

# Laboratorio de Procesado Digital de Señal - 3º GITT

## Práctica 3: filtros digitales FIR

En esta práctica se va a trabajar con diferentes técnicas para aplicar filtros FIR a una señal y se van a analizar diferentes conceptos de este tipo de filtros.

Se facilita al alumno la amplitud de la señal  $x(t)$  muestreada (vector  $\mathbf{x}$ ) en función del tiempo (vector  $\mathbf{t}$ ); y los coeficientes  $b_k$  (vector  $\mathbf{b}$ ) de un filtro FIR diseñado por el profesor.

Programa en Matlab los siguientes apartados, comentando el código.

Al finalizar la práctica, suba a la plataforma web de la asignatura (Moodle) un archivo PDF, en el que se responda a los apartados de la práctica y en el que se presenten las figuras que se piden; y un archivo .m, con el código de Matlab con el que se obtienen los resultados.

### Filtrado de señales

En este apartado se va a realizar el filtrado de la señal  $x(t)$ , facilitada por el profesor, empleando, para ello, distintas técnicas.

Los filtros FIR pueden implementarse de distintas formas, tal y como se verá a continuación (y como se verá más adelante en el curso, en futuras prácticas). La primera de ellas consiste en implementar, manualmente, la operación de convolución **lineal** mediante la suma de los resultados de multiplicar las muestras de la señal por los coeficientes del filtro, de tal forma que:

$$y[n] = \sum_{k=1}^{M+1} b_k \cdot x[n-k]$$

Donde  $y[n]$  es la muestra n-ésima filtrada;  $M$  es el orden del filtro FIR;  $b_k$  son los coeficientes del filtro; y  $x[n]$  son las muestras de la señal original.

Otra forma de realizar la convolución **lineal** consiste en emplear la función que Matlab tiene a tal efecto:

$$\mathbf{y} = \text{conv}(\mathbf{b}_k, \mathbf{x}[\mathbf{n}]);$$

Consulte la ayuda de Matlab para conocer el significado de cada uno de los argumentos.

Por último, se puede emplear la función de filtrado que implementa Matlab, la cual tiene los siguientes argumentos:

$$\mathbf{y} = \text{filter}(\mathbf{b}_k, 1, \mathbf{x}[\mathbf{n}]);$$

Consulte la ayuda de Matlab para conocer el significado de cada uno de los argumentos.

A partir de la señal facilitada al alumno, realice los siguientes apartados, respondiendo a las preguntas que se plantean:

- Indique la frecuencia de muestreo ( $f_s$ ) de la señal facilitada ( $x(t)$ ).
- Filtre la señal  $x[n]$  con el filtro FIR facilitado, calculando el resultado ( $y[n]$ ) manualmente, es decir, calculando el sumatorio indicado anteriormente. Tenga en cuenta que, al principio, las muestras  $x[n - k] = 0$  mientras  $n \leq k$ , con  $n \geq 1$ .
- Filtre la señal  $x[n]$  con el filtro FIR facilitado, calculando el resultado ( $g[n]$ ) mediante la convolución.
- Filtre la señal  $x[n]$  con el filtro FIR facilitado, calculando el resultado ( $h[n]$ ) mediante la aplicación de filtros en Matlab.
- Analice, en el dominio del tiempo, las diferencias entre los resultados obtenidos (señales filtradas  $y[n]$ ,  $g[n]$  y  $h[n]$ ), y respecto de la señal original  $x[n]$ . Exponga y justifique gráficamente las conclusiones extraídas de dicho análisis. Preste especial atención al vector de tiempo de cada una de las señales.
- ¿Cuánto es, en milisegundos, el retardo (transitorio) del filtro para cada uno de los casos? ¿Con qué parámetro del filtro tiene relación este retardo?
- Analice, en el dominio de la frecuencia, las diferencias entre los resultados obtenidos (señales filtradas  $y[n]$ ,  $g[n]$  y  $h[n]$ ), y respecto de la señal original  $x[n]$ . Exponga y justifique gráficamente las conclusiones extraídas de dicho análisis. Preste especial atención al rango de frecuencia de cada una de las señales.

## Diseño de filtros FIR

En la vida real, para diseñar un filtro FIR se emplean herramientas que calculan los coeficientes de forma fácil y rápida para el usuario.

Matlab tiene una herramienta gráfica (*Filter Design & Analysis Tool*) que permite diseñar todo tipo de filtros. Esta herramienta se abre escribiendo `fdatoool` en la ventana de Comandos de Matlab.

Una vez indicados los parámetros y el tipo de filtro, hay que pulsar en el botón **Design Filter** en la parte inferior de la ventana.

Para exportar los coeficientes del filtro diseñado hay que ir al menú **File** -> **Export** e indicar los valores deseados en la ventana que aparece.

A partir de la señal facilitada al alumno, realice los siguientes apartados, respondiendo a las preguntas que se plantean:

- Diseñe un filtro paso bajo** con las siguientes características:
  - Tipo de respuesta: Lowpass
  - Método de diseño: FIR – Constrained Equiripple
  - Orden del filtro: 100
  - Especificación de frecuencias:
    - $F_s$ : frecuencia de muestreo (a especificar por el alumno)
    - Especificación: cutoff

- $F_c: f_{cL}$
- Especificación de magnitudes:
  - $A_{pass} = 0,1$  dB
  - $A_{stop} = 80$  dB

La frecuencia de corte ( $f_{cL}$ ) ha de ser tal que atenúe en más de 80 dB los dos armónicos fundamentales de mayor frecuencia de  $x(t)$ , y altere lo menos posible (menos de 3 dB) el resto de armónicos. Indique la frecuencia de corte ( $f_{cL}$ ) del filtro diseñado.

b) Justifique el correcto diseño del filtro mediante las gráficas que considere oportunas.

c) **Diseñe un filtro paso alto** con las siguientes características:

- Tipo de respuesta: Highpass
- Método de diseño: FIR – Constrained Equiripple
- Orden del filtro: 100
- Especificación de frecuencias:
  - $F_s$ : frecuencia de muestreo (a especificar por el alumno)
  - Especificación: cutoff
  - $F_c: f_{cH}$
- Especificación de magnitudes:
  - $A_{stop} = 80$  dB
  - $A_{pass} = 0,1$  dB

La frecuencia de corte ( $f_{cH}$ ) ha de ser tal que atenúe en más de 80 dB la componente continua y los dos armónicos fundamentales de menor frecuencia de  $x(t)$ , y que altere lo menos posible (menos de 3 dB) el resto de armónicos. Indique la frecuencia de corte ( $f_{cH}$ ) del filtro diseñado.

d) Justifique el correcto diseño del filtro mediante las gráficas que considere oportunas.

## Análisis de filtros

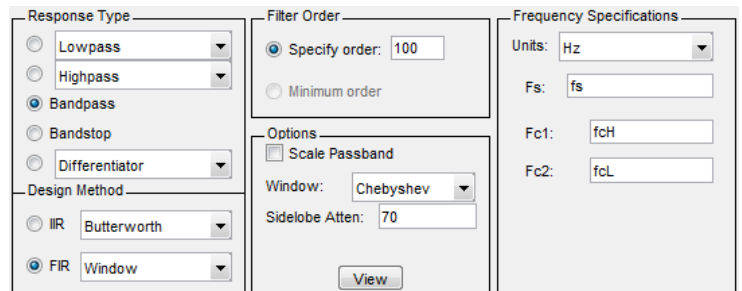
### Superposición

En este apartado se va a analizar el efecto de encadenar varios filtros.

A partir de la señal facilitada y de los resultados del bloque anterior, realice los siguientes apartados, respondiendo a las preguntas que se plantean:

- Empleando el filtro **paso bajo** diseñado en el bloque anterior, y empleando uno de los métodos vistos en el primer bloque de la práctica, filtre la señal  $x(t)$  y obtendrá la señal  $y[n]$ .
- Empleando el filtro **paso alto** diseñado en el bloque anterior, filtre la señal  $y[n]$  y obtendrá la señal  $g[n]$ .
- Diseñe un filtro paso banda** con las siguientes características:

- Tipo de respuesta: Bandpass
- Método de diseño: FIR – Window
- Orden del filtro: 100
- Opciones:
  - Window: Chebyshev
  - Sidelobe Atten: 70
- Especificación de frecuencias:
  - $F_s$ : frecuencia de muestreo (a especificar por el alumno)
  - $F_{c1}$ :  $f_{cH}$
  - $F_{c2}$ :  $f_{cL}$



- Filtre la señal  $x(t)$  con este filtro y obtendrá la señal  $h[n]$ .
- Analice, en el dominio de la frecuencia, las diferencias entre los espectros de  $x[n]$ ,  $y[n]$  y  $g[n]$ , prestando especial atención al rango de frecuencias de cada señal. Exponga y justifique gráficamente las conclusiones extraídas de dicho análisis.
- Analice las diferencias entre los espectros de  $x[n]$ ,  $g[n]$  y  $h[n]$ , desde  $-f_s/2$  hasta  $f_s/2$ . Exponga y justifique gráficamente las conclusiones extraídas de dicho análisis.

### Orden del filtro

- Modifique el orden del **filtro paso bajo** diseñado previamente a valores de 10, 20 y 50.
- Analice los espectros en frecuencia de los cuatro filtros (órdenes 10, 20, 50 y 100). Para ello, emplee la función `freqz` de Matlab. Exponga y justifique gráficamente las conclusiones obtenidas del análisis.
- ¿Cuántos milisegundos de retardo introduce cada uno de los cuatro filtros a la señal?

**NOTA:** Junto con el archivo PDF de respuestas de la práctica y el archivo .m de código de Matlab, suba a Moodle un archivo .mat con los coeficientes de todos los filtros diseñados en la práctica.

# Funciones de Matlab

Consulte la ayuda de Matlab para conocer el significado de cada uno de los argumentos.

| Miscelánea   |   |
|--|---|
| Borra todas las variables del Workspace  | <code>clear</code>                            |
| Cierra todas las figuras abiertas  | <code>close all</code>                        |
| Limpia la ventana de comandos  | <code>clc</code>                              |
| Carga en el Workspace las variables almacenadas en el archivo <code>nombre_archivo.mat</code>                                  | <code>load 'nombre_archivo.mat'</code>        |
| Devuelve la longitud del vector <b>x</b>   | <code>y = length(x);</code>                   |
| Devuelve una matriz de ceros de tamaño <b>a</b> filas y <b>b</b> columnas  | <code>y = zeros(a, b);</code>                 |
| Muestra el mensaje de <b>texto</b> por pantalla  | <code>disp('texto');</code>                   |
| Convierte en string (cadena de caracteres) el número <b>x</b>  | <code>y = num2str(x);</code>                  |
| Representación   |   |
| Abre una nueva ventana para representar gráficamente   | <code>figure;</code>                          |
| Representación en ejes x-y uniendo los puntos  | <code>plot(ejex, ejey, '-.');</code>          |
| Congela la figura activa para poder superponer más representaciones  | <code>hold on;</code>                         |
| Activa la rejilla de la representación   | <code>grid on;</code>                         |
| Pone título a la representación  | <code>title('Texto');</code>                  |
| Pone etiqueta en eje <b>x</b> . La función <b>ylabel</b> hace lo mismo en el eje <b>y</b>                                      | <code>ylabel('Texto');</code>                 |
| Representa el eje <b>x</b> entre los valores <b>ini</b> y <b>fin</b> . La función <b>ylim</b> hace lo mismo en el eje <b>y</b> | <code>xlim([ini fin]);</code>                 |
| Muestra una leyenda en la figura, donde se muestran los textos indicados   | <code>legend('Texto1', 'Texto2', ...);</code> |

| Audio   |   |
|---|---|
| Lee el archivo de audio <b>file</b>   | <code>[y, fs] = audioread('file');</code>       |
| Reproduce la señal de audio <b>y</b> a una frecuencia de muestreo <b>fs</b>   | <code>sound(y, fs);</code>                      |
|   |   |
| Genera el vector de valores <b>v</b> entre <b>ini</b> y <b>fin</b> equiespaciados <b>delta</b>                      | <code>v = ini : delta : fin;</code>             |
| Genera el vector <b>v</b> de <b>N</b> valores equiespaciados entre <b>ini</b> y <b>fin</b> (inclusive)              | <code>v = linspace(ini, fin, N);</code>         |
| Genera el vector <b>v</b> con los elementos del vector <b>x</b> desde el <b>a</b> hasta el <b>b</b> (inclusive)     | <code>v = x(a:b);</code>                        |
| Concatena los vectores fila <b>x</b> e <b>y</b>   | <code>v = [x , y];</code>                       |
| Concatena los vectores columna <b>x</b> e <b>y</b>  | <code>v = [x ; y];</code>                       |
| Matemáticas   |   |
| Calcula el valor absoluto del número <b>x</b> , o el módulo de éste si es un número complejo                        | <code>y = abs(x);</code>                        |
| Calcula la transformada de Fourier del vector <b>x</b>  | <code>X = fft(x, length(x)) / length(x);</code> |
| Trasposición del espectro en frecuencias <b>X</b>   | <code>X = fftshift(abs(X));</code>              |
| Calcula el sumatorio de todos los valores del vector <b>x</b>   | <code>y = sum(x);</code>                        |
| Multiplica, elemento a elemento, los vectores/matrices <b>x</b> e <b>y</b> , los cuales deben tener el mismo tamaño | <code>z = x .* y;</code>                        |
| Calcula la potencia de la señal <b>x</b>  | <code>y = var(x);</code>                        |