Práctica 3

Table of Contents

Introducción 1
Diezmado por un factor entero 1
Apartado a) 1
Apartado b) 2
Apartado c) 2
Apartado d ,e y f) 3
Apartado g) 4
Apartado h) 5
Apartado i) 6
Interpolación por un factor entero 7
Apartado a, b y c) 7
Apartado d) 7
Apartado e) 8
Apartado f) 8
Apartado g) 8
Apartado h) 9
Apartado i) 10
Cambio de la frecuancia de muestreo por un factor facional 11
Apartado a) 11
Apartado b) 11
Apartado c) 11
Apartado d) 11
Apartado e) 12
Apartado f) 12
Apartado g) 12
Apartado h) 13

Teresa González y Miguel Oleo

Introducción

En esta práctica utilizaremos los conceptos dados en teoría de cambio de velocidad de muestreo en el dominio discreto: diezmado por factor entero, interpolación por un factor entero y cambio de frecuencia de muestreo por un factor racional con la combinación de ambos. El objetivo de la misma es entender mejor dichos conceptos y observar mediante la programación del código y las gráficas resultantes qué pasa utilizando las distintas técnicas (por ejemplo el espectro resultante), comprobando así lo dado en teoría. Además, esta práctica ayuda a afianzar que si se tiene una necesidad de modificar la velocidad de muestreo de un sistema es mucho más sencillo hacerlo en el dominio discreto sin tener que pasar por el analógico.

Diezmado por un factor entero

Apartado a)

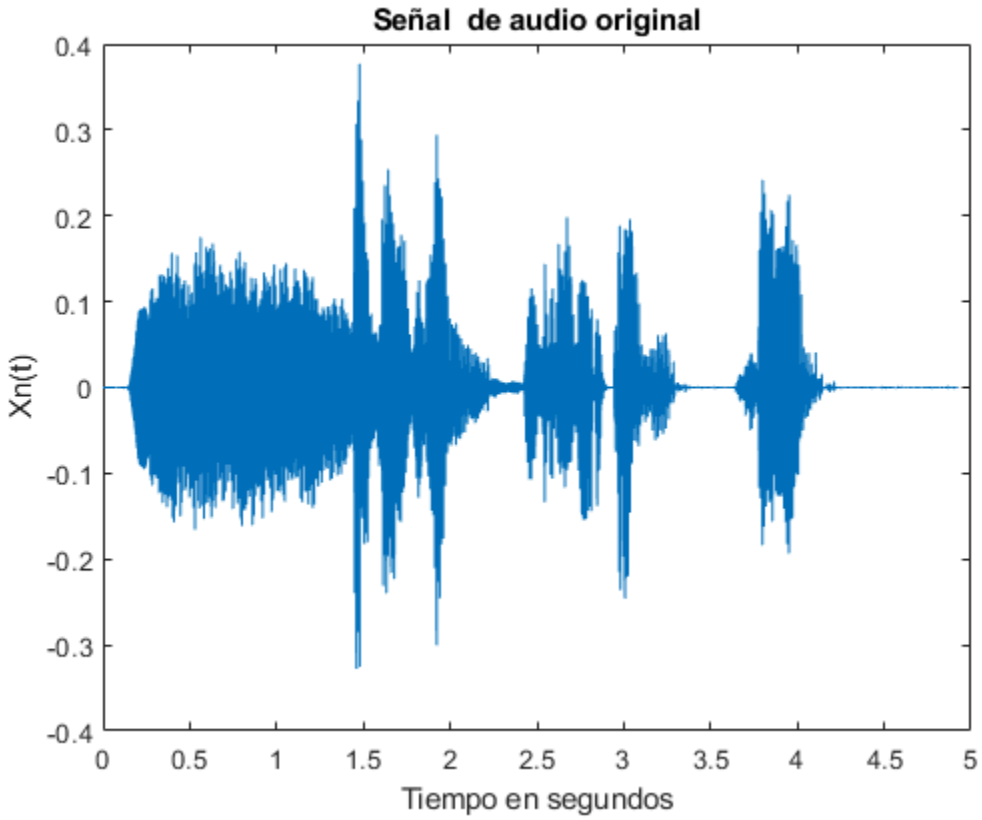
Cargamos el archivo y lo reproducimos.

```
[Xn, fs] = audioread('PDS_P3_LE2_G3.wav');
sound(Xn,fs);
```

Apartado b)

Creamos el vector de tiempos, tal y como lo hicimos en la práctica anterior. Con este vector podemos representar la señal Xn en el tiempo. Se sigue apreciando un pequeño ruido al principio y al final de la señal.

```
dt = 1/fs;
t= 0:dt:((length(Xn)-1)/fs);
figure();
plot(t,Xn)
title('Señal  de audio original')
xlabel('Tiempo en segundos')
ylabel('Xn(t)')
```

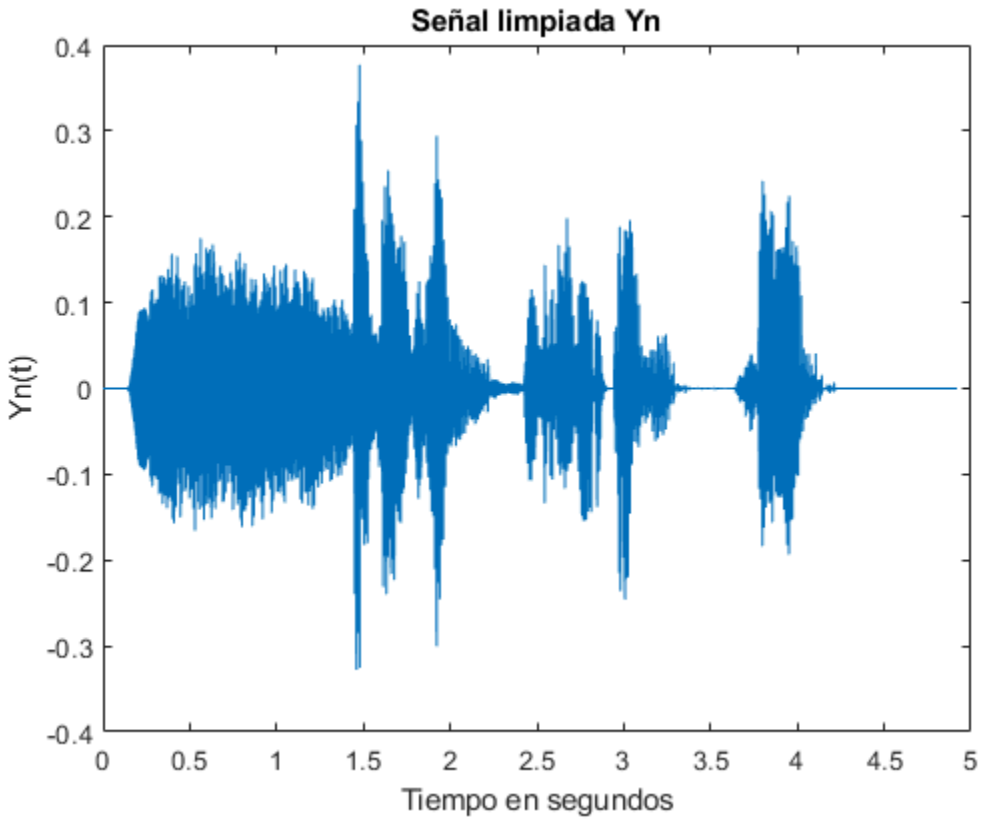


Apartado c)

Creamos Yn a partir de Xn y eliminamos el ruido mencionado en el apartado anterior. Mostramos la señal limpiada en el tiempo, donde se puede observar que dicho ruido ha sido eliminado.

```
Yn = Xn;
index1 = find(t <= 0.1412);
index2 = find(t >= 4.2303);
Yn(index1)=0;
Yn(index2)=0;
```

```
figure;
plot(t,Yn)
title('Señal limpiada Yn')
xlabel('Tiempo en segundos')
ylabel('Yn(t)')
```

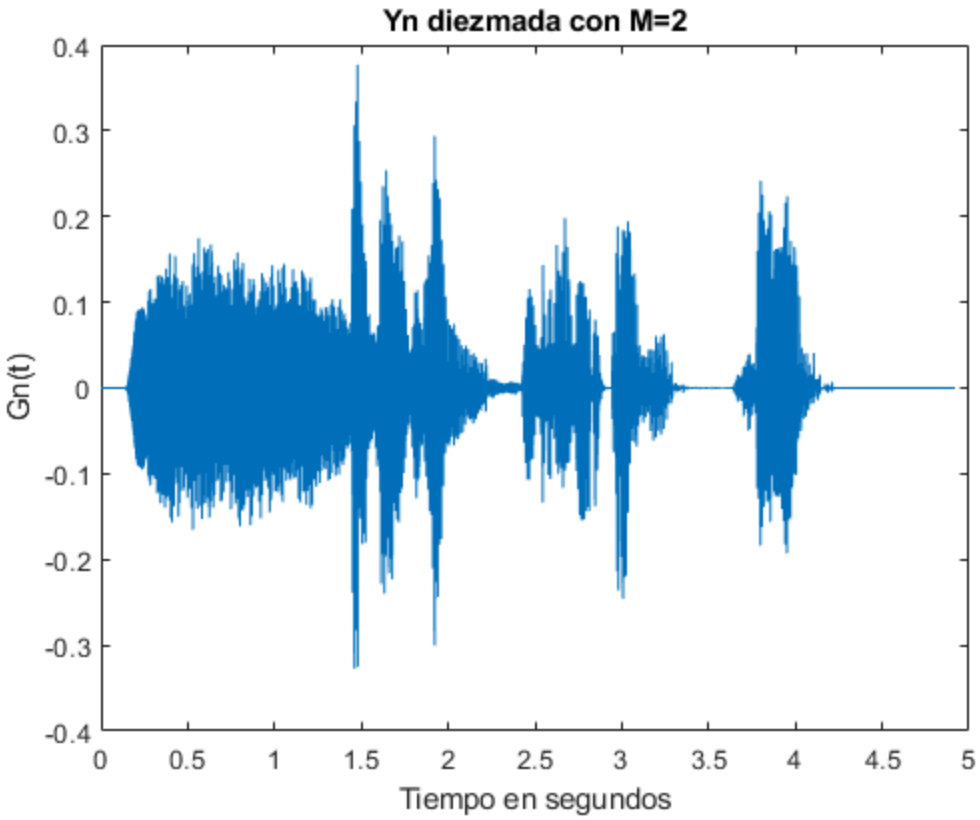


Apartado d ,e y f)

Creamos la función Diezmador que recibe M (factor de diezmado) y Yn (señal a diezmar). Esta función devuelve Gn, la cual es la señal Yn diezmada por M. Diezmar por un factor M, significa que la nueva señal se genera a partir de 1 de cada M muestras de la señal original.

Para poder representar esta señal, hay que calcular la nueva frecuencia de muestreo (=fs(original)/M) y crear el nuevo vector de tiempos.

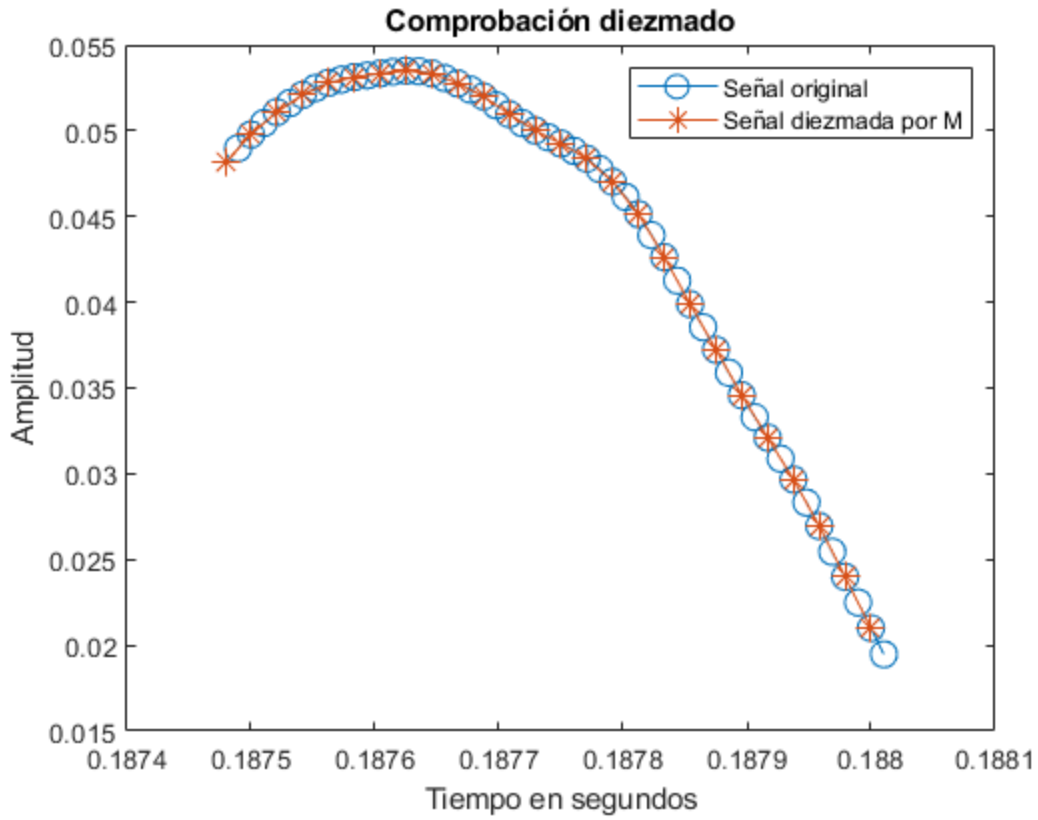
```
M = 2;
Gn = Diezmador(M,Yn);
fs_g = fs/M;
dt2 = 1/fs_g;
t_g = 0:dt2:((length(Gn)-1)/fs_g);
figure()
plot(t_g,Gn)
title('Yn diezmada con M=2')
ylabel('Gn(t)')
xlabel('Tiempo en segundos')
```



Apartado g)

Para poder ver gráficamente este proceso de diezmado, representamos una pequeña muestra superpuesta de las señales con distintos marcadores. En nuestro caso, como M=2, veremos que la señal diezmada toma una muestra de cada dos de la original.

```
vect_1 = 18000:18050;
vect_2 = (18000/M):(18050/M);

figure()
plot(t(vect_1),Yn(vect_1),'-o','MarkerSize',10)
hold on
plot(t_g(vect_2),Gn(vect_2),'-*','MarkerSize',10)
title('Comprobación diezmado');
xlabel('Tiempo en segundos')
ylabel('Amplitud')
legend('Señal original','Señal diezmada por M')
```



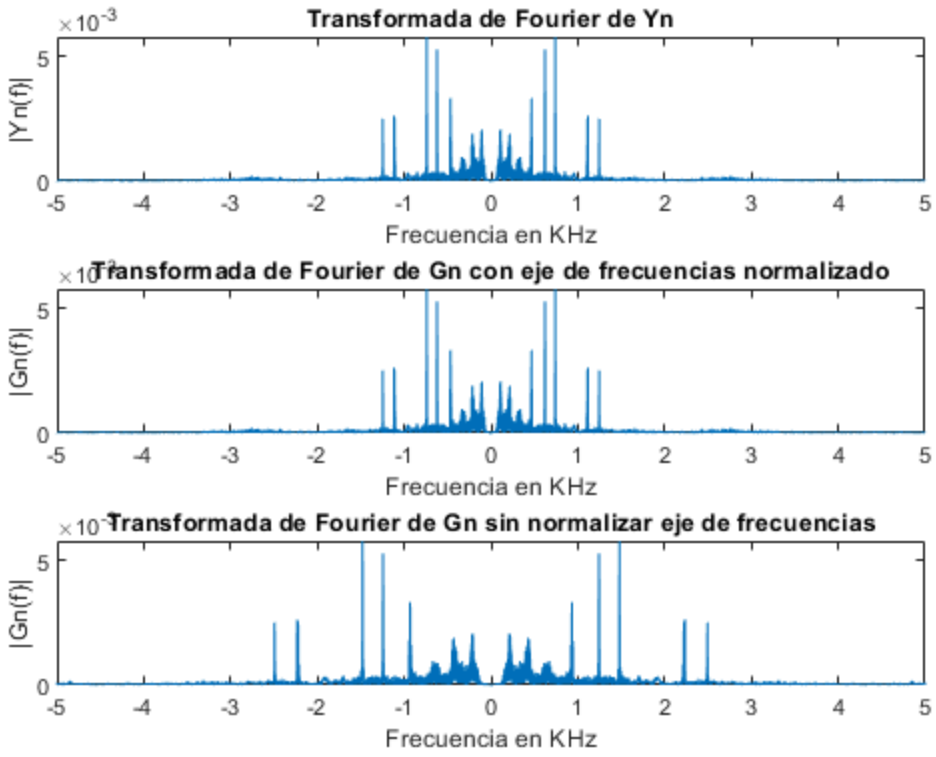
Apartado h)

A continuación se muestran los espectros de las señales original y diezmadas (con eje de frecuencias fs/M es decir, normalizados) y por último, la misma señal diezmada pero sin normalizar el eje de frecuencias con su nueva fs.

En los plot se puede apreciar como en la última gráfica el espectro se ha ensanchado por un factor M. En la gráfica en la que está normalizado con fs/M se puede apreciar mejor como es la misma señal y por ello, si quisieramos recuperar la misma señal, tendríamos que muestrear de nuevo a una fs' = fs/M.

```
figure()
subplot(3,1,1)
Yf = (fft(Yn,length(Yn)))/length(Yn);
f_y = linspace(-fs/2,fs/2,length(Yn));
plot(f_y/1000,fftshift(abs(Yf)));
xlim([-5 5]);
title('Transformada de Fourier de Yn')
xlabel('Frecuencia en KHz')
ylabel('|Yn(f)|')

subplot(3,1,2)
Gf = (fft(Gn,length(Gn)))/length(Gn);
f_g = linspace(-fs_g/2,fs_g/2,length(Gn));
plot(f_g/1000,fftshift(abs(Gf)));
xlim([-5 5]);
```

```
title('Transformada de Fourier de Gn con eje de frecuencias
 normalizado')
xlabel('Frecuencia en KHz')
ylabel('|Gn(f)|')


subplot(3,1,3)
f_g2 = linspace(-fs/2,fs/2,length(Gn));
plot(f_g2/1000,fftshift(abs(Gf)));
xlim([-5 5]);
title('Transformada de Fourier de Gn sin normalizar eje de
 frecuencias')
xlabel('Frecuencia en KHz')
ylabel('|Gn(f)|')
```



Apartado i)

En ese caso, al pasar por el diezmador (fs2 = fs1/2), por lo que si hay señal estre fs1/2 y fs1/4 (fs2/2), se producirá aliasing. Esto concuerda con lo visto en teoría, donde vimos que si la señal tiene componentes por encima de fs1/(2*M), al diezmar habrá aliassing.

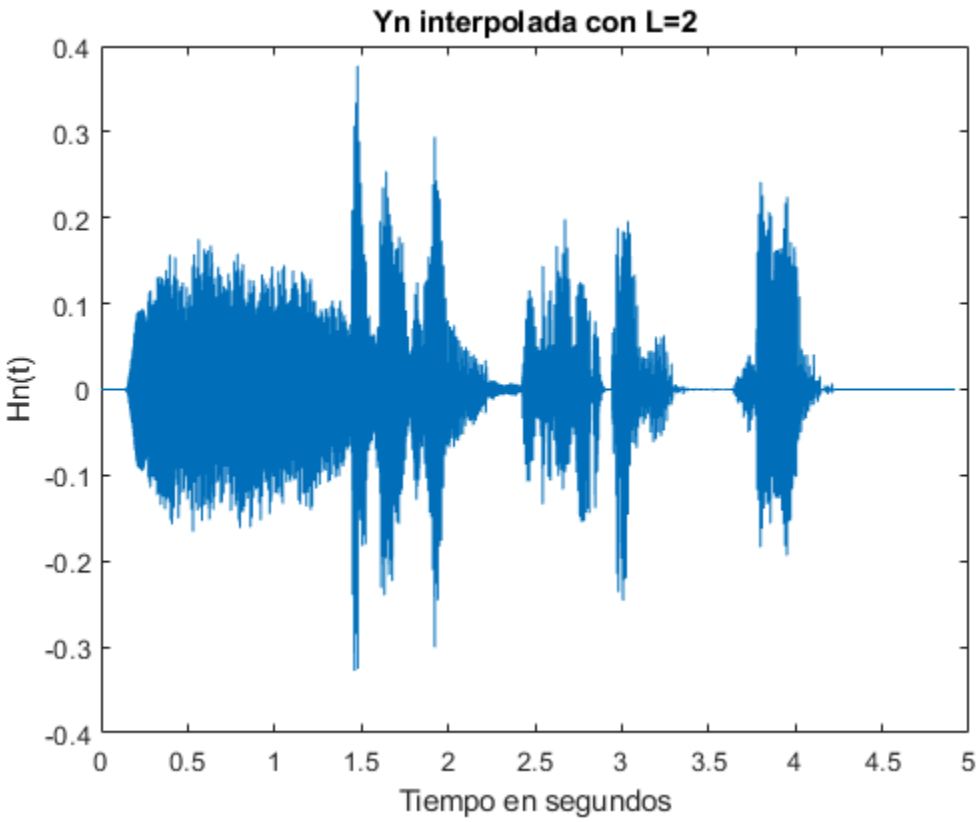
Para solucionar esto, proponemos poner un filtro de ancho de banda f1/4 (fs2/2) antes de diezmar.

Interpolación por un factor entero

Apartado a, b y c)

Creamos una funcion Interpolador, que recibe L (factor de interpolación) e Yn (señal original). Como resultado devuelve Hn, que es la señal original pero con L-1 ceros entre cada muestra. La nueva frecuencia de muestreo será fs1*L.

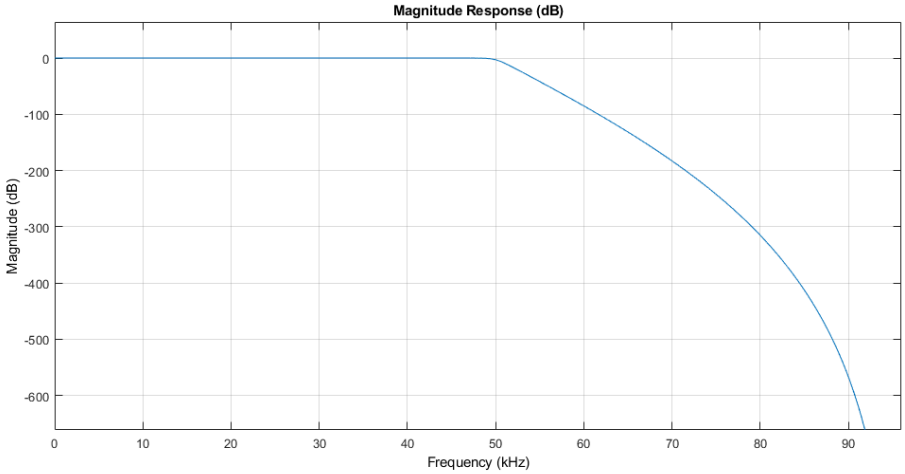
```
L = 2;
Hn = Interpolador(L,Yn);
fs_h = fs*L;
dt3 = 1/fs_h;
t_h= 0:dt3:((length(Hn)-1)/fs_h);

figure()
plot(t_h,Hn)
title('Yn interpolada con L=2')
ylabel('Hn(t)')
xlabel('Tiempo en segundos')
```



Apartado d)

```
Kn = Filtro(Hn,fs_h,L,fs_h/(2*L));
```

```
% función para quitar el retardo en tiempo del filtro
Ka = Kn;
index_1 = find(Yn~=0,1,'first')-1;
index_2 = find(Kn~=0,1,'first')-1;
index = 1:(index_2-index_1);
Ka(index)=[];
index_f = index+length(Ka);
Ka(index_f)=0;
```



Apartado e)

G=2 , porque L=2. El ancho se comprime por 1/L para que el área (potencia) se mantenga, hay que multiplicar la amplitud por L.

Apartado f)

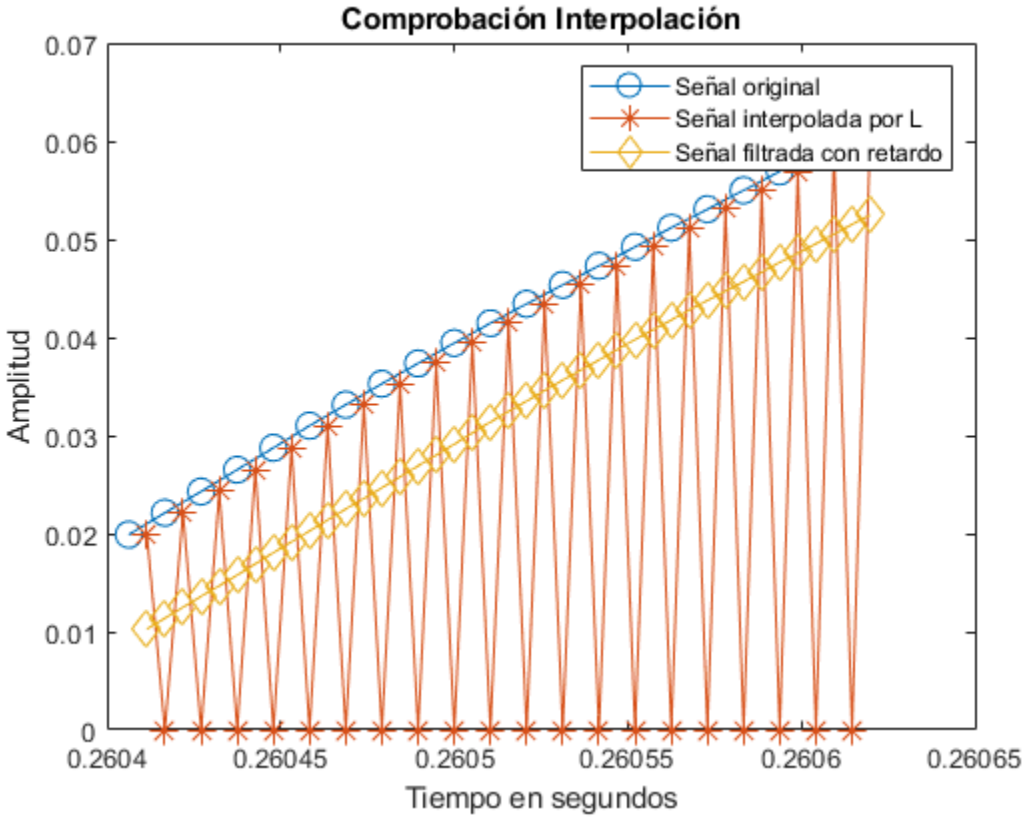
Frecuencia de corte=fs_h/(2*L), para quedarnos con el ancho de banda de interés y asegurarnos que borramos las réplicas (las cuales estáran L veces más juntas porque el espectro se comprime).

Apartado g)

A continuación se muestran las señales original, interpolada y filtrada. Se puede ver que el procesor es correcto, ya que en la señal interpolada se puede observar que introduce L-1 ceros entre cada muestra. En cuanto al filtro, también se observa que funciona correctamente, ya que vemos el retardo citado en el enunciado.

```
vect_1 = 25000:25020;
vect_2 = (25000*L):(25020*L);

figure()
plot(t(vect_1),Yn(vect_1),'-o','MarkerSize',10)
hold on
plot(t_h(vect_2),Hn(vect_2),'-*','MarkerSize',10)
hold on
```

```
plot(t_h(vect_2),Kn(vect_2),'-d','MarkerSize',10)
title('Comprobación Interpolación');
xlabel('Tiempo en segundos')
ylabel('Amplitud')
legend('Señal original','Señal interpolada por L','Señal filtrada con
 retardo')
```

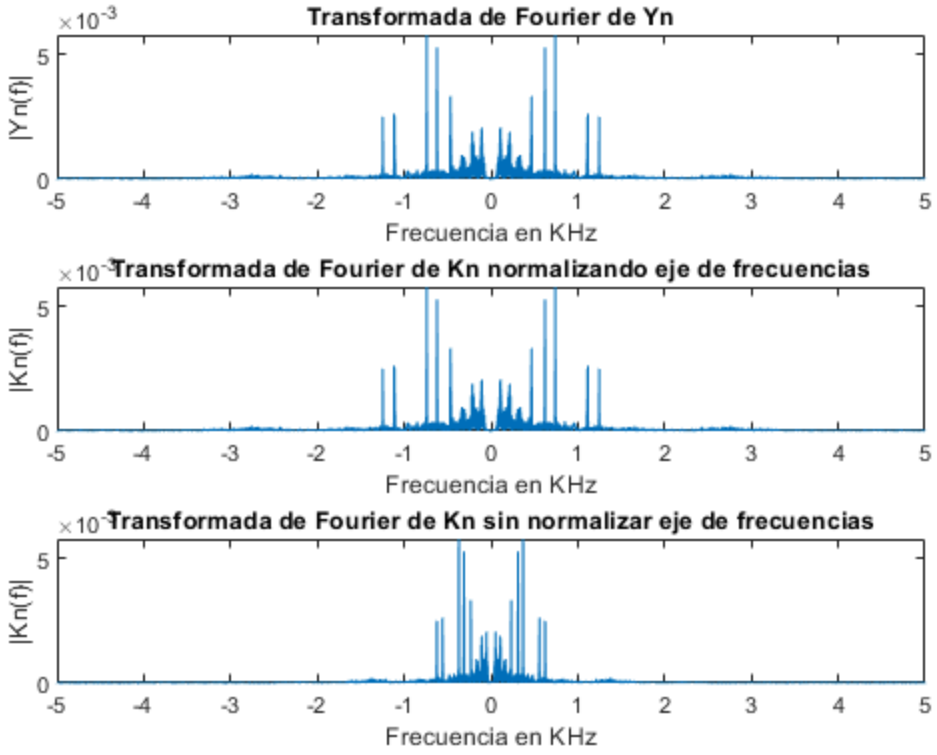


Apartado h)

El interpolador mete L-1 ceros entre cada muestra. Por ello, el espectro se ve estrechado y nos aparecen réplicas de la señal a la nueva frecuencia de muestreo. Para poder recuperar la señal original, filtramos esta señal interpolada para quitar dichas réplicas y para recuperar la amplitud de la señal original.

```
figure()
subplot(3,1,1)
Yf = (fft(Yn,length(Yn)))/length(Yn);
f_y = linspace(-fs/2,fs/2,length(Yn));
plot(f_y/1000,fftshift(abs(Yf)));
xlim([-5 5]);
title('Transformada de Fourier de Yn')
xlabel('Frecuencia en KHz')
ylabel('|Yn(f)|')

subplot(3,1,2)
Kf = (fft(Ka,length(Ka)))/length(Ka);
f_k = linspace(-fs_h/2,fs_h/2,length(Kf));
```

```
plot(f_k/1000,fftshift(abs(Kf)));
xlim([-5 5]);
title('Transformada de Fourier de Kn normalizando eje de frecuencias')
xlabel('Frecuencia en KHz')
ylabel('|Kn(f)|')

subplot(3,1,3)
Kf = (fft(Ka,length(Ka)))/length(Ka);
f_k2 = linspace(-fs/2,fs/2,length(Kf));
plot(f_k2/1000,fftshift(abs(Kf)));
xlim([-5 5]);
title('Transformada de Fourier de Kn sin normalizar eje de
 frecuencias')
xlabel('Frecuencia en KHz')
ylabel('|Kn(f)|')
```



Apartado i)

Ya que la interpolación consiste en insertar 0's en el dominio discreto, lo que se traduce en una compresión del espectro, otra alternativa es hacer la fft de la señal original (representación entre -0.5 y 0.5) y meter 0's a la derecha y la izquierda (si L=2, N/2 en cada lado, siendo N el BW del espectro original comprendido entre -0.5 y 0.5, por lo que el total del BW sería 2N). Por lo tanto, al hacer la fft inversa obtendríamos el espectro de la señal original comprimido (con la ganancia inicial), tal como si hubieramos realizado interpolación.

Cambio de la frecuancia de muestreo por un factor facional

Apartado a)

128 KHz = 4/3 * fs

```
L = 4;
M = 3;
```

Apartado b)

```
Hn = Interpolador(L,Yn);
```

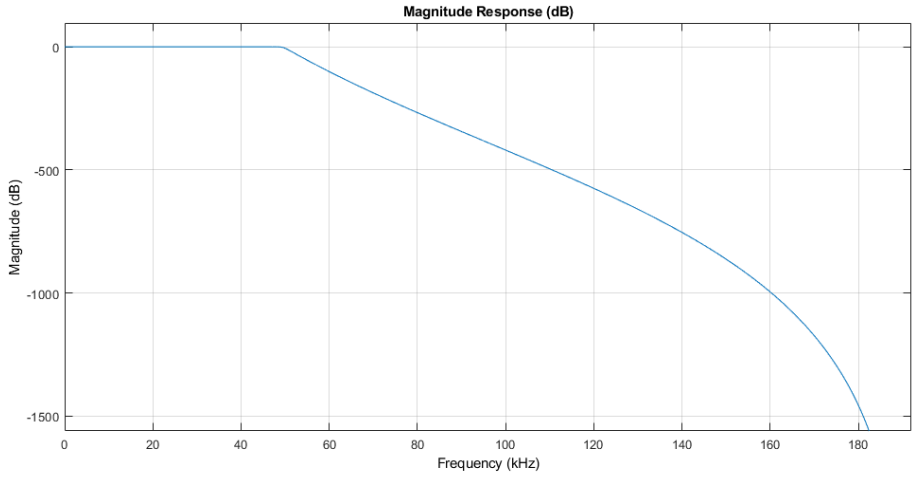
Apartado c)

```
fs_1 = fs*L;
```

Apartado d)

Los valores escogidos para filtrar la señal, siguen la misma explicación que el anterior filtro realizado.

```
Kn = Filtro(Hn,fs_1,L,fs_1/(2*L));

% función para quitar el retardo en tiempo del filtro
Ka = Kn;
index_1 = find(Yn~=0,1,'first')-1;
index_2 = find(Kn~=0,1,'first')-1;
index = 1:(index_2-index_1);
Ka(index)=[];
index_f = index+length(Ka);
Ka(index_f)=0;
```



Apartado e)

g = L fc = fs_1/(2*L)

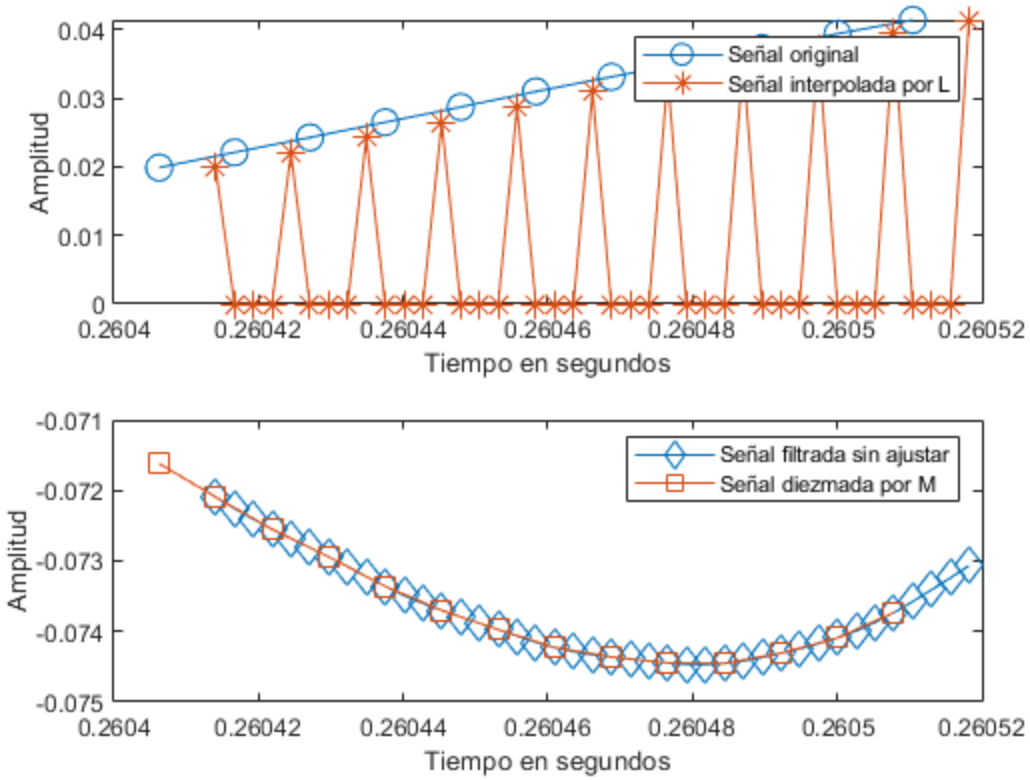
Apartado f)

```
Gn = Diezmador(M,Ka);
```

Apartado g)

A continuación se muestra la señal original, la señal interpolada por L, la señal filtrada y la señal diezmada por M. Se puede observar que el proceso es correcto, ya que al interpolar, mete L-1 (3) ceros cada muestras y al diezmar coge una de cada M (3) muestras.

```
dt3 = 1/fs_1;
t_h = 0:dt3:((length(Hn)-1)/fs_1);

fs_g = fs_1/M;
dt2 = 1/fs_g;
t_g = 0:dt2:((length(Gn)-1)/fs_g);

vect_1 = 25000:25010;
vect_2 = (25000*L):(25010*L);
vect_3 = round((25000*L/M):(25010*L/M));

figure()
subplot(2,1,1)
plot(t(vect_1),Yn(vect_1),'-o','MarkerSize',10)
hold on
plot(t_h(vect_2),Hn(vect_2),'-*','MarkerSize',10)
legend('Señal original','Señal interpolada por L')
xlabel('Tiempo en segundos')
ylabel('Amplitud')

subplot(2,1,2)
plot(t_h(vect_2),Ka(vect_2),'-d','MarkerSize',10)
hold on
plot(t_g(vect_3),Gn(vect_3),'-s','MarkerSize',10)
legend('Señal filtrada sin ajustar','Señal diezmada por M')
xlabel('Tiempo en segundos')
ylabel('Amplitud')
```



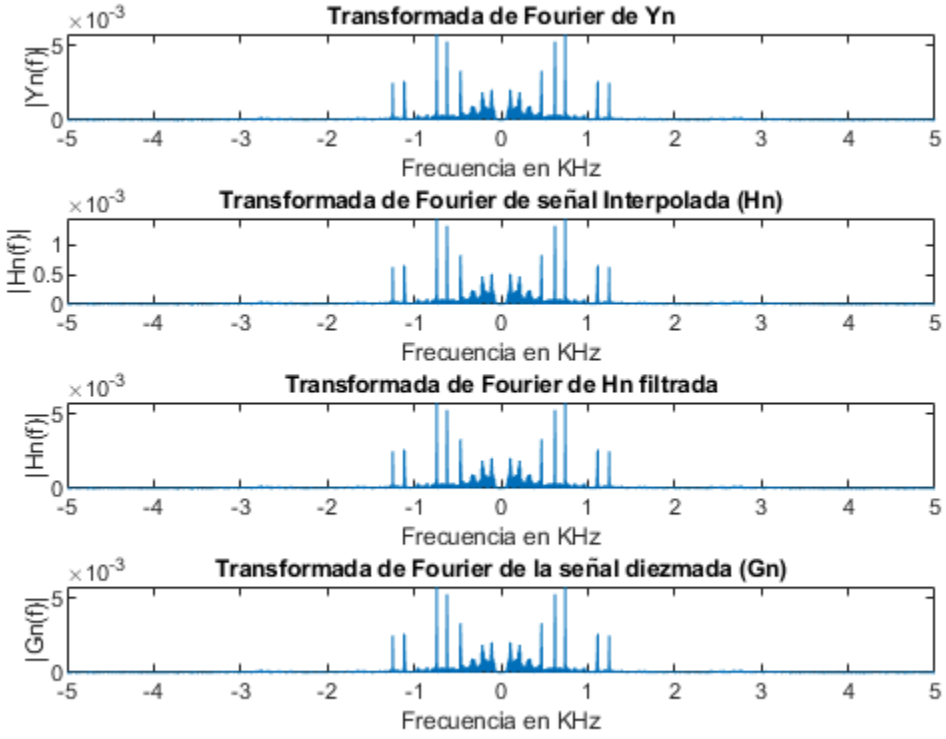
Apartado h)

Viendo los espectros de las señales originales, interpoladas, filtradas y diezmadas, podemos afirmar que el proceso ha sido correcto.

En las gráficas no se aprecia ni la expansión (diezmador), ni compresión (interpolador) del espectro. Esto se debe a que cada espectro ha sido representado respecto de su nueva frecuencia normalizada (frecuencia dividida entre su frecuencia de muestreo), por ello, todas tienen el mismo ancho de banda. Ya demostramos en plot anteriores como el espectro se ensancha al diezmar y se estrecha al interpolar.

```
figure()
subplot(4,1,1)
Yf = (fft(Yn,length(Yn)))/length(Yn);
f_y = linspace(-fs/2,fs/2,length(Yn));
plot(f_y/1000,fftshift(abs(Yf)));
xlim([-5 5]);
title('Transformada de Fourier de Yn')
xlabel('Frecuencia en KHz')
ylabel('|Yn(f)|')

subplot(4,1,2)
Hf = (fft(Hn,length(Hn)))/length(Hn);
f_h = linspace(-fs_1/2,fs_1/2,length(Hf));
plot(f_h/1000,fftshift(abs(Hf)));
xlim([-5 5]);
title('Transformada de Fourier de señal Interpolada (Hn)')
```

```
xlabel('Frecuencia en KHz')
ylabel('|Hn(f)|')

subplot(4,1,3)
Kf = (fft(Ka,length(Ka)))/length(Ka);
f_h = linspace(-fs_1/2,fs_1/2,length(Kf));
plot(f_h/1000,fftshift(abs(Kf)));
xlim([-5 5]);
title('Transformada de Fourier de Hn filtrada')
xlabel('Frecuencia en KHz')
ylabel('|Hn(f)|')

subplot(4,1,4)
Gf = (fft(Gn,length(Gn)))/length(Gn);
f_g = linspace(-fs_g/2,fs_g/2,length(Gf));
plot(f_g/1000,fftshift(abs(Gf)));
xlim([-5 5]);
title('Transformada de Fourier de la señal diezmada (Gn)')
xlabel('Frecuencia en KHz')
ylabel('|Gn(f)|')
```



Published with MATLAB® R2019b