TRABAJO PRACTICO : FILTROS DIGITALES – MATLAB

Objetivo:

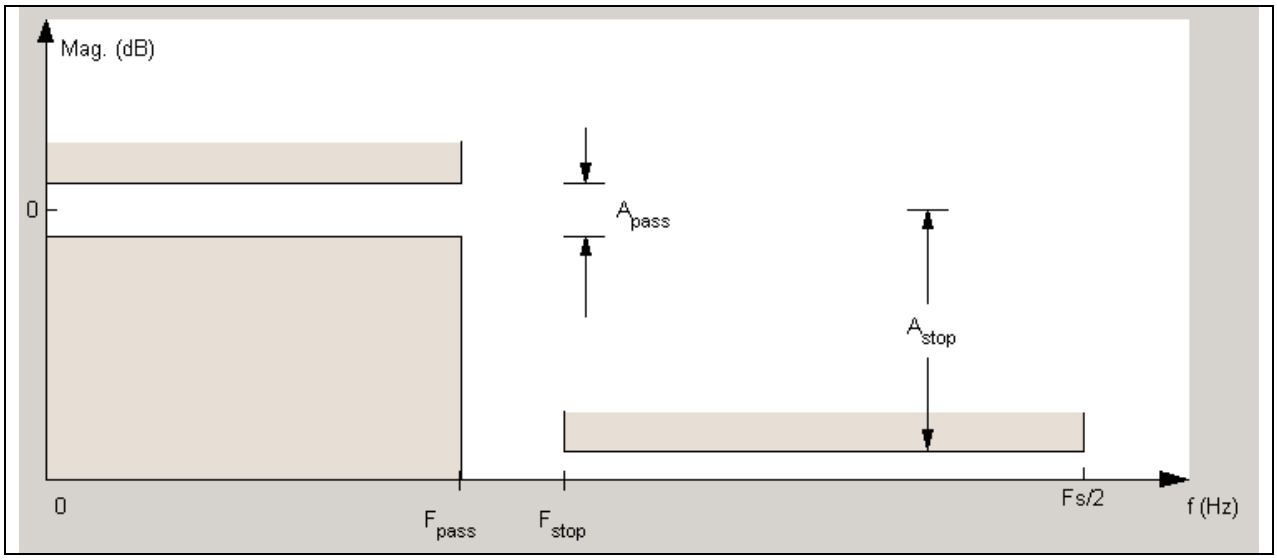
Se pretende conocer las herramientas que este software posee para el diseño de filtros tanto analógicos como digitales para afianzar los conocimientos vistos en clase.

Comparar entre distintos filtros digitales y a su vez con su par analógico.

Determinar los parámetros de un filtro y observar como podría ayudarnos a encontrar los coeficientes que luego podremos usar al programar un microprocesor.

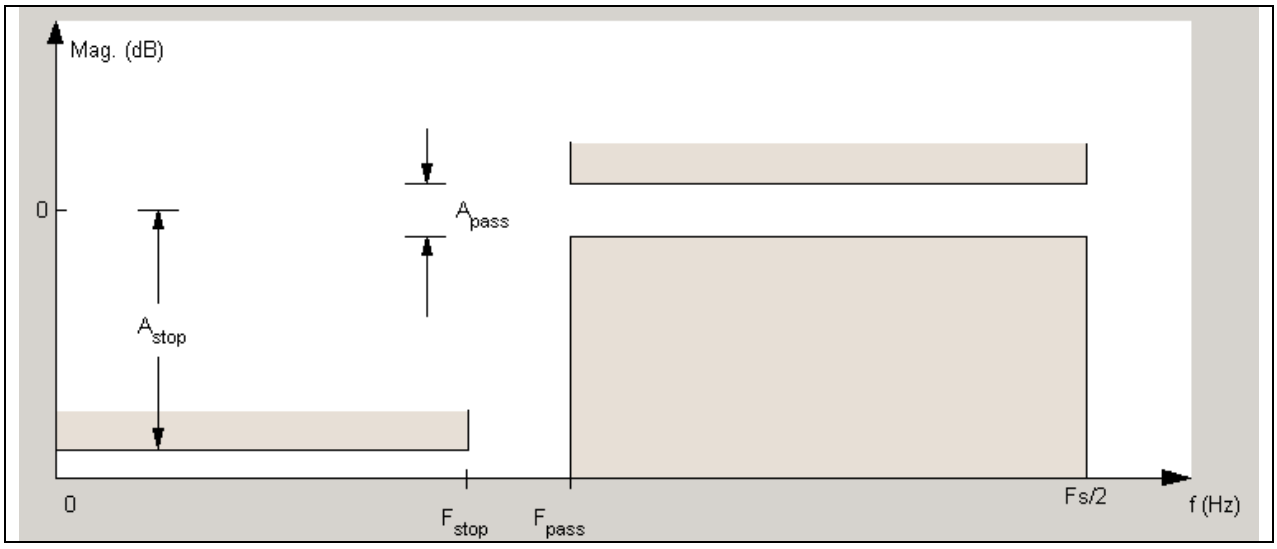
Breve Reseña a modo de introducción:

Filtro Pasa-Bajos:



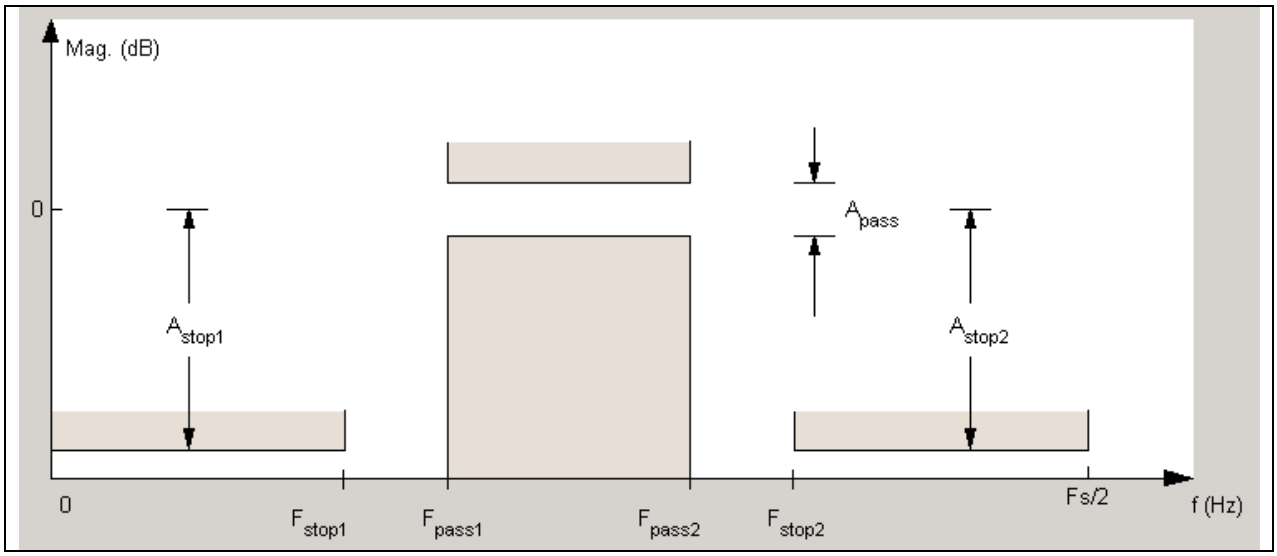
Donde Apass es el rizado de paso tolerable, Astop la atenuación en la banda de rechazo. Y en la banda de transición formada por la frecuencia de paso Fpass y la frecuencia de rechazo Fstop.

Filtro Pasa-Alto:

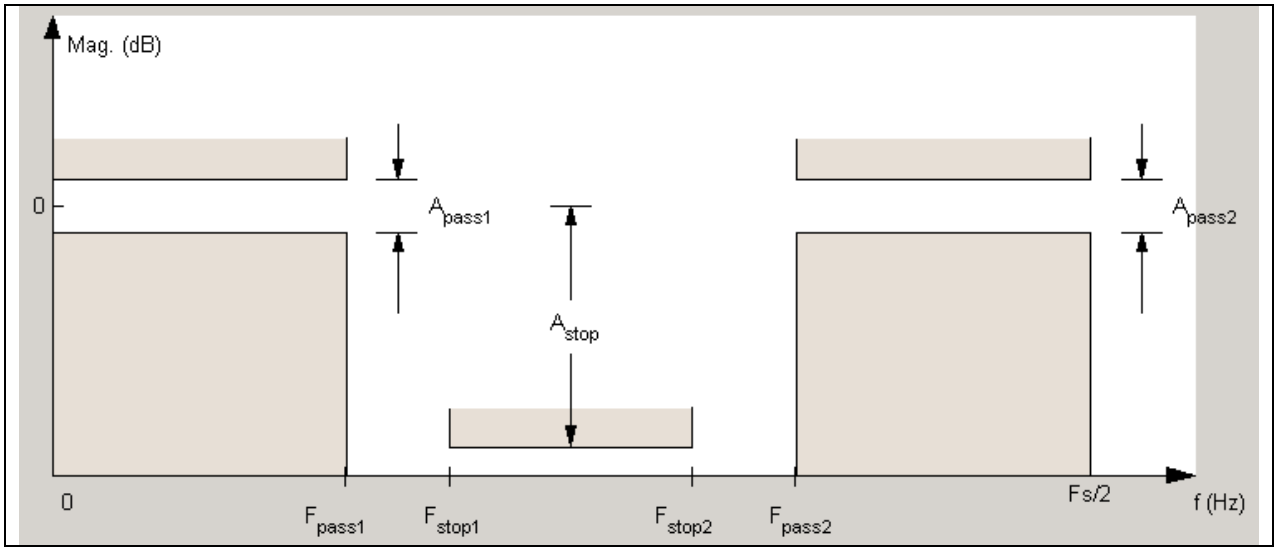


Aquí vemos a Fs/2 como la máxima frecuencia de paso debido al muestreo de la señal.

Filtro Pasa-Banda:



Filtro Rechaza-Banda:



Criterios de selección para filtros digitales

	FILTROS DIGITALES	
Tipos de filtros:	*FIR*	*IIR*
Ventajas:	• Fase lineal • Estable con coeficientes cuantificados	• Orden del filtro menor que el del filtro FIR equivalente

TABLA 1

Métodos de Diseño

	FILTROS DIGITALES	
Tipos de filtros:	*FIR*	*IIR*
Métodos de diseño:	• Aproximación directa de la respuesta en magnitud a través de la respuesta al impulso, mediante: o *Ventanas* o *Muestreo en frecuencia*	• Método basado en filtros analógicos mediante: o *Invarianza al impulso* o *Transformación bilineal*

TABLA 2

Nosotros en el práctico utilizaremos el método de las ventanas para el FIR y el de la transformación bilineal, embebidas en las funciones que detallamos a continuación.

Normalización de las frecuencias críticas:

Debido a que las funciones trabajan con frecuencia normalizada en radianes W en vez de hacerlo con F en Hz necesitaremos hacer un cambio en las variables y el modo de presentarlas en la fórmula.

Para normalizar éstas frecuencias fp y fs (frecuencias de paso y corte en Hz)

$$Wp = \frac{\Omega p}{Fs} = \frac{2.\pi.fp}{Fs} = 2.\pi.fp.Ts$$

$$Ws = \frac{\Omega s}{Fs} = \frac{2.\pi.fs}{Fs} = 2.\pi.fs.Ts$$

Donde Fs es la frecuencia de muestreo y Ts su recíproco, es decir el período de muestreo.

Ts=1/Fs

DISEÑO DE FILTROS FIR con MATLAB

Como ya hemos visto en clases anteriores este tipo de filtro está formado de solo ceros, y tienen las ventajas que se observan en la tabla 1. En la tabla 3 observamos un completo cuadro de ambos tipos de filtro.

Funciones:

Función FIR1

>> B = fir1(N,Wn,type,window);

Diseña un filtro FIR paso bajo de orden N (longitud N+1) y frecuencia de corte Wn (normalizada con respecto a la frecuencia de Nyquists, $0 \leq Wn \leq 1$). Se pueden especificar otro tipo de filtros de la misma forma que con los filtros IIR mediante el parámetro type. Por ejemplo, para un filtro parabanda:

>> B = fir1(N,[W1 W2],'stop');

Por defecto la función FIR usa la ventana de Hamming. Otro tipo de ventanas pueden también especificarse:

>> B = fir1(N,Wn,bartlett(N+1));

>> B = fir1(N,Wn,'high',chebwin(N+1,R));

De entre las ventanas que se pueden seleccionar tenemos: rectwin,kaiser,blackman o hamming entre otras.

Función FIR2

>> B = fir2(N,F,M,window);

Diseña un filtro FIR utilizando el método del muestreo frecuencial. Los parámetros de entrada es el orden del filtro N (longitud N+1) y dos vectores F y M que especifican la frecuencia y la magnitud, de forma que "plot(F,M)" es una gráfica de la respuesta deseada del filtro.

Se pueden indicar saltos bruscos en la respuesta frecuencial duplicando el valor de la frecuencia de corte. F debe estar entre 0 y 1, en orden creciente, siendo el primer elemento igual a 0 y el último 1. El parámetro window indica el tipo de ventana a utilizar. Por defecto, usa la ventana de Hamming.

>> B = fir2(N,F,M,'bartlett(N+1)');

Se pueden especificar más parámetros en esta función,

>> B = fir2(N,F,M,npt,lap,window);

La función fir2 interpola la respuesta frecuencial deseada (F,M) con npt puntos (por defecto, npt=512). Si dos valores sucesivos de F son iguales, se crea una región de lap puntos alrededor de este punto (por defecto, lap=25).

Función FIRLS

>> B = firls(N,F,M);

Diseño de filtros FIR usando la minimización del error por mínimos cuadrados. Los argumentos de entrada son el orden del filtro N, y dos vectores F y M, cuyo formato difiere de los análogos en la función fir2. El filtro obtenido es la mejor aproximación a (F,M) por mínimos cuadrados.

F es un vector que indica los límites de las bandas de frecuencia en parejas (por tanto el tamaño de F debe ser par), y en orden ascendente entre 0 y 1. M es un vector del mismo tamaño que F que indica la magnitud deseada para cada banda de frecuencias. La respuesta deseada es la línea que conecta los puntos (F(k),M(k)) y (F(k+1),M(k+1)) para k impar. Las bandas de frecuencia entre F(k+1) y F(k+2) para k impar son tratadas por firls como bandas de transición. También existe un argumento opcional que consiste en un vector W cuyo tamaño es la mitad de F. W es un factor de ponderación del error para cada banda de frecuencias.

>> B = firls(N,F,M,W);

Algoritmo de Parks-McClellan

Hay dos funciones en MATLAB para realizar este algoritmo: remezord y remez.

>> [N,Fo,Mo,W] = remezord(F,M,DEV,Fs)

Calcula el orden N, las bandas de frecuencia normalizadas Fo, las magnitudes en esas bandas Mo y los factores de ponderación W que luego serán utilizados como argumentos de entrada de la función remez. Estos valores cumplen las especificaciones dadas por F, M, DEV. F es un vector de frecuencias de corte en Hz, en orden ascendente entre 0 y Fs/2. Si no se especifica Fs, Fs=2 por defecto. El primer elemento de F es siempre 0 y el último es siempre Fs/2, pero no deben ser especificados en el vector F. El vector M indica la respuesta deseada en cada banda. Por tanto, el vector M tiene un tamaño igual a (length(F)+2)/2. DEV es un vector que indica el máximo rizado permitido en cada banda.

>> b = remez(N,Fo,Mo,W);

Con los valores obtenidos en la función remezord, podemos implemantar el algoritmo de Parks-McClellan. Fo y Mo son dos vectores de igual magnitud. Fo(k) y Fo(k+1) k impar especifica bandas de frecuencia y Mo(k) y Mo(k+1) la correspondiente magnitud para cada frecuencia. El filtro obtenido es la mejor aproximación por minimax.

Por lo tanto:
el orden de un filtro FIR puede determinarse usando la función de MATLAB remezord.
[ord, f, m, wgt] = remezord(fedge, mag, dev)
[ord, f, m, wgt] = remezord(fedge, mag, dev, FT)

donde:

fedge = vector de frecuencias de flancos de banda.
mag = vector de magnitudes en cada una de las bandas.
dev = vector que especifica las desviaciones máximas permitidas entre el filtro diseñado y la magnitud deseada.

La longitud de fedge es de dos veces la de mag menos 2; la de dev es la misma que la de mag. Los datos de salida son el valor estimado del orden del filtro ord, el vector normalizado de frecuencias de flancos de las bandas f, el vector de magnitud m y el vector de pesos wgt que satisface las especificaciones. Los datos de salida se pueden usar directamente en el diseño del filtro usando la función remez. La frecuencia de muestreo FT puede usarse en la segunda versión de esta función. Si no se especifica, el valor que se usa por defecto (default) es de 2 Hz. Una explicación mas detallada de estos parámetros puede obtenerse con el comado "help remezord" en la ventana de trabajo de MATLAB.

Ejemplo (ω)
Estimar la longitud de un filtro FIR con fase lineal con las siguientes especificaciones: flanco de la banda de paso = 180 Hz, flanco de la banda de rehazo = 200 Hz, rizo de la banda de paso $\delta p = 0.002$ y rizo de la banda de rechazo $\delta s = 0.001$, con razón de muestreo de 12 KHz.
Solución.
Como fp (= 180 Hz) < fs (= 200 Hz) se trata de un filtro pasa-bajas. Para determinar la longitud del filtro FIR con un programa en MATLAB, se puede usar la fórmula de Kaiser o la función remezord.

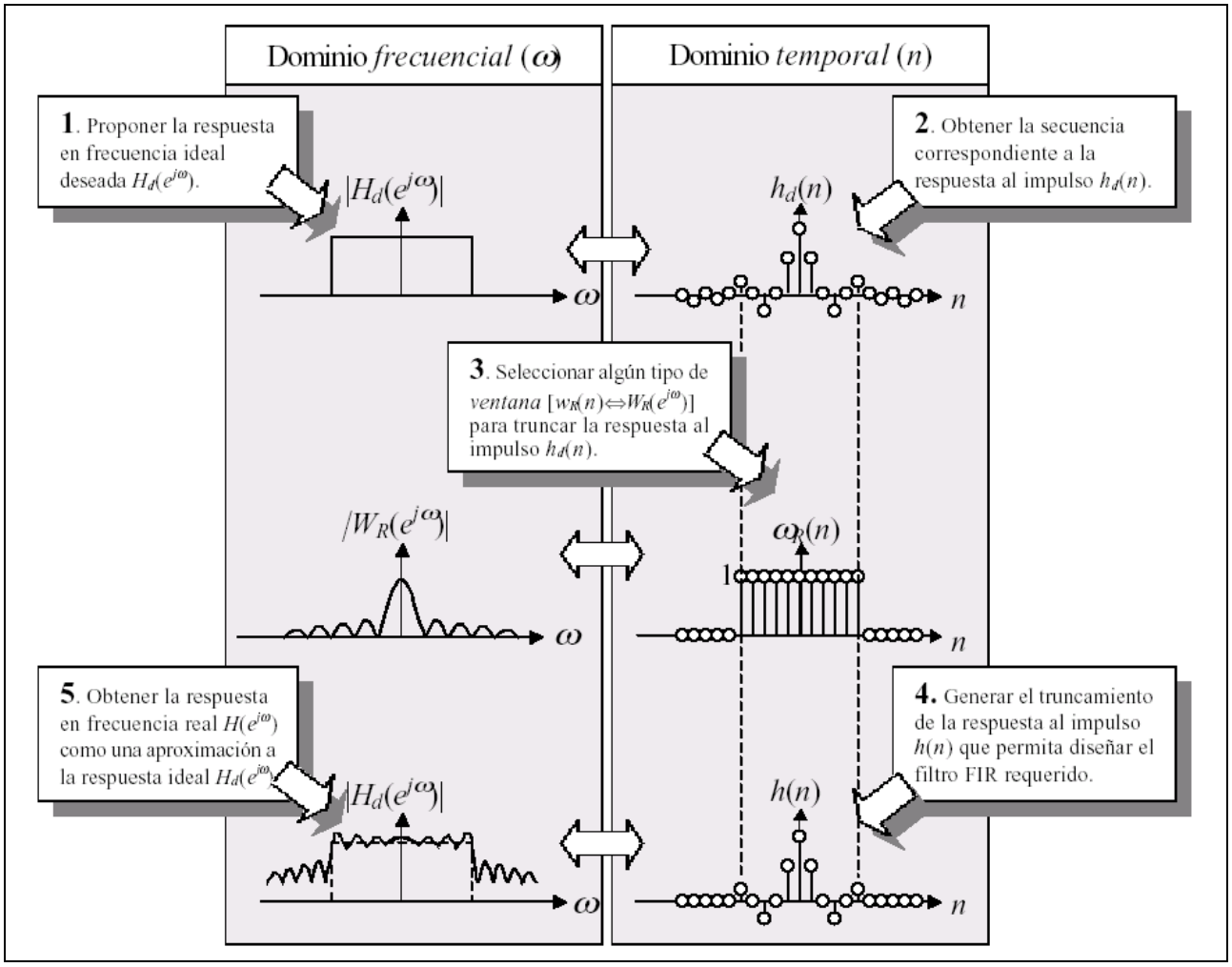
Con la función *remezord*:

[*ord, f, m, wgt*] = *remezord*(*fedge, mag, dev, FT*)

su codificación en MATLAB queda:
. . .
fedge = [180 200];
mag = [1 0];
dev = [0.002 0.001];
FT = 12000;
[*N, f, m, wgt*] = *remezord*(*fedge, mag, dev, FT*)
y el resultado es:
N = 1827
f = [0 0.030 0.0333 1.0000]
m = [1 1 0 0]
wgt = [1 2]

Método de ventanas para el diseño de filtros *FIR*

El método de diseño de filtros *FIR* mas sencillo es el *método de ventanas*. Este método se basa en obtener la respuesta en frecuencia del filtro $H(e^{jw})$ como una aproximación a la respuesta en frecuencia ideal deseada $H_d(e^{jw})$, mediante una determinada respuesta al impulso unitario $h(n)$ en el dominio temporal, a través de los pasos mostrados gráficamente en la Figura.



En la figura se ve el ejemplo gráfico para el diseño de un filtro pasa-bajas *FIR* mediante el *método de ventanas*.

MATLAB posee las funciones que ya nombramos para el diseño de este tipo de filtros. Estas funciones son: fir1 y fir2, y el formato en el que se usan éstas es:
b = fir1(L,Wn,window)
el cual genera un vector b de longitud L+1 que contiene los coeficientes del filtro FIR pasa-bajas, con frecuencia de corte Wn normalizada entre 0 y 1, y el parámetro window correspondiente a la ventana que se va a usar; por ejemplo window = kaiser, generaría los coeficientes del filtro FIR diseñado con la ventana Kaiser.

DISEÑO DE FILTROS IIR con MATLAB:

Funciones para determinar el orden necesario para implementar un determinado filtro :
[N, Wn] = buttord(Wp, Ws, Rp, Rs)
Calcula el orden de un filtro pasobajo digital de Butterworth, con Ws la frecuencia de pasabanda, Wp la parabanda, y Rp y Rs las Atenuaciones respectivas de pasabanda y parabanda en decibelios. Wp y Ws deben ser entre (0,1), siendo 1 la frecuenci
a de Nyquist (fs/2). N es el orden del filtro y Wn la frecuencia de 3db.

[N, Wn] = buttord(Wp, Ws, Rp, Rs,'s')
Lo mismo que antes, pero para un filtro pasobajo analógico. Aquí los valores de Wp y Ws pueden tomar cualquier valor en radianes. Para calcular el orden de otros tipos de filtro (pasoalto, parabanda o pasabanda) deberemos aplicar primero las transformaciones al prototipo de filtro pasobajo (Tabla 2).

[N, Wn] = cheb1ord(Wp, Ws, Rp, Rs)
Cálculo del orden necesario para un filtro digital pasobajo de Chebyshev I, con las especificaciones dadas. Las mismas consideraciones que en el caso del filtro de Butterworth.

[N, Wn] = cheb1ord(Wp, Ws, Rp, Rs, 's')
Lo mismo pero para el filtro analógico

[N, Wn] = cheb2ord(Wp, Ws, Rp, Rs)
Filtro digital de Chebyshev II

[N, Wn] = cheb2ord(Wp, Ws, Rp, Rs, 's')
Filtro analógico de Chebyshev II

[N, Wn] = ellipord(Wp, Ws, Rp, Rs)
Filtro digital elíptico

[N, Wn] = ellipord(Wp, Ws, Rp, Rs, 's')
Filtro analógico elíptico

Funciones para determinar los coeficientes del filtro:

[B,A] = butter(N,Wn)
B y A son los coeficientes del numerador y del denominador espectivamente, en orden decreciente de un filtro de Butterworth digital. N es el orden del filtro (calculado previamente) y Wn es la
frecuencia de corte. El valor de Wn debe estar normalizado con la frecuencia de Nyquist. Para diseñar un filtro pasobajo Wn es un escalar entre (0,1). La pasabanda es (0, Wn) y la parabanda es (Wn,1). Para diseñar un filtro de pasoalto, el comando a escribir es:

[B,A] = butter(N,Wn,'high')
donde Wn es un escalar.
Un filtro Parabanda se determina de la siguiente forma:

[B,A] = butter(N,[W1 W2])
Es decir, Wn es en este caso un vector que especifica las
frecuencias de pasabanda.

Finalmente, para un filtro Parabanda:

[B,A] = butter(N,[W1 W2],'stop')
[W1 W2] son las frecuencias de Parabanda.

[B,A] = cheby1(N,R,Wn)
Diseño de filtros digitales de Chebyshev I. Se deben especificar el orden del filtro N, el rizado de pasabanda permitido R y la frecuencia de corte normalizada con respecto a la frecuencia de Nyquist.
Para diseñar filtros de pasoalto, pasabanda y parabanda se siguen las mismas reglas que en el diseño de filtros de Butterworth.

[B,A] = cheby2(N,R,Wn)
Lo mismo que antes, pero R es el rizado de parabanda.

[B,A] = ellip(N,Rp,Rs,Wn)
Rp y Rs son los rizados de pasabanda y parabanda. Añadiendo a los comandos anterior la optión 's', los vectores B y A son los coeficientes del filtro analógico correspondiente. Sigue siendo válido lo que se mencionó anteriormente acerca del diseño de filtros
pasoalto, pasabanda y parabanda, pero Wn puede tomar cualquier valor en radianes (no está limitado entre (0,1)):
[B,A] = butter(N,Wn,'s')
[B,A] = cheby1(N,R,Wn,'s')
[B,A] = cheby2(N,R,Wn,'s')
[B,A] = ellip(N,Rp,Rs,Wn,'s')

Para Filtros pasabanda o parabanda se debe definir una variable con dos valores que determinan los extremos de la frecuencia de la banda
Ejemplo

Omega=[Omega1, Omega2]

Y usar el comando , por ejemplo

[num,den]=butter(n,Omega,´stop´) para el Parabanda
ó
[num,den]=butter(n,Omega) para el Pasabanda

Como obtener la respuesta frecuencial del filtro diseñado

Para los filtros analógicos
>> H = freqs(B,A,W)
Devuelve el vector H de números complejos, que es la respuesta frecuencial al filtro cuya función de transferencia en s viene dada por B y A. La respuesta frecuencial se evalúa en los puntos especificados por el vector W en radianes. Más opciones en el Help de MATLAB.

plot(W,abs(H))
Dibuja la magnitud de la respuesta frecuencial del filtro.

plot(W,unwrap(angle(H)))
Dibuja la fase de la respuesta frecuencial del filtro. La función unwrap hace que no haya discontinuidad en la fase por el paso de +pi a –pi .

Para los filtros digitales

>> H = freqz(B,A,F,Fs)

Devuelve el vector H de números complejos, que es la respuesta frecuencial al filtro cuya función de transferencia en z viene dada por B y A. La respuesta frecuencial se evalúa en los puntos especificados por el vector F en Hz, siendo la frecuencia de mustreo Fs Hz. Más opciones en el Help de MATLAB.

gd = grpdelay(B,A,F,Fs)

Calcula retraso de grupo (-dfi/dt) de la función de Transferencia
formada por los polinomios B y A. Se evalúa en los puntos especificados por W en radianes. Para más opciones de esta función ver el Help de MATLAB.

plot(F,abs(H))

Dibuja la magnitud de la respuesta frecuencial del filtro.

plot(F,unwrap(angle(H)))

Dibuja la fase de la respuesta frecuencial del filtro.

plot(F,gd)

Dibuja el retraso de grupo de la función de Transferencia Discreta.

<u>También podemos diseñar filtros digitales mediante métodos recursivos.</u>

Uno de estos métodos es el de Yule-Walker, que calcula los coeficientes del filtro de orden N utilizando mínimos cuadrados. Para ello debemos especificar la respuesta deseada para cada frecuencia. En MATLAB la función se denonima “yulewalk”:

[B,A]=yulewalk(N,F,M)
N es el orden del filtro, F y M son dos vectores de igual longitud. F es la frecuencia normalizada con respecto a la frecuencia de Nyquist (0-1). Debe estar en orden creciente y el primer y último elemento del vector deben ser 0 y 1 respectivamente. M es el vector que especifica la maginitud de la respuesta para cada elemento de F.

Ejemplo: Diseñar un filtro digital paso bajo de orden 5 con frecuencia de corte fc=1.3 KHz, por el método de Yule-Walker. Calcular las atenuaciones a 1 KHz y 2 KHz.
Primero hay que elegir una frecuencia de muestreo. Tomamos fs=5 KHz.

```
>>[B,A]=yulewalk(5,[0 1 1.3 2 2.5]/2.5,[1 1 0.708 0 0])
  B = [0.3155 0.8087 0.7811 0.4675 0.3284 0.1492]
  A = [1.0000 0.6490 0.5046 0.5031 0.1355 0.0577]
  >> H = 20*log10(abs(freqz(B,A,[1000 2000],5000)))
  ans = -0.2422 -20.9538 % En decibelios
  >> F=0:10:2500;Fs=5000;
  >> Hz = freqz(B,A,F,Fs);
  >> semilogy(F,abs(Hz));
```

Finalmente, cuando diseñemos un filtro digital nos interesará poder aplicar ese filtro a una señal temporal. Eso se consigue con la función de MATLAB “filter”.

```
  >> y = filter(B,A,x)
```

El vector x es la entrada y el vector y es la salida filtrada. B y A son los coeficientes del filtro digital. Existe otra función llamada "filtfilt", que funciona de la misma manera que "filter", pero hace dos filtrados. Primero filtra el vector x, y su respuesta la rota y le vuelve a aplicar el mismo filtro. La respuesta final evita la distorsión de fase
propia de los filtros IIR. Más detalles en el Help de MATLAB.

```
>> y = filtfilt(B,A,x)
```

Tabla auxiliar para el diseño de filtros digitales

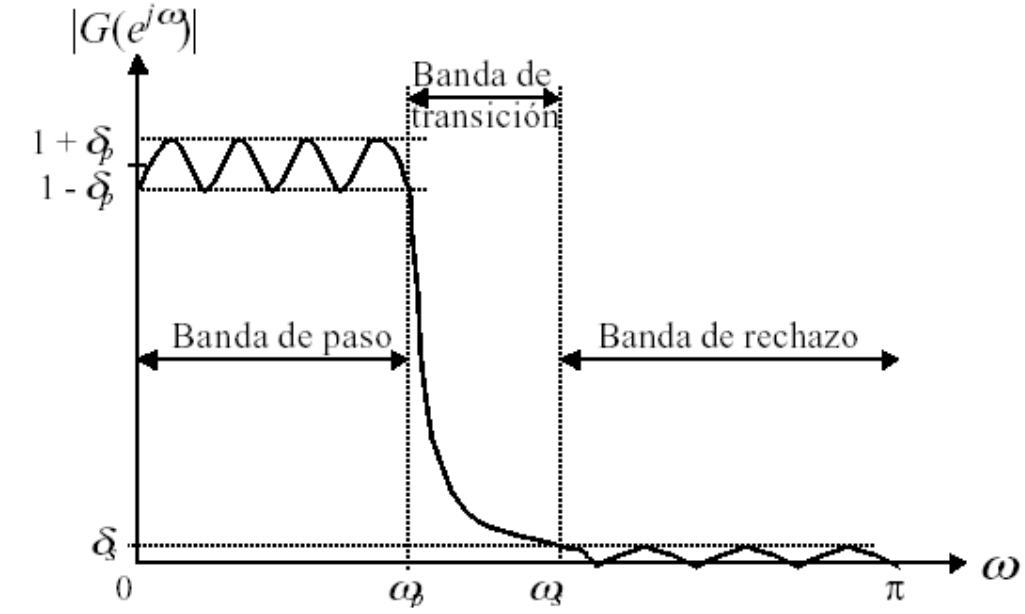
	FILTROS DIGITALES	
Tipos de filtros:	*FIR*	*IIR*
Ventajas:	• Fase lineal • Estable con coeficientes cuantificados	• Orden del filtro menor que el del filtro FIR equivalente
Métodos de diseño:	• Aproximación directa de la respuesta en magnitud a través de la respuesta al impulso, mediante: o *Ventanas* o *Muestreo en frecuencia*	• Método basado en filtros analógicos mediante: o *Invarianza al impulso* o *Transformación bilineal*
Estimación del orden del filtro:	• Fórmula de *Kaiser* • MATLAB: *[ord,f,m,wgt]=remezord(fedge,mag,dev)*	• Basado en el orden del *filtro analógico prototipo* • MATLAB: *[N,Wn]=filtord(Wp,Ws,Rp,Rs)*
Clases de filtros	*Pasa-bajas, pasa-altas, pasa-banda, rechazo de banda, peine (comb filter), fase lienal*	*Pasa-bajas, pasa-altas, pasa-banda, rechazo de banda, peine(comb filter), pasa-todo*
Tipo de respuesta (en frecuencia y en tiempo) de filtros:	Ventanas fijas: o *Rectangular* o *Hamming* o *Hanning* o *Blackman* Ventanas ajustables: o *Kaiser*	Polinomios de: o *Butterworth* o *Chebyshev I* o *Chebyshev II* o *Elíptico*
Tipos de estructuras básicas para la realización de filtros digitales:	• *Forma Directa* • *Cascada*	• *Forma Directa* • *Cascada*

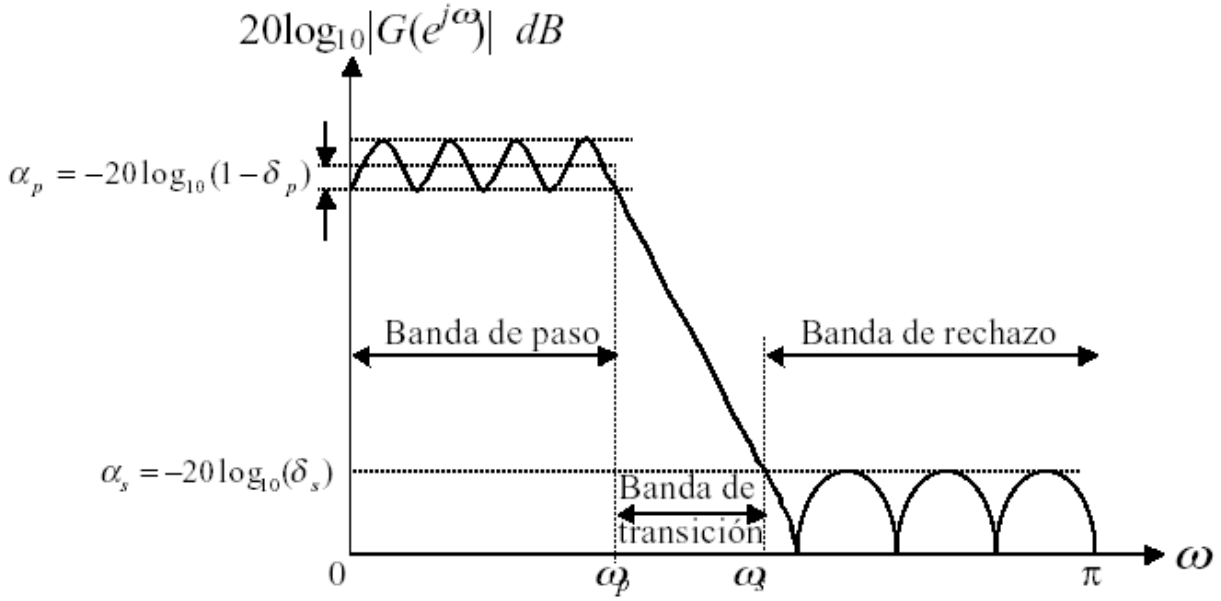
TABLA 3

Diseño 1

Ejemplo de Diseño de un Filtro de paso bajo usando las fórmulas anteriores:

En el siguiente ejemplo vemos la respuesta típica de un filtro pasa bajos que diseñaremos a continuación:





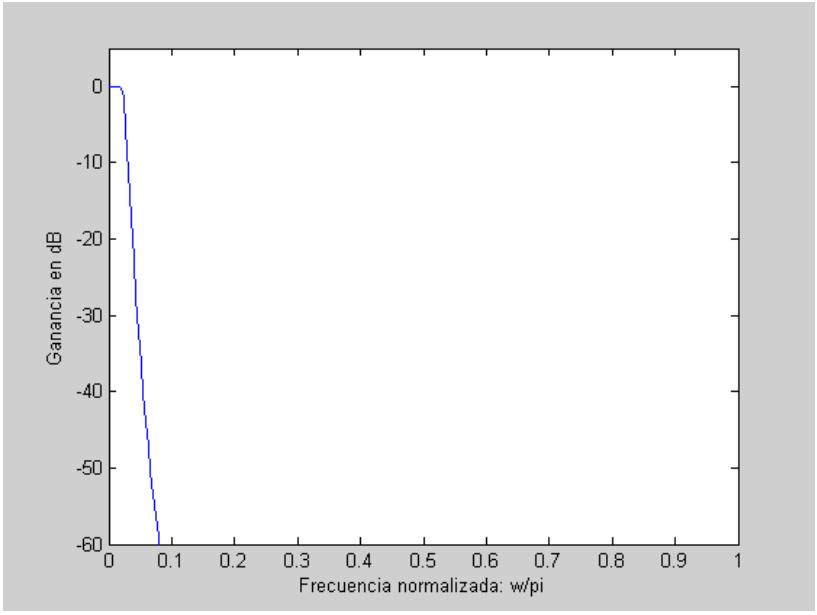
Se pretende diseñar un Filtro de paso bajo para extraer ruido blanco que ha contaminado una señal de 1 Vpp de 1 Hz.

Para comenzar el diseño comenzaremos con el filtro de Butterwhort,, para ello probaremos con una frecuencia de corte de 5 Hz. Las frecuencias normalizadas las calcularemos fácilmente teniendo en cuenta que Fs/2 sería nuestro valor de 1. Las demás frecuencias en Hz se pueden convertir usando una simple regla.

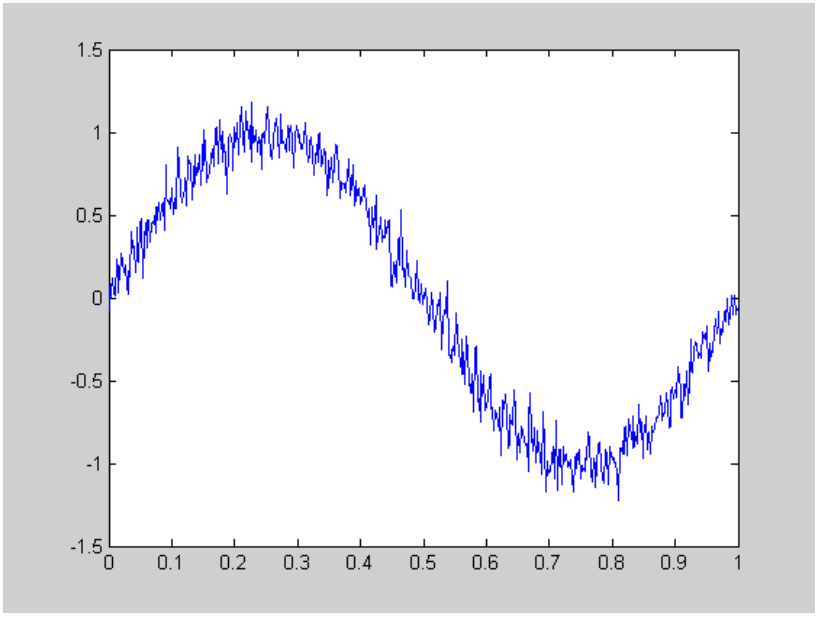
Nos quedaría Wc=0.01 , Ws=0.08 (supongo 40 Hz) , 0.5 de atenuación en la banda de paso en db y 60 db en la banda de rechazo.
Utilizaremos una frecuencia de muestreo de 1000 Hz.

```
[N,Wn]=buttord(0.01, 0.08, 0.5, 60); %Nos dá el orden y frec. De corte del filtro
[num,den]=butter(N,Wn);              %Calcula los coeficientes del numerador y denominador del filtro.
w=0:pi/255:pi;                       %Hacemos variar la frecuencia entre 0 y pi. Barrido
figure(1)
Hlp=freqz(num,den,w);                %Calcula la respuesta en frecuencia del filtro para ls Fs elegida.
%plot(F,abs(Hlp))
%plot(F,unwrap(angle(Hlp)))
%plot(F,unwrap(angle(Hlp)))
semilogy(w/pi,abs(Hlp))              %Escala logaritmica de amplitud
grid
H = 20*log10(abs(Hlp));
figure(2)
plot(w/pi,H)
axis([0 1 -60 5]);
ylabel('Ganancia en dB');
xlabel('Frecuencia normalizada: w/pi');
t=0:1/1024:1-1/1024;                 %Creo una señal de muestra
x=sin(2*pi*1*t);                     %Asigno esos valores calculados al vector x
figure(3)
plot(t,x)                            %Muestro Señal original
y=x+0.1*randn(size(t));              %Sumo ruido gaussiano de varianza 0.1
figure(4)
plot(t,y)                            %Muestro la señal con ruido
figure(5)
%stem(t,y)
Sal=filter(num,den,y)                %Aplico el filtro diseñado a la señal de prueba.
plot(t,Sal)                          %Muestro la señal Filtrada
```

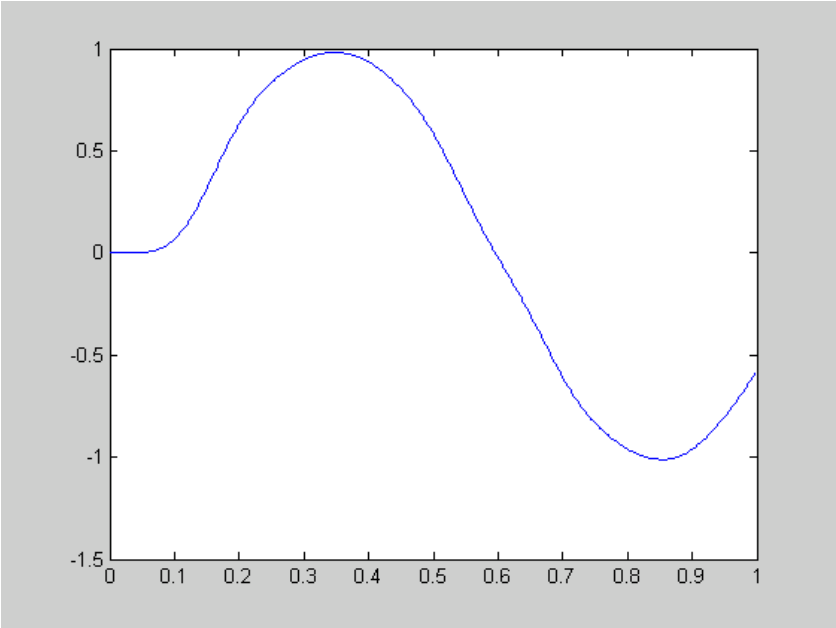
Función Transferencia del Filtro (Magnitud):



Señal Contaminada con ruido blanco:



Señal Filtrada:



Diseño 2:

Obtener la función de transferencia y graficar la respuesta en frecuencia de un filtro digital *IIR* pasa-bajas elíptico de 5º orden con frecuencia de paso de 0.4, y rizo de paso de 0.5 *dB*, y atenuación mínima en la banda de rechazo de 40 *dB*.

```
clear;
w=0:pi/255:pi; %Barrido de frecuencia
N = 5;
Rp = 0.5;
Rs = 40;
Wn = 0.4;
[b,a] = ellip(N,Rp,Rs,Wn);
disp('Polinomio del numerador:'); disp(b);
disp('Polinomio del denominador:'); disp(a);
% Obtención de la respuesta en frecuencia
h = freqz(b,a,w);
H = 20*log10(abs(h));
ph = angle(h)*180/pi;
% Graficación de la respuesta en frecuencia
figure(1);
subplot(211);
plot(w/pi,H);
 axis([0 1 -60 5]);
ylabel('Ganancia en dB');
xlabel('Frecuencia normalizada: w/pi');
grid;
subplot(212);
plot(w/pi,ph);
ylabel('Fase en grados');
xlabel('Frecuencia normalizada: w/pi');
```

```
grid;
% Detalle de la respuesta en frecuencia **********
figure(2);
plot(w/pi,H);
axis([0 0.5 -3.5 0.5]);
ylabel('Ganancia en dB');
xlabel('Frecuencia normalizada: w/pi');
grid
```