Scilab Manual for
DIGITAL SIGNAL PROCESSING & PROCESSORS

by Prof Leena Govekar
Electronics Engineering
Pvppcoe/mumbai University[1]

Solutions provided by
Prof Rajiv Suhas Tawde
Others
Mumbai University/pvpp College Of Engineering

May 18, 2024

[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

Experiment: 1

Compute Four Point DFT Using Matrix Approach Only.

Scilab code Solution 1.0 Experiment Number 1

```
// AIM:Compute four point DFT using matrix approach only.
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;

//Let x(n)={1,2,3,4}
//Let us first define the W4 matrix
W4=[1 1 1 1 ;1 -sqrt(-1) -1 sqrt(-1);1 -1 1 -1;1 sqrt(-1) -1 -sqrt(-1)];
disp(W4,'W4=')
//Now let us define the input sequence
xn=[1;2;3;4];//The input sequence x(n)has been arranged as a column matrix
//DFT is obtained by multiplying the twiddle matrix W4 and the input sequence
Xk=W4*xn;
disp(Xk,'DFT : X(k)=')
```

Experiment: 2

Derive The [W4*] Matrix Useful To Compute IDFT

Scilab code Solution 2.0 Experiment Number 2

```
//AIM: Derive the [W4*] matrix useful to compute IDFT.
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
W40_star=int(cos(0))+(sqrt(-1)*int(sin(0)));
W41_star=int(cos(%pi/2))+(sqrt(-1))*int((sin(%pi/2)));
W42_star=int(cos(%pi))+(sqrt(-1))*int((sin(%pi)));
W43_star=int(cos((3*%pi)/2))+(sqrt(-1))*int(sin((3*%pi)/2));

disp(W40_star,'W40_star=')
disp(W41_star,'W41_star=')
disp(W42_star,'W42_star=')
disp(W43_star,'W43_star=')

W44_star=W40_star;
```

```
W49_star=W41_star;
W46_star=W42_star;

W4_star= [W40_star W40_star W40_star W40_star;
          W40_star W41_star W42_star W43_star;
          W40_star W42_star W44_star W46_star;
          W40_star W43_star W46_star W49_star];
disp(W4_star,'W4_star=')
```

Experiment: 3

IDFT Computation Using Matrix Method

Scilab code Solution 3.0 Experiment Number 3

```
//AIM:IDFT computation using matrix method.
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;

//Let Y(k)={1,0,1,0}
//Computation for Inverse Discrete Fourier Transform (IDFT)
//Let us first define the W4* matrix
W4_star=[1 1 1 1 ;1 sqrt(-1) -1 -sqrt(-1);1 -1 1 -1;1 -sqrt(-1) -1 sqrt(-1)];
disp(W4_star,'W4*=')
Yk=[1;0;1;0];//The input sequence Y(k) has been arranged as a column matrix
yn=(1/4)*W4_star*Yk;
disp(yn,'IDFT : y(n)=')
```

Experiment: 4

N=8; DIT-FFT Without Using Inbuilt Scilab FFT Function

Scilab code Solution 4.0 Experiment Number 4

```
//AIM: N=8; DIT-FFT without using inbuilt Scilab FFT
    function
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let x(n)={1,2,1,2,0,2,1,2}
//Let us begin with the programming.For
   understanding,let us write the given data as
//x(0)=1;x(1)=2,x(2)=1,x(3)=2,x(4)=0,x(5)=2,x(6)=1,x
   (7)=2
x0=1;//DIT-FFT,so arranging the input in bit-
   reversed order
x4=0;//DIT-FFT,so arranging the input in bit-
   reversed order
x2=1;//DIT-FFT,so arranging the input in bit-
   reversed order
x6=1;//DIT-FFT,so arranging the input in bit-
   reversed order
```

```
x1=2;//DIT-FFT,so arranging the input in bit-reversed order
x5=2;//DIT-FFT,so arranging the input in bit-reversed order
x3=2;//DIT-FFT,so arranging the input in bit-reversed order
x7=2;//DIT-FFT,so arranging the input in bit-reversed order
//Stage I computation
x0a=x4+x0;//Computing Stage-I output at line 1
disp(x0a,'Stage-I output at line 1=')
x4b=(x4-x0)*(-1);//Computing Stage-I output at line 2
disp(x4b,'Stage-I output at line 2=')
x2c=x6+x2;//Computing Stage-I output at line 3
disp(x2c,'Stage-I output at line 3=')
x6d=(x6-x2)*(-1);//Computing Stage-I output at line 4
disp(x6d,'Stage-I output at line 4=')
x1e=x5+x1;//Computing Stage-I output at line 5
disp(x1e,'Stage-I output at line 5=')
x5f=(x5-x1)*(-1);//Computing Stage-I output at line 6
disp(x5f,'Stage-I output at line 6=')
x3g=x7+x3;//Computing Stage-I output at line 7
disp(x3g,'Stage-I output at line 7')
x7h=(x7-x3)*(-1);//Computing Stage-I output at line 8
disp(x7h,'Stage-I output at line 8=')

//Stage-I output at line 4 and line 8 is to be multiplied by twiddle factor having value (-j)
x6d1=(x6d)*(-sqrt(-1));
x7h1=(x7h)*(-sqrt(-1));
disp(x6d1,'Stage-I output(i.e. input to stage-II) after multiplication by twiddle factor value of (-j) at line 4 =')
disp(x7h1,'Stage-I output(i.e. input to stage-II)
```

```
after multiplication by twiddle factor value of (-j) at line 8 =')

//Stage-II Computations
x0a_stageII_output=x2c+x0a;//Computing Stage-II output at line 1
disp(x0a_stageII_output,'Stage-II output at line 1=')
x4b_stageII_output=x6d1+x4b;//Computing Stage-II output at line 2
disp(x4b_stageII_output,'Stage-II output at line 2=')
x2c_stageII_output=(x2c-x0a)*(-1);//Computing Stage-II output at line 3
disp(x2c_stageII_output,'Stage-II output at line 3=')
x6d_stageII_output=(x6d1-x4b)*(-1);//Computing Stage-II output at line 4
disp(x6d_stageII_output,'Stage-II output at line 4=')
x1e_stageII_output=x3g+x1e;//Computing Stage-II output at line 5
disp(x1e_stageII_output,'Stage-II output at line 5=')
x5f_stageII_output=x7h1+x5f;//Computing Stage-II output at line 6
disp(x5f_stageII_output,'Stage-II output at line 6=')
x3g_stageII_output=(x3g-x1e)*(-1);//Computing Stage-II output at line 7
disp(x3g_stageII_output,'Stage-II output at line 7=')
x7h_stageII_output=(x7h1-x5f)*(-1);//Computing Stage-II output at line 8
disp(x7h_stageII_output,'Stage-II output at line 8=')

//Stage-II output at line 6,line 7 and line 8 are to
```

```
be multiplied by twiddle factor having value (0.707-j0.707),(-j) and (-0.707-j0.707) respectively
x5f_stgII_op_multi_by_tw=(x5f_stageII_output)*(0.707-(sqrt(-1))*(0.707));
disp(x5f_stgII_op_multi_by_tw,'Stage-II output at line 6 after multiplication by twiddle factor=')
x3g_stgII_op_multi_by_tw=(x3g_stageII_output)*(-(sqrt(-1)));
disp(x3g_stgII_op_multi_by_tw,'Stage-II output at line 7 after multiplication by twiddle factor=')
x7h_stgII_op_multi_by_tw=(x7h_stageII_output)*(-0.707-(sqrt(-1))*(0.707));
disp(x7h_stgII_op_multi_by_tw,'Stage-II output at line 8 after multiplication by twiddle factor=')

//Stage-III Computations(i.e. Computations for the final stage)
X0=x1e_stageII_output+x0a_stageII_output;//Computing X(0) at last stage
X1=x5f_stgII_op_multi_by_tw+x4b_stageII_output;//Computing X(1) at last stage
X2=x3g_stgII_op_multi_by_tw+x2c_stageII_output;//Computing X(2) at last stage
X3=x7h_stgII_op_multi_by_tw+x6d_stageII_output;//Computing X(3) at last stage
X4=(x1e_stageII_output-x0a_stageII_output)*(-1);//Computing X(4) at last stage
X5=(x5f_stgII_op_multi_by_tw-x4b_stageII_output)*(-1);//Computing X(5) at last stage
X6=(x3g_stgII_op_multi_by_tw-x2c_stageII_output)*(-1);//Computing X(6) at last stage
X7=(x7h_stgII_op_multi_by_tw-x6d_stageII_output)*(-1);//Computing X(7) at last stage
disp(X0,'X(0)=')
disp(X1,'X(1)=')
disp(X2,'X(2)=')
disp(X3,'X(3)=')
```

```
disp(X4,'X(4)=')
disp(X5,'X(5)=')
disp(X6,'X(6)=')
disp(X7,'X(7)=')
disp({,X0,X1,X2,X3,X4,X5,X6,X7,},'So,the DFT of x(n) using Decimation-in-Time Fast Fourier Transform(DIT-FFT) is X(k)=')
```

Experiment: 5

N=8; IDIT-FFT Without Using Inbuilt Scilab FFT Function

Scilab code Solution 5.0 Experiment Number 5

```
//AIM:N=8; IDIT-FFT without using inbuilt Scilab FFT
    function
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let X(k)={11,1-1,1,-5,1,-1,1}
//Let us begin with the programming.For
   understanding,let us write the given data as
//X(0)=11;X(1)=1,X(2)=-1,X(3)=1,X(4)=-5,X(5)=1,X(6)
   =-1,X(7)=1
X0_conj=11;//IDIT-FFT,so arranging the input in bit-
   reversed order
X4_conj=-5;//IDIT-FFT,so arranging the input in bit-
   reversed order
X2_conj=-1;//IDIT-FFT,so arranging the input in bit-
   reversed order
X6_conj=-1;//IDIT-FFT,so arranging the input in bit-
   reversed order
```

```
X1_conj=1;//IDIT-FFT,so arranging the input in bit-reversed order
X5_conj=1;//IDIT-FFT,so arranging the input in bit-reversed order
X3_conj=1;//IDIT-FFT,so arranging the input in bit-reversed order
X7_conj=1;//IDIT-FFT,so arranging the input in bit-reversed order
disp(X0_conj,'X*(0)=')
disp(X4_conj,'X*(4)=')
disp(X2_conj,'X*(2)=')
disp(X6_conj,'X*(6)=')
disp(X1_conj,'X*(1)=')
disp(X5_conj,'X*(5)=')
disp(X3_conj,'X*(3)=')
disp(X7_conj,'X*(7)=')
//Stage I computation
X0a=X4_conj+X0_conj;//Computing Stage-I output at line 1
disp(X0a,'Stage-I output at line 1=')
X4b=(X4_conj-X0_conj)*(-1);//Computing Stage-I output at line 2
disp(X4b,'Stage-I output at line 2=')
X2c=X6_conj+X2_conj;//Computing Stage-I output at line 3
disp(X2c,'Stage-I output at line 3=')
X6d=(X6_conj-X2_conj)*(-1);//Computing Stage-I output at line 4
disp(X6d,'Stage-I output at line 4=')
X1e=X5_conj+X1_conj;//Computing Stage-I output at line 5
disp(X1e,'Stage-I output at line 5=')
X5f=(X5_conj-X1_conj)*(-1);//Computing Stage-I output at line 6
disp(X5f,'Stage-I output at line 6=')
X3g=X7_conj+X3_conj;//Computing Stage-I output at line 7
disp(X3g,'Stage-I output at line 7')
```

```
X7h=(X7_conj-X3_conj)*(-1);//Computing Stage-I output at line 8
disp(X7h,'Stage-I output at line 8=')

//Stage-I output at line 4 and line 8 is to be multiplied by twiddle factor having value (-j)
X6d'=(X6d)*(-sqrt(-1));
X7h'=(X7h)*(-sqrt(-1));
disp(X6d','Stage-I output(i.e. input to stage-II) after multiplication by twiddle factor value of (-j) at line 4 =')
disp(X7h','Stage-I output(i.e. input to stage-II) after multiplication by twiddle factor value of (-j) at line 8 =')

//Stage-II Computations
X0a_stageII_output=X2c+X0a;//Computing Stage-II output at line 1
disp(X0a_stageII_output,'Stage-II output at line 1=')
X4b_stageII_output=X6d'+X4b;//Computing Stage-II output at line 2
disp(X4b_stageII_output,'Stage-II output at line 2=')
X2c_stageII_output=(X2c-X0a)*(-1);//Computing Stage-II output at line 3
disp(X2c_stageII_output,'Stage-II output at line 3=')
X6d_stageII_output=(X6d'-X4b)*(-1);//Computing Stage-II output at line 4
disp(X6d_stageII_output,'Stage-II output at line 4=')
X1e_stageII_output=X3g+X1e;//Computing Stage-II output at line 5
disp(X1e_stageII_output,'Stage-II output at line 5=')
X5f_stageII_output=X7h'+X5f;//Computing Stage-II output at line 6
```

```
disp(X5f_stageII_output,'Stage-II output at line 6=')
X3g_stageII_output=(X3g-X1e)*(-1);//Computing Stage-II output at line 7
disp(X3g_stageII_output,'Stage-II output at line 7=')
X7h_stageII_output=(X7h'-X5f)*(-1);//Computing Stage-II output at line 8
disp(X7h_stageII_output,'Stage-II output at line 8=')

//Stage-II output at line 6,line 7 and line 8 are to be multiplied by twiddle factor having value (0.707-j0.707),(-j) and (-0.707-j0.707) respectively
X5f_stageII_output_multiplied_by_twiddle=(X5f_stageII_output)*(0.707-(sqrt(-1))*(0.707));
disp(X5f_stageII_output_multiplied_by_twiddle,'Stage-II output at line 6 after multiplication by twiddle factor=')
X3g_stageII_output_multiplied_by_twiddle=(X3g_stageII_output)*(-(sqrt(-1)));
disp(X3g_stageII_output_multiplied_by_twiddle,'Stage-II output at line 7 after multiplication by twiddle factor=')
X7h_stageII_output_multiplied_by_twiddle=(X7h_stageII_output)*(-0.707-(sqrt(-1))*(0.707));
disp(X7h_stageII_output_multiplied_by_twiddle,'Stage-II output at line 8 after multiplication by twiddle factor=')

//Stage-III Computations(i.e. Computations for the final stage)
x0_star=(1/8)*(X1e_stageII_output+X0a_stageII_output);//Computing x*(0) at last stage
x1_star=(1/8)*(X5f_stageII_output_multiplied_by_twiddle+X4b_stageII_output);//Computing x*(1) at last
```

```
stage
x2_star=(1/8)*(X3g_stageII_output_multiplied_by_twiddle+X2c_stageII_output);//Computing x*(2) at last stage
x3_star=(1/8)*(X7h_stageII_output_multiplied_by_twiddle+X6d_stageII_output);//Computing x*(3) at last stage
x4_star=(1/8)*((X1e_stageII_output-X0a_stageII_output)*(-1));//Computing x*(4) at last stage
x5_star=(1/8)*((X5f_stageII_output_multiplied_by_twiddle-X4b_stageII_output)*(-1));//Computing x*(5) at last stage
x6_star=(1/8)*((X3g_stageII_output_multiplied_by_twiddle-X2c_stageII_output)*(-1));//Computing x*(6) at last stage
x7_star=(1/8)*((X7h_stageII_output_multiplied_by_twiddle-X6d_stageII_output)*(-1));//Computing x*(7) at last stage
disp(x0_star,'x*(0)=')
disp(x1_star,'x*(1)=')
disp(x2_star,'x*(2)=')
disp(x3_star,'x*(3)=')
disp(x4_star,'x*(4)=')
disp(x5_star,'x*(5)=')
disp(x6_star,'x*(6)=')
disp(x7_star,'x*(7)=')
disp({,x0_star,x1_star,x2_star,x3_star,x4_star,x5_star,x6_star,x7_star,},'x*(n)=')
x0_star_real=real(x0_star);
x0_star_imag_conj=(-1)*(imag(x0_star));
x1_star_real=real(x1_star);
x1_star_imag_conj=(-1)*(imag(x1_star));
```

```
x2_star_real=real(x2_star);
x2_star_imag_conj=(-1)*(imag(x2_star));
x3_star_real=real(x3_star);
x3_star_imag_conj=(-1)*(imag(x3_star));
x4_star_real=real(x4_star);
x4_star_imag_conj=(-1)*(imag(x4_star));
x5_star_real=real(x5_star);
x5_star_imag_conj=(-1)*(imag(x5_star));
x6_star_real=real(x6_star);
x6_star_imag_conj=(-1)*(imag(x6_star));
x7_star_real=real(x7_star);
x7_star_imag_conj=(-1)*(imag(x7_star));
x0=x0_star_real+x0_star_imag_conj;
x1=x1_star_real+x1_star_imag_conj;
x2=x2_star_real+x2_star_imag_conj;
x3=x3_star_real+x3_star_imag_conj;
x4=x4_star_real+x4_star_imag_conj;
x5=x5_star_real+x5_star_imag_conj;
x6=x6_star_real+x6_star_imag_conj;
x7=x7_star_real+x7_star_imag_conj;
disp({,x0,x1,x2,x3,x4,x5,x6,x7,},'So,the IDFT of X(k
    ) using Inverse Decimation-in-Time Fast Fourier
    Transform(IDIT-FFT) is x(n)=')
```

Experiment: 6

N=8; DIF-FFT Without Using Inbuilt Scilab FFT Function

Scilab code Solution 6.0 Experiment Number 6

```
//AIM :N=8; DIF-FFT without using inbuilt Scilab FFT
    function
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let x(n)={1,2,1,2,0,2,1,2}
//Let us begin with the programming.For
   understanding,let us write the given data as
//x(0)=1;x(1)=2,x(2)=1,x(3)=2,x(4)=0,x(5)=2,x(6)=1,x
   (7)=2
x0=1;//DIF-FFT,so arranging the input in natural
   order
x1=2;//DIF-FFT,so arranging the input in natural
   order
x2=1;//DIF-FFT,so arranging the input in natural
   order
x3=2;//DIF-FFT,so arranging the input in natural
   order
```

```
x4=0;//DIF−FFT,so arranging the input in natural order
x5=2;//DIF−FFT,so arranging the input in natural order
x6=1;//DIF−FFT,so arranging the input in natural order
x7=2;//DIF−FFT,so arranging the input in natural order
//Stage I computation
x0a=x4+x0;//Computing Stage−I output at line 1
disp(x0a,'Stage−I output at line 1=')
x1b=x5+x1;//Computing Stage−I output at line 2
disp(x1b,'Stage−I output at line 2=')
x2c=x6+x2;//Computing Stage−I output at line 3
disp(x2c,'Stage−I output at line 3=')
x3d=x7+x3;//Computing Stage−I output at line 4
disp(x3d,'Stage−I output at line 4=')
x4e=(x4-x0)*(-1);//Computing Stage−I output at line 5
disp(x4e,'Stage−I output at line 5=')
x5f=(x5-x1)*(-1);//Computing Stage−I output at line 6
disp(x5f,'Stage−I output at line 6=')
x6g=(x6-x2)*(-1);//Computing Stage−I output at line 7
disp(x6g,'Stage−I output at line 7')
x7h=(x7-x3)*(-1);//Computing Stage−I output at line 8
disp(x7h,'Stage−I output at line 8=')

//Stage−I output at line 6,line 7 and line 8 are to be multiplied by twiddle factor having value (0.707−j0.707),(−j) and (−0.707−j0.707) respectively
x5f_stageI_output_multiplied_by_twiddle=(x5f)*(0.707-(sqrt(-1))*(0.707));
disp(x5f_stageI_output_multiplied_by_twiddle,'Stage−I output at line 6 after multiplication by
```

```
twiddle factor=')
x6g_stageI_output_multiplied_by_twiddle=(x6g)*(-(sqrt(-1)));
disp(x6g_stageI_output_multiplied_by_twiddle,'Stage-I output at line 7 after multiplication by twiddle factor=')
x7h_stageI_output_multiplied_by_twiddle=(x7h)*(-0.707-(sqrt(-1))*(0.707));
disp(x7h_stageI_output_multiplied_by_twiddle,'Stage-I output at line 8 after multiplication by twiddle factor=')

//Stage-II Computations
x0a_stageII_output=x2c+x0a;//Computing Stage-II output at line 1
disp(x0a_stageII_output,'Stage-II output at line 1=')
x1b_stageII_output=x3d+x1b;//Computing Stage-II output at line 2
disp(x1b_stageII_output,'Stage-II output at line 2=')
x2c_stageII_output=(x2c-x0a)*(-1);//Computing Stage-II output at line 3
disp(x2c_stageII_output,'Stage-II output at line 3=')
x3d_stageII_output=(x3d-x1b)*(-1);//Computing Stage-II output at line 4
disp(x3d_stageII_output,'Stage-II output at line 4=')
x4e_stageII_output=x6g+x4e;//Computing Stage-II output at line 5
disp(x4e_stageII_output,'Stage-II output at line 5=')
x5f_stageII_output=x7h_stageI_output_multiplied_by_twiddle+x5f_stageI_output_multiplied_by_twiddle;//Computing Stage-II output at line 6
disp(x5f_stageII_output,'Stage-II output at line 6='
```

```
)
x6g_stageII_output=(x6g_stageI_output_multiplied_by_twiddle-x4e)*(-1);//Computing Stage-II output at line 7
disp(x6g_stageII_output,'Stage-II output at line 7=')
x7h_stageII_output=(x7h'-x5f)*(-1);//Computing Stage-II output at line 8
disp(x7h_stageII_output,'Stage-II output at line 8=')

//Stage-II output at line 4 and line 8 are to be multiplied by twiddle factor having value (-j)
x3d_stageII_output_multiplied_by_twiddle=(x3d_stageII_output)*(-(sqrt(-1)));
disp(x3d_stageII_output_multiplied_by_twiddle,'Stage-II output at line 4 after multiplication by twiddle factor=')
x7h_stageII_output_multiplied_by_twiddle=(x7h_stageII_output)*(-(sqrt(-1)));
disp(x7h_stageII_output_multiplied_by_twiddle,'Stage-II output at line 8 after multiplication by twiddle factor=')

//Stage-III Computations(i.e. Computations for the final stage)
X0=x1b_stageII_output+x0a_stageII_output;//Computing X(0) at last stage
X4=(x1b_stageII_output-x0a_stageII_output)*(-1);//Computing X(4) at last stage
X2=x3d_stageII_output_multiplied_by_twiddle+x2c_stageII_output;//Computing X(2) at last stage
X6=(x3d_stageII_output_multiplied_by_twiddle-x2c_stageII_output)*(-1);//Computing X(6) at last stage
X1=(x5f_stageII_output+x4e_stageII_output);//Computing X(1) at last stage
X5=(x5f_stageII_output-x4e_stageII_output)*(-1);//
```

```
Computing X(5) at last stage
X3=x7h_stageII_output_multiplied_by_twiddle+x6g_stageII_output;//Computing X(3) at last stage
X7=(x7h_stageII_output_multiplid_by_twiddle-x6g_stageII_output)*(-1);//Computing X(7) at last stage
disp(X0,'X(0)=')
disp(X4,'X(4)=')
disp(X2,'X(2)=')
disp(X6,'X(6)=')
disp(X1,'X(1)=')
disp(X5,'X(5)=')
disp(X3,'X(3)=')
disp(X7,'X(7)=')
disp({,X0,X1,X2,X3,X4,X5,X6,X7,},'So,the DFT of x(n) using Decimation-in-Frequency Fast Fourier Transform(DIF-FFT) is X(k)=')
```

Experiment: 7

N=8;IDIF-FFT Without Using Inbuilt Scilab FFT Function

Scilab code Solution 7.0 Experiment Number 7

```
//AIM:N=8;IDIF-FFT without using inbuilt Scilab FFT
    function
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let X(k)={11,1,-1,1,-5,1,-1,1}
//Let us begin with the programming.For
    understanding,let us write the given data as
//X(0)=11;X(1)=1,X(2)=-1,X(3)=1,X(4)=-5,X(5)=1,X(6)
    =-1,X(7)=1
X0_conj=11;//IDIF-FFT,so arranging the input in
    natural order
X1_conj=1;//IDIF-FFT,so arranging the input in
    natural order
X2_conj=-1;//IDIF-FFT,so arranging the input in
    natural order
X3_conj=1;//IDIF-FFT,so arranging the input in
    natural order
```

```
X4_conj=-5;//IDIF-FFT,so arranging the input in natural order
X5_conj=1;//IDIF-FFT,so arranging the input in natural order
X6_conj=-1;//IDIF-FFT,so arranging the input in natural order
X7_conj=1;//IDIF-FFT,so arranging the input in natural order
disp(X0_conj,'X*(0)=')
disp(X1_conj,'X*(1)=')
disp(X2_conj,'X*(2)=')
disp(X3_conj,'X*(3)=')
disp(X4_conj,'X*(4)=')
disp(X5_conj,'X*(5)=')
disp(X6_conj,'X*(6)=')
disp(X7_conj,'X*(7)=')

// Twiddle factor
W0=cos(((2*%pi)/8)*0)-(sqrt(-1))*sin(((2*%pi)/8)*0)
W1=cos(((2*%pi)/8)*1)-sqrt(-1)*sin(((2*%pi)/8)*1)
W2=cos(((2*%pi)/8)*2)-sqrt(-1)*sin(((2*%pi)/8)*2)
W3=cos(((2*%pi)/8)*3)-sqrt(-1)*sin(((2*%pi)/8)*3)

//Stage I computation
x0a=X0_conj+X4_conj
x1b=X1_conj+X5_conj
x2c=X2_conj+X6_conj
x3d=X3_conj+X7_conj
x4e=X0_conj+(-1)*X4_conj
x5f=X1_conj+(-1)*X5_conj
x6g=X2_conj+(-1)*X6_conj
x7h=X3_conj+(-1)*X7_conj
disp('Stage-I values are')
disp(x0a)
disp(x1b)
disp(x2c)
disp(x3d)
disp(x4e)
```

```
disp(x5f)
disp(x6g)
disp(x7h)
//Stage I output at line 4,5,6 & 7 are to be multiplied by factors W0, W1, W2, and W3 respectively
x4e1=x4e*W0
x5f1=x5f*W1
x6g1=x6g*W2
x7h1=x7h*W3

//Stage II computation
x0a_stageII_output=x2c+x0a;//Computing Stage-II output at line 1
disp(x0a_stageII_output,'Stage-II output at line 1=')
x1b_stageII_output=x3d+x1b;//Computing Stage-I output at line 2
disp(x1b_stageII_output,'Stage-II output at line 2=')
x2c_stageII_output=x0a-x2c;//Computing Stage-I output at line 3
disp(x2c_stageII_output,'Stage-II output at line 3=')
x3d_stageII_output=x1b-x3d;//Computing Stage-I output at line 4
disp(x3d_stageII_output,'Stage-II output at line 4=')
x4e_stageII_output=x6g+x4e1;//Computing Stage-II output at line 5
disp(x4e_stageII_output,'Stage-II output at line 5=')
x5f_stageII_output=x7h+x5f1//Computing Stage-II output at line 6
disp(x5f_stageII_output,'Stage-II output at line 6=')
x6g_stageII_output=x4e1+(x6g1*(-1));//Computing Stage-II output at line 7
```

```
disp(x6g_stageII_output,'Stage-II output at line 7=')
x7h_stageII_output=x5f1+(x7h1*(-1));//Computing Stage-II output at line 8
disp(x7h_stageII_output,'Stage-II output at line 8=')

//Stage III computation
x0=x0a_stageII_output+x1b_stageII_output;//
x4=x0a_stageII_output+(-1)*x1b_stageII_output
x2=x3d_stageII_output*((-1)*sqrt(-1))+x2c_stageII_output;//at line 2 x3d_stageII_output is to be multiplied by factor -j
x6=x3d_stageII_output*((-1)*sqrt(-1))*(-1)+x2c_stageII_output;//at line 3 x3d_stageII_output is to be multioplied by factor -j*(-1)
x1=x4e_stageII_output+x5f_stageII_output
x5=x4e_stageII_output-x5f_stageII_output
x3=x7h_stageII_output*((-1)*sqrt(-1))+x6g_stageII_output;//at line 7 x7h_stageII_output is to b e multiplied by factor -j
x7=x7h_stageII_output*((-1)*sqrt(-1))*(-1)+x6g_stageII_output;// at line 8 x7h_stageII_output is to be multiplied by factor (-j)*(-1)
//final computation
x0_star=(1/8)*(x0)
disp(x0_star,'x*(0)=')
x4_star=(1/8)*(x4)
disp(x4_star,'x*(4)=')
x2_star=(1/8)*(x2)
disp(x2_star,'x*(2)=')
x6_star=(1/8)*(x6)
disp(x6_star,'x*(6)=')
x1_star=(1/8)*(x1)
disp(x1_star,'x*(1)=')
x5_star=(1/8)*(x5)
disp(x5_star,'x*(5)=')
```

```
x3_star=(1/8)*(x3)
disp(x3_star,'x*(3)=')
x7_star=(1/8)*(x7)
disp(x7_star,'x*(7)=')

disp({,x0_star,x1_star,x2_star,x3_star,x4_star,
   x5_star,x6_star,x7_star,},'x*(n)=')
x0_star_real=real(x0_star);
x0_star_imag_conj=(-1)*(imag(x0_star));
x1_star_real=real(x1_star);
x1_star_imag_conj=(-1)*(imag(x1_star));
x2_star_real=real(x2_star);
x2_star_imag_conj=(-1)*(imag(x2_star));
x3_star_real=real(x3_star);
x3_star_imag_conj=(-1)*(imag(x3_star));
x4_star_real=real(x4_star);
x4_star_imag_conj=(-1)*(imag(x4_star));
x5_star_real=real(x5_star);
x5_star_imag_conj=(-1)*(imag(x5_star));
x6_star_real=real(x6_star);
x6_star_imag_conj=(-1)*(imag(x6_star));
x7_star_real=real(x7_star);
x7_star_imag_conj=(-1)*(imag(x7_star));
x0=x0_star_real+x0_star_imag_conj;
x1=x1_star_real+x1_star_imag_conj;
x2=x2_star_real+x2_star_imag_conj;
x3=x3_star_real+x3_star_imag_conj;
x4=x4_star_real+x4_star_imag_conj;
x5=x5_star_real+x5_star_imag_conj;
x6=x6_star_real+x6_star_imag_conj;
x7=x7_star_real+x7_star_imag_conj;
disp({,x0,x1,x2,x3,x4,x5,x6,x7,},'So,the IDFT of X(k
   ) using Inverse Decimation-in-Frequency Fast
   Fourier Transform(IDIF-FFT) is x(n)=')
```

Experiment: 8

Compute Kaiser Window Parameter Beta & Its Minimum Length

Scilab code Solution 8.0 Experiment Number 8

```
//AIM:Compute Kaiser window parameter Beta & its minimum length
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let us consider the following specifications :
// H ( ) 0 .01 0 0 .25
//0.95 H ( ) 1 .05 0.35 0 .6
// H ( ) 0 .01 0.65
//The magnitude specifications of the FIR filter is given by
// 1- pH ( ) 1+ p for 0 p
// 0 H ( ) s for s
//On comparing,we get 1- p =0.95
del_p=0.05;
```

```
del_s=0.01;
omega_p=0.6*(%pi);
omega_s=0.65*(%pi);
del_omega=omega_s-omega_p;
//   ia minimum of  p  and minimum of  s
del=0.05;
//Attenuation A is given as
A=((-20)*(log10(del)));
disp("dB",A,"Attenuation(A)=")
//Calculating value of
beeta=(A-21)^(0.4)+0.07886+(A-21);
disp(beeta,"  =")
//The length of filter is (M+1)
//The value of M is calculated as follows
M=((A-8)/(2.285*(del_omega)));
disp(M,"M=")
```

Experiment: 9

Design High Pass Butterworth Filter Using Bilinear Transformation

Scilab code Solution 9.0 Experiment Number 9

```
//AIM:Design High pass Butterworth filter using
   Bilinear Transformation.
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
s=poly(0,"s")
T=1;//Assume T=1 second
Ap=0.8;//Attenuation in pass band
As=0.2;//Attenuation in stop band
wp=0.2*(%pi)
ws=0.6*(%pi)
ohmp=2/T*(tan(wp/2))
ohms=2/T*(tan(ws/2))
//ORDER CALCULATION(N);
a=(1/As^2-1)
b=(1/Ap^2-1)
```

```
c=log(a/b)
N=(1/2)*(c/(log(ohms/ohmp)))
Nr=int (N)
x=N-int(N)
if(x>0)
 Nr=Nr+1
ohmc=(ohmp/(1/Ap^2-1)^(1/(2*Nr)))
//calculation of poles
i=0:1:Nr-1;
pi_plus=ohmc*exp(%i*(Nr+2*i+1)*(%pi)/(2*Nr))
pi_minus=-ohmc*exp(%i*(2+2.*i+1)*(%pi)/(2*Nr))
disp(wp,' p =')
disp(ws,' s =')
disp(ohmp,'  p =')
disp(ohms,'  s =')
disp(N','Order (N)=')
disp(Nr,'Integer value of the order:(Nr)=')
disp(ohmc,'  c =')
disp(pi_plus,'Poles=')
disp(pi_minus,'Poles=')
h=ohmc/(s-(-0.53-0.53*%i))
h1=ohmc/(s-(-0.53+0.53*%i))
h2=h*h1;
disp(h,h1,'The analog transfer function will be the multiplication of the following two terms:');
disp(h2,'The analog transfer function H(s)=')
Z=poly(0,"Z")
s=(ohmc*ohmp)/((2/T)*((Z-1)/(Z+1)));
h3=0.56/(s^2+1.06*s+0.56);
disp(h3,"Transfer function of digital filter H(Z)=")
```

Experiment: 10

Overlap Add Method To Filter Long Sequences Using Linear Convolution

Scilab code Solution 10.0 Experiment Number 10

```
//AIM:Overlap add method to filter long sequences
   using linear convolution
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
//Let x(n)={1,2,3,4,5,6,7,8} and h(n)={1,2}
xn =[1 2 3 4 5 6 7 8]; //Nx=8
xon =[1 2];
xono=1;
xon1=2;
x1n =[3 4];
x1no=3;
x1n1=4;
x2n =[5 6];
x2no=5;
x2n1=6;
```

```
x3n =[7 8];
x3no=7;
x3n1=8;
hn =[1 2]; //Here length of impulse response array h(n) is 2 (i.e. M=2) or Nh=2
//Length of each partitioned input i.e x0n to x3n is 2(i.e. L=2)
//Since Nx=8,Nh=2 and we know Nx=m*Nh(so 8=m*2) giving m=4;and so x(n) has been partitioned into 4 blocks of length Nh=2
hno=1;
hn1=2;

a=xono*hno;
b=xon1*hno;
c=xono*hn1;
d=xon1*hn1;
yon=[a c+b d];
disp(yon,'yon=')

e=x1no*hno;
f=x1n1*hno;
g=x1no*hn1;
h=x1n1*hn1;
y1n=[e g+f h];
disp(y1n,'y1n=')

i=x2no*hno;
j=x2n1*hno;
k=x2no*hn1;
l=x2n1*hn1;
y2n=[i k+j l];
disp(y2n,'y2n=')

m=x3no*hno;
n=x3n1*hno;
o=x3no*hn1;
p=x3n1*hn1;
```

```
y3n=[m o+n p];
disp(y3n,'y3n=')

yon =[yon,0,0,0,0,0,0]
y1n =[0,0,y1n,0,0,0,0]
y2n =[0,0,0,0,y2n,0,0]
y3n =[0,0,0,0,0,0,y3n ]

yn=yon+ y1n+y2n+y3n
disp(yn,'So,the output using overlap add method(
    without using inbuilt functions) is y(n)= ')
```

Experiment: 11

Overlap Save Method For Sectioned Convolution Using Matrix Approach

Scilab code Solution 11.0 Experiment Number 11

```
//AIM: Overlap save method for sectioned convolution using matrix approach.
//Software version Scilab 5.5.2
//OS windows 10
clc;
clear;
xn =[1 2 -1 2 3 -2 -3 -1 1 1 2 -1]; //Nx=12
hn =[1 2 3];    //Nh=3
//L(approx.=)2*Nh,so L(approx.=)2*3
//So(approx=)6
//We consider the length as 5
//Nh-1=3-1=2
//So Nh-1 number of leading zeros to be added to x(n)
//So xn=[0 0 1 2 -1 2 3 -2 -3 -1 1 1 2 -1]
x0n =[0 0 1 2 -1]; //Partitioned input sequence
x1n =[2 -1 2 3 -2];//Partitioned input sequence
```

```
x2n =[3 -2 -3 -1 1];//Partitioned input sequence
x3n=[-1 1 1 2 -1];//Partitioned input sequence
x4n=[2 -1 0 0 0 ];//Partitioned input sequence
//Convolving each partitioned input sequence with hn
y0n =[0 -1 2 1 0; 0 0 -1 2 1; 1 0 0 -1 2; 2 1 0 0 -1; -1 2 1 0 0]*[1;2;3;0;0];
disp(y0n,"y0n=")
y1n =[2 -2 3 2 -1; -1 2 -2 3 2; 2 -1 2 -2 3; 3 2 -1 2 -2; -2 3 2 -1 2]*[1;2;3;0;0];
disp(y1n,"y1n=")
y2n =[3 1 -1 -3 -2; -2 3 1 -1 -3; -3 -2 3 1 -1;-1 -3 -2 3 1; 1 -1 -3 -2 3]*[1;2;3;0;0];
disp(y2n,"y2n=")
y3n=[-1 -1 2 1 1;1 -1 -1 2 1; 1 1 -1 -1 2;2 1 1 -1 -1; -1 2 1 1 -1]*[1;2;3;0;0];
disp(y3n,"y3n=")
y4n=[2 0 0 0 -1; -1 2 0 0 0; 0 -1 2 0 0; 0 0 -1 2 0; 0 0 0 -1 2]*[1;2;3;0;0];
disp(y4n,"y4n=")
yn0 = y0n (3:5)
//(3:5) means that from yon,select the element from 3rd to 5th
yn1 = y1n (3:5)
yn2 = y2n (3:5)
yn3 = y3n (3:5)
yn4 = y4n (3:5)
yn =[yn0;yn1;yn2;yn3;yn4] //Concatenating yno,yn1,yn2,yn3 and yn4
disp(yn,"y(n)=")
```