

# Práctica 4: DFT

Curso 2005/2006

## Resumen

El objetivo de esta práctica es manejar la transformada discreta de Fourier DFT, con ayuda del algoritmo rápido de cálculo FFT, en utilidades como la obtención de muestras del espectro, la determinación de la respuesta en frecuencia de un filtro y la implementación de uno de los métodos de filtrado basado en FFT's.

En las distintas partes de la práctica puede ser interesante el uso de las siguientes funciones MATLAB: `reshape()`, `etime()`, `clock()`, `flops()`, `fft()`, `plot()`, `subplot()`, `rand()`, `angle()`, `abs()`, `conv()`, `conj()`, `nextpow2()`, `fftshift()`, `ifftshift()`, `fftfilt()`. Consulte el manual o el “help” sobre las mismas.

## 1. Introducción

En la primera parte de la práctica se va a comprobar que la DFT supone un muestreo en frecuencia del espectro así como el efecto que produce el cálculo de la DFT con diferente número de puntos. En la segunda parte se pretende mostrar la FFT como una herramienta útil para determinar la respuesta en frecuencia de un filtro. Finalmente se programará el filtrado FIR de secuencias muy largas, mediante el método *overlap-save*.

## 2. Relleno con ceros y muestreo en frecuencia

Es conocido que si  $x[n]$  es una señal de duración finita  $N$ , su DFT son muestras de su transformada de Fourier  $X(e^{j\omega})$ . Dado que  $X(e^{j\omega})$  es una función continua, es posible muestrearla tan fino como se desee. Una forma de hacerlo es añadiendo ceros a  $x[n]$  antes de calcular la DFT. En este punto vamos a comprobar cómo aumenta la densidad de muestras en frecuencia al añadir ceros antes de tomar la DFT.

- Genere un vector **x** de 16 números aleatorios. Calcule su  $\text{FFT}_{16}$  y llame al resultado **X**.
- Genere un vector **x2** de longitud 32 de forma que los 16 primeros valores se correspondan con **x** y el resto sea cero. Calcule la  $\text{FFT}_{32}$  de **x2** y llame al resultado **X2**. Compare los valores de **X** con los de **X2**. Compruebe que algunos coinciden. ¿Cuáles? ¿Por qué?

- Genere ahora un vector **x3** de longitud 32 de modo que sus primeros 16 valores sean los de **x** y los segundos nuevamente los de **x**. Calcule su  $\text{FFT}_{32}$  y compárela con **X**. ¿Qué observa?

### 3. Determinación de la respuesta en frecuencia de un filtro

En anteriores prácticas para determinar la respuesta en frecuencia de un filtro se ha utilizado la función **freqz** de MATLAB. En este apartado se va a programar una nueva función para obtener la respuesta en frecuencia de un filtro, tanto FIR como IIR, utilizando FFTs.

#### 3.1. Respuesta en frecuencia de un filtro FIR

La función a programar será **[H,f]=respfreqfir(h,N)** donde :

- **h** es la respuesta impulsional del filtro.
- **N** es el número de puntos para el cálculo de la FFT.
- **H** será la respuesta en frecuencia del filtro.
- **f** será el vector de frecuencias discretas para las que corresponde cada valor de la respuesta. Dicho vector deberá estar ordenado de  $-0,5$  a  $0,5$ , por lo que el coeficiente  $k = 0$  del vector **H** debe corresponder a la frecuencia  $-0,5^1$ .

El programa deberá estar compuesto por las siguientes partes:

1. Cálculo de la fft del vector **h** con el número de puntos indicado **N**. Previamente convendrá comprobar si **N** es potencia entera de 2, y si no lo es habrá que sustituir **N** por un nuevo valor **N<sub>1</sub>** que sea igual a potencia de 2 inmediatamente superior a **N**.
2. Ordenación del resultado anterior para que la respuesta en frecuencia pedida quede representada de  $[-0,5 \dots 0,5]$ .
3. Cálculo del vector de frecuencias **f**.

### Experimentación

Vamos a obtener la respuesta en frecuencia de un filtro FIR que representa un canal telefónico, cuyos coeficientes están almacenados en el fichero **h.mat**.

---

<sup>1</sup>Recuerde que en MATLAB los vectores comienzan con el índice 1

- Cargue en MATLAB el fichero `h.mat` que contiene la respuesta impulsional del filtro del cual queremos calcular su respuesta en frecuencia.
- Ejecute el programa realizado introduciendo como parámetros  $h$ , y un  $N = 1024$ , obteniendo en la variable  $H$  la respuesta en frecuencia deseada y en  $f$  el vector de frecuencias discretas.
- Represente el módulo de  $H$  en función de frecuencias analógicas. Para pasar el vector  $f$  a frecuencias analógicas multiplique el vector  $f$  por la frecuencia de muestreo ( $f_s = 16KHz$ ). ¿Se corresponde el módulo de  $H$  con el resultado que esperaba obtener?.

### 3.2. Respuesta en frecuencia de un filtro IIR

Para determinar la respuesta en frecuencia de un filtro IIR el método a seguir no puede ser el mismo que para filtros FIR, porque la  $h[n]$  es infinita. Sin embargo, se puede calcular utilizando la  $H(z)$ .

La función a programar será `[H,f]=respfreciir(B,A,N)` donde :

- B y A son los coeficientes del numerador y del denominador, respectivamente, de la  $H(z)$  del filtro.
- N es el número de puntos para el cálculo de la FFT.
- H será la respuesta en frecuencia del filtro.
- f será el vector de frecuencias discretas para las que corresponde cada valor de la respuesta. Dicho vector deberá estar ordenado de  $-0,5$  a  $0,5$ , por lo que el coeficiente  $k = 0$  del vector  $H$  debe corresponder a la frecuencia - 0.5.

El programa, al igual que en el caso de un filtro FIR, deberá estar compuesto por las siguientes partes:

1. Cálculo de las ffts de los vectores B y A, con el número de puntos indicado  $N$ . Previamente convendrá comprobar si  $N$  es potencia entera de 2, y si no lo es habrá que sustituir  $N$  por un nuevo valor  $N_1$  que sea igual a potencia de 2 inmediatamente superior a  $N$ . División punto a punto de la fft de B entre la de A.
2. Ordenación del resultado anterior para que la respuesta en frecuencia pedida quede representada de  $[-0,5 \dots 0,5]$ .
3. Cálculo del vector de frecuencias  $f$ .

Para probar la nueva función determine la respuesta en frecuencia del filtro IIR que tiene como  $H(z)$  la siguiente función:

$$H(z) = \frac{0,1667 + 0,5 z^{-1} + 0,5 z^{-2} + 0,1667 z^{-3}}{1 + 0,33 z^{-2}}$$

Represente el módulo de la respuesta en frecuencia y compruebe que se trata de un filtro paso bajo de ganancia unidad en la banda de paso y frecuencia de corte de 0.25.

## 4. Filtrado lineal usando DFT's

Una aplicación muy interesante de la DFT es su uso en el filtrado FIR de señales. Se realizará un programa que implemente el método *solape y almacenamiento* (*overlap-save*) de filtrado de señales muy largas.

### El método Solape y Almacenamiento de filtrado

Se pretende implementar el método solape y almacenamiento (*overlap-save*) de filtrado FIR de una secuencia utilizando la FFT.

Programa una función de MATLAB `oversave(x,h)` que realice el filtrado lineal de la secuencia `x` con un filtro FIR de respuesta impulsiva `h`. La longitud de las ventanas deberá ser potencia de 2 para una máxima eficiencia del algoritmo. La longitud de la ventana se tomará el máximo entre 512 y la potencia de 2 inmediatamente superior al doble de la longitud de `h`.

El vector resultado tendrá por primera muestra la primera muestra no nula de la secuencia convolución, y por última muestra la última muestra no nula de la secuencia convolución. Es decir, su función debe devolver un resultado *idéntico* al de la función `conv()` de MATLAB<sup>2</sup>.

Su función deberá funcionar con vectores fila o columna en `x` y `h`.

**NOTA:** Conviene no usar vectores que crezcan en las distintas iteraciones de un bucle, pues esto ocasiona problemas de memoria. Hay que prever cual será el tamaño del vector tras la última iteración y reservarlo primero (con una instrucción `zeros()` de MATLAB, por ejemplo).

## Experimentación

Para demostrar la eficiencia computacional del filtrado FIR utilizando FFTs vamos a filtrar una señal muy larga (audio+ruido) con el filtro de respuesta `h` que teníamos en el apartado anterior. En primer lugar debe insertar en la función `oversave()` instrucciones para medir el número de operaciones<sup>3</sup> y el tiempo empleado en la ejecución. Después, deberá realizar una nueva función `miconv()` con los mismos argumentos y salidas que la `conv()` de MATLAB y que sea una llamada a ésta junto con la inserción de las instrucciones de medida. Además, aprovechando que la función `fftfilt()` implementa el método de solape y suma (*overlap-add*), deberá implementar una nueva función `overadd()` que use este método junto con las instrucciones de medida. De esta forma podremos determinar el número de operaciones y el tiempo de ejecución que se necesitará para filtrar la señal uti-

---

<sup>2</sup>Es posible, que la longitud del vector `x` haga necesario añadir ceros en la última ventana para que todas las ventanas tengan la misma longitud. Dichos ceros deberán ser eliminados del resultado. Observe que en *la vida real* dicho problema no existe pues la señal es de duración casi infinita.

<sup>3</sup>No es posible medir el número de operaciones para versiones de MATLAB superiores a la 6.

lizando la convolución lineal o la circular (con los métodos: solapamiento-almacenamiento y solapamiento-suma.).

- Cargue el archivo `audio1.wav` utilizando la función `wavread`.
- Filtre la señal utilizando la convolución lineal (`miconv()`) obteniendo la señal filtrada  $y_1$ . ¿Cuánto tiempo y cuántas operaciones han sido necesarias?
- Filtre la señal utilizando la convolución circular (`oversave()`) obteniendo la señal  $y_2$ <sup>4</sup>. ¿Cuánto tiempo y cuántas operaciones han sido necesarias ahora?
- Filtre la señal utilizando la convolución circular (`overadd()`) obteniendo la señal filtrada  $y_3$ . ¿Cuánto tiempo y cuántas operaciones han sido necesarias?
- Convierta las señales ( $y_1$ ,  $y_2$  e  $y_3$ ) en archivos '.WAV' utilizando la función `wavwrite`<sup>5</sup>.
- Escuche el audio original y las señales que acaba de obtener. ¿Qué diferencias observa entre las tres? Comente los resultados.

## 5. Resumen

En esta práctica se ha profundizado en los siguientes aspectos:

- Obtención de muestras en frecuencia.
- Determinación de la respuesta en frecuencia de un filtro.
- Implementación del método *overlap-save*.
- Filtrado de señales para eliminar ruido.

---

<sup>4</sup>Sería conveniente que calculase la parte real de  $y_2$ , ya que al hacer la IFFT en el programa `oversave()` pueden aparecer partes imaginarias sin sentido muy próximas a cero

<sup>5</sup>La frecuencia de muestreo es de 16 KHz.