TAREA 4 : Procesamiento digital de señales*

Pablo Andres Montufar Perez, 201902235[1]

[1] *Facultad de Ingeniería, Escuela de Mecánica Eléctrica, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

El procesamiento digital de señales (PDS) es una disciplina que estudia cómo manipular señales discretas en el dominio del tiempo o de la frecuencia para analizarlas, procesarlas o transformarlas. En este proyecto, desarrollamos un programa en Octave que permite grabar, reproducir, graficar y analizar la densidad espectral de potencia de señales de audio.

El proposito del programa en octave es poder realizar el procesamiento de señakes a traves de diferentes cases y codigo que nos permitiran en este caso grabar nuestra voz del 1 a l 10 y mostrar una grafica de su espectro.

I. INTRODUCCIÓN

El procesamiento digital de señales se utiliza ampliamente en aplicaciones como comunicaciones, reconocimiento de voz, procesamiento de imágenes y análisis de audio. Este proyecto tiene como objetivo implementar un sistema básico de manipulación de señales de audio en Octave, un entorno de código abierto similar a MATLAB, utilizando funciones integradas y herramientas de análisis espectral.

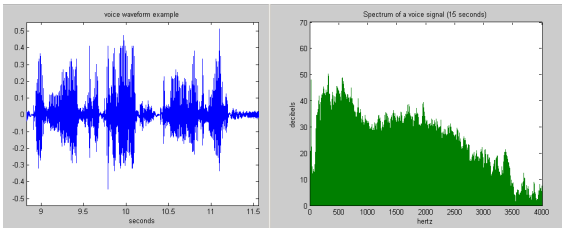

Figura 1: Representación de una señal de audio y su análisis espectral.

II. CONFIGURACIÓN INICIAL

A. Requisitos

Para ejecutar el programa, se utilizó Octave debido a su compatibilidad con herramientas de análisis digital y su soporte para bibliotecas como `signal`. Además, el entorno asegura una experiencia similar a MATLAB para proyectos académicos.

B. Estructura de Carpetas

El programa genera y almacena automáticamente los archivos de audio procesados (`audio.wav`) en el directorio de trabajo actual.

* PROYECTOS DE COMPUTACION APLICADA A I.E. Sección A

III. EXPLICACIÓN DEL CÓDIGO

El código implementa un menú interactivo que permite al usuario realizar cuatro operaciones principales: grabar audio, reproducirlo, graficar su forma de onda y analizar su densidad espectral de potencia. A continuación, se detalla cada opción:

A. Creación del Menú

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.

```
disp('Seleccione una opción:')
disp('1. Grabar')
disp('2. Reproducir')
disp('3. Graficar')
disp('4. Graficar densidad espectral de potencia')
disp('5. Salir')
```

```
>> Tarea4
Seleccione una opción:
1. Grabar
2. Reproducir
3. Graficar
4. Graficar densidad espectral de potencia
5. Salir
Ingrese su elección: 1
Ingrese la duración de la grabación en segundos: 10
Comenzando la grabación...
Grabación finalizada.
Archivo de audio grabado correctamente: audio.wav
Seleccione una opción:
1. Grabar
2. Reproducir
3. Graficar
4. Graficar densidad espectral de potencia
5. Salir
Ingrese su elección: 2
Reproduciendo audio...
Seleccione una opción:
1. Grabar
2. Reproducir
3. Graficar
4. Graficar densidad espectral de potencia
5. Salir
Ingrese su elección: 3
Gráfica generada.
```

Figura 2: codigo en octave

B. Opción 1: Grabar Audio

Esta opción permite al usuario grabar audio mediante el objeto `audiorecorder`. La duración de la grabación

se especifica en segundos, y el audio se guarda como un archivo `.wav`.

```
recObj = audiorecorder;
recordblocking(recObj, duracion);
data = getaudiodata(recObj);
audiowrite('audio.wav', data, recObj.SampleRate);
```

```
Ingrese su elección: 4
Graficando espectro de frecuencia...
Espectro de frecuencia generado.
Seleccione una opción:
1. Grabar
2. Reproducir
3. Graficar
4. Graficar densidad espectral de potencia
5. Salir
Ingrese su elección: 5
Saliendo del programa...
>>
```

Figura 3: codigo en octave

C. Opción 2: Reproducir Audio

Lee el archivo `audio.wav` previamente generado y lo reproduce utilizando la función `sound`.

```
[data, fs] = audioread('audio.wav');
sound(data, fs);
```

```
if(exist('OCTAVE_VERSION','builtin')~=0)

pkg load signal;
end
%MENU
opcion = 0;
while opcion ~=5
  disp('Seleccione una opcion')
  disp('1.Grabar')
  disp('2.Reproducir')
  disp('3.Graficar')
  disp('4.Graficar densidad')

  disp('5. Salir')
  opcion = input('Ingrese su elección:');
switch opcion
  case 1
    try
      duracion = input('Ingrese la duración de la grabación
      disp('Comenzando la grabación');
      recObj = audiorecorder;
      recordblocking(recObj, duracion);
      disp('Grabacion finalizada');
      data= getaudiodata(recObj);
      audiowrite('audio.wav', data, recObj.SampleRate);
      disp('Archivo de audio grabado correctamente');
    catch
      disp('Error al grabar audio');
    end_try_catch
```

Figura 4: codigo en octave

D. Opción 3: Graficar la Forma de Onda

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

```
tiempo = linspace(0, length(data)/fs, length(data));
plot(tiempo, data);
xlabel('Tiempo (s)');
ylabel('Amplitud');
```

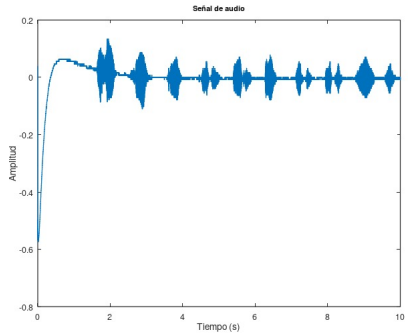

Figura 5: Forma de onda de la señal de audio.

E. Opción 4: Densidad Espectral de Potencia

Utilizando la función `pwelch`, 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.

```
[Sxx, f] = pwelch(audio, ventana, [], [], FS);
plot(f, 10*log10(Sxx));
xlabel('Frecuencia (Hz)');
ylabel('Densidad espectral de potencia (dB/Hz)');
```

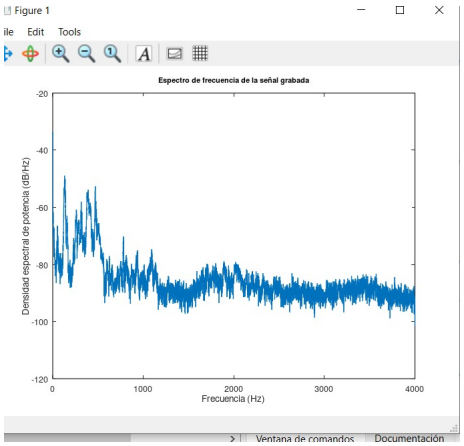

Figura 6: Densidad espectral de potencia de la señal grabada.

IV. APLICACIONES Y PROPÓSITO

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.

V. 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.

VI. REPOSITORIO DEL PROYECTO

El código fuente de este proyecto, junto con ejemplos y otros recursos, está disponible en el siguiente enlace:

Repositorio en GitHub

Este repositorio contiene el código original, las mejoras realizadas y las gráficas generadas por los programas descritos en este documento.