

Scilab Manual for
Simulation Lab
by Dr Piratla Srihari
Electronics and Telecommunication
Engineering
Geethanjali College Of Engineering And
Technology¹

Solutions provided by
Dr Piratla Srihari
Electronics and Telecommunication Engineering
Geethanjali College Of Engineering And Technology

April 4, 2026

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

Contents

List of Scilab Solutions	3
1 Verification of Gibb's Phenomenon	6
2 Verification of sampling theorem	11
3 Wave Form Synthesis	22
4 Location of Poles and Zeros of a given Transfer function in S-plane and Z-plane	28
5 Removal of Noise from the combination of signal and noise using Auto/Cross correlation	33
6 Verification of Physical realizability and Stability of a given LTI system	40
7 Plotting the CDF and pdf of a Random Variable	43
8 Computation of Moments of a Random variable	46
9 Verification of Central Limit Theorem	49
10 Checking the given random Process for Stationary	56
11 Verification of Weiner-Khynchine Relation	60
12 Simulation of Gaussian Random Vectors	64

List of Experiments

Solution 1.0	Gibbs Phenominon	6
Solution 2.0	Instantaneous Sampling	11
Solution 2.1	Natural sampling	13
Solution 2.2	Flat Top sampling	19
Solution 3.0	Staircase waveform	22
Solution 3.1	Triangular Pulse	24
Solution 4.0	SPlane	28
Solution 4.1	Zplane	29
Solution 5.0	Noise Removal for sequence	33
Solution 5.1	Noise Removal for signal	35
Solution 6.0	Causality and Stability	40
Solution 7.0	CDF and pdf	43
Solution 8.0	Moments Discrete	46
Solution 8.1	Moments Continuous	47
Solution 9.0	Central Identical	49
Solution 9.1	Central Non Identical	51
Solution 10.0	Stationarity	56
Solution 11.0	Wiener Khinchine Theorem	60
Solution 12.0	Gaussian	64

List of Figures

1.1	Gibbs Phenominon	7
1.2	Gibbs Phenominon	8
1.3	Gibbs Phenominon	10
1.4	Gibbs Phenominon	10
2.1	Instantaneous Sampling	13
2.2	Instantaneous Sampling	14
2.3	Instantaneous Sampling	14
2.4	Instantaneous Sampling	15
2.5	Natural sampling	17
2.6	Natural sampling	18
2.7	Flat Top sampling	21
2.8	Flat Top sampling	21
3.1	Staircase waveform	24
3.2	Staircase waveform	24
3.3	Triangular Pulse	26
3.4	Triangular Pulse	27
4.1	SPlane	30
4.2	Zplane	32
5.1	Noise Removal for sequence	35
5.2	Noise Removal for signal	37
5.3	Noise Removal for signal	38
5.4	Noise Removal for signal	38
5.5	Noise Removal for signal	39
6.1	Causality and Stability	42
6.2	Causality and Stability	42

7.1	CDF and pdf	45
7.2	CDF and pdf	45
8.1	Moments Discrete	47
8.2	Moments Continuous	48
9.1	Central Identical	51
9.2	Central Identical	52
9.3	Central Non Identical	54
9.4	Central Non Identical	55
10.1	Stationarity	58
10.2	Stationarity	59
10.3	Stationarity	59
11.1	Wiener Khinchine Theorem	62
11.2	Wiener Khinchine Theorem	63
12.1	Gaussian	67
12.2	Gaussian	67

Experiment: 1

Verification of Gibb's Phenomenon

Scilab code Solution 1.0 Gibbs Phenominon

```
1 //Verification of Gibb's Phenomenon
2 // Approximation of symmetric rectangular pulse
   defined as  $f(t) = 1$  for  $0 < t < \pi$ ;  $-1$  for  $\pi < t < 2\pi$ 
   using a sum of sinusoids
3 // $f(t) = \sin t + (1/3)\sin 3t + (1/5)\sin 5t + \dots$ 
4 //Windows 10
5 //Scilab 6.1.0
6 clear
7 clc
8 fs=input('Enter the sampling frequency:')
9 T=input('Enter the duration over which the f(t) is
   to be plotted:')
10 t=0:T/fs:T;
11 p=zeros(1,length(t));
12 q=p;
```

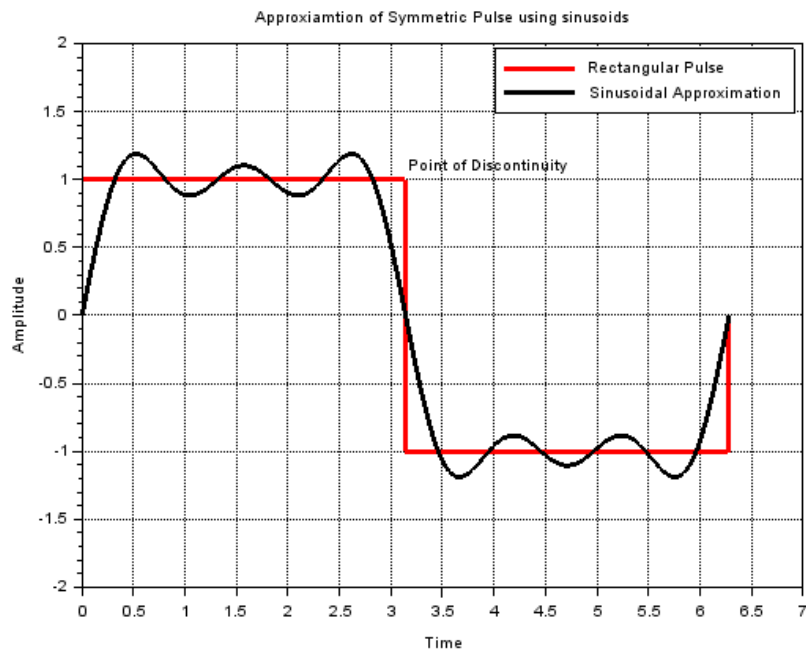


Figure 1.1: Gibbs Phenominon

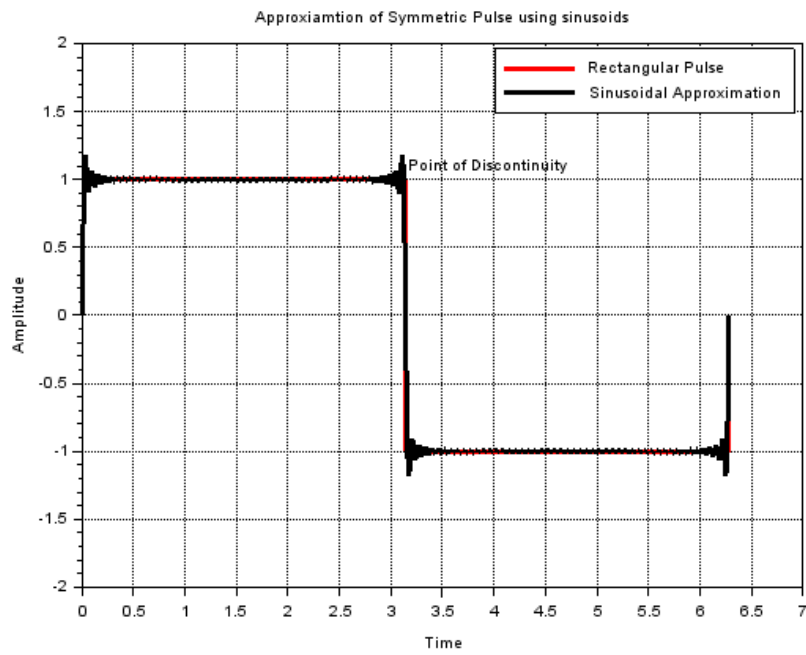


Figure 1.2: Gibbs Phenominon

```

13 n=input('Enter the number of sinusoids:')
14 // This loop generates the symmetric rectangular
    pulse
15 for i=1:floor(length(t)/2)
16     p(i)=1;
17     p(i+floor((length(t)/2)))=-1;
18 end
19 //This loop generates the approximation of the
    symmetric rectangular pulse
20 //using a set of mutually orthogonal sinusoidal
    functions
21 for i=0:n-1
22     k=1/(2*i+1);
23 for j=1:length(t)
24     q(j)=(q(j)+(4/%pi)*k*sin((1/k)*t(j)));
25 end
26 end
27 plot(t,p,'r',t,q,'k','linewidth',3)
28 xgrid
29 mtlb_axis([0 max(t) min(p)-1 max(p)+1])
30 xtitle("Approxiamtion of Symmetric Pulse using
    sinusoids","Time","Amplitude")
31 xstring(t(floor(length(t)/2)),p(floor(length(t)/2))
    ,[" Point of Discontinuity"])
32 legend(["Rectangular Pulse","Sinusoidal
    Approximation"])
33
34
35 //output test case
36 // sampling frequency:1000
37 // duration over which the f(t) is to be plotted:2*
    %pi
38 // number of sinusoids:3
39
40 //output test case
41 // sampling frequency:1000
42 // duration over which the f(t) is to be plotted:2*
    %pi

```

```
Scilab 6.1.0 Console ? ↗ ✕  
  
Enter the sampling frequency:1000  
  
Enter the duration over which the f(t) is to be plotted:2*%pi  
  
Enter the number of sinusoids:3
```

Figure 1.3: Gibbs Phenominon

```
Scilab 6.1.0 Console ? ↗ ✕  
  
Enter the sampling frequency:1000  
  
Enter the duration over which the f(t) is to be plotted:2*%pi  
  
Enter the number of sinusoids:50
```

Figure 1.4: Gibbs Phenominon

43 // number of sinusoids:50

Experiment: 2

Verification of sampling theorem

Scilab code Solution 2.0 Instantaneous Sampling

```
1 //Verification of sampling Theorem
2 //This program verifies Sampling Theorem for  $\sin(20.$ 
    $\pi.t)$  under instantaneous sampling
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 t1=input('Enter the lower limit of time axis:')
8 t2=input('Enter the upper limit of time axis:')
9 s=input('Enter the spacing between the adjacent
   value of time axis:')
10 f=input('Enter the baseband signal frequency:')
11 t=t1:s:t2;
12 x=sin(2*%pi*f*t);
13 s1=zeros(1,length(t));
14 n=input('Enter the integer which decides the
   sampling frequency:')
15 //Generation of sampling signal
16 for i=1:length(t)
```

```

17     if n*i<=length(t)
18         s1(n*i)=1;
19     end
20 end
21 //Generation of Sampled Signal
22 s11=s1.*x;
23 //Reconstruction Filter
24 RC=1/(2*%pi*f);
25 h=(1/RC)*exp(-t/RC);
26 //Signal reconstruction
27 y=conv(h,conv(h,s11));
28 subplot(4,1,1)
29 plot(t,x,'linewidth',2)
30 xgrid
31 xtitle("Baseband signal of frequency 10Hz","Time","
    Amplitude")
32 legend("Signal to be sampled",3)
33 subplot(4,1,2)
34 xset("thickness",2)
35 plot2d3(t,s1,style=-2)
36 xtitle("Sampling Signal")
37 subplot(4,1,3)
38 xset("thickness",2)
39 plot2d3(t,s11,style=-2)
40 xtitle("Sampled Signal")
41 subplot(4,1,4)
42 plot(t,y(1:length(t))/length(y),'linewidth',2)
43 xtitle("Signal at the output of the reconstruction
    Filter","Time","Amplitude")
44 legend("Recovered Signal",3)
45
46
47 //output Test case
48 //lower limit of time axis: 0
49 //upper limit of time axis: 0.2
50 //spacing between the adjacent value of time axis:
    0.001
51 //baseband signal frequency: 10

```

```
Scilab 6.1.0 Console ? ? x
Enter the lower limit of time axis:0
Enter the upper limit of time axis:0.2
Enter the spacing between the adjacent value of time axis:0.001
Enter the baseband signal frequency:10
Enter the integer which decides the sampling frequency:10
```

Figure 2.1: Instantaneous Sampling

```
52 //integer which decides the sampling frequency: 10
53
54 //output Test case
55 //lower limit of time axis: 0
56 //upper limit of time axis: 0.2
57 //spacing between the adjacent value of time axis:
    0.001
58 //baseband signal frequency: 10
59 //integer which decides the sampling frequency: 5
```

Scilab code Solution 2.1 Natural sampling

```
1 //This program generates the naturally sampled
    version of  $\sin(20.\pi.t)$  and simulates its
```

```

Scilab 6.1.0 Console
Enter the lower limit of time axis:0
Enter the upper limit of time axis:0.2
Enter the spacing between the adjacent value of time axis:0.001
Enter the baseband signal frequency:10
Enter the integer which decides the sampling frequency:5

```

Figure 2.2: Instantaneous Sampling

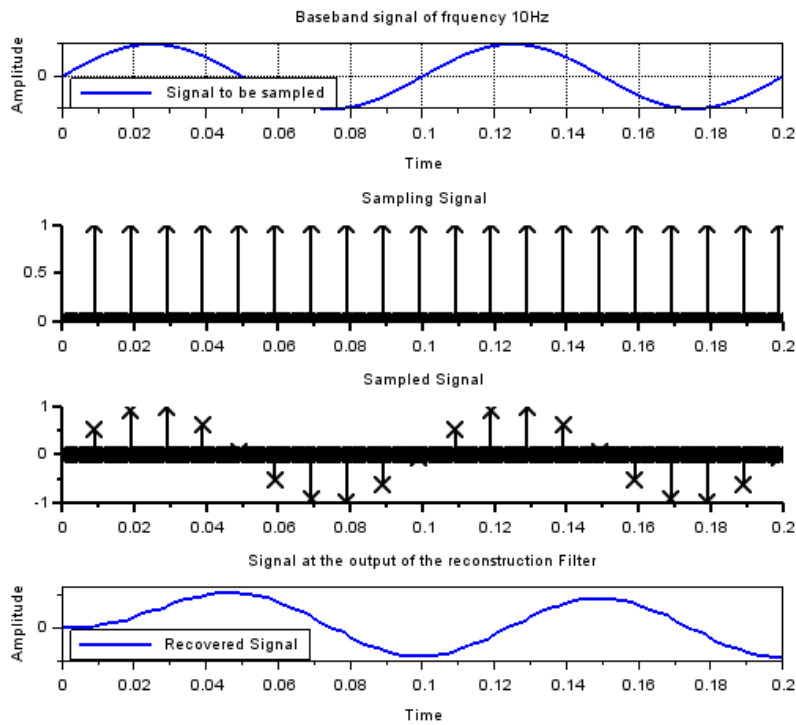


Figure 2.3: Instantaneous Sampling

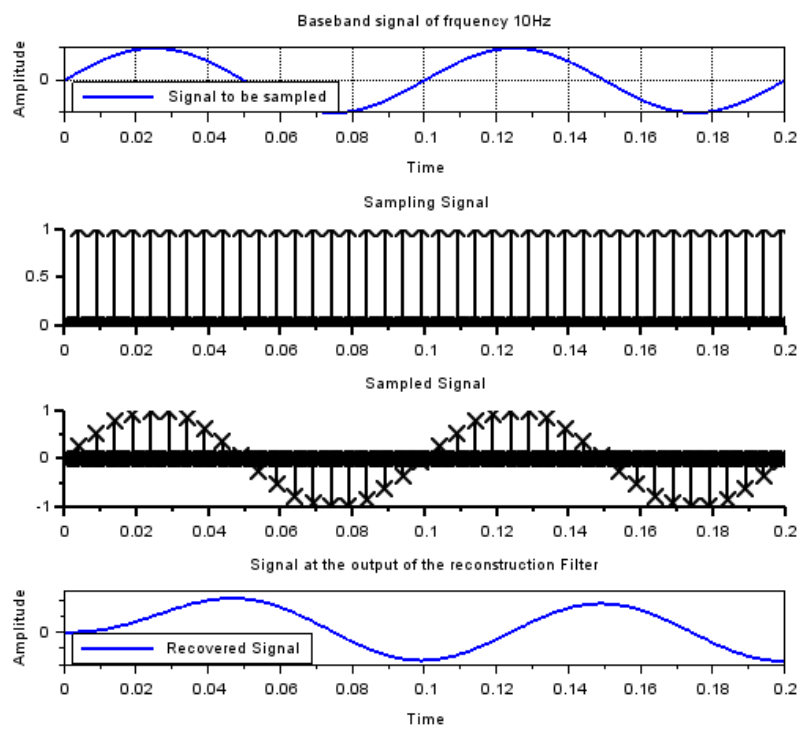


Figure 2.4: Instantaneous Sampling

```

    recovery from the sampled version //Windows 10
2 //Scilab 6.1.0
3 clear
4 clc
5 t1=input('Enter the lower limit of time axis:')
6 t2=input('Enter the upper limit of time axis:')
7 s=input('Enter the spacing between the adjacent
    value of time axis:')
8 t=t1:s:t2;
9 t1=ones(1,length(t));
10 f=input('Enter the baseband signal frequency:')
11 x=sin(2*%pi*f*t);
12 n=input('Enter the integer which decides the width
    of the pulse:')
13 sa=[0 ones(1,n) zeros(1,n)]
14 //Generation of Sampling signal which is a
    rectangular Pulse Train
15 while length(sa)<=length(t)
16     sa=[sa ones(1,n) zeros(1,n)]
17 end
18 sa(length(t)+1:length(sa))=[];
19 //Generation of sampled Signal
20 NAT=sa.*x;
21 //Reconstruction Filter
22 RC=1/(2*%pi*f);
23 h=(1/RC)*exp(-t/RC);
24 //Signal reconstruction
25 y=conv(h,conv(h,NAT));
26 subplot(4,1,1)
27 plot(t,x)
28 plot(t,x,'linewidth',3)
29 xgrid
30 xtitle("Baseband signal of frquency 10Hz(to be
    Sampled)", "Time", "Amplitude")
31 subplot(4,1,2)
32 plot(t,sa,'linewidth',3)
33 xgrid
34 xtitle("Rectangular Pulse Train(Sampling Signal)", "

```

```
Scilab 6.1.0 Console
Enter the lower limit of time axis:0
Enter the upper limit of time axis:0.2
Enter the spacing between the adjacent value of time axis:0.001
Enter the baseband signal frequency:10
Enter the integer which decides the width of the pulse:4
```

Figure 2.5: Natural sampling

```
    Time", "Amplitude")
35 subplot(4,1,3)
36 plot(t,NAT, 'linewidth', 3)
37 xgrid
38 xtitle(" Sampled Signal under Natural Sampling", "
    Time", "Amplitude")
39 subplot(4,1,4)
40 plot(t,y(1:length(t))/length(y), 'linewidth', 2)
41 xgrid
42 xtitle("Recovered Signal", "Time", "Amplitude")
43
44
45 //Testcase
46 // lower limit of time axis: 0
47 //upper limit of time axis: 0.2
48 //spacing between the adjacent value of time axis:
    0.001
49 // baseband signal frequency: 10
50 //integer which decides the width of the pulse: 4
```

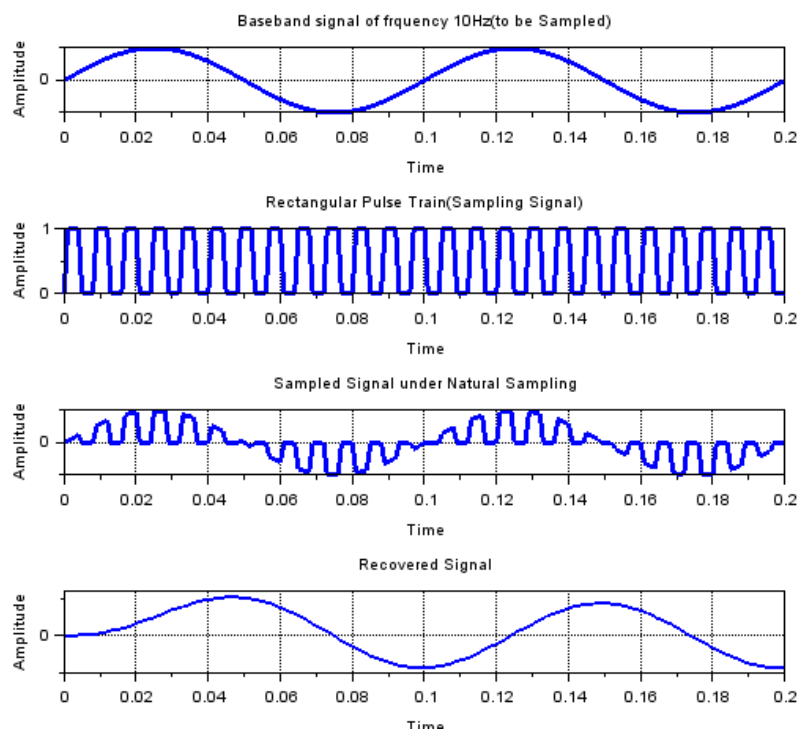


Figure 2.6: Natural sampling

Scilab code Solution 2.2 Flat Top sampling

```
1 //This program generates the FlatTop sampled version
   of sin(20.pi.t) and simulates its recovery from
   the sampled version
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 t1=input('Enter the lower limit of time axis:')
7 t2=input('Enter the upper limit of time axis:')
8 s=input('Enter the spacing between the adjacent
   value of time axis:')
9 t=t1:s:t2
10 f=input('Enter the baseband signal frequency:')
11 x=sin(2*%pi*f*t);
12 n=input('Enter the integer which decides the width
   of the pulse:')
13 sa=[0 ones(1,n) zeros(1,n)]
14 //Generation of rectangular Pulse Train which is the
   Sampling signal
15 while length(sa)<=length(t)
16     sa=[sa ones(1,n) zeros(1,n)]
17 end
18 sa(length(t)+1:length(sa))=[];
19 //Generation of sampled signal
20 FLA=sa.*x;
21 //Making the top of the sample Flat
22 for i=1:length(sa)
23     if sa(i)==1
24         FLA(i+1:i+n)=FLA(i+1)
25     end
26 end
27 //Reconstruction Filter
```

```

28 RC=1/(2*%pi*f);
29 h=(1/RC)*exp(-t/RC);
30 //Signal reconstruction
31 y=conv(h,conv(h,FLA));
32 subplot(4,1,1)
33 plot(t,x,'linewidth',3)
34 xgrid
35 xtitle("Baseband signal of frquency 10Hz(to be
        sampled)","Time","Amplitude")
36 subplot(4,1,2)
37 plot(t,sa,'linewidth',3)
38 xgrid
39 xtitle("Rectangular Pulse Train(Sampling Signal)","
        Time","Amplitude")
40 subplot(4,1,3)
41 plot(t,FLA,'linewidth',3)
42 xgrid
43 xtitle(" Flat Top Sampled Signal","Time","Amplitude"
        )
44 subplot(4,1,4)
45 plot(t,y(1:length(t))/length(y),'linewidth',3)
46 xgrid
47 xtitle(" Recovered Signal","Time","Amplitude")
48 //Testcase
49 // lower limit of time axis: 0
50 //upper limit of time axis: 0.2
51 //spacing between the adjacent value of time axis:
    0.001
52 // baseband signal frequency: 10
53 //integer which decides the width of the pulse: 4

```

```
Scilab 6.1.0 Console
Enter the lower limit of time axis:0
Enter the upper limit of time axis:0.2
Enter the spacing between the adjacent value of time axis:0.001
Enter the baseband signal frequency:10
Enter the integer which decides the width of the pulse:4
```

Figure 2.7: Flat Top sampling

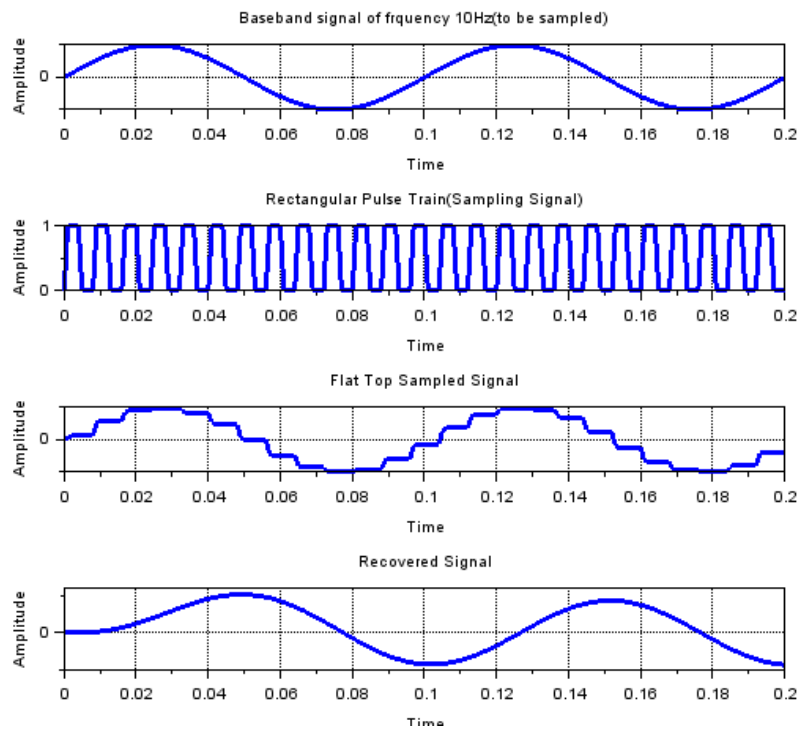


Figure 2.8: Flat Top sampling

Experiment: 3

Wave Form Synthesis

Scilab code Solution 3.0 Staircase waveform

```
1 //Waveform synthesis of  $x(t)=2u(t)-3u(t-2)+2u(t-4)$ 
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 //Plot of x(t)
7 f=input('Enter the sampling frequency:')
8 T=1/f;
9 L=input('Enter the lower bound for the time axis of
   x(t): ')
10 U=input('Enter the upper bound for the time axis of
   x(t): ')
11 t=L-1:T:U+2;
12 x=zeros(1,length(t));
13 y=x;
14 x(find(t==0):find(t==2))=2;
15 x(find(t==2):find(t==4))=-3;
16 x(find(t==4):length(t))=2;
17 //Synthesis
18 z=find(diff(x)==2);
19 y(z(1)+1:length(y))=2;
```

```

20 subplot(4,1,1)
21 xset("thickness",3)
22 plot2d2(t,x,rect=[L-1 min(x)-1 U+2 max(x)+1])
23 xtitle("x(t)","time","Amplitude")
24 legend('2u(t)-3u(t-2)+2u(t-4) with f=1000',3)
25 xstring(t(find(t==1)),x(find(t==1)),["2"])
26 xstring(t(find(t==3)),x(find(t==3)),["-3"])
27 xstring(t(find(t==5)),x(find(t==5)),["2"])
28 subplot(4,1,2)
29 xset("thickness",3)
30 plot2d2(t,y,rect=[L-1 min(y) U+2 max(y)+1])
31 xtitle("", "time", "Amplitude")
32 legend('The First Constituent Step Function',2)
33 xstring(t(find(t==1)),x(find(t==1)),["2"])
34 y=y-y;
35 z=find(diff(x)==-5);
36 y(z(1)+1:length(y))=-5;
37 subplot(4,1,3)
38 xset("thickness",3)
39 plot2d2(t,y,rect=[L-1 min(y)-1 U+2 max(y)])
40 xtitle("", "time", "Amplitude")
41 legend('The Second Constituent Step Function')
42 xstring(t(find(t==2)),x(find(t==2)),["-5"])
43 y=y-y;
44 z=find(diff(x)==5);
45 y(z(1)+1:length(y))=5;
46 subplot(4,1,4)
47 xset("thickness",3)
48 plot2d2(t,y,rect=[L-1 min(y) U+2 max(y)+1])
49 xtitle("", "time", "Amplitude")
50 legend('The Third Constituent Step Function',2)
51 xstring(t(find(t==4)),x(find(t==4)),["5"])
52
53 //output test case
54 //sampling frequency 1000
55 //lower bound for the time axis of x(t) 0
56 //upper bound for the time axis of x(t) 5

```

```

Scilab 6.1.0 Console
Enter the sampling frequency:1000
Enter the lower bound for the time axis of x(t):0
Enter the upper bound for the time axis of x(t):5
Warning: This feature will be permanently removed in Scilab 6.1

```

Figure 3.1: Staircase waveform

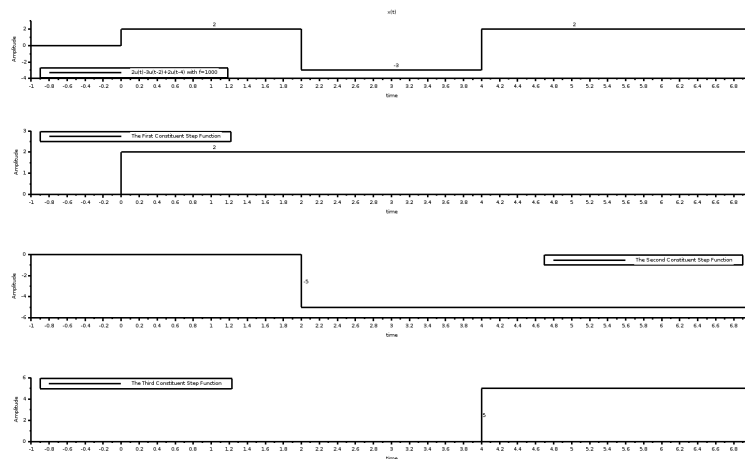


Figure 3.2: Staircase waveform

Scilab code Solution 3.1 Triangular Pulse

```

1 //Waveform synthesis of  $x(t)=r(t)-2r(t-1)+r(t-2)$ 
2 //Windows 10

```

```

3 //Scilab 6.1.0
4 clear
5 clc
6 f=input('Enter the sampling frequency:')
7 T=1/f;
8 U=input('Enter the upper bound for the time axis:')
9 t=0:T:U;
10 //Finding the first Constituent Ramp
11 x=t;
12 //Finding the second constituent Ramp
13 y=zeros(1,length(t));
14 z=y;
15 i=find(t==1);
16 j=length(i:length(t));
17 y(i:i+j-1)=2*x(1:j)
18 //Finding the third constituent Ramp
19 i=find(t==2);
20 j=length(i:length(t));
21 z(i:i+j-1)=x(1:j);
22 subplot(2,2,1)
23 xset("thickness",2)
24 plot2d(t,x-y+z,rect=[0 0 U 1])
25 xtitle('x(t)=r(t)-2r(t-1)+r(t-2) with f=10 and U=3'
        , 'Time', 'Amplitude')
26 legend('x(t)')
27 xgrid
28 subplot(2,2,2)
29 xset("thickness",2)
30 plot2d(t,x,rect=[0 0 U 1])
31 xtitle('First constituent ramp signal', 'Time', '
        Amplitude')
32 legend('r(t)')
33 xgrid
34 subplot(2,2,3)
35 xset("thickness",2)
36 plot2d(t,-y,rect=[0 min(-y) U max(y)])
37 xtitle('Second constituent ramp signal', 'Time', '
        Amplitude')

```

```
Scilab 6.1.0 Console ? ^ X
Enter the sampling frequency:10
Enter the upper bound for the time axis:3
```

Figure 3.3: Triangular Pulse

```
38 legend('-2r(t-1)')
39 xgrid
40 subplot(2,2,4)
41 xset("thickness",2)
42 plot2d(t,z,rect=[0 0 U 1])
43 xtitle('Third constituent ramp signal','Time','
    Amplitude')
44 legend('r(t-2)',3)
45 xgrid
46 //output Test case
47 //sampling frequency 10
48 // upper bound for the time axis 3
```

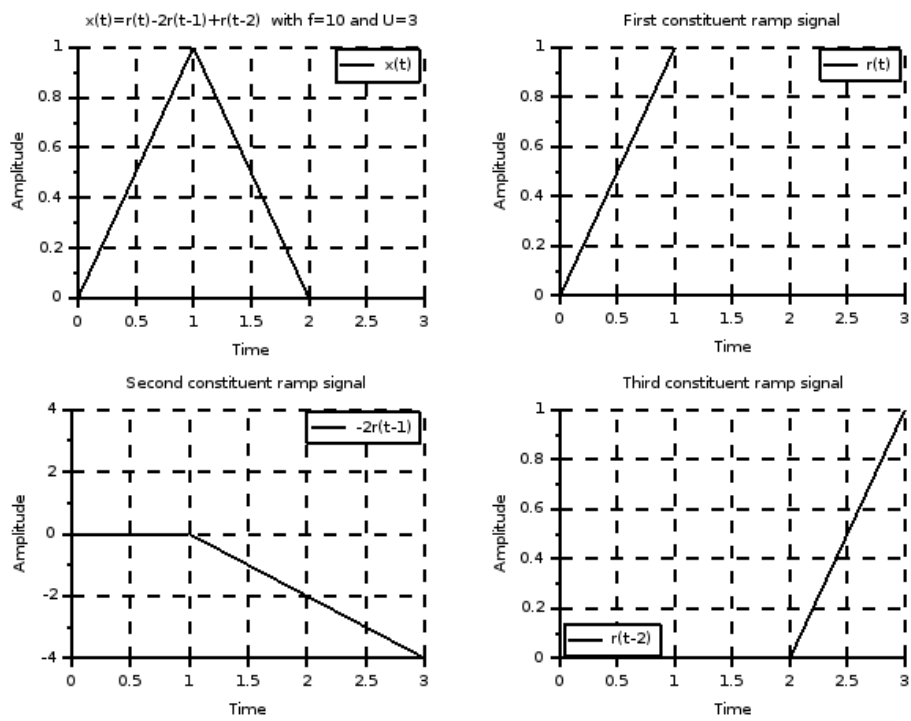


Figure 3.4: Triangular Pulse

Experiment: 4

Location of Poles and Zeros of a given Transfer function in S-plane and Z-plane

Scilab code Solution 4.0 SPlane

```
1 //This program finds the poles and zeros of  $H(s) = \frac{s^2+3s+4}{s^2+3s+12}$  and gives the pole-zero  
  plot in S-Plane  
2 //Windows 10  
3 //Scilab 6.1.0  
4 clear  
5 clc  
6 a=poly([4 3 1], "s", "coeff")  
7 b=poly([12 3 1], "s", "coeff")  
8 z=roots(a);  
9 p=roots(b);  
10 disp('The poles of the given H(s) are')  
11 disp(p)  
12 disp('The zeros of the given H(s) are')  
13 disp(z)  
14 h=syslin('c', a/b)  
15 disp('The Transfer Function is H(s)=', h)
```

```

16 plzr(h)
17 title('Pole-Zero plot of H(s)=(s^2+3*s+4)/(s^2+3*s
      +12)')
18
19 //output Testcase
20 //"The poles of the given H(s) are"
21 //-1.5 + 3.122499 i
22 //-1.5 - 3.122499 i
23 //"The zeros of the given H(s) are"
24 //-1.5 + 1.3228757 i
25 //-1.5 - 1.3228757 i
26 //"The Transfer Function is H(s)="
27 //4 +3s + s
28 //-----
29 //12 +3s + s

```

Scilab code Solution 4.1 Zplane

```

1 //This program finds the poles and zeros of H(z)=z
  ^2/(z^3+2.z^2-z-2) and gives the pole-zero plot
  in Z-Plane
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 a=poly([0 0 1],"z","coeff")
7 b=poly([-2 -1 2 1],"z","coeff")
8 z=roots(a);
9 p=roots(b);
10 disp('The poles of the given H(z) are')
11 disp(p)
12 disp('The zeros of the given H(z) are')
13 disp(z)

```

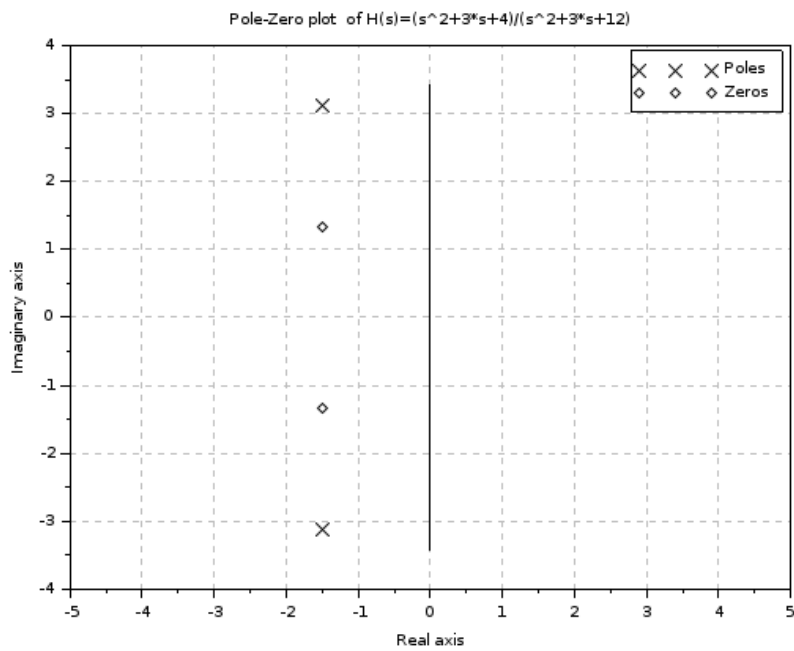


Figure 4.1: SPlane

```

14 h=syslin('d', a/b)
15 disp('The Transfer Function is H(z)=',h)
16 plzr(h)
17 title('Pole-Zero plot of H(z)=z^2/(z^3+2.z^2-z-2)')
18
19 //output Test Case
20 //"The poles of the given H(z) are"
21 // 1. + 0.i
22 // -2. + 0.i
23 // -1. + 0.i
24 //"The zeros of the given H(z) are"
25 // 0. + 0.i
26 // 0. + 0.i
27 // "The Transfer Function is H(z)="
28 //      z
29 //  -----
30 //  -2 -z +2 z  + z

```

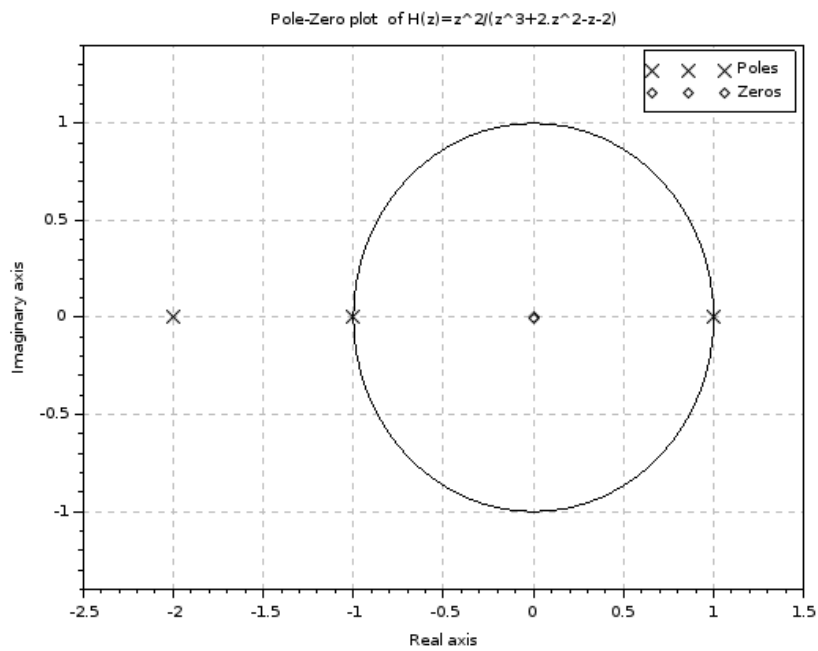


Figure 4.2: Zplane

Experiment: 5

Removal of Noise from the combination of signal and noise using Auto/Cross correlation

Scilab code Solution 5.0 Noise Removal for sequence

```
1 //Removal of Noise from the combination of discrete
   Signal and noise
2 //Noise removal is facilitated using the cross
   correlation of sequence plus Noise and an impulse
   train
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 //Finding the Length of the sequence using
   Autocorrelation
8 x=input('Enter the sequence')
9 x1=x;
10 //generation of random noise sequence
11 rand('normal')
12 N=rand(1,length(x));
13 y=x+N;
```

```

14 h=flipdim(y,2);
15 [m,n]=max(conv(y,h));
16 //Removal of Noise/Estimation of the Signal in the
    presence of Noise
17 x1=x;
18 l=length(x);
19 c=input('Enter the number of cycles')
20 I=eye(1,l);
21 I1=I;
22 //creating the periodic extensions
23 for i=1:c-1
24     x=[x x1];
25     I=[I I1];
26 end
27 rand('normal')
28 q=rand(1,length(x));
29 p=x+q;
30 //correlating the signal plus noise with an impulse
    train
31 for i=1:l
32     y(i)=sum(p.*I(i,:));
33 end
34 //The position of the maximum value of
    Autocorrelation will be the length of the
    sequence
35 disp('Period or length of the sequence x(n) is ',n)
36 disp('The estimate of x(n) is ',y/c)
37 //output test case
38 //sequence:[1 3 5 7]
39 //number of cycles:50
40 //Period or length of the sequence x(n) is 4.
41 //The estimate of x(n) is
42 //0.9119912    3.1947049    4.9297445    7.0546042

```

```
Scilab 6.1.0 Console
Enter the sequence[1 3 5 7]
Enter the number of cycles50
```

Figure 5.1: Noise Removal for sequence

Scilab code Solution 5.1 Noise Removal for signal

```
1 //Removal of Noise from the combination of
   sinusoidal Signal plus noise
2 //Noise removal is facilitated using the cross
   correlation of signal plus Noise and an impulse
   train
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 //Generation of signal plus Noise
8 f=input('Enter the frequency of the signal:')
9 T=1/f;
10 t=0:T/f:T;
11 x=sin(2*%pi*f*t);
12 x1=x;
13 //generation of random noise
14 rand('normal')
15 N=rand(1,length(t));
16 //signal+Noise
17 y=x+N;
18 h=flipdim(y,2);
19 //Finding the Period/length of a sinusoidal signal,
   mixed with Noise
20 [m,n]=max(conv(y,h));
21 l=length(x);
22 c=input('Enter the number of cycles:')
23 I=eye(1,l);
24 I1=I;
```

```

25 for i=1:c-1
26     x=[x x1]
27     I=[I I1]
28 end
29 rand('normal')
30 q=rand(1,length(x))
31 p=x+q;
32 for i=1:l
33     y(i)=sum(p.*I(i,:));
34 end
35 //The position of the maximum value of
    Autocorrelation will be the length of the
    sequence
36 disp('The length of the signal is',n)
37 subplot(3,1,1)
38 xset("thickness",3)
39 plot2d(t,x1,rect=[0 min(x1)-1 T max(x1)])
40 xtitle("x(t)","Time","Amplitude")
41 legend("sinusoidal signal of frequency 10 Hz")
42 subplot(3,1,2)
43 xset("thickness",3)
44 plot2d(t,p(1:length(x1)),rect=[0 min(p)-0.5 T max(p)
    ])
45 xtitle("x(t)+n(t)","Time","Amplitude")
46 legend("Signal plus noise")
47 subplot(3,1,3)
48 xset("thickness",3)
49 plot2d(t,y,rect=[0 min(y)-0.5 T max(y)])
50 xtitle("Estimated x(t)","Time","Amplitude")
51 legend("Estimated sinusoidal signal")
52
53
54 //output test case
55 //frequency of the signal:10
56 //number of cycles:10
57 //The length of the signal is :11.
58
59 //output test case

```

```
Scilab 6.1.0 Console ?  
  
Enter the frequency of the signal:10  
  
Enter the number of cycles:10  
  
"The length of the signal is"  
  
11.
```

Figure 5.2: Noise Removal for signal

```
60 //frequency of the signal:10  
61 //number of cycles:750  
62 //The length of the signal :11  
63  
64 //About the result  
65 //The rand function generates dissimilar data in  
    each run of the code  
66 //Hence, Result will vary from run to run of the  
    code
```

```

Scilab 6.1.0 Console
Enter the frequency of the signal:10
Enter the number of cycles:750

"The length of the signal is"

11.

```

Figure 5.3: Noise Removal for signal

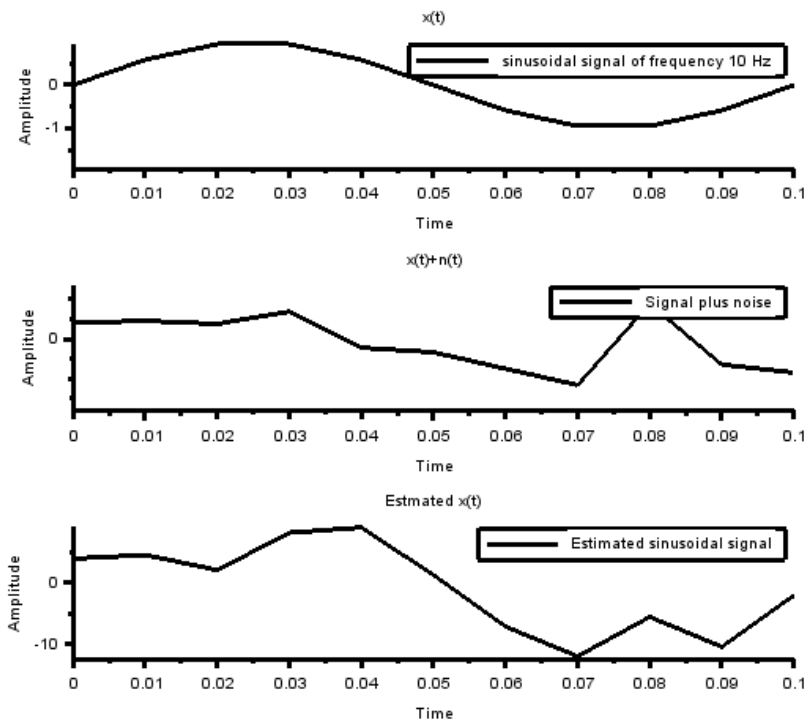


Figure 5.4: Noise Removal for signal

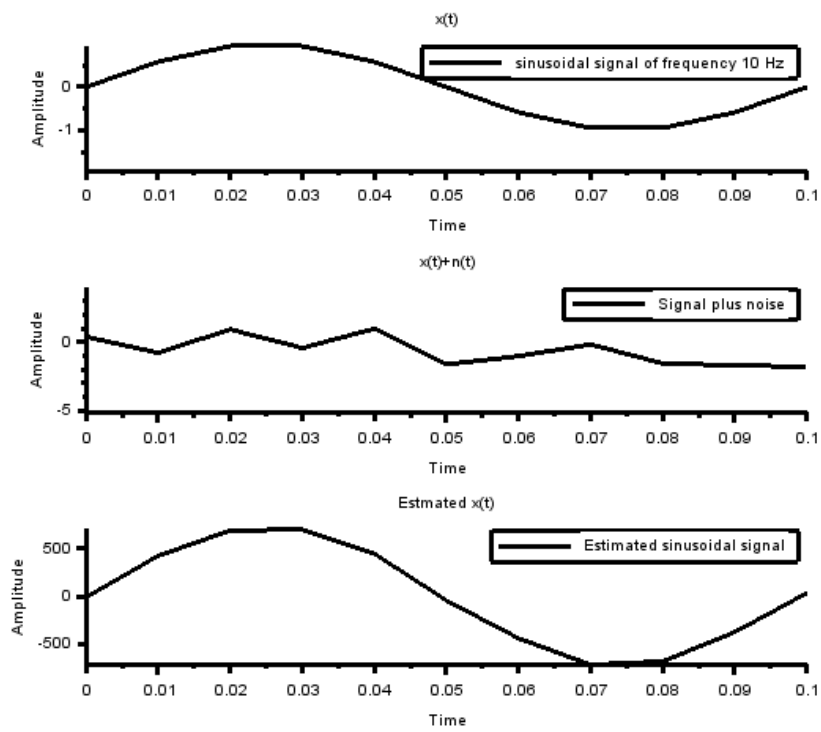


Figure 5.5: Noise Removal for signal

Experiment: 6

Verification of Physical realizability and Stability of a given LTI system

Scilab code Solution 6.0 Causality and Stability

```
1 //Checking the given Discrete LTI system for its
   physical realizability and stability
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 a=input('Enter the coefficients of numerator in the
   order of decreasing order of the variable z:')
7 b=input('Enter the coefficients of denominator in
   the order of decreasing order of the variable z:')
8 p=roots(a);
9 q=roots(b);
10 i=find(abs(q)<1);
11 R1=input('Enter the lower bound of ROC:')
12 R2=input('Enter the upper bound of ROC:')
13 //Checking for Causality
```

```

14 //The ROC of a causal Discrete LTI system should
    include infinity and H(z) should not have the
    order of the numerator greater than that of the
    denominator
15 if length(p)<=length(q) & R2==%inf
16     ans1='The system is ''causal''
17     ans2='Hence, is ''Physically Realizable''
18     disp(ans1)
19     disp(ans2)
20 else
21     ans1='The system is ''not causal''
22     ans2='Hence, is ''not Physically Realizable''
23     disp(ans1)
24     disp(ans2)
25 end
26 //checking for Stability
27 //ROC of a stable discrete system function should
    include the unit circle.
28 // A causal system is stable if all the poles lie
    within the unit circle.
29 if R1<1&R2>1 | length(i)==length(q)
30     disp('System is ''stable''')
31 else
32     disp('System is ''unstable''')
33 end
34
35 // output test case 1
36 //coefficients of numerator in the order of
    decreasing order of the variable z: [1 0]
37 //coefficients of denominator in the order of
    decreasing order of the variable z: [3 -4 1]
38 //lower bound of ROC: 1
39 //upper bound of ROC: %inf
40 //The system is 'causal'
41 //Hence, is 'Physically Realizable'
42 //System is 'unstable'
43
44 //output test case 2

```

```
Scilab 6.1.0 Console
Enter the coefficients of numerator in the order of decreasing order of the variable z:[1 0]
Enter the coefficients of denominator in the order of decreasing order of the variable z:[3 -4 1]
Enter the lower bound of ROC:1
Enter the upper bound of ROC:inf
```

Figure 6.1: Causality and Stability

```
Scilab 6.1.0 Console
Enter the coefficients of numerator in the order of decreasing order of the variable z:[1 0]
Enter the coefficients of denominator in the order of decreasing order of the variable z:[3 -4 1]
Enter the lower bound of ROC:1/3
Enter the upper bound of ROC:1
```

Figure 6.2: Causality and Stability

```
45 //coefficients of numerator in the order of
    decreasing order of the variable z: [1 0]
46 //coefficients of denominator in the order of
    decreasing order of the variable z: [3 -4 1]
47 //lower bound of ROC: 1/3
48 //upper bound of ROC: 1
49 //The system is 'not causal'
50 //Hence,is 'not Physically Realizable'
51 //System is 'unstable'
```

Experiment: 7

Plotting the CDF and pdf of a Random Variable

Scilab code Solution 7.0 CDF and pdf

```
1 //Finding the CDF and pdf of a discrete random
   variable from its pmf
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 x=input('Enter the values taken by Random Variable:')
   )
7 p=input('Enter the probabilities:')
8 //computation of CDF values
9 for i=2:length(x)
10     p(i)=p(i)+p(i-1);
11 end
12 x1=min(x)-1:0.01:max(x)+2;
13 cdf=zeros(1,length(x1));
14 cdf(find(x1==max(x)):length(x1))=1;
15 //creating CDF vector
16 for i=1:length(x)-1
17     cdf(find(x1==x(i)):find(x1==x(i+1)))=p(i);
```

```

18 end
19 //computation of pdf
20 pdf=diff(cdf);
21 pdf(length(pdf)+1)=0;
22 subplot(2,1,1)
23 xset("thickness",3)
24 plot2d2(x1,cdf,rect=[min(x1) min(p)-0.5 max(x1) max(
    p)+0.25 ])
25 a=gca();
26 a.x_location="origin";
27 a.y_location="origin";
28 xtitle('Plot of CDF','values taken by the random
    variable','Cumulative Distribution Function')
29 legend("x=[-1 1 3 5] , p=[1/2 1/8 1/8 1/4 ]",4)
30 xstring(x1(find(x1==-1)),cdf(find(cdf==1/2)),["0.5"
    ])
31 xstring(x1(find(x1==1)),cdf(find(cdf==0.625)),["
    0.625"])
32 xstring(x1(find(x1==3)),cdf(find(cdf==0.75)),["0.75"
    ])
33 xstring(x1(find(x1==5)),cdf(find(cdf==1)),["1"])
34 subplot(2,1,2)
35 xset("thickness",3)
36 plot2d3(x1,pdf,rect=[min(x1) 0 max(x1) max(p) ])
37 b=gca();
38 b.x_location="origin";
39 b.y_location="origin";
40 xtitle('Plot of pdf','values taken by the random
    variable','Probability Density Function')
41 legend("Probability Density Function")
42 xstring(x1(find(x1==-1)),pdf(find(pdf==1/2)),["1/2"
    ])
43 xstring(x1(find(x1==1)),pdf(find(pdf==1/8)),["1/8"])
44 xstring(x1(find(x1==3)),pdf(find(pdf==1/8)),["1/8"])
45 xstring(x1(find(x1==5)),pdf(find(pdf==1/4)),["1/4"])
46
47 //output testcase
48 //values taken by Random Variable: [-1 1 3 5]

```

```

Scilab 6.1.0 Console
Enter the values taken by Random Variable:[-1 1 3 5]
Enter the probabilities:[1/2 1/8 1/8 1/4]

```

Figure 7.1: CDF and pdf

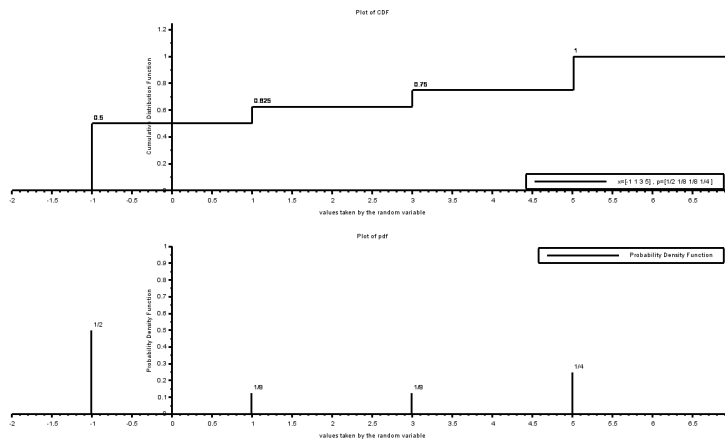


Figure 7.2: CDF and pdf

```

49 //probabilities:[1/2 1/8 1/8 1/4]
50 // If the probabilities are changed, the xstring
    statements are to be modified accordingly

```

Experiment: 8

Computation of Moments of a Random variable

Scilab code Solution 8.0 Moments Discrete

```
1 //Computation of Moments of a Discrete random
   variable
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 x=input('Enter the values taken by Random Variable')
7 p=input('Enter the probabilities')
8 //Computation fo First Moment about origin
9 M=sum(x.*p);
10 //Computation of Second Moment about origin
11 MS=sum((x.^2).*p)
12 //Computation of Variance
13 V=MS-M^2;
14 //Computation of Skew
15 S=sum(((x-M).^3).*p);
16 //Computation of Kurtosis
17 K=sum(((x-M).^4).*p)
18 disp('The mean, Mean Square value ,variance ,skew and
```

```

Scilab 6.1.0 Console
Enter the values taken by Random Variable[1 2 3 4]
Enter the probabilities[1/4 1/4 1/4 1/4]

```

Figure 8.1: Moments Discrete

```

    kurtosis of the given random variable are
    respectively ')
19 disp(M)
20 disp(MS)
21 disp(V)
22 disp(S)
23 disp(K)
24 //output testcase
25 // values taken by Random Variable [1 2 3 4]
26 // probabilities [1/4 1/4 1/4 1/4]
27 //The mean, Mean Square value, variance, skew and
    kurtosis of the given random variable are
    respectively 2.5 ,7.5, 1.25 ,0 2.5625

```

Scilab code Solution 8.1 Moments Continuous

```

1 //computation of Moments of a continuous random
    variable X with density  $f(x)=x/6$ , for  $2 \leq x \leq 4$ ;=0
    elsewhere.
2 //Windows 10
3 //Scilab 6.1.0
4 clear
5 clc
6 a=input('Lower bound for the density function')
7 b=input('Upper bound for the density function')
8 //Computation of Mean(M) of the Random Variable
9 M=integrate('x*x/6', 'x', a, b);

```

```
Scilab 6.1.0 Console
lower bound for the density function2
Upper bound for the density function4
```

Figure 8.2: Moments Continuous

```
10 //Computation of Mean Square value(MS) of the random
    variable
11 MS=integrate(' (x^2)*x/6 ', 'x', a, b);
12 //Computation of Variance (V) of the random variable
13 V=MS-M^2;
14 //Computation of Skew(S) of the random Variable
15 S=integrate(' ((x-M)^3)*x/6 ', 'x', a, b);
16 //computation of Kurtosis(K) of the random variable
17 K=integrate(' ((x-M)^4)*x/6 ', 'x', a, b);
18 mprintf('Mean, Mean Square value, Variance, Skew and
    Kurtosis of random variable are respectively %d,
    %d, %f, %f, %f', M, MS, V, S, K)
19
20 //output testcase
21 //Lower bound for the density function:2
22 //Upper bound for the density function:4
23 //The mean, Mean Square value, variance, skew and
    kurtosis of random variable are respectively
    3.1111111, 10, 0.3209877, // -0.0417010,
    0.1946045
```

Experiment: 9

Verification of Central Limit Theorem

Scilab code Solution 9.0 Central Identical

```
1 // verification of Central Limit Theorem for
  independent uniformly distributed random
  variables
2 // Density of sum of n number of independent
  identically distributed random variables
  approaches gaussian Density in the //limit n
  tends to infinity
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 a=input('Enter the lower bound for the density of
  the random variable:')
8 b=input('Enter the upper bound for the density of
  the random Variable:')
9 x=a:b;
10 x1=a-2:0.01:b+2;
11 //Generation of Uniform Density
12 f=(1/(b-a))*ones(1,length(x));
```

```

13 f11=f;
14 f1=zeros(1,length(x1));
15 f1(find(x1==a):find(x1==b))=1/(b-a);
16 n=input('Enter the number of Random variables:');
17 //finding the density of sum of random variables
18 for i=1:n-1
19 f=conv(f,f11);
20 end
21 x11=n*a:n*b;
22 subplot(2,1,1)
23 plot(x1,f1,'linewidth',3)
24 xgrid
25 xlabel('values taken by the random variable')
26 ylabel('Unifrom Density')
27 title('Uniform Random variable')
28 legend("Uniform density")
29 mtlb_axis([min(x1) max(x1) 0 max(f1)+0.1])
30 subplot(2,1,2)
31 plot(x11,f,'linewidth',3)
32 xgrid
33 xlabel('values taken by the Random variable=Sum of
         independent uniform random variables')
34 ylabel('Density of sum of independent uniform random
         varaibales')
35 title('Density of sum of 10 independent uniform
         random variables')
36 legend("Density of sum of independent random
         variables ")
37 mtlb_axis([min(x11) max(x11) 0 max(f)+0.25])
38
39 //output testcase
40 //lower bound for the density of the random variable
   :   -2
41 //upper bound for the density of the random Variable
   :     2
42 //number of Random variables: 10

```

```
Scilab 6.1.0 Console
Enter the lower bound for the density of the random variable:-2
Enter the upper bound for the density of the random Variable:2
Enter the number of Random variables:10
```

Figure 9.1: Central Identical

Scilab code Solution 9.1 Central Non Identical

```
1 //Verification of Central Limit Theorem for
   independent uniform and exponential random
   Variables
2 //Density of sum of n number of independent
   identically or non identically distributed
   random variables approaches //gaussian
   Density in the limit n tends to infinity
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 //Uniform density
8 a=input('Enter the lower bound for the density of
   the Uniform random variable:')
9 b=input('Enter the upper bound for the density of
   the Uniform random Variable:')
10 x=a:b;
11 U=(1/(b-a))*ones(1,length(x));
```

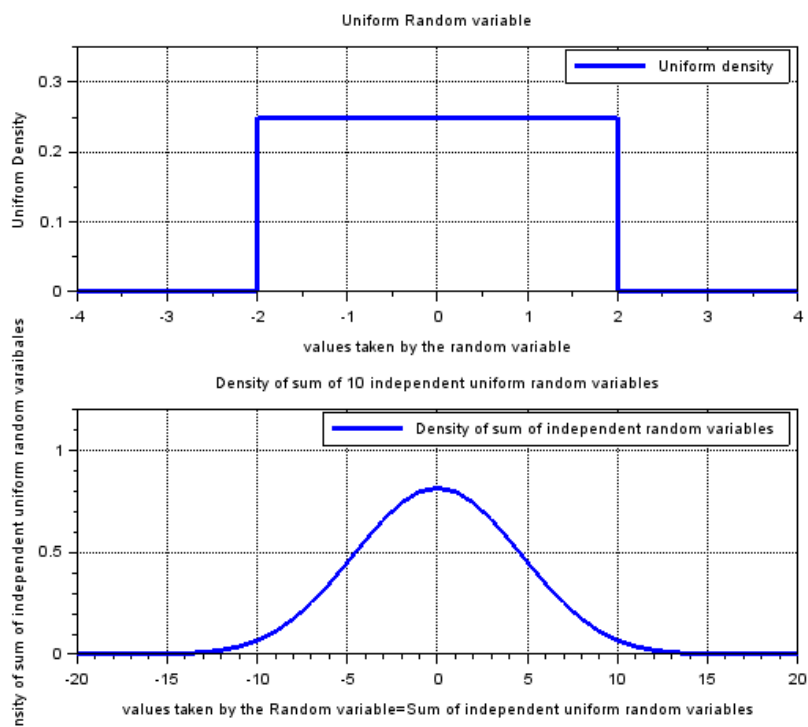


Figure 9.2: Central Identical

```

12 U1=U;
13 //Exponential density
14 k=input('Enter the parameter for the exponential
        random variable: ')
15 E=(1/(1-exp(-k)))*k*exp(-k*x);
16 n1=input('Enter the number of Uniform random
        variables: ')
17 n2=input('Enter the number of Exponenetial random
        variables: ')
18 //Density of sum of Uniform random Variables
19 for i=1:n1-1
20     U=conv(U,U1);
21 end
22 //Density of sum of Uniform and Exponential random
        variables
23 for i=1:n2
24     U=conv(E,U)
25 end
26 x1=(n1+n2)*a:(n1+n2)*b;
27 xgrid
28 plot(x1,U,'linewidth',3)
29 title('Density of sum of random variables')
30 xlabel('values taken by sum of random variables')
31 ylabel('Probability Density function of sum of
        random variables ')
32 legend("Density of sum of exponential and Uniform
        random variables")
33 //output test case
34 //lower bound for the density of the Uniform random
        variable: 0
35 //upper bound for the density of the Uniform random
        Variable: 4
36 //parameter for the exponential random variable: 1
37 //number of Uniform random variables: 10
38 //number of Exponenetial random variables: 10

```

```
Scilab 6.1.0 Console ? ↗ ✕  
Enter the lower bound for the density of the Uniform random variable:0  
Enter the upper bound for the density of the Uniform random Variable:4  
Enter the parameter for the exponential random variable:1  
Enter the number of Uniform random variables:10  
Enter the number of Exponenetial random variables:10
```

Figure 9.3: Central Non Identical

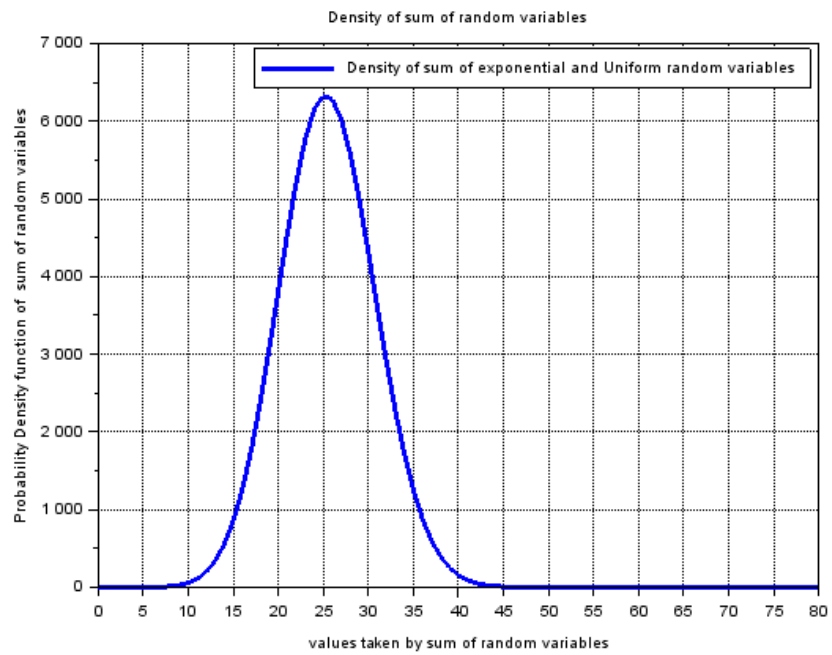


Figure 9.4: Central Non Identical

Experiment: 10

Checking the given random Process for Stationary

Scilab code Solution 10.0 Stationarity

```
1 //Checking thr Given random process for Stationarity
2 //This program checks the random process  $x(t)=A \cos(\omega t+\Theta)$ , where  $\Theta$  is a Uniform random
   variable over  $(0,2\pi)$ 
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 A=input('Enter the Peak of the Process:')
8 fo=input('Enter the value of fo:')
9 t=input('Enter the value of time instant:')
10 k=input('Enter k:')
11 a=input('Enter the lower bound for theta:')
12 b=input('Enter the upper bound for theta:')
13 deff(' [w]=f(theta)', 'w=A*cos(2*%pi*fo*t+theta)')
14 //Scilab's integrate routine tries to achieve
   absolute error atleast  $1e-8$  and relative error
   atleast  $1e-14$ .
15 //If it is less than that, it throws an error and
```

```

    will be displayed that "try with higher //
    tolerance '.
16 //The following syntax fixes the tolerance.
17 M=(1/(b-a))*intg(a,b,f,1e-8,1);
18 deff(' [y]=g(theta)', 'y=(A^2/2)*cos(2*2*%pi*fo*t+2*
    %pi*fo*k+2*theta)');
19 a1=(1/(b-a))*intg(a,b,g,1e-8,1);
20 deff(' [z]=h(theta)', 'z=(A^2/2)*cos(2*%pi*fo*k)');
21 a2=(1/(b-a))*intg(a,b,h);
22 R=a1+a2;
23 disp("The mean and the Autocorrelation of the
    process are respectively")
24 disp(M)
25 disp(" and")
26 disp(R)
27
28 //output testcase
29 //for given t and k
30 //Peak of the Process:1
31 //value of fo:1
32 //value of time instant:1
33 //k:1.5
34 //lower bound for theta:0
35 //upper bound for theta:2*%pi
36 //The mean and the Autocorrelation of the process
    are respectively 1.249D-16(=0) and -0.5
37
38 //Change in t and no Change in k
39 //Peak of the Process:1
40 //value of fo:1
41 //value of time instant:2
42 // k:1.5
43 //lower bound for theta:0
44 //upper bound for theta:2*%pi
45 //The mean and the Autocorrelation of the process
    are respectively 1.249D-16(=0) and -0.5
46
47 // No change in t and change in k

```

```
Scilab 6.1.0 Console
Enter the Peak of the Process:1
Enter the value of fo:1
Enter the value of time instant:1
Enter k:1.5
Enter the lower bound for theta:0
Enter the upper bound for theta:2*pi
```

Figure 10.1: Stationarity

```
48 //Peak of the Process:1
49 //value of fo:1
50 //value of time instant:2
51 // k:2
52 //lower bound for theta:0
53 //upper bound for theta:2*pi
54 //The mean and the Autocorrelation of the process
    are respectively 1.249D-16(=0) and 0.5
55
56 //Mean of the process is zero(constant), and is
    independent of time instant of measurement.
57 //Autocorrelation is a function of k
58 //Hence, the given process is stationary.
```

```
Scilab 6.1.0 Console
Enter the Peak of the Process:1
Enter the value of fo:1
Enter the value of time instant:2
Enter k:2
Enter the lower bound for theta:0
Enter the upper bound for theta:2*pi
```

Figure 10.2: Stationarity

```
Scilab 6.1.0 Console
Enter the Peak of the Process:1
Enter the value of fo:1
Enter the value of time instant:2
Enter k:1.5
Enter the lower bound for theta:0
Enter the upper bound for theta:2*pi
```

Figure 10.3: Stationarity

Experiment: 11

Verification of Weiner-Khinchine Relation

Scilab code Solution 11.0 Wiener Khinchine Theorem

```
1 // Verificatioin of Wiener-Khinchine relation for
   the signal  $x(t)=\sin(30.\pi.t)+\sin(60.\pi.t)$ 
2 // Autocorrelation function and Power spectral
   Density of a signal form a Fourier transform pair
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 fs=input('Enter the sampling Frequency:')
8 T=input('Enter the duration upto which the signal is
   to be plotted:')
9 t=0:1/fs:T;
10 x=sin(30*%pi*t)+sin(60*%pi*t);
11 N=input('Enter the DFT length:')
12 //making the length of x equal to N
13 if length(x)<N
14     x(length(x)+1:N)=0;
15 else
16     if length(x)>N
```

```

17     x(N+1:length(x))=[];
18 end
19 end
20 //computation of N point DFT of the sequence
21 X=fft(x);
22 f=fs*(0:N-1)/N;
23 //computation of PSD = (1/N)*abs(fft)^2 using the
    direct expression
24 PS=(1/N)*(abs(X).^2);
25 //computation of Autocorrelation function of the
    signal
26 R=xcorr(x,x);
27 //making length of R equal to N
28 if length(R)<N
29     R(length(R)+1:N)=0;
30 else
31     if length(R)>N
32         R(N+1:length(R))=[];
33 end
34 end
35 //computation of Power Spectral Density from
    Autocorrelation Function
36 PSD=fft(R);
37 subplot(2,1,1)
38 xset("thickness",3)
39 plot2d3(f,PS)
40 xtitle("Power spectral density computed", 'frequency',
    'Watts/Hz')
41 mtlb_axis([min(f) max(f) min(PS) max(PS)])
42 legend("PSD=(1/N) . |X(k)|^2")
43 subplot(2,1,2)
44 xset("thickness",3)
45 plot2d3(f,abs(PSD))
46 xtitle("Power spectral density computed from
    Autocorrelation Function", 'frequency', 'Watts/Hz')
47 mtlb_axis([min(f) max(f) min(abs(PSD)) max(abs(PSD))
    ])
48 legend("PSD=Fourier Transform of Autocorrelation

```

```
Scilab 6.1.0 Console
Enter the sampling Frequency:100
Enter the duration upto which the signal is to be plotted:10
Enter the DFT length:1024
```

Figure 11.1: Wiener Khinchine Theorem

```
Function”)
49 //sampling Frequency: 100
50 //duration upto which the signal is to be plotted:
    10
51 //DFT length: 1024
```

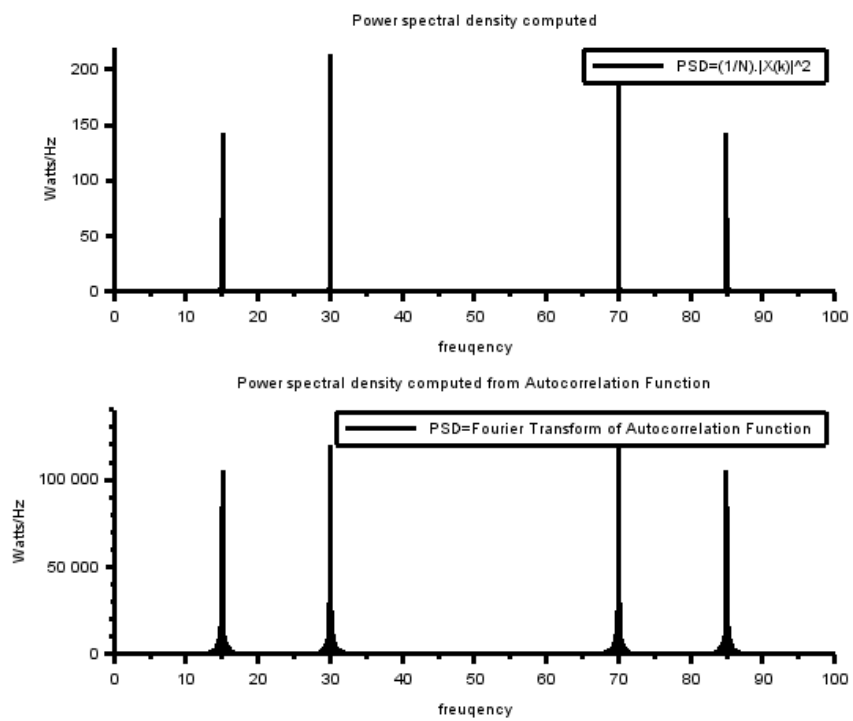


Figure 11.2: Wiener Khinchine Theorem

Experiment: 12

Simulation of Gaussian Random Vectors

Scilab code Solution 12.0 Gaussian

```
1 //Simulation of generation of Bivariate Gaussian
  random vector
2 //R and S are bivariate gaussian random Variables
  which are to be generated, with desired mean,
  sandard deviation and //covariance
3 //Windows 10
4 //Scilab 6.1.0
5 clear
6 clc
7 //sdR=standard deviation of R
8 sdR=input('Enter the desired standard deviation of R
  :')
9 //sdS=standard deviation of S
10 sdS=input('Enter the desired standard deviation of S
  :')
11 //mR=mean of R
12 mR=input('Enter the desired mean of R:')
13 //mS=mean of S
14 mS=input('Enter the desired mean of S:')
```

```

15 //k=correlation coefficient between R and S
16 k=input('Enter the desired correaltion coefficient
    between R and S:')
17 M=input('Enter the number of realizations:')
18 G=[sdR 0;k*sdS sdS*sqrt(1-k^2)];
19 for i=1:M
20 //generation of two standardized Gaussian random
    variables P and Q
21 P=grand(1,1,"nor",0,1);
22 Q=grand(1,1,"nor",0,1);
23 //Transformation of P and Q to the desired mean and
    covariance
24 //G=[sdR 0;k.sdS sdS.sqrt(1-k^2)]
25 //[R S]'=G.*[P Q]'+[mR mS]'
26 rs=G*[P Q]'+[mR mS]';
27 RS(:,i)=rs;
28 end
29 //verifying the mean of the individual random
    variables R and S
30 //meanest is the estimated mean
31 meanestR=mean(RS(1,:));
32 meanestS=mean(RS(2,:));
33 //verifying the covariance matrix
34 //COV(R,S)=E(RS)-E(R).E(S)
35 RS1(1,:)=RS(1,:)-meanestR;
36 RS1(2,:)=RS(2,:)-meanestS;
37 covest=[0 0;0 0];
38 //computation of covariance Matrix [var(R) k.sdR.sdS
    ; k.sdR.sdS var(S)]
39 for i=1:M
40     covest=covest+(RS1(:,i)*RS1(:,i)')/M
41 end
42 //computation of correlation coefficient between R
    and S
43 corcoe=covest(1,2)/sqrt(covest(1,1)*covest(2,2));
44 disp('The mean of the random variable R is ',meanestR
    )
45 disp('The mean of the random variable S is ',meanestS

```

```

)
46 disp('The Standard deviation of R is ',sqrt(covest
    (1,1)))
47 disp('The Standard deviation of S is ',sqrt(covest
    (2,2)))
48 disp('The covariance between R and S is ',corcoe)
49
50 //output testcase
51 //desired standard deviation of R: 1
52 //desired standard deviation of S: 1
53 //desired mean of R: 1
54 //desired mean of S: 1
55 //desired correlation coefficient between R and S:
    0.9
56 //number of realizations: 2000
57
58 //Result
59 //The mean of the random variable R is 0.9668127(
    against the desired = 1)
60 //The mean of the random variable S is 0.9748126(
    against the desired=1)
61 //The Standard deviation of R is 1.0276458(against
    the desired =1)
62 //The Standard deviation of S is 1.0232880(against
    the desired =1)
63 //The covariance between R and S is 0.9015213(
    against the desired =0.9)
64
65 //'grand' function of scilab generates distinct
    results in different runs of the code.
66 //Output in each run will be unique

```

```
Scilab 6.1.0 Console
Enter the desired standard deviation of R:1
Enter the desired standard deviation of S:1
Enter the desired mean of R:1
Enter the desired mean of S:1
Enter the desired correlation coefficient between R and S:0.9
Enter the number of realizations:2000
```

Figure 12.1: Gaussian

```
"The mean of the random variable R is"
0.9668127
"The mean of the random variable S is"
0.9748126
"The Standard deviation of R is"
1.0276458
"The Standard deviation of S is"
1.0232880
"The covariance between R and S is"
0.9015213
```

Figure 12.2: Gaussian