MÉTODOS MATEMÁTICOS PARA LA INGENIERÍA

Cálculo Numérico

EDUARDO CASAS RENTERIA

Universidad de Cantabria

2020

Contenidos

Capítulo 1

Cuestiones Básicas sobre Aritmética Computacional

1.1 Introducción

El objetivo de este capítulo es describir cómo el computador almacena los números y opera con ellos. Esto es fundamenal para comprender los errores que comete durante este proceso y cómo afecta al resultado de un algoritmo numérico. Esto nos ayudará a distinguir los algoritmos que son estables de los inestables y a organizar los cálculos de una forma más eficiente. El plan del capítulo es el siguiente. En primer lugar, vamos a presentar algunos ejemplos que muestran lo inesperadamente grandes que pueden resultar los errores que comete el computador en la ejecución de un algoritmo. Esto justifica el análisis a desarrollar en el resto del capítulo. En la sección 3 veremos cómo se pueden representar los números reales en diferentes bases. Prestaremos especial atención a la base 2 porque es el formato especificado por el Standard IEEE que rige el funcionamiento de la mayoría de los computadores. Analizaremos como cambiar los números entre la base 2 y la base 10. Seguidamente, se describirá cómo se almacenan los números sobre el computador de acuerdo a este Standard. Por último, en la sección cuarta del capítulo se describirá la aritmética de punto flotante y se darán cotas del error de redondeo que conllevan estas operaciones. Como material complementario sobre este tema el lector puede consultar los libros [1] y [6].

1.2 Algunos Ejemplos de Errores Cometidos por el Computador

Ejemplo 1: Ejecutamos $10^{50} + 812 - 10^{50} + 10^{35} + 511 - 10^{35}$.

Resultado exacto: 1323

Resultado proporcionado por el computador: 0

Ejemplo 2: Ejecutamos

x=0;for j=1:10;x=x+0.1;end,x-1

Resultado exacto: 0

Resultado proporcionado por el computador:
-1.110223024625157e-16

Ejemplo 3: Se define $\varepsilon_M = 2^{-53}$. Calculamos

$$1+\varepsilon_M-1, \quad -1+\varepsilon_M+1 \quad \text{y} \quad (1+2^{-52})+\varepsilon_M-(1+2^{-52}).$$

Resultados exactos: ε_M en todos los casos

Resultados proporcionado por el computador: 0, ε_M y $2\varepsilon_M$.

Ejemplo 4: Es sabido que $\mathrm{e}^x = \displaystyle\sum_{k=0}^{\infty} \frac{x^k}{k!}$, entonces nos planteamos los cálculos aproximados

$$\mathrm{e}^{-30} \approx \sum_{k=0}^{1000} \frac{(-30)^k}{k!} \quad \text{y} \quad \mathrm{e}^{-40} \approx \sum_{k=0}^{1000} \frac{(-40)^k}{k!}$$

Resultados exactos redondeados a 15 dígitos decimales:

$$\mathrm{e}^{-30} = 9.357622968840175 \times 10^{-14}$$
$$\mathrm{e}^{-40} = 4.248354255291589 \times 10^{-18}$$

Resultados proporcionado por el computador:
$-3.066812356356220 \times 10^{-5}$ y -3.165731894063124.

Si ahora hacemos

$$\mathrm{e}^{-30} = \frac{1}{\mathrm{e}^{30}} \approx \left(\sum_{k=0}^{100} \frac{30^k}{k!}\right)^{-1} \quad \text{y} \quad \mathrm{e}^{-40} = \frac{1}{\mathrm{e}^{40}} \approx \left(\sum_{k=0}^{100} \frac{40^k}{k!}\right)^{-1},$$

entonces obtenemos

Resultados proporcionado por el computador:
$9.357622968840172 \times 10^{-14}$ y $4.248354255291587 \times 10^{-18}$

Ejemplo 5: Se desea aproximar $f'(1)$, con f dada por

$$f(x) = (\mathrm{e}^x - 1)^2 + \left(\frac{1}{\sqrt{1+x^2}} - 1\right)^2$$

Teniendo en cuenta que

$$f'(1) = \lim_{h\to 0} \frac{f(1+h) - f(1)}{h},$$

entonces aproximamos la derivada usando la fórmula

$$f'(1) \approx \frac{f(1+h) - f(1)}{h}$$

con h pequeño. Entonces obtenemos

Resultado exacto: $9.54865532212975587\ldots$

Resultado proporcionado por el computador:

h	$\frac{f(1+h) - f(1)}{h}$
10^{-1}	10.855456716919706
10^{-2}	9.670876355889924
10^{-3}	9.560797238114205
10^{-4}	9.549868716067778
10^{-5}	9.548776653556246
10^{-6}	9.548667453174174
10^{-7}	9.548656536573217
10^{-8}	9.548655244273617
10^{-9}	9.548654755775486
10^{-10}	9.548646318080497
10^{-11}	9.548584145591120
10^{-12}	9.548362100986195
10^{-13}	9.530154443382342
10^{-14}	9.325873406851313
10^{-15}	9.325873406851315
10^{-16}	0
10^{-17}	0
10^{-18}	0

Ejemplo 6: Se desea aproximar $f''(1)$, con f dada por

$$f(x) = \frac{4970x - 4923}{4970x^2 - 9799x + 4830}$$

Teniendo en cuenta que

$$f''(1) = \lim_{h\to 0} \frac{f(1-h) - 2f(1) + f(1+h)}{h^2},$$

usamos la fórmula

$$f''(1) \approx \frac{f(1-h) - 2f(1) + f(1+h)}{h^2}$$

con h pequeño. Entonces obtenemos

Resultado exacto: $f''(1) = 94$

Resultado proporcionado por el computador:

h	$\frac{f(1-h)-2f(1)+f(1+h)}{h^2}$
10^{-1}	$-9.790002023881781e+03$
10^{-2}	$-9.176994887505759e+05$
10^{-3}	$-2.250198293097583e+03$
10^{-4}	70.787242378855808
10^{-5}	93.160323899610361
10^{-6}	$1.429683038622897e+02$
10^{-7}	$-2.934541498689212e+03$
10^{-8}	$8.947154128691178e+05$
10^{-9}	$4.321520918892927e+07$
10^{-10}	0
10^{-11}	0
10^{-12}	$4.365574568510054e+13$
10^{-13}	$-4.365574568510054e+15$
10^{-14}	0
10^{-15}	$4.365574568510055e+19$
10^{-16}	$-4.365574568510054e+21$
10^{-17}	0
10^{-18}	0
10^{-19}	0

Ejemplo 7: Ejecutamos las siguientes instrucciones MATLAB

- A = randn(25,25);
- A = A*A′;
- v = eig(A);
- t = max(v);
- det(A-t*eye(25))

La primera instrucción genera una matriz aleatoria A de orden 25×25. A continuación se multiplica A por su traspuesta, llamando A de nuevo al resultado. Así A es una matriz simétrica. El comando eig calcula todos los valores propios de A, que son almacenados en el vector v. Seguidamente se calcula t, el valor propio más grande. Finalmente, se calcula el determinante de A-tI, donde I es la matriz identidad 25×25. Si los cálculos fueran exactos, el resultado debería ser cero. Pero, tanto el cálculo de los valores propios como el del determinante es aproximado. Dado que la precisión con la que MATLAB determina el valor propio t es bastante buena y el determinante se calcula usando la factorización LU, se espera que el resultado sea próximo a cero. Sin embargo, repitiendo las instrucciones en tres ocasiones, obtenemos los siguientes resultados: $2.281842616071255 \times 10^{+27}$, $3.084889838349146 \times 10^{+30}$ y $4.046186281298098 \times 10^{+29}$.

Por supuesto el lector obtendrá resultados distintos dado que las matrices serán distintas al ser generadas aleatoriamente. Pero también obtendrá números sorprendentemente grandes. Para conseguir la misma matriz podemos fijar la semilla en la generación de los números aleatorios. Así podemos ejecutar

- randn('seed',n);
- A = randn(25,25);
- A = A*A';
- v = eig(A);
- t = max(v);
- det(A-t*eye(25))

para distintos valores de n. Entonces el lector podrá comprobar los siguientes resultados para los valores de n indicados:

n	Resultado
1	$1.140833929431342 \times 10^{+28}$
2	$4.399317258527512 \times 10^{+26}$
3	$3.063722766181041 \times 10^{+29}$

En las siguientes secciones veremos por qué se producen estos resultados.

1.3 Representación de los Números en Base b

Hoy en día utilizamos la base 10 para manejar los números. Así por ejemplo, el número 6485 no es otra cosa que $6 \times 10^3 + 4 \times 10^2 + 8 \times 10 + 5$. El número 794.10256 se corresponde con

$$7\times10^2+9\times10+4\times10^0+1\times10^{-1}+0\times10^{-2}+2\times10^{-3}+5\times10^{-4}+6\times10^{-5}.$$

De forma general, un número real positivo x se puede escribir en la forma $x = \sum_{k=-\infty}^{n} a_k \times 10^k$, donde cada término a_k es un número natural entre 0 y 9. Si x posee un número finito de decimales, digamos m decimales, entonces $a_k = 0$ para cada $k < -m$.

Sin embargo, no siempre se utilizó la base 10 para representar los números. No es el objetivo de este capítulo hacer un recorrido histórico de la representación de los números a lo largo de la historia de la humanidad. Simplemente recordemos los números romanos e imaginemos que deseamos multiplicar 553 por 25 usando números romanos: DLIII×XXV. En seguida nos daremos cuenta de la importante ventaja del sistema decimal, cuyo origen se encuentra en la India, adoptado por los árabes en el siglo IX, e introducido en Europa por los árabes en el siglo XIII.

Ahora bien, ¿por qué se elige la base 10? La representación de los números indicada en la base 10 también puede hacerse eligiendo cualquier otro número natural $b > 1$. En efecto cualquier número real positivo x puede escribirse en la forma $x = \sum_{k=-\infty}^{n} a_k \times b^k$ con $0 \leq a_k \leq b-1$ para cada k. Pensemos por un momento en la tabla de multiplicar que todos aprendemos en el colegio. Si elegimos $b = 5$, entonces la tabla sería mucho más corta y fácil de aprender. Sin embargo, la escritura de los números sería más larga. Por ejemplo, el número $x = 9757$ en base 10, sería 303012 en base 5, o $x = 1.000.000$ sería 224.000.000.

También podría elegirse una base $b > 10$. Por ejemplo, tomemos $b = 16$. Entonces, cada número real x se escribiría en la forma $x = \sum_{k=-\infty}^{n} a_k \times 16^k$ con $0 \leq a_k \leq 15$ para cada k. Esto requiere de la introducción de signos apropiados para denotar los números 10, 11, 12, 13, 14 y 15. Asignemos a estos números las letras A, B, C, D, E y F. Entonces, si $x = 1.000.000$ en base 10, en base 16 sería $F4240$. Esta es una expresión mucho más corta que la correspondiente en base 5. Sin embargo, la tabla de multiplicar en base 16 sería bastante más larga. Por lo tanto, la base 10 puede verse como un compromiso entre una representación no demasiado larga de los números y una complicación computacional admisible (la tabla de multiplicar).

Ahora bien, cuando un computador tiene que almacenar un número y tiene que operar, las razones para elegir una u otra base son distintas. Razones tecnológicas hacen que la elección de la base 2 sea la más adecuada para guardar un número en el computador ya que solamente debe distinguir entre dos dígitos: 0 y 1. Por otro lado, el uso de base 16, utilizada en algunos computadores, permite manejar números mucho más grandes y pequeños en el mismo espacio, lo que para ciertos cálculos resulta importante. Sin embargo, el Standard de IEEE propone el uso de la base 2 en el almacenamiento y la aritmética sobre el computador. Es por ello que nos vamos a centrar en la base 2. Como primer paso vamos a ver cómo averiguar el valor en base 10 de un número escrito en base 2, para proceder después con el cambio contrario. Sea $x = 101110.0111001$ un número en base 2 y deseamos conocer su valor en base 10. El proceso es muy sencillo si atendemos a lo que representan los dígitos, como ya se ha indicado antes:

$$x = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 0 \times 2^{-6} + 1 \times 2^{-7} = 46.4453125$$

Recíprocamente, supongamos que $x = 357.1909$ está en base 10 y queremos escribirlo en base 2. Aquí vamos a distinguir la parte real de la parte decimal. Primero pasaremos a base 2 el número 357. Tenemos que obtener los dígitos $a_n a_{n-1} \ldots a_2 a_1 a_0$, es decir:

$$357 = a_0 + a_1 \times 2 + a_2 \times 2^2 + \ldots + a_n \times 2^n = a_0 + 2[a_1 + a_2 \times 2 + \ldots + a_n \times 2^{n-1}].$$

Si dividimos 357 entre 2 nos quedará un cociente y un resto que llamaremos q y r (con $r < 2$, $r = 0$ o 1), respectivamente. Sabemos que $357 = r + 2q =$ resto $+$

divisor×cociente. Si comparamos esta identidad con la de arriba, tenemos que $r = a_0$ y $q = a_1 + a_2 \times 2 + \ldots + a_n \times 2^{n-1}$. Así, realizando la división, tenemos que el resto es $r = 1$ y el cociente $q = 178$. Por lo tanto, ya tenemos $a_0 = 1$ y ahora tenemos que repetir el algoritmo para determinar a_1:

$$178 = a_1 + 2[a_2 + a_3 \times 2 + \ldots + a_n \times 2^{n-2}].$$

Dividiendo 178 entre 2 obtenemos un nuevo cociente = 89 y el resto = 0. Por lo tanto, se tiene $a_1 = 0$ y

$$89 = a_2 + 2[a_3 + a_4 \times 2 + \ldots + a_n \times 2^{n-3}].$$

Dividiendo 89 entre 2 obtenemos el cociente = 44 y el resto = 1. Por lo tanto, se tiene $a_2 = 1$ y

$$44 = a_3 + 2[a_4 + \ldots + a_n \times 2^{n-4}].$$

Dividiendo 44 entre 2 obtenemos el cociente = 22 y el resto = 0. Por lo tanto, se tiene $a_3 = 0$ y

$$22 = a_4 + 2[a_5 + \ldots + a_n \times 2^{n-5}].$$

Dividiendo 22 entre 2 obtenemos el cociente=11 y el resto=0. Por lo tanto, se tiene $a_4 = 0$ y

$$11 = a_5 + 2[a_6 + \ldots + a_n \times 2^{n-6}].$$

Dividiendo 11 entre 2 obtenemos el cociente = 5 y el resto = 1. Por lo tanto, se tiene $a_5 = 1$ y

$$5 = a_6 + 2[a_7 + \ldots + a_n \times 2^{n-7}].$$

Dividiendo 5 entre 2 obtenemos el cociente = 2 y el resto = 1. Por lo tanto, se tiene $a_6 = 1$ y

$$2 = a_7 + 2[a_8 + \ldots + a_n \times 2^{n-8}].$$

Por último, dividiendo 2 entre 2 obtenemos el cociente = 1 y el resto = 0. Por lo tanto, se tiene $a_7 = 0$ y $a_8 = 1$. Con lo cual tenemos que 357 en base 10 coincide con 101100101 en base 2.

Procedamos ahora con la parte decimal de x: 0.1909. Ahora tenemos que calcular los dígitos $0.a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}\ldots$ de forma que

$$0.1909 = \frac{a_{-1}}{2} + \frac{a_{-2}}{2^2} + \frac{a_{-3}}{2^3} + \frac{a_{-4}}{2^4} + \frac{a_{-5}}{2^5} + \ldots$$

Para ello multiplicamos a ambos lados de la igualdad por 2 y obtenemos

$$0.3818 = a_{-1} + \frac{a_{-2}}{2} + \frac{a_{-3}}{2^2} + \frac{a_{-4}}{2^3} + \frac{a_{-5}}{2^4} + \ldots$$

Necesariamente a_{-1} debe ser 0 porque en caso contrario la suma de la derecha sería mayor o igual que 1, siendo imposible la identidad. Para calcular a_{-2} multiplicamos por 2 la identidad anterior, eliminado a_{-1} que ya es conocido, y obtenemos

$$0.7636 = a_{-2} + \frac{a_{-3}}{2} + \frac{a_{-4}}{2^2} + \frac{a_{-5}}{2^3} + \ldots$$

Entonces, al igual que antes, tenemos que $a_{-2} = 0$. Volvamos a multiplicar por 2 la igualdad anterior

$$1.5272 = a_{-3} + \frac{a_{-4}}{2} + \frac{a_{-5}}{2^2} + \dots$$

Ahora a_{-3} debe ser 1 necesariamente. ¿Por qué no puede ser 0? Si fuera cero, entonces el resto de los términos debe sumar 1.5272 y esto no es posible. En efecto, lo más grande que puede ser la suma se corresponde con $a_{-k} = 1$ para cada $k \geq 4$. Pero sabemos sumar los términos de una progresión geométrica

$$\sum_{k=1}^{\infty} \frac{1}{2^k} = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = 1.$$

Por lo tanto, la suma de los términos restantes no puede ser mayor que 1 y, en consecuencia, $a_{-3} = 1$ y

$$0.5272 = \frac{a_{-4}}{2} + \frac{a_{-5}}{2^2} + \frac{a_{-6}}{2^3} + \dots$$

Volviendo a multiplicar por 2 se obtiene

$$1.0544 = a_{-4} + \frac{a_{-5}}{2} + \frac{a_{-6}}{2^2} + \dots$$

Entonces $a_{-4} = 1$ y

$$0.0544 = \frac{a_{-5}}{2} + \frac{a_{-6}}{2^2} + \dots$$

Repitiendo este algoritmo iremos sacando los sucesivos decimales del número $x = 357.1909$. Así tenemos: $x = 101100101.0011\dots$ en base 2.

Es importante observar que, aunque el número x posea un número finito de decimales en base 10, puede tener infinitos decimales en base 2. Vamos a comprobar esto con el número decimal *más sencillo*: $x = 0.1$.

$$\begin{aligned}
0.1 &= \frac{a_{-1}}{2} + \frac{a_{-2}}{2^2} + \frac{a_{-3}}{2^3} + \frac{a_{-4}}{2^4} + \frac{a_{-5}}{2^5} + \dots \\
0.2 &= a_{-1} + \frac{a_{-2}}{2} + \frac{a_{-3}}{2^2} + \frac{a_{-4}}{2^3} + \frac{a_{-5}}{2^4} + \dots \Rightarrow a_{-1} = 0 \\
0.4 &= a_{-2} + \frac{a_{-3}}{2} + \frac{a_{-4}}{2^2} + \frac{a_{-5}}{2^3} + \frac{a_{-6}}{2^4} + \dots \Rightarrow a_{-2} = 0 \\
0.8 &= a_{-3} + \frac{a_{-4}}{2} + \frac{a_{-5}}{2^2} + \frac{a_{-6}}{2^3} + \frac{a_{-7}}{2^4} + \dots \Rightarrow a_{-3} = 0 \\
1.6 &= a_{-4} + \frac{a_{-5}}{2} + \frac{a_{-6}}{2^2} + \frac{a_{-7}}{2^3} + \frac{a_{-8}}{2^4} + \dots \Rightarrow a_{-4} = 1 \\
0.6 &= \frac{a_{-5}}{2} + \frac{a_{-6}}{2^2} + \frac{a_{-7}}{2^3} + \frac{a_{-8}}{2^4} + \frac{a_{-9}}{2^5} + \dots \\
1.2 &= a_{-5} + \frac{a_{-6}}{2} + \frac{a_{-7}}{2^2} + \frac{a_{-8}}{2^3} + \frac{a_{-9}}{2^4} + \dots \Rightarrow a_{-5} = 1 \\
0.2 &= \frac{a_{-6}}{2} + \frac{a_{-7}}{2^2} + \frac{a_{-8}}{2^3} + \frac{a_{-9}}{2^4} + \dots
\end{aligned}$$

Notemos que el número 0.2 ya había aparecido más arriba, por lo que si continuamos el algoritmo se repetirán los dígitos. Así obtenemos que 0.1 en base 2 es $0.0001100110011\ldots = 0.\widehat{0011}$, donde el período $\widehat{0011}$ se repite indefinidamente.

Terminamos esta sección con una observación sobre la unicidad en la representación de los números. La forma de representar un número que posee infinitos decimales es única, lo que sucede siempre con los números irracionales o aquellos racionales que repiten periódicamente e indefinidamente los decimales a partir de uno dado. Sin embargo, aquellos números racionales que poseen un número finito de decimales admiten dos representaciones distintas. Por ejemplo, si $x = 138.564$, también podemos escribirlo en la forma $x = 138.5639999999\ldots$ En el primer caso tenemos

$$x = 1 \times 10^2 + 3 \times 10^1 + 8 \times 10^0 + 5 \times 10^{-1} + 6 \times 10^{-2} + 4 \times 10^{-3}$$

y en el segundo

$$x = \sum_{k=-\infty}^{2} a_k \times 10^k \quad \text{con} \quad a_k = \begin{cases} 9 & \text{si } -\infty < k \leq -4, \\ 3 & \text{si } k = -3, \\ 6 & \text{si } k = -2, \\ 5 & \text{si } k = -1, \\ 8 & \text{si } k = 0, \\ 3 & \text{si } k = 1, \\ 1 & \text{si } k = 2. \end{cases}$$

Ambas expresiones son coincidentes, basta observar que la segunda opción satisface

$$\sum_{k=-\infty}^{2} a_k \times 10^k = 138.563 + \sum_{k=-\infty}^{-4} 9 \times 10^k = 138.563 + 9 \sum_{k=-\infty}^{-4} 10^k$$
$$= 138.563 + 9\frac{10^{-4}}{1 - 10^{-1}} = 138.564$$

En base 2 ocurre lo mismo. Si tomamos $x = 1011.1011101$, este número coincide con $1011.10111001111\ldots = 1011.10111001\hat{1}$. En efecto, se tiene

$$0.0000000\hat{1} = \sum_{k=8}^{\infty} \frac{1}{2^k} = \frac{\frac{1}{2^8}}{1 - \frac{1}{2}} = \frac{1}{2^7} = 2^{-7}.$$

Por lo tanto, se deduce que

$$1011.1011100\hat{1} = 2^3 + 2 + 1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + \sum_{k=8}^{\infty} \frac{1}{2^k}$$
$$= 2^3 + 2 + 1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-7} = 1011.1011101.$$

Obviamente, entre las dos representaciones siempre escogeremos la forma corta, es decir, la que tiene una representación finita. Con esta elección, la representación resulta única.

1.4 Almacenamiento de los Números en el Computador

Todo número real distinto de cero puede escribirse en base 10 en la forma: $x = (-1)^s \times 10^e \times a.f_1f_2f_3\ldots = (-1)^s \times 10^e \times a.f$, donde $s = 0$ si $x \geq 0$ y $s = 1$ si $x < 0$, $1 \leq a \leq 9$ es un número natural y $f = f_1f_2f_3\ldots$ es una sucesión finita o infinita de números naturales con $0 \leq f_i \leq 9$ para cada $i = 1, 2, 3, \ldots$ Pongamos dos ejemplos

$$x = 8765.098 \Rightarrow x = (-1)^0 \times 10^3 \times 8.765098$$
$$x = -0.000198654 \Rightarrow x = (-1)^1 \times 10^{-4} \times 1.98654$$

El número e es el exponente y f es la mantisa. El signo del número queda determinado por s. Es importante destacar que a debe ser distinto de cero. Cuando la base con la que trabajamos es 2, entonces los números se pueden representar en la forma $x = (-1)^s \times 2^e \times 1.f$. Esto se conoce como representación en la forma punto flotante. Cuando el computador almacena x, debe guardar s, el exponente e y la mantisa f, no necesita guardar el 1, ganando así una posición de memoria. A esto se le llama el bit oculto. Teniendo en cuenta que la mantisa puede estar formada por infinitos dígitos, debemos decidir cuántos de estos dígitos almacenerá el computador porque obviamente no puede guardarlos todos. Vamos a detallar aquí cómo se almacenan los números en la llamada *doble precisión*. Ésta es la precisión más utilizada en los cálculos y la que usaremos en los cálculos con MATLAB. El computador destina 64 bits para almacenar un número real, distribuidos como sigue: 1 bit para el signo (el primero), los 11 bits siguientes son para guardar el exponente y los 52 bits restantes son para la mantisa. Si el número es positivo, entonces el bit del signo es un 0, siendo un 1 en caso contrario. En 11 bits, el número más grande que se puede almacenar es

$$1 + 2 + 2^2 + 2^3 + \ldots + 2^{10} = 2^{11} - 1 = 2047$$

Ahora bien, en la representación en forma de punto flotante de x, el exponente será positivo si $x > 1$ y negativo si $x < 1$. Teniendo en cuenta que la parte entera de 2047/2 es 1023, los números que almacena el ordenador son los siguientes:

Signo (s)	Exponente (e)	Mantisa (f)
1 bit	11 bits	52 bits

Números Normalizados: $0 < e < 2047 \Rightarrow x = (-1)^s \times 2^{e-1023} \times 1.f$
Cero: $e = 0$ **y** $f = 0 \Rightarrow x = 0$
Números No Normalizados: $e = 0$ **y** $f \neq 0 \Rightarrow x = (-1)^s \times 2^{-1022} \times 0.f$
Infinito: $e = 2047$ **y** $f = 0 \Rightarrow x = (-1)^s \infty$
Not a Number: $e = 2047$ **y** $f \neq 0 \Rightarrow x = \text{NAN}$

Ahora podemos averiguar el número real más grande que puede guardar el computador. Consiste en tomar $e = 2046$ y $f_j = 1$ para $j = 1, 2, \ldots, 52$:

$$\text{realmax} = 2^{1023} \times \left(1 + \sum_{j=1}^{52} \frac{1}{2^j}\right) \approx 1.797693134862316 \times 10^{+308}$$

Si escribimos sobre MATLAB 2^{1024}, el resultado es Inf, es decir ∞. Entonces se produce lo que se llama un overflow. El computador no puede guardar un número tan grande.

El número estrictamente positivo más pequeño normalizado se alcanza para $e = 1$ y $f = 0$:

$$\text{realmin} = 2^{-1022} \approx 2.225073858507201 \times 10^{-308}$$

El número estrictamente positivo más pequeño no normalizado se alcanza con $e = 0$, $f_j = 0$ para $j = 1, \ldots, 51$ y $f_{52} = 1$: $2^{-1074} \approx 4.940656458412465 \times 10^{-324}$. Si escribimos sobre MATLAB 2^{-1075} el resultado es 0. En este caso se produce lo que se llama un underflow. El computador no puede guardar exactamente ningún número x con $0 < x < 2^{-1074}$. Lo que hace es redondearlo a 2^{-1074} o a 0 según esté más próximo a uno u otro.

Por supuesto, todos estos números cambian si extendemos la precisión del software o trabajamos con una base distinta a 2.

Dado un número real x se denota por $fl(x)$ el número guardado por el computador. La notación fl proviene del inglés *floating*. Ya hemos visto en la sección 1.3 que la representación binaria de un número real, aun teniendo un cantidad finita de decimales en base 10, puede tener infinitos dígitos decimales. Este era el caso del número 0.1, con un solo decimal en base 10 posee infinitos decimales en base 2. Dado que la mantisa f que guarda el computador posee una longitud de 52 bits, los dígitos decimales posteriores al 52 se pierden. Hay dos formas de proceder a la hora de eliminar los dígitos decimales posteriores al 52: truncación o redondeo. El proceso de truncación consiste simplemente en quedarse con los 52 primeros dígitos de la mantisa, prescindiendo del resto. El redondeo consiste en tomar el número almacenable exactamente en el computador más próximo al dado. Para entender este proceso vamos a poner algunos ejemplos en un computador imaginario que trabaja en base $b = 10$ y posee una mantisa de 4 dígitos. Vamos a tomar dos números $x = 1534.6809$ e $y = 0.00001287609$. Teniendo en cuenta que $x = 10^3 \times 1.5346809$ e $y = 10^{-5} \times 1.287609$ y que la longitud de la mantisa es 4, entonces si el computador trunca la mantisa, se tiene $fl(x) = 10^3 \times 1.5346$ y $fl(y) = 10^{-5} \times 1.2876$. Si por el contrario, el computador redondea, entonces $fl(x) = 10^3 \times 1.5347$ y $fl(y) = 10^{-5} \times 1.2876$. En el caso de y ambos procesos coinciden porque $fl(y)$ es el número almacenable en el computador más próximo a y. Sin embargo, en el caso de x ambos procesos conducen a distinto resultado. Es obvio que el redondeo produce un error más pequeño: $|x - fl(x)| = 0.0191$. El error de truncamiento es $|x - fl(x)| = 0.0809$.

El Standard de IEEE establece el redondeo como proceso a seguir. La cuestión que surge es cómo proceder cuando el redondeo al alza o a la baja producen el mismo error. Por ejemplo, tomemos sobre el mismo computador anterior $x = 12.8765 = 10 \times 1.28765$. Entonces 10×1.2876 y 10×1.2877, que se obtienen redondeando a la baja y al alza respectivamente, producen el mismo error absoluto: 0.00005. La regla a seguir en esta situación se conoce como redondeo al par, lo que significa que se escoge el número cuyo último dígito sea un número par. En este caso tendríamos $fl(x) = 10 \times 1.2876$. En el caso de la base 2, el redondeo al par consiste en tomar el número cuyo último dígito es 0.

Veamos cómo es $fl(0.1)$ en doble precisión de acuerdo al Standard de IEEE. Vimos en la sección 1.3 que 0.1 en base 2 es

$$0.0001100110011\ldots = 0.\widehat{00011} = 2^{-4} \times 1.100110011001\ldots$$
$$= 2^{-4} \times 1.\widehat{1001} = 2^{-4} \times 1.\overbrace{1001\ldots1001}^{52 \text{ bits}}1001\ldots$$

Ahora el redondeo conduce a

$$fl(x) = 2^{-4} \times 1.\overbrace{1001\ldots1001}^{48 \text{ bits}}1010$$

Restando $fl(x) - x$ se obtiene

$$2^{-4} \times 1.\overbrace{1001\ldots1001}^{48 \text{ bits}}1010$$
$$- 2^{-4} \times 1.\overbrace{1001\ldots1001}^{48 \text{ bits}}1001\,1001\,1001\ldots$$
$$= 2^{-4} \times 0.\overbrace{0000\ldots0000}^{48 \text{ bits}}0000\,0110\,0110\ldots$$

Por lo tanto $fl(x) - x = 2^{-56} \times 0.\widehat{0110} = 2^{-58} \times 1.\widehat{1001}$. De aquí sacamos dos consecuencias. En primer lugar, el computador no puede almacenar correctamente el número real 0.1 en formato punto flotante. La segunda consecuencia es que $fl(x) > x$. Sin embargo, si examinamos de nuevo el Ejemplo 2 de la sección 1.2, vemos que la suma de $fl(x)$ consigo mismo 10 veces produce un número menor que 1 y no mayor como cabría esperar del hecho de que $fl(x) > x$. Esto nos lleva a analizar el error que se comete al almacenar un número y al operar en aritmética de punto flotante.

1.5 Errores en Aritmética en Punto Flotante

Dado un número real $x = (-1)^s \times 2^{e_x} \times 1.f$ con $0 < e_x < 1024$, de acuerdo a las reglas de redondeo explicadas en la sección anterior, se tiene que $|x - fl(x)| \leq 2^{e_x} \times 2^{-53}$, por lo tanto $\frac{|x-fl(x)|}{|x|} \leq 2^{-53}$. A esta cantidad 2^{-53} se le denomina precisión de la máquina y en lo sucesivo la denotaremos por ε_M. Si definimos $\varepsilon_x = \frac{fl(x)-x}{x}$, entonces se tiene

$$fl(x) = x(1 + \varepsilon_x) \quad \text{con} \quad |\varepsilon_x| \leq \varepsilon_M. \tag{1.1}$$

Si denotamos por $\circledast$ una operación aritmética (suma, resta, producto o división) y $x, y \in \mathbb{R}$, entonces para calcular $x \circledast y$ el computador realiza el siguiente proceso: $fl(fl(x) \circledast fl(y))$. Es decir, primero almacena los números en formato punto flotante, después realiza la operación con los números obtenidos y finalmente vuelve a reducir el resultado a formato punto flotante. Para ilustrar este

proceso vamos a realizar la suma de dos números x e y en un máquina imaginaria que trabaja en base 10 con 4 dígitos de mantisa. Sean $x = 12.98557$ e $y = 0.1234567$, entonces

$$fl(x) = 10 \times 1.2986, \quad fl(y) = 10^{-1} \times 1.2346,$$
$$fl(x) + fl(y) = 12.986 + 0.12346 = 13.10946, \; fl(fl(x) + fl(y)) = 10 \times 1.3109.$$

Pongamos otro ejemplo: $x = 9875.3167$ e $y = 0.01876567$, entonces

$$\begin{aligned} fl(x) &= 10^3 \times 9.8753, \quad fl(y) = 10^{-2} \times 1.8766, \\ fl(x) + fl(y) &= 9875.3 + 0.018766 = 9875.318766, \\ fl(fl(x) + fl(y)) &= 10^3 \times 9.8753 = fl(x). \end{aligned}$$

Observamos que y actúa como si fuese un 0, no altera el valor de $fl(x)$. Esto sucede siempre que sumamos dos números con una diferencia grande en su tamaño. Analicemos el Ejemplo 1 de la sección 1.2. Primero observemos que $fl(10^{50}) = 2^{166} \times 1.f_x$. En efecto, basta observar que $2^{166} < 10^{50} < 2^{167}$. No necesitamos averiguar cuál es la mantisa para las operaciones que vamos a realizar. Para el resto de los números tenemos: $fl(812) = 2^9 \times 1.f_y$, $fl(10^{35}) = 2^{116} \times 1.f_z$ y $fl(511) = 2^8 \times 1.f_w$. Entonces sumamos $10^{50} + 812$

$$\begin{aligned} &fl(2^{166} \times 1.f_x + 2^9 \times 1.f_y) = fl(2^{166} \times (1.f_x + 0.\overbrace{0\ldots0}^{156 \text{ ceros}}\, 1f_y)) \\ &= 2^{166} \times 1.f_x = fl(10^{50}). \end{aligned}$$

Ahora, restando 10^{50} obtenemos $10^{50} + 812 - 10^{50} : fl(fl(10^{50}) - fl(10^{50})) = 0$. Sumando ahora $10^{35} + 511$ obtenemos

$$\begin{aligned} &fl(2^{116} \times 1.f_z + 2^8 \times 1.f_w) = fl(2^{116} \times (1.f_z + 0.\overbrace{0\ldots0}^{107 \text{ ceros}}\, 1f_w)) \\ &= 2^{116} \times 1.f_z = fl(10^{35}). \end{aligned}$$

Finalmente, $10^{35} + 511 - 10^{35}$ nos da $fl(fl(10^{35}) - fl(10^{35})) = 0$.

En base a lo explicado, dejamos como ejercicio realizar las operaciones señaladas en los Ejemplos 2 y 3 de la sección 1.2 para comprobar los resultados.

En lo que sigue vamos a establecer cotas del error que comete el computador en las operaciones aritméticas básicas.

1.5.1 Errores de redondeo en la suma.

Dados dos números x e y, vamos a estudiar el error relativo que se comete en el proceso de sumarlos. Para ello, al igual que en (1.1), podemos escribir:

$$fl(x) = x(1 + \varepsilon_x) \;\text{ y }\; fl(y) = y(1 + \varepsilon_y) \;\text{ con }\; |\varepsilon_x| \leq \varepsilon_M \;\text{ y }\; |\varepsilon_y| \leq \varepsilon_M. \quad (1.2)$$

Utilizando estas expresiones obtenemos

$$z = fl(x) + fl(y) = x + y + x\varepsilon_x + y\varepsilon_y.$$

Para concluir la suma debemos calcular $fl(z)$, que es la suma que proporciona el computador. De acuerdo a (1.1) tenemos

$$fl(z) = z(1+\varepsilon_z) = (x+y+x\varepsilon_x+y\varepsilon_y)(1+\varepsilon_z) = x+y+(x+y)\varepsilon_z+(x\varepsilon_x+y\varepsilon_y)(1+\varepsilon_z).$$

El error relativo será

$$\begin{aligned} E &= \frac{|(x+y) - fl(z)|}{|x+y|} = \frac{|(x+y)\varepsilon_z + (x\varepsilon_x + y\varepsilon_y)(1+\varepsilon_z)|}{|x+y|} \\ &\leq \frac{(|x|+|y|)|\varepsilon_z| + (|x||\varepsilon_x| + |y||\varepsilon_y|)(1+|\varepsilon_z|)}{|x+y|} \\ &\leq \frac{(|x|+|y|)\varepsilon_M + (|x|\varepsilon_M + |y|\varepsilon_M)(1+\varepsilon_M)}{|x+y|} = \frac{|x|+|y|}{|x+y|}(2+\varepsilon_M)\varepsilon_M. \end{aligned} \tag{1.3}$$

Tenemos que distinguir dos situaciones

$$\text{si signo(x)} = \text{signo(y)} \ \Rightarrow E \leq (2+\varepsilon_M)\varepsilon_M, \tag{1.4}$$

$$\text{si signo(x)} \neq \text{signo(y)} \ \Rightarrow E \leq \frac{|x|+|y|}{||x|-|y||}(2+\varepsilon_M)\varepsilon_M. \tag{1.5}$$

Por lo tanto, cuando los signos de x e y coinciden, es decir cuando sumamos, entonces el error relativo es esencialmente el doble de la precisión de la máquina. Sin embargo la resta puede proporcionar un fuerte error relativo si los números son de tamaño muy próximo: si $|x| \approx |y|$, entonces el denominador en (1.5) es próximo a 0, mientras el numerador puede ser grande, provocando un fuerte error relativo. Este error se llama *error de cancelación*. La denominación está justificada porque al calcular $fl(x)$ y $fl(y)$ se desechan decimales que son cruciales en el cómputo del resultado final. Veamos un ejemplo sobre la hipotética máquina que trabaja en base 10 con 4 dígitos de mantisa. Sean $x = 73.5645998$ e $y = 73.56451$. Entonces si calculamos sobre esta máquina $x - y$ obtenemos: $fl(fl(x) - fl(y)) = fl(10 \times 7.3565 - 10 \times 7.3565) = 0$. Este es un caso extremo en el que al restar dos números distintos el resultado es 0, produciéndose un enorme error relativo. Este tipo de error se produce en los cálculos realizados en la primera parte del Ejemplo 4 de la sección 1.2. En efecto, los signos de $\frac{(-30)^k}{k!}$ son positivos si k es par y negativos si k es impar, por lo que se producen restas que al principio son entre números muy grandes. Por ejemplo, para $k = 30$ tenemos que $\frac{(-30)^k}{k!} \approx 7.762070208797283 \times 10^{+11}$ y para $k = 31$ el valor es aproximadamente $-7.511680847223176 \times 10^{+11}$. Observemos que los decimales sextos y posteriores desparecen al redondear el número a la precisión de la máquina. Sin embargo, teniendo en cuenta que $\mathrm{e}^{-30} = 9.357622968840175 \times 10^{-14}$, estos decimales son fundamentales para calcular el resultado final. Sin embargo, cuando consideramos la aproximación

$$\mathrm{e}^{-30} = \frac{1}{\mathrm{e}^{30}} \approx \left(\sum_{k=0}^{100} \frac{30^k}{k!} \right)^{-1}$$

no se produce el error de cancelación y el resultado resulta muy preciso.

1.5.2 Errores de redondeo en la multiplicación.

Dados dos números x e y, vamos a estudiar el error relativo que se comete al calcular su producto. Utilizando las identidades (1.2), obtenemos

$$\begin{aligned} &z = fl(x) \cdot fl(y) = xy(1+\varepsilon_x)(1+\varepsilon_y), \\ &fl(z) = z(1+\varepsilon_z) = xy(1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_z) \\ &= xy(1+\varepsilon_x+\varepsilon_y+\varepsilon_z+\varepsilon_x\varepsilon_y+\varepsilon_x\varepsilon_z+\varepsilon_y\varepsilon_z+\varepsilon_x\varepsilon_y\varepsilon_z). \end{aligned}$$

Entonces el error relativo se acota como sigue

$$\begin{aligned} E = \frac{|xy - fl(z)|}{|xy|} = \frac{|xy||\varepsilon_x+\varepsilon_y+\varepsilon_z+\varepsilon_x\varepsilon_y+\varepsilon_x\varepsilon_z+\varepsilon_y\varepsilon_z+\varepsilon_x\varepsilon_y\varepsilon_z|}{|xy|} \\ = |\varepsilon_x+\varepsilon_y+\varepsilon_z+\varepsilon_x\varepsilon_y+\varepsilon_x\varepsilon_z+\varepsilon_y\varepsilon_z+\varepsilon_x\varepsilon_y\varepsilon_z| \leq (3+3\varepsilon_M+\varepsilon_M^2)\varepsilon_M. \end{aligned}$$

Por lo tanto, la cota del error relativo en la multiplicación es aproximadamente $3\varepsilon_M$.

1.5.3 Errores de redondeo en la división.

De nuevo tomamos dos números reales x e y y usamos las representaciones dadas en (1.2). Así obtenemos

$$\begin{aligned} z = \frac{fl(x)}{fl(y)} = \frac{x(1+\varepsilon_x)}{y(1+\varepsilon_y)}, \\ fl(z) = z(1+\varepsilon_z) = \frac{x}{y}\frac{(1+\varepsilon_x)(1+\varepsilon_z)}{(1+\varepsilon_y)}. \end{aligned}$$

Esto conduce a la siguiente cota del error relativo

$$\begin{aligned} E = \frac{\left|\frac{x}{y} - fl(z)\right|}{\left|\frac{x}{y}\right|} = \left|1 - \frac{(1+\varepsilon_x)(1+\varepsilon_z)}{(1+\varepsilon_y)}\right| \\ = \frac{|1+\varepsilon_y-(1+\varepsilon_x)(1+\varepsilon_z)|}{|1+\varepsilon_y|} = \frac{|\varepsilon_y-\varepsilon_x-\varepsilon_z-\varepsilon_x\varepsilon_z|}{1+\varepsilon_y} \leq \frac{3+\varepsilon_M}{1-\varepsilon_M}\varepsilon_M. \end{aligned}$$

Así el error de redondeo en la división es esencialmente el mismo que en la multiplicación: $3\varepsilon_M$.

1.6 Ejercicios

1. Ejecuta sobre el ordenador las siguientes operaciones y explica los resultados.

   (a) $1 + \varepsilon_M - 1$, $-1 + \varepsilon_M + 1$ y $(1 + 2\varepsilon_M) + \varepsilon_M - (1 + 2\varepsilon_M)$.

   (b) $0.1 \overbrace{+ \ldots +}^{10 \text{ veces}} 0.1$.

2. Dada la función

$$f(x) = (\mathrm{e}^x - 1)^2 + \left(\frac{1}{\sqrt{1 + x^2}} - 1 \right)^2,$$

   se desea aproximar el valor de $f'(1)$ mediante la fórmula

$$f'(1) \approx \frac{f(1 + h) - f(1)}{h}.$$

   Obtén dichas aproximaciones para los valores de $h = 10^{-k}$, para $k = 1, 2, \ldots, 17$. Explica los resultados. (NOTA: $f'(1) = 9.54865532212975587$.)

3. Dada la función

$$f(x) = \frac{4970x - 4923}{4970x^2 - 9799x + 4830},$$

   se desea aproximar $f''(1)$ mediante la fórmula

$$f''(1) \approx \frac{f(1 - h) - 2f(1) + f(1 + h)}{h^2}.$$

   Toma h como en el ejercicio anterior y explica los resultados. (NOTA: $f''(1) = 94$.)

4. Ejecuta las siguientes instrucciones MATLAB

   - A = randn(25,25);
   - A = A*A′;
   - v = eig(A);
   - l = max(v);
   - det(A-l*eye(25))

   Explica los resultados.

5. Reformula las siguientes expresiones de manera que su evaluación numérica sea estable:

   (a) $\dfrac{1}{1 + 2x} - \dfrac{1 - x}{1 + x}$ para $|x| \ll 1$.

   (b) $\dfrac{1 - \cos x}{x}$ para $x \neq 0$ y $|x| \ll 1$.

(c) $\sqrt{x+\dfrac{1}{x}}-\sqrt{x-\dfrac{1}{x}}$ para $x \gg 1$.

6. La función $\mathrm{tg}\frac{x}{2}$ puede ser evaluada empleando la siguiente fórmula

$$\mathrm{tg}\frac{x}{2} = \pm\sqrt{\frac{1-\cos x}{1+\cos x}}.$$

¿Es este método de evaluación numéricamente estable para $x \approx 0$ y $x \approx \frac{\pi}{2}$? Si es necesario, da alternativas numéricamente estables.

7. Sea $\{a_n\}_{n=1}^{\infty}$ la sucesión de números reales definida por la siguiente relación de recurrencia:

$$a_1 = 4; \qquad a_{n+1} = \frac{\sqrt{1+\frac{a_n^2}{2^{2(n+1)}}}-1}{a_n} 2^{2(n+1)+1}.$$

(a) Reescribe la recurrencia en una forma equivalente pero estable.

(b) Calcula a_{30} usando ambas fórmulas y comparalas. Explica los resultados.

8. Es bien conocida la siguiente fórmula para aproximar el número π:

$$\begin{cases} p_2 = 2\sqrt{2}, \\ p_{n+1} = 2^n\sqrt{2\left(1-\sqrt{1-\left(\frac{p_n}{2^n}\right)^2}\right)}, \quad n \geq 2. \end{cases}$$

(a) Explica por qué falla esta fórmula cuando se utiliza en el computador para aproximar π.

(b) Sea

$$r_{n+1} = 2\left(1-\sqrt{1-\left(\frac{p_n}{2^n}\right)^2}\right), \quad n \geq 2.$$

Prueba que

$$\begin{cases} r_3 = \dfrac{2}{2+\sqrt{2}}, \\ r_{n+1} = \dfrac{r_n}{2+\sqrt{4-r_n}}, \quad n \geq 3. \end{cases}$$

(c) Utiliza las fórmulas del apartado anterior para calcular $\{r_n\}$ y, escribiendo $p_{n+1} = 2^n\sqrt{r_{n+1}}$, obtén una aproximación de π con 14 dígitos.

9. Prueba que la sucesión de números reales $\{y_n\}_{n=0}^{\infty}$ definida por $y_n = \mathrm{e}^{-1}\int_0^1 \mathrm{e}^x x^n\,dx$ puede calcularse utilizando la recurrencia

$$\begin{cases} y_0 = \dfrac{\mathrm{e}-1}{\mathrm{e}} \\ y_{n+1} = 1-(n+1)y_n \quad \text{para} \quad n \geq 0. \end{cases}$$

(a) Prueba que $\lim_{n\to\infty} y_n = 0$.

(b) Usa la recurrencia anterior para calcular $\{y_n\}_{n=0}^{30}$. Interpreta los resultados.

(c) Comenzando con $y_{50} = 0$, utiliza la recurrencia hacia atrás para calcular $\{y_n\}_{n=0}^{30}$ y explica los resultados.

Capítulo 2

Resolución Aproximada de Ecuaciones Escalares No Lineales

2.1 Introducción

El objetivo de este capítulo es la descripción de los métodos numéricos más utilizados para resolver ecuaciones no lineales con una sola incógnita. Estas ecuaciones se escriben en la forma $\boxed{f(x) = 0}$, donde $f : \mathbb{R} \longrightarrow \mathbb{R}$ es una función. Usaremos métodos iterativos para resolver estas ecuaciones, comenzando las iteraciones en un punto x_0, a partir del cual se genera una sucesión $x_0, x_1, x_2, \ldots, x_k, \ldots$ El algoritmo es el proceso por el que se pasa de un punto x_k de la sucesión al siguiente punto x_{k+1}. Lo que se espera del algoritmo es que la sucesión generada $\{x_k\}_{k=0}^{\infty}$ sea convergente y su límite $\bar{x}$ sea solución de la ecuación, es decir, $f(\bar{x}) = 0$.

Además de la convergencia del algoritmo, es muy importante la velocidad de convergencia. Después daremos una definición precisa de lo que se entiende por velocidad de convergencia. Pero, de momento vamos a ver un ejemplo en el que usaremos dos algoritmos distintos para resolver la ecuación y en el que uno de ellos se muestra claramente más rápido que el otro. Vamos a resolver la ecuación $\boxed{x - \cos x = 0}$. Para ello implementamos primeramente el siguiente algoritmo:

$$\begin{cases} x_0 = 0.5, \\ x_{k+1} = \cos(x_k), \quad k = 0, 1, 2, \ldots \end{cases}$$

Entonces tenemos los siguientes resultados

$1 \leq k \leq 23$	$24 \leq k \leq 46$	$47 \leq k \leq 69$	$70 \leq k \leq 92$
0.877582561890373	0.739069001204012	0.739085135039528	0.739085133214954
0.639012494165259	0.739095999835755	0.739085131986245	0.739085133215300
0.802685100682335	0.739077813285175	0.739085134042973	0.739085133215067
0.694778026788006	0.739090063988251	0.739085132657536	0.739085133215224
0.768195831282016	0.739081811778109	0.739085133590783	0.739085133215118
0.719165445942419	0.739087370571036	0.739085132962137	0.739085133215189
0.752355759421527	0.739083626103480	0.739085133385601	0.739085133215141
0.730081063137823	0.739086148422879	0.739085133100350	0.739085133215174
0.745120341351440	0.739084449358649	0.739085133292498	0.739085133215152
0.735006309014843	0.739085593868961	0.739085133163065	0.739085133215167
0.741826522643246	0.739084822913141	0.739085133250253	0.739085133215157
0.737235725442231	0.739085342238298	0.739085133191522	0.739085133215163
0.740329651878263	0.739084992414645	0.739085133231084	0.739085133215159
0.738246238332233	0.739085228060075	0.739085133204435	0.739085133215162
0.739649962769661	0.739085069326482	0.739085133222386	0.739085133215160
0.738704539356983	0.739085176251341	0.739085133210294	0.739085133215161
0.739341452281210	0.739085104225471	0.739085133218439	0.739085133215160
0.738912449332103	0.739085152742964	0.739085133212952	0.739085133215161
0.739201444135799	0.739085120060997	0.739085133216648	0.739085133215160
0.739006779780813	0.739085142075963	0.739085133214159	0.739085133215161
0.739137910762293	0.739085127246417	0.739085133215836	0.739085133215161
0.739049580595209	0.739085137235778	0.739085133214706	0.739085133215161
0.739109081420527	0.739085130506825	0.739085133215467	0.739085133215161

Se observa la convergencia del método, pero no es hasta la iteración 89 que se estabiliza la sucesión. Es importante tener en cuenta que no se ha encontrado la solución exacta, que es un número irracional y, por lo tanto, tiene infinitos decimales. Simplemente hemos encontrado los primeros 15 decimales. En un método iterativo convergente, la solución de la ecuación es el límite de la sucesión, pero el método debe detenerse en algún momento, no se pueden realizar infinitas iteraciones. Nos detendremos cuando se haya alcanzado la precisión deseada. Teniendo en cuenta que la solución exacta puede poseer infinitos decimales, solamente podemos aspirar a obtener una aproximación de la solución. Sin embargo, en la mayoría de los casos, unos pocos decimales de precisión es suficiente.

Ahora implementamos un segundo algoritmo para resolver la ecuación anterior:

$$\begin{cases} x_0 = 0.5, \\ x_{k+1} = x_k - \dfrac{x_k - \cos(x_k)}{1 + \operatorname{sen}(x_k)}, \quad k = 0, 1, 2, \ldots \end{cases}$$

Ahora tenemos

0.755222417105636
0.739141666149879
0.739085133920807
0.739085133215161
0.739085133215161

Vemos cómo en la cuarta iteración ya tenemos la aproximación que con el algoritmo anterior nos costó 89 iteraciones. Por lo tanto, la convergencia es muy importante, pero la velocidad de convergencia no lo es menos. Estas dos cuestiones serán estudiadas en la próxima sección. En la sección 2.3 estudiaremos el método de Newton para resolver estas ecuaciones. También veremos como combinar el método Newton con el de bisección para elaborar un algoritmo robusto y eficiente. En la sección 2.4 veremos el método de la secante y estudiaremos su implementación combinado con el método de bisección. Finalmente, en la sección 2.5 estudiaremos el método de Maehly para resolver ecuaciones polinómicas.

El lector puede consultar los textos [9] y [12] para un mayor conocimiento sobre este tema.

2.2 Sobre la Convergencia de un Algoritmo

En esta sección vamos a considerar una sucesión de números reales $\{x_k\}_{k=0}^{\infty}$ generada por un algoritmo a partir de un punto inicial x_0. Queremos que esta sucesión converja hacia una solución $\bar{x}$ de la ecuación $\boxed{f(x) = 0}$. Se distinguen dos tipos de convergencia que pasamos a definir.

DEFINICIÓN 2.1 *Se dice que un algoritmo es globalmente convergente si la sucesión generada es convergente hacia una solución de la ecuación cualquiera que sea el punto inicial x_0 que tomemos. El algoritmo se dice que es localmente convergente si existe $\varepsilon > 0$ tal que la sucesión converge hacia la solución $\bar{x}$ siempre que $x_0 \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$.*

Parece razonable buscar algoritmos que sean globalmente convergentes, pero no lo es. En la resolución de la ecuación $x - \cos x = 0$ utilizamos un primer algoritmo que es globalmente convergente, sin embargo su convergencia es muy lenta. El segundo algoritmo que utilizamos es localmente convergente, pero su velocidad de convergencia es muy superior. La clave para resolver la ecuación está en analizar la función f para determinar un intervalo pequeño $[\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ en el que se encuentre la solución $\bar{x}$ e implementar un algoritmo de velocidad de convergencia rápida comenzando en un punto x_0 de dicho intervalo. El asunto es cómo conseguir este intervalo y cómo de pequeño debe ser. Esto lo veremos en la próxima sección. Seguidamente vamos a definir la velocidad de convergencia de una sucesión.

DEFINICIÓN 2.2 *Supongamos que la sucesión* $\{x_k\}_{k=0}^{\infty}$ *converge hacia* $\bar{x}$*. Se dice que la velocidad (o el orden) de convergencia es* p *con* $1 \leq p < \infty$ *si existe una constante* C_p *y un entero* k_0 *tal que*

$$|x_{k+1} - \bar{x}| \leq C_p |x_k - \bar{x}|^p \quad \forall k \geq k_0. \tag{2.1}$$

- *Si* $p = 1$ *y* $C_p \geq 1$*, se dice que la convergencia es sublineal.*
- *Si* $p = 1$ *y* $C_p < 1$*, se dice que la convergencia es lineal.*
- *Si* $p = 2$*, la convergencia se dice cuadrática y cúbica si* $p = 3$*.*

Cuando la convergencia es lineal, adquirir un decimal de precisión en la solución requiere típicamente varias iteraciones, su número dependiendo de si C_1 es próximo a 0 o a 1. Sin embargo, cuando la convergencia es cuadrática el número de decimales de precisión típicamente se dobla de una iteración a otra. Esto se puede observar en las tablas de resultados de los dos algoritmos usados en la introducción. Si la convergencia es cúbica, entonces se triplica el número de decimales de precisión de una iteración a otra.

Entre la convergencia lineal y la de orden $p > 1$ con p arbitrariamente próximo a 1 hay un tipo de convergencia intermedio que tiene su interés, y que definimos a continuación.

DEFINICIÓN 2.3 *Supongamos que la sucesión* $\{x_k\}_{k=0}^{\infty}$ *converge hacia* $\bar{x}$*. Se dice que la velocidad de convergencia es superlineal si existe una sucesión de números reales* $\{\varepsilon_k\}_{k=0}^{\infty}$ *satisfaciendo las siguientes condiciones*

$$\lim_{k\to\infty} \varepsilon_k = 0 \quad y \quad |x_{k+1} - \bar{x}| \leq \varepsilon_k |x_k - \bar{x}| \quad \forall k \geq 0. \tag{2.2}$$

Obviamente, la convergencia superlineal es más rápida que la simplemente lineal. Sin embargo, si $p > 1$ y $\{x_k\}_{k=0}^{\infty}$ tiene velocidad de convergencia p, entonces $\{x_k\}_{k=0}^{\infty}$ converge superlinealmente. En efecto, es suficiente tomar $\varepsilon_k = C_p |x_k - \bar{x}|^{p-1}$ para cada $k \geq k_0$ y, por ejemplo, $\varepsilon_k = \frac{|x_{k+1} - \bar{x}|}{|x_k - \bar{x}|}$ si $0 \leq k < k_0$. Entonces usando (2.1) y la definición de ε_k se tiene:

$$\lim_{k\to\infty} x_k = \bar{x} \text{ y } p > 1 \Rightarrow \lim_{k\to\infty} \varepsilon_k = 0,$$
$$|x_{k+1} - \bar{x}| \leq C_p |x_k - \bar{x}|^p = \varepsilon_k |x_k - \bar{x}| \quad \forall k \geq k_0.$$

Veamos la siguiente propiedad importante de las sucesiones que convergen superlinealmente.

TEOREMA 2.4 *Supongamos que* $\{x_k\}_{k=0}^{\infty}$ *converge superlinealmente hacia* $\bar{x}$*, entonces se cumple*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_k|}{|x_k - \bar{x}|} = 1. \tag{2.3}$$

Demostración. Sabemos que para cada par de números reales a y b se tiene que $||a|-|b|| \leq |a+b|$. Aplicando esta desigualdad con $a = x_{k+1}-x_k$ y $b = x_k-\bar{x}$ y utilizando (2.2), se obtiene

$$\Big|\frac{|x_{k+1}-x_k|}{|x_k-\bar{x}|}-1\Big| = \frac{\big||x_{k+1}-x_k|-|x_k-\bar{x}|\big|}{|x_k-\bar{x}|} \leq \frac{|x_{k+1}-\bar{x}|}{|x_k-\bar{x}|} \leq \varepsilon_k \overset{k\to\infty}{\to} 0. \qquad \square$$

Analicemos la importancia de la propiedad (2.3). Cualquier algoritmo destinado a calcular una aproximación de la solución de una ecuación debe tener un test de parada que finalice las iteraciones. Este test de parada debe recoger la precisión que se desee alcanzar. La forma natural y habitual para finalizar el algoritmo es comparar dos iteraciones consecutivas y examinar el grado de coincidencia de sus dígitos. Más concretamente, dado un número $\varepsilon > 0$ que indica la precisión deseada se realizará el test $|x_{k+1} - x_k| < \varepsilon$ para decidir si la precisión es satisfactoria. Esto tiene sentido cuando la sucesión converge superlinealmente porque, de acuerdo a (2.3), $|x_k - \bar{x}| \approx |x_{k+1} - x_k|$. Además $|x_{k+1} - \bar{x}| \leq \varepsilon_k|x_k - \bar{x}| \approx \varepsilon_k|x_{k+1} - x_k| < \varepsilon_k\varepsilon$ con $\varepsilon_k \to 0$ cuando $k \to \infty$. Sin embargo, cuando la convergencia es lineal, entonces puede ocurrir que $|x_{k+1}-x_k|$ sea pequeño y sin embargo x_{k+1} esté muy lejos de la solución.

2.3 Método de Newton

Un algoritmo destinado a resolver la ecuación $\boxed{f(x) = 0}$ debe establecer el mecanismo para pasar de la iteración x_k a x_{k+1}. La idea del método de Newton es hacer una aproximación lineal de $f(x)$ en el punto x_k usando la derivada de f en dicho punto: $f(x) \approx f(x_k) + f'(x_k)(x - x_k)$ y tomar x_{k+1} como la solución de la ecuación lineal $f(x_k) + f'(x_k)(x - x_k) = 0$, lo que proporciona

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{2.4}$$

Ésta es la fórmula del Método de Newton. Podemos interpretar geométricamente este algoritmo. Dado el punto x_k, se calcula la recta tangente a la curva $y = f(x)$ en el punto $(x_k, f(x_k))$: $y = f(x_k) + f'(x_k)(x - x_k)$. Seguidamente se calcula el punto de corte de la tangente con el eje OX y éste es el punto x_{k+1}: $y = 0 \Rightarrow f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0$, lo que conduce a la fórmula (2.4).

El método de Newton es el segundo algoritmo utilizado en la introducción para resolver la ecuación $x - \cos x = 0$. Notemos que el algoritmo (2.4) no se puede implementar si $f'(x_k) = 0$. Sin embargo, tenemos el siguiente resultado.

TEOREMA 2.5 *Sean $r > 0$ y $\bar{x} \in \mathbb{R}$ tal que $f(\bar{x}) = 0$. Supongamos que f es dos veces derivable en el intervalo $[\bar{x} - r, \bar{x} + r]$, que f'' es una función continua en dicho intervalo y que $f'(\bar{x}) \neq 0$. Entonces, existe $\varepsilon > 0$ con $\varepsilon \leq r$ tal que si $x_0 \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ se cumple que el método de Newton (2.4) está bien definido, la sucesión $\{x_k\}_{k=0}^{\infty}$ generada por el algoritmo está contenida en el intervalo $[\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ y converge cuadráticamente hacia $\bar{x}$.*

OBSERVACIÓN 2.6 *La afirmación indicando que el método de Newton* (2.4) *está bien definido implica que la derivada* $f'(x_k)$ *existe y no se anula para cada* k. *Una consecuencia obvia del teorema es que el método de Newton converge localmente si* $f'(\bar{x}) \neq 0$. *Además, la convergencia es cuadrática.*

Demostración del Teorema 2.4. Puesto que por hipótesis $f'(\bar{x}) \neq 0$ y la función $f' : [\bar{x} - r, \bar{x} + r] \longrightarrow \mathbb{R}$ es continua, entonces existe un número $\varepsilon_0 > 0$ con $\varepsilon_0 \leq r$ tal que $f'(x) \neq 0$ para cada punto $x \in [\bar{x} - \varepsilon_0, \bar{x} + \varepsilon_0]$. Ahora definimos la función $g : [\bar{x} - \varepsilon_0, \bar{x} + \varepsilon_0] \longrightarrow \mathbb{R}$ por

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Dado que $f'(x) \neq 0$ para cada punto del intervalo indicado, g está bien definida y es derivable, siendo la derivada una función continua dada por la fórmula

$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}.$$

Además, es obvio que las iteraciones definidas en (2.4) pueden escribirse como $x_{k+1} = g(x_k)$. Puesto que $f(\bar{x}) = 0$, entonces se tiene que $g(\bar{x}) = \bar{x}$ y $g'(\bar{x}) = 0$. Utilizando ahora la continuidad de g' y el hecho de que $g'(\bar{x}) = 0$, deducimos la existencia de $\varepsilon > 0$ con $\varepsilon \leq \varepsilon_0$ tal que $|g'(x)| \leq \frac{1}{2}$ para cada punto $x \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$. Vamos a elegir $x_0 \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ y probaremos primeramente que la sucesión $\{x_k\}_{k=0}^{\infty}$ está bien definida y contenida en el intervalo $[\bar{x} - \varepsilon, \bar{x} + \varepsilon]$. Para ello demostraremos que si $x_k \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$, entonces x_{k+1} también pertenece a este intervalo. Como x_0 ha sido elegido en este intervalo, entonces tendremos que la sucesión entera pertenece al intervalo. Dado x_k en el intervalo, utilizando el teorema del valor medio se tiene

$$|x_{k+1} - \bar{x}| = |g(x_k) - g(\bar{x})| = |g'(\xi_k)(x_k - \bar{x})| \leq \frac{1}{2}|x_k - \bar{x}| \leq \frac{\varepsilon}{2},$$

lo que demuestra que $x_{k+1} \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$.

A continuación probamos que $\lim_{k\to\infty} x_k = \bar{x}$. Reiterando el uso del teorema del valor medio y el hecho de que $|g'(x)| < \frac{1}{2}$ para cada $x \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$, tenemos

$$|x_{k+1} - \bar{x}| \leq \frac{1}{2}|x_k - \bar{x}| \leq \frac{1}{2^2}|x_{k-1} - \bar{x}| \leq \frac{1}{2^3}|x_{k-2} - \bar{x}| \leq \ldots \leq \frac{1}{2^{k+1}}|x_0 - \bar{x}| \overset{k\to\infty}{\to} 0.$$

Finalmente probamos que la convergencia es cuadrática. Realizando un desarrollo de Taylor de $f(\bar{x})$ entorno de x_k obtenemos

$$0 = f(\bar{x}) = f(x_k) + f'(x_k)(\bar{x} - x_k) + \frac{1}{2}f''(\eta_k)(\bar{x} - x_k)^2$$

para algún punto η_k comprendido entre $\bar{x}$ y x_k. Como $f'(x) \neq 0$ para cada $x \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ se deduce de la identidad de arriba que

$$x_k - \bar{x} - \frac{f(x_k)}{f'(x_k)} = \frac{f''(\eta_k)}{2f'(x_k)}(\bar{x} - x_k)^2.$$

Usando (2.4) tenemos

$$|x_{k+1} - \bar{x}| = \frac{|f''(\eta_k)|}{2|f'(x_k)|}|x_k - \bar{x}|^2 \leq C_2|\bar{x} - x_k|^2 \quad \text{para } k = 0, 1, 2, \ldots,$$

donde

$$C_2 = \sup_{x,y\in[\bar{x}-\varepsilon,\bar{x}+\varepsilon]} \frac{|f''(x)|}{2|f'(y)|} = \frac{\sup_{x\in[\bar{x}-\varepsilon,\bar{x}+\varepsilon]} |f''(x)|}{2\inf_{y\in[\bar{x}-\varepsilon,\bar{x}+\varepsilon]} |f'(y)|}.$$

Puesto que f' no se anula en el intervalo $[\bar{x} - \varepsilon, \bar{x} + \varepsilon]$, entonces el denominador $\inf_{y\in[\bar{x}-\varepsilon,\bar{x}+\varepsilon]} |f'(y)| > 0$ y, por lo tanto, $C_2 < \infty$ y la convergencia es cuadrática. □

La convergencia local del método de Newton aparece como un inconveniente del método porque no sabemos a priori cómo de cerca debe estar x_0 de la solución. Para evitar este problema, el método de Newton puede programarse en combinación con el método de biscección que describimos seguidamente.

2.3.1 Método de Bisección

Aquí suponemos que $a < b$ son dos números reales y $f : [a, b] \longrightarrow \mathbb{R}$ es una función continua tal que el signo de $f(a)$ es distinto de $f(b)$, más precisamente $f(a)f(b) < 0$. Entonces, aplicando el Teorema de Bolzano, sabemos que existe al menos un punto $\bar{x} \in (a, b)$ tal que $f(\bar{x}) = 0$. Se trata de calcular uno de estos puntos. Para ello comenzamos el proceso iterativo tomando $a_0 = a$, $b_0 = b$, y $x_0 = \frac{a_0+b_0}{2}$. Así, en cada iteración del proceso tendremos un intervalo (a_k, b_k) con $f(a_k)f(b_k) < 0$ y un punto $x_k = \frac{a_k+b_k}{2}$. Entonces x_{k+1} y (a_{k+1}, b_{k+1}) se calculan como sigue

$$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k, x_k) & \text{si } f(a_k)f(x_k) < 0, \\ (x_k, b_k) & \text{en caso contrario,} \end{cases} \qquad x_{k+1} = \frac{a_{k+1} + b_{k+1}}{2}.$$

Si en alguna iteración sucede que $f(a_k)f(x_k) = 0$ o $f(b_k)f(x_k) = 0$, entonces x_k es la solución buscada. Puesto que x_k es el punto medio del intervalo $[a_k, b_k]$ y $\bar{x} \in (a_k, b_k)$, y como la longitud de cada intervalo es la mitad del anterior, se tiene

$$|x_k - \bar{x}| < \frac{1}{2}(b_k - a_k) = \frac{1}{2^2}(b_{k-1} - a_{k-1}) = \ldots = \frac{1}{2^{k+1}}(b - a) \overset{k\to\infty}{\to} 0.$$

Esto demuestra la convergencia del método de bisección. Sin embargo, esta convergencia es muy lenta, sublineal en general. Vamos a aplicar el método de bisección para resolver la ecuación $\boxed{x - \cos x = 0}$. Tomamos $a = 0$, $b = 1$ y $f : [a, b] \longrightarrow \mathbb{R}$ definida por $f(x) = x - \cos x$. Entonces, se tiene

$$f(a) = -1, \ f(b) = 0.459697694131860 \Rightarrow f(a)f(b) < 0.$$

Iniciamos las iteraciones:
- $a_0 = a; b_0 = b; x_0 = \frac{a+b}{2}; [a_0, x_0, b_0] = [0, 0.5, 1]$
- $f(x_0)f(a_0) > 0 \Rightarrow a_1 = x_0; b_1 = b_0; x_1 = \frac{a_1+b_1}{2}; [a_1, x_1, b_1] = [0.5, 0.75, 0.1]$

- $f(x_1)f(a_1) < 0 \Rightarrow a_2 = a_1; b_2 = x_1; x_2 = \frac{a_2+b_2}{2}; [a_2, x_2, b_2] = [0.5, 0.625, 0.75]$
- $f(x_2)f(a_2) > 0 \Rightarrow a_3 = x_2; b_3 = b_2; x_3 = \frac{a_3+b_3}{2}; [a_3, x_3, b_3] = [0.625, 0.6875, 0.75]$
- $f(x_3)f(a_3) > 0 \Rightarrow a_4 = x_3; b_4 = b_3; x_4 = \frac{a_4+b_4}{2}; [a_4, x_4, b_4] = [0.6875, 0.71875, 0.75]$
- $f(x_4)f(a_4) > 0 \Rightarrow a_5 = x_4; b_5 = b_4; x_5 = \frac{a_5+b_5}{2}; [a_5, x_5, b_5] = [0.71875, 0.734375, 0.75]$
- $f(x_5)f(a_5) > 0 \Rightarrow a_6 = x_5; b_6 = b_5; x_6 = \frac{a_6+b_6}{2}; [a_6, x_6, b_6] = [0.734375, 0.7421875, 0.75]$
- $f(x_6)f(a_6) < 0 \Rightarrow a_7 = a_6; b_7 = x_6; x_7 = \frac{a_7+b_7}{2}; [a_7, x_7, b_7] = [0.734375, 0.73828125, 0.7421875]$

Como vemos, después de 7 iteraciones la aproximación obtenida es $x_7 = 0.73828125$, lejos de la solución que es $\bar{x} \approx 0.739085133215161$. Solamente hemos obtenido 2 decimales de precisión en 7 iteraciones, cuando el método de Newton nos proporcionó 15 dígitos de precisión en 4 iteraciones. Observemos también que x_1 está bastante más cerca de la solución que x_2. La razón es que el método de bisección no presta atención al tamaño de $f(x_k)$, solamente mira su signo. Esto puede producir situaciones en las que x_k está muy cerca de la solución y sin embargo x_{k+1} está lejos. Sin embargo, describimos aquí el método de bisección porque el resultado de combinar el método de Newton con el de bisección resulta altamente satisfactorio. Veamos cómo podemos combinar ambos métodos.

2.3.2 Método de Newton combinado con Bisección

Este algoritmo se inicia tomando un intervalo $[a, b]$ en el que $f(a)f(b) < 0$. Como consecuencia, la ecuación $\boxed{f(x) = 0}$ posee al menos una solución en dicho intervalo. Además elegimos un punto $x_0 \in [a, b]$. Al igual que en el método de bisección, en cada iteración vamos a contar con un intervalo $[a_k, b_k]$ con $f(a_k)f(b_k) < 0$ y un punto $x_k \in [a_k, b_k]$. Para calcular x_{k+1} siempre se intentará el método de Newton, aceptando el punto que se obtiene si pertenece al intervalo $[a_k, b_k]$ en el que se encuentra la solución. Si no pertenece a dicho intervalo, se rechaza el punto y se aplica el método de bisección. Otro motivo para no aplicar el método de Newton se produce cuando el denominador $f'(x_k)$ es muy pequeño, o todavía mejor cuando el cociente $\frac{f(x_k)}{f'(x_k)}$ es muy grande. Más concretamente, si $\left|\frac{f(x_k)}{f'(x_k)}\right| > \frac{1}{\varepsilon_M}$, entonces no aplicamos el método de Newton y usamos el método de bisección. En la práctica, para evitar un overflow la condición anterior no se implementa como está escrito. Intentaremos el método de Newton si $|f'(x_k)| > \varepsilon_M |f(x_k)|$, lo que es equivalente pero evita problemas sobre el computador.

Como en todo algoritmo iterativo, debemos establecer un test de parada. La parada depende de la pecisión que se desee en la solución. Esta precisión vendrá dada por un número pequeño $\varepsilon > 0$. El algoritmo debe cumplir dos condiciones para pararse. La primera es que la diferencia o diferencia relativa al tamaño entre dos iteraciones consecutivas sea menor que la precisión deseada, indicada por ε. El test correspondiente es $|x_{k+1} - x_k| < \varepsilon \max\{1, |x_{k+1}|\}$. Este test de parada es una combinación del error relativo y absoluto. Si $|x_{k+1}| \leq 1$, entonces se mira el error absoluto. En caso contrario se examina el error relativo. En definitiva, miramos el número de dígitos de precisión. Por otro lado, se debe

tener en cuenta que se está buscando la solución de la ecuación $\boxed{f(x)=0}$. Por lo tanto, una exigencia para parar el algoritmo será también que $|f(x_{k+1})|$ sea pequeño, lo que testaremos en la forma: $|f(x_{k+1})|<\varepsilon$. Si la primera condición del test de parada no se cumple, pero $|f(x_{k+1})|$ es más pequeño que la precisión con la que el computador puede evaluar la función f, entonces debemos detener el algoritmo. Esto se decide con el test $|f(x_{k+1})|<\varepsilon_f$, donde ε_f denota la precisión con la que se evalúa $f(x)$. Con mucha frecuencia este número es desconocido. Para funciones sencillas un valor habitual es tomar $\varepsilon_f=\varepsilon_M^{3/4}$.

Resumiendo, el algoritmo es el siguiente:

Algoritmo de Newton-Bisección

1. Tomar $x_0\in[a,b]$; $k=0$; $\varepsilon>0$;
   - Si $f(a)f(x_0)<0\Rightarrow a_0=a,\ b_0=x_0$.
   - Si no $\Rightarrow a_0=x_0,\ b_0=b$.
2.
   - Si $|f'(x_k)|>\varepsilon_M|f(x_k)|\Rightarrow y_{k+1}=x_k-\dfrac{f(x_k)}{f'(x_k)}$.
     - Si $y_{k+1}\in[a_k,b_k]\Rightarrow x_{k+1}=y_{k+1}$.
     - Si no $\Rightarrow x_{k+1}=\dfrac{a_k+b_k}{2}$.
   - Si no $\Rightarrow x_{k+1}=\dfrac{a_k+b_k}{2}$.
3.
   - Si $f(a_k)f(x_{k+1})<0\Rightarrow a_{k+1}=a_k,\ b_{k+1}=x_{k+1}$.
   - Si no $\Rightarrow a_{k+1}=x_{k+1},\ b_{k+1}=b_k$.
4.
   - Si $|x_{k+1}-x_k|<\varepsilon\max\{1,|x_{k+1}|\}$ y $|f(x_{k+1})|<\varepsilon\Rightarrow$ STOP.
   - Si $|f(x_{k+1})|<\varepsilon_f\Rightarrow$ STOP.
   - Si no $\Rightarrow k\to k+1$ y volver al paso 2.

2.4 Método de la Secante

Hay ocasiones en las que la derivada de la función f no puede calcularse o resulta computacionalmente costosa. Para evitar esta circunstancia se utiliza el algoritmo de la secante. Vimos la interpretación geométrica del método de Newton, donde x_{k+1} es el punto de corte del la tangente a la curva $y=f(x)$ en el punto $(x_k,f(x_k))$ con el eje OX. Es bien sabido que la secante a la curva en los puntos $(x_{k-1},f(x_{k-1}))$ y $(x_k,f(x_k))$ se aproxima a la tangente cuando

$x_{k-1} \approx x_k$. El Teorema del Valor Medio nos da una explicación clara de este asunto. En efecto, la ecuación de la recta secante es $y = f(x_k) + m_k(x - x_k)$, donde $m_k = \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$ es su pendiente. La ecuación de la recta tangente es $y = f(x_k)+f'(x_k)(x-x_k)$, donde $f'(x_k)$ es la pendiente de la tangente. Usando el Teorema del Valor Medio se obtiene para algún punto ξ_k situado entre x_{k-1} y x_k:

$$m_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} = \frac{f'(\xi_k)(x_k - x_{k-1})}{x_k - x_{k-1}} = f'(\xi_k).$$

Cuanto más próximo esté x_{k-1} a x_k, más próximo estará $m_k = f'(\xi_k)$ a $f'(x_k)$, y más próximas estarán las rectas secante y tangente. Consecuentemente, más próximos estarán los puntos de corte con el eje OX.

El método de la secante proporciona el punto x_{k+1} como solución de la ecuación $f(x_k) + m_k(x - x_k) = 0$. Entonces el método se formula como sigue

$$x_{k+1} = x_k - \frac{f(x_k)}{m_k} \quad \text{con} \quad m_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}. \tag{2.5}$$

OBSERVACIÓN 2.7

A diferencia del método de Newton, el método de la secante requiere dos puntos para iniciar las iteraciones: x_0 y x_1. Esto en la práctica no es ninguna restricción porque, al igual que en el método de Newton, buscaremos un intervalo $[a, b]$ con $f(a)f(b) < 0$ y entonces tomaremos $x_0 = a$ y $x_1 = b$ lo que nos proporciona un punto $x_2 \in (a, b)$. En efecto, la recta secante $y = f(x_1) + m_0(x - x_1)$ con $m_0 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$ pasa por los puntos $(x_0, f(x_0))$ y $(x_1, f(x_1))$ con $f(x_0)f(x_1) < 0$, por lo tanto el punto de corte de la recta con el eje OX está entre x_0 y x_1.

Las propiedades de convergencia del método de Newton se establecieron en el Teorema 2.5. A continuación presentamos sin demostración el teorema de convergencia correspondiente al método de la secante.

TEOREMA 2.8 *Sean $r > 0$ y $\bar{x} \in \mathbb{R}$ tal que $f(\bar{x}) = 0$. Supongamos que f es dos veces derivable en el intervalo $[\bar{x} - r, \bar{x} + r]$, que f'' es una función continua en dicho intervalo y que $f'(\bar{x}) \neq 0$. Entonces, existe $\varepsilon > 0$ y $\varepsilon \leq r$ tal que si $x_0, x_1 \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ con $x_0 \neq x_1$ se cumple que la sucesión $\{x_k\}_{k=0}^{\infty}$ generada por el algoritmo (2.5) está contenida en el intervalo $[\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ y converge hacia $\bar{x}$ con velocidad de convergencia $p = \frac{1+\sqrt{5}}{2} \approx 1.618033988749895$.*

Del teorema anterior se desprende que el método de la secante, al igual que el método de Newton, converge localmente. Por ello, resulta muy conveniente programarlo en combinación con el método de bisección. Describimos a continuación el algoritmo correspondiente.

Al igual que en el algoritmo de Newton combinado con bisección, partiremos de un intervalo $[a, b]$ tal que $f(a)f(b) < 0$. También fijaremos la precisión requerida en la solución a través de los parámetros $\varepsilon > 0$ y $\varepsilon_f > 0$. Entonces, procedemos como sigue.

Algoritmo de la Secante-Bisección

1. Tomar $x_0 = a$; $x_1 = b$; $a_1 = a$; $b_1 = b$; $\varepsilon > 0$;
2. Calcular
$$m_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad \text{y} \quad x_2 = x_1 - \frac{f(x_1)}{m_0}.$$
3. $k = 2$.
   - Si $f(x_2)f(a_1) < 0 \Rightarrow a_2 = a_1,\ b_2 = x_2,\ x_1 = x_0$.
   - Si no $\Rightarrow a_2 = x_2,\ b_2 = b_1$.
4. $m_k = \dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$.
   - Si $|m_k| > \varepsilon_M |f(x_k)| \Rightarrow y_{k+1} = x_k - \dfrac{f(x_k)}{m_k}$.
     - Si $y_{k+1} \in [a_k, b_k] \Rightarrow x_{k+1} = y_{k+1}$.
     - Si no $\Rightarrow x_{k+1} = \dfrac{a_k + b_k}{2}$.
   - Si no $\Rightarrow x_{k+1} = \dfrac{a_k + b_k}{2}$.
5.
   - Si $f(a_k)f(x_{k+1}) < 0 \Rightarrow a_{k+1} = a_k,\ b_{k+1} = x_{k+1}$.
   - Si no $\Rightarrow a_{k+1} = x_{k+1},\ b_{k+1} = b_k$.
6.
   - Si $|x_{k+1} - x_k| < \varepsilon \max\{1, |x_{k+1}|\}$ y $|f(x_{k+1})| < \varepsilon \Rightarrow$ STOP.
   - Si $|f(x_{k+1})| < \varepsilon_f \Rightarrow$ STOP.
   - Si no $\Rightarrow k \to k + 1$ y volver al paso 4.

2.5 Cálculo de las Raíces de un Polinomio

El objetivo de esta sección es mostrar un método para calcular todas las raíces reales y complejas de un polinomio: $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$, donde los coeficientes a_j pueden ser números reales o complejos. En definitiva se trata de hallar todas las soluciones de la ecuación

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = 0.$$

El teorema fundamental del álgebra nos dice que esta ecuación posee exactamente n raíces en el campo de los números complejos $\mathbb{C}$, donde cada una de ellas se cuenta con su multiplicidad. Por ejemplo, la ecuación $x^4 - 2x^3 + 2x^2 - 2x + 1 =$

0 posee exactamente 4 raíces: $\xi_1 = i$, $\xi_2 = -i$, $\xi_3 = 1$ y $\xi_4 = 1$. Es decir, la raíz 1 es doble.

Otro aspecto importante es el siguiente: si los coeficientes $\{a_j\}_{j=0}^n$ del polinomio son todos ellos números reales, y si $\xi = \alpha + \beta i$ es una raíz compleja ($\beta \neq 0$), entonces el conjugado de ξ ($\bar{\xi} = \alpha - \beta i$) también es raíz del polinomio. Esta propiedad no se cumple si alguno de los coeficientes a_j es un número complejo.

Aunque hemos descrito el método de Newton para calcular las soluciones reales de una ecuación $\boxed{f(x) = 0}$, siendo f una función real de variable real, lo cierto es que el método también es aplicable para calcular las soluciones de ecuaciones complejas. Las propiedades del método de Newton señaladas en el Teorema 2.5 también son válidas cuando $f : \mathbb{C} \longrightarrow \mathbb{C}$ es una función compleja. Por otro lado, teniendo en cuenta que la derivada de un polinomio p es sencilla de calcular ($p'(x) = a_1 + 2a_2x + 3a_3x^2 + \ldots + na_nx^{n-1}$), el método de Newton resulta adecuado para calcular las raíces del polinomio p. Ahora bien, imaginemos que por aplicación del método de Newton hemos obtenido una raíz ξ_1 del polinomio p y deseamos calcular una segunda raíz: ¿cómo evitar que las iteraciones del método de Newton vuelvan a converger hacia el punto ξ_1? La primera idea que uno puede tener consiste en tomar un punto de partida x_0 diferente al anterior. Sin embargo, esto no garantiza que el algoritmo converja hacia otro punto distinto de ξ_1. Además, la situación se complica cuando ya hemos calculado un determinado número de raíces $\{\xi_1, \xi_2, \ldots, \xi_m\}$ con $m < n$. Entonces, la probabilidad de aplicar el algoritmo y volver hacia una de esas raíces es alta.

Una segunda idea para evitar la convergencia hacia una raíz ya calculada es la siguiente. Dado que todo polinomio p puede factorizarse en la forma $p(x) = a_n(x - \xi_1)(x - \xi_2)\ldots(x - \xi_n)$, donde $\{\xi_1, \xi_2, \ldots, \xi_n\}$ son sus raíces, una vez calculada una raíz ξ_1 podemos hacer el cociente

$$p_1(x) = \frac{p(x)}{x - \xi_1} = a_n(x - \xi_2)(x - \xi_3)\ldots(x - \xi_n).$$

Ahora p_1 ya no contiene la raíz ξ_1 y podríamos aplicar el método de Newton a este polinomio. Este procedimiento es conocido como deflación. Sin embargo, hay que tener en cuenta que el método de Newton nos proporciona solamente una aproximación $\tilde{\xi}_1$ de ξ_1 por lo que el cociente $\frac{p(x)}{x - \tilde{\xi}_1}$ ya no nos proporciona el resultado anterior, siendo incluso el resto de la división diferente de 0. Pero, aun si la raíz calculada fuera exacta, la operación de división conlleva errores de redondeo, como ya sabemos, por lo que en lugar de obtener el polinomio p_1 obtenemos una aproximación $\tilde{p}_1(x) = \tilde{a}_n(x - \tilde{\xi}_2)(x - \tilde{\xi}_3)\ldots(x - \tilde{\xi}_n)$. Se puede argumentar que, aunque el polinomio $\tilde{p}_1$ es diferente de p_1, son polinomios próximos $\tilde{p}_1(x) \approx p(x)$. Sin embargo esto no implica que las raíces $\tilde{\xi}_j$ sean próximas a ξ_j. Vamos a ver un ejemplo. James Hardy Wilkinson en sus primeros cálculos numéricos implementó el método de Newton para calcular las raíces del polinomio de grado 20: $p(x) = (x-1)(x-2)(x-3)\ldots(x-20)$. Este polinomio lleva el nombre de polinomio de Wilkinson. Primeramente, realizó un

programa para calcular los coeficientes $\{a_j\}_{j=0}^{20}$ del polinomio. El comando de MATLAB *poly(1:20)* nos proporciona los coeficientes de este polinomio. Claro que, como ya sabemos, estas operaciones tienen errores de redondeo y el resultado es un polinomio $\tilde{p}$ próximo a p. Si calculamos las raíces del polinomio $\tilde{p}$ usando MATLAB, *roots($\tilde{p}$)*, veremos que hay una diferencia significativa con las raíces de p, que son $1, 2, 3, \ldots, 20$. Consideremos además la siguiente modificación de $\tilde{p}$. El coeficiente del término x^{19} de $\tilde{p}$, que coincide con el de p, es -210. Entonces, construimos un polinomio q que posee los mismos coeficientes que $\tilde{p}$, excepto el correspondiente al término x^{19}, donde cambiamos -210 por $-210 + 2^{-23}$ ($2^{-23} \approx 1.192092895507813e-07$). Finalmente, comparamos las raíces de ambos polinomios:

Raíces de $\tilde{p}$	Raíces de q
$19.999874055724192 + 0.000000000000000i$	$20.476776817076942 + 1.039020302403400i$
$19.001295393676987 + 0.000000000000000i$	$20.476776817076942 - 1.039020302403400i$
$17.993671562737585 + 0.000000000000000i$	$18.181332951492898 + 2.548951360228183i$
$17.018541647321989 + 0.000000000000000i$	$18.181332951492898 - 2.548951360228183i$
$15.959717574548915 + 0.000000000000000i$	$15.305945141621015 + 2.775398103616350i$
$15.059326234074415 + 0.000000000000000i$	$15.305945141621015 - 2.775398103616350i$
$13.930186454760916 + 0.000000000000000i$	$12.821765409086765 + 2.123564850794045i$
$13.062663652011070 + 0.000000000000000i$	$12.821765409086765 - 2.123564850794045i$
$11.958873995343460 + 0.000000000000000i$	$10.892965459081427 + 1.149555526766399i$
$11.022464271003383 + 0.000000000000000i$	$10.892965459081427 - 1.149555526766399i$
$9.991190949230132 + 0.000000000000000i$	$9.501627781547665 + 0.000000000000000i$
$9.002712743189727 + 0.000000000000000i$	$9.147472561171277 + 0.000000000000000i$
$7.999394310958664 + 0.000000000000000i$	$7.993034045900652 + 0.000000000000000i$
$7.000096952230211 + 0.000000000000000i$	$7.000300798429224 + 0.000000000000000i$
$5.999989523351082 + 0.000000000000000i$	$5.999992985747001 + 0.000000000000000i$
$5.000000705531480 + 0.000000000000000i$	$5.000000162694260 + 0.000000000000000i$
$3.999999973862455 + 0.000000000000000i$	$3.999999988082382 + 0.000000000000000i$
$3.000000000444877 + 0.000000000000000i$	$3.000000000500764 + 0.000000000000000i$
$1.999999999998383 + 0.000000000000000i$	$1.999999999999695 + 0.000000000000000i$
$0.999999999999949 + 0.000000000000000i$	$0.999999999999815 + 0.000000000000000i$

Obviamente el cálculo de los coeficientes del polinomio sufre de errores de redondeo propios del ordenador. En particular, en los cálculos realizados por Wilkinson, el coeficiente del término de x^{19} sufría un error del orden de 2^{-23}. Como el programa realizado por Wilkinson estaba preparado para calcular raíces reales y el polinomio resultante poseía raíces complejas, fallaba la convergencia del algoritmo. Le llevó a Wilkinson muchas horas de trabajo comprender lo que sucedía; ver [13]. Después de esta experiencia y otras similares, Wilkinson se convirtió en uno de los más destacados analistas numéricos, habiendo dejado estudios muy importantes sobre el efecto de los errores de redondeo en los algoritmos numéricos.

En la tabla de arriba, observamos que las 10 primeras raíces de q son complejas con una parte imaginaria nada pequeña. Por lo tanto, concluimos que

el hecho de que dos polinomios tengan coeficientes muy próximos no implica que las correspondientes raíces también sean próximas. Consecuentemente, el procedimiento de deflación no es recomendable para el cálculo de las raíces de un polinomio. Un método alternativo que vamos a utilizar es el **Método de Maehly**. La idea de este método es la siguiente: una vez calculadas las raíces aproximadas $\{\xi_1, \xi_2, \ldots, \xi_m\}$ de p, se aplica el método de Newton a la función

$$f_m(x) = \frac{p(x)}{(x-\xi_1)(x-\xi_2)\ldots(x-\xi_m)}.$$

No se realiza la división, sino que se deja la función tal cual. Entonces calculamos la derivada de f_m:

$$\begin{aligned} f'_m(x) &= \frac{p'(x)}{(x-\xi_1)(x-\xi_2)\ldots(x-\xi_m)} \\ &- \frac{p(x)\big[(x-\xi_2)\ldots(x-\xi_m)+(x-\xi_1)(x-\xi_3)\ldots(x-\xi_m)+\ldots+(x-\xi_1)\ldots(x-\xi_{m-1})\big]}{(x-\xi_1)^2(x-\xi_2)^2\ldots(x-\xi_m)^2} \\ &= \frac{p'(x)-p(x)\Big[\frac{1}{x-\xi_1}+\frac{1}{x-\xi_2}+\ldots+\frac{1}{x-\xi_m}\Big]}{(x-\xi_1)(x-\xi_2)\ldots(x-\xi_m)}. \end{aligned}$$

Ahora tenemos

$$\frac{f_m(x)}{f'_m(x)} = \frac{p(x)}{p'(x)-p(x)\Big[\frac{1}{x-\xi_1}+\frac{1}{x-\xi_2}+\ldots+\frac{1}{x-\xi_m}\Big]} = \frac{p(x)}{p'(x)-p(x)\displaystyle\sum_{j=1}^{m}\frac{1}{x-\xi_j}}.$$

Entonces, el método de Newton aplicado a la función f_m se formula como sigue:

$$\begin{cases} x_0 \in \mathbb{C} \\ x_{k+1} = x_k - \dfrac{p(x_k)}{p'(x_k)-p(x_k)\displaystyle\sum_{j=1}^{m}\frac{1}{x_k-\xi_j}}, \quad k \geq 0. \end{cases} \tag{2.6}$$

OBSERVACIONES 2.9

• *En el cálculo de la primera raíz se tiene que $m = 0$, por lo tanto el algoritmo* (2.6) *se reduce al método de Newton aplicado al polinomio p:*

$$\begin{cases} x_0 \in \mathbb{C} \\ x_{k+1} = x_k - \dfrac{p(x_k)}{p'(x_k)}, \quad k \geq 0. \end{cases} \tag{2.7}$$

• *Si los coeficientes del polinomio son todos números reales y x_0 es un número real, entonces la sucesión $\{x_k\}_{k=0}^{\infty}$ generada por el algoritmo* (2.7) *está formada por números reales, por lo que será imposible que converja a una raíz compleja. Lo mismo sucede con el algoritmo* (2.6) *si las raíces $\{\xi_1, \xi_2, \ldots, \xi_m\}$ son todas ellas reales. Por lo tanto resulta imprescindible tomar x_0 como un número*

complejo en estas circunstancias. Si empezamos en un número complejo y las raíces restantes son reales, lo que sucederá es que la parte imaginaria de x_k convergerá hacia cero y obtendremos una raíz real.

• *En el caso de haber calculado las raíces $\{\xi_1, \xi_2, \ldots, \xi_m\}$, para calcular ξ_{m+1} debemos empezar en un punto x_0 lejos de las raíces ya calculadas. Observemos que si, por ejemplo, ξ_m es la aproximación de la raíz exacta $\bar{\xi}_m$, entonces $|f'_m(\xi_m)| \approx \infty$, lo que provoca que la convergencia de $\{x_k\}_{k=0}^{\infty}$ hacia $\bar{\xi}_m$ solamente se producirá si x_0 es muy próximo a $\bar{\xi}_m$, lo que se evita si tomamos x_0 distante de ξ_m.*

Sabemos que el método de Newton es localmente convergente. Por lo tanto, podemos iniciar las iteraciones en un punto x_0 sin obtener la convergencia. Veamos un par de ejemplos.

Ejemplo 1 *Se considera el polinomio $p(x) = x^2 - 3$. Aplicamos el método de Newton comenzando en $x_0 = +i$:*

$$x_1 = x_0 - \frac{p(x_0)}{p'(x_0)} \;\Rightarrow\; x_1 = -i$$
$$x_2 = x_1 - \frac{p(x_1)}{p'(x_1)} \;\Rightarrow\; x_2 = +i$$

Ejemplo 2 *Sea $p(x) = x^6 - 3x^5 + 4x^4 - 3x^3 - 3x^2 + 4x - 4$. Aplicando de nuevo el método de Newton obtenemos*

$$x_0 = 0$$
$$x_1 = x_0 - \frac{p(x_0)}{p'(x_0)} \;\Rightarrow\; x_1 = 1$$
$$x_2 = x_1 - \frac{p(x_1)}{p'(x_1)} \;\Rightarrow\; x_2 = 0$$

Como vemos, en ambos ejemplos los valores de x_k se alternan entre dos puntos que no son raíces del polinomio y la convergencia falla. Por lo tanto, si ejecutamos las primeras iteraciones del algoritmo (2.6) y los valores x_k varían sin apreciarse convergencia, entonces debemos concluir que x_0 está demasiado lejos de cualquier raíz. Hay que cambiar el punto de inicio.

• *El método de Newton converge cuadráticamente hacia una raíz $\bar{x}$ si $p'(\bar{x}) \neq 0$. Por lo tanto, si vemos que los valores de x_k convergen, pero la convergencia es muy lenta, entonces puede deberse a que nos estamos aproximando a una raíz múltiple o hay raíces muy próximas. En este caso, aplicamos el método de Newton al polinomio derivada, comenzando las iteraciones en el último punto x_k calculado en las iteraciones sobre p. Si la convergencia todavía fuera lenta, debemos reiterar el proceso y aplicar el método de Newton a la segunda derivada. Si calculamos una raíz ξ de $p'(x)$ (o $p^{(j)}(x)$, con $j \geq 2$) y observamos que $|p(\xi)| \gg |p'(\xi)|$, entonces debemos concluir que ξ no es raíz de p. Sucede que p posee dos o más raíces muy próximas. En caso contrario, tendremos que ξ es una raíz doble de p (de multiplicidad $j+1$ is $p(\xi), p'(\xi), \ldots, p^{(j)}(\xi)$ toman valores muy pequeños). Si lo que sucede es que hay raíces próximas, entonces la*

convergencia será lenta y, en general, no podremos obtener una precisión alta en la solución.
• *Conviene recordar que si los coeficientes del polinomio p son números reales y hemos calculado una raíz compleja* $\xi = \alpha + \beta i$*, entonces su conjugada* $\bar{\xi} = \alpha - \beta i$ *también es raíz del polinomio. Si además,* ξ *tiene multiplicidad* j*, entonces* $\bar{\xi}$ *también posee multiplicidad* j*.*

La siguiente función MATLAB (*itraiz.m*) realiza una iteración del método de Maehly:

```
function y=itraiz(c,d,v,x)
%
%     function y=itraiz(c,d,v,x)
%
%      Copyright: Eduardo Casas (Febrero, 1995)
%
% Función que realiza una iteración del método de Maehly
% Variables de Entrada:
%
% c      Coeficientes del polinomio cuya raíz se desea calcular.
% d      Coeficientes del polinomio derivada.
% v      Vector que contiene las raices ya calculadas.
% x      Punto sobre el que se desea iterar.
%
% Variable de Salida:
%
% y      Resultado de la iteración.
%
n=length(v);
px=polyval(c,x);
if n==0,
    y=x-px/polyval(d,x);
else
    s=sum(1./(x-v));
    y=x-px/(polyval(d,x)-px*s);
end
```

2.6 Ejercicios

1. Resuelve las siguientes ecuaciones
   (a) $x - \cos x = 0$.
   (b) $8x - \cos x - 2x^2 = 0$.
   (c) $\mathrm{e}^x + 3x - x^2 = 2$.
   (d) $\mathrm{e}^x + 2^{-x} + 2\cos x = 6$.
   (e) $x - 0.2\,\mathrm{sen}\, x - 0.8 = 0$.

2. Dada la ecuación $x^3+3x^2+2 = 0$, aplica el método de Newton comenzando las iteraciones en el punto $x_0 = 1$. Explica los resultados. Elige un punto más conveniente para iniciar las iteraciones. Por último, toma $x_0 = i$ y repite las iteraciones.

3. Dado el polinomio $p(x) = 27x^4 + 27x^3 - 45x^2 + 17x - 2$, prueba que posee una raiz en el intervalo $[0, 1]$ y aplica el método de Newton para calcularla. ¿Es la convergencia cuadrática? Explica los resultados y propón un método alternativo para calcularla. Calcula todas las raíces del polinomio.

4. Dado el polinomio $p(x) = 16x^4 - 40x^3 + 41x^2 - 40x + 25$, prueba que posee una raiz en el intervalo $[1, 2]$ y aplica el método de Newton para calcularla. ¿Es la convergencia cuadrática? Explica los resultados y propón un método alternativo para calcularla. Calcula todas las raíces del polinomio.

5. Dada la ecuación $x^4 - 5x^3 + 3.9999x^2 + 2.9999x + 8.9999 = 0$, aplica el método de Newton comenzando las iteraciones en el punto $x = 3$. ¿Es la velocidad de convergencia cuadrática? Explica los resultados.

6. Dada la ecuación $2x^6 - 6x^5 + x^4 + 8x^3 - x^2 - 4x - 1 = 0$, se pide lo siguiente.
   (a) Prueba que hay al menos una solución en el intervalo $[1, 2]$. Aplica el método de Newton comenzando en $x_0 = 1.5$. ¿Es la convergencia cuadrática? Explica los resultados y calcula la raíz con la máxima precisión que te permita tu ordenador. ¿Hay más raices en el intervalo $[1, 2]$?
   (b) Repite el análisis en el intervalo $[-1, 0]$.
   (c) Calcula todas las soluciones de la ecuación.

7. Dada la ecuación
   $$\log\left(1 + \mathrm{sen}^3 x + x^2\right) + \mathrm{e}^{x^2} = 1,$$
   se sabe que $\bar{x} = 0$ es la única solución. Aplica el método de Newton para resolver dicha ecuación (sin combinarlo con el método de bisección) comenzando las iteraciones primero en el punto $x_0 = -1$ y luego en $x_0 = +1$. ¿Es la velocidad de convergencia cuadrática en alguno de los dos casos? ¿Converge hacia la solución? Responde razonadamente a estas preguntas explicando lo que sucede.

8. Halla todas las soluciones de la ecuación $x^6+2x^5+5x^4+8x^3+8x^2+8x+4 = 0$.

9. Se considera la ecuación $\boxed{\det A(x) = 0}$, donde x es un número real y $A(x)$ denota la siguiente matriz

$$A(x) = \begin{pmatrix} \operatorname{sen} x & \cos x & x & 1 & 2 & 3 \\ 0 & 1 & 0 & 1 & 2 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & x & x^2 & x^3 & 5 & -2 \\ 1 & 7 & -5 & 6 & 2 & 8 \\ -x & -2x & 1+x & \cos x & 2 & 5 \end{pmatrix}$$

   Determina el número de soluciones de la ecuación y calcúlalas.

10. La función $\operatorname{erf} : [0, +\infty) \longrightarrow \mathbb{R}$ se define por

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2}\, dt.$$

    Determina el valor de x para el que erf(x)= 1/2. (NOTA: Utiliza la función Matlab erf para la evaluación de las integrales.)

11. Calcula los mínimos y máximos de la función $f(x) = \mathrm{e}^{-x^2} + x^3 - 3x$ en el intervalo $[-2, +2]$, así como los puntos de inflexión.

12. Dada la matriz

$$A = \begin{pmatrix} x & 2 & -3 & 5 & 0 & x^2 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & x & x & 0 & 5 & -2 \\ 1 & -1 & 1 & -1 & 2x & 0 \\ 1 & 0 & 1 & 0 & x^3 & -2 \\ 0 & 0 & 1 & 0 & \cos x & -x \end{pmatrix}$$

    determina un valor de x tal que -2 sea un valor propio de A.

13. Calcula todas la raíces de los polinomios siguientes:

    (a) $p(x) = x^3 - 2x - 5$
    (b) $p(x) = 2x^5 - x^4 - 4x^3 - 2x^2 - 6x + 3$
    (c) $p(x) = x^4 - 12x^3 + 47x^2 - 60x$
    (d) $p(x) = x^4 - 12x^3 + 47x^2 - 60x + 24$
    (e) $p(x) = x^4 - 12x^3 + 47x^2 - 60x + 24.1$
    (f) $p(x) = 8x^7 + 26x^6 + 54x^5 + 27x^4 - 128x^3 - 576x^2 - 864x - 432$
    (g) $p(x) = 3x^7 + 5x^6 + 21x^5 + 35x^4 + 48x^3 + 80x^2 + 36x + 60$
    (h) $p(x) = 25x^8 + 85x^7 + 181x^6 + 238x^5 + 226x^4 + 142x^3 + 61x^2 + 13x + 1$

14. Se considera la ecuación $\log(x^2+1) - \mathrm{e}^{0.4x}\cos\pi x = 0$. Se pide

    (a) Prueba que existe un única solución en el intervalo $[-1, 0]$. Calculala.

    (b) Demuestra que

    $$\log(x^2+1) > \mathrm{e}^{0.4x} \geq \mathrm{e}^{0.4x}\cos\pi x \quad \forall x < -1.$$

    Concluye que sólo existe un número negativo solución de la ecuación.

    (c) ¿Cuántas soluciones posee la ecuación? Argumenta la respuesta.

Capítulo 3

Aproximación de Funciones de una Variable Real por Polinomios

3.1 Introducción

Dada una función $f : [a, b] \longrightarrow \mathbb{R}$ con $-\infty < a < b < +\infty$, se desea aproximar dicha función por un polinomio de grado dado $\leq n$. Hay varias razones que motivan obtener tal aproximación. Imaginemos que deseamos calcular la integral $\int_a^b f(x)\,dx$, pero no se conoce una primitiva de f o incluso puede no existir una primitiva elemental de dicha función. Si conocemos un polinomio p que aproxima a f, entonces podemos aproximar la integral como sigue

$$\int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx,$$

y basta con calcular la integral del polinomio, que sabemos cómo hacerlo. También podemos aproximar $f'(x) \approx p'(x)$ cuando no disponemos de una expresión analítica de $f(x)$. Puede suceder incluso que f se conozca solamente en algunos puntos. Podemos imaginar circunstancias en las que $f(x)$ solamente puede obtenerse después de realizar algunas mediciones o experimentos en el laboratorio. En estos casos resulta muy apropiado disponer de una función con una representación analítica sencilla que aproxime a f. Los polinomios son especialmente sencillos para manejarlos analíticamente, por ejemplo para integrar o derivar.

En este capítulo vamos a ver dos técnicas para aproximar f por un polinomio: la *interpolación* y la *aproximación por mínimos cuadrados*. Ambas son ampliamente utilizadas. Dentro de la interpolación distinguiremos la interpolación de Lagrange y la de Hermite.

El lector puede consultar los textos [8] y [12] para un mayor estudio sobre los contenidos de este capítulo.

3.2 Interpolación de Lagrange

DEFINICIÓN 3.1 *Dados $n+1$ puntos distintos $\{x_0, x_1, \ldots, x_n\}$ del intervalo $[a, b]$ y una función $f : [a, b] \longrightarrow \mathbb{R}$, se dice que un polinomio p es el polinomio de interpolación de Lagrange de la función f en los puntos citados si p es de grado $\leq n$ y $p(x_i) = f(x_i)$ para cada $0 \leq i \leq n$. Los puntos $\{x_i\}_{i=0}^n$ se llaman nodos de interpolación. La interpolación se dice lineal si $n = 1$, cuadrática si $n = 2$ y cúbica si $n = 3$.*

TEOREMA 3.2 *El problema de interpolación de Lagrange posee una única solución. Además ésta viene dada mediante la fórmula*

$$p(x) = \sum_{i=0}^{n} f(x_i) l_i(x), \tag{3.1}$$

donde $\{l_i\}_{i=0}^n$ son los polinomios de base de Lagrange asociados a los nodos $\{x_i\}_{i=0}^n$ y que vienen dados por las expresiones

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Demostración. Primeramente notemos que cada polinomio de base l_j es un polinomio de grado exactamente n y que satisface la propiedad

$$l_j(x_k) = \begin{cases} 1 & \text{si } j = k, \\ 0 & \text{si } j \neq k, \end{cases} \qquad \text{para } \; k = 0, 1, 2, \ldots, n. \tag{3.2}$$

Entonces, es inmediato que el polinomio p dado por la fórmula (3.1) es de grado $\leq n$ y $p(x_k) = f(x_k)$ for $k = 0, 1, 2, \ldots, n$. Así obtenemos una solución del problema de interpolación de Lagrange mediante la fórmula (3.1). Probemos que no puede haber más de una solución. Para ello supongamos que p y q son dos soluciones del mismo problema de interpolación de Lagrange. Definimos $h = p - q$. Dado que p y q interpolan a f en los nodos $\{x_0, x_1, \ldots, x_n\}$, entonces

$$h(x_k) = p(x_k) - q(x_k) = f(x_k) - f(x_k) = 0 \;\; \text{para } k = 0, 1, 2, \ldots, n.$$

Entonces, h es un polinomio de grado $\leq n$ que se anula en $n+1$ puntos. Teniendo en cuenta que el Teorema Fundamental del Álgebra nos dice que un polinomio de grado n se anula en n puntos exactamente, la conclusión anterior sólo es posible si h es el polinomio nulo: $h = 0$ y por lo tanto $p = q$. Hemos probado que no puede haber dos soluciones distintas. ∎

Aunque la fórmula (3.1) posee un interés teórico importante, no es la forma más eficiente de calcular el polinomio interpolador. Un método sencillo y más rápido es el siguiente. Se trata de calcular $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ de forma que $p(x_k) = f(x_k)$ para $k = 0, 1, 2, \ldots, n$. Estas identidades definen unas ecuaciones que permiten determinar los coeficientes $\{a_k\}_{k=0}^n$ del polinomio. En efecto, se debe cumplir

$$\left\{\begin{array}{lcl} a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n & = & f(x_0) \\ a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_nx_1^n & = & f(x_1) \\ a_0 + a_1x_2 + a_2x_2^2 + \ldots + a_nx_2^n & = & f(x_2) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots & \ldots & \ldots \\ a_0 + a_1x_n + a_2x_n^2 + \ldots + a_nx_n^n & = & f(x_n) \end{array}\right.$$

lo que podemos escribir en forma matricial

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}}_{a} = \underbrace{\begin{pmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}}_{b}$$

El método consiste en construir la matriz A $(n+1) \times (n+1)$, el vector $b \in \mathbb{R}^{n+1}$ y resolver el sistema $Aa = b$, donde $a \in \mathbb{R}^{n+1}$ es el vector de los coeficientes. Dado que el problema de interpolación de Lagrange tiene un única solución, la matriz A es necesariamente inversible.

Vamos a considerar un ejemplo sencillo que resolveremos usando la fórmula (3.1) y también mediante la resolución del correspondiente sistema de ecuaciones. Sea $f : [0, 1] \longrightarrow \mathbb{R}$ la función definida por $f(x) = \text{sen}(\pi x) + 8x^2 - 2x + 3$. Vamos a calcular el polinomio de interpolación cuadrático asociado a los nodos $x_0 = 0$, $x_1 = \frac{1}{2}$ y $x_2 = 1$. Primero calculamos los polinomio de base de Lagrange

$$\begin{aligned} l_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - \frac{1}{2})(x - 1)}{(-\frac{1}{2})(-1)} = 2x^2 - 3x + 1, \\ l_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{x(x - 1)}{(\frac{1}{2})(\frac{1}{2} - 1)} = -4x^2 + 4x, \\ l_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{x(x - \frac{1}{2})}{1(1 - \frac{1}{2})} = 2x^2 - x. \end{aligned}$$

Ahora el polinomio de interpolación viene dado por la fórmula (3.1):

$$\begin{aligned} p(x) &= f(x_0)l_0(x) + f(x_1)l_1(x) + f(x_2)l_2(x) \\ &= 3(2x^2 - 3x + 1) + 5(-4x^2 + 4x) + 9(2x^2 - x) \\ &= 4x^2 + 2x + 3. \end{aligned}$$

El segundo procedimiento para calcular el polinomio interpolador consiste en plantear un sistema lineal de ecuaciones para hallar los coeficientes del polinomio. Como estamos buscando un polinomio cuadrático ($n = 2$), el polinomio será de la forma $p(x) = a_2x^2 + a_1x + a_0$. Imponiendo las condiciones $p(x_i) = f(x_i)$ para $i = 0, 1, 2$, se obtiene el sistema

$$\begin{cases} a_0 = 3 \\ \frac{1}{4}a_2 + \frac{1}{2}a_1 + a_0 = 5 \\ a_2 + a_1 + a_0 = 9 \end{cases}$$

La solución del sistema es $a_2 = 4$, $a_1 = 2$ y $a_0 = 3$. Así obtenemos la misma solución que antes, como ya sabíamos.

Una cuestión importante es saber qué error se comete cuando reemplazamos la función f por el polinomio de interpolación de Lagrange. El siguiente teorema proporciona una cota del error. Su demostración es una aplicación sencilla del Teorema de Rolle, pero que no detallaremos aquí. El lector interesado puede ver su demostración en [8, §5.2.1] y [12, Teorema 6.2].

TEOREMA 3.3 *Supongamos que la función f is $n + 1$ veces derivable en el intervalo $[a, b]$ y que $f^{(n+1)}$ es una función continua. Entonces, una cota del error de interpolación de Lagrange viene dada por la expresión*

$$|f(x) - p(x)| \leq \frac{\max\limits_{\xi\in[a,b]} |f^{n+1}(\xi)|}{(n+1)!} |\prod_{j=0}^{n}(x - x_j)|. \tag{3.3}$$

Vamos a utilizar este teorema para calcular una cota del error correspondiente al problema de interpolación resuelto antes. Teniendo en cuenta que $n = 2$, la fórmula (3.3) nos conduce a la desigualdad

$$|f(x) - p(x)| \leq \frac{\max\limits_{\xi\in[0,1]} |f'''(\xi)|}{3!} |x(x - \frac{1}{2})(x - 1)|.$$

Dado que $f'''(x) = -\pi^3 \cos(\pi x)$, entonces

$$\frac{\max\limits_{\xi\in[0,1]} |f'''(\xi)|}{3!} = \frac{\pi^3}{6}.$$

Además se tiene: $x(x - \frac{1}{2})(x - 1) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x$. Vamos a calcular el valor máximo de $|x^3 - \frac{3}{2}x^2 + \frac{1}{2}x|$ en el intervalo $[0, 1]$. Como en los extremos del intervalo esta función se anula, el valor máximo se alcanza en un punto del intervalo abierto $(0, 1)$. Será por lo tanto un máximo o mínimo relativo del polinomio. Calculemos los máximos y mínimos relativos del polinomio $x^3 - \frac{3}{2}x^2 + \frac{1}{2}x$. Su derivada es $3x^2 - 3x + \frac{1}{2}$ y las raíces de este polinomio son 0.788675134594813 y 0.211324865405187. Evaluando la expresión $|x^3 - \frac{3}{2}x^2 + \frac{1}{2}x|$

en estos dos puntos obtenemos el mismo valor: 0.048112522432469. Por lo tanto, la cota del error es

$$|f(x) - p(x)| \leq \frac{\pi^3}{6} 0.048112522432469 \approx 0.497263394109420$$

OBSERVACIÓN 3.4 *A la vista de la fórmula* (3.3)*, es natural plantearse la elección de los nodos* $\{x_j\}_{j=0}^n$ *de forma que la cantidad*

$$\max_{x\in[a,b]} |\prod_{j=0}^{n}(x - x_j)|$$

sea mínima. Esto se consigue tomando los llamados puntos de Chebyshev

$$x_j = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{(2j+1)\pi}{2n+2} \quad 0 \leq j \leq n. \tag{3.4}$$

Con esta elección de los nodos se tiene

$$\max_{x\in[a,b]} |\prod_{j=0}^{n}(x - x_j)| = \frac{(b-a)^{n+1}}{2^{2n+1}}. \tag{3.5}$$

Combinando (3.3) *y* (3.5)*, obtenemos la siguiente fórmula del error para los nodos de Chebyshev*

$$|f(x) - p(x)| \leq \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \max_{\xi\in[a,b]} |f^{n+1}(\xi)|. \tag{3.6}$$

Para los detalles el lector puede consultar [8, §5.4.2] y [12, §8.5].

3.3 Interpolación de Hermite

DEFINICIÓN 3.5 *Sea* $f : [a,b] \longrightarrow \mathbb{R}$ *una función,* $\{x_0, x_1, \ldots, x_k\}$ $k+1$ *puntos distintos del intervalo* $[a,b]$ *y* $\{\alpha_0, \alpha_1, \ldots, \alpha_k\}$ $k+1$ *enteros no negativos. El problema de interpolación de Hermite consiste en determinar un polinomio de grado* $\leq n = k + \alpha_0 + \alpha_1 + \ldots + \alpha_k$ *satisfaciendo*

$$p^{(j)}(x_i) = f^{(j)}(x_i), \quad 0 \leq j \leq \alpha_i \quad \text{y} \quad 0 \leq i \leq k. \tag{3.7}$$

En el caso particular en que $\alpha_i = 1$ *para cada* $0 \leq i \leq n$*, entonces el problema se denomina interpolación de Hermite clásica.*

Es importante notar que si sobre un nodo x_i se impone una condición sobre la derivada de orden α_i, entonces también se debe imponer una condición sobre cada derivada inferior a α_i y sobre el valor de la función. El siguiente problema de interpolación no es un problema de interpolación de Hermite:

Sea $f : [-1,+1] \longrightarrow \mathbb{R}$, $f(x) = \text{sen}(\pi x)$.
Hallar un polinomio p de grado 2 satisfaciendo:
$p(-1) = f(-1)$, $p(+1) = f(+1)$, $p'(0) = f'(0)$.

En la interpolación de Hermite, si se impone una condición sobre la derivada $p'(0)$ se debe imponer también un condición sobre el valor de la función $p(0)$. Veamos otro ejemplo:

$$\begin{aligned}&\text{Sea } f:[0,\pi]\longrightarrow\mathbb{R},\quad f(x)=\cos(x).\\&\text{Hallar un polinomio } p \text{ de grado 2 satisfaciendo:}\\&p''(0)=f''(0),\ \ p'(\frac{\pi}{2})=f'(\frac{\pi}{2}),\ \ p'(\pi)=f'(\pi).\end{aligned}$$

Este problema tampoco es un problema de interpolación de Hermite.

Los problemas anteriores son problemas de interpolación de Birkhoff. Este tipo de interpolación consiste en imponer condiciones sobre el polinomio y/o sus derivadas (no necesariamente consecutivas) en determinados nodos. El problema de interpolación de Birkhoff no posee solución en todos los casos. De hecho, en los dos ejemplos anteriores es fácil comprobar que no existe solución. Sin embargo el problema de interpolación de Hermite siempre posee solución y es única.

TEOREMA 3.6 *El problema de interpolación de Hermite posee una única solución.*

Demostración. Al igual que en la interpolación de Lagrange, las condiciones (3.7) se pueden escribir como un sistema de ecuaciones lineales: $Aa = b$, aunque en este caso la matriz A y el vector b son obviamente diferentes. La existencia y unicidad de solución del problema de interpolación de Hermite es equivalente a la inversibilidad de la matriz A. Sabemos que A es inversible si y sólo si su núcleo se reduce al vector nulo, es decir, si $Aa = 0$ implica que $a = 0$. O lo que es lo mismo, si todos los términos $f^{(j)}(x_i)$ de (3.7) son nulos, entonces $p = 0$. Supongamos entonces que $p^{(j)}(x_i) = 0$ para $0 \leq j \leq \alpha_i$ y $0 \leq i \leq k$. Esto significa que cada punto x_i es una raíz de p con multiplicidad $\alpha_i + 1$. Por lo tanto, p posee al menos $k + 1 + \alpha_0 + \alpha_1 + \ldots + \alpha_k = n + 1$ raíces, contadas con su multiplicidad. Pero el grado de p es n. Utilizando otra vez el Teorema Fundamental del Álgebra, esto es posible si y sólo si $p = 0$. □

TEOREMA 3.7 *Supongamos que la función f es $n+1$ veces derivable en el intervalo $[a, b]$ y que $f^{(n+1)}$ es una función continua, entonces una cota del error de interpolación de Hermite viene dada por la expresión*

$$|f(x) - p(x)| \leq \frac{\max\limits_{\xi\in[a,b]} |f^{n+1}(\xi)|}{(n+1)!} |\prod_{j=0}^{k}(x - x_j)^{\alpha_j+1}|. \tag{3.8}$$

Resolvamos el siguiente problema de interpolación de Hermite: dada la función $f : [0, 1] \longrightarrow \mathbb{R}$, $f(x) = \text{sen}(\pi x) + 8x^2 - 2x + 3$, se pide determinar el

polinomio de interpolación de Hermite satisfaciendo las siguientes condiciones

$$\begin{aligned} &p(0) = f(0), \quad p'(0) = f'(0), \quad p''(0) = f''(0), \\ &p(\frac{1}{2}) = f(\frac{1}{2}), \\ &p(1) = f(1), \quad p'(1) = f'(1). \end{aligned} \tag{3.9}$$

En este ejemplo disponemos de tres nodos: $x_0 = 0$, $x_1 = \frac{1}{2}$ y $x_2 = 1$ y por lo tanto $k = 2$. Asociados a estos nodos tenemos $\alpha_0 = 2$, $\alpha_1 = 0$ y $\alpha_2 = 1$ y, en consecuencia, $n = k + \alpha_0 + \alpha_1 + \alpha_2 = 5$. Por lo tanto se busca un polinomio de grado $\leq n = 5$:

$$\begin{aligned} &p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5, \\ &\text{con derivadas:} \\ &p'(x) = a_1 + 2a_2 x + 3a_3 x^2 + 4a_4 x^3 + 5a_5 x^4, \\ &p''(x) = 2a_2 + 6a_3 x + 12a_4 x^2 + 20a_5 x^3. \end{aligned}$$

Imponiendo a este polinomio las condiciones (3.9) obtenemos el sistema de ecuaciones lineales

$$\left\{ \begin{array}{ccccccccccccc} a_0 & & & & & & & & & & & = & 3 \\ & & a_1 & & & & & & & & & = & \pi - 2 \\ & & & & 2a_2 & & & & & & & = & 16 \\ a_0 & + & \frac{1}{2}a_1 & + & \frac{1}{4}a_2 & + & \frac{1}{8}a_3 & + & \frac{1}{16}a_4 & + & \frac{1}{32}a_5 & = & 5 \\ a_0 & + & a_1 & + & a_2 & + & a_3 & + & a_4 & + & a_5 & = & 9 \\ & & a_1 & + & 2a_2 & + & 3a_3 & + & 4a_4 & + & 5a_5 & = & 14 - \pi \end{array} \right.$$

Tomando

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16} & \frac{1}{32} \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix} \quad \text{y} \quad b = \begin{pmatrix} 3 \\ \pi - 2 \\ 16 \\ 5 \\ 9 \\ 14 - \pi \end{pmatrix}$$

el sistema de ecuaciones anterior se puede escribir en notación matricial como $Aa = b$. Resolviendo el sistema con Matlab, obtenemos los coeficientes del polinomio interpolador

$$a = \begin{pmatrix} 3.000000000000000 \\ 1.141592653589793 \\ 8.000000000000000 \\ -5.699111843077514 \\ 1.973445725385644 \\ 0.584073464102076 \end{pmatrix}$$

A continuación vamos a aplicar el Teorema 3.7 para obtener una cota del error de interpolación. Dado que $n = 5$, necesitamos calcular la derivada sexta $f^{(6)}(x) = -\pi^6 \operatorname{sen}(\pi x)$. Entonces, tenemos que

$$M_6 = \max_{\xi \in [0,1]} |f^{(6)}(\xi)| = \pi^6.$$

Utilizando (3.8) se deduce

$$|f(x) - p(x)| \leq \frac{\pi^6}{6!}|x^3(x - \frac{1}{2})(x-1)^2| \leq \frac{\pi^6}{720} \max_{x \in [0,1]} |x^3(x - \frac{1}{2})(x-1)^2|.$$

Tenemos que $q(x) = x^3(x - \frac{1}{2})(x-1)^2 = x^6 - \frac{5}{2}x^5 + 2x^4 - \frac{1}{2}x^3$ y $q'(x) = 6x^5 - \frac{25}{2}x^4 + 8x^3 - \frac{3}{2}x^2$. Las raíces de q' son $v = \{0, 0, \frac{1}{3}, \frac{3}{4}, 1\}$. Los valores de q en estas raíces son $\{0, 0, -0.002743484224966, 0.006591796875000, 0\}$. Consecuentemente, el máximo de $|q(x)|$ en el intervalo $[0, 1]$ es 0.006591796875000, lo que conduce a la estimación del error

$$|f(x) - p(x)| \leq \frac{\pi^6}{720} 0.006591796875000 \approx 0.008801780947040.$$

3.4 Mínimos Cuadrados

Contrariamente a lo que pueda parecer, incrementando el número de puntos de interpolación no siempre se consigue una mejor aproximación de la función. Un ejemplo sencillo en el que esto se observa es el dado por la función de Runge: $f : [-5, +5] \longrightarrow \mathbb{R}$ con $f(x) = \frac{1}{1+x^2}$. Si tomamos puntos igualmente espaciados $x_j = -5 + jh$, $0 \leq j \leq n$ con $h = \frac{10}{n}$ y llamamos p_n al polinomio interpolador de f en dichos puntos, entonces se observa la no convergencia de $p_n(x)$ hacia $f(x)$ cuando $n \to \infty$ en los puntos próximos a los extremos del intervalos. El lector puede resolver el ejercicio 8 para constatar la ausencia de convergencia. Esta ausencia de convergencia que se produce en algunos casos se conoce como fenómeno de Runge.

Otro inconveniente de la interpolación en un número grande de nodos es que la evaluación y manipulación, en general, de un polinomio de grado n elevado es muy costosa computacionalmente y puede conllevar errores de redondeo importantes.

Entonces, ¿cómo aprovechar gran cantidad de información sobre una función sin incrementar excesivamente el grado del polinomio? Una de las técnicas frecuentemente empleadas es el de la aproximación por mínimos cuadrados. El problema se plantea en los siguientes términos.

DEFINICIÓN 3.8 *Dados $n+1$ puntos distintos $\{x_0, x_1, \ldots, x_n\}$ del intervalo $[a, b]$, $n+1$ números reales positivos $\{\omega_0, \omega_1, \ldots, \omega_n\}$, una función $f : [a, b] \longrightarrow \mathbb{R}$ y un entero $1 \leq k \leq n$, el problema de mínimos cuadrados consiste en resolver*

$$\text{(P)} \qquad \begin{array}{c} \text{Minimizar} \\ p \in \mathcal{P}_k \end{array} \quad \sum_{j=0}^{n} \omega_j (p(x_j) - f(x_j))^2$$

donde $\mathcal{P}_k$ denota el espacio de los polinomios de grado $\leq k$.

Obviamente, si $k = n$ entonces la solución de (P) es el polinomio de interpolación de Lagrange asociado a los nodos $\{x_0, x_1, \ldots, x_n\}$. Sin embargo, en la práctica, típicamente n es bastante más grande que k. Supongamos por un momento que $\omega_j = 1$ para cada $j = 0, 1, \ldots, n$. Entonces, la solución del problema anterior es un polinomio que toma valores en los nodos x_j próximos a $f(x_j)$: $p(x_j) \approx f(x_j)$. Sin embargo, lo habitual es que $p(x_j) \neq f(x_j)$. Es imposible, en general, ecotrar un polinomio de grado $k < n$ de forma que $p(x_j) = f(x_j)$ para $0 \leq j \leq n$. Esto constituye un sistema de $n+1$ ecuaciones con $k+1$ incógnitas, sin solución en general. Sin embargo, resolviendo el problema de minimización anterior, sí podemos conseguir que $p(x_j) \approx f(x_j)$ para cada $0 \leq j \leq n$. Es más, lo habitual es que dicha solución coincida con f en muchos puntos, que son distintos de los nodos x_j.

¿Por qué se toman pesos ω_j diferentes de 1?. Para responder a esta cuestión primero observemos que en muchas ocasiones los valores $f(x_j)$ solamente se conocen de forma aproximada, este es el caso cuando no se dispone de una expresión analítica de f, sino que los valores $f(x_j)$ son resultado de unas medidas. Notemos que si ω_j es bastante mayor que ω_i, por ejemplo $\omega_j = 10$ y $\omega_i = 0.1$, entonces el mínimo del problema (P) se alcanza para un polinomio tal que el $|p(x_j) - f(x_j)| \ll |p(x_i) - f(x_i)|$. Entonces hay dos motivos por los que unos pesos pueden ser mayores que otros: el primero es que el valor que tenemos de $f(x_j)$ es más preciso que el de $f(x_i)$, por lo que ajustamos el polinomio más a los valores más precisos de f; la segunda razón puede ser que deseemos más precisión en la aproximación en unas zonas del intervalo que en otras.

3.4.1 Formulación algebraica del problema

De cara a la resolución del problema de mínimos cuadrados (P), vamos a formular algebraicamente el problema. Estamos buscando un polinomio p de grado k, así que tenemos que determinar sus coeficientes:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k.$$

Denotemos por a el vector de coeficientes del polinomio

$$a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix}$$

Sean la matriz A y el vector b definidos como sigue

$$A = \begin{pmatrix} \sqrt{\omega_0} & \sqrt{\omega_0}x_0 & \cdots & \sqrt{\omega_0}x_0^{k-1} & \sqrt{\omega_0}x_0^k \\ \sqrt{\omega_1} & \sqrt{\omega_1}x_1 & \cdots & \sqrt{\omega_1}x_1^{k-1} & \sqrt{\omega_1}x_1^k \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \sqrt{\omega_n} & \sqrt{\omega_n}x_n & \cdots & \sqrt{\omega_n}x_n^{k-1} & \sqrt{\omega_n}x_n^k \end{pmatrix} \quad b = \begin{pmatrix} \sqrt{\omega_0}f(x_0) \\ \sqrt{\omega_1}f(x_1) \\ \vdots \\ \sqrt{\omega_n}f(x_n) \end{pmatrix}$$

Entonces, se tiene

$$Aa-b=\begin{pmatrix} \sqrt{\omega_0}\Big([a_0+a_1x_0+a_2x_0^2+\ldots+a_kx_0^k]-f(x_0)\Big) \\ \sqrt{\omega_1}\Big([a_0+a_1x_1+a_2x_1^2+\ldots+a_kx_1^k]-f(x_1)\Big) \\ \sqrt{\omega_2}\Big([a_0+a_2x_2+a_2x_2^2+\ldots+a_kx_2^k]-f(x_2)\Big) \\ \vdots \\ \sqrt{\omega_n}\Big([a_0+a_1x_n+a_2x_n^2+\ldots+a_nx_n^k]-f(x_n)\Big) \end{pmatrix}$$
$$=\begin{pmatrix} \sqrt{\omega_0}\big(p(x_0)-f(x_0)\big) \\ \sqrt{\omega_1}\big(p(x_1)-f(x_1)\big) \\ \sqrt{\omega_2}\big(p(x_2)-f(x_2)\big) \\ \vdots \\ \sqrt{\omega_n}\big(p(x_n)-f(x_n)\big) \end{pmatrix}$$

Recordemos que la norma euclídea de un vector $v=(v_0,v_1,v_2,\ldots,v_n)^{\mathrm{T}}$ es $\|v\|=\sqrt{\sum_{j=0}^{n}v_j^2}$, o equivalentemente $\|v\|^2=\sum_{j=0}^{n}v_j^2$. Así se concluye que

$$\|Aa-b\|^2=\sum_{j=0}^{n}\omega_j(p(x_j)-f(x_j))^2.$$

Por lo tanto, los coeficientes del polinomio p son la solución del problema

$$\text{(Q)}\quad \begin{array}{l}\text{Minimizar } \|Ax-b\|^2 \\ x\in\mathbb{R}^{k+1}\end{array}$$

3.4.2 Resolución de (Q) mediante la factorización QR

El algoritmo que proponemos aquí para resolver el problema (Q) es el siguiente:

1- Hacemos la factorización QR de A: $A=QR$, Q matriz ortogonal $(n+1)\times(n+1)$ y R triangular superior $(n+1)\times(k+1)$.

2- Consideramos en Q y R las siguientes submatrices: $Q=[Y\ Z]$ y $R=\begin{pmatrix}\hat{R}\\ O\end{pmatrix}$, con Y $(n+1)\times(k+1)$, Z $(n+1)\times(n-k)$, $\hat{R}$ $(k+1)\times(k+1)$ y O $(n-k)\times(k+1)$. $\hat{R}$ es una matriz cuadrada triangular superior y O es una matriz formada por ceros.

3- Los coeficientes $a=(a_0,a_1\ldots,a_k)^{\mathrm{T}}$ del polinomio solución del problema de mínimos cuadrados se obtienen resolviendo el sistema de ecuaciones lineales

$$\hat{R}a=Y^{\mathrm{T}}b$$

y el residual es

$$\text{Res}=\|Z^{\mathrm{T}}b\|.$$

Comprobemos que este algoritmo nos proporciona la solución. Primero recordemos que una matriz ortogonal Q es aquella que su inversa coincide con su traspuesta: $Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I$. También sabemos que $\|v\|^2 = v^{\mathrm{T}}v$. Entonces, se obtiene para cada vector v

$$\|Qv\|^2 = (Qv)^{\mathrm{T}}(Qv) = v^{\mathrm{T}}(Q^{\mathrm{T}}Q)v = v^{\mathrm{T}}Iv = v^{\mathrm{T}}v = \|v\|^2.$$

De lo expuesto se deduce

$$\begin{aligned}\|Ax-b\|^2 &= \|QRx-b\|^2 = \|Q[Rx-Q^{\mathrm{T}}b]\|^2 = \|Rx-Q^{\mathrm{T}}b\|^2\\ &= \left\|\begin{pmatrix}\hat{R}\\ O\end{pmatrix}x - \begin{pmatrix}Y^{\mathrm{T}}\\ Z^{\mathrm{T}}\end{pmatrix}b\right\|^2 = \left\|\begin{pmatrix}\hat{R}x-Y^{\mathrm{T}}b\\ -Z^{\mathrm{T}}b\end{pmatrix}\right\|^2 = \|\hat{R}x-Y^{\mathrm{T}}b\|^2 + \|Z^{\mathrm{T}}b\|^2.\end{aligned}$$

De la expresión anterior se deduce que el mínimo de (Q) se alcanza para el valor de x que hace cero el término $\|\hat{R}x - Y^{\mathrm{T}}b\|$, o lo que es equivalente para la solución del sistema $\hat{R}x = Y^{\mathrm{T}}b$. Es importante notar que la matriz $\hat{R}$ es inversible debido a que los nodos $\{x_j\}_{j=0}^n$ son todos distintos. Por lo tanto, concluimos que la solución del problema de mínimos cuadrados (Q) existe y es única. En consecuencia, el problema de mínimos cuadrados (P) posee una única solución.

3.5 Ejercicios

1. Dados los nodos $x_0 = -1$, $x_1 = 1$ y $x_2 = 2$, halla los polinomios de base de Lagrange asociados a los mismos. Calcula el polinomio que interpola en dichos nodos a la función $f : [-1, 2] \longrightarrow \mathbb{R}$ que satisface $f(x_0) = 2$, $f(x_1) = 1$ y $f(x_2) = 1$. Da una cota del error de interpolación en términos del valor máximo de $|f^{n+1}(x)|$ en el intervalo $[-1, 2]$.

2. Tomando como nodos $x_0 = 0$ y $x_1 = 1$, obtén el polinomio de interpolación de Hermite que interpola a la función $f : [0, 1] \longrightarrow \mathbb{R}$, $f(x) = \text{sen}\, \frac{\pi x}{2}$, en la forma siguiente

$$p(x_i) = f(x_i), \quad p'(x_i) = f'(x_i), \quad p''(x_0) = f''(x_0), \quad i = 0, 1.$$

   Halla una cota del error de interpolación.

3. Sea la función $f : (0, \infty) \longrightarrow \mathbb{R}$ definida por

$$f(x) = \int_x^\infty \mathrm{e}^{-t} \frac{1}{t}\, dt.$$

   Sabiendo que $f(0.03) = 2.9591$ y $f(0.04) = 2.6813$, obtén mediante interpolación de Hermite clásica una aproximación de $f(0.0357)$. Halla una cota del error.

4. Interpola la función $f(x) = \log x$ en los nodos $\{0.9, 1, 1.1\}$ y utiliza el polinomio interpolador de Lagrange para obtener aproximaciones de $\log x$ en los puntos 0.95678 y 1.03597. Compara los valores obtenidos con los exactos. Aproxima ahora f en los mismos puntos mediante el polinomio interpolador de Lagrange de grado 2 asociado a los nodos de Chebyshev del intervalo $[0.9, 1.1]$.

5. Dada la función $f : [0, 1] \longrightarrow \mathbb{R}$, $f(x) = \log(1 + x)$, determina los polinomios de grado 3 que resuelven los siguientes problemas de interpolación

   (i) $p(x_i) = f(x_i)$, $x_i = \frac{i}{3}$, $0 \leq i \leq 3$.

   (ii) $p(x_i) = f(x_i)$, $0 \leq i \leq 3$, donde $\{x_i\}_{i=0}^3$ denotan los cuatro nodos de Chebyshev en el intervalo $[0, 1]$.

   (iii) $p^{(k)}(x_i) = f^{(k)}(x_i)$, $i = 0, 1$, $k = 0, 1$, $x_0 = 0$, $x_1 = 1$.

   Calcula una cota del error de interpolación en cada uno de los tres casos.

6. Dada la función $f : [0, \pi] \longrightarrow \mathbb{R}$, $f(x) = \text{sen}\, x$, determina los polinomios de grado 10 que resuelven los siguientes problemas de interpolación

   (i) $p(x_i) = f(x_i)$, $x_i = \dfrac{\pi i}{10}$, $0 \leq i \leq 10$.

(ii) $p(x_i) = f(x_i)$, $0 \leq i \leq 10$, donde $\{x_i\}_{i=0}^{10}$ denotan los nodos de Chebyshev en el intervalo $[0, \pi]$.

Calcula una cota del error de interpolación en cada uno de los casos.

7. Dada la función $f : [0, 1] \longrightarrow \mathbb{R}$, $f(x) = \log(1+x)$, determina el polinomio de interpolación de Hermite que satisface $p(x_i) = f(x_i)$ y $p'(x_i) = f'(x_i)$ con $x_i = i/5$, $0 \leq i \leq 5$. Halla una cota del error de interpolación.

8. Se desea tabular la función $f(x) = \operatorname{sen} x$ en el intervalo $[0, \frac{\pi}{2}]$. Para ello se eligen $(n+1)$ puntos uniformemente espaciados $x_j = jh$, con $j = 0, \ldots, n$ y $h = \frac{\pi}{2n}$. Para la determinación de los valores de f en un punto x fuera de la tabla, se utilizará una interpolación lineal en los nodos más próximos a x. Halla una cota del error de interpolación en términos de h. Halla el valor de h si deseamos tener un error inferior a 10^{-6}. Responde a las mismas cuestiones si en lugar de una interpolación lineal realizamos interpolación cúbica.

9. Se desea tabular la función de distribución gaussiana

$$F : \mathbb{R} \longrightarrow \mathbb{R}, \quad F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\, dt.$$

Teniendo en cuenta que $F(x) = 1 - F(-x)$ para cada $x \in \mathbb{R}$, y que en consecuencia $F(0) = 0.5$, es suficiente tabular la función

$$f : [0, +\infty) \longrightarrow \mathbb{R}, \quad f(x) = \frac{1}{\sqrt{2\pi}} \int_{0}^{x} \mathrm{e}^{-\frac{t^2}{2}}\, dt.$$

Para aproximar $f(x)$ en los puntos x no contenidos en la tabla, se hará una interpolación cúbica eligiendo los cuatro nodos de la tabla más próximos a x. La precisión que se desea en el resultado con este modo de proceder es de 10^{-6}. Si tenemos en cuenta que $f(x) - f(5) < 3 \times 10^{-7}$ para cada $x > 5$, será suficiente tabular la función f en el intervalo $[0, 5]$. Por razones de simplicidad los nodos de la tabla $\{x_0, x_1, \ldots, x_n\}$ se tomarán igualmente espaciados. Halla el número de nodos que son necesarios para obtener un error inferior al señalado por el procedimiento indicado. Además, determina el error que se cometería si tomásemos $n = 100$ y realizásemos una interpolación lineal en lugar de cúbica para aproximar los valores de $f(x)$ en los puntos x ausentes de la tabla.

10. Dada la función de Runge

$$f : [-5, +5] \longrightarrow \mathbb{R}, \quad f(x) = \frac{1}{1+x^2},$$

Halla los polinomios interpoladores de Lagrange de grado 10 y 20, respectivamente, que interpolan a f en puntos igualmente espaciados del intervalo $[-5, +5]$, con $x_0 = -5$ y $x_n = +5$. Haz un plot de la función y los correspondientes polinomios interpoladores.

11. Halla el polinomio de grado menor o igual a 20 mejor aproximación en el sentido de mínimos cuadrados de la función de Runge tomando como datos los valores de f en los nodos $x_j = -5 + jh$, $j = 0, \ldots, 100$ y $h = \frac{1}{10}$. Repite el ejercicio para la función $f(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}}$.

12. Halla el polinomio $p(x)$ de grado menor o igual a 10 mejor aproximación en el sentido de mínimos cuadrados de la función de $f : [0, 1] \longrightarrow \mathbb{R}$, $f(x) = \mathrm{e}^x \operatorname{sen}^3(1 + x)$, tomando como datos los valores de f en los nodos $x_j = jh$, $j = 0, \ldots, 100$ y $h = 0.01$. Calcula $\displaystyle\int_0^1 p(x)\, dx$.

Capítulo 4

Integración Numérica

4.1 Introducción

En este capítulo estudiamos el siguiente problema: dada una función continua $f : [a, b] \longrightarrow \mathbb{R}$, determinar una aproximación de la integral $\int_a^b f(x)\,dx$. La primera idea se basa en considerar una aproximación polinomial de la función f. A tal efecto se consideran $n+1$ puntos $\{x_0, x_1, \ldots, x_n\}$ diferentes del intervalo $[a, b]$ y se construye el polinomio interpolador de Lagrange $p_n \in \mathcal{P}_n$ de la función f relativa a los nodos $\{x_j\}_{j=0}^n$. Entonces, consideramos la aproximación $\int_a^b f(x)\,dx \approx \int_a^b p_n(x)\,dx$. De acuerdo a la formula (3.1), se tiene

$$\int_a^b f(x)\,dx \approx \int_a^b p_n(x)\,dx = \sum_{j=0}^{n} \left(\int_a^b l_j(x)\,dx \right) f(x_j),$$

donde las funciones $\{l_j\}_{j=0}^n$ son los polinomios de base de Lagrange. Si definimos

$$\omega_j = \frac{1}{b-a} \int_a^b l_j(x)\,dx \quad 0 \leq j \leq n, \tag{4.1}$$

entonces tenemos

$$\int_a^b f(x)\,dx \approx (b-a) \sum_{j=0}^{n} \omega_j f(x_j). \tag{4.2}$$

En la fórmula (4.2), los números ω_j se llaman pesos o coeficientes de la fórmula y los puntos $\{x_j\}_{j=0}^n$ son los nodos. Una fórmula del tipo (4.2) en la que los pesos vienen dados por la expresión (4.1), se dice que es una fórmula de integración numérica de tipo interpolatorio.

Las fórmulas del tipo (4.2) son eficientes si el intervalo $[a, b]$ es de longitud pequeña. En caso de intervalos de longitud grande, se descompone el mismo en subintervalos pequeños y se aplica la misma fórmula (o a veces distinta fórmula) a cada uno de los subintervalos. Consideremos una partición del intervalo $[a, b]$:

$a = \alpha_0 < \alpha_1 < \alpha_2 < \ldots < \alpha_k = b$, entonces

$$\int_a^b f(x)\,dx = \sum_{i=1}^{k} \int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx.$$

Aplicando la fórmula (4.2) a cada uno de los subintervalos $[\alpha_{i-1}, \alpha_i]$ obtenemos

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{k} (\alpha_i - \alpha_{i-1}) \sum_{j=0}^{n} \omega_j f\Big(\alpha_{i-1} + \frac{\alpha_i - \alpha_{i-1}}{b-a}(x_j - a)\Big), \qquad (4.3)$$

donde los nodos $t_j = \alpha_{i-1} + \frac{\alpha_i - \alpha_{i-1}}{b-a}(x_j - a)$, $0 \le j \le n$, son obtenidos mediante las transformaciones lineales

$$\begin{array}{rcll} [a,b] & \longrightarrow & [\alpha_{i-1}, \alpha_i] & \\ x & \longrightarrow & t = \alpha_{i-1} + \dfrac{\alpha_i - \alpha_{i-1}}{b-a}(x-a) & 1 \le i \le k. \end{array}$$

Las fórmulas (4.2) son fórmulas de cuadratura elementales y las fórmulas (4.3) son fórmulas de cuadratura compuestas.

En la sección 4.2 vamos a estudiar las fórmulas elementales: su construcción y el error asociado. Nos limitaremos al estudio de las fórmulas de Newton-Cotes. En la sección 4.3 veremos cómo construir fórmulas de cuadratura compuestas para aproximar la integral con un error inferior a la precisión deseada. A tal efecto, usaremos el método de mallado adaptativo, que consiste en determinar la partición $a = \alpha_0 < \alpha_1 < \alpha_2 < \ldots < \alpha_k = b$ en función del máximo error requerido en la aproximación de la integral.

Referencias apropiadas para los contenidos de este capítulo son los textos [8] y [9].

4.2 Fórmulas de Cuadratura de Newton-Cotes

En esta sección nos limitaremos al estudio de fórmulas de cuadratura del tipo (4.2) donde los nodos vienen dados por las expresiones

$$x_j = a + jh, \quad 0 \le j \le n, \quad \text{y} \quad h = \frac{b-a}{n}. \qquad (4.4)$$

Así tenemos que $a = x_0 < x_1 < x_2 < \ldots < x_n = b$ y los nodos son equidistantes: $h = x_j - x_{j-1}$ para cada $j = 1, \ldots, n$.

DEFINICIÓN 4.1 *Se llama fórmula de Newton-Cotes (también fórmula cerrada de Newton-Cotes) a aquella fórmula de cuadratura numérica elemental* (4.2) *de tipo interpolatorio, donde los nodos vienen dados por las expresiones* (4.4). *Recordamos aquí que la fórmula* (4.2) *se dice de tipo interpolatorio si los pesos* $\{\omega_j\}_{j=0}^n$ *están definidos por las identidades* (4.1).

De cara a la construcción de las fórmulas de Newton-Cotes, si seguimos las fórmulas (4.1), necesitamos construir los polinomios de base de Lagrange $\{l_j\}_{j=0}^n$ y seguidamente realizar las integrales (4.1). Ésta es una tarea penosa cuando el número de nodos crece. Vamos a ver un método alternativo para la construcción de estas fórmulas.

DEFINICIÓN 4.2 *Una fórmula de cuadratura elemental del tipo* (4.2) *se dice exacta en* $\mathcal{P}_m$ *si*

$$\int_a^b p(x)\,dx = (b-a)\sum_{j=0}^n \omega_j p(x_j) \quad \forall p \in \mathcal{P}_m. \tag{4.5}$$

Se dice que la fórmula (4.2) *posee grado de precisión* m *si es exacta en* $\mathcal{P}_m$ *pero no lo es en* $\mathcal{P}_{m+1}$.

TEOREMA 4.3 *La fórmula* (4.2) *es de tipo interpolatorio si y sólo si es exacta en* $\mathcal{P}_n$.

Demostración. Supongamos primero que la fórmula es de tipo interpolatorio y probemos que entonces es exacta en $\mathcal{P}_n$. Sea $p \in \mathcal{P}_n$, demostraremos que la igualdad (4.5) se cumple. Para ello observemos que

$$p(x) = \sum_{j=0}^n p(x_j) l_j(x). \tag{4.6}$$

En efecto, tomemos el polinomio $q(x) = p(x) - \sum_{j=0}^n p(x_j) l_j(x)$. Entonces, q es un polinomio de grado menor o igual que n y, de acuerdo a (3.2),

$$q(x_i) = p(x_i) - \sum_{j=0}^n p(x_j) l_j(x_i) = p(x_i) - p(x_i) = 0 \quad \text{para cada} \quad i = 0, 1, \ldots, n.$$

El único polinomio de grado $\leq n$ anulándose en $n+1$ puntos es el polinomio nulo. Por lo tanto $q = 0$, o lo que es lo mismo la igualdad (4.6) se cumple. Estamos suponiendo que la fórmula es de tipo interpolatorio, lo que significa que las identidades (4.1) se satisfacen. Entonces, usando la linealidad de la integral, tenemos

$$\int_a^b p(x)\,dx = \sum_{j=0}^n p(x_j) \int_a^b l_j(x)\,dx = \sum_{j=0}^n p(x_j)(b-a)\omega_j = (b-a)\sum_{j=0}^n \omega_j p(x_j).$$

Esto demuestra que la fórmula (4.2) es exacta en $\mathcal{P}_n$. Probemos el reccíproco. Para ellos suponemos que la fórmula es exacta en $\mathcal{P}_n$ y debemos probar que las identidades (4.1) se satisfacen. Dado que los polinomios $\{l_j\}_{j=0}^n$ son todos ellos

de grado n, usando que la fórmula (4.2) es exacta en $\mathcal{P}_n$ y las igualdades (3.2), se obtiene

$$\int_a^b l_k(x)\,dx = (b-a)\sum_{j=0}^{n}\omega_j l_k(x_j) = (b-a)\omega_k,\ 0\le k\le n,$$

lo que prueba las identidades (4.1). □

Este teorema sugiere un modo alternativo a las fórmulas (4.1) para calcular los pesos $\{\omega_j\}_{j=0}^n$. Efectivamente, dados los nodos $\{x_j\}_{j=0}^n$, la fórmula (4.2) es de tipo interpolatorio si y sólo si es exacta en $\mathcal{P}_n$. Puesto que los polinomios $p_k(x) = x^k$ son de grado $\le n$ para cada $k = 0, 1, \ldots, n$, esto implica que

$$\frac{b^{k+1}-a^{k+1}}{k+1} = \int_a^b x^k\,dx = (b-a)\sum_{j=0}^{n}\omega_j x_j^k \quad \text{para} \quad k = 0, 1, \ldots, n. \tag{4.7}$$

Como los nodos $\{x_j\}_{j=0}^n$ son conocidos, las igualdades anteriores constituyen un sistema lineal de $n+1$ ecuaciones y $n+1$ incógnitas, que son los pesos $\{\omega_j\}_{j=0}^n$. Resolver el sistema (4.7) es computacionalmente más sencillo que aplicar las fórmulas (4.1) para determinar los pesos $\{\omega_j\}_{j=0}^n$. Pero todavía vamos a simplificar más los cálculos. Si queremos hallar una fórmula de cuadratura para un intervalo genérico $[a, b]$, el sistema anterior todavía resulta muy complicado para valores de n grandes. Observemos que, fijados los nodos, los pesos quedan unívocamente determinados por las fórmulas (4.1), por lo tanto solamente hay una y sólo una fórmula exacta en $\mathcal{P}_n$ para dichos pesos. Consecuentemente, el sistema anterior posee una única solución.

Vamos a ver que los pesos $\{\omega_j\}_{j=0}^n$ son independientes del intervalo siempre que elijamos convenientemente los nodos. Sea $[c, d]$ un intervalo y consideremos la tranformación lineal

$$\begin{array}{ccc} [a,b] & \longrightarrow & [c,d] \\ x & \longrightarrow & t = c + \dfrac{d-c}{b-a}(x-a) \end{array}. \tag{4.8}$$

Dados los nodos $\{x_j\}_{j=0}^n \subset [a, b]$, consideramos sus transformados por la aplicación anterior

$$t_j = c + \frac{d-c}{b-a}(x_j - a), \quad 0 \le j \le n. \tag{4.9}$$

Ahora deseamos construir una fórmula de tipo interpolatorio en $[c, d]$ asociada a los nodos $\{t_j\}_{j=0}^n$:

$$\int_c^d g(t)\,dt \approx (d-c)\sum_{j=0}^{n}\hat{\omega}_j g(t_j).$$

Los pesos vienen dados por las fórmulas

$$\hat{\omega}_j = \frac{1}{d-c}\int_c^d \hat{l}_j(t)\,dt \quad \text{donde} \quad \hat{l}_j(t) = \prod_{\substack{i=0 \\ j\ne i}}^{n} \frac{t-t_i}{t_j-t_i}. \tag{4.10}$$

Los polinomios $\{\hat{l}_j\}_{j=0}^n$ son los polinomios de base de Lagrange asociados a los nodos $\{t_j\}_{j=0}^n$. Vamos a comprobar que los pesos $\{\hat{\omega}_j\}_{j=0}^n$ coinciden con $\{\omega_j\}_{j=0}^n$. Para ello observemos que de acuerdo a (4.8) y (4.9) se tiene

$$t - t_i = \frac{d-c}{b-a}(x - x_i) \quad \text{y} \quad t_j - t_i = \frac{d-c}{b-a}(x_j - x_i).$$

De aquí se sigue

$$\hat{l}_j(t) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{t - t_i}{t_j - t_i} = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{\frac{d-c}{b-a}(x - x_i)}{\frac{d-c}{b-a}(x_j - x_i)} = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{x - x_i}{x_j - x_i} = l_j(x).$$

Ahora, haciendo el cambio de variable (4.8) en la integral (4.10) y usando que $dt = \frac{d-c}{b-a}dx$ se deduce para cada $j = 0, 1, \ldots, n$

$$\hat{\omega}_j = \frac{1}{d-c}\int_c^d \hat{l}_j(t)\,dt = \frac{1}{d-c}\int_a^b l_j(x)\,\frac{d-c}{b-a}dx = \frac{1}{b-a}\int_a^b l_j(x)\,dx = \omega_j.$$

Hemos comprobado que los pesos $\{\omega_j\}_{j=0}^n$ son independientes del intervalo siempre que los nodos vengan relacionados por las transformaciones (4.8).

Vamos a utilizar lo expuesto hasta ahora para calcular las primeras fórmulas de Newton-Cotes.

Primera fórmula de Newton-Cotes ($n = 1$): de acuerdo a (4.4), los nodos deben ser $x_0 = a$ y $x_1 = b$. Vamos a calcular los pesos en el intervalo $[0, 1]$, donde los nodos son los extremos del intervalo. Para determinar los pesos planteamos el sistema (4.7) con $a = 0$ y $b = 1$:

$$\begin{cases} k = 0: & 1 & = \displaystyle\int_0^1 dx & = \omega_0 + \omega_1 \\ k = 1: & \frac{1}{2} & = \displaystyle\int_0^1 x\,dx & = \omega_1 \end{cases}$$

La solución del sistema es: $\omega_0 = \omega_1 = \frac{1}{2}$. Así tenemos la fórmula de Newton-Cotes de dos puntos, llamada *fórmula del trapecio*,

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}[f(a) + f(b)].$$

Segunda fórmula de Newton-Cotes ($n = 2$): utilizando de nuevo (4.4), se obtienen los nodos $x_0 = a$, $x_1 = \frac{a+b}{2}$ y $x_2 = b$. El cálculo de los pesos lo haremos en el intervalo $[-1, +1]$ donde los correspondientes nodos son $\{-1, 0, +1\}$.

Tomando $a = -1$ y $b = +1$ en (4.7) obtenemos

$$\begin{cases} k=0: & 2 & = \displaystyle\int_{-1}^{+1} dx & = 2[\omega_0 + \omega_1 + \omega_2] \\ k=1: & 0 & = \displaystyle\int_{-1}^{+1} x\,dx & = 2[-\omega_0 + \omega_2] \\ k=2: & \frac{2}{3} & = \displaystyle\int_{-1}^{+1} x^2\,dx & = 2[\omega_0 + \omega_2] \end{cases}$$

La solución del sistema es: $\omega_0 = \frac{1}{6}$, $\omega_1 = \frac{4}{6}$ y $\omega_2 = \frac{1}{6}$. Consecuentemente, la fórmula de Newton-Cotes de tres nodos, llamada *fórmula de Simpson*, es

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}[f(a) + 4f\big(\frac{a+b}{2}\big) + f(b)].$$

Siguiendo este procedimiento se pueden obtener las siguientes fórmulas de Newton-Cotes:

Regla Trapezoidal $(n = 1)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}[f(a) + f(b)]$$

Regla de Simpson $(n = 2)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}[f(a) + 4f\big(\frac{a+b}{2}\big) + f(b)]$$

Regla $\frac{3}{8}$ $(n = 3)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{8}[f(a) + 3f(\frac{2a+b}{3}) + 3f(\frac{a+2b}{3}) + f(b)]$$

Regla de Milne o Boole-Villarceau $(n = 4)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{90}[7f(a) + 32f(\frac{3a+b}{4}) + 12f(\frac{a+b}{2}) + 32f(\frac{a+3b}{4}) + 7f(b)]$$

$(n = 5)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{288}[19f(a) + 75f(\frac{4a+b}{5}) + 50f(\frac{3a+2b}{5}) + 50f(\frac{2a+3b}{5}) + 75f(\frac{a+4b}{5}) + 19f(b)]$$

Regla de Weddle o Hardy $(n = 6)$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{840}[41f(a) + 216f(\frac{5a+b}{6}) + 27f(\frac{2a+b}{3}) + 272f(\frac{a+b}{2}) + 27f(\frac{a+2b}{3}) + 216f(\frac{a+5b}{6}) + 41f(b)]$$

Como ejercicio, el lector puede deducir estas fórmulas utilizando el procedimiento anteriormente descrito. Se debe notar que hay una simetría en los pesos: el primero es igual al último, el segundo es igual al penúltimo, etc.

Terminamos esta sección analizando el error de las fórmulas de cuadratura de Newton-Cotes. El resultado básico es el siguiente; ver [8, Cap. 7-§1.1].

TEOREMA 4.4 *Supongamos que* (4.2) *una fórmula de Newton-Cotes. Si la función* $f : [a, b] \longrightarrow \mathbb{R}$ *es* $n+2$ *veces derivable y* $f^{(n+2)}$ *es una función continua cuando* n *es par, o* f *es* $n+1$ *veces derivable y* $f^{(n+1)}$ *es una función continua cuando* n *es impar, entonces existe un número* $\xi \in [a, b]$ *tal que el error de la fórmula viene dado por la siguiente expresión*

$$\begin{aligned} &\int_a^b f(x)\,dx - (b-a)\sum_{j=0}^{n} \omega_j f(x_j) \\ &= \begin{cases} \dfrac{h^{n+3}}{(n+2)!}\displaystyle\int_a^b x\pi(x)\,dx\, f^{(n+2)}(\xi) & \text{si } n \text{ es par,} \\ \dfrac{h^{n+2}}{(n+1)!}\displaystyle\int_a^b \pi(x)\,dx\, f^{(n+1)}(\xi) & \text{si } n \text{ es impar,} \end{cases} \end{aligned} \tag{4.11}$$

donde $h = \frac{b-a}{n}$ *y* $\pi(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$.

Como consecuencia del teorema anterior se deduce que si n es par, entonces el grado de precisión de la fórmula de Newton-Cotes (4.2) es $n + 1$. En efecto, si $f(x) = p(x)$, siendo p un polinomio de grado $n + 1$, entonces $f^{(n+2)}(x) = 0$ para cada $x \in \mathbb{R}$. Consecuentemente, el error de la fórmula es cero, es decir que la fórmula es exacta para los polinomio de grado $\leq n + 1$. Sin embargo, para un polinomio de grado $n + 2$, la derivada $f^{(n+2)}(x) = (n + 2)!$ y el error no es cero. Cuando n es impar, sabemos del Teorema 4.3 que las fórmulas de Newton-Cotes son exactas en $\mathcal{P}_n$, lo que también se deduce de la fórmula del error (4.11). Además si $f(x) = p(x)$ es un polinomio de grado $n + 1$, entonces $f^{(n+1)}(x) = (n+1)!$ y (4.11) muestra que el error no es nulo. Así, para n impar el grado de precisión de las fórmulas de Newton-Cotes es n.

Según el Teorema 4.4, el error en las fórmula de Newton-Cotes es de la forma

$$\int_a^b f(x)\,dx - (b-a)\sum_{j=0}^{n} \omega_j f(x_j) = C_n h^{m+1} f^{(m)}(\xi). \tag{4.12}$$

donde

$$m = \begin{cases} n+2 & \text{si } n \text{ es par} \\ n+1 & \text{si } n \text{ es impar} \end{cases} \quad \text{y } C_n = \begin{cases} \dfrac{1}{(n+2)!}\displaystyle\int_a^b x\pi(x)\,dx & \text{si } n \text{ es par,} \\ \dfrac{1}{(n+1)!}\displaystyle\int_a^b \pi(x)\,dx & \text{si } n \text{ es impar.} \end{cases}$$

Vamos a calcular la constante C_n para cada una de las fórmulas de Newton-Cotes. Para ello tomamos la función $f(x) = (x - a)^m$ y realizamos los cálculos

siguientes

$$\int_a^b f(x)\,dx = \int_a^b (x-a)^m\,dx = \frac{1}{m+1}(b-a)^{m+1} = \frac{n^{m+1}h^{m+1}}{m+1},$$
$$(b-a)\sum_{j=0}^{n}\omega_j f(x_j) = nh\sum_{j=0}^{n}\omega_j(x_j-a)^m = nh\sum_{j=0}^{n}\omega_j(jh)^m = nh^{m+1}\sum_{j=1}^{n}\omega_j j^m,$$
$$C_n h^{m+1} f^{(m)}(\xi) = C_n h^{m+1} m!,$$

donde hemos utilizado que, de acuerdo a (4.4), los nodos $\{x_j\}_{j=0}^n$ de las fórmulas de Newton-Cotes vienen dados por las expresiones $x_j = a + jh$ para $0 \le j \le n$ y $h = \frac{b-a}{n}$. De las expresiones obtenidas y (4.12) se sigue

$$\frac{n^{m+1}h^{m+1}}{m+1} - nh^{m+1}\sum_{j=1}^{n}\omega_j j^m = C_n h^{m+1} m!,$$

de donde

$$C_n = \frac{1}{m!}\left(\frac{n^{m+1}}{m+1} - n\sum_{j=1}^{n}\omega_j j^m\right). \tag{4.13}$$

Vamos a deducir el error para las primeras fórmulas de cuadratura que hemos visto antes.

Fórmula del trapecio ($n = 1$)*:* en este caso n es impar, por lo tanto $m = n + 1 = 2$. Teniendo en cuenta que $\omega_1 = \frac{1}{2}$, (4.13) implica

$$C_1 = \frac{1}{2}(\frac{1}{3} - \frac{1}{2}) = -\frac{1}{12}.$$

Usando (4.12) se obtiene

$$\int_a^b f(x)\,dx = \frac{b-a}{2}[f(a) + f(b)] - \frac{h^3}{12}f''(\xi)$$

para algún punto $\xi \in [a, b]$.

Fórmula de Simpson ($n = 2$)*:* como n es par tenemos que $m = n + 2 = 4$. Teniendo en cuenta que $\omega_1 = \frac{4}{6}$ y $\omega_2 = \frac{1}{6}$, deducimos de (4.13)

$$C_2 = \frac{1}{4!}\Big(\frac{2^5}{5} - 2[\frac{4}{6} + \frac{1}{6}2^4]\Big) = -\frac{1}{90}.$$

Una vez más (4.12) conduce a

$$\int_a^b f(x)\,dx = \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)] - \frac{h^5}{90}f^{(4)}(\xi).$$

Siguiendo este procedimiento se obtienen las siguientes fórmulas de Newton-Cotes junto con su error

$$\int_a^b f(x)\,dx = \frac{b-a}{2}[f(a)+f(b)] - \frac{h^3}{12}f''(\xi)$$

$$\int_a^b f(x)\,dx = \frac{b-a}{6}[f(a)+4f(\frac{a+b}{2})+f(b)] - \frac{h^5}{90}f^{(4)}(\xi)$$

$$\int_a^b f(x)\,dx = \frac{b-a}{8}[f(a)+3f(\frac{2a+b}{3})+3f(\frac{a+2b}{3})+f(b)] - \frac{3h^5}{80}f^{(4)}(\xi)$$

$$\int_a^b f(x)\,dx = \frac{b-a}{90}[7f(a)+32f(\frac{3a+b}{4})+12f(\frac{a+b}{2})+32f(\frac{a+3b}{4})+7f(b)] - \frac{8h^7}{945}f^{(6)}(\xi)$$

$$\int_a^b f(x)\,dx = \frac{b-a}{288}[19f(a)+75f(\frac{4a+b}{5})+50f(\frac{3a+2b}{5})+50f(\frac{2a+3b}{5}) + 75f(\frac{a+4b}{5})+19f(b)] - \frac{275h^7}{12096}f^{(6)}(\xi)$$

$$\int_a^b f(x)\,dx = \frac{b-a}{840}[41f(a)+216f(\frac{5a+b}{6})+27f(\frac{2a+b}{3})+272f(\frac{a+b}{2}) + 27f(\frac{a+2b}{3})+216f(\frac{a+5b}{6})+41f(b)] - \frac{9h^9}{1400}f^{(8)}(\xi)$$

Las fórmulas de Newton-Cotes con n par son preferidas, en general, a las correspondientes a valores de n impar. En efecto, una fórmula con $n+1$ nodos y n par posee grado de precisión $n+1$, mientras que la fórmula con $n+2$ nodos tiene el mismo grado de precisión y requiere una evaluación de función adicional. Por otro lado, se observará que los pesos en las fórmulas anteriores son todos mayores que cero. Esta propiedad la conservan las fórmulas para $n=7$ y $n=9$, pero n es impar en estos casos. En las fórmulas de Newton-Cotes con $n=8$ y $n \geq 10$ los pesos no mantienen el signo constante, siendo unos positivos y otros negativos, lo que introduce un error de cancelación en la aplicación de las fórmulas, que puede ser importante. Por otro lado, no es aconsejable utilizar fórmulas de Newton-Cotes con valores de n grandes porque el fenómeno de Runge puede ocurrir; ver sección 3.4.

4.3 Fórmulas Compuestas. Métodos Adaptativos

Si se analiza el error de las fórmulas de Newton-Cotes que hemos descrito antes, se observa que éste se hace pequeño si la distancia entre los nodos h es pequeña, o lo que es equivalente si la longitud del intervalo de integración $b-a$ es pequeña. Para intervalos de longitud grande ($h \gg 1$) el error puede ser grande. Por ello, las fórmulas de Newton-Cotes se aplican en intervalos de longitud pequeña.

Para ello, como ya indicamos en la introducción del capítulo, se considera una partición del intervalo $[a,b]$: $a=\alpha_0<\alpha_1<\alpha_2<\ldots<\alpha_k=b$, entonces

$$\int_a^b f(x)\,dx=\sum_{i=1}^{k}\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx.$$

A continuación se aplica la fórmula de cuadratura (4.2) a cada uno de los subintervalos $[\alpha_{i-1},\alpha_i]$:

$$\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx\approx(\alpha_i-\alpha_{i-1})\sum_{j=0}^{n}\omega_j f\Big(\alpha_{i-1}+\frac{\alpha_i-\alpha_{i-1}}{b-a}(x_j-a)\Big).$$

De acuerdo a (4.12) el error de esta fórmula viene dado por

$$\begin{aligned}E_i=&\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx-(\alpha_i-\alpha_{i-1})\sum_{j=0}^{n}\omega_j f\Big(\alpha_{i-1}+\frac{\alpha_i-\alpha_{i-1}}{b-a}(x_j-a)\Big)\\ =&C_n h_i^{m+1} f^{(m)}(\xi_i),\end{aligned}\tag{4.14}$$

donde $m-1$ es el grado de precisión de la fórmula, C_n es la constante solamente dependiente de n definida en (4.13), $h_i=\frac{\alpha_i-\alpha_{i-1}}{n}$ y ξ_i es un punto del intervalo $[\alpha_{i-1},\alpha_i]$. Por lo tanto, si consideramos la aproximación de la integral

$$\int_a^b f(x)\,dx\approx\sum_{i=1}^{k}(\alpha_i-\alpha_{i-1})\sum_{j=0}^{n}\omega_j f\Big(\alpha_{i-1}+\frac{\alpha_i-\alpha_{i-1}}{b-a}(x_j-a)\Big),\tag{4.15}$$

entonces el error que cometemos es la suma de los errores E_i

$$E=\sum_{i=1}^{k}E_i=\sum_{i=1}^{k}C_n h_i^{m+1}f^{(m)}(\xi_i)=\frac{C_n}{n^{m+1}}\sum_{i=1}^{k}(\alpha_i-\alpha_{i-1})^{m+1}f^{(m)}(\xi_i).\tag{4.16}$$

Denotando por

$$\hat{h}=\max_{1\le i\le k}(\alpha_i-\alpha_{i-1})$$

y usando que, gracias a la continuidad de la función $f^{(m)}:[a,b]\longrightarrow\mathbb{R}$, existe un punto $\xi\in[a,b]$ tal que

$$\sum_{i=1}^{k}(\alpha_i-\alpha_{i-1})|f^{(m)}(\xi_i)|=(b-a)|f^{(m)}(\xi)|,$$

se deduce de (4.16)

$$|E|\le\frac{C_n}{n^{m+1}}\hat{h}^m\sum_{i=1}^{k}(\alpha_i-\alpha_{i-1})|f^{(m)}(\xi_i)|=\frac{C_n(b-a)}{n^{m+1}}\hat{h}^m|f^{(m)}(\xi)|.\tag{4.17}$$

Esta estimación del error demuestra que el error tiende a cero cuando las longitudes de los intervalos $[\alpha_{i-1}, \alpha_i]$ tienden a cero. Por lo tanto, podemos usar fórmulas compuestas (4.15) para conseguir una buena aproximación de la integral.

Ahora nos planteamos la siguientes cuestiones: ¿cuántos intervalos $[\alpha_{i-1}, \alpha_i]$ debemos tomar para tener un error inferior a una cierta cantidad deseada ε?, ¿cómo elegir los intervalos? Para responder a estas cuestiones necesitamos un procedimiento computacionalmente eficiente para estimar los errores E_i que se cometen en cada aproximación. Vamos describir un procedimiento para realizar estas estimaciones. Sea

$$\int_a^b f(x)\,dx = (b-a)\sum_{j=0}^{n}\omega_j f(x_j) + C_n h^{m+1} f^{(m)}(\xi) = I + E,$$

donde $h = \frac{b-a}{n}$, $\xi \in [a,b]$, I es la aproximación de la integral y E es el error cometido al considerar esta aproximación. Supondremos que la longitud del intervalo $(b-a)$ es pequeña. Tenemos que estimar E. Para ello vamos a aplicar la misma fórmula en los subintervalos $[a, \frac{a+b}{2}]$ y $[\frac{a+b}{2}, b]$:

$$\int_a^{\frac{a+b}{2}} f(x)\,dx = \frac{b-a}{2}\sum_{j=0}^{n}\omega_j f\big(\frac{a+x_j}{2}\big) + C_n\Big(\frac{h}{2}\Big)^{m+1} f^{(m)}(\xi_1) = I_1 + E_1,$$

$$\int_{\frac{a+b}{2}}^b f(x)\,dx = \frac{b-a}{2}\sum_{j=0}^{n}\omega_j f\big(\frac{b+x_j}{2}\big) + C_n\Big(\frac{h}{2}\Big)^{m+1} f^{(m)}(\xi_2) = I_2 + E_2,$$

donde $\xi_1 \in [a, \frac{a+b}{2}]$ y $\xi_2 \in [\frac{a+b}{2}, b]$. Usando la continuidad de la función $f^{(m)}$ deducimos la existencia de un punto $\eta \in [a,b]$ tal que

$$E_1 + E_2 = \frac{C_n h^{m+1}}{2^m}\big[\frac{1}{2}f^{(m)}(\xi_1) + \frac{1}{2}f^{(m)}(\xi_2)\big] = \frac{C_n h^{m+1}}{2^m} f^{(m)}(\eta).$$

Puesto que $\xi, \eta \in [a,b]$ y éste es un intervalo de longitud pequeña, entonces $\xi \approx \eta$, y consecuentemente $f^{(m)}(\eta) \approx f^{(m)}(\xi)$. Por lo tanto, se tiene que

$$E_1 + E_2 \approx \frac{C_n h^{m+1}}{2^m} f^{(m)}(\xi) = \frac{1}{2^m} E.$$

Usando esta aproximación y la identidad $I + E = (I_1 + E_1) + (I_2 + E_2)$ se deduce

$$E = (I_1 + E_1) + (I_2 + E_2) - I \approx (I_1 + I_2 - I) + \frac{1}{2^m}E \Rightarrow \big(1 - \frac{1}{2^m}\big)E \approx I_1 + I_2 - I,$$

y por lo tanto

$$|E| \approx \left|\frac{2^m}{2^m - 1}(I_1 + I_2 - I)\right|. \tag{4.18}$$

Notemos que (4.18) nos proporciona una estimación del error que cometemos al tomar I como aproximación de la integral. Sin embargo, ya que hemos

calculado I_1 y I_2 para estimar E y teniendo en cuenta que I_1+I_2 nos proporciona una mejor aproximación de la integral que I, nos quedaremos con $I_1 + I_2$ como aproximación de la integral. En este caso, el error que estamos cometiendo es inferior a $|E|$. Por lo tanto, (4.18) nos proporciona una cota superior del error que cometemos al considerar $I_1 + I_2$ como aproximación de la integral.

La siguiente función MATLAB simplemente implementa la fórmula de Hardy.

```
function y=hardy(a,b,f)
x=linspace(a,b,7)';
w=[41 216 27 272 27 216 41];
fx=feval(f,x);
y=(b-a)/840*(w*fx);
end
```

Para usar esta función MATLAB, la función f debe programarse vectorialmente de forma que si $x = (x_i)$ es un vector, entonces $f(x)$ es el vector "columna" cuyas componentes son los valores $f(x_i)$.

Ahora la función siguiente implementa la fórmula de Hardy y proporciona una estimación del error cometido.

```
function [Itg,error]=iehardy(a,b,f)
c=(a+b)/2;
i=hardy(a,b,f);
i1=hardy(a,c,f);
i2=hardy(c,b,f);
Itg=i1+i2;
error=(256/255)*abs(Itg-i);
end
```

Se debe notar que el error es positivo debido a que se ha tomado el valor absoluto de Itg - i.

Fijada una precisión ε deseada en la integral, vamos a describir la estrategia que seguiremos para obtener la aproximación con la precisión deseada.

1. Elegimos una fórmula de Newton-Cotes para aproximar la integral en $[a, b]$.

2. Subdividimos el intervalo $[a, b]$ en dos subintervalos de la misma longitud y calculamos aproximaciones de las integrales I_1 y I_2 en ambos intervalos, así como las correspondientes estimaciones de los errores E_1 y E_2. Definimos los vector $I = [I_1, I_2]$ y $E = [E_1, E_2]$.

3. Si la suma de los errores $sum(E)$ es inferior a ε, tomamos la suma de las aproximaciones I como valor aproximado de la integral $Itg = sum(I)$ y hemos terminado. Si al contrario, $sum(E) \geq \varepsilon$, entonces vamos al paso 4.

4. Buscamos el error E_k más grande de los contenidos en el vector E. Subdividimos el intervalo asociado a E_k en dos mitades y calculamos las aproximaciones de la integral en cada una de estas mitades, así como los correspondientes errores. Eliminamos I_k y E_k de los vectores I y E y ponemos

en su lugar los dos valores de la integral calculados y los correspondientes errores. Volvemos al paso 3.

Con el algoritmo anterior, los vectores I y E van creciendo en base a ir tomando subdivisiones de alguno de los intervalos. En cada iteración, el criterio es subdividir el intervalo sobre el cual el error de la integral es más grande, eliminar los correspondientes valores I_k y E_k de I y E, respectivamente, y poner en su lugar las dos nuevas aproximaciones de la integral y sus estimaciones del error. Este procedimiento se conoce como mallado adaptativo.

En ocasiones, la precisón deseada en la solución se basa en el error relativo de la aproximación de la integral en lugar del error absoluto. En el caso de considerar el error absoluto el algoritmo se detiene si $E < \varepsilon$. Sin embargo, cuando se considera el error relativo, el algoritmo se detendrá si $E < \varepsilon|I|$. Finalmente, es muy frecuente considerar como test de parada una combinación de error absoluto y relativo: $E < \varepsilon_a + \varepsilon_r|I|$. En este caso, usualmente, si la integral es un número grande, el algoritmo se detiene cuando del error relativo es inferior a ε_r. Mientras que si $|I|$ es pequeño, el algoritmo finaliza cuando el error absoluto es inferior a ε_a.

4.4 Cálculo de Integrales Impropias

Hasta ahora hemos estudiado la aproximación de integrales en un intervalo $[a, b]$ acotado, $-\infty < a < b < +\infty$, de una función continua f. En esta sección veremos como abordar el caso en que a o b o ambos no son finitos (integrales impropias de primera especie) y el caso en que f tenga una singularidad en algún punto $x_0 \in [a, b]$ de tal forma que $\lim_{x \to x_0} |f(x)| = +\infty$ (integrales impropias de segunda especie).

4.4.1 Integrales Impropias de Primera Especie

Vamos a comenzar analizando la aproximación de la integral $\int_a^\infty f(x)\,dx$, donde $f : [a, \infty) \longrightarrow \mathbb{R}$ es continua. La integral se define en la forma siguiente

$$\int_a^\infty f(x)\,dx = \lim_{b \to \infty} \int_a^b f(x)\,dx.$$

Si el límite es $-\infty$ o $+\infty$ no hay nada que aproximar. Supondremos que la integral es convergente, es decir, el límite es un número real I. En este caso, deseamos calcular una aproximación Itg de I con una precisión determinada: $|I - Itg| < \varepsilon$. La forma general de proceder será la siguiente. Por definición de límite se tiene que

$$\lim_{b \to \infty} \int_b^\infty f(x)\,dx = 0.$$

Entonces, debemos hallar un valor de b tal que

$$\Big| \int_b^\infty f(x)\,dx \Big| < \frac{\varepsilon}{2}.$$

Seguidamente, calcularemos una aproximación de Itg de la integral en $[a, b]$ satisfaciendo

$$\Big| \int_a^b f(x)\,dx - Itg \Big| < \frac{\varepsilon}{2}.$$

Entonces se tiene

$$\Big| \int_a^\infty f(x)\,dx - Itg \Big| \leq \Big| \int_a^b f(x)\,dx - Itg \Big| + \Big| \int_b^\infty f(x)\,dx \Big| < \varepsilon.$$

De forma análoga se procederá con integrales del tipo $\int_{-\infty}^b f(x)\,dx$. En el caso de integrales de la forma $\int_{-\infty}^{+\infty} f(x)\,dx$ descomponemos la integral en dos partes:

$$\int_{-\infty}^{+\infty} f(x)\,dx = \int_{-\infty}^0 f(x)\,dx + \int_0^{+\infty} f(x)\,dx.$$

Cada una de las dos partes se aproximará con un error inferior a $\frac{\varepsilon}{2}$, así la suma sera una aproximación de la integral con un error inferior a ε. Alternativamente, se pueden calcular números $a < 0$ y $b > 0$ tales que

$$\Big| \int_{-\infty}^a f(x)\,dx \Big| < \frac{\varepsilon}{4} \quad \text{y} \quad \Big| \int_b^\infty f(x)\,dx \Big| < \frac{\varepsilon}{4}.$$

Seguidamente, se aproxima la integral $\int_a^b f(x)\,dx$ con un error inferior a $\frac{\varepsilon}{2}$.

A título de ejemplo vamos a aproximar la integral

$$\int_{-\infty}^{+\infty} \mathrm{e}^{-x^2} \operatorname{sen}(x + x^2)\,dx$$

con un error inferior a 10^{-14}. Vamos elegir $b > 0$ tal que

$$\Big| \int_b^{+\infty} \mathrm{e}^{-x^2} \operatorname{sen}(x + x^2)\,dx \Big| < \frac{10^{-14}}{4}.$$

Para ello procedemos como sigue

$$\Big| \int_b^{+\infty} \mathrm{e}^{-x^2} \operatorname{sen}(x + x^2)\,dx \Big| < \int_b^{+\infty} \mathrm{e}^{-x^2} |\operatorname{sen}(x + x^2)|\,dx < \int_b^{+\infty} \mathrm{e}^{-x^2}\,dx$$
$$< \int_b^{+\infty} \frac{x}{b} \mathrm{e}^{-x^2}\,dx = \Big[\frac{-\mathrm{e}^{-x^2}}{2b} \Big]_b^{+\infty} = \frac{\mathrm{e}^{-b^2}}{2b},$$

donde hemos usado que $\frac{x}{b} > 1$ para cada $x > b$. Tomamos b de forma que $\frac{\mathrm{e}^{-b^2}}{2b} < \frac{10^{-14}}{4}$ o equivalentemente $\frac{2\mathrm{e}^{-b^2}}{b} < 10^{-14}$. Simplemente evaluando en

Matlab para distintos valores de b tenemos

$$b = 5 \Rightarrow \frac{2\mathrm{e}^{-b^2}}{b} - 10^{-14} > 0,$$
$$b = 6 \Rightarrow \frac{2\mathrm{e}^{-b^2}}{b} - 10^{-14} < 0,$$
$$b = 5.5 \Rightarrow \frac{2\mathrm{e}^{-b^2}}{b} - 10^{-14} > 0,$$
$$b = 5.75 \Rightarrow \frac{2\mathrm{e}^{-b^2}}{b} - 10^{-14} < 0.$$

Así se tiene que $b = 5.75$ es aceptable. Análogamente determinamos $a < 0$:

$$\Big|\int_{-\infty}^{a} \mathrm{e}^{-x^2} \operatorname{sen}(x + x^2)\, dx\Big| < \int_{-\infty}^{a} \mathrm{e}^{-x^2} |\operatorname{sen}(x + x^2)|\, dx < \int_{-\infty}^{a} \mathrm{e}^{-x^2}\, dx$$
$$< \int_{-\infty}^{a} \frac{x}{a}\mathrm{e}^{-x^2}\, dx = \Big[\frac{-\mathrm{e}^{-x^2}}{2a}\Big]_{-\infty}^{a} = \frac{-\mathrm{e}^{-a^2}}{2a},$$

lo que conduce al valor $a = -5.75$. Por lo tanto, es suficiente determinar Itg como aproximación de la integral

$$\int_{-5.75}^{+5.75} \mathrm{e}^{-x^2} \operatorname{sen}(x + x^2)\, dx$$

con un error inferior a $\frac{1}{2}10^{-14}$.

4.4.2 Integrales Impropias de Segunda Especie

En este caso tenemos $-\infty < a < b < +\infty$, $x_0 \in [a, b]$ y $f : [a, b] \setminus \{x_0\} \longrightarrow \mathbb{R}$ es una función continua satisfaciendo que $\lim_{x \to x_0} |f(x)| = +\infty$. En este caso se define la integral como sigue:

$$\text{Si } x_0 = a \Rightarrow \int_a^b f(x)\, dx = \lim_{\delta \searrow 0} \int_{a+\delta}^{b} f(x)\, dx,$$
$$\text{Si } x_0 = b \Rightarrow \int_a^b f(x)\, dx = \lim_{\delta \searrow 0} \int_{a}^{b-\delta} f(x)\, dx,$$
$$\text{Si } a < x_0 < b \Rightarrow \int_a^b f(x)\, dx = \lim_{\delta \searrow 0} \int_{a}^{x_0-\delta} f(x)\, dx + \lim_{\delta \searrow 0} \int_{x_0+\delta}^{b} f(x)\, dx.$$

Aquí suponemos que los límites anteriores existen y son finitos. El objetivo es calcular la integral $\int_a^b f(x)\, dx$ con un error inferior a $\varepsilon > 0$. En el primer caso, se debe determinar un valor de $\delta > 0$ tal que

$$\Big|\int_a^{a+\delta} f(x)\, dx\Big| < \frac{\varepsilon}{2}$$

y seguidamente calcular una aproximación de $\int_{a+\delta}^{b} f(x)\,dx$ con un error inferior a $\frac{\varepsilon}{2}$. Análogamente se procede en el segundo caso. En el tercer caso se debe determinar un valor de $\delta > 0$ tal que

$$\Big|\int_{x_0-\delta}^{x_0} f(x)\,dx\Big| + \Big|\int_{x_0}^{x_0+\delta} f(x)\,dx\Big| < \frac{\varepsilon}{2}.$$

A continuación se deben calcular aproximaciones de las integrales $\int_{a}^{x_0-\delta} f(x)\,dx$ y $\int_{x_0+\delta}^{b} f(x)\,dx$ con errores inferior a $\frac{\varepsilon}{4}$ para cada una de ellas.

En ocasiones, un adecuado cambio de variable o una manipulación conveniente de la integral puede reducir la dificultad. Por ejemplo, consideremos la integral

$$\int_0^{\frac{\pi}{2}} \frac{\cos x}{\sqrt{x}}\,dx.$$

Es obvio que la función posee una singularidad en el cero. Pero, mediante una integración por partes podemos evitar la singularidad

$$u = \cos x, \quad dv = \frac{dx}{\sqrt{x}} \Rightarrow du = -\operatorname{sen} x\,dx, \quad v = 2\sqrt{x},$$

$$\int_0^{\frac{\pi}{2}} \frac{\cos x}{\sqrt{x}}\,dx = \Big[2\sqrt{x}\,\cos x\Big]_0^{\frac{\pi}{2}} + 2\int_0^{\frac{\pi}{2}} \sqrt{x}\operatorname{sen} x\,dx = 2\int_0^{\frac{\pi}{2}} \sqrt{x}\operatorname{sen} x\,dx.$$

Ahora podemos aproximar directamente la última integral.

4.5 Ejercicios

Calcula las integrales siguientes con un error inferior al indicado. Si no se precisa el valor de ε_r, entonces se tomará como cero.

1. $\displaystyle\int_0^{\pi} \exp\left(\operatorname{sen} x \cos x\right) dx$; $\varepsilon_a = 10^{-12}$.
2. $\displaystyle\int_0^{4} \mathrm{e}^{x^2}\, dx$; $\varepsilon_a = 10^{-12}$, $\varepsilon_r = 10^{-10}$
3. $\displaystyle\int_0^{\pi} \frac{\operatorname{sen} x}{x}\, dx$; $\varepsilon_a = 10^{-12}$.
4. $\displaystyle\int_0^{\infty} \mathrm{e}^{-x} \cos^2 x\, dx$; $\varepsilon_a = 10^{-12}$.
5. $\displaystyle\int_0^{\infty} \mathrm{e}^{-x^2} \log\left(2 + \operatorname{sen} x\right) dx$; $\varepsilon_a = 10^{-12}$.
6. $\displaystyle\int_0^{1} \frac{\cos x}{2\pi \operatorname{sen}\left(\sqrt{x}\right)}\, dx$; $\varepsilon_a = 10^{-12}$.
7. $\displaystyle\int_0^{\infty} \mathrm{e}^{-x^4}\, dx$; $\varepsilon_a = 10^{-12}$.
8. $\displaystyle\int_0^{\infty} \frac{\operatorname{sen} x}{1 + x^3}\, dx$; $\varepsilon_a = 10^{-12}$.
9. $\displaystyle\int_0^{\infty} \frac{\mathrm{e}^{-x}}{1 + x^4}\, dx$; $\varepsilon_a = 10^{-12}$.
10. $\displaystyle\int_0^{\infty} \frac{\mathrm{e}^{-x}}{\sqrt{x}}\, dx$; $\varepsilon_a = 10^{-12}$.
11. $\displaystyle\int_0^{\infty} \frac{\operatorname{sen} x}{x} \mathrm{e}^{-x^2}\, dx$; $\varepsilon_a = 10^{-12}$.

Capítulo 5

Integración Numérica de Ecuaciones Diferenciales Ordinarias

5.1 Introducción

En este capítulo vamos a estudiar la resolución numérica de dos problemas asociados a ecuaciones diferenciales ordinarias: el problema de Cauchy o problema de valor inicial y el problema de contorno. La formulación del problema de Cauchy admite la siguiente forma genérica:

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, t_f], \\ y(t_0) = y_0, \end{cases} \tag{5.1}$$

donde $-\infty < t_0 < t_f < \infty$, $f : [t_0, t_f] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ es una función y $y_0 \in \mathbb{R}^n$. Si $n = 1$, (5.1) es el problema de valor inicial asociado a una ecuación diferencial. Cuando $n > 1$, se trata de un sistema de ecuaciones diferenciales. En efecto, para $n > 1$ existen funciones $f_j : [t_0, t_f] \times \mathbb{R}^n \longrightarrow \mathbb{R}$, $1 \leq j \leq n$, tales que

$$f(t, y) = f(t, y_1, y_2, \ldots, y_n) = \begin{pmatrix} f_1(t, y_1, y_2, \ldots, y_n) \\ f_2(t, y_1, y_2, \ldots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \ldots, y_n) \end{pmatrix}.$$

Por lo tanto, (5.1) puede escribirse en la forma

$$\begin{cases} y_1'(t) = f_1(t, y_1(t), y_2(t), \ldots, y_n(t)), \\ y_2'(t) = f_2(t, y_1(t), y_2(t), \ldots, y_n(t)), \\ \qquad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t \in [t_0, t_f] \\ y_n'(t) = f_n(t, y_1(t), y_2(t), \ldots, y_n(t)) \\ y_1(t_0) = y_{01}, y_2(t_0) = y_{02}, \ldots, y_n(t_0) = y_{0n} \end{cases} \tag{5.2}$$

La ecuación diferencial (5.1) o (5.2) es una ecuación de primer orden. Se llama así porque solamente involucra la derivada primera de la función incógnita y. Consideremos ahora problemas de orden superior:

$$\begin{cases} y^{(m)}(t) = g(t, y(t), y'(t), y''(t), \ldots, y^{(m-1)}(t)), \quad t \in [t_0, t_f], \\ y(t_0) = y_{00}, y'(t_0) = y_{01}, y''(t_0) = y_{02}, \ldots, y^{(m-1)}(t_0) = y_{0m-1}. \end{cases} \tag{5.3}$$

Este es un problema de valor inicial asociado a una ecuación diferencial de orden m. Estos problemas pueden transformarse en otros equivalentes de la forma (5.1). En efecto, pongamos $y_1 = y$, $y_2 = y'$, $y_3 = y''$, ..., $y_m = y^{(m-1)}$. Entonces, la función $t \to (y_1(t), y_2(t), \ldots, y_m(t))^{\mathrm{T}}$ satisface el siguiente sistema de ecuaciones diferenciales de primer orden

$$\begin{cases} \begin{array}{lcll} y_1'(t) & = & y_2(t), & \\ y_2'(t) & = & y_3(t), & \\ & \vdots & & t \in [t_0, t_f], \\ y_{m-1}'(t) & = & y_m(t), & \\ y_m'(t) & = & g(t, y_1(t), y_2(t), \ldots, y_m(t)) & \end{array} \\ y_1(t_0) = y_{00}, y_2(t_0) = y_{01}, \ldots, y_m(t_0) = y_{0m-1}. \end{cases} \tag{5.4}$$

Este sistema puede reescribirse en la forma (5.1) tomando

$$f(t, y) = \begin{pmatrix} y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_m \\ g(t, y_1, y_2, \ldots, y_m) \end{pmatrix} \quad \text{y} \quad y_0 = \begin{pmatrix} y_{00} \\ y_{01} \\ y_{02} \\ \vdots \\ y_{0m-1} \end{pmatrix}.$$

Vemos que la formulación (5.1) es bastante potente, en el sentido de que muchos problemas pueden formularse de esta forma. Todavía podemos considerar sistemas de orden superior y transformarlos en la forma (5.1). Pongamos un ejemplo

$$\begin{cases} u''(t) + \dfrac{u(t)}{(u^2(t) + v^2(t))^{3/2}} = 0 \\ v''(t) + \dfrac{v(t)}{(u^2(t) + v^2(t))^{3/2}} = 0 \quad t \in [0, 20] \\ u(0) = 0.5, \ u'(0) = 0 \\ v(0) = 0, \ v'(0) = \sqrt{3}. \end{cases} \tag{5.5}$$

Tomando $t_0 = 0$, $t_f = 20$, $y_1 = u$, $y_2 = u'$, $y_3 = v$ e $y_4 = v'$, y definiendo

$$f(t, y) = \begin{pmatrix} y_2 \\ -\dfrac{y_1}{(y_1^2 + y_3^2)^{3/2}} \\ y_3 \\ -\dfrac{y_3}{(y_1^2 + y_3^2)^{3/2}} \end{pmatrix} \quad \text{y} \quad y_0 = \begin{pmatrix} 0.5 \\ 0 \\ 0 \\ \sqrt{3} \end{pmatrix},$$

entonces el sistema (5.5) se puede escribir en la forma (5.1).

En la última parte del capítulo estudiaremos la resolución de problemas de contorno. Un ejemplo de tales problemas es el siguiente:

$$\begin{cases} y''(t) = f(t, y(t), y'(t)) \ t \in [t_0, t_f], \\ y(t_0) = y_0, \ y(t_f) = y_f. \end{cases}$$

En este problema se busca la solución de una ecuación diferencial que satisfaga dos condiciones de contorno. En un problema de valor inicial las condiciones se establecen sobre el instante inicial. En un problema de contorno, las condiciones se establecen tanto en el instante inicial como en el final.

El plan del capítulo es el siguiente. En la próxima sección daremos algunos resultados sobre existencia y unicidad de solución del problema de valor inicial. En la sección 5.3, describiremos los métodos Runge-Kutta para la resolución del problema (5.1). En la sección 5.4 veremos los pares de métodos encajados, en particular los métodos Runge-Kutta-Fehlberg y Dormand y Prince, y la implementación de los métodos. Finalizamos el capítulo describiendo el método de tiro para la resolución de los problemas de contorno.

Para un mayor detalle sobre los métodos Runge-Kutta y su implementación el lector puede consultar los libros [3] y [4]. Otros libros clásicos sobre la resolución numérica de ecuaciones diferenciales ordinarias son [2, 10, 11].

5.2 Existencia y Unicidad de Solución del Problema de Cauchy

Respecto a la existencia y unicidad de solución del problema (5.1), nos vamos a limitar a dar unos resultados sencillos, sin mucha generalidad. Para más información el lector puede consultar el libro clásico de Hartman [7].

Si la función $f : [t_0, t_f] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ es continua, entonces una de las dos condiciones siguientes se cumple:

1. Existe un tiempo $0 < t^* \leq t_f$ y una función derivable $y : [t_0, t^*) \longrightarrow \mathbb{R}^n$ tal que su derivada es una función continua satisfaciendo

$$\begin{cases} y'(t) = f(t, y(t)) \ \ t \in [t_0, t^*), \\ y(t_0) = y_0, \\ \lim\limits_{t \to t^*} \|y(t)\| = +\infty. \end{cases} \tag{5.6}$$

2. Existe una función derivable $y : [t_0, t_f] \longrightarrow \mathbb{R}^n$ tal que su derivada es una función continua satisfaciendo (5.1).

En el primer caso se dice que y es una solución local. En el segundo caso diremos que y es una solución global o simplemente una solución de (5.1). Por lo tanto, es suficiente que f sea una función continua para que exista una solución local o global.

Veamos un par de ejemplos. Primero consideramos el problema

$$\begin{cases} y'(t) = y^2(t) \quad t \in [0,2], \\ y(0) = 1. \end{cases} \tag{5.7}$$

Obviamente la función $f : [0,2] \times \mathbb{R} \longrightarrow \mathbb{R}$ viene dada por $f(t,y) = y^2$ y es continua. Entonces, tomando $t^* = 1$ e $y : [0,1) \longrightarrow \mathbb{R}$ definida por $y(t) = \frac{1}{1-t}$, se observa que y es infinitamente derivable,

$$\begin{aligned} &y'(t) = \frac{1}{(1-t)^2} = y^2(t) \quad \text{para} \quad t \in [0,1), \\ &y(0) = 1, \ \text{y} \quad \lim_{t \to 1} y(t) = \infty. \end{aligned}$$

Por lo tanto y es una solución local del problema (5.7).

Consideremos ahora el problema

$$\begin{cases} y'(t) = ty^2(t) \quad t \in [0,t_f], \\ y(0) = -1, \end{cases} \tag{5.8}$$

donde t_f es un número positivo cualquiera, pudiendo ser arbitrariamente grande. En esta ocasión $f : [0,t_f] \times \mathbb{R} \longrightarrow \mathbb{R}$ viene dada por $f(t,y) = ty^2$. Si tomamos $y : [0,t_f] \longrightarrow \mathbb{R}$ definida por $y(t) = \frac{-2}{2+t^2}$, entonces tenemos

$$\begin{aligned} &y'(t) = \frac{4t}{(2+t^2)^2} = ty^2(t) \quad \text{para} \quad t \in [0,t_f], \\ &y(0) = -1. \end{aligned}$$

Por lo tanto y es una solución global de (5.8).

Una cuestión importante es la unicidad de solución. Simplemente diremos que si f es derivable respecto de y en cada punto $(t,y) \in [t_0,t_f] \times \mathbb{R}^n$ y la derivada parcial $\frac{\partial f}{\partial y} : [t_0,t_f] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ es una función continua, entonces o bien existe una única solución global o existe una única solución local. No pueden existir una solución local y otra global diferentes. Notemos que toda solución global también es local, siendo el recíproco falso. Como consecuencia de este resultado, tenemos que la solución local del problema (5.7) y la solución global del problema (5.8) son la únicas soluciones. Sin embargo, tomemos ahora la función $f : [0,t_f] \times \mathbb{R} \longrightarrow \mathbb{R}$, $t_f > 0$ arbitrario, definida por

$$f(t,y) = \begin{cases} \sqrt{y} & \text{si } y \geq 0, \\ 0 & \text{si } y < 0. \end{cases}$$

Entonces f es una función continua, pero no es derivable para $y = 0$. Si consideramos el problema

$$\begin{cases} y'(t) = f(t,y(t)) \quad t \in [0,t_f], \\ y(0) = 0, \end{cases} \tag{5.9}$$

se observa que $y(t) = 0$ para cada $t \in [0, t_f]$ es una solución de este problema. Pero tambien lo es la función $y(t) = \frac{1}{4}t^2$. De hecho existen infinitas soluciones. En efecto, para cada $0 < \rho < t_f$ es fácil comprobar que la función

$$y_\rho(t) = \begin{cases} 0 & \text{si } t \leq \rho, \\ \frac{1}{4}(t-\rho)^2 & \text{si } t > \rho \end{cases}$$

es también una solución de la ecuación.

Hay ocasiones en las que f posee una singularidad en un punto $(\hat{t}, \hat{y}) \in (t_0, t_f] \times (\mathbb{R}^n \setminus \{y_0\})$, pero es continua en un entorno del punto (t_0, y_0), entonces hay una solución local o incluso puede haber una solución global. Si además, f es derivable respecto de y en un entorno de (t_0, y_0) con derivada continua en dicho entorno, entonces la solución local es única. Por ejemplo, consideremos el problema de Cauchy

$$\begin{cases} y'(t) = -\dfrac{1}{y^2(t)} \quad t \in [0,1], \\ y(0) = 1. \end{cases} \tag{5.10}$$

Entonces $f : [0,1] \times \mathbb{R} \longrightarrow \mathbb{R}$ es continua y derivable respecto de y con derivada continua en $[t_0, t_f] \times (\mathbb{R} \setminus \{0\})$, que es un entorno de (t_0, y_0). En consecuencia, hay una única solución local, que es $y : [0, \frac{1}{3}) \longrightarrow \mathbb{R}$ con $y(t) = \sqrt[3]{1-3t}$. Observemos que

$$\lim_{t \to \frac{1}{3}} y(t) = y_0 = 0 \quad \text{y} \quad \lim_{t \to \frac{1}{3}} y'(t) = -\infty.$$

En este caso $(t, y(t))$ converge hacia la singularidad $(\hat{t}, \hat{y}) = (\frac{1}{3}, 0)$ de f cuando $t \to \frac{1}{3}$ y $y'(t)$ tiende a $-\infty$.

5.3 Métodos Runge-Kutta

La integración numérica del problema de Cauchy (5.1) consiste en determinar unos instantes $t_0 < t_1 < t_2 < \ldots < t_N = t_f$ y valores $\{y_0, y_1, y_2, \ldots, y_N\}$ de forma que $y(t_j) \approx y_j$ para $1 \leq j \leq N$. Es decir, se trata de aproximar los valores de la solución en puntos t_j con $0 \leq j \leq N$. Para ello partimos del conocimiento del valor de la solución en t_0: $y(t_0) = y_0$. Los algoritmos que vamos a ver se basan en el conocimiento aproximado de la solución $y(t)$ en un punto t_n para determinar la aproximación de $y(t)$ en el punto $t_{n+1} = t_n + h_n$, donde $h_n > 0$ se dice que es el paso del método en el instante t_n. Estos métodos se conocen como métodos de un paso porque se toma la aproximación $y(t_n) \approx y_n$ para determinar la aproximación $y(t_{n+1}) \approx y_{n+1}$. Los métodos multipaso, que no vamos a estudiar aquí, utilizan las aproximaciones $\{y_{n-k+1}, y_{n-k+2}, \ldots, y_n\}$ de $y(t)$ en los k instantes anteriores $\{t_{n-k+1}, t_{n-k+2}, \ldots, t_n\}$ para determinar la aproximación $y(t_{n+1}) \approx y_{n+1}$. Un método multipaso precisa de un método de

un paso para arrancar el algoritmo. Una forma de abordar la construción de un método de un paso se basa en la siguiente información:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} y'(t)\,dt = y(t_n) + \int_{t_n}^{t_n+h_n} f(t, y(t))\,dt. \tag{5.11}$$

A continuación utilizamos una fórmula de cuadratura del tipo (4.2) para aproximar la integral:

$$y(t_{n+1}) \approx y_n + h_n \sum_{j=1}^{m} b_j f(t_n + c_j h_n, y(t_n + c_j h_n)), \tag{5.12}$$

donde los números $\{c_j\}_{j=1}^m$ están todos en el intervalo $[0,1]$. Por lo tanto, los puntos $t_{n,j} = t_n + c_j h_n \in [t_n, t_n + h_n]$ son los nodos de la fórmula de cuadratura y los $\{b_j\}_{j=1}^m$ son los pesos. El caso más sencillo consiste en tomar $m = 1$, $c_1 = 0$ y $b_1 = 1$, entonces (5.12) se reduce a la expresión

$$y_{n+1} = y_n + h_n f(t_n, y_n), \quad n \geq 0. \tag{5.13}$$

Este método se conoce como **Método de Euler**. Una alternativa consiste en tomar $m = 1$, $c_1 = 1$ y $b_1 = 1$, entonces (5.12) se convierte en la siguiente fórmula

$$y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1}), \quad n \geq 0. \tag{5.14}$$

Éste es el **Método de Euler Implícito**. En efecto, para determinar y_{n+1} es necesario resolver una ecuación no lineal o un sistema de ecuaciones si (5.1) es un sistema de ecuaciones diferenciales. A tal efecto se suele utilizar el método de Newton. Aunque el método de Euler (5.13) es más sencillo de implementar que el método implícito (5.14), el método implícito posee un mejor comportamiento para cierto tipo de ecuaciones diferenciales y permite el uso de longitudes de paso h_n más grandes.

Salvo en los métodos de Euler explicados, la implementación de la fórmula (5.12) precisa obtener previamente aproximaciones de los valores $y(t_n + c_j h_n)$, las cuales pueden obtenerse mediante la aplicación de otras fórmulas de cuadratura. Veamos algunos ejemplos.

Método de Heun:

$$y_{n+1} = y_n + \frac{h_n}{2}[f(t_n, y_n) + f(t_n + h_n, y_n + h_n f(t_n, y_n))]. \tag{5.15}$$

Este método se corresponde con la fórmula de cuadratura del trapecio aplicada a la integral en (5.11):

$$y(t_{n+1}) \approx y_n + \frac{h_n}{2}[f(t_n, y(t_n)) + f(t_n + h_n, y(t_n + h_n))].$$

Utilizando la aproximación que ya tenemos $y_n \approx y(t_n)$ y la fórmula de Euler (5.13) para aproximar $y(t_n + h_n)$, se obtiene (5.15). En este ejemplo tenemos $m = 2$, $c_1 = 0$, $c_2 = 1$, $b_1 = b_2 = \frac{1}{2}$.

De cara a simplificar la expresión (5.15), así como otras expresiones más complicadas que veremos, podemos describir los métodos en la siguiente forma

$$\begin{aligned}
&t_{n,1} = t_n,\ t_{n,2} = t_n + h_n,\\
&y_{n,1} = y_n,\ K_{n,1} = f(t_{n,1}, y_{n,1}),\\
&y_{n,2} = y_n + h_n K_{n,1},\ K_{n,2} = f(t_{n,2}, y_{n,2}),\\
&y_{n+1} = y_n + \frac{h_n}{2}[K_{n,1} + K_{n,2}].
\end{aligned}$$

Es inmediato comprobar que el valor y_{n+1} obtenido mediante este esquema es el mismo que proporciona la fórmula (5.15).

Método de Heun de Tres Etapas:

$$\begin{aligned}
&t_{n,1} = t_n,\ t_{n,2} = t_n + \frac{1}{3}h_n,\ t_{n,3} = t_n + \frac{2}{3}h_n\\
&y_{n,1} = y_n,\ K_{n,1} = f(t_{n,1}, y_{n,1}),\\
&y_{n,2} = y_n + \frac{1}{3}h_n K_{n,1},\ K_{n,2} = f(t_{n,2}, y_{n,2}),\\
&y_{n,3} = y_n + \frac{2}{3}h_n K_{n,2},\ K_{n,3} = f(t_{n,3}, y_{n,3}),\\
&y_{n+1} = y_n + \frac{h_n}{4}[K_{n,1} + 3K_{n,3}].
\end{aligned}$$

En este método tenemos $m = 3$, $c_1 = 0$, $c_2 = \frac{1}{3}$, $c_3 = \frac{2}{3}$, $b_1 = \frac{1}{4}$, $b_2 = 0$ y $b_3 = \frac{3}{4}$.

Método de Kutta de Tres Etapas:

$$\begin{aligned}
&t_{n,1} = t_n,\ t_{n,2} = t_n + \frac{1}{2}h_n,\ t_{n,3} = t_n + h_n\\
&y_{n,1} = y_n,\ K_{n,1} = f(t_{n,1}, y_{n,1}),\\
&y_{n,2} = y_n + \frac{1}{2}h_n K_{n,1},\ K_{n,2} = f(t_{n,2}, y_{n,2}),\\
&y_{n,3} = y_n + h_n[-K_{n,1} + 2K_{n,2}],\ K_{n,3} = f(t_{n,3}, y_{n,3}),\\
&y_{n+1} = y_n + \frac{h_n}{6}[K_{n,1} + 4K_{n,2} + K_{n,3}].
\end{aligned}$$

En este método tenemos $m = 3$, $c_1 = 0$, $c_2 = \frac{1}{2}$, $c_3 = 1$, $b_1 = \frac{1}{6}$, $b_2 = \frac{4}{6}$ y $b_3 = \frac{1}{6}$. Este método utiliza como fórmula de cuadratura básica la fórmula de Simpson.

Método Runge-Kutta Clásico:

$$
\begin{aligned}
&t_{n,1} = t_n,\ t_{n,2} = t_n + \frac{1}{2}h_n,\ t_{n,3} = t_n + \frac{1}{2}h_n,\ t_{n,4} = t_n + h_n\\
&y_{n,1} = y_n,\ K_{n,1} = f(t_{n,1}, y_{n,1}),\\
&y_{n,2} = y_n + \frac{1}{2}h_n K_{n,1},\ K_{n,2} = f(t_{n,2}, y_{n,2}),\\
&y_{n,3} = y_n + \frac{1}{2}h_n K_{n,2},\ K_{n,3} = f(t_{n,3}, y_{n,3}),\\
&y_{n,4} = y_n + h_n K_{n,3},\ K_{n,4} = f(t_{n,4}, y_{n,4}),\\
&y_{n+1} = y_n + \frac{h_n}{6}[K_{n,1} + 2K_{n,2} + 2K_{n,3} + K_{n,4}].
\end{aligned}
$$

En este método tenemos $m = 4$, $c_1 = 0$, $c_2 = c_3 = \frac{1}{2}$, $c_4 = 1$, $b_1 = \frac{1}{6}$, $b_2 = b_3 = \frac{2}{6}$ y $b_4 = \frac{1}{6}$. Al igual que el método de Kutta de tres etapas, este método utiliza como fórmula de cuadratura básica la fórmula de Simpson, pero la evaluación en el punto medio se obtiene de dos formas diferentes que se promedian.

Todos los métodos descritos responden a la misma estructura y vienen determinados por los parámetros A, matriz $m \times m$, b y c, vectores de $\mathbb{R}^m$, con $c_j \in [0, 1]$ para cada $j = 1, \ldots, m$. En todos los casos presentados, excepto en el método de Euler impícito, $c_1 = 0$ y A es una matriz triangular inferior con ceros en la diagonal:

$$
A = \begin{pmatrix}
0 & 0 & 0 & \ldots & 0 & 0\\
a_{21} & 0 & 0 & \ldots & 0 & 0\\
a_{31} & a_{32} & 0 & \ddots & 0 & 0\\
\vdots & \vdots & \ddots & \ddots & \vdots & \vdots\\
a_{m-1,1} & a_{m-1,2} & \ldots & \ddots & 0 & 0\\
a_{m1} & a_{m2} & \ldots & \ldots & a_{mm-1} & 0
\end{pmatrix}
\quad b = \begin{pmatrix} b_1\\ b_2\\ \vdots\\ b_m \end{pmatrix}
\quad c = \begin{pmatrix} c_1\\ c_2\\ \vdots\\ c_m \end{pmatrix}
$$

Estos métodos se conocen como métodos de Runge-Kutta explícitos. El parámetro m es el número de etapas del método. En los métodos Runge-Kutta implícitos c_1 puede ser diferente de cero y la matriz A puede no tener ceros en la diagonal, de hecho puede ser una matriz plena. Los parámetros A, b y c se suelen disponer en un tablero, denominado tablero de Butcher, como sigue

$$
\begin{array}{c|c}
c & A\\
\hline
 & b^T
\end{array}
$$

El método Runge-Kutta explícito de m etapas tiene la siguiente forma

$$\begin{cases} t_{n,i} = t_n + c_i h_n, \quad 1 \leq i \leq m, \\ \begin{cases} y_{n,1} = y_n, \ K_{n,1} = f(t_{n,1}, y_{n,1}), \\ y_{n,i} = y_n + h_n \displaystyle\sum_{j=1}^{i-1} a_{ij} K_{n,j}, \ K_{n,i} = f(t_{n,i}, y_{n,i}), \quad 2 \leq i \leq m, \end{cases} \\ y_{n+1} = y_n + h_n \displaystyle\sum_{j=1}^{m} b_j K_{n,j}. \end{cases}$$

Incluso, podemos eliminar los términos $y_{n,i}$ y escribir el método como sigue

$$\begin{cases} t_{n,i} = t_n + c_i h_n, \quad 1 \leq i \leq m, \\ \begin{cases} K_{n,1} = f(t_n, y_n) \\ K_{n,i} = f(t_{n,i}, y_n + h_n \displaystyle\sum_{j=1}^{i-1} a_{ij} K_{n,j}), \quad 2 \leq i \leq m, \end{cases} \\ y_{n+1} = y_n + h_n \displaystyle\sum_{j=1}^{m} b_j K_{n,j}. \end{cases} \tag{5.16}$$

Volviendo a los métodos antes descritos, vamos a escribir su correspondiente tablero de Butcher.

Método de Euler:

0	0
	1

Método de Heun:

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$

Método de Heun de Tres Etapas:

0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0
$\frac{2}{3}$	0	$\frac{2}{3}$	0
	$\frac{1}{4}$	0	$\frac{3}{4}$

Método de Kutta de Tres Etapas:

0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
1	-1	2	0
	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$

Método Runge-Kutta Clásico:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

Terminamos esta sección analizando el error en los métodos Runge-Kutta. Para ello observemos que un método Runge-Kutta puede escribirse de forma abstracta como sigue

$$y_{n+1} = y_n + h_n \Phi_f(t_n, y_n, h_n),\ 0 \leq n \leq N-1. \tag{5.17}$$

Por ejemplo, en el método de Euler $\Phi_f(t, y, h) = f(t, y)$. En el método de Heun de dos etapas

$$\Phi_f(t, y, h) = \frac{1}{2}[f(t, y) + f(t + h, y + hf(t, y))].$$

En la aplicación de un método Runge-Kutta (5.17) debemos distinguir dos tipos de errores, que definimos a continuación.

DEFINICIÓN 5.1 *Se llama error local del método* (5.17) *en el paso* n *a la cantidad*

$$\varepsilon_n = y(t_n + h_n) - [y(t_n) + h_n \Phi_f(t_n, y(t_n), h_n)]. \tag{5.18}$$

Se llama error global a la cantidad

$$e_n = y(t_n + h_n) - [y_n + h_n \Phi_f(t_n, y_n, h_n)]. \tag{5.19}$$

Diremos que el método es de orden p *si existe una constante* $C > 0$ *tal que*

$$\|\varepsilon_n\| \leq C h_n^{p+1}. \tag{5.20}$$

Notemos que el error local es aquel que se comete al dar una paso partiendo del conocimiento del valor exacto de la solución en el instante t_n. Al pasar de t_0 a t_1, el error local y global coinciden porque se conoce el valor exacto de $y_0 = y(t_0)$. Sin embargo, al pasar de t_1 a t_2, el error global sufre del desconocimiento de $y(t_1)$, solamente se conoce una aproximación $y_1 \approx y(t_1)$. Por lo tanto, el error global es una acumulación de los errores de cada paso debido al error local del método. El error local solamente mide el error cometido suponiendo que partimos del conocimiento exacto de la solución en el punto t_n en el que estamos.

Si la función f es derivable respecto de y con derivada continua, entonces existe una constante $\Lambda > 0$ tal que

$$\|\Phi_f(t, y, h) - \Phi_f(t, z, h)\| \leq \Lambda \|y - z\| \tag{5.21}$$

para cada $h \leq h_{max}$ e y, z en un entorno de la solución. Se puede demostrar que si el método es de orden p, es decir (5.20) se cumple, y (5.21) se satisface, entonces el error global es estimado por la siguiente cantidad

$$\|e_n\| \leq \frac{C}{\Lambda}[\exp(t_n - t_0) - 1]h^p \tag{5.22}$$

donde $h = \max_n h_n$. El lector puede consultar [5, Cap. 3-§3].

Puesto que h_n es pequeño, a mayor orden se espera, en principio, más precisión en la solución. Obviamente el orden del método depende del número de etapas m y la elección de los parámetros A, b y c. Butcher [2, Cap. 3] desarrolló un formulismo que ha permitido obtener el orden del método en términos de los coeficientes del mismo. A todos los métodos se les exige que $Ae = c$, donde e es el vector de m componentes todas iguales a 1. Adicionalmente, las siguientes ecuaciones deben cumplirse para que el método alcance el orden indicado:

Orden 1: $b^T e = 1$.

Orden 2: La anterior más $b^T c = 1/2$.

Orden 3: Las anteriores más $b^T c^2 = 1/3$; $b^T Ac = 1/6$.

Orden 4: Las anteriores más $b^T c^3 = 1/4$; $b^T Ac^2 = 1/12$; $b^T A^2 c = 1/24$;
$(b \cdot c)^T Ac = 1/8$;

Orden 5: Las anteriores más $b^T c^4 = 1/5$; $(b \cdot c^2)^T Ac = 1/10$; $(b \cdot c)^T Ac^2 = 1/15$;
$(b \cdot c)^T A^2 c = 1/30$; $b^T (Ac)^2 = 1/20$; $b^T Ac^3 = 1/20$;
$b^T A(c \cdot Ac) = 1/40$; $b^T A^2 c^2 = 1/60$; $b^T A^3 c = 1/120$;

donde $x \cdot y$ denota el producto componente a componente de ambos vectores y x^k es el vector cuyas componentes son la potencia k de las componentes de x

$$x \cdot y = \begin{pmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_n y_n \end{pmatrix} \qquad x^k = \begin{pmatrix} x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{pmatrix}$$

En la siguiente tabla $m(p)$ designa el número mínimo de etapas necesarias para alcanzar orden p en un método explícito, y $n(p)$ es el número de ecuaciones que deben satisfacer los coeficientes del método:

p	$m(p)$	$n(p)$
1	1	1
2	2	2
3	3	4
4	4	8
5	6	17
6	7	37
7	9	85
8	11	200
9	$12 \leq m \leq 17$	486
10	$13 \leq m \leq 17$	1205

Esta tabla muestra la complicación de derivar métodos Runge-Kutta de orden elevado, así como el gran trabajo desarrollado por Butcher. El lector puede comprobar que el método de Euler es de orden 1; los de Heun de dos y tres etapas poseen orden 2 y 3, respectivamente; el método de Kutta de tres etapas es de orden 3; y el método Runge-Kutta clásico es de orden 4.

5.4 Métodos Runge-Kutta Encajados.

Cuando estudiamos la integración numérica en el Capítulo 4, vimos un procedimiento para estimar el error en la integración. Ese procedimiento nos permitió determinar cuántos intervalos se debían usar para implementar una fórmula compuesta conduciendo a un error inferior a una tolerancia prefijada. No todos los intervalos eran de la misma longitud, algunas partes del intervalo de integración podían requerir un mayor refinamiento. La misma cuestión nos planteamos aquí. Deseamos integrar la ecuación diferencial, es decir determinar unos instantes $t_0 < t_1 < \ldots < t_N = t_f$ y aproximaciones de la solución en esos instantes $\{y_0, y_1, \ldots, y_N\}$ de forma que el error sea menor que una tolerancia: $\|y(t_n) - y_n\| < TOL$ para cada $n = 0, 1, \ldots, N$. Para lograr esto debemos elegir los pasos h_n adecuadamente. Para decidir sobre la validez de un paso es necesario disponer de un mecanismo que nos proporcione una estimación del error. El procedimiento habitual entre los métodos Runge-Kutta consiste en utilizar dos métodos distintos durante la integración:

$$\begin{cases} y_{n+1} = y_n + h_n \Phi_f(t_n, y_n, h_n), \\ \hat{y}_{n+1} = y_n + h_n \Psi_f(t_n, y_n, h_n), \end{cases} \tag{5.23}$$

el primero de orden p y m etapas y el segundo de orden $q > p$ y $\hat{m}$ etapas. Siendo el segundo método más preciso que el primero, la diferencia $\varepsilon_n = \|\hat{y}_{n+1} - y_{n+1}\|$ proporciona una estimación del error local cometido por el primer método. No obstante, al igual que en los métodos adaptativos estudiados en el Capítulo 4, como $\hat{y}_{n+1}$ es mejor aproximación de la solución que y_{n+1}, entonces nos

quedaremos con $\hat{y}_{n+1}$ y la cantidad $\varepsilon_n = \|\hat{y}_{n+1} - y_{n+1}\|$ es una cota superior del error de integración.

La tolerancia deseada se define en función de dos parámetros: ε_a y ε_r, asociados a los errores absolutos y relativos. Fijados ε_a y ε_r, en el paso n se fija la tolerancia como $TOL = \varepsilon_a + \varepsilon_r\|\hat{y}_{n+1}\|$. Si $\varepsilon_n < TOL$, entonces se acepta el paso h_n. En caso contrario, se debe reducir el paso y volver a calcular $\hat{y}_{n+1}$ y ε_n con el nuevo paso. El siguiente algoritmo proporciona una estrategia para la determinación del paso.

CONTROL DEL PASO

1. Elección de h_0:

   $$h_0 = \max\{[\varepsilon_a + \varepsilon_r\|y_0\|]^{\frac{1}{p+1}}, h_{min}\}$$

   donde podemos fijar $h_{min} = (t_f - t_0)/10^6$ por ejemplo.

2. Criterio de aceptación del paso:

   Si $\|\varepsilon_n\| = \|y_{n+1} - \hat{y}_{n+1}\| \leq TOL = \varepsilon_a + \varepsilon_r\|\hat{y}_{n+1}\| \Rightarrow$ se acepta el paso.

3. Reducción del paso tras un fallo: dado

   $$r_n = 0.9\left(\frac{TOL}{\|\varepsilon_n\|}\right)^{\frac{1}{p+1}} \tag{5.24}$$

   se elige

   $$h_n \to \min\{r_n, r_{min}\}h_n,$$

   tomando $r_{min} \in [0.1, 0.5]$. (Nosotros tomaremos $r_{min} = 0.5$)

4. Previsión del paso siguiente: Si ha habido un fallo del paso en la determinación de una de las dos últimas aproximaciones de la solución, entonces se toma $r_{max} = 1$, en caso contrario se fija $r_{max} = 5$. Se calcula r_n como antes y se toma

   $$h_{n+1} = \begin{cases} h_n & \text{si } \ r_n \in [1, 1.1] \\ \min\{r_n, r_{max}\}h_n & \text{en otro caso.} \end{cases}$$

Observemos que si $\varepsilon_n > TOL$, entonces debemos reducir h_n. Para ello se calcula un factor de reducción $r_n < 1$. De acuerdo a la fórmula (5.20), el error local es de la forma Ch_n^{p+1}, lo que es aproximado por cantidad ε_n calculada. La elección del nuevo paso $r_n h_n$ debe ser tal que el error sea menor que la tolerancia, pero próximo a la tolerancia. En efecto, no debemos dar más pasos de los necesarios para integrar la ecuación diferencial. Si tomamos h_n demasiado pequeño, estamos haciendo el error más pequeño de lo requerido y estamos aumentando el tiempo de resolución de la ecuación diferencial. Si multiplicamos h_n por r_n tenemos que

$$C(r_n h_n)^{p+1} = r_n^{p+1} C h_n^{p+1} \approx r_n^{p+1}\varepsilon_n \lesssim TOL.$$

De aquí se sigue

$$r_n \lesssim \Big(\frac{TOL}{\varepsilon_n}\Big)^{\frac{1}{p+1}}.$$

Por lo tanto, la elección (5.24) es razonable. En todo caso, como medida de seguridad en orden a evitar computaciones innecesarias, el paso se reducirá al menos a la mitad en caso de un fallo.

En el caso de que el paso proporcione un error inferior a la tolerancia, tenemos que analizar la posibilidad de incrementar este paso para acelerar la integración. A tal fin se calcula también el factor r_n de acuerdo a (5.24). Observemos que en este caso $\frac{TOL}{\varepsilon_n} > 1$. Si $r_n > 1.1$, entonces incrementaremos el paso, en caso contrario no lo incrementamos para evitar riesgos de fallo. Tampoco se incrementa el paso si ha habido un fallo del paso en uno de los dos pasos anteriores. Por último, observemos que el factor de incremento del paso nunca será mayor que $r_{max} = 5$, esto también se hace para evitar fallos.

Terminamos esta sección estudiando la forma de elegir los métodos Runge-Kutta Φ_f y Ψ_f que permiten la integración de la ecuación diferencial con un control del error y permitiendo un paso de integración variable. En la búsqueda de un método eficiente para estimar el error se han ideado métodos Φ_f y Ψ_f de forma que uno va encajado en el otro. Por ejemplo, en los años 1968 y 1969 Fehlberg construyó pares de métodos encajados de orden 4 y 5. Veamos un primer ejemplo:

0						
$\frac{2}{9}$	$\frac{2}{9}$					
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$				
$\frac{3}{4}$	$\frac{69}{128}$	$-\frac{243}{128}$	$\frac{135}{64}$			
1	$-\frac{17}{12}$	$\frac{27}{4}$	$-\frac{27}{5}$	$\frac{16}{15}$		
	$\frac{1}{9}$	0	$\frac{9}{20}$	$\frac{16}{45}$	$\frac{1}{12}$	

Método de Orden 4

0						
$\frac{2}{9}$	$\frac{2}{9}$					
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$				
$\frac{3}{4}$	$\frac{69}{128}$	$-\frac{243}{128}$	$\frac{135}{64}$			
1	$-\frac{17}{12}$	$\frac{27}{4}$	$-\frac{27}{5}$	$\frac{16}{15}$		
$\frac{5}{6}$	$\frac{65}{432}$	$-\frac{5}{16}$	$\frac{13}{16}$	$\frac{4}{27}$	$\frac{5}{144}$	
	$\frac{47}{450}$	0	$\frac{12}{25}$	$\frac{32}{225}$	$\frac{1}{30}$	$\frac{6}{25}$

Método de Orden 5

Se observa que los parámetros A y c del método de orden 4 son coincidentes con las 5 primeras filas de los parámetros $\hat{A}$ y $\hat{c}$ del método de orden 5. Por lo tanto, los valores $K_{n,i}$ para $1 \leq i \leq 5$ son coincidentes para ambos métodos. Esto simplifica notablemente el número de cálculos, particularmente el número de evaluaciones de la función f. Entonces, para pasar de t_n a t_{n+1} aplicamos el método de orden 5 para obtener $\hat{y}_{n+1}$. Para hallar una cota del error local que se comete con esta aproximación de $y(t_{n+1})$ se aplica el método de orden 4 para obtener y_{n+1}. Entonces la cota del error local es estimada por $\varepsilon_n =$

$\|y_{n+1} - \hat{y}_{n+1}\|$. Observemos que

$$y_{n+1} = y_n + h_n\big(\frac{1}{9}K_{n,1} + \frac{9}{20}K_{n,3} + \frac{16}{45}K_{n,4} + \frac{1}{12}K_{n,5}\big),$$
$$\hat{y}_{n+1} = y_n + h_n\big(\frac{47}{450}K_{n,1} + \frac{12}{25}K_{n,3} + \frac{32}{225}K_{n,4} + \frac{1}{30}K_{n,5} + \frac{6}{25}K_{n,6}\big).$$

Entonces, tenemos

$$\begin{aligned} y_{n+1} - \hat{y}_{n+1} &= h_n\Big[(\frac{1}{9} - \frac{47}{450})K_{n,1} + (\frac{9}{20} - \frac{12}{25})K_{n,3} + (\frac{16}{45} - \frac{32}{225})K_{n,4} \\ &\quad + (\frac{1}{12} - \frac{1}{30})K_{n,5} - \frac{6}{25}K_{n,6}\Big] \\ &= h_n\Big[\frac{1}{150}K_{n,1} - \frac{3}{100}K_{n,3} + \frac{16}{75}K_{n,4} + \frac{1}{20}K_{n,5} - \frac{6}{25}K_{n,6}\Big]. \end{aligned}$$

Al vector de coeficientes

$$\begin{pmatrix} \frac{1}{150} \\ 0 \\ -\frac{3}{100} \\ \frac{16}{75} \\ \frac{1}{20} \\ -\frac{6}{25} \end{pmatrix}$$

se le conoce como el vector de los estimadores. En consecuencia, para implementar este método Runge-Kutta-Fehlberg es suficiente conocer los parámetros del método de orden 5 y los estimadores. Todo ello se dispone en un tablero:

0						
$\frac{2}{9}$	$\frac{2}{9}$					
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$				
$\frac{3}{4}$	$\frac{69}{128}$	$-\frac{243}{128}$	$\frac{135}{64}$			
1	$-\frac{17}{12}$	$\frac{27}{4}$	$-\frac{27}{5}$	$\frac{16}{15}$		
$\frac{5}{6}$	$\frac{65}{432}$	$-\frac{5}{16}$	$\frac{13}{16}$	$\frac{4}{27}$	$\frac{5}{144}$	
Orden 5	$\frac{47}{450}$	0	$\frac{12}{25}$	$\frac{32}{225}$	$\frac{1}{30}$	$\frac{6}{25}$
Estimadores	$\frac{1}{150}$	0	$-\frac{3}{100}$	$\frac{16}{75}$	$\frac{1}{20}$	$-\frac{6}{25}$

Runge-Kutta-Felhberg 4-5 #1

El siguiente tablero describe otro método Runge-Kutta-Fehlberg donde también

se encajan un método de orden 4 y otro de orden 5

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
Orden 5	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$
Estimadores	$-\frac{1}{360}$	0	$\frac{128}{4275}$	$\frac{2197}{75240}$	$-\frac{1}{50}$	$-\frac{2}{55}$

Runge-Kutta-Felhberg 4-5 #2

En los dos casos anteriores, el método de orden 4 es de 5 etapas, mientras que el de orden 5 es de 6 etapas. Fehlberg también obtuvo un par de métodos encajados de órdenes 7 y 8 de 11 y 13 etapas, respectivamente:

0													
$\frac{2}{27}$	$\frac{2}{27}$												
$\frac{1}{9}$	$\frac{1}{36}$	$\frac{1}{12}$											
$\frac{1}{6}$	$\frac{1}{24}$	0	$\frac{1}{8}$										
$\frac{5}{12}$	$\frac{5}{12}$	0	$-\frac{25}{16}$	$\frac{25}{16}$									
$\frac{1}{2}$	$\frac{1}{20}$	0	0	$\frac{1}{4}$	$\frac{1}{5}$								
$\frac{5}{6}$	$-\frac{25}{108}$	0	0	$\frac{125}{108}$	$-\frac{65}{27}$	$\frac{125}{54}$							
$\frac{1}{6}$	$\frac{31}{300}$	0	0	0	$\frac{61}{225}$	$-\frac{2}{9}$	$\frac{13}{900}$						
$\frac{2}{3}$	2	0	0	$-\frac{53}{6}$	$\frac{704}{45}$	$-\frac{107}{9}$	$\frac{67}{90}$	3					
$\frac{1}{3}$	$-\frac{91}{108}$	0	0	$\frac{23}{108}$	$-\frac{976}{135}$	$\frac{311}{54}$	$-\frac{19}{60}$	$\frac{17}{6}$	$-\frac{1}{12}$				
1	$\frac{2383}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{301}{82}$	$\frac{2133}{4100}$	$\frac{45}{82}$	$\frac{45}{164}$	$\frac{18}{41}$			
0	$\frac{3}{205}$	0	0	0	0	$-\frac{6}{41}$	$-\frac{3}{205}$	$-\frac{3}{41}$	$\frac{3}{41}$	$\frac{6}{41}$	0		
1	$-\frac{1777}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{289}{82}$	$\frac{2193}{4100}$	$\frac{51}{82}$	$\frac{33}{164}$	$\frac{12}{41}$	0	1	
Orden 8	0	0	0	0	0	$\frac{34}{104}$	$\frac{9}{35}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{9}{280}$	0	$\frac{41}{840}$	$\frac{41}{840}$
Estimadores	$\frac{41}{840}$	0	0	0	0	0	0	0	0	0	$\frac{41}{840}$	$-\frac{41}{840}$	$-\frac{41}{840}$

Runge-Kutta-Felhberg 7-8

De forma general, si (5.23) son dos pares encajados de órdenes p y q, $q > p$, de m y $\hat{m}$ etapas, de acuerdo a (5.16) y utilizando los estimadores est, un paso

del algoritmo definido por (5.23) se escribe en la forma

$$\begin{cases} t_{n,i} = t_n + c_i h_n, \quad 1 \le i \le \hat{m}, \\ \begin{cases} K_{n,1} = f(t_n, y_n) \\ K_{n,i} = f(t_{n,i}, y_n + h_n \sum_{j=1}^{i-1} a_{ij} K_{n,j}), \quad 2 \le i \le \hat{m}, \end{cases} \\ \hat{y}_{n+1} = y_n + h_n \sum_{j=1}^{\hat{m}} b_j K_{n,j}, \\ \varepsilon_n = h_n \Big\| \sum_{j=1}^{\hat{m}} est_j K_{n,j} \Big\|. \end{cases}$$

La siguiente función MATLAB, llamada rkf.m, implementa este algoritmo. Para ello se deben suministrar la siguientes variables de entrada:

- f: nombre de la función definiendo la ecuación diferencial
- tn: el instante t_n del que se parte
- hn: el paso h_n que vamos a dar
- yn: la aproximación de la solución conocida en el instante t_n: $y_n \approx y(t_n)$
- a,b,c,est: los parámetros del algoritmo A, b y c y los estimadores.

Como resultado se obtienen las variables de salida:

- yn1: la aproximación de $y(t_{n+1})$ con $t_{n+1} = t_n + h_n$
- en: una estimación del error $\varepsilon_n = \|y(t_{n+1}) - y_{n+1}\|$

```
function [yn1,en]=rkf(f,tn,hn,yn,a,b,c,est)
%
% Copyright: Eduardo Casas (Octubre, 2000)
%
m=length(b);
kn=feval(f,tn,yn);
for i=2:m
kn=[kn,feval(f,tn+hn*c(i),yn+hn*(kn*a(i,1:i-1)'))];
end
yn1=yn+hn*(kn*b);
en=hn*norm(kn*est);
```

Antes de aceptar el valor obtenido $yn1$, se debe comprobar si el error es menor que la Tolerancia $= \varepsilon_a + \varepsilon_r \|yn1\|$. Si es menor se acepta el paso y si es mayor se rechaza. En cualquier paso se debe proceder como se indicó en la estrategia de control del paso.

Como señalamos en la Sección 5.2, puede ocurrir que el problema de valor inicial (5.1) solamente tenga solución local. En este caso, se observa durante el proceso de resolución que $\|y_n\|$ (o $\|f(t_n, y_n)\|$) crece fuertemente conforme se progresa en la integración, lo que conduce a un paso h_n cada vez más pequeño, obligando al algoritmo a detenerse sin llegar al instante final t_f. En estas situaciones se observa la ausencia de una solución global, hallándose un solución local sin poder llegar al instante t^* en que se produce la explosión.

El enfoque de Fehlberg para deducir sus métodos consistió en determinar un buen método de orden p, en el sentido de que la constante de error fuera pequeña y tuviera buenas propiedades de estabilidad, y a continuación hallar un método de orden p+1 añadiendo las etapas necesarias al método de orden p. Una filosofía distinta fue la que usaron Dormand y Prince en los años 80 para proponer sus métodos Runge-Kutta encajados. Dado que en la integración de la ecuación diferencial tomamos como aproximación de $y(t_{n+1})$ el valor proporcionado por el método de orden más alto, el esfuerzo de Dormand y Prince consistió en desarrollar buenos métodos de orden $p + 1$ y a continuación buscar un método de orden p para la estimación del error. Veamos un par de métodos propuestos por estos investigadores.

En ambos tableros se observa que el método de orden 5 es de 6 etapas y el de orden 4 de 7. Entonces, los métodos de Dormand y Prince anteriores parecen menos eficientes que los Runge-Kutta-Fehlberg de órdenes 4 y 5 porque en éstos el número de etapas es 5 y 6, respectivamente. Así, los métodos RKF parecen requerir una evaluación menos en cada paso. Sin embargo, si observamos que la última fila de la matriz A de los tableros anteriores, vemos que coincide con el vector $\hat{b}$ del método de orden 5. Esto significa que $y_{n,7} = \hat{y}_{n+1}$ y, en consecuencia, $K_{n,7} = f(t_{n+1}, \hat{y}_{n+1})$. Por lo tanto, la evaluación adicional que es necesaria para estimar el error, se aprovecha en el siguiente paso. Solamente, se produce una evaluación de f adicional si hay un rechazo del paso o en el último paso, lo que no es significativo. Esto hace que la función rkf.m deba ser modificada para no repetir evaluaciones. La siguiente función, llamada dopri.m, puede utilizarse para integrar la ecuación diferencial con un método de Dormand y Prince:

```
function [yn1,en,kn1]=dopri(f,tn,hn,yn,kn,a,b,c,est)
%
% Copyright: Eduardo Casas (Octubre, 2000)
%
m=length(b);
for j=2:m
kn=[kn,feval(f,tn+hn*c(j),yn+hn*(kn*a(j,1:j-1)'))];
end
yn1=yn+hn*(kn*b);
kn1=feval(f,tn+hn,yn1);
kn=[kn,kn1];
en=hn*norm(kn*est);
```

Comparada esta función con rkf.m, se observa una entrada y una salida

adicionales: kn y kn1, respectivamente. El valor kn=$f(t_n, y_n)$ es proporcionado a dopri.m y esta función devuelve kn1= $f(t_{n+1}, y_{n+1})$, lo que será pasado a dopri.m en el siguiente paso.

0							
$\frac{2}{9}$	$\frac{2}{9}$						
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$					
$\frac{5}{9}$	$\frac{55}{324}$	$-\frac{25}{108}$	$\frac{50}{81}$				
$\frac{2}{3}$	$\frac{83}{330}$	$-\frac{13}{22}$	$\frac{61}{66}$	$\frac{9}{110}$			
1	$-\frac{19}{28}$	$\frac{9}{4}$	$\frac{1}{7}$	$-\frac{27}{7}$	$\frac{22}{7}$		
1	$\frac{19}{200}$	0	$\frac{3}{5}$	$-\frac{243}{400}$	$\frac{33}{40}$	$\frac{7}{80}$	
Orden 5	$\frac{19}{200}$	0	$\frac{3}{5}$	$-\frac{243}{400}$	$\frac{33}{40}$	$\frac{7}{80}$	
Orden 4	$\frac{431}{5000}$	0	$\frac{333}{500}$	$-\frac{7857}{10000}$	$\frac{957}{1000}$	$\frac{193}{2000}$	$-\frac{1}{50}$
Estimadores	$-\frac{11}{1250}$	0	$\frac{33}{500}$	$-\frac{891}{5000}$	$\frac{33}{250}$	$\frac{9}{1000}$	$-\frac{1}{50}$

Dormand y Prince 4–5 # 1

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
Orden 5	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
Orden 4	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$
Estimadores	$-\frac{71}{57600}$	0	$\frac{71}{16695}$	$-\frac{71}{1920}$	$\frac{17253}{339200}$	$-\frac{22}{525}$	$\frac{1}{40}$

Dormand y Prince 4–5 # 2

Observemos que la función dopri.m solamente requiere los parámetros a, b y c del algoritmo de orden 5 y los estimadores. Por lo tanto, los tableros anteriores se pueden reducir a los siguientes

0							
$\frac{2}{9}$	$\frac{2}{9}$						
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$					
$\frac{5}{9}$	$\frac{55}{324}$	$-\frac{25}{108}$	$\frac{50}{81}$				
$\frac{2}{3}$	$\frac{83}{330}$	$-\frac{13}{22}$	$\frac{61}{66}$	$\frac{9}{110}$			
1	$-\frac{19}{28}$	$\frac{9}{4}$	$\frac{1}{7}$	$-\frac{27}{7}$	$\frac{22}{7}$		
Orden 5	$\frac{19}{200}$	0	$\frac{3}{5}$	$-\frac{243}{400}$	$\frac{33}{40}$	$\frac{7}{80}$	
Estimadores	$-\frac{11}{1250}$	0	$\frac{33}{500}$	$-\frac{891}{5000}$	$\frac{33}{250}$	$\frac{9}{1000}$	$-\frac{1}{50}$

Dormand y Prince 4–5 # 1

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
Orden 5	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
Estimadores	$-\frac{71}{57600}$	0	$\frac{71}{16695}$	$-\frac{71}{1920}$	$\frac{17253}{339200}$	$-\frac{22}{525}$	$\frac{1}{40}$

Dormand y Prince 4–5 # 2

5.5 Problemas de Contorno. Método de Tiro.

En esta sección vamos a estudiar la resolución de problemas del tipo

$$\begin{cases} y''(t) = f(t, y(t), y'(t)) \; t \in [t_0, t_f], \\ y(t_0) = y_0, \; y(t_f) = y_f. \end{cases} \tag{5.25}$$

No podemos aplicar un método Runge-Kutta para integrar esta ecuación porque nos falta el valor de $y'(t_0)$. Vamos a llamar x a este valor, que es una incógnita. Procediendo como en (5.3) y (5.4), el problema de contorno anterior se puede transformar en un sistema colocando $y_1 = y$ e $y_2 = y'$, así tenemos

$$\begin{cases} y_1'(t) = y_2(t), \\ y_2'(t) = f(t, y_1(t), y_2(t)), \\ y_1(t_0) = y_0, \; y_2(t_0) = x. \end{cases} \quad t \in [t_0, t_f], \tag{5.26}$$

Para cada valor de $x \in \mathbb{R}$ tenemos una solución de (5.26), que llamaremos $y_x = (y_{x,1}, y_{x,2})^{\mathrm{T}}$. Tenemos que buscar el valor de x para el cual obtengamos $y_{x,1}(t_f) = y_f$. O lo que es lo mismo, si definimos la función $g : \mathbb{R} \longrightarrow \mathbb{R}$ por $g(x) = y_{x,1}(t_f) - y_f$, debemos resolver la ecuación $g(x) = 0$. Si encontramos

una solución $\bar{x}$ de dicha ecuación, entonces colocando este valor en (5.26) y aplicando un método Runge-Kutta hallamos la solución de (5.25). Para resolver la ecuación $g(x) = 0$ aplicaremos el método de la secante combinado con bisección tal y como lo estudiamos en el Capítulo 2. Para ello debemos buscar en primer lugar un intervalo $[a, b]$ de forma que el signo de $g(a)$ sea diferente del signo de $g(b)$. En este proceso es fundamental programar correctamente la función g. Para ello se deben seguir los siguientes pasos:

1. En primer lugar debemos programa la función f.m que define la ecuación diferencial (5.26).

2. Utilizaremos una función llamada oderkf.m para resolver el problema de valor inicial (5.26) para un valor de x dado. Esta función tiene la siguiente sintaxis:

   [t,y]=oderkf(f,t0,y0,tf,ea,er)

   donde

   • Tomaremos ea= 10^{-14} y er= 10^{-12}.
   • f es la función previamente definida, t0= t_0, y0= $(y_0, x)^{\mathrm{T}}$ y tf= t_f.
   • t es el vector fila de instantes t_n en los que se han calculado aproximaciones de la solución de (5.26)
   • y son dichas aproximaciones. El array y posee dos filas asociadas con $y_{x,1}$ e $y_{x,2}$, respectivamente y tantas columnas como t.

3. Finalmente se define la función g

```
function [gx,tx,yx]=g(x)
t0=...;
y0=[y0;x];
tf=...
ea=10^-14;
er=10^-12;
[tx,yx]=oderkf(f,t0,y0,tf,ea,er);
gx=yx(1,end)-yf;
end
```

Observemos que la función g, además de calcular el valor $g(x)$, proporciona la solución numérica de (5.26) para el valor de x. Así, cuando hayamos alcanzado un valor $\bar{x}$ tal que $g(\bar{x}) = 0$ (o muy pequeño), tendremos al mismos tiempo la solución de (5.26) correspondiente a $\bar{x}$ y por lo tanto también la del problema (5.25). El método así descrito se conoce como *Método de Tiro*. En los ejercicios al final de este capítulo se estudiarán otros problemas de contorno ligeramente distintos, pero la solución sigue los mismos pasos con pequeñas diferencias en la definición de g.

5.6 Ejercicios

1. Resuelve los siguientes problemas de Cauchy con $\varepsilon_a = 10^{-12}$ y $\varepsilon_r = 10^{-10}$.

1.1 $$\begin{cases} y'(t) = 10(1-t)y - 20y^2 \quad t \in [0,5] \\ y(0) = 1 \end{cases}$$

1.2 $$\begin{cases} y''(t) + y(t)^3 = 1 \quad t \in [0, 2\pi] \\ y(0) = 1,\ y'(0) = 1 \end{cases}$$

1.3 $$\begin{cases} y_1'(t) = 1 + y_1(t)^2 y_2(t) - 4y_1(t) \quad t \in [0,20] \\ y_2'(t) = 3y_1(t) - y_1(t)^2 y_2(t) \\ y_1(0) = \frac{3}{2},\ y_2(0) = 3 \end{cases}$$

1.4 $$\begin{cases} y''(t) + y(t) + 0.1\left(y(t)^2 - 1\right) y'(t) = 0\,,\ t \in [0, 2\pi] \\ y(0) = 0.1\,,\ y'(0) = 0.1 \end{cases}$$

1.5 $$\begin{cases} y'''(t) - 4y'(t) = t + 3\cos t + \mathrm{e}^{-2t}\,,\ t \in [0,2] \\ y(0) = 2\,,\ y'(0) = -\frac{23}{40},\ y''(0) = \frac{29}{4} \end{cases}$$

1.6 $$\begin{cases} y''(t) + \dfrac{y(t)}{(y^2(t) + z^2(t))^{3/2}} = 0 \quad t \in [0,20] \\ z''(t) + \dfrac{z(t)}{(y^2(t) + z^2(t))^{3/2}} = 0 \\ y(0) = 0.5,\ y'(0) = 0 \\ z(0) = 0,\ z'(0) = \sqrt{3} \end{cases}$$

1.7 $$\begin{cases} y'''(t) + \exp(t) y''(t) - 10ty'(t) + y(t) = 1 \quad t \in [0, 8.9] \\ y(0) = 1,\ y'(0) = -2,\ y''(0) = 0 \end{cases}$$

1.8 $$\begin{cases} y''(t) = 10t(y(t) + z(t)) \quad t \in [0, 2.42] \\ z'(t) = (1 - y'(t))z(t) + \mathrm{e}^t \\ y(0) = 1,\ y'(0) = 1,\ z(0) = 0 \end{cases}$$

2. Resuelve los siguientes problemas de contorno.

2.1 $$\begin{cases} y''(t) = ty^3(t) - 1 - \operatorname{sen} t \quad t \in [0, \dfrac{\pi}{2}] \\ y(0) = 0,\ y(\dfrac{\pi}{2}) = 1 \end{cases}$$

2.2 $$\begin{cases} y^{(4)}(t) = y''(t) - y^3(t) + 2t \quad t \in [0,1] \\ y(0) = y'(0) = 0, y''(0) = 1 \\ y'(1) + y''(1) = 1 \end{cases}$$

2.3 $$\begin{cases} y^{(4)}(t) = y'(t)\operatorname{sen}(y(t)) - ty''(t) \quad t \in [0, \pi] \\ y(0) = y'(0) = 1, y''(0) = 0, y(\pi) = 1 \end{cases}$$

Bibliografía

[1] A. Aubanell, A. Benseny, and A. Delshams. *Útiles Básicos de Cálculo Numérico*. Editorial Labor, S.A., Barcelona, 1993.

[2] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd., Chichester, England, 2003.

[3] M. Calvo and J.I. Montijano. *Resolución Numérica de Ecuaciones Diferenciales Ordinarias: Métodos de Runge-Kutta*. Dpto. de Ecuaciones Funcionales, Universidad de Zaragoza, Zaragoza, 1984.

[4] M. Crouzéix and A.L. Mignot. *Analyse Numérique des Équations Différentielles*. Masson, París, 2nd edition, 1989.

[5] E. Hairer and G. Wanner. *Introduction à l'Analyse Numérique*. Notas disponibles en la red, Universidad de Ginebra, 2005.

[6] G. Hämmerlin and K.H. Hoffmann. *Numerical Mathematics*. Springer Verlag, Heidelberg-Berlin-New York, 1991.

[7] P. Hartman. *Ordinary Differential Equations*. Mathematics and its Applications. John Wiley & Sons, Inc, New York, 1964.

[8] E. Isaacson and H.B. Keller. *Analysis of Numeical Methods*. Dover Publications, Inc, New York, 1994.

[9] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice Hall, Englewood Cliffs. New Jersey, 1989.

[10] J. D. Lambert. *Numerical methods for ordinary differential systems*. John Wiley & Sons, Ltd., Chichester, 1991. The initial value problem.

[11] L. F. Shampine. *Numerical solution of ordinary differential equations*. Chapman & Hall, New York, 1994.

[12] E. Süli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, Cambridge, 2003.

[13] J. H. Wilkinson. The perfidious polynomial. In *Studies in numerical analysis*, volume 24 of *MAA Stud. Math.*, pages 1–28. Math. Assoc. America, Washington, DC, 1984.