

Scilab Manual for  
Signal Processing  
by Mrs S. Chaya  
Electronics Engineering  
AIKTC <sup>1</sup>

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

May 5, 2026

<sup>1</sup>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 Scilab Solutions	3
1 Image Sampling and Quantization	6
2 Understanding basic relationship between pixel	13
3 Program for Image sharpening.	22
4 Program for lossless Image Compression.	30
5 Program for lossy Image Compression.	32
6 Program for generation and Manipulation of signal.	38
7 Program for Discrete Fourier Transform	50
8 Simulation of FIR Filters	54
9 Generation and Quantization of Binary Numbers	57
10 Introduction to Simulink Signal Analysis	60
11 Design and analysis of Butterworth Filter	63
12 Impulse response of first order and second order system	67
13 Circular convolution of two given sequences.	70
14 Linear convolution of two given sequences.	73

# List of Experiments

Solution 1.1	Exp1a	6
Solution 1.2	Exp1b	8
Solution 2.1	Exp2a	13
Solution 2.2	Exp2b	14
Solution 3.1	Exp3a	22
Solution 3.2	Exp3b	23
Solution 4.1	Exp4	30
Solution 5.1	Exp5	32
Solution 6.1	Exp6a	38
Solution 6.2	Exp6b	41
Solution 6.3	Exp6c	44
Solution 6.4	Exp6d	47
Solution 7.1	Exp7	50
Solution 8.1	Exp8	54
Solution 9.1	Exp9	57
Solution 10.1	Exp10a	60
Solution 10.2	Exp10b	60
Solution 11.1	Exp11	63
Solution 12.1	Exp12	67
Solution 13.1	Exp13	70
Solution 14.1	Exp14	73
AP 1	Camerman Image file	76
AP 2	Rice Image File	77
AP 3	Hestian Colour Image File	78
AP 4	Lenna Image File	79

# List of Figures

1.1	Exp1a	7
1.2	Exp1a	9
1.3	Exp1b	10
1.4	Exp1b	12
2.1	Exp2a	15
2.2	Exp2a	16
2.3	Exp2b	20
2.4	Exp2b	21
3.1	Exp3a	24
3.2	Exp3a	25
3.3	Exp3b	27
3.4	Exp3b	28
4.1	Exp4	30
5.1	Exp5	36
5.2	Exp5	37
6.1	Exp6a	41
6.2	Exp6a	42
6.3	Exp6b	43
6.4	Exp6c	45
6.5	Exp6d	48
7.1	Exp7	51
8.1	Exp8	55
9.1	Exp9	58

10.1 Exp10a . . . . .	61
10.2 Exp10b . . . . .	61
11.1 Exp11 . . . . .	64
12.1 Exp12 . . . . .	68
13.1 Exp13 . . . . .	71

# Experiment: 1

## Image Sampling and Quantization

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

cameraman.jpeg

Scilab code Solution 1.1 Expla

```
1 //Image Quantization
2 clear;
3 clc;
4 I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\cameraman.jpeg');
5 quanta = 50;
6 J = double(I)/255;
7 J = uint8(J*quanta);
8 J = double(J)/quanta;
9 figure
10 ShowImage(I, 'Original Image')
11 figure
12 ShowImage(J, 'Quantized Image')
```

---



Figure 1.1: Expla

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

cameraman.jpeg

### Scilab code Solution 1.2 Exp1b

```
1 //Image Sampling
2 clear;
3 clc;
4 I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\cameraman.jpeg');
5 J = imresize(I,0.5); //Reducing the sampling rate
6 K1 = imresize(J,2,'nearest'); //Increasing the
    sampling rate
7 K2 = imresize(J,2,'bilinear');
8 K3 = imresize(J,2,'bicubic');
9 figure
10 ShowImage(I,'Original Image')
11 figure
12 ShowImage(J,'Reducing the Sampling Rate by 2')
13 figure
14 ShowImage(K1,'Increasing the Sampling Rate by 2
    nearest neighbour method')
15 figure
16 ShowImage(K2,'Increasing the Sampling Rate by 2
    bilinear method')
17 figure
18 ShowImage(K3,'Increasing the Sampling Rate by 2
    bicubic method')
```

---



Figure 1.2: Expla



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

```
1 //Image Arithmetic –division , multiplication , image
  subtraction and image addition
2 clc;
3 clear;
4 close;
5 I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab
  \scilab\cameraman.jpeg'); //SIVP toolbox
6 J = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
  scilab\rice.jpg'); //SIVP toolbox
7 IMA = imadd(I,J); //SIVP toolbox
8 figure
9 ShowImage(IMA, 'Image Addition') //IPD toolbox
```

```

10 IMS = imabsdiff(I,J); //SIVP toolbox
11 figure
12 ShowImage(IMS, 'Image Subtraction '); //IPD toolbox
13 IMD = imdivide(I,J); //SIVP toolbox
14 IMD = imdivide(IMD,0.01); //SIVP toolbox
15 figure
16 ShowImage(uint8(IMD), 'Image Division '); //IPD toolbox
17 IMM = immultiply(I,I); //SIVP toolbox
18 figure
19 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

```

1 //Image Arithmetic– Distance and Connectivity: To
  understand the notion of connectivity
2 //and neighborhood defined for a point in an image.
3 clc;
4 clear;
5 close;
6 //function to convert gray to binary
7 function X = gray2bin(x)
8     xmean = mean2(x);
9     [m,n]= size(x);
10    X = zeros(m,n);
11    for i = 1:m

```



Figure 2.1: Exp2a



Figure 2.2: Exp2a

```

12         for j = 1:n
13             if x(i,j)> xmean then
14                 X(i,j) = 1;
15             end
16         end
17     end
18 endfunction
19 //function to find total length of two dimensional
    matrix
20 function n = numdims(X)
21     n = length(size(X));
22 endfunction
23 ////////////////////////////////////////////////////
24 //Funtion to pad zeros in columns and rows at both
    ends of an binary image
25 function B = padarray(b)
26     //pad zeros in columns and rows at both ends of
        an binary image
27 [m,n] = size(b);
28 num_dims = length(size(b));
29 B = zeros(m+num_dims,n+num_dims);
30 for i = num_dims:m+num_dims-1
31     for j = num_dims:m+num_dims-1
32         B(i,j) = b(i-1,j-1);
33     end
34 end
35 endfunction
36 ////////////////////////////////////////////////////
37 //[1]. Euclidean Distance between images and their
    histograms
38 I = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\lenna.jpg');
39 J = imread('C:\Users\senthilkumar\Desktop\Chaya_Lab\
    scilab\cameraman.jpeg');
40 h_I = CreateHistogram(I); //IPD toolbox
41 h_J = CreateHistogram(J); //IPD toolbox
42 I = double(I);
43 J = double(J);

```

```

44 E_dist_Hist = sqrt(sum((h_I-h_J).^2)); //Euclidean
    Distance between histograms of two images
45 E_dist_images = sqrt(sum((I(:)-J(:)).^2)); //
    Euclidean Distance between two images
46 disp(E_dist_images, 'Euclidean Distance between two
    images ');
47 disp(E_dist_Hist, 'Euclidean Distance between
    histograms of two images ')
48 //[2]. Connectivity – 8 connected to the background
49 //exec(gray2bin)
50 Ibin = gray2bin(I);
51 Jbin = gray2bin(J);
52 //conversion of gray image into binary image
53 conn = [1,1,1;1,1,1;1,1,1]; //8-connectivity
54 //exec('C:\Users\senthilkumar\Desktop\Gautam_PAL_Lab
    \numdims.sci ')
55 num_dims = numdims(I);
56 //exec('C:\Users\senthilkumar\Desktop\Gautam_PAL_Lab
    \padarray.sci ')
57 B = padarray(Ibin);
58 global FILTER_ERODE;
59 StructureElement = CreateStructureElement('square',
    3);
60 B_eroded = MorphologicalFilter(B,FILTER_ERODE,
    StructureElement.Data); //IPD toolbox
61 //note: StructureElement.Data and conn both are same
    values
62 //except that StructureElement.Data is boolean
    either true or false
63 p = B&~B_eroded;
64 [m,n] = size(p);
65 for i = num_dims:m+num_dims-2
66     for j = num_dims:n+num_dims-2
67         pout(i-1,j-1) = p(i,j);
68     end
69 end
70 figure
71 ShowImage(uint8(I), 'Gray Lenna Image')

```

```
72 figure
73 ShowImage(Ibin, 'Binary Lenna Image')
74 figure
75 ShowImage(pout, '8 neighbourhood connectivity in Lenna
    Image')
76 //RESULT
77 //Euclidean Distance between two images
78 //
79 //    19797.433
80 //
81 // Euclidean Distance between histograms of two
    images
82 //
83 //    5770.7
```

---



Figure 2.3: Exp2b



Figure 2.4: Exp2b

# Experiment: 3

## Program for Image sharpening.

Scilab code Solution 3.1 Exp3a

```
1 //Note: Details of scilab software version and OS
   version used:
2 //OS: Windows 7
3 //Scilab version: 5.4.1
4 //IPD Atom version:8.3.1-2
5 //SIVP Atom version:0.5.3.1-2
6 //2.Program to sharpen image
7 //Read image and display it.
8 //For Colour Image
9 clc;
10 clear all;
11 close;
12 a = imread('C:\Users\senthilkumar\Desktop\
   signal_processing_lab\hestian.jpg');
13 ShowColorImage(a, 'Original Image')
14 title('Original Image');
15 //Sharpen the image and display it.
16 //b = imsharpen(a);
17 //figure , imshow(b), title('Sharpened Image');
18
19
```

```

20 radius =1;
21 amount = 0.8000;
22 threshold = 0;
23 // Gaussian blurring filter
24 filtRadius = ceil(radius*2);
25 filtSize = 2*filtRadius + 1;
26 gaussFilt = fspecial('gaussian',[filtSize filtSize],
    radius);
27 // High-pass filter
28 sharpFilt = zeros(filtSize,filtSize);
29 sharpFilt(filtRadius+1,filtRadius+1) = 1;
30 sharpFilt = sharpFilt - gaussFilt;
31 sharpFilt = amount*sharpFilt;
32 sharpFilt(filtRadius+1,filtRadius+1) = sharpFilt(
    filtRadius+1,filtRadius+1) + 1;
33 B = imfilter(a,sharpFilt);
34 figure
35 ShowColorImage(B,'Sharpened Image');

```

---

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

hestian.jpg

### Scilab code Solution 3.2 Exp3b

```

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

```

Original Image

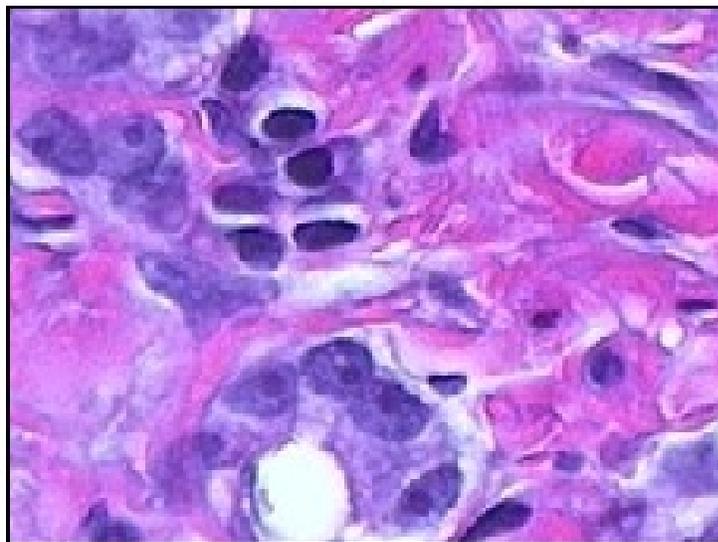


Figure 3.1: Exp3a

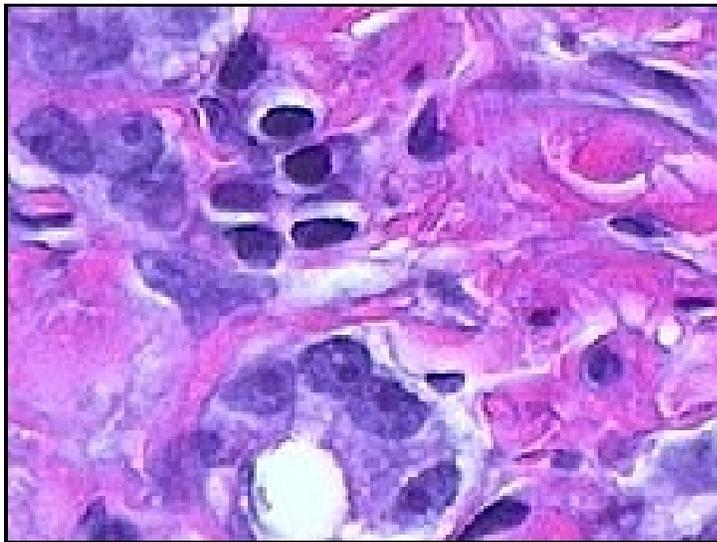


Figure 3.2: Exp3a

```

7 //Read image and display it.
8 //For Gray Image
9 clc;
10 clear all;
11 close;
12 a = imread('C:\Users\senthilkumar\Desktop\
    signal_processing_lab\rice.jpg'); //SIVP toolbox
13 ShowImage(a, 'Original Image') //SIVP toolbox
14 title('Original Image');
15 //Sharpen the image and display it.
16 //b = imsharpen(a);
17 //figure , imshow(b) , title('Sharpened Image');
18
19
20 radius =1;
21 amount = 0.8000;
22 threshold = 0;
23 // Gaussian blurring filter
24 filtRadius = ceil(radius*2);
25 filtSize = 2*filtRadius + 1;
26 gaussFilt = fspecial('gaussian',[filtSize filtSize],
    radius);
27 // High-pass filter
28 sharpFilt = zeros(filtSize,filtSize);
29 sharpFilt(filtRadius+1,filtRadius+1) = 1;
30 sharpFilt = sharpFilt - gaussFilt;
31 sharpFilt = amount*sharpFilt;
32 sharpFilt(filtRadius+1,filtRadius+1) = sharpFilt(
    filtRadius+1,filtRadius+1) + 1;
33 B = imfilter(a,sharpFilt);
34 figure
35 ShowImage(B, 'Sharpened Image'); //IPD toolbox

```

---

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

Original Image

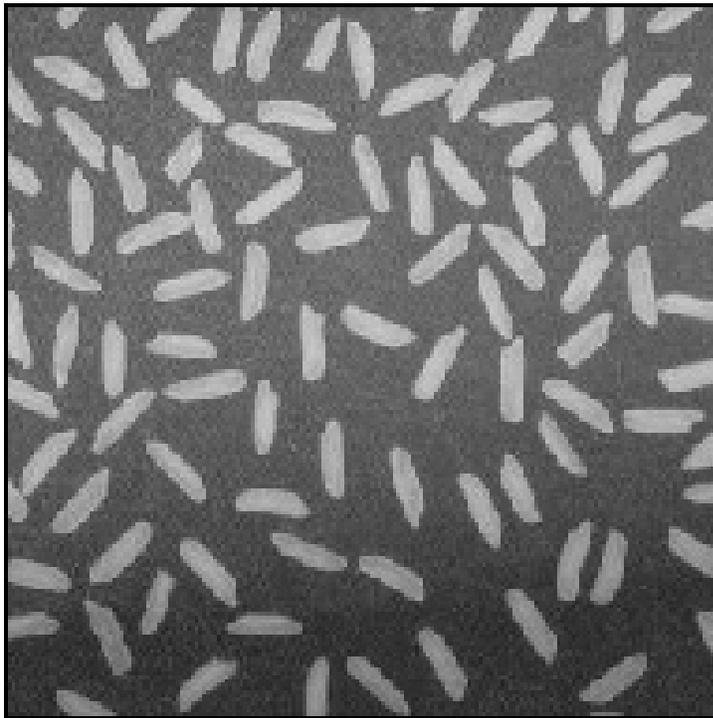


Figure 3.3: Exp3b

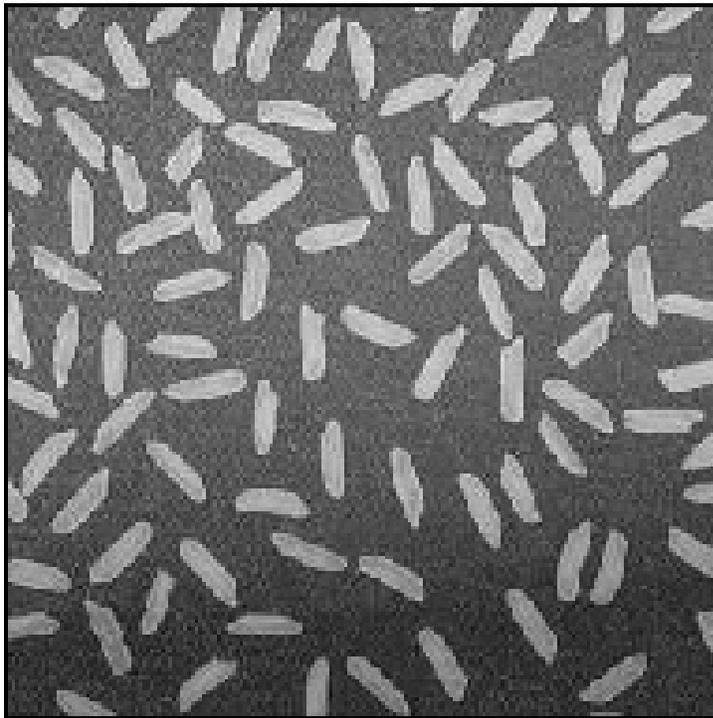


Figure 3.4: Exp3b

rice.jpg

# Experiment: 4

## Program for lossless Image Compression.

Scilab code Solution 4.1 Exp4

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

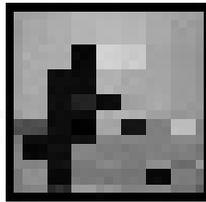


Figure 4.1: Exp4

```

    of small size only
5 //Software version
6 //OS Windows7
7 //Scilab5.4.1
8 //Image Processing Design Toolbox 8.3.1-1
9 //Scilab Image and Video Processing toolbox
    0.5.3.1-2
10 clear;
11 clc;
12 close;
13 a = imread('C:\Users\senthilkumar\Desktop\Chaya-Lab\
    scilab\cameraman.jpeg');
14 A = imresize(a,[16 16]); //Only Image of small size
    is possible to call huffcode
15 B = size(A);
16 A=A(:).';
17 A = double(A);
18 [QT,QM]=huffcode(A); //Huffman Encoding
19 disp('compressed Bit sequence:');
20 disp(QT);
21 disp('Code Table:');
22 disp(QM);
23 // Now, the reverse operation
24 C = huffdeco(QT,QM); //Huffman Decoding
25 for i=1:B(1)
26     E(i,1:B(2))= C((i-1)*B(2)+1:i*B(2));
27 end
28 D = E';
29 E = imresize(D,[32,32]);
30 figure
31 ShowImage(a,'Original cameraman Image 256x256')
32 figure
33 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

```
1 //Lossy Image Compression–Block Truncation Coding
2 //Note: Details of scilab software version and OS
   version used:
3 //OS: Windows 7
4 //Scilab version: 5.4.1
5 //IPD Atom version:8.3.1–2
6 //SIVP Atom version:0.5.3.1–2
7 clc;
8 clear;
9 close;
10 function out_put = btcimage(in_put,block_size)
11     //Note: Details of scilab software version and
       OS version used:
12 //OS: Windows 7
13 //Scilab version: 5.4.1
14 //IPD Atom version:8.3.1–2
15 //SIVP Atom version:0.5.3.1–2
16 X= imread(in_put);
17 Y=imfinfo(in_put);
```

```

18 K=block_size;
19 X1=double(X);
20 y1=size(X);
21 n=y1(1);
22 m=y1(2);
23 k=1;l=1;
24
25
26     if (Y.ColorType=='grayscale')
27
28         //                     IMAGE ENCODING
29
30         //
31         //                     FOR GRAY SCALE IMAGES
32         //
33         figure(1)
34         ShowImage(X,'Original')
35         title('ORIGINAL');
36         for i=1:K:n
37             for j=1:K:m
38                 tmp([1:K],[1:K])=X1([i:i+(K-1)],[j:j
39                     +(K-1)]);
40                 mn=mean(mean(tmp));
41                 tmp1([i:i+(K-1)],[j:j+(K-1)])=tmp>mn
42                     ;
43                 Lsmat=(tmp<mn);
44                 Mrmat=(tmp>=mn);
45                 Lsmn=sum(sum(Lsmat));
46                 Mrmn=sum(sum(Mrmat));
47                 Mu(k)=sum(sum(Lsmat.*tmp))/(Lsmn+.5)
48                     ;k=k+1;
49                 Mi(l)=sum(sum(Mrmat.*tmp))/Mrmn;l=l
50                     +1;
51             end
52         end
53         figure(2)
54         ShowImage(tmp1,'Encoded Image')
55         title('ENCODED');

```

```

52
53 // IMAGE DECODING
54
55 k=1;l=1;
56 for i=1:K:n
57     for j=1:K:m
58         tmp21([1:K],[1:K])=tmp1([i:i+(K-1)
59             ],[j:j+(K-1)]);
60         tmp22=(tmp21*round(Mu(k)));k=k+1;
61         tmp21=((tmp21==0)*round(Mi(l)));l=l
62             +1;
63         tmp21=tmp21+tmp22;
64         out_put([i:i+(K-1)],[j:j+(K-1)])=
65             tmp21;
66     end
67 end
68 figure(3)
69 ShowImage(uint8(out_put),'Decoded Image')
70 title('DECODED');
71
72 // FOR COLORED IMAGES
73
74 elseif (Y.ColorType=='truecolor')
75     R=X(:,:,1);
76     G=X(:,:,2);
77     B=X(:,:,3);
78 // IMAGE ENCODING
79 figure(1)
80 ShowColorImage(X,'Original')
81 title('ORIGINAL');
82 for b=1:3
83     for i=1:K:n
84         for j=1:K:m
85             tmp([1:K],[1:K])=X1([i:i+(K-1)
86                 ],[j:j+(K-1)],b);
87             mn=mean(mean(tmp));

```

```

86         tmp1([i:i+(K-1)],[j:j+(K-1)],b)=
           tmp>mn;
87         Lsmat=(tmp<mn);
88         Mrmat=(tmp>=mn);
89         Lsmn=sum(sum(Lsmat));
90         Mrmn=sum(sum(Mrmat));
91         Mu(b,k)=sum(sum(Lsmat.*tmp))/(
           Lsmn+.5);k=k+1;
92         Mi(b,l)=sum(sum(Mrmat.*tmp))/
           Mrmn;l=l+1;
93         end
94     end
95 end
96 end
97 endfunction
98
99 //MAIN PROGRAM
100 I = 'C:\Users\senthilkumar\Desktop\Chaya_Lab\scilab\
      cameraman.jpeg';
101 block_size = 2;
102 //exec('btcimage.sci')
103 //exec('C:\Users\senthilkumar\Desktop\Chaya_Lab\
      scilab\btcimage.sci');
104 out_put = btcimage(I,block_size);

```

---

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

cameraman.jpeg

ENCODED

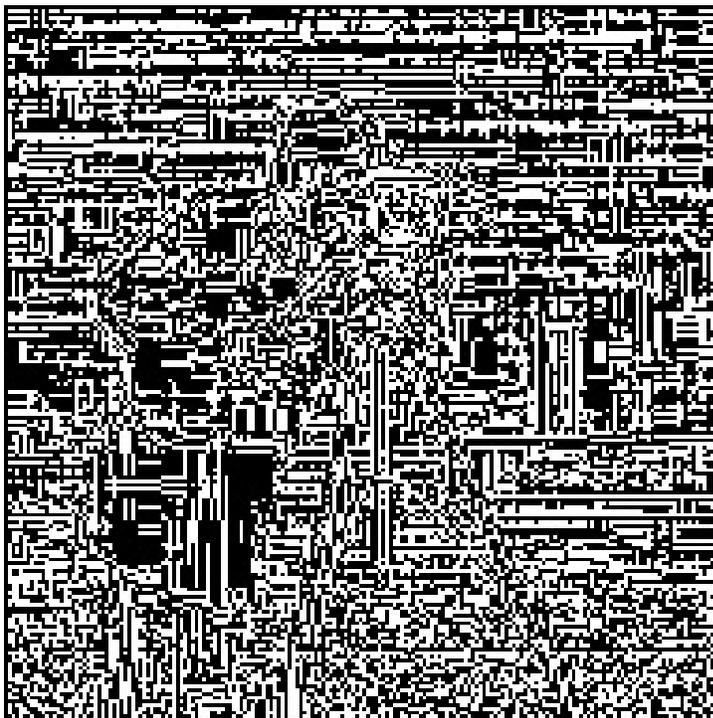


Figure 5.1: Exp5

DECODED



Figure 5.2: Exp5

# Experiment: 6

## Program for generation and Manipulation of signal.

Scilab code Solution 6.1 Exp6a

```
1 //Caption: Program to generate and plot different
  basic sequences
2 clear all;
3 clc;
4 close;
5 //Generation of Unit Impulse signal
6 L = 4; //Upperlimit
7 n = -L:L;
8 x = [zeros(1,L),1,zeros(1,L)];
9
10 b = gca();
11 b.y_location = "middle";
12 plot2d3('gnn',n,x)
13 a=gce();
14 a.children(1).thickness =4;
15 xtitle('Graphical Representation of Unit Sample
  Sequence ', 'n', 'x[n] ');
16 //Generation of Unit Step Signal
17 L = 10; //Upperlimit
```

```

18 t = -L:L;
19 x = [zeros(1,L), ones(1,L+1)];
20 figure(1)
21 subplot(2,1,1)
22 a=gca();
23 a.thickness =2;
24 a.y_location = "middle";
25 plot2d2(t,x)
26 xtitle('Graphical Representation of Unit Step Signal
        ', 't', 'x(t)');
27 //Generation of Unit Step Sequence
28 L = 4; //Upperlimit
29 n = -L:L;
30 x = [zeros(1,L), ones(1,L+1)];
31 subplot(2,1,2)
32 a=gca();
33 a.thickness = 2;
34 a.y_location = "middle";
35 plot2d3('gmn',n,x)
36 xtitle('Graphical Representation of Unit Step
        Sequence', 'n', 'x[n]');
37 //Generation of Ramp Sequence
38 L = 4; //Upperlimit
39 n = -L:L;
40 x = [zeros(1,L), 0:L];
41 figure(2)
42 subplot(2,1,1)
43 b = gca();
44 b.y_location = 'middle';
45 plot2d3('gmn',n,x)
46 a=gca();
47 a.children(1).thickness =2;
48 xtitle('Graphical Representation of Discrete Unit
        Ramp Sequence', 'n', 'x[n]');
49 //Generation of Ramp Signal
50 L = 4; //Upperlimit
51 t = -L:L;
52 x = [zeros(1,L), 0:L];

```

```

53 subplot(2,1,2)
54 b = gca();
55 b.y_location = 'middle';
56 plot2d(n,x)
57 a=gca();
58 a.children(1).thickness =2;
59 xtitle('Graphical Representation of Discrete Unit
        Ramp Sequence', 't', 'x(t)');
60 //Generation of Exponentially Increasing signal
61 a =1.5;
62 n = 0:10;
63 x = (a)^n;
64 figure(3)
65 subplot(2,1,1)
66 a=gca();
67 a.thickness = 2;
68 a.x_location = "origin";
69 a.y_location = "origin";
70 plot2d3('gnn',n,x)
71 xtitle('Graphical Representation of Exponential
        Increasing Signal', 'n', 'x[n]');
72 //Generation of Exponentially Decreasing Signal
73 a =0.5;
74 n = 0:10;
75 x = (a)^n;
76 subplot(2,1,2)
77 a=gca();
78 a.thickness = 2;
79 a.x_location = "origin";
80 a.y_location = "origin";
81 plot2d3('gnn',n,x)
82 xtitle('Graphical Representation of Exponential
        Decreasing Signal', 'n', 'x[n]');

```

---

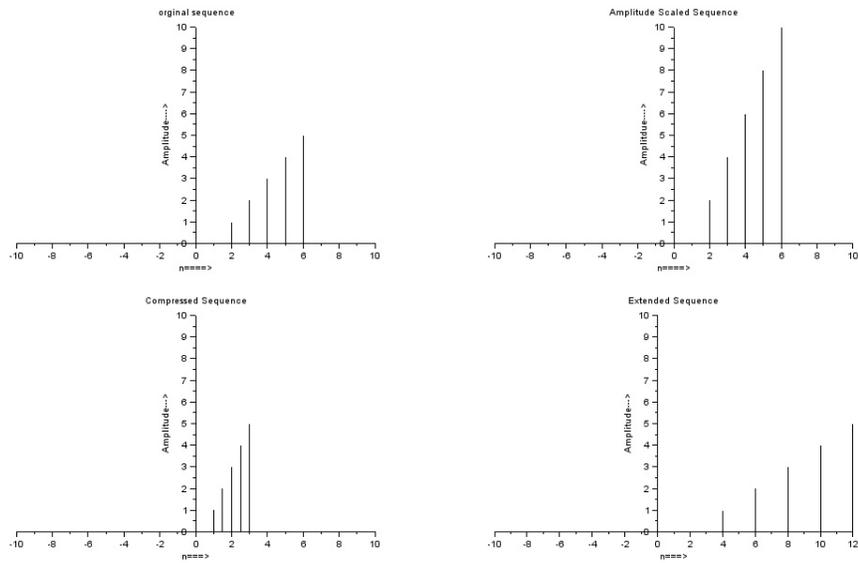


Figure 6.1: Exp6a

### Scilab code Solution 6.2 Exp6b

```

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

```

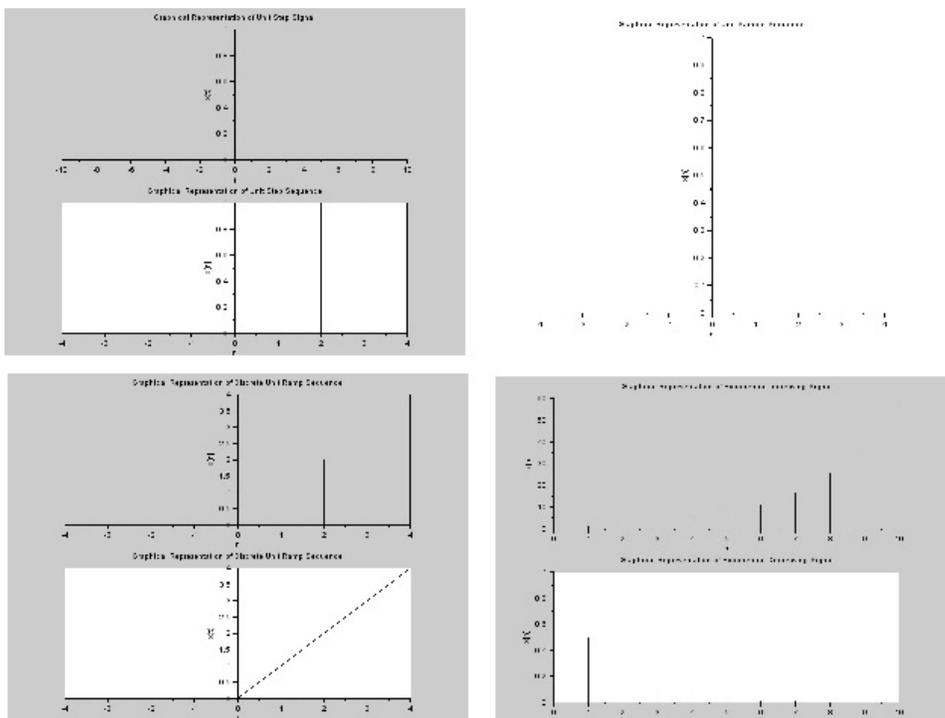


Figure 6.2: Exp6a

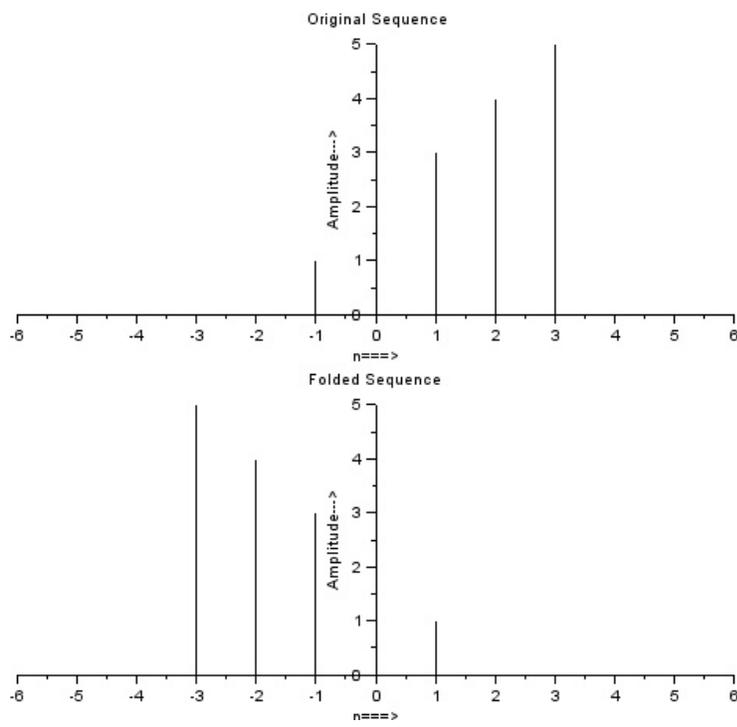


Figure 6.3: Exp6b

```

12 a.y_location = "origin";
13 a.data_bounds = [-5,0;5,5];
14 plot2d3('gnn',n,x)
15 xlabel('n====>')
16 ylabel('Amplitude——>')
17 title('Original Sequence')
18 subplot(2,1,2)
19 a = gca();
20 a.x_location = "origin";
21 a.y_location = "origin";
22 a.data_bounds = [-5,0;5,5];
23 plot2d3(-n,x)
24 xlabel('n====>')
25 ylabel('Amplitude——>')
26 title('Folded Sequence')
27 //Example
28
29 //Enter the input sequence:=[1,2,3,2,5]
30 //
31 //Enter the starting point of original signal=-1

```

---

### Scilab code Solution 6.3 Exp6c

```

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

```

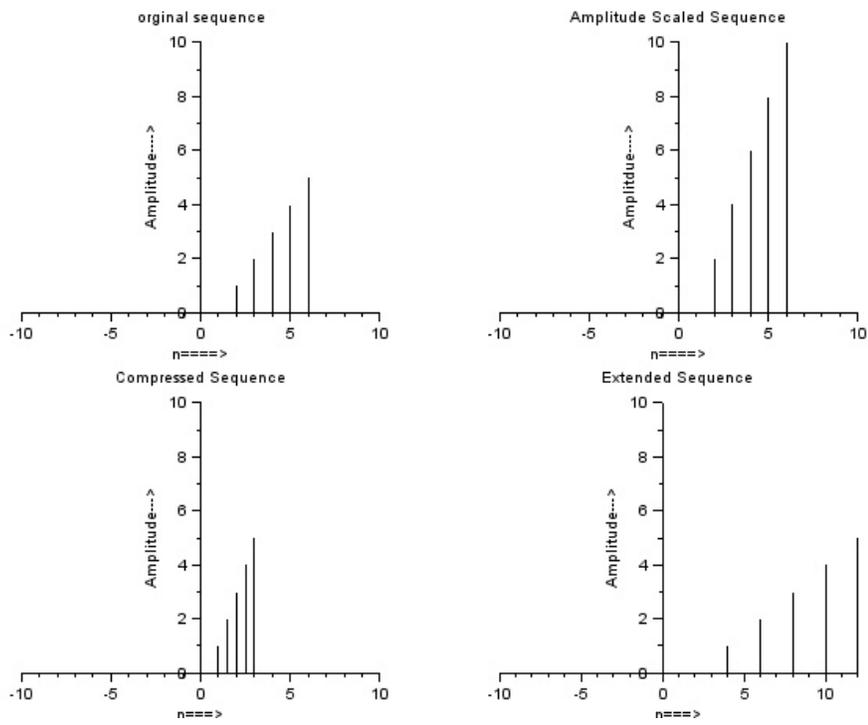


Figure 6.4: Exp6c

```

 9 subplot(2,2,1)
10 a = gca();
11 a.x_location = "origin";
12 a.y_location = "origin";
13 a.data_bounds = [-10,0;10,10];
14 plot2d3('gnn',n,x)
15 xlabel('n====>')
16 ylabel('Amplitude——>')
17 title('original sequence')
18 //Amplitude Scaling
19 a = input('Amplitude Scaling Factor:=')
20 y =a*x;
21 subplot(2,2,2)
22 a = gca();
23 a.x_location = "origin";
24 a.y_location = "origin";
25 a.data_bounds = [-10,0;10,10];
26 plot2d3('gnn',n,y)
27 xlabel('n====>')
28 ylabel('Amplitdue——>')
29 title('Amplitude Scaled Sequence')
30 //Time Scaling-Compression
31 C = input('Enter Compression factor-Time Scaling
           factor')
32 n = lx/C:1/C:hx/C;
33 subplot(2,2,3)
34 a = gca();
35 a.x_location = "origin";
36 a.y_location = "origin";
37 a.data_bounds = [-10,0;10,10];
38 plot2d3('gnn',n,x)
39 xlabel('n====>')
40 ylabel('Amplitude——>')
41 title('Compressed Sequence')
42 //Time Scaling-Expansion
43 d = input('Enter Extension factor-Time Scaling
           factor')
44 n = lx*d:d:hx*d;

```

```

45 subplot(2,2,4)
46 a = gca();
47 a.x_location = "origin";
48 a.y_location = "origin";
49 a.data_bounds = [-10,0;10,10];
50 plot2d3('gnn',n,x)
51 xlabel('n====>')
52 ylabel('Amplitude——>')
53 title('Extended Sequence')
54 //Example
55 //Enter input Sequence:=[1,2,3,4,5]
56 //
57 //Enter starting point of original signal:= 2
58 //
59 //Amplitude Scaling Factor:= 2
60 //
61 //Enter Compression factor—Time Scaling factor 2
62 //
63 //Enter Extension factor—Time Scaling factor 2

```

---

#### Scilab code Solution 6.4 Exp6d

```

1 //Caption:Program to demonstrate the shifting of the
  discrete time signal
2 clc;
3 clear;
4 close;
5 x = input('Enter the input sequence:=')
6 m = length(x);
7 lx = input('Enter the starting point of original
  signal:=')
8 hx = lx+m-1;
9 n = lx:1:hx;

```

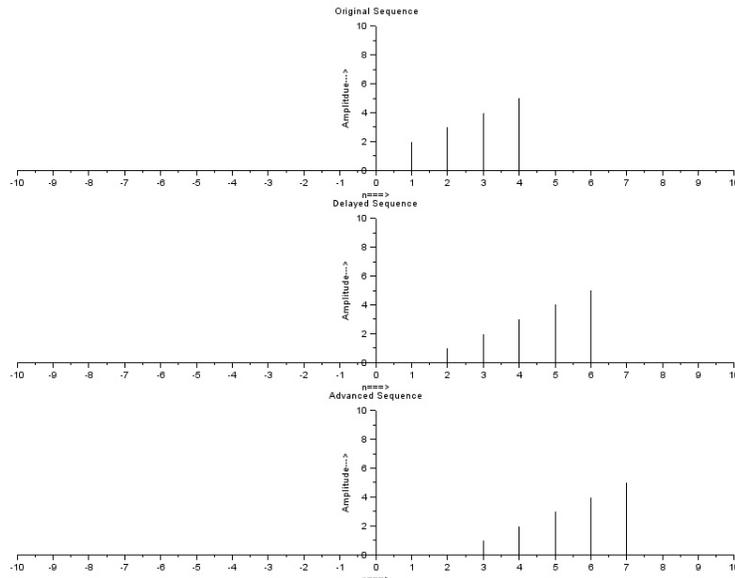


Figure 6.5: Exp6d

```

10 subplot(3,1,1)
11 a = gca();
12 a.x_location = "origin";
13 a.y_location = "origin";
14 a.data_bounds = [-10,0;10,10];
15 plot2d3('gnn',n,x);
16 xlabel('n====>')
17 ylabel('Amplitdue——>')
18 title('Original Sequence')
19 //
20 d = input('Enter the delay:=')
21 n = lx+d:1:hx+d;
22 subplot(3,1,2)
23 a = gca();
24 a.x_location = "origin";
25 a.y_location = "origin";
26 a.data_bounds = [-10,0;10,10];
27 plot2d3('gnn',n,x)
28 xlabel('n====>')

```

```

29 ylabel('Amplitude—>')
30 title('Delayed Sequence')
31 //
32 a = input('Enter the advance:=')
33 n = lx-a:1:hx-a;
34 subplot(3,1,3)
35 a = gca();
36 a.x_location = "origin";
37 a.y_location = "origin";
38 a.data_bounds = [-10,0;10,10];
39 plot2d3('gmn',n,x)
40 xlabel('n====>')
41 ylabel('Amplitude—>')
42 title('Advanced Sequence')
43 //Example
44 //Enter the input sequence:=[1,2,3,4,5]
45 //
46 //Enter the starting point of original signal:=0
47 //
48 //Enter the delay:=2
49 //
50 //Enter the advance:=3

```

---

# Experiment: 7

## Program for Discrete Fourier Transform

Scilab code Solution 7.1 Exp7

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

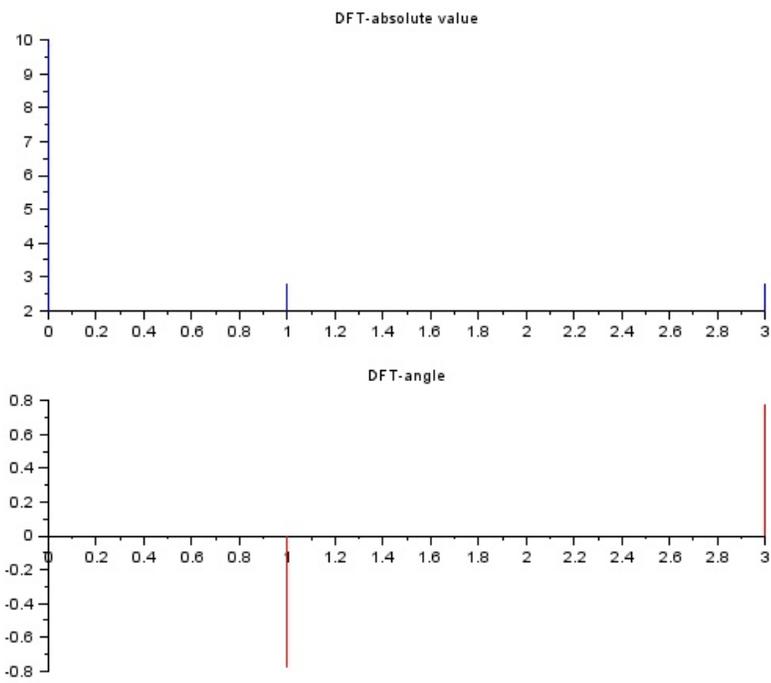


Figure 7.1: Exp7

```

17     for n=1:N
18         w=exp((-2*pi*i*(k-1)*(n-1))/N);
19         x(n)=w;
20         c(k,n)=x(n);
21     end
22
23 end
24 r=x2*c;
25 //plotting magnitude and angle
26 subplot(2,1,1)
27 plot2d3('gnn',0:N-1,abs(r),2);
28 title('DFT-absolute value');
29 subplot(2,1,2)
30 a = gca()
31 plot2d3('gnn',0:N-1,atan(imag(r)./(real(r)+0.0001))
    ,5);
32 a.x_location="origin";
33 title('DFT-angle');
34 disp(r,'Discrete Fourier Transform Result')
35 //RESULT
36 //Example 1
37 //Howmany point DFT do you want? 4
38 //Enter the sequence=[1,2,3,4]
39 //Discrete Fourier Transform Result
40 //      10. - 2. + 2.i - 2. - 9.797D-16i - 2. - 2.i
41 //
42 //Example 2
43 //Howmany point DFT do you want?8
44 //Enter the sequence=[1,1,1,1,1,1,1,1]
45 //Discrete Fourier Transform Result
46 //      column 1 to 5
47 //
48 //      8. - 5.551D-16 + 2.220D-16i - 4.286D-16 -
49 //      4.441D-16i - 2.220D-16 + 8.882D-16i - 4.899D-16
50 //      i
51 //      column 6 to 8

```

```

52 // - 2.109D-15 - 1.221D-15i - 2.933D-15 - 6.661D
    -16i      3.553D-15 + 1.110D-15i
53 //
54 //Example 3
55 //Howmany point DFT do you want? 8
56 //Enter the sequence= [0,1,2,3,4,5,6,7]
57 //Discrete Fourier Transform Result
58 //
59 //
60 //      column 1 to 7
61 //
62 //      28. - 4. + 9.6568542i - 4. + 4.i - 4. +
    1.6568542i - 4. - 3.429D-15i - 4. - 1.6568542i
    - 4. - 4.i
63 //
64 //      column 8
65 //
66 // - 4. - 9.6568542i
67 //

```

---

# Experiment: 8

## Simulation of FIR Filters

Scilab code Solution 8.1 Exp8

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

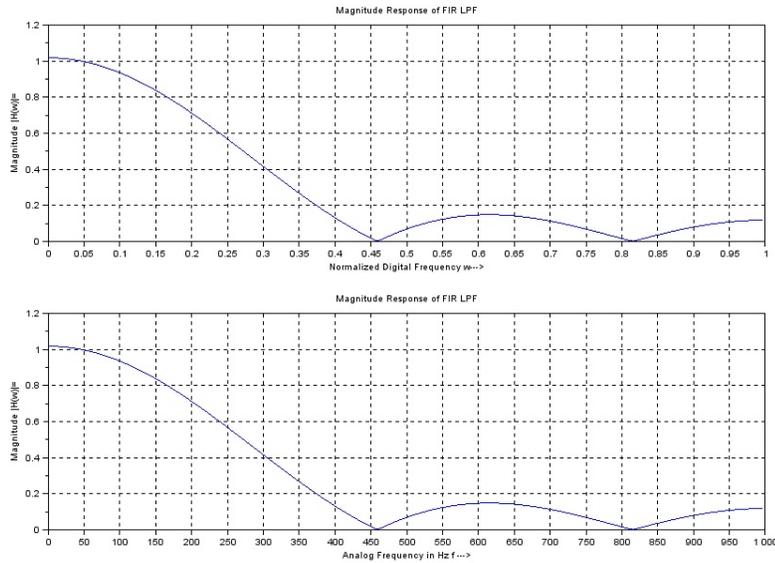


Figure 8.1: Exp8

```

17 subplot(2,1,1)
18 plot(2*fr,wfm)
19 xlabel('Normalized Digital Frequency w-->')
20 ylabel('Magnitude |H(w)|=')
21 title('Magnitude Response of FIR LPF')
22 xgrid(1)
23 subplot(2,1,2)
24 plot(fr*fs,wfm)
25 xlabel('Analog Frequency in Hz f -->')
26 ylabel('Magnitude |H(w)|=')
27 title('Magnitude Response of FIR LPF')
28 xgrid(1)
29 //Example
30 //Enter Analog cutoff freq. in Hz= 250
31 //
32 //Enter Analog sampling freq. in Hz= 2000
33 //
34 //Enter order of filter = 4
35 //

```

```
36 // Digital cutoff frequency in radians.cycles/  
    samples  
37 //  
38 //     0.7853982  
39 //  
40 // Normalized digital cutoff frequency in cycles/  
    samples  
41 //  
42 //     0.25  
43 //  
44 // Impulse Response of LPF FIR Filter:h[n]=  
45 //  
46 //     0.1591549     0.2250791     0.25     0.2250791  
    0.1591549
```

---

# Experiment: 9

## Generation and Quantization of Binary Numbers

### Scilab code Solution 9.1 Exp9

```
1 //Note: Details of scilab software version and OS
   version used:
2 //OS: Windows 7
3 //Scilab version: 5.4.1
4 //IPD Atom version:8.3.1-2
5 //SIVP Atom version:0.5.3.1-2
6 //1.Quantization and sampling
7
8 //Quantize a signal to n bits. This code assumes
   the signal is between -1
9 //and +1.
10 clc;
11 clear all;
12 close;
13 n=8; //Number of bits;
14 m= 120; //Number of samples;
15 t = 2*%pi*[0:(m-1)]/m;
```

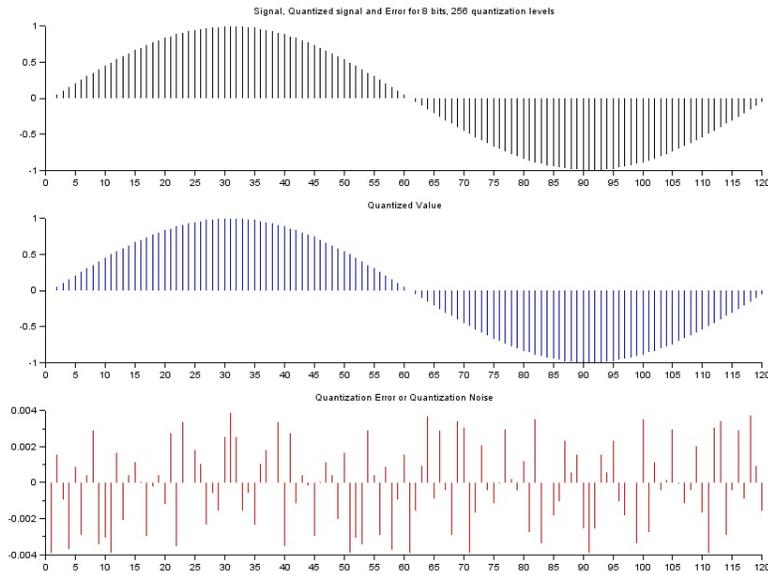


Figure 9.1: Exp9

```

16 x=sin(t); //signal between -1 and 1.
17 //Trying "sin()"
// instead of "
// sawtooth"
18 //results in more
// interesting error
// (to the
19 //extent that error
// is interesting).
20 x(find(x>=1))=(1-%eps); //Make signal from -1
// to just less than 1.
21 xq=floor((x+1)*2^(n-1)); //Signal is one of 2^n
// int values (0 to 2^n-1)
22 xq=xq/(2^(n-1)); //Signal is from 0 to 2 (
// quantized)
23 xq=xq-(2^(n)-1)/2^(n); //Shift signal down (
// rounding)
24
25 xe=x-xq; //Error

```

```
26 subplot(3,1,1)
27 plot2d3('gnn',1:length(x),x);
28 title(sprintf('Signal, Quantized signal and Error
    for %g bits, %g quantization levels',n,2^n));
29 disp(x,'exact value')
30 subplot(3,1,2)
31 plot2d3('gnn',1:length(xq),xq,2);
32 title('Quantized Value')
33 disp(xq,'Quantized value')
34 subplot(3,1,3)
35 plot2d3('gnn',1:length(xe),xe,5);
36 title('Quantization Error or Quantization Noise')
37 disp(xe,'Quantization error or noise')
```

---

# Experiment: 10

## Introduction to Simulink Signal Analysis

Scilab code Solution 10.1 Exp10a

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

---

This code can be downloaded from the website [www.scilab.in](http://www.scilab.in)

Scilab code Solution 10.2 Exp10b

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

---

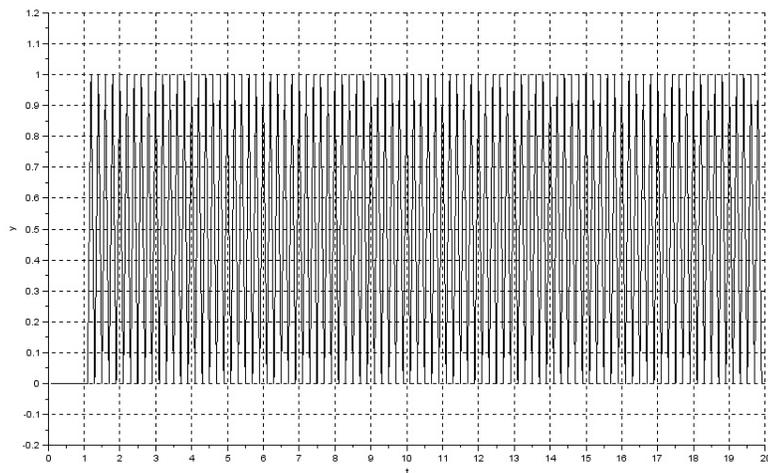


Figure 10.1: Exp10a

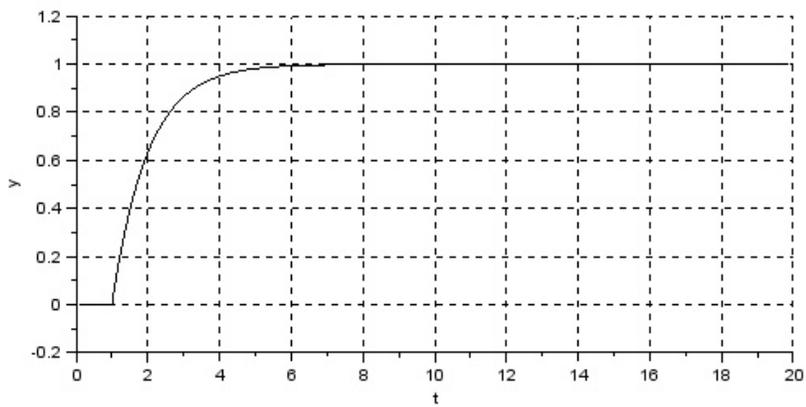


Figure 10.2: Exp10b

This code can be downloaded from the website [www.scilab.in](http://www.scilab.in)

# Experiment: 11

## Design and analysis of Butterworth Filter

Scilab code Solution 11.1 Exp11

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

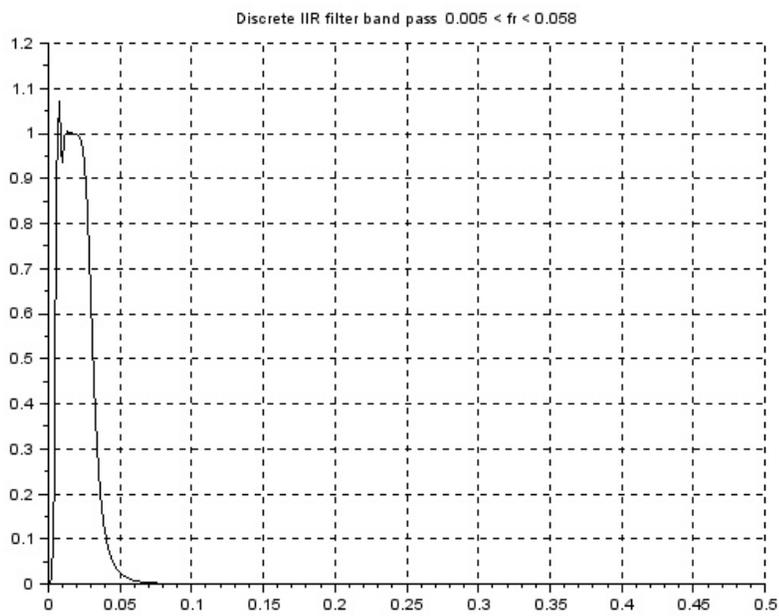


Figure 11.1: Exp11

```

16 [hzm,fr]=frmag(hz,256);
17 plot2d(fr',hzm');
18 xtitle('Discrete IIR filter band pass 0.005 < fr <
        0.058 ',' ',' ');
19 xgrid(1)
20 //Result
21
22 //-->hz(2)
23 // ans =
24 //
25 //
26 //      2      4
27 //      0.0000002 - 0.0000015z + 0.0000037z
28 //
29 //      6      8
30 //      - 0.0000049z + 0.0000037z
31 //
32 //      10     12
33 //      - 0.0000015z + 0.0000002z
34 //
35 //-->hz(3)
36 // ans =
37 //
38 //
39 //      2
40 //      0.5250468 - 6.6287407z + 38.377802z
41 //
42 //      3      4
43 //      - 134.73451z + 319.45814z
44 //
45 //      5      6
46 //      - 538.91189z + 663.25134z
47 //
48 //      7      8
49 //      - 600.03003z + 396.0233z
50 //
51 //      9      10
52 //      - 185.96443z + 58.974503z
53 //
54 //      11  12
55 //      - 11.340535z + z
56 //
57 //-->hz(1)
58 // ans =
59 //
60 //
61 //!r num den dt !
62 //

```

```

53 //→p
54 // p =
55 //
56 // 0.9964120 - 0.0154005 i
57 // 0.9892502 - 0.0129619 i
58 // 0.9822499 - 0.0060512 i
59 // 0.9822499 + 0.0060512 i
60 // 0.9892502 + 0.0129619 i
61 // 0.9964120 + 0.0154005 i
62 // 0.9464056 + 0.1686845 i
63 // 0.8907712 + 0.1177110 i
64 // 0.8651786 + 0.0429743 i
65 // 0.8651786 - 0.0429743 i
66 // 0.8907712 - 0.1177110 i
67 // 0.9464056 - 0.1686845 i
68 //
69 //→z
70 // z =
71 //
72 // 1.
73 // 1.
74 // 1.
75 // 1.
76 // 1.
77 // 1.
78 // - 1.
79 // - 1.
80 // - 1.
81 // - 1.
82 // - 1.
83 // - 1.
84 //
85 //→g
86 // g =
87 //
88 // 0.0000002
89 //

```

---

# Experiment: 12

## Impulse response of first order and second order system

Scilab code Solution 12.1 Exp12

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

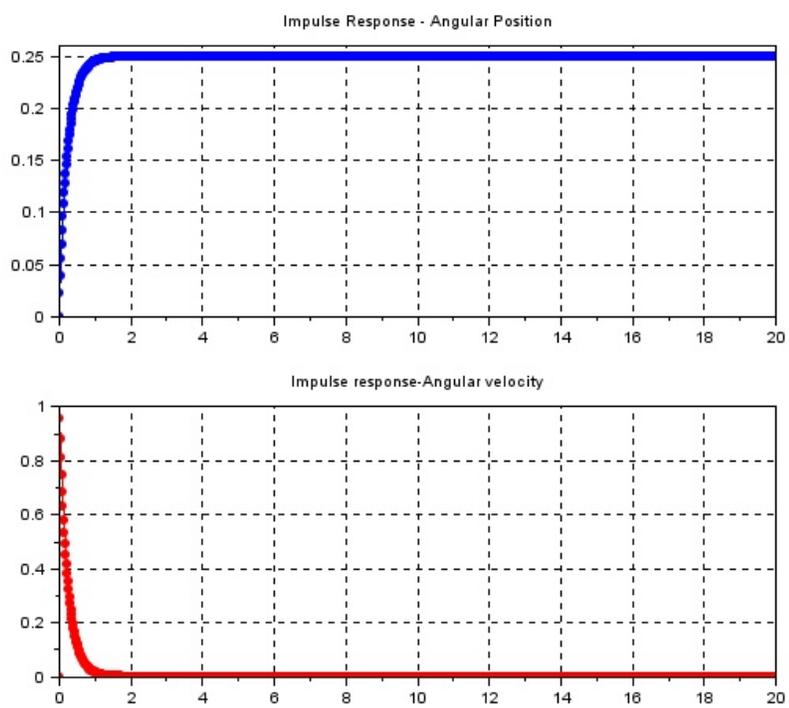


Figure 12.1: Exp12

```

14 model = [Angular_Position,Angular_velocity];
15 H1 = model(:,1); //Angular Position
16 H2 = model(:,2); //Angular velocity
17 np=20; //number of points
18 t = 0:0.02:20;
19 ysd1 = csim('impulse',t,model(:,1));
20 ysd2 = csim('impulse',t,model(:,2));
21 subplot(2,1,1)
22 plot(t,ysd1,'.-b')
23 title('Impulse Response – Angular Position')
24 xgrid(1)
25 subplot(2,1,2)
26 plot(t,ysd2,'.-r')
27 title('Impulse response–Angular velocity')
28 xgrid(1)
29 disp(model,'Model System Equations =')
30 disp(ysd1,'Impulse Resposne of Angular Position =')
31 disp(ysd2,'Impulse Response of Angular velocity=')
32
33 //RESULT
34 //Model System Equations =
35 //
36 //      0.9815 + 0.003127s          0.2756 + 1.04s
37 //      _____                _____
38 //                                 2                2
39 //      0.0000634 + 3.929s + s      1.096 + 4.461s + s

```

---

# Experiment: 13

## Circular convolution of two given sequences.

Scilab code Solution 13.1 Exp13

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

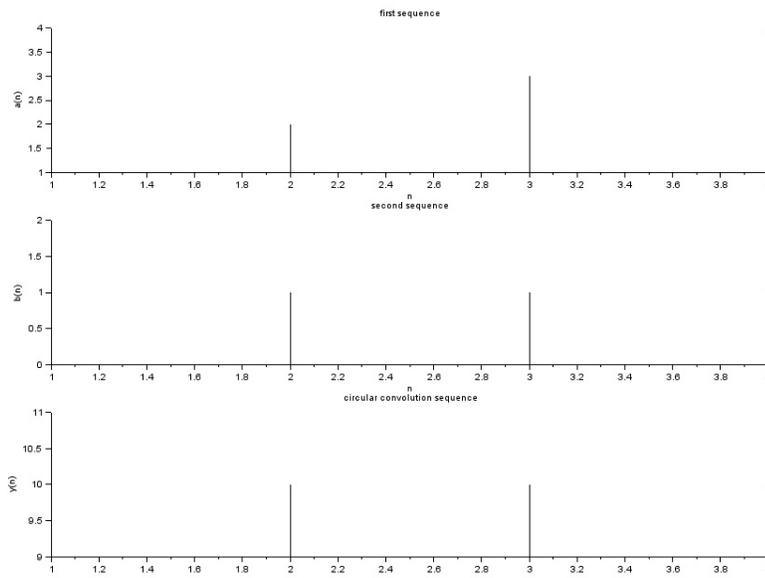


Figure 13.1: Exp13

```

17  a1=[a1,zeros(1,-n3)];
18  else
19  b1=[b1,zeros(1,n3)];
20  end
21  for r = 1:n
22      y(r)=0;
23      for i=1:n
24          j=r-i+1;
25          if (j<=0)
26              j=j+n;
27          end
28          y(r)=y(r)+b1(j)*a1(j);
29      end
30  end
31  disp(y,'circular convloution result')
32
33  subplot(3,1,1);
34  plot2d3('gnn',a1);
35  xlabel('n');

```

```

36 ylabel('a(n)');
37 title('first sequence');
38 subplot(3,1,2);
39 plot2d3('gnn',b1);
40 xlabel('n');
41 ylabel('b(n)');
42 title('second sequence');
43 subplot(3,1,3);
44 plot2d3('gnn',y);
45 xlabel('n');
46 ylabel('y(n)');
47 title('circular convolution sequence');
48
49 //RESULT
50 //Example 1
51 //1st Sequence x:[1,2,3,4]
52 //2nd Sequence h:[1,1,1,1]
53 //
54 // circular convloution result
55 //
56 //      10.      10.      10.      10.
57 //
58 //Example 2
59 //1st Sequence x:[1,2,3,4]
60 //2nd Sequence h:[1,1,1]
61 //
62 // circular convloution result
63 //
64 //      6.      6.      6.      6.

```

---

# Experiment: 14

## Linear convolution of two given sequences.

Scilab code Solution 14.1 Exp14

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

```

19         if q>m
20             q=q-1;
21         elseif j>n
22             k=k;
23         else k=x(q)*y(j)+k;
24             q=q-1;
25         end
26     end
27     z(i)=k;
28 end
29 disp(z,'convolution of two sequence is:')
30
31 //RESULT
32 //enter the first sequence [1,2,3]
33 //enter the second sequence [1,1,1,1]
34 //
35 // convolution of two sequence is:
36 //
37 //     1.
38 //     3.
39 //     6.
40 //     6.
41 //     5.
42 //     3.
43 //

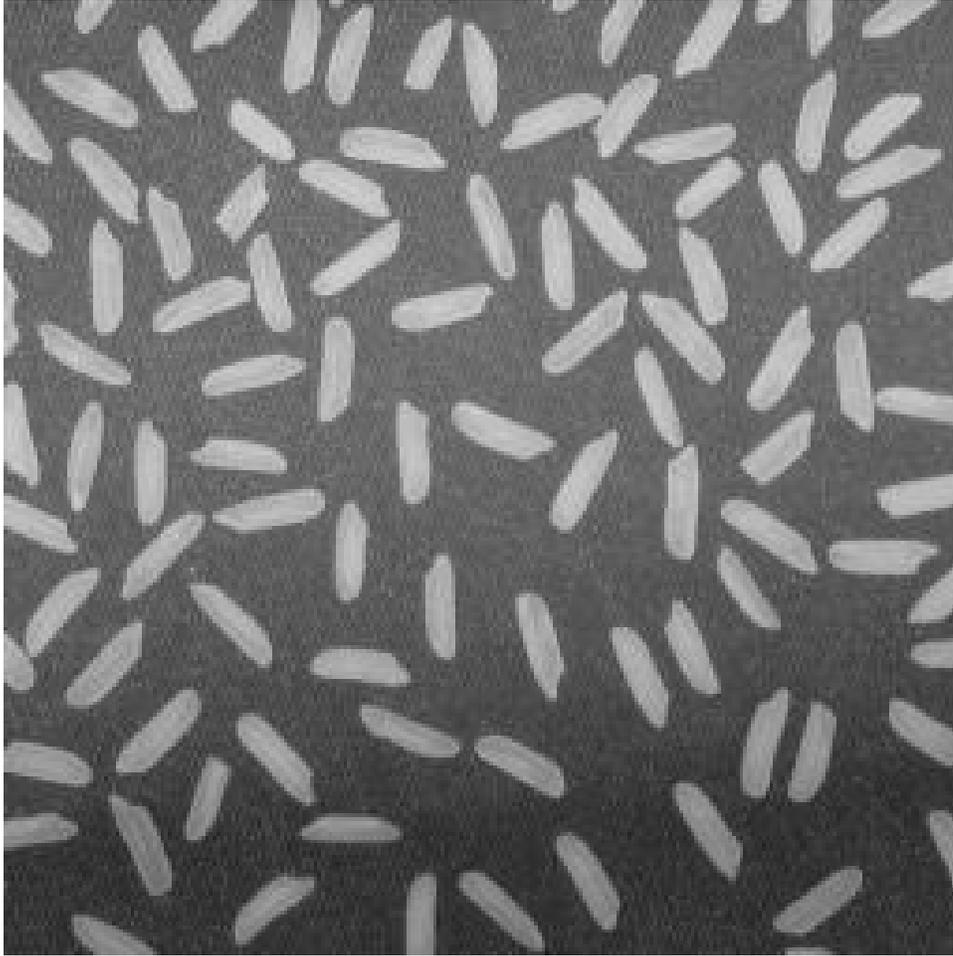
```

---

# Appendix

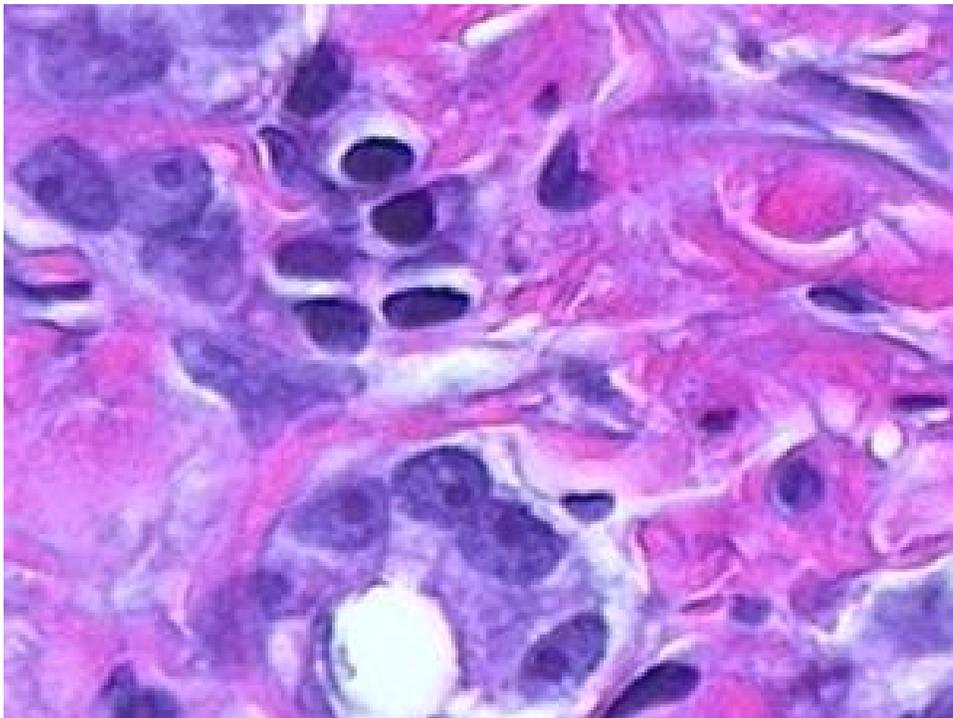


Cameraman Image file



Rice

Image File



tian Colour Image File

Hes-



Lenna Image File