

Capítulo 2

Transformada Discreta de Fourier

2.1. Introducción

Hasta ahora hemos visto que para realizar el análisis en frecuencia de una señal discreta $x[n]$, se aplica sobre ella la Transformada de Fourier, obteniéndose $X(e^{j\omega})$. Dicha transformada es una función continua y por lo tanto no se puede manejar de forma discreta. Sería interesante disponer de una representación frecuencial discreta que nos facilitara la manipulación de los datos en el dominio de la frecuencia de forma discreta por medio de un procesador de señal u ordenador.

En este capítulo se presenta la herramienta que nos permite representar el espectro de forma discreta, llamada Transformada Discreta de Fourier (DFT). Para el cálculo de dicha Transformada Discreta de Fourier existen algoritmos muy eficientes que permiten el cálculo de la misma con un número reducido de operaciones. Ello ha permitido que la DFT constituya una herramienta básica no sólo para el análisis frecuencial sino en numerosas aplicaciones.

2.2. Representación de Fourier de secuencias de duración limitada. La DFT

Supongamos una secuencia de duración limitada $x[n]$ tal que:

$$x[n] = \begin{cases} \neq 0 & 0 \leq n \leq N - 1 \\ 0 & \text{resto} \end{cases}$$

Se define la Transformada Discreta de Fourier (DFT) de dicha señal como:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad k = 0, \dots, N-1 \quad (2.1)$$

y la Transformada Discreta de Fourier Inversa (DFT⁻¹) como:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N} \quad n = 0, \dots, N-1 \quad (2.2)$$

Por tanto, la DFT puede considerarse como una transformación que toma las N muestras de una señal para dar lugar a N coeficientes $X[k]$ y viceversa.

Relación con la Transformada de Fourier

La señal $x[n]$ tiene una transformada de Fourier que viene dada por:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \quad (2.3)$$

donde la última igualdad se debe al hecho de que la señal $x[n]$ es de duración limitada.

Comparando la anterior ecuación con la definición de DFT se puede ver que:

$$X[k] = X(e^{j\omega_k}) \text{ donde } \omega_k = \frac{2\pi k}{N}, \quad f_k = \frac{k}{N}, \quad 0 \leq k \leq N-1 \quad (2.4)$$

es decir los N valores de la DFT son muestras de la Transformada de Fourier $X(e^{j\omega})$ de la señal $x[n]$.

Nótese que las muestras tienen un espaciamiento en frecuencia $\Delta f = 1/N$, y que cubren de forma completa un periodo de la Transformada de Fourier

Extensión de la señal con ceros

Es interesante notar que una señal de *duración real* M muestras puede ser considerada también como de *duración ficticia* N , siendo $N > M$. Para ello basta considerar que las muestras en exceso son nulas.

Esta observación nos permite, usando la DFT, calcular muestras de la Transformada de Fourier de una señal, con un espaciamiento arbitrariamente pequeño sin más que añadir ceros a la señal hasta completar una duración ficticia igual al número de muestras espectrales que deseemos.

La relación entre el valor del índice de la transformada y la frecuencia de la Transformada de Fourier es:

$$k \longrightarrow f_k = \frac{k}{N} \longrightarrow \omega_k = \frac{2\pi k}{N} \quad (2.5)$$

Periodicidad de la DFT

Si en la ecuación 2.1 intentamos calcular $X[k]$ para un valor $k \geq N$ se puede comprobar que:

$$X[k+N] = X[k] \quad 0 \leq k \leq N$$

Lo mismo sucede con la transformada inversa.

En otras palabras, aunque la DFT se calcula normalmente en el intervalo $0 \leq k \leq N-1$, la expresión 2.1 se puede calcular para cualquier valor de k y resulta ser periódica de periodo N .

Esta periodicidad tiene su reflejo en todas las propiedades de la DFT

Notación

Denotaremos como $((n))_N$, y leeremos n módulo N a lo siguiente:

$$((n))_N = \begin{cases} \text{resto}(n/N) & n \geq 0 \\ N - \text{resto}(|n|/N) & n < 0, |n| \text{ no múltiplo de } N \\ 0 & n < 0, |n| \text{ múltiplo de } N \end{cases}$$

Es fácil observar que se cumple siempre:

$$0 \leq ((n))_N \leq N-1 \quad \forall n$$

2.3. Propiedades de la DFT

Linealidad

Sean dos secuencias $x_1[n]$ y $x_2[n]$ de duraciones N_1 y N_2 .¹ La secuencia suma de ambas, $x_3[n]$, tendrá una duración N que será el máximo de N_1 y N_2 . La DFT de N puntos de $x_3[n]$ será:

$$x_3[n] = x_1[n] + x_2[n] \xrightarrow{\text{DFT}_N} X_3[k] = X_1[k] + X_2[k]$$

¹Las señales se consideran no nulas en el intervalo $0, \dots, N_1-1$ y $0, \dots, N_2-1$ respectivamente.



Figura 2.1: Desplazamiento circular de señales.

siendo $X_1[k]$ y $X_2[k]$ las DFTs de ($N = \max(N_1, N_2)$) muestras de $x_1[n]$ y $x_2[n]$ respectivamente (aumentando con ceros la duración de la más corta).

Desplazamientos circulares

Sea $x[n]$ una señal de duración finita N , definida entre 0 y $N - 1$. Se define el operador desplazamiento circular o cíclico como:

$$y[n] = x[((n - n_0))_N]$$

siendo n_0 un desplazamiento entero. El efecto que causa un desplazamiento cíclico sobre una señal se muestra en la figura 2.1. Puede observarse que:

- La señal desplazada cíclicamente también es de duración finita N y no nula en el intervalo $0 \leq n \leq N - 1$.
- El desplazamiento circular es similar a uno normal (llamado en este contexto lineal) salvo por el hecho de que las muestras que al desplazar se saldrían del intervalo $0 \leq n \leq N - 1$ reaparecen circularmente por el otro extremo.

Si la secuencia $x[n]$ de duración N tiene por DFT de N puntos (DFT_N) a $X[k]$ entonces:

$$x[((n - n_0))_N] \xleftarrow{\text{DFT}_N} e^{-j \frac{2\pi}{N} n_0 k} X[k]$$

Nótese que los desplazamientos circulares únicamente afectan a la fase de la DFT.



Figura 2.2: Inversión circular de señales.

Dualidad e inversión cíclica

Sea $x[n]$ una señal de duración finita N , definida entre 0 y $N - 1$. Se define el operador inversión circular o cíclica como:

$$y[n] = x[((-n))_N]$$

El efecto que causa una inversión cíclica sobre una señal se muestra en la figura 2.2.

Puede verse lo siguiente:

- La señal invertida circularmente también es de duración finita N y no nula en el intervalo $0 \leq n \leq N - 1$.
- La inversión cíclica conserva la muestra de $n = 0$ en su ubicación y el resto de muestras cambian de orden.
- La inversión circular de la inversión circular es la señal original, es decir si

$$y[n] = x[((-n))_N]$$

entonces

$$y[((-n))_N] = x[n]$$

Sea una secuencia $x[n]$ de duración finita N tal que:

$$x[n] \xrightarrow{\text{DFT}_N} X[k]$$

entonces:

$$X[n] \xrightarrow{\text{DFT}_N} N x[((-k))_N]$$

es decir, si se calcula la DFT (directa) sobre el resultado de una DFT se obtiene la inversión circular de la señal original (salvo el factor de escala N).

Esta propiedad puede ser útil para calcular DFTs inversas mediante DFTs directas y un cierto reordenamiento del resultado.

Simetrías

Sea una secuencia $x[n]$ de duración N tal que:

$$x[n] \xrightarrow{\text{DFT}_N} X[k]$$

entonces se tienen las siguientes relaciones:

- $x[((-n))_N] \xleftrightarrow{\text{DFT}_N} X[((-k))_N]$
- $x^*[n] \xleftrightarrow{\text{DFT}_N} X^*((-k))_N]$
- $x^*((-n))_N \xleftrightarrow{\text{DFT}_N} X^*[k]$

- Por tanto, la DFT de una señal real que cumple que

$$x[n] = x^*[n]$$

se tiene que

$$X[k] = X^*((-k))_N$$

Esta es la propiedad de simetría conjugada usual en las transformadas de Fourier de señales reales.

- Para que la DFT de una señal sea real debe cumplirse $x[n] = x^*((-n))_N$, que en el caso de que la señal $x[n]$ sea real implica $x[n] = x[((-n))_N]$.

Modulación

Sea una secuencia $x[n]$ de duración finita N tal que:

$$x[n] \xrightarrow{\text{DFT}_N} X[k]$$

entonces:

$$x[n] e^{j2\pi k_0 n/N} \xleftrightarrow{\text{DFT}_N} X[((k - k_0))_N]$$

Convolución circular

Sean dos secuencias $x_1[n]$ y $x_2[n]$ ambas de duración N ($N = \max\{N_1, N_2\}$, completando con ceros la más corta si fuera necesario) cuyas DFT _{N} son, respectivamente, $X_1[k]$ y $X_2[k]$. Supongamos que calculamos:

$$X_3[k] = X_1[k] X_2[k] \quad k = 0, \dots, N - 1$$

2.3. Propiedades de la DFT

y que obtenemos la secuencia $x_3[n]$:

$$x_3[n] = \text{DFT}^{-1}(X_3[k]) \quad n = 0, \dots, N - 1$$

La relación que existe entre $x_3[n]$ y $x_1[n]$ y $x_2[n]$ es:

$$x_3[n] = \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N] \quad (2.6)$$

y recibe el nombre de *convolución circular*. Se suele escribir:

$$x_3[n] = x_1[n] \circledast x_2[n]$$

En la figura 2.3 puede verse un ejemplo de convolución circular entre dos secuencias de duración limitada. El valor de N seleccionado para realizar la convolución es el máximo de las duraciones de ambas secuencias, en este caso $N = 5$. Dicho ejemplo puede compararse con la convolución lineal de ambas secuencias mostrada en la figura 2.4. Comparando ambas convoluciones podemos observar que la circular es distinta de la lineal por la aparición de las muestras que entran al intervalo de suma por la derecha debidas a la inversión $((-m))_N$, que coinciden con las que aparecen en la parte de $m < 0$ en la convolución lineal.

Con respecto a la convolución circular cabe hacer una serie de observaciones:

- En general, la convolución circular de dos secuencias NO coincide con la convolución lineal.
- La convolución circular, en la práctica, no se calcula nunca aplicando la ecuación (2.6).
- En la práctica, la convolución circular se calcula siempre mediante DFT inversa del producto de dos DFTs. Ello se debe a la existencia de algoritmos muy eficientes de cálculo de la DFT que se estudian al final del tema (FFT).
- La convolución lineal es útil para calcular la salida de filtros.
- La convolución circular no sirve para nada.

Podríamos resumir lo anterior diciendo que la convolución circular es algo que se calcula muy rápido pero no sirve para nada.

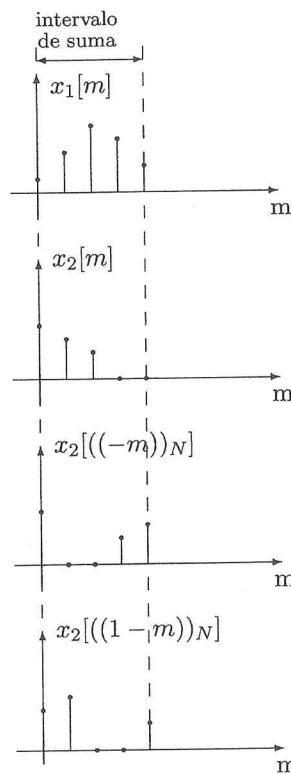


Figura 2.3: Procedimiento de convolución circular módulo 5 entre dos secuencias.

En realidad la convolución circular sí tiene una utilidad y ésta es que, bajo determinadas condiciones, puede lograrse que la convolución circular coincida con la convolución lineal (recordemos que en general esto no sucede). En ese caso se tiene la eficiencia computacional de la convolución circular y la utilidad de la lineal. En la siguiente sección nos ocuparemos de ver las condiciones para que esto suceda.

2.4. Relación entre convolución circular y lineal

En el apartado anterior hemos visto que en general la convolución circular y lineal son diferentes. En este punto veremos que bajo ciertas condiciones se

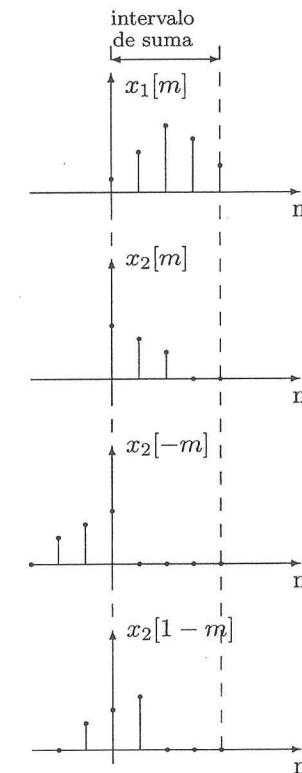


Figura 2.4: Procedimiento de convolución lineal entre dos secuencias $x_1[n]$ y $x_2[n]$.

puede lograr que algunos e incluso todos los valores de la convolución circular sean idénticos a la convolución lineal.

Esto nos será de utilidad en la siguiente sección en la que veremos cómo se pueden realizar filtrados (convoluciones lineales) usando convoluciones circulares (productos de DFTs, que se calculan con pocas operaciones).

2.4.1. Condiciones para la igualdad entre convolución lineal y circular

En este apartado vamos a establecer las condiciones bajo las cuales la convolución circular y la lineal coinciden. Supongamos que tenemos dos señales de duración finita:

- $x[n]$ de duración L muestras.
- $h[n]$ de duración P muestras.

Llamemos $N = \max(L, P)$ y calculemos $z[n] = x[n] \circledast h[n]$ e $y[n] = x[n] * h[n]$. En las figuras 2.3 y 2.4 podemos ver cuál es la causa de que convolución lineal y circular no coincidan. Las diferencias se deben a las muestras que

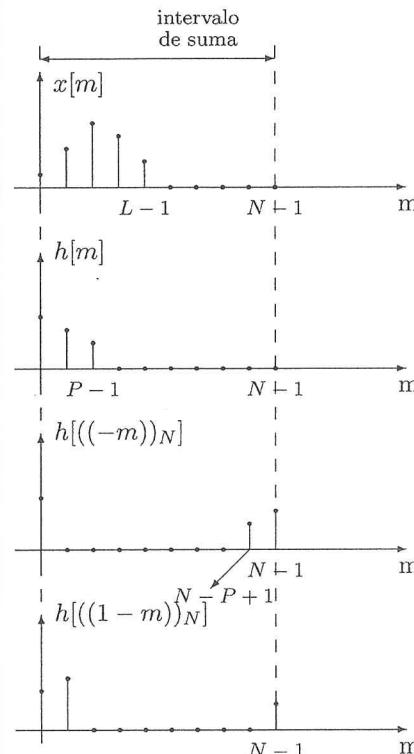


Figura 2.5: Determinación del periodo N para realizar la convolución circular. Hay que lograr que $(N - P + 1) > (L - 1)$.

2.4. Relación entre convolución circular y lineal

aparecen en la parte derecha en torno a $m = N - 1$ en la convolución circular (figura 2.5). La idea para lograr que ambas coincidan es hacer que dichas muestras se multipliquen por cero. Para ello, deberemos calcular la convolución circular de un tamaño superior a L y P . El valor mínimo de N necesario es:

$$N = L + P - 1 \quad (2.7)$$

Ya hemos indicado que las convoluciones circulares se calculan en la práctica mediante DFTs. Por tanto, podremos resumir el procedimiento a seguir para realizar la convolución lineal mediante DFTs del siguiente modo:

1. Añadir ceros por la derecha a $x[n]$ y $h[n]$ hasta completar un tamaño $N \geq (L + P - 1)$
2. Calcular las DFT_N tanto de $x[n]$ como de $h[n]$.
3. Realizar el producto elemento a elemento de los vectores $X[k]$ y $H[k]$, con $k = 0, \dots, N - 1$.
4. Calcular la DFT inversa del resultado del producto anterior.

Para finalizar, en la figura 2.6 se muestra gráficamente el método anterior.

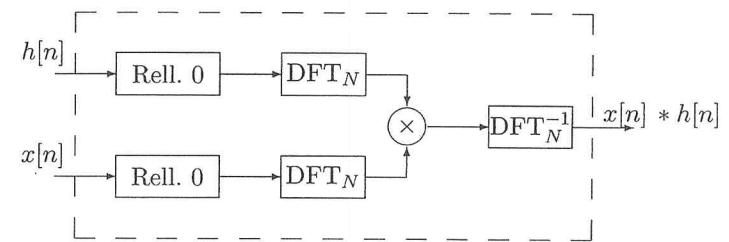


Figura 2.6: Método de cálculo de la convolución lineal usando DFT

2.4.2. Coincidencias entre la convolución lineal y la circular

En este apartado supondremos que tenemos dos señales:

- $x[n]$ de duración L muestras.
- $h[n]$ de duración P muestras.

donde $P < L$. Supongamos ahora que calculamos $c[n]$, la convolución circular módulo N de ambas señales, tomando $N = L$.

$$c[n] = x[n] \circledast h[n]$$

Sea $z[n]$ la convolución lineal entre $x[n]$ y $h[n]$.

$$z[n] = x[n] * h[n]$$

Para calcular dichas convoluciones, y teniendo en cuenta que $x[n]$ sólo es no nula para $0 \leq n \leq N - 1$

$$z[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m] h[n-m] = \sum_{m=0}^{N-1} x[m] h[n-m]$$

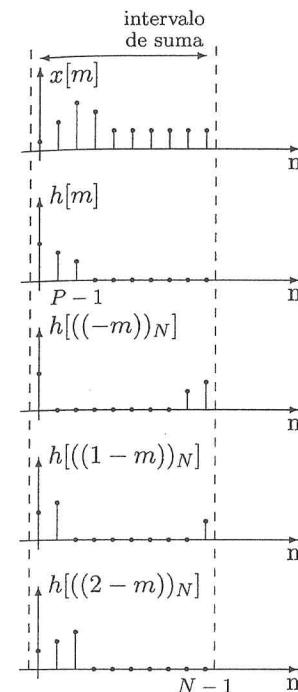
$$c[n] = x[n] \circledast h[n] = \sum_{m=0}^{N-1} x[m] h[((n-m))_N]$$

donde podemos observar que la única diferencia consiste en que la inversión y los desplazamientos de la señal h son lineales para la convolución lineal y circulares para la convolución cíclica. La figura 2.7 ilustra este hecho. En ella se puede apreciar que las diferencias surgen por las muestras de la derecha de la versión desplazada circularmente. Sin embargo, conforme n aumenta resulta que cada vez hay menos muestras en la parte de la derecha hasta que para $n \geq 2$ (en este ejemplo), se da la siguiente situación

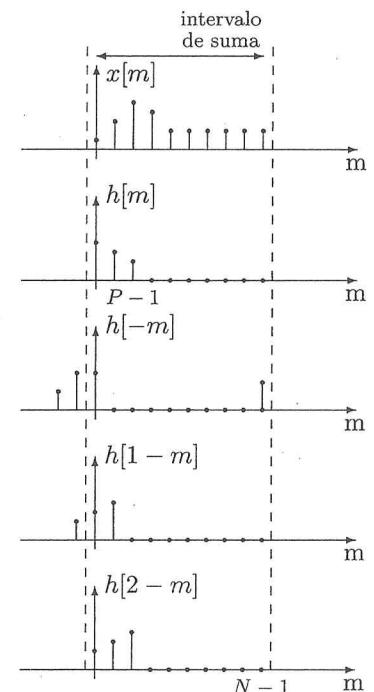
$$h[2-m] = h[((2-m))_N] \quad 0 \leq m \leq N-1$$

A partir de ese momento, si seguimos desplazando hasta llegar a $n = N - 1$, las versiones desplazadas cíclica y linealmente coinciden, y con ellas coinciden los correspondientes valores de la convolución lineal y circular.

2.4. Relación entre convolución circular y lineal



a)



b)

Figura 2.7: Comparación entre los desplazamientos que suceden cuando la convolución circular se realiza de un tamaño igual a la longitud de la señal más larga. a) Convolución circular. b) Convolución lineal.

Analizando un poco la figura, es fácil darse cuenta que en general los $P - 1$ primeros valores de la convolución circular no coinciden con los de la convolución lineal, es decir

$$z[n] \neq c[n] \quad 0 \leq n < P - 1$$

pero el resto sí que coinciden

$$z[n] = c[n] \quad P - 1 \leq n \leq N - 1$$

El hecho de que aunque no sean idénticas por completo, sin embargo tengan algunos valores comunes será utilizado por uno de los métodos que se detallan en la sección 2.5.

2.5. Implementación de filtros LTI utilizando DFT

Acabamos de ver cómo se puede utilizar la DFT para realizar convoluciones lineales. En esta sección veremos cómo utilizarla para filtrar señales. Consideraremos únicamente filtros FIR causales cuya respuesta impulsiva será de duración P muestras (definida entre $n = 0, \dots, P - 1$).

Si la señal a filtrar $x[n]$ es de duración finita, podemos aplicar lo visto en el punto anterior. No obstante, si la señal a filtrar tuviera duración infinita, el método anterior no podría llevarse a la práctica. En general tendremos problemas cuando:

- La señal tenga un comienzo, pero no se sepa cuando termina. En ese caso deberemos esperar hasta que se dé por terminada $x[n]$. En aplicaciones interactivas (telefonía) esto puede ser inaceptable.
- Otro problema será el retardo: para calcular la DFT (y por tanto la convolución) se necesita que estén presentes *todas* las muestras de la señal a filtrar ($x[n]$), por lo que tendremos un retardo *algorítmico* igual a la duración de la señal.

Para mitigar los anteriores problemas se utilizan dos métodos basados en las siguientes ideas:

1. Se divide la señal de entrada en trozos de duración razonable para que el retardo no sea muy largo.
2. Se calcula la salida para cada trozo.
3. Finalmente, se combinan las respuestas de cada trozo de forma adecuada.

Los dos métodos para filtrar señales basándonos en la DFT se conocen como el método de *solape-suma* y el de *solape-almacenamiento*. En los siguientes apartados se describen los mismos.

Método de solape-suma

Los pasos a seguir para filtrar una señal $x[n]$ de duración infinita (o desconocida) con un filtro $h[n]$ de duración P utilizando este método son:

1. Se divide $x[n]$ en trozos de longitud L , sin solape y sin dejar huecos. Es decir, el primer bloque (bloque 0) incluirá las muestras de $n = 0, \dots, L - 1$, el segundo (bloque 1) las muestras $n = L, \dots, 2L - 1$, y

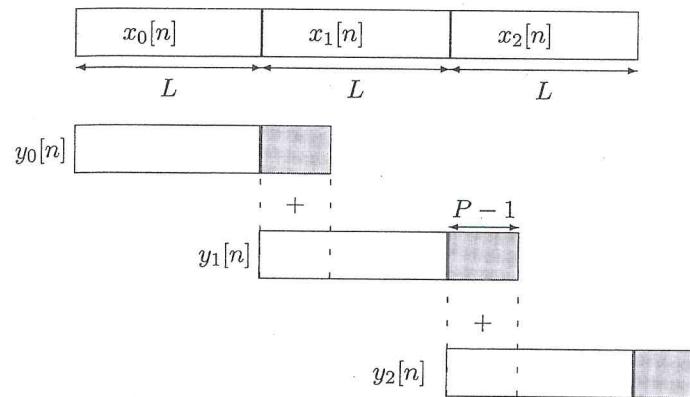


Figura 2.8: Método Solape-Suma

así sucesivamente. Llamaremos $x_k[n]$ al bloque k -ésimo de muestras de entrada. Nótese que

$$x[n] = \sum_{k=0}^{\infty} x_k[n]$$

2. Como cada bloque es de duración finita se calcula la respuesta a cada bloque utilizando el método visto en la sección 2.4. Para ello:

- Se desplaza previamente cada trozo al origen:

$$x'_k[n] = x_k[n + kL]$$

- Se calcula $y'[k]$ utilizando el esquema de la figura 2.6.
- Se desplaza la salida k -ésima a su posición.

$$y_k[n] = y'_k[n - kL]$$

3. La salida total se calcula como:

$$y[n] = \sum_{k=0}^{\infty} y_k[n]$$

Dado que los bloques $y[k]$ tienen una duración mayor que la de los bloques $x[k]$, existirá un solapamiento entre las respuestas de los distintos bloques, lo que obligará a tener que realizar sumas de las colas de un bloque con el comienzo del siguiente. La cantidad de muestras que se solapan es $P - 1$. (Ver figura 2.8).

Método de solape-almacenamiento

La señal, como en el caso anterior, $x[n]$ de duración infinita, se pretende filtrar con un filtro $h[n]$ de duración P , para lo cual se debe seguir:

1. Se divide $x[n]$ en trozos de longitud L , (en este método $L > P$). Llamaremos a cada trozo $x_k[n]$ con $k = 0, 1, \dots$
2. Se calculan las DFTs de L puntos tanto del trozo k -ésimo como de $h[n]$. Se multiplican y se calcula la DFT inversa de L puntos. El resultado, como sabemos, es un vector de L muestras que contiene la convolución circular.

$$z_k[n] = x_k[n] \circledR y_k[n]$$

De cada vector $z_k[n]$:

- Las primeras $P - 1$ muestras no coinciden con la convolución lineal y se desechan.
 - Las muestras de $n = P, \dots, L - 1$ coinciden con las de la convolución lineal.
3. Al tomar el bloque siguiente ($k + 1$), previendo que sus primeras $P - 1$ muestras de la convolución circular van a ser erróneas, se elige el bloque de modo que sus $P - 1$ primeras muestras coincidan con las $P - 1$ muestras finales del bloque k . Finalmente, se desplaza cada salida a su lugar correspondiente para componer la señal de salida.
 4. De ese modo, la salida final, se obtiene yuxtaponiendo las muestras de cada bloque que coinciden con la convolución lineal.
- En el bloque inicial, como no hay bloque anterior con el que solapar las $P - 1$ primeras muestras, se anteponen $P - 1$ ceros.

La figura 2.9 muestra el procedimiento.

2.6. Muestreo de la Transformada de Fourier

Sea $x[n]$ una secuencia cualquiera con transformada de Fourier $X(e^{j\omega})$. Dicha señal podrá tener duración finita o infinita.

Supongamos ahora que tomamos N muestras equiespaciadas de su transformada de Fourier en $f_k = \frac{k}{N}$ ($\omega_k = 2\pi k/N$) con $k = 0, \dots, N - 1$.

El anterior conjunto de N muestras de la Transformada de Fourier podríamos suponer que son los coeficientes de la DFS de una señal periódica $\tilde{x}[n]$. Lo que

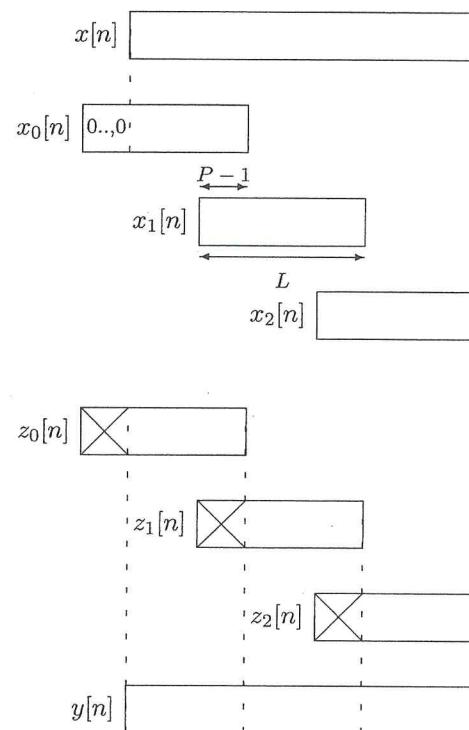


Figura 2.9: Método Solape-Almacenamiento

vamos a hacer en este apartado es encontrar la relación existente entre la señal $x[n]$ cuya transformada de Fourier se muestrea y la señal periódica $\tilde{x}[n]$ resultado del cálculo de la DFS^{-1} sobre las muestras de la Transformada de Fourier.

Las ecuaciones involucradas se muestran a continuación. En primer lugar se calcula la TF de la señal $x[n]$:

$$X(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} \quad (2.8)$$

En segundo lugar se muestrea dicha TF:

$$\tilde{X}[k] = X(e^{j\omega}) \Big|_{\omega=2\pi k/N} = X(e^{j2\pi k/N}) \quad (2.9)$$

Obsérvese que $\tilde{X}[k]$ es periódica por serlo $X(e^{j\omega})$. El periodo de $X(e^{j\omega})$ es 2π y la separación entre muestras es $2\pi/N$, con lo que el periodo de $\tilde{X}[k]$ resultará ser N . Por otro lado la DFS⁻¹ de los $\tilde{X}[k]$ obtenidos resulta ser:

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] e^{j2\pi kn/N} \quad (2.10)$$

Sustituyendo en la ecuación (2.10) la expresión de $X(e^{j\omega})$ dada por la ecuación (2.8) se obtiene:

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{m=-\infty}^{\infty} x[m] e^{-j2\pi km/N} e^{j2\pi kn/N}$$

Reordenando los sumatorios tenemos:

$$\tilde{x}[n] = \sum_{m=-\infty}^{\infty} x[m] \frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-m)k}$$

donde:

$$\frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-m)k} = \begin{cases} 1 & \text{si } n - m = rN \\ 0 & \text{resto} \end{cases} = \sum_{r=-\infty}^{\infty} \delta[n - m - rN]$$

obteniéndose finalmente:

$$\begin{aligned} \tilde{x}[n] &= \sum_{m=-\infty}^{\infty} x[m] \sum_{r=-\infty}^{\infty} \delta[n - m - rN] = x[n] * \sum_{r=-\infty}^{\infty} \delta[n - rN] \\ \tilde{x}[n] &= \sum_{r=-\infty}^{\infty} x[n - rN] \end{aligned} \quad (2.11)$$

La ecuación (2.11) indica que la secuencia periódica $\tilde{x}[n]$, que tiene como DFS los coeficientes obtenidos al muestrear la transformada de Fourier de la secuencia $x[n]$, se obtiene repitiendo la propia secuencia $x[n]$ cada N muestras y posteriormente sumando todas las repeticiones, como puede verse en el ejemplo de la figura 2.10.

Podemos realizar las siguientes observaciones:

- Si la duración de la secuencia original $x[n]$ es menor o igual que N (y por tanto finita) no habrá solape entre las distintas repeticiones. Si conocemos el intervalo en que la señal $x[n]$ era no nula será posible recuperar dicha señal a partir de $\tilde{x}[n]$.

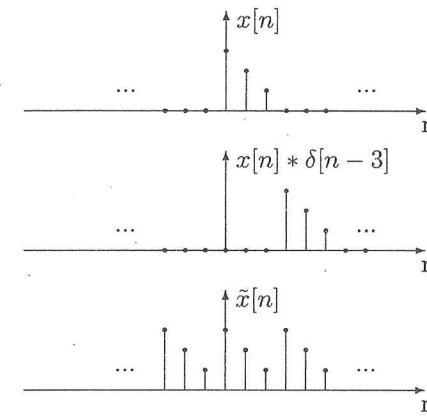


Figura 2.10: Obtención de $\tilde{x}[n]$ a partir de $x[n]$ (periodo $N=3$).

- Si la duración de la secuencia original $x[n]$ es mayor que N habrá solape. En ese caso no es posible conocer la secuencia inicial a partir de $\tilde{x}[n]$.

El razonamiento anterior nos viene a decir que si la señal es de duración finita y menor que el número de muestras tomadas de la TF, las muestras de la TF (coeficientes de la DFS de $\tilde{x}[n]$) contienen toda la información de $x[n]$ (o de forma equivalente de su TF $X(e^{j\omega})$).

Nótese la dualidad que existe entre el muestreo de la TF de una señal discreta y el muestreo de señales continuas en el tiempo:

- En el caso del muestreo en el dominio del tiempo, para que el muestreo de una señal continua $x(t)$ contenga toda la información de la señal, la separación de las muestras mínima debe ser:

$$\Delta t = \frac{1}{\text{Ancho del espectro}}$$

donde ancho del espectro es la anchura del intervalo del eje de frecuencias (positivas y negativas) en que la TF de la señal es no nula.

- En el caso del muestreo de la TF, la separación entre muestras espectrales debe ser:

$$\Delta f = \frac{1}{\text{Duración de la señal}}$$

en ambos casos la separación entre muestras consecutivas debe ser menor que la inversa de la extensión de la señal en el dominio *contrario* al que se muestrea.

2.7. Cálculo eficiente de la Transformada Discreta de Fourier (La FFT)

La FFT es un método rápido de cálculo de la transformada discreta de Fourier que fue publicado en 1964 por Cooley y Tukey, el cual es ya conocido universalmente por sus siglas en inglés FFT (Fast Fourier Transform). La eficiencia de este método se muestra en la tabla 2.1, donde se compara el número de operaciones a realizar para implementar la DFT de N puntos por el método directo (N^2) y por el algoritmo rápido ($N \log_2 N$), así como la eficiencia conseguida utilizando la FFT ($\frac{N^2}{N \log_2 N}$).

| N | Directo | FFT | Eficiencia |
|----------|-------------------|-------------------|-------------------|
| 32 | 1024 | 160 | 6.4 |
| 64 | 4096 | 384 | 10.7 |
| 128 | 16384 | 896 | 18.3 |
| 256 | 65536 | 2048 | 32 |
| . | . | . | . |
| . | . | . | . |
| 4096 | $\approx 17,10^6$ | $\approx 49,10^3$ | 341 |

Cuadro 2.1: Eficiencia de la FFT

La FFT es un método muy eficiente de cálculo, como puede verse en la tabla 2.1, si se requieren los N valores de $X[k]$; si no, puede resultar ineficiente.

En este capítulo veremos dos tipos de algoritmos para implementar la FFT:

- Algoritmos de diezmado en el tiempo.
- Algoritmos de diezmado en frecuencia.

2.7.1. Consideraciones previas

Antes de proceder a describir los algoritmos realizaremos algunas consideraciones previas.

Número de operaciones en el método directo

En este apartado trataremos de establecer de forma aproximada el número de operaciones necesario para el cálculo de la DFT directa e inversa. La DFT

(o su inversa) consisten en calcular las expresiones de las ecuaciones (2.12 y 2.13), que se reproducen a continuación:

Ecuación de Análisis (Transformada Directa)

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (2.12)$$

donde

$$W_N = e^{-j2\pi/N}$$

Ecuación de Síntesis (Transformada Inversa)

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad (2.13)$$

Así, según la ecuación (2.12), para implementar la DFT por el método directo, para cada valor de k (o para cada n si se trata de la DFT inversa) se deben realizar N productos complejos y $N - 1$ sumas complejas, por lo que en total, el número de operaciones a realizar será:

$$N \text{ valores de } k \left(N \text{ productos complejos} + (N - 1) \text{ sumas complejas} \right)$$

$$\text{Num. de ops.} = N^2 \text{ productos complejos} + N(N - 1) \text{ sumas complejas}$$

Los productos complejos tienen mayor carga computacional², por lo que para medir el número de operaciones se mide el número de productos complejos necesarios, que en el caso del cálculo de la DFT es N^2 . Por otro lado, el número de operaciones reales que habrá que realizar siempre será proporcional al número de productos complejos, ya que:

$$\text{Num. de ops.} \approx 4N^2 \text{ productos reales} + 4N^2 \text{ sumas reales} \propto N^2 \text{ productos complejos}$$

²Sean dos números complejos $z_1 = a + jb$ y $z_2 = c + jd$, el número de operaciones reales necesarias para realizar su suma es:

$$z_1 + z_2 = (a + c) + j(c + d) \longrightarrow 2 \text{ sumas reales}$$

y para calcular su producto:

$$z_1 \times z_2 = (ac - bd) + j(ad + bc) \longrightarrow 4 \text{ productos reales y 2 sumas reales}$$

Propiedades de las exponentiales complejas

Las propiedades de las exponentiales complejas a usar para disminuir el número de operaciones necesarias son:

- Simetría conjugada.

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

- Periodicidad en n y en k .

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

- Si N par, entonces:

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

- $W_{N/2} = W_N^2$

Transformada inversa

En el resto del tema, sólo desarrollaremos los distintos tipos de algoritmos para el cálculo de la DFT, ya que las estructuras necesarias para implementar la DFT inversa son similares (únicamente se diferencian en los coeficientes multiplicativos). Además, la DFT inversa se puede incluso calcular utilizando el algoritmo rápido de cálculo de la DFT ya que hay relaciones entre ellas que nos permiten hacerlo. Como ejemplo, presentaremos dos formas de relacionar ambas transformaciones:

1. DFT y DFT inversa se diferencian únicamente en el factor $1/N$ y en el signo de la exponencial $e^{j\frac{2\pi}{N}k}$, como puede observarse comparando las ecuaciones de análisis y síntesis (2.12 y 2.13). Por tanto, se podría realizar la DFT inversa usando una FFT y utilizando la expresión (2.14), resultando un algoritmo cuyo diagrama de bloques se muestra en la figura 2.11.

$$x[n] = \frac{1}{N} \left(\sum_{k=0}^{N-1} X^*[k] e^{-j\frac{2\pi}{N} n k} \right)^* \quad (2.14)$$

2. Aplicando la propiedad de la dualidad de la DFT. Si

$$x[n] \xrightarrow{\text{DFT}} X[k]$$

2.7. La FFT

entonces

$$X[k] \xrightarrow{\text{DFT}} N x[(-n)]_N \quad 0 \leq n \leq N-1$$

Basándonos en esta expresión podemos obtener $x[n]$ utilizando un algoritmo FFT al que se introduce como entrada los coeficientes de la DFT ($X[k]$), simplemente reordenando el resultado y multiplicando las muestras de salida por un factor $1/N$.

2.7.2. Algoritmos de diezmado en el tiempo

En esta sección se van a explicar el primer tipo de algoritmos FFT, los de diezmado en el tiempo. Estos algoritmos, al igual que los de diezmado en frecuencia que veremos en la sección 2.7.3, son más eficientes si la DFT a calcular es de N puntos, con N potencia de 2 ($N = 2^v$ $v \in \mathbb{Z}$). Sin embargo, en la sección 2.7.5 veremos una generalización de los mismos para el caso en que N no sea potencia de 2.

Los algoritmos de diezmado en el tiempo se basan en descomponer $x[n]$ en subsecuencias recursivamente, calcular las DFTs de cada subsecuencia y combinarlas para componer la DFT de subsecuencias mayores, hasta llegar a la secuencia $x[n]$ de N puntos.

Tomando como partida la ecuación de análisis de la DFT,

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

se separa el sumatorio en términos de n par e impar, resultando:

$$X[k] = \sum_{n \text{ par}} x[n] W_N^{kn} + \sum_{n \text{ impar}} x[n] W_N^{kn} =$$

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

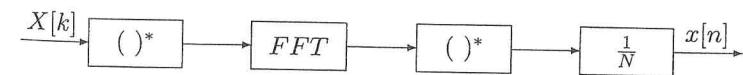


Figura 2.11: FFT inversa usando la FFT directa.

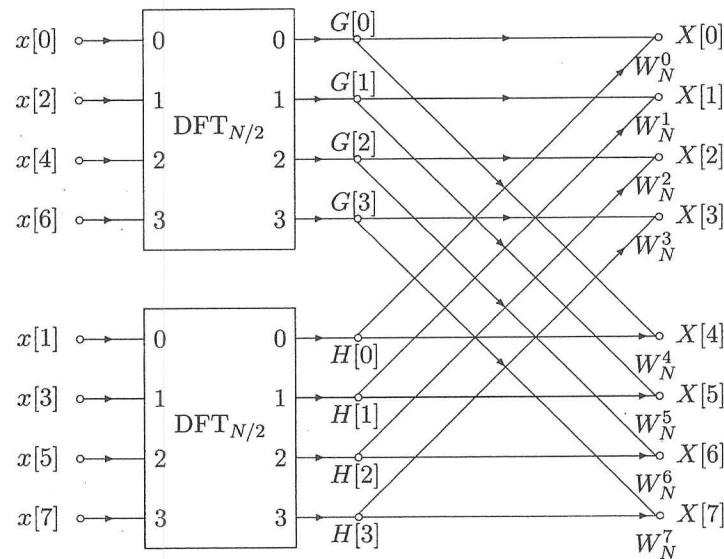


Figura 2.12: Algoritmo FFT de diezmado en el tiempo ($N = 8$): descomposición inicial.

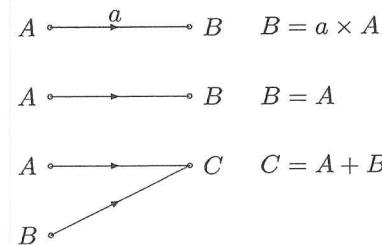


Figura 2.13: Símbolos utilizados en los diagramas de bloques del algoritmo FFT.

Si se tiene en cuenta que:

$$W_N^2 = W_{\frac{N}{2}}$$

2.7. La FFT

resulta:

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_{\frac{N}{2}}^{rk} + \left[\sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_{\frac{N}{2}}^{rk} \right] W_N^k$$

llamando:

$$G[k] = \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk}$$

$$H[k] = \sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{rk}$$

se obtiene la nueva ecuación de análisis reducida.

$$X[k] = G[k] + H[k] W_N^k \quad (2.15)$$

Obsérvese que $G[k]$ es la DFT de $N/2$ puntos de las muestras pares de $x[n]$ y $H[k]$ de las impares, sin embargo, la ecuación (2.15) es válida para $k = 0 \dots N - 1$. Para completar la DFT de N puntos se utiliza la propiedad de periodicidad, ya que ambas DFTs calculadas ($G[k]$ y $H[k]$) son periódicas en k , de periodo $N/2$.

La figura 2.12 muestra cómo sería el cálculo de la DFT realizada de la forma vista arriba para una secuencia de $N = 8$ muestras. El significado de los distintos elementos que aparecen en dicha figura están especificados en la figura 2.13.

El número de operaciones necesarias para llevar a cabo la DFT de la forma mostrada en la figura 2.12 serán las necesarias para realizar dos DFTs de 4 puntos y 8 productos complejos. Generalizando a una DFT de N puntos, el número de operaciones (productos complejos) será:

$$2 \times \left(\frac{N}{2} \right)^2 + N = N + \frac{N^2}{2} \text{ productos complejos}$$

es decir, las necesarias para realizar dos DFT _{$N/2$} más N productos. Si $N > 2 \rightarrow N + N^2/2 < N^2$, así pues, se demuestra que con este método el número de operaciones será menor (si $N > 2$) que con el método directo.

El proceso puede repetirse para cada una de las DFTs de $N/2$ puntos de la figura 2.12 (en nuestro ejemplo, DFTs de 4 puntos como la de la figura 2.14), para ello, cada bloque de DFT de $N/2$ se divide en dos de $N/4$ seleccionando muestras pares e impares, resultando el diagrama de la figura 2.15.

Si además, aplicamos a las exponenciales que aparecen en figura 2.15 la propiedad $W_{N/2}^k = W_N^{2k}$, obtenemos la figura 2.16 en la que todas las exponenciales aparecen expresadas en la misma base (W_N).

Iterando, se podría llegar a DFTs de dos puntos, como la de la figura 2.17. Si sustituimos en nuestro ejemplo cada bloque DFT de $N/4$ puntos por el grafo de la figura 2.17 se obtiene el diagrama de bloques de la FFT de 8 puntos (figura 2.18).

Como puede observarse en la figura 2.18, el algoritmo se divide en 3 etapas, en general en $\log_2 N$ etapas. Cada una de ellas parte de N valores de entrada y obtiene N valores de salida, combinando los valores de entrada por medio de una serie de productos y sumas.

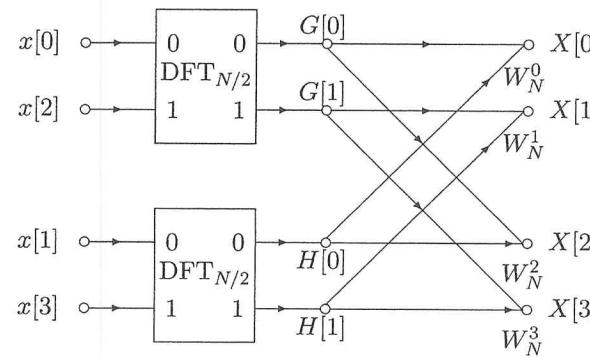


Figura 2.14: Algoritmo FFT de diezmado en el tiempo ($N = 4$).

2.7. La FFT

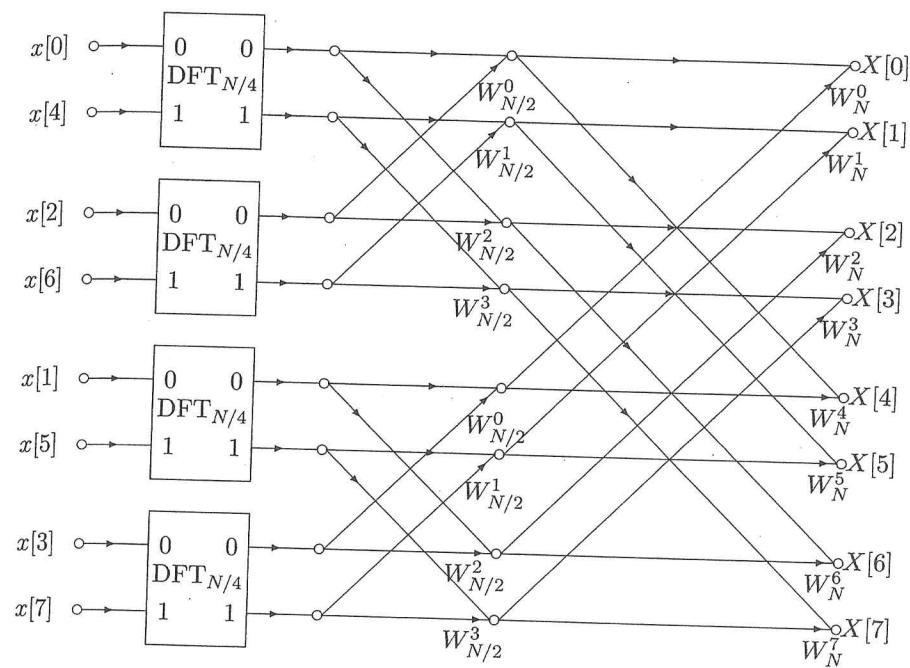


Figura 2.15: Algoritmo FFT de diezmado en el tiempo ($N = 8$): resultado de descomponer las DFTs de $N/2$ en DFTs de $N/4$.

El número de operaciones (productos complejos) del algoritmo FFT, desarrollado hasta las DFTs de 2 puntos (como el de la figura 2.18) será:³

$$N_{\frac{\text{productos}}{\text{etapa}}} \times \text{Num}_{\text{etapas}} = N \times \log_2 N$$

En la figura 2.18 se observa también que el procedimiento para pasar de una etapa a la siguiente se basa en el grafo de la figura 2.19, el cual a partir de un par de valores de una etapa y dos exponentiales, potencias de W_N con

³El número de operaciones calculado es para cálculos generalizados, pero particularizando se podrían optimizar más, por ejemplo, eliminando la multiplicación por coeficientes que valen la unidad. Por contra, no están contabilizadas las operaciones de control (por ejemplo el control de las iteraciones de los bucles). No obstante la mayoría de los DSPs tienen arquitecturas que hacen que el coste computacional de estas operaciones sea nulo.

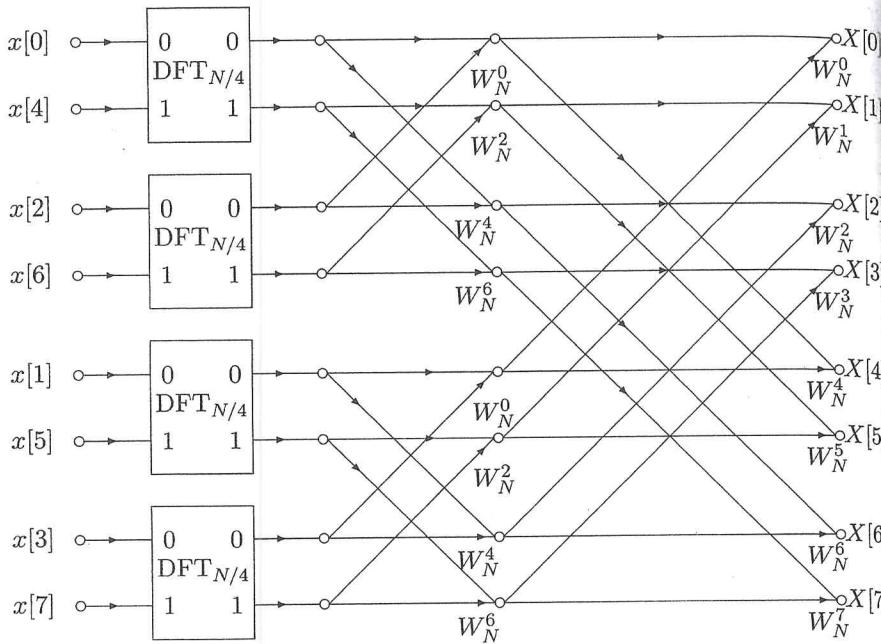


Figura 2.16: Algoritmo FFT de diezmado en el tiempo ($N = 8$): dos etapas finales.

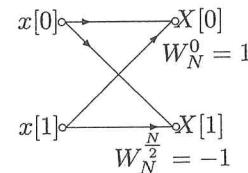


Figura 2.17: Algoritmo FFT de diezmado en el tiempo ($N = 2$).

exponentes separados $N/2$, obtiene un par de valores de la etapa siguiente. El grafo de la figura 2.19 representa la estructura de cálculo más básica y recibe el nombre de *mariposa*, debido a su forma.

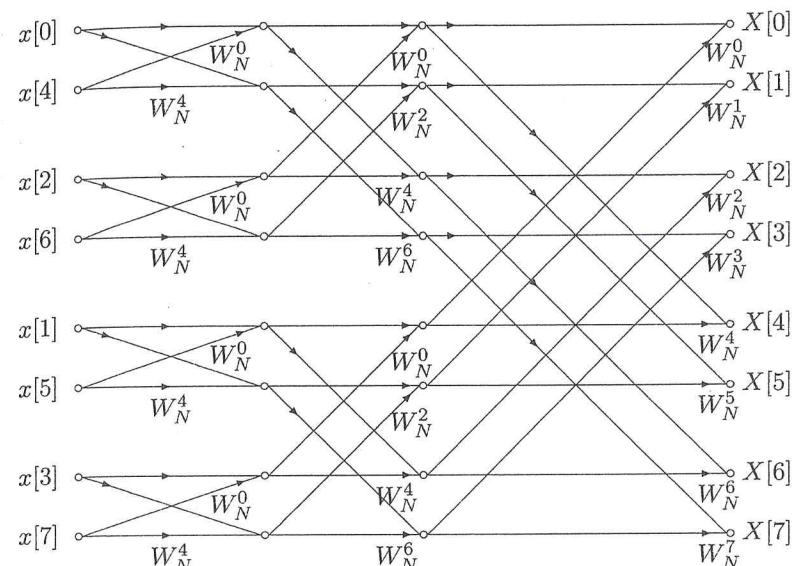


Figura 2.18: Algoritmo FFT de diezmado en el tiempo ($N = 8$).

Si se tiene en cuenta que:

$$W_N^{\frac{N}{2}} = e^{-j(\frac{2\pi}{N})\frac{N}{2}} = e^{-j\pi}$$

y por tanto el factor $W_N^{r+\frac{N}{2}}$ que aparece en la *mariposa* puede ser sustituido por:

$$W_N^{r+\frac{N}{2}} = W_N^{\frac{N}{2}} W_N^r = -W_N^r$$

el cálculo de la *mariposa* de la figura 2.19 puede simplificarse, resultando la

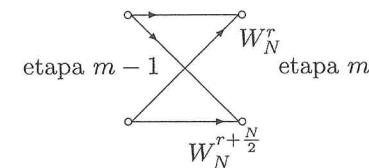


Figura 2.19: Grafo de una mariposa genérica de la figura 2.18.

de la figura 2.20, la cual requiere únicamente un producto complejo en lugar de dos para calcular los dos valores de salida.

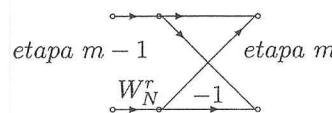


Figura 2.20: Grafo simplificado de una mariposa.

Usando la estructura de *mariposa simplificada* de la figura 2.20 en el grafo del algoritmo de la FFT de 8 puntos de la figura 2.18, se obtiene el grafo de la figura 2.21. Con esta simplificación se consigue reducir el número de operaciones finales por un factor 2, ya que el número de productos complejos a realizar en cada mariposa se reduce a la mitad. Así, el número de operaciones totales queda reducido a:

$$\text{num. de ops} = \frac{N}{2} \log_2 N$$

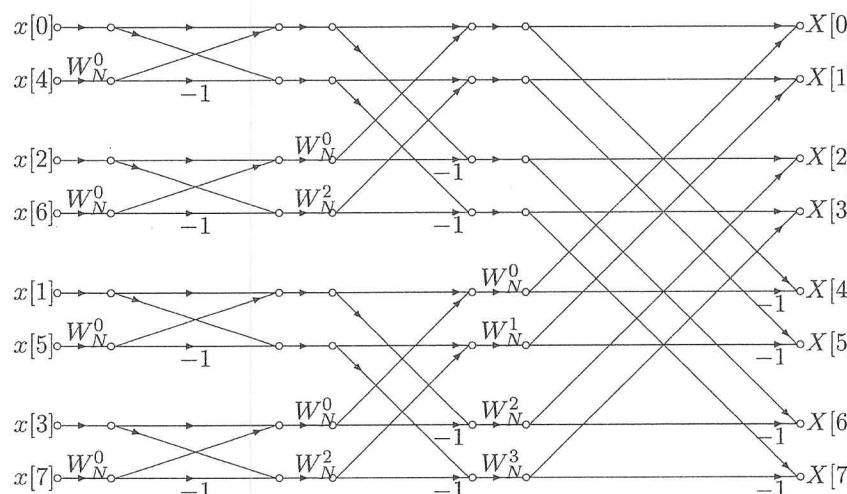


Figura 2.21: Algoritmo FFT de diezmado en el tiempo ($N = 8$) con mariposas simplificada.

2.7.3. Algoritmos de diezmado en frecuencia

Como hemos visto en el punto 2.7.2, los algoritmos de diezmado en el tiempo se basan en dividir la secuencia de entrada $x[n]$ en subsecuencias, las cuales se vuelven a dividir hasta obtener una subsecuencia de tamaño 2. Luego, las DFTs de estas subsecuencias se combinan para obtener la DFT completa. Si en vez de dividir $x[n]$ lo que se divide en subsecuencias es $X[k]$, el algoritmo resultante recibe el nombre de algoritmo de diezmado en frecuencia.

Sea $x[n]$ una secuencia de N puntos, donde N es potencia de 2 y cuya DFT es $X[k]$:

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

Si separamos $X[k]$ en dos subsecuencias de coeficientes en función del valor de k , tenemos:

$$k \text{ par} \longrightarrow k = 2r \longrightarrow X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n2r} \quad (2.16)$$

$$k \text{ impar} \longrightarrow k = 2r + 1 \longrightarrow X[2r+1] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r+1)} \quad (2.17)$$

Empezaremos el estudio por los valores que ocupan una posición par en la secuencia de coeficientes de la DFT. Separando el sumatorio de la ecuación (2.16) en dos sumatorios tenemos:

$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} + \sum_{n=N/2}^{N-1} x[n] W_{N/2}^{nr}$$

Arreglando el segundo sumatorio para que ambos tengan los mismos límites, se obtiene:

$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} + \sum_{n=0}^{N/2-1} x[n+N/2] W_{N/2}^{r(n+N/2)}$$

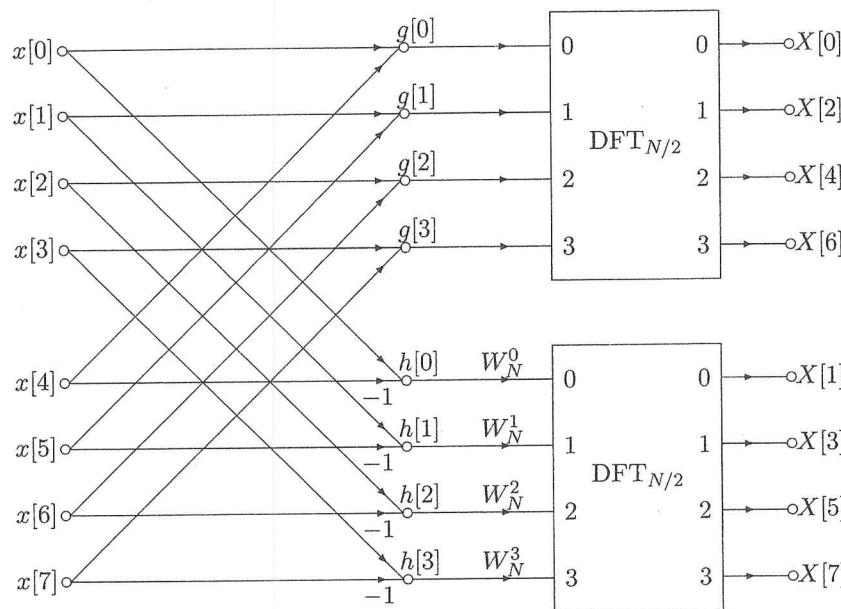


Figura 2.22: Algoritmo FFT de diezmado en frecuencia ($N = 8$): descomposición inicial.

Aplicando la propiedad de periodicidad de las exponentiales⁴

$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} + \sum_{n=0}^{N/2-1} x[n+N/2] W_{N/2}^{rn}$$

$$X[2r] = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2]) W_{N/2}^{nr} \quad (2.18)$$

⁴La exponencial es periódica, de periodo $N/2$, por lo que:

$$W_{N/2}^{r(n+N/2)} = W_{N/2}^{rn}$$

Procediendo de la misma forma con los coeficientes de orden impar, desarrollaremos la ecuación (2.17), obteniendo:

$$X[2r+1] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} W_N^n + \sum_{n=N/2}^{N-1} x[n] W_{N/2}^{nr} W_N^n$$

Arreglando los límites de los sumatorios:

$$X[2r+1] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} W_N^n + \sum_{n=0}^{N/2} x[n+N/2] W_{N/2}^{(n+N/2)r} W_N^{n+N/2}$$

Si aplicamos, ahora, la propiedad de periodicidad en $N/2$ de la exponencial:

$$X[2r+1] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} W_N^n + \sum_{n=0}^{N/2} x[n+N/2] W_{N/2}^{nr} W_N^{n+N/2}$$

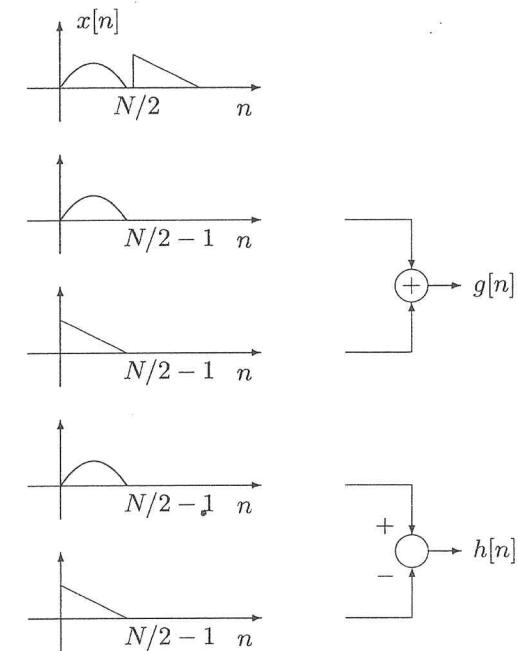


Figura 2.23: Obtención de $g[n]$ y $h[n]$ a partir de $x[n]$.

Teniendo en cuenta que $W_N^{n+N/2} = -W_N^n$:

$$X[2r+1] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} W_N^n + \sum_{n=0}^{N/2} x[n+N/2] W_{N/2}^{nr} (-W_N^n)$$

Finalmente, se obtiene la expresión (2.19) que nos permite obtener los coeficientes impares de $X[k]$ a partir de muestras de $x[n]$

$$X[2r+1] = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2]) W_{N/2}^{nr} W_N^n \quad (2.19)$$

Si en las ecuaciones (2.18) y (2.19) hacemos el cambio

$$x[n] + x[n + N/2] = q[n]$$

y

$$x[n] - x[n + N/2] = h[n]$$

se obtienen dos nuevas ecuaciones (2.20 y 2.21) que nos sirven de base para el desarrollo del grafo que representa el algoritmo de diezmado en frecuencia. Gráficamente, puede observarse en la figura 2.23 cómo se obtendrían $g[n]$ y $h[n]$ a partir de $x[n]$.

$$X[2r] = \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{nr} \quad (2.20)$$

$$X[2r+1] = \sum_{n=0}^{N/2-1} (h[n] W_N^n) W_{N/2}^{nr} \quad (2.21)$$

El grafo para realizar la FFT de 8 puntos con un algoritmo de diezmado en frecuencia sería el de la figura 2.22. De la misma manera que en los algoritmos de diezmado en el tiempo, se podría seguir iterando hasta descomponer $X[k]$ en subsecuencias de 2 elementos.

2.7.4. Consideraciones adicionales sobre la FFT

Una vez vistos los dos tipos de algoritmos para realizar la FFT, en este punto vamos a ocuparnos de una serie de consideraciones prácticas a tener en cuenta a la hora de implementarlos, como:

- Cantidad de memoria utilizada en la implementación del algoritmo, la cual se minimiza con los algoritmos *en el sitio*.

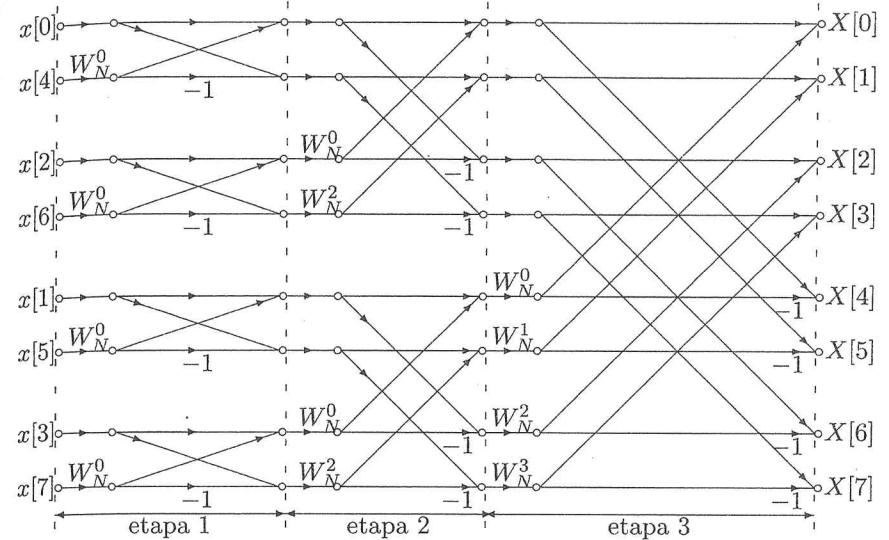


Figura 2.24: Algoritmo FFT de diezmado en el tiempo ($N = 8$) con mariposa simplificada.

- Ordenación de los datos de entrada al algoritmo.
 - Formas alternativas de realizar el algoritmo FFT.
 - Minimización de los cálculos de las exponenciales involucradas en el proceso utilizando distintas estrategias.

Algoritmos en el sitio

El diagrama de flujo de la FFT (figura 2.21) representa el algoritmo a implementar. Dentro de este grafo aparecen una serie de nodos que representan las variables necesarias para realizar el programa, es decir, la cantidad de almacenamiento necesario. En un grafo como el de la figura 2.24 se observa que el algoritmo se divide en $v = \log_2 N$ etapas, en nuestro ejemplo el número de etapas es 3. Cada etapa obtiene N complejos de salida (8 en el ejemplo), a partir de otros N complejos de entrada.

Una forma de almacenar los datos y resultados necesarios para realizar el programa que implemente un algoritmo como el de la figura 2.24 es utilizando vectores que permitan almacenar los valores complejos de las entradas y las salidas de las distintas etapas. Para tener almacenados N complejos de entrada

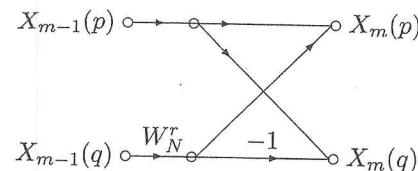


Figura 2.25: Obtención de los elementos p y q para la etapa m .

y otros N de salida en cada etapa se necesitan 2 vectores de complejos para todo el algoritmo, ya que el vector de salida de una etapa se convierte en el vector de entrada de la siguiente, y por tanto, en principio sólo se necesita el vector con los resultados de la etapa anterior y el correspondiente a los resultados que se están calculando en la etapa actual.

Los elementos de cada vector de salida se obtienen como resultado de aplicar sobre los elementos del vector de entrada estructuras *mariposas* como la de la figura 2.25, en la que $X_m(l)$ representa el elemento l -ésimo del vector de salida de la etapa m -ésima. En la primera etapa, X_0 será la secuencia de entrada $x[n]$, mientras que en la última, X_v serán los coeficientes de la DFT.

En la figura 2.25 puede observarse que para obtener los valores de cualquier par de elementos de una etapa ($X_m(p)$ y $X_m(q)$) sólo se necesitan los valores de los elementos que están en la misma posición en el vector de la etapa anterior ($X_{m-1}(p)$ y $X_{m-1}(q)$). Por este motivo, de los dos vectores que necesitábamos inicialmente para implementar el algoritmo, podemos prescindir de uno. Así, el almacenamiento necesario en el programa será solamente un vector de complejos junto con una variable auxiliar. Los cálculos realizados de esta forma reciben el nombre de *cálculos en el sitio*, ya que los resultados se almacenan en el mismo vector que contenía los datos.

Orden de los datos

En el punto anterior hemos visto que en el grafo del algoritmo FFT se utilizan estructuras del tipo de la de la figura 2.25, las cuales nos permiten realizar los *cálculos en el sitio*. Este tipo de estructuras ha sido propiciado por el hecho de que los datos de entrada están ordenados en orden *bit inverso*.

Para explicar el significado de orden *bit inverso* vamos a ver un ejemplo de una secuencia de 8 muestras, la cual hay que ordenar. En esta secuencia de 8 muestras, el orden que ocupa cada una de ellas puede almacenarse con 3 bits

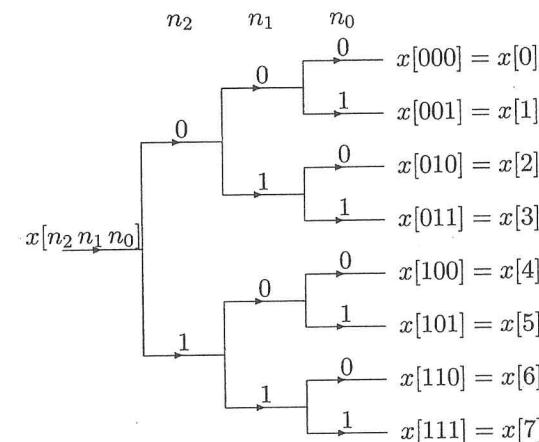


Figura 2.26: Ordenamiento de una secuencia en orden natural.

$x[n_2n_1n_0]$, donde n_0 es el bit menos significativo (LSB), de tal forma que el orden de cada muestra viene dado por una combinación de estos tres bits.

La forma de ordenación más normal es el orden natural, mostrado en la figura 2.26, el cual consiste en ir dividiendo la secuencia en función del valor de estos tres bits, empezando por el más significativo (n_2); es decir, si n_2 es 0 la muestra quedará ordenada en una mitad, si es 1 en la otra. A continuación se repite el proceso en ambas mitades, pero observando ahora el bit n_1 . Finalmente se repite el proceso para todas las subsecuencias obtenidas en el paso anterior teniendo en cuenta el bit n_0 .

Otra forma de ordenar la secuencia es en orden bit inverso, como en la figura 2.27, donde la filosofía de ordenación es la misma que en el orden natural pero el sentido de la misma es del bit menos significativo (n_0) al más significativo (n_2).

En los diagramas de flujo que representan el algoritmo de la FFT (2.18 y 2.21) se observa que antes de empezar el algoritmo, la secuencia debe ser ordenada en orden bit inverso, para poder así implementar un algoritmo *en el sitio*, resultando una secuencia de coeficientes $X[k]$ ordenada en orden natural.

En la figura 2.12 podemos observar que la secuencia $x[n]$ se divide en muestras pares (mitad superior) e impares (mitad inferior), lo cual equivale a una separación de las muestras examinando el LSB (n_0), es decir, las muestras con $n_0 = 0$ (pares) quedarán ordenadas en la mitad superior y las que tienen

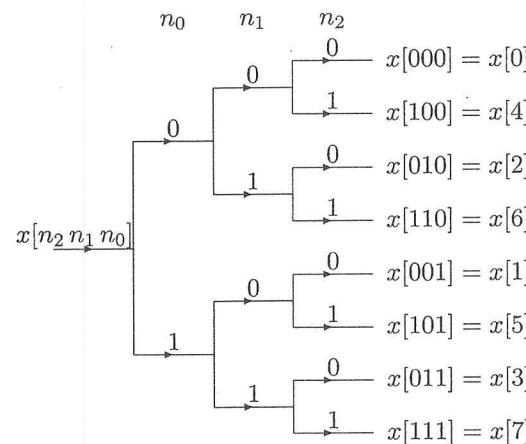


Figura 2.27: Ordenamiento de una secuencia en orden bit inverso.

$n_0 = 1$ (impares) pasará a la mitad inferior. Para las dos subsecuencias obtenidas se debe repetir el mismo proceso (N veces), hasta obtener subsecuencias de un solo elemento (figuras 2.17 y 2.15). Así pues, la ordenación *bit inverso* de la figura 2.27 es equivalente a separar la secuencia en muestras que ocupan lugares pares o impares en la misma.

Formas alternativas

Si observamos las figuras que representan el grafo del algoritmo FFT (figuras 2.18 y 2.21) podemos darnos cuenta que podríamos cambiar el orden de los datos de entrada al algoritmo FFT siempre y cuando, con la nueva ordenación, una muestra siga el mismo proceso, es decir, pase por las mismas ramas que en el algoritmo original. Lo realmente importante es que una muestra sufra los cálculos necesarios para que a la salida se obtengan los coeficientes de la DFT, sin importar el orden en que se realicen los mismos. Así pues, se podría pensar en nuevas ordenaciones de los datos que den lugar a nuevos grafos para representar el algoritmo FFT. En la figura 2.28 puede observarse la forma del grafo cuando el orden de la secuencia de entrada corresponde al orden natural, que es, precisamente, el grafo original presentado por Cooley y Tukey.

Las implementaciones con las muestras de entrada en orden natural son muy útiles en procesos en los que haya que concatenar FFT e FFT⁻¹, por ejemplo un filtrado (figura 2.29). De esta forma se pueden evitar varias ordenaciones

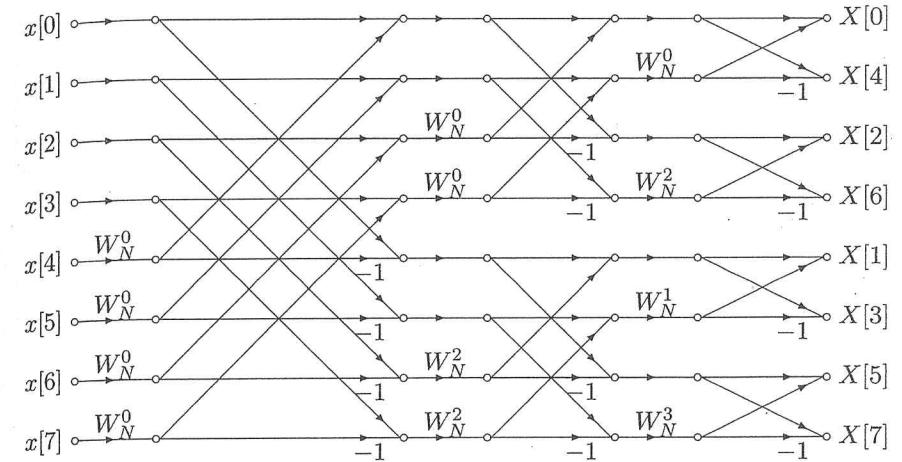


Figura 2.28: Algoritmo FFT de diezmado en el tiempo ($N = 8$) con entrada en orden natural.

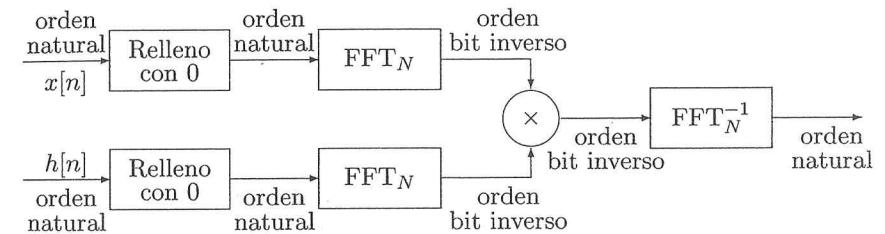


Figura 2.29: Filtrado utilizando FFTs.

de la secuencia, proceso que puede resultar cada vez más lento a medida que N crece.

Además de la estructura de la figura 2.28 existen otras formas derivadas de las distintas ordenaciones de los datos de entrada y los nodos intermedios, de las cuales no vamos a ocuparnos aquí.

Cálculo de coeficientes

Como hemos visto anteriormente, para implementar el algoritmo FFT hace falta una serie de coeficientes exponenciales W_N^r donde $r = 0 \dots (N/2) - 1$, cuyo cálculo es un proceso que debe ser optimizado. Para ello se utilizan dos métodos diferentes:

- Uso de una tabla: calcular previamente todos los coeficientes necesarios para implementar el algoritmo y guardarlos en una tabla. Este método minimiza el tiempo de cálculo, pero necesita memoria para almacenar la tabla.
- Cálculo recursivo: consiste en calcular los coeficientes cuando sean necesarios. En una etapa dada todos los coeficientes necesarios son múltiplos de W_N . Para optimizar se aplica la siguiente fórmula recursiva:

$$W_N^l = W_N W_N^{(l-1)}$$

es decir, para obtener el l -ésimo múltiplo de W_N se necesita esta base y el último coeficiente calculado. Este método minimiza la memoria necesaria optimizando también el tiempo de cálculo. Sin embargo, hay que tener en cuenta que los coeficientes sufren un error de cuantificación al almacenarlos, el cual se irá acumulando al ir aplicando la recursión, haciéndose inaceptable si N crece mucho. Este problema tiene un pequeño arreglo introduciendo puntos de inicialización en la recursión, por ejemplo $W_N^{\frac{N}{4}} = -j$.

2.7.5. Algoritmos con $N \neq 2^v$

Hasta ahora hemos visto únicamente algoritmos rápidos de cálculo de la DFT si la secuencia sobre la que se quiere realizar tiene un número de puntos potencia de 2 ($N = 2^v$). Estos algoritmos pueden generalizarse para valores de $N \neq 2^v$, aunque ya no son tan eficientes. Los pasos a llevar a cabo para implementar el algoritmo, en el caso general, serían los siguientes:

1. Descomponer el número de puntos en factores primos $N = n_1 n_2 \dots n_L$.
2. La DFT de N puntos se divide en n_1 DFTs de N/n_1 combinándolas por medio de unas operaciones auxiliares.
3. Cada DFT de N/n_1 puntos se dividen en n_2 DFTs de $N/(n_1 n_2)$ puntos.

2.7. La FFT

4. El proceso continuaría hasta tener que implementar DFTs de n_L puntos, las cuales habría que realizarlas por el método directo, por eso interesa que n_L sea el factor más pequeño.

El caso óptimo de este algoritmo general sería que todos los factores fueran iguales e igual, a su vez, al factor primo menor, es decir, $n_1 = n_2 = \dots = n_L = 2$; este sería el caso visto en secciones anteriores $N = 2^v$. El caso peor sería que N fuera un número primo, ya que entonces no se podría descomponer en factores y habría que calcular una DFT de N puntos por el método directo.

Vamos a ver un ejemplo que nos ayudará a comprender cómo se extenderían estos algoritmos de diezmado en el tiempo al caso general en que $N \neq 2^v$.

Ejemplo: $N = 3^v$

Tendremos que realizar 3 DFTs de $N/3$ puntos.

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

Separando las muestras en las situadas en $\overset{\circ}{3}, \overset{\circ}{3} + 1$ y $\overset{\circ}{3} + 2$, tenemos⁵:

$$X[k] = \sum_{r=0}^{\frac{N}{3}-1} x[3r] W_N^{3rk} + \sum_{r=0}^{\frac{N}{3}-1} x[3r+1] W_N^{(3r+1)k} + \sum_{r=0}^{\frac{N}{3}-1} x[3r+2] W_N^{(3r+2)k}$$

siendo

$$W_N^{3rk} = W_{\frac{N}{3}}^{rk}$$

tenemos finalmente que:

$$X[k] = \sum_{r=0}^{\frac{N}{3}-1} x[3r] W_{\frac{N}{3}}^{rk} + \left[\sum_{r=0}^{\frac{N}{3}-1} x[3r+1] W_{\frac{N}{3}}^{rk} \right] W_N^k + \left[\sum_{r=0}^{\frac{N}{3}-1} x[3r+2] W_{\frac{N}{3}}^{rk} \right] W_N^{2k} \quad (2.22)$$

En la ecuación (2.22) puede observarse que $\sum_{r=0}^{\frac{N}{3}-1} x[3r] W_{\frac{N}{3}}^{rk}$, $\sum_{r=0}^{\frac{N}{3}-1} x[3r+1] W_{\frac{N}{3}}^{rk}$

y $\sum_{r=0}^{\frac{N}{3}-1} x[3r+2] W_{\frac{N}{3}}^{rk}$, son DFTs de $N/3$ puntos combinadas entre sí por coeficientes exponenciales múltiplos de W_N . Este proceso debe iterarse hasta obtener DFTs simples de 3 puntos. Con esto queda visto que el algoritmo general

⁵La notación $\overset{\circ}{3}$ indica múltiplos de 3.

conserva la misma idea que los algoritmos de diezmado en el tiempo (idem para frecuencia) vistos en secciones anteriores para $N = 2^v$, aunque en nuestro ejemplo la base en vez de 2 es 3. Si los factores en los que se descompone N son diferentes ($n_1 \neq n_2 \neq \dots \neq n_L$), la base será mixta.

2.8. Problemas

- Demuestre las propiedades de la sección 2.3.
- Sea $x[n]$ una señal real de duración N . Se define la señal $x_1[n]$ de la siguiente forma:

$$x_1[n] = \begin{cases} x[n] & 0 \leq n \leq N-1 \\ x[2N-1-n] & N \leq n \leq 2N-1 \end{cases}$$

- Dibuje una señal arbitraria de 3 puntos y su correspondiente $x_1[n]$.
- Si llamamos $X_1[k]$ a la DFT de $2N$ puntos de $x_1[n]$, demuestre que:

$$C_x[k] = e^{j\pi k/2N} X_1[k]$$

es real.

- Encuentre qué simetrías existen en los valores de $C_x[k]$.
- Demuestre que se puede obtener $x[n]$ a partir de $C_x[k]$, $k = 0, \dots, N-1$ e indique cómo.⁶
- Considere que $x[n]$ sea una señal real y par que sea no nula en el intervalo $n = -M, \dots, M$. Indique de qué señal $z[n]$ (no nula de $n = 0, \dots, N-1$) habría de calcularse la DFT _{N} para que los N valores obtenidos constituyan muestras de $X(e^{j\omega})$.
- Demuestre que para una secuencia de N muestras, y su correspondiente DFT de N muestras se cumple:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

- Sean dos señales de duración finita $x[n]$ e $y[n]$ tal que

$$x[n] = \begin{cases} x[n] & \text{si } 0 \leq n \leq 19 \\ 0 & \text{resto} \end{cases} \quad y[n] = \begin{cases} y[n] & \text{si } 0 \leq n \leq 4 \\ 0 & \text{resto} \end{cases}$$

⁶A $C_x[k]$ se le denomina Transformada del coseno.

- a) ¿Cuál es el máximo número posible de valores distintos de 0 en el resultado de la convolución lineal entre $x[n]$ e $y[n]$? b) La convolución circular de 20 puntos de $x[n]$ e $y[n]$ es:

$$x[n] @ y[n] = 15 \quad 0 \leq n \leq 19$$

Los primeros 4 puntos de la convolución lineal de $x[n]$ e $y[n]$ son:

$$x[n] * y[n] = 4 \quad 0 \leq n \leq 3$$

Determine el valor del máximo número posible de puntos de la convolución lineal entre $x[n]$ e $y[n]$.

- Determine el número medio de operaciones por muestra en los métodos solape-suma y solape-almacenamiento usando FFTs.
- Indique cómo generalizar el método de diezmado en frecuencia al caso $N = 3^v$. Indique cuánto vale el número de productos complejos en este caso.
- Demuestre que una DFT de 4 puntos se puede realizar sin productos complejos.
- Desarrolla hasta la FFT₂ el algoritmo de diezmado en frecuencia para $N = 8$ de la figura 2.22.
- Dibuje el diagrama de flujo del algoritmo FFT de diezmado en el tiempo cuando $N = 9$ (ver sección 2.7.5). ¿Cuál será en este caso el orden de los datos de entrada?. ¿Cuántas operaciones por etapa se realizan? ¿Cuántas operaciones totales?.
- Determine el número de operaciones a realizar para implementar el algoritmo FFT de diezmado en el tiempo cuando $N = a^\gamma$ siendo $a, \gamma \in \mathbb{Z}$.