

# Laboratorio 2

## Procesamiento Digital de Señales

### Muestreo, Cuantificación, Señales Análogas y Digitales

Paula Pérez, Alejandro Escobar y Cristian Ríos

2024-1

#### NOTAS:

- Enviar el informe del laboratorio con el siguiente nombre: *Lab2\_PDS\_Apellido\_Nombre.ipynb*
- Enviar junto con el informe los archivos adicionales generados y descargados. Todo esto debe ir en un archivo comprimido con el siguiente nombre: *Lab2\_PDS\_Apellido\_Nombre.zip*
- OJO! Recuerde tener cuidado con la indentación y caracteres como el guion bajo y las llaves cuando copie y pegue el código entregado en esta guía.
- Las preguntas deberán ser resueltas en el notebook, indicando sus respectivos numerales.

## 1. Introducción

En este laboratorio se pretende afianzar los conceptos relacionados con muestreo y cuantificación de señales. Para ello se usarán señales de audio que se van a re-muestrear y re-cuantificar.

En el procesamiento digital de señales solo se pueden realizar operaciones aritméticas con números en un rango limitado. Para esto existe el muestreo que consiste en tomar muestras de una señal analógica a una frecuencia o tasa de muestreo constante, siempre considerando el teorema de Nyquist.

Por otro lado, la cuantificación es el proceso de mapear valores de amplitud continua en un conjunto de valores contables. Puede ser aplicado a señales análogas o a señales en tiempo discreto (Ver Figura 1).

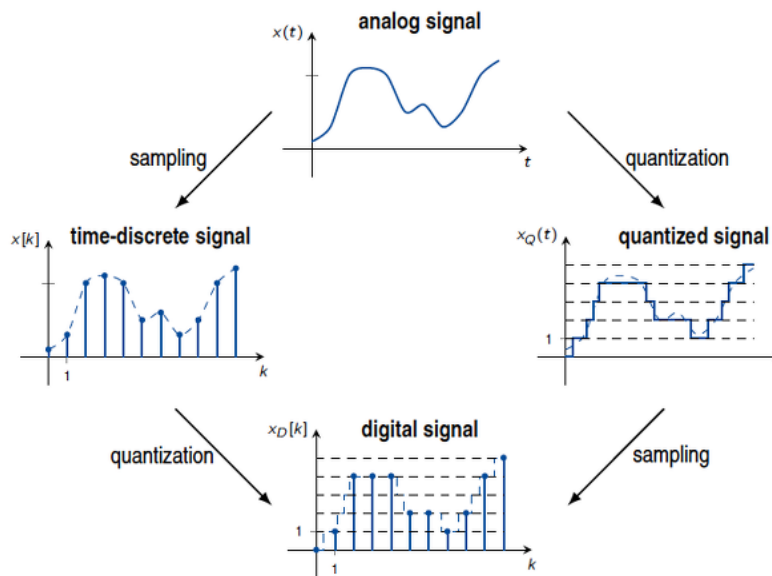


Figura 1: Proceso de muestreo y cuantificación

## 2. Muestreo

1. Cargue la señal correspondiente al ultimo número de su cédula. Con base en el número total de muestras y a la frecuencia de muestreo, grafique la señal en función del tiempo.
2. Genere un nuevo archivo de audio con la misma señal pero al doble y luego con la mitad de la frecuencia de muestreo original. Escuche el audio, ¿Qué diferencia nota? ¿A que se debe esto?. Tip: Para generar el archivo utilice las siguientes líneas:

```
from scipy.io.wavfile import write
write ('nuevarutadearchivo.wav', nueva_fs, senal)
```

3. Sub-muestree la señal usando un cuarto de la frecuencia de muestreo original. Reproduzca la señal resultante y la original, ambas usando una frecuencia de  $fs/3$ . Grafique ambas señales en la misma figura. ¿Que puede concluir?

Tip: use las siguientes instrucciones para el sub-muestreo:

```
from scipy.signal import resample
import numpy as np
senal2=resample(senal, int(len(senal)/3))
```

## 3. Frecuencia de Nyquist y Aliasing

1. Genere una señal sinusoidal con una frecuencia de 2 Hz y una frecuencia de muestreo de 15 Hz. Genere otra señal sinusoidal de frecuencia 2 Hz y muéstréela según el teorema de Nyquist. Grafique señales en la misma figura. ¿Qué diferencias encuentras?
2. Escriba un código que genere dos señales sinusoidales muestreadas con una frecuencia  $fs = (n + 1) * 20$  Hz. La señal 1 tendrá una frecuencia  $f_1 = (n + 1)$  Hz y la señal 2 tendrá una frecuencia  $f_2 = 2 * fs + (n + 3)$  Hz. Tome  $n$  como ultimo dígito de su cédula. ¿Ambas señales quedan bien representadas. ¿A que se debe esto?.

## 4. Cuantificación

1. Use la función mostrada a continuación para re-cuantificar la señal de audio utilizada en los numerales anteriores a 2, 4 y 12 bits. Escuche y grafique las señales re-cuantificadas. Compare gráficamente cada una de las señales obtenidas con la señal original. ¿Que pudo notar?.

```
def fxquant(s,bit):
    # s: senal de entrada, debe estar normalizada entre -1 y 1
    # bit: bits de cuantizacion
    Plus1=np.power(2, (bit-1))
    X=s*Plus1
    X=np.round(X)
    X=np.minimum(Plus1-1.0,X)
    X=np.maximum(-1.0*Plus1,X)
    X=X/Plus1
    return X
```

## 5. Transmisión Analógica vs. Digital

Se considerará el caso de la transmisión de una señal sobre un medio confinado de gran longitud, en el cual varios repetidores serán usados para compensar la atenuación introducida por el canal.

Cada segmento del medio introduce una atenuación  $1/G$ , por lo cual se puede recobrar la amplitud original adicionando un repetidor de ganancia  $G$ . Sin embargo, si la señal ha acumulado ruido aditivo, el ruido también será amplificado por el repetidor. Después de  $N$  repetidores, el ruido será amplificado  $N$  veces, y la señal en el receptor será:

$$\hat{x}_N(t) = x(t) + NG\sigma(t) \quad (1)$$

Por otro lado si se usa una señal digital, se puede establecer un umbral después de cada repetidor para re-digitalizar la señal y eliminar el ruido en cada etapa.

## 5.1. Generación de señales analógica y digital

1. Cargue la señal de audio correspondiente al ultimo número de su cédula.
2. Para obtener la señal analógica, normalice el audio en amplitud y luego re-escale en un factor de 460. Use el siguiente código como ayuda.

```
norm = 1.0 / np.max(np.abs(x))
sA = 460.0 * x * norm
```

3. La señal digital se obtiene aproximando cada punto de la señal analógica a un valor discreto que generalmente es un entero. Use la función `round()` de la librería `numpy` para aproximar la señal analógica y obtener la señal digital.
4. Para ver las diferencias entre la señal analógica y digital grafique, en una misma figura (`subplot`), 100 muestras intermedias de cada señal. ¿Se puede observar alguna diferencia entre las señales?

## 5.2. Transmisión

Ahora se va a definir un código que simule una transmisión por un medio confinado con  $N$  repetidores espaciados una longitud  $L$ . Para simular la transmisión deben seguirse los siguientes pasos:

- Atenuar la señal en un factor  $a < 1$  por cada segmento de longitud  $L$  (Antes de amplificar con el repetidor).
  - Sumar ruido con una determinada amplitud para simular la distorsión del canal.
  - Compensar la perdida de amplitud en cada uno de los repetidores.
1. Use la función llamada `repeater(...)` que recibe como entrada la señal a transmitir, la amplitud del ruido que se desea agregar y la constante de atenuación. Esta función retorna la señal afectada por el ruido, atenuada, y amplificada nuevamente.

```
def repeater(x, noise_amplitude, attenuation):
    x=x*attenuation
    Noise=np.random.uniform(-noise_amplitude, noise_amplitude, len(x))
    x=x+Noise
    x=x/attenuation
    return x
```

2. La transmisión analógica no es más que una secuencia de repetidores, la cual puede ser implementada con la siguiente función.

```
def analog_tx(x, num_repeater, noise_amplitude, attenuation):
    for n in range(0, num_repeater):
        x = repeater(x, noise_amplitude, attenuation)
    return x
```

3. En el caso de la señal digital, comúnmente se vuelve a cuantificar la señal después de cada repetidor, ya que sabemos que en este caso solo tenemos valores enteros, esto es implementado a partir de la siguiente función.

```
def digital_tx(x, num_repeater, noise_amplitude, attenuation):
    for n in range(0, num_repeater):
        x = np.round(repeater(x, noise_amplitude, attenuation))
    return x
```

4. Compare los esquemas de transmisión de las señales. Para ello defina el número de repetidores en 150, la amplitud del ruido 0.15, y la constante de atenuación en 0.35. Luego realice la transmisión de la señal análoga y digital en cada uno de sus respectivos esquemas de transmisión, y calcule la relación señal a ruido (SNR) tanto para la señal análoga como digital. ¿Qué diferencia encuentra entre las señales?

Use la siguiente función para calcular la SNR:

```
def SNR(noisy, original):
    # power of the error
    err = np.var(original-noisy)
    # power of the signal
    sig = np.var(original)
    # SNR in dBs
    return 10 * np.log10(sig/err)
```

5. Manteniendo constantes la amplitud de ruido y la atenuación, realice un barrido del número de repetidores entre 10 y 100 (no lo haga de 1 en 1). Para cada iteración realice la transmisión análoga y digital y calcule la SNR para cada una. Al final grafique la SNR en función del número de repetidores. ¿Qué diferencia encuentra entre las gráficas obtenidas?

## 6. Enlaces de interés

[https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/linear\\_uniform\\_characteristic.ipynb](https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/linear_uniform_characteristic.ipynb)

[https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/nonlinear\\_quantization\\_speech\\_signal.ipynb](https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/nonlinear_quantization_speech_signal.ipynb)

[https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/requantization\\_speech\\_signal.ipynb](https://github.com/spatialaudio/digital-signal-processing-lecture/blob/master/quantization/requantization_speech_signal.ipynb)