Scilab Manual for Signal Processing by Mrs S. Chaya Electronics Engineering AIKTC [1]

Solutions provided by
Mr R.Senthilkumar- Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology

July 2, 2025

[1]Funded 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 Experiments

List of Figures

Experiment: 1

Image Sampling and Quantization

check Appendix AP 1 for dependency:

`cameraman.jpeg`

Scilab code Solution 1.1 Exp1a

```
//Image Quantization
clear;
clc;
I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\cameraman.jpeg');
quanta = 50;
J = double(I)/255;
J = uint8(J*quanta);
J = double(J)/quanta;
figure
ShowImage(I,'Original Image')
figure
ShowImage(J,'Quantized Image')
```

Figure 1.1: Exp1a

check Appendix AP 1 for dependency:

`cameraman.jpeg`

Scilab code Solution 1.2 Exp1b

```
//Image Sampling
clear;
clc;
I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\cameraman.jpeg');
J = imresize(I,0.5); //Reducing the sampling rate
K1 = imresize(J,2,'nearest'); //Increasing the
    sampling rate
K2 = imresize(J,2,'bilinear');
K3 = imresize(J,2,'bicubic');
figure
ShowImage(I,'Original Image')
figure
ShowImage(J,'Reducing the Sampling Rate by 2')
figure
ShowImage(K1,'Increasing the Sampling Rate by 2
    nearest neighbour method')
figure
ShowImage(K2,'Increasing the Sampling Rate by 2
    bilinear method')
figure
ShowImage(K3,'Increasing the Sampling Rate by 2
    bicubic method')
```

Figure 1.2: Exp1a

Figure 1.3: Exp1b

Figure 1.4: Exp1b

Experiment: 2

Understanding basic relationship between pixel

check Appendix AP 1 for dependency:

`cameraman.jpeg`

check Appendix AP 2 for dependency:

`rice.jpg`

Scilab code Solution 2.1 Exp2a

```
//Image Arithmetic -division ,multiplication ,image
    subtraction and image addition
clc;
clear;
close;
I  = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab
    \scilab\cameraman.jpeg'); //SIVP toolbox
J = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\rice.jpg');//SIVP toolbox
IMA = imadd(I,J); //SIVP toolbox
figure
ShowImage(IMA,'Image Addition')//IPD toolbox
```

```
IMS = imabsdiff(I,J);//SIVP toolbox
figure
ShowImage(IMS,'Image Subtraction');//IPD toolbox
IMD = imdivide(I,J);//SIVP toolbox
IMD = imdivide(IMD,0.01);//SIVP toolbox
figure
ShowImage(uint8(IMD),'Image Division');//IPD toolbox
IMM = immultiply(I,I);//SIVP toolbox
figure
ShowImage(uint8(IMM),'Image Multiply');//IPD toolbox
```

check Appendix AP 1 for dependency:

`cameraman.jpeg`

check Appendix AP 4 for dependency:

`lenna.jpg`

Scilab code Solution 2.2 Exp2b

```
//Image Arithmetic- Distance and Connectivity: To
    understand the notion of connectivity
//and neighborhood defined for a point in an image.
clc;
clear;
close;
//function to convert gray to binary
function X = gray2bin(x)
    xmean = mean2(x);
    [m,n]= size(x);
    X = zeros(m,n);
    for i = 1:m
```

Figure 2.1: Exp2a

Figure 2.2: Exp2a

```
        for j = 1:n
            if x(i,j)> xmean then
                X(i,j) = 1;
            end
        end
    end
endfunction
//function to find total length of two dimensional matrix
function n = numdims(X)
    n = length(size(X));
endfunction
////////////////////////////////////
//Funtion to pad zeros in columns and rows at both ends of an binary image
function B = padarray(b)
    //pad zeros in columns and rows at both ends of an binary image
[m,n] = size(b);
num_dims = length(size(b));
B = zeros(m+num_dims,n+num_dims);
for i = num_dims:m+num_dims-1
    for j = num_dims:m+num_dims-1
        B(i,j) = b(i-1,j-1);
    end
end
endfunction
/////////////////////////////////////
//[1].Euclidean Distance between images and their histograms
I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\lenna.jpg');
J = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\cameraman.jpeg')
h_I = CreateHistogram(I);//IPD toolbox
h_J = CreateHistogram(J);//IPD toolbox
I = double(I);
J = double(J);
```

```
E_dist_Hist = sqrt(sum((h_I-h_J).^2));//Euclidean Distance between histograms of two images
E_dist_images = sqrt(sum((I(:)-J(:)).^2));//Euclidean Distance between two images
disp(E_dist_images,'Euclidean Distance between two images');
disp(E_dist_Hist,'Euclidean Distance between histograms of two images')
//[2]. Connectivity - 8 connected to the background
//exec(gray2bin)
Ibin = gray2bin(I);
Jbin = gray2bin(J);
//conversion of gray image into binary image
conn = [1,1,1;1,1,1;1,1,1];//8-connectivity
//exec('C:\Users\senthilkumar\Desktop\Gautam_PAL_Lab\numdims.sci')
num_dims = numdims(I);
//exec('C:\Users\senthilkumar\Desktop\Gautam_PAL_Lab\padarray.sci')
B = padarray(Ibin);
global FILTER_ERODE;
StructureElement = CreateStructureElement('square', 3);
B_eroded = MorphologicalFilter(B,FILTER_ERODE, StructureElement.Data);//IPD toolbox
//note:StructureElement.Data and conn both are same values
//except that StructureElement.Data is boolean either true or false
p = B&~B_eroded;
[m,n] = size(p);
for i = num_dims:m+num_dims-2
    for j = num_dims:n+num_dims-2
        pout(i-1,j-1) = p(i,j);
    end
end
figure
ShowImage(uint8(I),'Gray Lenna Image')
```

```
figure
ShowImage(Ibin,'Binary Lenna Image')
figure
ShowImage(pout,'8 neighbourhood connectiviy in Lenna
    Image')
//RESULT
//Euclidean Distance between two images
//
//     19797.433
//
// Euclidean Distance between histograms of two
   images
//
//     5770.7
```

Figure 2.3: Exp2b

Figure 2.4: Exp2b

Experiment: 3

Program for Image sharpening.

Scilab code Solution 3.1 Exp3a

```
//Note: Details of scilab software version and OS
    version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//2.Program to sharpen image
//Read image and display it.
//For Colour Image
clc;
clear all;
close;
a = imread('C:\Users\senthilkumar\Desktop\
    signal_processing_lab\hestian.jpg');
ShowColorImage(a,'Original Image')
title('Original Image');
//Sharpen the image and display it.
//b = imsharpen(a);
//figure, imshow(b), title('Sharpened Image');


```

```
radius =1;
amount = 0.8000;
threshold = 0;
// Gaussian blurring filter
filtRadius = ceil(radius*2);
filtSize = 2*filtRadius + 1;
gaussFilt = fspecial('gaussian',[filtSize filtSize],
    radius);
// High-pass filter
sharpFilt = zeros(filtSize,filtSize);
sharpFilt(filtRadius+1,filtRadius+1) = 1;
sharpFilt = sharpFilt - gaussFilt;
sharpFilt = amount*sharpFilt;
sharpFilt(filtRadius+1,filtRadius+1) = sharpFilt(
    filtRadius+1,filtRadius+1) + 1;
B = imfilter(a,sharpFilt);
figure
ShowColorImage(B,'Sharpened Image');
```

check Appendix AP 3 for dependency:

`hestian.jpg`

Scilab code Solution 3.2 Exp3b

```
//Note: Details of scilab software version and OS
    version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//2.b.Program to sharpen image
```

Figure 3.1: Exp3a

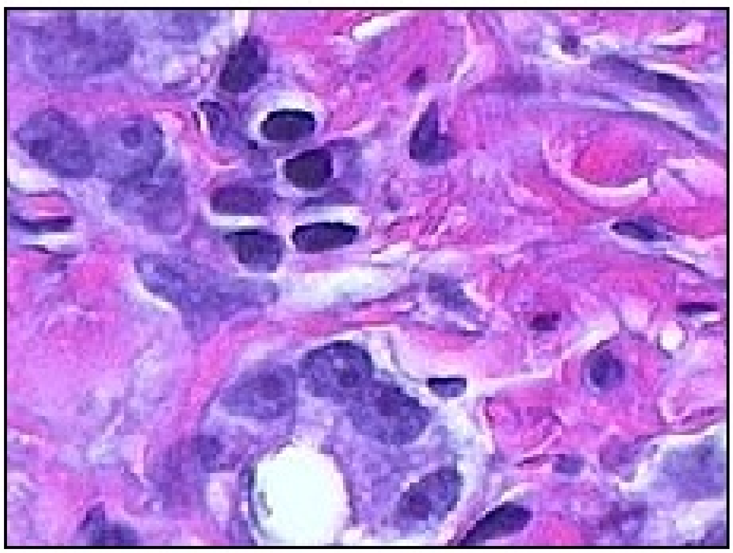


Figure 3.2: Exp3a

```
//Read image and display it.
//For Gray Image
clc;
clear all;
close;
a = imread('C:\Users\senthilkumar\Desktop\
    signal_processing_lab\rice.jpg');//SIVP toolbox
ShowImage(a,'Original Image')//SIVP toolbox
title('Original Image');
//Sharpen the image and display it.
//b = imsharpen(a);
//figure, imshow(b), title('Sharpened Image');


radius =1;
amount = 0.8000;
threshold = 0;
// Gaussian blurring filter
filtRadius = ceil(radius*2);
filtSize = 2*filtRadius + 1;
gaussFilt = fspecial('gaussian',[filtSize filtSize],
    radius);
// High-pass filter
sharpFilt = zeros(filtSize,filtSize);
sharpFilt(filtRadius+1,filtRadius+1) = 1;
sharpFilt = sharpFilt - gaussFilt;
sharpFilt = amount*sharpFilt;
sharpFilt(filtRadius+1,filtRadius+1) = sharpFilt(
    filtRadius+1,filtRadius+1) + 1;
B = imfilter(a,sharpFilt);
figure
ShowImage(B,'Sharpened Image');//IPD toolbox
```

check Appendix AP 2 for dependency:

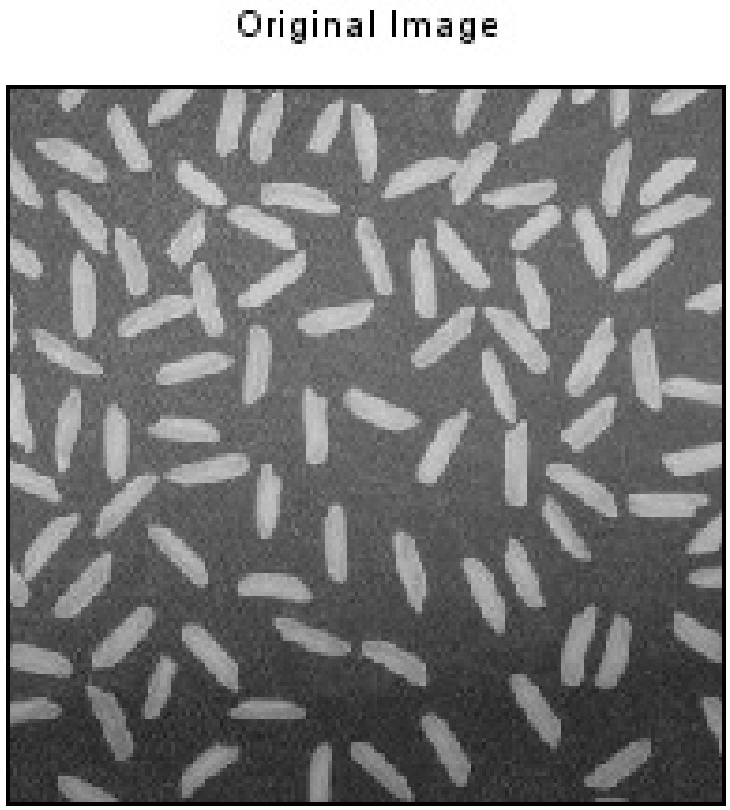


Figure 3.3: Exp3b

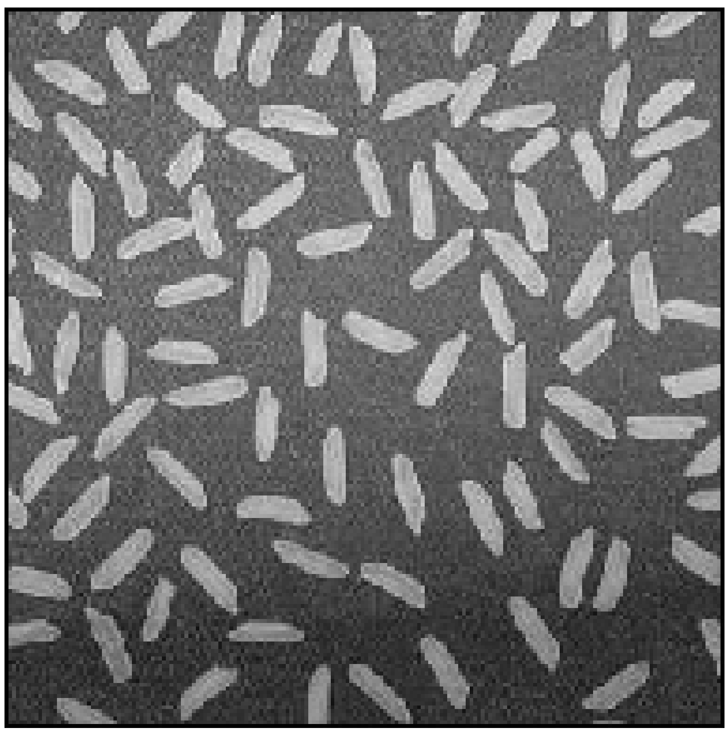


Figure 3.4: Exp3b

rice.jpg

Experiment: 4

Program for lossless Image Compression.

Scilab code Solution 4.1 Exp4

```
// Lossless Image Compression- Implementation of arithmetic coding for images
//Note 1: In order to run this program download Huffman toolbox from
//scilab atoms
//Note 2: The Huffman atom is used to encode images
```

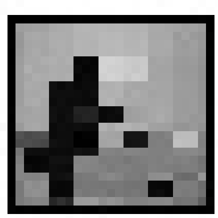


Figure 4.1: Exp4

```
of small size only
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clear;
clc;
close;
a = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\cameraman.jpeg');
A = imresize(a,[16 16]); //Only Image of small size is possible to call huffcode
B = size(A);
A=A(:).';
A = double(A);
[QT,QM]=huffcode(A); //Huffman Encoding
disp('compressed Bit sequence:');
disp(QT);
disp('Code Table:');
disp(QM);
// Now, the reverse operation
C = huffdeco(QT,QM); //Huffman Decoding
for i=1:B(1)
    E(i,1:B(2))= C((i-1)*B(2)+1:i*B(2));
end
D = E';
E = imresize(D,[32,32]);
figure
ShowImage(a,'Original cameraman Image 256x256')
figure
ShowImage(E,'Reconstructed cameraman Image 32x32');
```

check Appendix AP 1 for dependency:

`cameraman.jpeg`

Experiment: 5

Program for lossy Image Compression.

Scilab code Solution 5.1 Exp5

```
//Lossy Image Compression-Block Truncation Coding
//Note: Details of scilab software version and OS
    version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
clc;
clear;
close;
function out_put = btcimage(in_put,block_size)
    //Note: Details of scilab software version and
        OS version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
X= imread(in_put);
Y=imfinfo(in_put);
```

```
K=block_size;
X1=double(X);
y1=size(X);
n=y1(1);
m=y1(2);
k=1;l=1;


    if (Y.ColorType=='grayscale')

    //                      IMAGE ENCODING

    //
    //                   FOR GRAY SCALE IMAGES
    //
        figure(1)
        ShowImage(X,'Original')
        title('ORIGINAL');
        for i=1:K:n
            for j=1:K:m
                tmp([1:K],[1:K])=X1([i:i+(K-1)],[j:j+(K-1)]);
                mn=mean(mean(tmp));
                tmp1([i:i+(K-1)],[j:j+(K-1)])=tmp>mn;
                Lsmat=(tmp<mn);
                Mrmat=(tmp>=mn);
                Lsmn=sum(sum(Lsmat));
                Mrmn=sum(sum(Mrmat));
                Mu(k)=sum(sum(Lsmat.*tmp))/(Lsmn+.5);k=k+1;
                Mi(l)=sum(sum(Mrmat.*tmp))/Mrmn;l=l+1;
            end
        end
        figure(2)
        ShowImage(tmp1,'Encoded Image')
        title('ENCODED');
```

```

        //                                IMAGE DECODING

        k=1;l=1;
        for i=1:K:n
            for j=1:K:m
                tmp21([1:K],[1:K])=tmp1([i:i+(K-1)],[j:j+(K-1)]);
                tmp22=(tmp21*round(Mu(k)));k=k+1;
                tmp21=((tmp21==0)*round(Mi(l)));l=l+1;
                tmp21=tmp21+tmp22;
                out_put([i:i+(K-1)],[j:j+(K-1)])=tmp21;
            end
        end
        figure(3)
        ShowImage(uint8(out_put),'Decoded Image')
        title('DECODED');

    //
    //                              FOR COLORED IMAGES
    //

    elseif (Y.ColorType=='truecolor')
        R=X(:,:,1);
        G=X(:,:,2);
        B=X(:,:,3);
        //                              IMAGE ENCODING
        figure(1)
        ShowColorImage(X,'Original')
        title('ORIGINAL');
        for b=1:3
            for i=1:K:n
                for j=1:K:m
                    tmp([1:K],[1:K])=X1([i:i+(K-1)],[j:j+(K-1)],b);
                    mn=mean(mean(tmp));
```

```
                    tmp1([i:i+(K-1)],[j:j+(K-1)],b)=tmp>mn;
                    Lsmat=(tmp<mn);
                    Mrmat=(tmp>=mn);
                    Lsmn=sum(sum(Lsmat));
                    Mrmn=sum(sum(Mrmat));
                    Mu(b,k)=sum(sum(Lsmat.*tmp))/(Lsmn+.5);k=k+1;
                    Mi(b,l)=sum(sum(Mrmat.*tmp))/Mrmn;l=l+1;
                end
            end
        end
    end
endfunction

//MAIN PROGRAM
I = 'C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\cameraman.jpeg';
block_size = 2;
//exec('btcimage.sci')
//exec('C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\btcimage.sci');
out_put = btcimage(I,block_size);
```

check Appendix AP 1 for dependency:

`cameraman.jpeg`

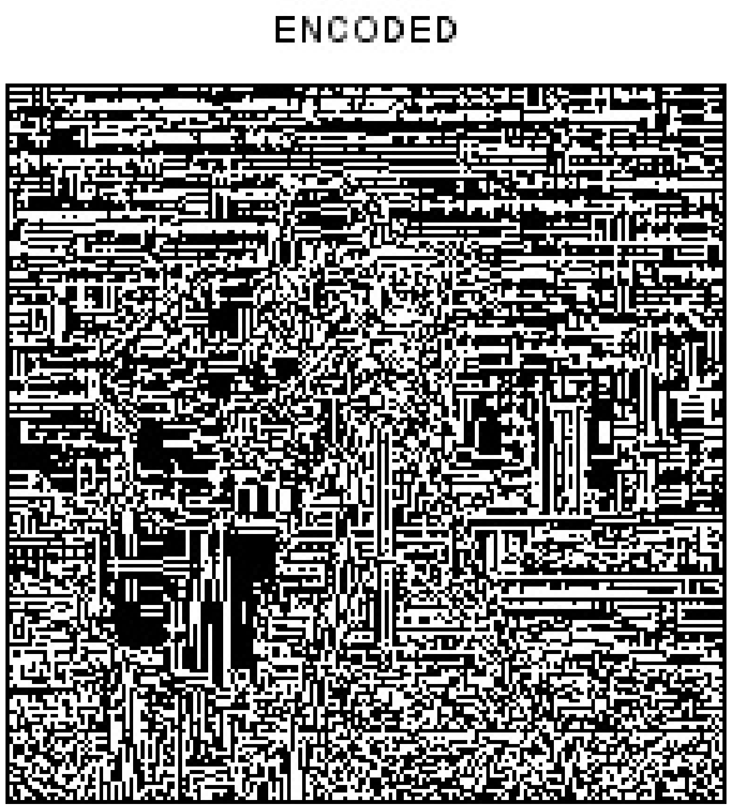


Figure 5.1: Exp5

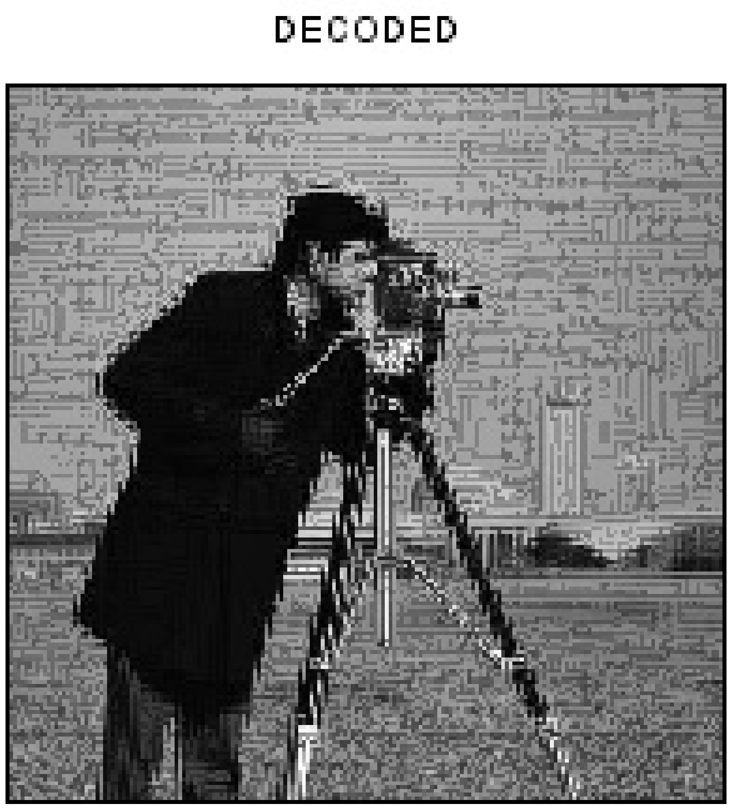


Figure 5.2: Exp5

Experiment: 6

Program for generation and Manipulation of signal.

Scilab code Solution 6.1 Exp6a

```
//Caption: Program to generate and plot different
   basic sequences
clear all;
clc;
close;
//Generation of Unit Impulse signal
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),1,zeros(1,L)];

b = gca();
b.y_location = "middle";
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =4;
xtitle('Graphical Representation of Unit Sample
   Sequence','n','x[n]');
//Generation of Unit Step Signal
L = 10;  //Upperlimit
```

```
t = -L:L;
x = [zeros(1,L),ones(1,L+1)];
figure(1)
subplot(2,1,1)
a=gca();
a.thickness =2;
a.y_location = "middle";
plot2d2(t,x)
xtitle('Graphical Representation of Unit Step Signal','t','x(t)');
//Generation of Unit Step Sequence
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
subplot(2,1,2)
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Unit Step Sequence','n','x[n]');
//Generation of Ramp Sequence
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
figure(2)
subplot(2,1,1)
b = gca();
b.y_location = 'middle';
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Discrete Unit Ramp Sequence','n','x[n]');
//Generation of Ramp Signal
L = 4;  //Upperlimit
t = -L:L;
x = [zeros(1,L),0:L];
```

```
subplot(2,1,2)
b = gca();
b.y_location = 'middle';
plot2d(n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Discrete Unit Ramp Sequence','t','x(t)');
//Generation of Exponentially Increasing signal
a =1.5;
n = 0:10;
x = (a)^n;
figure(3)
subplot(2,1,1)
a=gca();
a.thickness = 2;
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential Increasing Signal','n','x[n]');
//Generation of Exponentailly Decreasing Signal
a =0.5;
n = 0:10;
x = (a)^n;
subplot(2,1,2)
a=gca();
a.thickness = 2;
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential Decreasing Signal','n','x[n]');
```

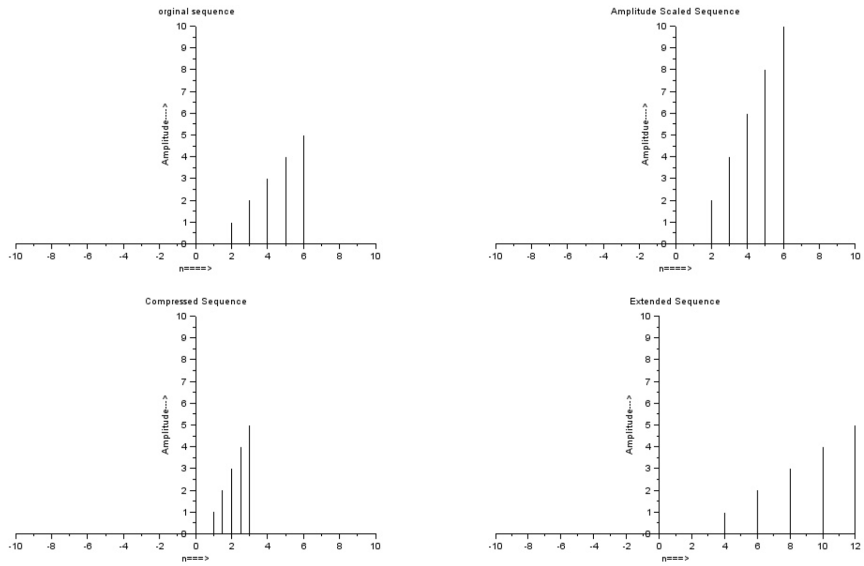


Figure 6.1: Exp6a

Scilab code Solution 6.2 Exp6b

```
//Caption: Program to Demonstrate the signal Folding
clc;
clear;
x = input('Enter the input sequence:=');
m = length(x);
lx = input('Enter the starting point of original
    signal=');
hx = lx+m-1;
n = lx:1:hx;
subplot(2,1,1)
a = gca();
a.x_location = "origin";
```

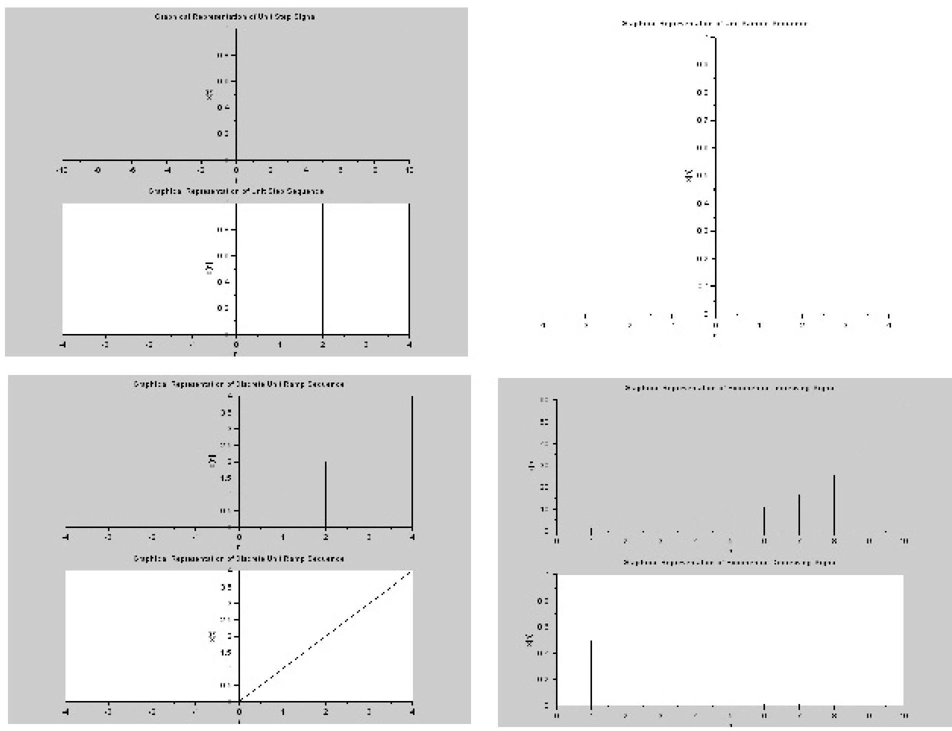


Figure 6.2: Exp6a

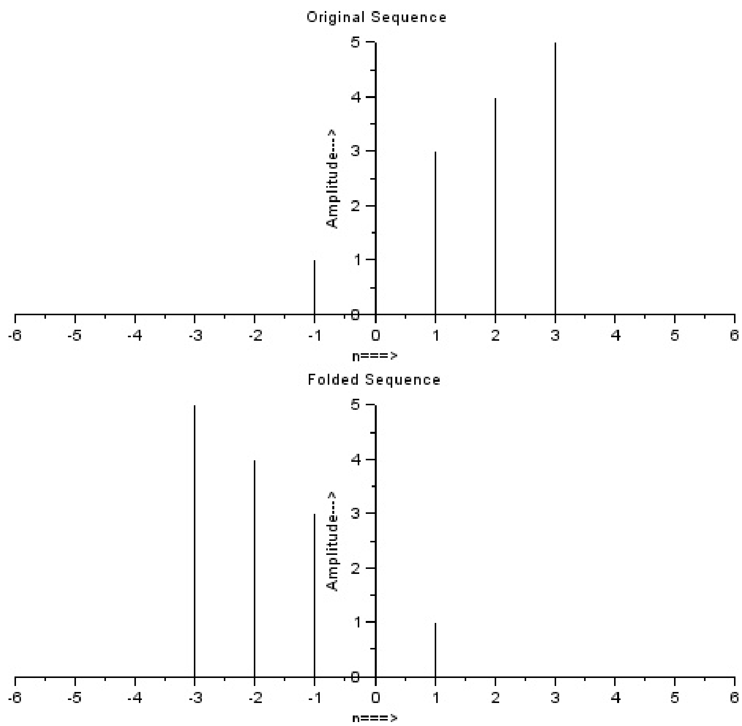


Figure 6.3: Exp6b

```
a.y_location = "origin";
a.data_bounds = [-5,0;5,5];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Original Sequence')
subplot(2,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-5,0;5,5];
plot2d3(-n,x)
xlabel('n===>')
ylabel('Amplitude--->')
title('Folded Sequence')
//Example

//Enter the input sequence:=[1,2,3,2,5]
//
//Enter the starting point of original signal=-1
```

Scilab code Solution 6.3 Exp6c

```
//Caption: Program to demonstrate the Amplitude & Time Scaling of a signal
clc;
clear;
x = input('Enter input Sequence:=');
m = length(x);
lx = input('Enter starting point of original signal:=')
hx = lx+m-1;
n = lx:1:hx;
```

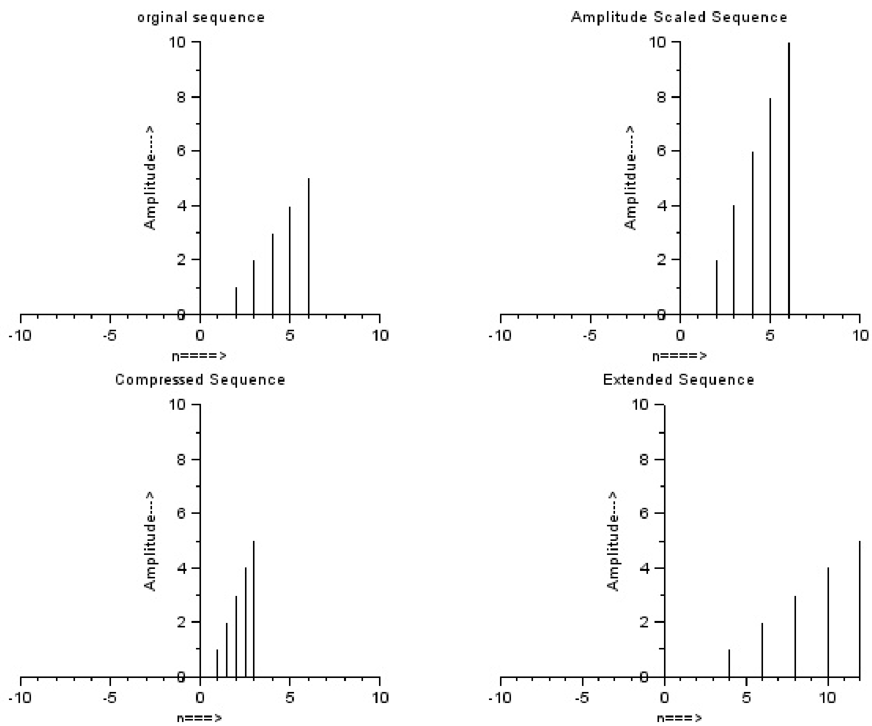


Figure 6.4: Exp6c

```
subplot(2,2,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n=====>')
ylabel('Amplitude------->')
title('orginal sequence')
//Amplitude Scaling
a = input('Amplitude Scaling Factor:=')
y =a*x;
subplot(2,2,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,y)
xlabel('n=====>')
ylabel('Amplitdue------->')
title('Amplitude Scaled Sequence')
//Time Scaling-Compression
C = input('Enter Compression factor-Time Scaling factor')
n = lx/C:1/C:hx/C;
subplot(2,2,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n====>')
ylabel('Amplitude------>')
title('Compressed Sequence')
//Time Scaling-Expansion
d = input('Enter Extension factor-Time Scaling factor')
n = lx*d:d:hx*d;
```

```
subplot(2,2,4)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude---->')
title('Extended Sequence')
//Example
//Enter input Sequence:=[1,2,3,4,5]
//
//Enter starting point of original signal:= 2
//
//Amplitude Scaling Factor:= 2
//
//Enter Compression factor–Time Scaling factor 2
//
//Enter Extension factor–Time Scaling factor 2
```

Scilab code Solution 6.4 Exp6d

```
//Caption:Program to demonstrate the shifting of the
    discrete time signal
clc;
clear;
close;
x = input('Enter the input sequence:=')
m = length(x);
lx = input('Enter the starting point of original
    signal:=')
hx = lx+m-1;
n = lx:1:hx;
```

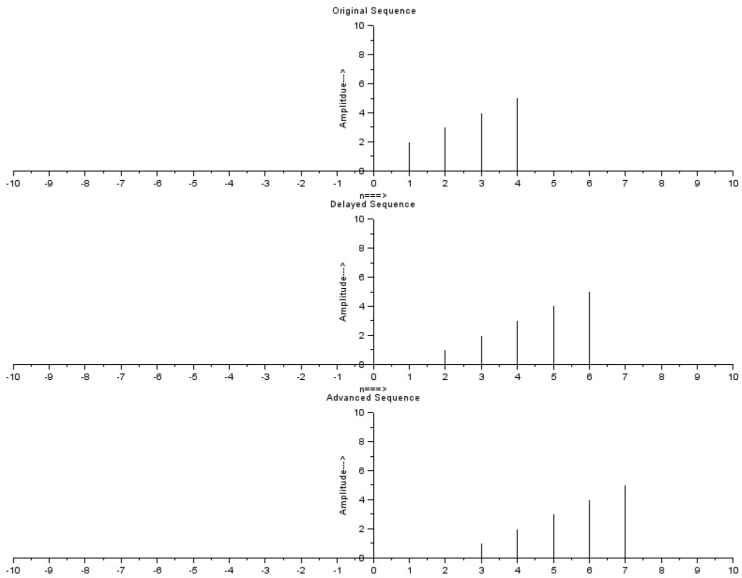


Figure 6.5: Exp6d

```
subplot(3,1,1)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x);
xlabel('n===>')
ylabel('Amplitdue--->')
title('Original Sequence')
//
d = input('Enter the delay:=')
n = lx+d:1:hx+d;
subplot(3,1,2)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
```

```
ylabel('Amplitude——>')
title('Delayed Sequence')
//
a = input('Enter the advance:=')
n = lx-a:1:hx-a;
subplot(3,1,3)
a = gca();
a.x_location = "origin";
a.y_location = "origin";
a.data_bounds = [-10,0;10,10];
plot2d3('gnn',n,x)
xlabel('n===>')
ylabel('Amplitude——>')
title('Advanced Sequence')
//Example
//Enter the input sequence:=[1,2,3,4,5]
//
//Enter the starting point of original signal:=0
//
//Enter the delay:=2
//
//Enter the advance:=3
```

Experiment: 7

Program for Discrete Fourier Transform

Scilab code Solution 7.1 Exp7

```
//Note: Details of scilab software version and OS
   version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//5.PROGRAM TO IMPLEMENT DISCRETE FOURIER TRANSFORM
//DFT
clc;
close;
clear all;
N=input('Howmany point DFT  do you want?');
x2=input('Enter the sequence=');
n2=length(x2);
c= zeros(N);
x2=[x2 zeros(1,N-n2)];
 for k=1:N
```

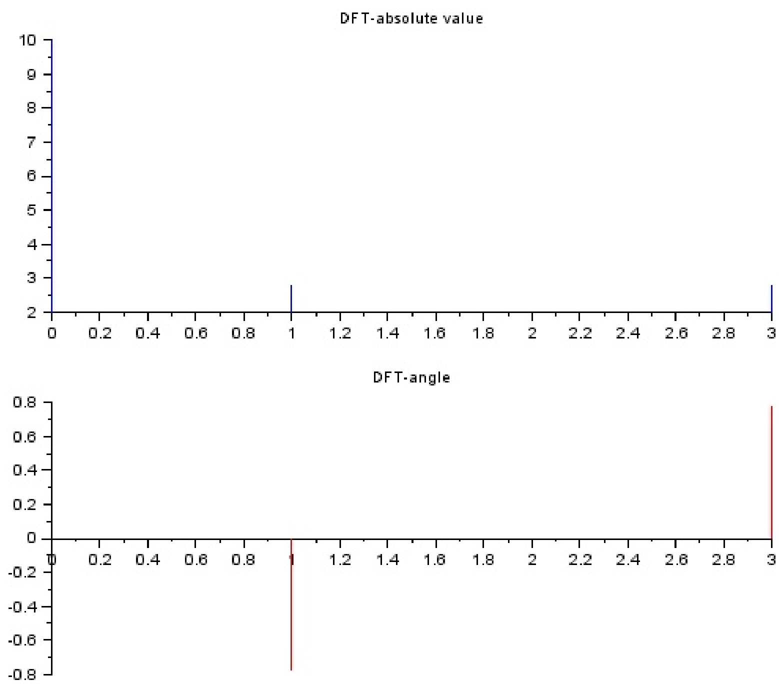


Figure 7.1: Exp7

```
    for n=1:N
        w=exp((-2*%pi*%i*(k-1)*(n-1))/N);
        x(n)=w;
        c(k,n)=x(n);
    end

end
r=x2*c;
//plotting magnitude and angle
subplot(2,1,1)
plot2d3('gnn',0:N-1,abs(r),2);
title('DFT—absolute value');
subplot(2,1,2)
a = gca()
plot2d3('gnn',0:N-1,atan(imag(r)./(real(r)+0.0001))
    ,5);
a.x_location="origin";
title('DFT—angle');
disp(r,'Discrete Fourier Transform Result')
//RESULT
//Example 1
//Howmany point DFT  do you want? 4
//Enter the sequence=[1,2,3,4]
//Discrete Fourier Transform Result
//     10.   - 2. + 2.i   - 2. - 9.797D-16i   - 2. - 2.i
//
//Example 2
//Howmany point DFT  do you want?8
//Enter the sequence=[1,1,1,1,1,1,1,1]
//Discrete Fourier Transform Result
//            column 1 to 5
//
//     8.   - 5.551D-16 + 2.220D-16i   - 4.286D-16 -
    4.441D-16i   - 2.220D-16 + 8.882D-16i   - 4.899D-16
    i
//
//           column 6 to 8
//
```

```
//  - 2.109D-15 - 1.221D-15i   - 2.933D-15 - 6.661D
   -16i      3.553D-15 + 1.110D-15i
//
//Example 3
//Howmany point DFT  do you want? 8
//Enter the sequence= [0,1,2,3,4,5,6,7]
//Discrete Fourier Transform Result
//
//
//          column 1 to 7
//
//     28.   - 4. + 9.6568542i   - 4. + 4.i   - 4. +
   1.6568542i   - 4. - 3.429D-15i   - 4. - 1.6568542i
    - 4. - 4.i
//
//          column 8
//
//  - 4. - 9.6568542i
//
```

Experiment: 8

Simulation of FIR Filters

Scilab code Solution 8.1 Exp8

```
//Caption: To Design an Low Pass FIR Filter
//Filter Length =5, Order = 4
//Window = Rectangular Window
clc;
clear;
xdel(winsid());
fc = input("Enter Analog cutoff freq. in Hz=")
fs = input("Enter Analog sampling freq. in Hz=")
M = input("Enter order of filter =")
w = (2*%pi)*(fc/fs);
disp(w,'Digital cutoff frequency in radians.cycles/samples');
wc = w/%pi;
disp(wc,'Normalized digital cutoff frequency in cycles/samples');
[wft,wfm,fr]=wfir('lp',M+1,[wc/2,0],'re',[0,0]);
disp(wft,'Impulse Response of LPF FIR Filter:h[n]=');
//Plotting the Magnitude Response of LPF FIR Filter
```

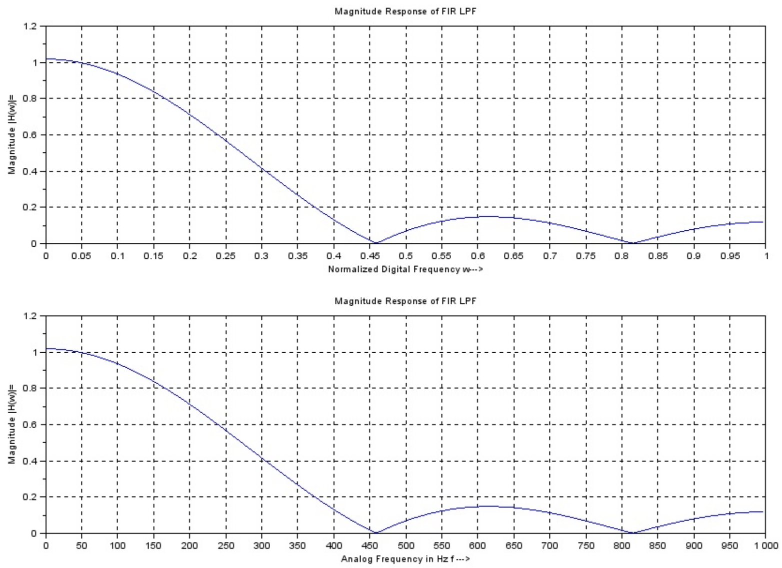


Figure 8.1: Exp8

```
subplot(2,1,1)
plot(2*fr,wfm)
xlabel('Normalized Digital Frequency w--->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)
subplot(2,1,2)
plot(fr*fs,wfm)
xlabel('Analog Frequency in Hz f --->')
ylabel('Magnitude |H(w)|=')
title('Magnitude Response of FIR LPF')
xgrid(1)
//Example
//Enter Analog cutoff freq. in Hz= 250
//
//Enter Analog sampling freq. in Hz= 2000
//
//Enter order of filter = 4
//
```

```
// Digital cutoff frequency in radians.cycles/
   samples
//
//    0.7853982
//
// Normalized digital cutoff frequency in cycles/
   samples
//
//    0.25
//
// Impulse Response of LPF FIR Filter:h[n]=
//
//    0.1591549    0.2250791    0.25    0.2250791
      0.1591549
```

Experiment: 9

Generation and Quantization of Binary Numbers

Scilab code Solution 9.1 Exp9

```
//Note: Details of scilab software version and OS version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//1.Quantization and sampling

//Quantize a signal to n bits.  This code assumes the signal is between -1
//and +1.
clc;
clear all;
close;
n=8;                          //Number of bits;
m= 120;                        //Number of samples;
t = 2*%pi*[0:(m-1)]/m;
```

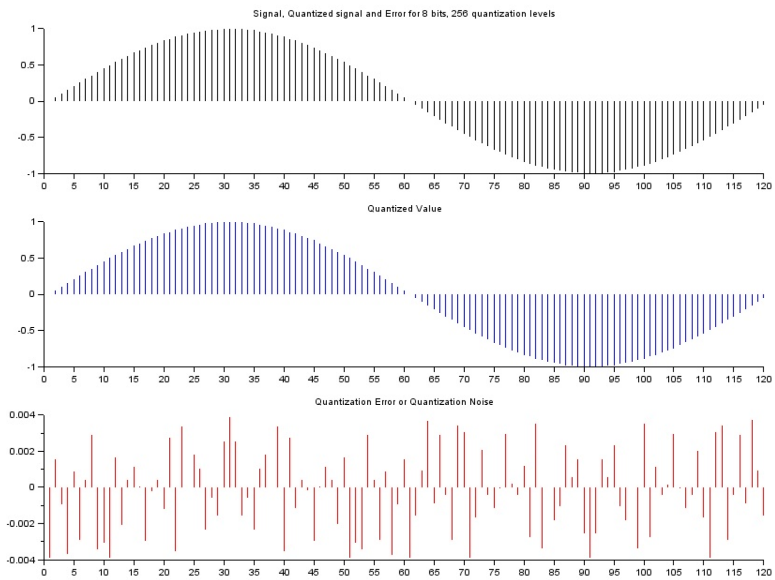


Figure 9.1: Exp9

```
x=sin(t);    //signal between -1 and 1.
                                    //Trying "sin()" instead of "sawtooth"
                                    //results in more interesting error (to the
                                    //extent that error is interesting).
x(find(x>=1))=(1-%eps);      //Make signal from -1 to just less than 1.
xq=floor((x+1)*2^(n-1));     //Signal is one of 2^n int values (0 to 2^n-1)
xq=xq/(2^(n-1));             //Signal is from 0 to 2 (quantized)
xq=xq-(2^(n)-1)/2^(n);       //Shift signal down (rounding)

xe=x-xq;                     //Error
```

```
subplot(3,1,1)
plot2d3('gnn',1:length(x),x);
title(sprintf('Signal, Quantized signal and Error for %g bits, %g quantization levels',n,2^n));
disp(x,'exact value')
subplot(3,1,2)
plot2d3('gnn',1:length(xq),xq,2);
title('Quantized Value')
disp(xq,'Quantized value')
subplot(3,1,3)
plot2d3('gnn',1:length(xe),xe,5);
title('Quantization Error or Quantization Noise')
disp(xe,'Quantization error or noise')
```

Experiment: 10

Introduction to Simulink Signal Analysis

Scilab code Solution 10.1 Exp10a

```
//Step response of discrete time systems
//Refer Exp10a.xcos file for simulink analysis
```

This code can be downloaded from the website wwww.scilab.in

Scilab code Solution 10.2 Exp10b

```
//Step response of Continuous time systems
//Refer Exp10b.xcos file for simulink analysis
```

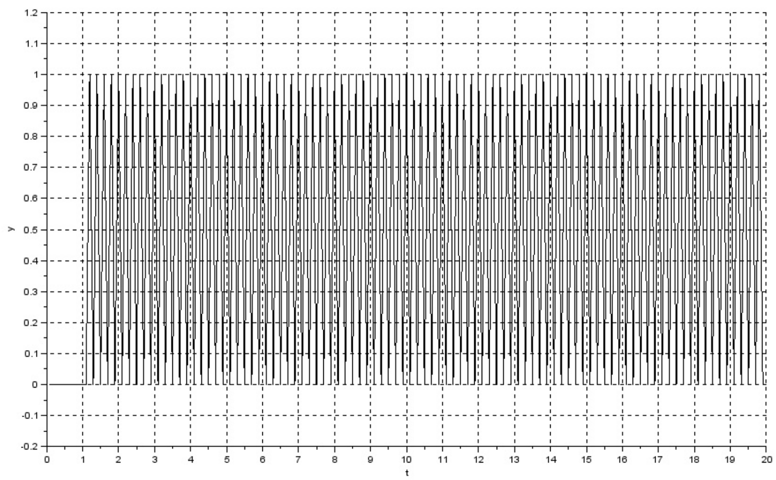


Figure 10.1: Exp10a

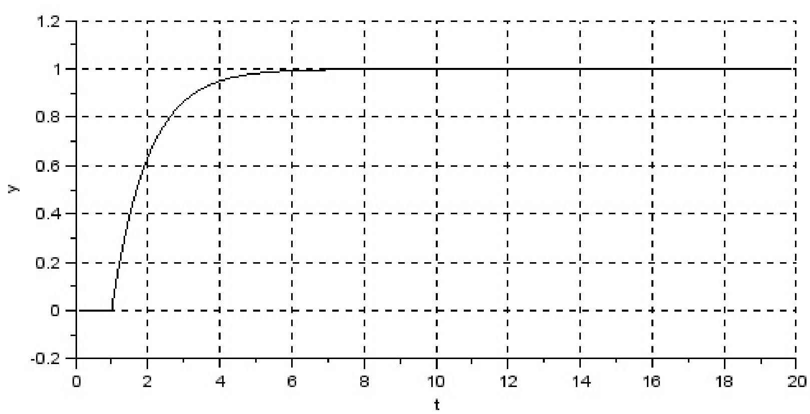


Figure 10.2: Exp10b

This code can be downloaded from the website wwww.scilab.in

Experiment: 11

Design and analysis of Butterworth Filter

Scilab code Solution 11.1 Exp11

```
//Note: Details of scilab software version and OS
    version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
clc;
clear all;
close;
n = 6; //filter order
Wn = [2.5e6,29e6]/500e6; //normalized cutoff
    frequencies[lower,upper]
ftype = 'bp'; //bandpass filter
fdesign = 'butt'; //Butterworth Filter
delta =[];
hz=iir(n,ftype,fdesign,Wn/2,delta)
[p,z,g]=iir(n,ftype,fdesign,Wn/2,delta)
```

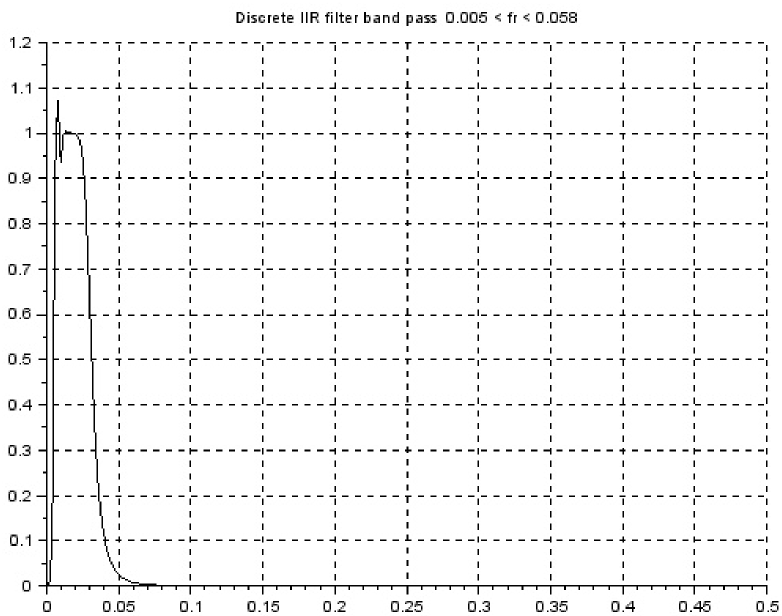


Figure 11.1: Exp11

```
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm');
xtitle('Discrete IIR filter band pass  0.005 < fr <
   0.058 ',' ',' ');
xgrid(1)
//Result

//-->hz(2)
// ans  =
//
//                              2               4
//     0.0000002 - 0.0000015z + 0.0000037z
//                            6               8
//             - 0.0000049z + 0.0000037z
//                            10              12
//             - 0.0000015z + 0.0000002z
//
//-->hz(3)
// ans  =
//
//                                              2
//     0.5250468 - 6.6287407z + 38.377802z
//                            3               4
//             - 134.73451z + 319.45814z
//                            5               6
//             - 538.91189z + 663.25134z
//                            7              8
//             - 600.03003z + 396.0233z
//                             9              10
//             - 185.96443z + 58.974503z
//                            11   12
//             - 11.340535z + z
//
//-->hz(1)
// ans  =
//
//!r  num  den  dt  !
//
```

```
//-->p
// p  =
//
//     0.9964120 - 0.0154005i
//     0.9892502 - 0.0129619i
//     0.9822499 - 0.0060512i
//     0.9822499 + 0.0060512i
//     0.9892502 + 0.0129619i
//     0.9964120 + 0.0154005i
//     0.9464056 + 0.1686845i
//     0.8907712 + 0.1177110i
//     0.8651786 + 0.0429743i
//     0.8651786 - 0.0429743i
//     0.8907712 - 0.1177110i
//     0.9464056 - 0.1686845i
//
//-->z
// z  =
//
//     1.
//     1.
//     1.
//     1.
//     1.
//     1.
//   - 1.
//   - 1.
//   - 1.
//   - 1.
//   - 1.
//   - 1.
//
//-->g
// g  =
//
//     0.0000002
//
```

Experiment: 12

Impulse response of first order and second order system

Scilab code Solution 12.1 Exp12

```
//Note: Details of scilab software version and OS
    version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
clc;
clear all;
close;
s=poly(0,'s');
//The parameters 1.Angular Position 2. Angular
    Velocity of DC Motors
//are obtained from MATLAB demos file.
Angular_Position =(0.003127*s+0.9815)/(s^2+3.929*s
    +6.343e-05);
Angular_velocity = (1.04*s+0.2756)/(s^2+4.461*s
    +1.096);
```

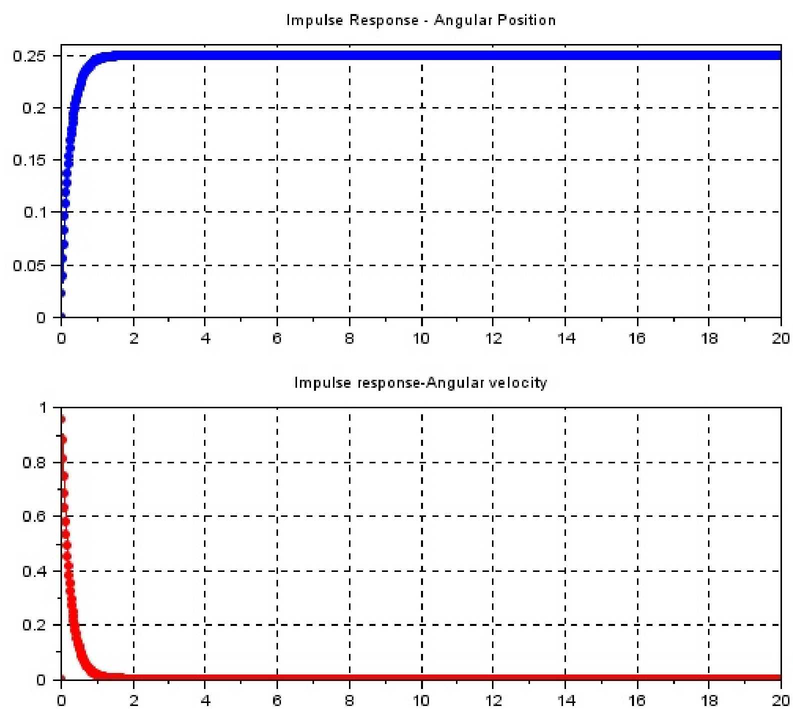


Figure 12.1: Exp12

```
model = [Angular_Position,Angular_velocity];
H1 = model(:,1); //Angular Position
H2 = model(:,2); //Angular velocity
np=20; //number of points
t = 0:0.02:20;
ysd1 = csim('impulse',t,model(:,1));
ysd2 = csim('impulse',t,model(:,2));
subplot(2,1,1)
plot(t,ysd1,'.-b')
title('Impulse Response - Angular Position')
xgrid(1)
subplot(2,1,2)
plot(t,ysd2,'.-r')
title('Impulse response-Angular velocity')
xgrid(1)
disp(model,'Model System Equations =')
disp(ysd1,'Impulse Resposne of Angular Position =')
disp(ysd2,'Impulse Response of Angular velocity=')

//RESULT
//Model System Equations =
//
//      0.9815 + 0.003127s          0.2756 + 1.04s
//    ----------------------     -------------------
//                           2                      2
//    0.0000634 + 3.929s + s     1.096 + 4.461s + s
```

Experiment: 13

Circular convolution of two given sequences.

Scilab code Solution 13.1 Exp13

```
//Note: Details of scilab software version and OS
   version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//3.CIRCULAR CONVOLUTION OF TWO SEQUENCES
clc;
close;
clear all;
a1= input('1st Sequence x:')
b1= input('2nd Sequence h:')
ax=length(a1);
bx=length(b1);
n=max(ax,bx);
n3=ax-bx;
if(n3<=0)
```

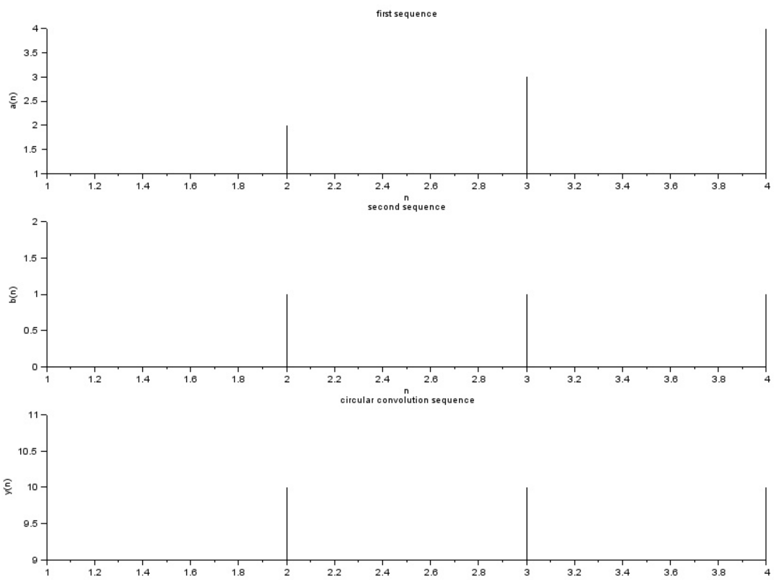


Figure 13.1: Exp13

```
 a1=[a1,zeros(1,-n3)];
else
 b1=[b1,zeros(1,n3)];
end
for r = 1:n
    y(r)=0;
    for i=1:n
        j=r-i+1;
        if (j<=0)
            j=j+n;
        end
        y(r)=y(r)+b1(j)*a1(j);
    end
end
disp(y,'circular convloution result')

subplot(3,1,1);
plot2d3('gnn',a1);
xlabel('n');
```

```
ylabel('a(n)');
title('first sequence');
subplot(3,1,2);
plot2d3('gnn',b1);
xlabel('n');
ylabel('b(n)');
title('second sequence');
subplot(3,1,3);
plot2d3('gnn',y);
xlabel('n');
ylabel('y(n)');
title('circular convolution sequence');

//RESULT
//Example 1
//1st Sequence x:[1,2,3,4]
//2nd Sequence h:[1,1,1,1]
//
// circular convloution result
//
//     10.     10.     10.     10.
//
//Example 2
//1st Sequence x:[1,2,3,4]
//2nd Sequence h:[1,1,1]
//
// circular convloution result
//
//     6.     6.     6.     6.
```

Experiment: 14

Linear convolution of two given sequences.

Scilab code Solution 14.1 Exp14

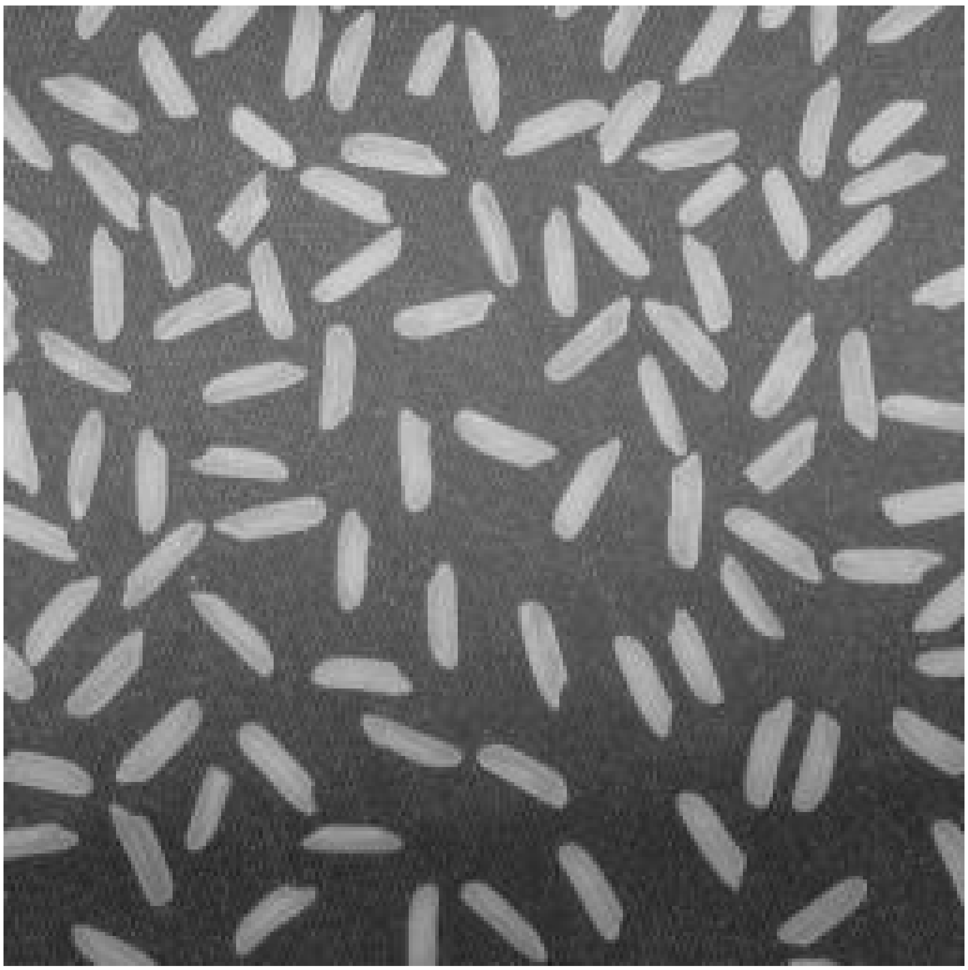
```
//Note: Details of scilab software version and OS version used:
//OS: Windows 7
//Scilab version: 5.4.1
//IPD Atom version:8.3.1-2
//SIVP Atom version:0.5.3.1-2
//4.program for linear convolution of to sequence
clc;
clear all;
close;
x = input('enter the first sequence');
y = input('enter the second sequence');
m = length(x);
n = length(y);
p = m+n-1;
for i=1:p
    q=i;
    k=0;
    for j=1:i
```

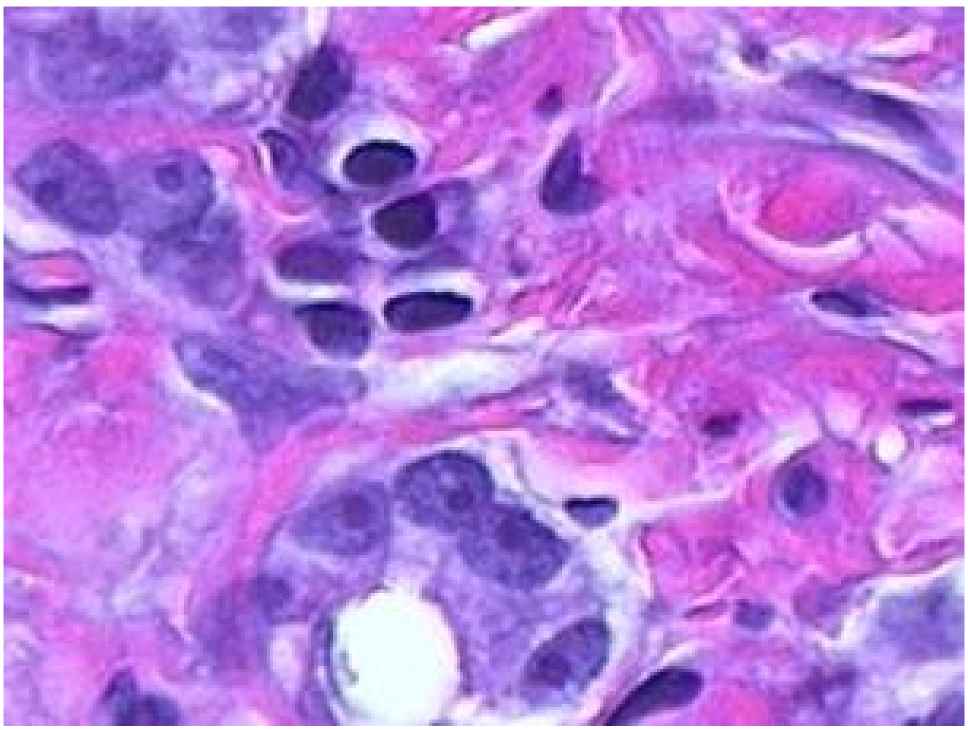
```
        if q>m
            q=q-1;
        elseif j>n
                k=k;
        else k=x(q)*y(j)+k;
            q=q-1;
        end
    end
    z(i)=k;
end
disp(z,'convolution of two sequence is:')

//RESULT
//enter the first sequence [1,2,3]
//enter the second sequence [1,1,1,1]
//
// convolution of two sequence is:
//
//     1.
//     3.
//     6.
//     6.
//     5.
//     3.
//
```

Appendix

Cameraman Image file

 Rice
Image File

 Hestian Colour Image File

Lenna Image File