Title: Fast Fourier Transform (FFT) in DIT (Decimation in time) and DIF (Decimation in frequency)

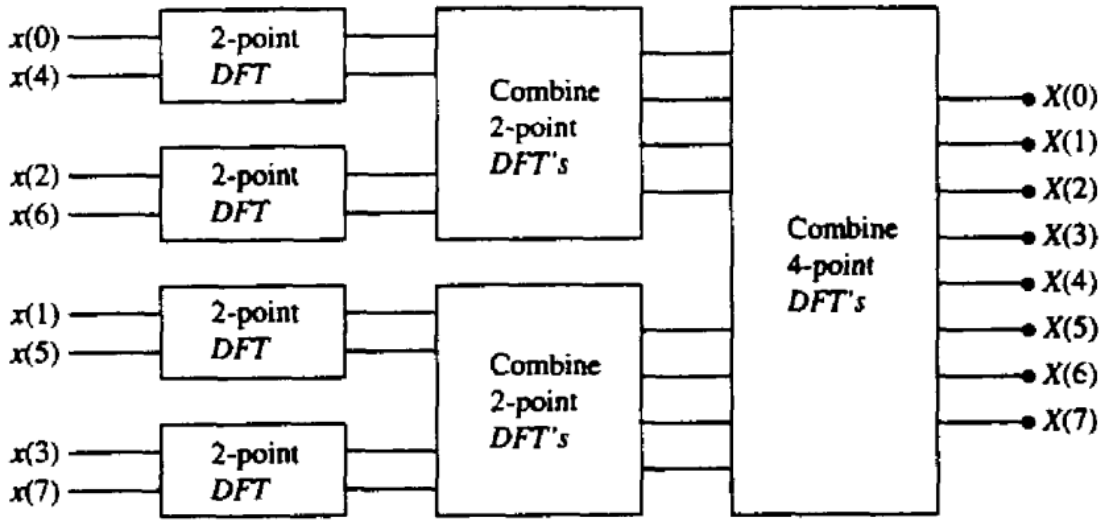
AIM: To perform FFT in DIT and DIF operations on a given pair of sequences.

Objective: To perform FFT in DIT and DIF operations on a given pair of sequences using MATlab.

Fast Fourier Transform (FFT) Fast Fourier Transform is an efficient algorithm developed by Cooley & Tukey in 1965, used to compute the DFT with reduced computations. Due to the efficiency of FFT, it is used for spectrum analysis, convolutions, correlations and linear filtering. FFT reduces the problem of calculating an N – point DFT to that of calculating many smaller – sized DFTs. The properties of the twiddle factor WN used in this algorithm are:

1. $W_N^{k+\frac{N}{2}} = e^{-\frac{j2\pi k}{N}} e^{-j\pi} = -W_N^k$ **(Symmetry Property)**
2. $W_N^{k+N} = e^{-\frac{j2\pi k}{N}} e^{-j2\pi} = W_N^k$ **(periodicity property)**
3. $W_N^m = e^{-\frac{j2\pi m}{N}} = e^{-\frac{j2\pi}{N/m}} = W_{N/m}$

The Decimation – In – Time (DIT) and Decimation – In – Frequency (DIF) FFT algorithms use the “divide – and – conquer” approach. This is possible if the length of the sequence N is chosen as N = rm. here, r is called the radix of the FFT algorithm. The most practically implemented choice for r = 2 leads to radix – 2 FFT algorithms. So, with N = 2m, the efficient computation is achieved by breaking the N – point DFT into two N 2 - point DFTs, then breaking each N 2 - point DFT into two N 4 - point DFTs and continuing this process until 2 – point DFTs are obtained. For N=8, the Decimation – In – Time algorithm decomposition would be



Decimation – In – Frequency (DIF) FFT algorithm

In this algorithm, we decimate the DFT sequence X(k) into smaller and smaller subsequences (Instead of the time – domain sequence x[n]).

$$X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x[n]W_N^{nk}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]W_N^{nk}\, W_N^{\frac{N}{2}k}$$

$$but, W_N^{\frac{N}{2}k} = (-1)^k$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left\{x[n] + (-1)^k x\left[n+\frac{N}{2}\right]\right\} W_N^{nk}, k = 0,1,\dots,N-1$$

Now, decimating X(k) into even and odd – indexed samples,

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left\{x[n] + x\left[n+\frac{N}{2}\right]\right\} W_{\frac{N}{2}}^{nk}, k = 0,1,\dots,\frac{N}{2}-1 \dots\dots eqn.1$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \left\{x[n] - x\left[n+\frac{N}{2}\right]\right\} W_N^n\, W_{\frac{N}{2}}^{nk}, k = 0,1,\dots,\frac{N}{2}-1 \dots\dots eqn.2$$

$$let\ g[n] = x[n] + x\left[n+\frac{N}{2}\right], 0 \le n \le \frac{N}{2}-1 \dots\dots eqn.3$$

$$and, h[n] = \left\{x[n] - x\left[n+\frac{N}{2}\right]\right\} W_N^n, 0 \le n \le \frac{N}{2}-1 \dots\dots eqn.4$$

Substituting equations 3 & 4 in equations 1 & 2 respectively

$$G(k) = X(2k), 0 \le k \le \frac{N}{2}-1$$

$$and, H(k) = X(2k+1)\ 0 \le k \le \frac{N}{2}-1$$

For example, for N=8, using equations 3 & 4, we get

$$g[0] = x[0] + x[4]$$

$$g[1] = x[1] + x[5]$$

$$g[2] = x[2] + x[6]$$

$$g[3] = x[3] + x[7]$$

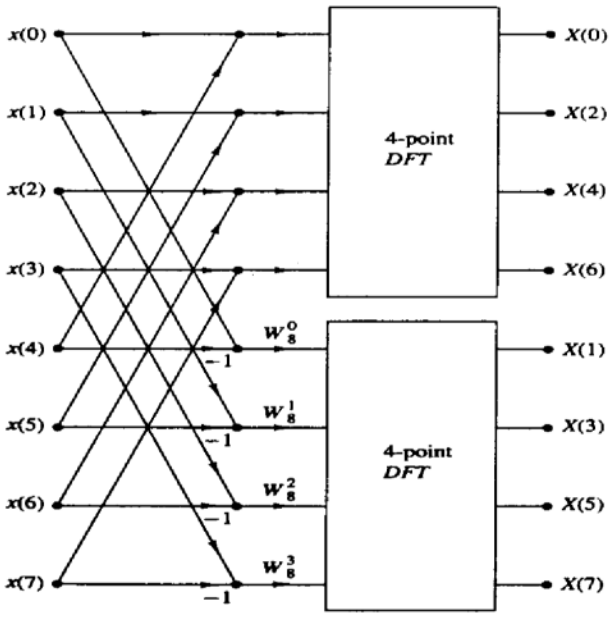
$$h[0] = \{x[0] - x[4]\}W_8^0$$

$$h[1] = \{x[1] - x[5]\}W_8^1$$

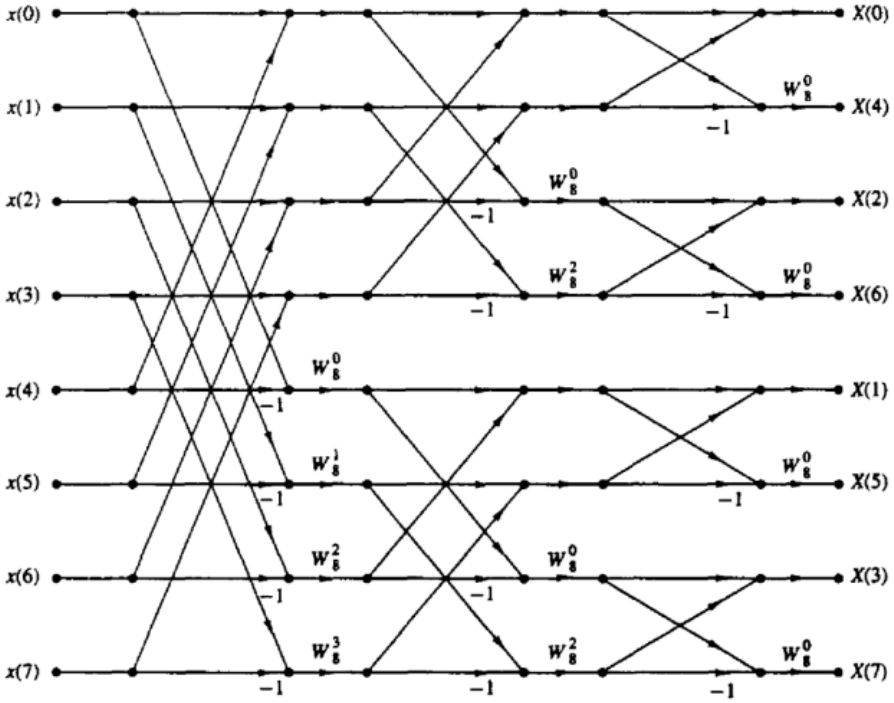
$$h[2] = \{x[2] - x[6]\}W_8^2$$

$$h[3] = \{x[3] - x[7]\}W_8^3$$

the butterfly diagram for the above set of equations is as follows



The above process of decimation is repeated for G(k) and H(k), until we reach a 2 – point sequence. The 2 – point DFT is calculated as in the previous section. The final butterfly diagram for Decimation – In – Frequency FFT algorithm for N = 8 is as follows



In this algorithm, the output X(k) is in bit –reversed order.

Write a function for DIT (Decimation in time) and DIF (Decimation in frequency) algorithms for implementing the Fast Fourier Transform (FFT) in Matlab.

Matlab function to perform FFT Radix-2 DIT

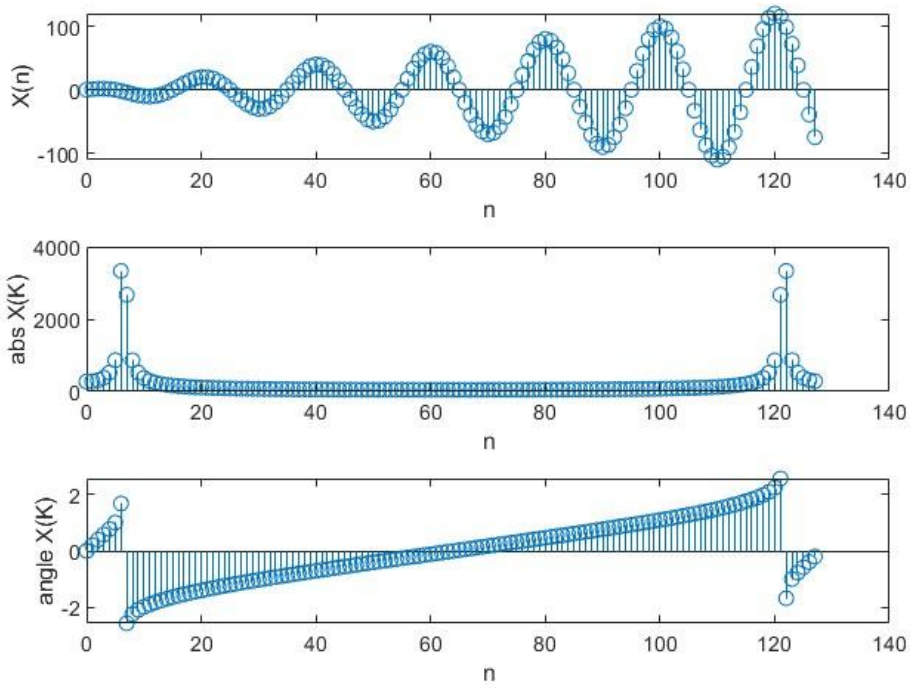
```
%{
To Calculate the FFT by DIT for a given sequence
Author : Sudip Biswas
%}
function xk = DITFFT(xn)
N = length(xn);         % Length of the input sample
r = log2(N);            % number of stages
if (r-fix(r))
error("Cannot find FFT Radix 2 via DIT");
else
  xk = BUTTERFLYDIT(xn);
end

%{
To Calculate the radix 2 butterfly in DIT for a given sequence
Author : Sudip Biswas
%}
function x = BUTTERFLYDIT(xn)
N = length(xn);         % Length of the input sample
r = log2(N);            % number of stages
```

```
if ( N == 2 )
   x(1) = xn(1)+xn(2); % 2 point butterfly when N =2
   x(2) = xn(1)-xn(2);
else
   xn = DITDecimation(xn,r);
   for n=1:N/2
       xeven(n) = xn(n);
   end
   xkeven = BUTTERFLYDIT(xeven);
   for n=(N/2)+1:N
       xodd(n-(N/2)) = xn(n);
   end
   xkodd =BUTTERFLYDIT(xodd);
   for n=1:N/2
        W = exp(-2i*pi*(n-1)/N);
       x(n) = xkeven(n)+(W * xkodd(n));
       x(n+(N/2)) = xkeven(n)-(W * xkodd(n));
   end
end

%{
To perform decimation shift
Author : Sudip Biswas
%}
function x = DITDecimation(xn,r)
N = length(xn);          % Length of the input sample
for n=1:N
       n1 = bitor(bitsll(n-1,1),fix(bitsra(n-1,r-1)),"uint8");
       n1 = bitand(n1,N-1,"uint8");
       x(n) = xn(n1+1);
End
```

Test Function

```
N = 0:127
X = N.*cos(2*pi*N*0.05);
K = DITFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT



Matlab function to perform FFT Radix-2 DIF

```
%{
To Calculate the FFT by DIT for a given sequence
Author : Sudip Biswas
%}
function xk = DITFFT(xn)
N = length(xn);         % Length of the input sample
r = log2(N);            % number of stages
if (r-fix(r))
error("Cannot find FFT Radix 2 via DIF");
else
  xk = BUTTERFLYDIF(xn);
  if (r>1)
      xk = DIFDecimation(xk,r);
  end

end

%{
To Calculate the radix 2 butterfly in DIT for a given sequence
Author : Sudip Biswas
%}
function x = BUTTERFLYDIF(xn)
N = length(xn);         % Length of the input sample
r = log2(N);            % number of stages
if(N>2)
  for n=1:N/2
       xeven(n) = xn(n);
```

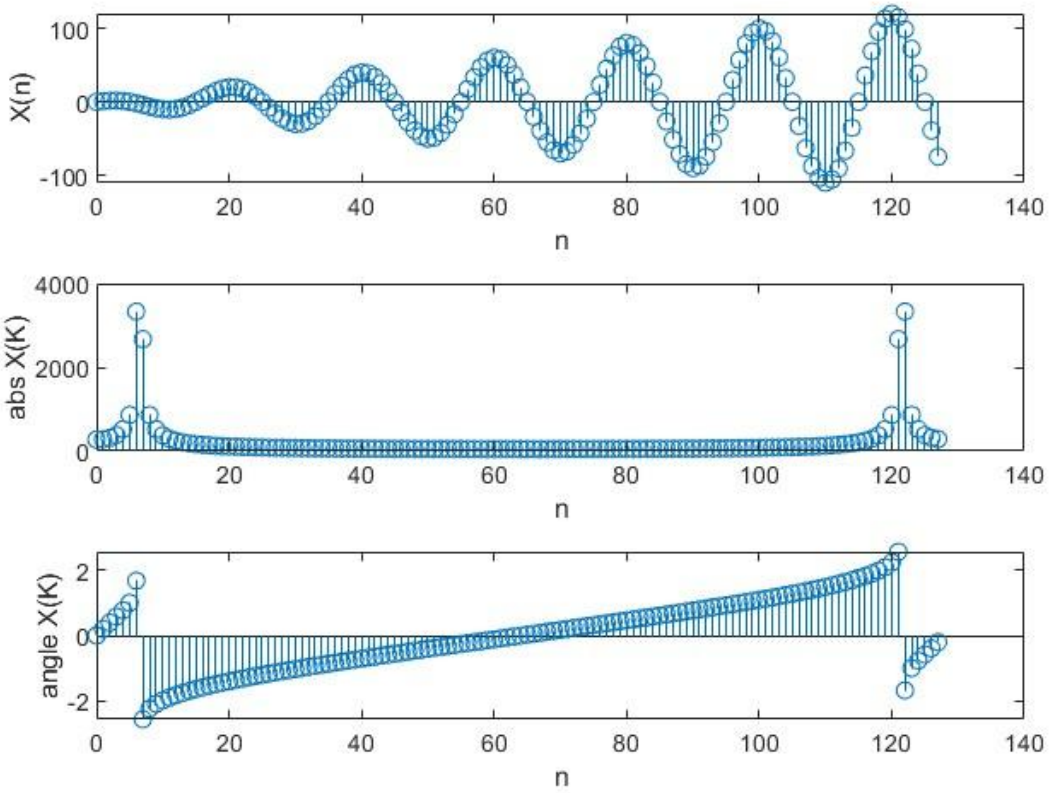
```
  end
  for n=(N/2)+1:N
       xodd(n-(N/2)) = xn(n);
  end
  for n=1:N/2
        W = exp(-2i*pi*(n-1)/N);
       xkeven(n) = xeven(n)+xodd(n);
       xkodd(n) = (xeven(n)-xodd(n))*W;
  end
  xeven = BUTTERFLYDIF(xkeven);
  xodd  = BUTTERFLYDIF(xkodd);
  for n=1:N/2
       x(n) = xeven(n);
  end
  for n=(N/2)+1:N
       x(n) = xodd(n-(N/2));
  end
else
   x(1) = xn(1)+xn(2);
   x(2) = xn(1)-xn(2);
end

%{
To perform decimation shift
Author : Sudip Biswas
%}
function x = DIFDecimation(xn,r)
N = length(xn);          % Length of the input sample
for n=1:N
       n1 = bitor(bitsll(n-1,1),fix(bitsra(n-1,r-1)),"uint8");
       n1 = bitand(n1,N-1,"uint8");
       x(n) = xn(n1+1);
end
r = log2(N/2);            % number of stages
if (r>1)
   for n=1:N/2
       xeven(n) = x(n);
   end
   for n=(N/2)+1:N
       xodd(n-(N/2)) = x(n);
   end
   xeven = DIFDecimation(xeven,r);
   xodd = DIFDecimation(xodd,r);
   for n=1:N/2
       x(n) = xeven(n);
   end
   for n=(N/2)+1:N
       x(n) = xodd(n-(N/2));
   end
end
```

Test Function

```
N = 0:127
```

```
X = N.*cos(2*pi*N*0.05);
K = DIFFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT



Use the function written for (1) to calculate 4-point (a and b) and 8-point (c and d) DFT of the flowing sequences.

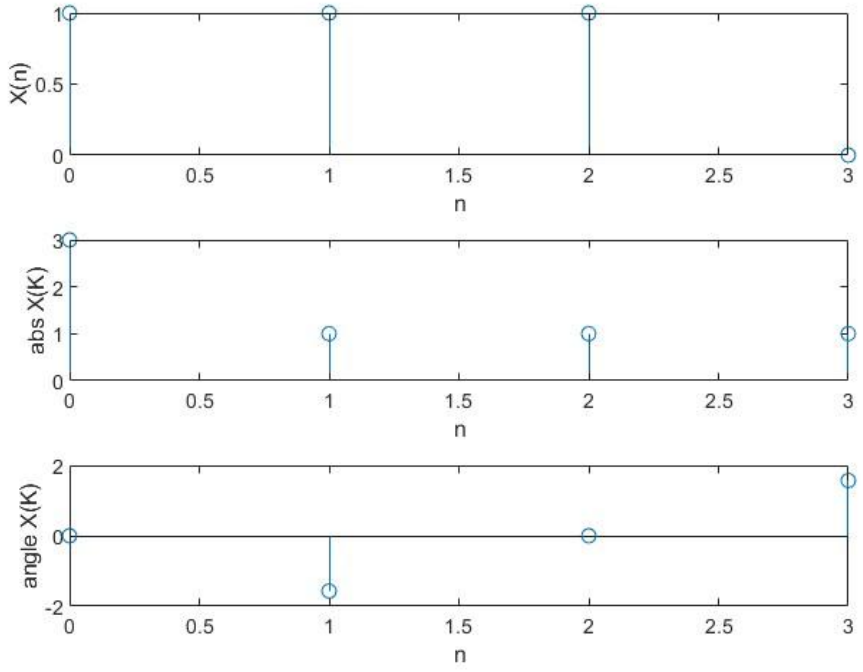
(a) **x(n)={1,1,1,0}**

Matlab Function for DIT

```
N = [0,1,2,3];
X = [1,1,1,0];
```

```
K = DITFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

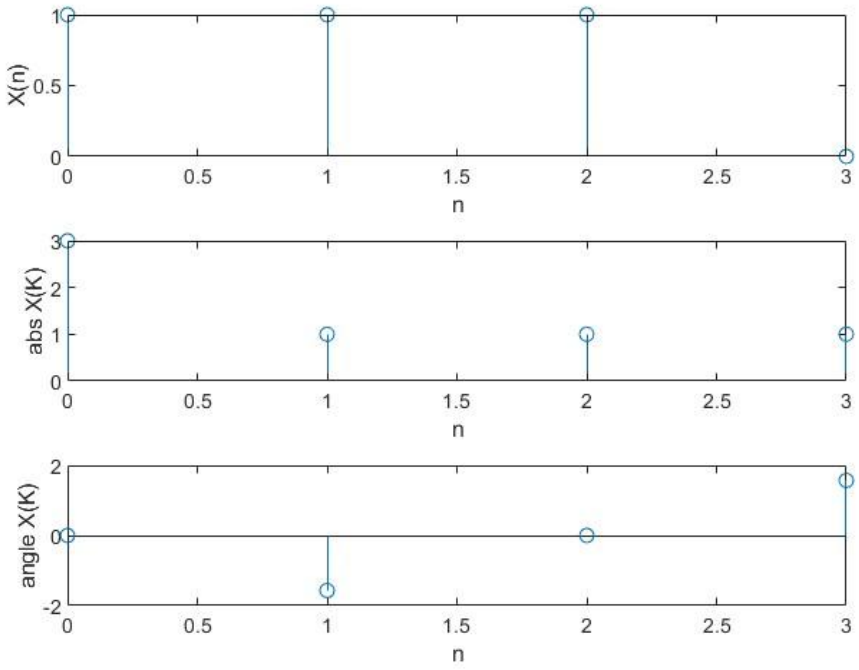
OUTPUT



Matlab Function for DIF

```
N = [0,1,2,3];
X = [1,1,1,0];
K = DIFFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT

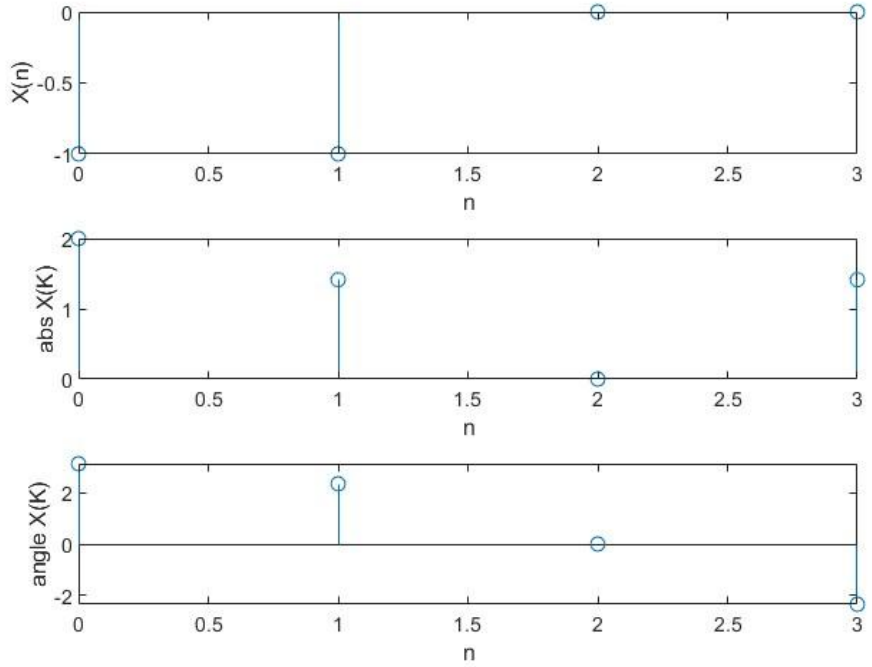


(b) **x(n)={-1, -1,0,0}**

Matlab Function for DIT

```
N = [0,1,2,3];
X = [-1,-1,0,0];
K = DITFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

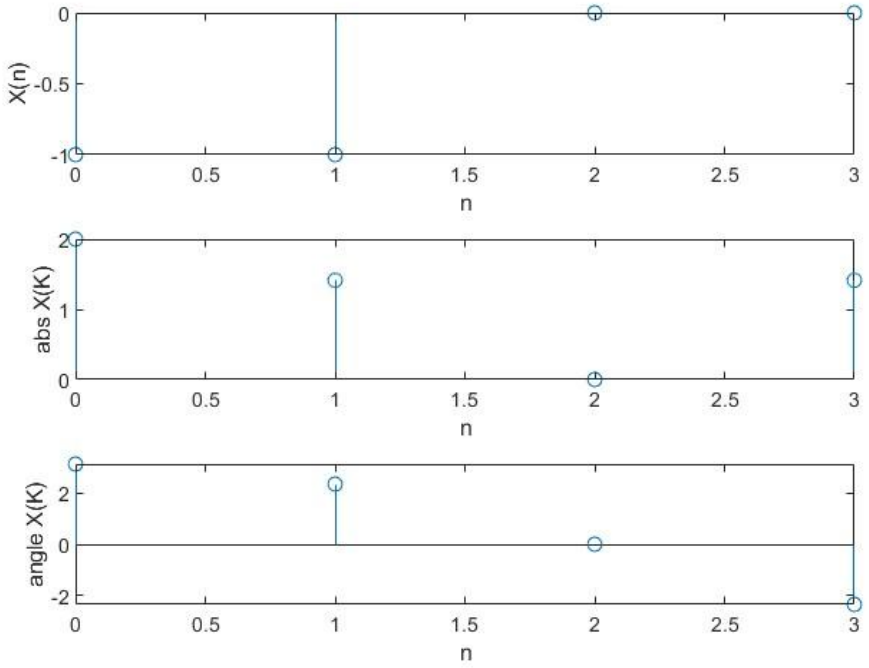
OUTPUT



Matlab Function for DIF

```
N = [0,1,2,3];
X = [-1,-1,0,0];
K = DIFFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT

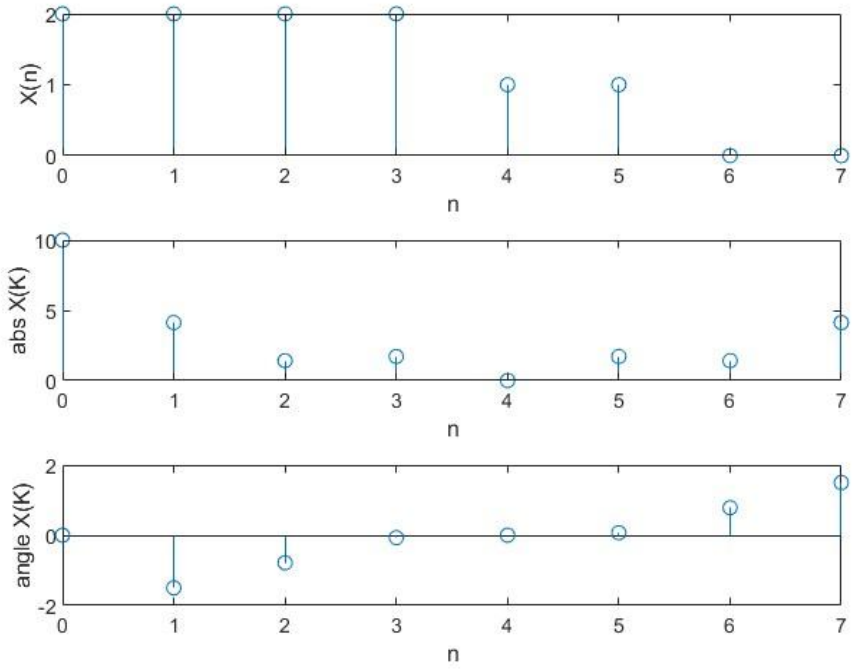


(c) x(n)={2,2,2,2,1,1}

Matlab Function for DIT

```
N = [0,1,2,3,4,5,6,7];
X = [2,2,2,2,1,1,0,0];
K = DITFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

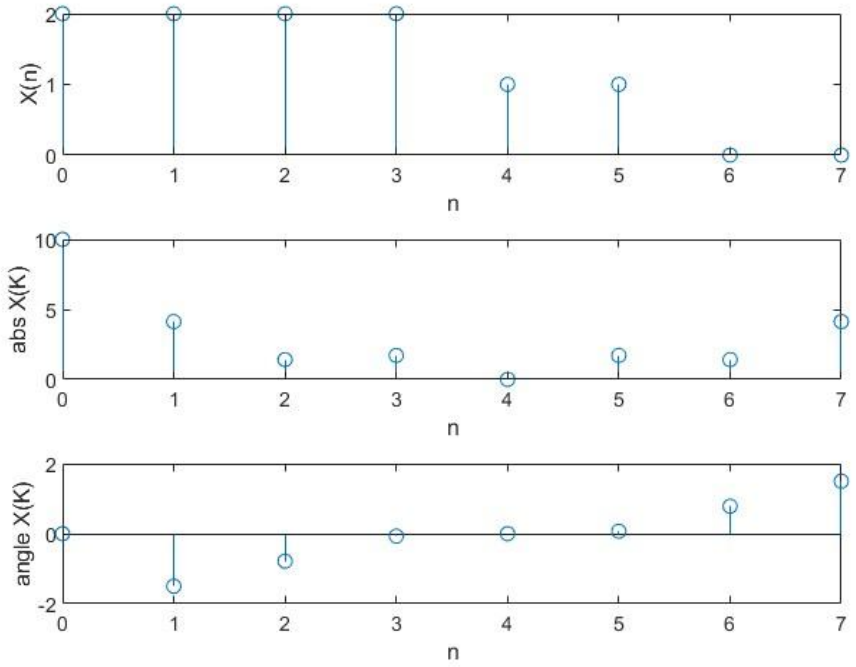
OUTPUT



Matlab Function for DIF

```
N = [0,1,2,3,4,5,6,7];
X = [2,2,2,2,1,1,0,0];
K = DIFFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT

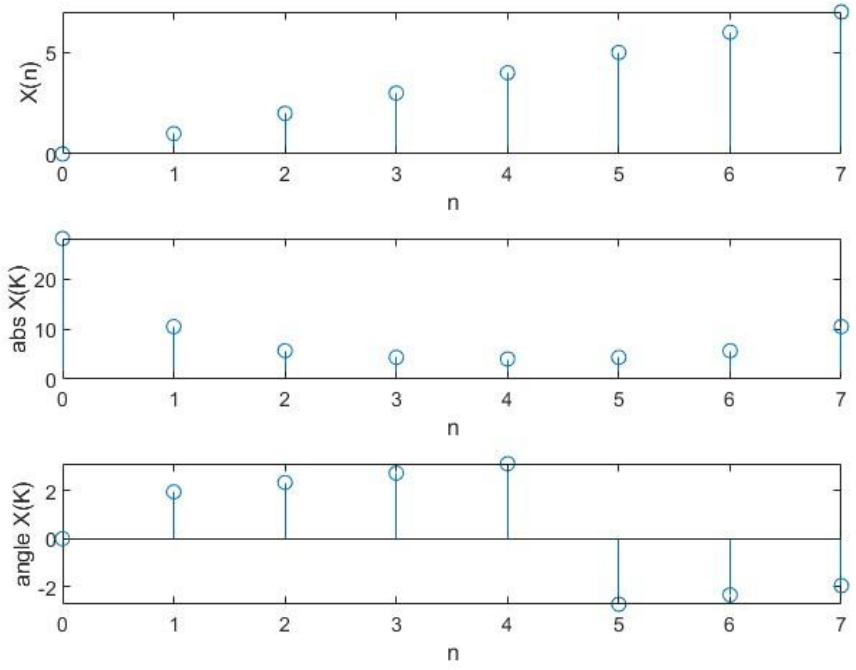


(d) x(n)=n for 0≤n≤7

Matlab Function for DIT

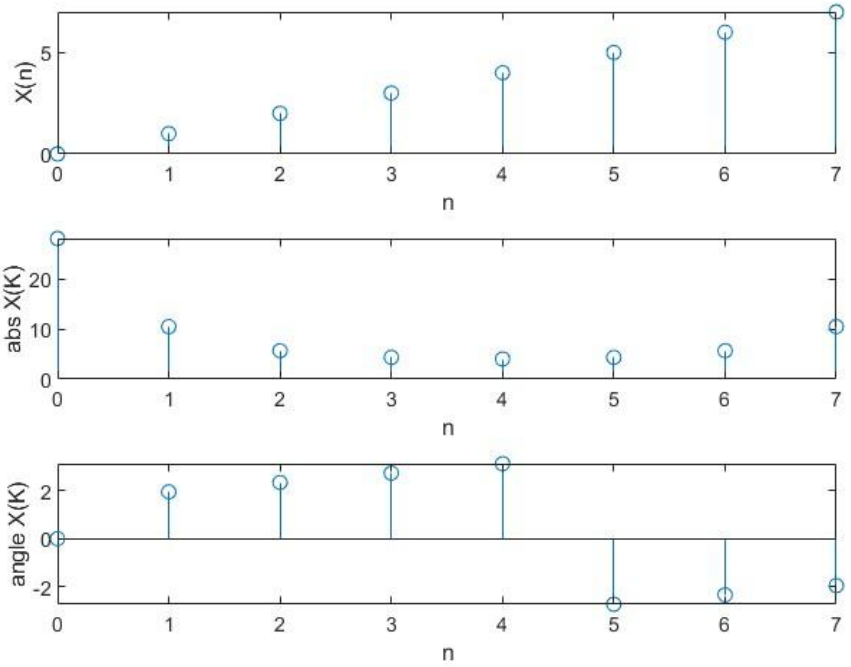
```
N = [0,1,2,3,4,5,6,7];
X = [0,1,2,3,4,5,6,7];
K = DITFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n");
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```

OUTPUT



Matlab Function for DIF

```
N = [0,1,2,3,4,5,6,7];
X = [0,1,2,3,4,5,6,7];
K = DIFFFT(X);
subplot(3,1,1);
stem(N,X);
ylabel("X(n)");
xlabel("n");
subplot(3,1,2);
stem(N,abs(K));
ylabel("abs X(K)");
xlabel("n")
subplot(3,1,3);
stem(N,angle(K));
ylabel("angle X(K)");
xlabel("n");
```


X(n)
5
0
0 1 2 3 4 5 6 7
n
abs X(K)
20
10
0
0 1 2 3 4 5 6 7
n
angle X(K)
2
0
-2
0 1 2 3 4 5 6 7
n