Índice general

Parte I: Análisis numérico matricial e interpolación **1**

1. Estabilidad y errores en el cálculo numérico **3**
1.1. Introducción 3
1.2. Representación de números en un computador 4
1.3. Aritmética en un sistema de representación finito 7
1.4. Estabilidad en los algoritmos numéricos 9
1.5. Ejercicios 10

2. Sistemas de ecuaciones numéricas lineales **15**
2.1. Introducción 15
2.2. Norma matricial subordinada a una norma vectorial 16
2.3. Estabilidad de un sistema de ecuaciones lineales 21
2.4. Sistemas lineales de gran dimensión 24
2.5. Matrices dispersas 25
2.6. Método de eliminación de Gauss 27
2.7. Métodos especiales para matrices simétricas 36
2.8. Factorización QR 38
2.8.1. Método de ortogonalización de Gram-Schmidt 38
2.8.2. Método de Householder 40
2.9. Métodos iterativos 44
2.10. Métodos iterativos clásicos 46
2.11. Ejercicios 55

3. Aproximación de autovalores **65**
3.1. Autovalores y vectores propios 65
3.2. Sucesiones de Krylov 67
3.3. Método de la potencia iterada 70
3.4. Método QR 73
3.5. Ejercicios 74

4. Aproximación de funciones **79**
4.1. Introducción 79
4.2. Evaluación de polinomios 80
4.3. Aproximación de funciones 81
4.4. Aproximación por mínimos cuadrados 85
4.5. Aproximación discreta por mínimos cuadrados 87
4.6. Polinomios de Chebyshev 97
4.7. Aproximación trigonométrica 100
4.8. Aproximación uniforme 107
4.9. Ejercicios 119

5. Interpolación de funciones **125**
5.1. Introducción 125
5.2. Interpolación de Lagrange 127
5.3. Método de Newton 129
5.4. Error en la interpolación de Lagrange 134
5.5. Algoritmos de Aitken y Neville 137
5.6. Interpolación compuesta 140
5.7. Interpolación de Hermite. 141
5.8. Interpolación por esplines cúbicos 146
5.9. Ejercicios 150

6. Derivación e integración numérica **157**
6.1. Introducción 157
6.2. Fórmulas de derivación numérica 158
6.3. Método de Extrapolación de Richardson 161
6.4. Cuadratura basada en la interpolación 164
6.5. Fórmulas cerradas de Newton-Cotes 167
6.6. Cuadratura compuesta 171
6.7. Fórmulas de Gauss 175
6.8. Ejercicios 178

Parte II: Resolución numérica de ecuaciones 185

7. Resolución numérica de ecuaciones no-lineales escalares 187
7.1. Introducción . . . 187
7.2. Método de dicotomía o bisección . . . 188
7.3. Métodos de punto fijo . . . 191
7.4. Velocidad de convergencia . . . 196
7.5. Método de la secante . . . 199
7.6. Método de Müller . . . 204
7.7. Método de Newton . . . 206
7.8. Método de Newton para raíces múltiples . . . 210
7.9. Raíces de ecuaciones polinómicas . . . 211
7.10. Ejercicios . . . 215

8. Resolución de sistemas de ecuaciones no lineales 223
8.1. Introducción . . . 223
8.2. Métodos de punto fijo . . . 224
8.3. Método de Newton . . . 227
8.4. Método de Broyden . . . 229
8.5. Raíces complejas de un polinomio . . . 231
8.6. Optimización sin restricciones . . . 233
8.7. Ejercicios . . . 237

9. Ecuaciones en diferencias finitas 241
9.1. Introducción . . . 241
9.2. Ecuaciones lineales homogéneas en diferencias con coeficientes constantes . . . 243
9.3. Ecuaciones lineales en diferencias con coeficientes constantes . 249
9.4. Estabilidad . . . 251
9.5. Ejercicios . . . 253

10. Problemas de valor inicial para ecuaciones diferenciales 261
10.1. Introducción . . . 261
10.2. Método de Euler . . . 262
10.3. Esquemas lineales multipaso . . . 265
10.4. Estabilidad de los métodos multipaso . . . 273
10.5. Convergencia de los métodos multipaso . . . 278
10.6. Estabilidad en intervalos que no están acotados . . . 281
10.7. Ejercicios . . . 282

11. Problemas de contorno para ecuaciones diferenciales 287
11.1. Introducción 287
11.2. Métodos de diferencias finitas 290
11.3. Análisis de la convergencia 293
11.4. Estabilidad, consistencia y convergencia 298
11.5. Otras condiciones de contorno 300
11.6. Ejercicios 301

CAPÍTULO 1

Estabilidad y errores en el cálculo numérico

1.1 Introducción

El saber contar y realizar operaciones aritméticas es el resultado de una larga evolución en el pensamiento humano. En todo este proceso, el hombre ha tratado de ayudarse en la realización de los cálculos mediante artilugios que simplificaran su labor. Primero fueron simples guijarros (calculi en latín), después fueron instrumentos mecánicos simples como el ábaco, instrumentos mecánicos articulados como las máquinas de Pascal o de Schickard y finalmente los computadores electrónicos. También, desde la programación en cilindros rotatorios o tarjetas perforadas hasta los desarrollos actuales en *software* basados en los lenguajes modernos de programación, se ha recorrido un largo camino. Todo este conocimiento y tecnología permite actualmente realizar cálculos de gran complejidad.

Sin embargo, un computador produce resultados en respuesta a cálculos programados que posiblemente difieren ligeramente de los valores exactos esperados. Ello es consecuencia de que trabajan con una aritmética discreta que no coincide plenamente con la aritmética exacta de los números enteros o reales. Todavía el ser humano no alcanza a realizar por medios físicos, todos los cálculos que su mente puede concebir, ni siguiera a representar en la memoria de un ordenador más que un subconjunto finito del conjunto de todos los números que puede manejar.

El propósito de este capítulo es estudiar las representaciones más comunes de números en un computador y las aritméticas que usan en sus cálculos. Una vez establecido que los errores respecto a la aritmética real pueden es-

tar presentes, se introduce el concepto de estabilidad numérica que permite seleccionar los algoritmos que son válidos para esta clase de cálculos.

1.2 Representación de números en un computador

La diferencia (en valor absoluto) entre el valor exacto x y el valor obtenido en un cálculo $\bar{x}$, determina el error absoluto cometido $E(x) = x \quad \bar{x}$. Una indicación más precisa del error cometido en el cálculo la da el error relativo, que se define por la siguiente expresión

$$E_R(x) = \frac{x \quad \bar{x}}{x}.$$

Si un determinado cálculo produce como resultado $\bar{x} = 1,2345 \diamond 10^{12}$ siendo el valor exacto esperado $x = 1.2331 \diamond 10^{12}$, el error absoluto sería $E(x) = 1.4 \diamond 10^{9}$. Este error parece muy alto si no se considera el orden de magnitud de los números con los que se realiza el cálculo. Por el contrario, el error relativo

$$E_R(x) = \frac{1.4 \diamond 10^{9}}{1.2331 \diamond 10^{12}} = 1.1353 \diamond 10^{\ 3}$$

da una idea más proporcionada de la imprecisión cometida.

Los computadores representan los números en sus memorias asignándoles una cantidad fija de posiciones a las que puede acceder. Esta cantidad puede variar de un computador a otro, de acuerdo con el estándar de representación que se haya atribuido a ese tipo de número. Es obvio que el modo de representación interna de un número en una memoria no tiene que coincidir necesariamente con el modo en que este número se muestra en una pantalla u otro dispositivo terminal.

Con el fin de comprender cómo usan los ordenadores la aritmética discreta, se recuerda el concepto de representación posicional de los números enteros y reales. Se considera el conjunto de números $\{$ que pueden ser representados en la forma

$$\leq 0.d_1 d_2 \times\times\times d_n \diamond b^e$$

donde $0 \ \mathcal{M} \, d_i < b$ para $i = 1, 2, \times\times\times, n$. El número entero positivo b es fijo y corresponde a la base de la representación. El entero e corresponde al exponente y puede variar en un determinado rango. Finalmente, el entero positivo fijo n controla la precisión de la representación. La parte fraccionaria es frecuentemente llamada mantisa. El valor numérico real asignado a esta representación es

$$x = \leq (d_1 b^{\ 1} + d_2 b^{\ 2} + \times\times\times + d_n b^{\ n}) \diamond b^e.$$

Por ejemplo, a la representación $0.1011 \diamond 2^3$ en base 2 le corresponde el valor decimal

$$x = 4 + 1 + \frac{1}{2} = 5.5$$

Idealmente, se puede ampliar el sistema admitiendo que n sea infinito y de este modo, se podrían incluir todos los números reales en esta representación. Obviamente, para un sistema de representación posicional que se pretenda poner en práctica en un computador, se requiere que n sea un número entero fijo y que se ajuste a la capacidad física de su memoria accesible. Por esta razón, todo sistema de representación que utilice un computador, tendrá un conjunto finito de números-máquina.

Habitualmente, para números enteros se emplea una representación en la que el exponente e es fijo e igual a n. De este modo, a la representación $\leq d_1 d_2 \times\times\times d_n$ le corresponde el valor entero

$$x = \leq (d_1 b^{n\ \ 1} + d_2 b^{n\ \ 2} + \times\times\times + d_n).$$

▪ Ejemplo **1** En un sistema binario ($b = 2$), con un número fijo de dígitos $n = 3$, el conjunto de números (que no son negativos) que pueden ser representados de este modo, son

Representación	000	001	010	011	100	101	110	111
Valor real	0	1	2	3	4	5	6	7

Un aspecto relevante de este sistema de representación es que puede producirse un desbordamiento en las operaciones aritméticas debido a que el conjunto es acotado. Por ejemplo, la suma de 3 y 5 produce un número que no puede ser representado en este sistema. ±

Si el exponente e es un entero variable en un determinado rango, el conjunto de números que pueden ser representados de este modo (llamado de punto flotante) es más amplio, como muestra el siguiente ejemplo

▪ Ejemplo **2** En un sistema binario $b = 2$, si el número de dígitos es $n = 2$, el conjunto de números (que no son negativos) que pueden ser representados en punto flotante con el rango $2\ \mathcal{M}e\ \mathcal{M}2$, es el siguiente

Representación	$.00 \diamond 2^e$	$.01 \diamond 2^e$	$.10 \diamond 2^e$	$.11 \diamond 2^e$
Valor	0	$\frac{1}{4} \diamond 2^e$	$\frac{1}{2} \diamond 2^e$	$\frac{3}{4} \diamond 2^e$

En definitiva, el conjunto de números que no son negativos y pueden ser representados de este modo son

$$\}0, \frac{1}{16}, \frac{1}{8}, \frac{3}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}, 2, 3|\ .$$

Los aspectos más relevantes de este sistema de representación son los siguientes:

- Los números representados no son equidistantes y se observa una mayor densidad en las proximidades de 0.
- Un número puede tener representaciones distintas. Por ejemplo, el número $\frac{1}{8}$ se puede representar por $.01 \diamond\ 2^{\ 1}$ o por $.10 \diamond\ 2^{\ 2}$.
- Se puede producir un desbordamiento en las operaciones aritméticas debido a que el conjunto es acotado. Por ejemplo, la suma de 1 y 3 produce un número que no puede ser representado en este sistema.
- El resultado de algunas operaciones aritméticas reales con números de este conjunto está fuera del conjunto aunque no se haya producido un desbordamiento del rango. Por ejemplo, la suma real de 2 y $\frac{1}{8}$ no pertenece al conjunto. $\pm$

Una representación en punto flotante está normalizada si el primer dígito en la parte fraccionaria es necesariamente distinto de 0. De este modo, se evita que un número pueda tener representaciones distintas en un mismo sistema. Actualmente, la representación que usan la mayoría de computadores es la llamada IEEE Standard 754 en punto flotante. Está basada en los tres elementos mencionados anteriormente, el signo, la mantisa y el exponente.

Si la representación es binaria, el primer dígito es 1 (dígito principal) y en la mayoría de las puestas en práctica de este estándar, no es almacenado en memoria (dígito principal implícito). El estándar de representación en punto flotante de IEEE que corresponde a lo que se conoce como precisión simple, utiliza 4 bytes de memoria (32 bits) de los cuales 8 bits son para el exponente E, 23 para la parte fraccionaria F y uno para el signo S. Cada bit almacena un 0 o un 1. El valor asignado a una representación es

$$(\ \ 1)^S \diamond\ 2^{E\ \ 127} \diamond\ 1.F$$

El campo del exponente necesita representar a la vez exponentes positivos y negativos. Para conseguirlo se añade un sesgo al valor positivo almacenado para lograr el exponente deseado. En el estándar IEEE 754 para precisión

simple, este valor es 127. De este modo, un valor almacenado E representa un valor real $E \quad 127$. Para doble precisión el campo del exponente tiene 11 bits y un sesgo de 1023. El bit del signo es 0 para positivos y 1 para negativos.

Por ejemplo, si en memoria está la siguiente información almacenada de acuerdo al estándar IEEE 754 con dígito principal implícito

$$\underbrace{1}\ \underbrace{01010011}_{\text{exponente}}\ \underbrace{10011110001010000101000}_{\text{mantisa}}$$

el número representado es

$$\left(1+\frac{1}{2}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^6}+\frac{1}{2^7}+\frac{1}{2^{11}}+\frac{1}{2^{13}}+\frac{1}{2^{18}}+\frac{1}{2^{20}}\right)\diamond\ 2^{83\ \ 127}$$

ya que $01010011_2 = 83$.

1.3 Aritmética en un sistema de representación finito

En general, cuando se utiliza un sistema discreto $\{\ \ $, inevitablemente aparecen errores al introducir los números reales convirtiéndolos a números-máquina pero también como el resultado de una operación entre números-máquinas que en general no coincide con el resultado que se obtendría en la aritmética real. Conceptualmente, los números reales pueden aproximarse por números del sistema discreto de dos modos ligeramente distintos que se conocen como truncamiento y redondeo.

La puesta en práctica de métodos de truncamiento está basada en la siguiente idea: Un numero real puede ser aproximado por un número de punto flotante con parte fraccionaria infinita en la base b

$$\leq .d_1 d_2 \times\times\diamond\ \ b^e.$$

Consecuentemente, si se trunca esta serie con n dígitos se obtiene una aproximación en el sistema discreto disponible en el computador. Sin embargo, es más usado el llamado método de redondeo que consiste en aproximar un número real positivo por el número-máquina más próximo. Este procedimiento sería el que nos proporcionaría mayor precisión. Para precisar estas ideas se representa la función parte entera de un número real por un corchete [ ] y se define para un número real positivo representado por $x = 0.m \diamond\ \ b^e$ donde m es una cifra con posiblemente una infinidad de dígitos, las siguientes aproximaciones por números máquina de n dígitos