**PRACTICAL 5 : DIF-FFT**

|  |  |
| --- | --- |
| **U19EC046 - Harsh Suthar** | **Date: 03-09-2021** |

**AIM**

Write a MATLAB program to compute N point DIT-FFT and verify the results using inbuilt FFT command

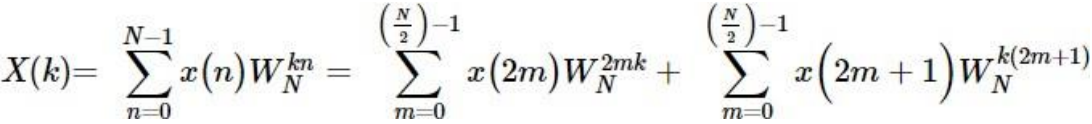
**THEORY**

Radix 2 Algorithm: Let us consider the computation of the N = 2v point DFT by the divide and conquer approach. We select M = N/2 and L = 2. This selection results in a split of the N-point data sequence into two N/2-point data sequences f1(n) and f2(n), corresponding to the even numbered and odd numbered samples of x(n), respectively, that is,

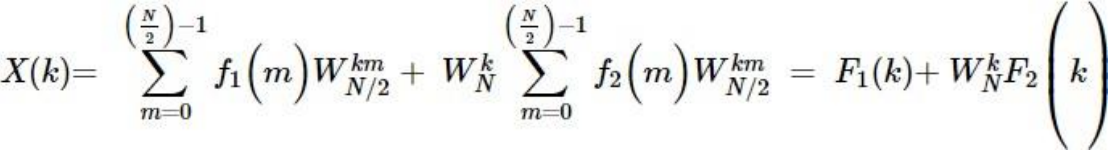
f1(n) = x(2n) f2(n) = x(2n+1), n = 0,1,…..,N – 1

Thus f1(n) and f2(n) are obtained by decimating x(n) by a factor of 2, and hence the resulting FFT algorithm is called a decimation in time (DIT) algorithm.

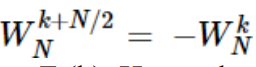
Now the N-point DFT can be expressed in terms of the DFTs of the decimated sequences as follows:



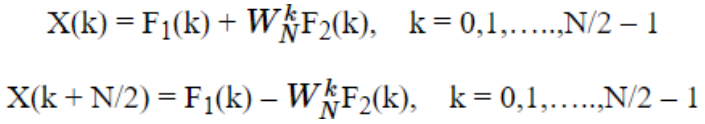




where F1(k) and F2(k) are the N/2-point DFTs of the sequences f1(m) and f2(m), respectively. Also



and F1(k+N/2) = F1(k) and F2(k+N/2) = F2(k). Hence the above expression can be expressed as



Hence, the first step results in reduction of the number of multiplications from N2 to (N2+ N)/2, which is about a factor of 2 for N large. The decimation of the data sequence can repeated again and again until the resulting sequences are reduced to one-point sequence. For N = 2v , this decimation can be performed v = log2N times. Thus the total number of complex multiplications is reduced to (N/2)log2N. The number of complex additions is Nlog2N.

**ALGORITHM**



**CODE**

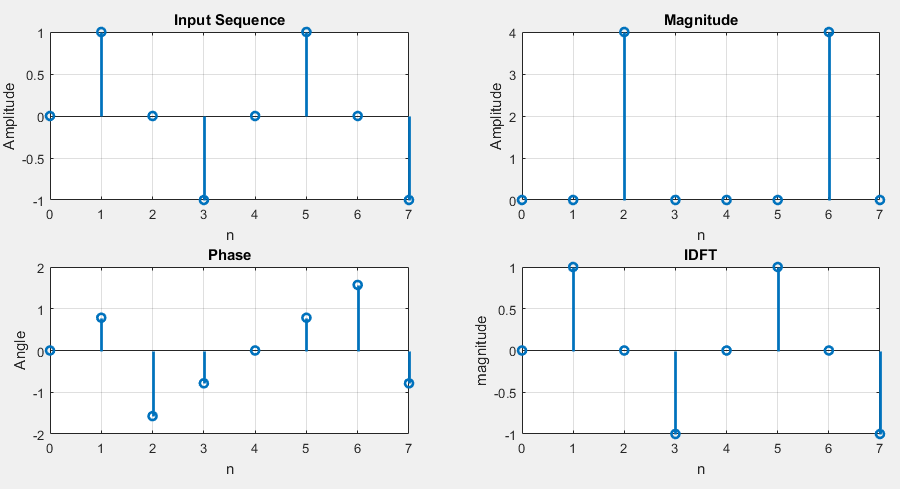
1. dif-fft without using in-built function

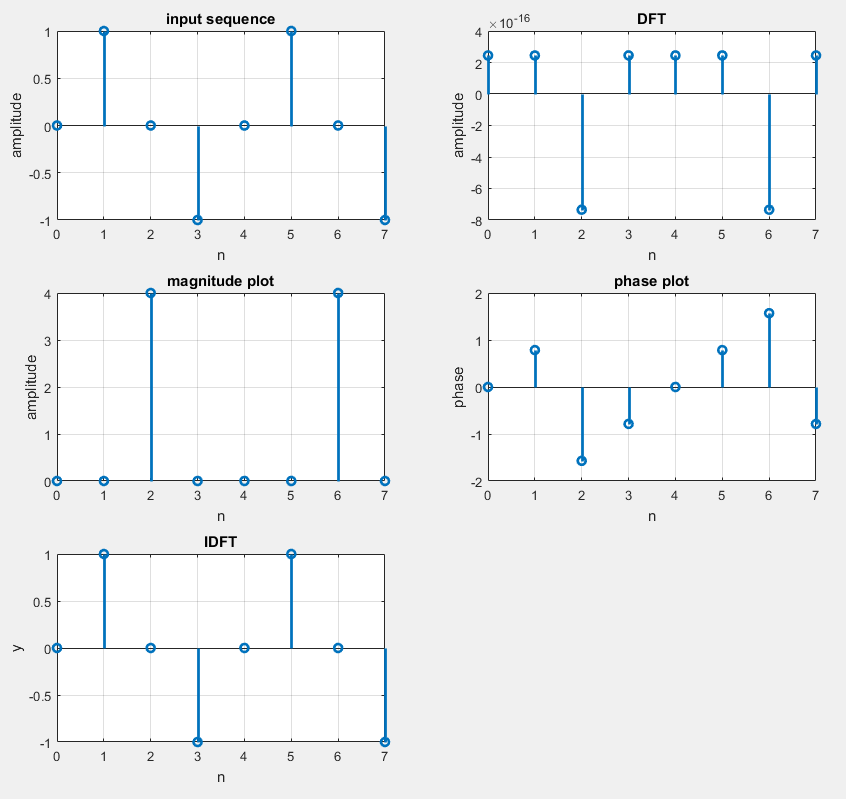
|  |
| --- |
| **clc;**  **clear all;**  **a = 0:7;**  **x = sin(pi\*a/2);**  **N = length(x);**  **l = nextpow2(N);**  **x = [x, zeros(1, (2^l)-N)];**  **N = length(x);**  **t=0:N-1;**  **subplot(3, 2, 1);**  **stem(t, x, 'linewidth', 2);**  **ylabel('Amplitude');**  **xlabel('n');**  **title('Input Sequence');**  **grid on;**  **for j = l:-1:1**  **L = 2^j;**  **for n = 1:L:N-L+1**  **for k = 0:L/2 - 1**  **w = exp(-1i\*2\*pi\*k/L); *%Twiddle Factor***  **A = x(n+k);**  **B = x(n+k+L/2);**  **x(n+k) = A+B;**  **x(n+k+L/2) = (A-B)\*w;**  **end**  **end**  **end**  **y = bitrevorder(x);**  **subplot(3, 2, 2);**  **stem(t, abs(y), 'linewidth', 2);**  **ylabel('Amplitude');**  **xlabel('n');**  **title('Magnitude');**  **grid on;**  **subplot(3, 2, 3);**  **stem(t, angle(y), 'linewidth', 2);**  **ylabel('Angle');**  **xlabel('n');**  **title('Phase');**  **grid on;**  ***% IDFT***  **for j = l:-1:1**  **L = 2^(j);**  **for n = 1:L:N-L+1**  **for k = 0:L/2 - 1**  **w = exp(1i\*2\*pi\*k/L); *%Twiddle Factor***  **C = y(n+k);**  **D = y(n+k+L/2);**  **y(n+k) = C+D;**  **y(n+k+L/2) = (C-D)\*w;**  **end**  **end**  **end**  **y=y/N;**  **y=bitrevorder(y);**  **n=0:N-1;**  **subplot(3,2,4)**  **stem(n,y, 'linewidth', 2);**  **xlabel('n');**  **ylabel('magnitude');**  **title('IDFT')**  **grid on;** |

1. dif-fft using in-built function

|  |
| --- |
| **clc;**  **clear all;**  **a=0:7;**  **xn=sin(pi\*a./2);**  **N=8;**  **L=length(xn);**  **n=0:N-1;**  **xn=[xn zeros(1,N-L)];**  **subplot(3,2,1)**  **stem(n,xn,'linewidth',2);**  **xlabel('n');**  **ylabel('amplitude');**  **title('input sequence');**  **grid;**  ***% DFT***  **Xk=fft(xn,N);**  **subplot(3,2,2)**  **stem(n,Xk,'linewidth',2);**  **xlabel('n');**  **ylabel('amplitude');**  **title('DFT');**  **grid;**  **magnitude=abs(Xk);**  **subplot(3,2,3)**  **stem(n,magnitude,'linewidth',2);**  **xlabel('n')**  **ylabel('amplitude');**  **title('magnitude plot')**  **grid;**  **phase=angle(Xk);**  **subplot(3,2,4)**  **stem(n,phase,'linewidth',2)**  **xlabel('n')**  **ylabel('phase')**  **title('phase plot')**  **grid;**  ***% IDFT***  **y=ifft(Xk,N);**  **subplot(3,2,5)**  **stem(n,y,'linewidth',2);**  **xlabel('n')**  **ylabel('y')**  **title('IDFT')**  **grid;** |

**OUTPUT**





**CONCLUSION**

In this practical, we have simulated a program to compute DFT & IDFT for a given signal in decimation in frequency algorithm using the in-built function and without using in-built function