UNIVERSIDAD DE VALPARAÍSO
Escuela de Ingeniería Civil Biomédica
CBM414 Procesamiento digital de señales biomédicas

Tarea de laboratorio 03, Grupo 1

Profesor: David Ortiz Puerta, david.ortiz@uv.cl

FECHA DE ENTREGA: 1 DE OCTUBRE, A LAS 23:59HRS

Cada estudiante debe desarrollar y entregar individualmente los análisis teóricos. Aunque es posible discutir los conceptos generales de los problemas en grupo, las soluciones no deben compartirse. **El informe debe ser digitado en un editor de texto (Word, LaTeX, Documentos de Google) y enviados en formato PDF. No se aceptarán informes escritos a mano.** Además, debe incluir todos los resultados, figuras y explicaciones solicitadas para la tarea. La evaluación del informe considerará la correcta diagramación, redacción y presentación, pudiendo aplicarse una reducción de hasta 2.0 puntos si no se cumple con estos criterios. **Antes de la fecha límite, el informe en formato PDF debe ser subido al *classroom*, en la pestaña *Trabajo de clase/Entregas de laboratorios*, y en la sección según corresponda a su grupo.** El incumplimiento de estas instrucciones resultará en una calificación de 1.0 en la tarea. No se aceptarán tareas después de la fecha límite.
Se otorgará una bonificación de 0.3 puntos si el informe está redactado en inglés, y una bonificación adicional de 0.3 puntos si se utiliza LaTeX para su redacción. Para verificar este último criterio, se debe adjuntar el archivo `.tex` *del informe en el mensaje de entrega.*

RECURSOS

- El profesor y la ayudante
- Módulos en python suficientes para el desarrollo de la tarea

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read # para leer archivos de audio
from IPython.display import Audio # para escuchar archivos de audio
import scipy.signal as signal
```

```
from scipy.fft import fft, ifft, fftfreq
```

PROBLEMA 1: CUANTIZACIÓN DE UNA SEÑAL DE AUDIO (2.5 PUNTOS)

Considere las siguientes funciones

```
def cuantizar(s,bit):
  # s: senal de entrada
  # bit: bits de cuantificación
  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

def rms(x, xq):
  return np.sqrt(np.mean((x-xq)**2))
```

Responda o realice lo siguiente:

1. (0.5 puntos) Explique detalladamente ¿Qué realiza la función `cuantizar`?
2. (0.5 puntos) Re-cuantice la señal de audio a 2, 4 y 8 bits. Escuche las señales y describa qué percibe
3. (0.5 puntos) Grafique la señal original y la cuantizada, para 2, 4 y 8 bits (son 3 gráficos) ¿Qué observa en los gráficos?
4. (0.5 puntos) Calcule el error (rms) de cuantización para 2, 4 y 8 bits. ¿Qué puede concluir de los resultados?
5. (0.5 puntos) Grafique la diferencia (punto a punto) entre la señal original y la señal re-cuantizada para 2, 4 y 8 bits (realice un solo gráfico). ¿Qué puede concluir del gráfico anterior?

PROBLEMA 2: RUIDO EN UNA SEÑAL DE AUDIO (2.5 PUNTOS)

Recordemos que el SNR_{dB} se puede calcular como

$$SNR_{dB} = 10\log_{10}\left(\frac{E[S^2]}{E[N^2]}\right)$$

donde $E[S^2]$ y $E[N^2]$ es la varianza de la señal y del ruido, respectivamente.Ahora, considere las siguientes funciones `add_gaussian_noise` y `calculate_snr`

```
def add_gaussian_noise(signal, snr_db):
  # Calcular la potencia de la señal
  signal_power = np.mean(signal**2)

  # Calcular la potencia del ruido deseada
  snr_linear = 10**(snr_db / 10)
  noise_power = signal_power / snr_linear

  # Calcular la varianza del ruido
  noise_variance = noise_power

  # Generar ruido gaussiano
  noise = np.sqrt(noise_variance) * np.random.randn(*signal.shape)

  # Agregar el ruido a la señal
  noisy_signal = signal + noise

  return noisy_signal, noise

def calculate_snr(signal, noise):
  """ Calcula el SNR entre una señal de entrada "signal" y el ruido "noise"
  """
  # Ensure numpy arrays
  signal, noise = np.array(signal), np.array(noise)

  # Calculate the power of the signal and the noise
  signal_power = np.mean(signal**2)
  noise_power = np.mean(noise**2)

  # Calculate the SNR in decibels (dB)
  snr = 10 * np.log10(signal_power / noise_power)
  return snr
```

1. (0.7 puntos) Explique brevemente ¿qué realiza la función `add_gaussian_noise`?. Utilice la definición de SNR_{dB} para explicar, paso a paso matemáticamente, como se extrajo la variable `noise_variance` ubicada dentro de la función.
2. (0.6 puntos) Contamine la señal de audio a con un ruido Gaussiano utilizando la función `noise = np.random.normal(0, var, len(signal))`, donde Var = 0.1, 0.01, 0.001. Es-

cuche las señales y describa brevemente qué percibe.

3. (0.6 puntos) Calcule el SNR_{dB} utilizando la función `calculate_snr` y reporte los valores para las 3 varianzas dadas. ¿El SNR_{dB} aumenta, disminuye o se mantiene igual? ¿Qué puede concluir de los resultados, haciendo énfasis en el significado de decibel?
4. (0.6 puntos) Utilize la función `add_gaussian_noise` y agregue ruido Gaussiano con un SNR_{dB} = 1, 10, 30. Utilice la segunda salida de la función `add_gaussian_noise`, i.e., `noise` y calcule la varianza. ¿Qué puede concluir de los resultados?

PROBLEMA 3: RUIDO EN UNA SEÑAL DE ECG (2 PUNTOS)

Considere la siguiente señal para generar una señal de ECG sintética.

```
def create_ECG_signal():
  # Simulated Beats per minute rate
  # For a health, athletic, person, 60 is resting, 180 is intensive exercising
  sampling_rate = 200
  samples_rest = 10
  bpm = 60
  bps = bpm / 60

  pqrst = signal.wavelets.daub(10)
  zero_array = np.zeros(samples_rest, dtype=float)
  pqrst_full = np.concatenate([pqrst,zero_array])

  # Simumated period of time in seconds that the ecg is captured in
  capture_length = 10

  # Concatonate together the number of heart beats needed
  ecg_template = np.tile(pqrst_full , int(capture_length * bps))

  ecg_sampled = signal.resample(ecg_template, int(sampling_rate * capture_length))
  ecg_sampled = ecg_sampled / ecg_sampled.max()
  ecg_sampled = 1.5 * ecg_sampled + 1

  return ecg_sampled

# ECG contaminada
time = np.arange(0, capture_length, 1/sampling_rate)
ecg_sampled = create_ECG_signal()

noise = np.random.normal(0, 0.1, len(ecg_sampled))
```

```
noise50hz = 0.5*np.sin(2*np.pi*50*time)

ecg_noisy_sampled = ecg_sampled + noise50hz + noise
```

Responda:

1. (0.6 puntos) Grafique la señal sin ruido y con ruido. Además, calcule el SNR_{dB} de las señal respecto al ruido que la contamina.
2. (0.6 puntos) Utilize la 'FFT' y obtenga el espectro de la señal sin ruido y con ruido. Analise ambos espectros y describa ¿Qué puede concluir de las gráficas?
3. (0.8 puntos) Normalice la señal de -1 a 1. Grafique la señal original y la cuantizada, para 2, 4 y 8 bits (son 3 gráficos) ¿Qué observa en los gráficos?