Guía N° 6: *Introducción a los Filtros FIR*

Objetivos:
Comprender los detalles de implementación de un filtro FIR

Bibliografía recomendada:

Título: **The Scientist and Engineer's Guide to Digital Signal Processing.**
Autor: Steven W. Smith.
Editorial: California Technical Publishing.
(Este libro puede ser obtenido en formato electrónico del sitio web: www.dspguide.com)

Título: **Tratamiento de Señales en Tiempo Discreto.**
Autores: Oppenheim – Schafer – Buck.
Editorial: Pentice Hall.

Título: **Tratamiento Digital de Señales.**
Autores: Proakis - Manolakis.
Editorial: Pentice Hall.

Título: **The Student Edition of MATLAB.**
Autores: Hanselman – Littlefield.
Editorial: Prentice Hall

Enunciado:

6 Filtros de respuesta finita al impulso - FIR

El filtro FIR es una aplicación directa de la operación matemática de convolución, que permite combinar una secuencia de entrada con una secuencia de coeficientes para producir una secuencia de salida con características de señal diferentes. Estas características que se modifican se definen a partir del conjunto de coeficientes que componen el filtro.

Suponiendo una secuencia de entrada *x[n]*, de longitud *L* y un filtro con secuencia *h[n]*, con longitud *N*, la convolución entre ambas es:

$$y[n]=h[n]*x[n]$$
$$y[n]=\sum_{i=0}^{i=L+N-1} h[i]x[n-i]$$

A la secuencia *h[n]* se la conoce como coeficientes, taps o kernel del filtro y es exactamente igual a la respuesta al impulso del filtro.

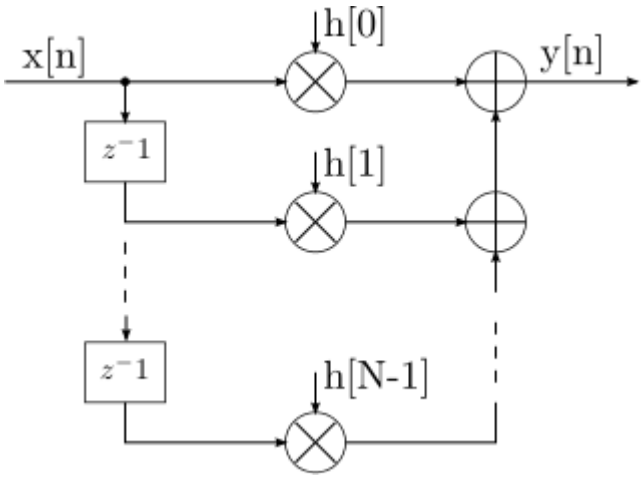
La expresión expandida de la ecuación anterior es

$$y[n]=h[0]x[n]+h[1]x[n-1]+\ldots+h[N-1]x[n-N+1]$$

La transformada z de esta expresión es:

$$Y[z]=\left[h[0]X[z]+h[1]z^{-1}X[z]+\ldots+h[N-1]z^{N-1}X[z]\right]$$

Lo que nos lleva al diagrama del filtro FIR



Por último la función de transferencia es

$$\frac{Y[z]}{X[z]}=h[0]+h[1]z^{-1}+\ldots+h[N-1]z^{N-1}$$

6.1 Convolución muestra a muestra mediante bucle FOR

6.1.1 Crear un programa que permita realizar la convolución entres una señal de *x* longitud *L* con un kernel *h* de longitud *N* utilizando un bucle FOR, de acuerdo a la ecuación de la convolución y donde

$$h[n]=\begin{cases}1 & \text{para } 0\le n<N\\ 0 & \text{de otro modo}\end{cases}$$

$$x[n]=\begin{cases}e^{-\frac{n}{2}} & \text{para } 0\le n<L\\ 0 & \text{de otro modo}\end{cases}$$

6.1.2 Como la señal resultante tiene *L+N-1* muestras, inicializar un vector *y* de 1 fila y *L+N-1* columnas, utilizando la función **zeros**.

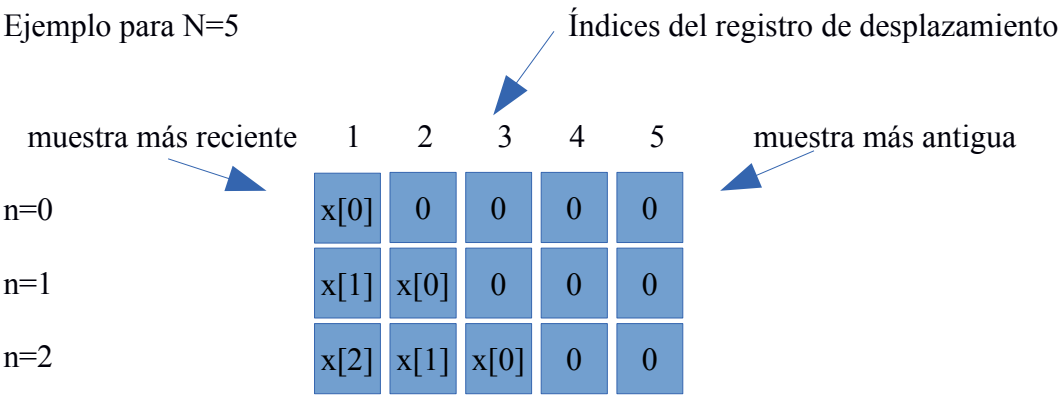
6.1.3 Inicializar a cero un vector de tamaño *N*. Este vector funcionará como registro de desplazamiento del filtro, también conocido como línea de retardo (delay line).

6.1.4 Extender el vector *x* con *N* muestras en cero al final del vector utilizando corchetes, de la siguiente manera:

```
[ x zeros(1,N) ]
```

6.1.5 Crear un bucle FOR que itere las *L+N-1* muestras de salida, con índice *n*.

6.1.6 En cada iteración, actualizar el registro de desplazamiento con las *N-1* muestras mas actuales del registro de desplazamiento y la muestra actual de la señal *x*, leída utilizando el índice *n* del FOR.



6.1.7 Agregar otro bucle FOR, interno al bucle FOR anterior y ubicado luego de la actualización del registro de desplazamiento, que barra N ciclos y con índice *i*.

6.1.8 En cada iteración del bucle anterior multiplicar una muestra del registro de desplazamiento con una muestra del filtro *h* y acumular el producto en la muestra *y[n]*.

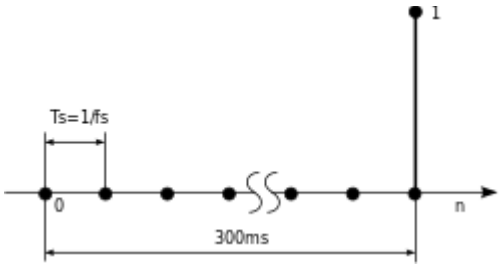
6.1.9 Luego de calcular todas las muestras de salida, graficar el vector *y. S*obre el mismo gráfico, graficar la convolución implementada con la función de Matlab **conv**, y verificar que los resultados sean exactamente iguales.

6.1.10 Reemplazar el bucle FOR interno por una multiplicación matricial de una matriz de 1 Fila x N columnas con otra matriz de N filas y 1 columna. El operador tilde **'** se puede utilizar para transponer los vectores de ser necesario.

6.2 Convolución con filtro de retardo

6.2.1 Leer el archivo *numeros.wav,* disponible en autogestión, mediante la función **wavread** y guardar la señal resultante en una variable *r*. Obtener al mismo tiempo con esta función la frecuencia de muestreo y cantidad de bits del archivo wav. Utilizar la ayuda de ser necesario.

6.2.2 Crear una señal *h* que consista en un impulso desplazado 300ms con respecto al origen, como se muestra en la siguiente figura:



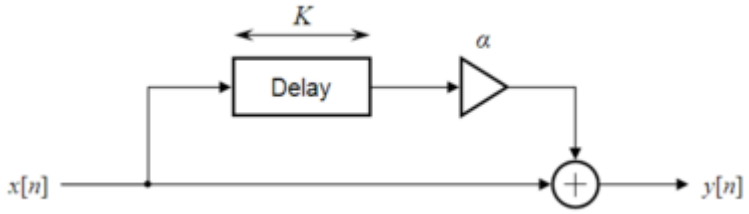
Utilice la frecuencia de muestreo para calcular la cantidad de muestras que equivalen a dicho retardo.

6.2.3 Convolucionar la señal *r* con la señal *h* para crear una señal *l*. Eliminar las ultimas muestras de la señal *l* de manera que las señales *l* y *r* tengan la misma longitud. Combinar ambas señales en una matriz de dos columnas mediante la operación **[r , l]** .

6.2.4 Escribir la señal resultante del punto anterior a un archivo .wav utilizando la función **wavwrite**. Utilizar la misma definición de frecuencia de muestreo y cantidad de bits de la señal original. Reproducir con un reproductor de audio para comprobar el resultado.

6.3 Filtro Peine (Comb Filter)

El filtro peine posee una estructura como la de la figura:



Se puede expresar la salida como $y[n]=x[n]+a\,x[n-K]$

y posee la siguiente respuesta al impulso:

$$h[n]=\begin{cases}1 & n=0\\ a & n=K\\ 0 & \text{de otro modo}\end{cases}$$

6.3.1 Crear un filtro peine de acuerdo a la siguiente ecuación

$$y[n]=x[n]+\sum_{i=1}^{N} a^i x[n-iK]$$

Calcular K para que cada retardo individual equivalga a 10ms. Calcular N de manera que el mayor retardo (para $i=N$) equivalga a un retraso total de 100ms. Calcular el valor de a para que cada versión retrasada de $x[n]$ posea 3dB menos de potencia que la anterior.

6.3.2 Convolucionar la señal leída en el punto 6.2.1 con el filtro. Normalizar la amplitud de la señal para que el máximo valor sea menor que 1 y grabar el resultado con la función **wavwrite**.

6.3.3 Repetir los últimos dos puntos para retardos individuales de 20 y 30ms.

6.4 Filtro Diferenciador

En tiempo discreto, la derivada de una señal se obtiene haciendo la diferencia entre sus muestras, de la siguiente forma, para las primeras tres derivadas:

$$\dot{x}[n]=x[n]-x[n-1]$$

$$\ddot{x}[n]=\dot{x}[n]-\dot{x}[n-1]=x[n]-2x[n-1]+x[n-2]$$

$$\dddot{x}[n]=\ddot{x}[n]-\ddot{x}[n-1]=\dot{x}[n]-2\dot{x}[n-1]+\dot{x}[n-2]=x[n]-3x[n-1]+3x[n-2]-x[n-3]$$

6.4.1 Obtener los kernel de los filtros a partir de las ecuaciones mostradas y filtrar la señal del punto 6.2.1 con los 3 filtros resultantes. Grabar la salida para cada uno e identificar de manera cualitativa escuchando la señal resultante el efecto del filtro sobre el espectro de frecuencias de la señal. Indicar las observaciones con comentarios en el código del script.