8,5

LE1:G2

Práctica 6: Implementación de filtros LTI utilizando DFT

Laboratorio de PDS

Álvaro Prado Moreno (201800742) y Javier Álvarez Martínez (201707599)

20-4-2021

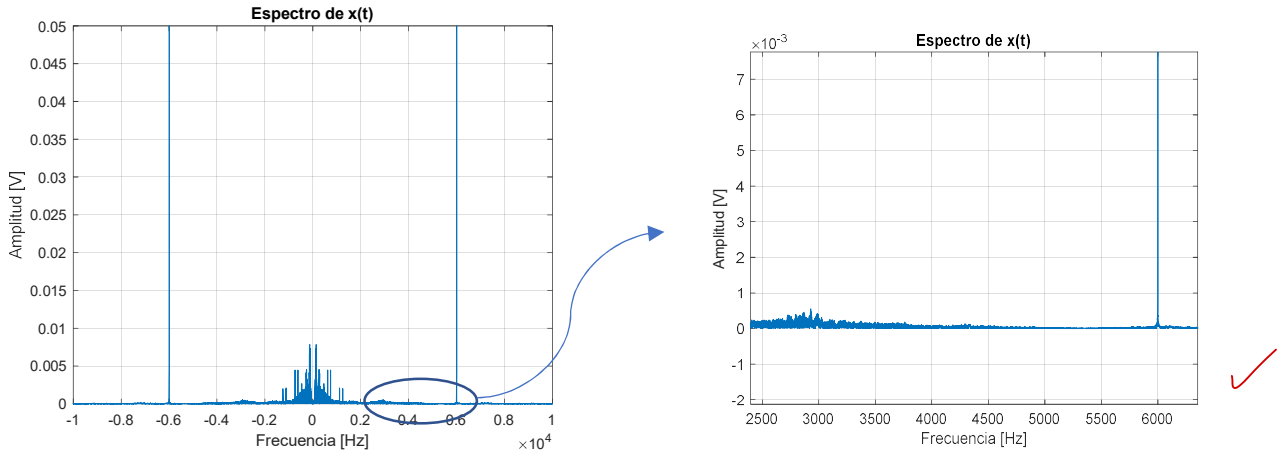
Introducción

En esta práctica vamos a realizar el filtrado de una señal de audio mediante la transformada discreta de Fourier utilizando el método de solape y almacenamiento. Además, nos serviremos de la herramienta de creación de filtros de Matlab para crear un filtro FIR y poder comparar los resultados obtenidos con el método de solape y almacenamiento.

Diseño del filtro FIR

En este apartado se va a diseñar un filtro FIR con el objetivo de eliminar un tono molesto de la señal $x[n]$ proporcionada para la práctica.

El espectro de la misma se muestra a continuación:



¿ Cuál es la máxima frecuencia de la señal deseada de audio ?

Puede observarse un tono de frecuencia $f = 6KHz$ el cual queremos eliminar. Para ello se ha diseñado un filtro en el que se ha escogido como frecuencia de corte $fc = 4KHz$. De esta manera se consigue que la atenuación sea lo suficientemente grande en $6KHz$ como para eliminar la componente espectral no deseada, sin eliminar información del espectro $X(f)$.

↳ No lo habéis caracterizado.

Con esta frecuencia de corte, una pequeña parte del espectro que sí nos interesa se verá atenuada. No obstante, si se escoge como $fc \geq 4KHz$ para evitar esto, no se atenuará lo suficiente el tono no deseado.

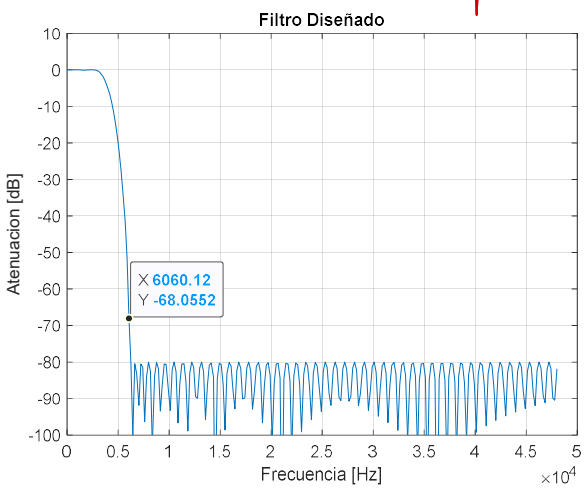
La frecuencia de muestreo elegida para el filtro tiene que ser la misma que la de la señal $x(t)$.

Nota: Además de la justificación teórica, se ha observado de manera empírica los efectos del filtro eligiendo distintas frecuencias de corte. Lo cual nos ha llevado a confirmar que con la banda de transición resultante la mejor opción es fc=4KHz para atenuar lo suficiente el tono no deseado.

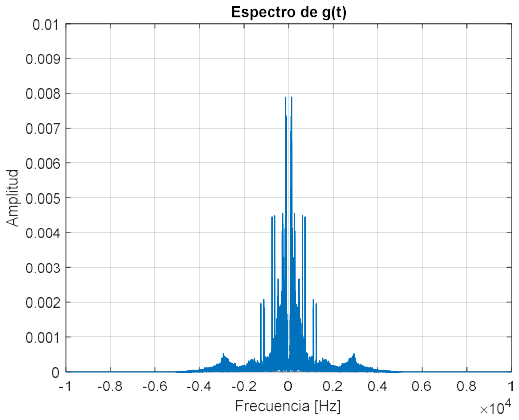
Esta comprobación empírica se ha hecho observando el espectro de la señal a la salida y reproduciéndola para distintas fc.

El filtro FIR resultante del diseño es el siguiente:

Tendrías que haber reducido un poco la frecuencia de corte.



Comprobamos si el diseño ha sido correcto sometiendo a la señal original a un filtrado y viendo si desaparece la componente espectral que queríamos eliminar. Como se muestra a continuación, vemos que el objetivo se cumple.



→ Aquí es complicado ver nada, mejor en dB

Algoritmo de solape y almacenamiento

En este apartado se va a mostrar el código que se ha realizado en Matlab para poder llevar a cabo el filtrado de la señal $x[n]$ usando la DFT. Para ello, se ha divido la señal original en bloques para poder realizar sobre los mismos el producto punto a punto con la DFT del filtro.

En el siguiente código se encuentran comentados los distintos pasos del algoritmo.

```
%La longitud del filtro FIR esta determinada por el numero de
%coeficientes=> el orden del mismo +1

longitud_filtro = M+1;

%Calculamos la fast fourier transform del filtro con 500 puntos para
%poder hacer la multiplicacion punto a punto

H_f=fft(Num_Filter, 500);

%Definimos la longitud de los trozos que se van cogiendo de la señal original
longitud_trozo= 500-longitud_filtro+1;

%Inicializamos la salida
y= zeros(1, length(x));


%El bucle tiene tantas iteraciones como trozos de 500-100= 400 muestras
%haya contenidos en la señal original

for i=1:ceil(length(x)/longitud_trozo)

    %En la primera iteracion necesitamos meter longitud_del_filtro-1 ceros
    if(i==1)
        %Creamos un trozo completo = trozo original+(longitud_filtro-1)zeros
        x_r=zeros(1, longitud_trozo+longitud_filtro-1);
        x_r(longitud_filtro: end)= x(i:longitud_trozo);
    else
        %En iteraciones posteriores: las longitud_filtro-1 primeras
        %muestras del trozo completo corresponden a las p-1 ultimas muestras del
        %trozo original anterior

        x_r=zeros(1, longitud_trozo+longitud_filtro-1);
        x_r(1: longitud_filtro-1)= x((i-1)*longitud_trozo-longitud_filtro+2: (i-
1)*longitud_trozo);

        %Hay que tener cuidado con la ultima iteracion ya que el ultimo
        %trozo no va a ser de 400 muestras asi que añadimos ceros a la
        %derecha
        if(i==ceil(length(x)/longitud_trozo))
            x_r(longitud_filtro: longitud_filtro+length(x((i-1)*longitud_trozo+1: end))-
1)= x((i-1)*longitud_trozo+1: end);
        else
            x_r(longitud_filtro: end)= x((i-1)*longitud_trozo+1: i*longitud_trozo);
```

```
        end
    end

    %Es ahora cuando se hace el producto punto a punto de las DFT del
    %filtro e y_r => equivale a convolucion circular

    fft_trozo=fft(x_r);

    producto_parcial=H_f.*fft_trozo;

    trozo_salida=ifft(producto_parcial);

    y((i-1)*(longitud_trozo)+1:  i*longitud_trozo)=trozo_salida(longitud_filtro:end);

end
```

Justificación de la elección del parámetro P

El parámetro P del algoritmo de solape y almacenamiento es la cantidad de muestras-1 que se deben de añadir al principio del trozo cogido de la señal original para poder obtener el trozo completo. P es siempre la longitud del filtro, en nuestro caso 101 muestras.

De esta manera se logra llenar la memoria del filtro. Nos da igual que esas muestras concatenadas al principio del trozo sean del trozo de señal anterior o bien ceros (en el caso del primer trozo) ya que serán descartadas tras el filtrado. Esto se debe a que la convolución circular no coincide con la convolución lineal durante las P-1 primeras muestras.

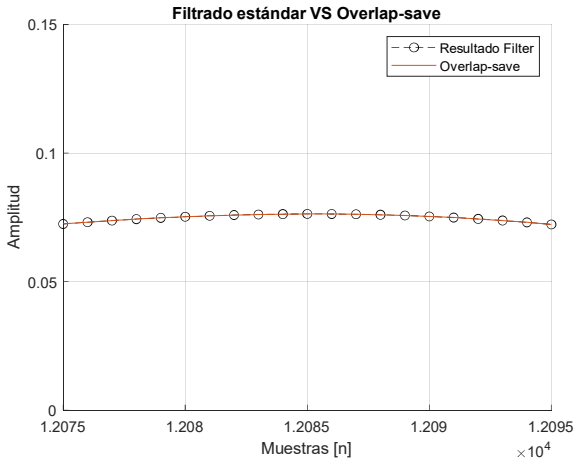
Esto no es cierto

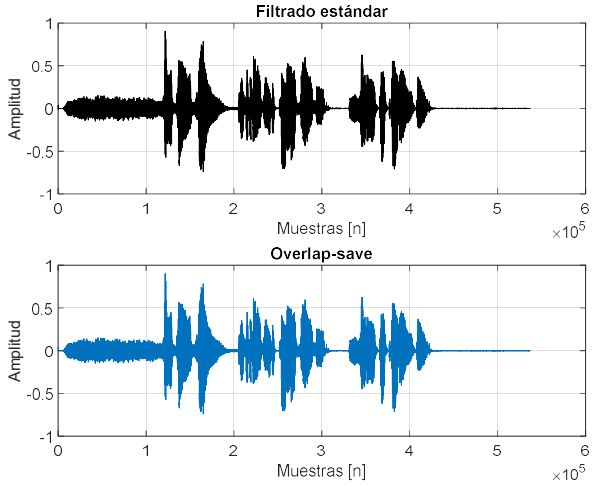
Análisis de resultados

En esta sección vamos a comparar los resultados obtenidos mediante el filtrado que implementamos con la función de Matlab *filter* y los coeficientes del filto FIR diseñado, y el filtrado creado con nuestro algoritmo de solape y almacenamiento.

Análisis temporal

A continuación, se muestra la señal resultante en el tiempo realizando el filtrado con el filtro FIR y la señal resultante filtrando con overlap-save. Se adjunta una gráfica ampliada con las señales superpuestas y otra gráfica con las señales representadas en ejes diferentes.





Como se observa la coincidencia es total por lo que nos ayuda a afirmar que el filtrado realizado con nuestro algoritmo es correcto y útil para el filtrado de señal. Como se verá a continuación con el cálculo del ECM, se obtiene un mismo resultado, pero ahorrando en complejidad computacional.

Además, existen situaciones en las que filtrar por bloques es necesario. Cuando se da la situación en la que no paramos de recibir datos en streaming hay que aplicar este método ya que ninguna memoria de un filtro aguantaría esa cantidad de datos.

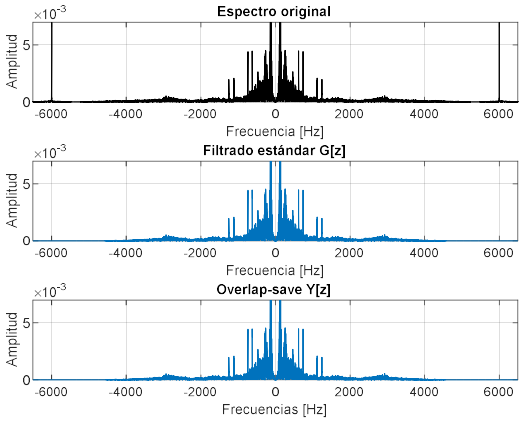
Error cuadrático medio

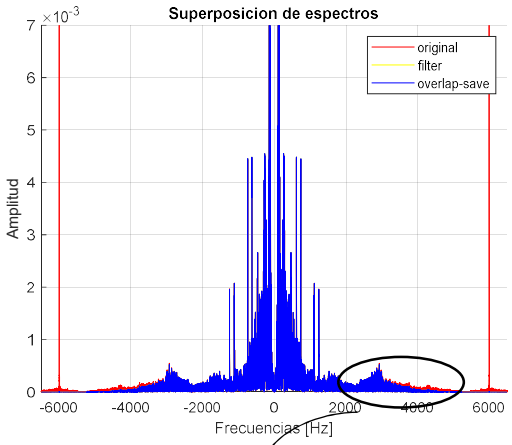
En este apartado hemos calculado el error cuadrático medio entre las dos señales filtradas. Una por overlap-save y otra usando la convolución lineal mediante *filter.*

El error cuadrático medio es de 2.43×10^{-33}, un error muy bajo ya que como se ha podido ver en la representación temporal la coincidencia es total.

Análisis del espectro en frecuencia

En este apartado vamos a comparar el espectro de las señales original, la señal filtrada mediante el filtrado con el filtro creado con el fdatool de Matlab y el filtrado realizado con nuestro algoritmo de solape y almacenamiento.





Cabe destacar que debido a lo ajustado que estaba el tono a la señal. Para poder eliminar completamente el tono se la elegido una fc que ha atenuado muy ligeramente las frecuencias superiores de la señal original.

Como se puede observar el filtrado es correcto mediante ambas herramientas. Ambos filtrados nos eliminan el tono no deseado que nos producía ese sonido que deseábamos quitar de nuestra señal original. Además, el espectro de $y[n]$ y de $g[n]$ es coincidente como se refleja en la gráfica de superposición.

Conclusión Análisis de resultados

Gracias al cálculo del ECM y a la representación en frecuencia y en tiempo podemos concluir que el resultado del filtrado por bloques mediante la DFT da el mismo resultado que el filtrado mediante la convolución lineal con un filtro FIR. Además, nos aporta un ahorro en recursos computacionales y una herramienta necesaria para el filtrado en streaming. Ya que si nos están llegando datos continuamente ninguna memoria de un filtro FIR podría aguantarlo.

Conclusión

En esta práctica hemos podido observar cómo es posible realizar un filtro completo mediante la realización de un algoritmo de solape y almacenamiento que parte de la realización de la convolución circular. Este filtrado es óptimo ya que hemos podido comparar gráfica y matemáticamente como el filtrado con un filtro FIR creado utilizando la herramienta de Matlab de creación de filtros es prácticamente una réplica de nuestro filtro creado mediante la transformada de Fourier en tiempo discreto.