La transformada rápida de Fourier (FFT)
y otros algoritmos para la implementación de la DFT

Existen diversas formas de implementar la transformada discreta de Fourier (DFT). Para estudiar algunas de ellas, considere una DFT de N puntos $X_N(k)$, la cual llamaremos también $X^{(N)}(k)$ por notación y donde $k=0,\ldots,N-1$:

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

1) Implementación directa o por definición

La DFT se puede calcular directamente con la expresión de (1) como una suma utilizando `for-loops` para cada bin de frecuencia, generando N operaciones complejas por cada bin y N^2 operaciones complejas en total. Esta es la implementación más sencilla conceptualmente, pero la menos eficiente.

2) Implementación matricial

Es posible representar la ecuación (1) como un producto entre una matriz y un vector, de modo que $\boldsymbol{X} = \boldsymbol{W_N x}$, donde $\boldsymbol{X}$ y $\boldsymbol{x}$ son vectores columna de N x 1 y A es una matriz cuadrada de N x N. Este producto es equivalente a $X(k) = \sum_{n=1}^{N} W_N^{kn} x_n$, donde W_N^{kn} es el elemento (k,n) de la matriz dado por $W_N^{kn} = e^{-j2\pi kn/N}$. Si llamamos a los coeficientes $W_N = e^{-j2\pi/N}$ entonces $W_N^{kn} = (W_N)^{kn}$. Es posible demostrar que $W_N^{k+N} = W_N^k$ y que $W_N^{k+N/2} = -W_N^k$, lo que explica la simetría de la matriz $\boldsymbol{W_N}$

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

$$\boldsymbol{X} = \boldsymbol{W_N x} \tag{3}$$

$$\boldsymbol{x} = \boldsymbol{W_N^{-1} X} = \frac{1}{N} \boldsymbol{W_N^* X} \tag{4}$$

Aunque, en teoría, la complejidad numérica de esta forma no varía significativamente de la forma directa, este método presenta ventajas notables producto del trabajo en matrices. Si $\boldsymbol{W_N}$ se calcula offline, el cálculo de la DFT y su inversa sólo requieren la multiplicación de una matriz por un vector.

3) Divide y conquista

Esta implementación descompone una DFT de N puntos como la suma de dos DFT de N/2 puntos, lo que reduce la complejidad numérica casi a la mitad y es la base conceptual del algoritmo FFT. Para derivar esta versión, podemos partir por dividir el cálculo para las muestras pares e impares

$$X^{(N)}(k) = \sum_{\substack{n=0 \\ n\,par}}^{N-1} x(n)e^{-j2\pi kn/N} + \sum_{\substack{n=0 \\ n\,impar}}^{N-1} x(n)e^{-j2\pi kn/N} \tag{5}$$

$$= \sum_{m=0}^{N/2-1} x(2m)e^{-j2\pi k2m/N} + \sum_{m=0}^{N/2-1} x(2m+1)e^{-j2\pi k(2m+1)/N} \tag{6}$$

$$= \sum_{m=0}^{N/2-1} x(2m)e^{-j2\pi km/(N/2)} + e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} x(2m+1)e^{-j2\pi km/(N/2)} \tag{7}$$

Si llamamos a la serie de muestras par $x_0(m) = x(2m)$ y a la impar $x_1(m) = x(2m+1)$ entonces

$$X^{(N)}(k) = \sum_{m=0}^{N/2-1} x_0(m)e^{-j2\pi km/(N/2)} + e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} x_1(m)e^{-j2\pi km/(N/2)} \tag{8}$$

$$X^{(N)}(k) = X_0^{(N/2)}(k) + e^{-j2\pi k/N} X_1^{(N/2)}(k) \tag{9}$$

Es posible simplificar esta expresión observando las simetrías del sistema, donde $X_0^{(N/2)}$ y $X_1^{(N/2)}$ son periódicas cada N/2. Si llamamos a los coeficientes $W_N = e^{-j2\pi/N}$ y notando además que cada DFT de N/2 puntos es periódica cada N/2 puntos y que $W_N^{k+N/2} = -W_N^k$, se puede obtener que

$$\begin{aligned} X^{(N)}(k) &= X_0^{(N/2)}(k) + W_N^k X_1^{(N/2)}(k) \\ X^{(N)}(k+N/2) &= X_0^{(N/2)}(k) - W_N^k X_1^{(N/2)}(k) \end{aligned} \tag{10}$$

para $k=0,\ldots,N/2-1$. Esta transformación reduce la complejidad numérica de la DFT de N^2 a $N^2/2+N/2$. La representación gráfica de (10) se denota en la figura 1.

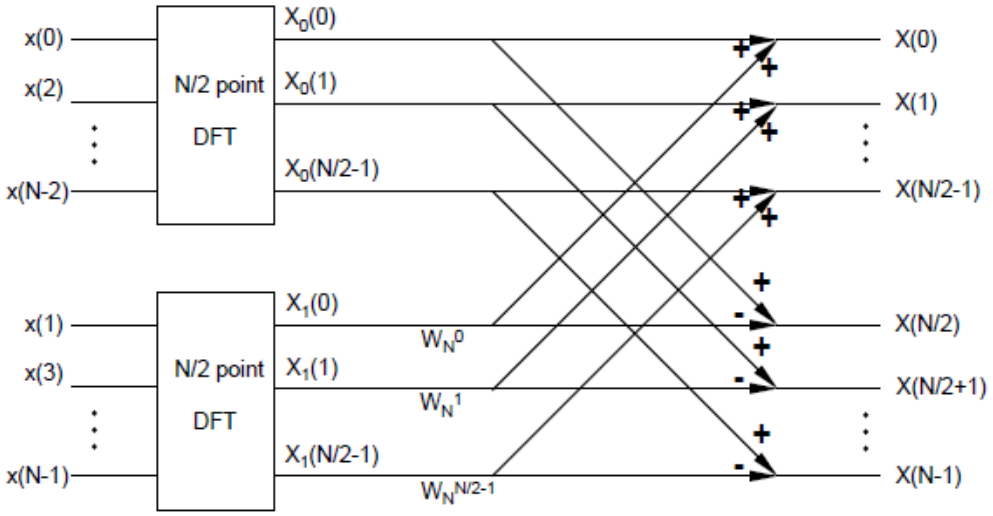


Figura 1: Cálculo de la DFT usando reducción de orden.
La DFT de N puntos $X(k)$ se obtiene mediante dos N/2 DFTs $X_0(k)$ y $X_1(k)$.

4) Transformada rápida de Fourier (FFT)

El algoritmo FFT entrega los mismos resultados que (1), pero optimiza el cómputo dividiendo el problema en cálculos de DFT de menor orden y una estructura recursiva. Para derivar una de las versiones más comunes del algoritmo FFT llamado Radix-2, asumimos que $N=2^m$. La división de la DFT de N puntos a dos de N/2 presentada por el algoritmo divide y conquista se puede volver a realizar recursivamente en cada bloque de la figura 1. Para ilustrar este punto, la reducción de orden se vuelve a aplicar sobre cada DFT de N/2 puntos en la figura 2. Es posible seguir esta reducción hasta llegar al bloque más pequeño posible de una DFT de dos puntos, a cual se obtiene directamente de la ecuación (1)

$$\begin{aligned} X^{(2)}(0) &= x[0] + x[1] \\ X^{(2)}(1) &= x[0] - x[1] \end{aligned} \tag{11}$$

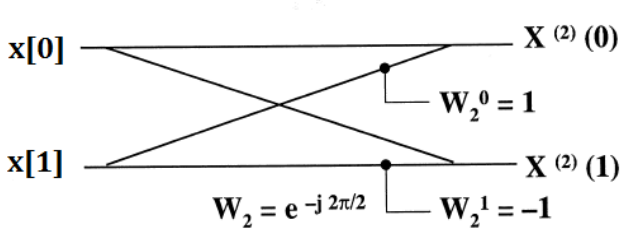


Figura 2: Cálculo de una FFT de 2 puntos

Esta implementación recursiva se conoce como FFT radix 2 y es una de las formas más comunes y el orden de su complejidad numérica baja de N^2 a $N\log_2(N)$. El número total de multiplicaciones complejas es $(N/2)\log_2(N)$ ya que la segunda mitad de la ecuación (10) no requiere multiplcaciones adicionales, sólo un cambio de signo en la suma.

La figura 3 presenta el esquema de una estructura de reducción de orden en la DFT, la figura 4 el caso de una FFT de 8 puntos, y la figura 5 la complejidad numérica en función al número de puntos N.

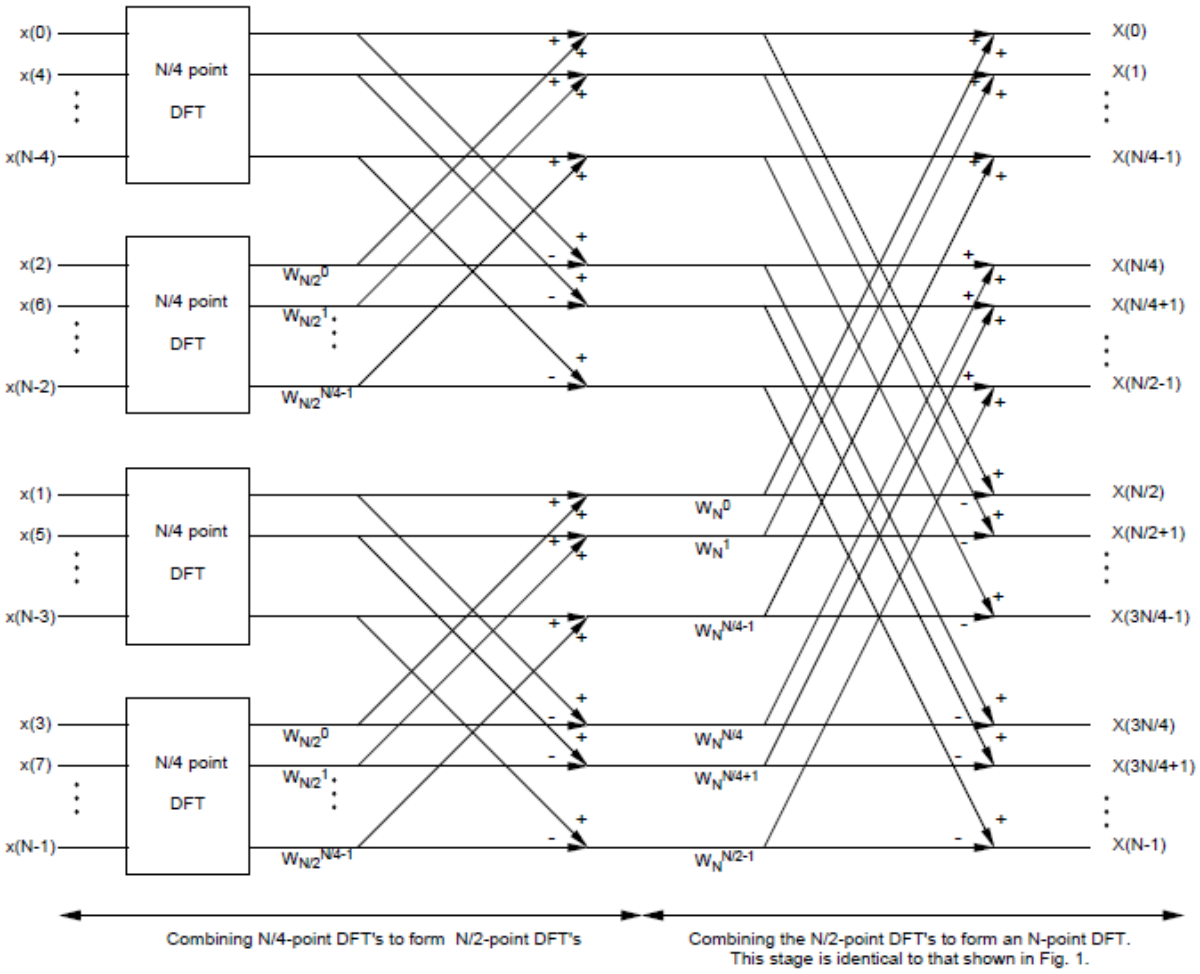
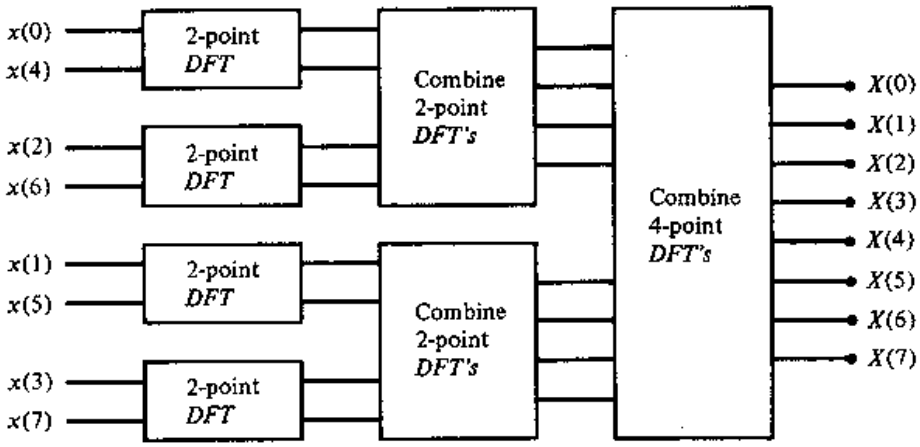


Figura 3: Cálculo de la DFT usando reducción de orden.
La DFT de N puntos $X(k)$ se obtiene mediante cuatro N/4 DFTs.

(a)



(b)

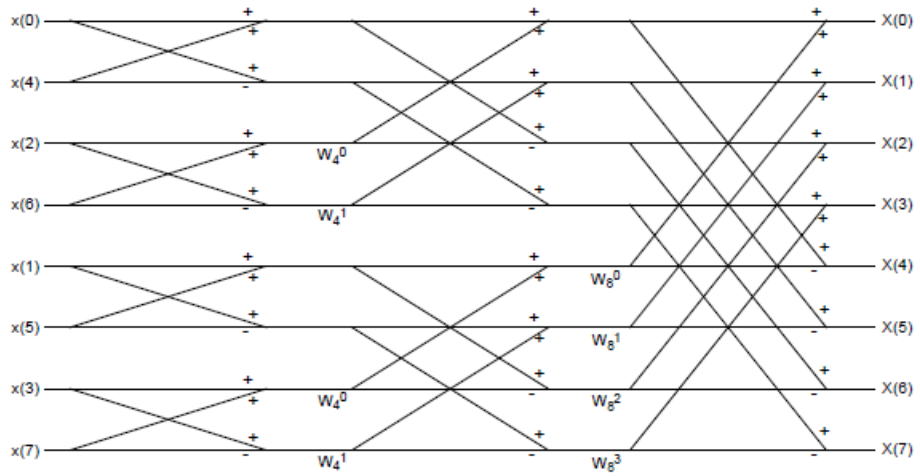


Figura 4: Cálculo de una FFT de 8 puntos usando el esquema radix 2.
(a) Esquema general (b) Operaciones aritméticas involucradas

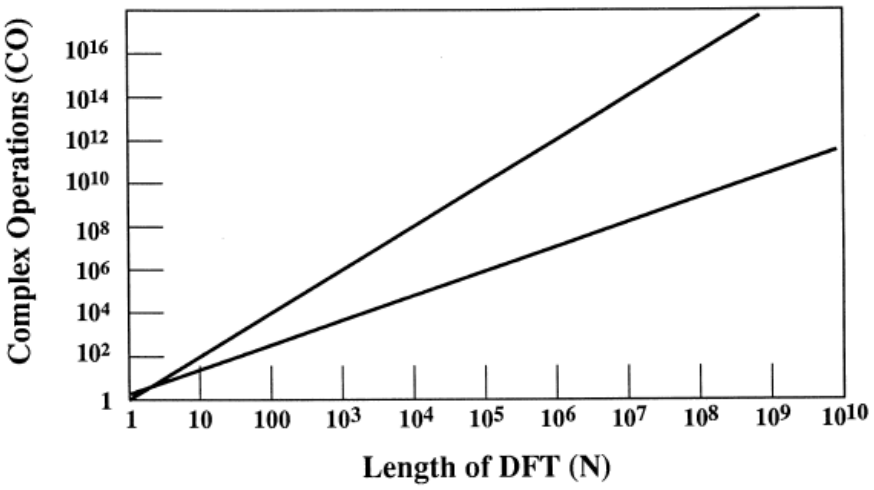


Figura 5: Complejidad numérica del algoritmo FFT radix-2 en función de N