Laboratorio de Procesado Digital de Señal - 3º GITT

Informe Práctica 6: implementación de filtros LTI utilizando DFT

Alumno 1:	Jorge Candia
Alumno 2:	
ID Grupo:	*3B_LE2_G2*
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.

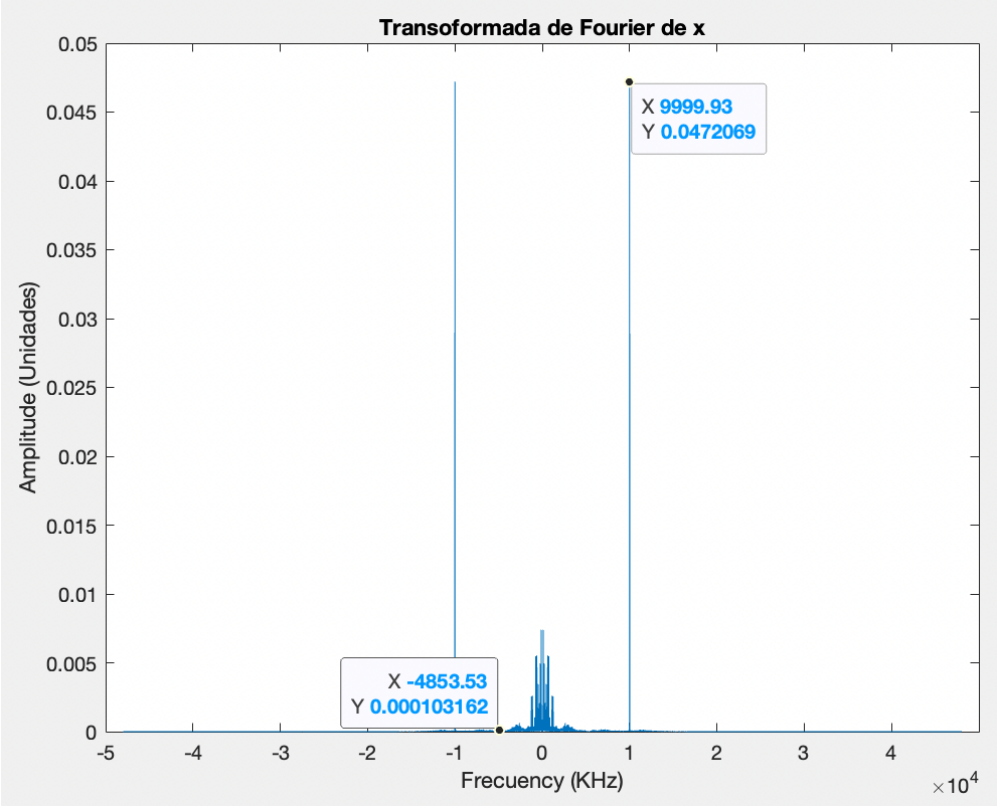
a) Lea el archivo de audio que le ha facilitado el profesor. La señal obtenida se denominará x[n]. Reproduzca esta señal de audio prestando mucha atención al volumen. Empiece con volumen bajo para evitar dañarse los oídos.

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.

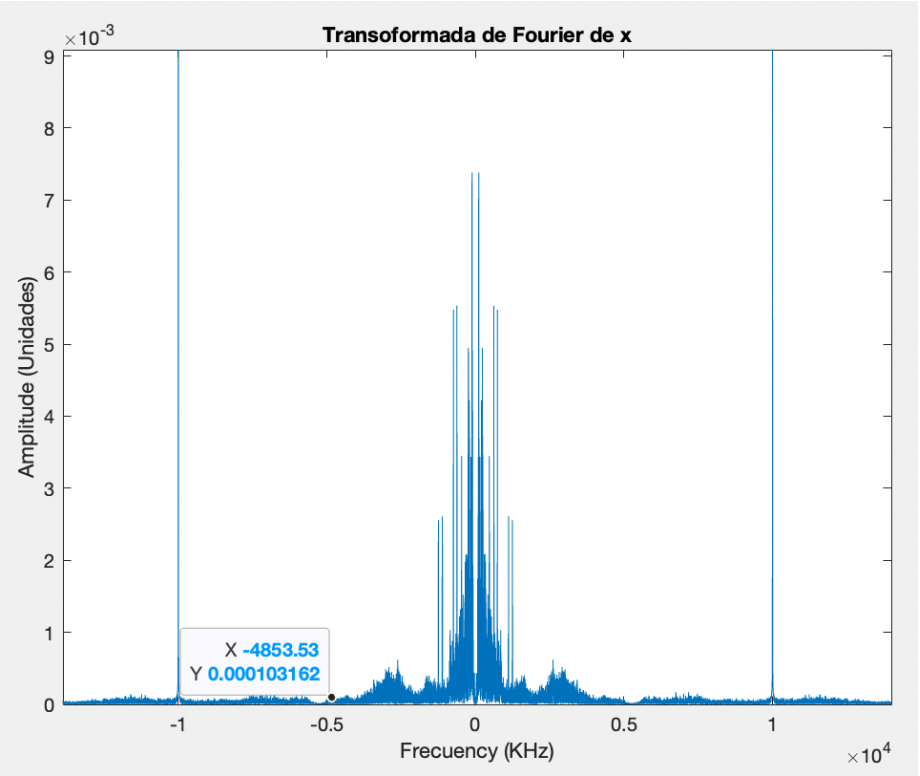
b) Indique los valores de la frecuencia de muestreo y de la frecuencia de corte que debe tener el filtro. Justifique de forma clara y contundente dichos valores.

La frecuencia de muestreo Fs del filtro será la misma que la de la señal para que las muestras coincidan en el tiempo y se pueda hacer un filtrado adecuado.

La frecuencia de corte fc se elegirá con el objetivo de dejar en la banda de atenuación el tono molesto que se escucha en el audio original.



Como se puede ver en la ilustración, el tono tiene una frecuencia de 10KHz. Pretendo mantener un audio dónde se reproduce una voz humana, la cual produce frecuencias de hasta 4KHz si lo que nos interesa es mantener una conversación de forma nítida (no música).



Se fija la fc del filtro a 6KHz para dar algo de margen, el pitido está en ~10KHz

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:

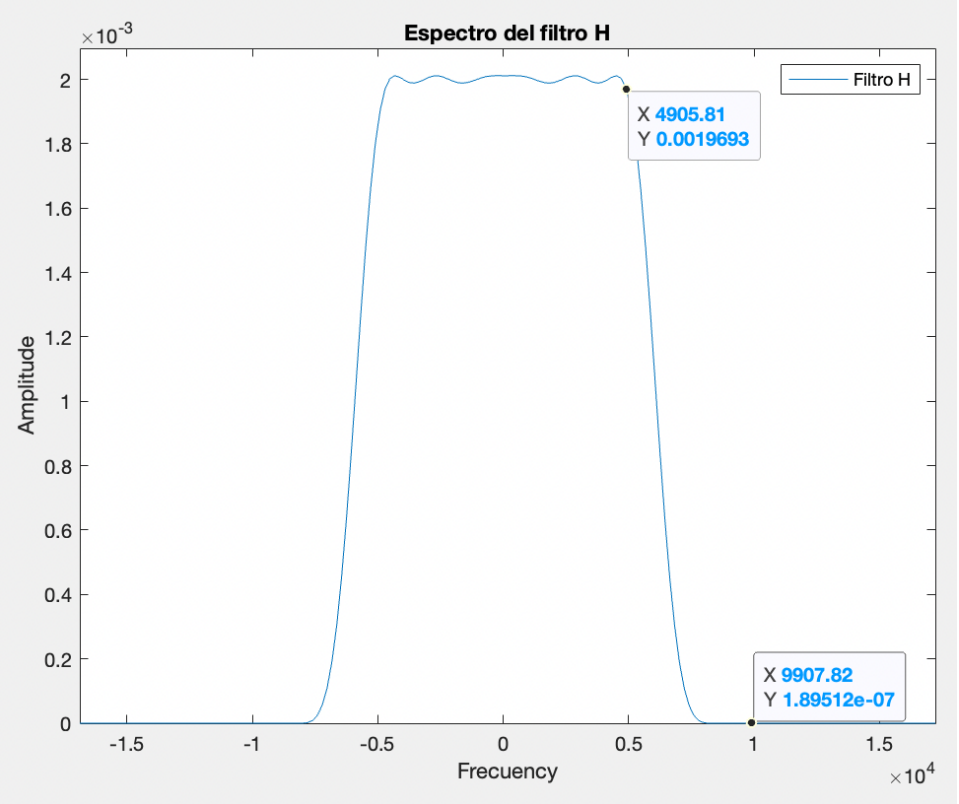
- Tipo de respuesta: Lowpass
- Método de diseño: FIR – Constrained Equiripple
- Orden del filtro: *M*
- Especificación de frecuencias:
  - Fs: *frecuencia de muestreo* (a especificar por el alumno)
  - Especificación: cutoff
  - Fc: *frecuencia de corte* (a especificar por el alumno)

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

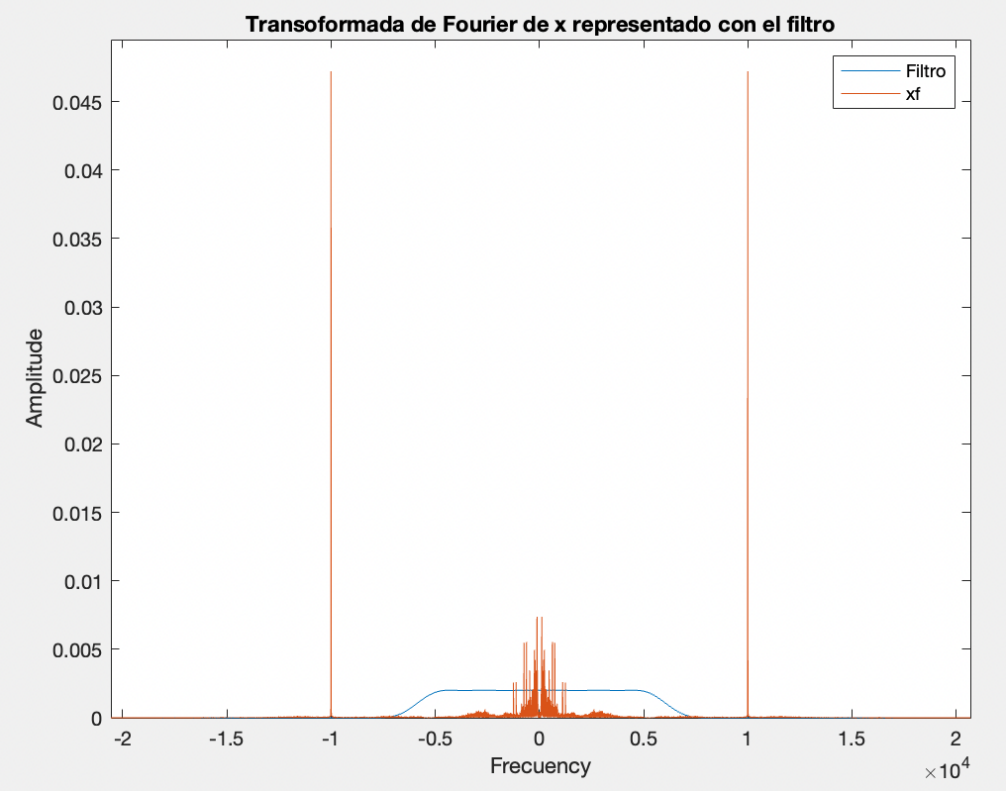
- Astop: Astop

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

La respuesta en frecuencia del filtro es la siguiente:



El filtro junto al espectro de x[n] se ve de la siguiente manera:



Como podemos ver, la voz humana cae en la banda de paso mientras que el armónico molesto se sitúa en la banda de atenuación.

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, una 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.

El objetivo del algoritmo a realizar es filtrar los paquetes que van llegando en *streaming* o tiempo real. Esto se hará con el método de Solape y Almacenamiento, que consiste en realizar la convolución circular entre el paquete y el filtro en el dominio de la frecuencia (multiplicación).

Para simular el procesamiento en streaming/tiempo real, se nos ha dado una señal, a la que llamaremos *paquete,* de una gran longitud, que iremos segmentando y aplicando el algoritmo a cada subpaquete.

Para evitar los solapes no deseados de la convolución circular, se realizará lo indicado a continuación dentro del bucle for. Cabe destacar que la resolución de todas las DFTs realizadas (con fft) ha sido de L=500, con el fin de que el subpaquete final sea de longitud L y sea trivial quitar las primeras P-1 muestras.

```
L = 500; %Longitud de segmento a filtrar
P = length(h); %P es la longitud del filtro. En las P-1 primeras muestras, la
CC ≠ CL, por lo que estas serán desechadas
N = L - P + 1; %Longitud del segmento (inicial y final) sin la parte sobrante

%Meto los P-1 ceros de primeras (filtro aún no cargado)
zx = [zeros(1,P-1) x'];

%Transformada del filtro (Matlab (fft) coge por defecto sólo 1 periodo)
H = fft(h, L);

%Creo mi salida del filtro
y = [];


for i = 1:floor(length(x)/(L-P+1))

    %Se extrae el subpaquete xr del vector, las P-1 primeras muestras sólo
sirven para 'cargar el filtro'
    xr = zx(N*i-N+1:(N*i+P-1));
```

```
    %Calculo la DFT del subpaquete
    Xr = fft(xr, L);
    %Multiplico el subpaquete con el filtro en frecuencia, operación
equivalente a la convolución en el tiempo
    Yr = Xr.*H;
    %Realizo la DFT inversa del subpaquete ya filtrado
    yr = ifft(Yr, L);

    %Anido el subpaquete ya filtrado a un vector en el que meto todos los
subpaquetes a medida que se van filtrando. Las P-1 primeras muestras se
eliminan (P porque matlab empieza a contar en el 1)
    y = [y yr(P: end)];

end
```

c) Indique y justifique el valor del parámetro P.

P es la longitud del filtro, que será el número de valores que iremos añadiendo antes de filtrar y retiraremos a la salida del filtro, como se ha explicado en el anterior apartado, con el fin de evitar los solapes indeseados de la convolución circular.

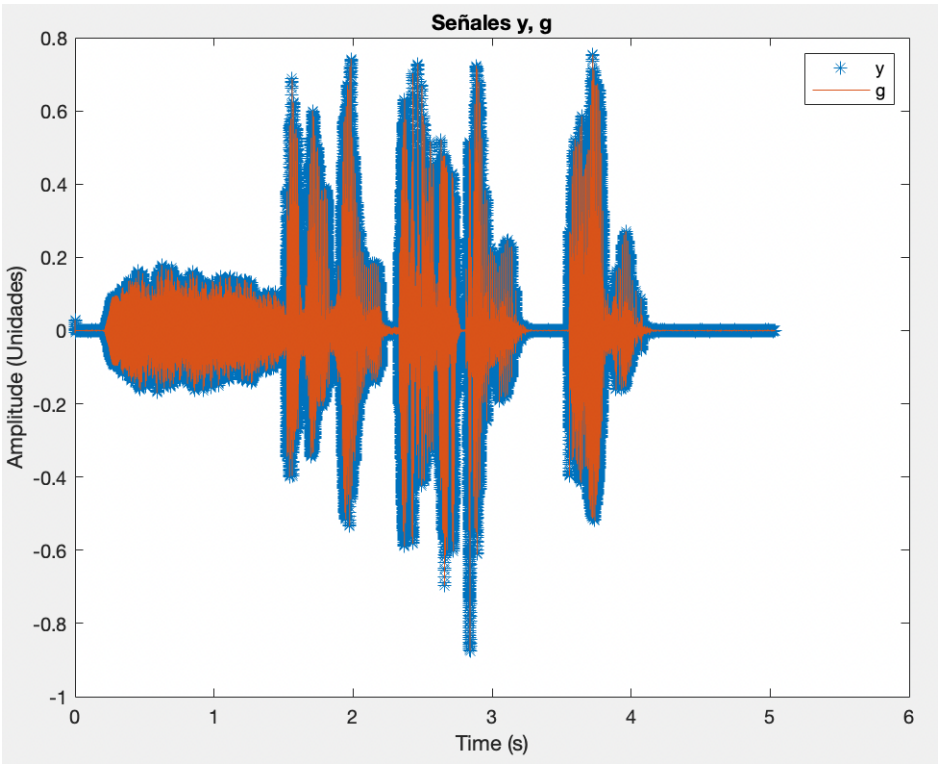
Análisis de resultados

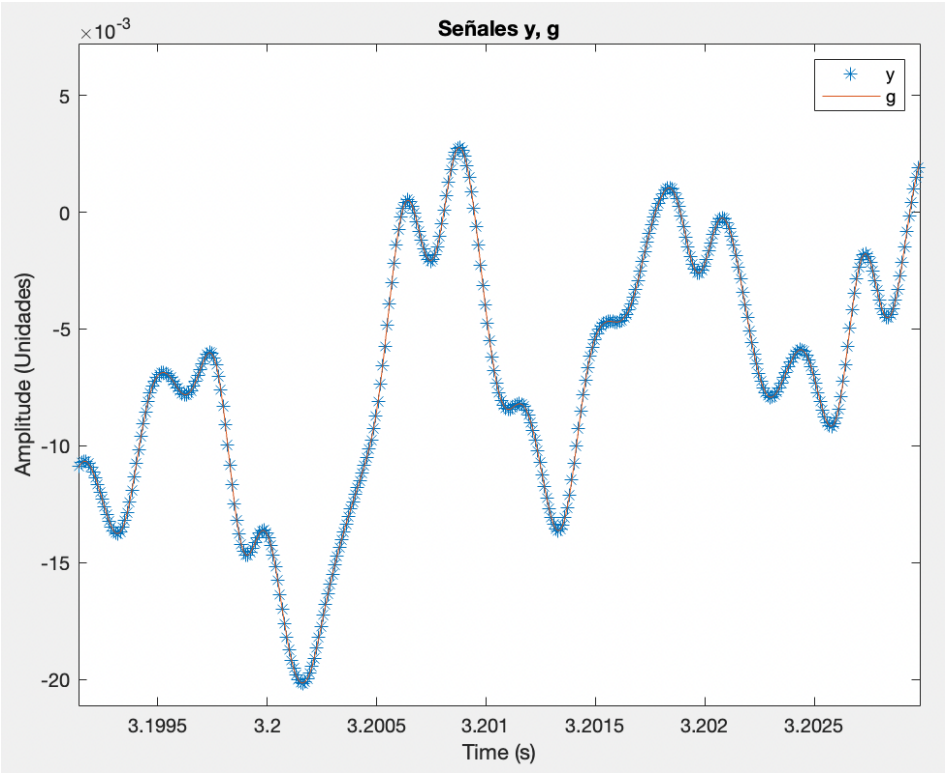
En esta última parte de la práctica se van a analizar los resultados del proceso anterior frente a los resultados que se obtendrían implementando el filtro con la función de Matlab.

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 las siguientes imágenes, ambas señales son idénticas, por lo que el algoritmo de filtrado es muy bueno.


Señales y, g
y
g
Amplitude (Unidades)
Time (s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6


Señales y, g
×10^-3
y
g
Amplitude (Unidades)
Time (s)
5
0
-5
-10
-15
-20
3.1995
3.2
3.2005
3.201
3.2015
3.202
3.2025

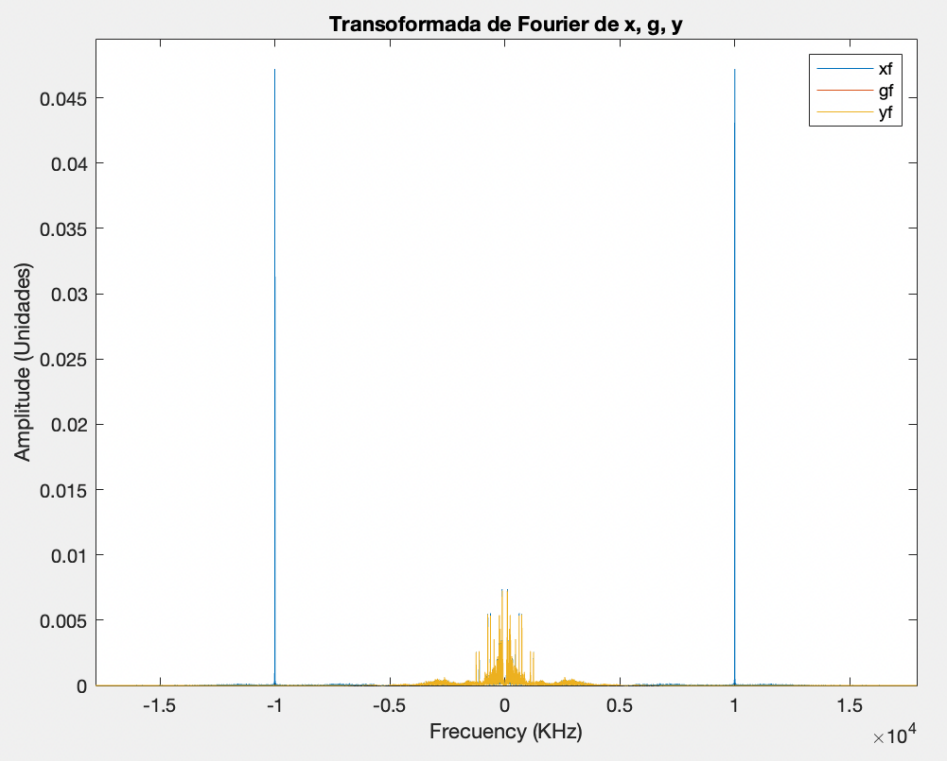
c) Calcule el error cuadrático medio entre y[n] y g[n], e indique el valor obtenido.

Como se puede ver a continuación y como era de esperar, el ECM es despreciable, prácticamente cero.

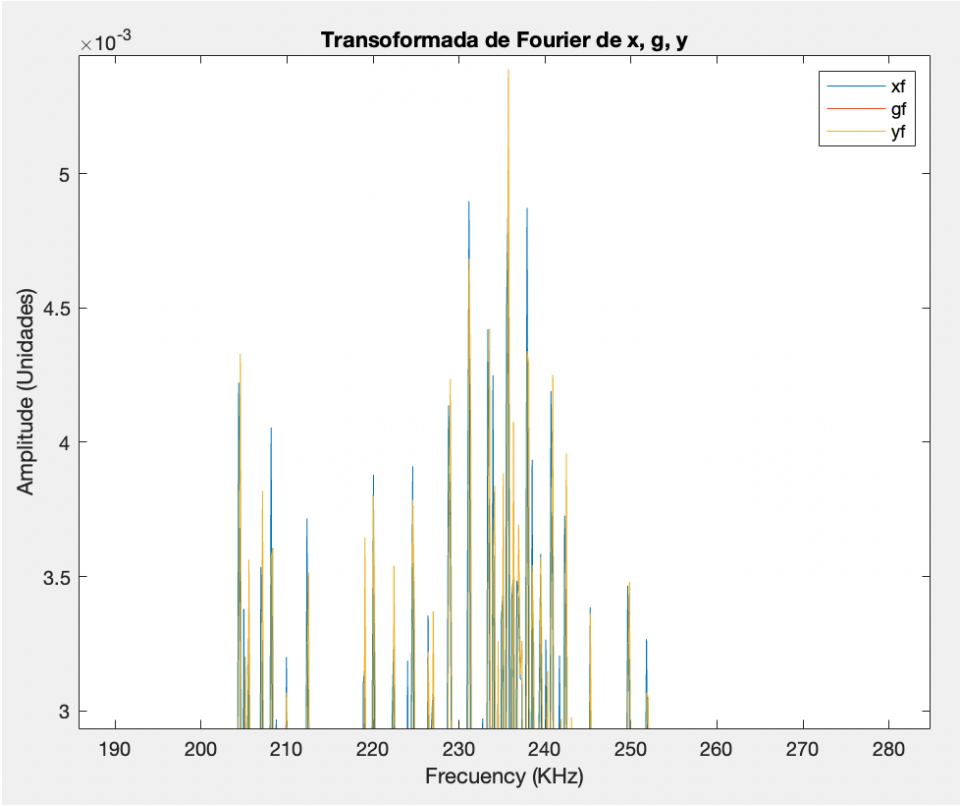
```
>> ECM = (1/length(g))*sum((g - y).^2)

ECM =

   2.1896e-33
```

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

Como se puede ver en las siguientes ilustraciones, los espectros de y[n] y g[n] no tienen el armónico molesto de x[n]. Sólo aparece en azul, correspondiente a la señal original.



Además, estos dos espectros coinciden con mucha exactitud:



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

Como se acaba de demostrar, el método de Solape y Almacenamiento es un algoritmo muy confiable y seguro para realizar procesado de señales en streaming, y es equivalente a un filtrado de la señal entera.

Por lo tanto, lo que habrá que evaluar a partir de estos resultados es la complejidad del algoritmo frente a otros que puedan ser más eficientes, con el fin de obtener el mejor resultado usando los menores recursos hardware posibles.