Análisis de Señales en Tiempo y Frecuencia

Pablo Bermeo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
pablo.bermeo@ucuenca.edu.ec

Sebastián Guazhima
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
sebastian.guazhima@ucuenca.edu.ec

Tyrone Novillo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
tyrone.novillo@ucuenca.edu.ec

***Resumen*—The following report focuses on the processing and analysis of audio signals using MATLAB. From the initial import of audio files in WAV format to the application of transforms and ideal filters in the frequency domain, the implemented code enables understanding and manipulation of sound signals across different domains. Emphasis is placed on spectral analysis through the Fourier Transform, exploring how frequency domain filtering selectively affects the spectral components of audio signals. The application of low-pass, high-pass, and band-pass filters on the spectrum reveals the fundamental relationship between time and frequency domains in audio signal processing, offering valuable insights for sound signal analysis and processing.**

***Keywords*—** Audio signals, MATLAB, Fourier Transform, Frequency domain filters, Signal processing, Spectral analysis

I. Introducción

El hecho de que la mayoría de señales tienen su representación en el dominio de la frecuencia mediante la Transformada de Fourier permite procesar señales desde el dominio de la frecuencia. Desde una perspectiva de sistemas lineales y señales, se explora como el estudio de las señales de audio en el dominio de la frecuencia permite una comprensión más profunda de su estructura espectral y facilita el análisis y procesamiento eficaz de dichas señales.

El enfoque en el dominio de la frecuencia proporciona una perspectiva única sobre las señales de audio al permitir la visualización y manipulación de sus componentes espectrales. Esto se enmarca en los principios de sistemas lineales, donde la respuesta de un sistema a una entrada específica puede ser analizada a través de la convolución en el dominio del tiempo o, de manera equivalente, mediante la multiplicación en el dominio de la frecuencia. El entendimiento de estas propiedades permite aplicar técnicas de filtrado y transformación que afectan selectivamente ciertas bandas espectrales de ciertas señales.

II. Marco teórico

II-A. Transformada de Fourier

La transformada de Fourier es un concepto matemático fundamental que descompone una señal en sus componentes de frecuencia. Esta herramienta, propuesta por Jean-Baptiste Joseph Fourier en el siglo XIX, permite analizar señales en el dominio de la frecuencia, revelando la presencia y magnitud de cada frecuencia que compone una señal continua en el dominio del tiempo. Al aplicar la transformada de Fourier a una señal, se obtiene su espectro de frecuencias, representando la contribución de cada componente de frecuencia a la señal original. Este espectro proporciona información detallada sobre las frecuencias presentes, facilitando el análisis de la estructura armónica y la composición de la señal en términos de sus frecuencias fundamentales y armónicas. [1]

El espectro de frecuencias derivado de la transformada de Fourier revela la composición espectral de una señal, mostrando las amplitudes y fases de las diferentes frecuencias. Esta representación es crucial en disciplinas como el procesamiento de señales, la comunicación, la física y la ingeniería, donde el análisis detallado de las características de frecuencia de una señal es esencial. La capacidad de analizar una señal en términos de sus componentes de frecuencia permite comprender su comportamiento en diferentes dominios, desde la música y el procesamiento de imágenes hasta la modulación de señales en sistemas de comunicación. [2]

En resumen, la transformada de Fourier es una operación matemática que transforma una señal desde el dominio del tiempo al dominio de la frecuencia. Su fórmula matemática se expresa generalmente como:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt$$

La antitransformada de Fourier, por otro lado, realiza la operación inversa, transformando una señal desde el dominio de la frecuencia al dominio del tiempo. Su fórmula matemática se expresa como:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df$$

II-A1. Transformada de Fourier de señales de audio:

La transformada de Fourier de un archivo de sonido implica la conversión de la representación de la señal de audio del dominio del tiempo al dominio de la frecuencia. Este proceso es fundamental para descomponer la señal en sus componentes espectrales, revelando las diversas frecuencias que la componen. En términos prácticos, este procedimiento implica el análisis de la señal de audio para determinar la contribución de cada frecuencia presente en la señal [3]. Esta transformación matemática permite entender la estructura armónica del sonido, mostrando qué frecuencias están presentes y en qué medida influyen en la señal de audio.

La transformada de Fourier aplicada al espectro de una señal de audio es fundamental para discernir y reconocer los distintos instrumentos musicales o las características vocales presentes en la señal sonora. Este análisis espectral permite identificar las frecuencias predominantes asociadas a instrumentos musicales específicos o a las distintas tonalidades de las voces humanas. La distribución y amplitud de las frecuencias en el espectro pueden revelar información sobre la instrumentación, el tono, la textura y otros atributos que caracterizan los distintos componentes de una señal de audio [4].

El espectro de frecuencias proporciona información detallada sobre la composición espectral del sonido. Al convertir una señal de audio al dominio de la frecuencia, se pueden identificar y sobre todo manipular las características espectrales del sonido mediante el procesamiento en frecuencia de la señal.

II-B. Transformada rápida de Fourier

La *Fast Fourier Transform* (FFT) es una forma óptima de realizar la Transformada Discreta de Fourier (DFT). Entiéndase *óptimo* en términos computacionales: cálculos realizados por computadores. Al enmarcar la FFT a entornos computacionales, se obliga a partir de muestreos tanto en el dominio de la frecuencia cuanto en el dominio

de tiempo. En este contexto, a continuación se presenta la transición entre la DFT la FFT.

En primer lugar, considérese una señal continua, presentada en la Figura $f(t)$ que sería muestreada en intervalos de tiempo T_s. Luego de ser muestreada, la señal resultante resulta en una señal discreta $x[n]$, que se presenta en la Figura 2.

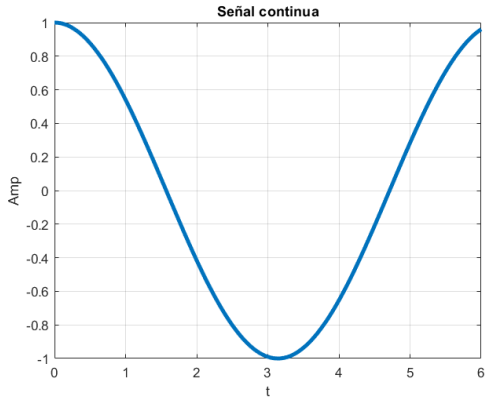

Figura 1: Señal continua de entrada

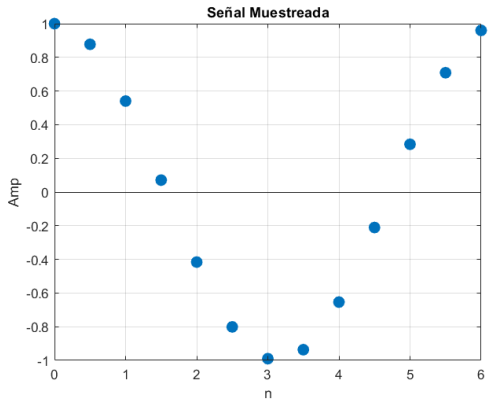

Figura 2: Señal discreta luego del muestreo.

Es en este punto que se introduce la transición al dominio de la frecuencia de la señal muestreada. Dado que se trata de una señal discreta $x[n]$, se emplea la Transformada de Tiempo Discreto de Fourier (DTFT) tal que:

$$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{j\omega n}$$

Dado que el muestreo ocurre en señales de tiempo medidas desde $t = 0$, las N muestras finitas producidas ocurren desde $n = 0$ hasta $n = N - 1$. Por lo tanto la DTFT se acopla tal que:

$$X(e^{j\omega}) = \sum_{0}^{N-1} x[n]e^{j\omega n}$$

En este punto, se requiere recalcar que se busca una expresión capaz de ser tratada mediante un computador: existe una memoria finita. En tanto, el dominio ω, que teóricamente se comprende de valores infinitos, no es aceptable. Por lo que la DTFT, que teóricamente debe ser una señal continua, pasa a ser una señal discreta medida en múltiplos k de una frecuencia $\frac{2\pi}{N}$. Finalmente, se deriva la expresión para una Transformada Discreta de Fourier (DFT):

$$X[k] = \sum_{0}^{N-1} x[n]e^{jk\frac{2\pi}{N}n}$$

El camino hacia la FFT parte de las simetrías generadas en la $X[k]$. Considerando

$$e^{-j\frac{2\pi}{N}} = W_N$$
$$X[k] = \sum_{0}^{N-1} x[n]W_N^k$$

Se origina la simetría entre conjugados:

$$W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$$

y la periodicidad en n y k sobre N:

$$W_N^{kn} = W_N^{K(N+n)} = W_N^{(k+N)n}$$

La manera de optimizar la DFT considera un partición entre los valores de n pares e impares. De tal forma que se tienen dos DFT:

$$X[k] = \sum_{n\text{ par}} x[n]W_N^{Kn} + \sum_{n\text{ impar}} x[n]W_N^{Kn}$$

Dado que cada DFT opera con $\frac{N}{2}$ puntos, el limite de las sumatorias cambia. Se considera, a fin de expresar paridades, la introducción del índice r. Con lo que resulta:

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_N^{K(2r)} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_N^{K(2r+1)}$$

Pero, dado que:

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_N^{K(2r)} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_N^{K(2r+1)}$$

Esto nos lleva a la expresión recursiva empleada para el desarrollo de la FFT. Se tiene que

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_{\frac{N}{2}}^{Kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_{\frac{N}{2}}^{Kr}$$

La última expresión es la muestra de que es posible dividir la DFT en procesos singulares entre pares e impares, resultando la expresión definitiva:

$$X[k] = X_{\text{par}}[k] + W_N^k X_{\text{impar}}[k]$$

Esta expresión se ha denominado recursiva ya que para un conjunto N de muestras, $X[k]$ se podría calcular a partir de una única partición de pares e impares sobre $x[n]$. El punto clave es que cada partición puede volver a dividirse entre pares e impares de forma iterativa.

A partir de esta última afirmación, es posible demostrar que la complejidad de la FFT es de $0(Nlog2(N))$ que es inferior a $O(N^2)$ de la DFT.

Por último, la implementación en *Matlab* de este proceso, puede visualizarse en la documentación del comando `fft`. En la documentación se indica que la expresión empleada, dado un vector de n elementos es la siguiente:

$$Y(k) = \sum_{j=1}^{n} X(j)W_n^{(j-1)(k-1))}$$

Expresión que es similar a la empleada para derivar la partición entre pares e impares de la FFT.

II-C. Respuesta de un sistema LTI en el dominio de la frecuencia

El comportamiento de los sistemas LTI, a partir de la Transformada de Fourier, puede ser caracterizado en el dominio del tiempo o de la frecuencia. Los sistemas LTI se caracterizan a través de la respuesta al impulso $h(t)$ debido a que determinan una salida $y(t)$ marcada por $y(t) = x(t) * h(t)$. Sin embargo, en el dominio de la frecuencia, la salida del sistema LTI, se caracteriza por la respuesta en frecuencia del sistema $H(j\omega)$. A partir de la propidedad de la convolución de la Transformada de Fourier, el espectro de la salida $y(t)$, $Y(j\omega)$, se consigue a partir de la multiplicación de la respuesta en frecuencia y el espectro de la entrada

$$Y(j\omega) = X(j\omega)H(j\omega)$$

II-D. Teorema del muestreo

Suponga $x(t)$ como una señal con ancho de banda limitado con $X(jw) = 0$ para $|\omega| > \omega_M$. Entonces $x(t)$ es **únicamente** determinado por sus muestras $x(nT)$, $n = 0, \pm1, \pm2, \cdot$, si

$$\omega_s > 2\omega_M$$

donde

$$\omega_s = \frac{2\pi}{T}$$

Dadas la muestras, es posible reconstruir la señal a través de un tren de pulsos ideal y un filtro paso bajo. El filtro paso bajo tiene un valor de ganancia T y una frecuencia de corte $\omega_M < \omega_c < \omega_s - \omega_M$. [2]

III. Desarrollo

A continuación, se explica el proceso para implementar un script de Matlab el cual ejecuta el filtrado entre un archivo de audio y diferentes filtros.

III-A. Importación del audio

Antes de importar el audio, se consideró trabajar con archivos con extensión `.wav` debido a que son un formato de audio sin pérdida que almacenan datos de audio en bruto sin comprimir. Para importar el archivo de audio se utilizó la función `audioread` de MATLAB. Esta función devuelve dos variables: una matriz que contiene los datos de audio y la frecuencia de muestreo del audio. Posteriormente, se calcula la duración total del audio, obteniendo la longitud de los datos de audio y dividiéndola por la frecuencia de muestreo.

Luego, se procede a obtener una señal promediada de ambos canales de audio. Esto se realiza sumando los datos de los dos canales y dividiendo el resultado entre 2 para obtener un promedio. El objetivo es generar una señal combinada que represente una mezcla equilibrada de ambos canales. Esta señal promediada, se utiliza comúnmente para simplificar análisis o procesamientos posteriores de la señal de audio, al combinar la información de ambos canales en una única señal.

III-B. Análisis en Frecuencia

Para realizar el análisis en el dominio de la frecuencia sobre la señal de audio promediada se aplica la Transformada de Fourier a la señal de audio, utilizando la función `fft`. La Transformada de Fourier convierte la señal del dominio del tiempo al dominio de la frecuencia, permitiendo visualizar las diferentes componentes de frecuencia presentes en la señal de audio.

Posteriormente, se utiliza la función `fftshift` para reorganizar el espectro de frecuencia resultante, de manera que la frecuencia cero quede situada en el centro del espectro. Esto es fundamental para una correcta interpretación de las frecuencias presentes en la señal, ya que sin esta reorganización, la información del espectro se distribuiría de forma no centrada.

Por motivos de graficación, se genera un eje de frecuencias utilizando la función `linspace`, el cual abarca desde la frecuencia negativa máxima hasta la frecuencia positiva máxima, manteniendo el mismo número de puntos que el espectro de frecuencia. Por último, se calcula la magnitud del espectro utilizando la función `abs`, lo que proporciona información acerca de la amplitud de cada componente de frecuencia en la señal de audio.

III-C. Creación y Análisis de Filtros

En un sistema LTI, una señal de entrada (en este caso audio), para pasar través de una respuesta al impulso, obteniendo a la salida el audio modificado. Por lo que es necesario generar una respuesta al impulso $h(t)$, sin embargo, necesitamos realizar un filtrado a la señal de entrada, es decir, dejar o rechazar frecuencias de acuerdo al tipo de filtro. Por lo tanto, es importante trabajar el filtro en el dominio de la frecuencia y usar la respuesta en frecuencia $H(j\omega)$ de la respuesta al impulso.

Entonces, al trabajar en frecuencia se genera un tipo de filtro $H(j\omega)$ con determinada frecuencia de corte.

Se generan tres filtros: paso bajo, pasa banda y paso alto en el dominio de la frecuencia utilizando una condición booleana que preserva frecuencias dentro de la curva de cada tipo de filtro.

III-C1. Filtro Paso Bajo: Para obtener un "LPF"(Low Pass Filter), se construye a partir de su representación matemática:

$$H_{LPF} = \begin{Bmatrix} 1 & |\omega| < w_c \\ 0 & o.w \end{Bmatrix} \quad (1)$$

Es decir, que se forma un pulso de magnitud 1, que comprende entre los valores de frecuencia de corte. También, puede ser expresado en frecuencia lineal (f_c).

Para construir este filtro, se toma del vector de frecuencias llamado `frecuencias`, comprendido entre los valores de frecuencia de la señal de audio.

Usando la función `abs` de valor absoluto, tomamos los valores del vector de frecuencias y limitamos el valor absoluto a frecuencias menores a una f_c específica y magnitud 1(normalizada), es decir por ejemplo: `abs(frecuencias)<=fc`, obtenemos el filtro pasabajos ideal como se muestra en la Figura 3.

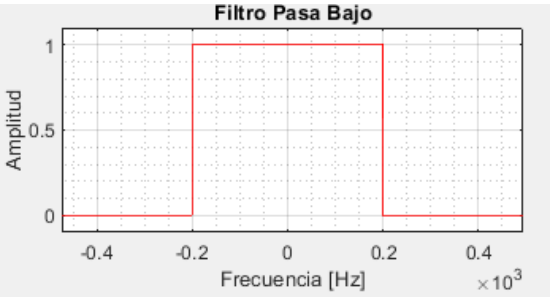

Figura 3: Filtro pasa bajo ideal

III-C2. Filtro Pasa Banda: En un "BPF"(Band Pass Filter), la construcción es similar, tomamos una representación del filtro:

$$H_{LPF} = \begin{Bmatrix} 1 & \omega_{c1} < |\omega| < \omega_{c2} \\ 0 & o.w \end{Bmatrix} \quad (2)$$

Como se observa en la ecuación, se da valores de magnitud 1 a valores de frecuencias ω que estén entre las frecuencias de corte c1 y c2 (siendo $\omega_{c1} < \omega_{c2}$).

Entonces para obtener este filtro pasa banda, determinamos dos frecuencias de corte distintas y usamos `abs(frecuencias)>= fc1`, que construirá una función a partir de una frecuencia $fc1$ y

`abs(frecuencias)<= fc2` que construirá otra función hasta valores con frecuencia $fc2$.

Entonces, unimos las dos partes realizando una intersección entre ellas o una multiplicación y encontramos el filtro BPF ideal de la Figura 4.

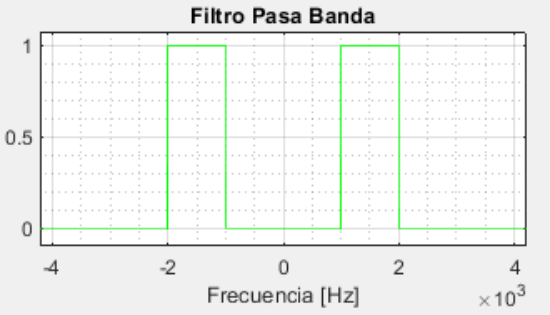

Figura 4: Filtro pasa banda ideal

III-C3. Filtro Paso Alto: Finalmente, para el "HPF"(High Pass Filter), tomamos su representación que es a siguiente:

$$H_{LPF} = \begin{Bmatrix} 1 & |\omega| > \omega_c \\ 0 & o.w \end{Bmatrix} \tag{3}$$

Entonces lo que realizamos es, tomar el valor absoluto de un rango de frecuencias y limitarlo a que construya a partir de una frecuencia de corte de determinada, es decir, `abs(frecuencias)>= fc`. Entonces obtendremos un filtro pasa altos ideal como se muestra en la figura 5

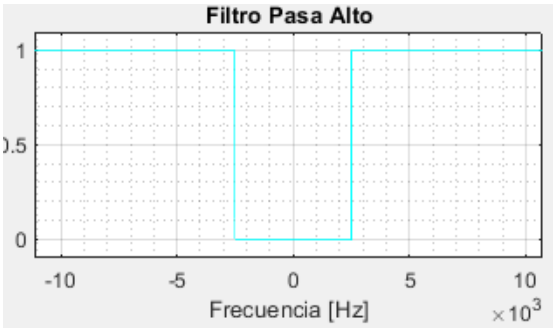

Figura 5: Filtro pasa alto ideal

III-D. Aplicación de los filtros al Espectro de Frecuencia

Una vez definido el comportamiento de los tres sistemas mediante la respuesta en frecuencia de cada uno de ellos, se aplican los filtros al espectro de frecuencia original multiplicándolos. En el dominio del tiempo, esta operación conlleva la operación de convolución, la cual se traduce a una multiplicación en el dominio de la frecuencia cuando se trabaja con la Transformada de Fourier. En este caso, los filtros pasa alto, bajo y banda modifican selectivamente las componentes espectrales de la señal original. Estos filtros alteran la respuesta en frecuencia del sistema, permitiendo el paso o la atenuación de diferentes rangos de frecuencia.

III-E. Transformada Inversa y Análisis en el Dominio del Tiempo

Al aplicar el sistema LTI a la señal de audio, usando el dominio en frecuencia, la Transformada de Fourier y la respuesta en frecuencia, es importante realizar un análisis del sistema nuevamente, en el dominio del tiempo, es decir, analizar el comportamiento de los filtros en el tiempo y posteriormente un análisis de la señal de salida en el tiempo.

Por ello, se aplica la transformada inversa de Fourier, en esta parte detallaremos la parte de filtros o las respuestas en frecuencias al pasar a través de la $\mathcal{F}^{-1}\{H(j\omega)\}$.

Entonces, aplicando la funcion de MATLAB `ifft`, que es la inversa de la transformada rápida de fourier, y realizando arreglos, encontramos la representación de cada tipo de filtro ahora en el dominio del tiempo, es decir $h(t)$.

Además, Como se conoce, un pulso cuadrado en frecuencia, equivale a un $sinc$ en el tiempo y al analizar las funciones, es mas sencillo aplicar la multiplicación de un filtro por un señal en frecuencia que convolucinar en el tiempo, se da la razón al trabajar aplicando la Transformada de Fourier para comprender que cambios se aplica al pasar por un filtro que hacerlo en el tiempo.

De la misma realizamos la transformada inversa de Fourier a la señal de salida del sistema, es decir al audio filtrado por los diferentes filtros. En frecuencia, el audio filtrado es claramente visible dejando en claro que frecuencias fueron rechazadas y cuales no. Entonces, en el dominio del tiempo para cada salida de audio filtrado, se obtendrá señales diferentes a la original, sin embargo, en el tiempo es complicado analizar que está ocurriendo en la señal y por ello se enfatizará que en frecuencia es mejor entender que ocurre con la manipulación de señales.

IV. Resultados

IV-A. Gráficas del procesamiento en tiempo y frecuencia

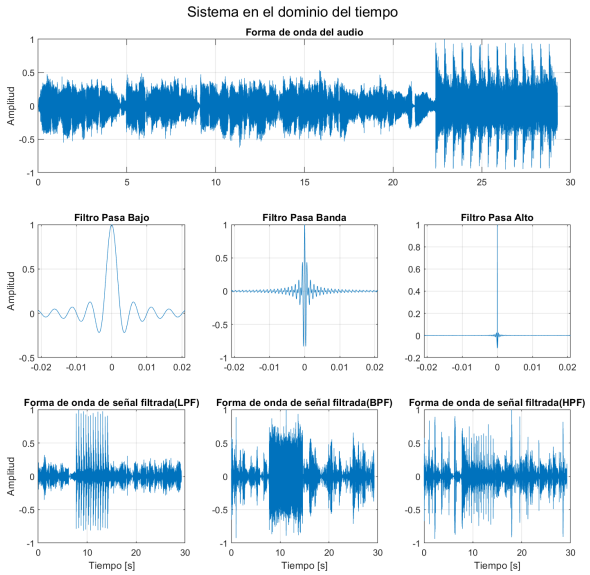

Figura 6: Señal de entrada, respuestas al impulso de los filtros y señales de salida en el dominio del tiempo

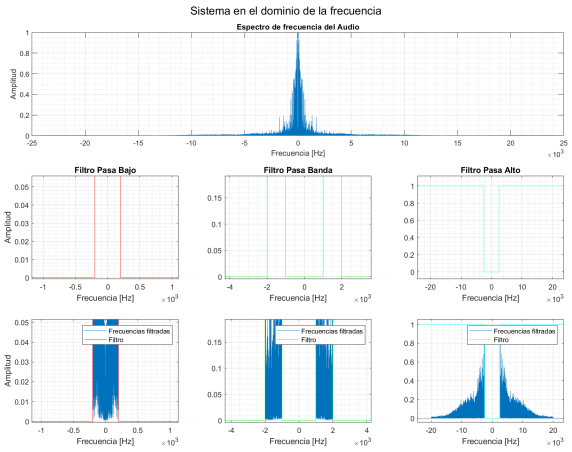

Figura 7: Señal de entrada, respuestas en frecuencia de los filtros y señales de salida en el dominio de la frecuencia

V. Conclusiones

Se ha evidenciado la relevancia del análisis en el dominio de la frecuencia para comprender las características espectrales de las señales de audio. La aplicación de la Transformada de Fourier ha permitido descomponer la señal de audio en formato .wav en sus componentes frecuenciales, facilitando la identificación y comprensión de sus distintas bandas espectrales. La propiedad de multiplicación en el dominio de la frecuencia ha demostrado ser crucial, ya que permite aplicar filtros ideales de manera eficiente, como pasa banda, pasa bajo y pasa alto. Esta propiedad ha posibilitado el análisis y manipulación selectiva de las componentes frecuenciales, destacando la relación entre las operaciones en frecuencia y la modificación de la señal original.

La implementación de los filtros ideales ha resultado en salidas de audio modificadas en comparación con la señal original. El filtro pasabanda ha mantenido selectivamente una banda específica de frecuencias, resultando en una salida que resalta esos rangos particulares. El filtro pasabajo ha atenuado las frecuencias superiores al valor de corte, resultando en una salida con menos presencia de agudos, mientras que el filtro pasaalto ha suprimido las frecuencias por debajo del valor de corte, generando una salida con menos presencia de graves. Estas diferencias auditivas resaltan la influencia directa de los filtros en la percepción del audio.

La utilización del `fftshift` en MATLAB ha sido esencial para asegurar que la frecuencia cero esté en el centro del espectro, garantizando una interpretación visual adecuada de las transformadas y facilitando la manipulación de las señales en el dominio de la frecuencia.

Referencias

[1] R. N. Bracewell, *The Fourier Transform and Its Applications*, 3rd ed. McGraw-Hill, 2000.

[2] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*, 3rd ed. Pearson Education, 2010.

[3] J. B. Allen, *The Art and Science of Acoustic Design*. ..., 1997.

[4] P. Stoica and R. L. Moses, *Spectral Analysis of Signals*. Pearson Education, 2005.

VI. Anexos

VI-A. Github

La dirección https://github.com/thyron001/procesamiento_senales_frecuencia corresponde al repositorio que contiene los ficheros utilizados en este reporte como el script y el audio de prueba.

VI-B. Código

```
clc;
clear all;
close all;

%% Importación del audio
% Se lee el archivo de audio "audio.wav"
[datos_audio, frecuencia_muestreo] = audioread("
    counting_stars_cut.wav");

% Se calcula la duración del audio
duracion_audio = length(datos_audio) /
    frecuencia_muestreo;

% Se obtiene la señal promediada de ambos canales
     de audio
senal_promediada = 0.5 * (datos_audio(:, 1) +
    datos_audio(:, 2)).'; % Transpuesta



%% Análisis en frecuencia
espectro_audio = fftshift(fft(senal_promediada));
     % Transformada de Fourier centrada
frecuencias = linspace(-frecuencia_muestreo/2,
    frecuencia_muestreo/2, length(espectro_audio)
    );
mag_espectro = abs(espectro_audio);

% Gráfico del espectro de frecuencia del audio
figure(2);
sgtitle('Sistema en el dominio de la frecuencia')
    ;
subplot(3,3,[1,3]);
plot(frecuencias, mag_espectro/max(mag_espectro))
    ;
title("Espectro de frecuencia del Audio");
xlabel("Frecuencia [Hz]");
ylabel("Amplitud");
grid on, grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

%% Creación de Filtros ideales

% Filtro paso bajo
filtro_paso_bajo = (abs(frecuencias)<=200); %
    Filtro que mantiene frecuencias menores a 400
     Hz
filtro_pasa_banda = ((abs(frecuencias)>=1000).*(
    abs(frecuencias)<=2000));
filtro_pasa_alta = (abs(frecuencias)>=2500);

% Gráfico del filtro paso bajo
subplot(3,3,4);
plot(frecuencias, filtro_paso_bajo,'r');
title("Filtro Pasa Bajo");
xlabel("Frecuencia [Hz]");
ylabel("Amplitud");
ylim([-0.1 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

% Gráfico del filtro paso banda
subplot(3,3,5);
```

```
plot(frecuencias, filtro_pasa_banda,'g');
title("Filtro Pasa Banda");
xlabel("Frecuencia [Hz]");
ylim([-0.1 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

% Gráfico del filtro paso alto
subplot(3,3,6);
plot(frecuencias, filtro_pasa_alta,'c');
title("Filtro Pasa Alto");
xlabel("Frecuencia [Hz]");
ylim([-0.1 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

%%---------- Aplicación del filtros
    ------------%%
filtrado_lpf = espectro_audio .* filtro_paso_bajo
    ;
filtrado_bpf = espectro_audio .*
    filtro_pasa_banda;
filtrado_hpf = espectro_audio .* filtro_pasa_alta
    ;
%%-----------------------------------------------%%

% Gráfico del espectro de frecuencia filtrado y
    superposición con el filtro
%%-------------------------filtro pasa bajo
    ------------------------------%%
figure(2);
subplot(3,3,7);
plot(frecuencias, abs(filtrado_lpf)/max(abs(
    filtrado_lpf))); % Espectro filtrado
hold on;
plot(frecuencias, filtro_paso_bajo,'r'); %
    Superposición del filtro
legend("Frecuencias filtradas","Filtro");
xlabel("Frecuencia [Hz]");
ylabel("Amplitud");
ylim([0 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

%%-------------------------filtro pasa banda
    ------------------------------%%
figure(2);
subplot(3,3,8);
plot(frecuencias, abs(filtrado_bpf)/max(abs(
    filtrado_bpf))); % Espectro filtrado
hold on;
plot(frecuencias, filtro_pasa_banda,'g'); %
    Superposición del filtro
legend("Frecuencias filtradas","Filtro");
xlabel("Frecuencia [Hz]");
ylim([0 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

%%-------------------------filtro pasa alto
    ------------------------------%%
figure(2);
subplot(3,3,9);
plot(frecuencias, abs(filtrado_hpf)/max(abs(
    filtrado_hpf))); % Espectro filtrado
hold on;
plot(frecuencias, filtro_pasa_alta,'c'); %
    Superposición del filtro
legend("Frecuencias filtradas","Filtro");
xlabel("Frecuencia [Hz]");
ylim([0 1.1])
grid on; grid minor;
ax = gca;
ax.XAxis.Exponent = 3;

%%-----------------Graficas del dominio del
    tiempo--------------------------
%%
%% Gráfico de la forma de onda del audio
tiempo = linspace(0, duracion_audio, length(
    senal_promediada));

figure(1);
sgtitle('Sistema en el dominio del tiempo');
subplot(3,3,[1,3]);
plot(tiempo, senal_promediada/max(
    senal_promediada));
title("Forma de onda del audio");
ylabel("Amplitud");
grid on;

%% Transformada inversa para pasar el filtro al
    dominio del tiempo
o = length(tiempo)/2;
r = o-1000:o+1000;
%%-------------------------filtro paso bajo
    ------------------------------%%
lpf_tiempo = fftshift(ifft(fftshift(
    filtro_paso_bajo))); % Transformada inversa
lpf_tiempo = lpf_tiempo/max(lpf_tiempo);
t_lpf = linspace(-duracion_audio/2,
    duracion_audio/2, length(lpf_tiempo));

%%-------------------------filtro pasa banda
    ------------------------------%%
bpf_tiempo = fftshift(ifft(fftshift(
    filtro_pasa_banda))); % Transformada inversa
bpf_tiempo = bpf_tiempo/max(bpf_tiempo);
t_bpf = linspace(-duracion_audio/2,
    duracion_audio/2, length(bpf_tiempo));

%%-------------------------filtro pasa alto
    ------------------------------%%
hpf_tiempo = fftshift(ifft(fftshift(
    filtro_pasa_alta))); % Transformada inversa
hpf_tiempo = hpf_tiempo/max(hpf_tiempo);
t_hpf = linspace(-duracion_audio/2,
    duracion_audio/2, length(hpf_tiempo));

% Gráfico del filtros en el dominio del tiempo
figure(1);
subplot(3,3,4);
plot(t_lpf(r), real(lpf_tiempo(r)));
title("Filtro Pasa Bajo");
ylabel("Amplitud");
grid on;

subplot(3,3,5);
plot(t_bpf(r), real(bpf_tiempo(r)));
title("Filtro Pasa Banda");
grid on;

subplot(3,3,6);
plot(t_hpf(r), real(hpf_tiempo(r)));
title("Filtro Pasa Alto");
grid on;

%% Transformada inversa para pasar la señal
    filtrada al dominio del tiempo
salida_tiempo_l = fftshift(ifft(fftshift(
    filtrado_lpf))); % Transformada inversa
salida_tiempo_l =  salida_tiempo_l/max(
    salida_tiempo_l).';
```

```
salida_tiempo_b = fftshift(ifft(fftshift(
    filtrado_bpf))); % Transformada inversa
salida_tiempo_b =  salida_tiempo_b/max(
    salida_tiempo_b).';

salida_tiempo_h = fftshift(ifft(fftshift(
    filtrado_hpf))); % Transformada inversa
salida_tiempo_h =  salida_tiempo_h/max(
    salida_tiempo_h).';

% Gráfico de la forma de onda del audio filtrado

tiempo_l = linspace(0, duracion_audio, length(
    salida_tiempo_l)); %%(LPF)
tiempo_b = linspace(0, duracion_audio, length(
    salida_tiempo_b)); %%(BPF)
tiempo_h = linspace(0, duracion_audio, length(
    salida_tiempo_h)); %%(HPF)

figure(1);
subplot(3,3,7);
plot(tiempo, real(salida_tiempo_l));
title("Forma de onda de señal filtrada(LPF)");
xlabel("Tiempo [s]");
ylabel("Amplitud");
grid on;

subplot(3,3,8);
plot(tiempo, real(salida_tiempo_b));
title("Forma de onda de señal filtrada(BPF)");
xlabel("Tiempo [s]");
grid on;

subplot(3,3,9);
plot(tiempo, real(salida_tiempo_h));
title("Forma de onda de señal filtrada(HPF)");
xlabel("Tiempo [s]");
grid on;

%-----------------------------------------

%% Reconstrucción de la señal a partir del
    espectro filtrado
audio_filtrado_l = ifft(fftshift(filtrado_lpf));
    % Transformada inversa
audio_filtrado_l = real(audio_filtrado_l); % Se
    toma solo la parte real

audio_filtrado_b = ifft(fftshift(filtrado_bpf));
audio_filtrado_b = real(audio_filtrado_b);

audio_filtrado_h = ifft(fftshift(filtrado_hpf));
audio_filtrado_h = real(audio_filtrado_h);


% Reproducción de audios procesados
sound(audio_filtrado_l,frecuencia_muestreo); %
    Audio filtrado (LPF)
pause( duracion_audio + 1 );

sound(audio_filtrado_b,frecuencia_muestreo); %
    Audio filtrado (BPF)
pause( duracion_audio + 1 );

sound(audio_filtrado_h,frecuencia_muestreo); %
    Audio filtrado (HPF)
pause( duracion_audio + 1 );

sound(senal_promediada, frecuencia_muestreo); %
    Audio original
```