# Universidad Técnica Federico Santa María

## DEPARTAMENTO DE INGENIERÍA ELECTRONICA


## ELO 314 - Laboratorio de procesamiento Digital de Señales

---

# Filtros Digitales en MatLab

---


| Estudiante | ROL |
|---|---|
| Rodrigo Graves | 201621009-1 |
| Ricardo Mardones | 201621036-9 |

**Paralelo:** 1

**Profesor**

Gonzalo Carrasco

**Ayudante**

Jaime Guzmán


**Fecha : 5 de junio de 2021**

# Índice

# Índice de figuras

# Índice de cuadros

# 1. Efectos de polos y ceros en filtros FIR e IIR

1. Se diseña un filtro que tiene dos ceros conjugados sobre el círculo unitario en un ángulo $\theta$ en el plano $\mathcal{Z}$. Se deben agregar dos polos para que el sistema sea causal, por lo que se escogen dos polos en $z = 0$ ya que no afectan la respuesta del filtro.

$$H(z) = \frac{(z - e^{j\theta})(z - e^{-j\theta})}{z^2} = \frac{z^2 - 2\ z\ cos(\theta) + 1}{z^2}$$

De donde se obtiene la ecuación de diferencias

$$y[n] = x[n] - 2\ cos(\theta)\ x[n-1] + x[n-2]$$

Se estudia la respuesta a impulso de este filtro para valores de $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$, el resultado obtenido se muestra en las gráficas de la figura 1

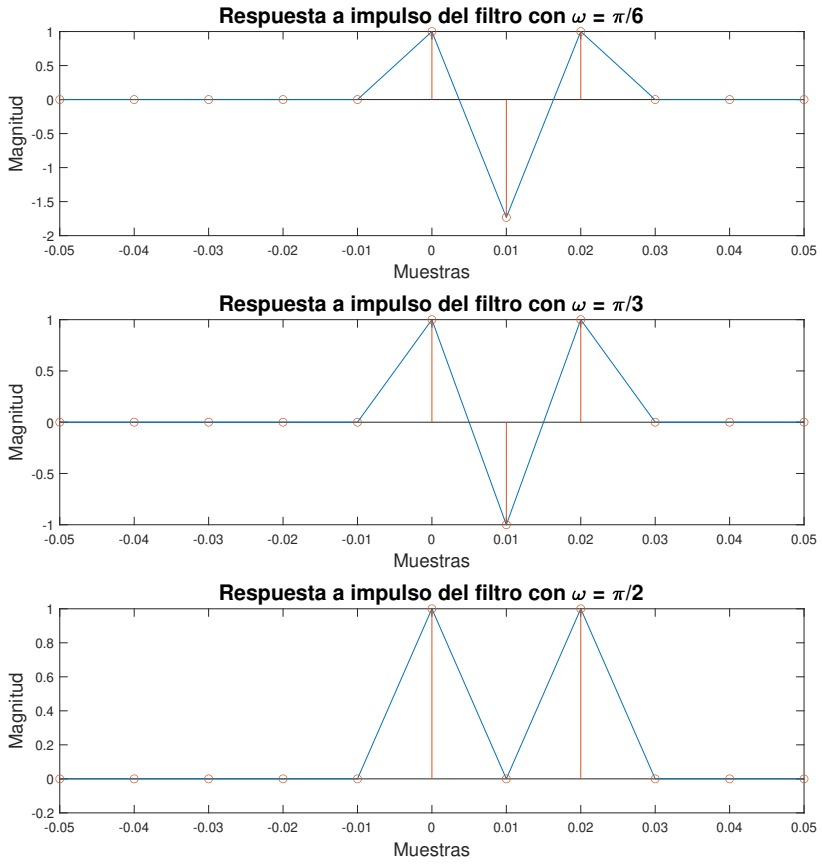


Figura 1: Respuesta impulso del filtro para ángulos $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$

Haciendo el reemplazo de $z = e^{jwT_s}$ se puede obtener una expresión analítica para la función de transferencia del filtro diseñado en función de la frecuencia $\omega$.

$$H(z) = \frac{e^{2j\omega T_s} - 2\ e^{j\omega T_s}\ cos(\theta) + 1}{e^{2j\omega T_s}}$$

Se obtiene la respuesta en frecuencia del filtro diseñado en función del ángulo para $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$ de donde tienen como resultado las gráficas de la figura 2

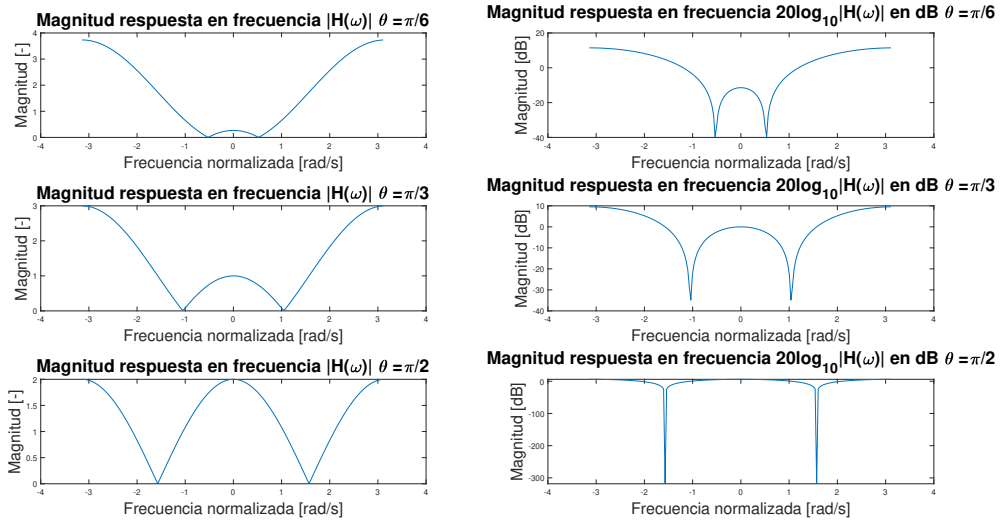


Figura 2: Respuesta en frecuencia del filtro diseñado para $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$

En la figura 3 se pueden observar las gráficas asociadas a los tres ángulos probados de manera superpuesta observar el efecto del ángulo en la respuesta en frecuencia del filtro.

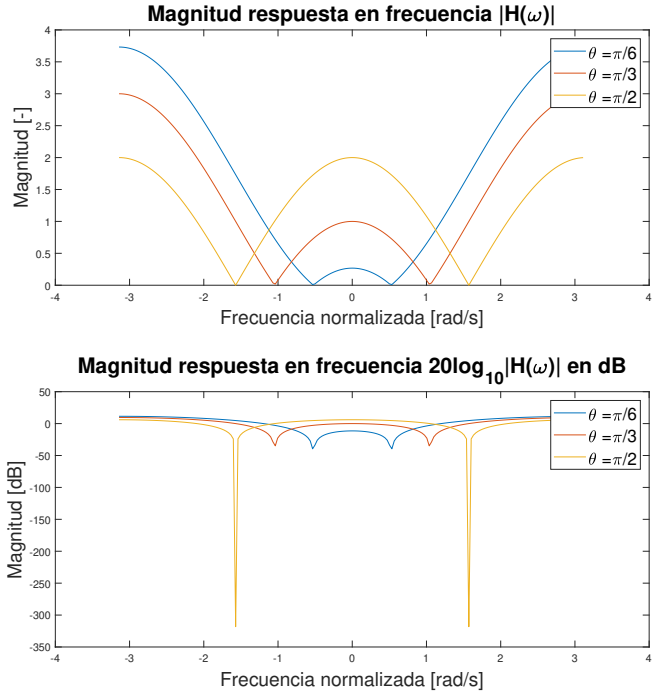


Figura 3: Respuestas en frecuencia superpuestas del filtro diseñado para $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$

De las gráficas anteriores se puede ver como la frecuencia en la que la

magnitud en frecuencia se hace cero, en una gráfica respecto a la frecuencia normalizada, corresponde justamente al valor de $\theta$, ya que para el conjunto de valores probados $\theta = \{\pi/6,\ \pi/3,\ \pi/2\}$, la respuesta en frecuencia del sistema se hace cero en $rad/s = \{0{,}5235,\ 1{,}0471,\ 1{,}5707\}$

La relación que existe entre el ángulo $\theta$ con la respuesta en frecuencia del filtro con una frecuencia de muestreo $fs$ esta dada por

$$\theta = \frac{2\pi f}{fs}$$

Ya que existe la razón directa entre $2\pi$ que corresponde a una vuelta completa al circulo unitario en el plano $\mathcal{Z}$ al igual que $fs$. Por lo que de esta forma se puede normalizar el valor de una frecuencia dada y llevarlo a la variable $\theta$. Observando el gráfico de la figura 3, se puede concluir también que el valor de $f$ en la expresión anterior corresponde al valor de la frecuencia de corte del filtro en $Hz$ y por consiguiente $\theta$ es la frecuencia de corte del filtro frecuencia normalizada medida en $rad/muestra$.

2. Se diseña un filtro que tiene dos polos conjugados dentro del círculo unitario en un ángulo $\theta$ en el plano $\mathcal{Z}$, con ganancia $1-r$. La función de transferencia queda dada por

$$\begin{aligned} H(z) &= \frac{1-r}{(z - re^{j\theta})(z - re^{-j\theta})} = \frac{1-r}{z^2 - 2zr\cos(\theta) + r^2} \\ &= \frac{(1-r)z^{-2}}{1 - 2rz^{-1}\cos(\theta) + r^2 z^{-2}} \end{aligned} \tag{1}$$

De donde se obtiene la ecuación de diferencias:

$$y[n] = (1-r)x[n-2] + 2r\cos(\theta)y[n-1] - r^2 y[n-2]$$

Posteriormente se grafican las respuestas impulso a partir de la ecuación de recurrencia, para $r = 0.99,\ 0.9,\ 0.7$ y $\theta = \pi/3$. Los gráficos se muestran en la figura 4. Notar que el decaimiento de las respuesta a impulso es más lento a medida que $r$ está mas cercano al circulo unitario, pues se acerca a la zona inestable, y que el factor sinusoidal de las 3 respuestas a impulso comparten frecuencia.

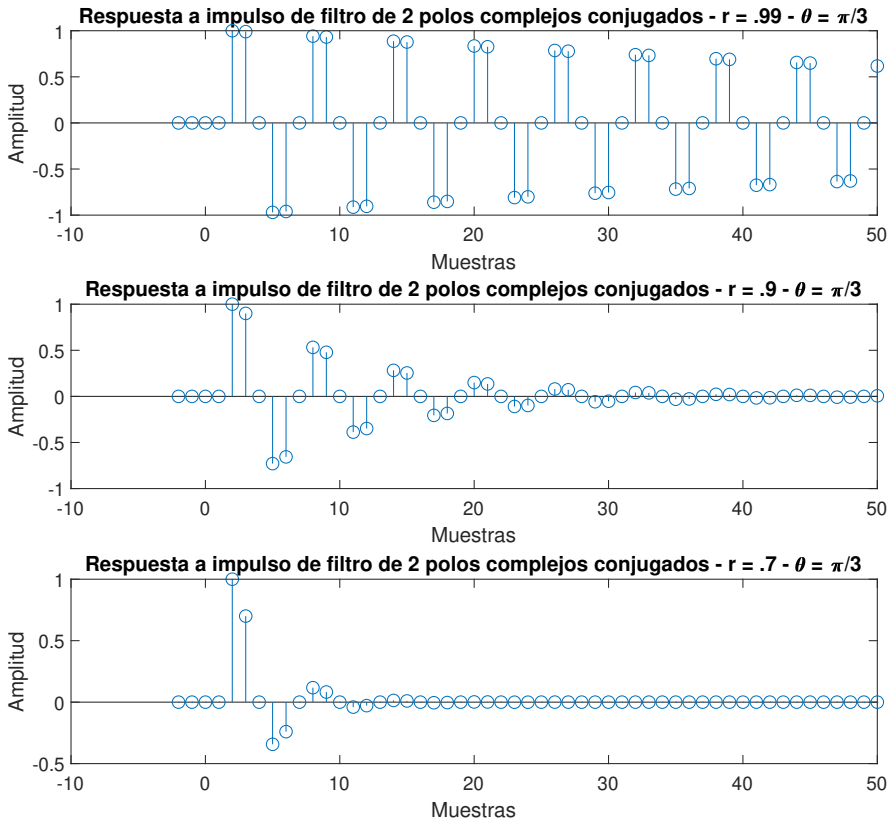


Figura 4: Respuestas a impulso de sistema de 2 polos complejos conjugados.

Luego, a partir de la función de transferencia (1) se obtiene la respuesta en frecuencia a partir de $H(z = e^{j\omega Ts})$, obteniendo:

$$H(e^{j\omega Ts}) = \frac{1 - r}{e^{2j\omega Ts} - 2ze^{j\omega Ts}\cos(\theta) + r^2}$$

expresión que se usa para graficar las respuestas en frecuencia del filtro para $r = 0.99$, $0.9$, $0.7$ y $\theta = \pi/3$. Las respuestas en frecuencia se muestran en la figura 5.

Notar que:

- La frecuencia de resonancia se mantiene en $\theta = \frac{\pi}{3}$ para los 3 filtros.
- El peak es mayor a medida que $r$ a 1 , es decir, cuando los polos se acercan a la zona inestable.
- El filtro corresponde a un pasa banda el cual es mas selectivo (disminuye ancho de banda) a medida que $r$ crece

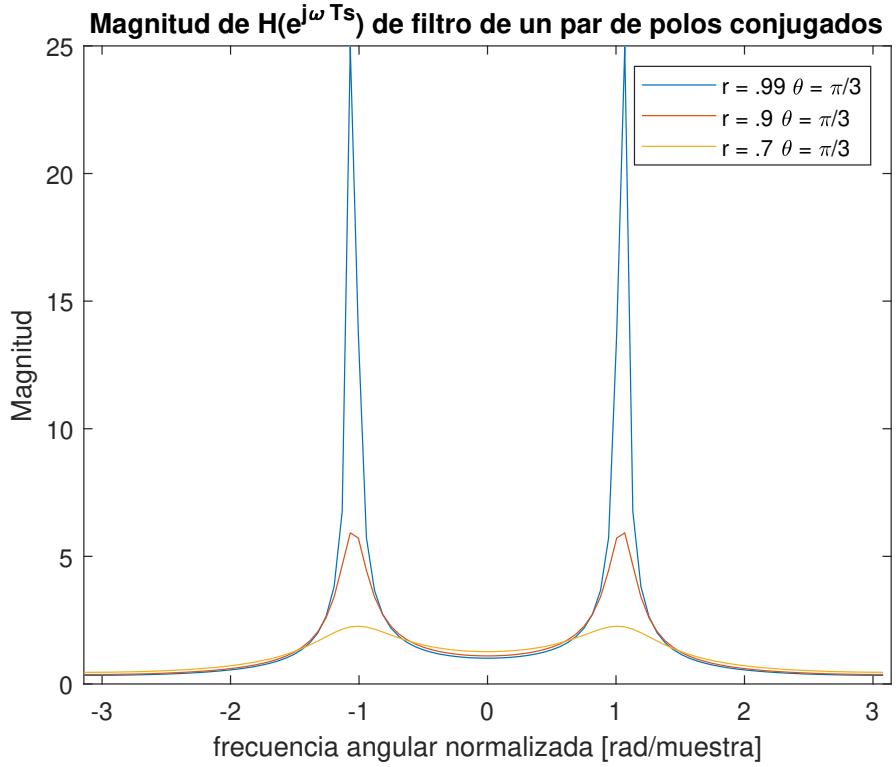


Figura 5: Respuestas en frecuencia de sistema de 2 polos complejos conjugados.

3. Haciendo uso de la función `DTFT.m` entregada en los recursos para el laboratorio se puede obtener el espectro en frecuencias correspondiente al archivo `nspeech.mat`, y identificar la frecuencia del tono puro presente en el archivo, como se ve en la figura 6

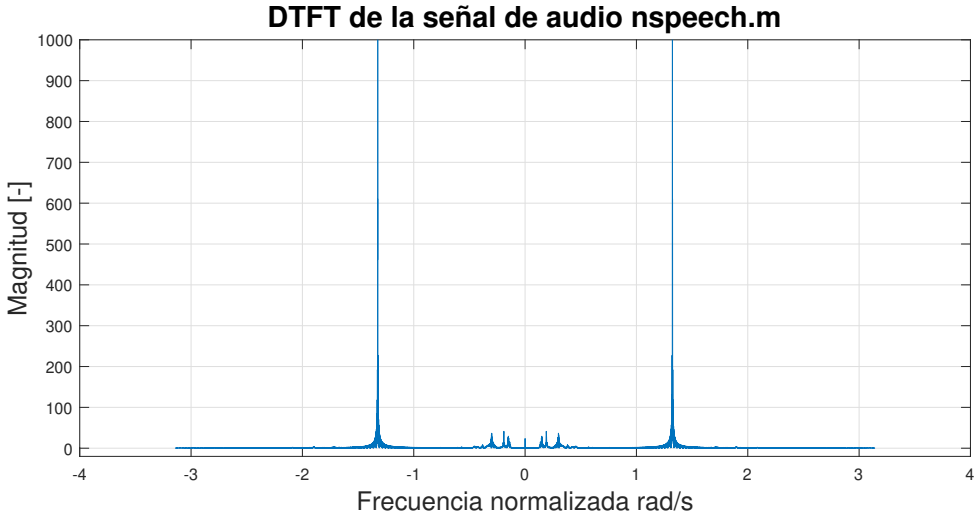


Figura 6: Espectro en frecuencia normalizada de `nspeech.mat`

Donde se puede ver que el tono se encuentra en la frecuencia 1.323. Como la gráfica está en frecuencia normalizada, este valor corresponde al ángulo $\theta = 1{,}323$;

Luego se puede diseñar un filtro FIR que rechace esta frecuencia mediante dos ceros conjugados en esta frecuencia $\theta$

$$H(z) = (z - e^{j\theta})(z - e^{-j\theta}) = z^2 - 0{,}49 + 1$$

Por lo que los coeficientes $b_k$ del filtro serían

$$B = [1 \quad -0{,}49 \quad 1]$$

Al implementar el filtro FIR mediante convolución, y aplicar el filtrado a la señal se obtienen los resultados presentes en las gráficas de la figura 7

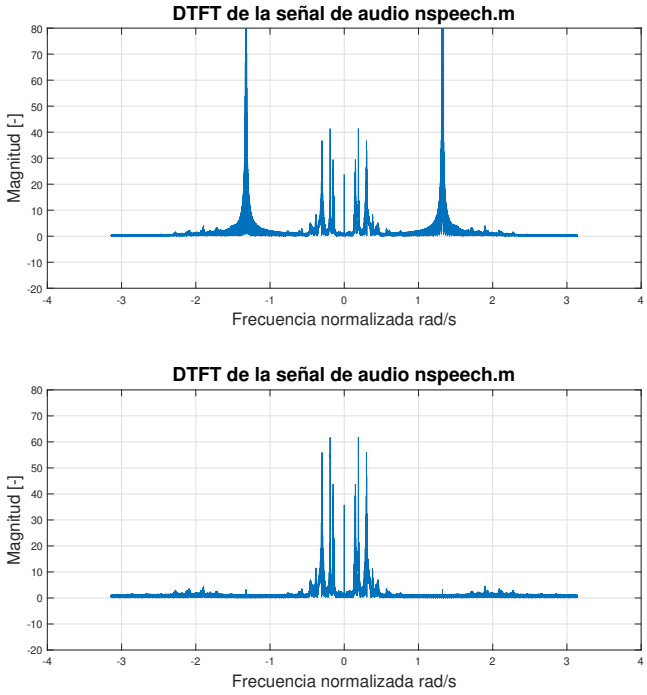


Figura 7: Espectro en frecuencia del archivo `nspeech.mat` antes y después de ser filtrado.

En donde se puede ver claramente que el tono puro que se encontraba en el archivo original se encuentra visualmente imperceptible luego de aplicar el filtro.

Cabe mencionar que el archivo original correspondía a un vector que contenía 13061 muestras, pero el resultado obtenido al aplicar la convolución con el filtro diseñado corresponde a un arreglo con tamaño distinto al original. Esto se debe a que para aplicar el comando de convolución `conv(u,v)` por lo que el tamaño del vector obtenido corresponde a `length(u)` + `length(b)` − 1 basado en cómo se desarrolla el proceso de convolución.

Se busca filtrar el ruido presente en el archivo `pcm.mat` para lo que se diseña un filtro de tipo IIR. Se grafica el espectro en frecuencia de el archivo de audio para poder identificar cual es la frecuencia que contiene la información que se desea conservar, esto se muestra en la gráfica de la figura 8

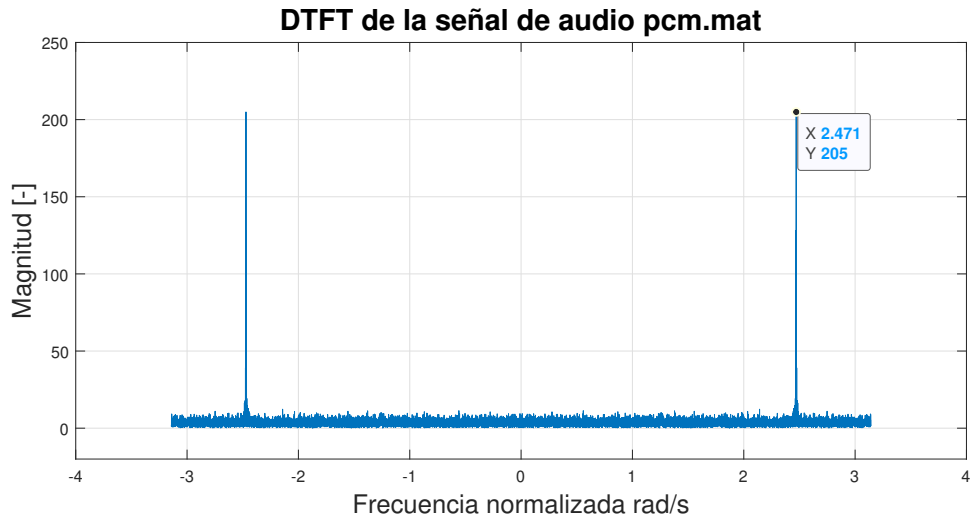


Figura 8: Espectro en frecuencia normalizada de `pcm.mat`.

Donde se puede ver que el tono se encuentra en la frecuencia 1.323. Como la gráfica está en frecuencia normalizada, este valor corresponde al ángulo $\theta = 2{,}471$;

Se diseña entonces un filtro IIR con $r = 0{,}99$ y $\theta = 2{,}471$, con función de transferencia

$$H(z) = \frac{(1-r)z^2}{(z - re^{j\theta})(z - re^{-j\theta})} = \frac{(1-r)z^2}{z^2 - 2\ z\ cos(\theta) + r^2}$$

Con coeficientes $a_k$

$$a_k = [1 \quad -2cos(\theta) \quad r^2]$$

Al implementar el filtro IIR mediante recurrencia, y aplicar el filtrado a la señal se obtienen los resultados presentes en las gráficas de la figura 9

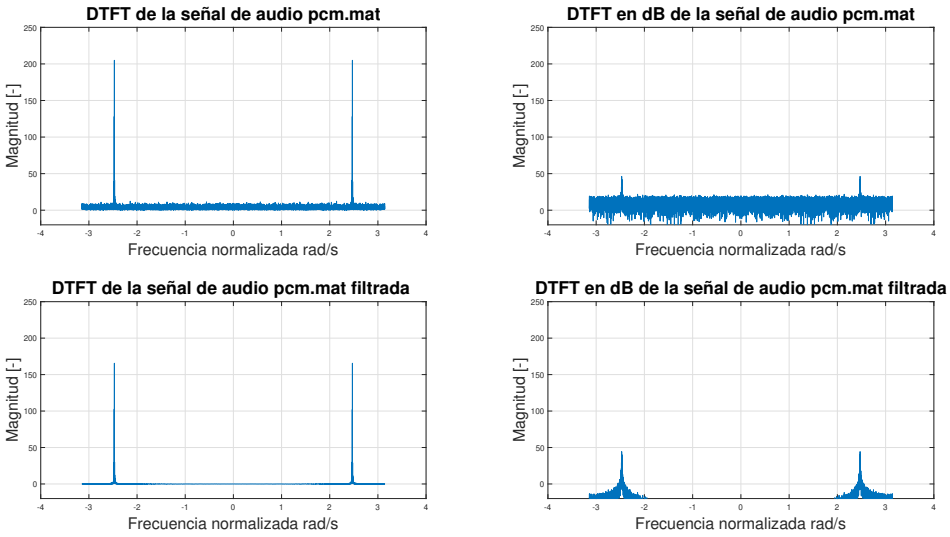


Figura 9: Espectro en frecuencia del archivo `pcm.mat` antes y después de ser filtrado.

En la figura anterior se puede observar tanto en magnitud como en decibeles cómo se ha eliminado de forma significativa el ruido presente en el archivo original.

# 2. Diseño de filtros FIR usando ventanas

1. Se estudia la aproximación a un filtro pasa bajos mediante al truncamiento de la respuesta a impulso haciendo uso de una ventana rectangular con el filtro dado por

$$h[n] = \begin{cases} \frac{w_c}{pi}\ sinc(\frac{w_c}{}\pi(n - \frac{N-1}{2})) & , \quad n = 0, 1, ..., N-1 \\ \\ 0 & , \qquad e.o.c \end{cases}$$

Donde $_c$ corresponde a la frecuencia de corte del filtro, en este caso este parámetro tomará el valor de $\frac{2\pi}{3}$

Se crean tres filtros con $N = 21,\ 101,\ 1001$, para los cuales se obtuvieron las siguientes gráficas para la magnitud y fase de su respuesta en frecuencia utilizando la función `DTFT.mat`.

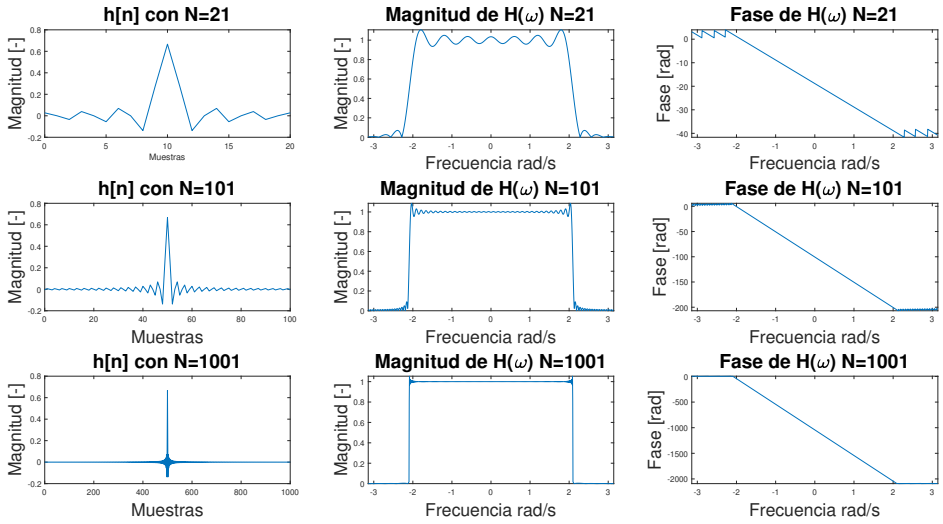


Figura 10: Magnitud y fase de la respuesta en frecuencia para los filtros de ventana rectangular con con $N = 21,\ 101,\ 1001$

Se puede observar que, como era de esperarse, una ventana con un valor de N mayor genera una respuesta en frecuencia más prolija, con menos ripple y con pendientes más pronunciadas en las frecuencias de corte acercándose de esta forma a el comportamiento de un filtro ideal, al utilizar un valor de N menor la respuesta en frecuencia se puede distorsionar debido a la presencia clara de ripple y la atenuación de las frecuencias posteriores a la frecuencia de corte no es tan efectiva.

Lo mismo ocurre con la fase, mientras más elevado es el valor de N para el filtro, la respuesta en la fase se vuelve más plana en las frecuencias que superan en magnitud a la frecuencia de corte del filtro, mientras que las frecuencias que no se atenúan por el filtro para todos los valores de N evaluados se comportan de forma similar.

2. Para comparar las propiedades espectrales y temporales de los filtros FIR construidos se hace uso de las ventanas predefinidas por MATLAB: `freqz`,

`rectwin`, `hann`, `hamming`, `blackman`, y `bartlett`; que corresponden a las ventanas de tipo rectangular, hamming, hanning, blackman, y bartlett respectivamente. Esto utilizando 1 *kps* como tasa de muestreo de cada ventana y una duración de 100 $ms$.

En la figura 11 se puede observar el comportamiento en el tiempo para cada tipo de ventana.

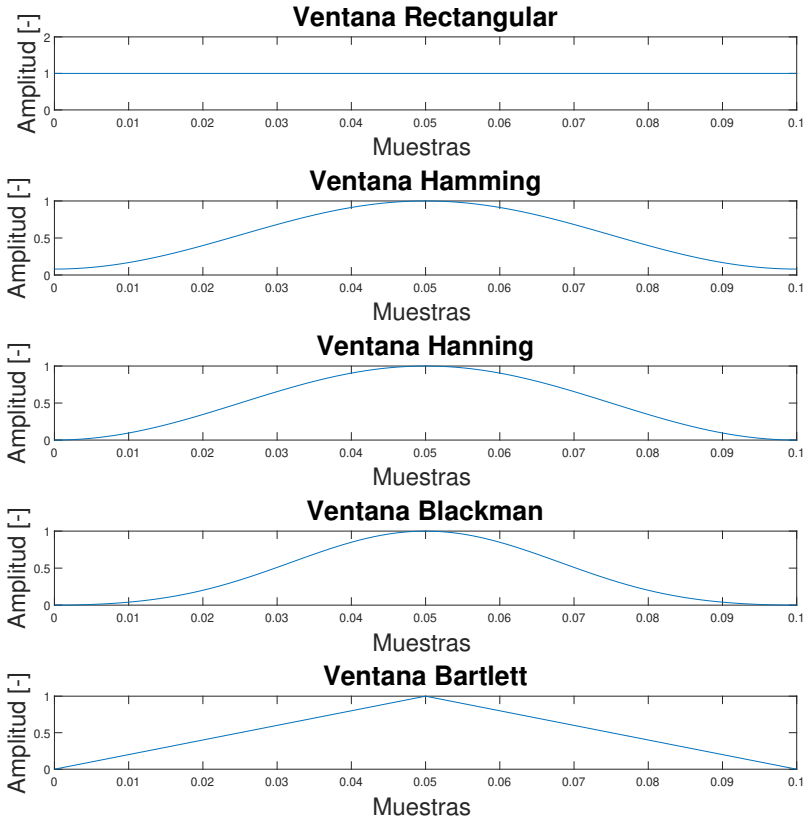


Figura 11: Ventanas rectangular, hamming, hanning, blackman, y bartlett en el tiempo.

En la figura 12 se puede ver cual es el comportamiento en frecuencia de cada uno de los tipos de ventanas utilizadas.

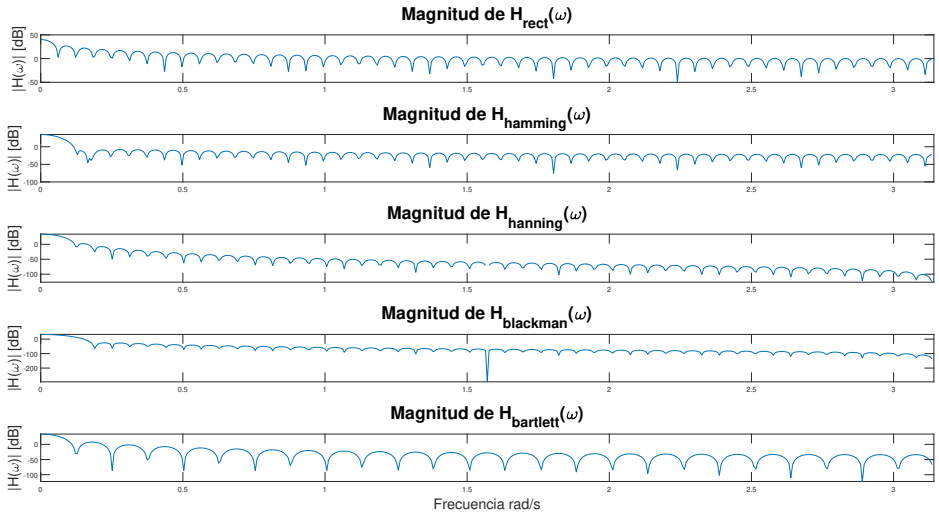


Figura 12: Magnitud de ventanas rectangular, hamming, hanning, blackman, y bartlett en frecuencia.

En la figura 13 se observa el comportamiento de las ventanas en frecuencia con énfasis en las frecuencias cercanas a cero, donde se pueden observar el ancho del *mainlobe* de la ventana, además de la fase correspondiente a la ventana.

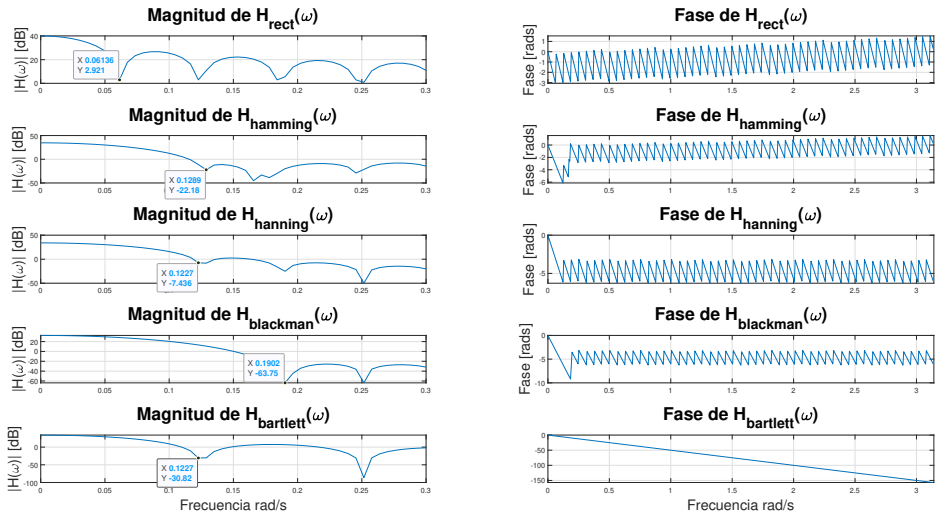


Figura 13: Magnitud y fase de ventanas rectangular, hamming, hanning, blackman, y bartlett en frecuencia.

En la figura 14, se pueden ver los valores máximos que toman el *mainlobe* y el *sidelobe* para cada tipo de ventana, valores se resumen en el cuadro 1 junto al ancho del *mainlobe* de cada ventana obtenido anteriormente

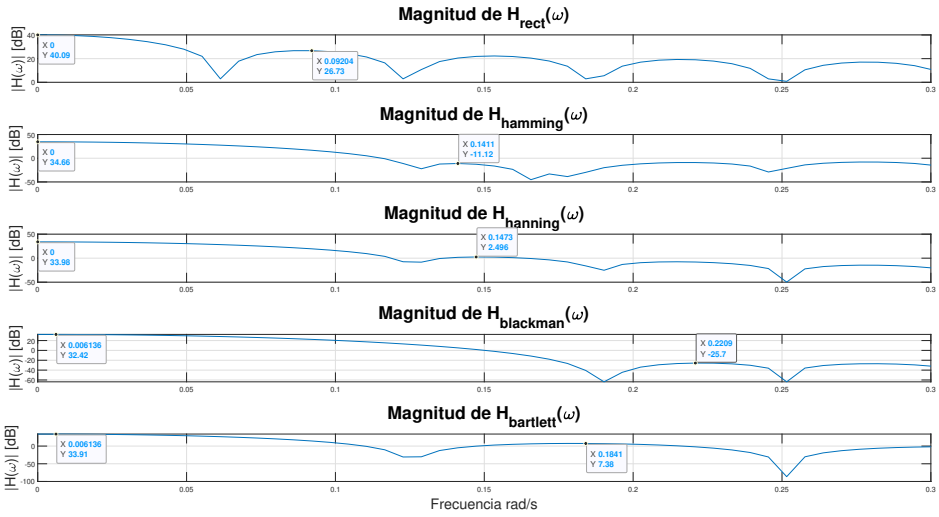


Figura 14: Magnitud de ventanas rectangular, hamming, hanning, blackman, y bartlett en frecuencia identificando la amplitud del *mainlobe* y el *sidelobe*.

Como todas las magnitudes se encuentran medidas en *db* para encontrar la amplitud relativa de el *sidelobe* basta con hacer la resta de la amplitud de este último con la amplitud del *mainlobe*

| Ventana | Ancho mainlobe | Max mainlobe | Max sidelobe | Amplitud relativa sidelobe |
|---|---|---|---|---|
| Rectangular | 0.06136 | 40.09 | 26.73 | -13.36 |
| Hamming | 0.1289 | 34.66 | -11.12 | -45.78 |
| Hanning | 0.1227 | 33.98 | 2.496 | -31.484 |
| Blackman | 0.1902 | 32.42 | -25.7 | -58.12 |
| Bartlett | 0.1227 | 33.91 | 7.38 | -26.53 |

Cuadro 1: Ancho *mainlobe* y relación entre amplitud de *mainlobe* y *sidelobe* en *db*

De la tabla anterior se observa que la ventana rectangular es la ventana que tiene un sidelobe con mayor amplitud relativa, es decir, en cuanto a eficacia del filtro este es el menos óptimo. Los filtros que poseen una mayor atenuación para las frecuencias fuera de su frecuencia de corte son los implementados con ventanas *Hamming* y *Blackman*, pero cabe mencionar que esta última compromete el ancho del lóbulo principal para lograr la atenuación que provee, esto podría generar conflictos dependiendo de lo que se busca en la implementación del filtro. Los filtros con ventanas de tipo *Hanning* y *Bartlett* tienen un desempeño intermedio entre el desempeño de ventana rectangular y el desempeño con ventanas *Hamming* y/o *Blackman*.

# 3. Diseño de Filtros FIR usando herramientas de MATLAB

Inicialmente se lee sobre las herramientas de diseños FIR en MATLAB `fir1`, `fir2` y `firpm`. Sobre dichos comandos se puede comentar que:

- `fir1`: Diseño de filtro FIR basado en ventanas. Retorna los coeficientes $b$ de un filtro FIR en base a un orden dado, una ventana y el tipo que se requiera.
- `fir2`: Diseño de filtro FIR basado en muestreo de la respuesta en frecuencia. Retorna los coeficientes $b$ de un filtro FIR en base a un orden dado y a un set de puntos de la magnitud de la respuesta en frecuencia.
- `firpm`: Diseño de filtro FIR óptimo dado una respuesta en frecuencia deseada usando el algoritmo Parks-McClellan. En secciones posteriores se analizará más en detalle.

1. En primer lugar se diseñan filtros FIR de orden 70, frecuencia de corte 3 kHz y frecuencia de muestreo 16 kHz, usando ventana rectangular y blackman, con el comando `fir1`.

   La sección de código utilizada se muestra a continuación.

```
%%% 1. Diseno de filtros FIR usando fir1

%orden
n1 = 70;
%frecuencia de corte normalizada
Fs = 16000; fc_Hz = 3000; fc = fc_Hz/(Fs/2);
%ventanas
win11 = ones(n1+1,1);win12 = blackman(n1+1);
%diseno
a1 =[1 zeros(1,n1-1)];
b11 = fir1(n1,fc,win11); b12 = fir1(n1,fc,win12);
```

   Los gráficos de la respuesta en frecuencia de los filtros diseñados con ventana rectangular y blackman se muestran en las figuras 15 y 16 respectivamente. Se aprecia que:

   - En el caso del filtro diseñado con ventana de rectangular se observa una región de transición más corta y menos atenuación en la banda de rechazo.
   - El filtro diseñado con ventana blackman presenta una atenuación mayor en la banda de rechazo.

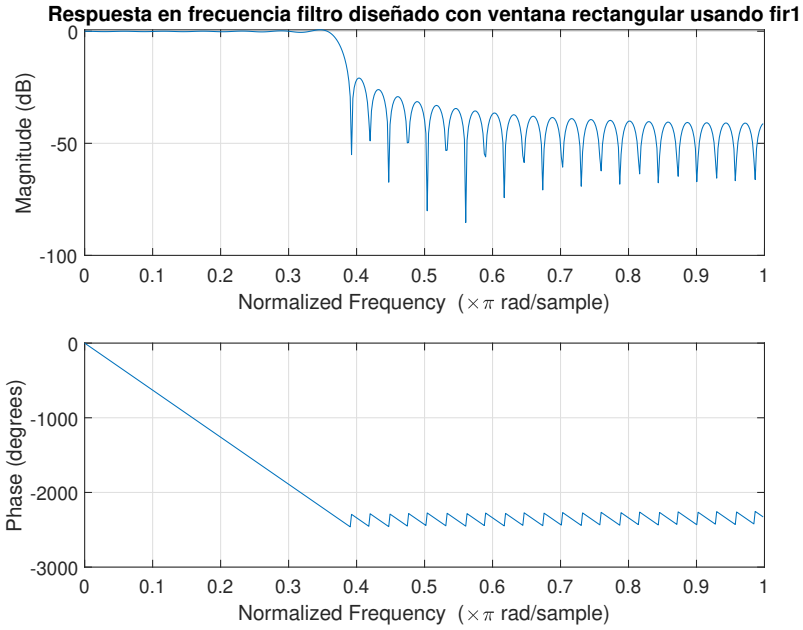


Figura 15: Respuesta en frecuencia filtro diseñado con ventana rectangular usando `fir1`.

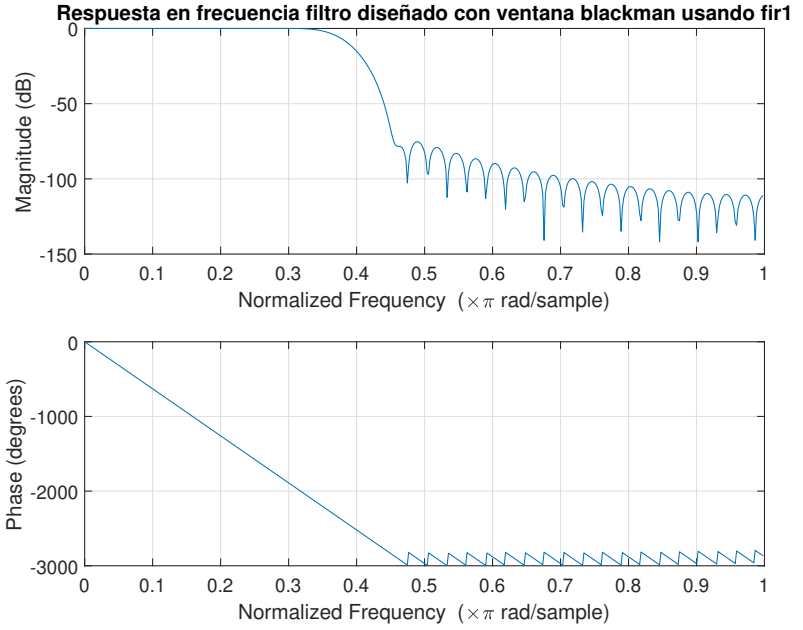


Figura 16: Respuesta en frecuencia filtro diseñado con ventana blackman usando `fir1`.

2. Se pide encontrar el filtros de orden 70 y 150 usando el comando `fir2` de MATLAB, el cual recibe el orden del filtro y un set de puntos de respuesta en frecuencia los cuales son interpolados linealmente. La respuesta en frecuencia ideal a emular se muestra en la figura 17

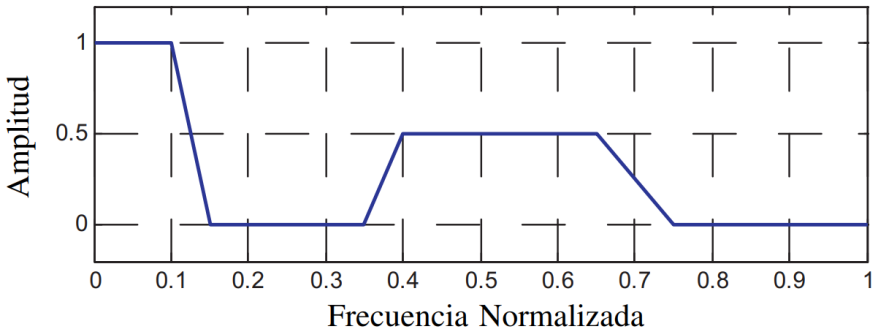


Figura 17: Respuesta en frecuencia ideal a emular.

La sección de código con la cual se diseñan los filtros se muestra a cotinuación:

```
%%% 2. Diseno de filtros FIR usando fir2

%orden
n21 = 70; n22 = 150;
%Objetivo en frecuencia
f_obj = [0 .1 .15 .35 .4 .65 .75 1];
h_obj = [1  1   0   0 .5  .5   0 0];
%diseno
b21 = fir2(n21,f_obj, h_obj); b22 = fir2(n22,f_obj, h_obj);
```

donde simplemente se reconocieron los puntos en la respuesta en frecuencia tal que al interpolar linealmente se llegara a la respuesta en frecuencia ideal.

En primer lugar se grafica la magnitud de la respuesta en frecuencia del filtro ideal y los 2 diseñados, lo cual se muestra en la figura 18. Se aprecia un diseño correcto según las especificaciones en frecuencia.

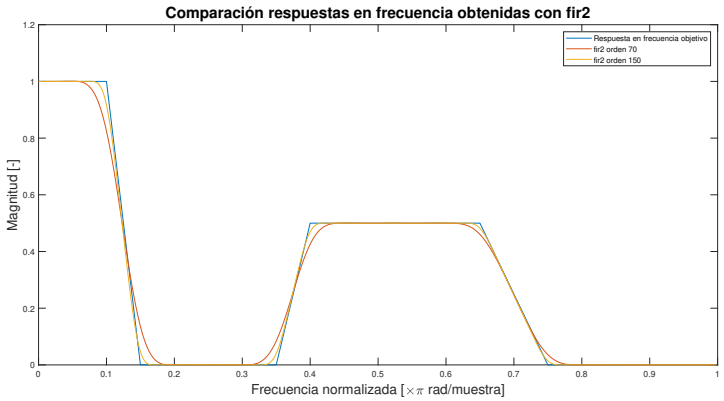


Figura 18: Magnitud de la respuesta en frecuencia de filtro ideal y de diseñados con `fir2`.

Posteriormente, para una mejor comparación entre la respuesta en frecuencia de ambos filtros diseñados, se grafica la magnitud de la respuesta en frecuencia en dB, lo cual se muestra en la figura 19.

Se aprecia que no hay gran diferencia en atenuación entre ambos filtros, siendo obviamente el filtro de mayor orden de atenuación más fuerte. La

mayor diferencia se aprecia en la banda de transición, la cual es claramente mas abrupta en el filtro de mayor orden. Dependiendo la aplicación esa podría ser una ventaja a considerar.

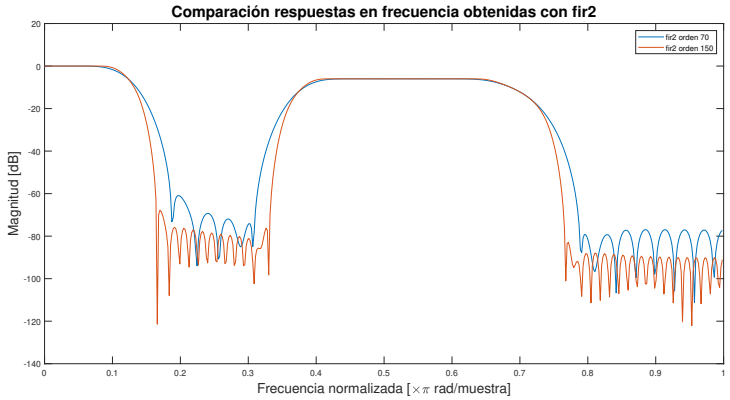


Figura 19: Magnitud de la respuesta en frecuencia de filtro ideal y de diseñados con `fir2` en dB.

3. El comando `firpm` hace referencia al diseño de filtro FIR ocupando el algoritmo Parks-McClellan.

   Dicho algoritmo corresponde a un método iterativo para el diseño óptimo de filtros FIR dada una respuesta en frecuencia deseada y sujeto a un orden. Cabe destacar que el criterio de optimalidad es en el sentido de la norma $\infty$ del error en frecuencia o, en otras palabras, del máximo error absoluto entre la respuesta en frecuencia deseada y la respuesta frecuencia obtenida situando los ceros del filtro.

   La sintaxis típica del comando es `b = firpm(n,f,a)` donde:

   - `b`: Vector de coeficientes MA del filtro FIR.
   - `n`: Orden del filtro a diseñar.
   - `f`: Vector de frecuencias.
   - `a`: Vector de amplitudes de la respuesta en frecuencia correspondientes al vector de frecuencias.

   Una forma de utilizar este comando es en el diseño de filtros chebyshev cuando existe una restricción de ripple para obtener un filtro de bajo orden.

   Para lo anterior en primer lugar se ocuparía el comando `firmord` para estimar el orden del filtro óptimo dada una respuesta en frecuencia deseada junto a sus restricciones de ripple y luego ocupar esa misma respuesta en frecuencia deseada junto con el orden obtenido para la obtención de los coeficientes de `b` usando `firpm`.

# 4. Diseño de Filtros IIR usando herramientas de MATLAB

1. En primer lugar se diseñan 4 filtros elípticos usando el comando `ellip` de MATLAB. Las especificacones de dichos filtros son:

   - Filtro a: Pasa bajos orden 2 ($f_c = 2kHz$).
   - Filtro b: Pasa altos orden 2 ($f_c = 4kHz$)
   - Filtro c: Pasa banda orden 4 ($f_1 = 2kHz, f_2 = 4kHz$)
   - Filtro d: Elimina banda orden 4 ($f_1 = 2kHz, f_2 = 4kHz$)

   como suposición se escogió:

   - Ripple en banda de paso: 5 dB.
   - Atenuación en banda de rechazo: 20 dB.

   La sintaxis típica del comando `ellip` es: `[b,a] = ellip(n,Rp, Rs, Wn, Ftype)` donde:

   - `b`: Coeficientes de la parte MA del filtro.
   - `a`: Coeficientes de la parte AR del filtro.
   - `n`: Orden del filtro
   - `Rp`: Ripple máximo tolerado en banda de paso.
   - `Rs`: Atenuación en banda de rechazo
   - `Wn`: Frecuencia o frecuencias de corte en $\pi$·rad/muestra
   - `Ftype`: Tipo de filtro (pasa bajos, pasa altos, pasabanda o elimina banda).

   Con las suposiciones tomadas y las condiciones del diseño del filtro se diseñan los filtros usando el siguiente segmento de código:

```
%%% 1. Diseno de filtros IIR usando ellip

%ordenes
n1a = 2; n1b = 2; n1c = 4; n1d = 4;
%frecuencias de corte normalizadas (pasa bajos y pasa altos)
Fs = 16000; fc1_a1_Hz = 2000; fc1_a1 = fc1_a1_Hz/Fs;%filtro a
fc1_b1_Hz = 4000; fc1_b1 = fc1_b1_Hz/Fs;            %filtro b
%frecuencias normalizadas (pasa banda y elimina banda)
f1_c1_Hz = 2000; f1_c1 = f1_c1_Hz/Fs;               %filtro c
f1_c2_Hz = 4000; f1_c2 = f1_c2_Hz/Fs;
f1_d1_Hz = 2000; f1_d1 = f1_d1_Hz/Fs;               %filtro d
f1_d2_Hz = 4000; f1_d2 = f1_d2_Hz/Fs;
%Suposiciones de ripple y banda de rechazo
Rp = 5; Rs = 20; %Ripple banda de paso y atenuacion (rechazo)
%Diseno de Filtros
[b1_a, a1_a]=ellip(n1a, Rp, Rs, 2*fc1_a1, "low");
[b1_b, a1_b]=ellip(n1b, Rp, Rs, 2*fc1_b1, "high");
[b1_c, a1_c]=ellip(n1c, Rp, Rs, 2*[f1_c1 f1_c2], "bandpass");
[b1_d, a1_d]=ellip(n1d, Rp, Rs, 2*[f1_d1 f1_d2], "stop");
```

Las respuestas en frecuencia de los filtros diseñados se muestran en las figuras 20, 21, 22 y 23. Se aprecia que se cumplen los requisitos de diseño.

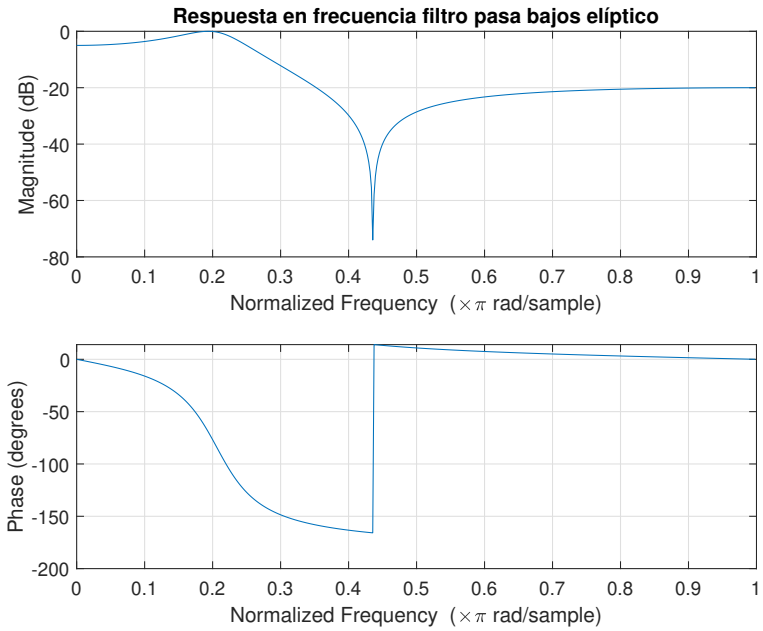


Figura 20: Respuesta en frecuencia de filtro pasa bajos diseñado usando `ellip`, de orden 2, $f_c = 2kHz$, ripple en banda de paso 5 dB y atenuación de 20 dB en banda de rechazo.

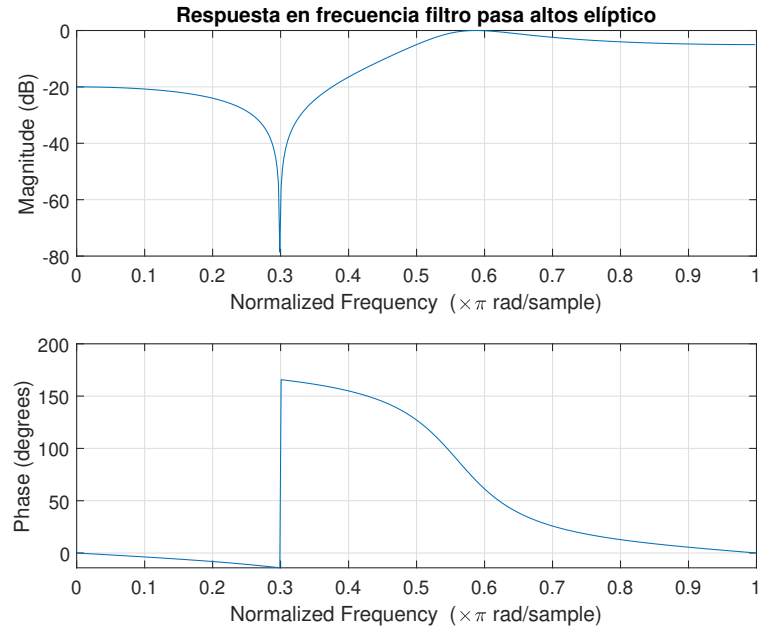


Figura 21: Respuesta en frecuencia de filtro pasa altos diseñado usando `ellip`, de orden 2, $f_c = 4kHz$, ripple en banda de paso de 5 dB y atenuación de 20 dB en banda de rechazo.

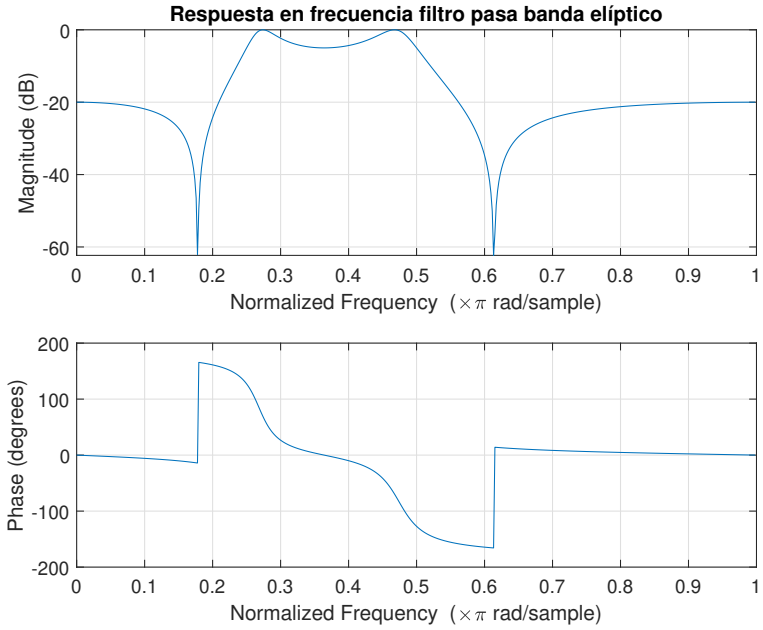


Figura 22: Respuesta en frecuencia de filtro pasa banda diseñado usando `ellip`, de orden 4, $f_1 = 2kHz$, $f_2 = 4kHz$, ripple en banda de paso de 5 dB y atenuación de 20 dB en banda de rechazo.

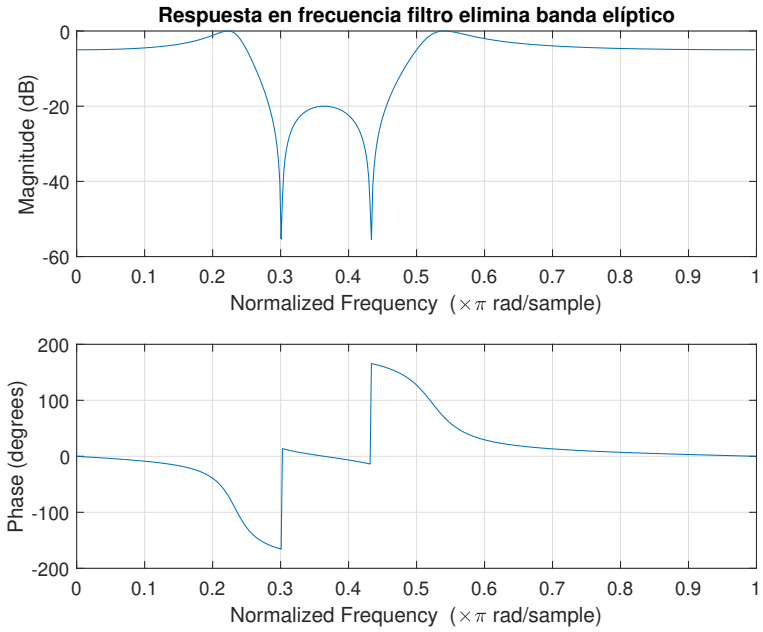


Figura 23: Respuesta en frecuencia de filtro elimina banda diseñado usando `ellip`, de orden 4, $f_1 = 2kHz$, $f_2 = 4kHz$, ripple en banda de paso de 5 dB y atenuación de 20 dB en banda de rechazo.

2. Se pide diseñar 2 filtros pasabanda usando los comandos de MATLAB `cheby1` y `cheby2`, considerando frecuencias de corte $f_1 = 2$ kHz y $f_2 = 4$ kHz, una frecuencia de muestreo de 16 kHz y orden 4.

   Para un filtro se pide 2 dB máximo de ripple en la banda de paso y para el otro 20 dB de atenuación en la banda de rechazo. Debido a que el filtro de

chebishev tipo I es un filtro que permite una región de transición corta dado un máximo de ripple en la banda de paso se uso `cheby1` para el primer filtro. Para el segundo se usó `cheby2` pues el filtro de chebishev tipo II permite asegurar un mínimo de atenuación.

La sección de código para el diseño de los filtros se muestra a continuación

```
%% 2. Diseno de filtros chebyshev tipo I y II

%orden
n2 = 4;
%frecuencias de corte
f2_1_Hz = 2000; f2_1 = f2_1_Hz/Fs;
f2_2_Hz = 4000; f2_2 = f2_2_Hz/Fs;
%Restricciones de Ripple
Rp = 2; Rs = 20;
%Diseno de Filtros [b,a] = cheby1(n,Rp,Wp,ftype)
[b2_a, a2_a] = cheby1(n2, Rp, 2*[f2_1 f2_2], "bandpass");
[b2_b, a2_b] = cheby2(n2, Rs, 2*[f2_1 f2_2], "bandpass");
```

Las imágenes de respuesta en frecuencia para el filtro a y b diseñados se muestran en las figuras 10 y 10 respectivamente. Se aprecia que se cumplen los requisitos de diseño.

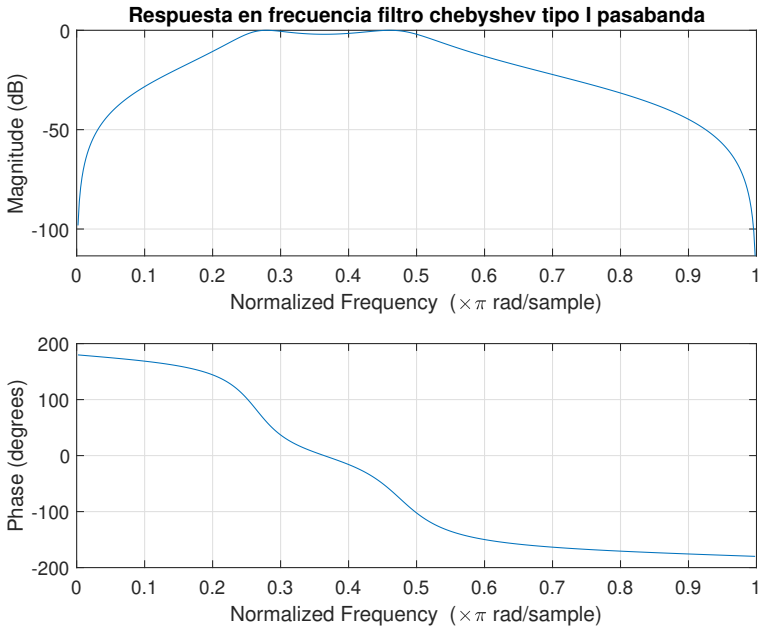


Figura 24: Respuesta en frecuencia de filtro pasa banda con ripple máximo de 2 dB en banda de paso, diseñado usando estructura de filtro chebyshev tipo I.

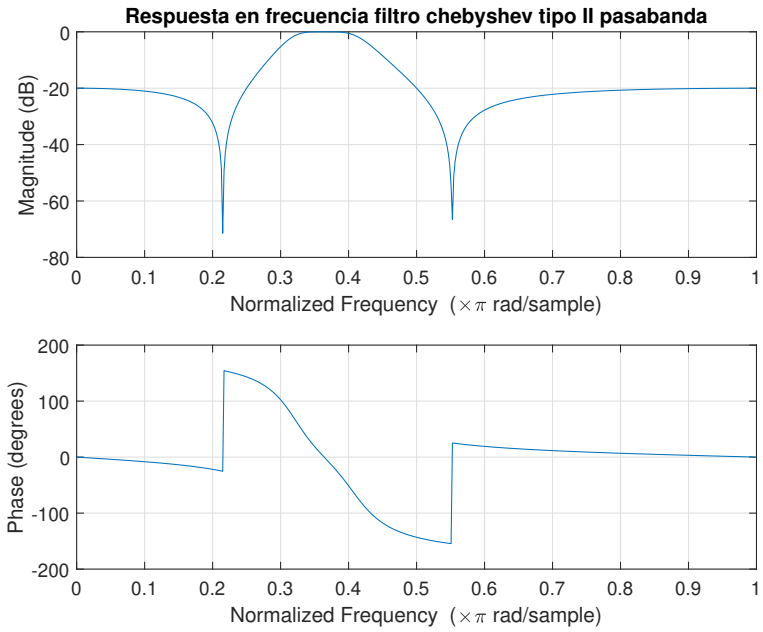


Figura 25: Respuesta en frecuencia de filtro pasa banda con atenuación mínima de 20 dB en banda de rechazo, diseñado usando estructura de filtro chebyshev tipo II.

De evaluar la respuesta en frecuencia del filtro chebyshev tipo I diseñado se obtiene que:

$$h_I(f_1 = 2000 \text{ Hz}) = h_I\left(0{,}25\frac{\pi\text{rad}}{\text{sample}}\right) \approx -32{,}76 \text{ dB}$$
$$h_I(f_1 = 4000 \text{ Hz}) = h_I\left(0{,}5\frac{\pi\text{rad}}{\text{sample}}\right) \approx -17{,}81 \text{ dB}$$

De evaluar la respuesta en frecuencia del filtro chebyshev tipo II diseñado se obtiene que:

$$h_{II}(f_1 = 2000 \text{ Hz}) = h_{II}\left(0{,}25\frac{\pi\text{rad}}{\text{sample}}\right) \approx -20{,}61 \text{ dB}$$
$$h_{II}(f_1 = 4000 \text{ Hz}) = h_{II}\left(0{,}5\frac{\pi\text{rad}}{\text{sample}}\right) \approx -23{,}87 \text{ dB}$$

3. Se pide diseñar un filtro butterworth pasa banda de orden 4, con frecuencias de corte $f_1 = 800$ Hz y $f_2 = 1600$ Hz usando el comando `butter`, considerando una frecuencia de muestreo de 16 kHz. La sección de código se muestra a continuación:

```
%%% 3. Diseno de filtros butterworth

%orden
n3 = 4/2;
%frecuencias de corte
f3_1_Hz = 800 ; f3_1 = f3_1_Hz/Fs;
f3_2_Hz = 1600; f3_2 = f3_2_Hz/Fs;
%D i s e   o   de Filtros
[b3, a3] = butter(n3, 2*[f3_1 f3_2], "bandpass");
```

La respuesta en frecuencia del filtro diseñado se muestra en la figura 26.

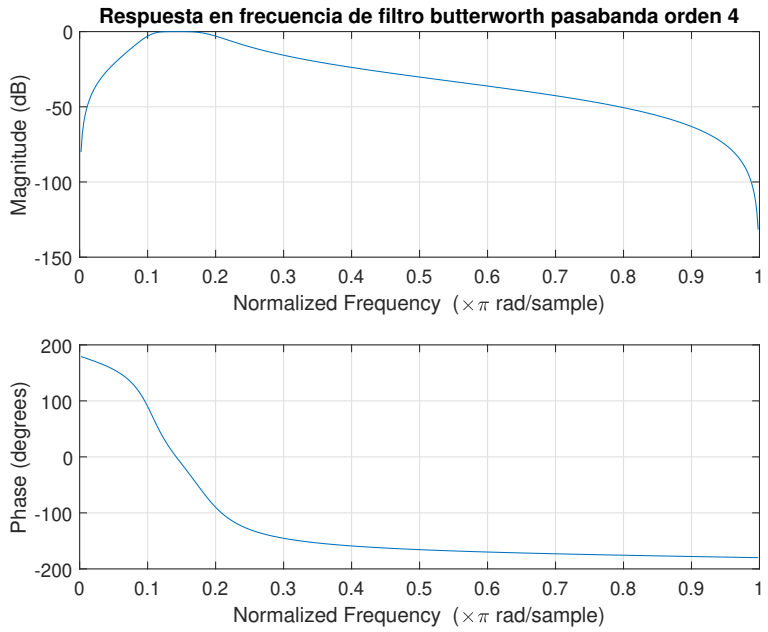


Figura 26: Respuesta en frecuencia de filtro butterworth pasa banda de orden 4 diseñado.

Posteriormente se hace un análisis de que ocurre con la ubicación de polos y ceros al diseñar filtros butterworth a medida que se aumenta el orden. La sección de código para aquello se muestra a continuación:

```
%Analisis de polos y ceros butterworth
n = [4 6 8 10 12 14];
figure
for i = 1:6
    [z,p,~] = butter(n(i)/2, 2*[f3_1 f3_2], "bandpass");
    subplot(3,2,i); zplane(z,p);
    title({"Diagrama de polos y ceros para filtro ", strcat(
        "butterworth pasabandas de orden ", int2str(n(i)))})
end
```

Los diagramas de polos y ceros se muestran en la figura 27. Notar que a medida que aumenta el orden del filtro los polos dominantes se acercan más al círculo unitario. Lo anterior podría volver el sistema inestable por temas numéricos.

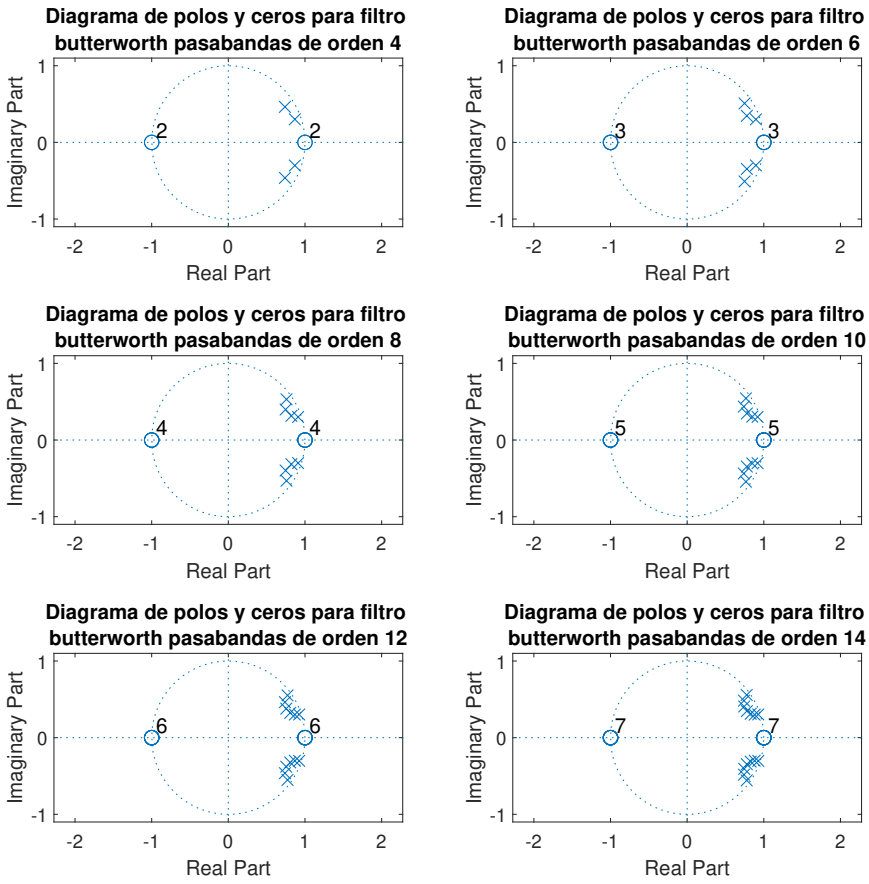


Figura 27: Diagramas de polos y ceros de filtros butterworth pasa banda de orden 4, 6, 8, 10, 12 y 14.

# 5. Comparación de diseños óptimos de filtros FIR e IIR

Se diseñan filtros IIR del tipo Butterworth, Chebyshev, y Elliptic, además de un filtro FIR de tipo equiripple que cumplan con las especificaciones: Frecuencia de banda de paso $fp = 2{,}4\ kHz$ ,frecuencia de corte $f_{sp} = 4\ kHz$, Frecuencia de muestreo $fs = 8\ kHz$, Ripple en la banda de paso $Rp = 0,5\ dB$, rechazo en frecuencias superiores a la frecuencia de corte $Rs = 40dB$.

Haciendo uso de los comandos recomendados de MATLAB `buttord`, `cheb1ord`, `ellipord`, y `firpmord` se identifican como orden óptimo den cada caso para las especificaciones dadas los que se presentan en el cuadro 2

| Filtro | FIR/IIR | Orden óptimo |
|---|---|---|
| Butterworth | IIR | 11 |
| Chebyshev | IIR | 6 |
| Elliptic | IIR | 4 |
| equiripple | FIR | 16 |

Cuadro 2: Cuadro resumen de los órdenes óptimos para que cada tipo de filtro estudiado cumpla con las especificaciones dadas.

El filtro IIR de que requiere menor orden es el de tipo Elliptic, el que le sigue es el de tipo Chebyshev y finalmente el de tipo Butterworth. En el caso del filtro FIR, se puede concluir que este tipo de filtro requiere mayor orden en general para cumplir con las mismas especificaciones que un tipo IIR.

# 6. Implementación en s-Functions: Filtro Biquad

1. Se desarrolla en lenguaje C una s-Function que implementa un filtro Notch con estructura Biquad basado en la estructura entregada como recurso para el laboratorio que se muestra a continuación

```
1     typedef struct bqState_t {
2     double bqA1;
3     double bqA2;
4     double bqB0;
5     double bqB1;
6     double bqB2;
7     double bqInput[3];
8     double bqOutput[3];
9  } bqState_t;
```

Esta estructura contiene los coeficientes de los polinomios A y B de la función de transferencia de un filtro Biquad de la forma

$$H(z) = \frac{B(z)}{H(z)} = \frac{b_0 + b_1 z^1 + b_2 z^{-2}}{1 + a_1 z^1 + a_2 z^{-2}}$$

Como lo que se desea implementar es un filtro de tipo Notch, la función de transferencia toma la forma

$$H_{BSF} = \frac{1-d}{2} \quad \frac{1 - z^{-1}}{1 - (1+d)cos(\theta)z^{-1} + d^{-z}} \tag{2}$$

Donde los parámetros $\theta = \frac{2\pi\ f_0}{f_s}$, siendo $f_0$ la frecuencia central de la banda que se desea suprimir con el filtro y $f_s$ la frecuencia de muestreo de la señal que se pretende filtrar.

El parámetro $d$ es el parámetro que permite ajustar el ancho de banda $B_w$ del filtro con la relación

$$cos(B_w)d^2 - 2\ d + cos(B_w) = 0 \tag{3}$$

Donde $B_w$ corresponde al ancho de banda normalizado con la relación

$$B_w = \frac{2\pi\ (f_{c2} - f_{c1})}{f_s}$$

Al resolver esta ecuación cuadrática en función del ancho de banda que se desea, obtienen dos raíces del polinomio y se debe escoger aquella que se encuentre dentro del circulo unitario para mantener la estabilidad del sistema ( $|d| < 1$).

Con estas relaciones y considerando $f_0 = 440$, un ancho de banda de 100 $Hz$ y $fs = 16\ kHz$ se pueden calcular los coeficientes para el filtro Notch para filtrar el archivo de audio *alternate_tones_16_16.wav*. Se tiene entonces que

$$\theta = \frac{2\pi \cdot 440\ Hz}{16000\ Hz} = 0{,}172787$$

Y por otro lado

$$B_w = \frac{2\pi\ 100\ Hz}{16000\ Hz} = 0{,}0392699 \Rightarrow d = 0{,}9614814516$$

$$a_0 = 1$$
$$a_1 = -(1-d)cos(\theta) = -1{,}9322$$
$$a_2 = d = 0{,}9614814516$$
$$b_0 = \frac{1+d}{2} = 0{,}9807407258$$
$$b_1 = -2cos(\theta) \cdot \frac{1+d}{2} = -1{,}932273671$$
$$b_2 = \frac{1+d}{2} = 0{,}9807407258$$

Ya con estos coeficientes se puede definir la estructura asociada a este filtro

```
1          bqState_t Notch440 = {
2                          -1.932273671,  // a1
3                          0.9614814516, // a2
4                          0.9807407258, //b0
5                          -1.932273671, //b1
6                          0.9807407258,  //b2
7
8                          {0 ,0 ,0}, //Inputs buffer
9                          {0,0,0} //Outputs buffer
10                       };
```

La función que ejecuta el filtrado de la señal haciendo uso del filtro diseñado se implementó con el codigo que se muestra a continuación

```
1  static double filterBiquad(bqState_t *filterNState,
2                             double filterInput){
3
4  //Desplazamiento de datos en la linea de retardo de tama o 3
5  filterNState->bqInput[2] = filterNState->bqInput[1];
6  filterNState->bqInput[1] = filterNState->bqInput[0];
7  filterNState->bqInput[0] = filterInput;
8
9  filterNState->bqOutput[2] = filterNState->bqOutput[1];
10 filterNState->bqOutput[1] = filterNState->bqOutput[0];
11
12 //y[n] = -a1*y[n] -a2*y[n-2] + b0*x[n] + b1*x[n-1] + b2*x[n-2]
13
14 double w =  filterNState->bqB0*filterNState->bqInput[0]
15             + filterNState->bqB1*filterNState->bqInput[1]
16             + filterNState->bqB2*filterNState->bqInput[2];
17
18 double y = w
19          - filterNState->bqA1*filterNState->bqOutput[1]
20          - filterNState->bqA2*filterNState->bqOutput[2];
21
22 filterNState->bqOutput[0] = y;
23 return y;
24 }
```

Y el llamado a dicha función para poder filtrar

```
1 extern double notch(double data){
2     return filterBiquad(&Notch440 , data);
3 }
```

Al aplicar el filtro diseñado a la señal de audio *alternate_tones_16_16.wav* y graficar la salida en el tiempo se obtiene la gráfica presente en la figura 28

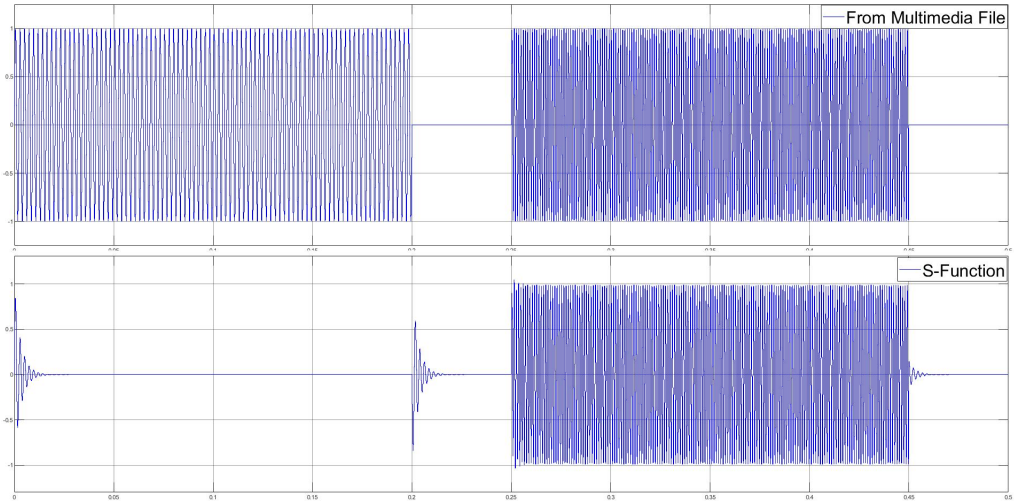


Figura 28: Gráfica en el tiempo de la señal de audio *alternate_tones_16_16.wav* antes y después de ser filtrada con ancho de banda de 100 $Hz$

Se diseñan nuevos filtros Notch que realicen la misma tarea pero esta vez con un ancho de banda de 50 $Hz$ y 200 $Hz$ respectivamente, los resultados obtenidos se muestran en las figuras 29 y 30

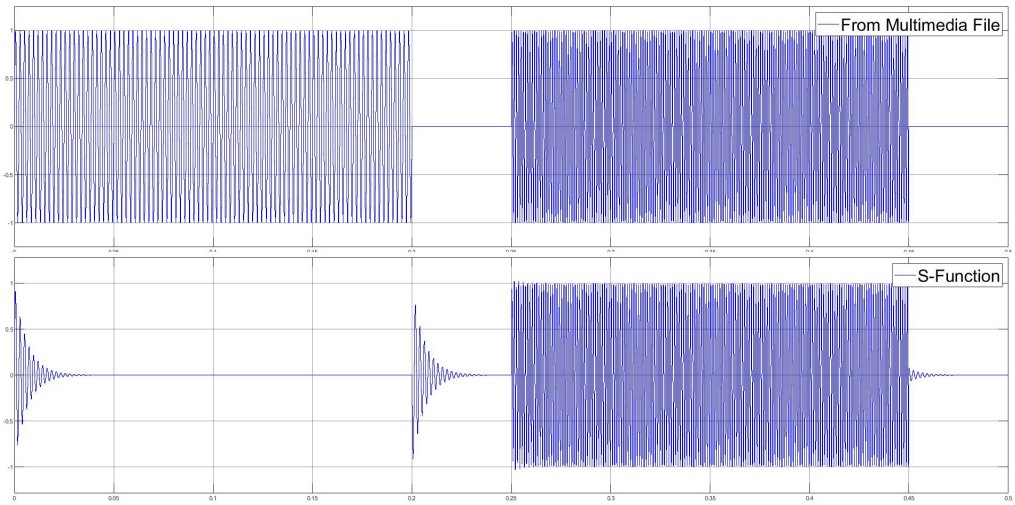


Figura 29: Gráfica en el tiempo en segundos de la señal de audio *alternate_tones_16_16.wav* antes y después de ser filtrada con ancho de banda de 50 $Hz$

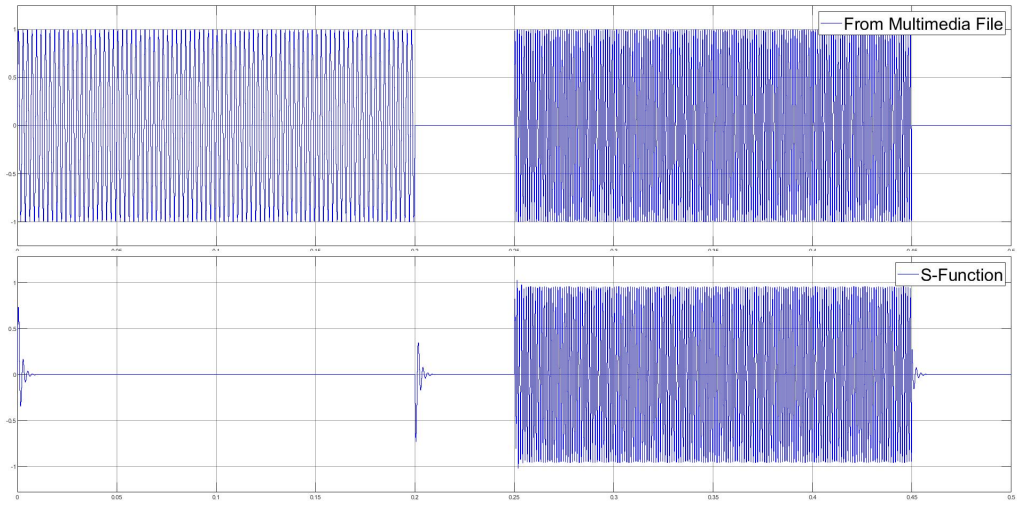


Figura 30: Gráfica en el tiempo en segundos de la señal de audio *alternate_tones_16_16.wav* antes y después de ser filtrada con ancho de banda de 200 $Hz$

Se puede observar como en todos los casos existe un transciente antes de que se realice efectivamente el filtrado, se concluir por las pruebas realizadas que dicho transciente tiene menor duración en la medida en que aumenta el ancho de banda del filtro. Esto se puede ver de mejor manera observando la respuesta a impulso de los tres filtros diseñados, las respuestas a impulso se encuentran en las figuras 31, 32 y 33 para los anchos de banda de 50 $Hz$, 100 $Hz$ y 200 $Hz$ respectivamente.

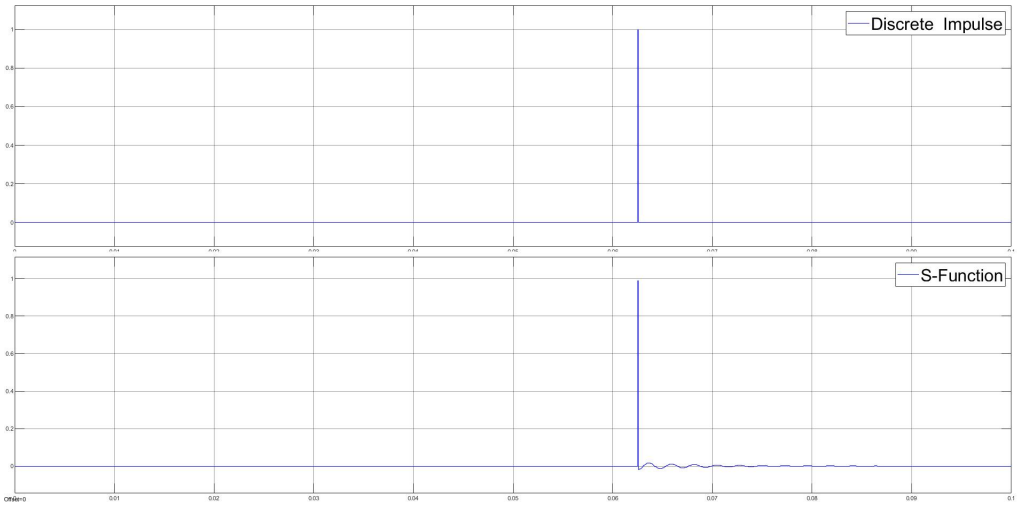


Figura 31: Gráfica en el tiempo en segundos de la respuesta impulso del filtro diseñado con ancho de banda de 50 $Hz$.

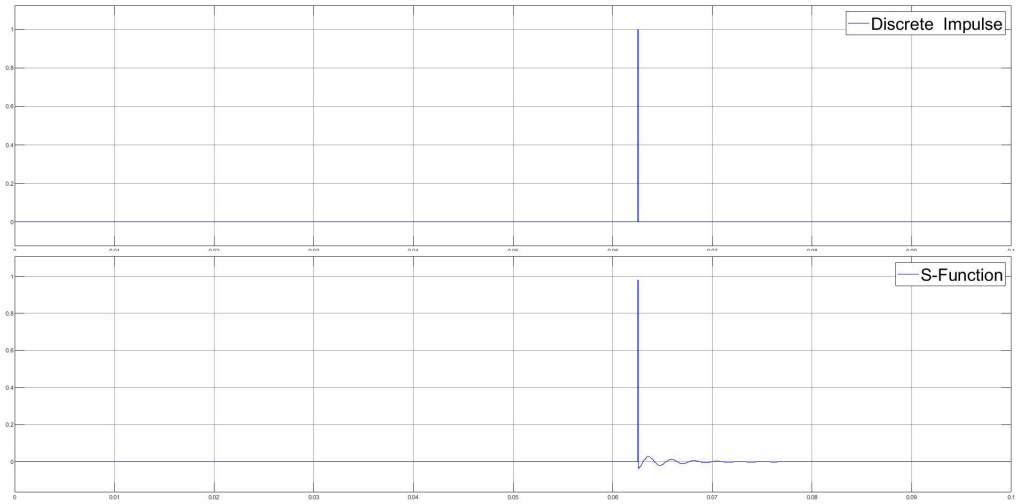


Figura 32: Gráfica en el tiempo en segundos de la respuesta impulso del filtro diseñado con ancho de banda de 100 $Hz$.

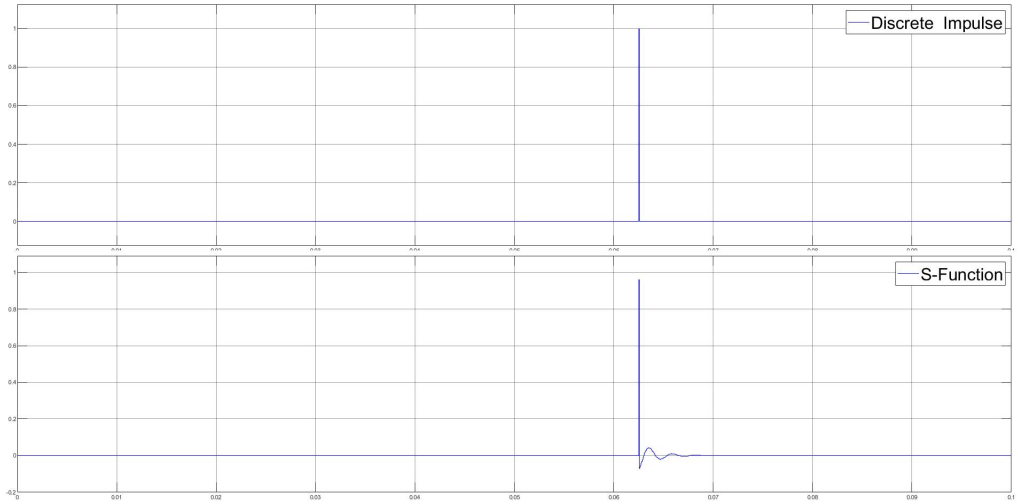


Figura 33: Gráfica en el tiempo en segundos de la respuesta impulso del filtro diseñado con ancho de banda de 200 $Hz$.

En las tres gráficas anteriores se puede comprobar lo antes mencionado sobre la relación inversa que existe entre la duración de la respuesta temporal del filtro con el ancho de banda permitido por este.

2. En esta sección se implementa filtro notch (elimina banda angosta) de segundo orden cuya frecuencia central pueda ser sintonizada en tiempo real. Para lo anterior se creó la función `filterInterface()` en C, la cual actualiza los parámetros del filtro, llama a la función `filterBiquad()` y actualiza la salida.

   La primera parte de la sección de actualización de parámetros del filtro se muestra a continuación:

```
void filterInterface(double input1, double input2,
                     double* output1) {

    //Calculo par metros filtro
    double BW    = 2 * PI * 100 / 16000;    // (BW Norm)
    double theta = 2 * PI * input2 / 16000; // (wc Norm)

    double p[] = { cos(BW), -2, cos(BW) };
    double d1 = (-p[2] + sqrt(p[2] * p[2] - 4 * p[1] * p[3]))
                / 2 * p[1];
    double d2 = (-p[2] - sqrt(p[2] * p[2] - 4 * p[1] * p[3]))
                / 2 * p[1];

    double d;
    if (abs(d1) < abs(d2)) {
        d = d1;
    }
    else {
        d = d2;
    }
```

donde se utiliza la expresión de la solución de la ecuación cuadrática característica del filtro notch (3) y se extrae la extrae la solución que está dentro del círculo unitario por estabilidad.

La segunda parte de la sección de actualización de parámetros del filtro se muestra a continuación:

```
    //Actualizacion de parametros filtro
    static bqState_t Notch_variable = {
                        0,  // a1
                        0, // a2
                        0, //b0
                        0,//b1
                        0,  //b2

                        {0, 0, 0}, //Inputs buffer
                        {0, 0, 0} //Outputs buffer
    };

    (&Notch_variable)->bqA1 = -(1 + d) * cos(theta);
    (&Notch_variable)->bqA2 = d;
    (&Notch_variable)->bqB0 = (1 + d) / 2;
    (&Notch_variable)->bqB1 = (1 + d) / 2 * -2 * cos(theta);
    (&Notch_variable)->bqB2 = (1 + d) / 2;
```

donde se utiliza la función de transferencia dependiente de $d$ (2) para obtener los coeficientes $a_i$ y $b_i$.

Finalmente se llama a la función `filterBiquad()` como se muestra a continuación con el filtro ya actualizado.

```
//Llamado a funcion de filtrado y seteo de salida
    *output1 = filterBiquad(&Notch_variable, input1);
}
```

Ya con `filterInterface()` implementada, se genera la función para crear la s-Function:

```
extern double notch(double data1, double data2)
{
    double output;
    filterInterface(data1, data2, &output);
    return output;
}
```

Posteriormente se creó un diagrama en simulink para probar la s-Function. Dicho diagrama se muestra en la figura 34. Aquí se ocupo el bloque *Repeating Sequence Interpolated* para generar la señal de control de frecuencia central, donde la señal se mantiene en 440 Hz el primer segundo, luego aumenta a razón constante hasta el segundo 3 donde se queda en 880 Hz hasta los 3.9 s (donde se termina la simulación).

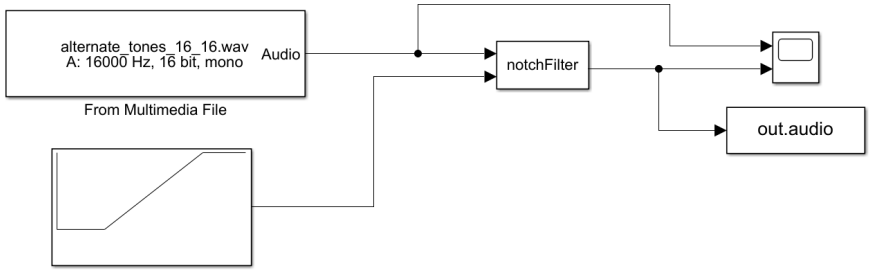


Figura 34: Diagrama en simulink para el testeo de la s-Function de notch ajustable

La señal de audio a filtrar(*alternate_tones_16_16.wav*) y la filtrada se muestra en la figura 35. Se observa que durante el primer segundo se filtra la primera señal en 440 Hz, que a partir del segundo 3 se filtra la segunda en 880 Hz y que existe una transición entre el segundo 1 y 3. Resulta interesante destacar que el filtro también afecta la amplitud de la señal que se desea dejar pasar.

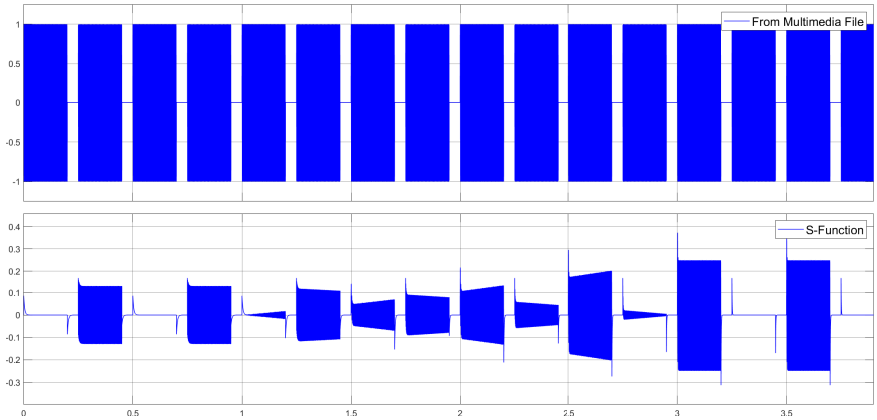


Figura 35: Amplitud [-] vs tiempo [s] de señal de entrada al filtro notch ajustable (arriba) y señal resultante del filtrado (abajo).

La señal de salida se guarda en el archivo de audio `p6_2_alternate_audio_16.wav`

el cual se adjunta en la entrega. Auditívamente se aprecia correcto funcionamiento del esquema (no percepción de la frecuencia baja al inicio y solo percepción de la frecuencia alta al final).

3. Se diseñan 7 filtros pasa banda angosta para poder identificar tonos dtmf presentes en el archivo de audio *dtmf_secuence.wav*. Para esto se deben escoger los coeficientes $a_k$ y $b_k$ asociados a la función de transferencia de la forma de un filtro Biquad que se ha estado utilizando, de manera que cada filtro permita pasar una des las frecuencias de los tonos dtmf es decir

$$\begin{aligned} BPF0: &\quad 696\ Hz \\ BPF1: &\quad 770\ Hz \\ BPF2: &\quad 852\ Hz \\ BPF3: &\quad 941\ Hz \\ BPF4: &\quad 1209\ Hz \\ BPF5: &\quad 1336\ Hz \\ BPF6: &\quad 1447\ Hz \end{aligned}$$

Se desarrolló un script en MATLAB que permite obtener los coeficientes necesarios para filtrar cada una de las frecuencias dado un ancho de banda y una frecuencia de muestreo. Se escogió un ancho de banda de 20 $Hz$, ya que es un valor que permite aislar completamente cada frecuencia deseada asegurando que no aparezcan componentes de otras frecuencias que no le corresponden al filtro asociado. Se hicieron pruebas con anchos de banda mas holgados considerando las consecuencias en la respuesta temporal que este parámetro implica (un ancho de banda mayor genera una respuesta transciente menor) pero no se logró filtrar de manera precisa la frecuencias de interés.

El cuadro 3 muestra los coeficientes $a_k$ y $b_k$ correspondientes a cada filtro asociado a una frecuencia específica

| Frecuencia [Hz] | $a_1$ | $a_2$ | $b_0$ | $b_1$ | $b_2$ |
|---|---|---|---|---|---|
| 697 | -1.91801 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 770 | -1.90179 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 852 | -1.88170 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 941 | -1.85769 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 1209 | -1.77183 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 1336 | -1.72423 | 0.99217 | 0.00391 | 0 | -0.00391 |
| 1447 | -1.66636 | 0.99217 | 0.00391 | 0 | -0.00391 |

Cuadro 3: Cuadro resumen de los coeficientes de $a_k$ y $b_k$ de los filtros asociados a cada frecuencia dtmf.

Para implementar los filtros se creó para cada uno estructura del tipo `bqState_t` como la que se ha estado usando hasta el momento con los coeficientes asociados a cada uno de los mismos, se reutiliza la función `filterBiquad` ya implementada y se desarrolla la función `decodeDtmf` que llama a las funciones entregadas como recurso para el laboraorio para poder

ejecutar la decodificación que se busca. El código de la función `decodeDtmf` es el siguiente

```
1  extern  void decodeDtmf(double input1, int32_t *output1){
2    gDtmfTones[0] = filterBiquad(&BPF0,input1);
3    gDtmfTones[1] = filterBiquad(&BPF1,input1);
4    gDtmfTones[2] = filterBiquad(&BPF2,input1);
5    gDtmfTones[3] = filterBiquad(&BPF3,input1);
6    gDtmfTones[4] = filterBiquad(&BPF4,input1);
7    gDtmfTones[5] = filterBiquad(&BPF5,input1);
8    gDtmfTones[6] = filterBiquad(&BPF6,input1);
9
10   *output1 =   dtmfDetection(gDtmfTones);
11 }
```

Al simular el filtrado de la señal de audio *dtmf_secuence.wav* con los filtros implementados se obtiene la gráfica que se muestra en la figura 36

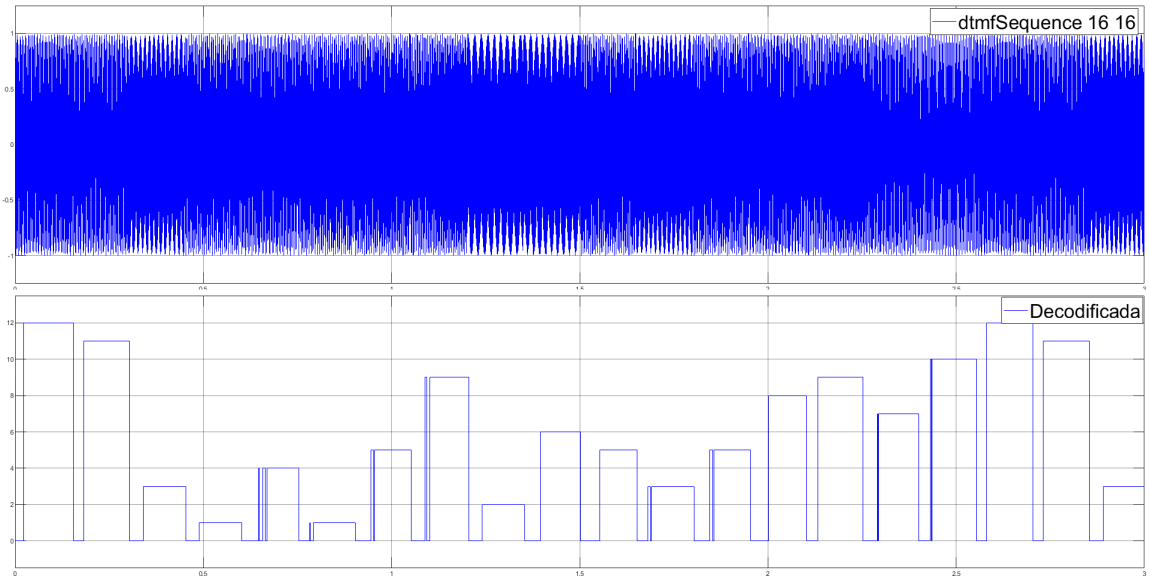


Figura 36: Gráfica en el tiempo en segundos de la simulación de filtrado de la señal de audio *dtmf_secuence.wav* con 7 filtros BPF en paralelo.

La gráfica superior muestra el archivo de audio en el tiempo y la gráfica inferior muestra el resultado obtenido luego de filtrar la señal, donde se puede reconocer la secuencia dtmf era `#-*-3-1-4-1-5-9-2-6-5... - #`.

# 7. Fe de erratas

## 7.1. Etiquetas de imagenes con errores

- Figura 2 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 3 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 6 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 7 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 8 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 9 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 10 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 12 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 13 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***
- Figura 14 etiqueta en el eje x: ***rad/s***. Debería ser eje x: ***rad/muestra***