PROCESAMIENTO DIGITAL DE SEÑALES

PROFESOR: ING. JORGE A. POLANÍA P.

FILTROS DIGITALES – FILTROS FIR
(Respuesta Impulsional Finita)

1. INTRODUCCIÓN

Los filtros FIR tienen las siguientes ventajas:

- Tienen fase lineal
- Son siempre estables
- Eficientes realizaciones en hardware
- Respuesta al impulso 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.

Función de transferencia:

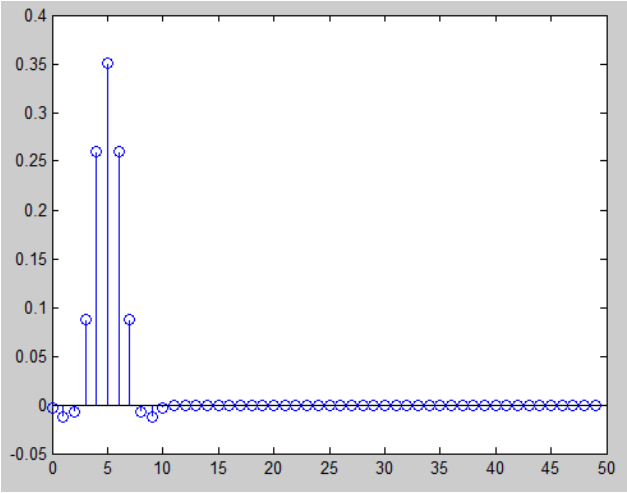
$$H(z) = B(z) = b(1) + b(2)z^{-1} + \cdots + b(M)z^{-M}$$

Ecuación en diferencias:

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

Respuesta impulsional:

```
t=0:49;
imp=[1;zeros(49,1)];
b=fir1(10,0.35);
y=filter(b,1,imp);
stem(t,y)
```



Respuesta en frecuencia:

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

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

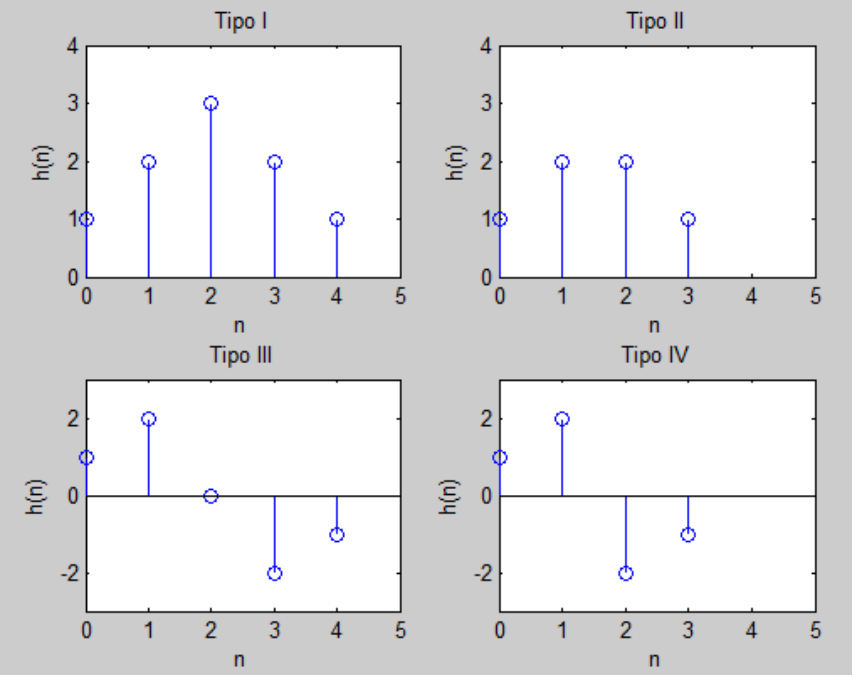
Tipos de filtros:

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

$$\phi(w) = k_1 + k_2 w \quad (Fase\ lineal)$$

$$Si \quad k_1 = 0, \qquad se\ tiene\ simetría\ par,$$
$$Si \quad k_1 = \pi/2 \quad se\ tiene\ simetría\ impar$$

Tipo I: Longitud N par y simetría par
Tipo II: Longitud N impar y simetría par
Tipo III: Longitud N par y simetría impar (antisimétrico)
Tipo IV: Longitud N impar y simetría impar (antisimétrico)



2. DISEÑO DE FILTRO FIR

Se tienen los siguientes métodos de diseño:

- Método de ventaneo
- Filtros simétricos
- Filtros antisimétricos
- Mínimo cuadrado restringido

a) MÉTODO DE VENTANEO

La respuesta de un filtro FIR ES DE FASE LINEAL. Su forma de implementar es truncando la respuesta de un filtro IIR ideal, $h_d(n)$.

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

Si W(n) es una ventana, esto es,

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

Entonces,

$$h(n) = h_d(n)W(n)$$

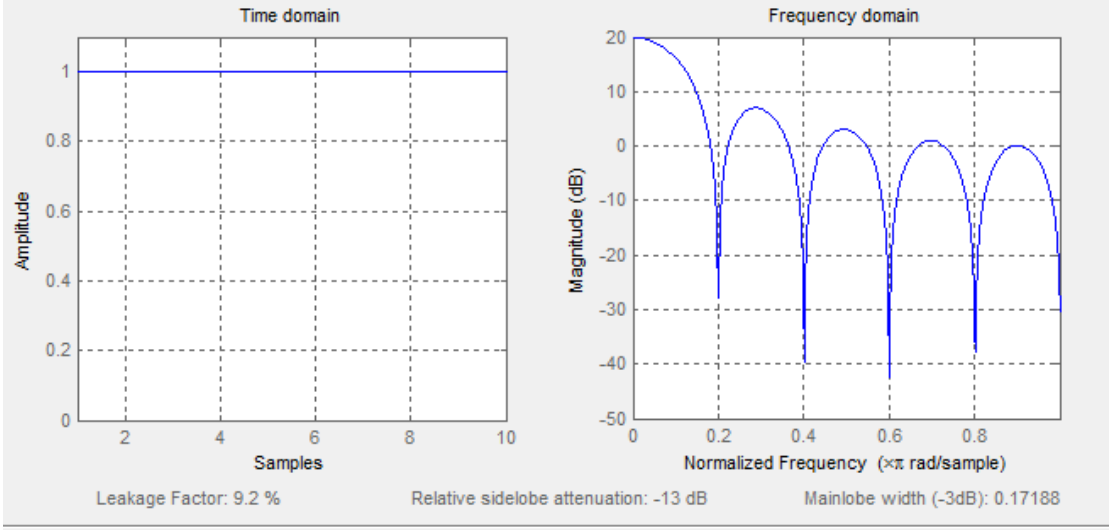
La multiplicación de estas dos funciones en el dominio del tiempo corresponde a la convolución de sus Transformadas de Fourier en frecuencia,

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

Ventana Rectangular

MATLAB

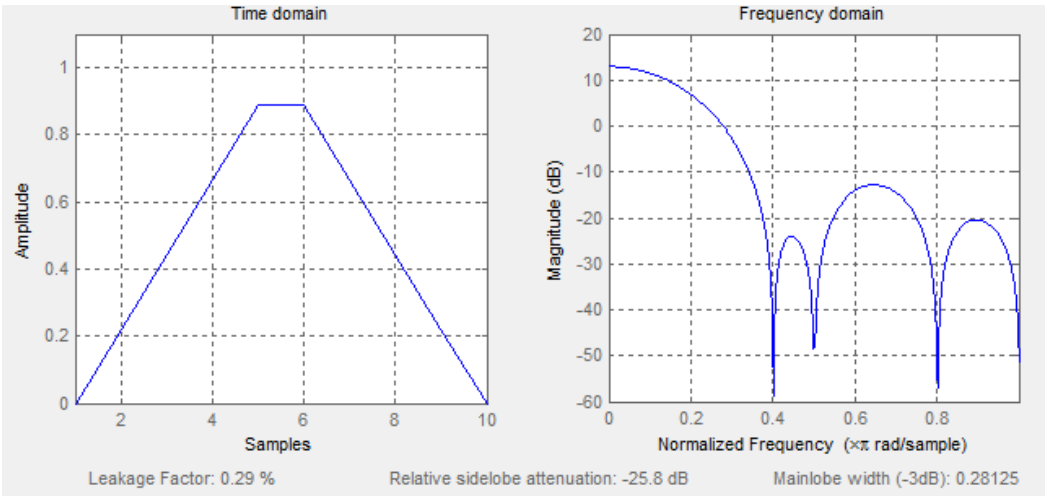
```
% Ventana rectangular

w=rectwin(10);
wvtool(w)
```



Hay varios algoritmos de ventanas que permiten minimizar el riple del lóbulo principal y/o l0s lóbulos laterales: Rectangular, Bartlett, Hamming, Blackman, Hanning, Kaiser.

Ventana triangular de Bartlett: w=Bartlett(N)

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



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

w=bartlett(N)
w= blackman(N)
w=hammin(N)
w=han(N)

Ventana Kaiser:

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

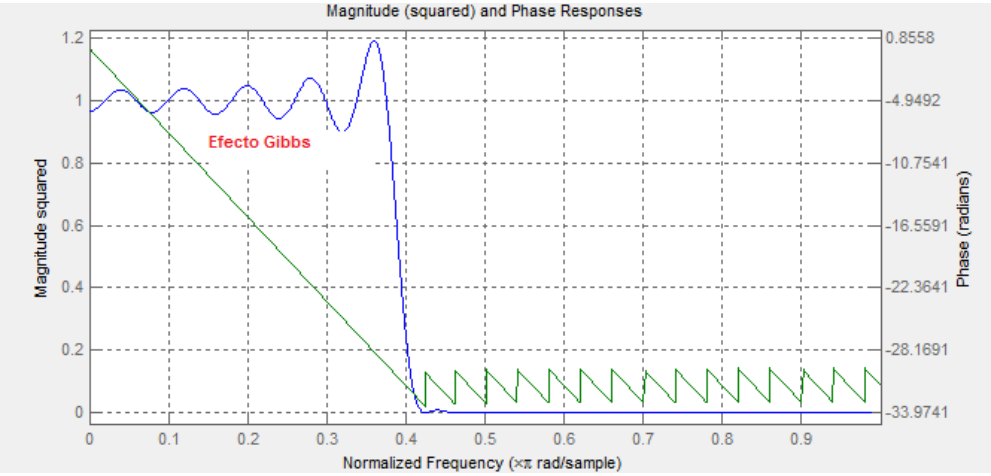
w=káiser(N) % β=50
w=káiser(N,β)
Al aumentar β disminuyen los lóbulos laterales

La ventana rectangular es la más simple pero presenta el fenómeno de Gibbs. La de Bartlett o triangular reduce el overshoot pero dispersa considerablemente la banda de transición. Las de Hanning, Hamming y Blackman proveen una truncación más 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.

Ejemplo:

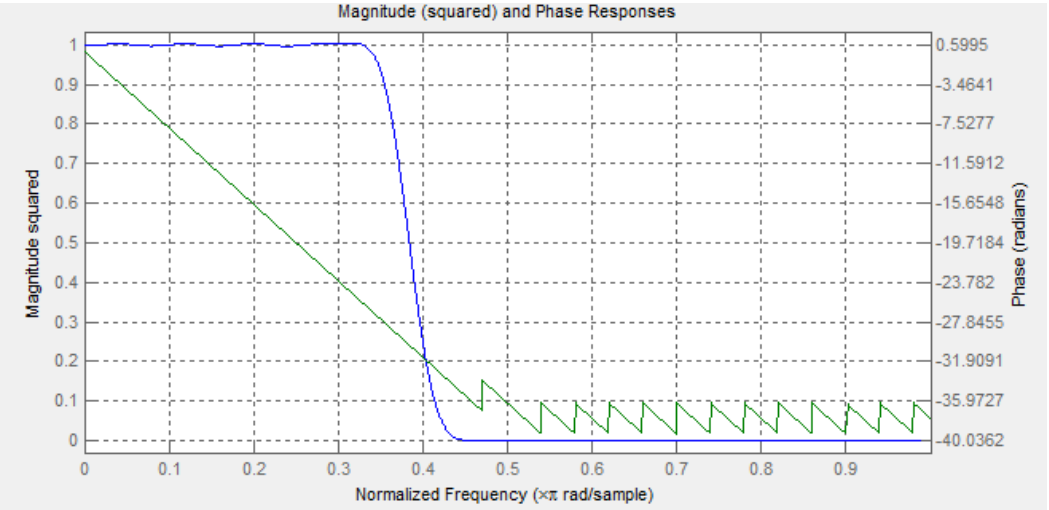
Filtro pasa bajo de orden 51 con frecuencia de corte normalizada de 0.4,

```
%longitud 51 ventana rectangular
b=fir1(50,0.4,rectwin(51));
fvtool(b,1)
```



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

```
%longitud 51 ventana Hamming
b=fir1(50,0.4,hamming(51));
fvtool(b,1)
```



Funciones en Matlab para método de ventanas

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.

USO DE LA FUNCIÓN: fir1

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

Ejemplo:

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

En general, para obtener los coeficientes del filtro,

b = fir1(n,Wn,'Tipo',ventana)

- 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

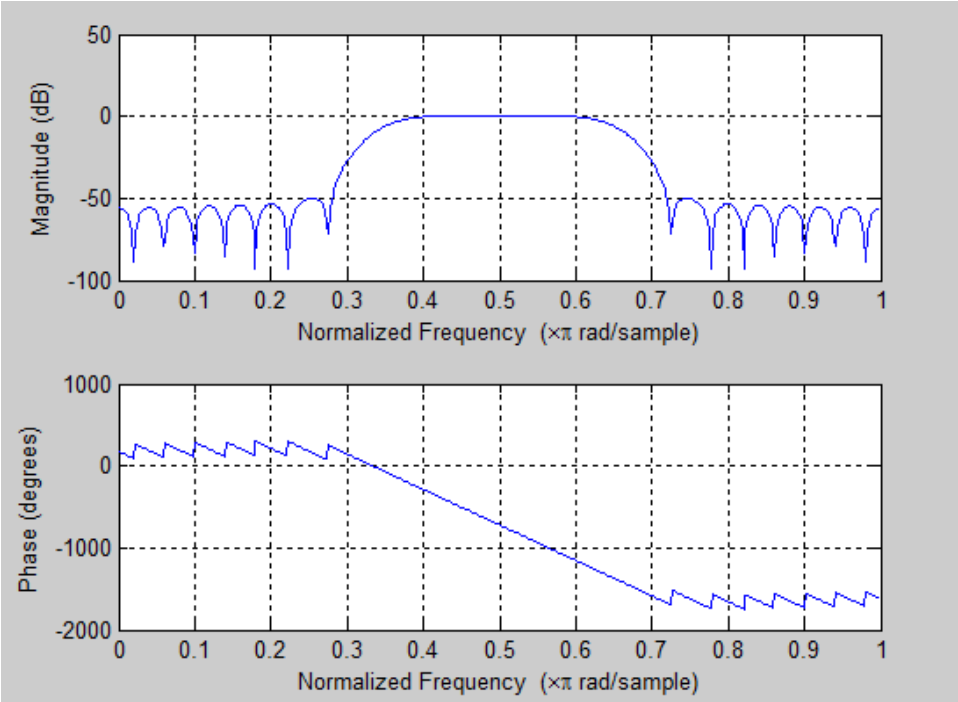
El tipo de filtro se programa con Tipo:

- '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

Ejemplo

Diseñe un filtro FIR pasa banda de orden 48 con pasa banda de 0.35 ≤ w ≤ 0.65, ventana Hamming.

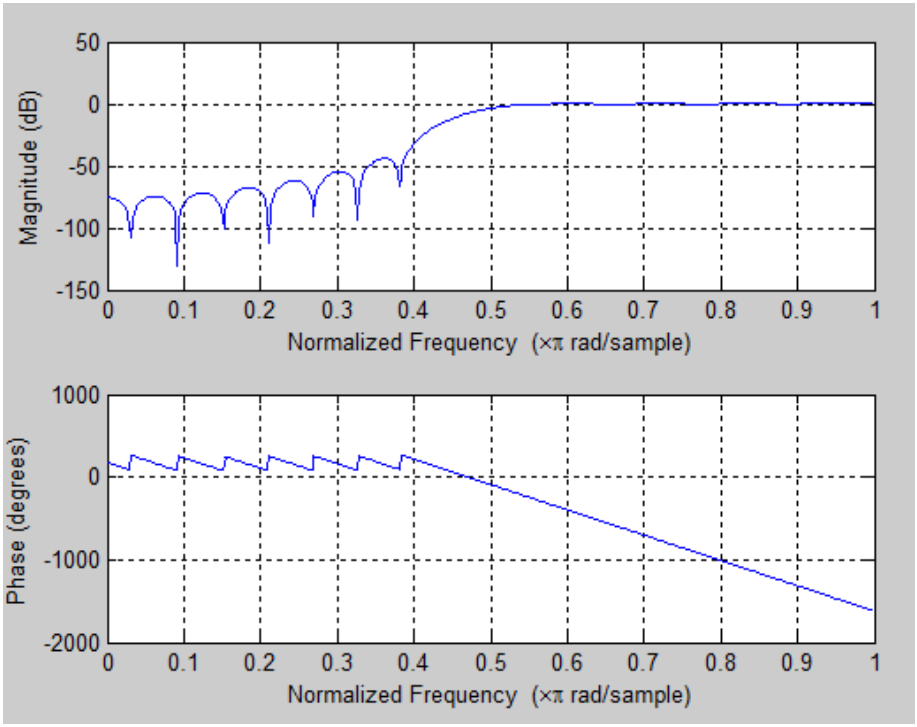
```
% uso de fir1
b = fir1(48,[0.35 0.65]);
freqz(b,1,512)
```



Ejemplo

Diseñe un filtro FIR pasa alto de orden 34 para que atenúe las frecuencias posteriores a fc = 0.48 y ventana Hanning.

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

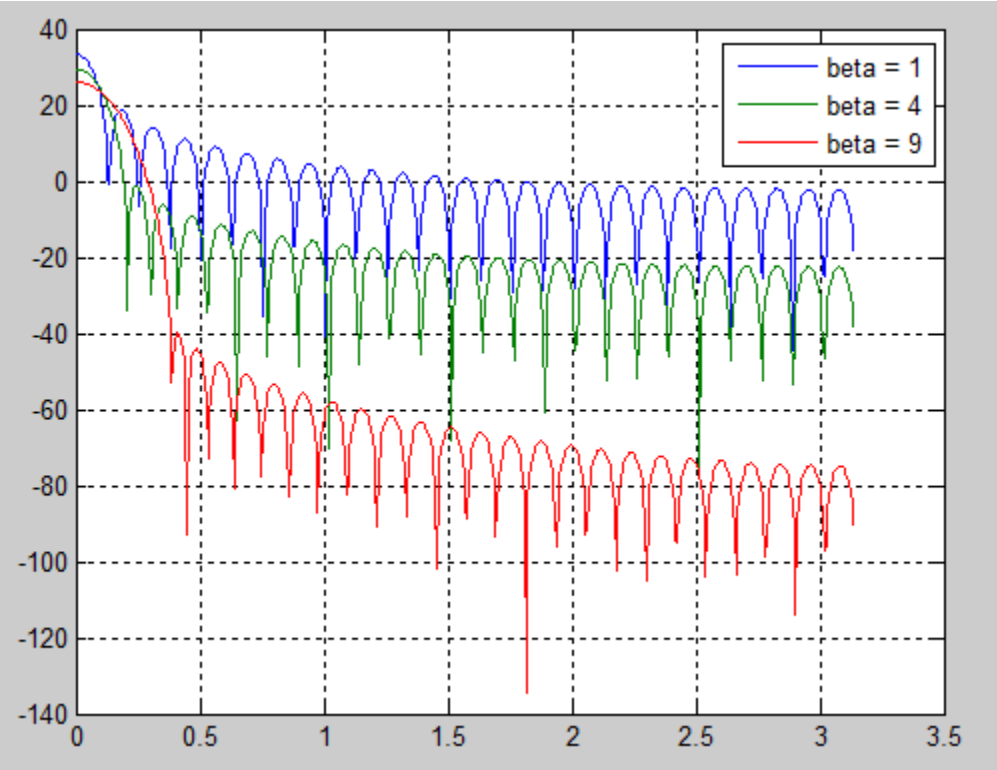


Ventana Kaiser

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

Ejemplo

```
n = 50;
w1 = kaiser(n,1);
w2 = kaiser(n,4);
w3 = kaiser(n,9);
[H1,f] = freqz(w1,1,512);
[H2,f] = freqz(w2,1,512);
[H3,f] = freqz(w3,1,512);
plot(f,20*log10(abs([H1 H2 H3])));
grid;
legend('beta = 1','beta = 4','beta = 9')
```



Para encontrar el orden del filtro se utiliza la función:

[n,wn,beta,Tipo] = 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:

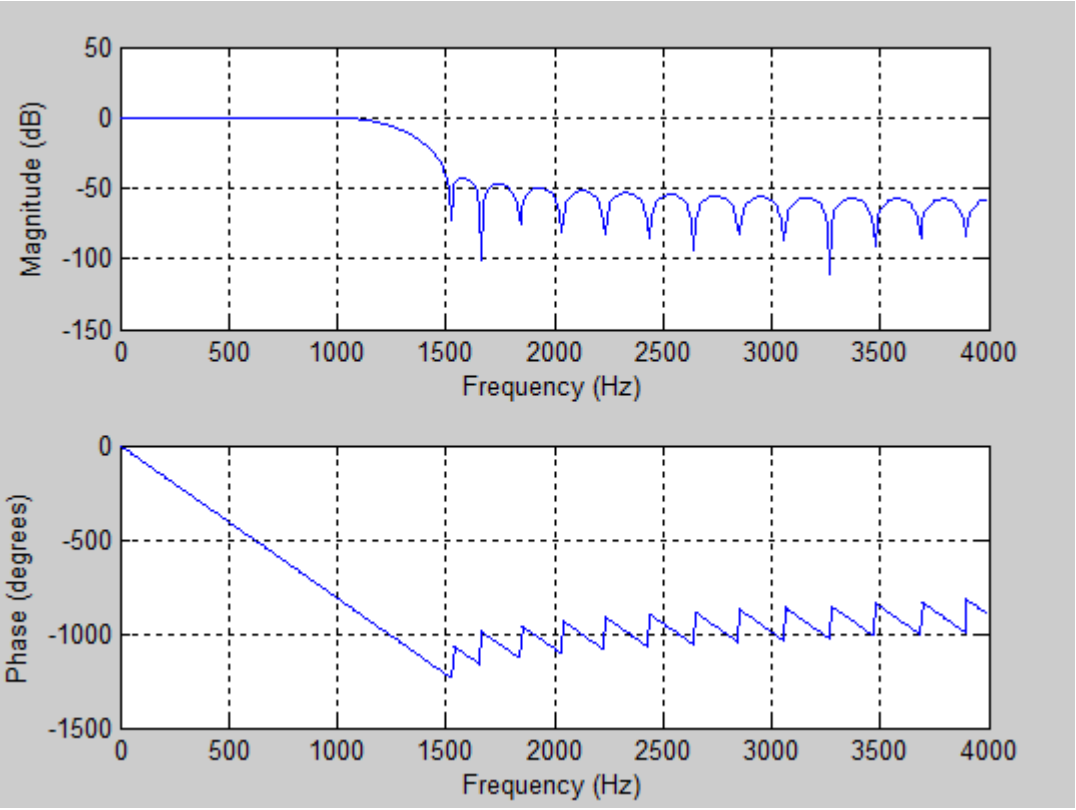
b = fir1(n,wn,kaiser(n+1,beta),Tipo,'noscale')

Tipo: es ‘high’ para pasa alto y ‘stop’ para banda stop. Para pasa banda puede ser ‘dc-1’ so la primera banda es banda stop o ‘dc-1’si es pasa banda.

Ejemplo:

Diseñe un filtro pasa bajo con banda de paso definida de 0 a 1000 Hz y banda stop de 1500 Hz a 4000 Hz El riple de la banda de paso es del 5% y la atenuación en la banda stop de 40 dB.

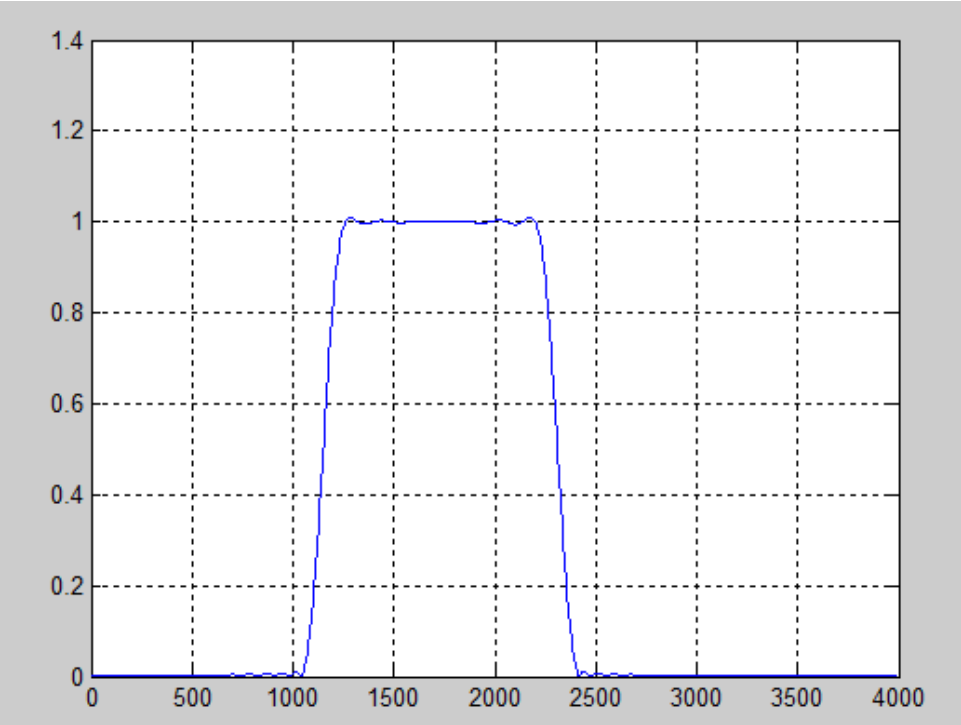
```
%orden del filtro kaiser
fs = 8000;
fc = [1000 1500];
mag = [1 0];
%5% =0.05, -40dB =20log(0.01)
dev = [0.05 0.01];
[n,wn,beta,Tipo] = kaiserord(fc,mag,dev,fs);
B = fir1(n,wn,Tipo,kaiser(n+1,beta),'noscale');
freqz(B,1,512,8000)
```



Ejemplo:

```
fs = 8000;
fc = [1000 1300 2210 2410];
mag = [0 1 0];
dev = [0.01 0.05 0.01];
[n,Wn,beta,tipo] = kaiserord(fc,mag,dev,fs);
b = fir1(n,Wn,tipo,kaiser(n+1,beta),'noscale');
```

```
[H,f] = freqz(b,1,1024,fs);
plot(f,abs(H)), grid on
%n = 90, beta = 3.3953, tipo = DC-0
```



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.

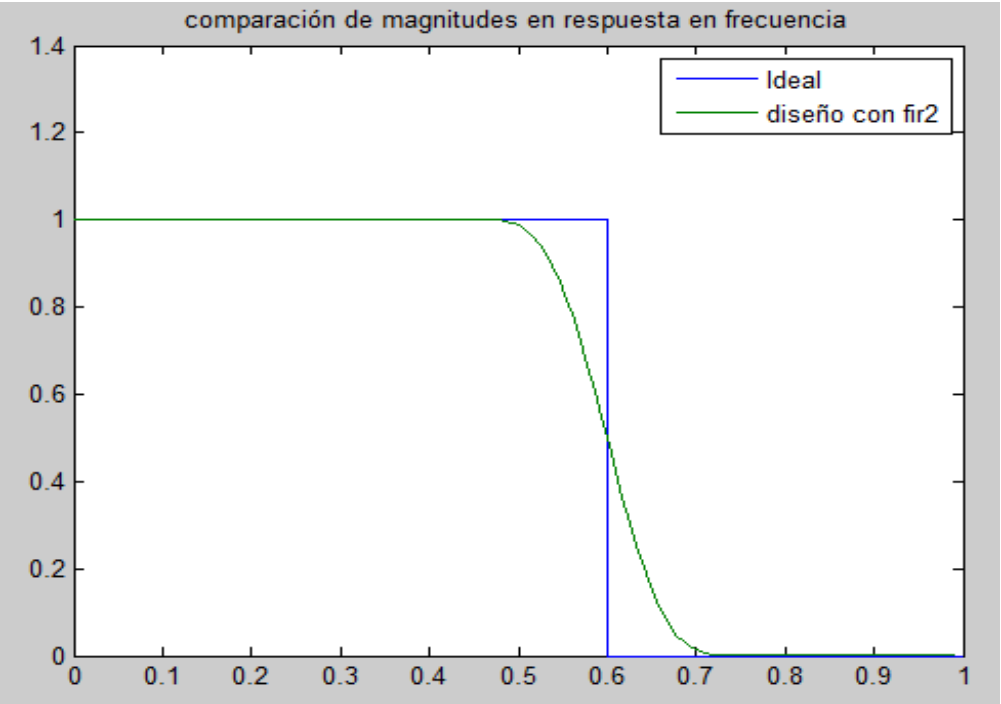
b = fir2(n,f,m,ventana)

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 dadas 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.

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

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

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')

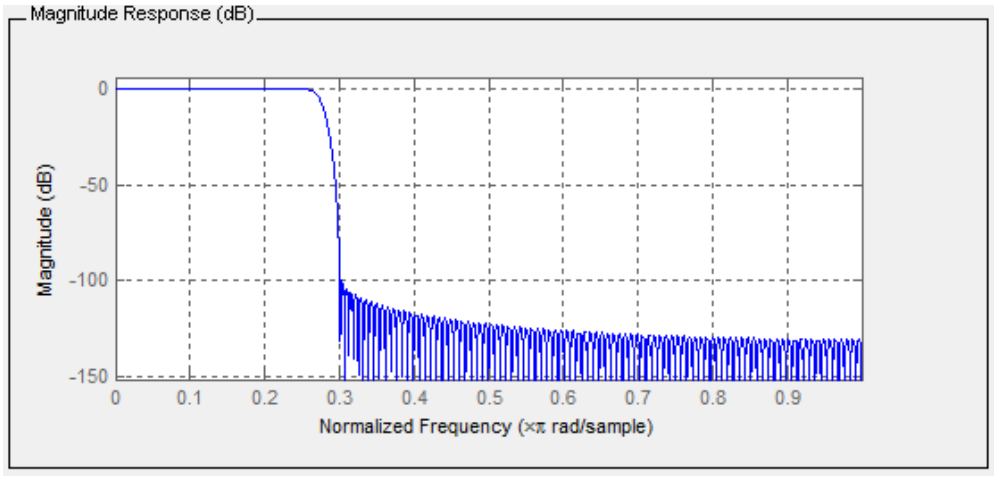
w: es el peso en la banda de frecuencia
ftype:
‘hilbert’ para filtros de fase lineal con simetría impar (tipo III y tipo IV)
b(k) = - b(n+2-k). Usa la Transformada de Hilbert
“differentiator’ usa técnica especial de pesos en las bandas

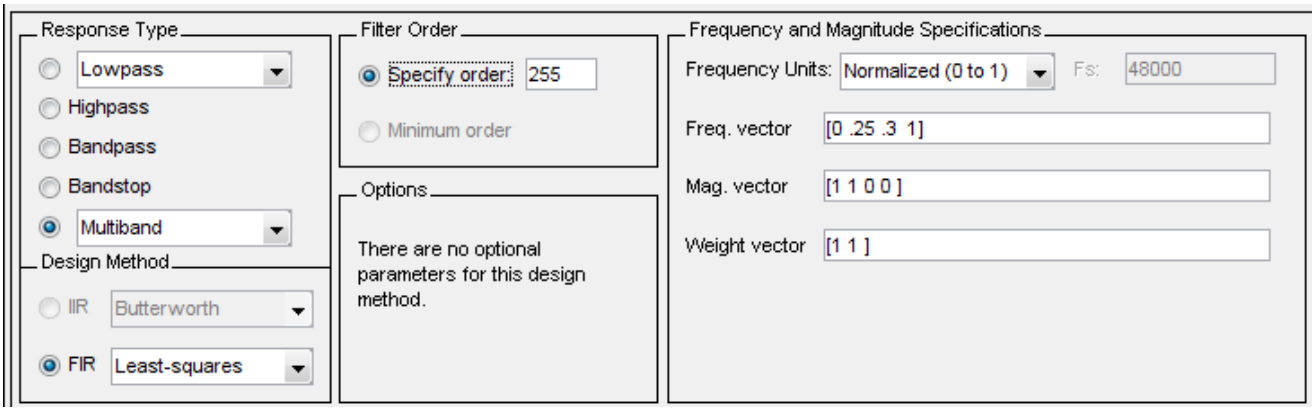
Ejemplo:

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]);
```

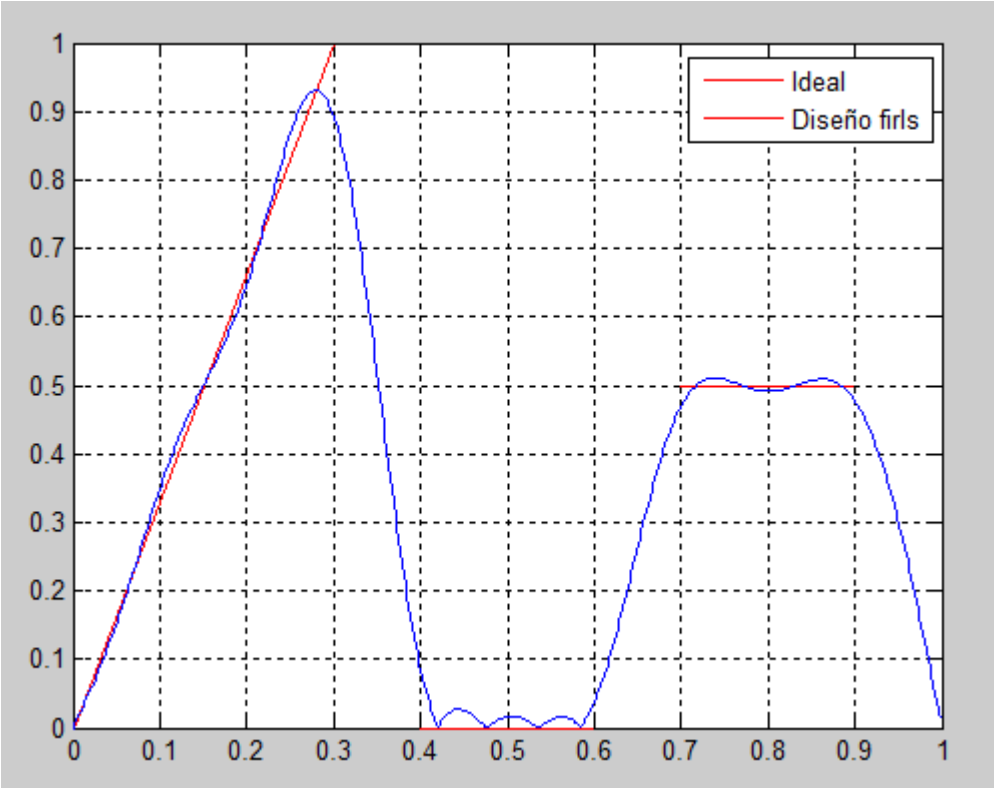
Usando fdatool



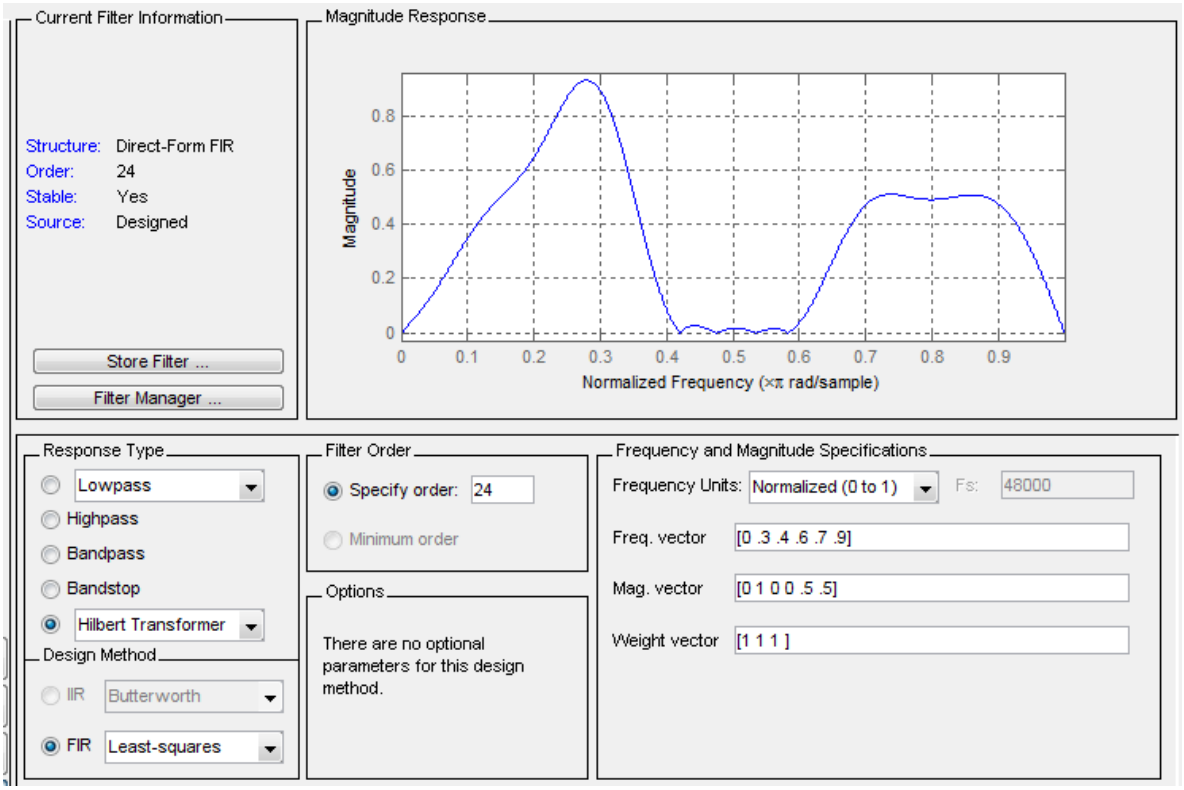


Diseñar un filtro antisimétrico de orden 24

```
%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:

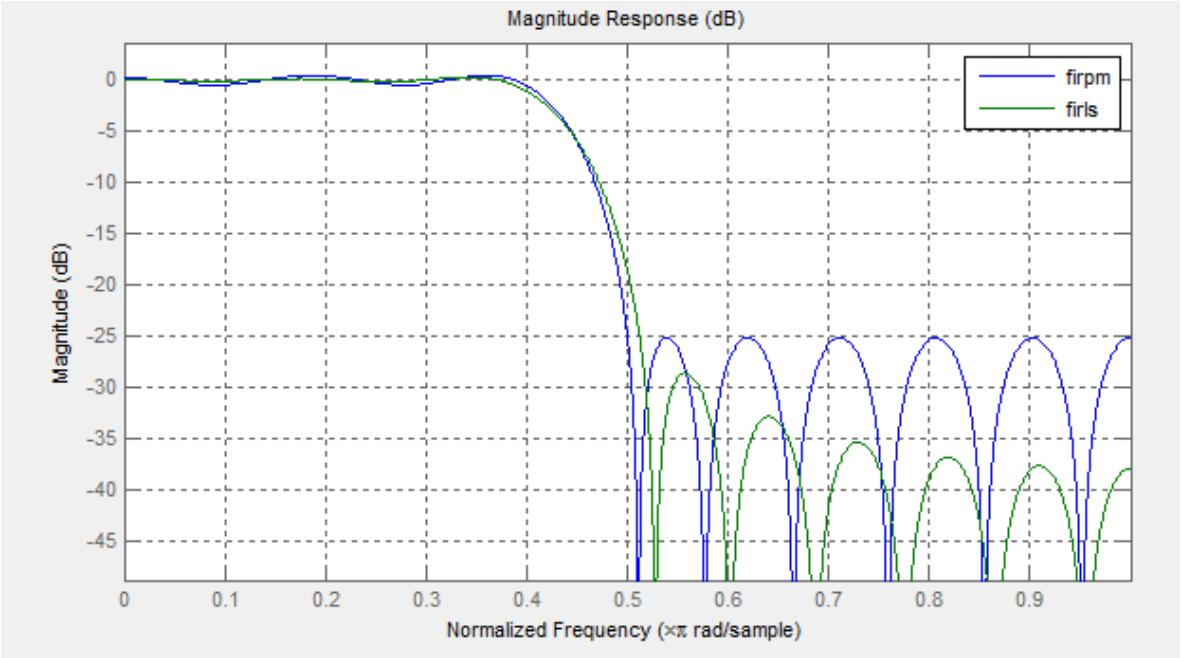


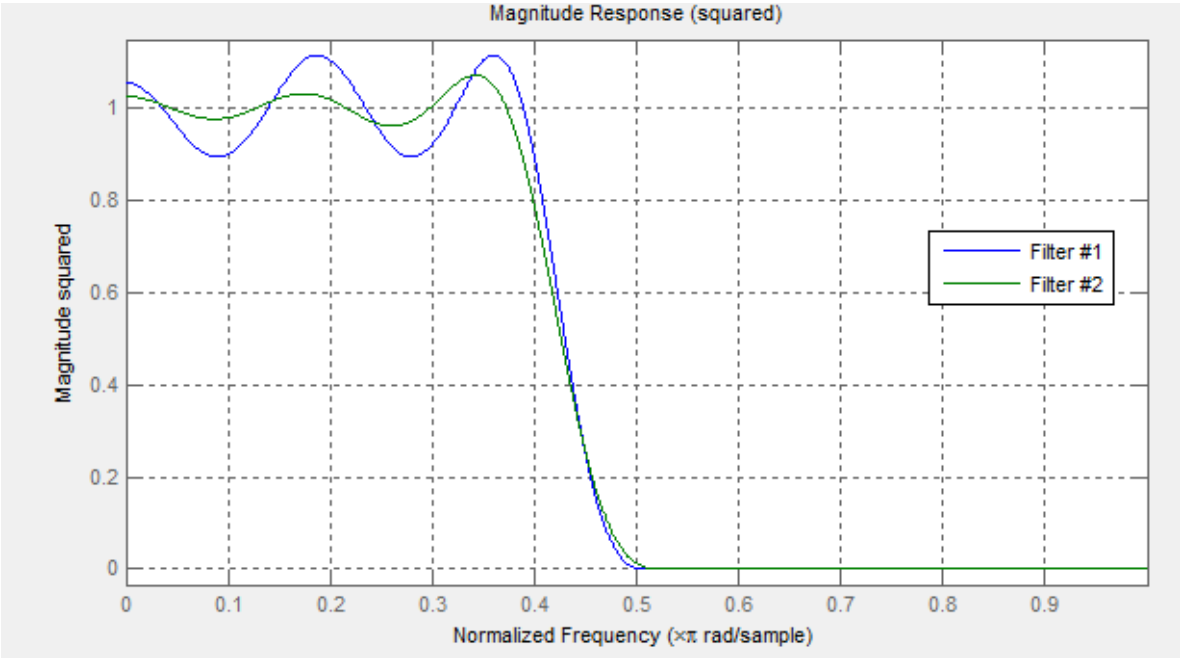
En resumen esta function diseña filtros de fase lineal tipo I, II, III, IV. Los tipo I y II son por defecto para n par e impar respectivamente. Los 'hilbert' y 'differentiator' producen tipo III (n par) y IV (n impar)

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.

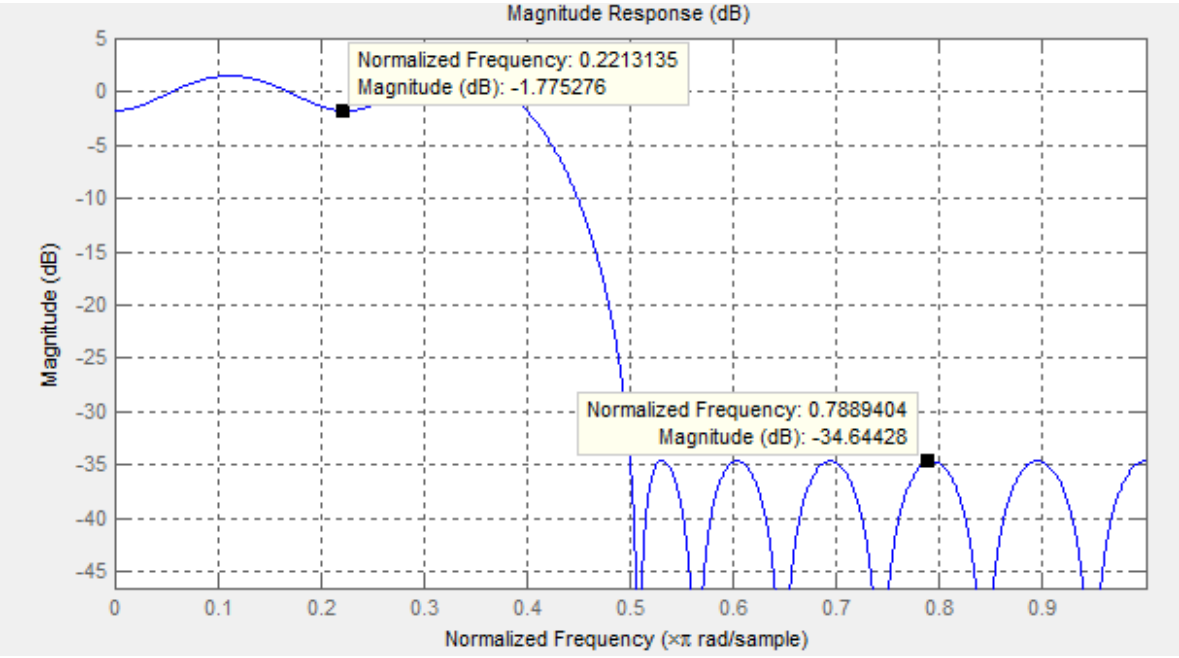
```
%tipos de filtros con firls o firpm
%pasa banda
f = [0 0.3  0.4  0.7  0.8  1];
```

```
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:

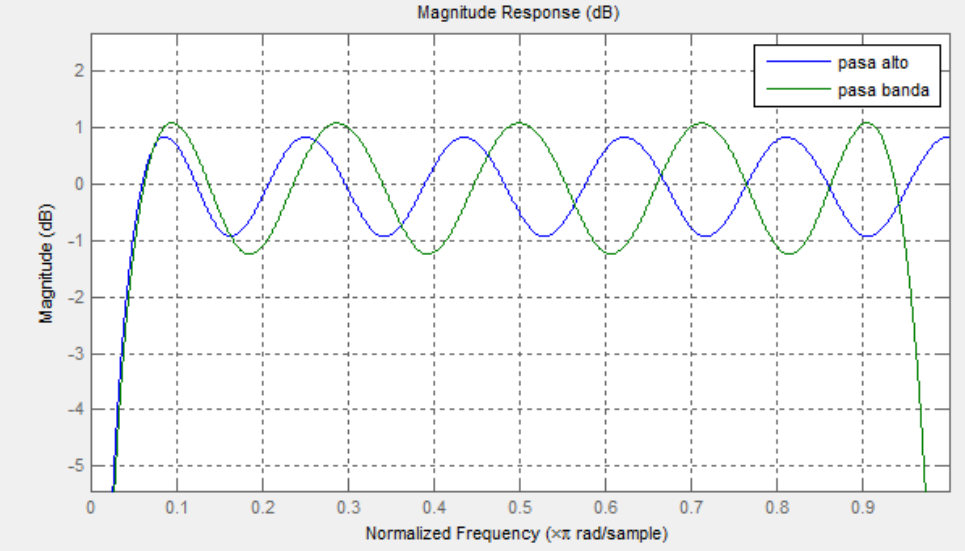
```
%uso de pesos en las bandas
n = 20;                % orden del fitro
f = [0 0.4 0.5 1];     % bordes de la bandas de frecuencias
a = [1   1    0   0];    % amplitudes deseadas
w = [1 10];            % vector peso
b = firpm(n,f,a,w);
fvtool(b,1)
```



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).

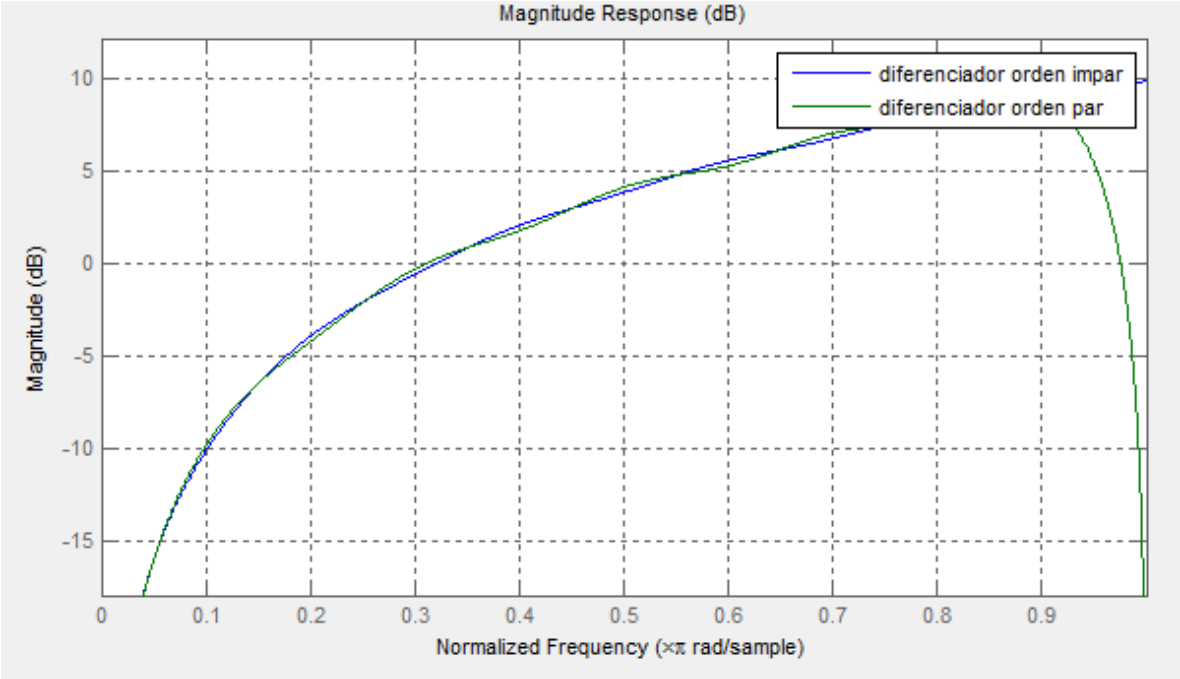
```
%señal analítica
fs = 1000;              % frecuencia de muestreo
t = (0:1/fs:2)';        % vector tiempo
x = sin(2*pi*300*t);    % señal senoidal de 300 Hz
xh = filter(bb,1,x);    % Transformada Hilbert de x
xd = [zeros(10,1); x(1:length(x)-10)];      % 10 muestras de
retardo
xa = xd + j*xh;   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

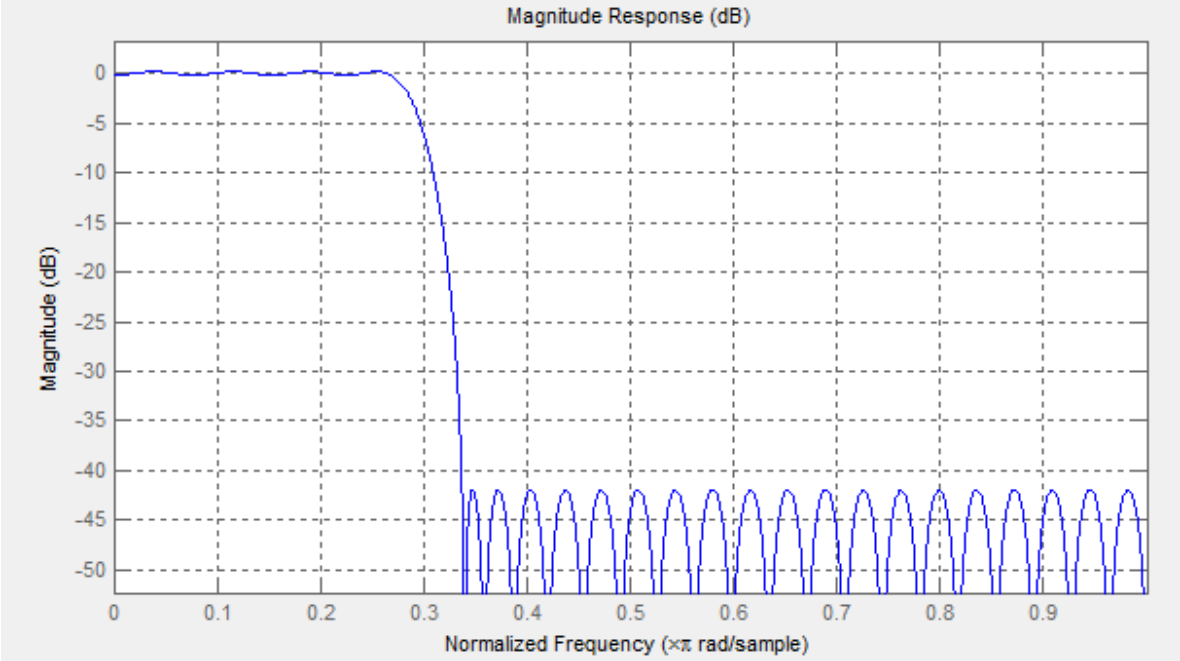
Función fircls1

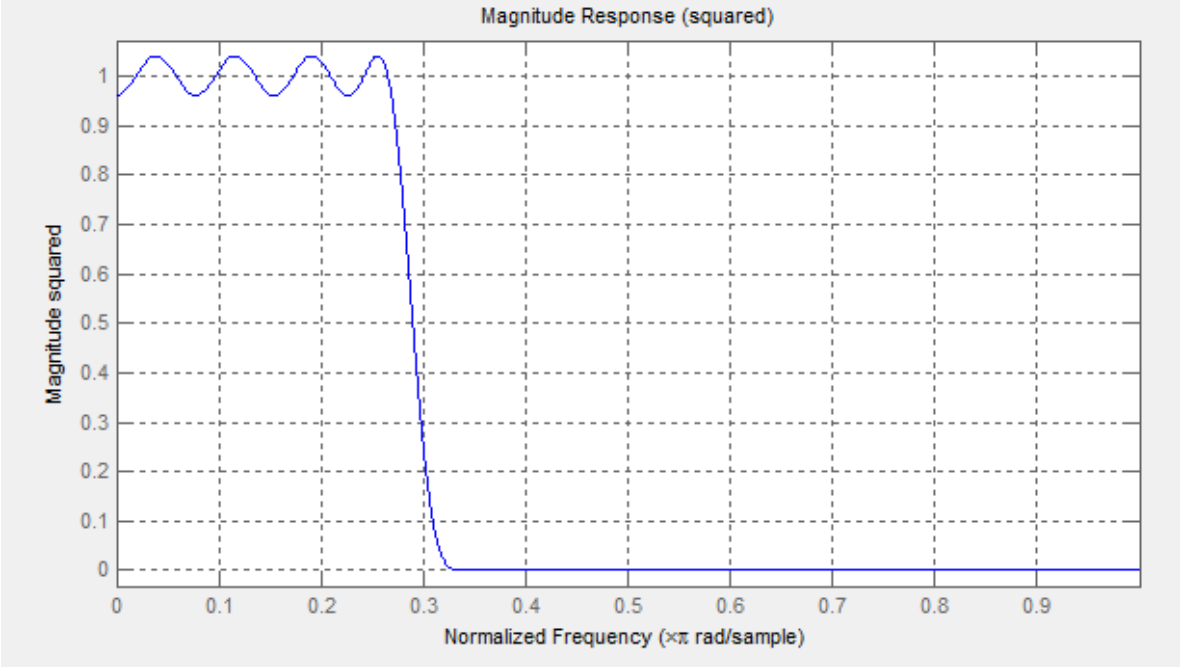
Se usa para el diseño de pasa bajo o pasa alto.

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)
```



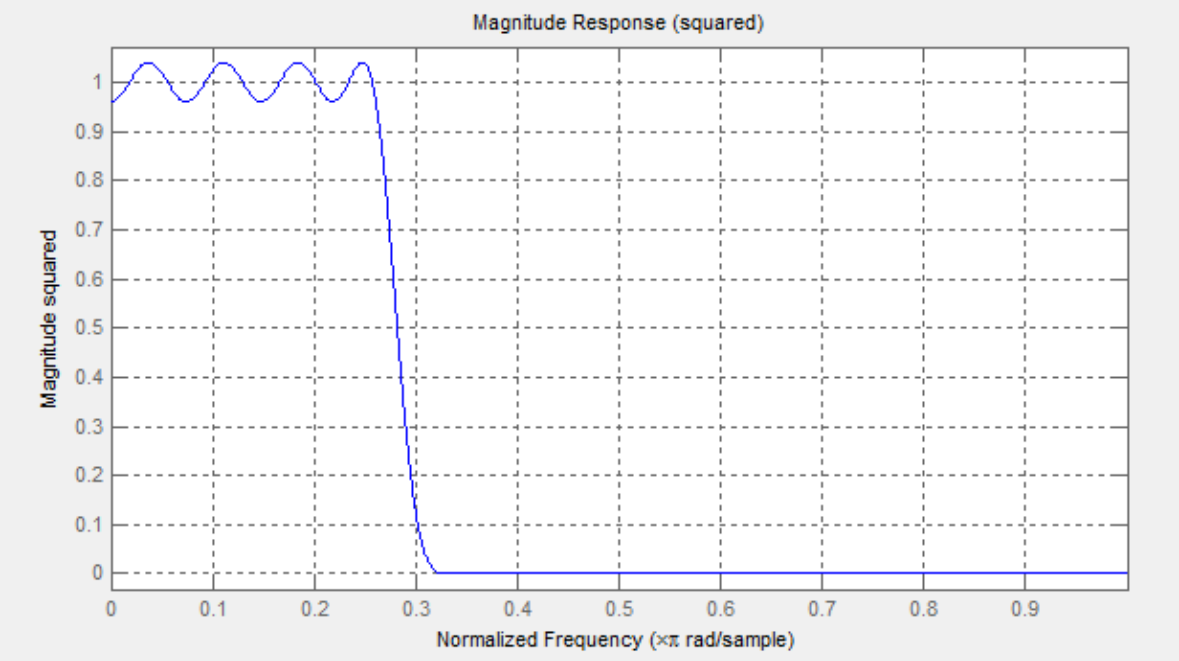


Usando pesos en las bandas:
b = fircls1(n,wo,dp,ds,wp,ws,k)

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

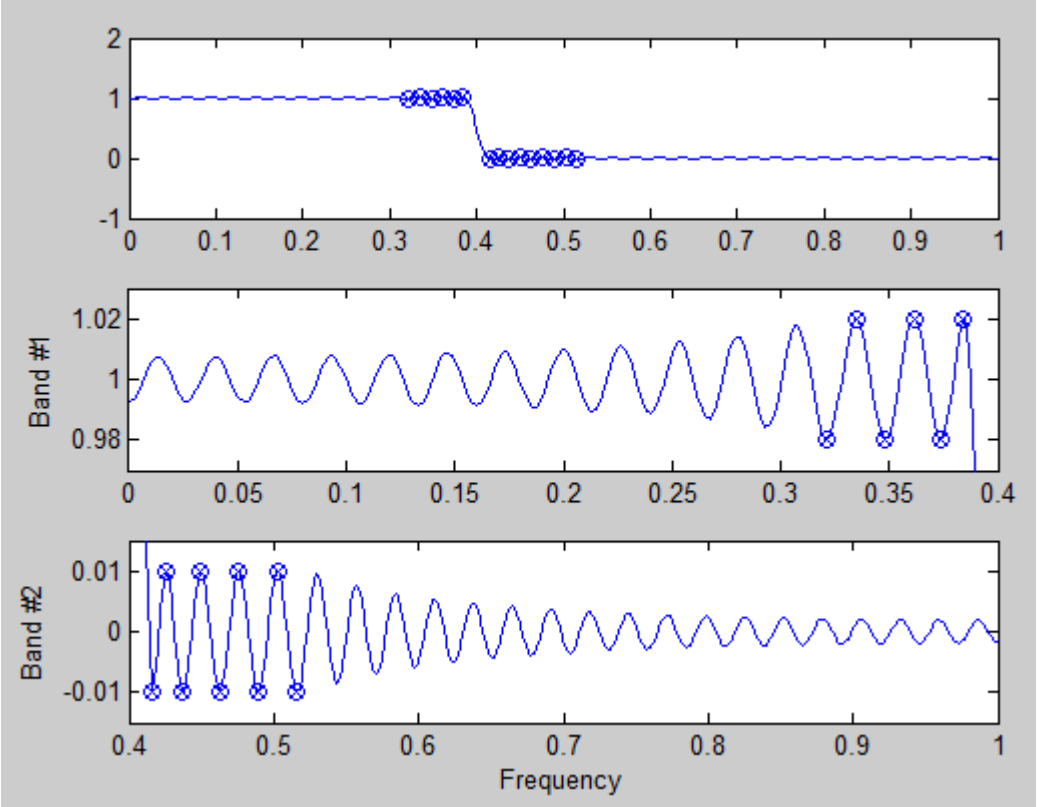
Se usa para filros FIR multibanda en el cual se pueden especificar la cantidad máxima de riple en cada banda.

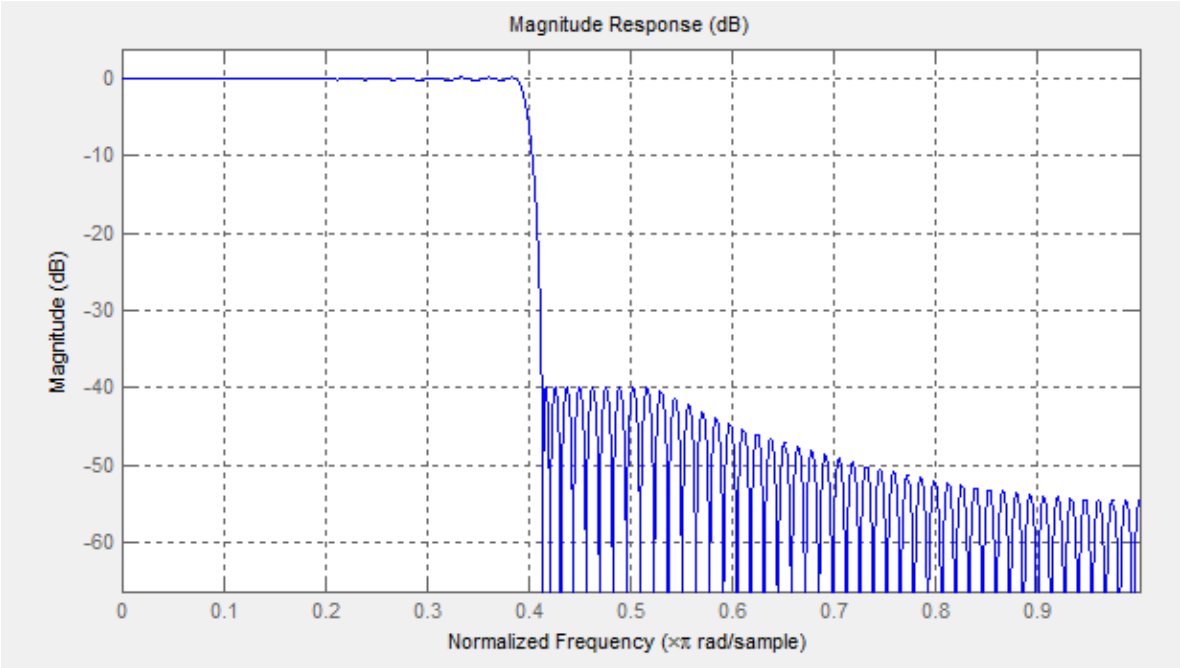
b = fircls(n,f,amp,up,lo)

f: vector de frecuencias de transición
amp: magnitudes

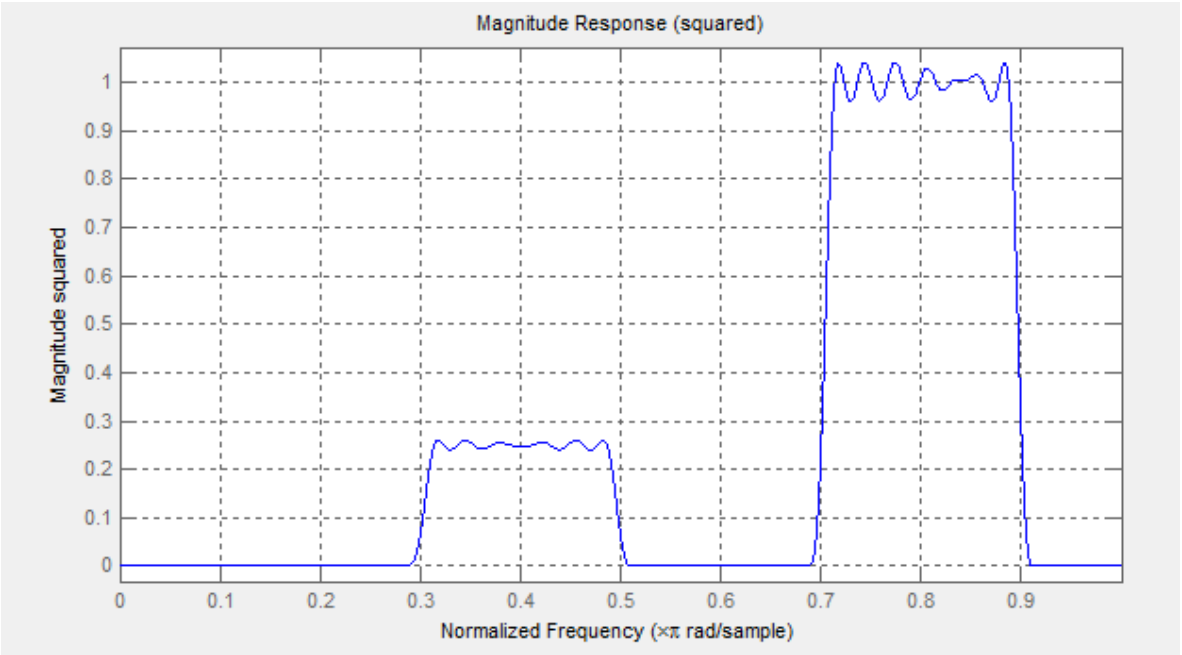
up, lo: define los límites alto y bajo para cada banda

```
%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)
```



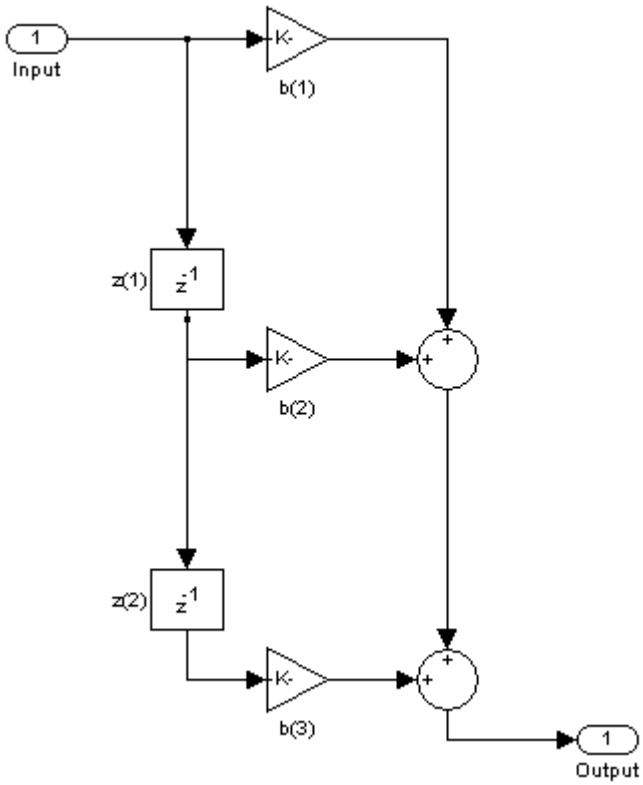



```
n = 129;
f = [0 0.3 0.5 0.7 0.9 1];
a = [0 0.5 0 1 0];
up = [0.005 0.51 0.03 1.02 0.05];
lo = [-0.005 0.49 -0.03 0.98 -0.05];
h = fircls(n,f,a,up,lo);
fvtool(h,1)
```



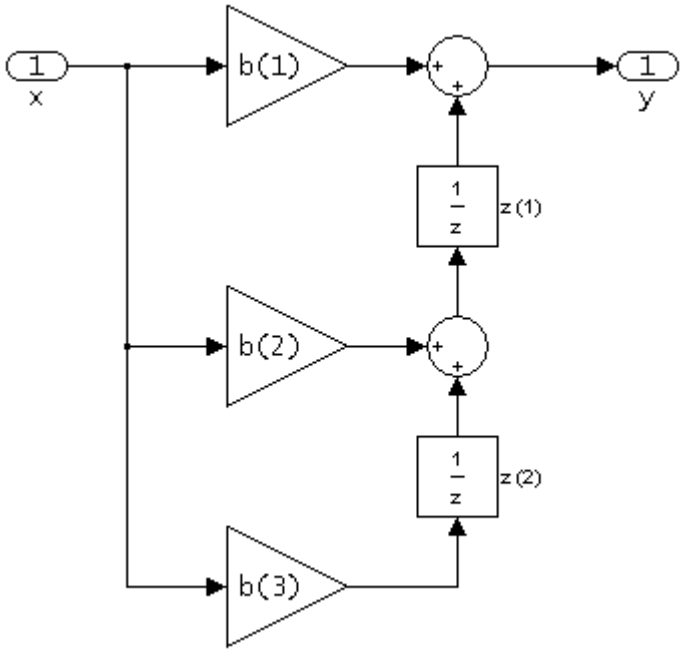
1. IMPLEMENTACIÓN DE FILTROS FIR

a) Forma directa FIR: dfilt.dffir




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

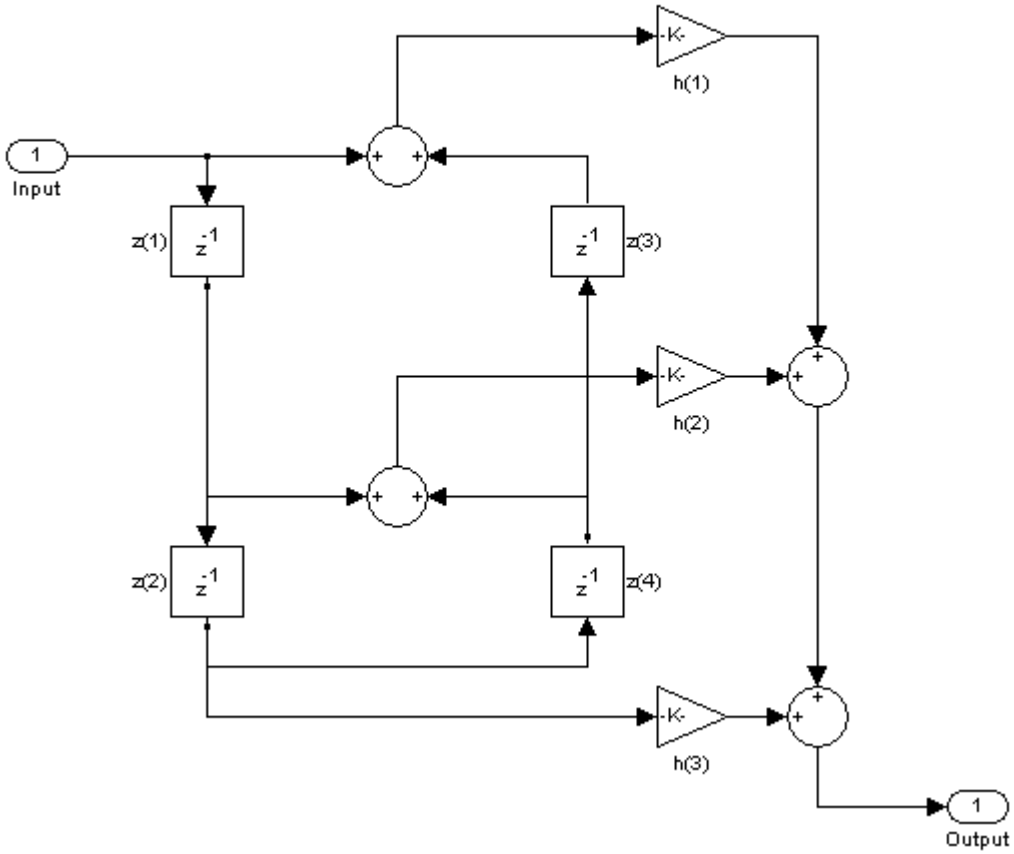
b) Forma directa transpuesta FIR: dfilt.dffirt

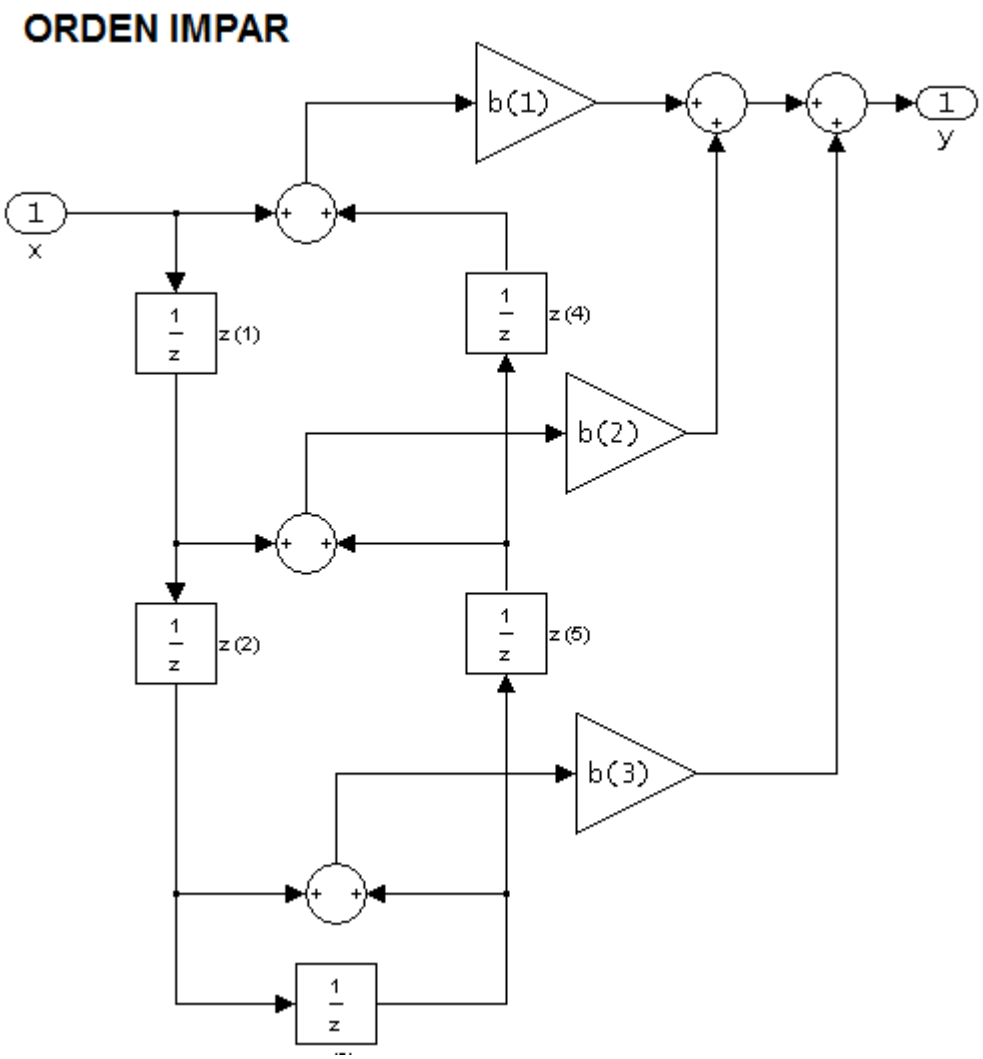



```
b = firpm(30,[0 .1 .2 .5]*2,[1 1 0 0]);
Hd = dfilt.dffirt(b)
```

c) Forma directa simétrica FIR: dfilt.dfsymfir

ORDEN PAR





Ejemplo: Orden Par

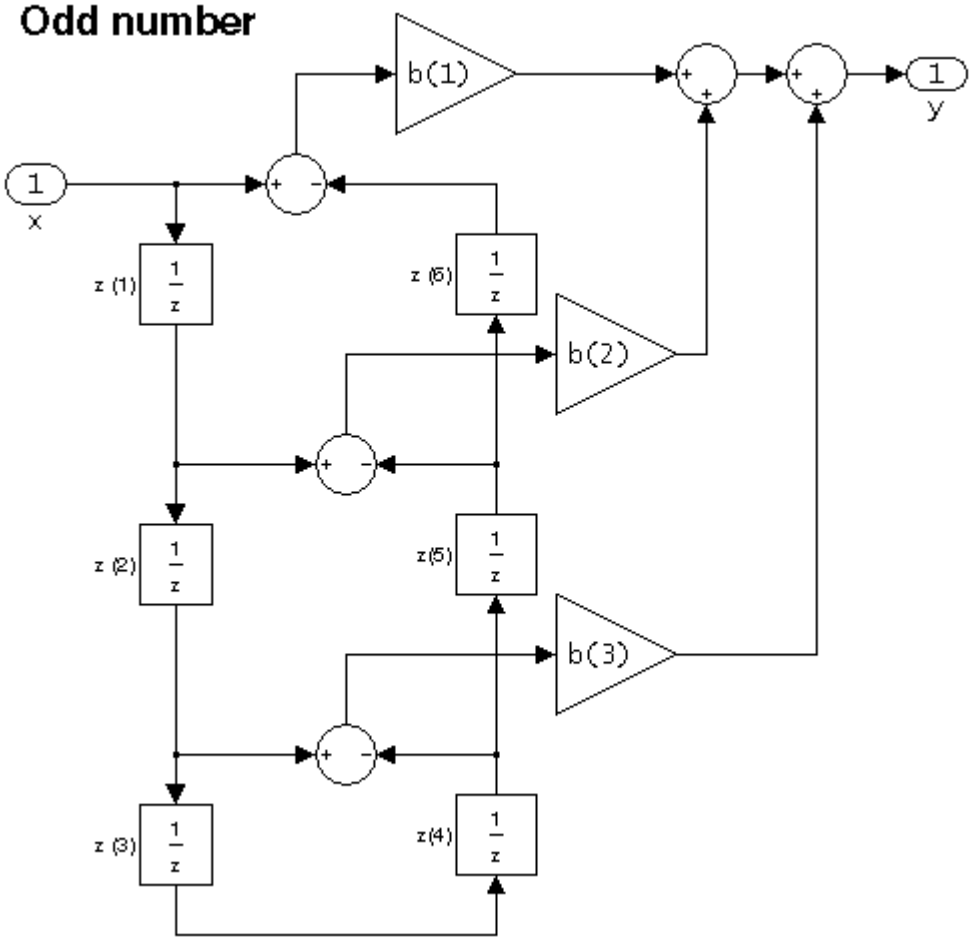
```
b = [-0.008 0.06 0.44 0.44 0.06 -0.008];
Hd = dfilt.dfsymfir(b)
fvtool(Hd)
```

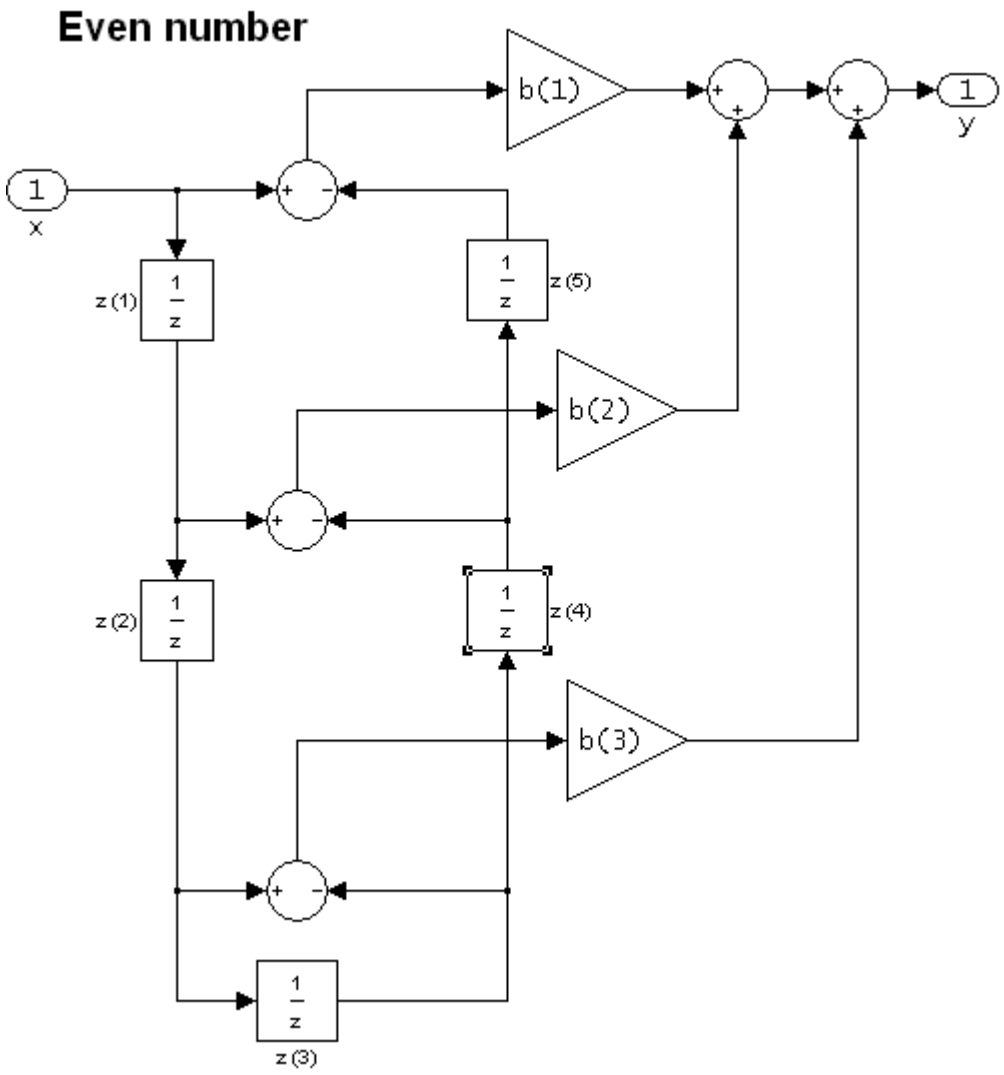
Ejemplo: Orden Impar

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

d) Forma directa antisimétrica FIR: dfilt.dfasymfir

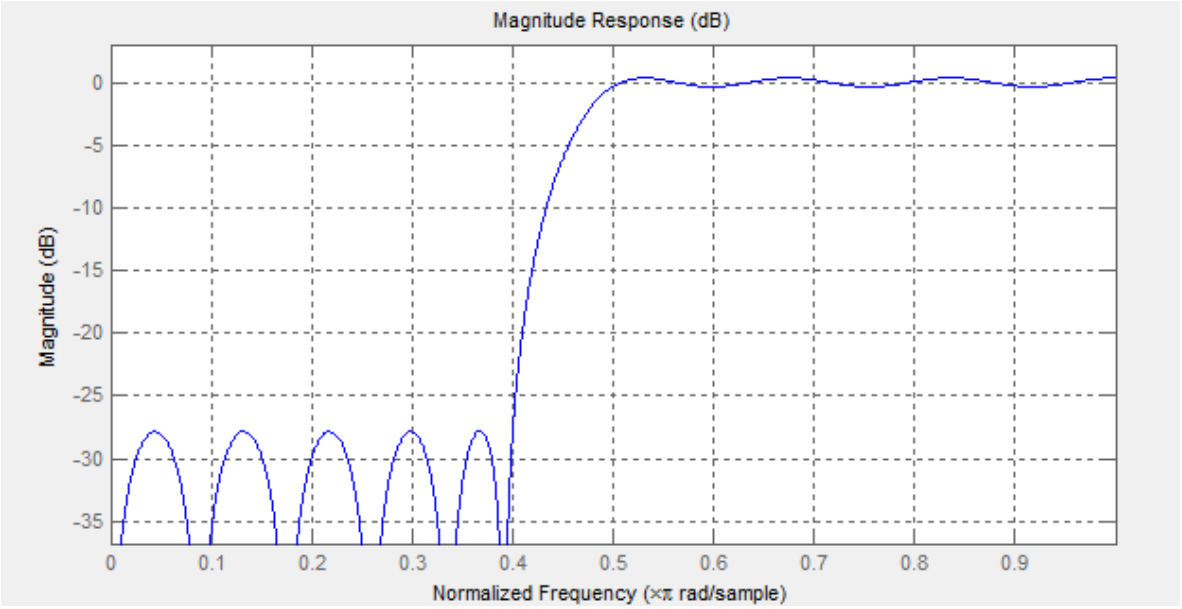
Odd number

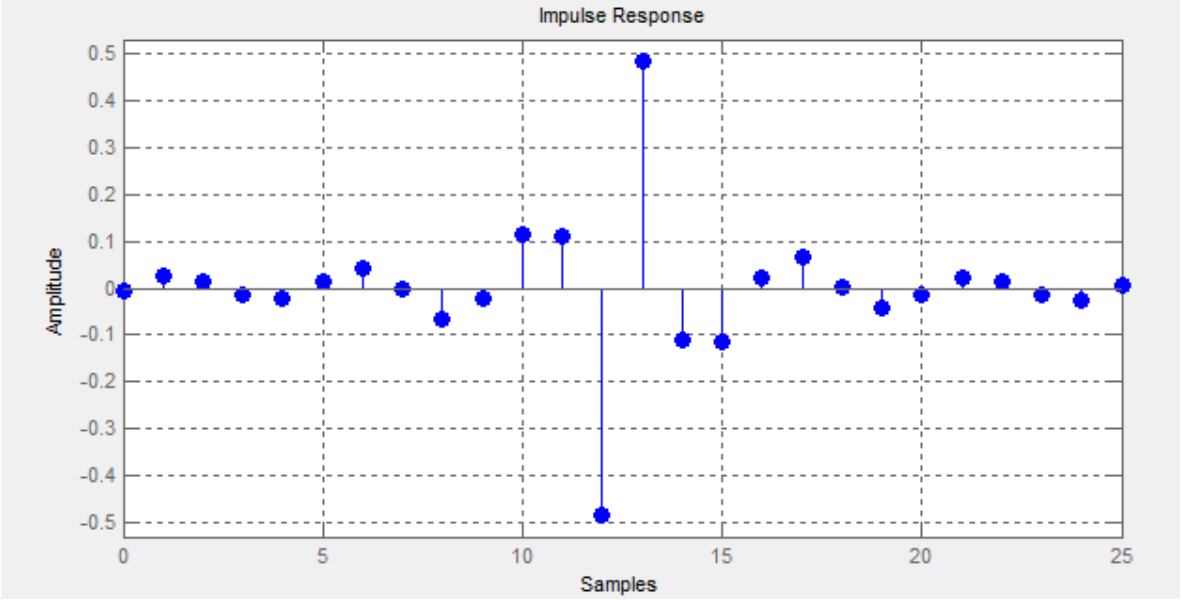




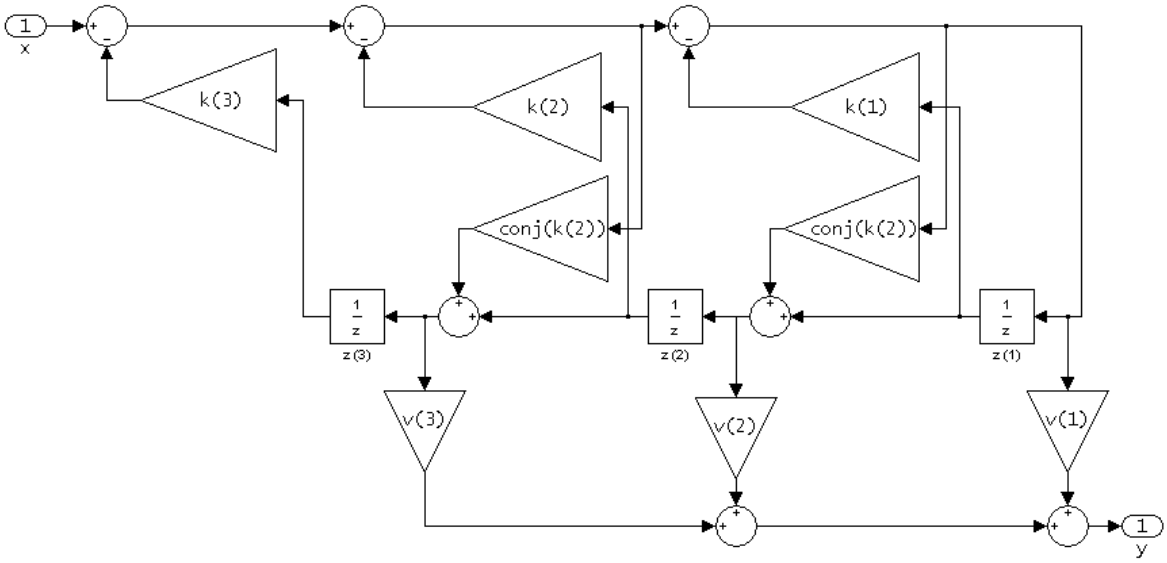
Ejemplo:

```
%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



:

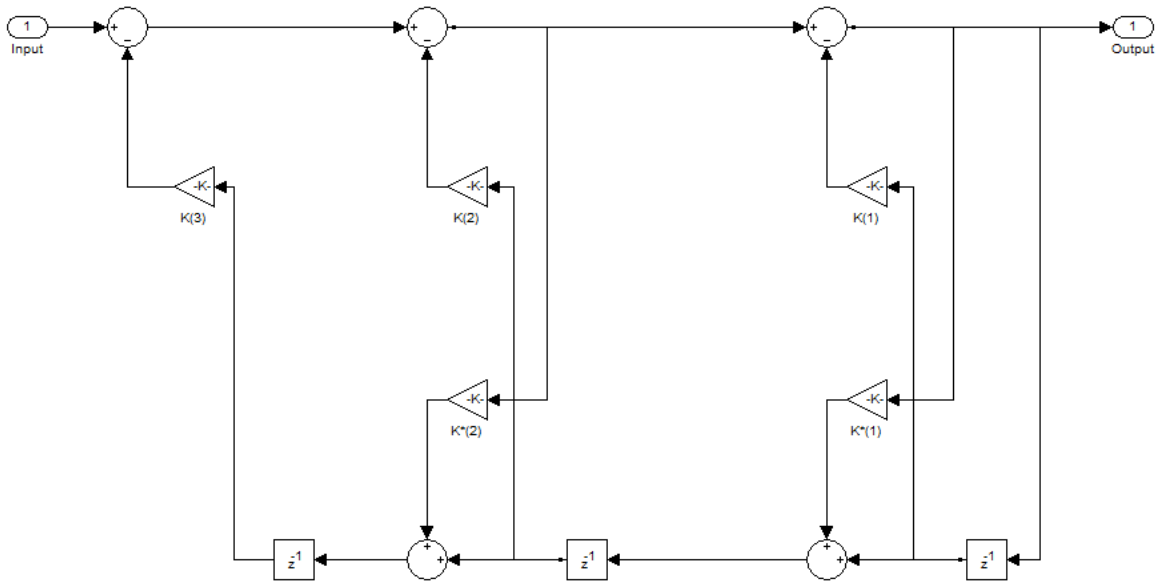
k: coeficientes lattice, v: coeficientes ladder

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)
```



MATLAB

1) Se tiene una señal senoidal de frecuencia 1Hz, y se la muestrea con Fs=30 Hz. Ver la señal en forma continua y discreta. Comprobar que t=n*Ts (tener en cuenta que el primer valor del vector de la señal discreta es 1, no 0).

```
» syms t;f=sin(2*pi*1*t);
» subplot(2,1,1),ezplot(f,[0 2]),grid;
» fs=30;t=0:1/fs:2-1/fs;f=sin(2*pi*1*t);
» subplot(2,1,2),stem(f),axis([1 60 -1 1]),grid;
```

2) Definir un sistema dinámico y(n) = 4*x(n-1) – 3*x(n) y obtener probando los valores h(n) para definir este sistema como una convolución.

```
» n=0:10;
» x=n*10/2;
» n=2:11;
>>f=4*x(n-1)-3*x(n)
» h=[-3,4];
» conv(h,x)
```

3) Realizar la convolución anterior con lazos for, que es tal como lo haríamos al programar en un PIC, DSP, etc.

```
» for n=1:11,
   suma(n)=0;
   for k=1:length(h),
   if (n>k) suma(n)=suma(n)+x(n-k)*h(k);
   end,
   end,
end;
» suma
```

4) Implementar en forma recursiva y no recursiva el sistema y(n) = 1/(n+1) * sum(x(k), k=0:n). (En forma recursiva y(n)=n/(n+1)*y(n-1)+1/(n+1)*x(n))

```
» clear all
»
»
» Fs=20;t=1/Fs:1/Fs:1;x(t*Fs)=sin(2*pi*1*t); %19 valores
» subplot(3,1,1),stem(x);
» for n=1:19,
   m=0;
   for k=1:n,
         m=m+x(k);
   end;
   y(n)=m/n;
end;
» subplot(3,1,2),stem(y);
```

5) Implementación del sistema en forma recursiva:

```
» for n=0:18,
   if (n ~=0)
         yy(n+1)=yy(n)*n/(n+1)+x(n+1)/(n+1); %aparece
                              % +1 porque Matlab
   else      %     toma como primera muestra a la x(1)
         yy(n+1)=x(n+1)/(n+1);
   end;
end;
» subplot(3,1,3),stem(yy);
```

Se puede ver que el resultado es el mismo. La ventaja de implementación en forma recursiva es la rapidez, ya que me ahorro un ciclo for. La desventaja es la memoria que pudiera necesitar.

6) Obtener la transformada discreta de Fourier de una señal discreta. Graficar la señal y los espectros de amplitud y fase.

```
» Fs=300;t=0:1/Fs:1;f=sin(2*pi*5*t);
```

```
» fw=fft(f);
» afw=abs(fw);
» ffw=angle(fw);
» ffwgrados=ffw*180/pi;
» subplot(3,1,1),stem(f(1:100));
» subplot(3,1,2),stem(afw/length(afw));
» subplot(3,1,3),stem(ffwgrados);
```

```
» Fs=300;t=0:1/Fs:1;f=10*sin(2*pi*5*t)+cos(2*pi*20*t);
» fw=fft(f);
» afw=abs(fw);
» ffw=angle(fw);
» ffwgrados=ffw*180/pi;
» subplot(3,1,1),stem(f(1:200));
» subplot(3,1,2),stem(afw/length(afw));
» subplot(3,1,3),stem(ffwgrados);
```

7) Se tiene una señal senoidal de 1 Hz con ruido aleatorio, Fs = 1000 Hz. Dibujar la señal y su espectro de amplitud. Filtrar el ruido lo mejor posible sin usar técnicas de diseño de filtros.

```
» clear all;
» Fs=1000;t=0:1/Fs:10-1/Fs;
» x=sin(2*pi*1*t);
» y=x+0.3*randn(size(t));
» n=1:length(yw);filtrow(n)=0;
» n=5:12;filtrow(n)=1;  % f=0.001. 10000 muestras => en 10
                        %(10/10000=0.001) tengo la componente
» filtrow(10000-n)=1;
» filtron=real(ifft(filtrow));
» senalf=conv(filtron,y);
» subplot(2,1,1),plot(y(1:8000));
» subplot(2,1,2),plot(senalf(1:8000));
```

8) Realizar el mismo filtro anterior, sólo que h(n) tenga 20 valores, y no 10000. Sacar conclusiones porque el filtrado no es tan bueno. Implementar con lazos for que es cómo se implementaría en un sistema microprocesado.

```
» n=1:20;filtro(n)=0;
» filtro(1)=1;filtro(20)=1;
» filtron=real(ifft(filtro));
» senalf=conv(filtron,y);
» subplot(2,1,1),plot(y(1:5000));
» subplot(2,1,2),plot(senalf(1:5000));
```

La señal filtrada no es tan buena porque al tener 20 valores la frecuencia discreta más chica que puede tener es 1/20 = 0.05. La frecuencia discreta de mi señal es F/Fs = 0.001. La cantidad mínima de valores de h(n) para tener un filtrado “perfecto” sería 1000 => 1/1000=0.001.
Al hacer que filtro(1) = 1 entonces dejamos pasar todas las frecuencias menores a 1/20 = 0.05.

```
» for n=1:length(y),
   suma(n)=0;
   for k=1:length(filtron),
            if (n-k)>0 suma(n)=suma(n)+filtron(k)*y(n-k);
            end;
   end;
   end;
» figure(2),plot(suma(1:5000));
```

9) Obtener coeficientes para una implementación IIR a partir de los coeficientes de una implementación FIR. Suponer que H(s) es tal que h(t)= exp(-t)+sin(t)+cos(2*pi*10*t). Comparar las respuestas en frecuencia H(z) con los coeficientes FIR e IIR.

```
tt=0:1/100:3;
h=exp(-tt)+sin(tt)+cos(2*pi*10*tt);
t=0:1/100:10;
[hh,ww]=freqz(h,1,100);
figure(1),subplot(211),plot(abs(hh));
%Variar los polos y ceros y ver como varia la resp en frec con la original.
[biir,aiir]=invfreqz(hh,ww,6,6)
[hhiir,wwiir]=freqz(biir,aiir,100);
subplot(212),plot(abs(hhiir))
vi=sin(t);
figure(2),subplot(211),plot(filter(h,1,vi))
sis=tf(biir,aiir);
figure(2),subplot(212),plot(filter(biir,aiir,vi))
```

10) Diseño de filtros digitales dando como parámetros la amplitud para las diferentes frecuencias normalizadas (1=Fs/2). Calcular los coeficientes de un filtro FIR y los de un IIR si Fs=100 Hz, fc1= 16.66 Hz y fc2 = 23.33 Hz. Probar con una señal de entrada utilizando este filtro.

```
%Cálculo de filtros digitales
clear all;
fs=100;
f=0:1/15:1; %f=0:fs/(2*15):fs/2 Hz.
m=[0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0];%16.66Hz=5/15*50Hz
                                    %23.33Hz=7/15*50Hz
%El tamaño de m, y por lo tanto f dependerá de la resolución requerida
%para el filtro.
%Definición de la señal de entrada.
```

```
t=0:1/fs:3;
senal=sin(2*pi*16*t)+sin(2*pi*2*t);
figure(1),plot(t,senal);
%Diseño del filtro IIR (40 coeficientes 20 a y 20 b)
[b,a]=yulewalk(20,f,m);
[H,wt]=freqz(b,a,150);
figure(2),subplot(211),plot(wt/pi*fs/2,abs(H)),ylabel('Implementacion IIR');
subplot(212),plot(t,filter(b,a,senal));
```

Diseño de filtros digitales dando como parámetros la amplitud para las diferentes frecuencias normalizadas (1=Fs/2). Calcular los coeficientes de un filtro FIR y los de un IIR si Fs=100 Hz, fc1= 16.66 Hz y fc2 = 23.33 Hz. Probar con una señal de entrada utilizando este filtro.

```
%Diseño del filtro FIR (100 coeficientes)
clear H wt;
h=remez(100,f,m);
[H,wt]=freqz(h,1,150);
figure(3),subplot(311),plot(wt/pi*fs/2,abs(H)),ylabel('Implementacion FIR');
%Para aplicar el filtro anterior definido por b y a a x se hace:
%Con convolución
res=conv(h,senal);
subplot(312),plot(t,res(1:length(t)));
%Con filter
subplot(313),plot(t,filter(h,1,senal));
```