ELO 314 - Procesamiento Digital de Señales Laboratorio 3:Filtros Digitales en MatLab

Preparado por Juan Aguilera e-mail: Juan.Aguileraca@sansano.usm.cl
Cristóbal Huidobro e-mail: cristobal.huidobro@sansano.usm.cl

I. Efectos de polos y ceros en filtros FIR e IRR

1) Para encontrar la ecuación de diferencia inferimos una función de transferencia para un sistema causal. Ecuación que desarrollamos de la siguiente manera.

$$\begin{aligned} H(z) &= \frac{(z-e^{j\theta})(z-e^{-j\theta})}{z^2} \\ &= \frac{z^2 - z(e^{j\theta}+e^{-j\theta})+1}{z^2} \\ &= 1 - 2z^{-1}Cos(\theta) + z^{-2} / Z^{-1}\{\cdot\} \\ \Rightarrow y[n] &= x[n] - 2x[n-1]Cos(\theta) + x[n-2] \end{aligned}$$

Por lo que la respuesta a impulso corresponde a lo siguiente:

$$h[n] = \delta[n] - 2\delta[n-1]Cos(\theta) + \delta[n-2]$$

Implementando el filtro con $\theta = \pi/6$ tenemos el resultado siguiente.

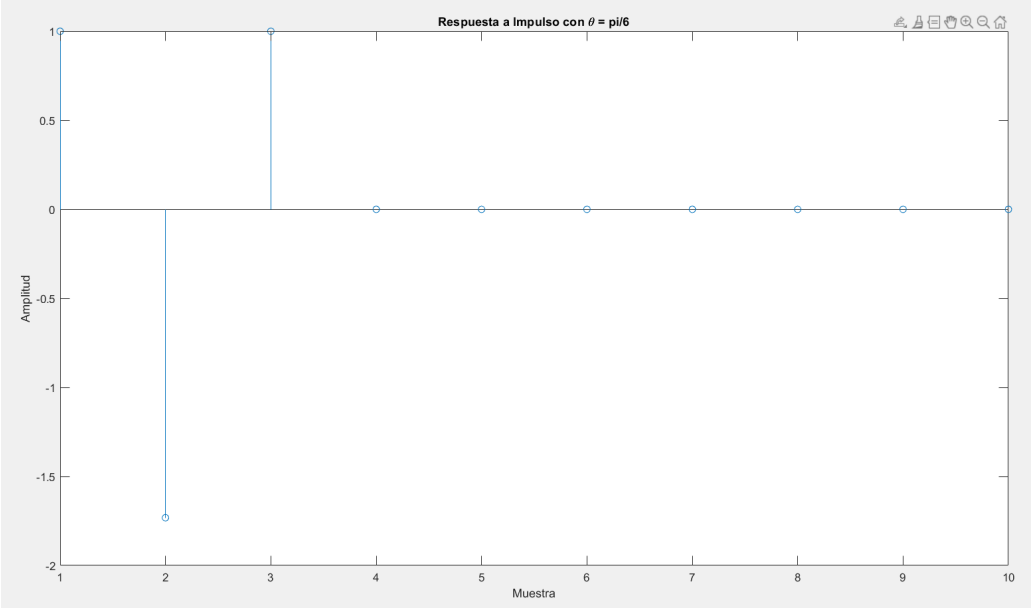

Figura 1: Respuesta impulso con θ = $\pi/6$.

Donde los coeficientes del filtro $[1 \ \ -2Cos(\theta) \ 1]$ coinciden con las magnitudes.
Notamos que se tiene tres muestras distintas de cero para $\theta \neq \frac{\pi(2n-1)}{2} \ \forall n = 1, 2, ..$, para los casos en que $Cos(\theta) = 0$ se tienen 2 muestras distintas de cero, es decir, el sistema tiene una respuesta impulso finita.

Para la expresión analítica tenemos en función de ω tenemos:

$$\begin{aligned} H(z) &= 1 - 2z^{-1}Cos(\theta) + z^{-2} \\ \Rightarrow H(\omega) = H(z = e^{j\omega}) &= \frac{e^{j2\omega} - 2e^{j\omega}Cos(\theta) + 1}{e^{j2\omega}} \end{aligned}$$

Cuyo gráfico se muestra a continuación

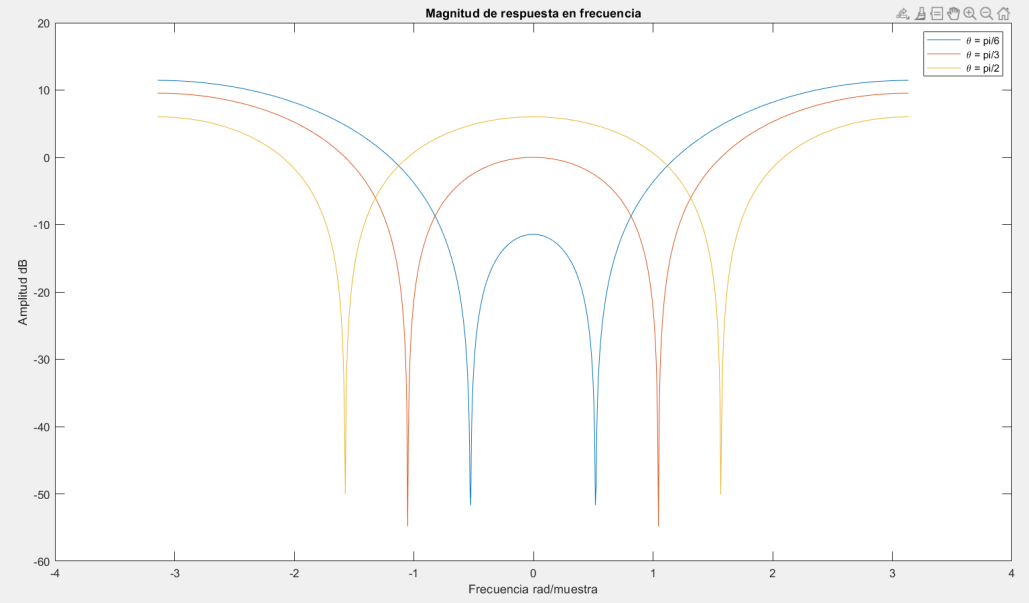

Figura 2: Magnitud Respuesta impulso para diferentes valores θ.

Podemos notar de la Figura 2, que la magnitud es disminuida donde $\omega = \theta$. Esto ocurre porque los ceros del filtros están ubicados en la frecuencia θ, por lo que la respuesta en frecuencia del filtro se anula en esos puntos. ω corresponde a la frecuencia normalizada en $rad/muestra$, la cual está dada por la relación $\omega = 2 * pi * (f_0/f_s)$. Donde f_0 Hz es la frecuencia filtrada.

2) Obtenemos la ecuación de diferencias del filtro descrito considerando un sistema causal como se muestra a continuación.

$$\begin{aligned} H(z) &= \frac{z^2(1-r)}{(z - re^{j\theta})(z - re^{-j\theta})} \\ &= \frac{z^2(1-r)}{z^2 - zr(e^{j\theta} + e^{-j\theta}) + r^2} \\ &= \frac{1-r}{1 - z^{-1}rCos(\theta) + r^2z^{-2}}\{\cdot\} \\ \Rightarrow y[n] &= (1-r)x[n] + 2ry[n-1]Cos(\theta) - r^2y[n-2] \end{aligned}$$

La gráfica de la respuesta a impulso se muestra en la Figura 3 para un ángulo $\theta = \pi/6$ y $r = 0{,}3$.
Obtenemos la función analítica en función de ω de la siguiente manera:

$$\begin{aligned} H(z) &= \frac{z^2(1-r)}{z^2 - zrCos(\theta) + r^2} \\ \Rightarrow H(\omega) = H(z = e^{j\omega}) &= \frac{e^{j2\omega}(1-r)}{e^{j2\omega} - e^{j\omega}rCos(\theta) + r^2} \end{aligned}$$

El gráfico de la respuesta en frecuencia para diferentes valores de r con $\theta = \pi/3$ se presenta en la Figura 4 donde podemos ver los efectos de r, notamos que al aumentarlo, la frecuencia de paso se acerca al valor deseado de θ_0. Esto ocurre porque los polos están ubicados dentro del circulo unitario, pero la respuesta en frecuencia es evaluada sobre el circulo unitario. Al aumentar r, al evaluar la respuesta en frecuencia en $e^{j\theta_0}$, el valor del polo $e^{j\theta_0} - re^{j\theta_0}$, se va acercando a 0, por lo que la ganancia aumenta.

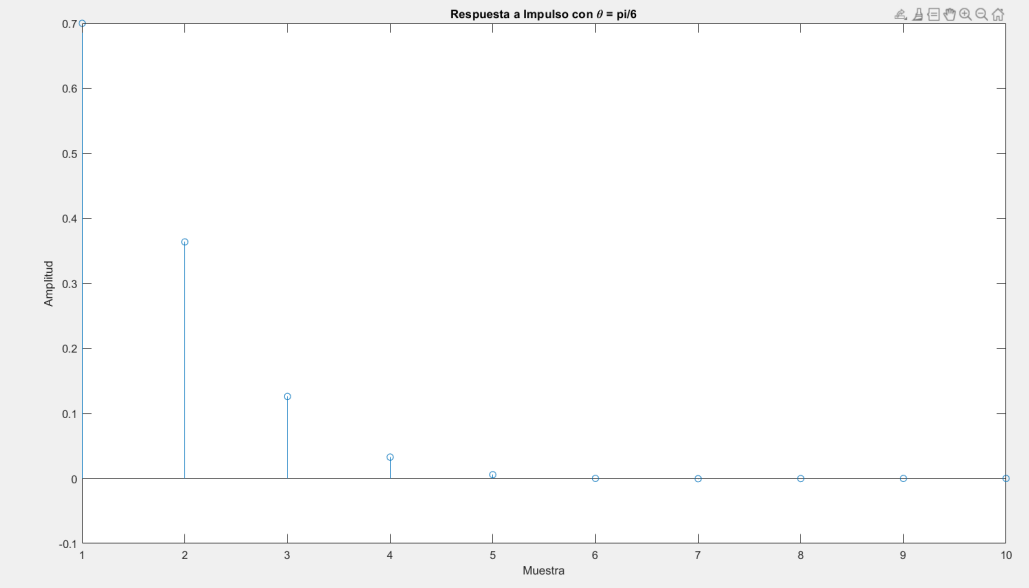

Figura 3: Respuesta impulso con $\theta = \pi/6$ y $r = 0{,}3$.

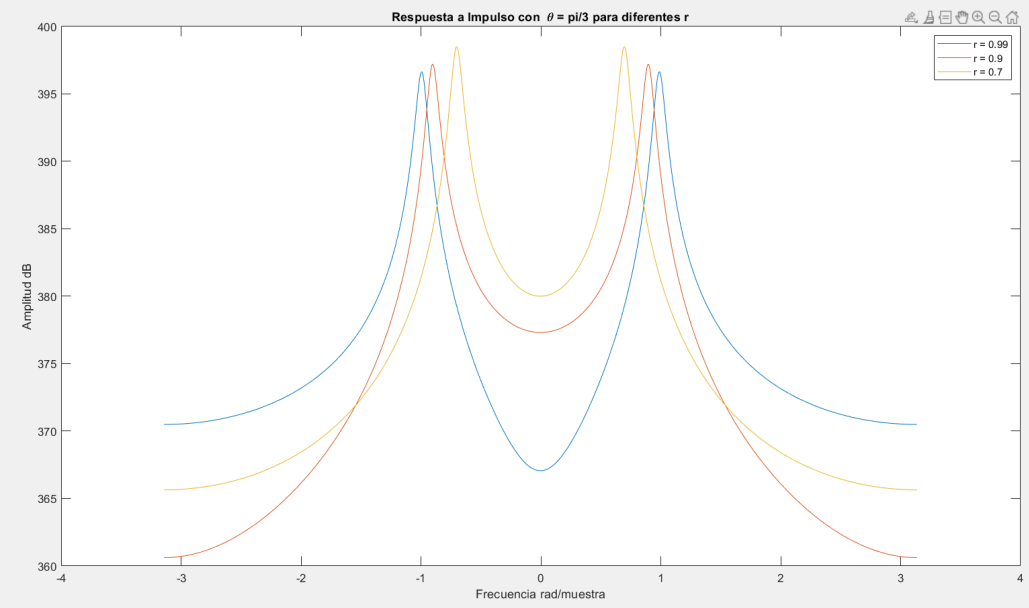

Figura 4: Magnitud Respuesta impulso para diferentes valores de r.

3) Para implementar el filtro FIR utilizamos el siguiente código:

```
function y = FIR_filter(x,theta)
b = [1 -2*cos(theta) 1];
a = [1 0 0];
h = impz(b,a);
y = conv(h,x);
end
```

Aplicamos el comando $plot\_fft\_mag$ a la señal $nspeech$ lo que se muestra en la Figura 5 donde rescatamos el tono a filtrar $frec = 1685\ Hz$

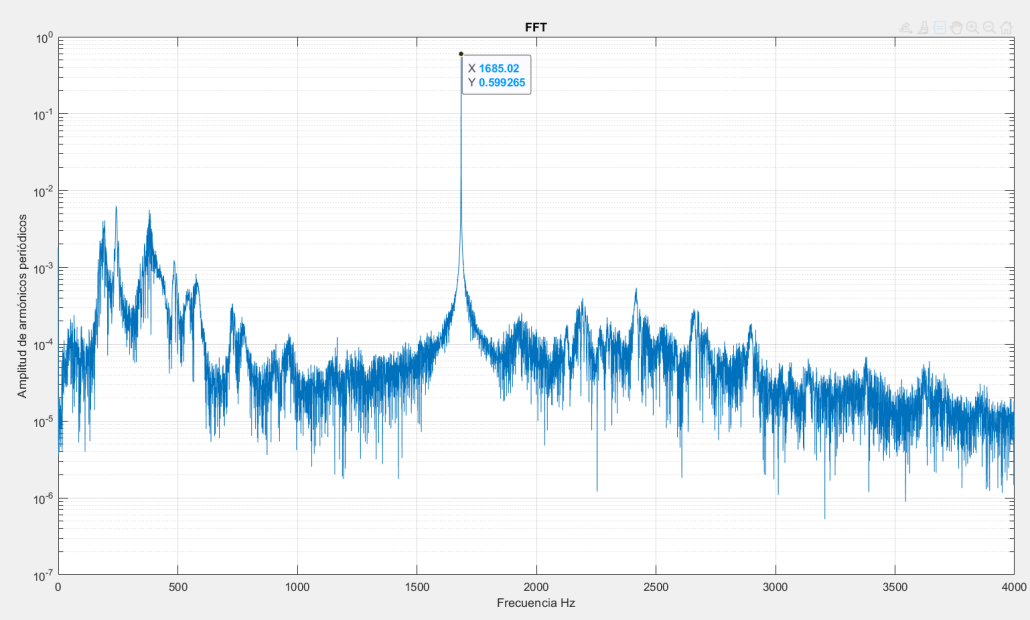

Figura 5: FFT de nspeech.

tenemos que la frecuencia de muestreo es $f_s = 8000\ Hz$ por lo que normalizamos la frecuencia a filtrar de la siguiente manera:

$$f_n = \frac{frec \cdot 2\pi}{f_s} = \frac{1685 \cdot 2\pi}{8000} = 1{,}3234\ rad/muestra$$

Al aplicar como entrada $nspeech$ junto a f_n al filtro FIR antes descrito, se obtienen los coeficientes $b_k = [1\ \ -0{,}4898\ 1]$. El resultado al aplicar el filtro se observa en la Figura 6, donde notamos que se filtra perfectamente el tono especificado.

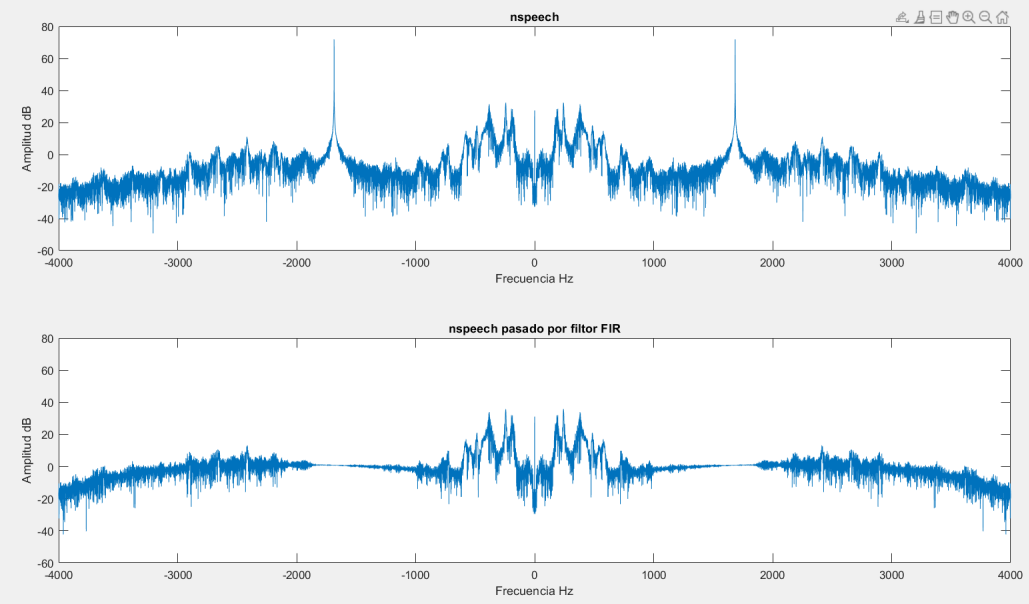

Figura 6: Filtrado usando Convolución.

Comparando el filtro FIR implementado anteriormente con el filtro de $Matlab$ implementado de la siguiente manera:

```
senal_filtrada2 = filter([1 -2*cos(fn) 1],[1 0 0], nspeech);
```

Obtenemos el resultado mostrado en la Figura 7. En ambos casos se logra eliminar el tono esperado.
Notamos que el largo del resultado de filtro por convolución es más largo debido a que al operar la convolución el largo sera la suma de el largo de las señales convolucionadas menos uno.

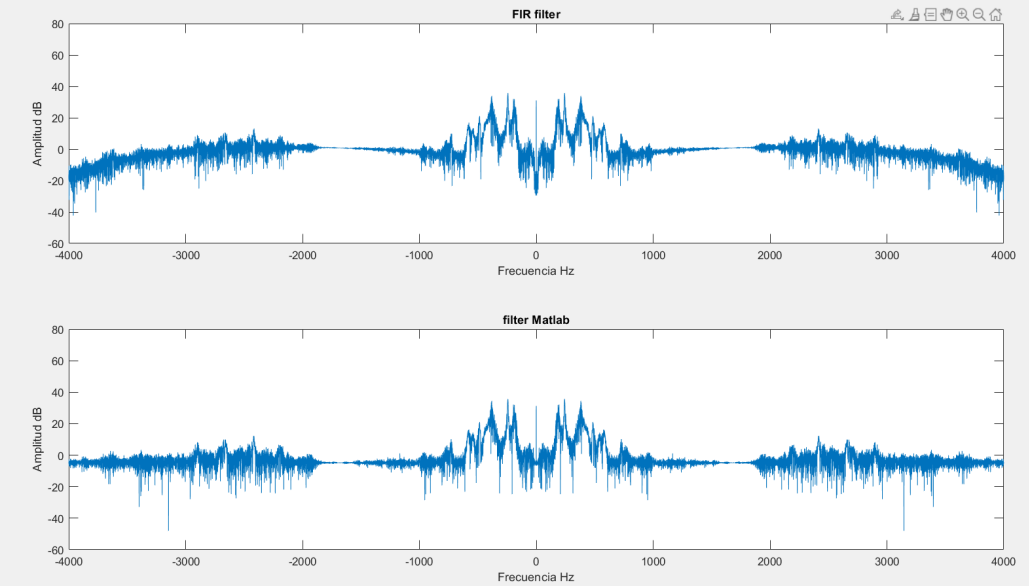

Figura 7: FIR filter vs filter Matlab.

4) Implementamos el filtro IIR con el siguiente código:

```
function y = IIR_filter(yBuff, aCoeff, x)
y_new = x - aCoeff(1)*yBuff(1) - aCoeff(2)*yBuff(2);
y = [y_new yBuff(1)];
end
```

Graficamos la FFT de la señal pcm, que se muestra en la Figura 8, donde obtenemos la frecuencia deseada del pasa-banda, la cual corresponde $frec = 3146\ Hz$, frecuencia que normalizamos para aplicar al filtro quedando $f_n = \frac{frec \cdot 2\pi}{f_s} = \frac{3156 \cdot 2\pi}{8000} = 2{,}4709\ rad/muestra$.

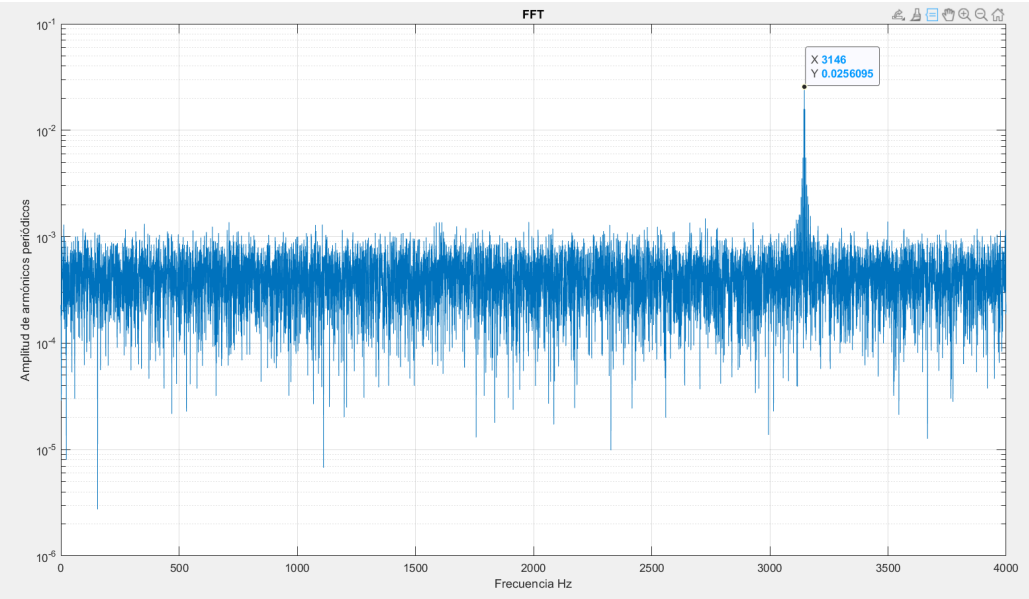

Figura 8: FFT de señal pcm.

Los coeficientes del filtro utilizados corresponden a $a_k = [-2Cos(\theta)\ r^2] = [1{,}5511\ 0{,}9801]$ Con los cuales obtenemos los resultados presentados en las Figura 9 y 10, que muestran la comparación entre la señal pcm original y la respuesta del filtro IIR, en su espectro en frecuencia y en el dominio temporal respectivamente. Notamos que se logra pasar el tono deseado perfectamente, donde en la Figura 10 se aprecia la recuperación de la señal sin ruido.

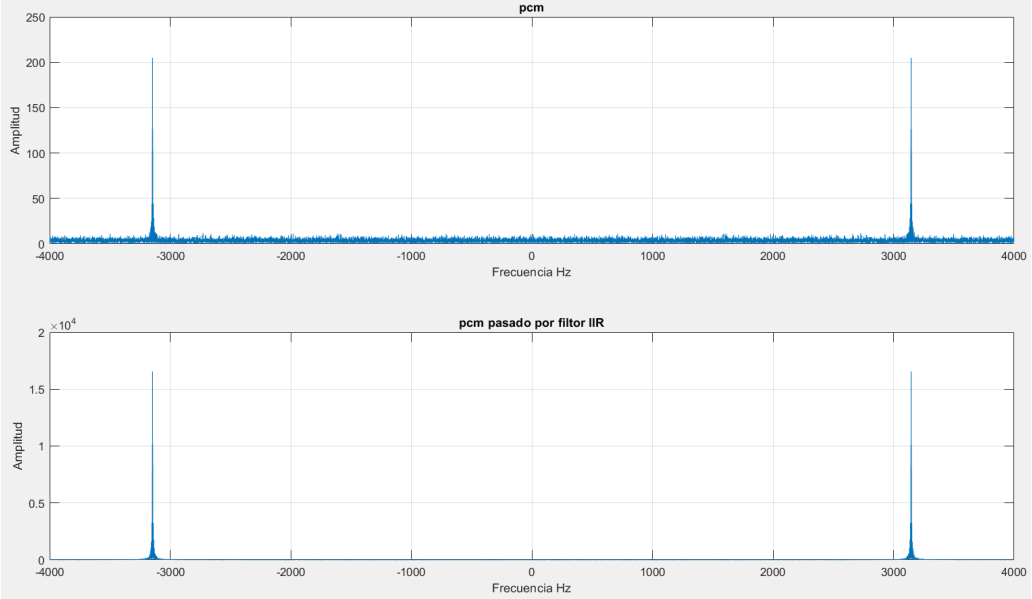

Figura 9: Señal pcm vs pcm filtrada, espectro en frecuencia.

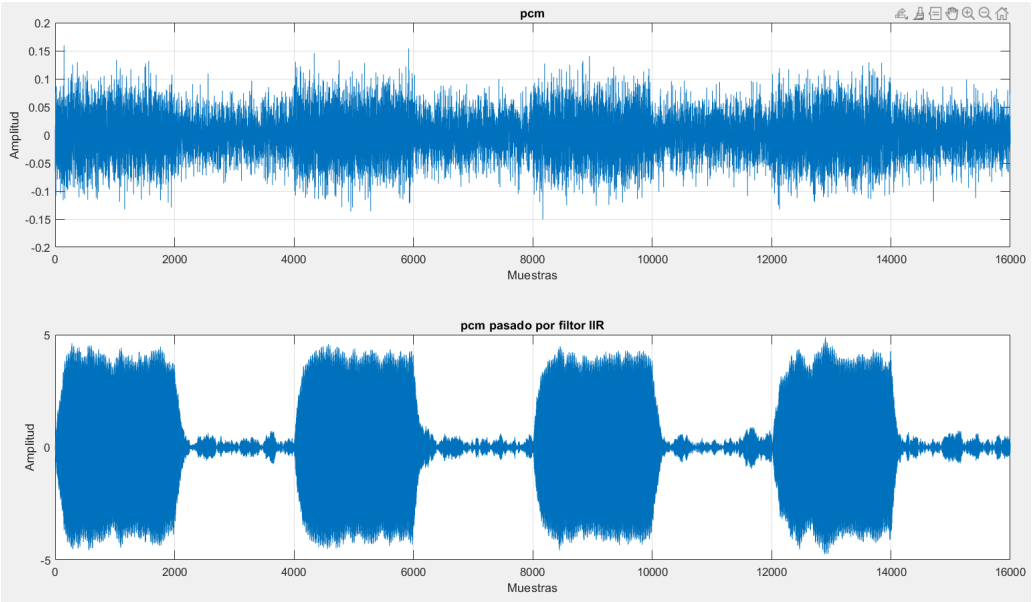

Figura 10: Señal pcm vs pcm filtrada, dominio temporal.

II. Diseño de filtros FIR usando ventanas

1) Obtenemos la magnitud y la fase de los filtros pedidos que se muestran en las Figuras 11, 12 y 13 para N iguales a 21, 101, 1001 respectivamente.
   De las figuras podemos notar que al aumentar N, la respuesta en magnitud se aproxima más a una ventana rectangular ideal, aumentando su pendiente y aumentando el rechazo de las bandas de rechazo. Notamos en la fase que al aumentar las muestras se asemeja más a una fase lineal, y que su pendiente aumenta.

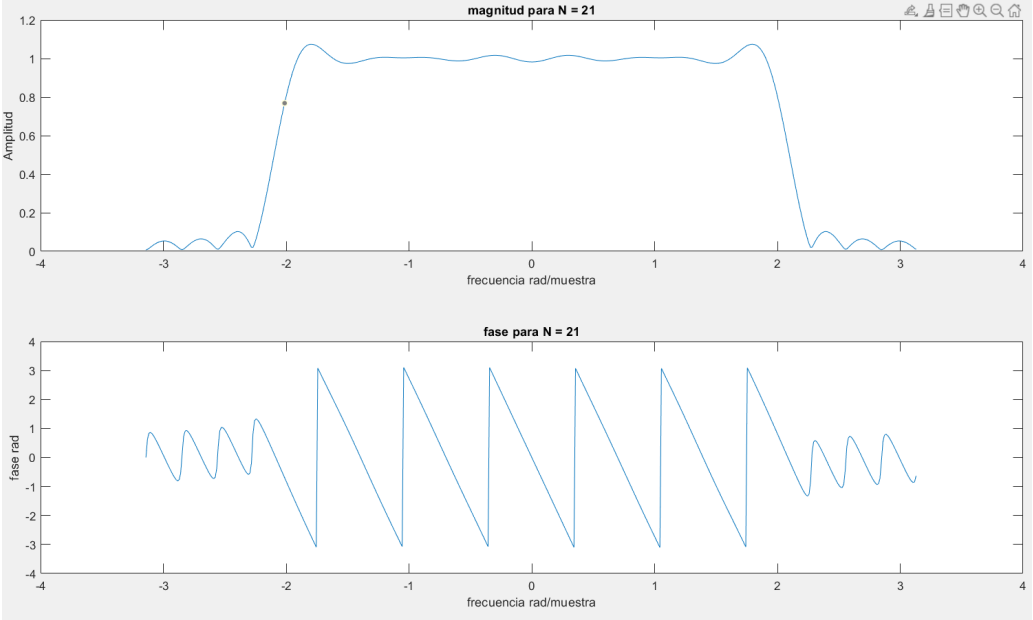

Figura 11: Amplitud y fase para filtro FIR de ventana con N = 21.

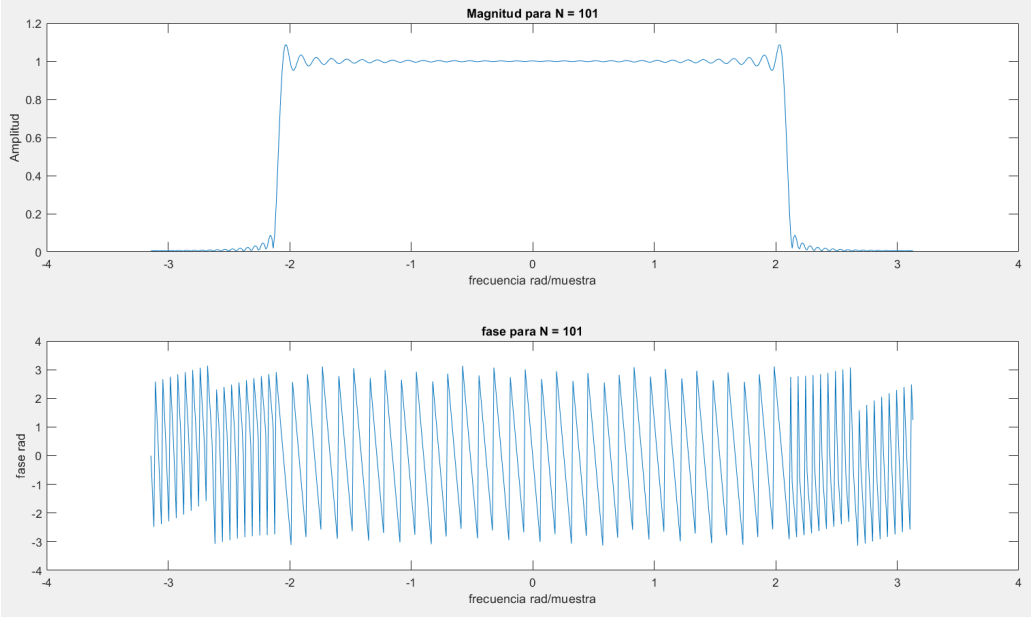

Figura 12: Amplitud y fase para filtro FIR de ventana con N = 101.

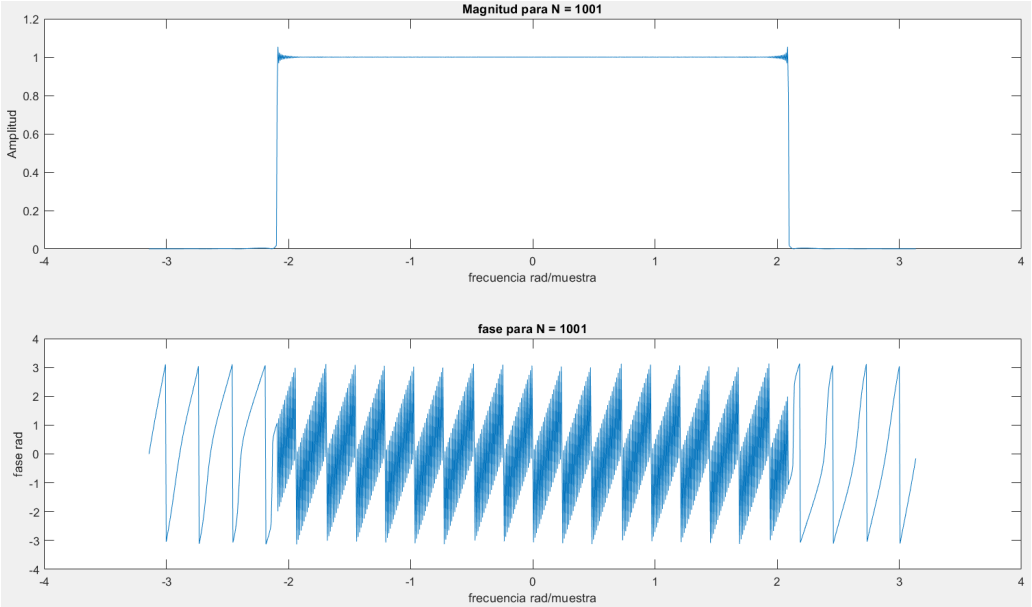

Figura 13: Amplitud y fase para filtro FIR de ventana con N = 1001.

2) En la Figura 14 se muestran las 5 ventanas.

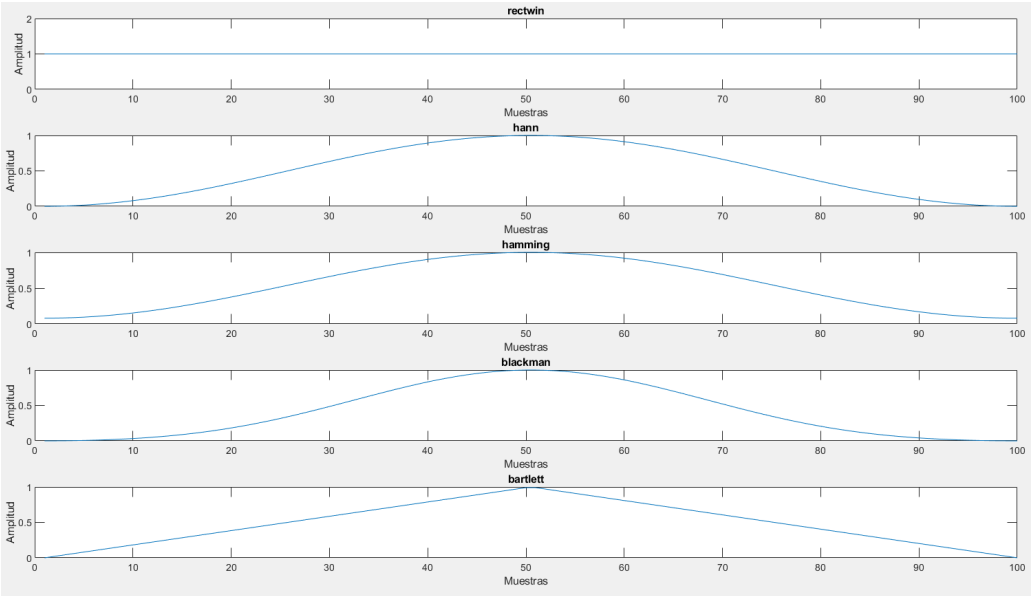

Figura 14: Ventanas generadas.

A continuación se presenta la magnitud y la fase del espectro de cada una de las ventanas.

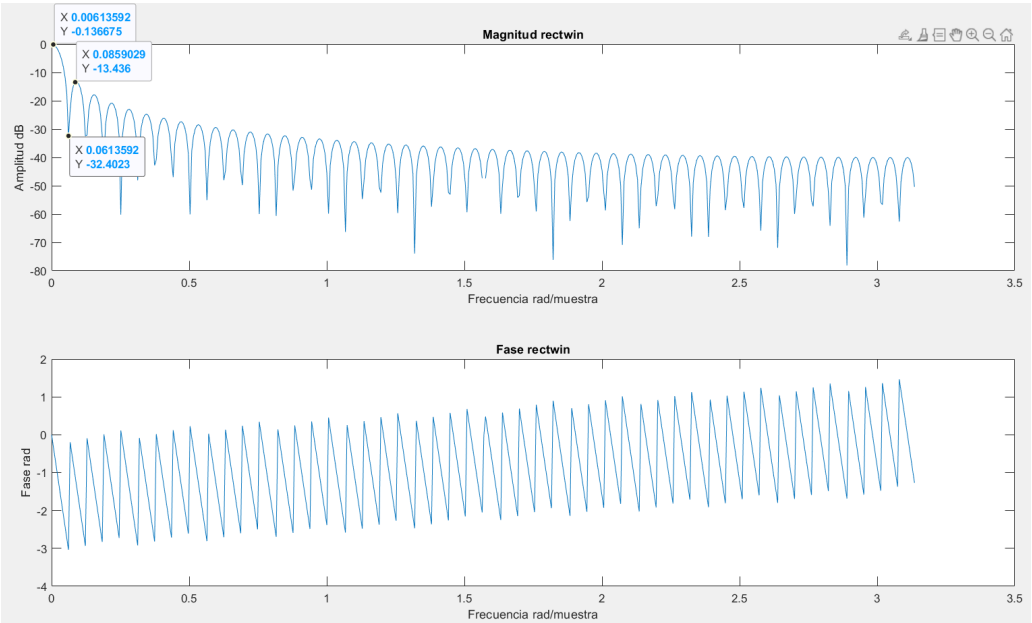

Figura 15: Ventana rectangular.

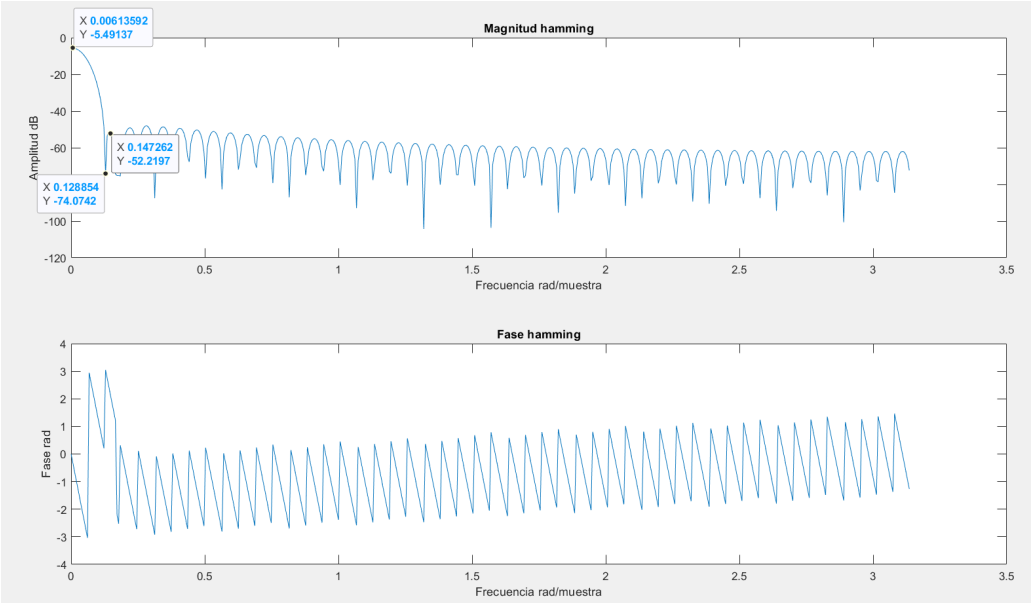

Figura 16: Ventana hamming.

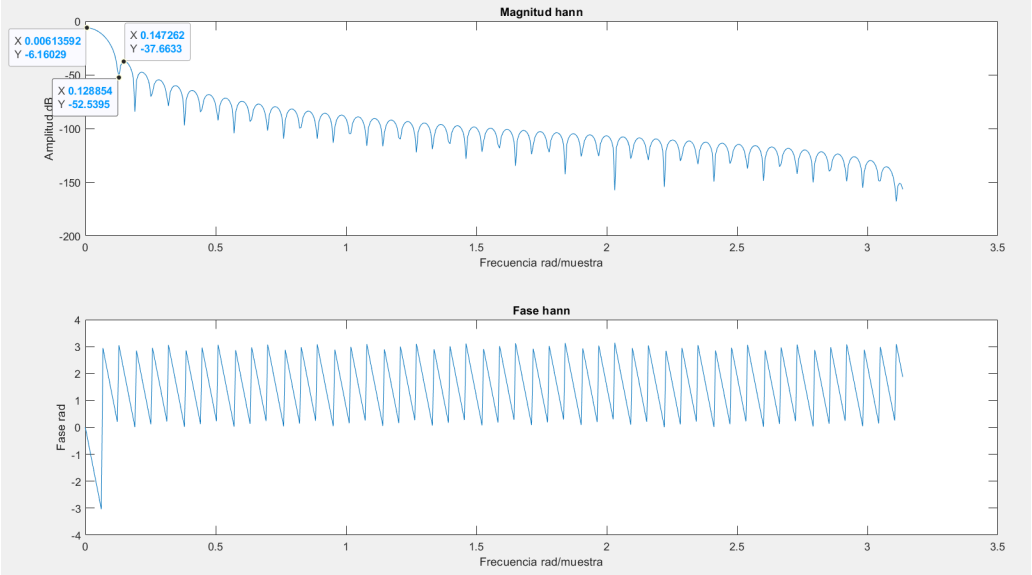

Figura 17: Ventana hanning.

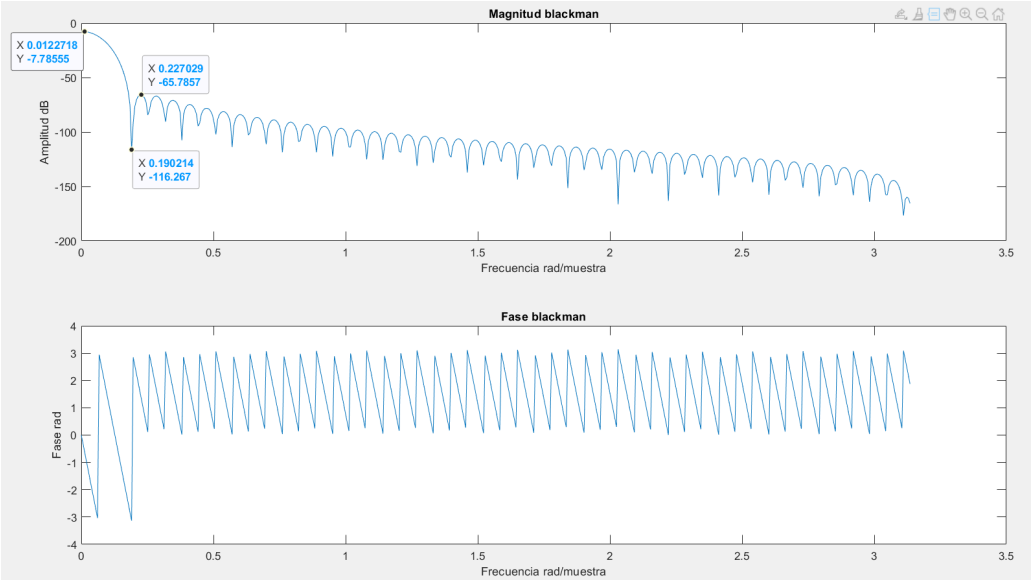

Figura 18: Ventana blackman.

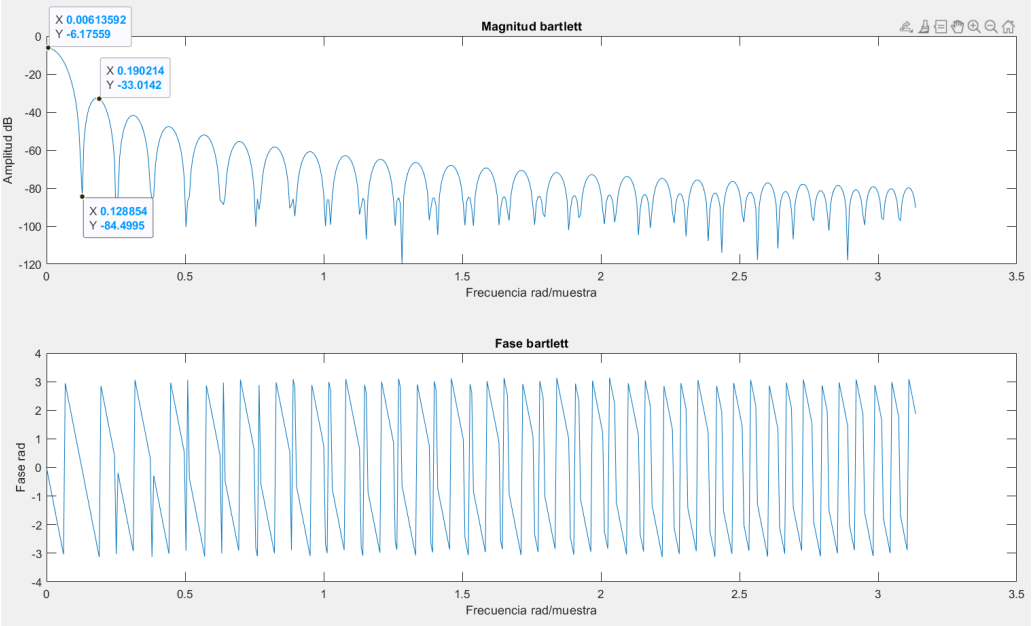

Figura 19: Ventana bartlett.

La tabla de comparación se muestra a continuación

Ventana	Ancho de Mainlobe rad/muestra	Amplitud relativa a sidelobe dB
Rectangular	0.061	13
Hamming	0.128	46
Hanning	0.128	31
Blackman	0.190	58
Bartlet	0.128	26

Tabla I: Tiempos de ejecución del buffer lineal en segundos

Se observa que ventanas como la rectangular o la Barlett, la amplitud relativa de los lobulos es mayor que para otras ventanas como Blackman o Hamming. Por lo que esas ventanas pueden resultar menos convenientes si se quiere eliminar completamente el ruido.

Por otro lado la ventana rectangular tiene el ancho de banda más bajo de todos, mientras que la Blackman (la que parecía más conveniente según la razón de amplitud relativa a sidelobe) tiene el más alto. Por lo que la rectagular logra generar filtros de banda angosta más facilmente.

III. Diseño de filtros FIR usando herramientas de Matlab

Algunas de las características de los comandos a utilizar se presentan a continuación.

- $fir1$ recibe el orden, la frecuencia de corte normalizada del filtro y el tipo de ventana, que es Hamming por defecto. el comando diseña un filtro digital FIR que por defecto es pasa-bajos y retorna su vector de coeficientes B.
- $fir2$ recibe el orden y la respuesta en frecuencia del filtro deseado, con lo que diseña un filtro digital FIR de fase lineal y retorna su vector de coeficientes B.
- $firpm$ recibe el orden y la respuesta en frecuencia del filtro deseado, con lo que diseña un filtro Parks-McClellan optimal equiripple, es decir, un filtro FIR Chebyshev óptimo y retorna su vector de coeficientes B.

1) Los resultados de implementar el filtro $fir1$ de orden 70 con una ventana rectangular y Blackman se muestran en las Figuras 20 y 21 respectivamente, que se consiguieron al ejecutar el siguiente script.

```
N = 70; %ordem
fc = 3000; %frecuencia de corte
tm = 16000;  %tiempo de muestreo
W =  fc*2/tm; %frecuencia normalizada
rect = rectwin(N+1);
bckmn = blackman(N+1);
B1 = fir1(N,W,rect);
B2 = fir1(N,W,bckmn);
```

Notamos que el filtro diseñado con ventana rectangular tiene una caída más abrupta a la banda de rechazo que el diseñado con Blackman, mientras que este ultimo tiene una mayor atenuación en la banda de rechazo con respecto al filtro de ventana rectangualar.

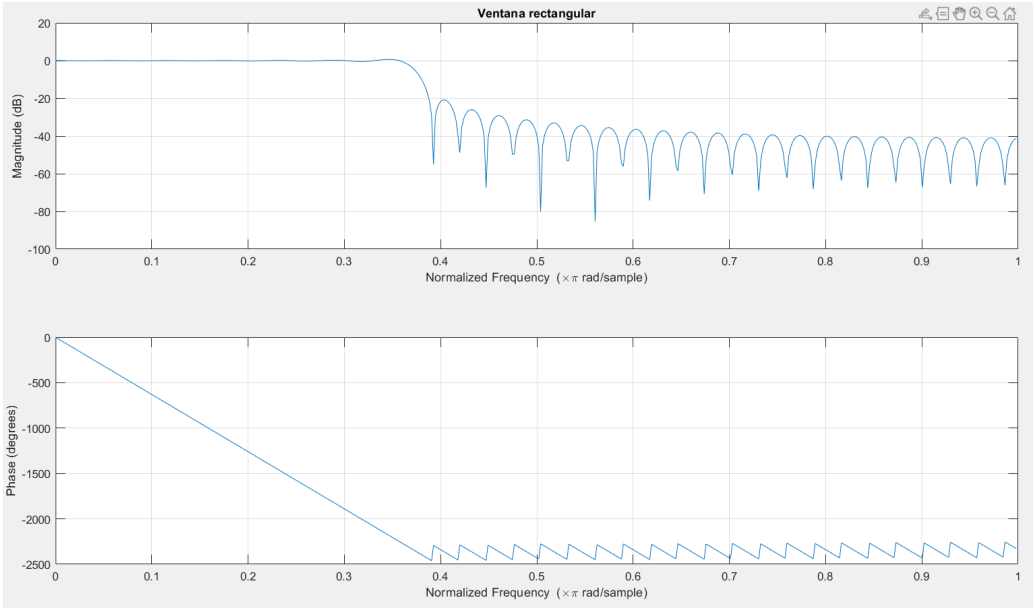

Figura 20: Magnitud y fase del filtro con ventana cuadrada.

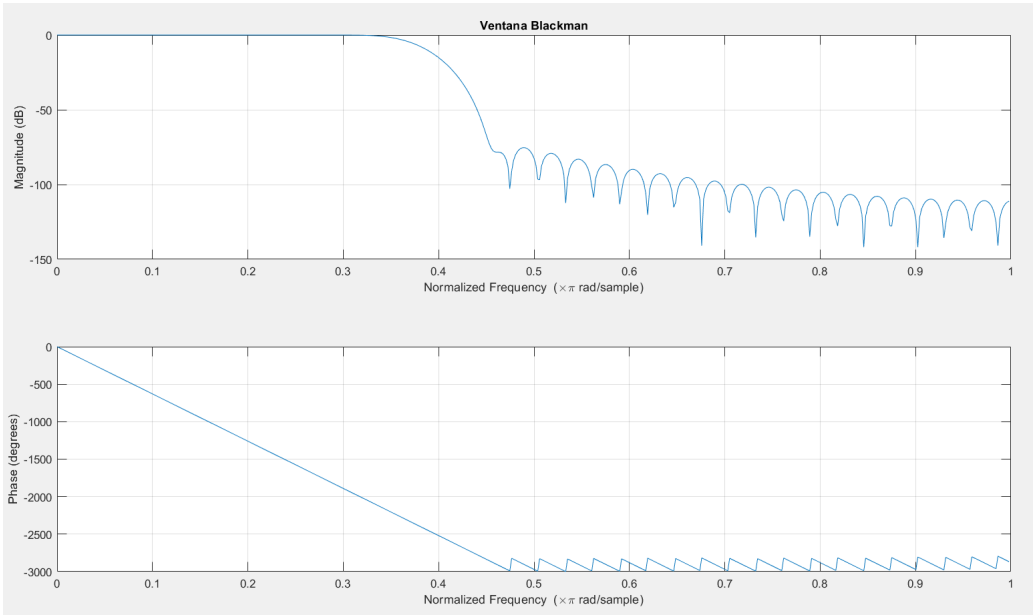

Figura 21: Magnitud y fase del filtro con ventana blackman.

2) Las respuestas en magnitud para valores de N iguales a 70 y 150 se encuentran en las Figuras 22 y 23 respectivamente. Se aprecia que al aumentar el orden del filtro la atenuación de la banda de rechazo es mayor, en este caso de unos 10 dB, además, el paso a la banda de rechazo es más abrupta. Teniendo esto en cuenta, si la aplicación deseada es muy exigente se podría considerar el filtro de mayor orden por la mejora en la banda de transición.
   Por lo tanto se genera una mayor atenuación en las zonas de rechazo, además de una zona de transición entre banda de paso y de rechazo más corta.

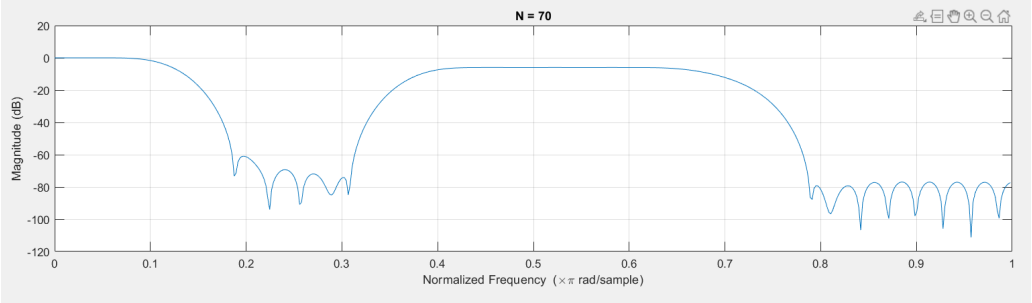

Figura 22: Magnitud del filtro de orden 70.

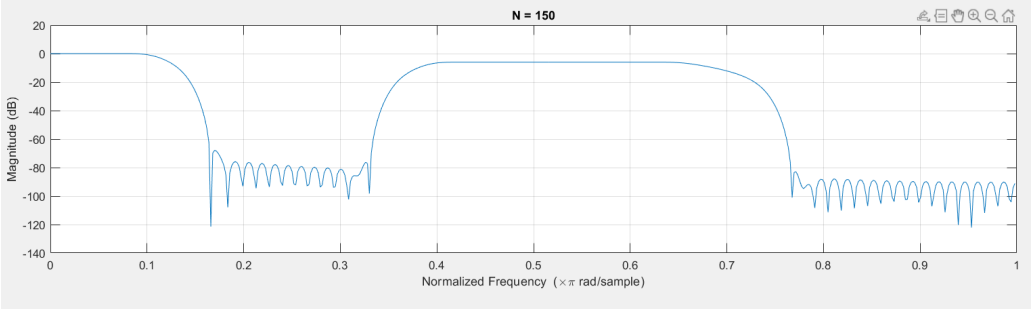

Figura 23: Magnitud del filtro de orden 150.

3) El comando firpm, es un algorítmo que diseña un filtro según el algoritmo de Parks-McClellan. Este entrega los coeficientes del filtro óptimo, recibiendo como entradas el orden del filtro, y la respuesta en frecuencia deseada (siendo esta ingresada mediante un vector de frecuenias normalizadas y un vector de amplitudes para cada frecuencia).
   La función tiene la forma `b = firpm(n,f,a)`, donde b son los coeficientes del filtro, n el orden del filtro, f el vector de frecuencias y a el vector de amplitudes requeridas para cada frecuencia.
   Estos valores pueden ser obtenidos desde el comando *firpmord*, el cual entrega el orden del filtro requerido a partir de condiciones como las frecuencias de paso y de corte y el ripple de banda de paso y de rechazo. También entrega los arreglos f y a para obtener el filtro óptimo a partir de *firpm*.

IV. Diseño de filtros IIR usando herramientas de Matlab

1) Siguiendo las recomendaciones de $help\ ellip$ usamos los parámetros $Rp = 0{,}5$ y $Rs = 20$, donde Rp es el máximo rizado en la banda de paso y Rs es la atenuación mínima en la banda de paso, por lo que tendremos un máximo $ripple$ de $0{,}5\ dB$ en las bandadas de paso y una atenuación mínima de $20\ dB$ en las bandas de rechazo. Esto se cumple en todos los filtros diseñados a continuación.

   a) Para el filtro pasa-bajos el resultado se muestra en la Figura 24.

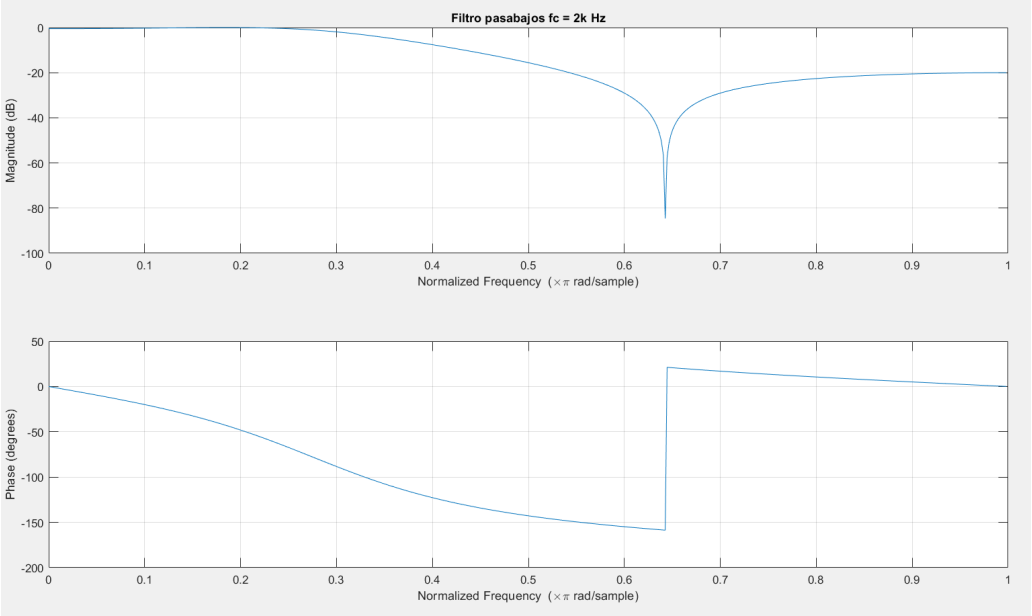

Figura 24: Magnitud y fase del filtro pasa-bajos de orden 2.

b) Para el filtro pasa-altos el resultado se muestra en la Figura 25.

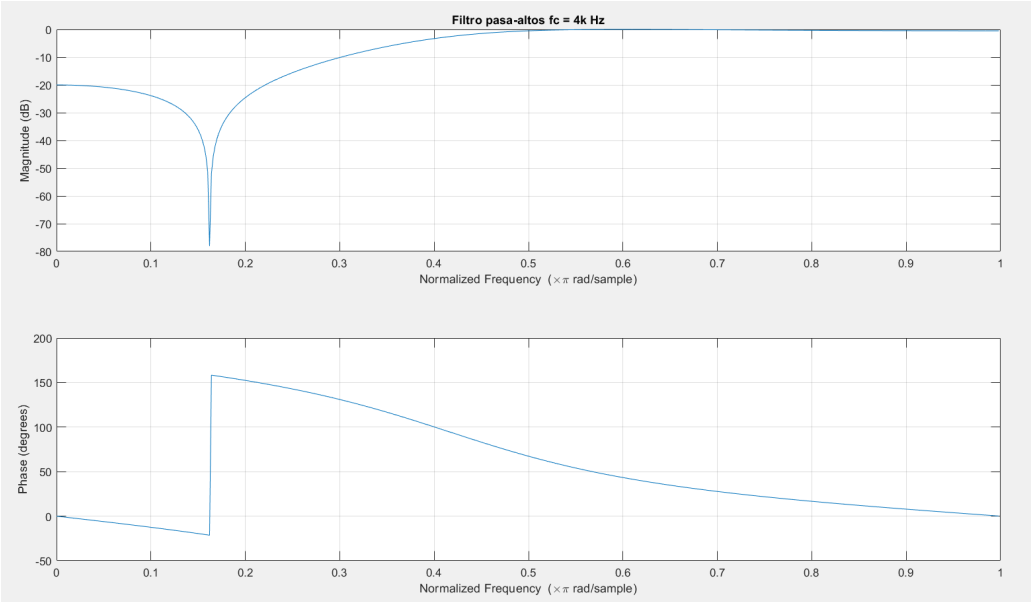

Figura 25: Magnitud y fase del filtro pasa-altos de orden 2.

c) Para el filtro pasa-bandas el resultado se muestra en la Figura 26.

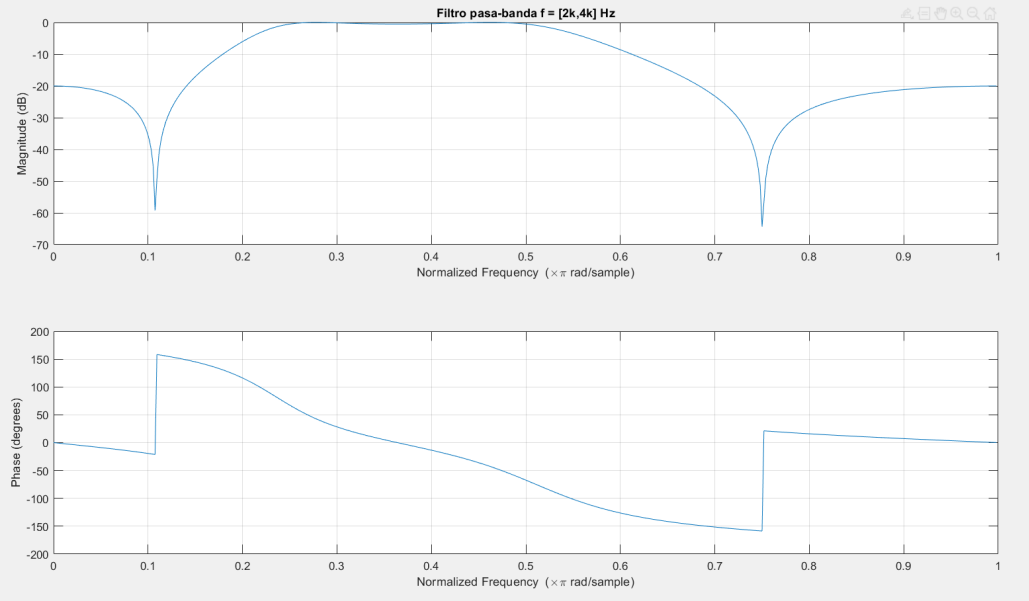

Figura 26: Magnitud y fase del filtro pasa-bandas de orden 4.

d) Para el filtro elimina-bandas el resultado se muestra en la Figura 27.

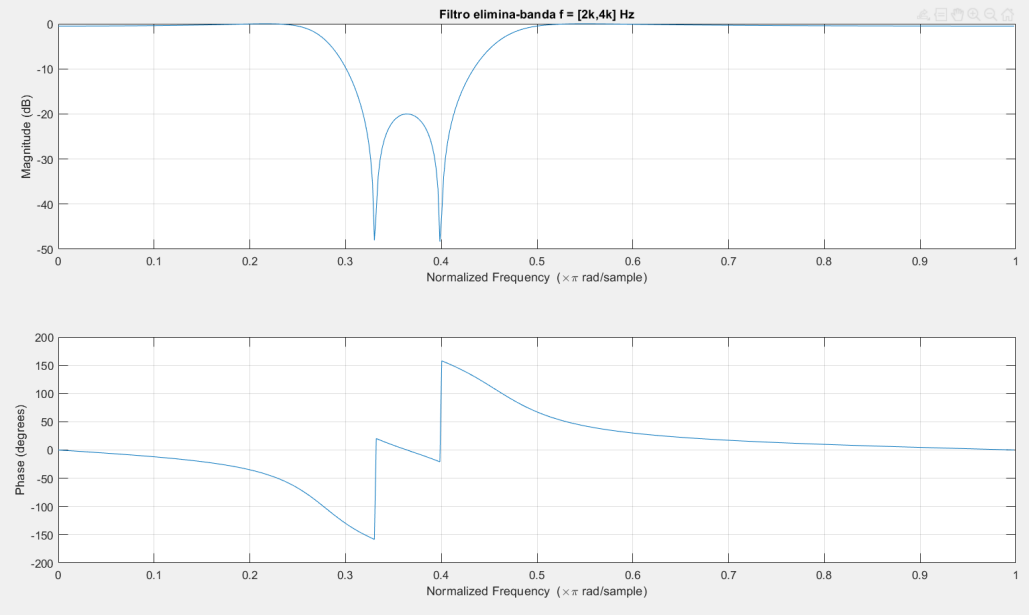

Figura 27: Magnitud y fase del filtro elimina-bandas de orden 4.

2) El comando *Cheby1* obtiene los coeficientes de función de transferencia de un filtro Chebyshev especificando la frecuencia de borde de banda de paso y el ripple de la banda de paso, mientras que el comando *Cheby2* lo hace especificando la frecuencia de borde de banda de rechazo y la atenuación de banda de rechazo.
   a) Elegimos el comando $Cheby1$ ya que este nos permite configurar el nivel peak-to-peak en decibelios del rizado en la banda de paso, en este caso $R = 2$. El resultado se muestra en la Figura 28.

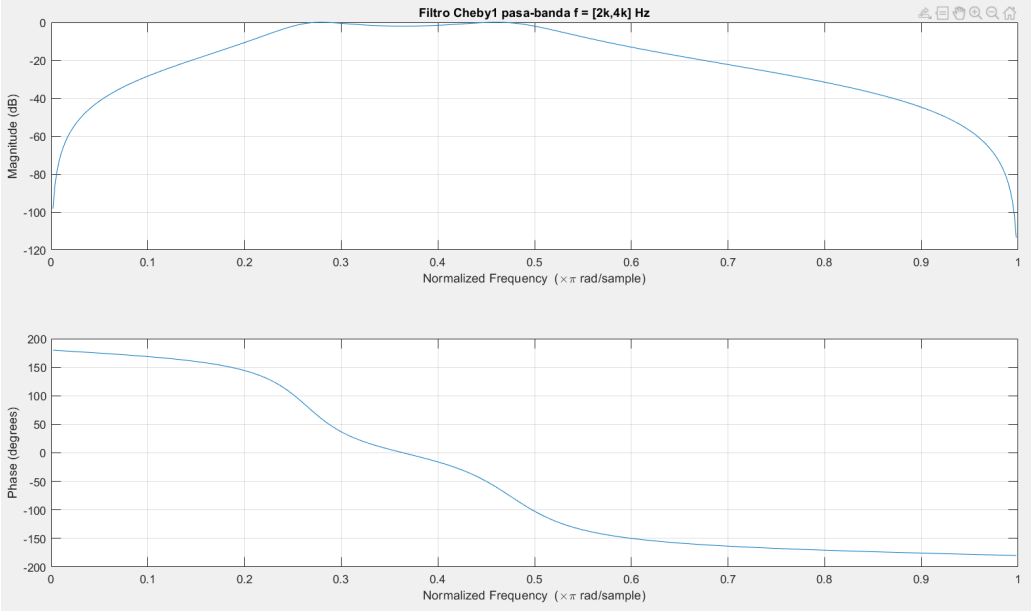

Figura 28: Magnitud y fase del filtro pasa-bandas de orden 4 para Chevy1.

b) Elegimos el comando $Cheby2$ ya que nos permite configurar la mínima atenuación en decibelios en la banda de rechazo, en este caso $R = 20$. El resultado se muestra en la Figura 29.

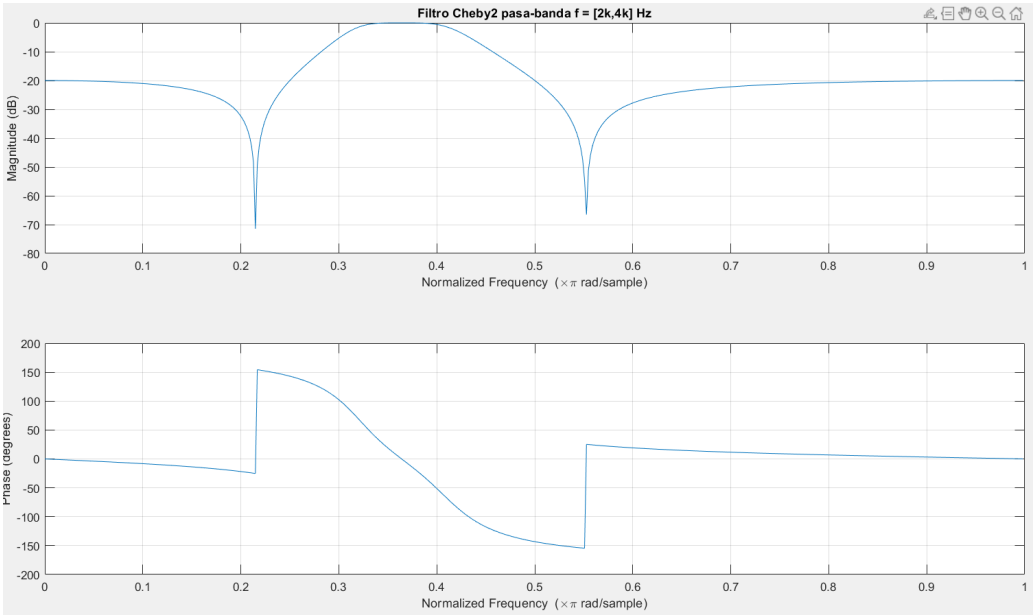

Figura 29: Magnitud y fase del filtro pasa-bandas de orden 4 para Chevy2.

3) El resultado del la respuesta en frecuencia del filtro Butterworth se muestra en la Figura 30.

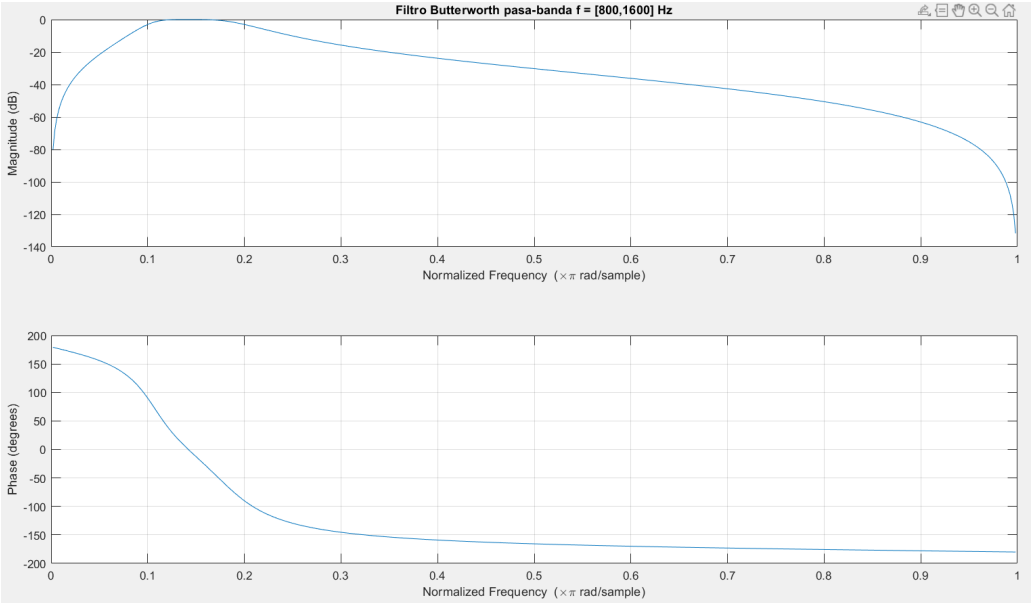

Figura 30: Magnitud y fase del filtro Butterworth pasa-bandas de orden 4.

Notamos que al aumentar en N el orden del filtro Butterworth, aumentan en N tanto los polos como los ceros del sistema. Para el filtro Butterworth de orden 4, tenemos 2 pares de polos conjugados y 4 ceros, de forma similar para el filtro de orden 8, tenemos 4 pares de polos conjugados y 8 ceros. Esto se muestra en las Figuras 31 y 32 para los ordenes 4 y 8 respectivamente.

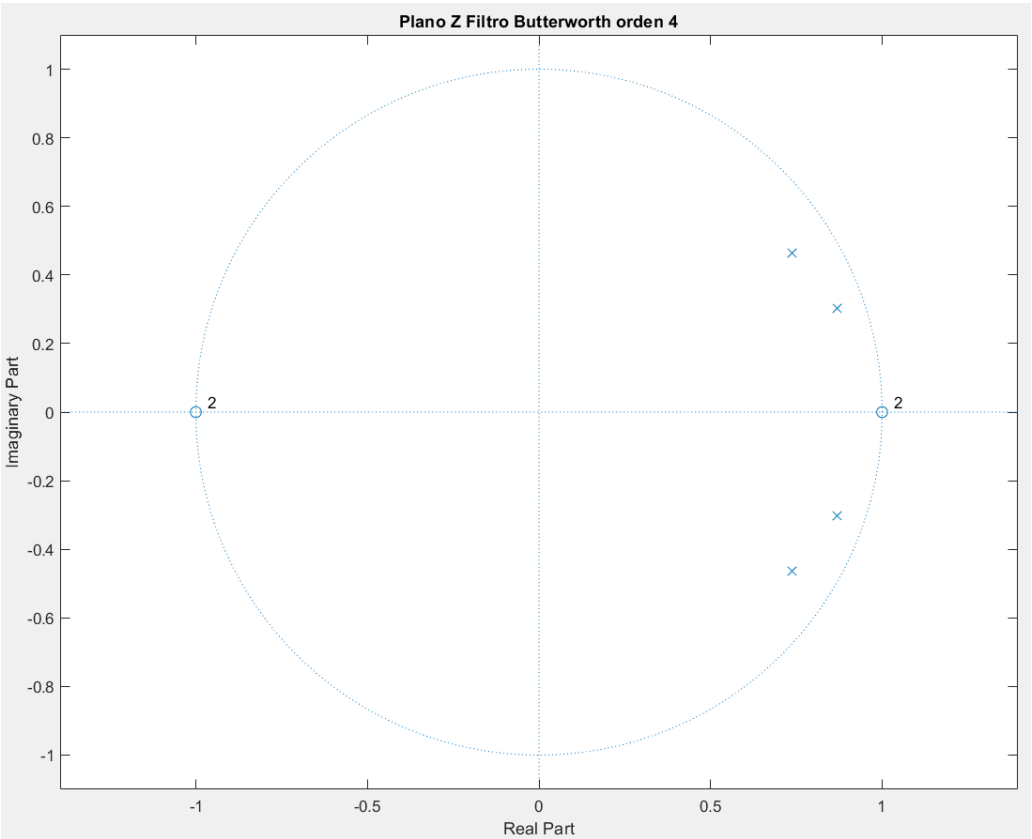

Figura 31: Plano Z de Filtro Butterworth orden 4

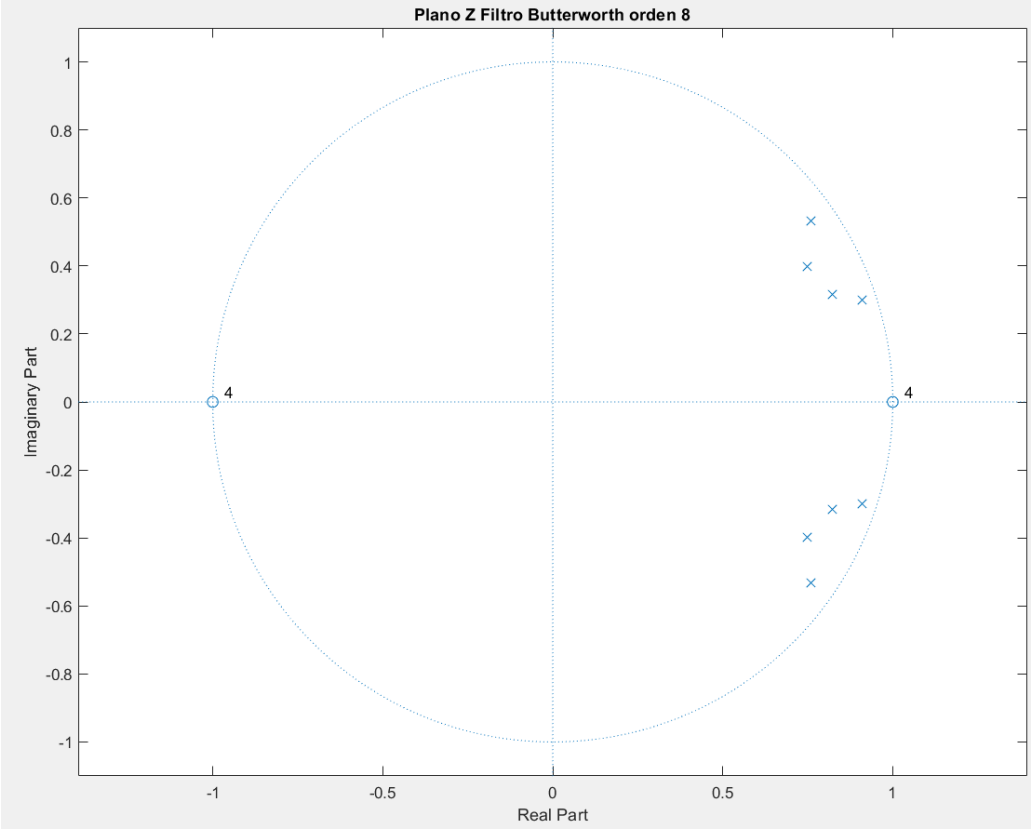

Figura 32: Plano Z de Filtro Butterworth orden 8

V. Comparación de diseños óptimos de filtros FIR e IIR

1) Se calculó el orden de los filtros Butterworth, Chebyshev, Elliptic y FIR equiripple para que cumplan los requisitos requeridos.
   Para esto se utilizó el siguiente código.

```
fp = 2400;
fsp = 4000;
fs = 16000;
Rp = 0.5; %decibeles
Rs = 40; %decibeles

[n_butter,wn_butter] = buttord(fp/fs,fsp/fs,Rp,Rs);
[n_cheb,wn_cheb] = cheb1ord(fp/fs,fsp/fs,Rp,Rs);
[n_ellip,wn_ellip] = ellipord(fp/fs,fsp/fs,Rp,Rs);

f = [fp fsp];
a = [1 0];
dev = [(10^(Rp/20)-1)/(10^(Rp/20)+1)  10^(-Rs/20)];

[n_fir,fo_fir,ao_fir,w_fir] = firpmord(f,a,dev,fs);
```

La tabla de comparación se muestra a continuación:

Filtro	Orden	Tipo
Butterworth	11	IIR
Chebyshev	6	IIR
Elliptic	4	IIR
Equiripple	16	FIR

Tabla II: Orden requerido para que los filtros cumplan los requisitos.

Se comprueba que los filtro FIR requieren orden mayor que los filtros IIR para cumplir los mismos requisitos.

VI. Implementación en s-function: filtro Biquad

1) Se creó la función `float filterBiquad(bqState_t *filterNState, float filterInput)` la cual filtra la señal de entrada según un filtro bicuadrático cuyos parámetros se determinan en la estructura `bqState_t`.
El código de la función es el siguiente:

```
1           static double filterBiquad(bqState_t *filterNState, double filterInput)
2   {
3       filterNState->bqInput[2]=filterNState->bqInput[1];
4       filterNState->bqInput[1]=filterNState->bqInput[0];
5       filterNState->bqInput[0]=filterInput;
6       filterNState->bqOutput[2]=filterNState->bqOutput[1];
7       filterNState->bqOutput[1]=filterNState->bqOutput[0];
8       filterNState->bqOutput[0]=((filterNState->bqB0*filterNState->bqInput[0]+
9       filterNState->bqB1*filterNState->bqInput[1]
10              +filterNState->bqB2*filterNState->bqInput[2])-(filterNState->bqA1*filterNState->bqOutput[1]
11              +filterNState->bqA2*filterNState->bqOutput[2]));
12
13      return filterNState->bqOutput[0];
14  }
```

Al experimentar alimentando el sistema con un delta de Kronecker, se observa que se generan oscilaciones levemente mayores al aumentar el ancho de banda.
Por otro lado, al probar la función en la señal `alternate_tones.wav`, se observa que al disminuír el ancho de banda, la respuesta transitoria entre la señal filtrada y la parte nula de la señal, es mayor.
Las figuras 33 y 34 muestran los resultados del filtrado de la señal `alternate_tones.wav` para un ancho de banda de 50 y de $200\ Hz$. Corroborando el correcto filtrado del tono correspondiente a $400\ Hz$, y observando la diferencia en las respuestas transiente previamente mencionadas.

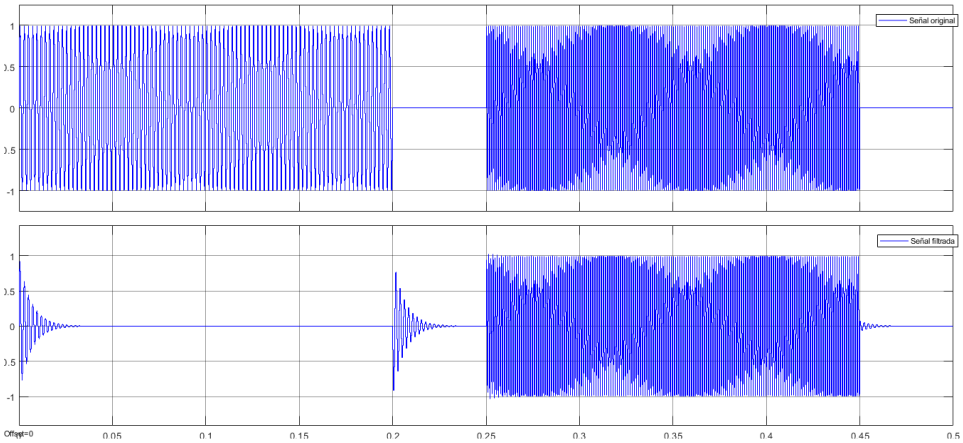

Figura 33: Señal original y señal filtrada para frecuencia central $440\ Hz$ y ancho de banda de $50\ Hz$

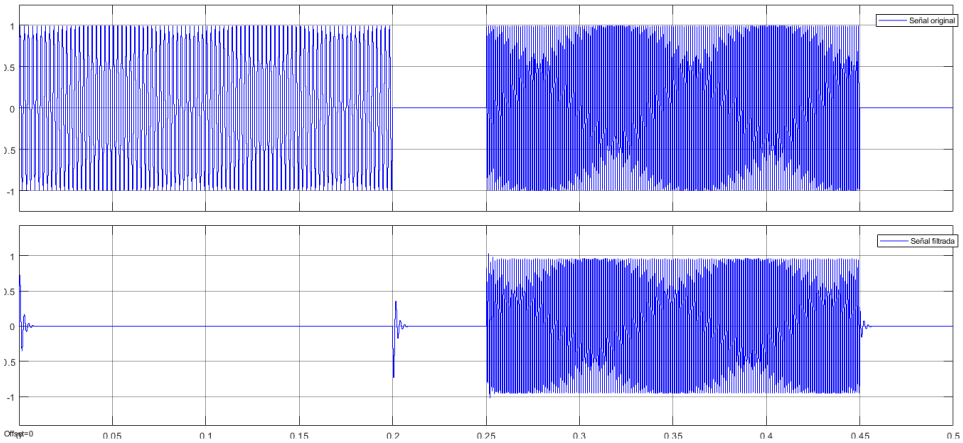

Figura 34: Señal original y señal filtrada para frecuencia central $440\ Hz$ y ancho de banda de $200\ Hz$

2) Se creó la siguiente función que calcula la respuesta de un filtro Notch en relación a una frecuencia sintonizable mediante otra entrada a la `s-Function`.

```
    void filterInterface(double data, double frec, double *output1)
{
    int BW = 200;
    int fs = 16000;

    double theta = 2*3.14151692653589793238462*frec/fs;
    double bw = 2*3.14151692653589793238462*BW/fs;
    double d = (1-sin(bw))/cos(bw);
    double gain = (1+d)/2;
    static bqState_t filtro_notch = {
        0,
        0,
        0,
        0,
        0,
        {0,0,0},
        {0,0,0}
    };
    filtro_notch.bqA1=-(1+d)*cos(theta);
    filtro_notch.bqA2=d;
    filtro_notch.bqB0=gain;
    filtro_notch.bqB1=-2*cos(theta)*gain;
    filtro_notch.bqB2=gain;

    *output1 = filterBiquad(&filtro_notch, data);
}
```

La figura 35 muestra los resultados obtenidos al aplicar está funcion utilizando como segunda entrada una frecuencia variable obtenida a partir del bloque *Repeating Sequence Interpolated*. Donde se observa como pasa de eliminar el primer tono hasta eliminar el segundo tono.

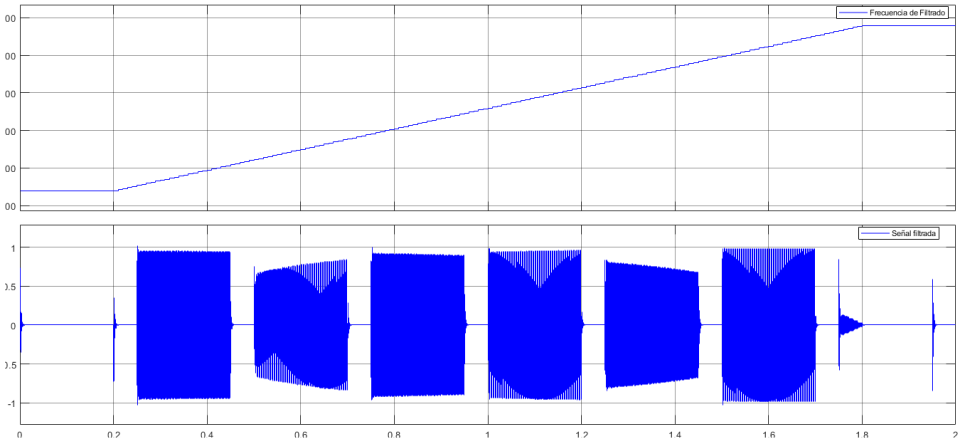

Figura 35: Señal filtrada mediante frecuencia ajustable

Se generó un archivo de audio con el filtrado lineal de frecuencias, donde se puede observar la transición entre escuchar solo el tono de $440\ Hz$, escuchar ambos y escuchar el de $880\ Hz$. El archivo ha sifo incluido bajo el nombre `filtered_alternate_tones.wav`.

3) Se completaron las funciones requeridas dentro del archivo `dtmf.c` de tal forma que la señal de entrada entre a 7 filtros pasabanda diferentes, y la salida de cada uno de estos filtros entra como vector a la función `dtmfDetection`.
Como las frecuencias de interés mas cercanas entre si están separadas por $73[Hz]$, y el filtro a usar es solamente de orden 2, se necesita usar un ancho de banda considerablemente menor a $73[Hz]$. En este caso se utilizarán $15[Hz]$ para cada filtro.
Se ingresó la entrada a 7 filtros diferentes con la misma estructura (pero distintos parámetros) y luego el arreglo resultante a la función de deteccion, de la siguiente manera:

```
    tonesInputs[0] = bandpass_filter(input1,&filtro0,697);
    tonesInputs[1] = bandpass_filter(input1,&filtro1,770);
    tonesInputs[2] = bandpass_filter(input1,&filtro2,852);
```

```
    tonesInputs[3] = bandpass_filter(input1,&filtro3,941);
    tonesInputs[4] = bandpass_filter(input1,&filtro4,1209);
    tonesInputs[5] = bandpass_filter(input1,&filtro5,1336);
    tonesInputs[6] = bandpass_filter(input1,&filtro6,1477);

    *output1 = dtmfDetection(tonesInputs);
```

Donde la función de filtraje es la siguiente:

```
    double bandpass_filter(double data, bqState_t *filterNState, int frec)
{
    int BW = 15;
    int fs = 16000;

    double theta = 2*3.14159265358979323846*frec/fs;
    double bw = 2*3.14159265358979323846*BW/fs;
    double d = (1-sin(bw))/cos(bw);
    double gain = (1-d)/2;
    filterNState->bqA1=-(1+d)*cos(theta);
    filterNState->bqA2=d;
    filterNState->bqB0=gain;
    filterNState->bqB1=0;
    filterNState->bqB2=-gain;

    return filterBiquad(filterNState, data);
```

La señal obtenida luego del filtraje se puede observar en la figura 36. Donde se obtiene que la secuencia ingresada corresponde a $\#031415926535897*$.

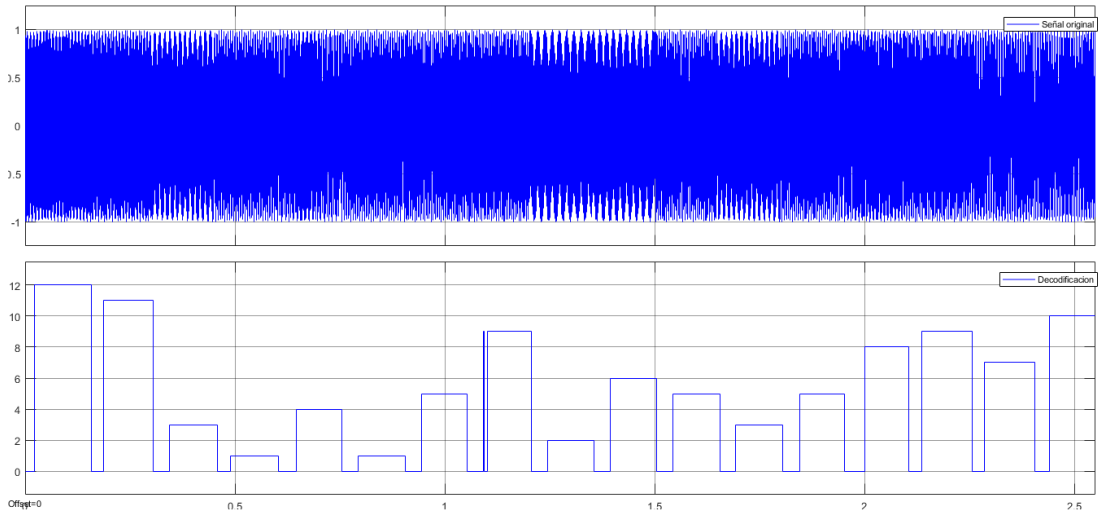

Figura 36: Secuencia DTMF original y decodificación obtenida por la s-Function

Las carácterísticas para cada filtro se observan en la siguiente tabla, en cada caso se utilizó un filtro pasa banda bicuadrático en el cual los parámetros fueron obtenidos directamente de la frecuencia central y el ancho de banda.

	f_0	BW	b_0	b_1	b_2	a_1	a_2
$Filtro_0$	697 Hz	15 Hz	0,0029	0	−0,0029	−1,9199	0,9941
$Filtro_1$	770 Hz	15 Hz	0,0029	0	−0,0029	−1,9037	0,9941
$Filtro_2$	852 Hz	15 Hz	0,0029	0	−0,0029	−1,8835	0,9941
$Filtro_3$	941 Hz	15 Hz	0,0029	0	−0,0029	−1,8595	0,9941
$Filtro_4$	1209 Hz	15 Hz	0,0029	0	−0,0029	−1,7736	0,9941
$Filtro_5$	1336 Hz	15 Hz	0,0029	0	−0,0029	−1,7259	0,9941
$Filtro_6$	1477 Hz	15 Hz	0,0029	0	−0,0029	−1,6680	0,9941

Tabla III: Características de filtros aplicados.