Laboratorio de Procesado Digital de Señal - 3º GITT

Informe Práctica 6: implementación de filtros LTI utilizando DFT

Alumno 1:	*Jaime Arana Cardelús*
Alumno 2:	Guillermo Fernández Pérez
ID Grupo:	3ª_LE2_G6
Calificación:	
Comentarios:	

Diseño de un filtro FIR

En este bloque de apartados, el alumno analizará la señal de audio facilitada y diseñará un filtro FIR para eliminar un molesto tono que tiene la señal.

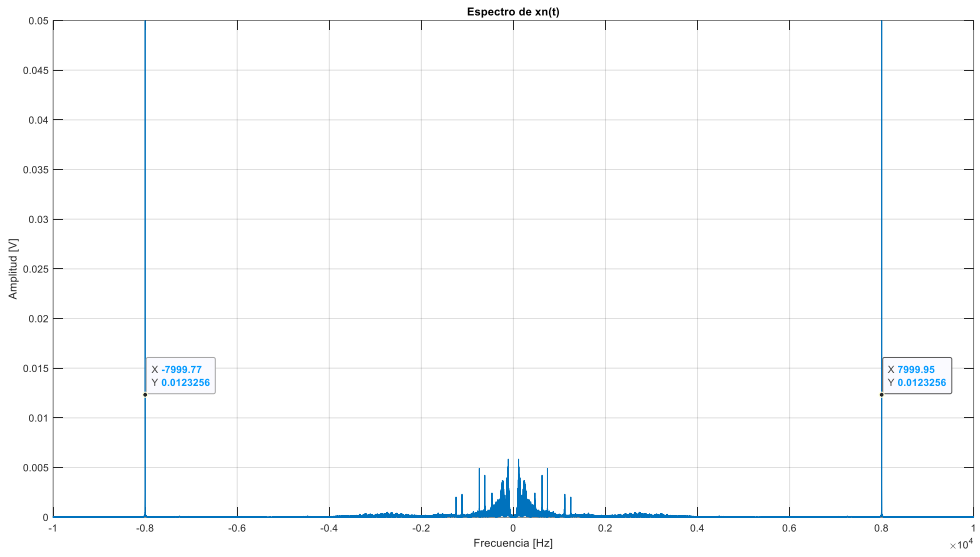
A partir de los parámetros facilitados al alumno, realice los siguientes apartados.

a) Lea el archivo de audio que le ha facilitado el profesor. La señal obtenida se denominará x[n]. Reproduzca esta señal de audio prestando mucha atención al volumen. Empiece con volumen bajo para evitar dañarse los oídos.

Como habrá podido comprobar, sobre la persona que habla hay un tono de sonido muy molesto, el cual se desea eliminar mediante un filtro FIR.

b) Indique los valores de la frecuencia de muestreo y de la frecuencia de corte que debe tener el filtro. Justifique de forma clara y contundente dichos valores.

Como se puede observar en la gráfica inferior, el tono con frecuencia 8 kHz pertenece al ruido que queremos eliminar. Por lo tanto, la fc que hay que escoger debe eliminar dicho tono, sin modificar la señal de audio. Es por eso por lo que hay que tener en cuenta el ancho de banda de la señal de audio. Dicho BW llega hasta f = 4 kHz, como se puede ver en la gráfica, y con estos valores escogemos una frecuencia de corte de fc = 5.5 kHz.



c) Usando la herramienta `fdatool`, obtenga los coeficientes `b` de un filtro FIR **paso bajo** con las siguientes características, con el objetivo de atenuar notablemente dicho tono:

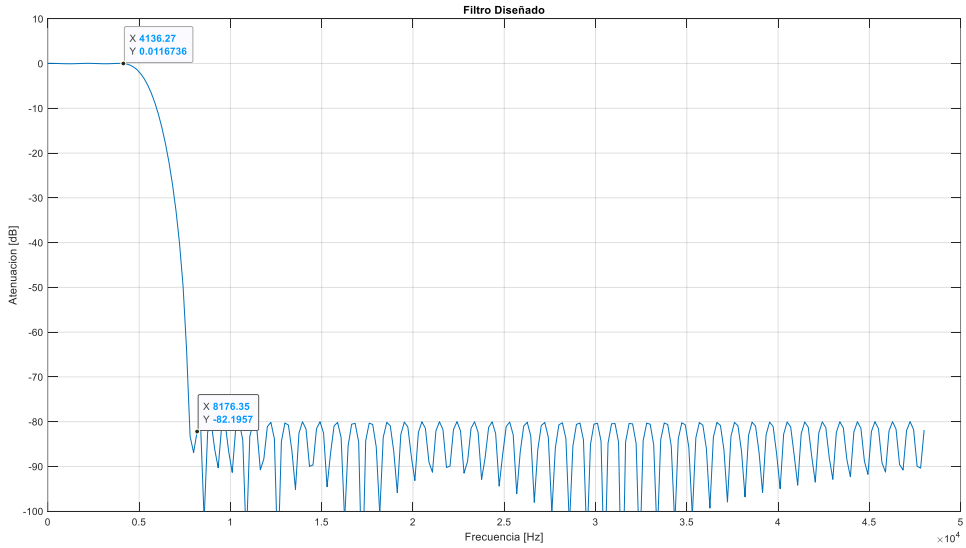
- Tipo de respuesta: Lowpass
- Método de diseño: FIR – Constrained Equiripple
- Orden del filtro: *M*
- Especificación de frecuencias:
  - Fs: *frecuencia de muestreo* (a especificar por el alumno)
  - Especificación: cutoff

- Fc: *frecuencia de corte* (a especificar por el alumno)

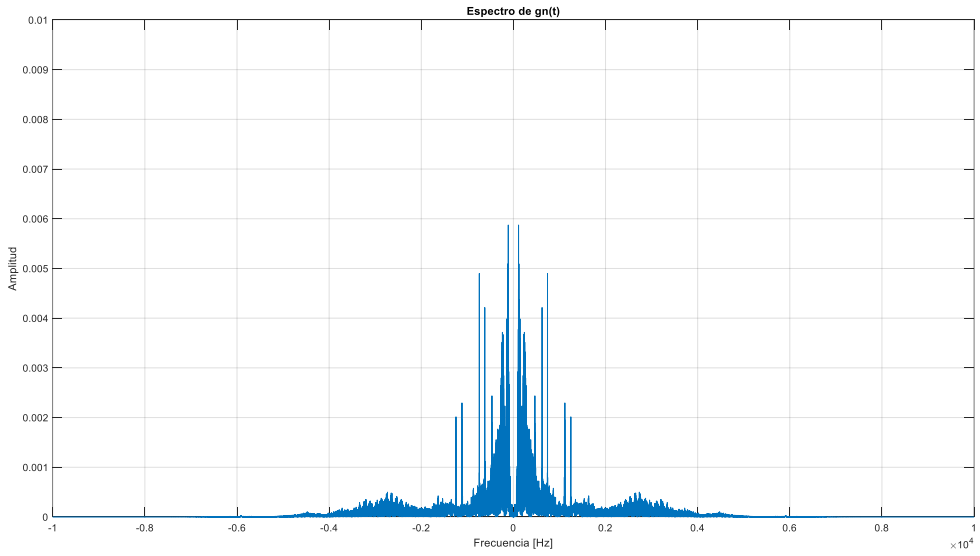
- Especificación de magnitudes:
  - Unidades: dB
  - Apass: Apass
  - Astop: Astop

d) Represente la respuesta en frecuencia del filtro diseñado, y justifique que se cumple el objetivo solicitado.

Como se puede ver en la gráfica inferior, el filtro atenúa las frecuencias a partir de 8 kHz y no modifica las frecuencias inferiores a 4 kHz. Por lo tanto, el tono del ruido queda eliminado y el audio de la señal queda intacto.



Representando el espectro de la señal filtrada se confirman los resultados anteriores y se corrobora el correcto diseño y funcionamiento del filtro.



Implementación de un filtro FIR utilizando DFT

Como bien sabe el alumno, la operación de filtrado puede implementarse mediante una convolución lineal. No obstante, la operación de convolución es computacionalmente compleja (tiene una complejidad de $O(N^2)$, siendo N la longitud de la señal a filtrar). Debido a que el filtrado es un proceso muy común en comunicaciones, sería deseable optimizar los recursos dedicados a esta operación. Tal y como se ha estudiado en clase, la Transformada Discreta de Fourier (DFT) puede ser utilizada para filtrar señales discretas con una salvedad: la DFT produce convoluciones circulares que, por regla general, no son equivalente a las convoluciones lineales. No obstante, uno de los claros beneficios del uso de la DFT es su rápida capacidad de cálculo: el filtrado con DFTs supone una complejidad de $O(N \cdot \log_2(N))$, que es mejor (es decir, menos complejo) que $O(N^2)$ para valores de N bajos. De manera que, lo único que hay que hacer para acceder a las ventajas en la rapidez de cálculo es conseguir hacer la convolución lineal equivalente a la convolución circular. Una vez conseguido esto, tendremos una manera rápida de filtrar señales discretas. Tal y como veremos en la sesión de hoy, el algoritmo de solape y almacenamiento consigue igualar el resultado de la convolución circular con el de la convolución lineal mediante una determinada gestión de las muestras de entrada y salida.

En este bloque de apartados el alumno implementará el algoritmo del método de solape-almacenamiento visto en clase para aplicar el filtro diseñado en el bloque anterior sobre la señal de audio facilitada por el profesor. Puede consultar la documentación subida a Moodle sobre dicho algoritmo.

A partir de los resultados obtenidos en el bloque anterior, realice los siguientes apartados:

a) Implemente el algoritmo de solape-almacenamiento, con una longitud de $L = 500$ muestras para cada trozo de señal, y aplíquelo a la señal original x[n]. La señal resultante final se denominará y[n]. Reproduzca esta señal.
b) Incluya el código del algoritmo en este apartado del informe de la práctica, comentando las líneas más significativas.

```
function [yn] = AlgoritmoSolape(xn, Num, L)
    % numero de coefientes del filtro = 101
    p = length(Num);

    % n° de muestras del bloque o tamaño del bloque real
    longitud_bloque = L - (p-1); % 400

    % tamaño del trozo de solapamiento o n° de ceros en bloque inicial
    longitud_solape = p - 1; % 100

    % longtiud de nuestra señal de entrada
    longitud_xn = length(xn); % 442354

    % creamos la señal final con ceros --> tiene misma longitud que señal
    % de entrada xn
    yn = zeros(longitud_xn, 1);

    % nos hace falta calcular la DFT del filtro para luego poder hacer la
    % concvolución circular
    filtro = [Num zeros(1, L - p)];
    filtro_dft = fft(filtro, length(filtro));

    % dividimos nuestra señal en bloques y para cada bloque hacemos 1) DFT,
    % 2) multiplicar por DFT del filtro y 3) IDFT
    for j = 0:longitud_bloque:(longitud_xn-longitud_bloque)
        if j == 0   % primer bloque
            % creamos el primer el bloque que consta de parte de solape con
            % ceros y la parte de la señal correspondiente
            x_subj = [zeros(longitud_solape, 1); xn(1:longitud_bloque, 1)];

            % hacemos DFT del bloque mediante la fft()
            x_subj_dft = fft(x_subj, length(x_subj));

            % convolución circular --> muestra a muestra
            convolucion_circular = x_subj_dft.*filtro_dft.';

            % calculamos la IDFT
            y_subj = ifft(convolucion_circular, length(convolucion_circular));

            % nos quedamos con el trozo sin la zona de solape y metemos en
            % nuestra señal final
            yn(1:longitud_bloque, 1) = y_subj(p:end, 1);
```

```
        else % no primer bloque
            % los bloques que no son el primero cogen p-1 muestras del
            % anterior y L-(p-1) muestras de la señal
            x_subj =  xn((j-p+2):(j+longitud_bloque), 1);

            % hacemos DFT del bloque mediante la fft()
            x_subj_dft = fft(x_subj, length(x_subj));

            % convolución circular --> muestra a muestra
            convolucion_circular = x_subj_dft.*filtro_dft.';

            % calculamos la IDFT
            y_subj = ifft(convolucion_circular, length(convolucion_circular));

            % nos quedamos con el trozo sin la zona de solape y lo metemos
            % en nuestra señal final
            yn(j+1:j+longitud_bloque, 1) = y_subj(p:end, 1);
        end
    end
end
```

c) Indique y justifique el valor del parámetro P.

El parámetro p es la longitud del filtro o el número de coeficientes del filtro. En este caso es un parámetro que viene dado.

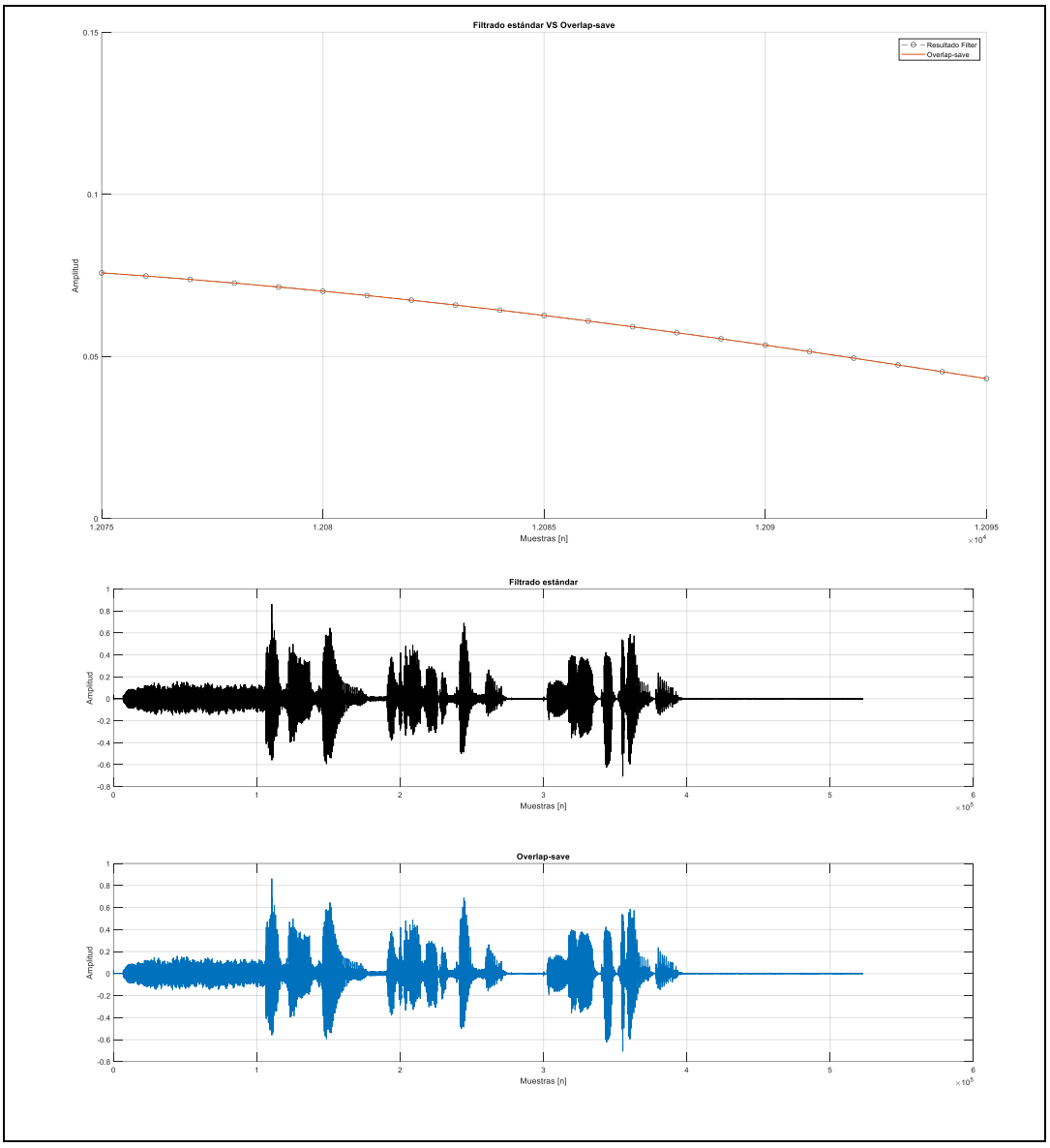
Análisis de resultados

En esta última parte de la práctica se van a analizar los resultados del proceso anterior frente a los resultados que se obtendrían implementando el filtro con la función de Matlab.

Realice los siguientes apartados, a partir de los resultados de los bloques anteriores:

a) Aplique el filtro FIR diseñado, a la señal original x[n], con la función `filter` de Matlab. La señal resultante se denominará g[n].
b) Represente en el tiempo y superpuestas en una misma figura las señales y(t) y g(t).

Como se puede ver en la gráfica inferior la coincidencia entre la señal filtrada con el filtro FIR y la señal filtrada con el algoritmo Overlap-save es total. La gráfica está ampliada para que se pueda observar con mayor claridad la similitud en los resultados de ambos filtrados.



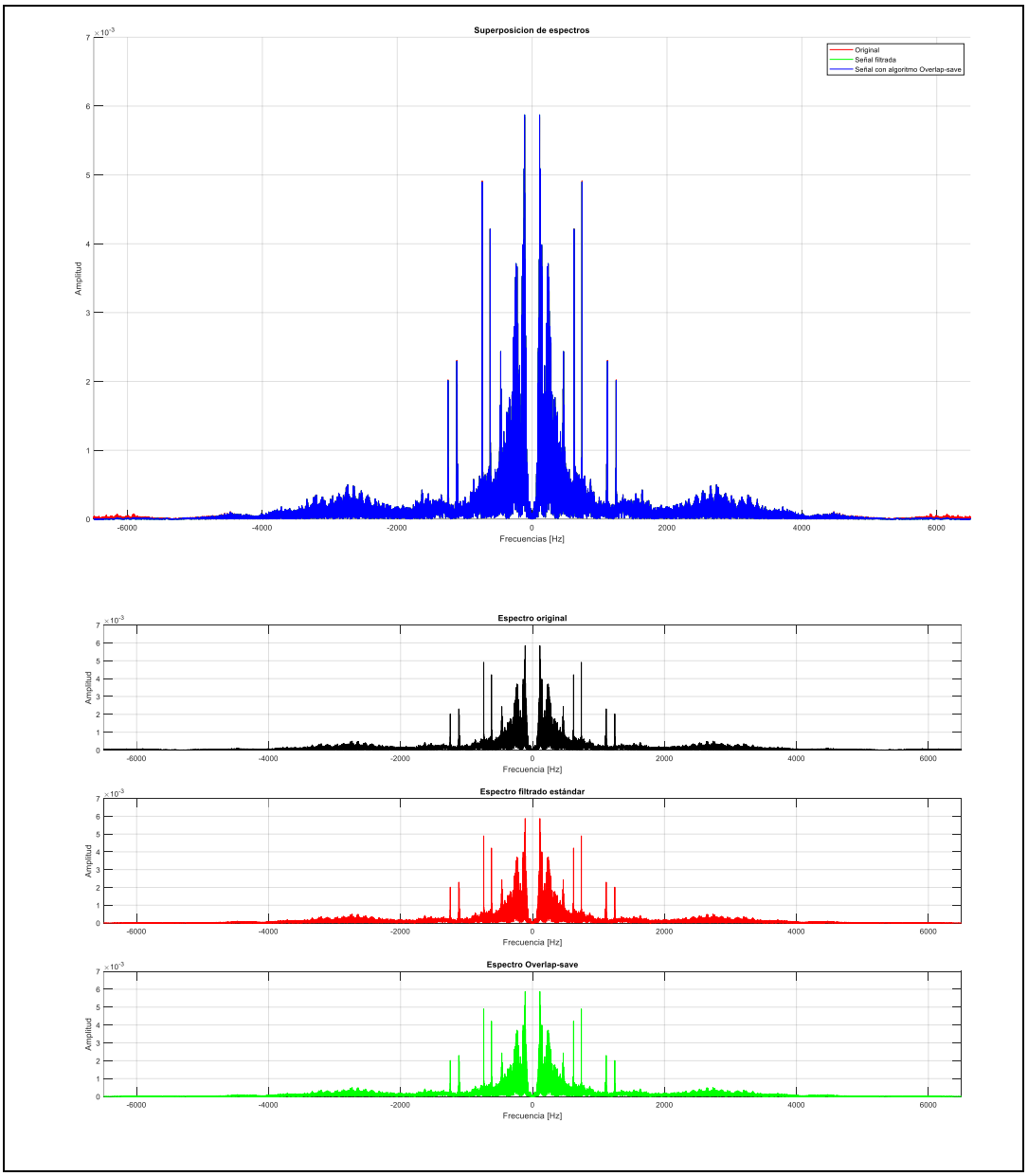
c) Calcule el error cuadrático medio entre y[n] y g[n], e indique el valor obtenido.

Como era de esperar el ECM entre ambas señales es mínimo ya que como se podía observar en el resultado del apartado superior, ambas señales prácticamente solapaban perfectamente.

```
ecm =

   5.9666e-11
```

d) Superponga el espectro en frecuencia de x[n], y[n] y g[n].



e) Exponga las conclusiones que obtiene al analizar los resultados obtenidos en los apartados anteriores. Justifique adecuadamente dichas conclusiones.

Como se ha podido comprobar, los resultados obtenidos en todos los apartados son congruentes entre ellos y son los esperados.

Con respecto a los espectros del apartado anterior, se observa cómo una vez más ambos filtrados son prácticamente iguales, solapándose los espectros completamente.

Concluyendo es importante mencionar que, aunque ambos tipos de filtrados obtienen el mismo resultado, como se ha podido comprobar en los apartados anteriores, la carga computacional en términos de recursos entre ambos métodos es diferente. El método de Overlap-save tiene una complejidad computacional menor y, por lo tanto, es conveniente usarlo. Obteniendo resultados prácticamente iguales, como se ha comprobado a lo largo

de la práctica, siempre es mejor hacer el menor uso computacional posible. Además, en casos como streaming resulta muy útil ir dividiendo la señal en trozos, ya que ninguna memoria de un filtro podría soportar tal cantidad de datos.