Variable compleja y Análisis de Fourier

PRÁCTICA 3: CONSTRUCCIÓN DE UN FILTRO

Helena García Escudero

8 de abril, 2019

0.1. Introduction

Un filtro es un sistema que permite el paso de señales eléctricas a un rango de frecuencias determinadas e impide el paso del resto. Todo proceso de medida de una señal viene acompañado de variaciones aleatorias de mayor o menor intensidad a las que se les da el nombre genérico de ruido. El procedimiento para reducir o eliminar el ruido de una señal se conoce comúnmente como filtrado.Se define *filtro* , A , a todo sistema continuo e invariante en el sentido que

$$A(\tau_a x) = \tau_a(Ax) \tag{1}$$

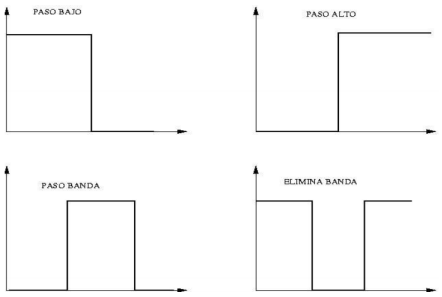


Figura 1: Diferentes tipos de filtros

En comparación con el filtro ideal, los filtros reales poseen ciertos defectos:

- La transición entre la banda que se quiere dejar pasar y la que se quiere eliminar no es abrupta, sino que tiene una determinada pendiente que depende del número de orden del filtro.
- La respuesta en fase no es linear, esto aumenta la distorsión de la señal significativamente.

No obastante se pueden construir distintos tipos de filtros de tal forma que se pretenda optimizar uno de los siguientes tres criterios:

- Una respuesta máxima plana en la banda de paso.
- Una transición rápida entre la banda de la señal deseada y la no deseada.
- Una respuesta de fase lineal.

Para conseguir este propósito, la función de transferencia (describe la relación entre la señal de salida y la de entrada) deberá tener polos complejos:

$$F(s) = \frac{A_0}{(1 + a_1 \cdot s + b_1 \cdot s^2) \cdot (1 + a_2 \cdot s + b_2 \cdot s^2) \cdots (1 + a_n \cdot s + b_n \cdot s^2)} \quad (2)$$

Los filtros que se pueden implementar a partir de este polinomio serán:

- Butterworth: Optimiza la respuesta plana en la banda de paso.
- Tschebyscheff: Tiene una respuesta más abrupta. Optimiza, por tanto, la transición.
- Bessel: Optimiza la respuesta en fase

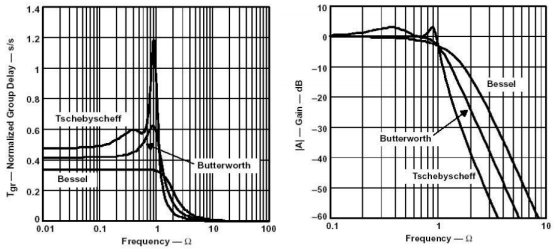


Figura 2: Comparación entre los tres tipos de filtros

0.2. Filtros de Butterworth

Comezaremos la práctica centrándonos en los filtros Butterworth. Los filtros Butterworth son filtros definidos en el espectro de energía por la expresión:

$$|H(\lambda)|^2 = \frac{1}{1 + (\frac{\lambda}{\lambda_c})^{2n}} \quad (3)$$

con $\lambda_c > 0$.

Debido a su respuesta plana, se suele usar en aplicaciones de conversión de datos; en general, donde sea necesario conseguir una buena precisión de medida en la banda de paso.

Estos filtros pueden obtenerse usando la función *butter* de MATLAB. En ella debemos dar como parámetros n (el orden) y la frecuendia de corte (que para la respuesta del filtro es $\frac{1}{\sqrt{2}}$, con lo que la frecuencia se reduce a la mitad), que debe estar normalizada en el intervalo [0,1]. Si queremos diseñar un filtro para

señales cuyo rango de frecuencias está entre 0 y 500 Hz y la frecuencia de corte deseamos que esté en 200 Hz, debemos tomar 200/500 como segundo parámetro para llamar a la función butter. Esta función devuelve un array en la que aparecen los coeficientes del filtro, los b son los del numerador y los a los del denominador, listados en potencias crecientes de λ. Una vez obtenidos estos coeficientes, emplearemos la función *freqz* que toma como parámetros de entrada los coeficientes b y a obtenidos por *butter*, el número de muestras con las que queremos representar la respuesta en frecuencia H y el doble de la frecuencia máxima de las señales a filtrar ($1000 = 500 \times 2ennuestrocaso$) y da como resultado H y las frecuencias λ en las que se muestrea (128 muestras entre 0 y 500 Hz). Por lo tanto, con un plot de abs(H) respecto de λ tenemos una representación de la respuesta del filtro.

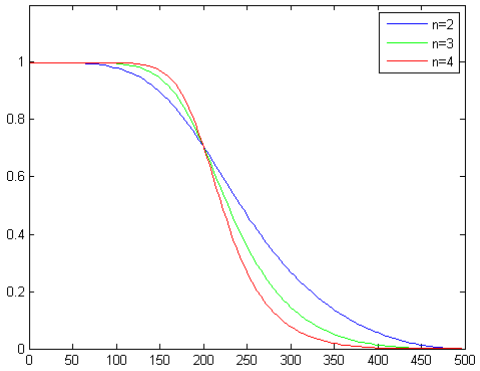


Figura 3: Filtro Butterworth para n=2, n=3 y n=4

Comenzaremos la práctica creando una colección de filtros de Butterworth de paso bajo de distintos órdenes, para ello, crearemos un script con el siguiente código:

```
%Filtros de Butterworth

%Creamos un array de colores para usar en la gráfica

color=['b','g','r','c','m','y','k'];
for n=2:8
    [b,a] = butter(n,200/500);
    [H,w]=freqz(b,a,128,1000);
    plot(w,abs(H),color(n-1))
    axis([0 500 0 1.2]);
legend({'n=2','n=3','n=4','n=5','n=6','n=7','n=8'})
```

```
    hold on
    pause

end
hold off

clear all
close all
```

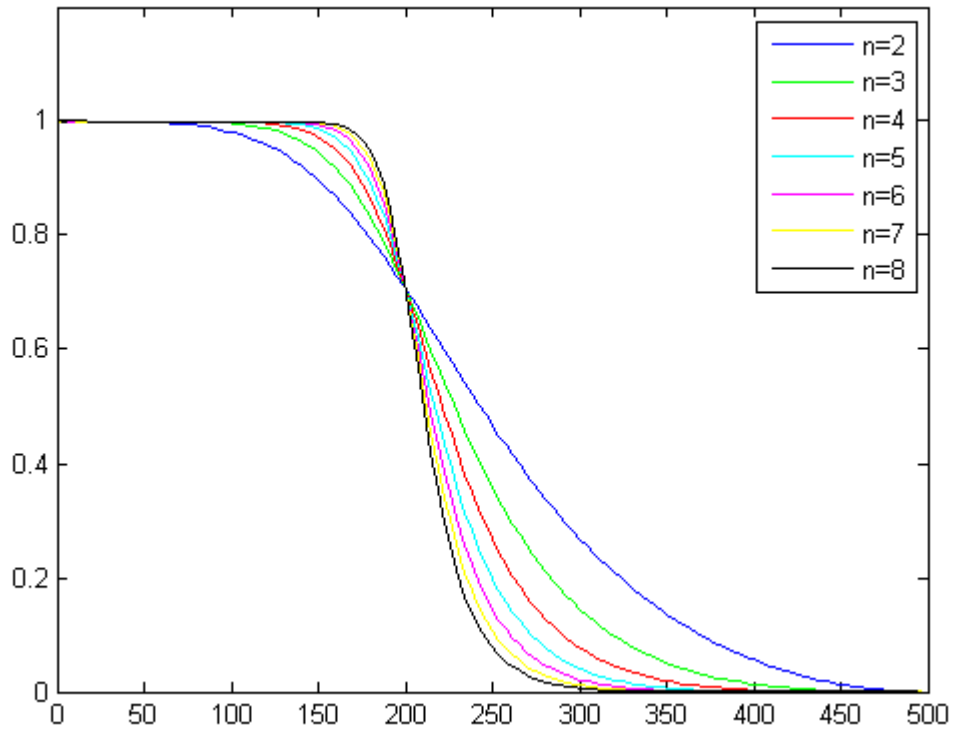


Figura 4: Filtro Butterworth para n=2 hasta n=8

El punto de corte es el (200, $\frac{1}{}sqrt2$). Cuanto mayor es el orden, mejor es la aproximación.

0.3. Filtros de Paso banda

Un filtro de banda atenúa las altas y bajas frecuencias, pero mantiene intactas las frecuencias que se encuentran en una banda determinada. En el caso del filtro ideal, solo deja pasar las frecuencias que están entre dos frecuencias de corte. Se puede obtener un filtro de banda multiplicando uno de paso bajo por uno de paso alto, en el que la frecuencia de corte del de paso bajo sea superior a la de paso alto. El opuesto al filtro de paso de banda seria de "rechazo de banda" en el que se atenúan las frecuencias de la banda pero se mantienen las frecuencias fuera de ella. Ahora construiremos un filtro paso-banda para filtrar una señal.

Para construir nuestro filtro vamos a considerar una señal que es la suma de 3 senos de frecuencias 5, 15 y 30 Hz, respectivamente. Queremos un filtro que mantenga el seno de 15 Hz y elimine los otros 2.

```
Fs=100;
t=linspace(0,1,Fs);
s1=sin(2*pi*t*5);
s2=sin(2*pi*t*15);
s3=sin(2*pi*t*30);
s=s1+s2+s3;
plot(t,s)
pause
```

Antes de generar el filtro vamos a detectar la frecuencia de la señal s (que es una suma de tres senos de frecuencias 5, 10 y 30 Hz, pero este dato en principio es desconocido cuando uno recibe una señal que él no generó). Usaremos la función *fft* de MATLAB.

El comando *fft* calcula la transformada discreta de Fourier (DFT) de X (parámetro de entrada) usando un algoritmo de transformada rápida de Fourier (FFT).

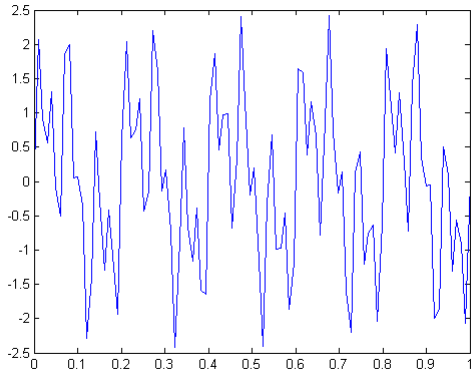


Figura 5: Detección de la frecuencia de la señal usando el comando *fft* de Matlab

Tomamos 512 valores de la *fft* para obtener una gráfica con bastante información. Como la fft es periódica, sólo nos interesa mostrar la mitad de sus valores

```
F=fft(s,512);
w=linspace(0,1,256)*(Fs/2);
plot(w,abs([F(1:256)]))
pause

figure
```

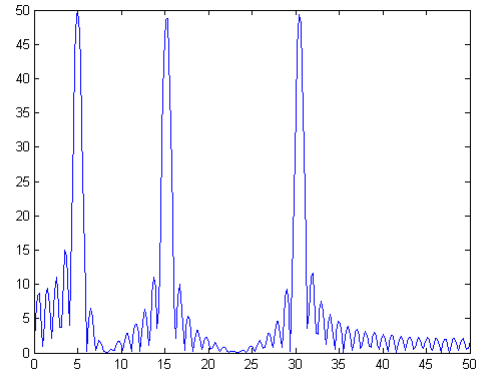


Figura 6:

Dada la periodicidad de la *fft* debemos diseñar el filtro simétrico respecto al origen con un array de 512 valores

```
f=round(256/(Fs/2));
H=[zeros(1,10*f) ones(1,(20-10)*f) zeros(1,256-20*f)];
H=[H fliplr(H)];
plot(w,H(1:256))
axis([0 Fs/2 0 1.2])

pause

figure
```

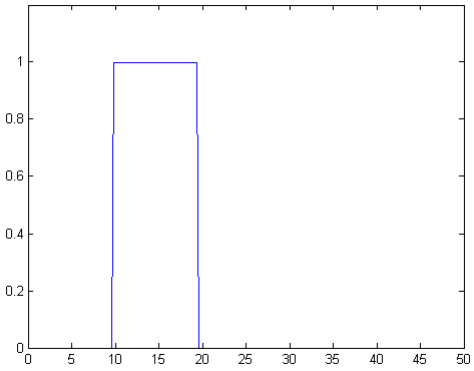


Figura 7:

```
HF=F.*H;
plot(w,abs(HF(1:256)))

pause

figure
```

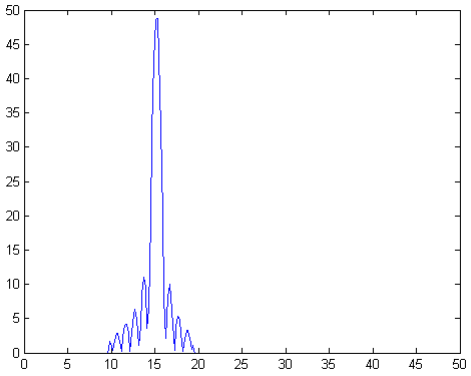


Figura 8:

```
inversa=ifft(HF);
plot(t,real(inversa(1:Fs)))

pause
```

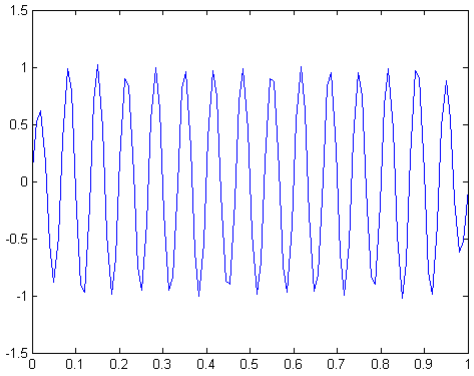


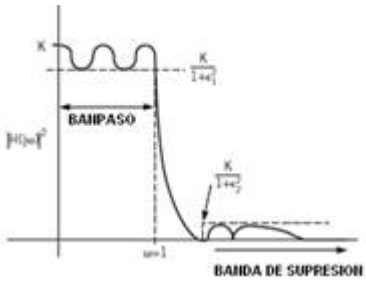
Figura 9:

Observamos que tenemos tres picos de frecuencia correspondientes a los valores de 5, 15 y 30 Hz. Efectivamente, la señal es la suma de tres senos con las frecuencias mencionadas anteriormente. Para quedarnos sólo con el seno de frecuencia 15 Hz hemos de diseñar un filtro que anule las otras dos.

¿Como podemos conseguir esto?

0.3.1. Filtros elípticos

El filtro elíptico a diferencia del resto de filtros estudiados, tiene un rizado en la banda de paso y en la banda de atenuación, este filtro es lo más cercano a un filtro ideal ya su banda de transición tiene una caída muy rápida, posee una caída abrupta en la banda de transición, que es su cambio desde la banda de paso a la banda de supresión . La mayor desventaja es que para su cálculo requiere de funciones elípticas por lo que su función de transferencia se hace muy complicada de resolver. En nuestro caso, lo construiremos un filtro elíptico usando la función ELLIP de MATLAB. Esa función requiere como parámetros de entrada: el orden, la caída en potencia en la banda de paso (tomaremos aquí 0.01dB), la caída de potencia en la banda atenuada (tomaremos aquí 40dB), un intervalo de las frecuencias a conservar, en este caso [10,20] multiplicado por la mitad de la frecuencia de muestreo de la señal (por el criterio de Nyquist).



[scale=0.5]

Figura 10: Filtro Elíptico de paso bajo

La función ELLIP devuelve, lo análogo a lo que devuelve BUTTER, y a continuación debemos usar FREQZ con los arrays b y a que devuelve ELLIP y la longitud del array de valores de H que queramos, tomaremos 512. Vamos a hacer con un bucle for 5 filtros, de órdenes: 2, 3,..., 6.

```
figure
color=['c','g','m','b','r'];
for n=2:6
    [b,a]=ellip(n,0.1,40,[10 20]*2/Fs);
    [H,w]=freqz(b,a,512);
    plot(w*Fs/(2*pi),abs(H),color(n-1))
```

```
    legend({'n=2','n=3','n=4','n=5','n=6'})
    grid
    hold on
end
    hold off

pause
```

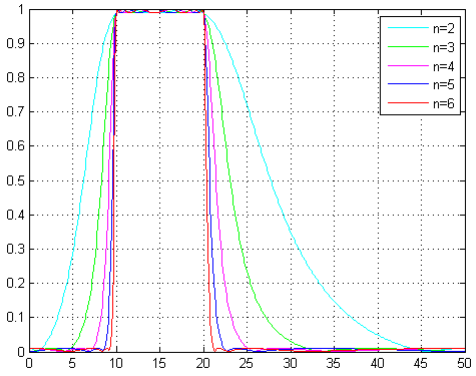


Figura 11:

Ahora vamos a usar la función FILTER para filtrar la señal s con el filtro H que acabamos de construir. Deben introducirse como parámetros los coeficientes b y a y la señal a filtrar.

```
for i=2:3:6
    [b,a]=ellip(i,0.1,40, [10 20]*2/Fs);
    [H,w]=freqz(b,a,512);
    sf=filter(b,a,s);
    subplot(2,1,1)
    plot(t,s2)

    axis([0 1 -2 2])
    title('seno de frecuencia 15 Hz original')

    subplot(2,1,2)
    plot(t,sf,color(i-1))
```

```
    legend({'n=2','n=6'})
    title('seno de frecuencia 15 Hz recuperada')
    grid
    hold on
end
    hold off

pause

close all
```

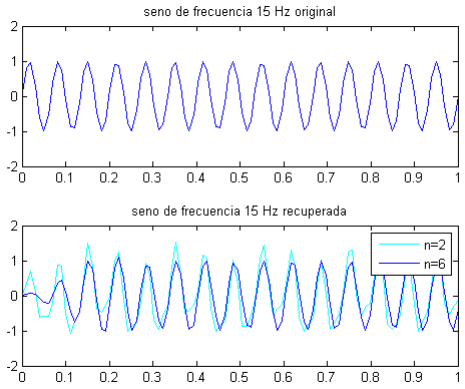


Figura 12: