Experiment -2.

Aim: Evaluation of FFT using
(a) Decimation in time
(b) Decimation in frequency algorithm

Tools/equipment: System with matlab or octave installed

Theory:

* FFT: The fast fourier Transform (FFT) is an efficient algorithm to compute the discrete fourier transform (DFT) and its inverse. The DFT is a mathematical method to convert a finite sequence of equally spaced samples of a function into a sequence of coefficients of a sum of sinusoidal functions. ordered by frequencies.

(a) **Decimation in Time (DIT):** is one of the primary techniques used to implement the Fast fourier Transform (FFT). It involves breaking down the Discrete fourier Transform (DFT) of a sequence into smaller, more mangeable parts which are combined to give the final result.

DIT approach specifically deals with dividing the sequence in time domain:
(i) Divide the sequence
(ii) Recursive computation
(iii) Combine the results

Mathematically for a sequence $x[n]$ of length N, DIT FFT can be expressed as:

$$X[k] = X_e[k] + W_N^k X_o[k]$$

$$X[k+N/2] = X_e[k] - W_N^k X_o[k]$$

★ properties of ~~DFT~~ Twidde factor (in DFT) used for calculating FFT.

① Symmetry property:

$$W_N^{kn+N/2} = -W_N^{kn}$$

② periodicity:

$$W_N^{kn} = W_N^{k(N+n)}$$

Radix-2 FFT (DIT method):

In DIT radix-2 FFT the time domain sequence is decimated into 2-point sequences. For each 2-point sequence, 2-point DFT can be computed. From result of 2-point DFT the 4-point DFT can be calculated and from result of 4-point DFT the 8-point DFT is calculated, hence this FFT algorith is called radix-2 FFT and since we are decimating in time therefore DIT.

Let $x(n)$ be N sample sequence, we can decimate $x(n)$ into two sequence N/2 samples be $f_1(n)$ & $f_2(n)$. Let $f_1(n)$ consists of even numbered samples of $x(n)$ & $f_2(n)$ consists of odd numbered samples of $x(n)$

$$f_1(n) = x(2n) \quad \text{for } n = 0, 1, 2, 3 \ldots \frac{N}{2}-1 \quad (\text{even})$$

$$f_2(n) = x(2n+1) \quad \text{for } n = 0, 1, 2, 3 \ldots \frac{N}{2}-1 \quad (\text{odd})$$

$$F_1(k) = \sum_{n=0}^{N/2-1} f_1(n)\, W_{N/2}^{kn}$$

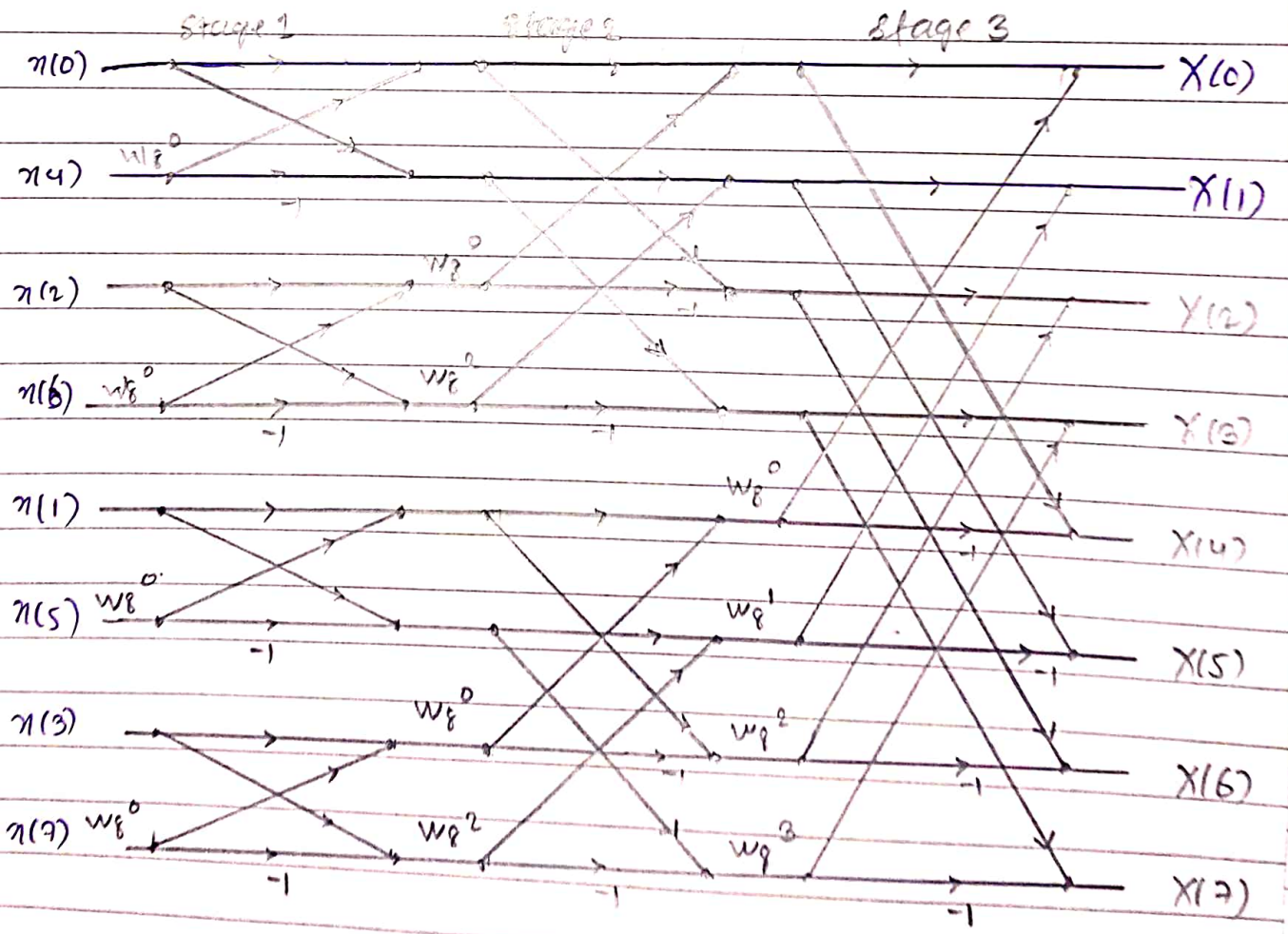
$$F_2(k) = \sum_{n=0}^{N/2-1} f_2(n)\, W_{N/2}^{kn}$$

Now-point DFT $X(K)$ in terms of $N/2$ point DFTs $F_1(K)$ & $F_2(K)$ is given by

$$X(K) = F_1(K) + W_N^K F_2(K) \quad \text{where} \quad K = 0, 1, 2, \cdots (N-1)$$

The decimation of the data sequence can be repeated again & again until the resulting sequences are reduced to 2-point sequences.

Flow graph for 8 point DFT (Butterfly computation) using DIT algorithm:



where $X[k]$ = DFT of original sequence
$X_E[k]$ = DFT of even-indexed elements
$X_O[k]$ = DFT of odd-indexed elements
W_N^k = twiddle factor

code: for DIT of a sequence:

```
clear;
close all;
x=input('Enter the sequence:');
N=length(x);
a=zeros(1,N);
z=zeros(1,N);
y=1:N;
p=bitreverorder(y);
t=exp(-i*2*pi/N);
for i=1:N
  b=p(i);
  a(b)=x(i);
end

for m=1:log2(N)
    k=0:(2^(m-1))-1;
    j=1;
    i=1;
    for q=1:N/2
        z(i)=a(i)+a(i+2^(m-1))*t^k(j);
        z(i+2^(m-1))=a(i)-a(i+2^(m-1))*t^k(j);
        j=j+1;
      if(mod(i,2^(m-1))==0)
          i=i+2^(m-1)+1;
```

```
        j=1;
    else
        i=i+1;
    end
end
a=z;
end
k=0:1:N-1;
subplot(3,1,1);
disp(x);
stem(k,x);
xlabel('time'); ylabel('magnitude');
title('original sequence'); grid on;
disp(abs(a));
subplot(3,1,2);
stem(k, abs(a));
xlabel('Frequency'); ylabel('magnitude');
title('magnitude spectrum'); grid on;
disp(angle(a));
subplot(3,1,3);
stem(k, angle(a));
xlabel('Frequency'); ylabel('phase');
title('phase spectrum'); grid on;
```

result:

Enter the sequence: [0,1,2,3,4,5,6,7]

~~k=0~~	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7
magnitude:	28.0	13.65	5.65	2.3431	4.00	5.6569	5.6569	5.6569
angle:	0	2.3562	2.3562	-3.1416	3.1416	-2.356	-2.356	-1.57

(b) **Decimation in Frequency (DIF) RADIX-2 FFT:**

In Radix-2 decimation-in-frequency (DIF) FFT algorithm, original sequence s(n) is decomposed into two subsequences as first half and second half of a sequence. There is no need of recording (shuffling) the original sequence as in Radix-2 decimation in time (DIT) FFT algorithm.

In this algorithm N-point time domain sequence is converted into two numbers of N/2 sequence. Then each N/2 point sequence is converted to two numbers of N/4 point sequence and then to N/8 point sequences.

The N-point DFT of x(n) can be realized from two numbers of N/2 point DFTs & N/2 point DFTs can be realized from two numbers of N/4 and N/4 point DFTs can be realised by N/8 and so on.

$$X(2K) = \sum_{n=0}^{N-1} x(n)\, W_N^{-2kn}$$

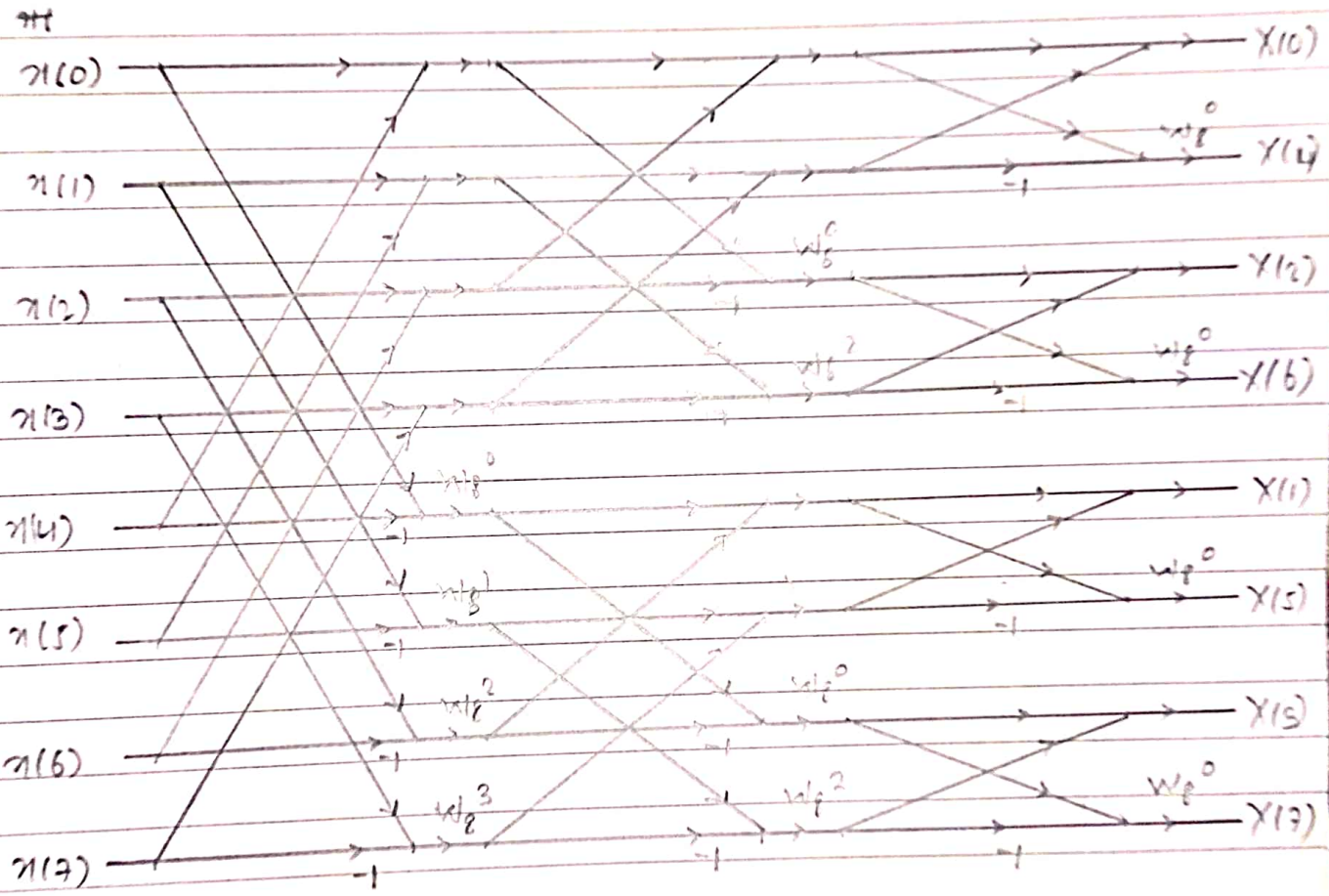
$$= \sum_{n=0}^{N-1} x(n)\, W_{N/2}^{-kn}$$

$$X(2K) = \sum_{n=0}^{N/2-1} x(n)\, W_{N/2}^{-kn} + \sum_{n=N/2}^{N-1} x(n)\, W_{N/2}^{-kn}$$

$$= \sum_{n=0}^{N/2-1} x(n)\, W_{N/2}^{-kn} + \sum_{n=0}^{N/2-1} x[n+N/2]\, W_{N/2}^{-k(n+N/2)}$$

$$X(2K) = \sum_{n=0}^{N/2} \left(x(n) + x(n+N/2) \right) W_{N/2}^{-kn}$$

Flow graph for 8-point FFT (Butterfly computation) using DIF algorithm:



code:

```
x = input ('enter the sequence:');
o = x;
N = input ('enter the required length of dft: ');
a = length (x);
if (a < N)
    x = [x zeros (1,(N-a))];
end
```

```
x = bitrevorder(x);
n = log2(N);
X = zeros(1,N);
for m = 1:1:n
    l = 2^(m-1);
    i = 1
    while (i<=(N-1))
        for k = 0:1:(l-1)
            X(i) = x(i) + x(i+1) * exp(-i*2*pi*k/N*(2^(n-m)));
            X(i+1) = x(i) - x(i+l) * exp(-i*2*pi*k/N*(2^(n-m)));
            i = i+1;
            if (k == (l-1))
                i = i+1;
            end
        end
    end
    x = X;
end
disp(['the output is : ', num2str(X)]);
p = abs(X);
q = angle(x);
subplot(3,1,1);
disp(0);
stem(0:7,p);
xlabel('time');
ylabel('magnitude');
title('original sequence');
grid on;
subplot(3,1,2);
```

```
disp(p);
stem(0:7,P);
xlabel('Frequency');
ylabel('magnitude');
title('magnitude spectrum of dft');
grid on;
subplot(3,1,3);
disp(q);
stem(0:7,q);
xlabel('Frequency');
ylabel('phase');
title('phase spectrum of dft');
grid on;
```

Result:

Enter the sequence: [4, -2, 1, -6, -7, 8, -9, 11]

Index:	K=0	K=1	K=2	K=3	K=4	K=5	K=6	K=7
magnitude:	8.00	1.3531	2.9911	2.5671	30.00	5.5112	5.0090	2.5687
angle:	3.1416	0	3.1416	0	3.1416	0	3.1416	0