PROCESAMIENTO DIGITAL DE SEÑALES

MÓDULO 2. FILTROS DIGITALES

El filtro digital es la implementación en hardware o software de una ecuación en diferencias con una entrada digital.

$$y(k) = a(2)y(k-1) + \cdots + a(m+1)y(k-m) = b(1)x(k) + \cdots + b(n+1)x(k-n)$$

Coeficientes del filtro: a(1)=1, a(2),…., b(1), b(2),……..

$y(1) = b(1)x(1)$

$y(2) = b(1)x(2) + b(2)x(1) - a(2)y(1)$

$y(3) = b(1)x(3) + b(2)x(2) + b(3)x(1) - a(2)y(2) - a(3)y(1)$

……..

$y(k) = b(1)x(k) + \cdots + b(n+1)x(k-n) - a(2)y(k-1) - \cdots - a(m+1)y(k-m)$

La función de transferencia del filtro digital es aplicando Transformada Z:

$$Y(z) = H(z)X(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \cdots + a(m+1)z^{-m}} X(z)$$

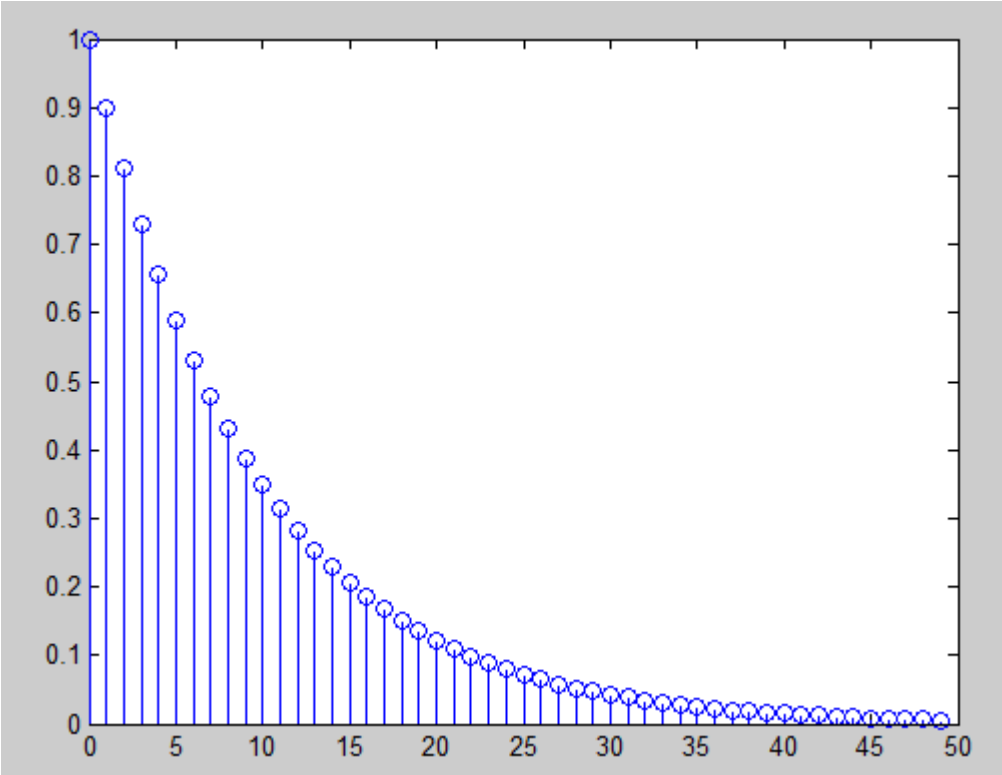
Respuesta al impulso (convolución) del filtro:

$$y(k) = h(k) * x(k) = \sum_{n=0}^{\infty} x(k-n)h(n)$$

Ejemplo:

Respuesta al impulso de un filtro con coeficientes a(1)=1, a(2)=-0.9, b(1)=1 con MatLab.

```
n=0:49;
%señal impulso
imp = [1; zeros(49,1)];
%coeficientes del filtro
b=1; a=[1 -0.9];
%respuesta al impulso
h = filter(b,a,imp);
stem(n,h)
```

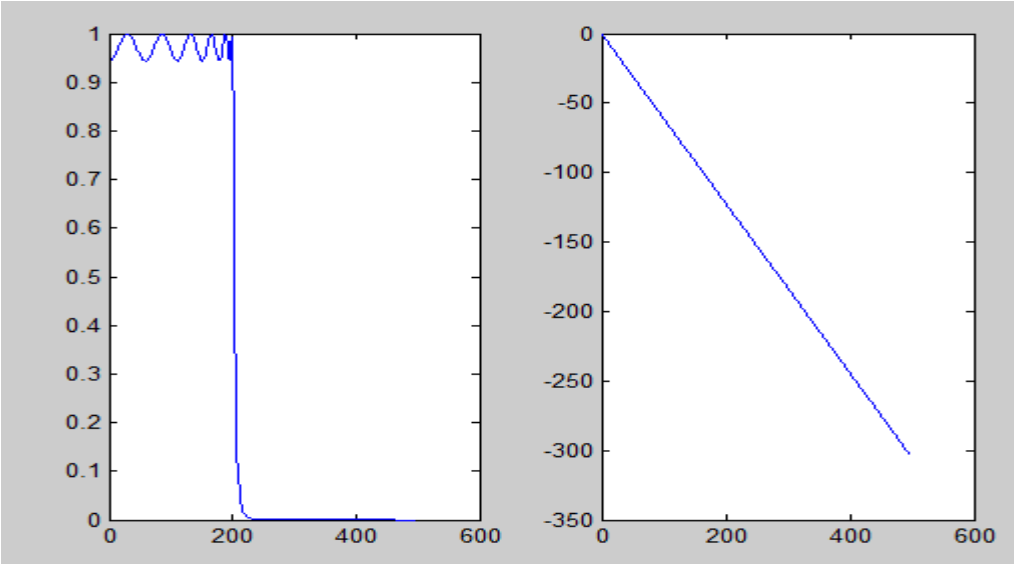


Respuesta en frecuencia del filtro:

$$H(e^{jw}) = \frac{b(1)+b(2)e^{-jw}+\ldots+b(n+1)e^{-jw(n)}}{a(1)+a(2)e^{-jw}+\ldots+a(m+1)e^{-jw(m)}}$$

Ejemplo:

```
% coeficientes del filtro
[b,a] = cheby1(12,0.5,200/500);
%respuesta en frecuencia
[h,f] = freqz(b,a,256,1000);
%grafica de la magnitud
mag=abs(h);
subplot(121)
plot(f,m)
%grafica de la fase
fase=unwrap(f*180/pi);
subplot(122)
plot(f,fase)
```



Los filtros digitales tienen:

- Alta inmunidad al ruido
- Alta precisión, limitada por los errores de redondeo en la aritmética empleada
- Fácil modificación de las características del filtro
- Muy bajo costo

Los filtros se clasifican en filtros FIR (Respuesta impulsional finita) y filtros IIR (Respuesta impulsional Infinita)

FILTROS FIR

Un filtro FIR de orden M tiene la siguiente función de transferencia:

$$y(n) = b(1)x(n) + b(2)x(n-1) + \cdots + b(m+1)x(n-M)$$

Tiene como función de transferencia:

$$H(z) = b(1) + b(2)z^{-1} + \cdots + b(m+1)z^{-M}$$

- La secuencia b(k) son los coeficientes del filtro
- Es no recursivo, o sea, la salida depende solamente de las entradas y no de las salidas pasadas
- La función de transferencia sólo tiene ceros, excelente estabilidad.

FILTROS IIR

Tiene como ecuación en diferencias:

$$y(n) + a(2)y(n-1) + \cdots + a(n+1) + y(n-N) = b(1)x(n) + b(2)x(n-1) + \cdots + b(m+1)x(n-M)$$

Tiene como función de transferencia:

$$H(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(m+1)z^{-M}}{1 + a(2)z^{-1} + \cdots + a(n+1)z^{-N}}$$

- Es recursivo, o sea, que su salida además de las entradas depende de las salidas pasadas.
- Tiene polos y ceros, tiene problemas de estabilidad.
- La fase no es lineal con la frecuencia
- El orden del filtro es mucho menor que un filtro FIR para la misma aplicación

DISEÑO DE FILTROS DIGITALES

El diseño consiste en obtener los *coeficientes* del filtro para conseguir unos requerimientos específicos. Su implementación obedece en escoger y aplicar a una *estructura* particular del filtro esos coeficientes.

Los filtros se normalizan a la frecuencia de Nyquist, o sea, a la frecuencia de muestreo dividida por dos:

$$f_N = \frac{f_s}{2}$$

Por ejemplo, para filtrar 30 Hz con un filtro pasabajas y f_s =*100 Hz* con un Butterworth de orden 5:

[b,a] = butter(5,30/50) = butter(5,0.6)

Para convertir la frecuencia normalizada a frecuencia angular se debe multiplicar por π.
Una especificación más rigurosa podría ser riple en la banda de paso (passband-R_s), atenuación en la banda de rechazo (stopband-R_p) o en la banda de transición (w_s-w_p), etc.

1. DISEÑO DE FILTROS IIR

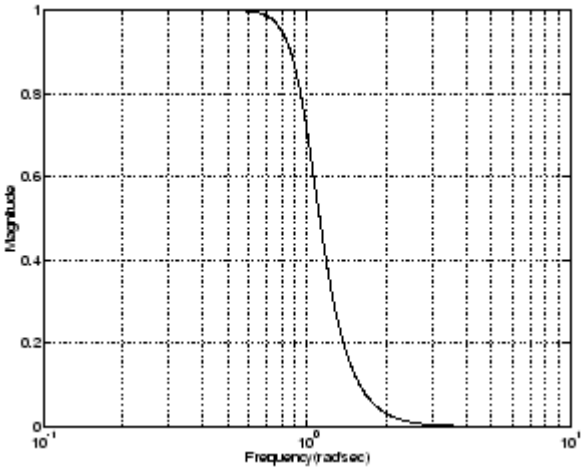
Hay varios métodos:

a) IIR USANDO FILTROS ANÁLOGOS

Filtro Butterworth

Comprende diseños de filtros pasabajo, pasa banda, pasa alto, y banda rechazo. La respuesta en magnitud es plana en la banda de paso. Filtro pasabajo de orden n con frecuencia de corte w_n

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \cdots + a(n+1)z^{-n}}$$



[b,a] = butter(n,w_n)
[b,a] = butter(n,w_n, 'ftype')
[z,p,k] = butter(n,w_n)
[z,p,k] = butter(n,w_n, 'ftype')

Tipo= high, low, bandpass, bandstop (w_n=[w_1 w_2])

La frecuencia de corte es donde la magnitud del filtro es $|H(j\Omega)| = \frac{\sqrt{2}}{2} = 0.707$ en Ω=1

Para encontrar el orden y la frecuencia de corte dadas las especificaciones:

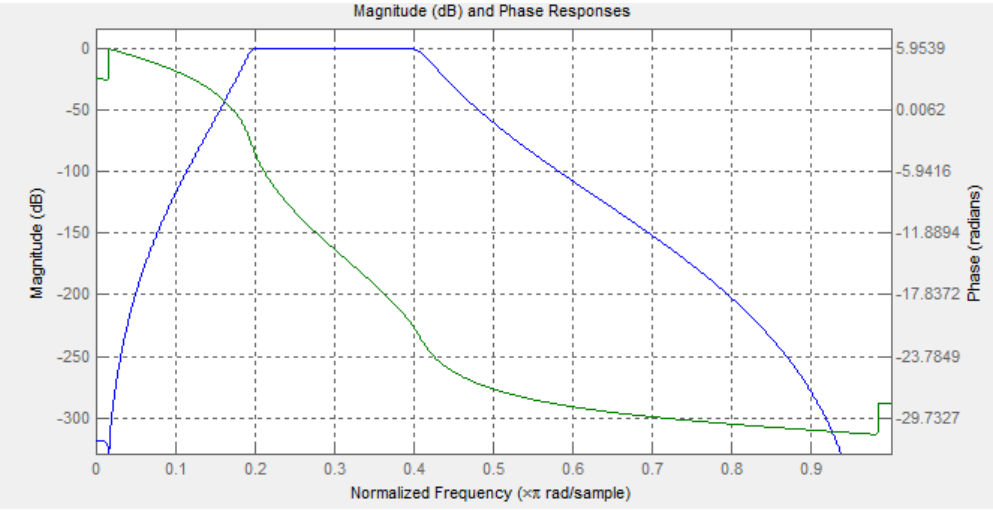
[n,Wn] = buttord(Wp,Ws,Rp,Rs)

Filter Type	Order Estimation Function
Butterworth	`[n,Wn] = buttord(Wp,Ws,Rp,Rs)`
Chebyshev Type I	`[n,Wn] = cheb1ord(Wp, Ws, Rp, Rs)`
Chebyshev Type II	`[n,Wn] = cheb2ord(Wp, Ws, Rp, Rs)`
Elliptic	`[n,Wn] = ellipord(Wp, Ws, Rp, Rs)`

Ejemplo: Pasa banda

Se quiere un filtro pasabanda de 100 a 2000 Hz, la banda stop arranca en 500 Hz, la frecuencia de muestreo es de 10 KHz, al menos 1 dB de riple en la banda de paso y al menos 60 dB de atenuación en la banda stop.

```
[n,Wn] = buttord([1000 2000]/5000,[500 2500]/5000,1,60)
%n = 12, Wn =    0.1951    0.4080
[b,a] = butter(n,Wn);
[sos,g] = tf2sos(b,a);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd);
set(h,'Analysis','freq')
```

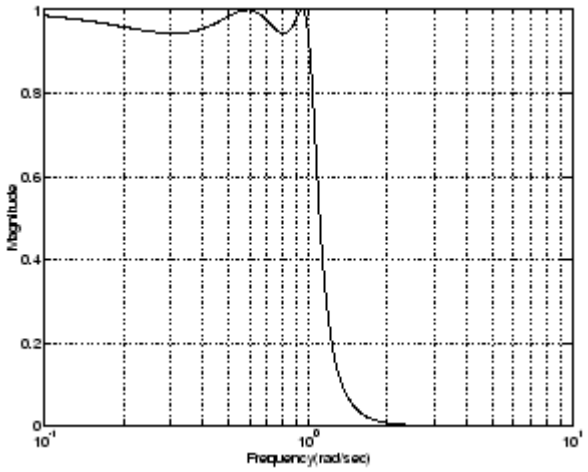


Filtro Chebyshev Tipo I

Minimiza la diferencia entre el ideal y la respuesta de frecuencia actual sobre la banda de paso incorporando un equiriple de R_p dB en la banda de paso. La respuesta en la banda rechazo es plana (maximally flat). La transición de la banda de paso a la banda de rechazo es más rápida que en el de Butterworth.

.

$$|H(j\Omega)| = 10^{-\frac{R_p}{20}} \quad en \quad \Omega = 1$$



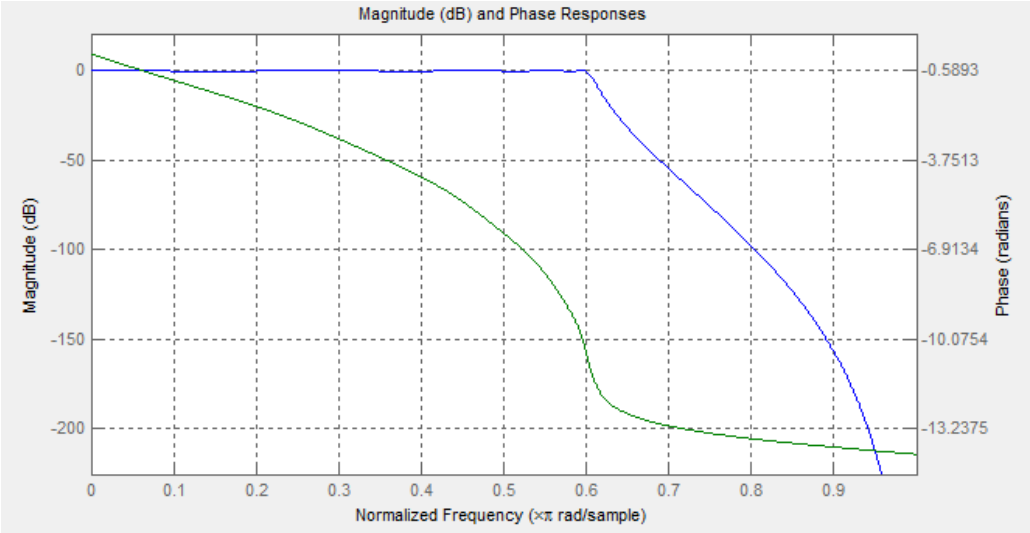
[z,p,k] = cheby1(n,R,Wp) : Filtro pasa bajo
z,p,k] = cheby1(n,R,Wp,'ftype')
[b,a] = cheby1(n,R,Wp) : Filtro pasa bajo
[b,a] = cheby1(n,R,Wp,'ftype')

El orden del filtro es n con frecuencia de corte en la banda de paso normalizada en W_p y R dB de riple pico a pico en la banda de paso.

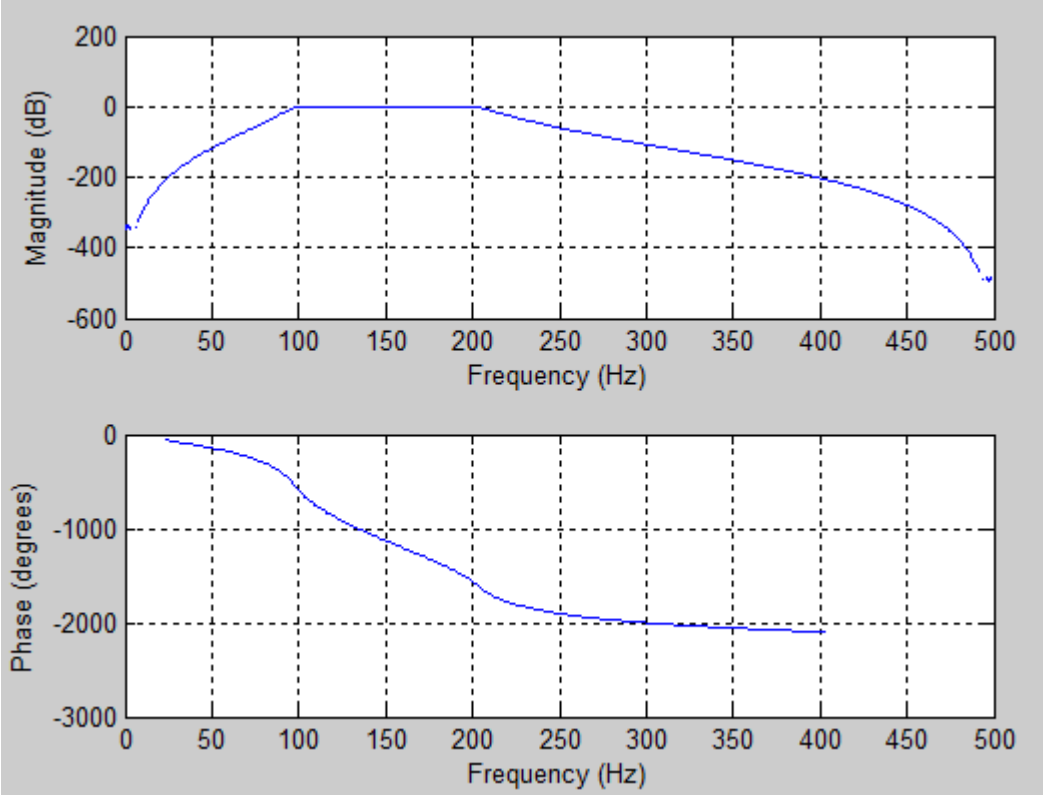
Ejemplo: Pasa bajo

Para una frecuencia de muestreo de 1000 Hz diseñar un filtro Chebyshev Tipo I con 0.5 dB de riple en la banda de paso y una frecuencia de borde en la banda de paso de 300 Hz.

```
[z,p,k] = cheby1(9,0.5,300/500);
[sos,g] = zp2sos(z,p,k);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd)
set(h,'Analysis','freq')
```



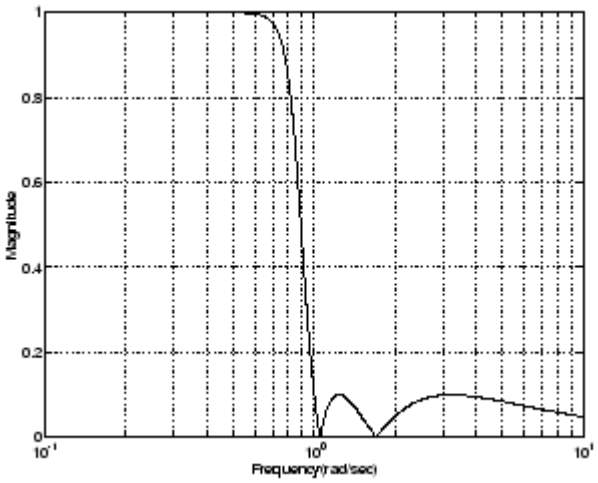

```
%Respuesta en frecuencia
freqz(b,a,512,1000)
```



Filtro Chebyshev Tipo II

Minimiza la diferencia con el filtro ideal en la banda stop incorporando un equiriple de R_s dB en la banda stop. La respuesta en la banda de paso es plana (Maximally flat).

$|H(j\Omega)| = 10^{-\frac{R_S}{20}} \quad en \quad \Omega = 1$

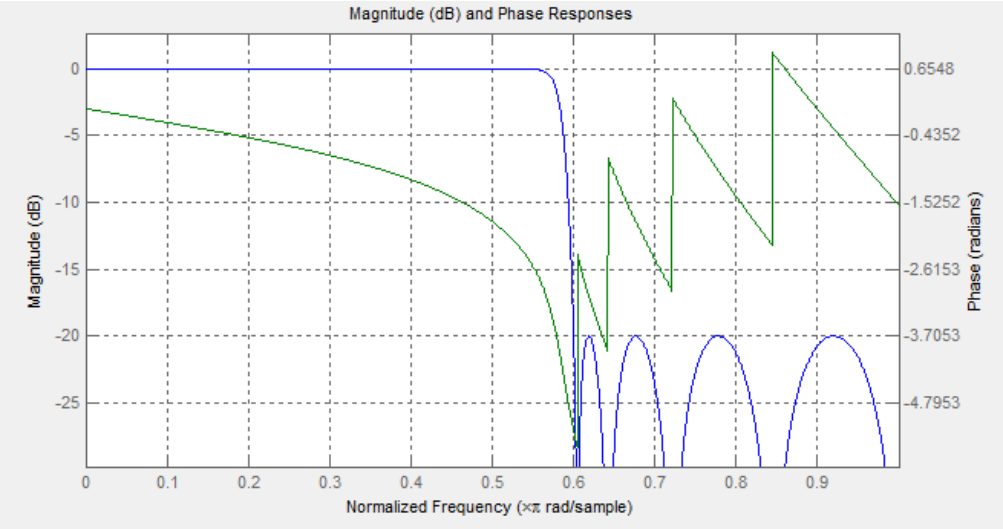


[z,p,k] = cheby2(n,R,Wst)
[z,p,k] = cheby2(n,R,Wst,'ftype')
[b,a] = cheby2(n,R,Wst)
[b,a] = cheby2(n,R,Wst,'ftype')

Ejemplo: Pasa bajo

Para una frecuencia de muestreo de 1000 Hz diseñe un filtro pasa bajo Chebyshev II con atenuación en la banda stop de 20 dB debajo de la banda de paso y una frecuencia de borde en banda stop de 300 Hz.

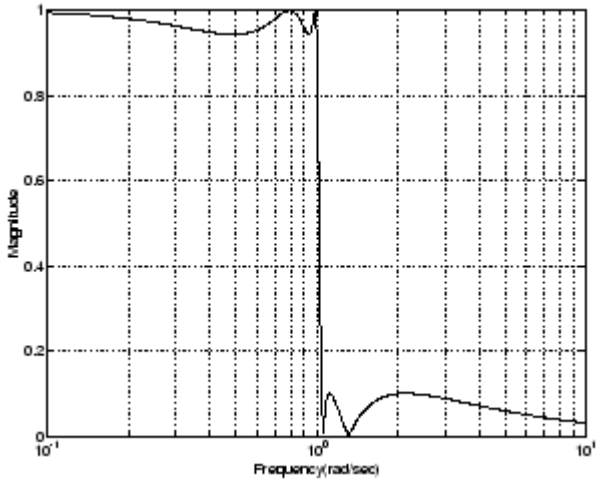
```
[z,p,k] = cheby2(9,20,300/500);
[sos,g] = zp2sos(z,p,k);              % Convert to SOS form
Hd = dfilt.df2tsos(sos,g);       % Create a dfilt object
h = fvtool(Hd);                       % Plot magnitude response
set(h,'Analysis','freq')              % Display frequency response
```



Filtro Elíptico

Es un filtro equiriple tanto en la banda de paso como en la banda de rechazo. Riple en la banda de paso R_p, riple en la banda stop R_s. Minimiza el ancho de la transición.

$$|H(j\Omega)| = 10^{-\frac{R_p}{20}} \quad en \quad \Omega = 1$$



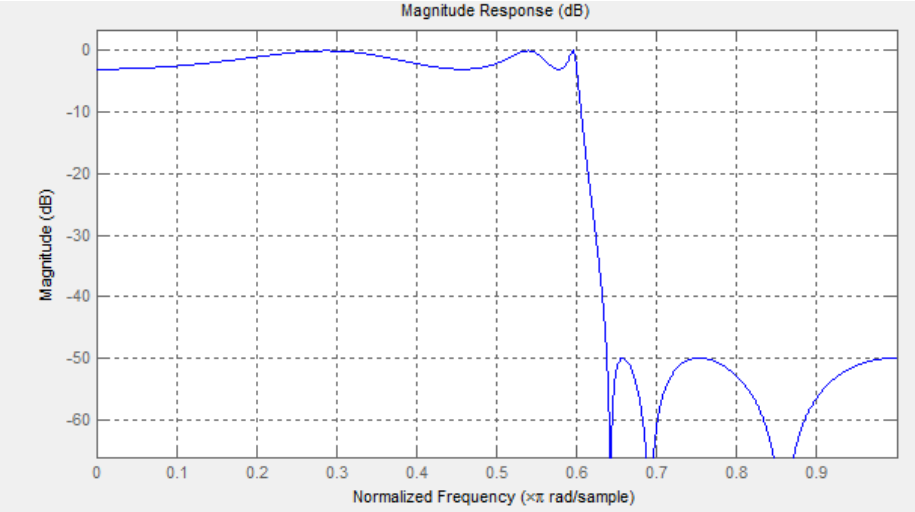
[z,p,k] = ellip(n,Rp,Rs,Wp)
[z,p,k] = ellip(n,Rp,Rs,Wp,'ftype')
[b,a] = ellip(n,Rp,Rs,Wp)
[b,a] = ellip(n,Rp,Rs,Wp,'ftype')

W_p frecuencia normalizada en banda de paso, R_p riple en dB en la banda de paso, R_s riple en dB en la banda rechazo.

Ejemplo: Pasa bajo

Diseñar un filtro pasa bajo Elíptico de orden 6 con f_p=300 Hz, 3 dB en la banda de paso y 50 dB de atenuación en la banda rechazo.

```
[z,p,k] = ellip(6,3,50,300/500);
[sos,g] = zp2sos(z,p,k);           % Convert to SOS form
Hd = dfilt.df2tsos(sos,g);     % Create a dfilt object
h = fvtool(Hd)                 % Plot magnitude response
set(h,'Analysis')
```



RESUMEN

.

Butterworth	`[b,a] = butter(n,Wn,options)`
	`[z,p,k] = butter(n,Wn,options)`
	`[A,B,C,D] = butter(n,Wn,options)`

Chebyshev Type I	`[b,a] = cheby1(n,Rp,Wn,options)`
	`[z,p,k] = cheby1(n,Rp,Wn,options)`
	`[A,B,C,D] = cheby1(n,Rp,Wn,options)`

Chebyshev Type II	`[b,a] = cheby2(n,Rs,Wn,options)`
	`[z,p,k] = cheby2(n,Rs,Wn,options)`
	`[A,B,C,D] = cheby2(n,Rs,Wn,options)`

Elliptic	`[b,a] = ellip(n,Rp,Rs,Wn,options)`
	`[z,p,k] = ellip(n,Rp,Rs,Wn,options)`
	`[A,B,C,D] = ellip(n,Rp,Rs,Wn,options)`

Ejemplos:

```
[b,a] = butter(5,0.4);                    % Lowpass Butterworth
[b,a] = cheby1(4,1,[0.4 0.7]);            % Bandpass Chebyshev Type I
[b,a] = cheby2(6,60,0.8,'high');          % Highpass Chebyshev Type II
[b,a] = ellip(3,1,60,[0.4 0.7],'stop');   % Bandstop elliptic
```

b) DISEÑO DE IIR EN FORMA DIRECTA

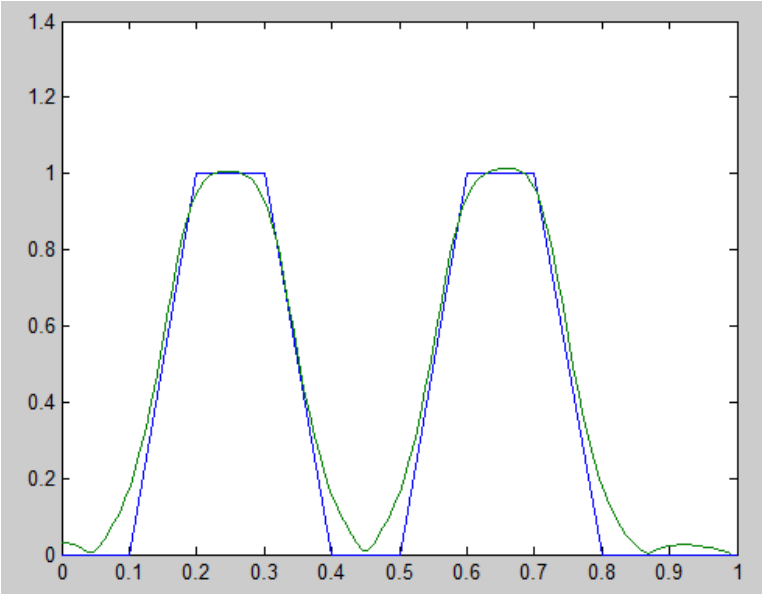
Se diseña en forma directa especificando la respuesta en frecuencia. El método encuentra la transformada inversa FFT y la resuelve utilizando la ecuación Yule – Walker.

[b,a] = yulewalk(n,f,m)

La frecuencia f es un vector de 0 a 1, donde 1 representa la frecuencia de Nyquist. La magnitud m es un vector que contiene la respuesta de la magnitud deseada en los puntos de f.

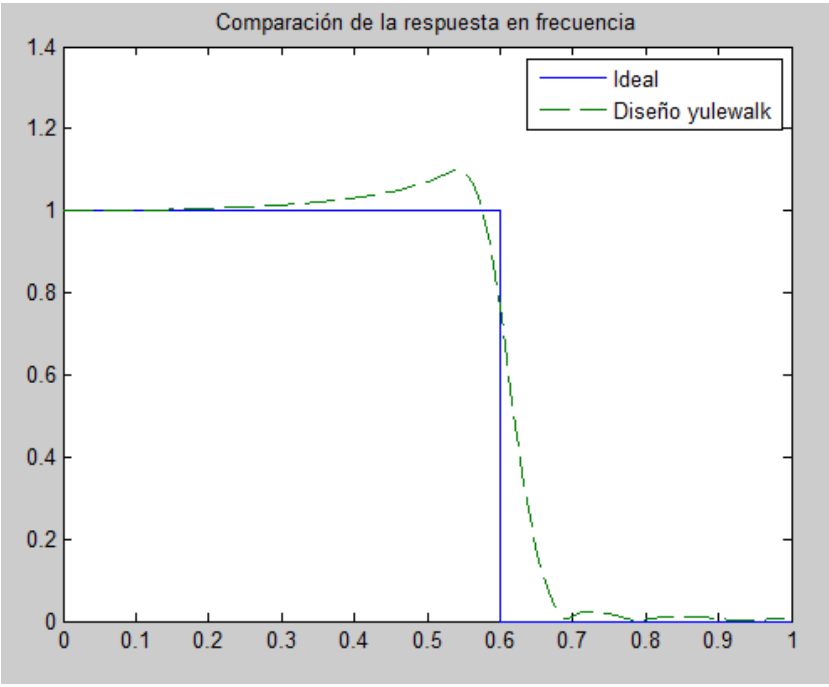
Ejemplo: Filtro multibanda de orden 10

```
m = [0   0   1   1   0   0   1   1   0 0];
f = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1];
[b,a] = yulewalk(10,f,m);
[h,w] = freqz(b,a,128);
plot(f,m,w/pi,abs(h))
```



Ejemplo: Filtro pasa bajo de orden 8

```
f = [0 0.6 0.6 1];
m = [1 1 0 0];
[b,a] = yulewalk(8,f,m);
[h,w] = freqz(b,a,128);
plot(f,m,w/pi,abs(h),'--')
legend('Ideal','Diseño yulewalk ')
title('Comparación de la respuesta en frecuencia')
```



2. DISEÑO DE FILTROS FIR

Los filtros FIR tienen las siguientes ventajas:

- Tienen fase lineal
- Son siempre estables
- Eficientes realizaciones en hardware
- Transientes de duración finita

La principal desventaja es que el orden del filtro para una similar respuesta es mucho más alto que la de un filtro IIR.

$$H(z) = B(z) = b(0) + b(1)z^{-1} + \cdots + b(M)z^{-M}$$

$$H\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n}$$

$$\left(e^{j\frac{2\pi k}{N}}\right) = \sum_{n=0}^{N-1} h(n)e^{-j\frac{2\pi k}{N}n} \qquad Transformada\ Discreta\ de\ Fourier\ \ DFT$$

$$h(n) = \frac{1}{N}\sum_{k=0}^{N-1} H\left(e^{j\frac{2\pi k}{N}}\right)e^{j\frac{2\pi}{N}n}\ \ Transformada\ inversa\ de\ Fourier\ \ IDFT$$

$$H\left(e^{j\omega}\right) = \left|H(e^{j\omega})\right|e^{j\phi(\omega)} \quad Magnitud\ y\ Fase$$

$$H(w) = A(w)e^{j\phi(\omega)}, \quad A(w): amplitud$$

Respuesta en frecuencia:

$$H(w) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n}$$

$$H(w) = e^{-j\omega M}\sum_{n=0}^{N-1} h(n)e^{j\omega(M-n)}$$

$$H(w) = A(w)e^{j(k_1+Mw)}$$

A(w) es real si: $k1 = 0, \quad o, \quad k1 = \frac{\pi}{2}$

Simetría par: $k1 = 0$

$h(n) = h(N - n - 1) \qquad H(w) = A(w)e^{jM\omega}$

Definiendo un $M = \frac{N-1}{2}$

Si N es impar: Tipo I

$$A(w) = \sum_{n=1}^{M} 2h(M - n)cos\,(wn) + h(M)$$

Si N es par: Tipo II

$$A(w) = \sum_{n=1}^{N/2} 2h\left(\frac{N}{2} - n\right) cos\,(w\left(n - \frac{1}{2}\right))$$

Simetría Impar: $k1 = \frac{\pi}{2}$

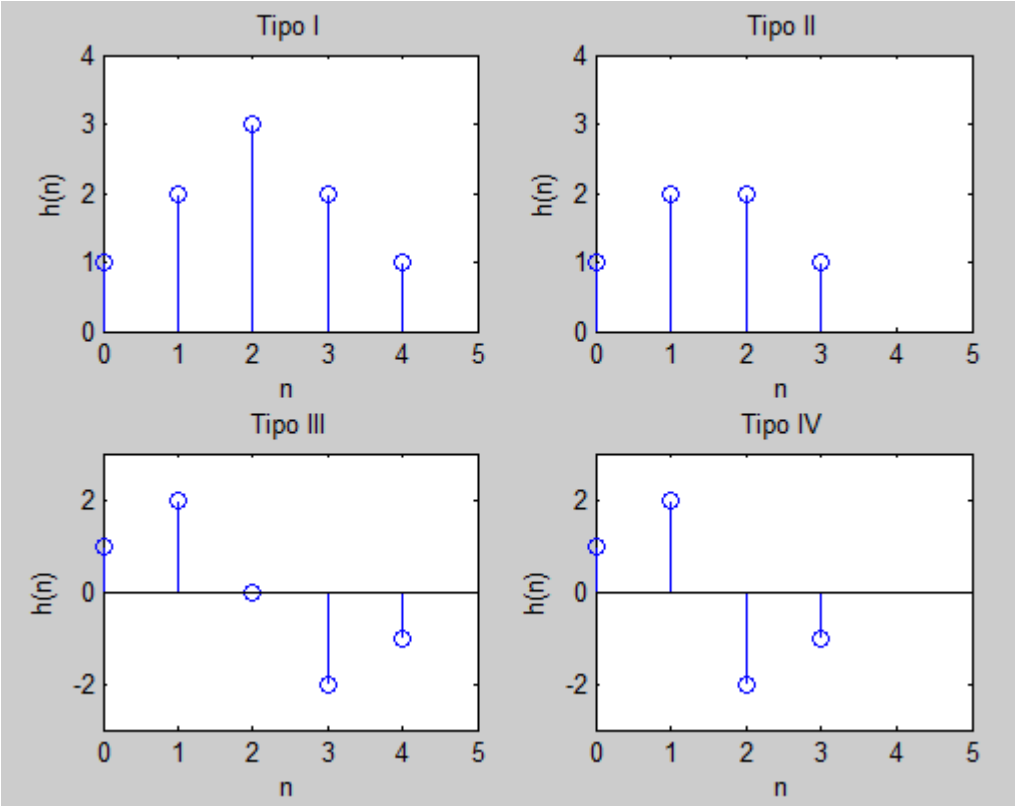
$h(n) = -h(N - n - 1) \qquad H(w) = jA(w)e^{-jM\omega}$

Si N es Impar: Tipo III

$$A(w) = \sum_{n=0}^{M-1} 2h(n)cos\,(w(M - n))$$

Si N es Par: Tipo IV

$$A(w) = \sum_{n=0}^{\frac{N}{2}-1} 2h(n)cos\,(w(M - n))$$



MÉTODOS DE DISEÑO

$$h(n) = \frac{1}{N}\left[A_o + \sum_{k=0}^{N-1} H\left(e^{j\frac{2\pi k}{N}}\right) e^{j\frac{2\pi}{N}n}\right]$$

a) MÉTODO DE VENTANEO

La truncación de una secuencia en el dominio del tiempo causa el fenómeno de Gibbs como una discontinuidad en el dominio de la frecuencia.

Si $h_d(n)$ es la secuencia del prototipo ideal, si se trunca por una ventana rectangular:

$$r(n) = \begin{cases} 1, & -M \leq n \leq M \\ 0, & en\ caso\ contrario \end{cases}$$

$$h(n) = h_d(n)r(n)$$

La multiplicación de dos funciones en el dominio del tiempo corresponde a la convolución de sus Transformadas de Fourier,

$$R(w) = \sum_{n=-\infty}^{\infty} r(n)e^{-j\omega n} = \sum_{n=-M}^{M} r(n)e^{-j\omega n} = \frac{sen(\frac{wN}{2})}{sen(\frac{w}{2})} \qquad (Fenómeno\ de\ Gibbs)$$

$$H(w) = H_d(w) * R(w)$$

Ventana triangular de Bartlett:

$$W(n) = \begin{cases} \frac{2(n+1)}{N+1}, & n = 0,1,2,\dots,\frac{N-1}{2} \\ 2 - \frac{2(n+1)}{N+1}, & n = \frac{N-1}{2},\dots,N-1 \\ 0, & En\ otro\ caso \end{cases}$$

Ventanas de coseno generalizado: Rectangular, Hanning, Hamming y Blackman

$$W(n) = \begin{cases} a - b\cos\left(\frac{2\pi(n+1)}{N+1}\right) + c\cos\left(\frac{4\pi(n+1)}{N+1}\right), & n = 0,1,2,\dots,N-1 \\ 0, & en\ otro\ caso \end{cases}$$

Ventana	a	b	c
Rectangular	1.0	0.0	0.0
Hanning	0.5	0.5	0.0
Hamming	0.54	0.46	0.0
Blackman	0.42	0.5	0.8

Ventana Kaiser:

$$W(n) = \begin{cases} \frac{I_o(\beta\sqrt{1 - \left(\frac{2(n+1)}{(N+1)^2}\right)}}{I_o(\beta)}, & n = 0,1,2,\dots,N-1 \\ 0, & En\ otro\ caso \end{cases}$$

La ventana rectangular es la mas simple pero presenta el fenómeno de Gobbs. La de Bartlett o triangular reduce el overshoot pero dispersa considerablemente la banda de transición. Las hanning, Hamming y Blackman proveen una truncación mas suave y una respuesta de frecuencia mejor. La mejor puede ser la de kaiser que permite ajustar el compromiso entre overshoot y banda de transición con el parámetro β.

Considérese un filtro digital ideal pasa bajo con frecuencia de corte w_o en rad/seg. Este filtro tiene una magnitud igual a 1 en frecuencias menores a w_o y magnitud igual a 0 en frecuencias entre w_o y π. Su respuesta al impulso es:

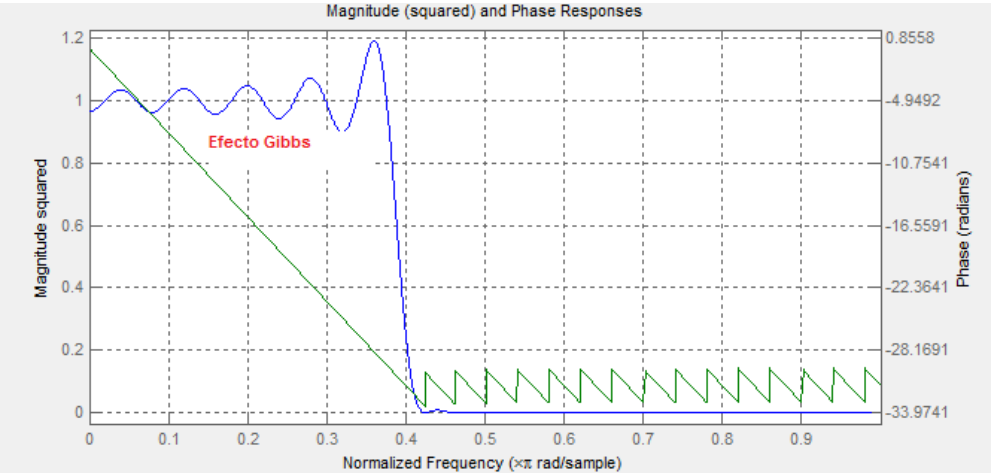
$$h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(w)e^{j\omega n}dw = \frac{1}{2\pi}\int_{-w_o}^{w_o} e^{j\omega n}dw = \frac{w_o}{\pi}sinc\left(\frac{w_o}{\pi}n\right)$$

Este filtro no es implementable porque su respuesta al impulso es infinito y no causal. Para crear una respuesta de duración finita hay que truncarlo por medio de una ventana.

Ejemplo:

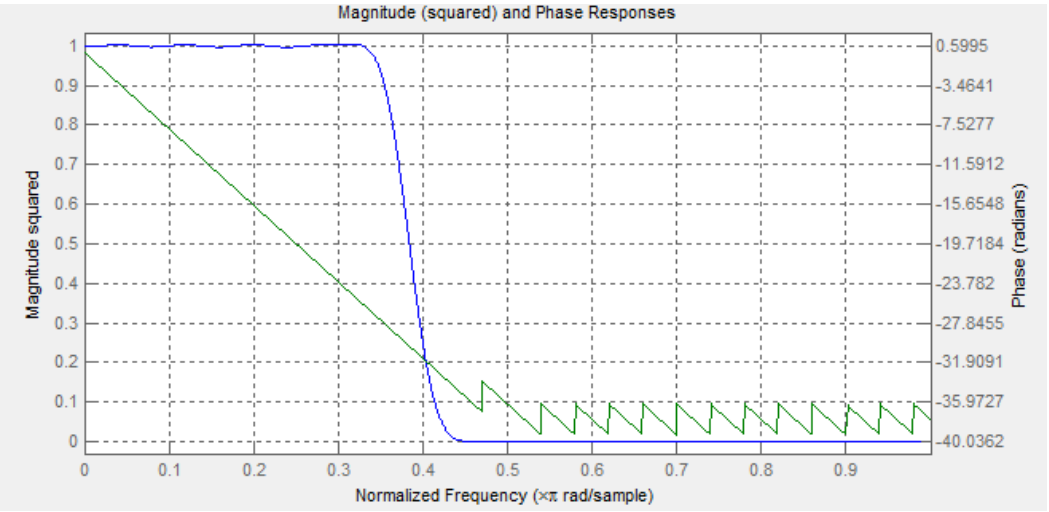
Filtro pasabajo de orden 51 con frecuencia de corte w_o=0.4π rad/seg, tiene coeficientes:

```
%longitud 51 ventana rectangular
n=-25:25;
%coeficientes
b=0.4*sinc(0.4*n);
fvtool(b,1)
```



Se presenta el efecto Gibbs en la banda de paso, esta distorsión se disminuye si se aplica una ventana tipo haming:

```
%longitud 51 ventana Hamming
n=-25:25;
%coeficientes
b=0.4*sinc(0.4*n);
b = b.*hamming(51)';
fvtool(b,1)
```



Las Funciones fir1 y fir2 de Matlab basan el proceso en el uso de ventanas. Dado el orden del filtro y la descripción del filtro deseado, estas funciones retornan la Transformada de Fourier Inversa del filtro con ventana.

La multiplicación de una ventana en el dominio del tiempo causa una convolución en el dominio de la frecuencia.

VENTANAS

En diseño de filtros digitales se escoge una ventana para amortiguar los efectos Gibbs que resulta de la truncación de una señal infinita en el tiempo.

Window	Function
Bartlett-Hann window	barthannwin
Bartlett window	bartlett
Blackman window	blackman
Blackman-Harris window	blackmanharris
Bohman window	bohmanwin
Chebyshev window	chebwin
Flat Top window	flattopwin

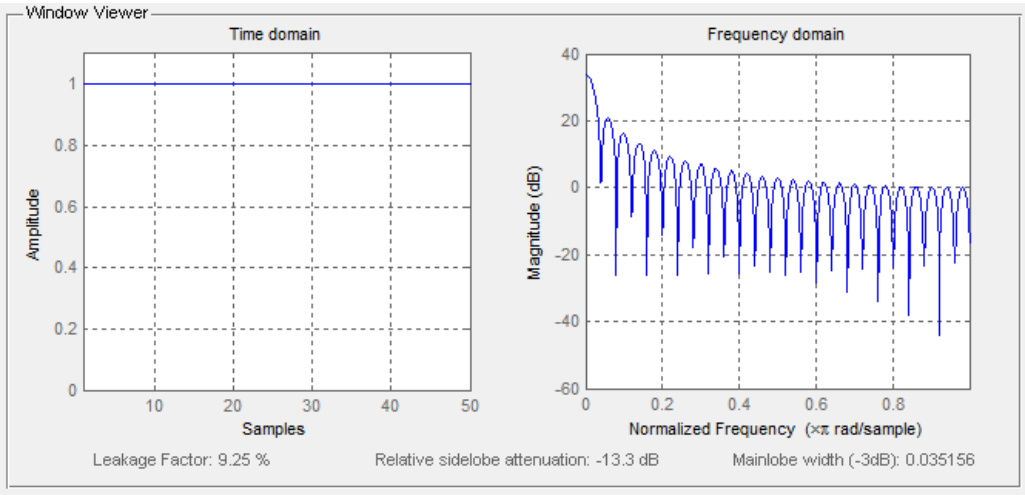
Gaussian window	gausswin
Hamming window	hamming
Hann window	hann
Kaiser window	kaiser
Nuttall's Blackman-Harris window	nuttallwin
Parzen (de la Valle-Poussin) window	parzenwin
Rectangular window	rectwin
Tapered cosine window	tukeywin
Triangular window	triang

EJEMPLOS:

Ventana Rectangular

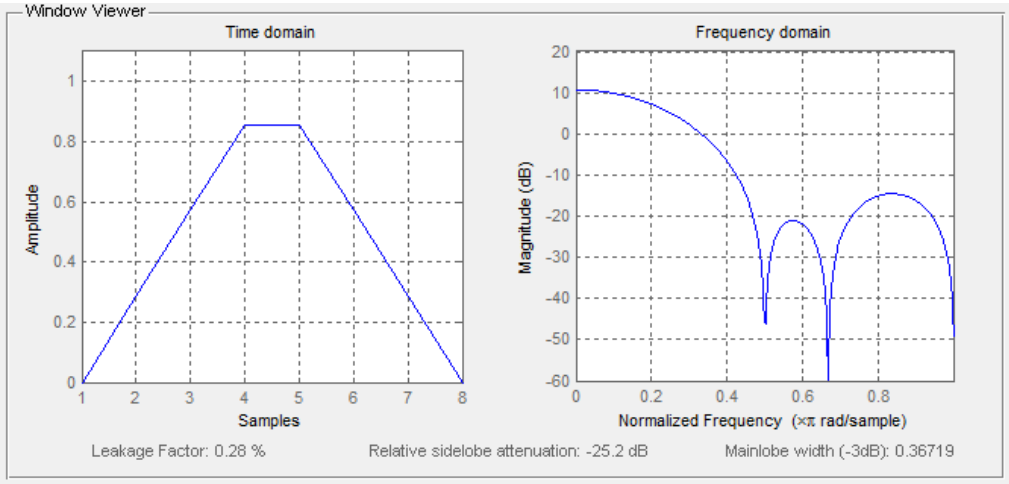
Ventana rectangular de longitud de 50

```
%ventana rectangular
n = 50;
w = rectwin(n);
wintool
```



Ventana Bartlett

```
%ventana de Bartlett de 8
n = 8;
w = bartlett(n);
wintool
```



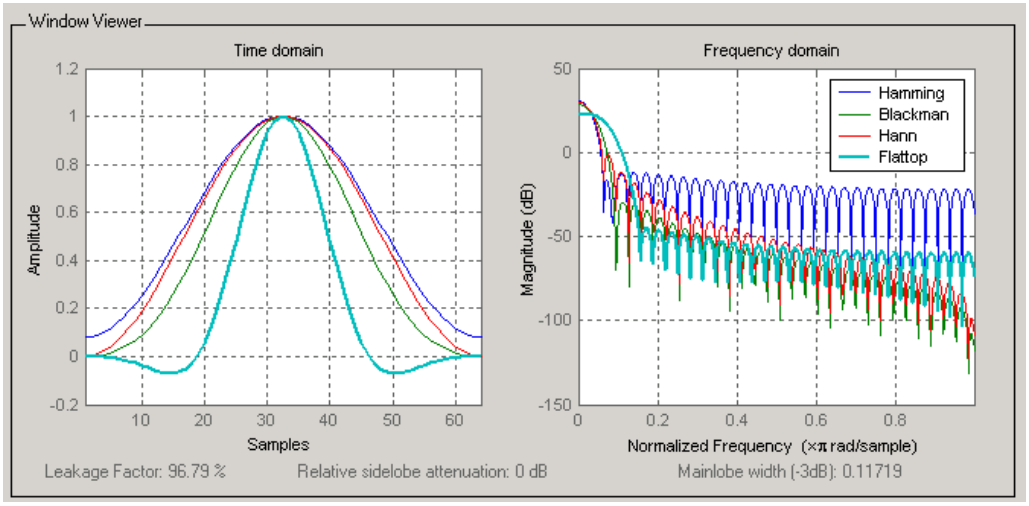

```
%w =[0;0.2857;0.5714;0.8571;0.8571;0.5714;0.2857;0]
```

Ventanas cosenoidales

Las ventanas Blackman, Flat Top, Hamming, Hann, and rectangular son casos especiales de de ventanas de coseno. Son combinaciones de secuencias senoidales con frecuencias de 0, 2π/N-1, 4π/N-1 donde N es el número de puntos de la ventana.

ind = (0:n-1)' *2 *pi/(n-1);
w = A - B* cos(ind) + C* cos(2*ind);

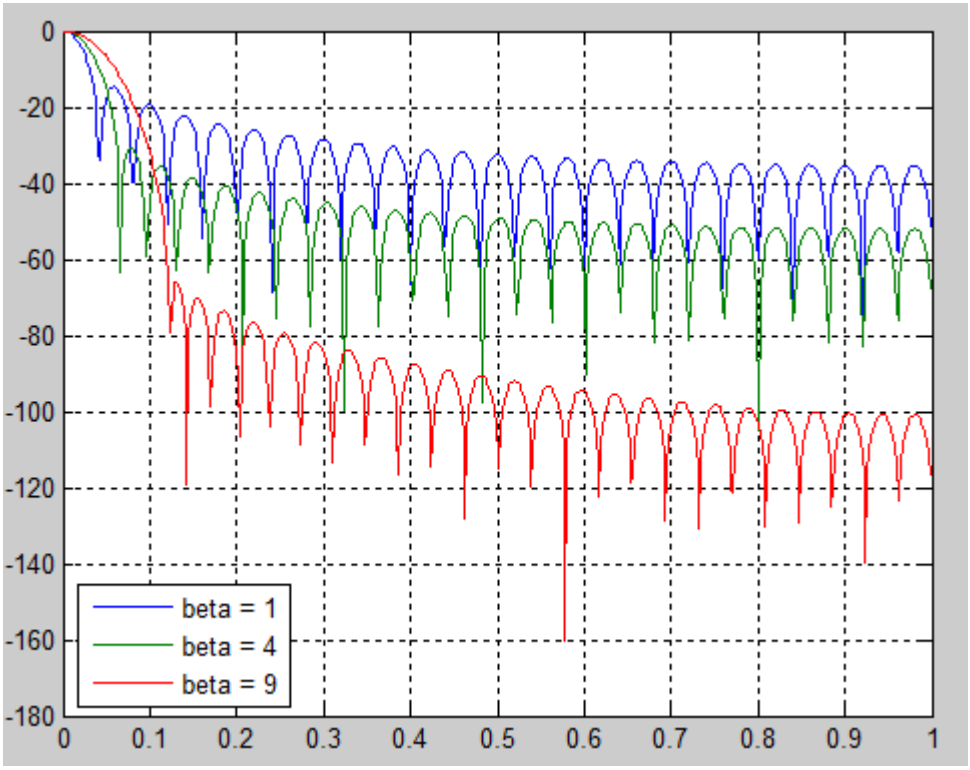
Para N=64:



Ventana Kaiser

La ventana de Kaiser es una aproximación para maximizar la energía del lóbulo principal frente a los lóbulos laterales. El parámetro β controla este peso del lóbulo principal.

```
n = 50;
w1 = kaiser(n,1);
w2 = kaiser(n,4);
w3 = kaiser(n,9);
[W1,f] = freqz(w1/sum(w1),1,512,2);
[W2,f] = freqz(w2/sum(w2),1,512,2);
[W3,f] = freqz(w3/sum(w3),1,512,2);
plot(f,20*log10(abs([W1 W2 W3]))); grid;
legend('beta = 1','beta = 4','beta = 9',3)
```



Para encontrar el orden del filtro se utiliza la función:

[n,Wn,beta,ftype] = kaiserord(f,a,dev)

f: frecuencia de corte de las bandas
a: amplitud deseada en las bandas
dev: especifica el riple de la banda de paso y la atenuación de la banda stop, en forma de ganancia absoluta no en dB

Para calcular los coeficientes del filtro se usa la función fir1:

b = fir1(n,Wn,kaiser(n+1,beta),ftype,'noscale')

ftype: es 'high' para pasa alto y 'stop' para banda stop. Para pasa banda puede ser 'dc-0' so la primera banda es banda stop o 'dc-1'si es pasa banda.

Algoritmo

$$\beta = \begin{cases} 0.1102(\alpha - 8.7), & \alpha > 50 \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 21), & 50 \geq \alpha \geq 21 \\ 0, & \alpha < 21 \end{cases}$$

Donde α es la atenuación en banda stop en dB

$\alpha = -20log_{10}\delta, \ \ con\ la\ condición\ de\ que \ \ \delta_p = \delta_s$

$\delta_p: riple\ en\ banda\ de\ paso, \qquad \delta_s = riple\ en\ banda\ stop$

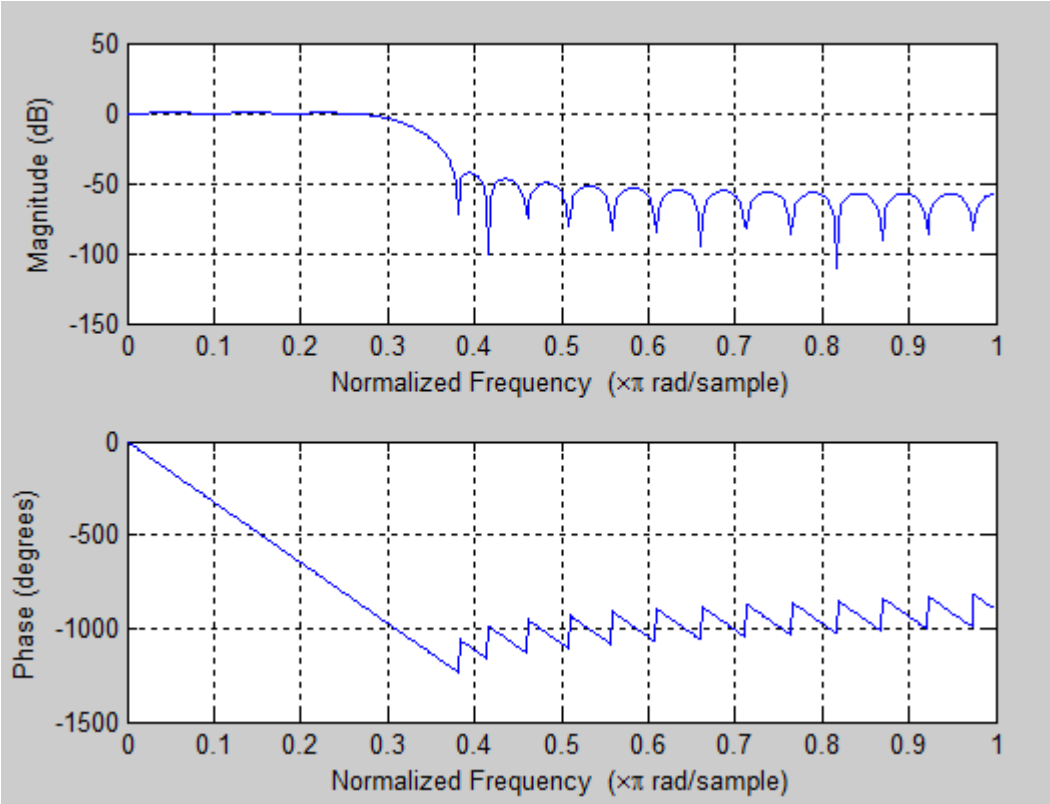
$$n = \frac{\alpha - 7.95}{2.285(\Delta w)}$$

Donde n es el orden del filtro y Δw es la región de transición más pequeña.

EJEMPLOS:

Diseñe un filtro pasabajo con banda de paso definida de 0 a 1 KHz y banda stop de 1500 Hz a 4 KHz. El riple de la banda de paso es del 5% y la atenuación en la banda stop de 40 dB.

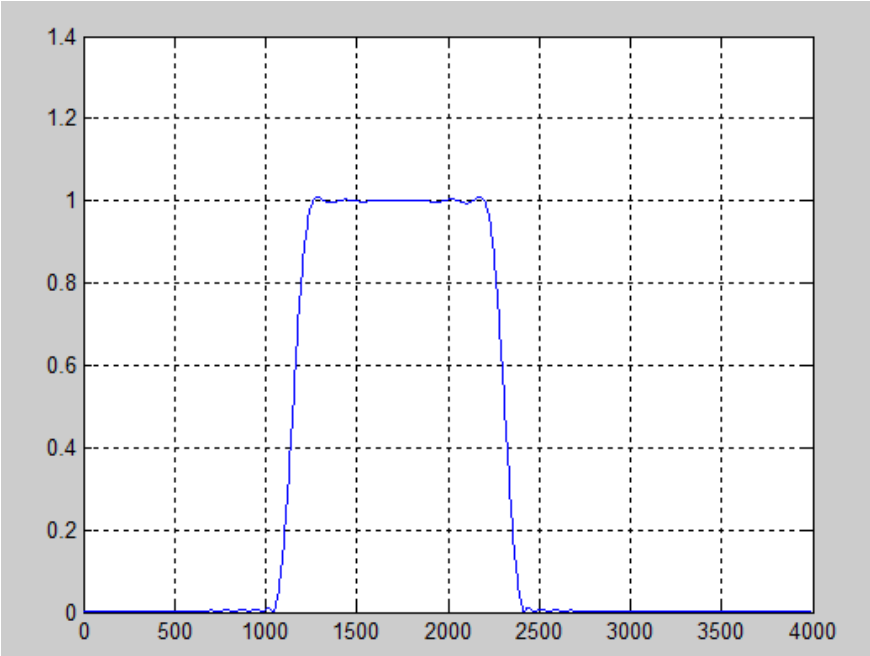
```
%orden del filtro kaiser
fs = 8000;
fc = [1000 1500];
mag = [1 0];
%5% =0.05, -40dB =20 log(0.01)
dev = [0.05 0.01];
[n,Wn,beta,ftype] = kaiserord(fc,mag,dev,fs);
h = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
freqz(h)
%n = 36, Wn = 0.3125, ftype =low, beta = 3.3953
```



Ejemplo:

Diseñar un filtro pasa banda de longitud impar, usar fir1

```
%filtro de longitud impar (orden par)
fs = 8000;
fc = [1000 1300 2210 2410];
mag = [0 1 0];
dev = [0.01 0.05 0.01];
[n,Wn,beta,ftype] = kaiserord(fc,mag,dev,fs);
n = n + rem(n,2);
h = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
[H,f] = freqz(h,1,1024,fs);
plot(f,abs(H)), grid on
%n = 90, beta = 3.3953, ftype = DC-0
```



Ejemplo:

Diseñar un filtro pasabajo con corte en la banda de paso de 1500 Hz y corte en banda stop de 2000 Hz, riple en banda de paso de 0.01 y en banda stop de 0.1, frecuencia de muestreo de 8000 Hz.

```
[n,Wn,beta,ftype] = kaiserord([1500 2000],[1 0],[0.01 0.1],8000);
b = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
%n = 36, Wn = 0.4375, ftype = low
```

USO DE LA FUNCIÓN: fir1

Para diseño de filtros estándar pasa bajo, pasa alto, pasa banda y banda stop se utiliza fir1

$$B(z) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

Ejemplo:

n = 50;
Wn = 0.4;
%por defecto es pasa bajo con ventana de hamming
b = fir1(n,Wn);

En general:

b = fir1(n,Wn,'ftype',window)

- W_n es un número entre 0 y 1, donde 1 corresponde a la frecuencia de Nyquist.
- Si W_n = [w1 w2] fir1 retorna un filtro pasa banda con w1<w<w2
- Si W_n es un vector multielemento W_n =[w1 w2 w3 …wn] fir1 retorna un filtro multibanda de orden n con bandas 0<w<w1, w1<w<w2, …, wn<w<1

El tipo de filtro se programa con ftype:

- 'high' filtro pasa alto
- 'stop' para un banda stop, si wn=[w1 w2] es el rango de frecuencias de la banda stop
- 'dc-1' la primera banda del filtro multibanda es pasa banda
- 'dc-0' la primera banda del filtro multibanda es un banda stop

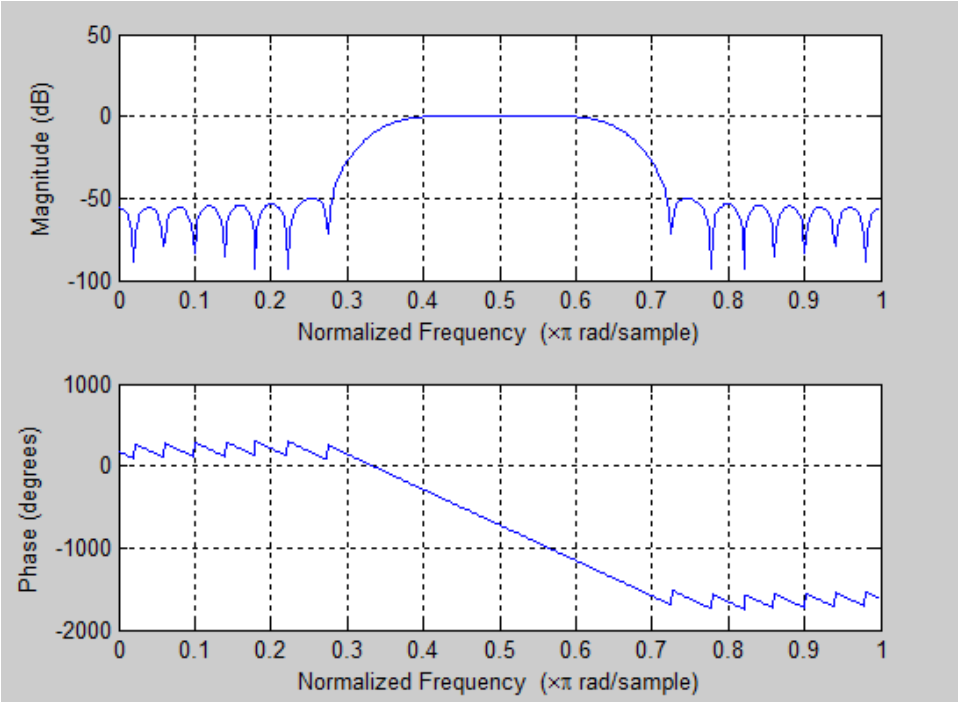
Si h(n) es la Transformada de Fourier Inversa de la respuesta en frecuencia ideal y w(n) es la ventana, los coeficientes del filtro son:

$B(n)= w(n)\ h(n), \quad 1 \leq n \leq N$

EJEMPLOS

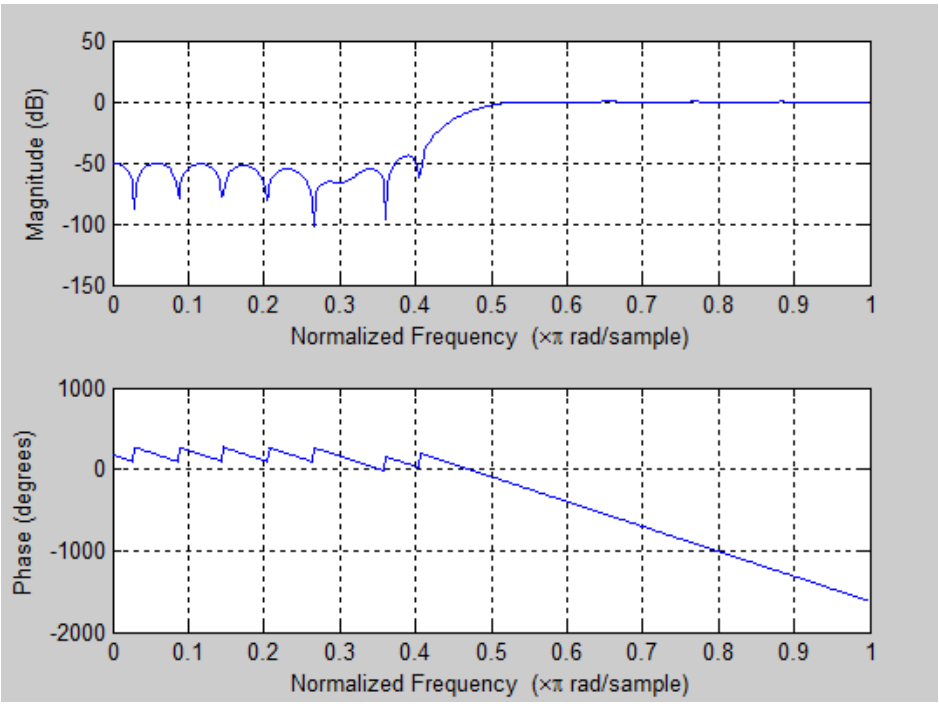
Diseñe un filtro FIR pasa banda de orden 48 con pasa banda de $0.35 \leq w \leq 0.65$

```
% uso de fir1
b = fir1(48,[0.35 0.65]);
freqz(b,1,512)
```



Diseñe un filtro FIR pasa alto para que atenúe las frecuencias posteiores a fc = 0.48 y ventana Chebyshev de 30 dB de riple.

```
%pasa alto
b = fir1(34,0.48,'high',chebwin(35,30));
freqz(b,1,512)
```



USO DE LA FUNCIÓN: fir2

La función fir2 también diseña filtros FIR ventaneados pero con respuesta de frecuencia lineal arbitraria. En contraste con fir1 que solamente diseña filtros estándar.

```
n = 50;
f = [0 .4 .5 1];
m = [1  1  0 0];
b = fir2(n,f,m);
```

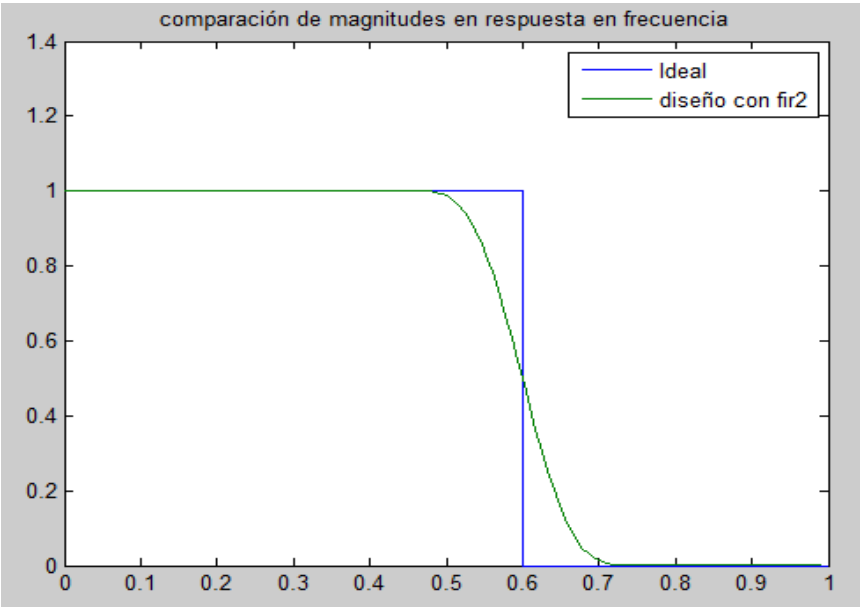
En general:

b = fir2(n,f,m,window)

Ejemplo:

Filtro pasa bajo de orden 30.

```
%uso de fir2
f = [0 0.6 0.6 1]; m = [1 1 0 0];
b = fir2(30,f,m);
[h,w] = freqz(b,1,128);
plot(f,m,w/pi,abs(h))
legend('Ideal','diseño con fir2')
title('comparación de magnitudes en respuesta en frecuencia')
```



b) MÉTODO DE MULTIBANDA CON BANDAS DE TRANSICIÓN

LEAST SQUARED ERROR

Las especificaciones de los filtros están dada generalmente en el dominio de la frecuencia y puesto que la energía de la señal está relacionada con el cuadrado de la señal, el criterio de error cuadrático es el indicado. Un método es considerar la integral del cuadrado del error (Integral squared error).

$$E = \frac{1}{2\pi}\int_{-\pi}^{\pi} |A(w) - A_d(w)|^2 dw$$

Se trata es de minimizar este error.

La DFT es:

$$H(w) = A(w) = \sum_{-\infty}^{\infty} h(n)e^{-j\omega n}$$

La IDFT es:

$$h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} A(w)e^{j\omega n} dw$$

$$E = \sum_{n=-\infty}^{\infty} |h(n) - h_d(n)|^2$$

Diferenciador

Un diferenciador ideal puede ser un filtro de fase lineal FIR. La respuesta en frecuencia del diferenciador es:

$$H(w) = jw \quad o \quad A(w) = w$$

Los coeficientes del filtro son:

$$h(n) = -\frac{1}{2\pi}\int_{-\pi}^{\pi} w\ sen(wn)dw = \frac{cos(\pi n)}{n},\ \ n \neq 0,\quad h(n) = 0, n = 0$$

Banda de transición:

Otra forma es minimizar el error aplicándolo a la banda de transición:

$$E = \int_{0}^{w1} |A(w) - A_d(w)|^2 dw + \int_{w2}^{\pi} |A(w) - A_d(w)|^2 dw$$

Uso de pesos:

Introducir una función de peso W(w) al error.

$$E = \frac{1}{\pi}\int_{0}^{\pi} W(w)|A(w) - A_d(w)|^2 dw$$

APROXIMACIÓN DE CHEBYSHEV

Minimiza el valor del máximo error y tienen un equiriple en el comportamiento de la respuesta en frecuencia.

$$|E(w)| = maxW(w).|D(w) - A(w)|$$

$D(w)$: $función\ deseada;\quad W(w)$: $función\ peso$

El algoritmo de Remes Exchange desarrollado por Parks y McClellan, hace que la función de error tome valores de ±δ para un conjunto de r+1 frecuencias f_m , m=1,…, r+1.

$$D(f_m) = \sum_{k=0}^{r-1} c_k\ cos(2\pi k f_m) + (-1)^m \delta,\qquad m = 1, \dots, r+1$$

$$max\left|D(f) - \sum_{k=0}^{r-1} c_k\ cos(2\pi k f_m)\right| = |\delta|$$

Función: firls (Least Squares)

Minimiza el error cuadrático medio entre la respuesta de frecuencia deseada y la respuesta de frecuencia actual.

$$E = \sum_{i=1}^{M} \left[\left| H(e^{j\omega i}) \right| - \left| H_d(e^{j\omega i}) \right| \right]^2$$

b = firls(n,f,a)

b(k) = b(n+2-k)

f: es un vector de parejas de puntos de frecuencias entre 0 y 1, donde 1 corresponde a la frecuencia de Nyquist.
a: Es un vector que contiene la amplitud deseada en los puntos especificados de f

En forma general:

b = firls(n,f,a,w,'ftype')

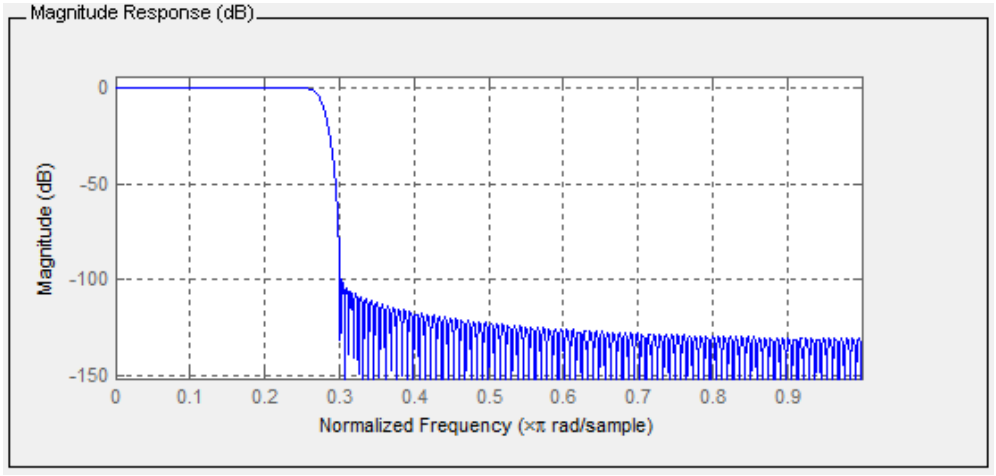
w: es el peso en la banda de frecuencia
ftype:
'hilbert' para filtros de fase lineal con simetría impar (tipo III y tipo IV)
b(k) = - b(n+2-k). Usa la Transformada de Hilbert
"differentiator' usa técnica especial de pesos en las bandas

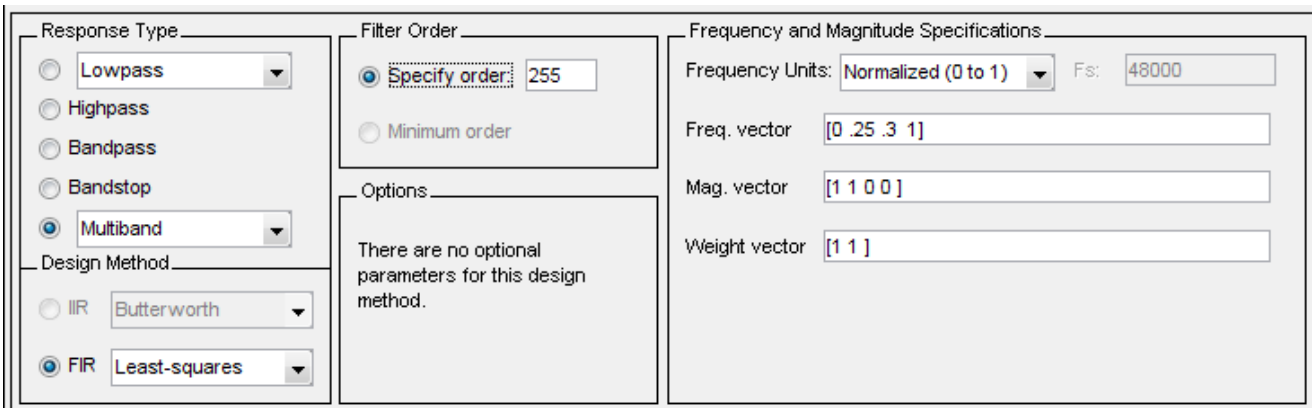
Ejemplo:

Diseñe un filtro pasa bajo de orden 255 con banda de transición:

```
%uso firls
b = firls(255,[0 0.25 0.3 1],[1 1 0 0]);
```

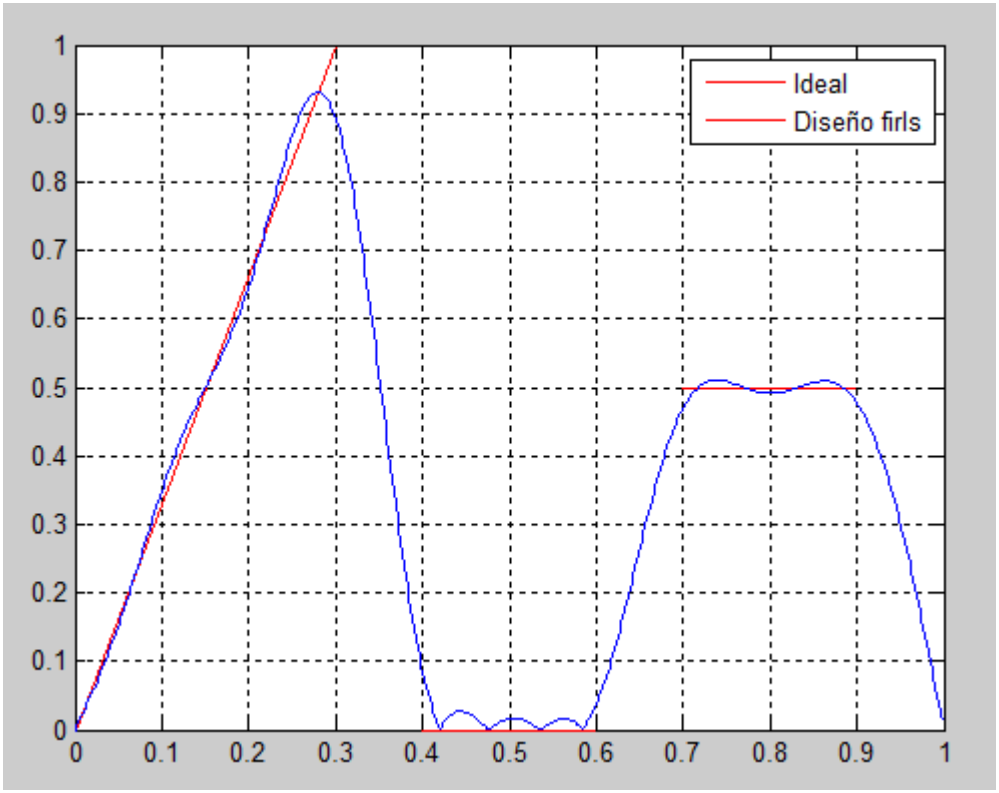
Usando fdatool



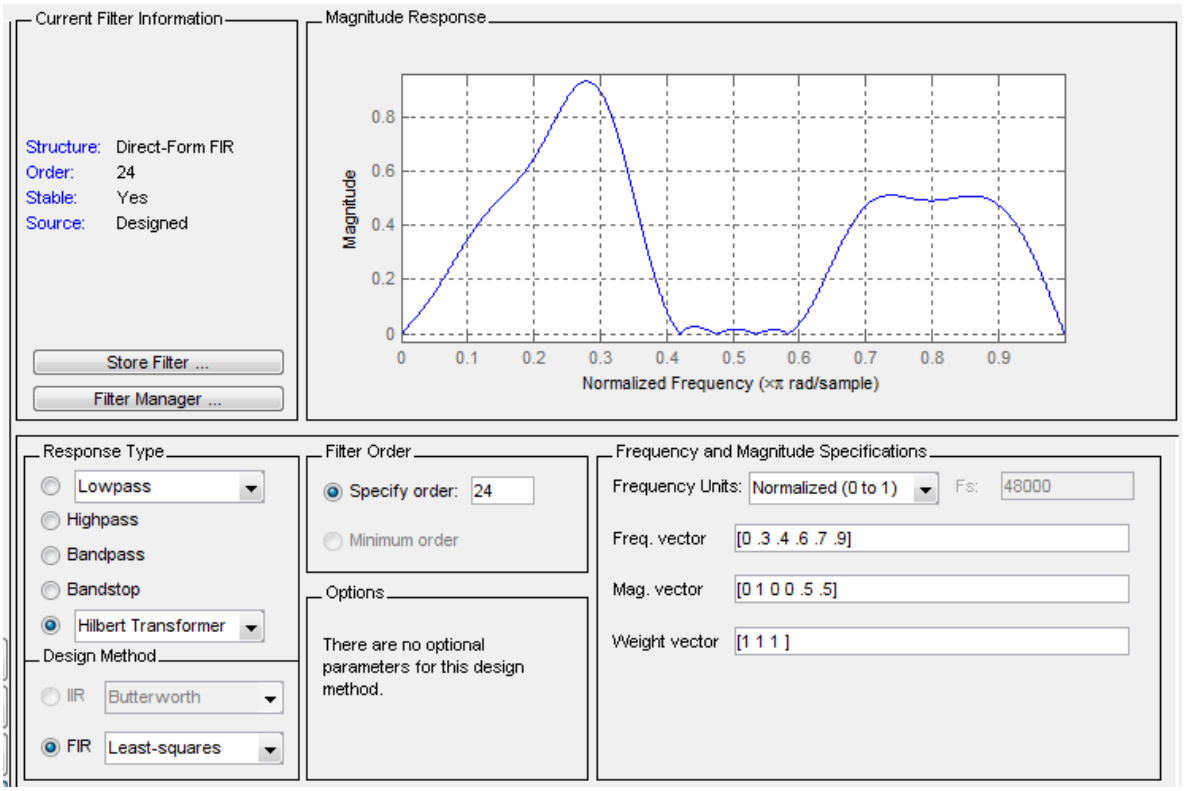


Diseñar un filtro antisimétrico de orden 24

```
%filtro antisimétrico de orden 24
F = [0 0.3  0.4 0.6  0.7 0.9];
A = [0  1   0  0  0.5 0.5];
b = firls(24,F,A,'hilbert');
for i=1:2:6,
   plot([F(i) F(i+1)],[A(i) A(i+1)],'r'), hold on
end
[H,f] = freqz(b,1,512,2);
plot(f,abs(H)), grid on, hold off
legend('Ideal','Diseño firls ')
```



Usando fdatool:

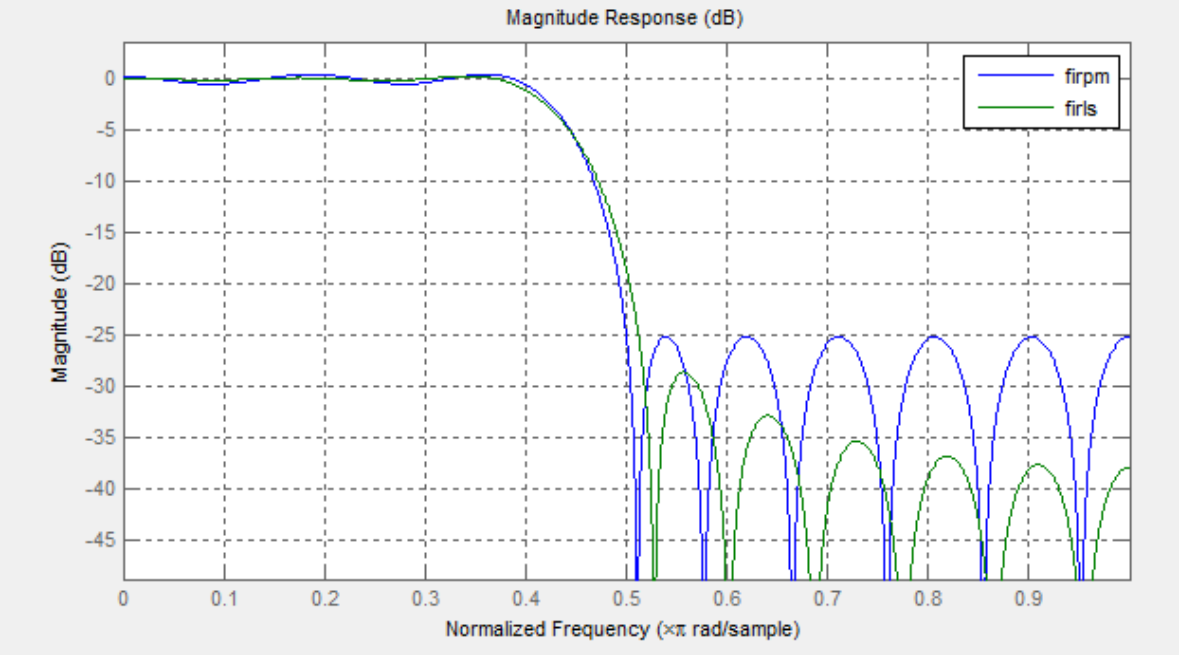


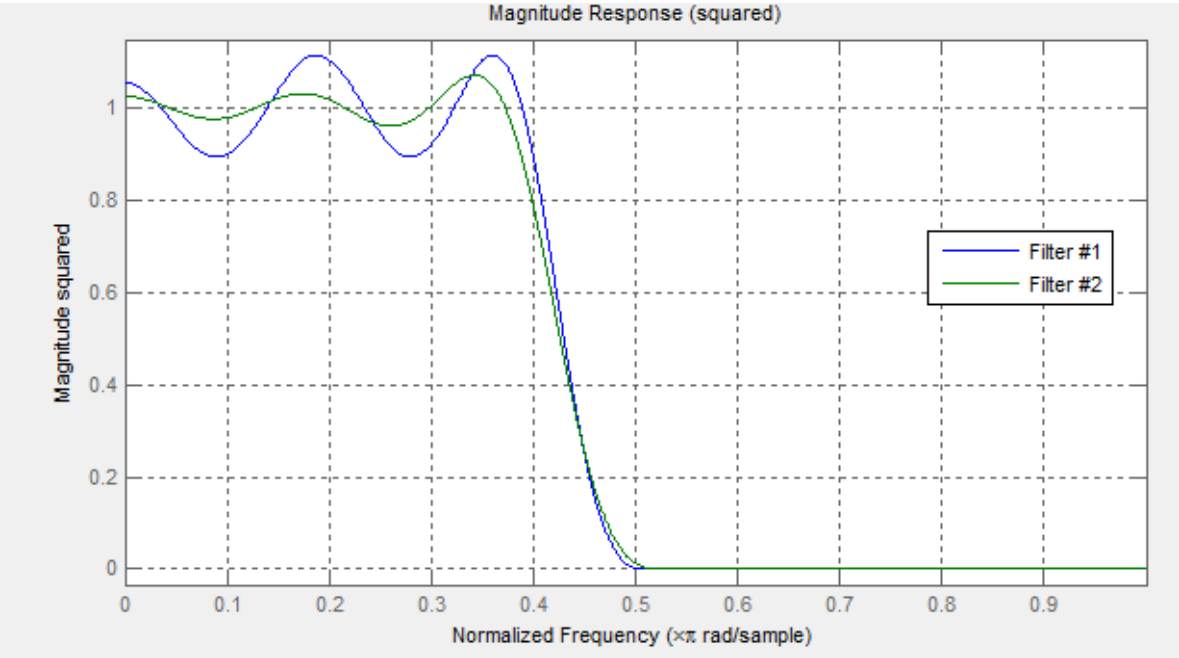
En resumen esta function diseña filtros de fase lineal tipo I, II, III, IV. Los tipo I y II son por defecto para n par e impar respectivamente. Los ‘hilbert’ y ‘differentiator’ producen tipo III (n par) y IV (n impar)

Función: firpm (Parks-McClellan)

Esta function implementa el algoritmo de Parks-McClellan que usa la teoría de Remez y la aproximación de Chebyshev. Minimizan el máximo error entre la respuesta de frecuencia deseada y la actual. Son filtros equiriple.

```
%uso de firpm
n = 20;                     % orden del filtro
f = [0 0.4 0.5 1];          % bordes de la banda de frecuencias
a = [1   1   0 0];          % amplitudes deseadas
b = firpm(n,f,a);
bb = firls(n,f,a);
fvtool(b,1,bb,1)
legend('firpm','firls')
```





Nótese que con firls presenta mejor respuesta en la banda de paso y en la banda stop, pero firpm es mejor en la banda de transición.

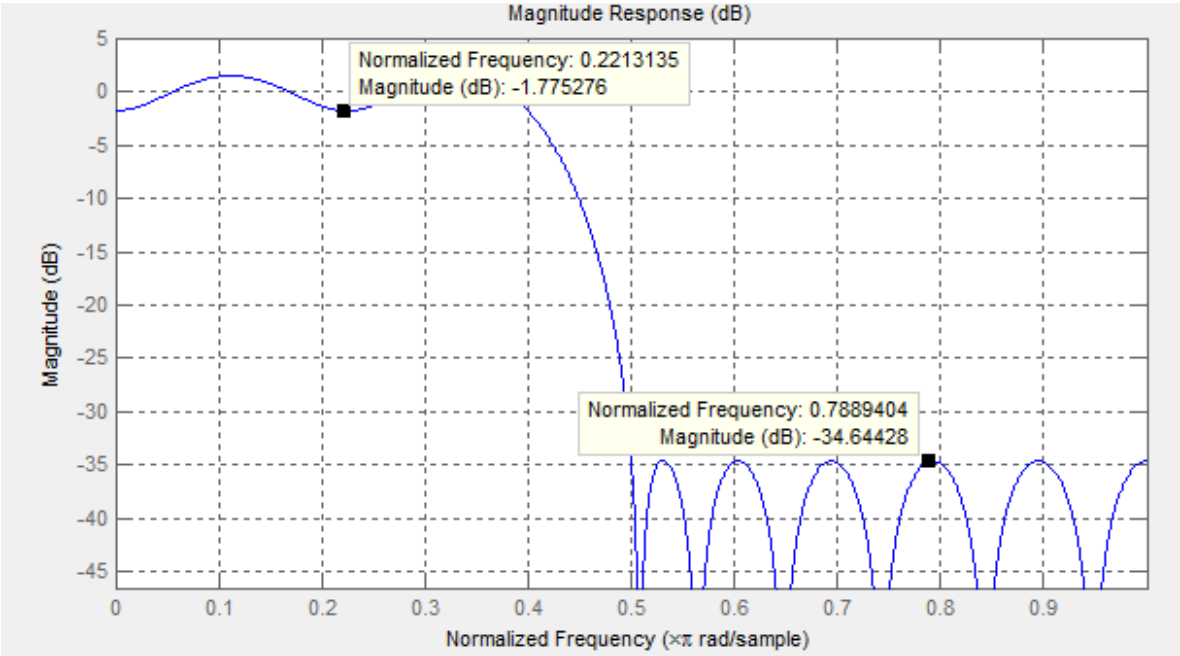
Mediante trazos de líneas firls y firpm pueden realizar filtros, pasa bajo, pasa alto, banda stop, psasa banda, multibanda.

```
%tipos de filtros con firls o firpm
%pasa banda
f = [0 0.3  0.4  0.7  0.8  1];
a = [0  0    1    1    0   0];
%pasa alto
a = [0  0    1   1];
f = [0 0.3  0.4  0.5  0.8  1];
%banda stop
a = [1  1    0    0    1   1];
%multibanda
f = [0 0.1 0.15 0.25 0.3 0.4 0.45 0.55 0.6 0.7 0.75 0.85 0.9 1];
a = [1  1   0    0    1   1   0    0    1   1   0    0    1  1];
```

Uso del vector peso

Tanto firls como firpm permiten minimizar el error en ciertas bandas de frecuencias especificando un vector peso. Por ejemplo, un filtro equiriple pasa bajo con riple 10 veces menor en la banda stop que en la banda de paso es:

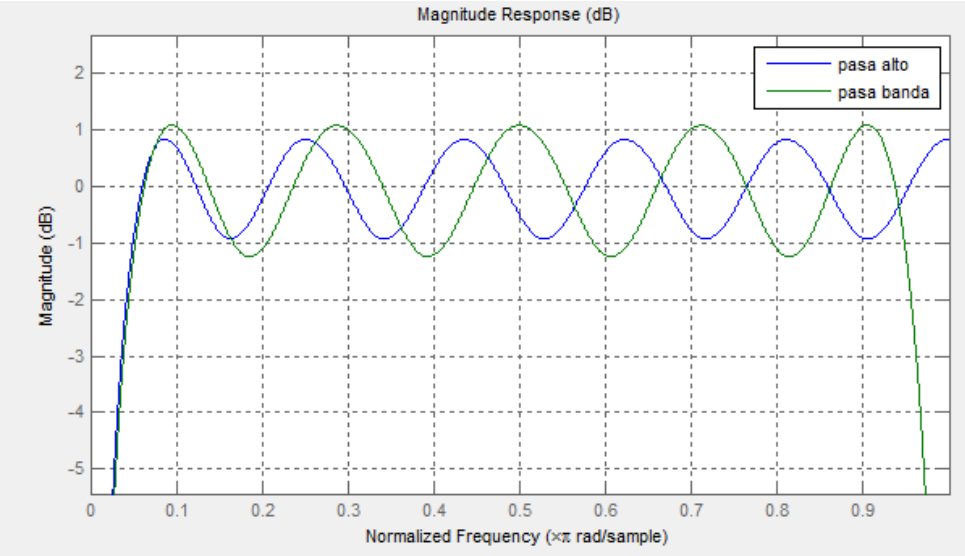
```
%uso de pesos en las bandas
n = 20;                 % orden del fitro
f = [0 0.4 0.5 1];      % bordes de la bandas de frecuencias
a = [1   1    0   0];   % amplitudes deseadas
w = [1 10];             % vector peso
b = firpm(n,f,a,w);
fvtool(b,1)
```



Filtros antisimétricos/Transformada de Hilbert.

Para filtros de simetría impar Tipo III (orden par) o Tipo IV (orden impar)

```
%filtros antisimétricos
b = firpm(21,[0.05 1],[1 1],'h');        % Pasa alto Hilbert
bb = firpm(20,[0.05 0.95],[1 1],'h');    % pasa banda Hilbert
fvtool(b,1,bb,1)
legend('pasa alto', 'pasa banda')
```



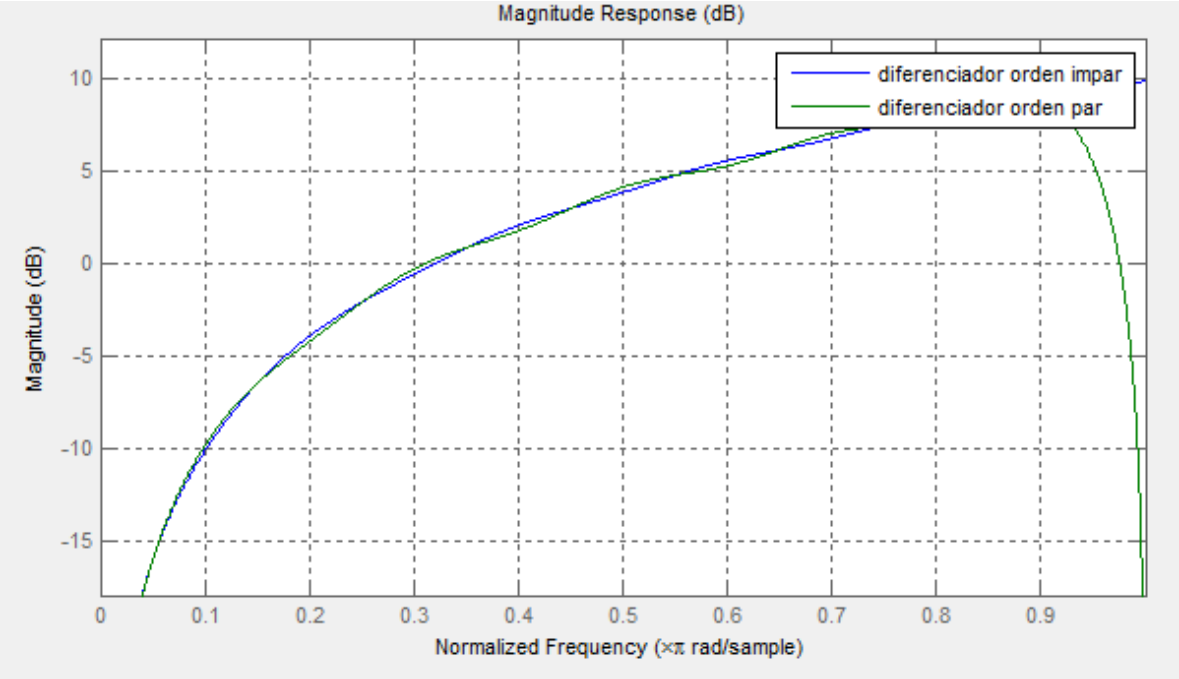
La señal analítica de una señal compleja x está compuesta por la parte real de x y la Transformada de Hilbert como la parte imaginaria (orden par).

```
%señal analítica
fs = 1000;              % frecuencia de muestreo
t = (0:1/fs:2)';        % vector tiempo
x = sin(2*pi*300*t);    % señal senoidal de 300 Hz
xh = filter(bb,1,x);    % Transformada Hilbert de x
xd = [zeros(10,1); x(1:length(x)-10)];      % 10 muestras de
retardo
xa = xd + j*xh;                             % señal analítica
```

Diferenciadores

La diferenciación de una señal en el dominio del tiempo es equivalente a la multiplicación de su transformada de Fourier por una función rampa. Diferenciar una señal es pasarla por un filtro H(w)= jw

```
%diferenciador
b = firpm(21,[0 1],[0 pi],'d');
bb = firpm(20,[0 0.9],[0 0.9*pi],'d');
fvtool(b,1,bb,1)
legend('diferenciador orden impar','diferenciador orden par')
```



c) MÉTODO DEL MÍNIMO CUADRADO RESTRINGIDO
Constrained Least Squares (CLS)

La restricción consiste en que no es necesario definir las bandas de transición. Permite considerar umbrales más altos o más bajos con riple máximo permisible.

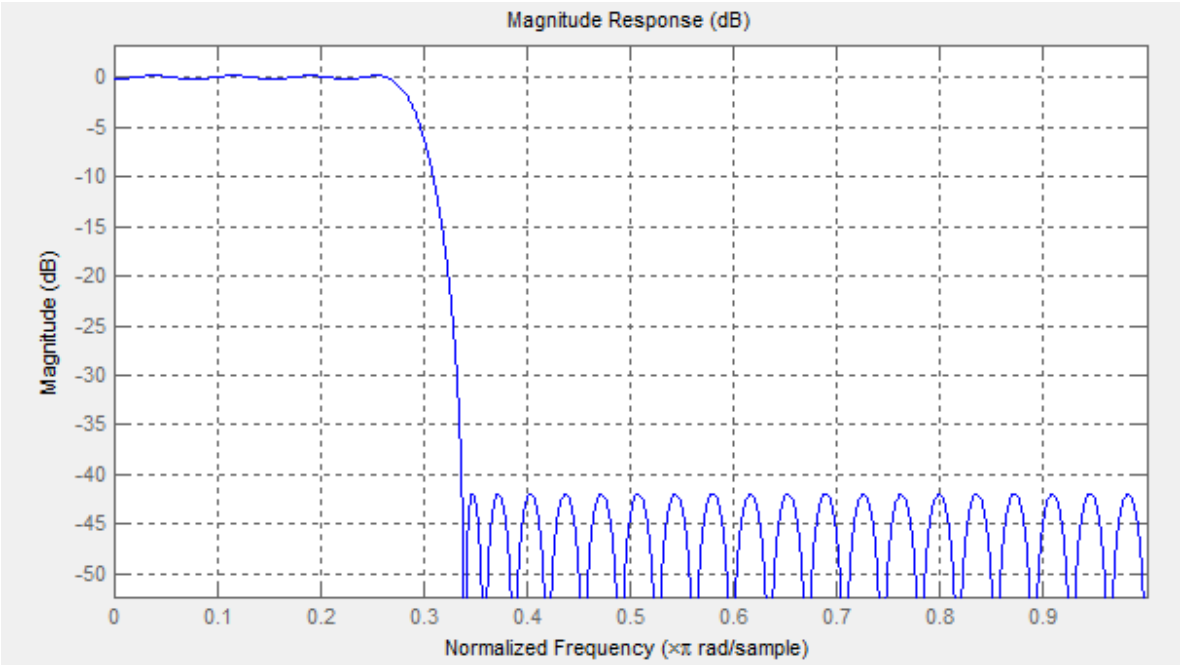
Función fircls1

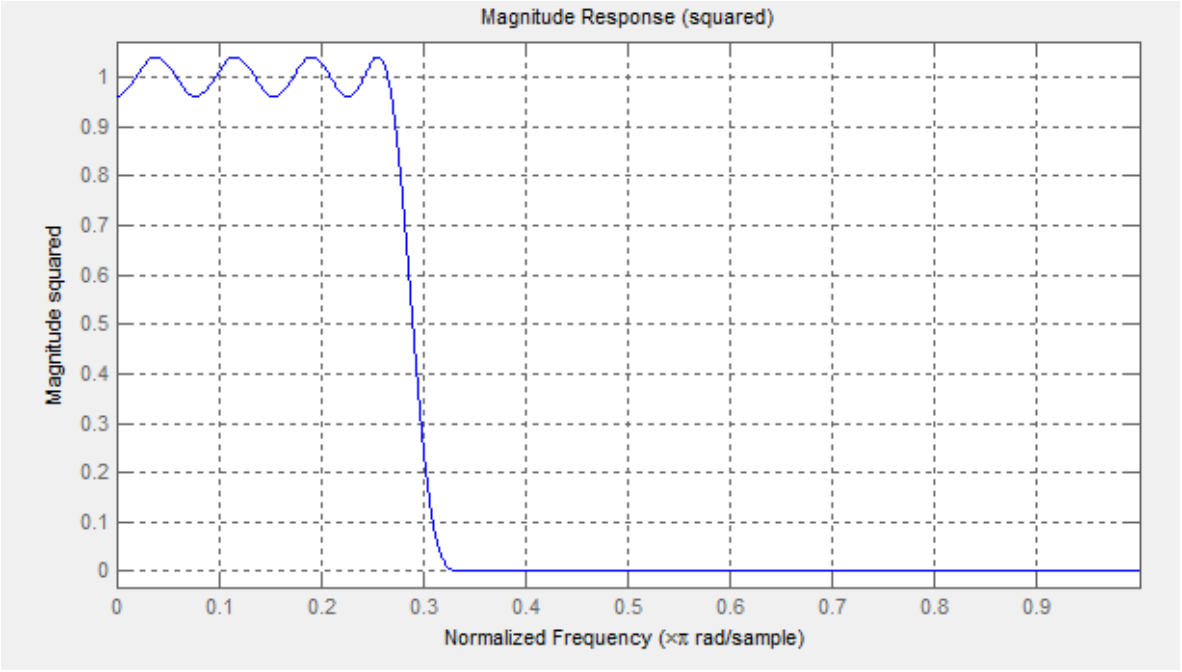
Se usa para el diseño de pasa bajo o pasa alto.

b = fircls1(n,wo,dp,ds)
b = fircls1(n,wo,dp,ds,'high')

w_o: es la frecuencia de corte normalizada
d_p: riple en la banda de paso
d_s: riple en la banda stop

```
%uso de fircls1
n = 55;      wo = 0.3;
dp = 0.02;   ds = 0.008;
b = fircls1(n,wo,dp,ds);
fvtool(b)
```



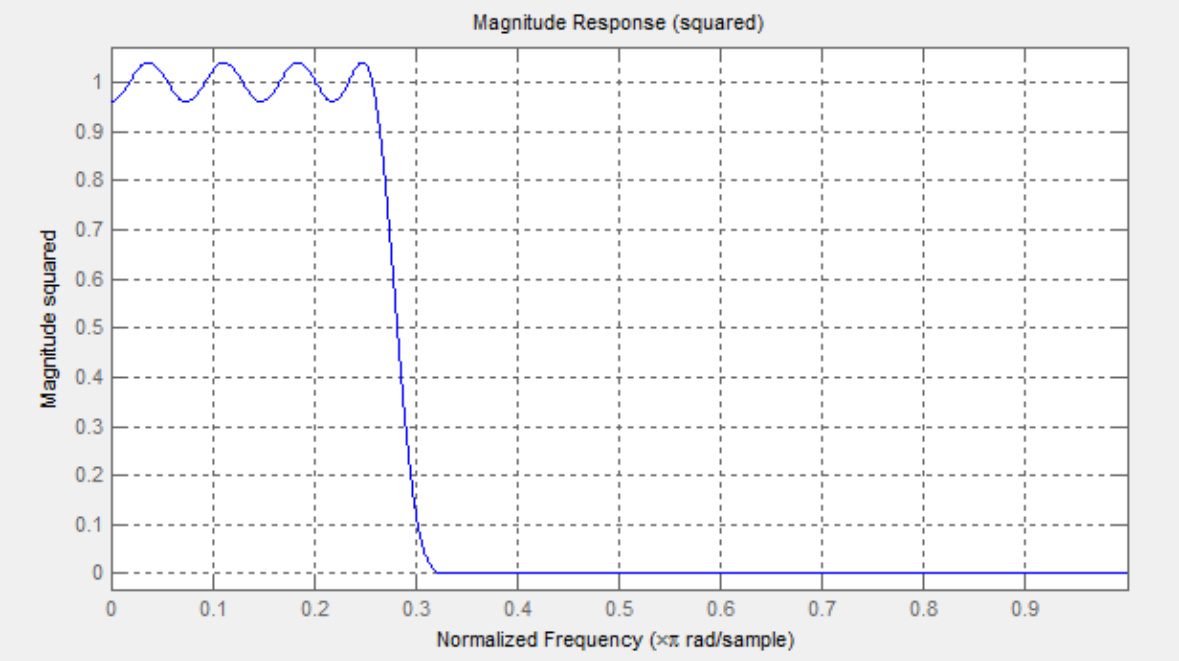


Usando pesos en las bandas:
b = fircls1(n,wo,dp,ds,wp,ws,k)

Donde k es la relación entre el error en la banda de paso y el error en la banda stop

$$k = \frac{\int_0^{\omega_p} |A(\omega) - D(\omega)|^2 d\omega}{\int_{\omega_s}^{\pi} |A(\omega) - D(\omega)|^2 d\omega}$$

```
%Uso de pesos en fircls1
n = 55;
wo = 0.3;
dp = 0.02;
ds = 0.004;
wp = 0.28;
ws = 0.32;
k = 10;
h = fircls1(n,wo,dp,ds,wp,ws,k);
fvtool(h,1)
```



Función fircls

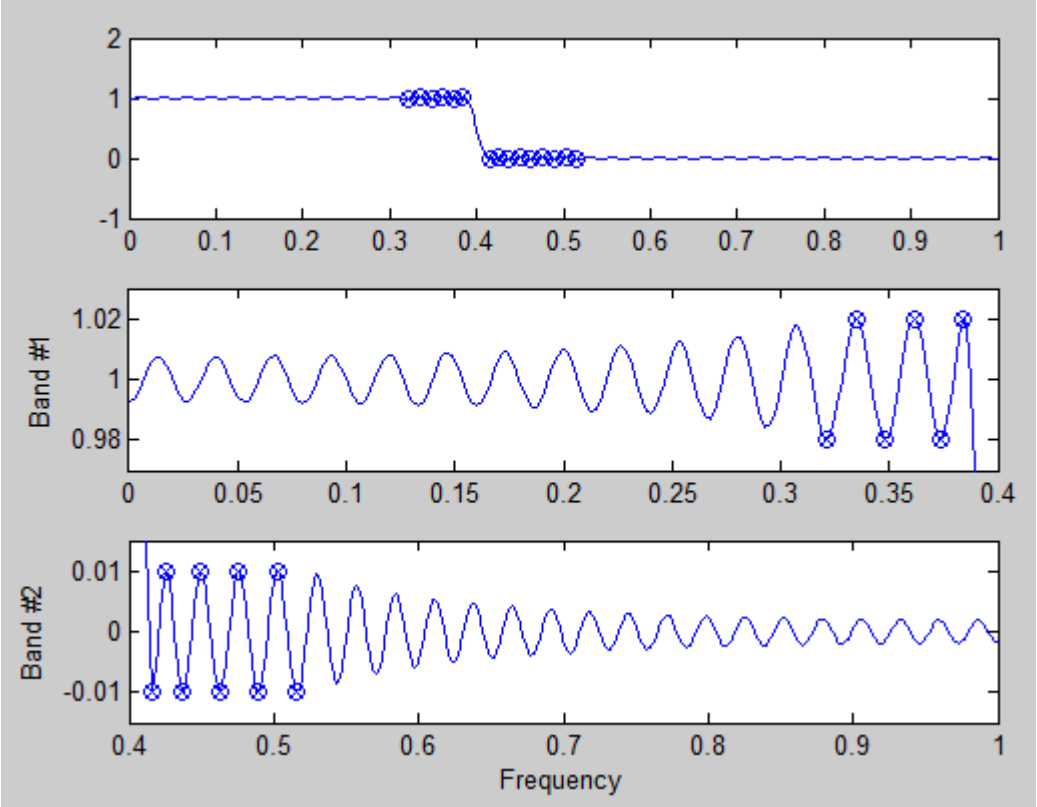
Se usa para filros FIR multibanda en el cual se pueden especificar la cantidad máxima de riple en cada banda.

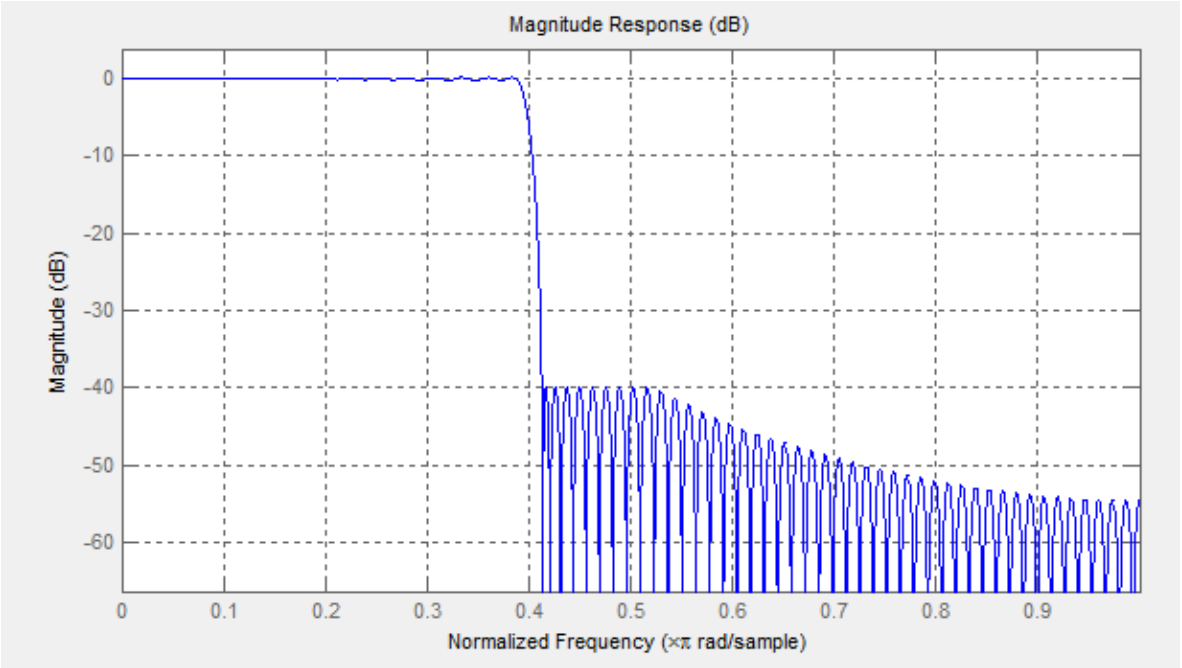
b = fircls(n,f,amp,up,lo)

f: vector de frecuencias de transición
amp: magnitudes

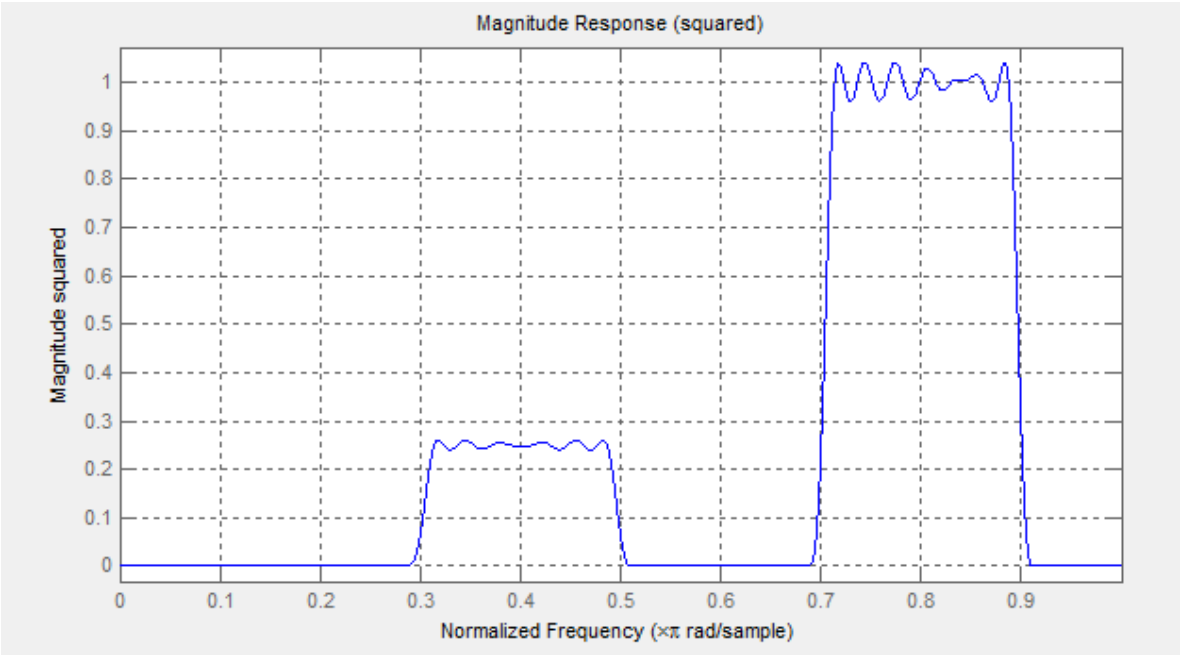
up, lo: define los límites alto y bajo para cada banda

```
%uso de fircls
n=150;
f=[0 0.4 1];
a=[1 0];
up=[1.02 0.01];
lo =[0.98 -0.01];
%con despliegue de las bandas
b = fircls(n,f,a,up,lo,'both');
fvtool(b)
```






```
n = 129;
f = [0 0.3 0.5 0.7 0.9 1];
a = [0 0.5 0 1 0];
up = [0.005 0.51 0.03 1.02 0.05];
lo = [-0.005 0.49 -0.03 0.98 -0.05];
h = fircls(n,f,a,up,lo);
fvtool(h,1)
```



IMPLEMENTACIÓN DE FILTROS DIGITALES

Después de haber diseñado el filtro, la función de transferencia o ecuación en diferencias del filtro debe implementarse y hay varias maneras de hacerlo. Estas maneras de realizarlas son representadas en diagramas en bloques y son llamadas estructuras del filtro y se escoge aquella que sea menos sensitiva a los errores en los coeficientes o que minimice el ruido introducido por la cuantización de los coeficientes.

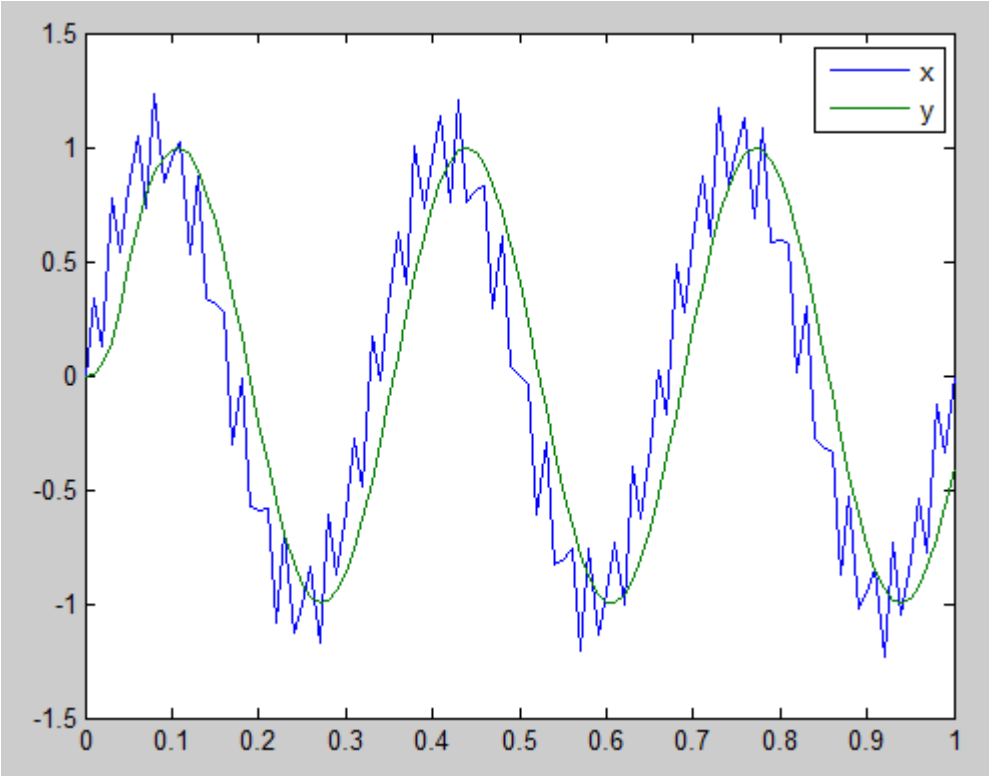
Función dfilt, filter

Procedimiento para la implementación:

1. Se generan los coeficientes del filtro IIR o FIR por el método de diseño apropiado
2. Crear el objeto del filtro de los coeficientes usando dfilt
3. Aplicar al objeto del filtro al dato x usando la función filter

Por ejemplo, diseñar, implementar como estructura forma directa traspuesta II y aplicar a un filtro Butterworth

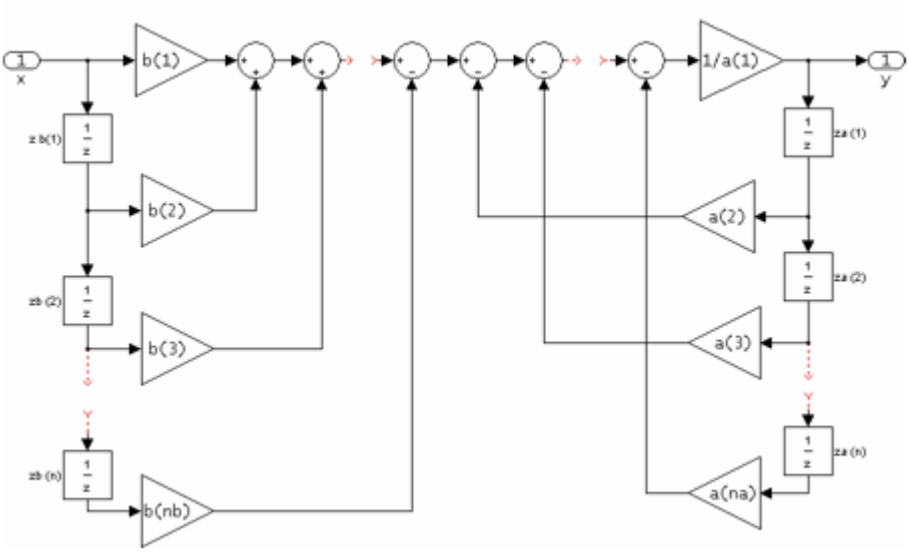
```
%implementación de filtros
[b,a] = butter(5,0.4);
Hd = dfilt.df2t(b,a);     % Forma directa II transpuesta
fs=100;
t=0:1/fs:1;
x=sin(2*pi*t*3)+0.25*sin(2*pi*t*40);
y=filter(Hd,x);
plot(t,x,t,y)
legend('x','y')
```



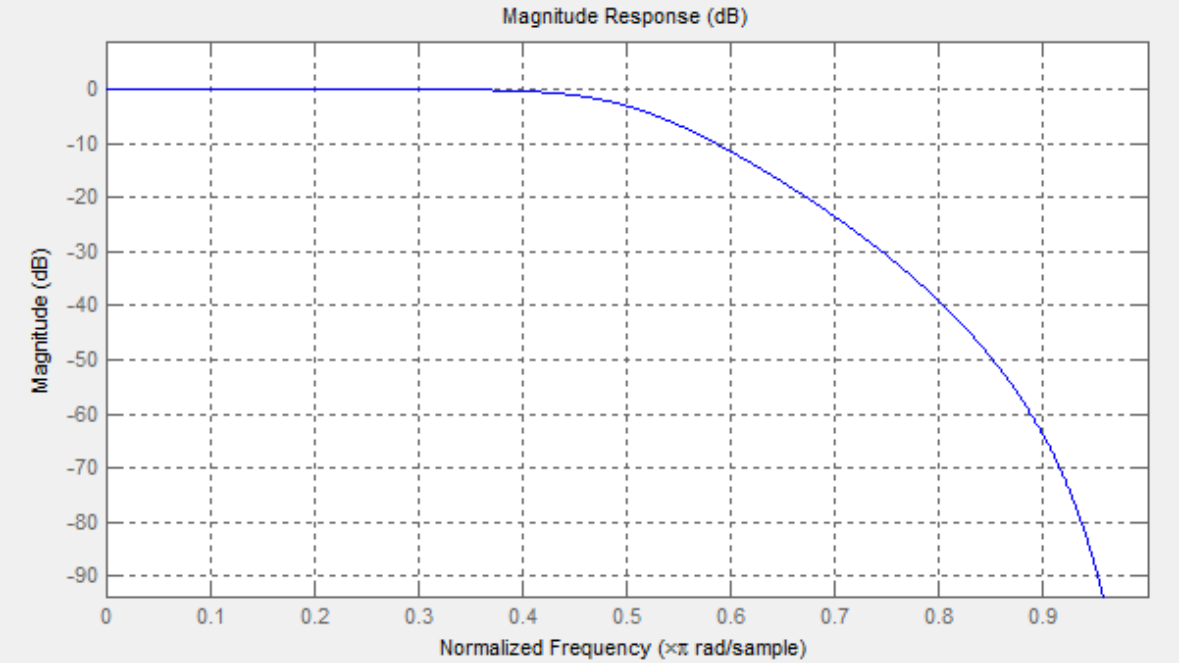
ESTRUCTURA DE FILTROS

1. IMPLEMENTACIÓN DE FILTROS IIR

a) Forma Directa I: dfilt.df1

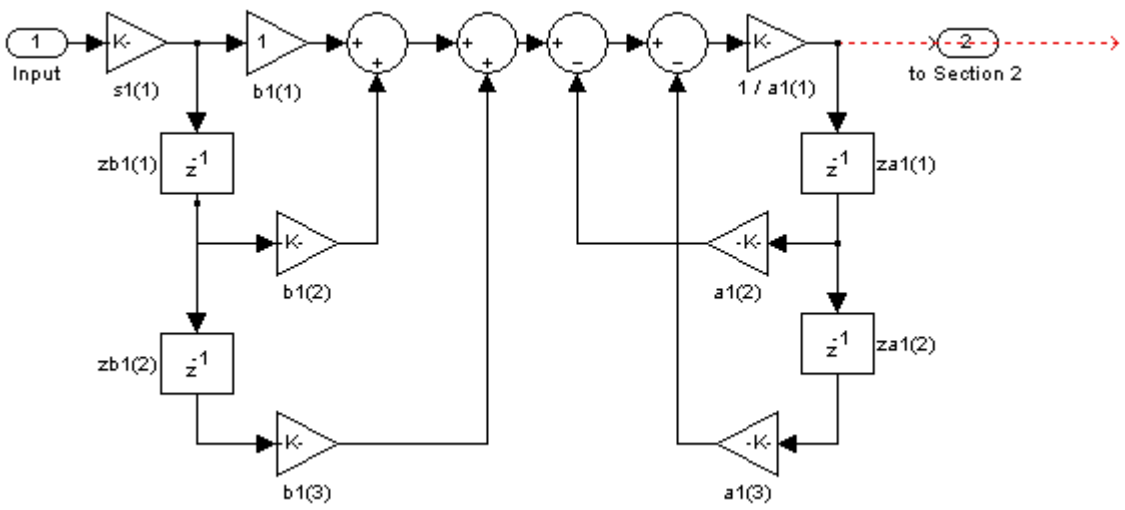


```
%forma directa I
[b,a] = butter(4,.5);
Hd = dfilt.df1(b,a);
num=get(Hd,'Numerator')
%num = 0.0940    0.3759    0.5639    0.3759    0.0940
den=get(Hd,'Denominator')
%den =  1.0000   -0.0000    0.4860   -0.0000    0.0177
fvtool(b,a)
```



b) Forma Directa I sos (second order section): dfilt.df1sos

El filtro es implementado en cascada por secciones de Segundo orden (tres coeficientes en el numerador y tres coeficientes en el denominador).

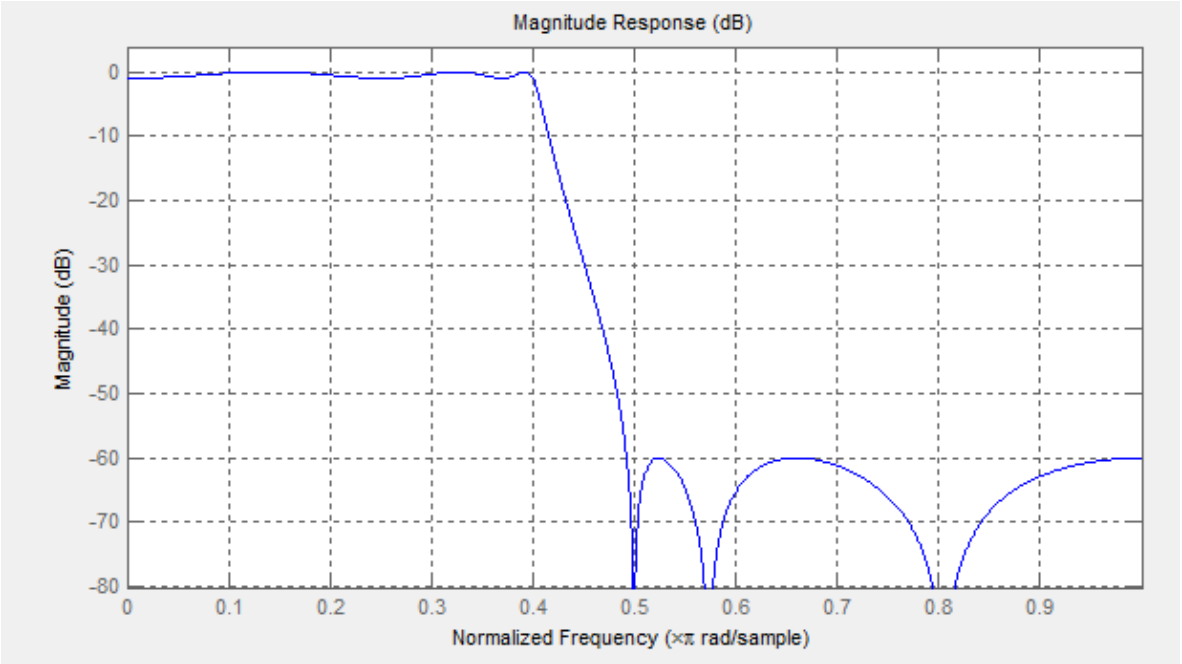


Ejemplo: para un filtro ellíptico , pasa bajode orden 6, Rs=1, Rp=60 dB, Wp=0.4

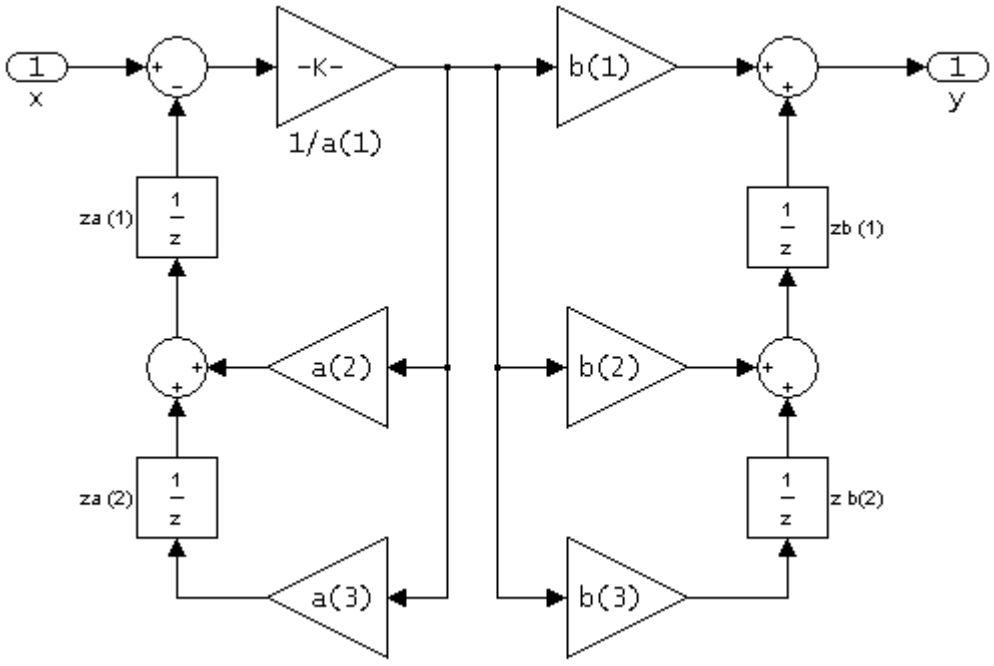
```
%forma directa I sos
[b,a] = ellip(6,1,60,.4);  % coeficientes del filtro
% s: coeficientes   g: ganancia de la sección
[s,g] = tf2sos(b,a);       % Convierte a SOS
s =
```

	Numeradores			Denominadores			
	1.0000	1.6355	1.0000	1.0000	-1.2134	0.4706	Sección 1
	1.0000	0.4551	1.0000	1.0000	-0.8266	0.7254	Sección 2
	1.0000	-0.0023	1.0000	1.0000	-0.5982	0.9248	Sección 3

```
g =

    0.0153

Hd=dfilt.df1sos(s,g);
get(Hd,'sosmatrix')

Hd =

         FilterStructure: 'Direct-Form I, Second-Order Sections'
               sosMatrix: [3x6 double]
             ScaleValues: [0.0153271894258813;1;1;1]
        PersistentMemory: false

fvtool(Hd)
```



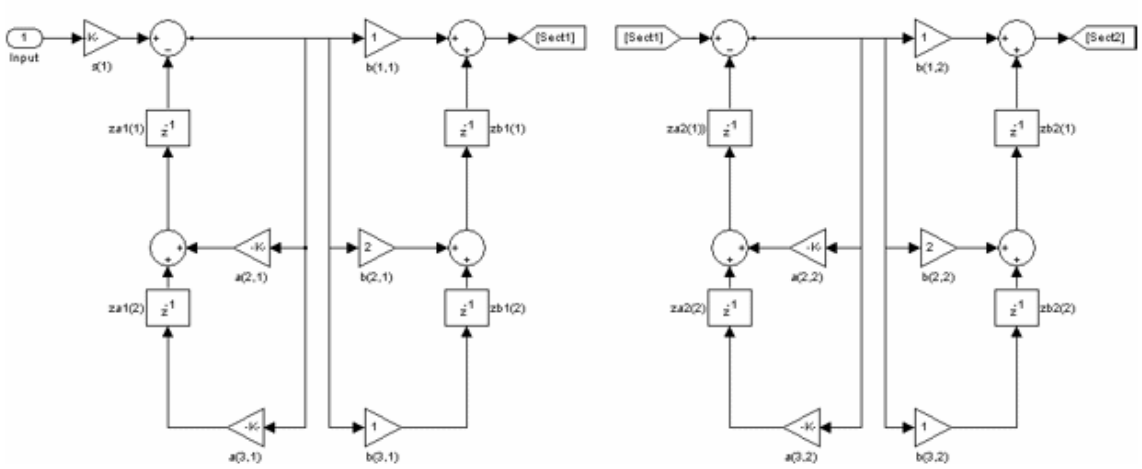
c) Forma Directa I Transpuesta: dfilt.df1t



Ejemplo:

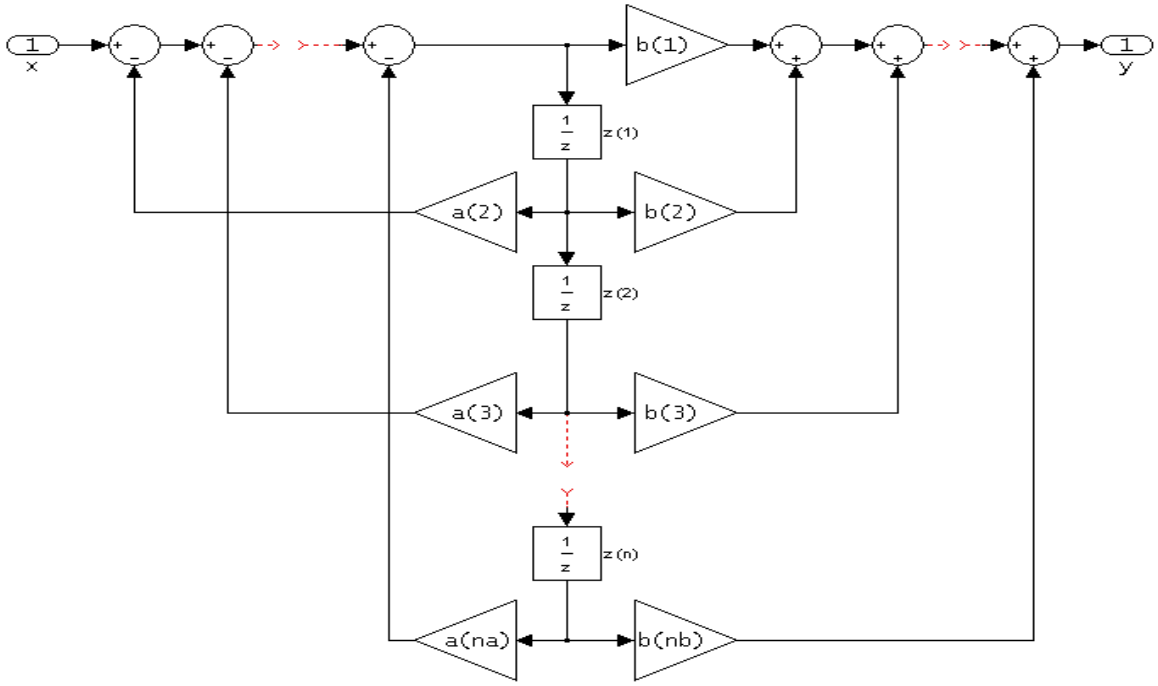
```
[b,a] = butter(4,.5);
Hd = dfilt.df1t(b,a);
num=get(Hd,'Numerator')
%num = 0.0940    0.3759    0.5639    0.3759    0.0940
den=get(Hd,'Denominator')
%den =  1.0000   -0.0000    0.4860   -0.0000    0.0177
fvtool(b,a)
```

d) Forma Directa I Transpuesta sos: dfilt.df1tsos




```
%forma directa I sos
[b,a] = ellip(6,1,60,.4);  % coeficientes del filtro
% s: coeficientes   g: ganancia de la sección
[s,g] = tf2sos(b,a);       % Convierte a SOS
Hd=dfilt.df1tsos(s,g);
get(Hd,'sosmatrix')
```

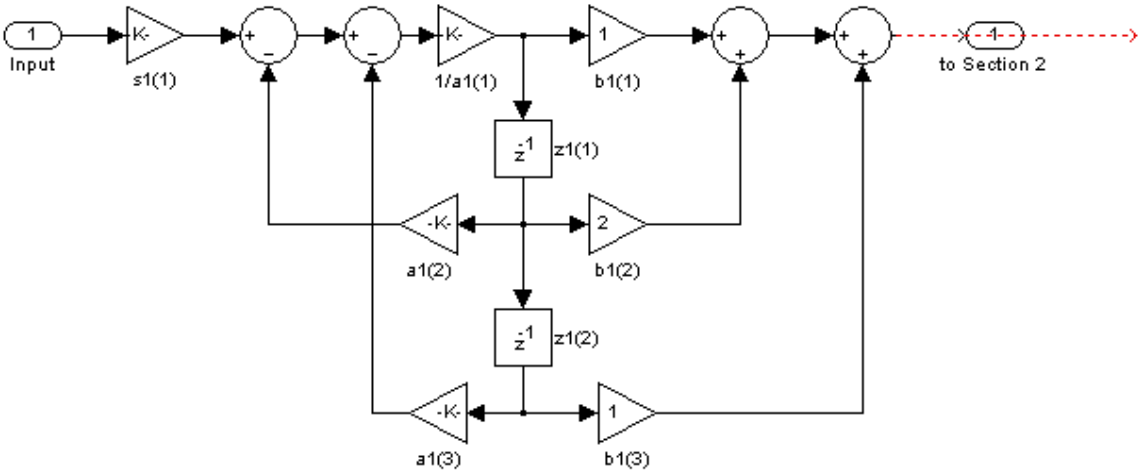
e) Forma Directa II: dfilt.df2



Ejemplo:

```
[b,a] = butter(4,.5);
Hd = dfilt.df2(b,a)
coeffs(Hd)
realizemdl(Hd)
```

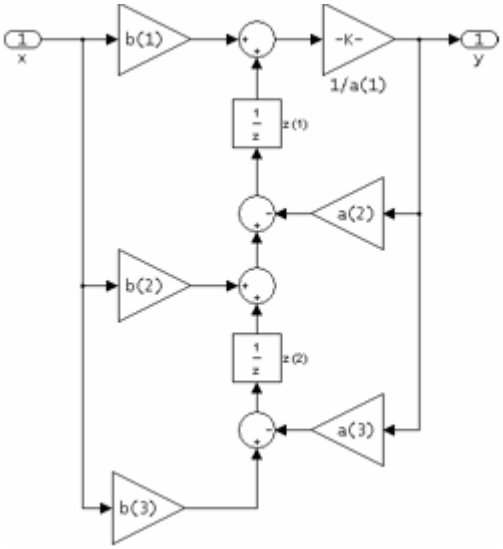
f) Forma Directa II sos: dfilt.df2sos



Ejemplo:

```
[z,p,k] = ellip(6,1,60,.4);
[s,g] = zp2sos(z,p,k);
Hd = dfilt.df2sos(s,g)
```

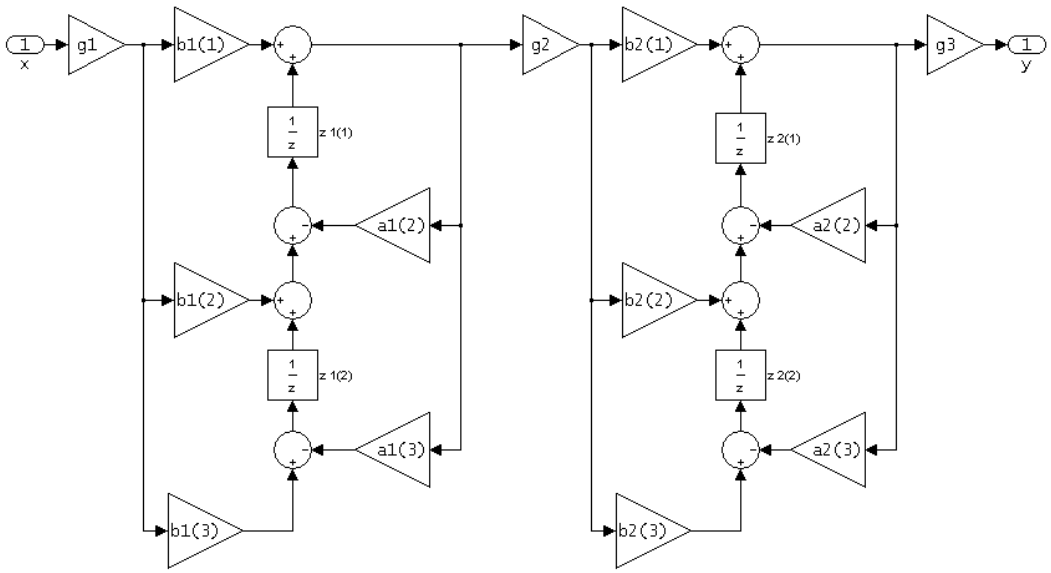
g) Forma Directa II Transpuesta: dfilt.df2t



Ejemplo:

```
[b,a] = butter(4,.5);
Hd = dfilt.df2t(b,a)
```

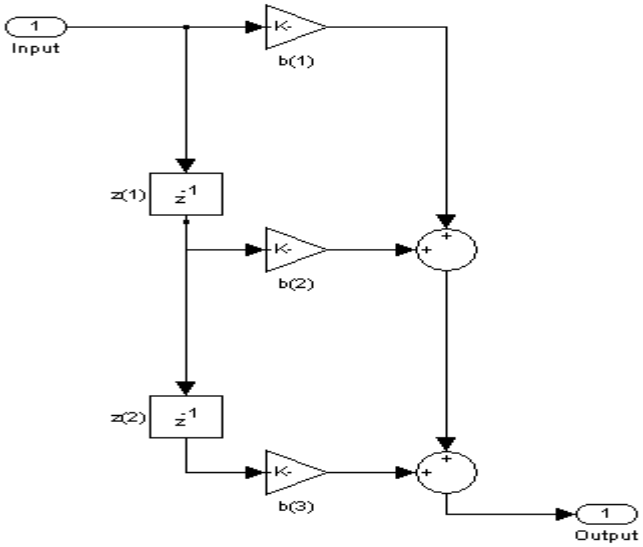
h) Forma Directa II Transpuesta sos: dfilt.df2tsos




```
[z,p,k] = ellip(6,1,60,.4);
[s,g] = zp2sos(z,p,k);
Hd = dfilt.df2tsos(s,g)
```

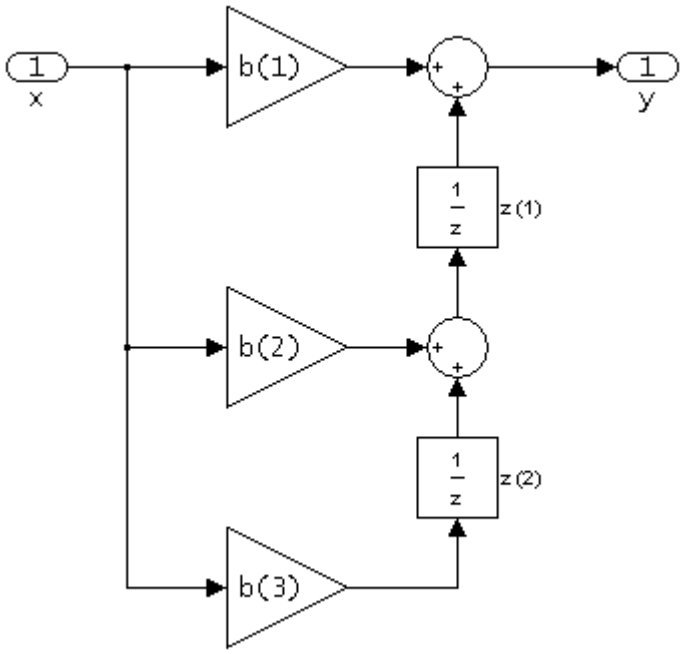
2. IMPLEMENTACIÓN DE FILTROS FIR

a) Forma directa FIR: dfilt.dffir



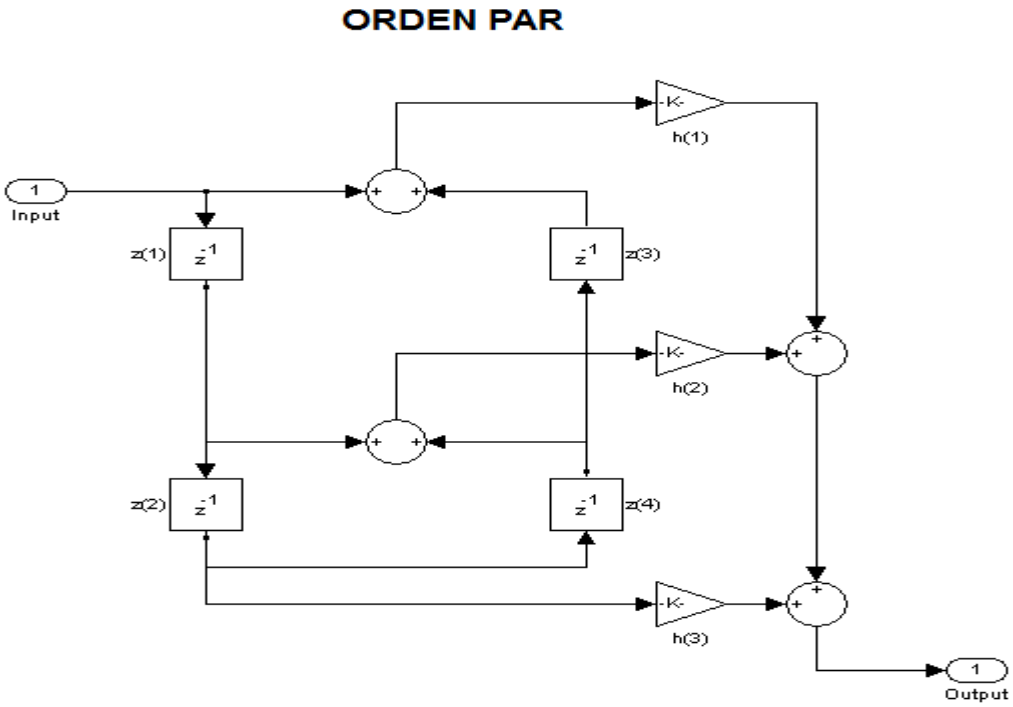

```
b = firpm(30,[0 .1 .2 .5]*2,[1 1 0 0]);
Hd = dfilt.dffir(b)
fvtool(Hd)
```

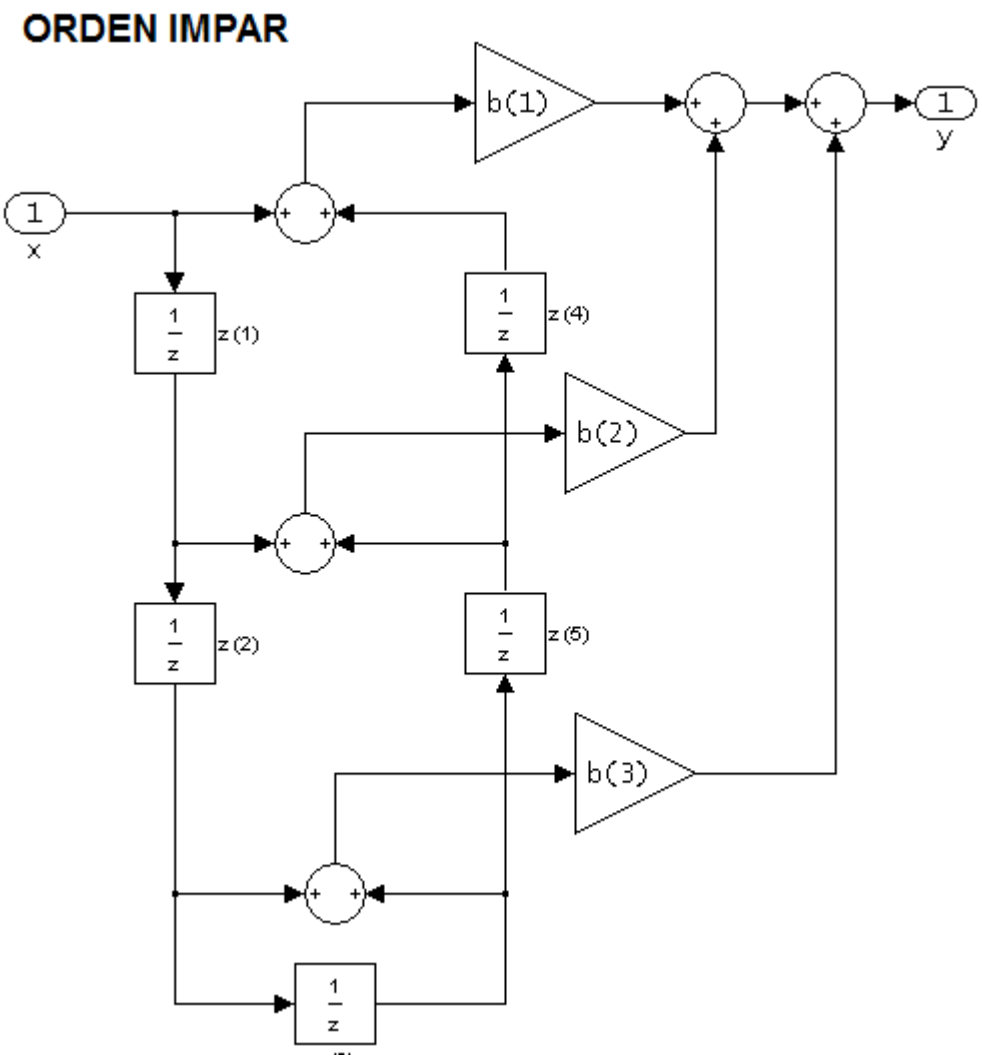
b) Forma directa transpuesta FIR: dfilt.dffirt




```
b = firpm(30,[0 .1 .2 .5]*2,[1 1 0 0]);
Hd = dfilt.dffirt(b)
```

c) Forma directa simétrica FIR: dfilt.dfsymfir





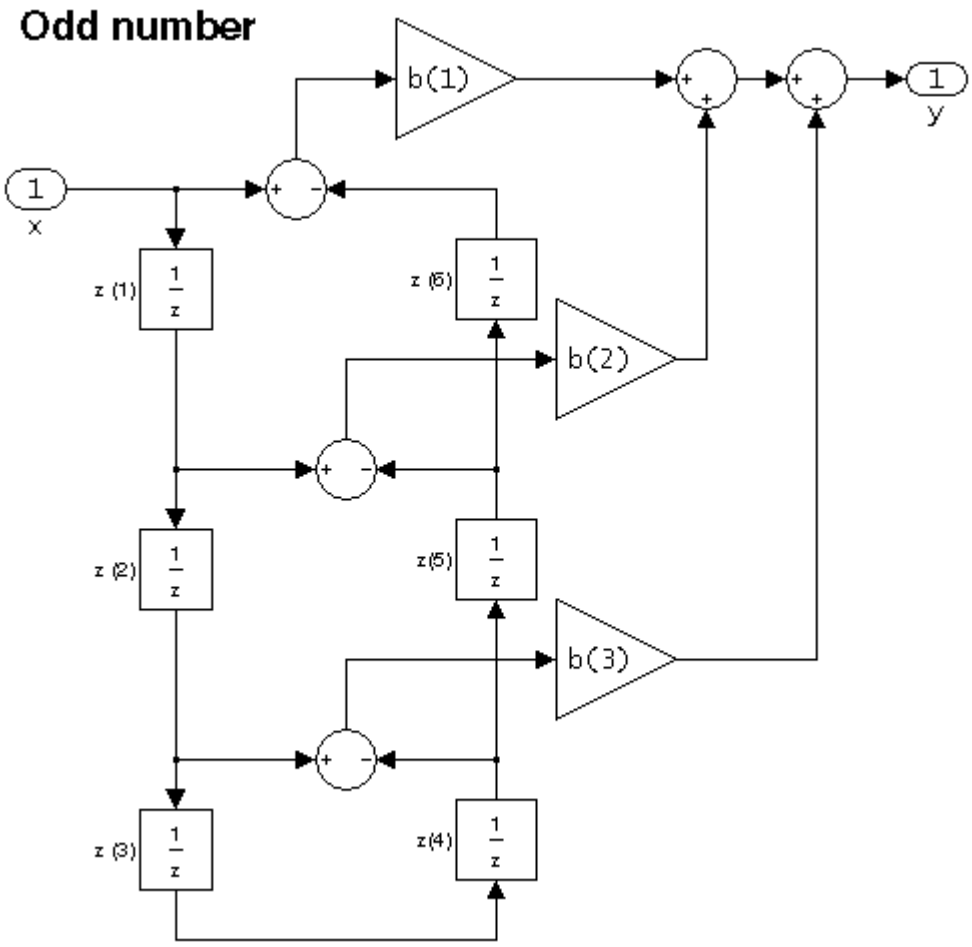
Ejemplo: Orden Par

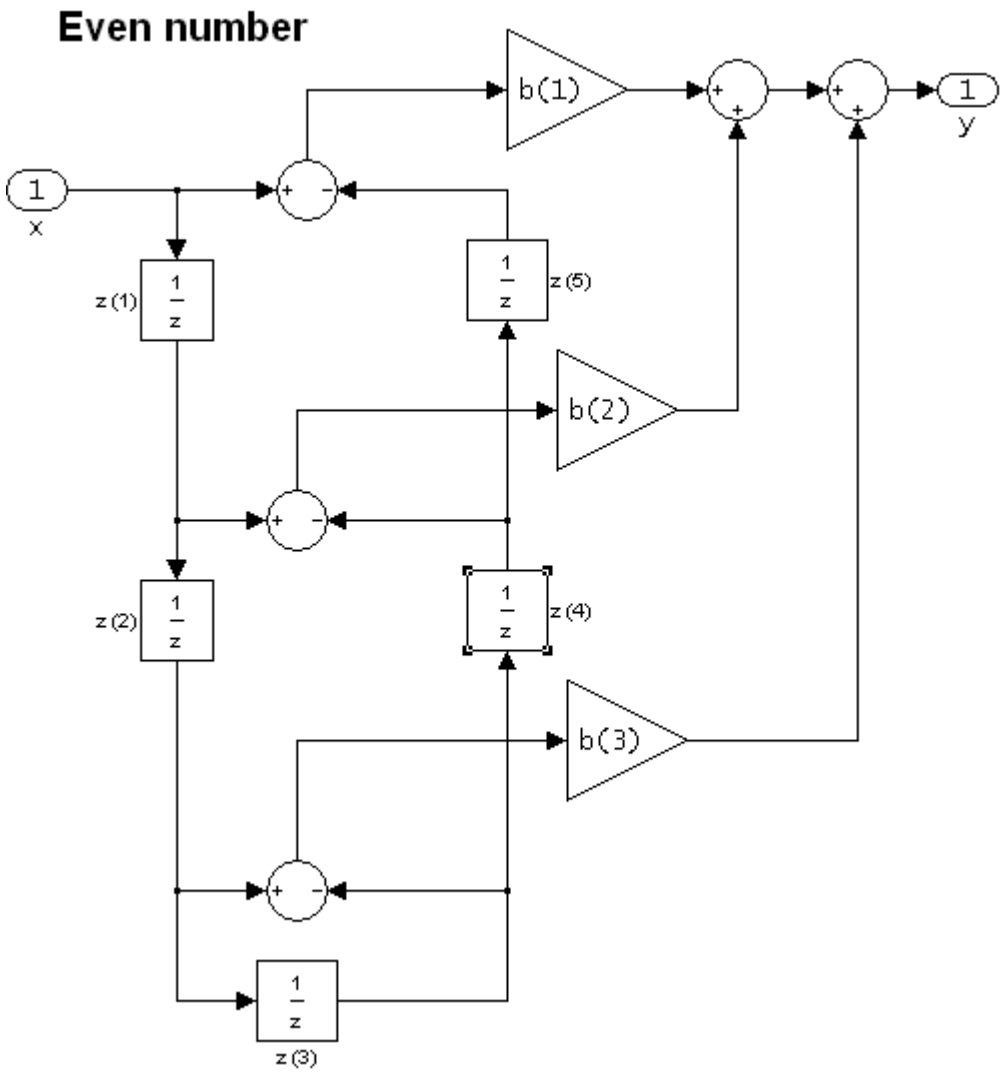
```
b = [-0.008 0.06 0.44 0.44 0.06 -0.008];
Hd = dfilt.dfsymfir(b)
fvtool(Hd)
```

Ejemplo: Orden Impar

```
b = [-0.01 0.1 0.8 0.1 -0.01];
Hd = dfilt.dfsymfir(b)
fvtool(Hd)
```

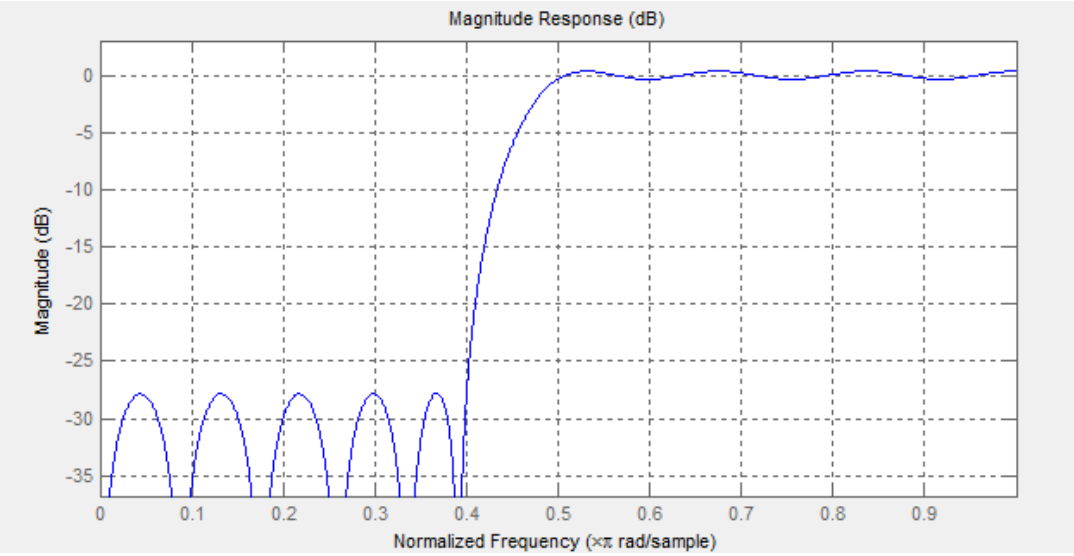
d) Forma directa antisimétrica FIR: dfilt.dfasymfir

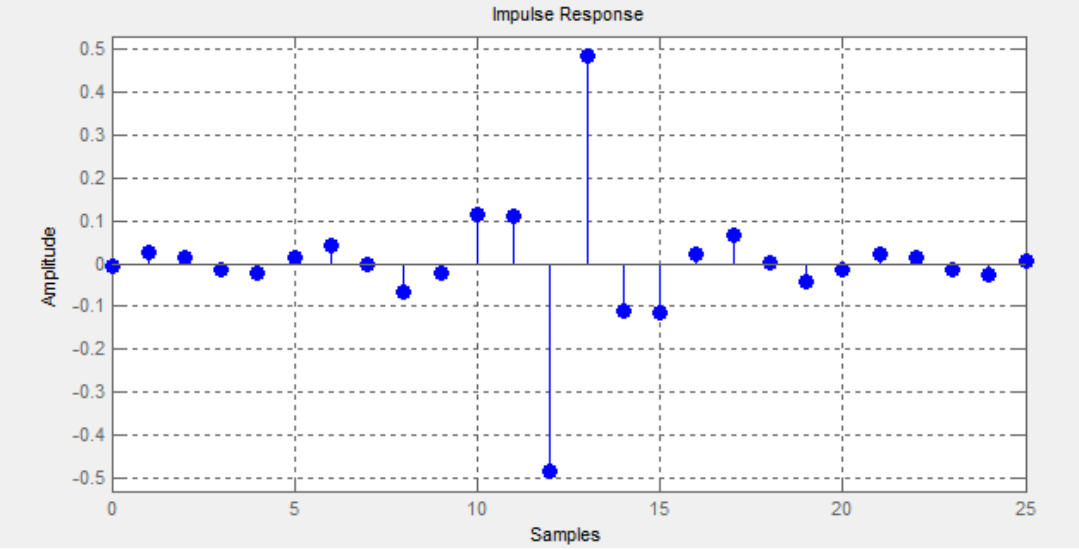




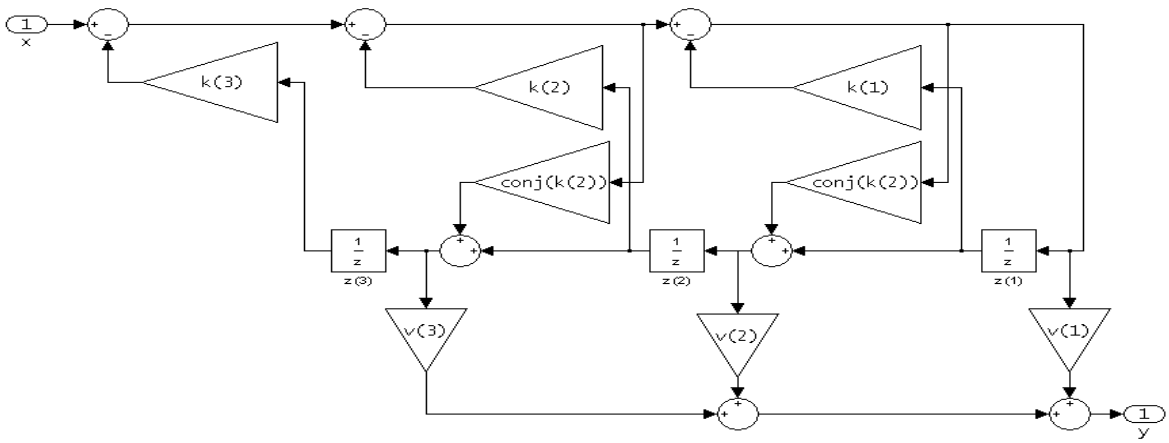
Ejemplo:

```
%orden impar tipo IV
Hd = firpm(25,[0 .4 .5 1],[0 0 1 1],'h');
fvtool(Hd)
%orden par tipo III
h=firpm(44,[0 .3 .4 1],[0 .2 0 0],'differentiator');
fvtool(Hd)
```





e) Forma lattice ARMA (autoregressive, moving-average): dfilt.latticearma



k: coeficientes lattice, v: coeficientes ladder

Ejemplo:

```
k = [.66 .7 .44];
Hd = dfilt.latticearma(k)
```

```
Hd =

        FilterStructure: 'Lattice Autoregressive Moving-Average (ARMA)'
                Lattice: [0.66 0.7 0.44]
                 Ladder: 1
       PersistentMemory: false
```

```
fvtool(Hd)
```

```
realizemdl(Hd)
```