Transformada Rápida de Fourier

Michael Joel Spilsbury[1] y Armando Euceda[2]

[1] *Universidad Nacional Autónoma de Honduras en el Valle de Sula, mail: michael.spilsbury@unah.edu.hn*
[2] *Universidad Nacional Autónoma de Honduras, mail: aeunah@gmail.com*

Recibido: 10 de Mayo del 2016 / Aceptado: 10 de Agosto del 2016

Resumen

The following article is a brief introduction of fast Fourier transform (FFT), which is an algorithm for the calculation of the discrete Fourier transform that reduce the runtime of the program for calculating the Fourier transform greatly. Since 1965[6], when James W. Cooley and John W. Tukey published this algorithm, its use has expanded quickly and personal computers have generated an explosion of further FFT applications. Examples of the FFT application are circuit design, spectroscopy, crystallography, signal processing and communications, images, etc.

Keywords: Fast Fourier transform, algorithm, discrete Fourier transform, Cooley, Tukey.

El siguiente artículo es una breve introducción de la transformada rápida de Fourier (FFT por sus siglas en inglés), el cual es un algoritmo para el cálculo de la transformada discreta de Fourier que reduce el tiempo de ejecución del programa en gran medida. Desde 1965[6], cuando James W. Cooley y John W. Tukey publicaron dicho algoritmo, su uso se ha expandido rápidamente y las computadoras personales han impulsado una explosión de aplicaciones adicionales de la FFT. Algunos ejemplos de la aplicación de la FFT son diseño de circuitos, espectroscopia, cristalografía, procesamiento de señales y comunicaciones, imágenes, etc.

Palabras clave: Transformada rápida de Fourier, algoritmo, transformada discreta de Fourier, Cooley, Tukey.

I. INTRODUCCIÓN

La transformada rápida de Fourier (FFT por sus siglas en ingles—*fast Fourier transform*) tiene una amplia e interesante historia, pero debido a que el tiempo no lo permite no se darán detalles en este documento, pero si esta interesado puede consultar [4], [7], [8] y [13]. En términos simples, la FFT es un método eficiente para calcular la transformada discreta de Fouier. En palabras de G. D. Bergland:

> "El algoritmo de la transformada rápida de Fourier puede reducir el tiempo necesario para encontrar la transformada discreta de Fourier de varios minutos a menos de un segundo, y puede reducir los costos de varios dolares a varios centavos."[2]

Después de la publicación del artículo de Cooley and Tukey en 1965 [6], su amplia aceptación y difusión se considera hasta la fecha (véase [2], [5], [9], [10], [11], [12], [14], [15]). En dicho artículo se detalla la esencia del algoritmo y su desarrollo.

El algoritmo de Cooley-Tukey hace a la FFT extremadamente útil al reducir el número de cálculos a partir de algo del orden de n^2 a $n \log n$, lo que obviamente ofrece una enorme reducción en el tiempo de cálculo. Es tan útil, de hecho, que la FFT hizo que la revista *Computing in Science & Engineering* la ubicara en la lista de los 10 principales algoritmos en un artículo que indica que el algoritmo es, "quizás, el algoritmo más ubicuo en uso hoy en día." El método usado en el algoritmo de Cooley-Tukey se puede aplicar a otras transformaciones ortogonales tales como la Hadamard, Hartley, y Haar[11].

A continuación se presentan dos metódos para la comprensión del algoritmo, uno gráfico y otro analítico, ambos considerando que el número de muestras es $N = 2^\gamma$, para γ entero.

II. Desarrollo teórico

La interpretación de los resultados de la transformada rápida de Fourier no requiere una enseñanza bien fundada en el propio algoritmo, sino más bien un conocimiento profundo de la transformada discreta de Fourier. Esto es consecuencia del hecho de que la FFT es simplemente un algoritmo (es decir, un método particular para realizar una serie de cálculos) que puede calcular la transformada discreta de Fourier mucho más rápidamente que otros algoritmos disponibles. Por esta razón, la discusión de la FFT se enfoca sólo al aspecto computacional del algoritmo.

Un ejemplo de factorización matricial sencillo se utiliza para justificar intuitivamente el algoritmo FFT. Las matrices factorizadas se representan alternativamente mediante gráficos de flujo de señales. A partir de estos gráficos, se construye la lógica de un programa de FFT para computadora. A continuación, se presentan los desarrollos teóricos de diversas formas del algoritmo FFT.

A. Formulación matricial

Considere la transformada discreta de Fourier:

$$X(n) = \sum_{k=0}^{N-1} x_0(k) e^{-j2\pi nk/N}, \qquad n = 0, 1, \ldots, N-1 \tag{1}$$

Nótese que la ecuación (1) describe el cálculo de N ecuaciones. Por ejemplo, si $N = 4$ y tomamos:

$$W = e^{-j2\pi/N} \tag{2}$$

entonces la ecuación (1) puede ser escrita como:

$$\begin{aligned} X(0) &= x_0(0)W^0 + x_0(1)W^0 + x_0(2)W^0 + x_0(3)W^0 \\ X(1) &= x_0(0)W^0 + x_0(1)W^1 + x_0(2)W^2 + x_0(3)W^3 \\ X(2) &= x_0(0)W^0 + x_0(1)W^2 + x_0(2)W^4 + x_0(3)W^6 \\ X(3) &= x_0(0)W^0 + x_0(1)W^3 + x_0(2)W^6 + x_0(3)W^9 \end{aligned} \tag{3}$$

Las ecuaciones (3) pueden ser representadas de forma matricial como:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{4}$$

o de forma más compacta:

$$\boldsymbol{X(n) = W^{nk} x_0(k)} \tag{5}$$

Denotando el tipo de letra, en negrita y cursiva, una matriz.

Examinando la ecuación (4) se observa que debido a que $\boldsymbol{W}$ y posiblemente $\boldsymbol{x_0(k)}$ son complejos, son necesarias N^2 multiplicaciones complejas y $(N)(N-1)$ adiciones complejas para desarrollar el cálculo matricial requerido. La FFT debe su éxito al hecho de que el algoritmo reduce el número de multiplicaciones y adiciones requeridas en el cálculo de la ecuación (4). Ahora vamos a discutir, a nivel intuitivo, cómo se lleva a cabo esta reducción.

B. Desarrollo intuitivo

Para ilustrar el algoritmo FFT, es conveniente escoger el número de muestras de $x_0(k)$ de acuerdo a la relación $N = 2^\gamma$, donde γ es un entero. En otros desarrollos se puede remover esta restricción (véase [1] y [3]). Recordemos que la ecuación (4) resulta de tomar $N = 4 = 2^\gamma = 2^2$; por lo tanto, podemos aplicar la FFT para el cálculo de la ecuación (4).

El primer paso en el desarrollo del algoritmo FFT para este ejemplo es reescribir la ecuación (4) como:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{6}$$

La matriz de la ecuación (6) deriva de la ecuación (4) usando la relación $W^{nk} = W^{nk \bmod(N)}$. Recuerde que $[nk \bmod(N)]$ es el resto o residuo de la división de nk por N; entonces si $N = 4$, $n = 2$ y $k = 3$, entonces:

$$W^6 = W^2 \tag{7}$$

porque

$$\begin{aligned} W^{nk} = W^6 &= \exp\left[\left(\frac{-j2\pi}{4}\right)6\right] = \exp\left(-j3\pi\right) \\ &= \exp\left(-j\pi\right) = \exp\left[\left(\frac{-j2\pi}{4}\right)2\right] \\ &= W^2 = W^{nk \bmod(N)} \end{aligned} \tag{8}$$

El segundo paso en el desarrollo es factorizar la matriz cuadrada de la ecuación (6) como sigue:

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{9}$$

El método de factorización esta basado en la teoría del algoritmo FFT que se desarrollará más adelante. Por el momento, es suficiente mostrar que el producto de las matrices cuadradas de la ecuación (9) resulta en la matriz cuadrada de la ecuación (6) con la excepción de que las filas 1 y 2 han sido intercambiadas (las filas están numeradas como 0, 1, 2 y 3). Nótese que este intercambio ha sido tomado en cuenta en la ecuación (9) reescribiendo el vector columna $\boldsymbol{X(n)}$; denotando al vector con intercambio de filas como:

$$\overline{\boldsymbol{X(n)}} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \tag{10}$$

esta factorización es la clave del algoritmo FFT.

Aceptando el hecho que la ecuación (9) es correcta, a pesar que los resultados están *desorganizados*, se debe

examinar el número de multiplicaciones requeridas para calcular la ecuación. Primero, tomemos:

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{11}$$

Entonces, el vector columna $\boldsymbol{x_1(k)}$ es igual al producto de las dos matrices a la derecha en la ecuación (9). El elemento $x_1(0)$ es calculado con una multiplicación compleja y una suma compleja (W^0 no es reducido a la unidad para desarrollar el resultado general).

$$x_1(0) = x_0(0) + W^0 x_0(2) \tag{12}$$

El elemento $x_1(1)$ también es determinado por una multiplicación y una adición compleja. Solo una adición compleja es requerida para calcular $x_1(2)$. Esto debido a que $W^0 = -W^2$; por lo tanto,

$$\begin{aligned} x_1(2) &= x_0(0) + W^2 x_0(2) \\ &= x_0(0) - W^0 x_0(2) \end{aligned} \tag{13}$$

donde la multiplicación compleja $W^0 x_0(2)$ ya ha sido determinada al calcular $x_1(0)$ [ecuación (12)]. Por la misma razón, $x_1(3)$ es calculada por una adición compleja sin multiplicación. El vector intermedio $\boldsymbol{x_1(k)}$ es determinado por cuatro adiciones complejas y dos multiplicaciones complejas.

Siguiendo el cálculo de la ecuación (9), del resultado anterior obtenemos:

$$\begin{aligned} \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} &= \begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} \\ &= \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \end{aligned} \tag{14}$$

El término $x_2(0)$ es determinado por una multiplicación y una adición compleja:

$$x_2(0) = x_1(0) + W^0 x_1(1) \tag{15}$$

El elemento $x_2(1)$ es calculado por una adición debido a que $W^0 = -W^2$. Similarmente, $x_2(2)$ se determina por una multiplicación y una adición compleja, y $x_2(3)$ por solo una adición.

El cálculo de $\overline{\boldsymbol{X(n)}}$ por medio de la ecuación (9) requiere un total de cuatro multiplicaciones complejas y ocho adiciones complejas. El cálculo de $\boldsymbol{X(n)}$ por medio de la ecuación (4) requiere 16 multiplicaciones complejas y 12 adiciones complejas. Nótese que el proceso de factorización matricial introduce ceros en las matrices factorizadas y, como resultado, reduce el número requerido de multiplicaciones. Para este ejemplo, el proceso de factorización matricial redujo el número de multiplicaciones por un factor de dos. Debido a que el tiempo de cálculo se rige en gran medida por el número de multiplicaciones requeridas, observamos la razón de la eficiencia del algoritmo FFT.

Para $N = 2^\gamma$, el algoritmo FFT es entonces un simple procedimiento de factorizar una matriz de $N \times N$ en γ matrices (cada una de $N \times N$), tal que cada una de las matrices factorizadas tiene la propiedad especial de minimizar el número de multiplicaciones y adiciones complejas. Si generalizamos el resultado del ejemplo previo, se observa que la FFT requiere $N\gamma/2 = 4$ multiplicaciones *complejas* y $N\gamma = 8$ adiciones *complejas*, mientras que el método directo [ecuación (4)] requiere N^2 multiplicaciones *complejas* y $N(N-1)$ adiciones *complejas*. Si asumimos que el tiempo de computo es proporcional al número de multiplicaciones, entonces la relación aproximada del tiempo de computo directo al de la FFT esta dada por:

$$\frac{N^2}{N\gamma/2} = \frac{2N}{\gamma} \tag{16}$$

donde si $N = 1024 = 2^{10}$ se tiene una reducción de computo de más de 200 a 1. La Figura 1 ilustra la relación entre el número de multiplicaciones requeridas usando el algoritmo FFT comparado con el número de multiplicaciones usando el método directo.

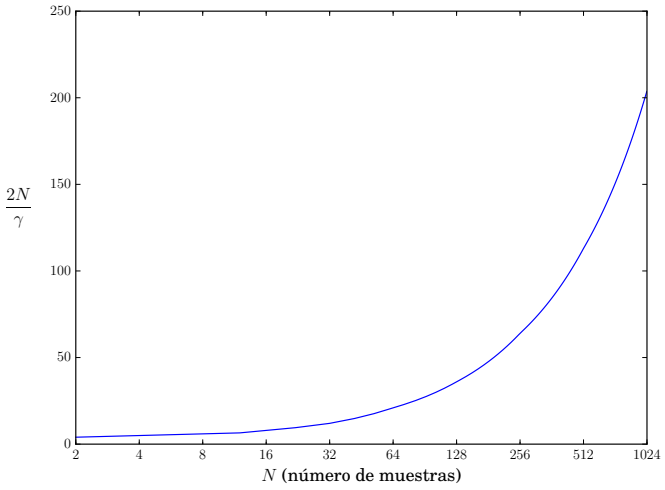

Figura 1: *Relación de multiplicaciones entre el cálculo directo y la FFT*

El proceso de factorización matricial introduce una discrepancia. Recordemos que el cómputo de la ecuación (9) arroja $\overline{\boldsymbol{X(n)}}$ en lugar de $\boldsymbol{X(n)}$; es decir,

$$\overline{\boldsymbol{X(n)}} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \text{ en lugar de } \boldsymbol{X(n)} = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \tag{17}$$

Este reordenamiento es inherente en el proceso de factorización matricial y es un problema menor ya que es fácil generalizar una técnica para *reorganizar* $\overline{\boldsymbol{X(n)}}$ y

obtener $\boldsymbol{X(n)}$.

Reescribiendo $\overline{\boldsymbol{X(n)}}$ con el reemplazo del argumento $\boldsymbol{n}$ por su equivalente binario:

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \text{ se cambia por } \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \tag{18}$$

Observar que si a los argumentos binarios de la ecuación (18) se les gira o hace la inversión de bit (p. ej., 01 se cambia por 10, 10 por 01, etc.), entonces:

$$\overline{\boldsymbol{X(n)}} = \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \text{ cambia a } \begin{bmatrix} X(00) \\ X(01) \\ X(10) \\ X(11) \end{bmatrix} = \boldsymbol{X(n)} \tag{19}$$

es fácil desarrollar un resultado general para reorganizar la FFT.

Para N mayor que 4, es complicado describir el proceso de factorización matricial análogo a la ecuación (9). Por esta razón, interpretamos (9) de una forma gráfica. Usando esta formulación gráfica, podemos describir suficientes generalidades para desarrollar un gráfico de flujo para un programa de computadora.

C. Diagrama de flujo de señal

Ahora representaremos la Ec. (9) en un diagrama de flujo de señal ilustrado en la Figura 2. Como se muestra, se representa el vector de datos o arreglo $\boldsymbol{x_0(k)}$ por una columna vertical de nodos a la izquierda del diagrama. El segundo arreglo vertical de nodos es el vector $\boldsymbol{x_1(k)}$ calculado en la ecuación (11), y el siguiente arreglo vertical corresponde al vector $\boldsymbol{x_2(k)} = \overline{\boldsymbol{X(n)}}$, ecuación (14). En general, habrán γ arreglos computacionales donde $N = 2^\gamma$.

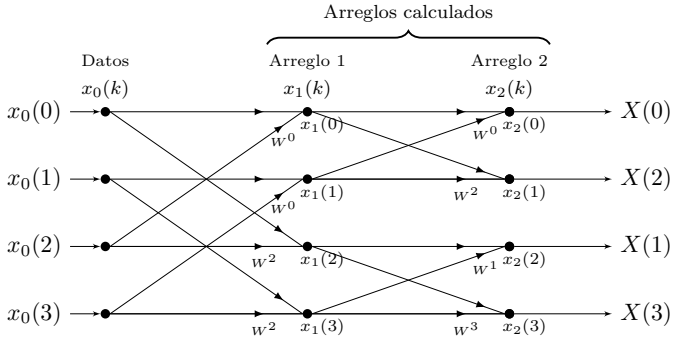

Figura 2: *Diagrama de flujo de señal para la FFT, $N = 4$*

El diagrama de flujo de señal es interpretado como sigue. A cada nodo entran dos líneas representando trayectorias de *transmisión* de los nodos previos. Una trayectoria transmite o lleva una cantidad desde un nodo en un arreglo, multiplica la cantidad por W^p, e ingresa el resultado en el nodo del siguiente arreglo. El factor W^p se encuentra cerca de la punta de la flecha de la trayectoria de transmisión; ausencia de este factor implica que $W^p = 1$. Los resultados que ingresan a un nodo de dos vías de transmisión se combinan de forma aditiva.

Para ilustrar la interpretación del diagrama de flujo de señal, considere el nodo $x_1(2)$ en la Figura 2. De acuerdo a las reglas para la interpretación del diagrama,

$$x_1(2) = x_0(0) + W^2 x_0(2) \tag{20}$$

que es simplemente la ecuación (13). Cada nodo del diagrama es expresado de forma similar.

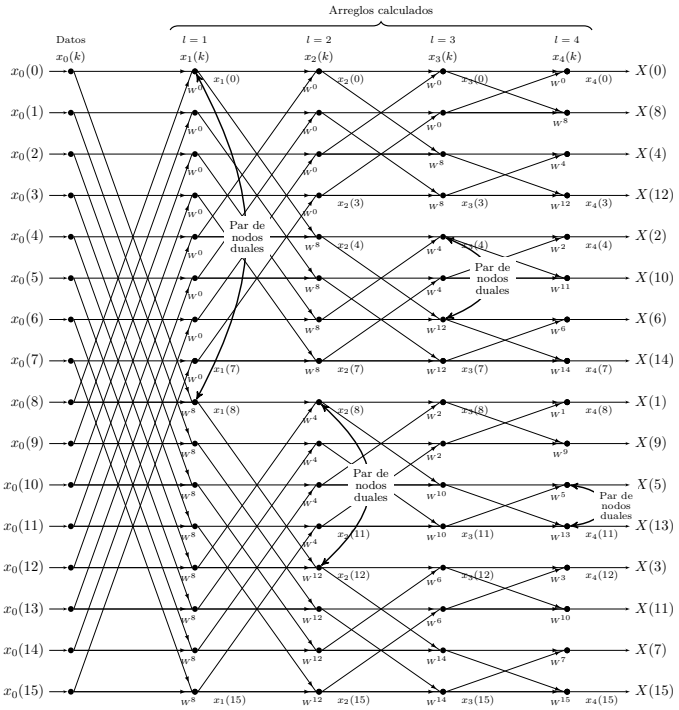

Figura 3: *Ejemplo de nodos duales*

El diagrama de flujo de señal es entonces un método conciso para la representación de los cálculos requeridos en la matriz factorizada del algoritmo FFT de la ecuación (9). Cada columna calculada en el diagrama corresponde a una matriz factorizada; γ arreglos verticales de N puntos cada uno ($N = 2^\gamma$) son requeridos. El uso de esta representación gráfica nos permite fácilmente describir el proceso de factorización matricial para valores grandes de N.

Se muestra en la Figura 3 el diagrama de flujo de señal para $N = 16$. Con un diagrama de flujo de este tamaño, es posible desarrollar propiedades generales relativas al proceso de factorización matricial y así proporcionar un marco para el desarrollo de un diagrama de flujo para computadora de la FFT.

D. Nodos duales

Por inspección de la Figura 3 se observa que en cada arreglo siempre podemos encontrar dos nodos cuyas tra-

yectorias de transmisión de entrada provienen del mismo par de nodos del arreglo anterior. Por ejemplo, los nodos $x_1(0)$ y $x_1(8)$ se calculan a partir de los nodos $x_0(0)$ y $x_0(8)$. Nótese que los nodos $x_0(0)$ y $x_0(8)$ no entran en el cálculo de cualquier otro nodo. Definimos a dos nodos con estas características como un *par de nodos duales*.

Debido a que el cálculo de un par de nodos duales es independiente de otros nodos, es posible llevar a cabo el cálculo *en la misma dirección de memoria*. Para ilustrar esto, obsérvese que a partir de la Figura 3 podemos calcular de forma simultánea $x_1(0)$ y $x_1(8)$ en términos de $x_0(0)$ y $x_0(8)$ y devolver los resultados a las direcciones de memoria previamente ocupadas por $x_0(0)$ y $x_0(8)$. Los requerimientos de memoria son entonces limitados al arreglo de datos $\boldsymbol{x_0(k)}$ solamente. A medida que cada arreglo es calculado, los resultados son devueltos a este arreglo.

Separación de nodos duales. Analicemos la separación (medida verticalmente en términos del indice k) entre un par de nodos duales. A lo siguiente refiérase a la Figura 3. En el arreglo $l = 1$, un par de nodos duales, por ejemplo $x_1(0)$ y $x_1(8)$, están separados por $k = 8 = N/2^l = N/2^1$. En el arreglo $l = 2$, un par de nodos duales, entre ellos $x_2(8)$ y $x_2(12)$, están separados por $k = 4 = N/2^l = N/2^2$. De forma similar, un par de nodos duales, siendo estos $x_3(4)$ y $x_3(6)$, en el arreglo $l = 3$ están separados por $k = 2 = N/2^l = N/2^3$; y en el arreglo $l = 4$, un par de nodos duales, tomando $x_4(10)$ y $x_4(11)$, están separados por $k = 1 = N/2^l = N/2^4$.

Generalizando, se observa que la separación entre nodos duales en el arreglo l esta dado por $N/2^l$. Así, si se considera un nodo particular $x_l(k)$, entonces su nodo dual es $x_l(k + N/2^l)$. Esta propiedad nos permite identificar fácilmente un par de nodos duales.

Cálculo de nodos duales. El cálculo del par de nodos duales requiere solo una multiplicación compleja. Para clarificar este punto, considere el nodo $x_2(8)$ y su dual $x_2(12)$, como se ilustra en la Figura 3. Las trayectorias de transmisión derivadas del nodo $x_1(12)$ son multiplicadas por W^4 y W^{12} previo a ingresar en los nodos $x_2(8)$ y $x_2(12)$, respectivamente. Es importante notar que $W^4 = -W^{12}$ y que solo una multiplicación es requerida porque el mismo dato $x_1(12)$ se debe multiplicar por estos términos. En general, si el factor multiplicador en un nodo es W^p, entonces el factor multiplicador en el nodo dual es $W^{p+N/2}$. Debido a que $W^p = -W^{p+N/2}$, solo una multiplicación es requerida en el cálculo de un par de nodos duales. El cálculo de cualquier par de nodos duales esta dado por el par de ecuaciones:

$$\begin{aligned} x_l(k) &= x_{l-1}(k) + W^p x_{l-1}(k + N/2^l) \\ x_l(k + N/2^l) &= x_{l-1}(k) - W^p x_{l-1}(k + N/2^l) \end{aligned} \qquad (21)$$

En el cálculo de un arreglo, normalmente se comienza con el nodo $k = 0$ y secuencialmente se continua calculando en el arreglo, utilizando el par de ecuaciones de la

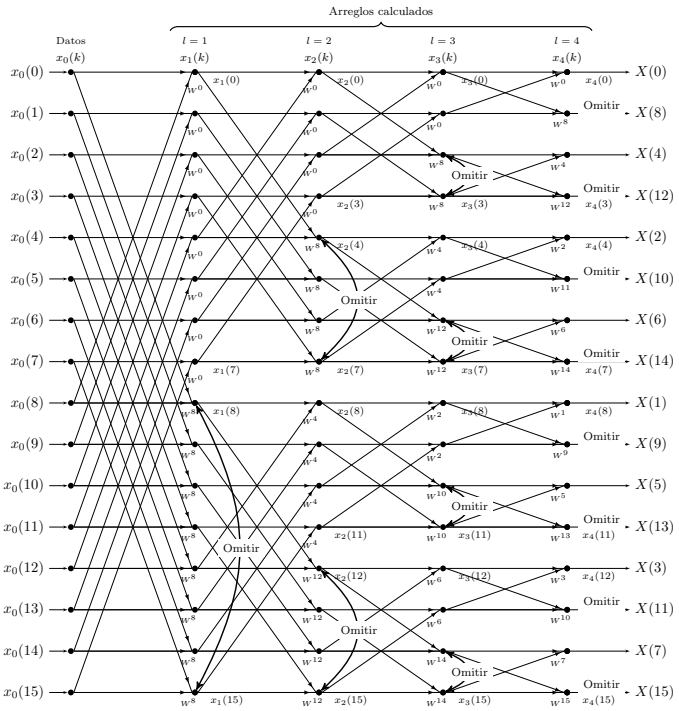

Figura 4: *Ejemplo de nodos a omitir*

ecuación (21). Como se apunto previamente, el dual de cualquier nodo en el l-ésimo arreglo está siempre a $N/2^l$ nodos en el arreglo. Debido a que el espaciado es $N/2^l$, entonces se concluye que se deben *omitir* después de cada $N/2^l$ nodos, los siguientes $N/2^l$ nodos. Para ilustrar este punto, considere el arreglo $l = 2$ en la Figura 4. Si se comienza con el nodo $k = 0$, entonces de acuerdo a lo expuesto anteriormente, el nodo dual está localizado en $k = N/2^2 = 4$, que puede ser verificado por inspección en la Figura 4. Continuando en el arreglo, se observa que el nodo dual está siempre a 4 nodos en el arreglo hasta que se alcanza el nodo 4. En este punto, tenemos un conjunto de nodos calculados indirectamente, es decir, tenemos los duales de los nodos $k = 0, 1, 2$ y 3. Es necesario *omitir* los nodos $k = 4, 5, 6$ y 7. Los nodos $8, 9, 10$ y 11 siguen la convención original del nodo dual localizado a 4 nodos en el arreglo. En general, si se trabaja de arriba hacia abajo en el l-ésimo arreglo, entonces con la ecuación (21) se calcularan los primero $N/2^l$ nodos, se omitirán los siguientes $N/2^l$ y así sucesivamente hasta que se alcance el nodo con indice mayor a $N - 1$.

E. Determinación de W^p

Basados en lo argumentado anteriormente, se han definido las propiedades de cada arreglo con excepción del valor p en la ecuación (21). El valor de p esta determinado por (a) la representación del índice k en forma binaria con γ bits, (b) escalar o deslizar este número binario $\gamma - l$ bits a la derecha, rellenando las posiciones de bits liberados a la izquierda con ceros, y al final (c) invertir el orden de los bits. El número con bits invertidos es el término p.

Para ilustrar este procedimiento, refiérase a la Figura

4 y considere el nodo $x_3(8)$. Debido a que $\gamma = 4$, $k = 8$ y $l = 3$, entonces k en binario es 1000. Se escala este número $\gamma - l = 4 - 3 = 1$ posición a la derecha y rellenamos con ceros; el resultado es 0100. Entonces se invierte el orden de los bits para obtener 0010 o el entero 2. El valor de p es entonces 2.

Consideremos el siguiente procedimiento para implementar la operación de inversión de bit. Sabemos que un número binario, es decir $b_3b_2b_1b_0$, puede ser escrito en base 10 como $b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$. El número con bits invertidos que se trata de encontrar está dado por $b_0 \times 2^3 + b_1 \times 2^2 + b_2 \times 2^1 + b_3 \times 2^0$. Si se describe una técnica para determinar los bits (del número binario) b_3, b_2, b_1 y b_0, entonces se habrá definido la operación de inversión de bit.

Ahora, asuma que M es un número decimal igual a $b_3b_2b_1b_0$. Divida M por 2, tome la parte entera, y este resultado truncado multiplíquelo por 2. Entonces calcule $(b_3b_2b_1b_0) - 2(b_3b_2b_1)$. Si el bit b_0 es 0, entonces esta diferencia es cero porque la división por 2, truncamiento, y subsecuente multiplicación por 2 no modifica M. Sin embargo, si el bit b_0 es 1, el truncamiento cambia el valor de M y la diferencia anterior no es cero. Mediante esta técnica se puede determinar si el bit b_0 es 0 o 1.

Se puede determinar de forma similar el bit b_1. La expresión para la diferencia sería $(b_3b_2b_1) - 2(b_3b_2)$. Si la diferencia es cero, entonces b_1 es cero. Los bits b_2 y b_3 son determinados de forma similar.

F. Reorganizando la FFT

El paso final en el cálculo de la FFT es *reorganizar* los resultados de acuerdo a la Ec. (19). Recordando que el procedimiento para reorganizar el vector $\overline{\boldsymbol{X(n)}}$ es representar el índice n en binario y luego invertir o girar dicho número binario. En la Figura 5 se muestra el resultado de la operación de inversión de bit: los términos $x_4(k)$ y $x_4(i)$ simplemente han sido intercambiados, donde i es el entero obtenido por la inversión de bit del entero k.

Obsérvese que una situación similar al concepto de nodo dual existe cuando se reorganiza el arreglo de salida. Si se opera en el arreglo, intercambiando $x(k)$ con el apropiado $x(i)$, eventualmente se encontrará con un nodo que ha sido previamente intercambiado. Por ejemplo, en la Figura 5, el nodo $k = 0$ permanece en su ubicación, los nodos $k = 1$, 2 y 3 son intercambiados con los nodos 8, 4 y 12 respectivamente. El siguiente nodo a ser considerado es el nodo 4, pero este nodo fue previamente intercambiado con el nodo 2. Para eliminar la posibilidad de considerar un nodo que ha sido previamente intercambiado, simplemente se verifica si i (el entero obtenido de la inversión de bit de k) es menor que k. Si es así, implica que el nodo ha sido intercambiado por una operación previa. Con esta verificación, se puede fácilmente asegurar el proceso de reorganizado.

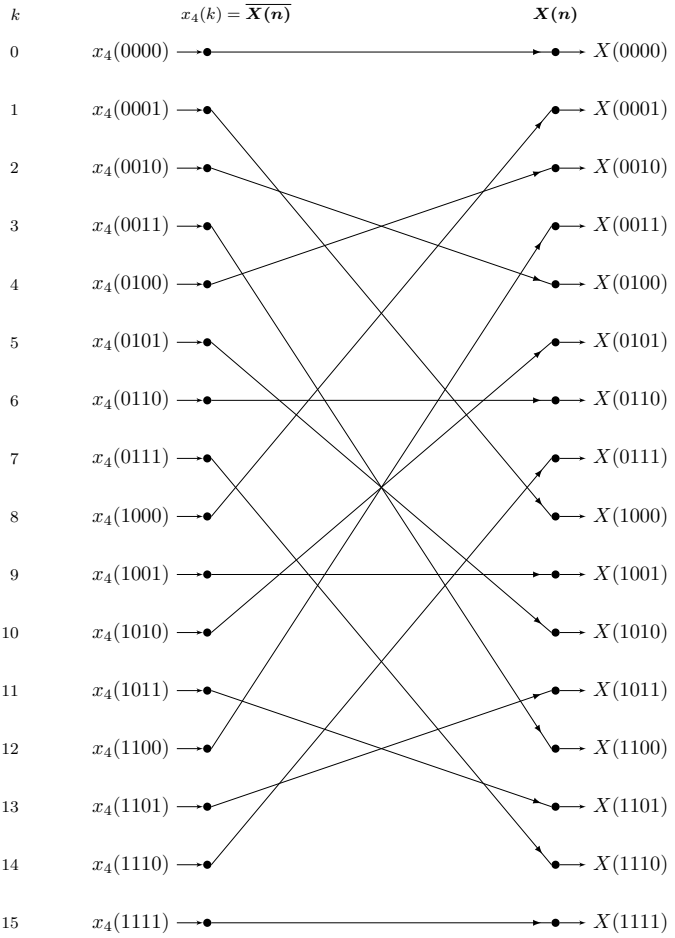

Figura 5: *Ejemplo de la operación de inversión de bit para* $N = 16$

G. Desarrollo teórico del algoritmo FFT de base-2

En la Subsección B, se uso un argumento matricial para desarrollar un entendimiento del porqué la FFT es un algoritmo eficiente. Se contruyó entonces un diagrama de flujo de señal que describe el algoritmo para cualquier $N = 2^\gamma$. Ahora relacionaremos cada uno de estos argumentos en una base teórica. Se desarrollará una prueba teórica del algoritmo para el caso de $N = 4$. Para el caso $N = 2^\gamma$, donde γ es un valor entero, se puede consultar la formulación original de Cooley-Tukey[6] y el libro de Brigham[3].

Definición de la notación. Considérese la transformada de Fourier discreta presentada en la Ec. (1)

$$X(n) = \sum_{k=0}^{N-1} x_0(k) W^{nk}, \qquad n = 0, 1, \ldots, N-1 \quad (22)$$

donde se define $W = e^{-j2\pi/N}$. Es conveniente representar los enteros n y k como números binarios; por lo tanto, si se toma $N = 4$, entonces $\gamma = 2$ y se pueden representar

k y n como números binarios de dos bits,

$$k = 0,\ 1,\ 2,\ 3 \qquad \text{o} \qquad k = (k_1, k_0) = 00,\ 01,\ 10,\ 11$$
$$n = 0,\ 1,\ 2,\ 3 \qquad \text{o} \qquad n = (n_1, n_0) = 00,\ 01,\ 10,\ 11$$

Un método compacto de escribir k y n es:

$$k = 2k_1 + k_0 \qquad n = 2n_1 + n_0 \tag{23}$$

donde k_0, k_1, n_0 y n_1 pueden tomar solamente los valores de 0 y 1. La ecuación (23) es simplemente el método para escribir un número binario en su equivalente en base-10.

Usando la representación de la ecuación (23), se puede reescribir la ecuación (22) para el caso de $N = 4$, como:

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{(2n_1+n_0)(2k_1+k_0)} \tag{24}$$

Observe que la única sumatoria en la ecuación (22) debe ser reemplazada por γ sumatorias con el fin de evaluar todos los bits de la representación binaria de k.

Factorización de W^p. Ahora considere el término W^p. Debido a que $W^{a+b} = W^a W^b$, entonces:

$$\begin{aligned} W^{(2n_1+n_0)(2k_1+k_0)} &= W^{(2n_1+n_0)2k_1} W^{(2n_1+n_0)k_0} \\ &= \left(W^{4n_1k_1}\right) W^{2n_0k_1} W^{(2n_1+n_0)k_0} \\ &= W^{2n_0k_1} W^{(2n_1+n_0)k_0} \end{aligned} \tag{25}$$

Nótese que el término entre paréntesis es igual a la unidad debido a que:

$$W^{4n_1k_1} = \left(W^4\right)^{n_1k_1} = \left(e^{-j2\pi \cdot 4/4}\right)^{n_1k_1} = (1)^{n_1k_1} = 1 \tag{26}$$

Por lo tanto, la ecuación (24) puede ser escrita en la forma:

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0k_1} \right] W^{(2n_1+n_0)k_0} \tag{27}$$

Esta ecuación representa el fundamento del algoritmo FFT. Para demostrar este punto, considere individualmente cada una de las sumatorias de la ecuación (27). Primero, reescriba la sumatoria entre corchetes como:

$$x_1(n_0, k_0) = \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0k_1} \tag{28}$$

Enumerando las ecuaciones representadas por la ecuación (28), se obtiene:

$$\begin{aligned} x_1(0,0) &= x_0(0,0) + W^0 x_0(1,0) \\ x_1(0,1) &= x_0(0,1) + W^0 x_0(1,1) \\ x_1(1,0) &= x_0(0,0) + W^2 x_0(1,0) \\ x_1(1,1) &= x_0(0,1) + W^2 x_0(1,1) \end{aligned} \tag{29}$$

Si reescribimos la ecuación (29) en notación matricial, obtenemos:

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0,0) \\ x_0(0,1) \\ x_0(1,0) \\ x_0(1,1) \end{bmatrix} \tag{30}$$

Nótese que la ecuación (30) es exactamente la matriz factorizada de la ecuación (11), desarrollada en la Subsección B, con el indice k escrito en notación binaria. Por lo tanto, la sumatoria interna de la ecuación (27) especifica la primera de las matrices factorizadas para el ejemplo desarrollado en la Subsección B o, equivalentemente, el arreglo $l = 1$ de el diagrama de flujo de señal ilustrado en la Figura 2.

De forma similar, si escribimos la sumatoria externa de la ecuación (27) como:

$$x_2(n_0, n_1) = \sum_{k_0=0}^{1} x_1(n_0, k_0) W^{(2n_1+n_0)k_0} \tag{31}$$

y representamos el resultado en forma matricial, se obtiene:

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(1,0) \\ x_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \tag{32}$$

la cual es la ecuación (14). Así, la sumatoria externa de la ecuación (27) determina la segunda de las matrices factorizadas del ejemplo de la Subsección B.

De las ecuaciones (27) y (31) tenemos:

$$X(n_1, n_0) = x_2(n_0, n_1) \tag{33}$$

Es decir, los resultados finales $x_2(n_0, n_1)$ tal como se obtuvieron de la sumatoria externa están en orden inverso de bit respecto a los valores deseados $X(n_1, n_0)$. Esto es simplemente la desorganizació que resulta del algoritmo FFT.

Si combinamos las ecuaciones (28), (31) y (33),

$$\begin{aligned} x_1(n_0, k_0) &= \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0k_1} \\ x_2(n_0, n_1) &= \sum_{k_0=0}^{1} x_1(n_0, k_0) W^{(2n_1+n_0)k_0} \\ X(n_1, n_0) &= x_2(n_0, n_1) \end{aligned} \tag{34}$$

donde el conjunto de la ecuación (34) representa la formulación original de Cooley-Tukey[6] del algoritmo FFT para $N = 4$. A este tipo de ecuaciones se les conoce como *recursivas*, ya que la segunda se calcula a partir de la primera.

III. Conclusiones

El algoritmo FFT es de gran difusión y uso en la actualidad debido a su extremada simplificación en el cálculo de la transformada discreta de Fourier, traducido en un tiempo de computo muy reducido.

La implementación del algoritmo a lenguaje de programación es rápido y sin contratiempo por la simplicidad del mismo.

Referencias

[1] Bergland, G. D. (1968). A fast Fourier transform algorithm using base 8 iterations. *Mathematics of Computation*, 22(102), 275, doi:10.1090/S0025-5718-1968-0226899-X.

[2] Bergland, G. D. (1969). A guided tour of the fast Fourier transform. *IEEE Spectrum*, 6(7), 41–52, doi:10.1109/MSPEC.1969.5213896.

[3] Brigham, E. O., Editorial (©1988). *The fast Fourier transform and its applications*. Englewood Cliffs, N.J.: Prentice Hall. URL `http://www.worldcat.org/oclc/17384259`.

[4] Cooley, J. W.; Lewis, P. y Welch, P. D. (1967). Historical notes on the fast Fourier transform. *Proceedings of the IEEE*, 55(10), 1675–1677, doi:10.1109/PROC.1967.5959.

[5] Cooley, J. W.; Lewis, P. A. W. y Welch, P. D. (1969). The fast Fourier transform and its applications. *IEEE Transactions on Education*, 12(1), 27–34, doi:10.1109/TE.1969.4320436.

[6] Cooley, J. W. y Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90), 297, doi:10.1090/S0025-5718-1965-0178586-1.

[7] Departamento de Ingeniería Eléctrica y Computadoras., Métodos rápidos para el cálculo de la TDF. Universidad Nacional del Sur. URL `http://www.ingelec.uns.edu.ar/pds2803/Materiales/Cap12/12-Cap12.pdf`.

[8] Dima Batenkov (2005). Fast Fourier transform: key papers in computer science, seminar 2005. *Weizmann Institute of Science*. URL `http://www.wisdom.weizmann.ac.il/~naor/COURSE/fft-lecture.pdf`.

[9] Donnelly, D. (2006). The fast Fourier transform for experimentalists, part V: filters. *Computing in Science & Engineering*, 8(1), 92–95, doi:10.1109/MCSE.2006.14.

[10] Donnelly, D. (2006). The fast Fourier transform for experimentalists, part VI: chirp of a bat. *Computing in Science & Engineering*, 8(2), 72–78, doi:10.1109/MCSE.2006.33.

[11] Donnelly, D. y Rust, B. (2005). The fast Fourier transform for experimentalists, part I: concepts. *Computing in Science and Engineering*, 7(2), 80–88, doi:10.1109/MCSE.2005.42.

[12] Donnelly, D. y Rust, B. (2005). The fast Fourier transform for experimentalists, part II: convolutions. *Computing in Science and Engineering*, 7(4), 92–95, doi:10.1109/MCSE.2005.82.

[13] Heideman, M. T.; Johnson, D. H. y Burrus, C. S. (1985). Gauss and the history of the fast Fourier transform. *Archive for History of Exact Sciences*, 34(3), 265–277, doi:10.1007/BF00348431.

[14] Rust, B. y Donnelly, D. (2005). The fast Fourier transform for experimentalists, part III: classical spectral analysis. *Computing in Science and Engineering*, 7(5), 74–78, doi:10.1109/MCSE.2005.103.

[15] Rust, B. y Donnelly, D. (2005). The fast Fourier transform for experimentalists, part IV: autoregressive spectral analysis. *Computing in Science and Engineering*, 7(6), 85–90, doi:10.1109/MCSE.2005.126.