La Transformación Wavelet y sus Aplicaciones en el Procesamiento de Imágenes

César Miravete Zarazaga

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Directores del trabajo: Francisco Gaspar y Carmen Rodrigo

10 de julio de 2024

Abstract

The growing need to handle large amounts of data and the importance of maintaining information integrity in critical applications have driven the development of advanced signal analysis and processing techniques. In this context, the Wavelet Transform (WT) offers a robust and efficient solution, providing powerful tools for compression, noise reduction, and multiresolution data analysis.

Wavelet theory was not consistently developed until the 1980s. Nevertheless, in 1909 Alfred Haar presented the Haar wavelet, and it wasn't until 1987 that Belgian mathematician Ingrid Daubechies presented the first orthogonal compactly supported wavelet (known as Daubechies wavelets). From that moment, wavelets began to be a powerful computational tool. Subsequently, in 1989, Stéphane Mallat published what is known as multiresolution analysis (MRA), see [6].

In the first chapter of this work, we will cover all the necessary mathematical concepts to rigorously define what wavelets and the wavelet transform are. A brief description of the Fourier transform will also be provided as motivation. The most important concepts include the definition of Hilbert spaces such as the L^2 space, as well as its completeness and the decomposition of L^2 functions into series as follows. Given $\{\varphi_n\}_{n\in\mathbb{N}}$ a basis of L^2,

$$\forall f \in L^2(a,b),\ f = \sum_{n=1}^{\infty} \langle f, \varphi_n \rangle \varphi_n \text{ in } L^2(a,b).$$

In the second chapter, wavelets will be introduced. Mathematically, a wavelet is a function $\psi \in L^2(\mathbb{R})$ that generates a family of functions through translation and scaling of a mother function:

$$\psi_{j,k} = 2^{j/2}\psi(2^j x - k) \quad \forall j,k \in \mathbb{Z}$$

This family of functions forms an orthonormal basis of $L^2(\mathbb{R})$.

Attention will be given to Haar wavelets and how a function can be decomposed using scaling and wavelet functions as follows,

$$f_N = \sum_{k\in\mathbb{Z}} c_k B_k + \sum_{j=0}^{N-1} \sum_{k\in\mathbb{Z}} d_{j,k} H_{j,k},$$

where B_k is the scaling function and $H_{j,k}$ are the wavelet functions. In a similar way, Daubechies wavelets will be studied,

$$f(t) = \sum_{k=0}^{L-1} a_k \phi_L(t-k) + \sum_{j=0}^{J} \sum_{k=0}^{L-1} d_{j,k} \psi_L(2^j t - k),$$

where $\phi_L(t-k)$ is the scaling function and $\psi_L(2^j t - k)$ are the wavelet functions. The WT will also be studied for its ability to decompose a signal into different frequencies, levels, and resolutions, allowing for detailed analysis. Unlike the Fourier Transform, which provides global frequency information of the signal, the WT offers additional details due to its local nature, which is especially useful for non-stationary and short-duration signals.

Although theoretically continuous signals can exist, in practice, all signals are discrete. Therefore, the Discrete Wavelet Transform (DWT) is primarily used, which transforms the input signal into a series of approximation and detail coefficients, providing a multilevel representation that allows for multiresolution analysis (MRA).

Finally, the third chapter will look into the applications of the Discrete Wavelet Transform. One of the most prominent applications of the DWT is image compression. Algorithms like JPEG2000 use the DWT to decompose the image into frequency sub-bands, addressing some of the issues of previous algorithms like JPEG, which uses the Discrete Cosine Transform (DCT). By applying thresholds to the approximation and detail coefficients, a significant reduction in file size is achieved without considerable loss in visual quality. The DWT enables greater efficiency in compression due to its ability to concentrate the signal's information into a few significant coefficients.

Another important application of the DWT is noise reduction (denoising). This process involves transforming a noisy signal into its wavelet coefficients (approximation and detail) and then, through techniques such as coefficient thresholding and reconstruction using the Inverse Wavelet Transform, recovering the original signal noise free.

Thresholding can be hard or soft, depending on the criteria used to nullify or reduce coefficients below a certain threshold. This method is especially effective for signals with high-frequency noise.

$$\text{Hard thresholding:} \quad \tilde{d}_{j,k} = \begin{cases} d_{j,k}, & \text{if } |d_{j,k}| > \lambda \\ 0, & \text{if } |d_{j,k}| \leq \lambda \end{cases}.$$

$$\text{Soft thresholding:} \quad \tilde{d}_{j,k} = \begin{cases} \operatorname{sgn}(d_{j,k})(|d_{j,k}| - \lambda), & \text{if } |d_{j,k}| > \lambda \\ 0, & \text{if } |d_{j,k}| \leq \lambda \end{cases},$$

where $d_{j,k}$ are the wavelet coefficients, λ is the threshold and $\operatorname{sgn}(d_{j,k})$ is the sign of $d_{j,k}$.

The ability of the DWT to efficiently represent data and its applications in compression and noise reduction make it crucial in various fields. Besides those mentioned, the DWT is used in image and video transmission, time series analysis, anomaly detection, and many other areas. Its flexibility and efficiency ensure its relevance in modern signal processing.

The future of the WT and DWT looks promising, with potential applications in emerging areas such as biomedical signal processing, pattern recognition in large datasets, and artificial intelligence. Continued research in improving algorithms and adapting the WT to new technologies and needs ensures its place as an indispensable tool in signal analysis and processing.

In summary, the Discrete Wavelet Transform (DWT) has proven to be an essential tool in signal analysis and processing, providing an efficient and detailed representation of signals. Its applications in image compression and noise reduction stand out for their effectiveness and efficiency, and its ability to adapt to non-stationary and short-duration signals makes it invaluable in various fields.

Introducción

La creciente necesidad de manejar grandes cantidades de datos y la importancia de mantener la integridad de la información en aplicaciones críticas han impulsado el desarrollo de técnicas avanzadas de análisis y procesamiento de señales. La transformada wavelet ofrece una solución robusta y eficiente para estos desafíos, proporcionando herramientas poderosas para la compresión, la eliminación de ruido y el análisis de datos multirresolución.

El objetivo de este trabajo es mostrar cómo la transformada wavelet proporciona herramientas para las aplicaciones anteriormente nombradas, para ello, en el primer capítulo trataremos todos los conceptos matemáticos necesarios para poder definir de manera rigurosa qué son las wavelets y la transformada wavelet. También se describirá brevemente la transformada de Fourier como motivación. Los conceptos más importantes son la definición de espacios de Hilbert tales como el espacio L^2, así como la completitud del mismo.

En el segundo capítulo se introduciran las wavelets. Matemáticamente, una wavelet es una función $\psi \in L^2(\mathbb{R})$ que genera una familia de funciones mediante traslación y escalado de una función madre:

$$\psi_{j,k} = 2^{j/2}\psi(2^j x - k), \quad \forall j,k \in \mathbb{Z}.$$

Esta familia de funciones forma una base ortonormal de $L^2(\mathbb{R})$. Se prestará especial atención a las wavelets de Haar y a las wavelets de Daubechies.

También se estudiará cómo la transformada wavelet descompone una señal en diferentes frecuencias, niveles y resoluciones, permitiendo así analizar dicha señal. A diferencia de la transformada de Fourier, que proporciona información de la frecuencia global de la señal, la transformada wavelet ofrece detalles adicionales por su naturaleza, que son especialmente útiles para señales no estacionarias y de corta duración.

A pesar de que teóricamente podemos tener señales continuas, en la práctica, todas las señales son discretas. Por eso, trataremos principalmente la conocida como transformada wavelet discreta (DWT), la cual transforma la señal de entrada en un serie de coeficientes de aproximación y detalle, proporcionando una representación por niveles que permite un análisis multirresolución (MRA).

Por último, en el tercer capítulo, se utilizará lo descrito en los dos capítulos anteriores de manera aplicada. Una de las aplicaciones más destacadas de la DWT es la compresión de imágenes, algoritmos como JPEG2000 utilizan la DWT para descomponer la imagen en sub-bandas de frecuencia, solucionando así algunos de los problemas de algoritmos anteriores como el JPEG que utiliza la transformada discreta del coseno. Al aplicar umbrales a los coeficientes de aproximación y detalle, mencionados anteriormente, se consigue una reducción considerable del tamaño de los archivos sin una pérdida significativa de la calidad visual.

Otra de las aplicaciones más destacadas de la DWT es la técnica del *denoising wavelet*, que implica la transformación de una señal ruidosa en sus coeficientes wavelet y, posteriormente, a través de diversas técnicas como la umbralización de los coeficientes y la reconstrucción utilizando la transformada wavelet inversa recuperar la imagen libre de ruido.

En conclusión, la capacidad de la DWT para representar datos es crucial en aplicaciones como la transmisión de imágenes y videos, el análisis de series temporales, la detección de anomalías y muchas otras áreas.

Índice general

Abstract III

Introducción V

1. Espacios de Hilbert, Espacios L^2 y Transformada de Fourier 1
1.1. Espacios de funciones . . . 1
1.2. Transformada de Fourier . . . 4

2. Construcción de wavelets 7
2.1. Wavelet de Haar . . . 7
2.2. Transformada discreta y transformada rápida . . . 10
2.3. Análisis de multirresolución (MRA) . . . 11
2.3.1. Ejemplo . . . 12
2.4. Wavelets de Daubechies . . . 15
2.5. Transformada continua . . . 18

3. Aplicaciones 19
3.1. Reducción de ruido . . . 19
3.2. Compresión de imágenes . . . 22
3.3. Otras aplicaciones . . . 23

A. Código 25
A.1. Representación de series . . . 25
A.2. Wavelets de Daubechies . . . 27
A.3. Algortimo DWT en una señal . . . 28
A.4. Tratamiento de ruido (DWT) . . . 29
A.5. Tratamiento de ruido (MRA) . . . 31
A.6. Compresión . . . 34

Bibliografía 37

Capítulo 1

Espacios de Hilbert, Espacios L^2 y Transformada de Fourier

Para poder abordar aplicaciones de la transformación wavelet, como el procesamiento de imágenes, primero es necesario definir matemáticamente esta transformada así como la función wavelet y los conceptos matemáticos necesarios para su desarrollo.

Nuestro objetivo en este capítulo es presentar los espacios L^2 y los espacios de Hilbert, donde posteriormente definiremos las funciones wavelet. Además, se va a introducir la transformada de Fourier con el objetivo de comparar y motivar la introducción de otro tipo de transformadas.

1.1. Espacios de funciones

Para el contenido de esta sección nos basamos en [[1], capítulos 1 y 5].

Definición 1.1. Un **espacio vectorial** sobre un campo $\mathbb{F}$ es un conjunto V de elementos llamados vectores tal que existe una operación de adición, denotada $+$, definida como

$$\begin{aligned} V \times V &\to V \\ (u, v) &\mapsto u + v, \end{aligned}$$

y $(V, +)$ forma un grupo abeliano. El elemento neutro de este grupo se denota como 0. También definimos una operación de multiplicación por escalar

$$\begin{aligned} \mathbb{F} \times V &\to V \\ (f, v) &\mapsto fv. \end{aligned}$$

Para $f \in \mathbb{F}$ y $v \in V$, denotamos el producto escalar como fv. La multiplicación por escalar satisface las siguientes propiedades.
Para todo $a, b \in \mathbb{F}$ y para todo $u, v \in V$,

(a) $a(u+v) = au + av$,

(b) $(a+b)v = av + bv$,

(c) $a(bv) = (ab)v$,

(d) $1v = v$,

(e) $0v = 0$.

Denotamos este espacio vectorial como $< \mathbf{V}, \mathbb{F} >$.

Definición 1.2. Dados $u, v \in V$, definimos la **distancia** entre u y v como $d(u,v) = |u-v|$. Está definida sobre el espacio vectorial $< V, \mathbb{C} >$, y cumple lo siguiente:

- $d(u,v) \geq 0$ para todo $u, v \in V$. Si $d(u,v) = 0$, entonces $u = v$.
- $d(u,v) = d(v,u)$.
- $d(u,v) \geq d(u,w) + d(w,v)$.

Dado un espacio vectorial $< V, \mathbb{C} >$

Definición 1.3. Un espacio vectorial se dice **espacio pre-Hilbert** si existe una función $< \cdot, \cdot >: V \times V \to \mathbb{C}$ de forma que para $u, v, w \in V$ y $a \in \mathbb{C}$ se tiene

- $< v, v > \in \mathbb{R}$ y $< v, v > \geq 0$. Se tiene que $< v, v > = 0$ si y solo si $v = 0$.
- $< u, v+w > = < u, v > + < u, w >$.
- $< au, v > = a < u, v >$.
- $< u, v > = \overline{< v, u >}$, donde la barra representa la operación conjugar compleja.

Definición 1.4. Dado un espacio pre-Hilbert se dice que una sucesión, $\{v_n\}_{n \in \mathbb{N}}$ es de **Cauchy** si para todo $\varepsilon > 0$ existe $N \in \mathbb{N}$ de forma que para todo $m, n > N$ se tiene que $d(v_m, v_n) < \varepsilon$.
Decimos que la sucesión es **convergente** si existe $v \in V$ de forma que para cualquier $\varepsilon > 0$ existe $N \in \mathbb{N}$ tal que para todo $n > N$ se tiene que $d(v, v_n) < \varepsilon$. Decimos entonces que v es el límite de la sucesión.

Definición 1.5. Un espacio pre-Hilbert decimos que es **completo** si toda sucesión de Cauchy es convergente. Los espacios pre-Hilbert completos reciben el nombre de **espacios de Hilbert**.

Definición 1.6. El espacio $\mathbf{L^2(a,b)}$ es el conjunto de funciones $f : (a,b) \to \mathbb{R}$, tales que $\int_b^a f^2(x)\, dx < \infty$.

Definición 1.7. El **producto escalar** de dos funciones en $L^2(a,b)$ es $< f, g > = \int_a^b f(x)g(x)\, dx$.
Definimos una norma asociada de la siguiente manera, $||f|| = \sqrt{< f, f >} = \sqrt{\int_a^b f^2(x)\, dx}$.

Teorema 1.1. *El espacio $L^2(a,b)$ es completo.*

Demostración. Sea f_N una sucesión de Cauchy en $L^2(a,b)$. Consideramos f_n una subsucesión tal que

$$||f_n(x) - f_{n+1}(x)|| = \sqrt{\int_a^b |f_n(x) - f_{n+1}(x)|^2\, dx} < 2^{-n}.$$

Sea entonces

$$g_m(x) = \sqrt{\sum_{n=1}^{m} |f_n(x) - f_{n+1}(x)|^2},$$

$$\int_a^b |g_m(x)|^2\, dx = \int_a^b \sum_{n=1}^{m} |f_n(x) - f_{n+1}(x)|^2\, dx = \sum_{n=1}^{m} \int_a^b |f_n(x) - f_{n+1}(x)|^2\, dx = \sum_{n=1}^{m} ||f_n(x) - f_{n+1}(x)||^2 < 1,$$

$$\int_a^b |g_\infty(x)|^2\, dx = \int_a^b |\lim_{m\to\infty} g_m(x)|^2\, dx = \int_a^b \lim_{m\to\infty} |g_m(x)|^2\, dx = \lim_{m\to\infty} \int_a^b |g_m(x)|^2\, dx \leq 1.$$

En el desarrollo anterior utilizamos que g_m converge monótonamente. Entonces, por definición de espacio L^2, tenemos que $g_\infty \in L^2(a,b)$. Por lo tanto, $|g_\infty(x)| < \infty$ en casi todo punto de (a,b) y

$$f_m(x) = f_1(x) - \sum_{n=1}^{m-1} (f_n(x) - f_{n+1}(x))$$

converge puntualmente a f. Veamos pues que f es el limite de f_n en sentido L^2. Para cada x tenemos que

$$\begin{aligned}|f_m(x)| &\leq |f_1(x)| + \sum_{n=1}^{m-1} |f_n(x) - f_{n+1}(x)| \\ &= |f_1(x)| + |g_m(x)| \leq |f_1(x)| + |g_\infty(x)|,\end{aligned}$$

y

$$|f_m(x)|^2 \leq (|f_1(x)| + |g_\infty(x)|)^2 \leq 4\,\text{máx}\{|f_1(x)|^2, |g_\infty(x)|^2\}.$$

Por lo tanto $f_1, g_\infty \in L^2$, y por el teorema de convergencia dominada

$$\lim_{m\to\infty} \int_a^b |f_m(x)|^2\,dx = \int_a^b \lim_{m\to\infty} |f_m(x)|^2\,dx = \int_a^b |f(x)|^2\,dx,$$

y

$$\int_a^b |f(x)|^2\,dx = \lim_{m\to\infty} \int_a^b |f_m(x)|^2\,dx < \infty,$$

en consecuencia $f \in L^2$.

$$|f(x) - f_m(x)| \leq |f(x)| + |f_1(x)| + |g_\infty(x)|,$$

por lo tanto

$$|f(x) - f_m(x)| \leq 9\,\text{máx}\{|f(x)|^2, |f_1(x)|^2, |g_\infty(x)|^2\},$$

de nuevo por el teorema de la convergencia dominada

$$\lim_{m\to\infty} ||f - f_m||^2 = \lim_{m\to\infty} \int_a^b |f(x) - f_m(x)|^2\,dx = \int_a^b \lim_{m\to\infty} |f(x) - f_n(x)|^2\,dx = 0.$$

□

Ahora que ya hemos visto que el espacio es completo veamos los conceptos de sistemas ortonormales y bases en L^2 que serán fundamentales para la introducción de las wavelets.

Definición 1.8. Sea $\{\varphi_n\}_{n\in\mathbb{N}}$ un **sistema ortonormal**, es decir $< \varphi_n, \varphi_m >= \begin{cases} 0 & \text{si } n \neq m \\ 1 & \text{si } n = m \end{cases}$.
Además, si no existe otro sistema ortonormal que lo contenga se dice que es un **sistema ortonormal maximal** o una **base** .

Teorema 1.2. *Sea $\{\varphi_n\}_{n\in\mathbb{N}}$ un sistema ortonormal en el espacio $L^2(a,b)$, las siguientes afirmaciones son equivalentes.*

1) *$\{\varphi_n\}_{n\in\mathbb{N}}$ es una base.*

2) *Si existe una función $f \in L^2(a,b)$ tal que $< f, \varphi_n >= 0$ para todo $n \in \mathbb{N}$ entonces $f = 0$.*

3) *$\forall f \in L^2(a,b)$, $f = \sum_{n=1}^{\infty} < f, \varphi_n > \varphi_n$ en L^2.*

4) *$\forall f \in L^2(a,b)$, $||f||^2 = \sum_{n=1}^{\infty} < f, \varphi_n >^2$.*

Demostración. Para la demostración empezamos viendo que 1) implica 2).
Supongamos que existe $f \neq 0$ tal que $< f, \varphi_n >= 0\ \forall n \in \mathbb{N}$. Sea $\bar{f} = \frac{f}{||f||}$. Si consideramos $\{\varphi_n\}_{n\in\mathbb{N}} \bigcup \bar{f}$, este sistema es ortonormal y contiene al anterior, por lo tanto tenemos una contradicción ya que $\{\varphi_n\}_{n\in\mathbb{N}}$ es base.

Veamos ahora que 2) implica 3). Dada $f \in L^2(a,b)$, sea $S_m = \sum_{k=1}^{m} < f, \varphi_k > \varphi_k$, y llamamos $g = \lim_{m\to\infty} S_m$.
Basta entonces probar que $< f-g, \varphi_n >= 0$. $\forall n \in \mathbb{N}$.
Como

$$< S_m, \varphi_n >= \begin{cases} < f, \varphi_n > & si\ m \geq n \\ 0 & si\ m < n \end{cases},$$

entonces,

$$< f-g, \varphi_n >=< f - \lim_{m\to\infty} S_m, \varphi_n >=< f, \varphi_n > - \lim_{m\to\infty} < S_m, \varphi_n >=< f, \varphi_n > - < f, \varphi_n >= 0.$$

Seguimos con 3) implica 4). Sea $S_n = \sum_{k=1}^{n} < f, \varphi_k > \varphi_k$, por lo que $||f - S_n||^2 = ||f||^2 - \sum_{k=1}^{n} < f, \varphi_k >^2$.
Por otro lado, tenemos que, ya que S_n tiende a f, $\lim_{n\to\infty} ||f - S_n|| = 0$, y se tiene que $||f||^2 = \sum_{k=1}^{n} < f, \varphi_k >^2$.

Por último, vemos que 4) implica 1). Suponiendo que $\{\varphi_n\}_{n\in\mathbb{N}}$ no es base, existe un sistema ortonormal S que contiene estrictamente a $\{\varphi_n\}_{n\in\mathbb{N}}$. Sea $f \in S$ tal que $f \neq \varphi_n\ \forall n \in \mathbb{N}$, $||f||^2 = \sum_{n=1}^{\infty} < f, \varphi_n >^2 = 0$.

□

1.2. Transformada de Fourier

El contenido de esta sección esta basado en [[1] capítulo 6].
Consideramos el siguiente sistema ortonormal,

$$\left\{ \frac{1}{\sqrt{2\pi}}, \frac{\cos(x)}{\sqrt{\pi}}, \frac{\sin(x)}{\sqrt{\pi}}, \frac{\cos(2x)}{\sqrt{\pi}}, \frac{\sin(2x)}{\sqrt{\pi}}, \dots \right\} = \left\{ \frac{1}{\sqrt{2\pi}}, \frac{\cos(nx)}{\sqrt{\pi}}, \frac{\sin(nx)}{\sqrt{\pi}} \right\}_{n\in\mathbb{N}}$$

que es base de $L^2(-\pi,\pi)$. Por el Teorema 1.1, toda función $f \in L^2(-\pi,\pi)$ puede escribirse como

$$f = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx) \right),$$

con

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\ dx,$$
$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\ dx.$$

A la función escrita de esta manera la llamamos serie de Fourier. Es importante darse cuenta de que una función de esta forma es periódica e infinita. Más adelante introduciremos las wavelets, las cuales tienen diferentes propiedades, entre ellas que son localizadas, veremos esto en el siguiente capítulo con un ejemplo.

Definición 1.9. (Transformada de Fourier). Sea $f \in L^1(\mathbb{R})$, la transformada de Fourier de f es

$$\mathscr{F}[f](w) = \tilde{f}(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-iwt}\ dt.$$

Definición 1.10. (Transformada inversa de Fourier). Dada $\tilde{f}(w)$ la transformada de Fourier de f, definimos la transformación inversa como,

$$\mathscr{F}^{-1}[\tilde{f}(w)] = f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \tilde{f}(w) e^{iwt}\ dw.$$

Ejemplo 1. Veamos la utilidad de representar señales por series de Fourier. La función

$$f(x) = |x|, \tag{1.1}$$

escrita como serie de Fourier es

$$\begin{aligned} f(x) &\sim \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx)) \\ &= \frac{\pi}{2} + \sum_{n=1}^{\infty} \frac{2}{\pi n^2} ((-1)^n - 1)\cos(nx)). \end{aligned}$$

La reprentación con 5 términos viene dada en la Figura 1.1, dónde se observa una buena aproximación debido a que la extensión 2π periódica es continua.

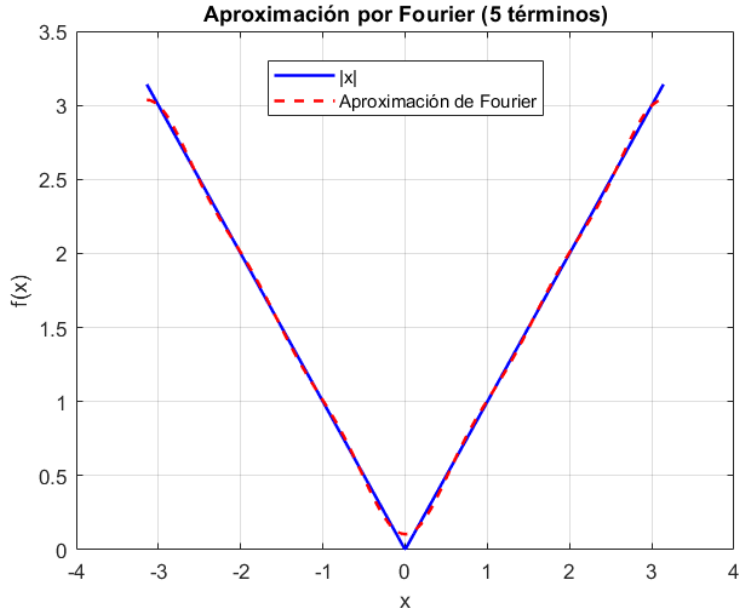

Figura 1.1: Aproximación por Fourier de (1.1).

Ejemplo 2. Veamos un ejemplo de serie de Fourier que motive la introducción de las wavelets. La función

$$f(x) = \begin{cases} \pi/2, & x \in (0,\pi), \\ 0, & x = 0, \\ -\pi/2 & x \in (-\pi,0), \end{cases} \tag{1.2}$$

es constante en dos tramos como vemos en la Figura 1.2 (ver trazado en color azul). Al escribirla como serie de Fourier tenemos que

$$f(x) = 2\sum_{n=1}^{\infty} \frac{\sin((2n-1)x)}{2n-1},$$

lo cual para valores pequeños de n no da una buena aproximación en torno al 0.

En ambas Figuras 1.1 y 1.2 vemos en azul la función y en rojo discontinuo la aproximación por serie de Fourier utilizando 5 y 20 términos de cada serie respectivamente. Como se puede observar la aproximación es mejor en el Ejemplo 1 ya que la función es continua y su extensión periódica también lo es. En el Ejemplo 2 tenemos el problema en torno al 0 y el conocido como fenomeno de Gibbs (mala aproximación en los extremos donde la función presenta una discontinuidad).

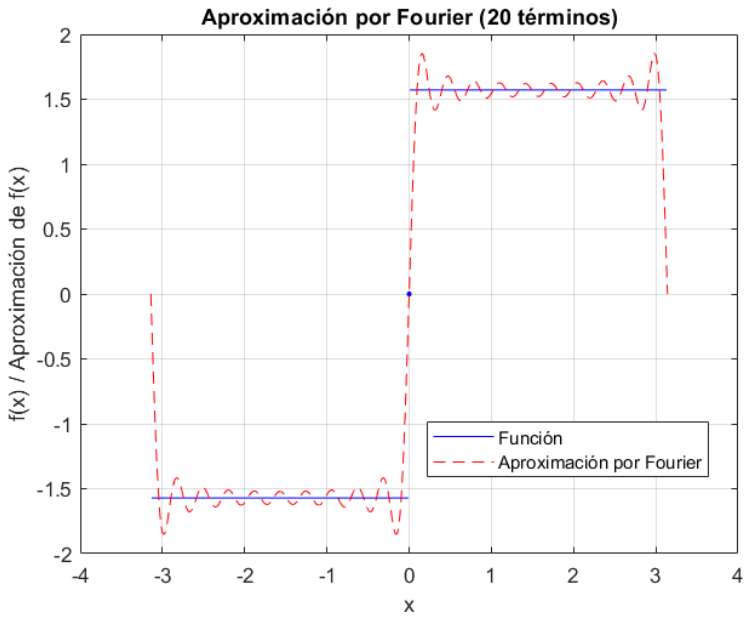

Figura 1.2: Aproximación por Fourier de (1.2).

Capítulo 2

Construcción de wavelets

A diferencia de la transformada de Fourier, que utiliza funciones periódicas para descomponer señales, la transformada wavelet es especialmente útil para analizar funciones no periódicas y transitorias. En lugar de la periodicidad, las wavelets emplean funciones localizadas tanto en el tiempo como en la frecuencia, permitiendo una mejor representación del comportamiento local de las señales. Este enfoque se conoce como transformación en ventana. Nuestro objetivo es encontrar una base ortonormal alternativa que supere las limitaciones de Fourier en el análisis local de las funciones. Después de introducir el concepto de wavelets y sus propiedades fundamentales, volveremos a examinar el ejemplo presentado en las Figura 1.1 y 1.2 para ilustrar la mejora en la representación de señales no periódicas.

Definición 2.1. Llamaremos **wavelet** a una función $\psi \in L^2(\mathbb{R})$ si la famila de funciones

$$\psi_{j,k} = 2^{j/2}\psi(s^j x - k) \quad \forall j,k \in \mathbb{Z} \tag{2.1}$$

forma una base ortonormal de $L^2(\mathbb{R})$. Entonces dicha famila de funciones se llama **base wavelet**.

2.1. Wavelet de Haar

Cualquier función de L^2 puede ser aproximada con funciones indicadoras,

$$\chi_{n,k}(x) = \begin{cases} 1, & 2^{-n}k \leq x < 2^{-n}(k+1), \\ 0, & \text{en otro caso.} \end{cases}$$

Por lo tanto, para toda función $f \in L^2$ existen funciones indicadoras tales que

$$f_n(x) = \sum_{k\in\mathbb{Z}} c_{n,k}\chi_{n,k}(x), \;\; n \in \mathbb{N},$$

de manera que,

$$\lim_{n\to\infty} ||f_n - f|| = 0.$$

Definimos

$$V_n = \left\{ g_n \middle| g_n = \sum_{k\in\mathbb{Z}} a_{n,k}\chi_{n,k},\ (a_{n,k})_{k\in\mathbb{Z}} \in l^2 \right\}.$$

Entonces, $\{V_n\}$ es una sucesión de subespacios de L^2 que aproxima L^2 en sentido que para toda función $f \in L^2$ hay funciones $f_n \in V_n$ tales que

$$\lim_{n\to\infty} ||f_n - f|| = 0.$$

Llamamos a $\{V_n\}$ una aproximación de L^2. Es evidente que conforme más grande es la n el espacio V_n tiene una mejor resolución. También es claro que

$$\overline{\bigcup_{n\in\mathbb{Z}} V_n} = L^2, \tag{2.2}$$

y

$$\bigcap_{n\in\mathbb{Z}} V_n = \emptyset. \tag{2.3}$$

Para construir la base del espacio V_n, tomamos una base ortonormal de V_0. Sea

$$B(x) = \begin{cases} 1, & 0 \le x < 1, \\ 0, & \text{en otro caso,} \end{cases} \tag{2.4}$$

que es la función característica del intervalo $[0,1)$. Es claro que el conjunto de funciones $\{B(x-k)\}_{k\in\mathbb{Z}}$ forma una base ortonormal de V_0. Por lo tanto, una base ortonormal de V_n puede ser construida ampliando el sistema de V_0. Para una función $f \in L^2$, se define

$$f_{n,k}(x) = 2^{n/2} f(2^n x - k),\ k \in \mathbb{Z},\ n \in \mathbb{Z}. \tag{2.5}$$

Entonces, el sistema de funciones $\{B_{n,k}\}_{k\in\mathbb{Z}}$, donde $B_{n,k}$ se define como en (2.5) a partir de B en (2.4), es una base ortonormal para V_n.
Apoyándonos en lo anterior, podemos construir una base ortonormal para L^2. Sea W_n el complemento ortogonal de V_n con respecto a V_{n+1}, es decir

$$W_n \oplus V_n = V_{n+1},\ \ W_n \perp V_n.$$

Por lo descrito anteriormente en (2.2) y (2.3), se tiene que,

$$L^2 = \bigoplus_{n\in\mathbb{Z}} W_n,\ \ W_n \perp W_{n'}, n \neq n'. \tag{2.6}$$

Como los espacios estan construidos a base de 2^n-dilataciones basta con encontar una base ortonormal para W_0.Para ello, definimos la función de Haar, introducida por el matemático alemán Haar (1909), y dada por

$$H(x) = \begin{cases} 1, & 0 \le x < \frac{1}{2}, \\ -1, & \frac{1}{2} \le x < 1, \\ 0, & \text{en otro caso.} \end{cases} \tag{2.7}$$

Podemos usar las funciones correspondientes a la base wavelet definida por $\{H_{n,k}\}_{k\in\mathbb{Z}}$, donde $H_{n,k}$ se define como en (2.5) a partir de H en (2.7) para descomponer cualquier función como una serie de funciones de Haar. Veamos primero que la familia de funciones $\{H_{n,k}(x)\}_{n,k\in\mathbb{N}}$, definida como en (2.5) es una base del espacio L^2.

Teorema 2.1. *Sea $H_k(x) = H(x-k), k \in \mathbb{Z}$. Entonces H_k es base ortonormal de W_0. En consecuencia $H_{n,k}$, definida como en* (2.5) *a partir de H en* (2.7)*, es base ortonormal del espacio W_n.*

Demostración. Tenemos que probar que H_k forma una base ortonormal de W_0. Es claro que $\{H_k(x)\}_{k\in\mathbb{Z}}$ es un sistema ortonormal de W_0. Veamos que también es una base. Sea g una función de W_0, entonces $g \in V_1$ y existe una sucesión $(c_k) \in l^2$ de manera que

$$g = \sum_{k\in\mathbb{Z}} c_k B_{1,k} = \sum_{l\in\mathbb{Z}} (c_{2l} B_{1,2l} + c_{2l+1} B_{1,2l+1}).$$

Como $g \perp V_0$, tenemos que $c_{2l+1} = -c_{2l}$. Véase que $H_l = \frac{1}{\sqrt{2}}(B_{1,2l} - B_{1,2l+1})$. Entonces,

$$g = \sqrt{2} \sum_{l\in\mathbb{Z}} c_{2l} H_l,\ \ (c_{2l}) \in l^2.$$

Por lo tanto tenemos ya que el sistema $\{H_{n,k}(x)\}_{n,k\in\mathbb{Z}}$ forma una base ortonormal de L^2. □

Visto lo anterior tenemos que cualquier función de L^2 puede ser expresada como serie de funciones de Haar de la forma que sigue,

$$f = \sum_{n,k\in\mathbb{Z}} d_{k,n} H_{n,k},$$

donde $d_{k,n} = < f, H_{n,k} >$.

Siguiendo con el desarrollo, debido a que $V_j = \bigoplus_{k<j} W_k$ y que V_j se reduce al espacio trivial cuando j tiende a menos infinito, tenemos que f puede ser representada como

$$f = \sum_{k\in\mathbb{Z}} c_{j,k} B_{j,k} + \sum_{n\geq j} \sum_{k\in\mathbb{Z}} d_{n,k} H_{n,k}.$$

Sin perdida de generalidad, se puede tomar $j = 0$ y definir la serie de Haar truncada como

$$f_N = \sum_{k\in\mathbb{Z}} c_k B_k + \sum_{j=0}^{N-1} \sum_{k\in\mathbb{Z}} d_{j,k} H_{j,k}. \tag{2.8}$$

La forma de calcular los coeficientes es lo que veremos en la Sección 2.4 y es conocido como el algoritmo de la trasformada rápida.

Veamos ahora los ejemplos propuestos en el capítulo anterior aproximados con funciones Haar.

Ejemplo 3. La función $f(x) = |x|$ aproximada por la base wavelet de Haar usando 20 términos viene dada por la Figura 2.1.

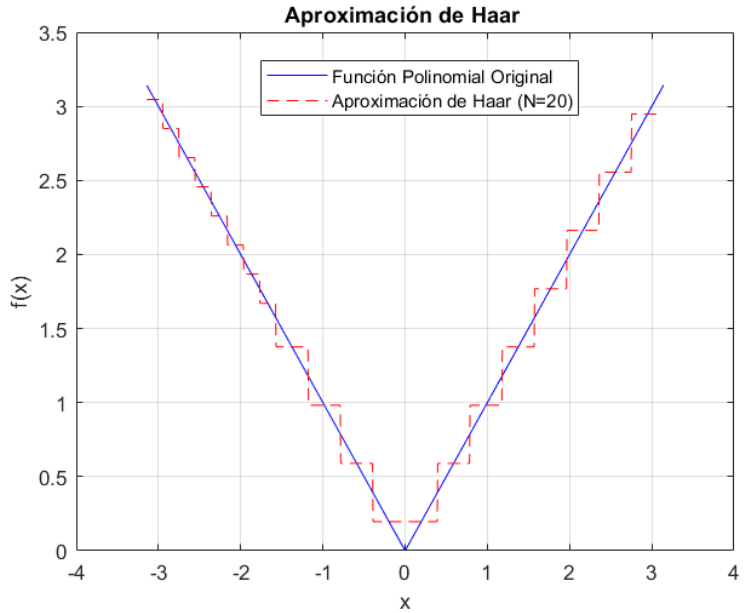

Figura 2.1: Aproximación por Haar de (1.1).

Ejemplo 4. La función

$$f(x) = \begin{cases} \pi/2, & x \in (0,\pi), \\ 0, & x = 0, \\ -\pi/2 & x \in (-\pi,0), \end{cases}$$

aproximada por la base wavelet de Haar queda representada en la Figura 2.2

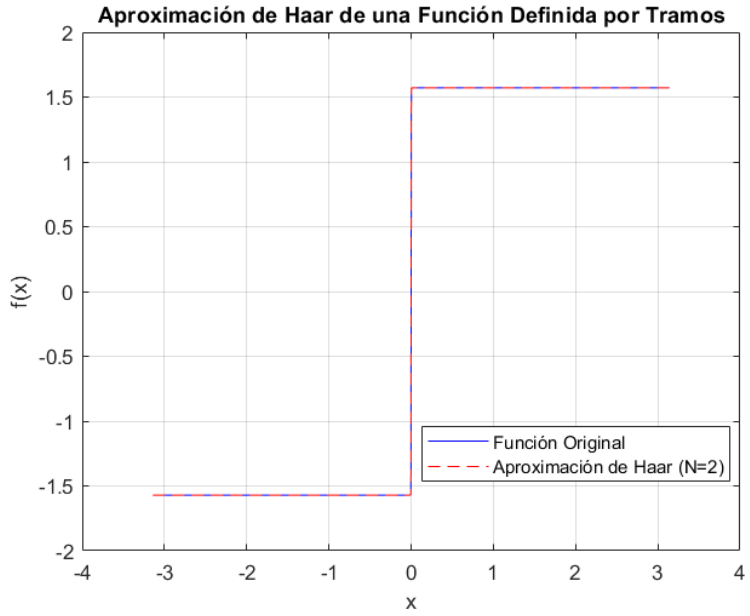

Figura 2.2: Aproximación por Haar de (1.2).

Como podemos observar, dependiendo de las características de cada función, será más conveniente usar una base de funciones u otra. En general, veremos que para las aplicaciones en imágenes y señales, será mejor utilizar las wavelets de Haar u otros tipos que manejen adecuadamente los cambios locales de las funciones.

2.2. Transformada discreta y transformada rápida

Definición 2.2. Una **señal discreta** es una función que se define solo en valores discretos de la variable. Matemáticamente, una señal discreta $x[n]$ se define como:

$$x : \mathbb{N} \to \mathbb{R},$$

donde $n \in \mathbb{N}$ es la variable discreta . Una señal discreta, de longitud N se puede representar como un vector en el espacio $\mathbb{R}^N$:

$$x = \{x[0], x[1], x[2], \ldots, x[N-1]\}.$$

A nivel de aplicaciones de la transformada wavelet, como la eliminación de ruido y la compresión, es necesaria la transformada discreta (DWT por sus siglas en inglés), la cual descompone una señal discreta en componentes de diferente frecuencia y resolución lo cual es útil para analizar características locales de una señal. Antes de pasar con el algoritmo de la transformada, primero es necesario presentar lo que llamamos filtros.

Definición 2.3. Un **filtro** es una sucesión que transforma una señal de entrada en una señal de salida mediante la atenuación, amplificación o modificación de ciertas componentes de frecuencia de la señal original. Los filtros pueden ser utilizados para eliminar el ruido, resaltar ciertas características de la señal, o separar distintas componentes de la señal. En términos matemáticos, la operación de un filtro se describe mediante la **convolución**. Para señales discretas, la salida $y[n]$ con entrada $x[n]$ y filtro $f[n]$ de longitud K se define como:

$$y[n] = (x * f)[n] = \sum_{k=0}^{K-1} x[k] f[n-k]. \tag{2.9}$$

Los tipos de filtros que utilizaremos en las siguientes secciones son los siguientes.
Un **filtro pasa-bajos** (LPF por sus siglas en inglés) permite el paso de frecuencias bajas y atenúa las frecuencias altas. En el contexto de la DWT este filtro extrae las aproximaciones de la señal,es decir, la parte suave o de baja frecuencia que contiene la información más global de la señal.
Un **filtro pasa-altos** (HPF por sus siglas en inglés) permite el paso de las frecuencias altas y atenúa las frecuencias bajas. Este filtro extrae los detalles de la señal, capturando las variaciones rápidas o las componentes de alta frecuencia que contienen información sobre los cambios locales en la señal.
Los filtros no son más que una succesión de números, distinta para cada wavelet en particular, la cual se utiliza para hacer una convolución con la señal, más adelante veremos un ejemplo.

2.3. Análisis de multirresolución (MRA)

Definición 2.4. Un **análisis de multirresolución** (MRA por sus siglas en inglés) de L^2 es un anidamiento de subespacios de L^2

$$\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots$$

que cumple lo siguiente.

(1) $\cap_{j\in\mathbb{Z}} V_j = \{0\}$,

(2) $\cup_{j\in\mathbb{Z}} V_j = L^2$,

(3) $f(\cdot) \in V_j$ si y solo si $f(2\cdot) \in V_{j+1}$,

(4) existe una función (**función escala**) $\phi \in V_0$ de forma que $\{\phi(x-n)\}_{n\in\mathbb{Z}}$ es una base de V_0, y existen dos constantes A,B > 0 de forma que, para todo $(c_n) \in l^2$ se cumple la siguiente inecuación,

$$A \sum_{n\in\mathbb{N}} |c_n|^2 \leq \left|\left| \sum_{n\in\mathbb{N}} c_n \phi(x-n) \right|\right|^2 \leq B \sum_{n\in\mathbb{N}} |c_n|^2.$$

Las propiedades anteriores aseguran que las funciones pueden ser representadas a diferentes niveles de resolución. Así, las funciones pueden ser descompuestas en diferentes resoluciones y luego recombinadas para recuperar la función original.

El análisis de multirresolución (MRA) es una herramienta fundamental en el procesamiento de señales mediante wavelets. Permite descomponer una señal en componentes de diferentes resoluciones, lo que facilita la extracción de características y el análisis de detalles a distintas escalas. Este proceso de descomposición es esencial para aplicaciones en compresión de datos, eliminación de ruido y análisis de frecuencias.
Partiendo de una señal discreta en el contexto de MRA, se utilizan wavelets ortogonales que permiten descomponer y reconstruir una señal sin pérdida de información. El proceso de descomposición se realiza aplicando filtros de paso bajo y de paso alto a la señal original para obtener aproximaciones y detalles. Este proceso se repite sobre las aproximaciones obtenidas en cada nivel para obtener las aproximaciones y detalles del siguiente nivel. Veamos como se aplican los filtros, que no son más que una señal $l[n]$ para los LPF y $h[n]$ para los HPF, ambas de longitud K. La señal $x[n]$ es de longitud N

- **Aproximaciones** (LPF)

$$A_1[n] = \sum_{k=-\infty}^{\infty} x[k] l[n-k], \tag{2.10}$$

- **Detalles** (HPF)

$$D_1[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]. \tag{2.11}$$

Además de los filtros luego hay que aplicar el submuestreo,es decir, tomar uno de cada dos valores para ir reduciendo el tamaño en cada nivel. Lo cual es lo mismo que utilizar la siguiente fórmula,

$$y[n] = \sum_{k=-\infty}^{\infty} x[2n+k]f[k], \tag{2.12}$$

donde y es la señal de salida y f el filtro correspondiente El resultado de esta primera descomposición son las aproximaciones A_1 y los detalles D_1. Las aproximaciones A_1 se vuelven a descomponer utilizando los mismos filtros para obtener A_2 y D_2. Este proceso se repite hasta alcanzar el nivel deseado J, obteniendo A_J y D_J.

El proceso inverso se utiliza para reconstruir la señal original a partir de las componentes de aproximación y detalle. Para reconstruir la señal en el nivel $j-1$,

$$A_{j-1}[n] = \sum_{k=-\infty}^{\infty} \left(A_j^{\uparrow}[k] \cdot l[n-k] + D_j^{\uparrow}[k] \cdot h[n-k] \right), \tag{2.13}$$

donde $A_j^{\uparrow}$ significa la señal A_j sobremuestreada, explicaremos este término más adelante en un ejemplo. Si el índice quedara fuera de la longitud del filtro o la señal este sería 0. Este proceso se repite hasta reconstruir la señal original a partir de A_J y todos los D_j (donde $j = 1,2,\ldots,J$).
Como es evidente, el MRA no es más que aplicar el algoritmo de DWT sucesivamente primero sobre la señal y posteriormente sobre las aproximaciones de cada nivel hasta llegar al nivel deseado.

2.3.1. Ejemplo

Para entender el algoritmo de la DWT y el MRA, veamos un ejemplo sencillo de una señal basado en la wavelet de Haar, puede verse el códido en el Apéndice A.3. Consideremos una señal discreta $x[n]$ de longitud 8:

$$x = \{4,6,10,12,14,16,18,20\}.$$

Los filtros utilizados en la transformada de Haar son los siguientes:

1. **Filtro Pasa-Bajos (LPF)** l:

$$l = \left\{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\}.$$

2. **Filtro Pasa-Altos (HPF)** h:

$$h = \left\{ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right\}.$$

Para cada par de muestras en $x[n]$, aplicamos los filtros como en (2.12) con sus filtros correspondientes.

Aproximaciones (LPF):

$$A_1[0] = \sum_{k=-\infty}^{\infty} (x[2\cdot 0+k]l[k]) = x[0]l[0]+x[1]l[1] = 4\cdot\frac{1}{\sqrt{2}} + 6\cdot\frac{1}{\sqrt{2}} = \frac{10}{\sqrt{2}} = 5\sqrt{2},$$

$$A_1[1] = \sum_{k=-\infty}^{\infty} (x[2\cdot 1+k]l[k]) = x[2]l[0]+x[3]l[1] = \frac{1}{\sqrt{2}}(10+12) = \frac{22}{\sqrt{2}} = 11\sqrt{2},$$

$$A_1[2] = \frac{1}{\sqrt{2}}(14+16) = \frac{30}{\sqrt{2}} = 15\sqrt{2},$$

$$A_1[3] = \frac{1}{\sqrt{2}}(18+20) = \frac{38}{\sqrt{2}} = 19\sqrt{2}.$$

Detalles (HPF):

$$D_1[0] = \sum_{k=0}^{1} (x[2\cdot 0+k]h[k]) = x[0]h[0]+x[1]h[1] = 4\cdot\frac{1}{\sqrt{2}} - 6\cdot\frac{1}{\sqrt{2}} = \frac{-2}{\sqrt{2}} = -\sqrt{2},$$
$$D_1[1] = \frac{1}{\sqrt{2}}(10-12) = \frac{-2}{\sqrt{2}} = -\sqrt{2},$$
$$D_1[2] = \frac{1}{\sqrt{2}}(14-16) = \frac{-2}{\sqrt{2}} = -\sqrt{2},$$
$$D_1[3] = \frac{1}{\sqrt{2}}(18-20) = \frac{-2}{\sqrt{2}} = -\sqrt{2}.$$

A_1 y D_1 son las aproximaciones y detalles de primer nivel o los coeficientes de la transformada wavelet. Con ellos podemos seguir aplicando el mismo algoritmo para obtener los siguientes niveles, lo que sería el MRA.
Por lo tanto, aplicamos de nuevo los filtros a las aproximaciones A_1 obtenidas en el primer paso.

Aproximaciones del Segundo Nivel (LPF):

$$A_2[0] = \sum_{k=0}^{1} (A_1[2\cdot 0+k]l[k]) = A_1[0]l[0]+A_1[1]l[1] = 5\sqrt{2}\cdot\frac{1}{\sqrt{2}} + 11\sqrt{2}\cdot\frac{1}{\sqrt{2}} = \frac{16\sqrt{2}}{\sqrt{2}} = 16,$$
$$A_2[1] = \frac{1}{\sqrt{2}}(15\sqrt{2}+19\sqrt{2}) = \frac{34\sqrt{2}}{\sqrt{2}} = 34.$$

Detalles del Segundo Nivel (HPF):

$$D_2[0] = \sum_{k=0}^{1} (A_1[2\cdot 0+k]h[k]) = A_1[0]h[0]+A_1[1]h[1] = 5\sqrt{2}\cdot\frac{1}{\sqrt{2}} - 11\sqrt{2}\cdot\frac{1}{\sqrt{2}} = \frac{-6\sqrt{2}}{\sqrt{2}} = -6,,$$
$$D_2[1] = \frac{1}{\sqrt{2}}(15\sqrt{2}-19\sqrt{2}) = \frac{-4\sqrt{2}}{\sqrt{2}} = -4.$$

Aplicamos nuevamente los filtros a las aproximaciones A_2.

Aproximaciones del Tercer Nivel (LPF):

$$A_3[0] = \sum_{k=0}^{1} (A_2[2\cdot 0+k]l[k]) = A_2[0]l[0]+A_2[1]l[1] = 16\cdot\frac{1}{\sqrt{2}} + 34\cdot\frac{1}{\sqrt{2}} = \frac{50}{\sqrt{2}} = 25\sqrt{2}.$$

Detalles del Tercer Nivel (HPF):

$$D_3[0] = \sum_{k=0}^{1} (A_2[2\cdot 0+k]h[k]) = A_2[0]h[0]+A_2[1]h[1] = 16\cdot\frac{1}{\sqrt{2}} - 34\cdot\frac{1}{\sqrt{2}} = \frac{-18}{\sqrt{2}} = -9\sqrt{2}.$$

Hemos descompuesto la señal original $x[n]$ en sus componentes de aproximación y detalles a varios niveles, obteniendo lo siguiente:

- **Aproximaciones del Tercer Nivel** A_3= $\{25\sqrt{2}\}$,
- **Detalles del Tercer Nivel** D_3= $\{-9\sqrt{2}\}$,
- **Detalles del Segundo Nivel** D_2= $\{-6,-4\}$,
- **Detalles del Primer Nivel** D_1= $\{-\sqrt{2},-\sqrt{2},-\sqrt{2},-\sqrt{2}\}$.

Ahora vamos a utilizar los filtros y el denominado sobremuestreo que explicaremos más adelante, para reconstruir las aproximaciones del nivel anterior. Por lo que empezamos tomando las aproximaciones y los detalles del nivel tres para obtener las aproximaciones del nivel dos.

Filtro Pasa-Bajos (LPF) l:

$$l = \left\{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\}.$$

Filtro Pasa-Altos (HPF) h:

$$h = \left\{ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right\}.$$

El sobremuestreo consiste en insertar tantos ceros como posiciones en los detalles y aproximaciones de manera intercalada. Veamos el primer paso con detalle. Insertamos un cero en A_3 y en D_3 resultando en $A_3^{\uparrow} = (25\sqrt{2}, 0)$ y $D_3^{\uparrow} = (-9\sqrt{2}, 0)$. Ahora aplicamos los filtros correspondiente a detalles y aproximaciones de nivel tres y sumamos para obtener las aproximaciones de nivel dos, ver (2.13).

Aproximaciones del Segundo Nivel (LPF):

$$\begin{aligned}
A_2[0] &= \sum_{k=-\infty}^{\infty} \left(A_3^{\uparrow}[k] \cdot l[0-k] + D_3^{\uparrow}[k] \cdot h[0-k] \right) = A_3^{\uparrow}[0]h[0] + D_3^{\uparrow}[0]h[0] + A_3^{\uparrow}[-1]l[1] + D_3^{\uparrow}[-1]h[1] = \\
&= 25\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-9\sqrt{2}) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = 25 - 9 = 16, \\
A_2[1] &= \sum_{k=-\infty}^{\infty} \left(A_3^{\uparrow}[k] \cdot l[1-k] + D_3^{\uparrow}[k] \cdot h[1-k] \right) = A_3^{\uparrow}[0]l[1] + D_3^{\uparrow}[0]h[1] + A_3^{\uparrow}[1]l[0] + D_3^{\uparrow}[1]h[0] = \\
&= 25\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-9\sqrt{2}) \cdot \frac{-1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} = 25 + 9 = 34.
\end{aligned}$$

La variable del sumatorio k recorre todos los valores pero donde las señales no toman valor se toma por cero, por lo tanto, solo hay dos sumandos, ya que los filtros tienen longitud dos. Reconstruimos las aproximaciones del primer nivel a partir de las del segundo nivel. Primero hacemos el sobremuestreo y obtenemos $A_2^{\uparrow} = (16, 0, 34, 0)$ y $D_2^{\uparrow} = (-6, 0, -4, 0)$.

Aproximaciones del Primer Nivel (LPF):

$$\begin{aligned}
A_1[0] &= \sum_{k=-\infty}^{\infty} \left(A_2^{\uparrow}[k] \cdot l[0-k] + D_2^{\uparrow}[k] \cdot h[0-k] \right) = A_2^{\uparrow}[0]l[0] + D_2^{\uparrow}[0]h[0] + A_2^{\uparrow}[-1]l[1] + D_2^{\uparrow}[-1]h[1] = \\
&= 16 \cdot \frac{1}{\sqrt{2}} + (-6) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = \frac{10}{\sqrt{2}} = 5\sqrt{2}, \\
A_1[1] &= A_2^{\uparrow}[1]l[0] + D_2^{\uparrow}[1]h[0] + A_2^{\uparrow}[0]l[1] + D_2^{\uparrow}[0]h[1] = \\
&= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 16 \cdot \frac{1}{\sqrt{2}} + (-6) \cdot \frac{-1}{\sqrt{2}} = \frac{22}{\sqrt{2}} = 11\sqrt{2}, \\
A_1[2] &= A_2^{\uparrow}[2]l[0] + D_2^{\uparrow}[2]h[0] + A_2^{\uparrow}[1]l[1] + D_2^{\uparrow}[1]h[1] = \\
&= 34 \cdot \frac{1}{\sqrt{2}} + (-4) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = \frac{30}{\sqrt{2}} = 15\sqrt{2}, \\
A_1[3] &= A_2^{\uparrow}[3]l[0] + D_2^{\uparrow}[3]h[0] + A_2^{\uparrow}[2]l[1] + D_2^{\uparrow}[2]h[1] = \\
&= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 34 \cdot \frac{1}{\sqrt{2}} + (-4) \cdot \frac{-1}{\sqrt{2}} = \frac{38}{\sqrt{2}} = 19\sqrt{2}.
\end{aligned}$$

Reconstruimos la señal original a partir de las aproximaciones del primer nivel. Primero hacemos el sobremuestreo y obtenemos $A_1^{\uparrow} = (5\sqrt{2}, 0, 11\sqrt{2}, 0, 15\sqrt{2}, 0, 19\sqrt{2}, 0)$ y $D_1^{\uparrow} = (-\sqrt{2}, 0, -\sqrt{2}, 0, -\sqrt{2}, 0, -\sqrt{2}, 0)$.

Señal Original Reconstruida (LPF):

$$x[0] = \sum_{k=-\infty}^{\infty} \left(A_1^{\uparrow}[k] \cdot l[0-k] + D_1^{\uparrow}[k] \cdot h[0-k] \right) = A_1^{\uparrow}[0]l[0] + D_2^{\uparrow}[0]h[0] + A_1^{\uparrow}[-1]l[1] + D_1^{\uparrow}[-1]h[1] =$$
$$= 5\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = 5 - 1 = 4,$$
$$x[1] = A_1^{\uparrow}[1]l[0] + D_2^{\uparrow}[1]h[0] + A_1^{\uparrow}[0]l[1] + D_1^{\uparrow}[0]h[1] =$$
$$= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 5\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{-1}{\sqrt{2}} = 5 + 1 = 6,$$
$$x[2] = A_1^{\uparrow}[2]l[0] + D_2^{\uparrow}[2]h[0] + A_1^{\uparrow}[1]l[1] + D_1^{\uparrow}[1]h[1] =$$
$$= 11\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = 11 - 1 = 10,$$
$$x[3] = A_1^{\uparrow}[3]l[0] + D_1^{\uparrow}[3]h[0] + A_1^{\uparrow}[2]l[1] + D_1^{\uparrow}[2]h[1]$$
$$= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 11\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{-1}{\sqrt{2}} = 11 + 1 = 12,$$
$$x[4] = A_1^{\uparrow}[4]l[0] + D_1^{\uparrow}[4]h[0] + A_1^{\uparrow}[3]l[1] + D_1^{\uparrow}[3]h[1]$$
$$= 15\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = 15 - 1 = 14,$$
$$x[5] = A_1^{\uparrow}[5]l[0] + D_1^{\uparrow}[5]h[0] + A_1^{\uparrow}[4]l[1] + D_1^{\uparrow}[4]h[1]$$
$$= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 15\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{-1}{\sqrt{2}} = 15 + 1 = 16,$$
$$x[6] = A_1^{\uparrow}[6]l[0] + D_1^{\uparrow}[6]h[0] + A_1^{\uparrow}[5]l[1] + D_1^{\uparrow}[5]h[1]$$
$$= 19\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{-1}{\sqrt{2}} = 19 - 1 = 18,$$
$$x[7] = A_1^{\uparrow}[7]l[0] + D_1^{\uparrow}[7]h[0] + A_1^{\uparrow}[6]l[1] + D_1^{\uparrow}[6]h[1]$$
$$= 0 \cdot \frac{1}{\sqrt{2}} + 0 \cdot \frac{1}{\sqrt{2}} + 19\sqrt{2} \cdot \frac{1}{\sqrt{2}} + (-\sqrt{2}) \cdot \frac{-1}{\sqrt{2}} = 19 + 1 = 20.$$

- **Señal Original Reconstruida**:

$$x = \{4, 6, 10, 12, 14, 16, 18, 20\}.$$

En el siguiente capítulo veremos como podemos aplicar lo anterior en el tratamiento de imágenes.

2.4. Wavelets de Daubechies

Para lo descrito en esta sección se utiliza [[1], capítulo 8 sección 2 y [2] capítulo 10]

Las wavelets de Daubechies se construyen a partir de una secuencia de coeficientes, conocidos como coeficientes de filtro, que satisfacen ciertas propiedades. El proceso involucra la construcción de la **función de escala** $\phi(t)$ y la **función wavelet** $\psi(t)$.

La función de escala $\phi_L(t)$ de orden L, satisface la siguiente relación de escala,

$$\phi_L(t) = \sqrt{2} \sum_{k=0}^{2L-1} l[k] \phi_L(2t - k),$$

donde l_k son los coeficientes de escala y L es el orden de Daubechies. Estos coeficientes deben cumplir ciertas condiciones para garantizar la ortogonalidad y compacidad de la wavelet. Son los que usamos como filtros cuando aplicamos la DWT o MRA.

Los coeficientes de escala $l[k]$ deben satisfacer las siguientes condiciones de ortogonalidad,

$$\sum_{k=0}^{L-1} l[k]l[k+2m] = \delta_{m,0}$$

donde $\delta_{m,0}$ es el delta de Kronecker, que es 1 si $m = 0$ y 0 en caso contrario. Esto asegura que las funciones $\phi_L(t-k)$ sean ortogonales entre sí.

La función wavelet $\psi(t)$ se define en términos de la función de escala $\phi(t)$ y los coeficientes $h[k]$, que están relacionados con l_k como sigue,

$$h[k] = (-1)^k l[2L-1-k]. \tag{2.14}$$

La función wavelet $\psi(t)$ se expresa como,

$$\psi_L(t) = \sqrt{2}\sum_{k=0}^{L-1} h[k]\phi_L(2t-k).$$

Esta función es la conocida como **wavelet de Daubechies** de orden $L \geq 2$.

Para ilustrar con un ejemplo concreto, consideremos la wavelet de Daubechies de orden $L = 2$, denotemosla a partir de ahora por Db2, que tiene 4 coeficientes de filtro .
Los coeficientes $l[k]$ para Db2 son,

$$l[0] = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad l[1] = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad l[2] = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad l[3] = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

Los coeficientes $h[k]$ se calculan como se ha descrito en (2.14), es decir,

$$h[0] = l[3] = \frac{1-\sqrt{3}}{4\sqrt{2}}, \quad h[1] = -l[2] = -\frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h[2] = l[1] = \frac{3+\sqrt{3}}{4\sqrt{2}},$$

$$h[3] = -l[0] = -\frac{1+\sqrt{3}}{4\sqrt{2}}.$$

En la práctica, la transformada wavelet se realiza de manera discreta. La señal $f(t)$ se descompone en una serie de coeficientes de detalle y aproximación usando las funciones de escala y wavelet.
La señal $f(t)$ puede ser representada como,

$$f(t) = \sum_{k=0}^{L-1} a_k \phi_L(t-k) + \sum_{j=0}^{J}\sum_{k=0}^{L-1} d_{j,k}\psi_L(2^j t-k),$$

donde a_k son los coeficientes de aproximación y $d_{j,k}$ son los coeficientes de detalle en diferentes niveles de resolución j. Como puede observarse, es una generalización de la ecuación en (2.8) ya que la wavelet de Haar es la de Daubechies de grado 1.

Para la aplicación como en el Ejemplo 2.3.1 se usará el mismo método de DWT aplicando ahora los nuevos filtros l y h.

- **Descomposición**: La señal se descompone aplicando los filtros de escala y wavelet en diferentes niveles. Esto produce una serie de coeficientes que representan la señal en distintas resoluciones.
- **Reconstrucción**: La señal original se puede reconstruir a partir de estos coeficientes utilizando la transformada inversa de wavelet, que implica aplicar los filtros inversos correspondientes.

Para la descomposición y reconstrucción de la señal utilizamos el algoritmo de la DWT aplicando los filtros correspondientes a la wavelet que estemos utilizando en cada caso, para este caso DbL. A continuación, vemos algunos de los coeficientes para diferentes órdenes de Daubechies, ver Cuadro 2.1.

A continuación, en la Figura 2.3 vemos las representaciones de las funciones escala y wavelet de Daubechies de órdenes 1 a 4, calculadas en MATLAB, ver Apéndice A.2. Cabe resaltar, como ya se ha mencionado anteriormente, que para orden 1 la wavelet de Daubechies es la wavelet de Haar.

Orden	Coeficientes del Filtro l_k
Db1(Haar)	$l[0] = \frac{1}{\sqrt{2}}$ $l[1] = \frac{1}{\sqrt{2}}$
Db2	$l[0] = \frac{1+\sqrt{3}}{4\sqrt{2}}$ $l[1] = \frac{3+\sqrt{3}}{4\sqrt{2}}$ $l[2] = \frac{3-\sqrt{3}}{4\sqrt{2}}$ $l[3] = \frac{1-\sqrt{3}}{4\sqrt{2}}$
Db3	$l[0] \approx 0{,}3326705529500826$ $l[1] \approx 0{,}8068915093110928$ $l[2] \approx 0{,}4598775021184915$ $l[3] \approx -0{,}1350110200103908$ $l[4] \approx -0{,}0854412738820267$ $l[5] \approx 0{,}0352262918857095$
Db4	$l[0] \approx 0{,}2303778133088964$ $l[1] \approx 0{,}7148465705529154$ $l[2] \approx 0{,}6308807679298587$ $l[3] \approx -0{,}0279837694168599$ $l[4] \approx -0{,}1870348117188811$ $l[5] \approx 0{,}0308413818359869$ $l[6] \approx 0{,}0328830116666778$ $l[7] \approx -0{,}0105974017850690$

Cuadro 2.1: Valores de los coeficientes del filtro l_k para las wavelets de Daubechies de orden 1 a 4.

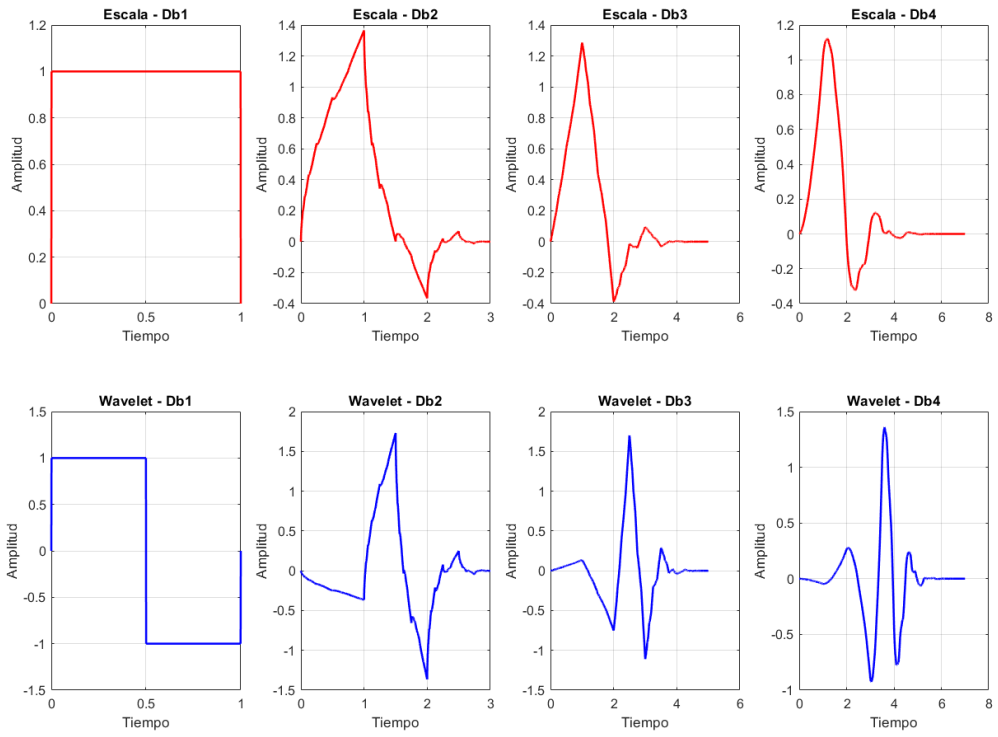

Figura 2.3: Wavelets de Daubechies de órdenes 1 a 4.

En resumen, las wavelets de Daubechies se construyen utilizando una serie de coeficientes de filtro que cumplen ciertas propiedades de ortogonalidad y compacidad.

2.5. Transformada continua

Definición 2.5. La **transformada wavelet continua** de una función $f(t)$ se define como

$$Wf(j,k) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t)\overline{\psi_{j,k}(t)}\,dt,$$

donde $\psi_{j,k}$ es una wavelet definida como en (2.1).

A veces se utilizan parámetros de reescala para que la función tenga media cero, por lo que la transformada queda definida por

$$Wf(s,u) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-u}{s}\right) dt.$$

Los parámetros también sirven para ajustar la anchura. Si s es mayor que 1, se realiza un análisis de baja frecuencia, y si está entre 0 y 1, se realiza un análisis de alta frecuencia. El parámetro u permite localizar temporalmente desplazando la wavelet a lo largo del tiempo.

Para que una función $\psi(t)$ sea considerada una **wavelet madre**, debe cumplir la **condición de admisibilidad**, que se expresa en términos de su transformada de Fourier $\hat{\psi}(\omega)$:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}\,d\omega < \infty.$$

Esta condición asegura que la energía de la wavelet madre esté bien distribuida en la frecuencia, permitiendo que la transformada wavelet continua sea invertible y la señal original pueda ser recuperada. La **wavelet madre** $\psi(t)$ es la función base a partir de la cual se generan todas las demás wavelets en la familia , como en (2.1), mediante escalamiento y traslación:

$$\psi_{s,u}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right),$$

donde s es el parámetro de escala y u es el parámetro de traslación.

Capítulo 3

Aplicaciones

En este capítulo, utilizaremos algunos de los métodos descritos en los capítulos anteriores para el tratamiento de imágenes. En particular, vamos a tratar la reducción de ruido y la compresión de imágenes. Los códigos que han sido programados en MATLAB para la realización de los experimentos de este capítulo pueden encontrarse en los apéndices.

3.1. Reducción de ruido

Para el contenido de esta sección utilizaremos algunos métodos descritos en [3]. Se utilizaran también funciones de la librería de MATLAB *Wavelet Toolbox*.

Empecemos describiendo qué es el ruido en una señal para poder entender como trabajan los siguientes métodos. Para la lectura de una imagen en MATLAB se asocian a cada pixel tres valores uno para cada componente de color de la escala RGB, estos valores estan comprendidos entre 0 y 255. Por lo tanto si la imagen tiene m píxeles de alto y n píxeles de ancho, ahora tenemos la información en una matriz de tamaño $m \times n \times 3$. La transformamos a escala de grises, si es necesario, para obtener una matriz de tamaño $m \times n$. Por último se dividen todos los valores entre 255 para obtener la matriz con valores de 0 a 1. El ruido lo añadimos sumando a la imagen (ahora una matriz) otra matriz aleatoria siguiendo una distribución normal (0,1) multiplicada por un factor ruido, en nuestro caso $30/255$. Se toma un factor de ruido de casi el 12 %, este podría ser otro siempre y cuando no sea ni excesivamente grande, lo que haría imposible el tratamiendo de ruido, ni muy pequeño ya que entonces apenas afectaría a la imagen.

$$\text{Imagen Ruidosa} = \text{Imagen} + \frac{30}{255} \cdot N(0,1). \tag{3.1}$$

A partir de lo descrito, los métodos para tratar el ruido siguen el mismo proceso:

1. **Descomposición**: Elegimos la wavelet a utilizar para la descomposición y el nivel de descomposición en el sentido de MRA para aplicar el algoritmo de DWT a la imagen ruidosa. Dicho algoritmo, de manera similar a lo detallado en el Ejemplo 2.3.1, devuelve como resultado una matriz de coeficientes de aproximación y tres con coeficientes de detalles, horizonales, verticales y diagonales, respectivamente.

2. **Umbralización**: Esta es la parte fundamental del proceso, donde tratamos los coeficientes proporcionados en el paso anterior para reducir el ruido. En general, lo que haremos es buscar un umbral, es decir un número, para eliminar en las matrices de los detalles los coeficientes que sean inferiores a dicho umbral. Existen diferentes métodos de umbralización así como muchas formas de calcular el umbral, lo que nos llevaría a realizar un estudio estadístico sobre los coeficientes, lo cual queda fuera del alcance de este trabajo.

3. **Recomposición**: Una vez hemos aplicado la umbralización tenemos los coeficientes wavelet actualizados y utilizando los mismos parámetros que para la descomposición y el algoritmo inverso de DWT obtenemos la imagen procesada.

Uno de los métodos para la eleción del umbral es el conocico como *Universal Threshold* (ver [3] Sección 3.5). El umbral se calcula de la siguiente forma,

$$\lambda = \sigma\sqrt{2\log(n)},$$

donde σ la calculamos como la mediana de todos los datos a los que les aplicamos el umbral y n es la longitud de la señal, en este caso la resolución de la imagen (tamaño de la matriz que estamos utilizando).

Para ver un ejemplo a color utilizamos el código en Apéndice A.4, donde estamos usando el algoritmo de la DWT de la librería de MATLAB *Wavelet Toolbox* para la descomposición wavelet en dos dimensiones. Así, obtenemos una matriz con los coeficientes de aproximación y tres con los coeficientes de detalles, horizontales, verticales y diagonales, respectivamente. Sobre las matrices de los detalles aplicamos la umbralización, y después con las matrices actualizadas reconstruímos la imagen, ver Figura 3.1. La wavelet utilizada en el proceso de descomposición y reconstrucción ha sido en este caso la wavelet de Daubechies de orden 2 , es decir, Db2.

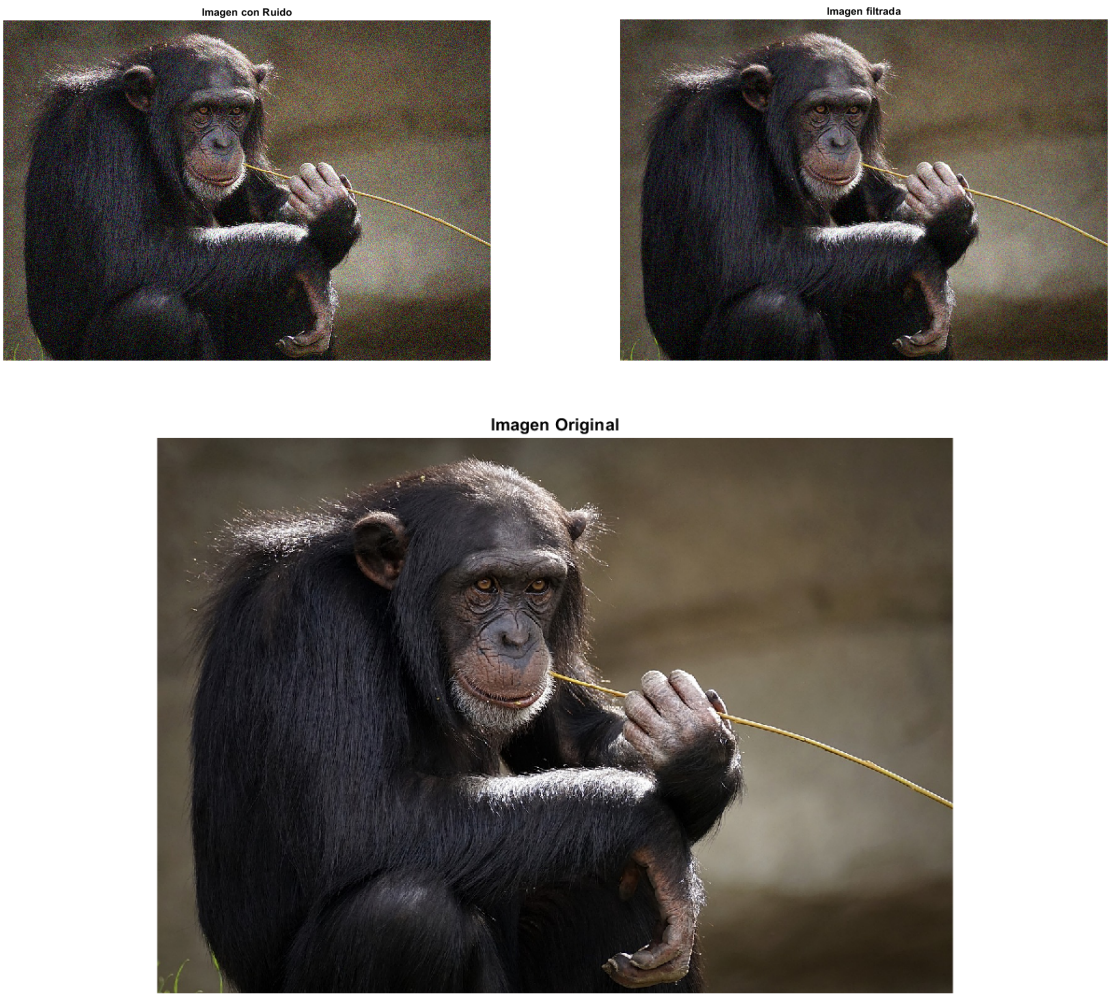

Figura 3.1: Filtrado en DWT, ver apéndice A.4.

Veamos ahora otro ejemplo utilizando el análisis de multirresolución, viendo las imágenes a varios niveles de resolución, y aplicando el mismo proceso de la umbralización universal en el último nivel de los detalles. Para este caso, será necesario utilizar imágenes en blanco y negro ya que la función utilizada en el código, ver Apéndice A.5, no trabaja con matrices de tres dimensiones como en el caso anterior.
Para este ejemplo, utilizaremos un factor de ruido de 30/255, el mismo que antes, la descomposición se

hará hasta nivel cinco y la wavelet que usaremos en el proceso será la de Haar, es decir, Db1. Veamos primero la imagen en varios niveles de resolución, ver Figura 3.2.

Figura 3.2: Imagen en diferentes niveles de resolución.

A pesar de que puede parecer que el ruido se va eliminando en las sucesivas aproximaciones, si viesemos las imágenes en grande se observaría que la pérdida de calidad en cada nivel es significativa, lo cual podemos evidenciar a partir de los niveles cuatro y cinco. Por lo tanto, veamos ahora la imagen una vez hemos filtrado los detalles de niveles uno y dos, ver Figura 3.3.

Figura 3.3: Filtrado con umbralización universal de detalles en niveles uno y dos.

3.2. Compresión de imágenes

La compresión de imágenes es un área fundamental de la tecnología en la que buscamos reducir el tamaño de los archivos sin comprometer la resolución de los mismos, o al menos comprometiendo la resolución en la menor medida posible.

Uno de los métodos de compresión de imágenes más conocido y utilizado es el JPEG, que utiliza para la compresión la transformada discreta del coseno (DCT por sus siglas en inglés). De manera similar a la DWT la DCT transforma la imagen en coeficientes que representan la información en térmnios de frecuencias. A menudo, los detalles más finos de la imagen pueden ser eliminados reduciendo así el tamaño de la imagen sin comprometer excesivamente la resolución. Uno de los principales problemas de esta transformada es que trabaja con bloques de tamaño fijo, generalmente de 8x8 píxeles, lo que puede dar lugar a que en la imagen se perciban las fronteras de los bloques.

Con el objetivo de superar algunas de las limitaciones del modelo JPEG, se desarrolló el modelo JPEG2000 el cual en lugar de utlizar la DCT para la compresión utiliza la DWT. Gracias al análisis multirresolución el modelo JPEG2000 permite descomponer la imagen en varios niveles de resolución, lo cual sirve para reducir los detalles más finos sin comprometer las componentes de baja frecuencia, lo más relevante para la imagen. Además, permite una primera versión de baja resolución y, conforme se obtienen más datos, esta puede ir siendo actualizada. También permite solucionar el problema de los bloques, si se trabaja con la wavelet adecuada, ya que a diferencia de la DCT trabaja con la imagen en su totalidad y no en bloques de píxeles.

Para ilustar la compresión bajo el algoritmo de la DWT se ha utilizado el código en el Apéndice A.6. Se ha utilizado una descomposición MRA en cinco niveles utilizando la wavelet Db6. A pesar de que existen métodos para elegir el umbral necesario para realizar una compresión, en este caso se ha tomado como umbral 0,09, de manera que este puede ser aumentado, si se desea más compresión a razón de una perdida de calidad en la imagen, o disminuido si se prefiere mejor resolución y más tamaño. Con el objetivo de tener una medida de compresión, se ha utilizado el cociente entre el número de bytes de la imagen original y el número de bytes de la imagen comprimida, resultando en este caso un valor para el cociente de 1,2194, es decir, una reducción del tamaño de casi el 18 % (ver Figura 3.4).

Figura 3.4: Compresión de una imagen.

3.3. Otras aplicaciones

Además de lo descrito anteriormente, existen otras muchas aplicaciones. También, haciendo ligeras modificaciones, todo lo anterior podría ser aplicado tanto a señales de audio como a archivos en tres dimensiones. Otra aplicación de la transformada wavelet discreta es, por ejemplo, la fusión de dos imágenes utilizando métodos similares a los decritos en [3] para combinar las matrices de aproximación y detalles con el fin de luego utilizar las matrices combinadas en la reconstrucción.

En resumen, la transformada wavelet discreta es una herramienta poderosa en el análisis y procesamiento de señales, permitiendo una representación multirresolución que es crucial en sus diversas aplicaciones. La capacidad de la DWT para descomponer señales en coeficientes y aplicar umbrales a los mismos permite tanto reducir de manera significativa el tamaño de los archivos como la recuperación de señales libres de ruido. Es decir, la DWT juega un papel fundamental en la transmisión de imágenes y videos, y la detección de anomalías, entre otras áreas, demostrando su relevancia en el procesamiento moderno de señales.

Apéndice A

Código

A.1. Representación de series

```
    %% FIGURA 1.1
% número de términos de la serie de Fourier
N = 5;

% intervalo de x
x = linspace(-pi, pi, 1000);

fx = abs(x);

a0 = pi / 2;

% Inicializar la aproximación de la serie de Fourier
f_approx = a0 * ones(size(x)) ;

% coeficientes an y construir la serie de Fourier
for n = 1:N
    an = (2 / (pi * n^2)) * ((-1)^n -1);
    f_approx = f_approx + an * cos(n * x);
end

% función original y su aproximación de Fourier
figure;
plot(x, fx, 'b', 'LineWidth', 1.5);
hold on;
plot(x, f_approx, 'r--', 'LineWidth', 1.5);
legend('|x|', 'Aproximación de Fourier');
title('Aproximación por Fourier (5 términos)');
xlabel('x');
ylabel('f(x)');
grid on;

%% FIGURA 1.2
% Número de términos en la serie de Fourier
N = 20;

% funciones para cada parte del dominio
f_neg = @(x) -pi/2 .* (x < 0 & x > -pi);
f_zero = @(x) 0 .* (x == 0);
f_pos = @(x) pi/2 .* (x > 0 & x < pi);
```

```
x_neg = linspace(-pi+0.01, -0.01, 500); % Intervalo para la parte negativa
x_zero = 0;                             % Punto x = 0
x_pos = linspace(0.01, pi-0.01, 500); % Intervalo para la parte positiva

% Intervalo X
x = linspace(-pi, pi, 1000);

a0 = (1 / (2 * pi)) * integral(@(x) f_neg(x) + f_zero(x) + f_pos(x), -pi, pi);
sum_fourier = a0 * ones(size(x)) / 2;

% coeficientes an y bn, y sumar términos de Fourier
for n = 1:N
    an = (1 / pi) * integral(@(x) (f_neg(x) + f_zero(x) + f_pos(x)) .* cos(n * x),
        -pi, pi);
    bn = (1 / pi) * integral(@(x) (f_neg(x) + f_zero(x) + f_pos(x)) .* sin(n * x),
        -pi, pi);
    sum_fourier = sum_fourier + an * cos(n * x) + bn * sin(n * x);
end

%  función original y la aproximación por Fourier
figure;
plot(x_neg, f_neg(x_neg), 'b'); % Parte negativa en rojo
hold on;
plot(0,0, 'bo', 'MarkerFaceColor','b', 'MarkerSize', 2); % Punto en x = 0 en verde
plot(x_pos, f_pos(x_pos), 'b'); % Parte positiva en azul
plot(x, sum_fourier, 'r', LineStyle='--');
hold off;
title('Aproximación por Fourier (20 términos)');
xlabel('x');
ylabel('f(x) / Aproximación de f(x)');
legend('Función', '', '', 'Aproximación por Fourier');
grid on;

%% Figura 2.1
% Parámetros
L = pi; % Intervalo [-L, L]
N = 20; % Número de términos en la serie de Haar
x = linspace(-L, L, 1024); % Puntos en los que se evalúa la aproximación,
                        % 1024 para que funcione el log2 con resultado
                           %entero

f = @(x) abs(x);

% Muestras de la función
num_points = length(x);
f_samples = f(x);

% Transformada de Haar completa
[coeffs, lengths] = wavedec(f_samples, log2(num_points), 'haar');

% Seleccionar los primeros N términos
coeffs_truncated = zeros(size(coeffs));
coeffs_truncated(1:N) = coeffs(1:N);

% Reconstrucción con la serie de Haar truncada
f_approx = waverec(coeffs_truncated, lengths, 'haar');

% función original y su aproximación
```

```
figure;
plot(x, f_samples, 'b-', 'DisplayName', 'Función Polinomial Original');
hold on;
plot(x, f_approx, 'r--', 'DisplayName', ['Aproximación de Haar (N=', num2str(N),
    ')']);
legend;
title('Aproximación de Haar ');
xlabel('x');
ylabel('f(x)');
grid on;
%% FiGura 2.2
% Parámetros
L = pi; % Intervalo [-L, L]
N = 2; % Número de términos en la serie de Haar
x = linspace(-L, L, 1024); % Puntos en los que se evalúa la aproximación

% Definición de la función por tramos
f = @(x) (x > 0) .* (pi / 2) + (x < 0) .* (-pi / 2)+ (x == 0) .* 0;

num_points = length(x);
f_samples = f(x);

% Transformada de Haar completa
[coeffs, lengths] = wavedec(f_samples, log2(num_points), 'haar');

% Seleccionar los primeros N términos
coeffs_truncated = zeros(size(coeffs));
coeffs_truncated(1:N) = coeffs(1:N);

% Reconstrucción con la serie de Haar truncada
f_approx = waverec(coeffs_truncated, lengths, 'haar');

% función original y su aproximación
figure;
plot(x, f_samples, 'b-', 'DisplayName', 'Función Original');
hold on;
plot(x, f_approx, 'r--', 'DisplayName', ['Aproximación de Haar (N=', num2str(N),
    ')']);
legend;
title('Aproximación de Haar de una Función Definida por Tramos');
xlabel('x');
ylabel('f(x)');
grid on;
```

A.2. Wavelets de Daubechies

```
    %% Figura 2.3
% Definir el rango de órdenes de Daubechies
orders = 1:4;


figure;

for idx = 1:numel(orders)
    order = orders(idx);
    % Obtener la wavelet y la función de escala
```

```
    [phi, psi, xval] = wavefun(['db', num2str(order)], 10);

    % subgráficos para la función de escala y la wavelet
    subplot(2, numel(orders), idx);
    plot(xval, phi, 'r', 'LineWidth', 1.5);
    title(['Escala - Db', num2str(order)]);
    xlabel('Tiempo');
    ylabel('Amplitud');
    grid on;

    subplot(2, numel(orders), idx + numel(orders));
    plot(xval, psi, 'b', 'LineWidth', 1.5);
    title(['Wavelet - Db', num2str(order)]);
    xlabel('Tiempo');
    ylabel('Amplitud');
    grid on;
end

%  tamaño de la figura
set(gcf, 'Position', [100, 100, 1200, 800]);
```

A.3. Algortimo DWT en una señal

```
x = [4 6 10 12 14 16 18 20]; %señal


[cA1, cD1] = HaarWaveletTransform(x); %descomposición nivel 1
[cA2, cD2] = HaarWaveletTransform(cA1);%descomposición nivel 2
[cA3, cD3] = HaarWaveletTransform(cA2);%descomposición nivel 3

disp('Detalles de nivel 1:');
disp(cD1);
disp('Detalles de nivel 2:');
disp(cD2);
disp('Detalles de nivel 3:');
disp(cD3);
disp('Aproximaciones de nivel 3:');
disp(cA3);

A2_rec=HaarInverseTransform(cA3,cD3);   %recostrución nivel 2
A1_rec=HaarInverseTransform(A2_rec,cD2); %recostrución nivel 1
x_rec=HaarInverseTransform(A1_rec,cD1); %recostrución de la señal

disp('Señal recostruida:');
disp(x_rec);

%%
% Función para la transformada Haar
function [cA, cD] = HaarWaveletTransform(x)
    N = length(x);
    if N == 1
        cA = x;
        cD = [];
    else
        cA = zeros(1, N/2);
```

```
        cD = zeros(1, N/2);
        for i = 1:N/2
            cA(i) = (x(2*i-1) + x(2*i)) / sqrt(2);
            cD(i) = (x(2*i-1) - x(2*i)) / sqrt(2);
        end
    end
end


% Función para la anti-transformada Haar
function x_reconstructed = HaarInverseTransform(cA, cD)
    N = length(cA) + length(cD);
    x_reconstructed = zeros(1, N);

    for i = 1:length(cA)
        x_reconstructed(2*i-1) = (cA(i) + cD(i)) / sqrt(2);
        x_reconstructed(2*i) = (cA(i) - cD(i)) / sqrt(2);
    end
end
```

A.4. Tratamiento de ruido (DWT)

```
    % Cargar la imagen
I = imread('prueba2.jpg');
%I = rgb2gray(I); % Convertir a escala de grises si es necesario
I = im2double(I); % Convertir a formato double

% Descomposición utilizando la DWT
waveletName = 'db2'; % Wavelet a utilizar

[CA, CH, CV, CD] = dwt2(I, waveletName); %desscomposicion dwt

% Añadimos ruido a los detalles con un factor de ruido
noise_factor= 30/255;
I_noisy=I+noise_factor*randn(size(I));

[CA, CH_noisy, CV_noisy, CD_noisy] = dwt2(I_noisy, waveletName);

%%


sigma_H = median(median(abs(CH_noisy))) / 0.6745; % universal Threshold
sigma_V = median(median(abs(CV_noisy))) / 0.6745; %
sigma_D = median(median(abs(CD_noisy))) / 0.6745; %

%Aplicamos el umbral universal

n = size(I, 1)*size(I,2)*3;
umbral_H = sigma_H * sqrt(2 * log(n))
umbral_V = sigma_V * sqrt(2 * log(n))
umbral_D = sigma_D * sqrt(2 * log(n))

CH_filt = umbralizar(CH_noisy,umbral_H);
CV_filt = umbralizar(CV_noisy,umbral_V);
CD_filt = umbralizar(CD_noisy,umbral_D);
```

```
% Reconstrucción de la imagen utilizando la IDWT
I_filt = idwt2(CA, CH_filt, CV_filt, CD_filt, waveletName);

figure;
imshow(I);
title('Imágen Original');

figure;
imshow(I_filt);
title('Imagen filtrada');

figure;
imshow(I_noisy);
title('Imagen con Ruido');
%%
figure;
subplot(1, 2, 1);
imshow(I_noisy);
title('Imagen Con ruido añadido');
subplot(1, 2, 2);
imshow(I_filt);
title('Imagen Filtrada con umbral ');

% Comparamos la original con la de después del filtrado
figure;

subplot(2, 2, 1);
imshow(I);
title('Imagen Original');

subplot(2, 2, 2);
imshow(I_noisy);
title('Imagen con Ruido');

subplot(2, 2, 3);
imshow(I_filt);
title('Imagen Filtrada');

%visualizar todo junto

h1 = subplot(2, 2, 1);
h2 = subplot(2, 2, 2);
h3 = subplot(2, 2, 3);
annotation('arrow', [0.43 0.61], [0.8 0.8], 'LineWidth', 2);
annotation('arrow', [0.75 0.45], [0.55 0.3], 'LineWidth', 2);
annotation('textbox', [0.42 0.77 0.2 0.1], 'String', 'Añadir Ruido', 'EdgeColor',
    'none', 'FontSize', 12, 'HorizontalAlignment', 'center');
annotation('textbox', [0.6 0.3 0.2 0.1], 'String', 'Filtrar Ruido', 'EdgeColor',
    'none', 'FontSize', 12, 'HorizontalAlignment', 'center');

%%
I = imread('prueba4.jpg');
I = rgb2gray(I); % Convertir a escala de grises si es necesario
I = im2double(I); % Convertir a formato double
I;

B= binarizar(I,0.6);
```

```
figure;
subplot(1, 2, 1);
imshow(I);
title('Imagen Original');
subplot(1, 2, 2);
imshow(B);
title('Imagen Filtrada');
%%
figure;
imshow(I_filt)
figure;
imshow(I_noisy)
%%

function B = umbralizar(A, umbral)

    % Crear una copia de A para B
    B = A;

    % Aplicar la umbralización
    B(A < umbral) = 0;
end
```

A.5. Tratamiento de ruido (MRA)

```
    % Leer y convertir la imagen a escala de grises
I = imread('prueba2.jpg');
I = rgb2gray(I); % Convertir a escala de grises si es una imagen en color
I = im2double(I);
%Añadir ruido

noise_factor= 16/255;
I_noisy=I+noise_factor*randn(size(I));

% Descomposición wavelet de la imagen con ruido
waveletName = 'db2'; % Wavelet a utilizar

level = 6; % Nivel de descomposición

[C, S] = wavedec2(I_noisy, level, waveletName);

%% Extraer y visualizar los coeficientes de aproximación en diferentes niveles(MRA)

% Extraer las aproximaciones en diferentes niveles
A1 = appcoef2(C, S, waveletName, 1);
A3 = appcoef2(C, S, waveletName, 3);
A5 = appcoef2(C, S, waveletName, 5);

figure;

subplot(2, 2, 1);
imshow(I_noisy,[]);
title('Imagen Ruidosa');

subplot(2, 2, 2);
imshow(A1,[]);
```

```
title('Aproximación Nivel 1');

subplot(2, 2, 3);
imshow(A3,[]);
title('Aproximación Nivel 3');

subplot(2, 2, 4);
imshow(A5,[]);
title('Aproximación Nivel 5');

%% Tratamiento del ruido método de universal Threshold

% Tomar los coeficientes de detalle de nivel uno
[H, V, D] = detcoef2('all', C, S,1); % Detalles de nivel uno

sigma_H = median(abs(H(:))) / 0.6745;
sigma_V = median(abs(V(:))) / 0.6745;
sigma_D = median(abs(D(:))) / 0.6745;

% Aplicar el umbral universal
n = size(I, 1)*size(I,2);
umbral_H = sigma_H * sqrt(2 * log(n))
umbral_V = sigma_V * sqrt(2 * log(n))
umbral_D = sigma_D * sqrt(2 * log(n))

H_filt= umbralizar(H,umbral_H);
V_filt= umbralizar(V,umbral_V);
D_filt= umbralizar(D,umbral_D);

% Actualizar C y filtrar otros niveles si es necesario

C_filt=replaceSubvector(C,H,H_filt); %se filtran los detalles de nivel uno
C_filt=replaceSubvector(C_filt,V,V_filt);
C_filt=replaceSubvector(C_filt,D,D_filt);

%% Repetir a otro nivel

nivelfiltrado = 2; % si se desea filtrar los detalles a otro nivel ejecutar este
           % trozo de código cambiando el nivel a filtar deseado

[H, V, D] = detcoef2('all', C, S, nivelfiltrado);
sigma_H = median(abs(H(:))) / 0.6745;
sigma_V = median(abs(V(:))) / 0.6745;
sigma_D = median(abs(D(:))) / 0.6745;

% Aplicar el umbral universal
n = size(I, 1)*size(I,2);
umbral_H = sigma_H * sqrt(2 * log(n))
umbral_V = sigma_V * sqrt(2 * log(n))
umbral_D = sigma_D * sqrt(2 * log(n))

H_filt= umbralizar(H,umbral_H);
V_filt= umbralizar(V,umbral_V);
D_filt= umbralizar(D,umbral_D);

%Actualizar C y repetir si se desea filtar otro nivel

C_filt=replaceSubvector(C_filt,H,H_filt);
```

```
C_filt=replaceSubvector(C_filt,V,V_filt);
C_filt=replaceSubvector(C_filt,D,D_filt);

%%
I_filt = waverec2(C_filt,S,waveletName); %recostrucción despues del filtrado

figure;
subplot(1, 2, 1);
imshow(I_noisy);
title('Imagen Original');

subplot(1, 2, 2);
imshow(I_filt);
title('Imagen filtrada');
figure;
imshow(I_filt)
%%
figure;
imshow(I_noisy);
title('Imagen con ruido artificial');
figure;
imshow(I_filt);
title('Imagen filtrada');
%%
function B = umbralizar(A, umbral)
    % Crear una copia de A para B
    B = A;

    % Aplicar la umbralización
    B(A < umbral) = 0;
end

function updatedVector = replaceSubvector(vector, subvectorToFind, subvectorToReplace)
    % Convertir los vectores a filas
    vector = vector(:)';
    subvectorToFind = subvectorToFind(:)';
    subvectorToReplace = subvectorToReplace(:)';

    % Longitudes de los vectores
    lenVector = length(vector);
    lenSubvectorToFind = length(subvectorToFind);
    lenSubvectorToReplace = length(subvectorToReplace);

    % Inicializar el vector actualizado con el original
    updatedVector = vector;

    % Iterar sobre el vector principal para encontrar el subvector
    i = 1;
    while i <= lenVector - lenSubvectorToFind + 1
        % Extraer el subvector actual del vector principal
        currentSubvector = vector(i:i + lenSubvectorToFind - 1);

        % Comparar el subvector actual con el subvector a encontrar
        if isequal(currentSubvector, subvectorToFind)
            % Reemplazar el subvector encontrado por el subvector de reemplazo
            updatedVector = [updatedVector(1:i-1), subvectorToReplace, updatedVector(i
                + lenSubvectorToFind:end)];
```

```
            % Actualizar el tamaño del vector
            lenVector = length(updatedVector);

            % Saltar adelante por el tamaño del subvector de reemplazo
            i = i + lenSubvectorToReplace;
        else
            i = i + 1;
        end
    end
end
```

A.6. Compresión

```
    % Leer una imagen y convertirla a escala de grises
img = imread('prueba2.jpg');
%img_gray = im2double(rgb2gray(img)); % si fuera nesesario se pasa a blanco
%y negro
img=im2double(img);

% Mostrar la imagen original
figure;
imshow(img);
title('Imagen Original');

% Nivel de descomposición
level = 5;
waveletName = 'db6'; % Wavelet a utilizar

% Descomposición de la imagen utilizando la DWT
[c, s] = wavedec2(img, level, waveletName);

% Calcular el umbral de compresión
% threshold = wthrmngr('dw2dcompGBL','rem_n0',c,s);

threshold=0.09 % Se elije el umbral segun lo que se desee o se calcula con lo
    descrito arriba


% Aplicar el umbral para comprimir
c_compressed = wthresh(c, 'h', threshold);

% Reconstrucción de la imagen comprimida
img_compressed = waverec2(c_compressed, s, waveletName);

% Mostrar la imagen comprimida
figure;
imshow(img_compressed,[]);
title('Imagen Comprimida');

%se guardan las imagenes para calcular la tasa de compresión.

imwrite((img_compressed), 'compressed_image.jpg');
imwrite((img), 'ProgramaPrueba2.jpg');


originalFileInfo = dir('ProgramaPrueba2.jpg');
```

```
compressedFileInfo = dir('compressed_image.jpg');

originalFileSize = originalFileInfo.bytes;
compressedFileSize = compressedFileInfo.bytes;

% Calcular la tasa de compresión
compressionRatio = originalFileSize / compressedFileSize
```

Bibliografía

[1] D.HONG, J.WANG Y R.GARDEN. , *Real analysis with an introduction to wavelets and applications.* Elsevier, 2004.

[2] PEREYRA, MARÍA CRISTINA; WARD, LESLEY A. *Harmonic analysis: from Fourier to wavelets.* American Mathematical Soc., 2012.

[3] ALGORITMOS PARA REDUCCIÓN DE RUIDO EN SEÑALES. `http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/hernandez_d_m/capitulo3.pdf`.

[4] SCHELKENS, PETER, ATHANASSIOS SKODRAS, AND TOURADJ EBRAHIMI, *The JPEG 2000 suite.*, John Wiley & Sons, 2009.

[5] GARCÍA RAMOS, ROMÁN. *Compresión de imágenes fijas en MATLAB a través de DCT y WAVELET.* Tesis de Maestría. Departamento de Ingeniería Electrónica. Universidad de las Américas, Puebla. Enero 2003.

[6] MALLAT, STEPHANE. *Multiresolution approximations and wavelet orthonormal bases of* $L^2(\mathbb{R})$. Transactions of the American mathematical society, 1989, vol. 315, no 1, p. 69-87.