UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

Departamento de Electrónica

GUÍA DE LABORATORIO 3
TRANSFORMADA RÁPIDA DE FOURIER FFT

CURSO	LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES
SIGLA	ELO–385
PROFESOR	MARCO E. RIVERA A.
AYUDANTE	IGNACIO LIZAMA A.

Valparaíso, Abril del 2009

Guía de Laboratorio 3
Transformada Rápida de Fourier FFT

En esta guía, se presentan los conceptos básicos de la Transformada de Fourier y se estudiarán los efectos de diversos factores en el cálculo de ésta, utilizando MatLab para el uso de ventanas y filtrado de datos, destinados a calcular una Transformada de Fourier, además del uso de ceros para el llenado de los buffers. La Transformada de Fourier Discreta, juega un papel importante en varias aplicaciones de procesamiento de señales digitales, entre las cuales destacan el filtrado lineal, el análisis de correlación y análisis espectral. **El objetivo de este laboratorio** es implementar la Transformada Rápida de Fourier (FFT) para analizar el espectro de una señal de entrada en tiempo real. Después de calcular la FFT, de un bloque de muestras de entrada, se calculará la magnitud del resultado para posteriormente visualizarlo en el Code Composer y en el osciloscopio, creando así, un analizador de espectros.

1. Serie de Fourier

Esta transformada se usa para la descomposición en frecuencia de señales temporales, así como la búsqueda de patrones. Al considerar una señal periódica de periodo T, se puede definir:

$$f(t) = a_0 + a_1 \cdot \cos(w_0 \cdot t) + a_2 \cdot \cos(2 \cdot w_0 \cdot t) + ... + b_1 \cdot \sin(w_0 \cdot t) + b_2 \cdot \sin(2 \cdot w_0 \cdot t) + ... \quad (1)$$

$$f(t) = \sum_{k=0}^{\infty} \left[a_k \cdot \cos(k \cdot w_0 \cdot t) + b_k \cdot \sin(k \cdot w_0 \cdot t) \right], \text{ con } w_0 = \frac{2 \cdot \pi}{T} \quad (2)$$

El problema es determinar los coeficientes a_k y b_k, los que se obtienen mediante:

$$a_k = \frac{1}{T} \cdot \int_0^T f(\tau) \cdot \cos(k \cdot w_0 \cdot \tau) \cdot d\tau \quad (3)$$

$$b_k = \frac{1}{T} \cdot \int_0^T f(\tau) \cdot \sin(k \cdot w_0 \cdot \tau) \cdot d\tau \quad (4)$$

Otra manera de representar el resultado anterior es a través de notación compleja, considerando:

$$W = e^{-j \cdot k \cdot w_0 \cdot t} = \cos(k \cdot w_0 \cdot t) - j \cdot \sin(k \cdot w_0 \cdot t) \quad (5)$$

Así, se tendrá por lo tanto:

$$a_k + j \cdot b_k = c_k = \frac{1}{T} \cdot \int_0^T f(\tau) \cdot e^{-j \cdot k \cdot w_0 \cdot \tau} \cdot d\tau \quad (6)$$

Todo lo anterior, puede replicarse para señales discretas, así, (6) puede ser representada como:

$$a_k + j \cdot b_k = c_k = \frac{1}{T} \cdot \sum_{n=0}^{N} f(n \cdot \Delta) \cdot e^{-j \cdot k \cdot w_0 \cdot n \cdot \Delta}, \text{ donde } N \cdot \Delta = T \quad (7)$$

Esta transformada de Fourier Discreta DFT, se caracteriza porque de una secuencia de N puntos se obtienen N componentes espectrales equiespaciadas en N / f_s, y además, porque de las N componentes $N / 2$ representan frecuencias positivas y $N / 2$ frecuencias "negativas" (parte compleja negativa). Por lo que los espectros son simétricos con respecto a $N / 2$.

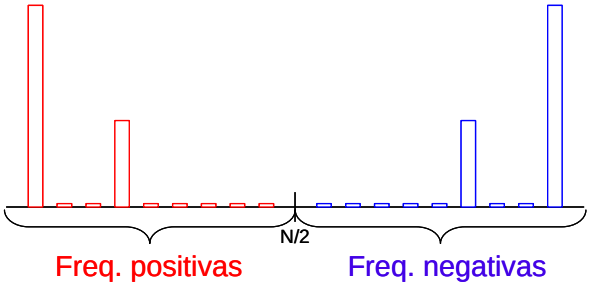

Fig. 1. Componentes DFT.

Para un correcto resultado se necesita de una cantidad **entera** de periodos. De otro modo aparecerán componentes espurias, las que pueden minimizarse con el uso de ventanas. La DFT requiere de *NxN* multiplicaciones por los coeficientes, los que además, son números complejos, por lo que estas multiplicaciones son más complejas de lo normal.

2. Transformada Rápida de Fourier FFT

La FFT utiliza ciertos aspectos de simetría de estos factores de manera de agrupar multiplicaciones, reduciendo la cantidad de éstas a $N \log_2(N)$, (N es una potencia de 2).

La FFT es un eficiente algoritmo de cómputo de la Transformada de Fourier Discreta (DFT) la cual requiere de una gran poder de cálculo para ser implementada.

$$X(n) = \sum_{k=0}^{N-1} x(k) \cdot e^{-j\frac{2\pi}{N}nk}, \; k = 0,1,\ldots, \; N-1 \quad (8)$$

En (8), puede observarse que existen dos secuencias de datos:

$x(n)$: arreglo de señales adquiridas y cuyo espectro se desea conocer.

$X(k)$: arreglo con la secuencia de salida que posee el espectro de la entrada $x(n)$.

Debido a que la FFT calcula el espectro de un bloque de muestras de entrada, estos cálculos serán realizados una vez para cada bloque de datos recolectados. Vale decir, se empleará un buffer lineal o circular que almacene *N* muestras consecutivas antes de realizar el algoritmo FFT. Para evitar efectos indeseados por cortes abruptos del bloque de señal y adquisición de múltiplos no enteros de componentes armónicas, se empleará una ventana adecuada sobre las muestras de la señal. La ventana, al igual que en el caso de los

filtros FIR, es simplemente una multiplicación muestra a muestra de la señal de entrada por una función de largo fijo dependiente del número de puntos de la FFT. Como se verá, la elección de la ventana afecta las propiedades del espectro resultante.

3. Efecto de la Aplicación de Ventanas

El bloque de la señal de entrada almacenado para el cálculo de la FFT genera ciertos efectos sobre el espectro obtenido debido a los cambios abruptos al comienzo y final de la trama de datos y a que generalmente se procesan señales cuyas frecuencias no son múltiplos enteros de la frecuencia de muestreo. Para evitar y/o disminuir la aparición de frecuencias espurias se utilizan ventanas, de manera similar al caso de los filtros FIR, pero en este caso aplicado a los datos obtenidos. En el siguiente script, se genera una FFT de 256 puntos, donde una función ventana es multiplicada por las muestras de la señal de entrada (línea 10).

```
1.  close all
2.  clear all
3.  N = 256;                          % números de puntos de data
4.  M = 256;                          % número de muestras
5.  Fs = 8000;                        %Frecuencia de muestreo
6.  f = 500;                          % Frecuencia de la señal a analizar
7.  t = [0 : M-1]' / Fs;              %genera un vector de tiempo
8.  x = sin(2*pi*f*t);                %genera el tono de 1 KHz
9.  x = [x;zeros(N-M,1)];             %Rellena de ceros según M - N
10. x = x .* boxcar(N);               % multiplica por la función ventana boxcar
11. s = abs(fft(x)).^2;               % calcula la magnitud cuadrada de la FFT
12. s = s / max(s);                   % normaliza de modo que el máximo sea 1.0
13. fp= Fs * [0:N-1] /N;              %genera un vector de frecuencia
14. plot(x)                           %Graficado y presentación
15. figure;
16. subplot(2,1,1);
17. a=stem(fp(1:N/2),s(1:N/2));
18. set(a,'MarkerSize',2)
19. xlabel('frecuencia en Hz');
20. title('Magnitud al cuadrado de FFT');
21. grid on;
22. subplot(2,1,2);
23. a=stem(fp,s);
24. set(a,'MarkerSize',2)
25. axis([0 1000 0 1])
26. xlabel('frecuencia en Hz');
27. title('Vista Detallada');
28. grid on;
```

Script de MatLab para la FFT.

4. Efecto de Rellenar con Ceros (Padding)

Si se tienen N ($=2^n$) muestras de una señal, el número de coeficientes de la FFT que se obtendrán empleando el comando de MatLab *"fft"* es también N en el intervalo $[0,2\pi]$. En algunos casos no se cuenta con una cantidad de datos que sea una potencia de 2, por lo que existen dos alternativas:

- Agregar o eliminar datos para llegar a la potencia de 2 más cercana.
- Agregar ceros a los datos obtenidos hasta alcanzar la potencia de 2 superior más cercana.

Eliminar datos tiene la desventaja adicional de disminuir la resolución de la FFT, debido a que la cantidad de datos N con la que se opera es menor. Mientras que el agregar ceros a los datos puede introducir alguna

“contaminación armónica”, pero ésto puede mejorarse con el uso de las ventanas vistas en el punto anterior. En MatLab, si el vector original (muestreado) es de longitud $M<N$, la cantidad de ceros con la que debe ser rellenado el vector es de N-M, por lo que el vector x de datos a ser procesado es (línea 9 del script):

x = [x;zeros(N-M,1)];

La línea 3 en el código, define la longitud del vector de datos después de rellenar con ceros. Así $M = N = 256$ indica que no se tiene relleno con ceros. Notar que N es el total de puntos y N-M es la cantidad de ceros con que se rellenará el arreglo, es claro que N debe ser mayor que M.

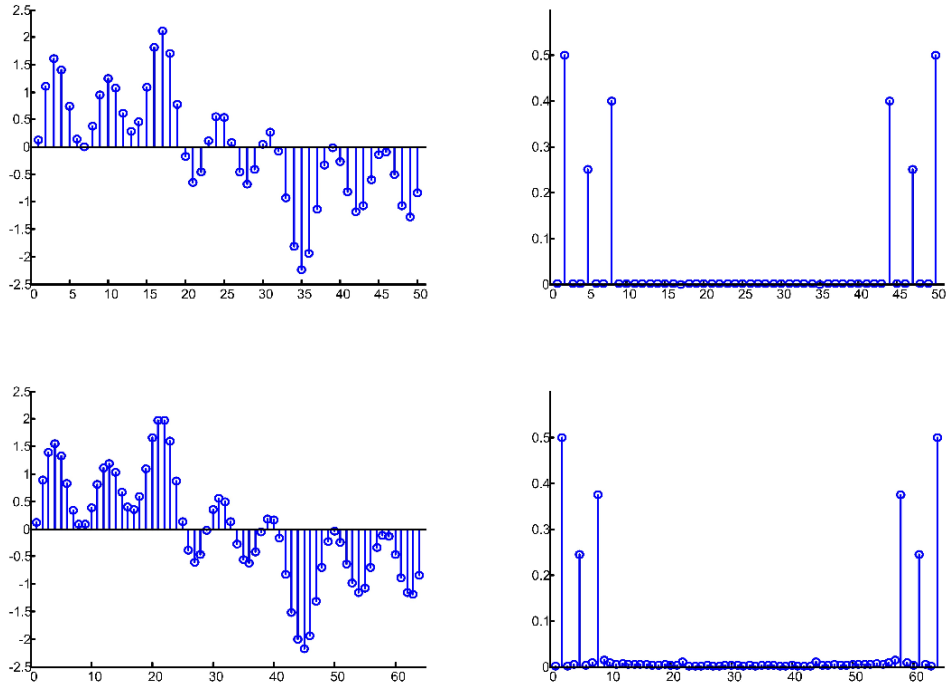

Fig. 2. Interpolación de los datos.

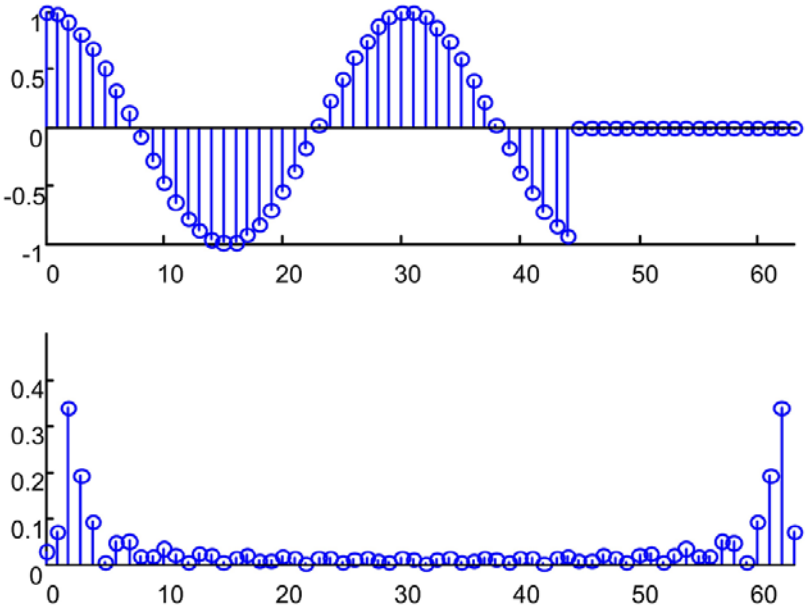

Fig. 3. Relleno de Ceros.

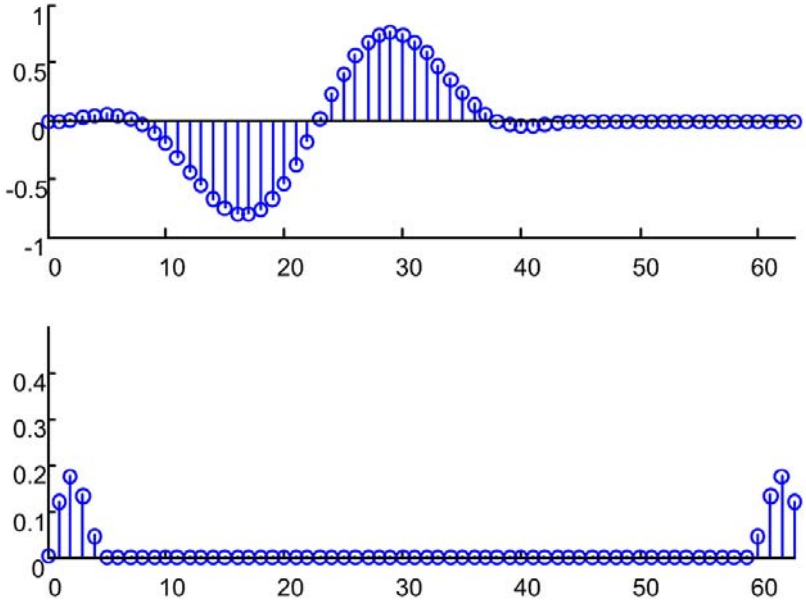

Fig. 4. Relleno de Ceros con Ventana Auxiliar.

5. Bit Reverse

Note que la FFT es una manera diferente de implementar una DFT, por lo que hereda todas sus propiedades, ventajas y desventajas. El vector de salida de la FFT contiene todas las componentes espectrales, pero se encuentran “desordenadas”, por lo que es necesario implementar un algoritmo al final del cálculo para reordenarlos (Bit Reverse).

Este algoritmo permite que una secuencia de entrada o salida del cálculo de la FFT sea reordenada para obtener un resultado deseado. En este caso, la salida es la que se desea reordenar, por lo que se recomienda considerar el siguiente código a la hora de implementar:

```
p=1
for (q=0;q<N;q++)
    bit_rev[q]=q;
    while(p<N)
    {
        for (q=0;q<p;q++)
        {
            bit_rev[q]=bit_rev[q]*2;
            bit_rev[q+p]=bit_rev[q]+1;
        }
        p=p*2;
    }
```

Este cálculo es independiente de los valores de los N puntos por lo que puede realizarse fuera de línea utilizándose solamente su resultado. Una vez obtenido el resultado de la FFT en la variable *fft_temp*, es necesario aplicarle el algoritmo *Bit Reverse* para reordenarlo, lo que se obtiene fácilmente con:

```
for(i=0;i<N;i++){
        ffft_out[bit_rev[i]]=fft_temp[i]; // fft_out FFT ordenada
```

6. Actividades

6.1. Parte Teórica

6.1.1. Ejecute el código anterior para $f = 500[Hz]$ y $f = 420[Hz]$. Analice y comente los resultados obtenidos.

6.1.2. Cambie la ventana *"boxcar"* por una ventana *"hamming"* y realice las mismas simulaciones que en 6.1.1. ¿Cuál es el efecto del uso de la ventana?.

Padding y Eliminación de Datos.

Modifique el script de modo que para N=256:

6.1.3. Obtenga la FFT reduciendo a 128 los puntos utilizados (M=128, N=128), para $f = 500[Hz]$ ***y*** $f = 420[Hz]$**.**

6.1.4. Obtenga la FFT rellenando con ceros de manera de obtener una FFT de 226 puntos utilizados para $f = 500[Hz]$ ***y*** $f = 420[Hz]$, con *boxcar*. (N = 256, M = 226).

6.1.5. Obtenga la FFT rellenando con ceros de manera de obtener una FFT de 256 puntos utilizados para $f = 500[Hz]$ ***y*** $f = 420[Hz]$, con ventana *hamming*. (N = 256, M = 226).

6.1.6. Compare los resultados obtenidos en los últimos tres puntos anteriores.

6.1.7. Realice pruebas para valores menores de *M*, $f = 420[Hz]$ y ventanas *boxcar* y *hamming*. Analice y Comente.

6.2. Parte Experimental

Para la implementación de la FFT se empleará el algoritmo radix-2, con decimación en frecuencia. Escriba el código necesario para calcular la FFT. No olvide su análisis y comentarios. Tenga en cuenta que debe almacenar las muestras en un buffer de entrada, calcular la FFT y luego usar un buffer de salida, recuerde que debe utilizar el algoritmo *Bit Reverse* para reordenar los datos de la FFT. Calcule los valores de W en Matlab y genere un archivo .h con dichos coeficientes. Hay muchas maneras hábiles de reducir el movimiento de datos entre buffers. Se recomienda que intente varias alternativas para aumentar el *"throughout"*. Note que debido a que la FFT es calculada luego que son almacenadas las N muestras, la eficiencia de la rutina FFT y otros procesos (buffering) son importantes para la implementación en tiempo real.

6.2.1. Genere el algoritmo que procese una FFT de 64 puntos en tiempo real. La salida de la FFT deberá escribirse en un arreglo de salida donde cada término corresponderá a la magnitud cuadrática de la respectiva componente espectral:

$$\left|X(k)^2\right| = \mathrm{Re}\left\{X(k)\right\}^2 + \mathrm{Im}\left\{X(k)\right\}^2 \tag{6}$$

Haga los arreglos necesarios para visualizar el resultado de la FFT en las ventanas *Graph* de Code Composer.

6.2.2. Modifique el algoritmo del punto 6.2.1 de manera de tomar 1024 puntos y posteriormente (off-line) procesarlos, una vez obtenida la FFT obtenga la magnitud máxima del arreglo de salida, escale apropiadamente los valores de manera que puedan ser procesados por el codec de audio. Despliegue en el osciloscopio el resultado de la FFT. Como pulso de sincronismo se puede utilizar un valor negativo en la posición correspondiente a la componente DC.