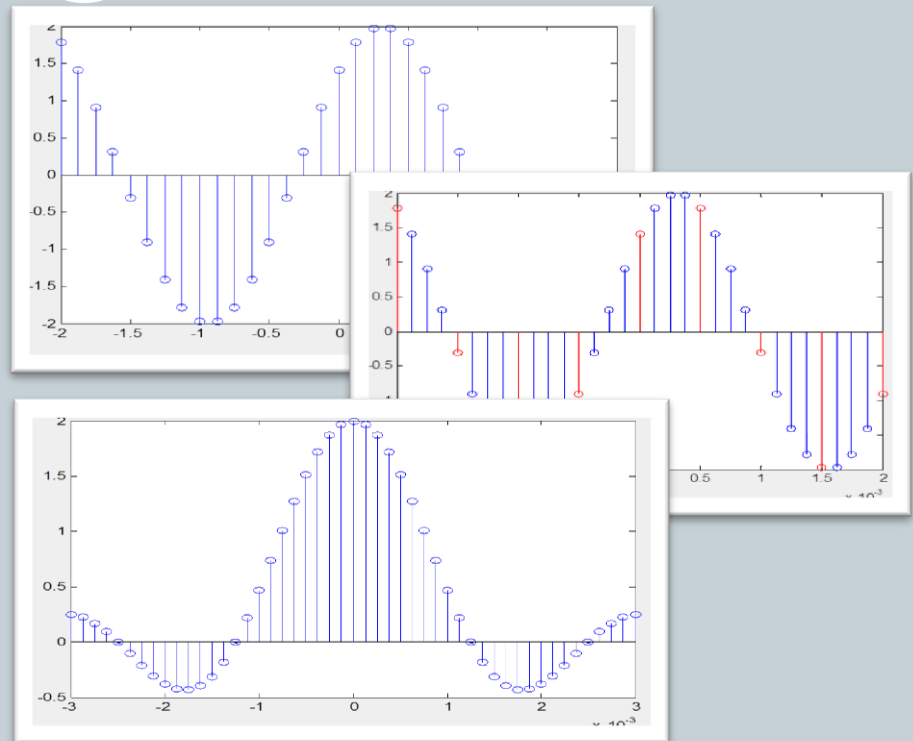


# PROCESAMIENTO DIGITAL DE SEÑALES CON MATLAB

1



# Generación y Gráfica de Señales

2

Sea una señal:

$$x(t) = A \cdot \sin(2 \cdot \pi \cdot F_o \cdot t + \phi)$$

**Dónde:**

A: Amplitud

Fo: Frecuencia

phi: Fase

La representación de esta señal en tiempo discreto está dada por:

$$x_d[n] = x(n \cdot T_s) = A \cdot \sin(2 \cdot \pi \cdot F_o \cdot n \cdot T_s + \phi)$$

**Donde:**

Fs=1/Ts: Frecuencia de muestreo.

La gráfica de señales discretas se realiza con el comando:

**stem(t,xt)**

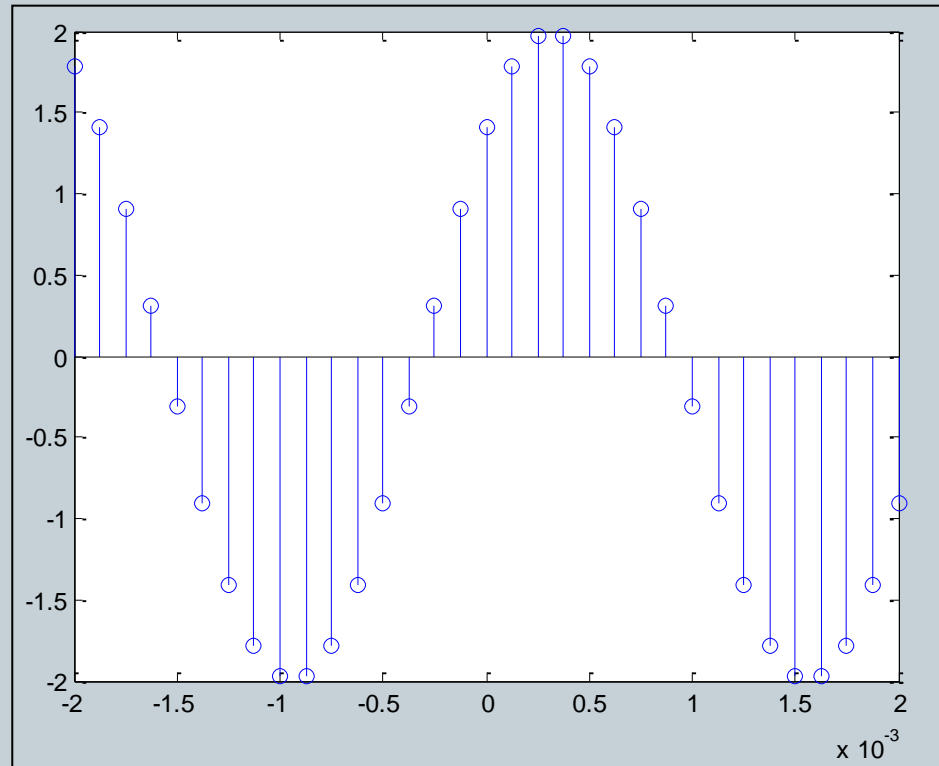
# Ejemplo: Generación y gráfica de una señal muestreada a 8 KHz.

3

Se genera una señal sinusoidal de 400 Hz y amplitud 2.

$$x(t) = 2 \cdot \sin\left(2 \cdot \pi \cdot 400 \cdot t + \frac{\pi}{4}\right)$$

```
>> F0=400;  
>> A=2;  
>> phi=pi/4;  
>> Fs=8000;  
>> Ts=1/Fs;  
>> t=-0.002:Ts:0.002;  
>> xt=A*sin(2*pi*F0*t+phi);  
>> stem(t,xt)
```

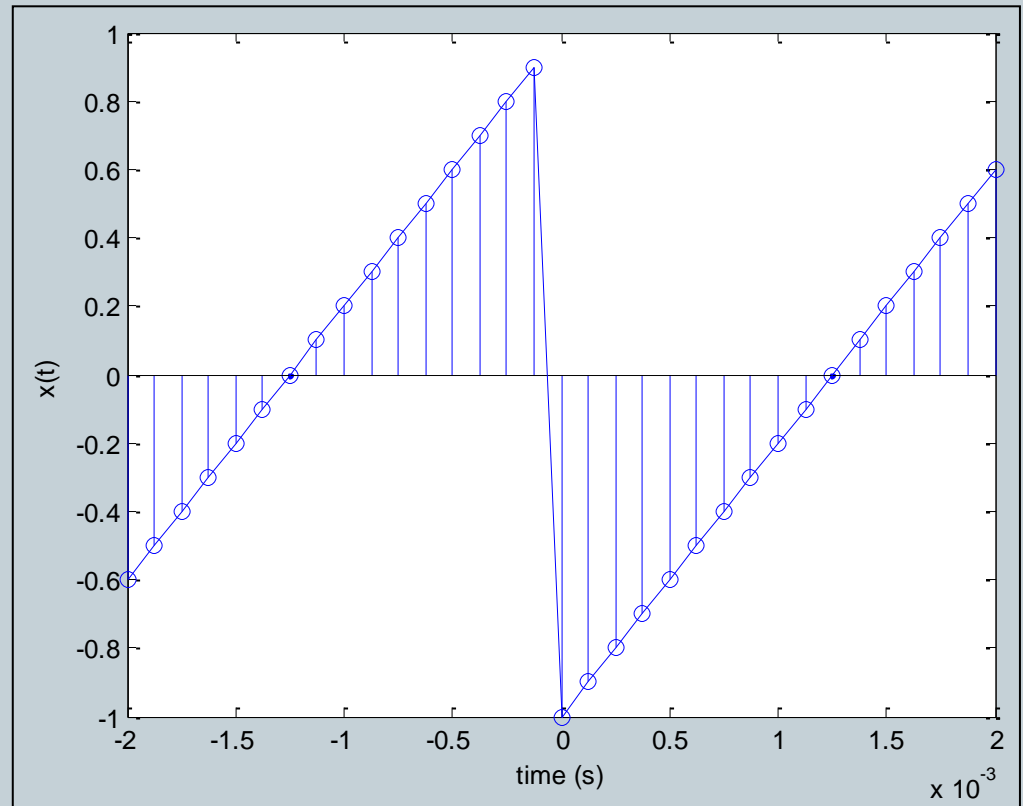


# Ejemplo: Señales no sinusoidales periódicas.

4

## Generación de una señal diente de sierra.

```
>> F0=400;  
>> A=2;  
>> Fs=8000;  
>> Ts=1/Fs;  
>> t=-0.002:Ts:0.002;  
>> xt=sawtooth(2*pi*F0*t);  
>> stem(t,xt)  
>> hold on  
>> plot(t,xt)  
>> xlabel('time (s)');  
>> ylabel('x(t)');
```

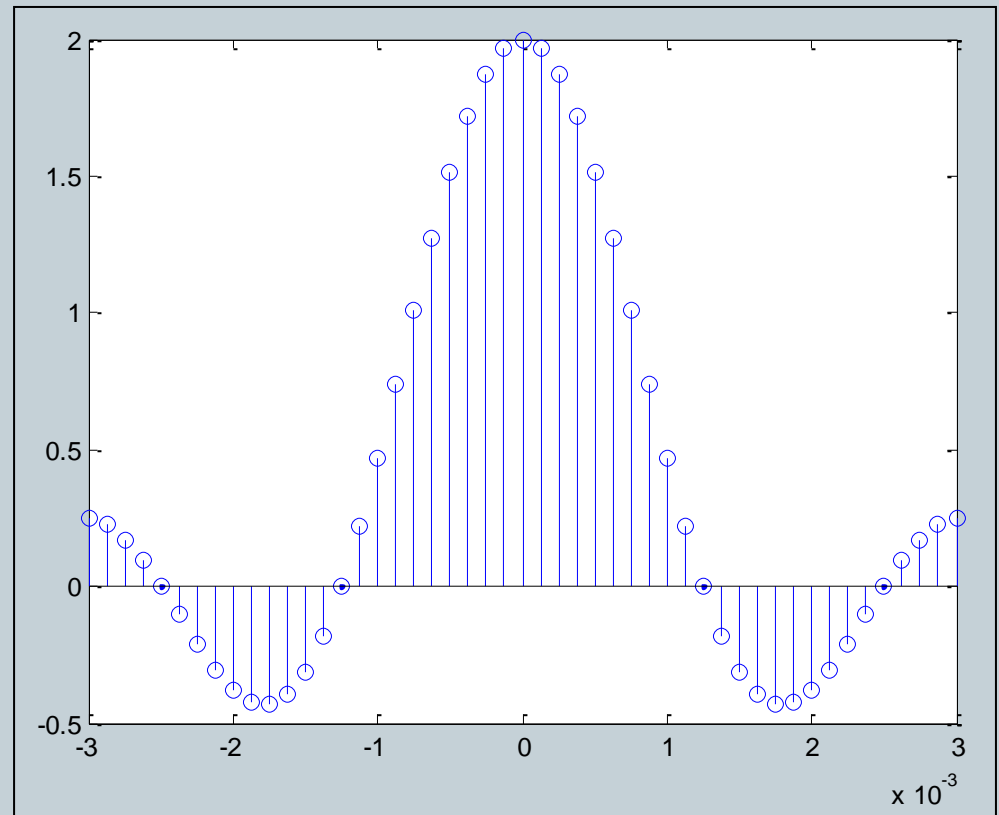


# Ejemplo: Señal aperiódica.

5

## Muestreo de la señal Sinc.

```
>> F0=400;  
>> A=2;  
>> Fs=8000;  
>> Ts=1/Fs;  
>> t=-0.003:Ts:0.003;  
>> xt=A*sinc(2*F0*t);  
>> stem(t,xt)
```



# Generación de Ruido

6

Se utilizan los comandos:

**Y=rand(rows,cols):** Genera ruido con distribución uniforme.

**Y=randn(rows,cols):** Genera ruido Gaussiano.

**Dónde:**

**rows, cols:** Indica la dimensión de la matriz de ruido aleatorio a generar.

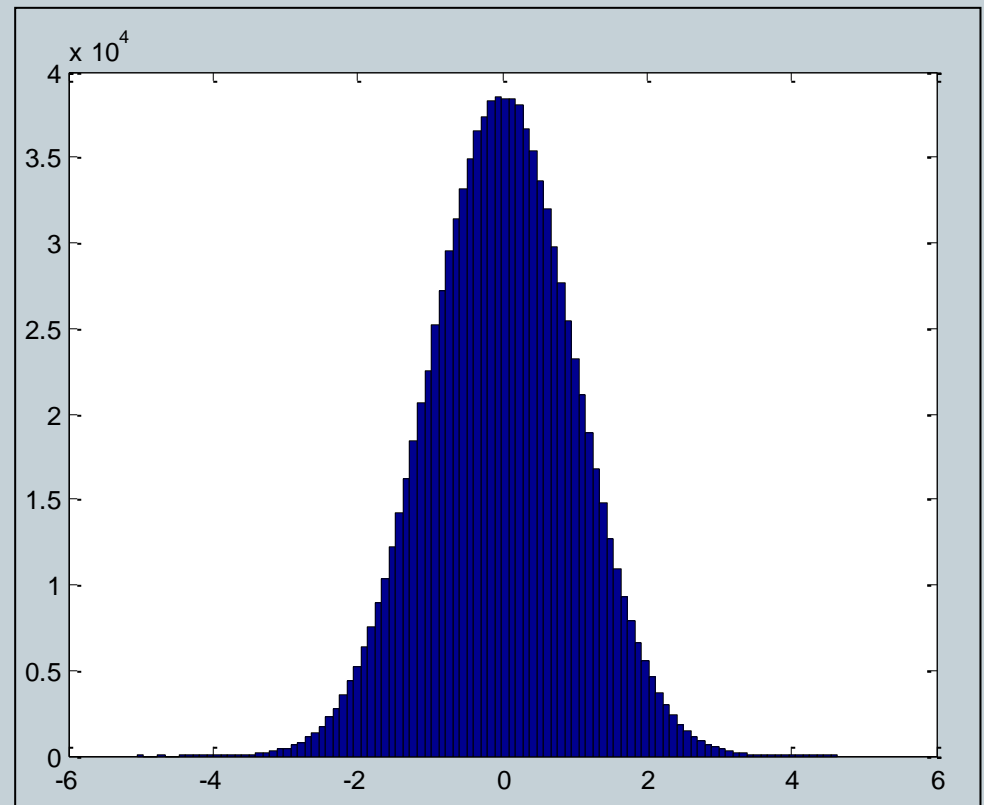
Para observar el histograma se utiliza el comando:

**hist(y,m):** Representa el ruido "y" mediante m "contenedores".

# Ejemplo: Generación y Representación de Ruido Gaussiano

7

```
>> Gnoise=randn(1,1e6);  
>> hist(Gnoise,100);
```



# Submuestreo y Sobremuestreo

8

- **Submuestreo**

$$\mathbf{xDown} = \text{downsample}(\mathbf{x}, N)$$

La señal **xDown** tendrá una frecuencia de muestro  $F_s/N$ .

Submuestrear la señal significa conservar cada  $N$ -ésima muestra y eliminar las muestras restantes.

- **Sobremuestreo**

$$\mathbf{xUp} = \text{upsample}(\mathbf{x}, N)$$

La señal **xUp** tendrá una frecuencia de muestreo  $N \cdot F_s$ .

Sobremuestrear la señal significa introducir  $N-1$  ceros entre muestras consecutivas.

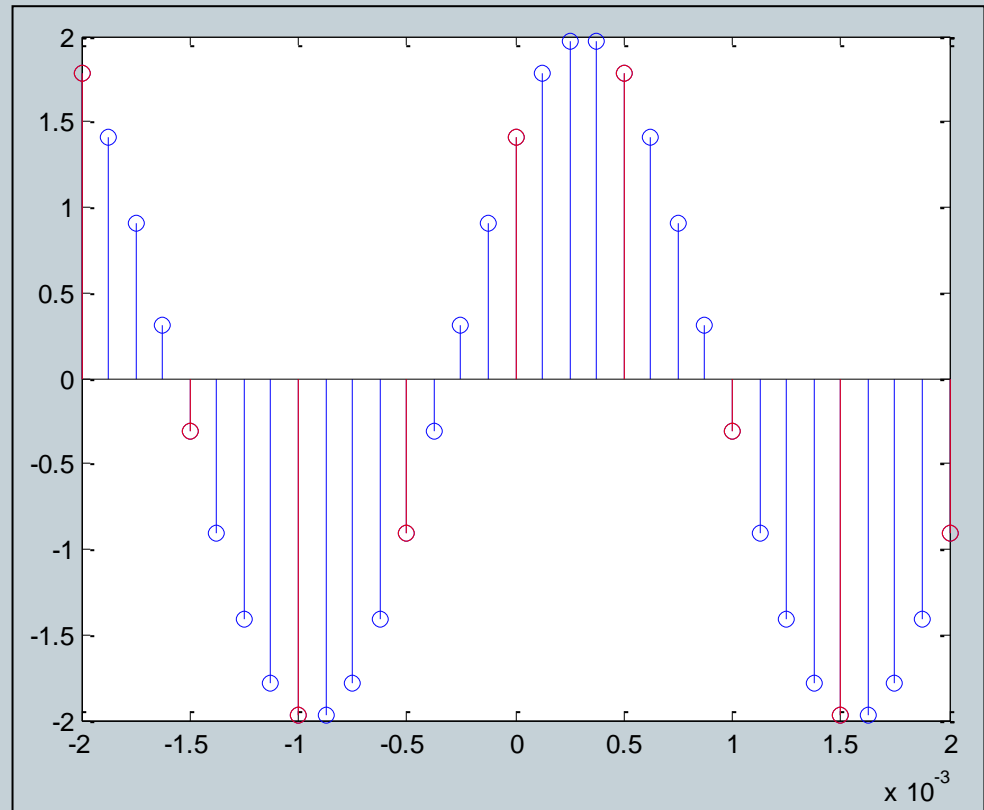


# Ejemplo: Submuestreo.

9

Se submuestra la señal sinusoidal generada anteriormente ( $F_s=8$  KHz) para obtener una señal  $xtDown$  muestreada a 2 KHz.

```
>> F0=400;  
>> A=2;  
>> phi=pi/4;  
>> Fs=8000;  
>> Ts=1/Fs;  
>> t=-0.002:Ts:0.002;  
>> xt=A*sin(2*pi*F0*t+phi);  
>> stem(t,xt)  
>> xtDown=downsample(xt,4);  
>> tDown=downsample(t,4);  
>> hold on  
>> stem(tDown,xtDown,'r');
```



# Procesamiento de Audio

10

Para escuchar un tono de señal en Matlab se utiliza el comando:

**soundsc(xt,Fs)**

**Donde:**

**xt:** Tono a escuchar.

**Fs:** Frecuencia de muestreo.

**Por ejemplo:** Para escuchar la señal sinusoidal generada anteriormente:

```
>> soundsc(xt,Fs)
```

# Grabación de Audio

11

Para grabar una señal audible mediante la tarjeta de sonido y un micrófono se utilizan los siguientes comandos:

**r=audiorecorder:** Crea un objeto de grabación.

**record(r):** Inicio de grabación.

**pause(r) ,stop(r):** Pausa y finalización.

**play(r):** Escuchar la grabación

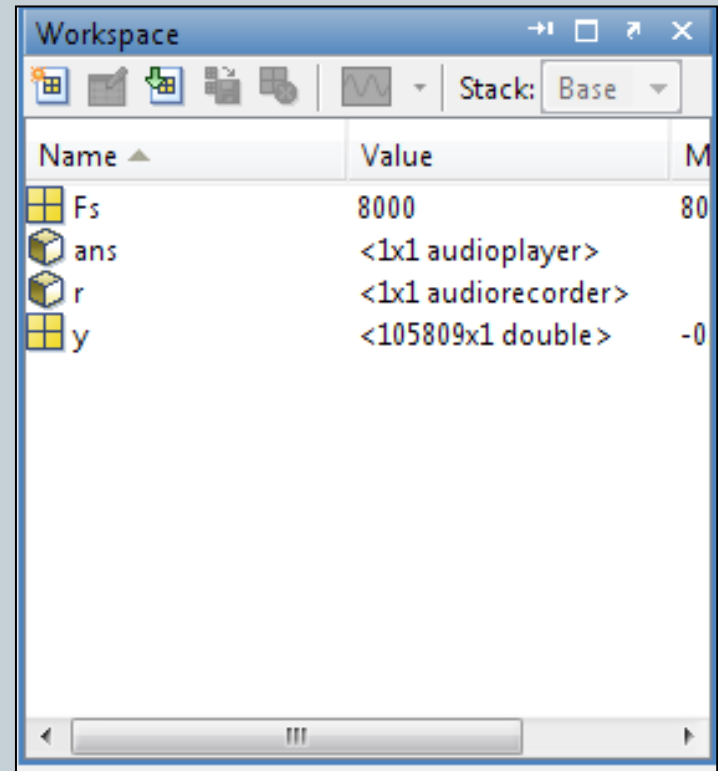
**y=getaudiodata(r):** Para obtener la matriz que contiene las muestras de la señal audible. Ésta es la señal que se puede procesar.

**Fs=r.SampleRate:** Para obtener la frecuencia de muestreo.

# Ejemplo:

12

```
>> r=audiorecorder;  
>> record(r)  
>> pause(r)  
>> record(r)  
>> stop(r)  
>> play(r)  
>> y=getaudiodata(r);  
>> Fs=r.SampleRate  
Fs =  
    8000
```



# Guardar como Archivo de Audio

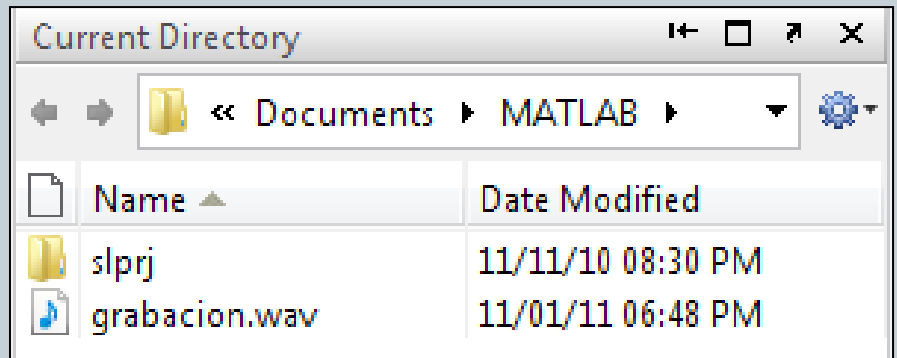
13

Para guardar la señal de audio que se acaba de grabar se utiliza el comando:

**wavwrite(y,Fs,'grabacion')**

Se guarda en formato wav en la carpeta de trabajo de Matlab.

```
>> wavwrite(y,Fs,'grabacion');  
>> which grabacion.wav  
C:\Users\Casa\Documents\MATLAB\  
grabacion.wav
```



# Leer un Archivo de Audio

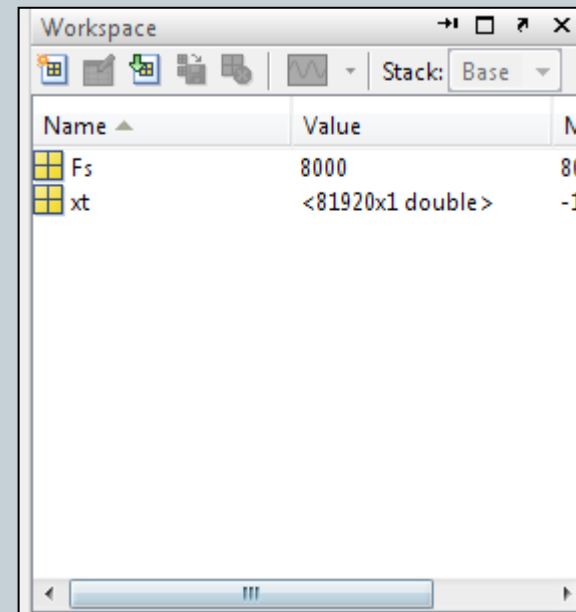
14

Para leer un archivo de audio en formato wav que se encuentra en la carpeta de trabajo de Matlab se utiliza la función:

**[xt,Fs]=wavread('nombre\_de\_archivo')**

Se guardan las muestras y la frecuencia de muestro en **xt** y **Fs** respectivamente.

```
>> [xt,Fs]=wavread('tuner1');
```



# Análisis en Frecuencia

15

## Transformada Rápida de Fourier:

$$\mathbf{xF} = \text{fft}(\mathbf{xt}, \mathbf{nFFT})$$

**Donde:**

**nFFT**=número de puntos a representar.

## Comandos adicionales:

**magxF=abs(xF):** Magnitud de la fft de xt.

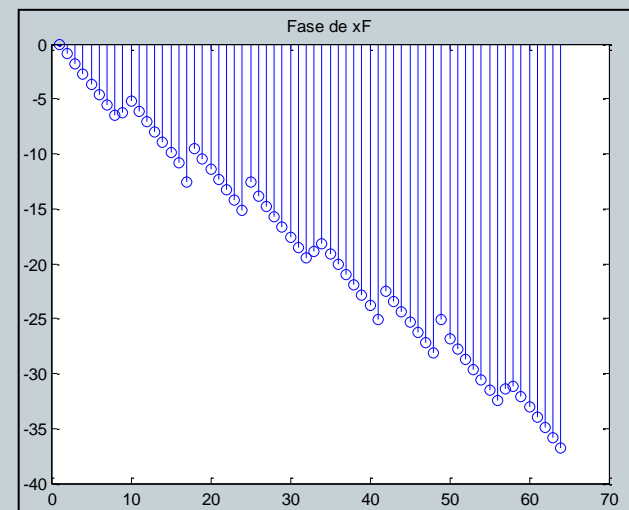
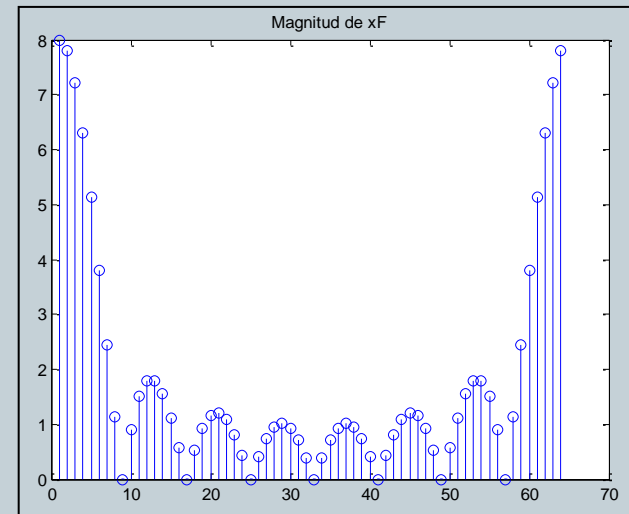
**phasexF=angle(xF):** Fase de la fft de xt.

**phasexF=unwrap(phasexF):** Para obtener la apropiada información de fase.

# Ejemplo: Obtención de La FFT de la Función Compuerta.

16

```
>> Fs=2000;  
>> Ts=1/Fs;  
>> t=-0.005:Ts:0.005;  
>> xt=rectpuls(t,0.004);  
>> nFFT=64;  
>> xF=fft(xt,nFFT);  
>> magxF=abs(xF);  
>> phasexF=angle(xF);  
>> phasexF=unwrap(phasexF);  
>> figure,stem(magxF);  
>> title('Magnitud de xF')  
>> figure,stem(phasexF)  
>> title('Fase de xF')
```



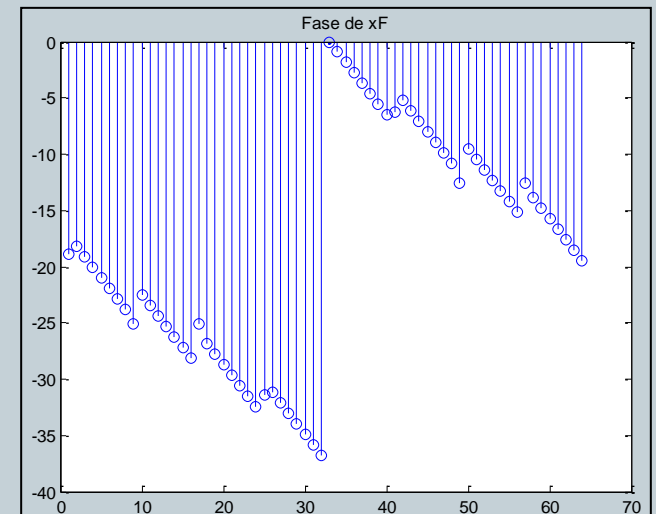
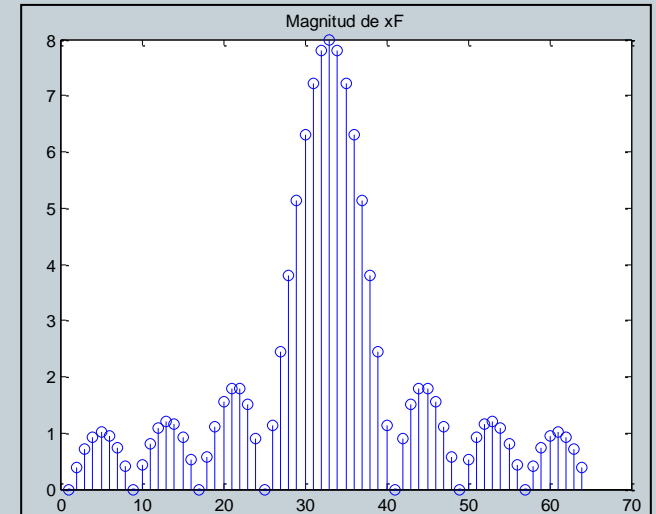


# Ejemplo: Obtención de La FFT de la Función Compuerta.

17

El comando **fft** obtiene la transformada de Fourier en la región comprendida entre 0 y la frecuencia de muestro  $F_s$ . Se requiere desplazar una parte de la gráfica para obtener la región negativa de la transformada de Fourier.

```
>> magxF=fftshift(magxF);  
>> phasexF=fftshift(phasexF);  
>> stem(magxF);  
>> title('Magnitud de xF')  
>> figure,stem(phasexF);  
>> title('Fase de xF')
```

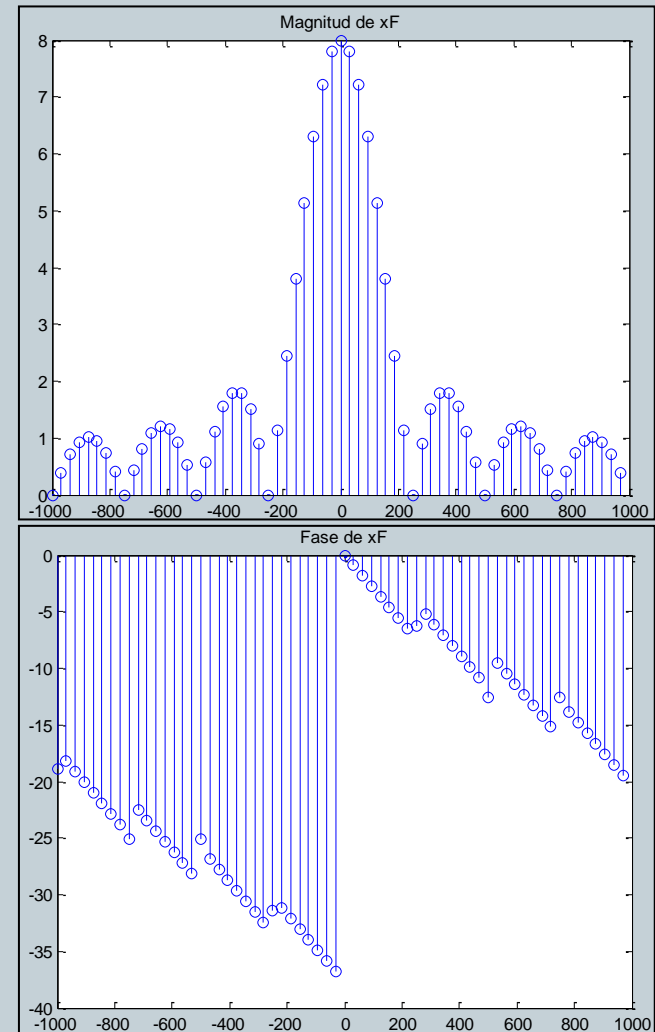


# Ejemplo: Obtención de La FFT de la Función Compuerta.

18

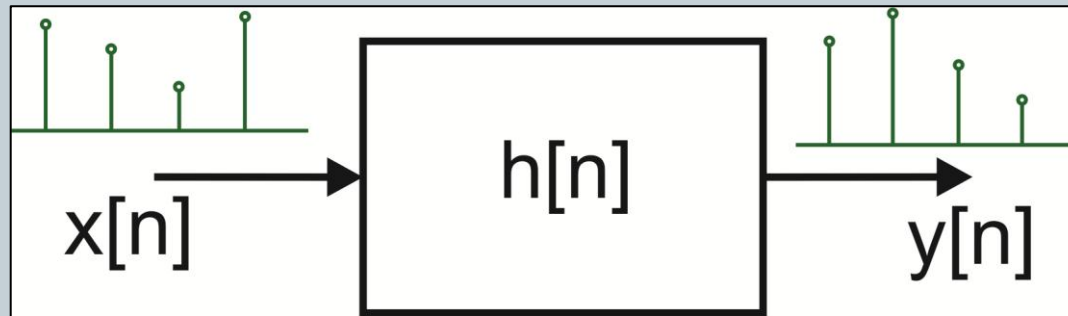
Los gráficos anteriores no presentan el eje de frecuencia correcto. Se debe crear el vector de eje de frecuencias para graficar correctamente.

```
>> f_esp=Fs/nFFT;  
>> fNyquist=Fs/2;  
>> f_inicio=-fNyquist;  
>> f_fin=fNyquist-f_esp;  
>> f_eje=f_inicio:f_esp:f_fin;  
>> stem(f_eje,magxF)  
>> figure,stem(f_eje,phasexF)
```

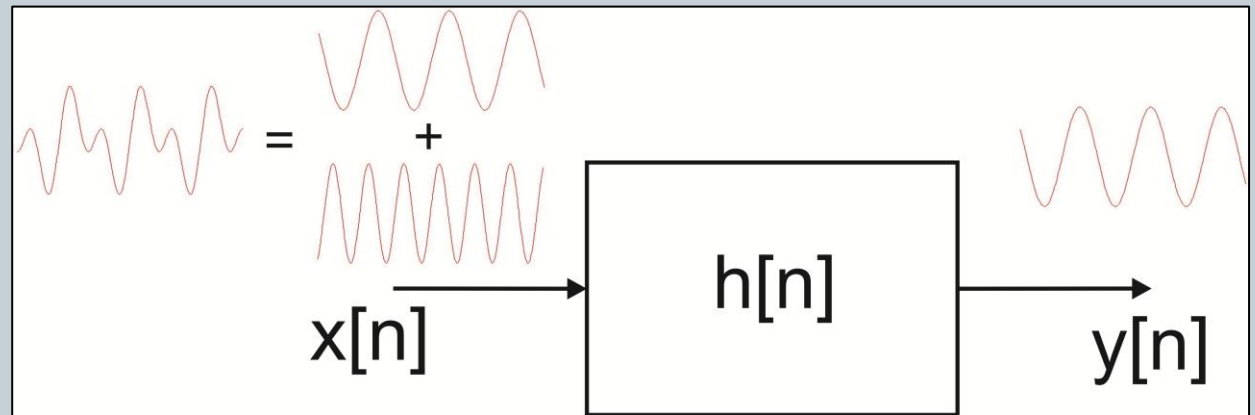


# Sistemas LTI

19



Todos los sistemas LTI pueden ser analizados como Filtros LTI discretos.



# Ecuación de Diferencias Para Filtros Causales

20

**Dónde:**  $x[n]$ : Entrada  
 $y[n]$ : Salida

$$\sum_{k=0}^N a_k \cdot y[n-k] = \sum_{k=0}^M b_k \cdot x[n-k]$$

En el dominio Z

$$\sum_{k=0}^N a_k \cdot z^{-k} \cdot Y(z) = \sum_{k=0}^M b_k \cdot z^{-k} X(z)$$

Función de transferencia  
en el dominio Z

$$Y(z) = H(z) \cdot X(z)$$

$$H(z) = \frac{Y(z)}{X(z)}$$

$$H(z) = \frac{\sum_{k=0}^M b_k \cdot z^{-k}}{\sum_{k=0}^N a_k \cdot z^{-k}}$$

# Representación de Filtros en Matlab

21

Para representar la función de transferencia del filtro:

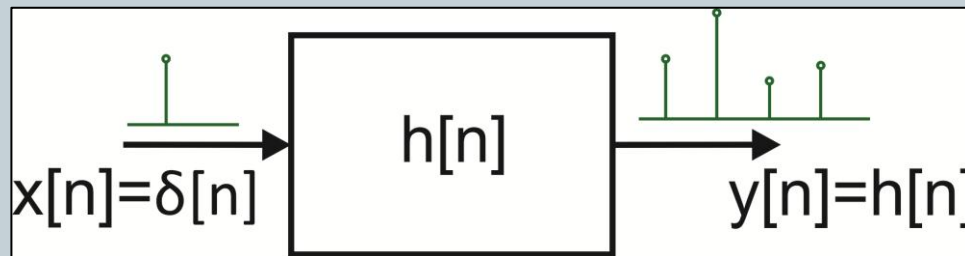
$$H(z) = \frac{1 + 1.1 \cdot z^{-1}}{1 - 0.1 \cdot z^{-2}}$$

en Matlab:

```
>> num=[1 1.1];  
>> den=[1 0 -0.1];
```

# Respuesta al Impulso y Convolución

22



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

Para obtener la respuesta al impulso se utilizan los comandos:

**Impz(num,den):** Grafica la respuesta al impulso del filtro. Por defecto se considera  $F_s=1$  Hz y grafica las 10 primeras muestras.

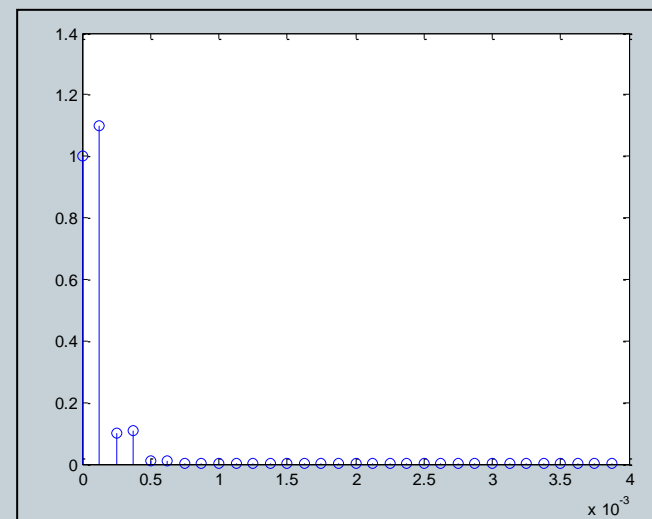
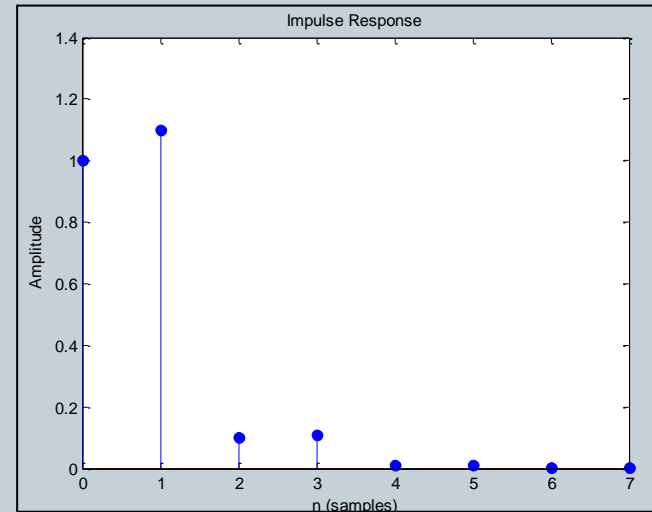
**Impz(num,den,n,Fs):** Grafica la respuesta al impulso. Se especifica el número de muestras a graficar y la frecuencia de muestro utilizada.

**[h,t]= Impz(num,den,n,Fs):** Guarda los vectores de magnitud y eje de tiempo en **h** y **t** respectivamente.

# Ejemplo: Graficando la respuesta al impulso del filtro creado anteriormente.

23

```
>> num=[1 1.1];  
>> den=[1 0 -0.1];  
>> impz(num,den);  
>> Fs=8000;  
>> [h,t]=impz(num,den,32,Fs);  
>> figure,stem(t,h)
```



# Señal de Salida del Filtro Para una Señal de Entrada

24

Se realiza mediante los comandos:

$$\mathbf{y=conv(h,x)}$$

**Donde:** **h** es la respuesta al impulso del filtro.  
**x** es la señal de entrada.

$$\mathbf{y=filter(num,den,x)}$$

**Donde:** **x** es la señal de entrada.

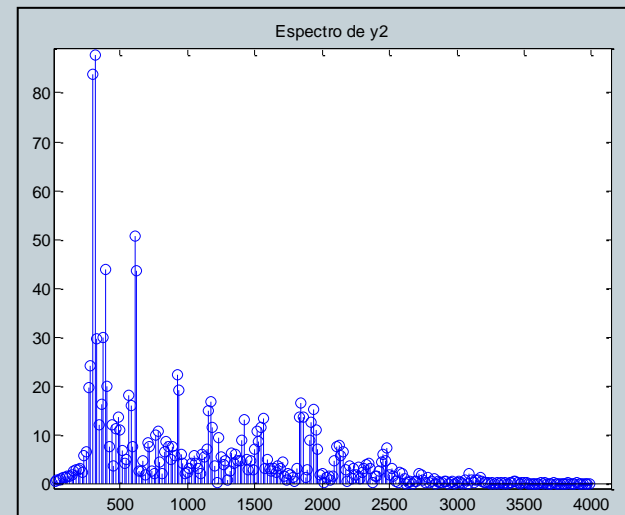
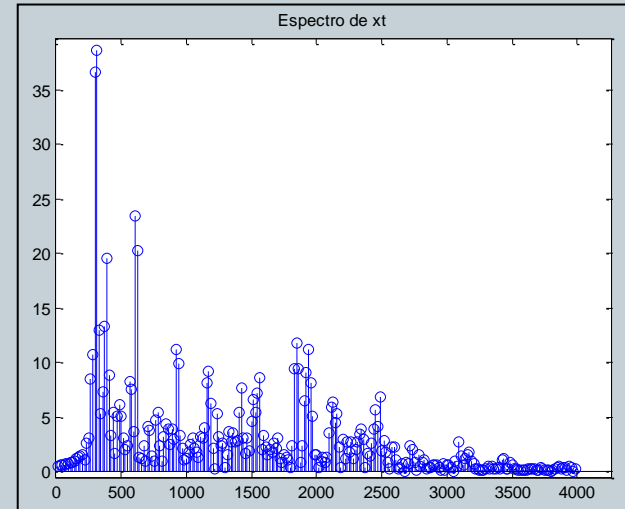


# Ejemplo:

25

Se obtendrá la señal de salida del filtro para una señal de entrada audible:

```
>> [xt,Fs]=wavread('tuner1');  
>> [h,t]=impz(num,den,10,Fs);  
>> y1=convn(h,xt);  
>> soundsc(xt,Fs);  
>> soundsc(y1,Fs);  
>> y2=filter(num,den,xt);  
>> soundsc(xt,Fs);  
>> soundsc(y2,Fs);
```



# Análisis de Filtros en el Dominio de la Frecuencia

26

Relación Entrada – Salida de un filtro en el dominio de la frecuencia.

$$Y(\omega) = X(\omega) \cdot H(\omega)$$

Respuesta de magnitud:

$$|H(\omega)|$$

Respuesta de fase:

$$\angle H(\omega)$$

Retardo de grupo:

$$-\frac{d}{d\omega} \angle H(\omega)$$

# Respuesta de Magnitud y Fase de Filtros

27

Se utilizan los comandos:

**[H,F]=freqz(num,den):** Por defecto asume 512 puntos equidistantemente espaciados y el eje de frecuencias es normalizada.

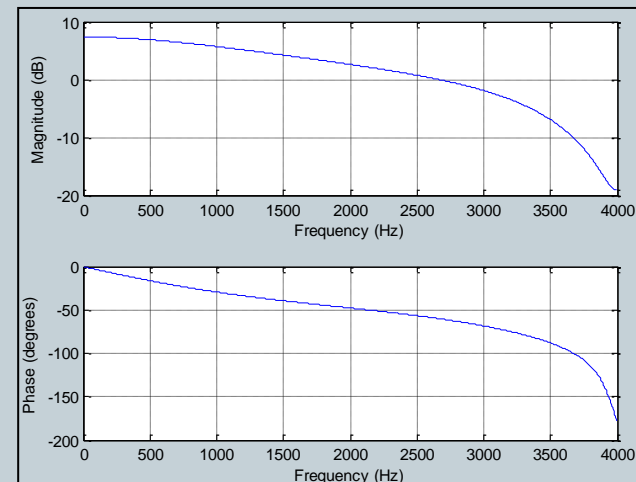
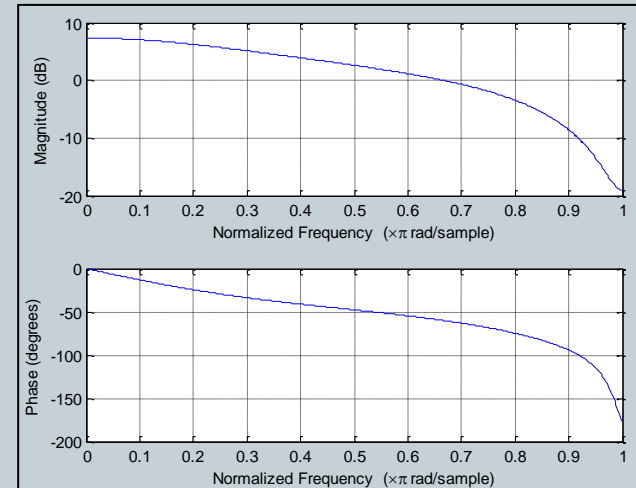
**[H,F]=freqz(num,den,n,Fs):** Donde se especifica número de puntos  $n$  y la frecuencia de muestreo  $F_s$ .

**[H,F]=freqz(num,den,f,Fs):** Donde  $f$  es un vector en el que se especifica las frecuencias particulares en las que se obtendrá la respuesta de magnitud y fase.

# Ejemplo: Respuesta en frecuencia del filtro creado anteriormente.

28

```
>> num=[1 1.1];  
>> den=[1 0 -0.1];  
>> Fs=8000;  
>> freqz(num,den)  
>> figure,freqz(num,den,1024,Fs);  
>> f=[0,200,400];  
>> [H,F]=freqz(num,den,f,Fs);
```



# Gráfico de Retardo de Grupo

29

Se utilizan los comandos:

**$[G,F]=\text{grpdelay}(\text{num},\text{den})$**

**$[G,F]=\text{grpdelay}(\text{num},\text{den},n,F_s)$**

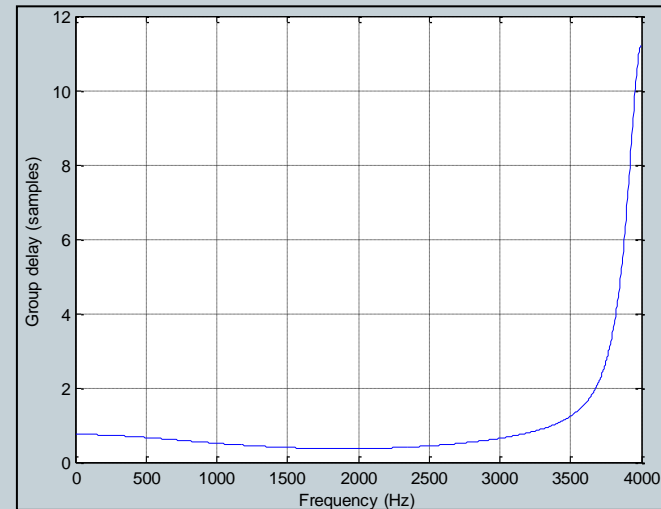
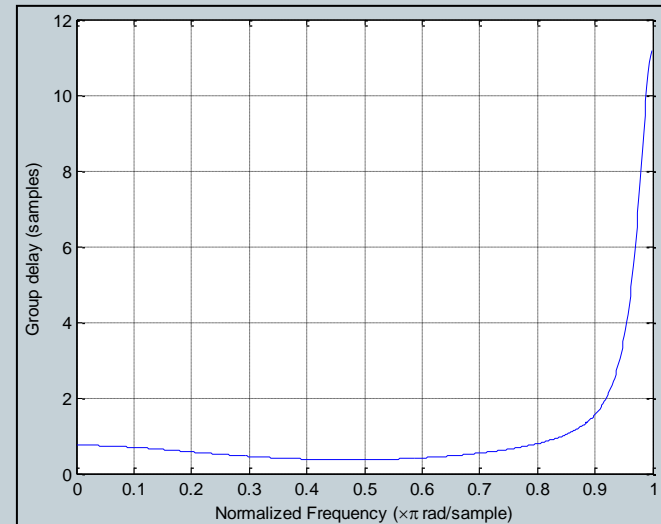
**$[G,F]=\text{grpdelay}(\text{num},\text{den},f,F_s)$**

Los argumentos son iguales a los del caso anterior.

# Ejemplo: Continuando con el ejemplo anterior

30

```
>> [G,F]=grpdelay(num,den);  
>> grpdelay(num,den);  
>> figure,grpdelay(num,den,1024,Fs);
```



# Análisis de Polos y Ceros de Funciones de Transferencia

31

Para estudiar la estabilidad de los sistemas LTI.

Función de transferencia:

$$H(z) = \frac{\sum_{k=0}^M b_k \cdot z^{-k}}{\sum_{k=0}^N a_k \cdot z^{-k}}$$

Ganancia polo – cero:

$$H(z) = K \cdot \frac{\prod_{k=1}^M (z - z_k)}{\prod_{k=1}^N (z - p_k)}$$

# Obtención de Polos y Ceros

32

Se realiza una transformación del dominio **tf** al dominio **zp**.

$$[z,p,k]=\text{tf2zp}(\text{num},\text{den})$$

Dónde los vectores:

**z:** ceros

**p:** polos

**k:** ganancia de cada numerador de tf.

De manera inversa:

$$[\text{num},\text{den}]=\text{zp2tf}(z,p,k)$$



# Ejemplo: Con el filtro implementado anteriormente.

33

```
>> num=[1 1.1];  
>> den=[1 0 -0.1];  
>> [z,p,k]=tf2zp(num,den)  
z =  
    -1.1000  
p =  
    0.3162  
   -0.3162  
k =  
     1  
>> [num,den]=zp2tf(z,p,k)  
num =  
     0    1.0000    1.1000  
den =  
    1.0000     0   -0.1000
```

# Gráfico de Polos y Ceros

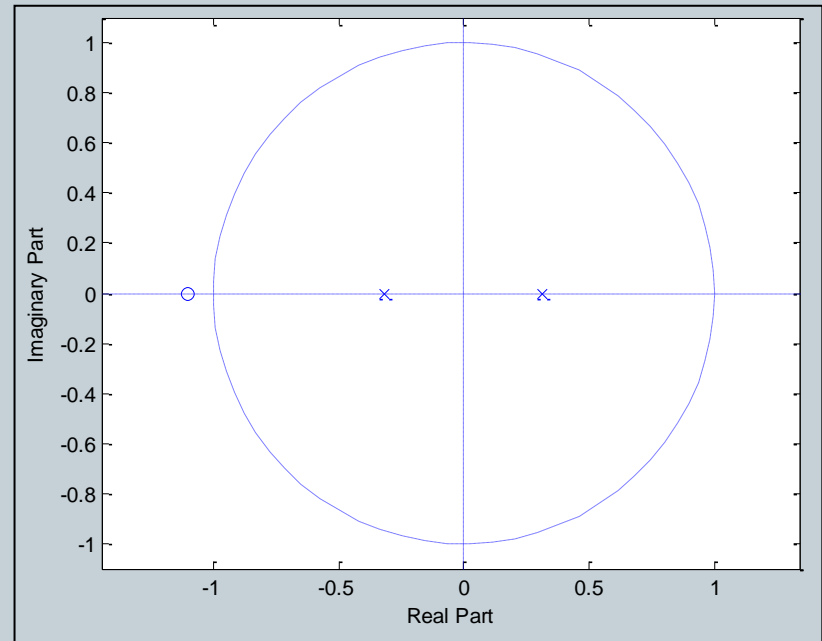
34

Se utiliza el comando:

**zplane(z,p)**

Si hay polos fuera del círculo unitario significa que el sistema es inestable.  
Continuando con el ejemplo anterior:

```
>> zplane(z,p)
```

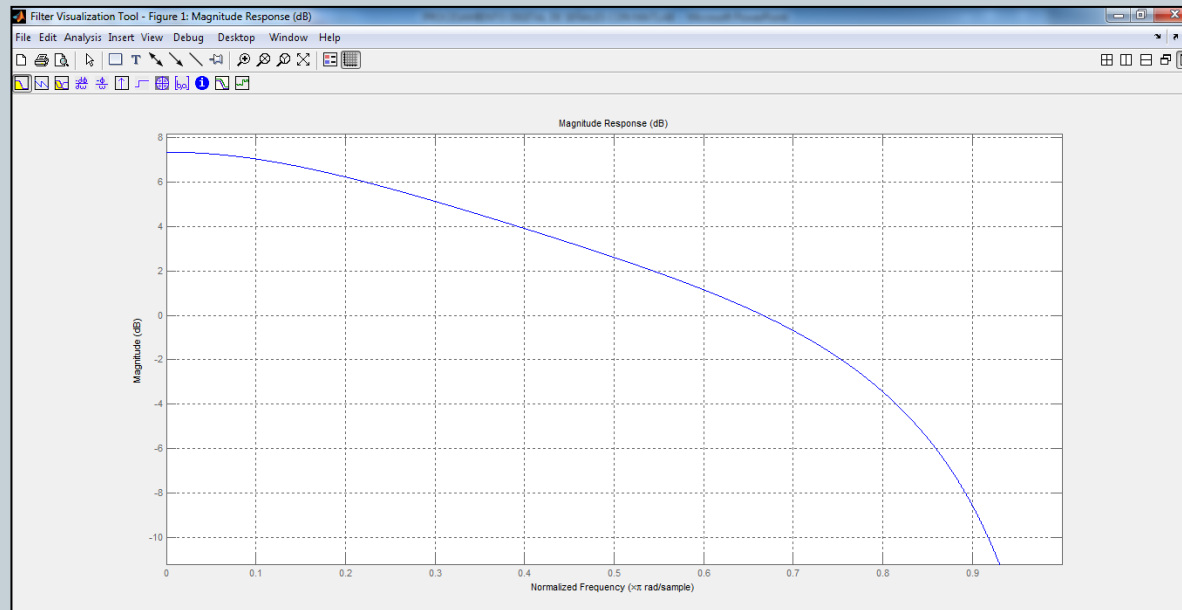


# Uso del FVTool

35

**FVTool** (Filter Visualization Tool) Es una herramienta que simplifica el trabajo de estudiar el comportamiento de los filtros.

Se trata de una interfaz GUI para el análisis de filtros.



# Llamada al FVtool

36

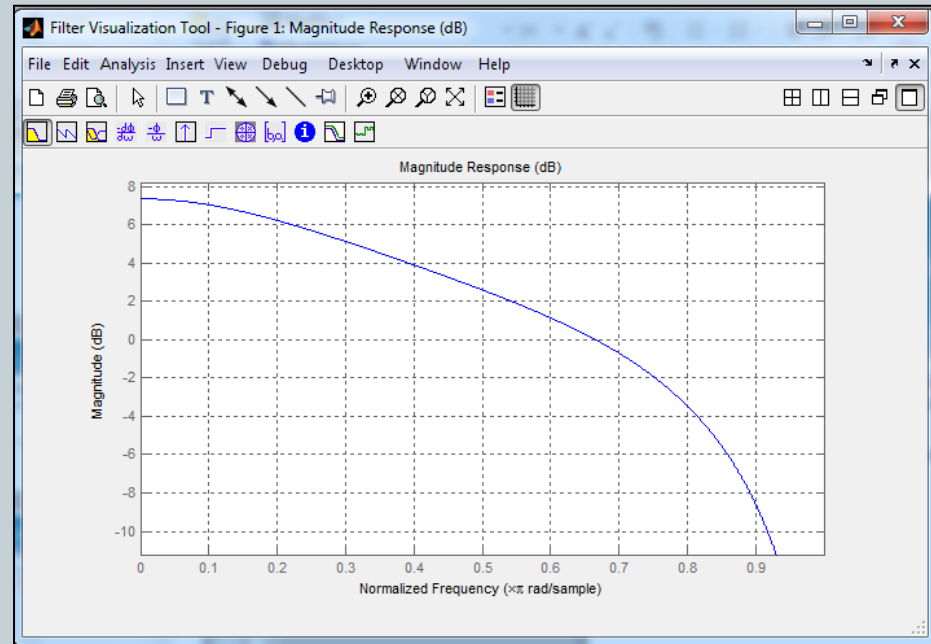
Se llama a la herramienta con el comando:

**fvtool(num,den)**

**Donde:** **num** y **den** corresponden al filtro en estudio.

Continuando con el ejemplo anterior:

>> fvtool(num,den)

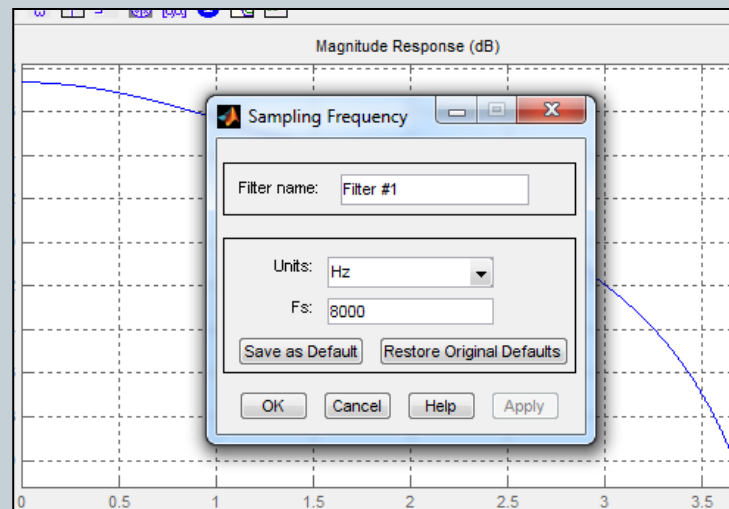


# Configuración de FVtool

37

El eje de frecuencias por defecto no se corresponde con los valores correctos. Para configurarlo correctamente:

Menú **Analysis>Sampling Frequency**. En el cuadro **F<sub>s</sub>** se introduce el valor de la frecuencia de muestreo.



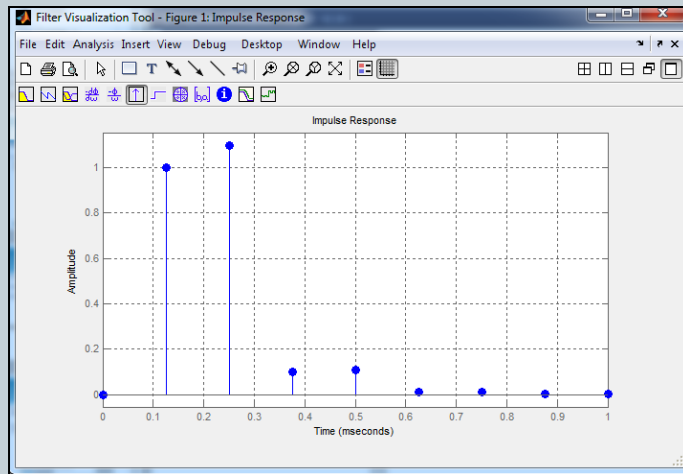
# La herramienta Fvtool permite visualizar diferentes respuestas del filtro en estudio

38

- Respuesta de Magnitud
- Respuesta de fase
- Respuestas de magnitud y fase
- Respuesta de retardo de grupo
- Respuesta de retardo de fase
- Respuesta al impulso
- Respuesta al escalón
- Gráfico de polos y ceros
- Etc

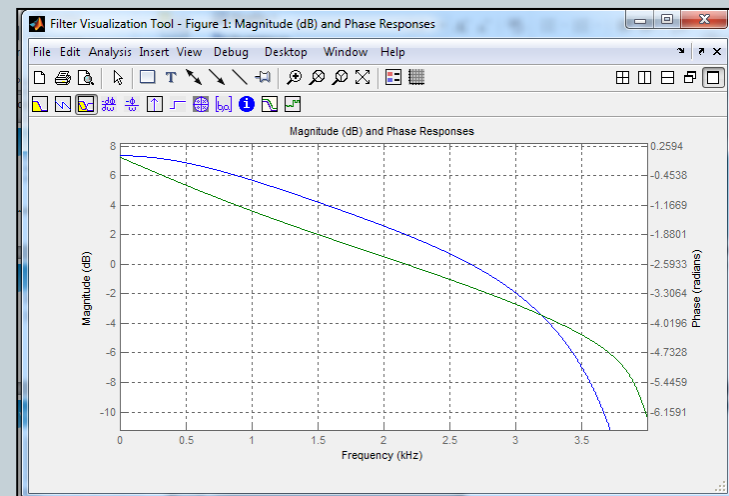
# Ejemplo:

39



Respuesta al impulso

Respuestas de Magnitud y fase



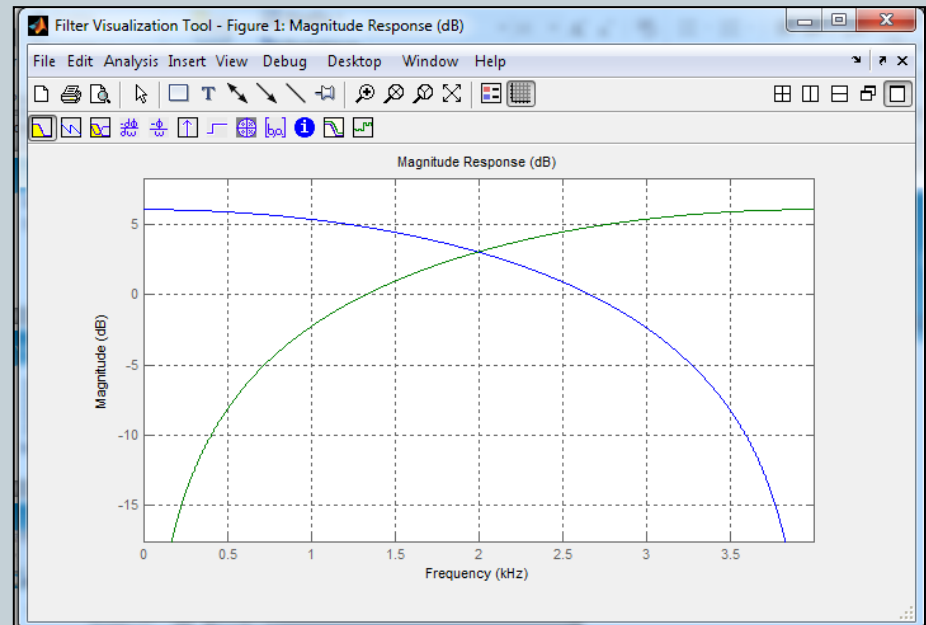
# Visualización de Múltiples Filtros

40

Es posible visualizar la respuesta de varios filtros en simultáneo.

## Ejemplo:

```
>> num1=[1 1];  
>> den1=[1];  
>> num2=[1 -1];  
>> den2=[1];  
>> fvtool(num1,den1,num2,den2)
```

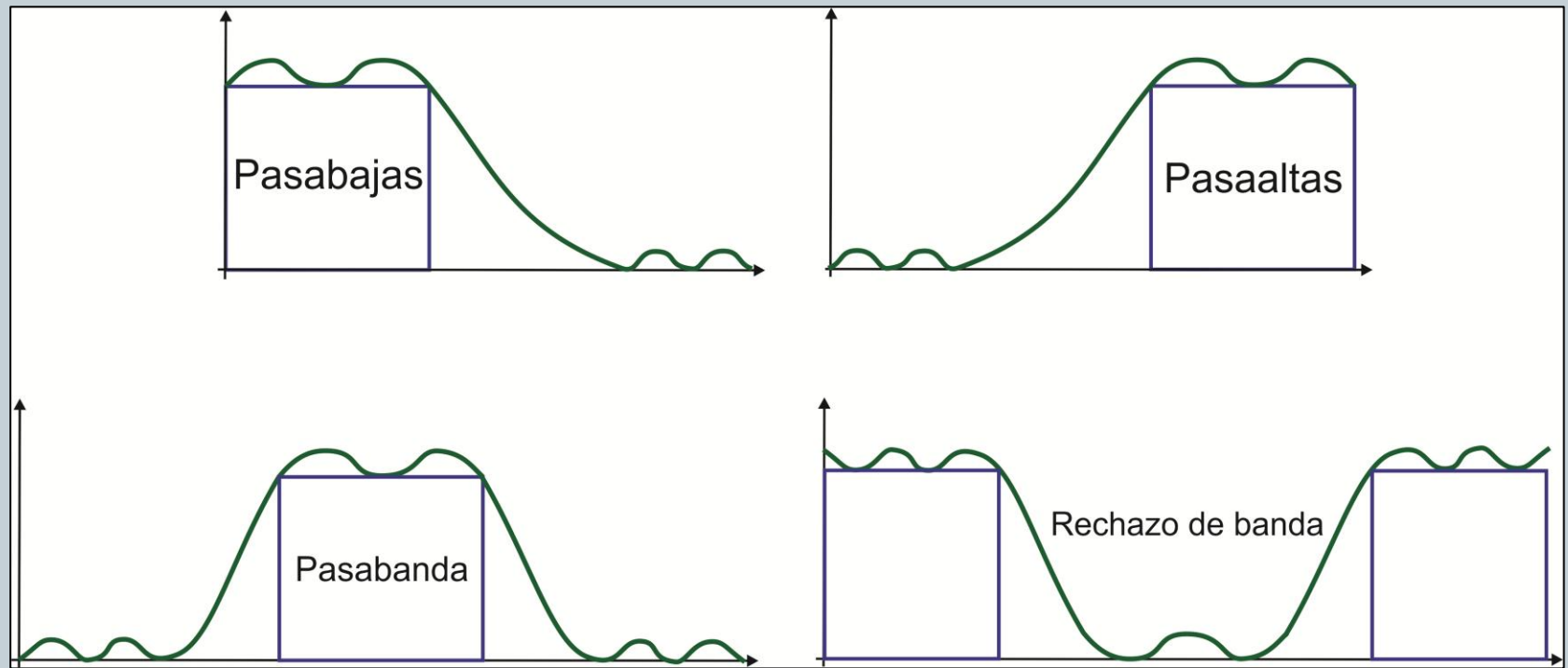




# Diseño de Filtros Digitales en Matlab

41

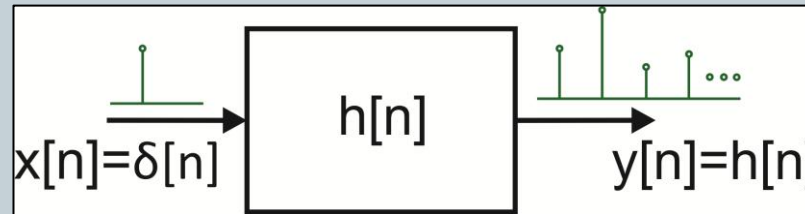
Los filtros elementales son los siguientes:



# Diseño de Filtros IIR

42

**IIR:** Infinite Impulse Response.



Función de transferencia:

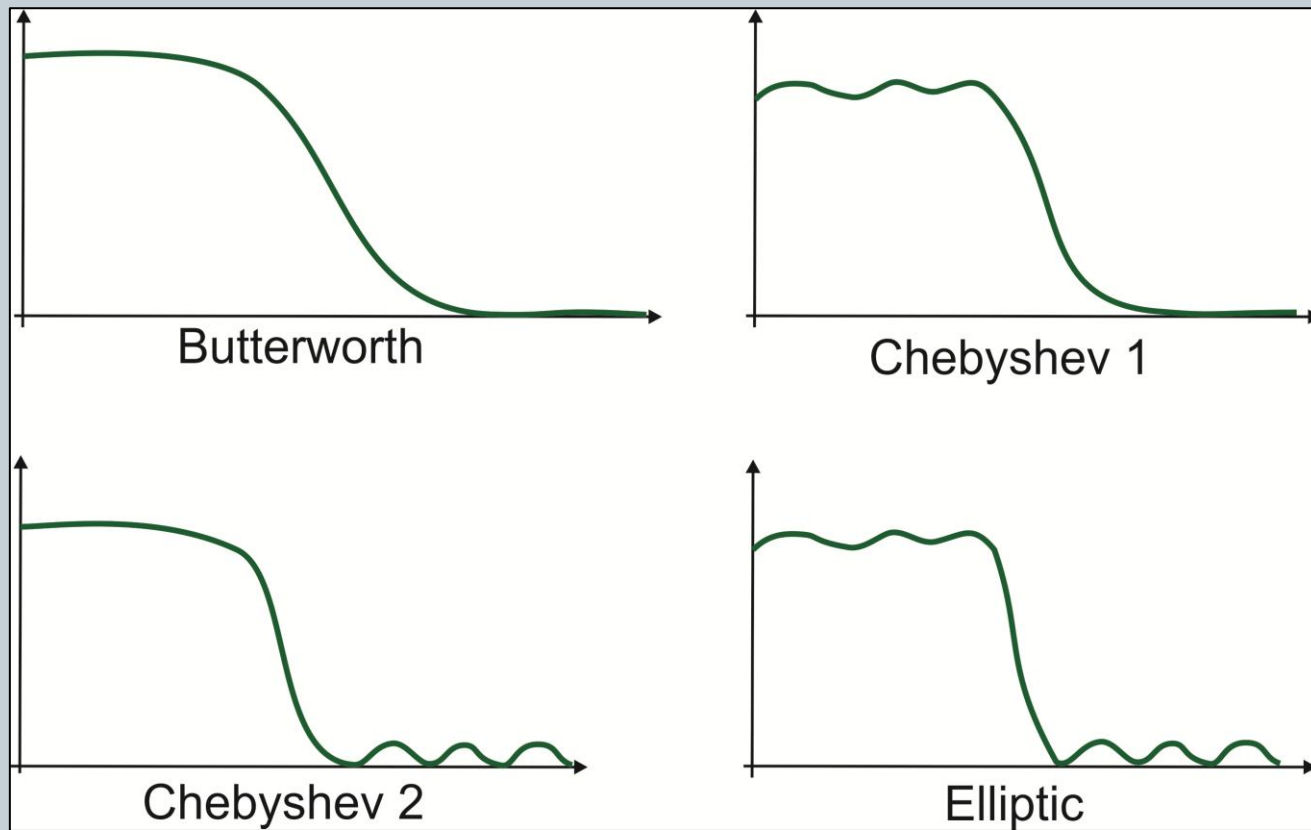
$$H(z) = \frac{\sum_{k=0}^M b_k \cdot z^{-k}}{\sum_{k=0}^N a_k \cdot z^{-k}}$$

Respuesta en frecuencia:

$$H(\omega) = \frac{\sum_{k=0}^M b_k \cdot e^{-j \cdot k \cdot \omega}}{\sum_{k=0}^N a_k \cdot e^{-j \cdot k \cdot \omega}}$$

# Tipos y Características de Filtros IIR

43

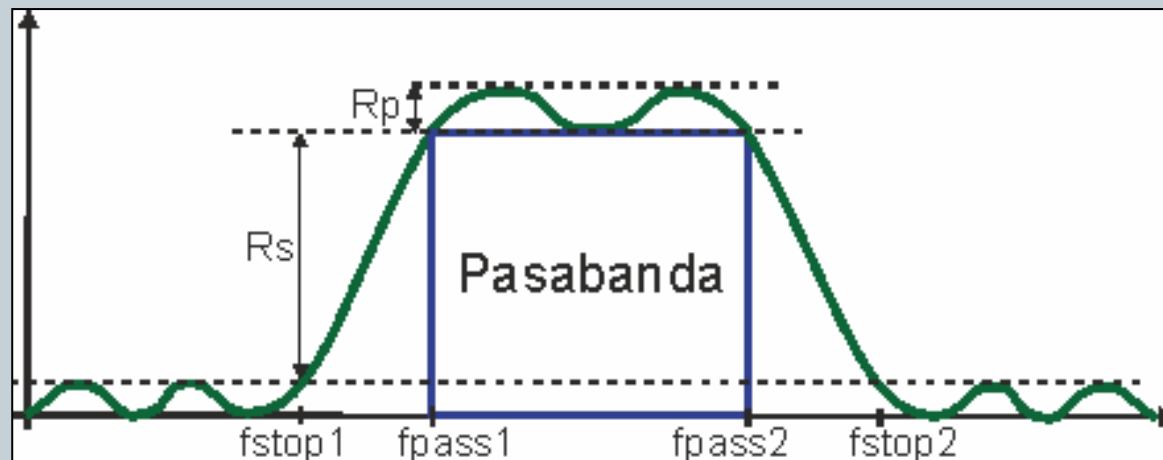


# Especificaciones Para un Filtro Pasabanda

44

Para el diseño de un filtro pasabanda se especifica:

- **fstop1**: Frecuencia stop 1.
- **fpass1**: Frecuencia pass 1.
- **fpass2**: Frecuencia pass 2.
- **fstop2**: Frecuencia stop 2.
- **Rp**: Rizado en la banda de paso en dB.
- **Rs**: Atenuación en la banda de supresión en dB.

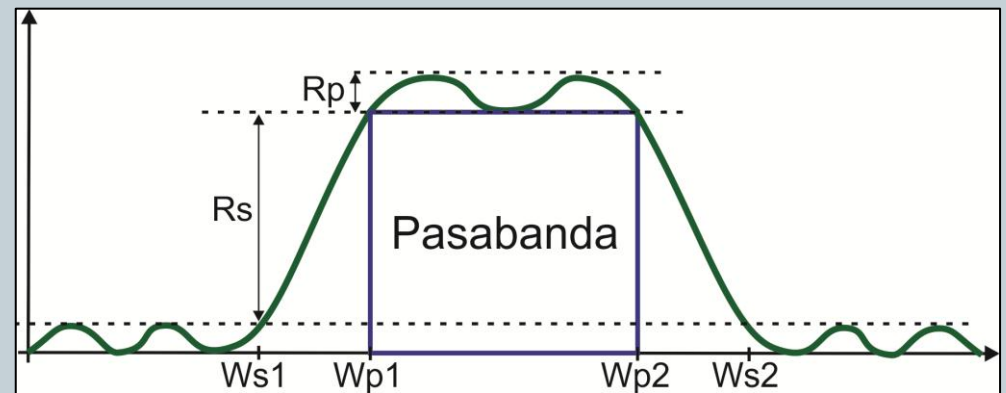


# Diseño de un Filtro Pasabanda

45

El diseño de filtros en Matlab se realiza con valores normalizados. Para esto se realiza:

- Especificar  $F_s$ .
- $F_{nyquist} = F_s/2$
- $W_s(1) = f_{stop1}/F_{nyquist}$
- $W_p(1) = f_{pass1}/F_{nyquist}$
- $W_p(2) = f_{pass2}/F_{nyquist}$
- $W_s(2) = f_{stop2}/F_{nyquist}$



Dónde **Ws** y **Wp** son vectores de frecuencias normalizadas.

# Diseño de un Filtro Butterworth

46

1. Determinar el orden del filtro para las especificaciones:

$$[nB, WnB] = \text{buttord}(Wp, Ws, Rp, Rs)$$

**Wp, Ws, Rp y Rs** corresponden a las especificaciones.

**WnB**: Frecuencia a -3 dB.

2. Obtener los coeficientes del filtro.

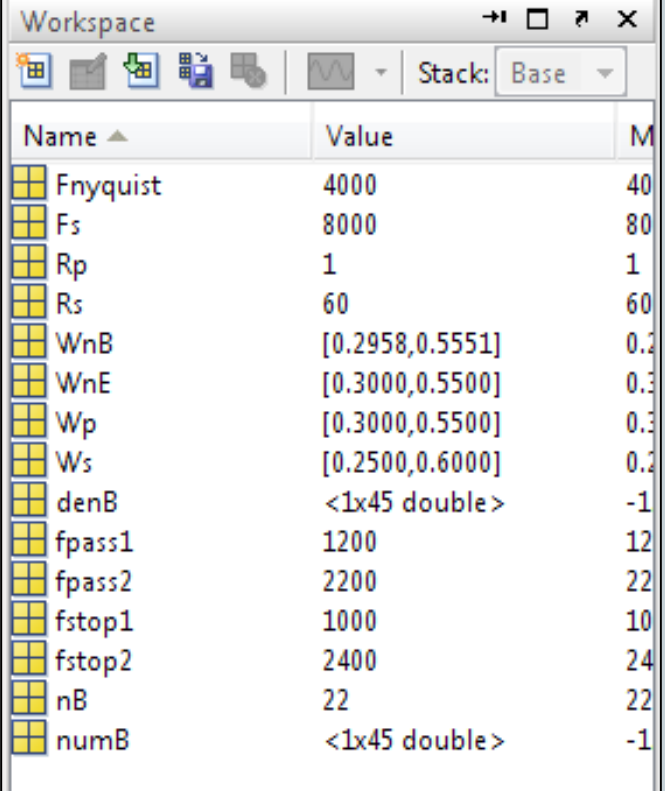
$$[\text{numB}, \text{denB}] = \text{butter}[nB, WnB]$$

En el diseño de filtros Pasabanda o rechazo de banda el orden del filtro resulta ser en realidad el doble del especificado por el comando **buttord**.

# Ejemplo: Diseño de un filtro Butterworth

47

```
>> fstop1=1000;
>> fpass1=1200;
>> fpass2=2200;
>> fstop2=2400;
>> Rp=1;
>> Rs=60;
>> Fs=8000;
>> Fnyquist=Fs/2;
>> Ws(1)=fstop1/Fnyquist;
>> Wp(1)=fpass1/Fnyquist;
>> Wp(2)=fpass2/Fnyquist;
>> Ws(2)=fstop2/Fnyquist;
>> [nB,WnB]=buttord(Wp,Ws,Rp,Rs)
nB =
    22
WnB =
    0.2958    0.5551
>> [numB,denB]=butter(nB,WnB);
```



The image shows a screenshot of the MATLAB Workspace window. It displays a list of variables and their values. The variables are: Fnyquist (4000), Fs (8000), Rp (1), Rs (60), WnB ([0.2958, 0.5551]), WnE ([0.3000, 0.5500]), Wp ([0.3000, 0.5500]), Ws ([0.2500, 0.6000]), denB (<1x45 double>), fpass1 (1200), fpass2 (2200), fstop1 (1000), fstop2 (2400), nB (22), and numB (<1x45 double>). The window also shows a toolbar with icons for workspace, command window, and other MATLAB functions.

Name	Value	M
Fnyquist	4000	40
Fs	8000	80
Rp	1	1
Rs	60	60
WnB	[0.2958, 0.5551]	0.2
WnE	[0.3000, 0.5500]	0.3
Wp	[0.3000, 0.5500]	0.3
Ws	[0.2500, 0.6000]	0.2
denB	<1x45 double>	-1
fpass1	1200	12
fpass2	2200	22
fstop1	1000	10
fstop2	2400	24
nB	22	22
numB	<1x45 double>	-1

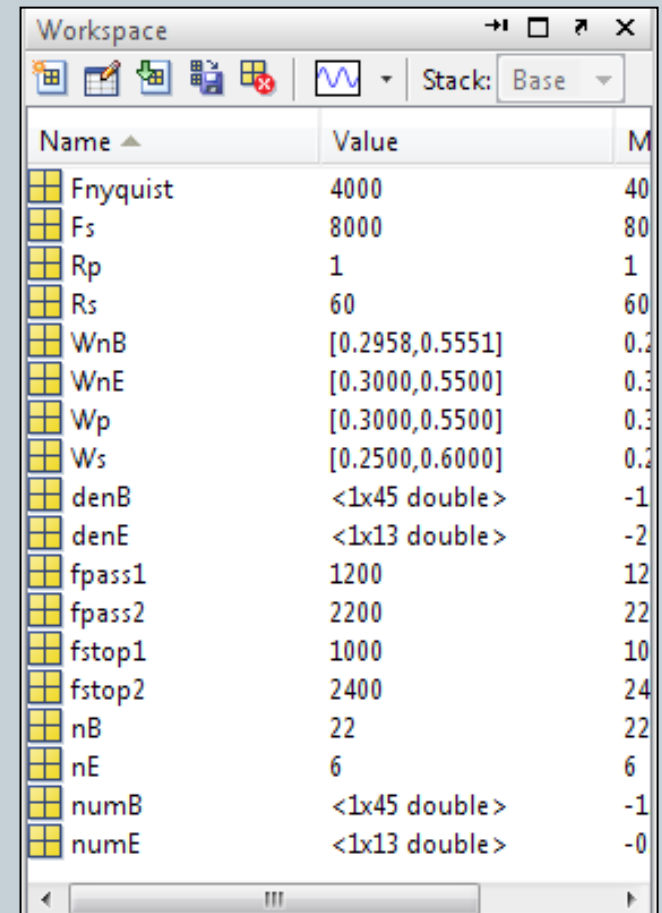
# Diseño de un Filtro Elíptico

48

Se realiza un procedimiento similar al caso anterior.

```
>> [nE,WnE]=ellipord(Wp,Ws,Rp,Rs)
nE =
    6
WnE =
    0.3000    0.5500
>> [numE,denE]=ellip(nE,Rp,Rs,Wp);
```

Se puede observar que la cantidad de coeficientes es mucho menor que en el caso de un filtro Butterworth.



Name	Value	Memory
Fnyquist	4000	40
Fs	8000	80
Rp	1	1
Rs	60	60
WnB	[0.2958,0.5551]	0.2
WnE	[0.3000,0.5500]	0.3
Wp	[0.3000,0.5500]	0.3
Ws	[0.2500,0.6000]	0.2
denB	<1x45 double>	-1
denE	<1x13 double>	-2
fpass1	1200	12
fpass2	2200	22
fstop1	1000	10
fstop2	2400	24
nB	22	22
nE	6	6
numB	<1x45 double>	-1
numE	<1x13 double>	-0



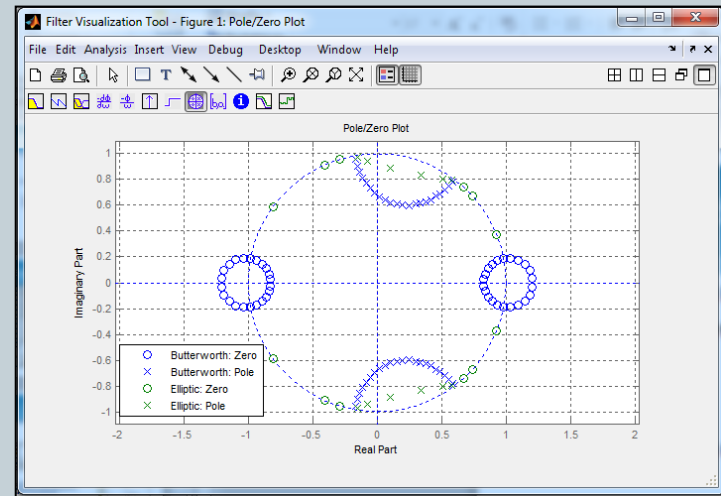
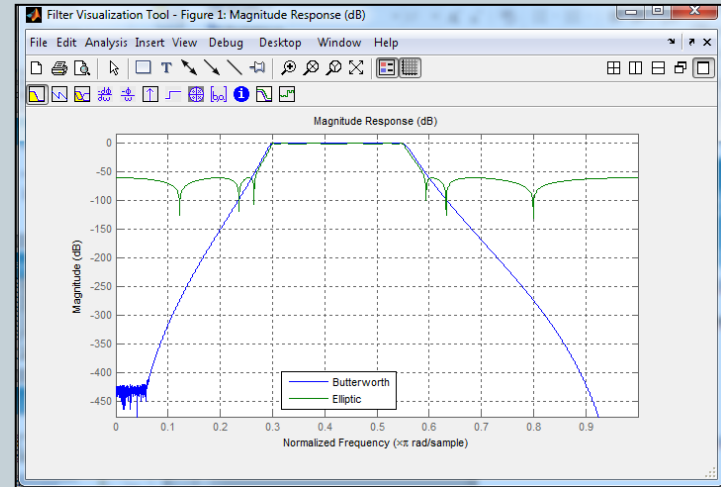
# Verificación del Diseño de Filtros

49

Con la herramienta Fvtool.

```
>> fvtool(numB,denB,numE,denE)
```

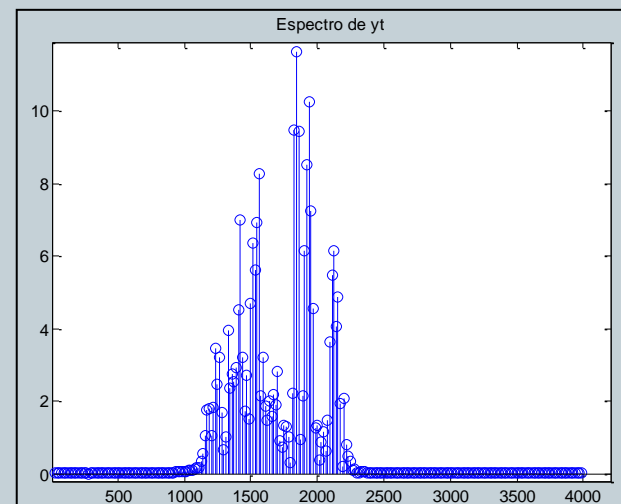
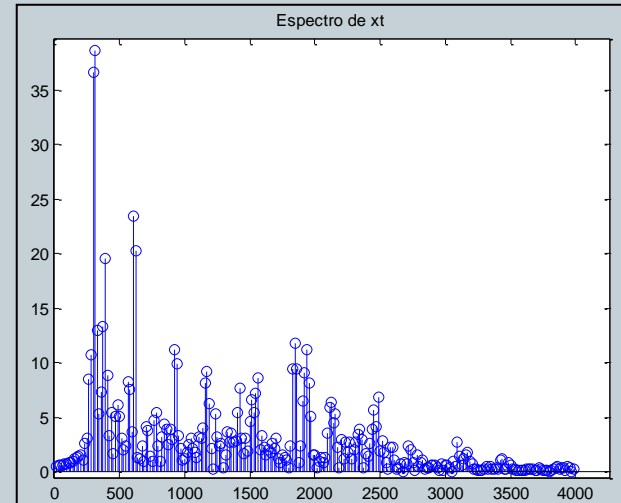
Se observa que ambos filtros son estables.



# Ejemplo: Filtrando el archivo «tuner1.wav»

50

```
>> [xt,fs]=wavread('tuner1');  
>> yt=filter(numE,denE,xt);  
>> soundsc(xt,fs)  
>> soundsc(yt,fs)
```



# Llamada a los Comandos de Diseño de Filtros

51

Los más usuales son los siguientes:

- Butterworth – Pasabajas

**[num,den]=butter(nB,Wn)**

- Elíptico – Pasabajas

**[num,den]=ellip(nE,Rp,Rs,Wp)**

- Butterworth – Pasaaltas

**[num,den]=butter(nB,Wn,'high')**

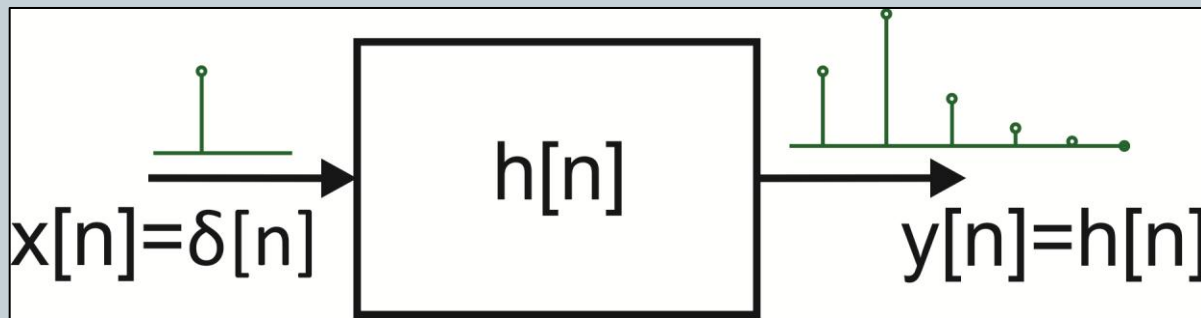
- Elíptico – Pasaaltas

**[num,den]=ellip(nE,Rp,Rs,Wp,'high')**

# Diseño de Filtros FIR

52

**FIR:** Finite Impulse Response.



Función de transferencia:

$$H(z) = \sum_{k=0}^M b_k \cdot z^{-k}$$

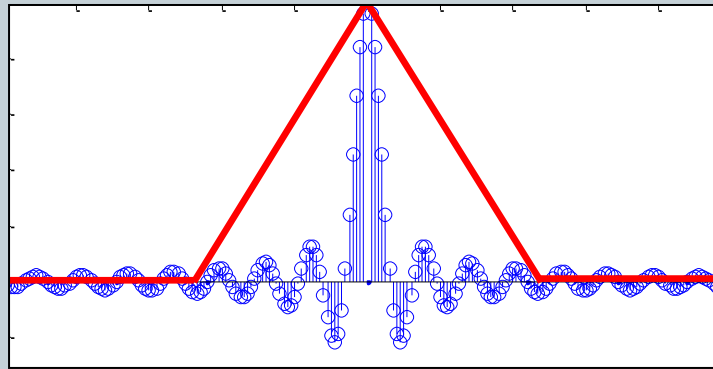
Respuesta en frecuencia:

$$H(\omega) = \sum_{k=0}^M b_k \cdot e^{-j \cdot k \cdot \omega}$$

# Diseño Mediante Método de Ventanas

53

Se multiplica la respuesta en el tiempo de un filtro ideal con una función ventana. Truncando la respuesta ideal a un número finito de elementos.



Se realiza mediante el comando:

**num=fir1(n,Wp)**

**Dónde:**

**n:** Orden del filtro

**Wp:** Vector que contiene las frecuencias de corte. Si sólo tiene un elemento se obtiene un filtro pasabajas y si contiene dos elementos se obtiene un filtro pasabanda.

# Ejemplo:

54

Se diseña un filtro mediante el método de ventanas para las siguientes especificaciones:

$F_s = 8000$  Hz

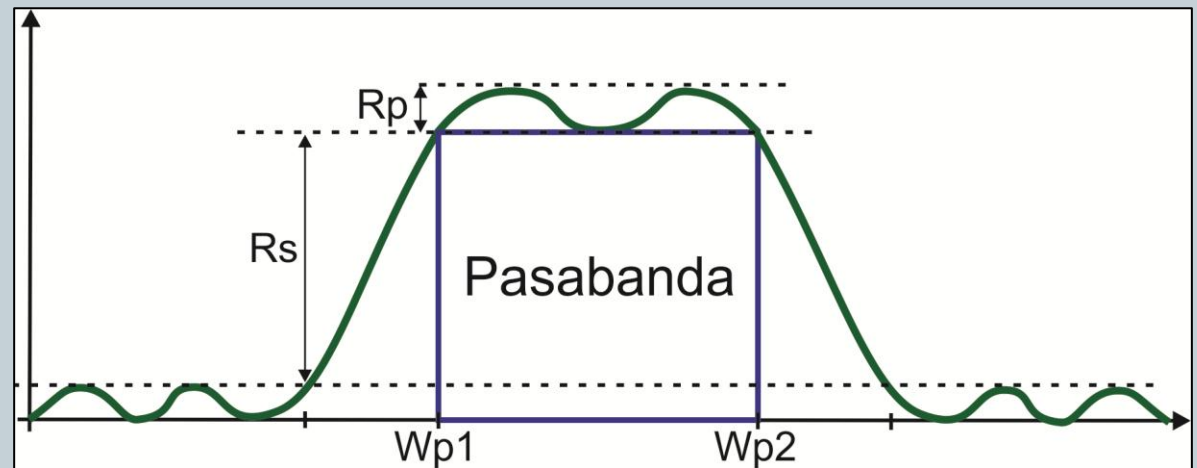
$f_{\text{pass1}} = 2200$  Hz

$f_{\text{pass2}} = 3200$  Hz

$R_p = 1$  dB

$R_s = 60$  dB

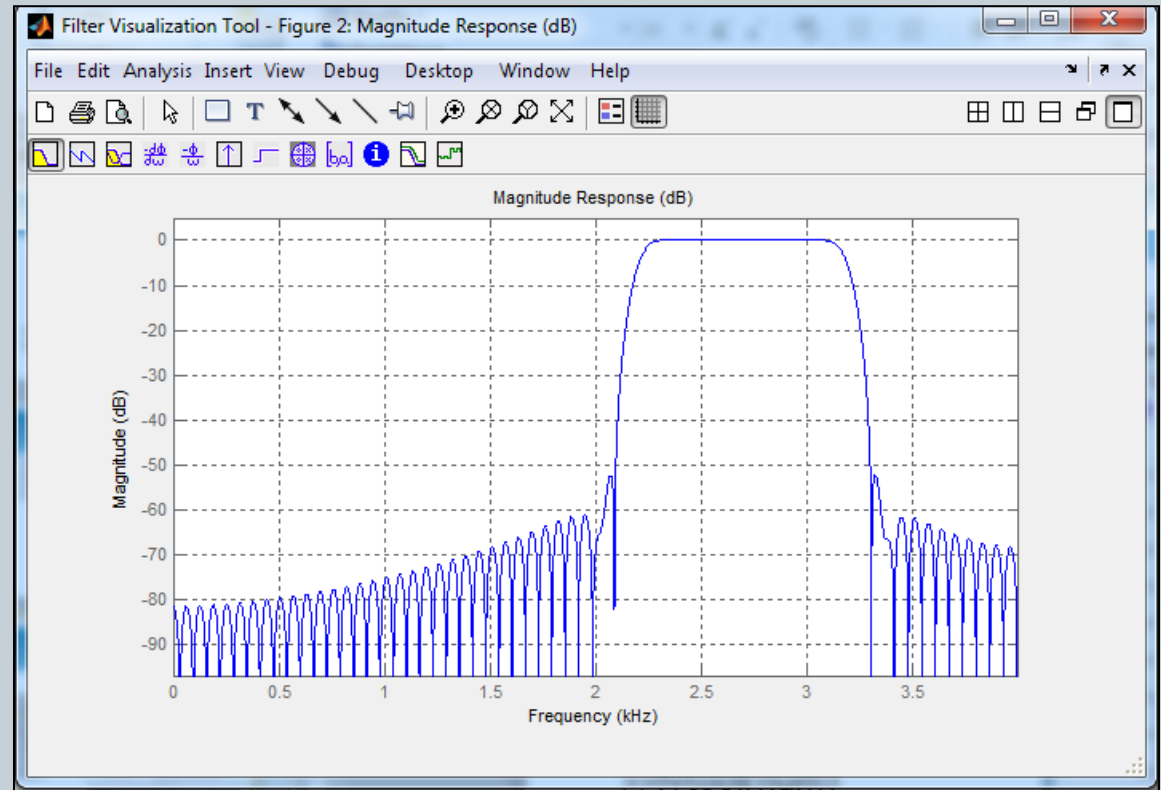
Orden=127



# Ejemplo:

55

```
>> Fs=8000;  
>> Fnyquist=Fs/2;  
>> fpass1=2200;  
>> fpass2=3200;  
>> Rp=1;  
>> Rs=60;  
>> Wp(1)=fpass1/Fnyquist;  
>> Wp(2)=fpass2/Fnyquist;  
>> n=127;  
>> num=fir1(n,Wp);  
>> fvtool(num)
```



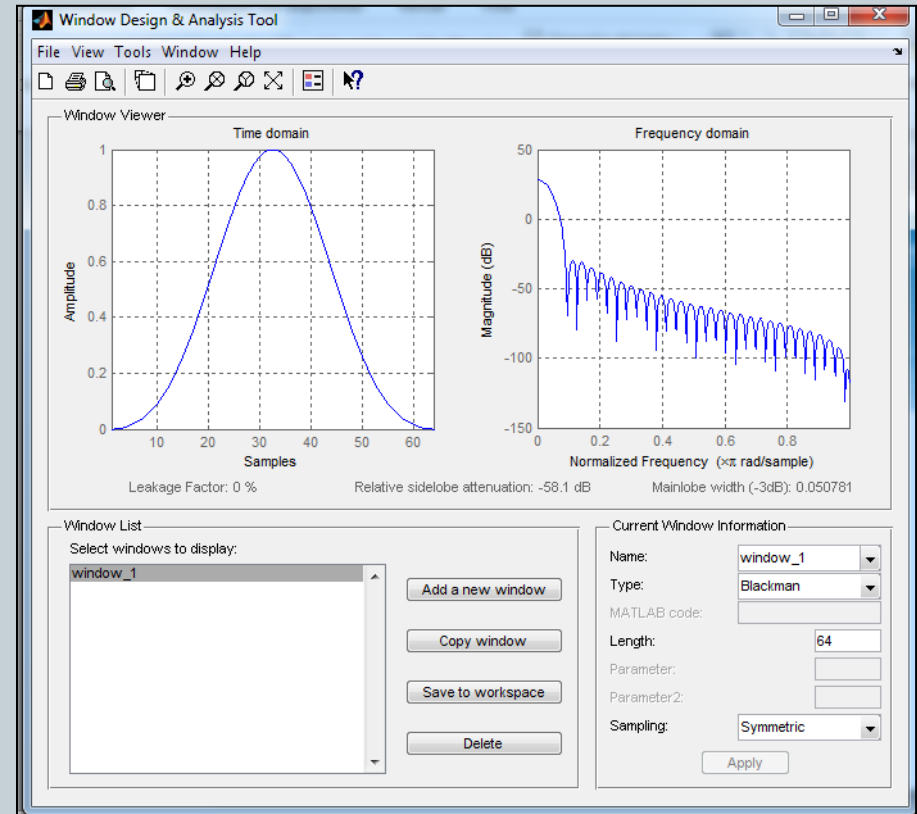
# Wintool Para diseño de Funciones Ventana

56

```
>> wintool
```

Initializing Window Design & Analysis Tool ..... done.

Herramienta con interfaz GUI que permite diseñar y visualizar la respuesta en el tiempo y en frecuencia de varios tipos de funciones ventana.





# Propiedades de Algunas Funciones Ventana de Longitud M

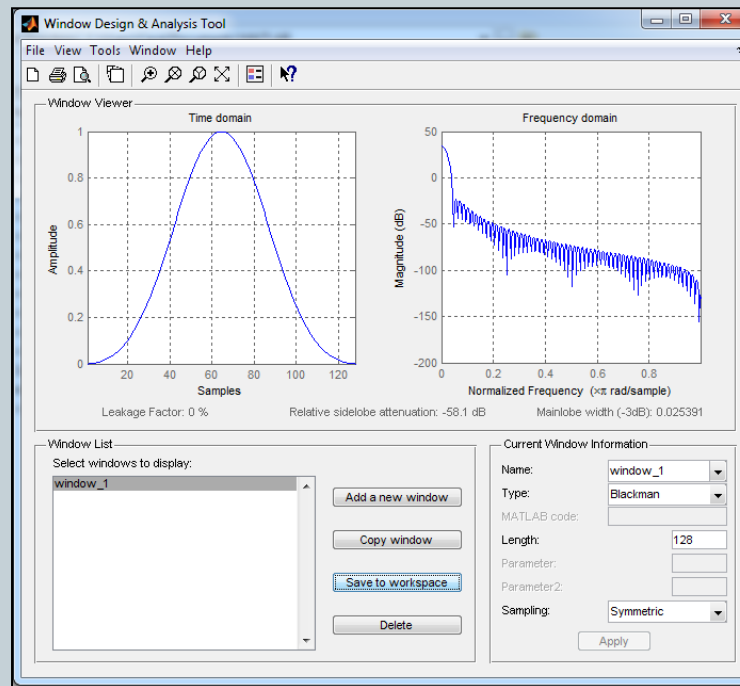
57

Tipo	Anchura de Lóbulo Principal	Mínima Atenuación en la Banda de Rechazo	Anchura de Banda de Transición
Rectangular	$2 \cdot F_s / (2 \cdot M + 1)$	20.9	$0.46 \cdot F_s / M$
Hann	$4 \cdot F_s / (2 \cdot M + 1)$	43.9	$1.55 \cdot F_s / M$
Hamming	$2 \cdot F_s / (2 \cdot M + 1)$	54.5	$1.55 \cdot F_s / M$
Blackman	$6 \cdot F_s / (2 \cdot M + 1)$	75.3	$2.78 \cdot F_s / M$

# Diseño de un Filtro FIR Utilizando una Ventana Diseñada Previamente

58

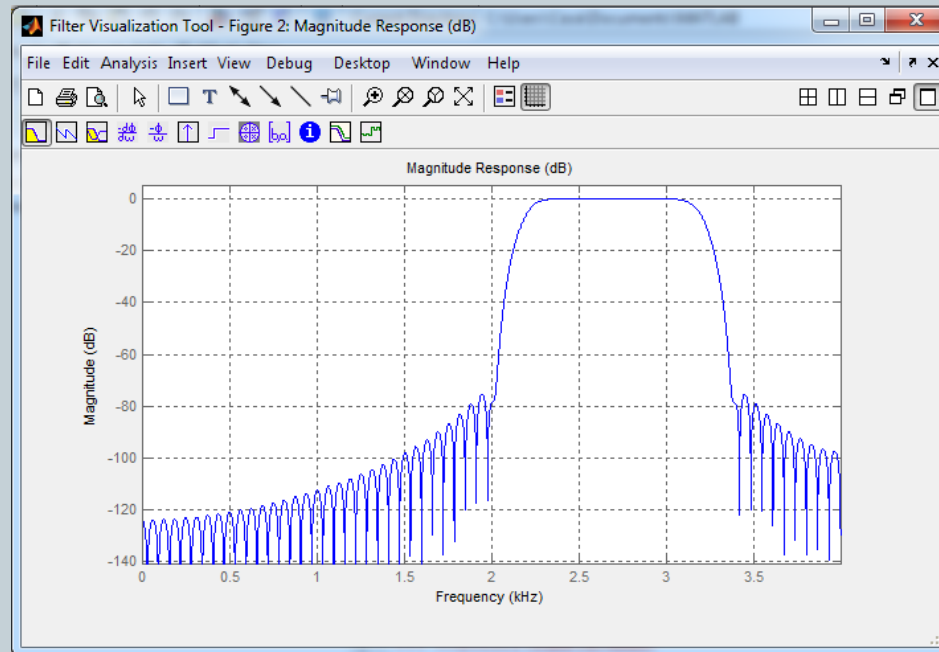
Se diseña la función ventana Blackman de longitud 128 mediante la herramienta wintool y se exporta a Matlab con «Save to workspace». El nombre por defecto es window\_1.



# Continuación...

59

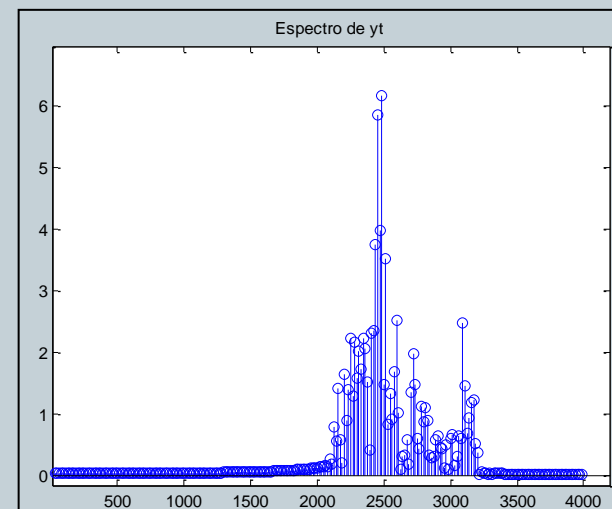
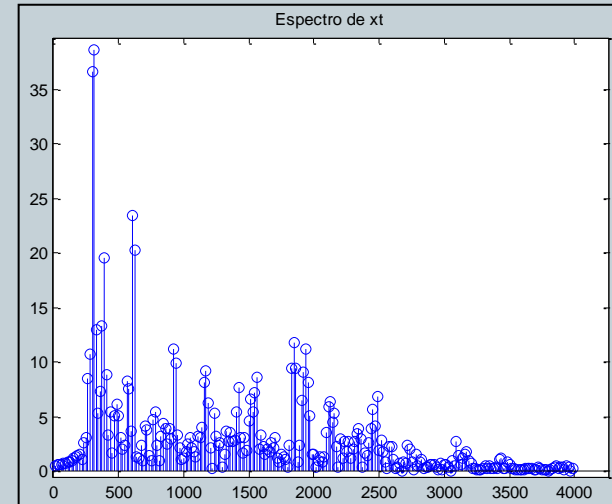
```
>> numBlackman=fir1(n,Wp>window_1);  
>> fvtool(numBlackman)
```



# Ejemplo: Filtrando el archivo de audio «tuner1.wav» con el filtro diseñado.

60

```
>> [xt,Fs]=wavread('tuner1');  
>> yt=filter(numBlackman,1,xt);  
>> soundsc(xt,Fs)  
>> soundsc(yt,Fs)
```

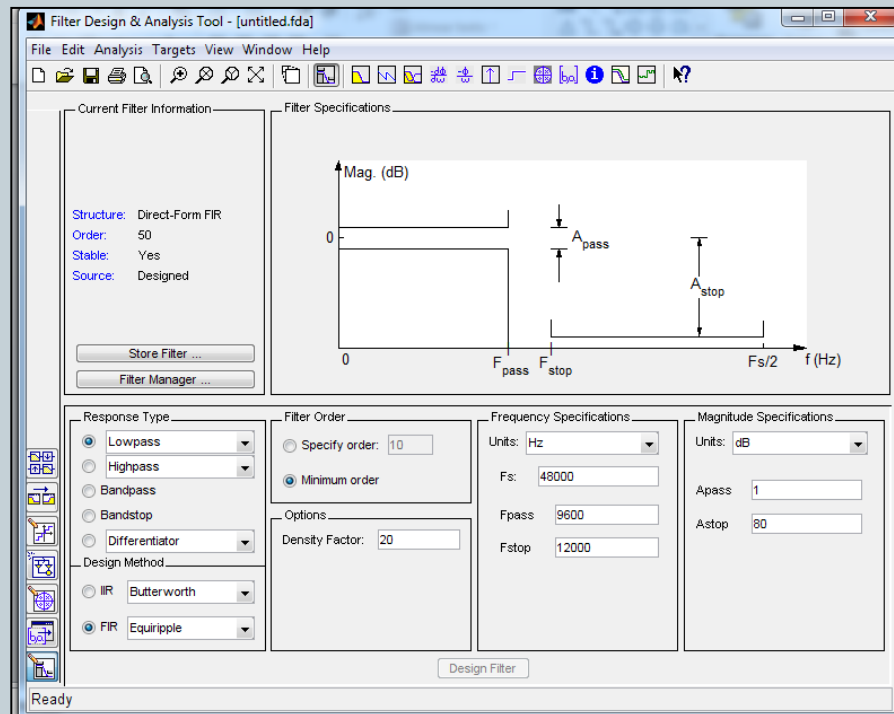


# Diseño de Filtros Mediante la Herramienta FDATool

61

**Fdatool** (Filter Design and Analysis tool) es una herramienta con interfaz GUI para el diseño de filtros digitales.

>> fdatool



# Diseño de un Filtro usando FDATool

62

Se diseñará un filtro pasaaltas con las siguientes especificaciones:

Pasaaltas

Tipo IIR – Elíptico

Mínimo orden posible

$F_s = 8000$  Hz

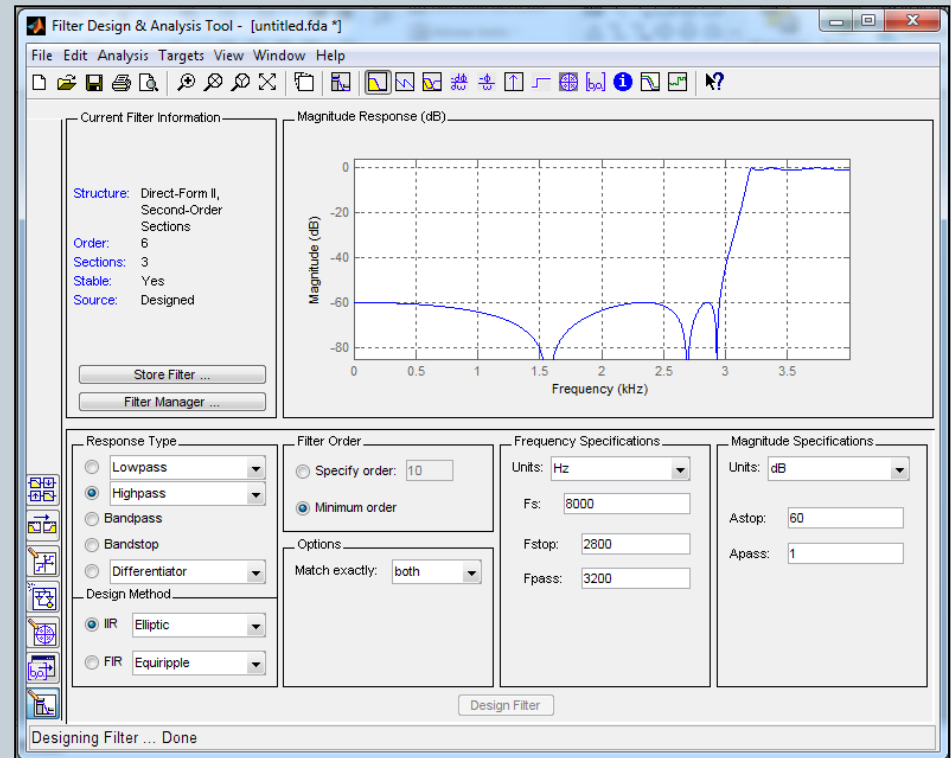
$F_{stop} = 2800$  Hz

$F_{pass} = 3200$  Hz

$R_s = 60$  dB

$R_p = 1$  dB

Se introducen todas estas Especificaciones en la herramienta y un clic en Design Filter.

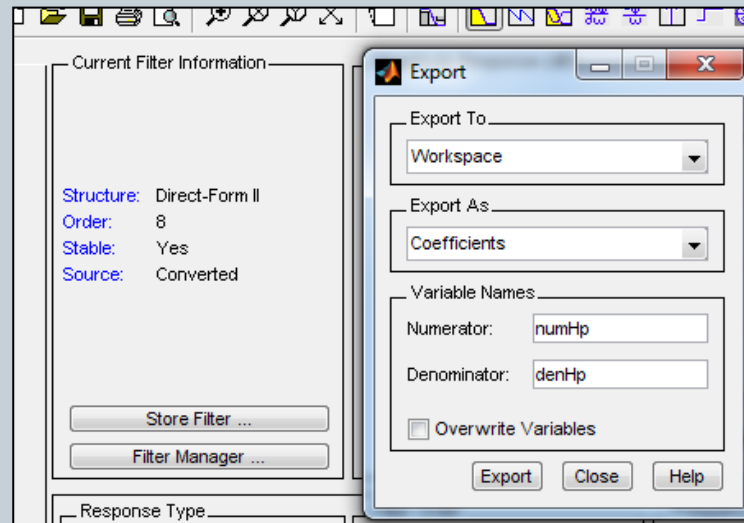


# Exportación al Área de Trabajo

63

Se realiza el siguiente procedimiento:

1. Clic derecho en «**Structure**» en el campo «**Current filter information**» y en seguida se escoge «**convert to single section**».
2. Menú **File>Export...** Se le da nombres al numerador y denominador del filtro. Luego clic en **Export**.



# Ejemplo: Filtrando el archivo de audio «tuner1.Wav»

64

```
>> [xt,Fs]=wavread('tuner1');  
>> yt=filter(numHp,denHp,xt);  
>> soundsc(xt)  
>> soundsc(yt)
```

