Guía N° 4: *Diseño de filtros - Filtros IIR*

Objetivo:

Diseñar filtros FIR e IIR a medida.

Fecha de entrega:

2/08/2019

Formato de entrega:

Tanto Matlab como Octave son entornos donde se pueden hacer operaciones directamente por consola y también crear programas en archivos. En este práctico algunas operaciones se realizarán por consola, esas operaciones luego deberán ser copiadas a un programa para su entrega.

Cada ejercicio se corresponde a un archivo .m, con nombre e*N_M.m*, donde N es el nombre de la guía y M el número de ejercicio. El contenido de cada archivo .m debe contener resuelto el último punto de cada ejercicio, ignorando aquellos puntos intermedios que no sean necesarios para completar el punto final.

Todos los archivos se deberán comprimir en un solo archivo .zip o .rar y enviarse por correo con título *Entrega Guía 4*

Paquetes para octave:

Octave posee muchos paquetes que generalmente no vienen incluídos cuando lo instalamos.

Paquetes que pueden ser de utilidad para este práctico.

Control:

Este paquete, posee entre otras cosas la función **mag2db**

https://octave.sourceforge.io/control/function/mag2db.html

Geometry:

Este paquete, otorga la posibilidad de dibujar “flechas” en nuestros gráficos con la función **drawArrow**.

https://octave.sourceforge.io/geometry/function/drawArrow.html

Signal:

Este paquete contiene las ventanas que utilizaremos posteriormente.

https://octave.sourceforge.io/signal/

Como instalar paquetes:

Para instalar paquetes se utiliza una herramienta que viene integrada en octave, la misma se llama **pkg**. Por ejemplo:

```
pkg install -forge geometry
pkg install -forge control
```

Utilizar paquetes:
Para poder utilizar los paquetes es necesario cargarlos en primer lugar. Esto se hace con el mismo comando:

```
pkg load control
pkg load geometry
```

Estos comandos de carga pueden agregarse al archivo .octaverc en el home del usuario para que la carga se haga al iniciar el programa.

Bibliografía recomendada:

The Scientist and Engineer's Guide to Digital Signal Processing

1. Respuesta en frecuencia de un filtro FIR

   El Filtro FIR está caracterizado completamente por sus coeficientes, que componen al mismo tiempo la respuesta temporal al impulso *h[n]*. De la teoría de señales, la transformada de Fourier de esta respuesta al impulso, es la respuesta en frecuencia del sistema *H[k]*. En tiempo discreto, dicha respuesta se obtiene utilizando la Transformada Discreta de Fourier (DFT) definida por la siguiente ecuación:

$$H[k] = \sum_{n=0}^{N-1} h[n] e^{-\frac{2\pi i}{N} kn}$$

   Esto puede interpretarse como la correlación de *N* muestras de la secuencia *h[n]* con *N* exponenciales complejas de *N* muestras cada una.
   En matlab, podemos calcular la DFT fácilmente utilizando la función **fft.**
   Como salida tendremos otra secuencia de *N* muestras, donde los primeros *K/2* elementos corresponden a las frecuencias positivas y los siguientes *K/2* elementos a las frecuencias negativas.
   Cuando se habla de frecuencia en el dominio digital, se suele utilizar la frecuencia normalizada, conocida directamente como frecuencia digital.
   La definición de esta frecuencia digital depende de la convención utilizada. Las más usadas son las siguientes:

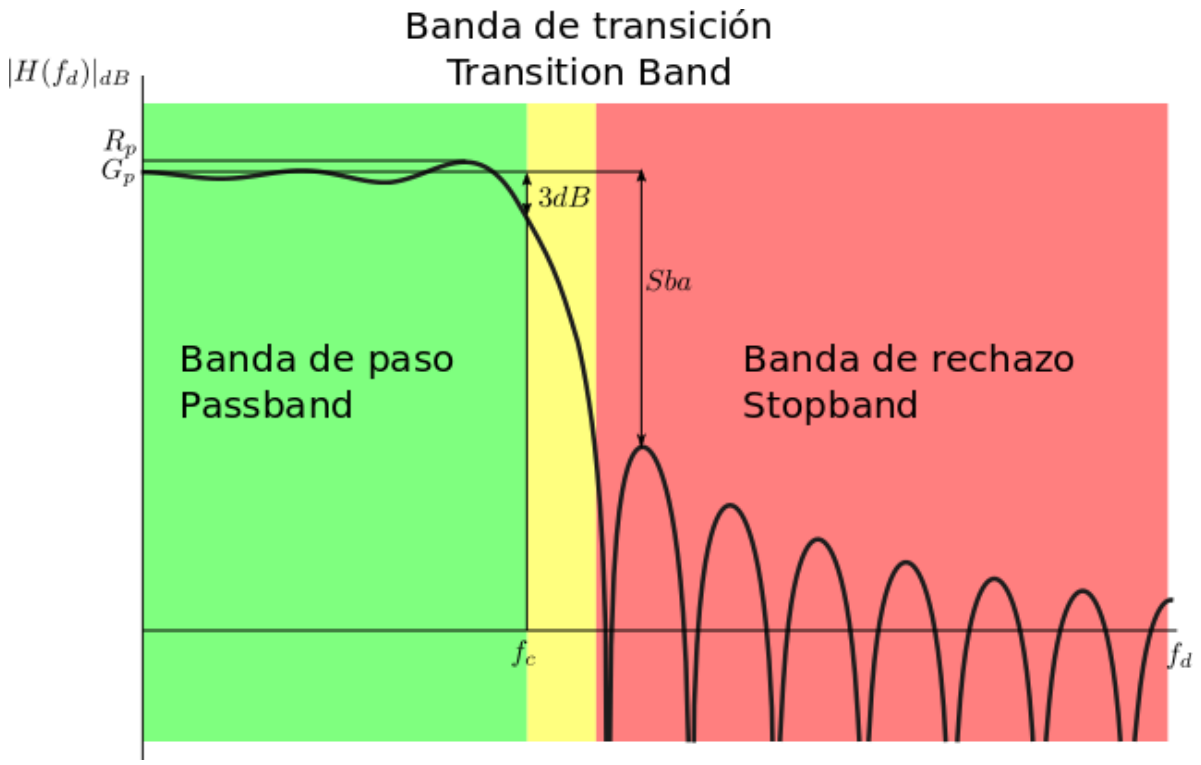
$$f_d = \frac{2f}{f_s} \qquad -1 \leq f_d \leq 1$$

$$f_d = \frac{2\pi f}{f_s} \qquad -\pi \leq f_d \leq \pi$$

$$f_d = \frac{f}{f_s} \qquad -\frac{1}{2} \leq f_d \leq \frac{1}{2}$$

Lo importante es que frente a una frecuencia digital, siempre hay que tener en cuenta la convención para poder relacionarla correctamente con la frecuencia de muestreo y de allí a la frecuencia real.
En nuestro caso, la convención que usaremos es la primera, por lo que una frecuencia digital de valor 1 significa una frecuencia real igual a la frecuencia de muestreo dividido 2.
Estas fórmulas son de extrema importancia, pues nos permite relacionar el dominio digital con el analógico.

Regiones y puntos de interés en la respuesta de un filtro
Al trabajar con filtros es necesario familiarizarse con la terminología empleada para definir las características de este.
En este se identifican tres tipos de zonas principales:
Banda de paso: Comprende la región de frecuencias de interés, aquellas cuyas amplitudes se busca mantener.
Banda de rechazo: Comprende la región cuyas frecuencias se busca eliminar.
Banda de transición: Región comprendida entre la banda de paso y banda de rechazo.
Además se identifican las siguientes características de interés:
Frecuencia de corte: Frecuencia en la que potencia de salida se reduce a la mitad (-3dB).
Ganancia en la banda de paso: Relación entre la amplitud de salida y la amplitud de entrada en región de interés.
Atenuación en la banda de rechazo: Relación entre la amplitud de salida y la amplitud de entrada en región de interés. También conocido como ripple en la banda de rechazo.
Ripple en la banda de paso: Variación máxima de la ganancia en la banda de paso.
Podemos observar estas definiciones en la siguiente figura:



En el gráfico:
G_p: Ganancia en la banda de paso
R_p: Ripple en la banda de paso
f_c: Frecuencia de corte
Sba: Atenuación en la banda de rechazo
Es importante destacar que cuando se trata del diseño de filtros, estos parámetros se definen como requerimientos y serán especificaciones que buscaremos cumplir, mientras que al evaluar un filtro existente, trataremos de identificar estos parámetros desde la respuesta en frecuencia.

Representación de la respuesta en frecuencia de un filtro FIR pasa-bajos

1.1. Cargar el archivo low_pass.dat en un variable h, utilizando la función **load** y graficar la respuesta al impulso del filtro mediante la función **stem.**

1.2. Obtener la DFT de la respuesta al impulso mediante

```
H = fft(h,1000);
```

El segundo parámetro le pide a la función fft que calcule 1000 puntos de frecuencia. Como el vector h tiene menos de 1000 elementos, la función se encargará de completarlos con ceros antes de calcular la DFT. El resultado de utilizar más puntos es una mayor resolución en frecuencia.
Graficar la respuesta en frecuencia del filtro, teniendo en cuenta lo visto en la guía 2, calculando el valor absoluto de H mediante la función **abs** y convirtiéndolo a dB, presentando solamente las frecuencias positivas.
Expresar el eje x en frecuencia digital, de 0 a casi 1, teniendo en cuenta que el

total de puntos es 1000 y el esquema de presentación de la fft. Agregar etiquetas a los ejes con las funciones **xlabel** e **ylabel**, y el título mediante la función **title**.

1.3. Incorporar en el gráfico anterior los siguientes valores:

- Ripple en la banda de paso
- Ganancia en continua
- Atenuación en la banda de rechazo
- Frecuencia de corte

Para esto se puede usar la función **text**, que permite agregar texto en coordenadas específicas.

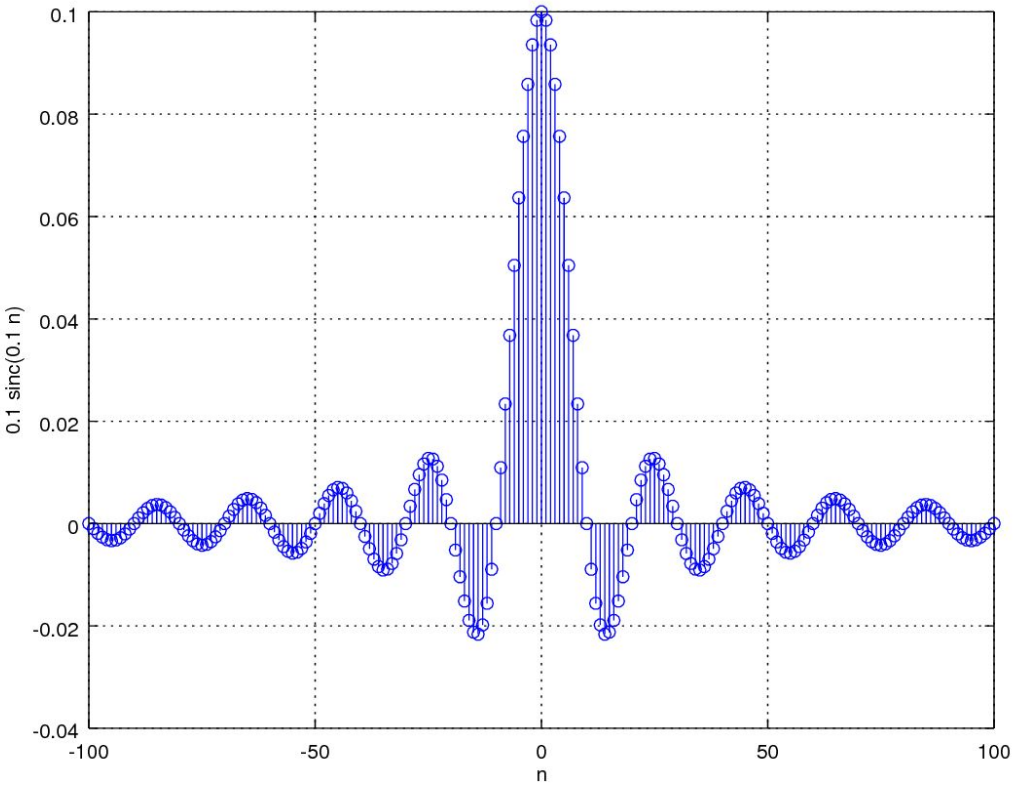
2. Método de la función sinc enventanada para el diseño de filtros FIR pasa bajos
Supongamos que queremos crear un filtro pasa-bajos ideal. Este filtro tendría la siguiente respuesta en frecuencia:

$$H(f) = \begin{cases} 1 & \text{si} \quad |f| \leq f_c \\ 0 & \text{si} \quad f_c \leq |f| < 1 \end{cases}$$

La clave entonces es encontrar aquella función en el tiempo cuya respuesta sea la función anterior. Se puede demostrar que esa función es la siguiente, denominada función *sinc*.

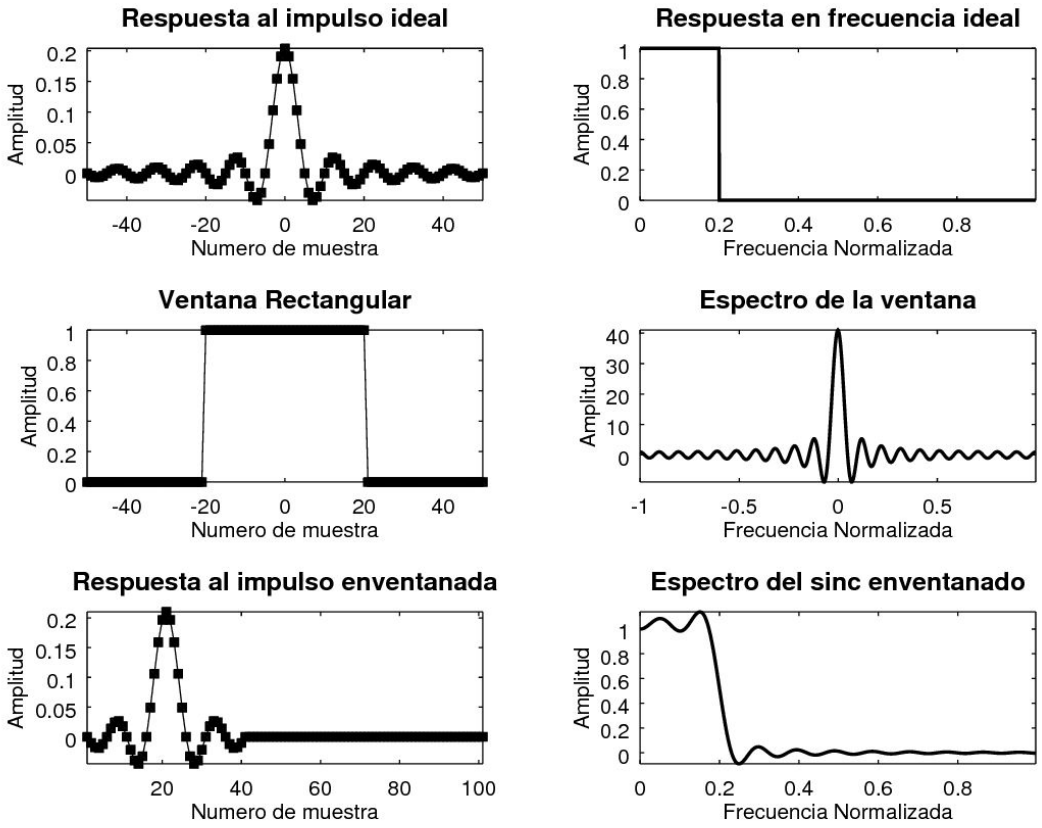
$$h[n] = f_c \,\mathrm{sinc}(f_c n) = \frac{\sin(f_c \pi n)}{\pi n}$$

Podemos ver en la siguiente figura un ejemplo de esta función para f_c = 0.1

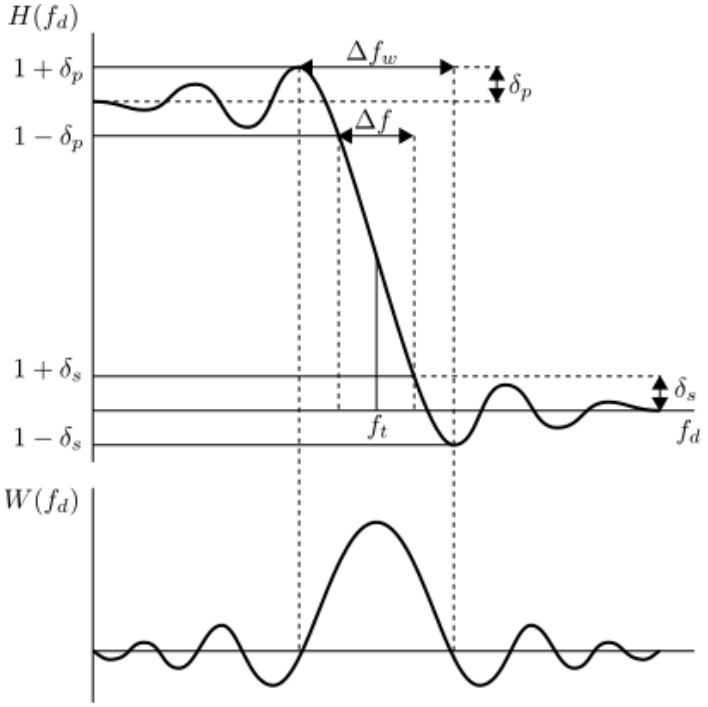


Esta secuencia tiene 2 problemas fundamentales
- Tiene infinita cantidad de coeficientes
- Es anticausal

Una primer solución a estos problemas consiste en truncar la respuesta de la función *sinc*, de manera que sólo tengamos *M* elementos distintos de cero. Al hacer esto luego podemos desplazar la respuesta en (*M-1)/2*, para que nuestro filtro sea causal. El inconveniente ahora es que la respuesta en frecuencia de nuestro filtro ya no será la respuesta de un filtro pasabajos ideal.
Para comprender lo que sucede con la respuesta en frecuencia, tengamos en cuenta que la operación de truncado es análoga a multiplicar en el tiempo la función *sinc* por una ventana rectangular. En frecuencia, esta multiplicación se convierte en una convolución, por lo que estaremos convolucionando las respuestas en frecuencia de la función *sinc* y de la ventana rectangular. ¿Cuál es la respuesta en frecuencia de una ventana rectangular? -> Una función sinc cuyo ancho de lóbulo principal es inversamente proporcional a *M*.
Resumiendo: truncar la secuencia original equivale a convolucionar la respuesta ideal cuadrada con una función *sinc*
Gráficamente:



Como vemos la respuesta de nuestro filtro truncado está lejos de ser ideal. Hemos introducido ripple, la atenuación en la banda de stop ya no es cero y ahora existe una banda de transición. Estos defectos surgen de las características de la respuesta en frecuencia de nuestra ventana rectangular.

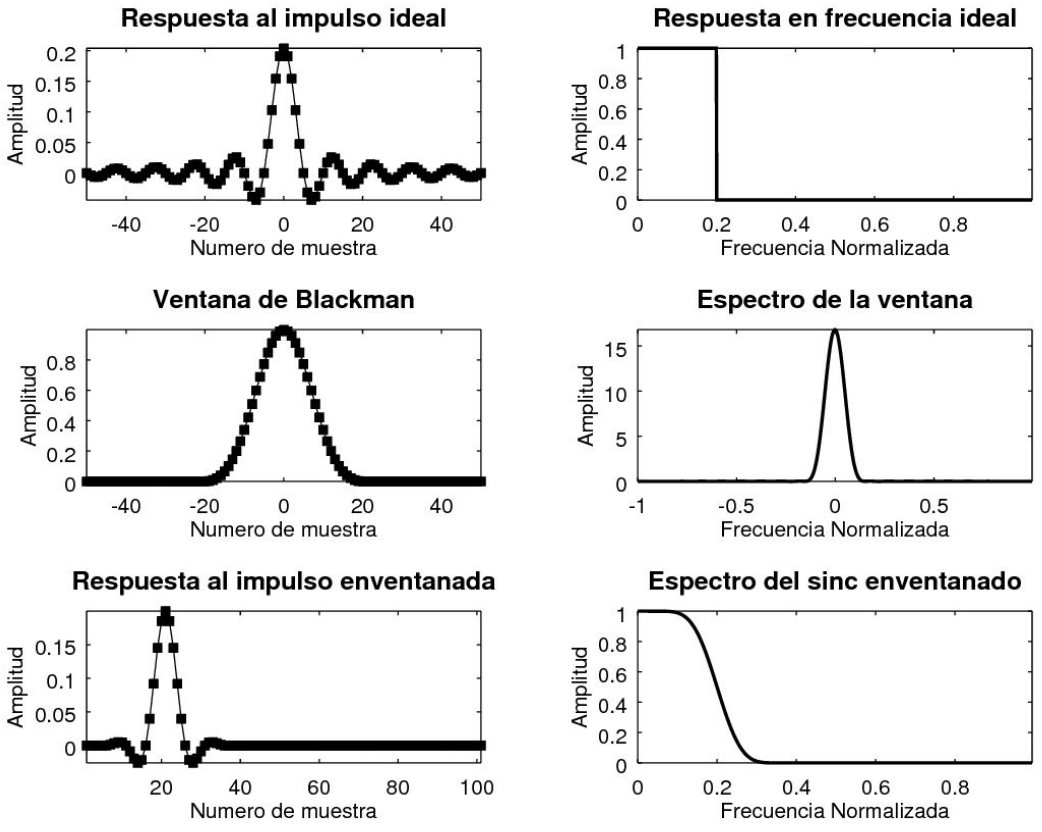


Podemos resumir las siguientes propiedades entre la ventana y el filtro resultante
- El ancho de banda de transición es igual a ambos lados de la frecuencia de corte ideal
- El error de aproximación (ripple) es igual en la banda de paso y la banda de rechazo
- La distancia entre picos de ripple es aproximadamente igual al ancho del lóbulo principal
- El tamaño del lóbulo principal es mayor que el ancho de banda de transición
- El ripple es determinado por la forma de la ventana y es independiente del orden del filtro

Lo que surge naturalmente de esta relación entre la forma de la ventana y la respuesta del filtro resultante es que existen otras ventanas[1] que en algunos parámetros del filtro dan mejores resultados que la ventana rectangular.
Un ejemplo es la ventana de blackman, descrita por la siguiente función:

$$w[n] = 0.42 - 0.5\cos\left(\frac{2\pi n}{M-1}\right) + 0.08\cos\left(\frac{4\pi n}{M-1}\right) \qquad 0 \leq n \leq L-1$$

Donde *L* es *M/2* para *M* par y *(M+1)/2* para *M* impar
Podemos ver el resultado del proceso con esta ventana en la siguiente figura:



Podemos ver que el filtro resultante tiene mejor atenuación en la banda de rechazo y mínimo ripple en la banda de paso, aunque el ancho de banda de transición es mayor.

[1] https://en.wikipedia.org/wiki/Window_function

Los pasos de diseño de un filtro mediante el método de la ventana entonces son

1) Determinar la frecuencia de corte ideal f_t haciendo el promedio entre la frecuencias de paso f_{pass} y la frecuencia de rechazo f_{stop}.
2) Determinar la ventana a usar mediante las especificaciones de ripple en la banda de paso y atenuación en la banda de rechazo. (ver tabla)
3) Determinar la cantidad M de taps a utilizar en el filtro de acuerdo al ancho de banda de transición $B = f_{stop} - f_{pass}$. (ver tabla)
4) Calcular M puntos de la función *sinc* para la frecuencia de corte ideal f_t.

$$h_{sinc}[n] = f_t * \text{sinc}\left(f_t\left(n - \frac{M-1}{2}\right)\right) \qquad 0 \leq n < M$$

5) Aplicar la ventana elegida multiplicando elemento a elemento.
6) Evaluar la respuesta en frecuencia del filtro y repetir los pasos de ser necesario.

Ventana	Ancho lóbulo principal	Lóbulo principal / secundario	Amplitud del mayor pico
Rectangular	4/M	-13dB	-21dB
Hanning	8/M	-32dB	-44dB
Hamming	8/M	-43dB	-53dB
Blackman	12/M	-58dB	-74dB

Podemos hacer uso de estas ventanas en Matlab/Octave mediante las siguientes funciones:

rectwin
hann
hamming
blackman
gausswin
bartlett
chebwin
blackmanharris
kaiser
flattopwin

Tener en cuenta que algunas de estas ventanas poseen parámetros extra de configuración. Para poder comparar las distintas ventanas se puede acceder al enlace de wikipedia.

2.1. Generar un filtro pasabajos con las siguientes especificaciones:
- f_{pass}: 0.1
- f_{stop}: 0.2

- Ventana: rectangular

Graficar coeficientes y respuesta en frecuencia (Expresar las frecuencias en frecuencias normalizadas y las amplitudes en dB).

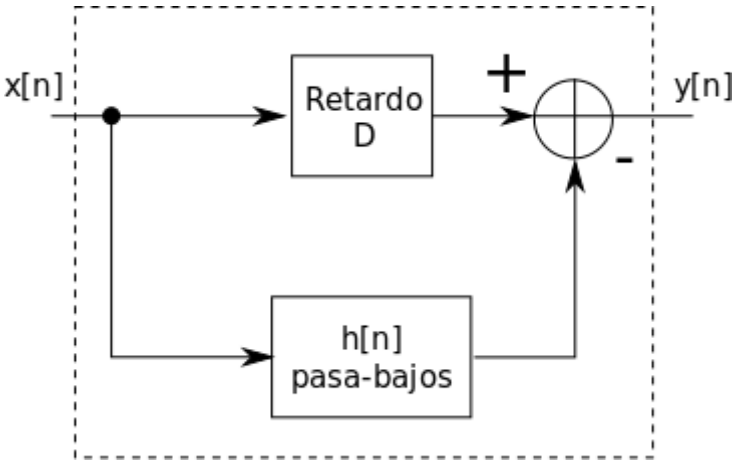
2.2. Generar un filtro pasabajos con las siguientes especificaciones

- fs: 44100Hz
- f_{pass}: 3000Hz
- f_{stop}: 4000Hz
- R_p: 0.5dB (max)
- Sba: 50dB (min)
- Mínima cantidad de taps

Graficar coeficientes y respuesta en frecuencia (Expresar las frecuencias en Hz y las amplitudes en dB).

3. Obtención de filtro pasa-altos a partir de filtro pasa bajos

3.1. Método de inversión de espectro: Se puede obtener una respuesta pasa-altos sustrayendo una respuesta pasa-bajos de una respuesta pasa-todo, como se ve en el siguiente diagrama:



El retardo es simplemente un impulso desplazado, el cual es efectivamente un filtro pasatodo.

Transformar el filtro del ejercicio 2.2 en pasa altos, asegurando primero una cantidad impar de coeficientes para que el retardo D resulte en un número entero.
Graficar ambas respuestas superpuestas indicando con el comando **legend** la curva de cada filtro.

3.2. Método de reversión de espectro: Se logra multiplicando los coeficientes del filtro por una secuencia de 1 y -1 alternados. Dicho de otro modo, se cambia el

signo de todas las muestras impares.

$$h_{pa}[n] = h_{pb}(-1)^n$$

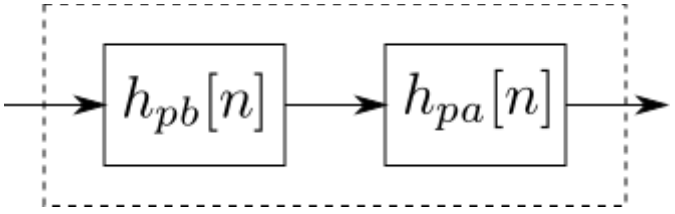
Esto revierte el espectro resultante, obteniendo un filtro pasa-altos con distinta frecuencia de corte.
Aplicar reversión de espectro al filtro pasa bajos de ejercicio 2.2 y graficar respuesta en frecuencia original y modificada.

3.3. Diseñar un filtro pasa-altos con las siguientes características:

- f_s: 44100Hz
- f_{pass}: 6100Hz
- f_{stop}: 6000Hz
- Sba: 80dB
- R_p: <0.1dB
- Se permite ripple en la banda de transición
  Usar cualquiera de los métodos y presentar la respuesta en frecuencia del filtro resultante en Hz.

4. Obtención de filtros pasa banda a partir de filtros pasa altos y pasa bajos

Se puede obtener un filtro pasa banda utilizando un filtro pasa bajos en serie con un pasa altos, como se ve en la siguiente figura:



Es importante que la frecuencia de corte del filtro pasa bajos sea mayor a la frecuencia de corte del pasa altos para que se solapen los espectros y existe una respuesta pasa banda, es decir

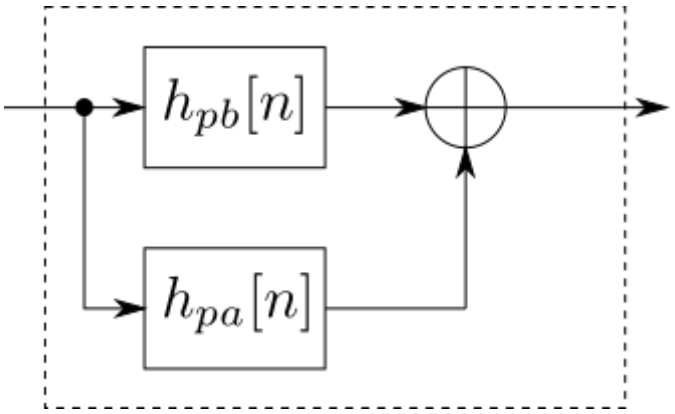
$$f_{cpb} > f_{cpa}$$

La obtención del kernel final se realiza convolucionando los kernels de los dos filtros.

4.1. Generar un filtro pasa banda con las siguientes especificaciones

- f_{c1}: 0.1
- f_{c2}: 0.2
- B: 0.01
- Sba: 60dB

5. Obtención de filtros rechaza banda a partir de filtros pasa altos y pasa bajos
Se puede obtener un filtro rechaza banda mediante la combinación de un filtro pasa bajos y un filtro pasa altos en paralelo



La obtención del kernel final se realiza sumando los kernels de los dos filtros

5.1. Diseñar el filtro rechaza banda utilizado en el ejercicio 5 de la guía 2 para remover un tono de interferencia de 1Khz de la señal audio_play.zip. Este filtro debe tener las siguientes especificaciones

- f_s: 44100Hz
- f_{c1}: 950Hz
- f_{c2}: 1050Hz
- Sba: > 60dB

Graficar la respuesta y generar una versión filtrada del audio mediante la función **filter**. Verificar escuchando el audio la remoción del tono, de lo contrario ajuste los parámetros del filtro y vuelva a probar.

6. Diseño óptimo de filtros

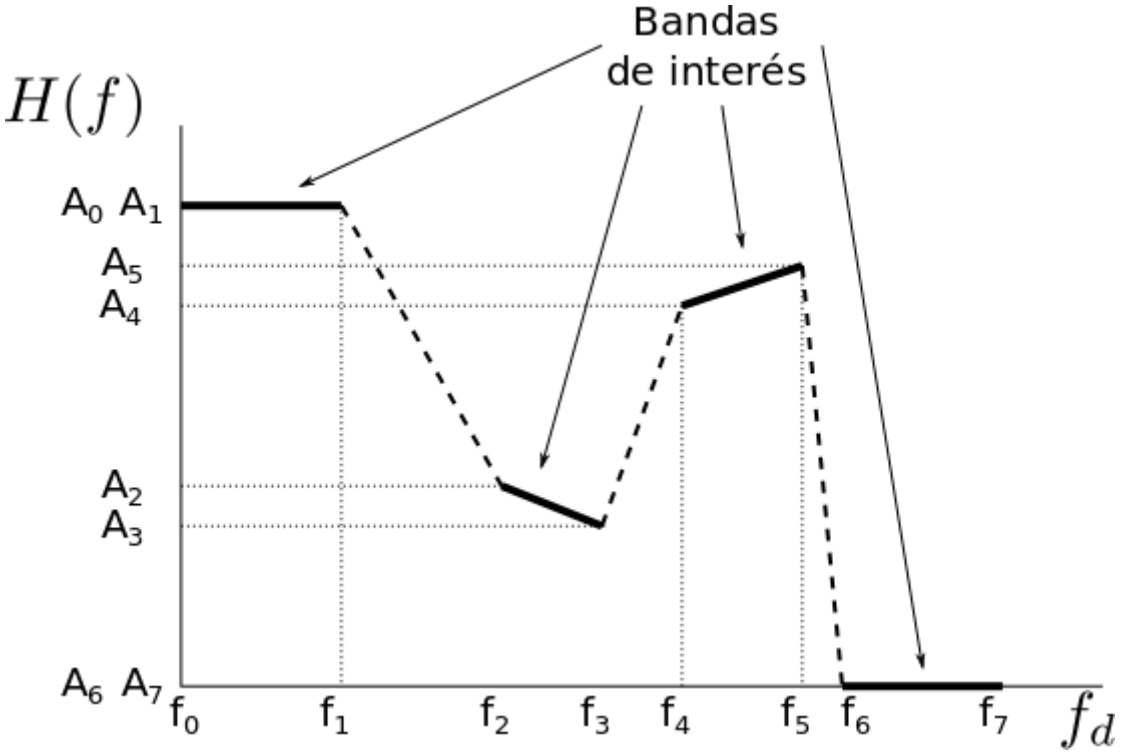
El método de la ventana es una de las tantas maneras existentes para aproximar la respuesta ideal de un filtro con una cantidad finita de coeficientes.
Existen otro métodos que buscan minimizar el error entre la respuesta ideal y la real, con una cantidad finita de coeficientes. Estos métodos son los considerados óptimos, en el sentido que solución es la mejor para la manera elegida de minimizar el error.
Los métodos que vamos a ver son dos
- Minimización del error promedio
- Minimización del error máximo

El primer método se conoce como algoritmo LMS [2] y el segundo como algoritmo de Remez [3] [4]

En nuestro caso vamos a utilizar las funciones disponibles en Matlab/Octave para el diseño de estos filtros sin entrar en detalles de los algoritmos.
Estas funciones se utilizan de igual manera, y permiten especificar filtros con formas

[2] https://en.wikipedia.org/wiki/Least_mean_squares_filter
[3] https://en.wikipedia.org/wiki/Remez_algorithm
[4] http://mathworld.wolfram.com/RemezAlgorithm.html

arbitrarias, como se observa en la siguiente figura



El filtro se define por un conjunto de pares de banda de interés. Cada banda de interés está definida por dos frecuencias digitales (de 0 a 1) y dos amplitudes. Los algoritmos intentarán converger la respuesta del filtro a la respuesta especificada usando la cantidad de taps deseada. Es importante remarcar que el filtro solo busca igualar la respuesta en las bandas de interés, mientras que las bandas de transición no se consideran en la iteración, lo que en algunos casos puede resultar en convergencias erróneas.

Las funciones a usar son **firls** y **remez**, de la siguiente manera

```
h = firls(N, F,  A);
```

y

```
h = remez(N, F, A); % En matlab se recomienda utilizar la función firpm
```

Donde N es la cantidad deseada de taps (coeficientes), F el vector de pares de frecuencias y A el vector de pares de amplitudes.
Para el ejemplo de la figura el vector F nos quedaría

```
F = [ f0, f1, f2, f3, f4, f5, f6, f7];
```

y el vector A

```
A = [ A0, A1, A2, A3, A4, A5, A6, A7 ];
```

6.1. Crear una señal compuesta por la suma de dos tonos senoidales de amplitud 0.5 y frecuencias f_1: 1000Hz y f_2: 1100Hz, con una frecuencia de muestreo de 22050Hz, y una duración de 2 segundos.
Graficar el espectro de la señal y reproducirla mediante la función **sound**.

6.2. Crear un filtro que permita atenuar 40dB la frecuencia f_2 con respecto a la f_1 con la menor cantidad de taps usando las funciones **firls** y **remez**. Determinar *N* por prueba y error hasta lograr una atenuación apenas mayor a 40dB con ambos métodos. Filtrar la señal mediante la función **conv** y reproducirla con la función **sound**. Graficar los espectros de ambos filtros, de la señal de entrada y de la señal de salida. Identificar cada gráfico con la función **legend**.

7. Filtros IIR. Implementación y respuesta en frecuencia

El filtro IIR, o filtro de respuesta infinita al impulso es un filtro realimentado, en el que la salida depende de las entradas actuales o pasadas además de las salidas pasadas.

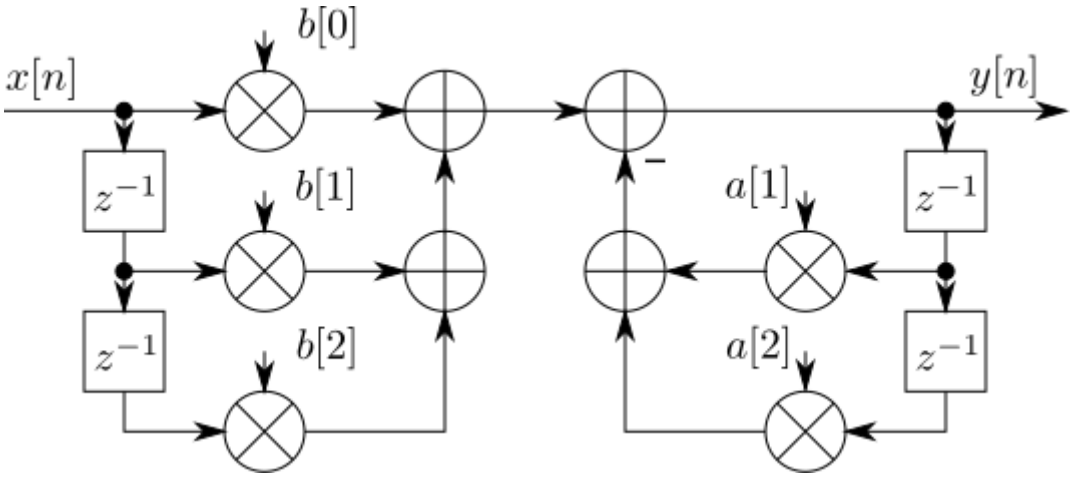
$$y[n] = \frac{1}{a[0]} \left(\sum_{i=0}^{P} b[i]x[n-i] - \sum_{i=1}^{Q} a[i]y[n-i] \right)$$

Esta ecuación en diferencias nos lleva a la siguiente función de transferencia en *z*

$$H[z] = \frac{b[0] + b[1]z^{-1} + \cdots + b[N]z^{-P}}{a[0] + a[1]z^{-1} + \cdots + a[N]z^{-Q}}$$

Cada salida se obtiene calculando dos sumatorias de productos. La primera es exactamente igual a la sumatoria de productos del filtro FIR. La segunda se obtiene de manera similar, pero operando sobre las muestras de salida. Luego ambas sumatorias de productos se restan entre sí y el resultado se escala al valor del coeficiente a[0].

Suponiendo P=2, Q=2 y a[0]=1 podemos visualizar la operación de la siguiente manera:



Vemos que la primera parte (de la mitad hacia la izquierda) es exactamente igual a la estructura del filtro FIR. Una diferencia en la ecuación es que en el filtro fir usamos el número de taps en la ecuación, mientras que en el filtro IIR usamos el orden del filtro, que es igual a la cantidad de elementos de retardo. Esta primera parte se conoce como sección directa del filtro.
La segunda parte (de la mitad hacia la derecha) es similar a la sección directa, con las siguientes diferencias:
- Usa las muestras de salida.
- No se usa la muestra de salida actual.
- Los productos se restan en la sumatoria.

Esta sección se conoce como sección realimentada.
En general, cuando se habla de un filtro IIR de orden N, se asume el mismo orden en la sección directa y en la sección realimentada.

La función de transferencia en este caso es

$$H[z] = \frac{b[0] + b[1]z^{-1} + \cdots + b[N]z^{-N}}{1 + a[1]z^{-1} + \cdots + a[N]z^{-N}}$$

7.1. Crear un programa que calcule la salida de un filtro IIR de orden N, tomando como referencia el ejercicio 2.1 de la guía 3, donde se debe agregar un segundo registro de desplazamiento para las muestras de salida que luego se multiplicará por los coeficientes A. Luego se sumarán las sumas de productos de ambas estructuras (Coeficientes a y b), teniendo en cuenta el signo en la sumatoria.

7.2. Asumiendo una frecuencia de muestreo de 256 Hz, crear una señal compuesta por la suma de dos tonos senoidales, de 50Hz y 100Hz, con una duración de 8 segundos. Filtrar la señal con el programa del punto anterior utilizando los siguientes coeficientes:
B = [0.0528556 0.0017905 0.0017905 0.0528556]

A = [1.00000 -2.12984 1.78256 -0.54343]
y determinar a partir del espectro de la señal de salida la amplitud de ambos tonos.

7.3. Filtrar la misma señal utilizando la función **filter** (ver ayuda). Graficar la salida sobre la salida del punto anterior y comprobar que ambas señales son iguales.

7.4. Utilizando:

```
[N,w]=freqz(B,A);
```

obtener la respuesta en frecuencia del filtro y graficar magnitud en dB y la frecuencia en Hz. Determinar valores de magnitud en 50Hz y 100Hz y exhibirlos sobre el gráfico usando la función **text**.

8. Respuesta al impulso de un filtro IIR y filtros en cascada

8.1. Utilizando la función **butter**, obtener los coeficientes de un filtro IIR pasa-altos, de orden 3, con frecuencia de corte en 50Hz y frecuencia de muestreo de 256Hz.

8.2. Generar una función impulso con longitud total de 4 segundos y obtener la respuesta del filtro mediante la función **filter**. Convertir la respuesta a frecuencia y graficar la respuesta del filtro.

8.3. Determinar la cantidad de etapas mínimas necesarias para obtener una atenuación mayor o igual a 120dB a 20Hz.

8.4. Tomar la respuesta al impulso del punto 8.2 y filtrarlo sucesivamente hasta completar la cantidad de etapas del punto 8.3. Graficar en cada iteración la respuesta en frecuencia.

8.5. Repetir los puntos 8.1 a 8.4 para filtros de chebyshev de tipo 1 y 2 y filtro elíptico, generados mediante las funciones **cheby1**, **cheby2** y **ellip**. El ripple en banda de paso máximo en todos los casos será de 1 dB y el ripple en banda de stop queda a consideración del alumno.