Tarea No.4: Procesamiento Digital de Señales

Marianne Nicté, Rodríguez Canek, 202000656[1, *]

[1] *Facultad de Ingeniería, Departamento de Electronica, Universidad de San Carlos, Ciudad Universitaria, Zona 12, Guatemala.*

El procesamiento digital de señales (PDS) es el campo que se encarga de estudiar cómo modificar y trabajar con señales discretas, ya sea en el dominio del tiempo o de la frecuencia, con el fin de analizarlas, transformarlas o procesarlas. En este proyecto se desarrolló un programa en Octave que ofrece funciones para grabar, reproducir, graficar y examinar la densidad espectral de potencia de señales de audio. El objetivo principal del programa es facilitar el procesamiento de señales mediante un menú interactivo, intuitivo y fácil de manejar, que permite elegir la duración de la grabación y, posteriormente, visualizar las gráficas correspondientes al espectro y la potencia del audio.

INTRODUCCIÓN

El procesamiento digital de señales tiene un uso extendido en áreas como las comunicaciones, el reconocimiento de voz, el procesamiento de imágenes y el análisis de audio. Este proyecto busca implementar un sistema básico para la manipulación de señales de audio en Octave, un entorno de código abierto similar a MATLAB, empleando sus funciones integradas y herramientas de análisis espectral. Además, se desarrolló una versión equivalente en Python con el propósito de comparar las ventajas y desventajas de ambos entornos.

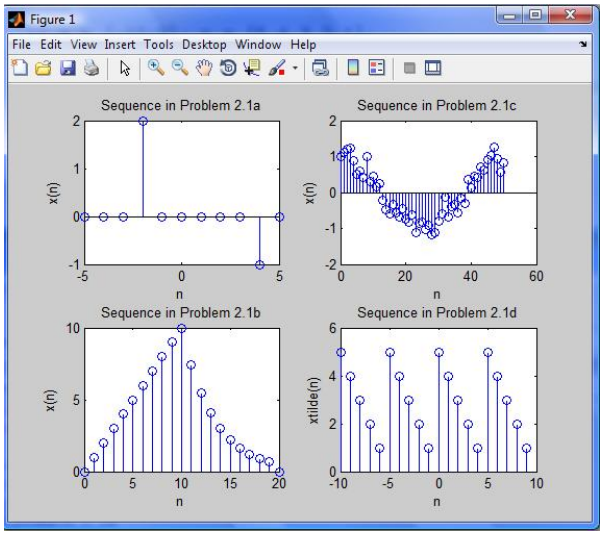

Figura 1: Ejemplo de procesamiento de señales.

I. CONFIGURACIÓN INICIAL

Para ejecutar el programa, se utilizó Octave debido a su compatibilidad con herramientas de análisis digital y su soporte para bibliotecas como signal. El programa genera y almacena automáticamente los archivos de audio procesados (audio.wav) en el directorio de trabajo actual.

* e-mail: 3243383091703@ingenieria.usac.edu.gt

II. EXPLICACIÓN DEL CÓDIGO

A. Menú principal

El menú interactivo permite al usuario seleccionar una acción mediante un sistema de switch-case. Cada caso ejecuta una función específica basada en la entrada del usuario, de la misma forma se realizo en python, con la unica diferencia que en vez de utilizar un switch-case se utiliza el equivalente en python que es un While.

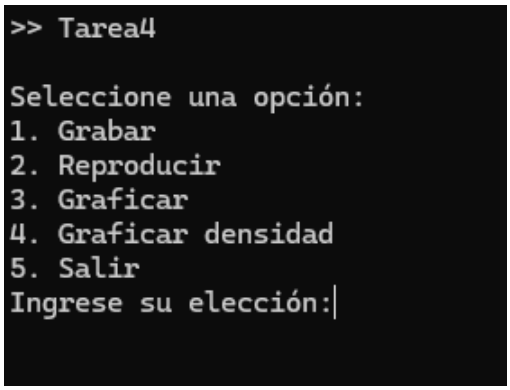

Figura 2: Menú principal en Octave.

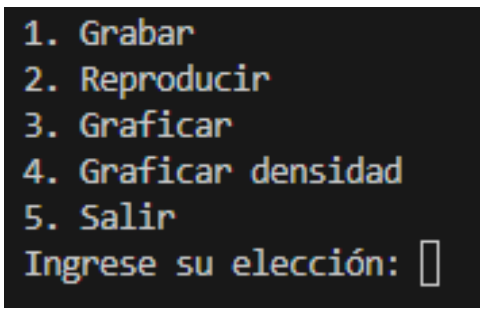

Figura 3: Menú principal en Python.

```
% Menú principal
opcion = 0;
while opcion ~= 5
    % opcion = input('Seleccione una opción\n 1. Grabar audioln 2. Reproducir audioln 3. Graficar audioln 4. Salir\n');
    % Menú de opciones
    disp('Seleccione una opción:')
    disp('1. Grabar')
    disp('2. Reproducir')
    disp('3. Graficar')
    disp('4. Graficar densidad')
    disp('5. Salir')
    opcion = input('Ingrese su elección:');
```

Figura 4: Código del menú para Octave.

```
def main():
    while True:
        print("Seleccione una opción:")
        print("1. Grabar")
        print("2. Reproducir")
        print("3. Graficar")
        print("4. Graficar densidad")
        print("5. Salir")
        opcion = input("Ingrese su elección: ")
```

Figura 5: Código del menú para Python.

B. Grabación del audio

Esta función brinda al usuario la posibilidad de grabar audio utilizando el objeto audiorecorder en Octave, mientras que en Python se recurre a la librería sounddevice para realizar la captura. La duración de la grabación se define en segundos y el resultado se almacena en un archivo con formato .wav.

```
switch opcion
    case 1
        % Grabación de audio
        try
            duracion = input('Ingrese la duración de la grabación en segundos:');
            disp('Comenzando la grabación...');
            recObj = audiorecorder;
            recordblocking(recObj, duracion);
            disp('Grabación finalizada.');
            data = getaudiodata(recObj);
            audiowrite('audio.wav', data, recObj.SampleRate);
            disp('Archivo de audio grabado correctamente.');
        catch
            disp('Error al grabar el audio.');
        end
```

Figura 6: Código para grabación de audio en Octave.

```
def grabar_audio(duracion, fs=44100):
    print("Comenzando la grabación...")
    grabacion = sd.rec(int(duracion * fs), samplerate=fs, channels=1)
    sd.wait()  # Espera hasta que la grabación termine
    print("Grabación finalizada.")
    write('audio.wav', fs, grabacion)  # Guarda la grabación en un archivo .wav
    print("Archivo de audio grabado correctamente.")
    return grabacion, fs
```

Figura 7: Código para grabación de audio en Python.

C. Reproducción de audio

Lee el archivo audio.wav previamente generado y lo reproduce utilizando la función sound, y en python usamos la misma libreria sounddevice con la opción de play para poder reproducir el audio guardado

```
case 2
    % Reproducción de audio
    try
        [data, fs] = audioread('audio.wav');
        sound(data, fs);
    catch
        disp('Error al reproducir el audio.');
    end
```

Figura 8: Código para reproducción de audio en Octave.

```
def reproducir_audio():
    try:
        fs, data = read('audio.wav')
        sd.play(data, fs)
        sd.wait()  # Espera hasta que la reproducción termine
    except Exception as e:
        print("Error al reproducir el audio:", e)
```

Figura 9: Código para reproducción de audio en Python.

D. Gráfica del audio

Se genera una representación gráfica de la señal en el dominio del tiempo, mostrando la amplitud en función del tiempo.

```
case 3
    % Gráfico de audio
    try
        [data, fs] = audioread('audio.wav');
        tiempo = linspace(0, length(data)/fs, length(data));
        plot(tiempo, data);
        xlabel('Tiempo (s)');
        ylabel('Amplitud');
        title('Audio');
    catch
        disp('Error al graficar el audio.');
    end
```

Figura 10: Código de gráfica de audio en Octave.

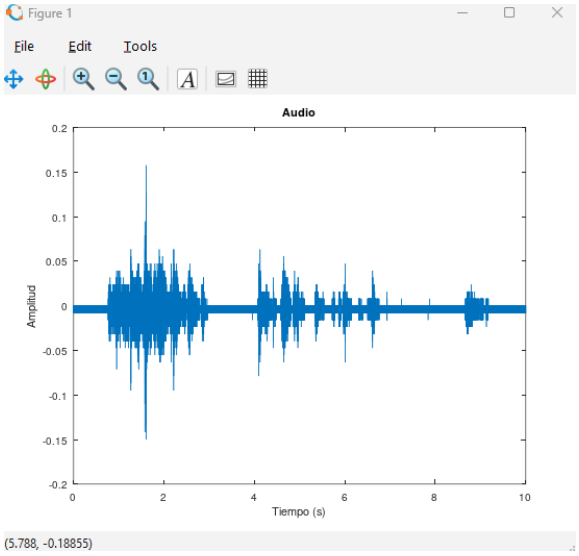

Figura 11: Gráfica de audio en Octave.

```python
def graficar_audio():
    try:
        fs, data = read('audio.wav')
        tiempo = np.linspace(0, len(data) / fs, num=len(data))
        plt.plot(tiempo, data)
        plt.xlabel('Tiempo (s)')
        plt.ylabel('Amplitud')
        plt.title('Audio')
        plt.show()
    except Exception as e:
        print("Error al graficar el audio:", e)
```

Figura 12: Código de gráfica de audio en Python.

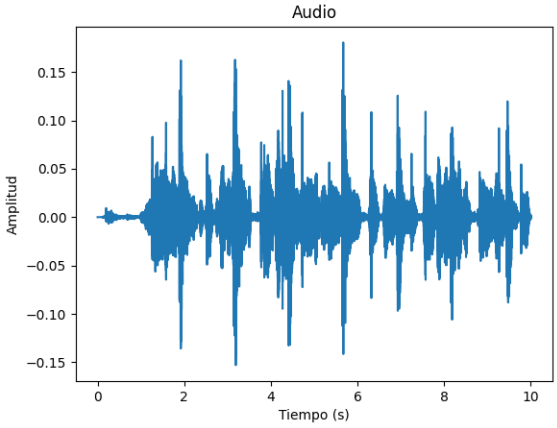

Figura 13: Gráfica de audio en Python.

E. Gráfico de Densidad Espectral de Potencia

Utilizando la función pwelch mientras que en python se utiliza la libreria scipy.signal, se calcula y grafica la densidad espectral de potencia de la señal, que muestra cómo se distribuye la energía en el dominio de la frecuencia.

```
case 4
    % Graficando espectro de frecuencia
    try
        disp('Graficando espectro de frecuencia...');
        [audio, fs] = audioread('audio.wav'); % Lee la señal desde el archivo .wav
        N = length(audio); % Número de muestras de la señal
        f = linspace(0, fs/2, N/2+1); % Vector de frecuencias
        ventana = hann(N); % Ventana de Hann para reducir el efecto de las discontinuidades al calcular la FFT
        Sxx = pwelch(audio, ventana, 0, N, fs); % Densidad espectral de potencia
        plot(f, 10*log10(Sxx(1:N/2+1))); % Gráfica el espectro de frecuencia en dB
        xlabel('Frecuencia (Hz)');
        ylabel('Densidad espectral de potencia (dB/Hz)');
        title('Espectro de frecuencia de la señal grabada');
    catch
        disp('Error al graficar el audio.');
    end
```

Figura 14: Código gráfica de densidad en Octave.

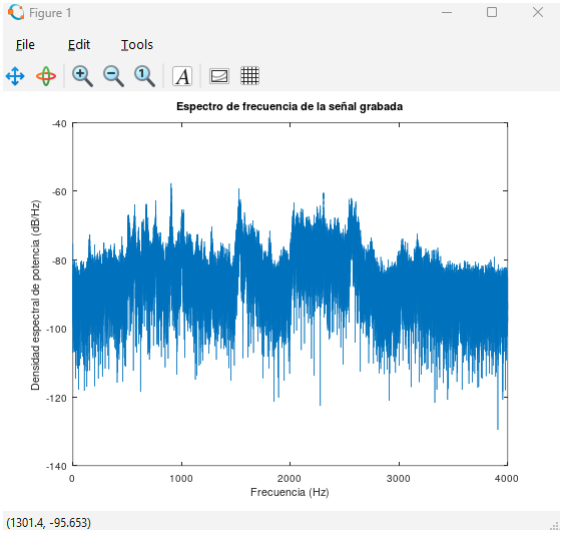

Figura 15: Gráfica de densidad en Octave.

```python
def graficar_espectro_frecuencia():
    try:
        print("Graficando espectro de frecuencia...")
        fs, audio = read('audio.wav')
        N = len(audio)  # Número de muestras de la señal
        f, Sxx = welch(audio, fs, window=hann(N), nperseg=N, noverlap=0)
        plt.plot(f, 10 * np.log10(Sxx))  # Gráfica el espectro de frecuencia en dB
        plt.xlabel('Frecuencia (Hz)')
        plt.ylabel('Densidad espectral de potencia (dB/Hz)')
        plt.title('Espectro de frecuencia de la señal grabada')
        plt.show()
    except Exception as e:
        print("Error al graficar el espectro de frecuencia:", e)
```

Figura 16: Código gráfica de densidad en Python.

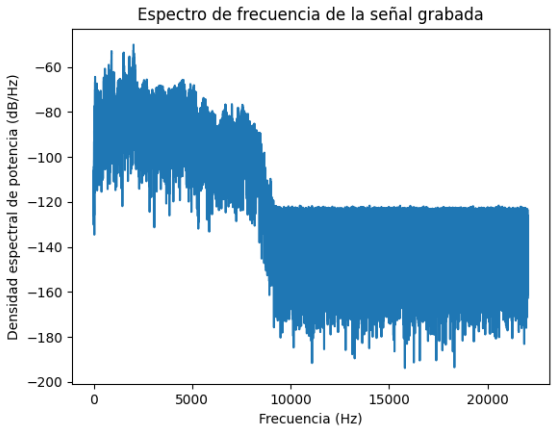

Figura 17: Gráfica de densidad en Python.

III. APLICACIONES

El programa tiene múltiples aplicaciones:

- Análisis básico de señales de audio en tiempo y frecuencia.
- Preparación de señales para su procesamiento en proyectos más avanzados.

- Uso educativo para entender conceptos fundamentales del PDS.

En este caso en particular se capturaron 300 imagenes del video antes mencionado, las cuales sirven para crear un sistema de reconcimiento, entre mas muestras con distitas expresiones faciales, lo hace mas fiable y robusto.

IV. CONCLUSIÓN

- Este programa ejemplifica cómo utilizar Octave para realizar operaciones básicas de procesamiento digital de señales. Las funciones de grabación, reproducción y análisis espectral ofrecen una introducción práctica a los fundamentos del PDS, con aplicaciones en áreas como el análisis de audio y la educación en ingeniería, así mismo se puede realizar este código en python, teniendo un poco mas de complicaciones y necesitando importar muchas mas librerias de las que se necesita en octave, decidiendo que es mas eficiente hacer este tipo de programas en Octave que en Python por su simplicididad y practicidad

V. REPOSITORIO EN GITHUB

`https://github.com/Marianne8934/Tareas-y-proyectos`