PROCESAMIENTO DIGITAL DE SEÑALES

PROFESOR: ING. JORGE A. POLANÍA P.

FILTROS DIGITALES – FILTROS IIR
(Respuesta Impulsional Infinita)

1. INTRODUCCIÓN

El filtro digital es la implementación en hardware o software de una ecuación en diferencias que tiene una entrada digital x(n) y una salida y(n), como la siguiente,

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

Despejando y(n):

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

La función de transferencia del filtro digital se obtiene aplicando Transformada Z:

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

Donde $a(1) = 1, a(2), \ldots., b(1),\ b(2), \ldots\ldots$ son los coeficientes del filtro.

Respuesta al impulso (convolución) de un filtro:

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

Ejemplo:

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

Ecuación en diferencias del filtro:

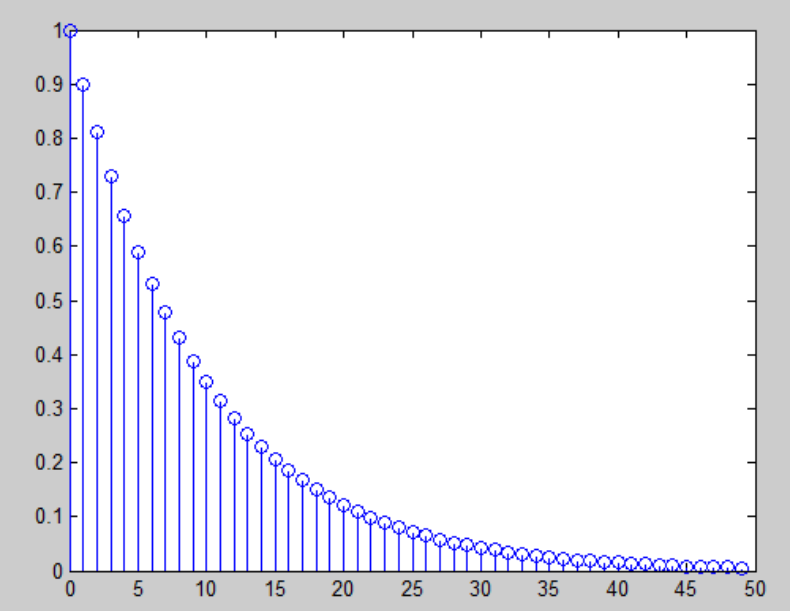
$$y(n) = x(n) + 0.9y(n-1)\,, \quad x(n) = \delta(n),$$

Función de transferencia:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - 0.9z^{-1}}$$

Usando Matlab, para 50 muestras:

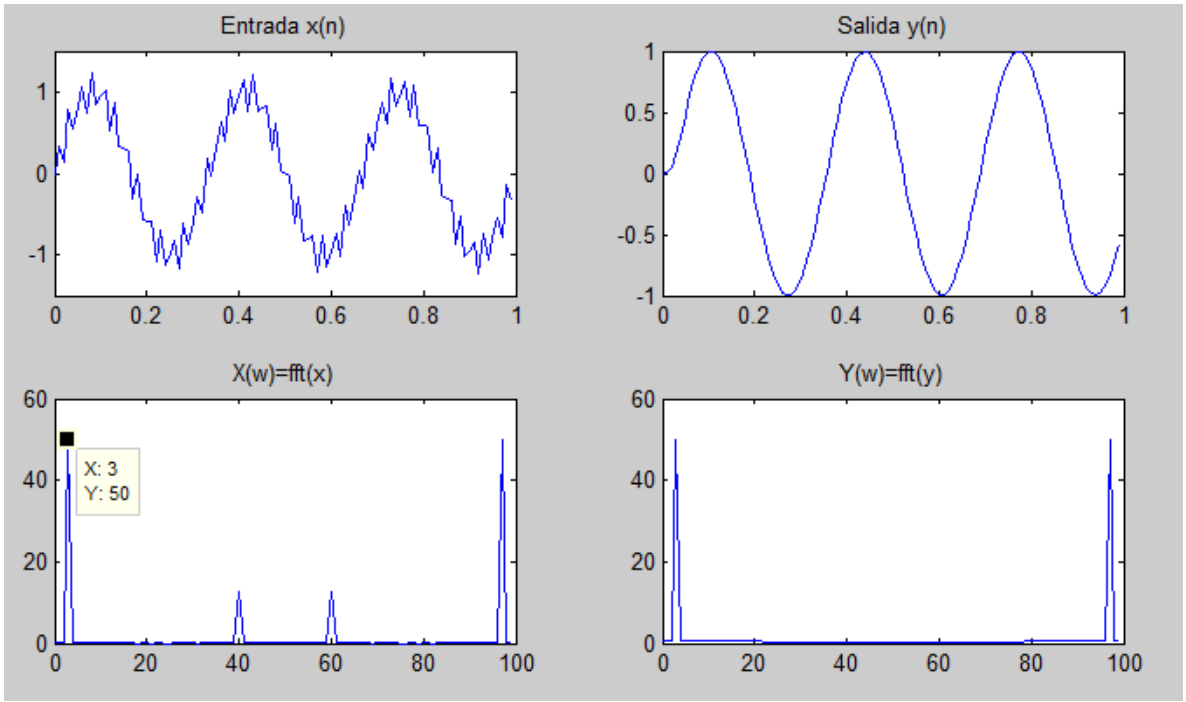
```
n=0:49;
%señal impulso
imp = [1; zeros(49,1)];
%coeficientes del filtro
b=1;
a=[1 -0.9];
%respuesta al impulso
h = filter(b,a,imp);
stem(n,h)
```



Respuesta del filtro a una señal de entrada

```
%implementación de filtros
% a) Coeficientes del filtro
[b,a] = butter(5,0.4);
% b) Estructura del filtro
Hd = dfilt.df2t(b,a);     % Forma directa II transpuesta
```

```
fs=100;    %Frecuencia de muestreo
t=0:1/fs:0.99;
% c)Señal de entrada
x=sin(2*pi*t*3)+0.25*sin(2*pi*t*40);
% d)Respuesta del filtro
y=filter(Hd,x);
subplot(221)
plot(t,x)
title('Entrada x(n)')
subplot(222)
plot(t,y)
title('Salida y(n)')
%Respuesta en frecuencia
f=0:99;
Xw=fft(x);
MagXw=abs(Xw);
subplot(223)
plot(f,MagXw)
title('X(w)=fft(x)')
Yw=fft(y);
MagYw=abs(Yw);
subplot(224)
plot(f,MagYw)
```



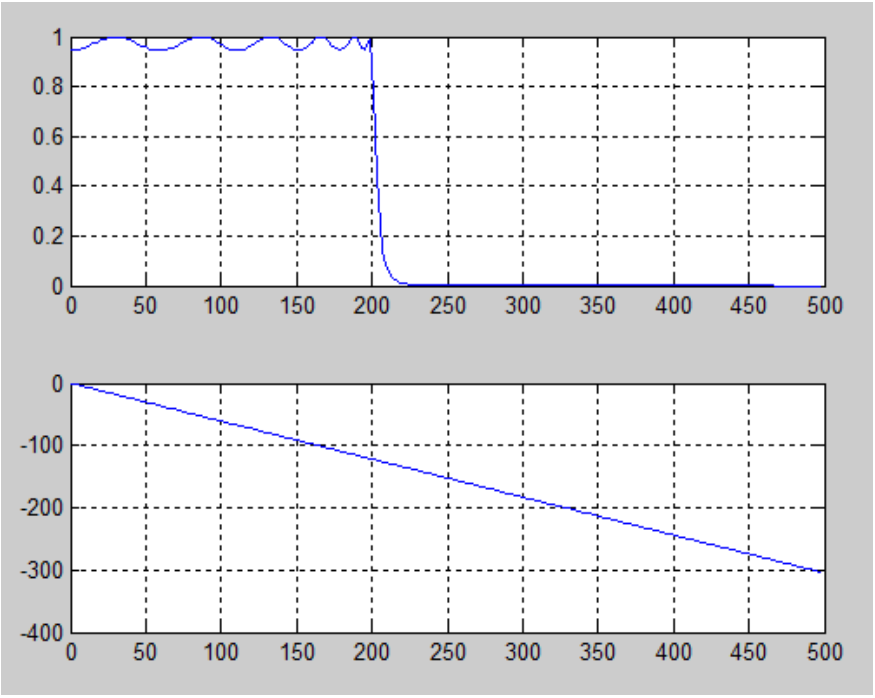
Respuesta en frecuencia de un filtro:

$$H(e^{jw}) = \frac{b(1) + b(2)e^{-jw} + \cdots + b(N+1)e^{-jwn})}{a(1) + a(2)e^{-jw} + \cdots + a(M+1)e^{-jwm})}$$

Ejemplo para un Chebyshev Tipo 1:

Obtener la respuesta en frecuencia para un filtro Chebyshev tipo1 de orden 12, con riple o rizado en la banda de paso de 0.5 dB en una frecuencia de 200Hz muestreado con 1000Hz

```
% coeficientes del filtro
%[b,a]=cheby1(n,Rp,wp/wN)
[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(211)
plot(f,mag)
grid
%grafica de la fase
fase=unwrap(f*180/pi);
subplot(212)
plot(f,fase)
grid
```



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 ecuación en diferencias:

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

Y como función de transferencia:

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

Características

- 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.
- Tiene fase lineal con la frecuencia.

FILTROS IIR

La salida de los filtros IIR depende de las entradas actuales y pasadas, y además de las salidas en instantes anteriores. Esto se consigue mediante el uso de realimentación de la salida.

Tiene como ecuación en diferencias:

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

Y como función de transferencia:

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

Características

- 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

2. 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 pasa bajo 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.

3. DISEÑO DE FILTROS IIR (Respuesta Impulsional Infinita)

Hay varios métodos para realizar el diseño:

- Usando los algoritmos de los filtros análogos y digitalizando
- En forma directa especificando la respuesta en frecuencia

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.

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

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

Coeficientes del filtro:

[b,a] = butter(n,w_n)
[b,a] = butter(n,w_n, 'Tipo')

Polos y ceros:

[z,p,k] = butter(n,w_n)
[z,p,k] = butter(n,w_n, 'Tipo')
Tipo= high, low, bandpass, bandstop (w_n=[w_1 w_2])

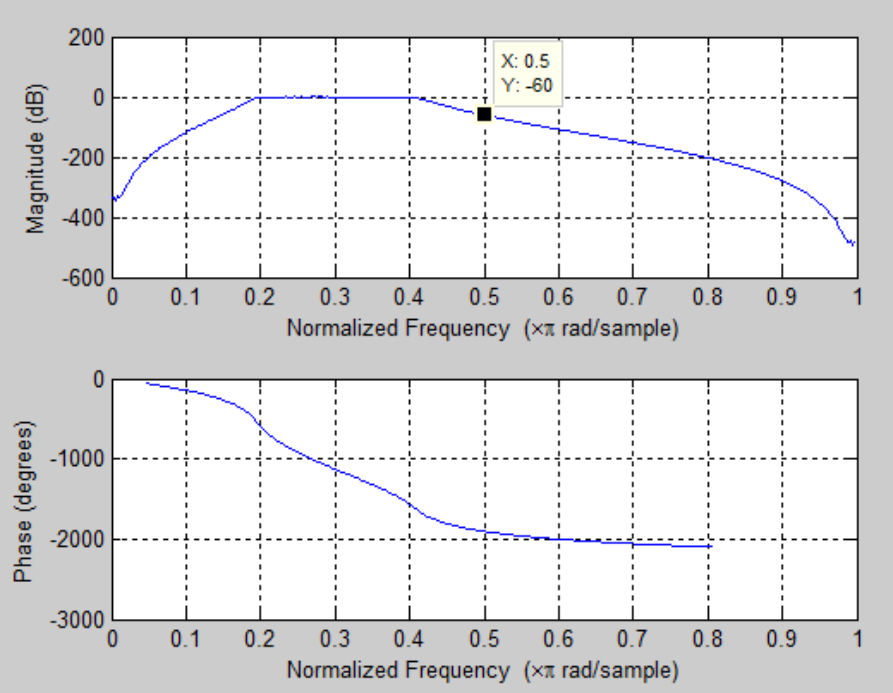
Respuesta en frecuencia:

freqz(b,a): Grafica la respuesta en frecuencial en magnitud y fase
H=freqz(b,a,w): Retorna la respuesta de frecuencial en las frecuencias designadas por el vector w
[H,f]=freqz(b,a,N,Fs): Retorna el vector H y el vector f en Hz, donde es la frecuencial de muestreo.

Ejemplo: Pasa banda

Se quiere un filtro pasa banda de 1000 a 2000 Hz, la banda stop entre 500 Hz y 2500 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);
freqz(b,a)
```



- **Filtro Chebyshev Tipo I**

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

Commandos de Matlab:

[n,wn]=cheb1ord(wp,ws,Rp,Rs)
z,p,k] = cheby1(n,R,Wp,'Tipo')

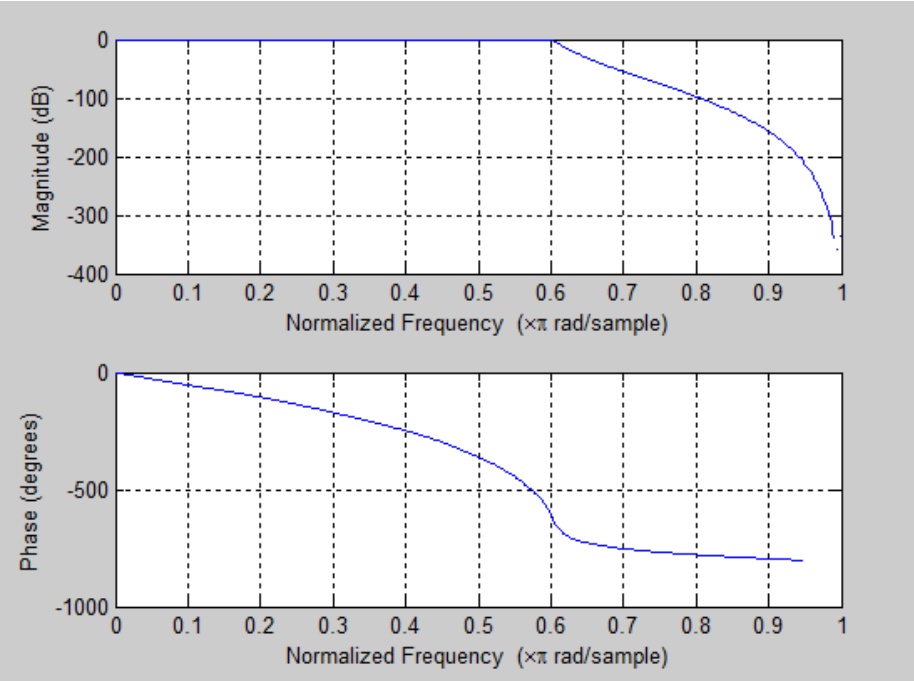
[b,a] = cheby1(n,R,Wp,'Tipo')

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.

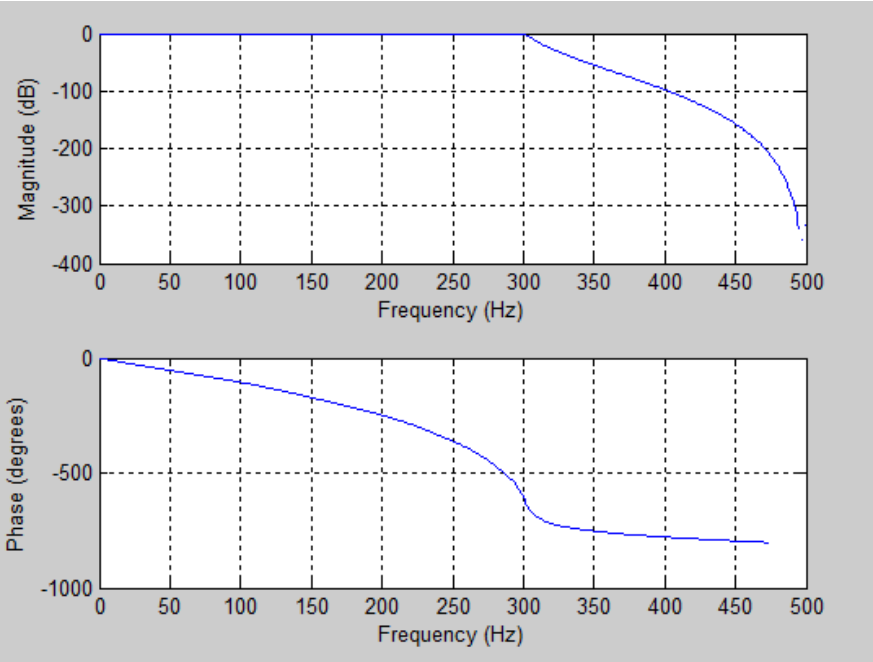
Ejemplo: Pasa bajo

Para una frecuencia de muestreo de 1000 Hz diseñar un filtro Chebyshev Tipo I de orden 9, pasa bajo, con 0.5 dB de riple en la banda de paso y una frecuencia de corte en la banda de paso de 300 Hz.

```
[b,a]=cheby1(9,0.5,300/500);
freqz(b,a)
```




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



- **Filtro Chebyshev Tipo II**

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

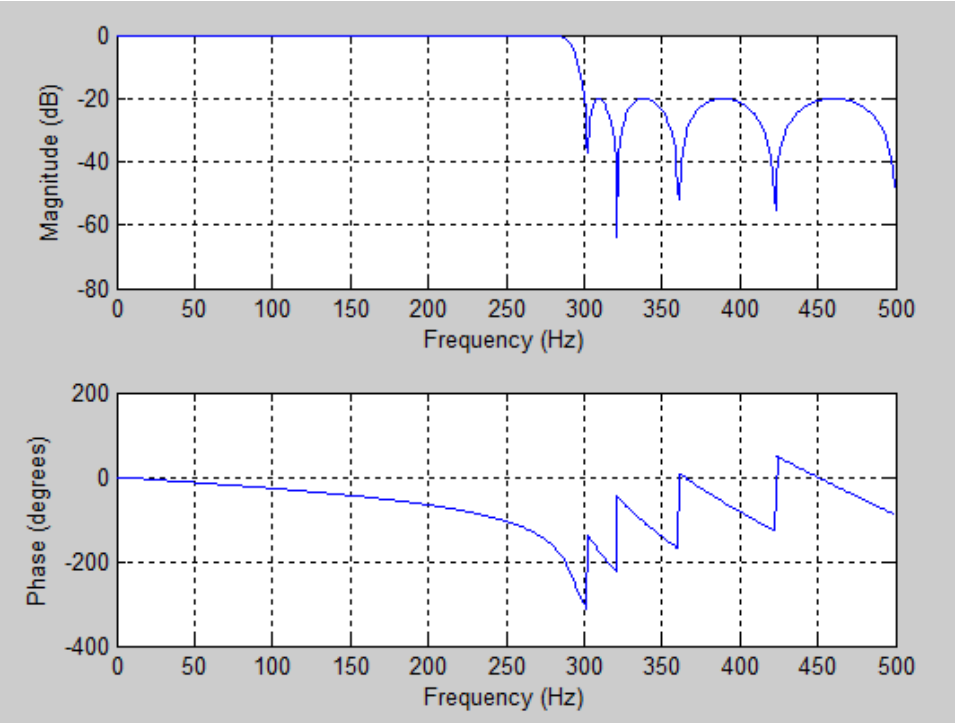
Comandos de Matlab

[n,wn]=cheb2ord(n,Rs,ws)
[z,p,k] = cheby2(n,R,Ws,'Tipo')
[b,a] = cheby2(n,R,Ws,'Tipo')

Ejemplo: Pasa bajo

Para una frecuencia de muestreo de 1000 Hz diseñe un filtro pasa bajo Chebyshev II de orden 9, 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.

```
[b,a] = cheby2(9,20,300/500);
freqz(b,a,512,1000)
```



- **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.

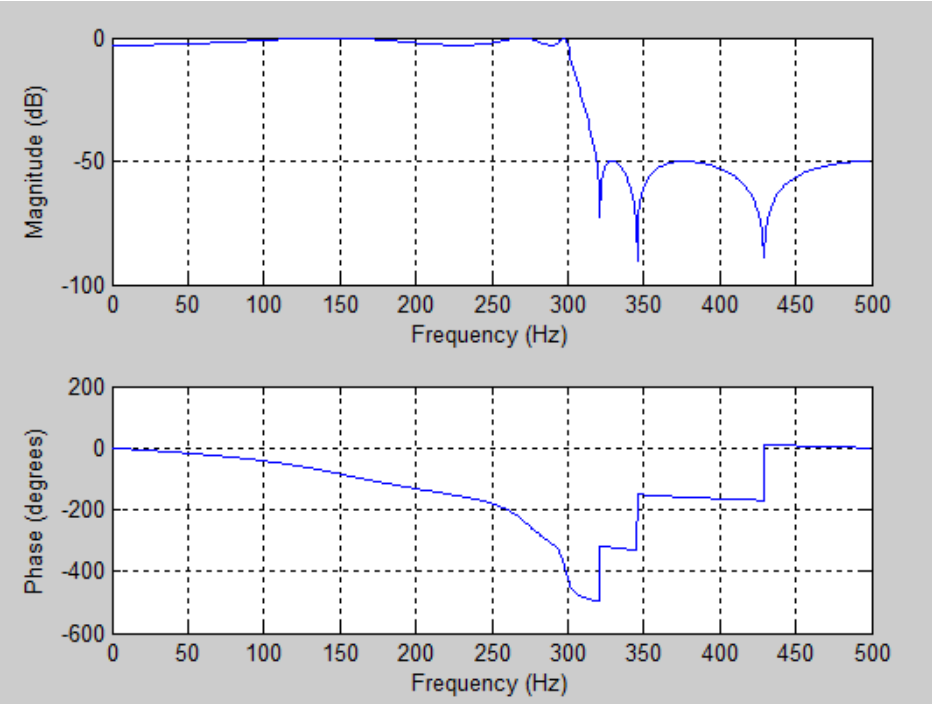
[n,wn]=ellipord(wp,ws,Rp,Rs)
[z,p,k] = ellip(n,Rp,Rs,Wp)
[z,p,k] = ellip(n,Rp,Rs,Wp,'Tipo')
[b,a] = ellip(n,Rp,Rs,Wp)
[b,a] = ellip(n,Rp,Rs,Wp,'Tipo')

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. Fs=1000.

```
[b,a] = ellip(6,3,50,300/500);
freqz(b,a,512,1000)
```



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

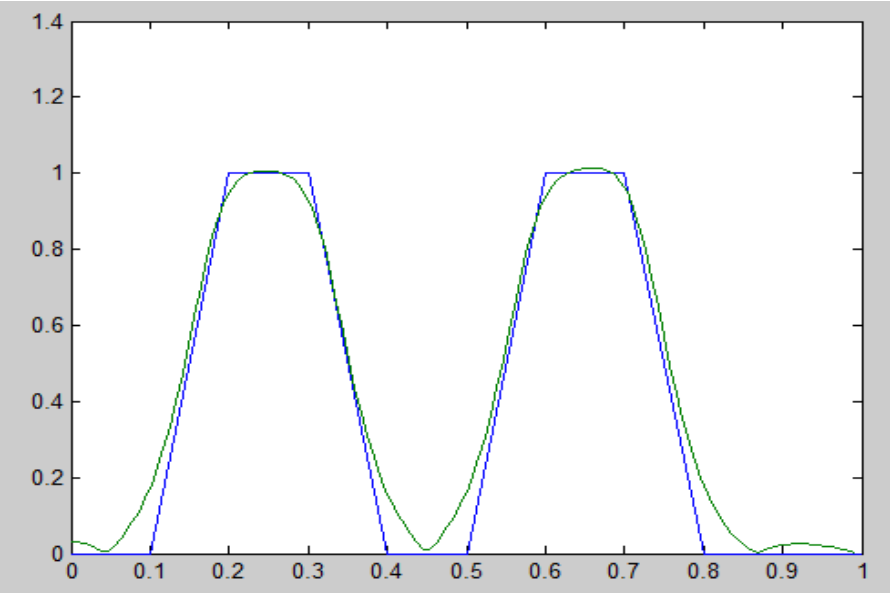
Se diseña en forma directa especificando la respuesta en frecuencia. El método encuentra la transformada inversa FFT y la resuelve utilizando la ecuación Yule – Walker.

[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.

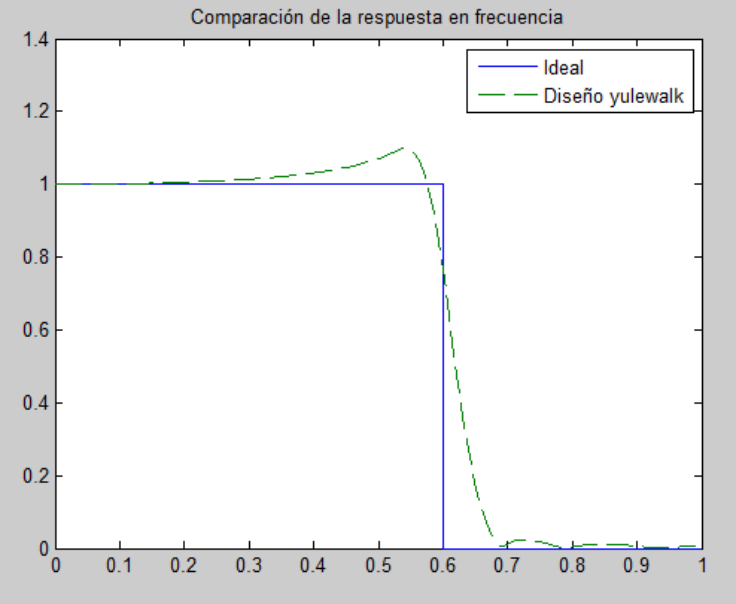
Ejemplo: Filtro multibanda de orden 10

```
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')
```



4. IMPLEMENTACIÓN DE FILTROS DIGITALES

Después de haber diseñado el filtro, la función de transferencia o ecuación en diferencias del filtro debe implementarse. 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.

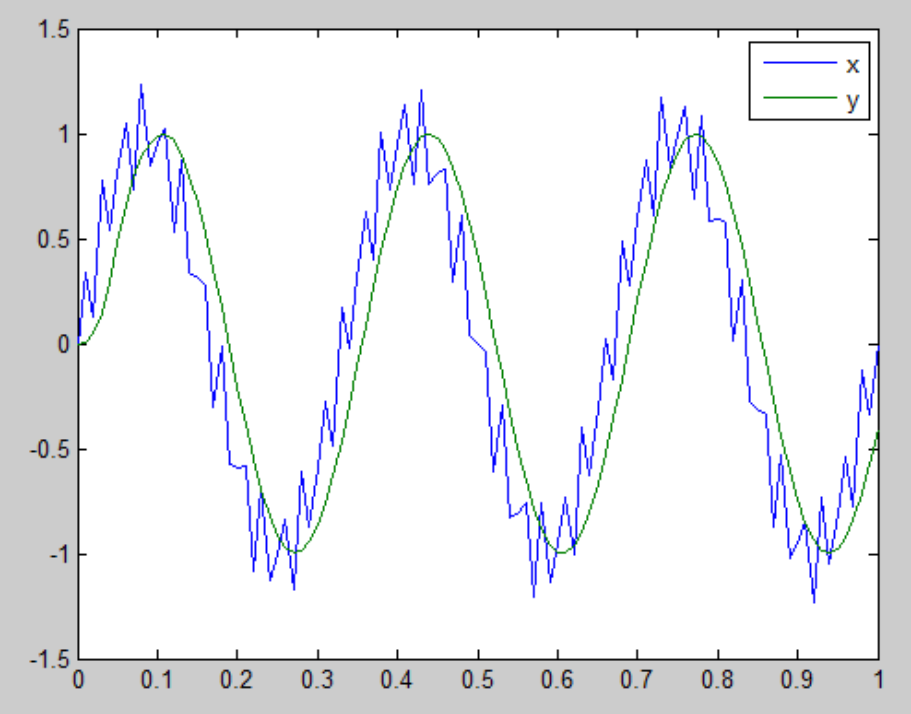
Función dfilt, filter

Procedimiento para la implementación:

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

Por ejemplo, diseñar e implementar con estructura *forma directa traspuesta II* para un filtro Butterworth.

```
%implementación del filtro
[b,a] = butter(5,0.4);


% Estructura: Forma directa II transpuesta
Hd = dfilt.df2t(b,a);
fs=100; % frecuencia de muestreo
t=0:1/fs:1;
% Señal de entrada
x=sin(2*pi*t*3)+0.25*sin(2*pi*t*40);
% Filtrado
y=filter(Hd,x);
% Gráficas
plot(t,x,t,y)
legend('x','y')
```



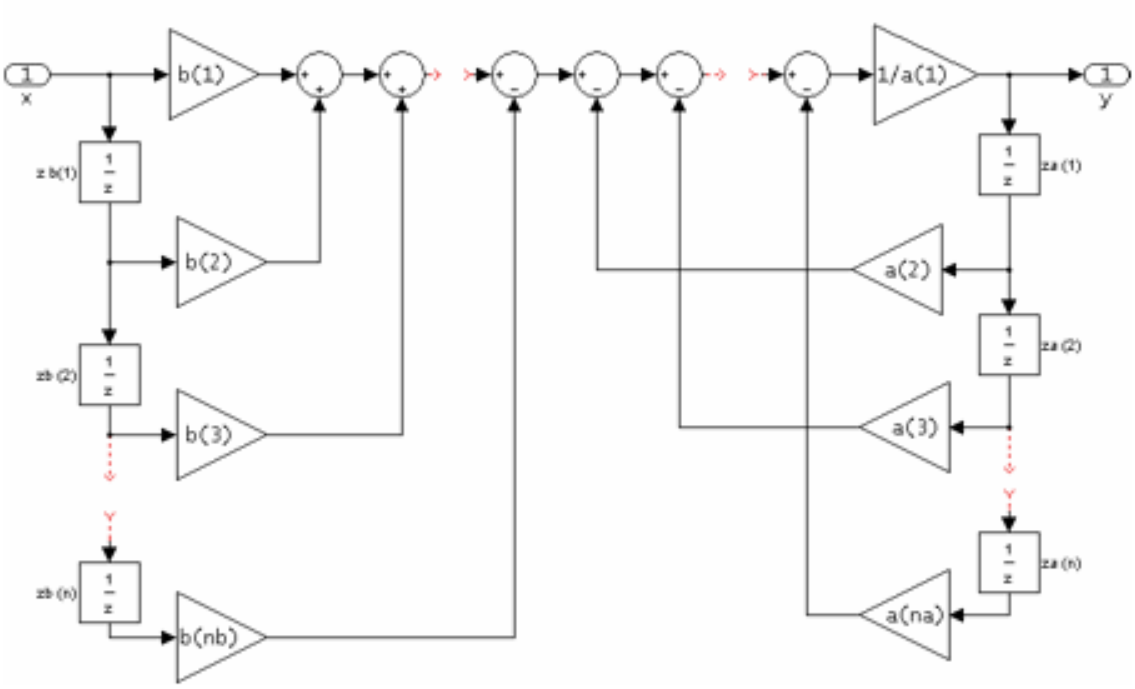
5. ESTRUCTURA DE FILTROS IIR

Se tienen las siguientes estructuras para su implementación:

- Forma Directa I
- Forma Directa I SOS (second order section)
- Forma Directa I Transpuesta
- Forma Directa I Transpuesta SOS
- Forma Directa II
- Forma Directa II SOS
- Forma Directa II Transpuesta
- Forma Directa II Transpuesta SOS
-

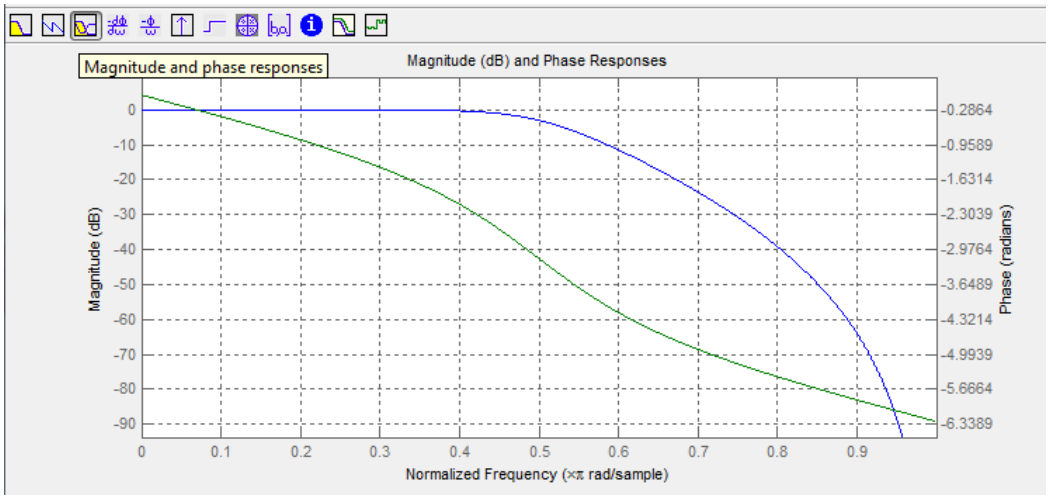
a) Forma Directa I: dfilt.df1

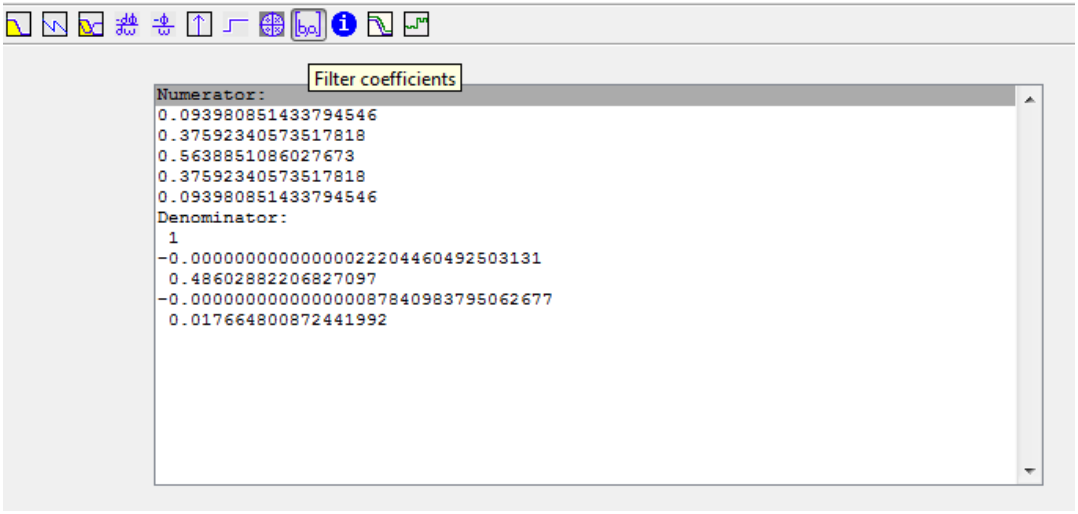
$$y(n) = b(1)x(n) + b(2)x(n-1) + b(3)x(n-2) + \cdots$$
$$-a(2)y(n-1) - a(3)y(n-2) - \cdots$$

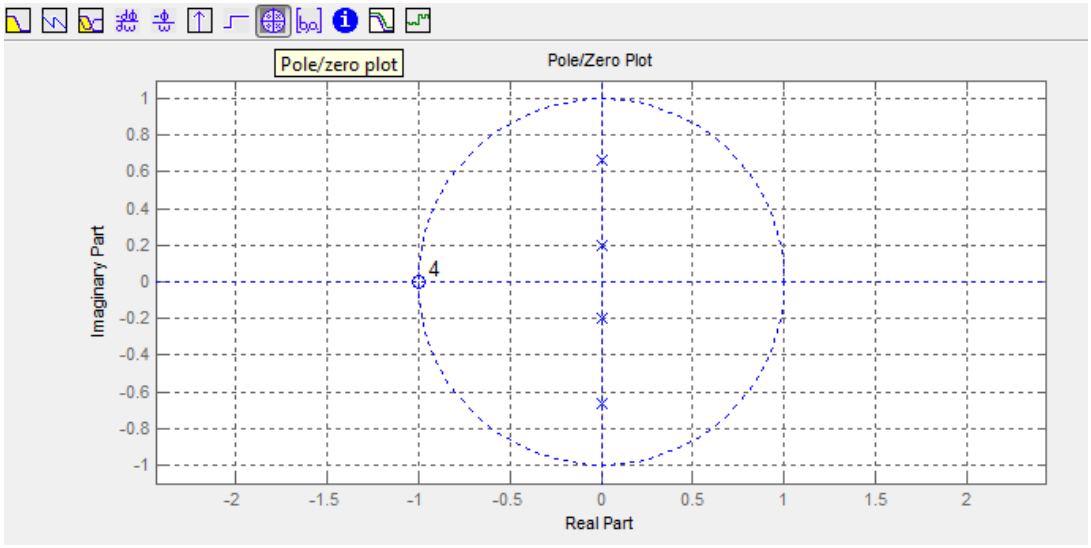


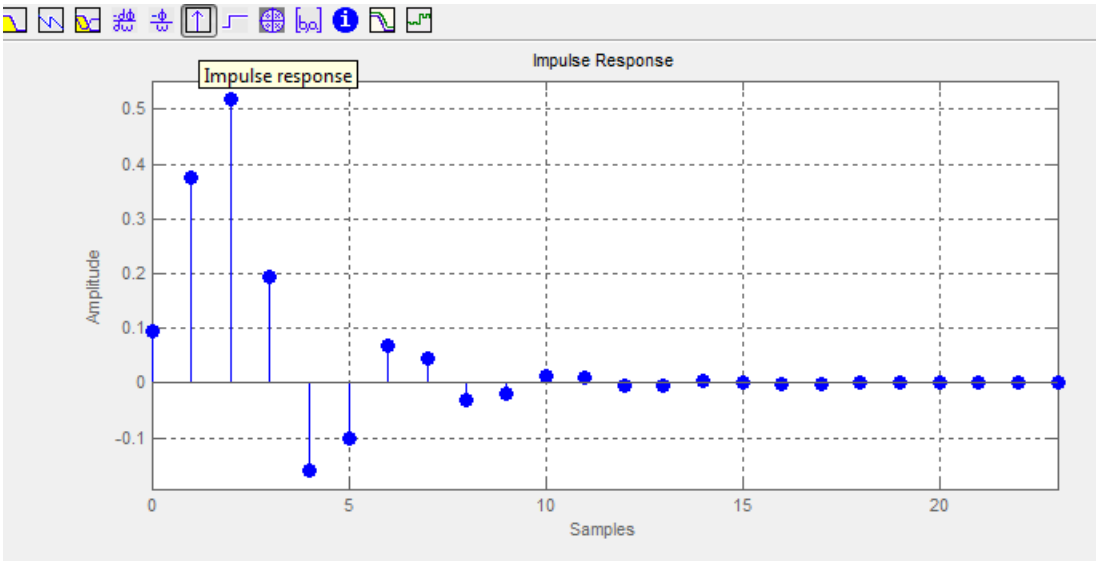
Ejemplo

```
%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)
```



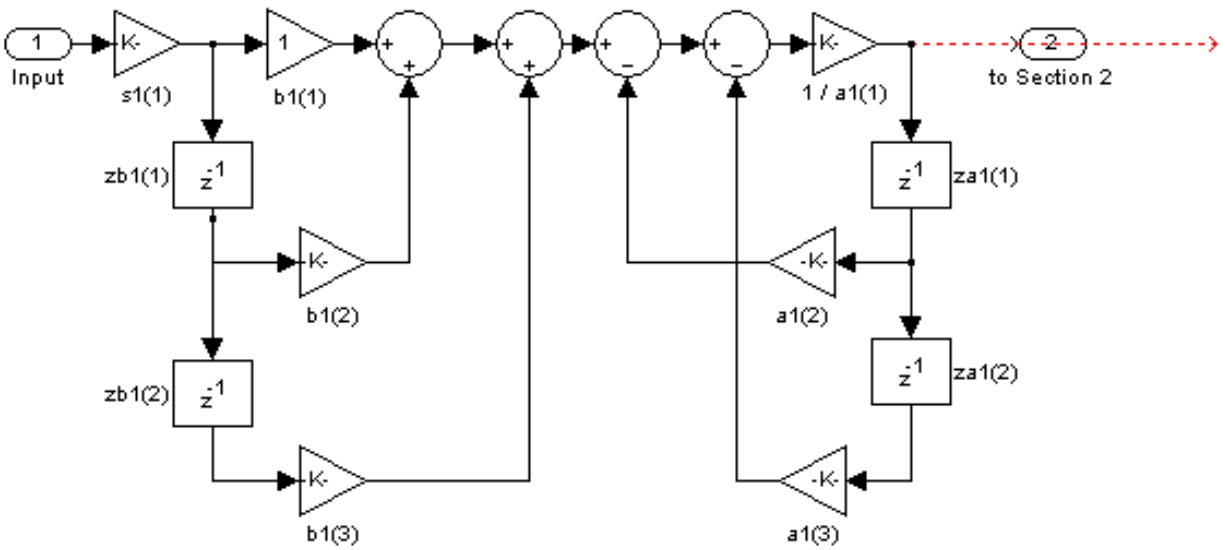






b) Forma Directa I sos (second order section): dfilt.df1sos

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 elíptico, pasa bajo de orden 6, Rs=1, Rp=60 dB, Wp=0.4

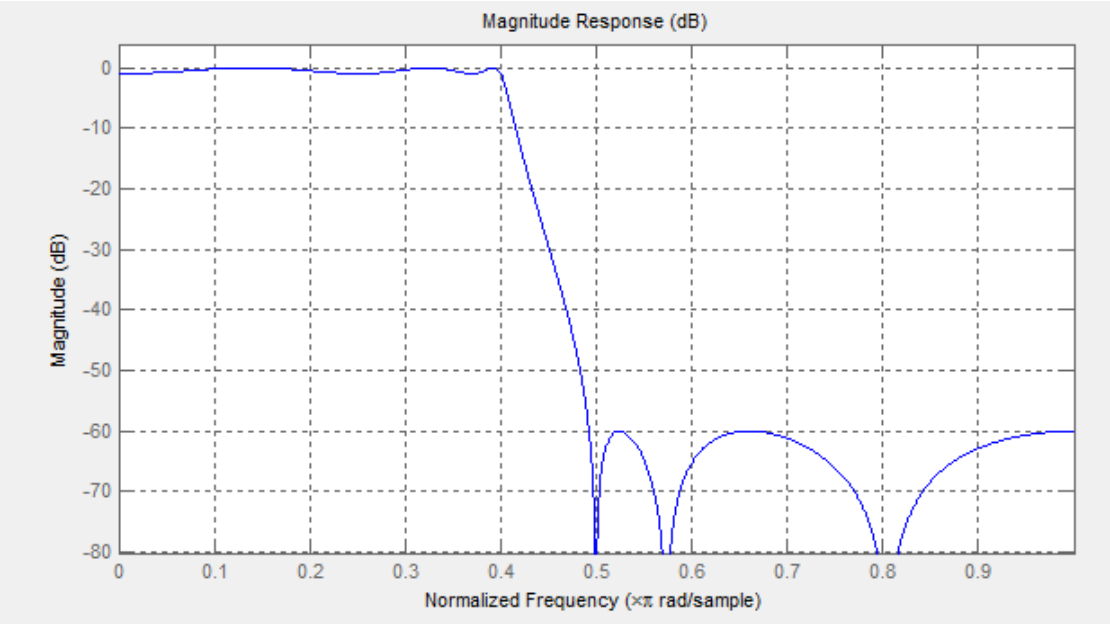
```
%forma directa I sos
[b,a] = ellip(6,1,60,.4);  % coeficientes del filtro
% s: coeficientes   g: ganancia de la sección
[s,g] = tf2sos(b,a);       % Convierte a SOS
```

```
s =
```

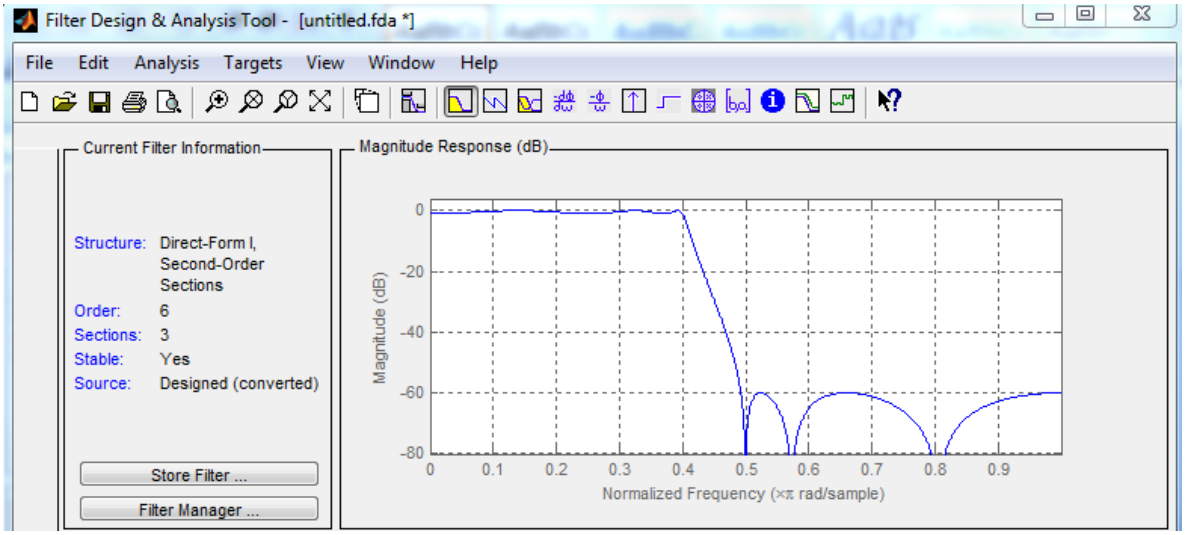
Numeradores			Denominadores			
1.0000	1.6355	1.0000	1.0000	-1.2134	0.4706	Sección 1
1.0000	0.4551	1.0000	1.0000	-0.8266	0.7254	Sección 2
1.0000	-0.0023	1.0000	1.0000	-0.5982	0.9248	Sección 3

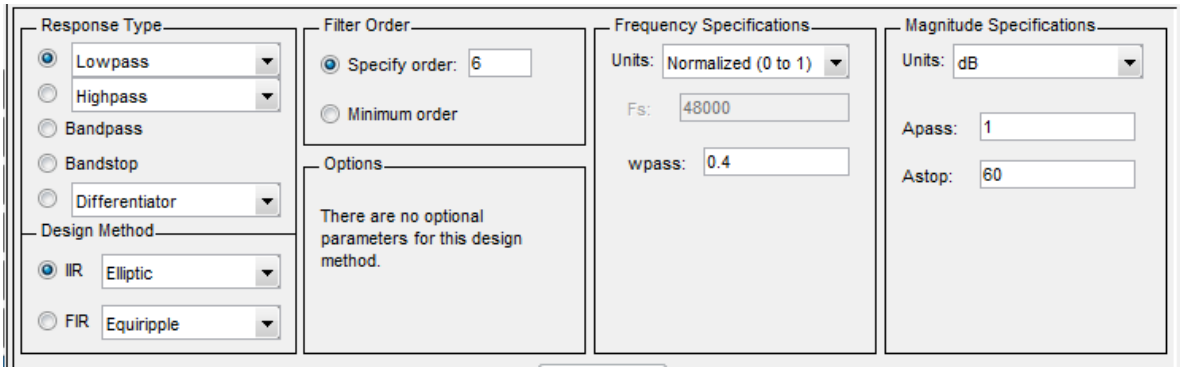
```
g =

    0.0153
```

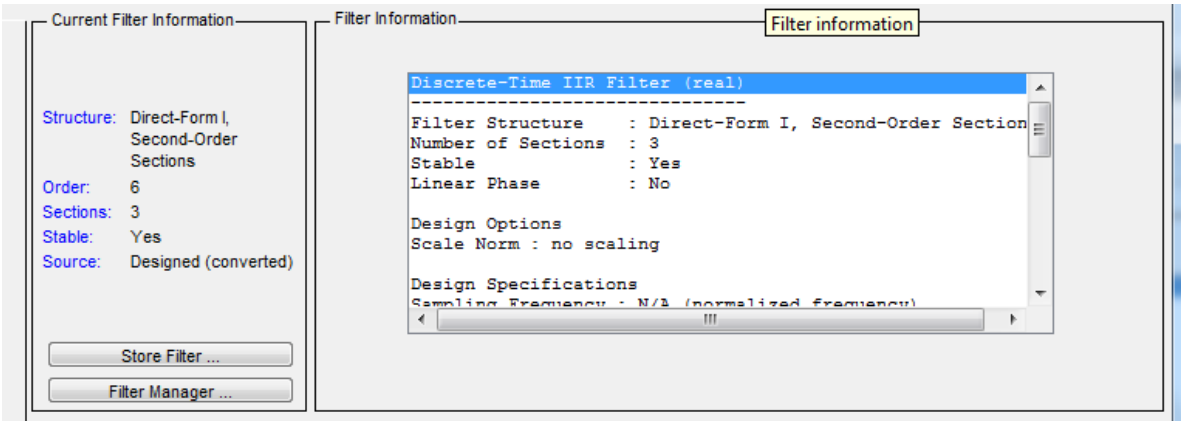
```
Hd=dfilt.df1sos(s,g);
get(Hd,'sosmatrix');

fvtool(Hd)
realizemdl(Hd)
```

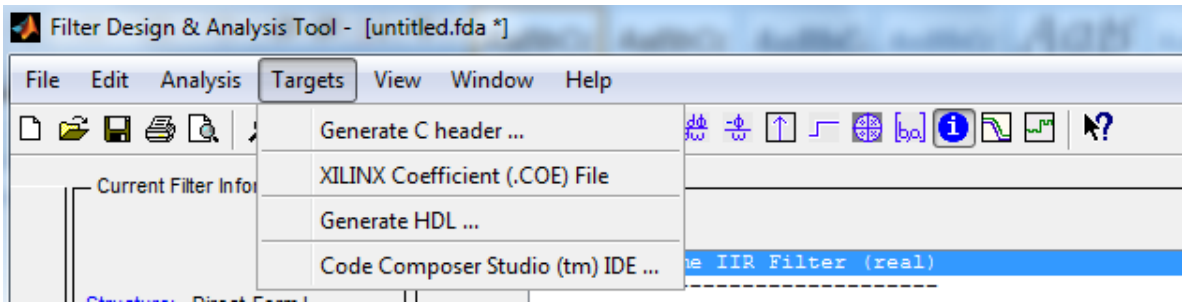


Utilizando fdatool





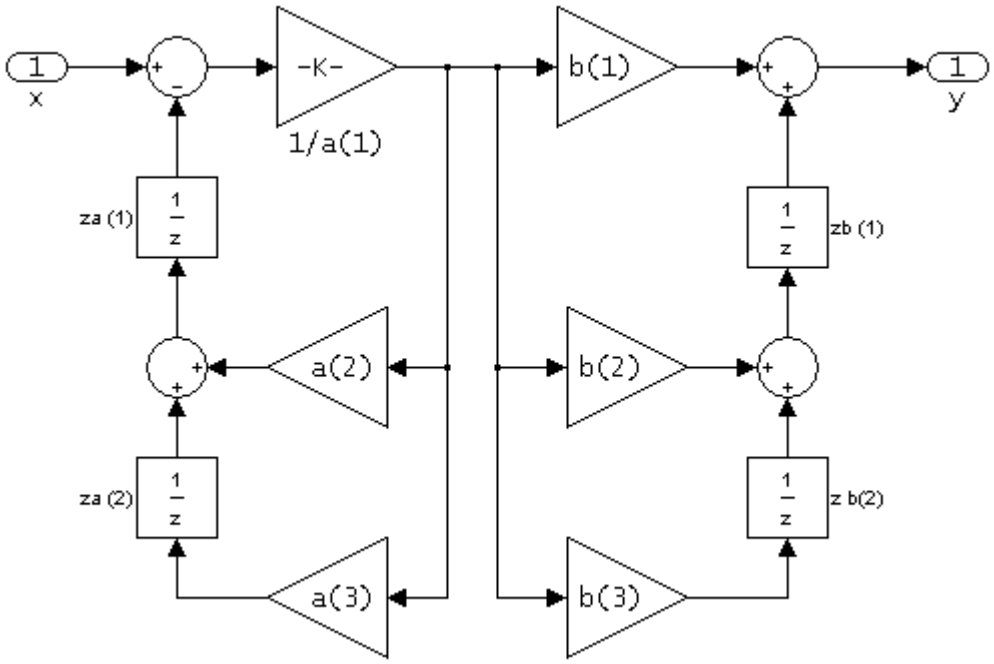





```
/*
 * Filter Coefficients (C Source) generated by the Filter Design
and Analysis Tool
 *
 * Generated by MATLAB(R) 7.11 and the Signal Processing Toolbox
6.14.
 *
 * Generated on: 15-Apr-2016 11:17:22
 *
 */

/*
 * Discrete-Time IIR Filter (real)
 * -------------------------------
 * Filter Structure    : Direct-Form I, Second-Order Sections
 * Number of Sections  : 3
 * Stable              : Yes
 * Linear Phase        : No
 */

/* General type conversion for MATLAB generated C-code  */
#include "tmwtypes.h"
/*
```

```
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2010b\extern\include\tmwtypes.h
 */
#define MWSPT_NSEC 7
const int NL[MWSPT_NSEC] = { 1,3,1,3,1,3,1 };
const real64_T NUM[MWSPT_NSEC][3] = {
  {
     0.7758727937989,                0,                0
  },
  {
                  1,   0.4551150565578,                1
  },
  {
     1.631694765106,                0,                0
  },
  {
                  1,    1.635472321101,                1
  },
  {
    0.01210690350844,                0,                0
  },
  {
                  1,-0.002268039834126,                1
  },
  {
                  1,                0,                0
  }
};
const int DL[MWSPT_NSEC] = { 1,3,1,3,1,3,1 };
const real64_T DEN[MWSPT_NSEC][3] = {
  {
                  1,                0,                0
  },
  {
                  1,  -0.8266491310456,   0.7253983320208
  },
  {
                  1,                0,                0
  },
  {
                  1,   -1.213444837777,   0.4706346277503
  },
  {
                  1,                0,                0
  },
  {
                  1,  -0.5982288101164,   0.9248294227288
  },
  {
                  1,                0,                0
  }
};
```

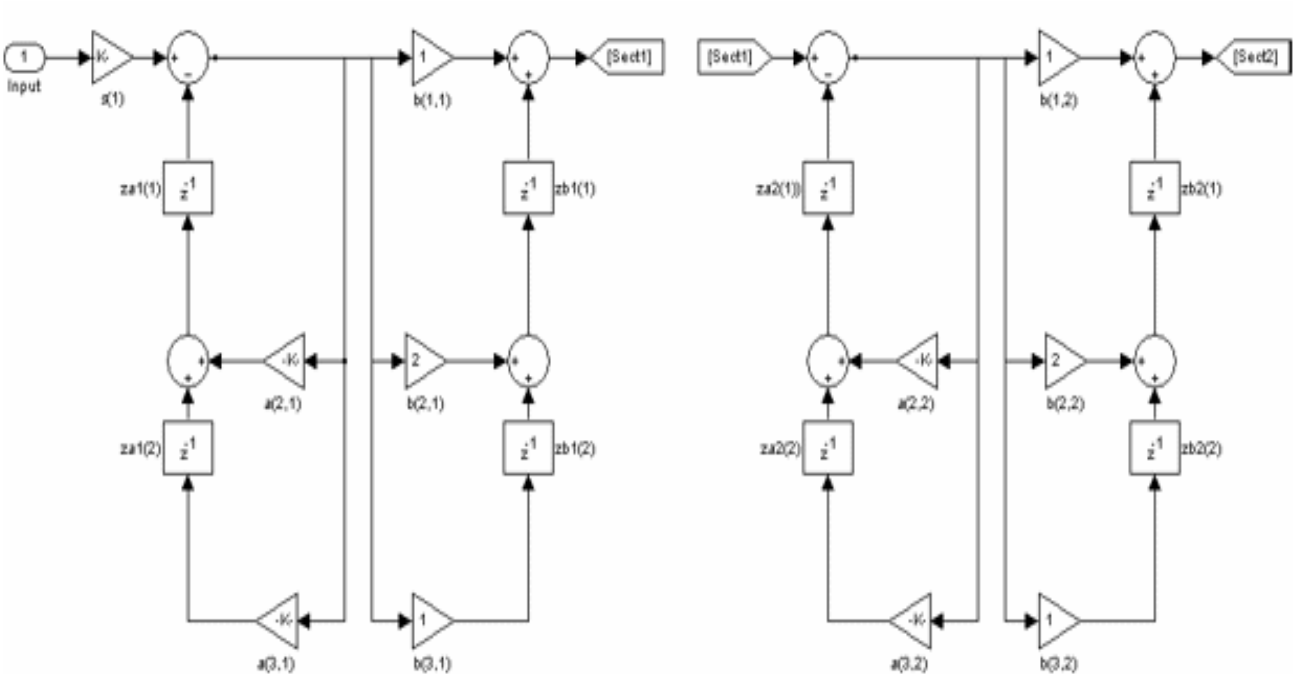
c) Forma Directa I Transpuesta: dfilt.df1t



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)
realizemdl(Hd)
```

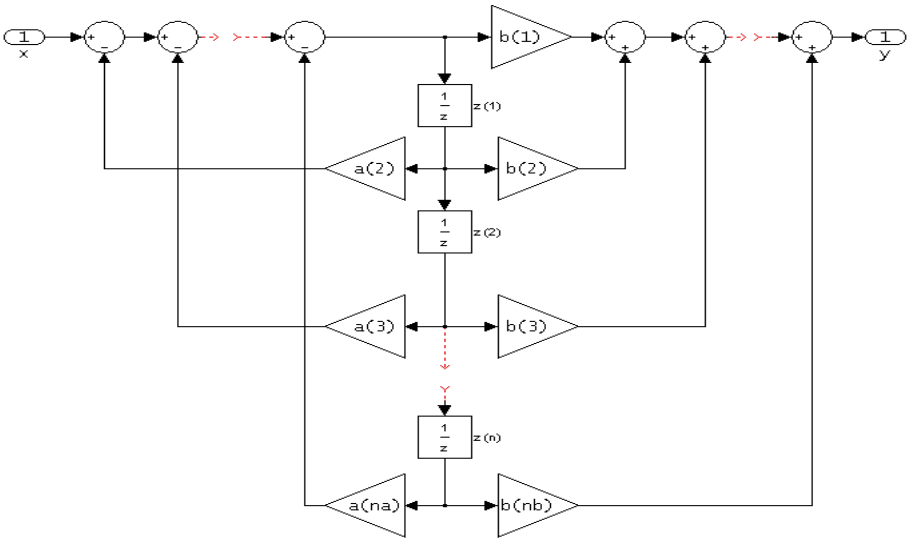
d) Forma Directa I Transpuesta sos: dfilt.df1tsos



Ejemplo

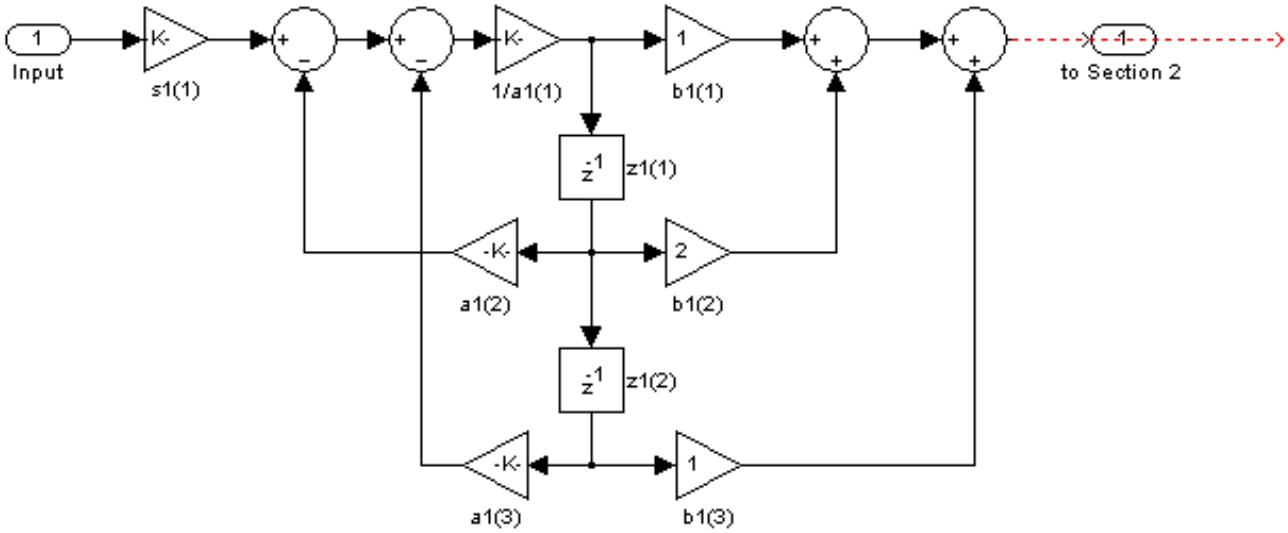
```
%forma directa I sos
[b,a] = ellip(6,1,60,.4);  % coeficientes del filtro
% s: coeficientes   g: ganancia de la sección
[s,g] = tf2sos(b,a);       % Convierte a SOS
Hd=dfilt.df1tsos(s,g);
get(Hd,'sosmatrix')
```

e) Forma Directa II: dfilt.df2



Ejemplo:

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



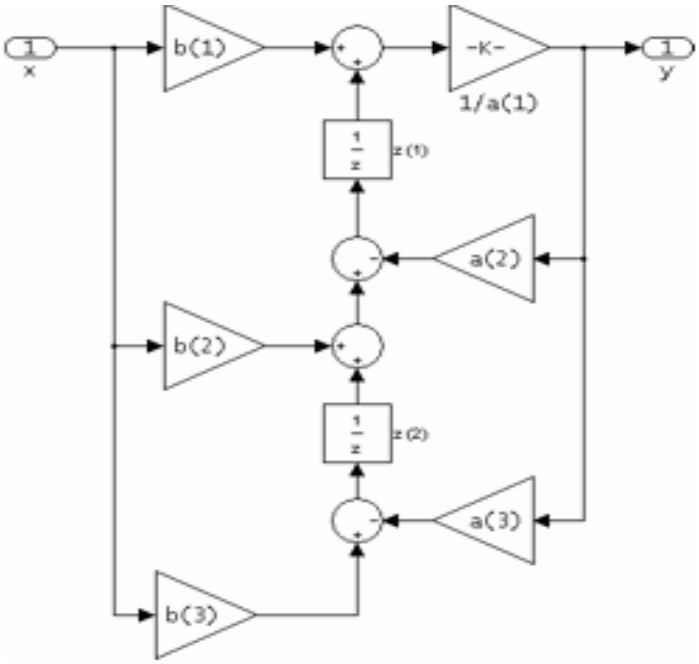
Ejemplo:

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

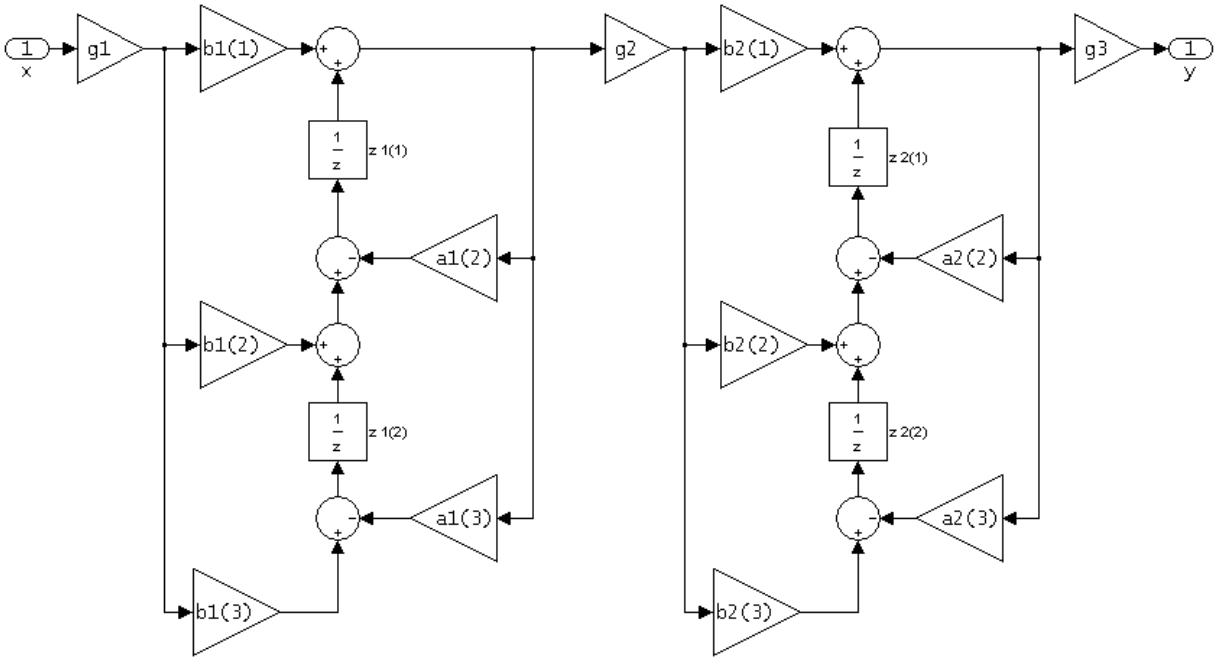
f) Forma Directa II Transpuesta: dfilt.df2t

Ejemplo:

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



g) Forma Directa II Transpuesta sos: dfilt.df2tsos



Ejemplo

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