Scilab Manual for
Discrete Time Signal Processing
by Prof Mr. Rajiv Suhas Tawde
Others
Mumbai University/padmabhushan
Vasantdada Patil Pratishthan'S College Of
Engineering (Pvppcoe)[1]

Solutions provided by
Prof Mr. Rajiv Suhas Tawde
Others
Mumbai University/padmabhushan Vasantdada Patil Pratishthan'S College Of F

April 22, 2026

Contents

3

List of Experiments

Experiment: 1

Linear Convolution Without Using Inbuilt Scilab Convolution Function

Scilab code Solution 1.0 Experiment Number 1

```
1 //AIM: Linear convolution without using inbuilt
     Scilab convolution function.
2 // Software version Scilab 5.5.2
3 //OS windows 10
4 clc ;
5 clear ;
6 // Given that x(n)={1,2,3,4,5,6,7,8} and h(n)={1,2}
7 // Soln :
8 x0 =1;
9 x1 =2;
10 x2 =3;
11 x3 =4;
12 x4 =5;
13 x5 =6;
14 x6 =7;
15 x7 =8;
16 h0 =1;
```

```
17  h1 =2;
18  xhn0_a = x0 * h0;
19  xhn1_a = x1 * h0;
20  xhn2_a = x2 * h0;
21  xhn3_a = x3 * h0;
22  xhn4_a = x4 * h0;
23  xhn5_a = x5 * h0;
24  xhn6_a = x6 * h0;
25  xhn7_a = x7 * h0;
26
27  xhn0_b = x0 * h1;
28  xhn1_b = x1 * h1;
29  xhn2_b = x2 * h1;
30  xhn3_b = x3 * h1;
31  xhn4_b = x4 * h1;
32  xhn5_b = x5 * h1;
33  xhn6_b = x6 * h1;
34  xhn7_b = x7 * h1;
35
36  y0 = xhn0_a;
37  y1 = xhn0_b + xhn1_a;
38  y2 = xhn1_b + xhn2_a;
39  y3 = xhn2_b + xhn3_a;
40  y4 = xhn3_b + xhn4_a;
41  y5 = xhn4_b + xhn5_a;
42  y6 = xhn5_b + xhn6_a;
43  y7 = xhn6_b + xhn7_a;
44  y8 = xhn7_b;
45  disp({,y0,y1,y2,y3,y4,y5,y6,y7,y8,},'Output of
        linear  convolution  i.e  y(n)=')
```

Experiment: 2

Derive The Four Point Twiddle Factor Matrix

Scilab code Solution 2.0 Experiment Number 2

```
1  //AIM: Derive the 4 point twiddle factor matrix.
2  //Software version Scilab 5.5.2
3  //OS windows 10
4  clc;
5  clear;
6  W40=cos(0)-(sqrt(-1)*sin(0));
7  W41=cos(%pi/2)-(sqrt(-1))*(sin(%pi/2));
8  W42=cos(%pi)-(sqrt(-1))*(sin(%pi));
9  W43=cos((3*%pi)/2)-(sqrt(-1))*sin((3*%pi)/2);
10
11 disp(W40,'W40=')
12 disp(W41,'W41=')
13 disp(W42,'W42=')
14 disp(W43,'W43=')
15
16 W44=W40;
17 W49=W41;
18 W46=W42;
19
```

```scilab
20  W4_raw_matrix= [W40 W40 W40 W40;
21             W40 W41 W42 W43;
22             W40 W42 W44 W46;
23             W40 W43 W46 W49]
24  disp(W4_raw_matrix,'W4_raw_matrix=')
25  disp('Type resume in console and press enter')
26  pause
27  a=ceil(- 1.225D-16);
28  disp(a,'Imaginary part of W42 & Imaginary part of
       W46 i.e. ceil(- 1.225D-16)=')
29  b=ceil(- 1.837D-16);
30  disp(b,'Real part of W43 i.e. ceil(- 1.837D-16)=')
31  c=int(6.123D-17);
32  disp(c,'Real part of W41 & Real part of W49 i.e. int
       (6.123D-17)=')
33
34  //So W4 can be modified as follows :
35  d=real(W42);//Imaginary part is ignored since it is
       zero
36  e=imag(W43)*sqrt(-1);//Real part is ignored since it
       is zero
37  f=imag(W41)*(sqrt(-1));//Real part is ignored since
       it is zero
38
39  W4=[W40 W40 W40 W40;
40             W40 f d e;
41             W40 d W44 d;
42             W40 e d f]
43  disp(W4,'Final W4 matrix =')
```

Experiment: 3

Four Point Dit-Fft Without Using Inbuilt Scilab Fft Function

Scilab code Solution 3.0 Experiment Number 3

```
1  //AIM: Four  point  DIT-FFT  without  using  inbuilt
       Scilab  FFT  function
2
3  //Computing  four  point  DFT  for  x(n)={1,2,3,4}  using
       Decimation  in  Time-Fast
4  //Fourier  transform  (i.e.  DIT-FFT )
5  //without  using  readymade  inbuilt  Scilab  functions
       for  DFT/FFT.
6
7  //Software  version  Scilab  5.5.2
8  //OS  windows  10
9  clc;
10  clear;
11  //Let  us  begin  with  the  programming.For
       understanding ,let  us  write  the  given
12  //data  as
13  //x(0)=1;x(1)=2,x(2)=3,x(3)=4
```

```
14  x0 =1;
15  x2 =3;
16  x1 =2;
17  x3 =4;
18  // Stage I computation
19  x0a = x2 + x0 ; // Computing Stage−I output at position 1
20  disp ( x0a , 'Stage−I output at position 1 ')
21  x2b =( x2 - x0 ) *( -1) ; // Computing Stage−I output at
        position 2
22  disp ( x2b , 'Stage−I output at position 2 ')
23  x1c = x3 + x1 ; // Computing Stage−I output at position 3
24  disp ( x1c , 'Stage−I output at position 3 ')
25  x3d =( x3 - x1 ) *( -1) ; // Computing Stage−I output at
        position 4
26  disp ( x3d , 'Stage−I output at position 4 ')
27  // Stage−II computation
28  x3d1 = x3d *( - sqrt ( -1) ) ; // Multiply by (−j) in the last
          line
29  disp ( x3d1 , 'Stage−II input at the fourth line ')
30  X0 = x1c + x0a ; // Computing final stage output value X(0)
31  disp (X0 , 'The final stage output X(0)=')
32  X1 = x3d1 + x2b ; // Computing final stage output value X
        (1)
33  disp (X1 , 'The final stage output X(1)=')
34  X2 =( x1c - x0a ) *( -1) ; // Computing final stage output
        value X(2)
35  disp (X2 , 'The final stage output X(2)=')
36  X3 =( x3d1 - x2b ) *( -1) ; // Computing final stage output
        value X(3)
37  disp (X3 , 'The final stage output X(3)=')
38  disp ({ , X0 , X1 , X2 , X3 ,} , 'So , the DFT of x(n) using
        Decimation−in−Time Fast Fourier Transform ( DIT−FFT
        ) is X(k)=')
```

Experiment: 4

Four Point Idit-Fft Without Using Inbuilt Scilab Fft Function

Scilab code Solution 4.0 Experiment Number 4

```
1  //AIM: Four point IDIT-FFT without using inbuilt
       Scilab FFT function
2
3  //Computing four point IDFT for X(k)={10,-2+2j
       ,-2,-2-2j} using
4  //Inverse Decimation in Time-Fast Fourier transform
       (i.e. IDIT-FFT )
5  //without using readymade inbuilt Scilab functions
       for IDFT/IFFT.
6
7  //Software version Scilab 5.5.2
8  //OS windows 10
9  clc;
10 clear;
11 //Let us begin with the programming.For
       understanding ,let us write the given
12 //data as
```

```
13  //X(0)=10;X(1)=−2+2j ,X(2)=−2,X(3)=−2−2j
14  X0c=10;//X0c means complex conjugate of XO
15  X2c=-2;//X2c means complex conjugate of X2
16  X1c=(-2)+((-1)*(2)*(sqrt(-1)));//X1c means complex
       conjugate of X1
17  X3c=(-2)-((-1)*(2)*(sqrt(-1)));//X3c means complex
       conjugate of X3
18  disp(X0c,'X*(0)=')
19  disp(X2c,'X*(2)=')
20  disp(X1c,'X*(1)=')
21  disp(X3c,'X*(3)=')
22  x0_star=(((X3c+X1c)*(1))+((X2c+X0c)*(1)))*(1/4)
23  disp(x0_star,'x*(0)=')
24  x1_star=(((X3c-X1c)*(-1)*(-sqrt(-1)))+(X2c-X0c)*(-1)
       )*(1/4);
25  disp(x1_star,'x*(1)=')
26  x2_star=(((X3c+X1c)*(1)-(X2c+X0c))*(-1))*(1/4);
27  disp(x2_star,'x*(2)=')
28  x3_star=((((X3c-X1c)*(-1)*(-sqrt(-1)))-((X2c-X0c)
       *(-1)))*(-1))*(1/4);
29  disp(x3_star,'x*(3)=')
30  disp({,x0_star,x1_star,x2_star,x3_star,},'x*(n)=')
31  //The computed value is x*(n). But we need x(n) as
       final output.
32  //We will separate real part of x*(n)
33  //We will separate imaginary part of x*(n) and take
       its complex conjugate by
34  //multiplying by a factor of (−1)
35  x0_star_real=real(x0_star);
36  x0_star_conj=(-1)*(imag(x0_star));
37  x1_star_real=real(x1_star);
38  x1_star_conj=(-1)*(imag(x1_star));
39  x2_star_real=real(x2_star);
40  x2_star_conj=(-1)*(imag(x2_star));
41  x3_star_real=real(x3_star);
42  x3_star_conj=(-1)*(imag(x3_star));
43  //Finally, we will add the real part and the
       imaginary part whose complex
```

```
44 //conjugate is taken to get x(0),(1),x(2) and x(3)
45 x0=x0_star_real+x0_star_conj
46 x1=x1_star_real+x1_star_conj
47 x2=x2_star_real+x2_star_conj
48 x3=x3_star_real+x3_star_conj
49 disp({,x0,x1,x2,x3,},'So,the IDFT of X(k) using
      Inverse Decimation-in-Time Fast Fourier Transform
      (IDIT-FFT) is x(n)=')
```

Scilab code Solution 4.1 Experiment Number 4 extra solution

```
1 //AIM:Four point IDIT-FFT without using inbuilt
      Scilab FFT function
2 // (This is an extra solution)
3
4 //Computing four point IDFT for X(k)={10,-2+2j
      ,-2,-2-2j} using
5 //Inverse Decimation in Time-Fast Fourier transform
      (i.e. IDIT-FFT )
6 //without using readymade inbuilt Scilab functions
      for IDFT/IFFT.
7
8 //Software version Scilab 5.5.2
9 //OS windows 10
10 clc;
11 clear;
12 //Let us begin with the programming.For
      understanding ,let us write the given
13 //data as
14 //X(0)=10;X(1)=-2+2j ,X(2)=-2,X(3)=-2-2j
15 X0_conj=10;//X0_conj means complex conjugate of XO
16 X2_conj=-2;//X2_conj means complex conjugate of X2
17 X1_conj=(-2)+((-1)*(2)*(sqrt(-1)));//X1_conj means
      complex conjugate of X1
18 X3_conj=(-2)-((-1)*(2)*(sqrt(-1)));//X3_conj means
```

```
              complex  conjugate  of  X3
19  disp(X0_conj ,'X*(0)=')
20  disp(X2_conj ,'X*(2)=')
21  disp(X1_conj ,'X*(1)=')
22  disp(X3_conj ,'X*(3)=')
23  //Stage  I  computation
24  X0a=X2_conj+X0_conj;//Computing  Stage−I  output  at
        line  1
25  disp(X0a ,'Stage−I  output  at  line  1')
26  X2b=(X2_conj-X0_conj)*(-1);//Computing  Stage−I
        output  at  line  2
27  disp(X2b ,'Stage−I  output  at  line  2')
28  X1c=X3_conj+X1_conj;//Computing  Stage−I  output  at
        line  3
29  disp(X1c ,'Stage−I  output  at  line  3')
30  X3d=(X3_conj-X1_conj)*(-1);//Computing  Stage−I
        output  at  line  4
31  disp(X3d ,'Stage−I  output  at  line  4')
32  //Stage  II  computation
33  X3d1=X3d*(-sqrt(-1));//  Multiply  by  (−j)  in  the  last
         line
34  disp(X3d1 ,'Stage−II  input  at  the  fourth  line  ')
35  x0_star=X1c+X0a;//Computing  stage−II  output  value  at
        line  1
36  disp(x0_star ,'The  stage−II  output  value  at  line  1=')
37  x1_star=X3d1+X2b;//Computing  stage−II  output  value
        at  line  2
38  disp(x1_star ,'The  stage−II  output  value  at  line  2 ='
        )
39  x2_star=(X1c-X0a)*(-1);//Computing  stage−II  output
        value  at  line  3
40  disp(x2_star ,'The  stage−II  output  valueat  line  3=')
41  x3_star=(X3d1-X2b)*(-1);//Computing  stage−II  output
        value  at  line  4
42  disp(x3_star ,'The  stage−II  output  value  at  line  4=')
43  //Now  we  will  multiply  stage−II  output  values
        individually  with  a  factor  of
44  //(1/N).  Here  N=4,So  we  will  multiply  by  (1/4)
```

```scilab
45  x0_star_on_multiplication=(1/4)*(x0_star); //Multiply
        by(1/4)
46  disp(x0_star_on_multiplication,'x*(0)=')
47  x1_star_on_multiplication=(1/4)*(x1_star); //Multiply
        by(1/4)
48  disp(x1_star_on_multiplication,'x*(1)=')
49  x2_star_on_multiplication=(1/4)*(x2_star); //Multiply
        by(1/4)
50  disp(x2_star_on_multiplication,'x*(2)=')
51  x3_star_on_multiplication=(1/4)*(x3_star); //Multiply
        by(1/4)
52  disp(x3_star_on_multiplication,'x*(3)=')
53  disp({,x0_star_on_multiplication,
        x1_star_on_multiplication,
        x2_star_on_multiplication,
        x3_star_on_multiplication,},'x*(n)=')
54  x0_star_real=real(x0_star_on_multiplication);
55  x0_star_conj=(-1)*(imag(x0_star_on_multiplication));
56  x1_star_real=real(x1_star_on_multiplication);
57  x1_star_conj=(-1)*(imag(x1_star_on_multiplication));
58  x2_star_real=real(x2_star_on_multiplication);
59  x2_star_conj=(-1)*(imag(x2_star_on_multiplication));
60  x3_star_real=real(x3_star_on_multiplication);
61  x3_star_conj=(-1)*(imag(x3_star_on_multiplication));
62  x0=x0_star_real+x0_star_conj
63  x1=x1_star_real+x1_star_conj
64  x2=x2_star_real+x2_star_conj
65  x3=x3_star_real+x3_star_conj
66  disp({,x0,x1,x2,x3,},'So,the IDFT of X(k) using
        Inverse Decimation-in-Time Fast Fourier Transform
        (IDIT-FFT) is x(n)=')
```

Experiment: 5

Four Point Dif-Fft Without Using Inbuilt Scilab Fft Function

Scilab code Solution 5.0 Experiment Number 5

```
1  //AIM: Four point DIF-FFT without using inbuilt
       Scilab FFT function
2
3  //Computing four point DFT for x(n)={1,2,3,4} using
4  //Decimation in Frequency-Fast Fourier transform (i.
       e. DIF-FFT )without using
5  //readymade inbuilt Scilab functions for DFT/FFT.
6
7  //Software version Scilab 5.5.2
8  //OS windows 10
9  clc;
10 clear;
11 x0=1;
12 x1=2;
13 x2=3;
14 x3=4;
15 //Stage I computation
```

```
16  x0a=x2+x0;//Computing Stage-I output at position 1
17  disp(x0a,'Stage-I output at position 1')
18  x1b=(x3+x1);//Computing Stage-I output at position 2
19  disp(x1b,'Stage-I output at position 2')
20  x2c=(x2-x0)*(-1);//Computing Stage-I output at
        position 3
21  disp(x2c,'Stage-I output at position 3')
22  x3d=(x3-x1)*(-1);//Computing Stage-I output at
        position 4
23  disp(x3d,'Stage-I output at position 4')
24  //Stage-II computation
25  x3d1=x3d*(-sqrt(-1));// Multiply by (-j) in the last
         line
26  disp(x3d1,'Stage-II input at the fourth line ')
27  X0=x1b+x0a;//Computing final stage output value X(0)
28  disp(X0,'The final stage output X(0)=')
29  X2=(x1b-x0a)*(-1);//Computing final stage output
        value X(1)
30  disp(X2,'The final stage output X(2)=')
31  X1=(x3d1+x2c);//Computing final stage output value X
        (2)
32  disp(X1,'The final stage output X(1)=')
33  X3=(x3d1-x2c)*(-1);//Computing final stage output
        value X(3)
34  disp(X3,'The final stage output X(3)=')
35  disp({,X0,X1,X2,X3,},'So,the DFT of x(n) using
        Decimation-in-Frequency Fast Fourier Transform (
        DIF-FFT) is X(k)=')
```

Experiment: 6

Four Point Idif-Fft Without Using Inbuilt Scilab Fft Function

Scilab code Solution 6.0 Experiment Number 6

```
1 //AIM: Four point IDIF-FFT without using inbuilt
     Scilab FFT function
2
3 //Computing four point IDFT for X(k)={10,-2+2j
     ,-2,-2-2j} using
4 //Inverse Decimation in Frequency-Fast Fourier
     transform (i.e. IDIF-FFT )
5 //without using readymade inbuilt Scilab functions
     for IDFT/IFFT.
6
7 //Software version Scilab 5.5.2
8 //OS windows 10
9 clc;
10 clear;
11 //Let us begin with the programming.For
     understanding, let us write the given
12 //data as
```

```scilab
13  //X(0)=10;X(1)=-2+2j ,X(2)=-2,X(3)=-2-2j
14  X0c=10;//X0c means complex conjugate of XO
15  X1c=(-2)+((-1)*(2)*(sqrt(-1)));//X1c means complex
        conjugate of X1
16  X2c=-2;//X2c means complex conjugate of X2
17  X3c=(-2)-((-1)*(2)*(sqrt(-1)));//X3c means complex
        conjugate of X3
18  disp(X0c,'X*(0)=')
19  disp(X1c,'X*(1)=')
20  disp(X2c,'X*(2)=')
21  disp(X3c,'X*(3)=')
22  x0_star=((X3c+X1c)*(1)+(X2c+X0c)*(1))*(1/4);//
        Computing x*(0)
23  disp(x0_star,'x*(0)=')
24  x2_star=((((X3c+X1c)*(1))-((X2c+X0c)*(1)))*(-1))
        *(1/4);//Computing x*(2)
25  disp(x2_star,'x*(2)=')
26  x1_star=((X3c-X1c)*(-1)*(-sqrt(-1))+(X2c-X0c)*(-1))
        *(1/4);//Computing x*(1)
27  disp(x1_star,'x*(1)=')
28  //Computing x*(3)
29  x3_star=((((X3c-X1c)*(-1)*(-sqrt(-1))-(X2c-X0c)*(-1)
        ))*(-1))*(1/4);
30  disp(x3_star,'x*(3)=')
31  disp({,x0_star,x1_star,x2_star,x3_star,},'x*(n)=')
32  //The computed value is x*(n). But we need x(n) as
        final output.
33  //We will separate real part of x*(n)
34  //We will separate imaginary part of x*(n) and take
        its complex conjugate by
35  //multiplying by a factor of (-1)
36  x0_star_real=real(x0_star);
37  x0_star_conj=(-1)*(imag(x0_star));
38  x1_star_real=real(x1_star);
39  x1_star_conj=(-1)*(imag(x1_star));
40  x2_star_real=real(x2_star);
41  x2_star_conj=(-1)*(imag(x2_star));
42  x3_star_real=real(x3_star);
```

```
43  x3_star_conj=(-1)*(imag(x3_star));
44  //Finally, we will add the real part and the
        imaginary part whose complex
45  //conjugate is taken to get xx(0),(1),x(2) and x(3)
46  x0=x0_star_real+x0_star_conj;//Computing x(0)
47  x1=x1_star_real+x1_star_conj;//Computing x(1)
48  x2=x2_star_real+x2_star_conj;//Computing x(2)
49  x3=x3_star_real+x3_star_conj;//Computing x(3)
50  disp({,x0,x1,x2,x3,},'So,the IDFT of X(k) using
        Inverse Decimation−in−Frequency Fast Fourier
        Transform(IDIF−FFT) is x(n)=')
```

Scilab code Solution 6.1 Experiment Number 6 extra solution

```
1  //AIM:Four point IDIF−FFT without using inbuilt
        Scilab FFT function
2  // (This is an extra solution)
3
4  //Computing four point IDFT for X(k)={10,−2+2j
        ,−2,−2−2j} using Inverse
5  //Decimation in Frequency−Fast Fourier transform (i.
        e. IDIF−FFT )
6  //without using readymade inbuilt Scilab functions
        for IDFT/IFFT.
7
8  //Software version Scilab 5.5.2
9  //OS windows 10
10 clc;
11 clear;
12 //Let us begin with the programming.For
        understanding,let us write the given
13 //data as
14 //X(0)=10;X(1)=−2+2j ,X(2)=−2,X(3)=−2−2j
15 X0_conj=10;//X0_conj means complex conjugate of X(O)
16 X1_conj=(-2)+((-1)*(2)*(sqrt(-1)));//X1_conj means
```

```
      complex conjugate of X(1)
17  X2_conj=-2;//X2_conj means complex conjugate of X(2)
18  X3_conj=(-2)-((-1)*(2)*(sqrt(-1)));//X3_conj means
      complex conjugate of X(3)
19  disp(X0_conj,'X*(0)=')
20  disp(X1_conj,'X*(1)=')
21  disp(X2_conj,'X*(2)=')
22  disp(X3_conj,'X*(3)=')
23  //Stage I computation
24  X0a=X2_conj+X0_conj;//Computing Stage−I output at
      line 1
25  disp(X0a,'Stage−I output at line 1')
26  X1b=(X3_conj+X1_conj);//Computing Stage−I output at
      line 2
27  disp(X1b,'Stage−I output at line 2')
28  X2c=(X2_conj-X0_conj)*(-1);//Computing Stage−I
      output at line 3
29  disp(X2c,'Stage−I output at line 3')
30  X3d=(X3_conj-X1_conj)*(-1);//Computing Stage−I
      output at line 4
31  disp(X3d,'Stage−I output at line 4')
32  //Stage II computation
33  X3d1=X3d*(-sqrt(-1));// Multiply by (−j) in the last
      line
34  disp(X3d1,'Stage−II input at the fourth line ')
35  x0_conj=X1b+X0a;//Computing stage−II output value at
      line 1
36  disp(x0_conj,'The stage−II output value at line 1=')
37  x2_conj=(X1b-X0a)*(-1);//Computing stage−II output
      value at line 2
38  disp(x2_conj,'The stage−II output value at line 2=')
39  x1_conj=X3d1+X2c;//Computing stage−II output value
      at line 3
40  disp(x1_conj,'The stage−II output value at line 3=')
41  x3_conj=(X3d1-X2c)*(-1);//Computing stage−II output
      value at line 4
42  disp(x3_conj,'The stage−II output value at line 4=')
43  //Now we will multiply stage−II output values
```

```
          individually with a factor of
44  // (1/N). Here N=4,So we will multiply by (1/4)
45  x0_conj_final=(1/4)*(x0_conj)
46  disp(x0_conj_final,'x*(0)=')
47  x2_conj_final=(1/4)*(x2_conj)
48  disp(x2_conj_final,'x*(2)=')
49  x1_conj_final=(1/4)*(x1_conj)
50  disp(x1_conj_final,'x*(1)=')
51  x3_conj_final=(1/4)*(x3_conj)
52  disp(x3_conj_final,'x*(3)=')
53  disp({,x0_conj_final,x1_conj_final,x2_conj_final,
        x3_conj_final,},'So,the IDFT of X(k) using
        Inverse Decimation-in-Frequency Fast Fourier
        Transform(IDIF-FFT) is x(n)=')
```

Experiment: 7

Derive The Six Point Twiddle Factor Matrix [w6] Useful For Dft Computation

Scilab code Solution 7.0 Experiment Number 7

```
1 //AIM: Derive the six point twiddle factor matrix [W6
     ] useful for DFT computation
2 //Software version Scilab 5.5.2
3 //OS windows 10
4 clc;
5 clear;
6 //Computing the twiddle factor values for W60,W61,
     W62,W63,W64,W65 :
7 W60=int(cos(0))-(sqrt(-1)*int(sin(0)));
8 W61=cos((2*%pi*1)/6)-(sqrt(-1))*sin((2*%pi*1)/6);
9 W62=(cos((2*%pi*2)/6))-(sqrt(-1))*sin((2*%pi*2)/6);
10 W63=cos((2*%pi*3)/6)-int((sqrt(-1))*sin((2*%pi*3)/6)
     );
11 W64=cos((2*%pi*4)/6)-(sqrt(-1))*(sin((2*%pi*4)/6));
12 W65=cos((2*%pi*5)/6)-(sqrt(-1))*sin((2*%pi*5)/6);
13
14 disp(W60,'W60=')
```

```
15  disp(W61,'W61=')
16  disp(W62,'W62=')
17  disp(W63,'W63=')
18  disp(W64,'W64=')
19  disp(W65,'W65=')
20
21  //W60=W612=W618=W624=W630=W636;//Cyclic property of
       twiddle factor
22  W66=W60;
23  W612=W66;
24  W618=W612;
25  W624=W618;
26  W630=W624;
27  W636=W630;
28
29  //W61=W67=W613=W619=W625 //Cyclic property of
       twiddle factor
30  W67=W61;
31  W613=W67;
32  W619=W613;
33  W625=W619;
34
35  //W62=W68=W614=W620 //Cyclic property of twiddle
       factor
36  W68=W62;
37  W614=W68;
38  W620=W614;
39
40  //W63=W69=W615=W621 //Cyclic property of twiddle
       factor
41  W69=W63;
42  W615=W69;
43  W621=W615;
44
45  //W64=W610=W616=W622 //Cyclic property of twiddle
       factor
46  W610=W64;
47  W616=W610;
```

```
48  W622=W616;
49
50  //W65=W613=W621=W629 //Cyclic property of twiddle
        factor
51  W613=W65;
52  W621=W613;
53  W629=W621;
54
55  W6= [W60 W60 W60 W60 W60 W60;W60 W61 W62 W63 W64 W65
        ;W60 W62 W64 W66 W68 W610;W60 W63 W66 W69 W612
        W615;W60 W64 W68 W612 W616 W620;W60 W65 W610 W615
         W620 W625];
56  //Displaying the W6 matrix :
57  disp(W6,'[W6]=')
```

Experiment: 8

Derive The Eight Point Twiddle Factor Matrix For Computing Inverse Dft

Scilab code Solution 8.0 Experiment Number 8

```
1  //AIM: Derive the 8 point twiddle factor matrix for
       computing inverse DFT
2
3  // i.e. W8* matrix derivation
4
5  //Software version Scilab 5.5.2
6  //OS windows 10
7  clc;
8  clear;
9  //Computing the twiddle factor values for W80,W81,
       W82,W83,W84,W85,W86,W87 :
10 W80=int(cos(0))+(sqrt(-1)*int(sin(0)));
11 W81=cos((2*%pi*1)/8)+(sqrt(-1))*sin((2*%pi*1)/8);
12 W82=int(cos((2*%pi*2)/8))+(sqrt(-1))*sin((2*%pi*2)
       /8);
13 W83=cos((2*%pi*3)/8)+(sqrt(-1))*sin((2*%pi*3)/8);
14 W84=cos((2*%pi*4)/8)+(sqrt(-1))*int(sin((2*%pi*4)/8)
```

```scilab
     );
15  W85=cos((2*%pi*5)/8)+(sqrt(-1))*sin((2*%pi*5)/8);
16  W86=int(cos((2*%pi*6)/8))+(sqrt(-1))*sin((2*%pi*6)
     /8);
17  W87=cos((2*%pi*7)/8)+(sqrt(-1))*sin((2*%pi*7)/8);
18
19  disp(W80,'W80=')
20  disp(W81,'W81=')
21  disp(W82,'W82=')
22  disp(W83,'W83=')
23  disp(W84,'W84=')
24  disp(W85,'W85=')
25  disp(W86,'W86=')
26  disp(W87,'W87=')
27
28  //W80=W88=W816=W824=W832=W840=W848;//Cyclic property
        of twiddle factor
29  W88=W80;
30  W816=W88;
31  W824=W816;
32  W832=W824;
33  W840=W832;
34  W848=W840;
35
36  //W81=W89=W817=W825=W833=W841=W849;//Cyclic property
        of twiddle factor
37  W89=W81;
38  W817=W89;
39  W825=W817;
40  W833=W825;
41  W841=W833;
42  W849=W841;
43
44  //W82=W810=W818=W826=W834=W842;//Cyclic property of
     twiddle factor
45  W810=W82;
46  W818=W810;
47  W826=W818;
```

```
48  W834=W826;
49  W842=W834;
50
51  //W83=W811=W819=W827=W835=W843;//Cyclic property of
        twiddle factor
52  W811=W83;
53  W819=W811;
54  W827=W819;
55  W835=W827;
56  W843=W835;
57
58  //W84=W812=W820=W828=W836=W844;//Cyclic property of
        twiddle factor
59  W812=W84;
60  W820=W812;
61  W828=W820;
62  W836=W828;
63  W844=W836;
64
65  //W85=W813=W821=W829=W837=W845;//Cyclic property of
        twiddle factor
66  W813=W85;
67  W821=W813;
68  W829=W821;
69  W837=W829;
70  W845=W837;
71
72  //W86=W814=W822=W830=W838=W846;//Cyclic property of
        twiddle factor
73  W814=W86;
74  W822=W814;
75  W830=W822;
76  W838=W830;
77  W846=W838;
78
79  //W87=W815=W823=W831=W839=W847;//Cyclic property of
        twiddle factor
80  W815=W87;
```

```
81  W823=W815;
82  W831=W823;
83  W839=W831;
84  W847=W839;
85
86  W8_star= [W80 W80 W80 W80 W80 W80 W80 W80;W80 W81
        W82 W83 W84 W85 W86 W87;W80 W82 W84 W86 W88 W810
        W812 W814;W80 W83 W86 W89 W812 W815 W818 W821;W80
         W84 W88 W812 W816 W820 W824 W828;W80 W85 W810
        W815 W820 W825 W830 W835;W80 W86 W812 W818 W824
        W830 W836 W842;W80 W87 W814 W821 W828 W835 W842
        W849];
87
88  disp(W8_star,'[W8*]=')
```

Experiment: 9

Filtering Of Long Data Sequences

Scilab code Solution 9.0 Experiment Number 9

```scilab
1  //AIM: Filtering of long data sequences
2
3  //Overlap-add method using FFT-IFFT technique(
       without using inbuilt Scilab
4  //functions for DFT/IDFT or FFT/IFFT)
5
6  //Software version Scilab 5.5.2
7  //OS windows 10
8  clc;
9  clear;
10 //Given that x(n)={1,2,3,4,5,6,7,8} and h(n)={1,2}
11 //Soln:
12 xn =[1 2 3 4 5 6 7 8]; //Nx=8
13 hn =[1 2];
14 //Since Nx=8,Nh=2 and we know Nx=m*Nh(so 8=m*2)
       giving m=4;and so x(n) has been
15 //partitioned into 4 blocks of length Nh=2
16 x0n =[1 2];
17 x1n =[3 4];
```

```
18  x2n =[5 6];
19  x3n =[7 8];
20  //Length  of  each  partitioned  input  i.e  x0n,x1n,x2n
        and  x3n  is  2(i.e.  L=2)
21  //Here  length  of  impulse  response  array  h(n)  is  2  (i
        .e.  M=2)  or  Nh=2
22  //Hence,we  pad  the  partitioned  sequence  and  h(n)
        with  zeros  such  that  the  length
23  //  of  each  one  becomes  'atleast'  L+M-1  i.e.  3
24  //But  since  we  know  4  point  DFT  matrix,we  can  pad  an
        additional  zero  &  make  the
25  //length  of  each  sequence  =4
26  x0n_z=[1 2 0 0];//x0n_z  represents  zero  pad  is  done
        to  x0n
27  x1n_z=[3 4 0 0];//x1n_z  represents  zero  pad  is  done
        to  x1n
28  x2n_z=[5 6 0 0];//x2n_z  represents  zero  pad  is  done
        to  x2n
29  x3n_z=[7 8 0 0];//x3n_z  represents  zero  pad  is  done
        to  x3n
30  hn_z=[1 2 0 0];//hn_z  represents  zero  pad  is  done  to
        hn
31
32  //Computing  FFT  for  x0n_z
33  x0n_z_0=1;//x0n_z_0  represents  the  0th  sample  of
        x0n_z
34  x0n_z_1=2;//x0n_z_1  represents  the  1st  sample  of
        x0n_z
35  x0n_z_2=0;//x0n_z_2  represents  the  2nd  sample  of
        x0n_z
36  x0n_z_3=0;//x0n_z_3  represents  the  3rd  sample  of
        x0n_z
37  X0_0=(x0n_z_2+x0n_z_0)*(1)+(x0n_z_3+x0n_z_1)*(1)
38  X0_1=(x0n_z_3-x0n_z_1)*(-1)*(-sqrt(-1))+(x0n_z_2-
        x0n_z_0)*(-1);
39  X0_2=((x0n_z_3+x0n_z_1)*(1)-(x0n_z_2+x0n_z_0)*(1))
        *(-1);
40  X0_3=((x0n_z_3-x0n_z_1)*(-1)*(-sqrt(-1))-(x0n_z_2-
```

```
      x0n_z_0 ) * ( -1) ) * ( -1) ;
41  disp ({ , X0_0 , X0_1 , X0_2 , X0_3 ,} , 'So , the  DFT  of  x0 ( n )
      using  Decimation - in - Time  Fast  Fourier  Transform (
      DIT - FFT )  is  X0 ( k )= ')
42
43  // Computing  FFT  for  x1n_z
44  x1n_z_0 =3; // x1n_z_0  represents  the  0th  sample  of
      x1n_z
45  x1n_z_1 =4; // x1n_z_1  represents  the  1st  sample  of
      x1n_z
46  x1n_z_2 =0; // x1n_z_2  represents  the  2nd  sample  of
      x1n_z
47  x1n_z_3 =0; // x1n_z_3  represents  the  3rd  sample  of
      x1n_z
48  X1_0 =( x1n_z_2 + x1n_z_0 ) * (1) +( x1n_z_3 + x1n_z_1 ) * (1)
49  X1_1 =( x1n_z_3 - x1n_z_1 ) * ( -1) * ( - sqrt ( -1) ) +( x1n_z_2 -
      x1n_z_0 ) * ( -1) ;
50  X1_2 =(( x1n_z_3 + x1n_z_1 ) * (1) -( x1n_z_2 + x1n_z_0 ) * (1) )
      * ( -1) ;
51  X1_3 =(( x1n_z_3 - x1n_z_1 ) * ( -1) * ( - sqrt ( -1) ) -( x1n_z_2 -
      x1n_z_0 ) * ( -1) ) * ( -1) ;
52  disp ({ , X1_0 , X1_1 , X1_2 , X1_3 ,} , 'So , the  DFT  of  x1 ( n )
      using  Decimation - in - Time  Fast  Fourier  Transform (
      DIT - FFT )  is  X1 ( k )= ')
53
54  // Computing  FFT  for  x2n_z
55  x2n_z_0 =5; // x2n_z_0  represents  the  0th  sample  of
      x2n_z
56  x2n_z_1 =6; // x2n_z_1  represents  the  1st  sample  of
      x2n_z
57  x2n_z_2 =0; // x2n_z_2  represents  the  2nd  sample  of
      x2n_z
58  x2n_z_3 =0; // x2n_z_3  represents  the  3rd  sample  of
      x2n_z
59  X2_0 =( x2n_z_2 + x2n_z_0 ) * (1) +( x2n_z_3 + x2n_z_1 ) * (1)
60  X2_1 =( x2n_z_3 - x2n_z_1 ) * ( -1) * ( - sqrt ( -1) ) +( x2n_z_2 -
      x2n_z_0 ) * ( -1) ;
61  X2_2 =(( x2n_z_3 + x2n_z_1 ) * (1) -( x2n_z_2 + x2n_z_0 ) * (1) )
```

```
          *( -1) ;
62  X2_3 =(( x2n_z_3 - x2n_z_1 )*( -1)*( -sqrt ( -1))-( x2n_z_2 -
          x2n_z_0 )*( -1))*( -1) ;
63  disp ({ , X2_0 , X2_1 , X2_2 , X2_3 ,} , 'So , the  DFT  of  x2 ( n )
          using  Decimation -in -Time  Fast  Fourier  Transform (
          DIT -FFT)  is  X2(k)= ')
64
65  // Computing  FFT  for  x3n_z
66  x3n_z_0 =7; // x3n_z_0  represents  the  0th  sample  of
          x3n_z
67  x3n_z_1 =8; // x3n_z_1  represents  the  1st  sample  of
          x3n_z
68  x3n_z_2 =0; // x3n_z_2  represents  the  2nd  sample  of
          x3n_z
69  x3n_z_3 =0; // x3n_z_3  represents  the  3rd  sample  of
          x3n_z
70  X3_0 =( x3n_z_2 + x3n_z_0 )*(1) +( x3n_z_3 + x3n_z_1 )*(1)
71  X3_1 =( x3n_z_3 - x3n_z_1 )*( -1)*( -sqrt ( -1)) +( x3n_z_2 -
          x3n_z_0 )*( -1) ;
72  X3_2 =(( x3n_z_3 + x3n_z_1 )*(1) -( x3n_z_2 + x3n_z_0 )*(1) )
          *( -1) ;
73  X3_3 =(( x3n_z_3 - x3n_z_1 )*( -1)*( -sqrt ( -1))-( x3n_z_2 -
          x3n_z_0 )*( -1))*( -1) ;
74  disp ({ , X3_0 , X3_1 , X3_2 , X3_3 ,} , 'So , the  DFT  of  x3 ( n )
          using  Decimation -in -Time  Fast  Fourier  Transform (
          DIT -FFT)  is  X3(k)= ')
75
76  // Computing  FFT  for  hn_z
77  hn_z_0 =1; // hn_z_0  represents  the  0th  sample  of  hn_z
78  hn_z_1 =2; // hn_z_1  represents  the  1st  sample  of  hn_z
79  hn_z_2 =0; // hn_z_2  represents  the  2nd  sample  of  hn_z
80  hn_z_3 =0; // hn_z_3  represents  the  3rd  sample  of  hn_z
81  Hk_0 =( hn_z_2 + hn_z_0 )*(1) +( hn_z_3 + hn_z_1 )*(1)
82  Hk_1 =( hn_z_3 - hn_z_1 )*( -1)*( -sqrt ( -1)) +( hn_z_2 - hn_z_0
          )*( -1) ;
83  Hk_2 =(( hn_z_3 + hn_z_1 )*(1) -( hn_z_2 + hn_z_0 )*(1) )*( -1) ;
84  Hk_3 =(( hn_z_3 - hn_z_1 )*( -1)*( -sqrt ( -1))-( hn_z_2 -
          hn_z_0 )*( -1))*( -1) ;
```

```
85  disp({,Hk_0,Hk_1,Hk_2,Hk_3,},'So,the DFT of h(n)
        using Decimation-in-Time Fast Fourier Transform(
        DIT-FFT) is H(k)=')
86
87  Y0k0=(X0_0).*(Hk_0);
88  Y0k1=(X0_1).*(Hk_1);
89  Y0k2=(X0_2).*(Hk_2);
90  Y0k3=(X0_3).*(Hk_3);
91  Y0k=[Y0k0 Y0k1 Y0k2 Y0k3];
92  disp(Y0k,'Y0(k)=')
93
94  //Computing IDFT of Y0k using IDIT-FFT :
95
96  Y0k0_0c=real(Y0k0)-(sqrt(-1))*imag(Y0k0);//Y0K0_0c
        means complex conjugate of Y0k0
97  Y0k0_1c=real(Y0k1)-(sqrt(-1))*imag(Y0k1);//Y0K0_1c
        means complex conjugate of Y0k1
98  Y0k0_2c=real(Y0k2)-(sqrt(-1))*imag(Y0k2);//Y0K0_2c
        means complex conjugate of Y0k2
99  Y0k0_3c=real(Y0k3)-(sqrt(-1))*imag(Y0k3);//Y0K0_3c
        means complex conjugate of Y0k3
100
101 a0=(((Y0k0_3c+Y0k0_1c)*(1))+((Y0k0_2c+Y0k0_0c)*(1)))
        *(1/4);
102 a1=(((Y0k0_3c-Y0k0_1c)*(-1)*(-sqrt(-1)))+(Y0k0_2c-
        Y0k0_0c)*(-1))*(1/4);
103 a2=(((Y0k0_3c+Y0k0_1c)*(1)-(Y0k0_2c+Y0k0_0c))*(-1))
        *(1/4);
104 a3=((((Y0k0_3c-Y0k0_1c)*(-1)*(-sqrt(-1)))-((Y0k0_2c-
        Y0k0_0c)*(-1)))*(-1))*(1/4);
105
106 a0_real=real(a0);
107 a0_conj=(-1)*(imag(a0));
108 a1_real=real(a1);
109 a1_conj=(-1)*(imag(a1));
110 a2_real=real(a2);
111 a2_conj=(-1)*(imag(a2));
112 a3_real=real(a3);
```

```
113  a3_conj =(-1)*(imag(a3));
114  //Finally, we will add the real part and the
         imaginary part whose complex
115  //conjugate is taken
116  a0'=a0_real+a0_conj
117  a1'=a1_real+a1_conj
118  a2'=a2_real+a2_conj
119  a3'=a3_real+a3_conj
120  a=[a0',a1',a2',a3']
121  disp(a,'So, the IDFT of Y0(k) using Inverse
         Decimation-in-Time Fast Fourier Transform(IDIT-
         FFT) is y0(n)=')
122
123  Y1k0=(X1_0).*(Hk_0);
124  Y1k1=(X1_1).*(Hk_1);
125  Y1k2=(X1_2).*(Hk_2);
126  Y1k3=(X1_3).*(Hk_3);
127  Y1k=[Y1k0 Y1k1 Y1k2 Y1k3];
128  disp(Y1k,'Y1(k)=')
129
130  //Computing IDFT of Y1k using IDIT-FFT :
131  Y1k0_0c=real(Y1k0)-(sqrt(-1))*imag(Y1k0);//Y1K0_0c
         means complex conjugate of Y1k0
132  Y1k0_1c=real(Y1k1)-(sqrt(-1))*imag(Y1k1);//Y1K0_1c
         means complex conjugate of Y1k1
133  Y1k0_2c=real(Y1k2)-(sqrt(-1))*imag(Y1k2);//Y1K0_2c
         means complex conjugate of Y1k2
134  Y1k0_3c=real(Y1k3)-(sqrt(-1))*imag(Y1k3);//Y1K0_3c
         means complex conjugate of Y1k3
135
136  b0=(((Y1k0_3c+Y1k0_1c)*(1))+((Y1k0_2c+Y1k0_0c)*(1)))
         *(1/4);
137  b1=(((Y1k0_3c-Y1k0_1c)*(-1)*(-sqrt(-1)))+(Y1k0_2c-
         Y1k0_0c)*(-1))*(1/4);
138  b2=(((Y1k0_3c+Y1k0_1c)*(1)-(Y1k0_2c+Y1k0_0c))*(-1))
         *(1/4);
139  b3=((((Y1k0_3c-Y1k0_1c)*(-1)*(-sqrt(-1)))-((Y1k0_2c-
         Y1k0_0c)*(-1)))*(-1))*(1/4);
```

```
140
141  b0_real=real(b0);
142  b0_conj=(-1)*(imag(b0));
143  b1_real=real(b1);
144  b1_conj=(-1)*(imag(b1));
145  b2_real=real(b2);
146  b2_conj=(-1)*(imag(b2));
147  b3_real=real(b3);
148  b3_conj=(-1)*(imag(b3));
149  //Finally, we will add the real part and the
          imaginary part whose complex
150  //conjugate is taken
151  b0'=b0_real+b0_conj
152  b1'=b1_real+b1_conj
153  b2'=b2_real+b2_conj
154  b=[b0',b1',b2',b3']
155  disp(b,'So,the IDFT of Y1(k) using Inverse
          Decimation-in-Time Fast Fourier Transform(IDIT-
          FFT) is y1(n)=')
156
157  Y2k0=(X2_0).*(Hk_0);
158  Y2k1=(X2_1).*(Hk_1);
159  Y2k2=(X2_2).*(Hk_2);
160  Y2k3=(X2_3).*(Hk_3);
161  Y2k=[Y2k0 Y2k1 Y2k2 Y2k3];
162  disp(Y2k,'Y2(k)=')
163
164  //Computing IDFT of Y2k using IDIT-FFT :
165  Y2k0_0c=real(Y2k0)-(sqrt(-1))*imag(Y2k0);//Y2K0_0c
          means complex conjugate of Y2k0
166  Y2k0_1c=real(Y2k1)-(sqrt(-1))*imag(Y2k1);//Y2K0_1c
          means complex conjugate of Y2k1
167  Y2k0_2c=real(Y2k2)-(sqrt(-1))*imag(Y2k2);//Y2K0_2c
          means complex conjugate of Y2k2
168  Y2k0_3c=real(Y2k3)-(sqrt(-1))*imag(Y2k3);//Y2K0_3c
          means complex conjugate of Y2k3
169
170  c0=(((Y2k0_3c+Y2k0_1c)*(1))+((Y2k0_2c+Y2k0_0c)*(1)))
```

```
    *(1/4) ;
171 c1 =((( Y2k0_3c - Y2k0_1c )*( -1)*( -sqrt ( -1)))+( Y2k0_2c -
    Y2k0_0c )*( -1))*(1/4) ;
172 c2 =((( Y2k0_3c + Y2k0_1c )*(1) -( Y2k0_2c + Y2k0_0c ))*( -1))
    *(1/4) ;
173 c3 =(((( Y2k0_3c - Y2k0_1c )*( -1)*( -sqrt ( -1))) -(( Y2k0_2c -
    Y2k0_0c )*( -1)))*( -1))*(1/4) ;
174
175 c0_real = real ( c0 );
176 c0_conj =( -1)*( imag ( c0 ));
177 c1_real = real ( b1 );
178 c1_conj =( -1)*( imag ( c1 ));
179 c2_real = real ( c2 );
180 c2_conj =( -1)*( imag ( c2 ));
181 c3_real = real ( c3 );
182 c3_conj =( -1)*( imag ( c3 ));
183 // Finally , we will add the real part and the
        imaginary part whose complex
184 // conjugate is taken
185 c0 '= c0_real + c0_conj
186 c1 '= c1_real + c1_conj
187 c2 '= c2_real + c2_conj
188 c3 '= c3_real + c3_conj
189 c =[ c0 ', c1 ', c2 ', c3 ']
190 disp ( c , 'So , the IDFT of Y2 ( k ) using Inverse
        Decimation -in -Time Fast Fourier Transform ( IDIT -
        FFT ) is y2 ( n )=')
191
192 Y3k0 =( X3_0 ).*( Hk_0 );
193 Y3k1 =( X3_1 ).*( Hk_1 );
194 Y3k2 =( X3_2 ).*( Hk_2 );
195 Y3k3 =( X3_3 ).*( Hk_3 );
196 Y3k =[ Y3k0 Y3k1 Y3k2 Y3k3 ];
197 disp ( Y3k , 'Y3 ( k )=')
198
199 // Computing IDFT of Y3k using IDIT -FFT :
200 Y3k0_0c = real ( Y3k0 ) -( sqrt ( -1))* imag ( Y3k0 ); // Y3K0_0c
        means complex conjugate of Y3k0
```

```
201  Y3k0_1c=real(Y3k1)-(sqrt(-1))*imag(Y3k1);//Y3K0_1c
         means  complex  conjugate  of  Y3k1
202  Y3k0_2c=real(Y3k2)-(sqrt(-1))*imag(Y3k2);//Y3K0_2c
         means  complex  conjugate  of  Y3k2
203  Y3k0_3c=real(Y3k3)-(sqrt(-1))*imag(Y3k3);//Y3K0_3c
         means  complex  conjugate  of  Y3k3
204
205  d0=(((Y3k0_3c+Y3k0_1c)*(1))+((Y3k0_2c+Y3k0_0c)*(1)))
         *(1/4);
206  d1=(((Y3k0_3c-Y3k0_1c)*(-1)*(-sqrt(-1)))+(Y3k0_2c-
         Y3k0_0c)*(-1))*(1/4);
207  d2=(((Y3k0_3c+Y3k0_1c)*(1)-(Y3k0_2c+Y3k0_0c))*(-1))
         *(1/4);
208  d3=((((Y3k0_3c-Y3k0_1c)*(-1)*(-sqrt(-1)))-((Y3k0_2c-
         Y3k0_0c)*(-1)))*(-1))*(1/4);
209
210  d0_real=real(d0);
211  d0_conj=(-1)*(imag(d0));
212  d1_real=real(d1);
213  d1_conj=(-1)*(imag(d1));
214  d2_real=real(d2);
215  d2_conj=(-1)*(imag(d2));
216  d3_real=real(d3);
217  d3_conj=(-1)*(imag(d3));
218  //Finally , we will  add  the  real  part  and  the
         imaginary  part  whose  complex
219  //conjugate  is  taken
220  d0'=d0_real+d0_conj
221  d1'=d1_real+d1_conj
222  d2'=d2_real+d2_conj
223  d3'=d3_real+d3_conj
224  d=[d0',d1',d2',d3']
225  disp(d,'So,the  IDFT  of  Y3(k)  using  Inverse
         Decimation-in-Time  Fast  Fourier  Transform(IDIT-
         FFT)  is  y3(n)=')
226  w=[a 0 0 0 0 0 0];
227  x=[0 0 b 0 0 0 0];
228  y=[0 0 0 0 c 0 0];
```

```
229  z=[0 0 0 0 0 0 d];
230  disp(z,y,x,w,'After overlapping ,the sequences will
         be seen as follows:')
231  yn=w+x+y+z;
232  disp(yn,'The output : y(n)=')
```

Experiment: 10

Implement Impulse Invariant Method

Scilab code Solution 10.0 Experiment Number 10

```
1  //AIM:  Implement  Impulse  Invariant  method
2
3  //Find  out  H(z)  using  impulse  invariances  method  at
       5Hz  sampling  frequency
4  //from  H(s)  where  H(s)=1/(s+1)(s+2)
5
6  //Software  version  Scilab  5.5.2
7  //OS  windows  10
8  clc;
9  clear;
10  s=%s;
11  s2=-2;
12  s1=-1;
13  d1=(s-s1);
14  p2=(s-s2);
15  if (s1) then    // When  pole=-1
16      s1=-1;
17      s2=-2;
18      s=s1;
```

```
19      d2=(s-s2);
20      num1=1/d2;    // Value of A1
21 h1=syslin('c', num1/d1)
22 end
23 disp(h1)
24 disp(num1,'value of A1=')
25 if (s2) then    // When pole=-2
26      s1=-1;
27      s2=-2;
28      s=s2;
29      p1=(s-s1);
30      num2=1/p1;    // Value of A2
31 h2=syslin('c', num2/p2)
32 end
33 disp(h2)
34 disp(num2,'Value of A2=')
35 Hs=(h1)+(h2);
36 disp(Hs,'Transfer function of analog filter H(s)=')
37 //Obtain the Z-transform using impulse invariance
       transformation equation
38 //1/(s-pk)=1/[1-exp(pk*Ts)*Z^(-1)]
39 Fs=5;
40 Ts=1/Fs;
41 disp('sec',Ts,'Sampling time Ts=')
42 //we have poles at s1=-1 and s2=-2
43 Z=poly(0,"Z")
44 //1/(s+1)=a;We consider
45 a=num1/(1-exp(s1*(Ts))*Z^(-1));
46 //1/(s+2)=b;We consider
47 b=num2/(1-exp(s2*(Ts))*Z^(-1));
48 disp(a,'1/s+1=')
49 disp(b,'1/s+2=')
50 //The Transfer function of digital filter is given
       by,
51 //H(Z)=   _   (k=1)^N(Ak/(1-e^(pk*Ts)*Z^(-1) )
52 //H(Z)=A1/1-exp(p1*Ts)*Z^(-1)+A2/1-exp(p1*Ts)*Z^(-1)
53 Hz=(a+b);
54 disp(Hz,'The required transfer function for digital
```

IIR filter H(Z)=')

Experiment: 11

To Design Butterworth Filter With Minimum Readymade Scilab Functions

Scilab code Solution 11.0 Experiment Number 11

```
1  //AIM:To design Butterworth filter with minimum
       readymade Scilab functions
2
3  //To compute the order and the poles of Butterworth
       low pass filter using
4  //Bilinear transformation(ASSUME T=1SEC);
5  //Attenuation in passband=1.93dB
6  //Attenuation in stopband=13.97dB
7  //Passband edge frequency=0.2
8  //Stopbandband edge frequency=0.6
9
10 //Software version Scilab 5.5.2
11 //OS windows 10
12 clc;
13 clear;
14 s=poly(0,"s")
15 T=1;
```

```
16  Ap=1.93; //in dB
17  As=13.97; //in dB
18  wp=0.2*(%pi)
19  ws=0.6*(%pi)
20  ohmp=2/T*(tan(wp/2))
21  ohms=2/T*(tan(ws/2))
22  //ORDER CALCULATION :
23  N=(0.5)*(log((((10^(0.1*As))-1)/((10^(0.1*Ap))-1))))
        /(log(ohms/ohmp))
24
25  Nr=int (N)
26  x=N-int(N)
27  if(x>0)
28    Nr=Nr+1
29  ohmc=(ohmp/(10^(0.1*Ap)-1)^(1/(2*Nr)))
30  //Calculation of poles
31  i=0:1:Nr-1;
32  pi_plus=ohmc*exp(%i*(Nr+2*i+1)*(%pi)/(2*Nr))
33  pi_minus=-ohmc*exp(%i*(2+2.*i+1)*(%pi)/(2*Nr))
34  disp(wp,'wp=')
35  disp(ws,'ws=')
36  disp(ohmp,'ohmp=')
37  disp(ohms,'ohms=')
38  disp(N,'N=')
39  disp(Nr,'Roundoff value of N now denoted as Nr =')
40  disp(ohmc,'Cutoff frequency : ohmc=')
41  disp('Displaying the poles')
42  disp(pi_plus,'pi_plus=')
43  disp(pi_minus,'pi_minus=')
44  h2=zeros(1,2)
45  h=ohmc/(s-(-0.53-0.53*%i))
46  h1=ohmc/(s-(-0.53+0.53*%i))
47  h2=h*h1;
48  disp(h,h1,'Now the analog transfer function H(s) is
        the multiplication of the following two terms :')
        ;
49  disp(h2,'After multiplication ,H(s)=')
50  g=numer(h2);
```

```
51  disp(g,'Numerator of the analog transfer function=')
52  //Obtaining H(z) using Bilinear Transformation
       Method :
53  z=poly(0,"z")
54  s=(2/T)*((z-1)/(z+1));//Bilinear Transformation
       Method
55  disp('Type resume in Console')
56  pause
57  a=0.5618 +1.06*s+s^2;
58  b=(1/a)
59  c=0.5645360*b;
60  disp(c,'The digital transfer function H(z)=')
```

Experiment: 12

To Design Chebyshev Filter With Minimum Readymade Scilab Functions.

Scilab code Solution 12.0 Experiment Number 12

```
1  //AIM:To design Chebyshev filter with minimum
        readymade Scilab functions
2
3  //Design of low pass 1 rad/sec bandwidth Chebyshev
        filter
4  //Acceptable passband ripple=2 db
5  //cut off radian frequency 1 radian/sec
6  //stop band attenuation of 20db or greater beyond
        1.3 radian/sec
7
8  //Software version Scilab 5.5.2
9  //OS windows 10
10 clc;
11 clear;
12 //Given cut-off frequency is 1 rad/sec. This means
        that it is a normalized low
13 //pass Chebyshev filter
```

```
14  fc=1;//from the above question
15  Ap=2;//from the above question
16  As=20;//from the above question
17  ohms=1.3;//from the above question
18  ohmp=1;//It is a normalized filter.Hence the value
        is 1 rad/sec
19
20  //Steps for calculation of
21  a=(0.1*2)
22  b=10^a
23  c=b-1
24  episelon=c^(1/2)
25  disp(episelon,"  =")
26
27  //Steps for calculation of order
28  d=(-20)*((log(episelon)/log(10)))
29  e=(20)*(log(ohms)/log(10))
30  f=6+e
31  g=26+d
32  N=g/f
33  Nr=int (N)
34  x=N-int(N)
35  if(x>0)
36   Nr=Nr+1
37  //N=(((-20)*log(episelon))+6+20)/(6+20*(log(ohms)))
38  D=-20;//given in db
39  //N=-(20*log10(episelon))-6(N-1)-(20*log(ohms))
40  h1=((episelon)^2)
41  h2=1+h1
42  h3=h2^(1/2)
43  h4=h3+1
44  g=(episelon)
45  h5=h4/g
46  beta=(h5)^(1/Nr)
47  disp(N,"N=")
48  disp(Nr,"The round-off value of N(now called as Nr)=
        ")
49  disp(beta,"  =")
```

```scilab
50
// for the determination of poles we have
r=(ohmp)*(((beta^2)-1)/(2*beta))
disp(r,"The minor axis of the ellipse(r)=")
R=ohmp*((beta^2+1)/(2*beta))
disp(R,"The major axis of the ellipse(R)=");
// thetai=(%pi/2)+((2i+%pi)/2*N)// i=0123
// for i=0
theta0=((%pi/2)+((2*0*%pi+%pi)/(2*4)))
disp(theta0,"theta0=")
// for i=1
theta1=((%pi/2)+((2*1*%pi+%pi)/(2*4)))
disp(theta1,"theta1=")
// for i=2
theta2=((%pi/2)+((2*2*%pi+%pi)/(2*4)))
disp(theta2,"theta2=")
// for i=3
theta3=((%pi/2)+((2*3*%pi+%pi)/(2*4)))
disp(theta3,"theta3=")
// the pole position is given by
// sp=r*cos(thetai)+jsin(thetai)
i=0:1:Nr-1
// Computing real and imaginary part of s0,s1,s2,s3
h6=((r)*(cos(theta0))) // Computing real part of s0
h7=(%i)*(R)*(sin(theta0)) // Computing imaginary part
    of s0
s0=h6+h7; // Combining real and imaginary part of s0
disp(s0,"s0=")
h8=((r)*(cos(theta1))) // Computing real part of s1
h9=(%i)*(R)*(sin(theta1)) // Computing imaginary part
    of s1
s1=h8+h9; // Combining real and imaginary part of s1
disp(s1,"s1=")
h10=((r)*(cos(theta2))) // Computing real part of s2
h11=(%i)*(R)*(sin(theta2)) // Computing imaginary
    part of s2
s2=h10+h11; // Combining real and imaginary part of
    s2
```

```
84  disp(s2 ,"s2=")
85  h12=((r)*(cos(theta3)))  //Computing real part of s3
86  h13=(%i)*(R)*(sin(theta3))  //Computing imaginary
        part of s3
87  s3=h12+h13;  //Combining real and imaginary part of
        s13
88  disp(s3 ,"s3=")
89
90  //Calculation of transfer function :
91  s=poly(0 ,"s")
92  h=1/((s-(s0))*(s-(s1))*(s-(s2))*(s-(s3)))
93  disp(h ,"h=")
94  disp('Now type resume and press enter in the Console
         window ')
95  pause
96  //Now value of b0 is required which is nothing but
        the value of the constant
97  //term in the denominator of h (obtained by seeing
        the calculated value in the
98  //console window by inserting a 'pause' in the
        program )
99  b0=0.2057651;
100 //Also we see in the Console window that the rounded
         value of the order is 4
101 //and  since 4 is an even number ,so the formula for
        calculation of i will be
102 //  i=b0/(sqrt(1+   ^2))
103 i=b0/(sqrt(1+(episelon)^2))
104 disp(i ,'i=')
105 Hs=i*h//Calculated value of Ha(s)
106 disp(Hs ,'The required transfer function Ha(s)=')  //
        Displaying the calculated value of the transfer
        function
```

Experiment: 13

Designing Two Stage Decimator

Scilab code Solution 13.0 Experiment Number 13

```
1 //AIM: To implement a two stage decimator for the
       following specifications:
2 // Sampling rate of the input signal=20,000 Hz
3 //D=100,Passband=0 to 40Hz,Transition Band=40 to 50
      Hz,Passband ripple=0.02,Stopband ripple=0.002
4
5 //Software version Scilab 5.5.2
6 //OS windows 10
7 clc;
8 clear;
9
10 // Let us consider the combination 25*4
11 D1=25;
12 D2=4;
13 //such that D=D1*D2
14 F0=20000;//Fo is the sampling frequency(or sampling
      rate) of the input signal(given in the question)
15 Fp=40; //Fp is the passband edge frequency(given in
      the question)
```

```scilab
16  Fstop =50;  // Fstop  is  the  stopband  edge  frequency (
        given  in  the  question )
17  F1 = F0 / D1 ;
18  F2 = F1 / D2 ;
19  disp ( " Hertz " ,F0 , " The  given  value  of  F0=" )
20  disp ( " Hertz " ,F1 , " The  calculated  value  of  F1=" )
21  disp ( " Hertz " ,F2 , " The  calculated  value  of  F2=" )
22  Dp =0.02 ;
23  Ds =0.002 ;
24  //  calculations  for  stage  I ...
25  // Step  1: Passband :0 <= F <= Fp  that  means  0 <=  F <=  40
        Hertz
26  // Step  2: Stopband :  Fi - Fstop <= F <= F ( i -1)/2
27  // Here  i =1  for  Stage -I
28  LROS1 = F1 - Fstop ;  // Here ,  LROS1= Lowest  limit  of
        stopband  for  Stage -I
29  HROS1 = F0 /2; // Here ,  HROS1= Highest  limit  of  stopband
        for  Stage -I
30  disp ( " Hertz " ,LROS1 , " The  lowest  limit  of  stopband ( for
         Stage -I )  =" )
31  disp ( " Hertz " ,HROS1 , " The  highest  limit  of  stopband (
        for  Stage -I )  =" )
32  disp ( " Hertz " ,HROS1 , " <= F <=" , " Hertz " ,LROS1 , " So  the
        range  of  stopband  for  Stage  I  is : " , )
33  Tmax1 = LROS1 ;
34  Tmin1 = Fp ;
35  DF1 =( Tmax1 - Tmin1 )/ F0
36  Dp1 =( Dp /2) ;
37  Ds1 = Ds ;
38  disp ( ' Hertz ' ,DF1 , " The  calculated  value  of   F1  =" )
39  disp ( Dp1 , " The  calculated  value  of   p1  =" )
40  disp ( Ds1 , " The  value  of   s1  =  s  =" )
41  N1 =((( -10* log10 ( Dp1 * Ds1 ) -13) /(14.6* DF1 ) )+1) ;  //
        Computing  the  filter  length ( N1 )  of  Stage -I
42  disp ( N1 , " Filter  length ,N1=" )
43  NR1 = int ( N1 ) ; // Extracting  only  the  integer  part  from
        N1
44  x1 = N1 - int ( N1 ) ; // x1  is  the  decimal  part  of  overall  N1
```

```scilab
45  if (x1 >0) // If the decimal part is greater than zero
46      NR1 = NR1 +1 // Then increment the extracted integer
              part i.e. NR1 by 1 to get a round−off value
           of the length of filter of Stage−I
47  disp(NR1 ," Filter length N1 (round−off value )now known
        as NR1=")
48  // calculations for stage−II ...
49  // Step 1: Passband:0<=F<=Fp that means 0<= F<= 40
       Hertz
50  // Step 2: Stopband : Fi−Fstop<=F<=F(i −1)/2
51  // Here i=2 for Stage−II
52  disp("−−−−−−−−−−−−−−−Now displaying the values for
        stage−II−−−−−−−−−−−−−−")
53  LROS2 = F2 - Fstop ; // Here , LROS2=Lowest limit of
        stopband for Stage−II
54  HROS2 = F1/2 // Here , HROS2=Highest limit of stopband
        for Stage−II
55  disp(" Hertz" , LROS2 ," The lowest limit of stopband ( for
        Stage−II ) =")
56  disp(" Hertz" , HROS2 ," The highest limit of stopband (
        for Stage−II ) =")
57  disp(" Hertz" , HROS2 ,"<=F<=" ," Hertz" , LROS2 ," So the
        range of stopband for Stage−II is :" ,)
58  // If transition band is given in the question , then
        always given transition width is applicable for
        the second stage .
59  // Given transition width is 40Hz to 50Hz .
60  // It indicates that for this stage , the stopband
        should start at 50Hz .
61  a =50;
62  disp(" Hertz" , a ," The new value of the lowest limit of
        stopband ( for Stage−II ) =")
63  disp(" Hertz" , HROS2 ," The highest limit of stopband is
        re−written ( for Stage−II ) which is =")
64  disp(" Hertz" , HROS2 ,"<=F<=" ," Hertz" , a ," So the new
        modified range of stopband for Stage−II is :" ,)
65  Tmax2 =50;
66  Tmin2 = Fp ;
```

```
67  DF2=(Tmax2-Tmin2)/F1
68  Dp2=(Dp/2);
69  Ds2=Ds;
70  disp('Hertz',DF2,"The calculated value of   F2  =")
71  disp(Dp2,"The calculated value of   p2  =")
72  disp(Ds2,"The value of   s2  = s1  = s  =")
73  N2=(((-10*log10(Dp2*Ds2)-13)/(14.6*DF2))+1);//
        Computing the filter length (N2) of Stage-II
74  disp(N2,"Filter length ,N2=")
75  NR2=int(N2);//Extracting only the integer part from
        N2
76  x2=N2-int(N2);//x is the decimal part of overall N2
77  if (x2>0)//If the decimal part is greater than zero
78      NR2=NR2+1 //Then increment the extracted integer
             part i.e. NR2 by 1 to get a round-off value
           of the length of filter of Stage-II
79  disp(NR2,"Final filter length ,N2(round-off value)now
        known as NR2=")
80  //Calculation of MPS(Multiplications per second) and
        TSR(Total Storage requirement)...
81  //MPS=    of [i=1 to I](Ni*Fi)
82  //Here I=Total No. of stages=2
83  MPS=(NR1*F1)+(NR2*F2);
84  disp(MPS,"The value of No. of MPS(Multiplications
        per second)=")
85  //TSR=    of [i=1 to I](Ni)
86  //Here I=Total No. of stages=2
87  TSR=NR1+NR2
88  disp(TSR,"The value of TSR(Total storage requirement
        )=")
```

Experiment: 14

Compute Dft Using Matrix Approach And Then Using Dft Properties.

Scilab code Solution 14.0 Experiment Number 14

```scilab
1  //Compute DFT using matrix approach & then using DFT
        properties.
2  //  (i)  :  x(n)={1,2,3,4), find DFT X(k)
3  //  (ii) :  Using results obtained in part (i) & not
        otherwise,
4  // find DFT of following sequences :
5  //  x1(n)={4,1,2,3}
6  //  x2(n)={2,3,4,1}
7  //  x3(n)={3,4,1,2}
8  //  x4(n)={4,6,4,6}
9  //Software version Scilab 5.5.2
10 //OS windows 10
11 clc;
12 clear;
13 //Let us first define the W4 matrix
14 W4=[1 1 1 1 ;1 -sqrt(-1) -1 sqrt(-1);1 -1 1 -1;1
        sqrt(-1) -1 -sqrt(-1)];
```

```scilab
15  disp(W4,'W4=')
16  //Now let us define the input sequence
17  xn=[1;2;3;4];//The input sequence x(n)has been
        arranged as a column matrix
18  //DFT is obtained by multiplying the twiddle matrix
        W4 and the input sequence
19  Xk=W4*xn;
20  disp(Xk,'DFT : X(k)=')
21  disp('Type resume in console and press enter')
22  pause
23  X0=10
24  X1=-2+2*sqrt(-1);
25  X2=-2
26  X3=-2-2*sqrt(-1);
27
28  //(ii) :x1(n)={4,1,2,3} and x(n)={1,2,3,4}
29  //x1(n) is obtained by delaying x(n) by 1 position
        which means x1(n)=x(n-1)
30  //According to the circular time shift property : x(
        n-l) gives DFT as X(k)*e^(-j*2*%pi*k*l/N)
31  //But l=-1
32  a1=cos(0)-(sqrt(-1)*sin(0));
33  //So,for k=0,
34  X10=X0.*real(a1)-X0.*(sqrt(-1)*imag(a1))
35  disp(X10,'X1(0)=')
36  //So,for k=1,
37  b1=int(cos(%pi/2))-(sqrt(-1)*sin(%pi/2))
38  X11=X1*b1;
39  disp(X11,'X1(1)=')
40  //For k=2,
41  c1=int(cos(%pi))-int((sqrt(-1)*sin(%pi)));
42  X12=X2*c1;
43  disp(X12,'X1(2)=')
44  //For k=3,
45  d1=int(cos((3*%pi)/2))-int((sqrt(-1)*sin((3*%pi)/2))
        );
46  X13=X3*d1;
47  disp(X13,'X1(3)=')
```

```
48  disp({,X10,X11,X12,X13,},'So, X1(k)=')
49
50  // (iii) : Now moving ahead to find X2(k)
51  //x2(n)={2,3,4,1} and x(n)={1,2,3,4}
52  //x2(n) is obtained by advancing x(n) by 1 position
        which means x2(n)=x(n+1)
53  //According to the circular time shift property : x(
        n-l) gives DFT as X(k)*e^(-j*2*%pi*k*l/N)
54  //But l=2
55  a2=cos(0)+(sqrt(-1)*sin(0));
56  //So,for k=0,
57  X20=X0.*real(a2)-X0.*(sqrt(-1)*imag(a2))
58  disp(X20,'X2(0)=')
59  //So,for k=1,
60  b2=int(cos(%pi/2))+(sqrt(-1)*sin(%pi/2))
61  X21=X1*b2;
62  disp(X21,'X2(1)=')
63  //For k=2,
64  c2=int(cos(%pi))+int((sqrt(-1)*sin(%pi)));
65  X22=X2*c2;
66  disp(X22,'X2(2)=')
67  //For k=3,
68  d2=int(cos((3*%pi)/2))+int((sqrt(-1)*sin((3*%pi)/2))
        );
69  X23=X3*d2;
70  disp(X23,'X2(3)=')
71  disp({,X20,X21,X22,X23,},'So, X2(k)=')
72
73  // (iv) : Now moving ahead to find X3(k)
74  //x3(n)={3,4,1,2} and x(n)={1,2,3,4}
75  //x3(n) is obtained by shifting x(n) by 2 positions
        which means x3(n)=x[n(+/-)2]
76  //According to the circular time shift property : x[
        n(+/-)2] gives DFT as X(k)*e^(-j*2*%pi*k*l/N)
77  a3=cos(0)+(sqrt(-1)*sin(0));
78  //So,for k=0,
79  X30=X0.*real(a3)-X0.*(sqrt(-1)*imag(a3))
80  disp(X30,'X3(0)=')
```

```
81  //So , for  k=1,
82  b3=int(cos(%pi))-(sqrt(-1)*sin(%pi))
83  X31=X1*b3;
84  disp(X31,'X3(1)=')
85  //For  k=2,
86  c3=int(cos(2*%pi))-int((sqrt(-1)*sin(2*%pi)));
87  X32=X2*c3;
88  disp(X32,'X3(2)=')
89  //For  k=3,
90  d3=int(cos(3*%pi))-int((sqrt(-1)*sin(3*%pi)));
91  X33=X3*d3;
92  disp(X33,'X3(3)=')
93  disp({,X30,X31,X32,X33,},'So,  X3(k)=')
94
95  //  (v)  : Now  moving  ahead  to  find  X4(k)
96  //x4(n)={4,6,4,6}  and  x(n)={1,2,3,4}
97  //Both  are  related  as  x4(n)=x(n)+x[n(+/-)2]
98  //Using  half  period  shift  property ,  X4(k)=X(k)+[(-1)
       ^k]*X(k)
99  //For  k=0,
100 X40=X0+[(-1)^0]*X0
101 disp(X40,'X40=')
102 //For  k=1,
103 X41=X1+[(-1)^1]*X1
104 disp(X41,'X41=')
105 //For  k=2,
106 X42=X2+[(-1)^2]*X2
107 disp(X42,'X42=')
108 //For  k=3,
109 X43=X3+[(-1)^3]*X3
110 disp(X43,'X43=')
111 disp({,X40,X41,X42,X43,},'So,  X4(k)=')
```