INFORME PRÁCTICA 6 PDS

JAIME ARANA CARDELÚS

JOSÉ IGNACIO MAGDALENA

GRUPO 2 PAREJA 8

Práctica 6: Implementación de filtros LTI utilizando DFT

1. Diseño de un filtro FIR
2. Implementación de un filtro utilizando DFT
3. Análisis de resultados

Objetivos de la práctica

El objetivo de esta práctica es aplicar los conocimientos adquiridos en las clases de teoría sobre la transformada discreta de fourier y su uso práctico como el filtrado por bloques en streaming.

Diseño de un filtro FIR

Apartado a)

Cargamos la señal mediante la función **audioread()** como hemos hecho en prácticas anteriores. Como parámetros de entrada tenemos la señal de audio y como parámetros de salida nos devuelve la señal muestreada a la que hemos llamado **xn** y su frecuencia de muestreo correspondiente **fs**.

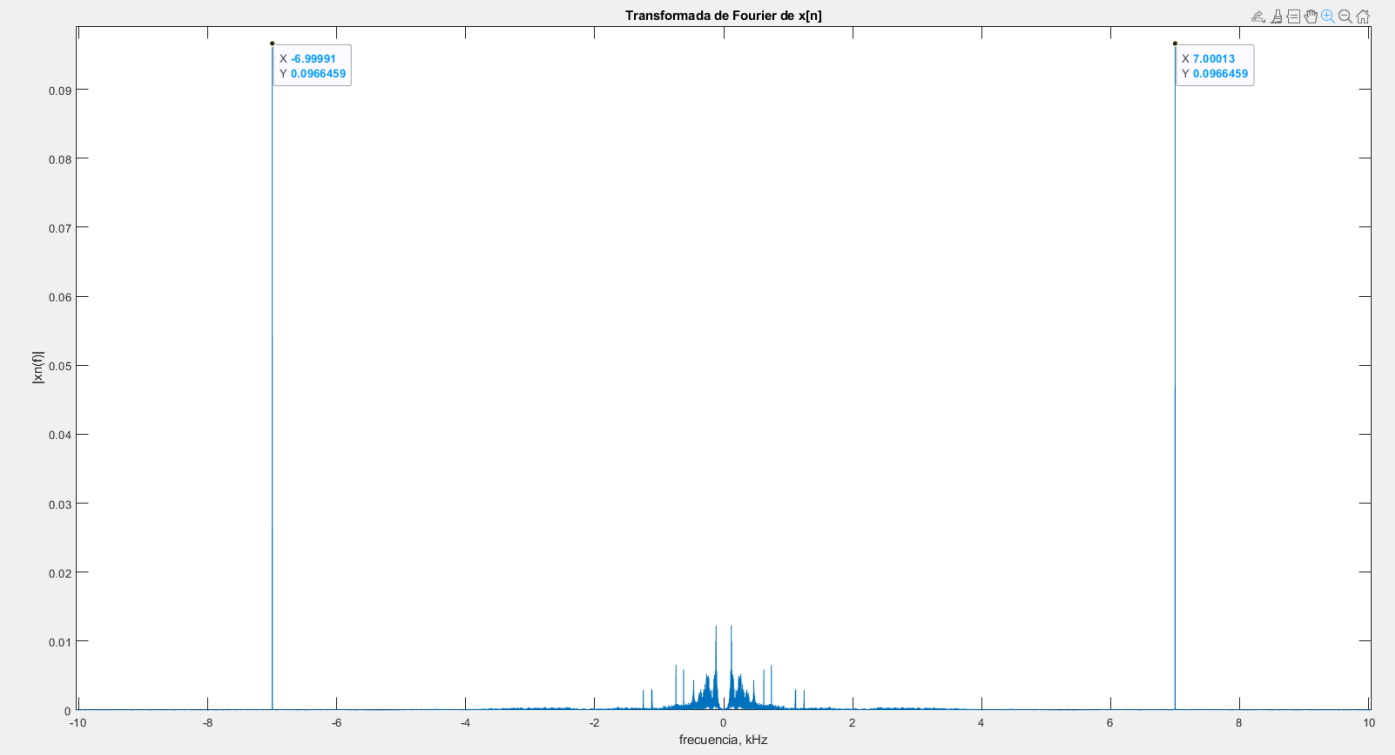
Ahora que tenemos la señal y su frecuencia de muestreo podemos reproducir la señal mediante la función **sound(x[n], fs)**.

Como avisan en el enunciado la señal de audio tiene un pitido muy agudo, el cual eliminaremos mediante el filtrado a lo largo de la práctica.

Apartado b)

Representamos la respuesta en frecuencia de nuestra señal x[n] para poder visualizar que frecuencias tenemos que eliminar con el filtro que vamos a diseñar. Dichas frecuencias que vamos a eliminar son las del pitido que se oye en la grabación.

Como podemos ver en la gráfica tenemos dos deltas en 7 kHz, que corresponden al pitido que queremos eliminar.



Inicialmente habíamos cogido una fc = 6 kHz, no obstante las deltas no se eliminaban completamente sino que solo se atenuaban ligeramente. Esto se debe a que los 6 kHz no entran completamente en la banda de rechazo del filtro. Es por esa razón que la frecuencia de corte con la que hemos diseñado el filtro es, **fc = 5 kHz**. Con esta frecuencia de corte conseguimos filtrar el pitido completamente y no eliminar nada del resto de la señal.

Para la frecuencia de muestreo del filtro cogemos la misma que la frecuencia de muestreo de nuestra señal original muestreada, **fs = 96 kHz**.

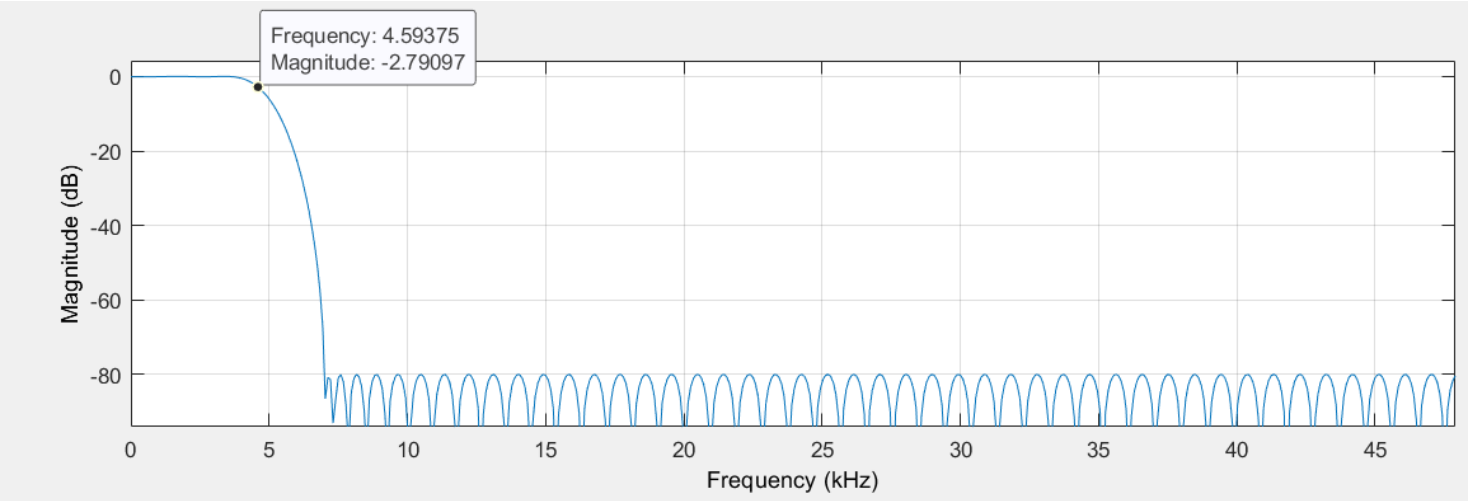
Apartado c)

Diseñamos el filtro con los parámetros especificados:

Lowpass, **FIR – Constrained Equiripple**, **M = 100**, **fs = 96 kHz**, **fc = 5 kHz**, **Apass = 0. 1 dB**, **Astop = 80 dB**

Apartado d)

Representación en escala logarítmica:



En la representación en frecuencia del filtro podemos observar que la frecuencia de corte efectivamente se encuentra en 5 kHz, ya que la atenuación cae 3 dB con respecto a la banda de paso en 5 kHz.

Además se puede comprobar que es un filtro paso bajo ya que la banda de paso se encuentra en las frecuencias más bajas y las banda de rechazo en las altas.

Comprobación del filtrado:

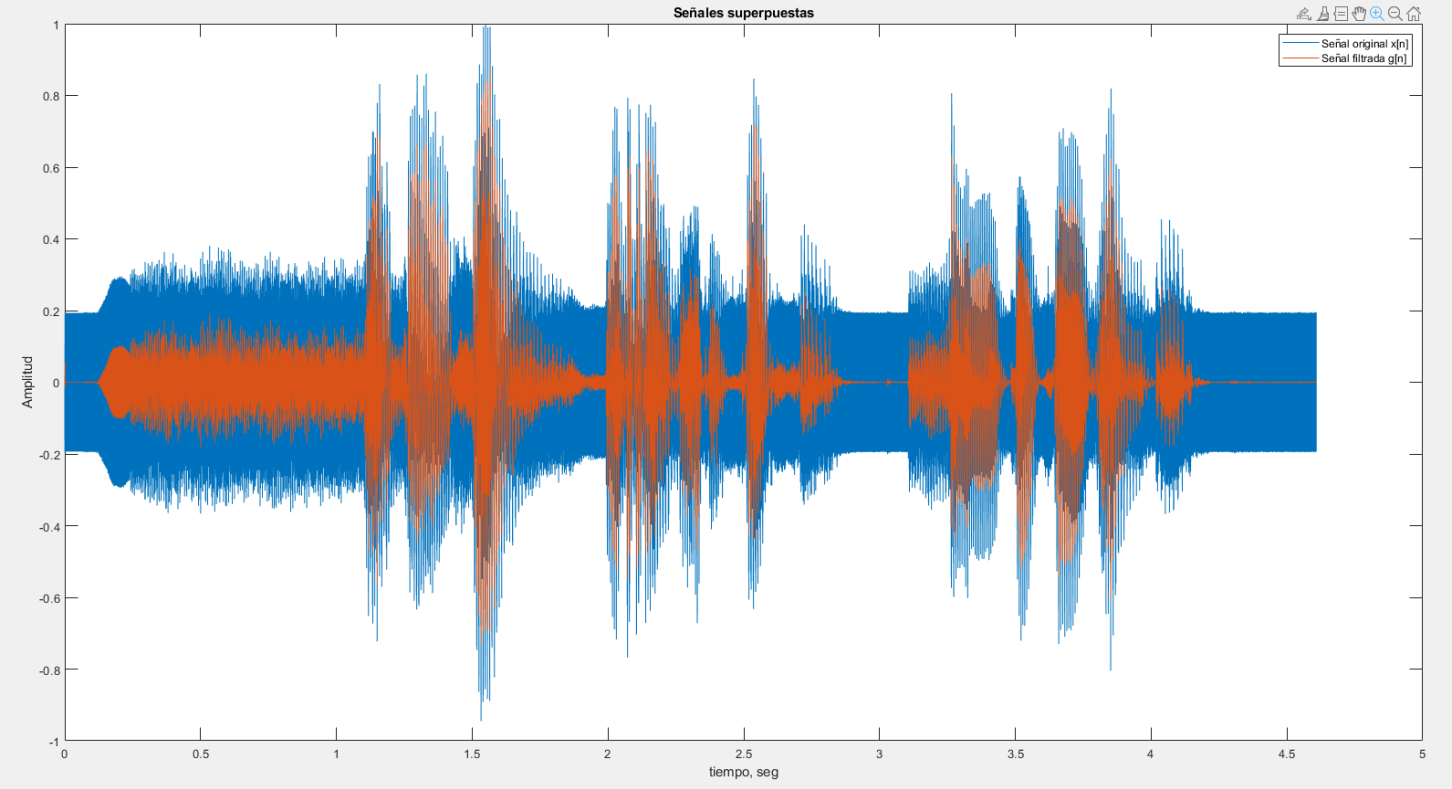
Para ver que el filtrado ha sido correcto, vamos a representar la señal filtrada en el dominio de la frecuencia y en el dominio del tiempo.

Al representar ambas señales en el dominio del tiempo superpuestas podemos ver que, al haber filtrado los agudos del pitido, la señal filtrada g[n] (señal x[n] filtrada) tiene una amplitud mucho menor que la señal original x[n].

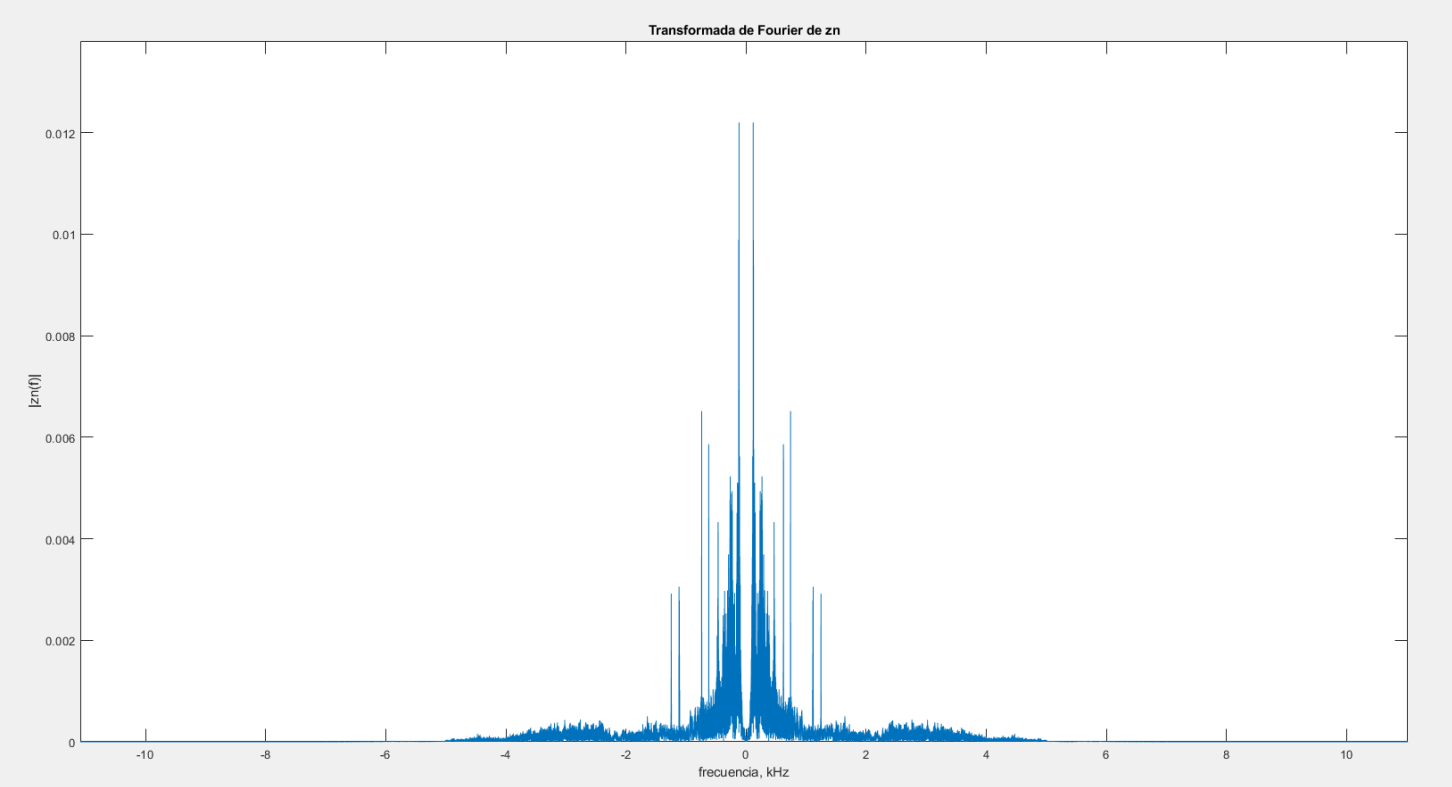
Esta diferencia en la amplitud se puede ver de forma muy clara en las secciones de la señal en las cuales el hombre que habla en la grabación está en silencio y solo se escucha el pitido.

El pitido de fondo es ruido que tiene nuestra señal y que queremos eliminar. Nuestra señal original está por lo tanto compuesta por ruido + la grabación del hombre que está hablando. Al estar el hombre en silencio solo escuchamos el pitido y es ahí cuando la amplitud de la señal filtrada es aproximadamente cero pero la amplitud de la señal original no ya que sigue teniendo el ruido.

En la gráfica inferior hemos representado la señal original y la señal filtrada en una misma gráfica para ver la diferencia en amplitud, mencionada anteriormente, y el efecto del filtrado con mayor claridad.



Para corroborar la explicación anterior y por lo tanto ver que el filtrado es correcto representamos la señal filtrada en el dominio de la frecuencia. Como se puede apreciar en la gráfica las deltas correspondientes al pitido han sido eliminadas.



La última comprobación para ver que el filtrado ha sido exitoso es reproducir la nueva señal filtrada **g[n]** y ver si el pitido se ha eliminado. Al reproducir la señal con **sound(gn, fs)** se escucha con claridad que el pitido ha desaparecido.

Implementación de un filtro FIR utilizando DFT

Apartados a), b) y c)

El algoritmo de solape-almacenamiento lo hemos programado en una función aparte, que se llama **AlgoritmoSolape().m**

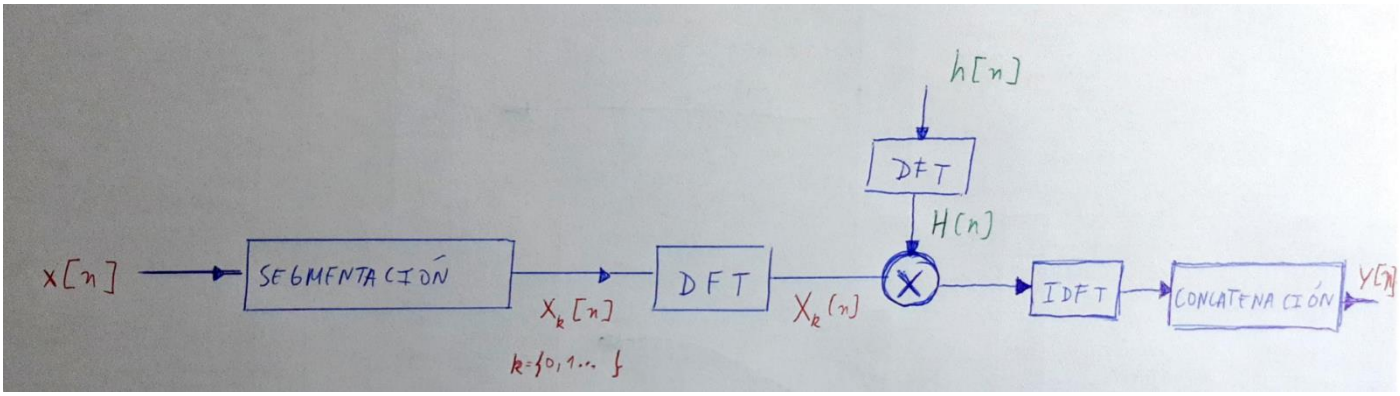
Como parámetros de entrada tiene la señal a filtrar, en este caso nuestra señal original x[n], los coeficientes del filtro que hemos diseñado en el apartado anterior y por último la longitud de los bloques en los cuales dividimos nuestra señal original, L = 500. Como salida tiene la señal filtrada y[n].

Primero procedemos a explicar el funcionamiento del algoritmo overlap-save y después comentaremos como hemos implementado dicho algoritmo en Matlab.

Inicialmente tenemos nuestra señal **x[n]** y nuestro filtro que llamaremos **h[n]**.

1. Lo primero que hacemos es dividir nuestra señal a filtrar en bloques o trozos de longitud L, que en nuestro caso viene especificado en el enunciado como **L = 500 muestras**. A cada bloque lo vamos a llamar **xk[n]**, siendo **k = {0, 1, …}**.
2. Para cada bloque xk[n] y para nuestro filtro h[n] calculamos las **DFTs** de L muestras. Una vez tenemos los dos vectores de longitud L los multiplicamos muestra a muestra, obteniendo como resultado un vector de **longitud L** que llamaremos **zk[n]**. Dicho vector contiene la convolución circular.
3. Calculamos ahora de cada vector zk[n] la **IDFT**.
    (a) Como las primeras **p-1** muestras de cada vector zk[n] no coinciden con la convolución lineal, las eliminamos.
    (b) Por lo tanto nos quedamos solo con las muestras que van desde **n = {p, …, L-1}**, donde **p es el número de coeficientes de nuestro filtro**, que si coinciden con la convolución lineal.
4. Para coger el siguiente bloque **(k+1)** hay que tener en cuenta esto último. Sabiendo que vamos a desechar las primeras p-1 muestras porque dichas muestras de la convolución circular van a ser erróneas, hacemos que las primeras p-1 muestras de nuestro siguiente bloque (k+1), solapen con las p-1 últimas muestras del bloque anterior (bloque k).
    a. Como en el bloque inicial no tenemos bloque anterior lo que se hace es añadir **p-1 ceros** al inicio.
5. La señal de salida y[n] por lo tanto se obtiene concatenando las muestras de cada bloque que coinciden con la convolución lineal.

Adjuntamos un diagrama de bloques para ayudar a entender la explicación del algoritmo:



Ahora vamos a ver la implementación del algoritmo en Matlab.

Primero creamos algunas variables como p, longitud_bloque … etc que vamos a utilizar continuamente a lo largo de la función. Las más importantes de estas variables son:

- **p** = longitud del filtro, nº de coeficientes del filtro
- **longitud_bloque** = nº de muestras de cada bloque donde la convolución circular es igual a la convolución lineal
- **longitud_solape** = nº de muestras de cada bloque donde la convolución circular no es igual a la convolución lineal

```
1   function [yn] = AlgoritmoSolape(xn, Num, L)
2       % numero de coefientes del filtro = 101
3       p = length(Num);
4
5       % nº de muestras del bloque o tamaño del bloque real
6       longitud_bloque = L - (p-1); % 400
7
8       % tamaño del trozo de solapamiento o nº de ceros en bloque inicial
9       longitud_solape = p - 1; % 100
10
11      % longtiud de nuestra señal de entrada
12      longitud_xn = length(xn); % 442354
13
14      % creamos la señal final con ceros --> tiene misma longitud que señal
15      % de entrada xn
16      yn = zeros(longitud_xn, 1);
17
18      % nos hace falta calcular la DFT del filtro para luego poder hacer la
19      % concvolución circular
20      filtro = [Num zeros(1, L - p)];
21      filtro_dft = fft(filtro, length(filtro));
```

También calculamos la DFT del filtro que hemos diseñado anteriormente en la práctica para luego poder hacer la convolución circular.

Como para cada bloque vamos a hacer el mismo procedimiento, salvo en el primero, hemos optado por hacer un **bucle for** que itere la señal x[n] en saltos de **longitud_bloque**. De esa forma nos aseguramos dividir bien la señal y que vamos a iterar por todas las muestras.

Como en el primer bloque tenemos que añadir p-1 ceros, hacemos una sentencia if - else para separar el primer bloque del resto. La única diferencia entre el if y el else es a la hora de crear el bloque x_subj. En el else cogemos las p-1 muestras del bloque anterior y en el if metemos p-1 ceros. El resto del procedimiento es prácticamente el mismo.

La variable que en el código hemos llamada **x_subj se corresponde al xk[n]** de la explicación teórica del algoritmo. La variable **convolución_lineal** se corresponde a su vez **zk[n]** en la explicación teórica.

```
23        % dividimos nuestra señal en bloques y para cada bloque hacemos 1) DFT,
24        % 2) multiplicar por DFT del filtro y 3) IDFT
25 -      for j = 0:longitud_bloque:(longitud_xn-longitud_bloque)
26 -          if j == 0   % primer bloque
27                % creamos el primer el bloque que consta de parte de solape con
28                % ceros y la parte de la señal correspondiente
29 -              x_subj = [zeros(longitud_solape, 1); xn(1:longitud_bloque, 1)];
30
31                % hacemos DFT del bloque mediante la fft()
32 -              x_subj_dft = fft(x_subj, length(x_subj));
33
34                % convolución circular --> muestra a muestra
35 -              convolucion_circular = x_subj_dft.*filtro_dft.';
36
37                % calculamos la IDFT
38 -              y_subj = ifft(convolucion_circular, length(convolucion_circular));
39
40                % nos quedamos con el trozo sin la zona de solape y metemos en
41                % nuestra señal final
42 -              yn(1:longitud_bloque, 1) = y_subj(p:end, 1);
43
44 -          else % no primer bloque
45                % los bloques que no son el primero cogen p-1 muestras del
46                % anterior y L-(p-1) muestras de la señal
47 -              x_subj =  xn((j-p+2):(j+longitud_bloque), 1);
48
49                % hacemos DFT del bloque mediante la fft()
50 -              x_subj_dft = fft(x_subj, length(x_subj));
51
52                % convolución circular --> muestra a muestra
53 -              convolucion_circular = x_subj_dft.*filtro_dft.';
54
55                % calculamos la IDFT
56 -              y_subj = ifft(convolucion_circular, length(convolucion_circular));
57
58                % nos quedamos con el trozo sin la zona de solape y lo metemos
59                % en nuestra señal final
60 -              yn(j+1:j+longitud_bloque, 1) = y_subj(p:end, 1);
61 -          end
62 -      end
63 -  end
```

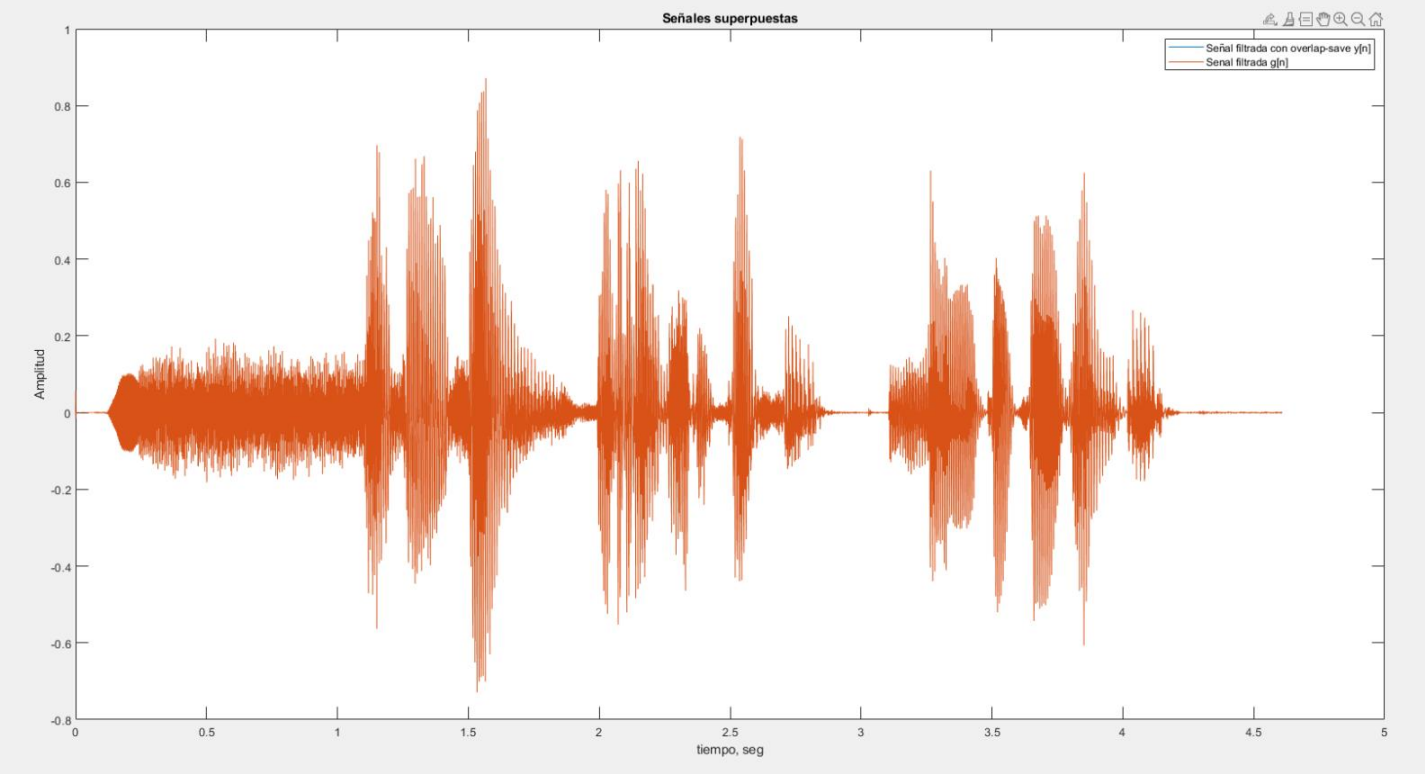
Análisis de los resultados

Apartado a)

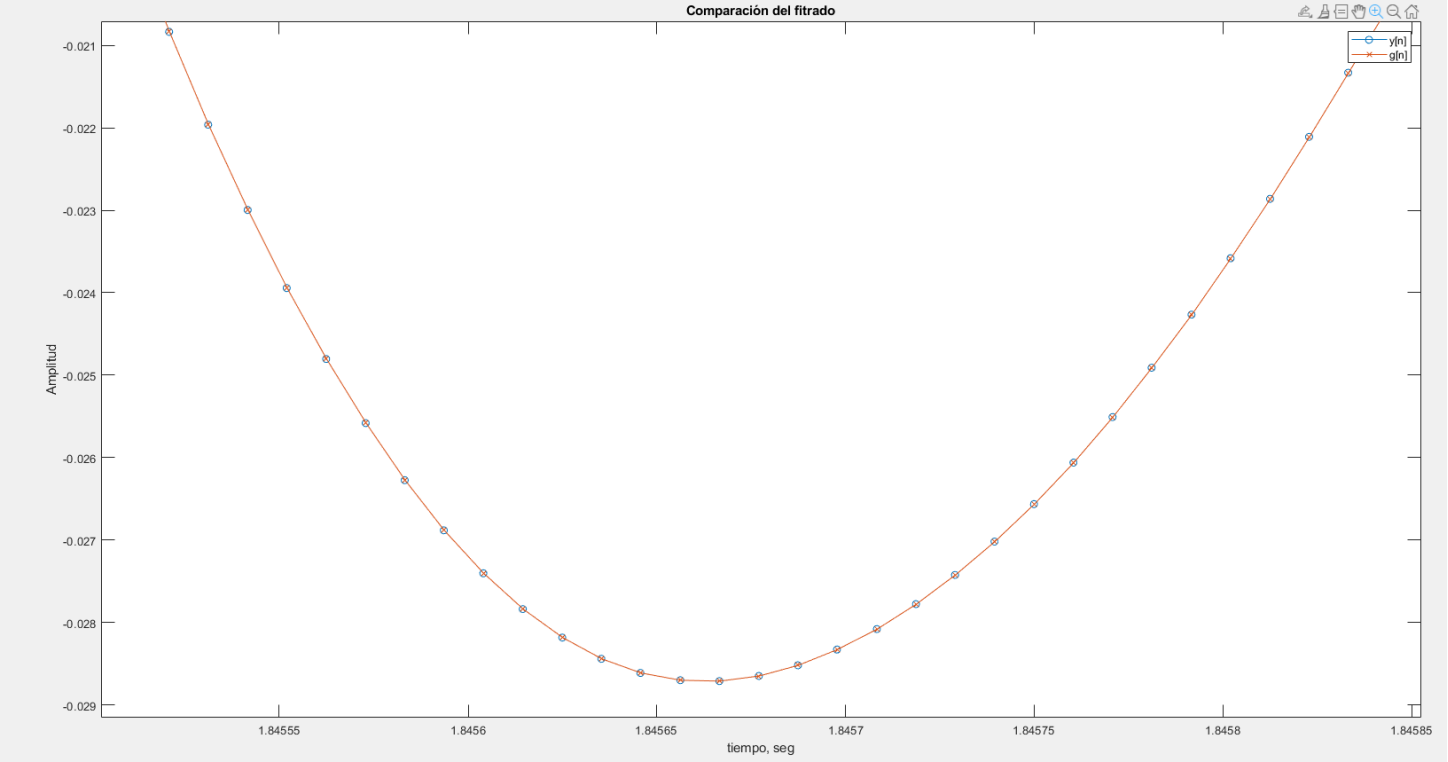
En el apartado d) del primer bloque (Diseño de un filtro FIR) hemos representado ambas señales superpuestas en el dominio del tiempo. Las explicaciones y el análisis la gráfica también se encuentra en ese apartado.

Apartado b)

Señales y[n] y g[n] representadas en el dominio temporal y superpuestas:



Como se puede ver las señales son prácticamente las mismas por lo tanto están perfectamente solapadas. Esto se confirma con la gráfica inferior, en la cual se puede ver que ambas señales son idénticas, por lo tanto el filtrado o los efectos de los dos distintos tipos de filtrado sobre la señal x[n] son prácticamente los mismos.



Apartado c)

Para calcular el error cuadrático medio utilizamos la función **Función_ECM()** que programamos y usamos en la práctica anterior. Como parámetros de entrada introducimos las señales **y[n]** y **g[n]**. Por lo tanto el ECM lo calculamos con todas las muestras. Esto lo podemos hacer ya que sabemos con certeza que tanto y[n] como g[n] tienen la misma longitud, que es la de x[n].

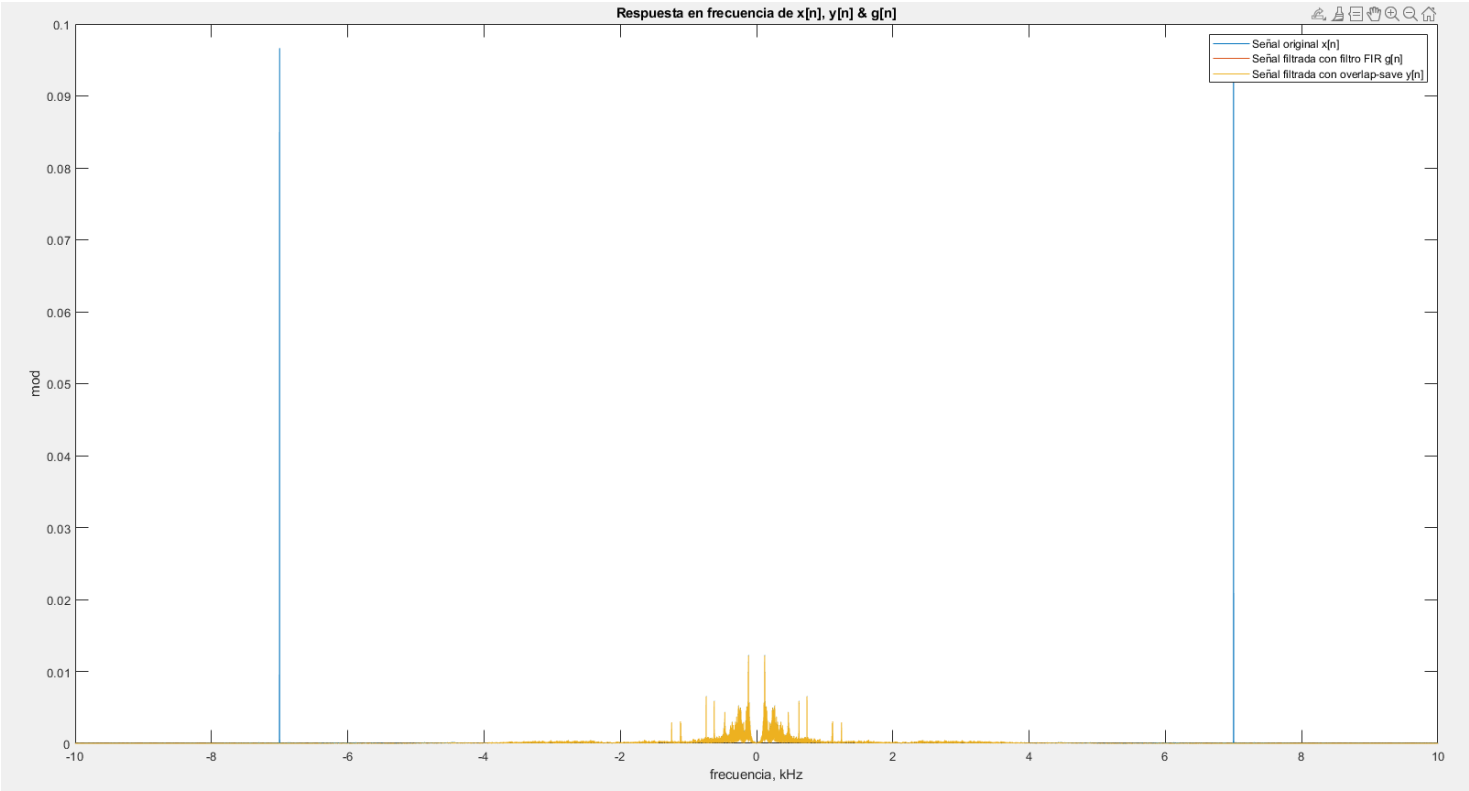
```
>> ecm = Funcion_ECM(yn, gn);
>> ecm

ecm =

    9.8573e-10
```

Como se puede ver el error cuadrático medio es despreciable siendo del orden de 10^-10.

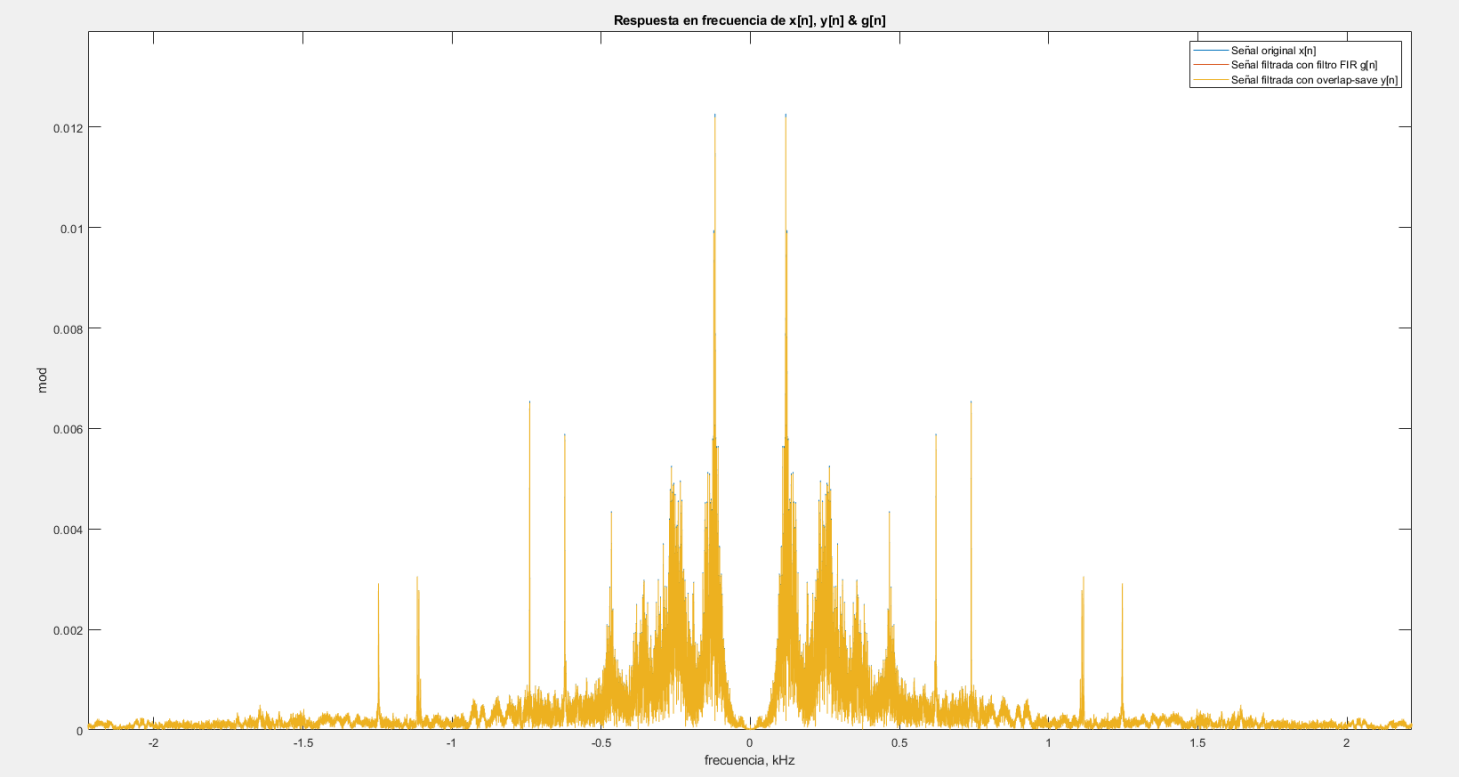
Apartado d)



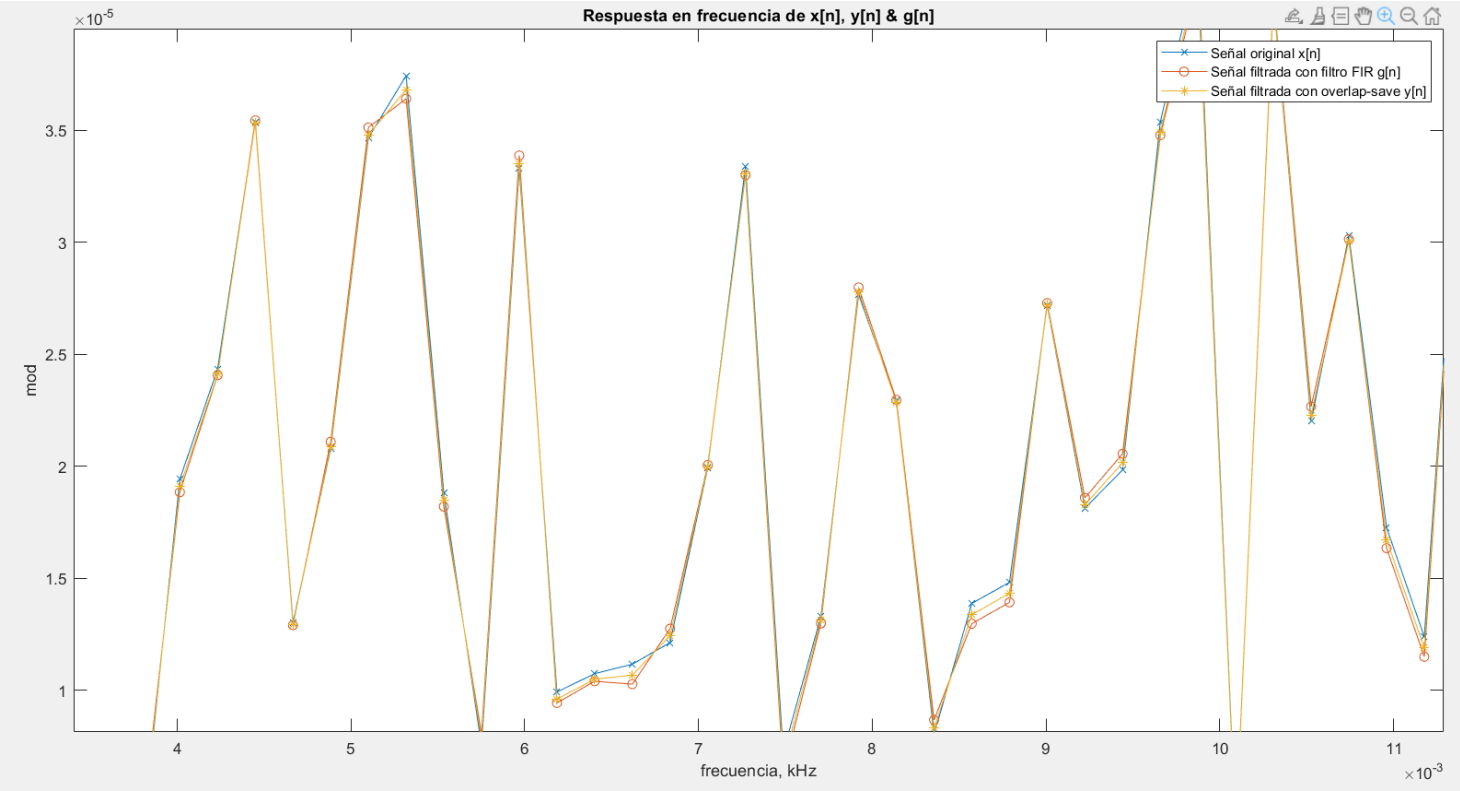
Como era de esperar las señales y[n] y g[n] tienen el mismo espectro. En la siguiente gráfica se amplía la respuesta en frecuencia para ver las señales filtradas sin las deltas correspondientes al ruido.

Se puede apreciar que la respuesta en frecuencia de las señales filtradas, son prácticamente idénticas y salvo por las deltas del ruido en la señal original, son iguales a x[n].

Espectro ampliado para no ver las bandas del ruido:



Espectro ampliado para ver las diferencias entre las señales filtradas:



Apartado e)

En este apartado vamos a resumir las conclusiones y resultados que hemo obtenido en los apartados anteriores.

Los resultados más importantes y relevantes son:

1) ECM de un valor despreciable 10^-10
2) Solape de las señales filtradas en el dominio del tiempo
3) Solape de las señales filtradas en el dominio de la frecuencia

Lo más importante de dichos resultados es que concuerdan y son coherentes entre sí. Con esto nos referimos a que, por ejemplo el ECM que obtenemos nos indica que ambas señales filtradas van a ser prácticamente iguales, ya que el valor del EMC es del orden de 10^-10 así que cercano a cero.

Esto lo podemos a su vez corroborar visualmente con los resultados que obtenemos al representar ambas señales tanto en el dominio del tiempo como en la frecuencia. En las gráficas de apartados anteriores como el d) o el b) se puede ver que las señales filtradas se solapan, pareciendo ser casi idénticas. Solo al hacer zoom sobre las gráficas como en la última gráfica del apartado d) por ejemplo, podemos apreciar que no son idénticas al cien por cien. Esa mínima diferencia es la que indica el valor del ECM.

Esto significa que ambos filtrados vistos en la práctica, por bloques mediante la DFT o un filtro FIR paso bajo, tienen el mismo efecto sobre la señal x[n], ya que las diferencias entre ambos resultados son mínimas.

La última comprobación aparte de la visual también es la auditiva. El resultado esperado es que, al ser ambos filtrados iguales, ambas señales se vayan a oír igual, sin el pitido de fondo. Y en efecto ese es el caso.