TP5 - Procesamiento de Señales

RESPUESTA AL IMPULSO Y CONVOLUCIÓN

Daremos el nombre se sistema a cualquier transformación que un determinado artefacto o elemento provoque sobre una determinada señal. Consideremos que x(t) es la señal original y que y(t) es la señal transformada por el sistema. Tomemos como ejemplo de sistema en esta primera etapa a un filtro pasa bajos de primer orden para simplificar la explicación.

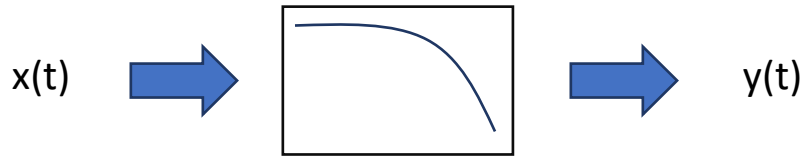


Figura 1 - La señal x(t) ingresa al sistema. Luego del proceso se obtiene y(t)

Nuestro sistema transformará la señal original x(t) modificando la amplitud de sus componentes atenuando aquellos que se encuentren en frecuencias altas. El siguiente gráfico muestra el contorno de amplitudes (en decibeles) que corresponde a ese proceso de filtrado de la señal. En el gráfico se marcó un lugar con un círculo rojo que corresponde a la frecuencia de corte de ese filtro (240 Hz). En realidad, el gráfico superior es el que muestra amplitudes y el inferior muestra cambios de fase que provocará dicho filtro. Es importante aclarar que la modificación de fases no es algo particularmente deseado, sino que resulta un efecto secundario de lo que se pretende, que es modificar las amplitudes para atenuar las altas frecuencias.

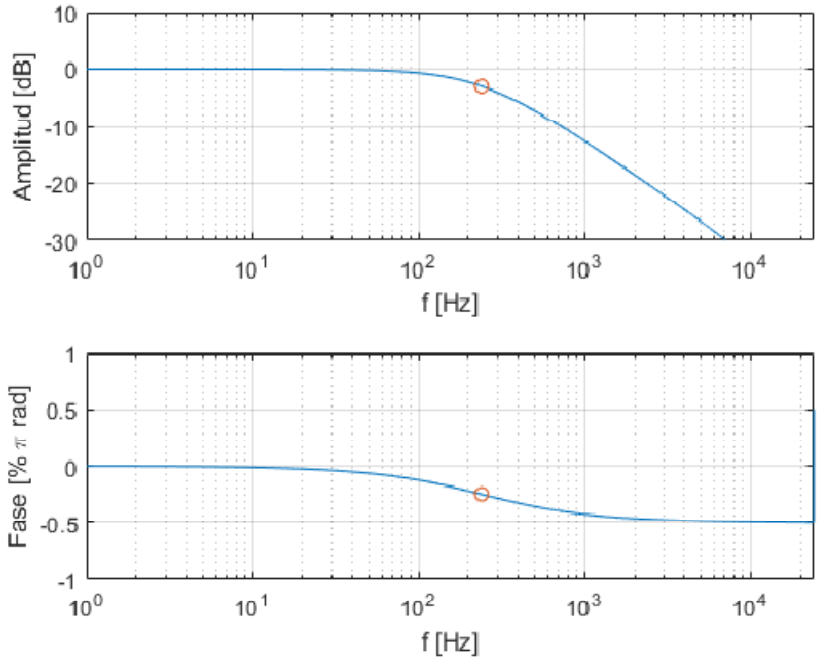


Figura 2 - Diagrama de amplitudes y fases de un filtro pasabajos de primer orden

El gráfico que representa el filtro es una especie de "Transformada de Fourier" ya que en principio muestra amplitudes para cada frecuencia. Pero, ¿representa a algún tipo de señal? Dicho con otras palabras. Cada vez que pienso en un espectro de Fourier puedo imaginar una señal que se obtiene de superponer todos esos componentes con esas amplitudes y fases. ¿Hay entonces alguna señal que produzca un espectro de Fourier como el de este filtro?
En ese camino transitaremos.

Para el análisis nos concentraremos en las amplitudes. ¿Qué es lo que nos dice ese contorno gráfico que representa el filtro? Observamos que en frecuencias bajas el gráfico se mantiene casi como una línea horizontal en 0 dB. Tengamos en cuenta que los decibeles del gráfico indican el valor en dB que provocará el filtro a las componentes de la señal x(t). Esto significa que aquellas componentes de la señal x(t) que se encuentren en bajas frecuencias no se verán alteradas. Si hay una componente de x(t) justo en la frecuencia de corte, dicha componente será reducida en un valor de 3 dB, y a partir de allí cuanto más alta sea la frecuencia de alguna componente mayor será su reducción.

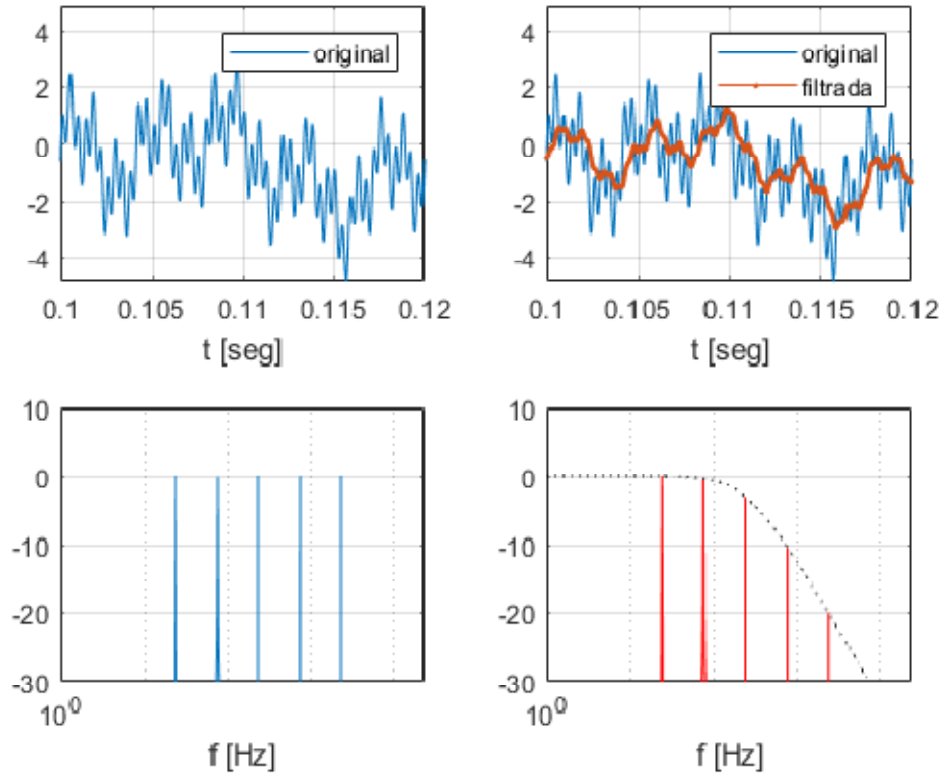


Figura 3 - La figura superior izquierda representa una determinada señal x(t). Debajo de ella se muestra su espectro de Fourier. En la figura inferior derecha puede verse el espectro modificado luego de haber sido procesado por el filtro (se dibuja además el contorno del filtro como referencia). Arriba de este espectro se muestra la señal temporal y(t) en color rojo que se obtiene a la salida del filtro (en la que se volvió a dibujar la señal original para facilitar la comparación)

Por el momento sigámonos apoyando en la idea intuitiva de que ese contorno gráfico de amplitudes representa lo que el filtro hace con la señal. Lo que debería quedar bien claro es el proceso que deberíamos realizar nosotros en MATLAB para imitar matemáticamente lo que se supone que haría dicho filtro. En principio necesitaríamos disponer de un vector con toda la información contenida en el gráfico de respuesta en frecuencia del filtro (el gráfico de contorno de la Figura 2). Habría unos en la banda de paso del filtro y valores menores que uno para las frecuencias más altas.
Cuando la amplitud en decibeles que muestra el filtro es de 0 dB, esto significa que una componente que se encuentre en esa zona no debe cambiar su amplitud. Si expresamos esto en escala lineal es equivalente a multiplicar por uno. Cuando la amplitud en decibeles del filtro es de - 3dB, eso quiere decir que debo quitarle 3 dB a la componente de la señal original que se encuentre en ese lugar. Un modo de hacer esto es multiplicar dicha amplitud por 0,707. Recordemos aquí que justamente 20.log(1) = 0 dB, y que 20.log(0,707) = -3 dB.
¿Por qué hacerlo en escala lineal y no en logarítmica? En principio podríamos hacerlo de cualquiera de las dos maneras, pero si disponemos de x(t) en MATLAB y obtenemos su espectro mediante la función fft() los resultados que se obtienen son en escala lineal. Nosotros podemos graficar la frecuencia en escala logarítmica, pero si ploteamos los puntos individuales notaremos que los primeros puntos están muy separados y luego se van amontonando. Esa escala de frecuencias está dibujada en forma logarítmica, pero las unidades son de frecuencia y no el log de la frecuencia.
En la escala vertical sucede que los datos que nos da la fft() hay que aplicarles 20.log10() para graficarlos en escala logarítmica.

En el siguiente gráfico se muestra el contorno de amplitudes del filtro en escala vertical logarítmica en decibeles junto al mismo contorno en escala vertical lineal.

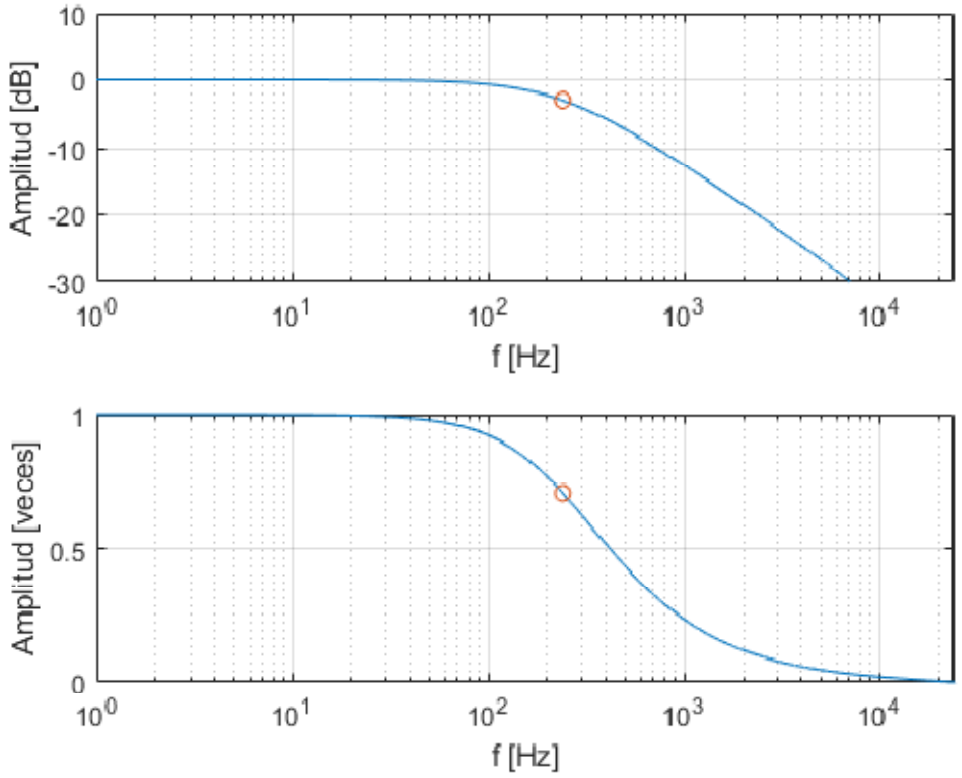


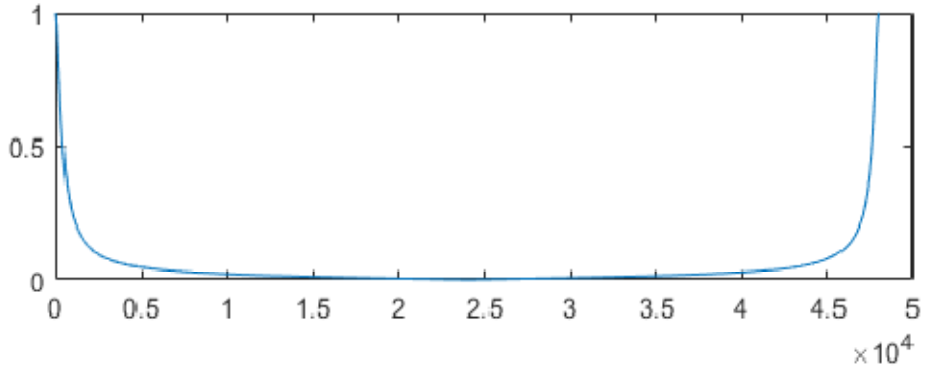
Figura 4 - Gráfico de amplitudes de respuesta del filtro en escala logarítmica (arriba) y lineal (abajo)

1. Representación de un filtro en escala lineal y logarítmica

En el campus se tiene un archivo denominado H.mat que contiene el perfil del filtro mostrado en la figura 4 en escala lineal. La variable H tiene 48000 elementos que van desde frecuencia f=0 hasta frecuencia f=47999 (que corresponde a fs-1). Se trata además de un vector cuyos componentes son números complejos (de los cuales podremos obtener módulo y fase). Toda representación espectral contiene componentes hasta la frecuencia de muestreo (en realidad 1 valor menos) pero las representaciones se muestran hasta la frecuencia de Nyquist (fs/2).

a. Cargar el archivo H.mat y generar una variable de frecuencia que vaya desde 0 hasta fs-1 con saltos de 1 Hz.

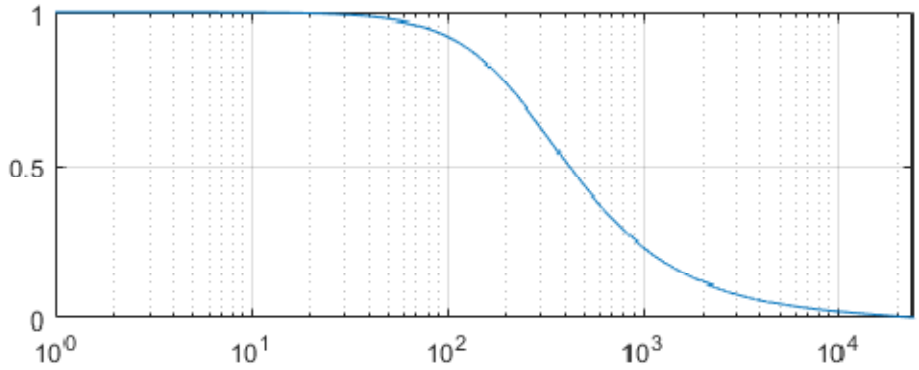
b. Representar en un gráfico el módulo de H en función de f. Recordar que para representar el módulo de un valor complejo se utiliza abs(). Deberían obtener un gráfico como el siguiente



Este gráfico representa el filtro que mencionamos, pero en escala lineal y con frecuencias hasta fs.

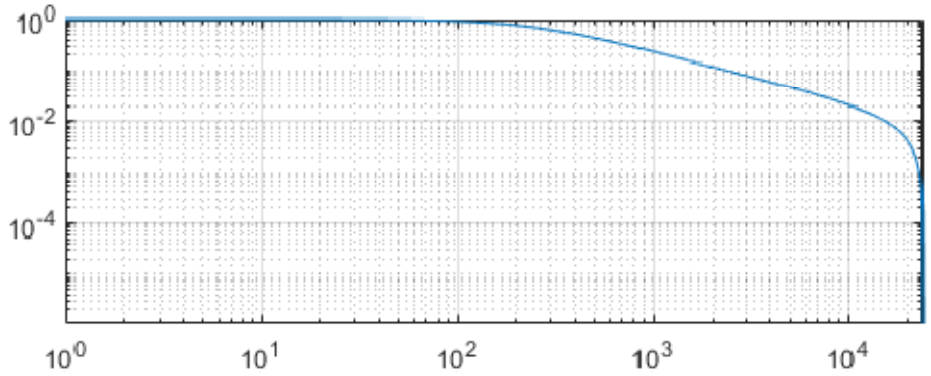
c. Explorar el uso de semilogx() que es equivalente al plot() pero que utiliza una escala logarítmica en el eje horizontal y volver a graficar con el eje de frecuencias en escala logarítmica. Agregar líneas de grilla con 'grid on'. Seleccionar los ejes de modo que la escala de frecuencias llegue hasta la frecuencia de Nyquist y la escala de amplitudes varíe entre 0 y 1.

Debería obtenerse un gráfico como el siguiente que coincide con la parte inferior de la Figura 4.

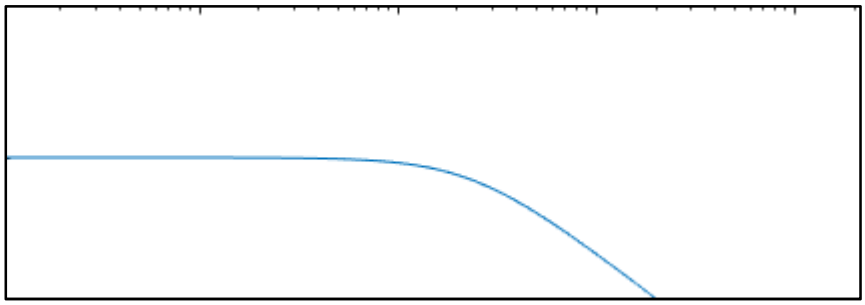


NOTA: MATLAB dispone también de la posibilidad de hacer que el eje vertical sea logarítmico con la función semilogy(), o que ambos sean logarítmicos utilizando la función loglog(). Sin embargo hay que tener en cuenta que loglog() no está en dB, ya que la escala vertical de loglog()

d. Graficar nuevamente la variable H utilizando loglog() manteniendo los límites mencionados de escalas vertical y horizontal.
Debería obtenerse el siguiente gráfico, que si bien se aproxima más al contorno esperado de un filtro de primer orden, aún tiene algunas diferencias.



La escala vertical está representada en forma logarítmica, pero los valores del eje siguen siendo lineales (sólo que ubicados de manera logarítmica). Teniendo esto en cuenta podríamos preguntarnos a cuántos dB corresponde el valor más alto de la escala vertical mostrada. ¿Y a cuántos dB corresponde el valor más bajo de la escala vertical mostrada?
e. Repetir modificando la escala para que el eje vertical muestre el rango de -40 dB hasta 0 dB. Notar que se pide el valor equivalente a estos valores, ya que aún se sigue utilizando el comando loglog(), por lo que axis() responderá a los valores lineales de la escala. Esto quiere decir que -40 dB será 0.01 y 0 dB será 1 para la instrucción axis().
f. Para poner a prueba la comprensión de cómo funciona el eje vertical logarítmico y los límites que se colocan en la función axis(), volver a graficar H pero de forma tal que la línea horizontal de la banda de paso (0 dB) quede justo en mitad del gráfico.



Para que la escala vertical esté en decibeles debe calcularse el valor en dB y utilizarse escala vertical lineal en la función de graficar. Si se quiere graficar un valor en dB SPL sería necesario obtener 20.log(Presión/20 µPa), pero lo que se

pretende graficar en un filtro es su valor de atenuación (que es lo mismo que su amplificación, sólo que con valores menores que uno). En ese caso el valor a calcular es 20.log(A), ya que la propia A sería igual a Psalida/Pentrada, y resultarían dB relativos entre la entrada y la salida.
En MATLAB el logaritmo decimal debe especificarse claramente mediante la función log10(), de otro modo calcularía el logaritmo en base e (logaritmo natural o neperiano).
g. Calcular algunos valores en dB correspondiente a valores de amplificación fáciles de identificar (para probar el uso de la función y el cálculo). Obtener por MATLAB el valor en dB equivalente a una amplificación de valor 1 (equivalente a multiplicar la entrada por 1 para obtener la salida) y luego obtener el valor en dB equivalente a una amplificación de valor 0,1 (que resulta en una atenuación, en la práctica).
h. Obtener el gráfico equivalente a la parte superior de la figura 4 donde la escala vertical de la variable H esté en decibeles.
e. Generar la figura 4 completa utilizando subplot(), para poder trazar dos gráficos en la misma figura. Agregar además los puntos que marcan el lugar de la frecuencia de corte (f=240), tal como se muestran en la figura 4. Para ello puede utilizarse un segundo par de variables en la función de graficación e indicar que el plot lo haga con un círculo en lugar de un punto agregando el parámetro 'o' (o minúscula). Por ejemplo: si se utilizase semilog(f, 20*log10(abs(H))) para uno de los gráficos, se puede hacer semilogx(f,20*log10(abs(H)), 240,20*log10(abs(H(241))),'o')

NOTA: En este último ejemplo, el uso de 240 y luego 241 no es un error. Si bien sacamos provecho del hecho de que los saltos de muestras del eje de frecuencia son de 1 Hz, para ser precisos hay que tener en cuenta esta diferencia.
Estamos pidiendo que en el valor de frecuencia 240, se dibuje el punto cuya altura vertical coincida con el 20log10 del módulo de una muestra de H. ¿Cuál muestra? La que corresponde a la frecuencia 240, pero resulta que la muestra 1 de frecuencia contiene el valor 0Hz, por lo tanto será la muestra 241 la que contenga la frecuencia 240.
Dicho con otras palabras, si tipean en la ventana de comandos f(241) permitiendo eco en pantalla, verán que el valor es 240. De modo que la instrucción anterior podría haberse escrito ... ,f(241),20*log10(abs(H(241))),'o')
El valor 240 es la frecuencia, mientras que el 241 es el índice de los vectores que contiene la información que necesitamos (la frecuencia 240 y el valor de H para la frecuencia 240).
¿Y cómo podríamos saber cuál es el índice en caso de tener otra resolución? Teníamos 1 Hz, porque la duración total de la muestra era de 1 segundo. Justamente la resolución en frecuencia no es otra cosa que uno dividido por la duración total. Si la muestra hubiese sido de una duración diferente, es necesario primero encontrar cuál es el valor del índice, que podríamos llamar nfc (índice que indica la frecuencia de corte). Esto podría ser por razonamiento, por cálculo o a mano, pero existe un modo más eficiente de pedirle al propio Matlab que nos lo diga. Esto se logra utilizando la instrucción find(). Al final del TP hay un apartado donde se explica con más detalle el uso de la instrucción find() y se comenta cómo se aplicaría en este problema.

Esto fue un ejercicio para poner en práctica el trabajo con MATLAB e ir comprendiendo los detalles más finos del tema.
Retomemos ahora la idea general.
Si tenemos una variable en MATLAB conteniendo la información del gráfico inferior de la Figura 4 (en veces) y quisiéramos simular el efecto del filtro sobre una señal x(t) en forma matemática lo que tendríamos que hacer en primer lugar es obtener el espectro de x(t) que sería X(f). Este espectro resulta ser un vector que para cada valor de frecuencia tiene asignada una amplitud correspondiente a cada componente de esa señal. Disponemos ya de ese vector que representa los valores de respuesta en frecuencia del filtro (es la variable H que se ha utilizado).
Supongamos para simplificar que tenemos una señal x(t) de duración 1 segundo y frecuencia de muestreo fs=48000 Hz. Si calculamos la fft(x(t)) obtendremos X(f) con 48000 componentes (uno para cada frecuencia entera entre 0 Hz y 47999 Hz). Trabajaremos con la x(t) mostrada en la Figura 3 que es la superposición de cinco componentes de distinta frecuencia y amplitud 1.
Una vez que disponemos de X(f) con 48000 valores (casi todos cero, excepto los correspondientes a cinco valores de frecuencia que valen 1), y además tenemos el vector H(f) que contiene los 48000 valores que indican por cuánto multiplica el filtro a cada frecuencia solamente nos queda multiplicar cada componente de X(f) por el valor de H(f) correspondiente a la frecuencia de esa componente. Dicho de otro modo, se toma el primer elemento de X(f) que corresponde a la amplitud que tiene la señal x(t) para una componente de frecuencia 1 Hz (en este caso es cero, porque no hay componente de 1 Hz) y se lo multiplica por el valor correspondiente a 1 Hz del filtro. En este caso el valor es 1

que corresponde a 0 dB. El resultado es igual a cero (0 de amplitud x 1 de efecto del filtro). Este mismo resultado se obtendrá para todas las frecuencias que tengan amplitud cero en la señal original. En algún momento llegaremos a una frecuencia cuya componente sea distinta de cero en X(f). En el ejemplo de la Figura (3) ese valor de frecuencia es f=24 Hz. Para esa frecuencia la amplitud es 1 y el valor del filtro para esa frecuencia también es prácticamente 1, con lo cual el resultado de multiplicar ambos da 1 (que significa que esa componente que está por debajo de la frecuencia de corte pasó el filtro prácticamente sin alteración). Cuando lleguemos a una frecuencia igual a la de corte, tendremos en la señal una componente de amplitud 1 y frecuencia 240 (que se corresponde con el ejemplo de la Figura 3). Para esta frecuencia el valor que indica la transformación del filtro es de 0,707. El resultado es que la señal de salida tendrá una componente allí que resultará de multiplicar el valor 1 de la componente de X(f) por el valor 0,707 que indica cuánto atenúa el filtro para dicha frecuencia.
Como conclusión. Si disponemos de un vector H(f) que representa al filtro y queremos ver cómo afecta dicho filtro a x(t), debemos obtener su transformada X(f) y luego multiplicar punto a punto cada componente de ambos vectores para obtener la transformada de la señal de salida Y(f). Si queremos llegar a conocer y(t) es necesario aplicar la "antitransformada". En MATLAB existe una instrucción para esto, que se denomina iff(), por *Inverse Fast Fourier Transform.*
NOTA: Lo anterior requiere una aclaración. Se mencionó que el componente que podría estar justo en la frecuencia de corte se multiplica por 0.7071, pero esto requiere una aclaración. El componente de H en la frecuencia de corte tiene una amplitud de 0.7071 correspondiente a su módulo, pero es un valor complejo que también tiene una fase. Esto significa que no solamente varía la amplitud de x en esa frecuencia (multiplicando por 0.7071), sino que como es una multiplicación de complejos, le suma (o resta) fase. ¿Debe preocuparnos esto? En realidad no para implementarlo, ya que si multiplicamos H.*X, el valor de H tendrá su módulo y su fase. Recordemos que para multiplicar dos números complejos se multiplicaban los módulos y se sumaban las fases (con sus signos respectivos).

2. Filtrar una señal multiplicando su espectro por la respuesta en frecuencia del filtro H(f)
a. Generar una variable x(t) de duración 1 segundo que tenga 5 componentes de frecuencia. Todos de amplitud 1, y frecuencias que comiencen en 24 Hz, y que aumenten en forma logarítmica de modo que respeten la siguiente ley

$$x(t) = \sum_{n=0}^{4} \sin\left(2.\pi.f_n.t\right) \qquad considerando\ que\ f_n = 24\,.\,10^{n/2}$$

De este modo se obtendrá la función x(t) mostrada en la figura 3. Donde la primera componente distinta de cero tiene frecuencia 24 Hz, la tercera tiene 240 Hz (para que coincida con la frecuencia de corte del filtro), la quinta tiene 2400 Hz, y las cinco frecuencias se distribuyen en una escala logarítmica de forma homogénea. Al final de todo el texto, en el apéndice, se hace una descripción del tema de cómo hacer para lograr una separación homogénea en una escala logarítmica.
b. Graficar la señal (x) y su espectro (X) para obtener los dos gráficos de la izquierda de la figura 3 (pág. 2). Revisar la escala horizontal temporal del gráfico superior para verificar que se obtiene la misma señal.
c. Obtener el espectro de la señal de salida (Y) multiplicando punto a punto la variable H y la variable X. Esta multiplicación debe hacerse sin utilizar abs() ya que el proceso que un filtro realiza incluye también efectos sobre la fase (aunque en las explicaciones anteriores lo hayamos pasado por alto para volver más comprensible lo que se pretendía describir).
d. Graficar el espectro en dB de la señal de salida como se muestra en el gráfico inferior derecho de la figura 3.
e. Obtener la señal de salida temporal (y), calculando la antitransformada de (Y) mediante la función ifft(). y graficarla para obtener la curva en color rojo de la figura (3), arriba a la derecha.
f. Escuchar primero la señal original y luego la filtrada.
NOTA: La instrucción sound() asume una amplitud máxima de señal igual a 1 (del mismo modo que la información que se guarda en un archivo .wav), de manera que si el resultado de las operaciones realizadas da mayor que 1, la señal sonará con saturación. Como la señal x(t) fue generada sumando 5 senoidales de amplitud 1, en algún momento podrán coincidir sus máximos y llegar a un total de 5, por este motivo sugerimos dividir por 5 la señal al escucharla. Se sugiere además utilizar algunas de las siguientes alternativas para que no se superpongan los sonidos al escucharlos.

Alternativa 1
sound(x/5,fs)
pause(1) % pausa de 1 segundo
sound(y/5,fs)

Alternativa 2
sound([x/5 zeros(1,fs) y/5],fs)

En la segunda alternativa se generó un vector encadenando la variable x/5, con 48000 ceros (1 segundo) y luego la variable y/x. Si se desea una duración diferente en segundos de la pausa entre sonidos puede usarse zeros(1,duracion*fs)

Existe una instrucción en MATLAB que autoescala el sonido para no saturar. Se denomina soundsc(). Es útil en algunos casos, pero es necesario tener en cuenta que realiza un proceso de normalización automática que puede llevar a conclusiones erróneas respecto del nivel de lo que estamos escuchando. Esto es, si se está haciendo un cambio de amplitud para bajar la intensidad, al usar soundsc() comparando ambos sonidos no se notará diferencia si se utiliza la alternativa 1 anterior.

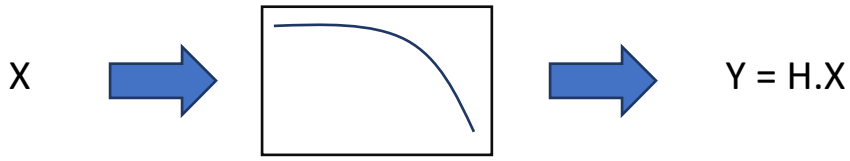


Figura 5 - Representación del proceso de filtrado utilizando variables espectrales (en frecuencia)

NOTA: En la ecuación que permite calcular Y como H.X no se ha utilizado ".*" (punto por), ya que en realidad dicha operación no se especifica normalmente en matemática de ese modo. La diferencia entre lo que MATLAB haría al escribir solamente "*" y escribir ".*" se debería comprender del contexto en el que se esté utilizando. Dentro de esta asignatura no utilizaremos el producto de tipo matricial (que es el que MATLAB interpreta con solamente un asterisco), de modo que en las ecuaciones de explicación utilizaremos solamente el "." para representar mutliplicación, recordando que al hacer la cuenta en MATLAB con dos vectores será necesario escribir ".*".

3. Cargar archivo de audio uno.wav y filtrarlo con H. Escuchar los resultados comparados. NOTA: Recordar que si H es un vector fila puede ser necesario transponerlo (con la función transp) para que sea un vector columna como los audios.

FUNCION DE TRANSFERENCIA DE UN SISTEMA

La función H es algo que define perfectamente lo que el filtro hace. Por intentar ser poético, podría decir que "contiene su esencia". Esta función tiene un nombre especial, se denomina "Función de Transferencia" y suele representarse en la literatura específica mediante la letra H mayúscula. No solamente se utiliza para este y en realidad cualquier otro filtro, sino que puede representar lo que una gran mayoría de los sistemas de procesamiento de señales, como por ejemplo sistemas que produzcan eco o reverberancia, también líneas de retardo, e incluso algo mucho más complejo como lo que representa el efecto completo de transformaciones producidas por la cabeza humana a las señales y que permite percibir sensación de audio 3D. En este caso del audio 3D se habla de la función de transferencia de la cabeza (HRTF, por Head Related Transfer Function), y ellas se utilizan para generar señales binaurales que produzcan sensación de posicionamiento 3D (en audio inmersivo o en videojuegos, por ejemplo). Técnicamente se puede representar mediante una función de transferencia H cualquier sistema que sea lineal e invariante en el tiempo (se los conoce como sistemas LTI). Dejaremos por el momento una discusión más fina de lo que significa un sistema LTI y de las condiciones que debería cumplir.

La función de transferencia H es una función matemática (o un conjunto de datos digitalizados de frecuencia)
¿Cómo podría alguien obtener la función de transferencia de un sistema práctico existente?
Una manera trabajosa podría ser colocar a la entrada una señal x(t) de una sola frecuencia, registrar la señal y(t) y determinar cuánto cambió su amplitud y cuánto cambió su fase, anotando estos valores como los que corresponden a la función de transferencia H exactamente para esa frecuencia. En realidad, los valores que contiene H son los valores real e imaginario, por lo que habría que convertir el módulo y la fase obtenidos por la medición al formato binomial de un número complejo y eso colocarlo como el valor complejo de H en esa frecuencia.
El proceso es trabajoso porque habría que repetirlo cuidadosamente frecuencia por frecuencia. Si se desea obtener H para lo que hicimos previamente habría que repetir la medición decenas de miles de veces para recorrer f completamente.

Respuesta al Impuso de un Sistema LTI

Por suerte, existe un procedimiento más directo, que nos lleva a considerar el concepto de "respuesta al impulso"
Sabemos a esta altura que si se conoce la función de transferencia H y el espectro de una señal de entrada X y se multiplica cada componente de H por cada componente de X se obtiene el espectro de la señal de salida (la señal procesada por el sistema). Si pudiéramos tener una señal de entrada cuyo espectro fuese perfectamente plano (de amplitud 1 en el caso ideal), resultaría que la salida Y sería igual a multiplicar cada componente de H por 1 y entonces tendríamos a H exactamente representada en Y (que se puede registrar porque lo podemos medir externamente). En otras palabras, normalmente sabemos que H existe, pero no podemos registrarla a menos que logremos tener algo en la salida del sistema. Si logramos que Y (que se puede registrar desde la salida del sistema) sea igual a H, entonces podríamos afirmar que "logramos robarle el alma al sistema".
¿Qué tipo de señal deberíamos colocar a la entrada como x(t) para que su espectro sea completamente plano?
Un impulso ideal tiene una respuesta plana de amplitudes lo largo de todas las frecuencias, y por lo tanto al aplicar un impulso a la entrada de un sistema, la respuesta que produzca el sistema se llamará con toda propiedad "respuesta al impulso" y su espectro tendrá que coincidir con H.

4. Generar un impulso unitario de 1 segundo de duración (frecuencia de muestreo a elección) y verificar que su fft() es un espectro plano. Observar las fases. NOTA: Recordamos aquí que un impulso unitario se genera creando un vector con todos ceros y luego haciendo que su primer elemento valga 1.
5. Exportar el impulso como wav, abrir desde un editor de audio, filtrar (con cualquier filtro que deseen, preferiblemente algún pasa bajos por claridad del ejemplo) y guardar como "rtaImpulso.wav". Cargar en MATLAB y observar la fft() de la rta al impulso. Podrá verse el perfil del filtro utilizado, y también cómo afecta las fases, si se grafican las fases del mismo.
6. Aplicar el filtro anterior a un audio (uno.wav)
7. Repetir con un EQ gráfico y aplicarlo al audio uno.wav y a x(t), escuchando los resultados

LA FUNCIÓN DE TRANSFERENCIA ES LA TRANSFORMADA DE LA RESPUESTA AL IMPULSO

Colocamos un impulso en la entrada para tener un espectro plano, y registramos la salida (respuesta al impulso. Sabiendo que el sistema hará algo semejante a multiplicar H por el espectro de entrada y dado que se trata de un espectro plano, se obtendrá a la salida un espectro cuyo perfil es el de H. Esto significa que H no es otra cosa que la transformada de Fourier de la respuesta al impulso.
Es por este motivo que es común identificar a la respuesta al impulso de un sistema con una letra h minúscula. Es muy común también identificar la señal impulso con la letra griega delta minúscula, escribiendo el impulso como δ(t). En la teoría matemática de señales continuas se conoce a la función impulso como Función Delta de Dirac, en honor al físico Paul Dirac que la propuso. En señales continuas la función Delta de Dirac (impulso) es mucho más complicada de generar, ya que teóricamente debería tener duración exactamente cero, pero de todas maneras debería tener energía, lo que a primera vista parece un contrasentido. La función propuesta por Dirac tiene una amplitud infinita con duración cero (algo complicado de explicar con detalle en este contexto), pero por suerte con señales discretas la duración es la de la mínima diferencia de tiempos entre muestras (dt = 1/fs).
Resumamos el procedimiento que fue aplicado en los ejercicios anteriores. Supongamos que deseamos aplicar utilizando MATLAB un tipo de proceso de filtrado que es implementado por un software en particular. Generamos en MATLAB un impulso de un segundo de duración, lo abrimos desde el software, aplicamos el proceso deseado y guardamos el resultado de la respuesta al impulso (h). Regresamos a MATLAB y cargamos la respuesta al impulso, obtenemos su fft() y la guardamos como H (Función de transferencia de ese proceso). Luego tomamos cualquier señal temporal x de un segundo de duración, obtenemos su espectro X y lo multiplicamos por la función de transferencia H para obtener Y (que es el espectro de la señal procesada). Antitransformamos Y para obtener la respuesta final del sistema en el que se aplicó el proceso determinado por H.
NOTA: Para poder multiplicar la función de transferencia H, por el espectro de la señal X, ambas deben tener la misma cantidad de muestras. Luego analizaremos qué problemas habría que enfrentar si la duración de alguna de las señales no tuviese la misma duración (de un segundo en estos ejemplos). Por el momento seguiremos con esa característica para facilitar la comprensión general de los fenómenos.

CONVOLUCIÓN DE SEÑALES

Existe una operación muy importante en el área de procesamiento de señales que se denomina convolución y que corresponde a operar con dos señales para obtener una nueva señal como resultado. La definición de cómo obtener la convolución entre dos señales en sistemas continuos es bastante complicada de plantear y de comprender. Formalmente si tenemos una señal x(t) y otra señal h(t), la convolución entre estas dos señales para obtener una señal y(t) se expresa de la siguiente manera

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(t-\tau)\,.\; x(\tau)\, d\tau$$

NOTA: El símbolo * es utilizado en Matlab para realizar una multiplicación. Sin embargo, en la notación formal en matemática es uno de los símbolos que indican que la operación a realizar es la convolución entre señales.
En señales discretas también es bastante difícil de comprender en primera instancia, y se expresa del siguiente modo

$$y[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} h[n-k] - x[k]$$

¿Qué se pretende obtener con la convolución entre dos señales?
En principio podríamos decir que es un modo de realizar todo el proceso que fue realizado en los ejercicios previos pero utilizando solamente las variables en el dominio del tiempo. Dicho en otras palabras, si x(t) es la señal de audio y h(t) es la respuesta al impulso de un filtro, por ejemplo, entonces la convolución de x(t) con h(t) da por resultado la señal y(t) que es lo que se obtendría si se ingresa x(t) en ese filtro.

Si se opera en el dominio de la frecuencia hay que partir de x(t), para obtener X(f), multiplicar luego punto apunto por H(f) para llegar a Y(f) y por último antitransformar para llegar a y(t). Si en cambio se opera en el dominio del tiempo, se parte de x(t), se convoluciona con h(t) y con ello se obtiene la señal filtrada y(t).

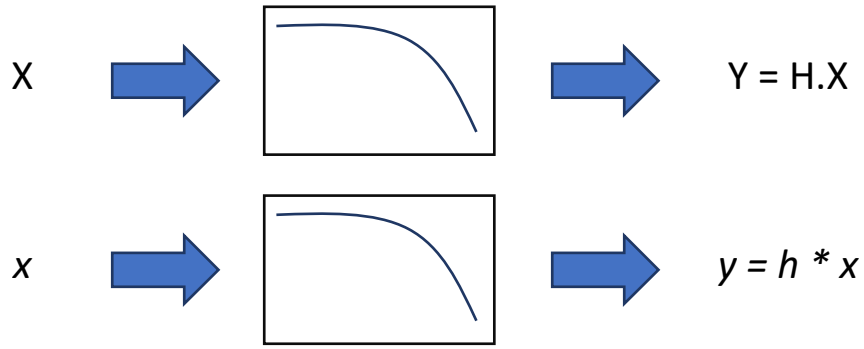


Figura 6. El diagrama superior ilustra el proceso a realizar en el dominio de la frecuencia, que corresponde a multiplicar el espectro de la señal X por la Función de Transferencia H. En el diagrama inferior puede verse el mismo proceso en el dominio del tiempo, en el cual se convoluciona la señal x(t) con la respuesta al impulso del sistema h(t) para obtener y(t)

En MATLAB existe la función conv() que realiza la convolución entre dos variables que se colocan como argumentos. La función de convolución tiene el mismo comportamiento si se convoluciona x con h, que si se convoluciona h con x. Esto significa que será igual hacer conv(h,x), que hacer conv(x,h). Esta situación es más clara si se piensa en el dominio de la frecuencia ya que multiplicar H por Xgenera el mismo resultado que multiplicar X por H.

8. Repetir el ejercicio 2, pero utilizar la función conv(x,h) para obtener la señal filtrada y(t).
Aclaración importante: Cuando multiplicamos dos espectros en Matlab, la cantidad de muestras de cada señal debe ser la misma, y el resultado de la multiplicación tendrá igual cantidad de muestras. Cuando se utiliza la convolución, cada muestra de la señal genera una respuesta al impulso a continuación. Esto hace que la última muestra de la señal tenga que tener 'cola' y por lo tanto la duración de una convolución es mayor que la de la señal. Exactamente es igual a la duración de la señal más la duración de la respuesta al impulso menos uno. Si se desea graficar con un eje de tiempos o de frecuencias el resultado de la convolución será necesario redefinir estos ejes (con nuevos nombres de variables, como por ej. taux, faux. El eje de tiempos mantendrá el mismo dt=1/fs y el eje de frecuencias tendrá la misma f final (fs).

9. Obtener el filtrado del audio uno.wav utilizando la respuesta al impulso de un filtro a elección (H original o uno obtenido mediante la respuesta al impulso con un filtro de algún editor de audio).

RESPUESTA AL IMPULSO DE RETARDO, ECO O REVERBERANCIA

En ciertos casos es relativamente fácil imaginarse lo que puede esperarse como respuesta al impulso. Tomemos por ejemplo un sistema que provoque retardo (delay). ¿Qué esperamos que resulte como respuesta si se coloca un impulso a la entrada? Es fácil comprender que lo que se espera obtener es un impulso pero que no comience en tiempo cero, sino después de un tiempo equivalente al delay. Esto significa que un modo de obtener un retardo en una señal podría ser construir manualmente una respuesta al impulso que sea simplemente el mismo impulso desplazado cierta cantidad de muestras.

10. Generar una respuesta al impulso que corresponda a un impulso desplazado medio segundo (24000 muestras). Realizar la convolución entre el audio uno.wav y la respuesta al impulso creada. Plotear el audio original y el procesado (tener en cuenta que será necesario definir dos variables de tiempo ya que tendrán diferente duración). Escuchar los resultados. NOTA: Analizar si es necesario transponer H para que se vuelva vector columna

Utilizando un razonamiento similar podemos imaginar qué tipo de respuesta al impulso puede esperarse de un eco sencillo. Es fácil comprender que si se entrega un impulso a un sistema que provoque eco, la respuesta al impulso consistirá en una secuencia de impulsos de amplitud decreciente que van apareciendo en distintos momentos a lo largo del tiempo.

11. Generar una respuesta al impulso que corresponda a un eco simple mediante una secuencia de cuatro impulsos de amplitudes 0.8, 0.5, 0.3, 0,1 ubicados en t=0, t=0.3, t=0.6, t=0.9 segundos. Realizar la convolución entre el audio uno.wav y la respuesta al impulso creada. Plotear el audio original y el procesado (tener en cuenta que será necesario definir dos variables de tiempo ya que tendrán diferente duración). Plotear los resultados
Repetir con clap.wav, plotear los resultados y escuchar. NOTA: Analizar si hay que transponer H

Siguiendo la misma lógica podría construirse una reverb muy elemental creando una respuesta al impulso que colocase valores impulsivos en diferentes tiempos, sólo que a diferencia del eco estos deberían encontrarse mucho más juntos entre sí (tiempos menores a 50 o 100 ms).

12. Generar una respuesta al impulso que corresponda a una reverb artificial muy sencilla como la que se ve en la figura.

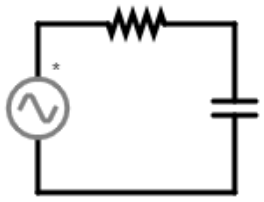
Cada nuevo valor es la mitad del anterior, el eje de tiempos está en segundos. Convolucionar esta respuesta al impulso con el audio uno.wav, plotear y escuchar los resultados.
Podrá notarse que la reverb obtenida no es muy natural, pero que genera un resultado que podría catalogarse como aceptable.
Repetir todo con clap.wav. En este caso podrá detectarse que la sensación es muy pobre. Esto es debido a la breve duración del sonido en clap.wav que deja que se perciban correctamente las repeticiones dando lugar a una sensación de "flutter eco", o a una mala "hardreverb".

13. Generar un impulso de 1 segundo de duración, ingresarlo a un editor de audio, programar allí una reverb y guardar la respuesta al impulso. Ingresar esta respuesta a MATLAB y plotear la respuesta al impulso. Podrá notarse que si bien aparecen pulsos a distintos tiempos no todos son perfectamente impulsos y algunos incluso tienen amplitud negativa. Para comprender esto podemos pensar en lo que sucede en un recinto si se generase un impulso y se registra la respuesta al impulso que provocaría dicho recinto. Es perfectamente razonable suponer que si bien el impulso provocará nuevos picos en distintos tiempos no todos serán una copia exacta del impulso sin modificación de su forma. Dependiendo del tipo de materiales con los cuales se refleje, podrá haber cambios de fase (una reflexión invertida, por ejemplo), o una modificación en su espectro (por distintos valores de absorción para cada frecuencia de una superficie determinada).
Procesar los audios uno.wav y clap.wav para comparar los resultados del proceso realizado.

APÉNDICE

1. Función de transferencia de un filtro pasa bajos (analógico). Gráficos de módulo y fase.

El diseño de filtros digitales tiene sus vueltas, que analizaremos más adelante. La intención en esta parte es explorar posibilidades de observar la respuesta al impulso de un filtro de tipo analógico y comentar luego algunas particularidades del espectro que se obtiene al aplicar fft().
Un filtro analógico es el que puede resultar de utilizar una resistencia y un capacitor en electrónica. La siguiente configuración corresponde al filtro pasa bajos RC. Partiremos de este análisis para obtener una ecuación general de un filtro que se independizará de R y C, dependiendo solamente de la frecuencia de corte deseada.



Si se entrega una señal x(t) en la fuente y se mide tensión y(t) en el capacitor, se obtendrá una versión filtrada de la señal. La frecuencia de corte de este pasabajos es

$$f_c = \frac{1}{2\pi RC}$$

Donde R es el valor de resistencia en ohms y C el valor del capacitor en faradios. La reactancia del capacitor se calcula como $X_c = \frac{1}{2\pi fC}$ y su impedancia (siempre con -90°) será $Z_c = -j\frac{1}{2\pi fC}$, donde j es la unidad imaginaria (suele utilizarse en lugar de i, para no confundir con la corriente).
Para resolver el circuito había que calcular la impedancia total.

$$Z_{total} = R - jX_c$$

La corriente total (considerando que v e i son fasores) será

$$i_{total} = \frac{v_f}{Z_{total}}$$

La tensión en el capacitor será

$$v_c = Z_c . i_c = Z_c . \frac{v_f}{Z_{total}} = v_f . \frac{Z_c}{Z_{total}}$$

En conclusión, dada una señal vf, se la multiplica por un factor Zc/Ztotal (división de complejos) que depende de f.
Cada componente del espectro de una señal es un fasor (con su módulo y fase). La función de transferencia H será entonces

$$H = \frac{Z_c}{Z_{total}} = \frac{-j\frac{1}{2\pi fC}}{R - j\frac{1}{2\pi fC}}$$

Dividiendo el numerador y el denominador por R queda lo siguiente

$$H = \frac{Z_c}{Z_{total}} = \frac{-j\frac{1}{2\pi fRC}}{1 - j\frac{1}{2\pi fRC}}$$

Pero resulta que la frecuencia de corte era $f_c = \frac{1}{2\pi RC}$, por lo cual el término que está junto a $-j$ puede expresarse como la división entre f_c y f

$$\frac{f_c}{f} = \frac{\frac{1}{2\pi RC}}{f} = \frac{1}{2\pi f RC}$$

Entonces la función de transferencia de ese filtro queda expresada en forma más sencilla y más general

$$H = \frac{-j\frac{f_c}{f}}{1 - j\frac{f_c}{f}}$$

Donde fc es constante para ese filtro y f varía en todo el rango, dando el módulo y la fase de H para cada frecuencia.
Codifiquemos esta función de transferencia en Matlab
Primero las líneas de código generales para definir parámetros y los ejes

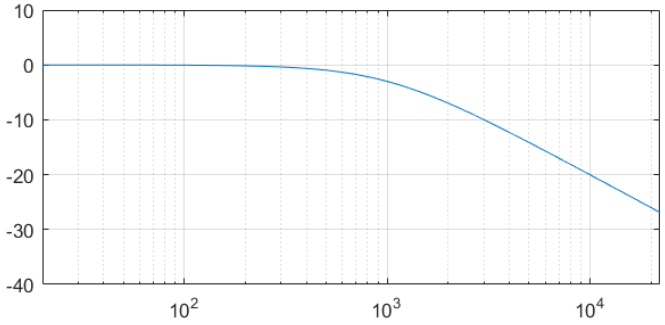
```
clear; clc
fs=44100;
dt=1/fs;
duracion=1;
t=0:dt:duracion-dt;
f=(0:length(t)-1)/length(t)*fs;
fN=fs/2; % frecuencia de Nyquist
fc=1000;
```

El centro del código será una sola instrucción que calcule H

```
H=-1i*(fc ./ f) ./ (1 - 1i*(fc ./ f));
```

Recordar que es necesario aquí utilizar ./ como requiere Matlab ya que estamos dividiendo por vectores, pero queremos que se divida punto a punto cada elemento de cada vector.

```
figure(1)
semilogx(f,20*log10(abs(H)))% ...
%    ,fc,20*log10(abs(H(fc+1))),'or')
axis([20 fs/2 -40 10])
grid on
```

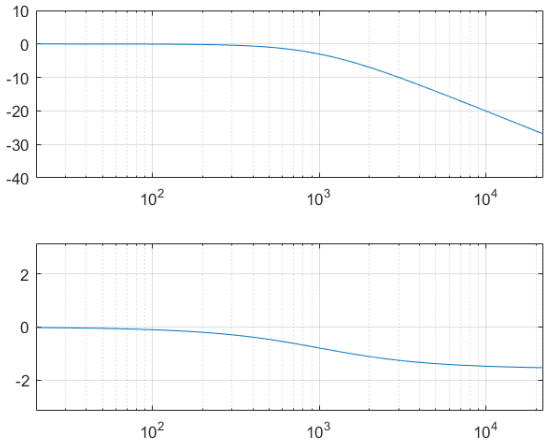


Deberíamos entender que este gráfico representa una sucesión de fasores (uno para cada muestra de frecuencia) que tiene módulo y fase.

¿Cómo representamos la fase?
Podríamos calcularla utilizando las partes real e imaginaria, y aplicando arco tangente con las instrucciones real(), imag(), atan(). Pero hay un camino más sencillo. Matlab dispone de la función angle(z), donde z es un valor complejo cualquiera y angle() devuelve el valor de su fase en radianes.
No es necesario hacer ningún cálculo nuevo. Al graficar el módulo usamos abs(H), y si lo hacemos en dB agregamos 20.log10(). Al graficar fase usamos angle(H). Los valores de ángulo van de -π hasta π, de modo que cambiaremos esto en el eje vertical del axis.

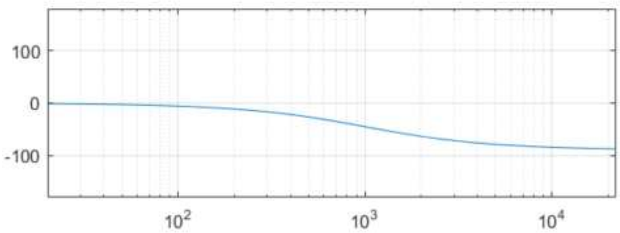
Modifiquemos el código para que puedan hacerse dos gráficos en la misma figura.

```
figure(1)

subplot(2,1,1)
semilogx(f,20*log10(abs(H)))
axis([20 fs/2 -40 10])
grid on

subplot(2,1,2)
semilogx(f,angle(H))
axis([20 fs/2 -pi pi])
grid on
```



Esta es la representación de módulo y fase de nuestro filtro (con la frecuencia de corte elegida de 1000 Hz).
Podría resultar conveniente cambiar el eje vertical para que muestre los valores en grados. Para ello solamente tendremos que dividir el resultado de angle() sobre pi y multiplicar por 180, cambiando además axis.

```
semilogx(f,angle(H)/pi*180)
axis([20 fs/2 -180 180])
```

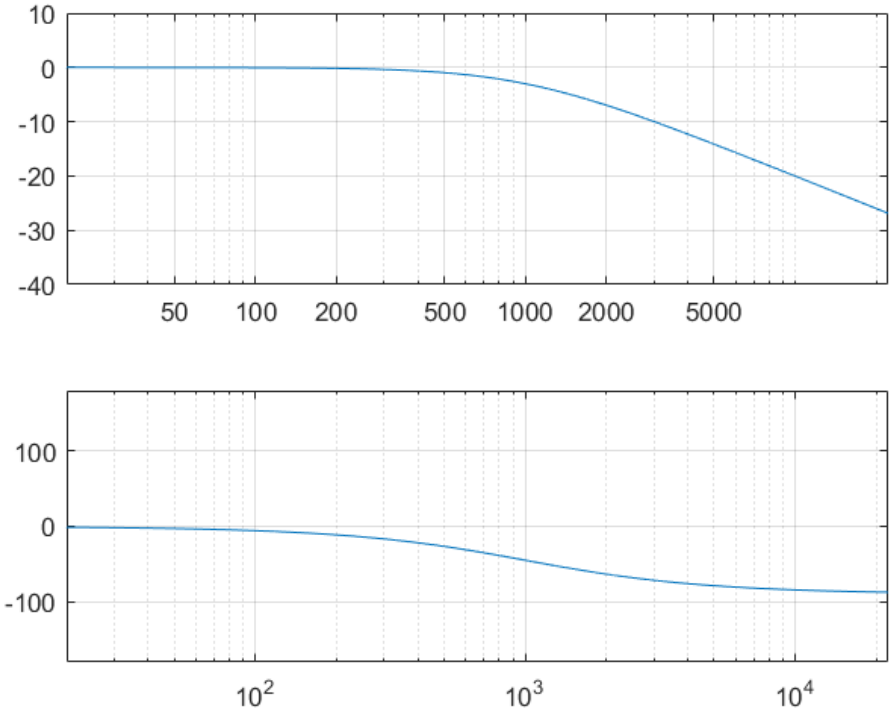


TIPS de Matlab para mejorar el aspecto del gráfico anterior

Matlab ofrece mucha posibilidad de variar cosas en los gráficos. Comentaremos aquí el uso de la instrucción xticks()
Se refiere a la posibilidad de indicar exactamente en qué valores de los ejes queremos que se hagan las marcas con valores numéricos.
La instrucción xticks() espera un vector que contenga la secuencia ordenada de valores que queremos que se indiquen.
Coloquemos, por ejemplo, luego de cada ploteo (en este caso semilogx) lo siguiente
xticks([50 100 200 500 1000 2000 5000])

NOTA: xticks() puede ir en cualquier lugar posterior a plot() o a semilogx(), afectará al último ploteo realizado.

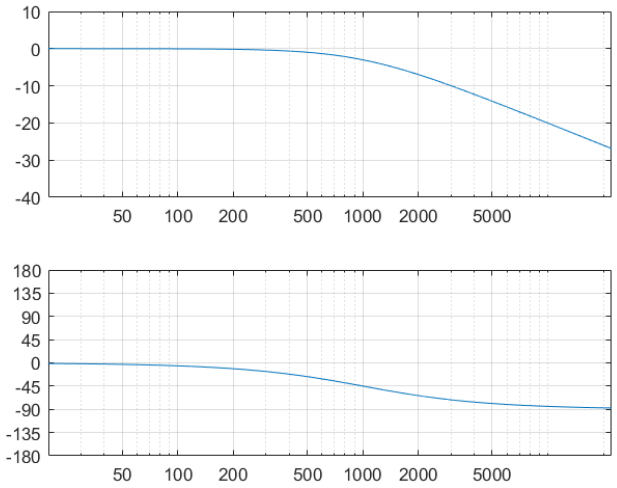
```
figure(1)
subplot(2,1,1)
semilogx(f,20*log10(abs(H)))
xticks( [50 100 200 500 1000 2000 5000] )
axis([20 fs/2 -40 10])
grid on
subplot(2,1,2)
semilogx(f,angle(H))
xticks( [50 100 200 500 1000 2000 5000] )
axis([20 fs/2 -pi pi])
grid on
```



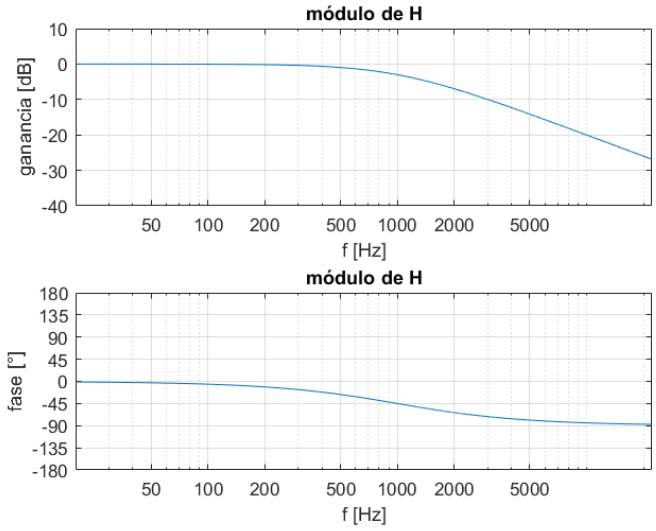
Los ejes están numerados justo donde queremos.

Podemos entonces hacer algo con los ángulos, como por ejemplo mostrar marcas cada 45°, utilizando luego del segundo ploteo la instrucción yticks()

```
yticks([-180:45:180])
```



Por último podemos hacer uso de title(), xlabel() e ylabel() para incluir información en los gráficos.



Puede notarse que en este filtro pasa bajos, para valores bajos de frecuencia no hay alteración del módulo de la señal (se lo multiplica por uno, que es 0dB) y tampoco cambio de fase (se suma 0°). Justo en la frecuencia de corte la ganancia es de 3 dB (se multiplica el valor del módulo por 0.7071, dado que 20.log10(0.7071) es -3) y tiene una fase de -45°, por lo que se restan 45° a la fase de la señal en ese componente de frecuencia. Para valores elevados de frecuencia la amplitud disminuye progresivamente y la fase tiende a 90°.

2. ¿Por qué el espectro obtenido por fft() tiene esa rara simetría?

El espectro que se construyó en el punto anterior fue realizado pensando en un caso analógico (como si se tratase de una señal x(t) continua). Sin embargo cuando se aprovechan las instrucciones que existen en los programas de procesamiento de señales se presentarán algunas diferencias, como esta de la "simetría" de la transformada de Fourier.
Hay dos cuestiones a tener en cuenta para comprender por qué los espectros obtenidos mediante la fft() tienen esa característica de simetría en módulo y antisimetría en fase.

En matemática existe algo que se conoce como la identidad de Euler (ser pronuncia óiler) y que expresa lo siguiente

$$e^{j\phi} = \cos(\phi) + j.sen(\phi)$$

Para que esta ecuación resulte válida es obligatorio que el ángulo ϕ esté expresado en radianes.
Consideremos por ejemplo un ángulo de 60°, que será (60/180).π = π/3 radianes y veamos lo que hace Matlab
Tipeen en la consola de comandos lo siguiente (dejando que haga eco en pantalla)

```
>> exp(1i*pi/3)
```

ans = 0.5 + 0.86603i

Veamos ahora cuánto vale el coseno y el seno de π/3

```
>> cos(pi/3)
```

ans = 0.5

```
>> sin(pi/3)
```

ans = 0.86603

Podemos notar que los valores coinciden con lo que indica la identidad de Euler.
Por otra parte, si cambiamos el signo del ángulo, solamente cambia el valor del seno (por propiedades del seno y el coseno).

$$e^{-j\phi} = \cos(-\phi) + j.sen(-\phi) = \cos(\phi) - j.sen(\phi)$$

Por motivos de eficiencia (difíciles de comprender con lo que sabemos hasta ahora), las transformadas se calculan utilizando exponenciales complejas. Por lo que curiosamente, el valor de la componente coseno de un ángulo se obtiene de un modo sumamente extraño para quienes no están familiarizados.

$$\cos(\phi) = \frac{1}{2}\,(e^{j\phi} + e^{-j\phi})$$

Esto es cierto porque

$$\cos(\phi) = \frac{1}{2}\left(e^{j\phi} + e^{-j\phi}\right) = \frac{1}{2}\,(\cos(\phi) + j.sen(\phi) + \cos(\phi) - j.sen(\phi))$$

Los términos imaginarios se cancelan y quedan dos cosenos, que al multiplicar por 1/2 termina siendo coseno.

¿Por qué tan complicado? No es sencillo comentar aquí las ventajas, pero así es como se hace en los cálculos internos de la transformada.

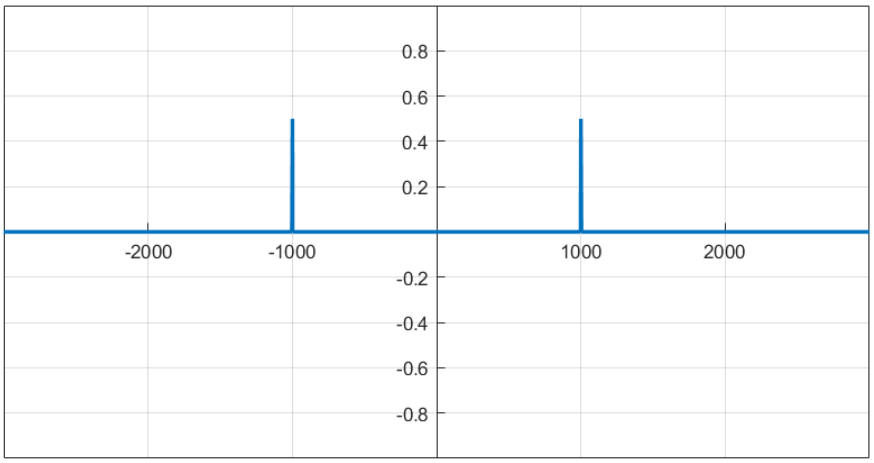
La cuestión es que si estamos representando una oscilación con frecuencia f, entonces el componente coseno será

$$\cos(2\pi f\ t) = \frac{1}{2}\ (e^{j\ 2\pi f\ t} + \ e^{-j\ 2\pi f\ t})$$

Las transformadas entregan componentes de exponenciales complejas (mientras que los humanos principantes en estos temas pensamos en componentes coseno). De modo que el código dirá que existe una componente 1/2 que corresponde a la primera exponencial y otro componente 1/2 que corresponde a la segunda. Pero como ambos tienen signos cambiados se interpreta que el primer componente corresponde a la frecuencia f, y el segundo a la frecuencia -f.

¿Frecuencia negativa? Si, tan raro como suena. Es ilógico pensar en una oscilación con frecuencia negativa, pero los matemáticos siempre han sido imaginativos para extender algunas ideas, por lo que asumen que es posible pensar en frecuencias negativas para los componentes de exponenciales complejas.

Si utilizamos, por ejemplo, la señal cos(2π1000 t) y le aplicamos la transformada de Fourier de exponenciales complejas, obtendríamos algo como lo siguiente



Con el seno es ligeramente más complicado (más?????), ya que en lugar de sumar hay que restar para que se cancele la parte de cosenos de la identidad de Euler, y hay que dividir por 2j, para quitar también la j imaginaria.

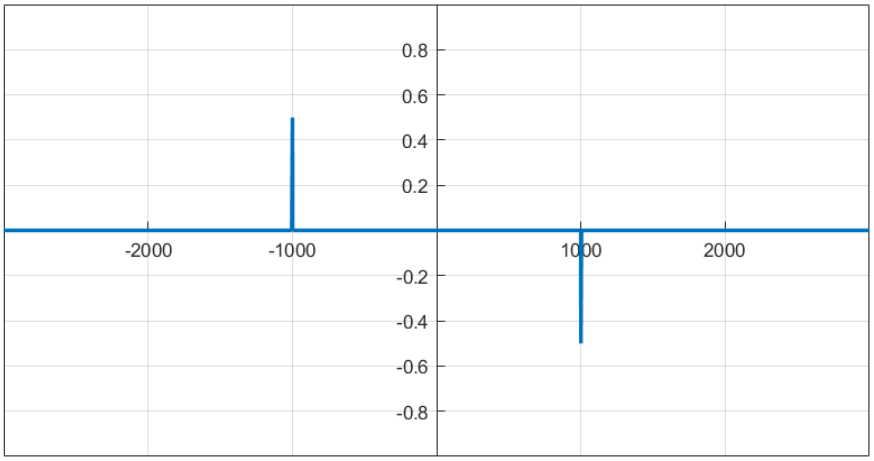
$$\text{sen}(\phi) = \frac{1}{2j}\ (e^{j\phi} - e^{-j\phi})$$

Si lo pensamos para una señal sen(2π1000 t), será

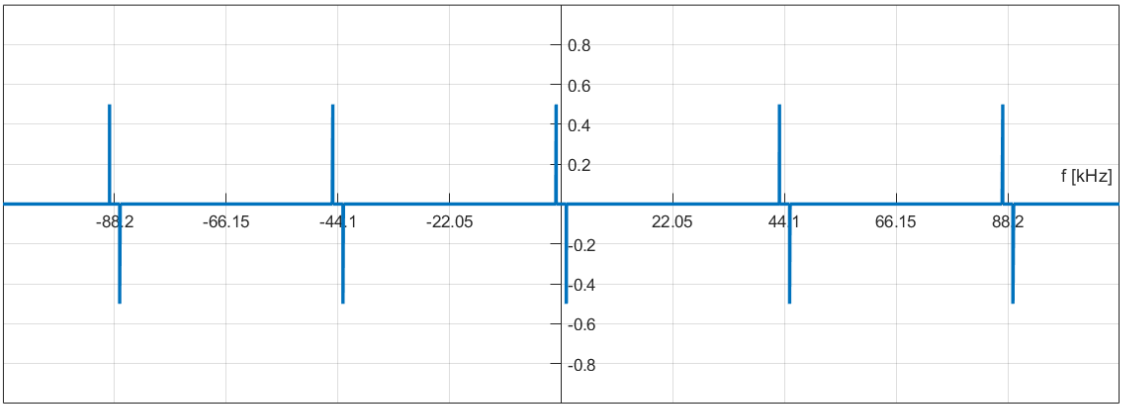
$$sen(2\pi 1000\ t) = \frac{1}{2j}\ (e^{j2\pi 1000\ t} - e^{-j2\pi 1000\ t})$$

El código de la transformada de Fourier basada en exponenciales complejas dirá que la componente de frecuencia 1000 vale 1/2j y la componente de frecuencia -1000 vale -1/2j.
Por las extrañas características de los números complejos, resulta que dividir por j, es lo mismo que multiplicar por -j. Esto se debe a que j por j es igual a menos uno (el inicio de toda la complicada historia de los números complejos era encontrar un valor que elevado al cuadrado diera por resultado menos uno).

Como resultado, el espectro de Fourier de esa senoidal de frecuencia 1000 se verá del siguiente modo (donde ahora el eje vertical es el eje imaginario j).



Todavía falta algo más (uffffff!!!!).
Los programas informáticos (Matlab, Octave, Phyton, y otros), no representan el espectro en frecuencias negativas. Los motivos de esto no son demasiado claros, pero todos hacen lo mismo. Representan la parte "negativa" de las frecuencias, como si estuvieran a continuación del eje, después de la frecuencia de Nyquist.
¿Es válido matemáticamente hacer esto? En realidad sí. Por motivos que no detallaremos aquí, el espectro de una señal discreta resulta ser repetitivo (se repite una y otra vez, en frecuencias más altas, como una especie de señal periódica en el dominio de las frecuencias).



Los programas informáticos muestran los componentes entre 0 y fs de la parte positiva.
Los cosenos tendrán un valor real positivo en la frecuencia correcta de 1 kHz en el ejemplo, y un "eco" simétrico real positivo en fs-1kHz. Los senos tendrán un valor imaginario negativo en la frecuencia correcta de 1 kHz, y un "eco" antisimétrico imaginario positivo en fs-1kHz. Obviamente si sólo se representan los módulos con la instrucción abs() entonces siempre se verán resultados simétricos (espejados respecto de la frecuencia Nyquist).
En general, esto no es algo que complique ya que sabemos que solamente las representaciones útiles de componentes serán hasta la frecuencia Nyquist, de modo que utilizando axis entre 0 y fN se tendrá el espectro de utilidad práctica.

3. Uso de la instrucción find()

En los ejercicios de mostrar la transferencia de un filtro se trazó la curva en color azul y se marcó el punto exacto de la frecuencia de corte con un círculo rojo.
El trazado de la curva completa es simplemente utilizando semilogx() con f completa y con H completa. Para trazar un sólo punto se necesita especificar en el eje x el valor elegido (la frecuencia de corte) y en el eje y el valor de H para la frecuencia de corte. El tema es que Matlab tiene un vector para f y otro para H. La frecuencia de corte es un valor para cierto elemento n del vector f. Para poder pedir que grafique el H correspondiente, necesitamos hallar el valor de esa n.
La instrucción find() permite hallar el lugar en que se encuentra el valor buscado.

Veamos primero ejemplos sencillos, para comprender qué hace la instrucción find.

Veamos qué sucede al tipear lo siguiente en la consola de comandos (dejando eco en pantalla)

```
>> a=[1 2 4 6 8 4 3 2 1]
>> find(a>7)

a =    1    2    4    6    8    4    3    2    1

ans =    5
```

Vemos primero el vector completo y luego la respuesta de find() que me dice que el quinto elemento tiene un valor mayor que 7

El tema es que find podría encontrar varios elementos que cumplen la condición. Esto sucedería si usamos a>5

```
>> a=[1 2 4 6 8 4 3 2 1]
>> find(a>5)

a =    1    2    4    6    8    4    3    2    1

ans =    4     5
```

También puede consultarse todos los que sean mayores o iguales que 4, por ejemplo

```
>> a=[1 2 4 6 8 4 3 2 1]
>> find(a>=4)

a =    1    2    4    6    8    4    3    2    1

ans =    3    4    5    6
```

Es posible pedir que nos muestre solamente el primero que cumple la condición, agregando una coma y el 1 dentro del find()

```
>> a=[1 2 4 6 8 4 3 2 1]
>> find(a>=4 ,1)

a =    1    2    4    6    8    4    3    2    1

ans =    3
```

La condición puede preguntar por valores mayores >, mayores o iguales >=, menores <, y menores o iguales <=.
Si queremos preguntar por valores iguales, es necesario utilizar un doble igual. Esto es por características de sintaxis de Matlab. Si colocamos a=4 cree que queremos que a valga 4, pero si colocamos a==4 considera que estamos preguntando si a es igual a 4.

```
>> a=[1 2 4 6 8 4 3 2 1]
>> find(a==4)
```

```
a =    1    2    4    6    8    4    3    2    1

ans =    3    6
```

En el caso de la frecuencia de corte, necesitamos el valor del índice del vector f, cuando f sea igual a fc. Podemos llamar a este índice nfc (número que corresponde a fc).

```
nfc=find(f==fc)
```

Sin embargo, hay que hacer una aclaración por cuestiones prácticas. El eje de frecuencias no es continuo, sino que tiene valores discretos. Podría darse el caso de que el salto que elegimos de f no valga nunca fc (porque en una muestras está poco antes de fc y en la siguiente se pasó por un poquito). Es más seguro entonces preguntar por el primer valor de f que sea igual o mayor que fc.

```
nfc=find(f>=fc ,1)
```

Luego utilizamos este valor de fc para graficar solamente ese punto
semilogx(... , f(nfc), 20*log10(H(nfc)), 'or')

4. ¿Cómo generalizar los casos cuando la cantidad de muestras de H y de X no son iguales?

Vimos que si conocemos la función de transferencia de un filtro H, entonces multiplicando cada muestra en frecuencia de H, por la muestra en frecuencia de X (espectro de la señal x(t)), obtendremos el espectro de la salida Y. Antitransformando mediante la instrucción ifft() podemos obtener la señal y(t). Si H representa un filtro, la señal y(t) quedará filtrada. Si representa una reverb, la señal y(t) habrá quedado afectada por esa reverb.

El problema con este procedimiento es que solamente sirve cuando H y X tienen la misma cantidad de muestras. La cantidad de muestras de X es la misma que la de la señal x(t). De modo que si tenemos una H de 44100 muestras, entonces este método solamente servirá para fragmentos de 1 segundo de duración de señal x(t).

¿Cómo hacer esto de modo más práctico?
Pues, un camino es utilizar la convolución, que es realizar este mismo procedimiento en el dominio del tiempo.

Recapitulemos:
El proceso en el dominio de la frecuencia, teniendo H y x(t) sería.
a. Obtener X=fft(x)*2/length(x);
b. Hallara Y multiplicando los espectros Y=H.*X
c. Obtener y(t) antitransformando y=ifft(Y);

El proceso en el dominio del tiempo, teniendo H y x(t) sería
a. Obtener la respuesta al impulso h(t) que corresponde a H. h=ifft(H)
b. Obtener la señal de salida y(t) convolucionando la señal x(t) con la respuesta al impulso h(t). y=conv(x,h)

Al trabajar en el dominio del tiempo no es necesario que ambas tengan igual cantidad de muestras. Lo que cambia es que la cantidad de muestras de la señal de salida y es mayor. Nada grave, pero hay que recordar esto al intentar graficar ya que será necesario definir nuevos ejes de tiempo y frecuencia para mostrar la señal y o su espectro Y.