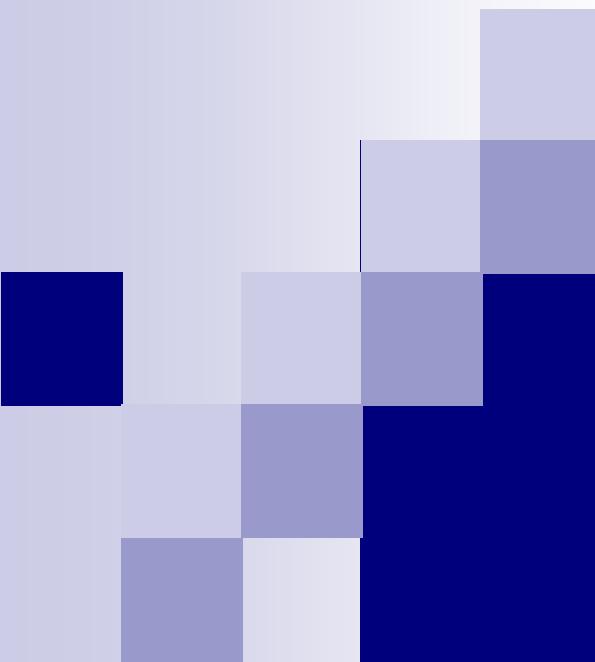


Grado en Ingeniería Informática



# Procesamiento Digital de Señales

Tema 1. Señales y sistemas en tiempo discreto

Javier Ramírez Pérez de Inestrosa

# Índice

1. Introducción
2. Definición y clasificación de señales
3. Ventajas del procesado digital frente al procesado analógico
4. Elementos de un sistema de procesamiento digital de señales
  - Acondicionamiento
  - Conversión A/D (muestreo)
  - Cuantización: logarítmica, adaptable y predictiva
5. Análisis de señales continuas en el dominio de la frecuencia:
  - Series de Fourier y Transformada de Fourier
6. Señales y sistemas en tiempo discreto
  - Señales elementales
  - Sistemas en tiempo discreto: lineales, invariantes en el tiempo y LTI
  - Respuesta impulsiva
  - Convolución
  - Estabilidad y causalidad
  - Clasificación de sistema LTI: FIR e IIR
  - Ecuaciones en diferencias
  - Transformada de Fourier en tiempo discreto (DTFT)
  - Respuesta en frecuencia de sistemas LTI

# 1. Introducción

## ■ ¿Qué es el procesado digital de señal?

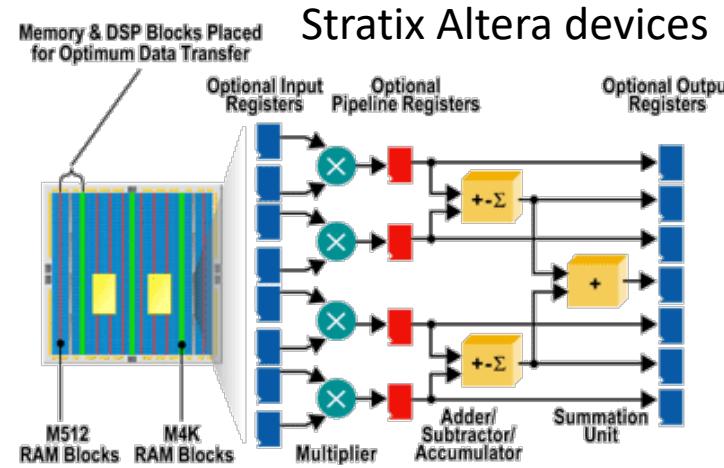
- Disciplina de la ciencia y la ingeniería que trata con la representación, transformación y manipulación numérica de señales y con la información que estas contienen.



- Se ha desarrollado fundamentalmente durante los últimos 40 años debido al:

1. Rápido desarrollo de los computadores digitales y microprocesadores
2. Avance de la tecnología VLSI
3. Desarrollo de los fundamentos del procesado digital de señales:
  - Algoritmos de transformada rápida de Fourier (FFT)
  - Modelado paramétrico de señal
  - Procesamiento multitasa
  - Implementación de filtros polifase
  - Nuevas formas de representación de señales (transformadas)

4. Numerosas aplicaciones

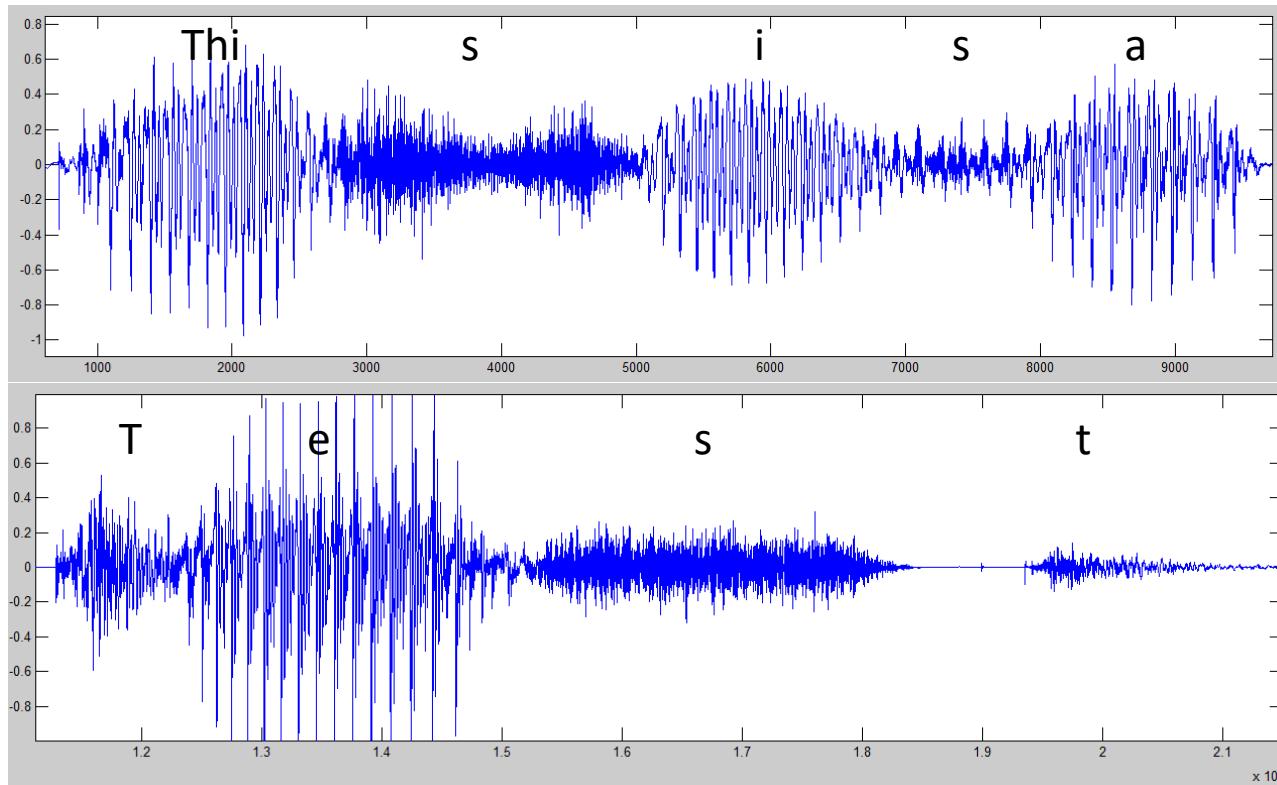


# Definición de procesado digital de señal

## ■ Representación de señales

- Señales voz: Análisis de voz  
(sonidos sordos/sonoros,  
estimación de frecuencia de *pitch*).

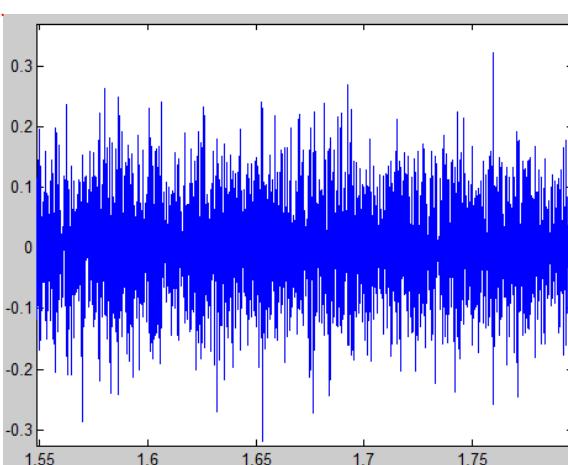
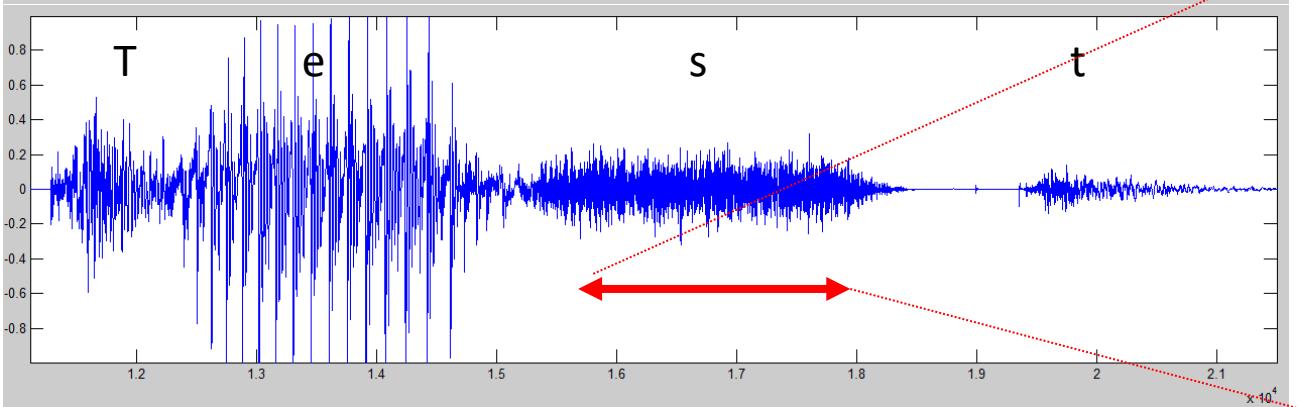
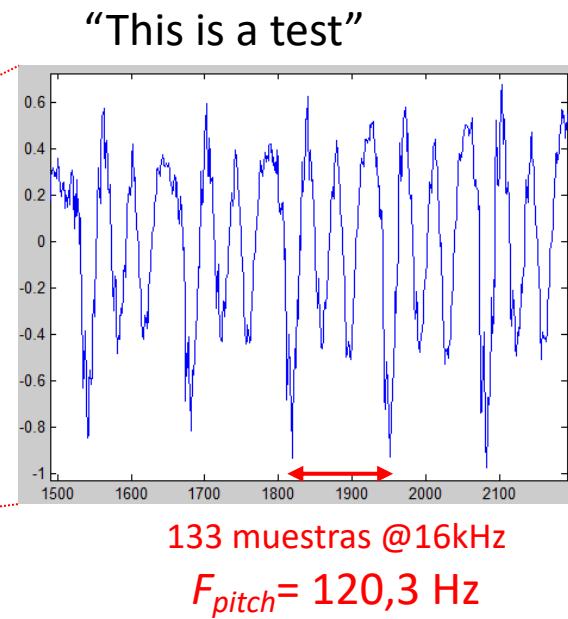
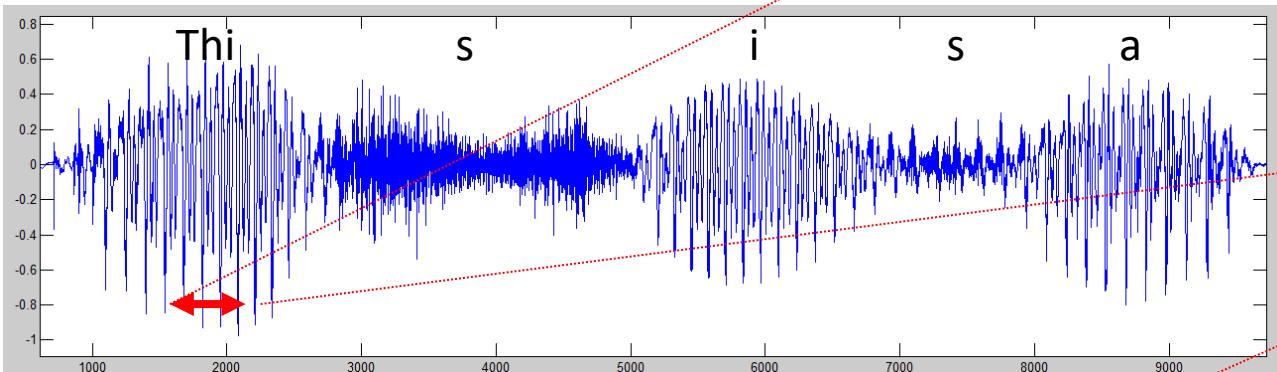
“This is a test”



# Definición de procesado digital de señal

## ■ Representación de señales

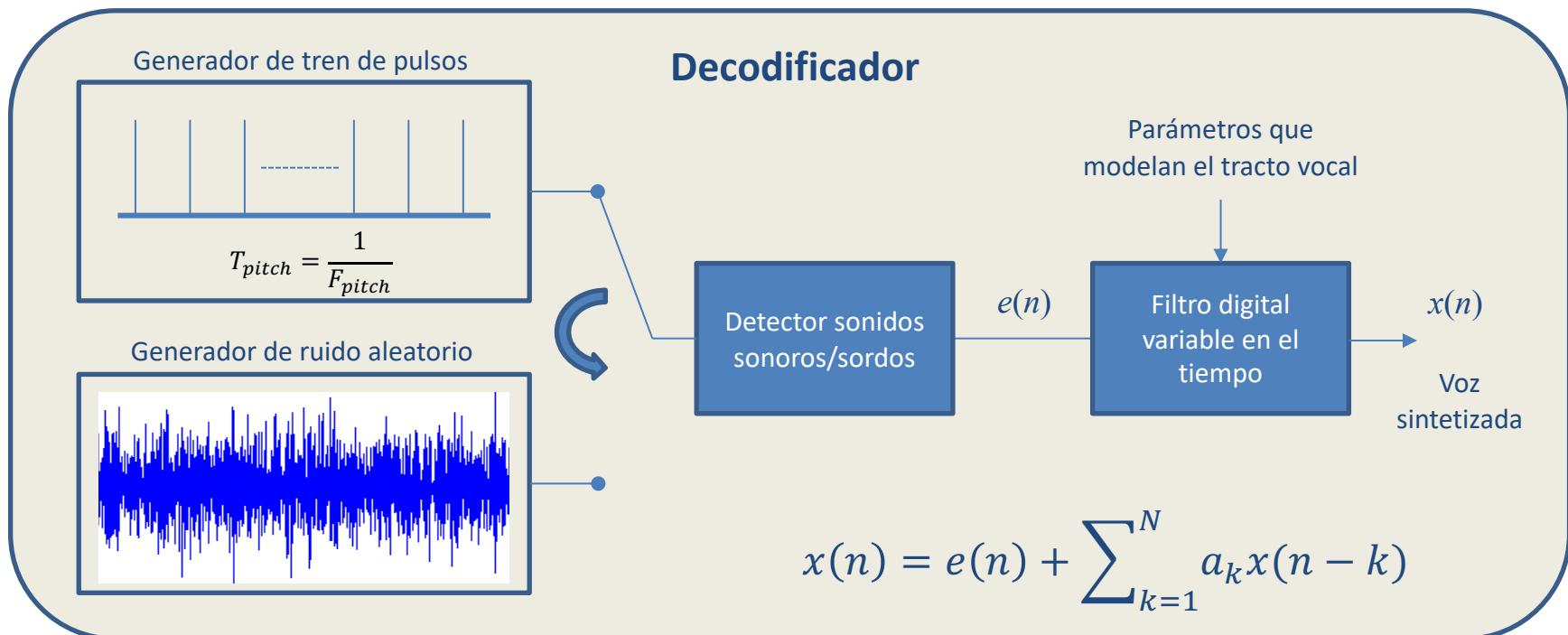
- Señales voz: Análisis de voz  
(sonidos sordos/sonoros, estimación de frecuencia de *pitch*).



# Definición de procesado digital de señal

## ■ Representación de señales

- Señales voz: Análisis de voz  
(sonidos sordos/sonoros, estimación de frecuencia de *pitch*).

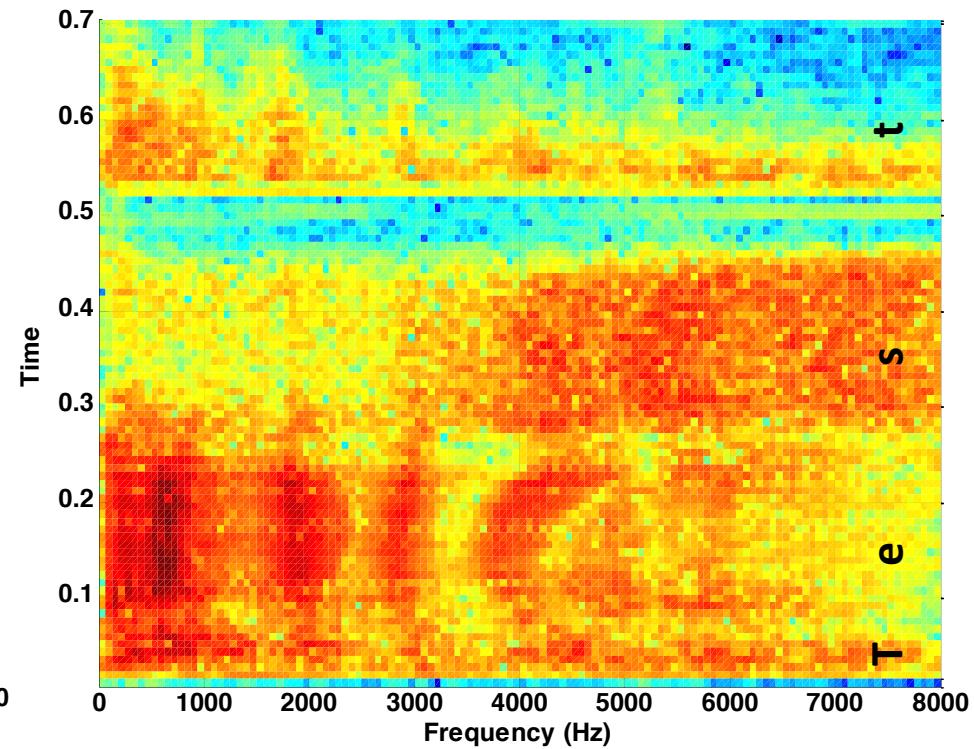
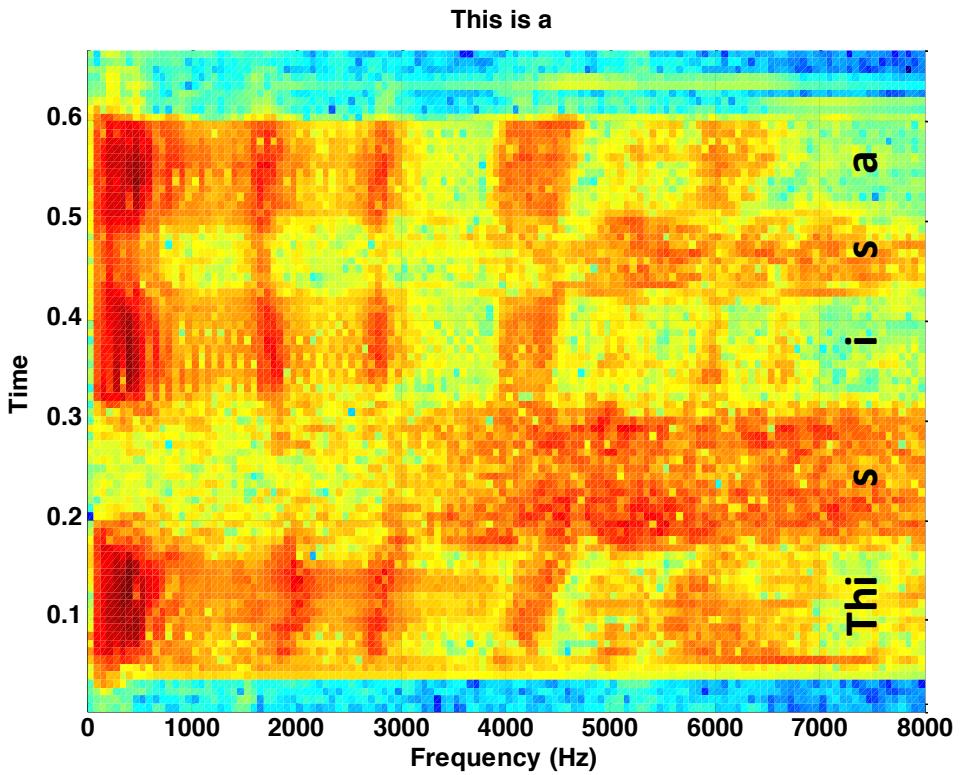
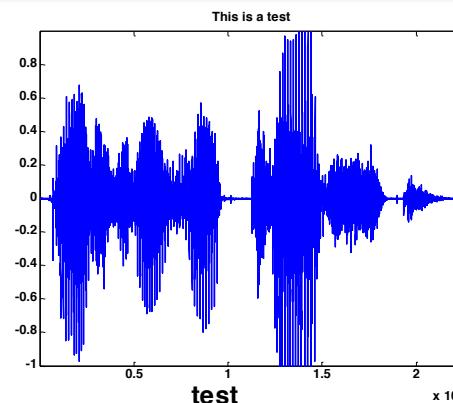


Síntesis de voz empleada en la mayoría de los codecs de voz.

# Definición de procesado digital de señal

## ■ Representación de señales

- Señales voz: Análisis de voz  
(sonidos sordos/sonoros,  
estimación de frecuencia de *pitch*).



# Definición de procesado digital de señal

## Ejemplo en Matlab:

```
wav_file= 'This_is_a_test.wav';

[x,Fs]= audioread(wav_file);

plot(x);
title('This is a test');

window=      256;
noverlap=    128;
nfft=        256;

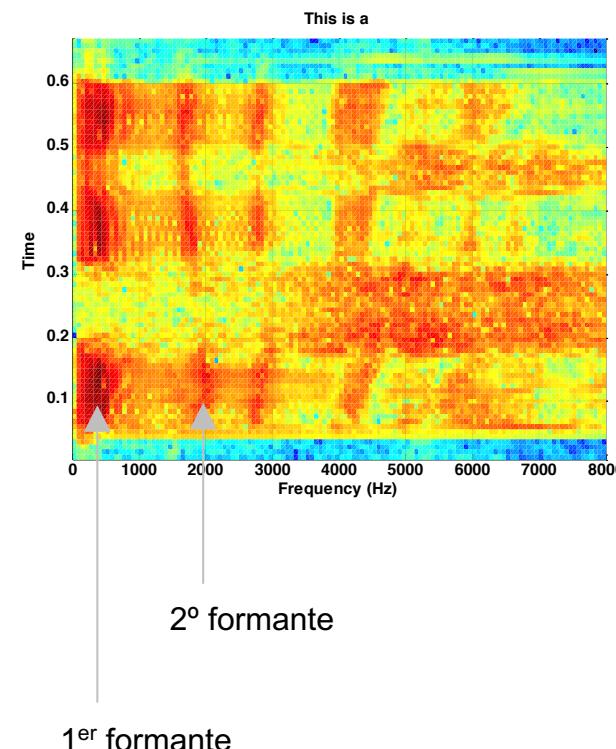
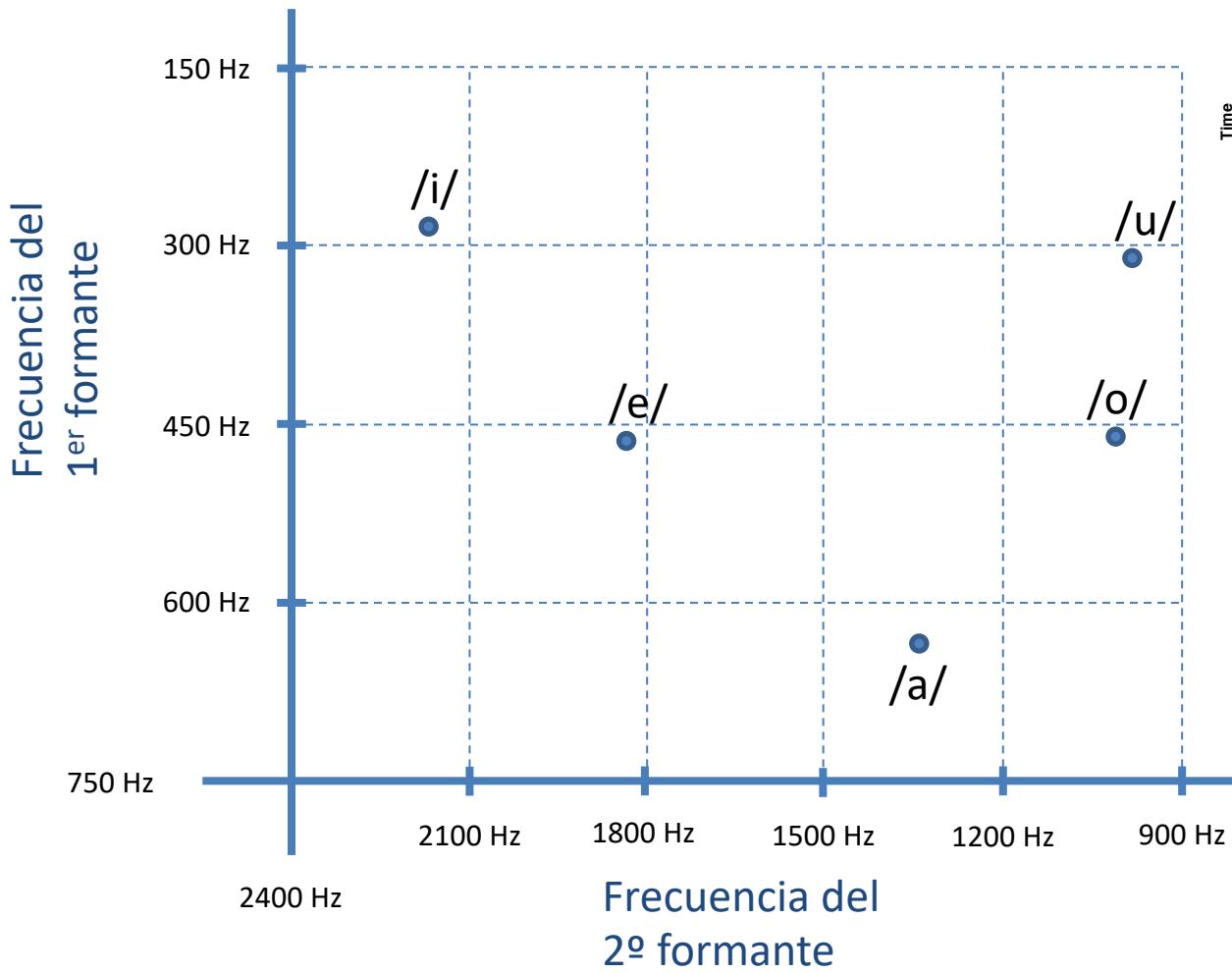
T1= 11000;

figure(1);
spectrogram(x(1:T1),window,noverlap,nfft,Fs);
title('This is a ')

figure(2);
spectrogram(x(T1:end),window,noverlap,nfft,Fs);
title('test')
```

# Definición de procesado digital de señal

- Formantes de la voz:

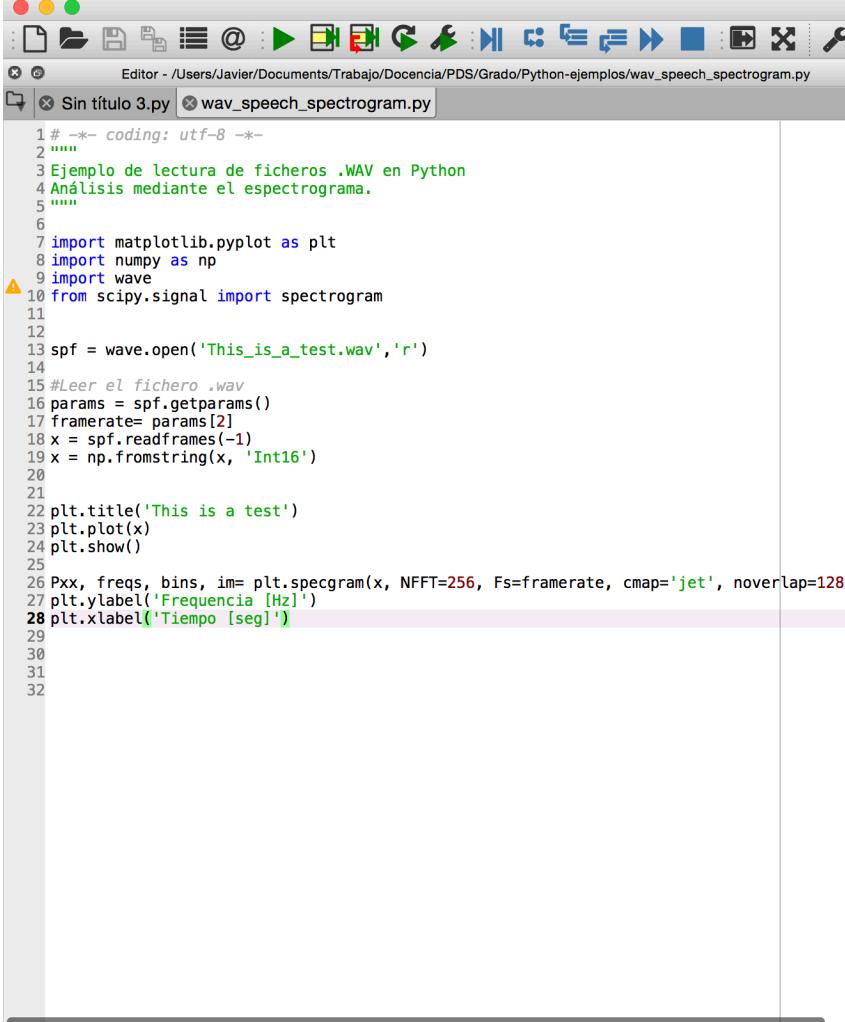


# Definición de procesado digital de señal

## Spyder IDE (Anaconda)

### Tutorial recomendado

[https://www.youtube.com/playlist?list=PLZzoUVCENTa0e1emDvvcW2vghyMj\\_w\\_bR](https://www.youtube.com/playlist?list=PLZzoUVCENTa0e1emDvvcW2vghyMj_w_bR)



The screenshot shows the Spyder IDE interface. On the left, the code editor displays a script named 'wav\_speech\_spectrogram.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Ejemplo de lectura de ficheros .WAV en Python
4 Análisis mediante el espectrograma.
5 """
6
7 import matplotlib.pyplot as plt
8 import numpy as np
9 import wave
10 from scipy.signal import spectrogram
11
12
13 spf = wave.open('This_is_a_test.wav', 'r')
14
15 #Leer el fichero .wav
16 params = spf.getparams()
17 framerate= params[2]
18 x = spf.readframes(-1)
19 x = np.fromstring(x, 'Int16')
20
21
22 plt.title('This is a test')
23 plt.plot(x)
24 plt.show()
25
26 Pxx, freqs, bins, im= plt.specgram(x, NFFT=256, Fs=framerate, cmap='jet', noverlap=128)
27 plt.ylabel('Frecuencia [Hz]')
28 plt.xlabel('Tiempo [seg]')
29
30
31
32
```

The right side of the interface shows two plots generated by the code. The top plot is a time-domain waveform titled "This is a test", showing amplitude over time from 0 to 20,000 seconds. The bottom plot is a spectrogram showing Frequency [Hz] from 0 to 8,000 Hz against Time [seg] from 0 to 1.2 seconds.

# Definición de procesado digital de señal

Ejemplo en Python:



wav\_speech\_spectrogram.py

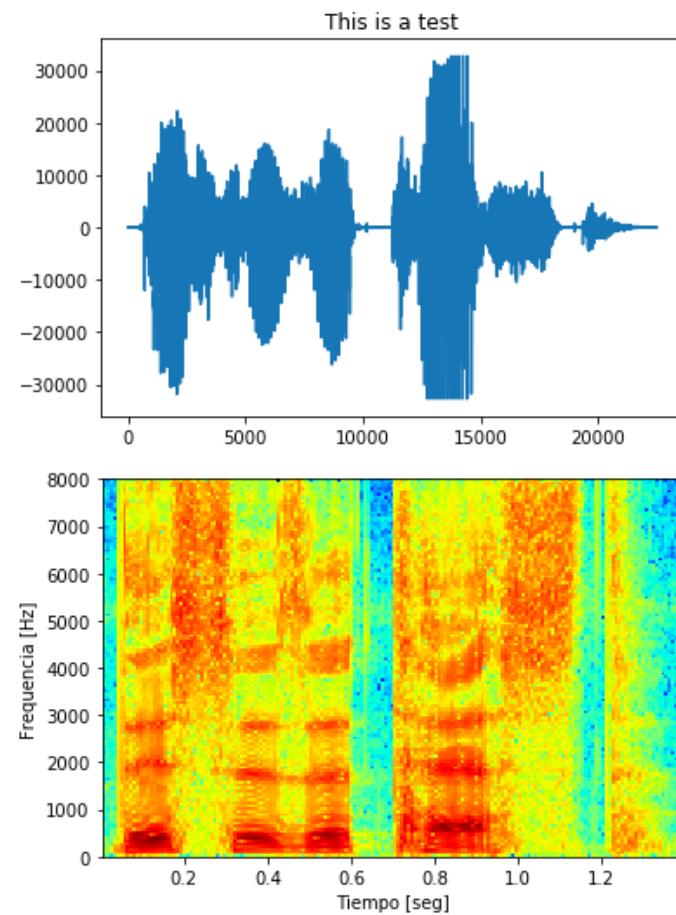
```
import matplotlib.pyplot as plt
import numpy as np
import wave

#Leer el fichero .wav
spf = wave.open('This_is_a_test.wav', 'r')

params = spf.getparams()
framerate= params[2]
x = spf.readframes(-1)
x = np.fromstring(x, 'Int16')

plt.title('This is a test')
plt.plot(x)
plt.show()

Pxx, freqs, bins, im= plt.specgram(x, NFFT=256, Fs=framerate, cmap='jet',
noverlap=128)
plt.ylabel('Frecuencia [Hz]')
plt.xlabel('Tiempo [seg]')
```



# Definición de procesado digital de señal

## Ejemplo en Python (funciones):

```
import matplotlib.pyplot as plt
import numpy as np
import wave

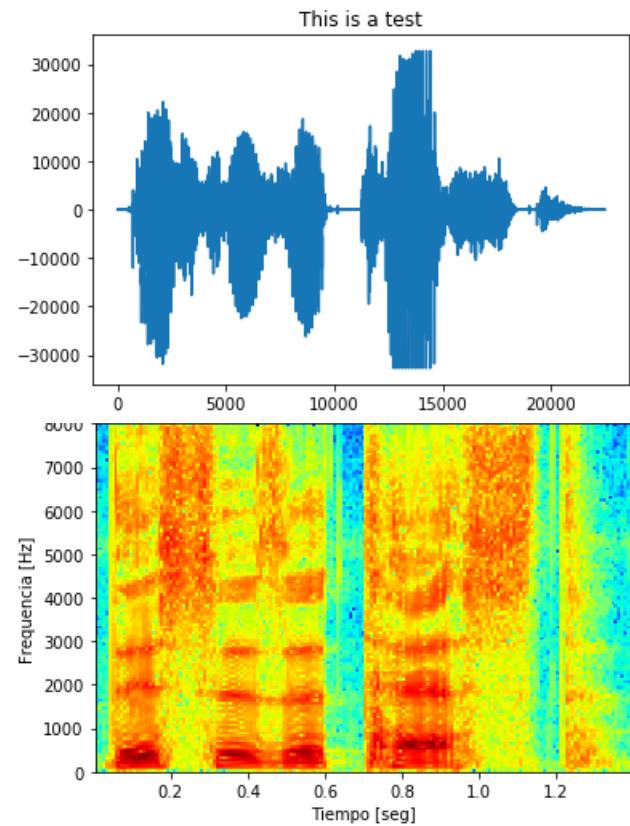
def leer_wave(filename):
    spf = wave.open(filename, 'r')
    params = spf.getparams()
    framerate= params[2]
    x = spf.readframes(-1)
    x = np.fromstring(x, 'Int16')
    plt.title(filename)
    plt.plot(x)
    plt.show()
    return x, framerate

def representa_espectrograma(x,NFFT,Fs,nooverlap):
    Pxx, freqs, bins, im= plt.specgram(x, NFFT=NFFT, Fs=Fs, cmap='jet', nooverlap=nooverlap)
    plt.ylabel('Frecuencia [Hz]')
    plt.xlabel('Tiempo [seg]')
    return Pxx, freqs

x, framerate= leer_wave('This_is_a_test.wav')
representa_espectrograma(x,256,framerate,128)
```

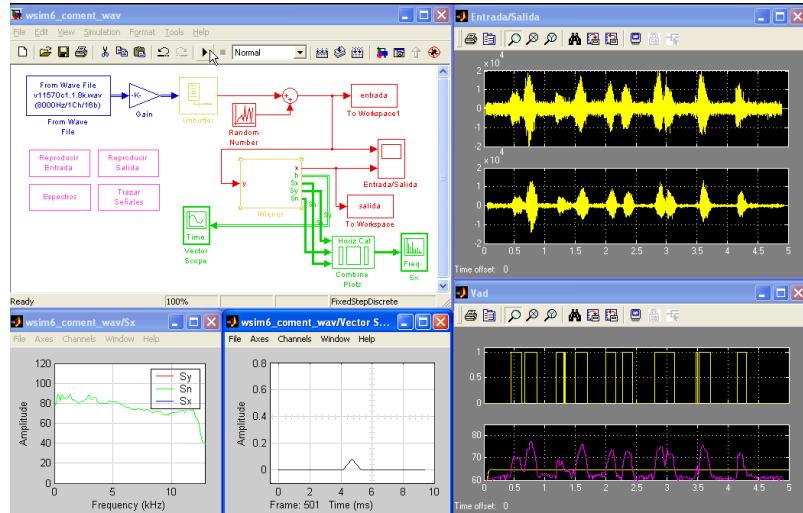


wav\_speech\_spectrogram\_functions.py

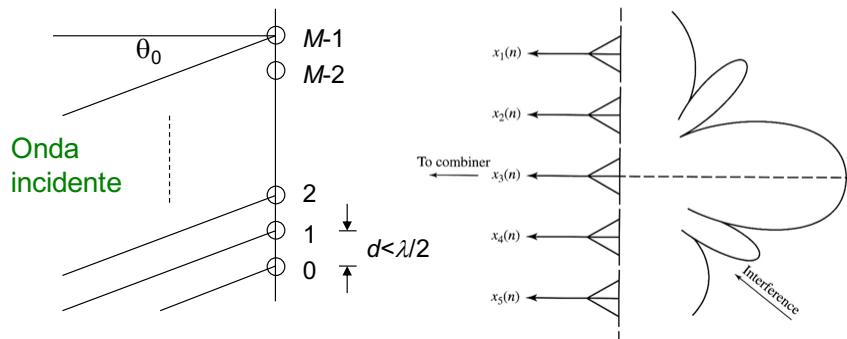


# Definición de procesado digital de señal

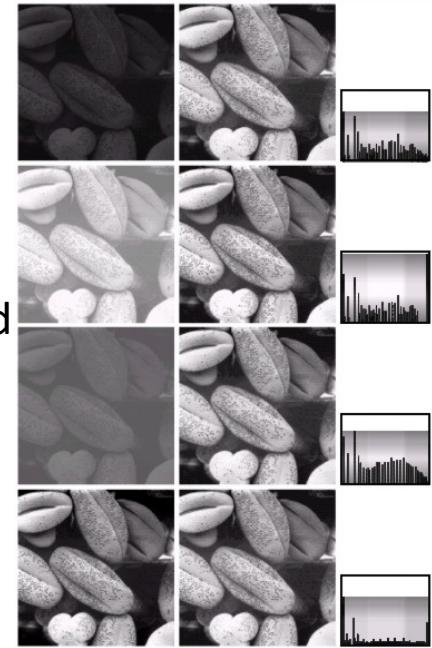
## ■ Transformación y manipulación



Filtrado de ruido en señales de voz/audio



Eliminación de **interferencias** en señales de radar por medio de **arrays de antenas**



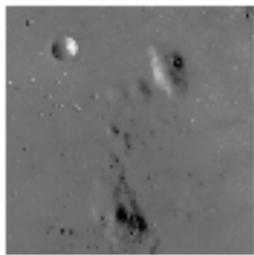
Ecualización del  
nivel de intensidad  
en imágenes

Separación de **señales mezcladas** (suma,  
multiplicación, convolución, ...)

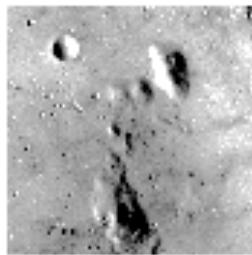
- Reconocimiento robusto de voz
- ECG fetal

# Definición de procesado digital de señal

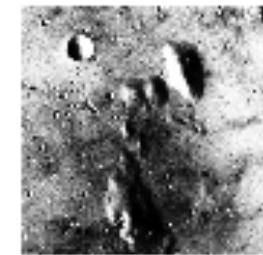
Low contrast image



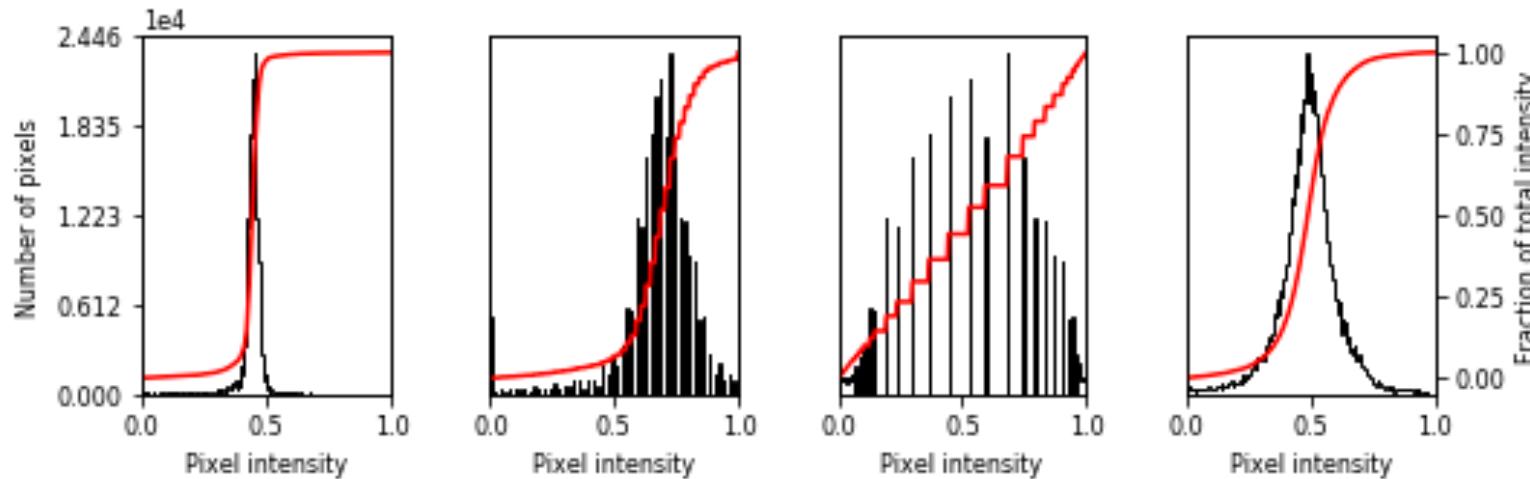
Contrast stretching



Histogram equalization



Adaptive equalization



`histogram_equalization.py`

# Definición de procesado digital de señal

## Realce del contraste (imágenes):

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

from skimage import data, img_as_float
from skimage import exposure

matplotlib.rcParams['font.size'] = 8

def plot_img_and_hist(img, axes, bins=256):
    img = img_as_float(img)
    ax_img, ax_hist = axes
    ax_cdf = ax_hist.twinx()

    # Visualización de imagen
    ax_img.imshow(img, cmap=plt.cm.gray)
    ax_img.set_axis_off()
    ax_img.set_adjustable('box')

    # Visualización del histograma
    ax_hist.hist(img.ravel(), bins=bins, histtype='step', color='black')
    ax_hist.ticklabel_format(axis='y', style='scientific', scilimits=(0, 0))
    ax_hist.set_xlabel('Pixel intensity')
    ax_hist.set_xlim(0, 1)
    ax_hist.set_yticks([])

    # Visualización de la distribución acumulada
    img_cdf, bins = exposure.cumulative_distribution(img, bins)
    ax_cdf.plot(bins, img_cdf, 'r')
    ax_cdf.set_yticks([])

    return ax_img, ax_hist, ax_cdf

img = data.moon()
```

histogram\_equalization.py

```
# Realce de contraste
p2, p98 = np.percentile(img, (2, 98))
img_rescale = exposure.rescale_intensity(img, in_range=(p2, p98))

# Ecualización
img_eq = exposure.equalize_hist(img)

# Ecualización adaptativa
img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)

# Resultados
fig = plt.figure(figsize=(8, 5))
axes = np.zeros((2,4), dtype=np.object)
axes[0,0] = fig.add_subplot(2, 4, 1)
for i in range(1,4):
    axes[0,i] = fig.add_subplot(2, 4, 1+i, sharex=axes[0,0], sharey=axes[0,0])
for i in range(0,4):
    axes[1,i] = fig.add_subplot(2, 4, 5+i)

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img, axes[:, 0])
ax_img.set_title('Imagen de bajo contraste')

y_min, y_max = ax_hist.get_ylimits()
ax_hist.set_ylabel('Número de pixels')
ax_hist.set_yticks(np.linspace(0, y_max, 5))

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_rescale, axes[:, 1])
ax_img.set_title('Realce de contraste')

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_eq, axes[:, 2])
ax_img.set_title('Ecualización de histograma')

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_adapteq, axes[:, 3])
ax_img.set_title('Ecualización adaptativa')

ax_cdf.set_ylabel('Fracción de la intensidad total')
ax_cdf.set_yticks(np.linspace(0, 1, 5))

# prevent overlap of y-axis labels
fig.subplots_adjust(wspace=0.4)
plt.show()
```

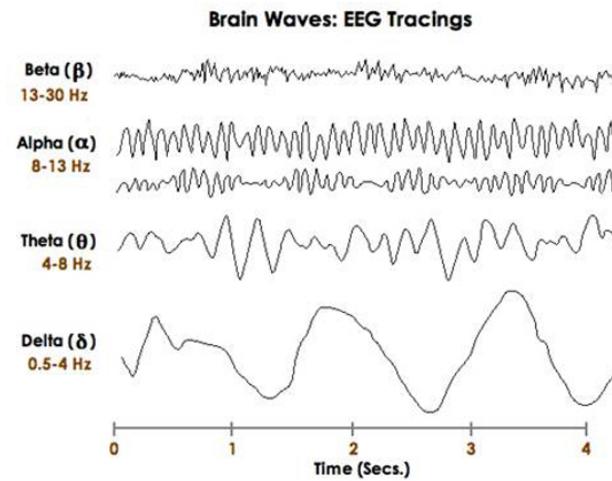


# Definición de procesado digital de señal

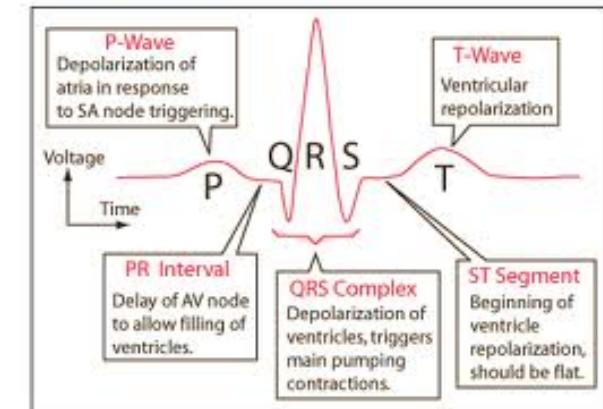
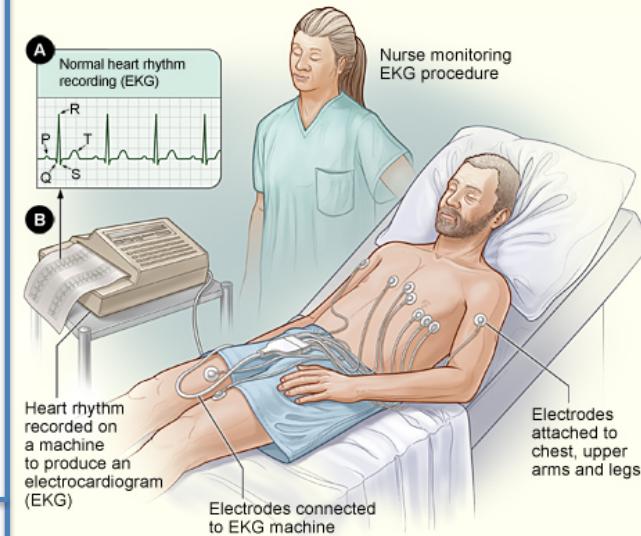
## ■ Extraer o procesar información

- **Electrocardiografía (ECG)**
  - Actividad eléctrica del corazón
  - Diagnóstico de enfermedades cardiovasculares
- **Electroencefalografía (EEG)**
  - Exploración neurofisiológica de la actividad bioeléctrica cerebral
  - Estudios de epilepsia, encefalopatía, tumores, demencia, traumatismos, cefaleas, etc.

## Electroencefalograma



## Electrocardiograma



# Definición de procesado digital de señal

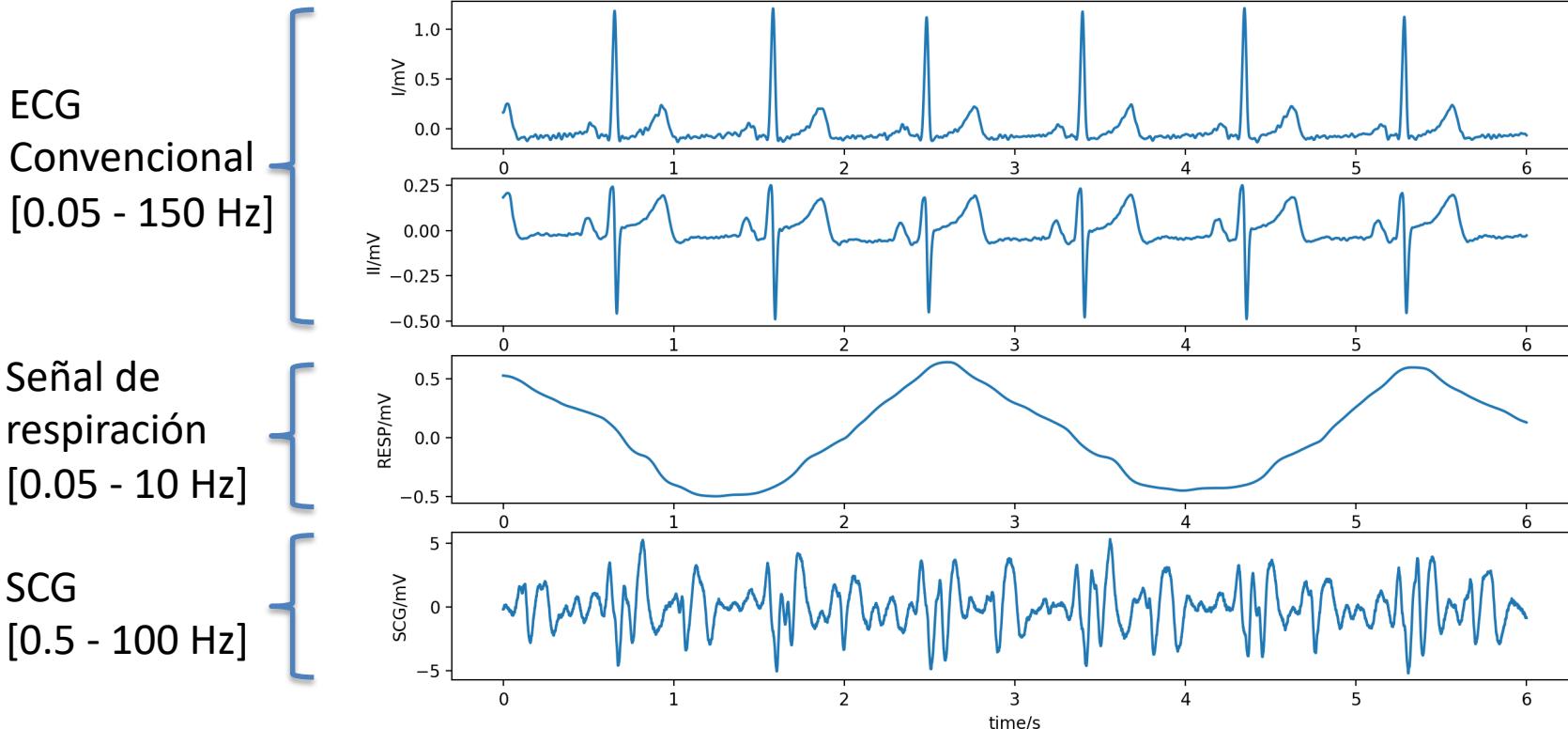
- Bases de datos de bioseñales: **Polisomnografía**

reading\_physionet.py



- PhysioNet: <http://www.physionet.org>

```
import wfdb  
sig, fields= wfdb.rdsamp('b001')  
wfdb.plotwfdb(sig[1:30000,:], fields, title='Record b001 from Physionet CEBS database')
```



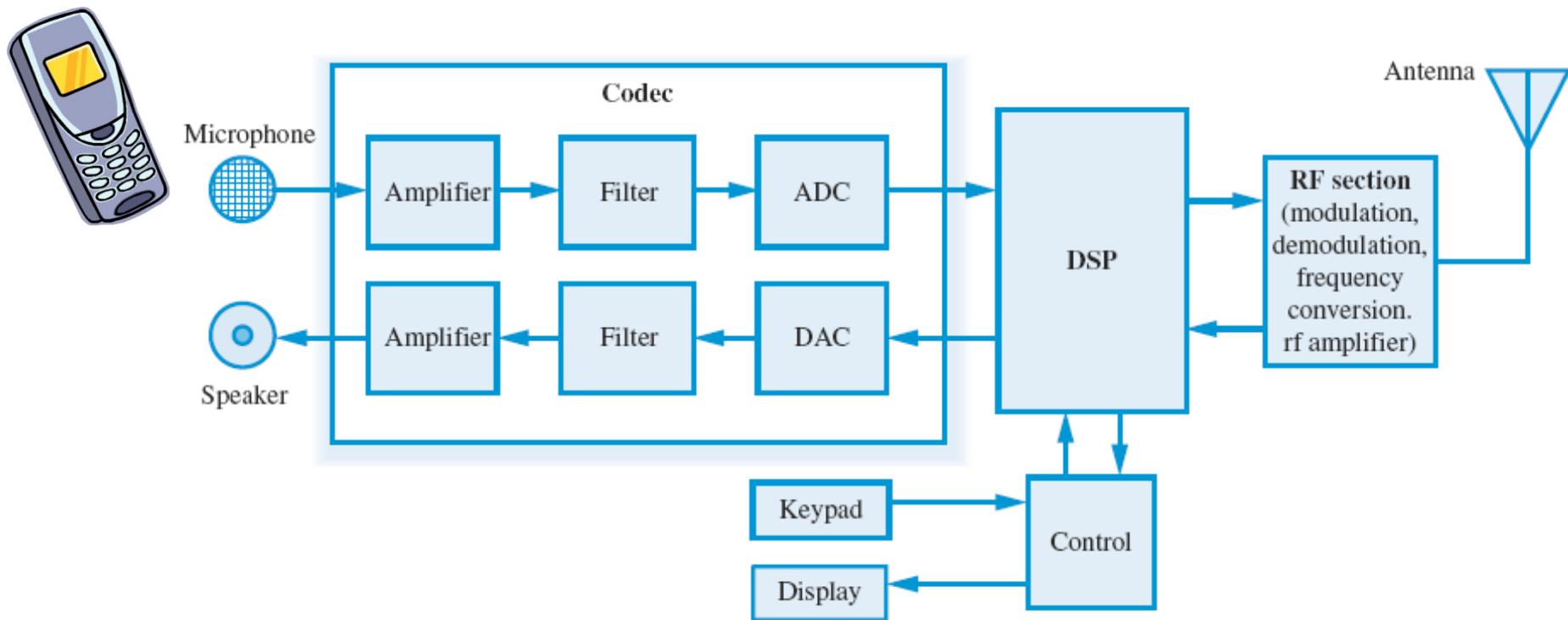
SCG (seismocardiografía): recording of body vibrations induced by the heart beat.

# Definición de procesado digital de señal

- Extraer o procesar información

- Codificación: voz/audio, imágenes, video.

- DSPs como núcleo de todos los teléfonos móviles modernos

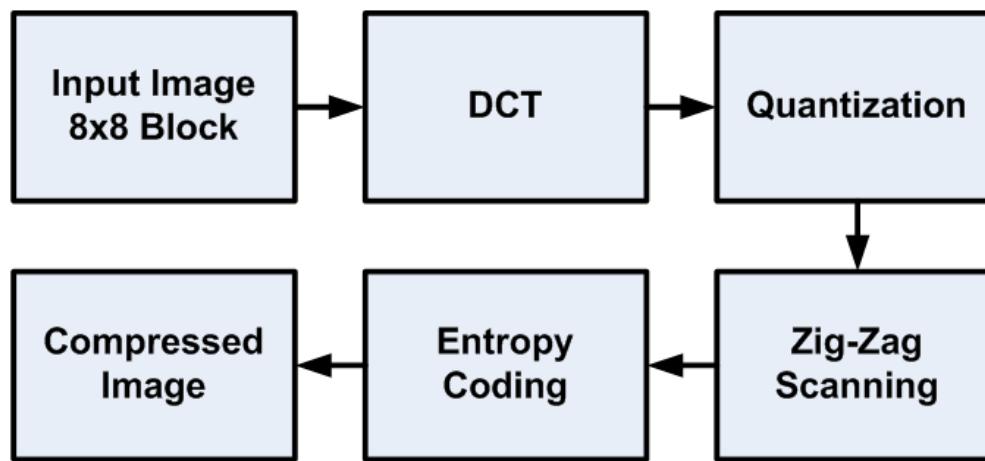


Telefonía móvil: Codificación/decodificación de la señal de voz.

# Definición de procesado digital de señal

## ■ Extraer o procesar información

- Codificación: voz/audio, imágenes, video.
  - Compresión de imágenes digitales JPEG
  - Codificadores de audio y video MPEG



# Definición de procesado digital de señal



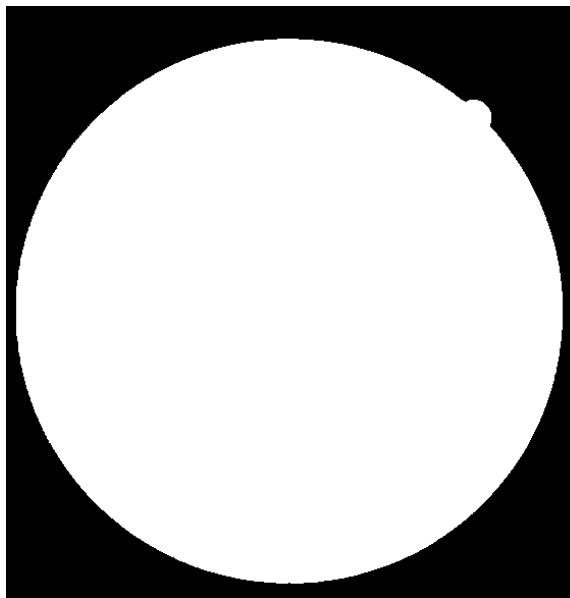
<http://scikit-image.org/>

- Segmentación de venas en imágenes de la retina mediante la librería scikit-image de Python.

21\_training.tif



21\_training\_mask.gif



21\_manual1.gif

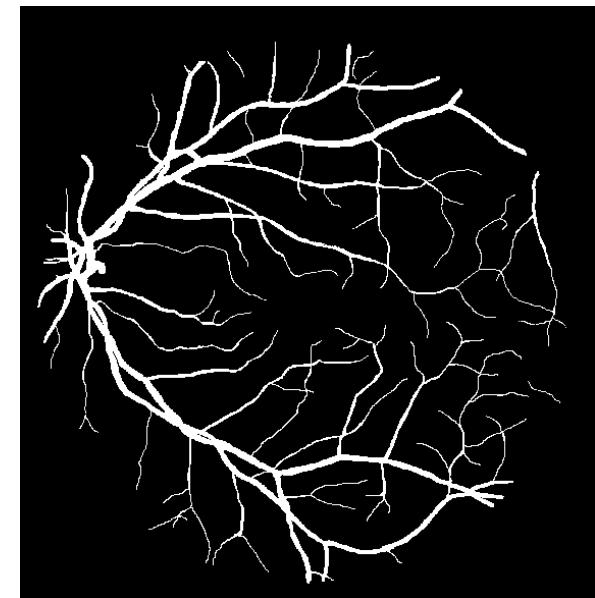


Imagen de entrada

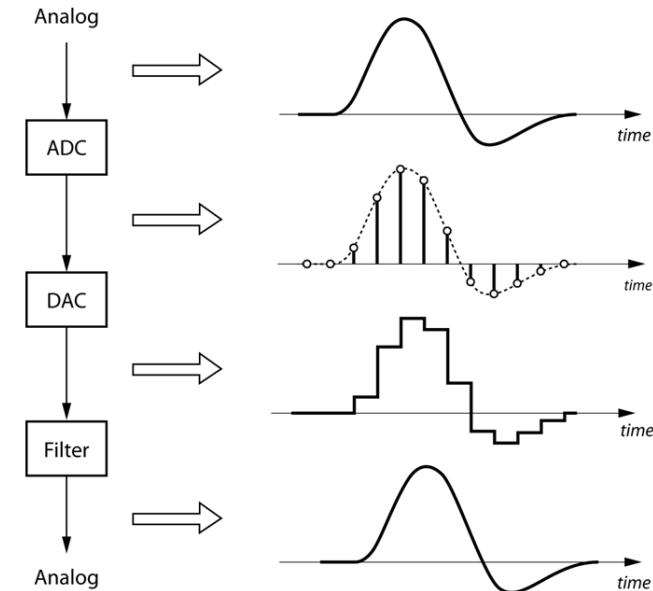
Imagen de salida

# 1. Introducción

- DSP ofrece numerosas ventajas frente al procesado analógico.
- Campo en continua expansión por las nuevas aplicaciones:
  - televisión digital,
  - radio digital,
  - comunicaciones inalámbricas,
  - sensores MEMs,
  - atención médica,
  - diagnóstico, etc.
- No es la solución apropiada para todos los problemas de procesado de señal (ej.: señales de gran ancho de banda).
  - Mejor solución cuando:
    - Disponibilidad del hardware a la velocidad requerida.

# 1. Introducción

- Señales analógicas por naturaleza.
- Señales representan magnitudes físicas.
- Sensores traducen magnitudes físicas en señales eléctricas
- Etapas de un sistema de procesado digital de señales:
  - Conversión analógico/digital (**ADC**)
  - Procesador digital de señal (**DSP**)
  - Conversión digital/analógico (**DAC**)



## 2. Definición y clasificación de señales

Def.: Una **señal** es una función de una o varias variables independientes que almacena información de una magnitud física

- Ej.:  $x(t)$ ,  $I(x,y)$  (señal de voz, ECG, EEG, imagen, etc.)

**Clasificación (4 criterios):**

- A. Atendiendo al **número de fuentes** que la generan
- B. Atendiendo al **número de variables independientes**
- C. Atendiendo al **valor continuo o discreto** de las variables independientes y de la propia señal
- D. Atendiendo al carácter **determinista o aleatorio** de la señal

## 2. Definición y clasificación de señales

Def.: Una **señal** es una función de una o varias variables independientes que almacena información de una magnitud física

- Ej.:  $x(t)$ ,  $I(x,y)$  (señal de voz, ECG, EEG, imagen, etc.)

**Clasificación (4 criterios):**

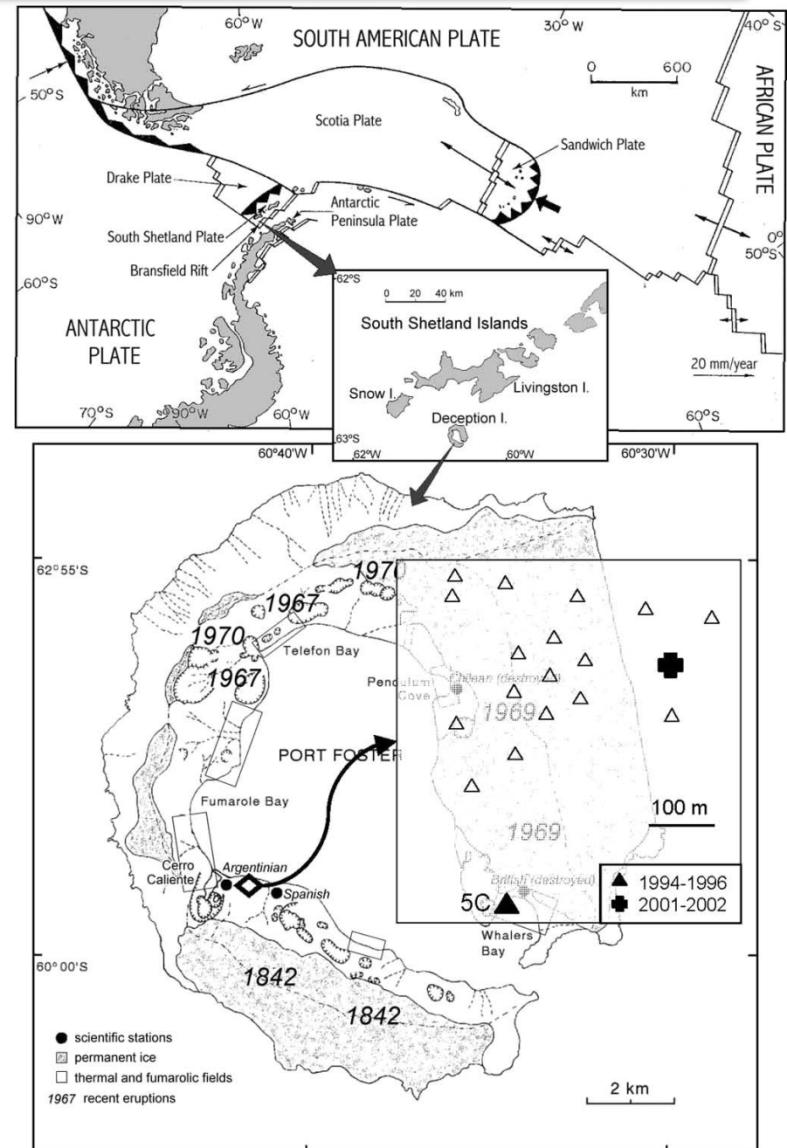
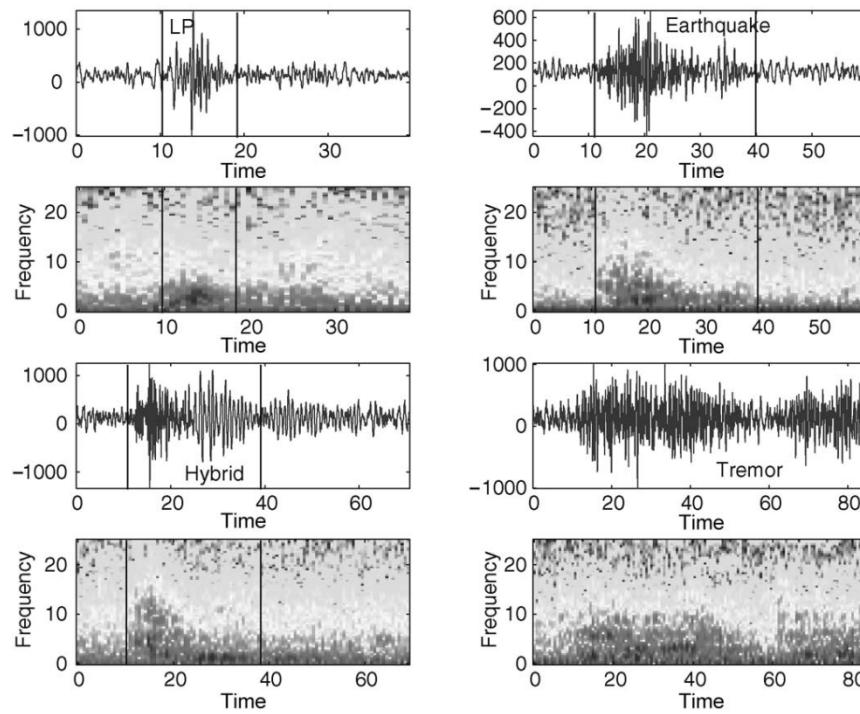
A. Atendiendo al **número de fuentes** que la generan:

- **Monocanal:**
  - Generada (registrada) por una única fuente (sensor).
- **Multicanal:**
  - Generada (registrada) por múltiples fuentes (sensores).

# Ejemplos de señales multicanal

## ■ Ejemplo: señales multicanal

- Array sísmico desplegado en la isla Decepción (Antártida) para estudiar actividad eruptiva.



## 2. Definición y clasificación de señales

**Clasificación (criterios):**

B. Atendiendo al **número de variables independientes**

○ **Unidimensional:**

$$x(t)$$

- La señal depende de una única variable independiente

○ **Multidimensional:**

$$I(x,y, \dots)$$

- La señal depende de dos o más variables independientes

■ **Ejemplos:**

• Imagen (bidimensional):

$$I(x,y)$$

• TV (blanco y negro):

$$I(x,y,t) \text{ (brillo)}$$

• TV (color-RGB):  $[I_r(x,y,t) \ I_g(x,y,t) \ I_b(x,y,t)]$  Tridimensional, tricanal

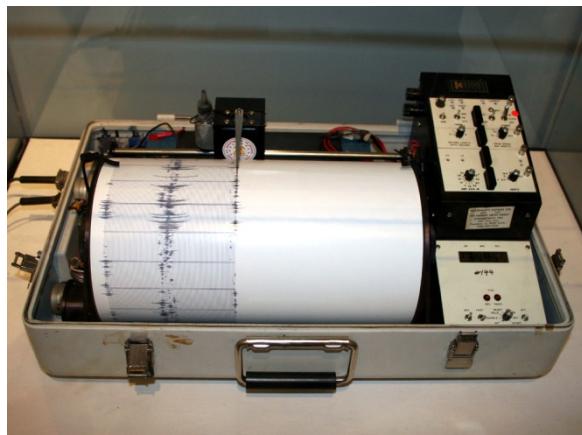
# Ejemplos de señales 1D



Señal de voz



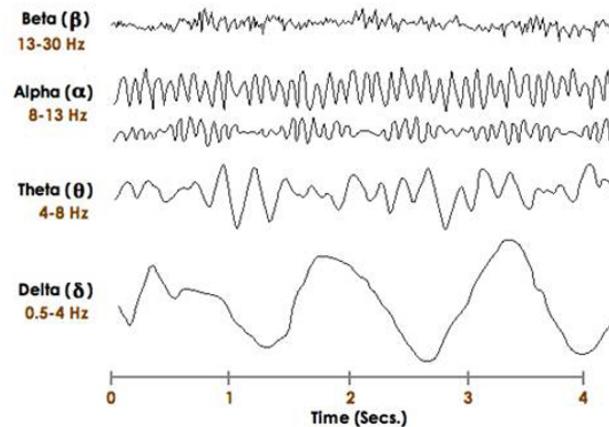
Sismógrafo o sismómetro



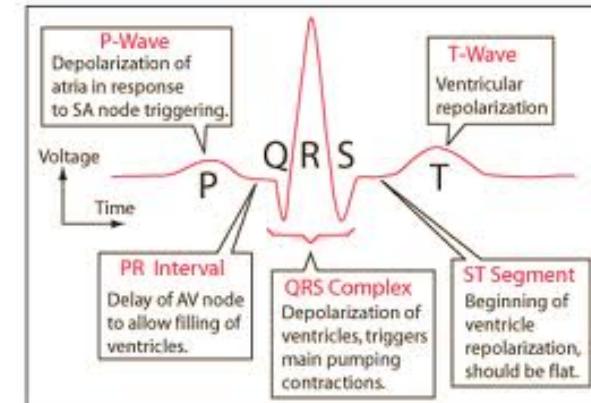
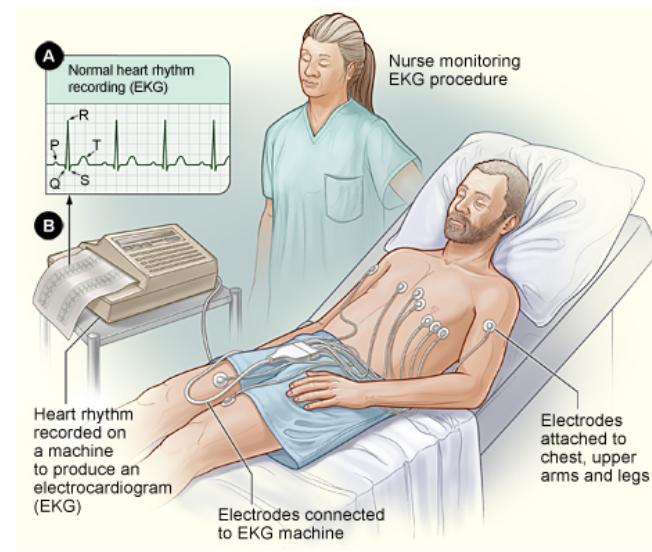
Electroencefalograma



Brain Waves: EEG Tracings



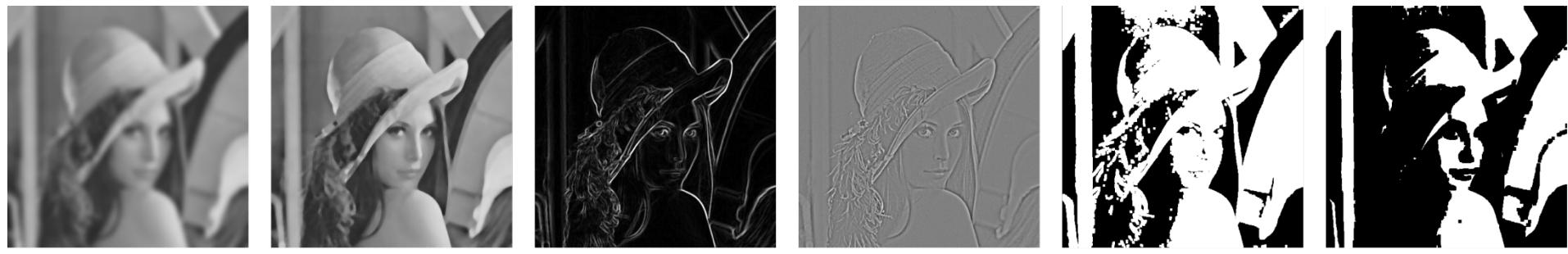
Electrocardiograma



# Ejemplos de señales multidimensionales (2D)

## ■ Ejemplos: Señales multidimensionales

- Procesado de **imagen** (filtrado, operaciones morfológicas)



Blur

Median

Edge-Detect

High-Pass

Dilate

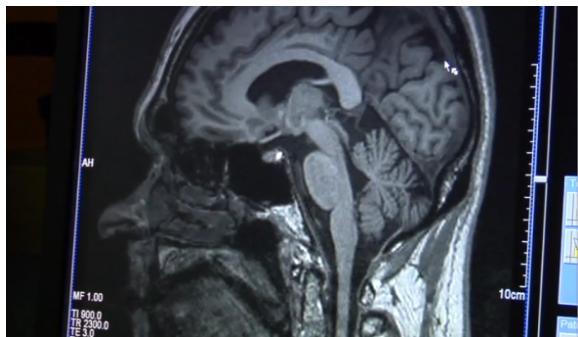
Erode

- Estimación de movimiento en señal de **video** en color (señal tridimensional ( $x, y, t$ ) y tricanal (R, G, B))

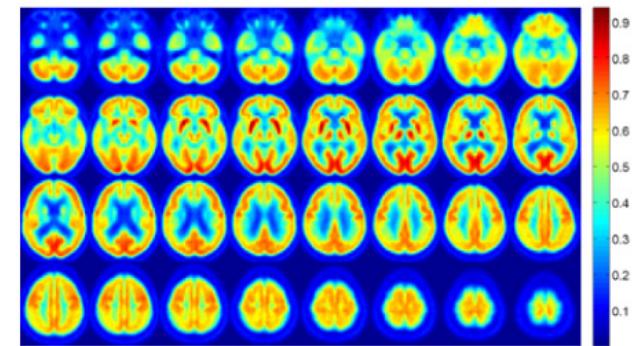
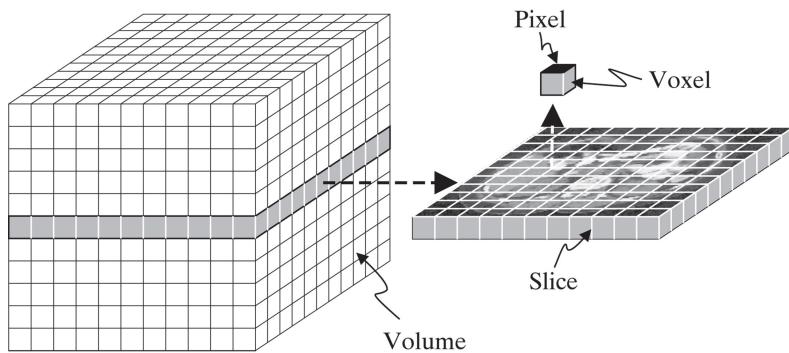


# Ejemplos de señales multidimensionales (3D)

Resonancia magnética



Tomografía de emisión de positrones

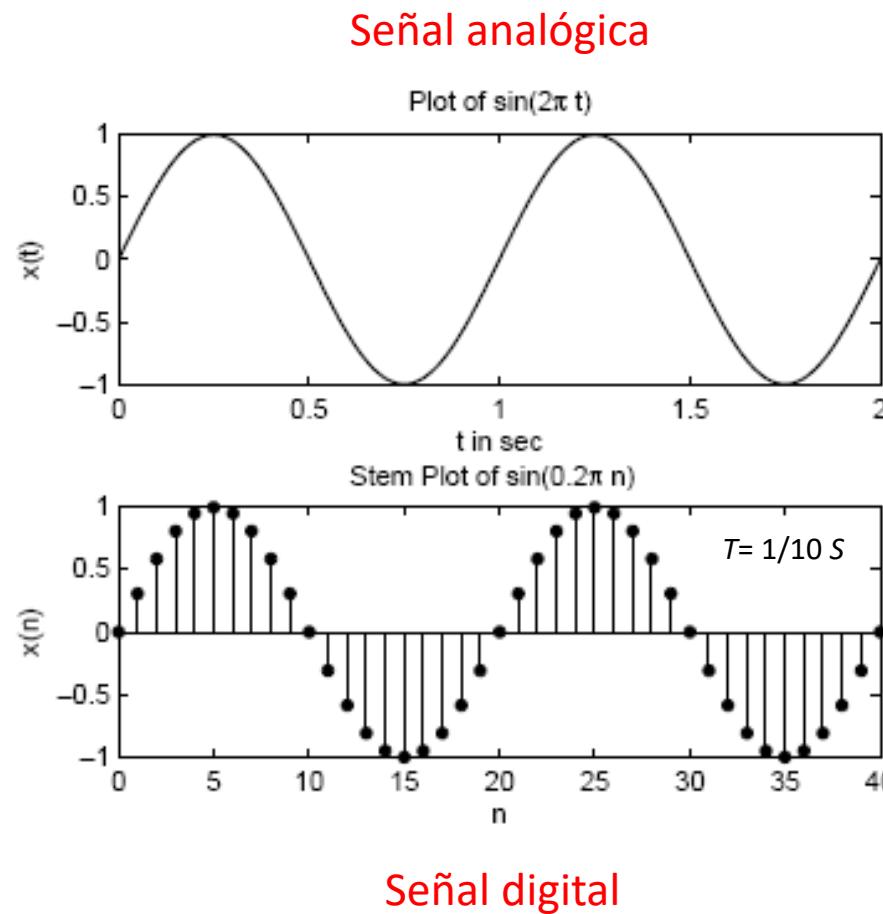


# Clasificación de señales (continuación)

Clasificación (criterios):

C. Atendiendo al **valor continuo o discreto** de las variables independientes y de la propia señal

- Señales en **tiempo continuo** o analógicas
  - Valor continuo (señal analógica)
  - Valor discreto
- Señales en **tiempo discreto**
  - Valor continuo
  - Valor discreto (señal digital)



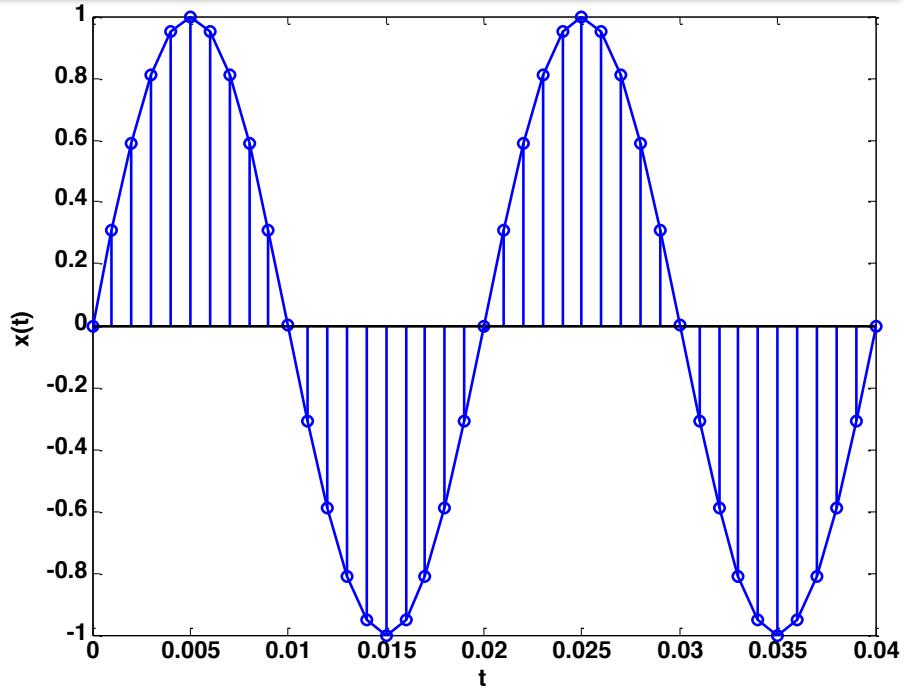
# Clasificación de señales (continuación)

## Clasificación (criterios):

D. Atendiendo al carácter **determinista o aleatorio** de la señal

- **Deterministas:** Se pueden definir mediante una forma matemática explícita, un conjunto de datos o una regla bien definida.

Ejemplo:  $x(t) = \sin(\omega_0 t)$



```
f0= 50;  
w0= 2*pi*f0;  
Fs= 1000;  
Ts= 1/Fs;  
t=0:Ts:0.04;  
x= sin(w0*t);  
stem(t,x);  
hold on; plot(t,x); hold off;  
xlabel('t');  
ylabel('x(t)');
```

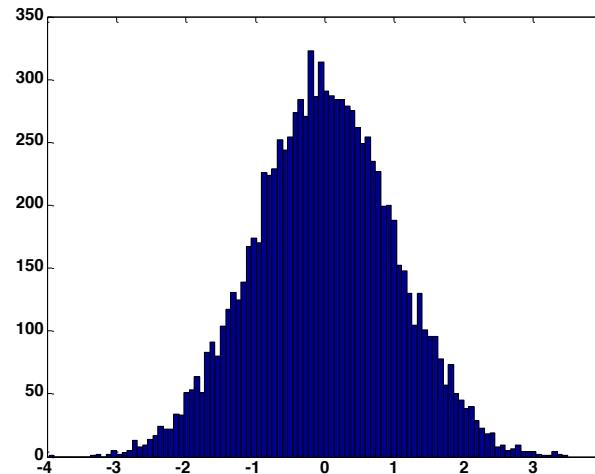
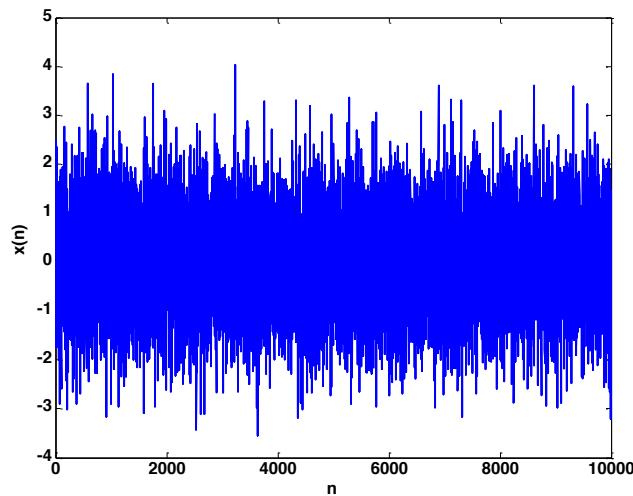
# Clasificación de señales (continuación)

## Clasificación (criterios):

D. Atendiendo al carácter **determinista o aleatorio** de la señal

- **Aleatorias:** No pueden describirse con un grado de precisión razonable mediante fórmulas matemáticas explícitas.
  - Ej.: Señales sísmicas, señales de voz, el ruido.
  - Se describen mediante técnicas estadísticas (histograma, distribución de probabilidad).

**Ejemplo:** Ruido aleatorio de media nula y varianza unidad

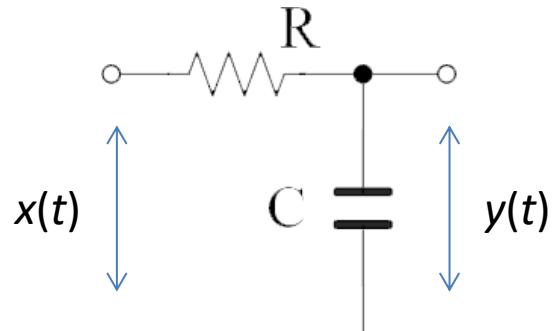


```
x= randn(10000,1);
mean(x)
var(x)

subplot(1,2,1);
plot(x);
xlabel('n');
ylabel('x(n)');
subplot(1,2,2);
hist(x,100);
```

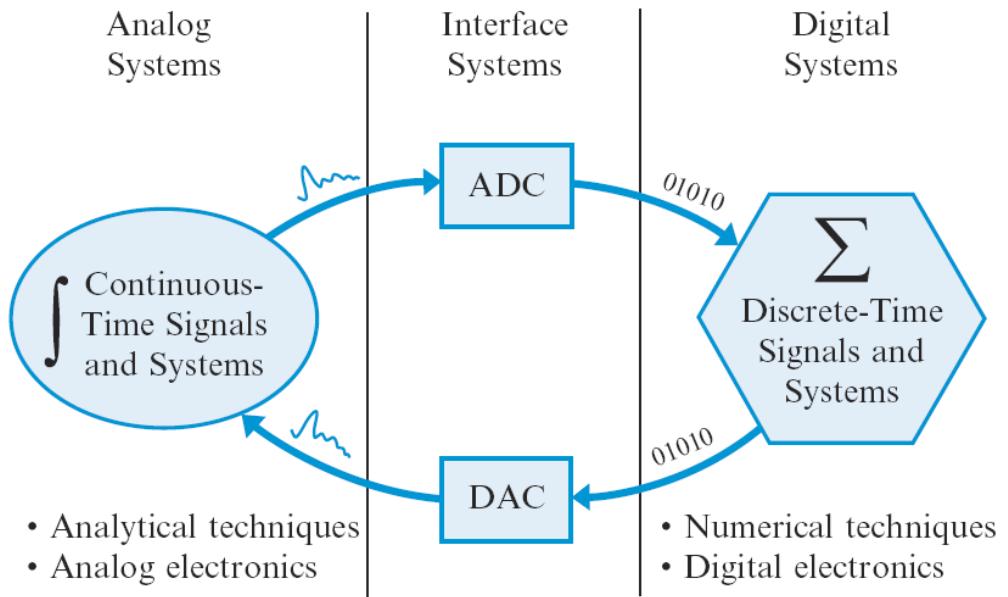
### 3. Ventajas del procesado digital de señales

#### Filtro analógico de primer orden

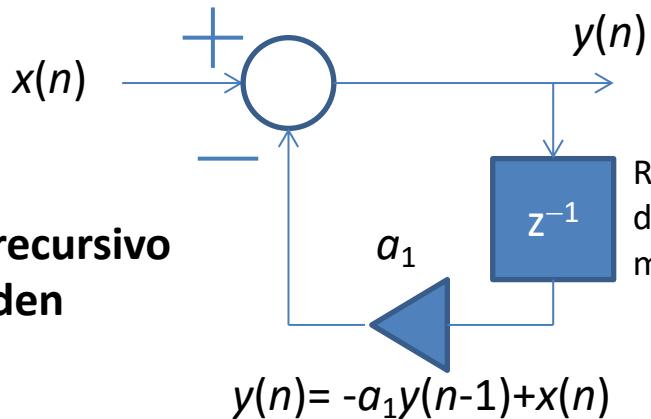


$$RC \frac{dy}{dt} + y(t) = x(t)$$

$$H(s) = \frac{1}{1 + RCs}$$



#### Filtro digital recursivo de primer orden



$$y(0) = 0$$

```
for n=1 to Nsamples  
    y(n) = -a1*y(n-1) + x(n)  
end
```

$$H(z) = \frac{1}{1 + a_1 z^{-1}}$$

### 3. Ventajas del procesado digital de señales

#### ■ Flexibilidad

- La **función** de un sistema se puede **actualizar o modificar** reprogramando el software.
- Ej.: Cámara digital: actualización firmware.

#### ■ Reproducibilidad

- La **operación** de dos unidades distintas es **idéntica**.
- Los sistemas analógicos no tienen este comportamiento debido a la **tolerancia** de los componentes.

#### ■ Seguridad

- Los microprocesadores **no se deterioran** con el **tiempo** como lo hacen los componentes analógicos.
- Su operación no cambia con las **condiciones ambientales**.

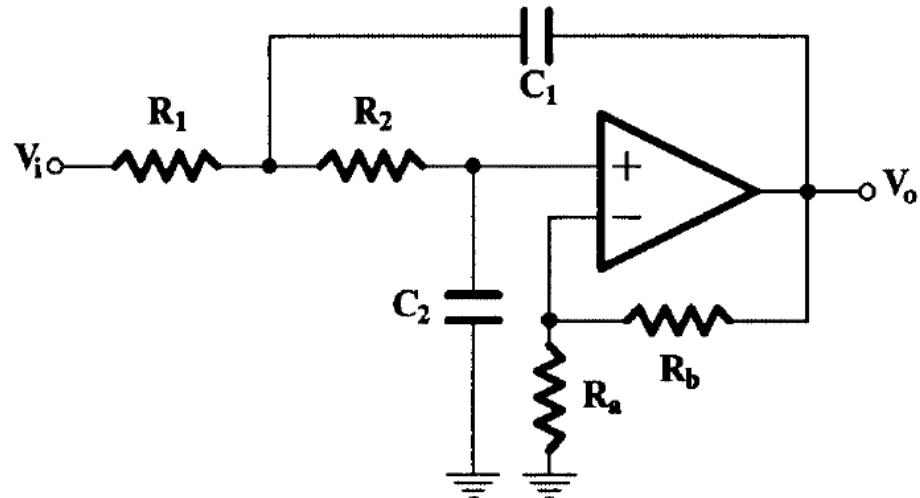
#### ■ Complejidad

- Permite realizar **operaciones más sofisticadas** (reconocimiento de voz, imágenes).
- Existen **algoritmos que no tienen su equivalente analógico** (códigos de corrección de errores, transmisión de datos y almacenamiento, compresión de datos, filtros de fase lineal).

# Ej.: Filtrado digital frente a analógico (1/5)

- Un filtro analógico se implementa por medio de amplificadores operacionales y componentes pasivos (resistencias, condensadores).
- Función de transferencia:

Filtro paso-baja de Sallen-Key:

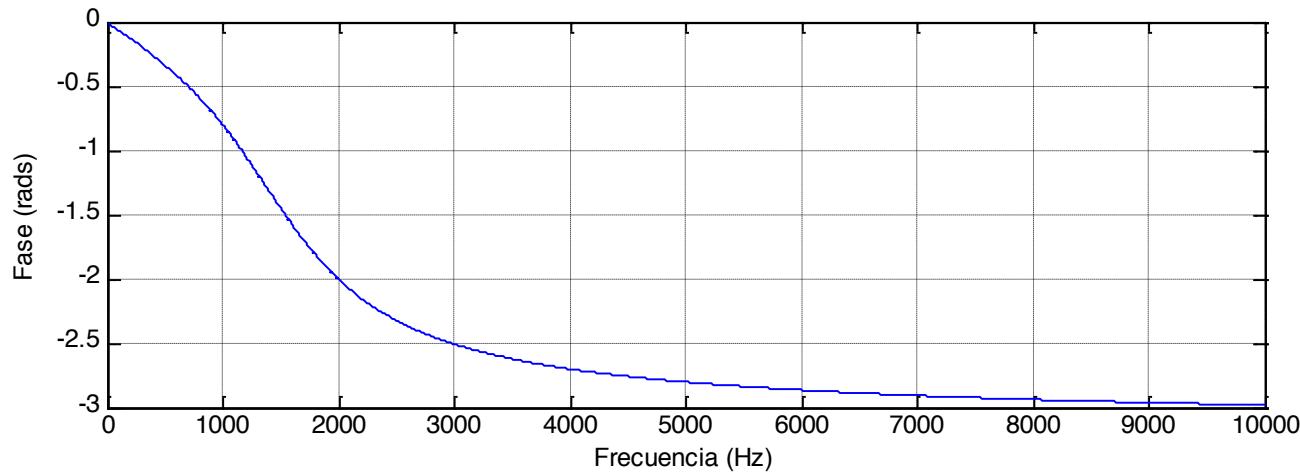
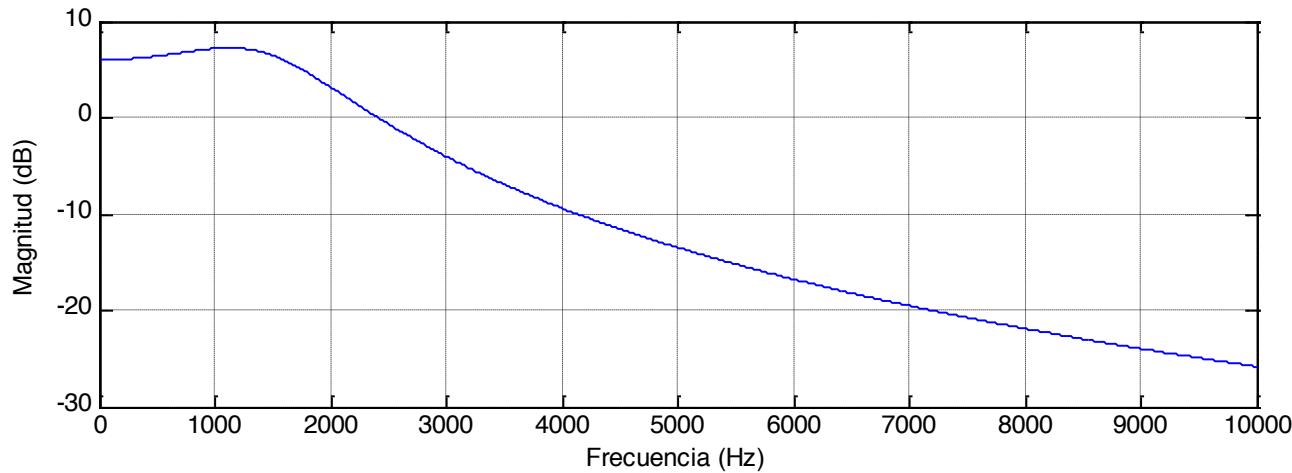


$$H(s) = \frac{V_0}{V_i} = \frac{G/(C_1 C_2 R_1 R_2)}{s^2 + \left( \frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} + \frac{1-G}{R_2 C_2} \right)s + \frac{1}{R_1 R_2 C_1 C_2}}$$

$$G = 1 + \frac{R_b}{R_a} \quad f_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

# Ej.: Filtrado digital frente a analógico (2/5)

- Respuesta en frecuencia:



# Ej.: Filtrado digital frente a analógico (3/5)

## ■ Inconvenientes:

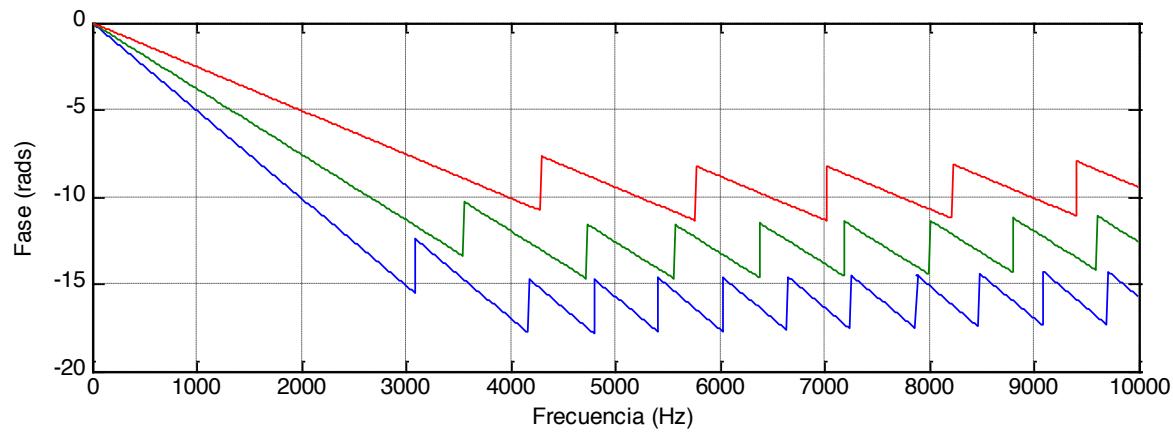
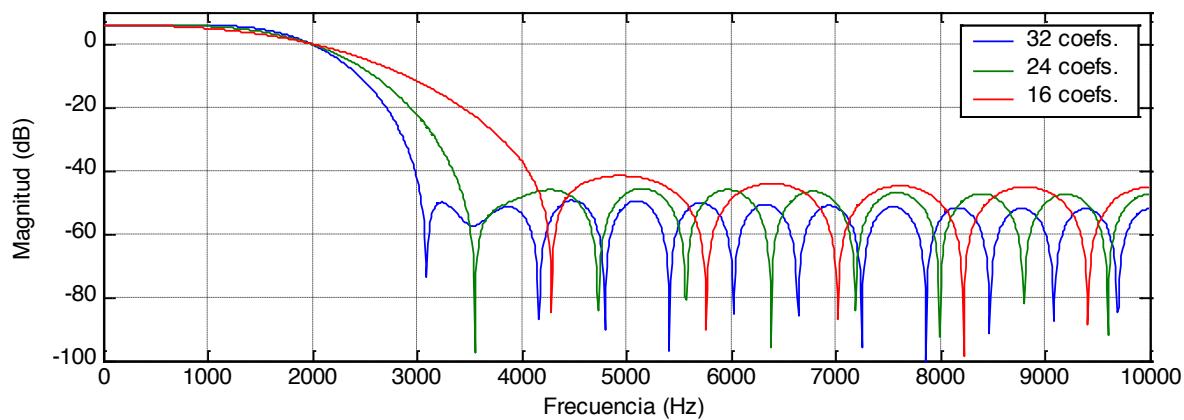
- La **respuesta** del filtro **depende** de los **componentes pasivos**.
  - El filtro **no es reproducible** con total exactitud debido a la tolerancia de los componentes.
  - La respuesta del filtro puede variar con las **condiciones ambientales**.
- Filtros de **orden superior** necesitan redes RC más complejas o conectar varios filtros en cascada.
- **Modificar la respuesta** del filtro exige la sustitución de los componentes pasivos.
- La frecuencia de operación queda **limitada por** la respuesta del **amplificador operacional**.
- La respuesta en **fase** es **no lineal**. Introduce un retardo en la señal que es variable con la frecuencia.

# Ej.: Filtrado digital frente a analógico (4/5)

- Respuesta en frecuencia de filtros digitales FIR

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

- Fácil diseño e implementación.
- Mayor atenuación.
- Mayor selectividad
- Fase lineal.
- Reproducibles.
- Actualizables (concepto de filtro adaptativo).



# Ej.: Filtrado digital frente a analógico (5/5)

- Algoritmo de filtrado digital FIR:
  - Puede ser fácilmente descrito en lenguaje C para su implementación en un procesador digital de señal.

The screenshot shows the Texas Instruments Code Composer Studio interface. The title bar reads "JC6211 DSK (Texas Instruments)/CPU\_0 - C6211 Code Composer Studio- fir\_simple.mak - [main.c]". The left sidebar displays the project structure under "Files": GEL files, Project (with "fir\_simple.mak" selected), DSP/BIOS Config, Include, Libraries, lnx.cmd, and Source (containing dsk\_init.c, low\_fir.c, main.c, and vectors.asm). The main window contains the C code for the "fir\_filter" function:

```
/*
Function:      fir_filter()
Description:  Sample by sample FIR filtering
*/
#pragma CODE_SECTION (fir_filter, ".iprog")
short fir_filter (short input)
{
    int i;
    short output;
    int Acc;

    for (i=N-1; i>0; i--)          /* Shift delay samples. */
        R_in[i]=R_in[i-1];
    R_in[0] = input;                /* Update most recent sample */

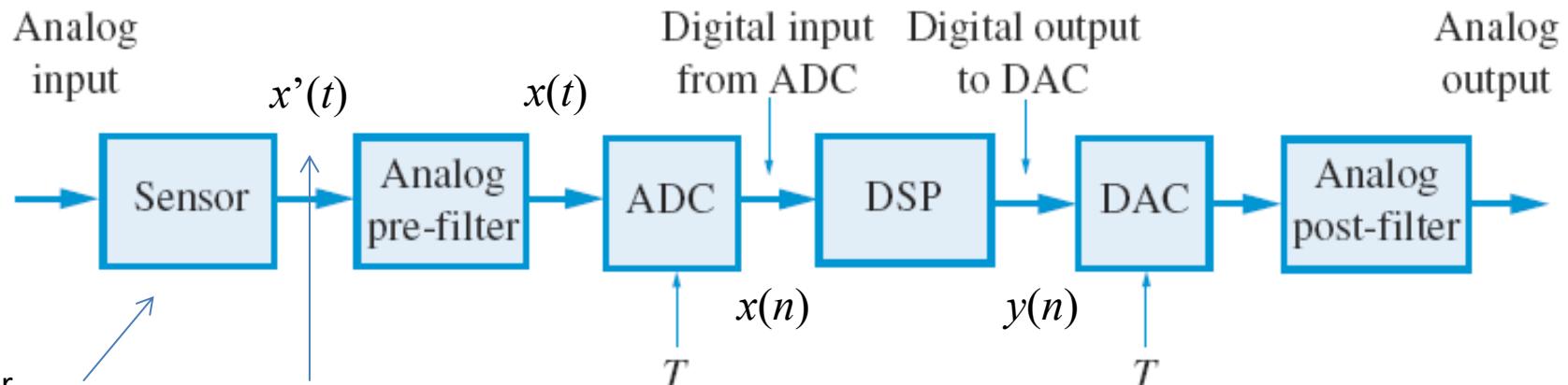
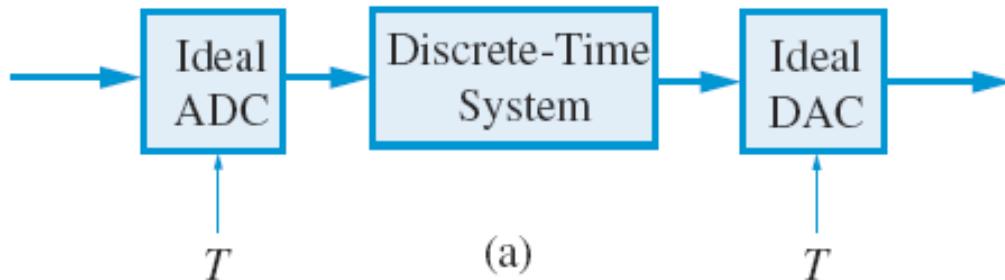
    Acc = 0;
    for (i=0; i<N; i++)
        Acc += (int)(short)h[i] * (int)(short)R_in[i]; /*Correct multiplies in C*/

    output = (short) (Acc>>15);   /* Cast output to 16-bits */

    return output;
}
```

The status bar at the bottom shows "DSP HALTED", "For Help, press F1", "Ln 142, Col 44", and "NUM".

# 4. Elementos de un sistema para DSP



Sensor  
transforma una  
magnitud física  
(presión,  
temperatura,  
sonido) en una  
señal eléctrica

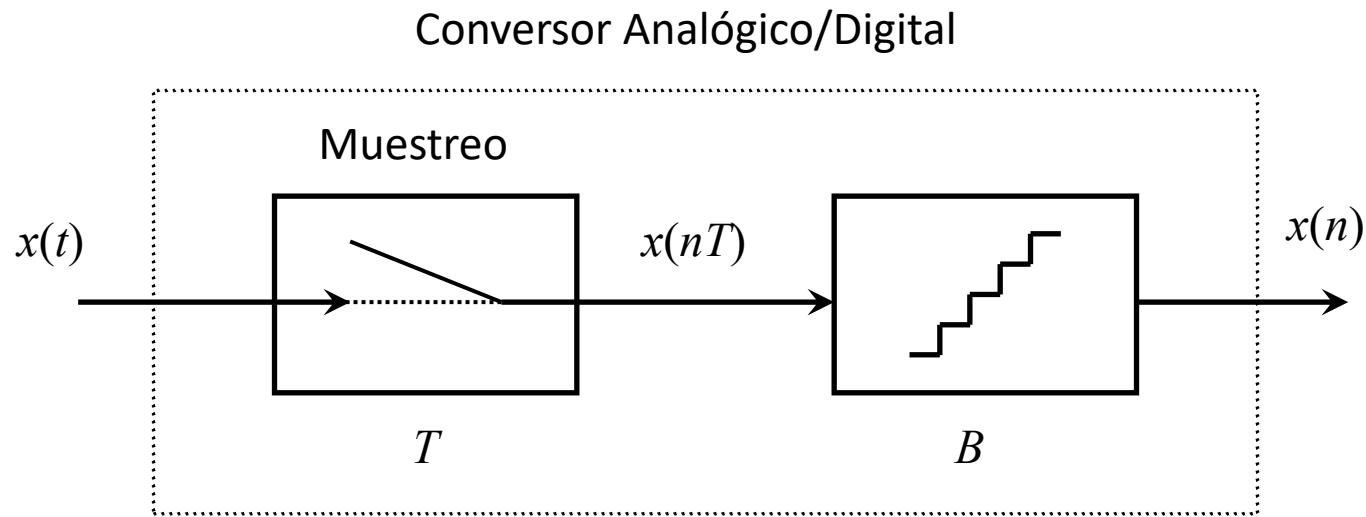
**La señal  $x'(t)$  recogida por medio del sensor necesita ser acondicionada.** Un amplificador adapta el rango dinámico de la señal  $x'(t)$  al rango dinámico del convertidor A/D.

(b)

**Control Automático de Ganancia (AGC).**  
En la práctica, resulta difícil seleccionar una ganancia  $G$  fija puesto que el nivel de  $x'(t)$  puede ser desconocido o variante con el tiempo. El AGC permite que  $G$  varíe con el tiempo.

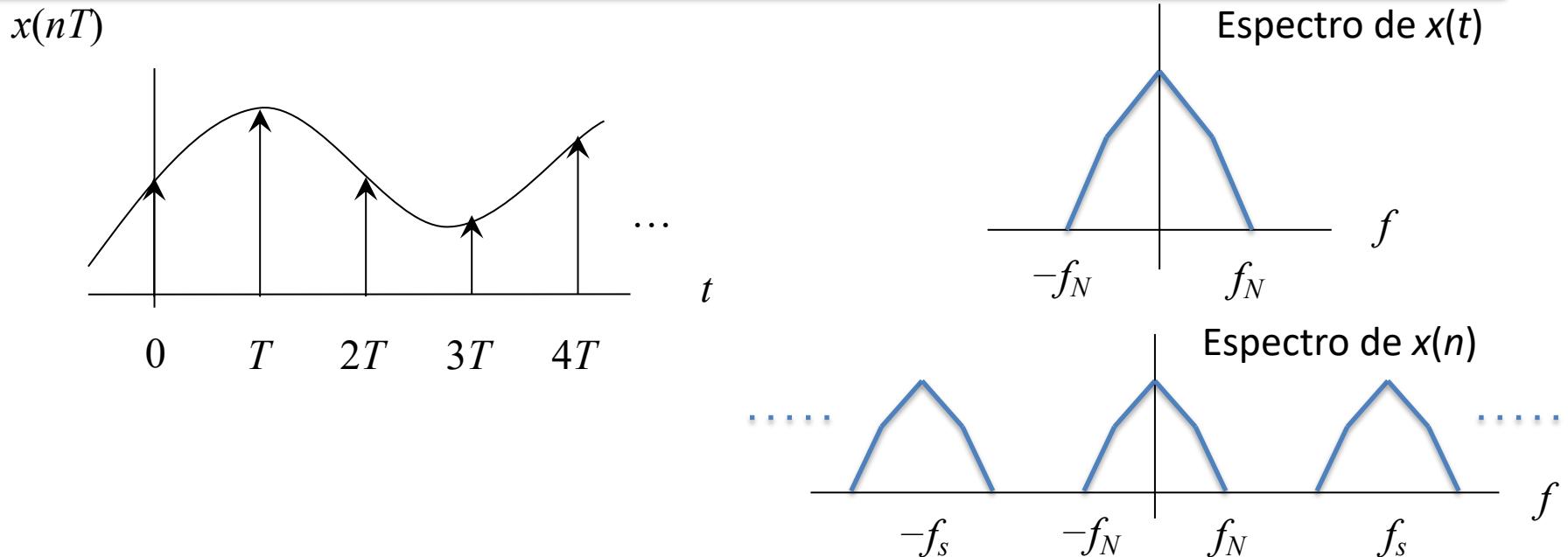
# Conversión A/D

- El convertidor A/D transforma la señal analógica  $x(t)$  en una secuencia digital  $x(n)$ .



- Consta de dos etapas:
  - **Muestreo (sample&hold):** se toma una muestra cada  $T$  segundos.
  - **Cuantización:** se aproxima la señal asignando un valor discreto a cada muestra. La salida se da codificada en binario con  $B$  bits de precisión.

# Muestreo



- **Teorema de muestreo.** Para que la señal muestreada  $x(nT)$  represente fielmente la señal analógica  $x(t)$ :
  - $x(t)$  debe ser una señal limitada en banda de ancho de banda  $f_N$ .
  - La **frecuencia de muestreo**  $f_s$  debe ser al menos el doble de la máxima frecuencia de la señal analógica  $x(t)$ .
$$f_s \geq 2f_N \quad (\text{frecuencia de Nyquist}, 2*f_N)$$
- Filtrado previo al muestreo.
  - Filtro anti-solapamiento (*anti-aliasing*).

# Ejemplo ilustrativo: Tono de 5 kHz

```
f0= 5000;  
Fsmin= 2*f0;
```

$$x(t) = \sin(2\pi f_0 t + \pi / 6)$$

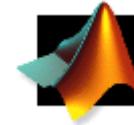
```
window= 256;  
noverlap= 128;  
nfft= 256;
```

```
%%  
Fs= 4*Fsmin;  
t=0:1/Fs:1;  
x= sin(2*pi*f0*t+pi/6);  
soundsc(x,Fs)
```

```
spectrogram(x,window,noverlap,nfft,Fs);
```

```
%%  
Fs= 2*Fsmin;  
t=0:1/Fs:1;  
x= sin(2*pi*f0*t+pi/6);  
soundsc(x,Fs)
```

```
spectrogram(x,window,noverlap,nfft,Fs);
```



ejemplo\_tono\_5\_khz\_muestreo.m

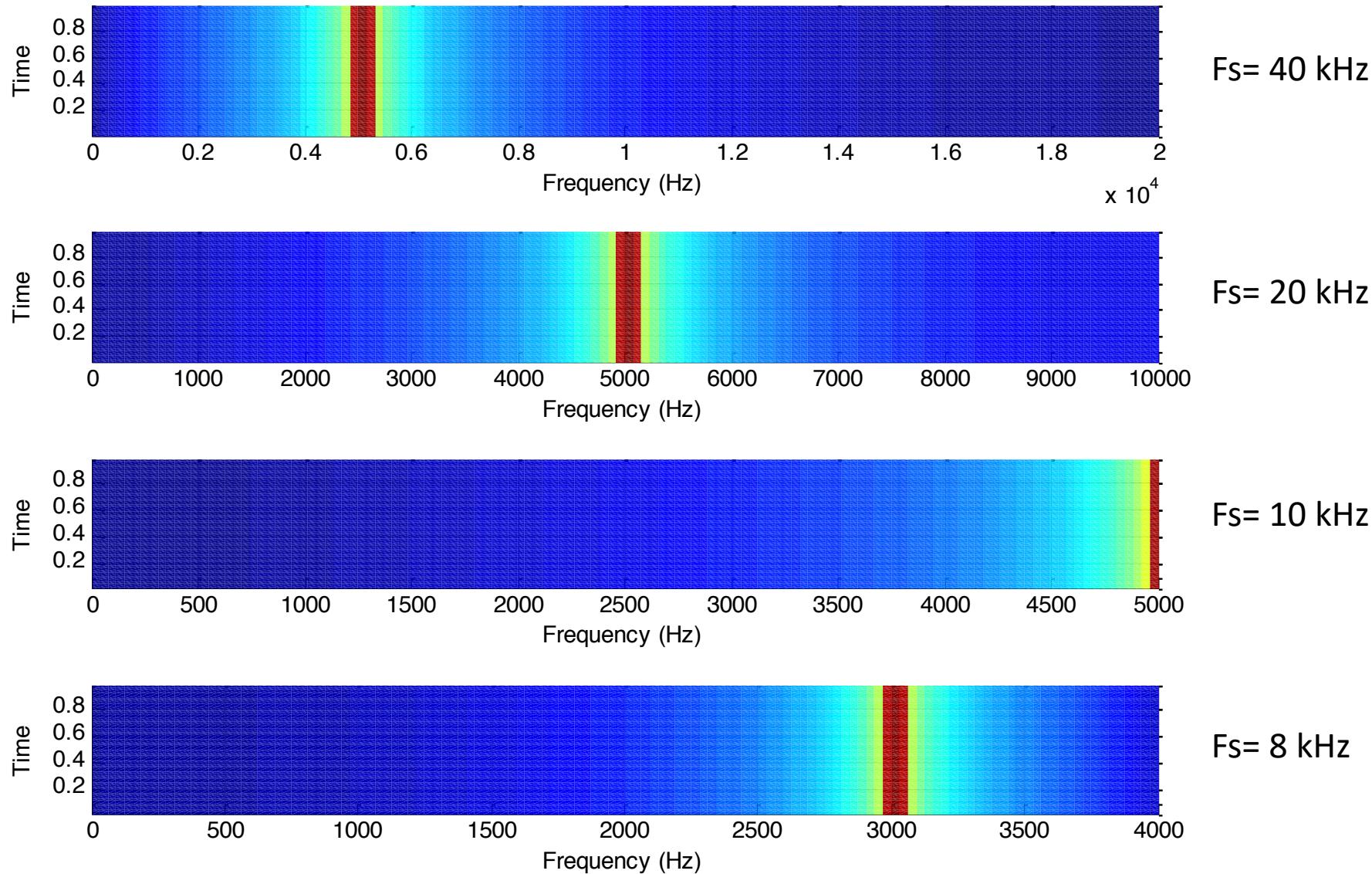
```
%%  
Fs= Fsmin;  
t=0:1/Fs:1;  
x= sin(2*pi*f0*t+pi/6);  
soundsc(x,Fs)
```

```
spectrogram(x,window,noverlap,nfft,Fs);
```

```
%%  
Fs= 4/5*Fsmin;  
t=0:1/Fs:1;  
x= sin(2*pi*f0*t+pi/6);  
soundsc(x,Fs)
```

```
spectrogram(x,window,noverlap,nfft,Fs);
```

# Ejemplo ilustrativo: Tono de 5 kHz



# Ejemplo ilustrativo: Tono de 5 kHz

```
import matplotlib.pyplot as plt
import numpy as np

def representa_espectrograma(x,NFFT,Fs,noverlap):
    Pxx, freqs, bins, im= plt.specgram(x, NFFT=NFFT, Fs=Fs, cmap='jet',
noverlap=noverlap)
    plt.ylabel('Frecuencia [Hz]')
    plt.xlabel('Tiempo [seg]')
    plt.show()
    return Pxx, freqs

f0= 5000
Fsmin= 2*f0

window= 256
noverlap= 128
nfft= 256

#%%
Fs= 4*Fsmin
t= np.arange(0,1,1./Fs)
x= np.sin(2*np.pi*f0*t+np.pi/6)

representa_espectrograma(x,nfft,Fs,noverlap)

#%%
Fs= 2*Fsmin;
t= np.arange(0,1,1./Fs)
x= np.sin(2*np.pi*f0*t+np.pi/6)

representa_espectrograma(x,nfft,Fs,noverlap)

#%%
Fs= Fsmin
t= np.arange(0,1,1./Fs)
x= np.sin(2*np.pi*f0*t+np.pi/6)

representa_espectrograma(x,nfft,Fs,noverlap)

#%%
Fs= 4./5*Fsmin
t= np.arange(0,1,1./Fs)
x= np.sin(2*np.pi*f0*t+np.pi/6)

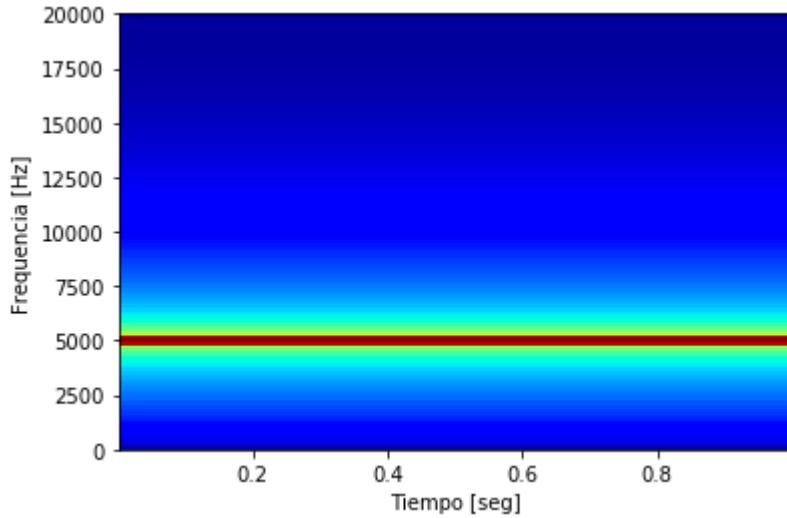
representa_espectrograma(x,nfft,Fs,noverlap)
```



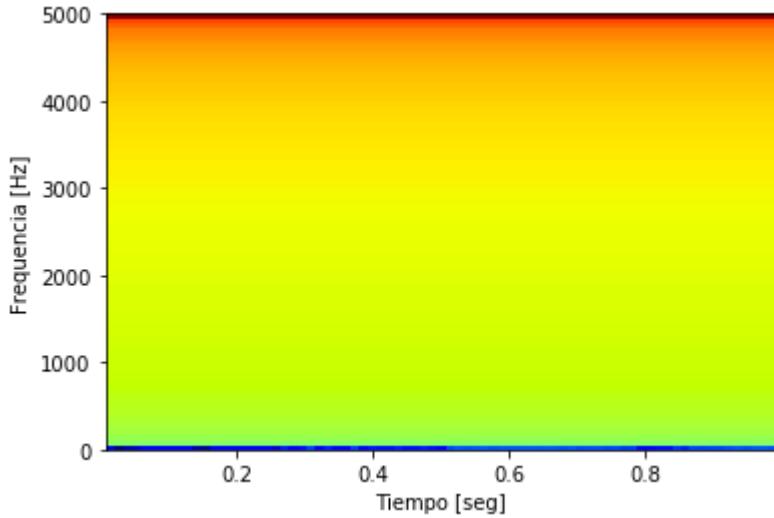
tono5khz.py

# Ejemplo ilustrativo: Tono de 5 kHz

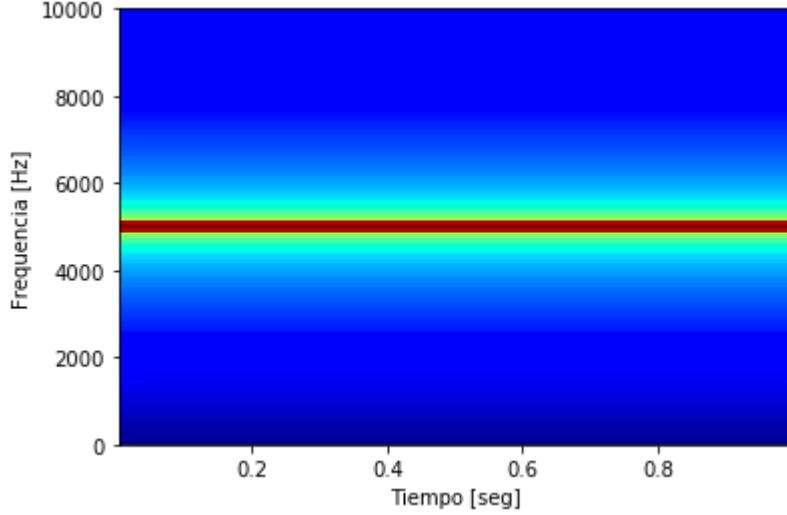
$F_s = 40 \text{ kHz}$



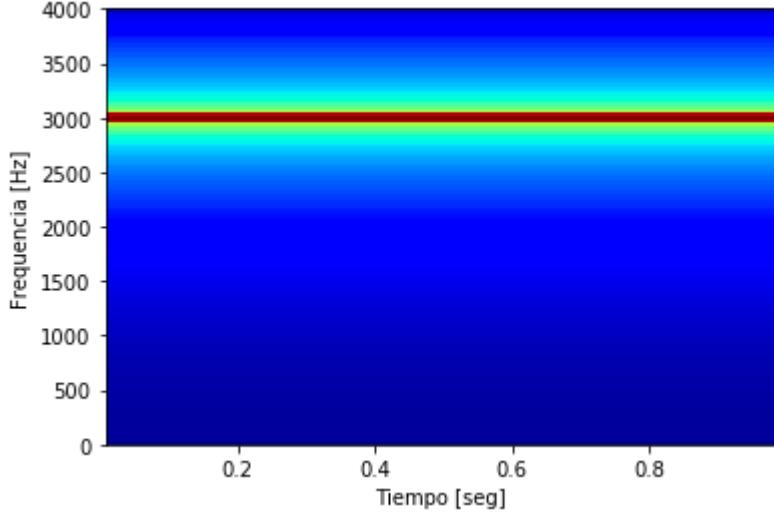
$F_s = 10 \text{ kHz}$



$F_s = 20 \text{ kHz}$

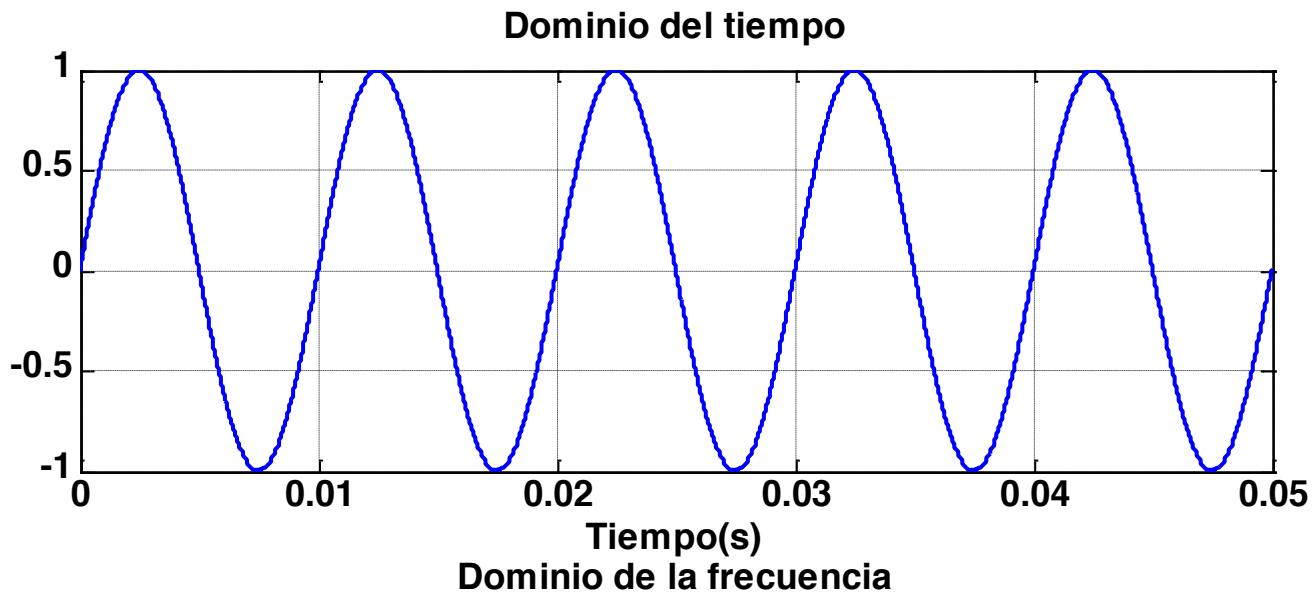


$F_s = 8 \text{ kHz}$



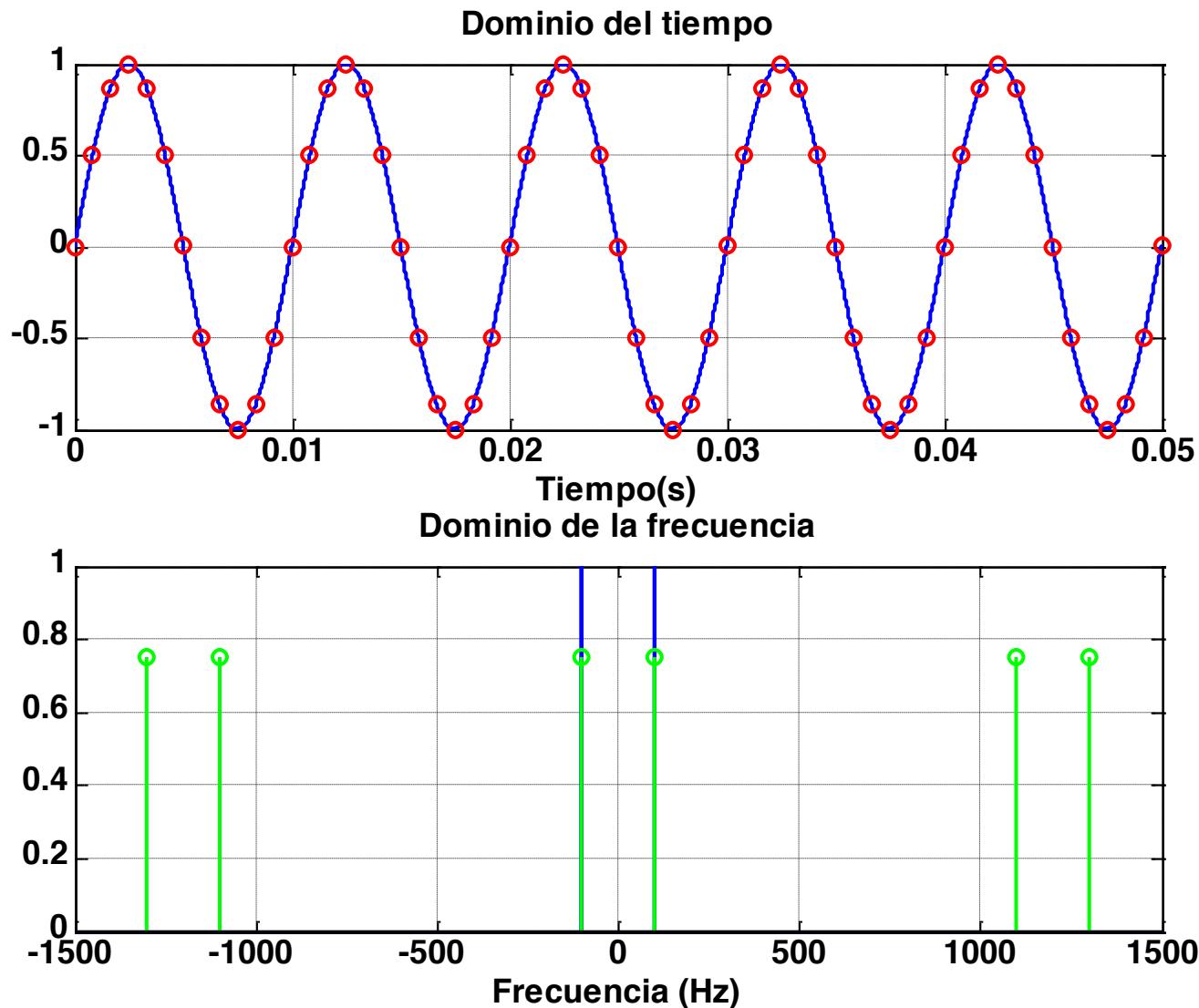
# Ilustración del aliasing (1/4)

- Tono:
  - $f_0 = 100$  Hz.
- Ancho de banda
  - $f_M = 100$ Hz.
- T<sup>a</sup> de muestreo:
  - $f_s \geq f_N = 200$ Hz



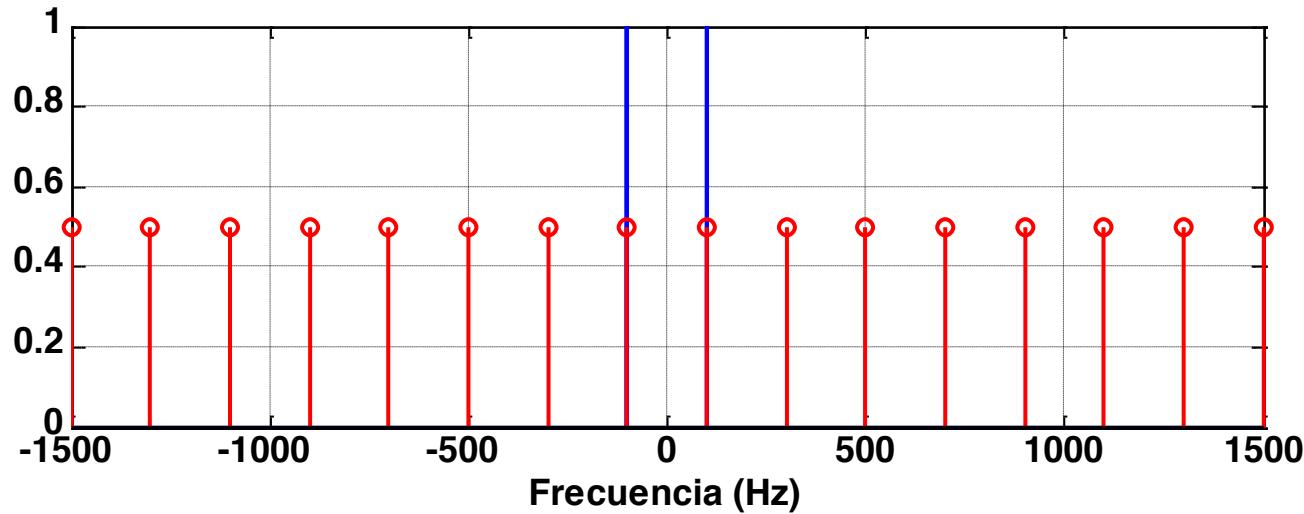
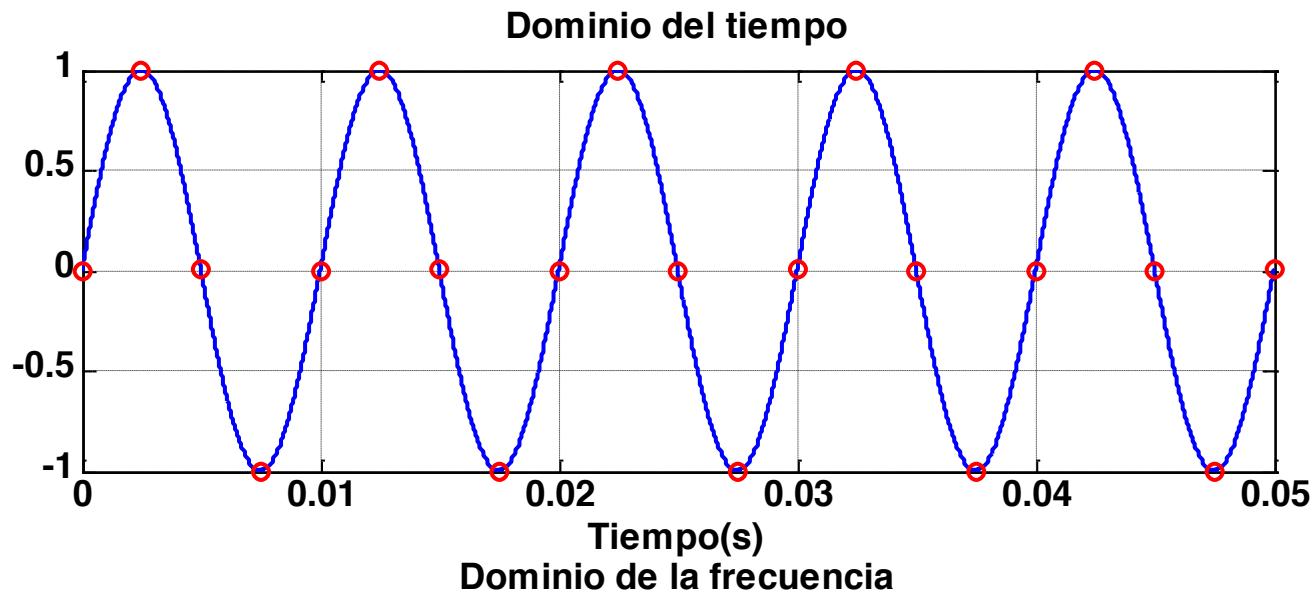
# Ilustración del aliasing (2/4)

- $f_s = 6 * f_N$
- $f_s = 1200\text{Hz}$



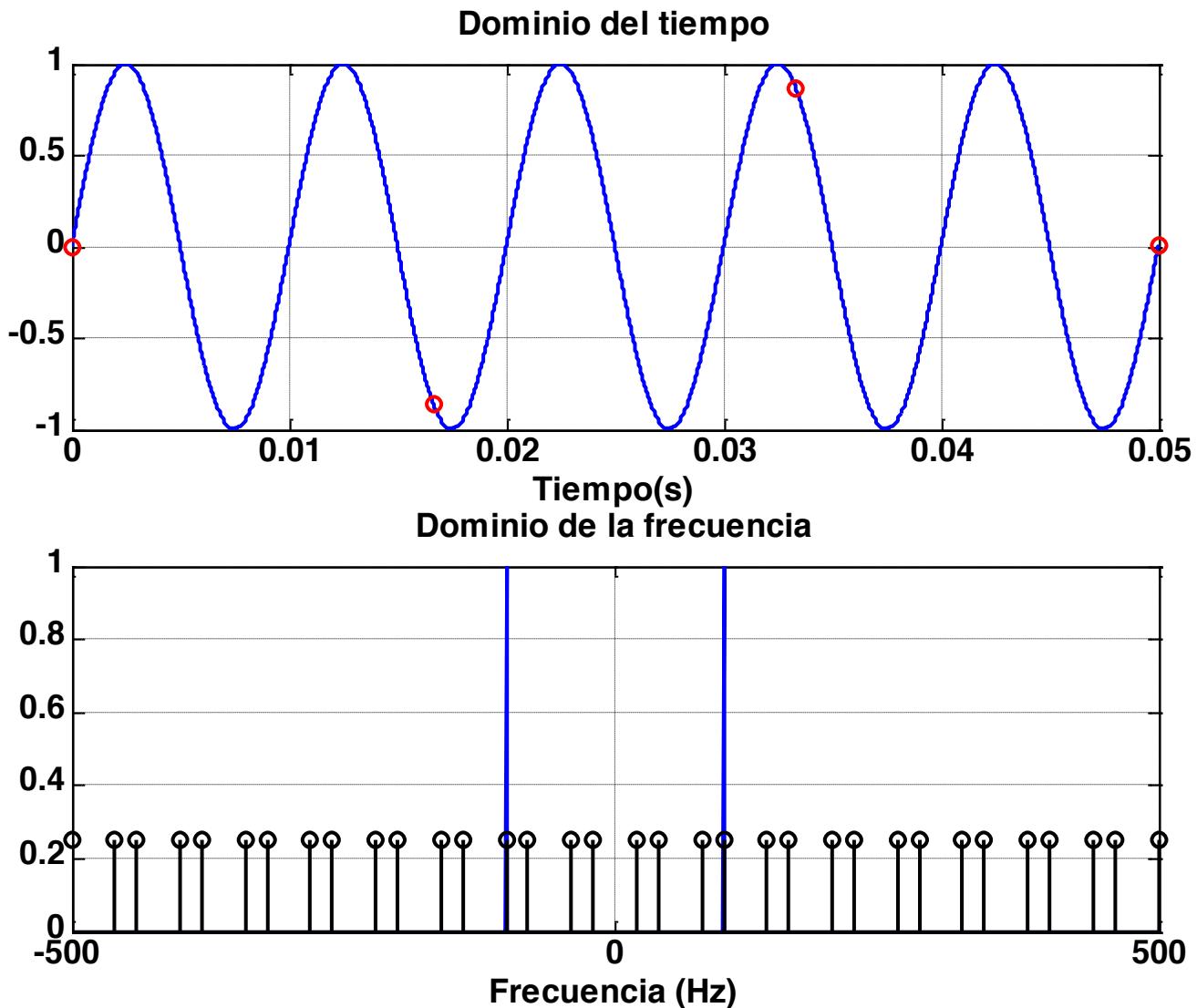
# Ilustración del aliasing (3/4)

- $f_s = 2 * f_N$
- $f_s = 400\text{Hz}$

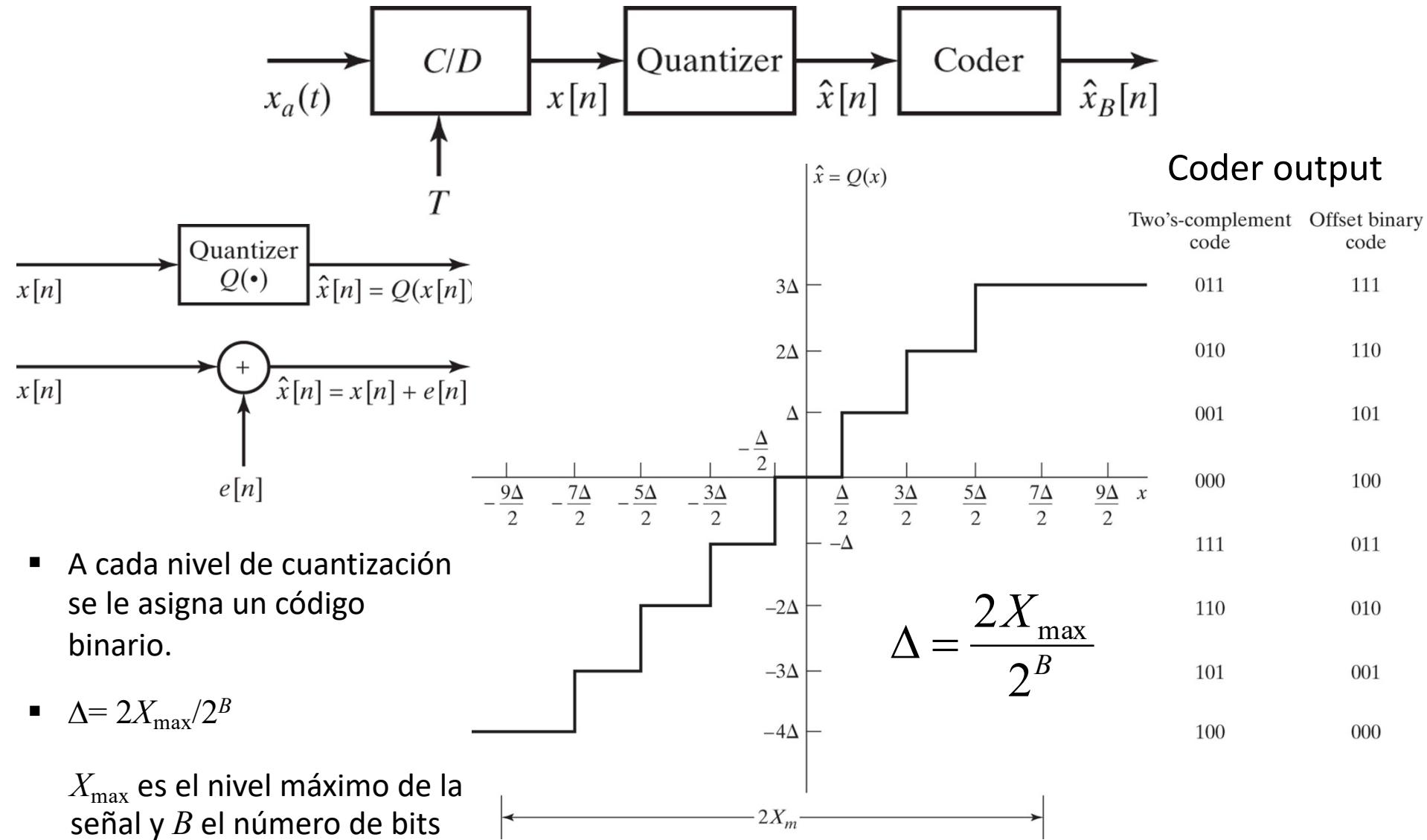


# Ilustración del aliasing (4/4)

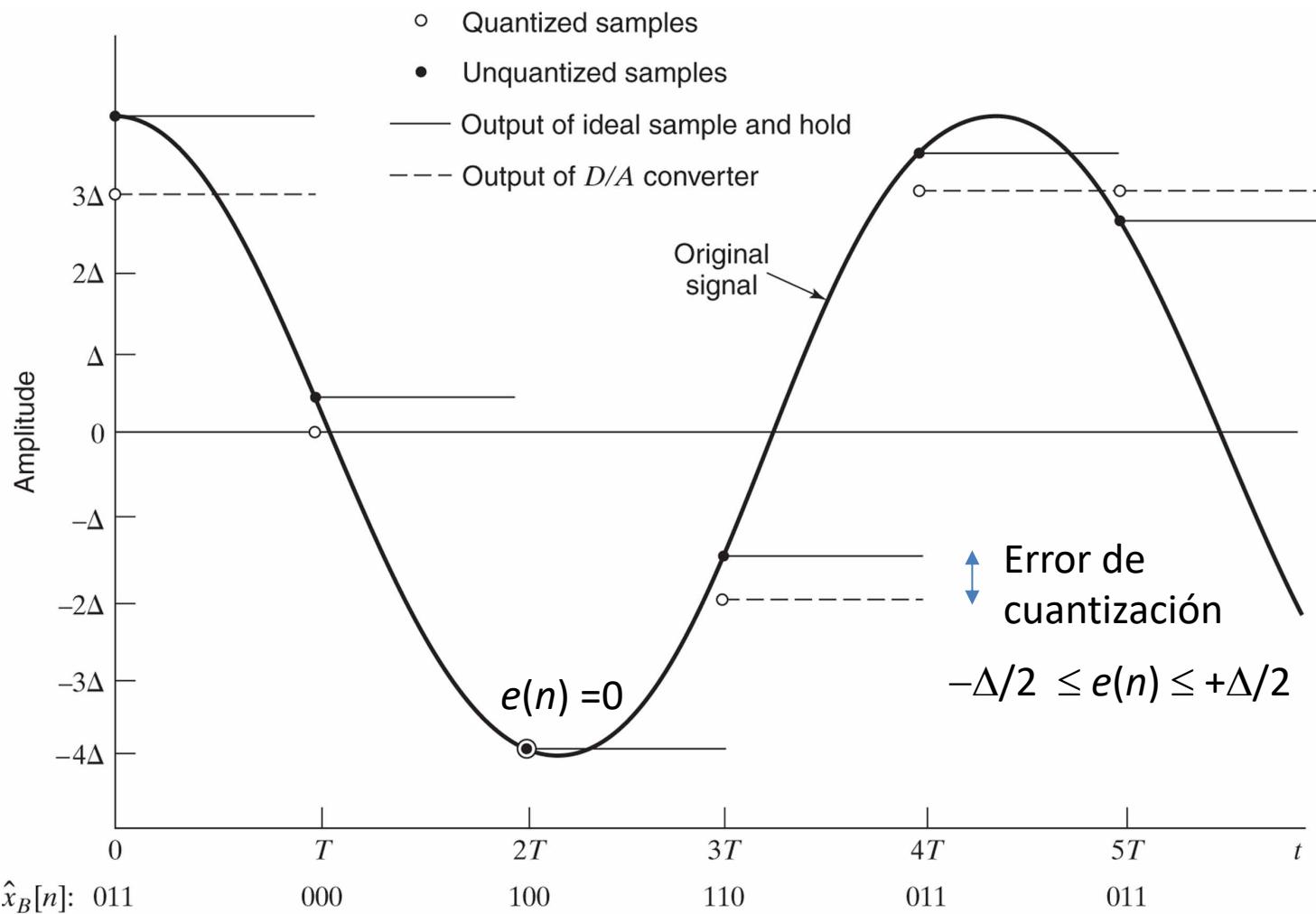
- $f_s = 0.3 * f_N$
- $f_s = 66.67 \text{ Hz}$
- No se respeta el teorema de muestreo



# Cuantización

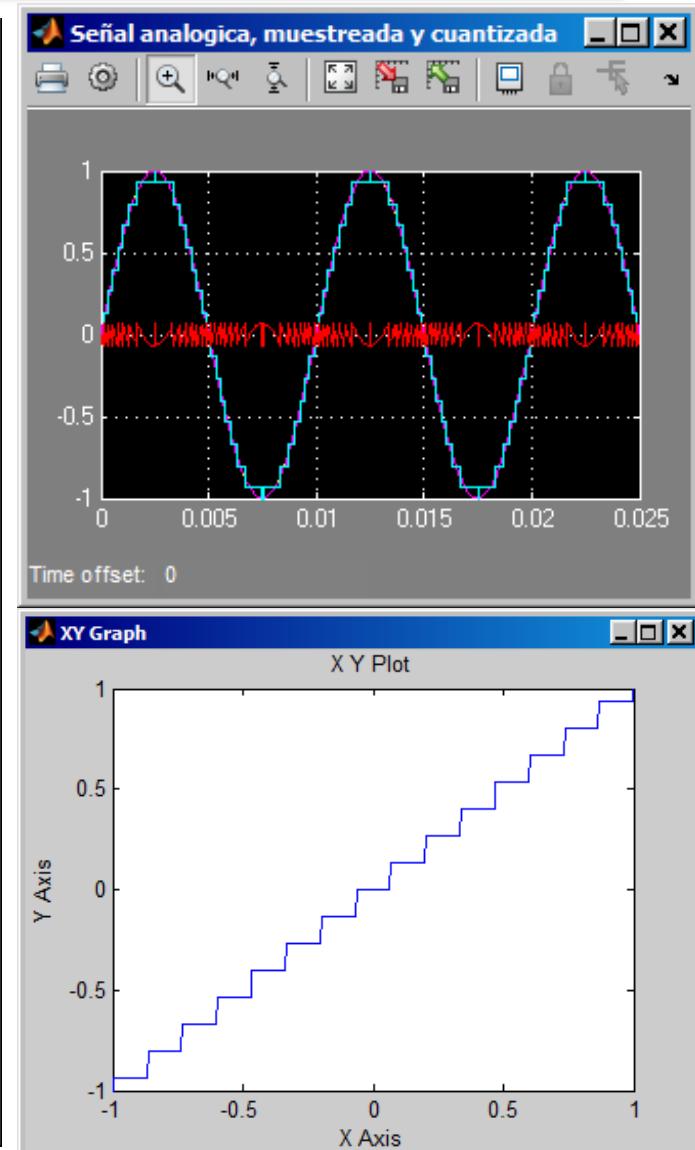
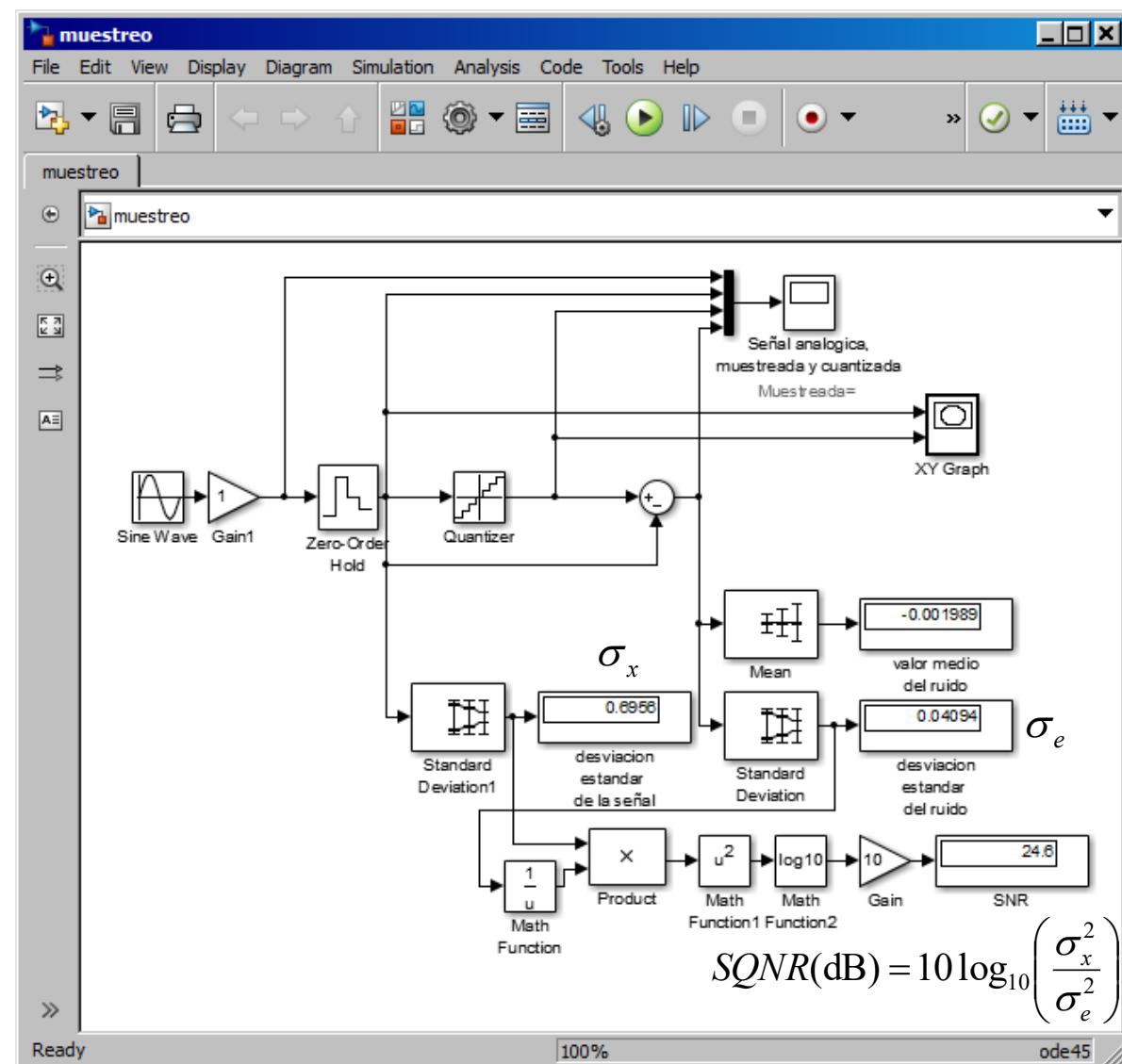


# Cuantización



muestreo

# Ejemplo de simulación: Simulink

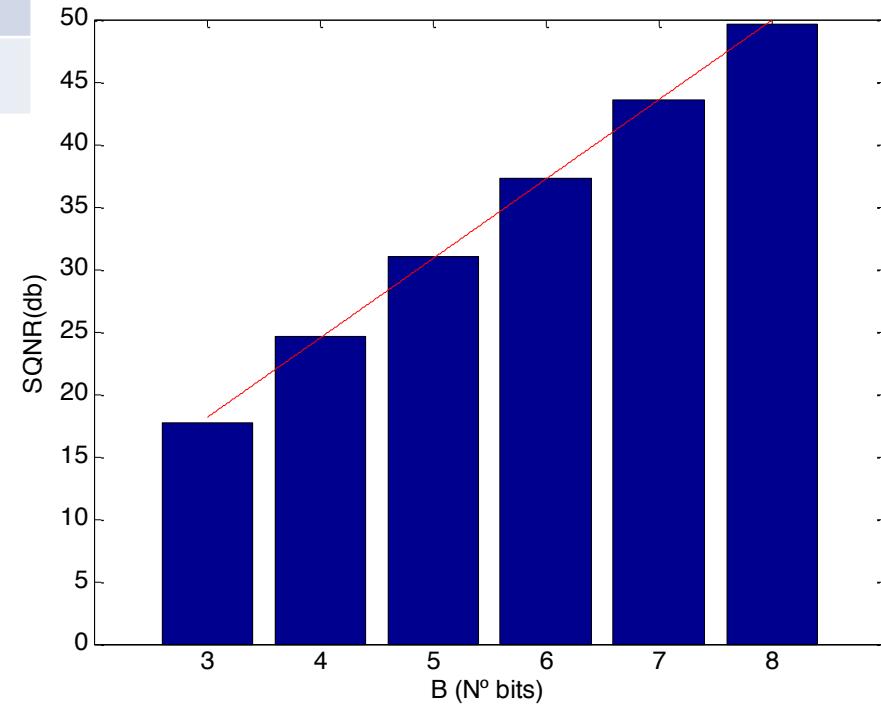


# Ejemplo de simulación: Simulink

B (Nº bits)	$\sigma_x$	$\sigma_e$	SQNR (dB)
3	0.6956	0.08997	17.76
4	0.6956	0.04094	24.60
5	0.6956	0.01946	31.06
6	0.6956	0.00947	37.32
7	0.6956	0.00462	43.55
8	0.6956	0.00228	49.67

$$SQNR(\text{dB}) = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right)$$

La SQNR aumenta aproximadamente 6 dB por bit.



## Análisis de regresión lineal:

$$SQNR(\text{dB}) = m \times B + b$$

$$r = 0.998$$

$$m = 6.3617$$

$$b = -0.9961$$

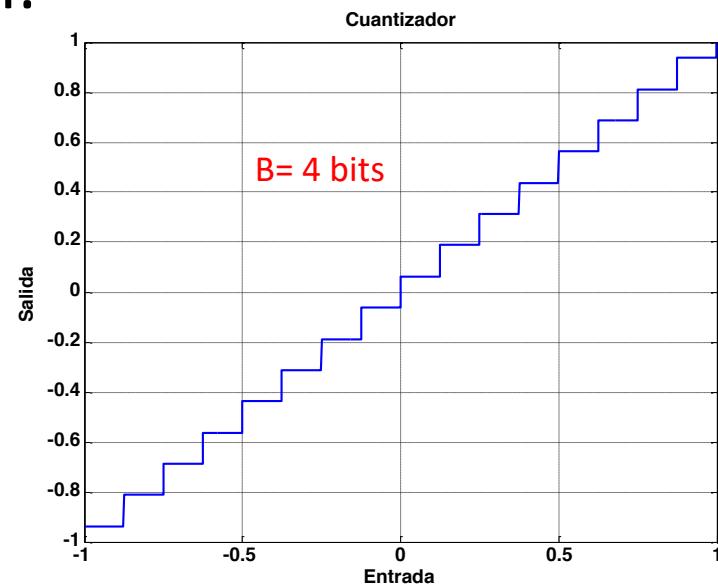
# Ejemplo: error (ruido) de cuantización

## ■ Función para realizar la cuantización:

```
function [Y,Q] = quantization(X,N)

% function [Y,Q] = quantization(X,N)
% Cuantizacion uniforme
% X Entrada
% N Numero de niveles de cuantización
% Y Salida
% Q Valor cuantizado (entre 0 y 2^log2(N)-1)

high = ceil(max(X));
low = floor(min(X));
qstep = (high-low)/N;           % Paso de cuantización.
Q = floor((X-low)/qstep);
low = low + qstep/2;
Y = low + qstep*Q;
```



```
x= -1:0.001:1;
y= quantization(x,2^4);
plot(x,y);
axis([-1 1 -1 1]);
grid;
title('Cuantizador');
xlabel('Entrada');
ylabel('Salida');
```

# Ejemplo: error (ruido) de cuantización

```
import numpy as np
import matplotlib.pyplot as plt

def quantization(X,N):
    """
    Cuantizacion uniforme
    X Entrada
    N Numero de niveles de cuantización
    Y Salida
    Q Valor cuantizado (entre 0 y 2^log2(N)-1)
    """

    high = np.ceil(max(X))
    low = np.floor(min(X))
    qstep = (high-low)/N      # Paso de cuantización.
    Q = np.floor((X-low)/qstep)
    low = low + qstep/2
    Y = low + qstep*Q
    return Y,Q

x= np.arange(-1,1,0.001)
y, Q = quantization(x,2**4);
plt.plot(x,y);
plt.axis([-1, 1, -1, 1]);
plt.grid
plt.title('Cuantizador');
plt.xlabel('Entrada');
plt.ylabel('Salida');
plt.show()
```

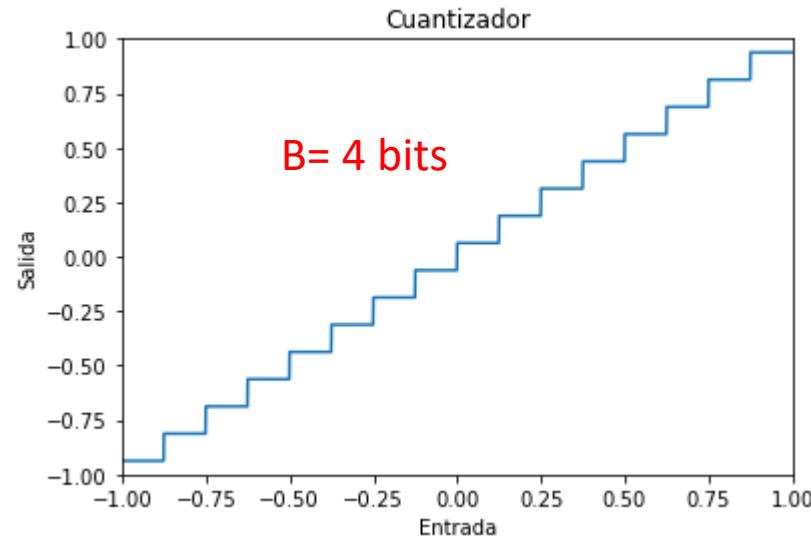
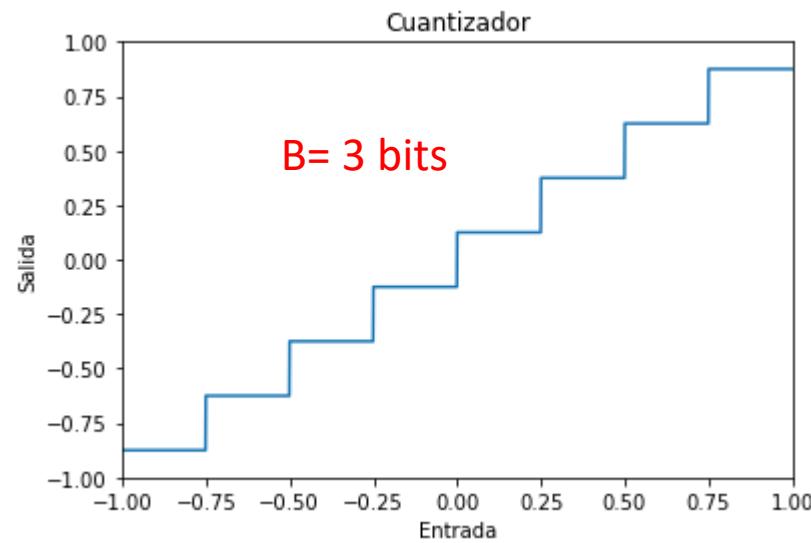


```
# Gráficos con dos ejes Y.

fig,ax = plt.subplots()
ax.plot(x,y);
ax.grid
ax.set_xlabel('x');
ax.set_ylabel('y');

ax2=ax.twinx()

ax2.plot(x,Q);
ax2.grid
ax2.set_ylabel('Q');
plt.show()
```

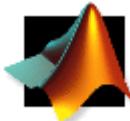


# Ejemplo: error (ruido) de cuantización

Open Speech Repository:

[http://www.voiptroubleshooter.com/open\\_speech/](http://www.voiptroubleshooter.com/open_speech/)

```
wav_file= 'OSR_us_000_0010_8k.wav';  
  
[x,Fs]= audioread(wav_file);  
  
% Cuantización  
B= 8; y8= quantization(x,2^B);  
B= 6; y6= quantization(x,2^B);  
B= 4; y4= quantization(x,2^B);  
B= 3; y3= quantization(x,2^B);  
  
% Análisis de la SQNR.  
subplot(2,2,1);  
plot(x,'b');  
title('B= 8 bits');  
hold on;  
plot(x-y8,'r');  
hold off;
```

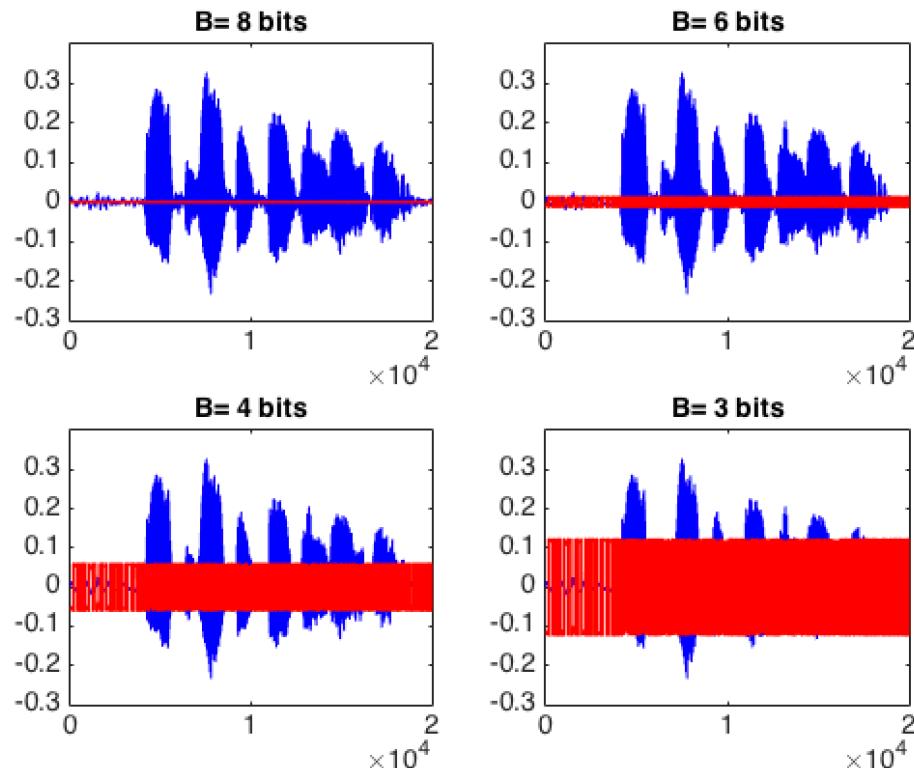


error\_cuantizacion\_ejemplo.m

```
subplot(2,2,2);  
plot(x,'b'); hold on;  
plot(x-y6,'r'); hold off;  
title('B= 6 bits');  
hold off;  
  
subplot(2,2,3);  
plot(x,'b'); hold on;  
plot(x-y4,'r'); hold off;  
title('B= 4 bits');  
  
subplot(2,2,4);  
plot(x,'b'); hold on;  
plot(x-y3,'r'); hold off;  
title('B= 3 bits');  
  
SQNR8= 10*log10(var(x)/var(x-y8))  
SQNR6= 10*log10(var(x)/var(x-y6))  
SQNR4= 10*log10(var(x)/var(x-y4))  
SQNR3= 10*log10(var(x)/var(x-y3))  
  
soundsc([x ; y8 ; y6 ; y4 ; y3] ,Fs);
```

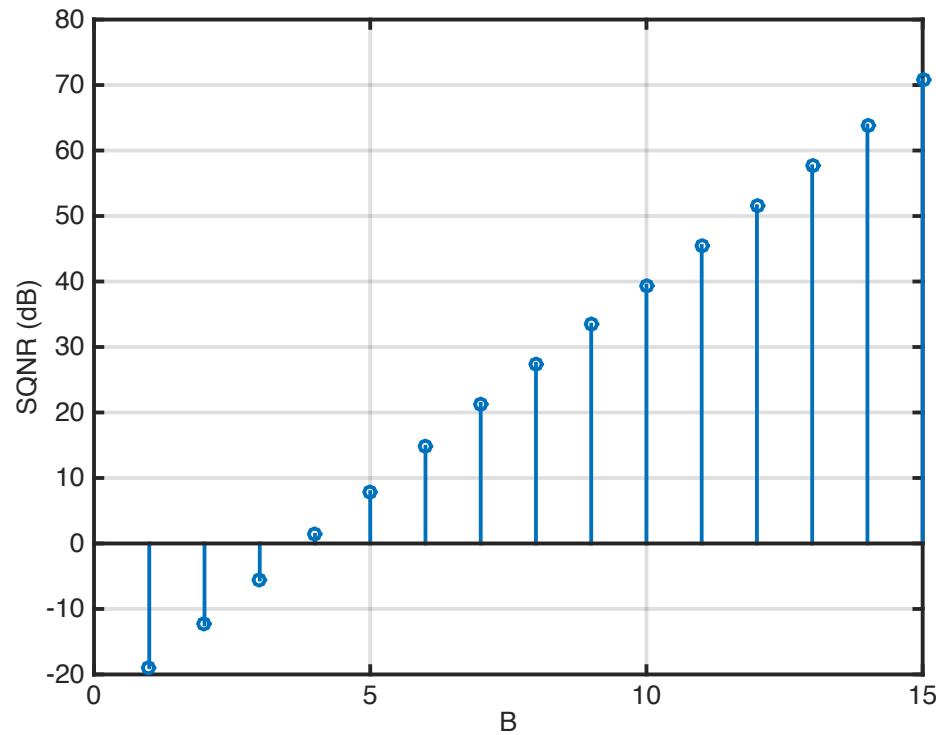
# Ejemplo: error (ruido) de cuantización

Señal original y error de cuantización



**Ejercicio:** Reproducir este ejemplo mediante Python.

Relación señal – ruido de cuantización



$$SQNR = \frac{\sigma_x^2}{\sigma_e^2} \quad ; \quad SQNR(\text{dB}) = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right)$$

# Ejemplo: error (ruido) de cuantización

```
import numpy as np
import matplotlib.pyplot as plt
import wave

def quantization(X,N):
    """
    Cuantizacion uniforme
    X Entrada
    N Numero de niveles de cuantización
    Y Salida
    Q Valor cuantizado (entre 0 y 2^log2(N)-1)
    """
    high = np.ceil(max(X))
    low = np.floor(min(X))
    qstep = (high-low)/N      # Paso de cuantización.
    Q = np.floor((X-low)/qstep)
    low = low + qstep/2
    Y = low + qstep*Q
    return Y, Q

def leer_wave(filename):
    spf = wave.open(filename,'r')
    #Leer el fichero .wav
    params = spf.getparams()
    framerate= params[2]
    x = spf.readframes(-1)
    x = np.fromstring(x, 'Int16')
    #plt.title(filename)
    #plt.plot(x)
    #plt.show()
    return x, framerate
```



```
x, Fs= leer_wave('OSR_us_000_0010_8k.wav')
t = np.arange(start= 0, stop= 1.0*x.size/Fs, step= 1./Fs)
plt.plot(t,x)
plt.xlabel('t (s)')
plt.show()
x= x/2.0**15

%%% Cuantización
B= 8
y8, Q = quantization(x,2**B)
error8= x-y8
plt.plot(t,x,t,error8)
plt.show()

B= 6
y6, Q= quantization(x,2**B)
error6= x-y6
plt.plot(t,x,t,error6)
plt.show()

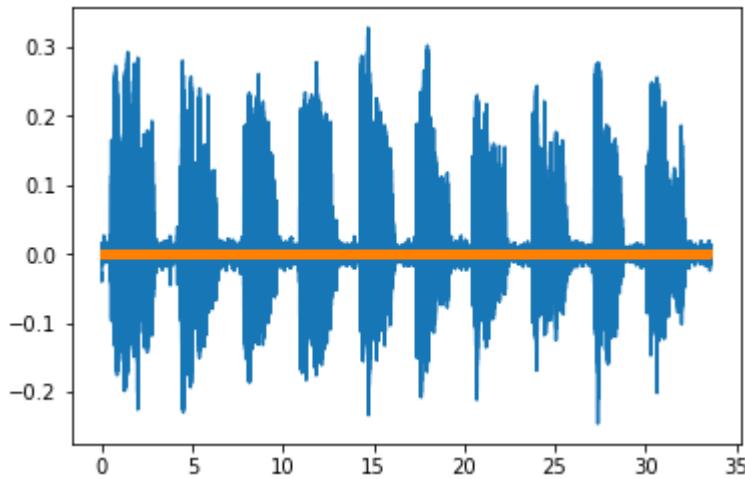
B= 4
y4, Q= quantization(x,2**B)
error4= x-y4
plt.plot(t,x,t,error4)
plt.show()

B= 3
y3, Q= quantization(x,2**B)
error3= x-y3
plt.plot(t,x,t,error3)
plt.show()

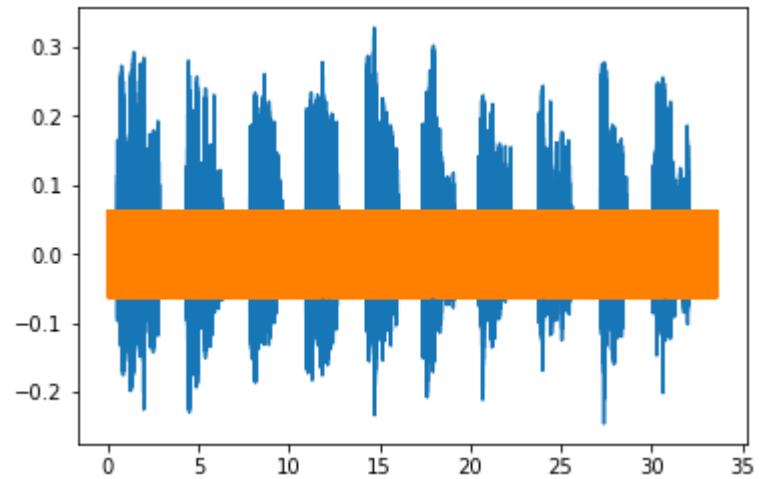
%%% Calculo de la relación señal-ruido de cuantización (SQNR)
SQNR8= 10*np.log10(np.var(x)/np.var(error8))
SQNR6= 10*np.log10(np.var(x)/np.var(error6))
SQNR4= 10*np.log10(np.var(x)/np.var(error4))
SQNR3= 10*np.log10(np.var(x)/np.var(error3))
```

# Ejemplo: error (ruido) de cuantización

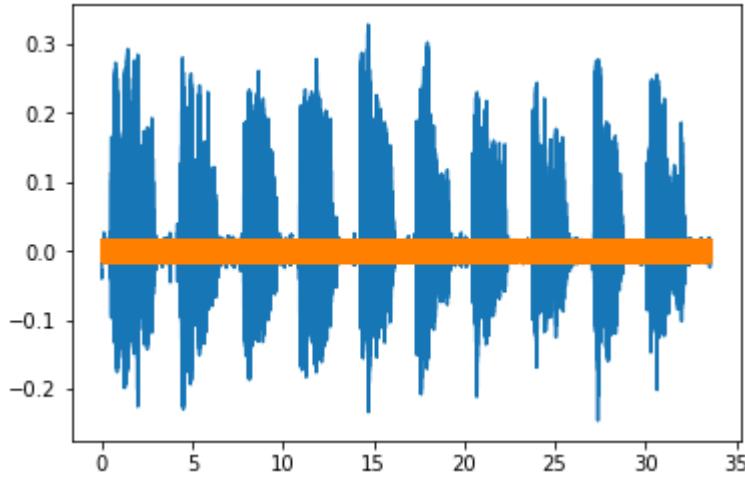
B= 8 bits



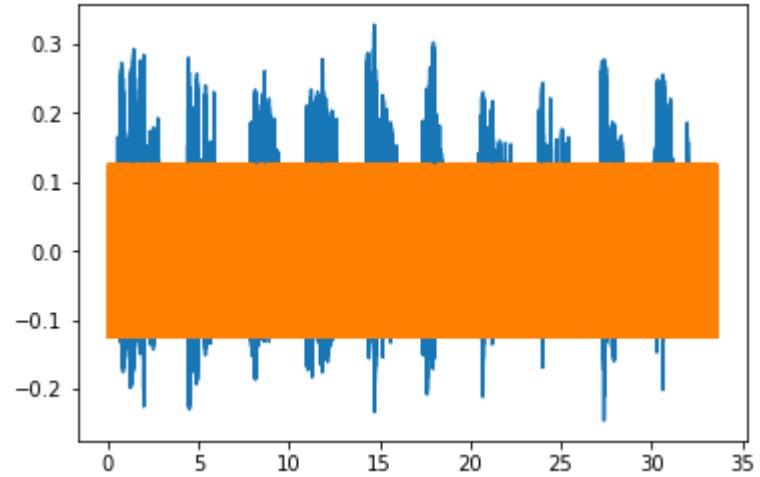
B= 4 bits



B= 6 bits



B= 3 bits



# Ejemplo: error (ruido) de cuantización

```
import numpy as np
import matplotlib.pyplot as plt
import wave

def quantization(X,N):
    """
    Cuantizacion uniforme
    X Entrada
    N Numero de niveles de cuantización
    Y Salida
    Q Valor cuantizado (entre 0 y 2^log2(N)-1)
    """
    high = np.ceil(max(X))
    low = np.floor(min(X))
    qstep = (high-low)/N      # Paso de cuantización.
    Q = np.floor((X-low)/qstep)
    low = low + qstep/2
    Y = low + qstep*Q
    return Y, Q

def leer_wave(filename):
    spf = wave.open(filename, 'r')
    #Leer el fichero .wav
    params = spf.getparams()
    framerate= params[2]
    x = spf.readframes(-1)
    x = np.fromstring(x, 'Int16')
    #plt.title(filename)
    #plt.plot(x)
    #plt.show()
    return x, framerate
```

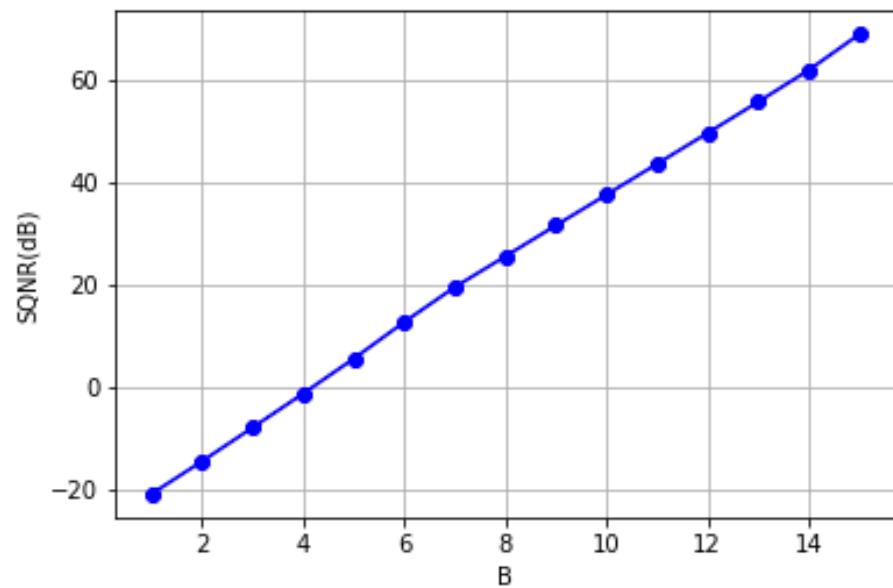


```
x, Fs= leer_wave('OSR_us_000_0010_8k.wav')
x= x/2.0**15

SQNR= []

for B in range(1,16):
    y, Q= quantization(x,2**B)
    error= x-y
    SQNR.append(10*np.log10(np.var(x)/np.var(error)))

plt.plot(range(1,16), SQNR, 'bo-')
plt.xlabel('B')
plt.ylabel('SQNR (dB)')
plt.grid()
plt.show()
```

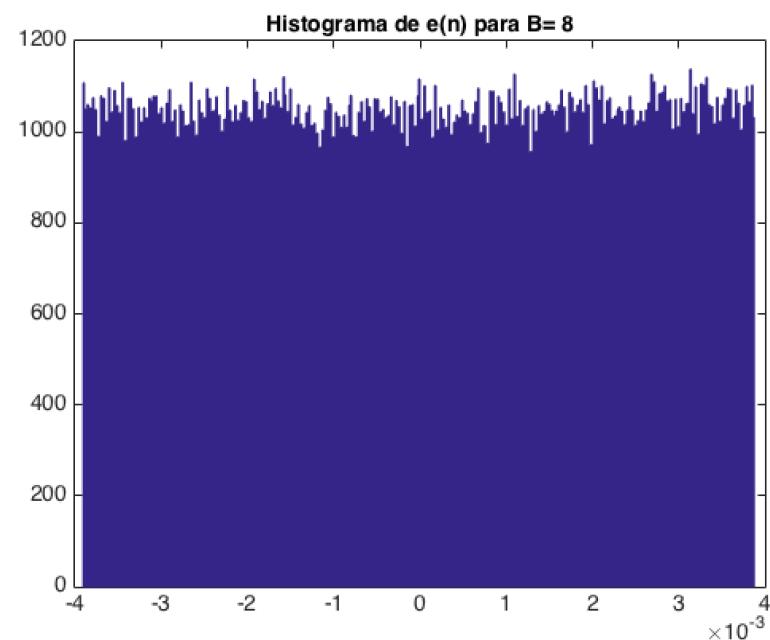
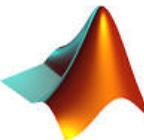


# Ejemplo: error (ruido) de cuantización

## ■ Modelado del error de cuantización:

```
wav_file= 'OSR_us_000_0010_8k.wav';
[x,Fs]= audioread(wav_file);
B= 8; y= quantization(x,2^B);
e= x-y;
hist(e,500);
title('Histograma de e(n) para B= 8')
```

```
B= 8
y, Q = quantization(x,2**B)
error= x-y
plt.hist(error, bins=300)
plt.title('Histograma del error')
plt.show()
```



## ■ Distribución uniforme:

$$p(e) = \begin{cases} K & -\Delta/2 \leq e \leq +\Delta/2 \\ 0 & \text{en otro caso} \end{cases}$$

$$1 = \int_{-\Delta/2}^{+\Delta/2} p(e)de = \int_{-\Delta/2}^{+\Delta/2} Kde = K[\Delta/2 - (-\Delta/2)] = K\Delta$$

$$K = \frac{1}{\Delta}$$

# Error de cuantización

- El proceso de cuantización introduce un **error**:

$$\hat{x}(n) = Q[x(n)] = x(n) + e(n)$$

- El **ruido de cuantización** se puede modelar estadísticamente con una **distribución uniforme**:

$$p(e) = 1/\Delta, \quad -\Delta/2 \leq e \leq \Delta/2$$

$$\sigma_e^2 = \int_{-\Delta/2}^{+\Delta/2} e^2 p(e) de = \frac{\Delta^2}{12} = \frac{1}{3} \frac{X_{\max}^2}{2^{2B}}$$

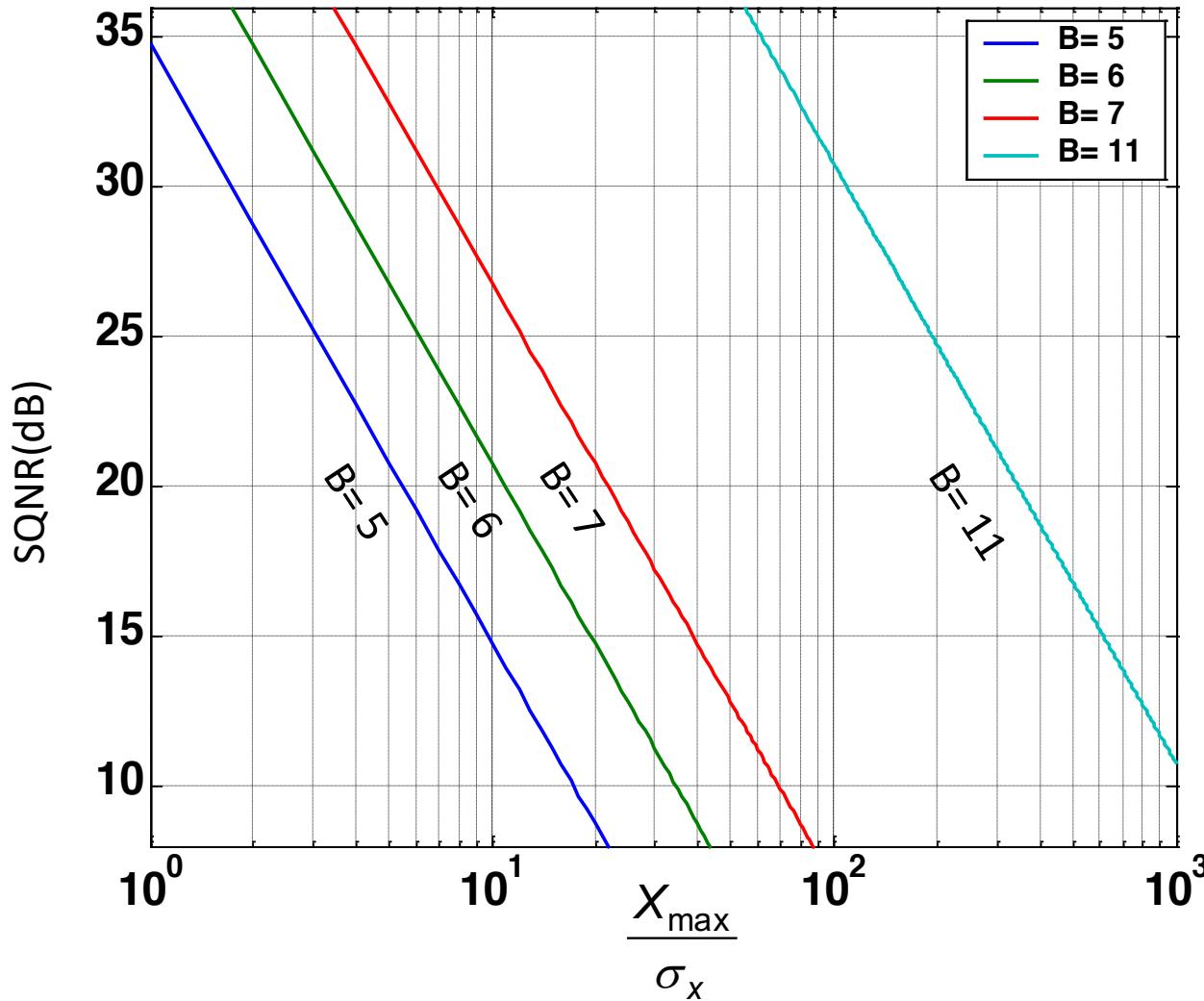
- Relación señal ruido de cuantización (**SQNR**):

$$SQNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{\sigma_x^2}{X_{\max}^2 / (3 \cdot 2^{2B})} = \frac{3 \cdot 2^{2B}}{X_{\max}^2 / \sigma_x^2}$$

$$SQNR(dB) = 6B + 4.77 - 20 \log_{10} \left( \frac{X_{\max}}{\sigma_x} \right)$$

# Relación señal ruido de cuantización

$$SQNR(dB) = 6B + 4.77 - 20 \log_{10} \left( \frac{X_{\max}}{\sigma_x} \right)$$



# Diferentes esquemas de cuantización

- Inconvenientes de la cuantización uniforme:
  - La **SQNR depende** de la **potencia** de la señal de entrada.
    - Al disminuir la varianza disminuye rápidamente.
  - Ej.: Señal de voz (amplitudes pequeñas son más probables)

$$SQNR(dB) = 6B + 4.77 - 20 \log_{10} \left( \frac{X_{\max}}{\sigma_x} \right)$$

- Otros **esquemas de cuantización** más eficientes:
  - **Compresión logarítmica.**
    - Ley A.
    - Ley  $\mu$ .
  - **Cuantización adaptable.**
    - Adaptación hacia delante.
    - Adaptación hacia atrás.
  - **Cuantización diferencial.**
  - **Cuantización predictiva.**

# Ejemplo: histograma de una señal de voz

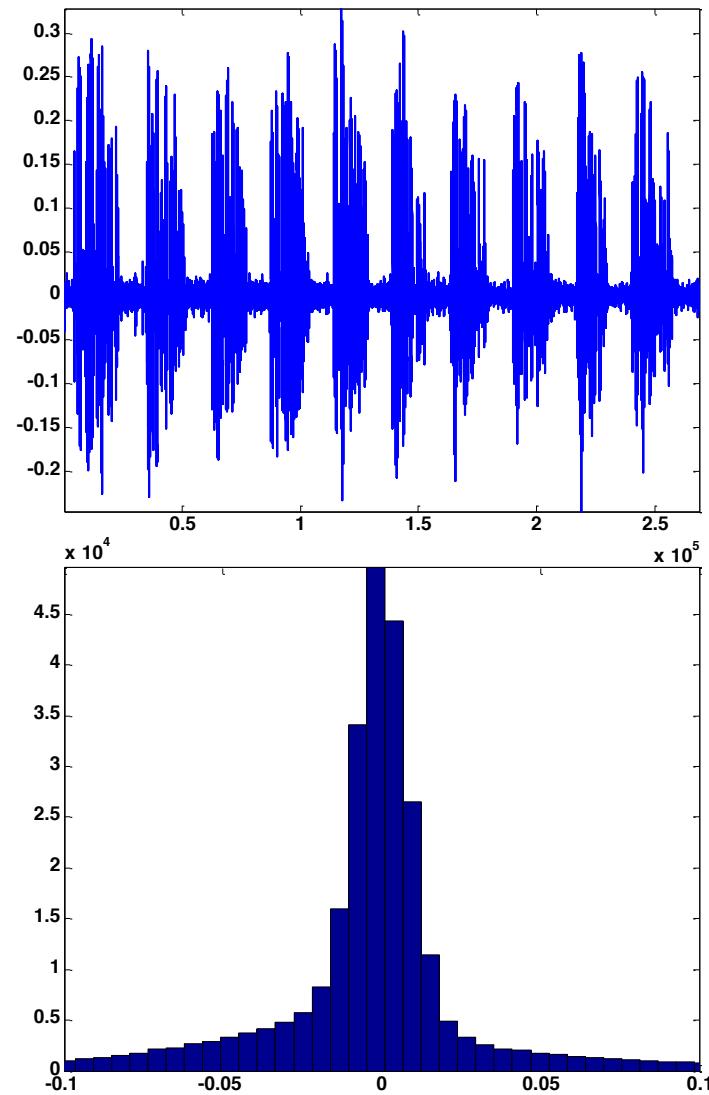
- Open Speech Repository:

[http://www.voiptroubleshooter.com/open\\_speech/](http://www.voiptroubleshooter.com/open_speech/)

- American English Sentences:

- OSR\_us\_000\_0010\_8k.wav

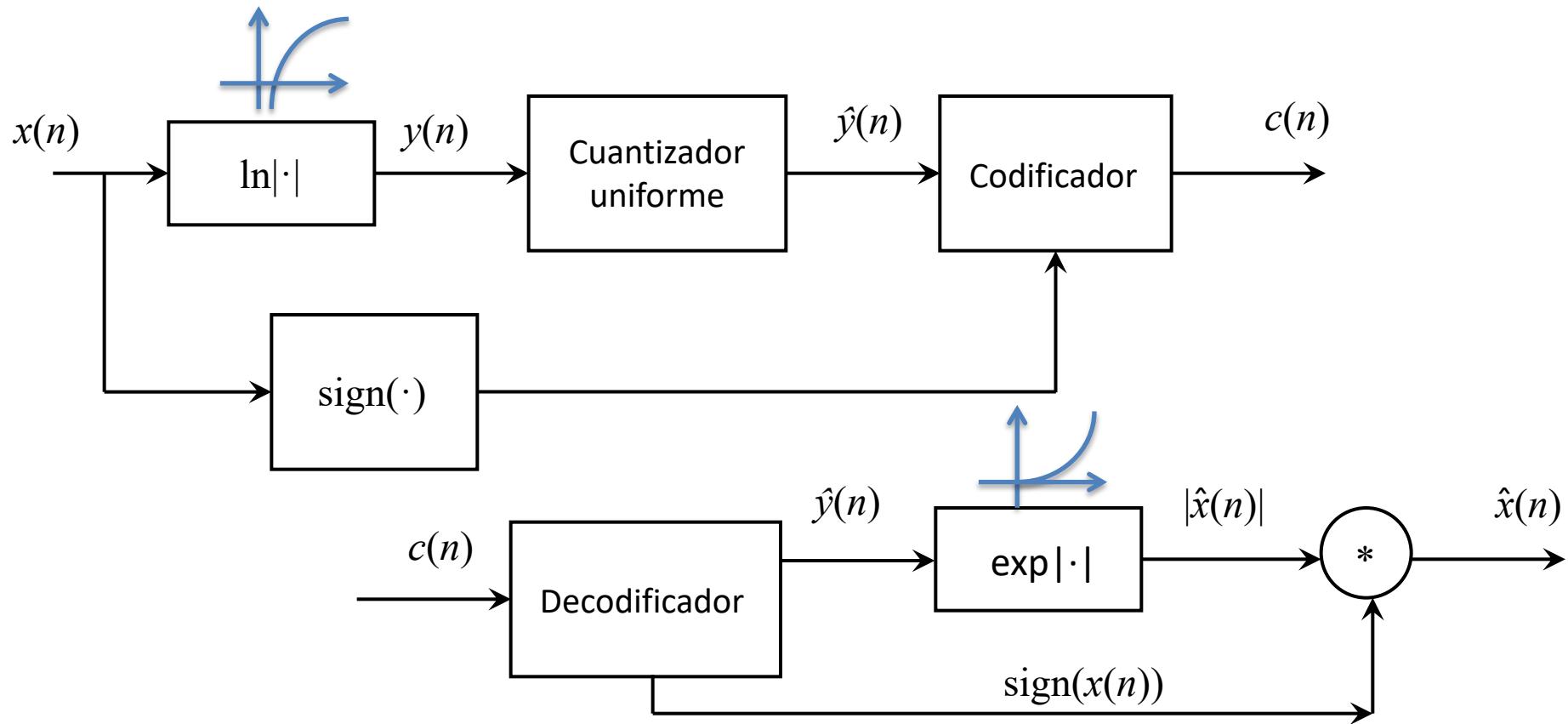
```
wav_file= 'OSR_us_000_0010_8k.wav';  
  
[x,Fs,NBits]= wavread(wav_file);  
  
figure(1);  
plot(x);  
axis([1 Inf -Inf Inf]);  
  
figure(2);  
hist(x,100);  
axis([-0.1 0.1 -Inf Inf]);
```



# Compresión logarítmica

## ■ Cuantización no uniforme

- Comprime la señal y utiliza un cuantizador uniforme.
- La SQNR no depende del nivel de señal.



# Error de cuantización. SQNR.

- Compresión logarítmica:

$$y(n) = \ln |x(n)|$$

- Error de cuantización:

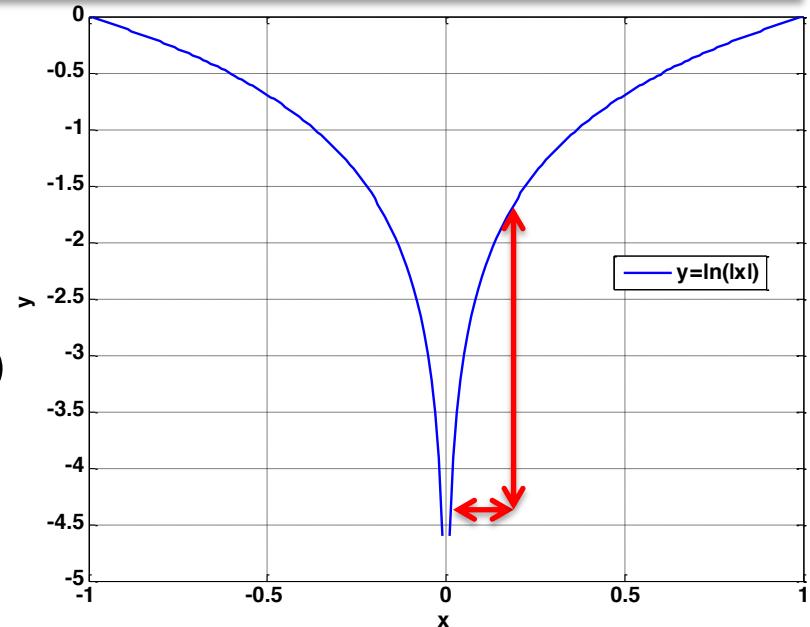
$$\hat{y}(n) = Q[\ln |x(n)|] = \ln |x(n)| + \varepsilon(n)$$

- Cálculo de la SQNR:

$$\begin{aligned}\hat{x}(n) &= \text{sign}[x(n)] \exp[\hat{y}(n)] = \\&= \text{sign}[x(n)] \exp[\ln|x(n)| + \varepsilon(n)] = \text{sign}[x(n)] \exp[\ln|x(n)|] \exp[\varepsilon(n)] \\&= \text{sign}[x(n)] |x(n)| \exp[\varepsilon(n)] = x(n) \exp[\varepsilon(n)]\end{aligned}$$

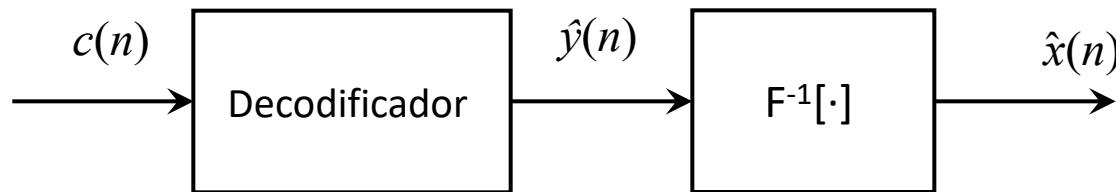
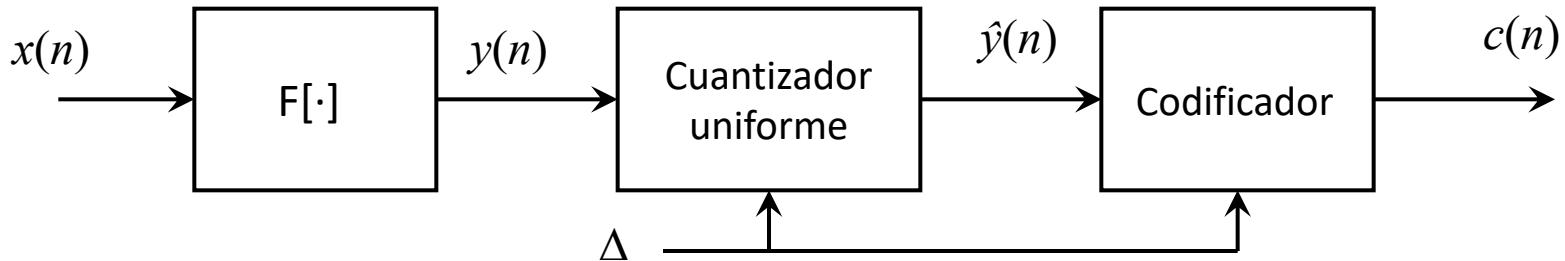
○ Si  $\varepsilon(n)$  es pequeño  $\Rightarrow \exp[\varepsilon(n)] \cong 1 + \varepsilon(n)$

$$\begin{aligned}\hat{x}(n) &= x(n) \exp[\varepsilon(n)] = x(n)[1 + \varepsilon(n)] = \\&= x(n) + x(n)\varepsilon(n) = x(n) + f(n)\end{aligned}$$



$$SQNR = \frac{\sigma_x^2}{\sigma_f^2} = \frac{\sigma_x^2}{\sigma_x^2 \sigma_\varepsilon^2} = \frac{1}{\sigma_\varepsilon^2}$$

# Ley $\mu$ y ley A



- Ley  $\mu$  (EE.UU. y Japón):

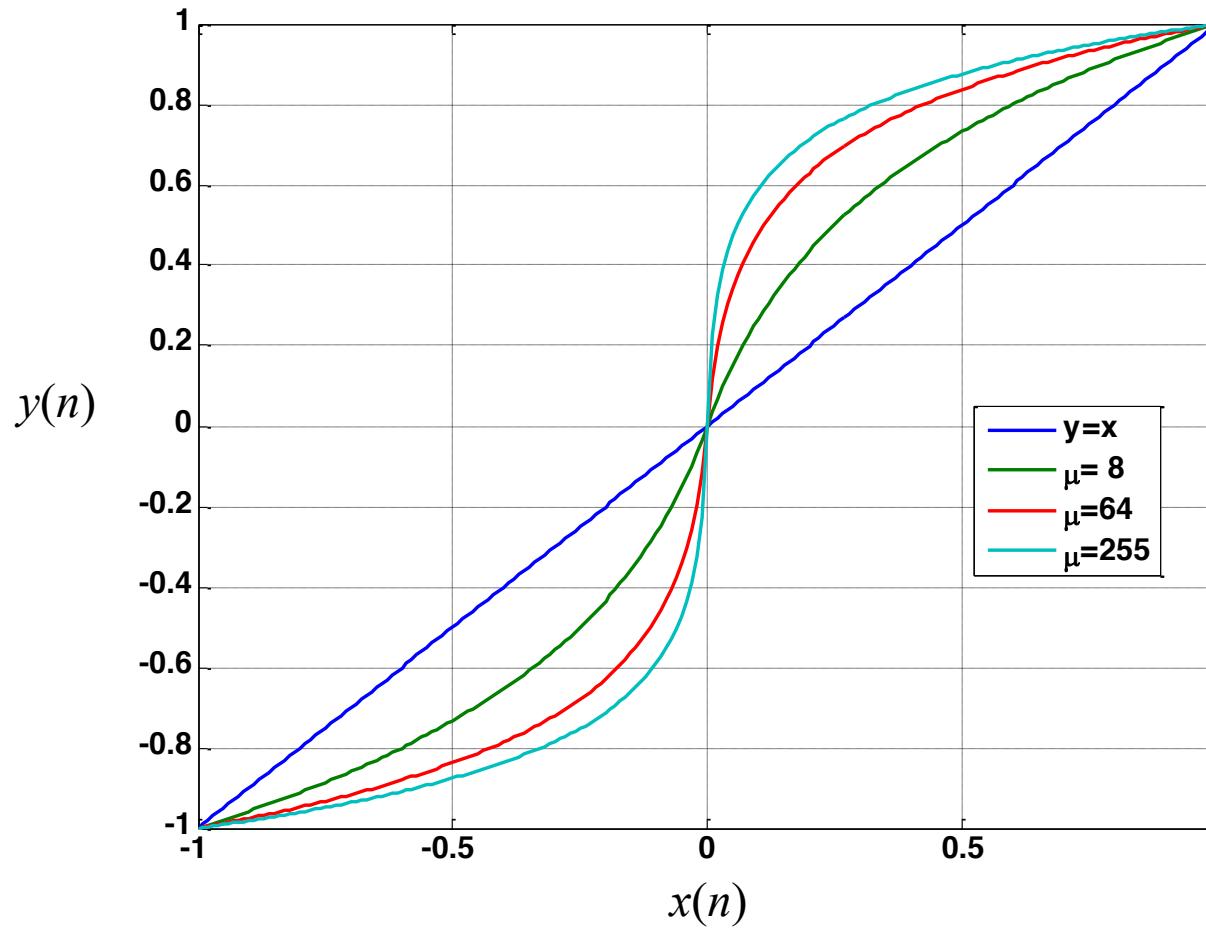
$$y(n) = F[x(n)] = X_{\max} \frac{\ln \left[ 1 + \mu \frac{|x(n)|}{X_{\max}} \right]}{\ln[1 + \mu]} \text{sign}[x(n)]$$

- Ley A (Europa):

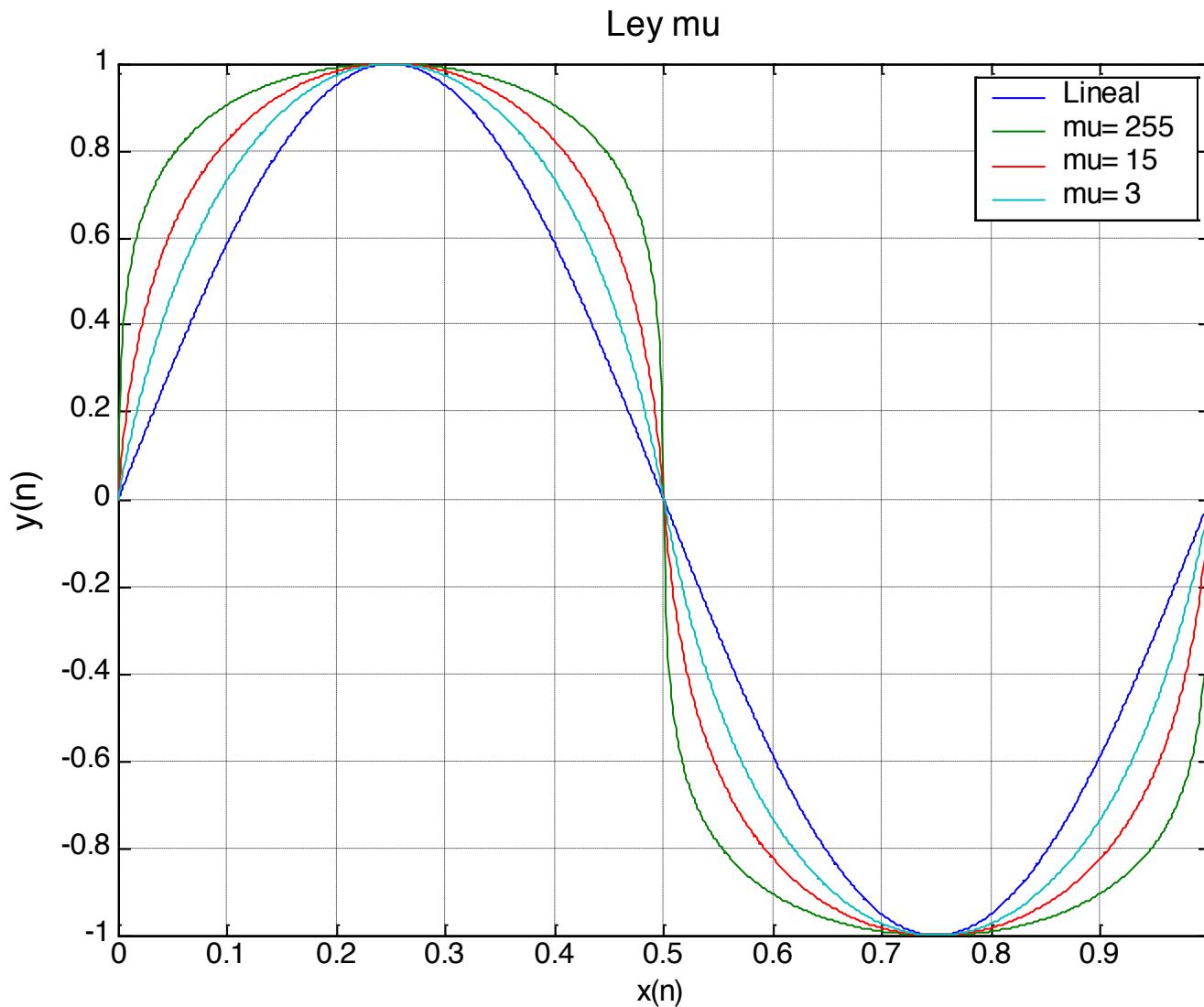
$$y(n) = F[x(n)] = \begin{cases} \frac{A|x(n)|}{1 + \ln(A)} \text{sign}[x(n)] & 0 \leq \frac{|x(n)|}{X_{\max}} < \frac{1}{A} \\ X_{\max} \frac{\left( 1 + \ln \left( A \frac{|x(n)|}{X_{\max}} \right) \right)}{1 + \ln A} \text{sign}[x(n)] & \frac{1}{A} \leq \frac{|x(n)|}{X_{\max}} \leq 1 \end{cases}$$

# Ley $\mu$

$$y(n) = F[x(n)] = X_{\max} \frac{\ln \left[ 1 + \mu \frac{|x(n)|}{X_{\max}} \right]}{\ln [1 + \mu]} \text{sign}[x(n)]$$



# Ejemplo: Ley $\mu$ (señal senoidal)



# Ejemplo: Ley $\mu$

- Open Speech Repository:  
[http://www.voiptroubleshooter.com/open\\_speech/](http://www.voiptroubleshooter.com/open_speech/)

- American English Sentences:
  - OSR\_us\_000\_0010\_8k.wav

```
wav_file= 'OSR_us_000_0010_8k.wav';
[x,Fs,NBits]= wavread(wav_file);

subplot(2,2,1);
plot(x);
axis([1 Inf -Inf Inf])

subplot(2,2,2);
hist(x,100);
axis([-0.1 0.1 -Inf Inf])

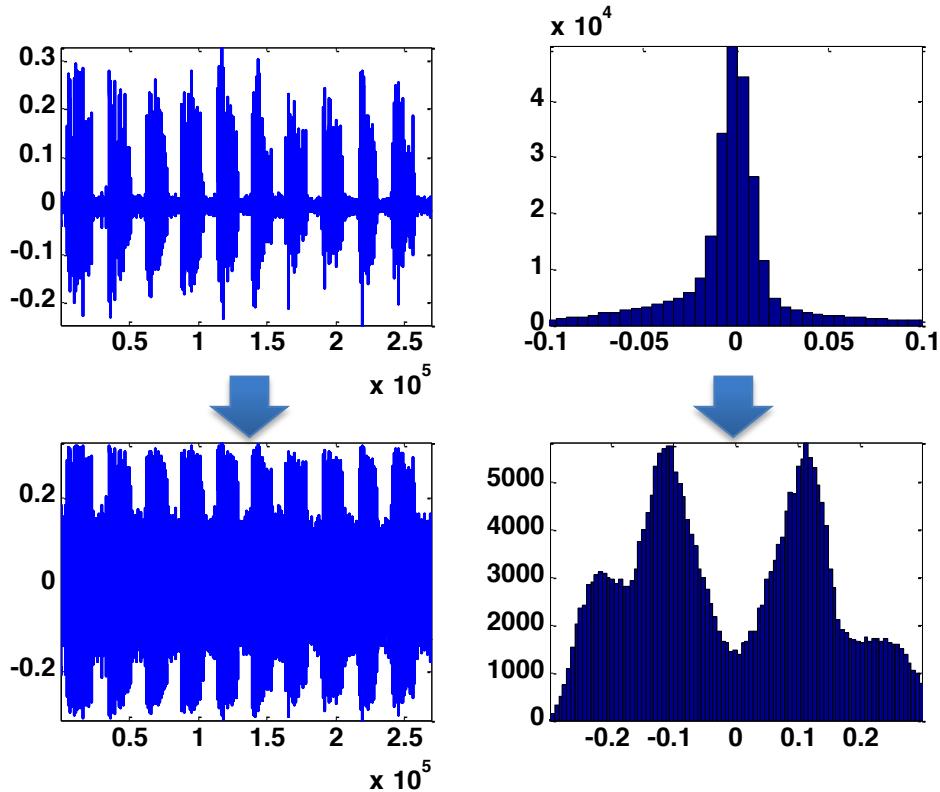
ymu= mulaw(x,255);

subplot(2,2,3);
plot(ymu);
axis([1 Inf -Inf Inf])

subplot(2,2,4);
hist(ymu,100);
axis([-0.3 0.3 -Inf Inf])
```

mulaw.m

```
function y= mulaw(x,mu)
Xmax= max(x);
y=
Xmax*log(1+mu*abs(x)/Xmax)/log(1+mu).*si
gn(x);
```



# Ejemplo: Ley $\mu$

```
import numpy as np
import matplotlib.pyplot as plt
import wave

def leer_wave(filename):
    spf = wave.open(filename,'r')
    #Leer el fichero .wav
    params = spf.getparams()
    framerate= params[2]
    x = spf.readframes(-1)
    x = np.fromstring(x, 'Int16')
    #plt.title(filename)
    #plt.plot(x)
    #plt.show()
    return x, framerate

def mulaw(x,mu):
    Xmax= np.max(x)
    y= Xmax*np.log(1+mu*np.abs(x)/Xmax)/np.log(1+mu)*np.sign(x)
    return y

## Lectura del fichero WAVE
x, Fs= leer_wave('OSR_us_000_0010_8k.wav')
x= x/2.0**15
t= np.arange(start= 0, stop= 1.0*x.size/Fs, step= 1./Fs)
plt.plot(t,x)
plt.xlabel('t (s)')
plt.show()
```

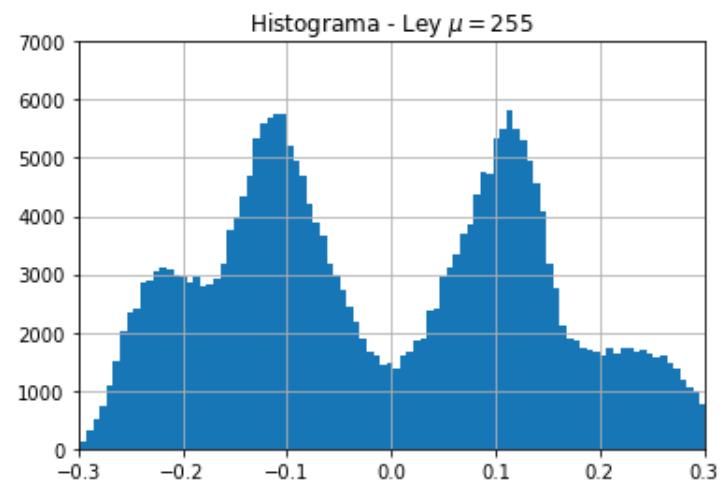
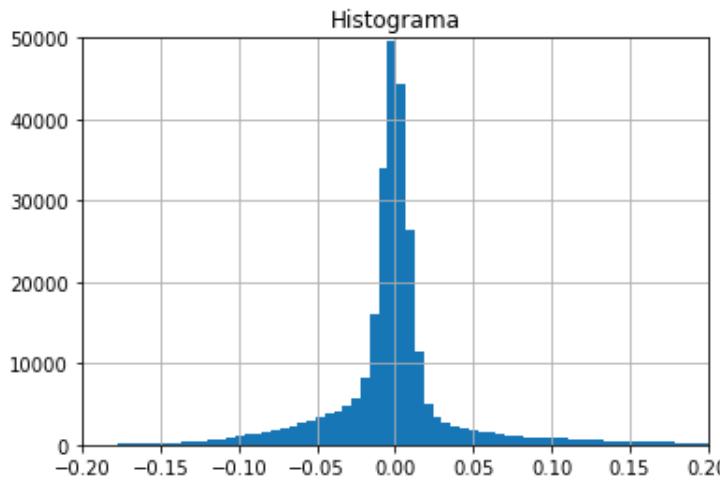
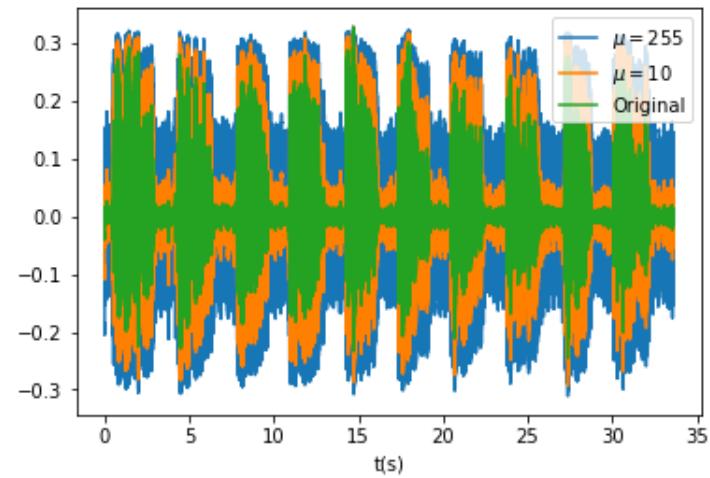
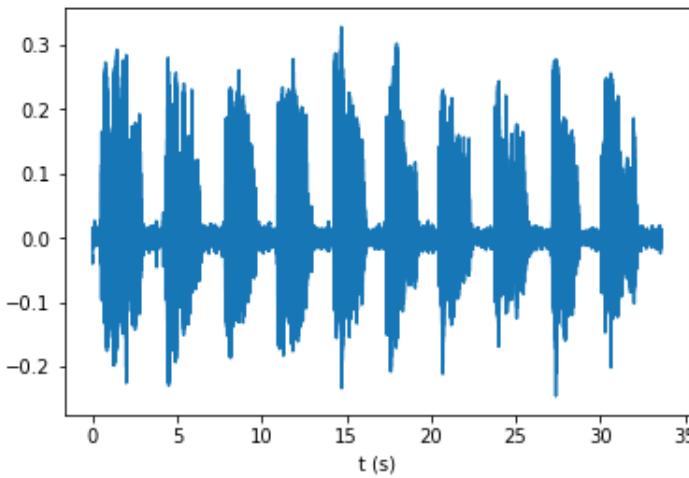


```
## Histograma de la señal original
plt.hist(x, bins=100)
plt.title('Histograma')
plt.axis([-0.2, 0.2, 0, 50000])
plt.grid()
plt.show()

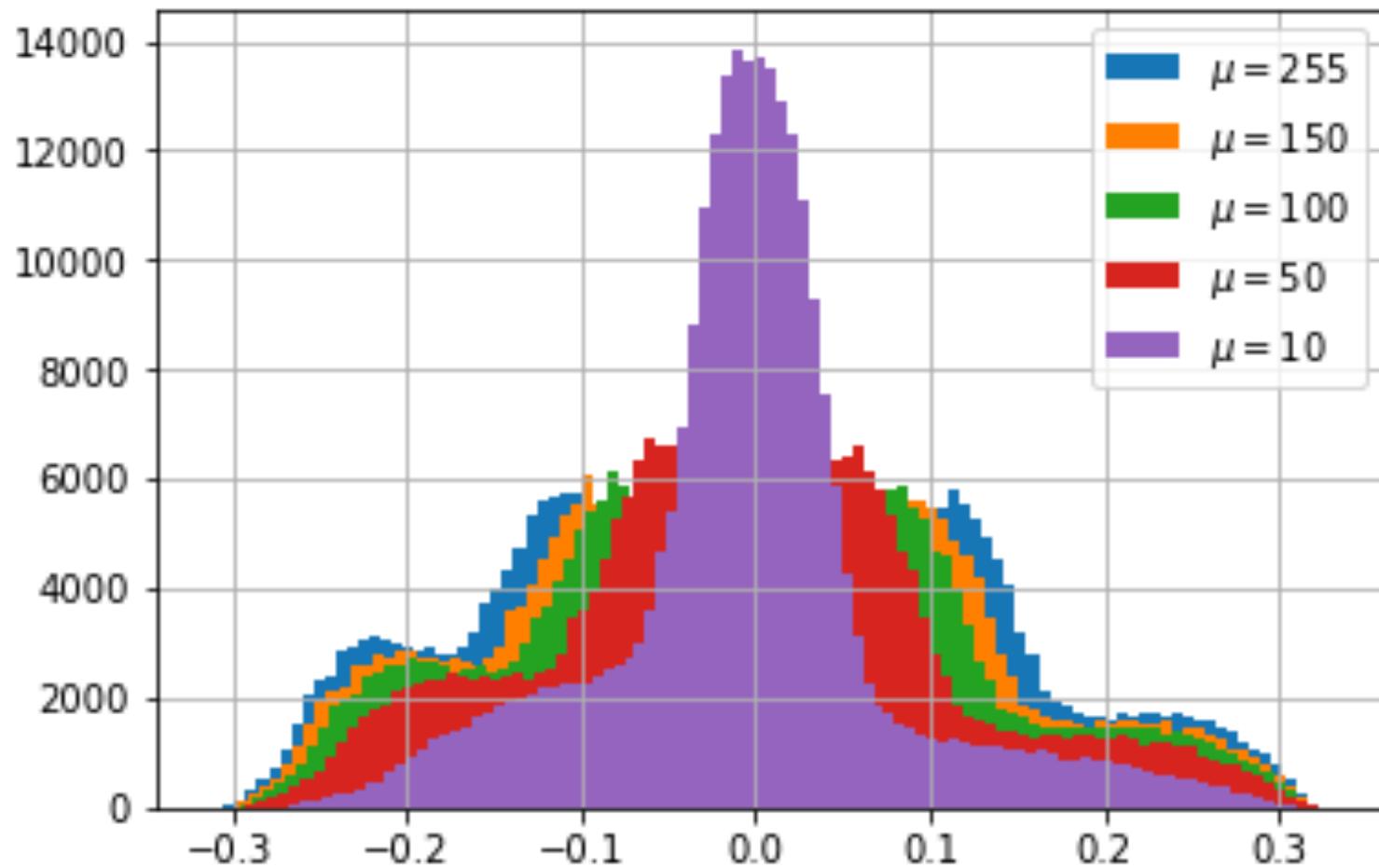
## Transformación ley mu
ymu= mulaw(x,255)
plt.plot(t,x,t,ymu)
plt.xlabel('t(s)')
plt.show()

## Histograma de la señal transformada
plt.hist(ymu, bins=100)
plt.title('Histograma – Ley $\mu$')
plt.axis([-0.3, 0.3, 0, 7000])
plt.grid()
plt.show()
```

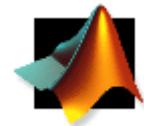
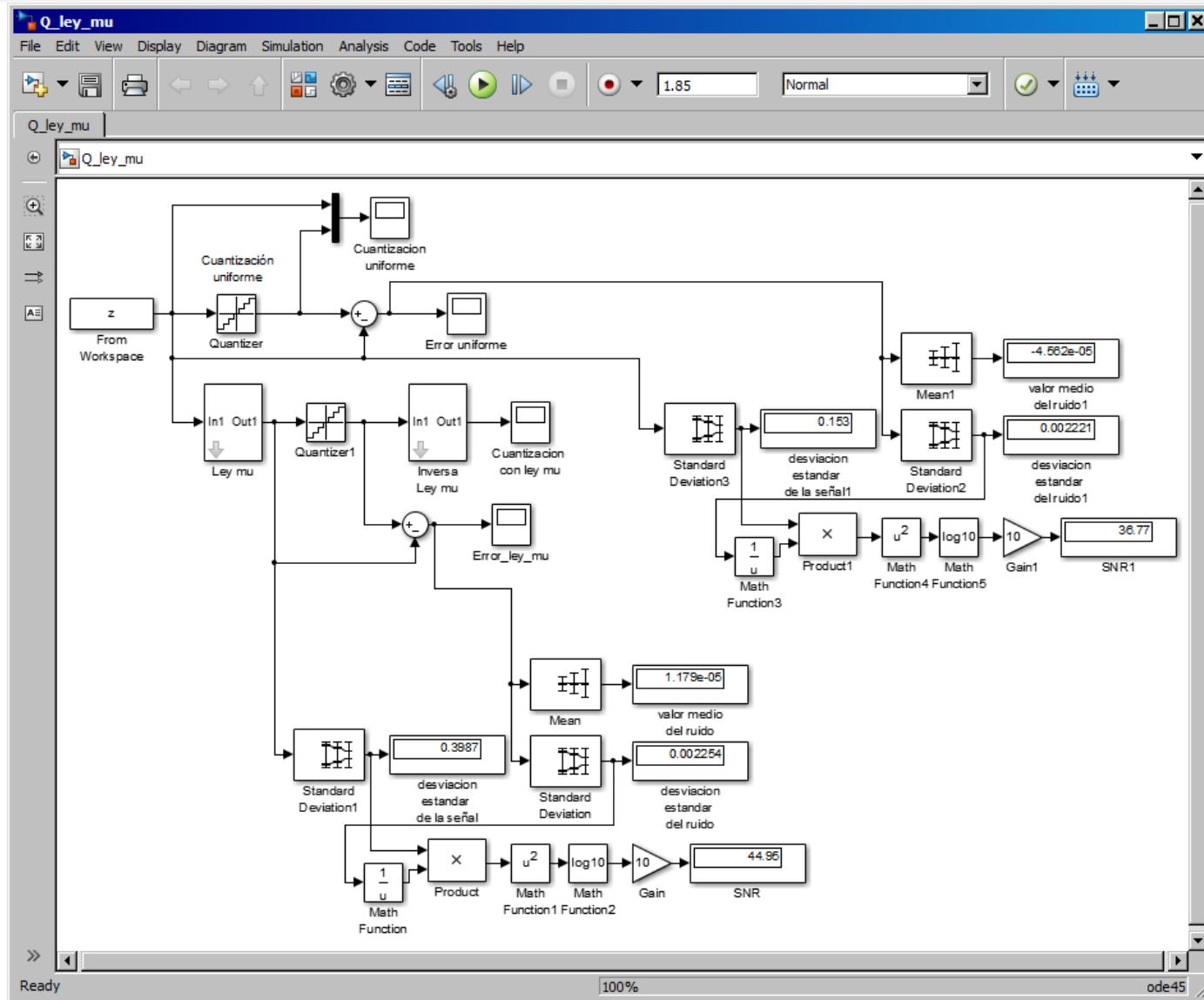
# Ejemplo: Ley $\mu$



# Ejemplo: Ley $\mu$

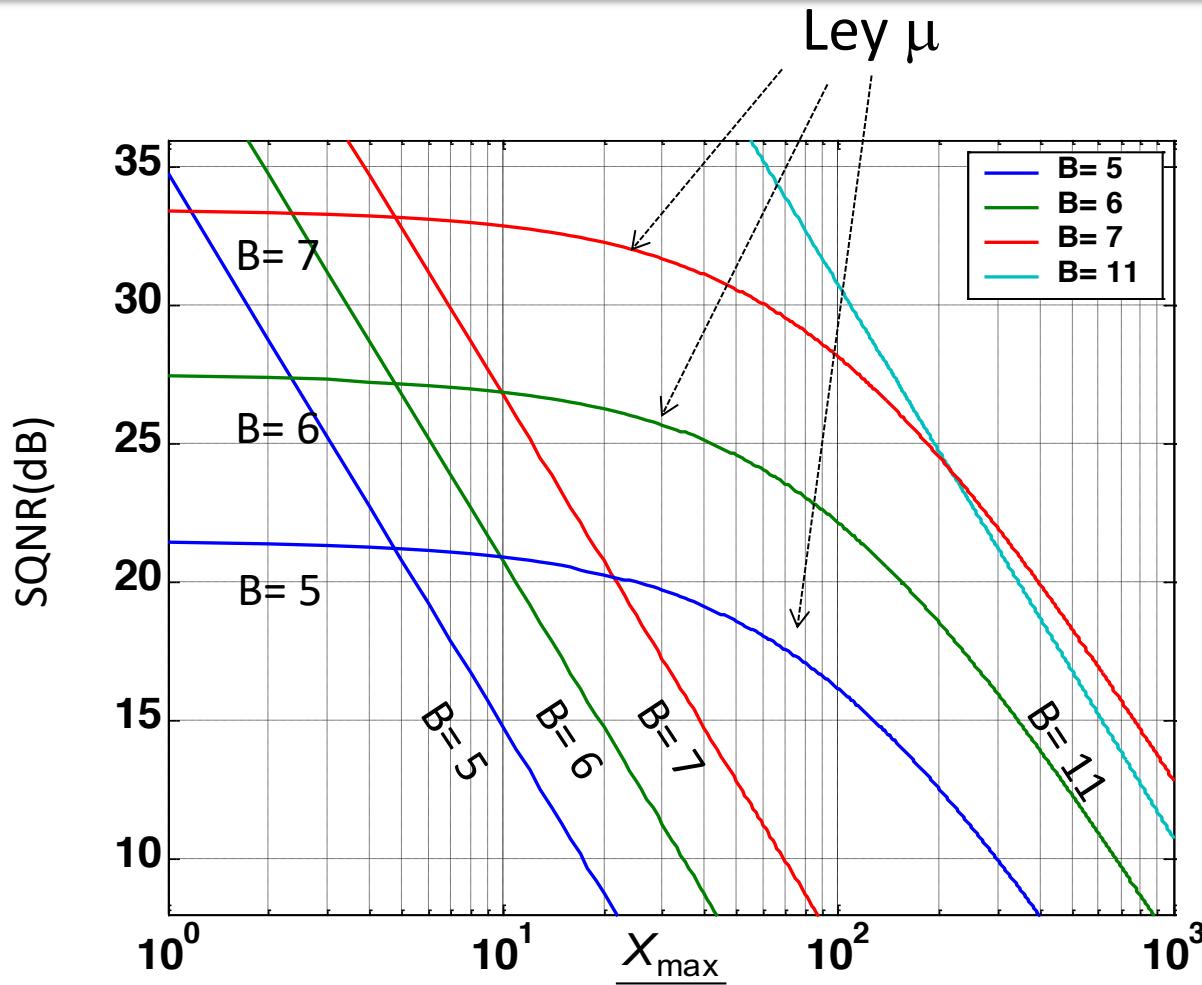


# Simulación: Compresión logarítmica mediante ley $\mu$



sim\_ley\_mu\_ini.m

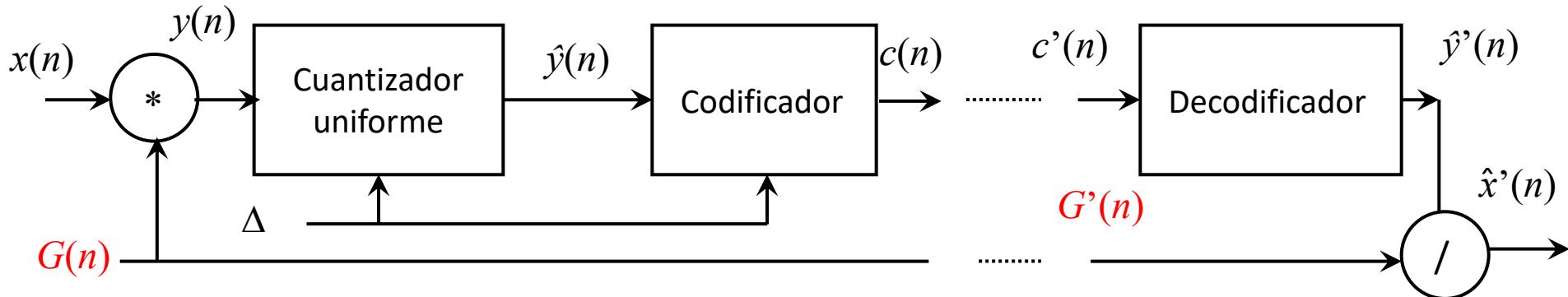
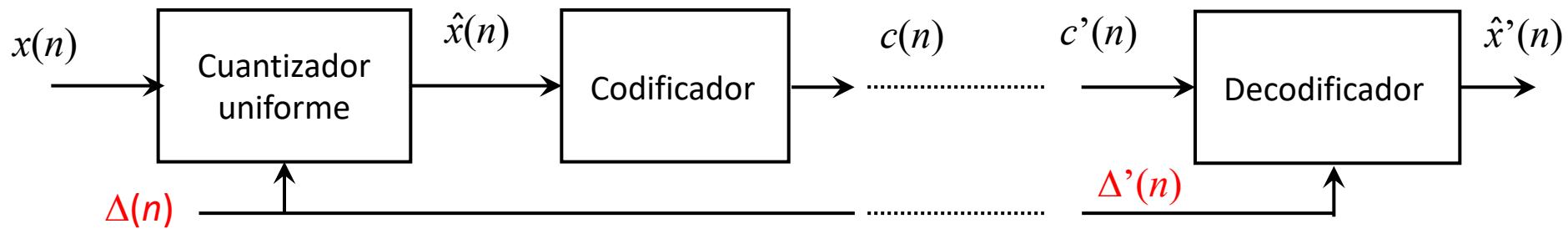
# Relación señal ruido de cuantización



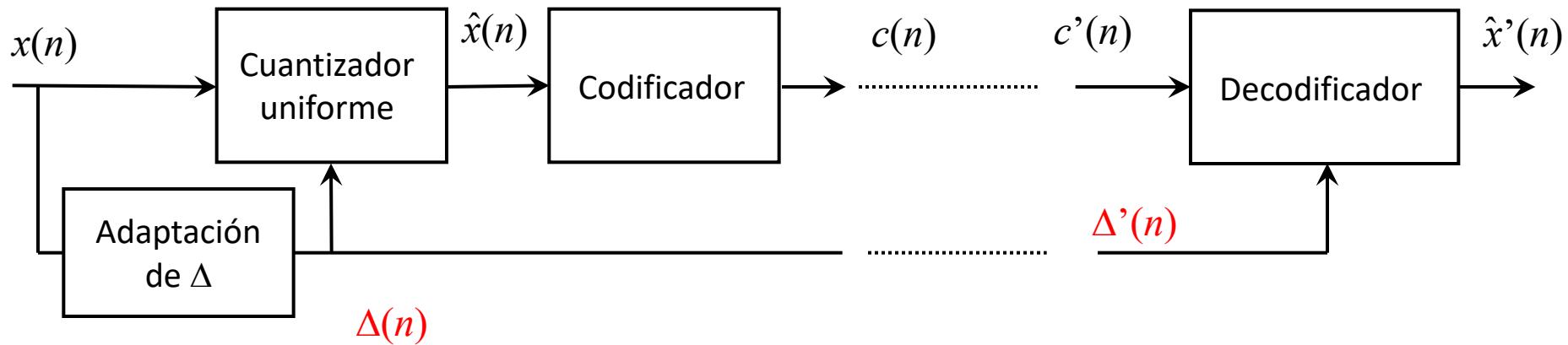
$$SQNR(dB) = 6B + 4.77 - 20 \cdot \log_{10} [\ln(1 + \mu)] - 10 \log_{10} \left[ 1 + \left( \frac{X_{\max}}{\mu \sigma_x} \right)^2 + \sqrt{2} \frac{X_{\max}}{\mu \sigma_x} \right]$$

# Cuantización adaptable

- Permite que  $\Delta$  varíe para adaptarse a la varianza de la señal de entrada.
- Representa una alternativa al control automático de ganancia.



# Adaptación del cuanto



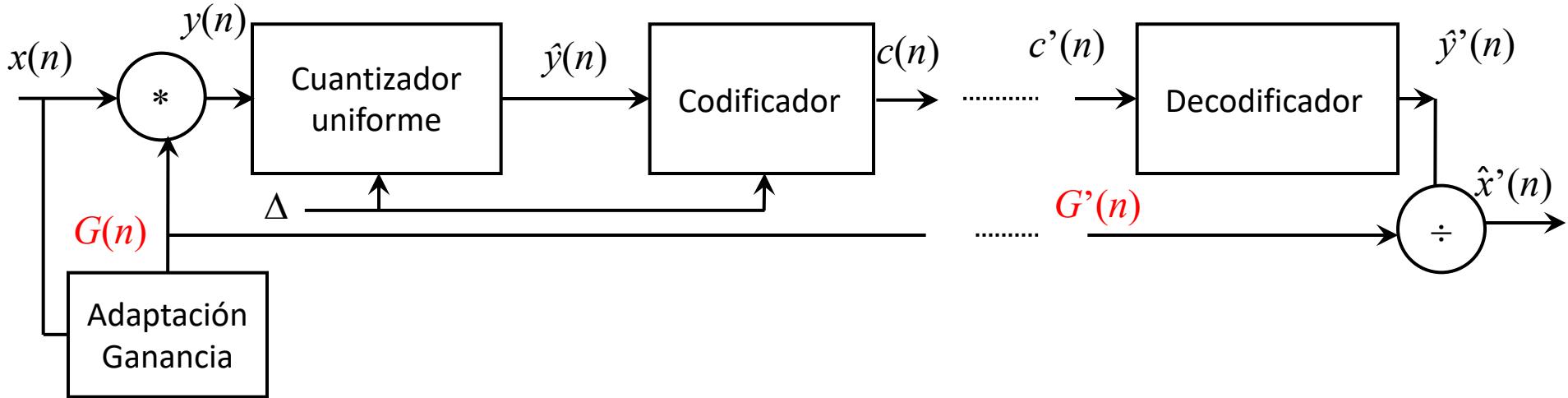
- $\Delta(n)$  se suele hacer proporcional a la desviación típica de  $x(n)$ :

$$\Delta(n) = \Delta_0 \cdot \sigma_x(n)$$

- Estimación de la varianza de  $x(n)$ :

$$\hat{\sigma}_x^2(n) = \frac{1}{M} \sum_{m=1}^M x^2(n-m)$$

# Adaptación de la ganancia



- $G(n)$  se suele hacer inversamente proporcional a la desviación típica de  $x(n)$ :

$$G(n) = G_0 / \sigma_x(n)$$

# Estimación de la desviación est醤dar

- Se emplea un **buffer** de  $M$  muestras de la señal:

$$[x(n-M), x(n-M+1), \dots, x(n-2), x(n-1)]$$

- Varianzas estimadas en los instantes  $n$  y  $n-1$ :

$$\hat{\sigma}_x^2(n) = \frac{1}{M} \sum_{m=1}^M x^2(n-m) \quad ; \quad \hat{\sigma}_x^2(n-1) = \frac{1}{M} \sum_{m=1}^M x^2(n-1-m)$$

- Restando ambas:

$$\hat{\sigma}_x^2(n) = \frac{1}{M} [x^2(n-1) + x^2(n-2) + \dots + x^2(n-M+1) + x^2(n-M)]$$

$$-\hat{\sigma}_x^2(n-1) = \frac{1}{M} [x^2(n-2) + x^2(n-3) + \dots + x^2(n-M) + x^2(n-M-1)]$$

---

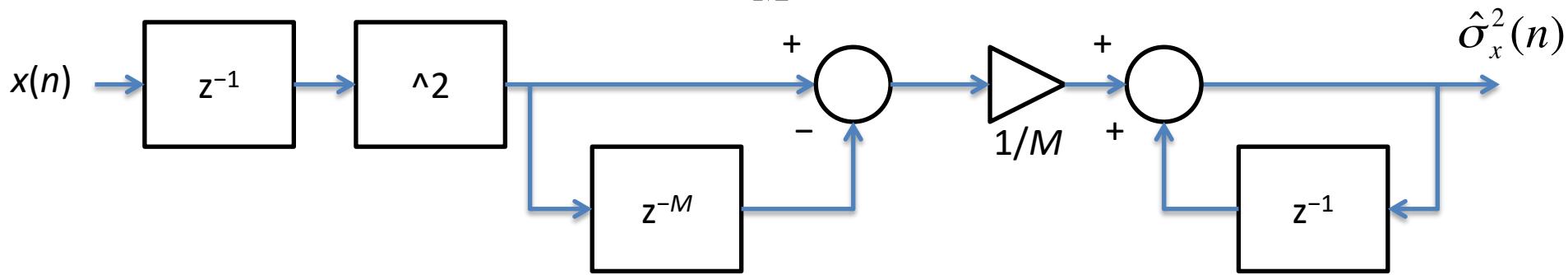
$$\hat{\sigma}_x^2(n) - \hat{\sigma}_x^2(n-1) = \frac{1}{M} [x^2(n-1) - x^2(n-M-1)]$$

Fórmula iterativa  
para estimación de  
la varianza

# Estimación de la desviación estándar

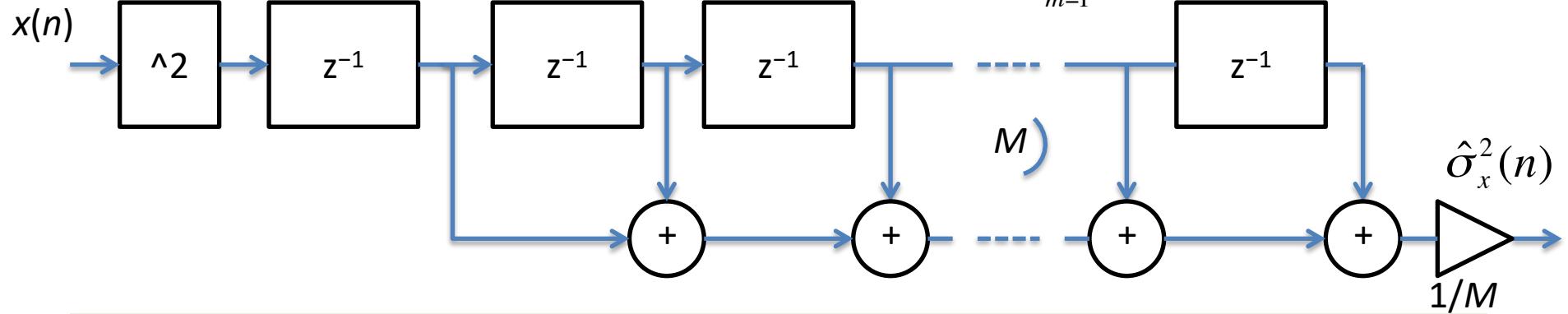
## ■ Implementación recursiva:

$$\hat{\sigma}_x^2(n) = \hat{\sigma}_x^2(n-1) + \frac{1}{M} [x^2(n-1) - x^2(n-M-1)]$$



## ■ Implementación directa:

$$\hat{\sigma}_x^2(n) = \frac{1}{M} \sum_{m=1}^M x^2(n-m)$$



# Ej.: Estimación de la desviación estándar

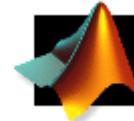
```
% Lectura del fichero WAV.  
wav_file= 'OSR_us_000_0010_8k.wav';  
[x,Fs,NBits]=wavread(wav_file);  
  
% Seleccionamos un segmento corto  
T= 25000;  
x= x(1:T);  
Nmues= length(x);  
  
% Inicializacion de la varianza.  
var= zeros(Nmues,1);  
  
% Número de muestras del buffer  
% Estimación de la varianza en  
% tiempo real  
  
N=30;  
var= est_var(x,N);
```

$$\hat{\sigma}_x^2(n) = \frac{1}{M} \sum_{m=1}^M x^2(n-m)$$

$$\hat{\sigma}_x^2(n) = \hat{\sigma}_x^2(n-1) + \frac{1}{M} [x^2(n-1) - x^2(n-M-1)]$$

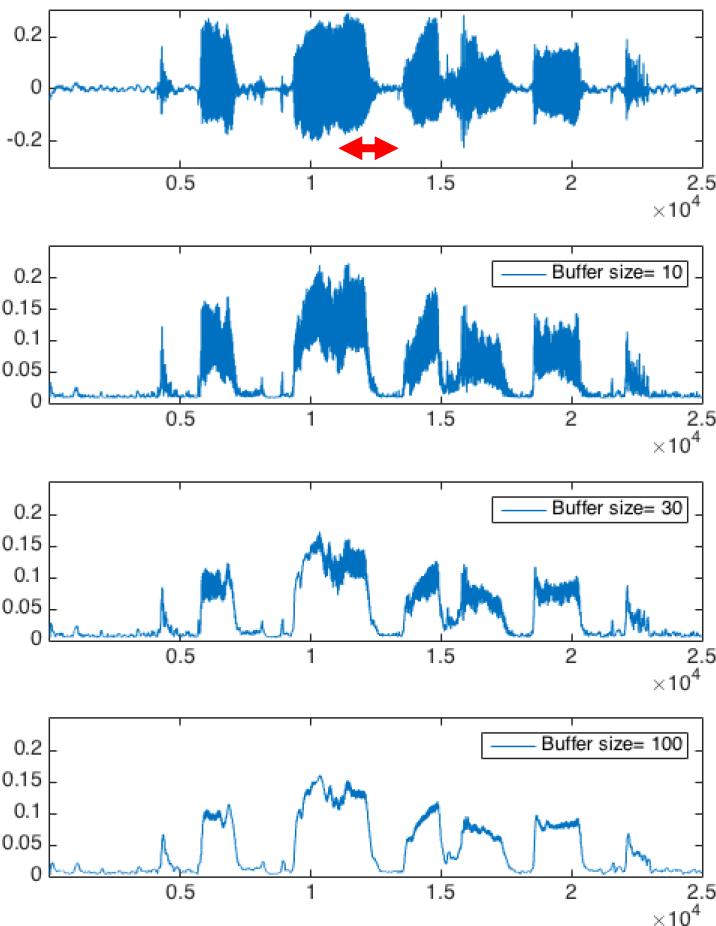
est\_var.m

```
function varianza= est_var(x,M)  
  
Nmues= length(x);  
  
varianza= zeros(Nmues,1);  
  
varianza(1)= 1/M*x(1)^2;  
  
for n=2:Nmues  
    if n>(M+1)  
        varianza(n)= varianza(n-1) + 1/M*(x(n-1)^2-x(n-M-1)^2);  
    else  
        varianza(n)= varianza(n-1)+1/M*x(n-1)^2;  
    end  
end
```



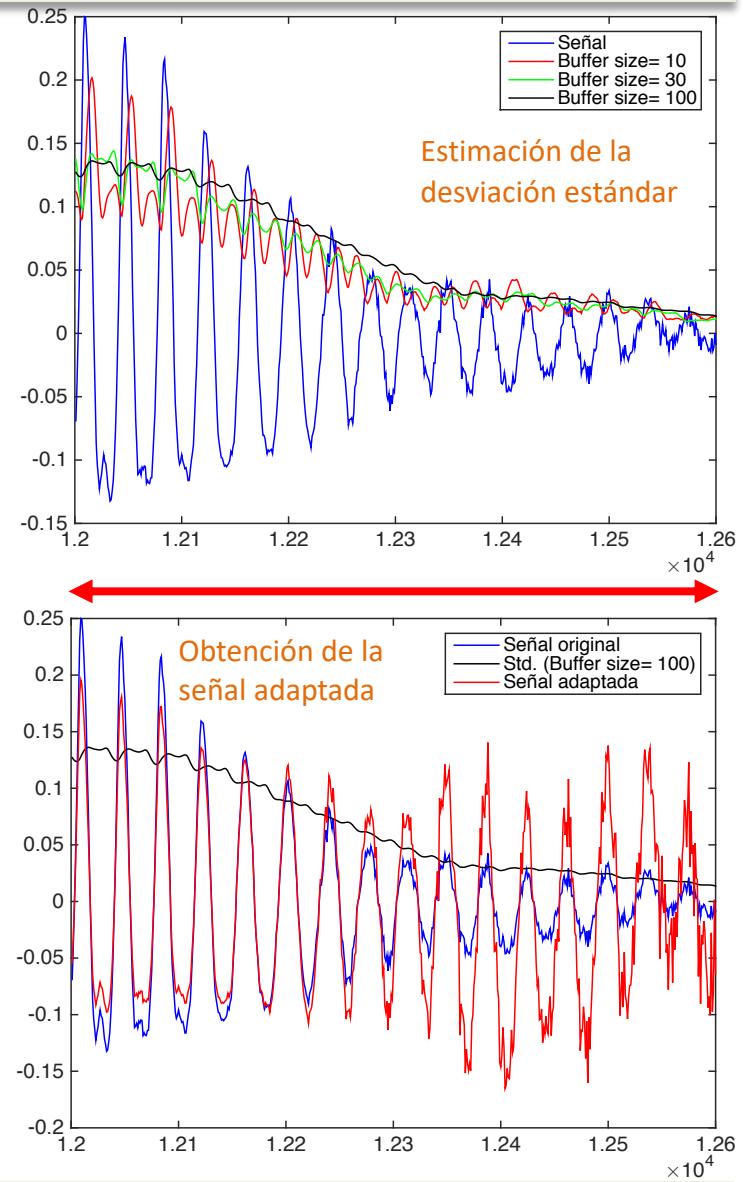
estimacion\_varianza.m

# Ej.: Estimación de la desviación estándar

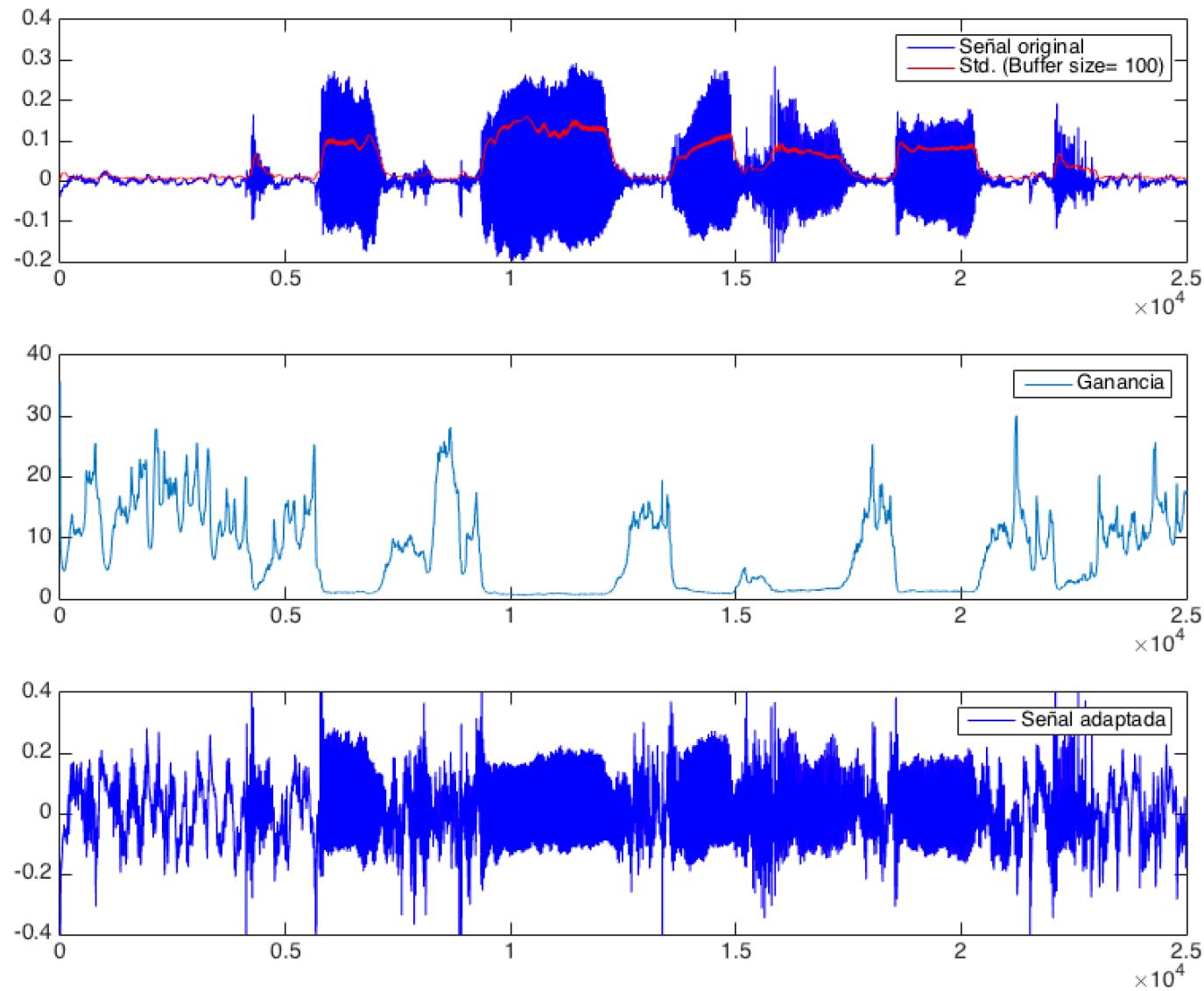


ADAPTACIÓN DE GANANCIA

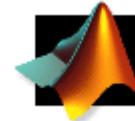
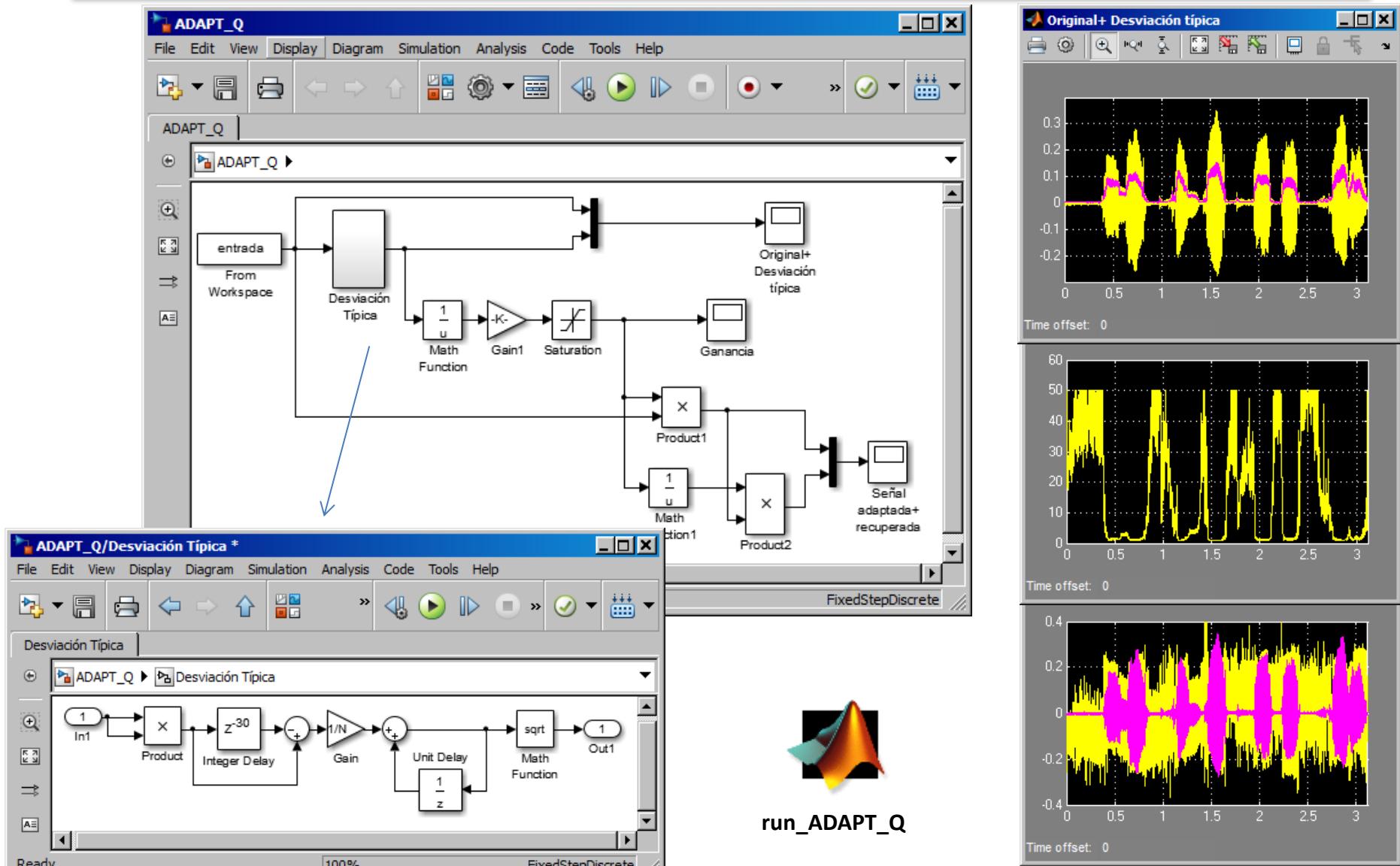
$$G(n) = G_0 / \sigma_x(n) \quad \text{con } G_0 = 0.1$$



# Ej.: Estimación de la desviación estándar



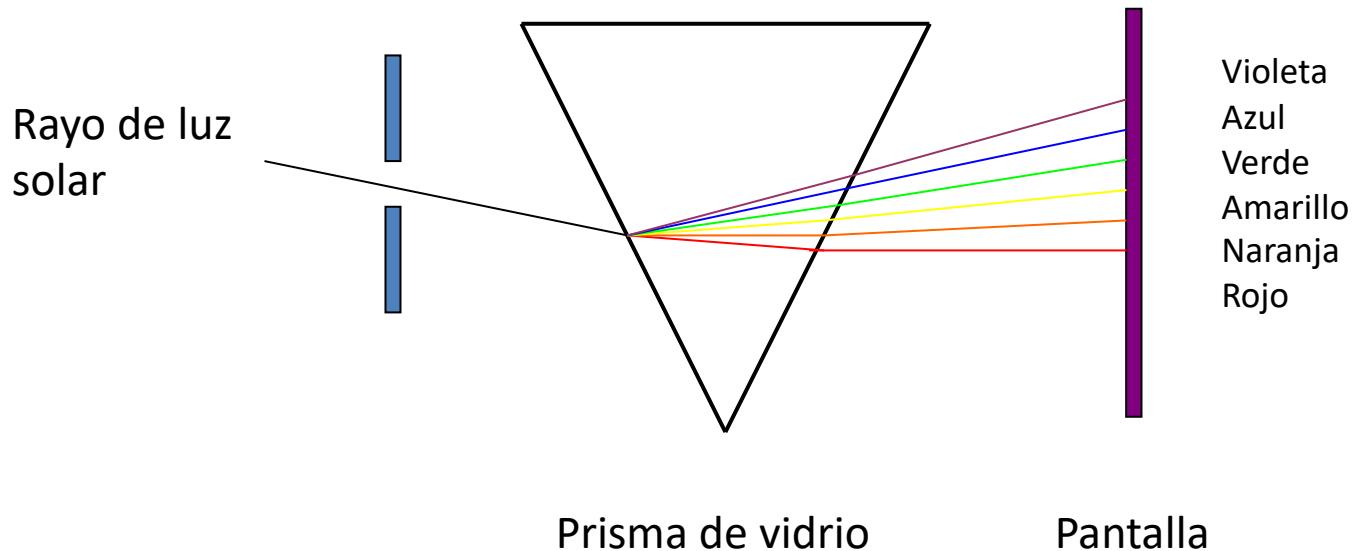
# Ejemplo de simulación: Cuantización adaptable



run\_ADAPTER\_Q

## 5. Análisis de señales en el dominio de la frecuencia

- Ejemplo:
  - Luz blanca que pasa a través de un prisma.



# Introducción a la transformada de Fourier

- Es una de las **herramientas más útiles** en procesado de señal.
- Se basa en la **descomposición** de una señal en términos de un conjunto de **funciones base** (sinusoides de diferente frecuencia).
- **Clasificación** técnicas de descomposición en frecuencias:

- **Señales continuas** (analógicas):
  - Periódicas: **Series de Fourier.**
  - No periódicas: **Transformada de Fourier.**

Señales y sistemas en tiempo discreto

- **Señales discretas** (digitales):
  - Periódicas: **Series de Fourier en tiempo discreto (DTFS).**
  - No periódicas: **Transformada de Fourier en tiempo discreto (DTFT).**

# Introducción a la transformada de Fourier

- **Ejemplo:** Consideremos la señal

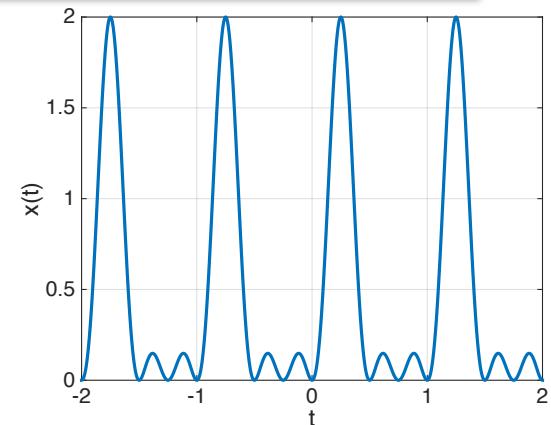
$$x(t) = \sin^2(\omega_0 t) + \sin^3(\omega_0 t)$$

- Utilizando las expresiones trigonométricas:

$$\sin^2(\omega_0 t) = \frac{1}{2} - \frac{1}{2} \cos(2\omega_0 t) \quad \sin^3(\omega_0 t) = \frac{3}{4} \sin(\omega_0 t) - \frac{1}{4} \sin(3\omega_0 t)$$

- Podemos expresar  $x(t)$  como:

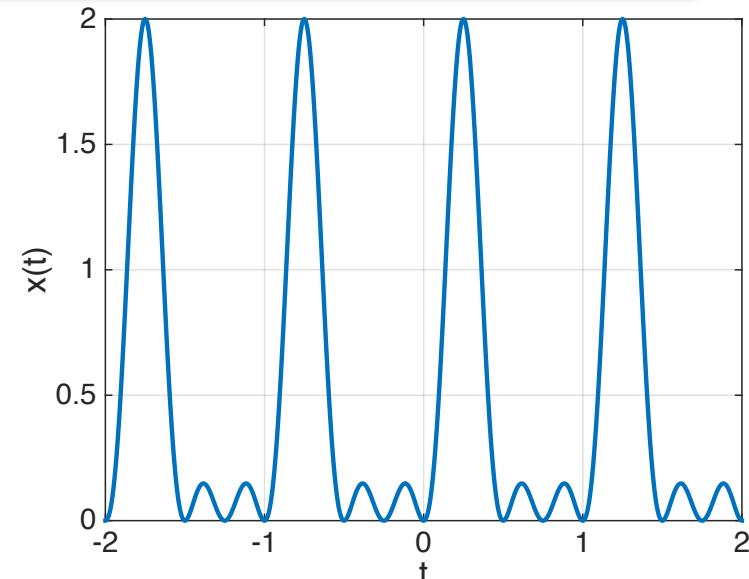
$$x(t) = \frac{1}{2} + \frac{3}{4} \sin(\omega_0 t) - \frac{1}{2} \cos(2\omega_0 t) - \frac{1}{4} \sin(3\omega_0 t)$$



# Introducción a la transformada de Fourier

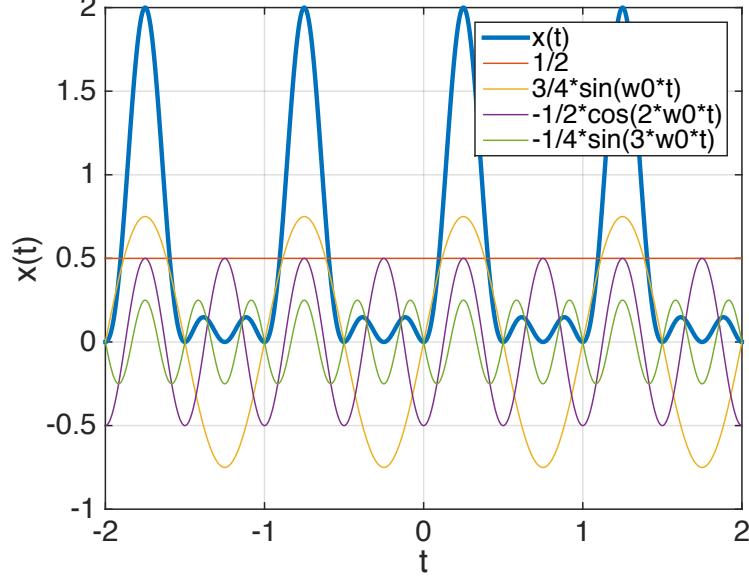
- Señal original:

$$x(t) = \sin^2(\omega_0 t) + \sin^3(\omega_0 t)$$



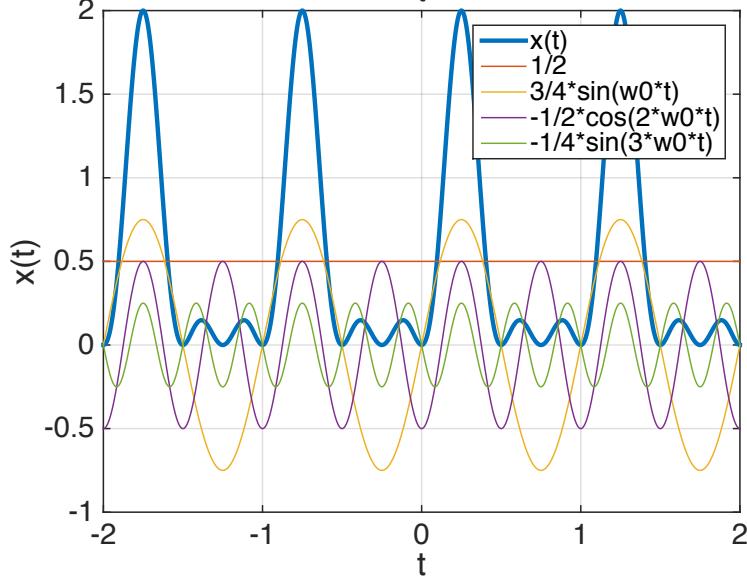
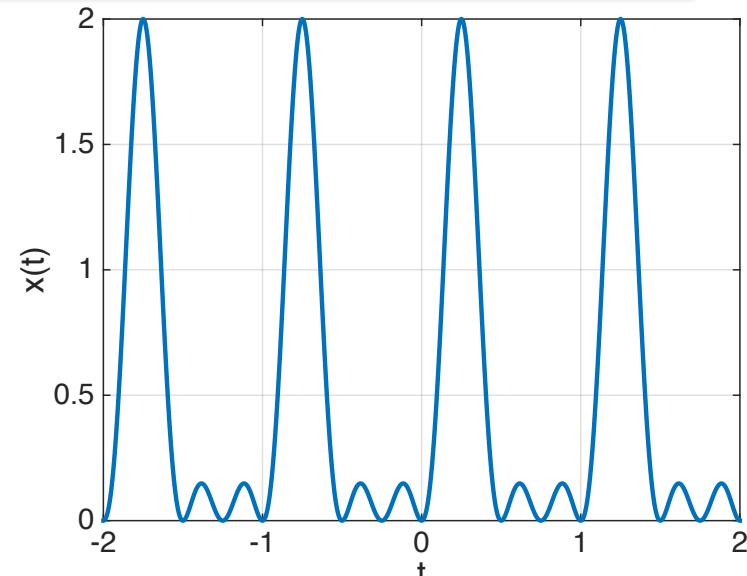
- Señal sintetizada:

$$x(t) = \frac{1}{2} + \frac{3}{4} \sin(\omega_0 t) - \frac{1}{2} \cos(2\omega_0 t) - \frac{1}{4} \sin(3\omega_0 t)$$



# Introducción a la transformada de Fourier

```
F0= 1;  
w0= 2*pi*F0;  
  
t= -2:0.001:2;  
  
x= sin(w0*t).^2+sin(w0*t).^3;  
  
plot(t,x); grid;  
xlabel('t');  
ylabel('x(t)');  
  
x1= 1/2*ones(1,length(t));  
x2= 3/4*sin(w0*t);  
x3= -1/2*cos(2*w0*t);  
x4= -1/4*sin(3*w0*t);  
  
xp= x1+x2+x3+x4;  
  
plot(t,xp,t,x1,t,x2,t,x3,t,x4);  
xlabel('t');  
ylabel('x(t)');  
legend('x(t)', '1/2', '3/4*sin(w0*t)',...  
      '-1/2*cos(2*w0*t)', '-1/4*sin(3*w0*t)');  
grid;
```



# Series de Fourier

- Toda **señal continua**  $x(t)$  de periodo  $T_p=1/F_0$  se puede expresar por medio una serie de Fourier:

$$x(t) = \sum_{k=-\infty}^{+\infty} c_k e^{j2\pi k F_0 t} ; \quad c_k = \frac{1}{T_p} \int_0^{T_p} x(t) e^{-j2\pi k F_0 t} dt$$

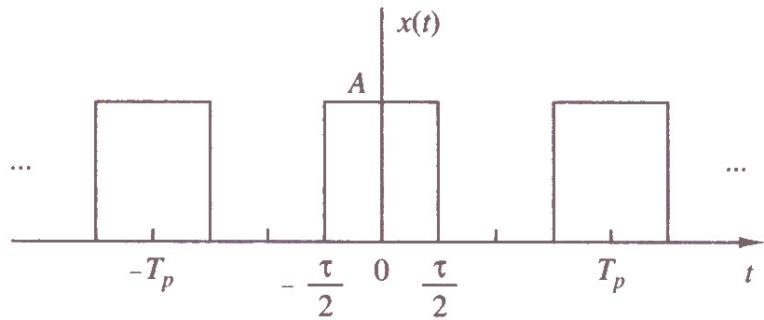
**Equivalentemente** (en términos de funciones seno y coseno):

$$x(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(2\pi k F_0 t) - b_k \sin(2\pi k F_0 t)]$$
$$a_0 = c_0 ; \quad a_k = 2|c_k| \cos \theta_k ; \quad b_k = 2|c_k| \sin \theta_k$$

siendo:

$$c_k = |c_k| e^{j\theta_k}$$

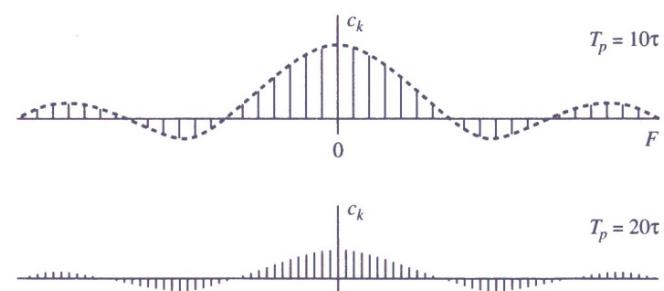
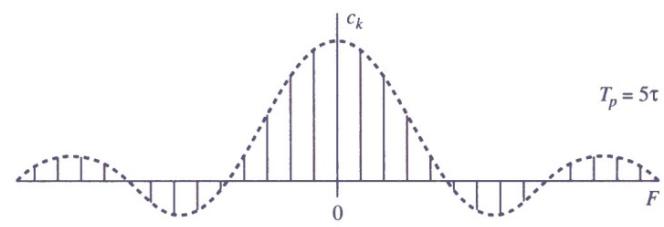
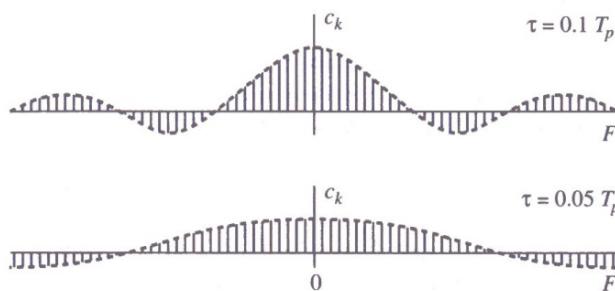
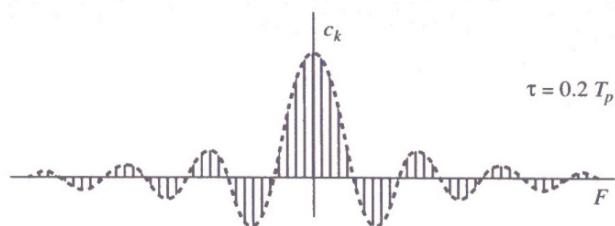
# Ejemplo 1: Tren periódico de pulsos rectangulares (1/3)



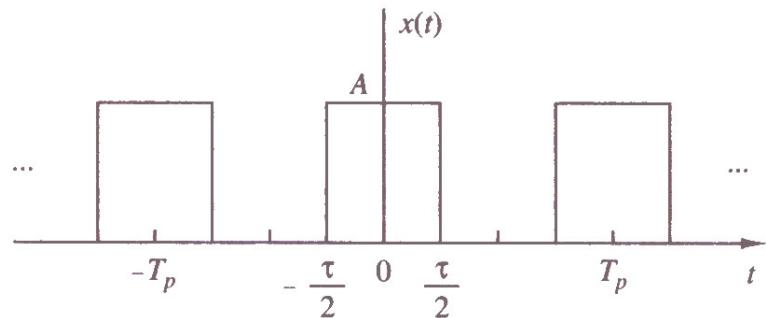
$$c_0 = \frac{1}{T_p} \int_{-T_p/2}^{T_p/2} x(t) dt = \frac{1}{T_p} \int_{-\tau/2}^{\tau/2} A dt = \frac{A \tau}{T_p}$$

$$c_k = \frac{1}{T_p} \int_{-\tau/2}^{\tau/2} A e^{-j2\pi k F_0 t} dt = \frac{A}{T_p} \left. \frac{e^{-j2\pi k F_0 t}}{-j2\pi k F_0} \right|_{-\tau/2}^{\tau/2}$$

$$= \frac{A \tau}{T_p} \frac{\sin(\pi k F_0 \tau)}{\pi k F_0 \tau} \quad k = \pm 1, \pm 2, \dots$$



# Ejemplo 1: Tren periódico de pulsos rectangulares (2/3)



$$x(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(2\pi k F_0 t) - b_k \sin(2\pi k F_0 t)]$$

$$a_0 = c_0 \quad ; \quad a_k = 2|c_k| \cos \theta_k \quad ; \quad b_k = 2|c_k| \sin \theta_k$$

$$c_0 = \frac{A\tau}{T_p}$$

$$a_0 = c_0 = \frac{A\tau}{T_p}$$

$$c_k = \frac{A\tau}{T_p} \frac{\sin(\pi k F_0 \tau)}{\pi k F_0 \tau}$$



$$a_k = 2|c_k| \cos \theta_k = 2|c_k| \cos 0 = 2|c_k| = 2 \frac{A\tau}{T_p} \frac{\sin(\pi k F_0 \tau)}{\pi k F_0 \tau} \quad k = 1, 2, \dots$$

$$b_k = 2|c_k| \sin \theta_k = 2|c_k| \sin 0 = 0$$

Caso particular:  $\tau = T_p/2$

$$a_0 = A/2$$

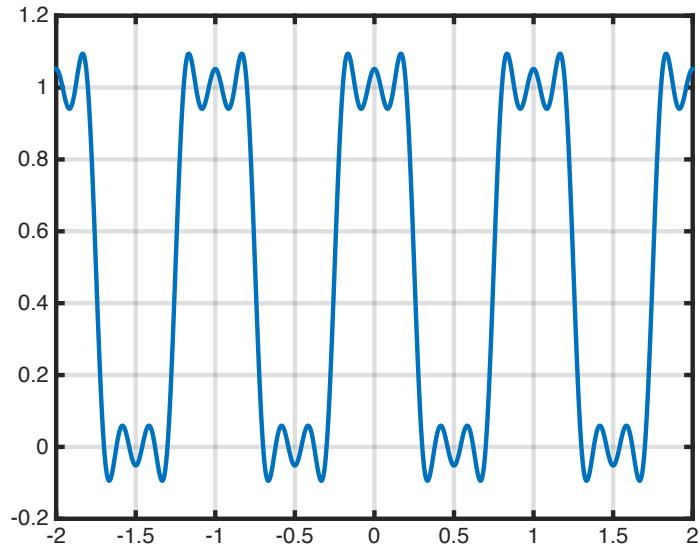
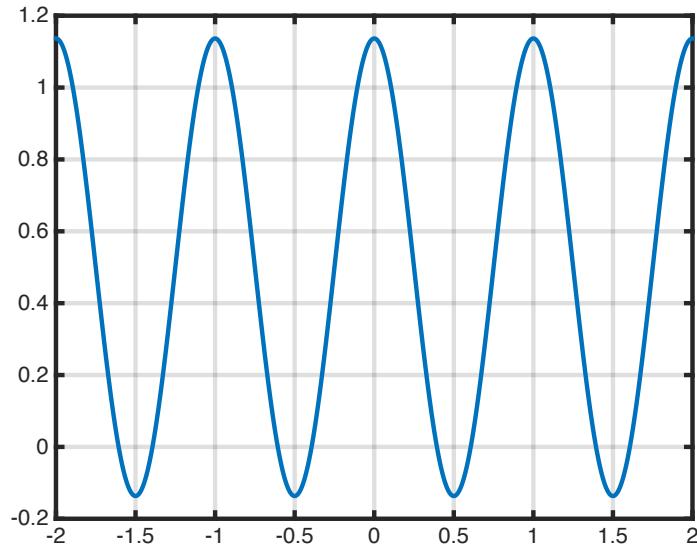
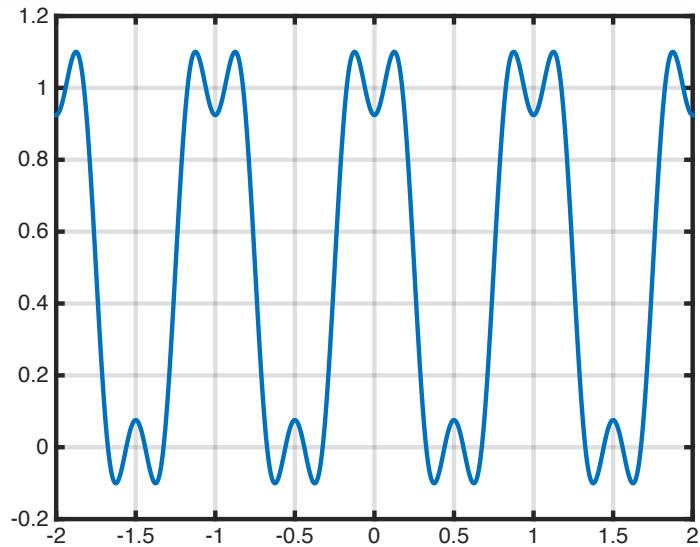
$$a_k = A \frac{\sin(\pi k / 2)}{\pi k / 2} \quad k = 1, 2, \dots$$

$$b_k = 0$$

$$x(t) = \frac{A}{2} + \frac{2A}{\pi} \left[ \cos(2\pi F_0 t) - \frac{1}{3} \cos(2\pi 3F_0 t) + \frac{1}{5} \cos(2\pi 5F_0 t) \dots \right]$$

# Ejemplo 1: Tren periódico de pulsos rectangulares (3/3)

```
A= 1;  
F0= 1;  
t= -2:0.001:2;  
  
x1= A/2+2*A/pi*(cos(2*pi*F0*t));  
x2= A/2+2*A/pi*(cos(2*pi*F0*t)-1/3*cos(2*pi*3*F0*t));  
x3= A/2+2*A/pi*(cos(2*pi*F0*t)-1/3*cos(2*pi*3*F0*t)+1/5*cos(2*pi*5*F0*t));  
  
figure(1); plot(t,x1); grid;  
figure(2); plot(t,x2); grid;  
figure(3); plot(t,x3); grid;
```



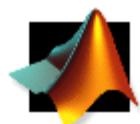
# Ejemplo 1 (continuación)

- La señal se encuentra mejor representada con un elevado número de armónicos.
- Al ser los coeficientes  $c_k$  reales,
  - $\theta_k = 0, \pi$  y  $b_k = 0 \quad \forall k$

$$c_k = A \frac{\tau}{T_p} \frac{\sin(\pi k \tau / T_p)}{\pi k \tau / T_p}$$

$$a_0 = c_0 \quad ; \quad a_k = 2|c_k| \cos \theta_k \quad ; \quad b_k = 2|c_k| \sin \theta_k$$

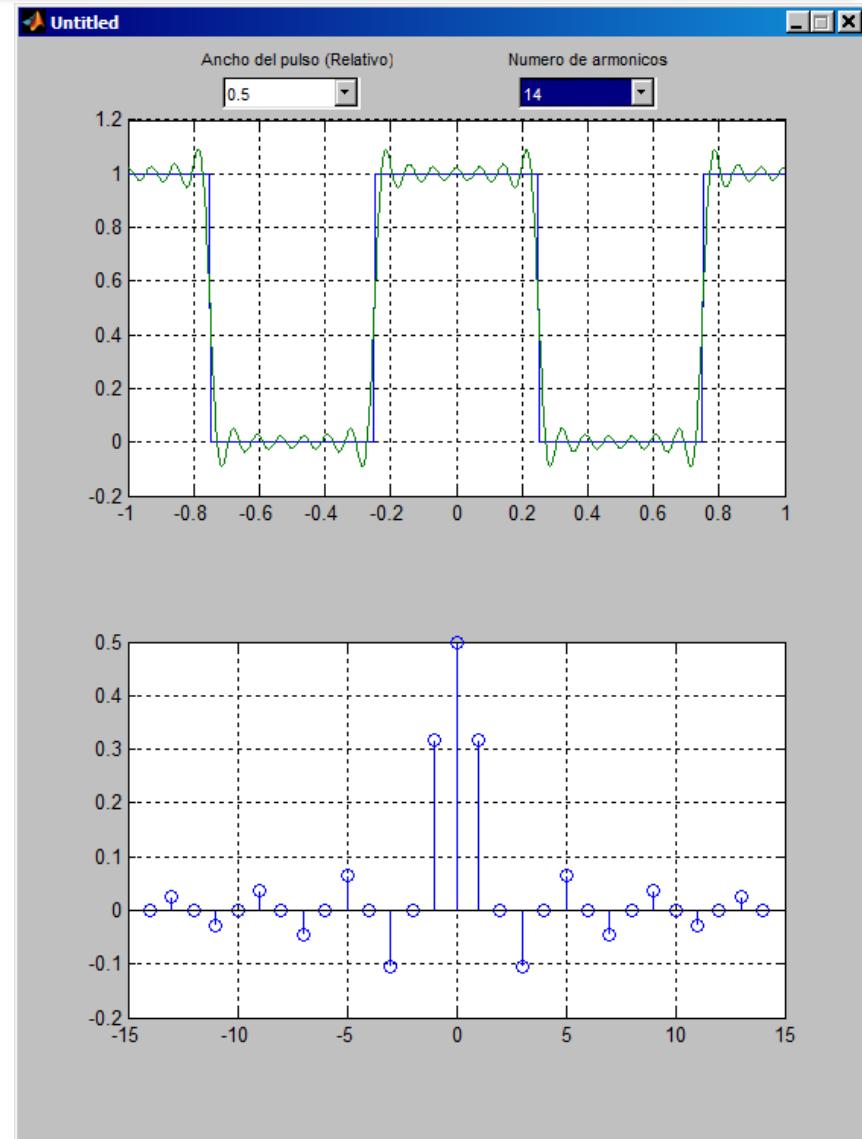
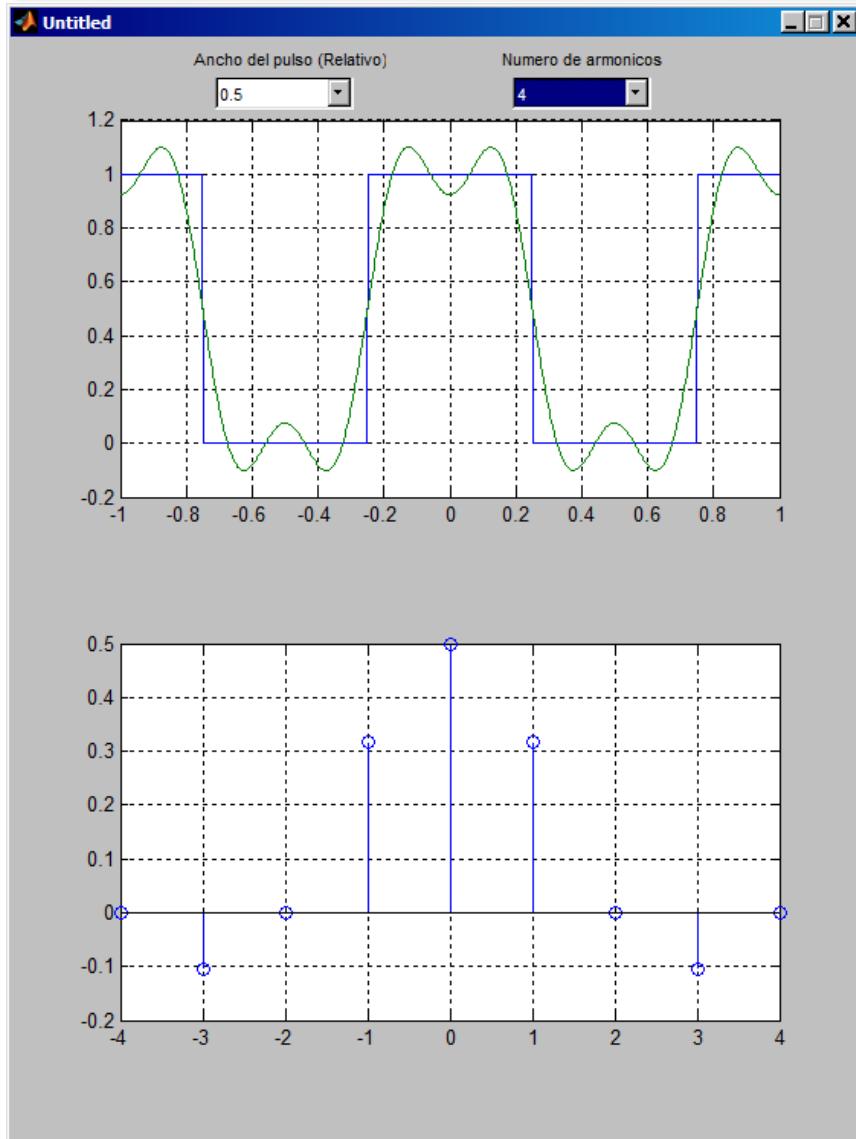
$$x(t) \cong \frac{A\tau}{T_p} + 2 \frac{A\tau}{T_p} \sum_{k=1}^N \frac{\sin(\pi k F_0 \tau)}{\pi k F_0 \tau} \cos(2\pi k t / T_p)$$



Aproximación  
(truncado)

sw\_Fourier\_series.m

# Ejemplo 1 (continuación)



# Series de Fourier

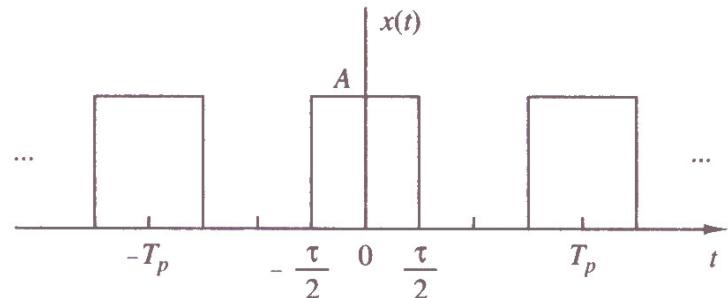
- Def.: **Potencia de una señal:**

$$P_x = \frac{1}{T_p} \int_0^{T_p} |x(t)|^2 dt = \sum_{k=-\infty}^{+\infty} |c_k|^2$$

**Teorema de Parseval**

- **Ejercicio:** Calcular la potencia de  $x(t)$ .

$$P_x = \frac{1}{T_p} \int_{-\tau/2}^{+\tau/2} A^2 dt = A^2 \frac{\tau}{T_p}$$



- Def.: **Densidad espectral de potencia:**

- Potencia del armónico  $kF_0$  de la señal

$$P_k = |c_k|^2$$

# Señales no periódicas

- Se define la **transformada de Fourier** de  $x(t)$  como:

$$X(F) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi F t} dt \quad ; \quad x(t) = \int_{-\infty}^{+\infty} X(F) e^{j2\pi F t} dF$$

- **Energía** de una señal:

$$E_x = \int_{-\infty}^{+\infty} |x(t)|^2 dt = \int_{-\infty}^{+\infty} |X(F)|^2 dF$$

- **Densidad espectral de energía:**

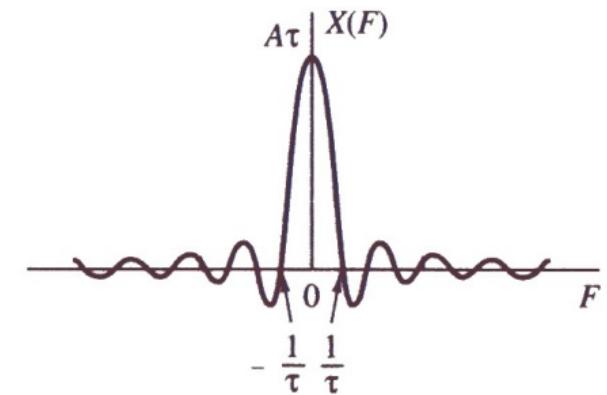
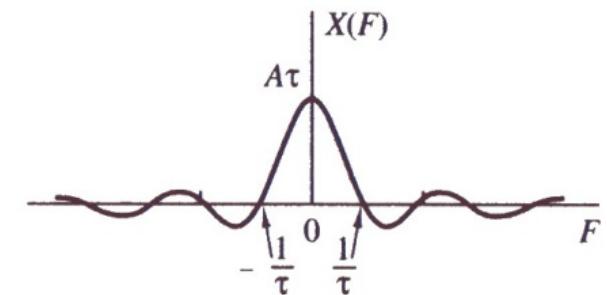
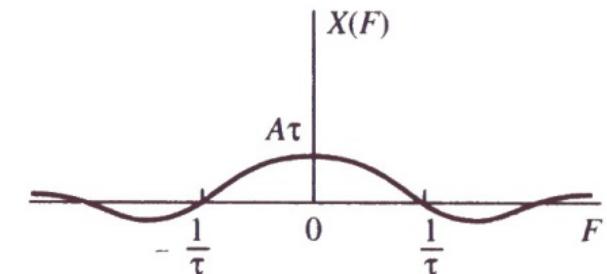
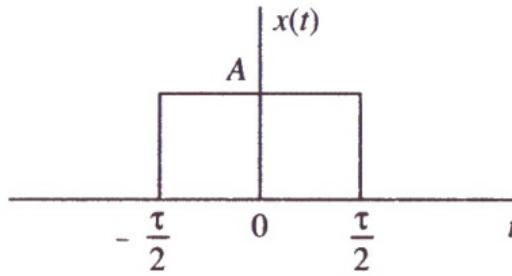
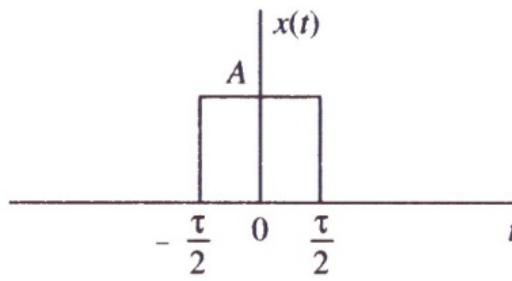
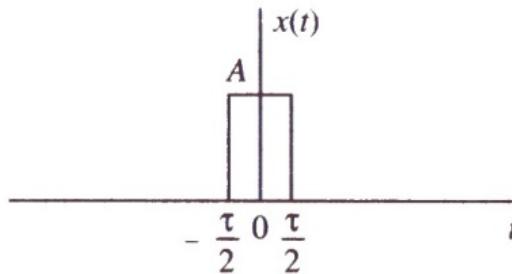
$$S_{xx}(F) = |X(F)|^2$$

# Ejemplo 1: Pulso rectangular

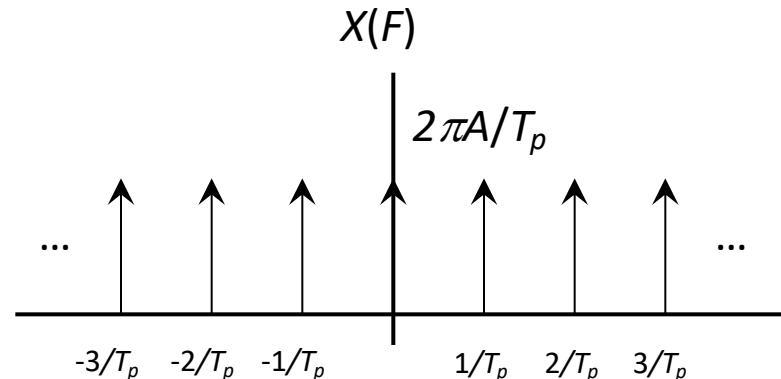
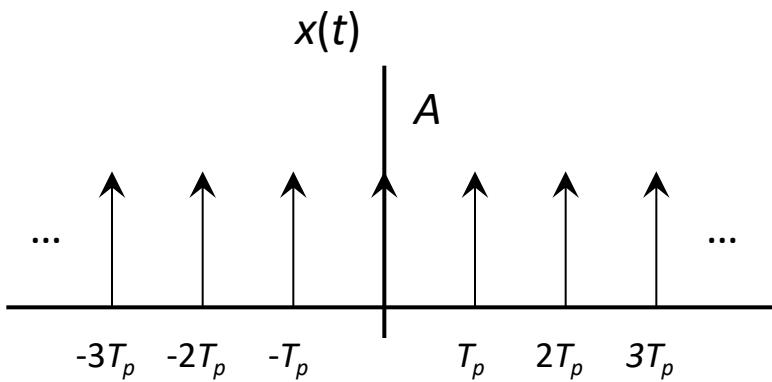
$$X(F) = \int_{-\tau/2}^{\tau/2} A e^{-j2\pi F t} dt$$

$$= A \tau \frac{\sin(\pi F \tau)}{\pi F \tau}$$

$$S_{xx}(F) = A^2 \tau^2 \left[ \frac{\sin(\pi F \tau)}{\pi F \tau} \right]^2$$



# Ejemplo 2: Tren de impulsos



$$x(t) = A \sum_{k=-\infty}^{+\infty} \delta(t - kT_p)$$

$$X(F) = \frac{2\pi A}{T_p} \sum_{k=-\infty}^{+\infty} \delta(F - \frac{k}{T_p})$$

# Propiedades de la transformada de Fourier

- Linealidad:
  - $\mathcal{F}\{a \cdot x_1(t) + b \cdot x_2(t)\} = a \cdot \mathcal{F}\{x_1(t)\} + b \cdot \mathcal{F}\{x_2(t)\}$
- Simetría:
  - $\mathcal{F}\{X(t)\} = X(-F)$
- Escalado:
  - $\mathcal{F}\{x(kt)\} = X(F/k)/k$
- Traslación en el tiempo:
  - $\mathcal{F}\{x(t-t_0)\} = \exp(-j2\pi F t_0) \cdot X(F)$
- Traslación en frecuencia:
  - $\mathcal{F}\{\exp(j2\pi F_0 t) \cdot x(t)\} = X(F-F_0)$
- Teorema de convolución:
  - $\mathcal{F}\{x(t) * h(t)\} = H(f)X(f)$
- Teorema de convolución en frecuencia:
  - $\mathcal{F}\{x(t) \cdot h(t)\} = H(f) * X(f)$
- Teorema de Parseval:
$$\int_{-\infty}^{+\infty} x^2(t) dt = \int_{-\infty}^{+\infty} |x(F)|^2 dF$$

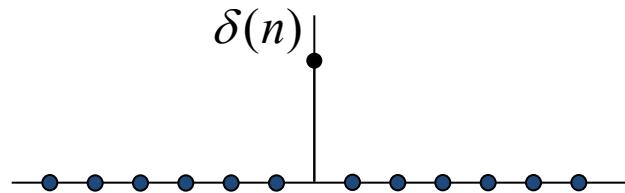
# 6. Señales y sistemas en tiempo discreto

- $x(n) \quad -\infty < n < \infty, n$  entero.

- Señales elementales

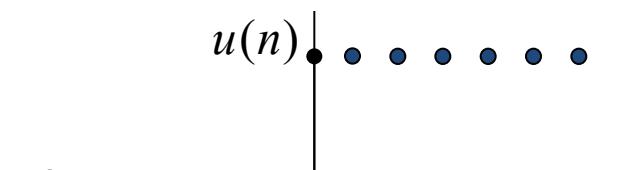
- Impulso unitario.

$$\delta(n) = \begin{cases} 1 & \text{para } n = 0 \\ 0 & \text{para } n \neq 0 \end{cases}$$



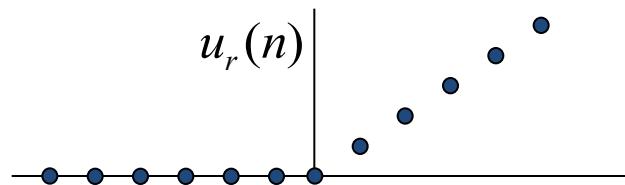
- Escalón unidad.

$$u(n) = \begin{cases} 1 & \text{para } n \geq 0 \\ 0 & \text{para } n < 0 \end{cases}$$



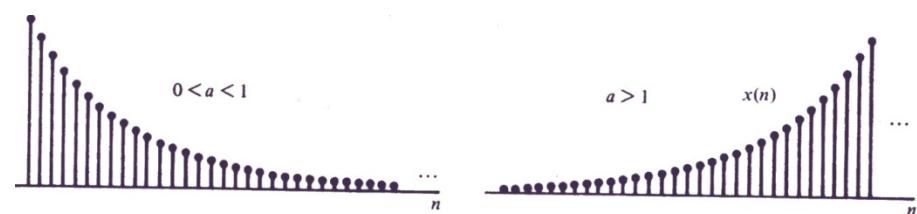
- Rampa unidad.

$$u_r(n) = \begin{cases} n & \text{para } n \geq 0 \\ 0 & \text{para } n < 0 \end{cases}$$



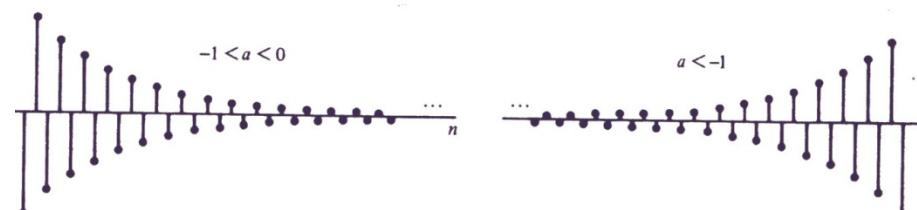
- Señal exponencial.

$$x(n) = a^n$$



- Señal periódica.

$$x(n) = x(n + N) \quad \forall n$$



- Señales aleatorias.

# Operaciones básicas con señales

- Suma:
  - $y(n) = x_1(n) + x_2(n)$
- Multiplicación:
  - $y(n) = x_1(n) \cdot x_2(n)$
- Desplazamiento:
  - $y(n) = x(n-k)$
- Inversión:
  - $y(n) = x(-n)$
- Energía de la señal:
- Potencia de la señal:

$$E_x = \sum_{n=-\infty}^{+\infty} x(n)x^*(n) = \sum_{n=-\infty}^{+\infty} |x(n)|^2$$

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$$

# Resultados interesantes

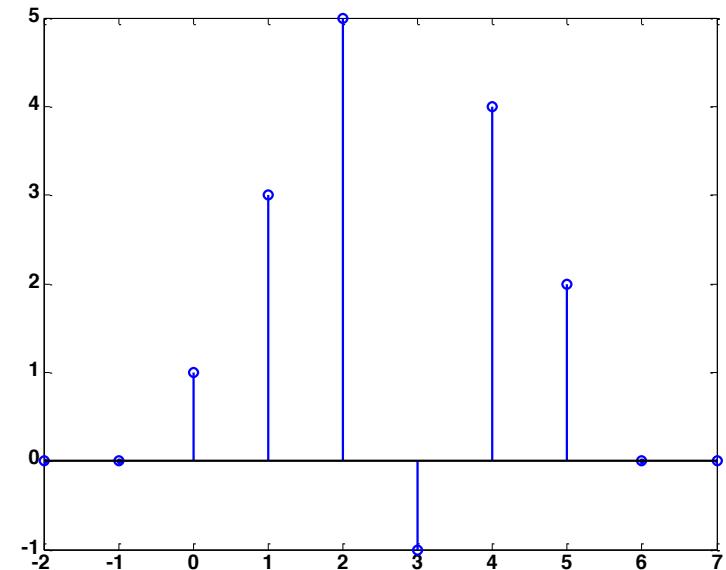
- Cualquier señal  $x(n)$  se puede expresar como una suma pesada de secuencias impulso,

$$x(n) = \sum_{k=-\infty}^{+\infty} x(k)\delta(n-k)$$

- Ejemplo:

$$x(n) = \{..., 0, \mathbf{1}, 3, 5, -1, 4, 2, 0, ....\}$$

$\uparrow$   
 $n=0$



$$x(n) = 1 \cdot \delta(n) + 3 \cdot \delta(n-1) + 5 \cdot \delta(n-2) - 1 \cdot \delta(n-3) + 4 \cdot \delta(n-4) + 2 \cdot \delta(n-5)$$

# Resultados interesantes

- Señal par o simétrica:

$$x_e(-n) = x_e(n)$$

- Señal impar o antisimétrica:

$$x_o(-n) = -x_o(n)$$

- Cualquier señal  $x(n)$  se puede descomponer en la suma de una señal par y otra impar:

$$x(n) = x_e(n) + x_o(n)$$

$$x_e(n) = \frac{1}{2} [x(n) + x(-n)]$$

$$x_o(n) = \frac{1}{2} [x(n) - x(-n)]$$

# Ejemplo: $x(n) = u(n) - u(n-10)$

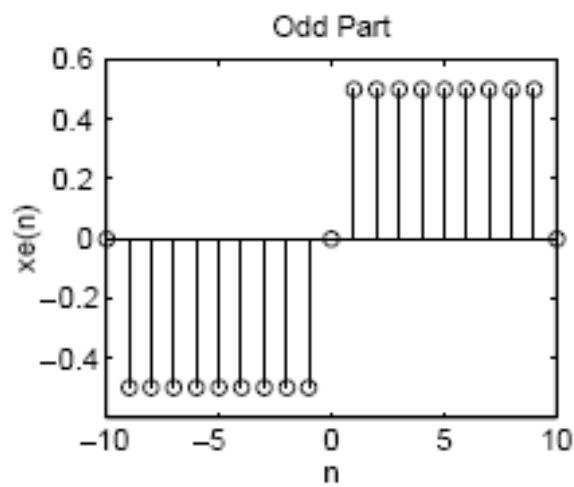
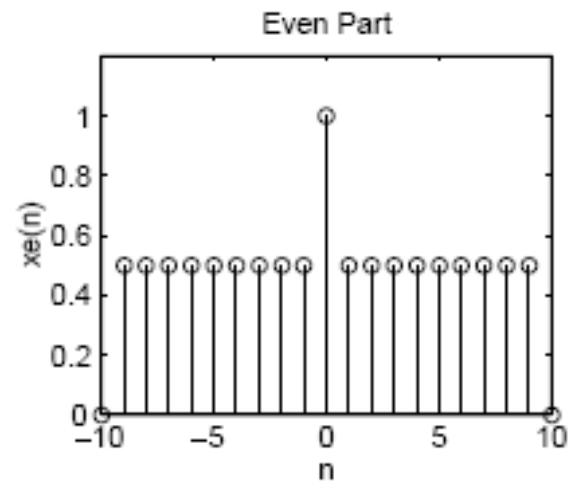
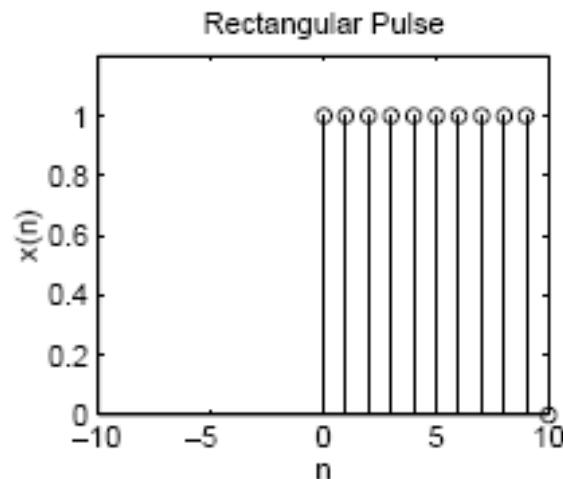
$$x(n) = x_e(n) + x_o(n)$$

$$x_e(n) = \frac{1}{2} [x(n) + x(-n)]$$

$$x_o(n) = \frac{1}{2} [x(n) - x(-n)]$$

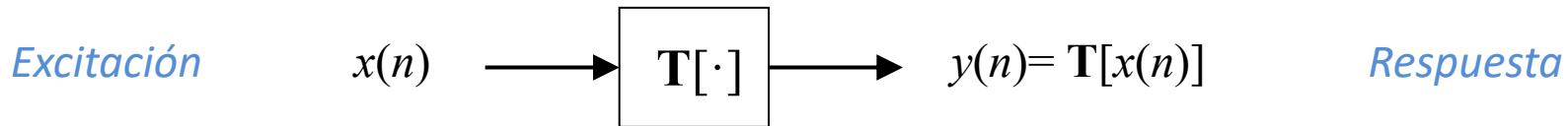
$$x_e(n) = \frac{1}{2} [u(n) - u(n-10) + u(-n) - u(-n-10)]$$

$$x_o(n) = \frac{1}{2} [u(n) - u(n-10) - u(-n) + u(-n-10)]$$

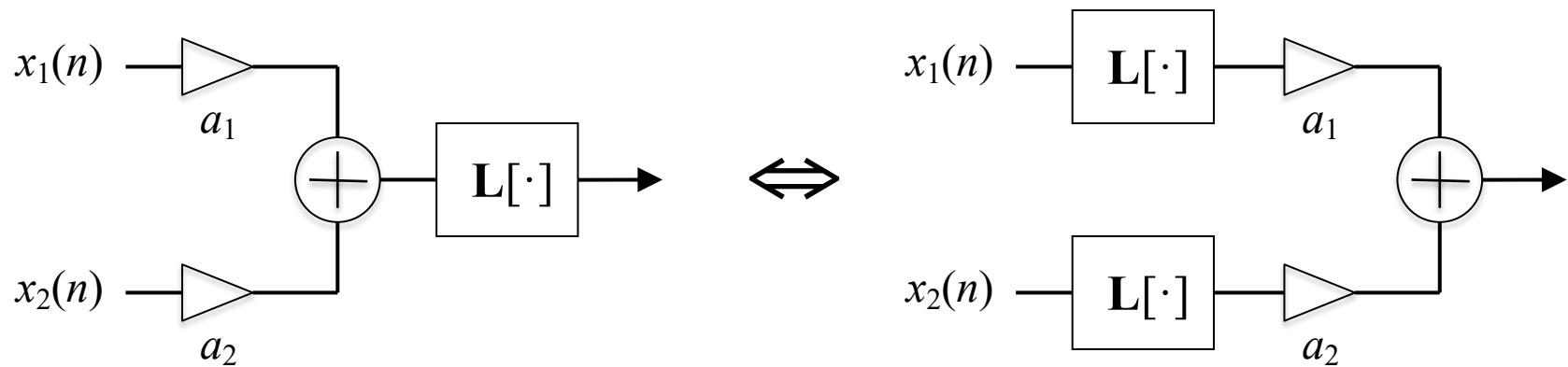


# Sistemas en tiempo discreto

- Se describen por medio de un operador  $\mathbf{T}[\cdot]$  que transforma una secuencia  $x(n)$  (excitación) en otra secuencia  $y(n)$  (respuesta):



- **Lineales:**  $\mathbf{L}[a_1x_1(n) + a_2x_2(n)] = a_1 \mathbf{L}[x_1(n)] + a_2 \mathbf{L}[x_2(n)]$



# Sistemas en tiempo discreto

- **Ejemplo:** Comprobar si los sistemas siguientes son lineales.

$$T[x(n)] = \sum_{k=n-n_0}^{n+n_0} x(k)$$
$$T[a_1x_1(n) + a_2x_2(n)] = \sum_{k=n-n_0}^{n+n_0} [a_1x_1(k) + a_2x_2(k)] =$$
$$= a_1 \sum_{k=n-n_0}^{n+n_0} x_1(k) + a_2 \sum_{k=n-n_0}^{n+n_0} x_2(k) =$$
$$= a_1 T[x_1(n)] + a_2 T[x_2(n)] \quad \text{LINEAL}$$

$$T[x(n)] = e^{x(n)}$$

$$T[a_1x_1(n) + a_2x_2(n)] = e^{a_1x_1(n) + a_2x_2(n)} = e^{a_1x_1(n)}e^{a_2x_2(n)} \neq$$
$$\neq a_1 e^{x_1(n)} + a_2 e^{x_2(n)} =$$
$$= a_1 T[x_1(n)] + a_2 T[x_2(n)] \quad \text{NO LINEAL}$$

# Sistemas en tiempo discreto

- **Lineales:**  $\mathbf{L}[a_1x_1(n) + a_2x_2(n)] = a_1 \mathbf{L}[x_1(n)] + a_2 \mathbf{L}[x_2(n)]$ 
  - Salida  $y(n)$  de un sistema lineal para una entrada arbitraria  $x(n)$ :

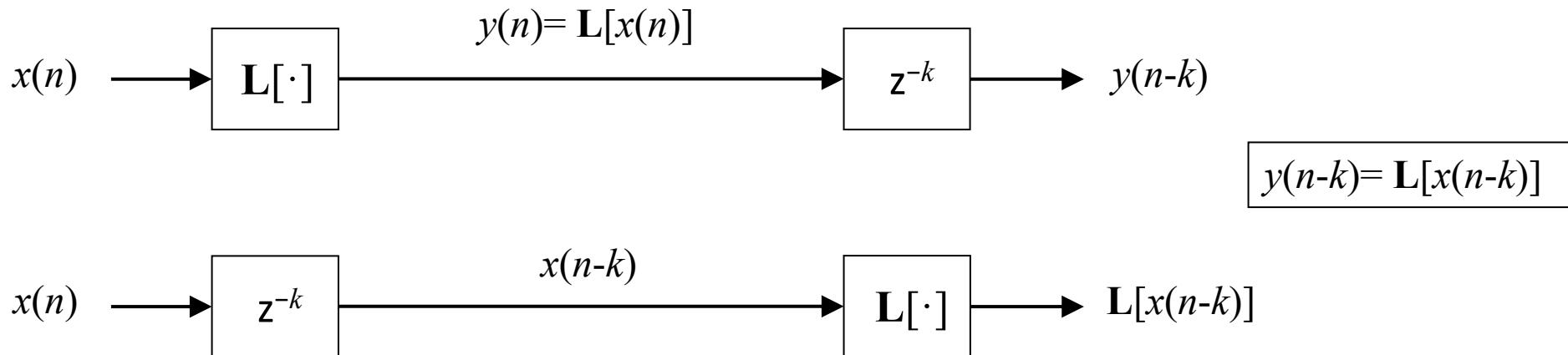
$$y(n) = \mathbf{L}[x(n)] = \mathbf{L}\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] = \sum_{k=-\infty}^{\infty} x(k)\mathbf{L}[\delta(n-k)]$$
$$h(n, k) = \mathbf{L}[\delta(n-k)]$$

- **$h(n, k)$ : respuesta al impulso centrado en  $n=k$**  en el instante  $n$ .

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n, k)$$

# Sistemas lineales invariantes en el tiempo

- Su salida es invariante a un desplazamiento en el tiempo.
- $L[\cdot]$  y el operador de desplazamiento temporal son **intercambiables**.



- Convolución lineal:

$$y(n) = \text{LTI}[x(n)] = \sum_{k=-\infty}^{\infty} x(k) \text{LTI}[\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k) h(n-k)$$
$$h(n-k) = \text{LTI}[\delta(n-k)]$$

- Si  $x(n) = \delta(n)$ :

$$y(n) = \sum_{k=-\infty}^{\infty} \delta(k) h(n-k) = h(n)$$

# Sistemas lineales invariantes en el tiempo

- **Ejemplos:** Compruebe si los sistemas siguientes son lineales e invariantes en el tiempo (LTI)

$$y(n) = L[x(n)] = \frac{1}{4}x(n) + \frac{1}{2}x(n-1) + \frac{1}{4}x(n-2)$$

$$\begin{aligned} y(n-k) &= \frac{1}{4}x(n-k) + \frac{1}{2}x(n-k-1) + \frac{1}{4}x(n-k-2) \\ L[x(n-k)] &= \frac{1}{4}x(n-k) + \frac{1}{2}x(n-k-1) + \frac{1}{4}x(n-k-2) \end{aligned} \quad \left. \begin{array}{c} \\ \end{array} \right\} \quad \rightarrow \text{LTI}$$

$$y(n) = L[x(n)] = 10 \sin[0.1\pi n]x(n)$$

$$\begin{aligned} y(n-k) &= 10 \sin[0.1\pi(n-k)]x(n-k) \\ L[x(n-k)] &= 10 \sin[0.1\pi n]x(n-k) \end{aligned} \quad \left. \begin{array}{c} \\ \end{array} \right\} \quad \rightarrow \text{No LTI}$$

# Sistemas lineales invariantes en el tiempo

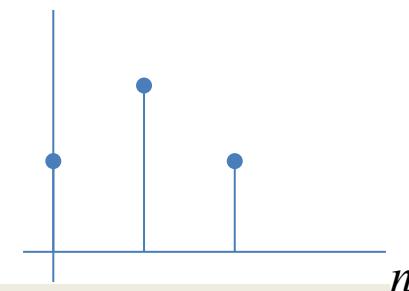
- ¿Cuál sería la respuesta impulsiva del siguiente sistema LTI?

$$y(n) = \frac{1}{4}x(n) + \frac{1}{2}x(n-1) + \frac{1}{4}x(n-2)$$

- Si comparamos con la expresión general de la salida de un sistema LTI:

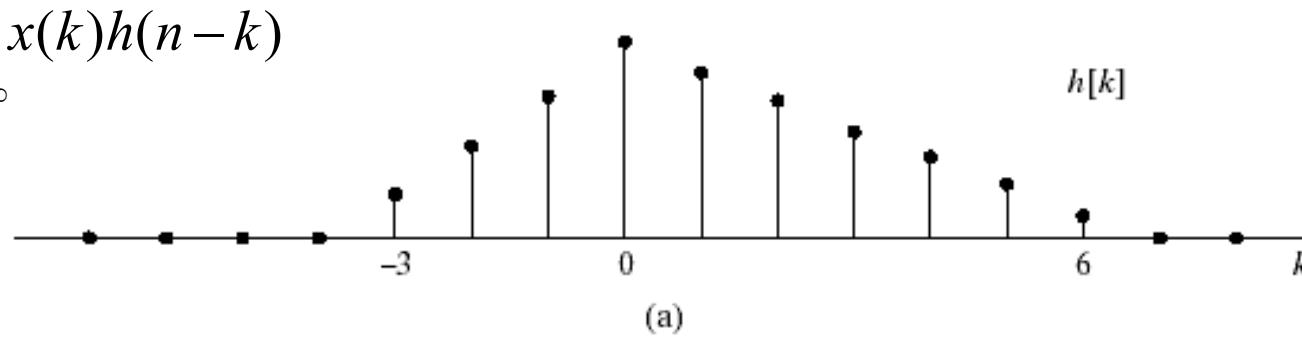
$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

Identificamos  $h(0)=\frac{1}{4}$ ,  $h(1)=\frac{1}{2}$  y  $h(2)=\frac{1}{4}$ ,  $h(n)$  para el resto de valores de  $n$ ,  $h(n)$  es nula.

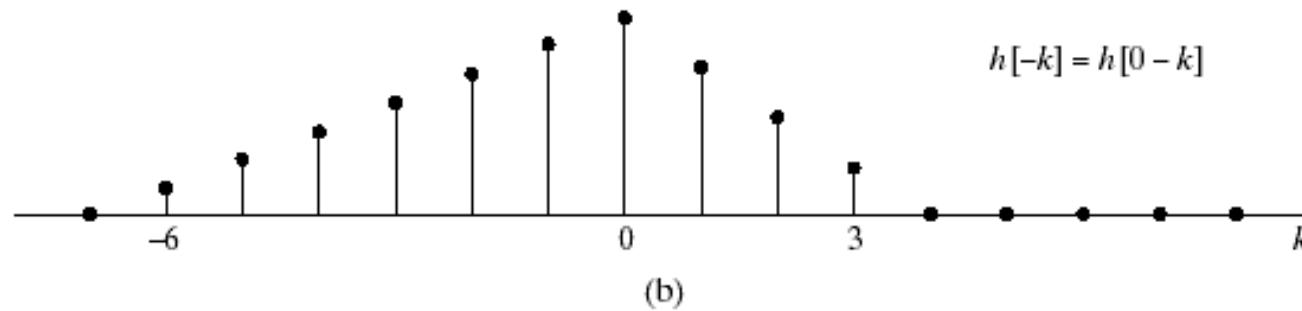


# Construcción de la secuencia $h(n-k)$

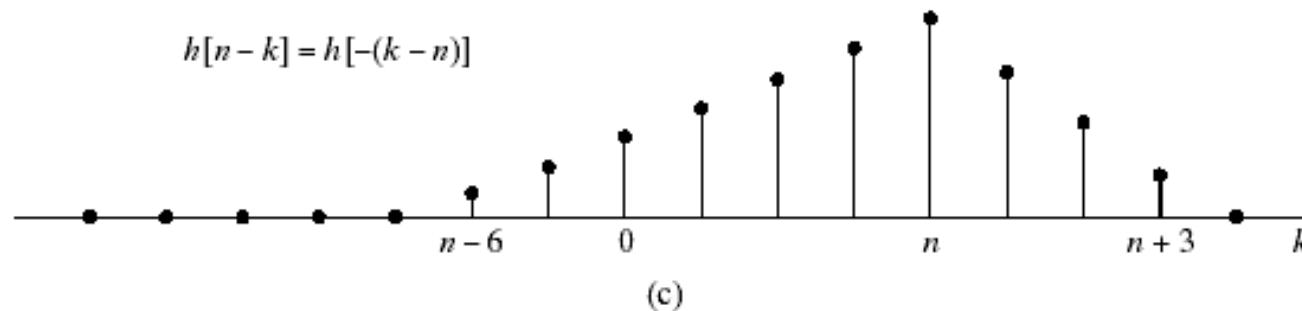
$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$



(a)

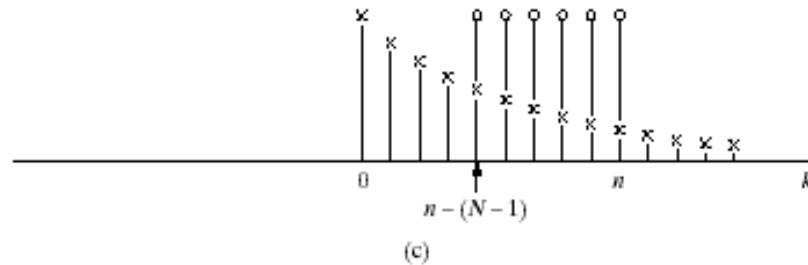
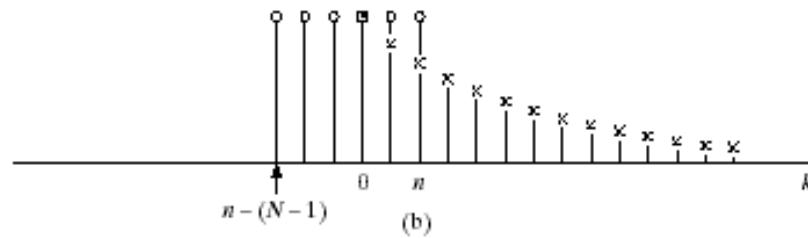
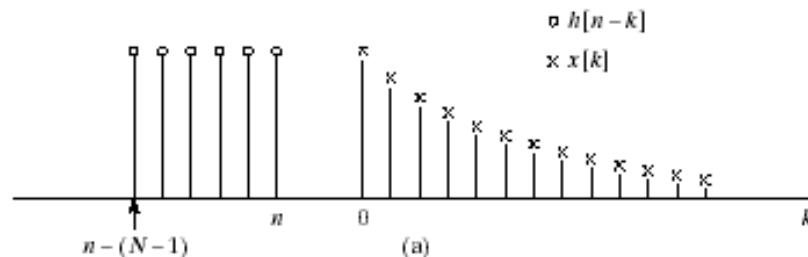


(b)

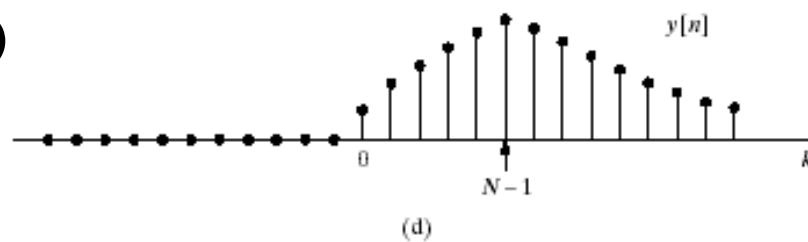


(c)

# Secuencias que intervienen en el cálculo de la convolución



$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$



# Ejemplo:

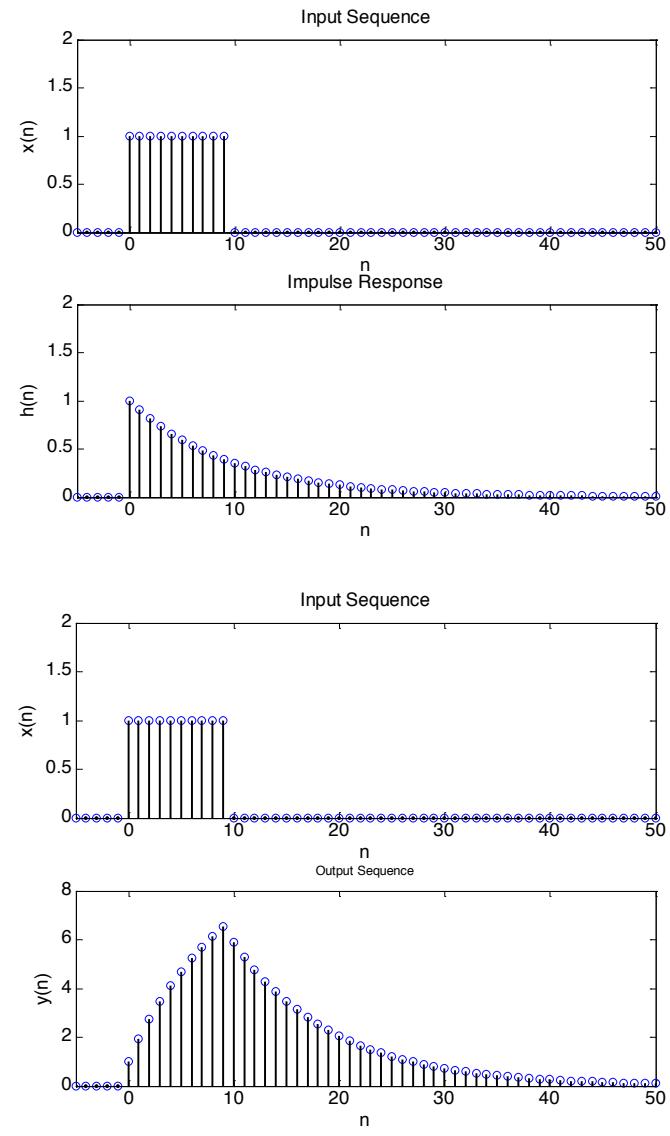
- Entrada:  $x(n) = u(n) - u(n-10)$
- LTI:  $h(n) = 0.9^n u(n)$
- Calcular la salida  $y(n)$ .

$$\begin{aligned}
 y(n) &= \sum_{k=-\infty}^{+\infty} x(k)h(n-k) \\
 &= \sum_{k=0}^9 [u(k) - u(k-10)] \cdot 0.9^{n-k} u(n-k) \\
 &= \sum_{k=0}^9 1 \cdot 0.9^{n-k} u(n-k) = 0.9^n \sum_{k=0}^9 0.9^{-k} u(n-k)
 \end{aligned}$$

$$n < 0 \quad u(n-k) = 0 \quad y(n) = 0.$$

$$0 \leq n < 9 \quad u(n-k) = 1 \quad k = 0, 1, \dots, n \quad y(n) = 10[1 - 0.9^{n+1}]$$

$$n \geq 9 \quad u(n-k) = 1, \quad 0 \leq k < 9 \quad y(n) = 10 \cdot 0.9^{n-9}[1 - 0.9^{10}]$$



# Ejemplo: Matlab

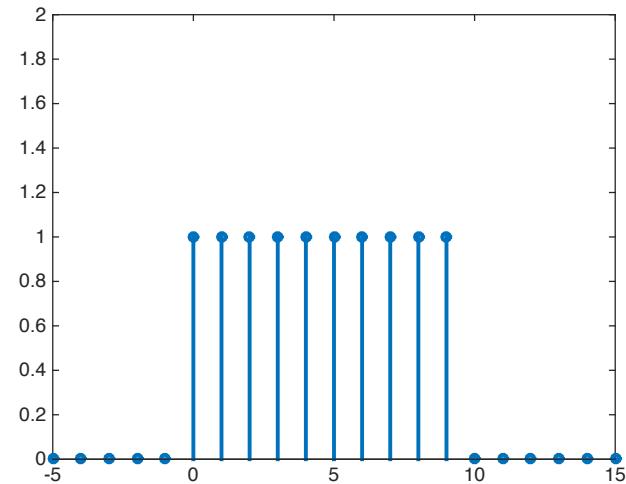
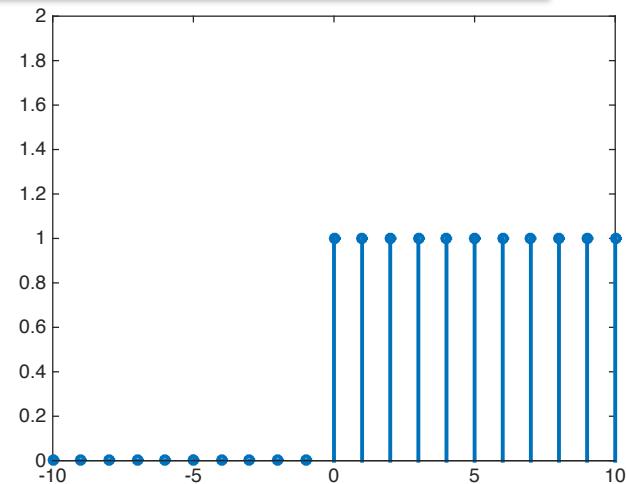
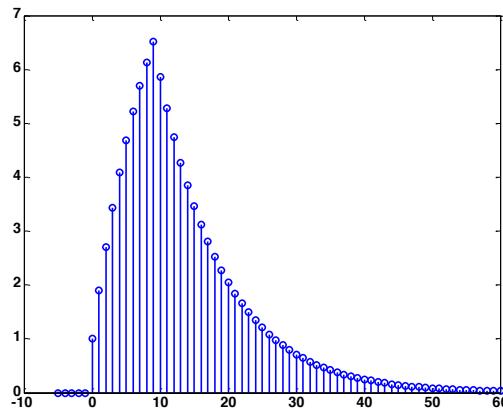
```
function [x,n] = stepseq(n0,n1,n2)
% Genera x(n) = u(n-n0); n1 <= n <= n2
%
% -----
% [x,n] = stepseq(n0,n1,n2)

n = [n1:n2]; x = [ (n-n0) >= 0 ];

>> [x,n]= stepseq(0,-10,10); stem(n,x);

>> n=-5:60;
>> x= stepseq(0,-5,60)-stepseq(10,-5,60);
>> stem(n,x)
>> axis(-5,15,0,2)

>> n=0:100
>> h= 0.9.^n
>> stem(n,h)
>> y= filter(h,1,x);
>> stem(n,y);
```



# Estabilidad y causalidad

## ■ Estabilidad.

- Def.: Cualquier entrada acotada produce una salida acotada.

$$|x(n)| < \infty \quad \Rightarrow \quad |y(n)| < \infty$$

- Un sistema LTI es estable si y sólo si la respuesta al impulso es absolutamente sumable.

$$\text{LTI es estable} \quad \Leftrightarrow \quad \sum_{n=-\infty}^{+\infty} |h(n)| < \infty$$

## ■ Causalidad.

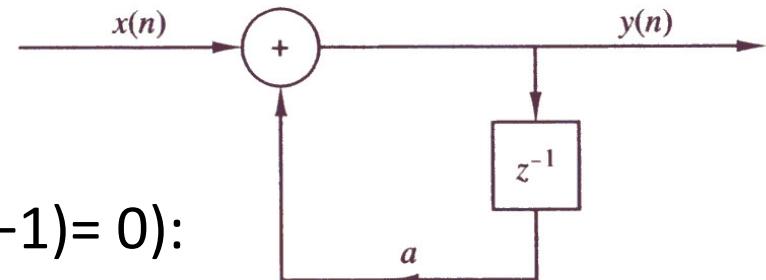
- La salida en  $n = n_0$  sólo depende de instantes  $n \leq n_0$ .
- Es decir, la salida no depende de valores futuros de la entrada.
- Criterio de causalidad:

$$\text{LTI es causal} \quad \Leftrightarrow \quad h(n) = 0, \quad n < 0$$

# Ejemplo: Estabilidad de un sistema (1/2)

- Determinar si el siguiente sistema es estable.

$$y(n) = a \cdot y(n-1) + x(n)$$



- Respuesta impulsiva (asumiendo  $y(-1)=0$ ):

$$x(n) = \delta(n) \quad \Rightarrow \quad y(n) = h(n)$$

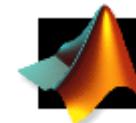
$$y(0) = h(0) = ay(-1) + \delta(0) = a \times 0 + 1 = 1$$

$$y(1) = h(1) = ay(0) + \delta(1) = a \times 1 + 0 = a$$

$$y(2) = h(2) = ay(1) + \delta(2) = a \times a + 0 = a^2$$

....

$$y(n) = h(n) = a^n u(n)$$

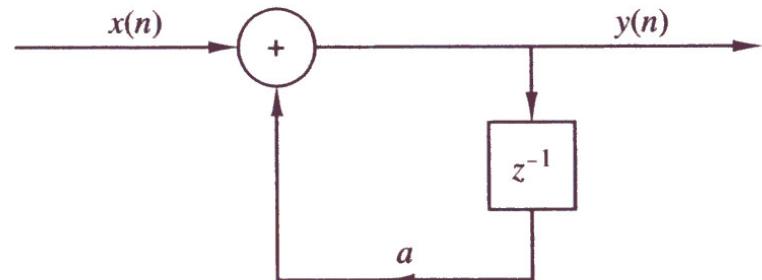


estabilidad

# Ejemplo: Estabilidad de un sistema (2/2)

- Determinar si el siguiente sistema es estable.

$$y(n) = a \cdot y(n-1) + x(n)$$



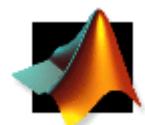
- Respuesta impulsiva:

$$h(n) = a^n u(n)$$

- Comprobación estabilidad:

$$\text{LTI es estable} \Leftrightarrow \sum_{n=-\infty}^{+\infty} |h(n)| < \infty$$

$$\sum_{n=-\infty}^{+\infty} |h(n)| = \sum_{n=-\infty}^{+\infty} |a^n u(n)| = \sum_{n=0}^{+\infty} |a^n| = \sum_{n=0}^{+\infty} |a|^n = \frac{1}{1-|a|} < \infty \quad \text{Si} \quad |a| < 1$$



estabilidad

# Propiedades de la convolución

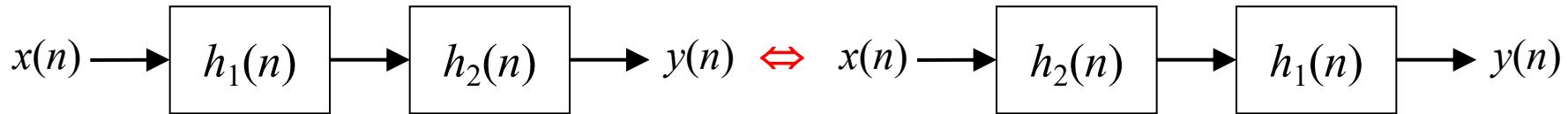
- Def.: Convolución:

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

- Propiedades:

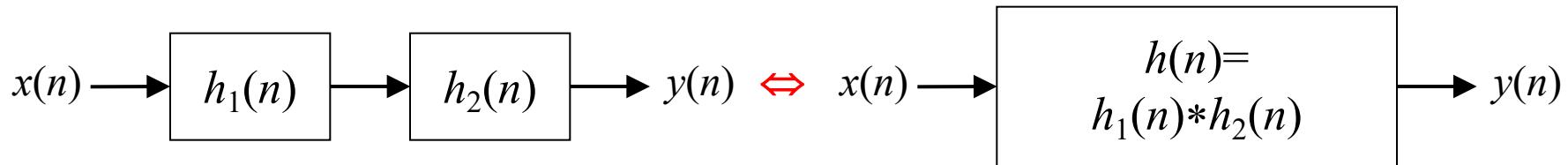
- Ley commutativa:

$$x(n) * h(n) = h(n) * x(n)$$



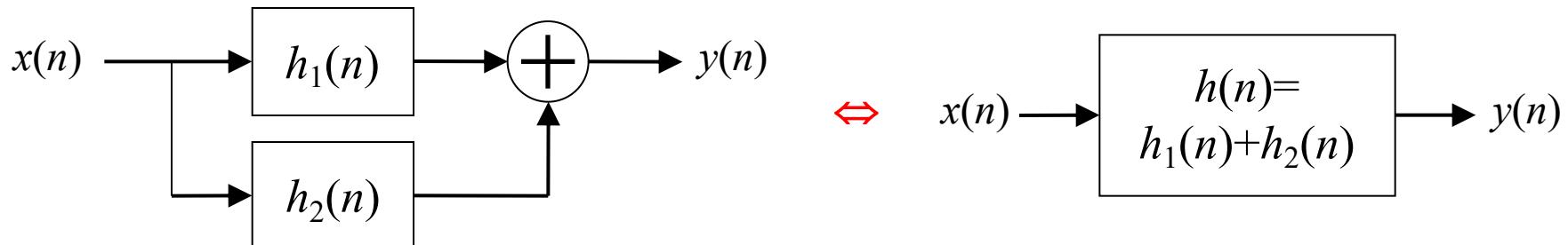
- Ley asociativa:

$$[x(n) * h_1(n)] * h_2(n) = x(n) * [h_1(n) * h_2(n)]$$



- Ley distributiva:

$$x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n)$$



# Clasificación de los sistemas LTI

## ■ FIR, Finite Impulse Response:

- Sistemas causal con respuesta al impulso  $h(n)$  finita.

$$h(n) = 0 \quad n < 0 \quad \text{y} \quad n \geq M$$

$$y(n) = x(n) * h(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

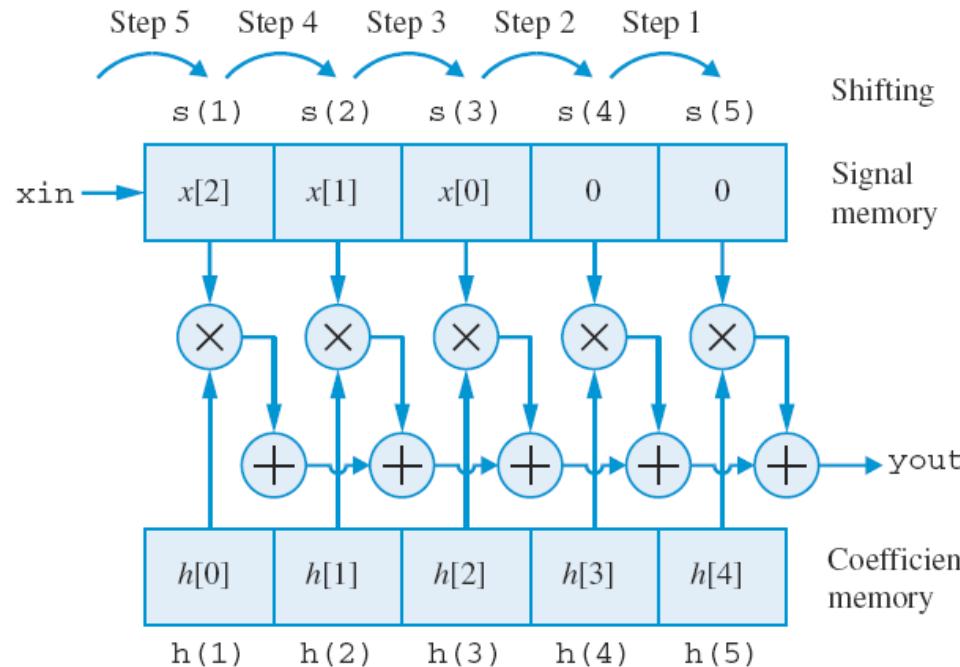
- Pondera las  $M$  muestras más recientes de la señal.
- Tiene una **memoria finita** de  $M$  muestras.

## ■ IIR, Infinite Impulse Response:

- Sistemas con respuesta al impulso  $h(n)$  infinita.
- Tiene **memoria infinita**.
- No es posible la implementación directa a través de la convolución.
- Si se tratara de un IIR causal.

$$y(n) = x(n) * h(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

# Implementación de un filtro FIR



$$y(n) = \sum_{k=0}^4 h(k)x(n-k)$$

```
% Script file: firstream.m
% FIR filter implementation using stream processing
% Generate an input signal sequence
N=20; ni=(0:N-1); x=(3/4).^ni+0.1*rand(size(ni));
% Store impulse response
M=5; h=ones(1,M)/M; % M-point Moving Average filter
% Initialize signal memory
s=zeros(1,M);
% Compute filter output sequence
for n=1:N      % Sampling-time index
    xin=x(n); % Get input sample from ADC or storage
    s(1)=xin;
    yout=h(1)*s(1);
    for m=M:-1:2
        yout=yout+h(m)*s(m); % Multiply, Accumulate
        s(m)=s(m-1);          % and Shift Operation
    end
    y(n)=yout; % Put output sample to DAC or storage
end
```

# Ecuaciones en diferencias (1/2)

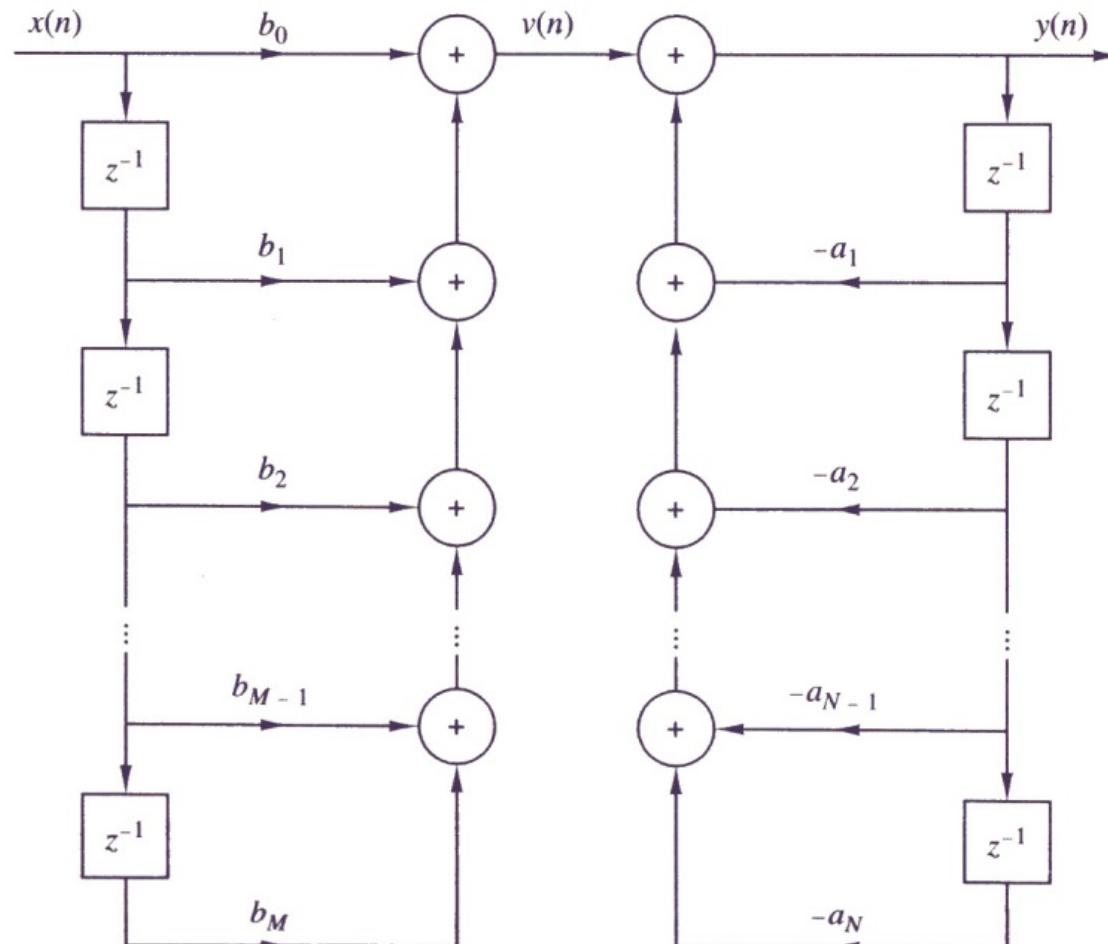
- **Implementación** de filtros FIR e IIR.
- Un sistema LTI puede ser igualmente descrito mediante una **ecuación en diferencias** con coeficientes constantes.
- Describe la **relación entrada/salida** del sistema LTI:

$$y(n) = F [y(n-1), y(n-2), \dots, y(n-N), x(n), x(n-1), \dots, x(n-M)]$$

- IIR: Implementación recursiva.
- FIR: Implementación no recursiva.

# Ecuaciones en diferencias (2/2)

$$y(n) = -\sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k)$$

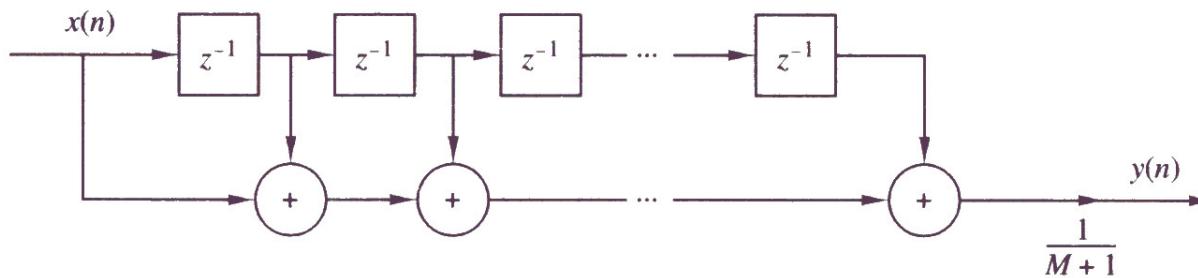


# Ejemplo 1: FIR recursivo y no recursivo

- **Media móvil:**

$$y(n) = \frac{1}{M+1} \sum_{k=0}^M x(n-k) \quad h(n) = \frac{1}{M+1} \quad 0 \leq n \leq M$$

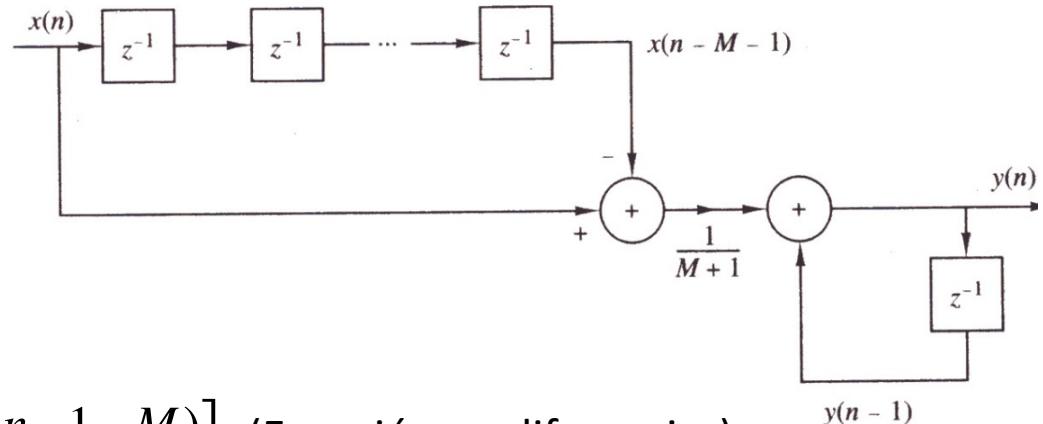
- **Realización no recursiva:**



- **Realización recursiva:**

$$y(n) = \frac{1}{M+1} \sum_{k=0}^M x(n-k)$$

$$= y(n-1) + \frac{1}{M+1} [x(n) - x(n-1-M)] \quad (\text{Ecuación en diferencias})$$



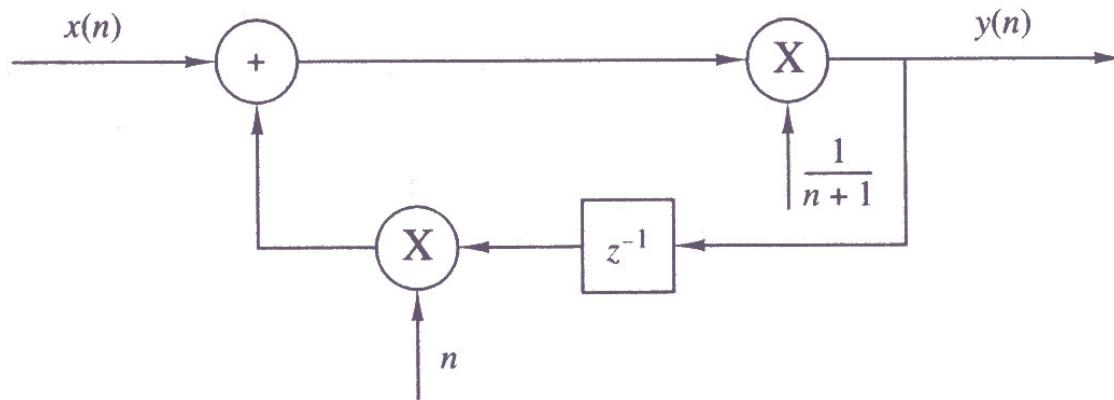
## Ejemplo 2: Media acumulativa

- **Media acumulativa** de  $x(n)$  en el intervalo  $[0, n]$ :

$$y(n) = \frac{1}{n+1} \sum_{k=0}^n x(k) \quad n = 0, 1, \dots$$

- Se puede calcular  $y(n)$  a partir de  $y(n-1)$ .
  - Se reducen las necesidades de memoria.

$$y(n) = \frac{n}{n+1} y(n-1) + \frac{1}{n+1} x(n) \quad (\text{Ecuación en diferencias})$$



# La transformada de Fourier en tiempo discreto (DTFT)

- Hasta ahora hemos visto cómo calcular la **respuesta temporal** de un sistema a una secuencia arbitraria:

$$x(n) \rightarrow \boxed{h(n)} \rightarrow y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

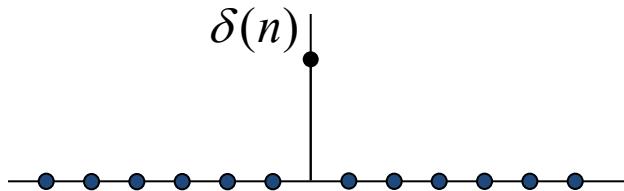
- La **respuesta en frecuencia** de un sistema LTI es la representación de mayor utilidad práctica.
- La **transformada de Fourier** en tiempo discreto de una secuencia  $x(n)$  se define como:

Si  $\sum_{n=-\infty}^{+\infty} |x(n)| < \infty \Rightarrow$

$$X(e^{j\omega}) \equiv \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$$
$$x(n) \equiv \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega$$

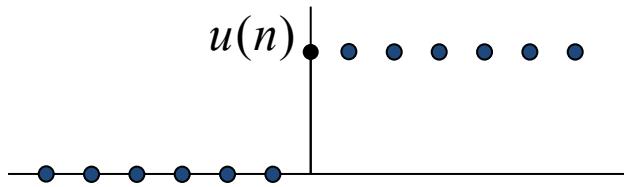
# Ejemplos DTFT (1/2)

- Secuencia impulso  $\delta(n)$ :



$$X(e^{j\omega}) \equiv \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} \delta(n)e^{-j\omega n} = 1$$

- Escalón unitario  $u(n)$ :



No existe porque la secuencia  $u(n)$  no es absolutamente sumable.

$$\sum_{n=-\infty}^{+\infty} |u(n)| \quad \text{no está acotada}$$

- Exponencial compleja:

$$x(n) = e^{j\omega_0 n} \quad \leftrightarrow \quad X(e^{j\omega}) = \delta(\omega - \omega_0)$$

-D-

$$x(n) \equiv \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \delta(\omega - \omega_0) e^{j\omega n} d\omega = e^{j\omega_0 n}$$

# Ejemplos DTFT (2/2)

## ■ Secuencias sinusoidales:

$$x(n) = e^{j\omega_0 n} \quad \Leftrightarrow \quad X(e^{j\omega}) = \delta(\omega - \omega_0)$$

- Aprovechando las relaciones:

$$\cos(\omega_0 n) = \frac{1}{2} [e^{j\omega_0 n} + e^{-j\omega_0 n}]$$

$$\sin(\omega_0 n) = \frac{1}{2j} [e^{j\omega_0 n} - e^{-j\omega_0 n}]$$

- Se obtiene:

$$x(n) = \cos(\omega_0 n) \quad \Leftrightarrow \quad X(e^{j\omega}) = \frac{1}{2} [\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$$

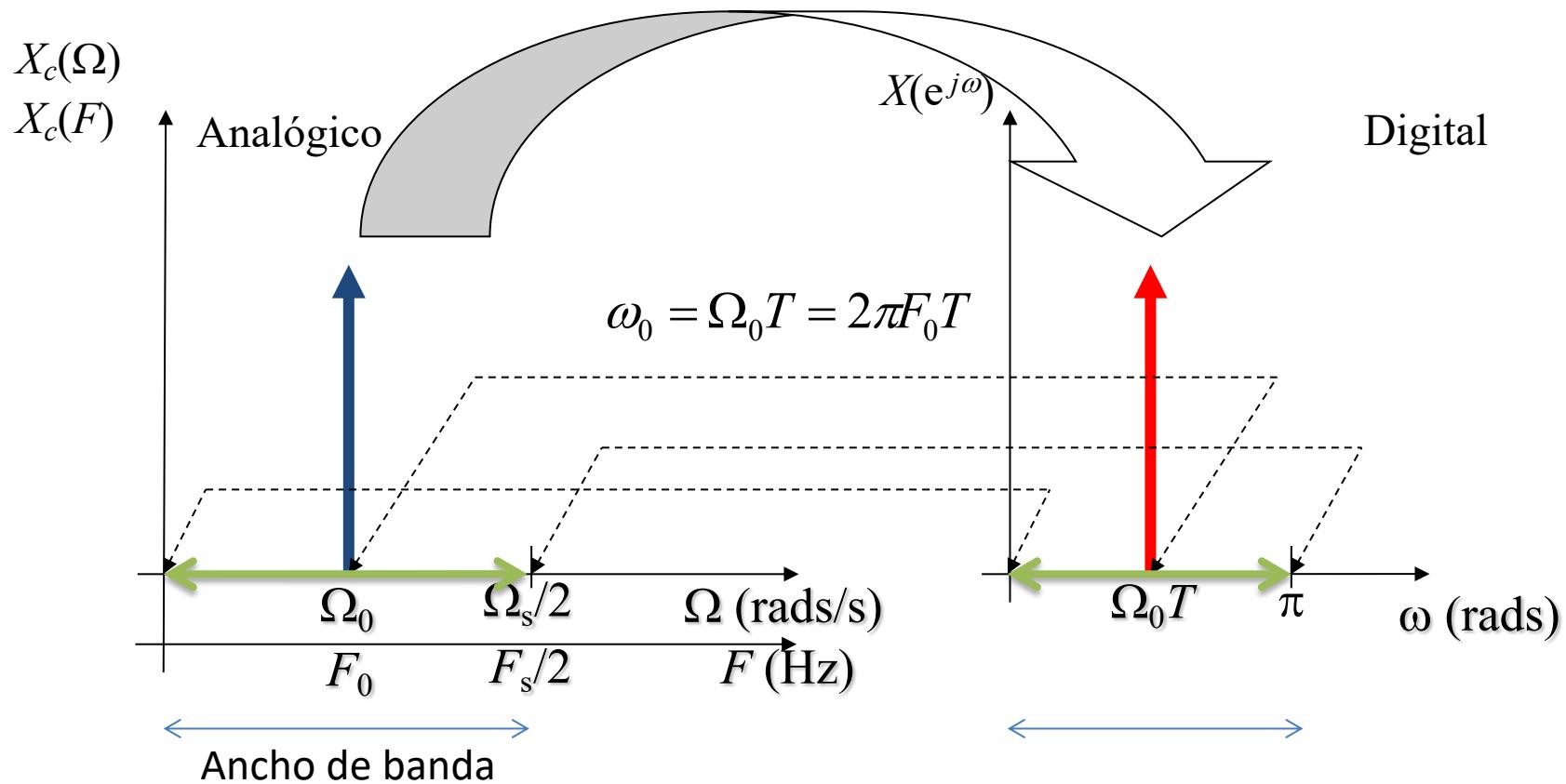
$$x(n) = \sin(\omega_0 n) \quad \Leftrightarrow \quad X(e^{j\omega}) = \frac{1}{2j} [\delta(\omega - \omega_0) - \delta(\omega + \omega_0)]$$

# Correspondencia entre frecuencias (A/D)

- Secuencias seno/coseno:

- Analógica:  $x_c(t) = A \cos(\Omega_0 t) = A \cos(2\pi F_0 t)$

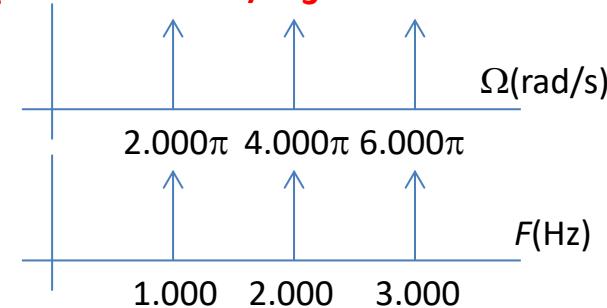
- Digital:  $x(n) = x_c(t=nT) = A \cos(\Omega_0 n T) = A \cos(\omega_0 n)$



# Ejemplo: 3 tonos

- Señal analógica de 3 tonos:  $F_1 = 1.000 \text{ Hz}$ ,  $F_2 = 2.000 \text{ Hz}$  y  $F_3 = 3.000 \text{ Hz}$

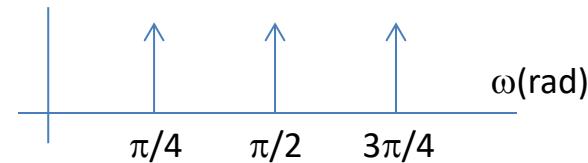
$$\begin{aligned}x(t) &= A_1 \cos(\Omega_1 t) + A_2 \cos(\Omega_2 t) + A_3 \cos(\Omega_3 t) = \\&= A_1 \cos(2\pi F_1 t) + A_2 \cos(2\pi F_2 t) + A_3 \cos(2\pi F_3 t) = \\&= A_1 \cos(2.000\pi t) + A_2 \cos(4.000\pi t) + A_3 \cos(6.000\pi t)\end{aligned}$$



- Se muestrea a una frecuencia  $F_s = 8.000 \text{ Hz}$

$$\begin{aligned}x(n) &= x(t = nT) = x(t = n / F_s) = \\&= A_1 \cos\left(\frac{2.000\pi n}{8.000}\right) + A_2 \cos\left(\frac{4.000\pi n}{8.000}\right) + A_3 \cos\left(\frac{6.000\pi n}{8.000}\right) = \\&= A_1 \cos\left(\frac{\pi}{4}n\right) + A_2 \cos\left(\frac{\pi}{2}n\right) + A_3 \cos\left(\frac{3\pi}{4}n\right)\end{aligned}$$

$$\omega = \Omega T = 2\pi \frac{F}{F_s} =$$



- Transformada de Fourier de la señal discreta:

$$X(e^{j\omega}) = \frac{1}{2} A_1 [\delta(\omega - \pi/4) + \delta(\omega + \pi/4)] + \frac{1}{2} A_2 [\delta(\omega - \pi/2) + \delta(\omega + \pi/2)] + \frac{1}{2} A_3 [\delta(\omega - 3\pi/4) + \delta(\omega + 3\pi/4)]$$

# Ejemplo: $x(n) = \alpha^n u(n)$ , $|\alpha| < 1$

- Transformada de Fourier en tiempo discreto (DTFT):

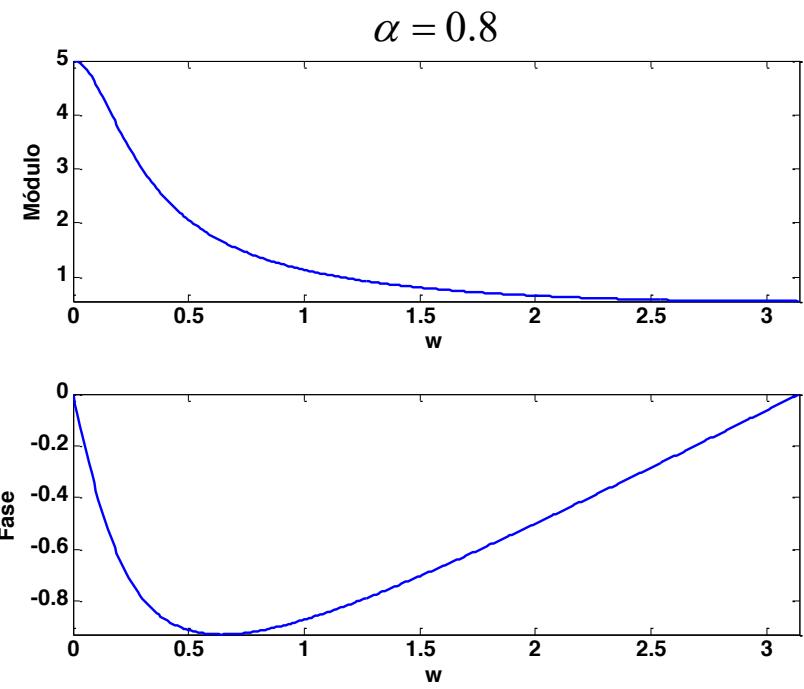
$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} \alpha^n u(n)e^{-j\omega n} = \\ &= \sum_{n=0}^{+\infty} \alpha^n e^{-j\omega n} = \sum_{n=0}^{+\infty} (\alpha e^{-j\omega})^n = \frac{1}{1 - \alpha e^{-j\omega}} \quad \text{Si } |\alpha e^{-j\omega}| < 1 \Leftrightarrow |\alpha| < 1 \end{aligned}$$

- Módulo:

$$|X(e^{j\omega})| = \frac{1}{\sqrt{1 + \alpha^2 - 2\alpha \cos(\omega)}}$$

- Fase:

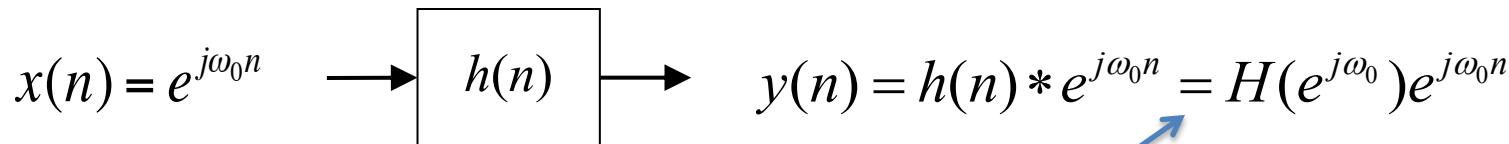
$$\angle X(e^{j\omega}) = \text{atan} \left( -\frac{\alpha \sin(\omega)}{1 - \alpha \cos(\omega)} \right)$$



# Respuesta en frecuencia de un sistema LTI

- Respuesta de un LTI a

$$x(n) = \exp(j\omega_0 n)$$



-D-

$$\begin{aligned} y(n) &= h(n) * e^{j\omega_0 n} = \sum_{k=-\infty}^{+\infty} h(k)e^{j\omega_0(n-k)} \\ &= e^{j\omega_0 n} \sum_{k=-\infty}^{+\infty} h(k)e^{-j\omega_0 k} = \underline{H(e^{j\omega_0})e^{j\omega_0 n}} \end{aligned}$$

- La entrada se ve modificada por la **respuesta en frecuencia** del sistema LTI:

$$H(e^{j\omega}) \equiv \sum_{n=-\infty}^{+\infty} h(n)e^{-j\omega n}$$

## Ejemplo: Respuesta a señales exponenciales y de tipo coseno

- Respuesta a una combinación lineal de exponenciales complejas de distinta frecuencia:

$$\sum_k A_k e^{j\omega_k n} \rightarrow \boxed{h(n)} \rightarrow y(n) = h(n) * \sum_k A_k e^{j\omega_k n} = \sum_k A_k h(n) * e^{j\omega_k n}$$
$$= \sum_k A_k H(e^{j\omega_k}) e^{j\omega_k n}$$

- Respuesta a secuencias sinusoidales:

$$A \cos(\omega_0 n + \theta_0) \rightarrow \boxed{h(n)} \rightarrow y(n) = A |H(e^{j\omega_0})| \cdot \cos[\omega_0 n + \theta_0 + \angle H(e^{j\omega_0})]$$

$$\sum_k A_k \cos(\omega_k n + \theta_k) \rightarrow \boxed{h(n)} \rightarrow y(n) = \sum_k A_k |H(e^{j\omega_k})| \cdot \cos[\omega_k n + \theta_k + \angle H(e^{j\omega_k})]$$

# Ejemplo 1: Respuesta a señales exponenciales y de tipo coseno

```
% Ilustración del efecto de la fase en filtrado digital.
```

```
wc= 0.5; % Frecuencia de corte  
n= 10; % Orden del filtro
```

```
% Diseño del filtro IIR
```

```
[b, a] = butter(n,wc);
```

```
% Analizamos la respuesta en frecuencia y el retardo de grupo.
```

```
figure(1); freqz(b,a);
```

```
figure(2); grpdelay(b,a);
```

```
% Generamos tres señales discretas de frecuencias f1, f2 y f3
```

```
% muestreadas con Fs y filtramos.
```

```
Fs= 200; T= 1/Fs;
```

```
f1= 5; f2= 10; f3= 29; n= 0:99;
```

```
x1= cos(2*pi*f1*n*T); y1= filter(b,a,x1);
```

```
x2= cos(2*pi*f2*n*T); y2= filter(b,a,x2);
```

```
x3= cos(2*pi*f3*n*T); y3= filter(b,a,x3);
```

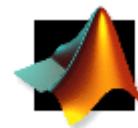
```
% Observamos el retardo en cada frecuencia.
```

```
figure(3);
```

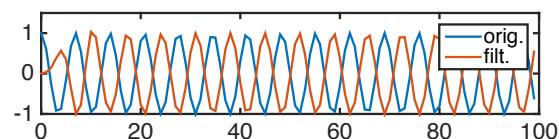
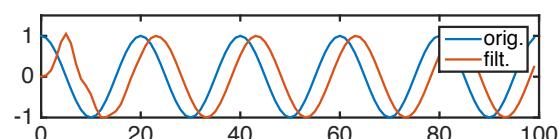
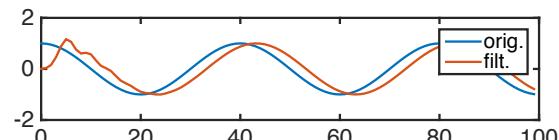
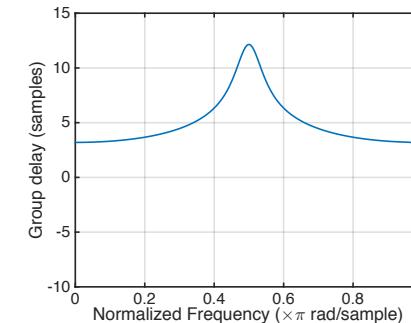
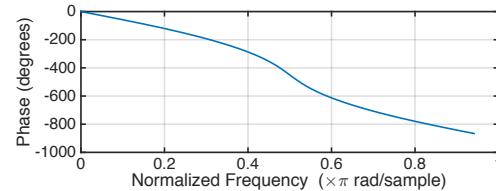
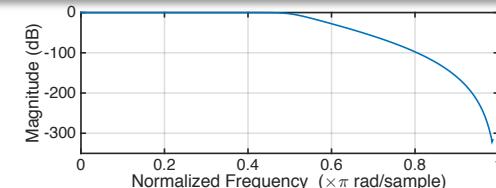
```
subplot(3,1,1); plot(n,x1,n,y1); legend('orig.', 'filt.');
```

```
subplot(3,1,2); plot(n,x2,n,y2); legend('orig.', 'filt.');
```

```
subplot(3,1,3); plot(n,x3,n,y3); legend('orig.', 'filt.');
```



fase\_filtro



# Respuesta en frecuencia de un sistema LTI

- Dado el sistema LTI, ¿qué relación hay entre la **entrada** y la **salida** en el **dominio de la frecuencia**?

Dominio del tiempo



Dominio de la frecuencia

$$x(n) \rightarrow \boxed{h(n)} \rightarrow y(n) = x(n) * h(n) = \sum_{k=-\infty}^{+\infty} x(k)h(n-k)$$

$$X(e^{j\omega}) \quad H(e^{j\omega}) \quad Y(e^{j\omega}) = ?$$

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$

$m = n - k$



-D-

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{+\infty} y(n)e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} x(n) * h(n)e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} x(k)h(n-k)e^{-j\omega n} = \\ &= \sum_{m=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} x(k)h(m)e^{-j\omega(k+m)} = \sum_{m=-\infty}^{+\infty} h(m)e^{-j\omega m} \sum_{k=-\infty}^{+\infty} x(k)e^{-j\omega k} = H(e^{j\omega})X(e^{j\omega}) \end{aligned}$$

# Ejemplo numérico: Respuesta en frecuencia

- Filtro de promediado de 5 muestras:

$$y(n) = \frac{1}{5} \sum_{k=0}^4 x(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

- Respuesta impulsiva:

$$h(0) = h(1) = \dots = h(4) = 1/5$$

- Respuesta en frecuencia:

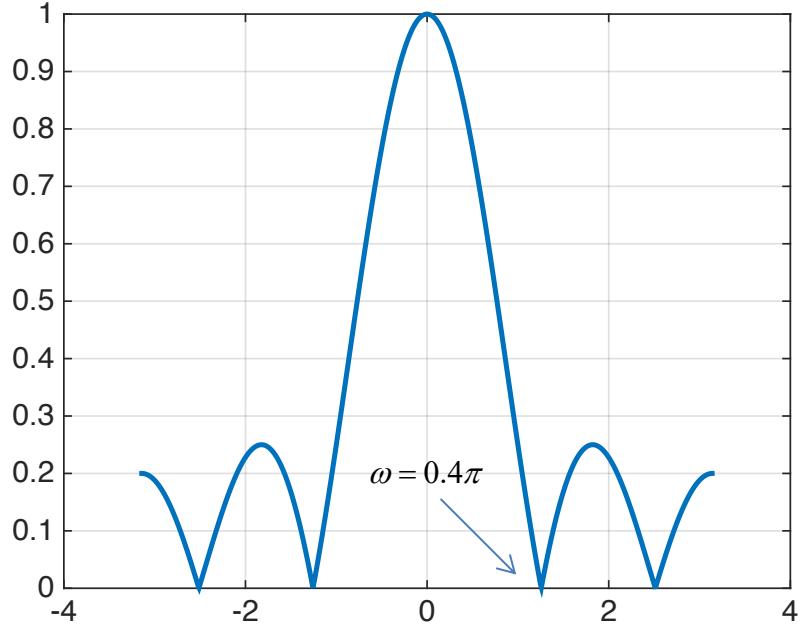
$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=-\infty}^{+\infty} h(n)e^{-j\omega n} = \frac{1}{5} \sum_{n=0}^4 e^{-j\omega n} \\ &= \frac{1}{5} [1 + e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega} + e^{-j4\omega}] \end{aligned}$$

% Matlab

```
w= -pi:0.01:pi;
H= 1/5*(1+exp(-j*w)+exp(-j*2*w)+ ...
          exp(-j*3*w)+exp(-j*4*w));
plot(w,abs(H));
```

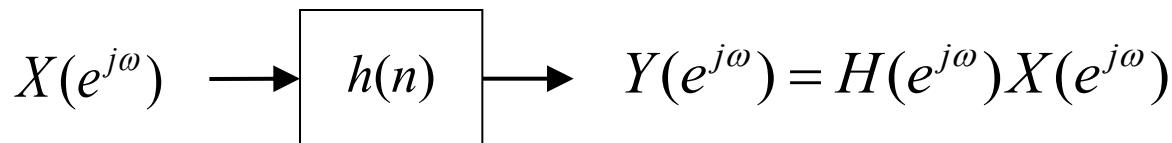
% Mediante toolboxes (modulo en dB y fase).

```
h= 1/5*ones(5,1);
freqz(h,1,512);
```



## Ejemplo 2: Respuesta en frecuencia de un sistema LTI

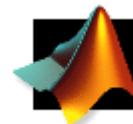
- Respuesta a una secuencia arbitraria:



$$h(n) = 0.9^n u(n)$$

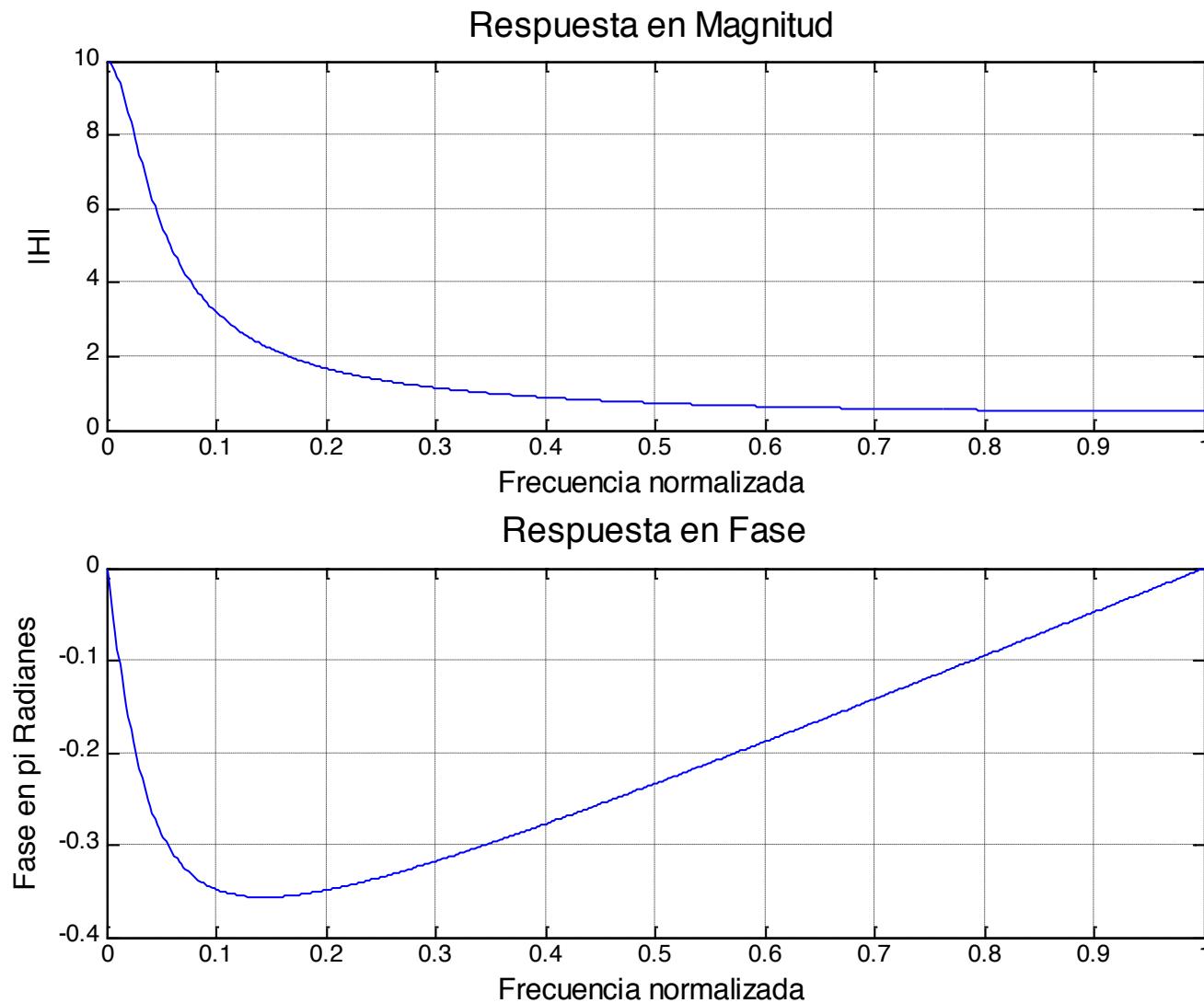
- Ejemplo:

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=-\infty}^{+\infty} h(n)e^{-j\omega n} = \sum_{n=0}^{+\infty} 0.9^n e^{-j\omega n} \Rightarrow |H(e^{j\omega})| = \frac{1}{\sqrt{1.81 - 1.8 \cos(\omega)}} \\ &= \sum_{n=0}^{+\infty} [0.9e^{-j\omega}]^n = \frac{1}{1 - 0.9e^{-j\omega}} \quad \angle H(e^{j\omega}) = -\arctan \left[ \frac{0.9 \sin(\omega)}{1 - 0.9 \cos(\omega)} \right] \end{aligned}$$



Res\_frec

## Ejemplo 2: Respuesta en frecuencia de un sistema LTI



# Respuesta en frecuencia (ecuaciones en diferencias)

- Respuesta en frecuencia de un sistema LTI descrito mediante la ecuación en diferencias:

$$y(n) + \sum_{l=1}^N a_l y(n-l) = \sum_{m=0}^M b_m x(n-m)$$

- Para una entrada  $x(n) = e^{j\omega n}$  la salida  $y(n) = H(e^{j\omega})e^{j\omega n}$

$$H(e^{j\omega})e^{j\omega n} + \sum_{l=1}^N a_l H(e^{j\omega})e^{j\omega(n-l)} = \sum_{m=0}^M b_m e^{j\omega(n-m)}$$

- Eliminando  $e^{j\omega n}$  y despejando:

$$H(e^{j\omega}) = \frac{\sum_{m=0}^M b_m e^{-j\omega m}}{1 + \sum_{l=1}^N a_l e^{-j\omega l}}$$

# Ejemplo 1: Filtro paso baja de 3<sup>er</sup> orden

- Sistema LTI descrito por la ecuación en diferencias:

- $y(n) = 0.0181 x(n) + 0.0543 x(n-1) + 0.0543 x(n-2) +$   
 $+ 0.0181 x(n-3) + 1.76 y(n-1) - 1.1829 y(n-2) + 0.2781 y(n-3)$

$$b_0 = 0.0181$$

$$b_3 = 0.0543$$

$$b_1 = 0.0543,$$

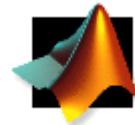
$$b_2 = 0.0181$$

$$a_1 = 1.76$$

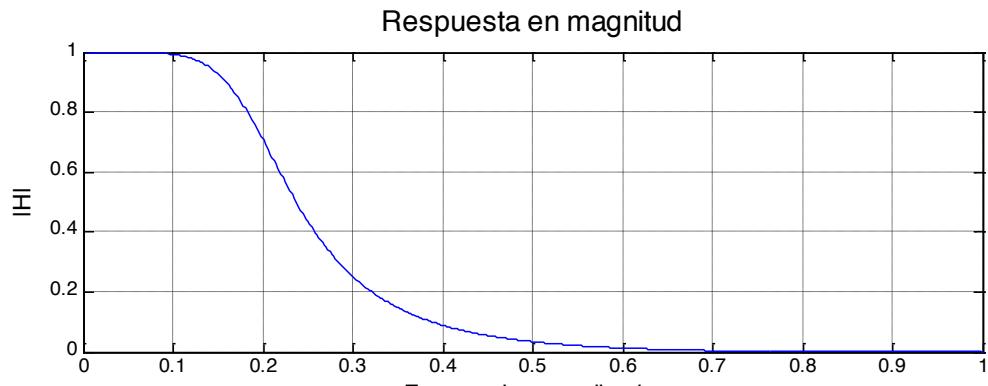
$$a_3 = 0.2781$$

$$a_2 = 1.1829,$$

$$H(e^{j\omega}) = \frac{\sum_{m=0}^M b_m e^{-j\omega m}}{1 + \sum_{l=1}^N a_l e^{-j\omega l}}$$



filtro\_3ord



## Ejemplo 2. Respuesta estacionaria (1/2)

- Sistema LTI definido mediante la ecuación en diferencias:

$$y(n) = 0.8 \cdot y(n-1) + x(n)$$

- Determinar  $H(e^{j\omega})$ .

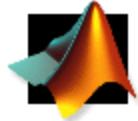
$$H(e^{j\omega}) = \frac{1}{1 - 0.8e^{-j\omega}}$$

- Calcular la respuesta estacionaria a la entrada:

$$x(n) = \cos(0.05\pi n)u(n)$$

- Cara calcular la respuesta a  $x(n)$  necesitamos  $H(e^{j0.05\pi})$ :
- $H(e^{j0.05\pi}) = 4.0928 \cdot e^{-j0.5377} \Rightarrow y_{ss} = 4.0928 \cdot \cos[0.05\pi(n-3.42)]$
- La entrada se escala en un factor 4.0928 y se retrasa 3.42 muestras.

## Ejemplo 2. Respuesta estacionaria (2/2)



rf\_ec\_dif

