

Scilab Manual for  
DIGITAL SIGNAL PROCESSING &  
PROCESSORS  
by Prof Leena Govekar  
Electronics Engineering  
Pvppcoe/mumbai University<sup>1</sup>

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

May 11, 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	4
1 Compute Four Point DFT Using Matrix Approach Only.	5
2 Derive The [W4*] Matrix Useful To Compute IDFT	7
3 IDFT Computation Using Matrix Method	9
4 N=8; DIT-FFT Without Using Inbuilt Scilab FFT Function	10
5 N=8; IDIT-FFT Without Using Inbuilt Scilab FFT Function	15
6 N=8; DIF-FFT Without Using Inbuilt Scilab FFT Function	21
7 N=8;IDIF-FFT Without Using Inbuilt Scilab FFT Function	26
8 Compute Kaiser Window Parameter Beta & Its Minimum Length	31
9 Design High Pass Butterworth Filter Using Bilinear Transformation	33
10 Overlap Add Method To Filter Long Sequences Using Linear Convolution	35

**11 Overlap Save Method For Sectioned Convolution Using Matrix Approach**

**38**

# List of Experiments

Solution 1.0	Experiment Number 1 . . . . .	5
Solution 2.0	Experiment Number 2 . . . . .	7
Solution 3.0	Experiment Number 3 . . . . .	9
Solution 4.0	Experiment Number 4 . . . . .	10
Solution 5.0	Experiment Number 5 . . . . .	15
Solution 6.0	Experiment Number 6 . . . . .	21
Solution 7.0	Experiment Number 7 . . . . .	26
Solution 8.0	Experiment Number 8 . . . . .	31
Solution 9.0	Experiment Number 9 . . . . .	33
Solution 10.0	Experiment Number 10 . . . . .	35
Solution 11.0	Experiment Number 11 . . . . .	38

# Experiment: 1

## Compute Four Point DFT Using Matrix Approach Only.

Scilab code Solution 1.0 Experiment Number 1

```
1 // AIM:Compute four point DFT using matrix approach
  only.
2 //Software version Scilab 5.5.2
3 //OS windows 10
4 clc;
5 clear;
6
7 //Let  $x(n)=\{1,2,3,4\}$ 
8 //Let us first define the W4 matrix
9 W4=[1 1 1 1 ;1 -sqrt(-1) -1 sqrt(-1);1 -1 1 -1;1
    sqrt(-1) -1 -sqrt(-1)];
10 disp(W4, 'W4=')
11 //Now let us define the input sequence
12 xn=[1;2;3;4]; //The input sequence x(n) has been
    arranged as a column matrix
13 //DFT is obtained by multiplying the twiddle matrix
    W4 and the input sequence
14 Xk=W4*xn;
15 disp(Xk, 'DFT : X(k)=')
```



## Experiment: 2

# Derive The [W4\*] Matrix Useful To Compute IDFT

Scilab code Solution 2.0 Experiment Number 2

```
1 //AIM: Derive the [W4*] matrix useful to compute
  IDFT.
2 //Software version Scilab 5.5.2
3 //OS windows 10
4 clc;
5 clear;
6 W40_star=int(cos(0))+(sqrt(-1)*int(sin(0)));
7 W41_star=int(cos(%pi/2))+(sqrt(-1))*int((sin(%pi/2))
  );
8 W42_star=int(cos(%pi))+(sqrt(-1))*int((sin(%pi)));
9 W43_star=int(cos((3*%pi)/2))+(sqrt(-1))*int(sin((3*
  %pi)/2));
10
11 disp(W40_star, 'W40_star=')
12 disp(W41_star, 'W41_star=')
13 disp(W42_star, 'W42_star=')
14 disp(W43_star, 'W43_star=')
15
16 W44_star=W40_star;
```

```
17 W49_star=W41_star;
18 W46_star=W42_star;
19
20 W4_star= [W40_star W40_star W40_star W40_star;
21           W40_star W41_star W42_star W43_star;
22           W40_star W42_star W44_star W46_star;
23           W40_star W43_star W46_star W49_star];
24 disp(W4_star, 'W4_star=')
```

---

## Experiment: 3

# IDFT Computation Using Matrix Method

Scilab code Solution 3.0 Experiment Number 3

```
1 //AIM:IDFT computation using matrix method.
2 //Software version Scilab 5.5.2
3 //OS windows 10
4 clc;
5 clear;
6
7 //Let Y(k)={1,0,1,0}
8 //Computation for Inverse Discrete Fourier Transform
  (IDFT)
9 //Let us first define the W4* matrix
10 W4_star=[1 1 1 1 ;1 sqrt(-1) -1 -sqrt(-1);1 -1 1
  -1;1 -sqrt(-1) -1 sqrt(-1)];
11 disp(W4_star, 'W4*=' )
12 Yk=[1;0;1;0]; //The input sequence Y(k) has been
  arranged as a column matrix
13 yn=(1/4)*W4_star*Yk;
14 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

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

```

13 x1=2; //DIT-FFT, so arranging the input in bit-
    reversed order
14 x5=2; //DIT-FFT, so arranging the input in bit-
    reversed order
15 x3=2; //DIT-FFT, so arranging the input in bit-
    reversed order
16 x7=2; //DIT-FFT, so arranging the input in bit-
    reversed order
17 //Stage I computation
18 x0a=x4+x0; //Computing Stage-I output at line 1
19 disp(x0a, 'Stage-I output at line 1=')
20 x4b=(x4-x0)*(-1); //Computing Stage-I output at line
    2
21 disp(x4b, 'Stage-I output at line 2=')
22 x2c=x6+x2; //Computing Stage-I output at line 3
23 disp(x2c, 'Stage-I output at line 3=')
24 x6d=(x6-x2)*(-1); //Computing Stage-I output at line
    4
25 disp(x6d, 'Stage-I output at line 4=')
26 x1e=x5+x1; //Computing Stage-I output at line 5
27 disp(x1e, 'Stage-I output at line 5=')
28 x5f=(x5-x1)*(-1); //Computing Stage-I output at line
    6
29 disp(x5f, 'Stage-I output at line 6=')
30 x3g=x7+x3; //Computing Stage-I output at line 7
31 disp(x3g, 'Stage-I output at line 7=')
32 x7h=(x7-x3)*(-1); //Computing Stage-I output at line
    8
33 disp(x7h, 'Stage-I output at line 8=')
34
35 //Stage-I output at line 4 and line 8 is to be
    multiplied by twiddle factor having value (-j)
36 x6d1=(x6d)*(-sqrt(-1));
37 x7h1=(x7h)*(-sqrt(-1));
38 disp(x6d1, 'Stage-I output(i.e. input to stage-II)
    after multiplication by twiddle factor value of
    (-j) at line 4 =')
39 disp(x7h1, 'Stage-I output(i.e. input to stage-II)

```

```

    after multiplication by twiddle factor value of
    (-j) at line 8 =')
40
41 //Stage-II Computations
42 x0a_stageII_output=x2c+x0a;//Computing Stage-II
    output at line 1
43 disp(x0a_stageII_output,'Stage-II output at line 1='
    )
44 x4b_stageII_output=x6d1+x4b;//Computing Stage-II
    output at line 2
45 disp(x4b_stageII_output,'Stage-II output at line 2='
    )
46 x2c_stageII_output=(x2c-x0a)*(-1);//Computing Stage-
    II output at line 3
47 disp(x2c_stageII_output,'Stage-II output at line 3='
    )
48 x6d_stageII_output=(x6d1-x4b)*(-1);//Computing Stage
    -II output at line 4
49 disp(x6d_stageII_output,'Stage-II output at line 4='
    )
50 x1e_stageII_output=x3g+x1e;//Computing Stage-II
    output at line 5
51 disp(x1e_stageII_output,'Stage-II output at line 5='
    )
52 x5f_stageII_output=x7h1+x5f;//Computing Stage-II
    output at line 6
53 disp(x5f_stageII_output,'Stage-II output at line 6='
    )
54 x3g_stageII_output=(x3g-x1e)*(-1);//Computing Stage-
    II output at line 7
55 disp(x3g_stageII_output,'Stage-II output at line 7='
    )
56 x7h_stageII_output=(x7h1-x5f)*(-1);//Computing Stage
    -II output at line 8
57 disp(x7h_stageII_output,'Stage-II output at line 8='
    )
58
59 //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
60 x5f_stgII_op_multi_by_tw=(x5f_stageII_output)
    *(0.707-(sqrt(-1))*(0.707));
61 disp(x5f_stgII_op_multi_by_tw,'Stage-II output at
    line 6 after multiplication by twiddle factor=')
62 x3g_stgII_op_multi_by_tw=(x3g_stageII_output)*(-(
    sqrt(-1)));
63 disp(x3g_stgII_op_multi_by_tw,'Stage-II output at
    line 7 after multiplication by twiddle factor=')
64 x7h_stgII_op_multi_by_tw=(x7h_stageII_output)
    *(-0.707-(sqrt(-1))*(0.707));
65 disp(x7h_stgII_op_multi_by_tw,'Stage-II output at
    line 8 after multiplication by twiddle factor=')
66
67 //Stage-III Computations(i.e. Computations for the
    final stage)
68 X0=x1e_stageII_output+x0a_stageII_output;//Computing
    X(0) at last stage
69 X1=x5f_stgII_op_multi_by_tw+x4b_stageII_output;//
    Computing X(1) at last stage
70 X2=x3g_stgII_op_multi_by_tw+x2c_stageII_output;//
    Computing X(2) at last stage
71 X3=x7h_stgII_op_multi_by_tw+x6d_stageII_output;//
    Computing X(3) at last stage
72 X4=(x1e_stageII_output-x0a_stageII_output)*(-1);//
    Computing X(4) at last stage
73 X5=(x5f_stgII_op_multi_by_tw-x4b_stageII_output)
    *(-1);//Computing X(5) at last stage
74 X6=(x3g_stgII_op_multi_by_tw-x2c_stageII_output)
    *(-1);//Computing X(6) at last stage
75 X7=(x7h_stgII_op_multi_by_tw-x6d_stageII_output)
    *(-1);//Computing X(7) at last stage
76 disp(X0,'X(0)=')
77 disp(X1,'X(1)=')
78 disp(X2,'X(2)=')
79 disp(X3,'X(3)=')

```

```
80 disp(X4, 'X(4)=')
81 disp(X5, 'X(5)=')
82 disp(X6, 'X(6)=')
83 disp(X7, 'X(7)=')
84 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

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

```

13 X1_conj=1;//IDIT-FFT,so arranging the input in bit-
    reversed order
14 X5_conj=1;//IDIT-FFT,so arranging the input in bit-
    reversed order
15 X3_conj=1;//IDIT-FFT,so arranging the input in bit-
    reversed order
16 X7_conj=1;//IDIT-FFT,so arranging the input in bit-
    reversed order
17 disp(X0_conj,'X*(0)=')
18 disp(X4_conj,'X*(4)=')
19 disp(X2_conj,'X*(2)=')
20 disp(X6_conj,'X*(6)=')
21 disp(X1_conj,'X*(1)=')
22 disp(X5_conj,'X*(5)=')
23 disp(X3_conj,'X*(3)=')
24 disp(X7_conj,'X*(7)=')
25 //Stage I computation
26 X0a=X4_conj+X0_conj;//Computing Stage-I output at
    line 1
27 disp(X0a,'Stage-I output at line 1=')
28 X4b=(X4_conj-X0_conj)*(-1);//Computing Stage-I
    output at line 2
29 disp(X4b,'Stage-I output at line 2=')
30 X2c=X6_conj+X2_conj;//Computing Stage-I output at
    line 3
31 disp(X2c,'Stage-I output at line 3=')
32 X6d=(X6_conj-X2_conj)*(-1);//Computing Stage-I
    output at line 4
33 disp(X6d,'Stage-I output at line 4=')
34 X1e=X5_conj+X1_conj;//Computing Stage-I output at
    line 5
35 disp(X1e,'Stage-I output at line 5=')
36 X5f=(X5_conj-X1_conj)*(-1);//Computing Stage-I
    output at line 6
37 disp(X5f,'Stage-I output at line 6=')
38 X3g=X7_conj+X3_conj;//Computing Stage-I output at
    line 7
39 disp(X3g,'Stage-I output at line 7')

```

```

40 X7h=(X7_conj-X3_conj)*(-1); //Computing Stage-I
    output at line 8
41 disp(X7h,'Stage-I output at line 8=')
42
43 //Stage-I output at line 4 and line 8 is to be
    multiplied by twiddle factor having value (-j)
44 X6d'=(X6d)*(-sqrt(-1));
45 X7h'=(X7h)*(-sqrt(-1));
46 disp(X6d', 'Stage-I output(i.e. input to stage-II)
    after multiplication by twiddle factor value of
    (-j) at line 4 =')
47 disp(X7h', 'Stage-I output(i.e. input to stage-II)
    after multiplication by twiddle factor value of
    (-j) at line 8 =')
48
49 //Stage-II Computations
50 X0a_stageII_output=X2c+X0a; //Computing Stage-II
    output at line 1
51 disp(X0a_stageII_output, 'Stage-II output at line 1='
    )
52 X4b_stageII_output=X6d'+X4b; //Computing Stage-II
    output at line 2
53 disp(X4b_stageII_output, 'Stage-II output at line 2='
    )
54 X2c_stageII_output=(X2c-X0a)*(-1); //Computing Stage-
    II output at line 3
55 disp(X2c_stageII_output, 'Stage-II output at line 3='
    )
56 X6d_stageII_output=(X6d'-X4b)*(-1); //Computing Stage
    -II output at line 4
57 disp(X6d_stageII_output, 'Stage-II output at line 4='
    )
58 X1e_stageII_output=X3g+X1e; //Computing Stage-II
    output at line 5
59 disp(X1e_stageII_output, 'Stage-II output at line 5='
    )
60 X5f_stageII_output=X7h'+X5f; //Computing Stage-II
    output at line 6

```

```

61 disp(X5f_stageII_output, 'Stage-II output at line 6='
    )
62 X3g_stageII_output=(X3g-X1e)*(-1); //Computing Stage-
    II output at line 7
63 disp(X3g_stageII_output, 'Stage-II output at line 7='
    )
64 X7h_stageII_output=(X7h'-X5f)*(-1); //Computing Stage
    -II output at line 8
65 disp(X7h_stageII_output, 'Stage-II output at line 8='
    )
66
67 //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
68 X5f_stageII_output_multiplied_by_twiddle=(
    X5f_stageII_output)*(0.707-(sqrt(-1))*(0.707));
69 disp(X5f_stageII_output_multiplied_by_twiddle, 'Stage
    -II output at line 6 after multiplication by
    twiddle factor=')
70 X3g_stageII_output_multiplied_by_twiddle=(
    X3g_stageII_output)*(-(sqrt(-1)));
71 disp(X3g_stageII_output_multiplied_by_twiddle, 'Stage
    -II output at line 7 after multiplication by
    twiddle factor=')
72 X7h_stageII_output_multiplied_by_twiddle=(
    X7h_stageII_output)*(-0.707-(sqrt(-1))*(0.707));
73 disp(X7h_stageII_output_multiplied_by_twiddle, 'Stage
    -II output at line 8 after multiplication by
    twiddle factor=')
74
75 //Stage-III Computations(i.e. Computations for the
    final stage)
76 x0_star=(1/8)*(X1e_stageII_output+X0a_stageII_output
    ); //Computing x*(0) at last stage
77 x1_star=(1/8)*(
    X5f_stageII_output_multiplied_by_twiddle+
    X4b_stageII_output); //Computing x*(1) at last

```

```

    stage
78 x2_star=(1/8)*(
    X3g_stageII_output_multiplied_by_twiddle+
    X2c_stageII_output); //Computing x*(2) at last
    stage
79 x3_star=(1/8)*(
    X7h_stageII_output_multiplied_by_twiddle+
    X6d_stageII_output); //Computing x*(3) at last
    stage
80 x4_star=(1/8)*((X1e_stageII_output -
    X0a_stageII_output)*(-1)); //Computing x*(4) at
    last stage
81 x5_star=(1/8)*((
    X5f_stageII_output_multiplied_by_twiddle -
    X4b_stageII_output)*(-1)); //Computing x*(5) at
    last stage
82 x6_star=(1/8)*((
    X3g_stageII_output_multiplied_by_twiddle -
    X2c_stageII_output)*(-1)); //Computing x*(6) at
    last stage
83 x7_star=(1/8)*((
    X7h_stageII_output_multiplied_by_twiddle -
    X6d_stageII_output)*(-1)); //Computing x*(7) at
    last stage
84 disp(x0_star, 'x*(0)=')
85 disp(x1_star, 'x*(1)=')
86 disp(x2_star, 'x*(2)=')
87 disp(x3_star, 'x*(3)=')
88 disp(x4_star, 'x*(4)=')
89 disp(x5_star, 'x*(5)=')
90 disp(x6_star, 'x*(6)=')
91 disp(x7_star, 'x*(7)=')
92 disp({,x0_star,x1_star,x2_star,x3_star,x4_star,
    x5_star,x6_star,x7_star,},'x*(n)=')
93 x0_star_real=real(x0_star);
94 x0_star_imag_conj=(-1)*(imag(x0_star));
95 x1_star_real=real(x1_star);
96 x1_star_imag_conj=(-1)*(imag(x1_star));

```

```

97 x2_star_real=real(x2_star);
98 x2_star_imag_conj=(-1)*(imag(x2_star));
99 x3_star_real=real(x3_star);
100 x3_star_imag_conj=(-1)*(imag(x3_star));
101 x4_star_real=real(x4_star);
102 x4_star_imag_conj=(-1)*(imag(x4_star));
103 x5_star_real=real(x5_star);
104 x5_star_imag_conj=(-1)*(imag(x5_star));
105 x6_star_real=real(x6_star);
106 x6_star_imag_conj=(-1)*(imag(x6_star));
107 x7_star_real=real(x7_star);
108 x7_star_imag_conj=(-1)*(imag(x7_star));
109 x0=x0_star_real+x0_star_imag_conj;
110 x1=x1_star_real+x1_star_imag_conj;
111 x2=x2_star_real+x2_star_imag_conj;
112 x3=x3_star_real+x3_star_imag_conj;
113 x4=x4_star_real+x4_star_imag_conj;
114 x5=x5_star_real+x5_star_imag_conj;
115 x6=x6_star_real+x6_star_imag_conj;
116 x7=x7_star_real+x7_star_imag_conj;
117 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

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

```

13 x4=0; //DIF-FFT, so arranging the input in natural
    order
14 x5=2; //DIF-FFT, so arranging the input in natural
    order
15 x6=1; //DIF-FFT, so arranging the input in natural
    order
16 x7=2; //DIF-FFT, so arranging the input in natural
    order
17 //Stage I computation
18 x0a=x4+x0; //Computing Stage-I output at line 1
19 disp(x0a, 'Stage-I output at line 1=')
20 x1b=x5+x1; //Computing Stage-I output at line 2
21 disp(x1b, 'Stage-I output at line 2=')
22 x2c=x6+x2; //Computing Stage-I output at line 3
23 disp(x2c, 'Stage-I output at line 3=')
24 x3d=x7+x3; //Computing Stage-I output at line 4
25 disp(x3d, 'Stage-I output at line 4=')
26 x4e=(x4-x0)*(-1); //Computing Stage-I output at line
    5
27 disp(x4e, 'Stage-I output at line 5=')
28 x5f=(x5-x1)*(-1); //Computing Stage-I output at line
    6
29 disp(x5f, 'Stage-I output at line 6=')
30 x6g=(x6-x2)*(-1); //Computing Stage-I output at line
    7
31 disp(x6g, 'Stage-I output at line 7')
32 x7h=(x7-x3)*(-1); //Computing Stage-I output at line
    8
33 disp(x7h, 'Stage-I output at line 8=')
34
35 //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
36 x5f_stageI_output_multiplied_by_twiddle=(x5f)
    *(0.707-(sqrt(-1))*(0.707));
37 disp(x5f_stageI_output_multiplied_by_twiddle, 'Stage-
    I output at line 6 after multiplication by

```

```

    twiddle factor=')
38 x6g_stageI_output_multiplied_by_twiddle=(x6g)*(-(
    sqrt(-1)));
39 disp(x6g_stageI_output_multiplied_by_twiddle, 'Stage-
    I output at line 7 after multiplication by
    twiddle factor=')
40 x7h_stageI_output_multiplied_by_twiddle=(x7h)
    *(-0.707-(sqrt(-1))*(0.707));
41 disp(x7h_stageI_output_multiplied_by_twiddle, 'Stage-
    I output at line 8 after multiplication by
    twiddle factor=')
42
43 //Stage-II Computations
44 x0a_stageII_output=x2c+x0a; //Computing Stage-II
    output at line 1
45 disp(x0a_stageII_output, 'Stage-II output at line 1='
    )
46 x1b_stageII_output=x3d+x1b; //Computing Stage-II
    output at line 2
47 disp(x1b_stageII_output, 'Stage-II output at line 2='
    )
48 x2c_stageII_output=(x2c-x0a)*(-1); //Computing Stage-
    II output at line 3
49 disp(x2c_stageII_output, 'Stage-II output at line 3='
    )
50 x3d_stageII_output=(x3d-x1b)*(-1); //Computing Stage-
    II output at line 4
51 disp(x3d_stageII_output, 'Stage-II output at line 4='
    )
52 x4e_stageII_output=x6g+x4e; //Computing Stage-II
    output at line 5
53 disp(x4e_stageII_output, 'Stage-II output at line 5='
    )
54 x5f_stageII_output=
    x7h_stageI_output_multiplied_by_twiddle+
    x5f_stageI_output_multiplied_by_twiddle; //
    Computing Stage-II output at line 6
55 disp(x5f_stageII_output, 'Stage-II output at line 6='

```

```

    )
56 x6g_stageII_output=(
    x6g_stageI_output_multiplied_by_twiddle-x4e)*(-1)
    ;//Computing Stage-II output at line 7
57 disp(x6g_stageII_output,'Stage-II output at line 7='
    )
58 x7h_stageII_output=(x7h'-x5f)*(-1);//Computing Stage
    -II output at line 8
59 disp(x7h_stageII_output,'Stage-II output at line 8='
    )
60
61 //Stage-II output at line 4 and line 8 are to be
    multiplied by twiddle factor having value (-j)
62 x3d_stageII_output_multiplied_by_twiddle=(
    x3d_stageII_output)*(-(sqrt(-1)));
63 disp(x3d_stageII_output_multiplied_by_twiddle,'Stage
    -II output at line 4 after multiplication by
    twiddle factor=')
64 x7h_stageII_output_multiplied_by_twiddle=(
    x7h_stageII_output)*(-(sqrt(-1)));
65 disp(x7h_stageII_output_multiplied_by_twiddle,'Stage
    -II output at line 8 after multiplication by
    twiddle factor=')
66
67 //Stage-III Computations(i.e. Computations for the
    final stage)
68 X0=x1b_stageII_output+x0a_stageII_output;//Computing
    X(0) at last stage
69 X4=(x1b_stageII_output-x0a_stageII_output)*(-1);//
    Computing X(4) at last stage
70 X2=x3d_stageII_output_multiplied_by_twiddle+
    x2c_stageII_output;//Computing X(2) at last stage
71 X6=(x3d_stageII_output_multiplied_by_twiddle-
    x2c_stageII_output)*(-1);//Computing X(6) at last
    stage
72 X1=(x5f_stageII_output+x4e_stageII_output);//
    Computing X(1) at last stage
73 X5=(x5f_stageII_output-x4e_stageII_output)*(-1);//

```

```

    Computing X(5) at last stage
74 X3=x7h_stageII_output_multiplied_by_twiddle+
    x6g_stageII_output;//Computing X(3) at last stage
75 X7=(x7h_stageII_output_multiplid_by_twiddle-
    x6g_stageII_output)*(-1);//Computing X(7) at last
    stage
76 disp(X0, 'X(0)=')
77 disp(X4, 'X(4)=')
78 disp(X2, 'X(2)=')
79 disp(X6, 'X(6)=')
80 disp(X1, 'X(1)=')
81 disp(X5, 'X(5)=')
82 disp(X3, 'X(3)=')
83 disp(X7, 'X(7)=')
84 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

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

```

13 X4_conj=-5;//IDIF-FFT,so arranging the input in
    natural order
14 X5_conj=1;//IDIF-FFT,so arranging the input in
    natural order
15 X6_conj=-1;//IDIF-FFT,so arranging the input in
    natural order
16 X7_conj=1;//IDIF-FFT,so arranging the input in
    natural order
17 disp(X0_conj,'X*(0)=')
18 disp(X1_conj,'X*(1)=')
19 disp(X2_conj,'X*(2)=')
20 disp(X3_conj,'X*(3)=')
21 disp(X4_conj,'X*(4)=')
22 disp(X5_conj,'X*(5)=')
23 disp(X6_conj,'X*(6)=')
24 disp(X7_conj,'X*(7)=')
25
26 // Twiddle factor
27 W0=cos(((2*pi)/8)*0)-(sqrt(-1))*sin(((2*pi)/8)*0)
28 W1=cos(((2*pi)/8)*1)-sqrt(-1)*sin(((2*pi)/8)*1)
29 W2=cos(((2*pi)/8)*2)-sqrt(-1)*sin(((2*pi)/8)*2)
30 W3=cos(((2*pi)/8)*3)-sqrt(-1)*sin(((2*pi)/8)*3)
31
32 //Stage I computation
33 x0a=X0_conj+X4_conj
34 x1b=X1_conj+X5_conj
35 x2c=X2_conj+X6_conj
36 x3d=X3_conj+X7_conj
37 x4e=X0_conj+(-1)*X4_conj
38 x5f=X1_conj+(-1)*X5_conj
39 x6g=X2_conj+(-1)*X6_conj
40 x7h=X3_conj+(-1)*X7_conj
41 disp('Stage-I values are')
42 disp(x0a)
43 disp(x1b)
44 disp(x2c)
45 disp(x3d)
46 disp(x4e)

```

```

47 disp(x5f)
48 disp(x6g)
49 disp(x7h)
50 //Stage I output at line 4,5,6 & 7 are to be
    multiplied by factors W0, W1, W2, and W3
    respectively
51 x4e1=x4e*W0
52 x5f1=x5f*W1
53 x6g1=x6g*W2
54 x7h1=x7h*W3
55
56 //Stage II computation
57 x0a_stageII_output=x2c+x0a;//Computing Stage-II
    output at line 1
58 disp(x0a_stageII_output,'Stage-II output at line 1='
    )
59 x1b_stageII_output=x3d+x1b;//Computing Stage-I
    output at line 2
60 disp(x1b_stageII_output,'Stage-II output at line 2='
    )
61 x2c_stageII_output=x0a-x2c;//Computing Stage-I
    output at line 3
62 disp(x2c_stageII_output,'Stage-II output at line 3='
    )
63 x3d_stageII_output=x1b-x3d;//Computing Stage-I
    output at line 4
64 disp(x3d_stageII_output,'Stage-II output at line 4='
    )
65 x4e_stageII_output=x6g+x4e1;//Computing Stage-II
    output at line 5
66 disp(x4e_stageII_output,'Stage-II output at line 5='
    )
67 x5f_stageII_output=x7h+x5f1//Computing Stage-II
    output at line 6
68 disp(x5f_stageII_output,'Stage-II output at line 6='
    )
69 x6g_stageII_output=x4e1+(x6g1*(-1));//Computing
    Stage-II output at line 7

```

```

70 disp(x6g_stageII_output, 'Stage-II output at line 7='
    )
71 x7h_stageII_output=x5f1+(x7h1*(-1)); //Computing
    Stage-II output at line 8
72 disp(x7h_stageII_output, 'Stage-II output at line 8='
    )
73
74 //Stage III computation
75 x0=x0a_stageII_output+x1b_stageII_output; //
76 x4=x0a_stageII_output+(-1)*x1b_stageII_output
77 x2=x3d_stageII_output*((-1)*sqrt(-1))+
    x2c_stageII_output; //at line 2 x3d_stageII_output
    is to be multiplied by factor -j
78 x6=x3d_stageII_output*((-1)*sqrt(-1))*(-1)+
    x2c_stageII_output; //at line 3 x3d_stageII_output
    is to be multiplied by factor -j*(-1)
79 x1=x4e_stageII_output+x5f_stageII_output
80 x5=x4e_stageII_output-x5f_stageII_output
81 x3=x7h_stageII_output*((-1)*sqrt(-1))+
    x6g_stageII_output; //at line 7 x7h_stageII_output
    is to be multiplied by factor -j
82 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)
83 //final computation
84 x0_star=(1/8)*(x0)
85 disp(x0_star, 'x*(0)=')
86 x4_star=(1/8)*(x4)
87 disp(x4_star, 'x*(4)=')
88 x2_star=(1/8)*(x2)
89 disp(x2_star, 'x*(2)=')
90 x6_star=(1/8)*(x6)
91 disp(x6_star, 'x*(6)=')
92 x1_star=(1/8)*(x1)
93 disp(x1_star, 'x*(1)=')
94 x5_star=(1/8)*(x5)
95 disp(x5_star, 'x*(5)=')

```

```

96 x3_star=(1/8)*(x3)
97 disp(x3_star, 'x*(3)=')
98 x7_star=(1/8)*(x7)
99 disp(x7_star, 'x*(7)=')
100
101 disp({,x0_star,x1_star,x2_star,x3_star,x4_star,
        x5_star,x6_star,x7_star,},'x*(n)=')
102 x0_star_real=real(x0_star);
103 x0_star_imag_conj=(-1)*(imag(x0_star));
104 x1_star_real=real(x1_star);
105 x1_star_imag_conj=(-1)*(imag(x1_star));
106 x2_star_real=real(x2_star);
107 x2_star_imag_conj=(-1)*(imag(x2_star));
108 x3_star_real=real(x3_star);
109 x3_star_imag_conj=(-1)*(imag(x3_star));
110 x4_star_real=real(x4_star);
111 x4_star_imag_conj=(-1)*(imag(x4_star));
112 x5_star_real=real(x5_star);
113 x5_star_imag_conj=(-1)*(imag(x5_star));
114 x6_star_real=real(x6_star);
115 x6_star_imag_conj=(-1)*(imag(x6_star));
116 x7_star_real=real(x7_star);
117 x7_star_imag_conj=(-1)*(imag(x7_star));
118 x0=x0_star_real+x0_star_imag_conj;
119 x1=x1_star_real+x1_star_imag_conj;
120 x2=x2_star_real+x2_star_imag_conj;
121 x3=x3_star_real+x3_star_imag_conj;
122 x4=x4_star_real+x4_star_imag_conj;
123 x5=x5_star_real+x5_star_imag_conj;
124 x6=x6_star_real+x6_star_imag_conj;
125 x7=x7_star_real+x7_star_imag_conj;
126 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

```
1 //AIM:Compute Kaiser window parameter Beta & its
  minimum length
2 //Software version Scilab 5.5.2
3 //OS windows 10
4 clc;
5 clear;
6 //Let us consider the following specifications :
7 // H ( ) 0 .01 0 0 .25
8 //0.95 H ( ) 1 .05 0.35
  0 .6
9 // H ( ) 0 .01 0.65
10 //The magnitude specifications of the FIR filter is
  given by
11 //  $1 - p^H$  ( )  $1 + p$  for  $0 < p$ 
12 //  $0 < H$  ( )  $s$  for  $s$ 
13 //On comparing ,we get  $1 - p = 0.95$ 
14 del_p=0.05;
```

```

15 del_s=0.01;
16 omega_p=0.6*(%pi);
17 omega_s=0.65*(%pi);
18 del_omega=omega_s-omega_p;
19 // ia minimum of p and minimum of s
20 del=0.05;
21 //Attenuation A is given as
22 A=(( -20)*(log10(del)));
23 disp("dB",A," Attenuation(A)=")
24 //Calculating value of
25 beeta=(A-21)^(0.4)+0.07886+(A-21);
26 disp(beeta," =")
27 //The length of filter is (M+1)
28 //The value of M is calculated as follows
29 M=((A-8)/(2.285*(del_omega)));
30 disp(M,"M=")

```

---

## Experiment: 9

# Design High Pass Butterworth Filter Using Bilinear Transformation

Scilab code Solution 9.0 Experiment Number 9

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

```

17 c=log(a/b)
18 N=(1/2)*(c/(log(ohms/ohmp)))
19 Nr=int (N)
20 x=N-int(N)
21 if(x>0)
22   Nr=Nr+1
23 ohmc=(ohmp/(1/Ap^2-1)^(1/(2*Nr)))
24 //calculation of poles
25 i=0:1:Nr-1;
26 pi_plus=ohmc*exp(%i*(Nr+2*i+1)*(%pi)/(2*Nr))
27 pi_minus=-ohmc*exp(%i*(2+2.*i+1)*(%pi)/(2*Nr))
28 disp(wp, ' p =')
29 disp(ws, ' s =')
30 disp(ohmp, ' p =')
31 disp(ohms, ' s =')
32 disp(N, 'Order (N)=')
33 disp(Nr, 'Integer value of the order:(Nr)=')
34 disp(ohmc, ' c =')
35 disp(pi_plus, ' Poles=')
36 disp(pi_minus, ' Poles=')
37 h=ohmc/(s-(-0.53-0.53*i))
38 h1=ohmc/(s-(-0.53+0.53*i))
39 h2=h*h1;
40 disp(h,h1, 'The analog transfer function will be the
multiplication of the following two terms:');
41 disp(h2, 'The analog transfer function H(s)=')
42 Z=poly(0, "Z")
43 s=(ohmc*ohmp)/((2/T)*((Z-1)/(Z+1)));
44 h3=0.56/(s^2+1.06*s+0.56);
45 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

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

```

17 x3n =[7 8];
18 x3no=7;
19 x3n1=8;
20 hn =[1 2]; //Here length of impulse response array h
              (n) is 2 (i.e. M=2) or Nh=2
21 //Length of each partitioned input i.e x0n to x3n is
              2(i.e. L=2)
22 //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
23 hno=1;
24 hn1=2;
25
26 a=xono*hno;
27 b=xon1*hno;
28 c=xono*hn1;
29 d=xon1*hn1;
30 yon=[a c+b d];
31 disp(yon, 'yon=')
32
33 e=x1no*hno;
34 f=x1n1*hno;
35 g=x1no*hn1;
36 h=x1n1*hn1;
37 y1n=[e g+f h];
38 disp(y1n, 'y1n=')
39
40 i=x2no*hno;
41 j=x2n1*hno;
42 k=x2no*hn1;
43 l=x2n1*hn1;
44 y2n=[i k+j l];
45 disp(y2n, 'y2n=')
46
47 m=x3no*hno;
48 n=x3n1*hno;
49 o=x3no*hn1;
50 p=x3n1*hn1;

```

```
51 y3n=[m o+n p];
52 disp(y3n, 'y3n=')
53
54 yon =[yon,0,0,0,0,0,0]
55 y1n =[0,0,y1n,0,0,0,0]
56 y2n =[0,0,0,0,y2n,0,0]
57 y3n =[0,0,0,0,0,0,y3n ]
58
59 yn=yon+ y1n+y2n+y3n
60 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

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

```

16 x2n =[3 -2 -3 -1 1]; // Partitioned input sequence
17 x3n=[-1 1 1 2 -1]; // Partitioned input sequence
18 x4n=[2 -1 0 0 0 ]; // Partitioned input sequence
19 // Convolution of each partitioned input sequence with hn
20 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];
21 disp(y0n,"y0n=")
22 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];
23 disp(y1n,"y1n=")
24 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];
25 disp(y2n,"y2n=")
26 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];
27 disp(y3n,"y3n=")
28 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];
29 disp(y4n,"y4n=")
30 yn0 = y0n (3:5)
31 // (3:5) means that from y0n, select the element from
      3rd to 5th
32 yn1 = y1n (3:5)
33 yn2 = y2n (3:5)
34 yn3 = y3n (3:5)
35 yn4 = y4n (3:5)
36 yn =[yn0;yn1;yn2;yn3;yn4] // Concatenating yn0,yn1,
      yn2,yn3 and yn4
37 disp(yn,"y(n)=")

```

---