DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Laboratorio 4.5

PROCESAMIENTO DE SEÑALES E IMÁGENES

Profesores:

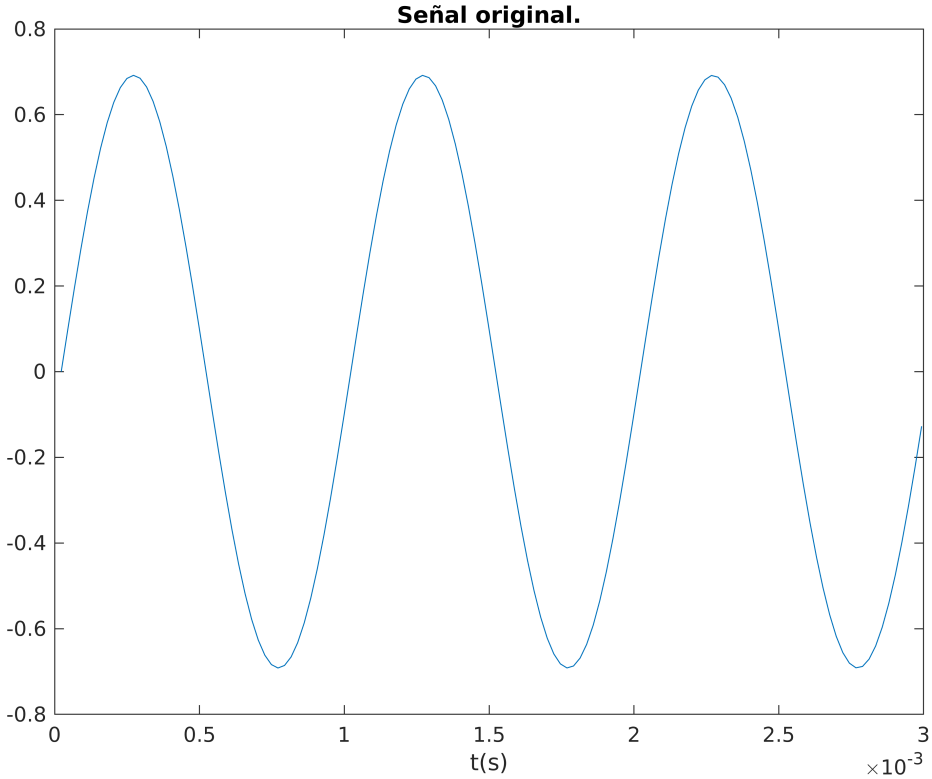
- Violeta Chang C.
- Leonel E. Medina

Ayudante: Luis Corral

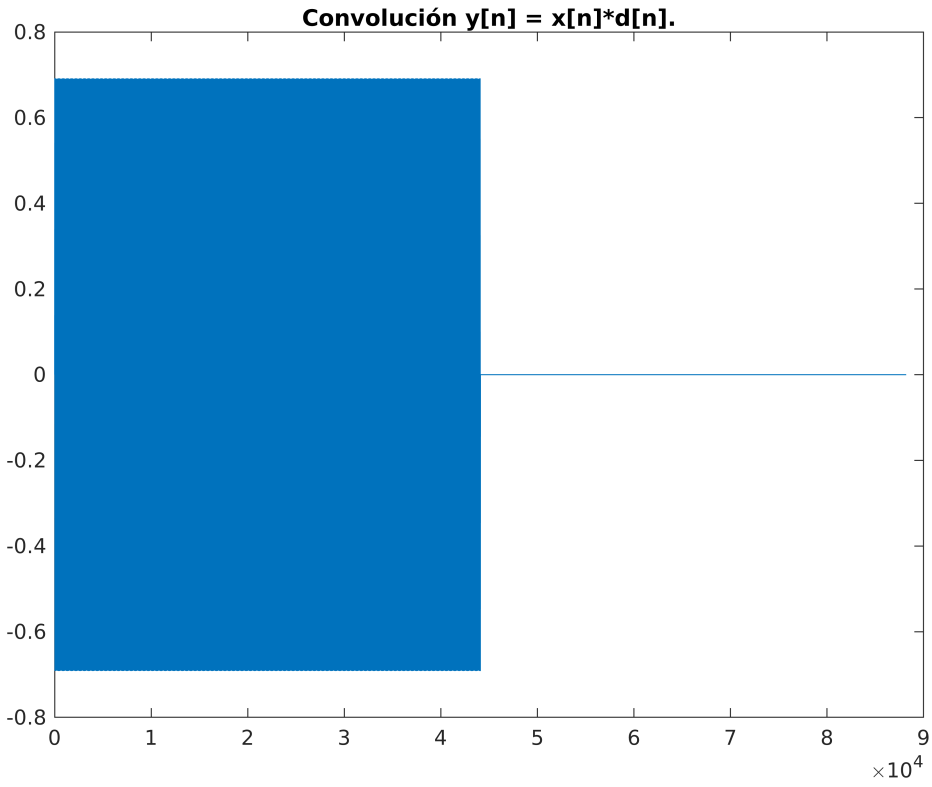
Convolución en tiempo discreto

Como vimos en ejemplos anteriores, la convolución en tiempo discreto se implementa en Matlab a partir de la función conv. Esta función, entrega un resultado de largo $N_1 + N_2 + 1$ muestras, siendo N_1 y N_2 los largos de las secuencias de entrada.

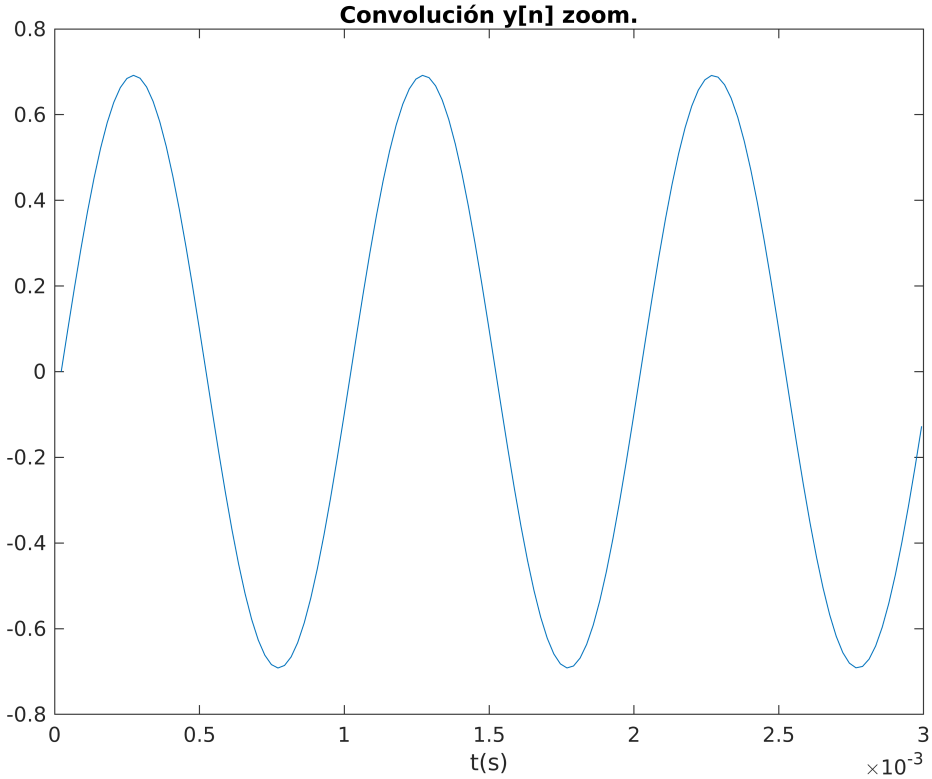
```
clearvars

[x_n,Fs] = audioread('44100Hz.wav');
x_n = x_n(1:Fs);                    % Señal 1 segundo
N = length(x_n);                    % Total de muestras N
figure
plot((1:3*round((1/1000)/(1/Fs)))/Fs,...
     x_n(1:3*round((1/1000)/(1/Fs))))  % Gráfico 3 ciclos
title('Señal original.')
xlabel('t(s)')
```




```
d_n = zeros(1,N);          % Impulso unitario de largo N
d_n(1) = 1;

y_n = conv(x_n,d_n);       % Convolución y[n] = x[n]*d[n]
figure
plot(y_n)
title('Convolución y[n] = x[n]*d[n].')
```

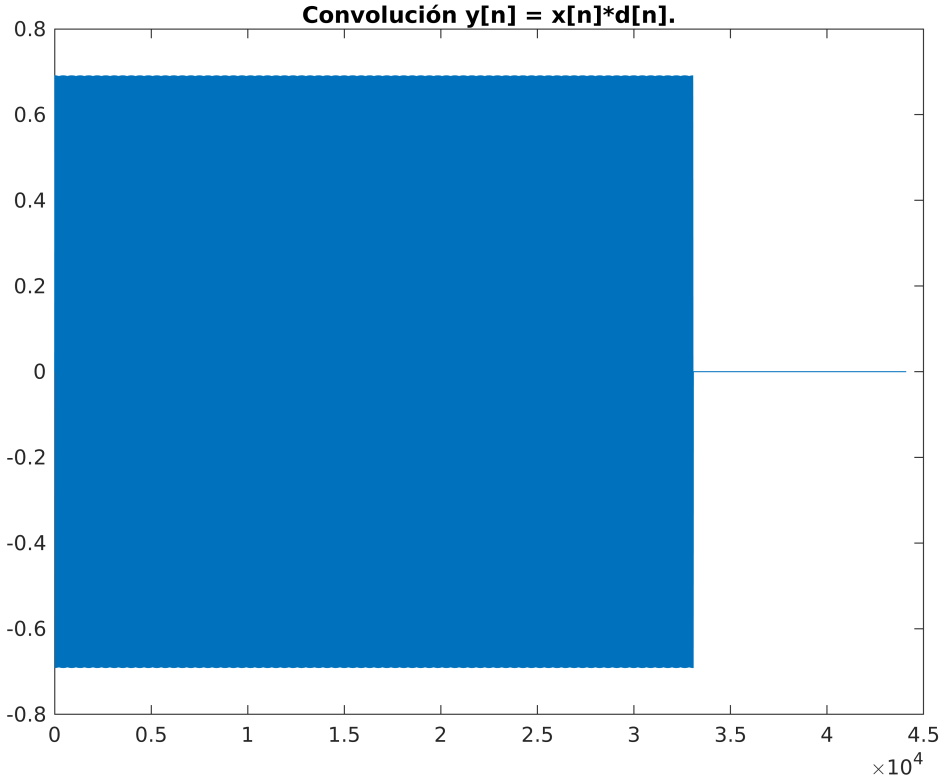



```
plot((1:3*round((1/1000)/(1/Fs)))/Fs,...
     y_n(1:3*round((1/1000)/(1/Fs))))
title('Convolución y[n] zoom.')
xlabel('t(s)')
```

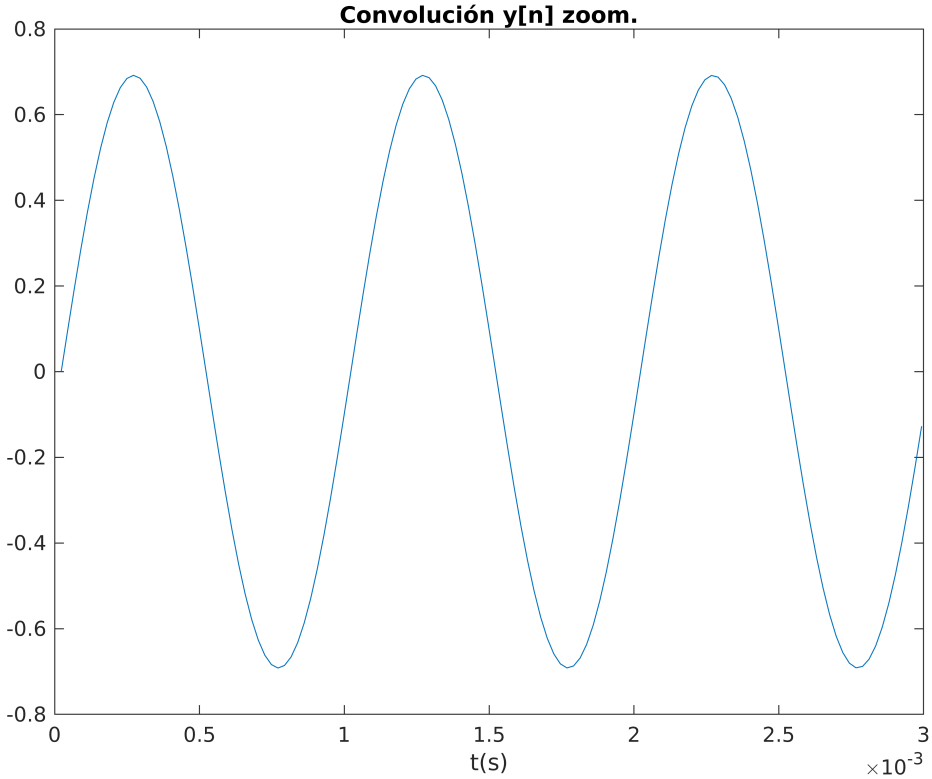


La opción 'same' de la función conv nos permite truncar el resultado para obtener una señal del mismo largo que la correspondiente al primer parámetro.

```
d_n = zeros(1,round(N/2)); % Impulso unitario de largo N/2
d_n(1) = 1;

y_n = conv(x_n,d_n,'same'); % Convolución y[n] = x[n]*d[n]
figure
plot(y_n)
title('Convolución y[n] = x[n]*d[n].')
```



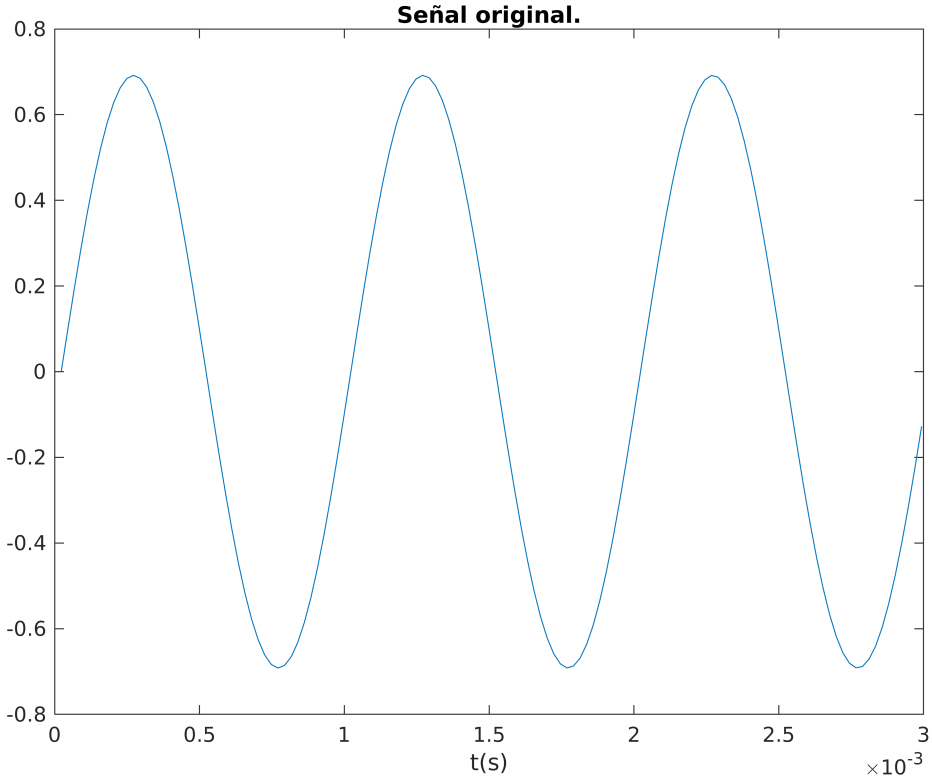

```
plot((1:3*round((1/1000)/(1/Fs)))/Fs,...
     y_n(1:3*round((1/1000)/(1/Fs))))
title('Convolución y[n] zoom.')
xlabel('t(s)')
```



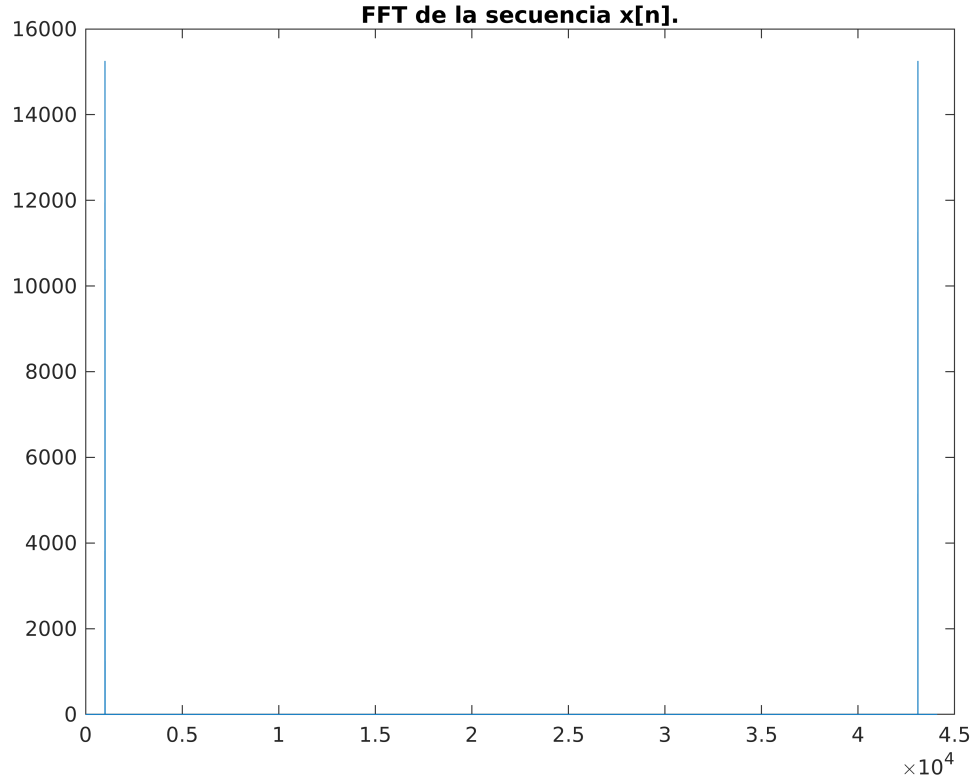
DFT y FFT

La transformada discreta de Fourier $X(e^{j\omega})$ o DFT de una secuencia $x[n]$ nos permite obtener su contraparte en el dominio de la frecuencia ω. Actualmente, la implementación más utilizada es la transformada rápida de Fourier o FFT, la cual es un caso especial donde el largo de las secuencias debe ser una potencia de dos. A diferencia de su contraparte continua, la DFT y su implementación en la FFT son siempre simétricas y al ser discretas obtendremos un resultado del mismo largo que la secuencia de entrada.

```
clearvars

[x_n,Fs] = audioread('44100Hz.wav');
x_n = x_n(1:Fs);

figure
plot((1:3*round((1/1000)/(1/Fs)))/Fs,...
     x_n(1:3*round((1/1000)/(1/Fs)))) % 1 segundo dominio tiempo
title('Señal original.')
xlabel('t(s)')
```

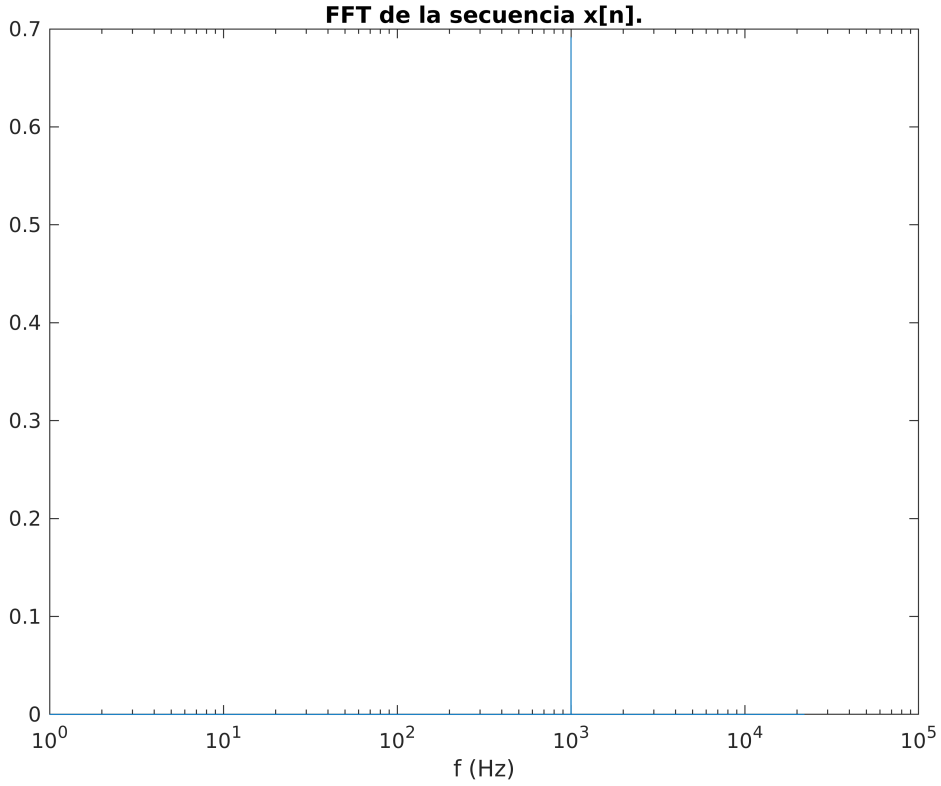



```
X_ejw = fft(x_n);                % FFT de 1 segundo

figure
plot(abs(X_ejw))
title('FFT de la secuencia x[n].')
```



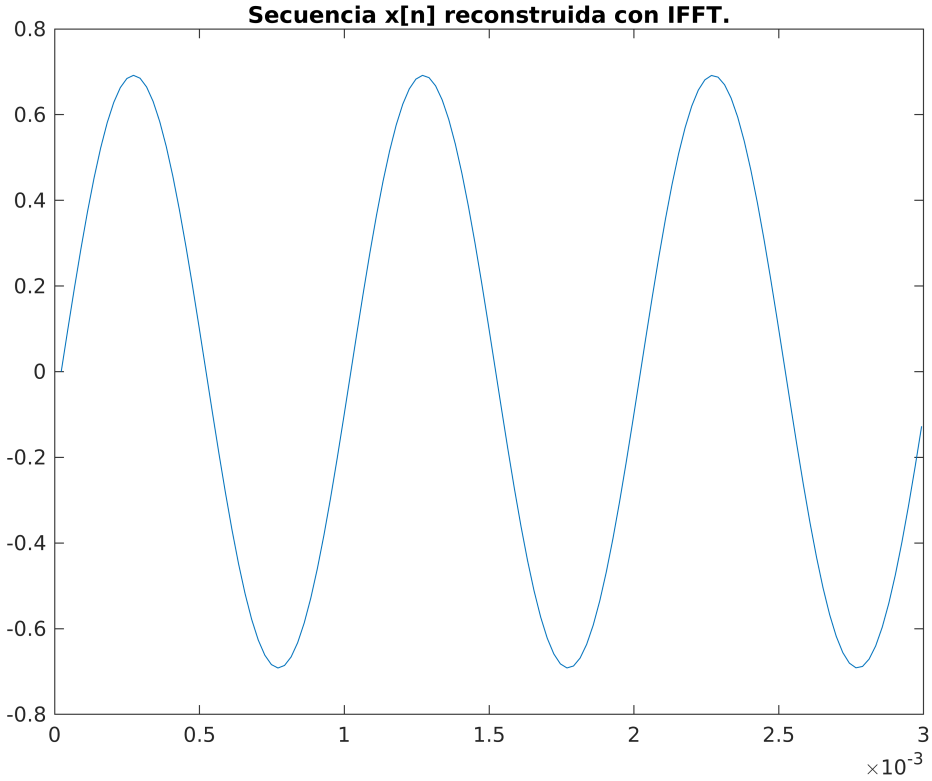
Al igual que luego de leer una secuencia podíamos obtener los valores para el dominio del tiempo t utilizando la frecuencia de muestreo, en este caso utilizamos el mismo valor para obtener el dominio de la frecuencia ω.

```
% https://la.mathworks.com/help/releases/R2020a/matlab/ref/fft.html#buuutyt-9

N = length(x_n);
X_ejw_s = 2*abs(X_ejw(1:round(N/2)+1))/N;
f_hz = Fs*(0:round(N/2))/N;

figure
semilogx(f_hz,X_ejw_s)    % Gráfico con eje x logarítmico
title('FFT de la secuencia x[n].')
xlabel('f (Hz)')
```



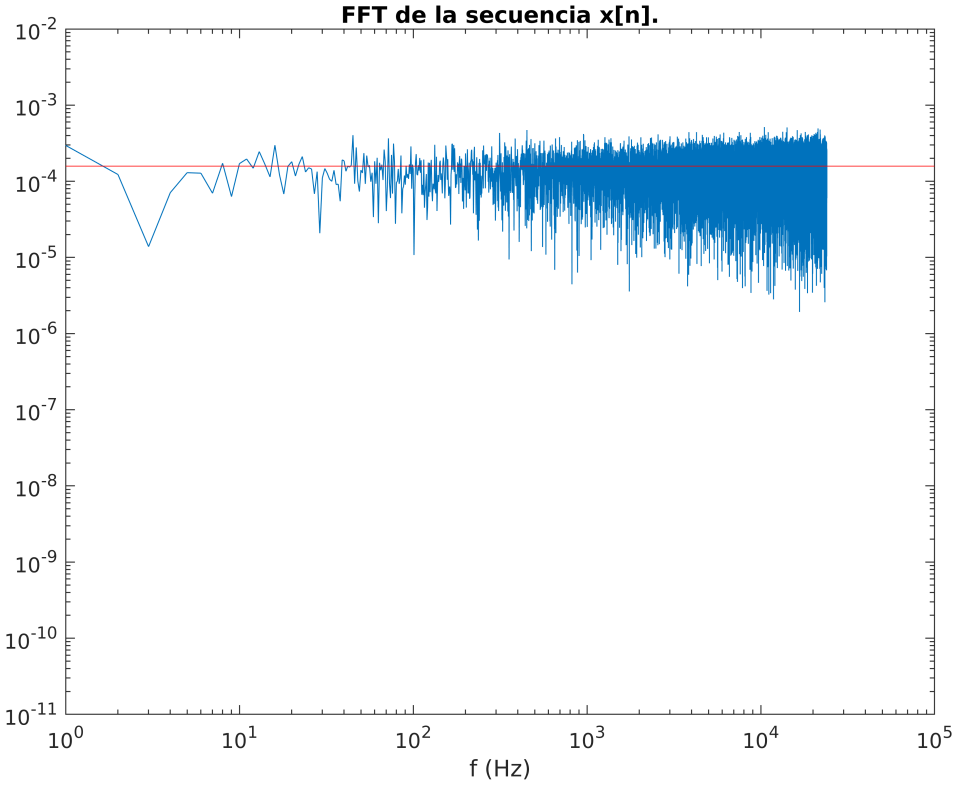
Podemos retornar al dominio del tiempo con la transformada rápida de Fourier inversa IFFT.

```
x_n_r = ifft(X_ejw);
figure
plot((1:3*round((1/1000)/(1/Fs)))/Fs,...
     x_n_r(1:3*round((1/1000)/(1/Fs))))
title('Secuencia x[n] reconstruida con IFFT.')
```



El gráfico de magnitud de la transformada de Fourier se conoce como espectro. En él, nos referimos al ancho de banda como la región del espectro donde se encuentra la mayor energía de la señal. Por ejemplo, un ruido blanco tiene igual energía en el espectro completo de frecuencias.

```
[x_n,Fs] = audioread('white_noise_263s_Matlab10_EDIT.wav');
x_n = x_n(1:Fs);
X_ejw = fft(x_n);
N = length(x_n);
X_ejw = X_ejw(1:round(N/2)+1);
f_hz = Fs*(0:round(N/2))/N;

figure
loglog(f_hz,2*abs(X_ejw)/N)    % Gráfico ejes x-y logarítmicos
ylim([10e-12 10e-3])
hold on
yline(rms(2*abs(X_ejw)/N),'-r')
title('FFT de la secuencia x[n].')
xlabel('f (Hz)')
```

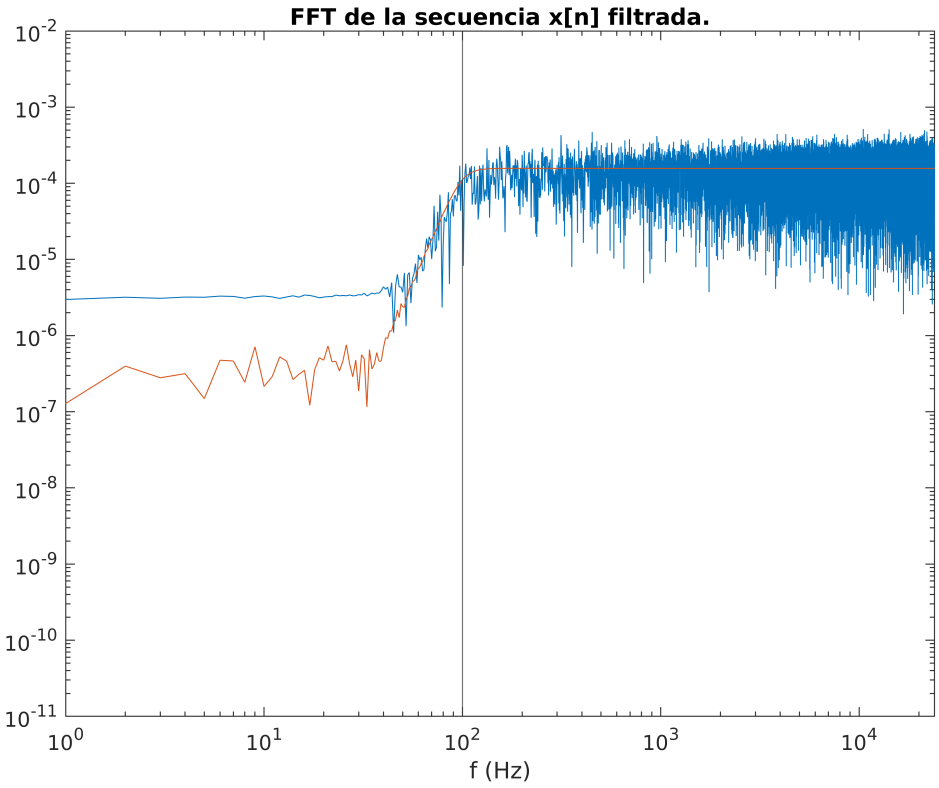


Filtros digitales

El estudio de los filtros digitales tiene como mayor dificultad lograr que la respuesta del sistema sea estable. En la literatura se pueden encontrar distintos tipos de filtros (Chebyshev, Bessel, entre muchos otros) que están implementados en el mundo digital. Matlab incluye varias funciones para esto, una de ellas es butter para diseño de filtros de tipo Butterworth y filter para su aplicación.

```
Fc = 100;                                   % frecuencia corte
orden = 6;                                  % orden del filtro
[b,a] = butter(orden,Fc/(Fs/2),"high");  % coeficientes del filtro
x_n_f = filter(b,a,x_n);
X_ejw = fft(x_n_f);                 % FFT señal resultante
N = length(x_n);
X_ejw = X_ejw(1:round(N/2)+1);
f_hz = Fs*(0:round(N/2))/N;

figure
[h,w] = freqz(b,a,(N/2)+1,Fs);      % Respuesta en frecuencia
                                    % del filtro.
```

```
figure
loglog(f_hz,2*abs(X_ejw)/N, ...
       f_hz,abs(h)*rms(2*abs(X_ejw)/N))
ylim([10e-12 10e-3])
xlim([0 max(f_hz)])
xline(Fc)
title('FFT de la secuencia x[n] filtrada.')
xlabel('f (Hz)')
```



Referencias

[1] Oppenheim, A.V. & Willsky, A.S. & Nawab, S.H. (1997). Señales y sistemas (2nd ed.). Prentice Hall.