

Scilab Textbook Companion for
Linear Systems And Signals
by B. P. Lathi¹

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

April 14, 2019

¹Funded 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	4
1 SIGNALS AND SYSTEMS	5
2 Time domain analysis of continuous time systems	24
3 Time domain analysis of discrete time systems	47
4 Laplace Transform	65
5 Discrete time analysis using Z transform	85
6 Continuous time signal analysis The Fourier series	100
7 Continuous time signal analysis The Fourier transform	119
8 Sampling	142
9 Fourier analysis of discrete time signals	154
10 State space analysis	167

List of Scilab Codes

Exa 0.b.1	express complex numbers in polar form	5
Exa 0.b.2	find Cartesian form of the given exponentials	5
Exa 0.b.3	find polar and Cartesian form of complex no	6
Exa 0.b.4	evaluate the given complex no operations . .	7
Exa 0.B.5	Cartesian and polar form	7
Exa 0.b.6	express given signals in single sinusoidal form	8
Exa 0.b.7	Cramers rule	9
Exa 0.b.8	Partial fraction	9
Exa 0.b.9	Partial fraction	10
Exa 0.b.10	Partial fraction	11
Exa 0.b.11	Partial fraction	12
Exa 0.b.12	Inverse of matrix	12
Exa 0.b.13	eigen value of matrix and exponential of the matrix	12
Exa 1.1	determine energy and power of given signals	13
Exa 1.3	show delayed and advanced signals	14
Exa 1.4	show compressed and expanded signals . . .	16
Exa 1.5	plot reversed signal	17
Exa 1.6	find expression for given signal and plot it .	19
Exa 1.7	find expression for the given signal and plot it	20
Exa 1.8	find even and odd component of the given sig- nal	21
Exa 2.1	Zero Input Response ZIR of the given systems	24
Exa 2.2	Finding loop current and initial values . . .	28
Exa 2.3	Finding impulse response of the system . . .	29
Exa 2.4	Finding impulse response of the system . . .	31
Exa 2.5	ZSR of given system	32
Exa 2.6	Finding loop current	33

Exa 2.7	Graphical convolution	35
Exa 2.8	Graphical convolution	36
Exa 2.9	Graphical convolution	38
Exa 2.10	Solving ODE of given system	40
Exa 2.11	Analysing system behaviour with different inputs	41
Exa 2.12	Finding loop current	44
Exa 3.1	energy and power of discrete signals	47
Exa 3.3	discrete signal representation	48
Exa 3.8	iterative solution of discrete time systems	49
Exa 3.9	iterative solution of discrete time systems	50
Exa 3.10	Zero Input Response of the various systems	51
Exa 3.11	impulse response	53
Exa 3.13	discrete convolution	54
Exa 3.14	Zero state response of discrete systems	55
Exa 3.15	discrete convolution	56
Exa 3.16	sliding tape discrete convolution	57
Exa 3.17	system response to auxiliary conditions	59
Exa 3.18	summation of given function	60
Exa 3.19	forced response	61
Exa 3.20	forced response	62
Exa 3.21	response of cascaded system	63
Exa 4.1	Laplace transform of exponential signal	65
Exa 4.2	Laplace transform of given signals	66
Exa 4.3	Inverse Laplace transform of given transfer functions	67
Exa 4.4	Laplace transform of given signals	68
Exa 4.5	Inverse Laplace transform of given transfer function	69
Exa 4.8	Time convolution using laplace transform	70
Exa 4.9	Initial and final values of laplace transform	70
Exa 4.10	solving system equation	71
Exa 4.11	Finding inductor current	72
Exa 4.12	Response of the given system	73
Exa 4.15	Finding loop current	73
Exa 4.16	Finding loop currents	74
Exa 4.17	Finding loop current	75
Exa 4.23	frequency response of the given system	75

Exa 4.24	frequency response of the given systems . . .	76
Exa 4.25	bode plots of the given transfer function . . .	77
Exa 4.26	bode plots of the given transfer function . . .	79
Exa 4.27	second order notch filter analysis	79
Exa 4.28	Inverse Laplace transform of given transfer function	80
Exa 4.29	Finding loop current in RC circuit	82
Exa 4.30	System response from transfer function . . .	82
Exa 4.31	System response from transfer function . . .	83
Exa 5.1	Z transform of the given signal	85
Exa 5.2	Z transform of the given signals	86
Exa 5.3	Inverse Z transform of the given transfer func- tions	87
Exa 5.5	Solving given system equation using z trans- form	88
Exa 5.6	solving system difference equation	89
Exa 5.10	System response to different inputs	90
Exa 5.12	Calculating maximum sampling interval or min- imum frequency	91
Exa 5.13	Calculating maximum sampling interval or min- imum frequency	92
Exa 5.14	frequency response of Bandpass filter	92
Exa 5.15	frequency response of Bandstop filter	93
Exa 5.16	frequency response of low pass filter	94
Exa 5.17	Z transform of the given signal	95
Exa 5.18	Inverse Z transform of the given function . .	96
Exa 5.19	ZSR of given system	97
Exa 5.20	ZSR of given system	98
Exa 6.1	Compact trigonometric Fourier series	100
Exa 6.2	Compact trigonometric Fourier series	101
Exa 6.3	Compact trigonometric Fourier series	103
Exa 6.4	Compact trigonometric Fourier series	104
Exa 6.5	exponential Fourier series	106
Exa 6.6	exponential Fourier series from trigonometric Fourier series	107
Exa 6.7	trigonometric Fourier series of impulse train	108
Exa 6.8	Harmonic distortion in audio amplifier . . .	111
Exa 6.9	dc value and rms value of rectified signal . .	112

Exa 6.10	Signal approximation with minimum error signal	114
Exa 6.11	Regeneration of signal from Fourier series coefficient	115
Exa 6.12	Legendre Fourier series	117
Exa 7.1	Fourier transform of given signal	119
Exa 7.2	Fourier transform of given rect signal	120
Exa 7.3	Fourier transform of given impulse signal	121
Exa 7.4	Inverse fourier transform of impulse signal	123
Exa 7.6	Fourier transform of given everlasting sinusoidal signal	124
Exa 7.9	Fourier transform of given unit step signal	126
Exa 7.10	Fourier transform of given Signum function	127
Exa 7.11	Duality property	128
Exa 7.12	Fourier transform of given signals	129
Exa 7.13	Fourier transform of time shifted signal	133
Exa 7.14	Fourier transform of time shifted rect signal	134
Exa 7.15	Fourier transform of modulated signal	135
Exa 7.17	Fourier transform of given triangle pulse	137
Exa 7.19	Filtered signals	138
Exa 7.20	Calculation of frequency range with signal energy	140
Exa 7.23	Toned modulated signals	140
Exa 8.1	Effects of different sampling rates	142
Exa 8.2	Practical sampling	145
Exa 8.5	Finding bit rate and sampling rate	147
Exa 8.7	Finding N_0 for DFT	148
Exa 8.8	Fourier transform of given exponential signal using DFT	149
Exa 8.9	Fourier transform of given pulse using DFT	150
Exa 8.10	Finding filtered output using DFT	151
Exa 9.1	Discrete time Fourier series	154
Exa 9.2	Discrete time Fourier series of periodic sampled gate function	155
Exa 9.3	Discrete time Fourier transform	156
Exa 9.4	Discrete time Fourier transform	158
Exa 9.5	DTFT of discrete time rectangular pulse	159
Exa 9.6	DTFT of discrete time rectangular pulse	160

Exa 9.9	DTFT of the given signal	161
Exa 9.10	DTFT of the given modulated signal	163
Exa 9.13	ZSR of LTID system	164
Exa 10.4	State space representation from transfer function	167
Exa 10.5	finding state vector	167
Exa 10.6	transfer function from state space equation	168
Exa 10.7	time domain method of solving the system	169
Exa 10.8	Determining exponential matrix	169
Exa 10.9	Finding new state equations	170
Exa 10.10	Diagonalized form of the state equations	170
Exa 10.11	controllability and observability of the system	171
Exa 10.12	finding output of the system	172
Exa 10.13	response of given system using z transform	172

Chapter 1

SIGNALS AND SYSTEMS

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

```
1  clc
2  clear
3
4  //defining the given complex numbers
5  a=complex(2,3);
6  b=complex(-2,1);
7  c=complex(-2,-3);
8  d=complex(1,-3);
9
10 printf("sl \t abs_value \t angle(degree)\n");//
    printing the abs_value and phase of all complex
    numbers for polar form
11 printf("a \t %f \t %f \t\n",abs(a),phasemag(a));
12 printf("b \t %f \t %f \t\n",abs(b),phasemag(b));
13 printf("c \t %f \t %f \t\n",abs(c),phasemag(c));
14 printf("d \t %f \t %f \t\n",abs(d),phasemag(d));
```

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

```

1  clc
2  clear
3
4  //defining all the given datas
5  a=2*exp(%i*%pi/3);
6  b=4*exp(-%i*3*%pi/4);
7  c=2*exp(%i*%pi/2);
8  d=3*exp(-%i*3*%pi);
9  e=2*exp(%i*4*%pi);
10 f=2*exp(-%i*4*%pi);
11
12 printf("s1 \t cartesian_form\n"); //showing all the
    cartesian forms
13 disp(" (a)      "+string(a));
14 disp(" (b)      "+string(b));
15 disp(" (c)      "+string(c));
16 disp(" (d)      "+string(d)); //scilab won't show
    exactly zero,so very small value can be regarded
    as 0 here
17 disp(" (e)      "+string(e));
18 disp(" (f)      "+string(f));

```

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

```

1  clc
2  clear
3
4  format(6)
5  z1=complex(3,4);
6  z2=complex(2,3);
7
8  mul=z1*z2;
9  div=z1/z2;
10 mag_mul=abs(mul); //finding magnitude of z1*z2
11 phas_mul=phasemag(mul); //finding angle of z1*z2 in

```

```

    degrees
12 mag_div=abs(div); //finding magnitude of z1/z2
13 phas_div=phasemag(div); //finding angle of z1/z2 in
    degrees
14
15 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
16 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

```

1 clc
2 clear
3
4 z1=2*exp(%i*%pi/4);
5 z2=8*exp(%i*%pi/3);
6
7 a=2*z1-z2;
8 b=1/z1;
9 c=z1/(z2)^2;
10 d=(z2)^(1/3);
11
12 disp(" a= ");disp(a)
13 disp(" b= ");disp(b)
14 disp(" c= ");disp(c)
15 disp(" d= ");disp(d)

```

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

```

1  clc
2  clear
3
4  w=input("enter the value of omega: ");//
    initialising w using value from users
5  X=(2+%i*w)/(3+%i*w*4);
6
7  rl_X=real(X);    //finding real part
8  img_X=imag(X);  //finding imaginary part
9  mag_X=abs(X);    //finding absolute part
10 phas_X=phasemag(X); //finding angle in degree
11
12 printf("\nX(w) in-\ncartesian form: real part= %f
    and imaginary part= %f\n\n",rl_X,img_X);
13 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

```

1  clc
2  clear
3
4  x1="cos(w0t)-sqrt(3)sin(w0t)";//given signal
5  a1=1;b1=-sqrt(3);
6  c1=sqrt(a1^2+b1^2);
7  theta_1=atan(-b1/a1);
8  theta_1=(180/%pi)*theta_1;//converting radian to
    degrees
9
10 x2="-3 cos(w0t)+4*sin(w0t)";//given signal
11 a2=-3;b2=4;
12 c2=sqrt(a2^2+b2^2);
13 theta_2=atan(-b2/a2);
14 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
15
16 //showing the signals as single sinusoids with
    values with accuracy upto 2 points
17 printf("(a) x1(t)= %.2f*cos(w0t+%.2f(deg))\n",c1,
    theta_1);
18 printf("(b) x2(t)= %.2f*cos(w0t%.2f(deg))\n",c2,
    theta_2);

```

Scilab code Exa 0.b.7 Cramers rule

```

1  clc
2  clear
3
4  A=[2,1,1;1,3,-1;1,1,1]; //co-efficient matrix
5  b=[3;7;1];             //b matrix in the equation Ax=
    b
6
7  //finding modified matrices to find unknown
    variables
8  A1=[b A(1:3,2:3)];
9  A2=[A(1:3,1) b A(1:3,3)];
10 A3=[A(1:3,1:2) b];
11
12 //finding unknown variables using Cramer's rule
13 x1=(det(A1)/det(A))
14 x2=det(A2)/det(A)
15 x3=det(A3)/det(A)
16
17 printf("using cramers rule ,we get: x1= %.0f , x2= %
    .0f , x3= %.0f \n",x1,x2,x3);

```

Scilab code Exa 0.b.8 Partial fraction

```

1  clc
2  clear
3
4  //for repeated roots use of pfss function is not
   efficient
5  //so manual calculation needed
6  s=%s
7  num=s^3+3*s^2+4*s+6; // given numerator
8  den1=(s+1); //given denominators
9  den2=(s+2);
10 den3=(s+3)^2;
11 root1=-1;root2=-2;root3=-3;
12
13 g1=num/(den2*den3);
14 g2=num/(den1*den3);
15 g3=num/(den1*den2);
16
17 k1=horner(g1,root1);           //finding co-
   efficient for (s+1)
18 k2=horner(g2,root2);           //finding co-
   efficient for (s+2)
19 k3=horner(derivat(g3),root3); //finding co-
   efficient for (s+3)
20 k4=horner(g3,root3);           //finding co-
   efficient for (s+3)^2
21
22 printf("the partial fractions are as below\n");
23 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

```

1  clc
2  clear
3

```

```

4 //pfss is efficient for non-repeated roots
5
6 num = -11+9*s + 2*s^2 ; //given numerator
7 den = (s+1)*(s-2)*(s+3); //given denominator
8 h2 = syslin('c',num/den);
9 d = pfss(h2);
10 disp("the obtained partial fractions are");
11 disp(d) //displaying the partial fractions

```

Scilab code Exa 0.b.10 Partial fraction

```

1 clc
2 clear
3
4 //for repeated roots use of pfss function is not
  efficient
5 s=%s
6 num=4*s^3+16*s^2+23*s+13;// given numerator
7 den1=(s+2);//given denominators
8 den2=(s+1)^3;
9 root1=-1;root2=-2;
10
11 g1=num/den1;
12 g2=num/den2;
13
14 k1=horner(g1,root1); //finding co
    -efficient for (s+1)^3
15 k2=horner(derivat(g1),root1); //finding co
    -efficient for (s+1)^2
16 k3=horner(derivat(derivat(g1)),root1)/2;//finding co
    -efficient for (s+1)
17 k4=horner(g2,root2); //finding co
    -efficient for (s+2)
18
19 printf("the partial fractions are as below\n");

```



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

Scilab code Exa 0.b.11 Partial fraction

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

Scilab code Exa 0.b.12 Inverse of matrix

```
1 clc
2 clear
3
4 A=[2 1 1;1 2 3;3 2 1];
5 disp("the inverse matrix is");
6 disp(inv(A));
```

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

```
1 clc
2 clear
3
```

```

4 //finding eigen values of the given matrix
5 A=[0 1;-2 -3];
6 egn=spec(A);
7 disp("the eigen values are ");disp(egn);
8
9 //finding exponential of a matrix
10 [N,d]=coeff(A);
11 b=N/d;
12 b=pfss(b);
13 disp("the laplace transform of each elements of the
      exponential matrix(e^(At)) are given as partial
      fractions below");
14 disp("the ILT of this will result in exponential
      matrix function")
15 ;disp(b);
16 //let P=exponential matrix(e^(At))
17 //then here from b we get
18 //P(1)=inv_laplace(2/(1+s)-1/(s+2))=2e^(-t)-e^(-2t)
19 //P(2)=inv_laplace(-2/(1+s)+2/(s+2))=-2e^(-t)+2e^(-2
      t)
20 //P(3)=inv_laplace(1/(1+s)-1/(s+2))=e^(-t)-e^(-2t)
21 //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

```

1 clc
2 clear
3
4 //defining unit step function
5 function y=u(x)
6     y=sign((sign(x)+1))
7 endfunction
8
9 format(12)

```

```

10 dt=0.0001; //defining increment in time vector
11 t=-2:dt:20;
12 f=2.*(u(t+1)-u(t))+2.*exp(-t/2).*u(t); //defining
    the energy signal in fig(a)
13 g=t.*(u(t+1)-u(t-1)); //defining one
    cycle of the periodoc signal from fig(b)
14
15 E=sum((f.^2)*dt); //calculating energy of f
16 P=0.5*sum((g.^2)*dt); //calculating power of g
17 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
18 printf("\nthe power of the signal b is %f",P);

```

Scilab code Exa 1.3 show delayed and advanced signals

```

1 clc
2 clear
3
4 //defining unit step function
5 function y=u(x)
6     y=sign((sign(x)+1))
7 endfunction
8
9 deff(' [y]=f(x) ', 'y=exp(-2.*x).*u(x) '); //defining the
    signal function
10 t=-2:0.01:7;
11
12 subplot(3,1,1);plot(t,f(t));set(gca()," grid",[1,1]);
    xtitle("original signal");//plotting original
    signal
13 subplot(3,1,2);plot(t,f(t-1));set(gca()," grid"
    ,[1,1]);xtitle("delayed signal");//plotting

```

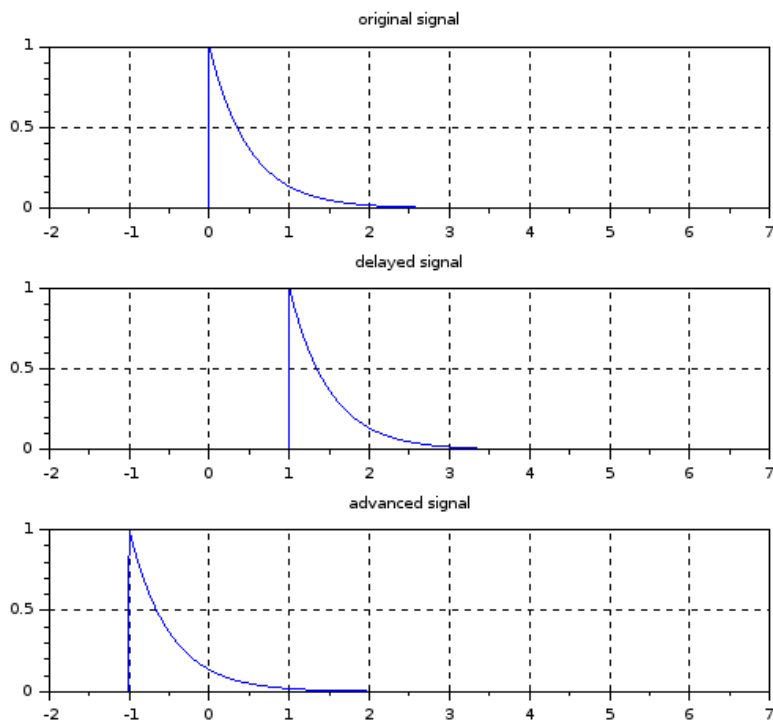


Figure 1.1: show delayed and advanced signals

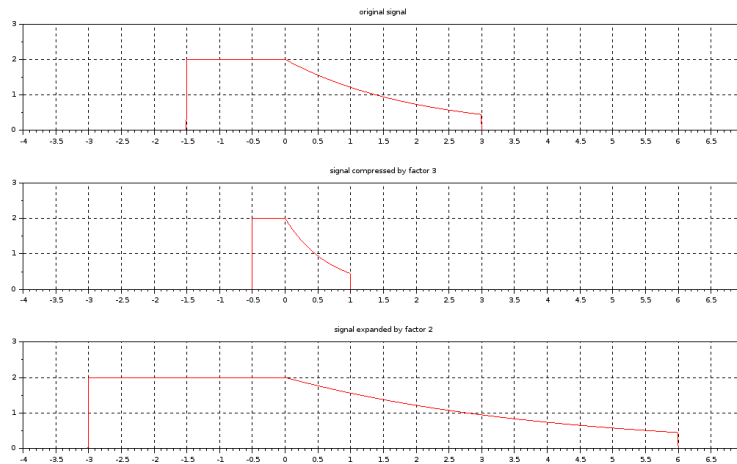


Figure 1.2: show compressed and expanded signals

```

14 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

```

1  clc
2  clear
3
4  //defining unit step function
5  function y=u(x)
6      y=sign((sign(x)+1))
7  endfunction
8
9  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
10 t=-4:0.01:7;

```

```

11
12 subplot(3,1,1);
13 h=gca();
14 zoom_rect(h,[-4 0 7 3]); //setting limits of the
    plot
15 plot(t,f(t),'r'); //plotting original
    signal
16 h.grid=[1,1];
17 xtitle("original signal");
18
19 subplot(3,1,2);
20 h=gca();
21 zoom_rect(h,[-4 0 7 3]);
22 plot(t,f(3.*t),'r'); //plotting
    compressed signal
23 h.grid=[1,1];
24 xtitle("signal compressed by factor 3");
25
26 subplot(3,1,3);
27 h=gca();
28 zoom_rect(h,[-4 0 7 3]);
29 plot(t,f(t./2),'r'); //plotting expanded
    signal
30 h.grid=[1,1];
31 xtitle("signal expanded by factor 2");

```

Scilab code Exa 1.5 plot reversed signal

```

1 clc
2 clear
3
4 //defining unit step function
5 function y=u(x)

```

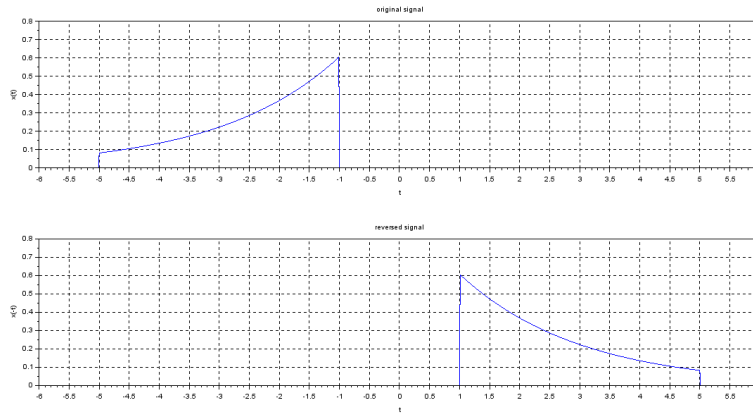


Figure 1.3: plot reversed signal

```

6     y=sign((sign(x)+1))
7     endfunction
8
9     deff(' [y]=f(x) ', 'y=exp(x./2).*(u(x+5)-u(x+1)) '); //
        defining the signal function
10    t=-6:0.01:6;
11
12    subplot(2,1,1);
13    h=gca();
14    zoom_rect(h,[-6,0,6,0.8]); //setting limits of the
        plot
15    plot(t,f(t)); //plotting original signal
16    h.grid=[1,1];
17    xtitle("original signal","t","x(t)");
18
19    subplot(2,1,2);
20    h=gca();
21    zoom_rect(h,[-6,0,6,0.8]);
22    plot(t,f(-t)); //plotting reversed
        signal
23    h.grid=[1,1];
24    xtitle("reversed signal","t","x(-t)");

```

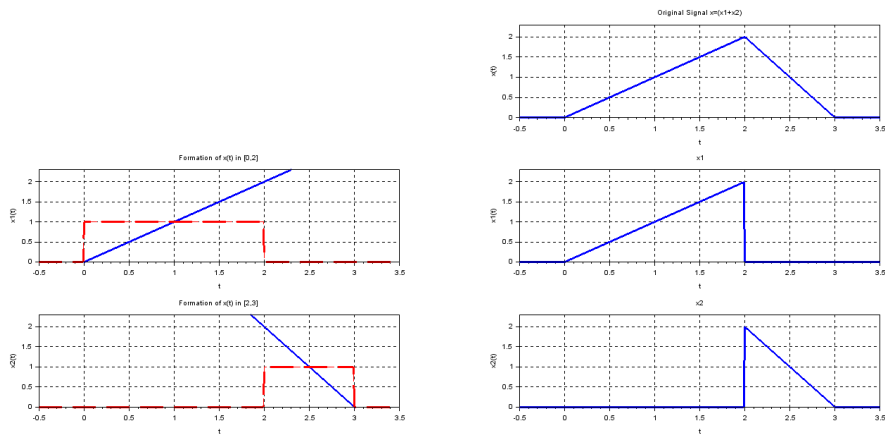


Figure 1.4: find expression for given signal and plot it

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

```

1  clc
2  clear
3
4  //defining unit step function
5  function y=u(x)
6      y=sign((sign(x)+1))
7  endfunction
8
9  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
10 t=-0.5:0.01:3.5;
11
12 subplot(3,2,2);plot(t,f(t)); //plotting the signal
13 h=gca();
14 h.grid=[1,1]

```



```

15 xtitle(" Original Signal  $x=(x_1+x_2)$ ", "t", "x(t)")
16 zoom_rect([-0.5, 0, 3.5, 2.3]);
17
18 subplot(3,2,3);plot(t,t);plot(t,1.*(u(t)-u(t-2)), '—r');
19 h=gca();
20 h.grid=[1,1]
21 xtitle(" Formation of  $x(t)$  in  $[0,2]$ ", "t", "x1(t)");
22 zoom_rect([-0.5, 0, 3.5, 2.3]);
23
24 subplot(3,2,4);plot(t,t.*(u(t)-u(t-2)));
25 h=gca();
26 h.grid=[1,1]
27 xtitle("x1", "t", "x1(t)");
28 zoom_rect([-0.5, 0, 3.5, 2.3]);
29
30 subplot(3,2,5);plot(t,-2*(t-3));plot(t,1.*(u(t-2)-u(t-3)), '—r');
31 h=gca();
32 h.grid=[1,1]
33 xtitle(" Formation of  $x(t)$  in  $[2,3]$ ", "t", "x2(t)");
34 zoom_rect([-0.5, 0, 3.5, 2.3]);
35
36 subplot(3,2,6);plot(t,(-2*(t-3)).*(u(t-2)-u(t-3)));
37 h=gca();
38 h.grid=[1,1]
39 xtitle("x2", "t", "x2(t)");
40 zoom_rect([-0.5, 0, 3.5, 2.3]);

```

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

```

1 clc
2 clear

```

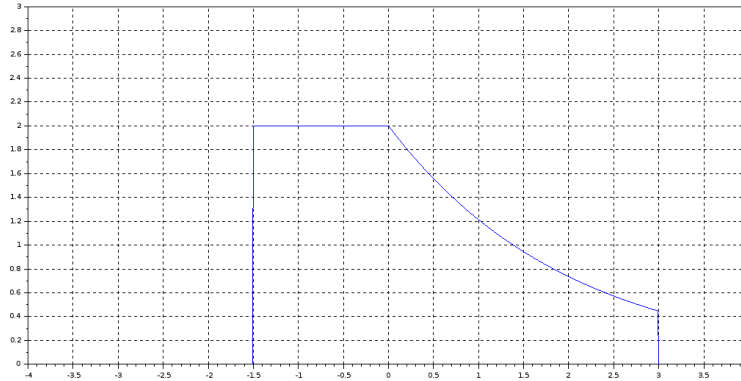


Figure 1.5: find expression for the given signal and plot it

```

3
4 //defining unit step function
5 function y=u(x)
6     y=sign((sign(x)+1))
7 endfunction
8
9 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
10 t=-4:0.01:4;
11
12 plot(t,f(t)); //plotting the signal
13 h=gca();
14 h.grid=[1,1];
15 zoom_rect(h,[-4 0 4 3])

```

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

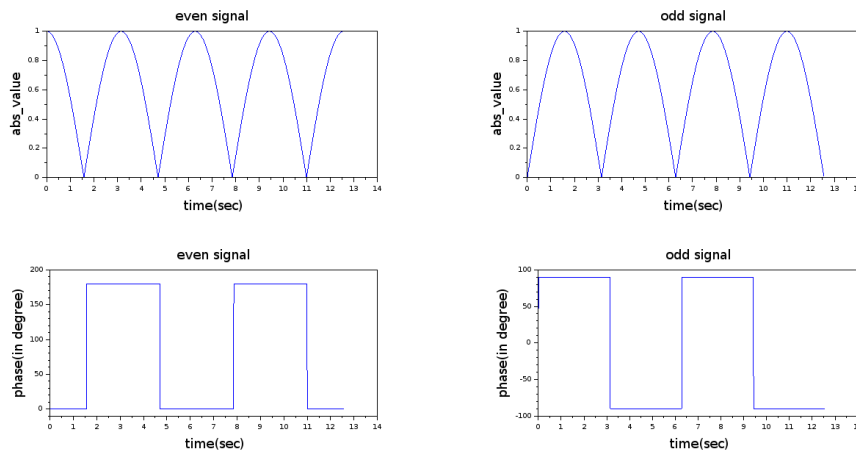


Figure 1.6: find even and odd component of the given signal

```

1  clc
2  clear
3
4  t=0:0.01:4*%pi;
5  odd=0.5.*(exp(%i.*t)-exp(-%i.*t)); //finding odd
   component of given signal
6  even=0.5.*(exp(%i.*t)+exp(-%i.*t)); //finding even
   component of given signal
7
8  subplot(2,2,1);plot(t,abs(even)); //plotting the even
   part
9  title("even signal","fontsize",4);
10 xlabel("time(sec)","fontsize",4),ylabel("abs_value",
    "fontsize",4);
11
12 subplot(2,2,3);plot(t,phasemag(even));
13 title("even signal","fontsize",4);
14 xlabel("time(sec)","fontsize",4);ylabel("phase(in
    degree)","fontsize",4);
15 h=gca();
16 zoom_rect(h,[0,-10,14,200]);
17

```

```
18 subplot(2,2,2);plot(t,abs(odd));//plotting the odd
    part
19 xlabel("time(sec)","fontsize",4),ylabel("abs_value",
    "fontsize",4);
20 title("odd signal","fontsize",4);
21
22 subplot(2,2,4);plot(t,phasemag(odd));
23 xlabel("time(sec)","fontsize",4);ylabel("phase(in
    degree)","fontsize",4);
24 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

```
1  clc
2  clear
3  close;
4  funcprot(0)
5
6  //*****part a(non-repeated real
   //      roots)*****
7  //-----
8  //-----
9  function dy= f(t,y)
10     dy=zeros(2,1);
11     dy(1)=y(2);
12     dy(2)=-3*y(2)-2*y(1);
13 endfunction
14
15 y0=[0;-5];
16 t0=0;
17 t=linspace(0,10,500);
18 y=ode(y0,t0,t,f); //solving the 2nd order ode
```

```

        system equation
19
20 figure(1)
21 subplot(1,3,1);plot(t,y(1,:));      //plotting the
    obtained result
22 title("ZIR of the system(non-repeated roots)_part_a"
    ,"fontsize",4);
23 xlabel("time","fontsize",4);
24 ylabel("y(ZIR)","fontsize",4);
25
26 subplot(1,3,2);plot(t,-5.*exp(-t)+5.*exp(-2.*t));//
    plotting the estimated answer for comparison
27 title("estimated ans for comparison_part a","
    fontsize",4);
28 xlabel("time","fontsize",4);
29 ylabel("-5.*exp(-t)+5.*exp(-2.*t)","fontsize",4);
30
31 //


---


32 //


---


33
34
35
36 //*****part b(repeated real
    roots)*****
37 function dy= f(t,y)
38     dy=zeros(2,1);
39     dy(1)=y(2);
40     dy(2)=-6*y(2)-9*y(1);
41 endfunction
42
43 y0=[3;-7];
44 t0=0;
45 t=linspace(0,10,500);
46 y=ode(y0,t0,t,f);    //solving the 2nd order ode

```

```

        system equation
47
48 subplot(1,3,3);plot(t,y(1,:)); //plotting the
    obtained result
49 title("ZIR of the system(repeated roots)_part b",
    fontsize",4);
50 xlabel("time", "fontsize",4);
51 ylabel("y(ZIR)", "fontsize",4);
52
53 figure(2)
54 subplot(1,3,1);plot(t,(3+2*t).*exp(-3*t)); //plotting
    the estimated answer for comparision
55 zoom_rect([0,-0.5,15,3])
56 title("estimated ans for comparison-part b",
    fontsize",4);
57 xlabel("time", "fontsize",4);
58 ylabel("(3+2*t).*exp(-3*t)", "fontsize",4);
59
60 //


---


61 //


---


62
63
64
65 //*****part c(non-repeated imaginary
    roots)*****8
66 function dy= f(t,y)
67     dy=zeros(2,1);
68     dy(1)=y(2);
69     dy(2)=-4*y(2)-40*y(1);
70 endfunction
71
72 y0=[2;16.78];
73 t0=0;
74 t=linspace(0,10,500);

```

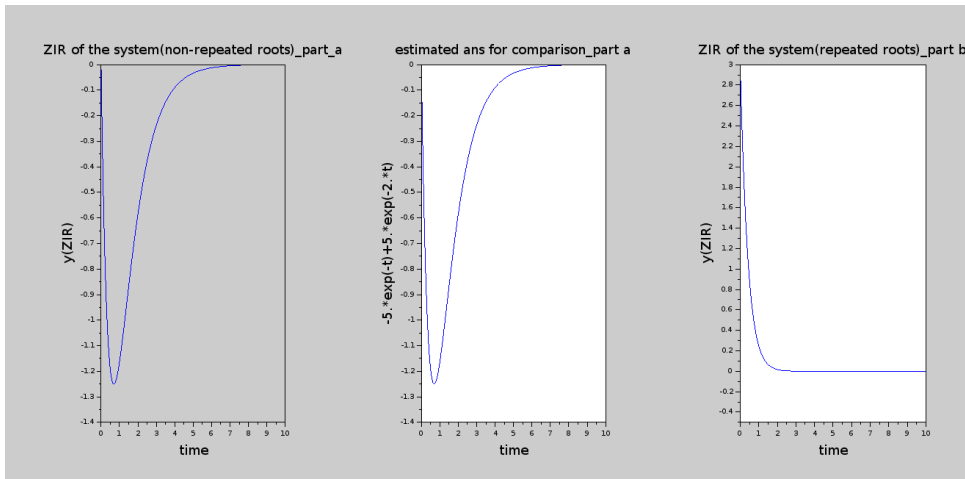


Figure 2.1: Zero Input Response ZIR of the given systems

```

75 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
76
77 subplot(1,3,2);plot(t,y(1,:)); //plotting the
    obtained result
78 title("ZIR of the system(imaginary roots)_part c",
    fontsize",4);
79 xlabel("time","fontsize",4);
80 ylabel("y(ZIR)","fontsize",4);
81
82
83 subplot(1,3,3);plot(t,4*exp(-2*t).*cos(6*t-%pi/3));
    //plotting the estimated answer for comparison
84 //zoom_rect([0,-0.5,15,3])
85 title("estimated ans for comparison_part c",
    fontsize",4);
86 xlabel("time","fontsize",4);
87 ylabel("4*exp(-2*t).*cos(6*t-%pi/3)","fontsize",4);

```

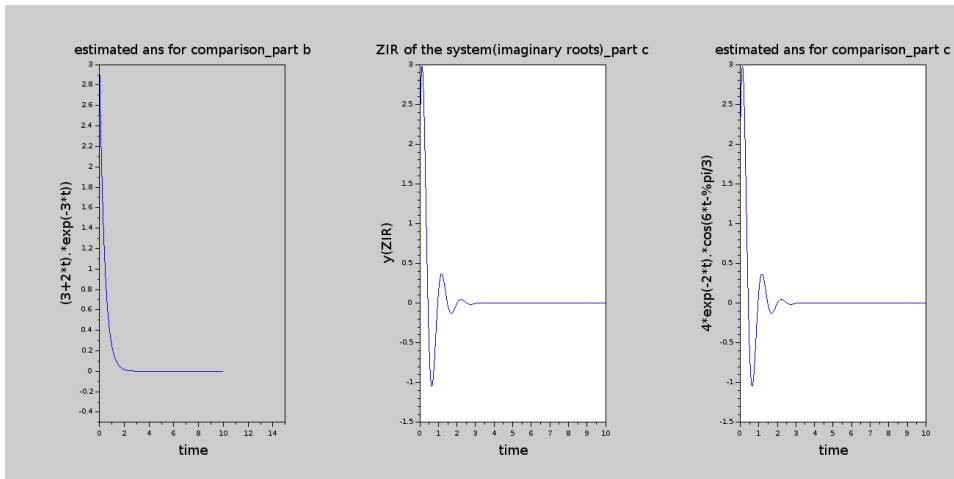


Figure 2.2: Zero Input Response ZIR of the given systems

Scilab code Exa 2.2 Finding loop current and initial values

```

1  clc
2  clear
3  close;
4
5  function dy= f(t,y)
6      dy=zeros(2,1);
7      dy(1)=y(2);
8      dy(2)=-3*y(2)-2*y(1);
9  endfunction
10
11 y0=[0;-5];
12 t0=0;
13 t=linspace(0,15,500);

```

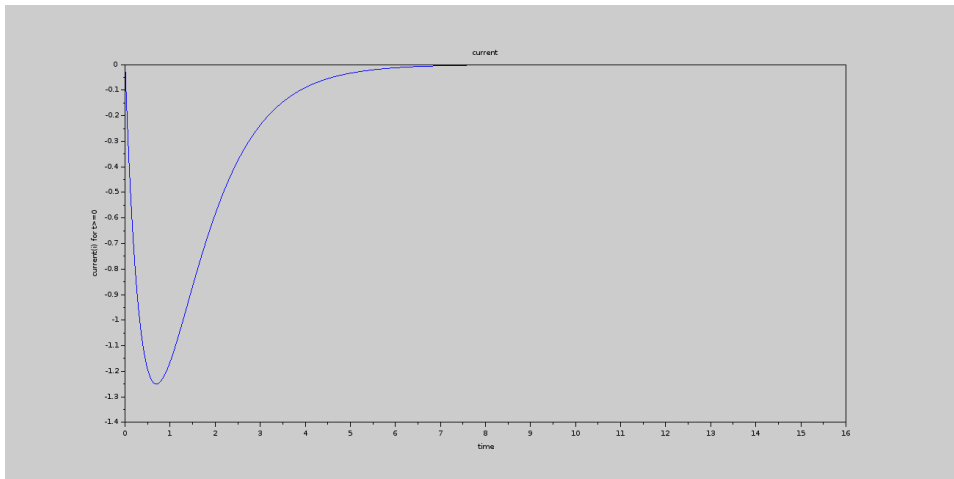


Figure 2.3: Finding loop current and initial values

```

14 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
15 v_0=5;
16 y_0=y0(1);
17 dy_0= 10-v_0-3*y_0;//eq obtained from circuit
18
19 figure(1)
20 plot(t,y(1,:)); //plotting the obtained result
21 xtitle("current ","time","current(i) for t>=0");
22 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

```

1 clc
2 clear
3 close;
4

```

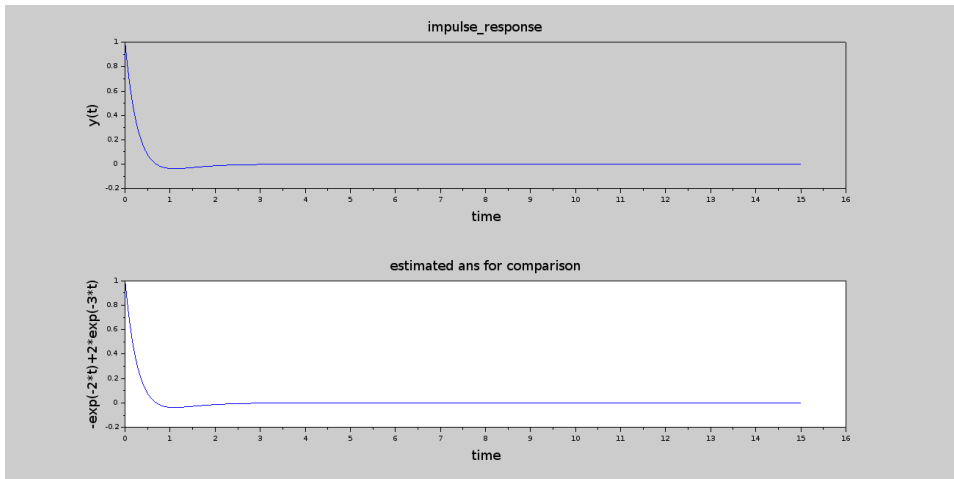


Figure 2.4: Finding impulse response of the system

```

5 function dy= f(t,y)
6     dy=zeros(2,1);
7     dy(1)=y(2);
8     dy(2)=-5*y(2)-6*y(1);
9 endfunction
10
11 y0=[1;-4]; //initial conditions for impulse response
12 t0=0;
13 t=linspace(0,15,500);
14 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
15
16 figure(1)
17 subplot(2,1,1);plot(t,y(1,:)); //plotting the
    obtained result
18 title("impulse_response","fontsize",4);
19 xlabel("time","fontsize",4);
20 ylabel("y(t)","fontsize",4);
21
22 //checking if the obtained ans is correct
23
24 subplot(2,1,2);plot(t,-exp(-2*t)+2*exp(-3*t)); //

```

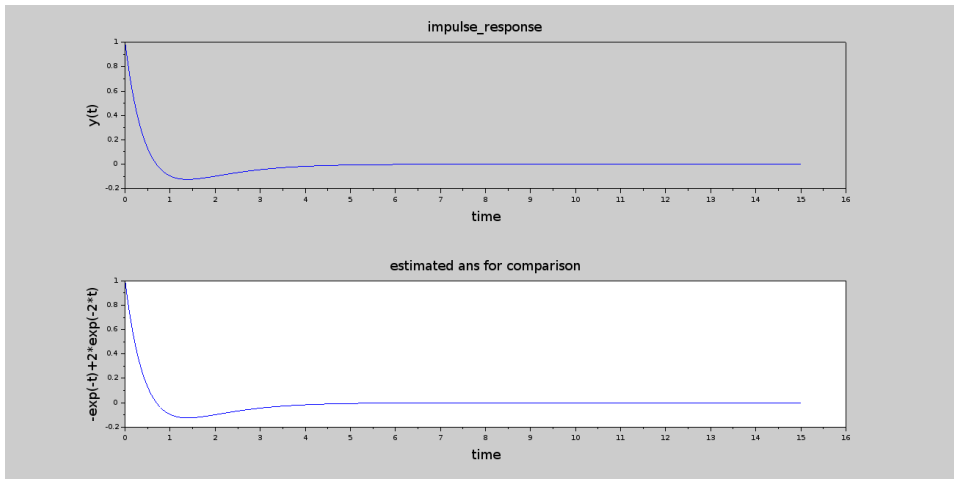


Figure 2.5: Finding impulse response of the system

```

plotting the estimated answer for comparison
25 title("estimated ans for comparison", "fontsize", 4);
26 xlabel("time", "fontsize", 4);
27 ylabel("-exp(-2*t)+2*exp(-3*t)", "fontsize", 4);

```

Scilab code Exa 2.4 Finding impulse response of the system

```

1  clc
2  clear
3  close;
4
5  function dy= f(t,y)
6      dy=zeros(2,1);
7      dy(1)=y(2);
8      dy(2)=-3*y(2)-2*y(1);
9  endfunction
10

```

```

11 y0=[1;-3]; //initial conditions for impulse response
    calculated from given system equation
12 t0=0;
13 t=linspace(0,15,500);
14 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
15
16 figure(1)
17 subplot(2,1,1);plot(t,y(1,:)); //plotting the
    obtained result
18 title("impulse_response","fontsize",4);
19 xlabel("time","fontsize",4);
20 ylabel("y(t)","fontsize",4);
21
22 //checking if the obtained ans is correct
23
24 subplot(2,1,2);plot(t,-exp(-t)+2*exp(-2*t)); //
    plotting the estimated answer for comparision
25 title("estimated ans for comparision","fontsize",4);
26 xlabel("time","fontsize",4);
27 ylabel("-exp(-t)+2*exp(-2*t)","fontsize",4);

```

Scilab code Exa 2.5 ZSR of given system

```

1 clc
2 clear
3 close;
4
5 t=[0:0.1:5];
6 // Defining unit step function
7     function y=u(x)
8         y=sign((sign(x)+1))
9     endfunction

```

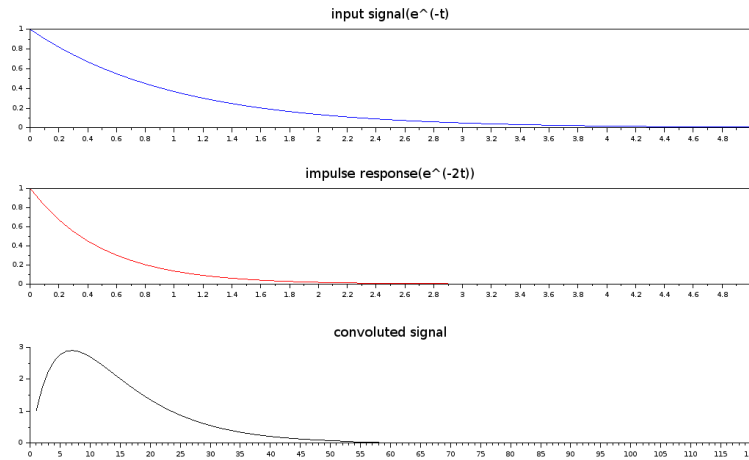


Figure 2.6: ZSR of given system

```

10
11 x=exp(-t).*u(t); //defining the input function
12 h=exp(-2*t).*u(t); //defining the impulse response
13
14 subplot(3,1,1);plot(t,x, 'b');
15 title("input signal(e^(-t))", "fontsize",4);
16
17 subplot(3,1,2);plot(t,h, 'r');
18 title("impulse response(e^(-2t))", "fontsize",4);
19
20 [z]=convol(x,h); //operating convolution operation
21 subplot(3,1,3);
22 plot2d(z);
23 title("convoluted signal", "fontsize",4);
24 printf("the obtained graph should be iinterpolated
    back to t=0 to get complete ZSR")

```

Scilab code Exa 2.6 Finding loop current

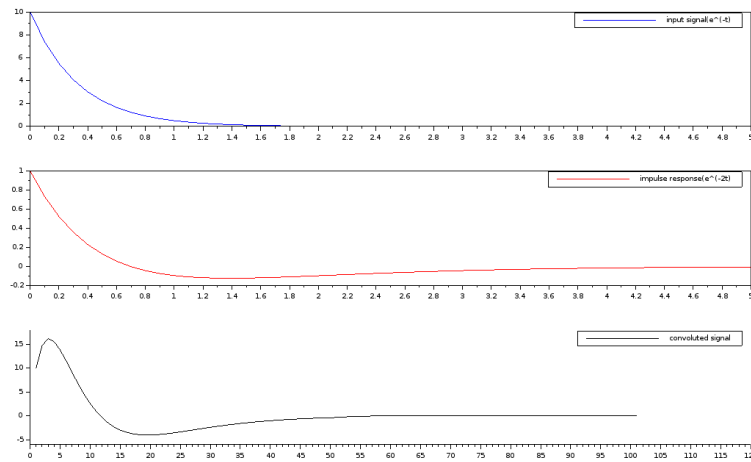


Figure 2.7: Finding loop current

```

1  clc
2  clear
3  close;
4
5  t=[0:0.1:5];
6  // Defining unit step function
7      function y=u(x)
8          y=sign((sign(x)+1))
9      endfunction
10
11 x=10.*exp(-3*t).*u(t); //defining the input function
12 h=(2.*exp(-2*t)-exp(-t)).*u(t); //defining the
    impulse response
13
14 subplot(3,1,1);plot(t,x,'b');
15 legend("input signal(e-t)");
16
17 subplot(3,1,2);plot(t,h,'r');
18 legend("impulse response(e-2t)");
19
20 [z]=convol(x,h); //operating convolution operation
21 subplot(3,1,3);

```

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

check Appendix [AP 3](#) for dependency:

conv_gui.sci

Scilab code Exa 2.7 Graphical convolution

```
1
2 clc
3 close;
4
5
6 //execute the dependency file first that is needed
  for graphical convolution
7 //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"
8
9 //all the inputs should be given in the form of
  strings
10
11 t='[-5:0.02:5]';
12 f1='exp(-t).*u(t)'; //defining x(t)
13 f2='exp(-2*t).*u(t)'; //defining h(t)
14 conv_gui(f1,f2,t);
15
16
17 //for analysis goto options > illustration > move
  right in the plot window
```

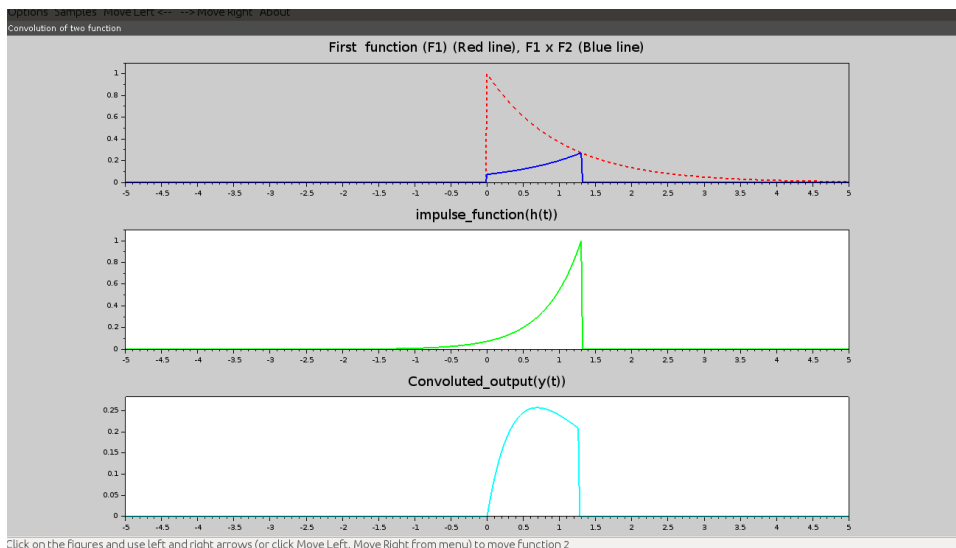


Figure 2.8: Graphical convolution

check Appendix [AP 3](#) for dependency:

`conv_gui.sci`

Scilab code Exa 2.8 Graphical convolution

```

1
2 clc
3 close;
4
5 //execute the dependency file first that is needed
   for graphical convolution
6 //exec ('/home/satyajit/Desktop/my-octave/
   scilab_project/chapter_2/conv_gui.sci ', -1)

```

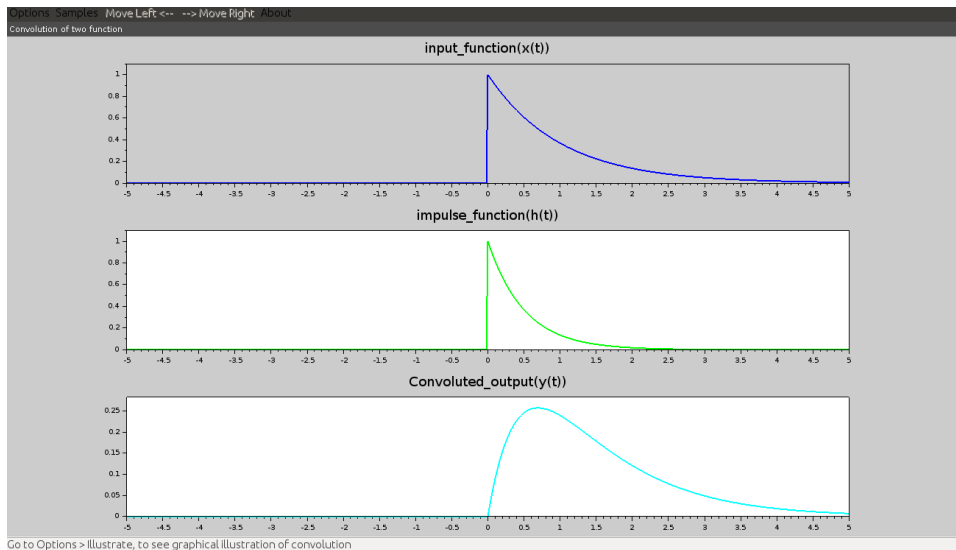


Figure 2.9: Graphical convolution

```

7 //change your path to the location of the file "
  conv_gui.sci"
8
9
10 //all the inputs should be given in the form of
    strings
11
12 t = '[-5:0.02:5]';
13 f1 = 'u(t)'; //defining x(t)
14 f2 = '-2*exp(2*t).*u(-t)+2*exp(-t).*u(t)'; //defining
    h(t)
15 conv_gui(f1,f2,t);
16
17
18 //for analysis goto options > illustration > move
    right in the plot window

```

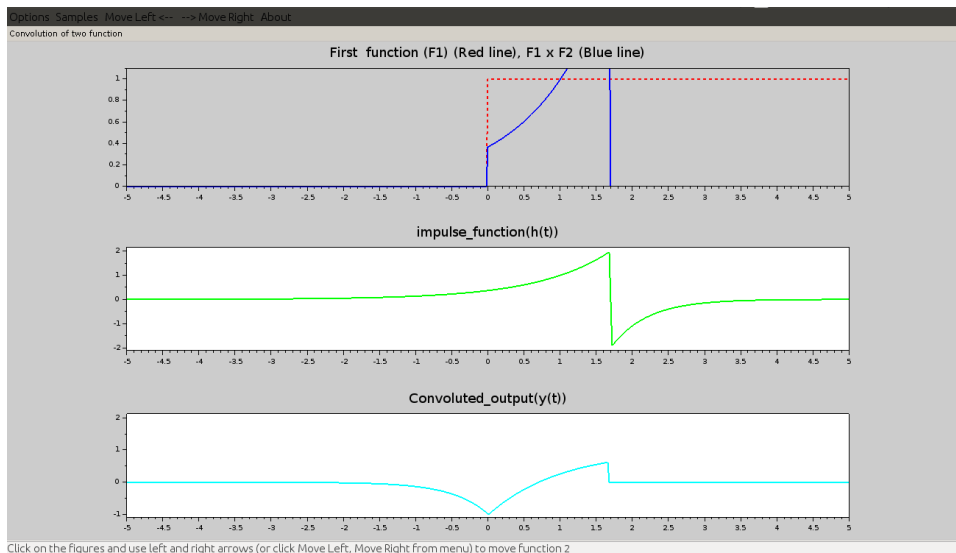


Figure 2.10: Graphical convolution

check Appendix [AP 3](#) for dependency:

`conv_gui.sci`

Scilab code Exa 2.9 Graphical convolution

```

1
2 clc
3 close;
4
5 //execute the dependency file first that is needed
  for graphical convolution
6 //exec ('/home/satyajit/Desktop/my-octave/
  scilab-project/chapter_2/conv_gui.sci ', -1)

```

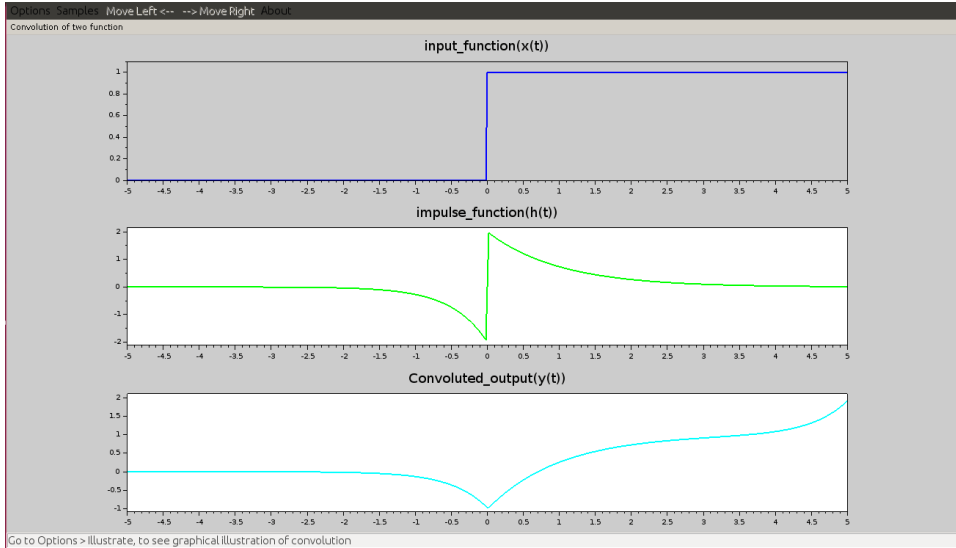


Figure 2.11: Graphical convolution

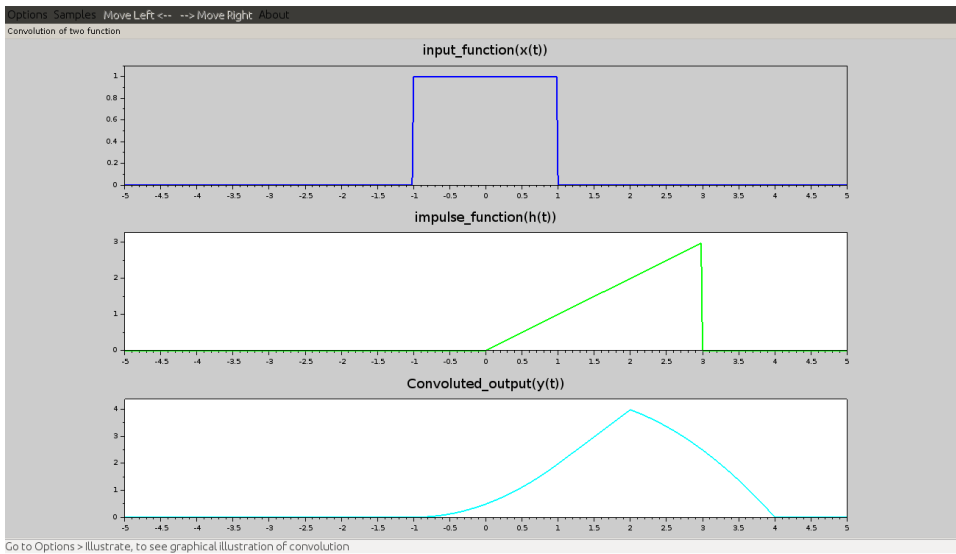


Figure 2.12: Graphical convolution

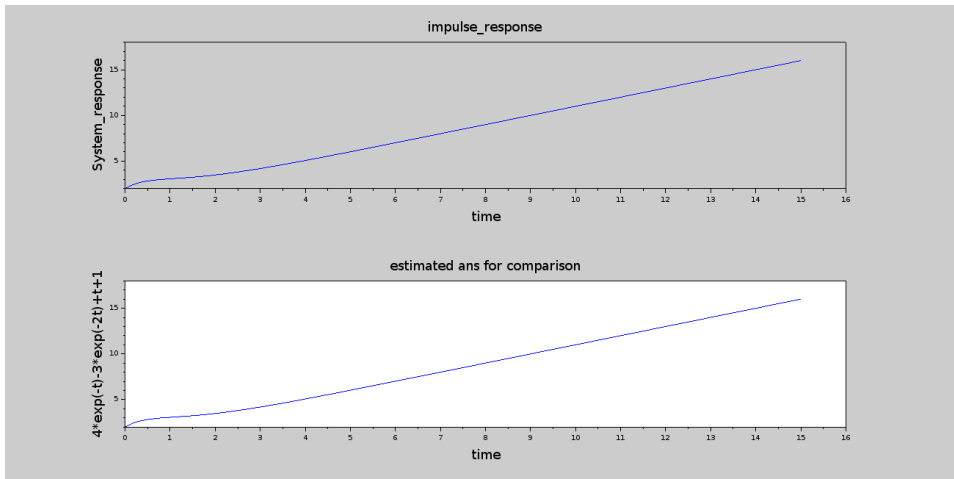


Figure 2.13: Solving ODE of given system

```

7 //change your path to the location of the file "
  conv_gui.sci"
8
9
10 //all the inputs should be given in the form of
    strings
11
12 t= ' [-5:0.02:5] ' ;
13 f1= ' u(t+1)-u(t-1) ' ; //defining x(t)
14 f2= ' t.*(u(t)-u(t-3)) ' ; //defining h(t)
15 conv_gui(f1,f2,t);
16
17
18 //for analysis goto options > illustration > move
    right in the plot window

```

Scilab code Exa 2.10 Solving ODE of given system

```

1  clc
2  clear
3  close;
4
5  function dy= f(t,y)
6      dy=zeros(2,1);
7      dy(1)=y(2);
8      dy(2)=-3*y(2)-2*y(1)+2*t+5;
9  endfunction
10
11 y0=[2;3]; //initial conditions for impulse response
12 t0=0;
13 t=linspace(0,15,500);
14 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
15
16 figure(1)
17 subplot(2,1,1);plot(t,y(1,:)); //plotting the
    obtained result
18 title("impulse_response","fontsize",4);
19 xlabel("time","fontsize",4);
20 ylabel("System_response","fontsize",4);
21
22
23 //checking if the obtained ans is correct
24
25 subplot(2,1,2);plot(t,4*exp(-t)-3*exp(-2*t)+t+1); //
    plotting the estimated answer for comparision
26 title("estimated ans for comparision","fontsize",4);
27 xlabel("time","fontsize",4);
28 ylabel("4*exp(-t)-3*exp(-2t)+t+1","fontsize",4);

```

Scilab code Exa 2.11 Analysing system behaviour with different inputs

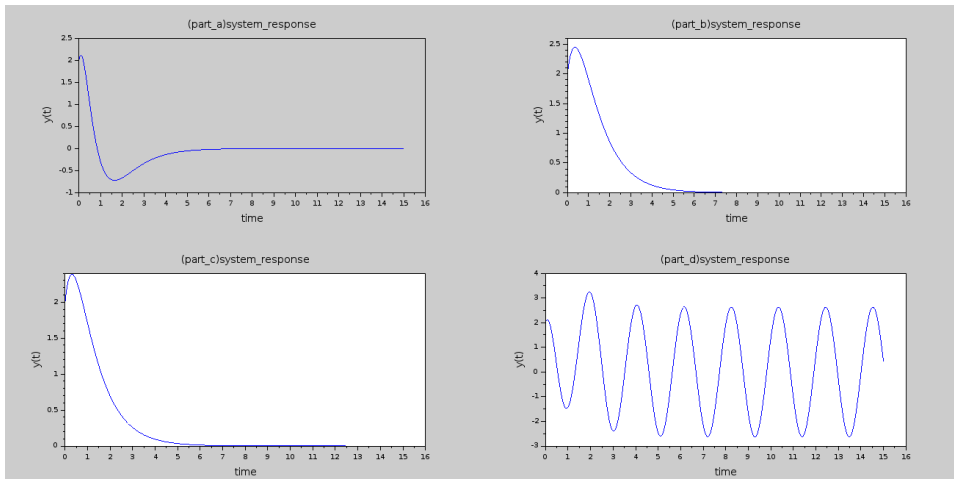


Figure 2.14: Analysing system behaviour with different inputs

```

1  clc
2  clear
3  close;
4
5  y0=[2;3]; //initial conditions for impulse response
6  t0=0;
7  t=linspace(0,15,500);
8
9  //***** part a (x(t)=10e-3t)
   *****
10 function dy= f(t,y)
11     dy=zeros(2,1);
12     dy(1)=y(2);
13     dy(2)=-3*y(2)-2*y(1)-30*exp(-3*t);
14 endfunction
15
16 y1=ode(y0,t0,t,f); //solving the 2nd order ode
   system equation
17
18 figure(1)
19 subplot(2,2,1);plot(t,y1(1,:)); //plotting the
   obtained result

```

```

20 title("(part_a)system_response","fontsize",3);
21 xlabel("time","fontsize",3);
22 ylabel("y(t)","fontsize",3);
23
24
25 //*****part-b(x(t)=5)
    *****
26 function dy= f(t,y)
27     dy=zeros(2,1);
28     dy(1)=y(2);
29     dy(2)=-3*y(2)-2*y(1);
30 endfunction
31
32 y2=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
33
34 subplot(2,2,2);plot(t,y2(1,:)); //plotting the
    obtained result
35 title("(part_b)system_response","fontsize",3);
36 xlabel("time","fontsize",3);
37 ylabel("y(t)","fontsize",3);
38
39
40
41 //*****part-c(x(t)=e^-2t)
    *****
42 function dy= f(t,y)
43     dy=zeros(2,1);
44     dy(1)=y(2);
45     dy(2)=-3*y(2)-2*y(1)-2*exp(-2*t);
46 endfunction
47
48 y3=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
49
50 subplot(2,2,3);plot(t,y3(1,:)); //plotting the
    obtained result
51 title("(part_c)system_response","fontsize",3);

```



```

52 xlabel("time","fontsize",3);
53 ylabel("y(t)","fontsize",3);
54
55
56
57 //*****part-d(x(t)=10cos(3t+pi
    /3)*****
58 function dy= f(t,y)
59     dy=zeros(2,1);
60     dy(1)=y(2);
61     dy(2)=-3*y(2)-2*y(1)-30*sin(3*t+%pi/3);
62 endfunction
63
64 y4=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
65
66 subplot(2,2,4);plot(t,y4(1,:)); //plotting the
    obtained result
67 title("(part_d)system_response","fontsize",3);
68 xlabel("time","fontsize",3);
69 ylabel("y(t)","fontsize",3);

```

Scilab code Exa 2.12 Finding loop current

```

1  clc
2  clear
3  close;
4
5  function dy= f(t,y)
6      dy=zeros(2,1);
7      dy(1)=y(2);
8      dy(2)=-3*y(2)-2*y(1)-30*exp(-3*t);
9  endfunction

```

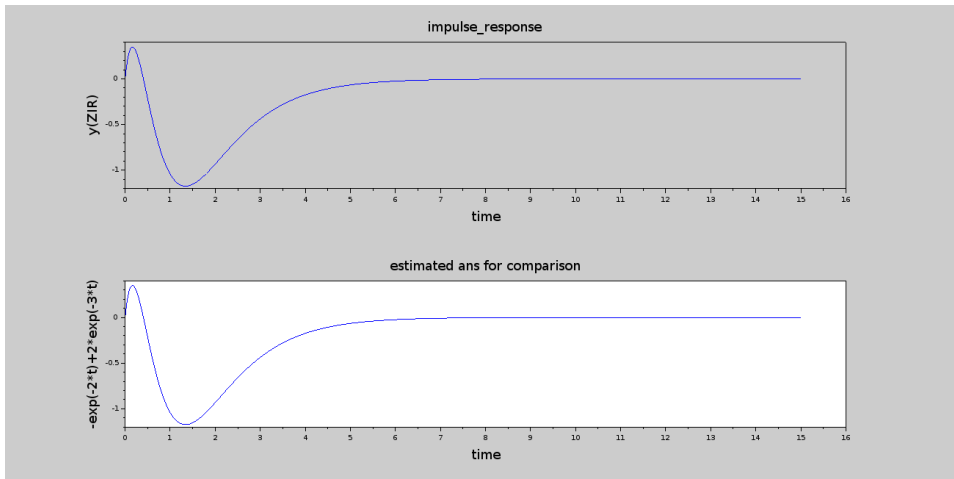


Figure 2.15: Finding loop current

```

10
11 y0=[0;5]; //initial conditions for impulse response
12 t0=0;
13 t=linspace(0,15,500);
14 y=ode(y0,t0,t,f); //solving the 2nd order ode
    system equation
15
16 figure(1)
17 subplot(2,1,1);plot(t,y(1,:)); //plotting the
    obtained result
18 title("impulse_response","fontsize",4);
19 xlabel("time","fontsize",4);
20 ylabel("y(ZIR)","fontsize",4);
21
22
23 //checking if the obtained ans is correct
24
25 subplot(2,1,2);plot(t,-10*exp(-t)+25*exp(-2*t)-15*
    exp(-3*t)); //plotting the estimated answer for
    comparison
26 title("estimated ans for comparison","fontsize",4);
27 xlabel("time","fontsize",4);

```

```
28 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

```
1  clc
2  clear
3
4  n=[0:1:5];
5  // Defining unit step function
6  function y=u(x)
7      y=sign((sign(x)+1))
8  endfunction
9
10 f=n.*(u(n)-u(n-6));//defining the given function
11 subplot(1,2,1);plot2d3(n,f);
12 title("figure (a)", "fontsize",4);
13 f2=repmat(f,1,5);
14 n1=0:1:length(f2)-1;
15 subplot(1,2,2);plot2d3(n1,f2,[5]);title("figure (b)",
    "fontsize",4);
16 energy=sum(f.^2);//finding energy of signal in fig(a)
```

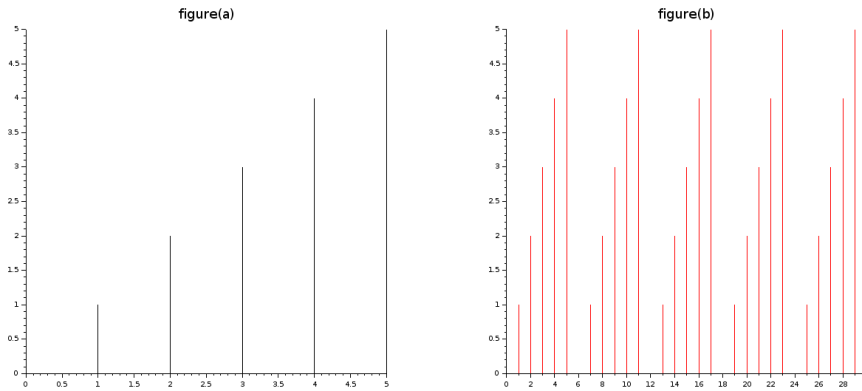


Figure 3.1: energy and power of discrete signals

```

)
17 power=(1/6).*sum(f.^2); //finding power of signal in
    fig(b)
18
19 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

```

1 clc
2 clear
3
4 n=[0:1:5];
5 // Defining unit step function
6 function y=u(x)
7     y=sign((sign(x)+1))
8 endfunction

```

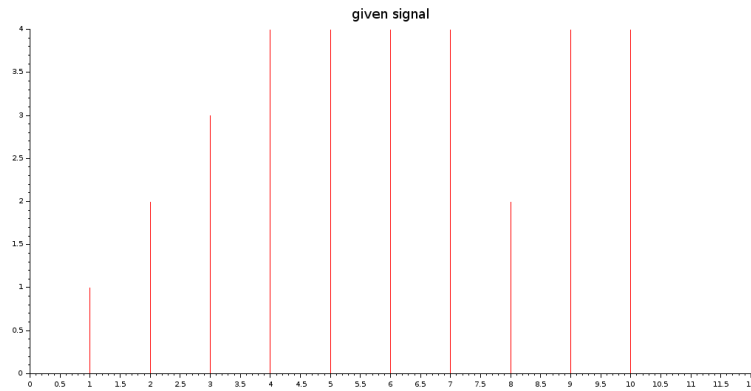


Figure 3.2: discrete signal representation

```

9
10 n=0:1:11;
11 f=n.*(u(n)-u(n-5))+4.*(u(n-5)-u(n-11))-2.*(n==8); //
    defining the given function
12 plot2d3(n,f,[5])
13 title("given signal","fontsize",4)

```

Scilab code Exa 3.8 iterative solution of discrete time systems

```

1 clc
2 clear
3 close
4
5 n=[-1:4]'; //defining the discrete time
6 y= [16; zeros(length(n)-1,1)];
7
8 //predefining the initial conditions + zero output
    array

```

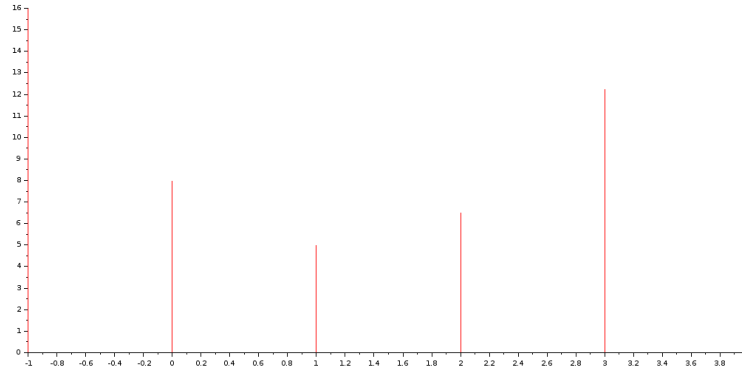


Figure 3.3: iterative solution of discrete time systems

```

9
10 x=[0;n(2:$).^2]; //defining input array with first
    entry corresponding to time instant -1
11
12 for k=1:length(n)-2
13 y(k+1) =0.5*y(k)+x(k+1);
14 end;
15
16 plot2d3(n,y,[5]);

```

Scilab code Exa 3.9 iterative solution of discrete time systems

```

1 clc
2 clear
3 close
4
5 n=[-2:10]'; //defining the discrete time
6 y= [1;2;zeros(length(n)-2,1)];

```

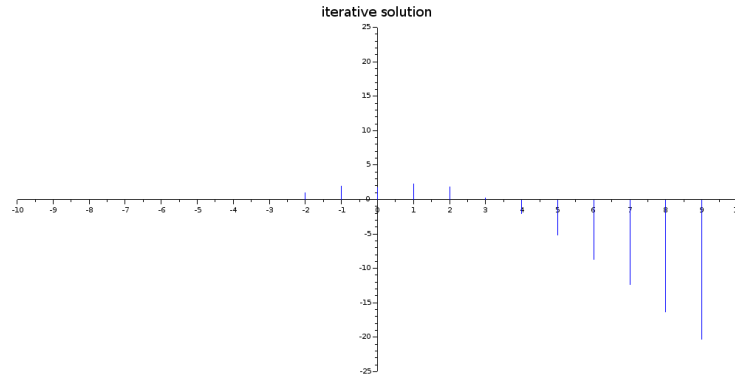


Figure 3.4: iterative solution of discrete time systems

```

7
8 //predefining the initial conditions + zero output
  array
9
10 x=[0;0;n(3:$)]; //defining input array with first
    two entries corresponding to time instants -2 and
    -1
11
12 for k=1:length(n)-2
13 y(k+2) = y(k+1) -0.24*y(k) + x(k+2) - 2*x(k+1);
14 end;
15
16 plot2d3(n,y,[2]);
17 set(gca(),"zoom_box",[-10 -25 10 25],"x_location",
    "middle","y_location","middle");
18 title("iterative solution","fontsize",4)

```

Scilab code Exa 3.10 Zero Input Response of the various systems

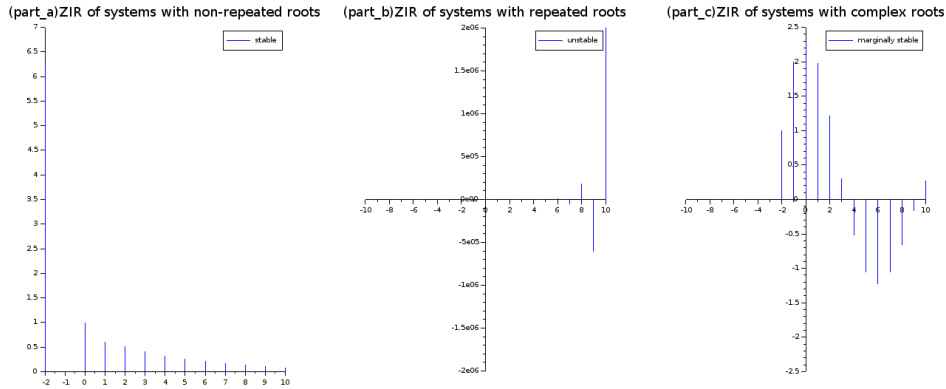


Figure 3.5: Zero Input Response of the various systems

```

1  clc
2  clear
3  close
4
5  n=[-2:10]'; //defining the discrete time
6
7  //*****part(a) (non-
   repeated roots)*****
8
9  y1= [25/4;0;zeros(length(n)-2,1)]; //predefining the
   initial conditions + zero output array
10 for k=1:length(n)-2
11 y1(k+2) = 0.6*y1(k+1) + 0.16*y1(k);
12 end;
13 subplot(1,3,1); plot2d3(n,y1,[2]); title("(part_a) ZIR
   of systems with non-repeated roots", "fontsize", 4)
   ;
14 legend("stable");
15
16 //*****part(b) (repeated
   roots)*****
17

```

```

18 y2= [-2/9;-1/3;zeros(length(n)-2,1)];//predefining
    the initial conditions + zero output array
19 for k=1:length(n)-2
20 y2(k+2) = -6*y2(k+1)-9*y2(k);
21 end;
22 subplot(1,3,2);plot2d3(n,y2,[2]);title("(part_b)ZIR
    of systems with repeated roots","fontsize",4);
23 set(gca(),"zoom_box",[-10 -max(y2) 10 max(y2)],"
    x_location","middle","y_location","middle");
24 legend("unstable");
25
26 //*****part(c)(complex
    roots)*****
27
28 y3= [1;2;zeros(length(n)-2,1)];//predefining the
    initial conditions + zero output array
29 for k=1:length(n)-2
30 y3(k+2) =1.56* y3(k+1) -0.81*y3(k);
31 end;
32 subplot(1,3,3);plot2d3(n,y3,[2]);title("(part_c)ZIR
    of systems with complex roots","fontsize",4);
33 set(gca(),"zoom_box",[-10 -2.5 10 2.5],"x_location",
    "middle","y_location","middle");
34 legend("marginally stable");

```

Scilab code Exa 3.11 impulse response

```

1 clc
2 clear
3 close;
4
5 n=[-2:20]';//defining the discrete time
6 y= [0;0;zeros(length(n)-2,1)];//predefining the

```

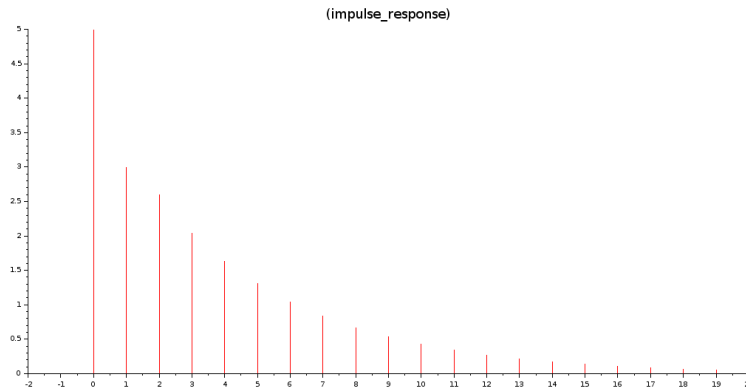


Figure 3.6: impulse response

```

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

```

Scilab code Exa 3.13 discrete convolution

```

1 clc
2 clear
3 close
4
5 function y=u(x)
6   y=sign(sign(x)+1);
7 endfunction
8

```

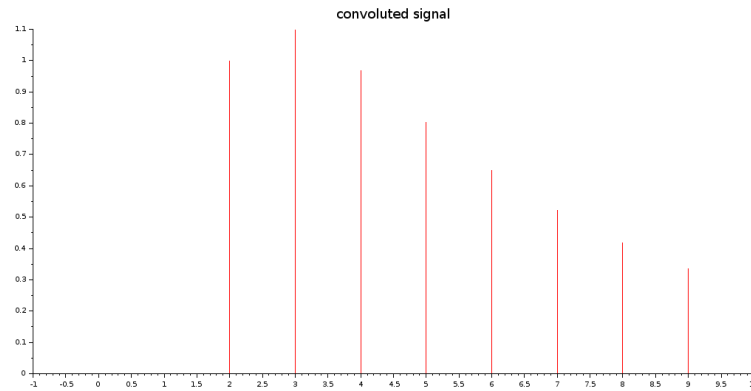


Figure 3.7: discrete convolution

```

9  n=-1:10;
10 x=((0.8).^n).*u(n);
11 g=((0.3).^n).*u(n);
12
13 c=convol2d(x,g); //discrete convolution
14
15 plot2d3(0:length(c)-1,c,[5]);
16 zoom_rect([-1 min(c) 10 max(c)]);
17 title("convoluted signal","fontsize",4)

```

Scilab code Exa 3.14 Zero state response of discrete systems

```

1  clc
2  clear
3  close
4
5  n=[0:20];
6  x=(4.^(-n));

```

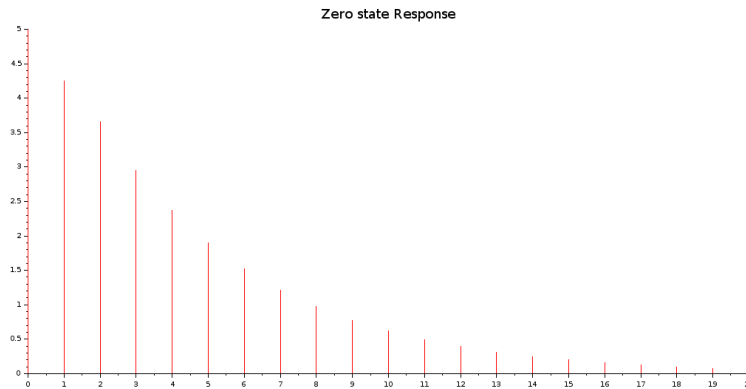


Figure 3.8: Zero state response of discrete systems

```

7 h = ((-0.2) .^ n + 4 * (0.8) .^ n);
8 c = convol(x, h); // discrete convolution
9
10 plot2d3(n, c(1:length(n)), [5]); title("Zero state
    Response", "fontsize", 4);

```

Scilab code Exa 3.15 discrete convolution

```

1 clc
2 clear
3 close
4
5 function y = u(x)
6     y = sign(sign(x) + 1);
7 endfunction
8
9 n = [0:6];
10 x = ((0.8) .^ n) .* u(n);

```

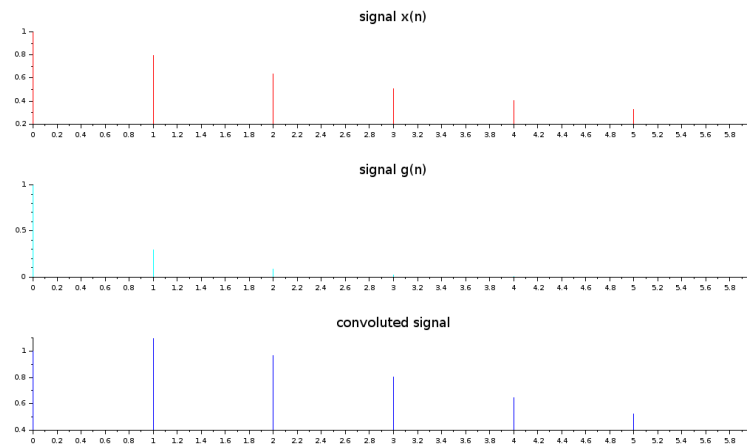


Figure 3.9: discrete convolution

```

11 g=((0.3).^n).*u(n);
12 c=convol2d(x,g); // discrete convolution
13
14 subplot(3,1,1); plot2d3(n,x,[5]); title("signal x(n)",
    "fontsize",4);
15 subplot(3,1,2); plot2d3(n,g,[4]); title("signal g(n)",
    "fontsize",4);
16 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

```

1 clc
2 clear
3 close
4
5 function y=u(x)
6     y=sign(sign(x)+1);

```

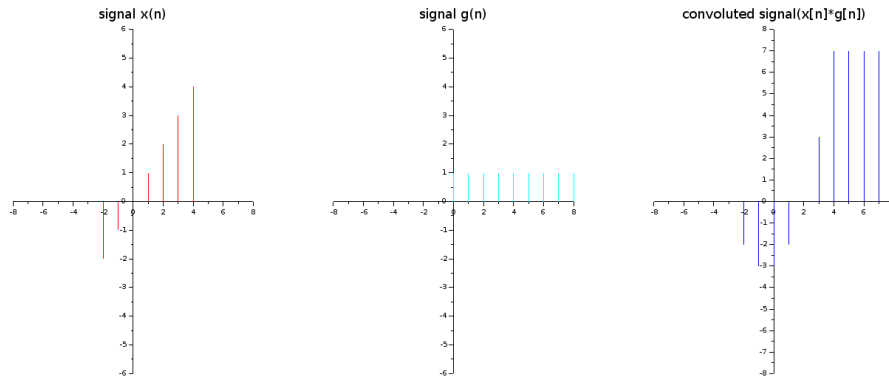


Figure 3.10: sliding tape discrete convolution

```

7  endfunction
8
9  n=[-2:15];
10 x=n.*(u(n+2)-u(n-5));
11 g=1.*u(n);
12 c=convol2d(x,g); //discrete convolution
13
14 subplot(1,3,1);plot2d3(n,x,[5]);title("signal x(n)",
    "fontsize",4); //plotting x[n]
15 set(gca(),"zoom_box",[-8 -6 8 6],"x_location",
    "middle","y_location","middle");
16 subplot(1,3,2);plot2d3(n,g,[4]);title("signal g(n)",
    "fontsize",4); //plotting g[n]
17 set(gca(),"zoom_box",[-8 -6 8 6],"x_location",
    "middle","y_location","middle");
18 subplot(1,3,3);plot2d3([-4:length(c)-5],c,[2]);title
    ("convoluted signal(x[n]*g[n])","fontsize",4); //
    plotting convoluted signal
19 set(gca(),"zoom_box",[-8 -8 8 8],"x_location",
    "middle","y_location","middle");

```

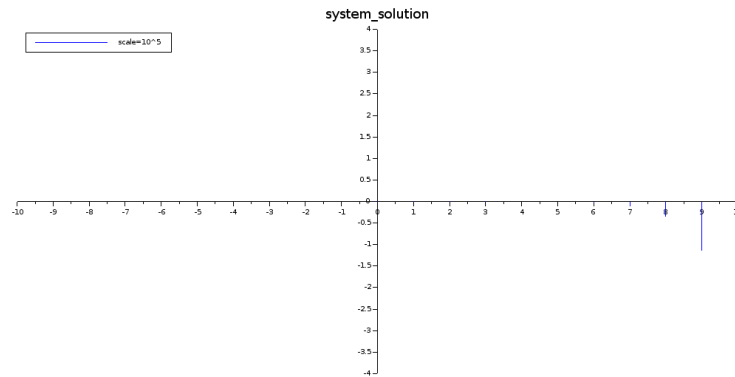


Figure 3.11: system response to auxiliary conditions

Scilab code Exa 3.17 system response to auxiliary conditions

```

1  clc
2  clear
3  close
4
5  n=[0:10]'; //defining the discrete time
6  y= [4;13;zeros(length(n)-2,1)]; //predefining the
   initial conditions + zero output array
7  x=(3*n+5);
8  for k=1:length(n)-2
9  y(k+2) = 5*y(k+1)-6*y(k)+x(k+1)-5*x(k);
10 end;
11
12 plot2d3(n,y/(10^5),[2]);title("system_solution",
   fontsize",4);
13 set(gca(),"zoom_box",[-10 -4 10 4],"x_location",

```

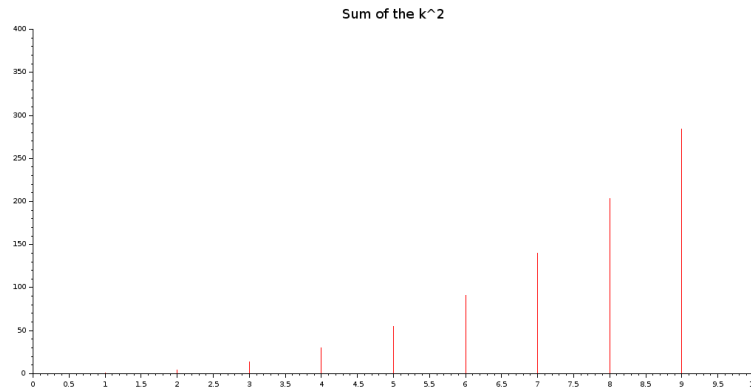



Figure 3.12: summation of given function

```

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

```

Scilab code Exa 3.18 summation of given function

```

1 clc
2 clear
3 close
4
5 n=[0:10]';
6 y= [0; zeros(length(n)-1,1)]; //predefining the
   initial conditions + zero output array
7 x=(n+1).^2;
8 for k=1:length(n)-1
9     y(k+1) =y(k)+x(k);
10 end;
11
12 plot2d3(n,y,[5]);title("Sum of the k^2", "fontsize"

```

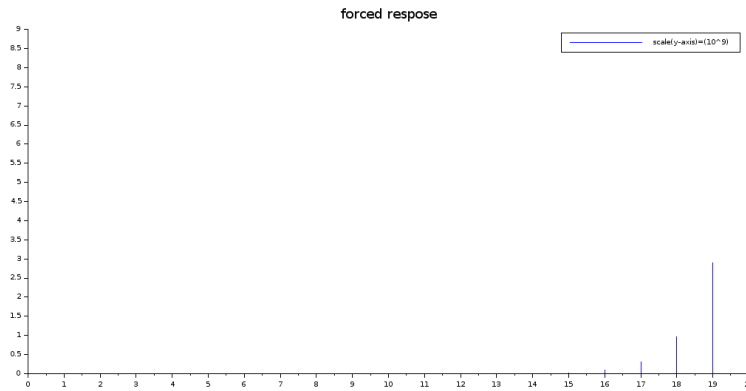


Figure 3.13: forced response

,4);

Scilab code Exa 3.19 forced response

```

1  clc ;
2  clear ;
3  close ;
4
5
6  n =0:20;
7  x=(3.^n);
8  a =[1 -3 2];
9  b =[0 1 2];
10 y= filter (b,a,x);
11 plot2d3 (n,y/(10^9) ,[2]);
12 title(" forced response", " fontsize",4);
13 legend(" scale (y-axis)=(10^9)")

```

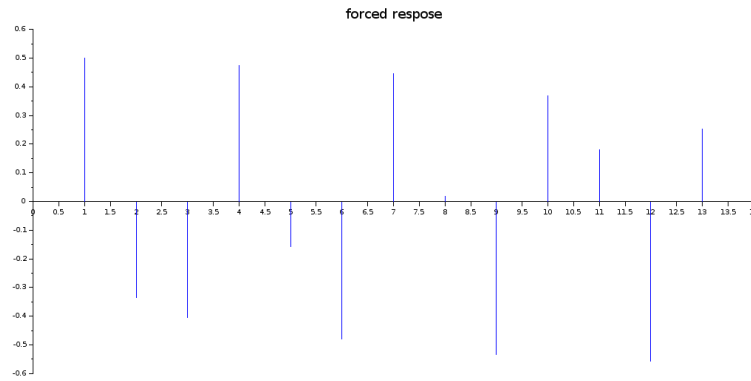


Figure 3.14: forced response

Scilab code Exa 3.20 forced response

```

1 clear ;
2 close ;
3 clc ;
4
5
6 n =0:14;
7 x= cos(2*n+%pi/3); //forcing function
8 a =[1 -1 0.16];
9 b =[0 1 0.32];
10 y= filter (b,a,x);
11 plot2d3 (n,y,[2]);
12 title(" forced response"," fontsize",4);
13 set(gca()," x_location"," middle");

```

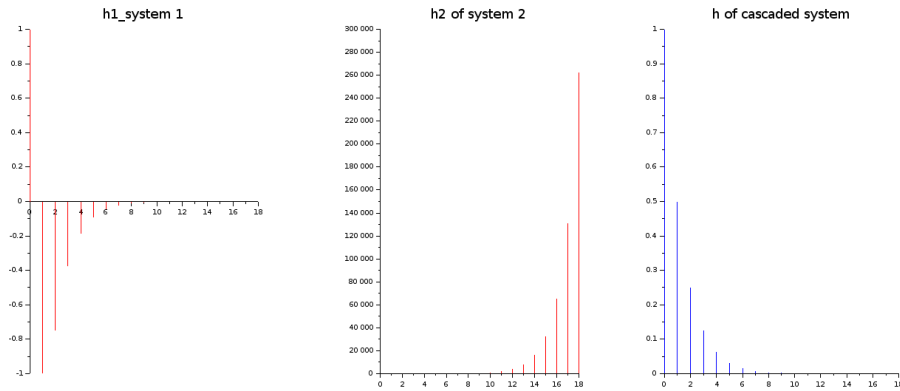


Figure 3.15: response of cascaded system

Scilab code Exa 3.21 response of cascaded system

```

1  clc
2  clear
3  close;
4
5  n=0:18;
6  h1=4*(n==0) -3*(0.5).^n;
7  h2=(2^n);
8  h_cas=conv(h1,h2); //response of cascaded system
9  subplot(1,3,1); plot2d3(n,h1,[5]); title("h1_system 1"
    ," fontsize",4);
10 set(gca(),"zoom_box",[0 -max(h1) 18 max(h1)],"
    x_location","middle");
11 subplot(1,3,2); plot2d3(n,h2,[5]); title("h2 of system
    2"," fontsize",4);
12 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

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

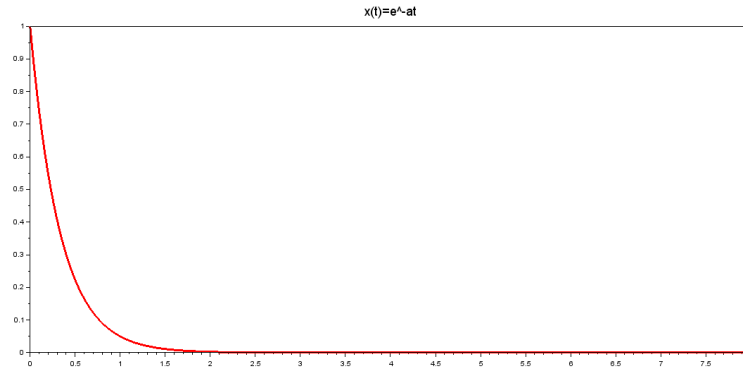


Figure 4.1: Laplace transform of exponential signal

```

transform
14 disp(y1)

```

check Appendix [AP 2](#) for dependency:

laplace.sci

Scilab code Exa 4.2 Laplace transform of given signals

```

1 //scilab symbolic toolbox(scilab-scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu-14.04 and Scilab-5.0.0 as it's not
  supported in higher versions
3
4 //use laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  laplace.sci', -1)
6
7 //(a) laplace transform of delta(t)

```

```

8 Syms t s;
9 y =laplace('0',t,s)
10 disp(y)
11
12 //(b) Laplace Transform of u(t)
13 y1 =laplace('1',t,s);
14 disp(y1)
15
16 //(c) laplace transform of cos(w0*t)u(t)
17 y2 =laplace('cos(w0*t)',t,s);
18 disp(y2)

```

check Appendix AP 1 for dependency:

inv_laplace.sci

Scilab code Exa 4.3 Inverse Laplace transform of given transfer functions

```

1 //scilab symbolic toolbox(scilab–scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu–14.04 and Scilab–5.0.0 as it's not
  supported in higher versions
3
4
5 clc
6 //use inv_laplace.sci before this code using correct
  path]
7 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
8 //*****part(a)
  *****
9 Syms t s;
10 [A]=pfs((7*s-6)/((s^2-s-6))); //partial fraction of
  F(s)
11 F1 = ilaplace(A(1),s,t)

```



```

12 F2 = ilaplace(A(2),s,t)
13 Fa = F1+F2;
14 disp(Fa,"fa(t)=")
15
16 //(*****part(b)
    *****
17 [A]=pfs((2*s^2+5)/((s^2-3*s+2))); //partial
    fraction of F(s)
18 F1 = ilaplace(A(1),s,t)
19 F2 = ilaplace(A(2),s,t)
20 Fb = F1+F2;
21 disp(Fb,"fb(t)=")
22
23 //*****part(c)
    *****
24 [A]=pfs((6*(s+34))/(s*(s^2+10*s+34))); //partial
    fraction of F(s)
25 F1 = ilaplace(A(1),s,t)
26 F2 = ilaplace(A(2),s,t)
27 Fc = F1+F2;
28 disp(Fc,"fc(t)=")

```

check Appendix [AP 2](#) for dependency:

laplace.sci

Scilab code Exa 4.4 Laplace transform of given signals

```

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

```

```

5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  laplace.sci', -1)
6 Syms t s;
7 a = 3;
8 T = 1;
9 y1 = laplace('t',t,s);
10 y2 = laplace('t',t,s);
11 y3 = laplace('1',t,s);
12 y=y1*(%e^(-s))+y2*(%e^(-2*s))+y3*(%e^(-4*s))
13 disp(y)

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.5 Inverse Laplace transform of given transfer function

```

1 //scilab symbolic toolbox(scilab-scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu-14.04 and Scilab-5.0.0 as it's not
  supported in higher versions
3 //use inv_laplace.sci before this code using correct
  path]
4 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
5 clc
6
7 s1 =%s ;
8 Syms t s;
9 [A]=pfss((s1+3)/((s1+1)*(s1+2))); //partial fraction
  of F(s)
10 F1 = ilaplace(A(1),s,t)
11 F2 = ilaplace(A(2),s,t)
12 Fa = F1+F2;
13 disp(Fa,"f1(t)=")

```

```

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

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.8 Time convolution using laplace transform

```

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

```

Scilab code Exa 4.9 Initial and final values of laplace transform

```

1 //scilab symbolic toolbox(scilab–scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu–14.04 and Scilab –5.0.0 as it’s not
  supported in higher versions
3 //limit can be used after installing symbolic
  toolbox only
4 clc
5
6 s=%s;
7 num =poly([30 20], 's', 'coeff')
8 den =poly([0 5 2 1], 's', 'coeff')
9 X = num/den;
10 disp (X,"X(s)=")
11 SX = s*X;
12 Initial_Value =limit(SX,s,%inf);
13 final_value =limit(SX,s,0);
14 disp(Initial_Value,"x(0)=")
15 disp(final_value,"x(inf)=")

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.10 solving system equation

```

1 //scilab symbolic toolbox(scilab–scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu–14.04 and Scilab –5.0.0 as it’s not
  supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)

```

```

6 clc
7 Syms t s;
8 [A] = pfss((2*s^2+20*s+45)/((s+2)*(s+3)*(s+4)));
9 F1 = ilaplace(A(1),s,t)
10 F2 = ilaplace(A(2),s,t)
11 F3 = ilaplace(A(3),s,t)
12 F = F1+F2+F3
13 disp(F,"y(t)=")

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.11 Finding inductor current

```

1 //scilab symbolic toolbox(scilab–scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu–14.04 and Scilab–5.0.0 as it’s not
  supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
6 clc
7 Syms t s;
8 [A] = pfss((2*s)/(s^2+2*s+5));
9 F1 = ilaplace(A(1),s,t)
10 F2 = ilaplace(A(2),s,t)
11 F3 = ilaplace(A(3),s,t)
12 F = F1+F2+F3
13 disp(F,"inductor_current=")

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.12 Response of the given system

```
1 //scilab symbolic toolbox(scilab-scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu-14.04 and Scilab-5.0.0 as it's not
  supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
6 clc
7 Syms t s;
8 [A] = pfss((3*s+3)/((s+5)*(s^2+5*s+6)));
9 F1 = ilaplace(A(1),s,t)
10 F2 = ilaplace(A(2),s,t)
11 F3 = ilaplace(A(3),s,t)
12 F = F1+F2+F3
13 disp(F,"system_solution=")
```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.15 Finding loop current

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

```

3
4 //use inv_laplace.sci before this code using correct
    path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
    inv_laplace.sci', -1)
6 clc
7
8 Syms s t;
9 [A] = pfss((10)/(s^2+3*s+2));
10 F1 = ilaplace(A(1),s,t)
11 F2 = ilaplace(A(2),s,t)
12 F3 = ilaplace(A(3),s,t)
13 F = F1+F2+F3
14 disp(F," Loop_current=");

```

check Appendix [AP 2](#) for dependency:

laplace.sci

Scilab code Exa 4.16 Finding loop currents

```

1 //scilab symbolic toolbox(scilab–scimax) is needed
    for this code
2 //symbolic toolbox should be strictly installed in
    Ubuntu–14.04 and Scilab –5.0.0 as it's not
    supported in higher versions
3
4 //use laplace.sci before this code using correct
    path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
    laplace.sci', -1)
6
7 clc
8 Syms t s;
9 y1 =laplace('24*%e^(-3*t)+48*%e^(-4*t)',t,s);
10 disp(y1,"y1=")

```

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

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.17 Finding loop current

```
1 //scilab symbolic toolbox(scilab-scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu-14.04 and Scilab-5.0.0 as it's not
  supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
6 clc
7 Syms t s;
8 [A] = pfs((2*s^2+9*s+4)/((s)*(s^2+3*s+1)));
9 F1 = ilaplace(A(1),s,t)
10 F2 = ilaplace(A(2),s,t)
11 F3 = ilaplace(A(3),s,t)
12 F = F1+F2+F3
13 disp(F,"output")
```

Scilab code Exa 4.23 frequency response of the given system

```
1
```

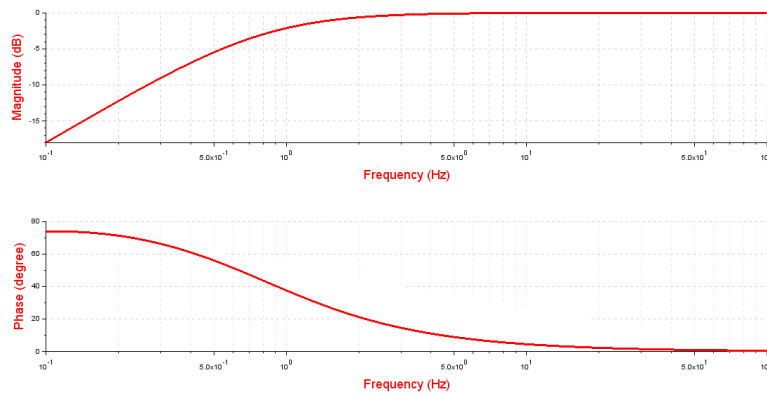



Figure 4.2: frequency response of the given system

```

2  clc
3  clear
4  close;
5
6  s=poly(0, 's '); //polynomial definition
7  h=syslin('c', (s+0.1)/(s+5)); //transfer function
8  bode(h,0.1,100); //plotting frequency response

```

Scilab code Exa 4.24 frequency response of the given systems

```

1  clc
2  clear
3  close;
4
5  s=poly(0, 's ') //polynomial definition
6
7  h=syslin('c', (s^2/s))
8  figure(1)
9  bode(h,0.1,100);
10
11 h1=syslin('c', (1/s)) //polynomial definition

```

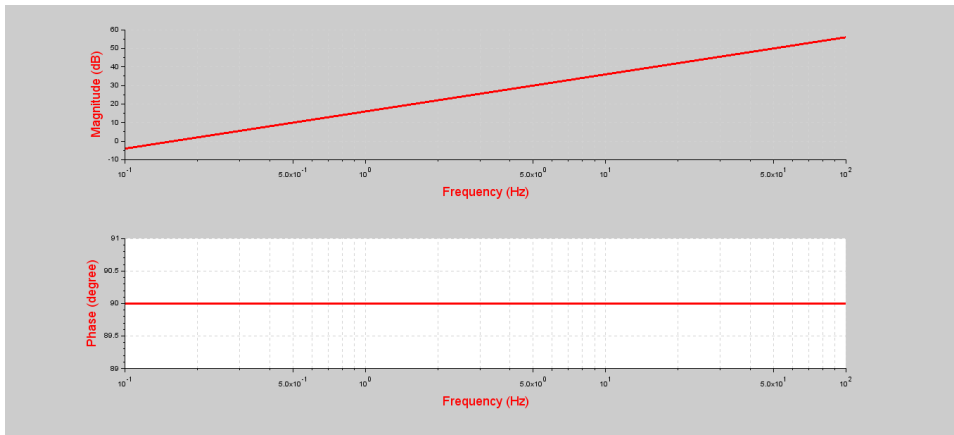


Figure 4.3: frequency response of the given systems

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

Scilab code Exa 4.25 bode plots of the given transfer function

```
1
2 clc
3 clear
4 close;
5
6 s=%s;
7 h=syslin('c',((20*s^2+2000*s)/(s^2+12*s+20)))//
   transfer function
8 bode(h,0.1,100);
```

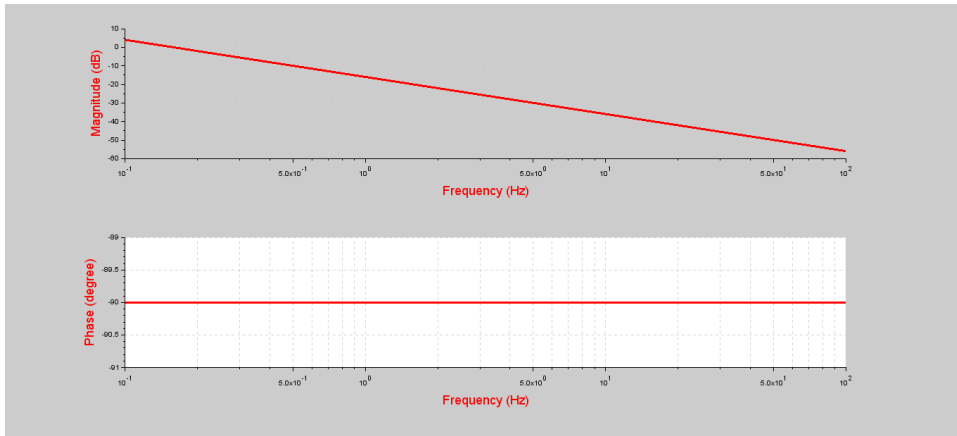


Figure 4.4: frequency response of the given systems

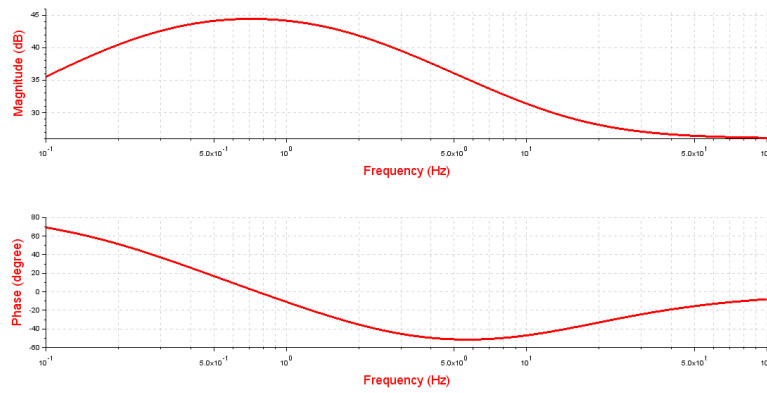


Figure 4.5: bode plots of the given transfer function

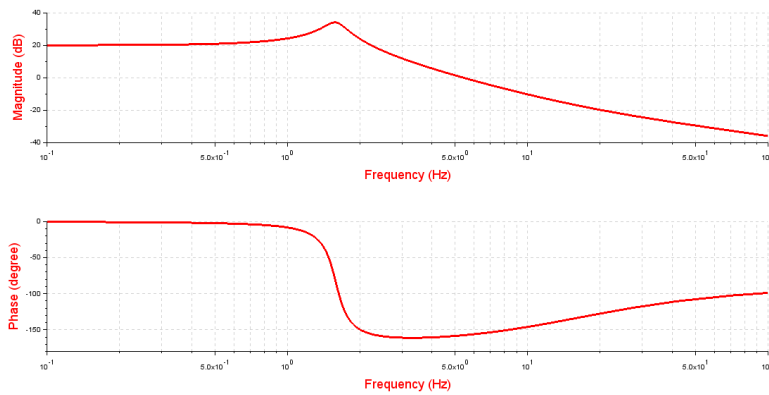


Figure 4.6: bode plots of the given transfer function

Scilab code Exa 4.26 bode plots of the given transfer function

```

1
2 clc
3 clear
4 close;
5
6 s=%s;
7 h=syslin('c',((10*s+1000)/(s^2+2*s+100)))/transfer
  function
8 bode(h,0.1,100);

```

Scilab code Exa 4.27 second order notch filter analysis

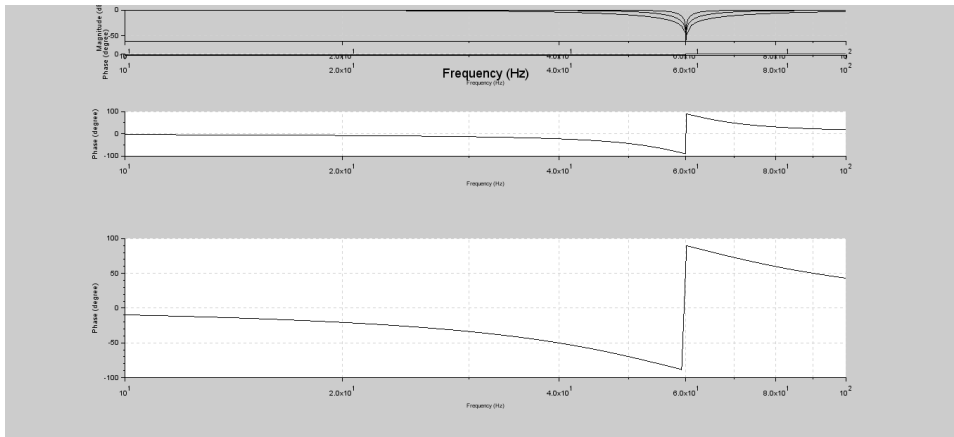


Figure 4.7: second order notch filter analysis

```

1  clc
2  clear
3  close
4
5  omega_0=2*%pi*60; theta = [60 80 87]*(%pi/180);
6  omega = (0:0.5:1000)';
7  mag = zeros(3,length(omega));
8  s=poly(0,'s')
9  figure(1)
10 for m =1:length(theta)
11     H=syslin('c',((s^2+omega_0^2)/(s^2+2*omega_0*cos(
12         theta(m))*s +omega_0^2)));
13     bode(H,10,100);
13 end

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.28 Inverse Laplace transform of given transfer function

```

1
2 //scilab symbolic toolbox(scilab–scimax) is needed
   for this code
3 //symbolic toolbox should be strictly installed in
   Ubuntu–14.04 and Scilab –5.0.0 as it 's not
   supported in higher versions
4
5 //use inv_laplace.sci before this code using correct
   path]
6 //exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
7
8 clc
9
10 s=%s;
11 t=%t;
12 //*****part(a)*****
13 [A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
14 F1 = ilaplace(A(1),s,t)
15 F2 = ilaplace(A(2),s,t)
16 F=F1+F2;
17 disp(F,"f(t)=")
18
19 //*****part(b)*****
20 [A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
21 F1 = ilaplace(A(1),s,t)
22 F2 = ilaplace(A(2),s,t)
23 F = -F1-F2;
24 disp(F,"f(t)=")
25
26 //*****part(c)*****
27 [A]=pfss(1/((s-1)*(s+2))) //partial fraction of F(s)
28 F1 = ilaplace(A(1),s,t)
29 F2 = ilaplace(A(2),s,t)
30 F = -F1+F2;
31 disp(F,"f(t)=")

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.29 Finding loop current in RC circuit

```
1 //scilab symbolic toolbox(scilab-scimax) is needed
  for this code
2 //symbolic toolbox should be strictly installed in
  Ubuntu-14.04 and Scilab-5.0.0 as it's not
  supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
  path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
  inv_laplace.sci', -1)
6 clc
7
8 Syms s t;
9 [A] = pfss((-s)/((s-1)*(s-2)*(s+1))); //partial
  fraction of transfer function
10 F1 = ilaplace(A(1),s,t)
11 F2 = ilaplace(A(2),s,t)
12 F3 = ilaplace(A(3),s,t)
13 F = F1+F2+F3
14 disp(F,"F=");
```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.30 System response from transfer function

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

```

2 //symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
   path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
6 clc
7
8 Syms s t;
9 [A] = pfss((-1)/((s-1)*(s+2))); //partial fraction of
   transfer function
10 F1 = ilaplace(A(1),s,t)
11 F2 = ilaplace(A(2),s,t)
12 F = F1+F2
13 disp(F)

```

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 4.31 System response from transfer function

```

1 //scilab symbolic toolbox(scilab-scimax) is needed
   for this code
2 //symbolic toolbox should be strictly installed in
   Ubuntu-14.04 and Scilab-5.0.0 as it's not
   supported in higher versions
3
4 //use inv_laplace.sci before this code using correct
   path]
5 //exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci', -1)
6 clc
7 Syms t s;

```



```
8 //for Re s>-1
9 [A] = pfs(1/((s+1)*(s+5))); //partial fraction of
    transfer function
10 F1 = ilaplace(A(1),s,t)
11 F2 = ilaplace(A(2),s,t)
12 F = F1+F2
13 disp(F," for Re(s)>-1")
14
15 //for -5< Re s <-2
16 [B] = pfs(-1/((s+2)*(s+5))); //partial fraction of
    transfer function
17 G1 = ilaplace(B(1),s,t)
18 G2 = ilaplace(B(2),s,t)
19 G = G1+G2
20 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

```
1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3
4 clc
5 Syms n z;
6 a = 0.5;
7 x =(a)^n;
8 n1=0:10;
9 plot2d3(n1,a^n1); xtitle('a^n','n');
10 plot(n1,a^n1,'r. ')
11 X = symsum(x*(z^(-n)),n,0,%inf)
12 disp(X,"ans=")
```

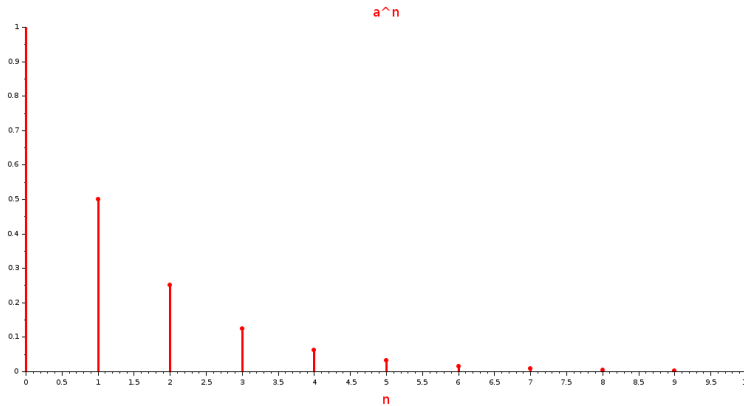


Figure 5.1: Z transform of the given signal

Scilab code Exa 5.2 Z transform of the given signals

```

1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3
4 clc
5 Syms n z;
6 Wo =%pi/4;
7 a = (0.33)^n;
8 x1=%e^(sqrt(-1)*Wo*n);
9 X1=symsum(a*x1*(z^(-n)),n,0,%inf)
10 x2=%e^(-sqrt(-1)*Wo*n)
11 X2=symsum(a*x2*(z^(-n)),n,0,%inf)
12 X = (1/(2*sqrt(-1)))*(X1+X2)
13 disp(X,"ans1=")
14

```

```

15 //*****part (a)*****8
16 X=symsum(1*(z^(-n)),n,0,0);
17 disp(X,"ans2=")
18
19 //*****part (b)*****8
20 X=symsum(1*(z^(-n)),n,0,4);
21 disp(X,"ans3=")
22
23 //*****part (c)*****8
24 X=symsum(1*(z^(-n)),n,0,%inf);
25 disp(X,"ans4=")

```

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

```

1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3 //limit function can be accessed after installing
  symbolic toolbox
4 clc
5
6 //*****part (a)
  *****
7 z = %z;
8 Syms n z1;//To find out Inverse z transform z must
  be linear z = z1
9 X = (8*z-19)/((z-2)*(z-3))
10 X1 = X.den; //don't use denom() or numer()
  , they'll be removed in upcoming versions
11 zp = roots(X1);
12 X1 = (8*z1-19)/((z1-2)*(z1-3))
13 F1 = X1*(z1^(n-1))*(z1-zp(1));
14 F2 = X1*(z1^(n-1))*(z1-zp(2));

```

```

15 h1 = limit(F1,z1,zp(1));
16 disp(h1,'h1[n]=')
17 h2 = limit(F2,z1,zp(2));
18 disp(h2,'h2[n]=')
19 h = h1+h2;
20 disp(h,'h[n]=')
21
22
23
24 // *****part (c)
      *****
25 X  =(2*z*(3*z+17))/((z-1)*(z^2-6*z+25))
26 X1 = X.den;
27 zp = roots(X1);
28 X1 = 2*z1*(3*z1+17)/((z1-1)*(z1^2-6*z1+25))
29 F1 = X1*(z1^(n-1))*(z1-zp(1));
30 F2 = X1*(z1^(n-1))*(z1-zp(2));
31 h1 = limit(F1,z1,zp(1));
32 disp(h1,'h1[n]=')
33 h2 = limit(F2,z1,zp(2));
34 disp(h2,'h2[n]=')
35 h = h1+h2;
36 disp(h,'h[n]=')

```

Scilab code Exa 5.5 Solving given system equation using z transform

```

1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab-5.5.0 as it may not work in
  higher versions
3 //limit function can be accessed after installing
  symbolic toolbox
4
5 clc

```

```

6 Syms n z;
7 H1 = (26/15)/(z-(1/2));
8 H2 = (7/3)/(z-2);
9 H3 = (18/5)/(z-3);
10 F1 = H1*z^(n)*(z-(1/2));
11 F2 = H2*z^(n)*(z-2);
12 F3 = H3*z^(n)*(z-3);
13
14 h1 = limit(F1,z,1/2);
15 disp(h1,'h1[n]=')
16 h2 = limit(F2,z,2);
17 disp(h2,'h2[n]=')
18 h3 = limit(F3,z,3);
19 disp(h3,'h3[n]=')
20 h = h1-h2+h3;
21 disp(h,'h[n]=')

```

Scilab code Exa 5.6 solving system difference equation

```

1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3 //limit function can be accessed after installing
  symbolic toolbox
4
5 clc
6 Syms n z;
7 H1 = (2/3)/(z+0.2);
8 H2 = (8/3)/(z+0.8);
9 H3 = (2)/(z+0.5);
10 F1 = H1*z^(n)*(z+0.2);
11 F2 = H2*z^(n)*(z+0.8);
12 F3 = H3*z^(n)*(z+0.5);

```

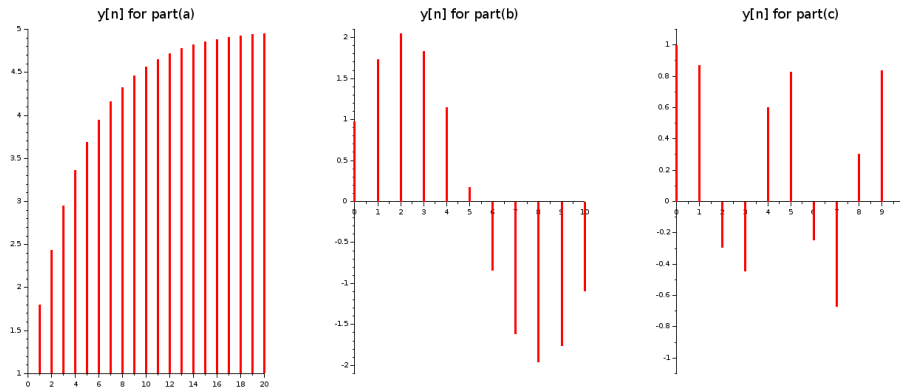


Figure 5.2: System response to different inputs

```

13
14 h1 = limit(F1,z,-0.2);
15 disp(h1,'h1[n]=')
16 h2 = limit(F2,z,-0.8);
17 disp(h2,'h2[n]=')
18 h3 = limit(F3,z,-0.5);
19 disp(h3,'h3[n]=')
20 h = h1-h2+h3;
21 disp(h,'h[n]=')

```

Scilab code Exa 5.10 System response to different inputs

```

1 clc ;
2 clear ;
3 close ;
4
5
6 n =0:20;

```

```

7 T=0.001;
8 x1=1.^n;
9 x2=cos(%pi/6*n-0.2);
10 x3=cos(1500*n*T); //sampled input
11 a =[1 -0.8];
12 b =[1 0];
13 y1= filter (b,a,x1);
14 y2= filter (b,a,x2);
15 y3= filter (b,a,x3);
16
17 subplot(1,3,1);plot2d3(n,y1,[5]);
18 title("y[n] for part(a)","fontsize",4);
19
20 subplot(1,3,2);plot2d3(n,y2,[5]);
21 title("y[n] for part(b)","fontsize",4);
22 set(gca(),"x_location","middle","zoom_box",[0 -2.1
    10 2.1]);
23
24 subplot(1,3,3);plot2d3(n,y3,[5]);
25 title("y[n] for part(c)","fontsize",4);
26 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

```

1 clc
2 clear
3
4 f=50*10^3;
5 T=0.5/f;
6 disp(1/(T*10^3),"the maximum sampling frequency in
    kHz=") //in seconds

```

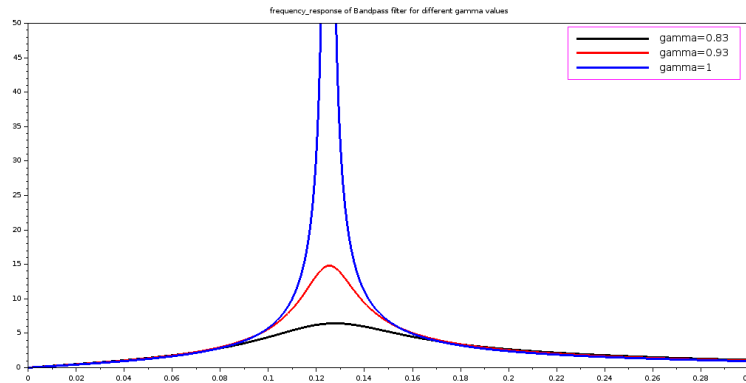


Figure 5.3: frequency response of Bandpass filter

Scilab code Exa 5.13 Calculating maximum sampling interval or minimum frequency

```

1  clc
2  clear
3
4  T=25*10-6;
5  disp(1/(T*103), "the maximum sampling frequency in
      kHz=") //in seconds

```

Scilab code Exa 5.14 frequency response of Bandpass filter

```

1  clc
2  clear
3  close
4
5  gm=[0.83 0.93 1.00];
6  [xm1,fr1]=frmag([1 0 -1],[1 -sqrt(2)*gm(1) gm(1)
      ^2],4097);

```

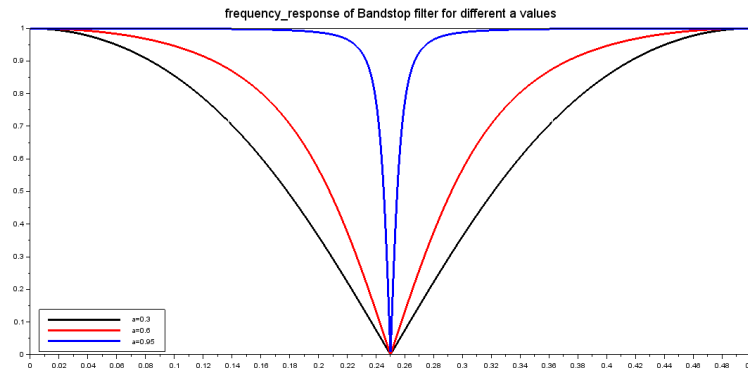


Figure 5.4: frequency response of Bandstop filter

```

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

```

Scilab code Exa 5.15 frequency response of Bandstop filter

```

1 clc
2 clear
3 close
4
5 a=[0.3 0.6 0.95];

```

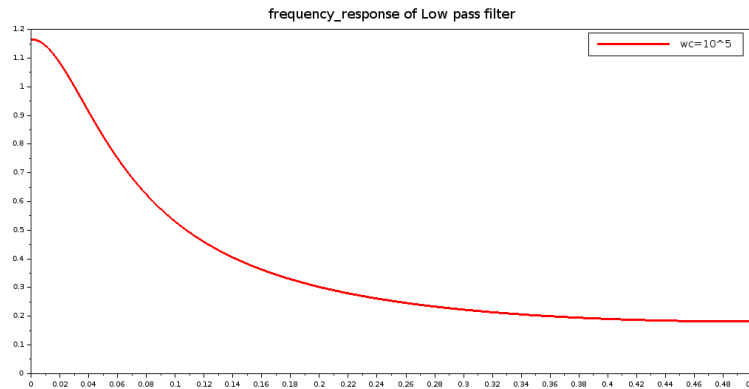


Figure 5.5: frequency response of low pass filter

```

6  [xm1,fr1]=frmag([1+a(1)^2 0 1+a(1)^2], [2 0 2*a(1)
      ^2],4097);
7
8  [xm2,fr2]=frmag([1+a(2)^2 0 1+a(2)^2], [2 0 2*a(2)
      ^2],4097);
9
10 [xm3,fr3]=frmag([1+a(3)^2 0 1+a(3)^2], [2 0 2*a(3)
      ^2],4097);
11
12 plot(fr1,xm1,'k');
13 plot(fr2,xm2,'r');
14 plot(fr3,xm3,'b');
15 zoom_rect([0 0 0.5 1])
16 title("frequency_response of Bandstop filter for
      different a values","fontsize",4)
17 legend("a=0.3","a=0.6","a=0.95",[3]);

```

Scilab code Exa 5.16 frequency response of low pass filter

```

1  clc
2  clear
3  close
4
5  wc=10^5;
6  [xm1,fr1]=fzmag([0.3142 0], [1 -0.7304],4097);//
   finding fr response from transfer functions
7
8  plot(fr1,xm1,'r');
9  zoom_rect([0 0 0.5 1.2])
10 title("frequency_response of Low pass filter",
   fontsize",4)
11 legend("wc=10^5");

```

Scilab code Exa 5.17 Z transform of the given signal

```

1  //This code needs symbolic tool(scilab-scimax) to be
   instaslled
2  //The tool should be installed properly in Ubuntu
   -14.04 and scilab -5.5.0 as it may not work in
   higher versions
3
4  clc
5  Syms n z;
6  a=0.9
7  b = 1.2;
8
9  x1=(a)^(n)
10 x2=(b)^(-n)
11 X1=symsum(x1*(z^(-n)),n,0,%inf)
12 X2=symsum(x2*(z^(n)),n,1,%inf)
13 X = X1+X2;
14 disp(X,"ans=")

```

Scilab code Exa 5.18 Inverse Z transform of the given function

```
1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3 //limit function can be accessed after installing
  symbolic toolbox
4
5 clc
6 z = %z;
7 Syms n z1;
8
9
10 // *****part (a)
    *****
11 X = -z*(z+0.4)/((z-0.8)*(z-2))
12 X1 = X.den;
13 zp = roots(X1);
14 X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2))
15 F1 = X1*(z1^(n-1))*(z1-zp(1));
16 F2 = X1*(z1^(n-1))*(z1-zp(2));
17 h1 = limit(F1,z1,zp(1));
18 disp(h1, 'h1[n]= ')
19 h2 = limit(F2,z1,zp(2));
20 disp(h2, 'h2[n]= ')
21 h = h1+h2;
22 disp(h, 'h[n]= ')
23
24 // *****part (b)
    *****
25 X = -z*(z+0.4)/((z-0.8)*(z-2))
26 X1 = X.den;
```

```

27 zp = roots(X1);
28 X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2));
29 F1 = X1*(z1^(n-1))*(z1-zp(1));
30 F2 = X1*(z1^(n-1))*(z1-zp(2));
31 h1 = limit(F1,z1,zp(1));
32 disp(h1*'u(n)', 'h1[n]=')
33 h2 = limit(F2,z1,zp(2));
34 disp((h2)*'u(-n-1)', 'h2[n]=')
35 disp((h1)*'u(n)'-(h2)*'u(n-1)', 'h[n]=')
36
37 // *****part(c)
   *****
38 X = -z*(z+0.4)/((z-0.8)*(z-2))
39 X1 = X.den;
40 zp = roots(X1);
41 X1 = -z1*(z1+0.4)/((z1-0.8)*(z1-2));
42 F1 = X1*(z1^(n-1))*(z1-zp(1));
43 F2 = X1*(z1^(n-1))*(z1-zp(2));
44 h1 = limit(F1,z1,zp(1));
45 disp(h1*'u(-n-1)', 'h1[n]=')
46 h2 = limit(F2,z1,zp(2));
47 disp((h2)*'u(-n-1)', 'h2[n]=')
48 disp(-(h1)*'u(-n-1)'-(h2)*'u(-n-1)', 'h[n]=')

```

Scilab code Exa 5.19 ZSR of given system

```

1 //This code needs symbolic tool(scilab-scimax) to be
   instaslled
2 //The tool should be installed properly in Ubuntu
   -14.04 and scilab -5.5.0 as it may not work in
   higher versions
3 //limit function can be accessed after installing
   symbolic toolbox
4
5 clc

```

```

6 syms n z;
7 H1 = -z/(z-0.5);
8 H2 = (8/3)*z/(z-0.8);
9 H3=(-8/3)*z/(z-2);
10 F1 = H1*z^(n-1)*(z-0.5);
11 F2 = H2*z^(n-1)*(z-0.8);
12 F3 = H3*z^(n-1)*(z-2);
13 h1 = limit(F1,z,0.5);
14
15 disp(h1, 'h1[n]= ');
16 h2 = limit(F2,z,0.8);
17 disp(h2, 'h2[n]= ');
18 h3 = limit(F3,z,2);
19 disp(h3, 'h3[n]= ');
20 h = h1+h2+h3;
21 disp(h, 'h[n]= ')

```

Scilab code Exa 5.20 ZSR of given system

```

1 //This code needs symbolic tool(scilab-scimax) to be
  instaslled
2 //The tool should be installed properly in Ubuntu
  -14.04 and scilab -5.5.0 as it may not work in
  higher versions
3 //limit function can be accessed after installing
  symbolic toolbox
4
5 clc
6 Syms n z;
7 H1 = (-5/3)*z/(z-0.5);
8 H2 = (8/3)*z/(z-0.8);
9 H3=5*z/(z-0.5);
10 H4=-6*z/(z-0.6);
11 F1 = H1*z^(n-1)*(z-0.5);
12 F2 = H2*z^(n-1)*(z-0.8);

```

```
13 F3 = H3*z^(n-1)*(z-0.5);
14 F4 = H4*z^(n-1)*(z-0.6);
15 h1 = limit(F1,z,0.5);
16
17 disp(h1,'h1[n]=')
18 h2 = limit(F2,z,0.8);
19 disp(h2,'h2[n]=')
20 h3 = limit(F3,z,0.5);
21 disp(h3,'h3[n]=')
22 h4 = limit(F4,z,0.6);
23 disp(h4,'h4[n]=')
24 h = h1+h2+h3+h4;
25 disp(h,'h[n]=')
```

Chapter 6

Continuous time signal analysis The Fourier series

Scilab code Exa 6.1 Compact trigonometric Fourier series

```
1  clc
2  clear
3  close;
4
5  //finding fourier series
6  w0=2;    T0=2*%pi/w0; //period and frequency
7  dt=0.01;
8  t=0:dt:%pi-dt;
9  y=(exp(-t/2)).*((t>=0)&(t<%pi));
10 N=length(y);
11 c0=sum(y.*dt)/T0;
12 n1=[1:10]';
13 for n=1:10
14     aa(n)=2*sum(y.*cos(n*w0*t).*dt)/T0;
15     bb(n)=2*sum(y.*sin(n*w0*t).*dt)/T0;
16     cn(n)=sqrt(aa(n).^2+bb(n).^2);
17     thetan(n)=atan(-bb(n)/aa(n));
```

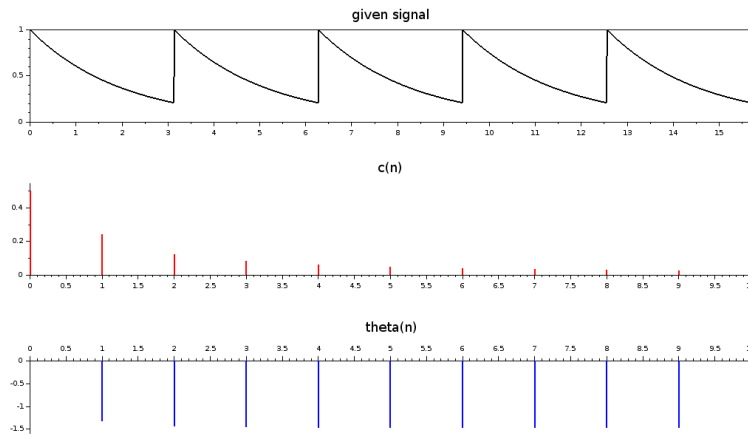


Figure 6.1: Compact trigonometric Fourier series

```

18 end
19
20 cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
21 y= repmat(y,1,5);t=0:dt:(length(y)-1).*dt; //periodic
    signal definition
22
23 subplot(3,1,1);plot(t,y,'r');title("given signal",
    fontsize",4);
24 set(gca(),"zoom_box",[t(1) 0 t($) 1]);
25 subplot(3,1,2);plot2d3(n,cn,[2]);title("c(n)",
    fontsize",4)
26 subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)
    ","fontsize",4)
27 set(gca(),"x_location","top");

```

Scilab code Exa 6.2 Compact trigonometric Fourier series

```

1 clc

```

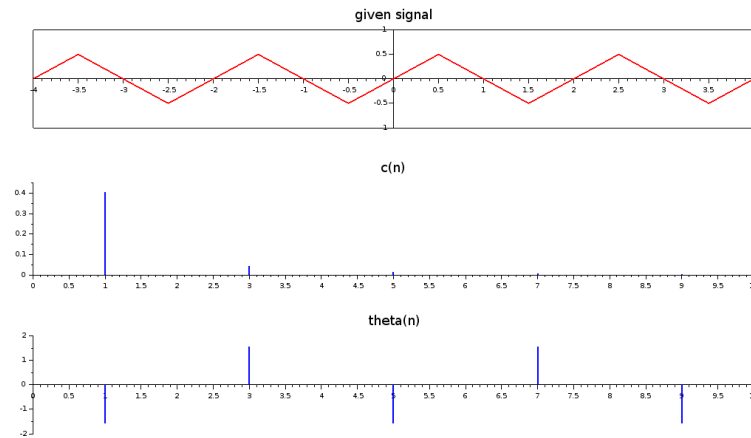


Figure 6.2: Compact trigonometric Fourier series

```

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

```

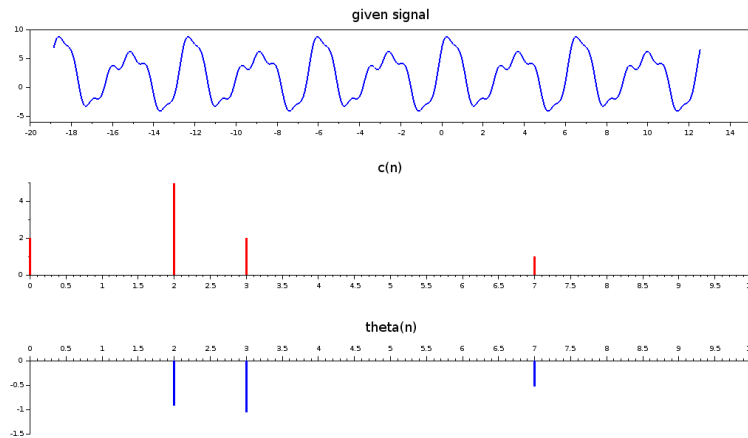


Figure 6.3: Compact trigonometric Fourier series

```

21 cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
22 y= repmat(y,1,5);t=-4.5:dt:(length(y)-1).*dt-4.5; //
    periodic signal definition
23
24 subplot(3,1,1);plot(t,y,'r');title("given signal",
    fontsize",4);
25 set(gca(),"zoom_box",[-4 -1 4 1],"x_location",
    "middle","y_location","middle");
26 subplot(3,1,2);plot2d3(n,cn,[2]);title("c(n)",
    fontsize",4)
27 subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)
    ","fontsize",4)
28 set(gca(),"x_location","middle");

```

Scilab code Exa 6.3 Compact trigonometric Fourier series

```

1 clc
2 clear

```

```

3 close;
4 //error in phase might be there as atan finds angle
  only in 1st quadrant
5
6 w0=1;    T0=2*%pi/w0;//period and frequency
7 dt=0.01;    //time increment
8 t=(0:dt:T0-dt);
9 y=2+3*cos(2*t)+4*sin(2*t)+2*sin(3*t+%pi/6)-cos(7*t
  +150*%pi/180);
10 N=length(y);
11 c0=sum(y.*dt)/T0;
12 n1=[1:10]';
13 for n=1:10
14     aa(n)=2*sum(y.*cos(n*w0*t).*dt)/T0;
15     bb(n)=2*sum(y.*sin(n*w0*t).*dt)/T0;
16     cn(n)=sqrt(aa(n).^2+bb(n).^2);
17     thetan(n)=atan(-bb(n)/aa(n));
18 end
19
20 cn=[c0;cn];thetan=[0;thetan];n=[0;n1];
21 y= repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0;//
  periodic signal definition
22
23 subplot(3,1,1);plot(t,y,'b');title("given signal",
  fontsize",4);
24 subplot(3,1,2);plot2d3(n,cn,[5]);title("c(n)",
  fontsize",4)
25 subplot(3,1,3);plot2d3(n,thetan,[2]);title("theta(n)
  ", "fontsize",4)
26 set(gca(),"x_location","top","zoom_box",[0 -1.5 10
  0]);

```

Scilab code Exa 6.4 Compact trigonometric Fourier series

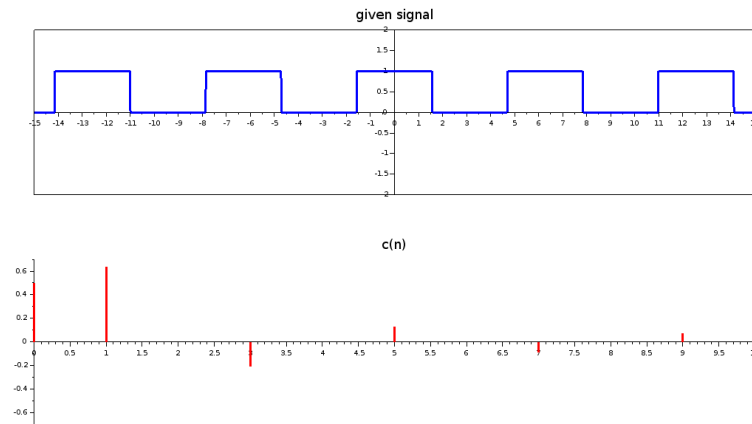


Figure 6.4: Compact trigonometric Fourier series

```

1  clc
2  clear
3  close;
4  //error in phase might be there as atan finds angle
   only in 1st quadrant
5
6  w0=1;T0=2*%pi/w0;//period and frequency
7  dt=0.01;          //time increment
8  t=(-%pi:dt:%pi-dt);
9  y=1.*((t>=-%pi/2)&(t<=%pi/2));
10 N=length(y);
11 c0=0.5;
12 n1=[1:10]';
13 for n=1:10
14     dn(n)=sum(y.*exp(-%i*w0*n*t).*dt)/T0;
15     cn(n)=2*dn(n);
16 end
17
18 cn=[c0;cn];n=[0;n1];
19 y= repmat(y,1,7);t=-3.5*T0:dt:(length(y)-1).*dt-3.5*
   T0;//periodic signal definition
20

```

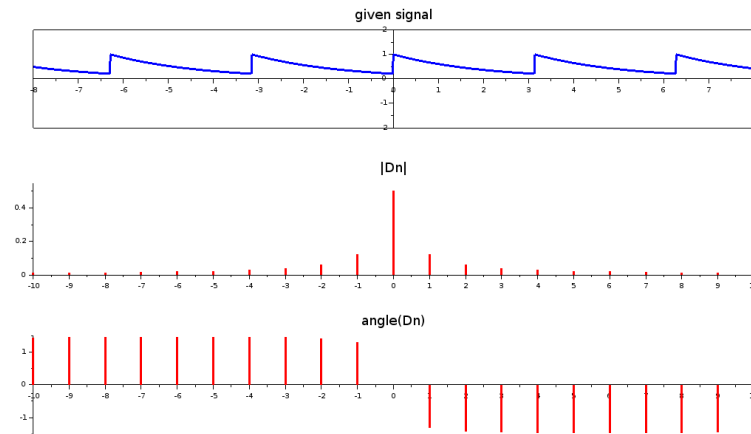


Figure 6.5: exponential Fourier series

```

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

```

Scilab code Exa 6.5 exponential Fourier series

```

1 clc
2 clear
3 close;
4 //error in phase might be there as atan finds angle
    only in 1st quadrant
5
6 w0=2;T0=2*%pi/w0;//period and frequency

```

```

7 dt=0.01;           //time increment
8 t=(0:dt:%pi-dt);
9 y=exp(-t/2);
10 N=length(y);
11 p=1;
12 for n=-10:10
13     dn(p)=sum(y.*exp(-%i*w0*n*t).*dt)/T0;
14     mag_dn(p)=abs(dn(p));
15     ang_dn(p)=phasemag(dn(p))*%pi/180;
16     p=p+1;
17 end
18
19 n=-10:10;
20 y= repmat(y,1,7); t=-3*T0:dt:(length(y)-1).*dt-3*T0; //
    periodic signal definition
21
22 subplot(3,1,1); plot(t,y,'b');
23 title("given signal","fontsize",4);
24 set(gca(),"zoom_box",[-8 -2 8 2],"x_location","
    middle","y_location","middle");
25 subplot(3,1,2); plot2d3(n,mag_dn,[5]); title("    |Dn|
    ","fontsize",4)
26 subplot(3,1,3); plot2d3(n,ang_dn,[5]); title(" angle(Dn
    )","fontsize",4);
27 set(gca(),"x_location","middle");

```

Scilab code Exa 6.6 exponential Fourier series from trigonometric Fourier series

```

1 clc
2 clear
3 close;
4
5
6 n=-12:3:12;
7 function cn=c(n)           //defining given cn

```



```

8     cn=(16-(4/3)*n).*(n>=0);
9     endfunction
10    function thetan=theta(n)    //defining given theta_n
11        thetan=((-%pi/12)*n).*((n>=0)&(n<=6))+((%pi/12)
            *(n-12)).*(n>6);
12    endfunction
13
14    dn=(c(n)+c(-n))/2;
15    d_theta=(theta(n)-theta(-n));
16    figure(1)
17    subplot(1,2,1);plot2d3(n,c(n));title("C_n given", "
        fontsize",4)
18    subplot(1,2,2);plot2d3(n,theta(n));title("Theta_n
        given", " fontsize",4)
19    set(gca(),"x_location","top");
20    figure(2)
21    subplot(1,2,1);plot2d3(n,dn);title("|Dn|_calculated"
        ," fontsize",4)
22    subplot(1,2,2);plot2d3(n,d_theta);title("Theta(Dn)
        _calculated", " fontsize",4)
23    set(gca(),"x_location","middle");

```

Scilab code Exa 6.7 trigonometric Fourier series of impulse train

```

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

```

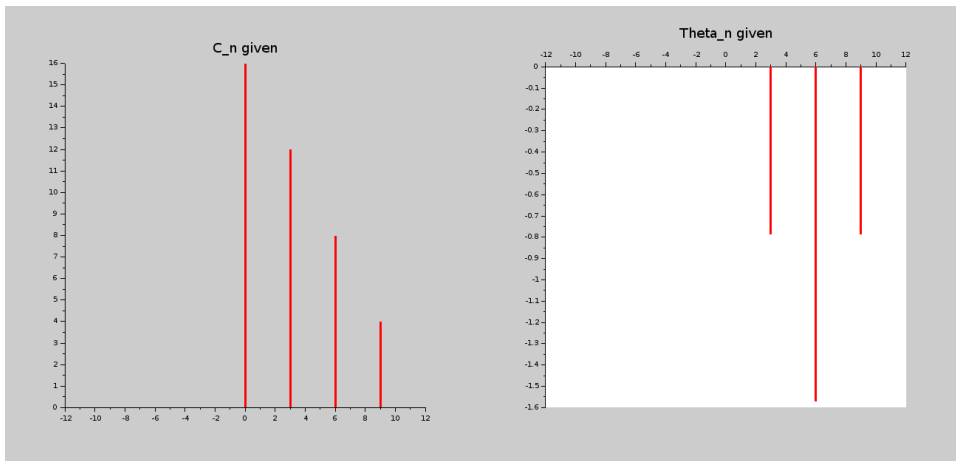


Figure 6.6: exponential Fourier series from trigonometric Fourier series

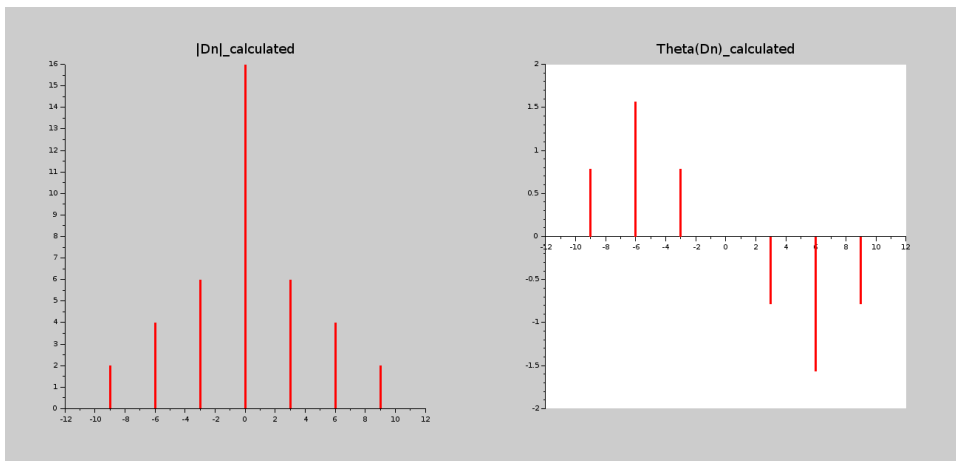


Figure 6.7: exponential Fourier series from trigonometric Fourier series

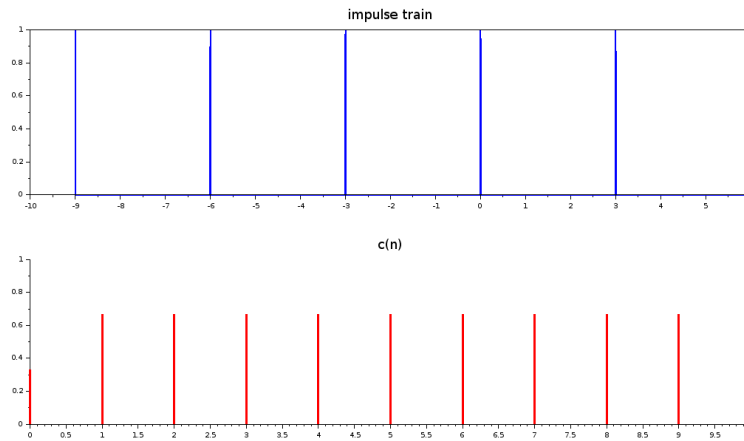


Figure 6.8: trigonometric Fourier series of impulse train

```

5
6 //taking period of 3
7 T0=3;
8 dt=0.01;           //time increment
9 t=(0:dt:T0-dt);
10 y=1.*(t==0); //defining impulse function
11 c0=1/T0;
12 n1=[1:10]';
13 for n=1:10
14     cn(n)=2/T0;
15 end
16
17 cn=[c0;cn];n=[0;n1];
18 y= repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0; //
    periodic signal definition
19
20 subplot(2,1,1);plot(t,y,'b');title("impulse train",
    fontsize",4);
21 subplot(2,1,2);plot2d3(n,cn,[5]);title("c(n)",
    fontsize",4);
22 set(gca(),"zoom_box",[0 0 10 1]);

```

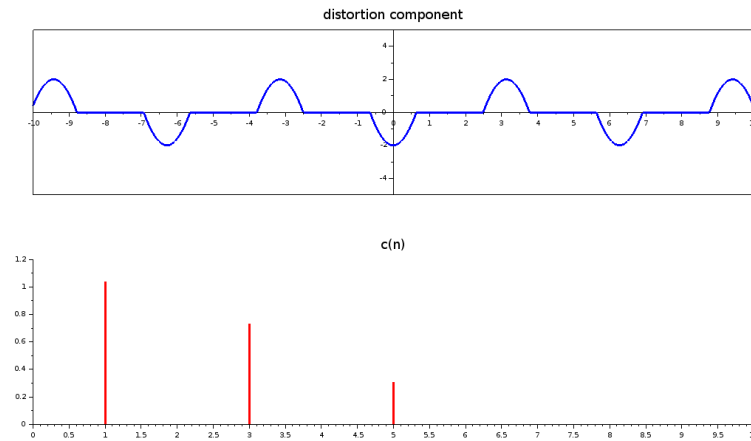


Figure 6.9: Harmonic distortion in audio amplifier

Scilab code Exa 6.8 Harmonic distortion in audio amplifier

```

1  clc
2  clear
3  close;
4  //error in phase might be there as atan finds angle
   only in 1st quadrant
5
6  w0=1;    T0=2*%pi/w0;//assuming frequency 1Hz
7  dt=0.01;    //time increment
8  t=(-T0/2:dt:T0/2-dt);
9  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
10 N=length(y);
11 c0=sum(y.*dt)/T0;

```

```

12 n1=[1:10]';
13 for n=1:10
14     dn(n)=sum(y.*exp(-%i*n*w0*t).*dt)/T0;
15     cn(n)=2*dn(n);
16 end
17
18 cn=[c0;cn];n=[0;n1];
19 y= repmat(y,1,5);t=-3*T0:dt:(length(y)-1).*dt-3*T0; //
    periodic signal definition
20 subplot(2,1,1);plot(t,y,'b');title("distortion
    component","fontsize",4);
21 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -5 10 5]);
22 subplot(2,1,2);plot2d3(n,cn,[5]);title("c(n)","
    fontsize",4)
23 set(gca(),"zoom_box",[0 0 10 1.2]);
24
25 //distortion calculation
26 en=50; //energy of undistorted signal
27 P=integrate("(10*cos(w0*t)-8).^2","t",0,0.1024*T0)
    *4/T0; //total harmonic power
28 P_3rd=((abs(cn(4))).^2)/2; // power of
    third harmonic
29 D_tot=(P/en)*100;
30 D_3rd=(P_3rd/en)*100;
31 printf("\nTotal harmonic distortion is %.2f percent
    \n 3rd harmonic distortion is %.2f percent\n",
    D_tot,D_3rd);
32 //round-off error may be there

```

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

```
1 clc
```

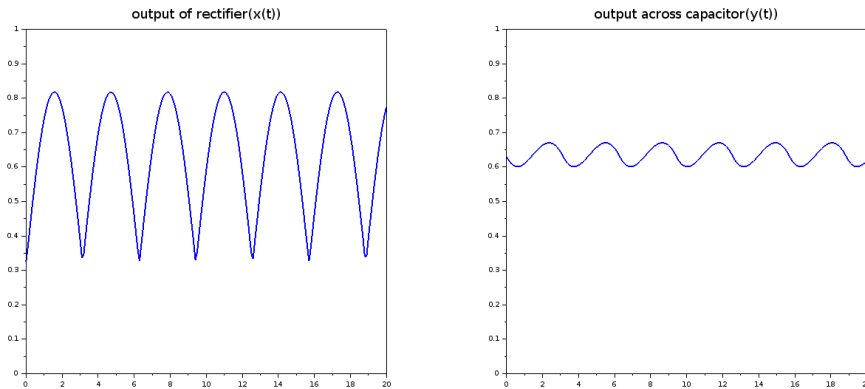


Figure 6.10: dc value and rms value of rectified signal

```

2  clear;
3  close;
4
5  t=0:0.1:20;
6  pi=%pi;
7  n=[0:20];
8  p=1;
9  //finding x(t) and y(t) from fourier series coefficients
10 for t1=t
11     cx=0;
12     cy=0;
13     for n1=n
14         cx=cx+(2/(pi*(1-4*n1.^2)))*exp(%i*2*n1*t1);
15         cy=cy+(2/(pi*(1-4*n1.^2)*(1+6*i*n1)))*exp(
            %i*2*n1*t1);
16     end
17     x(p)=cx;
18     y(p)=cy;
19     p=p+1;
20 end
21 subplot(1,2,1);plot(t',x);title("output of rectifier

```

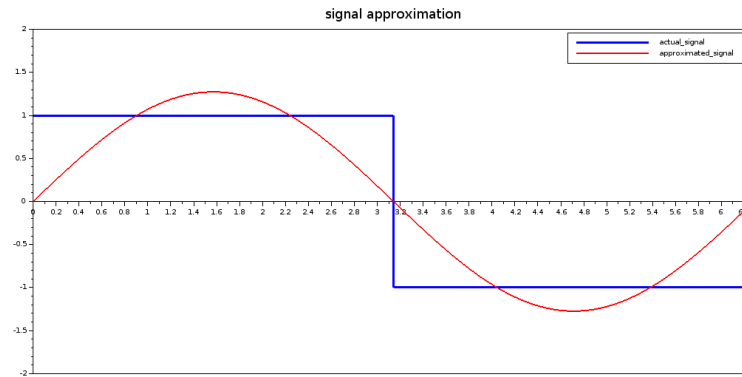


Figure 6.11: Signal approximation with minimum error signal

```

    (x(t))", "font_size", 4);
22 subplot(1,2,2); plot(t',y); title("output across
    capacitor(y(t))", "font_size", 4)
23 zoom_rect([t(1) 0 t($) 1]);
24 //finding ripple
25 DC=2/%pi;
26 power=0;
27 for n=1:50
28     Dn(n)=2/(pi*(1-4*n.^2)*(1+6*i*n));
29     power=power+2*sum((abs(Dn(n))).^2);
30 end
31 rms=sqrt(power);
32 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

```
1 clc
```

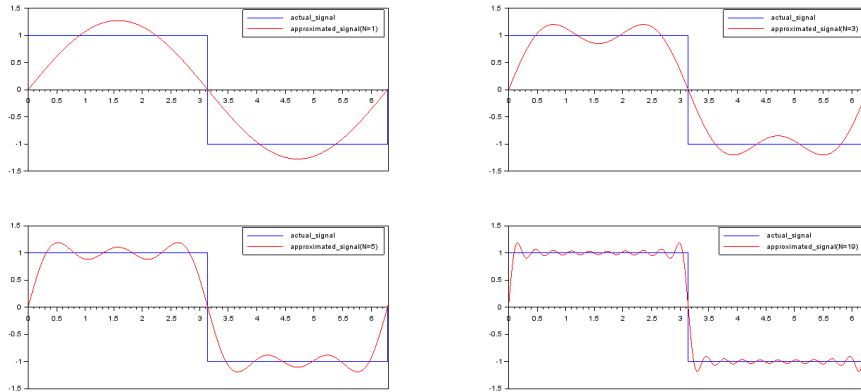


Figure 6.12: Regeneration of signal from Fourier series coefficient

```

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

```

Scilab code Exa 6.11 Regeneration of signal from Fourier series coefficient


```

1  clc
2  clear all
3  close;
4
5  t=0:0.001:6.3;
6  x=1.*((t>=0)&(t<%pi))-1.*((t>=%pi)&(t<=(2*%pi)));
7  p=[1,3,5,19];
8
9  x_approx=0;
10 for n=1:2:p(1)
11     x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
12 end
13 subplot(2,2,1);plot(t',x,'b');plot(t,x_approx,'r');
14 set(gca(),"x_location","middle","zoom_box",[t(1)
    -1.5 t($) 1.5]);
15 legend("actual_signal","approximated_signal(N=1)");
16
17 x_approx=0;
18 for n=1:2:p(2)
19     x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
20 end
21 subplot(2,2,2);plot(t',x,'b');plot(t,x_approx,'r');
22 set(gca(),"x_location","middle","zoom_box",[t(1)
    -1.5 t($) 1.5]);
23 legend("actual_signal","approximated_signal(N=3)");
24
25 x_approx=0;
26 for n=1:2:p(3)
27     x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);
28 end
29 subplot(2,2,3);plot(t',x,'b');plot(t,x_approx,'r');
30 set(gca(),"x_location","middle","zoom_box",[t(1)
    -1.5 t($) 1.5]);
31 legend("actual_signal","approximated_signal(N=5)");
32
33 x_approx=0;
34 for n=1:2:p(4)
35     x_approx=x_approx+(4/%pi).*sin(n*t).*(1/n);

```

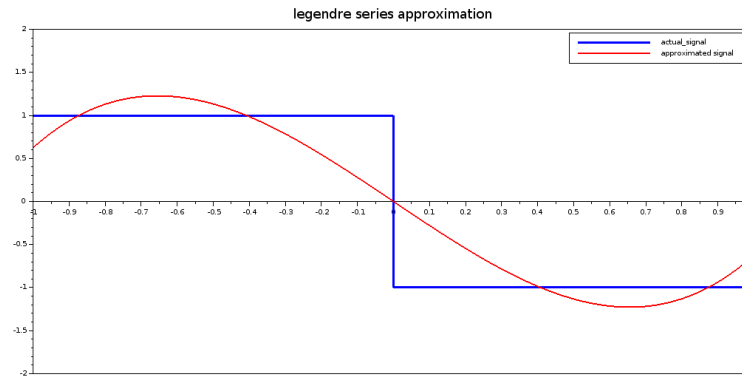


Figure 6.13: Legendre Fourier series

```

36 end
37 subplot(2,2,4);plot(t',x,'b');plot(t,x_approx,'r');
38 set(gca(),"x_location","middle","zoom_box",[t(1)
    -1.5 t($) 1.5]);
39 legend("actual_signal","approximated_signal(N=19)");
40
41 //Finding Error
42 E1=2*%pi-(16/%pi); //for N=1
43 E3=2*%pi-(16/%pi)*(1+1/9); //for N=3
44 print(E1)
45 print(E3)

```

Scilab code Exa 6.12 Legendre Fourier series

```

1 clc
2 clear all
3 close;
4

```

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

Chapter 7

Continuous time signal analysis The Fourier transform

Scilab code Exa 7.1 Fourier transform of given signal

```
1  clc
2  clear
3  close;
4  //phase error will be there due to round-off error
5
6  t=0:0.009:2;
7  dt=0.009;
8  x=exp(-2*t);
9  omega=-20:0.009:20;
10 p=0;
11 for om=-20:0.009:20
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;
16 subplot(1,3,1);plot(t,x, 'k');
17 title("given signal  $-(e^{-2t}) * u(t)$ ", "fontsize",4);
```

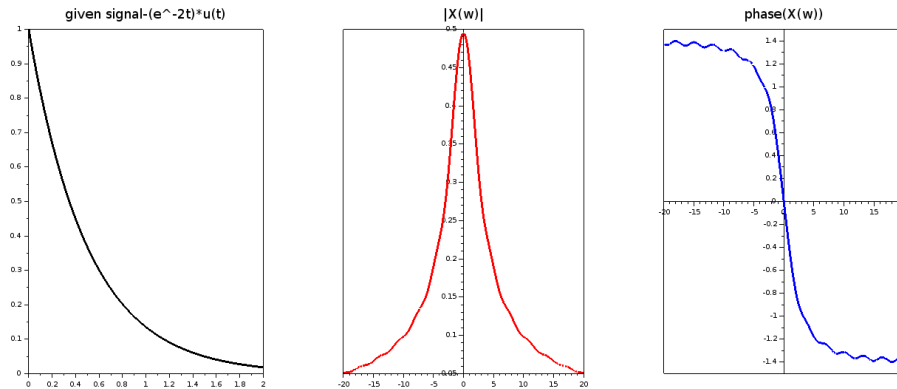


Figure 7.1: Fourier transform of given signal

```

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

```

Scilab code Exa 7.2 Fourier transform of given rect signal

```

1 clc
2 clear
3 close;
4 //taking tau=2
5 t=-4:0.009:4;
6 dt=0.009;
7 x=1.*((t>=-1)&(t<=1));
8 omega=-20:0.009:20;
9 p=0;
10 for om=-20:0.009:20

```

```

11     p=p+1;
12     X(p)=sum(x.*exp(-%i*om*t).*dt);
13 end
14 X=round(X*10000)/10000;
15 figure(1)
16 subplot(1,2,1);plot(t,x,'k');
17 title("given signal-rect(t/2)","fontsize",4);
18 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-4 -2 4 2]);
19 subplot(1,2,2);plot(omega,X,'r');
20 title("X(w)","fontsize",4);
21 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-20 -2 20 2]);
22
23 figure(2)
24 subplot(1,2,1);plot(omega,abs(X),'r');
25 title("|X(w)|","fontsize",4);
26 subplot(1,2,2);plot(omega,-phasemag(X)*%pi/180,'r');
27 title("phase(X(w))","fontsize",4);
28 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

```

1  clc
2  clear
3  close;
4
5  t=-2:0.001:2;

```

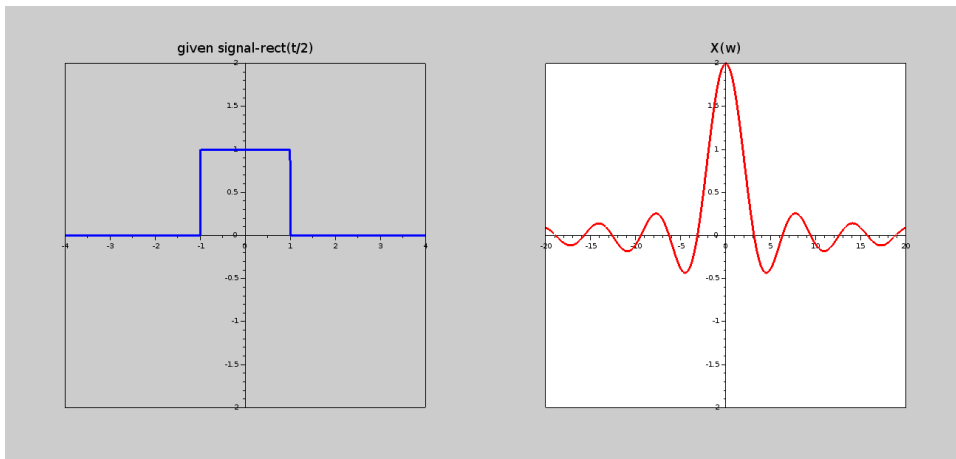


Figure 7.2: Fourier transform of given rect signal

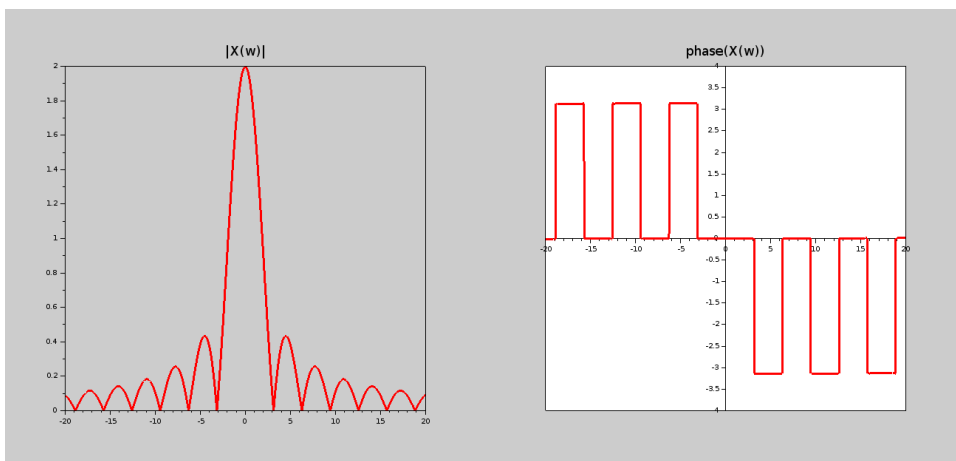


Figure 7.3: Fourier transform of given rect signal

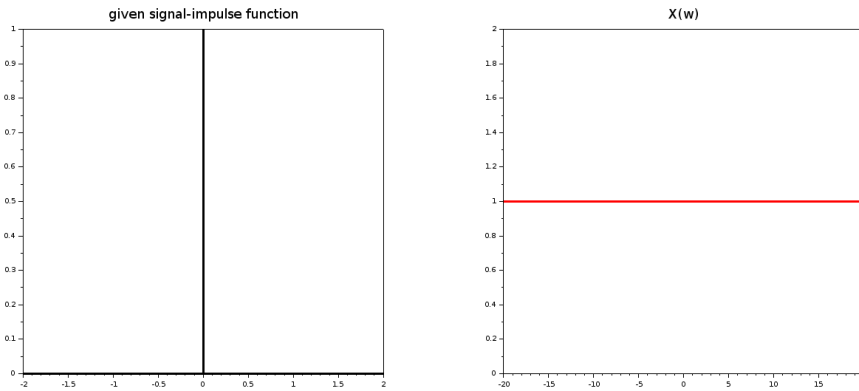


Figure 7.4: Fourier transform of given impulse signal

```

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

```

Scilab code Exa 7.4 Inverse fourier transform of impulse signal

```

1 clc
2 clear
3 close;
4
5 t=0:0.001:2;
6 dt=0.001;

```

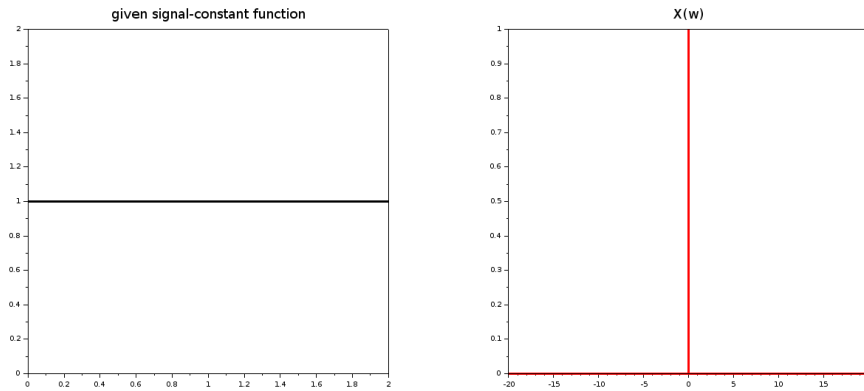



Figure 7.5: Inverse fourier transform of impulse signal

```

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

```

Scilab code Exa 7.6 Fourier transform of given everlasting sinusoidal signal

```

1 clc
2 clear
3 close;
4 //taking wo=2*%pi
5
6 t=-8:0.01:8;
7 dt=0.01;

```

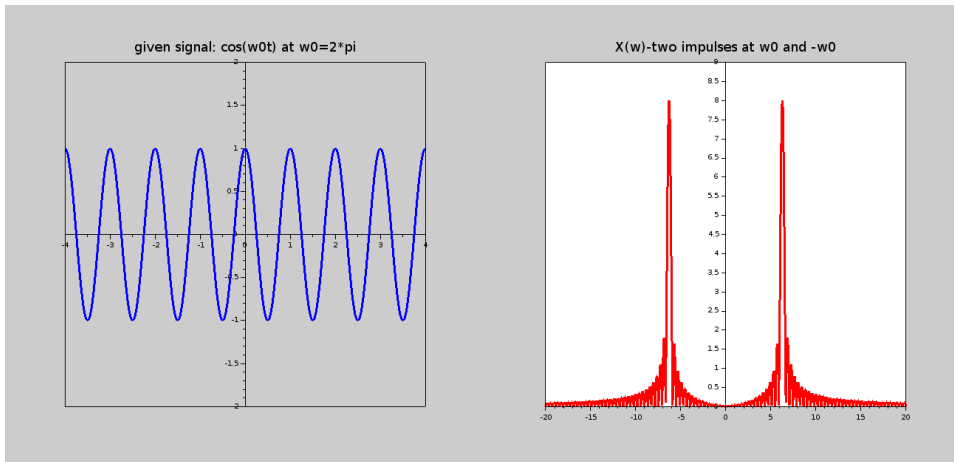


Figure 7.6: Fourier transform of given everlasting sinusoidal signal

```

8 x=cos(2*%pi*t);
9 omega=-20:0.01:20;
10 p=0;
11 for om=-20:0.01:20
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;
16 figure(1)
17 subplot(1,2,1);plot(t,x,'b');
18 title("given signal: cos(w0t) at w0=2*pi", "fontsize"
19     ,4);
19 set(gca(),"x_location","middle","y_location","middle"
20     ,"zoom_box",[-4 -2 4 2]);
20 subplot(1,2,2);plot(omega,abs(X),'r');
21 title("X(w)-two impulses at w0 and -w0", "fontsize"
22     ,4);
22 set(gca(),"y_location","middle");

```

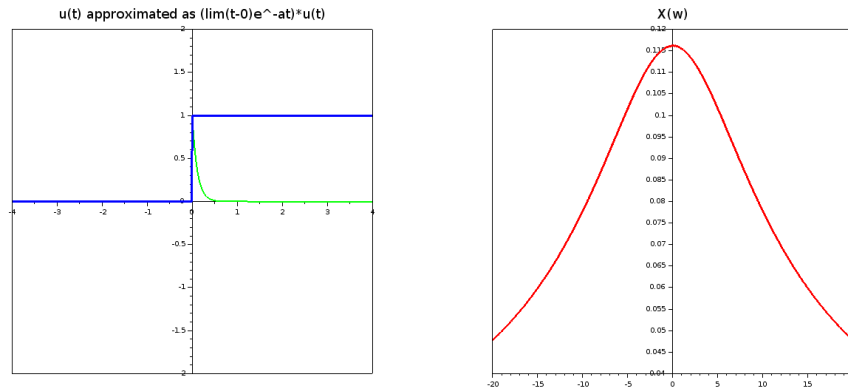


Figure 7.7: Fourier transform of given unit step signal

Scilab code Exa 7.9 Fourier transform of given unit step signal

```

1  clc
2  clear
3  close;
4
5  t=-4:0.01:4;
6  dt=0.01;
7  x=exp(-9*t).*(t>=0);
8  x1=1.*(t>=0);
9  omega=-20:0.01:20;
10 p=0;
11 for om=-20:0.01:20
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;
16 subplot(1,2,1);plot(t,x,'b');plot(t,x1,'g');
17 title("u(t) approximated as (lim(t-0)e^-at)*u(t)",)

```

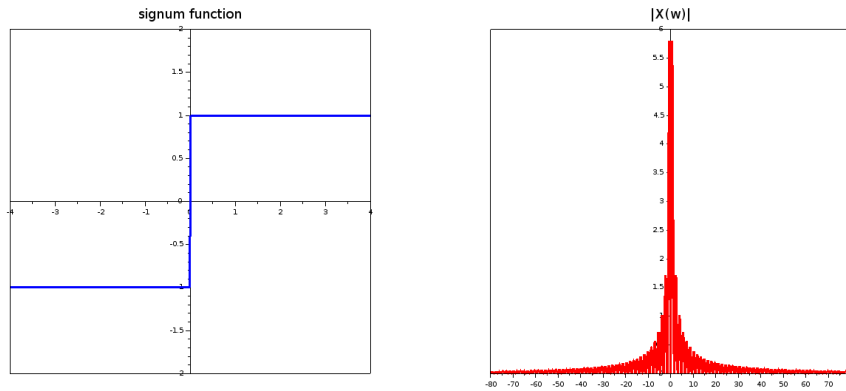


Figure 7.8: Fourier transform of given Signum function

```

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

```

Scilab code Exa 7.10 Fourier transform of given Signum function

```

1 //the result is the outer covering of the obtained
  plot
2 clc
3 clear
4 close;
5
6 t=-4:0.01:4;
7 dt=0.01;
8 x=sign(t);

```

```

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

```

Scilab code Exa 7.11 Duality property

```

1  //taking tau=2
2  clc
3  clear
4  close;
5
6  t=-20:0.01:20;
7  dt=0.01;
8  x1=1.*((t>=-1)&(t<=1));
9  x2=2*sinc(t);
10 omega=-20:0.01:20;
11 p=0;
12 for om=-20:0.01:20
13   p=p+1;
14   X1(p)=sum(x1.*exp(-%i*om*t).*dt);
15   X2(p)=sum(x2.*exp(-%i*om*t).*dt);

```

```

16 end
17 X1=round(X1*10000)/10000;
18 X2=round(X2*10000)/10000;
19 figure(1)
20 subplot(1,2,1);plot(t,x1,'b');
21 title("x(t): rect(t/2)","fontsize",4);
22 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -2 10 2]);
23 subplot(1,2,2);plot(omega,X1,'r');
24 title("X(w): 2*sinc(w)","fontsize",4);
25 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[omega(1) -2 omega($) 2]);
26 figure(2)
27 subplot(1,2,1);plot(t,x2,'b');
28 title("x(t): 2*sinc(t)","fontsize",4);
29 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[t(1) -2 t($) 2]);
30 subplot(1,2,2);plot(omega,X2,'r');
31 title("X(w): 2*pi*rect(w/2)","fontsize",4);
32 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -7 10 7]);

```

Scilab code Exa 7.12 Fourier transform of given signals

```

1 //taking a=2
2 clc
3 clear
4 close;
5
6 t=-10:0.01:10;
7 dt=0.01;

```

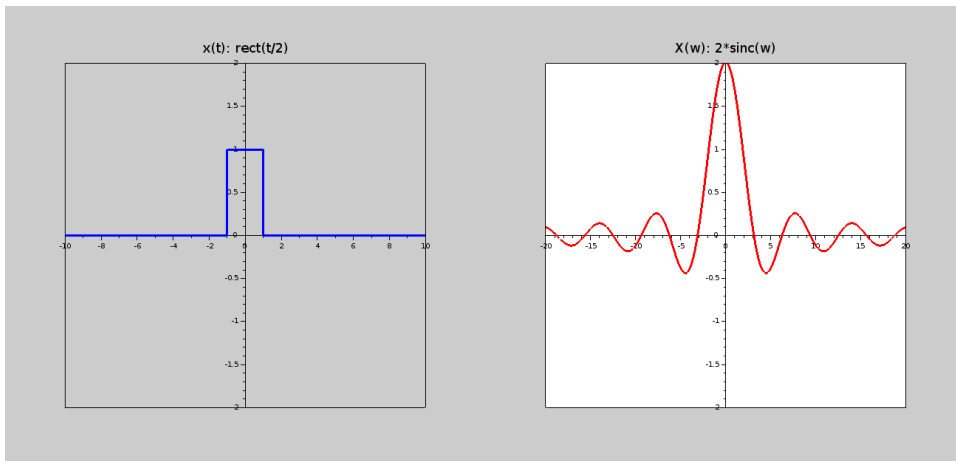


Figure 7.9: Duality property

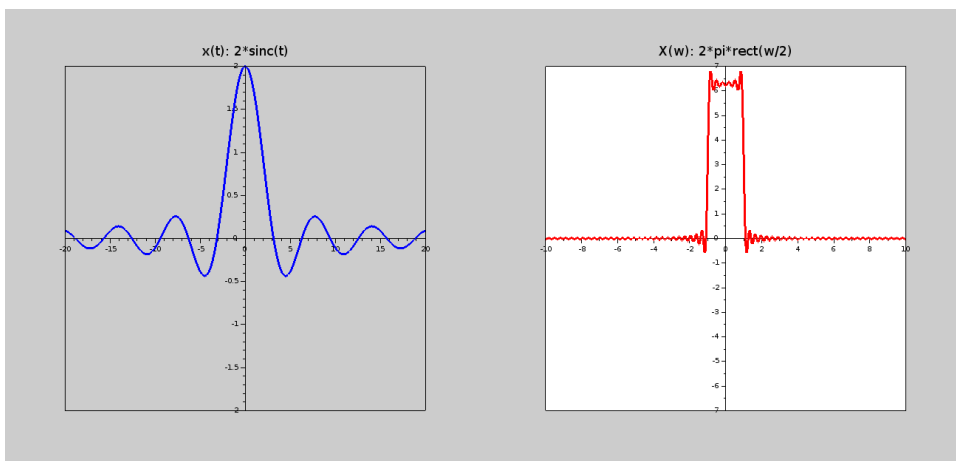


Figure 7.10: Duality property

```

8 x1=exp(2*t).*(t<=0); //defining e^at*u(-t)
9 x2=exp(-2*abs(t)); //defining e^a|t|
10 omega=-10:0.01:10;
11 p=0;
12 for om=-10:0.01:10
13     p=p+1;
14     X1(p)=sum(x1.*exp(-%i*om*t).*dt);
15     X2(p)=sum(x2.*exp(-%i*om*t).*dt);
16 end
17 X1=round(X1*10000)/10000;
18 X2=round(X2*10000)/10000;
19 figure(1)
20 subplot(1,2,1);plot(t,x1,'b');
21 title("x1(t): (e^at)*u(-t)", "fontsize",4);
22 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -1.2 10 1.2]);
23 subplot(1,2,2);plot(omega,X1,'r');
24 title("X1(w): 2*sinc(w)", "fontsize",4);
25 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -2 10 2]);
26 figure(2)
27 subplot(1,2,1);plot(t,x2,'b');
28 title("x2(t): e^-a|t|", "fontsize",4);
29 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -1.2 10 1.2]);
30 subplot(1,2,2);plot(omega,X2,'r');
31 title("X2(w)", "fontsize",4);
32 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -2 10 2]);

```

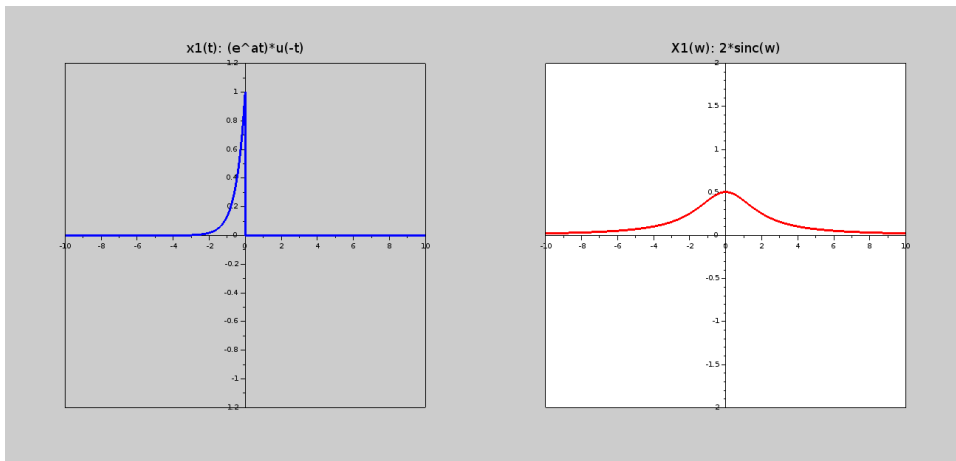


Figure 7.11: Fourier transform of given signals

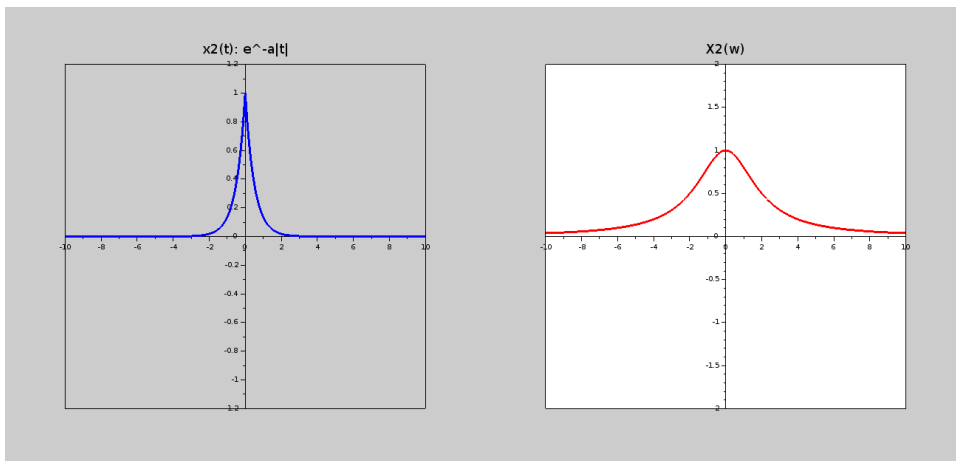


Figure 7.12: Fourier transform of given signals

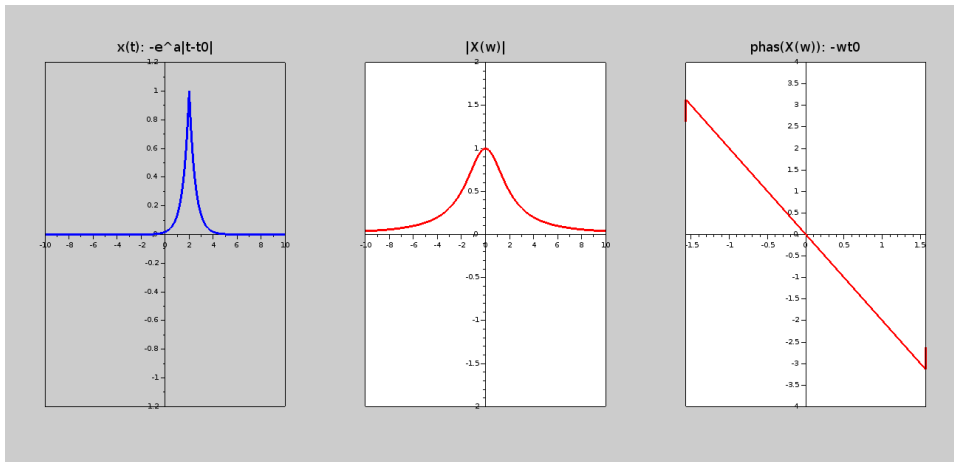


Figure 7.13: Fourier transform of time shifted signal

Scilab code Exa 7.13 Fourier transform of time shifted signal

```

1 //taking a=2,t0=2
2 clc
3 clear
4 close;
5
6 t=-10:0.01:10;
7 dt=0.01;
8 x=exp(-2*abs(t-2)); //defining e^a|t|
9 omega=-10:0.01:10;
10 p=0;
11 for om=-10:0.01:10
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;
16 figure(1)
17 subplot(1,3,1);plot(t,x,'b');
18 title("x(t): -e^a|t-t0|", "fontsize", 4);
19 set(gca(), "x_location", "middle", "y_location", "middle",
    "zoom_box", [-10 -1.2 10 1.2]);

```

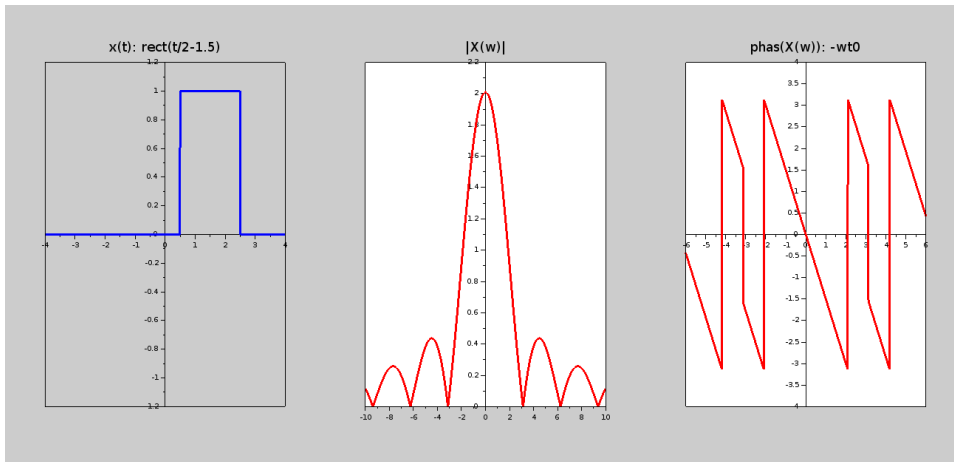


Figure 7.14: Fourier transform of time shifted rect signal

```

20 subplot(1,3,2);plot(omega,abs(X),'r');
21 title("|X(w)|","fontsize",4);
22 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-10 -2 10 2]);
23 subplot(1,3,3);plot(omega,phasemag(X)*%pi/180,'r');
24 title("phas(X(w)): -wt0","fontsize",4);
25 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-%pi/2 -4 %pi/2 4]);
26
27 //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

```

1 //taking tau=2
2 clc

```

```

3 clear
4 close;
5
6 t=-10:0.01:10;
7 dt=0.01;
8 x=1.*((t>=0.5)&(t<=2.5)); //rect(t/2) delayed by (3*
    tau/4)=1.5
9 omega=-10:0.01:10;
10 p=0;
11 for om=-10:0.01:10
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;
16 ph=round(phasemag(X)*10000)/10000;
17 figure(1)
18 subplot(1,3,1);plot(t,x,'b');
19 title("x(t): rect(t/2-1.5)","fontsize",4);
20 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-4 -1.2 4 1.2]);
21 subplot(1,3,2);plot(omega,abs(X),'r');
22 title("|X(w)|","fontsize",4);
23 set(gca(),"y_location","middle");
24 subplot(1,3,3);plot(omega,ph*%pi/180,'r');
25 title("phas(X(w)): -wt0","fontsize",4);
26 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-6 -4 6 4]);

```

Scilab code Exa 7.15 Fourier transform of modulated signal

```

1 clc
2 clear
3 close;

```

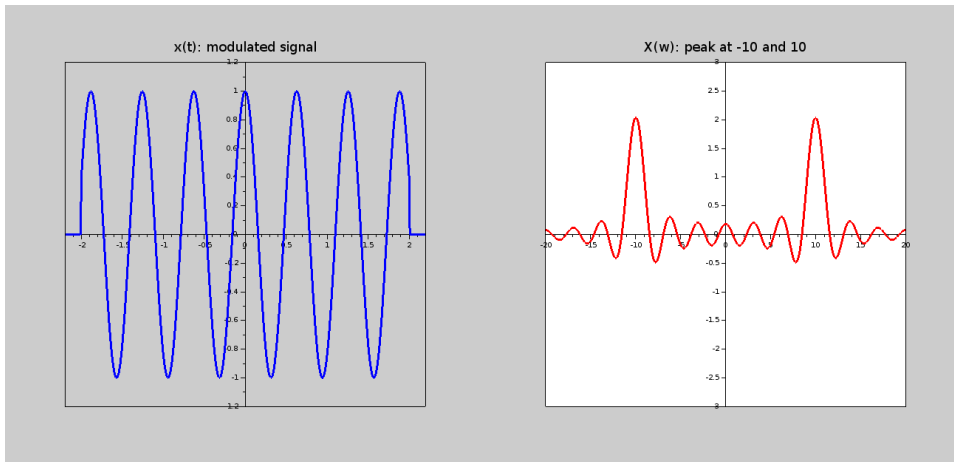


Figure 7.15: Fourier transform of modulated signal

```

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

```

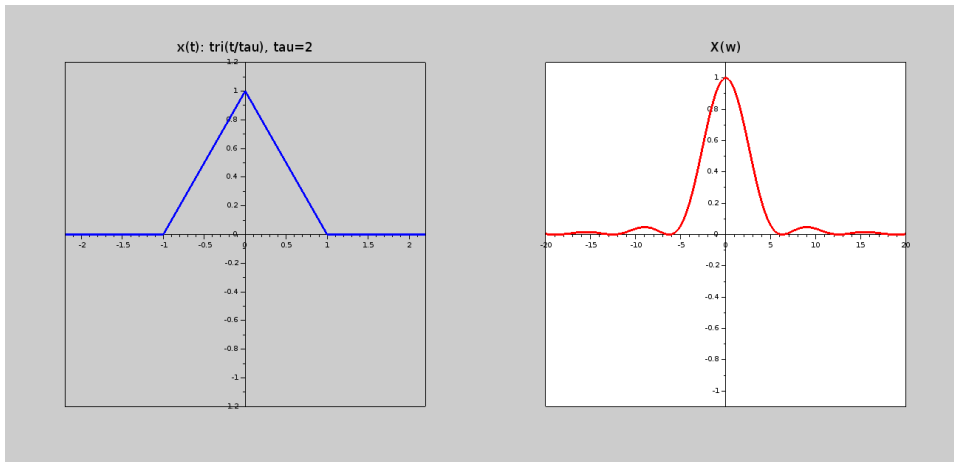


Figure 7.16: Fourier transform of given triangle pulse

Scilab code Exa 7.17 Fourier transform of given triangle pulse

```

1 //taking tau=2
2 clc
3 clear
4 close;
5
6 t=-3:0.01:5;
7 dt=0.01;
8 x=(t+1).*((t>=-1)&(t<=0))+(1-t).*((t>0)&(t<=1));
9 omega=-20:0.01:20;
10 p=0;
11 for om=-20:0.01:20
12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 X=round(X*10000)/10000;

```

```

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

```

Scilab code Exa 7.19 Filtered signals

```

1 clc
2 clear
3 close;
4
5 //*****part a
6 //defining x(t)
7 function x=p(t)
8     x=cos(10*pi*(t-0.05)).*((t>=0)&(t<=0.1));
9 endfunction
10
11 t=-0.12:0.0009:0.12;
12 wc=2000*pi;
13 x=p(t);
14 z=x.*cos(wc*t);
15 tg=10^-3;
16 y=2*p(t-tg).*cos(wc*(t-tg)-0.4*pi);
17
18 figure(1)
19 subplot(1,3,1);plot(t,x,'b');
20 title("x(t)","fontsize",4);
21 set(gca(),"zoom_box",[t(1) -2 t($) 2],"x_location","
    middle","y_location","middle");

```

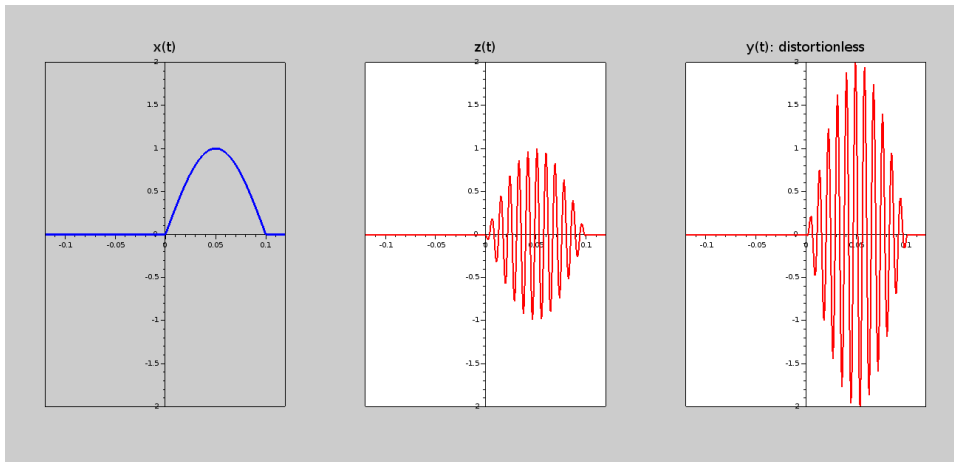


Figure 7.17: Filtered signals

```

22 subplot(1,3,2);plot(t,z,'r');
23 title("z(t)","fontsize",4);
24 set(gca(),"zoom_box",[t(1)-2 t($)+2],"x_location","
    middle","y_location","middle");
25 subplot(1,3,3);plot(t,y,'r');
26 title("y(t): distortionless","fontsize",4);
27 set(gca(),"zoom_box",[t(1)-2 t($)+2],"x_location","
    middle","y_location","middle");
28
29 //*****part b
    *****
30 wc=4000*%pi;
31 y2=1.5*p(t).*cos(wc*t-3.1*%pi);
32 figure(2)
33 plot(t,y2,'b');
34 title("(part_b) y2(t): distortionless","fontsize",4)
    ;
35 set(gca(),"zoom_box",[t(1)-2 t($)+2],"x_location","
    middle","y_location","middle");

```

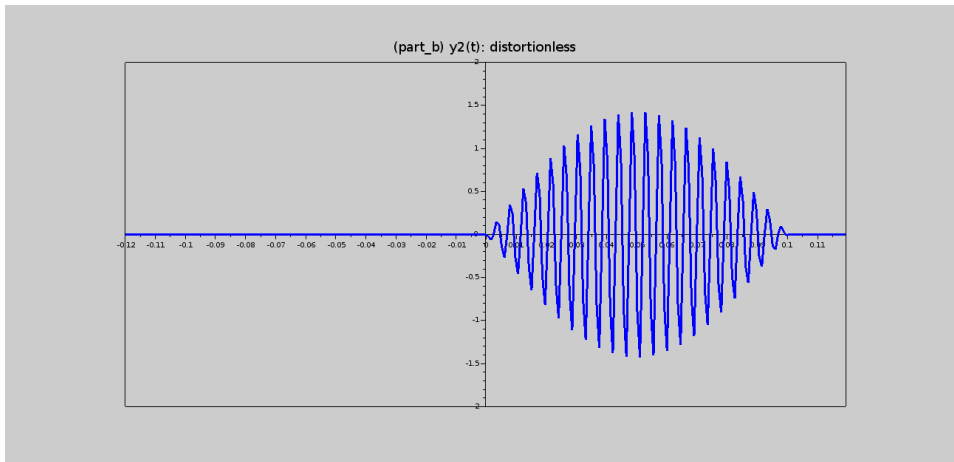



Figure 7.18: Filtered signals

Scilab code Exa 7.20 Calculation of frequency range with signal energy

```

1  clc
2  clear
3  close;
4
5  a=1; //taking a=1
6  E=1/(2*a);
7  W=a*tan(0.95*%pi*a*E);
8  printf("\nthe frequency upto W=%0.3f rad/s will
        contain \n95 percent of the signal energy\n",W);

```

Scilab code Exa 7.23 Toned modulated signals

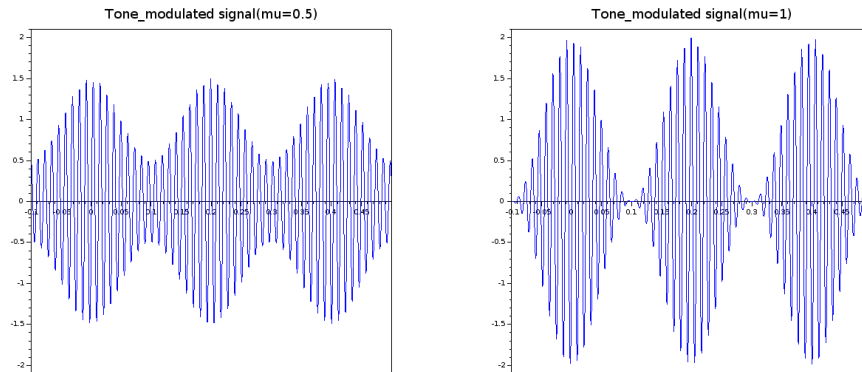


Figure 7.19: Toned modulated signals

```

1  clc
2  clear
3  close;
4  //assuming some random values
5  A=1;
6  wm=10*%pi;
7  wc=2000*%pi;
8  t=-0.1:0.00092:0.5;
9  m1=0.5;
10 m2=1;
11 y_am1=A*(1+m1*cos(wm*t)).*cos(wc*t);
12 y_am2=A*(1+m2*cos(wm*t)).*cos(wc*t);
13 subplot(1,2,1);plot(t,y_am1,'b');
14 title("Tone_modulated signal(mu=0.5)","fontsize",4);
15 set(gca(),"zoom_box",[t(1) -2.1 t($) 2.1],"
    x_location","middle");
16 subplot(1,2,2);plot(t,y_am2,'b');
17 title("Tone_modulated signal(mu=1)","fontsize",4);
18 set(gca(),"zoom_box",[t(1) -2.1 t($) 2.1],"
    x_location","middle");

```

Chapter 8

Sampling

Scilab code Exa 8.1 Effects of different sampling rates

```
1  clc
2  clear
3  close;
4  t=-5:0.01:5;
5  dt=0.01;
6  x=sinc(5*pi*t).^2;
7  omega=-30:0.01:30;
8  p=0;
9  for om=-30:0.01:30
10     p=p+1;
11     X(p)=sum(x.*exp(-%i*om*t).*dt);
12 end
13 X=round(X*10000)/10000;
14 figure(1)
15 subplot(2,2,1);plot(t,x,'b');
16 title("x(t)=sinc(5*pi*t).^2","fontsize",4);
17 set(gca(),"x_location","middle","y_location","middle",
    "","zoom_box",[-2 -1.2 2 1.2]);
18 subplot(2,2,2);plot(omega,X,'r');
19 title("X(w)","fontsize",4);
20 set(gca(),"y_location","middle","x_location","middle
```

```

    ", "zoom_box" , [-30 -0.4 30 0.4]);
21
22 //Under sampling(period=0.2)
23 p=1;
24 r=0.2;
25 for n=t(1)/r:1:t($)/r
26     x1(p)=sinc(5*%pi*r*n).^2;
27     p=p+1;
28 end
29 x1=x1';
30 t1=t(1)/r:1:t($)/r;
31 omega=-30:0.01:30;
32 p=0;
33 for om=-30:0.01:30
34     p=p+1;
35     X1(p)=sum(x1.*exp(-%i*om*t1));
36 end
37 X1=round(X1*10000)/10000;
38 subplot(2,2,3);plot2d3(t1,x1,[2]);
39 title("Under sampling(T=0.2)","fontsize",4);
40 set(gca(),"x_location","middle","y_location","middle
    ", "zoom_box" , [-2 -1.2 2 1.2]);
41 subplot(2,2,4);plot(omega,abs(X1),'r');
42 title("X(w): signal unrecoverable","fontsize",4);
43 set(gca(),"y_location","middle","x_location","middle
    ", "zoom_box" , [-20 -4 20 4]);
44
45 //Nyquist rate(period=0.1)
46 p=1;
47 r=0.1;
48 for n=t(1)/r:1:t($)/r
49     x2(p)=sinc(5*%pi*r*n).^2;
50     p=p+1;
51 end
52 x2=x2';
53 t2=t(1)/r:1:t($)/r;
54 omega=-30:0.01:30;
55 p=0;

```

```

56 for om=-30:0.01:30
57     p=p+1;
58     X2(p)=sum(x2.*exp(-%i*om*t2));
59 end
60 X2=round(X2*10000)/10000;
61 figure(2)
62 subplot(2,2,1);plot2d3(t2,x2,[2]);
63 title("sampling at Nyquist rate","fontsize",4);
64 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-4 -1.2 4 1.2]);
65 subplot(2,2,2);plot(omega,abs(X2),'r');
66 title("X(w)","fontsize",4);
67 set(gca(),"y_location","middle","x_location","middle
    ","zoom_box",[-20 -4 20 4]);
68
69 //oversampling(period=0.05)
70 p=1;
71 r=0.05;
72 for n=t(1)/r:1:t($)/r
73     x3(p)=sinc(5*pi*r*n).^2;
74     p=p+1;
75 end
76 x3=x3';
77 t3=t(1)/r:1:t($)/r;
78 omega=-30:0.01:30;
79 p=0;
80 for om=-30:0.01:30
81     p=p+1;
82     X3(p)=sum(x3.*exp(-%i*om*t3));
83 end
84 X3=round(X3*10000)/10000;
85 subplot(2,2,3);plot2d3(t3,x3,[2]);
86 title("oversampling","fontsize",4);
87 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-4 -1.2 4 1.2]);
88 subplot(2,2,4);plot(omega,abs(X3),'r');
89 title("X(w)","fontsize",4);
90 set(gca(),"y_location","middle","x_location","middle

```

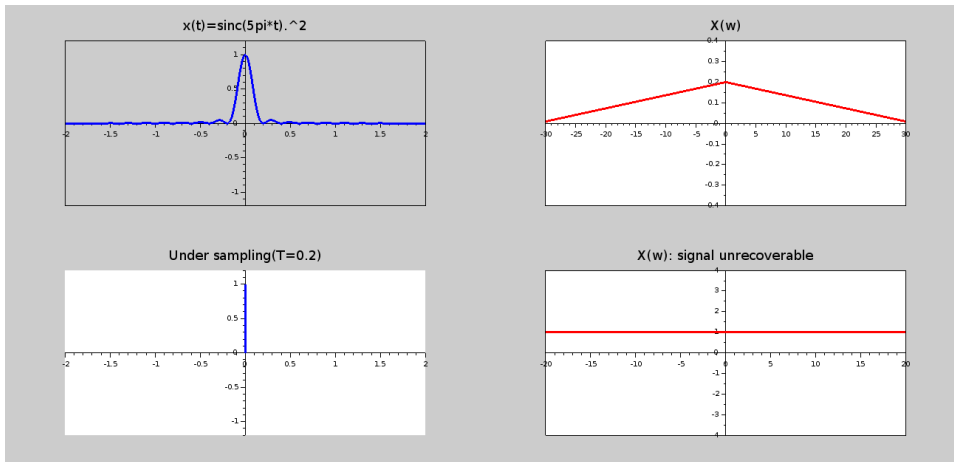


Figure 8.1: Effects of different sampling rates

”, ”zoom_box” , [-20 -4 20 4]);

Scilab code Exa 8.2 Practical sampling

```

1 //this code will take some time to give result as
  increments are very small
2 clc
3 clear
4 close;
5
6 t=-0.5:0.01:0.5;
7 dt=0.01;
8 x=sinc(5*%pi*t).^2;
9 omega=-30:0.01:30;
10 p=0;
11 for om=-30:0.01:30

```

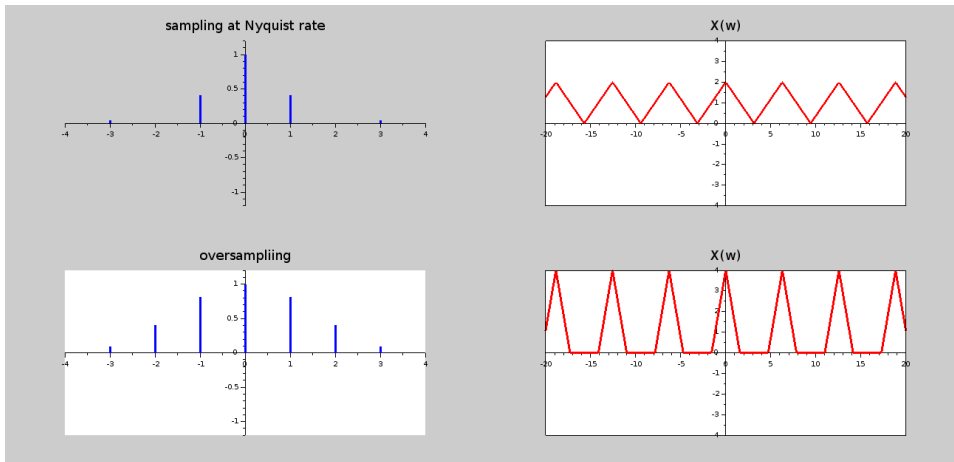


Figure 8.2: Effects of different sampling rates

```

12     p=p+1;
13     X(p)=sum(x.*exp(-%i*om*t).*dt);
14 end
15 figure(1)
16 subplot(1,2,1);plot(t,x,'b');
17 title("x(t)=sinc(5*pi*t).^2","fontsize",4);
18 set(gca(),"x_location","middle","y_location","middle",
19     "zoom_box",[-0.3 -1.2 0.3 1.2]);
19 subplot(1,2,2);plot(omega,X,'r');
20 title("X(w)","fontsize",4);
21 set(gca(),"y_location","middle","x_location","middle",
22     "zoom_box",[-30 -0.4 30 0.4]);
22
23 //practical sampling
24 T=0.1;
25 p=1;
26 x1=(sinc(5*%pi*t).^2).*((abs(modulo(t,T))<0.0125)|
27     (abs(modulo(t,T))>0.0875));
27 omega=-50*%pi:0.01:50*%pi;
28 p=0;
29 for om=-50*%pi:0.01:50*%pi
30     p=p+1;

```

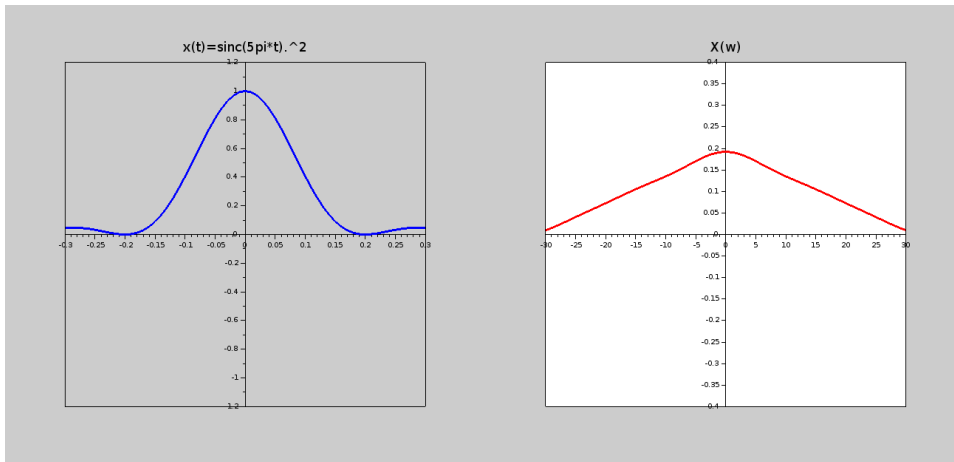


Figure 8.3: Practical sampling

```

31     X1(p)=sum(x1.*exp(-%i*om*t).*0.009);
32 end
33 figure(2)
34 subplot(1,2,1);plot(t,x1,'b');plot(t,x,'r');
35 title("practical_sampling of x(t)","fontsize",4);
36 set(gca(),"x_location","middle","y_location","middle
    ","zoom_box",[-0.3 -1.2 0.3 1.2]);
37 subplot(1,2,2);plot(omega,abs(X1),'r');
38 title("X(w) new","fontsize",4);
39 set(gca(),"y_location","middle","x_location","middle
    ","zoom_box",[omega(1) -0.06 omega($) 0.06]);
40
41 //there will be small error in amplitude

```

Scilab code Exa 8.5 Finding bit rate and sampling rate

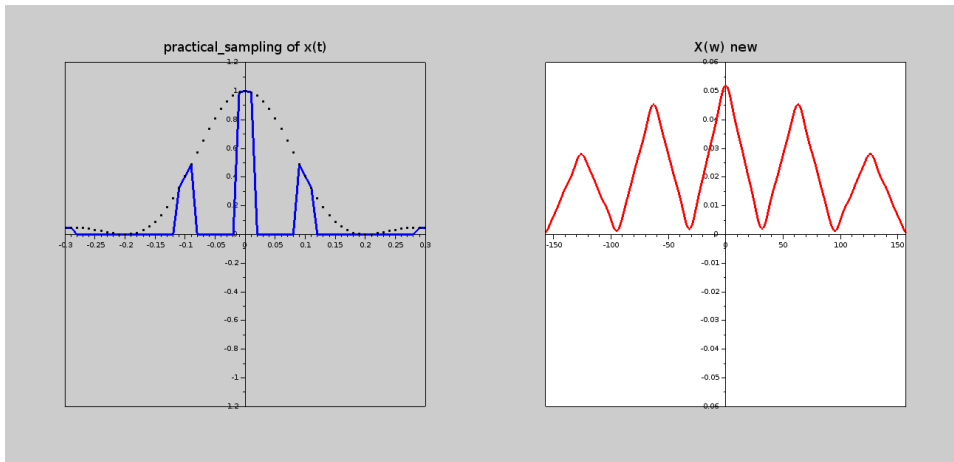


Figure 8.4: Practical sampling

```

1
2 clc
3 clear
4
5 fN=2*3000; //Nyquist sampling rate
6 fA=fN*(4/3); //actual sampling rate
7 L=256; //as it shouldn't be more than 0.05% and
      shouls be power of 2
8 bit=8;
9 bit_rate=bit*fA;
10 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 N_0 for DFT

```

1 clc
2 clear
3
4 f0=100;

```

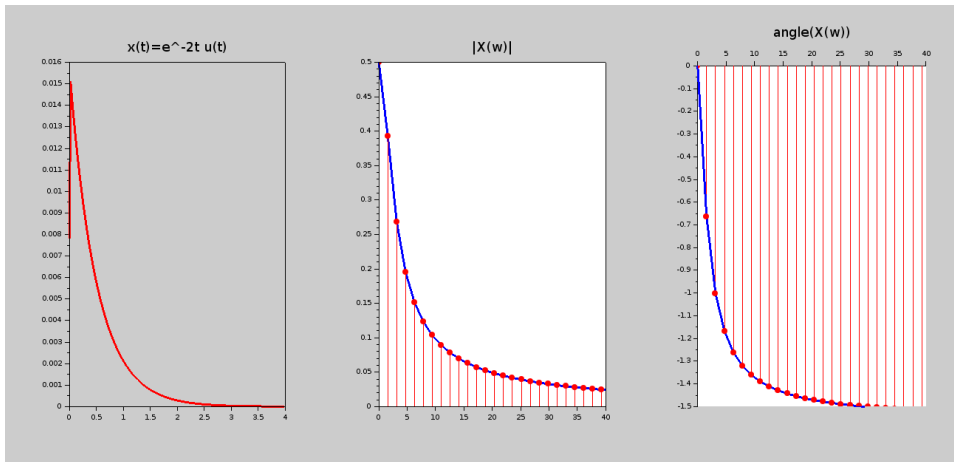


Figure 8.5: Fourier transform of given exponential signal using DFT

```

5 T0=1/f0;
6 B=10000;
7 fs=2*B;
8 T=1/fs;
9 N0=fs/f0;
10 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

```

1 clc
2 clear
3 close;
4
5 N0=256; //samples no
6 T0=4;
7 T=T0/N0;
8 t=(0:T:(N0-1)*T)';

```

```

9
10 x=T*exp(-2*t);
11 x(1)=T*(exp(-2*T0)+1)/2;
12 Xr=fft(x);
13 r=-N0/2:N0/2-1;
14 omega=r*2*%pi/T0;
15 Xr_abs=fftshift(abs(Xr));
16 Xr_phase=fftshift(phasemag(Xr)*%pi/180);
17
18 figure(1)
19 subplot(1,3,1);plot(t,x,'r');
20 title("x(t)=e-2t u(t)", "fontsize",4);
21 subplot(1,3,2);plot(omega,Xr_abs,'b');plot2d3(omega,
    Xr_abs,[5]);
22 title("|X(w)|", "fontsize",4);
23 set(gca(),"zoom_box",[0 0 40 0.5]);
24 subplot(1,3,3);plot2d3(omega,Xr_phase,[5]);plot(
    omega,Xr_phase,'b');
25 title("angle(X(w))", "fontsize",4);
26 set(gca(),"zoom_box",[0 -1.5 40 0],"x_location","top
    ");

```

Scilab code Exa 8.9 Fourier transform of given pulse using DFT

```

1 clc
2 clear
3 close;
4
5 N0=32; //samples no
6 T0=4;
7 T=T0/N0;
8 x=[ones(1,4) 0.5 zeros(1,23) 0.5 ones(1,3)]';
9 Xr=fft(x);

```

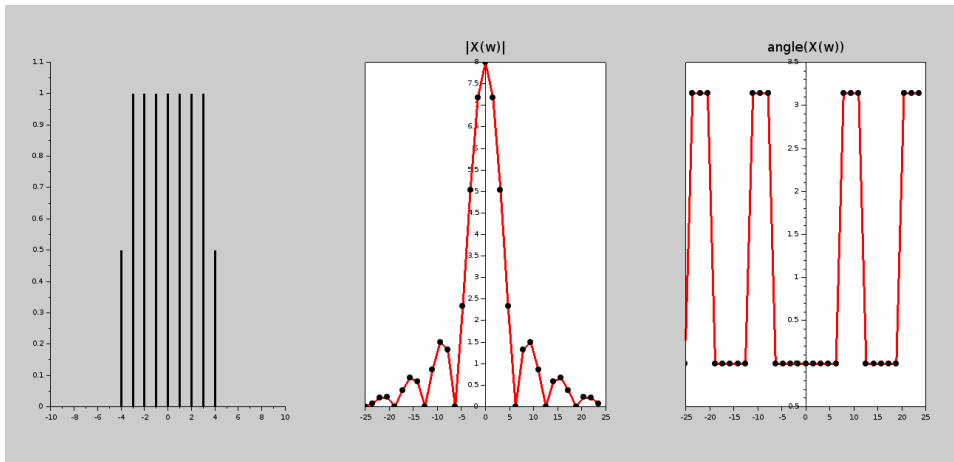


Figure 8.6: Fourier transform of given pulse using DFT

```

10 r=-N0/2:N0/2-1;
11 omega=r*2*%pi/T0;
12 Xr_abs=fftshift(abs(Xr));
13 X_phase=fftshift(phasemag(Xr)*%pi/180);
14
15 figure(1)
16 subplot(1,3,1);plot2d3([-32:63], repmat(x',1,3));
17 set(gca(),"zoom_box",[-10 0 10 1.1]);
18 subplot(1,3,2);plot(omega,Xr_abs,'b');
19 title("|X(w)|","fontsize",4);
20 set(gca(),"y_location","middle","zoom_box",[-25 0 25
    8]);
21 subplot(1,3,3);plot(omega,X_phase,'b');
22 title("angle(X(w))","fontsize",4);
23 set(gca(),"zoom_box",[-25 -0.5 25 3.5],"y_location",
    "middle");

```

Scilab code Exa 8.10 Finding filtered output using DFT

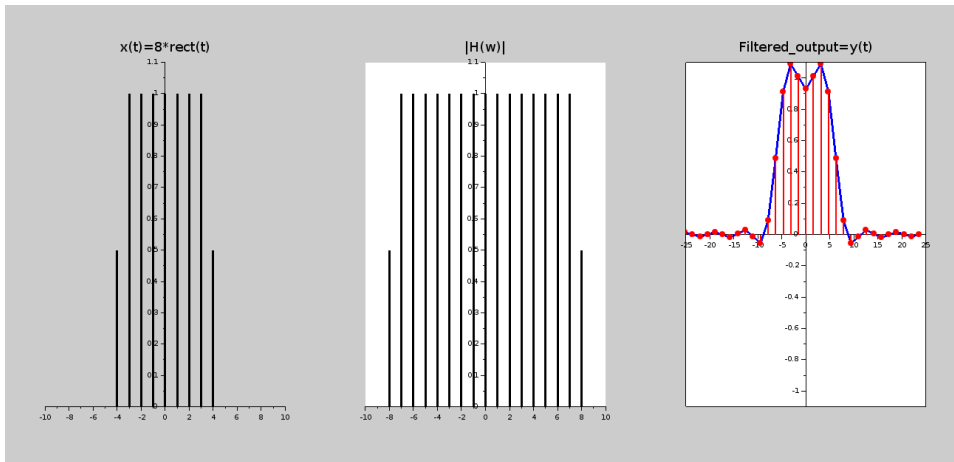


Figure 8.7: Finding filtered output using DFT

```

1  clc
2  clear
3  close;
4
5  N0=32; //samples no
6  T0=4;
7  T=T0/N0;
8  x=[ones(1,4) 0.5 zeros(1,23) 0.5 ones(1,3)]';
9  Xr=fft(x);
10 Hr=[ones(1,8) 0.5 zeros(1,15) 0.5 ones(1,7)]';
11 Yr=Xr.*Hr;
12 y=ifft(Yr);
13 r=-N0/2:N0/2-1;
14 n=r;
15 omega=r*2*%pi/T0;
16 y=fftshift(y);
17 figure(1)
18 subplot(1,3,1);plot2d3([-32:63], repmat(x',1,3));
19 set(gca(),"zoom_box",[-10 0 10 1.1],"y_location",
    middle");
20 title("x(t)=8*rect(t)","fontsize",4);
21 subplot(1,3,2);plot2d3([-32:63], repmat(Hr',1,3));

```

```
22 title("|H(w)|", "fontsize", 4);
23 set(gca(), "y_location", "middle", "zoom_box", [-10 0 10
    1.1]);
24 subplot(1,3,3); plot(omega, y, 'b'); plot2d3(omega, y
    , [5]);
25 title("Filtered_output=y(t)", "fontsize", 4);
26 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

```
1  clc
2  clear
3  close;
4
5  N0=20;
6  n=0:N0-1;
7  x=sin(0.1*%pi*n);
8  xr=fft(x)/N0;
9  xr=round(xr*10000)/10000;
10 n=-4*N0/2:4*N0/2-1;
11
12 subplot(1,3,1);plot2d3(n, repmat(x,1,4));
13 set(gca(),"x_location","middle","y_location","middle
    ");
14 title("x(n)=sin(0.1*pi*n)","fontsize",4);
15 subplot(1,3,2);plot2d3(n, repmat(abs(xr),1,4),[5]);
16 set(gca(),"y_location","middle");
```

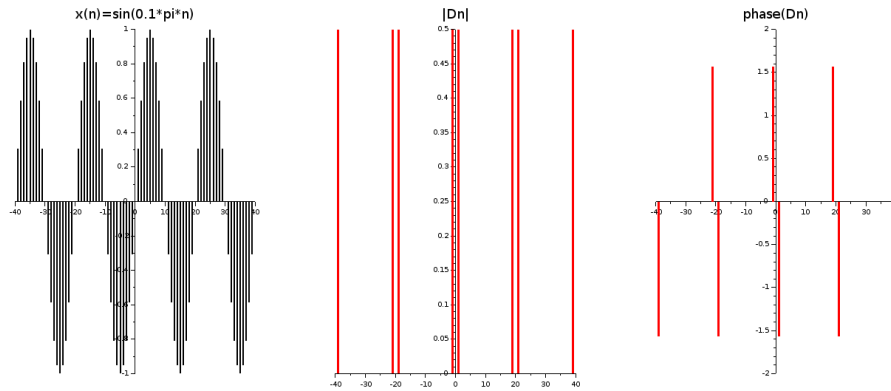


Figure 9.1: Discrete time Fourier series

```

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

```

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

```

1 clc
2 clear
3 close;
4
5 N0=32;
6 n=0:N0-1;
7 x=[ones(1,5) zeros(1,23) ones(1,4)];
8 xr=fft(x)/N0;
9 xr=round(xr*10000)/10000;

```

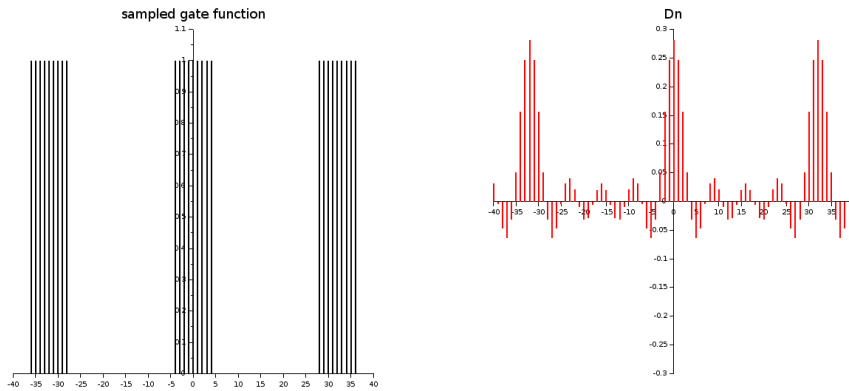



Figure 9.2: Discrete time Fourier series of periodic sampled gate function

```

10 n=-4*N0/2:4*N0/2-1;
11
12 subplot(1,2,1);plot2d3(n, repmat(x,1,4));
13 set(gca(),"y_location","middle","zoom_box",[-40 0 40
14     1.1]);
14 title("sampled gate function","fontsize",4);
15 subplot(1,2,2);plot2d3(n, repmat(xr,1,4),[5]);
16 set(gca(),"x_location","middle","y_location","middle
17     ","zoom_box",[-40 -0.3 40 0.3]);
17 title("Dn","fontsize",4);

```

Scilab code Exa 9.3 Discrete time Fourier transform

```

1 clc
2 clear
3 close;
4
5 N=1024; //samples number

```

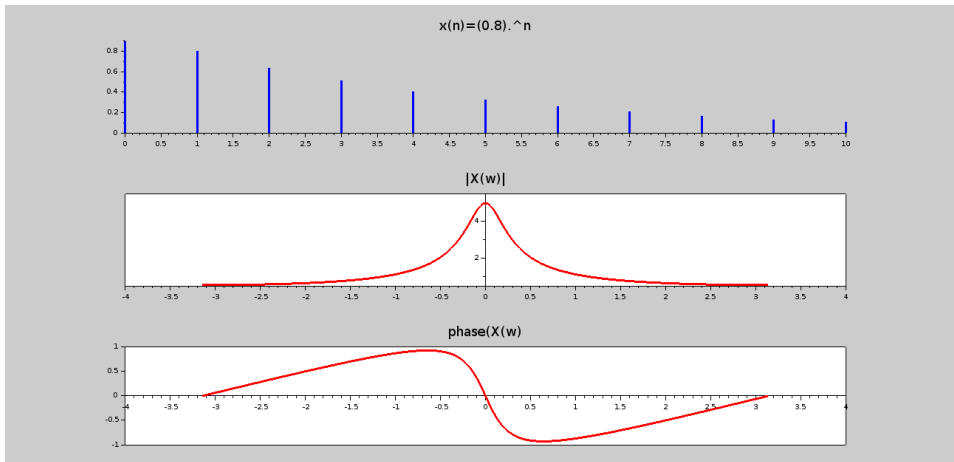


Figure 9.3: Discrete time Fourier transform

```

6 n=0:N-1;
7 x=(0.8).^n;
8 omega=[-N/2:(N/2)-1]*2*pi/N;
9 X=fft(x);
10 X_mode=fftshift(abs(X));
11 X=round(X*10000)/10000;
12 X_angle=fftshift(phasemag(X)*pi/180);
13
14 figure(1)
15 subplot(3,1,1);plot2d3(n,x,[2]);
16 title("x(n)=(0.8).^n","fontsize",4);
17 set(gca(),"zoom_box",[0 0 10 0.9]);
18 subplot(3,1,2);plot(omega,X_mode,'r');
19 set(gca(),"y_location","middle");
20 title("|X(w)|","fontsize",4);
21 subplot(3,1,3);plot(omega,X_angle,'r');
22 set(gca(),"x_location","middle");
23 title("phase(X(w))","fontsize",4);

```

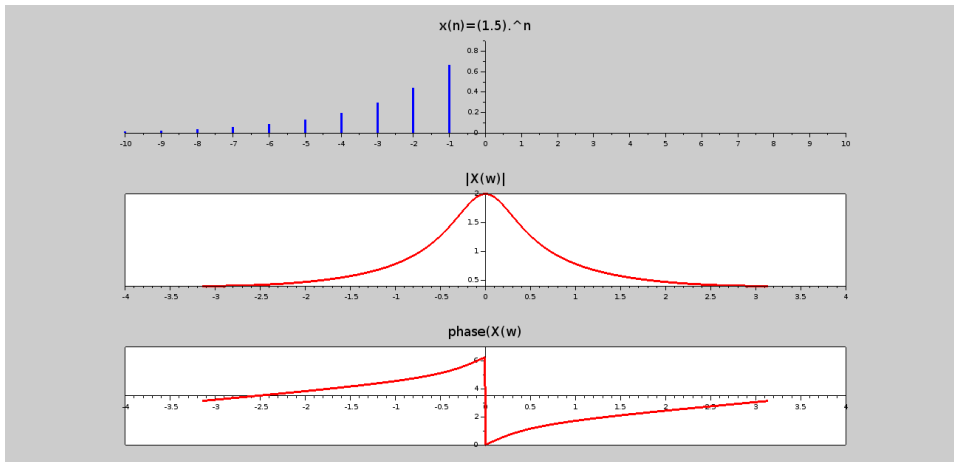


Figure 9.4: Discrete time Fourier transform

Scilab code Exa 9.4 Discrete time Fourier transform

```

1  clc
2  clear
3  close;
4
5  N=1024; //samples number
6  n=-N:-1;
7  x=(1.5).^n;
8  omega=[-N/2:(N/2)-1]*2*%pi/N;
9  X=fft(x);
10 X_mode=fftshift(abs(X));
11 X=round(X*10000)/10000;
12 X_angle=fftshift(phasemag(X)*%pi/180);
13
14 figure(1)
15 subplot(3,1,1);plot2d3(n,x,[2]);
16 title("x(n)=(1.5).^n","fontsize",4);
17 set(gca(),"zoom_box",[-10 0 10 0.9],"y_location",

```

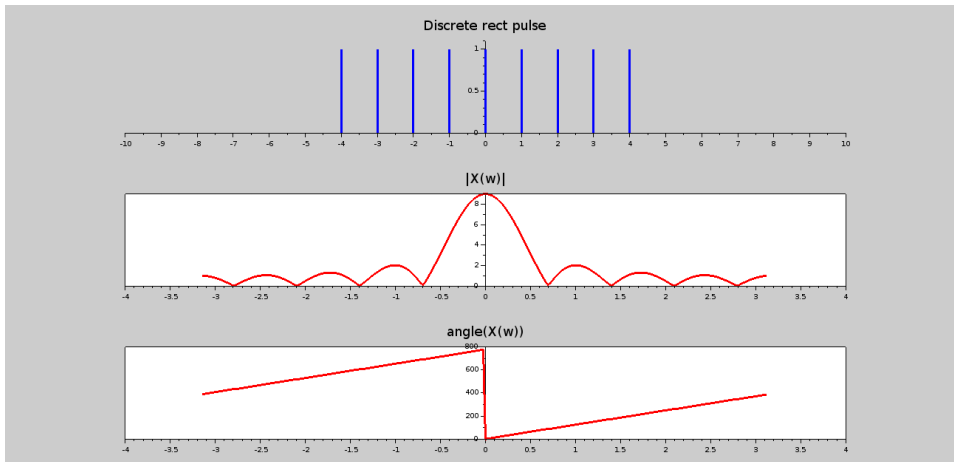


Figure 9.5: DTFT of discrete time rectangular pulse

```

middle");
18 subplot(3,1,2);plot(omega,X_mode,'r');
19 set(gca(),"y_location","middle");
20 title("|X(w)|","fontsize",4);
21 subplot(3,1,3);plot(omega,X_angle,'r');
22 set(gca(),"x_location","middle","y_location","middle");
23 title("phase(X(w))","fontsize",4);

```

Scilab code Exa 9.5 DTFT of discrete time rectangular pulse

```

1 clc
2 clear
3 close;
4
5 N=256; //samples number
6 n=-(N/2):N/2-1;
7 x=1.*((n>=-4)&(n<=4));

```

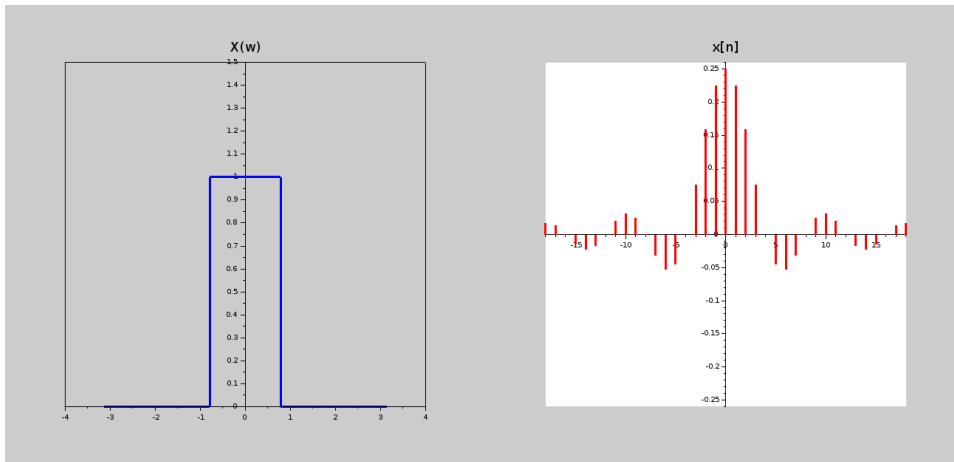


Figure 9.6: DTFT of discrete time rectangular pulse

```

8  omega=[-N/2:(N/2)-1]*2*%pi/N;
9  X=fft(x);
10 X_mode=fftshift(abs(X));
11 X=round(X*10000)/10000;
12 X_angle=fftshift(phasemag(X)*%pi/180);
13
14 figure(1)
15 subplot(3,1,1);plot2d3(n,x,[2]);
16 title("Discrete rect pulse","fontsize",4);
17 set(gca(),"zoom_box",[-10 0 10 1.1],"y_location",
    "middle");
18 subplot(3,1,2);plot(omega,X_mode,'r');
19 set(gca(),"y_location","middle");
20 title("|X(w)|","fontsize",4);
21 subplot(3,1,3);plot(omega,X_angle,'r');
22 set(gca(),"y_location","middle");
23 title("angle(X(w))","fontsize",4)

```

Scilab code Exa 9.6 DTFT of discrete time rectangular pulse

```
1  clc
2  clear
3  close;
4
5
6  function y=X(om)
7      y=1.*(abs(om)<=%pi/4);
8  endfunction
9
10 omega=linspace(-%pi,%pi,1024);
11 domega=omega(2)-omega(1);
12 q=length(omega);
13 n=-80:1:q-81;
14 p=1;
15 for n1=n
16     r(p)=(1/(2*%pi)).*sum(X(omega).*exp(%i*omega*n1)
17         .*domega);
18     p=p+1;
19 end
20 figure(1)
21 subplot(1,2,1);plot(omega,X(omega));
22 title("X(w)","fontsize",4);
23 set(gca(),"y_location","middle","zoom_box",[-4 0 4
24     1.5]);
25 subplot(1,2,2);plot2d3(n,r,[5]);
26 set(gca(),"zoom_box",[-18 -0.26 18 0.26],"x_location
27     ","middle","y_location","middle");
28 title("x[n]","fontsize",4);
```

Scilab code Exa 9.9 DTFT of the given signal

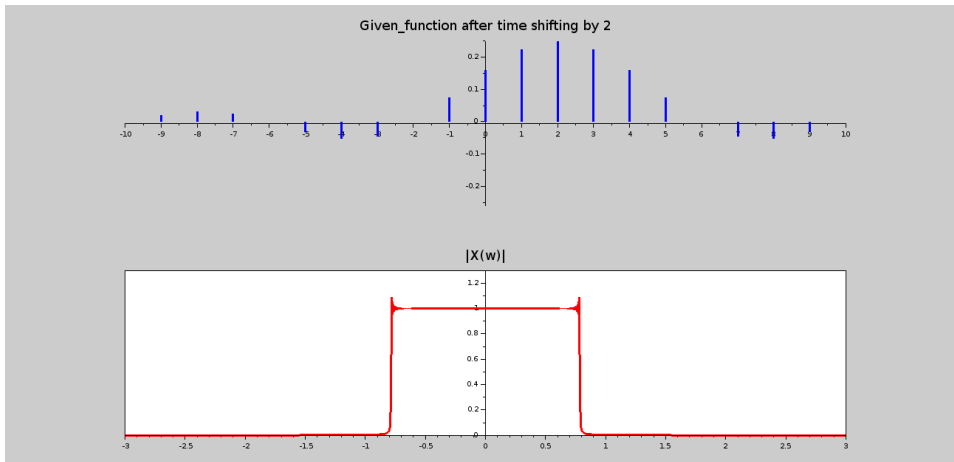


Figure 9.7: DTFT of the given signal

```

1  clc
2  clear
3  close;
4  //error in phase calculation will there
5  N=1024; //samples number
6  n=-(N/2):N/2-1;
7  function y=f(n)
8      y=(1/4)*sinc(%pi*n/4);
9  endfunction
10 x=f(n-2);
11 omega=[-N/2:(N/2)-1]*2*%pi/N;
12 X=fft(x);
13 X_mode=fftshift(abs(X));
14
15 figure(1)
16 subplot(2,1,1);plot2d3(n,x,[2]);
17 title("Given_function after time shifting by 2",
18       "fontsize",4);
19 set(gca(),"zoom_box",[-10 -0.26 10 0.25],"y_location
20       ","middle","x_location","middle");
21 subplot(2,1,2);plot(omega,X_mode,'r');
22 set(gca(),"y_location","middle","zoom_box",[-3 0 3

```

```

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

```

Scilab code Exa 9.10 DTFT of the given modulated signal

```

1  clc
2  clear
3  close;
4
5  // ***** part a (wc=pi/2)
   *****
6  N=1024; //samples number
7  n=-(N/2):N/2-1;
8  x=sinc(%pi*n/4).*cos(%pi*n/2);
9  omega=[-N/2:(N/2)-1]*2*%pi/N;
10 X=fft(x);
11 X_mode=fftshift(abs(X));
12 X=round(X*10000)/10000;
13 X_angle=fftshift(phasemag(X)*%pi/180);
14
15 figure(1)
16 subplot(2,1,1);plot2d3(n,x,[2]);
17 title(" part (a) _modulated signal", " fontsize", 4);
18 set(gca()," zoom_box", [-30 -1.1 30 1.1], " y_location",
   " middle", " x_location", " middle");
19 subplot(2,1,2);plot(omega,X_mode,'r');
20 set(gca()," y_location", " middle");
21 title(" |X(w) |", " fontsize", 4);
22
23 // ***** part b (wc=0.875*pi)
   *****
24 x=sinc(%pi*n/4).*cos(%pi*n*0.875);
25 X=fft(x);
26 X_mode=fftshift(abs(X));
27 X=round(X*10000)/10000;

```

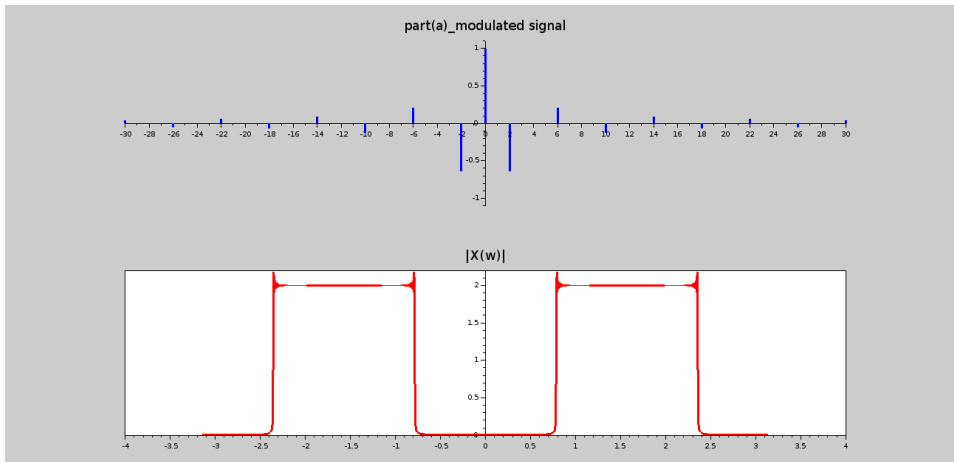



Figure 9.8: DTFT of the given modulated signal

```

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

```

Scilab code Exa 9.13 ZSR of LTID system

```
1 clc
```

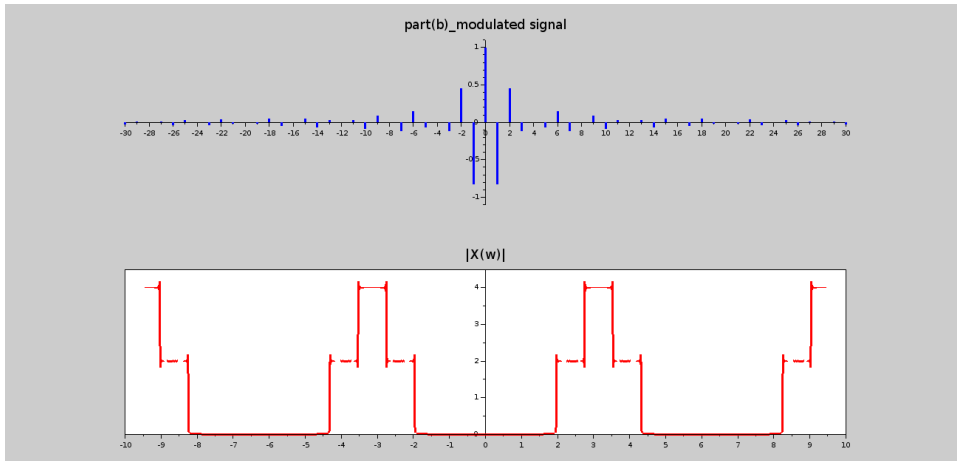


Figure 9.9: DTFT of the given modulated signal

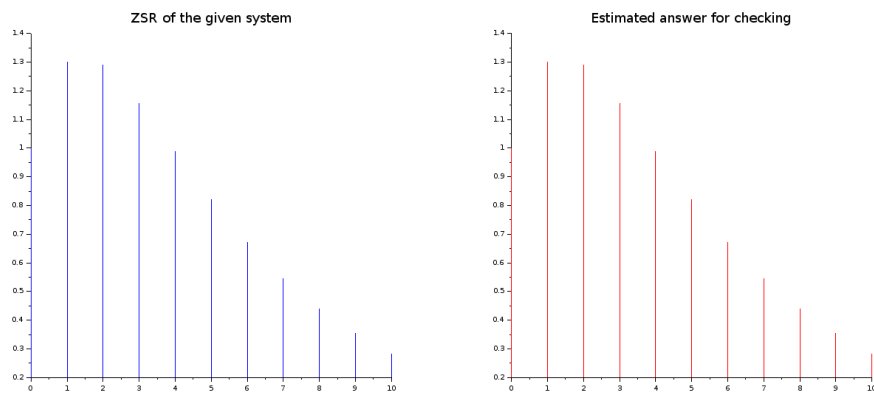


Figure 9.10: ZSR of LTID system

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

Chapter 10

State space analysis

Scilab code Exa 10.4 State space representation from transfer function

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

check Appendix [AP 1](#) for dependency:

inv_laplace.sci

Scilab code Exa 10.5 finding state vector

```
1 clc
```

```

2 //symbolic tool(scilab-scimax) is needed which
   should be installed properly in Ubuntu 14.04 and
   Scilab-5.5.0
3 Syms t s
4
5 //execute the file ilaplace.sci to get inverse
   laplace
6 //path should be correctly written
7 exec('C:\Users\Satyajit\Desktop\sample_codes\
   inv_laplace.sci',-1)
8
9 A=[-12 2/3;-36 -1]; B=[1/3;1]; q0=[2;1]; X=1/s;
10 size(A)
11 size(s*eye(2,2))
12 Q=inv(s*eye(2,2)-A)*(q0+B*X);
13 q=[];
14 q(1)=ilaplace(Q(1));
15 q(2)=ilaplace(Q(2));
16 disp(q*'u(t)', "[q1(t) ; q2(t)]")

```

Scilab code Exa 10.6 transfer function from state space equation

```

1 clc
2
3 s=%s;
4 A=[0 1;-2 -3];
5 B=[1 0;1 1];
6 C=[1 0;1 1;0 2];
7 D=[0 0;1 0; 0 1];
8 H=simp(C*inv(s*eye(2,2)-A)*B+D);
9 disp(H,"the transfer function matrix H(s)=")
10 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

```
1  clc ;
2  clear;
3
4  s=%s;
5  A=[s+12 -2/3;36 s+1];
6  y= roots(det(A))
7  t= poly(0, 't' );
8  beta =inv([1 y(1); 1 y(2)])*[%e^-y(1)*t;%e^-y(2)*t];
9  disp(beta,"*****beta=")
10 W= beta(1)*[1 0;0 1]+ beta(2) *[-12 2/3; -36 -1];
11 zir =W*[2;1];
12 disp(zir,"*****ZIR=");
13 zsr =W*[1/3;1];
14 disp(zsr,"*****ZSR= ");
15 total=zir +zsr;
16 disp(total,"*****total response=");
```

Scilab code Exa 10.8 Determining exponential matrix

```
1  clc
2  //symbolic tool(scilab-scimax) is needed which
   should be installed properly in Ubuntu 14.04 and
   Scilab -5.5.0
3
4  Syms t s;
5  //execute the file ilaplace.sci to get inverse
   laplace
6  //path should be correctly written
7  exec('C:\Users\Satyajit\Desktop\sample_codes\
   ilaplace.sci',-1)
```

```

8
9 F1=ilaplace((s+3)/((s+1)*(s+2)))
10 F2=ilaplace(1/((s+1)*(s+2)))
11 F3=ilaplace(-2/((s+1)*(s+2)))
12 F4=ilaplace(s/((s+1)*(s+2)))
13 F=[F1 F2;F3 F4];
14 disp(F,"f(t)=")
15 A=[1 0;1 1;0 2];
16 B=[0 0;1 0;0 1];
17 h=A*F*[1 0;1 1]+B*eye(2,2); //including delta(t)
    function
18 disp(h,"h(t)=")

```

Scilab code Exa 10.9 Finding new state equations

```

1 clc
2 clear
3
4 A =[0 1; -2 -3];
5 B =[1;2];
6 P=[1 1;1 -1];
7 Ahat =P*A*inv(P);Bhat=P*B;
8 disp(Bhat ," Bhat")
9 disp(Ahat ," Ahat=")

```

Scilab code Exa 10.10 Diagonalized form of the state equations

```

1 clc
2 clear
3
4 format(7)
5 A =[0 1; -2 -3];
6 [V,lambda ]= spec(A);

```

```

7 P=inv(V);
8 B =[1;2];
9 Bhat =P*B;
10 disp(P,"P=")
11 disp(Bhat ,"B^=")
12 disp(lambda," lambda=")

```

Scilab code Exa 10.11 controllability and observability of the system

```

1 clc
2 clear
3 close
4 //*****part(a)*****
5 A =[1 0;1 -1];
6 [V,lambda ]= spec(A);
7 B=[1;0];
8 C=[1 -2];
9 P= inv(V);
10 Bh=P*B
11 Ch=C*inv(P)
12 disp(' part(a): ')
13 disp(Bh ,"B^=")
14 disp(Ch ,"C^=")
15 //*****part(b)*****
16 A=[ -1 0; -2 1];
17 [V, lambda ]= spec (A);
18 B =[1;1];
19 C =[0 1];
20 P= inv (V);
21 Bh =P*B
22 Ch =C*inv(P)
23 disp(' Part(b): ')
24 disp(Bh ,"B^=")
25 disp(Ch ," C^=")

```

Scilab code Exa 10.12 finding output of the system

```
1  clc
2  clear
3  close
4
5  A =[0 1; -1/6 5/6];
6  B =[0;1];
7  C=[ -1 5];
8  D =0;
9  sys=syslin('d',A,B,C,D);
10 N =25;
11 x= ones(1,N +1) ;n =(0: N);
12 q0=[2;3];
13 [ y q]=csim('step',n,sys);
14 y=dsimul(sys ,x);
15 plot2d3(y);
16 title("y[n] ZSR", "fontsize",4);
```

Scilab code Exa 10.13 response of given system using z transform

```
1 //Inverse Z Transform
2 //symbolic tool(scilab-scimax) is needed which
   should be installed properly in Ubuntu 14.04 and
   Scilab-5.5.0
3 //limit function can be used only after installing
   symbolic toolbox
4
5  clc
6  Syms n z;
7  H1 = (-2*z)/(z-(1/3));
8  H2 = (3*z)/(z-0.5);
```

```
9 H3 = (24*z)/(z-1);
10 F1 = H1*z^(n-1)*(z-(1/3));
11 F2 = H2*z^(n-1)*(z-0.5);
12 F3 = H3*z^(n-1)*(z-1);
13 h1 = limit(F1,z,(1/3)); //finding symbolic limits
14 disp(h1, 'h1[n]= ')
15 h2 = limit(F2,z,0.5); //finding symbolic limits
16 disp(h2, 'h2[n]= ')
17 h3 = limit(F3,z,1); //finding symbolic limits
18 disp(h3, 'h3[n]= ')
19 h = h1+h2+h3;
20 disp(h, 'h[n]= ')
```

Appendix

Scilab code AP 1 Inverse laplace transform

```
1 function [sym1] = ilaplace(A,t,s)
2
3 // ilaplace
4 //


---


5 // PURPOSE
6 // Inverse laplace transform.
7 //
8 // SYNOPSIS
9 // B = ilaplace(A[,s[,t]])
10 //
11 // INPUT ARGUMENTS
12 // A      Symbolic expression.
13 // t      Time variable ('t' by default)
14 // s      Laplace variable ('s' by default)
15 //
16 // OUTPUT ARGUMENT
17 // B      Symbolic expression.
18 //
19 // See also: laplace
20 //


---


21 // // EXAMPLE
22 // syms t p
```

```

23 //      ilaplace('exp(-2*p)/(1+p^2)',p,t) // Maxima
        ignores Heaviside!
24 //      ilaplace('1/(p*(1+p^2))',p,t)
25 //      laplace(ilaplace(1/p^2,p,t),t,p)
26
27 // Author: : J.F. Magni
28 // 18-04-2006
29
30     [%nargout,%nargin] = argn(0);
31     if %nargin ~= 1 & %nargin ~= 2 & %nargin ~= 3;
        error('One to three arguments'); end;
32
33     if %nargin == 1; t = 's'; s = 't'; end;
34     if %nargin == 2; s = 't'; end;
35
36     A = sym(A);
37     t = sym(t);
38     s = sym(s);
39     str1 = 'ilt('+A.str1+', '+t.str1+', '+s.str1+')'
40
41     sym1 = str2max2sym(str1);
42
43 endfunction

```

Scilab code AP 2 Finding laplace transform

```

1 clc
2 clear
3
4 function [sym1] = laplace(A,t,s)
5
6 // laplace
7 //

```

```

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

```

```

10 // exp, log, sin, cos, sinh, cosh, erf and ilt (
    name of the Maxima
11 // inverse Laplace transform).
12 //
13 // SYNOPSIS
14 // B = laplace(A[,t[,s]])
15 //
16 // INPUT ARGUMENTS
17 // A      Symbolic expression.
18 // t      Time variable ('t' by default)
19 // s      Laplace variable ('s' by default)
20 //
21 // OUTPUT ARGUMENT
22 // B      Symbolic expression.
23 //
24 // See also: ilaplace, sym/integ
25 //
    #-----

26 // // EXAMPLE
27 //     syms t s
28 //     laplace('3*delta(t-2)+cos(t)*sin(t)',t,s)
29 //     laplace('3*delta(t-2)+cos(t)*sin(t)',t,'p')
30 //     laplace(diff(cos(t)^3,t),t,s)
31
32 // Author: : J.F. Magni
33 // 18-04-2006
34
35     [%nargout,%nargin] = argn(0);
36     if %nargin ~= 1 & %nargin ~= 2 & %nargin ~= 3;
37         error('One to three arguments'); end;
38
39     if %nargin == 1; t = 't'; s = 's'; end;
40     if %nargin == 2; s = 's'; end;
41
42     A = sym(A);
43     t = sym(t);
44     s = sym(s);

```

```

44     str1 = 'laplace('+A.str1+', '+t.str1+', '+s.str1+'
         )',
45
46     sym1 = str2max2sym(str1);
47
48 endfunction

```

Scilab code AP 3 Graphical_convolution

```

1 //this is the existing dependency file to be used
  for graphical convolution
2 //
  *****
3 //
  *****
4
5 //functions should be declared like this and the
  function should be called after the execution of
  function definition file
6 //t = '[-5:0.02:5]';
7 //f1='u(t)'; //defining x(t)
8 //f2='-2*exp(2*t).*u(-t)+2*exp(-t).*u(t)'; //
  defining h(t)
9 //conv_gui(f1,f2,t);(should be executed at the last
  part)
10
11 funcprot(0);
12
13 // Defining event handling for plot window
14
15 function conv_gui_event(win,x,y,ibut)
16     if ibut==37
17         conv_gui('move_left');
18     elseif ibut==39
19         conv_gui('move_right');
20     end

```

```

21 endfunction
22
23
24 function []=conv_gui(varargin)
25
26     // Defining unit step function
27     function y=u(x)
28         y=sign((sign(x)+1))
29     endfunction
30
31     action='init_gui';
32
33
34     if argn(2) ==1 then
35         action=varargin(1);
36     elseif argn(2) ==2 then
37         if varargin(1)=='load_sample' then
38             action="load_sample";
39             n_sample= varargin(2);
40         end
41     end
42
43     if action == 'init_gui' //
44         *****
45         h=findobj('tag','conv_gui_main_window');
46         if h == [] then
47             // Drawing the figure
48             h=figure('Tag','conv_gui_main_window','
49                 Figure_name','Convolution of two
50                 function');
51             h.event_handler="conv_gui_event";
52             h.event_handler_enable="on";
53
54             // creating axes
55             subplot(3,1,1);
56             ax1= gca();
57             ax1.tight_limits='on';
58             ax1.auto_clear='on';

```

```

56     subplot(3,1,2);
57     ax2= gca();
58     ax2.tight_limits='on';
59     ax2.auto_clear='on';
60     subplot(3,1,3);
61     ax3= gca();
62     ax3.tight_limits='on';
63     ax3.auto_clear='on';
64
65
66
67     // Deleting menus
68     delmenu(h.figure_id, gettext("&File"));
69     delmenu(h.figure_id, gettext("&Edit"));
70     delmenu(h.figure_id, gettext("&Tools"));
71     delmenu(h.figure_id, gettext("&?"));
72
73     // Hiding toolbar
74     toolbar(h, 'off');
75
76     // adding menus
77
78     mnu_options = uimenu(h, ...
79     'Label', 'Options');
80     mnu_options_set_fun = uimenu(mnu_options
81     , ...
82     'Label', 'Set Functions',...
83     'Callback', 'conv_gui(''set_functions'')
84     ');
85     mnu_options_illustrate=uimenu(
86     mnu_options, ...
87     'checked', 'off',...
88     'Label', 'Illustrate',...
89     'Callback', 'conv_gui(''illustrate'')');
90
91     mnu_samples = uimenu(h, ...
92     'Label', 'Samples');
93     mnu_samples_1 = uimenu(mnu_samples, ...

```



```

91         'Label', 'Sample 1', ...
92         'Callback', 'conv_gui(''load_sample'',1)
          ');
93     mnu_samples_2 = uimenu(mnu_samples, ...
94         'Label', 'Sample 2', ...
95         'Callback', 'conv_gui(''load_sample'',2)
          ');
96     mnu_samples_1 = uimenu(mnu_samples, ...
97         'Label', 'Sample 3', ...
98         'Callback', 'conv_gui(''load_sample'',3)
          ');
99
100    mnu_move_left = uimenu(h, ...
101        'Label', 'Move Left <--', ...
102        'Enable', 'off', ...
103        'Callback', 'conv_gui(''move_left'')');
104    mnu_move_right = uimenu(h, ...
105        'Label', ' --> Move Right', ...
106        'Enable', 'off', ...
107        'Callback', 'conv_gui(''move_right'')');
108    mnu_about = uimenu(h, ...
109        'Label', 'About', ...
110        'Callback', 'conv_gui(''about'')');
111
112
113
114    mnu=struct('illustrate',
              mnu_options_illustrate, 'left',
              mnu_move_left, 'right', mnu_move_right)
              ;
115
116    // Storing Data
117    move_step=0.1;
118    t_shift=0;
119    t_data=[];
120    t_min=[];
121    t_max=[];
122    t_step=[];

```

```

123         f1_data=[];
124         f2_data=[];
125         conv_data=[];
126         init_plot=1;
127         user_data = struct('t',t,'f1',f1,'f2',f2
            , 'move_step',move_step, ...
128         'ax1',[ax1],'ax2',[ax2],'ax3',[ax3],...
129         't_shift',[t_shift],'t_data',[t_data],'
            f1_data',[f1_data],...
130         'f2_data',[f2_data],'conv_data',[
            conv_data],'init_plot',init_plot,...
131         't_min',t_min,'t_max',t_max,'t_step',
            t_step,'mnu',mnu);
132
133         h userdata = user_data
134
135         // evaluating functions
136         conv_gui('eval_functions');
137
138     else
139         show_window(h);
140     end
141
142 elseif action == 'about' //
143     *****
144     messagebox(["Author: Mahmoud A. AlNaanah"
145     ...
146     "email: malnaanah@gmail.com" ...
147     "Last update: 10.08.2015 (DD.MM.YYYY)" ...
148     "License: GPL" ...
149     "tested on scilab v 5.5.0" ...
150     ], "About", "info");
151
152 elseif action == 'illustrate' //
153     *****
154     h=findobj('tag','conv_gui_main_window');
155
156     if (h userdata.mnu.illustrate.checked=='off'

```

```

    ) then
154     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";
155     h.userdata.mnu.illustrate.checked='on';
156     h.userdata.mnu.left.enable='on';
157     h.userdata.mnu.right.enable='on';
158     h.userdata.ax1.children.children(2).
        visible='on';
159     h.userdata.ax1.title.text=" First
        function (F1) (Red line), F1 x F2 (
        Blue line) "
160     conv_gui('plot')
161 else
162     conv_gui('eval_functions')
163 end
164
165 elseif action == 'load_sample' //
    *****
166     h=findobj('tag','conv_gui_main_window');
167
168     if n_sample == 1 then
169         h.userdata.t='[-5:0.02:5]';
170         h.userdata.f1='2*(u(t)-u(t-1))';
171         h.userdata.f2='(u(t)-u(t-2))';
172         conv_gui('eval_functions')
173     elseif n_sample == 2 then
174         h.userdata.t='[-5:0.02:5]';
175         h.userdata.f1='t .* (u(t)-u(t-1))';
176         h.userdata.f2='t .* (u(t)-u(t-1))';
177         conv_gui('eval_functions')
178     elseif n_sample == 3 then
179         h.userdata.t='[-5:0.02:5]';
180         h.userdata.f1='t .* (u(t+1)-u(t-1))';
181         h.userdata.f2='t .* (u(t+1)-u(t-1))';
182         conv_gui('eval_functions')
183     end

```

```

184
185     elseif action == 'move_right' //
186         *****
187         h=findobj('tag','conv_gui_main_window');
188         if h userdata.mnu.illustrate.checked=='on'
189             then
190                 h userdata.t_shift=h userdata.t_shift+h.
191                 userdata.move_step;
192                 if h userdata.t_shift > (h userdata.
193                 t_max -h userdata.t_min) then
194                     h userdata.t_shift=h userdata.
195                     t_shift-h userdata.move_step;
196                 else
197                     conv_gui('plot')
198                 end
199             end
200         end
201
202     elseif action == 'move_left' //
203         *****
204         h=findobj('tag','conv_gui_main_window');
205
206         if h userdata.mnu.illustrate.checked=='on'
207             then
208                 h userdata.t_shift=h userdata.t_shift-h.
209                 userdata.move_step;
210                 if h userdata.t_shift < (h userdata.
211                 t_min -h userdata.t_max) then
212                     h userdata.t_shift=h userdata.
213                     t_shift+h userdata.move_step;
214                 else
215                     conv_gui('plot')
216                 end
217             end
218         end
219
220     elseif action == 'set_functions' //
221         *****
222         h=findobj('tag','conv_gui_main_window');

```

```

211
212     labels = ["t=";" f1=";" f2=";" "Move step="];
213     vals = [h.userdata.t;h.userdata.f1;h.
214             userdata.f2;string(h.userdata.move_step)]
215     lst = list("str",1,"str",1,"str",1,"str",1);
216     [ok,t,f1,f2,move_step]=getvalue("Use t as
217         variable. u(t) is the unit step function.
218         Use .* instead of * for multiplication"
219         , labels,lst,vals);
220
221     if ok
222         h.userdata.t = t;
223         h.userdata.f1 = f1;
224         h.userdata.f2 = f2;
225         h.userdata.move_step = evstr(move_step);
226         conv_gui('eval_functions');
227     end
228
229     elseif action == 'eval_functions' //
230         *****
231         h=findobj('tag','conv_gui_main_window');
232         h.info_message="Go to Options > Illustrate ,
233             to see graphical illustration of
234             convolution";
235         t=evstr(h.userdata.t);
236         f1=evstr(h.userdata.f1);
237         f2=evstr(h.userdata.f2);
238         mov_step=h.userdata.move_step;
239
240         h.userdata.t_min=min(t);
241         h.userdata.t_max=max(t);
242         h.userdata.t_step=t(2)-t(1);
243
244         conv=convol(f1,f2)*h.userdata.t_step;

```

```

242     SZ = size(t,2);
243     STRT = round(abs(h.userdata.t_min/h.userdata
        .t_step));
244     END = STRT+SZ-1;
245     conv=conv(STRT:END);
246
247
248     h.userdata.t_data=[];
249     h.userdata.f1_data=[];
250     h.userdata.f2_data=[];
251     h.userdata.conv_data=[];
252
253     h.userdata.t_data=t;
254     h.userdata.f1_data=f1;
255     h.userdata.f2_data=f2;
256     h.userdata.conv_data=conv;
257     h.userdata.init_plot=1;
258     h.userdata.t_shift=0;
259
260
261     h.userdata.mnu.illustrate.checked='off'
262     h.userdata.mnu.left.enable='off';
263     h.userdata.mnu.right.enable='off'
264     h.userdata.t_shift=0;
265
266     conv_gui('plot');
267
268     h.userdata.ax1.children.children(2).visible=
        'off';
269
270     elseif action == 'plot' //
        *****
271
272     h=findobj('tag','conv_gui_main_window');
273     t=h.userdata.t_data;
274     f1_data=h.userdata.f1_data;
275     f2_data=h.userdata.f2_data;
276     conv_data=h.userdata.conv_data;

```

```

277
278     if h.userdata.init_plot then
279         sca(h.userdata.ax1)
280         plot(t,f1_data,t,f1_data);
281
282         data_bounds=h.userdata.ax1.data_bounds;
283         data_bounds=data_bounds .*[1,1.1;1,1.1];
284         h.userdata.ax1.title.text="
                input_function(x(t))";
285         h.userdata.ax1.title.font_size=4;
286         h.userdata.ax1.data_bounds=data_bounds;
287         h.userdata.ax1.children.children(1).
                polyline_style=1;
288         h.userdata.ax1.children.children(1).
                thickness=2;
289         h.userdata.ax1.children.children(1).
                foreground=2;
290
291         h.userdata.ax1.children.children(2).
                polyline_style=1;
292         h.userdata.ax1.children.children(2).
                line_style=8;
293         h.userdata.ax1.children.children(2).
                thickness=2;
294         h.userdata.ax1.children.children(2).
                foreground=5;
295         h.userdata.ax1.children.children(2).
                visible=' off ';
296
297
298         sca(h.userdata.ax2)
299         plot(t,f2_data);
300         h.userdata.ax2.title.text="
                impulse_function(h(t))";
301         h.userdata.ax2.title.font_size=4;
302         data_bounds=h.userdata.ax2.data_bounds;
303         data_bounds=data_bounds .*[1,1.1;1,1.1];
304         h.userdata.ax2.data_bounds=data_bounds;

```

```

305     h userdata .ax2 .children .children .
        polyline_style=1;
306     h userdata .ax2 .children .children .
        thickness=2;
307     h userdata .ax2 .children .children .
        foreground=3;
308
309     sca(h userdata .ax3)
310     plot(t, conv_data);
311     h userdata .ax3 .title .text="
        Convoluted_output(y(t))";
312     h userdata .ax3 .title .font_size=4;
313     data_bounds=h userdata .ax3 .data_bounds;
314     data_bounds=data_bounds .*[1,1.1;1,1.1];
315     h userdata .ax3 .data_bounds=data_bounds;
316     h userdata .ax3 .children .children .
        polyline_style=1;
317     h userdata .ax3 .children .children .
        thickness=2;
318     h userdata .ax3 .children .children .
        foreground=4;
319     h userdata .init_plot=0;
320     else
321         // evaluate f2 *****
322         f2_data=flipdim(f2_data,2);
323
324         // trimming f2
325         N=round((h userdata .t_max+h userdata .
            t_min-h userdata .t_shift)/h userdata .
            t_step);
326         SZ = size(h userdata .t_data,2);
327
328         if N >0 then
329             f2_data=[f2_data(N+1:$),zeros(1,N)];
330         elseif N<0
331             f2_data=[zeros(1,abs(N)),f2_data(1:$
                -abs(N))];
332     end

```



```

333
334         f2_data(1)=0;
335         f2_data($)=0;
336
337         h.userdata.ax2.children.children.data=[h
           .userdata.t_data',f2_data'];
338
339         // updating convolution plot
340         N_conv=round((h.userdata.t_shift-h.
           userdata.t_min)/h.userdata.t_step);
341
342         if ((N_conv <= SZ ) & (N_conv >= 1) )
343             conv_data(N_conv:$)=0;
344         elseif N_conv < 1 then
345             conv_data(1:$)=0;
346         end
347
348         h.userdata.ax3.children.children.data=[h
           .userdata.t_data',conv_data'];
349
350         f1_data=f1_data .* f2_data;
351         h.userdata.ax1.children.children(1).data
           =[h.userdata.t_data',f1_data'];
352     end
353
354     else
355         conv_gui();
356
357
358     end
359
360 endfunction
361 //
           *****
362 //
           *****

```

