LECTURE NOTES

ON

DIGITAL SIGNAL PROCESSING

III B.TECH II SEMESTER (JNTUK – R 13)

FACULTY : B.V.S.RENUKA DEVI (Asst.Prof) / Dr. K. SRINIVASA RAO (Assoc. Prof)

DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING

GVP COLLEGE OF ENGINEERING FOR WOMEN

MADHURAWADA, VISAKHAPATNAM-48

UNIT III – DISCRETE FOURIER SERIES & FOURIER TRANSFORMS

Syllabus

Properties of discrete Fourier series, DFS representation of periodic sequences, Discrete Fourier transforms: Properties of DFT, linear convolution of sequences using DFT, Computation of DFT, Fast Fourier transforms (FFT) - Radix-2 decimation in time and decimation in frequency FFT Algorithms, Inverse FFT.

Discrete – Fourier Series

Fourier Series is a mathematical tool that allows the representation of any periodic signal as the sum of harmonically related complex exponential signals. The Fourier Series representation of a discrete time periodic signal involves a finite number of terms.

DFS representation of periodic sequences

A discrete time sequence x[n] is periodic if

$$x[n] = x[n + mN] \ \forall\, m$$

The smallest value of N for which this holds is called the fundamental period. The fundamental frequency is $\omega_0 = \frac{2\pi}{N}\ rad/sample.$

A periodic sequence x[n] can be represented by a Discrete Fourier Series made up of complex exponential signals of fundamental frequency $\omega_0 = \frac{2\pi}{N}$ and its harmonics.

Similar to continuous Fourier Series, the discrete time exponential Fourier Series consists of exponentials $e^{j0n}, e^{\pm j\omega_0 n}, e^{\pm j2\omega_0 n}, \ldots\ldots, e^{\pm jk\omega_0 n}, \ldots\ldots\ldots$

The discrete time complex exponentials whose frequencies are separated by 2π are identical.

$$\varphi_k(n) = e^{j\frac{2\pi}{N}kn} = e^{j\frac{2\pi}{N}kn}e^{j2\pi n} = e^{j\frac{2\pi}{N}(k+N)n} = \varphi_{k+N}(n)$$

i.e., in general, the first harmonic is identical to the $(N+1)^{st}$ harmonic, the second harmonic to $(N+2)^{nd}$ harmonic, and so on.

This implies that there are only N independent or unique harmonics whose frequencies range over 2π. So, the Discrete Fourier Series can be expressed as

$$x[n] = \sum_{k=0}^{N-1} X_k e^{jk\omega_0 n} \ \ , \omega_0 = \frac{2\pi}{N}$$

Where, the Discrete Fourier Series Coefficients X_k are given by

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\omega_0 n}$$

Since X_k are complex, they can be expressed as

$$X_k = |X_k| e^{j\angle X_k}$$

The plot of $|X_k|$ versus ω is called the **magnitude spectrum**, and the plot of $\angle X_k$ versus ω is called the **phase spectrum**.

Properties of Discrete Fourier Series

A periodic signal x[n] with DFS coefficients X_k is represented as

$$x[n] \leftrightarrow X_k$$

1. **Linearity**
   If x[n] and y[n] are two periodic signals with period N, and their corresponding DFS coefficients are X_k and Y_k .i.e.,

   $$\text{If } x[n] \leftrightarrow X_k$$
   $$\text{And } y[n] \leftrightarrow Y_k$$
   $$\text{Then, } Ax[n] + By[n] \leftrightarrow AX_k + BY_k$$

2. **Time – shifting**

   $$\text{If } x[n] \leftrightarrow X_k$$
   $$\text{then } x[n - n_0] \leftrightarrow e^{-jk\omega_0 n_0} X_k$$

3. **Frequency – shifting**

   $$\text{If } x[n] \leftrightarrow X_k$$
   $$\text{Then } e^{jn\omega_0 k_0} x[n] \leftrightarrow X_{k-k_0}$$

4. **Time – reversal**

   $$\text{If } x[n] \leftrightarrow X_k$$
   $$\text{Then } x[-n] \leftrightarrow X_{-k}$$

5. **Periodic convolution**

   $$\text{If } x[n] \leftrightarrow X_k$$

   $$\text{And } y[n] \leftrightarrow Y_k$$

   $$\text{Then, } \sum_{r=\langle N \rangle} x[r]y[n-r] \leftrightarrow NX_kY_k$$

6. **Multiplication**

   $$\text{If } x[n] \leftrightarrow X_k$$
   $$\text{And } y[n] \leftrightarrow Y_k$$

   $$\text{Then } x[n]y[n] \leftrightarrow \sum_{r=\langle N \rangle} X_r Y_{k-r}$$

7. **Complex conjugation**

$$\text{If } x[n] \leftrightarrow X_k$$
$$\text{Then } x^*[n] \leftrightarrow X^*_{-k}$$

8. **Parseval's Relation**

$$\frac{1}{N}\sum_{n=\langle N\rangle}|x[n]|^2 = \sum_{k=\langle N\rangle}|X_k|^2$$

Discrete – Fourier Transform (DFT)

The Fourier transform of a discrete – time non periodic sequence x[n] is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

If the sequence is of finite length N, then

$$X(\omega) = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}$$

Now, sampling X(ω) at equally spaced frequencies in ω .i.e., at $\omega_k = \frac{2\pi}{N}k, k = 0,1,2,\cdots,N-1$, we obtain the **Discrete – Fourier Transform**

$$X(k) = X(\omega)|_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x[n]e^{\frac{-j2\pi kn}{N}} = \sum_{n=0}^{N-1} x[n]W_N^{nk}\ , k = 0,1,\cdots,N-1$$

The **Inverse Discrete – Fourier Transform** is given by

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X(k)e^{\frac{j2\pi kn}{N}} = \frac{1}{N}\sum_{n=0}^{N-1} X(k)W_N^{-nk}, n = 0,1,\cdots,N-1$$

Where $W_N = e^{-j\frac{2\pi}{N}} = Twiddle\ factor$

Properties of DFT

1. **Periodicity**

$$\text{If } x[n] \leftrightarrow X(k)$$

then

$$X(k+mN) = X(k), m = integer$$
$$and, x[n+mN] = x[n], m = integer$$

i.e., both DFT and Inverse DFT are periodic with period N.

2. **Linearity**

$$\text{If } x_1[n] \leftrightarrow X_1(k)$$
$$\text{And } x_2[n] \leftrightarrow X_2(k)$$

$$\text{Then } Ax_1[n] + Bx_2[n] \leftrightarrow AX_1(k) + BX_2(k)$$

3. **Circular Time Shifting**

An N – point sequence x[n] is defined for $0 \le n \le N-1$, and zero for other values of n. for any integer n_0, the shifted sequence $x[n-n_0]$ is no longer defined for the range $0 \le n \le N-1$. But, by definition, DFT requires signal values in the range $0 \le n \le N-1$. To achieve this, using the periodicity property of IDFT, we consider $x_p[n]$, the periodic extension of x[n]. We delay this by n_0 samples and consider N samples between $0 \le n \le N-1$. Let this sequence be represented as $x_c[n]$. This is equivalent to moving the last n_0 samples of x[n] to the beginning of the sequence. This is called **circular shift.**

Thus, a circular shift of an N – point sequence is equivalent to a linear shift of its periodic extension.

The finite – duration circular time shifted sequence $x_c[n]$ is related to the original sequence x[n] by a modulo operation.

$$x_c[n] = x[\langle n - n_0 \rangle_N]$$

Modulo Operation: if the argument $(n - n_0)$ is between 0 and N-1, then leave it as it is; otherwise, add or subtract multiples of N from the argument $(n - n_0)$ until the result is between 0 and N – 1.

$$\text{If } x[n] \overset{N-point\ DFT}{\longleftrightarrow} X[k]$$
$$\text{Then } x[\langle n - n_0 \rangle_N] \overset{N-point\ DFT}{\longleftrightarrow} X[k]W_N^{kn_0}$$

4. **Circular frequency shifting**

$$\text{If } x[n] \overset{N-point\ DFT}{\longleftrightarrow} X[k]$$
$$\text{Then } W_N^{-nk_0}x[n] \overset{N-point\ DFT}{\longleftrightarrow} X[\langle k - k_0 \rangle_N]$$

5. **Circular time reversal**

An N – point sequence x[n] is defined for $0 \le n \le N-1$, and zero for other values of n. The time reversed sequence x[- n] is no longer defined for the range $0 \le n \le N-1$. But, by definition, DFT requires signal values in the range $0 \le n \le N-1$. To achieve this, using the periodicity property of IDFT, we consider $x_p[n]$, the periodic extension of x[n]. We time reverse (or flip) this and consider N samples between $0 \le n \le N-1$. Let this sequence be represented as $x_c[n]$. If x[n] is plotted on a circle in anti-clockwise direction, time reversal is equivalent to plotting the sequence

in clockwise direction.. This is called **circular time reversal.** $x_c[n]$ is related to the original signal x[n] by the modulo operation

$$x_c[n] = x[\langle -n \rangle_N]$$

$$\text{If } x[n] \overset{N-point\ DFT}{\longleftrightarrow} X[k]$$

$$\text{Then } x[\langle -n \rangle_N] \overset{N-point\ DFT}{\longleftrightarrow} X[\langle -k \rangle_N]$$

Circularly even sequence

An N – point sequence x[n] is called circularly even if it is symmetric about the point zero on the circle. This implies that

$$x[n] = x[\langle -n \rangle_N] = x[N-n], 1 \le n \le N-1$$

Circularly odd sequence

An N – point sequence x[n] is called circularly odd if it is antisymmetric about the point zero on the circle. This implies that

$$x[n] = -x[\langle -n \rangle_N] = -x[N-n], 1 \le n \le N-1$$

6. <u>Conjugate symmetry</u>

$$\text{If } x[n] \overset{N-point\ DFT}{\longleftrightarrow} X[k]$$

$$\text{Then } x^*[n] \overset{N-point\ DFT}{\longleftrightarrow} X^*[\langle -k \rangle_N]$$

7. <u>Circular convolution</u>

$$\text{If } x_1[n] \overset{N-point\ DFT}{\longleftrightarrow} X_1(k)$$

$$\text{And } x_2[n] \overset{N-point\ DFT}{\longleftrightarrow} X_2(k)$$

$$\text{Then} x_1[n] \otimes x_2[n] \overset{N-point\ DFT}{\longleftrightarrow} X_1(k)X_2(k)$$

<u>DFT as a Linear Transformation</u>

The Discrete Fourier Transform can be calculated using matrix notation.

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

$$\text{Where } W_N = e^{-j\frac{2\pi}{N}}$$

Expanding the above equation

$$X(0) = x[0] + x[1] + x[2] + \cdots\cdots + x[N-1]$$

$$X(1) = x[0] + x[1]W_N + x[2]W_N^2 + \cdots\cdots + x[N-1]W_N^{(N-1)}$$

$$\vdots$$

$$X(N-1) = x[0] + x[1]W_N^{(N-1)} + x[2]W_N^{2(N-1)} + \cdots\cdots + x[N-1]W_N^{(N-1)(N-1)}$$

Expressing the above set of equations in matrix notation, we obtain

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \ldots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$or, \overline{\boldsymbol{X}} = \overline{\boldsymbol{W_N}}\ \overline{\boldsymbol{x}} \ldots\ldots\ldots Eqn.A$$

Similarly, for the IDFT equation, $x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X(k)W_N^{-nk}, n = 0, 1, \cdots, N-1$, the matrix notation would be

$$\overline{\boldsymbol{x}} = \frac{1}{N}\overline{\boldsymbol{W_N}}^*\ \overline{\boldsymbol{X}} \ldots\ldots\ldots Eqn\ B$$

But, from equation A,

$$\overline{\boldsymbol{x}} = \overline{\boldsymbol{W_N}}^{-1}\ \overline{\boldsymbol{X}} \ldots\ldots.. Eqn.C$$

From Equations B & C, we have

$$\overline{\boldsymbol{W_N}}^{-1} = \frac{1}{N}\overline{\boldsymbol{W_N}}^*$$

Linear Convolution Using Circular Convolution

The output of an LTI system is the linear convolution of the input x[n] and the system's impulse response h[n]. the DFT is a practical approach for implementing linear system operations in the frequency domain. But, the problem is, the DFT operations result in a circular convolution in time domain , and not the linear convolution. i.e., $X_1(k)X_2(k) \xleftrightarrow{N-point\ IDFT} x_1[n] \otimes x_2[n]$

Let x[n] be of length N_x and h[n] be of length N_h, and let $N_x > N_h$. then, the result of linear convolution is of length $N = N_x + N_h - 1$, whereas that of cicular convolution is of length $N = \max(N_x, N_h)$.

For circular convolution to yield the same result as the linear convolution, both the sequences must be zero – padded so that both are of length $N_x + N_h - 1$. Let the padded sequences be $x_p[n]$ and $h_p[n]$. then we compute the N ($= N_x + N_h - 1$) point circular convolution of $x_p[n]$ and $h_p[n]$, resulting in a sequence equal to the linear convolution x[n]*h[n].

i.e., $X_p(k)H_p(k) \xleftrightarrow{N-point\ IDFT} x[n] * h[n]$

Filtering of Long Data sequences using DFT

To filter signals like speech signals, etc., which are long data sequences, using DFT, it is required to compute a large DFT. Moreover, the output samples are not available until all the input samples are produced , thus

introducing an unacceptably large amount of delay. To overcome this problem, the input sequence is divided into smaller sections/blocks, and each block is processed via DFT and IDFT to produce a block of output data. The output blocks are fitted together to yield the final output sequence. This procedure is called **sectioned / block convolution**. There are two ways to do this: **overlap – save method** and **overlap – add method**.

Overlap – save method

Let h[n] be the impulse response of length M, and x[n] be the input sequence of length much greater than M.

- x[n] is divided into sections $x_r[n]$, each of length N, so that each input section overlaps the preceding section by M – 1 samples.
- The first M – 1 samples of the first section are set to zero.
- Then, the N – point circular convolution, $y_r[n]$ of each section $x_r[n]$ with h[n] is obtained.
- The first M – 1 samples of each output section must be discarded.
- The remaining N – M + 1 samples from each successive sections are concatenated to reconstruct the final filtered output.

Overlap – add method

- x[n] is represented as a sum of finite – length segments of length N

$$x[n] = \sum_r x_r\,[n - rN]$$

$$\text{Where } x_r[n] = \begin{cases} x[n + rN], 0 \le n \le N - 1 \\ 0, else \end{cases}$$

- the final output of the filter is given by the linear convolution

$$y[n] = h[n] * x[n] = h[n] * \sum_r x_r\,[n - rN]$$

$$= \sum_r y_r\,[n - rN]$$

$$\text{Where } y_r[n] = h[n] * x_r[n]$$

  Since h[n] is of length M and $x_r[n]$ is of length N, the linear convolution result $y_r[n]$ is of length N + M – 1, which can be obtained using N + M – 1 point DFTs.
- Since $x_r[n]$ is the section of x[n] starting at n = rN, so $y_r[n]$ also starts at n = rN. But, each $y_r[n]$ is of length N + M – 1, whereas each $x_r[n]$ is of length N.
  i.e.,

$$y_0[n] = h[n] * x_0[n], \quad 0 \le n \le N + M - 2$$
$$y_1[n] = h[n] * x_1[n], \quad N \le n \le 2N + M - 2$$
$$y_2[n] = h[n] * x_2[n], \quad 2N \le n \le 3N + M - 2, and\ so\ on$$

So, there is an overlap of M – 1 samples between every two consecutive sections. These overlapping samples must be added to obtain final output.

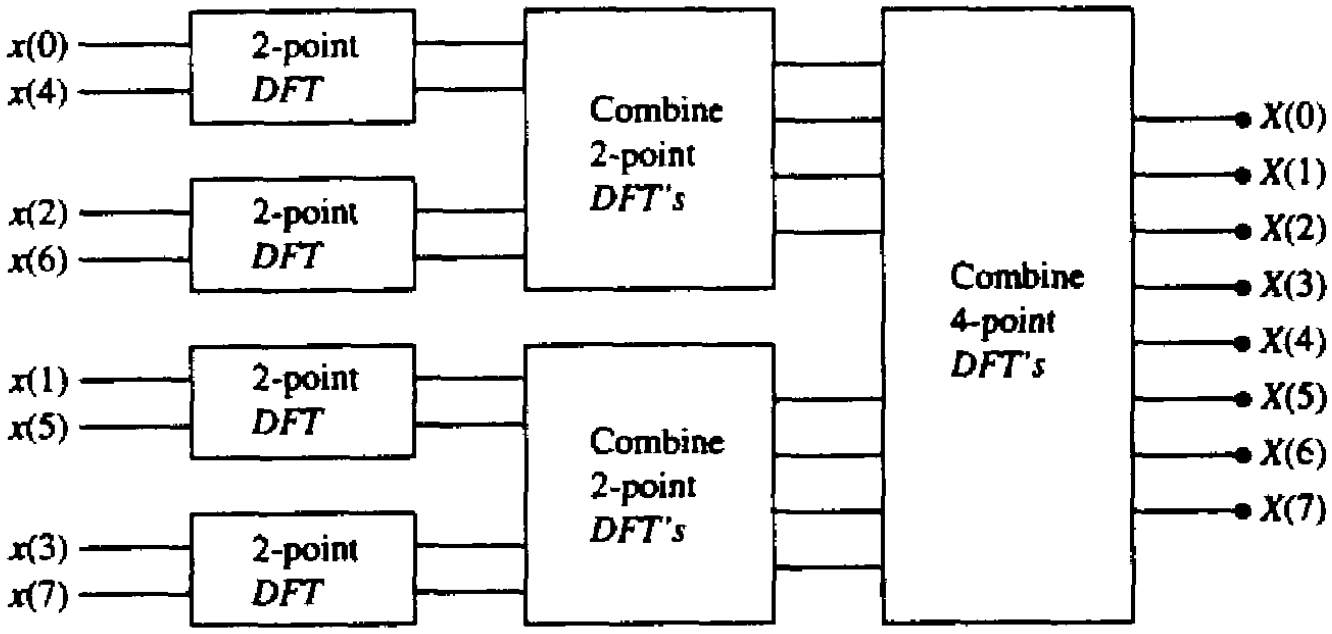
Fast Fourier Transform (FFT)

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 W_N 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 $N = r^m$. 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 $N = 2^m$, the efficient computation is achieved by breaking the N – point DFT into two $\frac{N}{2}$ - point DFTs, then breaking each $\frac{N}{2}$ - point DFT into two $\frac{N}{4}$ - 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 – Time (DIT) FFT algorithm

In this algorithm, the time – domain sequence x[n] is decimated into two $\frac{N}{2}$ – point sequences, one composed of even – indexed values of x[n], and other composed of odd – indexed values of x[n].i.e.,

$$g[n] = x[2n] \dots \dots \dots . eqn. 1$$

$$and, h[n] = x[2n+1] \dots \dots \dots . eqn. 2$$

The N – point DFT of x[n] is given by

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

This can be rewritten as

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

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

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

Using the third property of the twiddle factor, W_N, the above equation can be rewritten as

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

Using equations 1 & 2 in the above equation, we obtain

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

$$or, X(k) = G(k) + W_N^k H(k) \dots \dots eqn. 3$$

Where G(k) and H(k) are the N/2 – point DFTs of g[n] and h[n] respectively. So, G(k) and H(k) are periodic with period N/2 .i.e.,

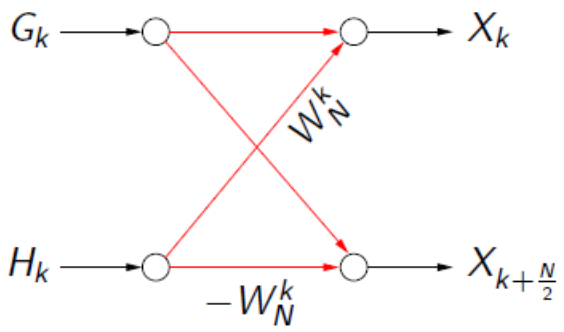
$$G\left(k + \frac{N}{2}\right) = G(k) \dots \dots eqn. 4$$

$$and, H\left(k + \frac{N}{2}\right) = H(k) \dots \dots eqn. 5$$

And using the symmetry property of the twiddle factor, W_N, and equations4 & 5

$$X(k + N/2) = G(k) - W_N^k H(k) \dots \dots eqn.6$$

Equations 3 and 6 result in the following butterfly diagram



For example, for N = 8, the DFT points in terms of G and H are

$$X(0) = G(0) + W_N^0 H(0)$$

$$X(1) = G(1) + W_N^1 H(1)$$

$$X(2) = G(2) + W_N^2 H(2)$$

$$X(3) = G(3) + W_N^3 H(3)$$

For the remaining 4 points X(4) to X(7), we use equations 4, 5, 6 and 7 to get

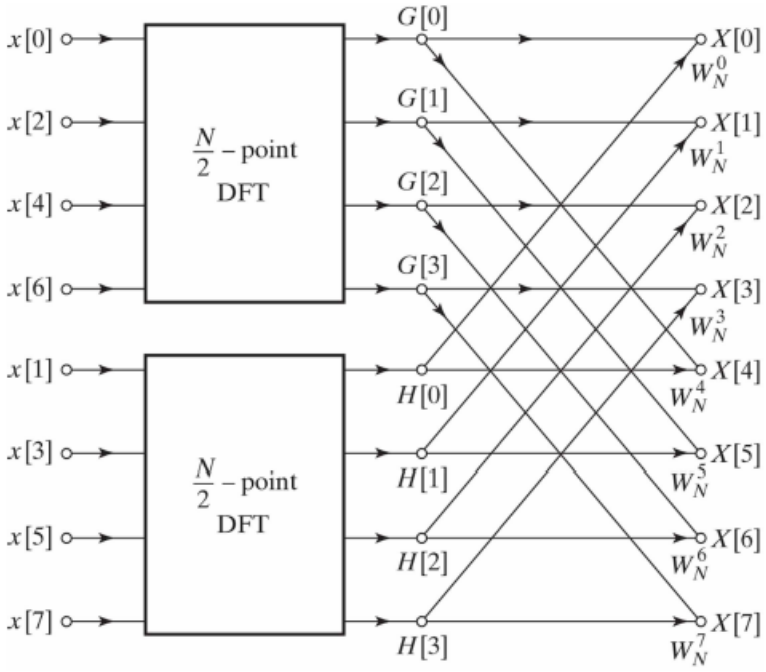
$$X(4) = G(0) - W_N^0 H(0)$$

$$X(5) = G(1) - W_N^1 H(1)$$

$$X(6) = G(2) - W_N^2 H(2)$$

$$X(7) = G(3) - W_N^3 H(3)$$

The butterfly diagram for the above set of equations is



The above process is repeated for calculating the N/2 point DFTs of g[n] and h[n], and this is continued till we get two point DFTs. Once we reach a two – point sequence, say p[n]={p[0], p[1]}, its 2 – point DFT would be

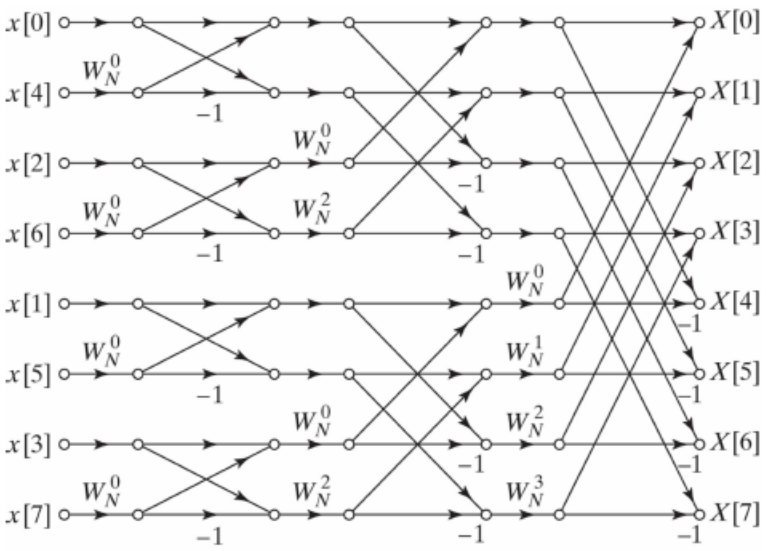
$$P(k) = \sum_{n=0}^{1} p[n] W_2^{nk} \ , k = 0,1$$

$$or, P(k) = p[0] + p[1] W_2^k, , k = 0,1$$

$$\Rightarrow P(0) = p[0] + p[1]$$

$$and\ P(1) = p[0] + W_2 p[1] = p[0] - p[1]$$

The overall butterfly diagram for DIT FFT algorithm for N =8 is



Due to repeated decimations, the input sequence is scrambled, and the order of the final input sequence is obtained as follows

Input sequence	Index	Binary form of index	Bit reversed form of index	Decimal representation	Final input sequence order
x[0]	0	000	000	0	x[0]
x[1]	1	001	100	4	x[4]
x[2]	2	010	010	2	x[2]
x[3]	3	011	110	6	x[6]
x[4]	4	100	001	1	x[1]
x[5]	5	101	101	5	x[5]
x[6]	6	110	011	3	x[3]
x[7]	7	111	111	7	x[7]

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

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

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

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

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

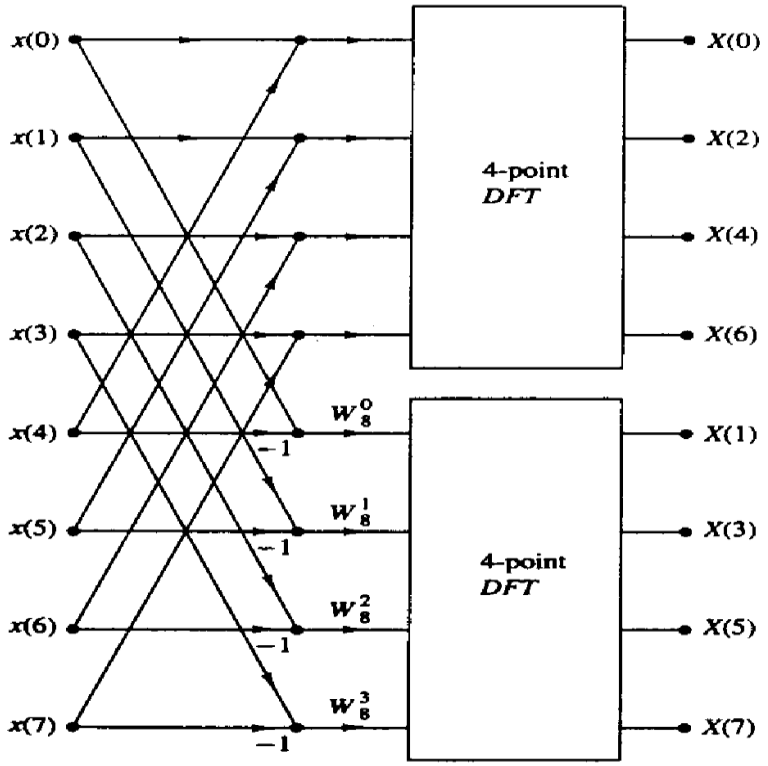
$$h[0] = \{x[0] - x[4]\} W_8^0$$

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

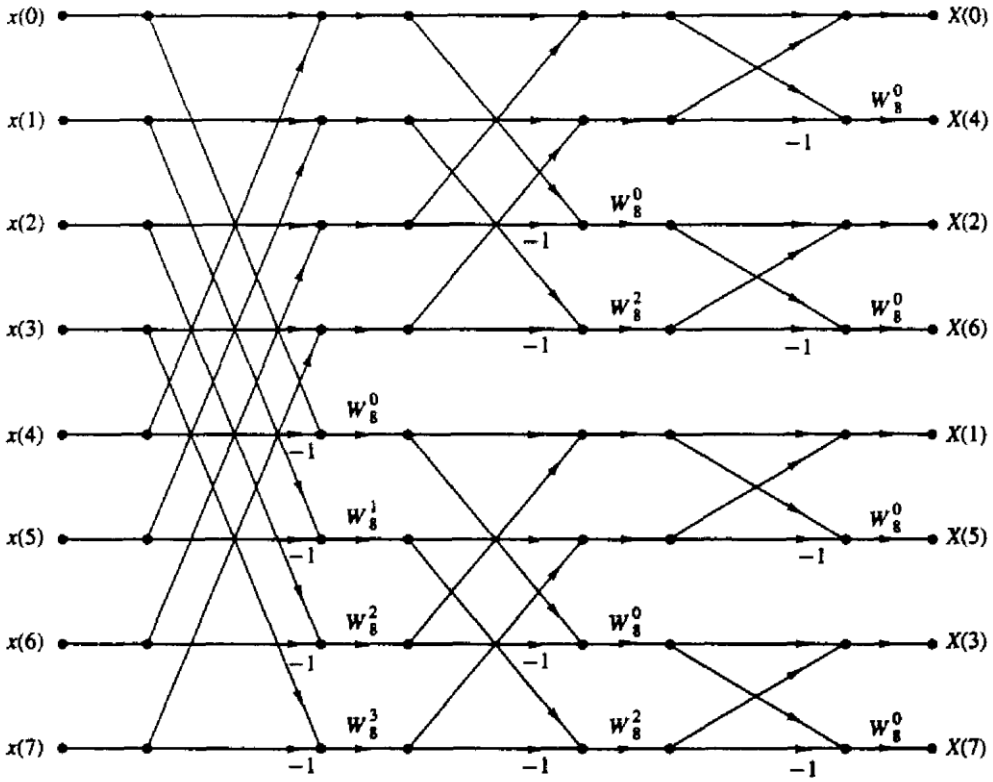
$$h[2] = \{x[2] - x[6]\} W_8^2$$

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

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.

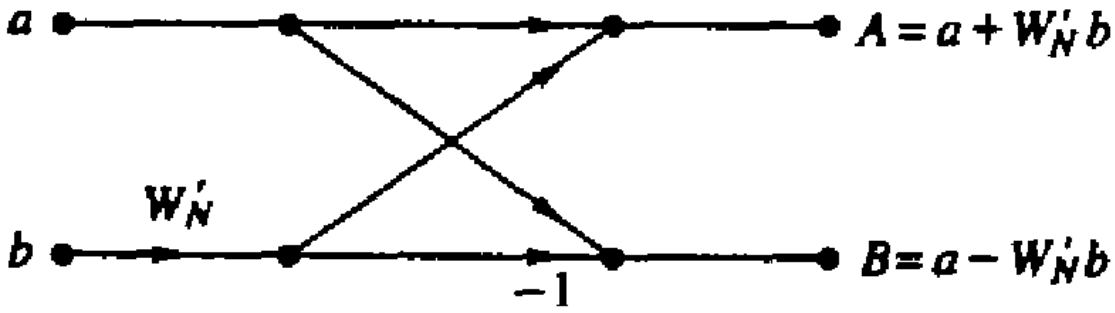
Efficiency of FFT algorithm

A direct computation of DFT requires a large number of multiplications and additions. An N – point DFT is given by

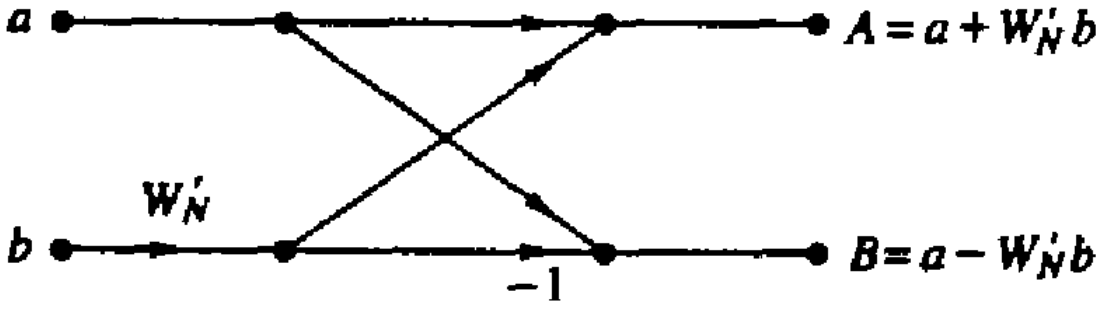
$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \ , k = 0,1, \cdots, N-1$$

In the above equation, each DFT point computation involves N complex multiplications and (N – 1) complex additions. So, the N – point DFT calculations involve N^2 complex multiplications and N(N – 1) complex additions. **This means approximately 10^6 complex multiplications and additions for a 1024 – point sequence.**

On the other hand, in an FFT algorithm, a basic butterfly in DIT – FFT algorithm is represented as



And for DIF – FFT algorithm, it is represented as



- Number of stages in a butterfly diagram =$\log_2 N$
- Number of butterflies in each stage=N/2
- Number of complex multiplications in each butterfly=1
- Number of complex additions in each butterfly=2.

From the above, for an N – point FFT algorithm,

- Total number of complex multiplications $=\frac{N}{2}\log_2 N$
- Total number of complex additions = $N\log_2 N$

For a 1024 – point sequence, this means approximately 5120 complex multiplications and 10240 complex additions. i.e., approximately 100 times less additions and 200 times less multiplications than direct computation of DFT.

As the number of input samples increase, the savings in the number of computations also increase.

In – Place computations

Another advantage of FFT algorithm is In – Place computations. As shown in the previous diagram of a basic butterfly, any butterfly calculation involves 2 complex inputs a and b, and 2 complex outputs A and B. once, these two outputs are calculated, the inputs of the butterfly are not used in any other calculation. So, A and B can be stored in the same memory locations as a and b. this is called **In – Place computation.**

Inverse FFT Algorithm

Let X(k) be the N – point DFT of a length – N sequence x[n]. The inverse DFT is given by

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

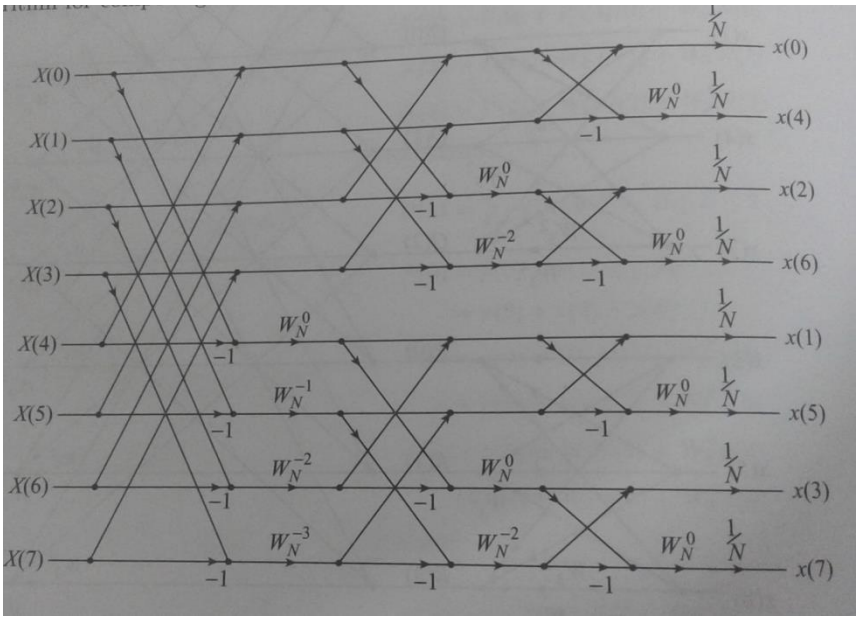
$$or, \qquad x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X(k)(W_N^{nk})^*, n = 0, 1, \cdots, N-1$$

An FFT algorithm can be used to compute the inverse DFT by replacing x[n] by X(k), taking the negative powers of W_N, and dividing the output by N. Hence, in order to compute the inverse DFT from an FFT algorithm, following steps can be followed

- Take X(k) as the input sequence and x[n] as the output sequence
- Compute FFT by replacing the twiddle factors W_N by W_N^{-1}
- Divide the output sequence by N

So, the DIT – FFT algorithm becomes DIF Inverse FFT algorithm and vice – versa.

For example the DIT – Inverse FFT butterfly diagram for N =8 would be as follows


X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
1/N
W_N^0
W_N^{-1}
W_N^{-2}
W_N^{-3}
−1