Elementos de Cálculo Numérico

Martín D. Maas

Contents

Introducción

> *The Mathematics are either pure or mixed... And as for the mixed mathematics, I may only make this prediction, that there cannot fail to be more kinds of them as nature grows further disclosed.*
> — Francis Bacon, *The Advancement of Learning*, Book II, 1605.

> *El estudio profundo de la naturaleza es la fuente más fértil de descubrimientos matemáticos.*
> — Joseph Fourier, *Prólogo de la Teoría analítica del calor*

La computación numérica es una rama de la matemática y las ciencias de la computación que se ocupa del desarrollo, análisis e implementación de algoritmos para resolver problemas matemáticos mediante el uso de computadoras.

Este apunte se organiza siguiendo el principio de mínima generalidad[1], según el cual es conveniente introducir las ideas en el marco conceptual más simple posible, no el más general. Este enfoque privilegia la comprensión profunda de casos específicos fundamentales por sobre la búsqueda de la máxima generalidad abstracta.

Inestabilidad

Por ejemplo, consieremos uno de los problemas centrales del cálculo numérico: la inestabilidad, que en sus diferentes manifestaciones, será un tema recurrente a lo largo de todo el apunte. ¿Cuál es el ejemplo más sencillo posible que nos permite ilustrar este concepto?

Consideremos el problema aparentemente inocente de calcular las integrales

$$I_n = \int_0^1 x^n e^x dx, \quad n = 0, 1, 2, \ldots$$

Integrando por partes, obtenemos la relación de recurrencia:

$$I_n = e - nI_{n-1}, \quad n \geq 1,$$

con condición inicial $I_0 = e - 1$.

[1] V.I. Arnold, prólogo de *Lectures on Partial Differential Equations*: "Instead of the principle of maximal generality that is usual in mathematical books, the author has attempted to adhere to the principle of minimal generality, according to which every idea should first be clearly understood in the simplest situation; only then should the method developed be extended to more complicated cases."

Esta fórmula sugiere un algoritmo simple: calcular I_0 y luego usar la recurrencia para obtener $I_1, I_2, \ldots$. Sin embargo, este método es numéricamente catastrófico. Cualquier error en la condición inicial (por ejemplo, al aproximar e con precisión finita) se amplifica enormemente: el error en I_n es aproximadamente $n!$ veces el error en I_0. Para $n = 20$, esto significa una amplificación del error por un factor de $\sim 10^{18}$, destruyendo completamente la precisión incluso con aritmética de doble precisión.

Veremos que la estabilidad numérica no es una cuestión técnica menor, sino una consideración fundamental que determina la viabilidad práctica de los métodos computacionales.

Complejidad y computabilidad

Además de la inestabilidad, otro tema central en computación numérica es la **complejidad computacional**: ¿cuántos recursos (tiempo, memoria) requiere resolver un problema con una precisión dada?

Para estudiar esta cuestión, hemos incorporado un primer capítulo sobre los fundamentos de la computación numérica. La introducción de conceptos de teoría de la computabilidad obedece a la necesidad de situar a la computación numérica dentro del marco de la **matemática constructiva**. En este paradigma, afirmar que un objeto matemático existe es equivalente a proporcionar un algoritmo para construirlo.

Este enfoque revela una tensión fundamental con el análisis matemático clásico. Mientras que en el análisis real trabajamos con el continuo de los números reales, desde el punto de vista constructivo descubrimos que el conjunto de números "efectivamente computables", como veremos en el capítulo 1, es apenas numerable. Esto implica la existencia de una inmensa mayoría de números reales **incomputables** para los cuales no existe ningún algoritmo de aproximación. Reconocer estas diferencias no es un ejercicio de generalidad abstracta, sino un punto de partida concreto para una disciplina basada en algoritmos.

Espacios con producto interno y bases ortogonales

Buscaremos situarnos, siempre que sea conveniente, en el contexto de los **espacios con producto interno** (espacios de Hilbert), donde la geometría es particularmente rica y las herramientas analíticas son más potentes.

Por ese motivo, trabajaremos casi exclusivamente con la norma 2 de un vector, definida mediante el producto interno:

$$\|x\|_2 = \sqrt{\langle x, x \rangle}.$$

y con la norma 2 de una matriz inducida por la norma vectorial:

$$\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2.$$

Esto nos sitúa en un marco teórico sólido para abordar problemas fundamentales del álgebra lineal numérica, como los problemas de cuadrados mínimos, los métodos basados en ortogonalización, la descomposición en valores singulares (SVD), entre otros problemas que serán abordados en el capítulo 2.

En relación a los espacios con producto interno, una cuestión de importancia central son las **bases ortogonales**. Una de ellas, con propiedades analíticas y computacionales cruciales, es la base de Fourier de exponenciales complejas:

$$\left\{e^{ikx}\right\}_{k\in\mathbb{Z}}, \quad x \in [0, 2\pi].$$

Esta base será estudiada en profundidad en el Capítulo 4, junto con las series de Fourier y la Transformada Rápida de Fourier (FFT). Aquí aplicaremos nuevamente el principio de mínima generalidad, y nos concentraremos en el caso de las funciones suaves donde la convergencia es rápida (típicamente exponencial o casi-exponencial) y por lo tanto más útil computacionalmente. Dejamos para cursos avanzados la pregunta sobre cuáles son los espacios de funciones más generales para los cuales las series de Fourier convergen de manera meramente puntual o en otros sentidos débiles.

Interpolación y teoría de aproximación

En el capítulo 3, estudiaremos el problema clásico de la **interpolación polinomial**: dado un conjunto de puntos, encontrar un polinomio que pase exactamente por todos ellos. Este problema admite una elegante formulación como un problema de álgebra lineal y constituye una de las herramientas fundamentales de la teoría de aproximación.

La interpolación tiene aplicaciones directas en la evaluación de funciones, diferenciación e integración numérica (las fórmulas de cuadratura de Newton-Cotes se basan en integrar polinomios interpoladores), y construcción de métodos para ecuaciones diferenciales. Además, el estudio de la convergencia del polinomio interpolador cuando aumentamos el número de puntos revela fenómenos sutiles como el fenómeno de Runge, que motiva la necesidad de elegir cuidadosamente los nodos de interpolación.

Aunque el enfoque principal será en aplicaciones analíticas sobre $\mathbb{R}$ y $\mathbb{C}$, presentaremos brevemente las sorprendentes aplicaciones de la interpolación sobre **cuerpos finitos** $\mathbb{Z}_p$ en teoría de códigos (códigos de Reed-Solomon) y criptografía (secreto compartido de Shamir), ilustrando la universalidad del problema de interpolación.

Polinomios Ortogonales y Series de Fourier Generalizadas

Después de estudiar interpolación, el curso prosigue generalizando los argumentos de aproximación mediante **polinomios ortogonales**. Familias como los polinomios de Legendre, Chebyshev, Laguerre y Hermite emergen naturalmente como autofunciones de problemas de Sturm-Liouville asociados, y heredan propiedades de convergencia similares a las de las series de Fourier cuando se aplican a funciones suficientemente suaves.

Este paralelismo no es accidental: tanto las bases de Fourier como las bases de polinomios ortogonales son casos especiales de un fenómeno más general relacionado con operadores diferenciales autoadjuntos y sus espectros. La comprensión profunda de estos casos específicos proporciona intuición para teorías más abstractas en análisis funcional y ecuaciones diferenciales.

Ecuaciones diferenciales: estabilidad, consistencia y convergencia

El curso prosigue con la resolución numérica de ecuaciones diferenciales, un tema central en matemática aplicada. Aquí estudiaremos la interacción entre dos ideas claves del cálculo numérico: la **consistencia** (qué tan bien el esquema discreto aproxima la ecuación diferencial continua) y la **estabilidad** (si los errores se amplifican o decaen durante la computación). El teorema de Lax establece que, bajo condiciones apropiadas, **consistencia + estabilidad = convergencia**.

Por simplicidad, limitaremos gran parte del desarrollo teórico a casos con **condiciones de frontera periódicas**, lo que nos permite apoyarnos en la teoría desarrollada sobre los modos discretos de Fourier para presentar un análisis riguroso de estabilidad conocido como el **análisis de von Neumann**. Este análisis, aunque restringido al caso periódico, proporciona intuición profunda que se extiende (con las modificaciones apropiadas) a otros tipos de condiciones de frontera. Si el tiempo lo permite, el apunte contiene material adicional para analizar cuándo la generalización de los resultados vistos para condiciones periódicas es válida para otras condiciones de frontera.

Optimización y sistemas no lineales

Finalmente, el último capítulo es una introducción a la optimización y la resolución de sistemas de ecuaciones no lineales. Este tema contiene aplicaciones importantes como el **entrenamiento de redes neuronales** mediante el método de descenso por gradiente estocástico, así como una introducción al **método de Newton**, sobre el cual se basan numerosos métodos de optimización avanzados como la programación cuadrática secuencial y los métodos de punto interior, que se estudian en cursos avanzados de optimización no-lineal.

Organización del apunte

Los capítulos están organizados para construir progresivamente las herramientas necesarias:

1. **Fundamentos**: Computabilidad, complejidad y aritmética de máquina.
2. **Álgebra lineal numérica**: Sistemas lineales, cuadrados mínimos, autovalores.
3. **Interpolación**: Polinomios interpoladores, base de Lagrange, y aplicaciones en teoría de aproximación y cuerpos finitos.
4. **Series de Fourier**: Bases ortogonales, FFT, convergencia espectral.
5. **Polinomios ortogonales**: Legendre, Chebyshev, cuadratura de Gauss.
6. **Ecuaciones diferenciales ordinarias**: Métodos de Runge-Kutta, estabilidad.
7. **Ecuaciones en derivadas parciales**: Métodos de diferencias finitas, análisis de von Neumann.
8. **Optimización**: Descenso por gradiente, Newton, cuadrados mínimos no lineales.

A lo largo de todo el texto, privilegiamos la comprensión profunda sobre la enciclopédica. Cada tema se desarrolla con suficiente detalle para proporcionar tanto intuición geométrica como rigor matemático, pero sin intentar cubrir todas las posibles extensiones y generalizaciones. El objetivo es que el lector termine el curso con una comprensión sólida de los principios fundamentales de la computación numérica, capaz de aplicarlos a problemas concretos y de consultar literatura especializada cuando sea necesario.

Prerequisitos

Este apunte asume familiaridad con los conceptos fundamentales del álgebra lineal. En particular, se requiere conocimiento de:

- **Espacios vectoriales**: subespacios, dimensión, bases, independencia lineal.
- **Transformaciones lineales**: núcleo, imagen, representación matricial, cambio de base.
- **Producto interno y ortogonalidad**: espacios con producto interno, bases ortogonales y ortonormales, proyección ortogonal, proceso de Gram-Schmidt.
- **Teorema espectral**: diagonalización de matrices simétricas (o hermitianas), el teorema espectral para matrices normales (matrices que conmutan con su adjunta: $A^*A = AA^*$).
- **Forma canónica de Jordan**: existencia de la forma de Jordan para matrices complejas, bloques de Jordan, y su relación con autovalores y autovectores generalizados.

El concepto de proyección ortogonal es importante para entender los métodos de cuadrados mínimos y la descomposición de funciones en bases ortogonales (Fourier, polinomios ortogonales). El teorema espectral, a su vez, es utilizado para obtener la descomposición en valores singulares (SVD) de una matriz, una herramienta fundamental en la solución de sistemas sobredeterminados, así como para el análisis de la propagación de errores en la solución de sistemas lineales. La forma de Jordan, aunque no es utilizada directamente, proporciona el marco teórico para entender el comportamiento de algunos métodos iterativos, así como para entender la estabilidad de ciertos esquemas numéricos.

Se asumen también conocimientos básicos de cálculo (series, convergencia, derivadas e integrales, ecuaciones diferenciales ordinarias, etc.).

Si bien no se asume familiaridad con la programación, si se asumirá la madurez conceptual como para entender algoritmos y estructuras de datos básicas (listas, arreglos, bucles, funciones).

Chapter 1

Fundamentos de la Computación Numérica

Este capítulo introductorio establece los fundamentos teóricos y prácticos de la computación numérica. Exploramos tres pilares fundamentales: la teoría de la computabilidad, que define qué problemas son resolubles algorítmicamente; la complejidad computacional, que cuantifica los recursos necesarios para resolver un problema; y la aritmética de máquina, que determina cómo se representan y manipulan los números en computadoras reales.

1.1 Máquinas de Turing y Computabilidad

1.1.1 Máquinas de Turing

Una **máquina de Turing** es un modelo matemático abstracto de computación propuesto por Alan Turing en 1936. Consiste en:

- Una **cinta infinita** dividida en celdas, cada una conteniendo un símbolo de un alfabeto finito Γ (que incluye un símbolo especial *blanco*).
- Un **cabezal de lectura/escritura** que puede leer y escribir símbolos en la cinta, y moverse una posición a la izquierda o derecha.
- Un **conjunto finito de estados** Q, con un estado inicial q_0 y estados de aceptación $F \subseteq Q$.
- Una **función de transición** $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ que, dado el estado actual y el símbolo leído, determina el nuevo estado, el símbolo a escribir, y la dirección del movimiento.

Ejemplo 1.1 (Máquina que suma 1)**.** *Consideremos una máquina de Turing que suma 1 a un número natural representado en unario. El número n se representa como una cadena de n unos:* 1^n.

Especificación:

- *Alfabeto:* $\Gamma = \{1, B\}$ *donde* B *representa el blanco.*

- *Estados:* $Q = \{q_0, q_1, q_{halt}\}$
- *Estado inicial:* q_0
- *Estado de aceptación:* $\{q_{halt}\}$

Función de transición:

$$\begin{aligned}
\delta(q_0, 1) &= (q_0, 1, R) && \textit{(avanza a la derecha mientras haya unos)} \\
\delta(q_0, B) &= (q_1, 1, L) && \textit{(encontró el final, escribe un 1 adicional)} \\
\delta(q_1, 1) &= (q_{halt}, 1, R) && \textit{(retrocede y se detiene)}
\end{aligned}$$

Ejemplo de ejecución *para la entrada* 111 *(representa el número 3):*

$\underline{1}$	1	1	B	B	...	*Estado:* q_0
1	$\underline{1}$	1	B	B	...	*Estado:* q_0
1	1	$\underline{1}$	B	B	...	*Estado:* q_0
1	1	1	$\underline{B}$	B	...	*Estado:* q_0
1	1	1	$\underline{1}$	B	...	*Estado:* q_1 *(escribió el 1 adicional)*
1	1	$\underline{1}$	1	B	...	*Estado:* q_{halt} *(se detiene)*

La cinta final contiene 1111*, que representa el número* $4 = 3 + 1$*. El subrayado indica la posición del cabezal.*

Observación 1.2. *Aunque este ejemplo es simple, ilustra los principios fundamentales:*

- *La máquina opera mediante una secuencia de pasos discretos.*
- *Cada paso depende únicamente del estado actual y del símbolo leído.*
- *La máquina tiene memoria finita (los estados), pero puede usar la cinta para almacenamiento ilimitado.*
- *Operaciones complejas se construyen componiendo transiciones simples.*

1.1.2 Codificación

Un aspecto fundamental para trabajar con máquinas de Turing es la **codificación**: cómo representar objetos matemáticos (números, funciones, e incluso otras máquinas) como cadenas de símbolos en un alfabeto finito.

Definición 1.3 (Codificación)**.** *Una* ***codificación*** *de un conjunto* S *es una función inyectiva* $\langle\cdot\rangle : S \to \Sigma^*$ *que asocia a cada elemento de* S *una cadena finita sobre un alfabeto finito* Σ*.*

Codificación de números naturales. La forma más simple es usar representación unaria: $\langle n \rangle = 1^{n+1}$ (o 0^{n+1}). Por ejemplo, $\langle 3 \rangle = 1111$. Alternativamente, podemos usar representación binaria: $\langle 5 \rangle = 101$.

Codificación de tuplas. Para codificar pares (m, n) podemos usar una función de emparejamiento de Cantor:

$$\langle (m, n) \rangle = \langle \frac{(m+n)(m+n+1)}{2} + n \rangle$$

Esta función es biyectiva $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, permitiendo representar cualquier par de naturales como un único natural.

Codificación de máquinas de Turing. Este es el paso crucial: una máquina de Turing $M = (Q, \Gamma, \delta, q_0, F)$ tiene una descripción finita, por lo que puede codificarse como una cadena. Podemos:

1. Enumerar los estados: $q_0, q_1, \ldots, q_k$.
2. Enumerar los símbolos: γ_0 (blanco), $\gamma_1, \ldots, \gamma_m$.
3. Codificar cada transición $\delta(q_i, \gamma_j) = (q_k, \gamma_\ell, D)$ como una tupla de números.
4. Concatenar todas las transiciones separadas por un símbolo especial.

Por ejemplo, si usamos el alfabeto $\{0, 1, \#\}$, podríamos codificar la transición $\delta(q_2, \gamma_3) = (q_5, \gamma_1, R)$ como:

$$0^{2+1}\#0^{3+1}\#0^{5+1}\#0^{1+1}\#0^{1}$$

donde el último 0 representa R (derecha) y 00 representaría L (izquierda).

Observación 1.4. *La posibilidad de codificar máquinas de Turing como cadenas es fundamental por dos razones:*

1. ***Máquinas universales**: Podemos construir una máquina de Turing U que recibe $\langle M \rangle$ y w como entrada, y simula el comportamiento de M sobre w. Esto es análogo a un computador moderno que ejecuta programas (ver siguiente subsección).*
2. ***Auto-referencia**: Una máquina puede recibir su propia descripción como entrada, lo que permite construir argumentos diagonales (como veremos en el problema de la detención).*

Definición 1.5 (Enumeración de máquinas de Turing)**.** *Dado un esquema de codificación fijo, podemos enumerar todas las máquinas de Turing: $M_0, M_1, M_2, \ldots$ donde M_i es la máquina cuya codificación es la i-ésima cadena válida en orden lexicográfico. Las cadenas que no codifican una máquina válida pueden asociarse a una máquina trivial que se detiene inmediatamente.*

1.1.3 La máquina de Turing universal

Uno de los resultados más profundos de la teoría de la computación es la existencia de una **máquina de Turing universal**.

Teorema 1.6 (Máquina de Turing Universal)**.** *Existe una máquina de Turing U (llamada **universal**) que toma como entrada la codificación de cualquier máquina de Turing M junto con una entrada w, y simula el comportamiento de M sobre w. Es decir:*

$$U(\langle M, w \rangle) = M(w)$$

Si M se detiene en w, entonces U se detiene y produce el mismo resultado. Si M no se detiene en w, entonces U tampoco se detiene.

Idea de la construcción. La máquina universal U funciona interpretando la descripción codificada de M:

1. Lee la codificación $\langle M \rangle$ para extraer la tabla de transiciones de M.
2. Mantiene en su cinta una representación del estado actual de M y del contenido de la cinta de M.
3. En cada paso, consulta la tabla de transiciones de M para determinar qué haría M, y actualiza la simulación en consecuencia.
4. Se detiene cuando M alcanzaría un estado de aceptación o rechazo.

Aunque U es más lenta que M (tiene que interpretar cada paso), eventualmente produce el mismo resultado. □

Observación 1.7 (Importancia de la máquina universal)**.** *La máquina de Turing universal es el precursor conceptual de la computadora moderna de propósito general:*

- *Una única máquina puede ejecutar cualquier algoritmo simplemente cambiando su "programa" (la codificación $\langle M \rangle$).*
- *El hardware (la máquina U) está separado del software (las máquinas M que se ejecutan).*
- *Esta idea se materializó en la arquitectura de von Neumann, donde programas y datos se almacenan en la misma memoria.*

Observación 1.8 (Arquitectura de von Neumann)**.** *En junio de 1945,* ***John von Neumann*** *escribió el influyente documento "First Draft of a Report on the EDVAC" (Electronic Discrete Variable Automatic Computer), donde formalizó por primera vez la* ***arquitectura de programa almacenado****.*

Los principios fundamentales de esta arquitectura son:

- ***Programa almacenado****: Las instrucciones del programa se almacenan en la memoria de la misma forma que los datos, codificadas en formato binario.*
- ***Memoria unificada****: Una única memoria contiene tanto el programa como los datos sobre los que opera.*
- ***Procesamiento secuencial****: Un contador de programa indica qué instrucción ejecutar, avanzando secuencialmente (salvo por instrucciones de salto).*
- ***Componentes básicos****: Unidad aritmético-lógica (ALU), unidad de control, memoria, y dispositivos de entrada/salida.*

Esta arquitectura es la ***realización física de la máquina de Turing universal****: así como U puede simular cualquier máquina M leyendo su codificación $\langle M \rangle$, una computadora de von Neumann puede ejecutar cualquier programa simplemente cargándolo en memoria. El "programa" es análogo a $\langle M \rangle$, y el procesador que lo interpreta es análogo a la máquina universal U.*

La arquitectura de von Neumann se convirtió en el modelo estándar para prácticamente todas las computadoras construidas desde entonces, desde mainframes hasta smartphones, materializando el concepto abstracto de computación universal de Turing.

1.1.4 La Tesis de Church-Turing

Hasta ahora hemos definido la computabilidad en términos de máquinas de Turing, pero existen otros modelos de computación: funciones recursivas (Church), cálculo lambda (Church), máquinas de registro, etc. Sorprendentemente, todos estos modelos resultan ser equivalentes.

Definición 1.9 (Tesis de Church-Turing)**.** *La **Tesis de Church-Turing** afirma que:*

> *Toda función efectivamente calculable (en el sentido intuitivo) es computable por una máquina de Turing.*

Equivalentemente: cualquier procedimiento algorítmico que pueda ser ejecutado por un ser humano siguiendo reglas mecánicas puede ser simulado por una máquina de Turing.

Observación 1.10. *La Tesis de Church-Turing **no es un teorema matemático**, sino una propuesta filosófica sobre la naturaleza de la computación. No puede ser probada formalmente porque involucra el concepto informal de "efectivamente calculable". Sin embargo, está fuertemente respaldada por:*

- *La equivalencia demostrada entre todos los modelos formales de computación propuestos.*
- *Décadas de experiencia: nunca se ha encontrado un procedimiento efectivo que no pueda ser simulado por una máquina de Turing.*
- *La máquina de Turing universal muestra que el modelo es suficientemente poderoso para autosimularse.*

Observación 1.11 (Implicaciones)**.** *Si aceptamos la Tesis de Church-Turing, entonces:*

- *Los límites de las máquinas de Turing son los límites de la computación en general.*
- *El problema de la detención es indecidible para cualquier modelo de computación, no solo para máquinas de Turing.*
- *No importa cuánto avance la tecnología (computadoras cuánticas, biológicas, etc.): no podrán resolver problemas incomputables como el problema de la detención.*

Definición 1.12 (Función computable)**.** *Una función $f : \mathbb{N} \to \mathbb{N}$ es **computable** (o **recursiva**) si existe una máquina de Turing que, dada una entrada n codificada en la cinta, se detiene con $f(n)$ codificado en la cinta.*

1.1.5 Números computables

Definición 1.13 (Número computable)**.** *Un número real x es **computable** si existe una máquina de Turing que, dado $n \in \mathbb{N}$, calcula un racional r_n tal que $|x - r_n| < 1/n$. Equivalentemente, existe un algoritmo que puede calcular cualquier dígito de la expansión decimal de x.*

Ejemplo 1.14 (Números computables)**.** *Son computables:*

- *Todos los números racionales: $\mathbb{Q} \subset \mathbb{R}_{comp}$.*

- *Raíces de polinomios con coeficientes racionales:* $\sqrt{2}$, $\sqrt[3]{7}$, *etc.*
- *Constantes matemáticas definidas por algoritmos:* π, e, $\ln 2$, *etc.*
- *Números definidos por series convergentes con términos computables:*

$$\sum_{n=0}^{\infty} \frac{1}{n!}, \quad \sum_{n=1}^{\infty} \frac{1}{n^2}.$$

1.1.6 Incomputabilidad

A pesar de la potencia de las máquinas de Turing, existen problemas que ninguna máquina de Turing puede resolver.

Teorema 1.15 (Problema de la detención)**.** *No existe ningún algoritmo que, dada una descripción de una máquina de Turing M y una entrada w, determine si M se detiene cuando se ejecuta con entrada w.*

Formalmente, definimos el **conjunto de detención** como:

$$H = \{\langle M, w\rangle : \text{la máquina } M \text{ se detiene cuando se ejecuta con entrada } w\}$$

El problema de la detención afirma que H es un conjunto **no computable**: no existe máquina de Turing que compute su función característica.

Idea de la demostración. La demostración usa un argumento diagonal similar al de Cantor, aprovechando que las máquinas de Turing pueden codificarse y recibirse a sí mismas como entrada.

Supongamos, buscando una contradicción, que existe una máquina de Turing H que resuelve el problema de la detención. Es decir, H recibe como entrada la codificación $\langle M, w\rangle$ y:

- Se detiene y acepta si M se detiene con entrada w.
- Se detiene y rechaza si M no se detiene con entrada w.

Usando H, construimos una nueva máquina D ("diagonal") que hace lo siguiente al recibir la codificación de una máquina $\langle M\rangle$:

1. Ejecuta $H(\langle M, \langle M\rangle\rangle)$ para determinar si M se detiene cuando recibe su propia descripción como entrada.
2. Si H acepta (es decir, M se detiene con entrada $\langle M\rangle$), entonces D entra en un **bucle infinito**.
3. Si H rechaza (es decir, M no se detiene con entrada $\langle M\rangle$), entonces D se **detiene** inmediatamente.

Ahora consideramos qué sucede cuando ejecutamos D con su propia descripción como entrada: $D(\langle D\rangle)$.

- **Caso 1**: Supongamos que $D(\langle D \rangle)$ se detiene. Entonces, por la construcción de D, esto significa que H rechazó, lo que implica que D no se detiene con entrada $\langle D \rangle$. Contradicción.
- **Caso 2**: Supongamos que $D(\langle D \rangle)$ no se detiene (entra en bucle). Entonces H debe haber aceptado, lo que significa que D sí se detiene con entrada $\langle D \rangle$. Contradicción.

En ambos casos llegamos a una contradicción. Por tanto, nuestra suposición inicial es falsa: no puede existir una máquina H que resuelva el problema de la detención. □

Observación 1.16 (Importancia de la codificación)**.** *La demostración del problema de la detención ilustra por qué la codificación es esencial:*

1. *Sin codificación, no podríamos hablar de "dar una máquina como entrada a otra máquina".*
2. *La auto-referencia $D(\langle D \rangle)$ es posible precisamente porque las máquinas son objetos finitos codificables.*
3. *El argumento diagonal funciona porque podemos enumerar todas las máquinas de Turing y aplicar cada máquina M_i a su propia codificación $\langle M_i \rangle$.*

Observación 1.17 (Consecuencias del problema de la detención)**.** *El problema de la detención tiene profundas implicaciones:*

- *No existe un depurador universal que detecte si un programa tiene bucles infinitos.*
- *No existe un verificador universal que determine si un programa es correcto.*
- *Muchos otros problemas son indecidibles por reducción al problema de la detención.*

Observación 1.18. *La mayoría de los números reales son **incomputables**. Como el conjunto de máquinas de Turing es numerable (cada máquina tiene una descripción finita), solo hay numerablemente muchos números computables. Pero $\mathbb{R}$ es no numerable, por lo que casi todos los reales son incomputables. Sin embargo, ningún número incomputable específico puede ser exhibido constructivamente.*

1.2 Complejidad Computacional

La teoría de la complejidad no pregunta si un problema es resoluble, sino *cuántos recursos* (tiempo, memoria) requiere su solución.

1.2.1 Notación asintótica

Para analizar algoritmos, necesitamos comparar cómo crece su tiempo de ejecución con el tamaño de la entrada.

Definición 1.19 (Notación O grande)**.** *Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones. Decimos que $f(n) = O(g(n))$ si existen constantes $c > 0$ y $n_0 \in \mathbb{N}$ tales que*

$$f(n) \leq c \cdot g(n) \quad \text{para todo } n \geq n_0.$$

*Interpretación: $g(n)$ es una **cota superior asintótica** de $f(n)$.*

Definición 1.20 (Notación Ω grande)**.** *Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones. Decimos que $f(n) = \Omega(g(n))$ si existen constantes $c > 0$ y $n_0 \in \mathbb{N}$ tales que*

$$f(n) \geq c \cdot g(n) \quad \text{para todo } n \geq n_0.$$

*Interpretación: $g(n)$ es una **cota inferior asintótica** de $f(n)$.*

Definición 1.21 (Notación Θ grande)**.** *Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones. Decimos que $f(n) = \Theta(g(n))$ si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$. Equivalentemente, existen constantes $c_1, c_2 > 0$ y $n_0 \in \mathbb{N}$ tales que*

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \text{para todo } n \geq n_0.$$

*Interpretación: $g(n)$ es el **orden exacto** de $f(n)$.*

Definición 1.22 (Notación o chica)**.** *Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones. Decimos que $f(n) = o(g(n))$ si para toda constante $c > 0$, existe $n_0 \in \mathbb{N}$ tal que*

$$f(n) < c \cdot g(n) \quad \text{para todo } n \geq n_0.$$

*Equivalentemente: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Interpretación: $f(n)$ crece **estrictamente más lento** que $g(n)$.*

Definición 1.23 (Notación ω chica)**.** *Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones. Decimos que $f(n) = \omega(g(n))$ si para toda constante $c > 0$, existe $n_0 \in \mathbb{N}$ tal que*

$$f(n) > c \cdot g(n) \quad \text{para todo } n \geq n_0.$$

*Equivalentemente: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$. Interpretación: $f(n)$ crece **estrictamente más rápido** que $g(n)$.*

Analogía con relaciones de orden: O es como $\leq$, o es como $<$, Ω es como $\geq$, ω es como $>$, y Θ es como $=$.

Ejemplo 1.24 (Órdenes de crecimiento comunes)**.** *De más lento a más rápido:*

$$O(1) \ll O(\log n) \ll O(n) \ll O(n \log n) \ll O(n^2) \ll O(n^3) \ll O(2^n) \ll O(n!).$$

Ejemplos:

- $3n^2 + 5n + 7 = \Theta(n^2)$.
- $n \log n + 100n = \Theta(n \log n)$.
- $2^n + n^{100} = \Theta(2^n)$.

1.2.2 Clases de complejidad temporal y espacial

Definición 1.25 (Clase temporal TIME($f(n)$))**.** *La clase TIME($f(n)$) contiene todos los lenguajes (problemas de decisión) que pueden ser resueltos por una máquina de Turing determinista en tiempo $O(f(n))$.*

Definición 1.26 (Clase espacial SPACE($f(n)$))**.** *La clase SPACE($f(n)$) contiene todos los lenguajes que pueden ser resueltos por una máquina de Turing determinista usando $O(f(n))$ celdas de memoria.*

Definición 1.27 (Clase P)**.** $\mathbf{P} = \bigcup_{k=0}^{\infty} TIME(n^k)$ *es la clase de problemas resolubles en tiempo polinomial. Representa los problemas tratables computacionalmente.*

Definición 1.28 (Clase NP)**.** **NP** *es la clase de problemas para los cuales, dada una solución propuesta, se puede verificar en tiempo polinomial si es correcta. Equivalentemente, son los problemas resolubles en tiempo polinomial por una máquina de Turing no determinista.*

Ejemplo 1.29 (Problemas en P)**.**

- *Determinar si un número es primo (algoritmo AKS, 2002).*
- *Ordenar n números: $O(n \log n)$ (mergesort, heapsort).*
- *Calcular el máximo común divisor: $O(\log n)$ (algoritmo de Euclides).*
- *Multiplicar dos matrices $n \times n$: $O(n^{2.371})$ (algoritmo de Coppersmith-Winograd).*

Ejemplo 1.30 (Problemas en NP)**.**

- ***SAT**: Dado un conjunto de cláusulas booleanas, ¿existe una asignación de valores que las satisface?*
- ***Camino Hamiltoniano**: Dado un grafo, ¿existe un camino que visita cada vértice exactamente una vez?*
- ***Factorización de enteros**: Dado n, ¿tiene un factor primo menor que k? (Verificar es fácil, encontrar es difícil).*
- ***Problema del viajante (TSP)**: ¿Existe una ruta que visita todas las ciudades con distancia total $< k$?*

Observación 1.31 (El problema P vs NP)**.** *Claramente* $\mathbf{P} \subseteq \mathbf{NP}$ *(si podemos resolver, podemos verificar). La pregunta del millón de dólares: ¿es* $\mathbf{P} = \mathbf{NP}$*? Se cree ampliamente que no, pero nadie ha podido demostrarlo. Es uno de los siete Problemas del Milenio del Clay Mathematics Institute.*

1.2.3 Cálculo de la complejidad de un algoritmo

Para determinar la complejidad de un algoritmo:

1. **Identificar operaciones elementales**: asignaciones, comparaciones, operaciones aritméticas. Cada una cuesta $O(1)$.
2. **Contar repeticiones**: analizar bucles y recursiones.

3. **Sumar las contribuciones**: obtener una función $T(n)$ del tamaño de entrada.

4. **Expresar en notación asintótica**: simplificar a $O(f(n))$.

Ejemplo 1.32 (Búsqueda lineal)**.** *Buscar un elemento en un array desordenado de tamaño n:*

```
for i = 1 to n:
    if array[i] == target:
        return i
return -1
```

- *Operaciones en cada iteración: $O(1)$ (comparación).*
- *Número de iteraciones: hasta n en el peor caso.*
- *Complejidad total: $T(n) = O(n)$.*

Ejemplo 1.33 (Búsqueda binaria)**.** *Buscar en un array ordenado de tamaño n:*

```
left = 1, right = n
while left <= right:
    mid = (left + right) / 2
    if array[mid] == target:
        return mid
    else if array[mid] < target:
        left = mid + 1
    else:
        right = mid - 1
return -1
```

- *En cada iteración, el espacio de búsqueda se reduce a la mitad.*
- *Número de iteraciones: k tal que $n/2^k = 1$, es decir, $k = \log_2 n$.*
- *Complejidad: $T(n) = O(\log n)$.*

Ejemplo 1.34 (Ordenamiento por inserción)**.**
```
for i = 2 to n:
    key = array[i]
    j = i - 1
    while j >= 1 and array[j] > key:
        array[j+1] = array[j]
        j = j - 1
    array[j+1] = key
```

- *Bucle externo: $n - 1$ iteraciones.*
- *Bucle interno: en la iteración i, hasta $i - 1$ comparaciones.*
- *Total: $T(n) = \sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2} = O(n^2)$.*

Ejemplo 1.35 (Multiplicación de matrices)**.** *Multiplicar dos matrices A, B de tamaño $n \times n$ con el algoritmo estándar:*

```
for i = 1 to n:
    for j = 1 to n:
        C[i,j] = 0
        for k = 1 to n:
            C[i,j] = C[i,j] + A[i,k] * B[k,j]
```

- *Tres bucles anidados, cada uno de tamaño n.*
- *Complejidad: $T(n) = O(n^3)$.*
- *(Existen algoritmos más eficientes: Strassen $O(n^{2.807})$, Coppersmith-Winograd $O(n^{2.371})$).*

1.2.4 Análisis de algoritmos recursivos

Los algoritmos recursivos requieren un enfoque diferente para analizar su complejidad. En lugar de contar iteraciones directamente, establecemos una **relación de recurrencia** que relaciona el costo de resolver un problema de tamaño n con el costo de resolver subproblemas más pequeños.

Definición 1.36 (Relación de recurrencia)**.** *Una **relación de recurrencia** para el tiempo de ejecución $T(n)$ de un algoritmo recursivo tiene la forma general:*

$$T(n) = \begin{cases} c & \text{si } n \leq n_0 \text{ (caso base)} \\ aT(n/b) + f(n) & \text{si } n > n_0 \text{ (caso recursivo)} \end{cases}$$

donde:

- *a es el número de llamadas recursivas.*
- *n/b es el tamaño de cada subproblema.*
- *$f(n)$ es el costo del trabajo fuera de las llamadas recursivas (dividir y combinar).*

Ejemplo 1.37 (Búsqueda binaria recursiva)**.** *El algoritmo de búsqueda binaria puede implementarse recursivamente:*

```
binary_search(array, target, left, right):
    if left > right:
        return -1
    mid = (left + right) / 2
    if array[mid] == target:
        return mid
    else if array[mid] < target:
        return binary_search(array, target, mid+1, right)
    else:
        return binary_search(array, target, left, mid-1)
```

***Análisis de complejidad**:*

- *Caso base ($n = 1$): $T(1) = O(1)$.*
- *Caso recursivo: una sola llamada recursiva con la mitad del tamaño, más trabajo constante para calcular `mid` y hacer comparaciones.*
- *Relación de recurrencia: $T(n) = T(n/2) + O(1)$.*

***Resolución**: En cada nivel de recursión, el problema se reduce a la mitad. Si empezamos con tamaño n y llegamos al caso base de tamaño 1, tenemos:*

$$n \to n/2 \to n/4 \to \cdots \to n/2^k = 1$$

Esto ocurre cuando $k = \log_2 n$. En cada nivel hacemos trabajo $O(1)$, por lo que:

$$T(n) = O(\log n).$$

Ejemplo 1.38 (Mergesort)**.** *Mergesort es un algoritmo de ordenamiento divide-y-vencerás:*

```
mergesort(array, left, right):
    if left >= right:
        return
    mid = (left + right) / 2
    mergesort(array, left, mid)      // Ordenar mitad izquierda
    mergesort(array, mid+1, right)   // Ordenar mitad derecha
    merge(array, left, mid, right)   // Combinar (O(n))
```

***Análisis de complejidad**:*

- *Caso base ($n = 1$): $T(1) = O(1)$.*
- *Caso recursivo: dos llamadas recursivas con la mitad del tamaño cada una, más el trabajo de combinar las dos mitades ordenadas (que cuesta $O(n)$).*
- *Relación de recurrencia: $T(n) = 2T(n/2) + O(n)$.*

***Resolución mediante árbol de recursión**:*

- ***Nivel 0**: Un problema de tamaño n, costo cn para combinar.*
- ***Nivel 1**: Dos problemas de tamaño $n/2$, costo total $2 \cdot c(n/2) = cn$ para combinar.*
- ***Nivel 2**: Cuatro problemas de tamaño $n/4$, costo total $4 \cdot c(n/4) = cn$ para combinar.*
- $\vdots$
- ***Nivel** k: 2^k problemas de tamaño $n/2^k$, costo total $2^k \cdot c(n/2^k) = cn$ para combinar.*

El árbol tiene profundidad $\log_2 n$ (hasta que $n/2^k = 1$), y en cada nivel el costo total es cn. Por lo tanto:

$$T(n) = cn \cdot \log_2 n = O(n \log n).$$

Ejemplo 1.39 (Fibonacci ingenuo)**.** *El algoritmo recursivo ingenuo para calcular el n-ésimo número de Fibonacci:*

```
fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

***Análisis de complejidad**:*

- *Relación de recurrencia:* $T(n) = T(n-1) + T(n-2) + O(1)$.
- *Esta recurrencia es similar a la secuencia de Fibonacci misma.*

Se puede demostrar que $T(n) = \Theta(\phi^n)$ *donde* $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ *es la razón áurea. Esto es* ***complejidad exponencial****, extremadamente ineficiente.*

Observación*: Este problema ilustra cómo un algoritmo recursivo natural puede ser muy ineficiente. Una implementación iterativa o con memoización reduce la complejidad a* $O(n)$.

1.2.5 El Teorema Maestro

Para recurrencias de la forma $T(n) = aT(n/b) + f(n)$ donde $a \geq 1, b > 1$, el Teorema Maestro proporciona una solución directa sin necesidad de expandir la recursión.

Teorema 1.40 (Teorema Maestro)**.** *Sea* $T(n)$ *definida por la recurrencia:*

$$T(n) = aT(n/b) + f(n)$$

donde $a \geq 1$, $b > 1$ *son constantes, y* $f(n)$ *es una función asintóticamente positiva. Sea* $c_{crit} = \log_b a$. *Entonces:*

1. ***Caso 1*** *(Trabajo dominado por las hojas): Si* $f(n) = O(n^c)$ *para algún* $c < c_{crit}$, *entonces:*

$$T(n) = \Theta(n^{c_{crit}}) = \Theta(n^{\log_b a}).$$

2. ***Caso 2*** *(Trabajo balanceado en todos los niveles): Si* $f(n) = \Theta(n^{c_{crit}} \log^k n)$ *para algún* $k \geq 0$, *entonces:*

$$T(n) = \Theta(n^{c_{crit}} \log^{k+1} n).$$

   En particular, si $f(n) = \Theta(n^{c_{crit}})$, *entonces* $T(n) = \Theta(n^{c_{crit}} \log n)$.

3. ***Caso 3*** *(Trabajo dominado por la raíz): Si* $f(n) = \Omega(n^c)$ *para algún* $c > c_{crit}$, *y además se satisface la* ***condición de regularidad****:*

$$af(n/b) \leq kf(n) \quad \text{para algún } k < 1 \text{ y } n \text{ suficientemente grande},$$

   entonces:

$$T(n) = \Theta(f(n)).$$

Demostración (idea intuitiva). Consideremos el árbol de recursión para $T(n) = aT(n/b) + f(n)$:

Estructura del árbol:

- **Nivel 0** (raíz): 1 nodo de tamaño n, costo $f(n)$.
- **Nivel 1**: a nodos de tamaño n/b, costo total $a \cdot f(n/b)$.
- **Nivel 2**: a^2 nodos de tamaño n/b^2, costo total $a^2 \cdot f(n/b^2)$.
- $\vdots$
- **Nivel** i: a^i nodos de tamaño n/b^i, costo total $a^i \cdot f(n/b^i)$.

Profundidad del árbol: El árbol tiene profundidad $\log_b n$ (cuando $n/b^k = 1$, es decir, $k = \log_b n$).

Número de hojas: En el nivel más profundo hay $a^{\log_b n}$ hojas. Usando propiedades de logaritmos:

$$a^{\log_b n} = n^{\log_b a} = n^{c_{\text{crit}}}.$$

Costo total: El costo total es la suma de los costos en todos los niveles:

$$T(n) = \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) + \Theta(n^{c_{\text{crit}}}).$$

El comportamiento de esta suma depende de cómo crece $f(n)$ comparado con $n^{c_{\text{crit}}}$:

Caso 1: Si $f(n) = O(n^c)$ con $c < c_{\text{crit}}$, entonces $a^i f(n/b^i)$ decrece geométricamente con i. La suma es dominada por las hojas del árbol (nivel más profundo), que contribuyen $\Theta(n^{c_{\text{crit}}})$.

Caso 2: Si $f(n) = \Theta(n^{c_{\text{crit}}})$, entonces $a^i f(n/b^i) \approx n^{c_{\text{crit}}}$ en cada nivel. Con $\log_b n$ niveles, el costo total es:

$$T(n) = \Theta(n^{c_{\text{crit}}} \log n).$$

Caso 3: Si $f(n) = \Omega(n^c)$ con $c > c_{\text{crit}}$, entonces $a^i f(n/b^i)$ crece geométricamente al acercarnos a la raíz. La suma es dominada por el trabajo en la raíz, que contribuye $\Theta(f(n))$ (bajo la condición de regularidad, que garantiza que la serie converge). □

Observación 1.41 (Interpretación intuitiva)**.** *El exponente crítico $c_{crit} = \log_b a$ representa el costo de las hojas del árbol (los casos base). El teorema compara este costo con el trabajo $f(n)$ hecho en cada nivel:*

- *Si las hojas dominan ($f(n)$ crece más lento que $n^{c_{crit}}$), entonces $T(n) = \Theta(n^{c_{crit}})$.*
- *Si todos los niveles contribuyen igual, el costo se multiplica por la profundidad $\log n$.*
- *Si la raíz domina ($f(n)$ crece más rápido), entonces $T(n) = \Theta(f(n))$.*

Ejemplo 1.42 (Aplicaciones del Teorema Maestro)**.** ***1. Búsqueda binaria**: $T(n) = T(n/2) + O(1)$.*

- $a = 1, b = 2, f(n) = O(1) = O(n^0)$.

- $c_{crit} = \log_2 1 = 0$.
- $f(n) = O(n^0) = \Theta(n^{c_{crit}})$, *así que estamos en el Caso 2 con* $k = 0$.
- *Conclusión:* $T(n) = \Theta(n^0 \log n) = \Theta(\log n)$.

***2. Mergesort**:* $T(n) = 2T(n/2) + O(n)$.

- $a = 2, b = 2, f(n) = O(n)$.
- $c_{crit} = \log_2 2 = 1$.
- $f(n) = \Theta(n) = \Theta(n^{c_{crit}})$, *así que estamos en el Caso 2 con* $k = 0$.
- *Conclusión:* $T(n) = \Theta(n \log n)$.

***3. Multiplicación de matrices por divide-y-vencerás (ingenuo)**:* $T(n) = 8T(n/2) + O(n^2)$.

- $a = 8, b = 2, f(n) = O(n^2)$.
- $c_{crit} = \log_2 8 = 3$.
- $f(n) = O(n^2)$ *con* $2 < 3$, *así que estamos en el Caso 1.*
- *Conclusión:* $T(n) = \Theta(n^3)$ *(¡no mejor que el algoritmo estándar!).*

***4. Algoritmo de Strassen**:* $T(n) = 7T(n/2) + O(n^2)$.

- $a = 7, b = 2, f(n) = O(n^2)$.
- $c_{crit} = \log_2 7 \approx 2.807$.
- $f(n) = O(n^2)$ *con* $2 < 2.807$, *así que estamos en el Caso 1.*
- *Conclusión:* $T(n) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.807})$ *(¡mejor que* $O(n^3)$*!).*

***5. Recurrencia con trabajo dominante**:* $T(n) = 2T(n/2) + O(n^2)$.

- $a = 2, b = 2, f(n) = O(n^2)$.
- $c_{crit} = \log_2 2 = 1$.
- $f(n) = \Omega(n^2)$ *con* $2 > 1$, *y se verifica la condición de regularidad:* $2f(n/2) = 2 \cdot (n/2)^2 = n^2/2 \leq kn^2$ *para* $k = 1/2 < 1$.
- *Estamos en el Caso 3.*
- *Conclusión:* $T(n) = \Theta(n^2)$.

Observación 1.43 (Limitaciones del Teorema Maestro)**.** *El Teorema Maestro no cubre todos los casos posibles:*

- *No aplica a recurrencias con subproblemas de tamaños desiguales, como* $T(n) = T(n/3) + T(2n/3) + O(n)$ *(Quickselect).*

- *No aplica cuando $f(n)$ oscila y no es monótona.*
- *Para recurrencias que no caen en ninguno de los tres casos (por ejemplo, $T(n) = 2T(n/2) + n \log n$), se requieren métodos más avanzados como el **teorema maestro generalizado** o análisis directo mediante árboles de recursión.*

1.3 Aritmética de Máquina

En computación práctica, trabajamos con representaciones finitas de números, lo que introduce errores y limitaciones.

1.3.1 Enteros de 64 bits

Los enteros con signo de 64 bits se representan típicamente en **complemento a dos**:

- Rango: $[-2^{63}, 2^{63} - 1] = [-9{,}223{,}372{,}036{,}854{,}775{,}808, 9{,}223{,}372{,}036{,}854{,}775{,}807]$.
- El bit más significativo indica el signo: 0 para positivo, 1 para negativo.
- Operaciones aritméticas $(+, -, \times)$ son exactas si el resultado cabe en el rango.
- **Overflow**: Si el resultado excede el rango, ocurre un desbordamiento (típicamente sin error, dando un resultado módulo 2^{64}).

Ejemplo 1.44 (Overflow de enteros)**.** *En aritmética de 8 bits con signo (rango $[-128, 127]$):*

$$100 + 50 = 150 \quad \textit{pero} \quad 150 > 127 \implies \textit{overflow} \implies -106 \ (\textit{módulo } 2^8).$$

1.3.2 Punto flotante IEEE 754

Los números reales se representan en **formato de punto flotante** según el estándar IEEE 754.

Formato de 64 bits (double precision):

$$x = (-1)^s \times 1.f \times 2^{e-1023},$$

donde:

- s: 1 bit de signo.
- e: 11 bits para el exponente (rango $0 \leq e \leq 2047$).
- f: 52 bits para la mantisa fraccionaria.
- **Números normales**: $1 \leq e \leq 2046$. El 1 implícito da 53 bits de precisión.
- **Números subnormales**: $e = 0$, permiten representar valores muy pequeños cerca de cero.
- **Infinito**: $e = 2047$, $f = 0$.
- **NaN** (Not a Number): $e = 2047$, $f \neq 0$. Resulta de operaciones indefinidas como $0/0$.

Ejemplo 1.45 (Rango de double precision). • *Mínimo positivo normalizado:* $\approx 2.225\times10^{-308}$.

- *Máximo:* $\approx 1.798 \times 10^{308}$.
- *Épsilon de máquina:* $\epsilon = 2^{-52} \approx 2.22 \times 10^{-16}$ *(precisión relativa).*

1.3.3 Errores de punto flotante

Las operaciones de punto flotante no son exactas. El error relativo típico es del orden del épsilon de máquina.

Definición 1.46 (Error de redondeo). *Cuando un número real x se representa en punto flotante como $fl(x)$, el error relativo satisface:*

$$\left|\frac{x - fl(x)}{x}\right| \leq \frac{\epsilon}{2},$$

donde $\epsilon = 2^{-52}$ para double precision.

Ejemplo 1.47 (Pérdida de significancia). *Restar números muy cercanos puede causar pérdida de dígitos significativos:*

$$a = 1.234567890123456, \quad b = 1.234567890123455.$$

En precisión finita:

$$a - b \approx 0.000000000000001 \quad \textit{(solo 1 dígito significativo restante).}$$

Ejemplo 1.48 (Suma no asociativa). *En punto flotante, $(a + b) + c \neq a + (b + c)$ en general:*

$$(10^{16} + 1) - 10^{16} = 0 \quad \textit{(el 1 se pierde)},$$

$$10^{16} + (1 - 10^{16}) = 10^{16} + (-10^{16} + 1) = 1 \quad \textit{(diferente orden).}$$

1.3.4 Precisión arbitraria por software

Para superar las limitaciones de punto flotante, existen bibliotecas de **precisión arbitraria**:

- **BigInt**: Enteros de tamaño ilimitado. Disponible nativamente en lenguajes modernos (Python, JavaScript ES2020).
- **BigFloat**: Números de punto flotante con precisión configurable. Bibliotecas: GMP (C), mpmath (Python), Arb (C).

Ejemplo 1.49 (Cálculo de π con alta precisión). *Usando mpmath en Python:*

```
from mpmath import mp
mp.dps = 100  # 100 dígitos decimales
pi_100 = mp.pi
print(pi_100)
```

Resultado:

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664
```

Costo: Las operaciones con precisión arbitraria son *mucho más lentas* que las de hardware (típicamente 10-1000 veces). Además, la complejidad crece con la precisión.

1.3.5 Ejemplos en geometría computacional

La aritmética finita puede causar errores graves en geometría computacional.

Ejemplo 1.50 (Test de orientación)**.** *Dados tres puntos A, B, C en el plano, determinar si C está a la izquierda, derecha, o sobre la línea AB. Se calcula el determinante:*

$$D = \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} = (x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A).$$

- $D > 0$*: C a la izquierda.*
- $D < 0$*: C a la derecha.*
- $D = 0$*: C colineal.*

Problema*: Con punto flotante, D puede ser muy pequeño pero no exactamente cero, llevando a decisiones erróneas sobre colinealidad.*

Ejemplo 1.51 (Intersección de segmentos)**.** *Determinar si dos segmentos AB y CD se intersectan requiere comparaciones geométricas precisas. Errores de redondeo pueden hacer que:*

- *Segmentos que se intersectan apenas se reporten como no intersectándose.*
- *Algoritmos de triangulación fallen al producir geometría inválida.*

Solución*: Usar predicados exactos (exact predicates) mediante aritmética de precisión arbitraria o expansión de flotantes (algoritmo de Shewchuk).*

Ejemplo 1.52 (Algoritmo de Graham para el casco convexo)**.** *Construir el casco convexo de n puntos requiere ordenar puntos por ángulo y hacer tests de orientación. Errores numéricos pueden:*

- *Hacer que el algoritmo produzca un polígono no convexo.*
- *Causar que el algoritmo entre en bucle infinito al no poder decidir correctamente la orientación.*

Observación 1.53 (Geometría computacional robusta)**.** *Los algoritmos geométricos comerciales (CAD, GIS) usan técnicas especiales:*

- ***Aritmética exacta*** *para predicados críticos.*
- ***Perturbación simbólica*** *para romper degeneraciones.*
- ***Snap rounding*** *para redondear coordenadas a una rejilla.*

Chapter 2

Álgebra Lineal Numérica

2.1 Eliminación Gaussiana y factorización LU

La **eliminación Gaussiana** es el método más antiguo y fundamental para resolver sistemas lineales. Consiste en transformar el sistema en uno equivalente de forma triangular superior, que se resuelve fácilmente por sustitución hacia atrás.

2.1.1 Eliminación Gaussiana sin pivoteo

Ejemplo 2.1 (Eliminación Gaussiana)**.** *Consideremos el sistema:*

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}.$$

Paso 1: *Eliminar x_1 de las ecuaciones 2 y 3.*

- *Multiplicador para fila 2: $m_{21} = 4/2 = 2$. Restar $2\times$ fila 1 de fila 2.*
- *Multiplicador para fila 3: $m_{31} = -2/2 = -1$. Restar $-1\times$ fila 1 de fila 3.*

Resultado:

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 8 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ -12 \\ 14 \end{pmatrix}.$$

Paso 2: *Eliminar x_2 de la ecuación 3.*

- *Multiplicador: $m_{32} = 8/(-8) = -1$. Restar $-1\times$ fila 2 de fila 3.*

Resultado (matriz triangular superior):

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ -12 \\ 2 \end{pmatrix}.$$

Paso 3: *Sustitución hacia atrás:*

$$\begin{aligned} x_3 &= 2, \\ x_2 &= \frac{-12 + 2x_3}{-8} = \frac{-12 + 4}{-8} = 1, \\ x_1 &= \frac{5 - x_2 - x_3}{2} = \frac{5 - 1 - 2}{2} = 1. \end{aligned}$$

2.1.2 Factorización LU

La eliminación Gaussiana puede interpretarse como una factorización matricial.

Teorema 2.2 (Factorización LU)**.** *Si la eliminación Gaussiana se puede realizar sin intercambio de filas (sin pivoteo), entonces A admite una factorización:*

$$A = LU,$$

donde:

- $L \in \mathbb{C}^{n\times n}$ *es triangular inferior con unos en la diagonal,*
- $U \in \mathbb{C}^{n\times n}$ *es triangular superior.*

Además, los elementos de L son precisamente los multiplicadores m_{ij} usados en la eliminación.

Idea. Cada paso de eliminación puede representarse como una multiplicación por una matriz triangular inferior. En el paso k, eliminamos los elementos debajo del pivote a_{kk} multiplicando por la izquierda por:

$$M_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1,k} & \ddots & \\ & & \vdots & & 1 \\ & & -m_{n,k} & & \end{pmatrix}.$$

Después de $n-1$ pasos:

$$M_{n-1} \cdots M_2 M_1 A = U.$$

Como cada M_k es triangular inferior invertible, podemos escribir:

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U = LU,$$

donde $L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1}$. Un cálculo directo muestra que L tiene los multiplicadores m_{ij} debajo de la diagonal. □

Ejemplo 2.3 (Construcción de L y U)**.** *Para la matriz del ejemplo anterior:*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{pmatrix} = LU.$$

Los elementos de L debajo de la diagonal son los multiplicadores: $m_{21} = 2$, $m_{31} = -1$, $m_{32} = -1$.

2.1.3 Pivoteo parcial y factorización PLU

La eliminación Gaussiana puede fallar o ser numéricamente inestable si los pivotes son pequeños. La solución es el **pivoteo parcial**: intercambiar filas para asegurar que el pivote sea el elemento de mayor magnitud en cada columna.

Teorema 2.4 (Factorización PLU)**.** *Toda matriz invertible $A \in \mathbb{C}^{n\times n}$ admite una factorización:*

$$PA = LU,$$

donde P es una matriz de permutación, L es triangular inferior con unos en la diagonal, y U es triangular superior.

Observación 2.5 (Solución de sistemas con PLU)**.** *Una vez calculada la factorización $PA = LU$, resolver $Ax = b$ requiere:*

1. *Permutar el vector: $\tilde{b} = Pb$*
2. *Resolver $Ly = \tilde{b}$ (sustitución hacia adelante): costo $O(n^2)$*
3. *Resolver $Ux = y$ (sustitución hacia atrás): costo $O(n^2)$*

El costo total es $O(n^3)$ para la factorización (hecha una sola vez) más $O(n^2)$ por cada sistema resuelto.

2.2 Ortogonalización de Gram-Schmidt y factorización QR

Otra factorización importante es la **factorización QR**, que descompone una matriz en un producto de una matriz ortogonal y una triangular superior. Se obtiene mediante el proceso de **Gram-Schmidt**.

2.2.1 Proceso de Gram-Schmidt clásico

Teorema 2.6 (Gram-Schmidt Clásico (CGS))**.** *Dado un conjunto de vectores linealmente independientes $\{a_1, \ldots, a_n\} \subset \mathbb{C}^m$, el proceso de Gram-Schmidt construye un conjunto ortonormal $\{q_1, \ldots, q_n\}$ que genera el mismo subespacio:*

$$span\{a_1, \ldots, a_k\} = span\{q_1, \ldots, q_k\}, \quad k = 1, \ldots, n.$$

Algoritmo (Gram-Schmidt Clásico):

1. Para $k = 1, \ldots, n$:
   (a) $\tilde{q}_k = a_k$
   (b) Para $j = 1, \ldots, k-1$:
   - $r_{jk} = (a_k, q_j)$
   - $\tilde{q}_k \leftarrow \tilde{q}_k - r_{jk} q_j$

   (c) $r_{kk} = \|\tilde{q}_k\|_2$
   (d) $q_k = \tilde{q}_k / r_{kk}$

Observación 2.7. *El proceso CGS ortogonaliza a_k restando de una sola vez las proyecciones sobre todos los vectores ortogonales $q_1, \ldots, q_{k-1}$ previamente calculados.*

2.2.2 Gram-Schmidt Modificado (MGS)

El problema con CGS es que es numéricamente inestable: los errores de redondeo pueden hacer que los vectores resultantes no sean ortogonales. El **Gram-Schmidt Modificado** (MGS) resuelve este problema.

Algoritmo (Gram-Schmidt Modificado):

1. Para $k = 1, \ldots, n$:
   (a) $\tilde{q}_k = a_k$
   (b) Para $j = 1, \ldots, k-1$:
   - $r_{jk} = (\tilde{q}_k, q_j)$ (*usar el vector parcialmente ortogonalizado*)
   - $\tilde{q}_k \leftarrow \tilde{q}_k - r_{jk} q_j$ (*ortogonalizar paso a paso*)

   (c) $r_{kk} = \|\tilde{q}_k\|_2$
   (d) $q_k = \tilde{q}_k / r_{kk}$

Observación 2.8 (Diferencia entre CGS y MGS)**.** *La diferencia clave es que MGS actualiza $\tilde{q}_k$ después de cada proyección y usa el vector actualizado en la siguiente proyección. Esto hace que MGS sea mucho más estable numéricamente, produciendo vectores con mejor ortogonalidad en presencia de errores de redondeo.*

2.2.3 Factorización QR

El proceso de Gram-Schmidt aplicado a las columnas de una matriz produce la factorización QR.

Teorema 2.9 (Factorización QR)**.** *Toda matriz $A \in \mathbb{C}^{m\times n}$ con $m \geq n$ y rango completo admite una factorización:*

$$A = QR,$$

donde:

- $Q \in \mathbb{C}^{m\times n}$ *tiene columnas ortonormales: $Q^*Q = I_n$,*
- $R \in \mathbb{C}^{n\times n}$ *es triangular superior con diagonal positiva.*

Si se extiende Q a una matriz unitaria $m \times m$, se puede escribir:

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R,$$

donde $Q_1 Q_1^* + Q_2 Q_2^* = I_m$.

Demostración. Aplicando Gram-Schmidt (modificado) a las columnas de $A = (a_1 \,|\, \cdots \,|\, a_n)$, obtenemos vectores ortonormales $q_1, \ldots, q_n$ y coeficientes r_{jk} tales que:

$$a_k = \sum_{j=1}^{k} r_{jk} q_j = \sum_{j=1}^{n} r_{jk} q_j,$$

donde $r_{jk} = 0$ para $j > k$ (triangularidad superior). Escribiendo esto en forma matricial:

$$A = QR, \quad Q = (q_1 \,|\, \cdots \,|\, q_n), \quad R = (r_{ij}).$$

□

Ejemplo 2.10 (Factorización QR explícita). *Sea*

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Aplicando MGS:

1. $a_1 = (1,1,0)^T$, $r_{11} = \|a_1\|_2 = \sqrt{2}$, $q_1 = (1/\sqrt{2}, 1/\sqrt{2}, 0)^T$.
2. $\tilde{q}_2 = a_2 = (1,0,1)^T$.
3. $r_{12} = (a_2, q_1) = 1/\sqrt{2}$.
4. $\tilde{q}_2 \leftarrow (1,0,1)^T - \frac{1}{\sqrt{2}}(1/\sqrt{2}, 1/\sqrt{2}, 0)^T = (1/2, -1/2, 1)^T$.
5. $r_{22} = \|\tilde{q}_2\|_2 = \sqrt{1/4 + 1/4 + 1} = \sqrt{3/2}$.
6. $q_2 = (1/\sqrt{6}, -1/\sqrt{6}, 2/\sqrt{6})^T$.

Entonces:

$$A = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{2} & -1/\sqrt{6} \\ 0 & 2/\sqrt{6} \end{pmatrix} \begin{pmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & \sqrt{3/2} \end{pmatrix} = QR.$$

1. **Solución de sistemas lineales**: Si $A = QR$ con A cuadrada e invertible, entonces:

$$Ax = b \quad \Rightarrow \quad QRx = b \quad \Rightarrow \quad Rx = Q^*b,$$

   que se resuelve por sustitución hacia atrás en $O(n^2)$ operaciones.

2. **Problemas de cuadrados mínimos**: Como veremos en el próximo capítulo, QR es superior a las ecuaciones normales para resolver problemas de mínimos cuadrados.

3. **Cálculo de autovalores**: El método QR iterativo (Capítulo 3) es el algoritmo estándar para calcular todos los autovalores de una matriz.

Observación 2.11 (Comparación: LU vs QR).

- ***Costo***: *La factorización LU cuesta $\frac{2}{3}n^3$ operaciones, mientras que QR (vía MGS) cuesta $2n^3$ operaciones. LU es más rápida.*
- ***Estabilidad***: *QR es más estable numéricamente, especialmente para matrices mal condicionadas.*
- ***Aplicaciones***: *LU se prefiere para sistemas lineales simples. QR se prefiere para cuadrados mínimos, autovalores, y cuando la estabilidad es crítica.*

2.3 Cuadrados mínimos y proyección ortogonal

El problema de **cuadrados mínimos** consiste en encontrar $x \in \mathbb{C}^n$ que minimiza el residuo $\|Ax - b\|_2$ para un sistema sobredeterminado $Ax = b$ ($m \geq n$).

Recordemos de Álgebra Lineal que si S es un subespacio de un espacio con producto interno V, la mejor aproximación de un vector $b \in V$ en S es su **proyección ortogonal** $P_S(b)$. Esta proyección se caracteriza por la propiedad de que el error es ortogonal al subespacio:

$$(b - P_S(b)) \perp S.$$

En el contexto de matrices, buscamos $Ax \in \mathrm{Im}(A)$ que esté lo más cerca posible de b. Por lo tanto, el residuo $r = b - Ax$ debe ser ortogonal a todo el espacio columna $\mathrm{Im}(A)$.

2.3.1 Ecuaciones normales para cuadrados mínimos

El problema de **cuadrados mínimos** consiste en encontrar $x \in \mathbb{C}^n$ que minimiza:

$$\|Ax - b\|_2, \quad A \in \mathbb{C}^{m \times n},\ b \in \mathbb{C}^m,\ m \geq n.$$

Teorema 2.12 (Ecuaciones normales)**.** *El vector x que minimiza $\|Ax - b\|_2$ satisface las* ***ecuaciones normales****:*

$$A^*Ax = A^*b.$$

Si A tiene rango completo (rango n), entonces A^*A es invertible y la solución única es:

$$x = (A^*A)^{-1}A^*b.$$

Demostración. Sea x el minimizador de $\|Ax - b\|_2$. Por el Teorema de proyección ortogonal, el error $r = Ax - b$ debe ser ortogonal al espacio columna de A, es decir, $r \perp \mathrm{Im}(A)$.

Esto significa que para todo $v \in \mathbb{C}^n$, tenemos $Av \perp r$, o equivalentemente:

$$(Av, r) = 0 \quad \text{para todo } v \in \mathbb{C}^n.$$

Usando la propiedad del producto interno:

$$(Av, Ax - b) = 0 \quad \text{para todo } v \in \mathbb{C}^n.$$

Por la antilinealidad en el segundo argumento y la propiedad $(u, v) = v^*u$ para vectores:

$$(Av)^*(Ax - b) = v^*A^*(Ax - b) = 0 \quad \text{para todo } v \in \mathbb{C}^n.$$

Como esto debe valer para todo $v \in \mathbb{C}^n$, necesariamente:

$$A^*(Ax - b) = 0,$$

lo que es equivalente a las ecuaciones normales:

$$A^*Ax = A^*b.$$

Para la segunda parte, si A tiene rango completo (rango n), entonces las columnas de A son linealmente independientes. Esto implica que A^*A es invertible, pues si $A^*Ax = 0$, entonces:

$$\|Ax\|^2 = (Ax, Ax) = x^*A^*Ax = x^*0 = 0,$$

lo que implica $Ax = 0$, y por el rango completo de A, necesariamente $x = 0$. Por lo tanto, A^*A es invertible y la solución única es:

$$x = (A^*A)^{-1}A^*b.$$

□

Observación 2.13 (Condicionamiento de las ecuaciones normales)**.** *El problema con las ecuaciones normales es que la matriz A^*A tiene número de condición que es el* cuadrado *del número de condición de A:*

$$\kappa_2(A^*A) = \kappa_2(A)^2.$$

Esto puede hacer que el problema de cuadrados mínimos se vuelva numéricamente intratable incluso cuando A está moderadamente condicionada.

2.3.2 QR para cuadrados mínimos

Una alternativa mucho mejor condicionada es usar la **factorización QR**.

Teorema 2.14 (Factorización QR)**.** *Toda matriz $A \in \mathbb{C}^{m\times n}$ con $m \geq n$ admite una factorización:*

$$A = QR,$$

donde $Q \in \mathbb{C}^{m\times m}$ es unitaria ($Q^*Q = I$) y $R \in \mathbb{C}^{m\times n}$ es triangular superior.

Si A tiene rango completo, podemos escribir:

$$A = Q\begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix},$$

donde $R_1 \in \mathbb{C}^{n\times n}$ es triangular superior invertible y $Q_1 \in \mathbb{C}^{m\times n}$ tiene columnas ortonormales.

Teorema 2.15 (Solución de cuadrados mínimos vía QR)**.** *Si $A = QR$ con Q unitaria y R triangular superior, entonces la solución de cuadrados mínimos de $\|Ax-b\|_2$ se reduce a resolver:*

$$R_1x = Q_1^*b,$$

un sistema triangular que se resuelve por sustitución hacia atrás en $O(n^2)$ operaciones.

Demostración. Como Q es unitaria, preserva la norma:

$$\|Ax - b\|_2 = \|QRx - b\|_2 = \|Rx - Q^*b\|_2 = \left\| \begin{pmatrix} R_1x \\ 0 \end{pmatrix} - \begin{pmatrix} Q_1^*b \\ Q_2^*b \end{pmatrix} \right\|_2.$$

Esto se minimiza cuando $R_1x = Q_1^*b$, dando un residuo mínimo de $\|Q_2^*b\|_2$. □

Observación 2.16 (Ventajas del método QR)**.**

1. *El número de condición del problema no se eleva al cuadrado: $\kappa_2(R_1) = \kappa_2(A)$.*
2. *El método es numéricamente estable.*
3. *No requiere formar explícitamente A^*A, evitando pérdida de precisión.*

2.4 Descomposición en Valores Singulares (SVD)

La **descomposición en valores singulares** (SVD) es una de las factorizaciones más importantes en álgebra lineal numérica.

Para construirla nos apoyamos en el **teorema espectral** para matrices hermíticas, aunque lo enunciamos aquí en el caso más general de matrices normales.

Teorema 2.17 (Teorema espectral para matrices normales)**.** *Sea $A \in \mathbb{C}^{n\times n}$ una matriz normal ($A^*A = AA^*$). Entonces A es diagonalizable por una matriz unitaria:*

$$A = U\Lambda U^*,$$

donde U es unitaria y Λ es diagonal.

Demostración. La demostración de este teorema se estudia en un curso de Álgebra Lineal. □

Teorema 2.18 (SVD)**.** *Toda matriz $A \in \mathbb{C}^{m\times n}$ admite una descomposición:*

$$A = U\Sigma V^*,$$

donde:

- *$U \in \mathbb{C}^{m\times m}$ es unitaria (columnas = vectores singulares izquierdos),*
- *$V \in \mathbb{C}^{n\times n}$ es unitaria (columnas = vectores singulares derechos),*
- *$\Sigma \in \mathbb{R}^{m\times n}$ es diagonal con entradas no negativas $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$ (valores singulares).*

Demostración. Consideremos la matriz $A^*A \in \mathbb{C}^{n\times n}$. Esta matriz es hermítica, pues:

$$(A^*A)^* = A^*(A^*)^* = A^*A,$$

y además es semidefinida positiva, ya que para todo $x \in \mathbb{C}^n$:

$$x^*(A^*A)x = (Ax)^*(Ax) = \|Ax\|_2^2 \geq 0.$$

Por el teorema espectral, existe una base ortonormal $\{v_1, \ldots, v_n\}$ de $\mathbb{C}^n$ formada por autovectores de A^*A, con autovalores reales no negativos $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Es decir:

$$A^*Av_j = \lambda_j v_j, \quad j = 1, \ldots, n.$$

Sea $V \in \mathbb{C}^{n\times n}$ la matriz unitaria cuyas columnas son estos autovectores: $V = [v_1 \mid v_2 \mid \cdots \mid v_n]$. Entonces:

$$A^*A = V\Lambda V^*, \quad \text{donde } \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

Definimos los **valores singulares** de A como:

$$\sigma_j = \sqrt{\lambda_j}, \quad j = 1, \ldots, n.$$

Estos son no negativos y están ordenados: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Sea $r = \text{rango}(A)$. Observemos que $\text{rango}(A) = \text{rango}(A^*A)$ (esto se sigue de que $\ker(A) = \ker(A^*A)$). Por lo tanto, exactamente r de los autovalores λ_j son positivos, y los valores singulares satisfacen:

$$\sigma_1 \geq \cdots \geq \sigma_r > 0, \quad \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Para $j = 1, \ldots, r$, definimos:

$$u_j = \frac{1}{\sigma_j} A v_j.$$

Estos vectores son ortonormales. En efecto, para $1 \leq i, j \leq r$:

$$u_i^* u_j = \frac{1}{\sigma_i \sigma_j} (Av_i)^* (Av_j) = \frac{1}{\sigma_i \sigma_j} v_i^* A^* A v_j = \frac{1}{\sigma_i \sigma_j} v_i^* (\lambda_j v_j) = \frac{\lambda_j}{\sigma_i \sigma_j} v_i^* v_j.$$

Como $\sigma_j^2 = \lambda_j$ y $v_i^* v_j = \delta_{ij}$ (delta de Kronecker), obtenemos:

$$u_i^* u_j = \frac{\sigma_j^2}{\sigma_i \sigma_j} \delta_{ij} = \frac{\sigma_j}{\sigma_i} \delta_{ij} = \delta_{ij}.$$

Por lo tanto, $\{u_1, \ldots, u_r\}$ es un conjunto ortonormal en $\mathbb{C}^m$.

Completamos $\{u_1, \ldots, u_r\}$ a una base ortonormal $\{u_1, \ldots, u_m\}$ de $\mathbb{C}^m$ (esto siempre es posible mediante el proceso de Gram-Schmidt aplicado a cualquier extensión de $\{u_1, \ldots, u_r\}$ a una base de $\mathbb{C}^m$).

Sea $U \in \mathbb{C}^{m \times m}$ la matriz unitaria cuyas columnas son estos vectores: $U = [u_1 \mid u_2 \mid \cdots \mid u_m]$.

Definimos $\Sigma \in \mathbb{R}^{m \times n}$ como la matriz con entradas:

$$\Sigma_{ij} = \begin{cases} \sigma_i & \text{si } i = j \leq r, \\ 0 & \text{en caso contrario.} \end{cases}$$

Es decir, Σ tiene los valores singulares en su diagonal principal (hasta la posición $\min(m, n)$) y ceros en el resto.

Ahora verificamos que $A = U \Sigma V^*$. Para ello, calculamos las columnas de ambas matrices. Para $j = 1, \ldots, n$, la j-ésima columna de A es Av_j, mientras que la j-ésima columna de $U\Sigma V^*$ es:

$$U \Sigma V^* v_j = U \Sigma e_j,$$

donde e_j es el j-ésimo vector canónico de $\mathbb{C}^n$ (pues $V^* v_j = e_j$ al ser v_j la j-ésima columna de V).

Si $j \leq r$, entonces $\Sigma e_j = \sigma_j e_j$ (el j-ésimo vector canónico de $\mathbb{C}^m$ escalado por σ_j), de donde:

$$U \Sigma e_j = \sigma_j U e_j = \sigma_j u_j = A v_j.$$

Si $j > r$, entonces $\sigma_j = 0$, así que $\Sigma e_j = 0$, y por lo tanto:

$$U \Sigma e_j = 0.$$

Pero también $Av_j = 0$ en este caso, pues $v_j \in \ker(A)$ (ya que $A^* A v_j = \lambda_j v_j = 0$ implica $\|Av_j\|^2 = v_j^* A^* A v_j = 0$).

Por lo tanto, $A = U \Sigma V^*$, lo que completa la demostración. □

La descomposición en valores singulares no es solo una herramienta teórica, sino que tiene profundas implicaciones prácticas en la resolución de problemas numéricos. A continuación, exploramos su relación con el condicionamiento y la regularización.

2.4.1 Condicionamiento y problemas mal condicionados

El condicionamiento mide la sensibilidad de la solución de un sistema lineal $Ax = b$ ante perturbaciones. Para cuantificar esto rigurosamente, nos restringiremos a la **norma 2** (o norma espectral), que está íntimamente ligada a la SVD.

Definición 2.19 (Norma 2 y Número de Condición)**.** *La **norma 2** de un vector x es $\|x\|_2 = \sqrt{x^*x}$. La norma matricial inducida (o norma espectral) se define como:*

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}.$$

*El **número de condición espectral** de una matriz inversible A se define como:*

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2.$$

Teorema 2.20. *La norma espectral de A es igual a su mayor valor singular: $\|A\|_2 = \sigma_1(A)$.*

Demostración. Usando la SVD $A = U\Sigma V^*$, y recordando que las matrices unitarias preservan la norma 2 (es decir, $\|Ux\|_2 = \|x\|_2$ para todo x), tenemos:

$$\|Ax\|_2 = \|U\Sigma V^* x\|_2 = \|\Sigma(V^* x)\|_2.$$

Hacemos el cambio de variable $y = V^*x$. Como V es unitaria, $\|y\|_2 = \|x\|_2$. Entonces maximizar $\frac{\|Ax\|_2}{\|x\|_2}$ es equivalente a maximizar $\frac{\|\Sigma y\|_2}{\|y\|_2}$ sobre $y \neq 0$.

$$\frac{\|\Sigma y\|_2^2}{\|y\|_2^2} = \frac{\sum_{i=1}^n \sigma_i^2 |y_i|^2}{\sum_{i=1}^n |y_i|^2}.$$

Como $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, esta suma ponderada se maximiza cuando todo el peso está en σ_1 (es decir, $y = e_1$, lo que corresponde a $x = v_1$). El valor máximo es σ_1^2. Tomando raíz cuadrada, obtenemos $\|A\|_2 = \sigma_1$. □

Dado que los valores singulares de A^{-1} son $1/\sigma_n \geq \cdots \geq 1/\sigma_1$, tenemos que $\|A^{-1}\|_2 = 1/\sigma_n$. Por lo tanto:

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_n}.$$

Este número gobierna la propagación de errores. En particular, si δb es una perturbación en b, la solución perturbada $x + \delta x$ satisface $A(x + \delta x) = b + \delta b$, de donde $A\delta x = \delta b$.

Usando la SVD, podemos analizar la magnitud de δx:

$$\delta x = A^{-1}\delta b = \sum_{i=1}^n \frac{u_i^* \delta b}{\sigma_i} v_i.$$

Tomando norma 2:

$$\|\delta x\|_2^2 = \sum_{i=1}^n \frac{|u_i^* \delta b|^2}{\sigma_i^2} \leq \frac{1}{\sigma_n^2} \sum_{i=1}^n |u_i^* \delta b|^2 = \frac{1}{\sigma_n^2} \|\delta b\|_2^2.$$

Por lo tanto, $\|\delta x\|_2 \leq \frac{1}{\sigma_n}\|\delta b\|_2$.

Por otro lado, como $Ax = b$, tenemos $b = \sum_{i=1}^{n} \sigma_i (v_i^* x) u_i$, y:

$$\|b\|_2^2 = \sum_{i=1}^{n} \sigma_i^2 |v_i^* x|^2 \leq \sigma_1^2 \sum_{i=1}^{n} |v_i^* x|^2 = \sigma_1^2 \|x\|_2^2.$$

De aquí, $\|x\|_2 \geq \frac{\|b\|_2}{\sigma_1}$.

Combinando ambas desigualdades, obtenemos la cota para el error relativo:

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \frac{\frac{1}{\sigma_n}\|\delta b\|_2}{\frac{1}{\sigma_1}\|b\|_2} = \frac{\sigma_1}{\sigma_n}\frac{\|\delta b\|_2}{\|b\|_2} = \kappa_2(A)\frac{\|\delta b\|_2}{\|b\|_2}.$$

Esta cota es alcanzable (es decir, es óptima). El peor caso ocurre cuando b está alineado con la dirección de máxima amplificación (u_1) y el error δb está alineado con la dirección de mínima amplificación (u_n), que corresponde a la máxima amplificación de la inversa. En ese caso, el error relativo en la solución se amplifica exactamente por un factor $\kappa_2(A)$.

Un problema es **mal condicionado** si $\kappa_2(A) \gg 1$, lo cual ocurre cuando $\sigma_n \approx 0$. Esto es consistente con la expansión de la solución en términos de la SVD, donde los términos asociados a σ_n pequeño dominan el error.

Ejemplo 2.21 (Matriz de Hilbert)**.** *La matriz de Hilbert H_n con entradas $H_{ij} = \frac{1}{i+j-1}$ está extremadamente mal condicionada. Para $n = 10$:*

$$\kappa_2(H_{10}) \approx 1.6 \times 10^{13}.$$

Esto significa que se pierden aproximadamente 13 dígitos de precisión al resolver sistemas lineales con H_{10}, haciéndolo prácticamente imposible usando aritmética de precisión doble (que tiene solo 16 dígitos). El mal condicionamiento no depende de la norma: se puede verificar que $\kappa_1(H_{10})$ y $\kappa_\infty(H_{10})$ tienen el mismo orden de magnitud.

2.4.2 Regularización de Tikhonov

Cuando nos enfrentamos a un problema mal condicionado (o incluso singular, donde $\sigma_n = 0$), la solución directa por mínimos cuadrados o inversión es inestable. Para obtener una solución útil, se utiliza la **regularización**, que impone restricciones adicionales a la solución (como que su norma no sea demasiado grande).

La **regularización de Tikhonov** reemplaza el problema de mínimos cuadrados original $\min_x \|Ax - b\|_2^2$ por un problema penalizado:

$$\min_x \left(\|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2 \right),$$

donde $\lambda > 0$ es un parámetro de regularización que controla el balance entre el ajuste a los datos y la norma de la solución.

Utilizando la SVD de $A = U\Sigma V^*$, podemos encontrar una expresión explícita para la solución regularizada x_λ.

Para ver por qué, observemos que minimizar el funcional $J(x) = \|Ax-b\|_2^2+\lambda^2\|x\|_2^2$ es equivalente a resolver un problema de mínimos cuadrados extendido. Definimos la matriz aumentada $\tilde{A}$ y el vector aumentado $\tilde{b}$ como:

$$\tilde{A} = \begin{pmatrix} A \\ \lambda I \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Entonces:

$$\|\tilde{A}x - \tilde{b}\|_2^2 = \left\| \begin{pmatrix} Ax - b \\ \lambda x \end{pmatrix} \right\|_2^2 = \|Ax - b\|_2^2 + \|\lambda x\|_2^2 = J(x).$$

La solución de mínimos cuadrados para el sistema $\tilde{A}x = \tilde{b}$ está dada por las ecuaciones normales $\tilde{A}^*\tilde{A}x = \tilde{A}^*\tilde{b}$. Calculando los productos:

$$\tilde{A}^*\tilde{A} = \begin{pmatrix} A^* & \lambda I \end{pmatrix} \begin{pmatrix} A \\ \lambda I \end{pmatrix} = A^*A + \lambda^2 I,$$

$$\tilde{A}^*\tilde{b} = \begin{pmatrix} A^* & \lambda I \end{pmatrix} \begin{pmatrix} b \\ 0 \end{pmatrix} = A^*b.$$

Por lo tanto, el problema es equivalente a resolver el sistema de ecuaciones normales regularizado:

$$(A^*A + \lambda^2 I)x = A^*b.$$

Sustituyendo $A^*A = V\Sigma^2V^*$, tenemos:

$$(V\Sigma^2V^* + \lambda^2 VIV^*)x = V(\Sigma^2 + \lambda^2 I)V^*x = V\Sigma U^*b.$$

Despejando x:

$$x_\lambda = V(\Sigma^2 + \lambda^2 I)^{-1}\Sigma U^*b.$$

En términos de las componentes, la solución es:

$$x_\lambda = \sum_{i=1}^{n} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}(u_i^*b)v_i.$$

Comparemos esto con la solución de mínimos cuadrados estándar (para A inversible):

$$x_{LS} = \sum_{i=1}^{n} \frac{1}{\sigma_i}(u_i^*b)v_i.$$

Los términos $\frac{\sigma_i}{\sigma_i^2+\lambda^2}$ actúan como un filtro. Definimos los **factores de filtro** como $f_i = \frac{\sigma_i^2}{\sigma_i^2+\lambda^2}$:

- Si $\sigma_i \gg \lambda$, entonces $\sigma_i^2 + \lambda^2 \approx \sigma_i^2$, por lo que el término se comporta como $1/\sigma_i$ (la solución original).
- Si $\sigma_i \ll \lambda$, entonces $\sigma_i^2 + \lambda^2 \approx \lambda^2$, y el término se comporta como $\sigma_i/\lambda^2 \approx 0$.

De esta forma, la regularización de Tikhonov "apaga" suavemente las componentes asociadas a valores singulares pequeños (que son las que causan inestabilidad por ruido), a costa de introducir un pequeño sesgo en la solución exacta.

2.4.3 Aproximación de bajo rango (Teorema de Eckart-Young)

Una de las aplicaciones más potentes de la SVD es la capacidad de aproximar una matriz por otra de rango menor de manera óptima.

Teorema 2.22 (Eckart-Young-Mirsky)**.** *Sea $A \in \mathbb{C}^{m\times n}$ con descomposición en valores singulares $A = U\Sigma V^*$, donde $\Sigma = diag(\sigma_1, \ldots, \sigma_p)$ con $p = \min(m, n)$. Para cualquier $k < p$, definimos la matriz truncada A_k como:*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^* = U\Sigma_k V^*,$$

donde Σ_k es igual a Σ pero con $\sigma_{k+1} = \cdots = \sigma_p = 0$.

Entonces, A_k es la mejor aproximación de rango k a A en la norma espectral (norma 2) y en la norma de Frobenius. Es decir:

1. $\|A - A_k\|_2 = \min_{rango(B)\leq k} \|A - B\|_2 = \sigma_{k+1}$.
2. $\|A - A_k\|_F = \min_{rango(B)\leq k} \|A - B\|_F = \sqrt{\sum_{j=k+1}^{p} \sigma_j^2}$.

Demostración. Demostraremos el caso de la norma espectral (norma 2).

Primero, es claro que $\text{rango}(A_k) = k$ y que:

$$A - A_k = \sum_{i=k+1}^{p} \sigma_i u_i v_i^*.$$

La norma 2 de esta matriz es su mayor valor singular, que es σ_{k+1}. Por lo tanto, $\|A-A_k\|_2 = \sigma_{k+1}$.

Ahora debemos probar que para cualquier matriz B con $\text{rango}(B) \leq k$, $\|A - B\|_2 \geq \sigma_{k+1}$.

Como $\text{rango}(B) \leq k$, la dimensión del núcleo de B satisface:

$$\dim(\ker(B)) = n - \text{rango}(B) \geq n - k.$$

Consideremos el subespacio $W \subset \mathbb{C}^n$ generado por los primeros $k+1$ vectores singulares derechos de A:

$$W = \text{span}\{v_1, \ldots, v_{k+1}\}.$$

Tenemos $\dim(W) = k + 1$.

La intersección de W y $\ker(B)$ no puede ser trivial, ya que:

$$\dim(W \cap \ker(B)) = \dim(W) + \dim(\ker(B)) - \dim(W + \ker(B))$$

$$\geq (k + 1) + (n - k) - n = 1.$$

Por lo tanto, existe un vector unitario $z \in W \cap \ker(B)$ (con $\|z\|_2 = 1$).

Como $z \in \ker(B)$, tenemos $Bz = 0$, así que:

$$\|A - B\|_2 \geq \|(A - B)z\|_2 = \|Az - Bz\|_2 = \|Az\|_2.$$

Como $z \in W$, podemos escribir $z = \sum_{i=1}^{k+1} c_i v_i$ con $\sum |c_i|^2 = 1$. Entonces:

$$Az = \sum_{i=1}^{k+1} c_i A v_i = \sum_{i=1}^{k+1} c_i \sigma_i u_i.$$

Calculando la norma al cuadrado:

$$\|Az\|_2^2 = \sum_{i=1}^{k+1} |c_i|^2 \sigma_i^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} |c_i|^2 = \sigma_{k+1}^2.$$

(Usamos que $\sigma_i \geq \sigma_{k+1}$ para $i \leq k+1$).

Finalmente:

$$\|A - B\|_2 \geq \|Az\|_2 \geq \sigma_{k+1}.$$

Esto prueba que A_k es óptima en norma 2. □

2.5 Métodos numéricos para problemas de autovalores

La estrategia principal de muchos algoritmos para calcular autovalores consiste en transformar la matriz original en una forma más simple (diagonal o triangular) preservando su espectro. Esto se justifica por el hecho de que **las relaciones de similaridad entre matrices no alteran los autovalores**. Si $B = S^{-1}AS$, entonces $\det(B - \lambda I) = \det(S^{-1}(A - \lambda I)S) = \det(A - \lambda I)$, por lo que A y B comparten los mismos autovalores.

2.5.1 Condicionamiento de autovalores

Antes de estudiar algoritmos para calcular autovalores, es fundamental entender la sensibilidad de estos ante perturbaciones en la matriz. Si A se perturba a $A + E$, ¿cuánto cambian sus autovalores?

Definición 2.23 (Condicionamiento de un autovalor simple)**.** *Sea λ un autovalor simple de $A \in \mathbb{C}^{n\times n}$ con autovector derecho x ($Ax = \lambda x$) y autovector izquierdo y ($y^*A = \lambda y^*$), normalizados tal que $\|x\|_2 = \|y\|_2 = 1$. El **número de condición** del autovalor λ es:*

$$\kappa(\lambda) = \frac{1}{|y^*x|}.$$

Si A sufre una perturbación E de norma pequeña $\epsilon = \|E\|_2$, el autovalor perturbado $\lambda(\epsilon)$ satisface:

$$|\lambda(\epsilon) - \lambda| \leq \kappa(\lambda)\epsilon + O(\epsilon^2).$$

Esto muestra que si el autovector izquierdo es casi ortogonal al derecho ($y^*x \approx 0$), el autovalor es muy sensible a perturbaciones.

Para el caso global, tenemos el siguiente teorema fundamental.

Teorema 2.24 (Teorema de Bauer-Fike)**.** *Sea $A \in \mathbb{C}^{n\times n}$ una matriz diagonalizable, $A = V\Lambda V^{-1}$, donde $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ y V es la matriz de autovectores. Si μ es un autovalor de la matriz perturbada $A + E$, entonces existe un autovalor λ_i de A tal que:*

$$|\mu - \lambda_i| \leq \kappa_2(V)\|E\|_2,$$

donde $\kappa_2(V) = \|V\|_2\|V^{-1}\|_2$ es el número de condición de la base de autovectores.

Demostración. Si μ es autovalor de $A+E$, entonces $\det(A+E-\mu I)=0$. Si μ es también autovalor de A, la cota se cumple trivialmente (distancia 0). Supongamos que μ no es autovalor de A. Entonces $A-\mu I$ es invertible. Escribimos:

$$A+E-\mu I=(A-\mu I)+E=(A-\mu I)[I+(A-\mu I)^{-1}E].$$

Como el lado izquierdo es singular y $A-\mu I$ es invertible, necesariamente el factor $[I+(A-\mu I)^{-1}E]$ debe ser singular. Esto implica que -1 es un autovalor de $(A-\mu I)^{-1}E$. Por lo tanto, el radio espectral satisface $\rho((A-\mu I)^{-1}E)\geq 1$. Usando propiedades de la norma matricial:

$$1\leq\|(A-\mu I)^{-1}E\|_2\leq\|(A-\mu I)^{-1}\|_2\|E\|_2.$$

Ahora usamos la diagonalización $A=V\Lambda V^{-1}$:

$$(A-\mu I)^{-1}=(V(\Lambda-\mu I)V^{-1})^{-1}=V(\Lambda-\mu I)^{-1}V^{-1}.$$

Tomando normas:

$$\|(A-\mu I)^{-1}\|_2\leq\|V\|_2\|(\Lambda-\mu I)^{-1}\|_2\|V^{-1}\|_2=\kappa_2(V)\max_i\frac{1}{|\lambda_i-\mu|}.$$

Sustituyendo en la desigualdad anterior:

$$1\leq\kappa_2(V)\frac{1}{\min_i|\lambda_i-\mu|}\|E\|_2.$$

Reordenando, obtenemos:

$$\min_i|\lambda_i-\mu|\leq\kappa_2(V)\|E\|_2.$$

□

Ejemplo 2.25. ***Matrices normales****: Si A es normal, admite una base ortonormal de autovectores. Entonces V puede elegirse unitaria, por lo que $\kappa_2(V)=1$. El teorema de Bauer-Fike dice que $|\mu-\lambda_i|\leq\|E\|_2$. Los autovalores de matrices normales son perfectamente condicionados.*

Teorema 2.26 (Generalización para matrices no diagonalizables)**.** *Si A no es diagonalizable, podemos usar su forma canónica de Jordan $A=VJV^{-1}$, donde J tiene bloques de Jordan $J_k(\lambda)$ de tamaño $m_k\times m_k$. Si μ es un autovalor de $A+E$, entonces:*

$$\min_i\frac{|\mu-\lambda_i|^p}{1+|\mu-\lambda_i|^{p-1}}\leq\kappa_2(V)\|E\|_2,$$

donde p es el tamaño del mayor bloque de Jordan asociado a λ_i. Para perturbaciones pequeñas, esto implica $|\mu-\lambda_i|\approx O(\|E\|_2^{1/p})$, confirmando la sensibilidad observada en el ejemplo anterior ($p=2$).

2.5.2 El ϵ-pseudoespectro

Una herramienta fundamental para analizar la sensibilidad de los autovalores de manera global y geométrica es el **ϵ-pseudoespectro**. Mientras que el espectro $\sigma(A)$ es un conjunto discreto de puntos, el pseudoespectro es una región del plano complejo que indica dónde pueden "moverse" los autovalores bajo perturbaciones de tamaño ϵ. Es la herramienta que mejor captura la inestabilidad espectral, especialmente en matrices no normales.

Definición 2.27 (ϵ-pseudoespectro)**.** *Para una matriz $A \in \mathbb{C}^{n\times n}$ y $\epsilon > 0$, el ϵ-pseudoespectro $\Lambda_\epsilon(A)$ se define de manera equivalente como:*

1. *El conjunto de autovalores de todas las matrices perturbadas $A + E$ con $\|E\|_2 \leq \epsilon$:*

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : z \in \sigma(A + E) \text{ para alguna } E \text{ con } \|E\|_2 \leq \epsilon\}.$$

2. *El conjunto de puntos $z \in \mathbb{C}$ donde la resolvente es grande:*

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}\} \cup \sigma(A).$$

3. *Usando la descomposición en valores singulares (SVD), el conjunto donde el valor singular mínimo de $zI - A$ es pequeño:*

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \sigma_{\min}(zI - A) \leq \epsilon\}.$$

Demostración de las equivalencias. Primero mostramos que (2) $\iff$ (3). Si $z \in \sigma(A)$, entonces $\sigma_{\min}(zI - A) = 0 \leq \epsilon$, y por convención $\|(zI - A)^{-1}\|_2 = \infty \geq \epsilon^{-1}$. Si $z \notin \sigma(A)$, la matriz $zI - A$ es invertible. Sabemos que $\|M^{-1}\|_2 = 1/\sigma_{\min}(M)$. Por lo tanto, $\|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}$ es equivalente a $1/\sigma_{\min}(zI - A) \geq \epsilon^{-1}$, lo que implica $\sigma_{\min}(zI - A) \leq \epsilon$.

Ahora mostramos que (1) $\iff$ (2). ($\Rightarrow$) Supongamos que $z \in \sigma(A + E)$ con $\|E\|_2 \leq \epsilon$. Si $z \in \sigma(A)$, se cumple trivialmente. Si $z \notin \sigma(A)$, existe un vector $v \neq 0$ tal que $(A + E)v = zv$. Reordenando, $(zI - A)v = Ev$. Como $z \notin \sigma(A)$, $zI - A$ es invertible, así que $v = (zI - A)^{-1}Ev$. Tomando normas:

$$\|v\|_2 \leq \|(zI - A)^{-1}\|_2\|E\|_2\|v\|_2.$$

Dividiendo por $\|v\|_2$ (que es no nulo):

$$1 \leq \|(zI - A)^{-1}\|_2\|E\|_2 \leq \|(zI - A)^{-1}\|_2\epsilon.$$

Por lo tanto, $\|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}$.

($\Leftarrow$) Supongamos que $\|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}$. Existe un vector unitario u tal que $\|(zI - A)^{-1}u\|_2 = \|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}$. Sea $v = (zI - A)^{-1}u$. Entonces $\|v\|_2 \geq \epsilon^{-1}$ y $(zI - A)v = u$. Queremos encontrar E tal que $(A + E)v = zv$, es decir, $Ev = (zI - A)v = u$. Construimos E como una matriz de rango 1: $E = \frac{1}{\|v\|_2^2}uv^*$. Verificamos que $Ev = \frac{1}{\|v\|_2^2}u(v^*v) = u$. La norma de E es:

$$\|E\|_2 = \frac{\|u\|_2\|v\|_2}{\|v\|_2^2} = \frac{1}{\|v\|_2} \leq \epsilon.$$

Así, hemos construido una perturbación E con $\|E\|_2 \leq \epsilon$ tal que $(A + E)v = zv$, por lo que $z \in \Lambda_\epsilon(A)$. □

Esta última caracterización conecta la SVD con el problema de autovalores: z está en el pseudoespectro si la matriz $zI - A$ está "cerca" de ser singular (en norma 2).

El tamaño de $\Lambda_\epsilon(A)$ alrededor de los autovalores nos dice cuán sensibles son. Podemos usar el Teorema de Bauer-Fike y su generalización para caracterizar este tamaño:

- **Matrices Normales:** Si A es normal, $\kappa_2(V) = 1$. El Teorema de Bauer-Fike implica que $\Lambda_\epsilon(A)$ es exactamente la unión de discos cerrados de radio ϵ centrados en los autovalores λ_i.
$$\Lambda_\epsilon(A) = \bigcup_i \{z : |z - \lambda_i| \leq \epsilon\}.$$
La perturbación es lineal y controlada; el pseudoespectro apenas "engorda" los autovalores.

- **Matrices Diagonalizables no normales:** Si A es diagonalizable, $\Lambda_\epsilon(A)$ está contenido en la unión de discos de radio $\kappa_2(V)\epsilon$.
$$\Lambda_\epsilon(A) \subseteq \bigcup_i \{z : |z - \lambda_i| \leq \kappa_2(V)\epsilon\}.$$
Si la base de autovectores está mal condicionada ($\kappa_2(V) \gg 1$), el pseudoespectro puede extenderse mucho más allá de ϵ, indicando alta sensibilidad.

- **Matrices no Diagonalizables:** Cerca de un autovalor defectivo con bloque de Jordan de tamaño p, la generalización del teorema indica que el radio del pseudoespectro crece como $O(\epsilon^{1/p})$. Para ϵ pequeño, $\epsilon^{1/p} \gg \epsilon$, por lo que $\Lambda_\epsilon(A)$ es una región significativamente grande. Por ejemplo, si $p = 2$ y $\epsilon = 10^{-16}$, el autovalor puede moverse 10^{-8}.

2.5.3 Teorema de los círculos de Gershgorin

Una herramienta sencilla pero poderosa para localizar autovalores sin calcularlos es el Teorema de Gershgorin. Este teorema proporciona regiones acotadas en el plano complejo donde deben residir los autovalores, basándose únicamente en las entradas de la matriz.

Teorema 2.28 (Teorema de los círculos de Gershgorin)**.** *Sea $A \in \mathbb{C}^{n\times n}$. Todos los autovalores de A se encuentran en la unión de los n discos de Gershgorin:*

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} D_i,$$

donde cada disco D_i está definido por:

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq R_i\}, \quad \text{con } R_i = \sum_{j\neq i} |a_{ij}|.$$

Es decir, cada autovalor está a una distancia de algún elemento diagonal a_{ii} no mayor que la suma de los módulos de los elementos restantes de esa fila.

Demostración. Sea λ un autovalor de A y x su autovector asociado, $Ax = \lambda x$. Sea x_i la componente de mayor módulo de x, es decir, $|x_i| \geq |x_j|$ para todo j. Como $x \neq 0$, $|x_i| > 0$. La i-ésima ecuación del sistema $Ax = \lambda x$ es:

$$\sum_{j=1}^{n} a_{ij} x_j = \lambda x_i.$$

Reordenando para aislar el término diagonal:

$$(\lambda - a_{ii}) x_i = \sum_{j \neq i} a_{ij} x_j.$$

Tomando valor absoluto y usando la desigualdad triangular:

$$|\lambda - a_{ii}||x_i| \leq \sum_{j \neq i} |a_{ij}||x_j|.$$

Dividiendo por $|x_i|$ (recordando que $|x_j|/|x_i| \leq 1$):

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \frac{|x_j|}{|x_i|} \leq \sum_{j \neq i} |a_{ij}| = R_i.$$

Por lo tanto, λ pertenece al disco D_i. Como esto vale para cualquier autovalor (eligiendo el i correspondiente a la componente máxima de su autovector), todo autovalor está en la unión de los discos. □

Observación 2.29. *Como los autovalores de A son los mismos que los de A^T, también se puede aplicar el teorema usando sumas por columnas en lugar de filas. La intersección de las regiones obtenidas por filas y por columnas da una localización aún más precisa.*

2.5.4 El método de la potencia

El método de la potencia calcula el autovalor dominante de una matriz.

Algoritmo (Método de la potencia):
Sea $A \in \mathbb{C}^{n \times n}$ y $v_0 \in \mathbb{C}^n$ un vector inicial no nulo. Para $k = 0, 1, 2, \ldots$:

1. $w_{k+1} = A v_k$
2. $v_{k+1} = w_{k+1} / \|w_{k+1}\|_2$ (normalización)
3. $\lambda_k = v_k^* A v_k$ (cociente de Rayleigh)

Teorema 2.30 (Convergencia del método de la potencia)**.** *Sea $A \in \mathbb{C}^{n \times n}$ diagonalizable con autovalores $\lambda_1, \ldots, \lambda_n$ ordenados por magnitud decreciente: $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Sean $u_1, \ldots, u_n$ los correspondientes autovectores que forman una base de $\mathbb{C}^n$.*

Si el vector inicial v_0 tiene componente no nula en la dirección de u_1 (es decir, $v_0 = \sum_{j=1}^{n} \alpha_j u_j$ con $\alpha_1 \neq 0$), entonces:

1. *$v_k \to \pm u_1$ cuando $k \to \infty$,*
2. *$\lambda_k \to \lambda_1$ cuando $k \to \infty$,*

3. La tasa de convergencia es $O(|\lambda_2/\lambda_1|^k)$.

Demostración. Escribimos $v_0 = \sum_{j=1}^n \alpha_j u_j$ con $\alpha_1 \neq 0$. Después de k iteraciones (antes de la normalización):

$$w_k \propto A^k v_0 = A^k \sum_{j=1}^n \alpha_j u_j = \sum_{j=1}^n \alpha_j A^k u_j = \sum_{j=1}^n \alpha_j \lambda_j^k u_j.$$

Factorizando λ_1^k:

$$w_k \propto \lambda_1^k \left(\alpha_1 u_1 + \sum_{j=2}^n \alpha_j \left(\frac{\lambda_j}{\lambda_1} \right)^k u_j \right).$$

Como $|\lambda_j/\lambda_1| < 1$ para $j \geq 2$, los términos con $j \geq 2$ decaen exponencialmente:

$$\left| \left(\frac{\lambda_j}{\lambda_1} \right)^k \right| \leq \left| \frac{\lambda_2}{\lambda_1} \right|^k \to 0 \quad \text{cuando } k \to \infty.$$

Por lo tanto:

$$w_k \approx \lambda_1^k \alpha_1 u_1 + O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right),$$

y después de la normalización:

$$v_k = \frac{w_k}{\|w_k\|_2} \to \pm u_1.$$

Para el cociente de Rayleigh:

$$\lambda_k = v_k^* A v_k \to u_1^* A u_1 = u_1^* \lambda_1 u_1 = \lambda_1 \|u_1\|^2 = \lambda_1.$$

La tasa de convergencia está dominada por el cociente $|\lambda_2/\lambda_1|$, que aparece en el término de error. □

Observación 2.31. *Si $|\lambda_1|$ está bien separado de $|\lambda_2|$ (es decir, $|\lambda_2/\lambda_1| \ll 1$), el método converge rápidamente. Si $|\lambda_1| \approx |\lambda_2|$, la convergencia es muy lenta. Si hay múltiples autovalores con la misma magnitud máxima, el método puede no converger a un autovector único.*

2.5.5 El método QR

El **método QR** es un algoritmo iterativo para calcular todos los autovalores (y autovectores) de una matriz. Converge a la **descomposición de Schur** (Teorema A.13).

Algoritmo (Método QR):
Sea $A_0 = A$. Para $k = 0, 1, 2, \ldots$:

1. Factorizar $A_k = Q_k R_k$ (QR)
2. Formar $A_{k+1} = R_k Q_k = Q_k^* A_k Q_k$

Teorema 2.32 (Convergencia del método QR)**.** *Si A es diagonalizable con $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, entonces A_k converge a una matriz triangular superior cuyos elementos diagonales son los autovalores de A, ordenados por magnitud decreciente.*

La convergencia es similar al método de la potencia: el elemento (i, j) con $i > j$ converge a cero como $O(|\lambda_j/\lambda_i|^k)$.

Demostración. La demostración se basa en la conexión profunda entre el método QR y el método de la potencia aplicado simultáneamente a múltiples vectores.

Paso 1: Factorización QR acumulada.

Sea $A_k = Q_k R_k$ la factorización QR en la iteración k. Definimos las matrices acumuladas:

$$\tilde{Q}_k = Q_0 Q_1 \cdots Q_{k-1}, \quad \tilde{R}_k = R_{k-1} \cdots R_1 R_0.$$

Entonces:

$$A^k = A_{k-1} A_{k-2} \cdots A_1 A_0 = (R_{k-1} Q_{k-1})(R_{k-2} Q_{k-2}) \cdots (R_1 Q_1)(R_0 Q_0).$$

Reordenando y usando que cada $A_j = Q_j R_j$:

$$A^k = \tilde{R}_k \tilde{Q}_k.$$

Por otro lado, A^k también admite una factorización QR directa: $A^k = \hat{Q}_k \hat{R}_k$. Por unicidad de la factorización QR (cuando las diagonales de R son positivas), tenemos $\tilde{Q}_k = \hat{Q}_k$ y $\tilde{R}_k = \hat{R}_k$.

Paso 2: Conexión con la potencia simultánea.

La factorización QR de A^k calcula una base ortonormal del espacio generado por $\{A^k e_1, A^k e_2, \ldots, A^k e_n\}$, donde $\{e_j\}$ son los vectores canónicos. Esto es equivalente a aplicar el método de la potencia a cada vector canónico simultáneamente y ortogonalizarlos.

Paso 3: Análisis espectral.

Supongamos que A es diagonalizable: $A = X \Lambda X^{-1}$ donde $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ y las columnas de $X = [x_1, \ldots, x_n]$ son los autovectores. Entonces:

$$A^k = X \Lambda^k X^{-1}.$$

Sea $X = [x_1, \ldots, x_n]$ y $X^{-1} = [y_1^*, \ldots, y_n^*]^T$. La columna j-ésima de A^k es:

$$A^k e_j = X \Lambda^k X^{-1} e_j = X \Lambda^k \begin{pmatrix} y_{1j} \\ \vdots \\ y_{nj} \end{pmatrix} = \sum_{i=1}^{n} \lambda_i^k y_{ij} x_i.$$

Factorizando λ_j^k (el autovalor con mayor magnitud en la suma):

$$A^k e_j = \lambda_j^k \left(y_{jj} x_j + \sum_{i \neq j} y_{ij} \left(\frac{\lambda_i}{\lambda_j} \right)^k x_i \right).$$

Como $|\lambda_j| > |\lambda_{j+1}|$ por hipótesis, el término dominante es $\lambda_j^k y_{jj} x_j$.

Paso 4: Convergencia de A_k.

Tenemos $A_k = \tilde{Q}_k^* A \tilde{Q}_k = \hat{Q}_k^* A \hat{Q}_k$, donde $\hat{Q}_k$ contiene las columnas ortonormalizadas de $\{A^k e_1, \ldots, A^k e_n\}$.

Por el análisis anterior, la columna j-ésima de $\hat{Q}_k$ converge a una dirección proporcional al autovector x_j (después de ortogonalización con las columnas anteriores). Por lo tanto:

$$\hat{Q}_k \to X \text{ (módulo signos y fases).}$$

Entonces:

$$A_k = \hat{Q}_k^* A \hat{Q}_k \to X^* A X = X^*(X \Lambda X^{-1}) X = X^* X \Lambda (X^{-1} X) = \Lambda,$$

donde usamos que X converge a una base de autovectores ortonormalizados.

Más precisamente, el elemento (i, j) de A_k con $i > j$ tiene error:

$$|(A_k)_{ij}| = O\left(\left|\frac{\lambda_j}{\lambda_i}\right|^k\right),$$

que es exactamente análogo al método de la potencia del Teorema 2.30. □

Observación 2.33. *La similitud con el método de la potencia es profunda: el método QR puede interpretarse como aplicar simultáneamente el método de la potencia a múltiples vectores ortogonales, obteniendo todos los autovalores a la vez mediante ortogonalización continua (proceso QR).*

2.5.6 Forma de Hessenberg y transformaciones de Householder

El costo de cada iteración del método QR es $O(n^3)$ (factorización QR completa). Este costo se puede reducir dramáticamente llevando primero la matriz a **forma de Hessenberg**.

Definición 2.34 (Reflector de Householder)**.** *Dado $v \in \mathbb{C}^n$ con $\|v\|_2 = 1$, el **reflector de Householder** es:*

$$H = I - 2vv^*.$$

Esta matriz es hermítica ($H = H^*$), unitaria ($H^* H = I$), e involuciona ($H^2 = I$).

Lema 2.35. *Para cualquier $x \in \mathbb{C}^n$, existe un reflector de Householder H tal que:*

$$Hx = \pm \|x\|_2 e_1,$$

donde e_1 es el primer vector canónico. El vector v se elige como:

$$v = \frac{x \mp \|x\|_2 e_1}{\|x \mp \|x\|_2 e_1\|_2}.$$

Reducción a forma de Hessenberg

Definición 2.36 (Matriz de Hessenberg)**.** *Una matriz $H \in \mathbb{C}^{n \times n}$ es de **Hessenberg superior** si $h_{ij} = 0$ para $i > j + 1$. Es decir, es casi triangular superior, con una subdiagonal adicional.*

Teorema 2.37 (Reducción a Hessenberg)**.** *Toda matriz $A \in \mathbb{C}^{n \times n}$ puede reducirse a forma de Hessenberg mediante transformaciones unitarias de Householder:*

$$A = QHQ^*,$$

donde Q es unitaria y H es de Hessenberg superior. El costo es $O(n^3)$ operaciones.

Algoritmo (Reducción a Hessenberg):
Para $k = 1, \ldots, n - 2$:

1. Sea x la columna k de A desde la fila $k+1$ en adelante
2. Construir H_k (reflector de Householder) para anular x debajo de la primera posición
3. Actualizar $A \leftarrow H_k A H_k^*$ (preserva similitud)

Resultado: A queda en forma de Hessenberg.

Observación 2.38 (Ventajas de la forma de Hessenberg)**.** *1. Una vez que A está en forma de Hessenberg, cada iteración QR cuesta solo $O(n^2)$ en lugar de $O(n^3)$.*

*2. Para matrices hermíticas, la forma de Hessenberg es **tridiagonal**, lo que reduce aún más el costo a $O(n)$ por iteración.*

3. La forma de Hessenberg se preserva bajo iteraciones QR: si H es Hessenberg y $H = QR$, entonces RQ también es Hessenberg.

2.5.7 Cálculo de la SVD

Los métodos para calcular la SVD numéricamente suelen basarse en variantes del método QR aplicado a matrices relacionadas.

Algoritmo (SVD mediante QR - Método 1: Diagonalizar A^*A):

1. Formar $B = A^*A$ (matriz hermítica de $n \times n$)
2. Aplicar el método QR a B para obtener $B = V\Lambda V^*$ con $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$
3. Los valores singulares son $\sigma_i = \sqrt{\lambda_i}$
4. Calcular $U = AV\Sigma^{-1}$ (para las columnas correspondientes a $\sigma_i > 0$)

Problema: Este método eleva al cuadrado el número de condición: $\kappa(A^*A) = \kappa(A)^2$.
Existe un método más estable evita formar A^*A explícitamente:
Algoritmo (SVD vía matriz ampliada):

1. Formar la matriz hermítica ampliada:

$$M = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix} \in \mathbb{C}^{(m+n)\times(m+n)}$$

2. Calcular la descomposición espectral $M = W\Lambda W^*$ usando QR
3. Los autovalores de M vienen en pares $\pm\sigma_i$, donde σ_i son los valores singulares de A
4. Extraer U y V de los autovectores de M

El método más eficiente en la práctica usa bidiagonalización:

Teorema 2.39 (Bidiagonalización)**.** *Toda matriz $A \in \mathbb{C}^{m\times n}$ puede reducirse a forma bidiagonal mediante transformaciones de Householder:*

$$A = UBV^*,$$

donde U y V son unitarias y B es bidiagonal (solo diagonal y superdiagonal no nulas).

Algoritmo (SVD mediante bidiagonalización):

1. Reducir A a forma bidiagonal B usando Householder: $A = U_1 B V_1^*$ (costo $O(mn^2)$)
2. Aplicar QR iterativo a B (mucho más eficiente por ser bidiagonal)
3. Obtener $B = U_2 \Sigma V_2^*$
4. La SVD final es $A = (U_1 U_2)\Sigma(V_1 V_2)^*$

Observación 2.40 (Matrices normales y tridiagonalización)**.** *Si A es normal ($AA^* = A^*A$), en particular si es hermítica, la forma de Hessenberg obtenida por transformaciones de Householder es **tridiagonal**. Esto hace que el método QR sea especialmente eficiente para matrices hermíticas, con costo $O(n)$ por iteración en lugar de $O(n^2)$.*

2.6 Métodos iterativos basados en subespacios de Krylov

Definición 2.41 (Subespacio de Krylov)**.** *Dados $A \in \mathbb{C}^{n\times n}$ y $v \in \mathbb{C}^n$ no nulo, el **subespacio de Krylov** de dimensión k es:*

$$\mathcal{K}_k(A, v) = span\{v, Av, A^2v, \ldots, A^{k-1}v\}.$$

Observación 2.42. *El subespacio de Krylov captura la información de cómo la matriz A actúa repetidamente sobre el vector inicial v. Para una matriz invertible, existe un $m \leq n$ tal que $\mathcal{K}_m(A, v) = \mathcal{K}_{m+1}(A, v)$, momento en el cual se dice que el método ha alcanzado convergencia exacta.*

2.6.1 Método de Arnoldi

El **método de Arnoldi** es una variante del método QR que aprovecha la estructura de subespacios de Krylov para matrices grandes y dispersas.

Definición 2.43 (Método de Arnoldi)**.** *Dado $A \in \mathbb{C}^{n\times n}$ y un vector inicial q_1 unitario, el método de Arnoldi construye una base ortonormal $\{q_1, \ldots, q_k\}$ del subespacio de Krylov $\mathcal{K}_k(A, q_1)$ tal que:*

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T,$$

donde $Q_k = (q_1 \,|\, \cdots \,|\, q_k)$ y H_k es una matriz de Hessenberg de tamaño $k \times k$.

Algoritmo (Arnoldi):

1. $q_1 = v/\|v\|_2$ (normalizar vector inicial)
2. Para $j = 1, \ldots, k$:
   (a) $w = Aq_j$
   (b) Para $i = 1, \ldots, j$ (Gram-Schmidt):
   - $h_{ij} = (w, q_i)$
   - $w \leftarrow w - h_{ij} q_i$

(c) $h_{j+1,j} = \|w\|_2$

(d) $q_{j+1} = w/h_{j+1,j}$

Observación 2.44 (Aplicación a autovalores)**.** *Los autovalores de la matriz pequeña H_k (llamados **valores de Ritz**) aproximan los autovalores de A. Se puede aplicar el método QR directamente a $H_k \in \mathbb{C}^{k\times k}$ (con $k \ll n$) para obtener aproximaciones de los autovalores de A con mucho menor costo computacional.*

2.6.2 GMRES

El método GMRES (Generalized Minimal RESidual) es un algoritmo iterativo para resolver sistemas lineales $Ax = b$ con $A \in \mathbb{C}^{n\times n}$ no singular. GMRES es particularmente efectivo para matrices grandes y ralas que no necesariamente son hermíticas definidas positivas, donde otros métodos iterativos pueden fallar.

Definición 2.45 (Iteración GMRES)**.** *Sea $Ax = b$ un sistema lineal con $A \in \mathbb{C}^{n\times n}$ no singular. Dada una aproximación inicial $x_0 \in \mathbb{C}^n$, sea $r_0 = b - Ax_0$ el residuo inicial. Para cada $k \geq 1$, GMRES define la k-ésima iteración como:*

$$x_k = x_0 + z_k, \quad \text{donde } z_k \in \mathcal{K}_k(A, r_0)$$

es el vector que minimiza el residuo:

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \|r_0 - Az_k\|_2 = \min_{z\in\mathcal{K}_k(A,r_0)} \|r_0 - Az\|_2.$$

Esta caracterización de GMRES como un problema de cuadrados mínimos es fundamental: en cada iteración, buscamos la mejor aproximación dentro del subespacio de Krylov que minimiza la norma del residuo.

La implementación práctica de GMRES se basa en el proceso de Arnoldi, que construye una base ortonormal del subespacio de Krylov mediante ortogonalización de Gram-Schmidt modificada.

Teorema 2.46 (Proceso de Arnoldi)**.** *Sea $A \in \mathbb{C}^{n\times n}$ y $v \in \mathbb{C}^n$ con $\|v\|_2 = 1$. El proceso de Arnoldi construye una matriz $Q_k = [q_1, \ldots, q_k] \in \mathbb{C}^{n\times k}$ con columnas ortonormales que forman una base de $\mathcal{K}_k(A, v)$, y una matriz de Hessenberg superior $H_k \in \mathbb{C}^{k\times k}$ tal que:*

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^*,$$

o equivalentemente, en forma compacta:

$$AQ_k = Q_{k+1} \tilde{H}_k,$$

donde $Q_{k+1} = [Q_k, q_{k+1}]$ y $\tilde{H}_k \in \mathbb{C}^{(k+1)\times k}$ es una matriz de Hessenberg superior con entradas h_{ij} para $i \leq j + 1$.

Demostración. El proceso es constructivo. Iniciando con $q_1 = v$, para $j = 1, \ldots, k$:

1. Calcular $w = Aq_j$.

2. Ortogonalizar contra $q_1, \ldots, q_j$: para $i = 1, \ldots, j$,

$$h_{ij} = (w, q_i), \quad w \leftarrow w - h_{ij} q_i.$$

3. Normalizar: $h_{j+1,j} = \|w\|_2$, $q_{j+1} = w/h_{j+1,j}$.

Por construcción, $\{q_1, \ldots, q_k\}$ es ortonormal y genera $\mathcal{K}_k(A, v)$. La relación $AQ_k = Q_{k+1}\tilde{H}_k$ se verifica directamente de la construcción. □

Teorema 2.47 (Formulación reducida de GMRES)**.** *Con la descomposición de Arnoldi $AQ_k = Q_{k+1}\tilde{H}_k$ y $r_0 = \beta q_1$ donde $\beta = \|r_0\|_2$, el problema de minimización de GMRES:*

$$\min_{z \in \mathcal{K}_k(A, r_0)} \|r_0 - Az\|_2$$

es equivalente al problema de cuadrados mínimos de dimensión reducida:

$$\min_{y \in \mathbb{C}^k} \|\beta e_1 - \tilde{H}_k y\|_2,$$

donde $e_1 \in \mathbb{C}^{k+1}$ es el primer vector canónico. La solución está relacionada por $z_k = Q_k y_k$.

Demostración. Como $z \in \mathcal{K}_k(A, r_0)$, podemos escribir $z = Q_k y$ para algún $y \in \mathbb{C}^k$. Entonces:

$$\|r_0 - Az\|_2 = \|\beta q_1 - AQ_k y\|_2 = \|\beta q_1 - Q_{k+1}\tilde{H}_k y\|_2.$$

Multiplicando por Q_{k+1}^* (que preserva la norma pues Q_{k+1} tiene columnas ortonormales):

$$\|r_0 - Az\|_2 = \|Q_{k+1}^*(\beta q_1 - Q_{k+1}\tilde{H}_k y)\|_2 = \|\beta e_1 - \tilde{H}_k y\|_2,$$

donde usamos que $Q_{k+1}^* q_1 = e_1$. Este problema de tamaño $(k+1) \times k$ se resuelve eficientemente mediante factorización QR de $\tilde{H}_k$. □

El costo computacional de GMRES después de k iteraciones es:

- **Productos matriz-vector**: k multiplicaciones Aq_j, cada una costando $O(n^2)$ para matrices densas u $O(n)$ para matrices dispersas.
- **Ortogonalizaciones**: $O(k^2 n)$ operaciones para mantener ortogonalidad con todos los vectores previos.
- **Problema de cuadrados mínimos**: $O(k^2)$ para resolver $\min \|\beta e_1 - \tilde{H}_k y\|_2$.
- **Memoria**: Almacenar Q_k requiere $O(kn)$ memoria.

Para problemas grandes, el crecimiento en memoria y costo de ortogonalización motiva el uso de **GMRES reiniciado** (GMRES(m)), que reinicia el método cada m iteraciones, sacrificando optimalidad teórica por eficiencia práctica.

Análisis de convergencia

Teorema 2.48 (Cota de convergencia óptima)**.** *Después de k iteraciones de GMRES, el residuo satisface:*

$$\frac{\|r_k\|_2}{\|r_0\|_2} = \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \|p(A)r_0\|_2,$$

donde $\mathcal{P}_k$ denota el conjunto de polinomios de grado a lo más k.

Demostración. De $x_k = x_0 + z_k$ con $z_k \in \mathcal{K}_k(A, r_0)$, tenemos:

$$r_k = r_0 - Az_k.$$

Como $z_k \in \mathcal{K}_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, existe un polinomio q de grado $\leq k-1$ tal que $z_k = q(A)r_0$. Entonces:

$$r_k = r_0 - Aq(A)r_0 = (I - Aq(A))r_0 = p(A)r_0,$$

donde $p(z) = 1 - zq(z)$ es un polinomio de grado $\leq k$ con $p(0) = 1$. La minimización en GMRES garantiza que elegimos el mejor polinomio posible. □

Teorema 2.49 (Cota espectral para matrices diagonalizables)**.** *Si A es diagonalizable, $A = X\Lambda X^{-1}$ con autovalores $\lambda_1, \ldots, \lambda_n$, entonces:*

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa(X) \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \max_{i=1,\ldots,n} |p(\lambda_i)|,$$

donde $\kappa(X) = \|X\|_2\|X^{-1}\|_2$ es el número de condición de la matriz de autovectores.

Demostración. Del Teorema 2.48, $r_k = p(A)r_0$ para algún polinomio p con $p(0) = 1$. Usando la diagonalización:

$$p(A) = Xp(\Lambda)X^{-1},$$

donde $p(\Lambda) = \text{diag}(p(\lambda_1), \ldots, p(\lambda_n))$. Entonces:

$$\|r_k\|_2 = \|p(A)r_0\|_2 \leq \|X\|_2\|p(\Lambda)\|_2\|X^{-1}\|_2\|r_0\|_2 = \kappa(X) \max_i |p(\lambda_i)|\|r_0\|_2.$$

□

Corolario 2.50 (Convergencia para matrices normales)**.** *Si A es normal ($A^*A = AA^*$), entonces X es unitaria y $\kappa(X) = 1$. En este caso:*

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|,$$

donde $\sigma(A)$ denota el espectro de A.

Teorema 2.51 (Convergencia para autovalores agrupados)**.** *Si los autovalores de A están agrupados en m clusters distintos con $m \ll n$, entonces GMRES converge en a lo más m iteraciones (ignorando errores de redondeo).*

Observación 2.52 (Interpretación geométrica)**.** *La cota del Teorema 2.49 muestra que GMRES converge rápido cuando:*

- *Los autovalores están bien agrupados (permite polinomios que son pequeños en el espectro).*
- *La matriz está bien condicionada espectralmente ($\kappa(X) \approx 1$).*
- *Los autovalores están alejados del origen (facilita la construcción de polinomios pequeños).*

Chapter 3

Interpolación

La interpolación polinomial es uno de los problemas centrales del análisis numérico y el álgebra computacional. Su objetivo es reconstruir una función (un polinomio) a partir de información parcial (sus valores en ciertos puntos). Aunque el problema puede plantearse de forma elemental, su estructura matemática es rica y conecta áreas aparentemente dispares como el análisis de Fourier, la integración numérica y la criptografía moderna.

Desde una perspectiva algebraica, la interpolación se entiende mejor observando la relación entre el espacio de polinomios y el espacio de sus valores. Si consideramos el espacio vectorial $\mathcal{P}_n(\mathbb{F})$ de polinomios de grado $\leq n$ sobre un cuerpo $\mathbb{F}$ (como $\mathbb{R}$, $\mathbb{C}$ o $\mathbb{Z}_p$), la operación de evaluar un polinomio en $n+1$ puntos distintos define una transformación lineal hacia $\mathbb{F}^{n+1}$.

El resultado fundamental de este capítulo es que esta transformación es invertible (un **isomorfismo** de espacios vectoriales). Esto garantiza que el problema de interpolación siempre tiene solución única. Más aún, diferentes bases del espacio de polinomios nos dan diferentes algoritmos y perspectivas: la base canónica lleva a sistemas de Vandermonde, mientras que la **base de Lagrange** diagonaliza el problema, haciendo que la recuperación de los coeficientes sea trivial.

3.1 El problema de interpolación como problema lineal

Hasta ahora hemos trabajado principalmente con los espacios vectoriales $\mathbb{R}^n$ y $\mathbb{C}^n$. Ahora introducimos otro objeto fundamental: los polinomios.

3.1.1 Polinomios sobre un cuerpo

Definición 3.1 (Polinomios sobre un cuerpo)**.** *Un polinomio en la variable x con coeficientes en un cuerpo $\mathbb{F}$ es una expresión formal:*

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n = \sum_{k=0}^{n} a_k x^k,$$

donde $n \geq 0$ es un entero y $a_k \in \mathbb{F}$. El conjunto de todos los polinomios se denota $\mathbb{F}[x]$.

En $\mathbb{F}[x]$ definimos las siguientes operaciones naturales:

- **Suma:** Se suman coeficiente a coeficiente.
- **Producto por escalar:** Se multiplica cada coeficiente por el escalar.
- **Multiplicación de polinomios:** Se usa la propiedad distributiva y la regla $x^i \cdot x^j = x^{i+j}$.
- **Evaluación:** Dado $\alpha \in \mathbb{F}$, $p(\alpha)$ es el elemento de $\mathbb{F}$ que resulta de sustituir x por α.

Para el problema de interpolación, trabajaremos con un subconjunto de dimensión finita:

Definición 3.2 (Espacio $\mathcal{P}_n$)**.** *Denotamos por $\mathcal{P}_n(\mathbb{F})$ (o simplemente $\mathcal{P}_n$) al conjunto de polinomios de grado a lo sumo n:*

$$\mathcal{P}_n = \{p \in \mathbb{F}[x] : \deg(p) \leq n\}.$$

Ejemplo 3.3 (Polinomios en diferentes cuerpos)**.** *La naturaleza del cuerpo $\mathbb{F}$ determina las propiedades de los polinomios:*

- ***Sobre** $\mathbb{C}$**:** Los coeficientes son números complejos. Por ejemplo, $p(x) = ix^2 + (1-i)x + 3 \in \mathcal{P}_2(\mathbb{C})$. Un caso de especial importancia son los polinomios de la forma $x^n - 1$, cuyas raíces son las **raíces n-ésimas de la unidad**: $1, \omega, \omega^2, \ldots, \omega^{n-1}$ donde $\omega = e^{2\pi i/n}$. Estas juegan un papel central en la Transformada Discreta de Fourier (DFT).*
- ***Sobre** $\mathbb{Z}_p$**:** Si $\mathbb{F} = \mathbb{Z}_p$ (enteros módulo un primo p), la aritmética de los coeficientes es modular. Por ejemplo, en $\mathbb{Z}_5[x]$, el polinomio $p(x) = 2x^2 + 4x + 1$ cumple que $p(3) = 2(3^2) + 4(3) + 1 = 18 + 12 + 1 = 31 \equiv 1 \pmod 5$.*

Teorema 3.4 (Estructura de espacio vectorial)**.** *$\mathcal{P}_n$ es un espacio vectorial de dimensión $n+1$. La base más simple es la **base canónica** o **monomial**: $\{1, x, x^2, \ldots, x^n\}$.*

3.1.2 Formulación del problema de interpolación polinomial

Definición 3.5 (Problema de interpolación polinomial)**.** *Dados $n+1$ puntos distintos $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y$ con $x_i \neq x_j$ para $i \neq j$, el problema consiste en encontrar un polinomio $p \in \mathcal{P}_n$ tal que:*

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n.$$

Interpretación lineal: Podemos definir una transformación lineal $\Phi : \mathcal{P}_n \to \mathbb{F}^{n+1}$ que toma un polinomio y devuelve sus valores en los nodos:

$$\Phi(p) = (p(x_0), p(x_1), \ldots, p(x_n)).$$

El problema de interpolación equivale a invertir esta transformación: dado un vector de valores $\mathbf{y} \in \mathbb{F}^{n+1}$, queremos encontrar $p = \Phi^{-1}(\mathbf{y})$. Como ambos espacios tienen la misma dimensión $(n+1)$, basta demostrar que Φ es inyectiva o sobreyectiva para asegurar que es un isomorfismo (invertible).

Teorema 3.6 (Existencia y unicidad)**.** *Existe un único polinomio $p \in \mathcal{P}_n$ que satisface las condiciones de interpolación.*

Demostración. **Método 1 (Matriz de Vandermonde):** Si escribimos $p(x) = \sum_{k=0}^{n} a_k x^k$ en la base canónica, las condiciones de interpolación se traducen en el sistema lineal:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

La matriz de Vandermonde V tiene determinante:

$$\det(V) = \prod_{0 \le i < j \le n} (x_j - x_i) \neq 0,$$

por lo que el sistema tiene solución única.

Método 2 (Argumento dimensional): Si existieran dos polinomios $p, q \in \mathcal{P}_n$ que interpolan los mismos datos, entonces $r = p - q \in \mathcal{P}_n$ satisface $r(x_i) = 0$ para $i = 0, \ldots, n$. Un polinomio no nulo de grado $\le n$ tiene a lo sumo n raíces, pero r tiene $n+1$ raíces. Por lo tanto, $r \equiv 0$. □

3.1.3 La base de Lagrange y la forma baricéntrica

Aunque la base canónica $\{1, x, \ldots, x^n\}$ es natural, genera sistemas mal condicionados (Vandermonde). La **base de Lagrange** diagonaliza el problema de interpolación:

Definición 3.7 (Polinomios base de Lagrange)**.** *Dados nodos $x_0, \ldots, x_n$:*

$$\ell_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}, \quad k = 0, \ldots, n.$$

Teorema 3.8 (Base de Lagrange)**.** *Los polinomios $\{\ell_0, \ldots, \ell_n\}$ forman una base de $\mathcal{P}_n$ con:*

1. ***Propiedad de Kronecker**: $\ell_k(x_j) = \delta_{kj}$.*
2. ***Partición de la unidad**: $\sum_{k=0}^{n} \ell_k(x) = 1$.*

El polinomio interpolante es:

$$p(x) = \sum_{k=0}^{n} y_k \ell_k(x).$$

Interpretación geométrica: En la base de Lagrange, la matriz de evaluación es la identidad. Esto convierte el problema de interpolación en trivial: los "coeficientes" son directamente los valores y_k.

Para estabilidad numérica y eficiencia, se usa la forma baricéntrica. Definiendo los **pesos baricéntricos**:

$$w_k = \frac{1}{\prod_{j \neq k}(x_k - x_j)},$$

el interpolante se escribe:

$$p(x) = \frac{\sum_{k=0}^{n} \frac{w_k y_k}{x - x_k}}{\sum_{k=0}^{n} \frac{w_k}{x - x_k}}.$$

Ventajas: (1) Los pesos se calculan una vez ($O(n^2)$). (2) Cada evaluación cuesta $O(n)$. (3) Agregar nodos es simple: $w_k^{\text{nuevo}} = w_k^{\text{viejo}}/(x_k - x_{n+1})$.

3.2 Interpolación en el plano complejo: La DFT

Un caso particularmente importante de interpolación ocurre cuando trabajamos sobre el cuerpo de los números complejos $\mathbb{C}$ y elegimos como nodos las **raíces n-ésimas de la unidad**.

Esta elección no es arbitraria: la geometría regular de estos puntos sobre el círculo unitario dota al problema de interpolación de una estructura algebraica excepcional. La matriz de Vandermonde asociada se convierte, salvo un factor de escala, en una matriz unitaria. Esto significa que el problema de interpolación (y su inverso, la evaluación) se vuelve numéricamente muy estable y computacionalmente muy eficiente.

Este caso específico de interpolación es lo que conocemos como la **Transformada Discreta de Fourier (DFT)**.

3.2.1 Raíces de la unidad y su estructura algebraica

Definición 3.9 (Raíces n-ésimas de la unidad)**.** *Sea $n \geq 1$ un entero. Las **raíces n-ésimas de la unidad** son las soluciones de la ecuación $z^n = 1$ en $\mathbb{C}$:*

$$\omega_n^k = e^{2\pi i k/n}, \quad k = 0, 1, \ldots, n-1.$$

El conjunto $\mu_n = \{1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1}\}$ forma un grupo cíclico bajo multiplicación.

Notación: Cuando n está claro del contexto, escribimos simplemente $\omega = \omega_n = e^{2\pi i/n}$.

Teorema 3.10 (Propiedades de las raíces de la unidad)**.** *Las raíces n-ésimas de la unidad satisfacen:*

1. ***Ecuación polinomial**: Son las n raíces (distintas) de $z^n - 1 = 0$.*
2. ***Periodicidad**: $\omega^{n+k} = \omega^k$ para todo $k \in \mathbb{Z}$.*
3. ***Conjugación**: $\overline{\omega^k} = \omega^{-k} = \omega^{n-k}$.*
4. ***Ortogonalidad discreta**: $\sum_{j=0}^{n-1} (\omega^k)^j = \begin{cases} n & \text{si } k \equiv 0 \pmod{n}, \\ 0 & \text{si } k \not\equiv 0 \pmod{n}. \end{cases}$*

Demostración. (1)-(3) son inmediatas de la definición exponencial.

(4) Si $k \equiv 0 \pmod{n}$, entonces $\omega^k = 1$ y la suma es $n \cdot 1 = n$. Si $k \not\equiv 0 \pmod{n}$, entonces $\omega^k \neq 1$ y usando la suma geométrica:

$$\sum_{j=0}^{n-1} (\omega^k)^j = \frac{1 - (\omega^k)^n}{1 - \omega^k} = \frac{1 - \omega^{kn}}{1 - \omega^k} = \frac{1 - 1}{1 - \omega^k} = 0,$$

donde usamos $\omega^{kn} = (\omega^n)^k = 1^k = 1$. □

3.2.2 Matriz de Vandermonde en raíces de la unidad

Consideremos el problema de interpolación con nodos en las raíces n-ésimas de la unidad: $\{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$.

Definición 3.11 (Matriz DFT como Vandermonde)**.** *La matriz de Vandermonde evaluada en las raíces n-ésimas de la unidad es:*

$$F_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}.$$

Elemento (j,k)*:* $[F_n]_{jk} = \omega^{jk}$ *para* $j,k = 0,1,\ldots,n-1$.

Observación clave: Esta es **exactamente** la matriz de la Transformada Discreta de Fourier (salvo convenciones de normalización).

Teorema 3.12 (DFT como interpolación en raíces de la unidad)**.** *Sea* $\mathbf{p} = (p_0, p_1, \ldots, p_{n-1})^T$ *el vector de coeficientes de un polinomio* $p(x) = \sum_{k=0}^{n-1} p_k x^k$*. Sus evaluaciones en las raíces n-ésimas de la unidad son:*

$$\mathbf{y} = F_n \mathbf{p},$$

donde $y_j = p(\omega^j)$ *para* $j = 0,1,\ldots,n-1$.

Inversamente, dados valores $\mathbf{y}$ *en las raíces de la unidad, los coeficientes se recuperan mediante:*

$$\mathbf{p} = F_n^{-1} \mathbf{y}.$$

Demostración. Evaluando $p(x)$ en $x = \omega^j$:

$$y_j = p(\omega^j) = \sum_{k=0}^{n-1} p_k (\omega^j)^k = \sum_{k=0}^{n-1} p_k \omega^{jk} = [F_n \mathbf{p}]_j.$$

Como F_n es invertible (Vandermonde con nodos distintos), la interpolación inversa es $\mathbf{p} = F_n^{-1}\mathbf{y}$. □

3.2.3 Unitariedad de la matriz DFT

Una propiedad extraordinaria de F_n es que es **unitaria** salvo un factor de escala.

Teorema 3.13 (Unitariedad de F_n)**.** *La matriz* F_n *satisface:*

$$F_n^* F_n = n I_n,$$

donde F_n^* *es la adjunta (transpuesta conjugada). Equivalentemente,* $\frac{1}{\sqrt{n}} F_n$ *es unitaria.*

Demostración. Calculamos el elemento (k, ℓ) de $F_n^* F_n$:

$$\begin{aligned}
[F_n^* F_n]_{k\ell} &= \sum_{j=0}^{n-1} \overline{[F_n]_{jk}} [F_n]_{j\ell} \\
&= \sum_{j=0}^{n-1} \overline{\omega^{jk}} \omega^{j\ell} \\
&= \sum_{j=0}^{n-1} \omega^{-jk} \omega^{j\ell} \\
&= \sum_{j=0}^{n-1} \omega^{j(\ell-k)} \\
&= \begin{cases} n & \text{si } \ell = k \quad \text{(por ortogonalidad, caso } k \equiv 0\text{)}, \\ 0 & \text{si } \ell \neq k \quad \text{(por ortogonalidad, caso } k \not\equiv 0\text{)}. \end{cases}
\end{aligned}$$

Por lo tanto, $F_n^* F_n = n I_n$. □

Corolario 3.14 (Inversa de la DFT)**.** *La matriz inversa de F_n es:*

$$F_n^{-1} = \frac{1}{n} F_n^*.$$

Explícitamente, $[F_n^{-1}]_{jk} = \frac{1}{n}\omega^{-jk}$.

Interpretación geométrica: Las columnas de F_n son **vectores ortogonales** en $\mathbb{C}^n$ (con norma $\sqrt{n}$). Estas columnas son los **modos de Fourier**:

$$\mathbf{v}_k = (\omega^{0\cdot k}, \omega^{1\cdot k}, \omega^{2\cdot k}, \ldots, \omega^{(n-1)\cdot k})^T, \quad k = 0, \ldots, n-1.$$

La ortogonalidad $\langle \mathbf{v}_k, \mathbf{v}_\ell \rangle = n\delta_{k\ell}$ es consecuencia directa de la propiedad de ortogonalidad discreta de las raíces de la unidad (Teorema anterior, propiedad 4).

3.2.4 Transformada Discreta de Fourier (DFT)

Definición 3.15 (DFT y su inversa)**.** *Dada una secuencia $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})^T \in \mathbb{C}^n$, su **Transformada Discreta de Fourier (DFT)** es:*

$$\mathbf{X} = F_n \mathbf{x}, \quad \textit{es decir, } X_k = \sum_{j=0}^{n-1} x_j \omega^{jk}, \quad k = 0, 1, \ldots, n-1.$$

*La **DFT inversa (IDFT)** es:*

$$\mathbf{x} = F_n^{-1} \mathbf{X} = \frac{1}{n} F_n^* \mathbf{X}, \quad \textit{es decir, } x_j = \frac{1}{n} \sum_{k=0}^{n-1} X_k \omega^{-jk}.$$

Conexión con interpolación:

- **DFT directa $\mathbf{X} = F_n\mathbf{x}$**: Evalúa el polinomio $p(z) = \sum_{j=0}^{n-1} x_j z^j$ (coeficientes $\mathbf{x}$) en las raíces de la unidad $\{\omega^k\}$. Resultado: evaluaciones $\mathbf{X}$.
- **IDFT $\mathbf{x} = F_n^{-1}\mathbf{X}$**: Interpola las evaluaciones $\mathbf{X}$ en las raíces de la unidad para recuperar los coeficientes $\mathbf{x}$ del polinomio.

Observación 3.16 (Convenciones de normalización)**.** *Diferentes textos usan distintas convenciones:*

- ***Convención asimétrica** (usada aquí y en numpy): Factor $1/n$ solo en la inversa.*
- ***Convención simétrica**: Factor $1/\sqrt{n}$ en ambas direcciones (hace a $F_n/\sqrt{n}$ unitaria).*
- ***Convención de ingeniería**: A veces se usa $\omega = e^{-2\pi i/n}$ (signo opuesto).*

Lo importante es la relación $F_n^{-1} = \frac{1}{n}F_n^*$, que es invariante.

3.2.5 Transformada Rápida de Fourier (FFT)

El cálculo directo de $\mathbf{X} = F_n\mathbf{x}$ requiere $O(n^2)$ operaciones (producto matriz-vector). La **FFT** (Cooley-Tukey, 1965) reduce esto a $O(n \log n)$ mediante factorización recursiva.

Idea clave: Explotar la estructura de F_n para descomponer la DFT de tamaño n en dos DFTs de tamaño $n/2$ (asumiendo n par).

Teorema 3.17 (Algoritmo FFT (Cooley-Tukey radix-2))**.** *Supongamos $n = 2m$ (par). Separamos la secuencia $\mathbf{x}$ en índices pares e impares:*

$$\mathbf{x}_{par} = (x_0, x_2, x_4, \ldots, x_{n-2}), \quad \mathbf{x}_{impar} = (x_1, x_3, x_5, \ldots, x_{n-1}).$$

Sea $X_k = \sum_{j=0}^{n-1} x_j \omega_n^{jk}$. Separando en pares e impares:

$$\begin{aligned} X_k &= \sum_{j=0}^{m-1} x_{2j}\omega_n^{2jk} + \sum_{j=0}^{m-1} x_{2j+1}\omega_n^{(2j+1)k} \\ &= \sum_{j=0}^{m-1} x_{2j}(\omega_n^2)^{jk} + \omega_n^k \sum_{j=0}^{m-1} x_{2j+1}(\omega_n^2)^{jk}. \end{aligned}$$

Observando que $\omega_n^2 = e^{4\pi i/n} = e^{2\pi i/m} = \omega_m$:

$$X_k = E_k + \omega_n^k O_k,$$

donde:

- *$E_k = \sum_{j=0}^{m-1} x_{2j}\omega_m^{jk}$ es la DFT de tamaño m de $\mathbf{x}_{par}$.*
- *$O_k = \sum_{j=0}^{m-1} x_{2j+1}\omega_m^{jk}$ es la DFT de tamaño m de $\mathbf{x}_{impar}$.*

Usando la simetría $E_{k+m} = E_k$ y $\omega_n^{k+m} = -\omega_n^k$:

$$\begin{cases} X_k = E_k + \omega_n^k O_k, & k = 0, \ldots, m-1, \\ X_{k+m} = E_k - \omega_n^k O_k, & k = 0, \ldots, m-1. \end{cases}$$

Análisis de complejidad: Sea $T(n)$ el tiempo para calcular una DFT de tamaño n. El algoritmo FFT satisface:

$$T(n) = 2T(n/2) + O(n),$$

pues calcula dos DFTs de tamaño $n/2$ más $O(n)$ sumas/multiplicaciones. Por el Teorema Maestro:

$$T(n) = O(n \log n).$$

Para $n = 2^{10} = 1024$:

- DFT directa: $\sim n^2 = 10^6$ operaciones.
- FFT: $\sim n \log_2 n = 1024 \cdot 10 \approx 10^4$ operaciones.

¡Aceleración de 100x!

Implementación: En Python, usar `numpy.fft.fft(x)` y `numpy.fft.ifft(X)`. Ver el archivo `interpolacion_algebraica.py` para una implementación didáctica de DFT vía Vandermonde (sin FFT, $O(n^2)$) y comparación con `numpy.fft`.

3.2.6 Convolución discreta

Una de las aplicaciones más poderosas de la DFT y, por extensión, de la FFT, es el **Teorema de Convolución**. Este teorema establece que la convolución circular de dos secuencias en el dominio del tiempo se convierte en una simple multiplicación punto a punto de sus DFT en el dominio de la frecuencia.

Si $x[n]$ y $y[n]$ son dos secuencias finitas de longitud $N+1$, su convolución circular $z[n] = (x * y)[n]$ está definida como:

$$z[n] = \sum_{m=0}^{N} x[m]y[(n-m) \mod (N+1)]$$

para $n = 0, 1, \ldots, N$.

El Teorema de Convolución Discreta nos dice que:

$$\text{DFT}\{x[n] * y[n]\} = \text{DFT}\{x[n]\} \cdot \text{DFT}\{y[n]\}$$

O, en notación de Fourier:

$$Z[k] = X[k] \cdot Y[k]$$

Donde $X[k]$, $Y[k]$ y $Z[k]$ son las DFT de $x[n]$, $y[n]$ y $z[n]$ respectivamente.

El Teorema de Convolución Discreta establece que la Transformada Discreta de Fourier (DFT) de la convolución circular de dos secuencias es igual al producto punto a punto de las DFT de cada secuencia.

Paso 1: Aplicar la DFT a la Convolución Queremos calcular la DFT de $z[n]$, es decir, $Z[k]$:

$$Z[k] = \sum_{n=0}^{N} z[n] e^{-j\frac{2\pi}{N+1}kn}$$

Sustituimos la definición de $z[n]$:

$$Z[k] = \sum_{n=0}^{N} \left(\sum_{m=0}^{N} x[m] y[(n-m) \mod (N+1)] \right) e^{-j\frac{2\pi}{N+1}kn}$$

Intercambiamos el orden de las sumatorias:

$$Z[k] = \sum_{m=0}^{N} x[m] \sum_{n=0}^{N} y[(n-m) \mod (N+1)] e^{-j\frac{2\pi}{N+1}kn}$$

Paso 2: Cambio de Variable Hacemos un cambio de variable en la sumatoria interna. Sea $p = (n-m) \mod (N+1)$. Cuando n recorre de 0 a N, p también recorrerá de 0 a N. De $p = (n-m) \mod (N+1)$, podemos expresar $n = (p+m) \mod (N+1)$.

Sustituimos esto en el término exponencial:

$$e^{-j\frac{2\pi}{N+1}kn} = e^{-j\frac{2\pi}{N+1}k(p+m)} = e^{-j\frac{2\pi}{N+1}kp} e^{-j\frac{2\pi}{N+1}km}$$

Ahora, la sumatoria interna se convierte en:

$$\sum_{p=0}^{N} y[p] e^{-j\frac{2\pi}{N+1}k(p+m)} = \sum_{p=0}^{N} y[p] e^{-j\frac{2\pi}{N+1}kp} e^{-j\frac{2\pi}{N+1}km}$$

Notamos que $e^{-j\frac{2\pi}{N+1}km}$ no depende de p, por lo que puede salir de esta sumatoria:

$$e^{-j\frac{2\pi}{N+1}km} \sum_{p=0}^{N} y[p] e^{-j\frac{2\pi}{N+1}kp}$$

La sumatoria restante es, por definición, la **DFT de** $y[n]$, es decir, $Y[k]$:

$$e^{-j\frac{2\pi}{N+1}km} Y[k]$$

Paso 3: Reagrupar los Términos Sustituimos este resultado de nuevo en la expresión de $Z[k]$:

$$Z[k] = \sum_{m=0}^{N} x[m] \left(Y[k] e^{-j\frac{2\pi}{N+1}km} \right)$$

Como $Y[k]$ no depende de m, puede salir de la sumatoria externa:

$$Z[k] = Y[k] \sum_{m=0}^{N} x[m] e^{-j\frac{2\pi}{N+1}km}$$

Y la sumatoria restante es, por definición, la **DFT de** $x[n]$, es decir, $X[k]$:

$$Z[k] = Y[k] \cdot X[k]$$

El teorema de convolución tiene numerosas aplicaciones prácticas en procesamiento de señales, resolución de ecuaciones y simulación numérica. A continuación presentamos algunas de las más importantes.

3.2.7 Filtrado digital de señales

El filtrado de señales es una de las aplicaciones más directas del teorema de convolución. Un **filtro digital** se representa mediante una secuencia $h[n]$ llamada *respuesta al impulso* del filtro. La señal de salida $y[n]$ se obtiene convolucionando la señal de entrada $x[n]$ con $h[n]$:

$$y[n] = (h * x)[n] = \sum_{m=0}^{N} h[m]x[(n-m) \mod (N+1)].$$

Calcular esta convolución directamente requiere $O(N^2)$ operaciones. Sin embargo, usando el teorema de convolución y la FFT, podemos reducir la complejidad a $O(N \log N)$:

Algoritmo de filtrado rápido vía FFT:

1. Calcular $H[k] = \text{FFT}\{h[n]\}$ (complejidad $O(N \log N)$).
2. Calcular $X[k] = \text{FFT}\{x[n]\}$ (complejidad $O(N \log N)$).
3. Calcular $Y[k] = H[k] \cdot X[k]$ (multiplicación punto a punto, complejidad $O(N)$).
4. Calcular $y[n] = \text{IFFT}\{Y[k]\}$ (complejidad $O(N \log N)$).

El costo total es $O(N \log N)$, mucho menor que el método directo para N grande.

Ejemplo 3.18 (Filtro pasa-bajos)**.** *Un filtro pasa-bajos ideal tiene respuesta en frecuencia:*

$$H[k] = \begin{cases} 1 & si\ |k| \le k_c \\ 0 & si\ |k| > k_c \end{cases}$$

donde k_c es la frecuencia de corte. *Este filtro elimina las componentes de alta frecuencia de la señal, dejando solo las frecuencias bajas (hasta k_c). En la práctica, se suele usar filtros con transiciones suaves en lugar de este corte abrupto para evitar el fenómeno de Gibbs.*

3.2.8 Multiplicación rápida de polinomios

Sean $p(x) = \sum_{k=0}^{N} p_k x^k$ y $q(x) = \sum_{k=0}^{N} q_k x^k$ dos polinomios de grado a lo sumo N. Su producto $r(x) = p(x)q(x)$ es un polinomio de grado a lo sumo $2N$:

$$r(x) = \sum_{k=0}^{2N} r_k x^k, \qquad r_k = \sum_{j=0}^{k} p_j q_{k-j}.$$

Observemos que los coeficientes r_k son precisamente la **convolución lineal** (no circular) de las secuencias $\{p_k\}$ y $\{q_k\}$. Para calcular la convolución lineal usando FFT, debemos trabajar con convolución circular de longitud suficiente para evitar el *aliasing circular*:

Algoritmo de multiplicación rápida:

1. Extender ambas secuencias con ceros hasta longitud $M \geq 2N+1$ (típicamente $M = 2^{\lceil \log_2(2N+1) \rceil}$ para eficiencia de FFT).
2. Calcular $P[k] = \text{FFT}\{p_0, \ldots, p_N, 0, \ldots, 0\}$ (longitud M).
3. Calcular $Q[k] = \text{FFT}\{q_0, \ldots, q_N, 0, \ldots, 0\}$ (longitud M).
4. Calcular $R[k] = P[k] \cdot Q[k]$.
5. Calcular $\{r_k\} = \text{IFFT}\{R[k]\}$.

La complejidad es $O(M \log M) = O(N \log N)$, frente a $O(N^2)$ del método directo (multiplicación término a término).

3.3 Interpolación sobre cuerpos finitos

Hasta ahora hemos trabajado sobre $\mathbb{R}$ o $\mathbb{C}$, pero la teoría de interpolación es completamente algebraica y se extiende a cualquier **cuerpo** (field). En particular, la interpolación sobre **cuerpos finitos** $\mathbb{Z}_p$ (aritmética módulo un primo p) tiene aplicaciones fundamentales en criptografía y teoría de códigos.

3.3.1 Cuerpos finitos: estructura algebraica

Definición 3.19 (Cuerpo finito $\mathbb{Z}_p$)**.** *Sea p un número primo. El conjunto $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ con suma y multiplicación módulo p forma un cuerpo finito de orden p.*

Operaciones:

- *Suma:* $(a + b) \bmod p$
- *Multiplicación:* $(a \cdot b) \bmod p$
- *Elemento neutro aditivo:* 0
- *Elemento neutro multiplicativo:* 1
- *Inverso aditivo de a:* $(-a) \bmod p = p - a$
- *Inverso multiplicativo de $a \neq 0$:* a^{-1} *tal que* $a \cdot a^{-1} \equiv 1 \pmod{p}$

Propiedad clave: Todo elemento no nulo de $\mathbb{Z}_p$ tiene inverso multiplicativo. Esto es esencial para que la interpolación funcione (necesitamos dividir por $x_i - x_j$ con $x_i \neq x_j$).

Lema 3.20 (Cálculo de inversos en $\mathbb{Z}_p$)**.** *Para $a \in \mathbb{Z}_p$ con $a \neq 0$, el inverso multiplicativo puede calcularse mediante:*

1. ***Algoritmo extendido de Euclides**: Encuentra x, y tales que $ax + py = \gcd(a, p) = 1$. Entonces $a^{-1} \equiv x \pmod{p}$.*
2. ***Pequeño teorema de Fermat**: $a^{-1} \equiv a^{p-2} \pmod{p}$ (exponenciación rápida).*

Ejemplo 3.21 (Cuerpo $\mathbb{Z}_7$)**.** *En* $\mathbb{Z}_7$*:*

- $3 + 5 = 8 \equiv 1 \pmod 7$
- $4 \cdot 6 = 24 \equiv 3 \pmod 7$
- $3^{-1} = 5$ *porque* $3 \cdot 5 = 15 \equiv 1 \pmod 7$
- $2^{-1} = 4$ *porque* $2 \cdot 4 = 8 \equiv 1 \pmod 7$

3.3.2 Interpolación en $\mathbb{Z}_p$: teoría

La teoría desarrollada para $\mathbb{R}$ se traslada palabra por palabra a $\mathbb{Z}_p$:

Teorema 3.22 (Interpolación en $\mathbb{Z}_p$)**.** *Sean* $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p$ *nodos distintos (módulo* p*) y* $y_0, y_1, \ldots, y_n \in \mathbb{Z}_p$ *valores arbitrarios. Entonces existe un* ***único*** *polinomio* $p(x) \in \mathcal{P}_n(\mathbb{Z}_p)$ *tal que:*

$$p(x_i) \equiv y_i \pmod p, \quad i = 0, 1, \ldots, n.$$

Demostración. **Existencia:** Construimos el polinomio usando la fórmula de Lagrange. Los polinomios base son:

$$\ell_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j} \pmod p.$$

Como $x_i \neq x_j \pmod p$, cada diferencia $x_k - x_j$ tiene inverso en $\mathbb{Z}_p$, por lo que $\ell_k(x)$ está bien definido. Satisface $\ell_k(x_j) \equiv \delta_{kj} \pmod p$.

El interpolante es:

$$p(x) = \sum_{k=0}^{n} y_k \ell_k(x) \pmod p.$$

Unicidad: Si $p(x)$ y $q(x)$ interpolan los mismos datos, entonces $r(x) = p(x) - q(x)$ satisface $r(x_i) \equiv 0 \pmod p$ para todo i. El polinomio $r(x)$ de grado $\leq n$ tiene $n + 1$ raíces distintas en $\mathbb{Z}_p$, por lo que debe ser el polinomio cero (módulo p). □

Diferencias con $\mathbb{R}$:

- **Aritmética exacta**: No hay errores de redondeo en $\mathbb{Z}_p$ (todo es exacto módulo p).
- **Estructura finita**: Solo hay p elementos, lo que limita el número de nodos posibles.
- **Aplicaciones distintas**: Criptografía y códigos en lugar de aproximación de funciones.

3.3.3 Implementación computacional

La interpolación en $\mathbb{Z}_p$ requiere aritmética modular cuidadosa:

Algoritmo (Interpolación de Lagrange en $\mathbb{Z}_p$):
Entrada: Nodos $x_0, \ldots, x_n \in \mathbb{Z}_p$, valores $y_0, \ldots, y_n \in \mathbb{Z}_p$, primo p.
Salida: Coeficientes $[a_0, a_1, \ldots, a_n]$ de $p(x) = \sum_{k=0}^{n} a_k x^k \pmod p$.
Procedimiento:

1. Inicializar coefs $= [0, 0, \ldots, 0]$ (longitud $n+1$).
2. Para $k = 0, 1, \ldots, n$:
   (a) Calcular denominador: $d = \prod_{j \neq k}(x_k - x_j) \bmod p$.
   (b) Calcular inverso: $d^{-1} \bmod p$ (Euclides extendido).
   (c) Construir numerador (polinomio): $\text{num}(x) = \prod_{j \neq k}(x - x_j) \bmod p$.
   (d) Actualizar: coefs $=$ coefs $+ y_k \cdot d^{-1} \cdot \text{num} \pmod p$.
3. Retornar coefs.

Implementación en Python: Ver el archivo adjunto `interpolacion_algebraica.py`, función `interpolacion_modular(nodos, valores, p)`.

Ejemplo 3.23 (Interpolación en $\mathbb{Z}_7$)**.** *Interpolar los puntos* $(0,1)$, $(1,3)$, $(2,2)$ *en* $\mathbb{Z}_7$.
Solución: *Usando la fórmula de Lagrange:*

$$\begin{aligned}\ell_0(x) &= \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{(x-1)(x-2)}{2} \equiv 4(x-1)(x-2) \pmod 7 \\ &= 4(x^2 - 3x + 2) \equiv 4x^2 - 5x + 1 \pmod 7\end{aligned}$$

Similarmente calculamos $\ell_1(x)$ *y* $\ell_2(x)$*. El interpolante es:*

$$p(x) = 1 \cdot \ell_0(x) + 3 \cdot \ell_1(x) + 2 \cdot \ell_2(x) \equiv 5x^2 + 6x + 1 \pmod 7.$$

Verificación:

$$\begin{aligned}&p(0) = 1 \equiv 1 \pmod 7 \quad \checkmark \\ &p(1) = 5 + 6 + 1 = 12 \equiv 5 \pmod 7 \quad \textit{(¡error de cálculo! revisar)} \\ &p(2) = 20 + 12 + 1 = 33 \equiv 5 \pmod 7\end{aligned}$$

3.3.4 Aplicación: Esquema de secreto compartido de Shamir

Una aplicación elegante de la interpolación en $\mathbb{Z}_p$ es el **esquema de Shamir** (1979) para compartir un secreto de manera segura.

Definición 3.24 (Esquema de Shamir (n,k)-threshold)**.** ***Objetivo:*** *Dividir un secreto* $s \in \mathbb{Z}_p$ *entre* n *participantes de modo que:*

- *Cualquier grupo de* k *participantes puede reconstruir* s*.*
- *Cualquier grupo de* $k-1$ *o menos participantes no obtiene ninguna información sobre* s*.*

Protocolo:

1. ***Distribución de shares:***
   (a) Elegir aleatoriamente $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_p$*.*

(b) Construir el polinomio (en $\mathbb{Z}_p$):

$$p(x) = s + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \pmod{p}.$$

*(c) Distribuir los **shares**: participante i recibe $(i, p(i) \bmod p)$ para $i = 1, 2, \ldots, n$.*

2. ***Reconstrucción del secreto:***

   (a) Reunir k shares cualesquiera: $(x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k})$.

   (b) Interpolar (Lagrange en $\mathbb{Z}_p$) para obtener $p(x)$.

   (c) Recuperar $s = p(0) \bmod p$.

Teorema 3.25 (Seguridad perfecta de Shamir)**.** *El esquema de Shamir tiene las siguientes propiedades:*

1. ***Corrección:** Con k shares, el secreto se reconstruye correctamente.*
2. ***Seguridad perfecta:** Con $k-1$ shares o menos, todos los valores de $s \in \mathbb{Z}_p$ son igualmente probables.*

Demostración. **(1) Corrección:** El polinomio $p(x)$ de grado $k-1$ está uniquamente determinado por k puntos (teorema de interpolación). Como $s = p(0)$, se recupera exactamente.

(2) Seguridad perfecta: Supongamos que un adversario tiene $k-1$ shares. Fijemos cualquier valor tentativo $s' \in \mathbb{Z}_p$ para el secreto. ¿Existe un polinomio $p'(x)$ de grado $k-1$ tal que:

- $p'(0) = s'$ (el secreto tentativo),
- p' pasa por los $k-1$ shares conocidos?

La respuesta es **sí**: esto especifica k condiciones (valor en 0 más $k-1$ shares) para un polinomio de grado $k-1$, que tiene k coeficientes libres. El sistema tiene solución única.

Más aún, como los coeficientes $a_1, \ldots, a_{k-1}$ se eligieron uniformemente al azar de $\mathbb{Z}_p$, todos los valores de $s \in \mathbb{Z}_p$ son igualmente probables dado cualquier conjunto de $k-1$ shares. Por lo tanto, el adversario no aprende *nada* sobre s. □

Ejemplo numérico:

Ejemplo 3.26 (Shamir en $\mathbb{Z}_{97}$)**.** *Queremos compartir el secreto $s = 42$ entre $n = 5$ participantes, requiriendo $k = 3$ para reconstruir. Trabajamos en $\mathbb{Z}_{97}$.*

***Paso 1:** Elegimos aleatoriamente $a_1 = 15, a_2 = 67$ en $\mathbb{Z}_{97}$. El polinomio es:*

$$p(x) = 42 + 15x + 67x^2 \pmod{97}.$$

***Paso 2:** Generamos shares evaluando $p(i)$ para $i = 1, \ldots, 5$:*

$$\begin{aligned}
&\textit{Share 1:} \quad (1, p(1)) = (1, 42 + 15 + 67) = (1, 27) \\
&\textit{Share 2:} \quad (2, p(2)) = (2, 42 + 30 + 268) = (2, 49) \\
&\textit{Share 3:} \quad (3, p(3)) = (3, 42 + 45 + 603) = (3, 11) \\
&\textit{Share 4:} \quad (4, p(4)) = (4, 42 + 60 + 1072) = (4, 89) \\
&\textit{Share 5:} \quad (5, p(5)) = (5, 42 + 75 + 1675) = (5, 16)
\end{aligned}$$

(todas las operaciones módulo 97).

Paso 3: *Para reconstruir, usamos shares 1, 3, 5 (por ejemplo):*

$$\textit{Interpolar en } \mathbb{Z}_{97} : \{(1, 27), (3, 11), (5, 16)\}$$

Por interpolación de Lagrange obtenemos $p(x) = 42 + 15x + 67x^2$ *(mod 97). El secreto es* $s = p(0) = 42$.

Implementación: Ver `interpolacion_algebraica.py`:

- `shamir_generar_shares(secreto, n, k, p)`: Genera n shares.
- `shamir_reconstruir(shares, p)`: Reconstruye el secreto a partir de $\geq k$ shares.

3.3.5 Aplicación: Códigos de Reed-Solomon

Los **códigos de Reed-Solomon** son códigos de corrección de errores basados en evaluación de polinomios sobre $\mathbb{Z}_p$ (o más generalmente, sobre $\mathbb{F}_q$).

Definición 3.27 (Código Reed-Solomon RS(n, k))**.** ***Parámetros:***

- k*: longitud del mensaje (número de símbolos de información).*
- n*: longitud del código (*$n > k$*, símbolos redundantes).*
- $\mathbb{F}_q$*: Cuerpo finito con* $q \geq n$ *elementos.*
- $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$*: puntos de evaluación distintos.*

Codificación: *Un mensaje* $(m_0, m_1, \ldots, m_{k-1}) \in \mathbb{F}_q^k$ *se codifica como:*

1. *Interpretar el mensaje como coeficientes:* $m(x) = m_0 + m_1 x + \cdots + m_{k-1}x^{k-1}$.
2. *Evaluar en los* n *puntos:* $\mathbf{c} = (m(\alpha_1), m(\alpha_2), \ldots, m(\alpha_n))$.

El vector $\mathbf{c} \in \mathbb{F}_q^n$ *es la* ***palabra código****.*

Decodificación: *Dada una palabra posiblemente corrupta* $\mathbf{r} = \mathbf{c} + \mathbf{e}$ *(error* $\mathbf{e}$*):*

1. *Si hay* $\leq \lfloor (n-k)/2 \rfloor$ *errores, usar algoritmos de decodificación (Berlekamp-Welch, Euclides, etc.) para recuperar* $m(x)$.
2. *Extraer el mensaje original de los coeficientes de* $m(x)$.

Teorema 3.28 (Capacidad de corrección de Reed-Solomon)**.** *El código RS*(n, k) *puede:*

- ***Detectar*** *hasta* $n - k$ *errores.*
- ***Corregir*** *hasta* $\lfloor (n - k)/2 \rfloor$ *errores.*

Idea. La **distancia mínima** del código es $d = n-k+1$. Dos palabras código distintas provienen de polinomios distintos $m_1(x), m_2(x) \in \mathcal{P}_{k-1}(\mathbb{F}_q)$. Su diferencia $\Delta(x) = m_1(x) - m_2(x)$ es no nula y tiene grado $< k$, por lo que tiene a lo sumo $k - 1$ raíces en $\mathbb{F}_q$.

Por lo tanto, las evaluaciones de m_1 y m_2 difieren en al menos $n - (k - 1) = n - k + 1$ posiciones. Esta distancia mínima permite:

- Detectar hasta $d - 1 = n - k$ errores (si hay $\leq n - k$ errores, la palabra recibida no es válida).
- Corregir hasta $\lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2 \rfloor$ errores (por teoría de codificación).

□

Aplicaciones de Reed-Solomon:

- **CDs y DVDs**: Protección contra rayones y defectos.
- **Códigos QR**: Recuperación de datos parcialmente dañados.
- **Comunicaciones espaciales**: NASA/ESA usan RS para transmisiones desde sondas.
- **Almacenamiento distribuido**: RAID 6 usa RS para tolerar fallas de discos.

Chapter 4

Teoría de Aproximación

4.1 Series de Fourier de funciones suaves periódicas

En el capítulo anterior estudiamos la teoría general de espacios con producto interno y vimos cómo representar vectores en bases ortogonales mediante proyecciones. Ahora aplicaremos estos conceptos a espacios de funciones periódicas.

En diversas áreas de la ciencia y la ingeniería, es fundamental representar funciones complejas como sumas de componentes más simples. Las **series de Fourier** nos permiten descomponer funciones periódicas en una suma de exponenciales complejas de diferentes frecuencias. Esta descomposición es la aplicación directa de la fórmula de proyección (A.1) del apéndice, donde la base ortogonal son las exponenciales complejas $\{e^{ikx}\}_{k\in\mathbb{Z}}$.

Consideremos el espacio de funciones suaves y periódicas $C^n_{per}([0,2\pi])$ definido como:

$$C^n_{per}([0,2\pi]) = \left\{ f \in C^n([0,2\pi]) : f^{(p)}(0) = f^{(p)}(2\pi) \text{ para todo } 0 \leq p \leq n \right\}$$

Para funciones complejas $f, g : [0,2\pi] \to \mathbb{C}$, definimos el producto interno:

$$(f,g) = \int_0^{2\pi} f(x)\overline{g(x)}\, dx \tag{4.1}$$

donde $\overline{g(x)}$ denota el conjugado complejo de $g(x)$.

Observación 4.1 (Notación)**.** *En este capítulo utilizamos i para denotar la unidad imaginaria ($i^2 = -1$), siguiendo la convención matemática estándar. En capítulos posteriores sobre la DFT, usaremos la notación j (común en ingeniería eléctrica y procesamiento de señales). Ambas notaciones son equivalentes: $i \equiv j$.*

Definición 4.2 (Coeficientes de Fourier complejos)**.** *Sea $f : [0,2\pi] \to \mathbb{C}$ una función 2π-periódica integrable. Los **coeficientes de Fourier complejos** de f se definen como:*

$$\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}\, dx, \qquad k \in \mathbb{Z}. \tag{4.2}$$

Observación 4.3 (Conexión con la fórmula de proyección)**.** *Observemos que esta definición es precisamente la fórmula de proyección* (A.1) *aplicada a la función f sobre la base (no normalizada) de exponenciales complejas. En efecto, si verificamos que $\|e^{ikx}\|^2 = \int_0^{2\pi} e^{ikx}e^{-ikx}\,dx = 2\pi$, entonces:*

$$\hat{f}_k = \frac{(f, e^{ikx})}{\|e^{ikx}\|^2} = \frac{\int_0^{2\pi} f(x)e^{-ikx}\,dx}{2\pi}.$$

Esto muestra que los coeficientes de Fourier no son más que las coordenadas de f en la base ortogonal de exponenciales complejas, calculadas mediante proyección.

La **serie de Fourier compleja** de f se escribe entonces como:

$$f(x) \sim \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}. \tag{4.3}$$

Para cada $N \in \mathbb{N}$, definimos el conjunto de funciones base

$$B_N = \left\{ e^{ikx} : k \in \mathbb{Z},\ |k| \leq N \right\} \subset C^{\infty}_{per}([0, 2\pi]). \tag{4.4}$$

Teorema 4.4 (Ortogonalidad de la base exponencial compleja)**.** *El conjunto de funciones*

$$B_N = \left\{ e^{ikx} : k \in \mathbb{Z},\ |k| \leq N \right\}$$

es ortogonal respecto al producto interno

$$(f, g) = \int_0^{2\pi} f(x)\overline{g(x)}\,dx$$

en el espacio $C^{\infty}_{per}([0, 2\pi])$. Más precisamente, para $k, l \in \mathbb{Z}$:

$$\int_0^{2\pi} e^{ikx}e^{-ilx}\,dx = \int_0^{2\pi} e^{i(k-l)x}\,dx = \begin{cases} 2\pi & si\ k = l \\ 0 & si\ k \neq l \end{cases}$$

Demostración. Si $k = l$, entonces $e^{i(k-l)x} = e^0 = 1$, y por lo tanto:

$$\int_0^{2\pi} e^{i(k-l)x}\,dx = \int_0^{2\pi} 1\,dx = 2\pi.$$

Si $k \neq l$, sea $m = k - l \neq 0$. Entonces:

$$\int_0^{2\pi} e^{imx}\,dx = \left[\frac{e^{imx}}{im}\right]_0^{2\pi} = \frac{e^{im2\pi} - e^0}{im} = \frac{1-1}{im} = 0,$$

ya que $e^{im2\pi} = \cos(2\pi m) + i\sin(2\pi m) = 1$ para todo $m \in \mathbb{Z}$.

Por lo tanto, las funciones exponenciales complejas forman una base ortogonal. La norma de cada función base es $\|e^{ikx}\|^2 = 2\pi$. □

Observación 4.5 (Relación con la base trigonométrica real)**.** *La representación compleja es equivalente a la versión real con senos y cosenos. Utilizando las identidades de Euler:*

$$e^{ikx} = \cos(kx) + i\sin(kx), \qquad e^{-ikx} = \cos(kx) - i\sin(kx),$$

podemos escribir:

$$\cos(kx) = \frac{e^{ikx} + e^{-ikx}}{2}, \qquad \sin(kx) = \frac{e^{ikx} - e^{-ikx}}{2i}.$$

Si f es real y tiene coeficientes de Fourier complejos $\hat{f}_k$, entonces $\hat{f}_{-k} = \overline{\hat{f}_k}$. La serie de Fourier puede escribirse en forma real como:

$$f(x) = \hat{f}_0 + \sum_{k=1}^{\infty} \left[(\hat{f}_k + \hat{f}_{-k})\cos(kx) + i(\hat{f}_k - \hat{f}_{-k})\sin(kx) \right].$$

Definiendo $\hat{a}_k = 2Re(\hat{f}_k)$ y $\hat{b}_k = -2Im(\hat{f}_k)$ para $k \geq 1$, recuperamos:

$$f(x) = \frac{\hat{a}_0}{2} + \sum_{k=1}^{\infty} \left[\hat{a}_k \cos(kx) + \hat{b}_k \sin(kx) \right].$$

4.1.1 Decaimiento de los coeficientes de Fourier y regularidad

El siguiente teorema fundamental relaciona la regularidad de una función con el decaimiento de sus coeficientes de Fourier.

Teorema 4.6 (Decaimiento de los coeficientes de Fourier)**.** *Sea $f \in C^n_{per}([0, 2\pi])$, es decir, f es periódica y su derivada de orden n es continua. Entonces existe una constante $C > 0$ tal que, para todo $k \in \mathbb{Z}$ con $k \neq 0$:*

$$|\hat{f}_k| \leq \frac{C}{|k|^n},$$

donde $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}\,dx$ son los coeficientes de Fourier de f.

Demostración. Por integración por partes:

$$\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}\,dx.$$

Integramos por partes una vez. Sea $u = f(x)$ y $dv = e^{-ikx}dx$. Entonces $du = f'(x)dx$ y $v = \frac{e^{-ikx}}{-ik}$:

$$\hat{f}_k = \frac{1}{2\pi}\left[f(x)\frac{e^{-ikx}}{-ik} \right]_0^{2\pi} + \frac{1}{2\pi ik}\int_0^{2\pi} f'(x)e^{-ikx}\,dx.$$

Usando que f es 2π-periódica (es decir, $f(2\pi) = f(0)$), el término evaluado en los extremos se anula:

$$\left[f(x)e^{-ikx} \right]_0^{2\pi} = f(2\pi)e^{-ik2\pi} - f(0) = f(0)(e^{-ik2\pi} - 1) = 0.$$

Por lo tanto:

$$\hat{f}_k = \frac{1}{ik} \cdot \frac{1}{2\pi}\int_0^{2\pi} f'(x)e^{-ikx}\,dx = \frac{1}{ik}\widehat{f'}_k, \tag{4.5}$$

donde $\widehat{f'}_k$ denota el k-ésimo coeficiente de Fourier de f'.

Repitiendo este proceso n veces (y usando que todas las derivadas de f hasta orden n son periódicas), y utilizando la Ecuación (4.5):

$$\hat{f}_k = \frac{1}{(ik)^n}\widehat{f^{(n)}}_k.$$

Por lo tanto:

$$|\hat{f}_k| = \frac{1}{|k|^n}|\widehat{f^{(n)}}_k| \leq \frac{1}{|k|^n} \cdot \frac{1}{2\pi}\int_0^{2\pi} |f^{(n)}(x)|\,dx = \frac{C}{|k|^n},$$

donde $C = \frac{1}{2\pi}\int_0^{2\pi} |f^{(n)}(x)|\,dx$.

Por lo tanto, los coeficientes de Fourier de funciones C^n decrecen al menos como $1/|k|^n$. □

Corolario 4.7. *Si $f \in C^\infty_{per}([0, 2\pi])$ (es decir, f es infinitamente diferenciable), entonces sus coeficientes de Fourier decrecen más rápido que cualquier potencia de $1/|k|$:*

$$\textit{Para todo } n \in \mathbb{N},\ \textit{existe } C_n > 0 \textit{ tal que } |\hat{f}_k| \leq \frac{C_n}{|k|^n}.$$

Esto se conoce como ***decaimiento espectral*** *o* ***decaimiento rápido****.*

Corolario 4.8 (Convergencia Uniforme)**.** *Si $f \in C^n_{per}([0, 2\pi])$ con $n \geq 2$, entonces la serie de Fourier de f, dada por $\sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}$, converge absoluta y uniformemente a $f(x)$.*

Demostración. Del teorema anterior, si $f \in C^n_{per}$ con $n \geq 2$, sabemos que existe una constante C tal que $|\hat{f}_k| \leq \frac{C}{|k|^n}$ para $k \neq 0$. Consideremos la suma de los valores absolutos de los términos de la serie de Fourier:

$$\sum_{k=-\infty}^{\infty} |\hat{f}_k e^{ikx}| = \sum_{k=-\infty}^{\infty} |\hat{f}_k|.$$

Esta serie de números reales positivos puede acotarse como sigue:

$$\sum_{k=-\infty}^{\infty} |\hat{f}_k| = |\hat{f}_0| + \sum_{k\neq 0} |\hat{f}_k| \leq |\hat{f}_0| + \sum_{k\neq 0} \frac{C}{|k|^n}.$$

Como $n \geq 2$, la serie $\sum_{k\neq 0} \frac{1}{|k|^n}$ es una p-serie convergente. Por lo tanto, $\sum_{k=-\infty}^{\infty} |\hat{f}_k|$ converge.

Dado que $|\hat{f}_k e^{ikx}| = |\hat{f}_k|$, por el Test M de Weierstrass, la serie de Fourier $\sum \hat{f}_k e^{ikx}$ converge absoluta y uniformemente. Como la serie de Fourier de una función continua converge a la función en el sentido de L^2, y aquí la convergencia es uniforme, el límite puntual debe coincidir con $f(x)$. □

Corolario 4.9 (Suavidad del Límite)**.** *Si los coeficientes de Fourier de una función f decrecen suficientemente rápido, tal que para algún $m \geq 0$, la serie $\sum_{k=-\infty}^{\infty} |k|^m |\hat{f}_k|$ converge, entonces f es de clase $C^m_{per}([0, 2\pi])$.*

Demostración. Consideremos la serie de Fourier de f, $S(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}$. Si la derivamos formalmente término a término m veces, obtenemos la serie:

$$S^{(m)}(x) = \sum_{k=-\infty}^{\infty} (ik)^m \hat{f}_k e^{ikx}.$$

Para verificar que esta es la derivada de $S(x)$, necesitamos que la serie derivada converja uniformemente. La suma de los valores absolutos de sus términos es:

$$\sum_{k=-\infty}^{\infty} |(ik)^m \hat{f}_k e^{ikx}| = \sum_{k=-\infty}^{\infty} |k|^m |\hat{f}_k|.$$

Por hipótesis, esta serie converge. Por el Test M de Weierstrass, la serie $S^{(m)}(x)$ converge absoluta y uniformemente. Un teorema estándar de análisis real establece que si una secuencia de funciones derivables converge en un punto y la secuencia de sus derivadas converge uniformemente, entonces la función límite es derivable y su derivada es el límite de las derivadas. Aplicando este resultado de forma inductiva, concluimos que $S(x)$ es m veces diferenciable, y su m-ésima derivada es $S^{(m)}(x)$, que es una función continua por ser límite uniforme de funciones continuas. Por lo tanto, $f \in C^m_{per}([0, 2\pi])$. □

4.1.2 Cuadratura de funciones suaves y periódicas

El cálculo numérico de los coeficientes de Fourier requiere evaluar las integrales:

$$\hat{f}_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx.$$

En la práctica, estas integrales raramente pueden calcularse de forma exacta, por lo que debemos recurrir a métodos de cuadratura numérica. Para funciones periódicas suaves, la regla del trapecio resulta ser extraordinariamente eficiente.

Definición 4.10 (Aproximación discreta de coeficientes de Fourier)**.** *Dados M puntos equiespaciados $x_j = \frac{2\pi j}{M}$ para $j = 0, 1, \ldots, M-1$, definimos los* ***coeficientes de Fourier discretos*** *como:*

$$\hat{f}_k^{(M)} = \frac{1}{M} \sum_{j=0}^{M-1} f(x_j) e^{-ikx_j}.$$

Esta fórmula corresponde precisamente a la regla del trapecio compuesta aplicada al cálculo de $\hat{f}_k$:

$$\int_0^{2\pi} f(x) e^{-ikx} dx \approx \frac{2\pi}{M} \sum_{j=0}^{M-1} f(x_j) e^{-ikx_j}.$$

La efectividad de este método para funciones periódicas suaves se debe a una propiedad notable de la regla del trapecio.

Para funciones periódicas suaves, la regla del trapecio tiene propiedades espectrales extraordinarias.

Consideremos el siguiente lema

Lema 4.11. *La cuadratura trapezoidal es exacta para todas las funciones en $\langle B_N \rangle$; es decir, si f es combinación lineal finita de senos y cosenos de frecuencias hasta N, entonces*

$$\int_0^{2\pi} f(x)\,dx = h\sum_{j=0}^{N} f(x_j), \quad \text{con } x_j = jh,\ h = \frac{2\pi}{N+1}.$$

Demostración. Basta demostrarlo para cada función base 1, $\sin(kx)$ y $\cos(kx)$ con $k \le N$, ya que la cuadratura es lineal.

Para $f(x) = 1$:

$$\int_0^{2\pi} 1\,dx = 2\pi, \qquad h\sum_{j=0}^{N} 1 = h(N+1) = 2\pi.$$

Para $f(x) = \sin(kx)$:

$$\int_0^{2\pi} \sin(kx)\,dx = 0.$$

Por otro lado,

$$h\sum_{j=0}^{N} \sin(kx_j) = h\sum_{j=0}^{N} \sin\left(kjh\right).$$

Como $h = \frac{2\pi}{N+1}$, entonces $kjh = \frac{2\pi kj}{N+1}$. La suma

$$\sum_{j=0}^{N} \sin\left(\frac{2\pi kj}{N+1}\right)$$

es la suma de una progresión aritmética de ángulos igualmente espaciados en el círculo, que suma cero para $1 \le k \le N$ (esto se puede ver sumando la correspondiente serie geométrica en la forma compleja).

De manera similar, para $f(x) = \cos(kx)$ con $k \ge 1$:

$$\int_0^{2\pi} \cos(kx)\,dx = 0,$$

y

$$h\sum_{j=0}^{N} \cos\left(\frac{2\pi kj}{N+1}\right) = 0.$$

Por lo tanto, la cuadratura trapezoidal es exacta para cualquier combinación lineal de estas funciones. □

Teorema 4.12 (Convergencia espectral del trapecio periódico)**.** *Si $f \in C^\infty[0, 2\pi]$ es periódica y sus coeficientes de Fourier decaen como $|\hat{f}_k| = O(|k|^{-m})$ para todo $m > 0$, entonces el error de la regla del trapecio con $N+1$ puntos satisface:*

$$\left| \int_0^{2\pi} f(x)dx - h\sum_{j=0}^{N} f(x_j) \right| = O(N^{-m})$$

para todo $m > 0$. Es decir, la convergencia es ***más rápida que cualquier potencia*** *de $1/N$ (convergencia espectral).*

La razón profunda es que para funciones periódicas suaves, la regla del trapecio captura exactamente la proyección sobre $\langle B_N \rangle$, y el error está determinado solo por los coeficientes de Fourier de alta frecuencia, que decaen rápidamente. Este resultado contrasta dramáticamente con el caso no periódico, donde el trapecio solo tiene error $O(N^{-2})$.

Observación 4.13 (Contraste con funciones no periódicas)**.** *Para una función $f \in C^m([0, 2\pi])$ que no es periódica, la regla del trapecio solo tiene error $O(M^{-2})$, independientemente de la suavidad de f. La periodicidad elimina los errores en los extremos del intervalo y permite aprovechar plenamente la suavidad de la función.*

Teorema 4.14 (Convergencia espectral para funciones analíticas)**.** *Si f es analítica y periódica, extendible al strip $|Im(z)| < \alpha$ para algún $\alpha > 0$, entonces:*

$$\left| \int_0^{2\pi} f(x)dx - \frac{2\pi}{M} \sum_{j=0}^{M-1} f(x_j) \right| \leq Ce^{-\alpha M},$$

*es decir, el error decrece exponencialmente con M (**convergencia espectral**).*

Esta propiedad hace que la regla del trapecio sea el método preferido para integrar funciones periódicas suaves: con relativamente pocos puntos de evaluación, se puede alcanzar alta precisión.

Ejemplo 4.15 (Cálculo de coeficientes de Fourier de $f(x) = e^{\sin x}$)**.** *La función $f(x) = e^{\sin x}$ es periódica y analítica. Para calcular sus coeficientes de Fourier, usamos $M = 32$ puntos:*

$$\hat{f}_k^{(32)} = \frac{1}{32} \sum_{j=0}^{31} e^{\sin(2\pi j/32)} e^{-ik(2\pi j/32)}.$$

Los primeros coeficientes son:

$$\begin{aligned} \hat{f}_0^{(32)} &\approx 1.266065878, \\ \hat{f}_1^{(32)} &\approx 0.565159103, \\ \hat{f}_2^{(32)} &\approx 0.135747669, \\ \hat{f}_3^{(32)} &\approx 0.022698425. \end{aligned}$$

La convergencia es extremadamente rápida: los coeficientes decaen exponencialmente con $|k|$.

Corolario 4.16 (Aliasing y frecuencias múltiplos de M)**.** *Los coeficientes discretos $\hat{f}_k^{(M)}$ son periódicos en k con período M:*

$$\hat{f}_{k+M}^{(M)} = \hat{f}_k^{(M)}.$$

Esto es consecuencia directa de la periodicidad de e^{-ikx_j} en los puntos de muestreo.

Observación 4.17 (Relación con la DFT)**.** *La fórmula para $\hat{f}_k^{(M)}$ es precisamente la **Transformada Discreta de Fourier (DFT)** de los valores muestreados $\{f(x_j)\}_{j=0}^{M-1}$. Por lo tanto, el cálculo eficiente de coeficientes de Fourier discretos se reduce al cálculo de la DFT, que puede realizarse en $O(M \log M)$ operaciones mediante el algoritmo FFT (Fast Fourier Transform), en lugar de las $O(M^2)$ operaciones que requeriría la evaluación directa.*

4.1.3 El fenómeno de Gibbs

Las series de Fourier son extraordinariamente efectivas para funciones periódicas. Sin embargo, presentan limitaciones significativas cuando se aplican a funciones no periódicas:

1. **Fenómeno de Gibbs**: Si f no es periódica, su extensión periódica presenta discontinuidades en los extremos, causando oscilaciones cerca de estos puntos que no desaparecen al aumentar el número de términos.

2. **Convergencia lenta**: Para funciones suaves pero no periódicas, la convergencia de la serie de Fourier es típicamente algebraica en lugar de exponencial.

3. **Condiciones de frontera**: Es difícil imponer condiciones de frontera específicas usando bases trigonométricas.

Ejemplo 4.18 (Función escalón)**.** *Consideremos la función escalón*

$$f(x) = \begin{cases} -1 & si\ x \in [-1, 0) \\ 1 & si\ x \in [0, 1] \end{cases}$$

en el intervalo $[-1, 1]$*. Su serie de Fourier presenta el fenómeno de Gibbs cerca de los puntos* $x = 0$*,* $x = -1$ *y* $x = 1$*, con oscilaciones que persisten independientemente del número de términos utilizados.*

4.2 Series de Chebyshev

Las series de Fourier son una herramienta poderosa para aproximar funciones periódicas, ofreciendo convergencia espectral para funciones suaves. Sin embargo, la mayoría de las funciones que encontramos en aplicaciones (como soluciones de ecuaciones diferenciales con condiciones de frontera) no son periódicas. Si intentamos aproximar una función no periódica $f(x)$ en $[-1, 1]$ usando series de Fourier, nos encontramos con el **fenómeno de Gibbs** en los extremos, lo que destruye la convergencia uniforme.

¿Podemos "salvar" la convergencia espectral para funciones no periódicas? La respuesta es sí, mediante un simple cambio de variable.

4.2.1 De Fourier a Chebyshev: El cambio de variable coseno

Consideremos una función $f(x)$ definida en $[-1, 1]$. Hacemos el cambio de variable:

$$x = \cos(\theta), \quad \theta \in [0, \pi].$$

Definimos una nueva función $g(\theta)$ compuesta:

$$g(\theta) = f(\cos(\theta)).$$

Esta función $g(\theta)$ tiene propiedades notables:

1. Está definida en $[0, \pi]$.

2. Podemos extenderla a $[-\pi, \pi]$ como una función **par** ($g(-\theta) = f(\cos(-\theta)) = f(\cos(\theta)) = g(\theta)$).

3. Podemos extenderla a todo $\mathbb{R}$ como una función 2π-periódica.

Lo más importante es que esta extensión periódica es **automáticamente continua y suave** en los puntos de pegado ($\theta = 0, \pm\pi, \ldots$), incluso si $f(x)$ no tenía ninguna periodicidad. Esto elimina el fenómeno de Gibbs.

Al desarrollar $g(\theta)$ en serie de Fourier (que solo tendrá términos cosenos por ser par):

$$g(\theta) = \sum_{k=0}^{\infty} a_k \cos(k\theta).$$

Volviendo a la variable original x (donde $\theta = \arccos x$ y $\cos(k\theta) = T_k(x)$ son los polinomios de Chebyshev):

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x).$$

¡Hemos obtenido una expansión en polinomios de Chebyshev!

4.2.2 Convergencia Espectral

El resultado fundamental es que la expansión de Chebyshev hereda las excelentes propiedades de convergencia de la serie de Fourier de la función compuesta $g(\theta)$.

Teorema 4.19 (Decaimiento de coeficientes de Chebyshev)**.** *Si $f(x)$ es una función $C^m([-1, 1])$, entonces sus coeficientes de Chebyshev a_k decaen algebraicamente:*

$$|a_k| = O(k^{-m}).$$

Si $f(x)$ es analítica en una elipse alrededor de $[-1, 1]$, los coeficientes decaen exponencialmente:

$$|a_k| = O(\rho^{-k}) \quad (\rho > 1).$$

Esto significa que para funciones suaves no periódicas, la serie de Chebyshev converge tan rápido como la serie de Fourier para funciones periódicas.

4.3 Polinomios Ortogonales

Los polinomios ortogonales generalizan las series de Fourier para la aproximación de funciones no periódicas en intervalos acotados (o infinitos con peso). Son fundamentales para la teoría de aproximación, la integración numérica (cuadratura de Gauss) y los métodos espectrales.

4.3.1 Definiciones y propiedades básicas

Dada una función de peso $w(x) > 0$ en (a, b), definimos el producto interno:

$$(f, g)_w = \int_a^b f(x)g(x)w(x)\,dx.$$

Una familia de polinomios $\{P_n\}_{n=0}^{\infty}$ es **ortogonal** si $\deg(P_n) = n$ y $(P_n, P_m)_w = 0$ si $n \neq m$.

Las familias clásicas más importantes son:

Familia	**Intervalo**	**Peso** $w(x)$	**Autovalor SL** λ_n
Legendre P_n	$[-1, 1]$	1	$n(n+1)$
Chebyshev T_n	$[-1, 1]$	$(1-x^2)^{-1/2}$	n^2
Laguerre L_n	$[0, \infty)$	e^{-x}	n
Hermite H_n	$(-\infty, \infty)$	e^{-x^2}	$2n$

Propiedades clave:

1. **Recurrencia:** $P_{n+1}(x) = (A_n x + B_n)P_n(x) - C_n P_{n-1}(x)$. Permite evaluar $P_n(x)$ de forma estable y eficiente.
2. **Raíces:** $P_n(x)$ tiene exactamente n raíces reales, simples y contenidas en (a, b). Estas raíces son los nodos óptimos para interpolación y cuadratura.

4.3.2 Cuadratura Gaussiana

La cuadratura de Gauss es óptima en el sentido que con n nodos, alcanza exactitud para polinomios de grado $\leq 2n-1$, el máximo posible.

Teorema 4.20 (Exactitud y Positividad)**.** *Sean $\{P_k\}$ los polinomios ortogonales respecto al peso w en $[a, b]$. Si los nodos $\{x_k\}_{k=1}^n$ son las raíces de P_n, entonces existe una única elección de pesos $\{w_k\}$ tal que:*

$$\int_a^b p(x)w(x)dx = \sum_{k=1}^{n} w_k p(x_k) \quad \textit{para todo } p \in \Pi_{2n-1}.$$

Además, los pesos son siempre positivos:

$$w_k = \int_a^b \ell_k(x)w(x)dx = \int_a^b \ell_k^2(x)w(x)dx > 0.$$

La positividad de los pesos es crucial para la estabilidad numérica, ya que evita cancelaciones catastróficas al sumar términos.

4.3.3 Tasa de convergencia para funciones suaves

El resultado más profundo de esta teoría conecta la suavidad de una función con la velocidad de convergencia de su expansión en polinomios ortogonales. Esta conexión surge porque los polinomios clásicos son autofunciones de operadores diferenciales de **Sturm-Liouville**:

$$L[P_n] = -\frac{1}{w}\frac{d}{dx}\left(p(x)\frac{dP_n}{dx}\right) = \lambda_n P_n.$$

Teorema 4.21 (Decaimiento espectral de coeficientes)**.** *Sea $f \in C^m([a,b])$ una función m veces diferenciable (compatible con las condiciones de frontera). Los coeficientes de su expansión $c_n = \frac{(f,P_n)_w}{\|P_n\|_w^2}$ satisfacen:*

$$|c_n| = O(\lambda_n^{-m/2}).$$

Para Legendre y Chebyshev ($\lambda_n \sim n^2$), esto implica $|c_n| = O(n^{-m})$.

Idea de la demostración. Usamos la propiedad de autofunción $P_n = \frac{1}{\lambda_n} L[P_n]$ e integración por partes repetida.

$$c_n \sim \int f P_n w = \frac{1}{\lambda_n} \int f L[P_n] w = \frac{1}{\lambda_n} \int L[f] P_n w.$$

Como el operador L es autoadjunto (gracias al peso w), podemos pasar las derivadas de P_n a f. Repitiendo el proceso k veces (si $2k \leq m$), obtenemos un factor $1/\lambda_n^k$ y derivadas de orden $2k$ de f. Esto muestra que la suavidad de f se traduce directamente en el decaimiento de los coeficientes. □

Este resultado tiene dos consecuencias inmediatas fundamentales para el cálculo numérico:

Teorema 4.22 (Convergencia de la Interpolación en nodos ortogonales)**.** *Si interpolamos $f \in C^m$ en las raíces de P_n (nodos de Gauss/Chebyshev), el error en norma L_w^2 se comporta asintóticamente como el error de truncamiento de la serie ortogonal:*

$$\|f - p_n\|_{L_w^2} \approx \left\| \sum_{k=n+1}^{\infty} c_k P_k \right\|_{L_w^2} = O(n^{-m}).$$

*Esto significa que **la interpolación en nodos ortogonales hereda la tasa de convergencia espectral**: si f es muy suave (C^∞), el error decae más rápido que cualquier potencia de n (convergencia exponencial).*

Teorema 4.23 (Convergencia de la Cuadratura Gaussiana)**.** *La cuadratura de Gauss $Q_n(f) = \sum w_k f(x_k)$ es exacta para polinomios de grado $2n-1$. El error de integración está acotado por el error de la mejor aproximación polinomial de grado $2n-1$:*

$$|I(f) - Q_n(f)| \leq C \|f - p_{2n-1}^*\|_\infty.$$

Gracias al teorema de decaimiento, para $f \in C^m$, este error es $O(n^{-m})$.

Conclusión: La cuadratura de Gauss converge con el mismo orden que la suavidad de la función. Para funciones analíticas, la convergencia es geométrica (duplicar n duplica el número de dígitos correctos).

4.3.4 Cálculo de nodos y pesos: Método de Golub-Welsch

Aunque las raíces de los polinomios ortogonales clásicos están tabuladas, para n grande o pesos no estándar es necesario calcularlas numéricamente. El método de Golub-Welsch (1969) es el algoritmo estándar para esto, transformando el problema de búsqueda de raíces en un problema de autovalores.

Recordemos la relación de recurrencia de tres términos para los polinomios ortogonales mónicos π_k:

$$\pi_{k+1}(x) = (x - \alpha_k)\pi_k(x) - \beta_k \pi_{k-1}(x),$$

donde $\beta_k > 0$. Esta relación puede escribirse en forma matricial como:

$$x\pi(x) = J_n \pi(x) + \pi_n(x)\mathbf{e}_n,$$

donde $\pi(x) = [\pi_0(x), \ldots, \pi_{n-1}(x)]^T$ y J_n es la **matriz de Jacobi** tridiagonal simétrica:

$$J_n = \begin{pmatrix} \alpha_0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_{n-1}} \\ & & & \sqrt{\beta_{n-1}} & \alpha_{n-1} \end{pmatrix}.$$

Teorema 4.24 (Golub-Welsch)**.** *Los nodos de la cuadratura de Gauss de n puntos son los* ***autovalores*** *de la matriz de Jacobi J_n. Los pesos están dados por:*

$$w_k = \mu_0 (v_{k,1})^2,$$

donde $\mu_0 = \int_a^b w(x)dx$ es el momento cero, y v_k es el autovector normalizado correspondiente al autovalor x_k, siendo $v_{k,1}$ su primera componente.

Este método es extremadamente robusto y eficiente ($O(n^2)$), permitiendo calcular reglas de cuadratura para cualquier peso siempre que se conozcan los coeficientes de recurrencia α_k, β_k.

4.4 Interpolación de funciones reales

Hasta este punto hemos tratado la interpolación principalmente como un problema algebraico: encontrar un polinomio que satisfaga ciertas restricciones puntuales. Esta perspectiva es válida sobre cualquier cuerpo. Sin embargo, cuando trabajamos sobre el cuerpo de los números reales $\mathbb{R}$, disponemos de una estructura mucho más rica que incluye nociones de distancia, límite, continuidad y diferenciabilidad.

Si bien la **Teoría de Aproximación** propiamente dicha (que busca la "mejor" función para representar a otra según algún criterio global) será el tema central del próximo capítulo, aquí daremos unos primeros pasos en esa dirección. Nos interesa analizar qué tan bien se comporta el polinomio interpolante como sustituto de la función original en todo el intervalo, no solo en los nodos.

Estos resultados preliminares sobre el error de interpolación son esenciales para justificar el uso de polinomios en tareas como la integración numérica (cuadratura), que veremos a continuación.

4.4.1 Error de interpolación

El análisis del error de interpolación es fundamental para entender las limitaciones y potencial de los métodos estudiados. El resultado central es:

Teorema 4.25 (Error de interpolación)**.** *Sea $f \in C^{n+1}([a,b])$ y sea $p(x)$ el polinomio que interpola f en los nodos $x_0, \ldots, x_n \in [a,b]$. Entonces, para cada $x \in [a,b]$, existe $\xi \in (a,b)$ tal que:*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^{n} (x - x_k). \tag{4.6}$$

Idea de la demostración. Para un x fijo, definimos $g(t) = f(t) - p(t) - C\prod_{k=0}^{n}(t - x_k)$, donde C se elige de modo que $g(x) = 0$. La función g tiene al menos $n+2$ raíces: $x_0, \ldots, x_n, x$. Aplicando el teorema de Rolle $(n+1)$ veces, $g^{(n+1)}$ tiene al menos una raíz ξ. Como $p^{(n+1)} \equiv 0$, tenemos $g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - C(n+1)! = 0$, de donde se obtiene el resultado. □

4.4.2 Cuadratura interpolatoria

Las **cuadraturas interpolatorias** aproximan integrales usando interpolación polinomial. Dado peso $w(x)$ y nodos $x_1, \ldots, x_n$, interpolando f por $p \in \mathcal{P}_{n-1}$:

$$\int_a^b f(x)w(x)dx \approx \int_a^b p(x)w(x)dx = \sum_{k=1}^{n} w_k f(x_k),$$

donde los **pesos** son functionales lineales:

$$w_k = \int_a^b \ell_k(x)w(x)dx.$$

Exactitud: La fórmula es exacta para $f \in \mathcal{P}_{n-1}$ por construcción.
Reglas clásicas ($w = 1$ en $[a,b]$, longitud $h = b - a$):

- **Trapecio**: $\int_a^b f \approx \frac{h}{2}[f(a) + f(b)]$. Error: $-\frac{h^3}{12}f''(\xi)$.
- **Simpson**: $\int_a^b f \approx \frac{h}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$. Error: $-\frac{h^5}{90}f^{(4)}(\xi)$.
- **Gauss-Legendre**: Nodos óptimos (raíces de polinomios de Legendre) hacen exacta la regla para $\mathcal{P}_{2n-1}$ usando solo n nodos.
- **Regla del punto medio** ($n = 1$, $x_1 = (a+b)/2$):

  $$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right).$$

  Exacta para Π_1 (polinomios de grado ≤ 1).
- **Regla del trapecio** ($n = 2$, $x_1 = a$, $x_2 = b$):

  $$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a) + f(b)].$$

  Exacta para Π_1.

- **Regla de Simpson** ($n = 3$, nodos equiespaciados):

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Exacta para Π_3 (¡sorprendentemente, grado 3 con solo 3 nodos!).

Reglas compuestas. Para intervalos grandes $[a, b]$, subdividimos en m subintervalos y aplicamos una regla simple en cada uno. Por ejemplo, el **trapecio compuesto** con $x_j = a + jh$, $h = (b-a)/m$:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(x_0) + 2\sum_{j=1}^{m-1} f(x_j) + f(x_m)\right].$$

El error es $O(h^2)$ si $f \in C^2$. Este método es robusto pero converge lentamente.

Lema 4.26 (Cambio de intervalo)**.** *Si $Q_0(f) = \sum_{j=0}^n A_j f(t_j)$ es una fórmula de cuadratura para aproximar la integral $\int_{-1}^1 f(x)\,dx$, entonces para cualquier intervalo $[a, b]$ la fórmula transformada es:*

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\sum_{j=0}^n A_j f\left(\frac{b-a}{2}t_j + \frac{a+b}{2}\right).$$

Demostración. Hacemos el cambio de variable $x = \frac{b-a}{2}t + \frac{a+b}{2}$, que transforma $[-1, 1] \to [a, b]$. Entonces $dx = \frac{b-a}{2}dt$ y:

$$\int_a^b f(x)\,dx = \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\frac{b-a}{2}\,dt \approx \frac{b-a}{2}\sum_{j=0}^n A_j f\left(\frac{b-a}{2}t_j + \frac{a+b}{2}\right).$$

□

Observación 4.27. *Este lema es fundamental pues permite trabajar siempre en el intervalo estándar $[-1, 1]$ y luego transformar a cualquier intervalo $[a, b]$. Además, el grado de exactitud se preserva bajo este cambio de variable.*

Teorema 4.28 (Cota general del error)**.** *Dada una regla de cuadratura $Q(f)$ en el intervalo $[a, b]$ tal que:*

(i) $Q(f)$ es lineal,

(ii) $Q(f)$ tiene grado de exactitud k,

(iii) $|Q(f)| \leq M(b-a)||f||_\infty$ para alguna constante M,

entonces, si $f \in C^{k+1}[a, b]$, se tiene:

$$|I(f) - Q(f)| \leq \frac{(1+M)(b-a)^{k+2}}{(k+1)!}||f^{(k+1)}||_\infty.$$

Demostración. Como $f \in C^{k+1}[a,b]$, existe un polinomio $p_k \in \Pi_k$ tal que

$$||f - p_k||_\infty \leq \frac{(b-a)^{k+1}}{(k+1)!}||f^{(k+1)}||_\infty.$$

En efecto, podemos tomar el polinomio de Taylor de f en cualquier punto del intervalo. Como la regla tiene exactitud k, tenemos $I(p_k) = Q(p_k)$. Luego, usando la linealidad de Q:

$$I(f) - Q(f) = I(f - p_k) - Q(f - p_k).$$

Por lo tanto, usando la hipótesis (iii) y que $|I(f-p_k)| \leq (b-a)||f-p_k||_\infty$, obtenemos:

$$|I(f)-Q(f)| \leq |I(f-p_k)|+|Q(f-p_k)| \leq (1+M)(b-a)||f-p_k||_\infty \leq \frac{(1+M)(b-a)^{k+2}}{(k+1)!}||f^{(k+1)}||_\infty.$$

□

Observación 4.29. *Si la regla está dada como $Q(f) = \sum_{j=0}^n A_j f(x_j)$ con $A_j > 0$ para todo j y tiene grado de exactitud al menos 0, entonces la hipótesis (iii) se cumple con $M = 1$. En efecto, como la regla es exacta para constantes, $\sum_{j=0}^n A_j = Q(1) = I(1) = b - a$, y por tanto:*

$$|Q(f)| = \left|\sum_{j=0}^n A_j f(x_j)\right| \leq \sum_{j=0}^n A_j |f(x_j)| \leq (b-a)||f||_\infty.$$

Para obtener fórmulas más precisas del error para reglas específicas, necesitamos una generalización del teorema del valor medio para integrales:

Teorema 4.30 (Teorema del Valor Medio Integral Generalizado)**.** *Si $g : [a,b] \to \mathbb{R}$, $\xi : [a,b] \to [c,d]$, y $h : [c,d] \to \mathbb{R}$ son tales que g es continua y no cambia de signo, h es continua y $h(\xi(x))g(x)$ es integrable, entonces existe $\eta \in (c,d)$ tal que:*

$$\int_a^b h(\xi(x))g(x)\,dx = h(\eta)\int_a^b g(x)\,dx.$$

Demostración. Supongamos $g(x) \geq 0$ (el caso $g(x) \leq 0$ es análogo). Sean m y M el mínimo y máximo de h, respectivamente. Entonces $m \leq h(\xi(x)) \leq M$ y por tanto $mg(x) \leq h(\xi(x))g(x) \leq Mg(x)$ para todo $x \in [a,b]$. Integrando:

$$m\int_a^b g(x)\,dx \leq \int_a^b h(\xi(x))g(x)\,dx \leq M\int_a^b g(x)\,dx.$$

Como $\int_a^b g(x)\,dx > 0$ (suponiendo g no idénticamente cero), resulta:

$$m \leq \frac{\int_a^b h(\xi(x))g(x)\,dx}{\int_a^b g(x)\,dx} \leq M.$$

Por el teorema del valor intermedio, como h es continua, existe $\eta \in (c,d)$ tal que:

$$h(\eta) = \frac{\int_a^b h(\xi(x))g(x)\,dx}{\int_a^b g(x)\,dx},$$

lo que concluye la demostración. □

Ahora podemos demostrar las fórmulas precisas del error para las reglas clásicas:

Teorema 4.31 (Error para la Regla del Trapecio)**.** *Si $f \in C^2[a,b]$, el error que se produce al aproximar $\int_a^b f(x)\,dx$ usando la regla del trapecio está dado por:*

$$I(f) - T(f) = -\frac{(b-a)^3}{12} f''(\eta), \quad \text{para algún } \eta \in (a,b).$$

Demostración. Recordemos que para la regla del trapecio los nodos son $\{a,b\}$, y por tanto el error de interpolación satisface (por el teorema del error de interpolación):

$$f(x) - p_1(x) = \frac{f''(\xi)}{2}(x-a)(x-b),$$

donde p_1 es el interpolante lineal y $\xi = \xi(x) \in (a,b)$. Integrando:

$$I(f) - T(f) = \int_a^b [f(x) - p_1(x)]\,dx = \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)\,dx.$$

Como $(x-a)(x-b) \leq 0$ en $[a,b]$, no cambia de signo. Aplicando el Teorema del Valor Medio Integral Generalizado:

$$I(f) - T(f) = \frac{f''(\eta)}{2} \int_a^b (x-a)(x-b)\,dx.$$

Para calcular la integral, integramos por partes derivando $(x-b)$ e integrando $(x-a)$:

$$\int_a^b (x-a)(x-b)\,dx = \left[(x-b)\frac{(x-a)^2}{2}\right]_a^b - \int_a^b \frac{(x-a)^2}{2}\,dx = -\frac{1}{2}\int_a^b (x-a)^2\,dx = -\frac{(b-a)^3}{6}.$$

Por lo tanto:

$$I(f) - T(f) = \frac{f''(\eta)}{2} \cdot \left(-\frac{(b-a)^3}{6}\right) = -\frac{(b-a)^3}{12} f''(\eta).$$

□

Teorema 4.32 (Error para la Regla de Simpson)**.** *Si $f \in C^4[a,b]$, el error que se produce al aproximar $\int_a^b f(x)\,dx$ usando la regla de Simpson está dado por:*

$$I(f) - S(f) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\eta), \quad \text{para algún } \eta \in (a,b).$$

Demostración. La demostración es más técnica. Consideremos el intervalo $[-h,h]$ donde $h = (b-a)/2$. Definamos:

$$F(x) = \int_0^x f(t)\,dt$$

y, para $t \in [-h,h]$:

$$e(t) = F(t) - F(-t) - t\left[\frac{1}{3}f(-t) + \frac{4}{3}f(0) + \frac{1}{3}f(t)\right],$$

de modo que $I(f) - S(f) = e(h)$. Derivando sucesivamente:

$$\begin{aligned}
e'(t) &= f(t) + f(-t) - \frac{1}{3}f(-t) - \frac{4}{3}f(0) - \frac{1}{3}f(t) - \frac{t}{3}f'(-t) + \frac{t}{3}f'(t) \\
&= \frac{2}{3}f(t) + \frac{2}{3}f(-t) - \frac{4}{3}f(0) + \frac{t}{3}[f'(t) - f'(-t)], \\
e''(t) &= \frac{1}{3}f'(t) - \frac{1}{3}f'(-t) + \frac{t}{3}[f''(t) + f''(-t)], \\
e'''(t) &= \frac{t}{3}[f'''(t) - f'''(-t)].
\end{aligned}$$

Observemos que $e(0) = e'(0) = e''(0) = 0$. Por el teorema del valor medio, existe $\xi_1(t)$ tal que:

$$e'''(t) = \frac{t}{3}[f'''(t) - f'''(-t)] = -\frac{2t^2}{3}f^{(4)}(\xi_1(t)).$$

Integrando desde 0 a h y aplicando el Teorema del Valor Medio Integral Generalizado:

$$e''(h) = \int_0^h e'''(t)\,dt = -\frac{2}{3}\int_0^h t^2 f^{(4)}(\xi_1(t))\,dt = -\frac{2}{3}f^{(4)}(\xi_2)\int_0^h t^2\,dt = -\frac{2h^3}{9}f^{(4)}(\xi_2).$$

Integrando de nuevo:

$$e'(h) = \int_0^h e''(t)\,dt = -\frac{2f^{(4)}(\xi_3)}{9}\int_0^h t^3\,dt = -\frac{h^4}{18}f^{(4)}(\xi_3).$$

Finalmente:

$$e(h) = \int_0^h e'(t)\,dt = -\frac{f^{(4)}(\eta)}{18}\int_0^h t^4\,dt = -\frac{h^5}{90}f^{(4)}(\eta) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\eta).$$

□

4.4.3 Interpolación en nodos de Gauss y proyección ortogonal

Una pregunta fundamental es: ¿Es el polinomio interpolante $p_n(x)$ en los nodos de Gauss igual a la proyección ortogonal $S_n f(x)$ (la mejor aproximación en L^2_w)?

En general, la respuesta es **no**. La proyección ortogonal $S_n f = \sum_{k=0}^{n} c_k P_k$ requiere calcular integrales continuas $c_k = \frac{(f,P_k)_w}{\|P_k\|_w^2}$. El interpolante, en cambio, fuerza la coincidencia puntual.

Sin embargo, existe una conexión profunda:

> **El polinomio interpolante es exactamente la proyección ortogonal si los coeficientes de Fourier se calculan aproximando la integral mediante Cuadratura Gaussiana.**

Si aproximamos c_k usando la regla de cuadratura con los mismos nodos de interpolación:

$$c_k \approx \tilde{c}_k = \frac{1}{\|P_k\|_w^2}\sum_{j=1}^{n+1} w_j f(x_j) P_k(x_j),$$

entonces el polinomio $\sum_{k=0}^{n} \tilde{c}_k P_k(x)$ es precisamente el interpolante de f en los nodos $\{x_j\}$.

La diferencia entre los coeficientes verdaderos c_k y los discretos $\tilde{c}_k$ se debe al error de cuadratura, fenómeno conocido como **aliasing**: las frecuencias altas de f (componentes c_m con $m > n$) se "pliegan" y contaminan los coeficientes de baja frecuencia. Para funciones suaves, donde c_m decae rápidamente, este error es despreciable, haciendo de la interpolación una excelente aproximación a la proyección L^2.

4.4.4 Nodos de Chebyshev y FFT

Para calcular numéricamente los coeficientes a_k, usamos interpolación. Los puntos equiespaciados en la variable θ corresponden a los **nodos de Chebyshev** en x:

$$\theta_j = \frac{(2j+1)\pi}{2(n+1)} \implies x_j = \cos\left(\frac{(2j+1)\pi}{2(n+1)}\right).$$

Estos nodos se agrupan cerca de los extremos ± 1 con densidad $\sim 1/\sqrt{1-x^2}$.

Observación 4.33 (Conexión con FFT)**.** *La interpolación en nodos de Chebyshev es equivalente a una DFT (Transformada Discreta de Fourier) de los datos en la malla de θ. Esto permite usar la **FFT (Fast Fourier Transform)** para calcular los coeficientes de interpolación en $O(n \log n)$ operaciones, haciendo que los métodos espectrales de Chebyshev sean extremadamente eficientes.*

4.5 El operador de interpolación

4.5.1 Constante de Lebesgue

Como vimos, el error de interpolación depende de dos factores fundamentales:

1. **Suavidad de la función**: $\frac{f^{(n+1)}(\xi)}{(n+1)!}$
2. **Distribución de los nodos**: $\prod_{k=0}^{n}(x - x_k)$

Definición 4.34 (Función de Lebesgue)**.** *Para un conjunto de nodos $\{x_k\}$, la **función de Lebesgue** se define como:*

$$\Lambda(x) = \sum_{k=0}^{n} |\ell_k(x)|.$$

*La **constante de Lebesgue** es $\Lambda_n = \max_{x\in[a,b]} \Lambda(x)$.*

Definición 4.35 (Mejor aproximación en L^∞)**.** *Dada una función continua f en $[a,b]$, la **mejor aproximación uniforme** de grado n es el polinomio $p^* \in \Pi_n$ que minimiza:*

$$\|f - p^*\|_\infty = \max_{x\in[a,b]} |f(x) - p^*(x)|.$$

Teorema 4.36 (Cota del error en términos de la constante de Lebesgue)**.** *Si p^* es la mejor aproximación uniforme de f en el espacio de polinomios de grado $\leq n$, entonces:*

$$\|f - p\|_\infty \leq (1 + \Lambda_n)\|f - p^*\|_\infty.$$

Demostración. La demostración se basa en descomponer el error de interpolación usando la mejor aproximación como término intermedio.

Para cualquier $x \in [a, b]$, podemos escribir:

$$f(x) - p(x) = [f(x) - p^*(x)] + [p^*(x) - p(x)],$$

donde p es el polinomio interpolante y p^* es la mejor aproximación uniforme. El primer término ya está acotado por $\|f - p^*\|_\infty$ por definición de la norma infinito. Debemos analizar el segundo término.

Como p^* es un polinomio de grado $\leq n$, podemos escribirlo en la base de Lagrange asociada a los nodos $\{x_k\}_{k=0}^n$:

$$p^*(x) = \sum_{k=0}^{n} p^*(x_k)\ell_k(x),$$

donde $\ell_k(x)$ son los polinomios base de Lagrange. Por otro lado, el polinomio interpolante p satisface $p(x_k) = f(x_k)$ para todo k, por lo que:

$$p(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x).$$

Restando estas expresiones obtenemos:

$$p^*(x) - p(x) = \sum_{k=0}^{n} [p^*(x_k) - f(x_k)]\ell_k(x).$$

Tomando valores absolutos y usando la desigualdad triangular:

$$\begin{aligned}
|p^*(x) - p(x)| &= \left| \sum_{k=0}^{n} [p^*(x_k) - f(x_k)]\ell_k(x) \right| \\
&\leq \sum_{k=0}^{n} |p^*(x_k) - f(x_k)| \cdot |\ell_k(x)| \\
&\leq \max_{k=0,\ldots,n} |p^*(x_k) - f(x_k)| \cdot \sum_{k=0}^{n} |\ell_k(x)| \\
&\leq \|f - p^*\|_\infty \cdot \Lambda(x),
\end{aligned}$$

donde en la última desigualdad usamos que el máximo puntual de $|p^*(x_k) - f(x_k)|$ está acotado por el supremo global $\|f - p^*\|_\infty$.

Juntando ambos términos de la descomposición inicial, obtenemos:

$$\begin{aligned}
|f(x) - p(x)| &\leq |f(x) - p^*(x)| + |p^*(x) - p(x)| \\
&\leq \|f - p^*\|_\infty + \Lambda(x)\|f - p^*\|_\infty \\
&= (1 + \Lambda(x))\|f - p^*\|_\infty.
\end{aligned}$$

Finalmente, tomando supremo sobre $x \in [a, b]$:

$$\|f - p\|_\infty = \sup_{x \in [a,b]} |f(x) - p(x)| \leq (1 + \Lambda_n)\|f - p^*\|_\infty,$$

donde $\Lambda_n = \max_{x \in [a,b]} \Lambda(x)$ es la constante de Lebesgue. □

Teorema 4.37 (La cota es ajustada)**.** *La cota* $(1+\Lambda_n)$ *del teorema anterior no puede mejorarse: para cualquier conjunto de nodos* $\{x_k\}_{k=0}^n$*, existen funciones continuas* f *para las cuales el error de interpolación alcanza (esencialmente) este valor. Más precisamente, para cualquier* $\varepsilon > 0$*, existe una función continua* f *tal que:*

$$\|f - p\|_\infty \geq (\Lambda_n - \varepsilon)\|f - p^*\|_\infty,$$

donde p *es el interpolante de* f *y* p^* *es la mejor aproximación uniforme de* f *de grado* n*.*

Idea de la demostración. Una construcción explícita de la función f que alcanza la cota puede realizarse considerando:

$$f(x) = \sum_{k=0}^{n} \operatorname{sgn}(\ell_k(\bar{x})) \cdot \ell_k(x) + h(x),$$

donde $\bar{x}$ es un punto donde $\Lambda(\bar{x}) = \Lambda_n$ y h es una función continua no polinomial apropiadamente elegida. La demostración es técnica y se basa en analizar el comportamiento del interpolante y la mejor aproximación para esta función específica. □

Observación 4.38 (Interpretación y consecuencias)**.** *Los resultados anteriores tienen varias implicaciones importantes para la interpolación polinomial:*

1. ***Cuasi-optimalidad****: La interpolación es cuasi-óptima: su error está controlado por* $(1 + \Lambda_n)$ *veces el error de la mejor aproximación posible. Esto significa que si existe un polinomio que aproxima bien a* f*, entonces el interpolante también lo hará razonablemente bien.*

2. ***La cota es ajustada****: El Teorema 4.37 muestra que no podemos mejorar el factor* $(1+\Lambda_n)$ *en general: existen funciones para las cuales el error de interpolación es esencialmente* $(1 + \Lambda_n)$ *veces el error de la mejor aproximación.*

3. ***Importancia de*** Λ_n*: La constante de Lebesgue determina completamente la calidad de la interpolación en el peor caso:*

   - *Si* Λ_n *es pequeña (como para nodos de Chebyshev, donde* $\Lambda_n \approx \frac{2}{\pi}\log(n+1)$*), la interpolación es casi óptima.*
   - *Si* Λ_n *es grande (como para nodos equiespaciados, donde* $\Lambda_n \approx \frac{2^n}{en\log n}$*), la interpolación puede ser mucho peor que la mejor aproximación.*

4. ***Factor de amplificación inevitable****: La constante de Lebesgue representa un factor de amplificación inevitable del error de aproximación. Es una propiedad intrínseca del conjunto de nodos elegido, independiente de la función particular que interpolamos.*

5. ***Criterio de selección de nodos****: Para obtener buenas interpolaciones, debemos elegir nodos que minimicen* Λ_n*. Esto justifica matemáticamente la preferencia por nodos de Chebyshev sobre nodos equiespaciados.*

4.5.2 Nodos equiespaciados y el fenómeno de Runge

Definición 4.39 (Nodos equiespaciados)**.** *En el intervalo $[a, b]$, los **nodos equiespaciados** se definen como:*

$$x_k = a + k\frac{b-a}{n}, \quad k = 0, 1, \ldots, n.$$

Los nodos equiespaciados son la elección más natural y se usan frecuentemente en aplicaciones prácticas. Sin embargo, como veremos, pueden llevar a problemas de convergencia serios.

Teorema 4.40 (Fenómeno de Runge)**.** *Existen funciones analíticas para las cuales la interpolación polinomial en nodos equiespaciados diverge cuando $n \to \infty$.*

Ejemplo 4.41 (Función de Runge)**.** *Consideremos la función $f(x) = \frac{1}{1+25x^2}$ en el intervalo $[-1, 1]$. Esta función es analítica en todo el plano complejo excepto en los polos $x = \pm\frac{i}{5}$.*

Para nodos equiespaciados $x_k = -1 + \frac{2k}{n}$, $k = 0, \ldots, n$, los polinomios interpolantes $p_n(x)$ satisfacen:

- *$p_n(x) \to f(x)$ uniformemente en $[-1/2, 1/2]$ cuando $n \to \infty$.*
- *$p_n(x)$ diverge cerca de los extremos $x = \pm 1$, con oscilaciones de amplitud creciente.*

Explicación del fenómeno. El polinomio interpolante puede escribirse usando la fórmula de error:

$$p_n(x) = f(x) - \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{k=0}^{n}(x - x_k).$$

Para nodos equiespaciados en $[-1, 1]$, se puede demostrar que:

$$\max_{x\in[-1,1]}\left|\prod_{k=0}^{n}(x - x_k)\right| \geq \frac{2^n}{4^n} = \left(\frac{1}{2}\right)^n.$$

Sin embargo, este crecimiento no es uniforme: cerca de los extremos, $|\prod(x-x_k)|$ crece mucho más rápido que cerca del centro. Aunque f es analítica en $\mathbb{R}$, tiene polos complejos cerca del eje real. Las derivadas $f^{(n+1)}$ crecen factorialmente, y la competencia entre el decaimiento $(n+1)!$ y el crecimiento exponencial del producto determina la convergencia. □

Teorema 4.42 (Constante de Lebesgue para nodos equiespaciados)**.** *Para nodos equiespaciados en $[-1, 1]$, la constante de Lebesgue satisface:*

$$\Lambda_n \geq \frac{2^n}{en\log n}.$$

Es decir, la constante de Lebesgue crece exponencialmente con n.

Esta cota inferior explica por qué los nodos equiespaciados pueden ser problemáticos: incluso si la función tiene una excelente aproximación polinomial, la interpolación puede amplificar enormemente el error debido a la constante de Lebesgue grande.

4.5.3 Convergencia de la interpolación polinomial en L^∞

Una pregunta fundamental es: ¿bajo qué condiciones los polinomios interpolantes p_n convergen a la función f cuando $n \to \infty$? La respuesta depende tanto de las propiedades de f como de la elección de los nodos de interpolación.

Teorema 4.43 (Teorema de Faber, 1914)**.** *Para cualquier conjunto de nodos triangular $\{x_k^{(n)}\}_{k=0}^n$ con $n = 0, 1, 2, \ldots$ en $[-1, 1]$ (es decir, un esquema donde para cada n se tienen $n+1$ nodos), existe una función continua $f \in C[-1, 1]$ tal que la sucesión de polinomios interpolantes $\{p_n\}$ diverge en al menos un punto de $[-1, 1]$.*

Observación 4.44. *Este resultado negativo muestra que **no existe un conjunto universal de nodos** que garantice convergencia uniforme para todas las funciones continuas. Incluso para funciones suaves, la interpolación puede diverger si los nodos no están bien elegidos.*

A pesar de este resultado negativo, existen resultados positivos importantes bajo hipótesis adicionales:

Teoremas fundamentales de aproximación polinomial

Antes de estudiar la convergencia de interpolación, recordemos dos resultados fundamentales sobre la mejor aproximación uniforme por polinomios:

Definición 4.45 (Módulo de continuidad)**.** *Para una función $f \in C[a, b]$, el **módulo de continuidad** se define como:*

$$\omega(f, \delta) = \sup_{\substack{x,y \in [a,b] \\ |x-y| \leq \delta}} |f(x) - f(y)|.$$

Esta función mide cuánto puede variar f en un intervalo de longitud δ.

Observación 4.46. *El módulo de continuidad satisface:*

1. *$\omega(f, \delta) \to 0$ cuando $\delta \to 0$ (por continuidad uniforme en $[a, b]$).*
2. *$\omega(f, \delta)$ es creciente en δ.*
3. *$\omega(f, \lambda\delta) \leq (1 + \lambda)\omega(f, \delta)$ para $\lambda \geq 0$.*
4. *Si $f \in C^1[a, b]$, entonces $\omega(f, \delta) \leq \|f'\|_\infty \cdot \delta$.*

Teorema 4.47 (Teorema de Jackson, 1911)**.** *Sea $f \in C[-1, 1]$. Para todo $n \geq 1$, existe un polinomio $p_n \in \Pi_n$ tal que:*

$$\|f - p_n\|_\infty \leq C \cdot \omega\left(f, \frac{1}{n}\right),$$

donde C es una constante absoluta (independiente de f y n).

En particular, si $f \in C^k[-1, 1]$ para algún $k \geq 1$, entonces existe $p_n \in \Pi_n$ tal que:

$$\|f - p_n\|_\infty \leq \frac{C_k \|f^{(k)}\|_\infty}{n^k}.$$

Idea de la demostración. La demostración original de Jackson usa un núcleo de suavización específico (el **núcleo de Jackson**). La construcción moderna más simple usa los **polinomios de Bernstein**:

$$B_n(f)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}.$$

Estos polinomios tienen las siguientes propiedades:

1. $B_n(f)$ es un polinomio de grado $\leq n$.
2. $B_n(f)(x) \to f(x)$ uniformemente cuando $n \to \infty$ (teorema de aproximación de Weierstrass).
3. El error satisface:
$$|f(x) - B_n(f)(x)| \leq \frac{3}{2}\omega\left(f, \frac{1}{\sqrt{n}}\right).$$

Para obtener la tasa óptima $\omega(f, 1/n)$, Jackson utilizó un proceso de suavización iterado. La idea es:

1. Convolucionar f con un núcleo suave K_n de soporte $\sim 1/n$.
2. El resultado es una función suavizada $f_n = f * K_n$ que es aproximadamente polinomial.
3. El error es $|f - f_n| \leq C\omega(f, 1/n)$ por la construcción del núcleo.
4. Aproximar f_n por un polinomio da el resultado.

La versión para $f \in C^k$ se obtiene integrando por partes k veces en la representación integral del error. □

Teorema 4.48 (Teorema de Bernstein, 1912)**.** *Sea $f \in C[-1,1]$ y sea $E_n(f) = \inf_{p \in \Pi_n} \|f - p\|_\infty$ el error de la mejor aproximación uniforme por polinomios de grado $\leq n$. Si $E_n(f) = O(n^{-\alpha})$ para algún $\alpha > 0$, entonces $f \in C^{\lfloor \alpha \rfloor}[-1,1]$.*

Más precisamente:

1. *Si $E_n(f) = O(n^{-\alpha})$ con $0 < \alpha < 1$, entonces $\omega(f, \delta) = O(\delta^\alpha)$ (es decir, f es Hölder continua de exponente α).*
2. *Si $E_n(f) = O(n^{-k})$ para $k \in \mathbb{N}$, entonces $f \in C^{k-1}[-1,1]$ y $\omega(f^{(k-1)}, \delta) = O(\delta)$.*
3. *Si $E_n(f) = O(\rho^{-n})$ para algún $\rho > 1$, entonces f es analítica en una región compleja que contiene a $[-1,1]$.*

Idea de la demostración. La demostración se basa en las siguientes observaciones:

Caso 1: Regularidad Hölder.

Sea p_n la mejor aproximación de f de grado n. Para $h > 0$ pequeño, consideremos la diferencia:

$$\Delta_h f(x) = f(x+h) - f(x).$$

Queremos acotar $|\Delta_h f(x)|$. Como p_n aproxima bien a f:

$$|\Delta_h f(x)| \leq |\Delta_h[f - p_n](x)| + |\Delta_h p_n(x)|.$$

El primer término satisface:

$$|\Delta_h[f - p_n](x)| \leq 2\|f - p_n\|_\infty \leq 2E_n(f).$$

El segundo término, usando que $|p_n'(x)| \leq Cn^2\|p_n\|_\infty$ (desigualdad de Markov para polinomios):

$$|\Delta_h p_n(x)| \leq h\|p_n'\|_\infty \leq Chn^2\|p_n\|_\infty \leq Chn^2(\|f\|_\infty + E_n(f)).$$

Eligiendo $n \sim 1/h$ y usando $E_n(f) = O(n^{-\alpha})$, obtenemos:

$$|\Delta_h f(x)| \leq Ch^\alpha,$$

lo que implica $\omega(f, h) = O(h^\alpha)$.

Caso 2: Diferenciabilidad.

Si $E_n(f) = O(n^{-k})$ con $k \geq 1$, se puede demostrar inductivamente que f tiene $k-1$ derivadas continuas. La idea es usar que:

$$E_n(f') \leq Cn \cdot E_{n-1}(f) = O(n^{1-k}).$$

Por el caso Hölder, esto implica que $f^{(k-1)}$ existe y es Hölder continua.

Caso 3: Analiticidad.

Si $E_n(f) = O(\rho^{-n})$, los polinomios de mejor aproximación convergen exponencialmente rápido. Por teoría de funciones complejas, esto caracteriza funciones que admiten extensión analítica a una elipse de Bernstein con focos en ± 1 y suma de semiejes ρ. □

Observación 4.49 (Jackson vs. Bernstein)**.** *Los teoremas de Jackson y Bernstein son* ***conversos*** *el uno del otro:*

- ***Jackson****: regularidad de f ⇒ velocidad de aproximación.*
- ***Bernstein****: velocidad de aproximación ⇒ regularidad de f.*

Juntos, caracterizan completamente la relación entre la suavidad de una función y qué tan bien puede ser aproximada por polinomios.

Convergencia de interpolación polinomial

Ahora podemos estudiar la convergencia de interpolación usando estos resultados:

Teorema 4.50 (Convergencia para funciones analíticas con nodos de Chebyshev)**.** *Sea $f : [-1, 1] \to \mathbb{R}$ una función analítica en una región que contiene a $[-1, 1]$ en su interior. Sea p_n el polinomio que interpola a f en los nodos de Chebyshev:*

$$x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right), \quad k = 0, 1, \ldots, n.$$

Entonces $p_n \to f$ uniformemente en $[-1, 1]$ cuando $n \to \infty$, con convergencia exponencial.

Idea. La convergencia se debe a dos factores clave:

1. Para nodos de Chebyshev, $\Lambda_n = O(\log n)$ crece logarítmicamente.
2. Para funciones analíticas, existe una aproximación polinomial p_n^* con $\|f - p_n^*\|_\infty = O(\rho^{-n})$ para algún $\rho > 1$ (relacionado con el radio de analiticidad).
3. Combinando el teorema de la constante de Lebesgue: $\|f - p_n\|_\infty \leq (1 + \Lambda_n)\|f - p_n^*\|_\infty = O(\log(n) \cdot \rho^{-n}) \to 0$.

□

Teorema 4.51 (No convergencia con nodos equiespaciados)**.** *Existen funciones analíticas $f : [-1, 1] \to \mathbb{R}$ (incluso infinitamente diferenciables en todo $\mathbb{R}$) tales que los polinomios interpolantes en nodos equiespaciados divergen cuando $n \to \infty$.*

Idea. La función de Runge $f(x) = \frac{1}{1+25x^2}$ es un ejemplo explícito. El crecimiento exponencial $\Lambda_n \approx \frac{2^n}{en \log n}$ para nodos equiespaciados domina cualquier decaimiento del error de aproximación, causando divergencia cerca de los extremos del intervalo. □

Teorema 4.52 (Convergencia uniforme para funciones Lipschitz)**.** *Sea $f \in C^1[-1, 1]$ una función con derivada acotada. Sea $\{x_k^{(n)}\}$ un conjunto de nodos tal que $\Lambda_n = O(n^\alpha)$ para algún $\alpha > 0$ (por ejemplo, nodos de Chebyshev con $\alpha = 0$, o incluso algunos nodos menos óptimos con α finito). Entonces el error de interpolación satisface:*

$$\|f - p_n\|_\infty = O\left(\frac{\Lambda_n}{n}\right) = O(n^{\alpha-1}).$$

En particular, si $\alpha < 1$, entonces $p_n \to f$ uniformemente.

Idea. Por el teorema de Jackson, existe un polinomio q_n de grado n tal que:

$$\|f - q_n\|_\infty \leq \frac{C\omega(f, 1/n)}{n} \leq \frac{C\|f'\|_\infty}{n},$$

donde $\omega(f, \delta)$ es el módulo de continuidad de f. Aplicando la cota de la constante de Lebesgue:

$$\|f - p_n\|_\infty \leq (1 + \Lambda_n)\|f - q_n\|_\infty \leq \frac{C(1 + \Lambda_n)\|f'\|_\infty}{n}.$$

□

Teorema 4.53 (Teorema de Erdős-Turán)**.** *Para nodos equiespaciados en $[-1, 1]$, existe una constante $c > 0$ tal que para toda función $f \in C[-1, 1]$:*

$$\limsup_{n\to\infty} \|f - p_n\|_\infty \geq c \cdot E_n(f),$$

donde $E_n(f) = \inf_{q\in\Pi_n} \|f - q\|_\infty$ es el error de la mejor aproximación uniforme.

En particular, si $E_n(f) \not\to 0$ rápidamente (como $O(\rho^{-n})$), entonces $\|f - p_n\|_\infty$ puede no converger a cero.

Observación 4.54 (Criterios prácticos de convergencia)**.** *En la práctica, la convergencia de interpolación depende de:*

1. ***Regularidad de*** f*: Funciones analíticas convergen con nodos de Chebyshev; funciones apenas continuas pueden divergir con cualquier esquema de nodos.*

2. ***Elección de nodos****: Nodos de Chebyshev garantizan* $\Lambda_n = O(\log n)$*, lo cual es casi óptimo. Nodos equiespaciados tienen* $\Lambda_n = O(2^n/n)$*, lo cual es catastrófico.*

3. ***Comportamiento asintótico****: Para funciones suaves pero no analíticas (como* C^k *pero no* C^∞*), la convergencia con nodos de Chebyshev es algebraica:* $\|f - p_n\|_\infty = O(n^{-k})$*.*

4. ***Criterio de selección****: Para funciones generales en aplicaciones prácticas, se recomienda usar nodos de Chebyshev o Gauss-Lobatto en lugar de nodos equiespaciados, a menos que la función sea conocidamente muy suave en todo el intervalo.*

Chapter 5

Ecuaciones Diferenciales Ordinarias

5.1 Discretización de derivadas mediante diferencias finitas

La base de los métodos numéricos para ecuaciones diferenciales consiste en aproximar derivadas usando valores de la función en puntos discretos de una malla. Consideremos una función $u(x)$ suficientemente suave y una malla equiespaciada con paso $h > 0$.

5.1.1 Aproximaciones de la primera derivada

Las aproximaciones más comunes de $u'(x)$ son:

(a) **Diferencias forward**:

$$u'(x) \approx \frac{u(x+h) - u(x)}{h}$$

(b) **Diferencias backward**:

$$u'(x) \approx \frac{u(x) - u(x-h)}{h}$$

(c) **Diferencias centradas**:

$$u'(x) \approx \frac{u(x+h) - u(x-h)}{2h}$$

Análisis del error. Para analizar la precisión de estas fórmulas, utilizamos el desarrollo de Taylor. Si $u \in C^2$, tenemos:

$$u(x \pm h) = u(x) \pm u'(x)h + \frac{u''(\xi_\pm)}{2}h^2, \quad \xi_\pm \in (x, x \pm h).$$

Sustituyendo en las fórmulas de diferencias forward y backward:

$$\frac{u(x+h) - u(x)}{h} = u'(x) + \frac{u''(\xi)}{2}h,$$

es decir, ambas son aproximaciones de **orden 1**: el error es $O(h)$.

Para diferencias centradas, necesitamos $u \in C^3$ y una expansión de orden superior:

$$u(x \pm h) = u(x) \pm u'(x)h + \frac{u''(x)}{2}h^2 \pm \frac{u'''(\xi_\pm)}{6}h^3.$$

Al restar las expansiones, los términos pares se cancelan:

$$\frac{u(x+h) - u(x-h)}{2h} = u'(x) + \frac{u'''(\xi)}{6}h^2,$$

resultando en una aproximación de **orden 2**: el error es $O(h^2)$.

5.1.2 Aproximaciones de la segunda derivada

La discretización más usual para $u''(x)$ es:

$$u''(x) \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Si $u \in C^4$, expandimos hasta cuarto orden:

$$u(x \pm h) = u(x) \pm u'(x)h + \frac{u''(x)}{2}h^2 \pm \frac{u'''(x)}{6}h^3 + \frac{u^{(4)}(\xi_\pm)}{24}h^4.$$

Sumando ambas expansiones, los términos impares se cancelan:

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2 + \frac{u^{(4)}(\xi)}{12}h^4.$$

Por lo tanto:

$$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} = u''(x) + \frac{u^{(4)}(\xi)}{12}h^2.$$

Esta aproximación es también de **orden 2**.

5.1.3 Conexión con interpolación polinomial

Las fórmulas de diferencias finitas pueden interpretarse como derivadas de polinomios interpoladores. Si $p(x)$ es el polinomio que interpola u en ciertos nodos, entonces $p'(x) \approx u'(x)$ y $p''(x) \approx u''(x)$.

- **Diferencias forward/backward**: Interpolación lineal en $\{x, x+h\}$ o $\{x-h, x\}$, respectivamente. El polinomio de grado 1 es $p(t) = u(x) + \frac{u(x+h)-u(x)}{h}(t-x)$, cuya derivada es constante e igual a la diferencia dividida.
- **Diferencias centradas (primera derivada)**: Interpolación lineal en $\{x-h, x+h\}$. La derivada en x del polinomio $p(t) = u(x-h) + \frac{u(x+h)-u(x-h)}{2h}(t-(x-h))$ evaluada en x da la fórmula centrada.
- **Diferencias centradas (segunda derivada)**: Interpolación cuadrática en $\{x-h, x, x+h\}$. Usando la fórmula de Lagrange o Newton, la segunda derivada del polinomio interpolador resulta constante e igual a $\frac{u(x+h)-2u(x)+u(x-h)}{h^2}$.

Esta observación es fundamental: las fórmulas de diferencias finitas son casos particulares de diferenciación numérica mediante interpolación polinomial. Los órdenes de precisión pueden entenderse en términos del error de interpolación y la suavidad de la función.

5.2 Problemas de valores iniciales

Consideremos el problema de valores iniciales (PVI):

$$\frac{du}{dt} = f(t, u), \quad u(t_0) = u_0,$$

donde buscamos $u(t)$ para $t \in [t_0, T]$. Los métodos numéricos construyen una aproximación discreta $u_n \approx u(t_n)$ en puntos $t_n = t_0 + nh$ con paso $h > 0$.

Método de Euler explícito. El más simple es aproximar $u'(t_n) \approx (u_{n+1} - u_n)/h$ (diferencias forward):

$$u_{n+1} = u_n + hf(t_n, u_n).$$

Este método tiene error local $O(h^2)$ (por desarrollo de Taylor) y error global $O(h)$ tras $N = T/h$ pasos. Es fácil de implementar pero requiere h pequeño para estabilidad.

Métodos de Runge-Kutta. Para mejorar la precisión sin usar derivadas de f, evaluamos f en puntos intermedios. El **método RK4** (Runge-Kutta de orden 4) es:

$$\begin{aligned} k_1 &= f(t_n, u_n), \\ k_2 &= f(t_n + h/2, u_n + hk_1/2), \\ k_3 &= f(t_n + h/2, u_n + hk_2/2), \\ k_4 &= f(t_n + h, u_n + hk_3), \\ u_{n+1} &= u_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

Este método tiene error global $O(h^4)$, muy superior a Euler con solo 4 evaluaciones de f por paso.

Estabilidad. Para el problema lineal de prueba $u' = \lambda u$ con $\lambda < 0$ (decaimiento), un método es **A-estable** si captura el comportamiento asintótico correcto para todo $h > 0$. Euler explícito solo es estable si $|1 + h\lambda| \leq 1$, i.e., $h \leq 2/|\lambda|$ (restricción severa para problemas rígidos). Los **métodos implícitos** como el trapecio implícito:

$$u_{n+1} = u_n + \frac{h}{2}[f(t_n, u_n) + f(t_{n+1}, u_{n+1})]$$

son A-estables pero requieren resolver una ecuación (posiblemente no lineal) en cada paso.

Métodos adaptativos. En la práctica, usamos esquemas que ajustan h automáticamente. El par **Dormand-Prince 5(4)** (usado en `scipy.integrate.solve_ivp` y MATLAB's `ode45`) combina dos métodos RK de órdenes 4 y 5 para estimar el error local:

$$\text{error estimado} \approx \|u_{n+1}^{(5)} - u_{n+1}^{(4)}\|.$$

Si el error excede la tolerancia, se reduce h; si es muy pequeño, se aumenta h.

5.2.1 Convergencia de métodos de un paso

Los resultados previos sobre Euler y Runge-Kutta son casos particulares de una teoría general de convergencia. El siguiente teorema establece que, para métodos de un paso, el error global está controlado por el error de truncamiento local.

Teorema 5.1. *Para $n \in \mathbb{N}$ y $h = (T - t_0)/n$ consideremos el método de un paso dado por*

$$u_{i+1} = u_i + h\Phi(t_i, u_i, h),$$

con Φ una función Lipschitz en la segunda variable, es decir, existe una constante K independiente de t y h tal que

$$|\Phi(t, x, h) - \Phi(t, y, h)| \leq K|x - y|$$

para todo $x, y \in \mathbb{R}$ y $t \in [t_0, T]$. Entonces, se tiene

$$|u(T) - u_n| \leq \frac{\tau_{\max}}{K}\left(e^{K(T-t_0)} - 1\right),$$

donde $\tau_{\max} = \max_{1\leq i\leq n}\{|\tau_i|\}$ con τ_i el error de truncamiento local del método en el paso i.

Demostración. Sea $e_i = u(t_i) - u_i$ el error global cometido hasta el paso i. Consideramos la ecuación del método y la definición de τ_i (el error de truncamiento local satisface $u(t_{i+1}) = u(t_i) + h\Phi(t_i, u(t_i), h) + h\tau_i$):

$$\begin{aligned} u_{i+1} &= u_i + h\Phi(t_i, u_i, h), \\ u(t_{i+1}) &= u(t_i) + h\Phi(t_i, u(t_i), h) + h\tau_i. \end{aligned}$$

Luego,

$$e_{i+1} = u(t_{i+1}) - u_{i+1} = e_i + h\left(\Phi(t_i, u(t_i), h) - \Phi(t_i, u_i, h)\right) + h\tau_i.$$

Usando que Φ es Lipschitz se obtiene

$$\begin{aligned} |e_{i+1}| &\leq |e_i| + h|\Phi(t_i, u(t_i), h) - \Phi(t_i, u_i, h)| + h|\tau_i| \\ &\leq |e_i| + hK|u(t_i) - u_i| + h|\tau_i| \\ &\leq (1 + Kh)|e_i| + h|\tau_i|. \end{aligned}$$

Como $|\tau_i| \leq \tau_{\max}$, llamando $A = 1 + Kh$, tenemos

$$|e_{i+1}| \leq A|e_i| + h\tau_{\max}.$$

Usando esta acotación para $i = 0$ y teniendo en cuenta que $e_0 = 0$ (la condición inicial es exacta) se tiene

$$\begin{aligned} &|e_1| \leq h\tau_{\max}, \\ &|e_2| \leq A|e_1| + h\tau_{\max} \leq Ah\tau_{\max} + h\tau_{\max} = (1 + A)h\tau_{\max}, \\ &|e_3| \leq A|e_2| + h\tau_{\max} \leq A(1 + A)h\tau_{\max} + h\tau_{\max} = (1 + A + A^2)h\tau_{\max}. \end{aligned}$$

Iterando el procedimiento, podemos ver que

$$|e_n| \leq (1 + A + A^2 + \cdots + A^{n-1})h\tau_{\max} = h\tau_{\max}\sum_{j=0}^{n-1} A^j.$$

Ahora, usando la fórmula para suma de potencias tenemos

$$|e_n| \leq h\tau_{\max}\frac{A^n - 1}{A - 1} = h\tau_{\max}\frac{(1 + Kh)^n - 1}{Kh} = \frac{\tau_{\max}}{K}\left((1 + Kh)^n - 1\right).$$

Como para $\alpha > 0$ se tiene $(1+\alpha)^n \leq e^{n\alpha}$, entonces

$$|e_n| \leq \frac{\tau_{\max}}{K}\left(e^{nKh} - 1\right) = \frac{\tau_{\max}}{K}\left(e^{K(T-t_0)} - 1\right),$$

lo que concluye la demostración. □

Observación 5.2. *Bajo las hipótesis del teorema anterior se deduce que un método de un paso es* ***convergente*** *(es decir, $|u(T) - u_n| \to 0$ cuando $h \to 0$) si y solo si $\lim_{h\to 0} \tau_{\max} = 0$. Esta condición se denomina* ***consistencia*** *del método. Puede demostrarse que, para un método de un paso con función de incremento Φ continua en sus tres variables, se tiene $\lim_{h\to 0} \tau_{\max} = 0$ si y solo si $\Phi(t, u, 0) = f(t, u)$. Los métodos de Euler, Taylor y Runge-Kutta satisfacen esta condición y, por lo tanto, son convergentes.*

5.2.2 Convergencia y Estabilidad de métodos multipaso

Para los métodos multipaso lineales (LMM), la relación entre consistencia y convergencia es más sutil que para los métodos de un paso. Mientras que para métodos de un paso (como Euler o Runge-Kutta) la consistencia implica convergencia (bajo condiciones de suavidad de f), para los LMM necesitamos una condición adicional de estabilidad.

Consideremos el método multipaso lineal general de k pasos:

$$\sum_{j=0}^{k} \alpha_j u_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j},$$

donde normalizamos asumiendo $\alpha_k = 1$. Asociamos a este método los polinomios característicos:

$$\rho(z) = \sum_{j=0}^{k} \alpha_j z^j, \quad \sigma(z) = \sum_{j=0}^{k} \beta_j z^j.$$

Definición 5.3 (Consistencia)**.** *Un LMM se dice* ***consistente*** *si tiene orden de consistencia al menos 1, es decir, si el error de truncamiento local satisface $\tau_n(h) \to 0$ cuando $h \to 0$. En términos de los polinomios característicos, esto equivale a las condiciones:*

$$\rho(1) = 0 \quad y \quad \rho'(1) = \sigma(1).$$

La consistencia por sí sola no es suficiente. Si las raíces del polinomio $\rho(z)$ no se comportan adecuadamente, los errores pueden amplificarse exponencialmente al avanzar los pasos, incluso si el error local es pequeño.

Ejemplo 5.4 (Consistencia no implica convergencia)**.** *Consideremos el método multipaso dado por:*

$$u^{n+2} - 3u^{n+1} + 2u^n = -hf(t_n, u^n).$$

El error de truncamiento local resulta:

$$\tau^n = \frac{1}{h}[u(t_{n+2}) - 3u(t_{n+1}) + 2u(t_n)] + u'(t_n) = \frac{5}{2}hu''(t_n) + O(h^2),$$

por lo que ***el método es consistente*** *(de orden 1).*

Sin embargo, el método ***no converge****. Para verlo, consideremos el problema test* $u'(t) = 0$, $u(0) = 0$. *El método toma la forma:*

$$u^{n+2} - 3u^{n+1} + 2u^n = 0.$$

El polinomio característico es $\rho(z) = z^2 - 3z + 2 = (z-1)(z-2)$, *con raíces* $z_1 = 1$ *y* $z_2 = 2$. *La solución general de la recurrencia es:*

$$u^n = C_1 \cdot 1^n + C_2 \cdot 2^n = C_1 + C_2 \cdot 2^n.$$

Para implementar el método necesitamos ***dos valores iniciales****,* u^0 *y* u^1. *Si tomamos* $u^0 = u^1 = 0$, *entonces* $C_1 = C_2 = 0$ *y el método converge a la solución exacta. Pero en la práctica,* u^1 *se obtiene con otro método (por ejemplo, Euler) y tendrá un error pequeño.*

Consideremos $u^0 = 0$ *y* $u^1 = h$ *(un error del orden del paso). Las constantes satisfacen:*

$$\begin{cases} C_1 + C_2 = 0 \\ C_1 + 2C_2 = h \end{cases} \quad \Rightarrow \quad C_1 = -h,\ C_2 = h.$$

Por lo tanto:

$$u^n = -h + h \cdot 2^n = h(2^n - 1).$$

En el tiempo final T, *con* $n = T/h$*:*

$$u^n = h(2^{T/h} - 1) \to \infty \quad \textit{cuando } h \to 0.$$

A pesar de ser consistente, ***el método diverge exponencialmente*** *debido a la raíz* $|z_2| = 2 > 1$.

Este ejemplo motiva la siguiente definición crucial:

Definición 5.5 (0-Estabilidad)**.** *Un LMM se dice* ***0-estable*** *(o que satisface la* ***condición de la raíz****) si todas las raíces* z *del primer polinomio característico* $\rho(z)$ *satisfacen:*

1. $|z| \leq 1$.
2. *Si* $|z| = 1$, *entonces* z *es una raíz simple.*

El siguiente teorema, debido a Germund Dahlquist, es el pilar fundamental de la teoría de métodos multipaso.

Teorema 5.6 (Teorema de Equivalencia de Dahlquist)**.** *Un método multipaso lineal consistente es convergente si y solo si es 0-estable.*

Demostración. **Necesidad (⇒):** Supongamos que el método es convergente. Consideremos la ecuación diferencial trivial $u' = 0, u(0) = 0$, cuya solución exacta es $u(t) = 0$. Al aplicar el método numérico a esta ecuación (donde $f(t,u) = 0$), obtenemos la relación de recurrencia lineal homogénea con coeficientes constantes:

$$\sum_{j=0}^{k} \alpha_j u_{n+j} = 0.$$

La solución general de esta recurrencia se puede expresar en términos de las raíces $z_1, \ldots, z_m$ de $\rho(z)$. Si z_i es una raíz de multiplicidad μ_i, los términos asociados son $z_i^n, nz_i^n, \ldots, n^{\mu_i - 1} z_i^n$.

- Si existiera una raíz con $|z_i| > 1$, el término z_i^n crecería exponencialmente en magnitud con n. Dado que $n = T/h$, cuando $h \to 0$, $n \to \infty$, y $|u_n| \to \infty$ para casi cualquier condición inicial (errores de redondeo infinitesimales excitarían este modo). Esto contradice la convergencia a 0.

- Si existiera una raíz con $|z_i| = 1$ y multiplicidad $\mu_i > 1$, la solución contendría un término de la forma $n^{\mu_i - 1} z_i^n$. Su magnitud crece polinomialmente con n. Nuevamente, como $n \to \infty$ cuando $h \to 0$, el error no acotado impide la convergencia.

Por lo tanto, para que el método sea convergente, es necesario que todas las raíces satisfagan $|z| \leq 1$ y las de módulo 1 sean simples.

Suficiencia ($\Leftarrow$): Supongamos que el método es consistente y 0-estable. Sea $e_n = u(t_n) - u_n$ el error global. La solución exacta satisface el esquema numérico salvo por el error de truncamiento local τ_n:

$$\sum_{j=0}^{k} \alpha_j u(t_{n+j}) = h \sum_{j=0}^{k} \beta_j f(t_{n+j}, u(t_{n+j})) + h\tau_{n+k}.$$

Restando la ecuación del método numérico $\sum_{j=0}^{k} \alpha_j u_{n+j} = h \sum_{j=0}^{k} \beta_j f(t_{n+j}, u_{n+j})$, obtenemos la ecuación para el error:

$$\sum_{j=0}^{k} \alpha_j e_{n+j} = h \sum_{j=0}^{k} \beta_j [f(t_{n+j}, u(t_{n+j})) - f(t_{n+j}, u_{n+j})] + h\tau_{n+k}.$$

Usando la condición de Lipschitz para f con constante L, tenemos $|f(t,u) - f(t,v)| \leq L|u - v|$. Sea g_n tal que la diferencia de las f es $g_n e_{n+j}$ con $|g_n| \leq L$. Podemos reescribir la ecuación como:

$$\sum_{j=0}^{k} \alpha_j e_{n+j} = \gamma_n, \quad \text{donde } \gamma_n = h \sum_{j=0}^{k} \beta_j g_{n+j} e_{n+j} + h\tau_{n+k}.$$

Esta es una ecuación en diferencias lineal inhomogénea para e_n. Debido a la 0-estabilidad, las raíces de $\rho(z)$ se comportan bien, lo que implica que el operador inverso es estable. Existe una constante K (que depende solo de los coeficientes α_j) tal que la solución de $\sum \alpha_j e_{n+j} = \gamma_n$ está acotada por:

$$|e_n| \leq K \left(\sum_{i=0}^{k-1} |e_i| + \sum_{m=0}^{n-k} |\gamma_m| \right).$$

Sustituyendo γ_m:

$$|e_n| \leq K \left(\sum_{i=0}^{k-1} |e_i| + \sum_{m=0}^{n-k} \left(hL \sum_{j=0}^{k} |\beta_j| |e_{m+j}| + h|\tau_{m+k}| \right) \right).$$

Sea $B = \sum |\beta_j|$ y supongamos que el método es consistente de orden p, tal que $|\tau_m| \leq Ch^p$. Asumimos también que los errores iniciales son del orden adecuado $\sum_{i=0}^{k-1} |e_i| \leq C_0 h^p$.

$$|e_n| \leq KC_0 h^p + Kh \sum_{m=0}^{n-k} Ch^p + KhLB \sum_{m=0}^{n-k} \sum_{j=0}^{k} |e_{m+j}|.$$

El término de la suma de τ es $\approx n \cdot h \cdot h^p \approx Th^p$. Reorganizando los términos de la suma doble, podemos acotar por una suma simple sobre los errores pasados (ajustando la constante K'):

$$|e_n| \leq C'h^p + K'h \sum_{m=0}^{n-1} |e_m|.$$

Esta es la forma discreta de la desigualdad de Grönwall. Su resolución implica:

$$|e_n| \leq C'h^p \exp(K'nh) = C'h^p \exp(K'T).$$

Como T es fijo, esto demuestra que $|e_n| \to 0$ cuando $h \to 0$, y además que el orden de convergencia es p. □

Ejemplo 5.7 (Análisis de estabilidad).

- ***Método de Adams-Bashforth (2 pasos)****:* $u_{n+2} - u_{n+1} = \dots$. $\rho(z) = z^2 - z = z(z-1)$. *Raíces:* $0, 1$. *Ambas* ≤ 1, 1 *es simple.* ***Es 0-estable****.*
- ***Método inestable****:* $u_{n+2} + 4u_{n+1} - 5u_n = \dots$. $\rho(z) = z^2 + 4z - 5 = (z+5)(z-1)$. *Raíces:* $1, -5$. *Como* $|-5| > 1$, ***no es 0-estable*** *y no convergerá.*
- ***Diferencias centradas para*** $u' = f$*: El método* $\frac{u^{n+1} - u^{n-1}}{2h} = f(t_n, u^n)$ *tiene* $\rho(z) = z^2 - 1 = (z-1)(z+1)$. *Raíces:* ± 1. *Ambas sobre el círculo unitario, pero la raíz* -1 *genera oscilaciones espurias. Aunque es 0-estable, su comportamiento numérico es pobre.*

Corolario 5.8. *Los métodos de un paso son automáticamente 0-estables si son consistentes.*

Demostración. Para $k = 1$, el polinomio $\rho(z) = \alpha_0 + \alpha_1 z$ tiene una única raíz $z = -\alpha_0/\alpha_1$. La condición de consistencia $\rho(1) = 0$ implica $\alpha_0 + \alpha_1 = 0$, es decir, $\alpha_0 = -\alpha_1$. Por lo tanto, $z = 1$, que satisface la condición de la raíz. □

Observación 5.9. *Este corolario explica por qué en métodos de un paso (Euler, Runge-Kutta) solo necesitamos verificar consistencia para garantizar convergencia, mientras que en métodos multipaso debemos verificar además la 0-estabilidad.*

5.2.3 A-estabilidad y estabilidad práctica

Mientras que la 0-estabilidad es una propiedad intrínseca del método, la A-estabilidad caracteriza el comportamiento del método aplicado a la ecuación de prueba de Dahlquist $u' = \lambda u$ con $\mathrm{Re}(\lambda) < 0$.

Definición 5.10 (Ecuación de prueba de Dahlquist). *La* ***ecuación de prueba*** *es:*

$$u'(t) = \lambda u(t), \quad u(0) = u_0,$$

donde $\lambda \in \mathbb{C}$ *con* $Re(\lambda) \leq 0$. *La solución exacta es* $u(t) = u_0 e^{\lambda t}$.

Esta ecuación es fundamental porque captura el comportamiento de sistemas lineales y de la linealización local de sistemas no lineales. Para $\mathrm{Re}(\lambda) < 0$, la solución exacta decae exponencialmente a cero.

Definición 5.11 (Función de estabilidad para métodos multipaso)**.** *Aplicando un método multipaso a la ecuación de prueba $u' = \lambda u$, obtenemos:*

$$\sum_{j=0}^{k} \alpha_j u_{n+j} = h\lambda \sum_{j=0}^{k} \beta_j u_{n+j}.$$

Buscando soluciones de la forma $u_n = r^n$, donde r es un número complejo, llegamos a:

$$\sum_{j=0}^{k} \alpha_j r^j = h\lambda \sum_{j=0}^{k} \beta_j r^j,$$

o equivalentemente:

$$\rho(r) = h\lambda\sigma(r),$$

donde $\rho(z) = \sum_{j=0}^{k} \alpha_j z^j$ y $\sigma(z) = \sum_{j=0}^{k} \beta_j z^j$ son los polinomios característicos del método.

Definición 5.12 (Región de estabilidad absoluta)**.** *La **región de estabilidad absoluta** S de un método multipaso es el conjunto:*

$$S = \{z \in \mathbb{C} : \text{todas las raíces } r \text{ de } \rho(r) - z\sigma(r) = 0 \text{ satisfacen } |r| \leq 1\}.$$

Alternativamente, S es el conjunto de valores $z = h\lambda$ para los cuales el método produce soluciones acotadas.

Definición 5.13 (A-estabilidad)**.** *Un método numérico es **A-estable** si su región de estabilidad absoluta contiene todo el semiplano izquierdo complejo:*

$$S \supseteq \{z \in \mathbb{C} : Re(z) < 0\}.$$

Observación 5.14 (Interpretación física)**.** *Un método A-estable preserva la propiedad cualitativa de la solución exacta (decaimiento para $Re(\lambda) < 0$) para cualquier tamaño de paso $h > 0$. Esto es especialmente importante para **problemas rígidos** (stiff), donde diferentes componentes de la solución decaen a tasas muy diferentes.*

Teorema 5.15 (Primer teorema de barrera de Dahlquist)**.** *No existe ningún método multipaso lineal explícito que sea A-estable.*

Idea de la demostración. Para un método explícito, tenemos $\beta_k = 0$ (el lado derecho no involucra f_{n+k}). Por lo tanto, $\sigma(z)$ tiene grado menor que k, mientras que $\rho(z)$ tiene grado exactamente k.

Para valores grandes de $|z|$, la ecuación $\rho(r) = z\sigma(r)$ se comporta asintóticamente como $\alpha_k r^k = z \cdot O(r^{k-1})$, lo que da $r \sim \frac{z\beta_{k-1}}{\alpha_k}$ (más otras raíces).

Para $z \to -\infty$ (que está en el semiplano izquierdo), al menos una de estas raíces satisface $|r| > 1$, por lo que $z \notin S$. Por tanto, S no puede contener todo el semiplano izquierdo, y el método no es A-estable. □

Teorema 5.16 (Segundo teorema de barrera de Dahlquist)**.** *El orden máximo de un método multipaso lineal A-estable es 2. Además, el método de orden 2 con menor error asintótico es la **regla del trapecio**:*

$$u_{n+1} - u_n = \frac{h}{2}(f_{n+1} + f_n).$$

Esquema de la demostración. La demostración completa utiliza técnicas de análisis complejo y teoría de polinomios. Las ideas principales son:

Paso 1: Para que un método sea A-estable, la función racional $R(z) = \frac{\rho(e^z)}{\sigma(e^z)}$ debe satisfacer ciertas propiedades en el semiplano izquierdo.

Paso 2: Usando la consistencia (que relaciona ρ y σ en $z = 1$) y condiciones de orden, se demuestra que el grado del numerador y denominador están restringidos.

Paso 3: Un análisis detallado muestra que solo métodos de orden ≤ 2 pueden satisfacer simultáneamente todas las condiciones requeridas para A-estabilidad.

Paso 4: La regla del trapecio, con $\rho(z) = z - 1$ y $\sigma(z) = \frac{1}{2}(z + 1)$, satisface todas las condiciones y alcanza orden 2. □

Ejemplo 5.17 (Análisis de estabilidad: Euler explícito vs. Euler implícito)**.** *Comparemos el comportamiento de los métodos de Euler en la ecuación de prueba $u' = \lambda u$ con $Re(\lambda) < 0$.*

***Euler explícito**: $u_{n+1} = u_n + h\lambda u_n = (1 + h\lambda)u_n = R(h\lambda)u_n$.*

La función de estabilidad es $R(z) = 1 + z$. Para estabilidad, requerimos $|1 + z| \leq 1$. Escribiendo $z = h\lambda = h(\alpha + i\beta)$ con $\alpha < 0$:

$$|1 + h\alpha + ih\beta|^2 = (1 + h\alpha)^2 + h^2\beta^2 \leq 1.$$

Expandiendo: $1 + 2h\alpha + h^2(\alpha^2 + \beta^2) \leq 1$, es decir:

$$h \leq \frac{-2\alpha}{\alpha^2 + \beta^2} = \frac{2|Re(\lambda)|}{|\lambda|^2}.$$

Si λ es real negativo ($\lambda = \alpha < 0$), esto se reduce a:

$$h < \frac{2}{|\lambda|}.$$

***Conclusión**: El método de Euler explícito requiere una* restricción sobre el paso temporal *para ser estable. No es A-estable.*

***Euler implícito**: $u_{n+1} = u_n + h\lambda u_{n+1}$, de donde:*

$$u_{n+1}(1 - h\lambda) = u_n \quad \Rightarrow \quad u_{n+1} = \frac{1}{1 - h\lambda}u_n = R(h\lambda)u_n.$$

La función de estabilidad es $R(z) = \frac{1}{1-z}$. Para $Re(z) < 0$, escribimos $z = x + iy$ con $x < 0$:

$$|R(z)|^2 = \frac{1}{|1 - z|^2} = \frac{1}{(1 - x)^2 + y^2}.$$

Como $x < 0$, tenemos $1 - x > 1$, por lo que $(1 - x)^2 > 1$, y así:

$$|R(z)|^2 < 1 \quad \text{para todo } Re(z) < 0.$$

***Conclusión**: El método de Euler implícito es A-estable. No hay restricción sobre h para estabilidad (aunque la precisión puede degradarse con h grande).*

Observación 5.18 (Implicaciones para problemas rígidos)**.** *La diferencia entre Euler explícito e implícito es dramática para **problemas rígidos** (stiff), donde el sistema tiene autovalores λ con $|Re(\lambda)| \gg 1$. Por ejemplo, si $\lambda = -1000$:*

- *Euler explícito requiere $h < 0.002$ para estabilidad, forzando pasos muy pequeños.*
- *Euler implícito permite pasos grandes arbitrarios sin inestabilidad numérica.*

Aunque el costo por paso del método implícito es mayor (resolver sistema lineal), puede compensarse con pasos mucho mayores.

Ejemplo 5.19 (Verificación de A-estabilidad de la regla del trapecio)**.** *Para la regla del trapecio: $u_{n+1} - u_n = \frac{h}{2}(f_{n+1} + f_n)$.*

Aplicando a $u' = \lambda u$:

$$u_{n+1} - u_n = \frac{h\lambda}{2}(u_{n+1} + u_n).$$

Reorganizando:

$$u_{n+1}\left(1 - \frac{h\lambda}{2}\right) = u_n\left(1 + \frac{h\lambda}{2}\right),$$

de donde:

$$u_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} u_n = R(h\lambda)u_n,$$

donde $R(z) = \frac{1+z/2}{1-z/2}$ es la función de estabilidad.

Para estabilidad, necesitamos $|R(z)| \leq 1$. Si $z = h\lambda$ con $Re(z) < 0$, escribamos $z = x + iy$ con $x < 0$:

$$|R(z)|^2 = \left|\frac{1 + z/2}{1 - z/2}\right|^2 = \frac{|1 + x/2 + iy/2|^2}{|1 - x/2 - iy/2|^2} = \frac{(1 + x/2)^2 + y^2/4}{(1 - x/2)^2 + y^2/4}.$$

Como $x < 0$, tenemos $1 + x/2 < 1 < 1 - x/2$, por lo que $(1 + x/2)^2 < (1 - x/2)^2$. Así:

$$|R(z)|^2 < 1 \quad \text{para todo } Re(z) < 0.$$

Por lo tanto, la regla del trapecio es A-estable.

Observación 5.20 (Consecuencias prácticas)**.** *Los teoremas de barrera de Dahlquist tienen consecuencias importantes:*

1. *Los métodos explícitos multipaso no pueden ser A-estables, lo que limita su uso para problemas rígidos.*
2. *Los métodos implícitos de orden alto ($p \geq 3$) tampoco pueden ser A-estables.*
3. *Para problemas rígidos, se debe elegir entre: (a) usar métodos de orden bajo A-estables como el trapecio, o (b) usar métodos de orden alto con regiones de estabilidad grandes pero no A-estables.*
4. *Esto ha motivado el desarrollo de los **métodos BDF** (Backward Differentiation Formulas), que sacrifican A-estabilidad por A(α)-estabilidad (una versión relajada) y pueden alcanzar orden hasta 6.*

5.3 Problemas de valores de contorno en 1D

A diferencia de los problemas de valores iniciales (PVI), en los problemas de valores de contorno (PVC) las condiciones se especifican en los extremos del intervalo. Esta aparente diferencia sutil tiene consecuencias profundas: mientras los PVI se resuelven marchando en el tiempo, los PVC requieren resolver un sistema de ecuaciones algebraicas.

5.3.1 Condiciones de borde de Dirichlet

Consideremos la ecuación de Poisson unidimensional con condiciones de Dirichlet:

$$\begin{cases} -u_{xx}(x) = & f(x), \quad \text{para } x \in (0,1) \\ u(0) = & \alpha \\ u(1) = & \beta. \end{cases} \tag{5.1}$$

Discretización por diferencias finitas. Consideramos la malla uniforme $\{x_j = jh,\ j = 0, 1, 2, \ldots, m+1\}$ con $h = 1/(m+1)$. Para los puntos interiores $x_j \in (0,1)$, el esquema de diferencias centradas para la derivada segunda conduce al sistema:

$$\frac{1}{h^2}\left(u_{j-1} - 2u_j + u_{j+1}\right) = f(x_j), \quad j = 1, 2, \ldots, m.$$

Utilizando las condiciones de borde $u_0 = \alpha$, $u_{m+1} = \beta$, obtenemos el sistema lineal:

$$A^h u^h = F^h, \tag{5.2}$$

donde $u^h = [u_1, u_2, \ldots, u_m]^T$ es el vector de incógnitas, y:

$$A^h = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}, \quad F^h = \begin{bmatrix} f(x_1) - \alpha/h^2 \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ f(x_m) - \beta/h^2 \end{bmatrix}. \tag{5.3}$$

Definición 5.21 (Error de discretización y de truncamiento)**.** *Para h fijo, definimos:*

- ***Error de discretización**: $e_j^h = u(x_j) - u_j^h$, donde u es la solución exacta y u^h la aproximada.*
- ***Error de truncamiento local**: $\tau_j^h = (A^h U)_j - f(x_j)$, donde U denota la solución exacta evaluada en la malla: $U_j = u(x_j)$.*

Teorema 5.22 (Error de truncamiento)**.** *Si $u \in C^4[0,1]$, existe una constante C independiente de h tal que:*

$$\max_{1 \le j \le m} \left|\tau_j^h\right| \le Ch^2.$$

Demostración. Como $u \in C^4$, podemos usar la fórmula del error de diferencias centradas: $\tau_j^h = -\frac{h^2}{12}u^{(4)}(\xi_j)$ para algún $\xi_j \in (x_{j-1}, x_{j+1})$. Como $u^{(4)}$ es continua en $[0,1]$, está acotada: $|u^{(4)}(x)| \leq M$ para todo $x \in [0,1]$. Por tanto:

$$|\tau_j^h| \leq \frac{h^2}{12}M = Ch^2.$$

□

Relación entre errores. Aplicando A^h al error de discretización:

$$A^h e^h = A^h U - A^h u^h = A^h U - F^h = \tau^h.$$

Por tanto: $e^h = (A^h)^{-1}\tau^h$, y:

$$\|e^h\| \leq \|(A^h)^{-1}\| \cdot \|\tau^h\|.$$

Para garantizar convergencia, necesitamos que $(A^h)^{-1}$ esté acotada uniformemente en h:

Definición 5.23 (Estabilidad)**.** *Una discretización es* ***estable*** *en norma $\|\cdot\|$ si existe una constante C independiente de h tal que:*

$$\|(A^h)^{-1}\| \leq C \quad \textit{para todo } h.$$

Teorema 5.24 (Autovalores y autovectores de la matriz tridiagonal)**.** *Dados $a, b, c \in \mathbb{R}$, consideremos la matriz tridiagonal $A \in \mathbb{R}^{n\times n}$:*

$$A = \begin{bmatrix} a & b & & & & \\ c & a & b & & & \\ & c & a & b & & \\ & & \ddots & \ddots & \ddots & \\ & & & c & a & b \\ & & & & c & a \end{bmatrix}.$$

Llamando $h = 1/(n+1)$ y $w^2 = c/b$, el q-ésimo autovector r^q de A está dado por:

$$r_j^q = w^j \sin(q\pi jh), \quad j = 1, \ldots, n,$$

y el correspondiente autovalor es:

$$\lambda^q = a + 2\sqrt{bc}\cos(q\pi h).$$

Demostración. Verificamos que $Ar^q = \lambda^q r^q$. Para $1 < j < n$:

$$\begin{aligned}(Ar^q)_j &= cr_{j-1}^q + ar_j^q + br_{j+1}^q \\ &= cw^{j-1}\sin(q\pi(j-1)h) + aw^j\sin(q\pi jh) + bw^{j+1}\sin(q\pi(j+1)h) \\ &= w^j\Big[cw^{-1}\sin(q\pi(j-1)h) + a\sin(q\pi jh) + bw\sin(q\pi(j+1)h)\Big].\end{aligned}$$

Usando la identidad $\sin(q\pi(j\pm1)h) = \sin(q\pi jh)\cos(q\pi h) \pm \cos(q\pi jh)\sin(q\pi h)$, y observando que el segundo término se cancela (por simetría), obtenemos:

$$(Ar^q)_j = w^j \sin(q\pi jh) \cdot \left(cw^{-1}\cos(q\pi h) + a + bw\cos(q\pi h)\right).$$

Como $w^2 = c/b$, tenemos $w = \sqrt{c/b}$, por lo que $cw^{-1} = \sqrt{bc}$ y $bw = \sqrt{bc}$. Así:

$$\lambda^q = a + 2\sqrt{bc}\cos(q\pi h).$$

Los casos $j = 1$ y $j = n$ se verifican análogamente usando que $r_0^q = r_{n+1}^q = 0$. □

Corolario 5.25 (Estabilidad de la discretización de Dirichlet)**.** *Para la matriz A^h en* (5.3) *(con $a = -2/h^2$, $b = c = 1/h^2$), los autovalores son:*

$$\lambda^q = \frac{2}{h^2}\left(\cos(q\pi h) - 1\right), \quad q = 1, \ldots, m.$$

El autovalor de mínimo módulo es:

$$\lambda_1 = \frac{2}{h^2}\left(\cos(\pi h) - 1\right) = \frac{2}{h^2}\left(-\frac{\pi^2 h^2}{2} + O(h^4)\right) = -\pi^2 + O(h^2).$$

Por tanto, como A^h es simétrica:

$$\|(A^h)^{-1}\|_2 = \frac{1}{|\lambda_1|} \leq \frac{1}{\pi^2} + O(h^2) = O(1).$$

La discretización es estable en norma 2, y el error global es:

$$\|e^h\|_2 \leq Ch^2.$$

5.3.2 Condiciones de borde de Neumann

Consideremos ahora un problema con condición de Neumann en un extremo:

$$\begin{cases} -u_{xx}(x) = & f(x), \quad \text{para } x \in (0,1) \\ u_x(0) = & \sigma \\ u(1) = & \beta. \end{cases} \tag{5.4}$$

Método 1: Nodo ficticio. Usamos diferencias centradas para $u_x(0)$:

$$\frac{u_1 - u_{-1}}{2h} = \sigma.$$

Evaluando la ecuación en $j = 0$:

$$\frac{1}{h^2}\left(u_{-1} - 2u_0 + u_1\right) = f(x_0).$$

Despejando $u_{-1} = u_1 - 2h\sigma$ y sustituyendo:

$$\frac{u_1 - u_0}{h} = \sigma + \frac{h}{2}f(x_0).$$

Esta fórmula es una corrección de orden $O(h^2)$ a la diferencia forward. En efecto, considerando $u'(x_0 + h/2)$:

$$u'\left(x_0 + \frac{h}{2}\right) = u'(x_0) + \frac{h}{2}u''(x_0) + O(h^2) = \sigma - \frac{h}{2}f(x_0) + O(h^2).$$

Usando diferencias centradas en $x_0 + h/2$:

$$u'\left(x_0 + \frac{h}{2}\right) \approx \frac{u_1 - u_0}{h}.$$

Igualando, obtenemos la fórmula deseada con error $O(h^2)$.

Método 2: Diferencia forward de orden 2. Usando la fórmula de tres puntos:

$$u'(x_0) = \frac{-3u_0 + 4u_1 - u_2}{2h} + O(h^2) = \sigma.$$

Esta aproximación también tiene error $O(h^2)$ y no requiere nodo ficticio.

5.3.3 Condiciones de borde periódicas

Consideremos la ecuación con periodicidad:

$$\begin{cases} -u_{xx}(x) = & f(x), \quad \text{para } x \in (0, 2\pi) \\ u(0) = & u(2\pi) \end{cases} \tag{5.5}$$

Discretizando con $x_j = jh$, $h = 2\pi/(m+1)$:

$$\frac{1}{h^2}\left(u_{j-1} - 2u_j + u_{j+1}\right) = f(x_j), \quad j = 0, 1, \ldots, m,$$

con $u_{-1} = u_m$ y $u_{m+1} = u_0$ (periodicidad).

Lema 5.26 (Modos de Fourier discretos como autofunciones)**.** *Considerando* $W_j(\xi) = e^{i\xi x_j} = e^{i\xi jh}$*, tenemos:*

$$\delta_{xx}(W(\xi))_j = \frac{1}{h^2}\left(W_{j+1} - 2W_j + W_{j-1}\right) = -\frac{4}{h^2}\sin^2\left(\frac{h\xi}{2}\right)W_j.$$

Demostración.

$$\begin{aligned} \delta_{xx}(W)_j &= \frac{1}{h^2}\left(e^{i(j+1)h\xi} - 2e^{ijh\xi} + e^{i(j-1)h\xi}\right) \\ &= \frac{1}{h^2}\left(e^{ih\xi} - 2 + e^{-ih\xi}\right)e^{ijh\xi} \\ &= \frac{2}{h^2}\left(\cos(h\xi) - 1\right)e^{ijh\xi} \\ &= -\frac{4}{h^2}\sin^2\left(\frac{h\xi}{2}\right)W_j, \end{aligned}$$

usando $\cos(2x) = 1 - 2\sin^2(x)$. □

Observación 5.27. *El autovalor discreto* $\lambda(\xi) = -\frac{4}{h^2}\sin^2\left(\frac{h\xi}{2}\right)$ *aproxima el autovalor continuo* $-\xi^2$ *con error* $O(h^2)$*:*

$$-\frac{4}{h^2}\sin^2\left(\frac{h\xi}{2}\right) = -\frac{4}{h^2}\cdot\frac{h^2\xi^2}{4} + O(h^4) = -\xi^2 + O(h^2).$$

Lema 5.28 (Ortonormalidad de modos de Fourier discretos)**.** *Los modos* $W(k) = \left\{\frac{e^{kijh}}{\sqrt{2\pi}}\right\}_{j=0}^{m}$ *con* $h = 2\pi/(m+1)$*,* $k \in \mathbb{Z}$*, son ortonormales respecto al producto interno discreto:*

$$(v, w) = h\sum_{j=0}^{m} v_j\overline{w}_j.$$

Demostración. Basta probar que para $k \neq 0$:

$$h\sum_{j=0}^{m} e^{kijh} = 0.$$

Usando la suma geométrica:

$$h\sum_{j=0}^{m}\left(e^{ikh}\right)^j = h\frac{1-e^{ikh(m+1)}}{1-e^{ikh}} = h\frac{1-e^{2\pi ik}}{1-e^{ikh}} = 0,$$

ya que $e^{2\pi ik} = 1$ para $k \in \mathbb{Z}$ y $e^{ikh} \neq 1$ para $|k| \leq m$. □

Corolario 5.29. *La matriz del sistema periódico se diagonaliza en una base ortonormal (los modos de Fourier discretos). Por tanto,* $\|(A^h)^{-1}\|_2 = 1/|\lambda_{\min}|$*, donde:*

$$\lambda_{\min} = -\frac{4}{h^2}\sin^2\left(\frac{\pi h}{2}\right) \sim -\pi^2 + O(h^2).$$

La discretización es estable con $\|(A^h)^{-1}\|_2 \leq 1/\pi^2 + O(h^2)$*.*

5.3.4 Caso general: término de reacción

Consideremos el problema más general:

$$\begin{cases} -u_{xx}(x) + q(x)u(x) = & f(x), \quad \text{para } x \in (a,b) \\ u(a) = & \alpha \\ u(b) = & \beta, \end{cases} \tag{5.6}$$

con $q(x) \geq 0$.

La discretización conduce al sistema:

$$A^h u^h = F^h,$$

donde:

$$A^h = \frac{1}{h^2}\begin{pmatrix} 2+h^2q_1 & -1 & 0 & \cdots & 0 \\ -1 & 2+h^2q_2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2+h^2q_{m-1} & -1 \\ 0 & \cdots & 0 & -1 & 2+h^2q_m \end{pmatrix}.$$

Teorema 5.30 (Dominancia diagonal estricta)**.** *Si $q(x) \geq q_0 > 0$ para todo $x \in (a,b)$, entonces A^h es estrictamente diagonalmente dominante por filas:*

$$|a_{jj}| = \frac{2+h^2 q_j}{h^2} > \frac{2}{h^2} = \sum_{k \neq j} |a_{jk}|.$$

Por el teorema de Gershgorin, A^h es no singular. Más aún, existe C independiente de h tal que $\|(A^h)^{-1}\|_\infty \leq C$.

Observación 5.31 (Principio de máximo discreto)**.** *Cuando $q(x) \geq 0$, la matriz A^h satisface un **principio de máximo discreto**: si $A^h w \geq 0$ y $w_0, w_{m+1} \geq 0$, entonces $w_j \geq 0$ para todo $j = 1, \ldots, m$. Esto generaliza el principio de máximo para la ecuación diferencial y permite demostrar estabilidad en norma infinito:*

$$\|(A^h)^{-1}\|_\infty \leq C(b-a)^2,$$

donde C depende solo de $\|q\|_\infty$.

Problemas en dimensiones superiores. Para la ecuación de Poisson 2D:

$$-\Delta u = f \quad \text{en } \Omega, \quad u = g \quad \text{en } \partial\Omega,$$

discretizamos en una malla (x_i, y_j) con $x_i = ih$, $y_j = jh$. La aproximación del Laplaciano es:

$$\Delta u_{ij} \approx \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}}{h^2}.$$

Esto da un sistema lineal de tamaño $N^2 \times N^2$ (para malla $N \times N$). La matriz es grande pero **esparsa** (cada fila tiene solo 5 elementos no nulos). Métodos iterativos como Gauss-Seidel, gradiente conjugado, o multigrid son esenciales para resolver sistemas grandes eficientemente.

Ejemplo 5.32 (Problema de Sturm-Liouville discreto)**.** *Considerar el problema de autovalores:*

$$-u''(x) = \lambda u(x), \quad x \in (0,1), \quad u(0) = u(1) = 0.$$

Discretizando con diferencias finitas, obtenemos:

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}.$$

Los autovalores de esta matriz aproximan $\lambda_k = (k\pi)^2$ con error $O(h^2)$. Los autovectores aproximan las autofunciones $\sin(k\pi x)$. Este es el fundamento de métodos espectrales discretos.

5.3.5 Métodos espectrales

El primer paso en la implementación de métodos espectrales es calcular derivadas de funciones representadas por sus coeficientes de Fourier. Si $f(x) = \sum_{k=-N/2}^{N/2} \hat{f}_k e^{ikx}$, entonces su derivada es:

$$f'(x) = \sum_{k=-N/2}^{N/2} ik\hat{f}_k e^{ikx}.$$

En el contexto discreto, dados valores $f_j = f(x_j)$ en puntos de malla equiespaciados:

1. Calcular $\hat{F}[k] = \text{FFT}\{f_j\}$.
2. Multiplicar por ik (análogo a la Ecuación (4.5)): $\hat{F}'[k] = ik \cdot \hat{F}[k]$ (con k apropiadamente definido para la DFT).
3. Calcular $f'_j = \text{IFFT}\{\hat{F}'[k]\}$.

Este método proporciona **precisión espectral**: si f es suave, el error decrece más rápido que cualquier potencia de $1/N$ (decaimiento exponencial). Es la base de los *métodos espectrales* para resolver numéricamente ecuaciones diferenciales.

Para problemas no periódicos en un intervalo $[a, b]$, los polinomios de Chebyshev son la base natural. Tras cambio de variable, trabajamos en $[-1, 1]$.

Derivadas espectrales de Chebyshev. Si $u(x) = \sum_{k=0}^{N} a_k T_k(x)$, su derivada es $u'(x) = \sum_{k=0}^{N} b_k T_k(x)$ donde los coeficientes $\{b_k\}$ se relacionan con $\{a_k\}$ mediante:

$$b_k = \sum_{j=k+1,\, j+k \text{ impar}}^{N} \frac{2ja_j}{c_k},$$

donde $c_0 = 2$, $c_k = 1$ para $k \geq 1$. Esta fórmula se deriva de la relación de recurrencia para derivadas de Chebyshev. En la práctica, dados valores $u_j = u(x_j)$ en los nodos de Chebyshev $x_j = \cos(j\pi/N)$:

1. Calcular coeficientes: $a_k = \text{DCT}\{u_j\}$ (transformada coseno discreta, $O(N \log N)$ vía FFT).
2. Aplicar matriz de derivación en espacio de coeficientes: $b_k = \sum_j D_{kj} a_j$.
3. Transformar de vuelta: $u'_j = \text{IDCT}\{b_k\}$.

Alternativamente, se puede usar la **matriz de derivación** directamente en espacio físico: $u'_j = \sum_{\ell=0}^{N} D_{j\ell} u_\ell$, donde D es densa pero tiene estructura conocida.

Método de colocación espectral. Para resolver un PVC:

$$Lu = f \quad \text{en } (-1, 1), \quad u(-1) = u_a,\, u(1) = u_b,$$

donde L es un operador diferencial, aproximamos $u \approx u_N = \sum_{k=0}^{N} \hat{u}_k T_k$ e imponemos que la ecuación se satisfaga exactamente en los nodos interiores de Chebyshev $\{x_j\}_{j=1}^{N-1}$:

$$(Lu_N)(x_j) = f(x_j), \quad j = 1, \ldots, N-1,$$

más las condiciones de frontera:

$$u_N(-1) = u_a, \quad u_N(1) = u_b.$$

Esto da un sistema algebraico (generalmente denso) de $N+1$ ecuaciones para los $N+1$ coeficientes.

Ejemplo 5.33 (Ecuación de Helmholtz 1D)**.** *Resolver* $-u''(x) - k^2 u(x) = f(x)$ *en* $(-1, 1)$ *con* $u(\pm 1) = 0$*. Discretizando con Chebyshev:*

$$-(D^2 u)_j - k^2 u_j = f_j, \quad j = 1, \ldots, N-1,$$

donde D^2 *es la segunda derivada espectral (obtenida aplicando* D *dos veces o usando fórmula directa). Con las condiciones de frontera, obtenemos un sistema lineal de tamaño* $(N-1) \times (N-1)$*. La matriz es densa pero el sistema se resuelve en* $O(N^2)$ *(directo) o* $O(N)$ *iteraciones con precondicionador.*

Convergencia espectral. Para funciones analíticas en una región que contiene $[-1, 1]$, el error decrece exponencialmente: $\|u - u_N\| = O(\rho^{-N})$ con $\rho > 1$. Para funciones C^m pero no analíticas, el error es $O(N^{-m})$, similar a diferencias finitas de orden alto pero con constantes más favorables.

Ventajas vs diferencias finitas:

- **Precisión superior**: Para funciones suaves, métodos espectrales alcanzan precisión de máquina con N moderado ($N \sim 10$–100), mientras que diferencias finitas requieren $N \sim 10^3$–10^6.
- **Derivadas de alto orden**: Calcular $u^{(k)}$ es directo en métodos espectrales; en diferencias finitas, stencils de alto orden son complejos e inestables.

Desventajas:

- **Matrices densas**: Los sistemas lineales no tienen la estructura de banda de diferencias finitas, aunque hay métodos rápidos (FFT, precondicionadores).
- **Geometrías complejas**: Métodos espectrales son naturales en dominios rectangulares; para geometrías irregulares, se necesitan extensiones (métodos de elementos espectrales).
- **Funciones no suaves**: Si u tiene singularidades o discontinuidades, la convergencia espectral se pierde (fenómeno de Gibbs). En estos casos, diferencias finitas o elementos finitos pueden ser preferibles.

Ejemplo 5.34 (Comparación numérica: $-u'' = \pi^2 \sin(\pi x)$, $u(\pm 1) = 0$)**.** *Solución exacta:* $u(x) = \sin(\pi x)$*. Errores* $\|u - u_N\|_\infty$*:*

N	*Diferencias finitas* ($O(h^2)$)	*Chebyshev espectral*
10	8.2×10^{-3}	1.3×10^{-12}
20	2.1×10^{-3}	4.4×10^{-15}
40	5.1×10^{-4}	$\sim 10^{-16}$ *(redondeo)*

El método espectral alcanza precisión de máquina con $N = 20$, mientras que diferencias finitas requiere $N \gg 100$ para similar precisión.

Extensión a múltiples dimensiones. En 2D, usamos productos tensoriales de polinomios de Chebyshev: $u(x, y) \approx \sum_{j,k} \hat{u}_{jk} T_j(x) T_k(y)$. Las derivadas se calculan aplicando D en cada dirección. Para la ecuación de Poisson $-\Delta u = f$ en el cuadrado $[-1, 1]^2$, el sistema discreto es de tamaño $N^2 \times N^2$ y se resuelve con métodos iterativos, usando FFT multidimensional para aplicar operadores rápidamente.

En resumen, los métodos espectrales son la herramienta de elección cuando la solución es suave y la geometría es simple. Su precisión superior (convergencia exponencial vs algebraica) los hace indispensables en simulaciones de fluidos, ondas, y otros problemas donde la resolución fina de estructuras suaves es crítica.

Chapter 6

Optimización Numérica

La optimización es un problema central en matemática aplicada, ciencia de datos y aprendizaje automático: dado un funcional $f : \mathbb{R}^n \to \mathbb{R}$, buscamos:

$$\min_{x \in \mathbb{R}^n} f(x).$$

A diferencia de la resolución de sistemas lineales, donde existe una solución exacta alcanzable algorítmicamente, en optimización típicamente solo podemos aproximarnos a un óptimo mediante iteración. Este capítulo desarrolla los algoritmos fundamentales de optimización, comenzando con métodos de primer orden (basados en gradientes) y progresando hacia métodos de segundo orden (basados en la aproximación cuadrática local).

6.1 Descenso por gradiente

El método de descenso por gradiente es el algoritmo más fundamental de optimización. Su idea es simple: moverse en la dirección de mayor decrecimiento de la función.

6.1.1 Motivación geométrica

Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función diferenciable. En un punto x, el gradiente $\nabla f(x)$ apunta en la dirección de mayor crecimiento de f. Por lo tanto, $-\nabla f(x)$ apunta en la dirección de mayor decrecimiento.

Definición 6.1 (Método de descenso por gradiente)**.** *Dado un punto inicial $x^{(0)} \in \mathbb{R}^n$ y una secuencia de tamaños de paso $\{\alpha_k\}_{k=0}^{\infty}$, el **método de descenso por gradiente** genera la secuencia:*

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)}), \quad k = 0, 1, 2, \ldots$$

El parámetro $\alpha_k > 0$ se llama **tamaño de paso** (step size) o **tasa de aprendizaje** (learning rate) en el contexto de aprendizaje automático.

6.1.2 Convergencia con paso constante para funciones fuertemente convexas

Para analizar la convergencia, necesitamos hipótesis sobre f.

Definición 6.2 (Función fuertemente convexa)**.** *Una función $f : \mathbb{R}^n \to \mathbb{R}$ es* ***μ-fuertemente convexa*** *si para todo $x, y \in \mathbb{R}^n$:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2.$$

Equivalentemente, si f es dos veces diferenciable, $\nabla^2 f(x) \succeq \mu I$ para todo x.

Definición 6.3 (Suavidad de Lipschitz)**.** *Una función $f : \mathbb{R}^n \to \mathbb{R}$ es* ***L-suave*** *(o tiene gradiente L-Lipschitz) si:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Equivalentemente, si f es dos veces diferenciable, $\nabla^2 f(x) \preceq LI$ para todo x.

Definición 6.4 (Número de condición)**.** *Para una función μ-fuertemente convexa y L-suave, el* ***número de condición*** *es:*

$$\kappa = \frac{L}{\mu}.$$

Teorema 6.5 (Convergencia de descenso por gradiente con paso constante)**.** *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función μ-fuertemente convexa y L-suave, con único minimizador $x^* = \arg\min f$. Si usamos paso constante $\alpha = \frac{1}{L}$, entonces:*

$$\|x^{(k)} - x^*\|^2 \leq \left(1 - \frac{1}{\kappa}\right)^k \|x^{(0)} - x^*\|^2,$$

donde $\kappa = L/\mu$ es el número de condición.

Demostración. Por la suavidad de f, tenemos la **desigualdad de descenso**:

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}\|y - x\|^2.$$

Aplicando con $y = x^{(k+1)} = x^{(k)} - \alpha \nabla f(x^{(k)})$ y $x = x^{(k)}$:

$$\begin{aligned} f(x^{(k+1)}) &\leq f(x^{(k)}) - \alpha\|\nabla f(x^{(k)})\|^2 + \frac{L\alpha^2}{2}\|\nabla f(x^{(k)})\|^2 \\ &= f(x^{(k)}) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla f(x^{(k)})\|^2. \end{aligned}$$

Con $\alpha = 1/L$, obtenemos:

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \frac{1}{2L}\|\nabla f(x^{(k)})\|^2.$$

Por la convexidad fuerte:

$$f(x^{(k)}) - f(x^*) \geq \nabla f(x^{(k)})^T (x^{(k)} - x^*) + \frac{\mu}{2}\|x^{(k)} - x^*\|^2.$$

Como $\nabla f(x^*) = 0$ (optimalidad), tenemos:

$$\nabla f(x^{(k)})^T (x^{(k)} - x^*) \geq f(x^{(k)}) - f(x^*).$$

Combinando estas desigualdades y usando la desigualdad de Cauchy-Schwarz optimizada para funciones fuertemente convexas y suaves, se obtiene:

$$f(x^{(k+1)}) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)(f(x^{(k)}) - f(x^*)).$$

Por inducción:

$$f(x^{(k)}) - f(x^*) \leq \left(1 - \frac{1}{\kappa}\right)^k (f(x^{(0)}) - f(x^*)).$$

Finalmente, usando nuevamente la convexidad fuerte:

$$f(x^{(k)}) - f(x^*) \geq \frac{\mu}{2}\|x^{(k)} - x^*\|^2,$$

se obtiene el resultado. □

Observación 6.6 (Tasa de convergencia lineal)**.** *El teorema muestra que descenso por gradiente converge* ***linealmente*** *con tasa* $(1 - 1/\kappa)$*. El número de iteraciones para reducir el error por un factor de* ϵ *es:*

$$k \approx \kappa \log(1/\epsilon).$$

Si el problema está mal condicionado ($\kappa \gg 1$*), la convergencia es lenta. Este es el análogo de precondicionamiento en sistemas lineales.*

6.1.3 Descenso por gradiente estocástico (SGD)

En aplicaciones de aprendizaje automático, minimizamos funciones de la forma:

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x),$$

donde f_i representa la pérdida en el ejemplo de entrenamiento i-ésimo. Para m grande (millones de ejemplos), calcular el gradiente completo $\nabla f(x) = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(x)$ es prohibitivamente costoso.

Definición 6.7 (Descenso por gradiente estocástico (SGD))**.** *En cada iteración, seleccionamos aleatoriamente un índice* $i_k \in \{1, \ldots, m\}$ *y actualizamos:*

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f_{i_k}(x^{(k)}).$$

El gradiente $\nabla f_{i_k}(x^{(k)})$ es un **estimador no sesgado** de $\nabla f(x^{(k)})$:

$$\mathbb{E}_{i_k}[\nabla f_{i_k}(x^{(k)})] = \nabla f(x^{(k)}).$$

Teorema 6.8 (Convergencia de SGD)**.** *Bajo hipótesis apropiadas (convexidad fuerte, suavidad, y gradientes de varianza acotada), con paso decreciente* $\alpha_k = \frac{c}{k}$*, SGD converge en expectativa:*

$$\mathbb{E}[f(x^{(k)}) - f(x^*)] = O\left(\frac{1}{k}\right).$$

Observación 6.9 (SGD vs GD). • ***Costo por iteración***: *SGD es $O(n)$ (un solo gradiente), GD es $O(mn)$ (suma de m gradientes).*

- ***Convergencia***: *GD tiene convergencia más rápida por iteración, pero SGD hace más iteraciones baratas.*
- ***Práctica***: *Para m grande, SGD con mini-batches (promedio sobre subconjuntos pequeños) es el método de elección.*

6.1.4 Aplicación: Entrenamiento de redes neuronales

Una red neuronal feedforward con L capas es una función compuesta:

$$f(x;\theta) = \sigma_L(W_L\sigma_{L-1}(\cdots\sigma_1(W_1x + b_1)\cdots) + b_L),$$

donde $\theta = \{W_1, b_1, \ldots, W_L, b_L\}$ son los parámetros (pesos y sesgos), y σ_i son funciones de activación no lineales (ReLU, sigmoide, tanh, etc.).

Problema de optimización: Dado un conjunto de entrenamiento $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, minimizamos:

$$\mathcal{L}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell(f(x^{(i)};\theta), y^{(i)}) + \lambda R(\theta),$$

donde:

- ℓ es la función de pérdida (cross-entropy para clasificación, MSE para regresión).
- $R(\theta)$ es un término de regularización (típicamente L^2: $R(\theta) = \|\theta\|^2$).
- $\lambda > 0$ es el parámetro de regularización.

Teorema 6.10 (Backpropagation). *El algoritmo de **backpropagation** calcula el gradiente $\nabla_\theta\mathcal{L}$ eficientemente mediante la regla de la cadena:*

1. ***Forward pass***: *Calcular las activaciones de cada capa propagando x hacia adelante.*
2. ***Backward pass***: *Calcular las derivadas propagando el error hacia atrás usando la regla de la cadena.*

El costo es $O(|\theta|)$, lineal en el número de parámetros.

Entrenamiento con SGD:

1. Inicializar $\theta^{(0)}$ aleatoriamente.
2. Para $k = 0, 1, 2, \ldots$ hasta convergencia:
   (a) Muestrear mini-batch $\mathcal{B}_k \subset \{1, \ldots, m\}$.
   (b) Calcular gradiente: $g_k = \frac{1}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k} \nabla_\theta \ell(f(x^{(i)};\theta^{(k)}), y^{(i)})$.
   (c) Actualizar: $\theta^{(k+1)} = \theta^{(k)} - \alpha_k g_k$.

Observación 6.11 (Variantes modernas de SGD)**.** *En la práctica, se usan variantes de SGD con momentum y adaptación de la tasa de aprendizaje:*

- ***Momentum****:* $v_{k+1} = \beta v_k + g_k$, $\theta^{(k+1)} = \theta^{(k)} - \alpha v_{k+1}$.
- ***Adam****: Combina momentum con escalamiento adaptativo basado en segundos momentos de los gradientes.*
- ***Learning rate scheduling****: Decaer* α_k *durante el entrenamiento (step decay, exponential decay, cosine annealing).*

6.2 Método de Newton para sistemas no lineales

El método de Newton es el algoritmo fundamental para resolver sistemas de ecuaciones no lineales y para optimización de segundo orden.

6.2.1 Método de Newton para sistemas

Consideremos el problema de encontrar $x \in \mathbb{R}^n$ tal que:

$$F(x) = 0,$$

donde $F : \mathbb{R}^n \to \mathbb{R}^n$ es una función diferenciable.

Definición 6.12 (Método de Newton para sistemas)**.** *Dado un punto inicial* $x^{(0)}$*, el método de Newton genera la secuencia:*

$$x^{(k+1)} = x^{(k)} - [J_F(x^{(k)})]^{-1} F(x^{(k)}),$$

donde $J_F(x) = \frac{\partial F}{\partial x}$ *es la matriz Jacobiana de* F*.*

Interpretación geométrica: En cada iteración, aproximamos F por su linealización:

$$F(x) \approx F(x^{(k)}) + J_F(x^{(k)})(x - x^{(k)}),$$

y resolvemos $F(x^{(k)}) + J_F(x^{(k)})\delta = 0$ para obtener $\delta = -[J_F(x^{(k)})]^{-1} F(x^{(k)})$.

Algoritmo (implementación estable):

1. En la iteración k, calcular $F(x^{(k)})$ y $J_F(x^{(k)})$.
2. Resolver el sistema lineal: $J_F(x^{(k)})\delta = -F(x^{(k)})$.
3. Actualizar: $x^{(k+1)} = x^{(k)} + \delta$.
4. Repetir hasta $\|F(x^{(k)})\| < \text{tol}$ o $\|\delta\| < \text{tol}$.

Observación 6.13 (Costo computacional)**.** *Cada iteración requiere:*

- *Evaluación de* F *y* J_F*: Costo depende del problema.*
- *Resolución de sistema lineal* $n \times n$*:* $O(n^3)$ *con factorización* LU*.*
- *Para sistemas grandes esparsos, se usan métodos iterativos (GMRES, etc.).*

6.2.2 Convergencia cuadrática

Teorema 6.14 (Convergencia local cuadrática de Newton)**.** *Sea $F : \mathbb{R}^n \to \mathbb{R}^n$ continuamente diferenciable en una vecindad de una raíz x^* con $F(x^*) = 0$. Supongamos:*

1. *$J_F(x^*)$ es invertible.*
2. *J_F es Lipschitz continua en una vecindad de x^*: $\|J_F(x) - J_F(y)\| \leq L\|x - y\|$.*

Entonces existe $\delta > 0$ tal que si $\|x^{(0)} - x^*\| < \delta$, la secuencia de Newton converge a x^* con:

$$\|x^{(k+1)} - x^*\| \leq C\|x^{(k)} - x^*\|^2,$$

donde $C = \frac{L}{2}\|[J_F(x^*)]^{-1}\|$.

Demostración. Por la fórmula de Taylor de orden 2 para funciones vectoriales:

$$F(x^*) = F(x^{(k)}) + J_F(x^{(k)})(x^* - x^{(k)}) + \int_0^1 (J_F(x^{(k)} + t(x^* - x^{(k)})) - J_F(x^{(k)}))(x^* - x^{(k)})dt.$$

Como $F(x^*) = 0$:

$$J_F(x^{(k)})(x^* - x^{(k)}) = -F(x^{(k)}) - \int_0^1 (J_F(x^{(k)} + t(x^* - x^{(k)})) - J_F(x^{(k)}))(x^* - x^{(k)})dt.$$

Multiplicando por $[J_F(x^{(k)})]^{-1}$:

$$x^* - x^{(k)} = -[J_F(x^{(k)})]^{-1}F(x^{(k)}) - [J_F(x^{(k)})]^{-1}R_k,$$

donde R_k es el término integral.

Por la definición de la iteración de Newton:

$$x^* - x^{(k+1)} = -[J_F(x^{(k)})]^{-1}R_k.$$

Acotando el término de error usando la condición Lipschitz de J_F:

$$\|R_k\| \leq \int_0^1 \|J_F(x^{(k)} + t(x^* - x^{(k)})) - J_F(x^{(k)})\| \cdot \|x^* - x^{(k)}\|dt \leq \frac{L}{2}\|x^* - x^{(k)}\|^2.$$

Por lo tanto:

$$\|x^{(k+1)} - x^*\| \leq \|[J_F(x^{(k)})]^{-1}\| \cdot \frac{L}{2}\|x^{(k)} - x^*\|^2.$$

En una vecindad suficientemente pequeña de x^*, $\|[J_F(x^{(k)})]^{-1}\|$ está acotado por $2\|[J_F(x^*)]^{-1}\|$, lo que completa la demostración. □

Observación 6.15 (Interpretación de convergencia cuadrática)**.** *Convergencia cuadrática significa que el número de dígitos correctos se duplica en cada iteración. Si $\|x^{(k)} - x^*\| \approx 10^{-d}$, entonces $\|x^{(k+1)} - x^*\| \approx 10^{-2d}$.*

***Ejemplo**: Partiendo de 1 dígito correcto, obtenemos aproximadamente:*

- *Iteración 1: 2 dígitos*
- *Iteración 2: 4 dígitos*
- *Iteración 3: 8 dígitos*
- *Iteración 4: 16 dígitos (precisión de máquina)*

6.2.3 Método de Newton para optimización

Para el problema de optimización $\min f(x)$, buscamos puntos críticos donde $\nabla f(x) = 0$. Aplicando el método de Newton a $F(x) = \nabla f(x)$:

$$x^{(k+1)} = x^{(k)} - [\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)}),$$

donde $\nabla^2 f$ es la matriz Hessiana.

Teorema 6.16 (Convergencia cuadrática para optimización)**.** *Si f es dos veces continuamente diferenciable, x^* es un minimizador local estricto (con $\nabla^2 f(x^*) \succ 0$), y $x^{(0)}$ está suficientemente cerca de x^*, entonces el método de Newton converge cuadráticamente a x^*.*

Observación 6.17 (Comparación: Newton vs Gradiente)**.**

- ***Convergencia**: Newton es cuadrático, gradiente es lineal.*
- ***Costo por iteración**: Newton requiere $O(n^3)$ (resolver sistema con Hessiana), gradiente requiere $O(n)$.*
- ***Convergencia global**: Gradiente con line search converge globalmente, Newton requiere estar cerca del óptimo.*
- ***Práctica**: Métodos cuasi-Newton (BFGS) combinan ventajas de ambos.*

6.3 Cuadrados mínimos no lineales

Muchos problemas de ajuste de modelos tienen la forma:

$$\min_{x \in \mathbb{R}^n} \|r(x)\|^2 = \min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} r_i(x)^2,$$

donde $r : \mathbb{R}^n \to \mathbb{R}^m$ es el vector de residuos con $m \geq n$. Esto generaliza cuadrados mínimos lineales a modelos no lineales.

Ejemplo 6.18 (Ajuste de curva no lineal)**.** *Ajustar un modelo exponencial $y = ae^{bx}$ a datos (x_i, y_i):*

$$r_i(a, b) = ae^{bx_i} - y_i, \quad i = 1, \ldots, m.$$

6.3.1 Método de Gauss-Newton

El método de Gauss-Newton es una especialización del método de Newton que explota la estructura de suma de cuadrados.

Sea $f(x) = \frac{1}{2}\|r(x)\|^2 = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2$. Entonces:

$$\nabla f(x) = J_r(x)^T r(x),$$

donde $J_r(x) = \frac{\partial r}{\partial x}$ es la matriz Jacobiana de r (dimensión $m \times n$).

La Hessiana es:

$$\nabla^2 f(x) = J_r(x)^T J_r(x) + \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x).$$

Definición 6.19 (Método de Gauss-Newton)**.** *El método de Gauss-Newton aproxima la Hessiana ignorando el segundo término (términos de segundo orden en los residuos):*

$$\nabla^2 f(x) \approx J_r(x)^T J_r(x).$$

La iteración es:

$$x^{(k+1)} = x^{(k)} - [J_r(x^{(k)})^T J_r(x^{(k)})]^{-1} J_r(x^{(k)})^T r(x^{(k)}).$$

Implementación estable: En lugar de formar $J^T J$ explícitamente (que puede estar mal condicionada), se resuelve:

$$J_r(x^{(k)})^T J_r(x^{(k)})\delta = -J_r(x^{(k)})^T r(x^{(k)}),$$

equivalentemente, el problema de cuadrados mínimos:

$$\min_{\delta} \|J_r(x^{(k)})\delta + r(x^{(k)})\|^2.$$

Esto se resuelve con factorización QR de $J_r(x^{(k)})$.

Teorema 6.20 (Convergencia de Gauss-Newton)**.** *Si los residuos en el óptimo son pequeños ($\|r(x^*)\| \approx 0$), el método de Gauss-Newton converge cuadráticamente. Si los residuos son grandes, la convergencia es solo lineal (comparable al descenso por gradiente).*

6.3.2 Método de Levenberg-Marquardt

El método de Gauss-Newton puede fallar si $J_r(x^{(k)})$ no tiene rango completo o si estamos lejos del óptimo. El método de Levenberg-Marquardt añade regularización para garantizar convergencia global.

Definición 6.21 (Método de Levenberg-Marquardt)**.** *En lugar de resolver:*

$$\min_{\delta} \|J_r(x^{(k)})\delta + r(x^{(k)})\|^2,$$

se resuelve el problema regularizado:

$$\min_{\delta} \|J_r(x^{(k)})\delta + r(x^{(k)})\|^2 + \lambda_k \|\delta\|^2,$$

donde $\lambda_k > 0$ es el ***parámetro de amortiguamiento*** *(damping parameter).*

Equivalentemente, se resuelve el sistema de ecuaciones normales:

$$(J_r(x^{(k)})^T J_r(x^{(k)}) + \lambda_k I)\delta = -J_r(x^{(k)})^T r(x^{(k)}).$$

Teorema 6.22 (Estrategia adaptativa para λ_k)**.** *El parámetro λ_k se ajusta dinámicamente:*

- *Si la iteración reduce f: Aceptar paso, reducir $\lambda_{k+1} = \lambda_k/\beta$ (típicamente $\beta = 10$).*
- *Si la iteración aumenta f: Rechazar paso, aumentar $\lambda_k \leftarrow \lambda_k \cdot \beta$ y reintentar.*

Observación 6.23 (Interpolación entre Gauss-Newton y Gradiente)**.** *El método de Levenberg-Marquardt interpola entre dos extremos:*

- $\lambda_k \to 0$*: Gauss-Newton (convergencia rápida cerca del óptimo).*
- $\lambda_k \to \infty$*: Descenso por gradiente con paso pequeño (convergencia global robusta).*

Esta es la razón por la que Levenberg-Marquardt es uno de los algoritmos más usados en la práctica para problemas de cuadrados mínimos no lineales.

Ejemplo 6.24 (Ajuste de modelo logístico)**.** *Consideremos el modelo logístico de crecimiento poblacional:*

$$P(t) = \frac{K}{1 + Ae^{-rt}},$$

con parámetros $\theta = (K, A, r)$ *a estimar a partir de datos* (t_i, P_i)*,* $i = 1, \ldots, m$*.*

Los residuos son:

$$r_i(\theta) = P(t_i) - P_i = \frac{K}{1 + Ae^{-rt_i}} - P_i.$$

La Jacobiana es:

$$\frac{\partial r_i}{\partial K} = \frac{1}{1 + Ae^{-rt_i}}, \quad \frac{\partial r_i}{\partial A} = \frac{-Ke^{-rt_i}}{(1 + Ae^{-rt_i})^2}, \quad \frac{\partial r_i}{\partial r} = \frac{KAt_ie^{-rt_i}}{(1 + Ae^{-rt_i})^2}.$$

Levenberg-Marquardt itera refinando θ *hasta convergencia.*

6.3.3 Cuadrados mínimos reponderados iterativamente (IRLS)

El método IRLS resuelve problemas de regresión robusta minimizando normas más generales que la norma L^2.

Definición 6.25 (Problema de cuadrados mínimos ponderados)**.** *Minimizar:*

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} w_i r_i(x)^2,$$

donde $w_i > 0$ *son pesos que indican la confianza en cada observación.*

Para regresión robusta, queremos minimizar:

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} \rho(r_i(x)),$$

donde ρ es una función de pérdida robusta (por ejemplo, función de Huber, L^1, etc.).

Definición 6.26 (Método IRLS)**.** *El método IRLS resuelve iterativamente problemas de cuadrados mínimos ponderados:*

1. *Inicializar* $x^{(0)}$ *(por ejemplo, con cuadrados mínimos ordinarios).*
2. *Para* $k = 0, 1, 2, \ldots$*:*

(a) Calcular residuos: $r_i^{(k)} = r_i(x^{(k)})$.

(b) Calcular pesos: $w_i^{(k)} = \frac{\rho'(r_i^{(k)})}{r_i^{(k)}}$ *(o una variante apropiada).*

(c) Resolver problema ponderado: $x^{(k+1)} = \arg\min_x \sum_{i=1}^m w_i^{(k)} r_i(x)^2$.

3. Repetir hasta convergencia.

Ejemplo 6.27 (Regresión L^1)**.** *Para minimizar* $\sum_{i=1}^m |r_i(x)|$*, usamos* $\rho(t) = |t|$*, lo que da pesos:*

$$w_i^{(k)} = \frac{1}{|r_i^{(k)}|}.$$

Los residuos grandes reciben pesos pequeños, haciendo el método robusto a outliers.

Teorema 6.28 (Convergencia de IRLS)**.** *Bajo condiciones apropiadas de convexidad de* ρ *y acotamiento de los residuos, IRLS converge a un minimizador del problema de regresión robusta.*

Observación 6.29 (Aplicaciones de IRLS)**.**

- ***Regresión logística****: Clasificación binaria minimizando entropía cruzada.*
- ***Modelos lineales generalizados (GLM)****: Regresión de Poisson, regresión gamma, etc.*
- ***Compresión de señales****: Minimización de normas* L^p *con* $p < 2$ *para promover esparsidad.*
- ***Estadística robusta****: Estimadores M de Huber para datos con outliers.*

Ejemplo 6.30 (Regresión logística mediante IRLS)**.** *En clasificación binaria con etiquetas* $y_i \in \{0, 1\}$*, el modelo logístico es:*

$$P(y = 1|x) = \frac{1}{1 + e^{-\beta^T x}}.$$

Maximizando la verosimilitud (equivalente a minimizar la entropía cruzada) se reduce a IRLS con:

$$w_i = p_i(1 - p_i), \quad p_i = \frac{1}{1 + e^{-\beta^T x_i}}.$$

Este es el algoritmo estándar para entrenar regresión logística.

Appendix A

Espacios de Pre-Hilbert

En este apéndice revisamos conceptos fundamentales de álgebra lineal en espacios con producto interno. Comenzamos con las definiciones geométricas básicas (ortogonalidad, proyección) para luego establecer resultados centrales sobre aproximación y bases ortonormales. Finalmente, aplicamos estas herramientas para demostrar teoremas de factorización matricial esenciales en análisis numérico, como la descomposición de Schur y la diagonalización de matrices normales.

A.1 Definiciones básicas

Definición A.1 (Espacio de pre-Hilbert)**.** *Un* ***espacio de pre-Hilbert*** *(o espacio con producto interno) es un par* $(V, (\cdot,\cdot))$ *donde* V *es un espacio vectorial sobre* $\mathbb{C}$ *(o* $\mathbb{R}$*) y* $(\cdot,\cdot) : V \times V \to \mathbb{C}$ *es un producto interno en* V*.*

Definición A.2 (Producto interno)**.** *Sea* V *un espacio vectorial sobre* $\mathbb{C}$ *(o* $\mathbb{R}$*). Un* ***producto interno*** *en* V *es una función* $(\cdot,\cdot) : V \times V \to \mathbb{C}$ *que satisface:*

1. ***Linealidad en el primer argumento****:* $(\alpha u + \beta v, w) = \alpha(u, w) + \beta(v, w)$ *para todos* $u, v, w \in V$ *y* $\alpha, \beta \in \mathbb{C}$*.*
2. ***Simetría hermítica****:* $(u, v) = \overline{(v, u)}$ *para todos* $u, v \in V$*.*
3. ***Positividad****:* $(u, u) \geq 0$ *para todo* $u \in V$*, con igualdad si y solo si* $u = 0$*.*

Observación A.3. *Para espacios reales, la condición de simetría hermítica se reduce a la simetría ordinaria:* $(u, v) = (v, u)$*. La linealidad en el primer argumento y la simetría hermítica implican antilinealidad (o sesquilinealidad) en el segundo argumento:* $(u, \alpha v + \beta w) = \overline{\alpha}(u, v) + \overline{\beta}(u, w)$*.*

Definición A.4 (Norma inducida)**.** *El producto interno induce una* ***norma*** *en* V *definida por:*

$$\|u\| = \sqrt{(u, u)}.$$

Esta norma satisface:

1. $\|u\| \geq 0$ *y* $\|u\| = 0 \Leftrightarrow u = 0$*,*

2. $\|\alpha u\| = |\alpha|\|u\|$ *para todo* $\alpha \in \mathbb{C}$,

3. $\|u + v\| \leq \|u\| + \|v\|$ *(desigualdad triangular).*

Teorema A.5 (Desigualdad de Cauchy-Schwarz)**.** *Para todos* $u, v \in V$*:*

$$|(u, v)| \leq \|u\|\|v\|,$$

con igualdad si y solo si u *y* v *son linealmente dependientes.*

Demostración. Si $v = 0$, el resultado es trivial. Supongamos $v \neq 0$. Para cualquier $t \in \mathbb{C}$, tenemos:

$$0 \leq \|u - tv\|^2 = (u - tv, u - tv) = \|u\|^2 - t(v, u) - \bar{t}(u, v) + |t|^2\|v\|^2.$$

Elegimos $t = (u, v)/\|v\|^2$, entonces:

$$0 \leq \|u\|^2 - \frac{|(u, v)|^2}{\|v\|^2} - \frac{|(u, v)|^2}{\|v\|^2} + \frac{|(u, v)|^2}{\|v\|^4} \cdot \|v\|^2 = \|u\|^2 - \frac{|(u, v)|^2}{\|v\|^2}.$$

Multiplicando por $\|v\|^2$, obtenemos $|(u, v)|^2 \leq \|u\|^2\|v\|^2$. La igualdad se da si y solo si $u = tv$, es decir, si son linealmente dependientes. □

A.2 Ortogonalidad y Proyección

Definición A.6 (Ortogonalidad)**.** *Dos vectores* $u, v \in V$ *son* ***ortogonales*** *si* $(u, v) = 0$*, y escribimos* $u \perp v$*.*

Un conjunto $\{v_k\}_{k \in I}$ *de vectores es* ***ortogonal*** *si* $(v_j, v_k) = 0$ *para todo* $j \neq k$*.*

Un conjunto $\{e_k\}_{k \in I}$ *es* ***ortonormal*** *si es ortogonal y además* $\|e_k\| = 1$ *para todo* $k \in I$*.*

Teorema A.7 (Teorema de Pitágoras)**.** *Si* $u \perp v$*, entonces:*

$$\|u + v\|^2 = \|u\|^2 + \|v\|^2.$$

Más generalmente, si $\{v_1, \ldots, v_n\}$ *es un conjunto ortogonal, entonces:*

$$\left\| \sum_{k=1}^{n} v_k \right\|^2 = \sum_{k=1}^{n} \|v_k\|^2.$$

Demostración. Si $u \perp v$, entonces $(u, v) = 0$. Expandiendo:

$$\|u + v\|^2 = (u + v, u + v) = (u, u) + (u, v) + (v, u) + (v, v) = \|u\|^2 + 0 + 0 + \|v\|^2 = \|u\|^2 + \|v\|^2.$$

El caso general se demuestra por inducción. □

Teorema A.8 (Proyección sobre una base ortogonal)**.** *Sea* $\{v_1, \ldots, v_n\}$ *una base ortogonal de un espacio vectorial* V *de dimensión finita. Entonces, todo vector* $u \in V$ *se puede escribir de manera única como:*

$$u = \sum_{k=1}^{n} c_k v_k,$$

donde los coeficientes están dados por la ***fórmula de proyección****:*

$$c_k = \frac{(u, v_k)}{(v_k, v_k)} = \frac{(u, v_k)}{\|v_k\|^2}. \tag{A.1}$$

Demostración. Supongamos que $u = \sum_{j=1}^{n} c_j v_j$. Tomando el producto interno con v_k:

$$(u, v_k) = \left(\sum_{j=1}^{n} c_j v_j, v_k \right) = \sum_{j=1}^{n} c_j (v_j, v_k).$$

Por ortogonalidad, $(v_j, v_k) = 0$ si $j \neq k$ y $(v_k, v_k) = \|v_k\|^2$. Por lo tanto:

$$(u, v_k) = c_k \|v_k\|^2,$$

de donde se obtiene la fórmula para c_k. □

Teorema A.9 (Proceso de Gram-Schmidt)**.** *Todo espacio vectorial de dimensión finita con producto interno posee una base ortonormal. A partir de una base cualquiera $\{u_1, \ldots, u_n\}$, se puede construir una base ortonormal $\{e_1, \ldots, e_n\}$ mediante la recursión:*

$$v_k = u_k - \sum_{j=1}^{k-1} (u_k, e_j) e_j, \quad e_k = \frac{v_k}{\|v_k\|}.$$

Teorema A.10 (Identidad de Parseval para bases ortonormales finitas)**.** *Sea $\{e_1, \ldots, e_n\}$ una base ortonormal de V. Si $u = \sum_{k=1}^{n} c_k e_k$ con $c_k = (u, e_k)$, entonces:*

$$\|u\|^2 = \sum_{k=1}^{n} |c_k|^2. \tag{A.2}$$

Demostración.

$$\|u\|^2 = (u, u) = \left(\sum_{j=1}^{n} c_j e_j, \sum_{k=1}^{n} c_k e_k \right) = \sum_{j=1}^{n} \sum_{k=1}^{n} c_j \overline{c_k} (e_j, e_k).$$

Por ortonormalidad, $(e_j, e_k) = \delta_{jk}$ (delta de Kronecker), así que:

$$\|u\|^2 = \sum_{k=1}^{n} c_k \overline{c_k} = \sum_{k=1}^{n} |c_k|^2.$$

□

Teorema A.11 (Teorema de la Mejor Aproximación en Subespacios Finitos)**.** *Sea V un espacio de pre-Hilbert y S un subespacio de* ***dimensión finita*** *de V. Para cualquier $u \in V$, existe un* ***único*** *vector $s_0 \in S$ tal que:*

$$\|u - s_0\| = \inf_{s \in S} \|u - s\|.$$

Este vector s_0 se llama la ***mejor aproximación*** *de u en S, o la* ***proyección ortogonal*** *$P_S(u)$.*

Demostración. La demostración se basa en tomar una base ortonormal $\{e_1, \ldots, e_n\}$ para S y definir explícitamente

$$s_0 = \sum_{k=1}^{n} (u, e_k) e_k.$$

Para cualquier otro $s \in S$, podemos escribir $s = \sum_{k=1}^{n} d_k e_k$ con coeficientes d_k. Entonces:

$$\|u - s\|^2 = \|(u - s_0) + (s_0 - s)\|^2 .$$

Como $(u - s_0) \perp (s_0 - s)$ (el vector $(u - s_0)$ es ortogonal a todo S), por el teorema de Pitágoras:

$$\|u - s\|^2 = \|u - s_0\|^2 + \|s_0 - s\|^2 \geq \|u - s_0\|^2,$$

con igualdad si y solo si $s = s_0$. □

Corolario A.12 (Caracterización Ortogonal)**.** *La mejor aproximación $s_0 = P_S(u)$ es el único vector en S que satisface:*

$$(u - s_0, s) = 0 \quad \textit{para todo } s \in S.$$

A.3 Factorización de Schur y Teorema Espectral

En espacios de dimensión finita, la existencia de bases ortonormales permite simplificar la representación matricial de operadores lineales.

Teorema A.13 (Factorización de Schur)**.** *Sea $A \in \mathbb{C}^{n \times n}$. Existe una matriz unitaria $U \in \mathbb{C}^{n \times n}$ (es decir, $U^*U = I$) y una matriz triangular superior $T \in \mathbb{C}^{n \times n}$ tales que:*

$$A = UTU^*.$$

Los elementos diagonales de T son los valores propios de A.

Demostración. Procedemos por inducción sobre la dimensión n. Para $n = 1$ es trivial. Supongamos que el teorema es cierto para matrices de tamaño $n - 1$. Sea λ_1 un valor propio de A y v_1 un vector propio asociado con $\|v_1\| = 1$. Completamos v_1 a una base ortonormal de $\mathbb{C}^n$, digamos $\{v_1, v_2, \ldots, v_n\}$. Sea $U_1 = [v_1, v_2, \ldots, v_n]$ la matriz unitaria formada por estos vectores. Entonces:

$$U_1^* A U_1 = \begin{pmatrix} v_1^* \\ \vdots \\ v_n^* \end{pmatrix} A \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} = \begin{pmatrix} v_1^* A v_1 & v_1^* A v_2 & \cdots \\ v_2^* A v_1 & v_2^* A v_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Como $Av_1 = \lambda_1 v_1$, la primera columna de AU_1 es $\lambda_1 v_1$. Por ortonormalidad, $v_1^*(\lambda_1 v_1) = \lambda_1$ y $v_k^*(\lambda_1 v_1) = 0$ para $k > 1$. Así, la matriz toma la forma:

$$U_1^* A U_1 = \begin{pmatrix} \lambda_1 & w^* \\ 0 & A_2 \end{pmatrix},$$

donde $A_2 \in \mathbb{C}^{(n-1)\times(n-1)}$. Por hipótesis de inducción, existe $V_2 \in \mathbb{C}^{(n-1)\times(n-1)}$ unitaria tal que $V_2^* A_2 V_2 = T_2$ es triangular superior. Definimos $U = U_1 \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix}$. Esta matriz es unitaria y transforma A en una triangular superior. □

Lema A.14 (Matrices triangulares normales)**.** *Una matriz triangular superior $T \in \mathbb{C}^{n\times n}$ es normal si y solo si es diagonal.*

Demostración. ($\Leftarrow$) Si T es diagonal, entonces T^* también es diagonal, y como las matrices diagonales conmutan entre sí, $TT^* = T^*T$.

($\Rightarrow$) Supongamos que T es triangular superior y normal. Escribimos $T = (t_{ij})$ con $t_{ij} = 0$ para $i > j$. Comparemos los elementos diagonales de TT^* y T^*T:

$$(TT^*)_{ii} = \sum_{k=1}^{n} t_{ik}\overline{t_{ik}} = \sum_{k=i}^{n} |t_{ik}|^2 \quad (\text{pues } t_{ik} = 0 \text{ para } k < i).$$

$$(T^*T)_{ii} = \sum_{k=1}^{n} \overline{t_{ki}}t_{ki} = \sum_{k=1}^{i} |t_{ki}|^2 \quad (\text{pues } t_{ki} = 0 \text{ para } k > i).$$

Como T es normal, $(TT^*)_{ii} = (T^*T)_{ii}$:

$$\sum_{k=i}^{n} |t_{ik}|^2 = \sum_{k=1}^{i} |t_{ki}|^2.$$

Para $i = 1$:

$$\sum_{k=1}^{n} |t_{1k}|^2 = |t_{11}|^2.$$

Esto implica que $\sum_{k=2}^{n} |t_{1k}|^2 = 0$, por lo que $t_{1k} = 0$ para $k > 1$.

Para $i = 2$, usando que $t_{12} = 0$:

$$\sum_{k=2}^{n} |t_{2k}|^2 = |t_{12}|^2 + |t_{22}|^2 = |t_{22}|^2.$$

Esto implica que $t_{2k} = 0$ para $k > 2$.

Procediendo inductivamente para $i = 3, 4, \ldots, n$, obtenemos que $t_{ik} = 0$ para todo $i < k$. Por lo tanto, T es diagonal. □

Teorema A.15 (Teorema Espectral para Matrices Normales)**.** *Una matriz $A \in \mathbb{C}^{n\times n}$ es **normal** (es decir, $AA^* = A^*A$) si y solo si es diagonalizable unitariamente. Es decir, existe una matriz unitaria U y una matriz diagonal Λ tales que:*

$$A = U\Lambda U^*.$$

Demostración. ($\Leftarrow$) Si $A = U\Lambda U^*$, entonces $A^* = U\Lambda^* U^*$. Como Λ y Λ^* son diagonales, conmutan ($\Lambda\Lambda^* = \Lambda^*\Lambda$). Entonces:

$$AA^* = U\Lambda U^* U\Lambda^* U^* = U\Lambda\Lambda^* U^* = U\Lambda^*\Lambda U^* = U\Lambda^* U^* U\Lambda U^* = A^*A.$$

($\Rightarrow$) Por la factorización de Schur, $A = UTU^*$ con T triangular superior. Como A es normal, T también debe serlo:

$$TT^* = (U^*AU)(U^*A^*U) = U^*AA^*U = U^*A^*AU = (U^*A^*U)(U^*AU) = T^*T.$$

Ahora bien, T es una matriz triangular superior y normal. Por lo tanto, por el lema anterior, T es diagonal, lo que concluye la demostración. □

Corolario A.16 (Ortogonalidad de autovectores de matrices normales)**.** *Si $A \in \mathbb{C}^{n\times n}$ es normal y v_i, v_j son autovectores correspondientes a autovalores distintos $\lambda_i \neq \lambda_j$, entonces v_i y v_j son ortogonales: $v_i^* v_j = 0$.*

Demostración. Consideremos $Av_i = \lambda_i v_i$ y $Av_j = \lambda_j v_j$. Calculamos:

$$\lambda_i(v_i^* v_j) = (Av_i)^* v_j = v_i^* A^* v_j.$$

Por otro lado, como A es normal, A^* también lo es, y comparten autovectores. Más precisamente, si $Av_j = \lambda_j v_j$, entonces $A^* v_j = \bar{\lambda_j} v_j$ (esto se sigue de que $A = U\Lambda U^*$ implica $A^* = U\Lambda^* U^*$). Por lo tanto:

$$v_i^* A^* v_j = v_i^*(\bar{\lambda_j} v_j) = \bar{\lambda_j}(v_i^* v_j).$$

Igualando las dos expresiones:

$$\lambda_i(v_i^* v_j) = \bar{\lambda_j}(v_i^* v_j).$$

Si $\lambda_i \neq \bar{\lambda_j}$ (lo cual es cierto si $\lambda_i \neq \lambda_j$ en el caso hermítico, o en general por la diagonalización unitaria), entonces necesariamente $v_i^* v_j = 0$. □

Corolario A.17 (Matrices Hermíticas)**.** *Si $A \in \mathbb{C}^{n\times n}$ es hermítica ($A = A^*$), entonces:*

1. *A es normal, por lo que admite diagonalización unitaria: $A = U\Lambda U^*$.*
2. *Todos los autovalores de A son **reales**: $\lambda_i \in \mathbb{R}$ para todo i.*

Demostración. (1) Es inmediato que $AA^* = AA = A^*A$.

(2) Sea λ un autovalor con autovector v: $Av = \lambda v$. Tomando el producto hermítico:

$$v^* Av = v^*(\lambda v) = \lambda v^* v = \lambda \|v\|^2.$$

Por otro lado, como $A^* = A$:

$$v^* Av = (Av)^* v = (\lambda v)^* v = \bar{\lambda} v^* v = \bar{\lambda}\|v\|^2.$$

Igualando, $\lambda = \bar{\lambda}$, por lo que $\lambda \in \mathbb{R}$. □

Teorema A.18 (Caracterización Variacional de Autovalores (Courant-Fischer))**.** *Sea $A \in \mathbb{C}^{n\times n}$ una matriz hermítica con autovalores $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Entonces:*

$$\lambda_k = \min_{S\subset\mathbb{C}^n, \dim(S)=k} \max_{x\in S, x\neq 0} \frac{x^* Ax}{x^* x} = \max_{S\subset\mathbb{C}^n, \dim(S)=n-k+1} \min_{x\in S, x\neq 0} \frac{x^* Ax}{x^* x}.$$

En particular, el autovalor máximo y mínimo están dados por el cociente de Rayleigh:

$$\lambda_1 = \min_{x\neq 0} \frac{x^* Ax}{x^* x}, \quad \lambda_n = \max_{x\neq 0} \frac{x^* Ax}{x^* x}.$$

Demostración. Sea $\{u_1, \ldots, u_n\}$ una base ortonormal de vectores propios de A asociados a los autovalores $\lambda_1 \leq \cdots \leq \lambda_n$. Para cualquier vector $x \neq 0$, podemos escribir $x = \sum_{i=1}^n c_i u_i$. El cociente de Rayleigh es:

$$R(x) = \frac{x^* Ax}{x^* x} = \frac{\sum_{i=1}^n \lambda_i |c_i|^2}{\sum_{i=1}^n |c_i|^2}.$$

Demostremos la caracterización $\max - \min$ para λ_k:

$$\lambda_k = \max_{S \subset \mathbb{C}^n, \dim(S) = n-k+1} \min_{x \in S, x \neq 0} R(x).$$

Sea $V_k = \text{span}\{u_k, \ldots, u_n\}$. La dimensión de V_k es $n - k + 1$. Para cualquier $x \in V_k$, tenemos $c_1 = \cdots = c_{k-1} = 0$, por lo que:

$$R(x) = \frac{\sum_{i=k}^{n} \lambda_i |c_i|^2}{\sum_{i=k}^{n} |c_i|^2} \geq \lambda_k \frac{\sum_{i=k}^{n} |c_i|^2}{\sum_{i=k}^{n} |c_i|^2} = \lambda_k.$$

Por lo tanto, $\min_{x \in V_k} R(x) \geq \lambda_k$. Esto implica que el máximo sobre todos los subespacios es al menos λ_k.

Ahora sea S cualquier subespacio de dimensión $n - k + 1$. Consideremos el subespacio $Z_k = \text{span}\{u_1, \ldots, u_k\}$, que tiene dimensión k. La suma de dimensiones es $\dim(S) + \dim(Z_k) = (n - k + 1) + k = n + 1$. Como la dimensión del espacio total es n, la intersección $S \cap Z_k$ debe tener dimensión al menos 1. Sea $x_0 \in S \cap Z_k$ con $x_0 \neq 0$. Como $x_0 \in Z_k$, es combinación lineal de $u_1, \ldots, u_k$, así que:

$$R(x_0) = \frac{\sum_{i=1}^{k} \lambda_i |c_i|^2}{\sum_{i=1}^{k} |c_i|^2} \leq \lambda_k.$$

Por lo tanto, $\min_{x \in S} R(x) \leq R(x_0) \leq \lambda_k$. Como esto vale para todo S, el máximo de los mínimos es $\leq \lambda_k$. Concluimos que el valor es exactamente λ_k. La otra igualdad se prueba de manera análoga. $\square$

Appendix B

Espacios de Hilbert

B.1 Definiciones y Ejemplos

Definición B.1 (Espacio de Hilbert). *Un espacio de pre-Hilbert $(H, (\cdot,\cdot))$ se dice* ***espacio de Hilbert*** *si es completo respecto a la norma inducida $\|u\| = \sqrt{(u,u)}$. Es decir, toda sucesión de Cauchy en H converge a un elemento de H.*

Definición B.2 (Separabilidad). *Un espacio de Hilbert H se dice* ***separable*** *si posee un subconjunto denso que es numerable. Es decir, existe un conjunto $D = \{d_n\}_{n=1}^{\infty} \subset H$ tal que $\overline{D} = H$.*

La separabilidad es una propiedad técnica crucial que garantiza la existencia de bases ortonormales numerables.

Ejemplo B.3 (El espacio ℓ^2). *El espacio de sucesiones de cuadrado sumable ℓ^2 es el prototipo de espacio de Hilbert separable. La base canónica $e_n = (0, \dots, 1, \dots)$ es una base ortonormal.*

Ejemplo B.4 (El espacio L^2). *El espacio $L^2([0, 2\pi])$ es separable. El sistema trigonométrico $\{\frac{1}{\sqrt{2\pi}} e^{ikx}\}_{k\in\mathbb{Z}}$ es una base ortonormal.*

B.2 Bases Ortonormales

En dimensión infinita, la noción de base algebraica (base de Hamel) es menos útil que la de base topológica u ortonormal.

Definición B.5 (Sistema Ortonormal). *Un subconjunto $S = \{e_k\}_{k\in I} \subset H$ es un* ***sistema ortonormal*** *si $(e_j, e_k) = \delta_{jk}$ para todos $j, k \in I$.*

Teorema B.6 (Desigualdad de Bessel). *Sea $\{e_k\}_{k=1}^{\infty}$ un sistema ortonormal en H. Para cualquier $u \in H$, se cumple:*

$$\sum_{k=1}^{\infty} |(u, e_k)|^2 \leq \|u\|^2.$$

Definición B.7 (Base Ortonormal). *Un sistema ortonormal $\{e_k\}_{k=1}^{\infty}$ en un espacio de Hilbert H se llama* ***base ortonormal*** *(o base de Hilbert) si el subespacio generado por $\{e_k\}$ es denso en H.*

El siguiente teorema conecta la separabilidad con la existencia de bases.

Teorema B.8 (Existencia de Bases Ortonormales)**.** *Sea H un espacio de Hilbert de dimensión infinita. Entonces H es separable si y solo si posee una base ortonormal numerable.*

Demostración. ($\Leftarrow$) Si $\{e_n\}$ es una base ortonormal numerable, el conjunto de combinaciones lineales finitas con coeficientes racionales (en $\mathbb{Q} + i\mathbb{Q}$) es denso en H y es numerable. ($\Rightarrow$) Sea $D = \{d_n\}_{n=1}^{\infty}$ un conjunto denso numerable. Aplicando el proceso de Gram-Schmidt a la sucesión $\{d_n\}$ (descartando los linealmente dependientes), obtenemos un sistema ortonormal $\{e_n\}$. El espacio generado por $\{e_n\}$ coincide con el generado por $\{d_n\}$, que es denso en H. Por tanto, $\{e_n\}$ es una base ortonormal. □

Teorema B.9 (Caracterizaciones de Bases Ortonormales)**.** *Sea $\{e_k\}_{k=1}^{\infty}$ un sistema ortonormal en H. Las siguientes afirmaciones son equivalentes:*

1. *$\{e_k\}$ es una base ortonormal (es completo).*
2. *(Identidad de Parseval) Para todo $u \in H$, $\|u\|^2 = \sum_{k=1}^{\infty} |(u, e_k)|^2$.*
3. *Para todo $u \in H$, $u = \sum_{k=1}^{\infty} (u, e_k) e_k$ (convergencia en norma).*
4. *Si $(u, e_k) = 0$ para todo k, entonces $u = 0$.*

B.3 Teoremas de Proyección y Representación de Riesz

Una de las propiedades geométricas más importantes de los espacios de Hilbert es la posibilidad de proyectar ortogonalmente sobre subespacios cerrados.

Teorema B.10 (Teorema de la Proyección)**.** *Sea H un espacio de Hilbert y M un subespacio cerrado de H. Entonces, para todo $x \in H$, existen únicos vectores $m \in M$ y $z \in M^{\perp}$ tales que*

$$x = m + z.$$

Además, $\|x - m\| = \inf_{y \in M} \|x - y\|$, es decir, m es la mejor aproximación de x en M.

Demostración. Sea $d = \inf_{y \in M} \|x - y\|$. Existe una sucesión $\{y_n\} \subset M$ tal que $\|x - y_n\| \to d$. Usando la identidad del paralelogramo:

$$\|a + b\|^2 + \|a - b\|^2 = 2\|a\|^2 + 2\|b\|^2,$$

aplicada a $a = x - y_n$ y $b = x - y_m$, se puede demostrar que $\{y_n\}$ es una sucesión de Cauchy. Como M es cerrado y H es completo, $\{y_n\}$ converge a un $m \in M$. Para ver que $x - m \in M^{\perp}$, sea $y \in M$ arbitrario. La función $f(t) = \|x - (m + ty)\|^2$ tiene un mínimo en $t = 0$, lo que implica que la derivada en 0 es nula, resultando en $\mathrm{Re}(x - m, y) = 0$. Considerando iy, se obtiene la parte imaginaria. La unicidad se sigue de que $M \cap M^{\perp} = \{0\}$. □

Este teorema permite identificar el espacio dual de un espacio de Hilbert con el espacio mismo.

Teorema B.11 (Teorema de Representación de Riesz)**.** *Sea H un espacio de Hilbert y sea $L : H \to \mathbb{C}$ un funcional lineal continuo (o acotado). Entonces existe un único vector $u \in H$ tal que*

$$L(v) = (v, u) \quad \textit{para todo } v \in H.$$

Además, la norma del funcional coincide con la norma del vector: $\|L\|_{H^*} = \|u\|_H$.

Demostración. Si $L = 0$, tomamos $u = 0$. Si $L \neq 0$, sea $M = \ker(L)$. Como L es continuo, M es cerrado. Por el Teorema de la Proyección, $H = M \oplus M^\perp$. Como $L \neq 0$, $M^\perp \neq \{0\}$. Existe $z \in M^\perp$ con $\|z\| = 1$. Para cualquier $v \in H$, el vector $w = L(v)z - L(z)v$ está en $\ker(L) = M$, pues $L(w) = L(v)L(z) - L(z)L(v) = 0$. Como $z \perp M$, tenemos $(w, z) = 0$.

$$0 = (L(v)z - L(z)v, z) = L(v)(z, z) - L(z)(v, z) = L(v) - (v, \overline{L(z)}z).$$

Despejando $L(v)$, obtenemos $L(v) = (v, u)$ con $u = \overline{L(z)}z$. □

Appendix C

Espacios de Sobolev

C.1 Los espacios de Sobolev periódicos H^s_{per}

En el capítulo anterior vimos cómo la regularidad de una función (ser C^n) se traduce en una cierta velocidad de decaimiento de sus coeficientes de Fourier. Ahora invertimos esta idea: usaremos el decaimiento de los coeficientes para *definir* una nueva noción de regularidad, que es más general y poderosa. Esto nos lleva a los espacios de Sobolev.

En las secciones anteriores definimos coeficientes de Fourier y calculamos su tasa de decaimiento para funciones suaves y periódicas. Sin embargo, las funciones suaves C^k no forman un espacio completo con el producto interno. En esta sección veremos una construcción muy simple de un espacio de funciones completo.

La idea central es que, en lugar de caracterizar funciones por su regularidad clásica (derivadas continuas), podemos caracterizarlas mediante el decaimiento de sus coeficientes de Fourier. Esta perspectiva conduce naturalmente a los **espacios de Sobolev periódicos** H^s_{per}, que son completos y muy útiles tanto en análisis como en cálculo numérico.

Motivados por el teorema de decaimiento de coeficientes, definimos los espacios de Sobolev periódicos mediante condiciones sobre el decaimiento de los coeficientes de Fourier (definidos en la Ecuación (4.2)).

Definición C.1 (Espacios de Sobolev periódicos)**.** *Sea $s \in \mathbb{R}$. El **espacio de Sobolev periódico** H^s_{per} se define como el conjunto de todas las funciones (distribuciones periódicas) f con coeficientes de Fourier $\hat{f}_k$ tales que:*

$$\sum_{k=-\infty}^{\infty} (1+k^2)^s |\hat{f}_k|^2 < \infty. \tag{C.1}$$

Es decir,

$$H^s_{\text{per}} = \left\{ f : [0, 2\pi] \to \mathbb{C} \text{ periódica} \,\middle|\, \sum_{k=-\infty}^{\infty} (1+k^2)^s |\hat{f}_k|^2 < \infty \right\}.$$

El espacio H^s_{per} se dota del **producto interno**:

$$(f, g)_{H^s} = \sum_{k=-\infty}^{\infty} (1+k^2)^s \hat{f}_k \overline{\hat{g}_k}, \tag{C.2}$$

y la **norma** correspondiente:

$$\|f\|_{H^s} = \left(\sum_{k=-\infty}^{\infty} (1+k^2)^s |\hat{f}_k|^2 \right)^{1/2}. \tag{C.3}$$

Observación C.2. *El factor $(1+k^2)^s$ en lugar de k^{2s} evita problemas con el término $k=0$ cuando $s>0$. Algunas referencias usan $(1+|k|)^{2s}$ o simplemente $\langle k\rangle^{2s}$ donde $\langle k\rangle = \sqrt{1+k^2}$.*

Teorema C.3 (Completitud de H^s_{per})**.** *Para cada $s \in \mathbb{R}$, el espacio $(H^s_{\text{per}}, \|\cdot\|_{H^s})$ es un **espacio de Hilbert**, es decir, un espacio vectorial completo con producto interno.*

Demostración. La demostración se basa en la identificación de H^s_{per} con el espacio de sucesiones ℓ^2 con peso. Sea $\{f_n\}$ una sucesión de Cauchy en H^s_{per}. Entonces, para cada $k \in \mathbb{Z}$, la sucesión de coeficientes $\{\widehat{f_n}_k\}$ es de Cauchy en $\mathbb{C}$, ya que:

$$|\widehat{f_n}_k - \widehat{f_m}_k|^2 \leq \frac{1}{(1+k^2)^s} \sum_{j=-\infty}^{\infty} (1+j^2)^s |\widehat{f_n}_j - \widehat{f_m}_j|^2 = \frac{\|f_n - f_m\|^2_{H^s}}{(1+k^2)^s}.$$

Por completitud de $\mathbb{C}$, existe $c_k \in \mathbb{C}$ tal que $\widehat{f_n}_k \to c_k$ para cada k.

Falta verificar que la función f definida por $\hat{f}_k = c_k$ pertenece a H^s_{per} y que $f_n \to f$ en H^s_{per}. Esto se sigue de la convergencia dominada en ℓ^2 con peso. Los detalles técnicos se omiten, pero la idea es estándar en teoría de espacios de sucesiones. □

Teorema C.4 (Caracterización por derivadas débiles)**.** *Para $s = m \in \mathbb{N}$, el espacio H^m_{per} coincide (con normas equivalentes) con el espacio de funciones $f \in L^2([0,2\pi])$ cuyas derivadas débiles hasta orden m pertenecen a $L^2([0,2\pi])$. Es decir:*

$$H^m_{\text{per}} \cong \left\{ f \in L^2([0,2\pi]) : f^{(j)} \in L^2([0,2\pi]) \text{ para } 0 \leq j \leq m \right\}.$$

Además, la norma $\|f\|_{H^m}$ es equivalente a:

$$\|f\|^2_{H^m} \sim \sum_{j=0}^{m} \|f^{(j)}\|^2_{L^2}.$$

Idea de la demostración. La derivación en el espacio de Fourier corresponde a multiplicar los coeficientes por ik, como vimos en la Ecuación (4.5):

$$\widehat{f^{(j)}}_k = (ik)^j \hat{f}_k.$$

Por lo tanto,

$$\sum_{k=-\infty}^{\infty} |\widehat{f^{(j)}}_k|^2 = \sum_{k=-\infty}^{\infty} k^{2j} |\hat{f}_k|^2.$$

Sumando sobre $j = 0, \ldots, m$, obtenemos una cantidad equivalente a $\sum_k (1+k^2)^m |\hat{f}_k|^2$, que es precisamente $\|f\|^2_{H^m}$. □

Teorema C.5 (Inmersiones de Sobolev periódicas)**.** *Sean $s_1, s_2 \in \mathbb{R}$ con $s_1 > s_2$. Entonces:*

1. *$H^{s_1}_{\text{per}} \subset H^{s_2}_{\text{per}}$ (inmersión continua).*
2. *Si $s > 1/2$, entonces $H^s_{\text{per}} \subset C_{\text{per}}([0, 2\pi])$ (funciones continuas), y la inmersión es compacta.*
3. *Más precisamente, si $s > k + 1/2$ con $k \in \mathbb{N}$, entonces $H^s_{\text{per}} \subset C^k_{\text{per}}([0, 2\pi])$.*

Idea de la demostración. (1) Es inmediato de la definición, ya que $(1 + k^2)^{s_2} \leq (1 + k^2)^{s_1}$ para $|k|$ grande.

(2) Para $s > 1/2$, la serie $\sum_k \hat{f}_k e^{ikx}$ converge absolutamente y uniformemente. En efecto, por Cauchy-Schwarz:

$$\sum_{k=-\infty}^{\infty} |\hat{f}_k| = \sum_k (1 + k^2)^{s/2} |\hat{f}_k| \cdot (1 + k^2)^{-s/2} \leq \|f\|_{H^s} \left(\sum_k (1 + k^2)^{-s} \right)^{1/2}.$$

La última serie converge si $s > 1/2$. Por lo tanto, f es continua.

(3) Derivando término a término, si $f \in H^s_{\text{per}}$ con $s > k + 1/2$, entonces $f^{(k)} \in H^{s-k}_{\text{per}} \subset C_{\text{per}}$ (usando el caso anterior). □

Ejemplo C.6. *Consideremos $f(x) = \sum_{k=1}^{\infty} \frac{1}{k^2} \cos(kx)$. Sus coeficientes de Fourier son $\hat{f}_k = \hat{f}_{-k} = \frac{1}{2k^2}$ para $k \geq 1$, y $\hat{f}_0 = 0$. Entonces:*

$$\|f\|^2_{H^s} = \sum_{k=1}^{\infty} (1 + k^2)^s \cdot \frac{2}{k^4} \sim \sum_{k=1}^{\infty} \frac{k^{2s}}{k^4} = \sum_{k=1}^{\infty} k^{2s-4}.$$

Esta serie converge si y solo si $2s - 4 < -1$, es decir, $s < 3/2$. Por lo tanto, $f \in H^s_{\text{per}}$ para todo $s < 3/2$. Como $s < 3/2$ no alcanza $s = 2$, concluimos que $f \notin H^2_{\text{per}}$, aunque f es continua (ya que $0 < 1/2$, tenemos $f \in C_{\text{per}}$).

El siguiente teorema conecta los espacios clásicos C^n_{per} con los espacios de Sobolev.

Teorema C.7. *Para todo $n \in \mathbb{N}$, se tiene:*

$$C^n_{\text{per}}([0, 2\pi]) \subset H^s_{\text{per}} \quad \text{para todo } s \leq n.$$

Más aún, si $f \in C^n_{\text{per}}([0, 2\pi])$, entonces $|\hat{f}_k| \leq C/k^n$ para alguna constante $C > 0$, y por lo tanto:

$$\|f\|_{H^s} < \infty \quad \text{para todo } s < n - 1/2.$$

Por otro lado, la unión de todos los espacios de Sobolev recupera las funciones suaves:

$$C^\infty_{\text{per}}([0, 2\pi]) = \bigcap_{s \in \mathbb{R}} H^s_{\text{per}}.$$

En resumen, los espacios H^s_{per} proporcionan una escala completa de espacios de funciones que interpolan entre distribuciones muy singulares ($s \ll 0$) y funciones infinitamente diferenciables ($s \to \infty$). Esta construcción es fundamental tanto para el análisis de ecuaciones diferenciales como para métodos numéricos basados en Fourier.