**Trabajo Práctico 7**

**Filtros Digitales**

**EJERCICIO 1:**

***Filtro con cuatro polos, ubicados en (0.95,45±), (0.95,-45±), (0.95,45±), (0.95,-45±) y cuatro ceros ubicados en (0.80,30±), (0.80,-30±), (0.80,60±), (0.80,-60±) (coordenadas polares, ángulos en grados).***

***a) Genere el diagrama de polos y ceros en el plano Z***

Obtenemos las posiciones en el plano complejo de los polos y ceros:

%polos

radio\_polo =0.95;

p(1)= radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(2)= radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

p(3)= radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(4)= radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

%ceros

radio\_cero =0.80;

c(1)= radio\_cero \*(cos(pi/6) + j\*sin(pi/6));

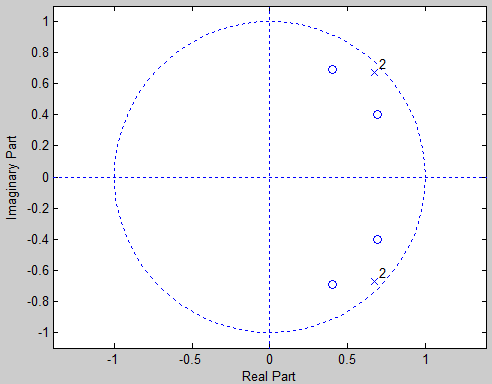
c(2)= radio\_cero \*(cos(-pi/6)+ j\*sin(-pi/6));

c(3)= radio\_cero \*(cos(pi/3) + j\*sin(pi/3));

c(4) =radio\_cero \*(cos(-pi/3)+ j\*sin(-pi/3));

figure(1)

zplane(c',p');



***b) Encuentre, evalúe y grafique la respuesta en frecuencia del filtro entre 0 y 2.***

Para evaluar la respuesta en frecuencias, primero obtenemos el polinomio al cual pertenecen esas raíces (de la figura 1).

b = poly(c);

a = poly(p);

fm=1000

y utilizamos la función freqz para obtener la respuesta en frecuencias entre 0-

r=freqz(b,a,fm/2,fm);

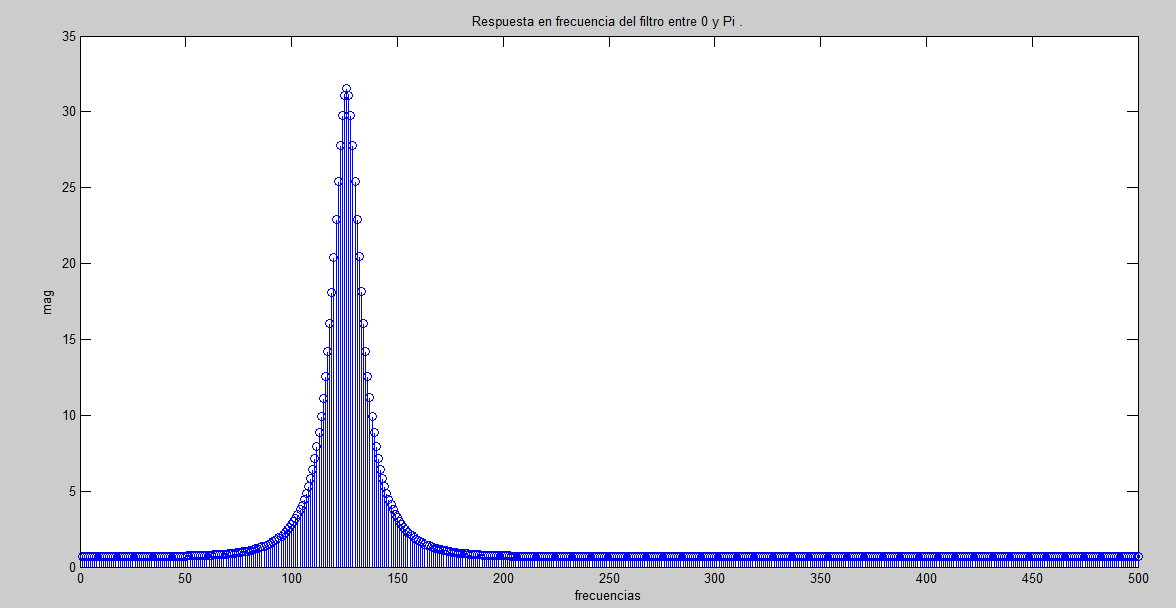
figure('name','Ejercicio 1 - item b');

stem(abs(r));

xlabel('frecuencias')

ylabel('mag')

title('Respuesta en frecuencia del filtro entre 0 y Pi .')



***c) Normalice los coeficientes del filtro, de manera que el valor máximo de la respuesta en frecuencia sea 1.***

Para realizar esto dividimos todo el vector por el valor máximo de frecuencia obtenido.

b=poly(c);

a=poly(p);

fm=1000

figure('name','Ejercicio 1 - item c');

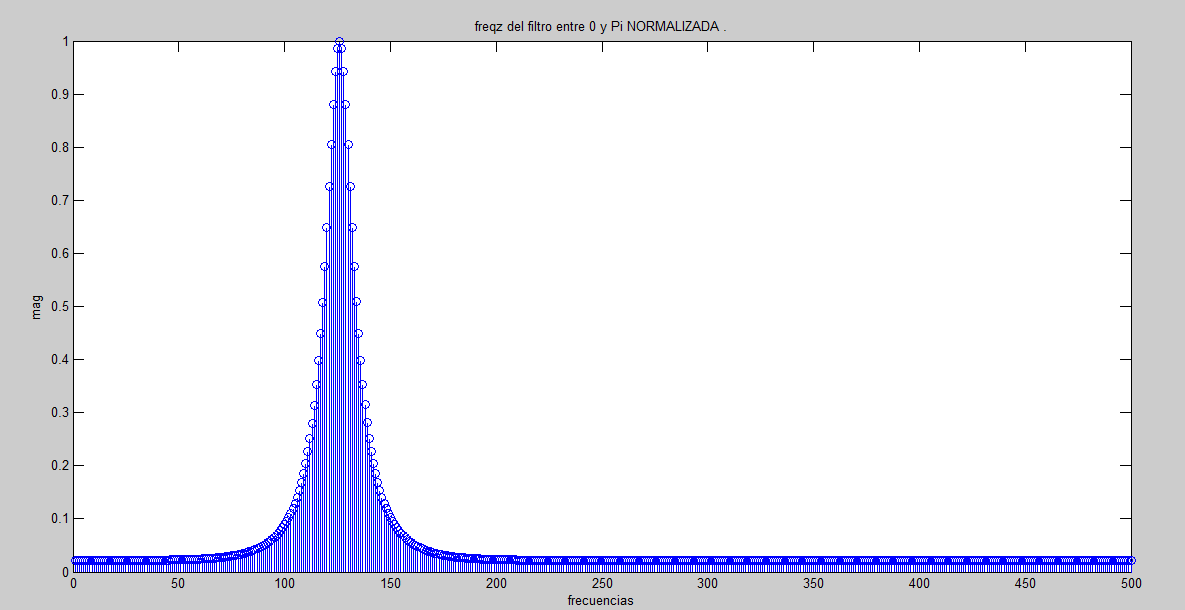
h=freqz(b,a,fm/2,fm) / max(freqz(b,a,fm/2,fm));

stem(abs(h));

xlabel('frecuencias')

ylabel('mag')

title('freqz del filtro entre 0 y Pi NORMALIZADA .')



***d) Modifique el radio de los polos (manteniendo los respectivos conjugados) y observe, en la gráfica, como cambia la respuesta en frecuencia.***

%polos con radio 0.7

radio\_polo2=0.7;

fm=1000;

p2(1)=radio\_polo2 \*(cos(pi/4)+sin(pi/4)\*i);

p2(2)=radio\_polo2 \*(cos(-pi/4)+sin(-pi/4)\*i);

p2(3)=radio\_polo2 \*(cos(pi/4)+sin(pi/4)\*i);

p2(4)=radio\_polo2 \*(cos(-pi/4)+sin(-pi/4)\*i);

%polos con radio 5

radio\_polo3=5;

p3(1)=radio\_polo3 \*(cos(pi/4)+sin(pi/4)\*i);

p3(2)=radio\_polo3 \*(cos(-pi/4)+sin(-pi/4)\*i);

p3(3)=radio\_polo3 \*(cos(pi/4)+sin(pi/4)\*i);

p3(4)=radio\_polo3 \*(cos(-pi/4)+sin(-pi/4)\*i);

%transformo a polinomio

b=poly(c);

a1=poly(p);

a2=poly(p2);

a3=poly(p3);

% calculo respuesta en frecuencia

rf1=freqz(b,a1,fm/2,fm);

rf2=freqz(b,a2,fm/2,fm);

rf3=freqz(b,a3,fm/2,fm);

figure('name','Ejercicio 1 - item d- Zplane');

subplot(3,1,1);zplane(c',p');title('polos r=0.95 - Radio Original del Ejercicio')

subplot(3,1,2);zplane(c',p2');title('polos r=0.7')

subplot(3,1,3);zplane(c',p3');title('polos r=5')

%grafico para comparar

figure('name','Ejercicio 1 - item d');

subplot(3,1,1); stem(abs(rf1));

xlabel('frecuencias')

ylabel('mag')

title('Respuesta en frecuencia Modificando el Radio de los polos - r=0.95 - Radio Original del Ejercicio')

subplot(3,1,2); stem(abs(rf2));

xlabel('frecuencias')

ylabel('mag')

title('Respuesta en frecuencia Modificando el Radio de los polos - r=0.7 ')

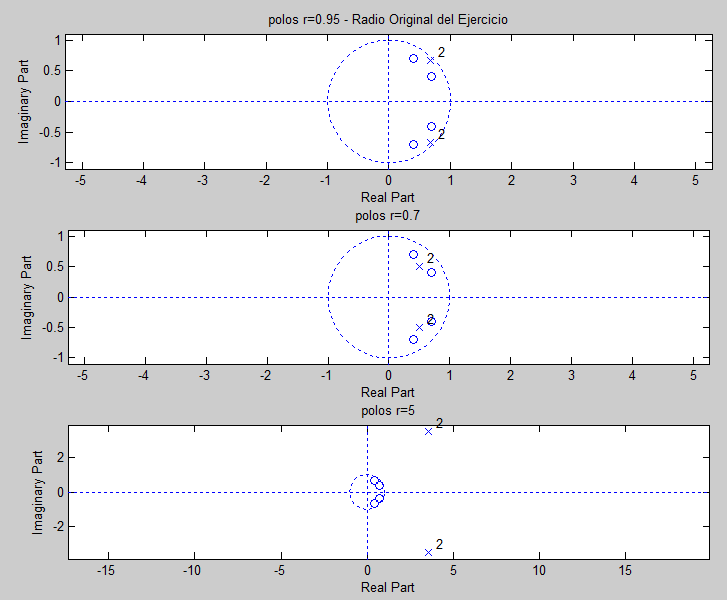
subplot(3,1,3); stem(abs(rf3));

xlabel('frecuencias')

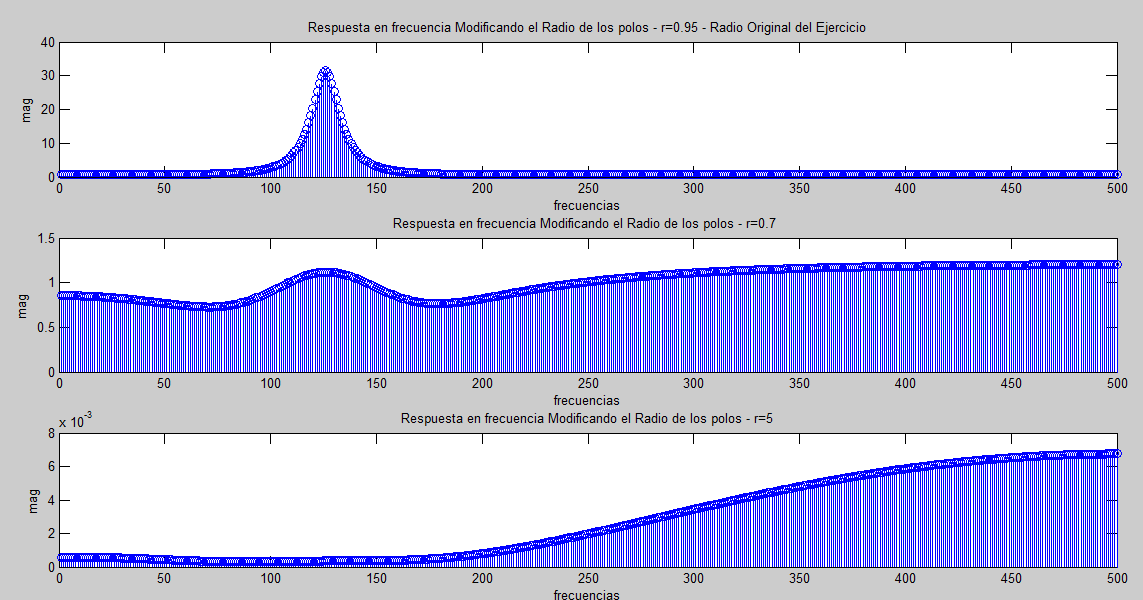
ylabel('mag')

title('Respuesta en frecuencia Modificando el Radio de los polos - r=5')

Diagrama de polos y ceros - zplane



Respuesta en frecuencia (rango 0 –pi)



En la gráfica podemos observar como al reducir el radio de los polos (figura del medio) reducimos la influencia que ejercen sobre la respuesta en frecuencias (circunferencia del círculo unitario), lo que provoca que la amplitud se reduzca y además la influencia de los ceros también cambia por la nueva posición relativa con respecto de los polos.

En la siguiente gráfica observamos como al aumentar el radio de los polos por sobre el círculo unitario el sistema se vuelve inestable.

Por todo esto es que la ubicación de los polos es fundamental en el desarrollo de filtros.

***e) Este filtro está diseñado de manera de que cuando se utiliza con señales muestreadas a 200 Hz la banda de paso se centre en 25 Hz. Para comprobar esto, genere una señal sumando dos senoidales de 15 Hz y 25 Hz, para luego filtrarla con el filtro normalizado. Grafique la señal original, la señal filtrada y sus espectros, y analice el resultado.***

Construimos la señal senoidal como la suma de otras dos senoidales de frecuencia 15hz y 25hz respectivamente. Y lo hago con una frecuencia de muestreo de 200hz durante 1 seg.

%construyo la señal senoidal muestreada a 200hz

fm=200;

dt=1/fm;

t\_tot=1;

t=[0:dt:t\_tot-dt];

s=sin(2\*pi\*t\*15) + sin(2\*pi\*t\*25);

Construimos el filtro normalizado como lo realizamos anteriormente:

%polos

radio\_polo = 0.95;

p(1) = radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(2) = radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

p(3) = radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(4) = radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

%ceros

radio\_cero = 0.80;

c(1) = radio\_cero \*(cos(pi/6) + j\*sin(pi/6));

c(2) = radio\_cero \*(cos(-pi/6)+ j\*sin(-pi/6));

c(3) = radio\_cero \*(cos(pi/3) + j\*sin(pi/3));

c(4) = radio\_cero \*(cos(-pi/3)+ j\*sin(-pi/3));

%transformo a polinomio

b = poly(c);

a = poly(p);

%normalizo los coeficientes

r=freqz(b,a,fm/2,fm); %respuesta en frecuencia 0,pi sin normalizar

alfa=1/max(abs(r)); %factor de ponderacion

b=b\*alfa; %normalizo

Introducimos la senoidal al filtro y graficamos tanto la visión temporal como la frecuencial de la misma antes y después del proceso

%filtro la señal

sfil=filter(b,a,s);

figure('name','Ejercicio 1 - item e');

%grafica de la señal(antes y después de filtrarla):

subplot(4,1,1), plot(t,s);

xlabel('tiempo')

title('grafica de la señal antes de filtrarla');

subplot(4,1,2), plot(t,sfil);

xlabel('tiempo')

title('grafica de la señal despues de filtrarla');

%espectro en frecuencia de la señal (antes y después):

subplot(4,1,3),stem(abs(fft(s)));

xlabel('frecuencias')

ylabel('mag')

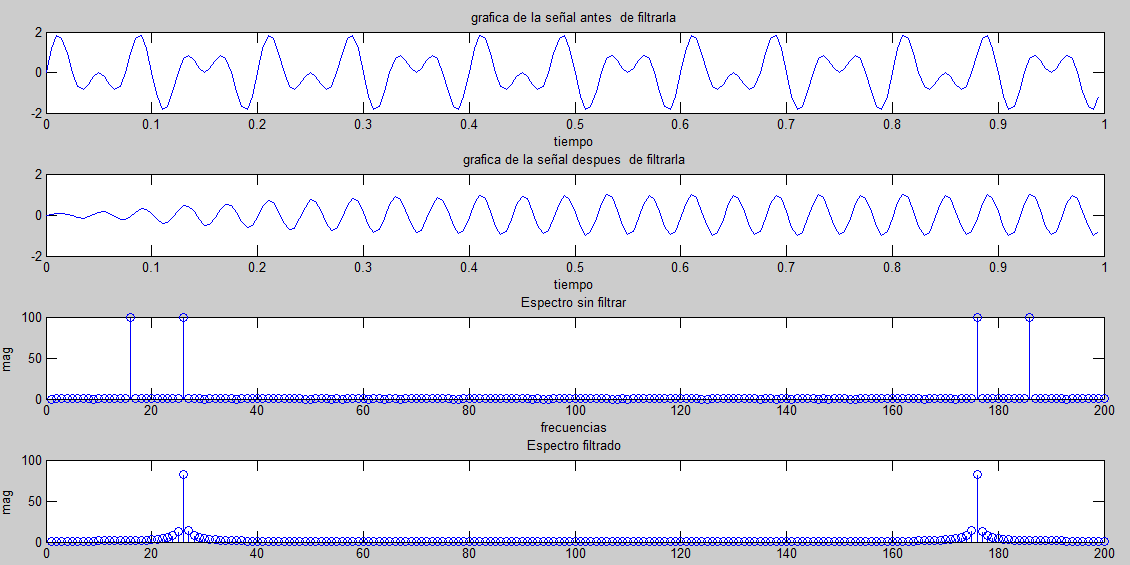
title('Espectro sin filtrar')

subplot(4,1,4),stem(abs(fft(sfil)));

xlabel('frecuencias')

ylabel('mag')

title('Espectro filtrado')



En la gráfica se observa como efectivamente se ha filtrado la señal senoidal de 15hz.

***f) Repita el ítem anterior pero esta vez genere la señal con una frecuencia de muestreo de 120 Hz. Compare el resultado con el caso anterior y obtenga conclusiones.***

Realizamos el mismo procedimiento que en el punto c) pero con la siguiente señal muestreada a 120hz durante 1 seg.

fm=120;

dt=1/fm;

t\_tot=1;

t=[0:dt:t\_tot-dt];

s=sin(2\*pi\*t\*15) + sin(2\*pi\*t\*25);

%diseño el filtro

%polos

radio\_polo =0.95;

p(1)= radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(2)= radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

p(3)= radio\_polo \*(cos(pi/4) + j\*sin(pi/4));

p(4)= radio\_polo \*(cos(-pi/4)+ j\*sin(-pi/4));

%ceros

radio\_cero =0.80;

c(1)= radio\_cero \*(cos(pi/6) + j\*sin(pi/6));

c(2)= radio\_cero \*(cos(-pi/6)+ j\*sin(-pi/6));

c(3)= radio\_cero \*(cos(pi/3) + j\*sin(pi/3));

c(4) =radio\_cero \*(cos(-pi/3)+ j\*sin(-pi/3));

%transformo a polinomio

b=poly(c);

a=poly(p);

%normalizo los coeficientes del polinomio

r=freqz(b,a,120); alfa=1/max(abs(r)); b=b\*alfa;

%filtro la señal

sfil=filter(b,a,s);

figure('name','Ejercicio 1 - item f');

%grafica de la señal(antes y después de filtrarla):

subplot(4,1,1), plot(t,s);

xlabel('tiempo')

title('grafica de la señal antes de filtrarla');

subplot(4,1,2), plot(t,sfil);

xlabel('tiempo')

title('grafica de la señal despues de filtrarla');

%espectro en frecuencia de la señal (antes y después):

subplot(4,1,3),stem(abs(fft(s)));

xlabel('frecuencias')

ylabel('mag')

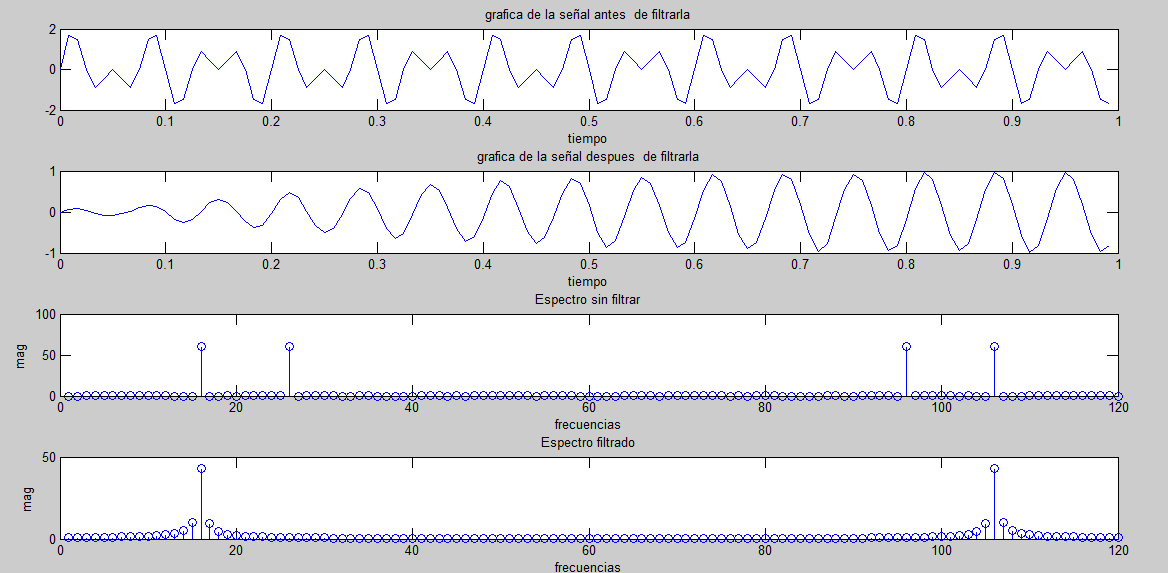
title('Espectro sin filtrar')

subplot(4,1,4),stem(abs(fft(sfil)));

xlabel('frecuencias')

ylabel('mag')

title('Espectro filtrado')



Esta vez es la otra componente frecuencial la que sobrevive al período de filtrado, esto se debe al cambio en la frecuencia de muestreo ya que se mantiene el mismo espectro entre 0 y fm/2, solo que de estar entre 0 y 100hz en el ejercicio anterior ahora esta mapeado de 0 a 60hz por lo que la banda de paso ahora se centra en 15hz.

**EJERCICIO 2:**

***Diseñe un filtro pasa-altos de tipo Butterworth con frecuencia de corte 500 Hz. Para este ejercicio realice todos los pasos del proceso de diseño, comenzando por el diseño analógico y realizando la transformación en frecuencia y la transformación conforme. Para obtener el filtro digital correspondiente, suponga que se procesarán señales con frecuencia de muestreo 2000 Hz. Utilice diferentes órdenes y compare los resultados graficando las respuestas en frecuencia.***

A partir de un orden dado N obtenemos los coeficientes qi del filtro analógico de Butterworth, obteniendo una expresión como la que se encuentra a continuación:



Luego sacamos las raíces del polinomio del denominador, haciendo r = roots(q) donde q es el vector con los coeficientes del polinomio. Obtenemos:



Aplicamos a cada cero del denominador (ri) la siguiente transformación para convertirlo en un filtro pasa alto:

Aplicamos la transformación bilineal para mapear al plano z:





Con un poco de trabajo algebraico:



Con lo cual obtenemos los coeficientes de la función de transferencia para un cero ri



Luego hacemos una convolución entre los resultados obtenidos para cada raíz **ri**, (lo que equivale a la multiplicación de los polinomios antes vistos), esto es, convolucionar cada coeficiente **a** con los restantes, obteniendo los coeficientes finales **A** del filtro diseñado y cada coeficiente **b** con los restantes, teniendo los coeficientes finales **B**.

Para realizarlo en matlab ejecutamos el siguiente código:

%datos del problema

orden = 6;

fm = 2000;

fcorte = 2\*pi\*500;

T = 1/fm;

%dependiendo del orden se seleccionan los coeficientes del denominador

%del polinomio de la función de transferencia analógica H(s)

switch(orden)

case 1

q = [1 1];

case 2

q = [1 sqrt(2) 1];

case 3

q = [1 2 2 1];

case 4

q = [1 2.6161 3.4142 2.6131 1];

case 5

q = [1 3.2361 5.2361 5.2361 3.2361 1];

case 6

q = [1 3.8637 7.4641 9.1416 7.4641 3.8637 1];

end

% calculamos las raíces del polinomio

p = roots(q);

n = length(p);

%A y B coeficientes “finales” del filtro

A = [1];

B = [1];

for i=1:n

%calculamos los coeficientes a y b de la función de transferencia para cada raíz

a(1) = fcorte \* T - 2 \* p(i);

a(2) = fcorte \* T + 2 \* p(i);

b(1) = 2;

b(2) = -2;

%luego vamos convolucionando con A y B para obtener los coeficientes

%finales del filtro

A = conv(A,a);

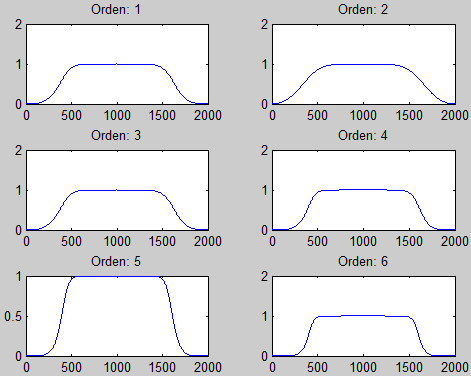
B = conv(B,b);

end

%graficamos

respfreq6 = freqz(B,A,2000,'whole');

Si vamos modificando los valores de orden, obtenemos los distintos filtros, como se observa en la siguiente gráfica desde orden 1 hasta 6.



En esta gráfica queda claramente demostrado como al elevar el orden del filtro se obtienen caídas más abruptas como así también bandas de transición más pequeñas. Es decir, el filtro se aproxima más al filtro ideal.

**EJERCICIO 3:**

***Diseñe un filtro pasa-banda con frecuencias de corte 2500 y 3000Hz. Para el diseño se requiere que la atenuación máxima en la banda de paso sea de 0,7 dB y la atenuación mínima en la banda de rechazo sea de 55dB. Considere además que las bandas de transición no deberían ser mayores a 200 Hz y la frecuencia de muestreo de las señales a procesar será de 10kHz. Compare los resultados obtenidos para filtros de Butterworth, Chebyshev tipos I y II, y para filtros elípticos. Compare los resultados obtenidos para los mismos filtros pero con el menor de todos los órdenes obtenidos anteriormente.***

Diseñamos los distintos filtros con los datos del enunciado:

%Filtro Butterworth (orden obtenido = 13)

[Nbut,Wnbut]= buttord([2500/5000 3000/5000],[2300/5000 3200/5000],0.7, 55);

[Bbut, Abut] = butter(Nbut, Wnbut);

%Filtro Chevyshev I (orden obtenido = 7)

[Nch1,Wnch1]=cheb1ord([2500/5000 3000/5000],[2300/5000 3200/5000],0.7,55);

[Bcheb1,Acheb1] = cheby1(Nch1, 0.7, Wnch1);

%Filtro Chebyshev II (orden obtenido = 7)

[Nch2,Wnch2]=cheb2ord([2500/5000 3000/5000],[2300/5000 3200/5000],0.7, 55);

[Bcheb2, Acheb2] = cheby2(Nch2, 55, Wnch2);

%Filtro Eliptico(orden obtenido = 5)

[Neli,Wneli]=ellipord([2500/5000 3000/5000],[2300/5000 3200/5000],0.7,55);

[Belip ,Aelip] = ellip(Neli, 0.7, 55, Wneli);

subplot(4,1,1)

plot(abs(freqz(Bbut,Abut,5000)));

title('Filtro Butterworth')

subplot(4,1,2)

plot(abs(freqz(Bcheb1,Acheb1,5000)));

title('Filtro Chevyshev I')

subplot(4,1,3)

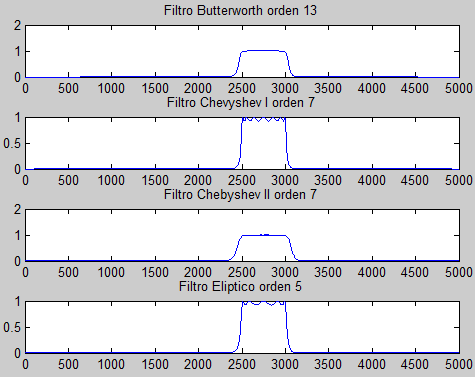
plot(abs(freqz(Bcheb2,Acheb2,5000)));

title('Filtro Chebyshev II')

subplot(4,1,4)

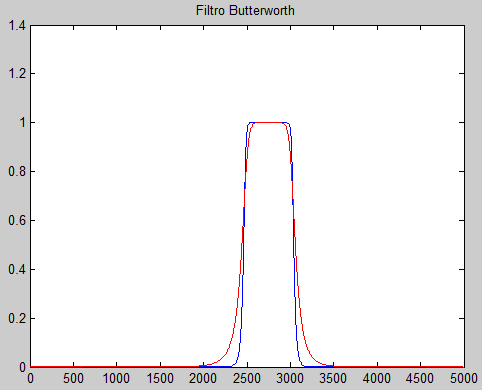
plot(abs(freqz(Belip,Aelip,5000)));

title('Filtro Eliptico')

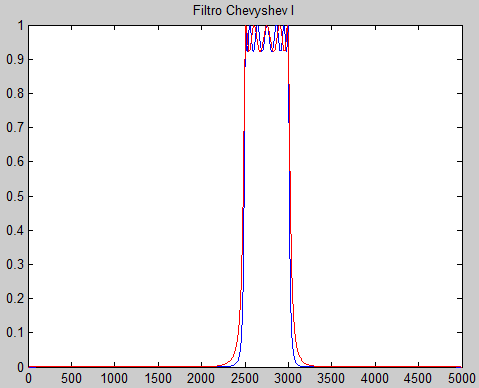


En la gráfica se observa que todos los filtros cumplen con las características pedidas y se observan las características distintivas de cada uno (rizado, pendientes más o menos pronunciadas, etc).

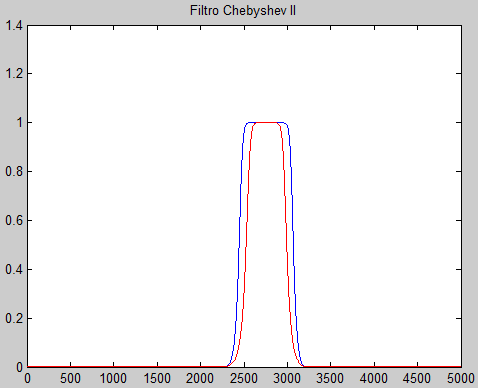
Realizando todos los filtros con orden 5 (el más pequeño obtenido) y los graficamos superpuestos con el de orden correspondiente (el dado por las funciones anteriores como el mínimo para cumplir los requisitos dados) observamos que obviamente el filtro Elíptico cumple las necesidades (5 era su orden requerido para realizarlo) pero los Chebyshev muestran cierta diferencia (necesitan 2 ordenes mas) y donde mas se hace presente la diferencia es el tipo Butterworth donde claramente ya no se cumplen los requisitos, esto se debe a que se encuentra a un orden muy inferior al requerido para cumplir con las condiciones establecidas inicialmente.



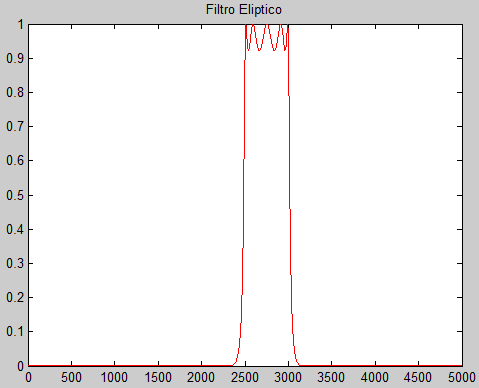
Butterworth: clara diferencia en la banda de paso



Chebyshev I: pequeña variación



Chebyshev II: variación más notoria de la banda de paso que el Chebyshev I



Elíptico: obviamente sigue cumpliendo, es el mismo.

**EJERCICIO 4:**

***Diseñe un filtro FIR mediante el método de ventanas (implementado por usted mismo), que permita eliminar el ruido de línea en una señal que fue muestreada a 300 Hz. Compare los resultados obtenidos con diferentes ventanas de truncado y diferentes cantidades de muestras en la respuesta al impulso.***

n = 300; %señal muestreada a 300hz (suponemos duracion 1 seg)

truncado = 100; %cantidad de muestras que tomo para el truncado temporal

ancho = 2; %ancho para eliminar componente de 50Hz (con tolerancias mas a menos 2)

fft\_f=ones(1,N);

elimino las muestras anterior y posterior a k = 51 (50hz) con el

ancho definido arriba

for i=-ancho:ancho,

fft\_f(51+i)=0; % frecuencia positiva

fft\_f(251+i)=0; % frecuencia negativa

end

%ploteo mi respuesta en frecuencia

figure('name','H[k] deseada');

plot(fft\_f);xlabel('frecuencia');ylabel('magnitug');title('Respuesta Frecuencia Filtro Deseado')

%APLICO LA TDF INVERSA

f=real(ifft(fft\_f)); %f esta en el tiempo

figure('name','ifft de H[k] -> h[n]');

plot(f);xlabel('muestras');title('antitransformada del filtro deseado') %se observa que la antitransformada es una sinc

%REALIZO EL TRUNCADO TEMPORAL

%f truncada

ftrunc=[f(1:truncado/2) f(N-truncado/2+1:N)]; %trunco en el tiempo

%MUESTRO DE [-fm/2 fm/2] CON FFTSHIFT

h=fftshift(ftrunc); %acomodo las muestras

figure('name','truncado temporal');

plot(h);xlabel('frecuencias');ylabel('magnitud'); title('Espectro de la transformada de la ventana cuadrada que se uso en el tiempo para truncar');

%VENTANAS

%ventana cuadrada --> dejo la respuesta en frecuencia como esta

%ventana de Hanning--###

windows1=h.\*hanning(length(h))';

%ventana de Hamming--####

windows2=h.\*hamming(length(h))';

%ventana de Blackman --####

windows3=h.\*blackman(length(h))';

whitebg([.5,.5,.5])

figure('name','ventana de Blackman');

subplot(2,1,1), plot(windows3,'y');title(['ventana de Blackman ',num2str(truncado),' muestras']); subplot(2,1,2), plot(abs(fft(windows3)),'y');

figure('name','ventana de Hamming');

subplot(2,1,1), plot(windows2,'y'); title(['ventana de Hamming ',num2str(truncado),' muestras']);subplot(2,1,2), plot(abs(fft(windows2)),'y');

figure('name','ventana de Hanning');

subplot(2,1,1), plot(windows1,'y');title(['ventana de Hanning ',num2str(truncado),' muestras']); subplot(2,1,2), plot(abs(fft(windows1)),'y');

figure('name','ventana cuadrada');

subplot(2,1,1), plot(h,'y');title(['ventana cuadrada ',num2str(truncado),' muestras']); subplot(2,1,2), plot(abs(fft(h)),'y');

**Respuesta el Frecuencia**

Fue diseñada con esta parte del scrip que aparece en la parte superior:

n = 300

ancho = 2;

fft\_f=ones(1,N);

elimino las muestras anterior y posterior a k = 51 (50hz) con el

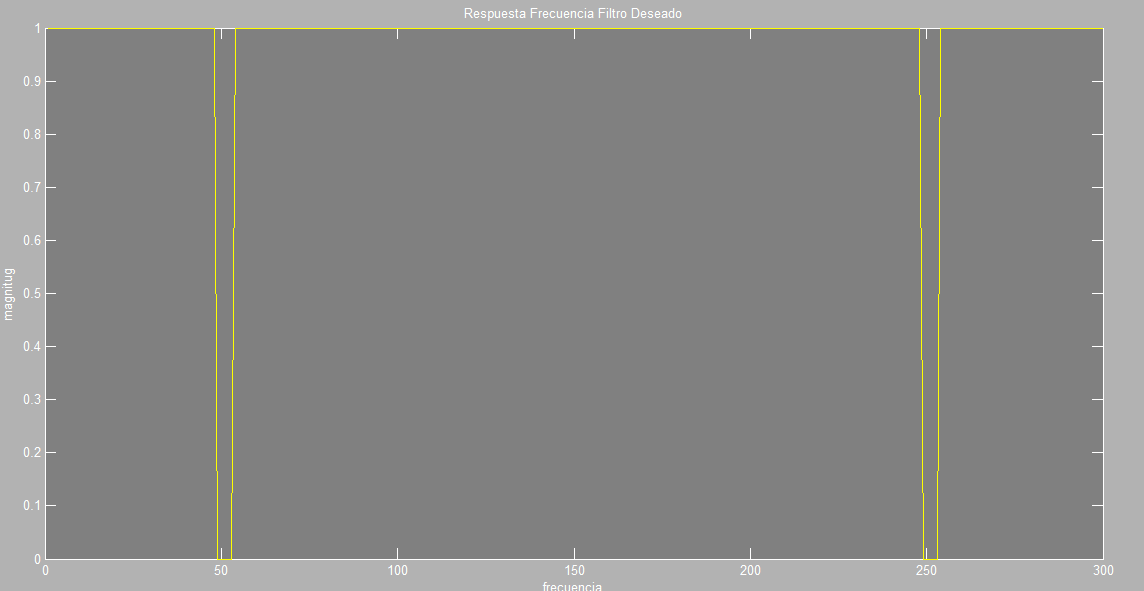
ancho definido arriba

for i=-ancho:ancho,

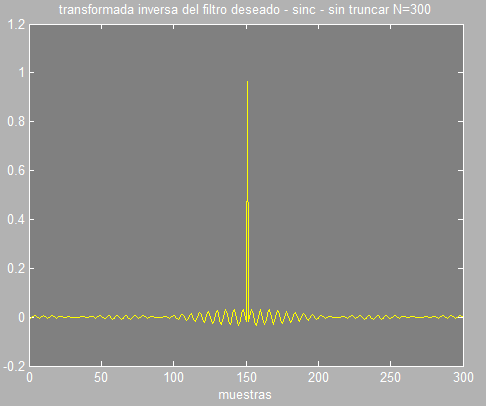
fft\_f(51+i)=0; % frecuencia positiva

fft\_f(251+i)=0; % frecuencia negativa

end



Transformada inversa de la Respuesta en frecuencia del filtro deseado, es una sinc:



Mi variable fft\_f esta en el dominio de la frecuencia entonces, la debo pasar el dominio del tiempo y realizar el truncado temporal:

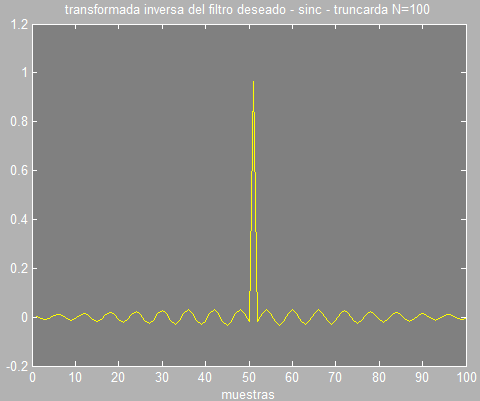
f=real(ifft(fft\_f)); %f esta en el tiempo

ftrunc=[f(1:truncado/2) f(N-truncado/2+1:N)]; %trunco en el tiempo

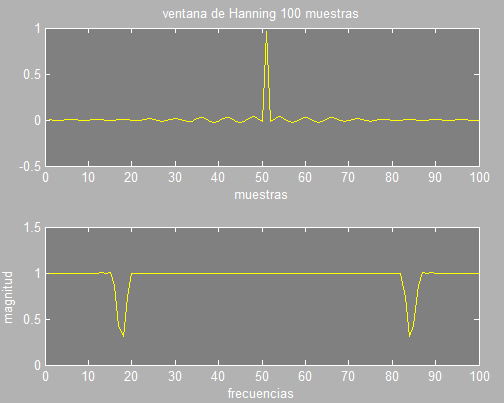
h=fftshift(ftrunc); %acomodo las muestras

plot(h);

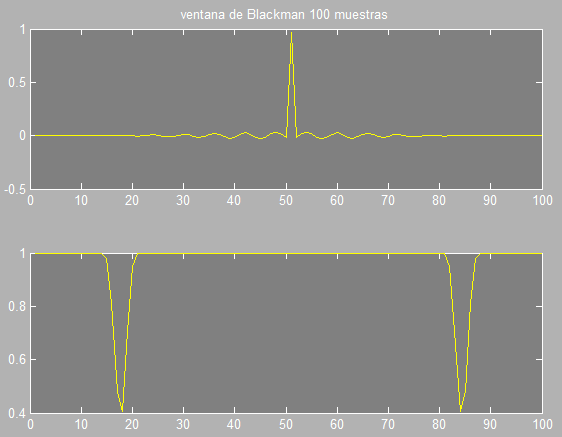
Grafico el mi sinc temporal(inverso de la respuesta en frecuencia del filtro deseado ) truncada en el tiempo a N=100 muestras

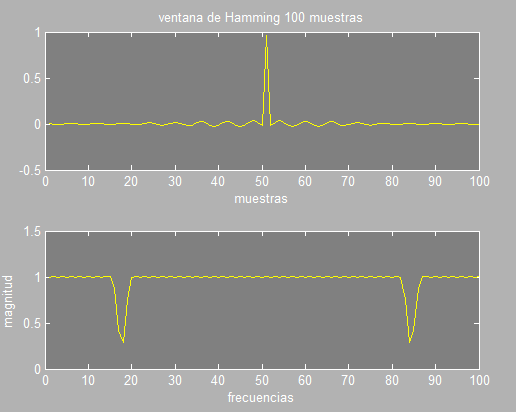


Se observa claramente un rizado en sus lóbulos laterales (característico de la ventana cuadrada en la frecuencia), los cuales me ingresan artefactos no deseados a mi señal cuando es convolucionada en la frecuencia contra ella.



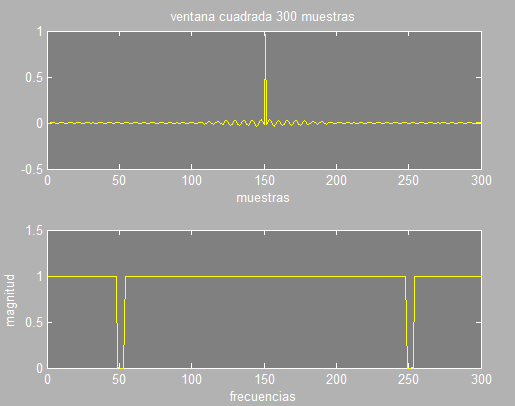
Aquí podemos observar como la variación de la energía de los lóbulos laterales hace que se note mucho menos el rizado que en el caso de la ventana cuadrada. Así mismo también varia levemente la cantidad a reducir la componente frecuencial a filtrar (50hz en este caso)

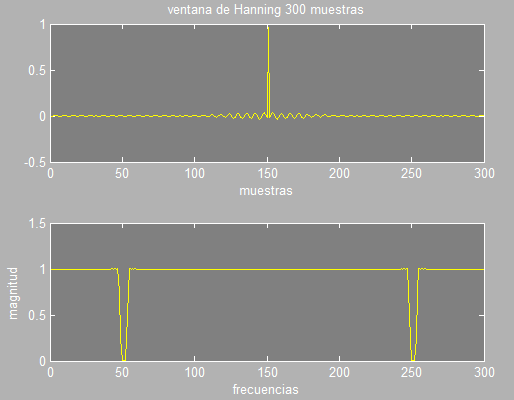


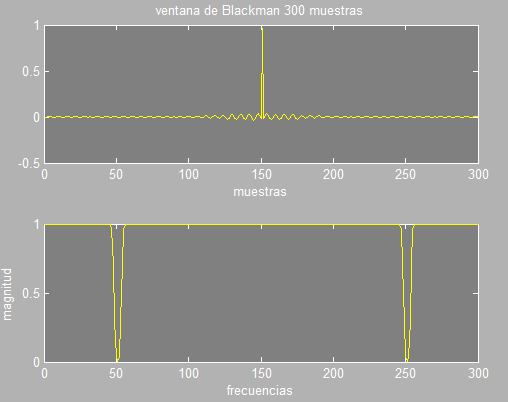


Ahora realizaremos las gráficas con un truncamiento temporal de 300 muestras, es decir suponiendo que la ventana tiene el mismo tamaño de la señal, implícitamente es como tener una ventana menos infinito a mas infinito:

truncado = 300; %cantidad de muestras (“no realizo truncado”)







En estas últimas gráficas observamos como se comporta el filtro sin hacerle truncado temporal