Digital Signal Processing Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinator
Dr. Divyang Rawal

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

September 2021

Contents

Chapter *8*

Experiment - 8

8.1 Aim of the Experiment

- Radix-2 FFT Algorithm

8.2 Software Used

- MATLAB
- Simulink

8.3 Theory

8.3.1 About Discrete Fourier Transform (DFT) :

The **discrete Fourier transform (DFT)** converts a finite sequence of **equally-spaced samples** of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. An **inverse DFT** is a **Fourier series**, using the DTFT samples as coefficients of complex sinusoids at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The **DFT** is therefore said to be a **frequency domain representation of the original input sequence**. If the original sequence spans all the non-zero values of a function, its DTFT is continuous (and periodic), and the DFT provides discrete samples of one cycle. If the original sequence is one cycle of a periodic function, the DFT provides all the non-zero values of one DTFT cycle.

The DFT is the most important discrete transform, used **to perform Fourier analysis** in many practical applications. It is used in **digital signal processing**. The DFT is also used to efficiently **solve partial differential equations**, and to perform other operations such as **convolutions** or **multiplying large integers**.

8.3.2 About Fast Fourier Transform (FFT) :

A **fast Fourier transform (FFT)** is an **algorithm** that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies. An FFT rapidly computes such transformations by **factorizing the DFT matrix into a product of sparse (mostly zero) factors**. As a result, it manages to reduce the complexity of computing the DFT from $O(N^2)$ which arises if one simply applies the definition of DFT, to $O(NlogN)$, where N is the **data size**.

Fast Fourier transforms are widely used for applications in engineering, music, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805.

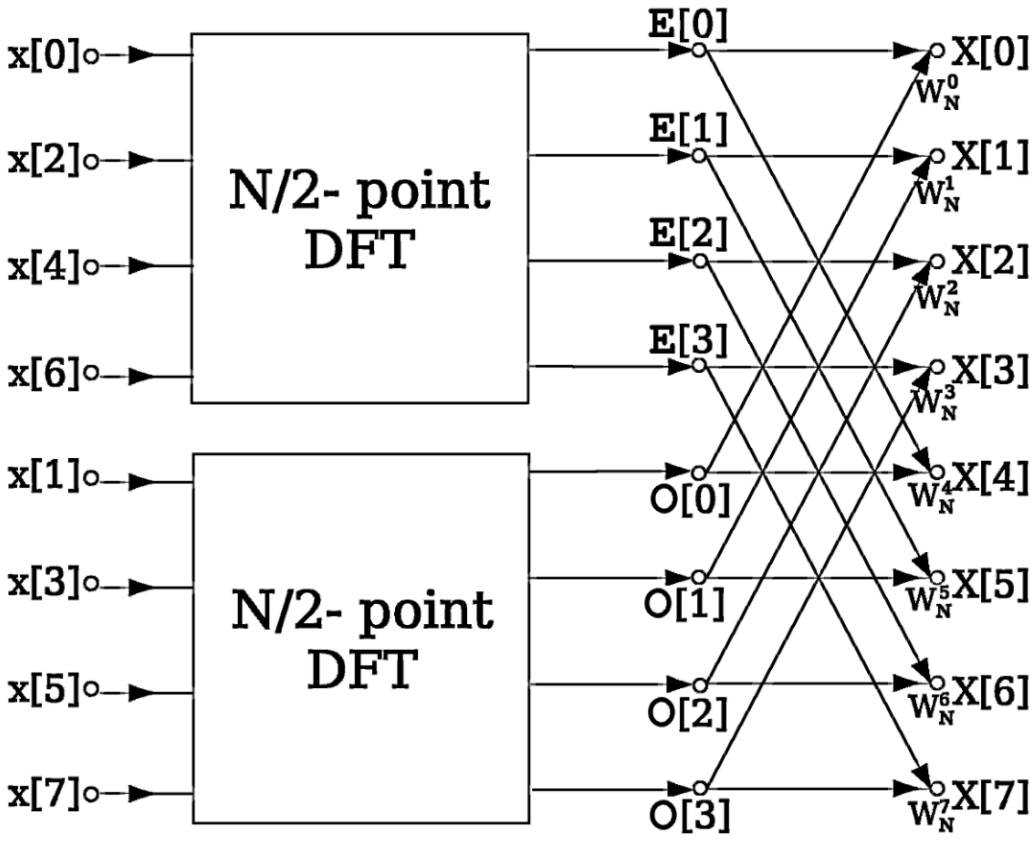

Figure 8.1 Butterfly structure of DIT-FFT Algorithm

The **FFT** is used in **digital recording**, **sampling**, **additive synthesis** and **pitch correction software**. Some of the important applications of the FFT include : fast large-integer and polynomial multiplication, efficient matrix–vector multiplication for Toeplitz, circulant and other structured matrices, filtering algorithms , solving difference equations and computation of isotopic distributions.

8.3.2.1 About DIT-FFT Algorithm :

The steps to be followed to find the **N-point DFT** of a sequence x[n] of length N is as follows :

- The N length sequence can be divided into two $\frac{N}{2}$ point data sequence f1[n] and f2[n], corresponding to the even numbered and odd numbered samples of x[n].
- The DFT of x[n] is given by :

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{8.1}$$

  where k = 0,1,2...N-1 and W_N = exp(-j*2*pi/N)

- We now break x[n] into **even** and **odd** sequence :

$$X[k] = \sum_{n=even} x[n] W_N^{kn} + \sum_{n=odd} x[n] W_N^{kn} \tag{8.2}$$

$$X[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] W_N^{(2m+1)k} \tag{8.3}$$

  The **final expression** for X[k] now becomes :

$$X[k] = \sum_{m=0}^{\frac{N}{2}-1} f1[m] W_{N/2}^{mk} + \sum_{m=0}^{\frac{N}{2}-1} f2[m] W_{N/2}^{mk} \tag{8.4}$$

- The above equation now becomes after simplifying :

$$X[k] = F_1[k] + F_2[k].W_N^k \tag{8.5}$$

  In the above equation , k=0,1,2...N-1 where $F_1[k]$ and $F_2[k]$ are **N/2 point DFT** sequence of $f_1[m]$ and $f_2[m]$.

- Due to **periodicity** of $F_1[k]$ and $F_2[k]$ (which is $F_i[k + \frac{N}{2}] = F_i[k]$), the equations for the both halves are as follows :

$$X[k] = F_1[k] + F_2[k].W_N^k \tag{8.6}$$

$$X[k + \frac{N}{2}] = F_1[k] - F_2[k].W_N^k \tag{8.7}$$

  In the above equation , k=0,1,2.... $\frac{N}{2}$ - 1

8.4 Code and results

8.4.1 Simulating 2-point fft using radix2-fft method and verifying with inbuilt function :

```
% 19ucc023
% Mohit Akhouri
% Experiment 8 - Observation 1

% In this code, we will take a random 2-length sequence x[n]
% Then we will find its FFT using both INBUILT function fft and
% USER-DEFINED function my_dit_fft and also plot the graphs
 respectively

clc;
clear all;
close all;

N = 2; % Size of the seqeunce x[n]
x = randperm(4,N); % using randperm(k,N) to select RANDOM N integers
 from range 1-k , here it selects N=2 integers from range 1-4

% Plot of random seqeunce x[n]
figure;
stem(x,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input Sequence x[n] of
 length = 2');
grid on;

fft_inbuilt = fft(x,N); % Calculation of N-point DFT via INBUILT
 function fft
fft_user_defined = my_dit_fft(x,N); % Calculation of N-point DFT using
 RADIX-2 fft algorithm via USER-DEFINED function my_dit_fft

% Plot of N-point DFT obtained via INBUILT function fft(x,N)
figure;
subplot(2,1,1);
stem(fft_inbuilt,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('2-point DFT of sequence x[n] using INBUILT function fft(x,N)');
grid on;

% Plot of N-point DFT obtained via USER-DEFINED function
 my_dit_fft(x,N)
subplot(2,1,2);
stem(fft_user_defined,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('2-point DFT of sequence x[n] using USER-DEFINED function my
\_dit\_fft(x,N)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.2 Code for the observation 1

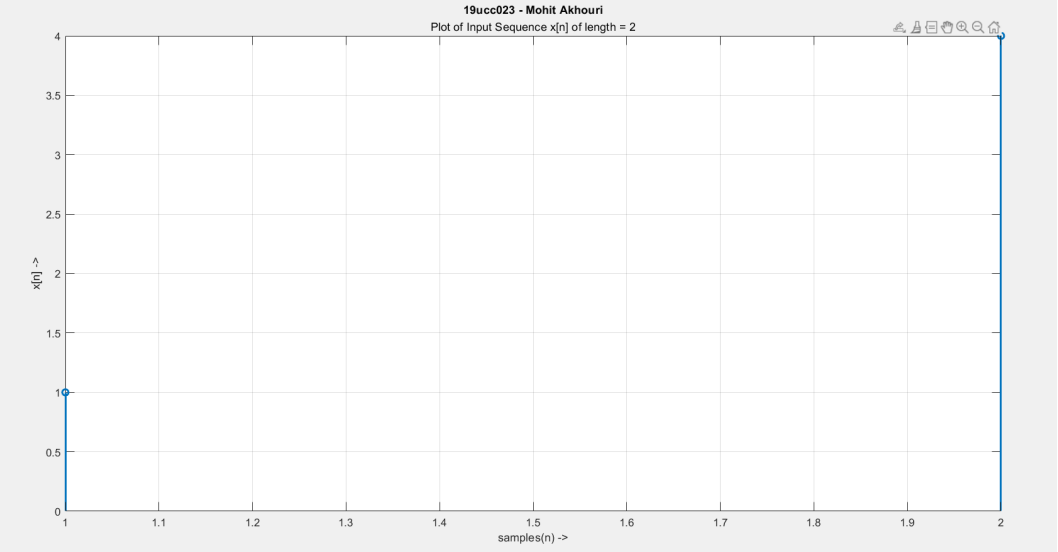

Figure 8.3 Plot of RANDOM Input Sequence of length = 2

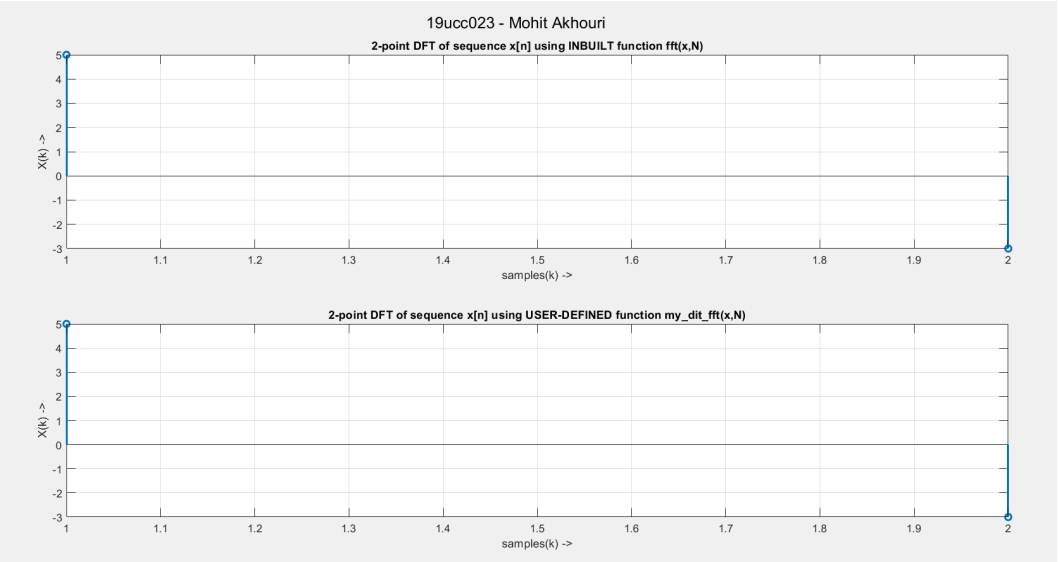

Figure 8.4 Plots of the DFT of input sequence x[n] via both INBUILT and USER-DEFINED functions

8.4.2 Simulating 4-point fft using radix2-fft method and verifying with inbuilt function :

```
% 19ucc023
% Mohit Akhouri
% Experiment 8 - Observation 2

% In this code, we will take a random 4-length sequence x[n]
% Then we will find its FFT using both INBUILT function fft and
% USER-DEFINED function my_dit_fft and also plot the graphs
 respectively

clc;
clear all;
close all;

N = 4; % Size of the seqeunce x[n]
x = randperm(6,N); % using randperm(k,N) to select RANDOM N integers
 from range 1-k , here it selects N=4 integers from range 1-6

% Plot of random seqeunce x[n]
figure;
stem(x,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input Sequence x[n] of
 length = 4');
grid on;

fft_inbuilt = fft(x,N); % Calculation of N-point DFT via INBUILT
 function fft
fft_user_defined = my_dit_fft(x,N); % Calculation of N-point DFT using
 RADIX-2 fft algorithm via USER-DEFINED function my_dit_fft

% Plot of N-point DFT obtained via INBUILT function fft(x,N)
figure;
subplot(2,1,1);
stem(fft_inbuilt,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('4-point DFT of sequence x[n] using INBUILT function fft(x,N)');
grid on;

% Plot of N-point DFT obtained via USER-DEFINED function
 my_dit_fft(x,N)
subplot(2,1,2);
stem(fft_user_defined,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('4-point DFT of sequence x[n] using USER-DEFINED function my
\_dit\_fft(x,N)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.5 Code for the observation 2

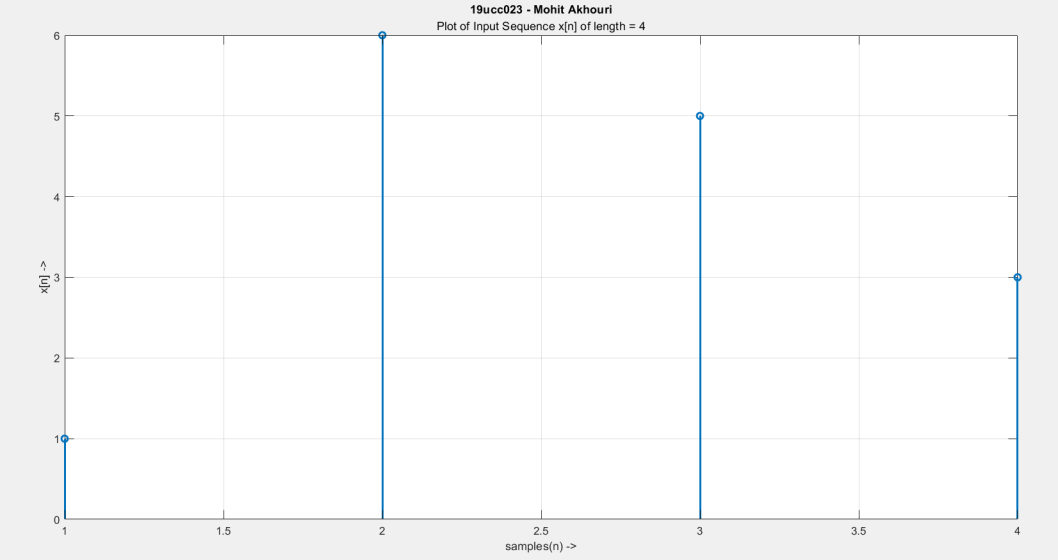

Figure 8.6 Plot of RANDOM Input Sequence of length = 4

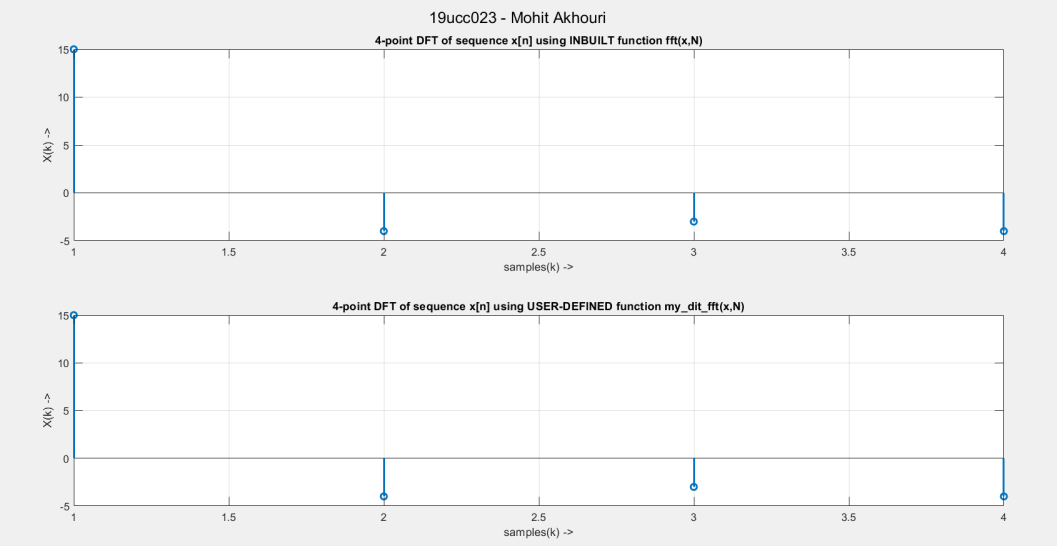

Figure 8.7 Plots of the DFT of input sequence x[n] via both INBUILT and USER-DEFINED functions

8.4.3 Simulating 8-point fft using radix2-fft method and verifying with inbuilt function :

```
% 19ucc023
% Mohit Akhouri
% Experiment 8 - Observation 3

% In this code, we will take a random 8-length sequence x[n]
% Then we will find its FFT using both INBUILT function fft and
% USER-DEFINED function my_dit_fft and also plot the graphs
 respectively

clc;
clear all;
close all;

N = 8; % Size of the seqeunce x[n]
x = randperm(10,N); % using randperm(k,N) to select RANDOM N integers
 from range 1-k , here it selects N=8 integers from range 1-10

% Plot of random seqeunce x[n]
figure;
stem(x,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input Sequence x[n] of
 length = 8');
grid on;

fft_inbuilt = fft(x,N); % Calculation of N-point DFT via INBUILT
 function fft
fft_user_defined = my_dit_fft(x,N); % Calculation of N-point DFT using
 RADIX-2 fft algorithm via USER-DEFINED function my_dit_fft

% Plot of N-point DFT obtained via INBUILT function fft(x,N)
figure;
subplot(2,1,1);
stem(fft_inbuilt,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('8-point DFT of sequence x[n] using INBUILT function fft(x,N)');
grid on;

% Plot of N-point DFT obtained via USER-DEFINED function
 my_dit_fft(x,N)
subplot(2,1,2);
stem(fft_user_defined,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('8-point DFT of sequence x[n] using USER-DEFINED function my
\_dit\_fft(x,N)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.8 Code for the observation 3

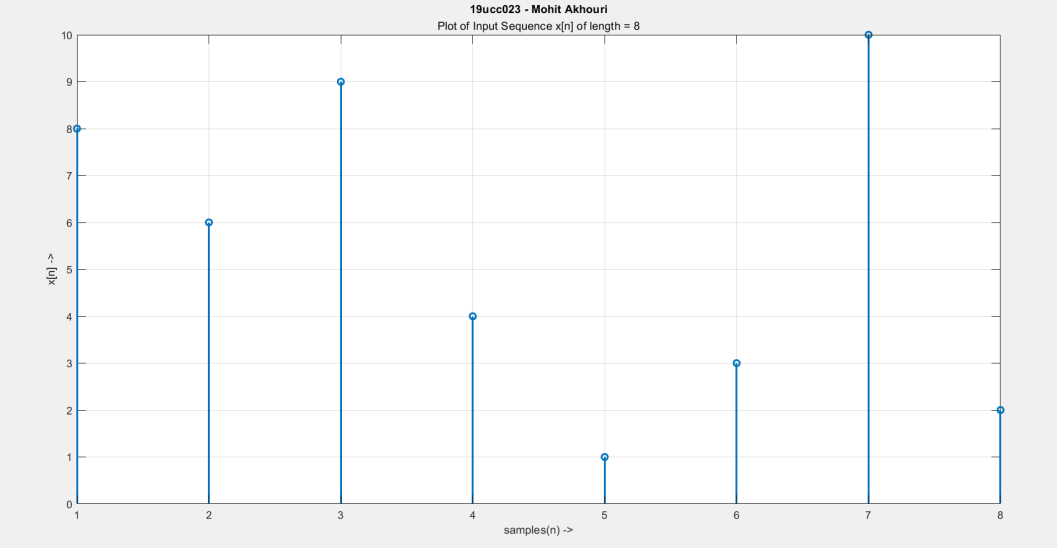

Figure 8.9 Plot of RANDOM Input Sequence of length = 8

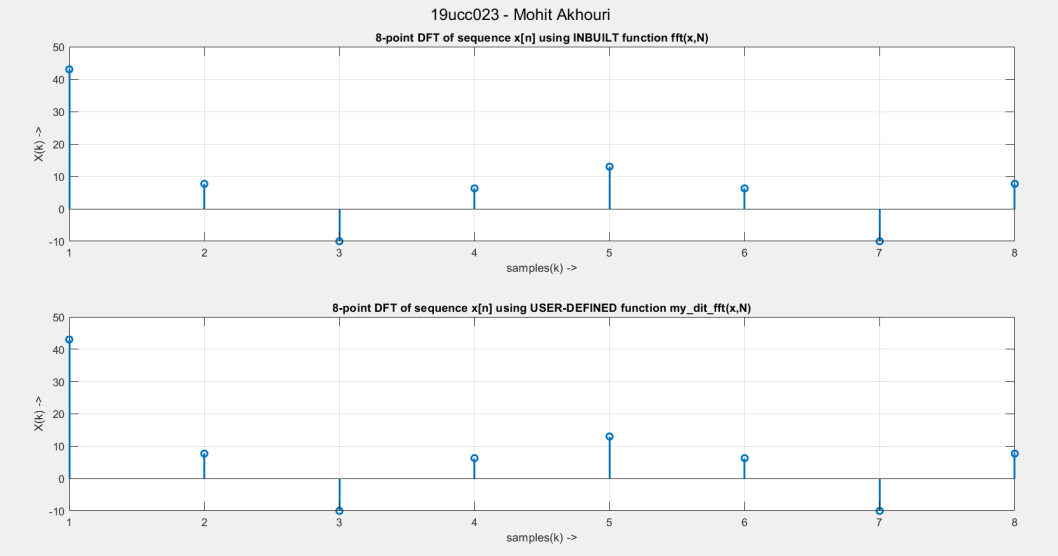

Figure 8.10 Plots of the DFT of input sequence x[n] via both INBUILT and USER-DEFINED functions

8.4.4 Calculation and Plot of Speed Factor vs. N for N=2,4,8,16 and 128 :

```
% 19ucc023
% Mohit Akhouri
% Experiment 8 - Observation 4

% In this code , we will calculate the number of complex
 multiplications in
% both DIRECT METHOD and RADIX-2 FFT METHOD , then we will plot the
 graph
% between speed factor vs. N ( for various values of N = 2,4,8...128 )

clc;
clear all;
close all;

% ALGORITHM : First we will calculate number of complex
 multiplications in
% DIRECT and RADIX-2 FFT METHOD which are as follows :

% Complex Multiplications in Direct Method      = N^2              -
 eqn 1
% Complex Multiplications in Radix-2 fft Method = (N/2)*(log2(N))  -
 eqn 2

% In the above equations N is the length of the input sequence x[n]
% Lastly we will calculate the ratio of eqn 1 and eqn 2 , which is
 defined
% as speed factor and then plot the graph between speed factor and N

N_array = [2 4 8 16 128]; % Array to store the values of N ( size of
 sequence x[n] )

Speed_Factor = zeros(1,5); % Array to store the calculated Speed
 factor for various values of N

comp_mult_direct = 0; % To store the number of complex multiplications
 obtained via Direct method
comp_mult_radix2 = 0; % To store the number of complex multiplications
 obtained via Radix-2 fft method

% Main loop algorithm for calculation of speed factor for various N
 values
for i=1:5

    N = N_array(i); % size stored in N variable

    comp_mult_direct = N^2; % Calculation of complex mult. in Direct
 Method
    comp_mult_radix2 = (N/2)*(log2(N)); % Calculation of complex mult.
 in Radix-2 fft Method

    Speed_Factor(i) = comp_mult_direct / comp_mult_radix2; %
 Calculation of speed factor
```

Figure 8.11 Part 1 of the code for observation 4

```
end

% Plot of Speed factor vs. Different values of N
figure;
stem(N_array,Speed_Factor,'Linewidth',1.8);
xlabel('N ->');
ylabel('Speed Factor ->');
title('19ucc023 - Mohit Akhouri','Plot of Speed Factor vs. N ( length
 of sequence x[n] ) for N = 2,4,8,16,128');
grid on;
```

Figure 8.12 Part 2 of the code for observation 4

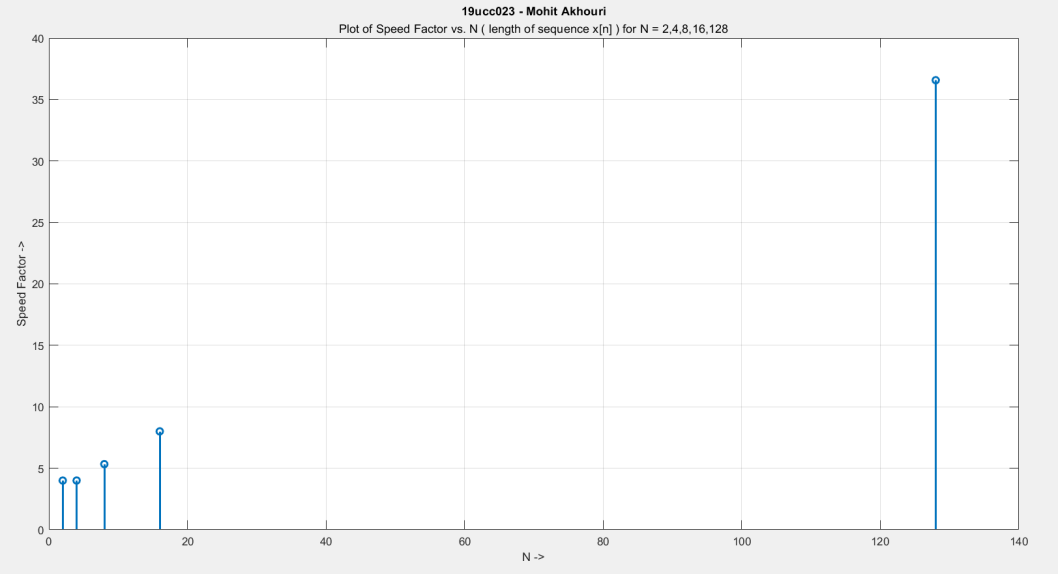

Figure 8.13 Plot of Speed Factor vs. N

8.4.5 Simulink based radix-2 dit fft algorithm :

```
% 19ucc023
% Mohit Akhouri
% Experiment 8 - Observation 5

% This code will make use of Simulink Model for calculation of radix-2
 fft
% of random sequences of length = 2,4 and 8. This code will also
 compare
% the fft calculated via inbuilt function fft with that calculated via
% Simulink Model

sim('Simulink_Observation_5'); % calling the simulink model for
 calculation of N-point fft

xn_2 = [1 2]; % random seqeunce x[n] of length = 2
xn_4 = [3 2 1 4]; % random seqeunce x[n] of length = 4
xn_8 = [5 1 8 7 6 2 3 4]; % random seqeunce x[n] of length = 8

fft_inb_2 = fft(xn_2,2); % 2-point fft of x[n] using INBUILT function
 fft
fft_inb_4 = fft(xn_4,4); % 4-point fft of x[n] using INBUILT function
 fft
fft_inb_8 = fft(xn_8,8); % 8-point fft of x[n] using INBUILT function
 fft

% Plot of input seqeunce x[n] of length = 2
figure;
stem(xn_2,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input sequence x[n] of
 length=2');
grid on;

% Plots of 2-point DFT (via SIMULINK MODEL and via INBUILT FUNCTION
 fft)
figure;
subplot(2,1,1);
stem(out.fft_2point.data,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('2-point DFT of sequence x[n] using SIMULINK MODEL');
grid on;
subplot(2,1,2);
stem(fft_inb_2,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('2-point DFT of sequence x[n] using INBUILT FUNCTION fft');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% Plot of input seqeunce x[n] of length = 4
```

Figure 8.14 Part 1 of the code for observation 5

```
figure;
stem(xn_4,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input sequence x[n] of
 length=4');
grid on;

% Plots of 4-point DFT (via SIMULINK MODEL and via INBUILT FUNCTION
 fft)
figure;
subplot(2,1,1);
stem(out.fft_4point.data,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('4-point DFT of sequence x[n] using SIMULINK MODEL');
grid on;
subplot(2,1,2);
stem(fft_inb_4,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('4-point DFT of sequence x[n] using INBUILT FUNCTION fft');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% Plot of input seqeunce x[n] of length = 8
figure;
stem(xn_8,'Linewidth',1.8);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of Input sequence x[n] of
 length=8');
grid on;

% Plots of 8-point DFT (via SIMULINK MODEL and via INBUILT FUNCTION
 fft)
figure;
subplot(2,1,1);
stem(out.fft_8point.data,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('8-point DFT of sequence x[n] using SIMULINK MODEL');
grid on;
subplot(2,1,2);
stem(fft_inb_8,'Linewidth',1.8);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('8-point DFT of sequence x[n] using INBUILT FUNCTION fft');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 8.15 Part 2 of the code for observation 5

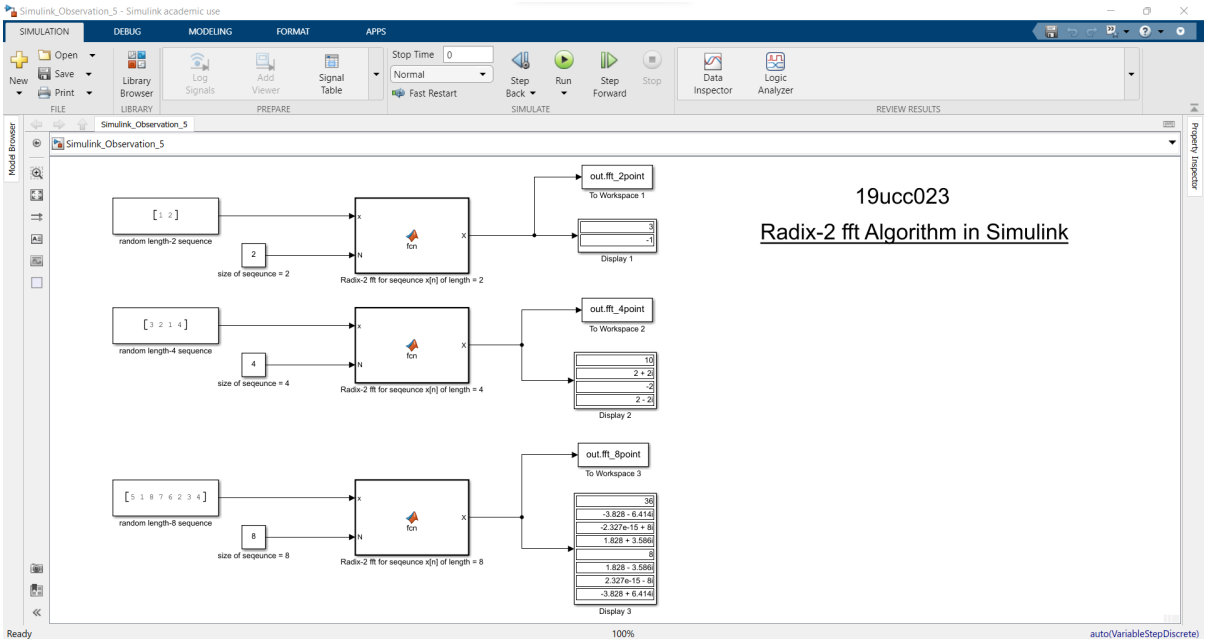

Figure 8.16 Simulink model used for the implementation of radix-2 dit fft algorithm

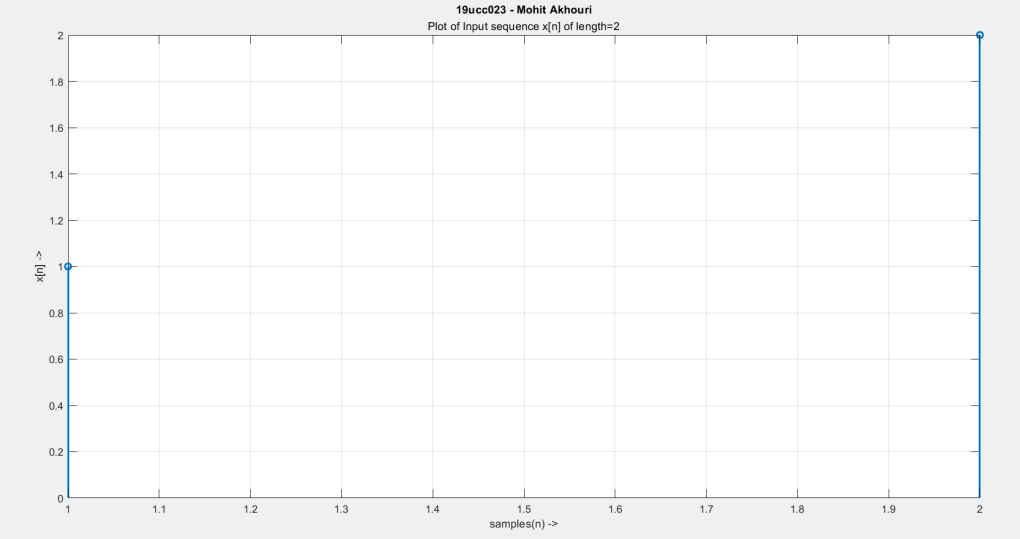

Figure 8.17 Plot of RANDOM Input Sequence of length = 2

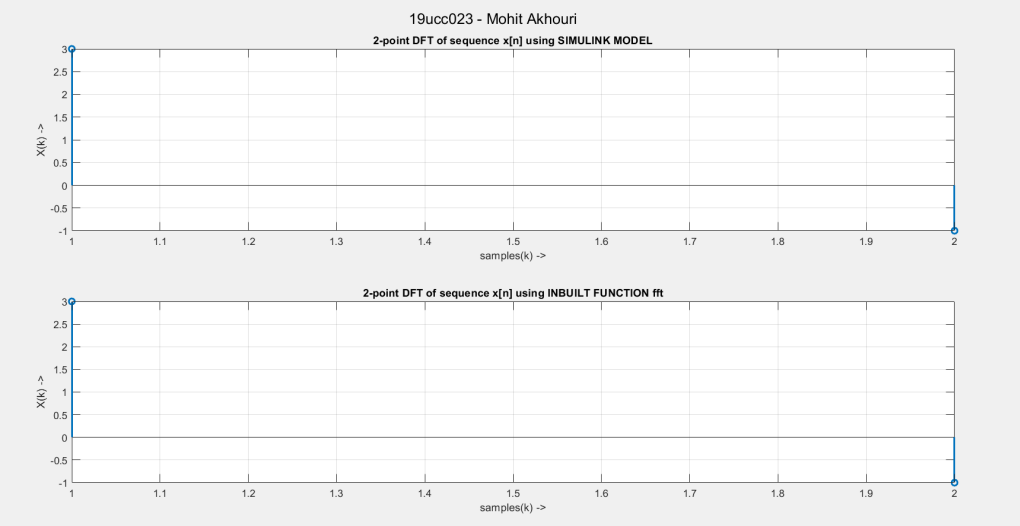

Figure 8.18 Plots of the DFT of input x[n] via both SIMULINK MODEL and INBUILT function

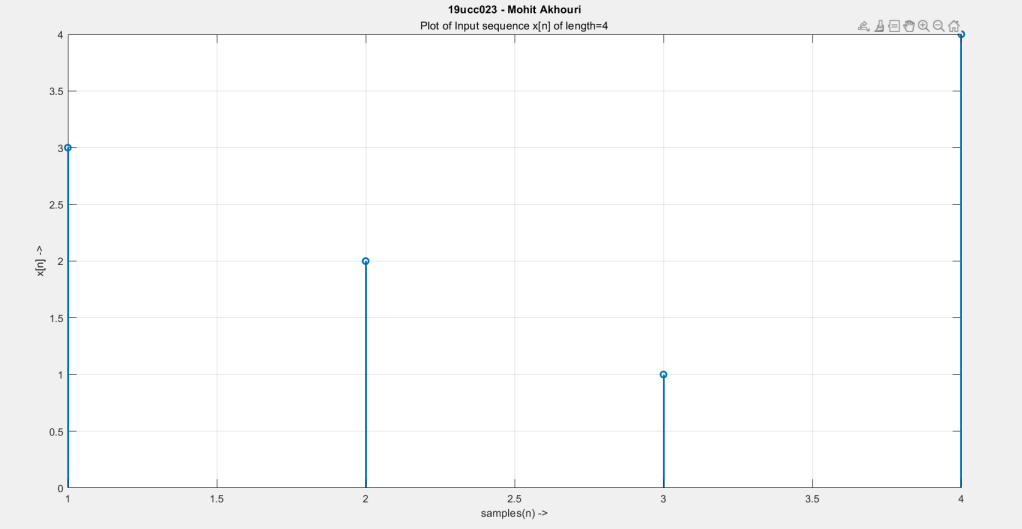

Figure 8.19 Plot of RANDOM Input Sequence of length = 4

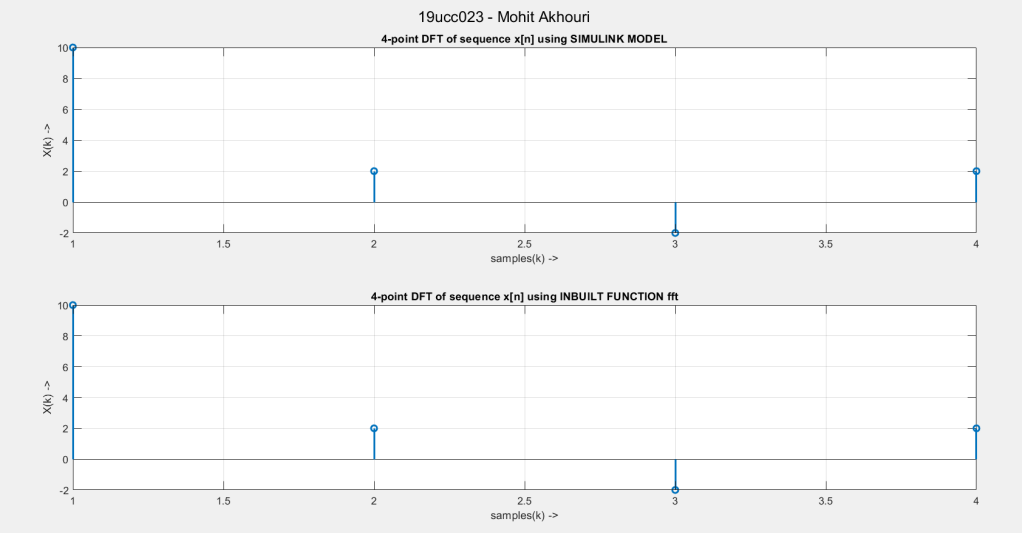

Figure 8.20 Plots of the DFT of input x[n] via both SIMULINK MODEL and INBUILT function

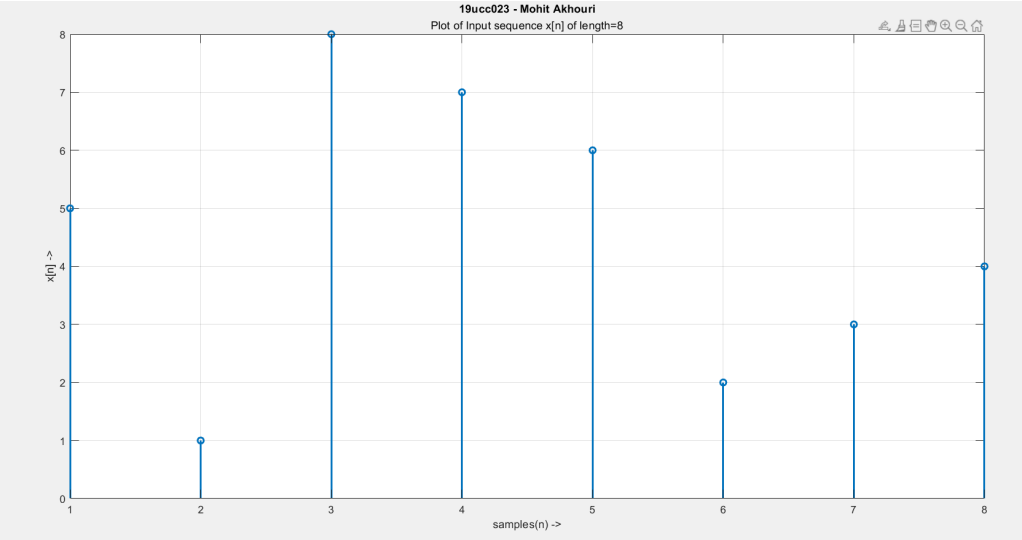

Figure 8.21 Plot of RANDOM Input Sequence of length = 8

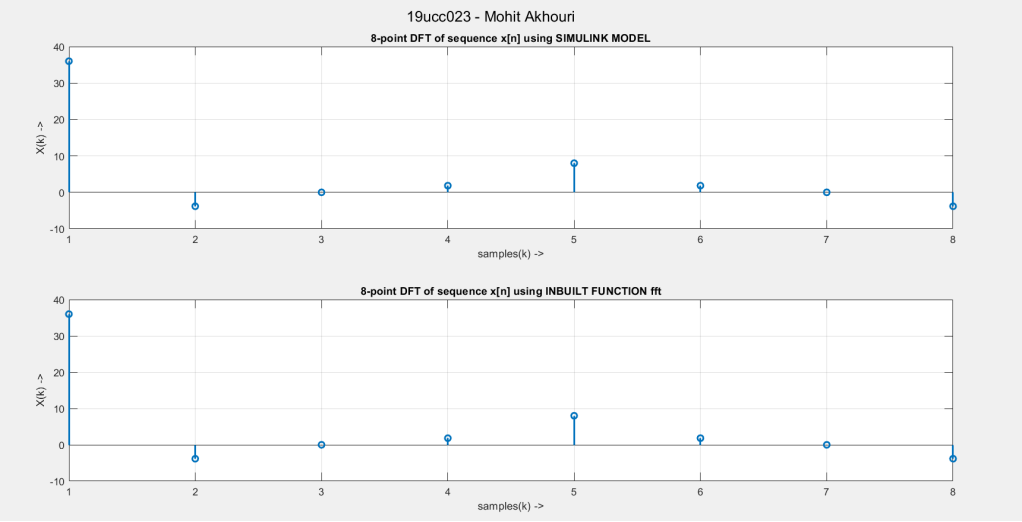

Figure 8.22 Plots of the DFT of input x[n] via both SIMULINK MODEL and INBUILT function

8.4.6 Function used in main codes implementation of radix-2 dit fft algorithm :

8.4.6.1 my_dit_fft.m function code :

```
function [X] = my_dit_fft(x,N)

% 19ucc023
% Mohit Akhouri

% ALGORITHM : This function will calculate the Radix-2 dit
 ( decimation in
% time ) fft for a sequence x[n]
% It divides the x[n] into seqeunce of EVEN and ODD seqeunces and
 calculate
% their N/2 point DFT separately. Lastly , it combines the calculated
 DFT
% to get the final answer of DFT.

X = zeros(1,N); % Output variable to store the calculated DFT
f1 = zeros(1,N/2); % To store the ODD parts of sequence x[n]
f2 = zeros(1,N/2); % To store the EVEN parts of sequence x[n]
Wn = exp(-1j*2*pi/N); % Twiddle factor used in the calculation of DFT

% Main loop algorithm for calculation of DFT via Radix-2 FFT algorithm
for k=1:N/2
    for m=1:N/2
        f1(k) = f1(k) + (x(2*(m-1)+1)*(Wn^(2*(m-1)*(k-1)))); %
 Calculation of N/2 point-DFT of ODD parts of seqeunce x[n]
        f2(k) = f2(k) + (x(2*(m-1)+2)*(Wn^(2*(m-1)*(k-1)))); %
 Calculation of N/2 point-DFT of EVEN parts of seqeunce x[n]
    end
    X(k)     = f1(k) + (f2(k)*(Wn^(k-1))); % Combination of f1(k) and
 f2(k) for N-point DFT of the first half of seqeunce x[n]
    X(k+N/2) = f1(k) - (f2(k)*(Wn^(k-1))); % Combination of f1(k) and
 f2(k) for N-point DFT of the second half of seqeunce x[n]
end

end
```

Figure 8.23 my_dit_fft.m function code to calculate the N-point DFT of a input sequence x[n]

8.5 Conclusion

In this experiment , we learnt the concepts of **Discrete Fourier Transform (DFT)** , **Fast fourier transform (FFT)** and **Radix-2 dit fft algorithm** of Digital Signal Processing. We learnt about the dit-fft algorithm to calculate the DFT of a input sequence x[n] faster. We learnt about two types of fft algorithm - **decimation in frequency (dif-fft)** and **decimation in time (dit-fft)**. We implemented the dit-fft algorithm in MATLAB by splitting the input sequence into **even** and **odd** parts. We also learnt about the **speed factor** for different values of length of input sequence (**N**) and also plotted the graph between them. We observed that speed factor **increases** as we move towards **higher N**. We also learnt about the **butterfly diagram** to efficiently calculate the FFT. We also implemented the algorithm in Simulink and compared the results obtained via MATLAB coding.