3. LA DFT Y FFT PARA EL ANÁLISIS FRECUENCIAL

Una de las herramientas más útiles para el análisis y diseño de sistemas LIT (lineales e invariantes en el tiempo), es la transformada de Fourier. Esta representación de señales implica la descomposición de las mismas en términos de componentes sinusoidales o exponentes complejas. Con esta descomposición, se dice que una señal está representada en el dominio de la frecuencia [3].

La representación en series de Fourier de una señal periódica en tiempo continuo puede tener infinitas componentes en frecuencia, donde la separación en frecuencia de dos componentes armónicas sucesivas es $1/T_p$, donde T_p es el periodo fundamental. Por lo tanto podemos tener un rango de frecuencias desde $-\infty$ hasta $+\infty$ con infinitas componentes en frecuencia. En cambio, el rango de frecuencias de señales en tiempo discreto se limita al intervalo $(-\pi, \pi)$. Una señal discreta de periodo fundamental N puede tener componentes en frecuencias separadas $2\pi/N$ radianes. Consecuentemente, la representación en series de Fourier de señales periódicas en tiempo discreto contendrá un máximo de N componentes de frecuencia [3]. Para calcular el espectro de señales en el tiempo continuo así como discreto, se requieren los valores de la señal para cada instante de tiempo, eso es en realidad imposible dado que nuestra señal es finita y el tiempo no, sin embargo con la DFT o Transformada Discreta de Fourier se puede lograr una excelente aproximación [10].

3.1 La Transformada Discreta de Fourier (DFT)

La función en el dominio de la frecuencia $X(\omega)$ correspondiente a una señal discreta x_n será definida por la suma infinita

$$X(\omega) = \sum_{k=-\infty}^{\infty} x_k e^{-j\omega k} \tag{3.1}$$

Donde $X(\omega)$ representa una función periódica compleja de ω con un periodo de 2π. Al ser x_n discreta, la sumatoria $X(\omega)$ resulta simétrica y puede estar caracterizada en su totalidad por sus valores dentro del intervalo $0 \leq \omega \leq \pi$ radianes por muestra [8]. Normalmente los valores de una señal discreta son muestras provenientes de una función continua en el tiempo $x(t)$,

$$X_k = x(kT_s) \tag{3.2}$$

donde T_s es el periodo de muestreo [8]. Este valor debe ser lo suficientemente pequeño de forma que se evite la pérdida de información durante las rápidas fluctuaciones en $x(t)$. Parece haber dos problemas, tanto para el humano como para la computadora, al querer evaluar $X(\omega)$. Uno es, que tenemos un número infinito de términos a sumar. El otro es que se desea que $X(\omega)$ esté en un rango continuo de frecuencias (un número infinito de valores para ω. Ni la computadora más rápida acabaría algún día de realizar la suma de todos los términos. Realmente el que nuestro rango de suma sea infinito no es problema, se puede sustituir por una sumatoria finita o transformada truncada. Lógicamente este pequeño cambio introducirá errores en la transformada discreta de Fourier $X(\omega)$. El tiempo que tarde una computadora en realizar la suma se reducirá, pero los errores en $X(\omega)$ aumentarán mientras menos muestras se tomen de una señal o conjunto de muestras finito. Por consiguiente existe una relación que habrá que tomar en cuenta entre el número de muestras y el espectro resultante (transformada discreta de Fourier) [8].

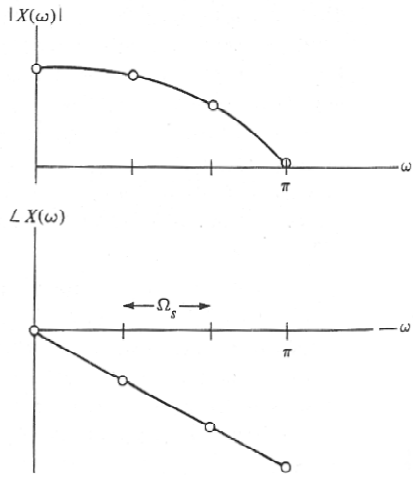

Figura 3.1 Espectro en frecuencia discreto. (arriba) Magnitud, (abajo) Fase.

En la Figura 3.1 se puede apreciar un espectro en frecuencia discreto. Para demostrar la elección apropiada de frecuencias discretas, podemos decir que son múltiplos $m\Omega_s$ de algún espaciamiento de frecuencias Ω_s. Si además, solamente tenemos N muestras del dato $x_n = x(kT_s)$, la transformada discreta de Fourier (3.1) se convierte en [8]:

$$X(m\Omega_s) = \sum_{k=0}^{N-1} x(kT_s) e^{-jkm\Omega_s T_s} \tag{3.3}$$

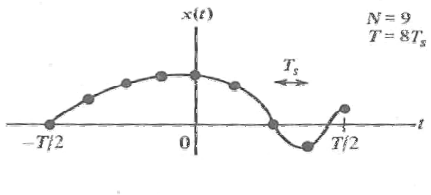

Figura 3.2 Señal limitada en el tiempo. Con N muestras y periodo T.

Ahora hay que encontrar una frecuencia de espaciamiento para Ω_s. Basándonos en la Figura. 3.2, una señal limitada en el tiempo $x(t)$ definida de $-T/2$ a $T/2$, de donde se obtienen N muestras y $T = (N-1)/Ts$. Podemos obtener la siguiente relación para una frecuencia de espaciamiento [8]:

$$\Omega_s < \frac{2\pi}{T} \tag{3.4}$$

Si realizamos la sumatoria de la transformada discreta de Fourier a frecuencias espaciadas por Ω_s que satisfaga (3.4), una ligera curva sobre los puntos resultantes debe ser una buena representación de $X(\omega)$, la transformada discreta de Fourier de $x(kT_s)$. Esto asume que la frecuencia de muestreo es suficientemente alta. Para llegar a la forma simplificada, habrá que tomar a $T = NT_s$ *de* $T = (N-1)T_s$, que nos daría $\Omega_s = 2\pi/NT_s$ y sustituyendo $X(m\Omega_s) = X_m$, $x(kT_s) = x_k$, obtenemos [8]:

$$X_m = \sum_{k=0}^{N-1} x_k (e^{-j2\pi/N})^{km} \qquad m = 0,1,\ldots, N-1 \tag{3.5}$$

Esta es la sumatoria de la DFT o Transformada Discreta de Fourier.

3.2 La DFT como una transformación lineal

De la ecuación (3.5) se puede proceder a la siguiente ecuación (3.6) de forma que podamos ver a la DFT como una transformación lineal, hay que hacer una sustitución, donde la raíz n-ésima de la unidad es $W_N = e^{-j2\pi/N}$ [3]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k = 0,1,2,\ldots, N-1 \tag{3.6}$$

El cálculo de cada punto de la DFT implica N multiplicaciones complejas y $(N-1)$ sumas complejas , por lo tanto los N puntos de la DFT pueden obtenerse tras N^2 multiplicaciones complejas y $N(N-1)$ sumas complejas. Definiendo el vector $\mathbf{x}_N$ de N puntos de la secuencia $x(n)$, $n = 0,1,\ldots,N-1$, el vector $\mathbf{X}_N$ de N puntos de las muestras en frecuencia y la matriz N x N, $\mathbf{W}_N$, como [3]:

$$x_N = \begin{bmatrix} x(0) \\ x(1) \\ \text{M} \\ x(N-1) \end{bmatrix}, \quad X_N = \begin{bmatrix} X(0) \\ X(1) \\ \text{M} \\ X(N-1) \end{bmatrix}$$
$$W_N = \begin{bmatrix} 1 & 1 & 1\Lambda & 1 & W_N^{N-1} \\ 1 & W_N & W_N^2 & \Lambda & W_N^{2(N-1)} \\ & W_N^2 & W_N^4 & \Lambda & \\ \text{M} & \text{M} & \text{M} & \text{MM} & \text{M} \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \Lambda & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{3.7}$$

De esta forma la DFT de N puntos se puede expresar en forma matricial como

$$\mathbf{X}_N = \mathbf{W}_N \ \mathbf{x}_N \tag{3.8}$$

donde $\mathbf{W}_N$ es la matriz de la transformación lineal. Cabe destacar que $\mathbf{W}_N$ es simétrica. Si suponemos que existe la inversa de $\mathbf{W}_\text{N}$, despejando $\mathbf{x}_N$ obtenemos una expresión para la IDFT (la DFT inversa) [3].

3.3 Propiedades de la DFT

Las propiedades de las series de Fourier para secuencias periódicas y las propiedades de la DFT para secuencias no periódicas, son en esencia las mismas, si vemos a las señales de duración finita de la DFT como periódicas [9].

Las siguientes propiedades son para secuencias de duración finita definidas en el intervalo $0 \leq n \geq N$-1.

3.3.1 Linealidad

Si dos secuencias de duración finita $x_1(n)$ y $x_2(n)$ son combinadas linealmente como en

$$x_3(n) = ax_1(n) + bx_2(n) \tag{3.9}$$

luego entonces la DFT de $x_3(n) = aX_1(k) + bX_2(k)$

Claramente si $x_1(n)$ tiene duración N_1 y $x_2(n)$ tiene duración N_2, entonces la duración máxima de $x_3(n)$ será N_3 = máx. [N_1, N_2]. Por lo tanto, la DFTs tiene que ser calculada con $N = N_3$ [10] .

3.3.2 Periodicidad

Si *x(n)* y X(k) son un par de transformadas DFT de *N* puntos, entonces

$$x(n + N) = x(n) \text{ para todo } n \tag{3.10}$$

$$X(k + N) = X(k) \text{ para todo } k \tag{3.11}$$

Estas periodicidades de $x(n)$ y $X(k)$ son consecuencia directa de las fórmulas de la DFT y la IDFT [3].

3.3.3 Corrimiento circular en el tiempo

La siguiente propiedad está representada por la ecuación (3.12):

$$x[n - i] = e^{-j(2\pi/N)ki} X[k] \tag{3.12}$$

donde *i* es un entero arbitrario [9].

Esta propiedad debe de interpretarse con cuidado si $x[n-i]$ ha sido obtenida de una continuación periódica de una señal $x[n]$ de duración finita, por ejemplo $x[n] = 0$ para $n < 0$ y $n \geq N$. Cuando se hace un corrimiento circular, se espera que el punto que desapareció a la derecha, aparezca a la izquierda, como si estuviera girando. En general para el corrimiento circular con periodo *N* [9]**:**

$$x[n - i] = x[n - i + lN] \text{ con } l = 0, \pm1, \pm2, \ldots \tag{3.13}$$

3.4 La Transformada Rápida de Fourier (FFT)

El problema al calcular la DFT, básicamente es la secuencia $\{X(k)\}$ de *N* números complejos dada la secuencia de datos $\{x(n)\}$ de longitud *N* según la ecuación (3.6) donde se había comentado ya que $W_N = e^{-j2\pi/N}$ [3]. Dado que tanto la DFT como la IDFT implican el mismo tipo de operaciones complejas, existe la FFT y la IFFT respectivamente, sin embargo, únicamente nos enfocaremos en la FFT, ya que el

análisis de esta Tesis esta basado en el tiempo, para obtener una representación en frecuencia. Han existido varios algoritmos desde antes que las computadoras fueran súper rápidas, aún para el cerebro humano simplificaban las operaciones, pero no fue sino hasta 1965 que Cooley y Tukey publicaron un algoritmo para realizar cálculos de forma mucho más eficiente, basándose en el hecho de si N es un número compuesto, formado por el producto de dos o más enteros [10]. A partir de entonces surgen los algoritmos en base 2, en base 4, de base partida, y muchos más llamados algoritmos FFT [3].

3.4.1 Cálculo directo de la DFT

Para una secuencia compleja $x(n)$ de N puntos, la DFT se puede expresar como [3]:

$$X_R(k) = \sum_{n=0}^{N-1}\left[x_R(n)\cos\frac{2\pi kn}{N} + x_I(n)sen\frac{2\pi kn}{N}\right] \tag{3.14}$$

$$X_I(k) = -\sum_{n=0}^{N-1}\left[x_R(n)\cos\frac{2\pi kn}{N} + x_I(n)sen\frac{2\pi kn}{N}\right] \tag{3.15}$$

El cálculo directo de (3.14) y (3.15) requiere [3]:

1. $2N^2$ cálculos de funciones trigonométricas.
2. $4N^2$ multiplicaciones reales.
3. $4N(N-1)$ sumas reales.
4. Numerosas operaciones de direccionamiento indexado.

Estas operaciones son típicas de los algoritmos computacionales de la DFT. Los puntos 2 y 3, corresponden a $X_R(k)$ y $X_I(k)$. Las operaciones de direccionamiento e indexado son necesarias para ir a buscar los datos $x(n)$, $0 \leq n \leq N-1$, y los factores de fase y

almacenar los resultados. De forma que para agilizar cada uno de los procesos computacionales, existen distintos algoritmos para la DFT [3].

3.4.2 Algoritmo para la FFT de una *N* factorizable (Algoritmo Cooley-Tukey)

Como se comentó con anterioridad, una forma más efectiva de realizar los cálculos es descomponiendo a N, en partes más pequeñas. Por ejemplo, si tenemos una DFT de N puntos, se puede representar como el producto de dos enteros $N = LM$ [3]. N por principio no debería ser un número primo, pero no hay problema, ya que se puede rellenar con ceros una secuencia al hacer N un número factorizable [10]. El siguiente paso es almacenar la secuencia $x(n)$, $0 \leq n \leq N\text{-}1$, ya sea una vector o una matriz bidimensional indexada por l y m, donde, $0 \leq l \leq L\text{-}1$, y $0 \leq m \leq M\text{-}1$. Tomemos a l como índice de renglones y a m de columnas, de esta forma [3]:

$$n = Ml + m \tag{3.16}$$

nos indica que el primer renglón contiene los primeros M elementos de $x(n)$, el segundo, los siguientes M elementos d $x(n)$ y así sucesivamente. En contraparte, tenemos la correspondencia [3]

$$n = l + mL \tag{3.17}$$

que contiene los primeros L elementos de $x(n)$ en la primera columna, y de la misma forma que M, se van guardando sucesivamente los L elementos de $x(n)$ en la siguiente columna. De esta misma forma se pueden almacenar los valores calculados de la DFT. Vamos a considerar la correspondencia desde el índice k a la pareja de índices (p,q), donde $0 \leq p \leq L\text{-}1$, y $0 \leq q \leq M\text{-}1$. Sustituyendo los respectivos valores en (3.16) y (3.17), ahora tenemos para los valores almacenados por renglones

$$k = Mp + q \tag{3.18}$$

y

$$k = ql + pL \tag{3.19}$$

para el almacenamiento por columnas de *X(k)*.

Suponiendo que *x(n)* se coloca en la matriz *x(l,m)*, de la misma forma que *X(k)* se almacena en la matriz *X(p,q)*. Ahora la DFT puede expresarse como una doble sumatoria. Veamos como queda al tomar el almacenamiento por columnas de *x(n)* (3.17)

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)} \tag{3.20}$$

Haciendo un poco de matemáticas para simplificar, nos queda

$$X(p,q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l,m) W_M^{mq} \right] \right\} W_L^{lp} \tag{3.21}$$

Para resolver (3.21), se requieren tres etapas: lo que está dentro de los paréntesis cuadrados como primer paso, segundo paso, lo que está dentro de los corchetes y finalmente la sumatoria de *L*. Parecería más complicado que el procedimiento de cálculo de DFT directo, sin embargo, se reducen tanto multiplicaciones complejas, como sumas complejas a $N(M + L + 1)$ y $N(M + L - 2)$ respectivamente (donde $N=ML$) [3].

3.4.3 Algoritmo para la FFT base 2 (diezmado en tiempo)

El método anterior es muy eficaz cuando *N* es un número no primo, es decir que se puede factorizar como $N = r_1 r_2 r_3 \ldots r_v$, donde los $\{r_j\}$ son primos. Cuando $r_1 = r_2 = \ldots = r_v \equiv r$, es decir, $N = r^v$, todas las DFTs son de tamaño *r*, de manera que el cálculo de las DFTs de *N* sigue un patrón regular y al número *r* se le llama base del algoritmo para la FFT. Tal es el caso de los algoritmos en base 2 [3] (los más usados según [3] [8] [9] [10]).

Explicaremos cómo trabaja este algoritmo, calculando la DFT de $N = 2^v$. Elegimos $M = N/2$ y $L = 2$, de esta forma podemos dividir la secuencia de datos de N puntos en dos secuencias de datos de $N/2$ puntos, $f_1(n)$ y $f_2(n)$, que corresponden a las muestras pares e impares de $x(n,)$ respectivamente. Estas funciones se obtienen diezmando (dividiendo) $x(n)$ por 2, de ahí el nombre de diezmado en tiempo [3].

La DFT de N puntos puede expresarse en términos de las DFTs de las secuencias diezmadas de la siguiente manera

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k = 0,1,\ldots, N-1 \\ &= \sum_{n\ par} x(n) W_N^{kn} + \sum_{n\ impar} x(n) W_N^{kn} \\ &= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)} \end{aligned} \tag{3.22}$$

Como $W_N^2 = W_{N/2}$. Entonces (3.22) quedaría de la siguiente forma

$$\begin{aligned} X(k) &= \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km} \\ &= F_1(k) + W_N^k F_2(k) \qquad k = 0,1,\ldots, N-1 \end{aligned} \tag{3.23}$$

donde $F_1(k)$ y $F_2(k)$ son las DFTs de $N/2$ puntos de las secuencias $f_1(m)$ y $f_2(m)$, respectivamente [3].

Se puede observar que el cálculo directo de $F_1(k)$ requiere $(N/2)^2$ multiplicaciones complejas. Lo mismo para $F_2(k)$. Además se requieren $N/2$ multiplicaciones complejas más para calcular $W_N^k F_2(k)$. Por lo tanto el cálculo de $X(k)$ requiere $2(N/2)^2 + N/2 = N^2/2 + N/2$ multiplicaciones complejas. El primer paso da lugar a una reducción en el número de multiplicaciones de N^2a $N^2/2 + N/2$, que equivale aproximadamente a dividir por 2 el número de multiplicaciones cuando N es grande [3].

Una vez hecho el diezmado en tiempo una vez, podemos repetir el proceso para cada una de las secuencias $f_1(n)$ y $f_2(n)$. Por lo tanto cada una dará lugar a otras dos secuencias de $N/4$ puntos. Después de realizar el primer diezmado, vemos ahora que $F_1(k)$ y $F_2(k)$ requieren $N^2/4 + N/2$ multiplicaciones complejas y $N/2$ más, para calcular $X(k)$ a partir de $F_1(k)$ y $F_2(k)$. Por consiguiente, el número total de multiplicaciones $N^2/4 + N$ se reduce una vez más por 2, aproximadamente. Siguiendo este sistema, el diezmado se puede repetir hasta que las secuencias resultantes sean de un punto. Para $N = 2^v$, el diezmado puede realizarse $v = \log_2 N$ veces. De esta forma, el número total de multiplicaciones complejas se reduce a $(N/2)\log_2 N$. El número de sumas complejas es $N \log_2 N$ [3] [9].

3.4.4 SRFFT (Split Radix Fast Fourier Transform) o Algoritmo de base partida

Este algoritmo es muy eficaz ya que combina el diezmado en base 2 y base 4 en un solo algoritmo. Es decir, si tenemos una $N = 32$, por ejemplo, el cálculo se dividirá en dos etapas, A y B. En la etapa A, se realizará una mariposa de base 2 y en lugar de continuar en base 2, ahora el resultado de 16 puntos de esa mariposa es el comienzo de la etapa B, donde entran dos mariposas de 16 puntos en base 4, y el resultado de cada una serán 8 puntos a la salida. Suponiendo que este ejemplo usa diezmado en frecuencia, en lugar de en tiempo, las entradas al sistema estarán en orden y las salidas serán en espejo (invertidas), habrá que ordenarlas [7] [3]. La siguiente tabla muestra la eficiencia de este algoritmo con relación a la base 2, base 4, base 8 [3]:

	Multiplicaciones Reales				Sumas Reales			
N	*Base 2*	*Base 4*	*Base 8*	*Base Partida*	*Base 2*	*Base 4*	*Base 8*	*Base Partida*

16	24	20		**20**	152	148		**148**
32	88			**68**	408			**388**
64	264	208	204	**196**	1,032	976	972	**964**
128	712			**516**	2,504			**2,308**
256	1,800	1,392		**1,284**	5,896	5,488		**5,380**
512	4,360		3,204	**3,076**	13,566		12,420	**12,292**
1024	10,248	7,856		**7,172**	30,728	28,336		**27,652**

Tabla 3.1 Eficiencia de diferentes algoritmos FFT.

Hay que recordar que el número de operaciones en el cálculo directo de de la DFT es de N^2, así que podemos ver que cualquiera de estos algoritmos ya es suficiente reducción de operaciones. En realidad lo que se quiere demostrar con esta tabla, es la eficiencia de la Base Partida con relación a Base 2. En la sección 5.4.2 hablaremos más a cerca de los algoritmos FFT que usa MATLAB.