Theory:

- Direct implementation of DFT is not efficient.
- In direct computation, total number of complex additions are N(N − 1) and complex multiplications are N^2.
- FFT is an algorithm to compute DFT in an efficient manner.
- Two common algorithms are
    1. Decimation-in-time FFT
    2. Decimation-in-frequency FFT

1. Radix-2 DIT-FFT:

- Radix 2 algorithm means N can be expressed as a power of 2, i.e., $N = 2^M$
- Input sequence is broken in to two sequence of length N/2.
    - $x_e(n)=x(2n)$; n=0,1,….,(N/2)-1 → even samples
    - $x_o(n)=x(2n+1)$; n=0,1,….,(N/2)-1 → odd samples

$$X(k) = \sum_{\substack{n=0 \\ (even)}}^{N-1} x(n)W_N^{kn} + \sum_{\substack{\mathrm{n=0} \\ \mathrm{(odd)}}}^{\mathrm{N-1}} x(n)W_N^{kn}$$

Or,

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

So,

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

Or,

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_e(n)W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_0(n)W_N^{2nk}$$

Where

$$W_N^2 = \left(e^{-j2\pi/N}\right)^2 = e^{-j2\pi/(N/2)} = W_{N/2}$$

Hence,

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_e(n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_0(n) W_{N/2}^{nk}$$

Or,

$$X(k) = X_e(k) + W_N^k X_o(k) \qquad \text{For } k \geq N/2$$

As we know that

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

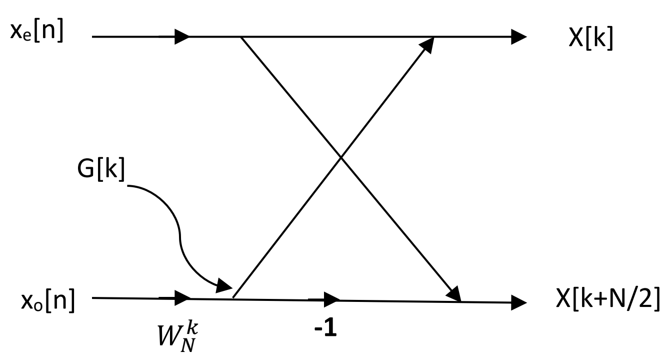
Therefore,

$$\therefore X(k) = X_e(k) - W_N^k X_o(k), \quad \text{for} \quad \mathrm{k} \geq N/2$$

So, three steps are used.

- Compute $\mathrm{N/2}$-point DFT $\mathrm{X_e[k]}$ of $\mathrm{x_e[n]}$ = even samples of $\mathrm{x[n]}$.
- Compute $\mathrm{N/2}$-point DFT $\mathrm{X_o[k]}$ of $\mathrm{x_o[n]}$ = odd samples of $\mathrm{x[n]}$.
- Compute $\mathrm{G[k]} = W_N^k X_o(k), \mathrm{k} = 0, \ldots, \mathrm{N/2} - 1$
- Combine: $\mathrm{X[k] = X_e[k] + G\,[k]}$
  $\mathrm{X[k + N/2] = X_e[k] - G\,[k]}, \quad \mathrm{k} = 0, \ldots, \mathrm{N/2} - 1$

Butterfly:

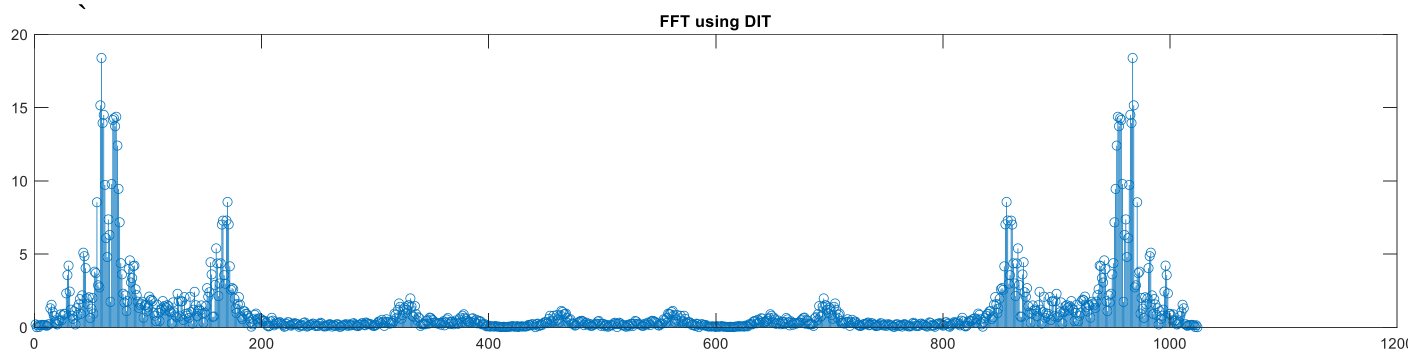


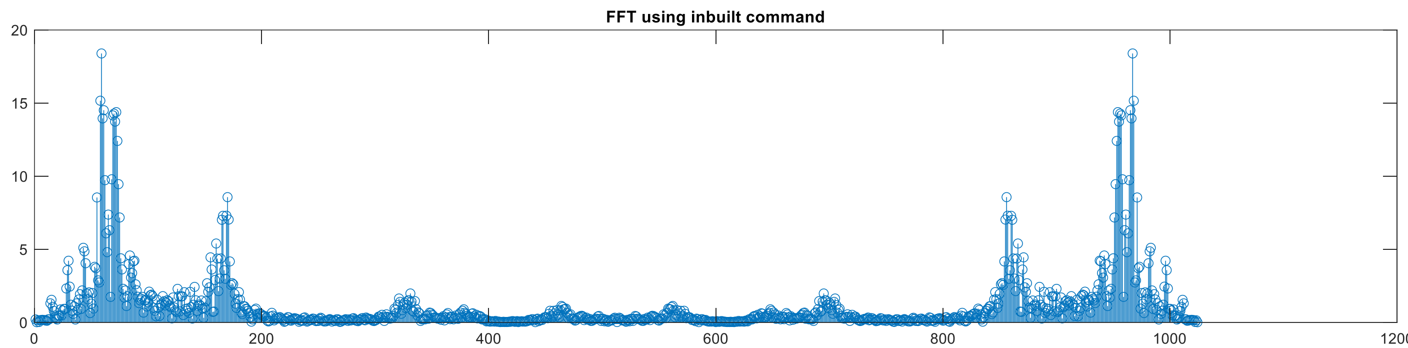
So basically key idea is: splitting into two sequences (even/odd) each of half the length, take DFT of each, then combine, saves a factor of 2 in complex multiplies.

Result:

We get following graph using while computing FFT using DIT:

`





2. Radix-2 DIF-FFT:

- Output sequence X(k) is divided into smaller and smaller subsequences.
- Input sequence x(n) partitioned into two sequences each of length N/2
    - x1(n) – first N/2 samples of x(n)
    - x2(n) – last N/2 samples of x(n)
- $x_1(n) = x(n), \mathrm{n} = 0,1,2, \dots \dots, \frac{\mathrm{N}}{2} - 1$
- $x_2(n) = x(n + \frac{\mathrm{N}}{2}), \mathrm{n} = 0,1,2, \dots \dots, \frac{\mathrm{N}}{2} - 1$
- The N - point DFT of x(n) can be written as:

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

$$= \sum_{\mathrm{n}=0}^{\frac{\mathrm{N}}{2}-1} x_1(n)\mathrm{W_N^{nk}} + \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\mathrm{W_N^{\left(n+\frac{N}{2}\right)k}}$$

$$\mathrm{X(k)} = \sum_{\mathrm{n}=0}^{\frac{N}{2}-1} x_1(n)\mathrm{W_N^{nk}} + \mathrm{W_N^{Nk/2}} \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\mathrm{W_N^{nk}}$$

$$= \sum_{\mathrm{n}=0}^{\frac{N}{2}-1} x_1(n)\mathrm{W_N^{nk}} + \mathrm{e}^{-j\pi k} \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\mathrm{W_N^{nk}}$$

- When k is even, $\mathrm{e}^{-j\pi k} = 1$.

$$\mathrm{X(2k)} = \sum_{\mathrm{n}=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)]\mathrm{W_N^{2nk}}$$

$$= \sum_{\mathrm{n}=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)]\mathrm{W_{N/2}^{nk}} \qquad \left(\because \mathrm{W_N^2} = \mathrm{W_{N/2}}\right)$$

- This is N/2-point DFT of the N/2-point sequence obtained by adding the first half and the last half of the input sequence.

$$\mathrm{X(2k)} = \sum_{\mathrm{n}=0}^{\frac{\mathrm{N}}{2}-1} f(n)\mathrm{W_N^{2nk}}$$

$$\text{where } f(n) = x_1(n) + x_2(n), \qquad \mathrm{n} = 0,1,2, \dots, \frac{\mathrm{N}}{2} - 1$$

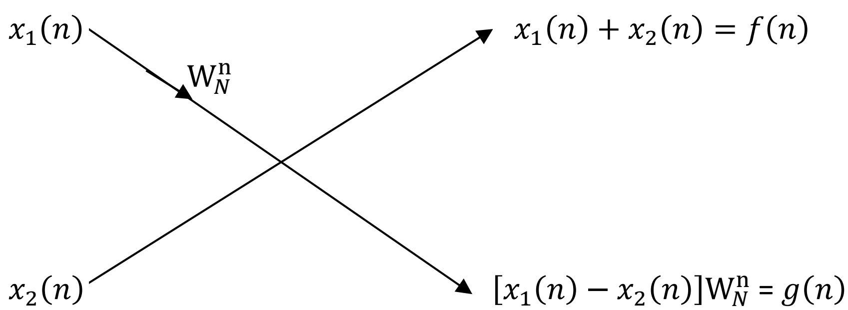
- When k is odd, $\mathrm{e}^{-j\pi k} = -1$.

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) - x_2(n)] W_N^{(2k+1)n}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) - x_2(n)] W_N^{2kn} W_N^{n}$$

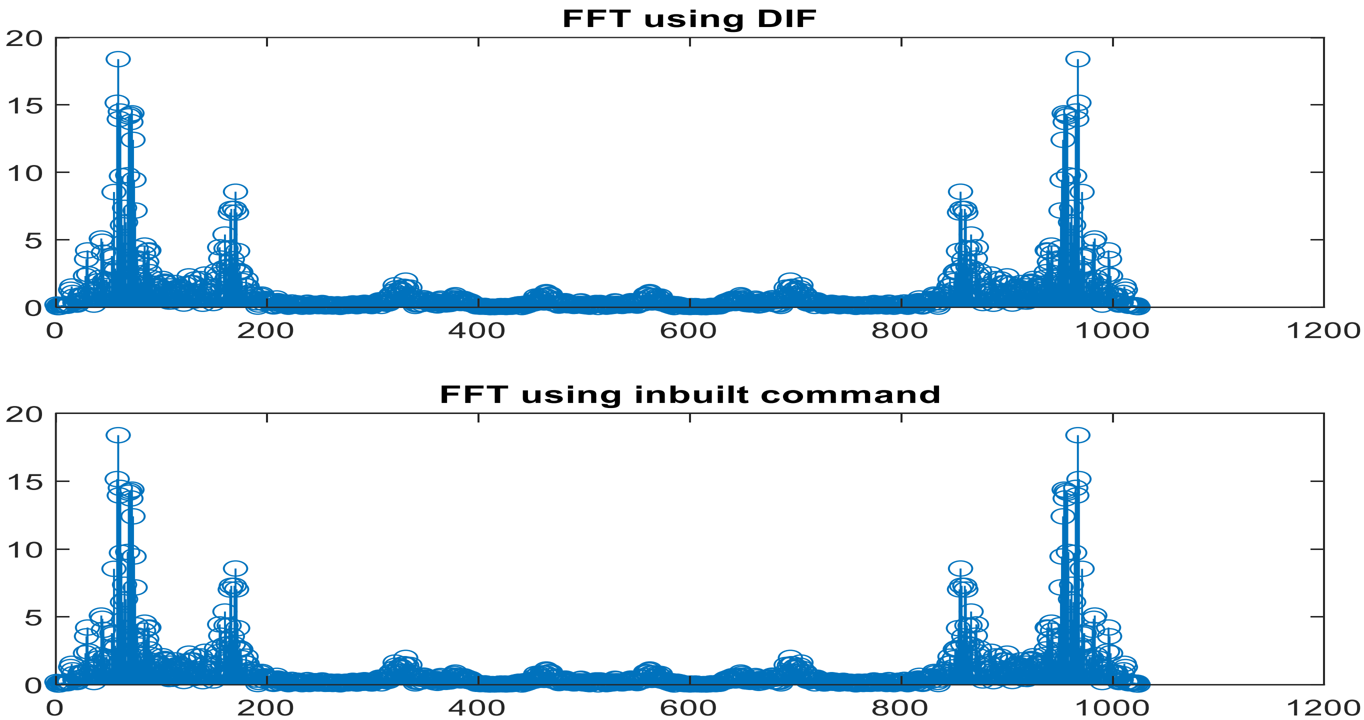
$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) - x_2(n)] W_N^{n} W_{N/2}^{nk} = \sum_{n=0}^{\frac{N}{2}-1} g(n) W_{N/2}^{nk}$$

- This is the N/2- point DFT of the sequence obtained by subtracting the second half of the input sequence from the first half and multiplying the resulting sequence by W_N^n.
- Where $g(n) = [x_1(n) - x_2(n)] W_N^n$.
- So the even and odd samples of the DFT can be obtained from the N/2-point DFTs of f(n) and g(n) respectively:
- $f(n) = x_1(n) + x_2(n), \qquad n = 0,1,2,\dots,\frac{N}{2}-1.$
- $g(n) = [x_1(n) - x_2(n)] W_N^n \qquad n = 0,1,2,\dots,\frac{N}{2}-1.$



Result:

We get following graph using while computing FFT using DIF:



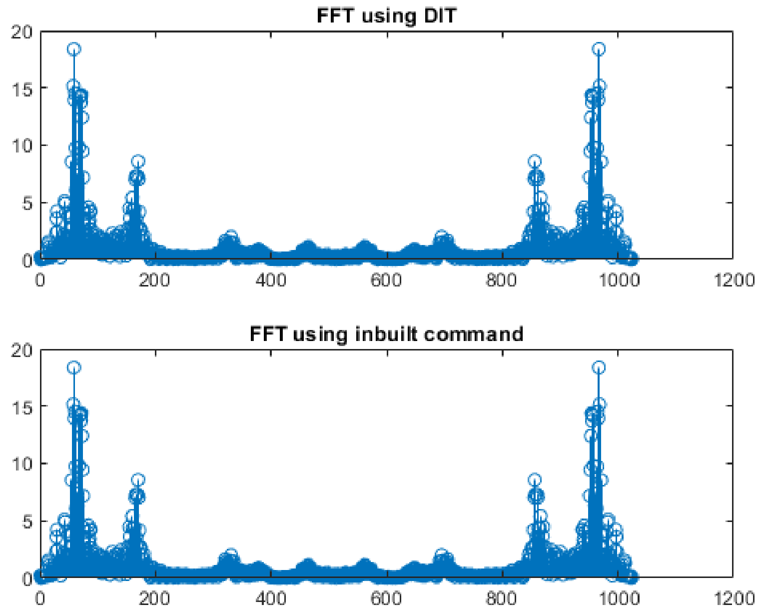
Conclusion:

- For DIT, the input is bit reversed while the output is natural order. For DIF, the input is in natural order while the output is bit reversed. 2.
- The DIF butterfly is slightly different from the DIT.
- In DIF, the complex complex multiplication takes place after the add-subtract operation.
- Both algorithms require N $\log_2$N operations to compute the DFT. Both algorithms can be done in-place and both need to perform bit reversal at some place during the computation.

APPENDIX

FFT Using DIT

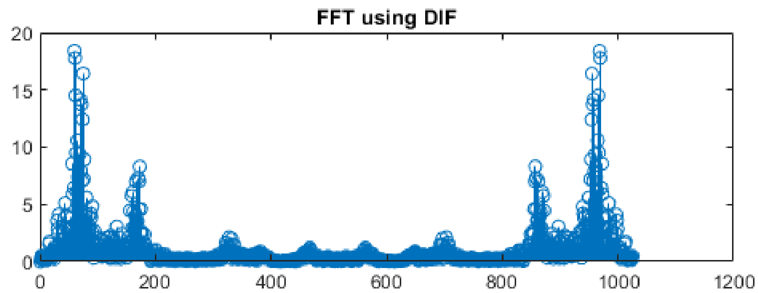
```
clear all
close all
clc
warning off
fs=1000;
[readmusic,fs]=audioread('1610958375116.wav');
N=1024;

%Taking first 1024 samples
for z = 0:1:(N-1)
    a(z+1)=readmusic(z+1);
end

%finding even and odd parts
for zz = 0:1:(N/2)-1
    a_even(zz+1) = a(2*zz+1);
    a_odd(zz+1) = a((2*zz)+2);
end

%For twiddle factor type 1 length=(N/2)
for k= 0:1:(N/2)-1
    for n= 0:1:(N/2)-1
        p=exp(-sqrt(-1)*pi*2*2*k*n/N);
        twiddle_fact_matrix(k+1, n+1)=p;
    end
end
%N/2-point DFT of even samples
fft_ev = twiddle_fact_matrix*a_even';
%N/2-point DFT of even samples
fft_odd = twiddle_fact_matrix*a_odd';

%Combine
for k=0:1:(N/2)-1;
    fft_simm(k+1) = fft_ev(k+1) + (exp(-sqrt(-1)*2*pi*k/N)*fft_odd(k+1));
    fft_simm((N/2)+k+1) = fft_ev(k+1) - (exp(-sqrt(-1)*2*pi*k/N)*fft_odd(k+1));
end

subplot(211);
stem(abs(fft_simm));
title('FFT using DIT')
fft_inbuilt=fft(a);
subplot(212);
stem(abs(fft_inbuilt));
title('FFT using inbuilt command')
```

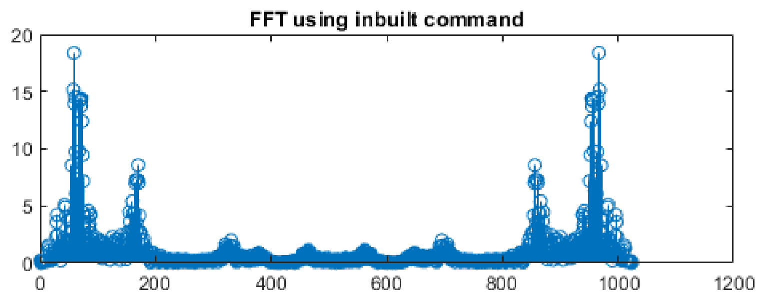


Published with MATLAB® R2020b

FFT Using DIF

```
clear all
close all
clc
warning off
fs=1000;
[readmusic,fs]=audioread('1610958375116.wav');
N=1024;

%Taking first 1024 samples
for z = 0:1:(N-1)
    a(z+1)=readmusic(z+1);
end

%Input sequence x(n) partitioned into two sequences each of length N/2
% 0 to 511 as x1[n] and 512 to 1023 as x2[n]
for zz = 0:1:(N/2)-1
    x1(zz+1) = a(zz+1);
    x2(zz+1) = a((N/2)+zz+1);
end


%For twiddle factor type 1 length=(N/2)
for k= 0:1:(N/2)-1
    for n= 0:1:(N/2)-1
        p=exp(-sqrt(-1)*pi*2*2*k*n/N);
        twiddle_fact_matrix(k+1, n+1)=p;
    end
end

f=x1+x2;
g=x1-x2;

%N/2-point even samples DFT
fft_even_2k = twiddle_fact_matrix*f';
%N/2-point odd samples DFT
fft_odd_2kplus1 = twiddle_fact_matrix*g';

%Combine
for k=1:1:(N/2)
    fft_simm(2*k) = fft_even_2k(k);
    fft_simm((2*k)+1) = fft_odd_2kplus1(k)*exp(-sqrt(-1)*2*pi*k/N);
end

subplot(211);
stem(abs(fft_simm));
title('FFT using DIF')
fft_inbuilt=fft(a);
subplot(212);
stem(abs(fft_inbuilt));
title('FFT using inbuilt command')
```





[*Published with MATLAB® R2020b*]