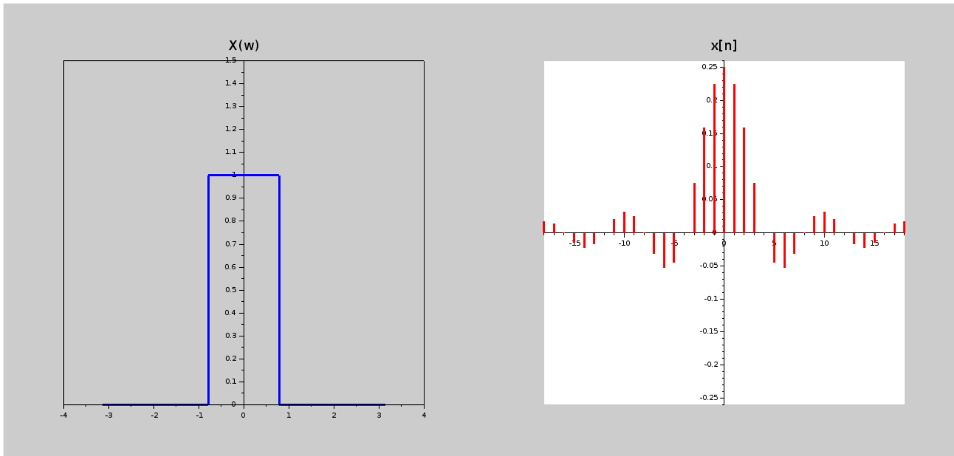

Figure 9.6: DTFT of discrete time rectangular pulse

```
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);

figure(1)
subplot(3,1,1);plot2d3(n,x,[2]);
title("Discrete rect pulse","fontsize",4);
set(gca(),"zoom_box",[-10 0 10 1.1],"y_location","
    middle");
subplot(3,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
subplot(3,1,3);plot(omega,X_angle,'r');
set(gca(),"y_location","middle");
title("angle(X(w))","fontsize",4)
```

Scilab code Exa 9.6 DTFT of discrete time rectangular pulse

```
clc
clear
close;


function y=X(om)
    y=1.*(abs(om)<=%pi/4);
endfunction

omega=linspace(-%pi,%pi,1024);
domega=omega(2)-omega(1);
q=length(omega);
n=-80:1:q-81;
p=1;
for n1=n
   r(p)=(1/(2*%pi)).*sum(X(omega).*exp(%i*omega*n1)
      .*domega);
   p=p+1;
end

figure(1)
subplot(1,2,1);plot(omega,X(omega));
title("X(w)","fontsize",4);
set(gca(),"y_location","middle","zoom_box",[-4 0 4
   1.5]);
subplot(1,2,2);plot2d3(n,r,[5]);
set(gca(),"zoom_box",[-18 -0.26 18 0.26],"x_location
   ","middle","y_location","middle");
title("x[n]","fontsize",4);
```

Scilab code Exa 9.9 DTFT of the given signal

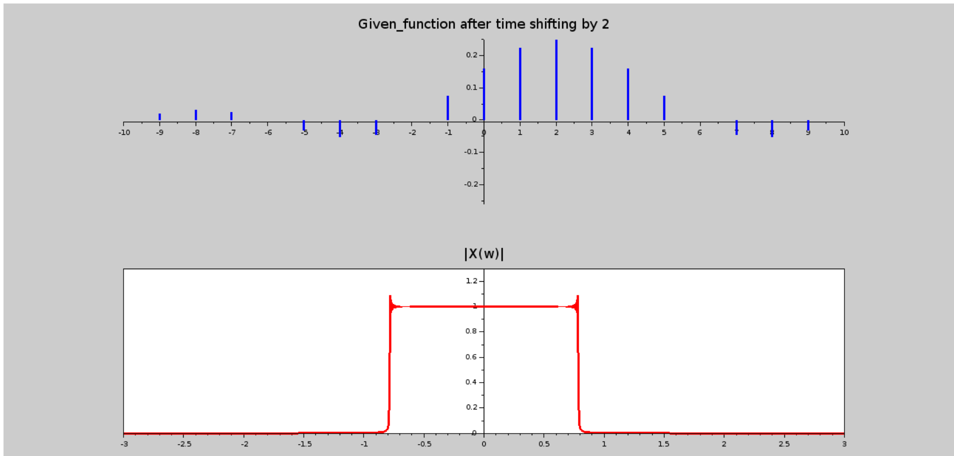

Figure 9.7: DTFT of the given signal

```
clc
clear
close;
//error in phase calculation will there
N=1024; //samples number
n=-(N/2):N/2-1;
function y=f(n)
    y=(1/4)*sinc(%pi*n/4);
endfunction
x=f(n-2);
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));

figure(1)
subplot(2,1,1);plot2d3(n,x,[2]);
title("Given_function after time shifting by 2","
   fontsize",4);
set(gca(),"zoom_box",[-10 -0.26 10 0.25],"y_location
   ","middle","x_location","middle");
subplot(2,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle","zoom_box",[-3 0 3
```

```
1.3]);
title("|X(w)|","fontsize",4);
```

Scilab code Exa 9.10 DTFT of the given modulated signal

```
clc
clear
close;

//*****************************part a(wc=pi/2)*********************
N=1024; //samples number
n=-(N/2):N/2-1;
x=sinc(%pi*n/4).*cos(%pi*n/2);
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);

figure(1)
subplot(2,1,1);plot2d3(n,x,[2]);
title("part(a)_modulated signal","fontsize",4);
set(gca(),"zoom_box",[-30 -1.1 30 1.1],"y_location","middle","x_location","middle");
subplot(2,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);

//**************************part b(wc=0.875*pi)***************************
x=sinc(%pi*n/4).*cos(%pi*n*0.875);
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
```

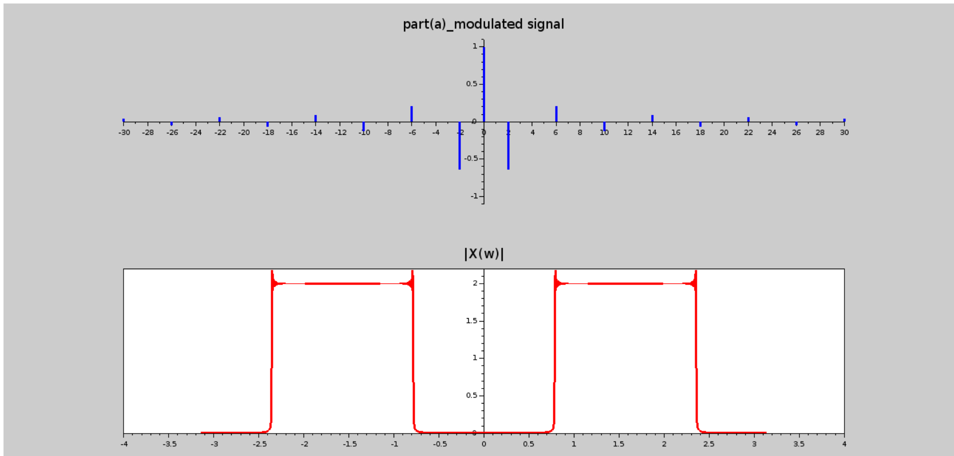

Figure 9.8: DTFT of the given modulated signal

```
X_angle=fftshift(phasemag(X)*%pi/180);
n1=[-3*N/2:3*N/2-1]*2*%pi/N;
figure(2)
subplot(2,1,1);plot2d3(n,x,[2]);
title("part(b)_modulated signal","fontsize",4);
set(gca(),"zoom_box",[-30 -1.1 30 1.1],"y_location","middle","x_location","middle");
subplot(2,1,2);plot(n1,repmat(X_mode,1,3),'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
```

Scilab code Exa 9.13 `ZSR of LTID system`

```
clc
```

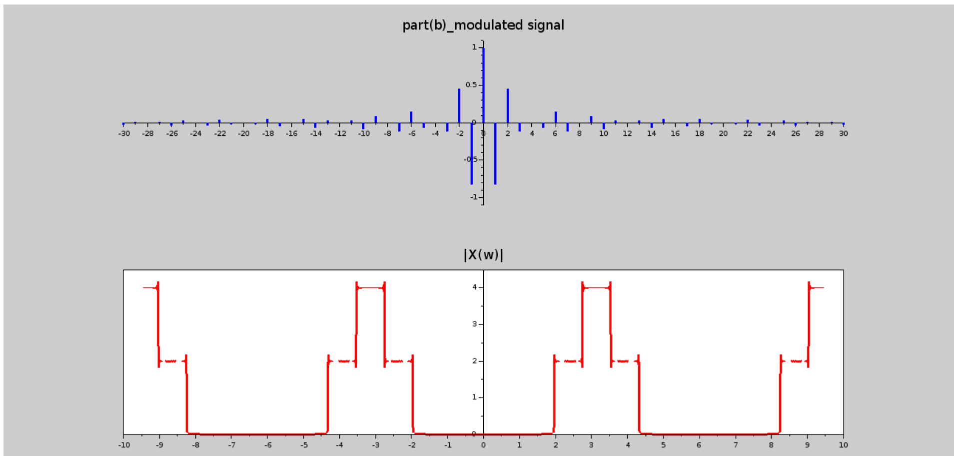

Figure 9.9: DTFT of the given modulated signal

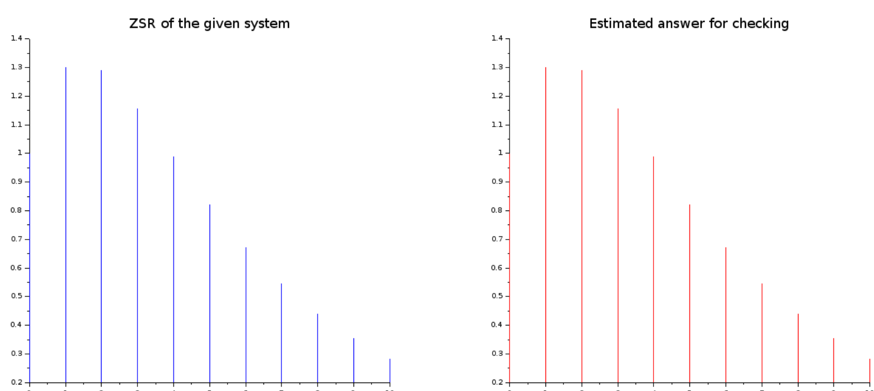

Figure 9.10: ZSR of LTID system

```
clear
close;


a=[1 -0.5]; b=[1 0];
n=[0:10]';x=(0.8).^n;
zsr=filter(b,a,x);
z_check=(-5/3)*(0.5).^n+(8/3)*(0.8).^n;
subplot(1,2,1);plot2d3(n,zsr,[2]);
title("ZSR of the given system","fontsize",4);
subplot(1,2,2);plot2d3(n,z_check,[5]);
title("Estimated answer for checking","fontsize",4);
```

Chapter 10

State space analysis

Scilab code Exa 10.4 `State space representation from transfer function`

```
clear ;
close ;
clc ;
s= poly(0,'s' );
H =[(4/3)/(1+ s),-2/(3+s),(2/3)/(4+ s)];
Sys = tf2ss(H);
clean(ss2tf(Sys));
printf("The state-space representation is as below\n
   ");
disp(Sys)
```

---

check Appendix AP 1 for dependency:

`inv_laplace.sci`

Scilab code Exa 10.5 `finding state vector`

```
clc
```

```
//symbolic tool(scilab-scimax) is needed which should be installed properly in Ubuntu 14.04 and Scilab -5.5.0
Syms t s

//execute the file ilaplace.sci to get inverse laplace
//path should be correctly written
exec('C:\Users\Satyajit\Desktop\sample_codes\inv_laplace.sci',-1)

A=[-12 2/3;-36 -1]; B=[1/3;1]; q0=[2;1]; X=1/s;
size(A)
size(s*eye(2,2))
Q=inv(s*eye(2,2)-A)*(q0+B*X);
q=[];
q(1)=ilaplace(Q(1));
q(2)=ilaplace(Q(2));
disp(q*'u(t)',"[q1(t) ; q2(t)]")
```

Scilab code Exa 10.6 transfer function from state space equation

```
clc

s=%s;
A=[0 1;-2 -3];
B=[1 0;1 1];
C=[1 0;1 1;0 2];
D=[0 0;1 0; 0 1];
H=simp(C*inv(s*eye(2,2)-A)*B+D);
disp(H,"the transfer function matrix H(s)=")
disp(H(3,2),"the transfer function relating y3 and x2 is H(s)=")
```

Scilab code Exa 10.7 time domain method of solving the system

```
clc ;
clear;

s=%s;
A=[s+12 -2/3;36 s+1];
y= roots(det(A))
t= poly(0,'t' );
beta =inv([1 y(1); 1 y(2)])*[%e^-y(1)*t;%e^-y(2)*t];
disp(beta,"***********beta=")
W= beta(1)*[1 0;0 1]+ beta(2) *[ -12 2/3; -36 -1];
zir =W*[2;1];
disp(zir,"*************ZIR=");
zsr =W*[1/3;1];
disp(zsr,"*************ZSR= ");
total=zir +zsr;
disp(total,"******total response=");
```

Scilab code Exa 10.8 Determining exponential matrix

```
clc
//symbolic tool(scilab-scimax) is needed which should be installed properly in Ubuntu 14.04 and Scilab-5.5.0

Syms t s;
//execute the file ilaplace.sci to get inverse laplace
//path should be correctly written
exec('C:\Users\Satyajit\Desktop\sample_codes\ilaplace.sci',-1)
```

```
F1=ilaplace((s+3)/((s+1)*(s+2)))
F2=ilaplace(1/((s+1)*(s+2)))
F3=ilaplace(-2/((s+1)*(s+2)))
F4=ilaplace(s/((s+1)*(s+2)))
F=[F1 F2;F3 F4];
disp(F,"f(t)=")
A=[1 0;1 1;0 2];
B=[0 0;1 0;0 1];
h=A*F*[1 0;1 1]+B*eye(2,2);//including delta(t) function
disp(h,"h(t)=")
```

Scilab code Exa 10.9 Finding new state equations

```
clc
clear

A =[0 1; -2 -3];
B =[1;2];
P=[1 1;1 -1];
Ahat =P*A*inv(P);Bhat=P*B;
disp(Bhat ,"Bhat")
disp(Ahat," Ahat=")
```

Scilab code Exa 10.10 Diagonalized form of the state equations

```
clc
clear

format(7)
A =[0 1; -2 -3];
[V,lambda ]= spec(A);
```

```
P=inv(V);
B =[1;2];
Bhat =P*B;
disp(P,"P=")
disp(Bhat ,"B^=")
disp(lambda," lambda=")
```

Scilab code Exa 10.11 controllability and observability of the system

```
clc
clear
close
//*************part(a)***************
A =[1 0;1 -1];
[V,lambda ]= spec(A);
B=[1;0];
C=[1 -2];
P= inv(V);
Bh=P*B
Ch=C*inv(P)
disp( ' part(a): ' )
disp(Bh ,"B^=")
disp(Ch ,"C^=")
//*******************part(b)*********
A=[ -1 0; -2 1];
[V, lambda ]= spec (A);
B =[1;1];
C =[0 1];
P= inv (V);
Bh =P*B
Ch =C*inv(P)
disp(' Part(b): ' )
disp(Bh ,"B^=")
disp(Ch ," C^=")
```

Scilab code Exa 10.12 finding output of the system

```
clc
clear
close

A =[0 1; -1/6 5/6];
B =[0;1];
C=[ -1 5];
D =0;
sys=syslin('d',A,B,C,D);
N =25;
x= ones(1,N +1) ;n =(0: N);
q0=[2;3];
[ y q]=csim('step',n,sys);
y=dsimul(sys ,x);
plot2d3(y);
title("y[n] ZSR","fontsize",4);
```

Scilab code Exa 10.13 response of given system using z transform

```
//Inverse Z Transform
//symbolic tool(scilab-scimax) is needed which
   should be installed properly in Ubuntu 14.04 and
   Scilab -5.5.0
//limit function can be used only after installing
   symbolic toolbox

clc
Syms n z;
H1 = (-2*z)/(z-(1/3));
H2 = (3*z)/(z-0.5);
```

```
H3 = (24*z)/(z-1);
F1 = H1*z^(n-1)*(z-(1/3));
F2 = H2*z^(n-1)*(z-0.5);
F3 = H3*z^(n-1)*(z-1);
h1 = limit(F1,z,(1/3));//finding symbolic limits
disp(h1,'h1[n]=')
h2 = limit(F2,z,0.5);//finding symbolic limits
disp(h2,'h2[n]=')
h3 = limit(F3,z,1);//finding symbolic limits
disp(h3,'h3[n]=')
h = h1+h2+h3;
disp(h,'h[n]=')
```

Appendix

Scilab code AP 1 `Inverse laplace transform`

```
function [sym1] = ilaplace(A,t,s)

// ilaplace
// ---------------------------------------------------------------------------
// PURPOSE
// Inverse laplace transform.
//
// SYNOPSIS
// B = ilaplace(A[,s[,t]])
//
// INPUT ARGUMENTS
// A        Symbolic expression.
// t        Time variable ('t' by default)
// s        Laplace variable ('s' by default)
//
// OUTPUT ARGUMENT
// B        Symbolic expression.
//
// See also: laplace
// #--------------------------------------------------------------------------
// // EXAMPLE
//    syms t p
```

```
//     ilaplace('exp(-2*p)/(1+p^2)',p,t) // Maxima
   ignores Heaviside!
//     ilaplace('1/(p*(1+p^2))',p,t)
//     laplace(ilaplace(1/p^2,p,t),t,p)

// Author: : J.F. Magni
// 18-04-2006

    [%nargout,%nargin] = argn(0);
    if %nargin ~= 1 & %nargin ~= 2 & %nargin ~= 3;
       error('One to three arguments'); end;

    if %nargin == 1; t = 's'; s = 't'; end;
    if %nargin == 2; s = 't'; end;

    A = sym(A);
    t = sym(t);
    s = sym(s);
    str1 = 'ilt('+A.str1+','+t.str1+','+s.str1+')'

    sym1 = str2max2sym(str1);

endfunction
```

Scilab code AP 2 Finding laplace transform

```
clc
clear

function [sym1] = laplace(A,t,s)

// laplace
//
   ------------------------------------------------------------------------

// PURPOSE
// Laplace transform. Recognizes the functions or
   strings: delta,
```

```
// exp, log, sin, cos, sinh, cosh, erf and ilt (
   name of the Maxima
// inverse Laplace transform).
//
// SYNOPSIS
// B = laplace(A[,t[,s]])
//
// INPUT ARGUMENTS
// A        Symbolic expression.
// t        Time variable ('t' by default)
// s        Laplace variable ('s' by default)
//
// OUTPUT ARGUMENT
// B        Symbolic expression.
//
// See also: ilaplace, sym/integ
//
   #------------------------------------------------------------------
// // EXAMPLE
//    syms t s
//    laplace('3*delta(t-2)+cos(t)*sin(t)',t,s)
//    laplace('3*delta(t-2)+cos(t)*sin(t)',t,'p')
//    laplace(diff(cos(t)^3,t),t,s)

// Author: : J.F. Magni
// 18-04-2006

    [%nargout,%nargin] = argn(0);
    if %nargin ~= 1 & %nargin ~= 2 & %nargin ~= 3;
       error('One to three arguments'); end;

    if %nargin == 1; t = 't'; s = 's'; end;
    if %nargin == 2; s = 's'; end;

    A = sym(A);
    t = sym(t);
    s = sym(s);
```

```
    str1 = 'laplace('+A.str1+','+t.str1+','+s.str1+')'

    sym1 = str2max2sym(str1);

endfunction
```

Scilab code AP 3 `Graphical_convolution`

```
//this is the existing dependency file to be used for graphical convolution
//*****************************************************************
//*****************************************************************

//functions should be declared like this and the function should be called after the execution of function definition file
//t='[-5:0.02:5]';
//f1='u(t)';    //defining x(t)
//f2='-2*exp(2*t).*u(-t)+2*exp(-t).*u(t)'; //defining h(t)
//conv_gui(f1,f2,t);(should be executed at the last part)

funcprot(0);

// Defining event handling for plot window

function conv_gui_event(win,x,y,ibut)
    if ibut==37
        conv_gui('move_left');
    elseif ibut==39
        conv_gui('move_right');
    end
```

```
endfunction


function []=conv_gui(varargin)

    // Defining unit step function
    function y=u(x)
        y=sign((sign(x)+1))
    endfunction

    action='init_gui';


    if argn(2) ==1 then
        action=varargin(1);
    elseif argn(2) ==2 then
        if varargin(1)=="load_sample" then
            action="load_sample";
            n_sample= varargin(2);
        end
    end

    if action == 'init_gui' //
       ********************************************
        h=findobj('tag','conv_gui_main_window');
        if h == [] then
            // Drawing the figure
            h=figure('Tag','conv_gui_main_window','
               Figure_name','Convolution of two
               function');
            h.event_handler="conv_gui_event";
            h.event_handler_enable="on";

            // creating axes
            subplot(3,1,1);
            ax1= gca();
            ax1.tight_limits='on';
            ax1.auto_clear='on';
```

```
            subplot(3,1,2);
            ax2= gca();
            ax2.tight_limits='on';
            ax2.auto_clear='on';
            subplot(3,1,3);
            ax3= gca();
            ax3.tight_limits='on';
            ax3.auto_clear='on';



            // Deleting menus
            delmenu(h.figure_id, gettext("&File"));
            delmenu(h.figure_id, gettext("&Edit"));
            delmenu(h.figure_id, gettext("&Tools"));
            delmenu(h.figure_id, gettext("&?"));

            // Hiding toolbar
            toolbar(h,'off');

            // adding menus

            mnu_options = uimenu(h, ...
            'Label','Options');
            mnu_options_set_fun = uimenu(mnu_options, ...
            'Label','Set Functions',...
            'Callback', 'conv_gui(''set_functions'')');
            mnu_options_illustrate=uimenu(mnu_options, ...
            'checked','off',...
            'Label','Illustratrate',...
            'Callback', 'conv_gui(''illustrate'')');

            mnu_samples = uimenu(h, ...
            'Label','Samples');
            mnu_samples_1 = uimenu(mnu_samples, ...
```

```
            'Label','Sample 1', ...
            'Callback', 'conv_gui(''load_sample'',1)');
            mnu_samples_2 = uimenu(mnu_samples, ...
            'Label','Sample 2', ...
            'Callback', 'conv_gui(''load_sample'',2)');
            mnu_samples_1 = uimenu(mnu_samples, ...
            'Label','Sample 3', ...
            'Callback', 'conv_gui(''load_sample'',3)');

            mnu_move_left = uimenu(h, ...
            'Label','Move Left <--',...
            'Enable','off',...
            'Callback','conv_gui(''move_left'')');
            mnu_move_right = uimenu(h, ...
            'Label',' --> Move Right',...
            'Enable','off',...
            'Callback','conv_gui(''move_right'')');
            mnu_about = uimenu(h, ...
            'Label','About',...
            'Callback','conv_gui(''about'')');



            mnu=struct('illustrate', mnu_options_illustrate,'left', mnu_move_left,'right',mnu_move_right);

            // Storing Data
            move_step=0.1;
            t_shift=0;
            t_data=[];
            t_min=[];
            t_max=[];
            t_step=[];
```

```
            f1_data=[];
            f2_data=[];
            conv_data=[];
            init_plot=1;
            user_data = struct('t',t,'f1',f1,'f2',f2,'move_step',move_step, ...
            'ax1',[ax1],'ax2',[ax2],'ax3',[ax3],...
            't_shift',[t_shift],'t_data',[t_data],'f1_data',[f1_data],...
            'f2_data',[f2_data],'conv_data',[conv_data],'init_plot',init_plot,...
            't_min',t_min,'t_max',t_max','t_step',t_step,'mnu',mnu);

            h.userdata = user_data

            // evaluating functions
            conv_gui('eval_functions');

        else
            show_window(h);
        end

    elseif action == 'about' //*******************************************
        messagebox(["Author: Mahmoud A. AlNaanah" ...
         "email: malnaanah@gmail.com" ...
         "Last update: 10.08.2015 (DD.MM.YYYY)" ...
         "License: GPL" ...
         "tested on scilab v 5.5.0" ...
         ], "About", "info");

    elseif action == 'illustrate' //*******************************************
        h=findobj('tag','conv_gui_main_window');

        if (h.userdata.mnu.illustrate.checked=='off'
```

```
            ) then
            h.info_message="Click on the figures and use left and right arrows (or click Move Left, Move Right from menu) to move function 2";
            h.userdata.mnu.illustrate.checked='on';
            h.userdata.mnu.left.enable='on';
            h.userdata.mnu.right.enable='on';
            h.userdata.ax1.children.children(2).visible='on';
            h.userdata.ax1.title.text="First function (F1) (Red line), F1 x F2 (Blue line) "
            conv_gui('plot')
        else
            conv_gui('eval_functions')
        end

    elseif action == 'load_sample' //*******************************************
        h=findobj('tag','conv_gui_main_window');

        if n_sample == 1 then
            h.userdata.t='[-5:0.02:5]';
            h.userdata.f1='2*(u(t)-u(t-1))';
            h.userdata.f2='(u(t)-u(t-2))';
            conv_gui('eval_functions')
        elseif n_sample == 2 then
            h.userdata.t='[-5:0.02:5]';
            h.userdata.f1='t .* (u(t)-u(t-1))';
            h.userdata.f2='t .* (u(t)-u(t-1))';
            conv_gui('eval_functions')
        elseif n_sample == 3 then
            h.userdata.t='[-5:0.02:5]';
            h.userdata.f1='t .* (u(t+1)-u(t-1))';
            h.userdata.f2='t .* (u(t+1)-u(t-1))';
            conv_gui('eval_functions')
        end
```

```
    elseif action == 'move_right' //*****************************************
        h=findobj('tag','conv_gui_main_window');
        if h.userdata.mnu.illustrate.checked=='on' then
            h.userdata.t_shift=h.userdata.t_shift+h.userdata.move_step;
            if h.userdata.t_shift > (h.userdata.t_max -h.userdata.t_min) then
                h.userdata.t_shift=h.userdata.t_shift-h.userdata.move_step;
            else
                conv_gui('plot')
            end
        end


    elseif action == 'move_left' //*****************************************
        h=findobj('tag','conv_gui_main_window');

        if h.userdata.mnu.illustrate.checked=='on' then
            h.userdata.t_shift=h.userdata.t_shift-h.userdata.move_step;
            if h.userdata.t_shift < (h.userdata.t_min -h.userdata.t_max) then
                h.userdata.t_shift=h.userdata.t_shift+h.userdata.move_step;
            else
                conv_gui('plot')
            end
        end

    elseif action == 'set_functions' //*****************************************
        h=findobj('tag','conv_gui_main_window');
```

```
        labels = ["t=";"f1=";"f2=";"Move step="];
        vals =  [h.userdata.t;h.userdata.f1;h.userdata.f2;string(h.userdata.move_step)]
        lst = list("str",1,"str",1,"str",1,"str",1);

        [ok,t,f1,f2,move_step]=getvalue("Use t as variable. u(t) is the unit step function. Use .* instead of * for multiplication", labels,lst,vals);

        if ok
            h.userdata.t = t;
            h.userdata.f1 = f1;
            h.userdata.f2 = f2;
            h.userdata.move_step = evstr(move_step);
            conv_gui('eval_functions');
        end



    elseif action == 'eval_functions' //********************************************

        h=findobj('tag','conv_gui_main_window');
        h.info_message="Go to Options > Illustrate, to see graphical illustration of convolution";
        t=evstr(h.userdata.t);
        f1=evstr(h.userdata.f1);
        f2=evstr(h.userdata.f2);
        mov_step=h.userdata.move_step;

        h.userdata.t_min=min(t);
        h.userdata.t_max=max(t);
        h.userdata.t_step=t(2)-t(1);

        conv=convol(f1,f2)*h.userdata.t_step;
```

```
        SZ = size(t,2);
        STRT = round(abs(h.userdata.t_min/h.userdata.t_step));
        END = STRT+SZ-1;
        conv=conv(STRT:END);


        h.userdata.t_data=[];
        h.userdata.f1_data=[];
        h.userdata.f2_data=[];
        h.userdata.conv_data=[];

        h.userdata.t_data=t;
        h.userdata.f1_data=f1;
        h.userdata.f2_data=f2;
        h.userdata.conv_data=conv;
        h.userdata.init_plot=1;
        h.userdata.t_shift=0;


        h.userdata.mnu.illustrate.checked='off'
        h.userdata.mnu.left.enable='off';
        h.userdata.mnu.right.enable='off'
        h.userdata.t_shift=0;

        conv_gui('plot');

        h.userdata.ax1.children.children(2).visible='off';

    elseif action == 'plot' //******************************************

        h=findobj('tag','conv_gui_main_window');
        t=h.userdata.t_data;
        f1_data=h.userdata.f1_data;
        f2_data=h.userdata.f2_data;
        conv_data=h.userdata.conv_data;
```

```
        if h.userdata.init_plot then
            sca(h.userdata.ax1)
            plot(t,f1_data,t,f1_data);

            data_bounds=h.userdata.ax1.data_bounds;
            data_bounds=data_bounds .*[1,1.1;1,1.1];
            h.userdata.ax1.title.text="input_function(x(t))";
            h.userdata.ax1.title.font_size=4;
            h.userdata.ax1.data_bounds=data_bounds;
            h.userdata.ax1.children.children(1).polyline_style=1;
            h.userdata.ax1.children.children(1).thickness=2;
            h.userdata.ax1.children.children(1).foreground=2;

            h.userdata.ax1.children.children(2).polyline_style=1;
            h.userdata.ax1.children.children(2).line_style=8;
            h.userdata.ax1.children.children(2).thickness=2;
            h.userdata.ax1.children.children(2).foreground=5;
            h.userdata.ax1.children.children(2).visible='off';


            sca(h.userdata.ax2)
            plot(t,f2_data);
            h.userdata.ax2.title.text="impulse_function(h(t))";
            h.userdata.ax2.title.font_size=4;
            data_bounds=h.userdata.ax2.data_bounds;
            data_bounds=data_bounds .*[1,1.1;1,1.1];
            h.userdata.ax2.data_bounds=data_bounds;
```

```
            h.userdata.ax2.children.children.polyline_style=1;
            h.userdata.ax2.children.children.thickness=2;
            h.userdata.ax2.children.children.foreground=3;

            sca(h.userdata.ax3)
            plot(t,conv_data);
            h.userdata.ax3.title.text="Convoluted_output(y(t))";
            h.userdata.ax3.title.font_size=4;
            data_bounds=h.userdata.ax3.data_bounds;
            data_bounds=data_bounds .*[1,1.1;1,1.1];
            h.userdata.ax3.data_bounds=data_bounds;
            h.userdata.ax3.children.children.polyline_style=1;
            h.userdata.ax3.children.children.thickness=2;
            h.userdata.ax3.children.children.foreground=4;
            h.userdata.init_plot=0;
        else
            // evalute f2 **************
            f2_data=flipdim(f2_data,2);

            // trimming f2
            N=round((h.userdata.t_max+h.userdata.t_min-h.userdata.t_shift)/h.userdata.t_step);
            SZ = size(h.userdata.t_data,2);

            if N >0 then
                f2_data=[f2_data(N+1:$),zeros(1,N)];
            elseif N<0
                f2_data=[zeros(1,abs(N)),f2_data(1:$-abs(N))];
            end
```

```
            f2_data(1)=0;
            f2_data($)=0;

            h.userdata.ax2.children.children.data=[h.userdata.t_data',f2_data'];

            // updating convolution plot
            N_conv=round((h.userdata.t_shift-h.userdata.t_min)/h.userdata.t_step);

            if ((N_conv <= SZ ) & (N_conv >= 1) )
                conv_data(N_conv:$)=0;
            elseif N_conv < 1 then
                conv_data(1:$)=0;
            end

            h.userdata.ax3.children.children.data=[h.userdata.t_data',conv_data'];

            f1_data=f1_data .* f2_data;
            h.userdata.ax1.children.children(1).data=[h.userdata.t_data',f1_data'];
        end

    else
        conv_gui();


    end

endfunction
//*************************************************************

//*************************************************************
```