Departamento de Matemáticas, Facultad de Ciencias
Universidad Autónoma de Madrid

Aspectos matemáticos del tratamiento de señales

Trabajo de fin de grado

Grado en Matemáticas

Autor: Marta Corchero Gijón
Tutor: Fernando Chamizo

Curso 2024-2025

Resumen

Este trabajo estudia el tratamiento de señales desde una perspectiva matemática, centrándose en herramientas como la Transformada de Fourier (discreta y continua), la Transformada Discreta del Coseno (DCT), el muestreo de señales y los filtros FIR. En los primeros capítulos se presentan los fundamentos teóricos, mostrando cómo las señales pueden analizarse mediante descomposición en frecuencias y cómo ciertas transformadas permiten representar y manipular la información de manera más eficiente. También se aborda el teorema de muestreo de Shannon, que garantiza la reconstrucción de señales a partir de muestras discretas bajo ciertas condiciones. En cuanto a los filtros FIR, se estudia su implementación y el fenómeno de Gibbs, además del uso de ventanas como la de Hamming o Hann para suavizar oscilaciones no deseadas. En la parte final se introduce la teoría de la información, incluyendo el concepto de entropía de Shannon y su aplicación en la compresión de datos mediante códigos binarios prefijo, con especial énfasis en los límites de compresión establecidos por la entropía. El trabajo concluye con aplicaciones prácticas como la compresión JPEG y la tomografía, mostrando la relevancia de las matemáticas en tecnologías modernas.

Abstract

This thesis explores signal processing from a mathematical perspective, focusing on tools such as the Fourier Transform (both discrete and continuous), the Discrete Cosine Transform (DCT), signal sampling, and FIR filters. The early chapters present the theoretical foundations, illustrating how signals can be analyzed through frequency decomposition and how certain transforms enable more efficient representation and manipulation of information. The work also discusses Shannon's sampling theorem, which ensures the reconstruction of signals from discrete samples under certain conditions. Regarding FIR filters, it studies their implementation and the Gibbs phenomenon, along with the use of window functions like Hamming and Hann to reduce undesirable oscillations. In the final part, the thesis introduces information theory, including Shannon's entropy and its application to data compression via binary prefix codes, with particular attention to the entropy-based limits of compression. The thesis concludes with practical applications such as JPEG compression and tomography, highlighting the importance of mathematics in modern technology.

Índice general

CAPÍTULO 1

Análisis de Fourier discreto y continuo

Antes de nada, vamos a aclarar el término "señales" que aparece en el título. Para nosotros, una señal será simplemente una función (por ejemplo, de $\{0, 1, \dots, N\}$ en $\mathbb{C}$) que pensamos que contiene alguna información relevante.

Fijemos también un par de cosas sobre la notación.

- La primera es que usaremos la abreviatura $e(x)$ para indicar $e^{2\pi i x}$.
- La segunda es que $\langle \vec{a}, \vec{b} \rangle$ denotará el producto escalar. Tomaremos $\langle \vec{a}, \vec{b} \rangle = \sum_j \overline{a}_j b_j$ como producto escalar usual en $\mathbb{C}^n$.

En esta primera sección nos dedicaremos sobre todo a aspectos básicos del análisis de Fourier discreto y continuo.

Una señal dada por N valores reales o complejos $\{x_n\}_{n=0}^{N-1}$, por ejemplo, correspondientes a diferentes instantes de tiempo, se puede representar como el vector $\vec{x} = (x_0, x_1, \dots, x_{N-1})$.

Definición 1. *Se llama* ***Transformada de Fourier Discreta****, abreviada como DFT por sus siglas en inglés, al vector*

$$DFT(\vec{x}) = (\widehat{x}_0, \widehat{x}_1, \dots, \widehat{x}_{N-1}) \qquad con \qquad \widehat{x}_n = \sum_{m=0}^{N-1} x_m \, e\Big(-\frac{nm}{N}\Big).$$

Sea el vector $\vec{v}_n = \Big(e\big(\frac{0\cdot n}{N}\big), e\big(\frac{1\cdot n}{N}\big), \dots, e\big(\frac{(N-1)\cdot n}{N}\big)\Big)$, nos fijamos en $\{\vec{v}_0, \dots, \vec{v}_{N-1}\}$. Estudiando su producto escalar $\langle \vec{v}_n, \vec{v}_m \rangle$, podemos notar que se trata de una base ortogonal de $\mathbb{C}^N$.

$$\langle \vec{v}_n, \vec{v}_m \rangle = \sum_{k=0}^{N-1} \overline{e\big(\frac{kn}{N}\big)} e\big(\frac{km}{N}\big) = \sum_{k=0}^{N-1} e\big(-\frac{kn}{N}\big) e\big(\frac{km}{N}\big) = \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} k(m-n)}. \tag{1.0.1}$$

Tenemos una suma geométrica en la que $r = e\big(\frac{m-n}{N}\big)$, por lo tanto:

$$\langle \vec{v}_n, \vec{v}_m \rangle = \frac{1 - r^N}{1 - r} = \frac{1 - e^{2\pi i (n-m)}}{1 - r} \underset{n \neq m}{=} \frac{1 - 1}{1 - r} = 0.$$

Hemos visto que los vectores son ortogonales en $\mathbb{C}^N$. Ahora, si normalizamos los vectores:

$$\langle \vec{v}_n, \vec{v}_n \rangle = \sum_{k=0}^{N-1} e\big(-\frac{kn}{N}\big) e\big(\frac{kn}{N}\big) = \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} k(n-n)} = N. \tag{1.0.2}$$

Se tiene entonces que $\vec{u}_n = \vec{v}_n/\sqrt{N}$ es una base ortonormal de $\mathbb{C}^N$, por lo tanto cualquier vector $\vec{x} \in \mathbb{C}^N$ se puede expresar como combinación lineal de $\{\vec{u}_0, \ldots, \vec{u}_{N-1}\}$. Es decir, dado un vector $\vec{x} = (x_0, x_1, \ldots, x_{N-1})$, podemos descomponerlo en términos de la base $\{\vec{u}_n\}$ como

$$\vec{x} = \sum_{m=0}^{N-1} \langle \vec{u}_m, \vec{x} \rangle \, \vec{u}_m \quad \Rightarrow \quad x_n = \sum_{m=0}^{N-1} \frac{\hat{x}_m}{\sqrt{N}} \frac{1}{\sqrt{N}} e\big(\frac{nm}{N}\big) = \frac{1}{N} \sum_{m=0}^{N-1} \hat{x}_m \, e\big(\frac{nm}{N}\big),$$

que prueba el siguiente resultado, llamado fórmula de inversión de la DFT.

Proposición 1.1. *Sea x un vector en $\mathbb{C}^N$, entonces*

$$x_m = \frac{1}{N} \sum_{n=0}^{N-1} \hat{x}_n \, e\big(\frac{nm}{N}\big), \quad \textit{con} \;\; m = 0, 1, \ldots, N-1.$$

Como acabamos de ver, podemos definir

$$\vec{x} = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \hat{x}_m \, \vec{u}_m \quad \text{e} \quad \vec{y} = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \hat{y}_m \, \vec{u}_m.$$

Por lo tanto, su producto escalar teniendo en cuenta que los $\vec{u}_m$ son ortonormales, es

$$\langle \vec{x}, \vec{y} \rangle = \frac{1}{N} \sum_{m=0}^{N-1} \overline{\hat{x}_m} \, \hat{y}_m.$$

Lo cual es exactamente la *identidad de Parseval*:

$$\langle \vec{x}, \vec{y} \rangle = \frac{1}{N} \langle DFT(\vec{x}), DFT(\vec{y}) \rangle.$$

Si tomamos $\vec{x} = \vec{y}$, obtenemos la *identidad de Parseval* para normas

$$\|\vec{x}\|^2 = \frac{1}{N} \sum_{m=0}^{N-1} |\hat{x}_m|^2 = \frac{1}{N} \|DFT(\vec{x})\|^2.$$

Al igual que los coeficientes de Fourier, las coordenadas de la DFT indican el contenido en frecuencias. De cara a las aplicaciones hay dos inconvenientes menores. El primero es que si uno parte de $\vec{x} \in \mathbb{R}^N$ en general $\mathrm{DFT}(\vec{x}) \notin \mathbb{R}^N$. Dependiendo del *software* disponible, esto puede ser un problema. El segundo es más sutil. Gracias a la igualdad $e\big(\frac{nm}{N}\big) = e\big(\frac{n(N-m)}{N}\big)$, en la fórmula de inversión $\widehat{x}_m$ y $\widehat{x}_{N-m}$ multiplican ambos a algo de frecuencia m/N, pues oscila m veces en $[0, N)$. Así, subíndices altos no implica frecuencias altas, lo cual es un poco lioso cuando uno quiere filtrar señales.

Estas dos dificultades se resuelven con una nueva transformada, algo menos importante que la DFT, pero, también, ampliamente usada.

Definición 2. *La **transformada de coseno discreta**, abreviada por sus siglas en inglés DCT, se define para* $\vec{x} = \{x_n\}_{n=0}^{N-1} \in \mathbb{C}^N$ *como*

$$DCT(\vec{x}) = (\widehat{x}^c_0, \widehat{x}^c_1, \ldots, \widehat{x}^c_{N-1}) \qquad con \qquad \widehat{x}^c_n = \sum_{m=0}^{N-1} x_m \cos\Big(\frac{\pi n}{N}\big(m+\frac{1}{2}\big)\Big).$$

En realidad la DCT es la DFT aplicada a una señal simetrizada para evitar la aparición de números complejos y discontinuidades.

Dada una señal $\vec{x} \in \mathbb{C}^N$, sea $\vec{y} \in \mathbb{C}^{2N}$ la señal definida por $y_n = x_n$ si $0 \le n < N$ e $y_n = x_{2N-1-n}$ si $N \le n < 2N$. Consideremos los subíndices de $\widehat{y}_n$ módulo $2N$, por ejemplo, $\widehat{y}_{-1}$ se define como $\widehat{y}_{2N-1}$. Se cumple

$$\begin{aligned}\widehat{y}_n &= \sum_{m=0}^{2N-1} y_m e\Big(-\frac{nm}{2N}\Big) = \sum_{m=0}^{N-1} x_m e\Big(-\frac{nm}{2N}\Big) + \sum_{m=N}^{2N-1} x_{2N-1-m} e\Big(-\frac{nm}{2N}\Big) = \\ &= \sum_{m=0}^{N-1} x_m e\Big(-\frac{nm}{2N}\Big) + \sum_{m=0}^{N-1} x_m e\Big(-\frac{n(2N-1-m)}{2N}\Big) = \sum_{m=0}^{N-1} x_m \Big[e\Big(-\frac{nm}{2N}\Big) + e\Big(\frac{n(m+1)}{2N}\Big)\Big].\end{aligned}$$

De aquí obtenemos lo siguiente:

$$\begin{aligned}\widehat{y}_n &= \sum_{m=0}^{N-1} x_m e\Big(\frac{n}{4N}\Big)\Big[e\Big(-\frac{n(m+\frac{1}{2})}{2N}\Big) + e\Big(\frac{n(m+\frac{1}{2})}{2N}\Big)\Big] = \\ &= \sum_{m=0}^{N-1} x_m e\Big(\frac{n}{4N}\Big) 2\cos\Big(\frac{\pi n}{N}(m+\frac{1}{2})\Big) = 2e\Big(\frac{n}{4N}\Big)\widehat{x}^c_n.\end{aligned}$$

Y por la fórmula de inversión de la DFT se tiene:

$$\begin{aligned}y_m &= \frac{1}{2N}\sum_{n=-N}^{N-1} \widehat{y}_n e\Big(\frac{nm}{2N}\Big) = \frac{1}{2N}\sum_{n=-N}^{N-1} 2\widehat{x}^c_n e\Big(\frac{n}{4N}\Big) e\Big(\frac{nm}{2N}\Big) = \frac{1}{N}\sum_{n=-N}^{N-1} \widehat{x}^c_n e\Big(\frac{n(m+\frac{1}{2})}{2N}\Big) \\ &= \frac{\widehat{x}^c_0}{N} + \frac{2}{N}\sum_{n=1}^{N-1} \widehat{x}^c_n e\Big(\frac{n(m+\frac{1}{2})}{2N}\Big) = \frac{\widehat{x}^c_0}{N} + \frac{2}{N}\sum_{n=1}^{N-1} \widehat{x}^c_n \cos\Big(\frac{\pi n}{N}(m+\frac{1}{2})\Big).\end{aligned}$$

En analogía con la DFT, a esta operación sobre los $\widehat{x}^c_n$ se le llama IDCT y entonces la fórmula de inversión afirma $\mathrm{IDCT}\big(\mathrm{DCT}(\vec{x})\big) = \vec{x}$.

En la DFT $\widehat{x}_n$ y $\widehat{x}_{N-n}$ daban ambas el contenido en la frecuencia n/N mientras que en la DCT, $\widehat{x}^c_n$ representa el contenido en la frecuencia $n/(2N)$, ahora sí un n grande corresponde a una frecuencia grande.

Veamos todo esto en un ejemplo práctico. Consideremos las funciones

$$f(x) = 4\Big(\frac{x}{117}\Big)^2\Big(1-\frac{x}{117}\Big)^2 \qquad \text{y} \qquad g(x) = \Big|\frac{2}{9}x - \lfloor\frac{2}{9}x + \frac{1}{2}\rfloor\Big| - \frac{1}{4} \tag{1.0.3}$$

donde $\lfloor x \rfloor$ indica la parte entera. La primera función es bastante suave en el intervalo $[0, 59)$, mientras que la segunda oscila unas 13 veces. La señal $\vec{x} = \{f(n) + 0{,}02g(n)\}_{n=0}^{58} \in \mathbb{R}^{59}$ puede considerarse como una perturbación con ruido periódico de $\vec{s} = \{f(n)\}_{n=0}^{58}$.

Se ha hecho un programa, que se muestra en B, en el cual se limpia la señal $\vec{x}$ siguiendo el procedimiento: $\vec{x} \mapsto \text{IDCT}\big(H_{10}(\vec{x})\big)$ donde $H_{10}(\vec{x}) = (\widehat{x}_0^c, \widehat{x}_1^c, \ldots, \widehat{x}_9^c, 0, 0, \ldots, 0)$.

Este procedimiento hace uso de la *Transformada Discreta del Coseno (DCT)*. Este programa toma la señal y la descompone en coeficientes de baja frecuencia (aquellos en los que la señal es suave) y en coeficientes de alta frecuencia (donde la señal tiene oscilaciones). A continuación, se toman los 10 primeros coeficientes de la DCT, que son aquellos que contienen la mayor parte de la información relevante de la señal, y se elimina el resto de coeficientes que es donde se producen las oscilaciones. Por último, aplicamos la fórmula de inversión de la DCT para recuperar la señal libre de oscilaciones. Tras aplicar todo esto, nos queda la señal limpia, que es muy similar a $\vec{s}$.

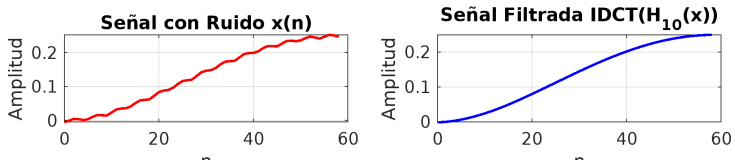


Figura 1.1: Señal limpia vs Señal ruidosa

Ahora, vamos a modificar el programa anterior, sustituyendo H_{10} por H_k con $0 < k < 59$. Es decir, un nuevo programa en el que se mantengan los k primeros coeficientes de la DCT. Queremos encontrar un rango aproximado de k para el cual la señal quede limpia de oscilaciones. Para ello, vamos a tomar distintos valores de k y veremos como afectan a la limpieza de la señal.

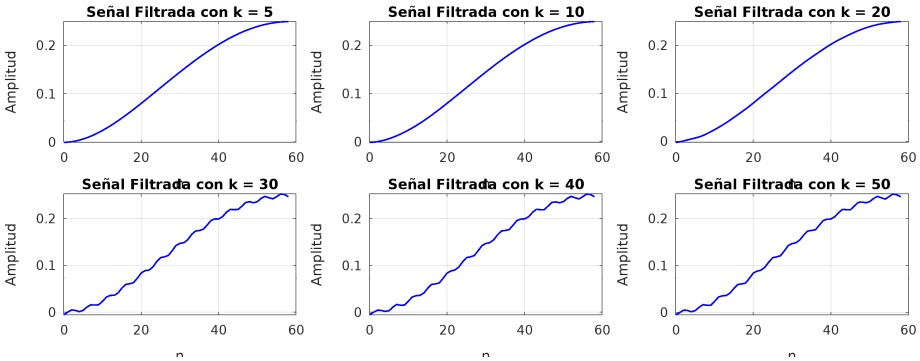


Figura 1.2: Comparación de limpieza con diferentes k

Podemos observar que cuanto mayor es el valor de k, peor el la limpieza de la señal. El rango óptimo de k para la limpieza de la señal parece estar entre 6 y 19. Esto se debe a que el ruido está contenido en los coeficientes de mayor índice de la DCT, mientras que la información más importante de la señal original está en los coeficientes más bajos.

Si tomásemos un k demasiado pequeño ($k = 1, 2$), perderíamos detalles importantes de la señal. Por el contrario, si k fuese demasiado grande, el ruido no se eliminaría del todo.

Estas transformaciones discretas en algunas aplicaciones se usan sobre señales muy largas, con N grande, y se vuelve crucial llevar a cabo las operaciones de forma eficiente. En principio, el cálculo de $\widehat{x}_n$ requiere N multiplicaciones y como hay N valores de n, parece que nada nos libra de al menos N^2 multiplicaciones para calcular la DFT. Sorprendentemente, hay un algoritmo llamado FFT (por *Fast Fourier Transform*) que cuando N es una potencia de dos (un caso importante) reduce enormemente esta cota.

El cálculo directo de la DFT implica un número de operaciones proporcional a N^2, lo que puede ser tedioso para valores grandes de N. La FFT reduce este costo computacional a $O(N \log N)$, utilizando una técnica conocida como "divide y vencerás". Esto hace que la FFT sea una herramienta esencial en aplicaciones prácticas.

El algoritmo parte del hecho de que los cálculos en la DFT contienen redundancias que pueden ser explotadas. Para un tamaño N que es una potencia de dos ($N = 2^k$), el algoritmo divide la señal en partes más pequeñas, realiza la DFT en estas partes y luego combina los resultados usando propiedades de periodicidad de las raíces de la unidad y los llamados "factores de giro" o *twiddle factors*. La relación básica del algoritmo es:

$$\hat{x}_n = y_{n0} + e^{-2\pi i n/N} y_{n1} \quad \text{donde} \quad y_{nj} = \sum_{m=0}^{N/2-1} x_{2m+j} e^{-2\pi i m n/(N/2)}, \quad j = 0, 1.$$

Aquí, y_{n0} y y_{n1} representan sumas parciales de los datos de entrada, cada una calculada con la mitad del tamaño original $N/2$. Este proceso se repite de manera recursiva hasta alcanzar el caso base ($N = 1$). Gracias a esta descomposición, el número de operaciones requeridas para calcular la FFT se reduce drásticamente. Para ($N = 2^k$), el número total de operaciones es $M(N) = 2N \log_2 N$, donde $M(N)$ representa la cantidad de multiplicaciones necesarias. Este cambio de $O(N^2)$ a $O(N \log N)$ es lo que hace a la FFT revolucionaria, permitiendo manejar grandes conjuntos de datos con una eficiencia sorprendente.

Una vez que hemos visto el análisis de Fourier discreto, repasemos el continuo. Recordemos que una función f que sea 1-periódica y que tenga regularidad suficiente, por ejemplo Lipschitz, coincide con su serie de Fourier, esto es,

$$f(x) = \sum_{n\in\mathbb{Z}} f_n\, e(nx) \qquad \text{con} \quad f_n = \int_0^1 f(t)\, e(-nt)\, dt.$$

Usando sumas de Riemann, este desarrollo de Fourier se puede ver como un límite de la fórmula de inversión de la DFT cuando $N \to \infty$, aunque no lo haremos aquí.

Sea $G(x) = g(9x/2)$ con g definida en (11), se tiene $G(x) = \big|x - \lfloor x + 1/2\rfloor\big| - 1/4$ y es fácil ver que es 1-periódica y Lipschitz. Antes de calcular los coeficientes de Fourier, notemos que se puede simplificar la función de modo que sea más sencilla.

$$\begin{aligned} &\text{Si} \quad 0 \le x \le 1/2: \quad \lfloor x + 1/2\rfloor = 0 \quad \Longrightarrow \quad G(x) = x - 1/4. \\ &\text{Si} \quad 1/2 \le x \le 1: \quad \lfloor x + 1/2\rfloor = 1 \quad \Longrightarrow \quad G(x) = |x-1| - 1/4 = 3/4 - x. \end{aligned}$$

Ahora nos queda lo siguiente:

$$f_n = \int_0^1 f(t)\, e(-nt)\, dt = \int_0^{1/2} (t - 1/4)\, e(-nt)\, dt + \int_{1/2}^1 (3/4 - t)\, e(-nt)\, dt.$$

Tras algunos cálculos, esto conduce a

$$f_n = \frac{(-1)^n - 1}{2\pi^2 n^2}.$$

Dado que $G(x)$ es impar y 1-periódica, el cálculo de los coeficientes de Fourier lleva a términos de senos y cosenos. Además, se observa que cuando $n = 2k$ los términos de f_n se anulan. Esto se debe a que la integral de una función impar multiplicada por una función par se anula al evaluarse sobre un intervalo simétrico de un período. Por otro lado, para $n = 2k+1$, los términos no se cancelan. Así pues se deduce

$$G(x) = \sum_{n\in\mathbb{Z}} f_n e(nx) = -\frac{2}{\pi^2}\sum_{k=0}^{\infty} \frac{e((2k+1)x)}{(2k+1)^2} = -\frac{2}{\pi^2}\sum_{k=0}^{\infty} \frac{\cos(2\pi(2k+1)x)}{(2k+1)^2}.$$

El análogo de los coeficientes de Fourier para una función $f : \mathbb{R} \to \mathbb{C}$ no periódica es la transformada de Fourier definida mediante

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e(-\xi x)\,dx.$$

Esta integral tiene sentido (Lebesgue) para $f \in L^1(\mathbb{R})$. La fórmula de inversión

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)e(-x\xi)\,d\xi\,,$$

recupera f a partir de $\hat{f}$ y es más exigente con la regularidad. Es cierta para funciones de decaimiento rápido, aunque modificando un poco el sentido de las integrales se puede extender a $L^2(\mathbb{R})$ [6, §2.3].

Para series e integrales de Fourier, la *identidad de Parseval* es, respectivamente,

$$\langle f, g\rangle = \langle \{f_n\}_{n\in\mathbb{Z}}, \{g_n\}_{n\in\mathbb{Z}}\rangle \quad \text{y} \quad \langle f, g\rangle = \langle \hat{f}, \hat{g}\rangle$$

donde el producto escalar de funciones es el habitual en $L2$, esto es, $\int_0^1 \overline{f}g$ en el primer caso y $\int_{-\infty}^{\infty} \overline{f}g$ en el segundo, y el de sucesiones es el de ℓ^2, esto es, el segundo miembro de la primera igualdad significa $\sum_{n\in\mathbb{Z}} \overline{f_n} g_n$.

Lema 1.2. *La transformada de Fourier de* $f(x) = e^{-\alpha\pi x^2}$ *con* $\alpha > 0$ *es* $\widehat{f}(\xi) = \alpha^{-1/2}e^{-\pi\xi^2/\alpha}$.

Observación 1. *La transformada de Fourier de* $e^{-\pi x^2}$ *es ella misma.*

En el apéndice A se muestra la demostración de este lema, que es un resultado que usaremos más adelante.

CAPÍTULO 2
Muestreo y fliltros FIR

Antes de digitalizar una señal, es fundamental entender qué información se conserva cuando tomamos solo ciertos valores de ella y cómo podemos controlar su contenido en frecuencias a partir de estos valores. En este capítulo abordaremos estas dos cuestiones, que son clave en el procesamiento digital de señales.

El primer paso en este proceso es el *muestreo*, que consiste en registrar el valor de la señal $f : \mathbb{R} \longrightarrow \mathbb{C}$ en instantes espaciados regularmente en el tiempo. Si denotamos por ν_s la *frecuencia de muestreo*, es decir, el número de muestras tomadas por unidad de tiempo, entonces esto equivale a considerar la secuencia $\{f(n/\nu_s)\}_{n\in\mathbb{Z}}$. A primera vista, podría parecer que al trabajar con una cantidad discreta de valores estamos perdiendo información. Sin embargo, el teorema de muestreo de Shannon, también conocido como teorema de Nyquist-Shannon, establece que, bajo ciertas condiciones, es posible recuperar la señal original a partir de sus muestras.

Teorema (de muestreo de Shannon)**.** *Sean $\nu_s > 0$ y $f : \mathbb{R} \longrightarrow \mathbb{C}$ tal que el soporte de $\widehat{f}$ está contenido en $I = [-\nu_s/2, \nu_s/2]$, entonces*

$$f(t) = \sum_{n=-\infty}^{\infty} f(n/\nu_s)\,\mathrm{sinc}(\nu_s t - n) \qquad \textit{con} \quad \mathrm{sinc}(x) = \frac{\mathrm{sen}(\pi x)}{\pi x} \ \textit{si}\ x \neq 0 \quad y \quad \mathrm{sinc}(0) = 1.$$

Este teorema se basa en la propiedad de la función sinc de interpolar exactamente los valores muestreados, garantizando así la reconstrucción de la señal original. Esto es, si una señal no contiene componentes de frecuencia superiores a $\nu_s/2$, entonces no hay pérdida de información en su reconstrucción. Conocer sus valores en los puntos n/ν_s es suficiente para determinarla completamente en cualquier otro instante.

Demostración. Bajo las hipótesis del teorema, sea g la extensión ν_s-periódica de $\widehat{f}$ restringida a I. Es decir, $g(x) = \widehat{f}(x - \nu_s n)$ con un $n \in \mathbb{Z}$ tal que $|x - \nu_s n| \leq \nu_s/2$. En primer lugar, vamos a representar $g(\xi)$ como una serie de Fourier:

$$g(\xi) = \sum_{n\in\mathbb{Z}} c_n e(n\xi/\nu_s), \quad \text{donde} \quad c_n = \nu_s^{-1} \int_I g(x) e(-nx/\nu_s)\, dx.$$

Como se tiene que g es una extensión periódica de $\widehat{f}$ restringida a I, en este intervalo $g(x) = \widehat{f}(x)$. Por lo tanto:

$$g(\xi) = \nu_s^{-1} \sum_{n=-\infty}^{\infty} e(n\xi/\nu_s) \int_I \widehat{f}(x) e(-nx/\nu_s)\, dx.$$

A continuación, usando la fórmula de inversión de Fourier ($f(n/\nu_s) = \int_I \widehat{f}(x) e(nx/\nu_s)\, dx$), y haciendo un cambio de $n \mapsto -n$, tenemos:

$$g(\xi) = \nu_s^{-1} \sum_{n=-\infty}^{\infty} f(n/\nu_s) e(-n\xi/\nu_s). \tag{2.0.1}$$

Como sabemos, la fórmula de inversión es $f(t) = \int_{\mathbb{R}} \widehat{f}(\xi) e(t\xi)\, d\xi$. Pero al tener $\widehat{f}(\xi)$ soporte en I, la integral puede restringirse a $f(t) = \int_I \widehat{f}(\xi) e(t\xi)\, d\xi = \int_I g(\xi) e(t\xi)\, d\xi$, y sustituyendo (2.0.1):

$$f(t) = \int_I \nu_s^{-1} \sum_{n=-\infty}^{\infty} f(n/\nu_s) e(-n\xi/\nu_s) e(t\xi)\, d\xi = \nu_s^{-1} \sum_{n=-\infty}^{\infty} f(n/\nu_s) \int_I e((t - n/\nu_s)\xi)\, d\xi.$$

Resolviendo esta integral se obtiene $\int_I e((t - n/\nu_s)\xi)\, d\xi = \nu_s \operatorname{sinc}(\nu_s t - n)$ y se concluye el resultado. □

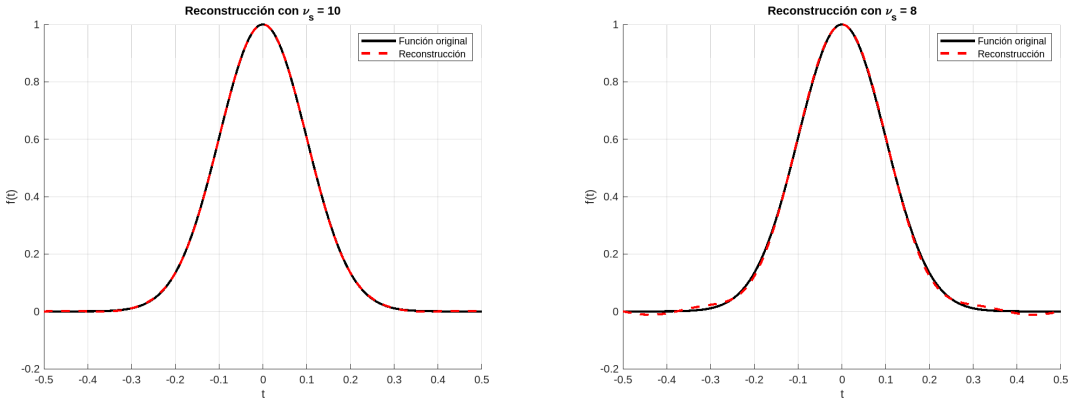
En la práctica, la reconstrucción exacta de una señal mediante el teorema de muestreo de Shannon requiere evaluar una suma infinita de términos sinc, lo cual no es posible directamente en un ordenador. Para hacer una representación computacional, es necesario aproximar la serie por una suma finita. Una forma natural de justificar este truncamiento es observar que, si la transformada de Fourier de la señal original f tiene su energía concentrada en un intervalo acotado, los términos asociados a frecuencias fuera de ese intervalo serán despreciables y no afectarán significativamente a la reconstrucción.

Para ilustrar este punto, tomemos como ejemplo la función $f(t) = e^{-\pi(4t)^2}$. En el capítulo anterior se ha visto que su transformada de Fourier está dada por $\widehat{f}(\xi) = \frac{1}{4} e^{-\pi(\xi/4)^2}$. Esta función alcanza su máximo en $\xi = 0$ con valor $\widehat{f}(0) = \frac{1}{4}$ y decrece rápidamente a medida que $|\xi|$ crece. En particular, para $|\xi| > 5$, el valor de $\widehat{f}(\xi)$ es más de 100 veces más pequeño que en su máximo, lo que indica que el contenido en frecuencias de f fuera del intervalo $[-5, 5]$ es prácticamente despreciable. Por tanto, podemos aproximar $\widehat{f}$ por una función de soporte contenido en $[-5, 5]$ sin perder demasiada precisión.

Aprovechando esta aproximación, vamos a aplicar el teorema de muestreo de Shannon y representar gráficamente la reconstrucción de la señal a partir de sus muestras discretas. En particular, estudiaremos cómo la señal original puede ser recuperada a partir de sus valores muestreados con distintas frecuencias de muestreo.

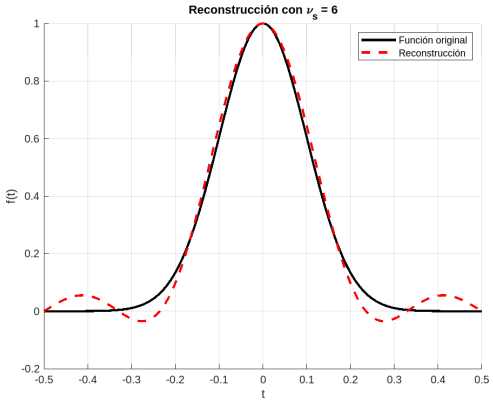
Primero, tomamos $\nu_s = 10$. Dado que el contenido en frecuencias de f está esencialmente contenido en $[-5, 5]$, la hipótesis de teorema de Shannon se cumple de manera aproximada para esta elección de ν_s. Por tanto, la reconstrucción mediante la interpolación sinc debería aproximarse casi perfectamente a la señal original.

Además, exploraremos qué ocurre cuando se utilizan frecuencias de muestreo menores, eligiendo $\nu_s = 4, 6, 8$. En estos casos, la condición del teorema no se satisface de forma tan clara, por lo que esperamos que la reconstrucción pierda precisión. Comparando las distintas gráficas, podremos visualizar cómo la calidad de la reconstrucción empeora a medida que la frecuencia de muestreo disminuye.

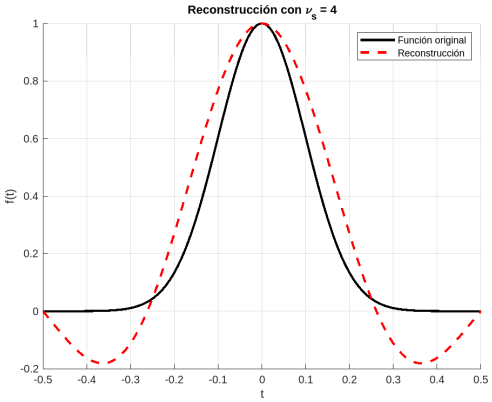


(a) $\nu_s = 10$

(b) $\nu_s = 8$



(c) $\nu_s = 6$



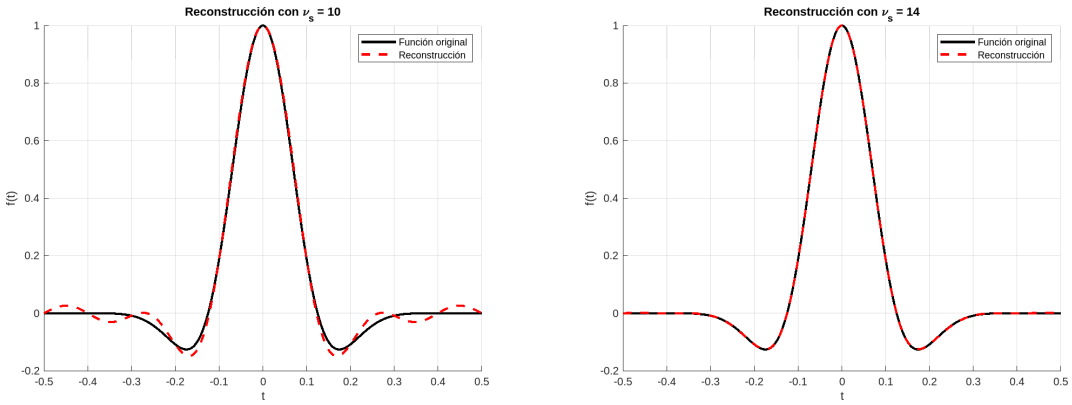
(d) $\nu_s = 4$

Figura 2.1: Reconstrucciones de $f(t) = e^{-\pi(4t)^2}$

Ahora consideramos la función $f(t) = e^{-\pi(4t)^2}\cos(4\pi t)$, que tiene una forma similar a la que acabamos de analizar. Sin embargo, al aplicar el teorema de Shannon con $\nu_s = 10$, observamos que la reconstrucción no es tan precisa como en el caso anterior. En cambio, al aumentar la frecuencia de muestreo a $\nu_s = 14$, la aproximación vuelve a ser prácticamente idéntica a la señal original.

Para comprender mejor por qué ocurre esto, calculemos la transformada de Fourier de $f(t)$. Podemos usar la propiedad de modulación en la transformada de Fourier: si $h(t) = g(t)\cos\xi_0\pi t$, entonces su transformada de Fourier es $\widehat{h}(\xi) = \frac{1}{2}\Big[\widehat{g}(\xi - \xi_0) + \widehat{g}(\xi + \xi_0)\Big]$. Aplicando esto con $g(t) = e^{-\pi(4t)^2}$ y $\xi_0 = 4$, obtenemos $\widehat{f}(\xi) = \big[e^{-\pi(\xi-4)^2/16} + e^{-\pi(\xi+4)^2/16}\big]/2$.

Este resultado nos dice que, en comparación con el caso anterior, la energía de $f(t)$ en el dominio de la frecuencia ya no está concentrada en torno a $\xi = 0$, sino que se ha desplazado hacia $\xi = \pm 4$. Por ello, la condición $|\xi| \leq 5$ que usábamos antes ya no es suficiente para asegurar que la señal esté bien definida dentro de un intervalo de ancho $\nu_s/2 = 5$ cuando $\nu_s = 10$. En cambio, al aumentar la frecuencia de muestreo a $\nu_s = 14$, el ancho permitido es 7, lo que hace que la reconstrucción vuelva a ser precisa.



(a) $\nu_s = 10$ (b) $\nu_s = 14$

Figura 2.2: Reconstrucciones de $f(t) = e^{-\pi(4t)^2}\cos(4\pi t)$

Para simplificar, en lo que sigue vamos a considerar que la frecuencia de muestreo es $\nu_s = 1$; es decir, evaluamos la señal en los números enteros. Además, trabajaremos en el caso en el que la señal $f(t)$ admite una representación en serie de Fourier, la cual es válida bajo ciertas condiciones de regularidad o periodicidad. Por lo tanto, en nuestro caso, una señal $f(t)$ se puede expresar como una suma de exponenciales:

$$f(t) = \sum_{\nu \in \mathcal{F}} a_\nu \, e(\nu t),$$

donde $\mathcal{F}$ es un conjunto discreto de frecuencias.

En este contexto, la modulación de frecuencias se entiende como el proceso de modificar cada coeficiente a_ν en la expansión de Fourier, es decir, multiplicarlo por un factor b_ν que depende de la frecuencia ν. Así, al aplicar un filtro lineal (o multiplicador), la señal se transforma en

$$f(t) \longmapsto \sum_{\nu \in F} a_\nu \, b_\nu \, e(\nu t).$$

En ingeniería, los filtros FIR (Finite Impulse Response) [21] son de gran importancia. Estos filtros se definen a través de su *función de transferencia*

$$H(z) = \sum_{k \in \mathbb{Z}} h_k \, z^{-k},$$

donde los coeficientes h_k tienen soporte finito (existe un N tal que $h_k = 0$ para $|k| > N$). La función $H(z)$ describe el comportamiento del filtro en el dominio z. Cuando se evalúa en $z = e(\nu)$,

se obtiene la respuesta en frecuencia del filtro, lo que nos permite relacionarla con la modulación de frecuencias $b_\nu = H(e(\nu))$.

Si aplicamos el filtro a la señal, obtenemos:

$$f(t) \longmapsto \sum_{\nu\in F} a_\nu \left(\sum_{k\in\mathbb{Z}} h_k\, e(-k\nu)\right) e(\nu t) = \sum_{k\in\mathbb{Z}} h_k \sum_{\nu\in F} a_\nu\, e(\nu(t-k)).$$

Ahora, al considerar la señal muestreada, definimos $x_n = f(n)$, es decir, los valores de la señal en los enteros. Del mismo modo, denotaremos por y_n la señal filtrada en esos puntos. Evaluando en $t = n$, obtenemos:

$$y_n = \sum_{k\in\mathbb{Z}} h_k \sum_{\nu\in F} a_\nu\, e(\nu(n-k))\,.$$

Dado que la suma interna es precisamente la expresión de $f(n-k) = x_{n-k}$, podemos escribir:

$$y_n = \sum_{k\in\mathbb{Z}} h_k\, x_{n-k}.$$

Con esto, hemos comprobado que cuando muestreamos la señal, es decir, tomamos los valores $x_n = f(n)$ para $n \in \mathbb{Z}$, el efecto de aplicar un filtro FIR se traduce en

$$x_n \longmapsto y_n = \sum_{k\in\mathbb{Z}} h_k\, x_{n-k},$$

lo que se conoce en matemáticas como una *convolución* discreta.

Uno de los filtros más fundamentales en el procesamiento de señales es el *filtro paso bajo*, diseñado para eliminar las frecuencias superiores a un cierto umbral, la *frecuencia de corte* $0 < \nu_c < 1/2$. Idealmente, un filtro de este tipo debería cumplir que, al aplicarlo a la señal $x_n = e(\nu n)$, se conserve x_n cuando $|\nu| < \nu_c$, mientras que se anule para $\nu_c < |\nu| \leq 1/2$.

La respuesta al impulso h_k de un filtro se obtiene a partir de su función de transferencia $H(e(\nu))$ mediante la fórmula para los coeficientes de Fourier:

$$h_k = \int_{-1/2}^{1/2} H(e(\nu))\, e(\nu k)\, d\nu.$$

Para un filtro paso bajo ideal, la función de transferencia se define como la función característica del intervalo $[-\nu_c, \nu_c]$, es decir, $H(e(\nu)) = 1$ para $|\nu| \leq \nu_c$ y $H(e(\nu)) = 0$ en el resto del dominio. Sustituyendo esto en la integral, obtenemos:

$$h_k = \int_{-\nu_c}^{\nu_c} e(\nu k)\, d\nu = 2\nu_c \operatorname{sinc}(2\nu_c k).$$

Dado que la función sinc no se anula exactamente para $K \neq 0$, el filtro resultante tiene una cantidad infinita de coeficientes h_k no nulos. Esto contradice la definición de los filtros FIR, que requieren que solo un número finito de coeficientes sean distintos de cero, es decir, que exista un entero N tal que $h_k = 0$ para $|k| > N$.

Para analizar mejor esta situación, consideremos la función periódica $p(x)$ de período 1 que toma el valor 1 en $(-\nu_c, \nu_c)$ y 0 en el resto del intervalo $[-1/2, 1/2]$ (con $|\nu_c| \leq 1/2$). Su serie de Fourier es

$$p(x) = 2\nu_c \sum_{k=-\infty}^{\infty} \operatorname{sinc}(2\nu_c k)\, e(kx).$$

Comparando con la expresión obtenida para h_k, vemos que si definimos los coeficientes del filtro como $h_k = 2\nu_c \operatorname{sinc}(2\nu_c k)$ para $k \in \mathbb{Z}$, la función de transferencia del filtro FIR queda expresada como

$$H(e(\nu)) = \sum_{k=-\infty}^{\infty} h_k\, e(-\nu k) = 2\nu_c \sum_{k=-\infty}^{\infty} \operatorname{sinc}(2\nu_c k)\, e(-\nu k).$$

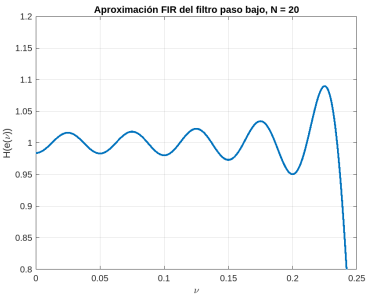
Esta es precisamente la serie de Fourier de $p(x)$, lo que confirma que al elegir los coeficientes del filtro FIR de esta forma, se obtiene un comportamiento en frecuencia equivalente al del filtro paso bajo ideal.

El problema se reduce a determinar unos coeficientes h_k que satisfagan aproximadamente

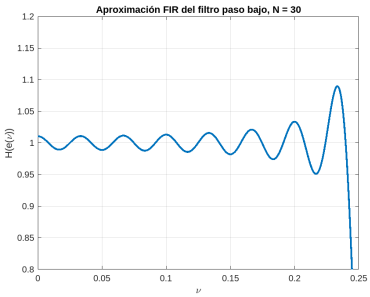
$$H(e(\nu)) = \sum_{k=-N}^{N} h_k\, e(-\nu k). \tag{2.0.2}$$

Por lo anterior, para $N = \infty$, se obtiene la solución exacta con $h_k = 2\nu_c \operatorname{sinc}(2\nu_c k)$. Cabría entonces esperar que si N es grande, obtendremos una buena aproximación con esta elección. Sin embargo, el famoso *fenómeno de Gibbs* implica que esto no es exactamente así: por grande que sea N, siempre se obtiene un pico que alcanza una altura ligeramente mayor que 1,08, aproximadamente un 9 % del máximo de la función característica que queremos aproximar.

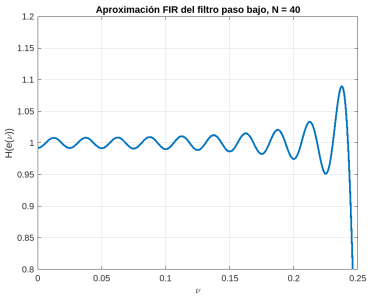
Vamos a dibujar las gráficas en $\nu \in [0, 1/2]$ de (2.0.2) para $N = 20$, 30 y 40 con $h_k = 2\nu_c \operatorname{sinc}(2\nu_c k)$ cuando $\nu_c = 1/4$, corroborando lo dicho sobre el fenómeno de Gibbs. Se limitará el eje Y a $[0{,}8, 1{,}2]$ y el eje X a $[0, 1/4]$ para apreciar mejor el tamaño del máximo.



(a) $N = 20$



(b) $N = 30$



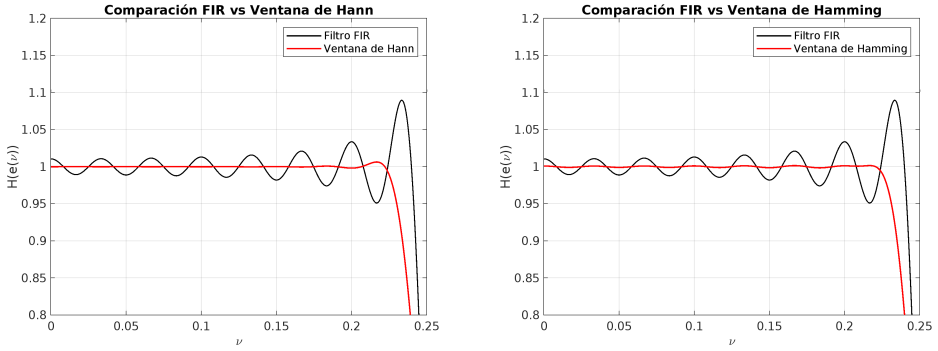
(c) $N = 40$

Para suavizar el efecto de Gibbs, se aplican *ventanas*. Esta técnica consiste en introducir un conjunto de coeficientes w_k que se anulan en el intervalo $|k| > N$ y que se multiplican por los coeficientes ideales h_k que, en teoría, requerirían un número infinito de términos para obtener el filtro exacto. En nuestro caso, se define la respuesta al impulso modificada como $h_k = 2\nu_c w_k \operatorname{sinc}(2\nu_c k)$.

Si no se aplica ninguna modificación, es decir, si se toma $w_k = 1$ para $|k| \leq N$, se está utilizando lo que se conoce como *ventana rectangular*. En contraste, en la práctica se utilizan ventanas más sofisticadas que ayudan a reducir las oscilaciones producidas por el fenómeno de Gibbs. Dos de las ventanas más utilizadas en ingeniería son la *ventana de Hann* y la *ventana de Hamming* que responden, respectivamente, a las fórmulas

$$w_k = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi k}{N}\right) \qquad \text{y} \qquad w_k = \frac{25}{46} + \frac{21}{46}\cos\left(\frac{\pi k}{N}\right) \qquad \text{para } |k| \leq N.$$

Estas ventanas suavizan la transición en la respuesta al impulso y reducen las oscilaciones indeseadas, mejorando así la aproximación al filtro paso bajo ideal.



(a) Ventana de Hann (b) Ventana de Hamming

Figura 2.4: $N = 30$

Imaginemos que tenemos un filtro lineal ideal que se expresa exactamente mediante ciertos coeficientes h_k. Hasta ahora, hemos considerado el caso particular de un filtro paso bajo, pero el mismo enfoque es aplicable a otros tipos de filtros. Queremos aproximar este filtro ideal mediante un filtro FIR, es decir, considerando un número finito de coeficientes N. En este caso, buscamos que

$$H_N(\nu) = \sum_{k=-N}^{N} h_k w_k\, e(-\nu k) \quad \text{se parezca a} \quad H_\infty(\nu) = \sum_{k=-\infty}^{\infty} h_k\, e(-\nu k).$$

En general, la aplicación de una ventana no solo limita el número de coeficientes considerados, sino que también modifica la respuesta en frecuencia del filtro. La siguiente proposición nos muestra cómo la versión truncada $H_N(\nu)$ se relaciona con el filtro ideal $H_\infty(\nu)$ mediante una convolución en el dominio de las frecuencias.

Proposición 2.1. *Sean H_∞ y H_N como antes con w_k una ventana de soporte $|k| \leq N$, se cumple*

$$H_N(\nu) = \int_{-1/2}^{1/2} H_\infty(\nu - t) W(t)\, dt \qquad \textit{donde} \quad W(t) = \sum_{k=-N}^{N} w_k\, e(-kt).$$

Demostración. Sustituimos la expresión $H_\infty(\nu)$ por $H_\infty(\nu - t)$:

$$H_\infty(\nu - t) = \sum_{l=-\infty}^{\infty} h_l\, e(-(\nu - t)l).$$

A continuación, multiplicamos esta expresión por $W(t)$ y tomamos la integral:

$$\int_{-1/2}^{1/2} H_\infty(\nu-t)W(t)\,dt = \int_{-1/2}^{1/2} \sum_{l=-\infty}^{\infty} h_l\, e(-\nu l)\, e(tl)\, W(t)\,dt = \sum_{l=-\infty}^{\infty} h_l\, e(-\nu l) \int_{-1/2}^{1/2} e(tl)\, W(t)\,dt.$$

Si sustituimos $W(t)$ por su definición en la integral, tendremos

$$\int_{-1/2}^{1/2} e(tl) \sum_{k=-N}^{N} w_k\, e(-kt)\,dt = \sum_{k=-N}^{N} w_k \int_{-1/2}^{1/2} e((l-k)t)\,dt.$$

Usando la propiedad de ortogonalidad de las exponenciales en $[-1/2, 1/2]$ se tiene que la integral se anula siempre que $l \neq k$ y vale 1 cuando $l = k$. Por tanto,

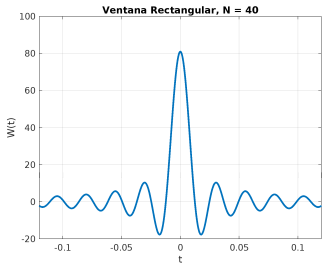
$$\int_{-1/2}^{1/2} H_\infty(\nu - t)W(t)\,dt = \sum_{k=-N}^{N} h_k\, w_k\, e(-\nu k)\,,$$

que es precisamente la definición de $H_N(\nu)$. □

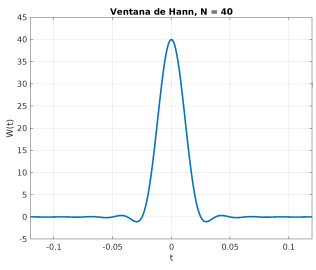
Para que la aproximación sea buena, es importante que W tenga ciertas propiedades. En particular, nos interesa que satisfaga $\int_{-1/2}^{1/2} W = 1$ y que esté concentrada alrededor del origen. Si esto ocurre, la integral $\int_{-1/2}^{1/2} H_\infty(\nu - t)W(t)\,dt$ será aproximadamente igual a $H_\infty(\nu)$, logrando así una buena aproximación.

Esta condición de normalización $\int_{-1/2}^{1/2} W = 1$ se cumple para las tres ventanas consideradas. Sin embargo, el problema de la ventana rectangular es que W está demasiado dispersa, lo que introduce oscilaciones no deseadas en $H_N(\nu)$.

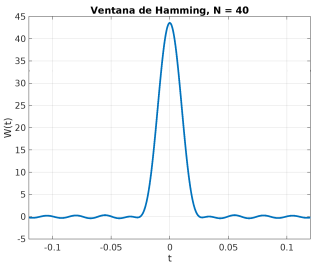
A continuación, representamos gráficamente la función $W(t)$ con $N = 40$ para las ventanas rectangular, de Hann y de Hamming. Estas gráficas ilustran cómo la ventana rectangular produce una función $W(t)$ más extendida, mientras que las ventanas de Hann y Hamming logran concentrar mejor su 'masa" cerca del origen, reduciendo la dispersión y minimizando los efectos indeseados en la aproximación del filtro FIR.



(a) Ventana regular



(b) Ventada de Hann



(c) Ventana de Hamming

Figura 2.5: $N = 40$

CAPÍTULO 3
Teoría de la información

Cuando se habla de información, es probable que se piense en la cantidad de conocimiento que obtenemos al observar un evento o recibir un mensaje. Si algo es completamente predecible, no nos aporta información nueva; en cambio, si el resultado es dudoso, al observarlo reducimos nuestra incertidumbre y obtenemos información.

Claude Shannon, en su famoso artículo [12], buscó una manera formal de medir esta incertidumbre. Su idea era encontrar una función que cuantificara cuánto "desorden" hay en un conjunto de eventos posibles y, en consecuencia, cuánto podemos aprender de ellos. A esta medida la llamó entropía. Demostró el siguiente resultado, que indica que solo hay una forma de definirla a partir de unas propiedades básicas.

Teorema (Teorema de Shannon)**.** *Sea H una función definida sobre distribuciones de probabilidad finitas $(p_1, \dots, p_n)$, con $p_j \geq 0$ y $\sum_{j=1}^{n} p_j = 1$, que satisface las siguientes propiedades:*

i) ***Continuidad:*** *H es continua respecto a las probabilidades p_j.*

ii) ***Monotonía:*** *La sucesión $\{\eta(n)\}_{n=1}^{\infty}$, donde $\eta(n) := H\left(\frac{1}{n}, \dots, \frac{1}{n}\right)$, es creciente.*

iii) ***Aditividad:*** *Para cualquier $a_j \in_{\geq 0}$ con $\sum_{j=1}^{k} a_j = n$, se cumple:*

$$\eta(n) = H\left(\frac{a_1}{n}, \dots, \frac{a_k}{n}\right) + \sum_{j=1}^{k} \frac{a_j}{n} \eta(a_j).$$

Entonces, existe una constante $K > 0$ tal que:

$$H(p_1, \dots, p_n) = -K \sum_{j=1}^{n} p_j \log p_j. \tag{3.0.1}$$

En particular, si se toma $K = 1/\log 2$, se obtiene la expresión clásica en bits:

$$H(p_1, \dots, p_n) = -\sum_{j=1}^{n} p_j \log_2 p_j.$$

La continuidad, nos dice que pequeños cambios en las probabilidades generan pequeños cambios en la entropía. La segunda propiedad es natural porque más opciones generan más incertidumbre. La aditividad dice que si una elección se puede descomponer en varios pasos, la entropía total es la suma de las entropías de cada paso.

Demostración. Sea H una función que cumple las tres propiedades del enunciado. Sea $\eta(n) := H\left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$. Aplicamos la propiedad de aditividad a una partición uniforme: supongamos que $n = km$, y tomamos $a_j = m$ para todo $j = 1, \ldots, k$. Entonces:

$$\eta(n) = H\left(\frac{1}{k}, \ldots, \frac{1}{k}\right) + \sum_{j=1}^{k} \frac{1}{k}\eta(m) = \eta(k) + \eta(m).$$

Es decir, $\eta(km) = \eta(k)+\eta(m)$. Además, como $\eta(n)$ es continua y creciente por hipótesis, el único tipo de soluciones posibles son de la forma $\eta(n) = K \log n$ para alguna constante real $K > 0$ y logaritmo en base arbitraria.

Supongamos ahora que tenemos una distribución de probabilidad dada por fracciones: $p_j = \frac{a_j}{n}$, con $a_j \in \mathbb{Z}_{\geq 0}$ y $\sum a_j = n$. Por la propiedad aditiva, se cumple:

$$\eta(n) = H\left(\frac{a_1}{n}, \ldots, \frac{a_k}{n}\right) + \sum_{j=1}^{k} \frac{a_j}{n}\eta(a_j).$$

Sustituyendo $\eta(n) = K \log n$ y $\eta(a_j) = K \log a_j$, obtenemos:

$$H\left(\frac{a_1}{n}, \ldots, \frac{a_k}{n}\right) = K \log n - \sum_{j=1}^{k} \frac{a_j}{n} K \log a_j = -K \sum_{j=1}^{k} \frac{a_j}{n} \log\left(\frac{a_j}{n}\right).$$

Esto es:

$$H(p_1, \ldots, p_k) = -K \sum_{j=1}^{k} p_j \log p_j.$$

Sea $(p_1, \ldots, p_n)$ una distribución de probabilidad con números reales. Dado que los racionales son densos en los reales y H es continua, podemos aproximar cada p_j por una sucesión de racionales. Entonces:

$$H(p_1, \ldots, p_n) = \lim_{m\to\infty} H(p_1^{(m)}, \ldots, p_n^{(m)}) = \lim_{m\to\infty} \left(-K \sum_{j=1}^{n} p_j^{(m)} \log p_j^{(m)}\right) = -K \sum_{j=1}^{n} p_j \log p_j.$$

Por tanto, toda función H que cumple las tres propiedades debe tener la forma:

$$H(p_1, \ldots, p_n) = -K \sum_{j=1}^{n} p_j \log p_j.$$

Si además tomamos $K = 1/\log 2$, se obtiene la entropía en bits que indica el enunciado. □

La teoría de la información, desarrollada por Shannon, busca determinar hasta qué punto se puede comprimir un mensaje sin perder información. Para ello, se considera un espacio de probabilidad $S = \{s_1, s_2, \ldots, s_n\}$, donde cada símbolo s_j tiene una probabilidad p_j, cumpliendo $\sum_j p_j = 1$. La entropía $H(S)$ mide la cantidad de información presente en la distribución de probabilidades de S.

Para transmitir estos símbolos de manera eficiente, se utilizan *códigos binarios prefijos*. Estos son funciones $C : S \longrightarrow \mathcal{B}$ donde $\mathcal{B}$ es el conjunto de cadenas finitas de *bits* (listas de ceros y unos) tal que ningún $C(s_i)$ sea "prefijo" de un $C(s_j)$, es decir, no existe $i \neq j$ tal que $C(s_j)$ comience con $C(s_i)$. Esta propiedad garantiza que la codificación sea única y descifrable sin ambigüedades.

Por ejemplo, si se asignan los códigos $C(\texttt{a}) = 1011$ y $C(\texttt{b}) = 010$, entonces la secuencia $C(\texttt{a}, \texttt{b}) = C(\texttt{ab})$ se codifica como 1011010. Si la condición de prefijo no se cumple, pueden surgir problemas en la decodificación. Por ejemplo, si $C(\texttt{a}) = 0$, $C(\texttt{b}) = 1$ y $C(\texttt{c}) = 01$, no se cumple la condición de prefijo porque $C(\texttt{a})$ es prefijo de $C(\texttt{c}) = 01$ y no sabríamos si 101 significa `bc` o `bab`.

Estos códigos prefijo cumplen que $C : S^N \longrightarrow \mathcal{B}$ es inyectiva porque cada secuencia de N símbolos en S^N se codifica de manera única en una cadena de bits, sin posibilidad de que dos secuencias distintas produzcan el mismo código.

Esto se debe a la propiedad de prefijo: como ningún código es prefijo de otro, al leer una cadena de bits de izquierda a derecha, podemos identificar cada símbolo de manera única sin ambigüedad, garantizando que la codificación es reversible y que no hay dos secuencias de S^N que correspondan a la misma cadena en $\mathcal{B}$.

Además, las imágenes de C para distintos valores de N son disjuntas porque las cadenas de bits generadas por secuencias de diferente longitud nunca se confunden entre sí.

Por esta razón, los códigos prefijo permiten siempre recuperar la secuencia original sin necesidad de separadores entre los códigos y garantizan una decodificación única y sin errores.

Si bien existen códigos descifrables que no cumplen la propiedad de prefijo, se puede demostrar que, en cierto sentido, siempre es posible transformarlos en códigos prefijo sin perder la capacidad de descifrado. Para simplificar la exposición, a partir de este punto se utilizará el término código como sinónimo de código binario prefijo. Además, vamos a suponer que todas las probabilidades cumplen $p_j \neq 0$, ya que no tiene sentido asignar codificaciones a símbolos que nunca aparecen en el mensaje.

Definición 3. *Dado un código binario prefijo $C : S \longrightarrow \mathcal{B}$, se define la **longitud** de una cadena de bits como el número total de bits que contiene. Para denotar esta cantidad, utilizamos la notación $|x|$, donde x es una secuencia de bits. Por ejemplo: $|110| = 3$, $|01| = 2$.*

Definición 4. *La **longitud media** de un código mide el número esperado de bits utilizados para representar un símbolo del conjunto S. Se define como:*

$$L(C) = \sum_{j=1}^{n} p_j \big|C(s_j)\big|,$$

donde p_j es la probabilidad de aparición del símbolo s_j, y $|C(s_j)|$ es la longitud de su codificación en bits.

De manera análoga, si consideramos la codificación de secuencias de longitud N en S^N, la ***longitud media del código en*** S^N *se define como:*

$$L_N(C) = \sum_{j_1,\dots,j_N=1}^{n} p_{j_1} \cdots p_{j_N} \big| C(s_{j_1}, \dots, s_{j_N}) \big|.$$

Estas definiciones tienen sentido porque la longitud de una secuencia codificada depende de la suma de las longitudes de los códigos de sus elementos, teniendo en cuenta con qué frecuencia aparece cada secuencia. Dicho de otra manera, los símbolos que aparecen más a menudo influyen más en la longitud media del código.

Estos conceptos de longitud y longitud media serán fundamentales en el estudio de la compresión de datos, ya que nos ayudan a medir qué tan eficiente es un código en términos de la cantidad de bits necesarios para transmitir la información.

Supongamos que tenemos el conjunto de símbolos $S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ con $p_1 = \frac{1}{2}$, $p_2 = p_3 = \frac{1}{4}$. Consideremos tres posibles códigos para representar estos símbolos en binario:

	a	b	c
C	00	01	10

	a	b	c
C'	10	0	11

	a	b	c
C''	0	10	11

El código C es el más directo, ya que asigna a cada símbolo un número consecutivo en binario con la misma cantidad de bits. Sin embargo, este método no es óptimo porque no tiene en cuenta la frecuencia de los símbolos: todos tienen una representación de la misma longitud, a pesar de que a es mucho más común que b y c.

Los códigos C' y C'' son más eficientes porque asignan representaciones más cortas a los símbolos más frecuentes. En particular, C'' es el mejor de los tres, ya que da la codificación más corta al símbolo más frecuente (a), lo que reduce la longitud media total del código. Este principio es la base de la compresión de datos: al reducir la cantidad de bits usados para los símbolos más comunes, se disminuye el tamaño total del mensaje, logrando una representación más eficiente de la información.

Un cálculo muestra $L(C) = 2$, $L(C') = 7/4$, $L(C'') = 3/2$ corroborando $L(C) > L(C') > L(C'')$.

El problema central en la compresión de datos es encontrar una codificación que minimice la longitud media de los mensajes, garantizando que la información se represente de la manera más eficiente posible. Shannon estableció límites fundamentales para la compresión, aunque la solución exacta a este problema fue desarrollada más tarde por Huffman. Estos límites, conocidos en inglés como el *source coding theorem* (*teorema de codificación de fuentes*), establecen cotas superior e inferior para la longitud media de cualquier código óptimo.

Teorema 3.1. *Sea S como antes y $H(S)$ la entropía asociada a sus probabilidades, entonces $H(S) \leq \min_C L(C) < H(S) + 1$ donde el mínimo es sobre todos los códigos C definidos en S.*

Un resultado aún más preciso se obtiene al considerar secuencias de longitud N en lugar de símbolos individuales. En este caso, la longitud media de un código óptimo definido sobre S^N sigue la relación:

Corolario 3.2. *Para cualquier $N \in \mathbb{Z}^+$ se tiene $NH(S) \leq \min_C L_N(C) < NH(S)+1$ donde el mínimo es sobre todos los códigos C definidos en S^N. En particular, cuando N tiende a infinito, $N^{-1}\min_C L_N(C) \to H(S)$.*

Si calculamos la entropía de $S = \{a, b, c\}$ con $p_1 = \frac{1}{2}$, $p_2 = p_3 = \frac{1}{4}$, se tiene

$$H(S) = -\sum_{j=1}^{3} p_j \log_2 p_j = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = \frac{3}{2},$$

que es exactamente igual que la longitud media de C'', es decir, $H(S) = L(C'')$. Por lo tanto, según el teorema 3.1 este código tiene la mínima longitud media posible.

A continuación, vamos a considerar un ejemplo numérico que muestre el significado de estos resultados. Supongamos que tenemos un fichero con un texto muy largo compuesto únicamente por dos caracteres, A y B, que aparecen aleatoriamente con frecuencias del 80 % y 20 %, respectivamente. Es decir, el conjunto de símbolos es $S = \{A, B\}$ con $p_1 = 0{,}8$ y $p_2 = 0{,}2$. La entropía de esta fuente es aproximadamente 0,72 bits por símbolo.

En la práctica, los ordenadores suelen codificar cada letra con 8 bits, por lo que un texto de T letras ocuparía $8T$ bits. Si codificamos cada letra de forma individual con la asignación natural $C(A) = 0$ y $C(B) = 1$, todo el texto ocupará T bits en memoria, lo que significa que la longitud media del código es $L(C) = 1$ bit por símbolo, que cuadra con los límites establecidos por el teorema 3.1.

Ahora, si agrupamos los caracteres en bloques de longitud N, el texto se divide en T/N bloques. Según el corolario, con un código adecuado cada bloque se representará con una longitud entre $0{,}72N$ y $0{,}72N + 1$ bits. Por lo tanto, el tamaño total del texto estará entre $0{,}72T$ y $(0{,}72 + 1/N)T$ bits. A medida que N crece (y T es lo suficientemente grande para que tenga sentido el promedio), el límite de la longitud media por símbolo se aproxima a 0,72 bits, lo que demuestra que la entropía determina el factor óptimo de compresión.

Según el esquema planteado, vamos a escribir un programa que genere un texto de 10^6 caracteres, compuesto únicamente por las letras A y B aleatoriamente con frecuencias del 80 % y 20 %, respectivamente.

Al analizar las propiedades del archivo generado, observamos que su tamaño es de 10^6 bytes, lo que equivale a 8 millones de bits. Sin embargo, al comprimir el fichero utilizando el formato `ZIP`, su tamaño se reduce aproximadamente a 135.000 bytes, es decir, poco más de 1 millón de bits. Este resultado concuerda con el Corolario 3.2, ya que la cantidad de bits utilizada después de la compresión sigue siendo mayor que $10^6 H(S)$ bits, donde:

$$10^6 H(S) = 10^6 \times 0{,}72 = 720{,}000 \text{ bits.}$$

Si modificamos el programa para generar un texto de 10^6 caracteres, pero esta vez compuesto por 10 caracteres diferentes, cada uno con probabilidad 1/10, veremos que el archivo comprimido, se acerca aún más a los $10^6 H(S)$ bits.

Este resultado se debe a que la entropía del sistema es mayor en este caso, ya que al tener los 10 caracteres la misma probabilidad, la incertidumbre sobre qué símbolo aparecerá es máxima. En términos de la teoría de la información, esto implica que la cantidad mínima de bits necesarios para representar el mensaje también será mayor, y la compresión reflejará mejor el límite teórico establecido por la entropía.

Ahora, vamos a ver por qué el corolario 3.2 es, en efecto, un corolario directo del teorema 3.1. Para ello, basta ver que la entropía $H(S^N)$ es simplemente N veces la entropía de un símbolo individual, es decir, $H(S^N) = NH(S)$. Empecemos recordando la defincón de la entropía de S^N:

$$H(S^N) = -\sum_{j_1,\dots,j_N=1}^{n} p_{j_1}\dots p_{j_N}\,\log_2(p_{j_1}\dots p_{j_N}),$$

que aplicando propiedades de los logaritmos y reordenando los sumatorios queda

$$H(S^N) = -\sum_{j_1,\dots,j_N=1}^{n} p_{j_1}\dots p_{j_N}\sum_{l=1}^{N}\log_2 p_{j_l} = -\sum_{l=1}^{N}\sum_{j_1,\dots,j_N=1}^{n} p_{j_1}\dots p_{j_N}\,\log_2 p_{j_l}.$$

A continuación, simplemente vamos a fijar un símbolo s_m y vamos a considerar el coeficiente de $\log_2 p_m$ en la suma. Esto es, para cada l consideramos todos los términos en los que $j_l = m$. La suma interna es:

$$\sum_{\substack{j_1,\dots,j_N=1\\ j_l=m}}^{n} p_{j_1}\dots p_{j_N} = p_m\Big(\sum_{j=1}^{N} p_j\Big)^{N-1} = p_m.$$

Esto se cumple para cada l de las N posiciones que hay, por lo que la suma en l de las cantidades de la izquierda es Np_m.

Finalmente, si sustituimos en la definición de entropía, se tiene

$$H(S^N) = -\sum_{m=1}^{N} Np_m\log_2 p_m \;= NH(S).$$

Por último, haremos la demostración del teorema 3.1, para la cual ultilizaremos el siguiente resultado:

Lema 3.3 (Desigualdad de Kraft)**.** *Dado un código sobre $S = \{s_1,\dots,s_n\}$, las longitudes de las codificaciones $l_j = \big|C(s_j)\big|$ satisfacen $\sum_{j=1}^{n} 2^{-l_j} \le 1$. Recíprocamente, dados $l_j \in \mathbb{Z}^+$ cualesquiera que verifiquen esta desigualdad, siempre existe un código C tal que $l_j = \big|C(s_j)\big|$.*

Demostración. Un código binario prefijo puede representarse mediante un árbol binario, en el que cada símbolo del alfabeto se asocia a una hoja del árbol. En esta representación, cada bit de un código indica un paso en el recorrido desde la raíz: por ejemplo, un 0 puede corresponder a un movimiento hacia la izquierda, y un 1 a un movimiento hacia la derecha. Dado que se trata de un código prefijo, ningún código asignado a un símbolo es prefijo del código de otro. Esto significa que, al recorrer el árbol, cada camino desde la raíz hasta una hoja corresponde a un único símbolo y no se superpone con los caminos de otros símbolos. Como resultado, el árbol es

completo, lo que garantiza que la decodificación pueda realizarse de forma inmediata leyendo los bits uno a uno hasta alcanzar una hoja del árbol.

El número total de nodos en cada nivel k de un árbol binario es, como máximo, 2^k. Cada código de longitud l_j se encuentra en el nivel l_j del árbol. Esto significa que, al asignar códigos a los símbolos, estamos ocupando cierto número de hojas en distintos niveles del árbol binario. Cada código de longitud l_j ocupa exactamente una hoja en el nivel l_j. Como hay un máximo de 2^{l_j} posiciones posibles en el nivel l_j, podemos interpretar $2^{l_n-l_j}$ como las posiciones que ocupa cada código de longitud l_j dentro del árbol. Si el código es prefijo, ninguna otra secuencia puede ocupar las hojas descendientes de un código ya asignado, lo que implica que la cantidad total de nodos ocupados no puede ser mayor que la cantidad de nodos disponibles en el nivel l_j. Es decir, la cantidad total de hojas utilizadas debe ser menor o igual que 2^{l_n}:

$$\sum_{j=1}^{n} 2^{l_n-l_j} \leq 2^{l_n} \quad \Rightarrow \quad \sum_{j=1}^{n} 2^{-l_j} \leq 1.$$

Ahora, veamos el recíproco. Si se nos da un conjunto de longitudes $l_1, l_2, \ldots, l_n$ que cumplen $\sum_{j=1}^{n} 2^{-l_j} \leq 1$, entonces es posible construir un código prefijo con esas longitudes.

La idea de la construcción es sencilla:

1. Ordenamos los símbolos según sus longitudes l_j.
2. Escogemos la primera palabra del código en el nivel l_1, que corresponde a seleccionar un nodo en el nivel l_1 del árbol. Como estamos construyendo un código binario prefijo, todas las palabras que comienzan con este nodo quedan excluidas. En total, eliminamos $2^{l_n-l_1}$ hojas de las disponibles en el nivel l_1.
3. Buscamos el siguiente nodo disponible en el nivel l_2 y lo seleccionamos como la segunda palabra del código. Como antes, eliminamos el subárbol que tiene este nodo como raíz, lo que bloquea otras palabras que compartirían este prefijo. Esto eliminaría $2^{l_n-l_2}$ hojas más.
4. Continuamos este proceso hasta elegir la n palabras del código, asegurándonos de que cada elección elimine $2^{l_n-l_j}$ hojas del nivel más profundo.

Inicialmente, el árbol tiene un total de 2^{l_n} hojas en el nivel l_n y durante el proceso, eliminamos un total de $\sum_{j=1}^{n} 2^{l_n-l_j}$. Como esta suma es menor o igual a 2^{l_n} por la hipótesis de Kraft, siempre habrá suficiente espacio en el árbol para asignar los códigos sin que uno sea prefijo de otro. □

Demostración teorema 3.1. Consideremos la función

$$F(x_1, \ldots, x_n) = -\sum_{j=1}^{n} p_j \log_2 x_j.$$

Esta función representa el valor esperado de $-\log_2 x_j$ bajo la distribución p_j. Vamos a minimizar $F(x_1, \ldots, x_n)$ en el conjunto

$$S = \left\{ \vec{x} \in (0,1)^n \, : \, \sum_{j=1}^{n} x_j = 1 \right\}.$$

Para ello, usaremos los multiplicadores de Lagrange. Definimos la función de Lagrange:

$$\mathcal{L}(x_1, \ldots, x_n, \lambda) = -\sum_{j=1}^{n} p_j \log_2 x_j \; + \; \lambda\big(\sum_{j=1}^{n} x_j - 1\big).$$

Derivamos con respecto a x_j e igualamos a cero:

$$\frac{\delta \mathcal{L}}{\delta x_j} = -p_j \, \frac{1}{x_j \log 2} + \lambda = 0 \qquad \Rightarrow \qquad \lambda = \frac{p_j}{x_j \log 2}.$$

Esto debe cumplirse para todos los valores de j, lo que implica que el cociente $\frac{p_j}{x_j}$ es constante para todos los j. Es decir, existe una constante C tal que $x_j = Cp_j$. Para encontrar C, usaremos la restricción S:

$$\sum_{j=1}^{n} x_j = 1 \quad \Rightarrow \quad \sum_{j=1}^{n} Cp_j = 1 \quad \Rightarrow \quad C = 1.$$

Se obtiene entonces que el mínimo de $F(x_1, \ldots, x_n)$ en S se alcanza para $x_j = p_j$, es decir, en $F(p_1, \ldots, p_n) = H(S)$.

Recordemos que, dado un código binario prefijo con n símbolos, denotamos por l_j la longitud del código asignado al símbolo j. Es decir, l_j indica el número total de bits que contiene un código.

Definamos $\sigma = \big(\sum_{j=1}^{n} 2^{-l_j}\big)^{-1}$, como el mínimo se alcanza en $(p_1, \ldots, p_n)$, sabemos que

$$H(S) = F(p_1, \ldots, p_n) \leq F(2^{-l_1}\sigma, \ldots, 2^{-l_n}\sigma).$$

Como $\sigma \leq 1$, por propiedades del logaritmo y de la función $F(x_1, \ldots, x_n)$, se cumple que:

$$F(2^{-l_1}\sigma, \ldots, 2^{-l_n}\sigma) \leq F(2^{-l_1}, \ldots, 2^{-l_n}) - \sum_{j=1}^{n} p_j \log_2 2^{-l_j} = \sum_{j=1}^{n} p_j l_j = L(C).$$

Por lo tanto, hemos obtenido la cota inferior del teorema, $H(S) \leq L(C)$. Ahora, vamos a obtener la cota superior $L(C) \leq H(S) + 1$.

Para cada j, elegimos l_j como un entero que satisface la desigualdad $-\log_2 p_j \leq l_j < 1 - \log_2 p_j$. Esto implica que $-l_j \leq \log_2 p_j$ y, a su vez, $2^{-l_j} \leq p_j$. Sumando sobre todos los j, obtenemos:

$$\sum_{j=1}^{n} 2^{-l_j} \leq \sum_{j=1}^{n} p_j = 1.$$

Por el lema 3.3, esto nos garantiza la existencia de un código prefijo con longitudes l_j cuya longitud media sería $L(C) = \sum_{j=1}^{n} p_j l_j$. Por la cota superior para l_j:

$$L(C) < \sum_{j=1}^{n} p_j(1 - \log_2 p_j) = \sum_{j=1}^{n} p_j - \sum_{j=1}^{n} \log_2 p_j = H(S) + 1.$$

Lo que concluye la prueba. □

CAPÍTULO 4
Aplicaciones

En este capítulo veremos dos aplicaciones relevantes sobre algunas cosas que hemos visto en los capítulos anteriores. La primera está relacionada con la compresión de imágenes digitales, concretamente con el formato JPEG, que es uno de los más utilizados en dispositivos electrónicos. Este formato combina transformadas como la DCT (vista en el capítulo 1) con técnicas de compresión óptima (vista en el capítulo 3). La segunda aplicación aborda un problema clásico de reconstrucción en tomografía a partir de proyecciones, en un caso simplificado donde es posible utilizar técnicas de análisis de Fourier y series trigonométricas para obtener aproximaciones numéricas.

4.1. El formato de imágenes JPEG

El formato JPEG es uno de los más utilizados para almacenar imágenes digitales. Su proceso de compresión incluye varias etapas, en una de ellas se emplea una técnica de compresión sin pérdida: la codificación de Huffman. Esta se utiliza para representar de manera eficiente los datos obtenidos tras la cuantificación, asignando códigos binarios más cortos a los valores más frecuentes. El procedimiento es el siguiente:

1. Se parte de un conjunto de símbolos, cada uno con una probabilidad de aparición.
2. Se seleccionan los dos símbolos menos probables y se agrupan en un nodo nuevo con la suma de sus probabilidades.
3. Se repite este proceso con la nueva lista, tratando los nodos agrupados como nuevos símbolos.
4. Se construye así un árbol binario, donde cada símbolo está en una hoja.
5. Para codificar cada símbolo, se sigue el camino desde la raíz hasta su hoja, asignando 0 a las ramas izquierdas y 1 a las derechas (o viceversa).

Una característica fundamental de este método es que el código resultante es prefijo, es decir, ningún código es el inicio de otro, lo cual permite una decodificación sin ambigüedad. Aunque puede haber múltiples árboles de Huffman válidos (cuando hay empates de probabilidad), todos ellos generan la misma longitud media.

Para entender mejor este método, veamos un ejemplo practico. Sea $S = \{s_1, \ldots, s_8\}$ un conjunto con probabilidades dadas por esta tabla:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Prob.	2/5	7/40	1/10	1/10	1/10	1/20	1/20	1/40

Se construye el árbol de Huffman agrupando sucesivamente los símbolos con menor probabilidad. A partir del árbol, se asignan los códigos binarios a cada símbolo según el recorrido desde la raíz. El resultado se puede visualizar en el siguiente diagrama:

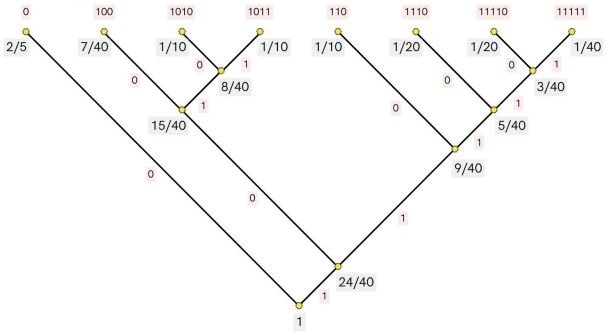


Figura 4.1: Diagrama de árbol

Símbolo	Código Huffman
s_1	0
s_2	100
s_3	1010
s_4	1011
s_5	110
s_6	1110
s_7	11110
s_8	11111

Tabla 4.1: Código

De este árbol se obtiene la codificación binaria óptima para cada símbolo indicada a la derecha. Este código es un ejemplo de una posible construcción del árbol. Dependiendo del orden de agrupación entre símbolos con la misma probabilidad, pueden aparecer códigos distintos pero con la misma longitud media.

Ahora vamos a centrarnos en cómo funciona el proceso de compresión JPEG en el caso concreto de las imágenes en blanco y negro. En este tipo de imágenes, cada píxel representa un nivel de intensidad (de negro a blanco), y la compresión se basa en eliminar información redundante o poco perceptible para el ojo humano. Así se consigue reducir el tamaño del archivo sin una pérdida de calidad apreciable.

El procedimiento consiste en las siguientes etapas:

1. **División en bloques de 8×8 píxeles:** La imagen se fragmenta en bloques de 64 píxeles. Este tamaño facilita la aplicación local de las transformaciones posteriores, permitiendo una compresión eficiente adaptada al contenido de cada región de la imagen.

2. **Transformada Discreta del Coseno (DCT):** A cada bloque se le aplica la DCT bidimensional, que transforma los valores de intensidad (espacio) en coeficientes de frecuencia. El primer coeficiente representa la media del bloque (frecuencia más baja), mientras que los restantes reflejan los detalles y variaciones (frecuencias más altas). En general, la mayor parte de la información visual está contenida en los primeros coeficientes.

3. **Cuantificación:** Este paso introduce la compresión con pérdida. Los coeficientes obtenidos de la DCT se dividen por una tabla de cuantificación y se redondean. Esto elimina muchas frecuencias altas, que son las menos relevantes perceptualmente porque suelen tener

coeficientes pequeños, reduciendo así la cantidad de datos necesarios para representar la imagen.

4. **Codificación sin pérdida (Huffman):** Finalmente, los coeficientes cuantificados se codifican utilizando métodos sin pérdida, como la codificación run-length (para secuencias largas de ceros) y la codificación de Huffman, que asigna códigos más cortos a los valores más frecuentes. Esta fase permite reducir aún más el tamaño del archivo resultante.

El archivo comprimido en formato JPEG puede descomprimirse reconstruyendo los bloques mediante la DCT inversa, obteniendo una imagen visualmente muy similar a la original, aunque no idéntica debido a la cuantificación. Este enfoque permite reducir aún más el tamaño del archivo, manteniendo una apariencia visual muy cercana a la original.

Para comprender mejor cómo funciona el proceso de compresión JPEG en la práctica, vamos a ver algunos experimentos sencillos. A través de estos ejemplos visuales, observaremos cómo ciertos aspectos de la imagen influyen en la eficiencia de la compresión.

El primer experimento consiste en analizar cómo la posición de un objeto dentro de una imagen afecta al tamaño del archivo JPEG resultante. Para ello se ha creado una imagen de tamaño 256×256 píxeles, completamente blanca, sobre la cual se ha insertado un cuadrado negro de 128×128 píxeles. Este cuadrado se ha colocado en distintas posiciones, variando las coordenadas del vértice superior izquierdo.

Se han considerado las siguientes posiciones:

$$(x_0, y_0) \in \{(0,0),\ (32,32),\ (33,32),\ (32,33),\ (33,33),\ (40,40)\}.$$

Para cada caso se ha generado la imagen correspondiente utilizando MATLAB y se ha guardado en formato JPEG. La siguiente tabla recoge el tamaño en bytes de cada archivo:

Posición (x_0, y_0)	**Tamaño (bytes)**
(0, 0)	1.146
(32, 32)	1.148
(33, 32)	1.628
(32, 33)	1.651
(33, 33)	2.186
(40, 40)	1.148

Los resultados muestran que el tamaño del archivo JPEG varía en función de la posición del cuadrado negro. Cuando este está completamente alineado con los bloques de 8×8 píxeles (por ejemplo, en las posiciones $(0,0)$, $(32,32)$ o $(40,40)$), el tamaño del archivo es menor. En cambio, cuando el cuadrado está desalineado (como en la posición $(33,33)$), la compresión resulta menos eficiente y el archivo generado ocupa más espacio.

Esta diferencia puede explicarse observando cómo actúa la compresión del formato JPEG. En informática, el píxel $(0,0)$ corresponde a la esquina superior izquierda de la imagen, y la compresión JPEG se basa en dividir la imagen en bloques de 8×8 píxeles. Si un objeto (como el

cuadrado negro) se alinea con esos bloques, cada bloque contiene zonas homogéneas que pueden comprimirse de forma muy eficiente. Sin embargo, si el cuadrado no está alineado, sus bordes atraviesan los bloques y hacen que dentro de ellos haya más cambios de valores entre píxeles.

Esto hace que, al aplicar la DCT, se obtengan coeficientes más variados dentro de los bloques, lo que complica la cuantificación y hace que la compresión sea menos eficaz. Por tanto, aunque las imágenes se vean casi iguales, su estructura interna puede influir mucho en el tamaño del archivo JPEG.

Comencemos el segundo experimento. Con este, comprenderemos mejor cómo afecta la pérdida parcial de información a distintos tipos de imágenes, se han realizado dos muestras: una fotografía con transiciones más suaves (una imagen del Templo de Diana, en Mérida) y una imagen artificial con bordes muy definidos (una estrella gris sobre fondo blanco). En ambos casos se ha trabajado con imágenes en escala de grises y se ha aplicado el mismo proceso con tres niveles distintos de pérdida: $p = 0{,}2$, $p = 0{,}5$ y $p = 0{,}8$.

En cada imagen se han comparado tres métodos de eliminación parcial de datos:

1. Eliminación de coeficientes DCT de menor magnitud según el cuantil correspondiente a p.
2. Píxeles al azar a cero: Se eliminan píxeles de forma aleatoria con probabilidad p.
3. Píxeles más oscuros a cero: Se eliminan los píxeles con menor valor de intensidad, según p.

La primera imagen utilizada es una fotografía en escala de grises de un templo, con variaciones suaves de luz y textura. Los resultados obtenidos en D.1 para los diferentes valores de p muestran lo siguiente:

- Con la DCT, incluso para valores altos como $p = 0{,}8$, la imagen sigue siendo claramente reconocible. La pérdida afecta principalmente a los detalles finos, pero la estructura general se mantiene.
- Al eliminar píxeles al azar, la imagen se degrada mucho más rápido. Ya con $p = 0{,}5$ comienza a aparecer ruido, y con $p = 0{,}8$ resulta difícil identificar la imagen.
- Al eliminar los píxeles más oscuros, las zonas en sombra desaparecen. Aunque el contorno general de la imagen se mantiene más que en el caso aleatorio, se pierde mucha información.

La segunda imagen es una figura de una estrella gris centrada sobre fondo blanco. A diferencia de la anterior, esta imagen contiene bordes abruptos y zonas homogéneas bien definidas. En este caso, los resultados son distintos:

- Con la DCT, la pérdida de coeficientes se nota más en la silueta de la figura.
- En el caso aleatorio, la imagen se llena de puntos negros dispersos que rompen la forma de la estrella, aunque se sigue reconociendo hasta cierto punto.
- Al eliminar los píxeles más oscuros, la estrella se va volviendo negra, lo que tiene sentido dado que es la región más oscura de la imagen.

Al hacer el experimento con la imagen de la estrella, podemos ver en E.1 que los resultados del tercer método son algo distintos al resto. Vemos que para $p = 0,2$, la estrella oscurece pero para

los valores más altos, la imagen aparece completamente negra. Esto ocurre porque la mayoría de los píxeles de la imagen pertenecen al fondo blanco, y solo una pequeña parte corresponde a la estrella. Al calcular el umbral con un cuantil sobre todos los píxeles, este resulta muy alto, por lo que los tonos del fondo se eliminan de golpe.

Este experimento confirma que el método basado en DCT es especialmente eficaz cuando la imagen tiene transiciones suaves, como sucede en fotografías reales. Sin embargo, cuando la imagen contiene bordes nítidos, la pérdida de información en la frecuencia puede afectar de forma significativa. En cambio, eliminar píxeles sin tener en cuenta la estructura (como en el caso aleatorio) tiende a generar ruido, y eliminar según la intensidad puede ser útil solo si el rango de valores está bien distribuido.

4.2. Tomografías radiales

Las tomografías axiales computarizadas (TAC) permiten observar el interior de un cuerpo mediante rayos X. La idea fundamental es que estos rayos se debilitan al atravesar distintos materiales, y esa atenuación depende de la densidad del objeto en cada punto. Matemáticamente, se puede modelar esta densidad como una función $\rho : \mathbb{R}^2 \to \mathbb{R}$, donde $\rho(x, y)$ representa la densidad en la posición (x, y).

Cuando tomamos una radiografía desde una dirección concreta, como por ejemplo desde arriba (en dirección vertical), lo que se recoge es, en cada posición horizontal x, la integral de la densidad a lo largo de la línea vertical que pasa por ese punto. Definimos así la función:

$$S(x) = \int_{-\infty}^{\infty} \rho(x, y)\, dy,$$

que describe la proyección unidimensional de la densidad. A esta función la denominaremos *sombra* de la densidad, ya que representa lo que un detector observaría como silueta del objeto.

En general, reconstruir la función ρ a partir de sus proyecciones en distintas direcciones es un problema complejo que requiere tomar radiografías desde múltiples ángulos. Sin embargo, cuando la función ρ es radial, es decir, solo depende de $x^2 + y^2$, el problema se simplifica notablemente: todas las proyecciones (en cualquier dirección) son iguales. Por tanto, una única sombra $S(x)$ contiene toda la información necesaria para determinar ρ. Este caso especial es el que se abordará en este apartado. Se podría resolver con la transformada de Abel o la transformada de Radon, pero se utilizará un enfoque más elemental que permite aplicar técnicas ya vistas anteriormente.

A continuación, veremos varios ejemplos en los que se mostrará cómo es posible deducir la estructura interna de un objeto simétrico con una sola proyección. Consideraremos las siguientes densidades: $\rho_1(x, y)$, $\rho_2(x, y)$ y $\rho_3(x, y)$, dadas por

$$\begin{cases} 1 & \text{si } x^2 + y^2 < 1, \\ 0 & \text{si } x^2 + y^2 > 1, \end{cases} \qquad \begin{cases} 0 & \text{si } x^2 + y^2 < \frac{1}{4}, \\ 1 & \text{si } \frac{1}{4} < x^2 + y^2 < 1, \\ 0 & \text{si } x^2 + y^2 > 1, \end{cases} \qquad \begin{cases} 1 - x^2 - y^2 & \text{si } x^2 + y^2 < 1, \\ 0 & \text{si } x^2 + y^2 > 1, \end{cases}$$

que corresponden a un disco unidad homogéneo, a otro en el que se le ha vaciado su parte central y a otro que es más denso cuando más cerca del centro estemos.

Para empezar, vamos a calcular la sombra $S_1(x)$ de $\rho_1(x,y)$ que corresponde a la proyección vertical de esta función:

$$S_1(x) = \int_{-\infty}^{\infty} \rho_1(x,y)\,dy.$$

Para un valor fijo de x, el valor de $\rho_1(x,y)$ será 1 solo en los puntos (x,y) que estén dentro del disco, es decir, aquellos que cumplan $x^2+y^2<1$. Para un x tal que $|x|<1$, esto equivale a:

$$y \in \left(-\sqrt{1-x^2},\ \sqrt{1-x^2}\right).$$

En ese intervalo, la función $\rho_1(x,y)$ vale 1, por lo que la integral se reduce a calcular la longitud del intervalo:

$$S_1(x) = \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} 1\,dy = 2\sqrt{1-x^2}.$$

En cambio, si $|x|\geq 1$, la recta vertical no corta al disco y $\rho_1(x,y)=0$ en toda la recta, por lo que la integral es cero. Así, la función sombra queda descrita por:

$$S_1(x) = \begin{cases} 2\sqrt{1-x^2} & \text{si } |x|<1, \\ 0 & \text{si } |x|\geq 1. \end{cases}$$

En el segundo caso, se desea calcular la sombra $S_2(x) = \int_{-\infty}^{\infty} \rho_2(x,y)\,dy$. Si fijamos un valor de x, la función $\rho_2(x,y)$ es igual a 1 cuando $\frac{1}{4} < x^2+y^2 < 1$. Esto se puede reescribir como:

$$\frac{1}{4} - x^2 < y^2 < 1 - x^2.$$

Para que esta desigualdad tenga solución, debe cumplirse $|x|<\frac{1}{2}$. Por lo tanto, los valores de y que cumplen la condición son:

$$y \in \left(-\sqrt{1-x^2},\ -\sqrt{\frac{1}{4}-x^2}\right) \cup \left(\sqrt{\frac{1}{4}-x^2},\ \sqrt{1-x^2}\right).$$

Por tanto, la integral se puede calcular directamente como la suma de las longitudes de estos dos intervalos:

$$S_2(x) = \begin{cases} 2\left(\sqrt{1-x^2} - \sqrt{\frac{1}{4}-x^2}\right) & \text{si } |x|<\frac{1}{2}, \\ 2\sqrt{1-x^2} & \text{si } \frac{1}{2}\leq |x|<1, \\ 0 & \text{si } |x|\geq 1. \end{cases}$$

En este último caso, se calculará la sombra $S_3(x) = \int_{-\infty}^{\infty} \rho_3(x,y)\,dy$. Al fijar un valor de x, se restringe la integración a los valores de y que satisfacen:

$$y \in \left(-\sqrt{1-x^2},\ \sqrt{1-x^2}\right).$$

En ese intervalo, se tiene $\rho_3(x,y) = 1-x^2-y^2$, y, por tanto, la integral se expresa como:

$$S_3(x) = \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} (1-x^2-y^2)\,dy = (1-x^2)\cdot 2\sqrt{1-x^2} - \frac{2}{3}(1-x^2)^{3/2} = \frac{4}{3}(1-x^2)^{3/2}.$$

Por tanto, la función sombra para esta densidad variable es:

$$S_3(x) = \begin{cases} \frac{4}{3}(1-x^2)^{3/2} & \text{si } |x| < 1, \\ 0 & \text{si } |x| \geq 1. \end{cases}$$

Como hemos dicho, vamos a estudiar el caso en que la densidad $\rho(x, y)$ es una función radial. Esto significa que existe una función $f : [0, \infty) \to \mathbb{R}$ tal que

$$\rho(x, y) = f\left(\sqrt{x^2 + y^2}\right).$$

En particular, esto implica que si $t = \sqrt{x^2 + y^2}$, entonces $\rho(x, y) = \rho(t, 0)$, ya que cualquier punto (x, y) puede sustituirse por otro con la misma distancia al origen sin que cambie el valor de ρ. Esta simetría se utilizará para simplificar el análisis. Además, dado que la densidad es simétrica respecto al eje x, la sombra que proyecta al integrarla en dirección vertical será siempre una función par. Al ser $\rho(x, y) = \rho(-x, y)$, se deduce que $S(-x) = S(x)$, por lo que S es par.

Por otro lado, como la función ρ es radial, también lo es su transformada de Fourier $\widehat{\rho}$. Esto es un resultado conocido (véase [7]): es decir,

$$\widehat{\rho}(\xi, \eta) = g\left(\sqrt{\xi^2 + \eta^2}\right)$$

para cierta función g.

Queremos expresar el valor de la densidad en un punto $(t, 0)$, con $t \geq 0$, mediante su transformada de Fourier. Para ello, se utiliza la fórmula de inversión de Fourier en dos dimensiones:

$$\rho(t, 0) = \iint_{\mathbb{R}^2} \widehat{\rho}(\xi, \eta)\, e(t\xi)\, d\xi\, d\eta,$$

donde se ha sustituido $x = t$ y $y = 0$. A continuación, cambiamos a coordenadas polares, la integral queda:

$$\rho(t, 0) = \int_0^{2\pi} \int_0^{\infty} r\, \widehat{\rho}(r\cos\theta, r\sin\theta)\, e(tr\cos\theta)\, dr\, d\theta.$$

Ahora, tendremos en cuenta que $\widehat{\rho}$ es radial, es decir, $\widehat{\rho}(r\cos\theta, r\sin\theta) = \widehat{\rho}(r, 0)$.

Además, se puede reorganizar la integral sobre el círculo completo como una integral sobre la media circunferencia $[0, \pi]$, permitiendo que r recorra todos los valores reales. Este cambio se justifica porque el punto de coordenadas polares $(-r, \theta)$ es el mismo que $(r, \theta + \pi)$, y por tanto se puede cubrir todo el dominio usando $r \in \mathbb{R}$, $\theta \in [0, \pi]$.

Al hacer este cambio de variable, el factor r se convierte en $|r|$ para mantener la orientación positiva del elemento de área, y se obtiene la siguiente expresión equivalente:

$$\rho(t, 0) = \int_0^{\pi} \int_{-\infty}^{\infty} |r|\, \widehat{\rho}(r, 0)\, e^{2\pi i t r\cos\theta}\, dr\, d\theta.$$

Finalmente, observamos que

$$\widehat{S}(r) = \int_{\mathbb{R}} S(x)\, e^{-2\pi i x r}\, dx = \widehat{\rho}(r, 0),$$

ya que

$$\widehat{\rho}(r,0) = \iint \rho(x,y)\, e^{-2\pi i x r}\, dx\, dy = \int_{\mathbb{R}} \left(\int_{\mathbb{R}} \rho(x,y)\, dy \right) e^{-2\pi i x r}\, dx = \widehat{S}(r).$$

Por tanto, la expresión final es:

$$\rho(t,0) = \int_0^{\pi} \int_{-\infty}^{\infty} |r|\, \widehat{S}(r)\, e^{2\pi i t r \cos\theta}\, dr\, d\theta.$$

Esta fórmula permite recuperar la función de densidad ρ a partir de su proyección unidimensional S. Sin embargo, utilizarla directamente en cálculos numéricos no es práctico, ya que exige conocer la transformada de Fourier $\widehat{S}(r)$ y realizar una integración doble.

Para simplificar este proceso, vamos a utilizar herramientas vistas previamente, como el desarrollo en serie de Fourier de funciones periódicas. Vamos a utilizar una representación en serie de Fourier de la función $r \mapsto |r|$, restringida al intervalo $[-\frac{N}{2}, \frac{N}{2}]$. Esta serie se obtiene a partir del desarrollo clásico de la función $G(x) = |x|$ en el intervalo $[-1/2, 1/2]$, cuya extensión periódica define una función continua y 1-periódica. La fórmula es bien conocida:

$$G(x) = \sum_{k=0}^{\infty} a_k \cos(2\pi k x),$$

donde los coeficientes son $a_0 = \frac{1}{4}$, $a_k = -\frac{2}{\pi^2 k^2}$ para k impar y 0 en otro caso.

A partir de esta expresión, se puede obtener una fórmula análoga para la función $r \mapsto |r|$ en el intervalo $[-\frac{N}{2}, \frac{N}{2}]$. Basta observar que $|r| = N \cdot \left|\frac{r}{N}\right| = N \cdot G\left(\frac{r}{N}\right)$, ya que $G(x)$ es precisamente la función valor absoluto escalada y normalizada.

Sustituyendo en la serie de Fourier de G, obtenemos:

$$G\left(\frac{r}{N}\right) = \sum_{k=0}^{\infty} a_k \cos\left(\frac{2\pi k r}{N}\right) \qquad \Rightarrow \qquad |r| = N \sum_{k=0}^{\infty} a_k \cos\left(\frac{2\pi k r}{N}\right).$$

Esto proporciona una forma alternativa de escribir $|r|$ como combinación lineal de funciones coseno de frecuencia discreta.

Esta representación es especialmente útil porque permite sustituir el factor $|r|$ que aparece en la fórmula integral de $\rho(t,0)$ por una suma finita de cosenos. De este modo, se evita calcular explícitamente $\widehat{S}(r)$ en muchos puntos y se reduce la resolución del problema a una combinación de valores de $\widehat{S}(r)$ en frecuencias concretas, que pueden calcularse de forma más eficiente mediante la DFT.

Ahora, podemos sustituir $|r|$ por su serie de Fourier dentro de la integral:

$$\int_{-\infty}^{\infty} |r|\, \widehat{S}(r)\, e(tr\cos\theta)\, dr = \int_{-\infty}^{\infty} \left(N \sum_{k=0}^{\infty} a_k \cos\left(\frac{2\pi k r}{N}\right) \right) \widehat{S}(r)\, e(tr\cos\theta)\, dr.$$

Dado que la suma es finita (o se puede truncar en la práctica), podemos intercambiar el orden de la suma y la integral. Para continuar, vamos a utilizar la siguiente identidad:

$$\cos\left(\frac{2\pi kr}{N}\right) = \frac{1}{2}\left(e\left(\frac{kr}{N}\right) + e\left(-\frac{kr}{N}\right)\right).$$

Combinando esta expresión con el término $e(tr\cos\theta)$, obtenemos:

$$\cos\left(\frac{2\pi kr}{N}\right)\cdot e(tr\cos\theta) = \frac{1}{2}\left(e\left(r(t\cos\theta + \tfrac{k}{N})\right) + e\left(r(t\cos\theta - \tfrac{k}{N})\right)\right).$$

Sustituimos esta expresión dentro de la integral y obtenemos:

$$\frac{1}{2}\left(\int_{-\infty}^{\infty}\widehat{S}(r)\,e\left(r(t\cos\theta + \tfrac{k}{N})\right)\,dr + \int_{-\infty}^{\infty}\widehat{S}(r)\,e\left(r(t\cos\theta - \tfrac{k}{N})\right)\,dr\right).$$

Ahora, si aplicamos la definición de la transformada de Fourier obtenemos:

$$\int_{-\infty}^{\infty}\widehat{S}(r)\,e\left(r(t\cos\theta \pm \tfrac{k}{N})\right)\,dr = S\left(-t\cos\theta \mp \tfrac{k}{N}\right).$$

Además, como ya se ha visto antes, la función S es par, es decir, $S(-x) = S(x)$. Por tanto:

$$S\left(-t\cos\theta \mp \tfrac{k}{N}\right) = S\left(t\cos\theta \pm \tfrac{k}{N}\right).$$

Sustituyendo todo en la expresión inicial, llegamos a la aproximación:

$$\int_{-\infty}^{\infty}|r|\,\widehat{S}(r)\,e(tr\cos\theta)\,dr \approx \frac{N}{2}\sum_{k=0}^{\infty}a_k\,\left(S(t\cos\theta + \tfrac{k}{N}) + S(-t\cos\theta + \tfrac{k}{N})\right).$$

Este resultado es muy útil desde el punto de vista computacional, ya que convierte la integral en una suma de valores de S, evaluados en puntos bien distribuidos. Cuando N es grande, esta aproximación se vuelve muy precisa, y nos permite calcular $\rho(t,0)$ sin tener que trabajar directamente con $\widehat{S}$. Además, gracias a la simetría par de la función S, podemos combinar ambos términos y escribir:

$$\int_{-\infty}^{\infty}|r|\,\widehat{S}(r)\,e(tr\cos\theta)\,dr \approx N\sum_{k=0}^{\infty}a_k\,S\left(\tfrac{k}{N} + t\cos\theta\right).$$

Esta integral depende únicamente de la variable θ, y su cálculo directo implicaría realizar muchas evaluaciones de la función S. Por tanto, para poder evaluarla numéricamente de manera eficiente, vamos a aplicar la *regla del trapecio*, una técnica de integración numérica que consiste en aproximar la integral por una suma de valores de la función en puntos equiespaciados [25].

Dividimos el intervalo $[0,\pi]$ en unos πN subintervalos, de forma que los puntos de evaluación son $\theta_\ell = \frac{\ell}{N}$, esto equivale a tomar un paso de integración $h = \frac{\pi}{\pi N} = \frac{1}{N}$. La regla del trapecio en este caso nos dice que la integral se puede aproximar por:

$$\int_0^{\pi}f(\theta)\,d\theta \approx \frac{1}{N}\sum_{0\le\ell<\pi N}f\left(\frac{\ell}{N}\right),$$

donde en nuestro caso la función f es:

$$f(\theta) = \sum_{k=0}^{2N} a_k \, S\left(\frac{k}{N} + t\cos\theta\right).$$

Sustituyendo esta aproximación en la expresión de $\rho(t,0)$, obtenemos:

$$\rho(t,0) \approx \frac{1}{N} \sum_{0\le\ell<\pi N} \sum_{k=0}^{2N} a_k \, S\left(\frac{k}{N} + t\cos\left(\frac{\ell}{N}\right)\right).$$

Esta fórmula aproxima la integral como una suma doble: la suma exterior recorre los distintos valores de θ, y la interior corresponde al truncamiento de la serie de Fourier de $|r|$. Se trata de una expresión completamente discreta, que puede evaluarse fácilmente en un ordenador.

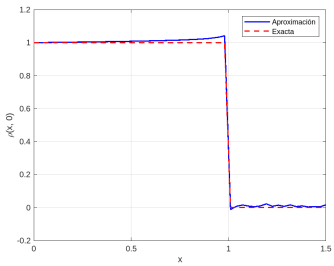
Finalmente, dado que la densidad $\rho(x,y)$ es radial, basta sustituir $t = \sqrt{x^2+y^2}$ para obtener una aproximación general en cualquier punto del plano:

$$\rho(x,y) \approx \sum_{0\le\ell<\pi N} \sum_{k=0}^{2N} a_k \, S\left(\frac{k}{N} + \sqrt{x^2+y^2}\cos\left(\frac{\ell}{N}\right)\right).$$

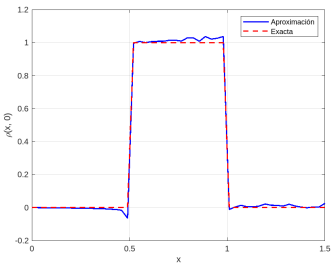
Este resultado concluye el desarrollo del método numérico para reconstruir la función de densidad a partir de sus proyecciones unidimensionales.

Finalmente, para ilustrar el método desarrollado, se ha implementado un programa que calcula la aproximación de $\rho(x,0)$ utilizando esta última fórmula en el caso particular $y = 0$, es decir $t = x$.

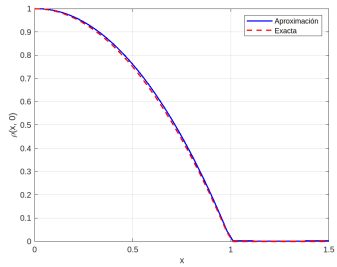
El código evalúa esta aproximación en 50 puntos equiespaciados del intervalo $x \in [0, 3/2]$, para tres funciones distintas $S(x)$ que corresponden a las sombras $S_1(x)$, $S_2(x)$ y $S_3(x)$ vistas al principio del capítulo. Para cada caso, se ha representado gráficamente la función aproximada $\rho(x,0)$, junto con la función exacta correspondiente $\rho(x,0) = \rho_j(x,0)$. Se ha utilizado un valor de $N = 100$, suficientemente grande como para obtener una buena aproximación sin que el tiempo de ejecución sea excesivo. Para cada valor de x, el algoritmo evalúa el doble sumatorio sobre los valores discretos de ℓ y k, tal y como se dedujo en el apartado anterior.



$\rho_1(x,0)$



$\rho_2(x,0)$



$\rho_3(x,0)$

La comparación gráfica entre la aproximación numérica y las funciones exactas muestra que el método implementado es efectivo, especialmente para las funciones suaves como ρ_3. En los casos con discontinuidades (como ρ_1 o ρ_2), es natural que aparezcan oscilaciones en la aproximación debido al fenómeno de Gibbs.

Bibliografía

[1] D. Austin. What is... JPEG? *Notices Amer. Math. Soc.*, 55(2):226–229, 2008.

[2] P. Brémaud. *Mathematical principles of signal processing*. Springer-Verlag, New York, 2002. Fourier and wavelet analysis.

[3] F. Chamizo. Prácticas de Cálculo Numérico I en la UAM. https://matematicas.uam.es/~fernando.chamizo/libreria/fich/APcni.pdf, Curso 2021/2022.

[4] F. Chamizo. LaTeX en la UAM en diez lecciones. https://matematicas.uam.es/~fernando.chamizo/libreria/fich/APlatex24.pdf, Curso 2023/2024.

[5] F. Chamizo. A course on signal processing. http://matematicas.uam.es/~fernando.chamizo/libreria/fich/signal_processing.pdf, Curso 2017/2018.

[6] H. Dym and H. P. McKean. *Fourier series and integrals*. Probability and Mathematical Statistics, No. 14. Academic Press, New York-London, 1972.

[7] G. B. Folland. *Fourier analysis and its applications*. Pacific Grove, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1992.

[8] Y. Katznelson. *An introduction to harmonic analysis*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, third edition, 2004.

[9] T. W. Körner. *Fourier analysis*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 2022. With a foreword by T. Tao.

[10] A. Rényi. *A diary on information theory*. Akadémiai Kiadó (Publishing House of the Hungarian Academy of Sciences), Budapest, 1984.

[11] S. Roman. *Coding and information theory*, volume 134 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992.

[12] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[13] K. T. Smith, D. C. Solmon, and S. L. Wagner. Practical and mathematical aspects of the problem of reconstructing objects from radiographs. *Bull. Am. Math. Soc.*, 83:1227–1270, 1977.

[14] G. Strang. The discrete cosine transform. *SIAM Rev.*, 41(1):135–147, 1999.

[15] R. Tesoro. Cuando la taba se eleva, brota el azar. *Gac. R. Soc. Mat. Esp.*, 16(3):543–559, 2013. https://gaceta.rsme.es/english/abrir.php?id=1162.

[16] S. Winder. *Analog and Digital Filter Design*. EDN Series for Design Engineers. Elsevier Science, 2002.

[17] Pr∞fWiki contributors. Fourier Transform of Gaussian Function — Pr∞fWiki. https://proofwiki.org/w/index.php?title=Fourier_Transform_of_Gaussian_Function&oldid=668452, 2023. [Online; accessed 26-December-2023].

[18] Wikipedia contributors. Abel transform — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Abel_transform&oldid=1239227961, 2024. [Online; accessed 7-April-2025].

[19] Wikipedia contributors. Discrete cosine transform — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Discrete_cosine_transform&oldid=1237214220, 2024. [Online; accessed 20-August-2024].

[20] Wikipedia contributors. Cooley-Tukey FFT algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Cooley%E2%80%93Tukey_FFT_algorithm&oldid=1237705262, 2024. [Online; accessed 21-August-2024].

[21] Wikipedia contributors. Finite impulse response — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Finite_impulse_response&oldid=1240945295, 2024. [Online; accessed 8-November-2024].

[22] Wikipedia contributors. JPEG — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=JPEG&oldid=1284340128, 2025. [Online; accessed 9-April-2025].

[23] Wikipedia contributors. Kraft-McMillan inequality — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kraft%E2%80%93McMillan_inequality&oldid=1254083406, 2024. [Online; accessed 1-February-2025].

[24] Wikipedia contributors. Radon transform — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Radon_transform&oldid=1250073731, 2024. [Online; accessed 7-April-2025].

[25] Wikipedia contributors. Trapezoidal rule — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Trapezoidal_rule&oldid=1279480977, 2025. [Online; accessed 8-April-2025].

APÉNDICE A
Demostración Lema

Esta es la demostración del Lema 1.2 sobre la transformada de Fourier de una gaussiana.

Demostración. Por definición de la transformada de Fourier, se tiene

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e(-\xi x)\,dx = \int_{-\infty}^{\infty} e^{-\alpha\pi x^2}e^{-2\pi i\xi x}\,dx.$$

Derivamos con respecto a ξ y nos queda

$$\frac{\partial}{\partial\xi}\hat{f}(\xi) = -2\pi i\int_{-\infty}^{\infty} e^{-\alpha\pi x^2}xe^{-2\pi i\xi x}\,dx.$$

Ahora integramos por partes tomando $u = e^{-2\pi i\xi x}$ y $dv = xe^{-\alpha\pi x^2}$. El primer término se cancela, por lo tanto tenemos

$$\frac{\partial}{\partial\xi}\hat{f}(\xi) = 2\pi i\int_{-\infty}^{\infty} -\frac{1}{2\alpha\pi}e^{-\alpha\pi x^2}(-2\pi i\xi)e^{-2\pi i\xi x}\,dx = \frac{2\pi i^2\xi}{\alpha}\int_{-\infty}^{\infty} e^{-\alpha\pi x^2}e^{-2\pi i\xi x}\,dx = \frac{-2\pi\xi}{\alpha}\hat{f}(\xi).$$

A continuación, resolvemos por el método de ecuaciones diferenciales de variables separables.

$$d(\hat{f}(\xi)) = \frac{-2\pi\xi}{\alpha}\hat{f}(\xi)d\xi \quad \Rightarrow \quad \frac{d(\hat{f}(\xi))}{\hat{f}(\xi)} = \frac{-2\pi\xi}{\alpha}d\xi \quad \Rightarrow \quad \int\frac{d(\hat{f}(\xi))}{\hat{f}(\xi)} = \int\frac{-2\pi\xi}{\alpha}d\xi$$

$$\Rightarrow \quad \log(\hat{f}(\xi)) = -\frac{\pi\xi^2}{\alpha} + C \quad \Rightarrow \quad \hat{f}(\xi) = Ae^{-\pi\xi^2/\alpha}.$$

Para terminar, calculamos $\hat{f}(0)$ para saber el valor de A.

$$\hat{f}(0) = \int_{-\infty}^{\infty} e^{-\alpha\pi x^2}\,dx.$$

Como conocemos el resultado de la integral Gaussiana, tomamos $a = \alpha\pi$ y resolvemos de la misma manera.

$$\int_{-\infty}^{\infty} e^{-\alpha\pi x^2}\,dx = \sqrt{\frac{\pi}{\alpha\pi}} = \frac{1}{\sqrt{\alpha}}.$$

Finalmente nos queda

$$\hat{f}(\xi) = \alpha^{-1/2}e^{-\pi\xi^2/\alpha}.$$

□

APÉNDICE B

Código MATLAB 1

Este es el código del programa que aplica la limpieza de frecuencias en el ejemplo que hay en el primer capítulo.

```
% Parametros
N = 59; % Longitud de la se al

% Definimos las funciones f y g
f = @(x) 4*(x/117).^2 .* (1 - x/117).^2;
g = @(x) abs(2/9*x - floor(2/9*x + 1/2)) - 1/4;

% Construimos la senal con ruido
n = 0:N-1; % Indices
s = f(n); % Senal original
x = s + 0.02 * g(n); % Senal con ruido

% Aplicamos la DCT a la senal con ruido
Xc = dct(x);

% Filtramos dejando solo los primeros 10 coeficientes
Xc_filtrada = zeros(size(Xc));
Xc_filtrada(1:10) = Xc(1:10);

% Reconstruimos la se al usando la IDCT
x_limpia = idct(Xc_filtrada);

% Graficamos los resultados
figure;
subplot(3,1,1);
plot(n, s, 'g', 'LineWidth', 1.3);
title('Se al Original s(n)');
xlabel('n');
ylabel('Amplitud');

subplot(3,1,2);
plot(n, x, 'r', 'LineWidth', 1.3);
title('Se al con Ruido x(n)');
xlabel('n');
```

```
35 ylabel('Amplitud');
36
37 subplot(3,1,3);
38 plot(n, x_limpia, 'b', 'LineWidth', 1.3);
39 title('Se al␣Filtrada␣IDCT(H_{10}(x))');
40 xlabel('n');
41 ylabel('Amplitud');
42
43 % Guardanos los resultados
44 print(gcf, 'resultados', '-dpng', '-r300');
```

APÉNDICE C
Código MATLAB 2

Este es el código del programa que genera un texto de 10^6 caracteres que se emplea en un ejmplo del tercer capítulo.

```
N = 1e6; % Numero total de caracteres

carac = ['A', 'B']; % Posibles caractereres
probs = [0.8, 0.2]; % Probabilidades de los caracteres

% Ahora generamos la secuencia aleatoria de caracteres
texto = carac(randsrc(1, N, [1 2; probs]));

% Por ultimo, guardamos el texto en un archivo
fid = fopen('textoAB.txt', 'w'); % Abrimos un archivo en modo escritura
fprintf(fid, '%s', texto); % Se escribe el texto
fclose(fid); % Cerramos el archivo

disp('Archivo "textoAB.txt" generado exitosamente.');
```

APÉNDICE D
Métodos de degradación Templo

En estas imágenes podremos ver cómo afectan distintos métodos de degradación a la imagen de un templo.

DCT, $p = 0{,}2$ DCT, $p = 0{,}5$ DCT, $p = 0{,}8$

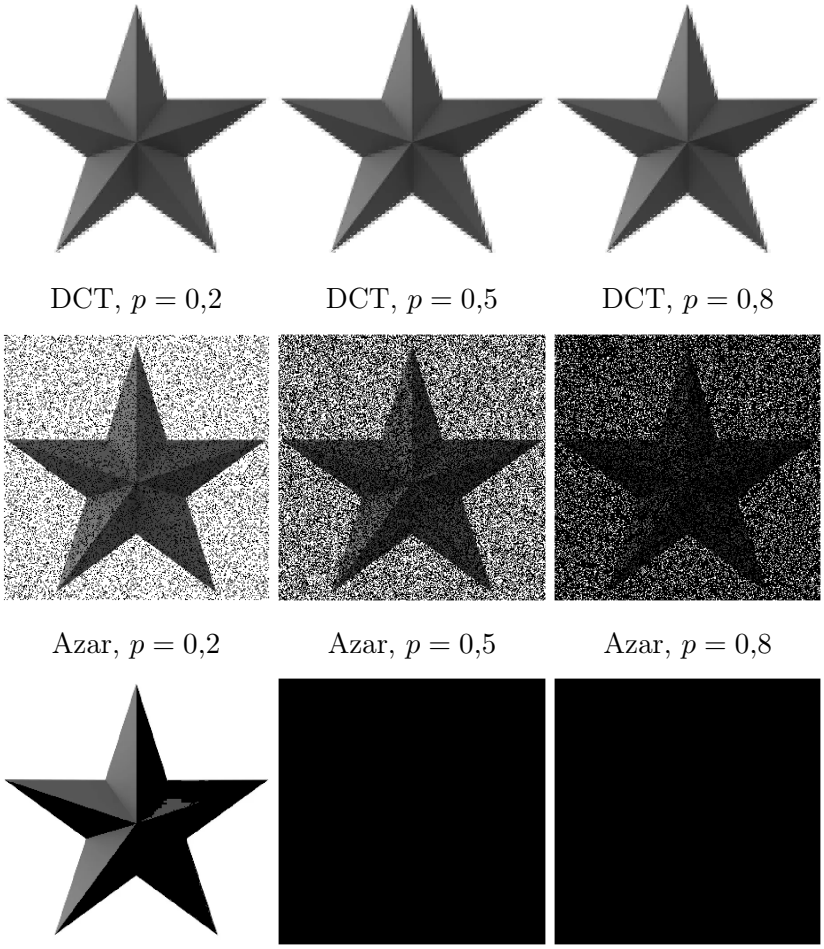
Azar, $p = 0{,}2$ Azar, $p = 0{,}5$ Azar, $p = 0{,}8$

Oscuros, $p = 0{,}2$ Oscuros, $p = 0{,}5$ Oscuros, $p = 0{,}8$

Figura D.1: Degradación Templo de Diana

APÉNDICE E
Métodos de degradación Estrella

En estas imágenes podremos ver cómo afectan distintos métodos de degradación a la imagen de una estrella sobre fondo blanco.



DCT, $p = 0{,}2$ DCT, $p = 0{,}5$ DCT, $p = 0{,}8$

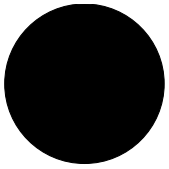
Azar, $p = 0{,}2$ Azar, $p = 0{,}5$ Azar, $p = 0{,}8$

Oscuros, $p = 0{,}2$ Oscuros, $p = 0{,}5$ Oscuros, $p = 0{,}8$

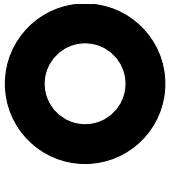
Figura E.1: Degradación Estrella

APÉNDICE F
Sombras de las densidades

Esta es una representación de las sombras que corresponden a las densidades vistas en el cuarto capítulo.



$$S_1(x) = \begin{cases} 2\sqrt{1-x^2} & \text{si } |x| < 1, \\ 0 & \text{si } |x| \geq 1, \end{cases}$$



$$S_2(x) = S_1(x) - \frac{1}{2}S_1(2x)$$

$$S_3(x) = \begin{cases} \frac{4}{3}(1-x^2)^{3/2} & \text{si } |x| < 1, \\ 0 & \text{si } |x| \geq 1. \end{cases}$$