Hızlı Fourier Dönüşümü (FFT)

Ayrık Fourier dönüşümünü tekrar hatırlayacak olursak,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad 0 \le k \le N-1 \tag{3.29}$$

Burada $W_N = e^{-j(2\pi/N)}$'dir. (3.29) denklemiyle gösterilen Ayrık Fourier Dönüşümünün (DFT) doğrudan hesaplanmasında her bir $X(k)$ değeri için N adet karmaşık çarpma ve $N-1$ adet karmaşık toplama işlemi yapılmaktadır. Bundan dolayı N adet DFT değeri bulunurken, N^2 adet çarpma ve $N(N-1)$ adet toplama işlemi gereklidir. Ayrıca her karmaşık çarpma işlemi dört gerçel çarpma ve iki gerçel toplama işlemi ve her bir karmaşık toplama iki gerçel toplama işlemi ile gerçeklenmektedir. Sonuç olarak, dizi uzunluğu olan N'nin büyük olması durumunda DFT'nin doğrudan bulunması çok fazla miktarda işlem yapılmasını gerektirir. Yani, N sayısı artarken yapılan işlem sayısı yüksek hızla artmakta ve işlem sayısı kabul edilemez bir seviyeye doğru gitmektedir. 1965 yılında Cooley ve Tukey Ayrık Fourier Dönüşümü için gerekli işlem miktarını azaltacak bir prosedür geliştirdiler[1]. Bu prosedür, sayısal işaret işleme ve diğer alanlarda DFT uygulamalarında ani bir artış olmasına sebep oldu. Ayrıca başka algoritmaların geliştirilmesine ön ayak olmuştur. Tüm bu algoritmalar **Hızlı Fourier Dönüşüm (FFT)** algoritmaları olarak bilinir. Bu algoritmalar ile DFT hesabı için yapılması gereken işlem sayısı büyük ölçüde azaltılarak işlem kolaylığı sağlanmıştır. Her ne kadar dönüşüm olarak adlandırılsa, Hızlı Fourier Dönüşümü (FFT) Ayrık Fourier Dönüşümü (DFT)'den farklı değildir. FFT, DFT hesaplanması için etkili ve ekonomik bir algoritmadır.

Verimli Hesaplama :

Verimli bir şekilde tasarlanmış algoritmada işlem sayısı veri örneği başına sabit olmalıdır ve bundan dolayı toplam işlem sayısı N'e bağlı olarak lineer olmalıdır. Genel olarak bir toplama işlemi için gereken işleme (processing) zamanı bir çarpma işlemi için gereken işleme zamanına göre çok daha azdır. Bu yüzden daha çok 4

[1] Cooley, J.W., Tukey, J.W., *An Algorithm for the Machine Computation of Complex Fourier Series*, Mathematics of Computation, Vol. 19, pp. 297-301, April 1965.

CamScanner ile tarandı

gerçel çarpma ve 2 gerçel toplama gerektiren komples çarpma işlemleri sayısını dikkate alacağız.

$\{W_N^{kn}\}$ faktörünün periyodiklik ve simetri özellikleri kullanılarak işlem sayısı azaltılabilir.

- Periyodiklik Özelliği

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- Simetri Özelliği

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

Periyodiklik ve simetri özelliklerinin işlem sayısını azaltmadaki faydasını bir örnekle görelim.

Örnek 3.3:

Dört noktalı bir DFT'nin hesaplanması ve hesaplanması için verimli bir algoritma geliştirilmesi,

$$X(k) = \sum_{n=0}^{3} x(n) W_4^{nk} \quad , \quad 0 \le n \le 3, \quad W_4 = e^{-j2\pi/4} = -j$$

Çözüm:

Yukarıdaki hesaplama 16 tane kompleks çarpma işlemi içeren matris formunda gösterilebilir:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

Verimli Yaklaşım: Periyodikliği kullanarak,

$$W_4^0 = W_4^4 = 1 \quad ; \quad W_4^1 = W_4^9 = -j$$
$$W_4^2 = W_4^6 = -1 \; ; \quad W_4^3 = j$$

bu eşitlikleri yukarıdaki matris formunda yerine koyarsak aşağıdaki ifadeyi elde ederiz.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

CamScanner ile tarandı

Buradan aşağıdaki denklemleri elde edebiliriz:

$$X(0) = x(0) + x(1) + x(2) + x(3) = \underbrace{[x(0) + x(2)]}_{g_1} + \underbrace{[x(1) + x(3)]}_{g_2}$$

$$X(1) = x(0) - jx(1) - x(2) + jx(3) = \underbrace{[x(0) - x(2)]}_{h_1} - j\underbrace{[x(1) - x(3)]}_{h_2}$$

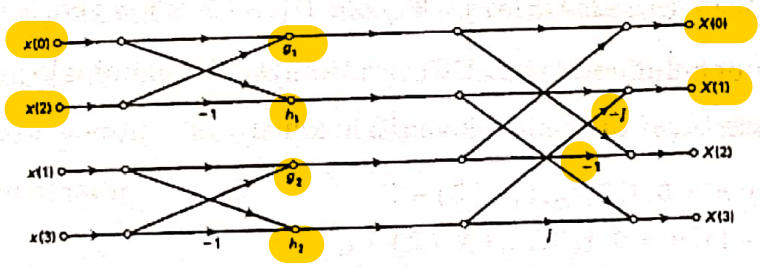
$$X(2) = x(0) - x(1) + x(2) - x(3) = \underbrace{[x(0) + x(2)]}_{g_1} - \underbrace{[x(1) + x(3)]}_{g_2}$$

$$X(3) = x(0) + jx(1) - x(2) - jx(3) = \underbrace{[x(0) - x(2)]}_{h_1} + j\underbrace{[x(1) - x(3)]}_{h_2}$$

Dolayısıyla verimli bir algoritma,

Basamak 1	*Basamak* 2
$g_1 = x(0) + x(2)$	$X(0) = g_1 + g_2$
$g_2 = x(1) + x(3)$	$X(1) = h_1 + jh_2$
$h_1 = x(0) - x(2)$	$X(2) = g_1 - g_2$
$h_2 = x(1) - x(3)$	$X(3) = h_1 + jh_2$

Bu algoritmaya göre yapılacak işlem sadece 2 kompleks çarpımayı içerir. Bu algoritmanın yapısı Şekil 3.5'deki gibidir.



Şekil 3.5: Örnek 3.3 için tasarlanmış algoritma yapısı

Hızlı Fourier Dönüşümü (FFT) Algoritmaları

DFT'nin hesaplanmasında birçok farklı algoritma geliştirilmiştir. Bu algoritmaların işlem sayısını azaltmak için temelde kullandığı özellikler temel fonksiyon olan W_N'in periyodiklik ve simetri özelliğidir. Burada *Parçala-Birleştir yaklaşımını* (divide-combine approach) kullanarak türetilen radix-R FFT algoritmalarından radix-2 (yani data boyu $N = 2^k$ olan veriler için) algoritması olan Zamanda Desimasyonlu FFT (decimation-in-time FFT) ve Frekansda Desimasyonlu FFT (decimation-in-frequency FFT) algoritmalarını inceleyeceğiz.

CamScanner ile tarandı

Zamanda Desimasyonlu FFT

İşaretin uzunluğu olan N sayısı FFT algoritmalarında önemli rol oynamaktadır. N'nin ikinin katları olması durumunda algoritma basit ve etkin olmaktadır. $N = 2^R$ olmakla birlikte bu algoritmaya Radix-2 FFT algoritması denir. N çift tamsayı olup $x(n)$ dizisi $N/2$ uzunluklu iki diziye ayrılabilir. Bu ayrımda, ilk dizinin elemanları tek sayı indisli, ikinci dizinin elemanları çift sayı indisli seçilir ve $n = 2r$ çift indisliler için, $n = 2r+1$ tek indisliler için kullanılır . Bu ayrım aşağıda gösterilmiştir;

$$X_k = \sum_{n\ çift} x(n)W_N^{nk} + \sum_{n\ tek} x(n)W_N^{nk}$$

$$= \sum_{r=0}^{(N/2)-1} x(2r)W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x(2r+1)W_N^{(2r+1)k} \qquad (3.30)$$

$$k = 0,1,2.....N-1$$

$W_N^2 = W_{N/2}$ ifadesi kullanılarak aşağıdaki şekle getirilir;

$$X_k = \sum_{r=0}^{(N/2)-1} x(2r)W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x(2r+1)W_{N/2}^{rk} \qquad (3.31)$$

$$= AFD[N/2\ çift\ noktalar] + W_N^k\, AFD[N/2\ tek\ noktalar]$$

Burada $N/2$ uzunluğunda iki DFT yardımıyla N uzunluklu dizinin DFT'si hesaplanmaktadır. $x(n)$ dizisinin tek ve çift noktaları;

$$x^c(n) = x(2n); n = 0,1,2,...,(N/2)-1$$
$$x^t(n) = x(2n+1); n = 0,1,2,...,(N/2)-1 \qquad (3.32)$$

şeklinde gösterelim. (3.32)'deki tanımlar (3.31)'de yerine konularak

$$X_k = \sum_{n=0}^{(N/2)-1} x^c(n)W_{N/2}^{nk} + W_N^k \sum_{n=0}^{(N/2)-1} x^t(n)W_{N/2}^{nk} \qquad (3.33)$$

bulunur. Bu bağlantıda $X_{k+(N/2)}$ hesaplanırsa

$$X_{k+(N/2)} = \sum_{n=0}^{(N/2)-1} x^c(n)W_{N/2}^{n(k+(N/2))} + W_N^{(k+(N/2))} \sum_{n=0}^{(N/2)-1} x^t(n)W_{N/2}^{n(k+(N/2))} \qquad (3.34)$$

ifade edilir.

$$W_N^{nN/2} = \left[e^{-j2\pi/(N/2)}\right]^{(nN/2)} = e^{-j2\pi n} = 1$$

$$W_{N/2}^{N/2} = 1$$

ifadeleri kullanılarak,

CamScanner ile tarandı

$$X_{k+(N/2)} = \sum_{n=0}^{(N/2)-1} x^{e}(n)W_{N/2}^{nk} - W_N^k \sum_{n=0}^{(N/2)-1} x^{o}(n)W_{N/2}^{nk} \qquad (3.35)$$

elde edilir. k indeksini $0 \le k \le (N/2)-1$ aralığında sınırlayarak çift ve tek noktaların DFT'leri ayrı ayrı belirtilebilir:

$$X_k^{e} = \sum_{n=0}^{(N/2)-1} x^{e}(n)W_{N/2}^{nk}$$
$$X_k^{o} = \sum_{n=0}^{(N/2)-1} x^{o}(n)W_{N/2}^{nk} \qquad (3.36)$$

Bu tanımlardan yararlanarak (3.33) ve (3.35) bağıntıları tekrar yazılabilir:

$$X_k = X_k^{e} + W_N^k X_k^{o}$$
$$X_{k+(N/2)} = X_k^{e} - W_N^k X_k^{o}$$
$$k = 0,1,2,\ldots,(N/2)-1$$

N-noktalı DFT'nin tek ve çift noktalarının oluşturduğu $N/2$ noktalı iki DFT'den elde edilebileceği görülmektedir. Aynı şekilde, $N/2$ noktalı iki DFT de yeniden belirlenecek $N/4$ noktalı tek ve çift noktalı dizilerden benzer biçimde elde edilir. $N = 2^L$ varsayıldığında L adım gidilecek olursa, sonuçta sadece 2 noktalı bir dizinin DFT'sinin hesabı yeterli olmaktadır. Takip eden örnekte bu durum ayrıntılı bir şekilde açıklanmaktadır.

Örnek 3.4:

Zamanda desimasyonlu FFT algoritmasını açıklayabilmek amacıyla sekiz noktalı bir dönüşümü ele alalım. Buna göre, $N = 8$ ve $N = 2^L$'den $L = 3$ olmaktadır. Şekil 3.6, sekiz noktalı DFT'nin iki adet dört noktalı DFT'ye ayrıştırılmasını göstermektedir. Bu ayrıştırma işlemine devam ederek dört noktalı DFT'ler de iki noktalı DFT'lere Şekil 3.7'de gösterildiği gibi ayrıştırılabilir. En son aşamada $N/2$ adet iki noktalı DFT elde edilecektir. DFT tanımından, bu işlem şu şekilde basitleşecektir

$$X(0) = x(0) + x(1)$$
$$X(1) = x(0) - x(1)$$

Şekil 3.8, sekiz noktalı zamanda desimasyonlu FFT'nin akış diagramını göstermektedir. Bu akış diagramı üzerinden üç adım ile (bkz. Şekil 3.9) $x(n), 0 \le n \le 7$ dizisinin FFT'si şu şekilde hesaplanabilir:

CamScanner ile tarandı

Adım 1

$$x'(0) = x(0) + W_N^0 x(4) = x(0) + x(4)$$
$$x'(4) = x(0) + W_N^4 x(4) = x(0) - x(4)$$
$$x'(2) = x(2) + W_N^0 x(6) = x(2) + x(6)$$
$$x'(6) = x(2) + W_N^4 x(6) = x(2) - x(6)$$
$$x'(1) = x(1) + W_N^0 x(5) = x(1) + x(5)$$
$$x'(5) = x(1) + W_N^4 x(5) = x(1) - x(5)$$
$$x'(3) = x(3) + W_N^0 x(7) = x(3) + x(7)$$
$$x'(7) = x(3) + W_N^4 x(7) = x(3) - x(7)$$

Adım 2

$$x''(0) = x'(0) + W_N^0 x'(2) = x'(0) + x'(2) = x(0) + x(4) + x(2) + x(6)$$
$$x''(4) = x'(4) + W_N^2 x'(6) = x(0) - x(4) + W_N^2 \{x(2) + x(6)\}$$
$$x''(2) = x'(0) + W_N^4 x'(2) = x'(0) - x'(2) = x(0) + x(4) - \{x(2) + x(6)\}$$
$$x''(6) = x'(4) + W_N^6 x'(6) = x(0) - x(4) + W_N^6 \{x(2) + x(6)\}$$
$$x''(1) = x'(1) + W_N^0 x'(3) = x'(1) + x'(3) = x(1) + x(5) + x(3) + x(7)$$
$$x''(5) = x'(5) + W_N^2 x'(7) = x(1) - x(5) + W_N^2 \{x(3) - x(7)\}$$
$$x''(3) = x'(1) + W_N^4 x'(3) = x'(1) - x'(3) = x(1) + x(5) - \{x(3) + x(7)\}$$
$$x''(7) = x'(5) + W_N^6 x'(7) = x(1) - x(5) + W_N^6 \{x(3) - x(7)\}$$

Adım 3

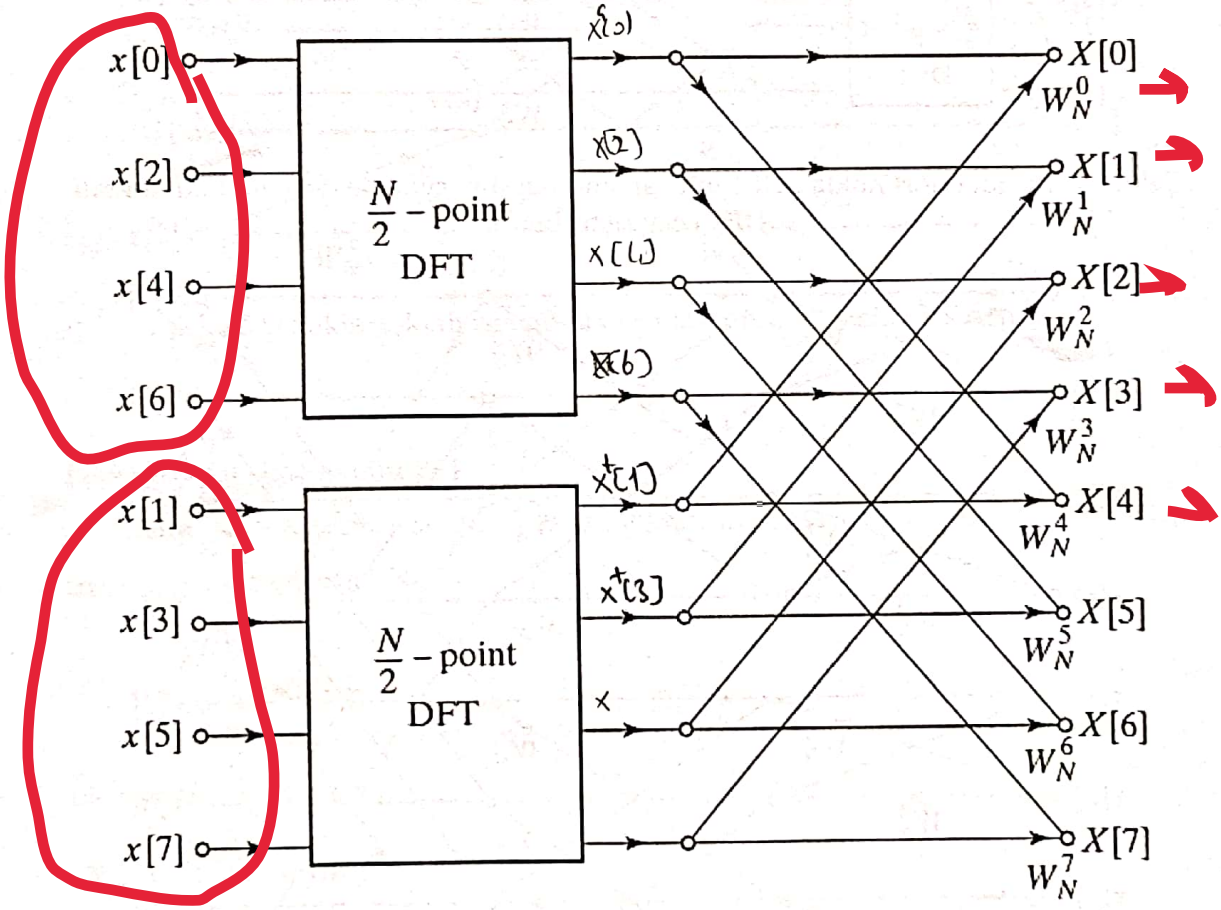
$$X(0) = x''(0) + W_N^0 x''(1) = x''(0) + x''(1)$$
$$X(1) = x''(4) + W_N^1 x''(5)$$
$$X(2) = x''(2) + W_N^2 x''(3)$$
$$X(3) = x''(6) + W_N^3 x''(7)$$
$$X(4) = x''(0) + W_N^4 x''(1) = x''(0) - x''(1)$$
$$X(5) = x''(4) + W_N^5 x''(5)$$
$$X(6) = x''(2) + W_N^6 x''(3)$$
$$X(7) = x''(6) + W_N^7 x''(7)$$

Son olarak 2. adımda bulunan değerlere 3. adımda yerlerine konulursa,

CamScanner ile tarandı

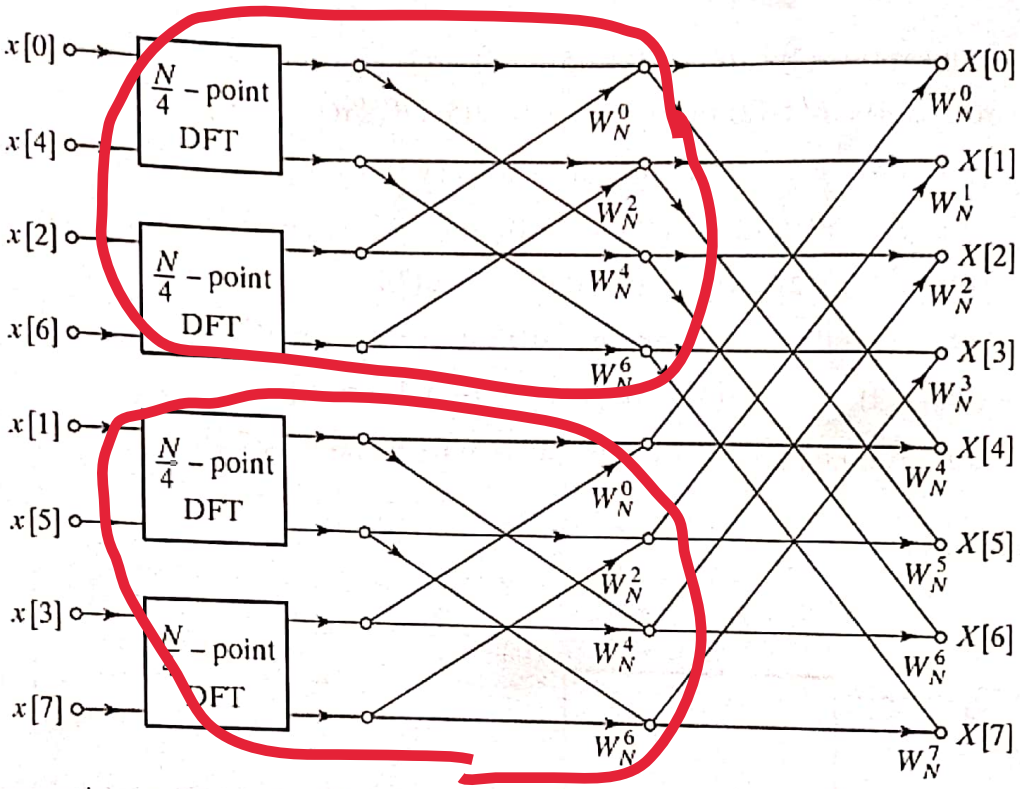
$$X(0) = x(0) + x(4) + x(2) + x(6) + x(1) + x(5) + x(3) + x(7)$$
$$X(1) = x(0) - x(4) + W_N^2\{x(2) + x(6)\} + W_N^1\{x(1) - x(5) + W_N^2\{x(3) - x(7)\}\}$$
$$X(2) = x(0) + x(4) - \{x(2) + x(6)\} + W_N^2\{x(1) + x(5) - \{x(3) + x(7)\}\}$$
$$X(3) = x(0) - x(4) + W_N^6\{x(2) + x(6)\} + W_N^3\{x(1) - x(5) + W_N^6\{x(3) - x(7)\}\}$$
$$X(4) = x(0) + x(4) + x(2) + x(6) - \{x(1) + x(5) + x(3) + x(7)\}$$
$$X(5) = x(0) - x(4) + W_N^2\{x(2) + x(6)\} + W_N^5\{x(1) - x(5) + W_N^2\{x(3) - x(7)\}\}$$
$$X(6) = x(0) + x(4) - \{x(2) + x(6)\} + W_N^6\{x(1) + x(5) - \{x(3) + x(7)\}\}$$
$$X(7) = x(0) - x(4) + W_N^6\{x(2) + x(6)\} + W_N^7\{x(1) - x(5) + W_N^6\{x(3) - x(7)\}\}$$

elde edilmiş olur.

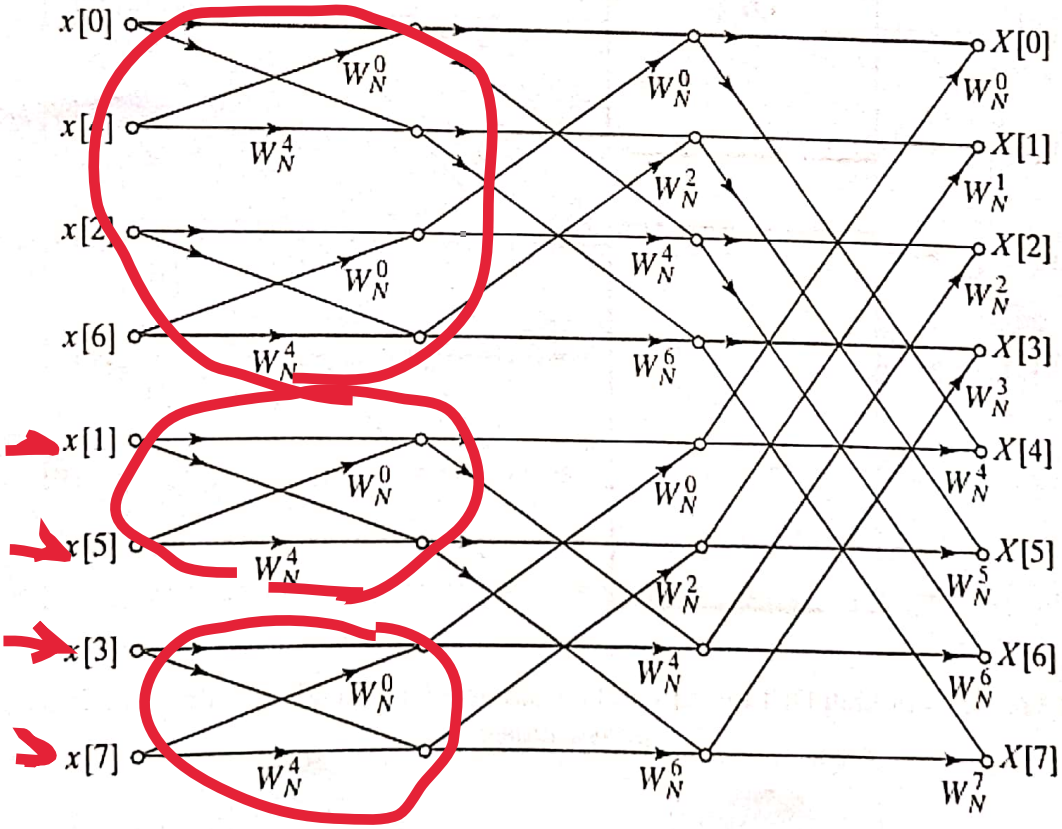


Şekil 3.6: Sekiz noktalı DFT'nin zamanda desimasyon ile iki adet dört noktalı DFT'ye ayrıştırılması

CamScanner ile tarandı

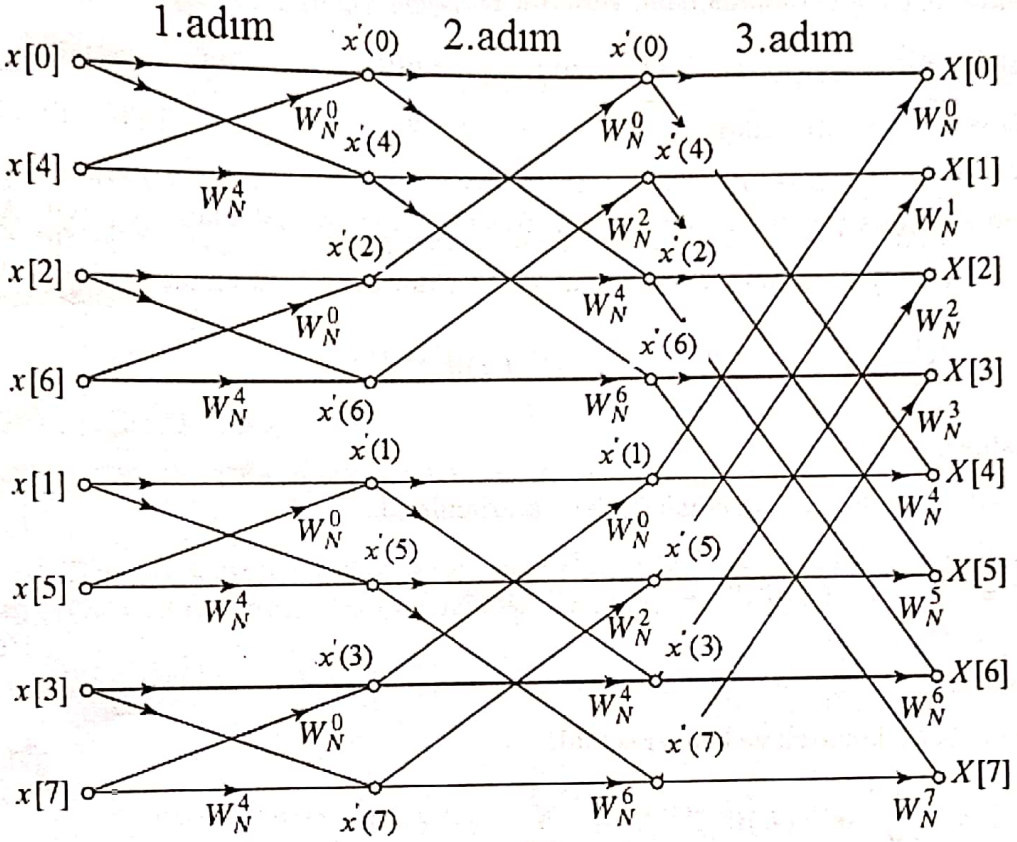


Şekil 3.7: İki adet dört noktalı DFT'nin zamanda desimasyon ile dört adet iki noktalı DFT'ye ayrıştırılması



Şekil 3.8: Sekiz noktalı zamanda desimasyonlu FFT için akış diagramı

CamScanner ile tarandı



Şekil 3.9: Sekiz noktalı zamanda desimasyonlu FFT için akış diagramı

Frekansta Desimasyonlu FFT

Alternatif FFT algoritması diziyi orta noktasından ikiye ayırarak ve aynı işlem uygulanarak geliştirilebilir.

$$x^{(1)}(n) = x(n)$$
$$x^{(2)}(n) = x(n+N/2)$$
$$n = 0,1,2.....(N/2)-1$$

biçiminde ikiye ayrılsın.

$$X_k = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$
$$= \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + \sum_{n=N/2}^{N-1} x(n) W_N^{nk} \qquad (3.37)$$
$$= \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + \sum_{n=0}^{(N/2)-1} x(n+N/2) W_N^{(n+(N/2))k}$$

CamScanner ile tarandı

şeklinde yazılabilir. Tanımlamalar yerlerine konursa,

$$X_k = \sum_{n=0}^{(N/2)-1} x^{(1)}(n) W_N^{nk} + \sum_{n=0}^{(N/2)-1} x^{(2)}(n) W_N^{(n+(N/2))k} \tag{3.38}$$

elde edilir.

Frekans desimasyonunda çift ve tek frekans domeni noktaları X_{2k} ve X_{2k+1} $k = 0,1,...,(N/2)-1$ aralığında kullanılarak DFT elde edilebilir. k yerine $2k$ koyarsak,

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

yazılabilir.

$W_N^{2nk} = W_{N/2}^{nk}$ ve $W_N^{kN} = 1$ özelliklerinden yararlanılarak,

$$X_{2k} = \sum_{n=0}^{(N/2)-1} x^{(1)}(n) W_{N/2}^{nk} + \sum_{n=0}^{(N/2)-1} x^{(2)}(n) W_{N/2}^{nk} \tag{3.39}$$

elde edilir.

k yerine $2k+1$ konarak ve benzer şekilde

$$\begin{aligned} X_{2k+1} &= \sum_{n=0}^{(N/2)-1} x^{(1)}(n) W_N^{n(2k+1)} + \sum_{n=0}^{(N/2)-1} x^{(2)}(n) W_N^{(n+(N/2)(2k+1))} \\ &= \sum_{n=0}^{(N/2)-1} W_N^n x^{(1)}(n) W_{N/2}^{nk} - \sum_{n=0}^{(N/2)-1} W_N^n x^{(2)}(n) W_{N/2}^{nk} W_N^{Nk} W_N^{N/2} \end{aligned}$$

elde edilir.

$W_N^{Nk} = 1$ ve $W_N^{N/2} = -1$ özelliklerinden yararlanarak,

$$X_{2k+1} = \sum_{n=0}^{(N/2)-1} W_N^n x^{(1)}(n) W_{N/2}^{nk} - \sum_{n=0}^{(N/2)-1} W_N^n x^{(2)}(n) W_{N/2}^{nk} \tag{3.40}$$

elde edilir.

$k = 0,1,...,(N/2)-1$ için X_{2k} ve X_{2k+1}'den N için frekansta desimasyonlu FFT görülmektedir:

$$\begin{aligned} X_{2k} &= \sum_{n=0}^{(N/2)-1} \left[x^{(1)}(n) + x^{(2)}(n) \right] W_{N/2}^{nk} \\ X_{2k+1} &= \sum_{n=0}^{(N/2)-1} \left[x^{(1)}(n) - x^{(2)}(n) \right] W_N^n W_{N/2}^{nk} \end{aligned} \tag{3.41}$$

CamScanner ile tarandı

Hesaplama Sayısı Karşılaştırılması

DFT hesaplanmasında N^2 karmaşık çarpma ve $N(N-1)$ karmaşık toplama gereklidir. FFT yardımıyla $N=2^R$ noktadan oluşan dizinin DFT'sinin hesaplanmasında $NR/2$ karmaşık çarpma ve NR harmaşık toplama işlemi yeterlidir. Adım sayısı $R=\log_2 N$ olarak yazılırsa işlem yoğunluğu açısından DFT ile FFT kıyaslanabilir.

Matlab Uygulamaları

Örnek 3.5: DFT ile FFT arasındaki işlem süresinin kıyaslanması.

TIC ve TOC fonksiyonu kullanılarak yapılabilir. TIC fonksiyonu zamanlayıcıyı başlatır TOC fonksiyonuda zamanlayıcı bilgisini okur.

```
N=2048;
x=rand(1,N);
tic
fft(x,N);
toc

tic
radix2hfd(x,N);
toc

tic
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=x*WNnk;
toc

elapsed_time =
   0.0940

elapsed_time =
  56.172000

elapsed_time =
  17.8750
```


CamScanner ile tarandı

Örnek 3.6: Zamanda desimasyonlu FFT algoritmasının incelenmesi.

```
function[X]=radix2hfd(x,M)
if nargin ==2
   N=M;
else
   N=length(x);
end
%global M;
N=length(x);
L=log2(N);
if L==1
   X=[x(1)+x(2) x(1)-x(2)]
end
if L>1
   WN=exp(-j*2*pi/N);
   W(1,:)=WN.^([0:1:N/2-1]);
   x0(1,:)=x(1:2:end);
   x1(1,:)=x(2:2:end);
   X0(1,:)= radix2hfd(x0)+W.* radix2hfd(x1);
   X1(1,:)= radix2hfd(x0)-W.* radix2hfd(x1);
   X=[X0 X1];
end
```

Ödev 3.2:

1) *N*=8 için frekansda desimasyonlu Hızlı Fourier Dönüşümünün akış diyagramını çiziniz ve sekiz noktalı FFT'yi Örnek 3.4'dekine benzer şekilde üç adımda hesaplayınız.

2) Matlab'da fft komutu kullanmadan frekans desimasyonlu FFT algoritmasını üretiniz.

CamScanner ile tarandı