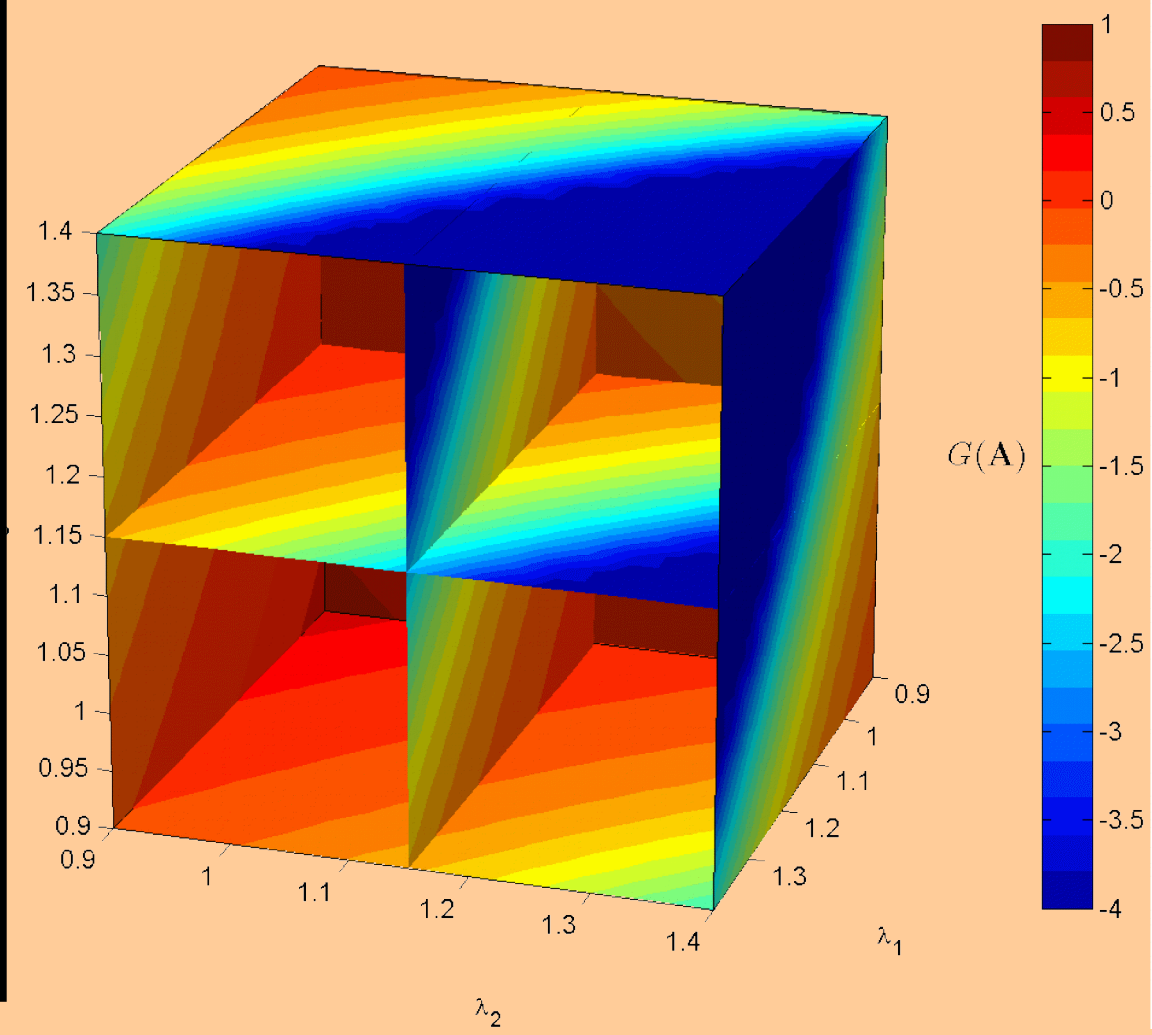
Fundamentos del Análisis Computacional

Pablo V. Negrón Marrero



Fundamentos del Análisis Computacional

Pablo V. Negrón Marrero
Universidad de Puerto Rico
Departamento de Matemáticas
Humacao, PR 00791

Tabla de Contenido

Prefacio

Sobre el Texto

En este libro presentamos una introducción a los métodos numéricos comúnmente utilizados en las aplicaciones científicas. Discutimos tanto aspectos computacionales como análisis de errores de los métodos, con un poco más de énfasis tal vez en lo último. Para la parte computacional utilizamos la plataforma numérica MATLAB ™(Versión 5.2) por su amplio uso en la comunidad científica, variedad de funciones numéricas, y por su sintaxis *transparente* que permite codificar muchos de los métodos numéricos en forma directa. El libro está diseñado para un curso en métodos numéricos para estudiantes de tercero o cuarto año de licenciatura o bachillerato. Los requisitos matemáticos del mismo son: cálculo diferencial e integral incluyendo conceptos básicos de sucesiones y series; y álgebra lineal, en particular el algebra de matrices, determinantes, y los métodos básicos para la solución de sistemas lineales. Se espera también que el estudiante tenga experiencia programando en algún lenguaje de alto nivel.

Los tópicos cubiertos en el libro son: aproximación de funciones mediante polinomios de Taylor, de interpolación, o cuadrados mínimos; métodos numéricos para sistemas lineales y su análisis de estabilidad; métodos para ecuaciones nolineales; diferenciación e integración mumérica; y la solución numérica de problemas de ecuaciones diferenciales con valor inicial. El material del texto es más que suficiente para un curso introductorio de un semestre en métodos numéricos. Una posible selección de temas para un curso con mas énfasis en los aspectos computacionales de los métodos podría ser: Caps. 1-2, Cap. 3 (Secciones 1–5, 8), Cap. 4 (Secciones 1–4,6–7), Cap. 5 (Secciones 1–4,6–7), Cap. 6 (Secciones 1.1–1.3, 2.1–2.2), Cap. 7 (Secciones 1,3–4). El resto del material del libro puede ser utilizado a discreción del instructor ya sea para diseñar un curso más avanzado o profundo o para lecturas y/o tra-

bajos adicionales para los estudiantes. Los ejercicios al final de cada capítulo incluyen ejercicios computacionales los cuales pueden ser utilizados en una sesión de laboratorio para el curso.

Sobre la Ciencia Computacional

El método que caracteriza a la ciencia computacional se puede describir como sigue:

i) el planteamiento del problema real o físico que se desea estudiar;

ii) la formulación del modelo matemático que se usa para describir el problema real;

iii) el análisis matemático y/o computacional del modelo matemático;

iv) la validación del modelo.

Todo modelo de un problema real conlleva una serie de hipótesis que se añaden para simplificar el modelo y que usualmente consisten en descartar o ignorar algunos efectos o parámetros del problema estudiado. Es importante conocer estas "limitaciones" del modelo para así evitar su mal uso. Por lo general el modelo matemático viene dado por un sistema de ecuaciones lineales o nolineales, una ecuación integral o diferencial, u otro tipo de ecuación o sistema de ecuaciones. En ocasiones estas ecuaciones se pueden estudiar matemáticamente, inclusive hasta resolver de forma exacta. Pero lo común es que sea muy difícil o imposible hallar estas soluciones exactas y es necesario obtener soluciones mediante la computadora. La validación del modelo consiste en comparar los resultados obtenidos, ya sea numéricamente o por análisis matemático, con datos reales u observaciones. Si estos resultados son consistentes con los observados, el modelo queda validado (al menos con relación a lo observado). De lo contrario el modelo debe ser revisado y se repite el análisis, computación, y validación. (Vea la Figura 1)

Los métodos computacionales pueden ser de tipo *algebraicos* o *numéricos*. Los métodos algebraicos intentan obtener soluciones exactas mediante la computación simbólica. Estos métodos, aunque poderosos, son computacionalmente complejos y un tanto limitados en su aplicabilidad. Los métodos numéricos, por el contrario, son más eficientes en cuanto a su implantación en la computadora. Esto se logra ya que no se insiste en soluciones exactas.

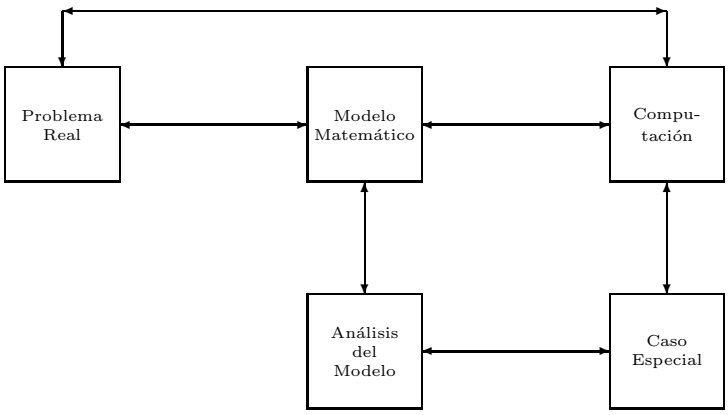


Figura 1: Elementos característicos de la ciencia computacional y sus interrelaciones.

Pero entonces tenemos que lidiar con las diferentes fuentes de error. Los errores son esencialmente de tres tipos:

- errores presentes en los datos iniciales o insumo;
- error al aproximar el problema original por uno más simple. Esto se conoce como el *error de truncación*[1].
- Errores debido a la aritmética *finita*[2].

Los errores en los datos iniciales provienen mayormente de las limitaciones o precisión en los datos que se entran a la computadora y que describen el problema que se va a resolver. Muchos de estos datos corresponden a medidas hechas en el laboratorio y su exactitud depende del equipo utilizado. No podemos controlar o eliminar estos errores de nuestros cómputos pero si debemos asegurarnos que los resultados que se obtengan no contengan más

[1]En algunos textos, este tipo de error se denomina como *error de discretización*.

[2]El termino *finito* aquí se refiere a las limitaciones de almacenamiento en la computadora.

error que el presente en los datos. A un método numérico que logre esto se le llama *estable*.

El error de truncación es el que estudiaremos mayormente en este libro y proviene de aproximar el problema *continuo* por uno *discreto*. Por ejemplo, aproximar un integral con una suma de Riemann, una derivada con un cociente diferencial, o una función nolineal con su aproximación lineal. Esto es, aproximamos el problema original con uno que es más simple para resolver. Este error se analiza usando las técnicas del cálculo avanzado siendo la herramienta más común el *Teorema de Taylor*.

Los errores debido a la aritmética finita son causados por el uso de las operaciones de punto flotante en la computadora. En cada suma, multiplicación, o división en la computadora, se puede introducir un error de redondeo si el resultado no cabe en el registro de la computadora. Aunque estos errores son pequeños (del orden del épsilon de la maquina), estos se pueden acumular en ocasiones teniendo resultados desastrosos[3]. Es importante, pues, entender como estos errores se propagan en los cálculos.

Otras fuentes de error que no hemos mencionado pero que están presentes al usar un computador son los errores de programación y los posibles defectos de la computadora que se utilice[4]. Aunque estos tipos de errores no los estudiaremos directamente en este libro, es importante tener en cuenta que la implantación del método numérico debe también ser validada. Para esto es útil un caso especial del modelo para el cual se conozca la solución exacta y así poder hacer comparaciones con lo calculado al igual que corroborar los resultados haciendo corridas de los métodos en diferentes computadores.

Agradecimientos

Las versiones preliminares de este libro fueron utilizadas por varios años en el curso de métodos numéricos de la Universidad de Puerto Rico en Humacao y Rio Piedras. Le agradezco a todos los estudiantes que con sus sugerencias y comentarios ayudaron a mejorar el texto. Quiero también agradecer a los compañeros José Sotero y Mariano Marcano quienes utilizaron el libro en sus cursos y me ofrecieron valiosos comentarios.

[3] Véase por ejemplo los artículos [20], [27], y [32].

[4] Estos defectos pueden ser de programado, como compiladores defectuosos, o de componentes de la computadora como el famoso problema con los primeros procesadores Pentium™. (Vea [7], [17] para discusiones y referencias adicionales sobre esto último.)

Capítulo 1

Conceptos Básicos

En este capítulo vamos a repasar algunos de los conceptos más importantes del cálculo los cuales son utilizados frecuentemente en el estudio de métodos numéricos. Uno de éstos resultados, quizás de los más utilizados, es el llamado *Teorema de Taylor*. Con este teorema, entre otras cosas, podemos aproximar funciones y a su vez obtener estimados de los errores en dichas aproximaciones. Otro resultado matemático bien importante es el del *Teorema de la función implícita*. En este capitulo discutimos este teorema pero con enfoque o punto de vista computacional donde construimos aproximaciones de las funciones implícitas con polinomios de Taylor.

Cuando usamos el Teorema de Taylor u otra técnica numérica para aproximar funciones, por lo general lo que hacemos es aproximar una cierta función con un polinomio. Por tal razón es de vital importancia el poder evaluar estos polinomios de forma eficiente en una computadora digital. Como veremos mas adelante, el método mas eficiente para hacer estas evaluaciones es el de *división sintética* que aprendimos en los cursos básicos de matemáticas.

De interesarle, el lector podrá encontrar más detalles sobre los resultados teóricos en este capítulo consultando alguna referencia sobre análisis matemático como por ejemplo [1], [24], [25].

1.1 Teorema de Taylor

Un problema común cuando trabajamos con aplicaciones de las matemáticas, es el de evaluar una función f para un valor dado de x. Aunque esto parece

un problema sencillo[1], la función f podría ser desconocida o complicada para evaluar. Para realizar estas evaluaciones lo que hacemos es aproximar a f con una función g que sea más *fácil* de evaluar que f. Los polinomios de Taylor son una forma de hacer esto aunque veremos otras alternativas más adelante.

Suponga que conocemos los datos $f(a)$ y $f'(a)$. Buscamos un polinomio $p_1(x)$ tal que $p_1(a) = f(a)$, $p_1'(a) = f'(a)$. Sabemos que $p_1(x) = a_1 + a_2x$ (¿por qué?) de modo que:

$$\begin{cases} a_1 + a_2 a &= p_1(a) = f(a), \\ a_2 &= p_1'(a) = f'(a), \end{cases}$$

de donde obtenemos que $p_1(x) = f(a) + f'(a)(x-a)$ el cual se conoce como *el polinomio de Taylor de grado uno para f en $x = a$.*

De igual forma, si conocemos $f(a)$, $f'(a)$, $f''(a)$ podemos construir un polinomio de grado dos $p_2(x)$ tal que $p_2(a) = f(a)$, $p_2'(a) = f'(a)$, $p_2''(a) = f''(a)$. Trabajando como antes obtenemos que

$$p_2(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2,$$

lo que se conoce como *el polinomio de Taylor de grado dos para f en $x = a$.* Note que $p_2(x)$ se obtiene a partir de $p_1(x)$ sumando un termino cuadrático que se puede interpretar como una *correción.*

Pasamos al caso general en que $f(a), f'(a), \ldots, f^{(n)}(a)$ son conocidas, y queremos hallar un polinomio $p_n(x)$ de grado a lo más n tal que

$$p_n^{(j)}(a) = f^{(j)}(a), \quad 0 \leq j \leq n. \tag{1.1}$$

Pensando en la observación al final del párrafo anterior, escribimos

$$p_n(x) = p_{n-1}(x) + q_n(x), \tag{1.2}$$

donde $p_{n-1}(x)$ es el polinomio de grado a lo más $n-1$ tal que

$$p_{n-1}^{(j)}(a) = f^{(j)}(a), \quad 0 \leq j \leq n-1, \tag{1.3}$$

[1]Aunque hoy en día podemos evaluar muchas funciones con calculadoras de mano, es importante tener en perspectiva que los procesos que utilizan estos dispositivos, están basados en el Teorema de Taylor.

y $q_n(x)$ es un polinomio de grado a lo más n. Note que (1.2) y (1.3) implican que

$$q_n^{(j)}(a) = 0, \quad 0 \leq j \leq n-1.$$

De aquí que $q_n(x) = a_n(x-a)^n$ y (1.2) reduce a

$$p_n(x) = p_{n-1}(x) + a_n(x-a)^n.$$

La condición $p_n^{(n)}(a) = f^{(n)}(a)$ implica ahora que $a_n = f^{(n)}(a)/n!$, es decir

$$p_n(x) = p_{n-1}(x) + \frac{f^{(n)}(a)}{n!}(x-a)^n. \tag{1.4}$$

Usando esta fórmula e inducción matemática, se puede verificar (Ejercicio 1.3) que

$$p_n(x) = \sum_{j=0}^{n} \frac{f^{(j)}(a)}{j!}(x-a)^j, \tag{1.5}$$

lo que se conoce como *el polinomio de Taylor de grado a lo más n para f en* $x = a$.

Ejemplo 1.1. Considere la función $f(x) = 1/(1+x)$ y tomemos $a = 0$. Entonces $f'(x) = -(1+x)^{-2}$, $f''(x) = 2(1+x)^{-3}$, $f'''(x) = -(2)(3)(1+x)^{-4}$, de donde podemos concluir en forma inductiva que

$$f^{(n)}(x) = \frac{(-1)^n n!}{(1+x)^{n+1}}, \quad n \geq 0.$$

Tenemos ahora que

$$\begin{aligned} p_n(x) &= \sum_{j=0}^{n} \frac{f^{(j)}(0)}{j!}(x-0)^j, \\ &= \sum_{j=0}^{n} \frac{(-1)^j j!}{j!}x^j, \\ &= 1 - x + x^2 - x^3 + \cdots + (-1)^n x^n. \end{aligned}$$

En particular

$$p_1(x) = 1 - x, \quad p_2(x) = 1 - x + x^2.$$

Podemos ahora trazar p_1, p_2, y f en el mismo sistema de coordenadas con las siguientes instrucciones en MATLAB:

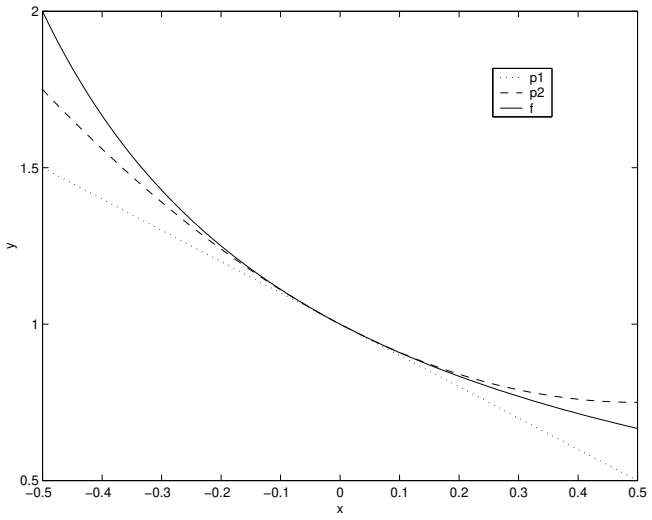


Figura 1.1: Gráfica de la función $f(x) = 1/(1+x)$ con sus polinomios de Taylor de grados uno y dos.

```
x=linspace(-0.5,0.5,100);
p1=1-x;
p2=p1+x.^2;
f=1./(1+x);
plot(x,p1,x,p2,x,f)
xlabel('x');ylabel('y');
legend('p1','p2','f')
```

La gráfica resultante se muestra en la Figura 1.1. Note que el polinomio de grado dos aproxima mejor a f en el intervalo en cuestión[2] pero la diferencia entre ambos polinomios y la función f (el error) aumenta según nos alejamos del centro $a = 0$. Observe también que al evaluar p_2 lo hacemos añadiendo solamente un termino cuadrático a p_1 lo cual reduce la cantidad de operaciones aritméticas en el computo de p_2. □

Queremos ahora de alguna forma cuantificar las observaciones del ejemplo anterior en cuanto al error de aproximación de los polinomios de Taylor.

[2]No necesariamente el aumentar el grado del polinomio implica una mejor aproximación. Esto se debe a las altas fluctuaciones en los polinomios de alto grado. Volveremos a este punto en la Sección (5.5).

Utilizamos el símbolo $C^n(\alpha, \beta)$ para denotar el conjunto de funciones que tiene n derivadas continuas en el intervalo (α, β). Tenemos pues:

Teorema 1.2 (de Taylor)**.** *Suponga que $f \in C^{n+1}(\alpha, \beta)$ y sea $a \in (\alpha, \beta)$. Sea $R_n(x) = f(x) - p_n(x)$ el error al aproximar f con p_n. Entonces*

$$R_n(x) = \frac{1}{n!}\int_a^x f^{(n+1)}(\xi)(x-\xi)^n \, \mathrm{d}\xi = \frac{f^{(n+1)}(c_x)}{(n+1)!}(x-a)^{n+1}, \quad x \in (\alpha, \beta), \tag{1.6}$$

donde c_x es un número entre a y x.

Ejemplo 1.3. Considere la función[3] $f(x) = \exp(2x)$ y tomamos nuevamente $a = 0$. Aquí $f^{(j)}(x) = 2^j \exp(2x)$ para toda $j \geq 0$. Así que

$$p_n(x) = \sum_{j=0}^{n} \frac{2^j}{j!}x^j = 1 + 2x + 2x^2 + \cdots + \frac{2^n}{n!}x^n.$$

El error de aproximación está dado por

$$R_n(x) = \frac{2^{n+1}\exp(2c_x)}{(n+1)!}x^{n+1},$$

donde c_x está entre cero y x. El caso especial $x = 1/2$ nos provee de una fórmula para aproximar el número e:

$$e = \exp(1) = 1 + 1 + \frac{1}{2} + \cdots + \frac{1}{n!} + R_n\left(\frac{1}{2}\right).$$

Note que

$$\left|R_n\left(\frac{1}{2}\right)\right| \leq \frac{\exp(2c)}{(n+1)!} < \frac{3}{(n+1)!},$$

donde usamos que como c está entre cero y $1/2$, entonces $\exp(2c) < e < 3$. Si queremos aproximar e con la fórmula de arriba con un error (absoluto) de 10^{-t} entonces podemos estimar n mediante el siguiente programa en MATLAB para el caso $t = 6$:

[3]Utilizamos la notación $\exp(t)$ para representar e^t. Esta es la notación que usan la mayoría de los lenguajes de alto nivel, como MATLAB, para representar la función exponencial natural.

```
t=6;
n=0;
R=3;
tol=10^(-t);
while R>tol
  n=n+1;
  R=3/factorial(n+1);
end
n
```

lo cual produce el resultado de $n = 9$. Note que la instrucción

```
R=3/factorial(n+1)
```

se puede sustituir con `R=R/(n+1)` lo que resulta más eficiente. □

Ejemplo 1.4. Vamos a aproximar la función $\text{sen}(x)$ en el intervalo $[-\pi/2, \pi/2]$ con un error de 10^{-4}. Sabemos que $f'(x) = \cos(x)$, $f''(x) = -\text{sen}(x)$, $f'''(x) = -\cos(x)$, etc.. Tomando $a = 0$ obtenemos pues que $f'(0) = 1$, $f''(0) = 0$, $f'''(0) = -1$, etc., de donde podemos concluir que

$$\begin{cases} f^{(2k)}(0) = 0, & k \geq 0, \\ f^{(2k-1)}(0) = (-1)^{k-1}, & k \geq 1. \end{cases}$$

Ahora

$$\begin{aligned} \text{sen}(x) &= \sum_{j=0}^{n} \frac{f^{(j)}(0)}{j!} x^j + \frac{f^{(n+1)}(c_x)}{(n+1)!} x^{n+1}, \\ &= \begin{cases} \displaystyle\sum_{k=1}^{n/2} \frac{(-1)^{k-1}}{(2k-1)!} x^{2k-1} + \frac{(-1)^{n/2}\cos(c_x)}{(n+1)!} x^{n+1}, & n \text{ par}, \\ \displaystyle\sum_{k=1}^{(n+1)/2} \frac{(-1)^{k-1}}{(2k-1)!} x^{2k-1} + \frac{(-1)^{(n+1)/2}\text{sen}(c_x)}{(n+1)!} x^{n+1}, & n \text{ impar}, \end{cases} \end{aligned}$$

donde c_x está en $[-\pi/2, \pi/2]$. Note que en cualquier caso, n par o impar, para $x \in [-\pi/2, \pi/2]$ el residual $R_n(x)$ está acotado por:

$$|R_n(x)| \leq \frac{(\pi/2)^{n+1}}{(n+1)!}.$$

Podemos calcular la n que hace el error menor de 10^{-4} digamos mediante el siguiente programa en MATLAB:

```
t=4;
n=0;
R=pi/2;
tol=10^(-t);
while R>tol
  n=n+1;
  R=(pi/2)^n/gamma(n+2);
end
n
```

lo cual produce el valor de $n = 9$. Tenemos pues que el polinomio

$$p_9(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!},$$

aproxima a la función $\text{sen}(x)$ en $[-\pi/2, \pi/2]$ con error no mayor de 10^{-4}. □

Suponga que tenemos una función f definida de alguna forma en términos de otra función g. Suponga también que conocemos o que podemos calcular un polinomio de Taylor de g. En ocasiones podemos calcular un polinomio de Taylor para f utilizando el de g, más fácilmente que trabajando directamente con f. Antes de ver un ejemplo de ésto, repasamos la siguiente fórmula:

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots + x^n + \frac{x^{n+1}}{1-x}, \quad x \neq 1. \tag{1.7}$$

Esta fórmula se demuestra fácilmente por inducción matemática. En el caso en que $|x| < 1$, podemos dejar que $n \longrightarrow \infty$, obteniendo así la formula para la *serie geométrica*:

$$\frac{1}{1-x} = \sum_{j=0}^{\infty} x^j. \tag{1.8}$$

Ejemplo 1.5. Vamos a calcular un polinomio de Taylor con su residuo para la función $\log(1 + x^2)$. Primero observamos que para $t > -1$,

$$\log(1+t) = \int_0^t \frac{dr}{1+r}.$$

Usando (1.7) podemos escribir

$$\frac{1}{1+r} = 1 - r + r^2 + \cdots + (-1)^n r^n + \frac{(-1)^{n+1} r^{n+1}}{1+r}, \quad r \neq -1.$$

De modo que

$$\begin{aligned}\log(1+t) &= \int_0^t (1-r+r^2+\cdots+(-1)^n r^n)\,dr + \int_0^t \frac{(-1)^{n+1}r^{n+1}}{1+r}\,dr,\\ &= t-\frac{t^2}{2}+\frac{t^3}{3}-\cdots+(-1)^n\frac{t^{n+1}}{n+1}+(-1)^{n+1}\int_0^t \frac{r^{n+1}}{1+r}\,dr.\end{aligned}$$

Sustituyendo ahora $t = x^2$ obtenemos

$$\begin{aligned}\log(1+x^2) &= x^2-\frac{x^4}{2}+\frac{x^6}{3}-\cdots+(-1)^n\frac{x^{2(n+1)}}{n+1}\\ &\quad +(-1)^{n+1}\int_0^{x^2}\frac{r^{n+1}}{1+r}\,dx,\\ &= x^2-\frac{x^4}{2}+\frac{x^6}{3}-\cdots+(-1)^n\frac{x^{2(n+1)}}{n+1}+\frac{(-1)^{n+1}}{(1+c_x)}\frac{x^{2(n+2)}}{n+2},\end{aligned}$$

donde usamos el Teorema del Valor Medio (Teorema (A.2)) para integrales y c_x es un número entre cero y x^2. Esta fórmula nos da un polinomio para aproximar $\log(1+x^2)$ con la correspondiente fórmula del error. □

Ejemplo 1.6. Sea f una función tal que:

$$f(2) = -1, \quad f'(2) = \frac{1}{2}, \quad f''(2) = -\frac{1}{6}, \quad f'''(2) = -5,$$

$$\left|f^{(iv)}(x)\right| \leq 0.145, \quad x \in [1,3].$$

El polinomio de Taylor de grado tres para f alrededor de $a = 2$ esta dado por:

$$p_3(x) = -1+\frac{1}{2}(x-2)-\frac{1}{12}(x-2)^2-\frac{5}{6}(x-2)^3.$$

El residual $R_3(x)$ del Teorema de Taylor para este polinomio en el intervalo $[1,3]$ es:

$$R_3(x) = \frac{f^{(iv)}(c_x)}{4!}(x-2)^4,$$

donde c_x está entre x y 2 y $x \in [1,3]$.

Suponga ahora que $F(x) = \int_2^x f(t)\,\mathrm{d}t$ con $x \in [1,3]$. Usando el polinomio de Taylor con su residual podemos escribir que

$$F(x) = \int_2^x [p_3(t)+R_3(t)]\,\mathrm{d}t,$$

$$= -(x-2) + \frac{1}{4}(x-2)^2 - \frac{1}{36}(x-2)^3 - \frac{5}{24}(x-2)^4 + \int_2^x R_3(t)\,\mathrm{d}t.$$

Suponga ahora que queremos aproximar $F(1.5)$. Tenemos de la expresión anterior que

$$F(1.5) \approx -(1.5-2) + \frac{1}{4}(1.5-2)^2 - \frac{1}{36}(1.5-2)^3 - \frac{5}{24}(1.5-2)^4 = 0.5530.$$

El error en esta aproximación lo podemos estimar usando el residual en la expresión de $F(x)$ de arriba:

$$\begin{aligned}\left|\int_2^{1.5} R_3(t)\,\mathrm{d}t\right| &\leq \int_{1.5}^{2} |R_3(t)|\ \mathrm{d}t, \\ &\leq \frac{0.145}{4!}\int_{1.5}^{2}(t-2)^4\mathrm{d}t = 3.7760 \times 10^{-5}.\end{aligned}$$

□

1.2 El Teorema de la Función Implícita

Suponga que $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ es una función suficientemente diferenciable y considere la ecuación

$$f(x, y) = 0.$$

Nos interesa saber cuando es posible despejar esta ecuación para y en términos de x. En el caso lineal $f(x, y) = ax + by + c$, es fácil ver que podemos hacer ésto siempre que $b \neq 0$. Pero en el caso general en que f sea no–lineal, la contestación no es tán obvia.

Ejemplo 1.7. Si consideramos la ecuación:

$$x^2 + y^2 - 1 = 0,$$

podemos despejar para y en términos de x pero la contestación no es única: $y = \pm\sqrt{1-x^2}$. Si (x_0, y_0) es un punto que satisface la ecuación, i.e., $x_0^2 + y_0^2 - 1 = 0$, entonces podemos preguntarnos si es posible despejar para y en términos de x de forma única, al menos "cerca" de x_0. En este caso si $y_0 > 0$, la contestación es $y = \sqrt{1-x^2}$. Si $y_0 < 0$ tenemos la misma formula pero con un menos al frente. Si $y_0 = 0$, entonces no es posible despejar para y en términos de x. ¿Qué representa la ecuación $x^2 + y^2 - 1 = 0$? □

Ejemplo 1.8. El análisis del ejemplo anterior ya no es posible para la ecuación

$$y\mathrm{e}^y - x = 0,$$

cerca del punto $(0,0)$ digamos. Aunque en este caso no podemos despejar para y en términos de x de forma *explícita*, veremos que todavía es posible garantizar de forma *implícita* que tal función de y en términos de x existe. Además veremos un procedimiento para aproximar dicha función. □

El siguiente resultado nos dá condiciones suficientes que garantizan que una ecuación no–lineal como $f(x,y) = 0$ se puede resolver para y en términos de x.

Teorema 1.9 (de la Función Implícita)**.** *Sea $D \subset \mathbb{R}^2$ un conjunto abierto, $f : D \to \mathbb{R}$ continua y con derivadas parciales continuas en D. Suponga que $(x_0, y_0) \in D$ es tal que*

$$f(x_0, y_0) = 0, \quad \frac{\partial f}{\partial y}(x_0, y_0) \neq 0. \tag{1.9}$$

Entonces existe un número $\epsilon > 0$ y una función $\phi : [x_0 - \epsilon, x_0 + \epsilon] \to \mathbb{R}$ continua y con derivada continua tal que

$$\begin{cases} f(x, \phi(x)) = 0, & x \in [x_0 - \epsilon, x_0 + \epsilon], \\ \phi(x_0) = y_0. \end{cases} \tag{1.10}$$

El teorema establece que dadas las condiciones (1.9) podemos resolver la ecuación $f(x,y) = 0$ para y en términos de x, esto es, $y = \phi(x)$, en un intervalo alrededor de x_0. Si en (1.9) cambiamos a $(\partial f/\partial x)(x_0, y_0) \neq 0$, entonces tenemos condiciones para resolver para x en términos de y.

Ejemplo 1.10. En el Ejemplo 1.8 tenemos con $f(x,y) = y\mathrm{e}^y - x$ que:

$$\left.\frac{\partial f}{\partial y}(x,y)\right|_{(0,0)} = (y+1)\mathrm{e}^y|_{(0,0)} = 1 \neq 0,$$

por lo que existe una función $y = \phi(x)$ con $\phi(0) = 0$ y tal que $\phi(x)\mathrm{e}^{\phi(x)} - x = 0$ para x en un intervalo alrededor de cero. □

Aunque la función $y = \phi(x)$ del teorema está dada de forma implícita, podemos utilizar la relación (1.10) para aproximar a ϕ. En particular veremos que es posible utilizar (1.10) para calcular polinomios de Taylor de ϕ

alrededor de $x = x_0$ del grado deseado siempre que la f sea suficientemente diferenciable. Para esto note que si diferenciamos $f(x, \phi(x)) = 0$ con respecto a x tenemos que

$$\frac{\partial f}{\partial x}(x, \phi(x)) + \frac{\partial f}{\partial y}(x, \phi(x))\phi'(x) = 0. \tag{1.11}$$

Si evaluamos esta ecuación ahora en $x = x_0$ y usamos que $\phi(x_0) = y_0$, llegamos a que:

$$\phi'(x_0) = -\frac{\frac{\partial f}{\partial x}(x_0, y_0)}{\frac{\partial f}{\partial y}(x_0, y_0)}. \tag{1.12}$$

Si diferenciamos ahora (1.11) con respecto a x tenemos que:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2}(x, \phi(x)) + 2\frac{\partial^2 f}{\partial x \partial y}(x, \phi(x))\phi'(x) \\ +\frac{\partial^2 f}{\partial y^2}(x, \phi(x))(\phi'(x))^2 + \frac{\partial f}{\partial y}(x, \phi(x))\phi''(x) = 0.\end{aligned} \tag{1.13}$$

Nuevamente evaluamos en $x = x_0$ y llegamos a que

$$\phi''(x_0) = -\frac{\frac{\partial^2 f}{\partial x^2}(x_0, y_0) + 2\frac{\partial^2 f}{\partial x \partial y}(x_0, y_0)\phi'(x_0) + \frac{\partial^2 f}{\partial y^2}(x_0, y_0)(\phi'(x_0))^2}{\frac{\partial f}{\partial y}(x_0, y_0)},$$

donde usamos que $\phi(x_0) = y_0$ y la expresión (1.12) se usa en el lugar de $\phi'(x_0)$. Este proceso se puede repetir cuantas veces sea necesario. Con las derivadas obtenidas de ϕ en $x = x_0$, podamos ahora construir un polinomio de Taylor para ϕ alrededor de $x = x_0$ con el cual podemos aproximar a la función implícita $y = \phi(x)$.

Ejemplo 1.11. Consideramos nuevamente el problema de resolver la ecuación $ye^y - x = 0$ para y en términos de x cerca del punto $(0, 0)$. Con $f(x, y) = ye^y - x$ tenemos que la función implícita $y = \phi(x)$ satisface que

$$\phi(x)e^{\phi(x)} - x = 0,$$

con $\phi(0) = 0$. Diferenciando con respecto a x obtenemos que:

$$\phi'(x)e^{\phi(x)} + \phi(x)\phi'(x)e^{\phi(x)} - 1 = 0. \tag{1.14}$$

Sustituyendo $x = 0$ y usando que $\phi(0) = 0$, llegamos a que:

$$\phi'(0) = 1.$$

Repetimos ahora el proceso pero diferenciando con respecto a x la ecuación (1.14):

$$\left[\phi''(x) + 2(\phi'(x))^2 + \phi(x)\phi''(x) + \phi(x)(\phi'(x))^2\right] \mathrm{e}^{\phi(x)} = 0. \tag{1.15}$$

Esta expresión la evaluamos en $x = 0$, usando que $\phi(0) = 0$ y $\phi'(0) = 1$ para obtener que

$$\phi''(0) = -2.$$

Note que (1.15) simplifica a:

$$\phi''(x) + 2(\phi'(x))^2 + \phi(x)\phi''(x) + \phi(x)(\phi'(x))^2 = 0.$$

Esta ecuación la diferenciamos ahora con respecto a x obteniendo así que:

$$\phi'''(x) + 5\phi'(x)\phi''(x) + \phi(x)\phi'''(x) + (\phi'(x))^3 + 2\phi(x)\phi'(x)\phi''(x) = 0.$$

Si evaluamos en $x = 0$ y usamos los valores de $\phi(0)$, $\phi'(0)$, y $\phi''(0)$ que ya tenemos, concluimos que

$$\phi'''(0) = 9.$$

Tenemos ahora que el polinomio de Taylor de grado tres para aproximar a $\phi(x)$ con x cerca de cero es:

$$p_3(x) = x + x^2 + \frac{3}{2}x^3,$$

y que

$$\phi(x) = x + x^2 + \frac{3}{2}x^3 + \text{ términos proporcionales a } x^4.$$

□

Este proceso lo podemos generalizar de la forma siguiente. Observe que si despejamos para $\phi'(x)$ en (1.11) tenemos que:

$$\phi'(x) = -\left.\frac{\frac{\partial f}{\partial x}(x,y)}{\frac{\partial f}{\partial y}(x,y)}\right|_{y=\phi(x)} \equiv f_1(x,y)|_{y=\phi(x)}\,.$$

Despejando para $\phi''(x)$ en (1.13) y rearreglando, usando la definición de $f_1(x,y)$, tenemos que:

$$\begin{aligned}\phi''(x) &= \frac{\partial f_1}{\partial x}(x,\phi(x)) + \frac{\partial f_1}{\partial y}(x,\phi(x))\phi'(x), \\ &= \left[\frac{\partial f_1}{\partial x}(x,y) + \frac{\partial f_1}{\partial y}(x,y)f_1(x,y)\right]_{y=\phi(x)} \equiv f_2(x,y)|_{y=\phi(x)}\,.\end{aligned}$$

De igual forma:

$$\phi'''(x) = \left[\frac{\partial f_2}{\partial x}(x,y) + \frac{\partial f_2}{\partial y}(x,y)f_1(x,y)\right]_{y=\phi(x)} \equiv f_3(x,y)|_{y=\phi(x)}\,, \quad \text{etc..}$$

En general, si definimos

$$f_1(x,y) = -\frac{\frac{\partial f}{\partial x}(x,y)}{\frac{\partial f}{\partial y}(x,y)}, \tag{1.16a}$$

$$f_n(x,y) = \frac{\partial f_{n-1}}{\partial x}(x,y) + \frac{\partial f_{n-1}}{\partial y}(x,y)f_1(x,y), \quad n>1, \tag{1.16b}$$

entonces es fácil ver usando inducción matemática que:

$$\phi^{(n)}(x) = f_n(x,\phi(x)), \quad n \geq 1. \tag{1.17}$$

Note que como $\phi(x_0) = y_0$, tenemos que:

$$\phi^{(n)}(x_0) = f_n(x_0,y_0), \quad n \geq 1. \tag{1.18}$$

Este procedimiento para calcular un polinomio de Taylor para la función implícita se puede implementar en MATLAB utilizando el paquete o "toolbox" simbólico. El programa es como sigue:

```
function implicit_taylor(f,x,y,x0,y0,n)
x=sym(x);y=sym(y);
f=sym(f);
F=sym(zeros(1,n));P=F;
F(1)=simplify(-diff(f,x)/diff(f,y));
P(1)=x-x0;
for k=2:n;
    F(k)=simple(diff(F(k-1),x)+diff(F(k-1),y)*F(1));
```

```
    P(k)=(x-x0)^k/factorial(k);
end
TCoefs=[y0,subs(F,[x,y],[x0,y0])];
TPoly=simple(TCoefs*[1,P].');
pretty(TPoly)
```

Ejemplo 1.12. El polinomio de Taylor de grado 5 para la función implícita que expresa y en términos de x en la ecuación $ye^y - x = 0$ alrededor de $(0, 0)$, se obtiene con la secuencia de llamada:

```
implicit_taylor('y*exp(y)-x','x','y',0,0,5)
```

que produce el resultado:

```
     2        3        4   125  5
x - x  + 3/2 x  - 8/3 x  + --- x
                           24
```

Un polinomio de Taylor de grado 5 para la función implícita de y en términos de x en la ecuación $x + y^3 + y = 0$ alrededor de $(2, -1)$, se obtiene con la secuencia de llamada:

```
implicit_taylor('x+y^3+y','x','y',2,-1,5)
```

Se obtuvo el resultado:

```
                              2                3    75          4
- 1/2 - 1/4 x + 3/64 (x - 2)  - 7/512 (x - 2)  + ----- (x - 2)
                                                 16384

            213          5
          - ------ (x - 2)
            131072
```

Note que el termino constante del polinomio no es -1 ya que hubo una simplicación al utilizar la función `simple` de MATLAB. □

1.3 Evaluación de Polinomios

Suponga que vamos a evaluar el polinomio de Taylor (1.5) en el punto $x = z$. Para esto no debemos expandir las potencias $(x-a)^j$, $j = 2, \ldots, n$, para luego simplificar y reducir el polinomio a sumas de potencias de x, para finalmente sustituir $x = z$. Lo que hacemos es que primero calculamos $y = z - a$, y entonces evaluamos

$$p_n(z) = \sum_{j=0}^{n} \frac{f^{(j)}(a)}{j!}\, y^j.$$

Así que sin perdida de generalidad podemos considerar el problema de evaluar un polinomio cualquiera de la forma

$$p(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_{n+1} x^n. \tag{1.19}$$

Podemos evaluar este polinomio de tres maneras:

1. Calculamos los términos $a_j x^{j-1}$, $j = 1, 2, \ldots, n+1$, y luego sumamos los resultados. Esto lo podemos escribir en forma algorítmica como:

   Algoritmo 1.13. Dado un número z en el cual queremos evaluar el polinomio (1.19), calculamos:

   (a) $p = a_1$.

   (b) Para $j = 2, \ldots, n+1$,

   (i) $p = p + a_j z^{j-1}$.

   El calculo del monomio $a_j z^{j-1}$ toma $j-1$ multiplicaciones, y sumando para $j = 2, \ldots, n+1$, tenemos un total de $n(n+1)/2$ multiplicaciones en la evaluación.

2. En este método, calculamos primero x^2, $x^3 = x(x^2)$, $x^4 = x(x^3)$, etc., luego multiplicamos por los coeficientes y finalmente sumamos. El algoritmo en este caso está dado por:

   Algoritmo 1.14. Dado un número z en el cual queremos evaluar el polinomio (1.19), calculamos:

   (a) $p = a_1$, $y = z$.

   (b) Para $j = 2, \ldots, n+1$,

(i) $p = p + a_j y$, $y = yz$.

Claramente este proceso toma un total de $2n$ multiplicaciones para la evaluación del polinomio.

3. El tercer método para evaluar el polinomio (1.19), se conoce como el *Método de Horner* y corresponde exactamente al proceso de evaluación o división sintética que se enseña en el curso preparatorio al cálculo. La esencia del método reside en poder escribir el polinomio de forma anidada como:

$$p(x) = a_1 + (a_2 + (\cdots(a_{n-1} + (a_n + a_{n+1}x)x)\cdots)x)x.$$

Este método se puede describir mediante las siguientes iteraciones:

Algoritmo 1.15. Método de Horner para evaluar polinomios. Dado un número z en el cual queremos evaluar el polinomio (1.19), calculamos:

(a) $b_{n+1} = a_{n+1}$.

(b) Para $i = n, n-1, \ldots, 1$,

(i) $b_i = a_i + z\, b_{i+1}$.

Examinando el ciclo principal de este algoritmo vemos que este requiere n multiplicaciones únicamente para evaluar el polinomio.

En la siguiente tabla resumimos los totales de multiplicaciones para los tres métodos de evaluar un polinomio:

Método	Número de Multiplicaciones
Algoritmo (1.13)	$n(n+1)/2$
Algoritmo (1.14)	$2n$
Algoritmo (1.15) (Método de Horner)	n

Examinamos ahora en más detalles el método de Horner. Tenemos ahora:

Teorema 1.16. *Para el polinomio* $p(x) = a_1 + a_2x + a_3x^2 + \cdots + a_{n+1}x^n$, *defina* $b_1, b_2, \ldots, b_{n+1}$ *mediante el Algoritmo (1.15). Entonces* $p(z) = b_1$.

Demostración: Por inducción en el grado n del polinomio.

1. *(Paso básico)* Para $n = 1$, $p(x) = a_1 + a_2 x$, $b_2 = a_2$, $b_1 = a_1 + z\, b_2 = a_1 + z\, a_2 = p(z)$, lo cual demuestra que el resultado es cierto para $n = 1$.

2. *(Paso inductivo)* Vamos a suponer que el resultado es cierto para cualquier polinomio de grado $k \geq 1$. Entonces para $p(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_{k+2} x^{k+1}$ definimos:

   (a) $b_{k+2} = a_{k+2}$

   (b) Para $i = k+1, k, \ldots, 1$

      i. $b_i = a_i + z\, b_{i+1}$

   Defina el polinomio $q(x)$ de grado k por: $q(x) = a_1 + a_2 x + a_3 x^2 + \cdots + (a_{k+1} + a_{k+2} z) x^k$ y los correspondientes b's por:

   (a) $\hat{b}_{k+1} = a_{k+1} + a_{k+2} z$

   (b) Para $i = k, k-1, \ldots, 1$

      i. $\hat{b}_i = a_i + z\, \hat{b}_{i+1}$

   Observe que $p(z) = q(z)$ y que $b_i = \hat{b}_i$ para $1 \leq i \leq k+1$. Como el grado de $q(x)$ es k, tenemos por la hipótesis de inducción que $q(z) = \hat{b}_1$. Pero $p(z) = q(z) = \hat{b}_1 = b_1$ de modo que el resultado es también cierto para polinomios de grado $k+1$. El resultado queda pues establecido para cualquier $n \geq 1$.

□

Podemos implementar el método de Horner en MATLAB en forma eficiente. Aquí el arreglo `a(1:n)` contiene los coeficientes a_1, a_2, etc. del polinomio. El arreglo `z` puede contener más de un valor de modo que la función evalúa el polinomio en el conjunto de datos dados por `z`:

```
function pval=hornerV(a,z)
%
% n es el grado del polinomio mas uno.
%
n=length(a);
pval=a(n)*ones(size(z));
for k=n-1:-1:1
   pval=a(k)+z.*pval;
end
```

Ejemplo 1.17. Para trazar el polinomio $p(x) = 2-3x+4x^2-5x^3+7x^4+2x^5$ en el intervalo de $[-1, 1]$ podemos escribir:

```
a=[2 -3 4 -5 7 2];
z=linspace(-1,1,100);
y=hornerV(a,z);
plot(z,y)
xlabel('x');
ylabel('y');
```

lo que produce la gráfica que se muestra en la Figura 1.2. □

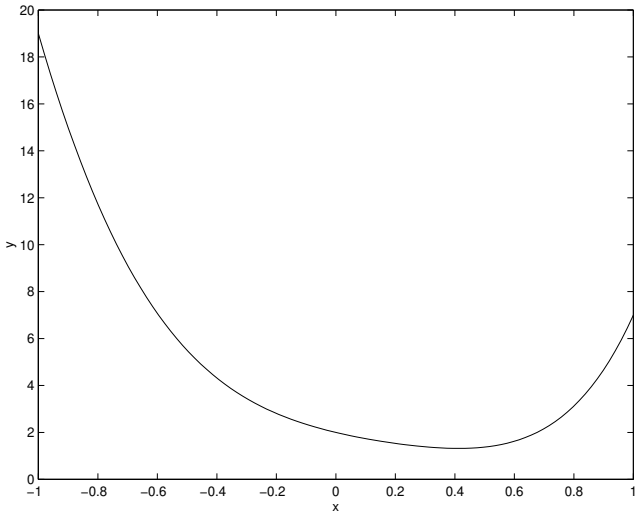


Figura 1.2: Gráfica del polinomio $p(x) = 2 - 3x + 4x^2 - 5x^3 + 7x^4 + 2x^5$ en el intervalo $[-1, 1]$.

Cuando el polinomio consiste de un número pequeño de términos, es posible hacer la evaluación del mismo de forma más eficiente que con el método de Horner (Ejercicio 1.6).

1.4 Rapidez de convergencia de sucesiones y la notación O

Muchos de los métodos numéricos que discutiremos más adelante son de carácter iterativo, es decir, que generan una sucesión de números, la cual bajo ciertas condiciones converge a la solución del problema estudiado. En ocasiones consideraremos dos o más métodos de solución y queremos poder comparar o medir de alguna forma la eficiencia de cada método. Una forma de comparar métodos iterativos es midiendo la rapidez o orden de convergencia de la sucesión de aproximaciones que estos generan. Vamos a precisar este concepto de rapidez de convergencia de una sucesión.

Definición 1.18. Suponga que la sucesión $(x_n)_{n=0}^{\infty}$ converge al número α. Decimos que $(x_n)_{n=0}^{\infty}$ converge a α con orden $p \geq 1$ si existen $M, N \geq 0$ tales que

$$|x_{n+1} - \alpha| \leq M |x_n - \alpha|^p, \quad n \geq N. \tag{1.20}$$

El número p, que no tiene que ser un entero, se llama el *orden de convergencia*. Si $p = 2$ la convergencia se llama *cuadrática*. Si $p = 1$ y $M < 1$ la convergencia se llama *lineal* con *tasa o razón* de convergencia M. También decimos que la convergencia es lineal con tasa o razón de convergencia M si para algún $0 \leq M < 1$ y $N \geq 0$, tenemos que

$$|x_n - \alpha| \leq CM^n, \quad n \geq N. \tag{1.21}$$

Nota: Mientras más grande sea p, más rápido converge la sucesión $(x_n)_{n=0}^{\infty}$ a α. Si $p = 1$, mientras más pequeño sea M, más rápido converge la sucesión. Es fácil también ver que si el límite

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^p}, \tag{1.22}$$

existe (como número real), entonces la sucesión (x_n) converge al número α con orden al menos p.

Ejemplo 1.19. Para la sucesión $x_n = 1/n^k$ donde $k > 0$ es un número dado, tenemos que con $\alpha = 0$,

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^p} = \lim_{n\to\infty} \frac{n^{pk}}{(n+1)^k} = \begin{cases} 1, & p = 1, \\ \infty, & p > 1. \end{cases}$$

Así que la convergencia es de orden uno pero como $M = 1$ (el límite cuando $p = 1$), la convergencia no es lineal. □

Ejemplo 1.20. Para la sucesión $x_n = a^n$ donde $0 < a < 1$ tenemos que con $\alpha = 0$:

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^p} = \lim_{n\to\infty} \frac{a^{n+1}}{a^{pn}} = \begin{cases} a, & p = 1, \\ \infty, & p > 1. \end{cases}$$

Tenemos pues que (x_n) converge a cero linealmente con razón o taza de convergencia $M = a$. En forma similar se puede verificar que $x_n = a^{2^n}$ converge a cero con orden dos. (Ver Ejercicio (1.18)). □

En ocasiones nos interesa medir o cuantificar el error en una cierta aproximación, como en el caso del Teorema de Taylor donde se aproxima la función f con el polinomio de Taylor p_n, sin entrar en los detalles de la representación o fórmula del error. Para esto utilizamos la notación O.

Definición 1.21. Sean $\psi(t), \phi(t)$ funciones definidas en una vecindad[4] N del punto $a \in \mathbb{R} \cup \{-\infty, \infty\}$. Decimos que

$$\psi(t) = O(\phi(t)), \quad t \longrightarrow a,$$

si existe una constante $C > 0$ tal que

$$|\psi(t)| \leq C\phi(t), \quad t \in N.$$

Decimos que

$$\psi(t) = o(\phi(t)), \quad t \longrightarrow a,$$

si

$$\lim_{t\longrightarrow a} \frac{\psi(t)}{\phi(t)} = 0.$$

Ejemplo 1.22. En el Ejemplo 1.4 vimos que el polinomio

$$p_9(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!},$$

aproxima a la función $\text{sen}(x)$ en $[-\pi/2, \pi/2]$ con error $R_9(x)$ donde[5]

$$|R_9(x)| \leq \frac{|x|^{10}}{10!}.$$

[4]Si $a \in \mathbb{R}$, una vecindad de a es un intervalo abierto que contiene al punto a. Si $a = \infty$ entonces una vecindad de a es un intervalo de la forma (b, ∞). El caso $a = -\infty$ es similar.

[5]En el Ejemplo 1.4 obtuvimos una cota superior para $|R_9(x)|$ en $[-\pi/2, \pi/2]$. En el ejemplo de ahora mantenemos la dependencia en x de $R_9(x)$ ya que caracterizamos su comportamiento asintótico y no el máximo o peor de los casos.

Podemos escribir esto como

$$R_9(x) = O(|x|^{10}), \quad x \longrightarrow 0.$$

(La constante en la Definición (1.21) sería 1/10!.) Note que en este caso también podemos escribir que

$$R_9(x) = o(|x|^9), \quad x \longrightarrow 0.$$

□

La Definición (1.21) aplica a funciones. En ocasiones usaremos la notación O para describir el *orden de magnitud* de constantes. En particular diremos que una cierta constante es $O(\beta^p)$, donde $\beta > 0$ y $p \in \mathbb{R}$ son números dados, si existe una constante C con $1 \leq C < \beta$, tal que $|O(\beta^p)| \leq C\beta^p$. Por ejemplo, $4.23 \times 10^{-3} = O(10^{-3})$, $0.57 \times 10^5 = O(10^4)$, y $15.4 \times 10^4 = O(10^5)$.

1.5 Ejercicios

Ejercicio 1.1. Considere la función $f(x) = \exp(x)$ en el intervalo $[-1, 1]$.

a) Calcule las fórmulas de p_1, p_2, p_3, p_4, i.e., los primeros cuatro polinomios de Taylor de $f(x)$ alrededor de $a = 0$.

b) Trace en el mismo sistema de coordenadas los cuatro polinomios.

c) Halle un estimado gráfico del error $\max_{-1 \leq x \leq 1} |f(x) - p_n(x)|$ para $n = 1, 2, 3, 4$.

d) Compare los resultados de la parte (c) con los que predice la cota del Teorema de Taylor, i.e., usando la representación (1.6) del residuo.

e) ¿Será p_4 siempre una aproximación mejor que p_1?

Ejercicio 1.2. Escriba dos programas en MATLAB que implementen los algoritmos (1.13) y (1.14) para evaluar un polinomio que se describen en el texto. Usando las funciones `tic` y `toc` de MATLAB calcule el tiempo de corrida para la evaluación del polinomio

$$p(x) = 2 - 3x + 4x^2 - 5x^3 + 7x^4 + 2x^5,$$

en 1000 puntos del intervalo de $[-2, 2]$. **Nota**: Dependiendo de la rapidez del computador que utilice, podría necesitar mucho más de 1000 puntos antes de notar diferencias en los tiempos de corrida.

Ejercicio 1.3. Tomando $p_0(x) = f(a)$ y usando la recursión (1.4) con $n \geq 1$, verifique usando inducción matemática que $p_n(x)$ está dado por la fórmula (1.5).

Ejercicio 1.4. Para $n \geq 1$, sea $t(n)$ el tiempo de ejecución para evaluar el polinomio

$$p_n(x) = 1 - x + x^2 - x^3 + \cdots + (-1)^n x^n.$$

Dibuje en un mismo sistema de coordenadas las $t(n)$ que resultan al usar los algoritmos (1.13), (1.14), y (1.15) para evaluar el polinomio.

Ejercicio 1.5. Suponga que z y $a(1:n)$ son vectores o arreglos dados, y que $p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1}$. Haciendo uso eficiente de la subrutina `hornerV`, escriba segmentos de programas en MATLAB para calcular las expresiones $p(z)/p(-z)$, $p(z) + p(-z)$, $p'(z)$, $\int_0^1 p(x)dx$, $\int_{-z}^{z} p(x)dx$.

Ejercicio 1.6. En este ejercicio consideramos la evaluación de monomios y de polinomios escasos, i.e., de pocos términos comparados con el grado del polinomio.

a) Suponga que $n = 2^k$ y considere el monomio x^n. Evaluamos el monomio organizando las multiplicaciones como sigue (caso $n = 8$):

$$a = x^2 \ , \ b = a^2 \ , \ c = b^2.$$

Describa e implemente un algoritmo para la evaluación de un monomio de grado n que requiere $O(\log n)$ multiplicaciones.

b) Usando el método de la parte (a), describa un método para evaluar polinomios escasos el cual supera al método de Horner cuando el número de términos del polinomio es mucho menor que el grado del polinomio.

c) Escriba los programas en MATLAB que implementen el método de la parte (b) y los algoritmos (1.13) y (1.15) para evaluar un polinomio. Usando las funciones `tic` y `toc` de MATLAB calcule los tiempos de corrida de los tres métodos para la evaluación del polinomio $q_n(x) = x^n - 1$ en 1000 puntos del intervalo de $[-2, 2]$ y para los valores de $n = 1, 2, \ldots, 50$. Trace los tiempos de corrida de los métodos contra n en el mismo sistema de coordenadas y explique las diferencias entre las gráficas.

Ejercicio 1.7. Una cierta función f tiene que:

$$f(1) = 2, \quad f'(1) = -1, \quad f''(1) = \frac{2}{3}, \quad f'''(1) = -3,$$

$$\left|f^{(4)}(x)\right| \leq 1.75, \quad x \in [0, 2].$$

a) Halle el polinomio de Taylor de f alrededor de $a = 1$ de grado a lo más tres.

b) Calcule una aproximación de $f(0.75)$.

c) Halle un estimado del error absoluto en la aproximación de $f(0.75)$.

Ejercicio 1.8. Considere la función $f(x) = \ln(x)$.

a) Halle el polinomio de Taylor de grado tres para $f(x)$ alrededor de $a = 1$.

b) Usando el polinomio en (a), aproxime $\ln(1.1)$.

c) Usando la fórmula (1.6) para el residual, halle un estimado del error en la aproximación de la parte (b).

Ejercicio 1.9. Evalúe el polinomio $p(x) = 6(x+3) + 9(x+3)^5 - 5(x+3)^8 - (x+3)^{11}$ lo más eficientemente posible.

Ejercicio 1.10. Considere el polinomio

$$\begin{aligned} p(x) &= -3 + 4(x-2) + 5(x-2)(x-1) + 7(x-2)(x-1)(x+3) \\ &\quad +3(x-2)(x-1)(x+3)(x-5). \end{aligned}$$

i) Escriba el polinomio de forma anidada de modo que se utilice la menor cantidad de multiplicaciones al evaluarlo. **Ojo:** No es necesario expandir ninguno de los productos que aparecen en $p(x)$.

ii) ¿Cúantas multiplicaciones y sumas o restas requiere evaluar el polinomio luego de escribirlo como en la parte (i)?

Ejercicio 1.11. Usando la función `exp(x)` de MATLAB, escriba un programa para evaluar en forma eficiente la expresión $y = 5e^{3x} + 7e^{2x} + 9e^x + 11$.

Ejercicio 1.12. Considere la función $f(x) = \sqrt{1+x}$.

a) Halle el polinomio de Taylor $p_3(x)$ de grado menor o igual a tres para la función $f(x)$ alrededor de $a = 0$.

b) Usando la fórmula (1.6) para el residual, halle un estimado del error $f(x) - p_3(x)$ para $x \in [0, 0.5]$.

c) Trace en un mismo sistema de coordenadas la función $f(x)$ y el polinomio $p_3(x)$.

Ejercicio 1.13. Usando la fórmula (1.6) para el residual, halle el entero n más pequeño para aproximar $f(x) = 1/x$ en $x = 1.25$ con una precisión de 10^{-8} usando un polinomio de Taylor de grado n alrededor de $a = 1$.

Ejercicio 1.14. Sea $F(x) = \int_0^x (1+t)^{-1} dt$. Usando un polinomio de Taylor de grado a lo más tres alrededor de $a = 0$ para $f(x) = (1+x)^{-1}$, aproxime $F(0.1)$. Usando la fórmula (1.6) para el residual, halle un estimado del error en dicha aproximación.

Ejercicio 1.15. Usando un polinomio de Taylor de grado n alrededor de $a = 0$ para $f(x) = e^x$ y su residuo, halle un polinomio y su residuo para e^{-t^2}. ¿Qué grado tiene este polinomio? Usando ahora el polinomio y residuo de e^{-t^2}, halle un polinomio y su residuo para $F(x) = (1/x) \int_0^x e^{-t^2} dt$.

Ejercicio 1.16. Suponga que f es una función continua en el intervalo $[a, b]$. Defina la suma $S = \sum_{i=1}^n f(x_i)$ donde los x_i's pertenecen a $[a, b]$. Demuestre que existe un número $\xi \in [a, b]$ tal que $S = nf(\xi)$. **Ayuda**: Demuestre que $m \leq S/n \leq M$ donde m, M son los valores mínimo y máximo de f en $[a, b]$. Argumente ahora usando el Teorema A.3 hasta obtener el resultado.

Ejercicio 1.17. El método de Horner para evaluar un polinomio se puede utilizar para evaluar la derivada del polinomio. Demuestre que para $n \geq 1$, si $b_1, b_2, \ldots, b_{n+1}$ son generados según el Método de Horner descrito en el texto, entonces

$$p(x) = (x - z)q(x) + b_1,$$

donde

$$q(x) = b_2 + b_3 x + \cdots + b_{n+1} x^{n-1}.$$

Verifique ahora que $p'(z) = q(z)$. Modifique la función `hornerV` dada en el texto para calcular la derivada del polinomio.

Ejercicio 1.18. Verifique que la sucesión $x_n = a^{2^n}$ donde $0 < a < 1$, converge a cero con orden dos.

Ejercicio 1.19. Verifique que el resultado del Teorema 1.2 se puede expresar como

$$R_n(x) = O(|x-a|^{n+1}), \quad x \longrightarrow a,$$

o como

$$R_n(x) = o(|x-a|^p), \quad x \longrightarrow a,$$

para cualquier $p < n+1$.

Ejercicio 1.20. Verifique que si f es una función diferenciable en x_0, i.e.,

$$\lim_{h \longrightarrow 0} \frac{f(x_0+h) - f(x_0)}{h} = f'(x_0),$$

entonces

$$f(x_0+h) = f(x_0) + hf'(x_0) + o(h), \quad h \longrightarrow 0.$$

Ejercicio 1.21. Usando el resultado del Ejercicio 1.20 verifique que la función $\phi(\cdot)$ del Teorema 1.9 se puede escribir como

$$\phi(x) = y_0 - \left[\frac{\frac{\partial f}{\partial x}(x_0, y_0)}{\frac{\partial f}{\partial y}(x_0, y_0)} \right] (x - x_0) + o(x - x_0), \quad x \longrightarrow x_0.$$

Ejercicio 1.22. Partiendo de la ecuación $f(x, \phi(x)) = 0$ y usando las definiciones (1.16a)–(1.16b), use inducción matemática para verificar (1.17).

Ejercicio 1.23. Encuentre o caracterice los valores de (x, y) para los cuales la ecuación:

$$y - \ln(x+y) = 0,$$

se puede resolver para y como función de x. Para estos valores de (x, y), calcule dy/dx.

Ejercicio 1.24. Usando el Teorema de la Función Implícita, verifique que la ecuación

$$x + y^3 - y = 0,$$

puede despejarse para y en términos de x cerca del punto $(0, -1)$. Halle el polinomio de Taylor de grado tres que aproxima a la función implícita cerca del punto $x = 0$.

Ejercicio 1.25. Modifique el Teorema 1.9 para el caso en que se busca despejar a x en términos de y. Modifique igualmente las ecuaciones (1.16)–(1.18) para este caso.

Ejercicio 1.26. Usando el Teorema de la Función Implícita, verifique que la ecuación

$$x^2 + xy - \ln(x + y^3) - 1 = 0,$$

puede despejarse para x en términos de y cerca del punto $(1, 0)$. Halle el polinomio de Taylor de grado tres que aproxima a la función implícita cerca del punto $y = 0$.

Ejercicio 1.27. Usando el Teorema del Valor Medio para integrales (Teorema A.2) verifique que existe $y \in (0, \pi/2)$ tal que

$$\int_0^{\pi/2} \mathrm{e}^x \cos x \, \mathrm{d}x = \mathrm{e}^y.$$

Ejercicio 1.28. Verifique que el número de multiplicaciones requeridas por el Algoritmo (1.13) para la evaluación de un polinomio de grado n es $O(n^2)$ según $n \longrightarrow \infty$. Verifique que los conteos correspondientes para los algoritmos (1.14) y (1.15) (método de Horner) son ambos O(n). ¿Cómo puede explicar que (1.14) y (1.15) sean ambos $O(n)$ cuando (1.15) es el método más eficiente?

Capítulo 2

Sistemas de Punto Flotante y Propagación de Errores

En este capítulo estudiaremos cómo se representan los números en la computadora. Por limitaciones físicas, los números solo se pueden representar con un número finito de cifras en la computadora. Al igual las operaciones aritméticas que se efectúan en la computadora no corresponden a las operaciones matemáticas exactas. Veremos pues como estas limitaciones en la representación de los números y las operaciones aritméticas afectan los cómputos y la propagación de errores.

2.1 Bases Numéricas y Cambios de Bases

Veamos primero cómo se representan los números en diferentes bases numéricas. Por ejemplo el número 231.12 base diez se interpreta como:

$$(231.12)_{10} = 2 \times 10^2 + 3 \times 10^1 + 1 \times 10^0 + 1 \times 10^{-1} + 2 \times 10^{-2}.$$

Otro ejemplo, cambiando de base dos a diez, sería:

$$\begin{aligned} (101.11)_2 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}, \\ &= 4 + 1 + 1/2 + 1/4 = (5.75)_{10}. \end{aligned}$$

Un ejemplo menos trivial es el de calcular la representación decimal de $(11\cdots1)_2$ donde hay n unos en la representación binaria. Note que

$$(\underbrace{11\cdots1}_{\text{n dígitos}})_2 = 2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 = (2^n - 1)_{10},$$

donde usamos que

$$1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1}, \quad x \neq 1.$$

Hemos visto aquí como convertir un número binario a su decimal correspondiente. Veamos ahora como convertimos de decimal a binario. Si

$$x = (b_n b_{n-1} \cdots b_2 b_1 b_0 . b_{-1} b_{-2} \cdots b_{-k+1} b_{-k} \cdots)_2,$$

es la representación binaria del número x, escribimos $x = x_e + x_f$ donde,

$$x_e = (b_n b_{n-1} \cdots b_2 b_1 b_0)_2, \quad x_f = (0.b_{-1} b_{-2} \cdots b_{-k+1} b_{-k} \cdots)_2.$$

Tenemos ahora para x_e que

$$\begin{aligned} x_e &= b_n \times 2^n + b_{n-1} \times 2^{n-1} + \cdots + b_2 \times 2^2 + b_1 \times 2 + b_0, \\ &= (b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_2 \times 2 + b_1) \times 2 + b_0, \\ &= ((b_n \times 2^{n-2} + b_{n-1} \times 2^{n-3} + \cdots + b_2) \times 2 + b_1) \times 2 + b_0, \\ &\vdots \end{aligned}$$

etc.. De aquí podemos ver que los dígitos de x_e base dos se obtienen de los residuos al dividir por dos a x_e y los cocientes sucesivos, hasta obtener un cociente de cero. Para x_f note que como

$$x_f = b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-k+1} \times 2^{-k+1} + b_{-k} \times 2^{-k} + \cdots,$$

entonces,

$$\begin{aligned} 2x_f &= b_{-1} + b_{-2} \times 2^{-1} + \cdots + b_{-k+1} \times 2^{-k+2} + b_{-k} \times 2^{-k+1} + \cdots, \\ &\equiv b_{-1} + q_{-1}, \\ 2q_{-1} &= b_{-2} + \cdots + b_{-k+1} \times 2^{-k+3} + b_{-k} \times 2^{-k+2} + \cdots, \\ &\equiv b_{-2} + q_{-2}, \\ &\vdots \end{aligned}$$

es decir, los dígitos de x_f base dos se obtienen multiplicando por dos y repitiendo el proceso con la parte fraccionaria del resultado. Esto se repite hasta que la parte fraccionaria se haga cero (representación binaria finita) o el proceso continua indefinidamente siendo el caso de una representación decimal infinita. Este proceso y el descrito arriba para x_e se combinan para obtener la representación binaria de x.

Ejemplo 2.1. Mostramos aquí el procedimiento convirtiendo $(217.732)_{10}$ a binario. Trabajamos primero con la parte entera del número. Dividimos esta sucesivamente entre dos y llevamos registro de los residuos sucesivos hasta que el cociente sea cero. Estos residuos, desde el último hasta el primero, forman la representación binaria de la parte entera. Veamos,

divisor	dividendo	cociente	residuo
2	217	108	1
2	108	54	0
2	54	27	0
2	27	13	1
2	13	6	1
2	6	3	0
2	3	1	1
2	1	0	1

Así que $(217)_{10} = (11011001)_2$. Para la parte fraccionar, multiplicamos sucesivamente por dos y llevamos cuenta cada vez que el resultado sea mayor de uno o no. Veamos,

2(0.732)=1.464	0.464	1
2(0.464)=0.928	0.928	0
2(0.928)=1.856	0.856	1
2(0.856)=1.712	0.712	1
2(0.712)=1.424	0.424	1
$\vdots$	$\vdots$	$\vdots$

En este caso el proceso sigue indefinidamente y escribimos

$$(0.732)_{10} = (0.10111\ldots)_2.$$

Combinando los dos cálculos tenemos que

$$(217.732)_{10} = (11011001.10111\ldots)_2.$$

Note que en este ejemplo, un número con representación decimal finita tiene representación binaria infinita (repetitiva). □

En los números *hexadecimales* o base 16, los dígitos son

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}.$$

Aquí $(A)_{16} = (10)_{10}$, $(B)_{16} = (11)_{10}$, etc.. Por ejemplo,

$$(CD.B9)_{16} = (12 \times 16^1 + 13 \times 16^0 + 11 \times 16^{-1} + 9 \times 16^{-2})_{10}.$$

Los métodos para cambiar de binario a hexadecimal y viceversa son extremadamente sencillos. Históricamente las bases binaria, hexadecimal, y octal han sido las más utilizadas para representar números en la computadora. Esto es básicamente porque el circuito eléctrico más elemental se puede representar por *uno* si hay corriente o *cero* si no hay corriente. De aquí que el sistema binario sea la forma natural de representar el estado del circuito. Esto combinado con lo relativamente fácil de pasar de una base a otra, hace que las bases binaria, octal, y hexadecimal hayan sido utilizadas eficientemente en el diseño computadoras.

2.2 Números de Punto Flotante

Sea x un número real base diez o dos. Podemos representar a x en forma única como sigue:

$$x = \sigma \cdot \bar{x} \cdot \beta^e, \tag{2.1}$$

donde $\sigma \in \{-1, 1\}$, e es un entero positivo o negativo, $(0.1)_\beta \leq \bar{x} < 1$, y β es la base. Esta forma de escribir el numero real x es lo que se conoce comúnmente como la *notación científica*. Por ejemplo:

$$(-117.32)_{10} = -(0.11732)_{10} \times 10^3, \quad (1101.11)_2 = (0.110111)_2 \times 2^4.$$

Un numero en notación científica puede tener un numero infinito de dígitos. Por ejemplo:

$$\left(\frac{1}{3}\right)_{10} = (0.333333\ldots)_{10} \times 10^0.$$

En general, no es posible almacenar un numero infinito de dígitos en una computadora digital. Esto nos lleva a considerar o estudiar los *sistemas de punto flotante*.

La representación de punto flotante del número x se denota por $\mathrm{fl}(x)$ y consiste de (2.1) con alguna restricción en el número de cifras en $\bar{x}$ y el tamaño de e. Tenemos pues que

$$\mathrm{fl}(x) = \sigma \cdot (0.a_1a_2\cdots a_t)_\beta \cdot \beta^e, \quad -N \leq e \leq M, \tag{2.2}$$

donde N, M son positivos y $1 \leq a_1 \leq \beta - 1$, $0 \leq a_i \leq \beta - 1$, $2 \leq i \leq t$. Los a_i's se obtienen de alguna forma a partir de $\bar{x}$. El número $(0.a_1a_2 \cdots a_t)_\beta$ se conoce como la *mantisa* y t el *número de cifras o largo de mantisa*. Si alguna operación aritmética produce un resultado con $e > M$ decimos que hubo una condición de *sobre–flujo* (*overflow*). Por el contrario si en el resultado, $e < -N$ decimos que hubo una condición de *sub–flujo* (*underflow*).

Ejemplo 2.2. Algunos ejemplos para las constantes (β, t, N, M) son:

Computadora	β	t	$-N$	M
VAX	2	24	-128	127
CRAY-1	2	48	-16384	16383
IEEE (presición simple)	2	24	-126	127
IEEE (presición doble)	2	53	-1022	1023

□

Se puede verificar que t cifras en la mantisa equivalen aproximadamente a t cifras significativas en la misma base. Además t dígitos binarios equivalen aproximadamente a $t \log_{10}(2)$ dígitos base diez.

Veamos ahora varias cantidades importantes del sistema (2.2):

- el número más grande en el sistema es:

$$(0.(\beta-1)(\beta-1)\cdots(\beta-1))_\beta \cdot \beta^M = (1 - \beta^{-t})\beta^M.$$

- El número más pequeño es:

$$(0.100\cdots0)_\beta \cdot \beta^{-N} = \beta^{-N-1}.$$

- El número positivo u más pequeño tal que $1+u \neq 1$, llamado el *épsilon de la máquina* es β^{1-t}. (¿por qué?)

La dos formas más comunes de calcular la mantisa son:

M1. *Truncación*: si $\bar{x} = (0.\delta_1\delta_2\cdots)_\beta$, entonces $a_i = \delta_i$, $1 \leq i \leq t$. Esto es, truncamos $\bar{x}$ a t dígitos.

M2. *Redondeo*: en el caso $\beta = 10$, tenemos

$$(0.a_1a_2\cdots a_t)_{10} = \begin{cases} (0.\delta_1\delta_2\cdots\delta_t)_{10}, & \delta_{t+1} < 5, \\ (0.\delta_1\delta_2\cdots\delta_t)_{10} + (0.\underbrace{00\cdots1}_{\text{t dígitos}})_{10}, & \delta_{t+1} \geq 5, \end{cases}$$

con las reglas correspondientes para base dos.

Ejemplo 2.3. Para una computadora con $\beta = 10$ y $t = 4$ tenemos que si $x = 10.2576$, entonces $\text{fl}(x) = 0.1025 \times 10^2$ para truncación y $\text{fl}(x) = 0.1026 \times 10^2$ usando redondeo. □

Ejemplo 2.4. Tomando $(\beta, t, N, M) = (10, 2, 1, 2)$, tenemos 90 posibles mantisas, y 4 exponentes, i.e., $-1, 0, 1, 2$. Como hay dos posibles signos, tenemos un total de $2(90)(4) + 1 = 721$ números en el sistema. Note que el sistema de punto flotante es finito. Estos números no se distribuyen en forma uniforme en la recta real. (Ver Figura 2.1). De hecho, se distribuyen en grupos de números donde los números en cada grupo difieren por $\beta^{e-t} = 10^{e-2}$ donde e asume los valores permitidos de los exponentes. En este sistema, el número positivo más pequeño es $0.10 \times 10^{-1} = 0.01$. El número positivo más grande es $0.99 \times 10^2 = 99$. □

El siguiente resultado nos provee de un estimado del error al utilizar las reglas (M1) o (M2) para calcular la mantisa:

Teorema 2.5. *Si $x \neq 0$, entonces $fl(x) = (1 + \epsilon)x$ donde*

$$|\epsilon| \leq \begin{cases} \beta^{1-t}, & \textit{en truncación,} \\ \frac{1}{2}\beta^{1-t}, & \textit{en redondeo.} \end{cases}$$

Demostración: Veamos la demostración de este resultado para el caso de truncación. En este caso sabemos que si $x = \sigma(0.a_1a_2\ldots)_\beta \times \beta^e$, entonces $\text{fl}(x) = \sigma(0.a_1a_2\ldots a_t)_\beta \times \beta^e$. De modo que

$$|x - \text{fl}(x)| = (0.a_{t+1}a_{t+2}\ldots)_\beta \times \beta^{e-t}.$$

Como $(0.a_1a_2\ldots)_\beta$ esta entre $(0.1)_\beta = \beta^{-1}$ y uno, tenemos que $|x| \geq \beta^{e-1}$. Así que

$$\frac{|x - \text{fl}(x)|}{|x|} \leq \frac{(0.a_{t+1}a_{t+2}\ldots)_\beta \times \beta^{e-t}}{\beta^{e-1}} \leq \frac{\beta^{e-t}}{\beta^{e-1}} = \beta^{1-t}.$$

□

Note que el error en redondeo es en general la mitad del error en truncación. También se puede verificar que como los errores en redondeo varían en signo, estadísticamente *en promedio* tienden a cancelarse. Esta cancelación no es tan probable con truncación. A pesar de estas ventajas de redondeo sobre truncación, la truncación es usada comúnmente en el diseño de computadoras por lo fácil de su implementación usando circuitos eléctricos.

2.3 Estándar de Aritmética de Punto Flotante de la IEEE

La gran mayoría de las computadoras construidas en los últimos quince años utilizaron lo que se conoce como el *estándar de aritmética de punto flotante de la IEEE*. Esto uniformizó los procesos de representación de números en computadoras distintas facilitando así las comparaciones y análisis de errores. Además, aprovechando que el primer dígito en (2.2) base dos es siempre uno, se puede ganar un dígito binario de precisión al tomar esto en consideración. En este estándar $\beta = 2$ y el flotante de un número tiene la forma:

$$\mathrm{fl}(x) = \sigma \cdot (1 + f) \cdot 2^e, \quad -N \leq e \leq M, \tag{2.3}$$

donde f es ahora la mantisa, satisface $0 \leq f < 1$, y se representa con t cifras binarias. Los valores usados para t y e son como sigue:

- en precisión sencilla, $\mathrm{fl}(x)$ se almacena en una palabra de 32 bits con un bit para σ, ocho para e, $t = 23$, $N = 126$, y $M = 127$;
- en presición doble, $\mathrm{fl}(x)$ se almacena en una palabra de 64 bits con un bit para σ, once para e, $t = 52$, $N = 1022$, y $M = 1023$.

La precisión sencilla economiza memoria pero no es mucho más rápida que la precisión doble en las computadoras modernas.

La distribución de los números de punto flotante en la recta numérica no es uniforme. De hecho para un exponente e dado, los números de punto flotante entre 2^e y 2^{e+1} están distribuidos de forma uniforme con incremento 2^{e-t}. Ilustramos aquí esto para el estándar de la IEEE con un sistema trivial donde f y e tienen dos y tres dígitos binarios respectivamente. Como f tiene dos dígitos binarios, tenemos $(.00)_2$, $(.01)_2$, $(.10)_2$, y $(0.11)_2$ como todas las posibles mantisas. Para aprovechar los tres dígitos del exponente e, se almacena su *complemento binario*, esto es, $e + 3$ en este caso[1]. De esta manera podemos almacenar exponentes enteros entre -3 y 4. Si reservamos los exponentes $-3, 4$ para las situaciones de sub–flujo y sobre–flujo respectivamente, tenemos que los números positivos de punto flotante en este sistema los podemos enumerar en la siguiente tabla:

[1]Si el sistema tiene ebits dígitos binarios para el exponente, entonces el complemento binario de e es $e + 2^{\mathrm{ebits}-1} - 1$.

$1+f$ / e	-2	-1	0	1	2	3
1.00	1/4	1/2	1	2	4	8
$(1.01)_2 = 1.25$	5/16	5/8	5/4	5/2	5	10
$(1.10)_2 = 1.50$	3/8	3/4	3/2	3	6	12
$(1.11)_2 = 1.75$	7/16	7/8	7/4	7/2	7	14

Al ordenar estos números en forma ascendente obtenemos:

$$\underbrace{\tfrac{1}{4}, \tfrac{5}{16}, \tfrac{3}{8}, \tfrac{7}{16}, \tfrac{1}{2}}_{\text{incremento } \frac{1}{16}}, \underbrace{\tfrac{5}{8}, \tfrac{3}{4}, \tfrac{7}{8}, 1}_{\text{incremento } \frac{1}{8}}, \underbrace{\tfrac{5}{4}, \tfrac{3}{2}, \tfrac{7}{4}, 2}_{\text{incremento } \frac{1}{4}}, \underbrace{\tfrac{5}{2}, 3, \tfrac{7}{2}, 4}_{\text{incremento } \frac{1}{2}}, \underbrace{5, 6, 7, 8}_{\text{incremento 1}}, \underbrace{10, 12, 14}_{\text{incremento 2}}.$$

Podemos observar aquí que la densidad de los números de punto flotante disminuye con su tamaño. (Vea la Figura 2.1.)

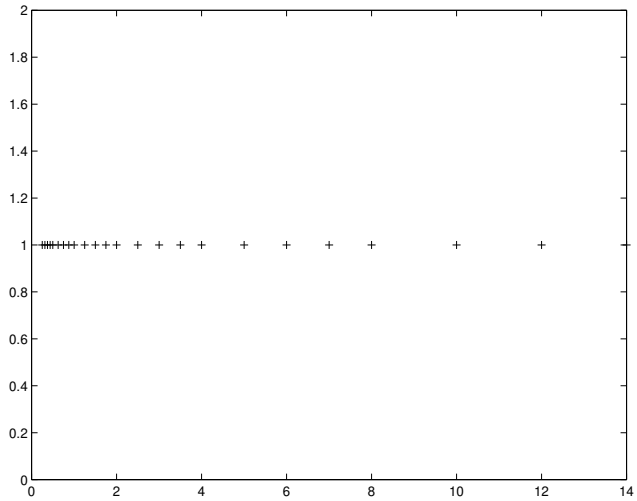


Figura 2.1: Estándar de la IEEE donde f y e tienen dos y tres dígitos binarios respectivamente.

El número positivo más grande en el estándar de la IEEE esta dado por:

$$1.\underbrace{11\cdots1}_{\text{t dígitos}} \times 2^M = (2 - 2^{-t})2^M.$$

Este número se denota en MATLAB por `realmax`. Cualquier resultado numérico que resulte en un número mayor que `realmax` se conoce como un

sobre–flujo, se denota por `Inf` (infinito), y se representa en (2.3) tomando $f = 0$ y $e = 1024$ en precisión doble. El número `Inf` satisface las relaciones `1/Inf=0` y `Inf+Inf=Inf`.

El número positivo más pequeño correspondiente es:

$$1.\underbrace{00\cdots0}_{\text{t dígitos}}\times 2^{-N} = 2^{-N},$$

y se denota en MATLAB por `realmin`. Cualquier resultado numérico que resulte en un número menor que `realmin` se conoce como un *sub–flujo* y en muchas computadoras el resultado se iguala a cero. Pero esto no es estándar. Cálculos como `0/0`, `Inf-Inf` que tradicionalmente se consideran indefinidos, se denotan por `NaN` ("not a number").

El épsilon de la máquína está dado en este sistema por 2^{-t} y se denota en MATLAB por `eps`. Este número corresponde al incremento de los números entre 1 y 2 en el sistema de punto flotante. `eps` también corresponde al error relativo (ver definición más adelante) mínimo posible en cualquier computo numérico en el sistema de punto flotante. Toda operación aritmética en la computadora o la representación de punto flotante de un número induce un error relativo no mayor de `eps`.

En la Tabla 2.1 ilustramos éstas cantidades especiales con sus respectivas representaciones binarias en la computadora en precisión doble. Note que los exponentes se almacenan usando el complemento binario que en este caso es $e + 2^{10} - 1 = e + 1023$.

Para más detalles sobre el estándar de la IEEE y los temas discutidos en esta sección se puede consultar, por ejemplo, [16] y [22].

2.4 Propagación de Errores

Como vimos antes, el uso de un número finito de cifras en la representación de punto flotante introduce errores de representación. Aún cuando los números en cuestión se puedan representar en forma exacta en la computadora, éstos podrían contener errores, por ejemplo si son datos experimentales. Queremos ver como estos errores en los datos se propagan al usar las operaciones aritméticas usuales o más general al evaluar funciones. Como las operaciones aritméticas de la computadora no corresponden a las operaciones exactas, también estudiaremos los efectos de éstas en los cómputos numéricos.

Número	Signo	Exponente	Mantisa
`realmax`	0	111 1111 1110 (si ponemos 1 al final corresponde a sobre–flujo)	1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
`realmin`	0	000 0000 0001 (exponente más pequeño antes de sub–flujo)	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
`-realmin`	1	000 0000 0001	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
`eps`	0	011 1100 1011 $(-52+1023)$	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0	0	000 0000 0000 (exponente de sub–flujo)	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1	0	011 1111 1111 $(0+1023)$	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
`realmin/2`	0	000 0000 0000 (situación de sub–flujo)	1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Tabla 2.1: Algunas cantidades especiales en el sistema IEEE en presición doble con sus representaciones binarias.

Denotamos por x_r el valor real o exacto de una cierta cantidad, y escribimos x_a para representar una aproximación de x_r, obtenida ya sea experimentalmente o al truncar x_r al momento de representarlo en la computadora. Definimos el *error absoluto* y el *error relativo* en x_a como aproximación de x_r por:

$$\text{Error}(x_a) = x_r - x_a, \quad \text{Rel}(x_a) = \frac{\text{Error}(x_a)}{x_r}, \quad x_r \neq 0,$$

respectivamente. Decimos que x_a tiene m *cifras significativas* como aproximación de $x_r = \sigma \cdot (0.\delta_1\delta_2\cdots)_{10} \times 10^e$ si

$$|x_r - x_a| = (0.00\cdots0\, a_{m+1}\, a_{m+2}\cdots)_{10} \times 10^e, \quad a_{m+1} < 5.$$

Se puede demostrar que si $|\text{Rel}(x_a)| \leq 5 \cdot 10^{-m-1}$, entonces x_a tiene por lo menos m cifras significativas como aproximación de x_r.

Ejemplo 2.6. Tomamos x_r como el valor calculado por MATLAB de exp(1). El error absoluto y relativo en $x_a = 2.718$ como aproximación de x_r, lo podemos calcular con el siguiente programa:

```
xr=exp(1);
xa=2.718;
error=xr-xa
rel=error/xr
```

lo cual produce los resultados `error`=2.8183e-004 y `rel`=1.0368e-004. De acuerdo al resultado mencionado en el párrafo anterior, nuestra aproximación tiene al menos tres cifras correctas. Note que este resultado nos da una cota inferior para el número de cifras correctas. De hecho nuestra aproximación en este ejemplo tiene cuatro cifras correctas, lo que todavía concuerda con el resultado general. □

Suponga que nos interesa evaluar una cierta función F en x_r. Denotamos por F_a la versión de F en la computadora y con x_a una aproximación de x_r. (¡En esta discusión, F no tiene que ser una función real de valor real!) Nos interesa calcular $F(x_r)$ pero terminamos calculando en la computadora $F_a(x_a)$. El error absoluto en $F_a(x_a)$ se puede escribir como

$$F(x_r) - F_a(x_a) = [F(x_r) - F(x_a)] + [F(x_a) - F_a(x_a)]. \tag{2.4}$$

El término $F(x_r) - F(x_a)$ del error se debe al error inicial en el dato x_a como aproximación a x_r y se conoce como el *error propagado*[2]. Este error es el que en ocasiones induce el llamado fenómeno de *pérdida de cifras significativas*. El término $F(x_a) - F_a(x_a)$ mide el error debido a la aritmética finita de la computadora[3]. Dependiendo de la complejidad de la función F, éste error es proporcional al épsilon de la máquina. Veamos ahora algunos ejemplos de estos tipos de errores.

2.4.1 Pérdida de cifras Significativas

Uno de los ejemplos más comunes de la propagación de errores o error propagado es en el fenómeno conocido como *perdida de cifras significativas*. Este tipo de situación se da cuando algún cálculo numérico involucra la resta de dos cantidades similares o en el cálculo de un número por medio de una sumatoria donde el resultado es mucho menor que los términos de la sumatoria, los cuales a su vez alternan en signo. Veamos algunos ejemplos.

Ejemplo 2.7. Considere el problema de buscar las raíces de la ecuación cuadrática:

$$x^2 - 30x + 0.0625 = 0.$$

[2]Note que aquí las operaciones aritméticas necesarias para calcular F son exactas. Así que sólo examinamos como el error en x_a se propaga al evaluar la función F.

[3]En este término se examinan los efectos de la aritmética de la computadora, la cual en general no es exacta, en el computo de F. Como el argumento en ambos términos es x_a, pensamos como si el dato x_a no contiene error alguno.

Usando la formula cuadrática tenemos que las raíces de ésta ecuación están dadas por:

$$x_{\pm} = \frac{30 \pm \sqrt{30^2 - 0.25}}{2} = \frac{30 \pm \sqrt{899.75}}{2}.$$

Suponga que tenemos que $\sqrt{899.75} = 29.996$ correcto a cinco cifras. Entonces podemos calcular

$$x_+ = \frac{30 + 29.996}{2} = 29.998, \quad x_- = \frac{30 - 29.996}{2} = 0.002.$$

El resultado del cálculo de la raíz x_+ tiene cinco cifras correctas, pero el de x_- tiene solo una cifra correcta. ¿Cuál fué el problema? Al restar cantidades similares ocurre el fenómeno de *perdida de cifras significativas*[4]. Para remediar este problema, usando todavía el valor aproximado de $\sqrt{899.75} = 29.996$ correcto a cinco cifras, rescribimos la expresión para x_- racionalizando, y re–calculamos:

$$\begin{aligned} x_- &= \left(\frac{30 - \sqrt{899.75}}{2}\right)\left(\frac{30 + \sqrt{899.75}}{30 + \sqrt{899.75}}\right) \\ &= \frac{30^2 - 899.75}{2(30 + \sqrt{899.75})} = \frac{0.25}{2(30 + 29.996)} = 0.0020835. \end{aligned}$$

Este nuevo cálculo para x_- tiene cinco cifras correctas, y solo utilizamos la aproximación $\sqrt{899.75} = 29.996$ correcta a cinco cifras.

En general tendremos el mismo problema al resolver la ecuación $x^2 + bx + c/4 = 0$ donde $b > 0$ y c es mucho más pequeño que b. Como el discriminante $\sqrt{b^2 - c} \approx b$, vamos a tener cancelación de cifras al calcular la raíz con el "+" en la fórmula cuadrática. Aquí nuevamente una racionalización de la fórmula problemática resuelve el problema. Si $b < 0$, la raíz problemática será la de la resta en la fórmula cuadrática. □

Ejemplo 2.8. Considere el problema de evaluar

$$f(x) = \frac{x - \operatorname{sen}(x)}{x^3},$$

para x cerca de cero. Como $\operatorname{sen}(x) \approx x$ para x cerca de cero, tenemos resta de cantidades similares en el numerador y por consiguiente habrá perdida

[4]En la Sección 2.4.3 veremos la explicación matemática de porque ocurre éste fenómeno.

de cifras significativas. Ilustramos aquí el uso de polinomios de Taylor para obtener representaciones o aproximaciones de la función que no sean susceptibles a perdida de cifras significativas. Usando el Teorema de Taylor tenemos que:

$$\text{sen}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!}\cos(\xi),$$

donde ξ está entre 0 y x. Tenemos ahora que

$$f(x) = \frac{1}{6} - \frac{x^2}{5!} + \frac{x^4}{7!} - \frac{x^6}{9!}\cos(\xi).$$

Note que si $|x| \leq 0.1$, entonces

$$\left|\frac{x^6}{9!}\cos(\xi)\right| \leq \frac{10^{-6}}{9!} \approx 2.76 \times 10^{-12}.$$

Podemos pues aproximar a $f(x)$ mediante

$$f(x) \approx \frac{1}{6} - \frac{x^2}{5!} + \frac{x^4}{7!}, \quad |x| \leq 0.1,$$

con un error absoluto del orden de 10^{-12}, y no hay perdida de cifras significativas al calcular con la fórmula aproximada para $|x| \leq 0.1$. Para $|x| > 0.1$ se utiliza la formula original de f. □

Ejemplo 2.9. Otro caso común de cancelación de cifras significativas ocurre en el cálculo de un número mediante una sumatoria donde los términos de la sumatoria son mucho mayores que el resultado y a la ves alternan en signo. Considere el problema de evaluar el polinomio $f(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$. El siguiente código en MATLAB evalúa el polinomio en el intervalo $[0.988, 1.012]$ y lo traza (vea la Figura 2.2):

```
x=linspace(0.988,1.012,100);
y=x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3-21*x.^2+7*x-1;
y1=(x-1).^7;
plot(x,y,'k-',x,y1,'k:')
legend('Forma expandida','Forma sin expandir')
```

Lo que debería ser una gráfica suave de un polinomio aparece altamente oscilatoria y de carácter aparentemente aleatorio. Esto se debe a la cancelación

severa de cifras significativas en el cálculo donde el resultado final es aproximadamente cero y se toman sumas y diferencias de números del tamaño de 35×1.012^4. El polinomio de este ejemplo corresponde a la forma expandida de $(x-1)^7$ donde trazamos cerca de $x = 1$. Note que si calculáramos con la fórmula sin expandir, el problema de cancelación de cifras no ocurriría. □

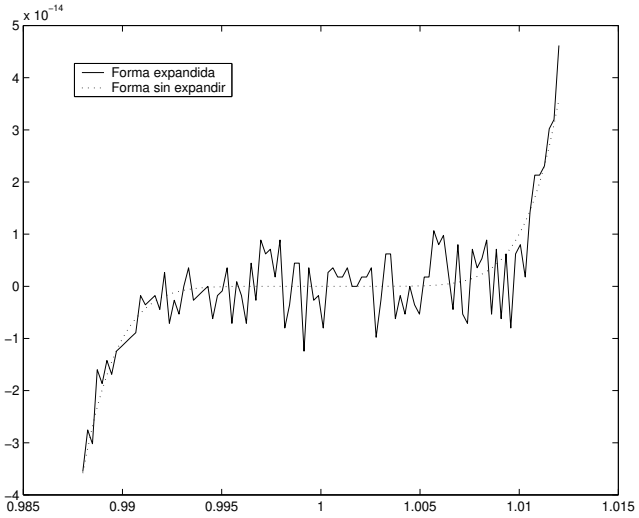


Figura 2.2: Gráficas de $y = (x-1)^7$ en forma expandida y sin expandir.

2.4.2 Evaluación de Funciones

En los ejemplos anteriores la perdida de cifras significativas se debe básicamente a los errores en los datos iniciales y no a las operaciones aritméticas de la computadora. Queremos estudiar esta situación en más detalles, esto es, suponiendo que las operaciones aritméticas son exactas, ¿en qué forma la función o fórmula que se use en los cálculos, propaga los errores en los datos iniciales? Estudiaremos ésto para funciones de una y dos variables.

Suponga que queremos evaluar una cierta función $f(x)$ en un argumento cuyo valor exacto es x_r. Suponga que x_a es una aproximación de x_r. ¿Cómo comparan $f(x_r)$ y $f(x_a)$? Note que implícitamente suponemos que f se puede calcular exactamente dado un argumento y lo que nos interesa es determinar como f propaga el error en x_a como aproximación de x_r. Usando el Teorema

del Valor Medio (Teorema A.1) podemos escribir que

$$f(x_r) - f(x_a) = f'(c)(x_r - x_a),$$

para algún c entre x_a y x_r. Dividiendo por $f(x_r)$ en ambos lados obtenemos la siguiente fórmula para los errores relativos:

$$\text{Rel}(f(x_a)) = \frac{f'(c)}{f(x_r)} x_r \, \text{Rel}(x_a).$$

Esta fómula sugiere que el número

$$K = \frac{f'(c)}{f(x_r)} x_r,$$

actúa como un factor de *magnificación* para el error relativo en x_a. Al factor K se le llama el *número de condición*. Si suponemos que el error absoluto en x_a como aproximación de x_r es pequeño, podemos aproximar c y x_r por x_a y tenemos que

$$\text{Rel}(f(x_a)) \approx \hat{K} \, \text{Rel}(x_a), \quad \hat{K} = \frac{f'(x_a)}{f(x_a)} x_a \approx K. \tag{2.5}$$

Ejemplo 2.10. Vamos a aproximar $\sqrt{81.1}$ con $\sqrt{81} = 9$. Tenemos entonces que $x_r = 81.1$, $x_a = 81$, y que $f(x) = \sqrt{x}$. ¿Cuán precisa es la aproximación de 9 con relación al valor exacto de $\sqrt{81.1}$? Usando la formula (2.5), y aproximando Rel(81) con 0.1/81, tenemos que

$$\text{Rel}(\sqrt{81}) \approx \frac{f'(81)}{f(81)} \, (81) \, \text{Rel}(81) \approx 6 \times 10^{-4},$$

por lo que podemos concluir que la aproximación $\sqrt{81.1} \approx 9$ tiene unas tres cifras correctas. □

Para una función f de dos variables (x, y), el Teorema del Valor Medio nos dice que si f tiene derivadas parciales continuas, entonces existen c y d tal que

$$f(x_r, y_r) - f(x_a, y_a) = \frac{\partial f}{\partial x}(c, d)(x_r - x_a) + \frac{\partial f}{\partial y}(c, d)(y_r - y_a). \tag{2.6}$$

Esta fórmula se puede usar para estimar el error al evaluar f en un argumento aproximado.

Ejemplo 2.11. Queremos cálcular la distancia focal f de un lente usando la fórmula

$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b},$$

donde[5] $a = 32 \pm 1\,\text{mm}$ y $b = 46 \pm 1\,\text{mm}$. aquí $x_a = 32$, $y_a = 46$ con errores absolutos de $\pm 1\,\text{mm}$. Tenemos pues que

$$\frac{1}{f} \approx \frac{1}{32} + \frac{1}{46} = 0.0529891.$$

Usando la función $g(x, y) = (1/x) + (1/y)$ en (2.6), este cálculo tiene un error absoluto de

$$\left|\frac{-1}{c^2}(\pm 1) + \frac{-1}{d^2}(\pm 1)\right| \leq \left|\frac{1}{31^2} + \frac{1}{45^2}\right| \leq 0.0015344.$$

Ahora obtenemos f tomando el recíproco del resultado de arriba para obtener que $f \approx 18.8718$ con un error absoluto no mayor de

$$\left|-\frac{1}{0.0529891^2}\,0.0015344\right| = 0.55,$$

i.e., $f = 18.9 \pm 0.6\,\text{mm}$. Note que aquí usamos la convención o notación de [29] de que las incertidumbres se dan solo a una cifra significativa y que la aproximación final solo incluye las cifras significativas consistentes con dicha incertidumbre. □

2.4.3 Propagación de Errores en las Funciones Aritméticas

Vamos a estudiar ahora los efectos de la propagación de errores en las funciones aritméticas de la computadora. En esta ocasión consideramos ambas fuentes de error: el uso de un número finito de cifras en los cálculos y como la función en si propaga los errores en los datos.

Las funciones aritméticas son funciones de dos variables de modo que denotamos por x_a y y_a aproximaciones de los valores exactos x_r y y_r. Denotamos por $\circ$ cualquiera de las operaciones $\{+, -, \times, \div\}$ y por $\circ^*$ la operación

[5] Al escribir digamos $a = 32 \pm 1\,\text{mm}$, el $\pm 1\,\text{mm}$ típicamente representa una *incertidumbre* experimental (cf. [29]). Esto es, el valor real a_r de a, *probablemente* pertenece al intervalo $(32 - 1, 32 + 1) = (31, 33)$, inclusive, podría no pertenecer a este intervalo.

aritmética correspondiente según se calcula en la computadora. Vamos a suponer que $\circ^*$ corresponde a la operación exacta $\circ$ pero truncada a la precisión de la máquina[6]. Esto es

$$a \circ^* b = \text{fl}(a \circ b). \tag{2.7}$$

Ejemplo 2.12. En el sistema de punto flotante $(\beta, t, N, M) = (10, 3, 9, 9)$ considere los números 12.3 y 5.27 los cuales tienen representación exacta en este sistema. El resultado exacto de la suma de estos dos números es 17.57. La representación de este resultado en el sistema de punto flotante usando redondeo es 0.176×10^2, lo cual se toma como el resultado de la suma en el sistema de punto flotante. Note que esta operación indujo un error, en este caso de 0.03. □

Queremos estimar el error

$$x_r \circ y_r - x_a \circ^* y_a.$$

Esto se puede escribir como

$$x_r \circ y_r - x_a \circ^* y_a = (x_r \circ y_r - x_a \circ y_a) + (x_a \circ y_a - x_a \circ^* y_a). \tag{2.8}$$

Los dos términos de la derecha de esta ecuación corresponden a el error propagado por la operación exacta $\circ$ debido a los errores en x_a y y_a, mientras que el segundo termino corresponde al error causado por el uso de la operación aproximada $\circ^*$. (Ver (2.4)). Usando el Teorema 2.5 y la hipótesis (2.7) tenemos que

$$x_a \circ^* y_a = \text{fl}(x_a \circ y_a) = (1 + \epsilon)(x_a \circ y_a),$$

donde ϵ esta acotado por el epsilon de la máquina. Tenemos pues que

$$\text{Rel}(x_a \circ^* y_a) = \frac{x_a \circ y_a - x_a \circ^* y_a}{x_a \circ y_a} = -\epsilon.$$

Es decir, el termino del error en (2.8) por la operación aritmética aproximada de la computadora, es del orden del epsilon de la máquina. No obstante, si la operación $\circ^*$ se repite muchas veces, estos errores pueden acumularse de forma que podrían en algunos casos llevar a resultados erróneos.

[6]En una computadora real, la operación $\circ^*$ corresponde al calculo de la operación aritmética correspondiente en precisión doble o cuádruple, y luego el resultado se redondea a precisión sencilla.

El termino del error propagado en (2.8) es en general el más significativo y su comportamiento depende de la operación aritmética en cuestión. Veamos primeramente el caso en que $\circ = \times$. Escribimos xy en lugar de $x \times y$. Sean

$$x_r = x_a + \delta, \quad y_r = y_a + \eta.$$

Entonces el error propagado se puede expresar como

$$x_r y_r - x_a y_a = (x_a + \delta)(y_a + \eta) - x_a y_a = \delta y_a + \eta x_a + \delta\eta.$$

Este error se podría estimar utilizando los estimados en δ, η y los valores de x_a, y_a. Podemos más aún calcular el error relativo en $x_a y_a$:

$$\begin{aligned} \mathrm{Rel}(x_a y_a) &= \frac{x_r y_r - x_a y_a}{x_r y_r} = \frac{x_r y_r - (x_r - \delta)(y_r - \eta)}{x_r y_r}, \\ &= \frac{\delta y_r + \eta x_r - \delta\eta}{x_r y_r} = \frac{\delta}{x_r} + \frac{\eta}{y_r} - \left(\frac{\delta}{x_r}\right)\left(\frac{\eta}{y_r}\right), \\ &= \mathrm{Rel}(x_a) + \mathrm{Rel}(y_a) - \mathrm{Rel}(x_a)\mathrm{Rel}(y_a). \end{aligned}$$

Si $|\mathrm{Rel}(x_a)|, |\mathrm{Rel}(y_a)|$ son ambos menor o igual que β^{-s}, donde $s \geq 1$ y $\beta \geq 2$, entonces $|\mathrm{Rel}(x_a y_a)| \leq 3\beta^{-s}$, i.e.,

$$\mathrm{Rel}(x_a y_a) = O(\beta^{-s}).$$

Decimos entonces que la multiplicación es una operación *estable*. En forma similar se demuestra que

$$\mathrm{Rel}(x_a / y_a) = O(\beta^{-s}),$$

obteniendo así que la división es también una operación estable. (Ver Ejercicio 2.8).

Para la suma y la resta la situación es distinta. Es fácil ver que

$$(x_r \pm y_r) - (x_a \pm y_a) = \delta \pm \eta,$$

que es pequeño dado que δ, η lo son. Para el error relativo tenemos que

$$\mathrm{Rel}(x_a \pm y_a) = \frac{\delta \pm \eta}{x_r \pm y_r},$$

el cual puede ser bien grande, aunque δ, η sean pequeños, si $x_r \approx \mp y_r$. Esto explica el fenómeno de perdida de cifras significativas que vimos anteriormente al restar cantidades similares.

2.4.4 Sumatorias

En esta sección estudiamos el problema de sumar muchos números en la computadora. Como cada suma introduce un error proporcional al épsilon de la máquina, queremos ver como estos errores se acumulan durante el proceso. El análisis que presentamos generaliza al problema del cálculo de productos interiores.

El problema que nos interesa es calcular la sumatoria

$$S = a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i,$$

usando aritmética de punto flotante. Con esto en mente, definimos las sumatorias parciales $\{S_j\}$ por:

$$\begin{cases} S_1 = a_1, \\ S_j = \mathrm{fl}(S_{j-1} + a_j), \qquad 2 \leq j \leq n. \end{cases}$$

Queremos determinar el error $S - S_n$. Motivados por el Teorema 2.5 definimos ϵ_j, $2 \leq j \leq n$ por

$$S_j = (S_{j-1} + a_j)(1 + \epsilon_j), \quad 2 \leq j \leq n.$$

Tenemos ahora:

Teorema 2.13. $S_n = a_1 \prod_{k=2}^{n}(1+\epsilon_k) + \sum_{j=2}^{n} a_j \prod_{k=j}^{n}(1+\epsilon_k), \quad n \geq 2.$

Demostración: Procedemos por inducción en el número de términos en la sumatoria.

1. *Paso básico*: aquí $S_2 = \mathrm{fl}(a_1 + a_2) = (a_1 + a_2)(1 + \epsilon_2) = a_1(1 + \epsilon_2) + a_2(1 + \epsilon_2)$ lo cual coincide con la fórmula del teorema.

2. *Paso inductivo*: Suponemos que el resultado es cierto para $n = m$, i.e.,

$$S_m = a_1 \prod_{k=2}^{m}(1+\epsilon_k) + \sum_{j=2}^{m} a_j \prod_{k=j}^{m}(1+\epsilon_k).$$

   Ahora verificamos que el resultado también es cierto para $n = m + 1$. Pero

$$S_{m+1} \;=\; \mathrm{fl}(S_m + a_{m+1}) = (S_m + a_{m+1})(1 + \epsilon_{m+1}),$$

$$
\begin{aligned}
&= a_{m+1}(1+\epsilon_{m+1}) \\
&\quad +(1+\epsilon_{m+1})\left[a_1\prod_{k=2}^{m}(1+\epsilon_k)+\sum_{j=2}^{m}a_j\prod_{k=j}^{m}(1+\epsilon_k)\right], \\
&= a_1\prod_{k=2}^{m+1}(1+\epsilon_k)+a_{m+1}(1+\epsilon_{m+1})+\sum_{j=2}^{m}a_j\prod_{k=j}^{m+1}(1+\epsilon_k), \\
&= a_1\prod_{k=2}^{m+1}(1+\epsilon_k)+\sum_{j=2}^{m+1}a_j\prod_{k=j}^{m+1}(1+\epsilon_k),
\end{aligned}
$$

lo cual es el resultado para $n = m+1$.

3. Por el Principio de Inducción Matemática, el resultado del teorema es cierto para toda $n \geq 2$.

□

Para analizar el resultado de este teorema, definimos $E_2, E_3, \ldots, E_n$ por

$$1 + E_j = \prod_{k=j}^{n}(1+\epsilon_k), \quad 2 \leq j \leq n.$$

Usando el Teorema 2.5 tenemos ahora que

$$(1-\epsilon)^{n-j+1} \leq 1 + E_j \leq (1+\epsilon)^{n-j+1},$$

donde $\epsilon = (1/2)\beta^{1-t}$ al usar redondeo. Esto implica que

$$|E_j| \leq (1+\epsilon)^{n-j+1} - 1, \quad 2 \leq j \leq n. \tag{2.9}$$

Usando la serie de Taylor de la función exponencial, se puede verificar ahora (Ejercicio 2.3) que si $n\epsilon \leq \frac{1}{2}$, entonces

$$(1+\epsilon)^p - 1 \leq 2p\epsilon,$$

para cualquier $p \leq n$. Usando este resultado en (2.9) obtenemos que

$$|E_j| \leq (n-j+1)\beta^{1-t}, \quad 2 \leq j \leq n. \tag{2.10}$$

Tenemos ahora que el resultado del Teorema 2.13 se puede escribir como:

$$S_n = a_1(1+E_2) + \sum_{j=2}^{n}a_j(1+E_j) = S + a_1E_2 + \sum_{j=2}^{n}a_jE_j.$$

De aquí obtenemos que

$$S - S_n = -a_1 E_2 - \sum_{j=2}^{n} a_j E_j.$$

Combinando esto con el estimado (2.10) tenemos que para minimizar el error en la sumatoria calculada, debemos ordenar los a_i's de modo que

$$|a_1| \leq |a_2| \leq \cdots \leq |a_n|,$$

i.e., debemos sumar los números más pequeños primero y luego los grandes. En particular tenemos el resultado curioso de que la suma de punto flotante *no es una operación conmutativa en la computadora*.

Ejemplo 2.14. En el sistema de punto flotante $(\beta, t, N, M) = (10, 3, 9, 9)$, calculamos la suma de los números 237, 52.6, y 0.726 de dos formas: $(237 +^* 52.6) +^* 0.726$ y también de la forma $(0.726 +^* 52.6) +^* 237$, donde $+^*$ se calcula usando truncación. Note que los tres números tienen representación exacta en el sistema de punto flotante. Pero

$$\begin{aligned} (237 +^* 52.6) +^* 0.726 &= 289 +^* 0.727 = 289, \\ (0.726 +^* 52.6) +^* 237 &= 53.3 +^* 237 = 290. \end{aligned}$$

El valor exacto de la suma de los tres números es 290.326 que en el sistema de punto flotante entraría como 290. Así que la suma donde los números se ordenan de menor a mayor, es mas correcta que la que suma con los números ordenados de mayor a menor. □

Un análisis similar (vea [11]), pero un tanto más complicado matemáticamente, se puede hacer para el producto interior

$$\sum_{k=1}^{n} a_k b_k.$$

Note que en el análisis anterior no se tomaron en consideración posibles errores en los datos a_i's. Estos efectos se pueden incluir también en el análisis. (Ver Ejercicio 2.7).

2.5 Ejercicios

Ejercicio 2.1. Suponga que $n = 2^k$ y que calculamos

$$S_n = \sum_{i=1}^{n} x_i,$$

en el orden siguiente (caso $n = 8$):

$$(((x_1 + x_2) + (x_3 + x_4)) + ((x_5 + x_6) + (x_7 + x_8))).$$

Obtenga un estimado del error de punto flotante al calcular esta sumatoria.

Ejercicio 2.2. El polinomio $p(x) = ax^2 + bx + c$ se evalúa en un sistema de punto flotante mediante la regla de Horner. Demuestre que el valor calculado $\hat{p}(x)$ satisface

$$|\hat{p}(x) - p(x)| \leq (4|ax^2| + 3|bx| + |c|)\mu,$$

donde μ es el epsilon de la máquina. (Los términos $O(\mu^2)$ se pueden descartar.)

Ejercicio 2.3. Usando la serie de Taylor de la función $f(x) = \mathrm{e}^x$, verifique que si $p\epsilon \leq \frac{1}{2}$, entonces

$$(1 + \epsilon)^p - 1 \leq 2p\epsilon.$$

Ejercicio 2.4. ¿Cúanta precisión se necesita en el número π para poder calcular $\sqrt{\pi}$ correcto a cuatro cifras?

Ejercicio 2.5. Trace la gráfica de la función $f(x) = x^3 - 3x^2 + 3x - 1$ en el intervalo $[1 - \delta, 1 + \delta]$ para $\delta = 0.01, 0.001, 0.0001$. Evalúe la función de dos formas:

normal: $-1 + 3x - 3x^2 + x^3$

anidada: $-1 + x(3 + x(-3 + x))$

Observe cualquier diferencia o características particulares en las gráficas. ¡Explique!.

Ejercicio 2.6. Considere la recurrencia $x_{k+1} = 2.25x_k - 0.5x_{k-1}$, $k = 2, 3, \ldots$ donde $x_1 = 1/3, x_2 = 1/12$. Verifique que la solución exacta de esta recurrencia es:

$$x_k = \frac{4^{1-k}}{3}, \quad k = 1, 2, 3, \ldots.$$

Resuelva esta recurrencia en la computadora. Haga una gráfica de $\log(x_k)$ como función de k. Explíque los resultados.

Ejercicio 2.7. Sean $a_i = a_i^A + \epsilon_i$, $b_i = b_i^A + \eta_i$, $i = 1, 2$ donde $|\epsilon_i|, |\eta_i| \leq 10^{-r}$. Defina

$$S_2 = \text{fl}(\text{fl}(a_1^A b_1^A) + \text{fl}(a_2^A b_2^A)).$$

Suponiendo que $\text{fl}(x) = (1 + \epsilon)x$, $|\epsilon| \leq 10^{-t}$ y que $r \leq t$, calcule el error

$$a_1 b_1 + a_2 b_2 - S_2.$$

Ejercicio 2.8. Demuestre que

$$\text{Rel}(x_a/y_a) = \frac{\text{Rel}(x_a) - \text{Rel}(y_a)}{1 - \text{Rel}(y_a)}.$$

Verifique ahora que si $|\text{Rel}(x_a)|, |\text{Rel}(y_a)|$ son ambos menor o igual que β^{-s} donde $s \geq 1$ y $\beta \geq 2$, entonces $|\text{Rel}(x_a/y_a)| \leq 4\beta^{-s}$, i.e.,

$$\text{Rel}(x_a/y_a) = O(\beta^{-s}), \quad \beta^{-s} \longrightarrow 0.$$

Ejercicio 2.9. La fórmula de Stirling

$$S_n = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n,$$

se usa para aproximar $n!$. Escriba un programa en MATLAB que para $1 \leq n \leq 13$ genere una tabla de los números de Stirling, $n!$, y los errores absolutos y relativos en la fórmula de Stirling como aproximación de $n!$.

Ejercicio 2.10. Considere el sistema de punto flotante (2.2) donde $\beta = 2$, $t = 24$, $N = M = 100$. ¿Cuál es el valor de n más grande para el cual $n!$ se puede representar en forma exacta en el sistema de punto flotante?

Ejercicio 2.11. En el sistema de punto flotante (2.2), halle el entero positivo U más grande para el cual todos los enteros positivos menor o igual a U tienen representación exacta en el sistema de punto flotante.

Ejercicio 2.12. En un sistema de punto flotante base diez con 8 cifras decimales, ¿cúal es el número de punto flotante en este sistema mayor de 183 pero a la vez más cerca de 183? ¿cúal es la distancia o separación entre los dos números?

Ejercicio 2.13. Explique porque en el sistema de punto flotante del estandar del IEEE con $(\beta, t, N, M) = (2, 53, 1022, 1023)$, la expresión $2(1-2^{-54})\times 2^{1023}$ produce un sobre–flujo (`Inf`), mientras que si calculamos $(2 - 2^{-53}) \times 2^{1023}$ obtenemos el valor de `realmax`. Haga los calculos en MATLAB o OCTAVE. Note que ambas expresiones son matemáticamente equivalentes.

Ejercicio 2.14. Para los números $x = 23.154545$, $y = 6.32976$, halle las representaciones de punto flotante $x_a = \text{fl}(x)$ y $y_a = \text{fl}(y)$ usando truncación en el sistema de punto flotante $(\beta, t, M, N) = (10, 3, -9, 9)$. Efectúe las operaciones $x_a +^* y_a$, $x_a \times^* y_a$ donde $+^*$ y $\times^*$ representan el resultado de truncar a la precisión del sistema de punto flotante, las operaciones aritméticas exactas correspondientes.

Ejercicio 2.15. Para $\alpha = 0.8717$ y $\beta = 0.8719$ calcule el punto medio del intervalo $[\alpha, \beta]$ mediante la fórmula $(\alpha+\beta)/2$ y usando redondeo y truncación con cuatro cifras decimales. Estime el error en cada cálculo. Calcule de igual forma pero usando una fórmula alterna para el punto medio y compare con los resultados anteriores.

Ejercicio 2.16. Halle una fórmula alterna para la expresión $-\ln(x-\sqrt{x^2-1})$ que no sea susceptible a la perdida de cifras significativas cuando x es grande y positiva. Modifique la fórmula alterna para que no sea susceptible a sobre–flujo al calcular x^2 cuando x es grande y positiva.

Ejercicio 2.17. Halle una expresión y el error correspondiente para aproximar la función $(\text{sen}(\theta)-\theta\cos(\theta))/\theta^3$ para θ pequeño la cual no sea susceptible a perdida de cifras significativas.

Ejercicio 2.18. Para la función $f(x) = \sqrt{x+1}-\sqrt{x}$, calcule $f(400)$ correcto a seis cifras utilizando únicamente las aproximaciones $\sqrt{400} = 20.0000$ y $\sqrt{401} = 20.0250$, ambas correctas a seis cifras.

Ejercicio 2.19. Calcule las raíces de las siguientes ecuaciones cuadráticas al número de cifras correctas de las aproximaciones dadas:

a) $x^2 + 62.10x + 1 = 0$ dado que $\sqrt{3852} \approx 62.06$ correcto a cuatro cifras. **Nota**: Las raíces de la ecuación, exactas a cuatro cifras, son -62.08 y -0.01611.

b) $2x^2 - 11x + 1/8 = 0$ dado que $\sqrt{60} = 7.7460$ o $\sqrt{120} = 10.954$ correcto a cinco cifras. **Nota**: Las raíces de la ecuación, exactas a cinco cifras, son 5.4886 y 0.011387.

c) $x^2 - 110x + 1 = 0$ dado que $\sqrt{12096} = 109.98$ correcto a cinco cifras. **Nota**: Las raíces de la ecuación, exactas a cinco cifras, son 109.99 y 9.0917×10^{-5}.

Ejercicio 2.20. Dado que $x_a = 2.653$ está correctamente redondeado al número de cifras mostradas, halle un estimado para el error $f(x_r) - f(x_a)$ y el error relatívo en $f(x_a) = e^{2.653}$.

Ejercicio 2.21. Dado que $\theta = 41 \pm 1°$, calcule $n = 1/\operatorname{sen}(\theta)$ y utilice la formula (2.5) para obtener un estimado del error relativo en el valor de n calculado. **Ayuda:** ¡Ojo con la variable θ que esta dada en grados!

Ejercicio 2.22. Dado que $x = 6.0 \pm 0.1$ y $y = 3.0 \pm 0.1$, calcule $q = xy + x^2/y$ y utilice la formula (2.6) para obtener un estimado del error absoluto en el valor de q calculado.

Ejercicio 2.23. Dado que $x = 10 \pm 2$, $y = 7 \pm 1$, y $\theta = 40 \pm 3°$, calcule

$$q = \frac{x + 2}{x + y\cos(4\theta)}.$$

Usando la formula (2.6), halle un estimado del error absoluto en el valor de q calculado.

Ejercicio 2.24. Sean a, b, c números con representación exacta en una computadora base diez con t dígitos en la mantisa y que usa truncación. Considere la expresión $\hat{E} = \text{fl}(a + \text{fl}(bc))$. Usando la expresión $\text{fl}(x) = (1 + \epsilon)x$, escriba $\hat{E}$ eliminando los fl. Usando este resultado, halle una cota para el error $|E - \hat{E}|$ donde $E = a + bc$. Haga lo mismo para las expresiones $S = (a + b)/c$ y $\hat{S} = \text{fl}(\text{fl}(a + b)/c)$.

Ejercicio 2.25. Los puntos $(x_0, y_0), (x_1, y_1)$ determinan una recta L en el plano. Sea $\hat{x}$ el intercepto en el eje de x de L. Podemos calcular $\hat{x}$ usando cualquiera de las dos siguientes fórmulas:

$$\hat{x} = \frac{x_0 y_1 - x_1 y_0}{y_1 - y_0}, \quad \hat{x} = x_0 - \frac{(x_1 - x_0) y_0}{y_1 - y_0} \quad .$$

Para los datos $(x_0, y_0) = (1.31, 3.24)$, $(x_1, y_1) = (1.93, 4.76)$ y usando aritmética de tres dígitos, calcule $\hat{x}$ de ambas formas y determine cual método es mejor.

Ejercicio 2.26. Explique qué calcula el siguiente programa en MATLAB:

```
u=1;
while 1+u~=1,
u=u/2;
end
u=2*u
```

La expresión `~=` equivale a *no es igual* en MATLAB.

Ejercicio 2.27. Enumere todos los números de punto flotante para el estándar de la IEEE (2.3) donde f y e tienen tres dígitos binarios ambos.

Capítulo 3

Sistemas de Ecuaciones Lineales

Los sistemas de ecuaciones lineales son una de las herramientas matemáticas de modelado más comunes en las aplicaciones. Una clasificación común de los sistemas lineales es por su tamaño. Los sistemas con cientos de variables se consideran pequeños y usualmente se utilizan los llamados *métodos directos* para su solución. Los sistemas con miles o más variables se consideran grandes o de gran escala y los métodos de solución más eficientes por lo general son los llamados *métodos iterativos* o *indirectos*. Otro criterio de clasificación importante de los sistemas lineales es por la cantidad o densidad de ceros de la matriz de coeficientes. Los sistemas con pocas entradas distintas de cero se llaman *escasos*. De lo contrario decimos que el sistema es *denso*. El aprovechar la estructura de ceros de la matriz de coeficientes nos lleva por lo general a algorítmos mucho más eficientes que los convencionales.

La solución directa de sistemas de ecuaciones lineales conlleva esencialmente dos etapas: transformación del sistema original a otro sistema equivalente más *simple* y luego la solución del nuevo sistema equivalente. La transformación del sistema original a uno más simple toma muchas formas la más común de ellas siendo el proceso de *eliminación gaussiana*. En este método, en su forma básica, si ninguno de los pivotes se hace cero, se producen como resultado matrices L y U triangulares inferior unitaria y superior respectivamente tal que $A = LU$ donde A es la matriz de coeficientes del sistema original. El sistema $Ax = b$ se puede resolver ahora en dos etapas adicionales.

3.1 Matrices, Valores Propios, y Normas

3.1.1 Sistemas lineales

Un sistema lineal de n ecuaciones en n desconocidas se puede escribir de la forma

$$\left\{\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &=& b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &=& b_2, \\ \vdots & \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &=& b_n, \end{array}\right. \tag{3.1}$$

donde los $a_{ij}, b_i \in \mathbb{R}$ son dados y $x_1, x_2, \ldots, x_n$ son las desconocidas. Si definimos la matriz A y los vectores b, x por

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \tag{3.2}$$

entonces podemos escribir el sistema (3.1) en forma matricial como $Ax = b$. La matriz A se conoce como la *matriz de coeficientes* del sistema y b como el *lado derecho*. Si $b = 0$, i.e., $b_i = 0$, $1 \leq i \leq n$, entonces decimos que el sistema es *homogéneo*. De lo contrario se dice que es *heterogéneo*.

Ejemplo 3.1. Considere el sistema

$$\left\{\begin{array}{rcl} 2x_1 + x_2 - x_3 &=& 1, \\ 3x_1 + x_2 + 2x_3 &=& 2, \\ x_1 + x_2 - 3x_3 &=& -1. \end{array}\right.$$

Si definimos

$$A = \begin{pmatrix} 2 & 1 & -1 \\ 3 & 1 & 2 \\ 1 & 1 & -3 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

entonces podemos escribir el sistema en forma matricial como $Ax = b$. □

El siguiente teorema nos da condiciones bajo las cuales (3.1) tiene solución y cuando esta es única. Para la demostración del mismo véase por ejemplo [13].

Teorema 3.2. *Sea A una matriz $n \times n$ y b un vector en $\mathbb{R}^n$. Las siguientes aseveraciones sobre el sistema $Ax = b$ son todas equivalentes:*

a) El sistema tiene solución que es única para todo lado derecho b.

b) El sistema tiene por lo menos una solución para todo lado derecho b.

c) El sistema homogéneo asociado tiene $x = 0$ como única solución.

d) $\det(A) \neq 0$.

e) A es invertible.

3.1.2 Valores y vectores propios

Dada una matriz A de tamaño $n \times n$ y entradas complejas, un número $\lambda \in \mathbb{C}$ (conjunto de los números complejos) es un *valor propio* de A si existe un vector $x \neq 0$ tal que $Ax = \lambda x$. (El vector x se llama *vector propio* correspondiente al valor propio λ.) Como $Ax = \lambda x$ es equivalente al sistema $(A-\lambda I)x = 0$, y como $x \neq 0$, tenemos por por la propiedad (d) en el Teorema 3.2 que

$$p_A(\lambda) \equiv \det(A - \lambda I) = 0, \tag{3.3}$$

es una condición necesaria y suficiente para que λ sea un valor propio de A. $p_A(\lambda)$ es un polinomio en λ y se llama el *polinomio propio o característico* de A. Vemos pues que el problema de hallar los valores propios de una matriz es equivalente al de hallar las raíces de un polinomio con coeficientes complejos. Por el Teorema Fundamental del Álgebra, tenemos ahora que toda matriz $n \times n$ tiene exactamente n valores propios contando multiplicidades. También tenemos por un resultado importante en la teoría de Galöis, que las raíces de un polinomio y por consiguiente los valores propios de una matriz, no se pueden calcular utilizando únicamente fórmulas algebraicas. Por tal razón el cálculo de los valores propios de matrices es un problema computacionalmente complejo y el poder estimar los mismos es de vital importancia.

Una matriz A es *simétrica* si $A^t = A$ donde $A^t = (a_{ji})$ si $A = (a_{ij})$. Decimos que A es *definida positiva* si

$$x^t Ax > 0, \quad \forall \ \ x \neq 0, \quad x \in \mathbb{R}^n. \tag{3.4}$$

Las *submatrices principales* de la matriz $A = (a_{ij})$, $n \times n$ son $A_1, A_2, \ldots, A_n$ donde $A_k = (a_{ij})$, $1 \leq i, j \leq k$. Note que A_k es de tamaño $k \times k$. Tenemos ahora:

Teorema 3.3. *Sea A una matriz simétrica $n \times n$. Entonces las siguientes son equivalentes:*

a) A es positiva definida.

b) Todos los valores propios de A son positivos.

c) $\det A_k > 0$, $k = 1, 2, \ldots, n$.

3.1.3 Normas

Para cualquier vector $x = (x_1, x_2, \ldots, x_n)^t$ y $1 \leq p \leq \infty$, definimos la *norma* p de x por:

$$\|x\|_p = \begin{cases} \left(\sum_{k=1}^{n} |x_k|^p \right)^{1/p}, & 1 \leq p < \infty, \\ \max\limits_{1 \leq k \leq n} |x_k|, & p = \infty. \end{cases} \tag{3.5}$$

Los casos más comunes de estas normas son para $p = 1, 2, \infty$. Note que $\|\cdot\|_2$ corresponde a la norma euclidiana o distancia al origen del punto terminal del vector. En la Figura 3.1 mostramos los *discos* unitarios en $\mathbb{R}^2$ para las normas $p = 1, 2, \infty$.

Las normas p tienen las siguientes propiedades (Ejercicio 3.1):

a) $\|x\|_p \geq 0$, y $\|x\|_p = 0$ si y solo si $x = 0$.

b) $\|\alpha x\|_p = |\alpha| \, \|x\|_p$ para cualquier $\alpha \in \mathbb{R}$.

c) $\|x + y\|_p \leq \|x\|_p + \|y\|_p$ para cualesquiera vectores x, y (desigualdad triangular).

d) $\lim_{p \longrightarrow \infty} \|x\|_p = \|x\|_\infty$.

Para definir normas matriciales, podemos considerar a una matriz $n \times n$ como un vector en $\mathbb{R}^{n^2}$ y entonces usar las normas p vectoriales. El caso $p = 2$ de este proceso se conoce como la *norma de Frobenius* y tiene la definición:

$$\|A\|_F = \left(\sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2}. \tag{3.6}$$

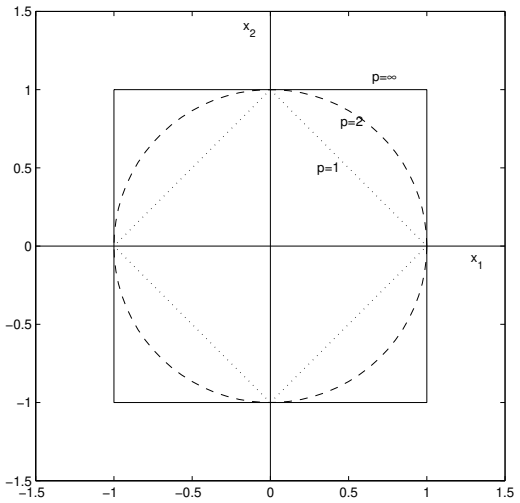


Figura 3.1: Discos unitarios en $\mathbb{R}^2$ para las normas $p = 1, 2, \infty$.

Aparte de la norma de Frobenius, las otras normas matriciales definidas considerando la matriz como un vector, no tienen todas las propiedades necesarias para que sean prácticas. La forma más útil de definir una norma matricial es con las llamadas *normas subordinadas o inducidas* por las normas vectoriales p que se definen por:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \tag{3.7}$$

Estas normas satisfacen las siguientes propiedades para cualquier vector x, escalar α, y matrices A, B ([2], [11]):

1. $\|A + B\|_p \leq \|A\|_p + \|B\|_p$.
2. $\|AB\|_p \leq \|A\|_p \|B\|_p$.
3. $\|Ax\|_p \leq \|A\|_p \|x\|_p$.

Para los casos $p = 1, 2, \infty$ el siguiente resultado nos da formas de calcular las correspondientes normas matriciales de forma más directa sin usar la definición. Pero antes mencionamos que el *radio espectral* de una matriz A se define por

$$\rho(A) = \max \{ |\lambda| \, : \, \lambda \text{ valor propio de } A \} . \tag{3.8}$$

Tenemos ahora:

Teorema 3.4. *Sea A una matriz $n \times n$. Entonces*

a) $\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$.

b) $\|A\|_2 = \sqrt{\rho(A^t A)}$.

c) $\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$.

3.2 Eliminación Gaussiana

Vamos ahora a estudiar uno de los métodos más conocidos para la solución directa de sistemas lineales, llamado el método de eliminación Gaussiana. Este método se utiliza comúnmente para la solución de sistemas de hasta cientos de variables, los llamados problemas "pequeños". Para problemas con miles o mas variables, los métodos más eficientes son los métodos de tipo "iterativos", que como bien sugiere el nombre, aproximan la solución del sistema utilizando algún tipo de iteración. En este libro no discutimos nada sobre los métodos iterativos pero el lector interesado puede encontrar material relacionado a esto en las referencias [2], [6], [11], y [30].

Vamos primero a repasar el método de eliminación Gaussiana con un ejemplo para luego discutirlo en el caso general.

Ejemplo 3.5. Considere el siguiente sistema de ecuaciones:

$$\begin{cases} x_1 + 2x_2 + x_3 &= 0, \\ 2x_1 + 2x_2 + 3x_3 &= 3, \\ -x_1 - 3x_2 &= 2. \end{cases} \tag{3.9}$$

Usamos la notación e_i para representar la i–esima ecuación del sistema. Una expresión como: $e_i = e_i + 2 * e_j$, lo que representa es la operación de sumarle a la ecuación e_i dos veces la ecuación e_j, y el resultado de esto sustituye a la ecuación e_i. Podemos resolver el sistema (3.9) mediante eliminación gaussiana como sigue:

$$\begin{array}{l} \\ e_2 = e_2 - 2e_1 \\ e_3 = e_3 + e_1 \end{array} \begin{cases} x_1 + 2x_2 + x_3 &= 0, \\ -2x_2 + x_3 &= 3, \\ -x_2 + x_3 &= 2, \end{cases}$$

$$e_3 = e_3 - \tfrac{1}{2}e_2 \quad \left\{ \begin{array}{rcl} x_1 + 2x_2 + x_3 &=& 0, \\ -2x_2 + x_3 &=& 3, \\ \tfrac{1}{2}x_3 &=& \tfrac{1}{2}. \end{array} \right.$$

Con este último sistema equivalente podemos obtener la solución sustituyendo para atrás:

- $\frac{1}{2}x_3 = \frac{1}{2} \Rightarrow x_3 = 1.$
- $-2x_2 + (1) = 3 \Rightarrow x_2 = -1.$
- $x_1 + 2(-1) + (1) = 0 \Rightarrow x_1 = 1.$

□

3.2.1 El procedimiento o algoritmo

Vamos ahora a generalizar el proceso del ejemplo anterior al caso de un sistema general 3×3. Escribimos el sistema original de la forma siguiente:

$$\left\{ \begin{array}{rcl} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 &=& b_1^{(1)}, \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 &=& b_2^{(1)}, \\ a_{31}^{(1)}x_1 + a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 &=& b_3^{(1)}. \end{array} \right. \tag{3.10}$$

Paso 1: Suponemos que $a_{11}^{(1)} \neq 0$ y definimos

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}}, \quad m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}}.$$

La entrada $a_{11}^{(1)}$ se llama el *pivote* en este paso. El sistema (3.10) reduce ahora a:

$$\begin{array}{l} e_2 = e_2 - m_{21}e_1 \\ e_3 = e_3 - m_{31}e_1 \end{array} \quad \left\{ \begin{array}{rcl} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 &=& b_1^{(1)}, \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 &=& b_2^{(2)}, \\ a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 &=& b_3^{(2)}, \end{array} \right. \tag{3.11}$$

donde

$$\left\{ \begin{array}{rcll} a_{ij}^{(2)} &=& a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, & i,j = 2,3, \\ b_j^{(2)} &=& b_j^{(1)} - m_{j1}b_1^{(1)}, & j = 2,3. \end{array} \right.$$

Paso 2: Suponemos ahora que $a_{22}^{(2)} \neq 0$ (pivote) y definimos

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}}.$$

Ahora (3.11) reduce a:

$$e_3 = e_3 - m_{32}e_2 \quad \left\{ \begin{array}{rcl} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 & = & b_1^{(1)}, \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 & = & b_2^{(2)}, \\ a_{33}^{(3)}x_3 & = & b_3^{(3)}, \end{array} \right.$$

donde

$$\left\{ \begin{array}{rcl} a_{33}^{(3)} & = & a_{33}^{(2)} - m_{32}a_{23}^{(2)}, \\ b_3^{(3)} & = & b_3^{(2)} - m_{32}b_2^{(2)}. \end{array} \right.$$

Paso 3: Hacemos ahora la sustitución para atrás para obtener la solución. Suponemos aquí que $a_{33}^{(3)} \neq 0$ (pivote):

$$x_3 = \frac{b_3^{(3)}}{a_{33}^{(3)}}, \quad x_2 = \frac{b_2^{(2)} - a_{23}^{(2)}x_3}{a_{22}^{(2)}}, \quad x_1 = \frac{b_1^{(1)} - a_{12}^{(1)}x_2 - a_{13}^{(1)}x_3}{a_{11}^{(1)}}.$$

Note que en los Pasos 1 y 2 del proceso que acabamos de describir, el sistema original se transforma a uno en forma *triangular*, esto es, en la ultima ecuación sólo aparece la variable x_3, en la segunda ecuación sólo aparecen x_2, x_3, y la primera ecuación tiene todas las variables. En el Paso 1, la primera columna se hizo cero a partir de la posición $(2,1)$, y luego en el Paso 2, la segunda columna se hizo cero a partir de la posición $(3,2)$. En el último paso, el sistema resultante se puede resolver fácilmente, trabajando desde la última ecuación hasta la primera.

Queremos ahora generalizar este proceso al caso de una matriz $n \times n$. Para describir este proceso en forma general, escribimos el sistema en el paso k, justo antes de hacer cero la columna k a partir de la posición $(k+1, k)$, por:

$$\left\{ \begin{array}{rcl} a_{11}^{(k)}x_1 + a_{12}^{(k)}x_2 + \cdots + a_{1n}^{(k)}x_n & = & b_1^{(k)}, \\ a_{21}^{(k)}x_1 + a_{22}^{(k)}x_2 + \cdots + a_{2n}^{(k)}x_n & = & b_2^{(k)}, \\ \vdots & & \vdots \\ a_{n1}^{(k)}x_1 + a_{n2}^{(k)}x_2 + \cdots + a_{nn}^{(k)}x_n & = & b_n^{(k)}, \end{array} \right.$$

para $k = 1, 2, \ldots, n$. Note que cuando $k = 1$ tenemos el sistema original. El proceso que ilustramos antes para el caso 3×3 se puede describir ahora, para el caso general, en dos etapas:

Eliminación: Para $k = 1, 2, \ldots, n-1$, suponemos que $a_{kk}^{(k)} \neq 0$ (el pivote). En el paso k, la mtriz de coeficientes del sistema anterior tiene la forma:

$$\left(\begin{array}{ccccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{1,k-1}^{(2)} & a_{1k}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ 0 & 0 & \cdots & 0 & a_{k+1,k}^{(k)} & \cdots & a_{k+1,n}^{(k)} & b_{k+1}^{(k)} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{array}\right)$$

Hacemos cero la columna k a partir de la posición $(k+1, k)$ mediante las operaciones de fila: $e_i = e_i - m_{ik} e_k$, $i = k+1, \ldots, n$, donde los *multiplicadores* están dados por:

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, \ldots, n. \tag{3.12}$$

La matriz y lado derecho del sistema para el paso $k+1$ quedan ahora dados por:

$$\begin{cases} a_{ij}^{(k+1)} = a_{ij}^{(k)}, & \begin{array}{l} i = 1, \ldots, k, \quad j = 1, \ldots, n, \\ i = k+1, \ldots, n, \quad j = 1, \ldots, k-1, \end{array} \\ a_{ik}^{(k+1)} = 0, & i = k+1, \ldots, n, \\ a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, & i, j = k+1, \ldots, n, \\ b_j^{(k+1)} = b_j^{(k)} - m_{jk} b_k^{(k)}, & j = k+1, \ldots, n. \end{cases} \tag{3.13}$$

Sustitución para Atrás: Es fácil ver que después del paso $n-1$ en la etapa

de eliminación, el sistema resultante tiene la forma:

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n &= g_1, \\ u_{22}x_2 + \cdots + u_{2n}x_n &= g_2, \\ \vdots & \\ u_{nn}x_n &= g_n, \end{cases} \tag{3.14}$$

donde

$$u_{ij} = a_{ij}^{(i)}, \quad g_i = b_i^{(i)}, \quad i, j = 1, \ldots, n. \tag{3.15}$$

Calculamos finalmente la solución del sistema mediante *sustitución para atrás*:

$$x_n = \frac{g_n}{u_{nn}}, \quad x_i = \frac{g_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}, \quad i = n-1, \ldots, 1. \tag{3.16}$$

Las fórmulas (3.12), (3.13), (3.15), (3.16) definen el *Método de Eliminación Gaussiana* en su forma básica. Las fórmulas (3.13) definen la parte de *eliminación* del método con la modificación del lado derecho, mientras que (3.16) se conoce como la *sustitución para atrás*.

En la practica, las cantidades del paso $k+1$, i.e., $a_{ij}^{(k+1)}$, $b_i^{(k+1)}$, se escriben o ocupan las posiciones de las cantidades correspondientes en el paso anterior. Además, como las entradas $a_{ik}^{(k+1)} = 0$, $i = k+1, \ldots, n$, podemos en su lugar almacenar en estas posiciones los multiplicadores (3.12). De modo que todo el proceso lo que requiere es el almacenaje de una matriz (los coeficientes) y un vector (el lado derecho).

3.2.2 Conteo operacional

Antes de discutir algunas modificaciones o variantes del método de eliminación gaussiana, vamos a hacer un estudio de la cantidad de operaciones aritméticas envueltas en dicho proceso. Esto se conoce como un *conteo operacional*. Examinando las fórmulas para los m_{ik}, $a_{ij}^{(k+1)}$ en (3.12) y (3.13), podemos construir la tabla siguiente:

PASO	SUMAS	MULTIPLICACIONES	DIVISIONES
1	$(n-1)^2$	$(n-1)^2$	$n-1$
2	$(n-2)^2$	$(n-2)^2$	$n-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	1	1	1
TOTAL	$n(n-1)(2n-1)/6$	$n(n-1)(2n-1)/6$	$n(n-1)/2$

donde usamos las fórmulas

$$\sum_{i=1}^{p} i = \frac{p(p+1)}{2}, \quad \sum_{i=1}^{p} i^2 = \frac{p(p+1)(2p+1)}{6}.$$

Es costumbre contar las operaciones de multiplicación y división juntas. De modo que la tabla de arriba la podemos resumir diciendo que en la parte de eliminación del método de eliminación gaussiana el total de:

$$\begin{aligned} \text{sumas y restas} &= \tfrac{n(n-1)(2n-1)}{6}, \\ \text{multiplicaciones y divisiones} &= \tfrac{n(n-1)(2n-1)}{6} + \tfrac{n(n-1)}{2}. \end{aligned}$$

Las fórmulas para los $b_i^{(k)}$ en (3.13) se conocen como la *modificación del lado derecho*. Estas conllevan:

$$\begin{aligned} \text{sumas y restas} &= n-1+(n-2)+\cdots+1 = \tfrac{n(n-1)}{2}, \\ \text{multiplicaciones y divisiones} &= n-1+(n-2)+\cdots+1 = \tfrac{n(n-1)}{2}. \end{aligned}$$

Las fórmulas (3.16) de la sustitución para atrás conllevan los siguientes totales de operaciones:

$$\begin{aligned} \text{sumas y restas} &= 1+2+\cdots+(n-1) = \tfrac{n(n-1)}{2}, \\ \text{multiplicaciones y divisiones} &= 1+2+\cdots+n = \tfrac{n(n+1)}{2}. \end{aligned}$$

Combinando todos los totales parciales hasta ahora, obtenemos que el proceso completo de eliminación gaussiana conlleva un total de:

$$\text{sumas y restas} = \frac{n(n-1)(2n+5)}{6}, \tag{3.17a}$$

$$\text{multiplicaciones y divisiones} = \frac{n(n^2+3n-1)}{3}. \tag{3.17b}$$

Note que para n *grande* ambos resultados son aproximadamente $(1/3)n^3$. Así que por ejemplo doblar n equivale a aproximadamente ocho veces más tiempo computacional. Observe también que la parte de eliminación es la que contribuye el termino proporcional a n^3. La modificación del lado derecho y la sustitución para atrás son ambas $O(n^2)$ según $n \longrightarrow \infty$. Note que estos tres procesos son independientes uno del otro. Por consiguiente si hay la necesidad de resolver varios sistemas todos con la misma matriz de coeficientes, la parte de eliminación debe hacerse una sola vez.

3.2.3 Pivoteo

Al derivar las fórmulas (3.13), (3.15), (3.16) asumimos que los pivotes $a_{kk}^{(k)}$ son todos distintos de cero. Si por el contrario algún $a_{kk}^{(k)} = 0$, entonces podemos argumentar matemáticamente que si la matriz de coeficientes del sistema original es invertible, entonces $a_{ik}^{(k)} \neq 0$ para algún índice i tal que $k < i \leq n$. En tal caso podemos intercambiar la fila i con la k y continuar con la eliminación. Este proceso de intercambiar filas se debe llevar a cabo también cuando ocurre un pivote pequeño, aunque distinto de cero, ya que utilizar un pivote pequeño puede causar que los efectos de redondeo debido a la aritmética finita de la computadora se propaguen rápidamente.

Ejemplo 3.6. Considere el sistema

$$\begin{cases} 3x - 2y + z &= 1, \\ x - \frac{2}{3}y + 2z &= 2, \\ -x + 2y - z &= 0. \end{cases}$$

El determinante de la matriz de coeficientes es $-20/3$ de modo que ésta es invertible. Vamos a resolver el sistema pero usando solo cuatro cifras decimales. Representamos el sistema con la matriz aumentada:

$$\left[\begin{array}{rrr|r} 3.000 & -2.000 & 1.000 & 1.000 \\ 1.000 & -0.6667 & 2.000 & 2.000 \\ -1.000 & 2.000 & -1.000 & 0.000 \end{array}\right].$$

Resolvemos ahora usando eliminación gaussiana usando solo cuatro cifras decimales:

$$\left[\begin{array}{rrr|r} 3.000 & -2.000 & 1.000 & 1.000 \\ 0.000 & -0.0001 & 1.667 & 1.667 \\ 0.000 & 1.333 & -0.6667 & 0.3333 \end{array}\right] \quad \begin{array}{l} m_{21} = \frac{1.000}{3.000} = 0.3333, \\ \\ m_{31} = \frac{-1.000}{3.000} = -0.3333, \end{array}$$

$$\left[\begin{array}{rrr|r} 3.000 & -2.000 & 1.000 & 1.000 \\ 0.000 & -0.0001 & 1.667 & 1.667 \\ 0.000 & 0.000 & 22220 & 22220 \end{array}\right] \quad m_{32} = \frac{1.333}{-0.0001} = -13330.$$

De aquí obtenemos que $x = 0.000$, $y = 0.000$, $z = 1.000$. ¡La solución exacta del sistema es $x = 1/2$, $y = 3/4$, $z = 1$! Tenemos pues que un error inicial del orden de 10^{-4} al entrar la matriz de coeficientes, induce un error del orden de uno en la solución calculada. Note que si antes de eliminar la segunda

columna, intercambiamos las filas dos y tres para evitar el pivote pequeño de -0.0001, tenemos

$$\left[\begin{array}{ccc|c} 3.000 & -2.000 & 1.000 & 1.000 \\ 0.000 & 1.333 & -0.6667 & 0.3333 \\ 0.000 & -0.0001 & 1.667 & 1.667 \end{array}\right] \quad e(2) \leftrightarrow e(3),$$

$$\left[\begin{array}{ccc|c} 3.000 & -2.000 & 1.000 & 1.000 \\ 0.000 & 1.333 & -0.6667 & 0.3333 \\ 0.000 & 0.000 & 1.667 & 1.667 \end{array}\right] \quad m_{32} = \tfrac{-0.0001}{1.333} = -0.0001,$$

de donde obtenemos que $x = 0.5000$, $y = 0.7502$, $z = 1.000$ los cuales tienen esencialmente cuatro cifras correctas. □

Para evitar el problema de pivotes distintos de cero pero pequeños usamos lo que se denomina como *pivoteo parcial*. Esto es, definimos el índice i_0 por:

$$\left|a_{i_0 k}^{(k)}\right| = \max_{k \le i \le n} \left|a_{ik}^{(k)}\right|. \tag{3.18}$$

Si $i_0 \neq k$, entonces intercambiamos las filas i_0 y k. Note que estamos haciendo el pivote $a_{kk}^{(k)}$ máximo en valor absoluto. De esta forma los multiplicadores m_{ik} satisfacen

$$|m_{ik}| \le 1, \quad 1 \le k < i \le n,$$

lo cúal ayuda a controlar la propagación de errores. Existe otra variante del pivoteo. En este caso se determinan los indices i_0 y j_0 tal que:

$$\left|a_{i_0 j_0}^{(k)}\right| = \max_{k \le i,j \le n} \left|a_{ij}^{(k)}\right|, \tag{3.19}$$

y se intercambian las filas i_0 y k y las columnas j_0 y k si $i_0 \neq k$ o $j_0 \neq k$ respectivamente. Esto se conoce como *pivoteo total* y se puede demostrar que es más efectivo que el pivoteo parcial en el control de la propagación de errores (cf. (3.45)). Pero el cálculo del máximo en (3.19) es mucho más costoso que en (3.18) por lo que en la práctica se prefiere el pivoteo parcial. Además se ha observado en pruebas usando matrices generadas aleatoriamente, que la diferencia entre ambos métodos no es significativa en términos de la propagación del error.

3.2.4 Cálculo de la Inversa de una Matriz

Suponga que A es una matriz invertible y A^{-1} su inversa. Escribimos A^{-1} y la matriz identidad I en forma particionada como

$$A^{-1} = (x_1, x_2, \ldots, x_n), \quad I = (e_1, e_2, \ldots, e_n),$$

donde los e_i's forman la base estándar de $\mathbb{R}^n$. Ahora como $AA^{-1} = I$, tenemos que

$$Ax_i = e_i, \quad 1 \leq i \leq n. \tag{3.20}$$

Así que para calcular A^{-1} debemos resolver n sistemas distintos pero con la misma matriz de coeficientes A. La eliminación de A la hacemos una vez lo cual requiere $(1/3)n^3$ operaciones aproximadamente. La modificación del lado derecho y la sustitución para atrás de cada uno de los sistemas en (3.20) conlleva aproximadamente n^2 operaciones cada uno. Así que en total tenemos $(1/3)n^3 + n(n^2) = (4/3)n^3$ operaciones aproximadamente. Este conteo se puede mejorar a n^3. (Vea los Ejercicios 3.38–3.39.) En muchas ocasiones las fórmulas que envuelven inversos de matrices se pueden reescribir en términos de sistemas lineales intermedios. Por el conteo operacional de arriba, las fórmulas con los sistemas lineales son preferibles ya que la solución de cada sistema envuelve aproximadamente $(1/3)n^3$ operaciones mientras que calcular los inversos toma $(4/3)n^3$ operaciones. Como regla general tenemos pues que:

REGLA: LOS INVERSOS DE MATRICES NO SE CALCULAN A MENOS QUE SE NECESITEN EXPLICITAMENTE.

Veamos una aplicación de este principio general.

Ejemplo 3.7. Supongamos que deseamos calcular la expresión $\alpha = c^t A^{-1} b$ donde $b, c \in \mathbb{R}^n$ y A es $n \times n$. Si calculamos A^{-1}, luego multiplicamos por b y finalmente el producto interior con c tenemos aproximadamente

Cálculo A^{-1}	$(4/3)n^3$
Multiplicación A^{-1} por b	n^2
Producto interior con c	n
TOTAL	$(4/3)n^3 + n^2 + n$

Suponga que en lugar de esto calculamos α mediante:

- Halle la solución x del sistema $Ax = b$.

- Calcule $\alpha = c^t x$.

Entonces un análisis similar al de arriba utilizando el resultado (3.17b) nos da un total de exactamente $(1/3)n^3 + n^2 + (2/3)n$ multiplicaciones y divisiones, lo cual es mucho mejor que la fórmula o el método directo. □

3.2.5 Factorización LU de una Matriz

Para discutir la factorización LU de una matriz necesitamos primero definir dos tipos especiales de matrices. Una matriz $A = (a_{ij})$ se dice que es *triangular superior* si $a_{ij} = 0$ para toda $i > j$. A es *triangular inferior* si $a_{ij} = 0$ para toda $i < j$. El adjetivo *unitaria* se añade en ambos casos si en adición $a_{ii} = 1$ para toda i. Una matriz que es ambas triangular superior e inferior se llama *diagonal*. Note que una matriz diagonal satisface $a_{ij} = 0$ para toda $i \neq j$.

Ejemplo 3.8. Las matrices

$$\begin{pmatrix} 1 & 2 & 2 \\ 0 & 5 & 0 \\ 0 & 0 & 7 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix},$$

son triangular superior, inferior (unitaria) y diagonal respectivamente. □

La matriz de coeficientes del sistema (3.14) es triangular superior y la denotamos por $U = (u_{ij})$. Usando los multiplicadores m_{ik} en (3.13) podemos definir la matriz triangular inferior unitaria L por:

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{pmatrix}. \tag{3.21}$$

Teorema 3.9. *Sea A una matriz $n \times n$ invertible. Defina las matrices L y U como arriba. Entonces si no se hace pivoteo en el proceso de eliminación gaussiana, tenemos que $A = LU$.*

Demostración: El elemento (i,j) del producto LU consiste del producto interior de la fila i de L con la columna j de U. Esto es

$$(LU)_{ij} = (m_{i1}, m_{i2}, \ldots, 1, 0, \ldots, 0) \begin{pmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{jj} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Si $i \leq j$, tenemos que

$$(LU)_{ij} = \sum_{k=1}^{i-1} m_{ik} u_{kj} + u_{ij}.$$

Usando las fórmulas (3.13) y la definición de U podemos escribir esto como

$$\begin{aligned} (LU)_{ij} &= \sum_{k=1}^{i-1} m_{ik} a_{kj}^{(k)} + a_{ij}^{(i)}, \\ &= \sum_{k=1}^{i-1} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) + a_{ij}^{(i)}, \\ &= (a_{ij}^{(1)} - a_{ij}^{(i)}) + a_{ij}^{(i)} = a_{ij}^{(1)} = a_{ij}, \end{aligned}$$

donde para simplificar la segunda sumatoria usamos que esta suma es telescópica. Ahora si $i > j$,

$$\begin{aligned} (LU)_{ij} &= \sum_{k=1}^{j} m_{ik} u_{kj} = \sum_{k=1}^{j-1} m_{ik} u_{kj} + m_{ij} u_{jj}, \\ &= \sum_{k=1}^{j-1} m_{ik} a_{kj}^{(k)} + \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}} a_{jj}^{(j)} = \sum_{k=1}^{j-1} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) + a_{ij}^{(j)}, \\ &= (a_{ij}^{(1)} - a_{ij}^{(j)}) + a_{ij}^{(j)} = a_{ij}^{(1)} = a_{ij}. \end{aligned}$$

Como $(LU)_{ij} = a_{ij}$ para toda i, j tenemos que $A = LU$. □

Ejemplo 3.10. Para la matriz

$$\begin{pmatrix} 1 & 2 & 1 \\ -1 & 4 & 2 \\ 3 & 1 & 1 \end{pmatrix},$$

tenemos mediante eliminación gaussiana:

$$\begin{pmatrix} 1 & 2 & 1 \\ -1 & 4 & 2 \\ 3 & 1 & 1 \end{pmatrix} \begin{array}{c} m_{21}=-1 \\ \rightarrow \\ m_{31}=3 \end{array} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 6 & 3 \\ 0 & -5 & -2 \end{pmatrix} \begin{array}{c} m_{32}=-\frac{5}{6} \\ \rightarrow \\ \ \end{array} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 6 & 3 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = U.$$

Si definimos ahora

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -\frac{5}{6} & 1 \end{pmatrix},$$

vemos que $A = LU$. □

Dado que $A = LU$, el sistema $Ax = b$ se puede escribir como $LUx = b$. La solución del sistema $Ax = b$ se puede ver entonces en tres etapas:

1. (**Eliminación**) Calcular la factorización $A = LU$.
2. (**Modificación del lado derecho**) Resolver el sistema triangular inferior $Lg = b$ para el vector g.
3. (**Sustitución para atrás**) Resolver el sistema triangular superior $Ux = g$ para el vector x.

Ejemplo 3.11. Considere el sistema

$$\begin{cases} x_1 + 2x_2 + x_3 &= 1, \\ -x_1 + 4x_2 + 2x_3 &= -1, \\ 3x_1 + x_2 + x_3 &= 0. \end{cases}$$

La matriz de coeficientes de este sistema coincide con la del Ejemplo 3.10 de modo que

$$A = \begin{pmatrix} 1 & 2 & 1 \\ -1 & 4 & 2 \\ 3 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -\frac{5}{6} & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 6 & 3 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = LU.$$

La solución de $Lg = b$ con $b = (1, -1, 0)^t$ esta dada por:

$$\begin{cases} g_1 & = & 1, & & \\ -g_1 + g_2 & = & -1 & \rightarrow & g_2 = -1 + 1 = 0, \\ 3g_1 - \frac{5}{6}g_2 + g_3 & = & 0 & \rightarrow & g_3 = -3(1) = -3. \end{cases}$$

Ahora $Ux = g$ nos da la solución del sistema original:

$$\begin{cases} \frac{1}{2}x_3 & = & -3 & \rightarrow & x_3 = -6, \\ 6x_2 + 3x_3 & = & 0 & \rightarrow & x_2 = -\frac{1}{2}x_3 = 3, \\ x_1 + 2x_2 + x_3 & = & 1 & \rightarrow & x_1 = 1 - 6 + 6 = 1. \end{cases}$$

□

En el caso de eliminación gaussiana con pivoteo parcial, el Teorema 3.9 sigue siendo cierto pero con la siguiente modificación: existe una matriz de permutación P (definida por las permutaciones de fila hechas durante el proceso) tal $PA = LU$. No obstante, las matrices L y U en esta factorización son diferentes a las del Teorema 3.9. De hecho, los multiplicadores que se almacenan en la matriz L son menor de uno en valor absoluto en eliminación gaussiana con pivoteo parcial, mientras que los del Teorema 3.9 pueden ser de cualquier tamaño.

Ejemplo 3.12. Considere la matriz

$$A = \begin{pmatrix} 1 & -1 & 2 \\ -2 & 1 & -1 \\ 4 & -1 & 2 \end{pmatrix}.$$

Luego de eliminar una columna de la matriz A, vamos a reemplazar los ceros bajo el pivote de dicha columna con los multiplicadores usados para eliminarla. De esta forma nos aseguramos que estos multiplicadores estarán sujetos a cualquier permutación hecha al eliminar las restantes columnas de A. Veamos:

$$\begin{pmatrix} 1 & -1 & 2 \\ -2 & 1 & -1 \\ 4 & -1 & 2 \end{pmatrix} \quad \begin{matrix} e_1 \leftrightarrow e_3 \\ \longrightarrow \end{matrix} \quad \begin{pmatrix} 4 & -1 & 2 \\ -2 & 1 & -1 \\ 1 & -1 & 2 \end{pmatrix}$$

$$\begin{array}{cc} m_{21} = -\frac{1}{2} & \begin{pmatrix} 4 & -1 & 2 \\ -1/2 & 1/2 & 0 \\ 1/4 & -3/4 & 3/2 \end{pmatrix} \\ \longrightarrow & \\ m_{31} = \frac{1}{4} & \end{array}$$

$$\begin{array}{cc} e_2 \leftrightarrow e_3 & \begin{pmatrix} 4 & -1 & 2 \\ 1/4 & -3/4 & 3/2 \\ -1/2 & 1/2 & 0 \end{pmatrix} \\ \longrightarrow & \end{array}$$

$$\begin{array}{cc} m_{32} = -\frac{2}{3} & \begin{pmatrix} 4 & -1 & 2 \\ 1/4 & -3/4 & 3/2 \\ -1/2 & -2/3 & 1 \end{pmatrix} \\ \longrightarrow & \end{array}$$

Tenemos pues que el método de eliminación gaussiana con pivoteo parcial produce las matrices:

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ -1/2 & -2/3 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & -1 & 2 \\ 0 & -3/4 & 3/2 \\ 0 & 0 & 1 \end{pmatrix},$$

las cuales cumplen que $PA = LU$. Note que en principio, si uno conoce la matriz de permutación P de antemano, podríamos aplicar los cambios de fila a A antes de comenzar el proceso de eliminación, el cuál entonces no requeriría cambios de fila. El problema o limitación con ésto es que en general no es posible saber de antemano las permutaciones que se requerirán en el proceso de eliminación. □

3.2.6 Cálculo de Determinantes

El método teórico de calcular determinantes utilizando los cofactores de la matriz tiene un conteo operacional de multiplicaciones $O(n!)$ para una matriz $n \times n$. Usando la factorización LU de la matriz, podemos obtener un método más eficiente para este cálculo. Esto es dado que $A = LU$ donde L es triangular inferior unitaria (unos en la diagonal) y U es triangular superior, tenemos que $\det(A) = \det(L)\det(U)$. Pero $\det(L) = 1$ porque es triangular inferior unitaria, y $\det(U)$ es el producto de la diagonal principal de U. En general si la factorización se obtiene mediante eliminación gaussiana con pivoteo parcial, entonces $PA = LU$ para una matriz de permutación P, y tenemos la fórmula

$$\det(A) = (-1)^k u_{11} u_{22} \cdots u_{nn},$$

donde k es el número de intercambios de filas hechos durante la eliminación. Para evitar la posibilidad de *overflow* en el computo del producto $u_{11}u_{22}\cdots u_{nn}$, se calcula el logarítmo del valor absoluto de esta expresión.

Ejemplo 3.13. Para la matriz del Ejemplo 3.12, tenemos que

$$\det(A) = (-1)^2(4)(-3/4)(1) = -3.$$

□

3.2.7 El operador "\" de MATLAB

La solución del sistema lineal $Ax = b$ se puede obtener en MATLAB mediante la instrucción `x=A\b`. Este computo se lleva a cabo usando eliminacción gaussiana con pivoteo parcial. Las tres etapas del proceso de eliminación gaussiana se pueden realizar en MATLAB combinando el operador "\" con la función `lu`. La secuencia de llamada:

```
[L,U,P]=lu(A);
```

produce las matrices triangular inferior unitaria, triangular superior, y de permutación `L`, `U`, `P` respectivamente, tal que `P*A=L*U` (aproximadamente). Así que las tres etapas del proceso de eliminación gaussiana: eliminación, modificación del lado derecho, y sustitución para atrás, se pueden llevar a cabo en MATLAB con las siguientes instrucciones:

```
[L,U,P]=lu(A);
g=L\(P*b);
x=U\g;
```

Si no nos interesa generar la matriz `P`, el sistema se puede resolver de la forma siguiente:

```
[L,U]=lu(A);
g=L\b;
x=U\g;
```

En este caso la `L` ya incluye la matriz de permutación. Esto es, en este último cálculo, la `L` es igual a `PL` donde `P` y `L` son las que calculamos arriba.

El operador "\" en los pasos de modificación del lado derecho y sustitución para atrás puede "detectar" la estructura triangular de las matrices `L` y `U`

y utiliza el método apropiado para resolver dichos sistemas. Este operador también puede detectar cuando la matriz es simétrica o escasa de forma que utiliza el método más eficiente para resolver el sistema correspondiente. (Vea los Ejemplos 3.21 y 3.22.)

MATLAB también cuenta con las funciones `det` e `inv` para calcular determinantes e inversos respectivamente de matrices utilizando para esto los métodos basados en eliminación gaussiana.

3.3 Variantes del Método de Eliminación Gaussiana

Vamos ahora a discutir dos métodos alternos para calcular la factorización LU de una matriz los cuales aprovechan alguna estructura particular de la matriz. El primero de ellos es la *Factorización de Cholesky* que se usa para matrices simétricas y positivas definidas. El otro método aprovecha la estructura escasa de la matriz en el caso que esta sea *tridiagonal*.

3.3.1 Factorización de Cholesky

En esta sección estudiamos el caso en que A es simétrica y positiva definida. Se puede entonces demostrar que existe una matriz triangular inferior L (no necesariamente unitaria) tal que

$$A = LL^t,$$

lo cual se conoce como la *factorización de Cholesky* de A. De hecho las entradas de $L = (l_{ij})$ se pueden calcular mediante las fórmulas:

Algoritmo 3.14. Para $j = 1, 2, \ldots, n$,

1. $l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$.
2. Para $i = j+1, \ldots, n$,
   (a) $l_{ij} = l_{jj}^{-1}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right)$.

Estas fórmulas se obtienen multiplicando las filas de L por las columnas de L^t e igualando a las entradas correspondientes de A. Un conteo operacional

de estas fórmulas muestra que el total de multiplicaciones y divisiones es aproximadamente $(1/6)n^3$, i.e., la mitad que en eliminación gaussiana básico. (Note sin embargo que hay que calcular n raíces cuadradas). La ganancia aquí se debe a que se utilizó la simetría de la matriz A. Note también que como A es simétrica solo hay que almacenar en la computadora la mitad de la matriz al igual que para L que es triangular inferior. La función `chol` de MATLAB se utiliza para calcular la factorización de Cholesky de una matriz. Además esta función se puede usar para verificar si una matriz dada es positiva definida. En la secuencia de llamada:

```
[L,p]=chol(A);
```

se obtiene que `p=0` si `A` es positiva definida y `p` es un entero positivo de lo contrario.

Ejemplo 3.15. Considere la matriz

$$A = \begin{pmatrix} 3 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{pmatrix},$$

que es simétrica y positiva definida ya que sus valores propios son $2, 2, 4$. Buscamos

$$L = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix},$$

tal que $A = LL^t$. Tenemos pues la ecuación matricial

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix} = \begin{pmatrix} 3 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{pmatrix},$$

lo cual produce las siguientes ecuaciones para las entradas de L:

$$l_{11}^2 = 3 \Rightarrow l_{11} = \sqrt{3}, \quad l_{11}l_{21} = 0 \Rightarrow l_{21} = 0, \quad l_{11}l_{31} = 1 \Rightarrow l_{31} = \frac{1}{\sqrt{3}},$$

$$l_{21}^2 + l_{22}^2 = 2 \Rightarrow l_{22} = \sqrt{2}, \quad l_{21}l_{31} + l_{22}l_{32} = 0 \Rightarrow l_{32} = 0,$$

$$l_{31}^2 + l_{32}^2 + l_{33}^2 = 3 \Rightarrow l_{33} = \frac{2\sqrt{6}}{3}.$$

Tenemos ahora que

$$L = \begin{pmatrix} \sqrt{3} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{2\sqrt{6}}{3} \end{pmatrix}.$$

El mismo calculo lo podemos hacer en MATLAB mediante:

```
[R,p]=chol([3,0,1;0,2,0;1,0,3])

R =
    1.7321         0    0.5774
         0    1.4142         0
         0         0    1.6330

p =
     0
```

La matriz `R` es la transpuesta de la L calculada arriba y `p=0` indica que la matriz original es positiva definida. □

3.3.2 Sistemas Tridiagonales

Una matriz A se llama *tridiagonal* si $a_{ij} = 0$ para toda i, j tal que $|i - j| > 1$. Esto es, todas las entradas de A son cero excepto posiblemente en las diagonales inferior, superior y principal. Podemos pues escribir A de la siguiente forma:

$$A = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_n & b_n \end{pmatrix}. \tag{3.22}$$

Para almacenar A en la computadora usamos tres vectores de tamaño n lo que da un total de $3n$ lugares de memoria en comparación con n^2 para una matriz densa. Vamos ahora a resolver el sistema $Ax = \tilde{b}$ aprovechando la estructura de ceros de A. Primero buscamos la factorización LU de A. Para

esto buscamos L y U de la forma:

$$A = LU = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \alpha_2 & 1 & 0 & \cdots & 0 \\ 0 & \alpha_3 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \alpha_n & 1 \end{pmatrix} \begin{pmatrix} \beta_1 & c_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & c_2 & \cdots & 0 \\ 0 & 0 & \beta_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \beta_n \end{pmatrix}. \quad (3.23)$$

Multiplicando e igualando a las entradas correspondientes de A obtenemos:

- fila uno de L por columna uno de U: $\beta_1 = b_1$.
- fila dos de L por columna uno de U: $\alpha_2\beta_1 = a_2$.
- fila dos de L por columna dos de U: $\alpha_2 c_1 + \beta_2 = b_2$.
- fila tres de L por columna dos de U: $\alpha_3\beta_2 = a_3$.
- fila tres de L por columna tres de U: $\alpha_3 c_2 + \beta_3 = b_3$, etc..

De esta forma obtenemos las fórmulas:

Algoritmo 3.16. Descomposición LU para matrices tridiagonales:

1. $\beta_1 = b_1$.
2. Para $j = 2, 3, \ldots, n$,
   (a) $\alpha_j = \dfrac{a_j}{\beta_{j-1}}$.
   (b) $\beta_j = b_j - \alpha_j c_{j-1}$.

El total de multiplicaciones y divisiones en estas fórmulas es de $2n - 2$ (compare con $(1/3)n^3$ para eliminación gaussiana básico). Para resolver $Lg = \tilde{b}$ es fácil ver que las fórmulas son:

Algoritmo 3.17. Modificación del lado derecho para sistemas tridiagonales:

1. $g_1 = \tilde{b}_1$.
2. Para $j = 2, 3, \ldots, n$,
   (a) $g_j = \tilde{b}_j - \alpha_j g_{j-1}$.

El total de multiplicaciones y divisiones en este cálculo es de $n-1$. Finalmente la solución de $Ux = g$ se obtiene mediante las fórmulas:

Algoritmo 3.18. Sustitución para atrás para sistemas tridiagonales:

1. $x_n = \dfrac{g_n}{\beta_n}$.

2. Para $j = n-1, \ldots, 1$,

   (a) $x_j = \beta_j^{-1}(g_j - c_j x_{j+1})$.

El conteo aquí de multiplicaciones y divisiones es de $2n-1$. Así que en total para resolver el sistema $Ax = \tilde{b}$ donde A es tridiagonal envuelve $5n-4$ multiplicaciones y divisiones. En todo este proceso necesitamos que los pivotes $\beta_i \neq 0$ y es conveniente que los α's sean menor de uno en valor absoluto. Esto se cumple si (ver [11]):

$$\begin{array}{l} |b_1| > |c_1| > 0, \\ |b_i| \geq |a_i| + |c_i|\,, \quad a_i, c_i \neq 0, \quad 2 \leq i \leq n-1, \\ |b_n| > |a_n| > 0. \end{array}$$

Ejemplo 3.19. Considere la matriz tridiagonal

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix}.$$

Aquí $a_j = 1$, $j = 2, 3$, $b_j = 3$, $j = 1, 2, 3$, $c_j = 1$, $j = 1, 2$. Así que

$$\beta_1 = 3, \quad \alpha_j = \beta_{j-1}^{-1}, \quad \beta_j = 3 - \alpha_j, \quad j = 2, 3,$$

de donde obtenemos que

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ 0 & \frac{3}{8} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 1 & 0 \\ 0 & \frac{8}{3} & 1 \\ 0 & 0 & \frac{21}{8} \end{pmatrix}.$$

□

3.4 Matrices Escasas en MATLAB

MATLAB ofrece varias funciones para manipular matrices escasas, como las tridiagonales, resultando en ahorros de memoria y rapidez de ejecución de los programas. La función básica para crear matrices escasas en MATLAB es `sparse` que tiene el siguiente formato:

`sparse(row,col,val,m,n)`,

donde `row` y `col` son vectores de los indices de filas y columnas respectivamente de las entradas distintas de cero de la matriz, `val` es el vector que contiene los valores distintos de cero, y `(m,n)` son las dimensiones de la matriz. Internamente MATLAB solo almacena los arreglos `row`, `col`, y `val` y utiliza la `(m,n)` para los cálculos en operaciones matriciales.

Ejemplo 3.20. La matriz

$$A = \begin{pmatrix} 1 & 0 & 3 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 0 & 4 & 1 & 3 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 1 & 0 & 0 & 3 \end{pmatrix},$$

se puede almacenar como una matriz escasa en MATLAB con la instrucción

```
A=sparse([1 1 2 2 3 3 3 4 5 5],[1 3 1 2 2 3 4 4 2 5],...
     [1 3 -1 2 4 1 3 5 1 3],5,5)
```

□

Otras funciones adicionales para trabajar con matrices escasas lo son:

full convierte una matriz escasa a una densa.

find devuelve los indices de las entradas distintas de cero en la matriz escasa.

nnz devuelve el número de elementos distintos de cero en la matriz.

Veamos ahora un ejemplo que ilustra el uso de matrices escasas en MATLAB y como estas se pueden utilizar para obtener algorítmos mas eficientes.

Ejemplo 3.21. Considere el sistema

$$\begin{pmatrix} 4 & 1 & 0 & \cdots & 0 \\ 1 & 4 & 1 & \cdots & 0 \\ 0 & 1 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{500} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ 500 \end{pmatrix}.$$

En el siguiente programa en MATLAB utilizamos matrices escasas para resolver este sistema y lo comparamos con la solución del mismo pero con una matriz densa. Veamos

```
udiag=sparse([1:499],[2:500],ones(1,499),500,500);
a=sparse([1:500],[1:500],4*ones(1,500),500,500);
a=udiag'+a+udiag;
b=full(a);
rightside=[1:500]';
tic;x=a\rightside;t1=toc;
tic;x=b\rightside;t2=toc;
```

Los resultados de la corrida fueron los siguientes:

```
Tiempo para el metodo con la matriz escasa: 0.11
Tiempo para el metodo con la matriz densa: 4.84
```

¡Note que el resolver el sistema con la matriz escasa corre 44 veces más rápido que usando la matriz densa! La función \ de MATLAB cuando detecta una matriz escasa, utiliza el método más apropiado para dicha estructura. En este ejemplo MATLAB usa automáticamente el Algorítmo (3.16)–(3.18) para sistemas tridiagonales. El uso de la matriz densa en este ejemplo es solo para ilustrar las diferencias en los tiempos de corrida. □

Ejemplo 3.22. La función `lu` de MATLAB cuando se aplica a una matriz simétrica y positiva definida lo que devuelve es la factorización LU de la matriz, donde L tiene unos en la diagonal. Este calculo se hace con el método de eliminación gaussiana para matrices generales que requiere aproximadamente $n^3/3$ multiplicaciones y divisiones vs $n^3/6$ para la factorización de Cholesky. En este ejemplo examinamos esto para una matriz 500×500 simétrica y positiva definida. Para construir esta matriz utilizamos la función `qr` de MATLAB que calcula la factorización QR de una matriz. (Ver Teorema 5.17.) En el siguiente programa calculamos las factorizaciones LU y de Cholesky de la matriz y comparamos los tiempos de ejecución:

```
A=rand(500,500);
[Q,R]=qr(A);
B=Q*diag(1:500)*Q';
tic;[L,U]=lu(B);t1=toc;
tic;[R,p]=chol(B);t2=toc;
disp(['Tiempo para la factorizacion LU: ' num2str(t1)]);
disp(['Tiempo para el metodo de Cholesky: ' num2str(t2)]);
```

Los resultados fueron los siguientes:

```
Tiempo para la factorizacion LU: 0.020623
Tiempo para el metodo de Cholesky: 0.005307
```

Se puede apreciar aquí que el computo de la factorización de Cholesky se realiza mucho más rápido (casi cuatro veces en este caso) que el de la factorización LU basado en el método de eliminación gaussiana básico. □

3.5 Estabilidad de Sistemas Lineales

En esta sección estudiaremos cúan sensitiva es la solución del sistema lineal $Ax = b$ a cambios o perturbaciones en los datos A y b. Note que A o b pueden contener errores ya sea por las truncaciones envueltas al entrar estos en la computadora o porque sean datos experimentales. De modo que queremos ver como esto puede afectar la solución calculada. Aquí suponemos que los sistemas bajo consideración se resuelven en forma exacta y que solo los errores en los datos iniciales afectan el cálculo. Más adelante estudiaremos los efectos del método numérico.

3.5.1 Teoremas de Perturbación

Primero consideramos variaciones en b únicamente. Sea $\hat{b}$ una aproximación de b y $\hat{x}$ la solución del sistema "perturbado" $A\hat{x} = \hat{b}$. Queremos estimar el error $x - \hat{x}$ en términos de la diferencia $b - \hat{b}$. En esta sección $\|\cdot\|$ se usa para denotar cualquiera de las normas p definidas en la Sección (3.1). Tenemos ahora el siguiente teorema:

Teorema 3.23. *Sea A una matriz invertible. Sean x, $\hat{x}$ las soluciones de los sistemas $Ax = b$ y $A\hat{x} = \hat{b}$. Entonces*

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \|A\| \left\|A^{-1}\right\| \frac{\left\|b - \hat{b}\right\|}{\|b\|}. \tag{3.24}$$

Demostración: De $Ax = b$ y $A\hat{x} = \hat{b}$ obtenemos que $A(x - \hat{x}) = b - \hat{b}$. Como A es invertible, entonces $x - \hat{x} = A^{-1}(b - \hat{b})$. Usando la quinta propiedad de las normas obtenemos que $\|x - \hat{x}\| \leq \|A^{-1}\| \left\|b - \hat{b}\right\|$. Dividiendo por $\|x\|$ en ambos lados obtenemos

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \frac{\|A^{-1}\|}{\|x\|} \left\|b - \hat{b}\right\| = \frac{\|A\| \|A^{-1}\|}{\|A\| \|x\|} \left\|b - \hat{b}\right\|.$$

Pero $\|b\| = \|Ax\| \leq \|A\| \|x\|$. Usando esto arriba obtenemos el resultado. □

Si definimos $\text{Rel}(\hat{x}) = \|x - \hat{x}\| / \|x\|$, y el *número de condición*[1] de A por

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|, \tag{3.25}$$

entonces podemos escribir el resultado del teorema como

$$\text{Rel}(\hat{x}) \leq \kappa(A)\text{Rel}(\hat{b}). \tag{3.26}$$

De modo que si $\kappa(A)$ es "grande", entonces $\text{Rel}(\hat{b})$ se puede magnificar por un factor hasta de $\kappa(A)$. Decimos pues que el sistema $Ax = b$ es *mal acondicionado* si $\kappa(A)$ es *grande* (en el orden de 10^6 o más). De lo contrario el sistema se dice que es *bien acondicionado*. Note que como $AA^{-1} = I$, entonces

$$1 = \|I\| = \left\|AA^{-1}\right\| \leq \|A\| \left\|A^{-1}\right\| = \kappa(A),$$

i.e., $\kappa(A) \geq 1$ para toda matriz A y por consiguiente, el error en los datos iniciales nunca disminuye.

En forma similar se puede demostrar el siguiente resultado que incluye variaciones en A y b ambas. (Ver [11]).

[1] El número de condición de A en la norma $\|\cdot\|_p$ se denota por $\kappa_p(A)$.

Teorema 3.24. *Sea A una matriz invertible y $\hat{A}$ otra matriz tal que*

$$\left\|A - \hat{A}\right\| < \frac{1}{\|A^{-1}\|}. \tag{3.27}$$

Entonces $\hat{A}$ es invertible y

$$Rel(\hat{x}) \leq \frac{\kappa(A)}{1 - \kappa(A)Rel(\hat{A})}\left(Rel(\hat{A}) + Rel(\hat{b})\right), \tag{3.28}$$

donde $Rel(\hat{A}) = \left\|A - \hat{A}\right\| / \|A\|$.

¡CUIDADO! El error relativo definido aquí para un vector o matriz, no es lo mismo que el error relativo en cada componente. De hecho se puede verificar (Ejercicio 3.2) que

$$\frac{\|x - \hat{x}\|_p}{\|x\|_p} \leq \max_i \left\{\left|\frac{x_i - \hat{x}_i}{x_i}\right|\right\},$$

para cualquier $p \in [1, \infty]$.

Ejemplo 3.25. Para $x = (1, 100)$ y $\hat{x} = (0.1, 100.1)$, tenemos que:

$$\left|\frac{x_1 - \hat{x}_1}{x_1}\right| = \frac{1 - 0.1}{1} = 0.9, \quad \left|\frac{x_2 - \hat{x}_2}{x_2}\right| = \left|\frac{100 - 100.1}{100}\right| = 10^{-3},$$

lo que claramente indica que $\hat{x}_1$ es una pobre aproximación de x_1 y que $\hat{x_2}$ tiene esencialmente tres cifras correctas como aproximación de x_2. Pero si calculamos el error relativo en $\hat{x}$ usando la definición de error relativo para vectores (usando la norma p con $p = \infty$), tenemos que:

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} = 9 \times 10^{-3},$$

lo cual daría la impresión incorrecta de que $\hat{x}$ aproxima a x con dos cifras correctas. □

En MATLAB podemos calcular el número de condición de A en la norma dos con la instrucción `cond(A)`. La instrucción `rcond(A)` calcula una aproximación del inverso del número de condición de A pero en la norma uno de la matriz. `rcond` es mucho más rápida que `cond` para matrices de tamaño grande.

Ejemplo 3.26. La matriz de Hilbert de orden n se define por

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix}. \tag{3.29}$$

Note que H_n es simétrica para toda $n \geq 1$. Usando la instrucción de MATLAB, `hilb(n)` podemos generar la matriz de Hilbert de orden n. La instrucción `invhilb(n)` calcula el inverso exacto[2] de H_n. Por ejemplo `hilb(5)` produce:

```
ans =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

que es la matriz de Hilbert de orden 5 truncada a cuatro cifras aunque MATLAB guarda más cifras internamente. Por otro lado `invhilb(5)` produce

```
ans =
   25     -300        1050       -1400         630
 -300     4800      -18900       26880      -12600
 1050   -18900       79380     -117600       56700
-1400    26880     -117600      179200      -88200
  630   -12600       56700      -88200       44100
```

que es el inverso exacto de H_5. Podemos calcular el número de condición de H_n en cualquier norma p mediante:

```
norm(hilb(n),p)*norm(invhilb(n),p)
```

Por ejemplo para $n = 5$ y $p = \infty$ tenemos que

[2]Para $n < 15$ la instrucción `invhilb(n)` calcula el inverso exacto de H_n. Para valores mayores de n el inverso es aproximado.

```
norm(hilb(5),inf)*norm(invhilb(5),inf)

ans =
      943656
```

lo que indica el mal acondicionamiento de la matriz de Hilbert apenas para $n = 5$. Este número compara en orden de magnitud con `cond(hilb(5))` la cual produce

```
ans =
  4.7661e+005
```

y con `1/rcond(hilb(5))`

```
ans =
  6.9409e+005
```

Usando la instrucción `flops` de MATLAB podemos estimar que el cómputo con `cond` requiere de 937 operaciones de punto flotante mientras que el cálculo con `rcond` toma 296 operaciones. En la siguiente tabla mostramos los numeros de condición para las matrices H_n, $n = 1, 2, \ldots, 10$:

n	cond(hilb(n))	n	cond(hilb(n))
1	1	6	1.4951e+7
2	1.9282e+1	7	4.7537e+8
3	5.2406e+2	8	1.5258e+10
4	1.5514e+4	9	4.9315e+11
5	4.7661e+5	10	1.6025e+13

□

3.5.2 Significado del número de condición

Vamos a examinar en más detalles el significado del número de condición de una matriz. Si $\kappa(A) = O(10^t)$, entonces de acuerdo al resultado (3.28) podemos perder hasta t cifras significativas al resolver $Ax = b$ independiente del método numérico usado.

Ejemplo 3.27. Considere el sistema

$$\begin{cases} x_1 + x_2 &= 1, \\ x_1 + 1.0001x_2 &= 1, \end{cases}$$

cuya solución exacta es $x_1 = 1, x_2 = 0$. El número de condición de la matriz de coeficientes es de aproximadamente 40,000 lo que la hace moderadamente mal acondicionada. Si cambiamos el lado derecho a $(1, 1.001)^t$ vemos que la solución del sistema resultante es ahora $(-9, 10)^t$, i.e., una perturbación del orden de 10^{-3} en el lado derecho induce un cambio del orden de diez en la solución. O sea se perdieron cuatro cifras significativas. □

El fenómeno de perdida de cifras significativas que predice la desigualdad (3.28) y que ilustra el Ejemplo 3.27, es en el peor de los casos. Esto es, no necesariamente se tienen que perder t cifras significativas pero es lo peor que puede suceder. En ocasiones un número de condición grande no es necesariamente indicativo de que vaya a ocurrir imprecisión numérica, como por ejemplo, en el caso de sistemas triangulares.

Ejemplo 3.28. Para la matriz

$$\begin{pmatrix} 0.001 & 0 \\ 1 & 10000 \end{pmatrix},$$

el número de condición es de al menos 10^7. (Vea el Teorema 3.30.) Considere ahora el sistema correspondiente a esta matriz:

$$\begin{cases} 0.001x_1 & = & b_1, \\ x_1 + 10000x_2 & = & b_2. \end{cases}$$

Note que las filas de este sistema representan rectas en el plano x_1x_2, la primera de éstas siendo perpendicular, y la segunda con pendiente 10^{-4}. Como estas rectas son casi perpendiculares entre si, entonces la matriz de coeficientes no se considera mal acondicionada a pesar de su número de condición tan alto. Vea también el Ejercicio 3.28 para un análisis más riguroso sobre este punto. □

A pesar del resultado ilustrado por el Ejemplo 3.28, el número de condición de una matriz se utiliza comúnmente como un indicador de mal acondicionamiento. Cuando hay número de condición alto, la matriz del sistema debe ser examinada más cuidadosamente para determinar si es efectivamente mal acondicionada o si es un problema de "balanceo", i.e., diferencias grandes entre los tamaños de las entradas de la matriz. De ser ésto ultimo, existen técnicas o procedimientos numéricos para balancear las entradas de la matriz.

Veamos ahora un par de resultados adicionales que nos ayudan a entender mejor el significado del número de condición. (Ver [2], [11]).

Teorema 3.29. *Sea A una matriz invertible. Entonces*

$$\frac{1}{\kappa(A)} = \min\left\{\frac{\|A-B\|}{\|A\|} \,:\, B \text{ es singular}\right\}. \tag{3.30}$$

O sea que si $\kappa(A)$ es grande, perturbaciones pequeñas en A pueden producir una matriz singular. O dicho de otra forma, si $\kappa(A)$ es grande, la matriz A se puede aproximar bien con matrices singulares.

Si λ es un valor propio de A, entonces es fácil ver que $|\lambda| \leq \|A\|$. Como los valores propios de A^{-1} son los recíprocos de los de A, si A es invertible, tenemos entonces el siguiente resultado.

Teorema 3.30. *Sea A una matriz invertible. Entonces*

$$\kappa(A) \geq \frac{\max\{|\lambda| \,:\, \lambda \in \sigma(A)\}}{\min\{|\lambda| \,:\, \lambda \in \sigma(A)\}} \equiv \kappa_*(A), \tag{3.31}$$

donde $\sigma(A)$ es el conjunto de todos los valores propios de A. Si A es simétrica, entonces $\kappa_2(A) = \kappa_*(A)$.

3.6 Refinamiento Iterativo–Método de Residuos

El Método de Eliminación Gaussiana produce la solución exacta de un sistema lineal en aproximadamente $(2/3)n^3$ operaciones aritméticas[3] donde n es el orden de la matriz de coeficientes A. En general, debido a que los cómputos se hacen con precisión finita, la solución obtenida por eliminación gaussiana no es la solución exacta del sistema (aunque está cerca de serlo si el sistema es bien acondicionado). Veamos ahora un proceso iterativo que toma la solución calculada por el Método de Eliminación Gaussiana y trata de mejorarla.

Sea $x^{(0)}$ la solución calculada por eliminación gaussiana y x la solución exacta del sistema, i.e., $Ax = b$. Para cualquier $x^{(k)}$, $k \geq 0$ definimos el *error exacto* y el *residual* respectivamente por

$$\hat{e}^{(k)} = x - x^{(k)}, \quad r^{(k)} = b - Ax^{(k)}. \tag{3.32}$$

[3]Aquí estamos contando colectivamente todas las operaciones de punto flotante en el proceso, esto es, sumas, restas, multiplicaciones, y divisiones, en un mismo total.

Entonces es fácil ver que

$$A\hat{e}^{(k)} = r^{(k)}, \tag{3.33}$$

de modo que utilizando la factorización LU de A ya calculada para obtener $x^{(0)}$, podemos resolver la ecuación (3.33). Pero este sistema tampoco se puede resolver en forma exacta obteniendo así una aproximación $e^{(k)}$ de $\hat{e}^{(k)}$. Definimos ahora

$$x^{(k+1)} = x^{(k)} + e^{(k)}, \quad k \geq 0, \tag{3.34}$$

la cual en general es una mejor aproximación de x que $x^{(k)}$. El proceso continua de esta manera en forma iterativa hasta que un cierto criterio de *convergencia* se cumpla y el proceso termina. En forma algorítmica tenemos:

Algoritmo 3.31. Proceso de refinamiento iterativo:

1. Sean L y U los factores (aproximados) de A.
2. Sea $x^{(0)}$ la solución aproximada de $Ax = b$ obtenida mediante eliminación gaussiana.
3. Para $k = 0, 1, 2, \ldots,$
   (a) Calcule $r^{(k)} = b - Ax^{(k)}$.
   (b) Usando los factores L y U del paso (1), resuelva $Ae^{(k)} = r^{(k)}$.
   (c) Ponga $x^{(k+1)} = x^{(k)} + e^{(k)}$.

Note que en el cálculo del residuo $r^{(k)}$ en el paso (3a) tenemos cancelación de cifras significativas ya que $Ax^{(k)} \approx b$. Para evitar ésto el cálculo debe hacerse en precisión extendida y el resto de los cálculos en precisión normal. El ciclo en (3) se puede detener con un criterio como

$$\left\|e^{(k)}\right\| \leq 10^{-t} \left\|x^{(k)}\right\|, \tag{3.35}$$

donde $\|\cdot\|$ representa una norma vectorial cualquiera, lo que garantiza aproximadamente t cifras correctas en la solución calculada en caso de convergencia. Normalmente, sin embargo, el ciclo no debe ejecutarse más de dos veces.

Note que el cálculo de $r^{(k)}$ en (3a) requiere de n^2 multiplicaciones con aproximadamente la misma cantidad de sumas y restas. Para la solución del sistema en (3b) se usa la factorización LU de A del paso (1) de modo que ésto toma aproximadamente n^2 operaciones. Así que cada pasada por el ciclo (3) conlleva aproximadamente $2n^2$ multiplicaciones y divisiones.

Ejemplo 3.32. Considere el sistema lineal

$$\begin{cases} 0.375x_1 + 0.271x_2 &= 0.612, \\ 0.491x_1 + 0.381x_2 &= 0.311. \end{cases}$$

La solución exacta a tres cifras significativas es $x_1 = 15.2$, $x_2 = -18.7$. Si resolvemos el sistema mediante eliminación gaussiana sin pivoteo y usando aritmética de tres cifras tenemos:

$$m_{21} = \frac{0.491}{0.375} = 1.31 \qquad \begin{cases} 0.375x_1 + 0.271x_2 &= 0.612, \\ 0.0260x_2 &= -0.491, \end{cases}$$

de donde obtenemos $x_1^{(0)} = 15.3$, $x_2^{(0)} = -18.9$. El residual de esta solución aproximada está dado por:

$$r^{(0)} = \begin{pmatrix} 0.612 \\ 0.311 \end{pmatrix} - \begin{pmatrix} 0.375 & 0.271 \\ 0.491 & 0.381 \end{pmatrix} \begin{pmatrix} 15.3 \\ -18.9 \end{pmatrix} = \begin{pmatrix} -0.00360 \\ -0.000400 \end{pmatrix},$$

correcto a tres cifras. Podemos calcular $e^{(0)}$ mediante:

$$\left(\begin{array}{cc|c} 0.375 & 0.271 & -0.00360 \\ 0.491 & 0.381 & -0.000400 \end{array}\right) \begin{array}{c} m_{21} = 1.31 \\ \rightarrow \end{array} \left(\begin{array}{cc|c} 0.375 & 0.271 & -0.00360 \\ 0.000 & 0.0260 & 0.000432 \end{array}\right),$$

de donde obtenemos que $e_1^{(0)} = -0.130$, $e_2^{(0)} = 0.166$. La solución mejorada queda ahora como:

$$x^{(1)} = \begin{pmatrix} 15.3 \\ -18.9 \end{pmatrix} + \begin{pmatrix} -0.130 \\ 0.166 \end{pmatrix} = \begin{pmatrix} 15.2 \\ -18.7 \end{pmatrix},$$

lo cual en este caso nos produce la solución exacta a tres cifras. En general el proceso se puede repetir una o dos veces mas. □

3.7 Propagación de Errores en la Solución de Sistemas Lineales

Cuando usamos eliminación gaussiana para resolver el sistema $Ax = b$ en una computadora normal, ocurren errores de redondeo en cada paso del proceso debido a la aritmética finita de la máquina. Queremos estudiar como estos errores se propagan o acumulan durante todo el proceso y estimar el error

en la solución calculada. En realidad al final del proceso de eliminación gaussiana obtenemos un vector $\hat{x}$ tal que $A\hat{x} \approx b$. Para medir el error en $\hat{x}$ como aproximación de x procedemos en dos etapas: primero vamos a caracterizar a $\hat{x}$ como la solución exacta de otro problema que esté cerca del problema original en cierto sentido (*análisis de errores revertido*), y luego combinando este resultado con los de la Sección 3.5.1, obtenemos el estimado del error en $\hat{x}$.

Primero repasamos algunas de las fórmulas del Capítulo (2). En particular recordamos que las operaciones aritméticas de la computadora se modelan suponiendo que éstas corresponden a la operación exacta pero truncada al número de cifras de la máquina. Esto es, si $\circ^*$ denota la operación aritmética $\circ$ en la computadora, entonces

$$a \circ^* b = \text{fl}(a \circ b) = (a \circ b)(1 + \varepsilon), \quad \circ \in \{+, -, \times, \div\}, \tag{3.36}$$

donde $|\varepsilon| \leq c\beta^{1-t}$, $c = 1/2, 1$ para el sistema de punto flotante (2.2) y para el estándar de la IEEE (2.3).

Antes de examinar los efectos de la aritmética finita en el proceso de eliminación Gaussiana para un sistema $n \times n$, vamos a examinar el caso de un sistema triangular 2×2. Esto es, consideramos el sistema:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{22}x_2 &= b_2. \end{aligned} \tag{3.37}$$

La solución exacta de este sistema es:

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22}}, \quad x_2 = \frac{b_2}{a_{22}}. \tag{3.38}$$

La solución calculada usando aritmética de punto flotante está dada por:

$$\hat{x}_2 = \text{fl}\left(\frac{b_2}{a_{22}}\right), \quad \hat{x}_1 = \text{fl}\left(\frac{\text{fl}(b_1 - \text{fl}(a_{12}\hat{x}_2))}{a_{11}}\right). \tag{3.39}$$

Usando (3.36) tenemos que:

$$\hat{x}_2 = \frac{b_2}{a_{22}}(1 + \varepsilon_1), \tag{3.40a}$$

$$\hat{x}_1 = \left[\frac{a_{22}b_1 - a_{12}b_2(1 + \varepsilon_1)(1 + \varepsilon_2)}{a_{11}a_{22}}\right](1 + \varepsilon_3)(1 + \varepsilon_4), \tag{3.40b}$$

donde $|\varepsilon_i| \leq \beta^{1-t}$, $i = 1, 2, 3, 4$. Note que podemos escribir (3.40) como:

$$\hat{x}_1 = \frac{\hat{a}_{22}b_1 - \hat{a}_{12}b_2}{\hat{a}_{11}\hat{a}_{22}}, \quad \hat{x}_2 = \frac{b_2}{\hat{a}_{22}}, \tag{3.41}$$

donde

$$\hat{a}_{11} = \frac{a_{11}}{(1+\varepsilon_3)(1+\varepsilon_4)}, \quad \hat{a}_{12} = a_{12}(1+\varepsilon_2), \quad \hat{a}_{22} = \frac{a_{22}}{1+\varepsilon_1}. \tag{3.42}$$

Se puede ver ahora que

$$\begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} \\ 0 & \hat{a}_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} + \hat{H},$$

donde $\|H\|_\infty \leq c \|A\|_\infty \beta^{1-t}$, para una constante c. En resumen tenemos que la solución calculada o aproximada (3.39), corresponde a la solución exacta de un sistema cuya matriz de coeficiente (3.42) está "bién" cerca de la matriz de coeficientes del sistema original.

Usando las fórmulas (3.36) repetidamente en (3.13), (3.16), un análisis similar se puede hacer en el caso general, obteniéndose así el siguiente resultado. (Ver [28]).

Teorema 3.33. *Sea $\hat{x}$ la solución calculada del sistema $Ax = b$ mediante eliminación gaussiana usando aritmética de punto flotante base β con una mantisa de t cifras. Sea u el épsilon de la máquina. Entonces $\hat{x}$ es la solución exacta del sistema $(A + H)\hat{x} = b$ donde la matriz H satisface*

$$\|H\|_\infty \leq C(n^3 + 3n^2)\rho u \|A\|_\infty . \tag{3.43}$$

Aquí n es el tamaño de A, C es una constante independiente de n y la matriz A, y

$$\rho = \frac{\max_{i,j,k} \left|a_{ij}^{(k)}\right|}{\max_{i,j} |a_{ij}|}. \tag{3.44}$$

El factor ρ del teorema mide la razón de crecimiento de las entradas de la matriz de coeficientes según el método numérico progresa. La cota superior de este factor depende de la estrategia de pivoteo que se use. En particular se puede demostrar que

$$\begin{aligned} \rho &\leq 2^{n-1}, \quad \text{para pivoteo parcial,} \\ \rho &\leq 1.8n^{(\ln n)/4}, \quad \text{para pivoteo completo o total.} \end{aligned} \tag{3.45}$$

En cualquiera de los dos casos, si n no es muy grande, el teorema lo que establece es que $\hat{x}$ es solución exacta de un problema que esta cerca (proporcional al épsilon de la maquina) del original. Esto garantiza que el método numérico no introduce más error que el presente en los datos iniciales A, b. En este sentido es que decimos que eliminación gaussiana es un método *estable*. Esto no implica que $\hat{x}$ esté cerca de la solución exacta x. De hecho si tomamos $\hat{A} = A + H$ en el Teorema 3.24 y usamos las cotas del Teorema 3.33 obtenemos que

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq \frac{\kappa_\infty(A)}{1 - \|A^{-1}\|_\infty \|H\|_\infty} \, 1.01(n^3 + 3n^2)\rho u, \tag{3.46}$$

lo que dice que $\hat{x}$ estará cerca de x si A es una matriz bien acondicionada. Por el contrario, si A es mal acondicionada, entonces la solución calculada $\hat{x}$ podría estar lejos de x pero de todos modos $\hat{x}$ sería solución exacta de un sistema cerca del original.

Podemos resumir los resultados más importantes de esta sección con lo siguiente:

1. En general el método de eliminación gaussiana no debe ser utilizado para matrices o sistemas de tamaño n muy grande. (Hay sus excepciones a esto como en el caso de los sistemas tri–diagonales.) Para valores de n grande se utilizan los llamados *métodos iterativos* que son más eficientes y que no sufren tanto de la propagación de errores.

2. El numero de de condición de la matriz debe ser monitoreado durante el proceso ya que puede ser indicativo de la perdida de cifras significativa en los cómputos.

3. El pivoteo juego un rol esencial para el control de la propagación de errores en el método de eliminación gaussiana.

3.8 El Teorema de Gerschgorin

El estimado más crudo de los valores propios de una matriz esta dado por la desigualdad $\rho(A) \leq \|A\|$, donde $\rho(A)$ es el radio espectral de A y $\|\cdot\|$ es cualquier norma matricial. Este estimado aunque útil en muchas ocasiones, no es muy preciso en cuanto a la localización de los valores propios de A.

El Teorema de Gerschgorin va más allá en este sentido. Para $A = (a_{ij})$ definimos los radios

$$r_i = \sum_{j \neq i} |a_{ij}|, \quad 1 \leq i \leq n, \tag{3.47}$$

y los discos

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq r_i\}, \quad 1 \leq i \leq n. \tag{3.48}$$

El Teorema de Gerschgorin establece que cada valor propio de A pertenece al menos a uno de los D_i y que si k de los discos de Gerschgorin se intersecan entre si y a la vez están aislados de los otros discos, entonces su unión contiene exactamente k de los valores propios de A. (Ver [2], [21], [28]).

Teorema 3.34. *Sea A una matriz $n \times n$ y defina los discos D_i por (3.47), (3.48). Entonces*

$$\lambda \in \bigcup_{i=1}^{n} D_i, \tag{3.49}$$

para cualquier valor propio λ de A. Además si $\mathcal{S}$ es la unión de m discos los cuales son disjuntos de los restantes $n-m$, entonces $\mathcal{S}$ contiene exactamente m valores propios de A.

Como A y A^t tienen los mismos valores propios, el Teorema de Gerschgorin también es cierto para los *discos columnas*:

$$R_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{k \neq i} |a_{ki}| \right\}. \tag{3.50}$$

Ejemplo 3.35. Considere la matriz

$$A = \frac{1}{16} \begin{pmatrix} -8 & -2 & 4 \\ -1 & 6 & 2 \\ 2 & 2 & -10 \end{pmatrix}.$$

Note que $\|A\|_1 = (1/16)\max\{14, 9, 14\} = 7/8$, de modo que los valores propios de A cumplen con $|\lambda| \leq 7/8$. Podemos mejorar este estimado con el Teorema de Gerschgorin. Tenemos aplicando (3.47) que: $r_1 = 3/8$, $r_2 = 3/16$, $r_3 = 1/4$. Los discos (3.48) son

$$\begin{aligned} D_1 &= \{z \in \mathbb{C} : |z + 1/2| \leq 3/8\}, \\ D_2 &= \{z \in \mathbb{C} : |z - 3/8| \leq 3/16\}, \end{aligned}$$

$$D_3 \;=\; \{z \in \mathbb{C} \,:\, |z + 5/8| \leq 1/4\}\,,$$

los cuales se ilustran en Figura 3.2. Así que tenemos exactamente dos valores propios en $D_1 \cup D_3$ y exactamente uno en D_2. Este último valor propio tiene que ser real ya que como A tiene entradas reales, el polinomio característico de A tiene coeficientes reales, y por consiguiente las raíces complejas vienen en pares conjugados. Así que tenemos un valor propio $\lambda \in [3/16, 9/16]$. Como el cero no esta en ninguno de los discos podemos concluir que cero no es valor propio de A, en particular A es invertible. Estos resultados se pueden refinar un poco más aplicando el teorema a la matriz transpuesta (cf. (3.50)). □

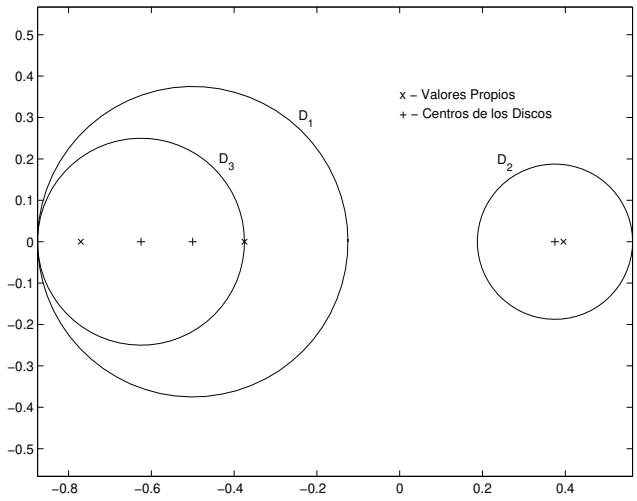


Figura 3.2: Discos de Gerschgorin para la matriz del Ejemplo 3.35.

El Teorema de Gerschgorin garantiza que cada valor propio de la matriz pertenece al menos a uno de los discos (3.48) pero pueden haber discos que no contengan valores propios. Claro por la segunda parte del teorema, estos discos *vacíos* no pueden estar disjuntos de los otros discos, i.e., se tienen que solapar con algún otro disco no vacío.

Ejemplo 3.36. Considere la matriz

$$A = \begin{pmatrix} \varepsilon & 1+\eta \\ -1+\eta & -\varepsilon \end{pmatrix},$$

donde $\varepsilon^2 + \eta^2 < 1$, $\eta \neq 0$. Los valores propios de esta matriz son $\lambda = \pm i\sqrt{1-\varepsilon^2-\eta^2}$ y los discos de Gerschgorin correspondientes:

$$D_1 = \{z \in \mathbb{C} : |z - \varepsilon| \leq 1 + \eta\}, \quad D_2 = \{z \in \mathbb{C} : |z + \varepsilon| \leq 1 - \eta\}.$$

Note que la distancia de los valores propios de A a cualquiera de los centros de estos discos es $\varepsilon^2 + (1 - \varepsilon^2 - \eta^2) = 1 - \eta^2$ que es mayor a $1 + \eta$ si $\eta < 0$, o es mayor que $1 - \eta$ cuando $\eta > 0$. (Note que además $-1 < \eta < 1$). Así que en cualquiera de los dos casos, uno de los discos no contiene ninguno de los dos valores propios. En la Figura 3.3 ilustramos el caso $\varepsilon = 0.7$, $\eta = -0.2$. □

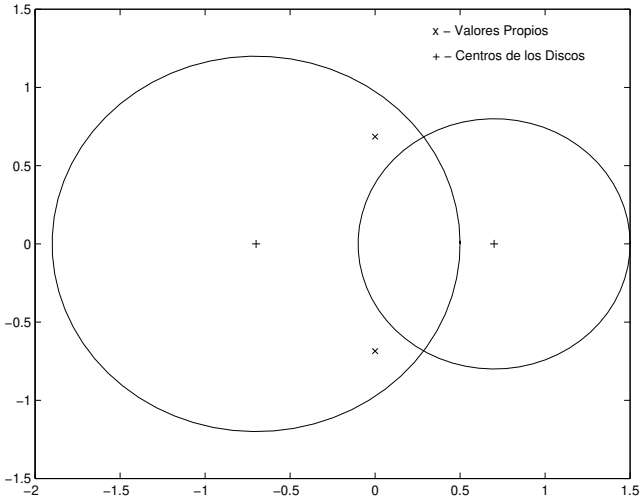


Figura 3.3: Discos de Gerschgorin y valores propios para la matriz del Ejemplo 3.36 con $\varepsilon = 0.7$ y $\eta = -0.2$. Note que uno de los discos no contiene ningún valor propio.

3.9 Ejercicios

Ejercicio 3.1. Para cualquier vector $x = (x_1, x_2, \ldots, x_n)$ y $1 \leq p \leq \infty$, verifique que $\|\cdot\|_p$ tiene las siguientes propiedades:

i) $\|x\|_p \geq 0$, y $\|x\|_p = 0$ si y solo si $x = 0$.

ii) $\|\alpha x\|_p = |\alpha| \, \|x\|_p$ para cualquier $\alpha \in \mathbb{R}$.

iii) $\|x + y\|_p \leq \|x\|_p + \|y\|_p$ para cualesquiera vectores x, y.

iv) $\lim_{p \longrightarrow \infty} \|x\|_p = \|x\|_\infty$.

Ejercicio 3.2. Verifique que para cualesquiera dos vectores $x, \hat{x}$, y $1 \leq p \leq \infty$, tenemos que

$$\frac{\|x - \hat{x}\|_p}{\|x\|_p} \leq \max_i \left\{ \left| \frac{x_i - \hat{x}_i}{x_i} \right| \right\},$$

donde se asume que los componentes de x sos todos distintos de cero.

Ejercicio 3.3. Sea A una matriz $n \times n$ y λ un valor propio de A. Verifique las siguientes:

a) $|\lambda| \leq \|A\|_p$ y por consiguiente que $\rho(A) \leq \|A\|_p$.

b) Si A es nosingular, entonces todos sus valores propios son distintos de cero.

c) Si A es nosingular, entonces los valores propios de A^{-1} son los recíprocos de los de A.

Ejercicio 3.4. Si dos matrices tienen el mismo polinomio característico (cf. (3.3)), entonces éstas tienen los mismos valores propios. Usando esto y las propiedades de la función det, verifique que:

a) A y A^t tienen los mismos valores propios.

b) Si A y B son *similares*, es decir $B = T^{-1}AT$ para alguna matriz nosingular T, entonces A y B tienen los mismos valores propios.

Ejercicio 3.5. Verifique que si A es triangular (superior o inferior), entonces sus valores propios son los elementos de la diagonal principal, i.e., a_{ii}, $1 \leq i \leq n$. Aunque el resultado del método de eliminación gaussiana es transformar la matriz original del sistema a una triangular superior, no podemos utilizar este método para calcular los valores propios de una matriz porque las operaciones elementales de fila no preservan los valores propios. Construya tres ejemplos, uno para cada operación elemental de fila, donde los valores propios de una matriz cambian luego de aplicar la operación elemental de fila correspondiente.

Ejercicio 3.6. Haga un conteo operacional para el total de multiplicaciones al multiplicar dos matrices si:

a) las matrices son $m \times n$ y $n \times p$ respectivamente;

b) ambas matrices son $n \times n$;

c) una es $n \times n$ y la otra es $n \times 1$, i.e., matriz por vector;

d) una es $n \times n$ tridiagonal y la otra es $n \times 1$;

e) una es $n \times n$ triangular inferior y la otra es $n \times 1$.

Ejercicio 3.7. Usando pivoteo parcial resuelva el sistema

$$\left\{\begin{array}{rcr} x - y + 2z & = & -2, \\ -2x + y - z & = & 2, \\ 4x - y + 2z & = & 1. \end{array}\right.$$

Ejercicio 3.8. Al resolver un sistema del tipo

$$\begin{pmatrix} 10 & -1 & 1 & -1 & 1 \\ -1 & 5 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ -1 & 0 & 0 & 5 & 0 \\ 1 & 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 42.8 \\ 1.5 \\ 9.1 \\ 12.5 \\ 4.7 \end{pmatrix},$$

es conveniente intercambiar primero las filas uno y cinco y luego las columnas uno y cinco (¿por qué?). Resuelva el sistema luego de permutar y calcúle la factorización LU de la matríz permutada.

Ejercicio 3.9. Sin usar pivoteo alguno, halle la factorización LU de la matriz

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & -1 \\ -1 & -2 & 3 \end{pmatrix}.$$

Ejercicio 3.10. Considere el sistema lineal

$$\left\{\begin{array}{rcr} x + 3y - z & = & 4, \\ 5x - 2y - z & = & -2, \\ 2x + 2y + z & = & 9. \end{array}\right.$$

a) Halle la factorización LU de la matriz de coeficientes de este sistema.

b) Usando la factorización LU de la primera parte, resuelva el sistema y calcule el determinante de la matriz de coeficientes.

Ejercicio 3.11. Escriba un programa en MATLAB para calcular la inversa de una matriz mediante las ecuaciones (3.20).

Ejercicio 3.12. Sean A, B matrices $n \times n$ donde A es nosingular. Describa un procedimiento para calcular el producto $A^{-1}B$ sin tener que calcular A^{-1}. ¿Cuál es el orden computacional (multiplicaciones), cuando n es "grande", para éste procedimiento? Compare éste resultado con el conteo correspondiente para el método que calcula la inversa de A primero, y luego hace el producto matricial.

Ejercicio 3.13. Resolver un sistema $Ax = b$ por Eliminación Gaussiana donde A es $n \times n$ tridiagonal, toma $O(n)$ operaciones. ¿Cuál es el orden computacional para el computo de A^{-1} en este caso?

Ejercicio 3.14. Sea $M(n)$ el numero de multiplicaciones requeridas para calcular el determinante de una matriz $n \times n$ por el método de cofactores. Verifique que

$$M(n) = \begin{cases} 2! & , \quad n = 2, \\ n\,M(n-1) & , \quad n > 2. \end{cases}$$

Usando inducción matemática verifique ahora que $M(n) = n!$ para $n \geq 2$.

Ejercicio 3.15. Se desea resolver un sistema $Ax = b$ donde A es $n \times n$, tridiagonal y $n = 100$. Por ciertas limitaciones de la computadora o paquete de computación utilizados, sólo se permite trabajar con matrices de tamaño máximo 32×32. ¿Se puede resolver este problema con las limitaciones dadas? Explique.

Ejercicio 3.16. Halle la factorización de Cholesky de las siguientes matrices. Usando esta factorización y el vector b dado, halle la solución del sistema $Ax = b$.

a) $A = \begin{bmatrix} 4 & -10 & 2 \\ -10 & 34 & -17 \\ 2 & -17 & 18 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ -2 \\ 3 \end{bmatrix}.$

b) $A = \begin{bmatrix} 4 & 2 & -2 \\ 2 & 10 & 2 \\ -2 & 2 & 5 \end{bmatrix}$, $b = \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix}$.

Ejercicio 3.17. Lleve a cabo el conteo operacional para el Algoritmo (3.14) del Método de Cholesky.

Ejercicio 3.18. Si A es simétrica y positiva definida, utilice la factorización de Cholesky de A para describir en forma matricial las tres etapas del proceso de eliminación gaussiana: eliminación, modificación del lado derecho, y sustitución para atrás.

Ejercicio 3.19. Defina una estructura escasa (*sparse*) en MATLAB para la matriz

$$B = \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ -7 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}.$$

Para `b=rand(5,1)`, resuelva el sistema `Bx=b` usando las representaciones densa y escasa de B. Usando la instrucciones `tic` y `toc` de MATLAB obtenga un estimado del tiempo de ejecución en cada caso.

Ejercicio 3.20. Considere el sistema

$$\begin{cases} 2x_1 & & -x_3 & & & & = & 1 \\ & 2x_2 & & -x_4 & & & = & 1 \\ x_1 & & +2x_3 & & -x_5 & & = & 1 \\ & x_2 & & +2x_4 & & -x_6 & = & 1 \\ & & x_3 & & +2x_5 & & = & 1 \\ & & & x_4 & & +2x_6 & = & 1 \end{cases}$$

a) Describa una estructura escasa en MATLAB para la matriz de coeficientes del sistema.

b) Escriba las instrucciones de MATLAB adicionales para resolver el sistema con la estructura de la parte (a).

Ejercicio 3.21. Dado que

$$A = \begin{pmatrix} 1 & -1 & -1 & -1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 5.00 \\ 1.02 \\ 1.04 \\ 1.10 \end{pmatrix},$$

calcule una solución aproximada de $Ax = b$ primero redondeando las entradas de b al entero más cercano y luego resolviendo el sistema resultante. Verifique que la solución calculada es $\hat{x} = (12, 4, 2, 1)^t$. Sea $r = b - A\hat{x}$ el vector residual.

a) Determine los valores de $\|r\|_\infty$ y $\kappa_\infty(A)$.

b) Utilize su respuesta de la primera parte para hallar una cota superior para el error relativo en la solución aproximada.

c) Calcule la solución exacta x y determine el error relativo exacto $\|x - \hat{x}\|_\infty / \|x\|_\infty$.

Ejercicio 3.22. Defina los siguientes:

```
A=round(10*rand(6));
s=ones(6,1);
b=A*s;
```

La solución al sistema lineal $Ax = b$ es claramente s. Resuelva el sistema utilizando la operación \ de MATLAB y sea x la solución calculada. Calcule el error $x - s$. Ahora perturbamos el sistema de la forma siguiente: deje que

```
t=1.0e-12;
E=rand(6)-0.5;
r=rand(6,1)-0.5;
```

y sean

```
M=A+t*E;
c=b+t*r;
```

Resuelva el nuevo sistema perturbado $Mz = c$ para z. Compare la solución z con la solución del sistema original calculando $z - s$. ¿Cómo compara el tamaño de la perturbación en la solución con el tamaño de las perturbaciones en A y b? Repita el análisis de perturbación con `t=1.0e-04` y `t=1.0e-02`. ¿Es o no el sistema $Ax = b$ bien acondicionado? Explique. Utilice MATLAB para calcular el número de condición de A.

Ejercicio 3.23. Considere el sistema

$$\begin{cases} 20x + 99y &= 218, \\ 101x + 500y &= 1101. \end{cases}$$

a) Decida si el sistema es mal acondicionado o no.

b) Si el lado derecho se cambia a $(217, 1100)^t$, halle un estimado del error relativo en la solución del nuevo sistema comparada con la solución del sistema original. (No resuelva ninguno de los dos sistemas).

Ejercicio 3.24. Sea $\varepsilon > 0$ un número pequeño y considere la matriz

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

a) Calcule el número de condición de A, i.e., $\kappa_\infty(A)$.

b) Basado en el cálculo de la primera parte, decida si el sistema lineal $Ax = b$ es o no mal acondicionado según $\varepsilon \to 0$.

c) Si $\varepsilon = 10^{-5}$ y los datos iniciales A, b tienen errores relativos de 10^{-12}, ¿cuántas cifras correctas pueden esperarse en la solución calculada?

Ejercicio 3.25. Para $n > 1$ defina la matriz A_n por:

$$A_n = \begin{pmatrix} 1 & 2 \\ 2 & 4 + \frac{1}{n^2} \end{pmatrix}.$$

a) Calcúle el número de condición de A_n en la norma $\|\cdot\|_\infty$.

b) Decida si $A_n x = b$ es un sistema mal acondicionado o no.

c) Queremos resolver $A_n x = b$ donde $b = (1, 2)^t$. Si en lugar de b, resolvemos con $b_n = (1, 2 - 1/n^2)^t$, halle un estimado para el error relativo en la solución del sistema con b_n.

Ejercicio 3.26. Calcule el número de condición para la matriz

$$A = \begin{pmatrix} 1 & -0.99 \\ -1 & 1 \end{pmatrix},$$

en la norma $\|\cdot\|_\infty$. Aproximadamente, ¿cuántas cifras decimales podrían perderse al resolver un sistema con esta matriz?

Ejercicio 3.27. Queremos resolver el sistema $Ax = b$ donde el único error en b es debido a truncación al entrarlo en la computadora, y A se conoce con un error relativo de 10^{-10}. Si el epsilon de la máquina es de 10^{-16}, ¿cúanto es lo más grande que puede ser el número de condición del sistema para que el error relativo en la solución calculada sea a lo más 10^{-5}?

Ejercicio 3.28. Sea $x = (x_1, x_2)$ la solución exacta del sistema $Ax = b$ donde

$$A = \begin{pmatrix} 0.001 & 0 \\ 1 & 10000 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

Sea $\hat{x} = (\hat{x}_1, \hat{x}_2)$ la solución exacta del sistema $(A + H)\hat{x} = b$ donde $H = \text{diag}(\varepsilon_1, \varepsilon_2)$. Defina el error relativo en $\hat{x}$ como aproximación de x por:

$$e_r = \left(\frac{\hat{x}_1 - x_1}{x_1}, \frac{\hat{x}_2 - x_2}{x_2} \right).$$

Verifique que si $\varepsilon_1 = 0.001u$, $\varepsilon_2 = 10000u$ (de modo que $\|H\| \leq \|A\| \, u$), entonces:

$$\begin{aligned} e_r &= \left(\frac{-u}{1+u}, \frac{u(-b_2 + 2000b_1 + 1000b_1 u - ub_2)}{(1 + 2u + u^2)(1000b_1 - b_2)} \right), \\ &= \left(-u + O(u^2), -\frac{(2000b_1 - b_2)u}{1000b_1 - b_2} + O(u^2) \right). \end{aligned}$$

¿Qué puede concluir ahora sobre el acondicionamiento del sistema con respecto a este tipo de perturbaciones en la matriz A? (El caso en que $1000b_1 - b_2 = 0$ debe ser analizado en forma separada usando el error absoluto en $\hat{x}_2$ en lugar del relativo.)

Ejercicio 3.29. Haga una análisis similar al del Ejercicio 3.28 para la matriz de coeficientes del sistema en el Ejemplo 3.27.

Ejercicio 3.30. Calcule la factorización LU de la matríz

$$A = \begin{pmatrix} 20 & 3 & 4 \\ 3 & 40 & 5 \\ 4 & 5 & 60 \end{pmatrix},$$

usando aritmética finita con dos cifras decimales. El sistema $Ax = b$ donde $b = (15, -360, 420)^t$ tiene la solución aproximada $x^{(0)} = (0.65, -9.8, 7.8)^t$. Calcule una mejor aproximación mediante un paso de refinamiento iterativo.

Ejercicio 3.31. El vector $x^{(0)} = (1.1, 0.88)^t$ es una solución aproximada del sistema $Ax = b$ donde

$$A = \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 2 \end{pmatrix}.$$

Usando aritmética de dos cifras decimales y un paso del método de residuos, encuentre una solución mejorada para el sistema.

Ejercicio 3.32. Defina

$$A_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & -1 \\ -1 & 1 & 1 & 0 & 1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \end{pmatrix},$$

y sea A_n la matriz correspondiente de orden n. Demuestre que el factor de crecimiento ρ definido en (3.43) es 2^{n-1} para eliminación gaussiana con pivoteo parcial.

Ejercicio 3.33. Para las matrices

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 10 \end{pmatrix}, \quad \begin{pmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{pmatrix}, \quad \begin{pmatrix} 5 & 2 & 0 \\ 1 & -6 & 0 \\ 0 & 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 5 & 1 & 1 \\ 0 & 6 & 1 \\ 1 & 0 & -5 \end{pmatrix},$$

utilize el Teorema de Gerschgorin para estimar los valores propios de cada matriz. Obtenga los estimados más exactos que pueda (aplique el teorema también a la matriz transpuesta).

Ejercicio 3.34. En este ejercicio mostramos como se hacen las Figuras 3.2 o 3.3. Defina

```
A=round(10*rand(5))+sqrt(-1)*round(10*rand(5));
```

Calcule el radio de los discos de Gerschgorin de A y guárdelos en un vector r. Para trazar los discos usamos el parámetro `t=[0:0.1:6.3]'`. Podemos generar dos matrices X y Y cuyas columnas contengan las coordenadas (x, y) de los discos. Primero iniciaremos X y Y a cero:

```
X=zeros(length(t),5); Y=X;
```

Las matrices X y Y se generan utilizando los siguientes comandos de MATLAB:

```
for i=1:5
    X(:,i)=r(i)*cos(t)+real(A(i,i));
    Y(:,i)=r(i)*sin(t)+imag(A(i,i));
end
```

Sea `e=eig(A)`. Podemos ahora trazar los valores propios y los discos de Gerschgorin con el comando

```
plot(X,Y,real(e),imag(e),'x')
```

Ejercicio 3.35. Defina

```
B = [3,0.1,2;0.1,7,2;2,2,50];
```

a) Utilice el método descrito en el problema (3.34) para calcular y trazar los discos Gerschgorin de B.

b) Como B es simétrica, sus valores propios son todos reales y por esto deben estar en el eje real. Sin calcular los valores propios, explique porqué B debe tener exactamente un valor propio en el intervalo $[46, 54]$. Multiplique las primeras dos filas de B por 0.1 y luego multiplique las primeras dos columnas por 10. Esto se logra mediante la transformación de similitud

```
D=diag([0.1, 0.1, 1]); C=D*B*inv(D);
```

La nueva matriz C tiene los mismos valores propios que B. ¿Por qué? Utilice C para hallar intervalos que contengan los otros dos valores propios de B. Calcule y trace los discos de Gerschgorin de C. ¿Cómo comparan los estimados de los valores propios usando C con los originales?

Ejercicio 3.36. Para el polinomio $p(x) = a_1 + a_2x + \cdots a_nx^{n-1}$, $a_n \neq 0$, defina su *matriz compañera* por:

$$A_p = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\dfrac{a_1}{a_n} & -\dfrac{a_2}{a_n} & -\dfrac{a_3}{a_n} & \cdots & -\dfrac{a_{n-2}}{a_n} & -\dfrac{a_{n-1}}{a_n} \end{pmatrix}.$$

Demuestre que los valores propios de A_p coinciden exactamente con las raíces de $p(x)$. Usando el Teorema de Gerschgorin (Teorema 3.34) concluya ahora que $|\alpha| \leq K$ donde α es cualquier raíz de $p(x)$ y

$$K = 1 + \max\left\{ \left| \frac{a_k}{a_n} \right| \,:\, 1 \leq k < n \right\}.$$

Ejercicio 3.37. Demuestre que productos e inversos de matrices de la forma (3.21) (triangulares unitarias) tienen otra vez la forma (3.21).

Ejercicio 3.38. Considere el sistema $Lg = e_i$ donde L es una matriz $n \times n$ triangular inferior con unos en la diagonal y e_i es la i–esima columna de la matriz identidad $n \times n$. La matriz L se puede particionar en bloques de la forma siguiente:

$$L = \begin{bmatrix} L_i & O_i \\ M_i & \hat{L}_i \end{bmatrix},$$

donde L_i es una matriz $i \times i$ triangular inferior con unos en la diagonal, O_i es una matriz $i \times (n-i)$ de ceros, M_i es una matriz $(n-i) \times i$, y $\hat{L}_i$ es una matriz $(n-i) \times (n-i)$ triangular inferior con unos en la diagonal.

a) Verifique que si $g = (g_1, \ldots, g_n)^t$ es la solución de $Lg = e_i$, entonces $g_1 = g_2 = \cdots = g_{i-1} = 0$ y que $g_i = 1$.

b) Argumente ahora que el sistema original $Lg = e_i$ se puede resolver con aproximadamente $\frac{1}{2}(n-i)^2$ multiplicaciones o divisiones. **Ayuda:** Si particionamos a g como $g = (x, y)^t$, donde $x = (g_1, g_2, \ldots, g_i)$ y $y = (g_{i+1}, \ldots, g_n)$, entonces el sistema original es equivalente a $\hat{L}_i y^t = -M_i x^t$. Note que por el resultado de la parte (a), $M_i x^t$ es igual a la última columna de M_i.

Ejercicio 3.39. Suponga que $A = LU$ es la factorización LU de la matriz A, por lo que $A^{-1} = U^{-1}L^{-1}$.

a) Usando el resultado del Ejercicio 3.38 verifique que L^{-1} se puede calcular con $\frac{n^3}{6}$ multiplicaciones o divisiones aproximadamente para n grande.

b) Usando el resultado del Ejercicio 3.38 verifique que U^{-1} se puede calcular con $\frac{n^3}{6}$ multiplicaciones o divisiones aproximadamente para n grande. **Ayuda:** Utilice que como $(U^t)^{-1} = (U^{-1})^t$, entonces $U^{-1} = ((U^t)^{-1})^t$.

c) Verifique que como U^{-1} y L^{-1} son triangular superior e inferior respectivamente, entonces el producto $U^{-1}L^{-1}$ se puede calcular con $\frac{n^3}{3}$ multiplicaciones o divisiones aproximadamente para n grande. **Ayuda:** Si escribimos $U^{-1} = (\mu_{ij})$, $L^{-1} = (\gamma_{ij})$, y $U^{-1}L^{-1} = (\alpha_{ij})$, verifique que $\alpha_{ij} = \sum_{k=j}^{n} \mu_{ik}\gamma_{kj}$ si $i < j$, mientras que $\alpha_{ij} = \sum_{k=i}^{n} \mu_{ik}\gamma_{kj}$ si $j < i$.

d) Concluya ahora que $A^{-1} = U^{-1}L^{-1}$ se puede calcular con n^3 multiplicaciones o divisiones aproximadamente para n grande.

Ejercicio 3.40. Considere un sistema $n \times n$ *complejo* de la forma

$$(A + iB)(u + iv) = b + ic,$$

donde A, B, u, v, b, c son matrices con entradas reales de las dimensiones apropiadas. Escriba este sistema como uno real de tamaño $2n \times 2n$. Compare el conteo operacional del método de eliminación gaussiana aplicado a ambos sistemas suponiendo que una multiplicación de números complejos equivale a cuatro multiplicaciones de números reales y dos sumas. Haga lo mismo para divisiones.

Ejercicio 3.41. Una matriz H se llama *Hessenberg* si $h_{ij} = 0$ para toda $i > j + 1$. ¿Cúantas operaciones requiere resolver $Hx = b$ mediante eliminación gaussiana? Si H se normaliza de modo que $|h_{ij}| \leq 1$ para todo i, j y se usa pivoteo parcial en eliminación gaussiana, demuestre que las entradas de U (c.f. (3.14)) permanecen menor de n en magnitud donde n es el tamaño de H.

Capítulo 4

Solución de Ecuaciones Nolineales

Suponga que $f : \mathbb{R} \to \mathbb{R}$ es una función continua. Deseamos calcular soluciones o lo mismo raíces de la *ecuación escalar nolineal*,

$$f(x) = 0. \tag{4.1}$$

Más general aún, podemos considerar el caso en que $f : \mathbb{R}^n \to \mathbb{R}^n$ donde $n \geq 1$. En este caso (4.1) representa un *sistema de ecuaciones nolineales*. Problemas del tipo (4.1) son comunes por ejemplo en la optimización de funciones y en la solución numérica de problemas de frontera nolineales. En este capítulo enfocaremos más en el problema escalar y discutiremos los aspectos básicos del caso de sistemas de ecuaciones.

4.1 Método de la Bisección

Este método tiene como base o motivación el Teorema del Valor Intermedio (Teorema A.3). En particular, si $f : \mathbb{R} \to \mathbb{R}$ es una función continua, y a, b son números dados tales que $f(a)f(b) \leq 0$, entonces definimos $c = (a+b)/2$. Si $f(c) = 0$, entonces terminamos ya que c sería una raíz. De lo contrario reemplazamos a o b con c manteniendo la diferencia en signos, etc.. Esto nos lleva al siguiente pseudo algoritmo llamado el *Método de la Bisección*:

Algoritmo 4.1. Suponga que $f : \mathbb{R} \to \mathbb{R}$ es una función continua y que a, b son números dados tales que $f(a)f(b) \leq 0$, y $\varepsilon > 0$ es también dado (*criterio de convergencia*).

1. Sean $a_1 = a$, $b_1 = b$, $n = 1$.

2. Defina $c_n = (a_n + b_n)/2$.

3. Si $b_n - a_n \leq \varepsilon$, entonces aceptamos c_n como una raíz aproximada y detenemos el proceso.

4. Si $f(b_n) \cdot f(c_n) \leq 0$, entonces ponemos $a_{n+1} = c_n$, $b_{n+1} = b_n$. De lo contrario ponemos $a_{n+1} = a_n$, $b_{n+1} = c_n$.

5. Ponga $n = n + 1$, y vuelva a (2).

El siguiente programa es una implementación de este algoritmo en MATLAB:

```
function x=mbisect(f,a,b,tol,iterm)
fa=feval(f,a);
fb=feval(f,b);
iter=1;
while (b-a>=tol)&&(iter<iterm)
    c=(a+b)/2;
    fc=feval(f,c);
    disp(['iter=' num2str(iter) '   a=' num2str(a)...
          '   c=' num2str(c) '   b=' num2str(b)...
          '   fa=' num2str(fa) '   fc=' num2str(fc)...
          '   fb=' num2str(fb)])
    if fb*fc<=0
        a=c;
        fa=fc;
    else
        b=c;
        fb=fc;
    end
    iter=iter+1;
end
x=c;
```

Ejemplo 4.2. Considere la función $f(x) = x^2 + x - 1$. Note que como $f(0) = -1$ y $f(1) = 1$, tenemos que existe una raíz de la ecuación $f(x) = 0$ en el intervalo $(0, 1)$. Vamos a aproximar esta raíz con el método de la bisección.

Los siguientes resultados los gensoneramos con el programa `mbisect` descrito antes:

n	a_n	c_n	b_n	$f(a_n)$	$f(c_n)$	$f(b_n)$
1	0	0.5	1	-1	-0.25	1
2	0.5	0.75	1	-0.25	0.3125	1
3	0.5	0.625	0.75	-0.25	0.015625	0.3125
4	0.5	0.5625	0.625	-0.25	-0.12109	0.015625
5	0.5625	0.59375	0.625	-0.12109	-0.053711	0.015625
6	0.59375	0.60938	0.625	-0.053711	-0.019287	0.015625
7	0.60938	0.61719	0.625	-0.019287	-0.0018921	0.015625
8	0.61719	0.62109	0.625	-0.0018921	0.0068512	0.015625
9	0.61719	0.61914	0.62109	-0.0018921	0.0024757	0.0068512
10	0.61719	0.61816	0.61914	-0.0018921	0.00029087	0.0024757

Para estos cálculos utilizamos $\varepsilon = 10^{-4}$ y tenemos que la aproximación a la raíz es 0.61816. La solución correcta a seis cifras, calculada con la función `fzero` de MATLAB, es 0.618034. Podemos ver que el método de la bisección aunque progresa hacia la solución, lo hace un tanto lento. Veamos ahora por qué esto es así luego de analizar la convergencia del método. □

4.1.1 Análisis del Error

Note que los intervalos $\{[a_n, b_n]\}$ generados por el método de la bisección cumplen con que

$$[a_{n+1}, b_{n+1}] \subset [a_n, b_n], \quad n \geq 1. \tag{4.2}$$

Del paso (4) del Algoritmo (4.1) podemos ver que

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n). \tag{4.3}$$

Esto es así ya que el largo del intervalo se divide a la mitad cada vez que pasamos por el paso (2) del Algoritmo (4.1). Usando inducción matemática tenemos ahora que

$$b_n - a_n = \frac{1}{2^{n-1}}(b - a), \quad n \geq 1. \tag{4.4}$$

En particular esto implica que $b_n - a_n \to 0$ según $n \to \infty$. Esto combinado con (4.2) (vea [24], [25]) nos da que

$$\bigcap_{n \geq 1} [a_n, b_n] = \{\alpha\}. \tag{4.5}$$

Veamos que $f(\alpha) = 0$. Como $f(a_n)f(b_n) \leq 0$, existe un $\alpha_n \in [a_n, b_n]$ tal que $f(\alpha_n) = 0$ (Teorema A.3). Ahora (4.5) implica que α pertenece a $[a_n, b_n]$ también, por lo que $|\alpha_n - \alpha| \leq b_n - a_n \to 0$, i.e., $\alpha_n \to \alpha$. Como f es continua y $f(\alpha_n) = 0$ para toda n, entonces $f(\alpha) = 0$. De igual forma como c_n pertenece a $[a_n, b_n]$, entonces $|c_n - \alpha| \leq b_n - a_n \to 0$, i.e., $c_n \to \alpha$.

Para la rapidez de convergencia de la sucesión (c_n) a la raíz α, observe que como $\alpha \in [a_n, c_n]$ o $\alpha \in [c_n, b_n]$ tenemos que

$$|c_n - \alpha| \leq c_n - a_n = b_n - c_n = \frac{1}{2}(b_n - a_n). \tag{4.6}$$

Combinando (4.4) y (4.6) obtenemos que

$$|c_n - \alpha| \leq \frac{1}{2^n}(b - a), \quad n \geq 1, \tag{4.7}$$

esto es, la convergencia de (c_n) a la raiz α es lineal con tasa o razón de convergencia $\frac{1}{2}$. Usando la desigualdad (4.7) podemos estimar el número de iteraciones necesarias para obtener que $|c_n - \alpha| \leq \varepsilon$. En particular, para que $(1/2^n)(b - a) \leq \varepsilon$, necesitamos que

$$n \geq \log_2 \left(\frac{b - a}{\varepsilon} \right). \tag{4.8}$$

El análisis anterior lo podemos resumir con las siguientes observaciones:

- ¡El método tiene convergencia segura! Esto es, dada la condición inicial $f(a)f(b) \leq 0$, el método converge a una raíz α en [a,b]. Esta propiedad conocida como *convergencia global* es una de las características fuertes del método de la bisección y la mayoría de los métodos no la poseen.

- El error en las aproximaciones generadas por el método, medido por el largo del intervalo donde se encuentra la raíz, se reduce por la mitad en cada paso.

- La convergencia lineal caracterizada por la desigualdad (4.7) es lenta comparada con la de otros métodos. Esto es así ya que el método de la bisección solo usa evaluaciones de la función y comparaciones. Veremos que podemos obtener métodos con orden de convergencia más rápido si utilizamos más información de la función f, como por ejemplo sus derivadas.

El método de la bisección no se puede generalizar al caso de sistemas no-lineales debido a que no es posible una generalización del Teorema A.3 a funciones vectoriales de variable vectorial.

4.2 Método de Newton

Suponemos ahora que la función f es diferenciable ($f \in C^1(\mathbb{R})$) e incorporamos información de la derivada en el método. La recta tangente a f en el punto $(x_0, f(x_0))$ está dada por la ecuación $y = f(x_0) + f'(x_0)(x - x_0)$. Ahora *aproximamos* a f con esta recta y definimos a x_1 como el intercepto en x de la recta, esto es:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}, \quad f'(x_0) \neq 0. \tag{4.9}$$

(Ver Figura 4.1). Este proceso lo podemos repetir ahora con el punto

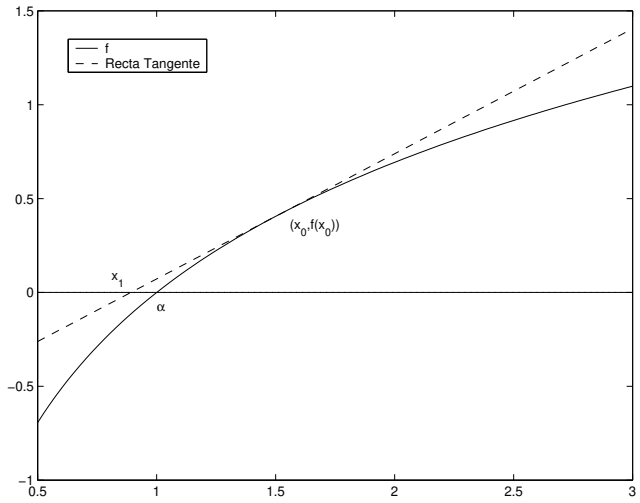


Figura 4.1: Método de Newton: se aproxima la función f con la recta tangente.

$(x_1, f(x_1))$, etc., obteniendo así la recursión

$$\begin{cases} x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}, & n \geq 0, \\ x_0 \text{ dado.} \end{cases} \tag{4.10}$$

Esta recursión define el *Método de Newton* y es un ejemplo de una *iteración de punto fijo*.

Ejemplo 4.3. Considere la función $f(x) = x^2 + x - 1$. Las ecuaciones (4.10) en este caso toman la forma:

$$x_{n+1} = x_n - \frac{x_n^2 + x_n - 1}{2x_n + 1} = \frac{x_n^2 + 1}{2x_n + 1}, \quad n \geq 0.$$

Comenzando con $x_0 = 1$ tenemos los siguientes resultados:

n	x_n	$f(x_n)$
0	1	1
1	0.66667	0.11111
2	0.61905	0.0022676
3	0.61803	1.0265e-006

Aquí podemos apreciar una convergencia más rápida que la del método de la bisección. Más adelante veremos que el método de Newton tiene lo que se conoce como *convergencia cuadrática local* pero en general no tiene la propiedad de convergencia global del método de la bisección. □

4.2.1 Aplicación 1 - Recíprocos

Vamos a suponer que queremos calcular el cociente a/b usando solo las operaciones $\{+, -, \times\}$. En la mayoría de las computadoras, estas operaciones se implementan a nivel de *hardware* mientras que la división se hace mediante programado (*software*) utilizando $\{+, -, \times\}$. Note que como $a/b = a \cdot (1/b)$, entonces es suficiente discutir el cómputo de recíprocos. Si definimos la función $f(x)$ por

$$f(x) = b - (1/x), \quad b > 0, \tag{4.11}$$

entonces $\alpha = 1/b$ es solución de $f(x) = 0$. Usando que $f'(x) = 1/x^2$, el método de Newton para la ecuación $f(x) = 0$ toma la forma:

$$x_{n+1} = x_n(2 - bx_n), \quad n \geq 0, \tag{4.12}$$

la cual utiliza únicamente las operaciones $\{+, -, \times\}$. Es fácil ver que el error absoluto en x_{n+1} como aproximación a $\alpha = 1/b$ está dado por:

$$x_{n+1} - \alpha = -\frac{(x_n - \alpha)^2}{\alpha},$$

de donde obtenemos la fórmula

$$\frac{\alpha - x_{n+1}}{\alpha} = \frac{(\alpha - x_n)^2}{\alpha^2}, \quad n \geq 0, \tag{4.13}$$

i.e.,

$$\text{Rel}(x_{n+1}) = \text{Rel}(x_n)^2, \quad n \geq 0. \tag{4.14}$$

Esta fórmula, exacta en este caso, aplica en general al método de Newton pero asintóticamente según las iteraciones convergen a la raíz (Teorema 4.7). De la ecuación (4.14) obtenemos que $x_n \to \alpha$ según $n \to \infty$ si y solo si $|\text{Rel}(x_0)| < 1$, i.e.,

$$0 < x_0 < \frac{2}{b}. \tag{4.15}$$

Dado que $x_0 > 0$, ¿cómo podemos asegurarnos que $x_0 < (2/b)$ cuando no conocemos $2/b$? Note que si $x_0 > 0$, entonces

$$x_0 \geq (2/b) \Leftrightarrow 2 - bx_0 \leq 0 \Leftrightarrow x_1 = x_0(2 - bx_0) \leq 0.$$

Así que si $x_1 \leq 0$, entonces sabemos que $x_0 \geq (2/b)$. En este caso reducimos x_0, digamos a $x_0/2$, y empezamos las iteraciones de nuevo. Note que despúes de esta etapa inicial, la convergencia es bién rápida, de hecho cuadrática de acuerdo a la ecuación (4.14). Esto es, en cada iteración el número de cifras significativas se duplica. A pesar de esta convergencia rápida cerca de la solución, la etapa inicial descrita arriba puede ser bien lenta aún despúes de que (4.15) se cumpla. Para mejorar esto, escribimos b en binario como $b = f \cdot 2^e$ donde $1/2 \leq f < 1$, entonces $b^{-1} = f^{-1} \cdot 2^{-e}$. Pero $1 < f^{-1} \leq 2$ de modo que $x_0 = 2^{-e}$ cumple con (4.15) y es usualmente una excelente aproximación de $1/b$. El exponente e se puede calcular tomando el entero más pequeño mayor o igual que $\log_2(b)$.

Ejemplo 4.4. Vamos a ilustrar el proceso descrito antes calculando el reciproco de $b = 217$. El siguiente programa utiliza las iteraciones (4.12) y el proceso descrito en el párrafo anterior para estimar el x_0:

```
b=217;
e=ceil(log2(b));
x0=2^(-e);
normx=1;
normz=1;
iter=0;
while normz>eps*normx
   x=x0*(2-b*x0);
   normx=abs(x0);
```

```
   normz=abs(x-x0);
   x0=x;
   iter=iter+1;
end
```

El programa calcula hasta que el error relativo (aproximado) en las iteraciones sea menor que el epsilon de la maquina (`eps`). Los resultados se muestran a continuación:

n	x_n
0	0.00390625
1	0.0045013427734375
2	0.004605812719091773
3	0.004608293593857127
4	0.004608294930875187
5	0.004608294930875576
6	0.004608294930875576

Note que la iteración cinco es el mismo resultado que se obtiene al escribir 1/217 en MATLAB. □

4.2.2 Aplicación 2 - Cálculo de Raíces

Vamos a discutir ahora el problema de calcular raíces cuadradas. Definimos para $a > 0$ la función $f(x) = x^2 - a$ la cual tiene como una de sus raíces a $\sqrt{a}$. Dado que $f'(x) = 2x$ tenemos que el método de Newton aplicado a la ecuación $f(x) = 0$ toma la forma:

$$\begin{cases} x_{n+1} = \dfrac{1}{2}\left(x_n + \dfrac{a}{x_n}\right), \\ x_0 \text{ dado.} \end{cases} \tag{4.16}$$

Este método se conocía o se le llamaba antes la *regla de escuela superior*, ya que era el procedimiento que se le enseñaba a los estudiantes para aproximar raíces cuadradas, cuando no se tenían las calculadoras de mano. No obstante la iteración (4.16) sigue siendo pertinente ya que es el procedimiento que utilizan las calculadoras de mano modernas para aproximar raíces cuadradas.

Ejemplo 4.5. Con $a = 2$ y tomando $x_0 = 2$ tenemos que las primeras cuatro iteraciones en (4.16) son:

n	x_n
1	1.5
2	1.41667
3	1.41422
4	1.41421

Note que ya la cuarta iteración tiene seis cifres correctas como aproximación a $\sqrt{2}$. □

Veamos ahora un resultado sobre la convergencia en general para las iteraciones (4.16).

Teorema 4.6. *La sucesión (x_n) definida por la iteración (4.16) converge a $\sqrt{a}$ para cualquier $x_0 > 0$, y tiene además orden de convergencia dos.*

Demostración: Si $x_0 > 0$ y $a > 0$ entonces sigue de (4.16) que $x_n > 0$ para toda $n \geq 0$. Además,

$$\begin{aligned} x_{n+1} - \sqrt{a} &= \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) - \sqrt{a}, \\ &= \frac{1}{2x_n}(x_n^2 - 2\sqrt{a}x_n + a), \\ &= \frac{1}{2x_n}(x_n - \sqrt{a})^2. \end{aligned} \tag{4.17}$$

De aquí y como $x_n > 0$, obtenemos que $x_n \geq \sqrt{a}$ para toda $n \geq 1$. Ahora (4.16) se puede reescribir como:

$$x_{n+1} - x_n = \frac{a - x_n^2}{2x_n}, \quad n \geq 0. \tag{4.18}$$

Esto combinado con $x_n \geq \sqrt{a}$ para toda $n \geq 1$ implica que $x_{n+1} \leq x_n$ para toda $n \geq 1$. Tenemos pues que (x_n) está acotada inferiormente por $\sqrt{a}$ y es decreciente de modo que es convergente. Falta ver que converge al número $\sqrt{a}$. Sea α el límite de (x_n). Entonces dejando $n \to \infty$ en la ecuación (4.18) obtenemos que $\alpha^2 = a$, i.e., $\alpha = \pm\sqrt{a}$. Pero como $x_n > 0$ para toda $n \geq 0$, tenemos que $\alpha \geq 0$, i.e., $\alpha = \sqrt{a}$. La convergencia cuadrática de la sucesión se obtiene ahora a partir de la expresión (4.17). □

Como la convergencia en (4.16) es de orden dos, esto garantiza que si $x_0 = N$ donde $N^2 \leq a < (N+1)^2$, entonces entre cuatro y seis iteraciones de (4.16) garantizan 16 cifras correctas en la aproximación calculada.

4.2.3 Análisis de Convergencia – Caso General

En las dos aplicaciones discutidas hemos visto que el método de Newton tiene convergencia cuadrática. Esto es cierto en general al menos en una vecindad de la raíz.

Teorema 4.7 (Convergencia Local del Método de Newton)**.** *Suponga que $f : \mathbb{R} \to \mathbb{R}$ es una función C^2 en una vecindad del número α para la cual $f(\alpha) = 0$, $f'(\alpha) \neq 0$. Entonces si x_0 se selecciona suficientemente cerca de α, las iteraciones (4.10) convergen a α. Además*

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{f''(\alpha)}{2f'(\alpha)}, \tag{4.19}$$

es decir, (x_n) converge a α con orden de convergencia $p = 2$.

Demostración: Usando el Teorema de Taylor (Teorema 1.2) podemos escribir

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2, \tag{4.20}$$

donde ξ_n está entre α y x_n. Como $f'(\alpha) \neq 0$ y f' es continua, existe un intervalo cerrado I alrededor de $x = \alpha$ tal que $f'(x) \neq 0$ para $x \in I$. Defina ahora el número M por

$$M = \frac{1}{2}\frac{\max_{x\in I} |f''(x)|}{\min_{x\in I} |f'(x)|}, \tag{4.21}$$

el cual existe y es finito (¿por qué?). Si $x_n \in I$, entonces podemos despejar para una de las α en (4.20) obteniendo así que

$$\alpha = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}\frac{f''(\xi_n)}{f'(x_n)}(\alpha - x_n)^2 = x_{n+1} - \frac{1}{2}\frac{f''(\xi_n)}{f'(x_n)}(\alpha - x_n)^2. \tag{4.22}$$

Definimos ahora el intervalo $\hat{I}$ por

$$\hat{I} = \{x \in I \,:\, M|x - \alpha| < 1\} \subseteq I. \tag{4.23}$$

Veamos ahora que si $x_0 \in \hat{I}$, entonces $x_n \in \hat{I}$ para toda $n \geq 0$. De hecho de (4.22) tenemos que si $x_n \in \hat{I}$, entonces $|\alpha - x_{n+1}| \leq M|\alpha - x_n|^2$. Multiplicando por M ambos lados obtenemos que

$$M|\alpha - x_{n+1}| \leq (M|\alpha - x_n|)^2 . \tag{4.24}$$

Ahora como $M|\alpha - x_n| < 1$, tenemos que $M|\alpha - x_{n+1}| < 1$. También de (4.24) obtenemos que $|\alpha - x_{n+1}| \leq |\alpha - x_n|$, i.e., $x_{n+1} \in I$. Combinando esto con $M|\alpha - x_{n+1}| < 1$ obtenemos que $x_{n+1} \in \hat{I}$. De aquí que si $x_0 \in \hat{I}$, entonces las iteraciones del Método de Newton están bien definidas, permanecen en $\hat{I}$, y (4.24) implica que

$$|\alpha - x_n| \leq \frac{1}{M} \left(M|\alpha - x_0|\right)^{2^n}, \quad n \geq 0. \tag{4.25}$$

Si $x_0 \in \hat{I}$ de modo que $M|\alpha - x_0| < 1$, entonces (4.25) implica que $x_n \to \alpha$ y la convergencia es de orden dos ya que $|\alpha - x_{n+1}| \leq M|\alpha - x_n|^2$. Para terminar note que (4.22) se puede escribir como

$$\frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{1}{2}\frac{f''(\xi_n)}{f'(x_n)}. \tag{4.26}$$

Dejando $n \to \infty$ y usando que $x_n \to \alpha$, y por consiguiente que $\xi_n \to \alpha$, y la continuidad de f', f'', obtenemos la expresión (4.19). □

La selección del punto inicial x_0 que garantiza la convergencia del Método de Newton no es un asunto trivial. De la demostración del Teorema 4.7 se observa que si $M|\alpha - x_0| < 1$ entonces las iteraciones del método convergen a la raíz. Este estimado del punto inicial no es práctico ya que M, que esta dada por (4.21), es por lo general difícil o imposible de calcular. Cualquier otro estimado o conocimiento previo de la localización de la raíz debe entonces utilizarse para asegurar la convergencia del método. Por ejemplo una gráfica de la función f podría arrojar una idea sobre la localización de la raíz. Luego de esto se podría utilizar un método como el de la bisección para corregir la aproximación inicial y finalmente pasar ésta al método de Newton. Este proceso se asemeja a los *métodos predictor–corrector* los cuales son comunes en la solución numérica de ecuaciones diferenciales ordinarias (c.f., Capitulo (7)).

Una vez tenemos un método iterativo, como el de Newton digamos, que no tiene en general una expresión como (4.8) para predecir el número de iteraciones necesarias para lograr un error pre–determinado, ¿cómo detenemos las iteraciones? Un criterio heurístico para detener el método es el siguiente: suponiendo que las iteraciones (x_n) están *cerca* de la raíz α, podemos escribir que:

$$\text{Rel}(x_n) = \frac{\alpha - x_n}{\alpha} \approx \frac{x_{n+1} - x_n}{x_n}. \tag{4.27}$$

Así que si $|x_{n+1} - x_n| \leq |x_n| 10^{-t}$ tenemos aproximadamente t cifras significativas en x_n como aproximación de α.

4.3 Método de la Secante

La motivación del método de la secante viene de que en ocasiones es complicado o quizás imposible calcular la derivada de la función f que aparece en el método de Newton. Por ejemplo la función f podría estar especificada por un número discreto de puntos o dada por un programa de computadora. En tales situaciones el método de Newton no es práctico y buscamos un método intermedio entre el de la bisección y el de Newton. Para esto suponemos que tenemos dos aproximaciones x_0, x_1 de la raíz α. Podemos ahora construir la *secante* a la función f en los puntos $(x_0, f(x_0)), (x_1, f(x_1))$, la cual está dada por:

$$y = f(x_1) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1). \tag{4.28}$$

Definimos la nueva aproximación x_2 como el intercepto en x de esta secante, i.e.,

$$x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1). \tag{4.29}$$

(Ver Figura 4.2). Este proceso lo podemos repetir ahora con x_1, x_2 para

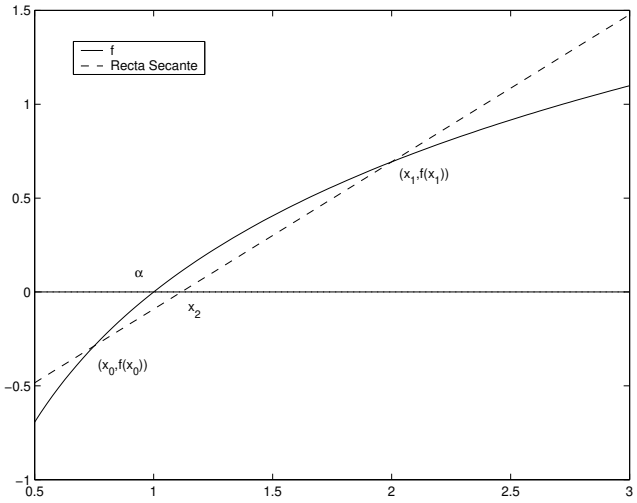


Figura 4.2: Método de la Secante: se aproxima la función f con la secante en los puntos dados.

generar x_3, etc.. Obtenemos así la recursión que define el *Método de la*

Secante:

$$\begin{cases} x_{n+1} = x_n - \dfrac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n), \quad n \geq 1, \\ x_0, x_1 \text{ dados.} \end{cases} \tag{4.30}$$

Ejemplo 4.8. Considere nuevamente la función $f(x) = x^2 + x - 1$. Con $x_0 = 0$ y $x_1 = 1$ tenemos los siguientes resultados de las iteraciones (4.30):

n	x_n	$f(x_n)$
0	0	-1
1	1	1
2	0.5	-0.25
3	0.6	-0.04
4	0.61905	0.0022676
5	0.61803	-1.842e-005

Podemos apreciar de estos simples resultados, una convergencia entre medio de la del método de la bisección y el de Newton. □

El método de la secante es un ejemplo de un *método iterativo de dos puntos* ya que predice en el paso $n+1$ basado en la información obtenida en los pasos n, $n-1$. Note también que

$$\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \approx \frac{1}{f'(x_n)}, \tag{4.31}$$

de modo que el método de la secante se puede ver como una discretización del método de Newton.

El análisis de convergencia del método de la secante es un tanto más complicado que el del método de Newton y requiere del concepto de *diferencias divididas* las cuales se discuten en el Capítulo (5). Usando propiedades de las diferencias dividas (Problema (4.11)) se puede demostrar que las iteraciones del método de la secante satisfacen:

$$\alpha - x_{n+1} = (\alpha - x_n)(\alpha - x_{n-1}) \left[-\frac{f''(\xi_n)}{2f'(\eta_n)} \right], \tag{4.32}$$

donde η_n está entre x_{n-1} y x_n, y ξ_n pertenece al intervalo mínimo que contiene a x_{n-1}, x_n, α. Usando esta fórmula se puede demostrar el siguiente teorema.

Teorema 4.9 (Convergencia Local del Método de la Secante)**.** *Suponga que $f : \mathbb{R} \to \mathbb{R}$ es una función C^2 en una vecindad del número α para la cual $f(\alpha) = 0, f'(\alpha) \neq 0$. Entonces si x_0, x_1 se seleccionan suficientemente cerca de α, las iteraciones (4.30) convergen a α. Además*

$$\lim_{n\to\infty} \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^r} = \left| \frac{f''(\alpha)}{2f'(\alpha)} \right|^{r-1}, \tag{4.33}$$

donde $r = (1 + \sqrt{5})/2$.

Demostración: Como $f'(\alpha) \neq 0$ y f' es continua, existe un intervalo cerrado I alrededor de $x = \alpha$ tal que $f'(x) \neq 0$ para $x \in I$. Defina el número M por

$$M = \frac{1}{2} \frac{\max_{x\in I} |f''(x)|}{\min_{x\in I} |f'(x)|}, \tag{4.34}$$

el cual existe y es finito. Si $x_{n-1}, x_n \in I$, entonces el lado derecho de (4.32) esta bien definido. Defina ahora el intervalo $\hat{I}$ por

$$\hat{I} = \{x \in I \,:\, M|x - \alpha| < 1\} \subseteq I. \tag{4.35}$$

Veamos ahora que si $x_0, x_1 \in \hat{I}$, entonces $x_n \in \hat{I}$ para toda $n \geq 0$. De hecho de (4.32) tenemos que si $x_{n-1}, x_n \in \hat{I}$, entonces

$$|\alpha - x_{n+1}| \leq M\,|\alpha - x_n|\,|\alpha - x_{n-1}|\,.$$

Multiplicando por M ambos lados obtenemos que

$$M\,|\alpha - x_{n+1}| \leq M\,|\alpha - x_n|\,M\,|\alpha - x_{n-1}|\,. \tag{4.36}$$

Ahora como $M|\alpha - x_i| < 1$, $i = n - 1, n$, tenemos que $M|\alpha - x_{n+1}| < 1$. También de (4.36) obtenemos que $|\alpha - x_{n+1}| \leq |\alpha - x_n|$, i.e, $x_{n+1} \in I$. Combinando esto con $M|\alpha - x_{n+1}| < 1$ obtenemos que $x_{n+1} \in \hat{I}$. De aquí que si $x_0, x_1 \in \hat{I}$, entonces las iteraciones del Método de la Secante están bien definidas, permanecen en $\hat{I}$, y (4.36) implica que

$$f_{n+1} \leq f_n f_{n-1}, \tag{4.37}$$

donde definimos $f_n = M\,|\alpha - x_n|$. Sea $\gamma = \max\{f_0, f_1\}$. Es fácil verificar ahora usando inducción matemática que (4.37) implica que:

$$f_n \leq \gamma^{F_n}, \quad n \geq 0, \tag{4.38}$$

donde

$$F_{n+1} = F_n + F_{n-1}, \quad F_0 = F_1 = 1. \tag{4.39}$$

Tenemos pues de (4.38) que si $\gamma < 1$, entonces $f_n \to 0$, i.e., (x_n) converge a la raíz α. La iteración (4.39) define a los *números de Fibonacci* y es bien conocido que para n grande:

$$F_n \approx \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{n+1}, \quad n \text{ grande.} \tag{4.40}$$

Esto en conjunto con (4.38) esencialmente demuestra que el orden de convergencia es al menos $r = (1+\sqrt{5})/2$.

Dado que las iteraciones convergen, entonces de (4.32), para n suficientemente grande, podemos escribir

$$|\alpha - x_{n+1}| \approx \hat{M} \, |\alpha - x_n| \, |\alpha - x_{n-1}| \,, \quad \hat{M} = \left| \frac{f''(\alpha)}{2f'(\alpha)} \right|.$$

Multiplicando esta aproximación por $\hat{M}$ en ambos lados, definiendo

$$\hat{F}_n = \log(\hat{M} \, |\alpha - x_n|),$$

y convirtiendo la aproximación en una igualdad, obtenemos la recursión:

$$\hat{F}_{n+1} = \hat{F}_n + \hat{F}_{n-1}, \quad \hat{F}_0 = \hat{F}_1 = 1, \tag{4.41}$$

donde por conveniencia hemos tomado $\hat{F}_0 = \hat{F}_1 = 1$. Asi que $(\hat{F}_n)$ se comporta como los números de Fibonachi y usando (4.40) podemos escribir que

$$|\alpha - x_n| \approx \frac{1}{\hat{M}} \exp\left(\frac{1}{\sqrt{5}} r^{n+1} \right), \quad n \text{ grande,} \tag{4.42}$$

donde usamos la notación $\exp(x) = \mathrm{e}^x$. Tenemos ahora que

$$\frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^r} \approx \frac{\frac{1}{\hat{M}} \exp\left(\frac{1}{\sqrt{5}} r^{n+2} \right)}{\frac{1}{\hat{M}^r} \exp\left(\frac{1}{\sqrt{5}} r^{n+2} \right)} = \hat{M}^{r-1}, \tag{4.43}$$

lo cual resulta, en forma asintótica, la fórmula (4.33). □

En la siguiente tabla resumimos las propiedades de los métodos de la bisección, de Newton y de la Secante:

Método	**Orden de convergencia**	**Evaluaciones por iteración (función y/o derivada)**
Bisección	Lineal con razón 1/2	1 para f
Newton	Dos	1 para f; 1 para f'
Secante	$(1+\sqrt{5})/2 \approx 1.62$	1 para f

Si f' es difícil o imposible de evaluar, entonces el método de la secante podría ser más conveniente que el de Newton. En ocasiones, f puede estar especificada por una tabla o programa de computadora, lo que hace el cálculo exacto de f' imposible. Si f' está disponible y es computacionalmente viable, entonces el método de Newton es preferible por su convergencia rápida. Note que para ambos métodos (secante y Newton), el principio básico para derivar las ecuaciones que definen estos procesos, es que en cada paso aproximamos la función original con una función lineal para la cual podemos calcular sus raíces fácilmente.

4.4 Otros Métodos

Podemos ahora proceder en forma similar a la del método de la secante y suponer que en lugar de dos, tenemos tres puntos: $(x_0, f(x_0))$, $(x_1, f(x_1))$, $(x_2, f(x_2))$. Buscamos ahora el polinomio cuadrático que interpola estos puntos y definimos x_3 como la raíz de este polinomio que este más cerca de x_2, etc.. El algoritmo que resulta se conoce como el *método de Müller* y tiene un orden de convergencia de 1.84 aproximadamente. (Ver [18], [3], [30]). También tiene la ventaja que puede producir iteraciones complejas a partir de valores reales. (Este no es el caso para los otros métodos descritos hasta ahora).

En ocasiones se desea buscar raíces de funciones que varían lentamente cerca de la raíz pero que crecen rápidamente al alejarnos de ésta. Una aproximación lineal o cuadrática de dicha función no es apropiada. En este caso es preferible interpolar la función con una de la forma $q(x) = (x-a)/(bx-c)$, lo cual requiere tres puntos para la interpolación, al igual que en el método de Müller. El algoritmo que resulta se conoce como el *método lineal fraccionado* y tiene un orden de convergencia igual al del método de Müller.

4.5 Iteraciones de Punto Fijo

Queremos estudiar el problema de resolver ecuaciones de la forma

$$x = g(x), \tag{4.44}$$

donde $g : D \subset \mathbb{R} \to \mathbb{R}$ es una función dada. Cualquier solución de esta ecuación se llama un *punto fijo* de g. Note que las soluciones de (4.44) corresponden a las intersecciones de las curvas $y = g(x)$ y $y = x$. (Vea la Figura 4.3.) El problema (4.1) es equivalente al (4.44) si tomamos, digamos,

$$g(x) = x - \alpha f(x),$$

para cualquier $\alpha \neq 0$.

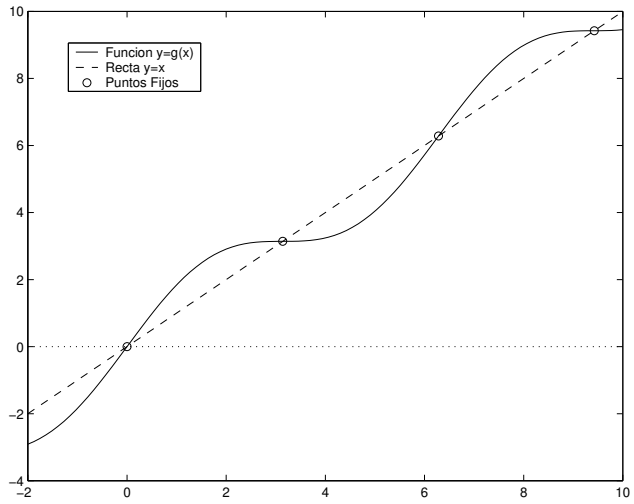


Figura 4.3: Puntos fijos como intersecciones de las curvas $y = g(x)$, $y = x$.

Si $g : D \to D$, decimos que la sucesión (x_n) es una *iteración de punto fijo* para (4.44) si

$$x_{n+1} = g(x_n), \quad n \geq 0, \quad x_0 \in D.$$

El método de Newton se puede ver como una iteración de punto fijo para (4.44) si definimos

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{4.45}$$

Tenemos ahora el primer resultado sobre la existencia de soluciones de (4.44).

Lema 4.10. *Sea $g : [a, b] \to [a, b]$ una función continua. Entonces (4.44) tiene al menos una solución en $[a, b]$.*

Demostración: Note que $g(a) \geq a$ y $g(b) \leq b$. Por lo tanto $g(x) - x$ cambia de signo en $[a, b]$. Por el Teorema A.3, existe un $\alpha \in [a, b]$ tal que $g(\alpha) - \alpha = 0$. □

Aunque el Lema (4.10) garantiza la existencia de al menos una solución de (4.44), éste no ofrece ninguna información sobre la unicidad de la solución o de como calcularla. El siguiente resultado aclara ambas de éstas situaciones.

Teorema 4.11 (Teorema de la Contracción)**.** *Sea $g : [a, b] \to [a, b]$ una función continua y suponga que g es una contracción, i.e., existe una constante $0 \leq \lambda < 1$ tal que*

$$|g(x) - g(y)| \leq \lambda |x - y| \quad \forall\ x, y \in [a, b]. \tag{4.46}$$

Entonces (4.44) tiene una solución α en $[a, b]$ que es única y las iteraciones (x_n) definidas por

$$x_{n+1} = g(x_n), \quad n \geq 0, \quad x_0 \in [a, b], \tag{4.47}$$

convergen a α linealmente con taza de convergencia λ. Además tenemos el estimado

$$|\alpha - x_n| \leq \frac{\lambda^n}{1 - \lambda} |x_1 - x_0|, \quad n \geq 1. \tag{4.48}$$

Demostración: Por el Lema (4.10), existe por lo menos un punto fijo $\alpha \in [a, b]$. Si α, β son dos puntos fijos, entonces

$$|\alpha - \beta| = |g(\alpha) - g(\beta)| \leq \lambda |\alpha - \beta|.$$

Como $0 \leq \lambda < 1$, tenemos que $|\alpha - \beta| = 0$, i.e., $\alpha = \beta$. Note que como $g : [a, b] \to [a, b]$, tenemos que $x_n \in [a, b]$ para toda n. Además

$$|\alpha - x_{n+1}| = |g(\alpha) - g(x_n)| \leq \lambda |\alpha - x_n|.$$

Usando esta expresión e inducción matemática obtenemos que

$$|\alpha - x_n| \leq \lambda^n |\alpha - x_0|,$$

para toda n. Como $0 \leq \lambda < 1$, obtenemos que $x_n \to \alpha$ linealmente con taza de convergencia λ. Finalmente observemos que

$$|\alpha - x_0| \leq |\alpha - x_1| + |x_1 - x_0| \leq \lambda |\alpha - x_0| + |x_1 - x_0|,$$

de donde obtenemos que

$$|\alpha - x_0| \leq \frac{1}{1-\lambda}|x_1 - x_0|.$$

Esto en conjunto con $|\alpha - x_n| \leq \lambda^n |\alpha - x_0|$ completa la demostración. □

La desigualdad (4.46) se conoce como la *condición de contractividad* y λ se llama la *constante de contracción*. Usualmente puede ser complicado verificar la condición de contractividad. Estudiamos ahora formas indirectas de obtener esta condición.

Corolario 4.12. *Sea $g : [a,b] \to [a,b]$ una función continuamente diferenciable tal que*

$$\max_{a \leq x \leq b} |g'(x)| < 1. \tag{4.49}$$

Entonces las condiciones del Teorema 4.11 se cumplen y además

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha). \tag{4.50}$$

Demostración: Defina

$$\lambda = \max_{a \leq x \leq b} |g'(x)|.$$

Entonces $0 \leq \lambda < 1$ y por el Teorema del Valor Medio (Teorema A.1) tenemos que para cualesquiera $x, y \in [a,b]$, existe un ξ entre x, y tal que

$$|g(x) - g(y)| = |g'(\xi)(x-y)| \leq \lambda |x-y|.$$

Así que las hipótesis del Teorema de la Contracción (Teorema 4.11) se cumplen y $x_n \to \alpha$ donde $x_{n+1} = g(x_n)$, $n \geq 0$, y $x_0 \in [a,b]$ es arbitrario. Note también que

$$\begin{aligned}
\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} &= \lim_{n\to\infty} \frac{g(\alpha) - g(x_n)}{\alpha - x_n}, \\
&= \lim_{n\to\infty} \frac{g'(\xi_n)(\alpha - x_n)}{\alpha - x_n}, \\
&= \lim_{n\to\infty} g'(\xi_n) = g'(\alpha),
\end{aligned}$$

donde usamos la continuidad de g' y que como ξ_n esta entre α y x_n, y $x_n \to \alpha$, entonces $\xi_n \to \alpha$. □

Ejemplo 4.13. Considere la ecuación de punto fijo

$$x = x + \operatorname{sen}(x), \quad x \in [2, 4].$$

Note que $\alpha = \pi$ es el único punto fijo de la ecuación en el intervalo dado. Vamos ahora a verificar las condiciones del Teorema de la contracción. Claramente $g(x) = x + \operatorname{sen}(x)$ es continua en el intervalo dado. Además, como $g'(x) = 1 + \cos(x) \geq 0$, tenemos que g es creciente. Como $\operatorname{sen}(2) > 0$ y $\operatorname{sen}(4) < 0$, tenemos que

$$g(2) = 2 + \operatorname{sen}(2) > 2, \quad g(4) = 4 + \operatorname{sen}(4) < 4.$$

Esto combinado con que g es creciente implica que $g : [2, 4] \longrightarrow [2, 4]$. Falta ver que es una contracción. Para esto utilizamos el Corolario 4.12. Para $x \in [2, 4]$ tenemos que $-1 \leq \cos(x) \leq \max\{\cos(2), \cos(4)\} < 0$, lo que implica que $0 \leq 1 + \cos(x) < 1$ para $x \in [2, 4]$. De modo que

$$\max_{x \in [2,4]} |g'(x)| < 1,$$

es decir que g es una contracción. Tenemos entonces que las iteraciones de punto fijo

$$x_{n+1} = x_n + \operatorname{sen}(x_n), \quad n \geq 0,$$

convergen (linealmente) al punto fijo $\alpha = \pi$ para cualquier $x_0 \in [2, 4]$. En la tabla siguiente mostramos las primeras iteraciones de este proceso:

n	x_n
0	3
1	3.141120008059867
2	3.141592653572196
3	3.141592653589793

Note que con solo tres iteraciones, ya la ultima iteración tiene 16 cifres correctas como aproximación a π. La razón para esta convergencia tan rápida es que como veremos más adelante, estas iteraciones en realidad convergen con orden tres al punto fijo. ¡Es decir, la cantidad de cifres significativas se triplica con cada iteración! □

La condición (4.49) garantiza la convergencia global de las iteraciones de punto fijo (4.47) para cualquier $x_0 \in [a, b]$. No obstante, ya que es una

condición que aplica en todo el intervalo $[a, b]$, ésta podría ser difícil o complicada de verificar en algunos problemas. Vamos a examinar ahora una condición más débil que (4.49) pero que es más fácil de verificar. No obstante, al debilitar o relajar (4.49) perdemos la propiedad de convergencia global de las iteraciones. También vamos a remplazar la condición $g : [a, b] \to [a, b]$ con una más sencilla, pero al hacer esto perdemos la garantía de la existencia del punto fijo.

Corolario 4.14. *Sea $g : [a, b] \to \mathbb{R}$ una función continuamente diferenciable y suponga que existe un $\alpha \in [a, b]$ tal que $\alpha = g(\alpha)$, y con $|g'(\alpha)| < 1$. Entonces los resultados del Corolario (4.12) siguen siendo ciertos pero para x_0 suficientemente cerca de α.*

Demostración: Como g' es continua y $|g'(\alpha)| < 1$, existe un $\varepsilon > 0$ tal que

$$\lambda \equiv \max_{x \in [\alpha-\varepsilon, \alpha+\varepsilon]} |g'(x)| < 1.$$

Tomando $I = [\alpha - \varepsilon, \alpha + \varepsilon]$, tenemos que para $x \in I$,

$$|g(x) - \alpha| = |g(x) - g(\alpha)| = |g'(\xi)(x - \alpha)| \leq \lambda\varepsilon < \varepsilon,$$

i.e., $g(x) \in I$. Así que $g : I \to I$ y el Corolario (4.12) aplica ahora. □

Los resultados anteriores los podemos resumir diciendo que si tenemos una contracción local o global, entonces la iteraciones de punto fijo asociadas convergen local o globalmente a un punto fijo, linealmente con una razón o taza de convergencia dada por la constante de contractividad.

La condición $|g'(\alpha)| < 1$ es necesaria para la convergencia. De hecho de la identidad

$$|\alpha - x_{n+1}| = |g(\alpha) - g(x_n)| = |g'(\xi_n)|\,|\alpha - x_n|\,,$$

vemos que si $|g'(\alpha)| > 1$ y ξ_n esta "cerca" de α, entonces en general $|g'(\xi_n)| > 1$ y tendríamos que $|\alpha - x_{n+1}| > |\alpha - x_n|$. O sea, a medida que nos acercamos al punto fijo α, este *repele* las iteraciones.

Ejemplo 4.15. Considere la ecuación $x = x + c(x^2 - a)$ donde $a > 0$. Note que si $c \neq 0$ esta ecuación tiene puntos fijos $\pm\sqrt{a}$. Queremos escoger c de modo que las iteraciones de punto fijo

$$x_{n+1} = x_n + c(x_n^2 - a), \quad n \geq 0,$$

converjan localmente a un punto fijo. Para ésto necesitamos que $|g'(\pm\sqrt{a})| < 1$ donde $g(x) = x + c(x^2 - a)$. Tomamos el caso del punto fijo $\sqrt{a}$. Entonces como $g'(x) = 1 + 2cx$, queremos hallar c tal que $-1 < 1 + 2c\sqrt{a} < 1$, i.e., $-1 < c\sqrt{a} < 0$. Note en particular que c tiene que ser negativo. Ahora si $a \geq 1$, entonces $\sqrt{a} \leq a$, y por consiguiente $c\sqrt{a} \geq ca$. Así que si $ca > -1$, i.e., $c > -1/a$, entonces $-1 < c\sqrt{a} < 0$ se cumple y la iteración de punto fijo converge localmente al punto fijo. Por ejemplo, si tomamos $c = -1/(2a)$, entonces tenemos que la iteración de punto fijo:

$$x_{n+1} = x_n - \frac{1}{2a}(x_n^2 - a), \quad n \geq 0,$$

converge linalmente (localmente) al punto fijo $\sqrt{a}$. ¿Puede decir para que valores de x_0 la iteración converge? (Vea el Ejercicio 4.17.)

Si $0 < a < 1$, entonces aplicamos el método a $b = 1/a$, que es mayor de uno, obteniendo así $\sqrt{b}$ y luego calculamos $\sqrt{a}$ como el recíproco de $\sqrt{b}$. □

Vamos a ver ahora condiciones bajo las cuales la iteración de punto fijo (4.47) tiene convergencia más rápida que lineal.

Teorema 4.16. *Sea $\alpha \in [a, b]$ un punto fijo de la ecuación $x = g(x)$. Suponga que la función g tiene p derivadas continuas en $[a, b]$, $p \geq 2$, y que*

$$g^{(j)}(\alpha) = 0, \quad 1 \leq j \leq p-1, \tag{4.51a}$$

$$g^{(p)}(\alpha) \neq 0. \tag{4.51b}$$

Entonces las iteraciones de punto fijo (4.47) convergen localmente al punto fijo α con un orden de convergencia p. Además

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^p} = \frac{(-1)^{p+1}}{p!} g^{(p)}(\alpha). \tag{4.52}$$

Demostración: Usando el Teorema de Taylor (Teorema 1.2) podemos escribir

$$\begin{aligned} x_{n+1} = g(x_n) &= \sum_{k=0}^{p-1} \frac{g^{(k)}(\alpha)}{k!}(x_n - \alpha)^k + \frac{g^{(p)}(\xi_n)}{p!}(x_n - \alpha)^p, \\ &= \alpha + \frac{g^{(p)}(\xi_n)}{p!}(x_n - \alpha)^p, \end{aligned}$$

donde ξ_n esta entre α y x_n. Tenemos pues que

$$\alpha - x_{n+1} = \frac{(-1)^{p+1}}{p!} g^{(p)}(\xi_n)(\alpha - x_n)^p.$$

Como $g'(\alpha) = 0$, el Corolario (4.14) implica que si x_0 se toma suficientemente cerca de α, entonces $x_n \to \alpha$. Esto en conjunto con la identidad de arriba implica que la convergencia es de orden p y usando la continuidad de $g^{(p)}(x)$ obtenemos el estimado (4.52). □

Ejemplo 4.17. La iteración de punto fijo $x_{n+1} = 2-(1+c)x_n+cx_n^3$, $n \geq 0$, tiene $\alpha = 1$ como un punto fijo. Con $g(x) = 2-(1+c)x+cx^3$, tenemos que $g'(x) = -(1+c)+3cx^2$. De aquí que $g'(\alpha) = 2c-1$. Tenemos pues que la iteración de punto fijo converge localmente a la raíz α para cualquier $c \in (0,1)$ que es donde $|g'(\alpha)| < 1$. Esta convergencia es al menos lineal. Ahora si $c = 1/2$, entonces $g'(\alpha) = 0$ y por el Teorema 4.16 la convergencia sería cuadrática. □

Ejemplo 4.18. El método de Newton se puede ver como la iteración de punto fijo de $x = g(x)$ donde $g(x)$ está dado por (4.45). Si suponemos que $f(\alpha) = 0$, $f'(\alpha) \neq 0$, y que f tiene tres derivadas continuas, entonces $\alpha = g(\alpha)$ y

$$\begin{aligned} g'(x) &= \frac{f(x)f''(x)}{(f'(x))^2}, \\ g''(x) &= \frac{f'(x)(f'(x)f''(x)+f(x)f'''(x))-2f(x)(f''(x))^2}{(f'(x))^3}. \end{aligned}$$

De aquí tenemos que

$$g'(\alpha) = 0, \quad g''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)}.$$

Por el Teorema 4.16, si x_0 se toma suficientemente cerca de α, entonces $x_n \to \alpha$ con orden al menos dos. Note que este resultado aunque un tanto más fácil de obtener que el del Teorema 4.7, requiere tres derivadas de f mientras que el Teorema 4.7 solo requiere dos. □

Ejemplo 4.19. La iteración de punto fijo:

$$x_{n+1} = x_n + \operatorname{sen} x_n, \quad n \geq 0,$$

tiene a $\alpha = \pi$ como punto fijo. Con $g(x) = x + \operatorname{sen} x$ tenemos que:

$$g'(x) = 1 + \cos x, \quad g'(\alpha) = 0,$$

$$\begin{aligned} g''(x) &= -\operatorname{sen} x, \quad g''(\alpha) = 0, \\ g'''(x) &= -\cos x, \quad g'''(\alpha) = 1. \end{aligned}$$

Tenemos ahora que por el Teorema 4.16, si x_0 se toma suficientemente cerca de α, entonces $x_n \to \alpha$ con orden al tres. □

4.6 Raíces Multiples

La condición $f'(\alpha) \neq 0$ en los Teoremas 4.7 y (4.9) garantiza que la función $f(x)$ corta el eje de x en $x = \alpha$ en forma *transversal*. Cuando esta condición de transversalidad no se cumple en la raíz $x = \alpha$, entonces el método numérico puede no converger, y si converge, lo hace más lento que en el caso transversal. Vamos a estudiar esta situación en más detalles, en particular, para el método de Newton.

Sea f una función continua, α un número real tal que $f(\alpha) = 0$, y m un entero positivo. Decimos que α es una raíz de f de *multiplicidad* m si existe una función $g(x)$ continua tal que

$$f(x) = (x - \alpha)^m g(x), \quad g(\alpha) \neq 0. \tag{4.53}$$

Si $m = 1$ decimos que la raíz α es *simple*.

Ejemplo 4.20. Para $f(x) = (x-2)^3(x-1)^2$ tenemos que $\alpha = 2$ es raíz de f de multiplicidad 3 y que $\alpha = 1$ es raíz de multiplicidad 2.

Para la función $f(x) = \operatorname{sen}(x) - x$ tenemos que $\alpha = 0$ es una raíz de f de multiplicidad tres. Para ver esto, note que podemos escribir $f(x) = x^3 g(x)$ con

$$g(x) = \begin{cases} \dfrac{\operatorname{sen}(x) - x}{x^3}, & x \neq 0, \\ \\ -\dfrac{1}{6}, & x = 0. \end{cases}$$

Es fácil ver que g es continua, y como $g(0) \neq 0$, tenemos que $\alpha = 0$ es raíz de multiplicidad tres de f. □

Con el proximo resultado, podemos calcular la multiplicidad de una raíz sin producir la función g de la definición. Esto se logra utilizando las derivadas de f en la raíz.

Teorema 4.21. *Sea $f \in C^m(a,b)$ y $\alpha \in (a,b)$. Entonces α es raíz de f de multiplicidad m si y solo si*

$$\begin{cases} f^{(j)}(\alpha) = 0, & 0 \leq j \leq m-1, \\ f^{(m)}(\alpha) \neq 0. \end{cases} \tag{4.54}$$

Demostración: Supongamos que (4.54) es cierta. Por el Teorema de Taylor tenemos que podemos escribir:

$$f(x) = \sum_{j=0}^{m-1} \frac{f^{(j)}(\alpha)}{j!}(x-\alpha)^j + \frac{f^{(m)}(\xi_x)}{m!}(x-\alpha)^m = \frac{f^{(m)}(\xi_x)}{m!}(x-\alpha)^m.$$

Como $f^{(m)}(\alpha) \neq 0$, ξ_x esta entre α y x, y $f^{(m)} \in C(a,b)$, tenemos que para x suficientemente cerca de α, la función $f^{(m)}(\xi_x) \neq 0$. Así que con $g(x) = f^{(m)}(\xi_x)/m!$, la ecuación (4.53) se cumple, i.e., α es raíz de f de multiplicidad m.

Suponga que por el contrario, (4.53) es dado donde $g \in C^m(a,b)$. Entonces usando la regla de Leibnitz obtenemos que

$$\begin{aligned} f^{(k)}(x) &= \sum_{j=0}^{k} \frac{d^j}{dx^j}(x-\alpha)^m \frac{d^{k-j}}{dx^{k-j}} g(x), \\ &= \sum_{j=0}^{k} m(m-1)\cdots(m-j+1)(x-\alpha)^{m-j} g^{(k-j)}(x), \end{aligned} \tag{4.55}$$

$0 \leq k \leq m$. Si $0 \leq k < m$ y $0 \leq j \leq k$, entonces $0 < m-j \leq m$ y (4.55) implica que $f^{(k)}(\alpha) = 0$. Si $k = m$, entonces (4.55) se puede escribir como

$$f^{(m)}(x) = m!g(x) + \sum_{j=0}^{m-1} m(m-1)\cdots(m-j+1)(x-\alpha)^{m-j} g^{(k-j)}(x).$$

Note que al evaluar en $x = \alpha$ los términos en la sumatoria son todos cero ya que las potencias de $x-\alpha$ son todas positivas. De modo que tenemos que $f^{(m)}(\alpha) = m!g(\alpha) \neq 0$. Combinando esto con lo anterior, obtenemos (4.54). □

Ejemplo 4.22. Podemos ahora verificar de forma más directa que $\alpha = 0$ es raíz de multiplicidad tres de $f(x) = \text{sen}(x) - x$. Note que como

$$f'(x) = \cos(x) - 1, \quad f''(x) = -\text{sen}(x), \quad f'''(x) = -\cos(x),$$

entonces $f'(0) = 0$, $f''(0) = 0$, y $f'''(0) = -1$. Por el teorema anterior, la raíz es de multiplicidad tres. □

Cuando el método de Newton, o el de la secante, se utiliza en un problema con raíces multiples, la convergencia, si alguna, se hace más lenta. Por ejemplo si consideramos el problema de buscar las raíces de la ecuación $x^3 = 0$ usando el método de Newton, entonces las iteraciones de Newton están dadas por la recursión:

$$x_{n+1} = x_n - \frac{x_n^3}{3x_n^2} = \frac{2}{3}x_n, \quad n \geq 0.$$

Como $\alpha = 0$ es la única solución en este caso, vemos que el método tiene convergencia lineal en este problema. Un fenómeno similar ocurre para el método de la secante (Ejercicio 4.12).

En general se puede demostrar que si $x = \alpha$ es una raíz de multiplicidad m de la ecuación $f(x) = 0$, entonces las iteraciones del método de Newton convergen localmente a la raíz $x = \alpha$ y satisfacen

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = \frac{m-1}{m} \equiv \lambda, \tag{4.56}$$

i.e., la convergencia es lineal, si $m > 1$, con taza o razón de convergencia λ. Note que si $m > 2$, entonces $\lambda > 1/2$ y el método de la bisección es más rápido que el de Newton en tal caso.

Otro problema con el cálculo de raíces multiples es provocado por la aritmética finita de la computadora y el hecho de que la función f no cruza el eje de x transversalmente en $x = \alpha$. Esto hace que el intervalo de incertidumbre para el cálculo de la raíz sea mucho mayor de lo que sería para una raíz simple. (Vea el Ejemplo 2.9.)

¿Cómo podemos entonces calcular raíces multiples en forma efectiva? Si la multiplicidad m de la raíz es conocida, entonces podemos mejorar la convergencia del método de Newton de dos formas:

1. Calculamos la sucesión (x_n) mediante la recurrencia

$$x_{n+1} = x_n - m\frac{f(x_n)}{f'(x_n)}, \quad n \geq 0. \tag{4.57}$$

   Se puede demostrar (Ejercicio 4.18) que $x_n \to \alpha$ con orden $p = 2$. Pero el problema de la incertidumbre mencionado arriba, no mejora debido a que la raíz multiple de f está todavía presente.

2. Calculamos analíticamente $f^{(m-1)}(x)$ y le aplicamos el método de Newton a la función $F(x) = f^{(m-1)}(x)$. Note que como α es raíz simple de F, tenemos convergencia cuadrática y no hay el problema de un intervalo de incertidumbre grande. Este método es efectivo si las derivadas de f son fáciles de calcular.

Los dos métodos mencionados arriba para calcular raíces multiples requieren que la multiplicad de la raíz, o sea m, sea conocida. Para estimar m en el caso de que no se conozca, usamos el método de Newton aplicado a la función original f y vamos calculando los cocientes

$$\frac{x_{n+1} - x_n}{x_n - x_{n-1}}, \tag{4.58}$$

los cuales de acuerdo a la fórmula (4.56) aproximan a $\lambda = (m-1)/m$. Usando esto podemos aproximar m y luego procedemos con uno de los métodos (1) o (2) descritos arriba.

Ejemplo 4.23. Para una cierta función f, las iteraciones del método de Newton producen los siguientes resultados:

n	x_n	$\hat{e}_n = x_n - x_{n-1}$	$\hat{e}_n/\hat{e}_{n-1}$
0	0.75	—	—
1	0.752710	0.00271	—
2	0.754795	0.00208	0.7675
3	0.756368	0.00157	0.7548
4	0.757552	0.00118	0.7516
5	0.758441	0.000889	0.7534

Las iteraciones del método están convergiendo pero lentamente. Podemos apreciar esto al examinar la cuarta columna que indica una convergencia lineal. (De estar convergiendo mejor que linealmente, la cuarta columna debería tender a cero.) Los cocientes de la cuarta columna están convergiendo aproximadamente a 0.75. De la fórmula (4.56) tenemos pues que la raíz es de multiplicidad aproximadamente cuatro. Podemos entonces usar la iteración (4.57) con $m = 4$ para calcular la raíz o mejor, si $f^{(3)}$ y $f^{(4)}$ se pueden calcular, usamos la iteración (4.10) pero aplicada a $f^{(3)}$. □

4.7 Sistemas Nolineales

Consideramos ahora el problema de resolver un sistema de ecuaciones nolineales de n ecuaciones en n desconocidas. Sean $f_i : \mathbb{R}^n \to \mathbb{R}$, $1 \leq i \leq n$ funciones (nolineales) suficientemente diferenciables. Un sistema nolineal $n \times n$ se puede escribir de la forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0, \\ f_2(x_1, x_2, \ldots, x_n) = 0, \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0. \end{cases} \tag{4.59}$$

Si definimos $\vec{F} : \mathbb{R}^n \to \mathbb{R}^n$ por $\vec{F} = (f_1, f_2, \ldots, f_n)^t$, entonces podemos escribir (4.59) en forma vectorial como:

$$\vec{F}(\vec{x}) = \vec{0}, \quad \vec{x} = (x_1, x_2, \ldots, x_n)^t. \tag{4.60}$$

Definimos la matriz $(n \times n)$ *jacobiana* del sistema (4.60) por:

$$\vec{F}'(\vec{x}) = \left(\frac{\partial f_i(\vec{x})}{\partial x_j} \right)_{i,j=1,\ldots,n}. \tag{4.61}$$

Sea $\vec{\alpha} \in \mathbb{R}^n$ tal que $\vec{F}(\vec{\alpha}) = \vec{0}$, i.e., una solución de (4.60). Suponga que $\vec{x}_0$ es una aproximación de $\vec{\alpha}$. Entonces usando el Teorema de Taylor para funciones de varias variables, podemos escribir que

$$\vec{F}(\vec{x}) = \vec{F}(\vec{x}_0) + \vec{F}'(\vec{x}_0)(\vec{x} - \vec{x}_0) + o(\|\vec{x} - \vec{x}_0\|).$$

Definimos ahora la aproximación siguiente como la solución $\vec{x}_1$ de

$$\vec{F}(\vec{x}_0) + \vec{F}'(\vec{x}_0)(\vec{x}_1 - \vec{x}_0) = \vec{0},$$

i.e.,

$$\vec{x}_1 = \vec{x}_0 - \vec{F}'(\vec{x}_0)^{-1}\vec{F}(\vec{x}_0).$$

De esta forma continuamos obteniendo así la versión para sistemas del *Método de Newton* dada por:

$$\begin{cases} \vec{x}_{k+1} = \vec{x}_k - \vec{F}'(\vec{x}_k)^{-1}\vec{F}(\vec{x}_k), \quad k \geq 0, \\ \vec{x}_0 \ \text{ dado.} \end{cases} \tag{4.62}$$

Si $\vec{F}'(\vec{\alpha})$ es nosingular, y $\vec{x}_0$ se toma suficientemente cerca de $\vec{\alpha}$, entonces se puede demostrar que las iteraciones $(\vec{x}_k)$ convergen a la raíz $\vec{\alpha}$. (Ver [23]). Las iteraciones (4.62) se pueden reescribir para que no haya que calcular el inverso de una matriz en cada iteración. Esto se hace de la forma:

$$\begin{cases} \vec{F}'(\vec{x}_k)\vec{z}_k = -\vec{F}(\vec{x}_k), \\ \vec{x}_{k+1} = \vec{x}_k + \vec{z}_k, \quad k \geq 0, \\ \vec{x}_0 \ \text{ dado.} \end{cases} \tag{4.63}$$

Ejemplo 4.24. Considere el problema de aproximar una solución del sistema:

$$\begin{cases} x^3 - xy^2 + y^3 = 0, \\ x\,\text{sen}(xy) + 1 = 0. \end{cases}$$

Tenemos con $\vec{x} = (x, y)^t$ que

$$\begin{aligned} \vec{F}(\vec{x}) &= \begin{bmatrix} x^3 - xy^2 + y^3 \\ x\,\text{sen}(xy) + 1 \end{bmatrix}, \\ \vec{F}'(\vec{x}) &= \begin{bmatrix} 3x^2 - y^2 & -2xy + 3y^2 \\ \text{sen}(xy) + xy\cos(xy) & x^2\cos(xy) \end{bmatrix}. \end{aligned}$$

Estas dos expresiones las calculamos en MATLAB mediante las siguientes funciones:

```
function z=f(w)
z=zeros(2,1);
x=w(1);y=w(2);
z(1)=x^3-x*y^2+y^3;
z(2)=x*sin(x*y)+1;

function z=fp(w)
z=zeros(2,2);
x=w(1);y=w(2);
z(1,1)=3*x^2-y^2;
z(1,2)=-2*x*y+3*y^2;
z(2,1)=sin(x*y)+x*y*cos(x*y);
z(2,2)=x^2*cos(x*y);
```

El siguiente programa en MATLAB es una implementación de la recursión (4.63) usando como punto inicial $\vec{x}_0 = (1, 0)^t$:

```
x0=[1,0]';
normx=1;
normz=1;
while normz>1.0e-6*normx
    f0=f(x0);
    fp0=fp(x0);
    z=-fp0\f0;
    normz=norm(z,2);
    normx=norm(x0,2);
    x0=x0+z;
end
x0
```

Con este programa obtenemos la siguiente solución aproximada del sistema: $(1.1674, -0.8812)^t$. □

4.8 Ejercicios

Ejercicio 4.1. Calcule la raíz más grande de la ecuación

$$16x^3 - 132x^2 - 12x + 99 = 0.$$

Despúes de haber calculado la raíz mayor, use división sintética y calcule las raíces restantes.

Ejercicio 4.2. Demuestre que en el método de la bisección las sucesiones (a_n) y (b_n) cumplen que $a_n b_n + a_{n-1} b_{n-1} = a_{n-1} b_n + a_n b_{n-1}$ para toda $n \geq 1$.

Ejercicio 4.3. Si el método de la bisección se utiliza para calcular una raíz comenzando las iteraciones en el intervalo $[a, a+1]$ donde $a \geq \beta^m$, $m \geq 0$, ¿cúantas iteraciones se necesitan para aproximar la raíz con toda la presición en el sistema de punto flotante (β, t, N, M)? Conteste la misma pregunta pero comenzando ahora con el intervalo $[2^m, 2^{m+1}]$.

Ejercicio 4.4. Sea A la matriz 3×3 y b el vector 3×1 dados por:

$$A = \begin{pmatrix} 6 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 6 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Escriba un programa en MATLAB basado en el método de la bisección, que aproxime a 12 cifras significativas el valor de $\alpha > 0$ tal que la solución x de $(A+\alpha I)x = b$ satisfaga $x^t x = 1$. **Ayuda**: La solución x de $(A+\alpha I)x = b$ se puede ver como una función $x(\alpha)$ la cual al sustituirse en la ecuación $x^t x = 1$ produce la ecuación (nolineal) en α:

$$g(\alpha) \equiv x(\alpha)^t x(\alpha) - 1 = 0.$$

Ejercicio 4.5. Dado que $a = 0.1$ y $b = 1.0$, ¿cúantos pasos del método de la bisección son necesarios para calcular una raíz en $[a, b]$ con un error absoluto menor que 0.5×10^{-8}.

Ejercicio 4.6. La siguiente ecuación tiene una raíz única en el intervalo $(-1, 0)$:

$$x + \exp(-Bx^2)\cos(x) = 0, \quad B > 0.$$

Use el Método de Newton para calcular la raíz lo más preciso posible para los valores de $B = 1, 5, 10, 25, 50$. Como punto inicial puede tomar entre otros $x_0 = 0$. Explique cualquier diferencia en el comportamiento del método según aumenta el valor de B. Utilice la gráfica de $f(x) = x + \exp(-Bx^2)\cos(x)$ para sustentar sus argumentos.

Ejercicio 4.7. Implemente el Método de Newton para buscar todas las raíces de un polinomio $p(x)$. Debe utilizar el Método de Horner para evaluar $p(x)$ y su derivada $p'(x)$. Si α es una raíz, como resultado del Método de Horner, se obtiene un polinomio $q(x)$ tal que $p(x) = (x-\alpha)q(x)$. Usando $q(x)$ podemos calcular las raíces restantes de $p(x)$. Utilice su programa para resolver las ecuaciones:

a) $x^3 - x^2 - x - 1 = 0$.

b) $32x^6 - 48x^4 + 18x^2 - 1 = 0$.

c) $x^3 + x - 1 = 0$.

Ejercicio 4.8. Considere la ecuación $x^2 + 3x + 1 = 0$.

a) Halle la fórmula recursiva para las iteraciones del método de Newton en este caso particular.

b) Si α es una raíz de la ecuación, i.e., $\alpha^2 + 3\alpha + 1 = 0$, verifique que

$$x_{n+1} - \alpha = \frac{(x_n - \alpha)^2}{2x_n + 3}, \quad n \geq 0.$$

Ejercicio 4.9. En este problema calculamos las raíces de la ecuación cuadrática $ax^2 + bx + c = 0$, donde $a > 0$, usando el método de Newton. (Un análisis similar se puede hacer para el caso $a < 0$.) Suponemos que las raíces de la ecuación son reales, es decir que $b^2 - 4ac \geq 0$.

a) Escriba y simplifique las ecuaciones recursivas del método de Newton aplicado a la función $f(x) = ax^2 + bx + c$.

b) Verifique que si $x_0 > -b/2a$, entonces $x_n > -b/2a$ para toda $n \geq 0$. **Ayuda:** Use inducción y el dato que $b^2 - 4ac \geq 0$.

c) Verifique que si $x_0 > -b/2a$, entonces $x_{n+1} \geq \beta$ para toda $n \geq 0$, donde β representa cualquiera de las dos raíces de $f(x) = 0$. En particular $x_{n+1} \geq \alpha$, donde α es la mayor de las raíces. **Ayuda:** Calcule $x_{n+1} - \beta$.

d) Verifique que $x_{n+1} \leq x_n$ para toda $n \geq 0$. **Ayuda:** Calcule $x_{n+1} - x_n$ y utilice la parte (b) para argumentar que esta diferencia es menor o igual que cero.

e) Combinando los resultados de las partes anteriores, verifique que (x_n) converge a α con orden dos para cualquier $x_0 > -b/2a$.

Ejercicio 4.10. La ecuación $x - 5x^{-1} = 0$ tiene como una de sus posibles soluciones $\alpha = \sqrt{5}$. Escriba y simplifique las ecuaciones recursivas que definen el método de Newton para éste problema. Partiendo de $x_0 = 2$ efectúe tres iteraciones del método.

Ejercicio 4.11. Las diferencias divididas de Newton de orden uno y dos respectivamente se definen por

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \quad f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

Más adelante veremos que

$$f[x_0, x_1] = f'(\xi), \quad f[x_0, x_1, x_2] = \frac{1}{2} f''(\eta), \tag{4.64}$$

donde ξ esta entre x_0, x_1, y η pertenece al intervalo más pequeño que contiene a x_0, x_1, x_2. Demuestre que en el método de la secante

$$\alpha - x_{n+1} = -(\alpha - x_n)(\alpha - x_{n-1}) \frac{f[x_{n-1}, x_n, \alpha]}{f[x_{n-1}, x_n]}.$$

Usando ahora (4.64) obtenga la identidad (4.32).

Ejercicio 4.12. Para la función $f(x) = x^2$ demuestre que las iteraciones del método de la secante están dadas por:

$$x_{n+1} = \frac{x_n x_{n-1}}{x_n + x_{n-1}}, \quad n \geq 1.$$

Tomando $x_0 = x_1 = 1$, demuestre que la solución a la recursión de arriba está dada por:

$$x_n = \frac{1}{F_n}, \quad F_n = F_{n-1} + F_{n-2}, \quad n \geq 2, \quad F_0 = F_1 = 1.$$

Usando que

$$\lim_{n\to\infty} \frac{F_n}{r^n} = 5^{-1/2}, \quad r = (1 + \sqrt{5})/2,$$

verifique ahora que

$$\lim_{n\to\infty} \frac{x_{n+1}}{x_n} = \frac{1}{r},$$

i.e., el método converge linealmente a la raíz $\alpha = 0$.

Ejercicio 4.13. La evaluación de una función $f(x)$ y su derivada $f'(x)$ requieren 300 y 200 operaciones aritméticas respectivamente.

a) ¿Cuántas operaciones requiere un paso del método de Newton para este problema?

b) ¿Cuántas operaciones requiere el paso inicial del método de la secante y los otros pasos?

c) Si el método de Newton requiere 4 iteraciones para calcular una raíz α de f, y el método de la secante 5 iteraciones, ¿cuál método usted utilizaría para calcular α?

Ejercicio 4.14. Considere la ecuación $x = 5 + \frac{1}{2}\text{sen}(x)$.

a) Verificando las condiciones del Teorema de la Contracción, demuestre que esta ecuación tiene una solución α que es única en el intervalo $[9/2, 11/2]$.

b) Partiendo de $x_0 = 5$, calcule las primeras tres iteraciones de la iteración de punto fijo que converge a α.

Ejercicio 4.15. La iteración de punto fijo $x_{n+1} = -1 + (2 + c^2)x_n - c^2 x_n^3$, $n \geq 0$, tiene $\alpha = 1$ como un punto fijo.

a) Determine los valores de c para los cuales la iteración converge (localmente) al punto fijo α.

b) ¿Para qué valor de c la convergencia es cuadrática?

Ejercicio 4.16. La iteración de punto fijo

$$x_{n+1} = x_n + \frac{4}{3}\operatorname{sen} x_n + \frac{1}{6}\operatorname{sen} 2x_n, \quad n \geq 0,$$

tiene $\alpha = \pi$ como punto fijo. Verifique que la iteración converge (localmente) al punto fijo con orden cinco.

Ejercicio 4.17. Verifique que para $a \geq 1$ las iteración de punto fijo:

$$x_{n+1} = x_n - \frac{1}{2a}(x_n^2 - a), \quad n \geq 0,$$

convergen (linealmente) a $\sqrt{a}$ para cualquier $x_0 \in [0.5, a]$. **Ayuda:** Verifique que las condiciones del Corolario 4.12 se cumplen en el intervalo $[0.5, a]$.

Ejercicio 4.18. Usando el Teorema 4.16 y un proceso similar al del Ejemplo 4.17, demuestre que las iteraciones (4.57) convergen localmente con orden al menos dos.

Ejercicio 4.19. Considere la siguiente variante del método de Newton para reducir a uno el número de evaluaciones de la derivada:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, \quad n \geq 0,$$

para un x_0 dado. Demuestre que (x_n) converge localmente a una raíz de $f(x) = 0$ con orden uno.

Ejercicio 4.20. Considere la siguiente iteración propuesta por Steffensen:

$$\begin{cases} g(x_n) = \dfrac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)}, \\ \\ x_{n+1} = x_n - \dfrac{f(x_n)}{g(x_n)}, \qquad\qquad n \geq 0. \end{cases}$$

¿Cúantas evaluaciones de la función f requiere este método por cada iteración? Demuestre que (x_n) converge localmente a una raíz de $f(x) = 0$ con orden dos.

Ejercicio 4.21. Considere la siguiente variante del método de Newton para reducir el número de evaluaciones de la derivada:

$$\begin{cases} x_{2n+1} = x_{2n} - \dfrac{f(x_{2n})}{f'(x_{2n})}, \\ \\ x_{2n+2} = x_{2n+1} - \dfrac{f(x_{2n+1})}{f'(x_{2n})}, \qquad n \geq 0. \end{cases}$$

Determine numéricamente el orden de convergencia de este método en varios casos particulares. Escriba esta iteración como una iteración de punto fijo y usando el Corolario (4.14) y el Teorema 4.16 confirme sus hallazgos numéricos.

Ejercicio 4.22. Verifique que la siguiente iteración:

$$x_{n+1} = x_n - \left(\frac{x_n - x_0}{f(x_n) - f(x_0)} \right) f(x_n), \quad n \geq 1,$$

donde x_0, x_1 son números dados, converge localmente a una raíz de $f(x) = 0$ con orden uno.

Ejercicio 4.23. Para una función de su selección, verifique numéricamente que el método:

$$x_{n+1} = x_n - \frac{f(x_n)}{\sqrt{f'(x_n)^2 - f(x_n) f''(x_n)}}, \quad n \geq 0,$$

converge con orden tres a un cero simple de la función f. Establezca este resultado para una función arbitraria usando el Teorema 4.16.

Ejercicio 4.24. Para una función de su selección, verifique numéricamente que el método:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2} \frac{f''(x_n)}{f'(x_n)} \left[\frac{f(x_n)}{f'(x_n)} \right]^2, \quad n \geq 0,$$

converge con orden tres a un cero simple de la función f. Verifique que el resultado también es cierto para una función arbitraria usando el Teorema 4.16.

Ejercicio 4.25. La ecuación $\text{sen}(xy) = y - x$ define a y implícitamente como función de x. La función $y(x)$ tiene un máximo para $(x, y) \approx (1, 2)$. Demuestre que las coordenadas $(\hat{x}, \hat{y})$ del máximo satisfacen las ecuaciones:

$$\begin{cases} \text{sen}(xy) - y + x = 0, \\ y\cos(xy) + 1 = 0. \end{cases}$$

Calcule una solución aproximada de este sistema usando el Método de Newton.

Ejercicio 4.26. Escriba las ecuaciones que definen el método de Newton para la solución del sistema de ecuaciones

$$\begin{cases} x^2 + y^2 = 4, \\ x^2 - y^2 = 1. \end{cases}$$

Partiendo del punto $(1, 1)$ calcule dos iteraciones del método.

Ejercicio 4.27. Use el Método de Newton para aproximar una solución cerca del punto $(0.5, 1.0, 0.0)$ con un error relativo de 10^{-4} para el sistema

$$\begin{cases} 2x^2 - x + y^2 - z = 0, \\ 32x^2 - y^2 - 20z = 0, \\ y^2 - 14xz = 0. \end{cases}$$

Capítulo 5

Interpolación de Polinomios

Suponga que en un cierto experimento de laboratorio, se mide la concentración de una sustancia en intervalos regulares de tiempo. ¿Cómo podemos estimar la concentración de la sustancia para tiempos intermedios o futuros sin tener que medirla directamente? Dadas las medidas de temperatura, humedad y velocidad del viento en distintos puntos de una cierta región, ¿cómo podemos estimar éstas cantidades en otros lugares de la región donde no tengamos medidas? En ambos casos necesitamos aproximar una función f desconocida a partir de los valores de ésta en un número finito de puntos, y las aproximaciones que nos interesan son de valores de f en otros puntos que no son los dados. Otro ejemplo pudiera ser el problema de la evaluación de una función que aunque conocida, puede que sea muy complicada o difícil de evaluar, y por consiguiente nos interesa aproximarla con otra cuya evaluación sea simple de efectuar.

Una manera de obtener éstas aproximaciones es mediante el Teorema de Taylor (Teorema 1.2). No obstante, para aplicar dicho teorema es necesario calcular las derivadas de f en un punto, lo cual no es posible en general si la función es conocida únicamente en un número finito de puntos. Otra forma de aproximar f es buscando una función $g(x)$ que pase *cerca* en algún sentido, de todos los puntos dado. Esto es lo que se conoce como aproximación en el *sentido de los cuadrados mínimos* lo cual estudiaremos más adelante en este capítulo en la Sección 5.7. Finalmente se puede aproximar a la función f con otra función $g(x)$ que pase exactamente por los puntos dados lo cual se conoce como el *problema de interpolación*. Esto es: dados los datos (x_i, y_i), $1 \leq i \leq n$, donde $y_i = f(x_i)$ para toda i, queremos hallar una función $g(x)$

tal que

$$g(x_i) = y_i, \quad 1 \leq i \leq n. \tag{5.1}$$

5.1 Interpolación Polinomial

El problema de interpolación planteado arriba, en general no tiene solución única. La función de interpolación $g(x)$ puede consistir de sumas de exponenciales, funciones trigonométricas, funciones racionales, etc.. Uno de los casos más comunes es cuando $g(x)$ es un polinomio. Esto se llama *interpolación polinomial* y el problemas básico ahora es: dados los datos (x_i, y_i), $1 \leq i \leq n$, queremos hallar un polinomio $p_{n-1}(x)$ de grado a lo más $n-1$, tal que

$$p_{n-1}(x_i) = y_i, \quad 1 \leq i \leq n. \tag{5.2}$$

Ejemplo 5.1. Considere los datos $(-2, 5), (1, 3)$. El polinomio de grado uno que interpola a estos datos es:

$$p_1(x) = 5 - (2/3)(x + 2).$$

Note que también podemos interpolar con una función de la forma $g(x) = a\exp(bx)$. De hecho con $b = -(1/3)\ln(5/3)$ y $a = 3(5/3)^{1/3}$ logramos esto. □

Vamos ahora a estudiar la existencia del polinomio de interpolación, en particular, como construirlo. Considere el caso de los datos $(-2, 10)$, $(-1, 4)$, $(1, 6)$, y $(2, 3)$. Entonces si escribimos $p_3(x) = a_1 + a_2x + a_3x^2 + a_4x^3$ tenemos que:

$$\begin{aligned} p_3(-2) = 10 &\Rightarrow a_1 - 2a_2 + 4a_3 - 8a_4 = 10, \\ p_3(-1) = 4 &\Rightarrow a_1 - a_2 + a_3 - a_4 = 4, \\ p_3(1) = 6 &\Rightarrow a_1 + a_2 + a_3 + a_4 = 6, \\ p_3(2) = 3 &\Rightarrow a_1 + 2a_2 + 4a_3 + 8a_4 = 3. \end{aligned}$$

Esto es equivalente al sistema lineal:

$$\begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 4 \\ 6 \\ 3 \end{pmatrix},$$

el cual se puede resolver con el siguiente código en MATLAB:

```
y=[10 4 6 3]';
V=[1 -2 4 -8;1 -1 1 -1;1 1 1 1;1 2 4 8];
a=V\y
```

Esto tiene como resultado

```
a =
    4.5000
    1.9167
    0.5000
   -0.9167
```

por lo que el polinomio de interpolación es:

$$p_3(x) = 4.5000 + 1.9167x + 0.5000x^2 - 0.9167x^3. \tag{5.3}$$

Este proceso lo podemos generalizar al caso de n puntos como sigue. Sea

$$p_{n-1}(x) = a_1 + a_2x + \cdots + a_nx^{n-1}. \tag{5.4}$$

Ahora (5.2) reduce a:

$$y_i = p_{n-1}(x_i) = a_1 + a_2x_i + \cdots + a_nx_i^{n-1}, \quad 1 \leq i \leq n, \tag{5.5}$$

lo cúal es equivalente al sistema lineal:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}. \tag{5.6}$$

La matriz de coeficientes de este sistema se conoce como la *matriz de Vandermonde* y se puede demostrar que es nosingular si los x_i's son todos distintos (ver Ejercicio 5.4). (Esta condición la asumiremos de aquí en adelante.) De modo que el polinomio de interpolación $p_{n-1}(x)$ existe por construcción. La unicidad de $p_{n-1}(x)$ se verifica usando el Teorema Fundamental del Álgebra. De hecho, si $q(x)$ es otro polinomio de grado $n-1$ que interpola a los datos,

entonces $p_{n-1}(x) - q(x)$ es polinomio de grado $n-1$ con n raíces, los x_i's. Pero esto es imposible a menos que $p_{n-1} = q$.

Para calcular la matriz de Vandermonde V, primero observamos que si $j > 1$, la columna j de V se obtiene multiplicando (componente a componente) el vector columna $(x_1, \ldots, x_n)^t$ con la columna $j-1$ de V. Esto es, usando la notación .* de MATLAB que denota multiplicación componente a componente, tenemos

$$\begin{bmatrix} v_{1,j} \\ \vdots \\ v_{n,j} \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} .* \begin{bmatrix} v_{1,j-1} \\ \vdots \\ v_{n,j-1} \end{bmatrix}. \tag{5.7}$$

Esto lo podemos escribir en MATLAB como `V(:,j)=x.*V(:,j-1)`, donde

```
x=[x(1),...,x(n)]';
```

Podemos ahora usar estas observaciones para escribir un programa en MATLAB que calcula los coeficientes de p_{n-1}. En el programa `x,y` son vectores de n componentes que contienen los datos $\{(x_i, y_i) : 1 \leq i \leq n\}$ y `x` tiene todos sus componentes distintos:

```
function a=interpV(x,y)
n=length(x);
V=ones(n,n);
for j=2:n
V(:,j)=x.*V(:,j-1);
end
a=V\y;
```

El polinomio p_{n-1} se puede evaluar ahora usando el método de Horner. (Ver Algoritmo (1.15)).

Ejemplo 5.2. Considere el caso de los datos $(-2, 10), (-1, 4), (1, 6)$, y $(2, 3)$ que vimos anteriormente. Podemos calcular y trazar (Figura 5.1) el polinomio de interpolación con el siguiente programa:

```
x=[-2 -1 1 2]';
y=[10 4 6 3]';
a=interpV(x,y);
x0=linspace(-3,3,100)';
```

```
y0=hornerV(a,x0);
plot(x0,y0,x,y,'x')
xlabel('x');ylabel('y')
```

□

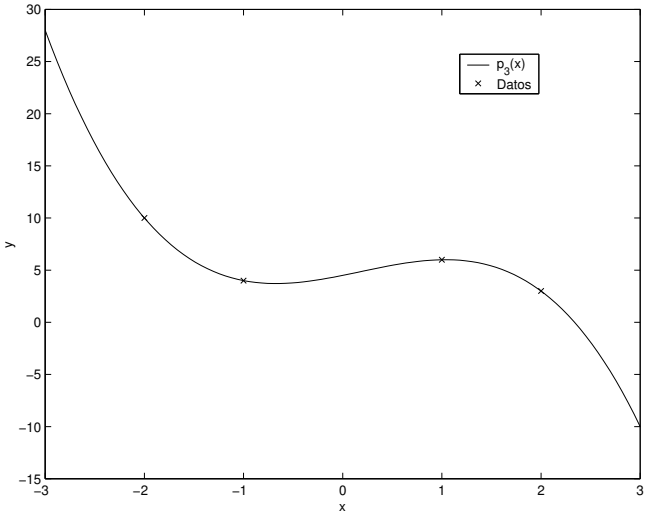


Figura 5.1: Polinomio que interpola los datos $(-2, 10)$, $(-1, 4)$, $(1, 6)$, $(2, 3)$.

5.2 Representación de Newton de p_{n-1}

Hemos visto que el polinomio de interpolación p_{n-1} es único, de modo que podemos buscar representaciones alternas del mismo sin peligro de obtener otro polinomio. La representación de Newton de p_{n-1} es tal que el sistema lineal para obtener los coeficientes en dicha representación es triangular inferior. Esto es una gran ventaja ya que la matriz (de Vandermonde) que se obtiene usando la representación (5.4) es densa y en general mal-acondicionada.

En el caso de los datos $(-2, 10), (-1, 4), (1, 6)$, y $(2, 3)$, en lugar de buscar p_3 de la forma $a_1 + a_2x + a_3x^2 + a_4x^3$, lo buscamos de la forma:

$$p_3(x) = c_1 + c_2(x + 2) + c_3(x + 2)(x + 1) + c_4(x + 2)(x + 1)(x - 1).$$

Note ahora que

$$\left\{\begin{array}{rcl} 10 &=& p_3(-2) = c_1, \\ 4 &=& p_3(-1) = c_1 + c_2, \\ 6 &=& p_3(1) = c_1 + 3c_2 + 6c_3, \\ 3 &=& p_3(2) = c_1 + 4c_2 + 12c_3 + 12c_4. \end{array}\right.$$

Esto es un sistema triangular inferior cuya solución es $c_1 = 10$, $c_2 = -6$, $c_3 = 7/3$, $c_4 = -11/12$. Tenemos entonces que:

$$p_3(x) = 10 - 6(x+2) + \frac{7}{3}(x+2)(x+1) - \frac{11}{12}(x+2)(x+1)(x-1). \quad (5.8)$$

Note que la representación canónica de $p_3(x)$ dada por la ecuación (5.3), expresa a éste polinomio en términos de la base $\{1, x, x^2, x^3\}$. Por otro lado, en (5.8) tenemos la representación de $p_3(x)$ en términos de la base $\{1, x+2, (x+2)(x+1), (x+2)(x+1)(x-1)\}$.

En el caso general, buscamos p_{n-1} de la forma:

$$\begin{array}{rcl} p_{n-1}(x) &=& \sum_{k=1}^{n} c_k \prod_{j=1}^{k-1}(x - x_j), \\ &=& c_1 + c_2(x - x_1) + c_3(x - x_1)(x - x_2) \\ && \quad + \cdots + c_n(x - x_1)\cdots(x - x_{n-1}). \end{array} \quad (5.9)$$

Examinamos el caso $n = 4$ para luego llegar al caso general. En éste caso al aplicar las condiciones de interpolación $p_3(x_i) = y_i$, $1 \leq i \leq 4$, obtenemos el sistema triangular inferior 4×4 dado por:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & (x_2 - x_1) & 0 & 0 \\ 1 & (x_3 - x_1) & \prod_{i=1}^{2}(x_3 - x_i) & 0 \\ 1 & (x_4 - x_1) & \prod_{i=1}^{2}(x_4 - x_i) & \prod_{i=1}^{3}(x_4 - x_i) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

Inmediatamente vemos que $c_1 = y_1$. Si restamos la primera fila a las ecuaciones dos, tres y cuatro, y dividiendo éstas por $(x_2 - x_1)$, $(x_3 - x_1)$, $(x_4 - x_1)$ respectivamente, obtenemos el sistema transformado:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & (x_3 - x_2) & 0 \\ 0 & 1 & (x_4 - x_2) & \prod_{i=2}^{3}(x_4 - x_i) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix},$$

donde

$$y_{i1} = \frac{y_i - y_1}{x_i - x_1}, \quad 2 \leq i \leq 4.$$

Note que hemos reducido el problema 4×4 al siguiente problema de tamaño 3×3:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & (x_3 - x_2) & 0 \\ 1 & (x_4 - x_2) & \prod_{i=2}^{3}(x_4 - x_i) \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} y_{21} \\ y_{31} \\ y_{41} \end{bmatrix}. \tag{5.10}$$

Este sistema corresponde al polinomio

$$q_2(x) = c_2 + c_3(x - x_2) + c_4(x - x_2)(x - x_3),$$

que interpola a los datos (x_2, y_{21}), (x_3, y_{31}), (x_4, y_{41}). Es fácil ver también que $p_3(x) = c_1 + (x - x_1)q_2(x)$. Note que el lado derecho del sistema (5.10) se puede calcular mediante el siguiente código en MATLAB:

$$\begin{bmatrix} y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = (y(2:4) - y(1))./(x(2:4) - x(1)).$$

En el caso general tenemos que $c_1 = y_1$ y que $p_{n-1}(x) = c_1 + (x - x_1)q_{n-2}(x)$ donde $q_{n-2}(x)$ es polinomio de grado $n - 2$ que interpola a los datos:

$$\left(x_i, \frac{y_i - y_1}{x_i - x_1}\right), \quad 2 \leq i \leq n. \tag{5.11}$$

Usando estas ideas podemos escribir el siguiente programa recursivo en MATLAB para calcular los coeficientes de la representación de Newton del polinomio de interpolación:

```
function c=interpNR(x,y)
n=length(x);
c=zeros(1,n);
c(1)=y(1);
if n>1
   c(2:n)=interpNR(x(2:n),(y(2:n)-y(1))./(x(2:n)-x(1)));
end
```

Repitiendo el proceso que nos llevó a la forma recursiva del algoritmo anterior, podemos resolver dicha recursión para obtener así la forma secuencial del algoritmo para la representación de Newton del polinomio de interpolación. El siguiente programa en MATLAB implementa dicha versión secuencial:

```
function c=interpN(x,y)
n=length(x);
c=y;
for k=1:n-1
  c(k+1:n)=(c(k+1:n)-c(k))./(x(k+1:n)-x(k));
end
```

La representación de Newton del polinomio de interpolación se puede evaluar eficientemente usando una variación del método de Horner donde escribimos el polinomio en forma anidada como sigue:

$$\begin{aligned} p_{n-1}(x) &= c_1 + (x - x_1)(c_2 + \cdots (c_{n-2} + \\ &\quad (x - x_{n-2})(c_{n-1} + c_n(x - x_{n-1}))) \cdots). \end{aligned} \tag{5.12}$$

En el caso $n = 4$ ésto reduce a :

$$p_3(x) = c_1 + (x - x_1)(c_2 + (x - x_2)(c_3 + c_4(x - x_3))).$$

Podemos ahora escribir un programa en MATLAB que implementa ésta versión del método de Horner para evaluar la representación de Newton de p_{n-1} escrita según (5.12):

```
function pval=hornerN(c,x,z)
n=length(c);
pval=c(n)*ones(size(z));
for k = n-1:-1:1
  pval=c(k)+(z-x(k)).*pval;
end
```

En general `interpNR` y `interpN` requieren menos operaciones de punto flotante que `interpV`, $O(n^2)$ en comparación con $O(n^3)$, ya que la matriz de Vandermonde es densa mientras que en la representación de Newton resolvemos un sistema triangular. No obstante hay métodos alternos que resuelven el sistema de Vandermonde en $O(n^2)$ operaciones al aprovechar su estructura particular.

Si comparamos `interpNR` y `interpN`, el segundo método es más eficiente en el uso de la memoria de la computadora que el primero. La naturaleza recursiva de `interpNR` requiere de $O(n^2)$ lugares de memoria en comparación con $O(n)$ para `interpN`.

5.3 Diferencias Divididas de Newton

Los coeficientes (c_k) en la representación de Newton (cf. (5.9)) se conocen como las *diferencias divididas de Newton* y se denotan por:

$$f[x_1, \ldots, x_k] \equiv c_k, \quad 1 \leq k \leq n. \tag{5.13}$$

(Aquí y en adelante suponemos que $y_i = f(x_i)$, $1 \leq i \leq n$, para alguna función f.) En particular (5.9) se puede escribir como

$$p_{n-1}(x) = \sum_{k=1}^{n} f[x_1, \ldots, x_k] \prod_{j=1}^{k-1} (x - x_j). \tag{5.14}$$

Las funciones `interpNR` y `interpN` representan algoritmos para calcular las diferencias divididas. El algoritmo `interpN` se puede justificar usando la siguiente propiedad de las diferencias divididas:

$$f[x_1, \ldots, x_k] = \begin{cases} f(x_1), & k = 1, \\ \dfrac{f[x_2, \ldots, x_k] - f[x_1, \ldots, x_{k-1}]}{x_k - x_1}, & k > 1. \end{cases} \tag{5.15}$$

Otra propiedad importante de las diferencias dividas es que

$$f[x_1, \ldots, x_k] = \frac{f^{(k-1)}(\eta)}{(k-1)!}, \quad \eta \in I\{x_1, \ldots, x_k\}, \tag{5.16}$$

donde $I\{x_1, \ldots, x_k\}$ es el intervalo más pequeño que contiene a $x_1, \ldots, x_k$. Las demostraciones de ambas propiedades se dejan para los ejercicios. (Ver Ejercicios 5.7 y 5.8).

Ejemplo 5.3. Vamos a utilizar la formula (5.15) para calcular nuevamente los coeficientes de la representación de Newton de $p_3(x)$. Usando la formula generamos las diferencias divididas de acuerdo a la tabla:

$$\begin{array}{llll} f[x_1] \searrow & & & \\ & f[x_1, x_2] \searrow & & \\ f[x_2] \nearrow \searrow & & f[x_1, x_2, x_3] \searrow & \\ & f[x_2, x_3] \nearrow \searrow & & f[x_1, x_2, x_3, x_4] \\ f[x_3] \nearrow \searrow & & f[x_2, x_3, x_4] \nearrow & \\ & f[x_3, x_4] \nearrow & & \\ f[x_4] \nearrow & & & \end{array}$$

donde:

$$\begin{aligned} f[x_1, x_2] &= \frac{f(x_2) - f(x_1)}{x_2 - x_1}, \quad \text{etc.}, \\ f[x_1, x_2, x_3] &= \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}, \quad \text{etc.}, \\ f[x_1, x_2, x_3, x_4] &= \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1}. \end{aligned}$$

Para los datos $(-2, 10), (-1, 4), (1, 6)$, y $(2, 3)$ la tabla queda como sigue:

$\boxed{f[x_1] = 10}$			
	$\boxed{f[x_1, x_2] = -6}$		
$f[x_2] = 4$		$\boxed{f[x_1, x_2, x_3] = \frac{7}{3}}$	
	$f[x_2, x_3] = 1$		$\boxed{f[x_1, x_2, x_3, x_4] = -\frac{11}{12}}$
$f[x_3] = 6$		$f[x_2, x_3, x_4] = -\frac{4}{3}$	
	$f[x_3, x_4] = -3$		
$f[x_4] = 3$			

Los coeficientes de la representación de Newton de $p_3(x)$ aperecen ahora en los recuadros de ésta última tabla. □

5.4 Representación de Lagrange de p_{n-1}

La representación de Newton de p_{n-1} es la representación más eficiente, desde el punto de vista computacional, para trabajar con el polinomio de interpolación. No obstante, hay otra forma de escribir el polinomio de interpolación, llamada la *representación de Lagrange* de p_{n-1}, la cual es una excelente herramienta teórica para estudiar las propiedades de los polinomios de interpolación. Antes de definir la representación de Lagrange p_{n-1}, necesitamos lo que se conoce como los *polinomios base de Lagrange* $\{\ell_1(x), \ldots, \ell_n(x)\}$. Estos polinomios base se definen mediante:

$$\ell_k(x) = \frac{\prod_{j \neq k}(x - x_j)}{\prod_{j \neq k}(x_k - x_j)}, \quad 1 \leq k \leq n. \tag{5.17}$$

Es fácil ver que cada ℓ_k es un polinomio de grado $n-1$ y que

$$\ell_k(x_i) = \begin{cases} 1, & k = i, \\ 0, & k \neq i. \end{cases} \tag{5.18}$$

De aquí que el polinomio de interpolación se puede escribir ahora como

$$p_{n-1}(x) = \sum_{k=1}^{n} f(x_k)\ell_k(x). \tag{5.19}$$

Esta formula para p_{n-1} se llama la *representación de Lagrange* de p_{n-1}.

Ejemplo 5.4. Considere el problema de calcular el polinomio de interpolación para los datos $(-2, 4)$, $(1, 2)$, $(2, 7)$. Vamos a calcular en este caso las tres representaciones del polinomio de interpolación para ilustrar las diferencias computacionales entre éstas. Primero escribimos p_2 de la forma $p_2(x) = a_1 + a_2x + a_3x^2$. Aplicando las condiciones de interpolación vemos que ésto es equivalente al sistema:

$$\begin{pmatrix} 1 & -2 & 4 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix},$$

cuya solución es $a_1 = -\frac{1}{6}$, $a_2 = \frac{3}{4}$, $a_3 = \frac{17}{12}$. Así que tenemos la representación *canónica* de p_2 dada por:

$$p_2(x) = -\frac{1}{6} + \frac{3}{4}x + \frac{17}{12}x^2.$$

Buscamos ahora la representación de Lagrange de p_2. Primero calculamos los polinomios base ℓ_1, ℓ_2, ℓ_3:

$$\begin{aligned} \ell_1(x) &= \frac{(x-1)(x-2)}{(-2-1)(-2-2)} = \frac{1}{12}(x-1)(x-2), \\ \ell_2(x) &= \frac{(x-(-2))(x-2)}{(1-(-2))(1-2)} = -\frac{1}{3}(x^2-4), \\ \ell_3(x) &= \frac{(x-(-2))(x-1)}{(2-(-2))(2-1)} = \frac{1}{4}(x+2)(x-1). \end{aligned}$$

Tenemos pues que la representación de Lagrange de p_2 está dada por:

$$p_2(x) = \frac{1}{3}(x-1)(x-2) - \frac{2}{3}(x^2-4) + \frac{7}{4}(x+2)(x-1).$$

Finalmente calculamos la representación de Newton de p_2, esto es buscamos c_1, c_2, c_3 tal que

$$p_2(x) = c_1 + c_2(x+2) + c_3(x+2)(x-1).$$

Aplicando las condiciones de interpolación vemos que ésto es equivalente al sistema:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 3 & 0 \\ 1 & 4 & 4 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix},$$

cuya solución es $c_1 = 4$, $c_2 = -\frac{2}{3}$, $c_3 = \frac{17}{12}$. Tenemos entonces que la representación de Newton de p_2 está dada por:

$$p_2(x) = 4 - \frac{2}{3}(x+2) + \frac{17}{12}(x+2)(x-1).$$

Considere ahora el problema de hallar el polinomio de interpolación si se añade el punto $(-1, -2)$ a los datos originales. En general los coeficientes en la representación canónica y la base de Lagrange tienen que ser re-calculados completamente en éste caso. De hecho, la representación canónica de p_3 para el conjunto de los cuatro datos es:

$$p_3(x) = -\frac{11}{6} + \frac{29}{12}x + \frac{11}{6}x^2 - \frac{5}{12}x^3.$$

Veamos que sucede con la representación de Newton. El nuevo polinomio p_3 tiene la forma:

$$p_3(x) = \hat{c}_1 + \hat{c}_2(x+2) + \hat{c}_3(x+2)(x-1) + \hat{c}_4(x+2)(x-1)(x-2),$$

para algunos $\hat{c}_1, \hat{c}_2, \hat{c}_3, \hat{c}_4$. Aplicando las condiciones de interpolación obtenemos el sistema:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 1 & 4 & 4 & 0 \\ 1 & 1 & -2 & 6 \end{pmatrix} \begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \hat{c}_3 \\ \hat{c}_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 7 \\ -2 \end{pmatrix}.$$

Note que las primeras tres ecuaciones de éste sistema coinciden con las del sistema para la representación de Newton de p_2. De modo que $\hat{c}_1 = c_1$, $\hat{c}_2 = c_2$, $\hat{c}_3 = c_3$, i.e., los primeros tres coeficientes en la representación de

Newton de p_3 son los mismos coeficientes que en la representación de Newton de p_2. De la cuarta ecuación del sistema anterior y usando los valores para los primeros tres coeficientes, tenemos que:

$$4 + \left(-\frac{2}{3}\right) - 2\left(\frac{17}{12}\right) + 6\hat{c}_4 = -2,$$

i.e., $\hat{c}_4 = -\frac{5}{12}$. Tenemos ahora que la representación de Newton de p_3 está dada por:

$$\begin{aligned} p_3(x) &= 4 - \frac{2}{3}(x+2) + \frac{17}{12}(x+2)(x-1) - \frac{5}{12}(x+2)(x-1)(x-2), \\ &= p_2(x) - \frac{5}{12}(x+2)(x-1)(x-2). \end{aligned}$$

(Vea el Ejercicio 5.9 para una generalización de este procedimiento.) □

5.5 Error de Interpolación

Vamos a estudiar ahora cuán bien aproxima el polinomio de interpolación p_{n-1} a la función f de la cual provienen los datos de interpolación. El resultado que vamos a ver, se puede considerar como una generalización del Teorema de Taylor (Teorema 1.2). De hecho, en el límite según los puntos $\{x_1, \ldots, x_n\}$ colapsan en el punto x_1, el resultado de abajo se reduce al Teorema de Taylor.

Teorema 5.5. *Suponga que p_{n-1} interpola a la función f en los puntos x_1, x_2,..., x_n, todos distintos, y que f tiene derivadas continuas hasta orden n en un intervalo I el cual contiene a $x_1, \ldots, x_n$. Entonces para cualquier $z \in I$ existe un η entre z y $x_1, \ldots, x_n$ tal que:*

$$f(z) - p_{n-1}(z) = \frac{f^{(n)}(\eta)}{n!}(z - x_1)\cdots(z - x_n). \tag{5.20}$$

Aunque el valor η que aparece en el teorema es en general desconocido, en muchas ocasiones es posible usar el teorema para obtener estimados prácticos del error de interpolación.

Ejemplo 5.6. Considere la función $f(x) = \mathrm{e}^x$, $x \in [0, 1]$ y sean x_1, x_2 puntos en $[0, 1]$. Entonces como $f''(x) = \mathrm{e}^x$ tenemos por el Teorema 5.5 que

$$f(x) - p_1(x) = \frac{1}{2}(x - x_1)(x - x_2)\mathrm{e}^{\eta(x)},$$

donde $\eta(x)$ está entre x, x_1, x_2. Vamos a suponer que $x_1 \leq x \leq x_2$ y sea $h = x_2 - x_1$. Entonces

$$|f(x) - p_1(x)| = \frac{1}{2}(x - x_1)(x_2 - x)\mathrm{e}^{\eta(x)} \leq \frac{h^2}{8}\,\mathrm{e}^{\eta(x)},$$

donde usamos que

$$\max_{x_1 \leq x \leq x_2} (x - x_1)(x_2 - x) = \frac{h^2}{4}.$$

Como $\eta(x) \in [x_1, x_2] \subset [0, 1]$, tenemos que $\mathrm{e}^{\eta(x)} \leq \mathrm{e} < 3$. Tenemos pues que

$$|f(x) - p_1(x)| \leq \frac{3h^2}{8}, \quad x \in [x_1, x_2].$$

Continuando con el ejemplo, si usamos ahora interpolación cuadrática con $x_1 < x_2 < x_3$, tenemos que

$$f(x) - p_2(x) = \frac{1}{6}(x - x_1)(x - x_2)(x - x_3)\mathrm{e}^{\eta(x)},$$

donde ahora $\eta(x)$ está entre x, x_1, x_2, x_3. Suponiendo que $h = x_3 - x_2 = x_2 - x_1$, tenemos que

$$\max_{x_1 \leq x \leq x_3} \left| \frac{1}{6}(x - x_1)(x - x_2)(x - x_3) \right| = \frac{h^3}{9\sqrt{3}}.$$

Esto combinado con el estimado $\mathrm{e}^{\eta(x)} < 3$ para $x \in [x_1, x_3] \subset [0, 1]$, nos da que

$$|f(x) - p_2(x)| \leq \frac{h^3}{3\sqrt{3}}, \quad x \in [x_1, x_3].$$

□

5.5.1 Aproximación en intervalos

Los estimados del Ejemplo 5.6 son para n fijo, es decir, el número de puntos de interpolación se mantiene constante (solo cambian las distancias relativas entre ellos). Consideramos ahora el caso de un intervalo fijo $[a, b]$ y una partición $x_1, \ldots, x_n$ del mismo. Queremos estudiar el comportamiento de p_{n-1} en $[a, b]$ en el límite según n aumenta indefinidamente. El Teorema 5.5 puede dar la impresión equivocada de que p_{n-1} debe aproximar a la función f

cada vez mejor según la n aumenta. Como veremos esto depende de la función f y del carácter de la distribución de puntos $x_1, \ldots, x_n$ en el intervalo $[a, b]$.

Sea f una función en $[a, b]$ y $x_1, \ldots, x_n$ una partición *uniforme* de $[a, b]$, i.e.,

$$x_i = a + (i-1)h, \quad 1 \leq i \leq n, \quad h = (b-a)/(n-1). \tag{5.21}$$

Entonces se puede demostrar que:

$$|f(z) - p_{n-1}(z)| \leq \frac{h^n}{n} \max_{x \in [a,b]} |f^{(n)}(x)|. \tag{5.22}$$

(Ver Ejercicio 5.15.) Si además

$$\max_{x \in [a,b]} |f^{(n)}(x)| \leq M, \tag{5.23}$$

donde M es una constante independiente de n, entonces el error de interpolación tiende a cero según $n \longrightarrow \infty$, i.e., hay *convergencia*. Este es el caso, por ejemplo para las funciones trigonométricas sen y cos, y para la exponencial e^x en cualquier intervalo acotado.

Esta propiedad de derivadas de todo orden acotadas uniformemente (c.f. (5.23)), no la tienen todas las funciones. El ejemplo clásico de este tipo de *mal* comportamiento en las derivadas, es el de la función de Runge dada por:

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1]. \tag{5.24}$$

El siguiente programa en MATLAB calcula los polinomios de interpolación de f de grados $9, 10, 11$, y 12 para los puntos (5.21) en el intervalo $[-1, 1]$, y los traza en el mismo sistema de coordenadas junto con f:

```
x=linspace(-1,1,100)';
y=1./(1+25*x.^2);
k=0;
for n=10:13
k=k+1;
xunif=linspace(-1,1,n)';
yunif=1./(1+25*xunif.^2);
cunif=interpN(xunif,yunif);
pvals=hornerN(cunif,xunif,x);
subplot(2,2,k)
```

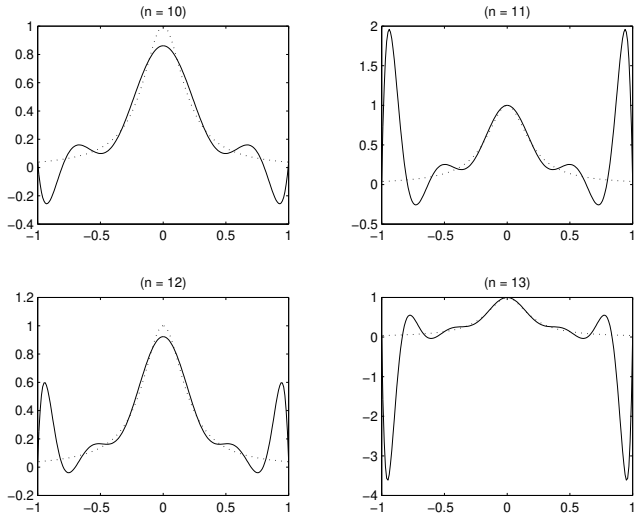


Figura 5.2: Polinomios de interpolación (gráficas sólidas) de grados 9, 10, 11, y 12 en puntos uniformemente distribuidos para la función de Runge (5.24) (gráfica entrecortada).

```
plot(x,y,x,pvals)
title(sprintf('(n = %2.0f)',n))
end
```

Note que el error (5.20) se hace arbitrariamente grande en los extremos del intervalo según el grado del polinomio aumenta. De hecho se puede demostrar (ver [11]) que

$$\lim_{n\to\infty} \max_{x\in[-1,1]} |f(x) - p_{n-1}(x)| = \infty, \tag{5.25}$$

para los puntos (5.21) y la función (5.24). Por esta razón la interpolación de polinomios de alto grado no se utiliza comúnmente en la práctica. El grado máximo del polinomio de interpolación se limita usualmente a cinco o seis.

Si los puntos de interpolación, es decir, la partición $x_1, \ldots, x_n$ de $[a, b]$, no es necesariamente uniforme, entonces ésta se puede seleccionar de forma que haya convergencia, i.e.,

$$\lim_{n\to\infty} \max_{x\in[-1,1]} |f(x) - p_{n-1}(x)| = 0. \tag{5.26}$$

Este es el caso cuando los puntos se seleccionan como los ceros de los *polinomios de Chebychev* (ver [11], [31]).

5.6 Interpolación Polinomial por Pedazos

El uso de polinomios de interpolación de alto grado puede producir errores grandes debido al alto grado de oscilación de este tipo de polinomios. Para evitar este problema, el intervalo de aproximación se subdivide en intervalos pequeños en los cuales la función desconocida se aproxima usando polinomios de grado bajo. Esto se conoce como *interpolación polinomial por pedazos*. El caso más común de la interpolación polinomial por pedazos es cuando se usan polinomios cúbicos.

5.6.1 Funciones cúbicas por pedazos de Lagrange

Suponemos que $n-1$ es divisible entre tres y que la partición es uniforme, esto es, $h = x_{i+1} - x_i$ es constante para toda i. Ahora dividimos la partición en grupos de cuatro puntos cada uno. Si $x_{3j-2}, x_{3j-1}, x_{3j}, x_{3j+1}$ es uno de esos grupos, donde $1 \leq j \leq (n-1)/3$, entonces buscamos un polinomio cúbico que interpola a la función en dichos puntos. Este polinomio está dado en su representación de Lagrange por:

$$\begin{aligned} p(x) &= -\frac{1}{6h^3}(x - x_{3j-1})(x - x_{3j})(x - x_{3j+1})y_{3j-2} \\ &\quad + \frac{1}{2h^3}(x - x_{3j-2})(x - x_{3j})(x - x_{3j+1})y_{3j-1} \\ &\quad - \frac{1}{2h^3}(x - x_{3j-2})(x - x_{3j-1})(x - x_{3j+1})y_{3j} \\ &\quad + \frac{1}{6h^3}(x - x_{3j-2})(x - x_{3j-1})(x - x_{3j})y_{3j+1}, \end{aligned}$$

$x_{3j-2} \leq x \leq x_{3j+1}$. Este proceso se repite para cada grupo de tres puntos, obteniendo así una función cúbica por pedazos (la cual en general no es globalmente un polinomio). Las funciones cúbicas por pedazos de Lagrange tienen la desventaja de que en general no son diferenciables en los puntos donde se unen o pegan dichos polinomios.

Ejemplo 5.7. Considere la función $f(x) = x + \text{sen}(x)$ para $x \in [0, 5\pi]$. Para una partición uniforme de $[0, 5\pi]$ con siete puntos, construimos el polinomio cúbico por pedazos de Lagrange que interpola a la función en los datos de la partición. El siguiente programa en MATLAB hace este cálculo:

```
x=linspace(0,5*pi,7);
```

```
y=x+sin(x);
a1=interpV(x(1:4)',y(1:4)');
a2=interpV(x(4:7)',y(4:7)');
x1=linspace(x(1),x(4),40);
x2=linspace(x(4),x(7),40);
p1=hornerV(a1,x1);
p2=hornerV(a2,x2);
z=linspace(0,5*pi,80);
y1=z+sin(z);
plot(z,y1,'k',x1,p1,'k:',x2,p2,'k:')
```

En la Figura 5.3 mostramos las gráficas de f y el correspondiente polinomio cúbico por pedazos. Note que la función de interpolación no es diferenciable ya que tiene una esquina en el cuarto punto de la interpolación. □

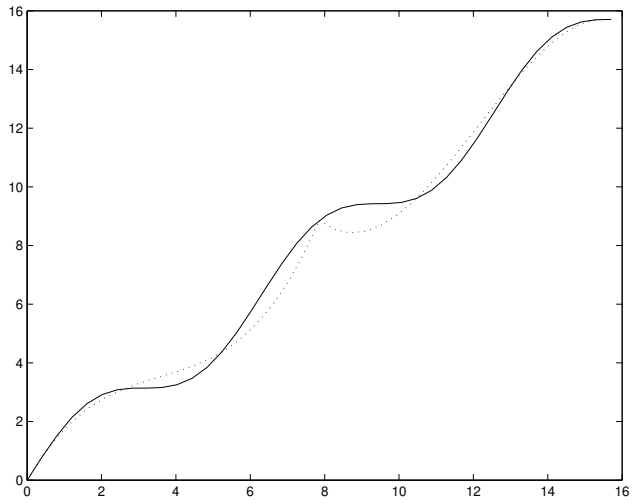


Figura 5.3: Gráficas de las funciones $f(x) = x+\text{sen}(x)$ (sólida) y el polinomio cúbico por pedazos de Lagrange (entrecortada) en una partición uniforme de $[0, 5\pi]$ con siete puntos.

5.6.2 Interpolación de Hermite

En este tipo de interpolación, buscamos una función $Q_n(x)$ que sea cúbica en cada subintervalo $[x_i, x_{i+1}]$, $1 \leq i \leq n-1$, y que interpole a $f(x)$ y

$f'(x)$ en los puntos $\{x_1, \ldots, x_n\}$. La función $Q_n(x)$ se puede determinar en forma única a partir de éstas condiciones. El cálculo de $Q_n(x)$ requiere de la solución de $n-1$ sistemas lineales, uno en cada intervalo de la partición, de tamaño 4×4 cada uno. (Vea el Ejercicio 5.20.) De hecho, si escribimos

$$Q_n(x) = a_i + b_i x + c_i x^2 + d_i x^3, \quad x \in [x_i, x_{i+1}], \quad 1 \leq i \leq n-1,$$

entonces la condiciones

$$Q_n(x_i) = f(x_i), \quad Q_n'(x_i) = f'(x_i), \quad 1 \leq i \leq n,$$

implican que

$$\begin{pmatrix} 1 & x_i & x_i^2 & x_i^3 \\ 0 & 1 & 2x_i & 3x_i^2 \\ 1 & x_{i+1} & x_{i+1}^2 & x_{i+1}^3 \\ 0 & 1 & 2x_{i+1} & 3x_{i+1}^2 \end{pmatrix} \begin{pmatrix} a_i \\ b_i \\ c_i \\ d_i \end{pmatrix} = \begin{pmatrix} f(x_i) \\ f'(x_i) \\ f(x_{i+1}) \\ f'(x_{i+1}) \end{pmatrix}, \tag{5.27}$$

donde $1 \leq i \leq n-1$.

La interpolación de Hermite tiene dos desventajas:

i) utiliza las derivadas $\{f'(x_i) : 1 \leq i \leq n\}$ las cuales no están disponibles en muchas aplicaciones;

ii) y en general, Q_n' no es diferenciable en los puntos de interpolación.

Ejemplo 5.8. Considere nuevamente la función $f(x) = x + \text{sen}(x)$ para $x \in [0, 5\pi]$. Para una partición uniforme de $[0, 5\pi]$ con tres puntos, construimos la función cúbica por pedazos de Hermite que interpola a la función en los datos de la partición. Esto lo hacemos con el siguiente programa en MATLAB el cual utiliza las ecuaciones (5.27):

```
x=linspace(0,5*pi,3);
y=x+sin(x);
yp=1+cos(x);
A1=[1 x(1) x(1)^2 x(1)^3;0 1 2*x(1) 3*x(1)^2;...
      1 x(2) x(2)^2 x(2)^3;0 1 2*x(2) 3*x(2)^2];
A2=[1 x(2) x(2)^2 x(2)^3;0 1 2*x(2) 3*x(2)^2;...
      1 x(3) x(3)^2 x(3)^3;0 1 2*x(3) 3*x(3)^2];
a1=A1\[y(1) yp(1) y(2) yp(2)]';
a2=A2\[y(2) yp(2) y(3) yp(3)]';
```

```
x1=linspace(x(1),x(2),40);
x2=linspace(x(2),x(3),40);
p1=hornerV(a1,x1);
p2=hornerV(a2,x2);
z=linspace(0,5*pi,80);
y1=z+sin(z);
plot(z,y1,'k',x1,p1,'k:',x2,p2,'k:')
```

En la Figura 5.4 se muestran las gráficas de f y Q_3 donde podemos observar que Q_3 es diferenciable. □

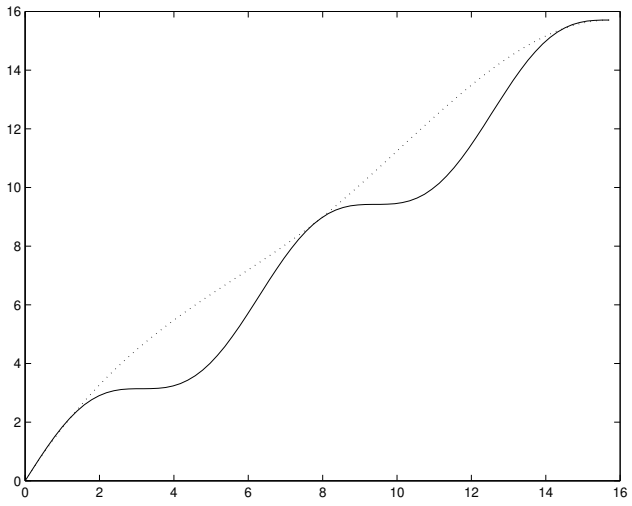


Figura 5.4: Gráficas de las funciones $f(x) = x + \text{sen}(x)$ (sólida) y el polinomio cúbico por pedazos de Hermite (entrecortada) en una partición uniforme de $[0, 5\pi]$ con tres puntos.

5.6.3 Interpolación usando Splines

Los dos tipos de funciones polinomiales por pedazos que hemos discutido hasta el momento, tienen la desventaja de que su segunda derivada no es continua en los puntos de interpolación. Se ha observado que en aplicaciones gráficas, el ojo humano es capaz de detectar discontinuidades en la segunda derivada de una función, haciendo que los gráficos con este tipo de funciones

no luzcan uniformes. Esto motiva el uso de los *splines* que son funciones $s(x)$ polinomiales por pedazos con las siguientes propiedades:

1. $s(x)$ es un polinomio cúbico en $[x_i, x_{i+1}]$, $1 \leq i \leq n-1$.
2. $s'(x)$, $s''(x)$ existen y son continuas en $[x_1, x_n]$.
3. $s(x)$ interpola a la función f en los datos $\{(x_i, y_i) : 1 \leq i \leq n\}$.
4. $s(x)$ es continua en $[x_1, x_n]$.

Si escribimos $s(x) = a_i + b_i x + c_i x^2 + d_i x^3$, $x \in [x_i, x_{i+1}]$, $1 \leq i \leq n-1$, entonces tenemos un total de $4(n-1)$ desconocidas. Las condiciones (2) y (4) nos dan $3(n-2)$ ecuaciones mientras que de (3) obtenemos n para un total de $4(n-1) - 3(n-2) - n = 2$ grados de libertad. Estos grados de libertad se fijan imponiendo condiciones de frontera adicionales en $s(x)$. Dos tipos de condiciones de frontera comunes son:

i) **Condición Natural:** Se supone que $s''(x_1) = 0 = s''(x_n)$. Esto es consistente con suponer que el spline es lineal en $(-\infty, x_1]$ y $[x_n, \infty)$.

ii) **Condición "not a knot":** Aquí suponemos que s''' es continua en x_2 y x_{n-1}

Ejemplo 5.9. Vamos a construir el spline cúbico natural que interpola a los datos $(-5, 20)$, $(-2, 4)$, $(3, -6)$, $(7, 40)$. La función $s(x)$ que buscamos es de la forma:

$$s(x) = \begin{cases} a_1x^3 + b_1x^2 + c_1x + d_1 & , \quad x \in [-5, -2], \\ a_2x^3 + b_2x^2 + c_2x + d_2 & , \quad x \in [-2, 3], \\ a_3x^3 + b_3x^2 + c_3x + d_3 & , \quad x \in [3, 7]. \end{cases} \tag{5.28}$$

Las condiciones de interpolación y continuidad de $s(x)$ resultan en las siguientes ecuaciones:

$$\begin{aligned} -125a_1 + 25b_1 - 5c_1 + d_1 &= 20, \\ -8a_1 + 4b_1 - 2c_1 + d_1 &= 4, \\ -8a_2 + 4b_2 - 2c_2 + d_2 &= 4, \\ 27a_2 + 9b_2 + 3c_2 + d_2 &= -6, \\ 27a_3 + 9b_3 + 3c_3 + d_3 &= -6, \end{aligned}$$

$$343a_3 + 49b_3 + 7c_3 + d_3 = 40.$$

La derivada $s'(x)$ tiene la representación:

$$s'(x) = \begin{cases} 3a_1x^2 + 2b_1x + c_1 & , \quad x \in [-5, -2], \\ 3a_2x^2 + 2b_2x + c_2 & , \quad x \in [-2, 3], \\ 3a_3x^2 + 2b_3x + c_3 & , \quad x \in [3, 7]. \end{cases}$$

La condición de que ésta derivada sea continua nos dá las ecuaciones:

$$\begin{aligned} 12a_1 - 4b_1 + c_1 &= 12a_2 - 4b_2 + c_2, \\ 27a_2 + 6b_2 + c_2 &= 27a_3 + 6b_3 + c_3. \end{aligned}$$

La segunda derivada $s''(x)$ tiene la representación:

$$s''(x) = \begin{cases} 6a_1x + 2b_1 & , \quad x \in [-5, -2], \\ 6a_2x + 2b_2 & , \quad x \in [-2, 3], \\ 6a_3x + 2b_3 & , \quad x \in [3, 7]. \end{cases}$$

La condición de que ésta derivada sea también continua nos dá las ecuaciones:

$$\begin{aligned} -12a_1 + 2b_1 &= -12a_2 + 2b_2, \\ 18a_2 + 2b_2 &= 18a_3 + 2b_3. \end{aligned}$$

Las condiciones naturales de que $s''(-5) = 0$, $s''(7) = 0$ nos dan dos ecuaciones más:

$$\begin{aligned} -30a_1 + 2b_1 &= 0, \\ 42a_3 + 2b_3 &= 0. \end{aligned}$$

Tenemos entonces un sistema de 12 ecuaciones en 12 desconocidas que en forma matricial se puede escribir como:

$$\begin{bmatrix} -125 & 25 & -5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 4 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -8 & 4 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 27 & 9 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 27 & 9 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 343 & 49 & 7 & 1 \\ 12 & -4 & 1 & 0 & -12 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 27 & 6 & 1 & 0 & -27 & -6 & -1 & 0 \\ -12 & 2 & 0 & 0 & 12 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 18 & 2 & 0 & 0 & -18 & -2 & 0 & 0 \\ -30 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 42 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \\ a_3 \\ b_3 \\ c_3 \\ d_3 \end{bmatrix} = \begin{bmatrix} 20 \\ 4 \\ 4 \\ -6 \\ -6 \\ 40 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Este sistema lo podemos resolver en MATLAB utilizando el paquete simbólico para así obtener soluciones exactas. Sustituyendo en la ecuación (5.28) los valores de los a_i's, b_i's, y c_i's que resultan, obtenemos que el spline queda finalmente dado por:

$$s(x) = \begin{cases} -\frac{5}{526}x^3 - \frac{75}{526}x^2 - \frac{4703}{789}x - \frac{5860}{789} & , \quad x \in [-5,-2], \\ \frac{1241}{7890}x^3 + \frac{2257}{2630}x^2 - \frac{15619}{3945}x - \frac{8012}{1315} & , \quad x \in [-2,3], \\ -\frac{299}{1578}x^3 + \frac{2093}{526}x^2 - \frac{10511}{789}x + \frac{860}{263} & , \quad x \in [3,7]. \end{cases}$$

□

Veamos ahora como se construye $s(x)$ en general. Definimos $M_i = s''(x_i)$, $1 \leq i \leq n$. Dado que $s(x)$ es cúbico en $[x_i, x_{i+1}]$, entonces $s''(x)$ es lineal en $[x_i, x_{i+1}]$ y está dado por:

$$s''(x) = \frac{(x_{i+1} - x)M_i + (x - x_i)M_{i+1}}{h_i}, \quad h_i = x_{i+1} - x_i, \tag{5.29}$$

donde $1 \leq i \leq n-1$. Integrando ésta ecuación dos veces, obtenemos que

$$s(x) = \frac{(x_{i+1} - x)^3 M_i + (x - x_i)^3 M_{i+1}}{6h_i} + C_i(x_{i+1} - x) + D_i(x - x_i),$$

donde $1 \leq i \leq n-1$, $x \in [x_i, x_{i+1}]$. Las condiciones $s(x_i) = y_i$, $1 \leq i \leq n$ implican que

$$C_i = \frac{y_i}{h_i} - \frac{h_i M_i}{6}, \quad D_i = \frac{y_{i+1}}{h_i} - \frac{h_i M_{i+1}}{6}, \quad 1 \leq i \leq n-1.$$

Sustituyendo ésto en la fórmula anterior de $s(x)$ obtenemos que

$$\begin{aligned} s(x) &= \frac{(x_{i+1} - x)^3 M_i + (x - x_i)^3 M_{i+1}}{6h_i} + \frac{(x_{i+1} - x)y_i + (x - x_i)y_{i+1}}{h_i} \\ &\quad -\frac{h_i}{6}[(x_{i+1} - x)M_i + (x - x_i)M_{i+1}], \end{aligned} \tag{5.30}$$

donde $1 \leq i \leq n-1$, $x \in [x_i, x_{i+1}]$. Note que $s(x)$ y $s''(x)$ son continuas por construcción y además tenemos que la condición de interpolación se cumple. Falta la condición de que $s'(x)$ sea continua, i.e.,

$$\lim_{x \to x_i^+} s'(x) = \lim_{x \to x_i^-} s'(x), \quad 1 < i < n.$$

En $[x_i, x_{i+1}]$, tenemos que

$$s'(x) = \frac{-(x_{i+1}-x)^2 M_i + (x-x_i)^2 M_{i+1}}{2h_i} + \frac{y_{i+1}-y_i}{h_i} - \frac{M_{i+1}-M_i}{6} h_i.$$

Usando una expresión similar en $[x_{i-1}, x_i]$, se obtiene que la condición de continuidad para $s'(x)$, expresada por los límites de derecha e izquierda de arriba, implica que

$$\frac{h_i}{6} M_i + \frac{h_i + h_{i+1}}{3} M_{i+1} + \frac{h_{i+1}}{6} M_{i+2} = y'_{i+1} - y'_i, \quad 1 \le i \le n-2, \qquad (5.31)$$

donde

$$y'_i = \frac{y_{i+1}-y_i}{h_i}, \quad 1 \le i \le n-1.$$

Tenemos aquí $n-2$ ecuaciones para las n desconocidas $M_1, \ldots, M_n$.

Condición Natural de los Splines

Para fijar los dos grados de libertad en el spline, requerimos que $s(x)$ sea lineal en los intervalos $(-\infty, x_1]$ y $[x_n, \infty)$ lo cual es equivalente a las condiciones $M_1 = M_n = 0$. Tenemos ahora el sistema de ecuaciones $Am = d$ para las restantes desconocidas $M_2, \ldots, M_{n-1}$, donde:

$$A = \begin{pmatrix} \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & 0 & \cdots & 0 \\ \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & 0 & \cdots & 0 \\ 0 & \frac{h_3}{6} & \frac{h_3+h_4}{3} & \frac{h_4}{6} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{h_{n-3}}{6} & \frac{h_{n-3}+h_{n-2}}{3} & \frac{h_{n-2}}{6} \\ 0 & 0 & \cdots & 0 & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} \end{pmatrix},$$

$$m = \begin{pmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-1} \end{pmatrix}, \quad d = \begin{pmatrix} y_2' - y_1' \\ y_3' - y_2' \\ y_4' - y_3' \\ \vdots \\ y_{n-1}' - y_{n-2}' \end{pmatrix}.$$

Note que la matriz de coeficientes de este sistema es tridiagonal y simétrica lo que hace que el spline $s(x)$ pueda ser calculado de forma bien eficiente. El siguiente programa en MATLAB ensambla la matriz A y el lado derecho según definidos arriba y resuelve el sistema para determinar $M_2, \ldots, M_{n-1}$. Los datos están dados por los vectores $x = [x_1, \ldots, x_n]$, $y = [y_1, \ldots, y_n]$:

```
function m=Mspline(x,y)
n=length(x);
dx=x(2:n)-x(1:n-1);
yp=(y(2:n)-y(1:n-1))./dx;
A=sparse([1:n-2],[1:n-2],(dx(1:n-2)+dx(2:n-1))/3,n-2,n-2);
if n>3
   udiag=sparse([1:n-3],[2,n-2],dx(2:n-2)/6,n-2,n-2);
   A=udiag'+A+udiag;
end
d=yp(2:n-1)-yp(1:n-2);
m=A\d;
```

El resultado de este programa se utiliza ahora en las ecuaciones (5.30) para ensamblar $s(x)$.

Ejemplo 5.10. Vamos a construir el spline cúbico (natural) que interpola a los datos $(-1, 3), (2, -2), (4, 5)$. Usamos la función `Mspline` que acabamos de describir. Los resultados se muestran en la Figura 5.5.

```
x=[-1,2,4];
y=[3,-2,5];
h1=x(2)-x(1);
h2=x(3)-x(2);
m=Mspline(x,y);
m=[0 m' 0];
x1=linspace(x(1)-1,x(2),40);
x2=linspace(x(2),x(3)+1,40);
s1=(m(1)*(x(2)-x1).^3+m(2)*(x1-x(1)).^3)/(6*h1)+...
   ((x(2)-x1)*y(1)+(x1-x(1))*y(2))/h1-...
```

```
   (h1/6)*(m(1)*(x(2)-x1)+m(2)*(x1-x(1)));
s2=(m(2)*(x(3)-x2).^3+m(3)*(x2-x(2)).^3)/(6*h2)+...
   ((x(3)-x2)*y(2)+(x2-x(2))*y(3))/h2-...
   (h2/6)*(m(2)*(x(3)-x2)+m(3)*(x2-x(2)));
plot(x1,s1,'k',x2,s2,'k',x,y,'k+')
```

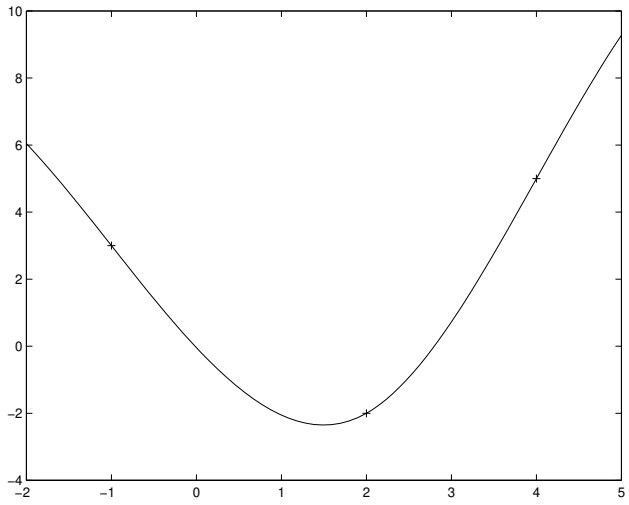


Figura 5.5: Gráfica del spline cubico natural que interpola los datos $(-1, 3)$, $(2, -2)$, $(4, 5)$.

□

Las funciones `spline` y `ppval` de MATLAB facilitan el proceso de calcular y evaluar el spline $s(x)$. `spline` calcula esencialmente la misma información que la función `Mspline` descrita antes pero utiliza la condición de frontera "not a knot" en lugar de la natural. `ppval` utiliza la información que devuelve `spline` para entonces evaluar el spline en los puntos deseados.

Ejemplo 5.11. En este ejemplo los datos se obtienen dividiendo el intervalo $[-5, 5]$ en cinco subintervalos y evaluando la función `atan` (tangente inversa) en los puntos de la partición. Luego usando la función `spline` construimos el spline cúbico que interpola en estos puntos, evaluamos el spline en 100 puntos usando la función `ppval`, y finalmente lo trazamos (Figura 5.6) junto con la función `atan` y los datos:

```
x=linspace(-5,5,6);
```

```
y=atan(x);
pp=spline(x,y);
z=linspace(-6,6,100);
sval=ppval(pp,z);
y1=atan(z);
plot(z,sval,'k',z,y1,'k',x,y,'k+')
legend('Spline','Atan','Datos')
xlabel('x'); ylabel('y');
```

□

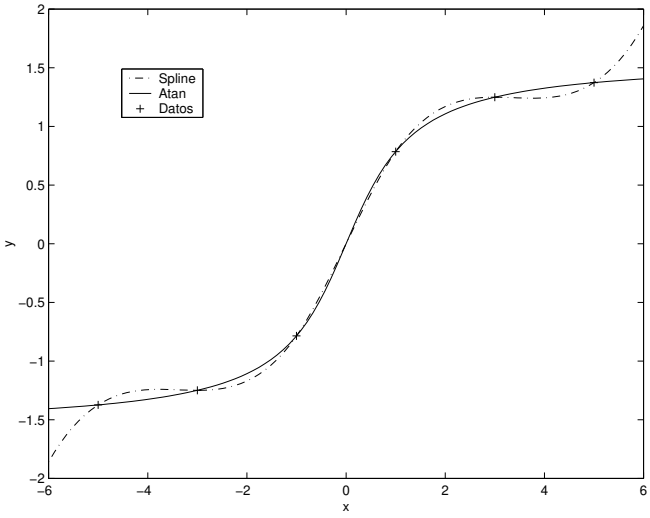


Figura 5.6: Spline $s(x)$ que interpola la función `atan` en los datos `linspace(-5,5,6)`.

En ocasiones es de interés obtener los coeficientes del spline que genera la función `spline`. Para ésto se utiliza la función `unmkpp`. La secuencia de llamada

```
pp=spline(x,y);
[z,c]=unmkpp(pp);
```

produce el arreglo `z` que es igual a `x`, y la matriz `c` cuyas filas contienen los coeficientes de los polinomios que forman el spline. En particular, el polinomio del intervalo $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n-1$, está dado por:

$$s(x) = c(i,1)(x - x_i)^3 + c(i,2)(x - x_i)^2 + c(i,3)(x - x_i) + c(i,4).$$

Ejemplo 5.12. Considere los datos $(-2, 40)$, $(-1, 7)$, $(1, -5)$, $(3, -20)$. Primero construimos en MATLAB el spline que interpola estos datos:

```
x=[-2 -1 1 3];
y=[40 7 -5 -20];
pp=spline(x,y);
```

Calculamos ahora los coeficientes de los tres polinomios cúbicos que componen el spline:

```
[z,c]=unmkpp(pp);
```

La matriz `c` está dada por:

```
c =

   -1.8750   16.5000  -47.6250   40.0000
   -1.8750   10.8750  -20.2500    7.0000
   -1.8750   -0.3750    0.7500   -5.0000
```

Las tres filas de `c` nos dan los coeficientes de los polinomios cúbicos que componen el spline para los intervalos $[-2, -1]$, $[-1, 1]$, y $[1, 3]$ respectivamente. Por ejemplo, el polinomio del intervalo $[-1, 1]$ es:

$$\begin{aligned} s(x) &= -1.8750\,(x+1)^3 + 10.8750\,(x+1)^2 \\ &\quad -20.2500\,(x+1) + 7.0000, \quad x \in [-1, 1]. \end{aligned}$$

□

5.6.4 Aplicación – Cómputo de funciones inversas

Suponga que una cierta función $y = f(x)$ para $x \in I$ (I intervalo), tiene función[1] inversa f^{-1}. Tenemos entonces que f y f^{-1} satisfacen la siguiente relación:

$$y = f(x) \quad \text{si y solo si} \quad x = f^{-1}(y) \quad \text{para toda } x \in I.$$

Esta propiedad nos permite diseñar un método numérico para aproximar la función inversa f^{-1}. En particular, suponga que tenemos los datos de la función f:

$$(x_i, f(x_i)), \quad 1 \leq i \leq n,$$

[1]Recuerde que una condición necesaria y suficiente para ésto es que f sea estrictamente creciente o decreciente en su dominio.

donde $x_i \in I$ para toda i. Entonces por la propiedad anterior:

$$(f(x_i), x_i), \quad 1 \leq i \leq n,$$

es un conjunto de datos para la función inversa f^{-1}. Podemos ahora aproximar f^{-1} utilizando construyendo un spline para este segundo grupo de datos.

Ejemplo 5.13. La función $f(x) = x^3 + x + 1$ es estrictamente creciente por lo que tiene una función inversa f^{-1}. Vamos a aproximar a f^{-1} para valores de x en el intervalo $I = [-1.2, 1]$ utilizando valores de la función f en 20 puntos de este intervalo. Usamos el siguiente código en MATLAB:

```
x=linspace(-1.2,1,20);
y=x.^3+x+1;
ppinv=spline(y,x);
xx=linspace(-1.2,1,100);
yy=xx.^3+xx+1;
xxinv=linspace(min(y),max(y),100);
yyinv=ppval(ppinv,xxinv);
xxaxis=linspace(-2,3,40);
plot(xxaxis,xxaxis,'k:',xx,yy,'k',xxinv,yyinv,'k-.');
legend('y=x','y=f(x)','y=f^-1(x)')
axis equal
axis([-2,3,-2,3])
```

Note que las primeras dos instrucciones de este programa calculan 20 puntos de la gráfica de $f(x) = x^3 + x + 1$, mientras que con la tercera instrucción calculamos el spline que aproxima a f^{-1}. Las restantes instrucciones en el programa evalúan a f y al spline que aproxima a f^{-1} en 100 puntos de sus correspondientes dominios para luego trazar sus gráficas. Las gráficas correspondientes a la recta $y = x$ y las funciones $y = f(x)$, y $y = f^{-1}(x)$ se muestran en la Figura 5.7. □

5.7 Problemas de Cuadrados Mínimos Polinomiales

Consideramos ahora el problema de aproximar o *ajustar* una función a un número grande de datos que contienen posiblemente un cierto grado de incertidumbre. En lugar de tratar de ajustar un polinomio de alto grado o insistir en interpolar datos que sabemos tienen un cierto grado de error, lo que hacemos es que buscamos una función que en cierto sentido suavice las fluctuaciones en los datos y a la vez resalte

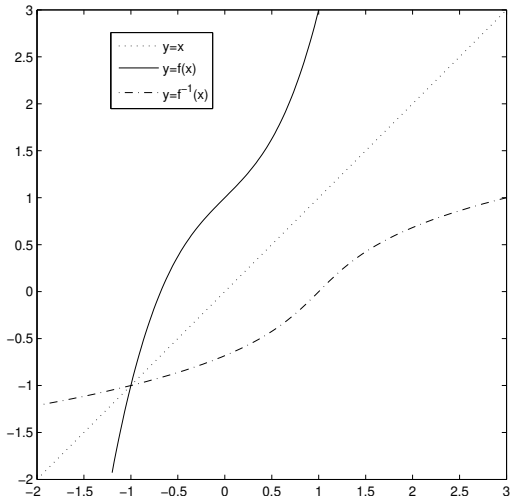


Figura 5.7: La función $f(x) = x^3 + x + 1$, $x \in [-1.2, 1]$ y el spline que aproxima su función inversa f^{-1}.

las características esenciales de éstos. El *método de cuadrados mínimos* se diseña precisamente con ésto en mente. En este método se minimiza la raíz cuadrada del promedio de los cuadrados de las diferencias entre los datos y la función que se usa para aproximar éstos[2].

Suponga que los datos están dados por (x_k, y_k) donde $k = 1, 2, \ldots, m$. La función que usamos para aproximar éstos datos tiene la forma general:

$$g(x) = a_1\phi_1(x) + a_2\phi_2(x) + \cdots + a_n\phi_n(x), \tag{5.32}$$

donde las funciones $\{\phi_i : 1 \leq i \leq n\}$ son dadas y se llaman las *funciones base*, y los $\{a_i : 1 \leq i \leq n\}$ son las desconocidas[3]. Un caso común es cuando se toma $\phi_i(x) = x^{i-1}$, $1 \leq i \leq n$. En éste caso se busca aproximar los datos con un polinomio de grado a lo más $n - 1$.

Volviendo al caso general, las diferencias entre los datos y la función $g(x)$ evaluada en los (x_k) están dadas por:

$$\varepsilon_k = y_k - (a_1\phi_1(x_k) + a_2\phi_2(x_k) + \cdots + a_n\phi_n(x_k)), \quad 1 \leq k \leq m. \tag{5.33}$$

Buscamos pues minimizar la raíz cuadrada del promedio de los cuadrados de éstas

[2]Esto es equivalente a minimizar la suma de los cuadrados de las diferencias entre los datos y la función que se usa para aproximar.

[3]Lo usual es que la m sea mucho más grande que la n.

diferencias, llamado el *error cuadrado medio*, dado por:

$$\text{ECM} = \left[\frac{1}{m}\sum_{k=1}^{m}\varepsilon_k^2\right]^{1/2}. \tag{5.34}$$

Este es un problema de optimización en n variables[4], es decir, $a_1, \ldots, a_n$. Se puede demostrar que si la matriz A definida abajo en (5.36) es de rango n donde $m \geq n$, entonces los valores de $\{a_i : 1 \leq i \leq n\}$ que minimizan a ECM son solución del sistema lineal de ecuaciones (*ecuaciones normales*):

$$A^t A a = A^t b, \tag{5.35}$$

donde

$$A = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \phi_3(x_2) & \cdots & \phi_n(x_2) \\ \phi_1(x_3) & \phi_2(x_3) & \phi_3(x_3) & \cdots & \phi_n(x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_m) & \phi_2(x_m) & \phi_3(x_m) & \cdots & \phi_n(x_m) \end{pmatrix}, \tag{5.36}$$

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{pmatrix}. \tag{5.37}$$

La función (5.32) donde los coeficientes (a_i) están dados por la solución a de las ecuaciones normales, se conoce como la *solución de cuadrados mínimos* para los datos $\{(x_k, y_k) : k = 1, 2, \ldots, m\}$ usando las funciones base $\{\phi_i : 1 \leq i \leq n\}$. El error cuadrado medio óptimo se puede calcular mediante:

$$\text{ECM}_{\text{opt}} = \left[\frac{1}{m}\left\|Aa - b\right\|_2^2\right]^{1/2} = \frac{1}{\sqrt{m}}\left\|Aa - b\right\|_2. \tag{5.38}$$

5.7.1 Cuadrados Mínimos Polinomiales

En el caso en que $\phi_i(x) = x^{i-1}$, $1 \leq i \leq n$, el problema de minimizar ECM se conoce como el problema de *cuadrados mínimos polinomial* y la matriz A tiene

[4]Como las variables $a_1, \ldots, a_n$ aparecen en forma lineal en (5.32), el problema de minimizar ECM se llama *un problema de cuadrados mínimos lineal.*

ahora la forma:

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{n-1} \end{pmatrix}. \tag{5.39}$$

Esta matriz tiene una estructura bien similar a la matriz de Vandermonde que vimos en la interpolación de polinomios. Imitando la discusión anterior sobre el cálculo de la matriz de Vandermonde, podemos entonces escribir el siguiente código en MATLAB que calcula la matriz A en (5.39):

```
function a=vandg(n,x);
m=length(x);
a=ones(m,n);
for j=2:n
  a(:,j)=x.*a(:,j-1);
end
```

El siguiente programa en MATLAB llama la función de arriba para luego ensamblar las ecuaciones normales, y resuelve éstas para obtener así la solución de cuadrados mínimos polinomial:

```
function [a,ecm]=leastsqn(n,x,y);
A=vandg(n,x);
m=length(x);
B=A'*A;
a=B\(A'*y);
ecm=norm(A*a-y)/sqrt(m);
```

El vector `a` que devuelve ésta función representa los coeficientes del polinomio de grado a lo más $n-1$ que mejor aproxima a los datos en el sentido de los cuadrados mínimos. Podemos ahora utilizar la función `hornerV` discutida anteriormente para evaluar dicho polinomio.

Ejemplo 5.14. Para ilustrar las ideas presentadas hasta ahora, considere el caso de aproximar 17 datos tomados de la función $y = \exp(x)$ en el intervalo $[0, 4]$. Construimos los polinomios de grado dos y tres que mejor aproximan los datos en el sentido de los cuadrados mínimos. Para esto usamos el siguiente programa en MATLAB (vea la Figura 5.8):

```
x=linspace(0,4,17)';
```

```
y=exp(x);
[a2,e2]=leastsqn(3,x,y);
[a3,e3]=leastsqn(4,x,y);
xx=linspace(0,4,100);';
pval2=hornerV(a2,xx);
pval3=hornerV(a3,xx);
yy=exp(xx);
plot(xx,yy,'k',xx,pval2,'k--',xx,pval3,'k-.',x,y,'ko')
xlabel('x');ylabel('y');
legend('y=e^x','Cuadratica','Cubica','Datos')
disp(['Error cuadrado medio para el polinomio cuadratico: '...
       num2str(e2)])
disp(['Error cuadrado medio para el polinomio cubico: '...
       num2str(e3)])
```

Los errores cuadrados medios óptimos fueron de 2.2853 y 0.56222 respectivamente para el polinomio cuadrático y cúbico. □

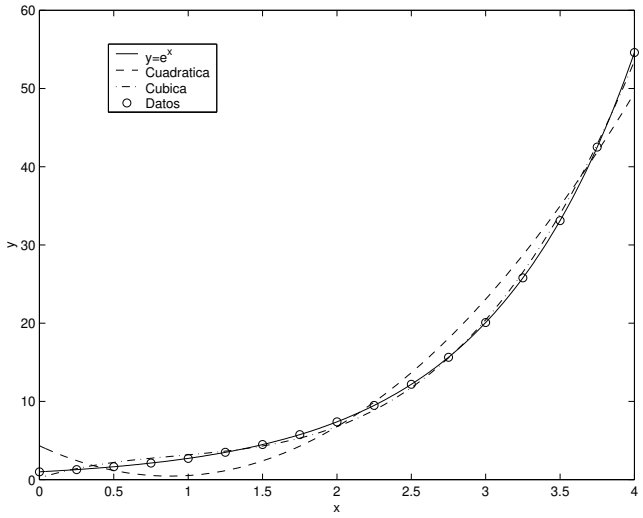


Figura 5.8: Aproximaciones de cuadrados mínimos de grados dos y tres para la función $f(x) = \exp(x)$.

Ejemplo 5.15. Considere los datos $(0, 5), (1, -6), (2, 7)$. Vamos a buscar la mejor recta que aproxima estos datos en el sentido de los cuadrados mínimos. La matriz

(5.39) toma la siguiente forma en este caso:

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix},$$

de modo que

$$A^t A = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}.$$

Las ecuaciones normales están dadas por el sistema:

$$\begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 5 \\ -6 \\ 7 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \end{pmatrix},$$

el cual tiene solución $a_1 = a_2 = 1$. Tenemos pues que la mejor recta que aproxima a los datos en el sentido de cuadrados mínimos es $y = a_1 + a_2 x$, i.e., $y = 1 + x$. En la Figura 5.9 trazamos ahora los datos y la mejor recta en el mismo sistema de coordenadas. □

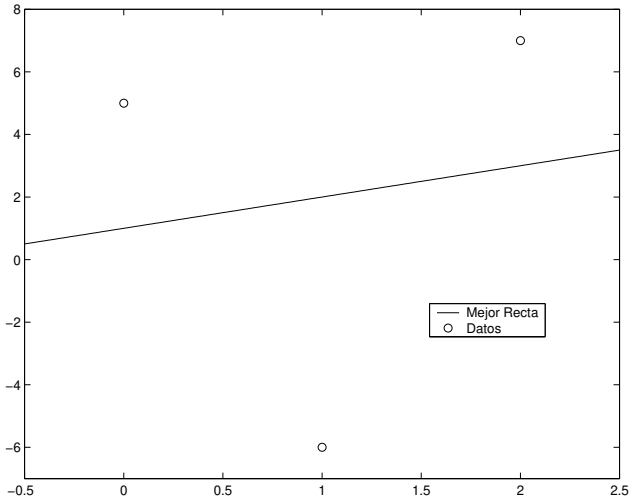


Figura 5.9: La mejor recta que aproxima a los datos (0,5),(1,-6),(2,7) en el sentido de los cuadrados mínimos.

5.7.2 Gráficas de Residuos

Un error cuadrado medio pequeño, no es necesariamente indicativo de que el modelo propuesto para aproximar los datos es bueno. Otra forma, al menos cualitativa, de medir la calidad de la aproximación, es examinando la gráfica de los residuos, es decir $\{(x_k, \varepsilon_k) : 1 \leq k \leq m\}$. Si en esta gráfica los residuos no aparentan seguir ningún patrón en específico, o sea que tienen o indican un comportamiento de tipo "aleatorio" alrededor de cero, entonces decimos que el modelo es adecuado para representar los datos. De lo contrario, si observamos un patrón claro en los residuos, indicativo de un sesgo en el modelo similar a la incertidumbre sistemática en medidas experimentales, entonces decimos que el modelo no representa adecuadamente los datos. Ilustramos estas ideas con un ejemplo.

Ejemplo 5.16. En el siguiente programa en MATLAB consideramos 19 datos que representan el crecimiento de unas células cancerosas como función del tiempo medido en horas. Primero consideramos un modelo lineal de los datos basado en la mejor recta en el sentido de los cuadrados mínimos:

```
time=[10,10,10,25,25,37,37,37,52,52,61,61,61,72,72,72,83,83,83];
cells=[39.58,69.64,59.97,110.32,99.9871,100.33,65.29,129.53,...
    153.62,167.27,234.08,244.47,229.78,282.12,300.47,342.85,...
    494.11,453.57,492.24];
[a,ecm]=leastsqn(2,time',cells');
disp(['Error cuadrado medio para la mejor recta: ' num2str(ecm)])
xx=linspace(min(time)-1,max(time)+1,100);
p1=hornerV(a,xx);
subplot(1,2,1)
plot(time,cells,'ko',xx,p1,'k')
xlabel('tiempo (horas)');ylabel('numero de celulas')
legend('Datos','Mejor Recta')
errors=cells-hornerV(a,time);
subplot(1,2,2)
plot(time,errors,'ko')
```

Los resultados se muestran en la Figura 5.10. Observe que los residuos muestran un patrón bien definido con una concavidad hacia arriba. Esto es indicativo de que el modelo no es adecuado para los datos. Podemos observar también que el error cuadrado medio es de 56.5468, un tanto grande. Cabe observar que el que éste error sea grande y que los residuos muestren algún patrón definido, no necesariamente ocurren simultáneamente.

Para éstos datos es fácil ver que un modelo exponencial de la forma $g(t) = ae^{bt}$ representa mucho mejor los datos. Note que el parámetro b aparece en forma

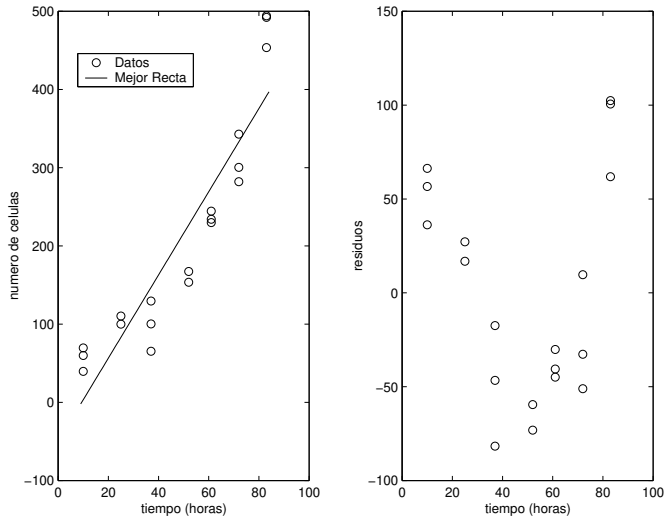


Figura 5.10: Gráficas de los datos, la mejor recta, y los residuos para los datos de células cancerosas.

nolineal en éste modelo por lo que no tiene la forma (5.32). No obstante, como $\ln(g(t)) = \ln a + bt$, podemos usar un modelo lineal para los datos transformados:

$$(t_i, \ln(y_i)), \quad i = 1, 2, \ldots, 19.$$

En el siguiente programa buscamos la mejor recta en el sentido de cuadrados mínimios que aproxima éstos datos transformados. Si la mejor recta resulta ser $\ell(t) = \alpha + \beta t$, entonces el mejor modelo exponencial (en este sentido) será $g(t) = ae^{bt}$ donde $a = e^{\alpha}$ y $b = \beta$. Al leer el programa, recuerde que la función `log` de MATLAB calcula el logaritmo natural:

```
[ae,ecme]=leastsqn(2,time',log(cells)');
disp(['Error cuadrado medio para la mejor exponencial: '...
     num2str(ecme)])
pe=exp(ae(1))*exp(ae(2)*xx);
subplot(1,2,1)
plot(time,cells,'ko',xx,pe,'k')
xlabel('tiempo (horas)');ylabel('numero de celulas')
legend('Datos','Mejor Exponencial')
errors=cells-exp(ae(1))*exp(ae(2)*time);
subplot(1,2,2)
plot(time,errors,'ko')
```

```
xlabel('tiempo (horas)');ylabel('residuos')
```

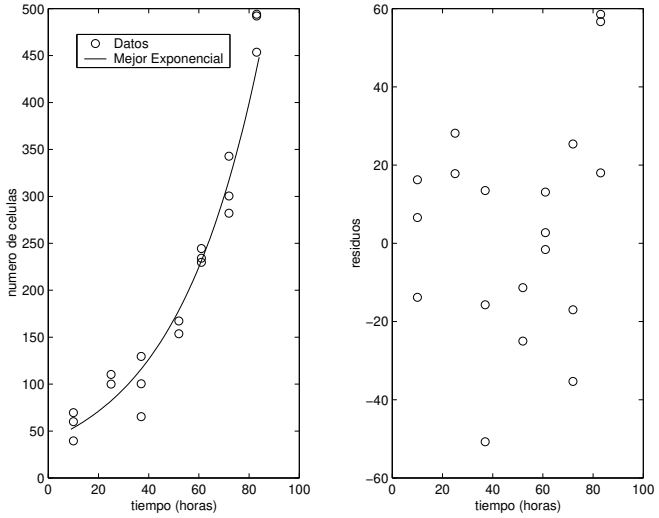


Figura 5.11: Gráficas de los datos, la mejor función exponencial, y los residuos para los datos de células cancerosas.

El error cuadrado medio en el modelo exponencial es de 0.19849, mucho mejor que en el modelo lineal. Note también de la gráfica de residuos en la Figura 5.11, que éstos no muestran ningún patrón específico lo que es indicativo de que el modelo exponencial describe mejor los datos. De hecho la mejor función exponencial calculada por el programa resulto ser $g(t) = 40.0448\mathrm{e}^{0.0288t}$. □

5.7.3 Factorización QR

La matriz de coeficientes A^tA en las ecuaciones normales es en general una matriz mal acondicionada según la m aumenta. De hecho en el caso del problema de cuadrados mínimos polinomial, es fácil ver que la entrada (k, l) de A^tA es de la forma:

$$\sum_{i=1}^{m} x_i^N, \tag{5.40}$$

donde $N = k+l-2$. Suponiendo que $x_1, \ldots, x_m$ representa una partición uniforme del intervalo $[0, 1]$, tenemos que

$$\lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} x_i^N = \int_0^1 x^N dx = \frac{1}{N+1}, \tag{5.41}$$

de modo que

$$\sum_{i=1}^{m} x_i^N \approx \frac{m}{N+1}, \tag{5.42}$$

para m grande. Así que A^tA es (aproximadamente) proporcional a la matriz de Hilbert de orden $n \times n$ para m grande la cual es mal acondicionada. El número de datos no tiene que ser muy grande. De hecho en el Ejemplo 5.14, los números de condición son de 983.3594 y 5.7299e+004 respectivamente para los casos $n = 3, 4$ lo cual indica que se pueden perder hasta 4 ó 5 cifras significativas al resolver las ecuaciones normales. Para resolver éstas ecuaciones en forma eficiente y estable, se utiliza la llamada factorización QR de la matriz A.

Teorema 5.17 (Factorización QR)**.** *Sea A una matriz $m \times n$ de rango n. Entonces existe una matriz Q de tamaño $m \times n$ con $Q^tQ = I$, y una matriz R de tamaño $n \times n$ triangular superior y nosingular tal que $A = QR$.*

La demostración de éste teorema utiliza el proceso de Gram-Schmidt para calcular una base ortonormal de un espacio vectorial. (Ver [13]). No discutiremos aquí los aspectos computacionales de como calcular la factorización QR de una matriz pues utilizaremos las funciones de MATLAB para ésto. Lo que si nos interesa en este punto mencionar, es que utilizando la factorización QR de la matriz A que aparece en las ecuaciones normales, podemos calcular la solución del problema de cuadrados mínimos en forma eficiente y sin problemas de mal acondicionamiento.

Teorema 5.18. *Sea A una matriz $m \times n$ de rango n y $A = QR$ la factorización QR de A dada por el teorema anterior. Entonces la solución a de las ecuaciones normales $A^tAa = A^tb$ se puede obtener resolviendo el sistema triangular $Ra = Q^tb$.*

Demostración: Dado que $A = QR$ tenemos que:

$$A^tAa = (R^tQ^t)(QR)a = R^t(Q^tQ)Ra = R^tIRa = R^tRa$$

De igual forma: $A^tb = (R^tQ^t)b = R^t(Q^tb)$. Como R es nosingular R^t también lo es, y tenemos entonces que $A^tAa = A^tb$ es equivalente al sistema $Ra = Q^tb$. □

Vale la pena recalcar que los sistemas triangulares se resuelven eficientemente mediante sustitución para atrás y son por lo general bien acondicionados. La función `qr` de MATLAB se utiliza para calcular las factorizaciones QR. Ahora podemos modificar la función `leastsqn` de arriba utilizando la factorización QR:

```
function a=leastsqr(n,x,y);
A=vandg(n,x);
[Q R]=qr(A);
a=R\(Q'*y);
```

Para el mismo problema del Ejemplo 5.14 pero usando ésta subrutina en lugar de `leastsqn`, obtenemos resultados similares (al número de cifras mostradas) pero ahora los números de condición de R son 31.3586 y 239.3714 para $n = 3, 4$ respectivamente los cuales son mucho mejor que antes.

5.7.4 Cuadrados mínimos nolineales

El problema de cuadrados mínimos utilzando la función *modelo* (5.32) se llama un problema de cuadrados mínimos lineal porque los parámetros o coeficientes $a_1, \ldots, a_n$ aparecen en forma lineal. Es bien común en diferentes aplicaciones que los parámetros del modelo aparezcan en forman nolineal en el modelo. Este podría ser el caso, por ejemplo, si se utilizan funciones bases periódicas del tipo $\{\operatorname{sen} a_i x\}$ o exponenciales como $\{\mathrm{e}^{a_i x}\}$. En algunos casos es posible transformar los datos para obtener un problema de cuadrados mínimos lineal en unos parámetros transformados. (Vea el Ejemplo 5.16 y el Ejercicio 5.23.) Pero en general, estas transformaciones no son posibles por lo que necesitamos un método general para calcular los valores óptimos de estos parámetros directamentamente.

Ejemplo 5.19. Consideremos el siguiente conjunto de 25 datos:

$$\begin{aligned} x = (x_i) &= [0.00, 0.08, 0.17, 0.25, 0.33, 0.42, 0.50, 0.58, \\ &\quad 0.67, 0.75, 0.83, 0.92, 1.00, 1.08, 1.17, 1.25, \\ &\quad 1.33, 1.42, 1.50, 1.58, 1.67, 1.75, 1.83, 1.92, 2.00], \\ y = (y_i) &= [334.80, 334.80, 334.14, 332.88, 330.64, 329.01, 328.62, 330.13, \\ &\quad 331.63, 332.74, 333.23, 334.91, 336.01, 336.82, 336.13, 334.75, \\ &\quad 332.53, 331.27, 331.21, 332.36, 333.47, 334.74, 335.22, 336.52, 337.77]. \end{aligned}$$

En la Figura 5.12 mostramos una gráfica de estos datos (circulos). Observando el comportamiento de estos datos, es razonable pensar en un modelo de la forma $a_1\mathrm{e}^{a_2 x} + a_3 \operatorname{sen}(a_4 x) + a_5 \cos(a_6 x)$. Poniendo $\vec{a} = (a_1, \ldots, a_6)$, escribimos:

$$f(x, \vec{a}) = a_1\mathrm{e}^{a_2 x} + a_3 \operatorname{sen}(a_4 x) + a_5 \cos(a_6 x). \tag{5.43}$$

Note que los parámetros aparecen en forma nolineal en el modelo. Los parámetros óptimos en el sentido de los cuadrados mínimos se definen como aquellos que minimizan el error cuadrado medio:

$$\sqrt{\frac{1}{25}\sum_{k=1}^{25}(y_k - f(x_k, \vec{a}))^2}.$$

Vamos ahora a discutir un método para aproximar estos parámetros. □

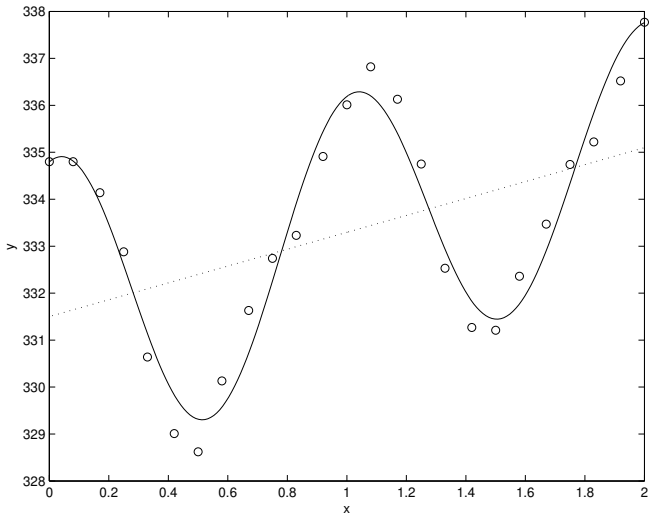


Figura 5.12: Gráficas de los datos (circulos) del Ejemplo 5.19, el modelo exponencial-periódico (5.43) (curva sólida), y la parte exponencial de dicho modelo (curva entrecortada).

Suponga que $f(x, \vec{a})$ representa el modelo propuesto para ajustar a los datos (x_k, y_k), $k = 1, \ldots, m$, donde $\vec{a} = (a_1, \ldots, a_n)$. Recuerde que lo usual es que m sea mucho más grande que n. La solución de cuadrados mínimos para este problema se define como el vector $\vec{a}^*$ que minimiza el error cuadrado medio:

$$\text{ECM} = \sqrt{\frac{1}{m} \sum_{k=1}^{m} (y_k - f(x_k, \vec{a}))^2}. \tag{5.44}$$

Minimizar esta expresión es equivalente a minimizar:

$$g(\vec{a}) = \sum_{k=1}^{m} (y_k - f(x_k, \vec{a}))^2 . \tag{5.45}$$

El método de Newton o su variante para problemas de minimización, se puede utilizar para aproximar la solución de cuadrados mínimos. En este método, partiendo de una cierta aproximación $\vec{a}_0$ de $\vec{a}^*$, se generan las aproximaciones sucesivas $\{\vec{a}_i\}$ de acuerdo a la recursión:

$$\vec{a}_{i+1} = \vec{a}_i - \alpha_i \left[\nabla^2 g(\vec{a}_i)\right]^{-1} \nabla g(\vec{a}_i), \quad i = 0, 1, 2, \ldots, \tag{5.46}$$

donde

$$\nabla g(\vec{a}) = \left(\frac{\partial g}{\partial a_j}(\vec{a})\right), \quad \nabla^2 g(\vec{a}) = \left(\frac{\partial^2 g}{\partial a_l \partial a_j}(\vec{a})\right), \tag{5.47}$$

y los α_i son constantes positivas tal que $\alpha_i \longrightarrow 1$ según $i \longrightarrow \infty$. Note que estas ecuaciones son las mismas que se obtienen al aplicar el método de Newton, según descrito en la Sección 4.7, al sistema $\nabla g(\vec{a}) = \vec{0}$. Para (5.45) es fácil verificar que:

$$\frac{\partial g}{\partial a_j}(\vec{a}) = -2\sum_{k=1}^{m} (y_k - f(x_k, \vec{a}))\, \frac{\partial f}{\partial a_j}(x_k, \vec{a}), \tag{5.48a}$$

$$\begin{aligned} \frac{\partial^2 g}{\partial a_l \partial a_j}(\vec{a}) &= 2\sum_{k=1}^{m} \left[\frac{\partial f}{\partial a_l}(x_k, \vec{a}) \frac{\partial f}{\partial a_j}(x_k, \vec{a}) \right. \\ &\qquad \left. - (y_k - f(x_k, \vec{a}))\, \frac{\partial^2 f}{\partial a_l \partial a_j}(x_k, \vec{a})\right]. \end{aligned} \tag{5.48b}$$

Ejemplo 5.20. Cosidere nuevamente los datos del Ejemplo 5.19. De la ecuación (5.43) tenemos que:

$$\begin{aligned} \nabla_{\vec{a}} f &= \left(\frac{\partial f}{\partial a_1}\frac{\partial f}{\partial a_2}, \frac{\partial f}{\partial a_3}, \frac{\partial f}{\partial a_4}, \frac{\partial f}{\partial a_5}, \frac{\partial f}{\partial a_6}\right)^t, \\ &= \left(\mathrm{e}^{a_2 x}, a_1 x \mathrm{e}^{a_2 x}, \operatorname{sen}(a_4 x), a_3 x \cos(a_4 x), \cos(a_6 x), -a_5 x \operatorname{sen}(a_6 x)\right)^t, \end{aligned}$$

mientras que la segunda derivada:

$$\nabla^2_{\vec{a}} f = \left(\frac{\partial^2 f}{\partial a_i \partial a_j}\right)_{i,j=1,\ldots,6},$$

está dada por

$$\begin{bmatrix} 0 & x\mathrm{e}^{a_2 x} & 0 & 0 & 0 & 0 \\ x\mathrm{e}^{a_2 x} & a_1 x^2 \mathrm{e}^{a_2 x} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x\cos(a_4 x) & 0 & 0 \\ 0 & 0 & x\cos(a_4 x) & -a_3 x^2 \operatorname{sen}(a_4 x) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -x\operatorname{sen}(a_6 x) \\ 0 & 0 & 0 & 0 & -x\operatorname{sen}(a_6 x) & -a_5 x^2 \cos(a_6 x) \end{bmatrix}.$$

Con estas formulas podemos ahora escribir una subrutina en MATLAB para calcular las ecuaciones (5.48):

```
function [G,H]=GH_lsnl(a,x,y)
m=length(x);
```

```
f=a(1)*exp(a(2)*x)+a(3)*sin(a(4)*x)+a(5)*cos(a(6)*x);
Gf=[exp(a(2)*x);a(1)*x.*exp(a(2)*x);sin(a(4)*x);...
    a(3)*x.*cos(a(4)*x);cos(a(6)*x);-a(5)*x.*sin(a(6)*x)];
Hf=zeros(6,6,m);
Hf(1,2,:)=x.*exp(a(2)*x); Hf(2,1,:)=Hf(1,2,:);
Hf(2,2,:)=a(1)*x.^2.*exp(a(2)*x);
Hf(3,4,:)=x.*cos(a(4)*x); Hf(4,3,:)=Hf(3,4,:);
Hf(4,4,:)=-a(3)*x.^2.*sin(a(4)*x);
Hf(5,6,:)=-x.*sin(a(6)*x); Hf(6,5,:)=Hf(5,6,:);
Hf(6,6,:)=-a(5)*x.^2.*cos(a(6)*x);
G=-2*Gf*(y-f)';
H=zeros(6,6);
for k=1:m
  H=H+Gf(:,k)*Gf(:,k)'-(y(k)-f(k))*Hf(:,:,k);
end
H=2*H;
```

Usando esta función junto con las iteraciones (5.46), donde tomamos:

$$\vec{a}_0 = [300, 0.1, 1, 6, 1, 6]^t,$$

obtuvimos un mínimo aproximado para (5.45) en:

$$\vec{a}^* = [331.5020, 0.0054, 0.6484, 5.6196, 3.2934, 6.2748]^t.$$

El modelo (5.43) queda entonces como:

$$f(x) = 331.5020\,\mathrm{e}^{0.0054x} + 0.6484\,\mathrm{sen}\,(5.6196x) + 3.2934\cos(6.2748x). \qquad (5.49)$$

En la Figura 5.12 mostramos la gráfica del modelo resultante junto con los datos. La grafica entrecortada en la figura muestra unicamente el término exponencial $331.5020\,\mathrm{e}^{0.0054x}$ del modelo. El error cuadrado medio según la formula (5.44) para el modelo es de 0.5315. En la Figura 5.13 mostramos la grafica de los residuos o errores $(y_k - f(x_k, \vec{a}^*))$. El error promedio del modelo con respecto a los datos fue de -3.5680×10^{-7}. □

5.8 Ejercicios

Ejercicio 5.1. Calcule la función $q(x) = a + b\cos(\pi x) + c\,\mathrm{sen}(\pi x)$ que interpola a los datos $(0, 2), (1/2, 5), (1, 4)$.

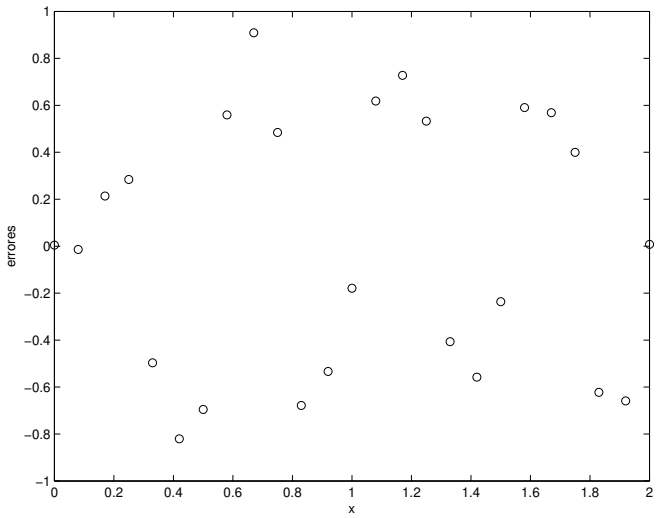


Figura 5.13: Gráfica de residuos para el ajuste o modelo (5.49) para los datos del Ejemplo 5.19.

Ejercicio 5.2. Halle el polinomio cuadrático que interpola a los datos (-2,-15), (-1,-8), (0,-3) por los tres métodos discutidos en el texto.

Ejercicio 5.3. Halle las tres representaciones (canónica, Newton, y Lagrange) del polinomio de interpolación para los datos $(-2,-3)$, $(1,4)$, $(3,6)$. Suponga que se añade el dato $(4,-1)$. Halle la representación canónica y la de Newton del nuevo polinomio de interpolación. ¿Cómo comparan los nuevos coeficientes en la representación canónica con los de dicha representación para los datos iniciales?

Ejercicio 5.4. Use inducción matemática para demostrar que si $V(x_1,\ldots,x_n)$ representa la matriz de Vandermonde para $\{x_1,\ldots,x_n\}$, entonces

$$\det V(x_1,\ldots,x_n) = \prod_{i<j}(x_i - x_j).$$

Usando este resultado concluya que $V(x_1,\ldots,x_n)$ es invertible si y sólo si $x_1,\ldots,x_n$ son todos distintos.

Ejercicio 5.5. Demuestre que si $q_{n-2}(x)$ es el polinomio de grado $n-2$ que interpola a los datos

$$\left(x_i, \frac{y_i - y_1}{x_i - x_1}\right), \quad 2 \le i \le n,$$

y p_{n-1} es el polinomio de grado $n-1$ que interpola a $\{(x_i, y_i) : 1 \leq i \leq n\}$, entonces

$$p_{n-1}(x) = y_1 + (x - x_1)q_{n-2}(x).$$

Ejercicio 5.6. Demuestre mediante inducción matemática que los coeficientes $c_1, \ldots, c_n$ de la representación de Newton de p_{n-1} están dados por el programa secuencial `interpN`.

Ejercicio 5.7. Para los datos $\{(x_k, y_k) : 1 \leq k \leq n\}$, sean $p_{n-1}(x)$, $q_{n-2}(x)$, y $r_{n-2}(x)$ los polinomios de grados $n-1$ y $n-2$ que interpolan a los datos $\{x_1, \ldots, x_n\}$, $\{x_1, \ldots, x_{n-1}\}$, y $\{x_2, \ldots, x_n\}$ respectivamente. Demuestre que

$$p_{n-1}(x) = \frac{(x_n - x)q_{n-2}(x) + (x - x_1)r_{n-2}(x)}{x_n - x_1}.$$

Usando ahora esta identidad y comparando los coeficientes de x^{n-1} en ambos lados de la misma, obtenga la identidad (5.15).

Ejercicio 5.8. Sean $p_{n-1}(x)$ y $p_n(x)$ los polinomios de interpolación a los datos $\{x_1, \ldots, x_n\}$ y $\{x_1, \ldots, x_n, x_{n+1}\}$ respectivamente. Usando la fórmula (5.14) demuestre que

$$p_n(x) = p_{n-1}(x) + f[x_1, \ldots, x_n, x_{n+1}] \prod_{i=1}^{n} (x - x_i).$$

Usando esta identidad y la fórmula (5.20) para el error de interpolación, verifique la propiedad (5.16) de las diferencias divididas de Newton.

Ejercicio 5.9. Las diferencias divididas de Newton tienen la ventaja de que al añadir un dato adicional (x_{n+1}, y_{n+1}) a los datos $\{(x_i, y_i)\}_{i=1}^{n}$, entonces los primeros n coeficientes en la representación de Newton de p_n coinciden con los de p_{n-1}. Escriba un programa en MATLAB que a partir de la representación de Newton de p_{n-1}, éste produzca como resultado el coeficiente adicional de p_n en su representación de Newton. ¿Cuál es el conteo operacional de dicho algoritmo?

Ejercicio 5.10. Verifique las propiedades (5.18) y (5.19) de los polinomios base de Lagrange además de la identidad

$$\sum_{k=1}^{n} \ell_k(x) = 1.$$

Ejercicio 5.11. Considere los datos:

i	1	2	3	4
x_i	-2	0	1	3
$f(x_i)$	2	-3	1	-2

a) Halle los polinomios base de Lagrange ℓ_1, ℓ_2, ℓ_3, ℓ_4.

b) Halle la fórmula de Lagrange del polinomio de interpolación para estos datos.

c) Calcule $f[x_1, x_2]$, $f[x_1, x_2, x_3]$, $f[x_1, x_2, x_3, x_4]$.

d) Halle la representación de Newton del polinomio de interpolación para estos datos.

e) Si $|f^{(4)}(x)| \leq 5$ para $x \in [x_1, x_4]$, halle un estimado del error $|f(x) - p_3(x)|$ para $x \in [x_1, x_4]$.

Ejercicio 5.12. Considere el problema de construir una tabla de valores para la función $f(x) = \ln(x)$ en puntos uniformemente distribuidos en el intervalo $[1, 10]$. Para calcular valores de la función intermedios a los dados en la tabla, se utilizará interpolación lineal. ¿Qué distancia deben tener entre si los puntos de la tabla para que el error de interpolación en los valores intermedios sea de a lo más 10^{-8}?

Ejercicio 5.13. En el Ejercicio 5.12, suponga que los datos $\{f(x_i)\}$ se conocen con un error absoluto no mayor de 10^{-5}. Esto es, suponga que $f(x_i) = y_i + \varepsilon_i$ donde $|\varepsilon_i| \leq 10^{-5}$, $1 \leq i \leq n$. Si la interpolación se hace con los datos aproximados $\{(x_i, y_i)\}$, ¿qué distancia deben tener entre si los puntos de la tabla para que el error de interpolación en los valores intermedios sea de a lo más 10^{-5}?

Ejercicio 5.14. Escriba una función en MATLAB, con secuencia de llamada `nbest(L,R,a,delta)`, que devuelva el entero menor n tal que si p_{n-1} es el polinomio de interpolación de $\exp(ax)$ en los puntos

$$L + (i-1)\frac{R-L}{n-1}, \quad 1 \leq i \leq n,$$

entonces

$$|p_{n-1}(z) - \exp(az)| \leq \text{delta}, \quad \forall z \in [L, R].$$

Ejercicio 5.15. Defina la función $\Phi_n(x) = \prod_{k=1}^{n}(x - x_k)$. Demuestre que para los puntos (5.21)

$$|\Phi_n(x)| \leq (n-1)!h^n, \quad x \in [a, b].$$

Use ésto para obtener la desigualdad (5.22).

Ejercicio 5.16. Determine los parámetros a, b, c, d, e, f, g, h tal que el spline cúbico natural

$$s(x) = \begin{cases} ax^3 + bx^2 + cx + d, & x \in [-1, 0], \\ ex^3 + fx^2 + gx + h, & x \in [0, 1], \end{cases}$$

cumpla con las condiciones de interpolación $s(-1) = 1$, $s(0) = 2$, $s(1) = -1$.

Ejercicio 5.17. Halle las ecuaciones que satisfacen los parámetros a, b, c, d del polinomio cúbico $p(x) = ax^3+bx^2+cx+d$ que cumple con $p(-1) = 1$, $p'(-1) = -1$, $p(1) = 2$, $p'(1) = 3$. Escriba las instrucciones de MATLAB necesarias para resolver dicho sistema.

Ejercicio 5.18. Considere el problema de construir un spline *cuadrático* para los datos $(-1, -2)$, $(0, 3)$, $(1, 1)$. Esto es buscamos una función $s(x) \in C^1[-1, 1]$ de la forma:

$$s(x) = \begin{cases} ax^2 + bx + c, & x \in [-1, 0], \\ dx^2 + ex + f, & x \in [0, 1]. \end{cases}$$

Usando las condiciones de interpolación, el dato que $s(x), s'(x)$ son continuas en $[-1, 1]$, y la condición adicional de que $s'(1) = 0$, determine los valores de a, b, c, d, e, f.

Ejercicio 5.19. La función

$$N(\nu) = \nu^5 - \frac{1}{\nu^3}, \quad \nu > 0,$$

es estrictamente creciente por lo que tiene una función inversa. En este ejercicio utilizamos splines para construir una aproximación de la función inversa.

a) Comience generando 20 valores de ν en el intervalo $[0.5, 2]$, y los denota por ν_i, $1 \leq i \leq 20$.

b) Evalúe la función $N(\nu)$ en los ν_i's del paso anterior para generar los datos $(\nu_i, N_i) = (\nu_i, N(\nu_i))$, $1 \leq i \leq 20$.

c) Construya un spline para los datos (N_i, ν_i), $1 \leq i \leq 20$.

d) Dibuje en un mismo sistema de coordenadas, las funciones $N(\nu)$ y el spline del paso anterior. **Nota:** $N(\nu)$ la puede trazar con 100 puntos en el intervalo $[0.5, 2]$ y el spline lo puede trazar con 100 puntos en el intervalo $[\min\{N_i\}, \max\{N_i\}]$.

Ejercicio 5.20. La función de interpolación de Hermite $Q_n(x)$ para los datos $\{x_1, \ldots, x_n\}$ satisface las condiciones

a) $Q_n(x)$ es polinomio cúbico en $[x_i, x_{i+1}]$ para $1 \leq i \leq n - 1$.

b) $Q_n(x_i) = f(x_i)$, $Q_n'(x_i) = f'(x_i)$ para $1 \leq i \leq n$.

Demuestre que para $x \in [x_i, x_{i+1}]$, $1 \leq i \leq n-1$, tenemos que

$$\begin{aligned} Q_n(x) &= \frac{1}{h^2}\left[1 + \frac{2}{h}(x - x_i)\right](x_{i+1} - x)^2 f(x_i) \\ &\quad + \frac{1}{h^2}\left[1 + \frac{2}{h}(x_{i+1} - x)\right](x - x_i)^2 f(x_{i+1}) \\ &\quad + \frac{(x - x_i)(x_{i+1} - x)^2}{h^2} f'(x_i) - \frac{(x - x_i)^2(x_{i+1} - x)}{h^2} f'(x_{i+1}). \end{aligned}$$

Ejercicio 5.21. Halle la recta que mejor aproxima a los datos $(-2, 1)$, $(0, 3)$, $(3, 0)$ en el sentido de los cuadrados mínimos.

Ejercicio 5.22. Halle el polinomio cuadrático que mejor aproxima a los datos $(-2, 2), (-1, 1), (0, 1), (1, 1), (2, 2)$ en el sentido de los cuadrados mínimos. Trace los datos y la cuadrática resultante en el mismo sistema de coordenadas.

Ejercicio 5.23. Usando una función de la forma

$$y = axe^{-bx},$$

donde a, b son constantes a ser determinadas, halle las ecuaciones normales para la mejor aproximación de cuadrados mínimos de ésta forma a los datos $(1, 1.2)$, $(1.5, 1.4)$, $(1.75, 1.45)$, $(2, 1.5)$, $(3, 1.3)$. **Nota:** Debe primero hacer una transformación apropiada de los datos.

Ejercicio 5.24. Considere los datos dados por los vectores

$$\begin{aligned} \text{x} &= [0{:}0.25{:}3], \\ \text{y} &= [6.3806,\ 7.1338,\ 9.1662,\ 11.5545,\ 15.6414,\ 22.7371,\ 32.0696, \\ &\quad\ 47.0756,\ 73.1596,\ 111.4684,\ 175.9895,\ 278.5550,\ 446.4441]. \end{aligned}$$

Aproxime estos datos con funciones de la forma:

a) $g(x) = a + b\exp(x) + c\exp(2x)$.

b) $g(x) = a + b/(1 + x) + c/(1 + x)^2$.

c) $g(x) = a + bx + cx^2 + dx^3$.

Modifique los programas de cuadrados mínimos dados anteriormente según sea necesario. Trace las tres funciones $g(x)$ y los datos originales en un mismo sistema de coordenadas. ¿Qué tan bien aproximan estas funciones a los datos?

Ejercicio 5.25. La función $\Gamma(x+1)$ se puede aproximar con un polinomio de grado cinco de la forma $a_1+a_2x+a_3x^2+a_4x^3+a_5x^4+a_6x^5$. Use la función `gamma` de MATLAB para generar valores de $\Gamma(x+1)$ para $x = 0 : 0.1 : 1$. Usando los programas desarrollados en este capítulo, construya el polinomio de grado cinco que mejor aproxima estos datos en el sentido de los cuadrados mínimos. Trace los datos, la función $\Gamma(x+1)$, y el polinomio calculado en el mismo sistema de coordenadas.

Ejercicio 5.26. Según los datos del Consejo de Calidad Ambiental, la cantidad de desechos (en millones de toneladas por año) generada en Estados Unidos de 1960 a 1990 fue:

Año	1960	1965	1970	1975	1980	1985	1990
Cantidad	81	100	120	124	140	152	164

a) Determine la recta de cuadrados mínimos que mejor aproxima estos datos con t medido en unidades de 5 años y $t = 0$ representando el año 1960.

b) Utilice el resultado de la parte (a) para estimar la cantidad de desechos generados en el año 2000, suponiendo que continúa la tendencia.

Ejercicio 5.27. Los siguientes datos describen los embarques de computadoras personales (en millones) a nivel mundial de 1992 a 1996:

Año	1992	1993	1994	1995	1996
Número	31	39	48	52	61

a) Determine la recta de cuadrados mínimos que mejor aproxima estos datos con el tiempo $t = 0$ representando el año 1992.

b) Utilice el resultado de la parte (a) para estimar los embarques de computadoras personales a nivel mundial en 1997, suponiendo que la tendencia continúa.

Ejercicio 5.28. Use el Método de Newton para aproximar los parámetros (α, β, γ) tal que la función $f(x) = \alpha \exp(\beta x)+\gamma x$ interpola a los datos $(1, 10), (2, 12), (3, 18)$, i.e.,

$$f(1) = 10, \quad f(2) = 12, \quad f(3) = 18.$$

Dibuje en un mismo sistema de coordenadas los datos y la función f obtenida. **Ayuda:** Utilice $\alpha = 15$, $\beta = 1$, $\gamma = -10$ como punto inicial en el método de Newton.

Capítulo 6

Diferenciación e Integración Numérica

Los procesos de diferenciación e integración forman parte esencial de una gran variedad de métodos numéricos. Como bien es conocido en el caso de integración, la mayoría de las funciones encontradas en la practica, no tienen antiderivadas explícitas en términos de funciones elementales y por consiguiente los integrales definidos de dichas funciones hay que aproximarlos numéricamente. Por otra parte, aunque podemos en teoría calcular la derivada en forma análitica de cualquier función conocida, hay situaciones que demandan aproximar éstas ya sea por lo complicado de la fórmula que define la función, o por que dicha función es dada únicamente en un número finito de puntos. Contrario a la situación analítica donde el proceso de diferenciación es más fácil o mecánico que el de integración, la integración numérica es más robusta que la diferenciación numérica ya que en la integración numérica ocurre un fenómeno de cancelación de errores en el promedio el cual favorece dichas aproximaciones, mientras que las formulas para diferenciación numérica son susceptibles a la perdida de cifras significativas.

6.1 Diferenciación Numérica

El cálculo de la derivada de una función puede ser un proceso *difícil* ya sea por lo complicado de la definición explícita de dicha función o por que ésta se conoce únicamente en un número discreto de puntos. (Este es el caso si la función representa el resultado de algún experimento). En esta sección estudiaremos técnicas para aproximar las derivadas de una función y veremos el análisis de error de dichas fórmulas.

6.1.1 Fórmulas para la primera derivada

La definición de la derivada de la función f en el punto x está dada en términos del siguiente límite:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{6.1}$$

De esta definición podemos decir que si h es pequeño entonces:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{6.2}$$

(Note el símbolo de aproximación.) Esto sugiere inmediatamente nuestra primera fórmula numérica para aproximar la derivada:

$$D_h f(x) = \frac{f(x+h) - f(x)}{h}. \tag{6.3}$$

Antes de ver algunos ejemplos donde usaremos esta fórmula, tratemos de contestar la pregunta de, ¿cuán buena es esta aproximación de la derivada? Por el Teorema de Taylor sabemos que:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi_x), \tag{6.4}$$

donde ξ_x esta entre x y $x+h$. Si despejamos en esta fórmula por $f'(x)$ y usamos la definición de $D_h f(x)$, tenemos que:

$$f'(x) = D_h f(x) - \frac{h}{2} f''(\xi_x). \tag{6.5}$$

Esta fórmula nos indica que $D_h f(x)$ aproxima a $f'(x)$ con un error proporcional a h, i.e., $O(h)$.

Ejemplo 6.1. Tomamos $f(x) = x^x$ y queremos aproximar $f'(2)$ cuyo valor exacto es $4(1+\ln(2)) = 6.77258\ldots$. En la Figura 6.1, generada con el siguiente programa en MATLAB, ilustramos los errores $|f'(2) - D_h f(2)|$ como función de h en escala logarítmica:

```
h=zeros(1,17);
df=zeros(1,17);
dx=0.5;
a=2;
fa=a^a;
for i=1:20
    h(i)=dx;
```

```
    df(i)=((a+h(i))^(a+h(i))-fa)/h(i);
    dx=dx/10;
end
loglog(h,abs(df-a^a*(1+log(a)))),'kx')
xlabel('h');ylabel('Error (loglog)');
```

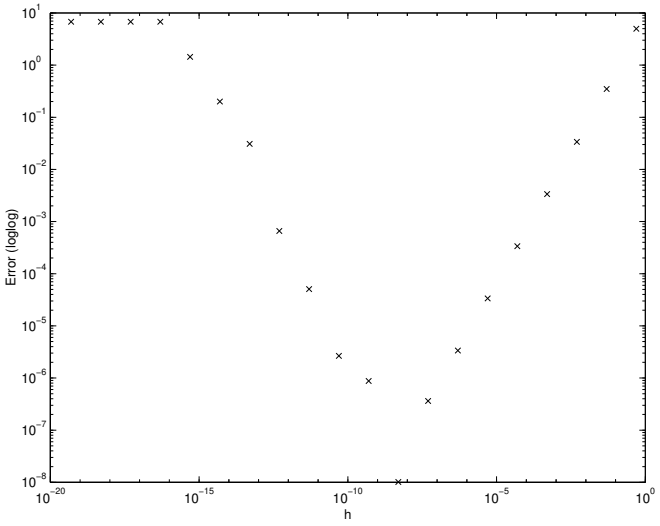


Figura 6.1: Aproximación de $f'(2)$ con $D_h f(2)$ para $f(x) = x^x$ y distintos valores de h.

Podemos ver que los errores disminuyen hasta un cierto valor critico h_{min} luego del cúal éstos aumentan según la h disminuye. ¿Contradice esto el resultado de arriba de $O(h)$ para el error? Pues no, ya que el resultado de arriba es sobre la convergencia si la aritmética es exacta, y por consiguiente se dice que es un resultado *asintótico*. La figura ilustra los efectos de redondeo debido a la aritmética finita (perdida de cifras significativas) los cuales se hacen más marcados para h pequeño, y pueden afectar cualquier fórmula numérica para aproximar la derivada. Por otro lado, una fórmula con un grado de aproximación más alto, digamos $O(h^2)$, es preferible a una $O(h)$ ya que los errores (teóricos) tienden a cero más rápido, obteniendo así aproximaciones bastante precisas sin tener que reducir la h demasiado, disminuyendo de ésta forma los efectos de los errores por la aritmética finita. □

El método de arriba usando la expansión de Taylor se puede utilizar para obtener fórmulas para aproximar la derivada con un grado de aproximación más

alto que uno. Vamos a ilustrar ésto construyendo una fórmula con un error $O(h^2)$. Si en lugar de llegar hasta términos de orden dos, expandimos hasta términos de orden tres en la expansión de Taylor, obtenemos las fórmulas:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_x), \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\eta_x). \end{aligned}$$

Si restamos éstas dos ecuaciones, despejamos para $f'(x)$, y usamos el teorema del valor intermedio (Teorema A.3) pero aplicado a $f'''(x)$, obtenemos la fórmula:

$$f'(x) = D_h^m f(x) + \frac{1}{6}h^2 f'''(\gamma_x), \tag{6.6}$$

donde

$$D_h^m f(x) = \frac{f(x+h) - f(x-h)}{2h}, \tag{6.7}$$

y γ_x está en $[x-h, x+h]$. Tenemos pues que la fórmula (6.7) para aproximar $f'(x)$ tiene un error proporcional a h^2, es decir $O(h^2)$.

Ejemplo 6.2. Comparamos las dos fórmulas obtenidas hasta ahora para aproximar $f'(x)$, el contexto del ejemplo en que $f(x) = x^x$ y para $f'(2)$. Los resultados los presentamos en forma tabulada para distintos valores de h:

h	$D_h f(2)$	$\lvert f'(2) - D_h f(2)\rvert$	$D_h^m f(2)$	$\lvert f'(2) - D_h^m f(2)\rvert$
0.1	7.4964	0.72379	6.8203	0.04775
0.05	7.1215	0.34892	6.7845	0.011914
0.025	6.9439	0.17136	6.7756	0.002977
0.0125	6.8575	0.084918	6.7733	0.00074415

Este ejemplo ilustra cúan efectiva es la fórmula $D_h^m f(x)$ comparada con $D_h f(x)$. Note que cada vez que h se divide entre dos, el error en $D_h f(x)$ se divide entre dos (aproximadamente) mientras que en la fórmula $D_h^m f(x)$ se divide (aproximadamente) por cuatro. ¿Por qué? □

En forma similar se pueden obtener fórmulas de orden mayor utilizando expansiones de Taylor que envuelvan $x \pm 2h$, $x \pm 3h$, etc.. Por ejemplo la expansión

$$f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4), \tag{6.8}$$

sugiere una fórmula de orden cuatro para $f'(x)$. Es importante observar que mientras más alto el grado de aproximación de la fórmula, más suave tiene que ser la función para que dicha aproximación sea válida. Por ejemplo ésta fórmula de orden cuatro requiere que la función tenga cinco derivadas continuas en el intervalo en cuestión, mientras que la fórmula de orden dos requiere únicamente tres derivadas continuas.

6.1.2 Fórmulas para la segunda derivada

El proceso descrito antes se puede usar para obtener fórmulas para las derivadas de orden mayor de uno de una función $f(x)$. En esta sección ilustramos este proceso construyendo una fórmula para la segunda derivada. Usando el Teorema de Taylor, podemos escribir las expansiones:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\xi_x), \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\eta_x). \end{aligned}$$

Sumando éstas dos expansiones y despejando para $f''(x)$ obtenemos que:

$$f''(x) = D_h^{(2)} f(x) - \frac{h^2}{12} f^{(4)}(\gamma_x), \tag{6.9}$$

donde

$$D_h^{(2)} f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \tag{6.10}$$

y γ_x esta entre $[x-h, x+h]$. Tenemos pues una fórmula de orden dos para $f''(x)$.

Ejemplo 6.3. Examinamos la fórmula de arriba en el caso $f(x) = x^x$ y para aproximar $f''(2) = 4(1+\ln(2))^2 + 2 = 13.4669\ldots$. Tenemos los resultados:

h	$D_h^{(2)} f(2)$	$\lvert f''(2) - D_h^{(2)} f(2)\rvert$
0.1	13.5208	0.053854
0.05	13.4804	0.013444
0.025	13.4703	0.0033598
0.0125	13.4678	0.00083989

Nuevamente se puede ver el factor de cuatro en el error al dividir la h entre dos, característico de la convergencia de orden dos. □

En forma similar se pueden obtener fórmulas de orden mayor utilizando expansiones de Taylor que envuelvan $x \pm 2h$, $x \pm 3h$, etc.. Por ejemplo la expansión

$$\begin{aligned} f''(x) &= \frac{1}{12h^2}\left[-f(x-2h) + 16f(x-h) - 30f(x)\right. \\ &\qquad \left. +16f(x+h) - f(x+2h)\right] + O(h^4), \end{aligned} \tag{6.11}$$

sugiere una fórmula de orden cuatro para $f''(x)$.

6.1.3 Diferenciación usando polinomios de interpolación

Suponga que $\{x_1, \ldots, x_n\}$ son puntos distintos y sea $p_{n-1}(x)$ el polinomio que interpola a $f(x)$ en éstos puntos. Entonces aproximamos $f^{(k)}(x)$ por:

$$f^{(k)}(x) \approx p_{n-1}^{(k)}(x). \tag{6.12}$$

Es fácil ver que las dos fórmulas que obtuvimos para aproximar $f'(x)$ se pueden obtener usando polinomios de interpolación de grados uno y dos respectivamente con $k = 1$ en (6.12).

Ejemplo 6.4. Para los datos $f(-2) = 3, f(-0.5) = -1, f(1) = 1, f(3) = 5$, el polinomio cúbico que interpola éstos datos está dado por:

$$p_3(x) = -\frac{27}{35} + \frac{106}{105}x + \frac{104}{105}x^2 - \frac{8}{35}x^3.$$

Si necesitamos aproximar $f'(0), f''(0)$ lo podemos hacer con:

$$f'(2) \approx p_3'(2) \approx 2.2286, \quad f''(2) \approx p_3''(2) \approx -0.76190.$$

□

Si suponemos que $h = |x_{i+1} - x_i|$, $1 \leq i \leq n-1$, entonces se puede demostrar ([11]) que:

$$f^{(k)}(x) - p_{n-1}^{(k)}(x) = O(h^{n-k}), \quad x \in [x_1, x_n]. \tag{6.13}$$

Veamos la demostración de ésta fórmula para el caso $n = 3$ y$k = 1$.

Teorema 6.5. *Sea $f \in C^3[a, b]$ y $p_2(x)$ el polinomio de grado a lo más dos que interpola a f en los datos $x_1, x_2, x_3 \in [a, b]$. Entonces existen $\xi_i \in [x_i, x_{i+1}]$, $i = 1, 2$ tal que si $x \neq \xi_i$, $i = 1, 2$,*

$$f'(x) = p_2'(x) + \frac{1}{2}f'''(\eta_x)(x - \xi_1)(x - \xi_2), \tag{6.14}$$

donde η_x está entre x, ξ_1, ξ_2. En particular, si $h = x_2 - x_1 = x_3 - x_2$, y $x \in [x_1, x_3]$, entonces $f'(x) - p_2'(x) = O(h^2)$.

Demostración: Defina $R_2(x) = f(x) - p_2(x)$. Las condiciones de interpolación implican que $R_2(x_i) = 0$, $i = 1, 2, 3$. Por el Teorema de Rolle ([24]), existen $\xi_i \in [x_i, x_{i+1}]$, $i = 1, 2$ tal que $R_2'(\xi_i) = 0$, $i = 1, 2$. Definimos ahora la función

$$F(z) = R_2'(z) + \alpha(z - \xi_1)(z - \xi_2),$$

donde α es una constante que se selecciona de modo que $F(x) = 0$. De aquí que $F(z)$ tiene tres ceros: x, ξ_1, ξ_2. Si aplicamos el Teorema de Rolle dos veces, tenemos que existe un η_x entre x, ξ_1, ξ_2 tal que $F''(\eta_x) = 0$, de donde obtenemos que

$$\alpha = \frac{1}{2} f'''(\eta_x),$$

ya que $F''(z) = f'''(z) - 2\alpha$. Tenemos ahora que

$$\begin{aligned} 0 = F(x) &= R_2'(x) + \alpha(x - \xi_1)(x - \xi_2), \\ &= R_2'(x) + \frac{1}{2} f'''(\eta_x)(x - \xi_1)(x - \xi_2), \end{aligned}$$

de donde obtenemos la identidad (6.14).

En el caso especial en que $h = x_2 - x_1 = x_3 - x_2$, y $x \in [x_1, x_3]$, tenemos que $|(x - \xi_1)(x - \xi_2)| \leq 2h^2$. Usando ésto y que $f \in C^3[a, b]$, podemos estimar el término del error en (6.14) obteniendo así que $f'(x) - p_2'(x) = O(h^2)$. □

Al igual que en el caso de interpolación directa, la aproximación en (6.12) es propensa a oscilaciones cuando el número de puntos de interpolación es grande. Una alternativa para resolver este problema sería el usar un spline en lugar de un polinomio de interpolación, y entonces aproximamos las derivadas de f con las del spline. En el caso de un spline cúbico, se puede utilizar éste para aproximar las primeras dos derivadas de f. (Vea el Ejercicio 6.7.)

6.1.4 Transformadas de Fourier Discreta y Diferenciación Espectral

Sea $f : \mathbb{R} \longrightarrow \mathbb{R}$ una función periódica con periodo 2π. Para cualquier entero positivo N, defina:

$$t_n = 2\pi \frac{n}{N}, \quad 0 \leq n \leq N - 1. \tag{6.15}$$

Suponga que tenemos los valores $y_n = f(t_n)$, $0 \leq n \leq N - 1$. La *transformada de Fourier discreta* para los datos $\{y_n\}$ se define por:

$$X(k) = \sum_{n=0}^{N-1} y_n \mathrm{e}^{-ikt_n}, \quad -\frac{N}{2} \leq k \leq \frac{N}{2} - 1, \tag{6.16}$$

donde $i^2 = -1$. Tenemos ahora el siguiente resultado el cual se conoce como la *transformada de Fourier inversa discreta*:

Lema 6.6.

$$y_n = \frac{1}{N}\sum_{k=-N/2}^{N/2-1} X(k)\mathrm{e}^{ikt_n}, \quad 0 \le n \le N-1. \tag{6.17}$$

Demostración: Usando la definición (6.16), tenemos que:

$$\begin{aligned}\frac{1}{N}\sum_{k=-N/2}^{N/2-1} X(k)\mathrm{e}^{ikt_n} &= \frac{1}{N}\sum_{k=-N/2}^{N/2-1}\sum_{j=0}^{N-1} y_j \mathrm{e}^{-ikt_j}\mathrm{e}^{ikt_n}, \\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j \left(\sum_{k=-N/2}^{N/2-1} \mathrm{e}^{-ikt_j}\mathrm{e}^{ikt_n}\right).\end{aligned}$$

Claramente $\sum_{k=-N/2}^{N/2-1} \mathrm{e}^{-ikt_j}\mathrm{e}^{ikt_n} = N$ si $n = j$. Por otro lado, si $n \neq j$, y usando que $\mathrm{e}^{-ik(t_n-t_j)} = \mathrm{e}^{(N-k)(t_n-t_j)}$ para $1 \le k \le N/2$, tenemos que

$$\begin{aligned}\sum_{k=-N/2}^{N/2-1} \mathrm{e}^{-ikt_j}\mathrm{e}^{ikt_n} &= \sum_{k=-N/2}^{N/2-1} \mathrm{e}^{ik(t_n-t_j)} = \sum_{k=0}^{N-1} \mathrm{e}^{ik(t_n-t_j)}, \\ &= \sum_{k=0}^{N-1} \left[\mathrm{e}^{i(t_n-t_j)}\right]^k = \frac{1-\left[\mathrm{e}^{i(t_n-t_j)}\right]^N}{1-\mathrm{e}^{i(t_n-t_j)}}, \\ &= \frac{1-\mathrm{e}^{2\pi i(n-j)}}{1-\mathrm{e}^{i(t_n-t_j)}} = 0,\end{aligned}$$

donde usamos que $\mathrm{e}^{i(t_n-t_j)} \neq 1$. Así que

$$\frac{1}{N}\sum_{k=-N/2}^{N/2-1} X(k)\mathrm{e}^{ikt_n} = \frac{1}{N}\sum_{j=0}^{N-1} y_j N\delta_{nj} = y_n.$$

□

Si los datos consisten de números reales, entonces es suficiente calcular $X(k)$ para k no–negativo. El caso especial de $k = N/2$ en el próximo lema es una instancia del fenómeno conocido como *aliasing*.

Lema 6.7. *Suponga que los datos $\{y_n\}$ consisten de números reales. Entonces la transformada de Fourier discreta $\{X(k)\}$ satisface que:*

$$X(k) = \overline{X(-k)}, \quad 1 \le k \le \frac{N}{2}.$$

Demostración: La demostración es una aplicación directa de la definición de $X(k)$ y de las propiedades de la operación del conjugado para números complejos. Veamos:

$$X(k) = \sum_{n=0}^{N-1} y_n \mathrm{e}^{-ikt_n} = \overline{\sum_{n=0}^{N-1} y_n \mathrm{e}^{ikt_n}} = \overline{X(-k)}.$$

□

MATLAB tiene dos funciones intrínsecas para calcular (6.16) y (6.17), éstas son `fft` y `ifft` respectivamente. Ambas rutinas usan una versión del algoritmo para la transformada de Fourier rápida. Los valores de $\left\{\hat{X}(k)\right\}$ calculados por `fft` están dados por:

$$\hat{X}(k) = \sum_{n=0}^{N-1} y_n \mathrm{e}^{-ikt_n}, \quad 0 \leq k \leq N-1. \tag{6.18}$$

Como

$$\mathrm{e}^{-ikt_n} = \mathrm{e}^{i(N-k)t_n}, \quad 1 \leq k \leq N/2,$$

la correspondencia entre los valores de $X(\cdot)$ y $\hat{X}(\cdot)$ está dada por:

$$\hat{X}(k) = \begin{cases} X(k), & 0 \leq k \leq N/2-1, \\ \\ X(k-N), & N/2 \leq k \leq N-1. \end{cases} \tag{6.19}$$

Esto es:

$$\hat{X} = [X(0), X(1), \ldots, X(N/2-1), X(-N/2), X(-N/2+1), \ldots, X(-1)]\,. \tag{6.20}$$

Consideramos ahora el problema de aproximar f' utilizando los $\{X(k)\}$. Primero examinamos la función $h : \mathbb{R} \longrightarrow \mathbb{R}$ dada por:

$$h(t) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} X(k) \mathrm{e}^{ikt}. \tag{6.21}$$

Usando el Lema (6.6), tenemos que h interpola a f en los puntos $\{t_n\}$. Además, si los $\{y_n\}$ son reales, el Lema (6.7) implica que:

$$h(t) = \frac{X(0)}{N} + \frac{2}{N} \sum_{k=1}^{N/2-1} (a_k \cos kt - b_k \sin kt) + \frac{X(-N/2)}{N}\, \mathrm{e}^{-iNt/2},$$

donde $X(k) = a_k + ib_k$ para toda k. Observe que $h'(t_n)$ es un número complejo cuando debería ser real. Para evitar este problema, en lugar de (6.21) se utiliza la función:

$$\begin{aligned} g(t) &= \frac{1}{N}\sum_{k=-N/2+1}^{N/2-1} X(k)\mathrm{e}^{ikt} \\ &\quad +\frac{1}{2N}\left(X(-N/2)\mathrm{e}^{-i\frac{N}{2}t} + X(N/2)\mathrm{e}^{i\frac{N}{2}t}\right), \end{aligned} \tag{6.22}$$

la cual todavía sigue interpolando a f en los $\{t_n\}$. Usando el Lema (6.7) tenemos que

$$g(t) = \frac{X(0)}{N} + \frac{2}{N}\sum_{k=1}^{N/2-1}(a_k\cos kt - b_k\sin kt) + \frac{X(-N/2)}{N}\cos(Nt/2),$$

cuyas derivadas son reales. Utilizamos ahora a g' para aproximar f' en los puntos de interpolación. De la formula (6.22) tenemos que

$$\begin{aligned} g'(t) &= \frac{1}{N}\sum_{k=-N/2+1}^{N/2-1} ikX(k)\mathrm{e}^{ikt} \\ &\quad +\frac{1}{4}\left(-iX(-N/2)\mathrm{e}^{-i\frac{N}{2}t} + iX(N/2)\mathrm{e}^{i\frac{N}{2}t}\right). \end{aligned} \tag{6.23}$$

Si $\{X'(k) : -N/2 \le k \le N/2-1\}$ representa la transformada de Fourier discreta para los datos $\{y'_n = g'(t_n)\}$, entonces usando la expresión anterior tenemos que:

$$X'(k) = ikX(k), \quad -N/2+1 \le k \le N/2-1, \quad X'(-N/2) = 0, \tag{6.24}$$

donde usamos el Lema (6.7) y que $\mathrm{e}^{-i\frac{N}{2}t_n} = \mathrm{e}^{i\frac{N}{2}t_n}$ para obtener que $X'(-N/2) = 0$. Este resultado nos da un método eficiente para el calculo de $\{y'_n\}$ en lugar de usar la expresión (6.23), ésto es, calculamos la transformada de Fourier inversa para los datos $\{X'(k)\}$ dados por la fórmula (6.24). Los datos $\{y'_n\}$ representan una aproximación a las derivadas exactas $\{f'(t_n)\}$. Este método se conoce como *colocación espectral* o *diferenciación espectral*.

Las fórmulas (6.24) deben ser modificadas antes de poder usarlas en MATLAB, para lidiar con el recorrido diferente en los valores de k. Si $\hat{X}'(\cdot)$ representa el vector $X'(\cdot)$ en MATLAB, entonces usando el Lema (6.7), obtenemos que las ecuaciones modificadas son:

$$\hat{X}'(k) = \begin{cases} ikX(k), & 0 \le k \le N/2-1, \\ 0, & k = N/2, \\ \overline{\hat{X}'(N-k)}, & N/2+1 \le k \le N-1. \end{cases} \tag{6.25}$$

Podemos escribir ahora un programa en MATLAB que a partir de los datos $\{y_n\}$, calcula las derivadas aproximadas:

```
function XP=dfft(x)
i=sqrt(-1);
X=fft(x);
N=length(X);
K=[0:N/2-1];
XPfft=zeros(1,N);
XPfft(1:N/2)=i*K.*X(1:N/2);
XPfft(N/2+1)=0;
XPfft(N:-1:N/2+2)=conj(XPfft(2:N/2));
XP=ifft(XPfft);
```

Veamos uno par de ejemplos donde usamos éste procedimiento, para aproximar las derivadas de una función a partir de los valores de la función un número finito de puntos.

Ejemplo 6.8. Considere la función

$$f(x) = \frac{3}{5 - 4\cos(x)}, \quad x \in \mathbb{R}, \tag{6.26}$$

la cual es periódica con periodo 2π. La derivada exacta de ésta función está dada por

$$f'(x) = -\frac{12\sin(x)}{(5 - 4\cos(x))^2}.$$

En el siguiente programa, calculamos las aproximaciones a la derivada en $N = 32$ puntos del intervalo $[0, 2\pi]$, a partir de los valores de la función en éstos puntos. Luego hacemos una gráfica de las aproximaciones y las derivadas exactas. Veamos:

```
per=2*pi;
h=per/32;
x=[0:h:per-h];
y=3./(5-4*cos(x));
yp=real(dfft(y));
z=[0:.01:per-h];
ypexact=-12*sin(z)./(5-4*cos(z)).^2;
plot(x,yp,'kx',z,ypexact,'k')
legend('Derivadas Aproximadas','Derivadas Exactas')
```

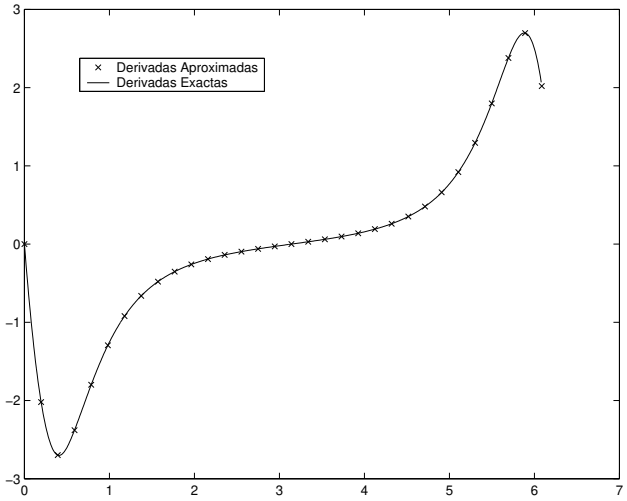


Figura 6.2: Derivadas aproximadas y exactas para la función (6.26).

Los resultados se muestran en la Figura 6.2. Observe el alto grado de precisión en las aproximaciones a las derivadas en éste caso. Esto se debe a la convergencia de orden exponencial de las aproximaciones de Fourier, ya que la función y todas sus derivadas son continuas y periódicas con periodo 2π. □

Ejemplo 6.9. Considere ahora la función

$$f(x) = \sin(x/2), \quad 0 \leq x \leq 2\pi. \tag{6.27}$$

Esta función cumple que $f(0) = f(2\pi)$ y por lo tanto puede ser extendida a una función continua y periódica en $\mathbb{R}$. Pero $f'(0) \neq f'(2\pi)$ de modo que f' no puede ser extendida a una función continua y periódica sobre $\mathbb{R}$. Así que debido al llamado fenómeno de Gibbs, esperamos una convergencia lenta de las derivadas aproximadas. Nuevamente calculamos las derivadas aproximadas en $N = 32$ puntos del intervalo $[0, 2\pi]$ como sigue:

```
per=2*pi;
h=per/32;
x=[0:h:per-h];
y=sin(x/2);
yp=real(dfft(y));
z=[0:.01:per-h];
ypexact=0.5*cos(z/2);
```

```
pp=spline(x,yp);
ypapprox=ppval(pp,z);
plot(x,yp,'kx',z,ypapprox,'k:',z,ypexact,'k')
legend('Aproximaciones','Derivada Aproximada','Derivada Exacta')
```

Los resultados se muestran en la Figura 6.3. □

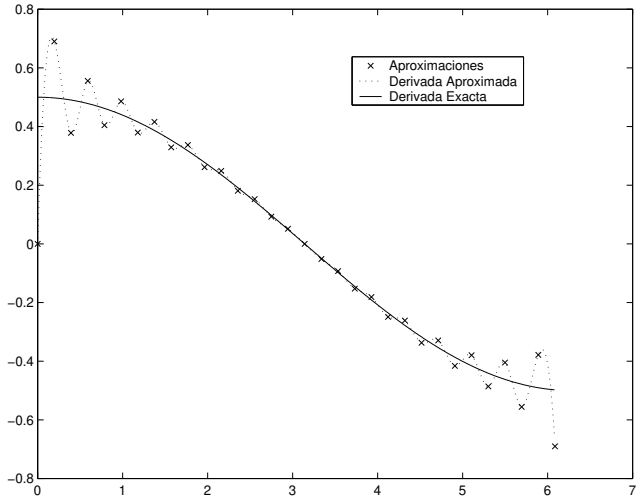


Figura 6.3: Derivadas aproximadas y exactas para la función (6.27).

Concluimos ésta sección con una aplicación de la diferenciación espectral a la solución de una ecuación *diferencial parcial*.

Ejemplo 6.10. Considere el problema de hallar una función $u(x,t)$ que satisfaga la ecuación diferencial[1]:

$$\frac{\partial u}{\partial t}(x,t) + c(x)\frac{\partial u}{\partial x}(x,t) = 0, \quad 0 < t < t_{\max}, \quad x \in \mathbb{R}, \tag{6.28}$$

donde $c(x) = 0.2 + \text{sen}^2(x-1)$. Buscamos una solución periódica en x y que cumpla con la condición inicial:

$$u(x,0) = \frac{3}{5 - 4\cos x}. \tag{6.29}$$

[1]La función $u(x,t)$ representa una onda o frente que se mueve en el tiempo. La ecuación diferencial especifica dicho movimiento por efectos de transporte activo únicamente.

Vamos ahora a discretizar la ecuación diferencial: la derivada en t la aproximamos con la fórmula (6.7) mientras que usamos derivadas espectrales en x aprovechando la periodicidad de u en esa dirección. Para $n, m \geq 0$ definimos

$$h = \frac{2\pi}{n}, \quad \delta t = \frac{t_{\max}}{m},$$

$$x_i = ih, \quad 0 \leq i \leq n, \quad t_j = j\delta t, \quad 0 \leq j \leq m.$$

Denotamos por u_{ij} una aproximación de $u(x_i, t_j)$, y por $u_{ij,x}$ la aproximación espectral de la derivada $u_x(x_i, t_j)$. La discretización de (6.28) que usamos está dada por:

$$u_{i\,j+1} = u_{i\,j-1} - 2\delta t\, c(x_i) u_{ij,x}, \quad 0 \leq i \leq n, \quad 1 \leq j \leq m. \tag{6.30}$$

Esto se conoce como un *método de colocación espectral*. Los valores de (u_{i0}) se obtienen de (6.29). Para calcular con la fórmula (6.30) necesitamos todavía los (u_{i1}). Para ésto usamos la discretización de orden uno de (6.28) en $t = \delta t$ basada en la fórmula (6.3), esto es:

$$u_{i1} = u_{i0} - \delta t\, c(x_i) u_{i0,x}, \quad 0 \leq i \leq n. \tag{6.31}$$

El siguiente programa en MATLAB incorpora las ecuaciones (6.30) y (6.31) para calcular una aproximación de $u(x, t)$ hasta $t_{\max} = 16$:

```
n=128;m=4000;
per=2*pi;tmax=16;
h=per/n;
dt=tmax/m;
il=100;
ml=round(m/il);
x=[0:h:per-h];
c=0.2+sin(x-1).^2;
u=zeros(ml+1,n);
u(1,:)=3./(5-4*cos(x));
w=real(dfft(u(1,:)));
u(2,:)=u(1,:)-dt*c.*w;
u1=u(1,:);
u2=u(2,:);
for j=2:ml+1
   for k=1:il
      w=real(dfft(u2));
      unew=u1-2*dt*c.*w;
```

```
      u1=u2;
      u2=unew;
   end
   u(j,:)=unew;
end
mesh(x,[0:(tmax/ml):tmax],u)
colormap([.0 .0 .0])
```

La solución aproximada se muestra en la Figura 6.4. □

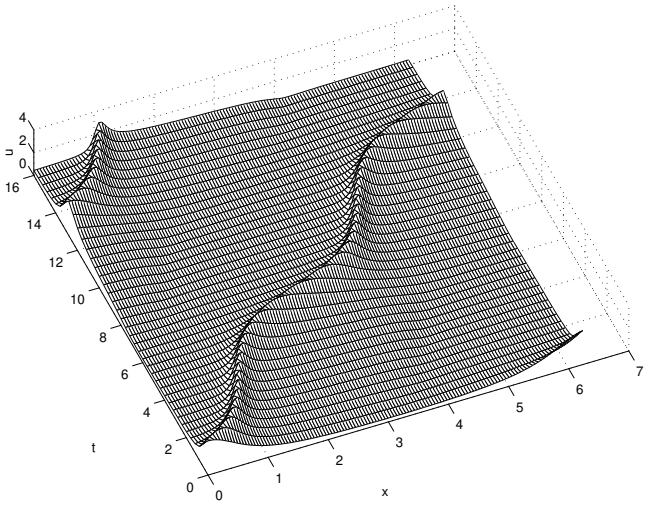


Figura 6.4: Solución aproximada para el problema (6.28), (6.29), con u periódica en x, calculada con un método de colocación espectral.

6.2 Integración Numérica

En esta sección comenzamos el estudio de métodos numéricos para la aproximación de integrales de la forma

$$I(f) = \int_a^b f(x)\,dx. \tag{6.32}$$

Un método común para aproximar $I(f)$ se basa en reemplazar o aproximar a $f(x)$ con un polinomio de interpolación. Este procedimiento se conoce como las reglas de *cuadratura de Newton*. Examinamos los primeros dos casos de éste método donde se usan polinomios de interpolación lineales y cuadráticos.

6.2.1 Método del Trapezoide: Fórmulas Básica y Compuesta

Sea $p_1(x)$ el polinomio lineal que interpola a $f(x)$ en $x = a$ y $x = b$, i.e.,

$$p_1(x) = \frac{(b-x)f(a) + (x-a)f(b)}{b-a}. \tag{6.33}$$

Usando la fórmula para el área de un trapezoide, o integrando $p_1(x)$ directamente, se obtiene que

$$\int_a^b p_1(x)\,dx = \frac{1}{2}(b-a)(f(a)+f(b)). \tag{6.34}$$

Así que podemos escribir la aproximación:

$$I(f) \approx \frac{b-a}{2}(f(a)+f(b)). \tag{6.35}$$

Esta formula se conoce como la *formula básica del método del trapezoide*. Más adelante analizaremos en detalles el error en ésta aproximación. Por el momento basta observar que la aproximación es buena siempre que f sea aproximadamente lineal. De hecho se puede verificar que la fórmula de arriba es exacta para polinomios de grado a lo más uno (Ejercicio 6.15).

En el caso general, dividimos el intervalo $[a, b]$ en subintervalos más pequeños y aplicamos la fórmula básica en cada subintervalo. Si los subintervalos son suficientemente pequeños, entonces f es aproximadamente lineal en cada subintervalo y la aproximación sería buena. Definimos el largo de los subintervalos por:

$$h = \frac{b-a}{n}, \quad n \geq 1. \tag{6.36}$$

El j–esimo subintervalo está dado por $[x_{j-1}, x_j]$ donde

$$x_j = a + jh, \quad 0 \leq j \leq n. \tag{6.37}$$

Note que

$$I(f) = \int_a^b f(x)\,dx = \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x)\,dx. \tag{6.38}$$

Usando la aproximación (6.35), podemos escribir que

$$\int_{x_{j-1}}^{x_j} f(x)\,dx \approx \frac{h}{2}(f(x_{j-1}) + f(x_j)). \tag{6.39}$$

Usando ésto en la fórmula anterior, obtenemos que

$$\begin{aligned} I(f) &\approx \frac{h}{2}\sum_{j=1}^{n}(f(x_{j-1})+f(x_j)), \\ &= \frac{h}{2}(f(x_0)+f(x_n))+h\sum_{j=1}^{n-1}f(x_j)\equiv T_n(f). \end{aligned} \tag{6.40}$$

Esta formula se conoce como la *regla (compuesta) del trapezoide* para aproximar $I(f)$.

Ejemplo 6.11. Usando la regla del trapezoide con $n = 2$ y $n = 4$, vamos a aproximar:

$$\int_1^2 \frac{dx}{x}.$$

Usamos que el valor exacto es $\ln(2) \approx 0.6931472$, correcto al número de cifras mostradas. Para $n = 2$ tenemos que $h = (2-1)/2 = 0.5$, $x_0 = 1$, $x_1 = 1.5$, $x_2 = 2$. Usando (6.40) tenemos que

$$\begin{aligned} \ln(2) &\approx \frac{h}{2}(f(x_0)+f(x_2))+hf(x_1), \\ &= \frac{0.5}{2}\left(\frac{1}{1}+\frac{1}{2}\right)+0.5\frac{1}{1.5}, \\ &= 0.708333. \end{aligned}$$

Con $n = 4$ tenemos $h = (2-1)/4 = 0.25$, $x_0 = 1$, $x_1 = 1.25$, $x_2 = 1.5$, $x_3 = 1.75$, $x_4 = 2$, y con (6.40) tenemos ahora que

$$\begin{aligned} \ln(2) &\approx \frac{h}{2}(f(x_0)+f(x_4))+h\sum_{j=1}^{3}f(x_j), \\ &= \frac{0.25}{2}\left(\frac{1}{1}+\frac{1}{2}\right)+0.25\left(\frac{1}{1.25}+\frac{1}{1.5}+\frac{1}{1.75}\right), \\ &= 0.69702. \end{aligned}$$

□

Ejemplo 6.12. En este problema usamos la regla del trapezoide para generar una tabla de aproximaciones del integral

$$\int_0^1 \mathrm{e}^x \operatorname{sen} x \,\mathrm{d}x,$$

cuyo valor exacto a seis cifras es 0.909331. Estos cálculos los podemos realizar utilizando la función `trapz` de MATLAB, la cual usa la regla (6.40) para aproximar integrales. En la tabla que generamos a continuación calculamos los cocientes sucesivos de los errores[2] y podemos así estimar el orden de convergencia de las aproximaciones. Veamos:

```
iexacto=(exp(1)*(-cos(1)+sin(1))+1)/2;
n=2;
error1=0;
for i=1:10
   x=linspace(0,1,n+1);
   y=exp(x).*sin(x);
   iaprox=trapz(x,y);
   error=iexacto-iaprox;
   ratio=error1/error;
   disp(['n=' num2str(n) ', iaprox=' num2str(iaprox,6) ',error='...
   num2str(error,6)  ',ratio=' num2str(ratio,6)])
   n=2*n;
   error1=error;
end
```

Los resultados fueron los siguientes:

n	$T_n(f)$	$e_n = I(f) - T_n(f)$	e_n/e_{2n}
2	0.967058	-0.0577277	—
4	0.923705	-0.0143741	4.0161
8	0.912921	-0.00358984	4.0041
16	0.910228	-0.000897229	4.00103
32	0.909555	-0.000224293	4.00026
64	0.909387	-5.60723e-005	4.00006
128	0.909345	-1.4018e-005	4.00002
256	0.909334	-3.5045e-006	4
512	0.909332	-8.76125e-007	4
1024	0.909331	-2.19031e-007	4

Estos resultados confirman claramente la convergencia del método del trapezoide en este ejemplo particular. Podemos ver que cada vez que se duplica la n, lo cual

[2]En este problema como conocemos el valor exacto del integral, podemos calcular de forma exacta los errores, es decir, la diferencia entre la aproximación generada por el método y el valor del integral.

equivale a dividir la h entre dos, el error disminuye por un factor de cuatro aproximadamente (última columna de la tabla). Esto es característico de convergencia $O(h^2)$. Esto lo vamos a confirmar teóricamente en la proxima sección. □

6.2.2 Método del Trapezoide: Análisis de Convergencia

Vamos ahora a examinar de forma más detallada el error de aproximación en la fórmula:

$$T_n(f) = \frac{h}{2}\sum_{j=1}^{n}(f(x_{j-1}) + f(x_j)). \tag{6.41}$$

Usando el Teorema 5.5 sobre el error de interpolación, tenemos que si $p_1(x)$ es el polinomio que interpola a $f(x)$ en x_{j-1}, x_j entonces:

$$f(x) = p_1(x) + \frac{1}{2}(x - x_j)(x - x_{j-1})f''(c_x), \quad x, c_x \in [x_{j-1}, x_j]. \tag{6.42}$$

De modo que

$$\begin{aligned}
\int_{x_{j-1}}^{x_j} f(x)\,dx &= \int_{x_{j-1}}^{x_j} p_1(x)\,dx + \frac{1}{2}\int_{x_{j-1}}^{x_j}(x - x_{j-1})(x - x_j)f''(c_x)\,dx, \\
&= \frac{h}{2}(f(x_{j-1}) + f(x_j)) + \frac{1}{2}f''(c_j)\int_{x_{j-1}}^{x_j}(x - x_{j-1})(x - x_j)\,dx, \\
&= \frac{h}{2}(f(x_{j-1}) + f(x_j)) - \frac{h^3}{12}f''(c_j), \quad c_j \in [x_{j-1}, x_j],
\end{aligned}$$

donde usamos el Teorema del Valor Medio para integrales (Teorema A.2). Tenemos entonces que el error en la fórmula del método del trapezoide está dado por

$$E_n^T(f) \equiv \int_a^b f(x)\,dx - T_n(f) = -\frac{h^3}{12}\sum_{j=1}^{n} f''(c_j). \tag{6.43}$$

Suponiendo que $f'' \in C[a, b]$ (continua en $[a, b]$) y aplicando el Teorema del Valor Intermedio para funciones, obtenemos que (ver Ejercicio 1.16):

$$\sum_{j=1}^{n} f''(c_j) = nf''(\xi), \quad \xi \in [a, b]. \tag{6.44}$$

Usando este resultado en la fórmula del error de arriba obtenemos que

$$E_n^T(f) = -\frac{h^2}{12}f''(\xi)(b - a). \tag{6.45}$$

Esta fórmula se conoce como la *fórmula exacta del error* y establece de forma clara que la fórmula del trapezoide $T_n(f)$ tiene un orden de convergencia de $O(h^2)$. Esto confirma nuestros resultados numéricos de la sección anterior.

El resultado (6.43), (6.45) es válido para funciones con dos derivadas continuas en $[a, b]$. Si la función f no es suficientemente diferenciable, entonces el método todavía converge si f es al menos continua, pero el orden de convergencia es menor de $O(h^2)$.

Ejemplo 6.13. La función $f(x) = 1 - \sqrt{1-(x-1)^2}$ es continua en $[0, 1]$ pero no es diferenciable en $x = 0$. Vamos a calcular numéricamente

$$\int_0^1 \left[1 - \sqrt{1-(x-1)^2}\right] \, \mathrm{d}x.$$

El valor exacto de este integral es $1 - \pi/4 = 0.2146\ldots$. Los resultados de la aplicación del método del trapezoide para calcular el integral se muestran en la Tabla 6.1. Note que las aproximaciones convergen al valor exacto pero con un orden de convergencia de 1.5 aproximadamente[3].

De hecho el orden de convergencia de las aproximaciones depende del grado o la severidad de la singularidad. Ilustramos ésto último considerando aproximaciones de la integral

$$\int_0^1 x^{1/\alpha} \, \mathrm{d}x = \frac{\alpha}{1+\alpha}.$$

En la Tabla 6.2 mostramos los cocientes $\gamma_n = e_n/e_{2n}$ del método del trapezoide para $n = 512$ y los estimados del orden de convergencia r_α para valores de $\alpha = 2, 3, \ldots, 10$. Podemos ver que las aproximaciones convergen con error de aproximadamente $O(h^{1+1/\alpha})$. □

Fórmula Asintótica del Error

La fórmula (6.45), aparte de su valía teórica, no es un estimador práctico del error ya que en general el punto ξ de la fórmula es desconocido, y f'' podría no estar disponible o que su cálculo no sea práctico. Un estimador más viable del error está dado por la fórmula

$$\hat{E}_n^T(f) = -\frac{h^2}{12}(f'(b) - f'(a)). \tag{6.46}$$

[3]Si las aproximaciones están convergiendo con orden $O(h^r)$, entonces para n grande, $\gamma_n \equiv e_n/e_{2n} \approx 2^r$. Así que el orden de convergencia se puede estimar con $r \approx \log(\gamma_n)/\log(2)$, para n grande.

n	$T_n(f)$	$e_n = I(f) - T_n(f)$	e_n/e_{2n}
2	0.316987	-0.102385	—
4	0.251073	-0.0364709	2.80732
8	0.227545	-0.0129434	2.81773
16	0.219187	-0.0045849	2.82304
32	0.216224	-0.00162256	2.82573
64	0.215176	-0.000573935	2.82707
128	0.214805	-0.000202965	2.82775
256	0.214674	-7.17677e-005	2.82809
512	0.214627	-2.53752e-005	2.82826
1024	0.214611	-8.97176e-006	2.82834

Tabla 6.1: El método del trapezoide aplicado a la función $f(x) = 1 - \sqrt{1-(x-1)^2}$ del Ejemplo 6.13.

α	γ_n	r_α	α	γ_n	r_α
2	2.821	1.4962	7	2.208	1.1428
3	2.5184	1.3325	8	2.1809	1.1249
4	2.3778	1.2496	9	2.16	1.1111
5	2.2971	1.1998	10	2.1435	1.1
6	2.2447	1.1665	—	—	—

Tabla 6.2: Estimados para el orden de convergencia del método del trapezoide aplicado a la función $f(x) = x^{1/\alpha}$, $x \in [0,1]$.

$\hat{E}_n^T(f)$ se conoce como el *estimador asintótico del error* y es una aproximación $O(h^4)$ de $E_n^T(f)$ (cf. (6.51)). De la fórmula asintótica $\hat{E}_n^T(f)$ tenemos que

$$\lim_{n\longrightarrow\infty} \frac{E_n^T(f)}{E_{2n}^T(f)} = \lim_{n\longrightarrow\infty} \frac{\hat{E}_n^T(f)}{\hat{E}_{2n}^T(f)} = \lim_{n\longrightarrow\infty} \frac{-\frac{h^2}{12}(f'(b)-f'(a))}{-\frac{(h/2)^2}{12}(f'(b)-f'(a))} = 4,$$

lo cual fue lo que observamos en nuestros cómputos numéricos. Note también que si casualmente $f'(a) = f'(b)$, entonces la convergencia del método será más rápida de $O(h^2)$.

Note que si el cómputo de f' es práctico, la fórmula asintótica envuelve cantidades conocidas y es posible calcularla como parte del proceso de aproximación. Más aun podemos utilizar el estimador para corregir la fórmula del trapezoide

obteniendo así la *fórmula del trapezoide corregida*:

$$CT_n(f) = T_n(f) + \hat{E}_n^T(f), \tag{6.47}$$

la cúal en general debe ser una mejor aproximación a $I(f)$ que $T_n(f)$.

Ejemplo 6.14. Consideremos nuevamente el problema de aproximar

$$\ln(2) = \int_1^2 \frac{dx}{x}.$$

En este caso $f(x) = 1/x$ de modo que $f'(x) = -1/x^2$, así que el estimador asintótico del error está dado por la fórmula

$$\hat{E}_n^T(f) = -\frac{h^2}{16}.$$

Como en este problema el valor exacto del error $E_n^T(f)$ se conoce, podemos compararlo con la fórmula asintótica. Veamos los resultados:

n	$T_n(f)$	$CT_n(f)$	$E_n^T(f)$	$\hat{E}_n^T(f)$
2	0.708333	0.692708	-0.0151862	-0.015625
4	0.697024	0.693118	-0.00387663	-0.00390625
8	0.694122	0.693145	-0.00097467	-0.000976562
16	0.693391	0.693147	-0.000244022	-0.000244141
32	0.693208	0.693147	-0.0000610277	-0.0000610352
64	0.693162	0.693147	-0.0000152583	-0.0000152588
128	0.693151	0.693147	-3.81467e-006	-3.8147e-006
256	0.693148	0.693147	-9.53672e-007	-9.53674e-007
512	0.693147	0.693147	-2.38418e-007	-2.38419e-007
1024	0.693147	0.693147	-5.96046e-008	-5.96046e-008

Podemos ver de esta tabla que la fórmula asintótica del error predice bastante bien el error real en este ejemplo. También podemos observar que la fórmula corregida del trapezoide produce seis cifras correctas en la aproximación con apenas $n = 16$ mientras que la fórmula sin corregir requiere hasta $n = 512$. □

Cuando la función $f \in C^4[a, b]$, podemos obtener una representación más detallada del error (6.45) la cual nos permite justificar la convergencia de $O(h^4)$ del estimador (6.46). También utilizaremos esta fórmula detalla en nuestra discusión de la técnica de extrapolación de Richardson (Sección (6.2.5)).

Usando el Teorema de Taylor (Teorema 1.2) y suponiendo que $f \in C^4[a, b]$, podemos escribir que

$$f(x) = p_1(x) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \frac{1}{6}\int_a^x f^{(4)}(\xi)(x-\xi)^3\,\mathrm{d}\xi,$$

donde $p_1(x) = f(a) + f'(a)(x-a)$. De aquí que

$$\begin{aligned} I(f) &= I(p_1) + \frac{1}{6}f''(a)(b-a)^3 + \frac{1}{24}f'''(a)(b-a)^4 \\ &\quad + \frac{1}{6}\int_a^b \int_a^x f^{(4)}(\xi)(x-\xi)^3 \,\mathrm{d}\xi \,\mathrm{d}x, \\ &= I(p_1) + \frac{1}{6}f''(a)(b-a)^3 + \frac{1}{24}f'''(a)(b-a)^4 \\ &\quad + \frac{1}{24}\int_a^b f^{(4)}(\xi)(b-\xi)^4 \,\mathrm{d}\xi. \end{aligned}$$

Tenemos también que

$$\begin{aligned} T_1(f) &= T_1(p_1) + \frac{1}{4}f''(a)(b-a)^3 + \frac{1}{12}f'''(a)(b-a)^4 \\ &\quad + \frac{1}{12}(b-a)\int_a^b f^{(4)}(\xi)(b-\xi)^3 \,\mathrm{d}\xi. \end{aligned}$$

Como la regla del trapezoide es exacta para polinomios de grado menor o igual a uno, tenemos que $I(p_1) = T_1(p_1)$. Usando esto y las expresiones para $I(f)$ y $T_1(f)$ derivadas arriba, tenemos que

$$\begin{aligned} E_1^T(f) &= -\frac{1}{12}f''(a)(b-a)^3 - \frac{1}{24}f'''(a)(b-a)^4 \\ &\quad + \int_a^b f^{(4)}(\xi)\left[\frac{1}{24}(b-\xi)^4 - \frac{1}{12}(b-a)(b-\xi)^3\right] \mathrm{d}\xi. \end{aligned} \tag{6.48}$$

Usando el Teorema de Taylor nuevamente, podemos escribir que

$$f'(b) = f'(a) + f''(a)(b-a) + \frac{1}{2}f'''(a)(b-a)^2 + \frac{1}{2}\int_a^b f^{(4)}(\xi)(b-\xi)^2 \,\mathrm{d}\xi.$$

Si despejamos en esta expresión para $f''(a)$ y sustituimos luego en (6.48), obtenemos que

$$\begin{aligned} E_1^T(f) &= -\frac{1}{12}(b-a)^2\left(f'(b) - f'(a)\right) \\ &\quad + \int_a^b f^{(4)}(\xi)\left[\frac{1}{24}(b-\xi)^4 - \frac{(b-a)}{12}(b-\xi)^3 + \frac{(b-a)^2}{24}(b-\xi)^2\right] \mathrm{d}\xi. \end{aligned}$$

Es fácil verificar ahora que la función que aparece entre corchetes dentro del integral, es positiva en $[a,b]$. Podemos pues aplicar el teorema del valor medio para integrales (Teorema A.2) para obtener que

$$E_1^T(f) = -\frac{1}{12}(b-a)^2\left(f'(b) - f'(a)\right)$$

$$
\begin{aligned}
&+f^{(4)}(\eta)\int_a^b \left[\frac{1}{24}(b-\xi)^4 - \frac{(b-a)}{12}(b-\xi)^3 + \frac{(b-a)^2}{24}(b-\xi)^2\right] \mathrm{d}\xi, \\
= \; & -\frac{1}{12}(b-a)^2\left(f'(b)-f'(a)\right) + \frac{1}{720}(b-a)^5 f^{(4)}(\eta), \qquad (6.49)
\end{aligned}
$$

donde $\eta \in [a,b]$.

Para la regla compuesta (6.41) tenemos, usando (6.38), que

$$
\begin{aligned}
E_n^T(f) &= I(f) - T_n(f), \\
&= \sum_{j=1}^{n}\left[\int_{x_{j-1}}^{x_j} f(x)\,\mathrm{d}x - \frac{h}{2}\left(f(x_{j-1})+f(x_j)\right)\right].
\end{aligned}
$$

Usando (6.49) obtenemos que

$$
\int_{x_{j-1}}^{x_j} f(x)\,\mathrm{d}x - \frac{h}{2}\left(f(x_{j-1})+f(x_j)\right) = -\frac{h^2}{12}\left(f'(x_j)-f'(x_{j-1})\right) + \frac{h^5}{720}f^{(4)}(\eta_j),
$$

donde $\eta_j \in [x_{j-1}, x_j]$, $j = 1, 2, \ldots, n$. Usando ésto en la expresión anterior de $E_n^T(f)$ tenemos que

$$
\begin{aligned}
E_n^T(f) &= \sum_{j=1}^{n}\left[-\frac{h^2}{12}\left(f'(x_j)-f'(x_{j-1})\right) + \frac{h^5}{720}f^{(4)}(\eta_j)\right], \\
&= -\frac{h^2}{12}\left(f'(b)-f'(a)\right) + \frac{h^4}{720}\frac{b-a}{n}\sum_{j=1}^{n} f^{(4)}(\eta_j), \\
&= -\frac{h^2}{12}\left(f'(b)-f'(a)\right) + \frac{h^4}{720}(b-a)f^{(4)}(\gamma), \qquad (6.50)
\end{aligned}
$$

donde $\gamma \in [a,b]$. Tenemos ahora usando la definición (6.46) que

$$
E_n^T(f) = \hat{E}_n^T(f) + O(h^4). \qquad (6.51)
$$

Podemos ahora también analizar el error en la fórmula corregida $CT_n(f)$:

$$
I(f) - CT_n(f) = I(f) - (T_n(f) + \hat{E}_n^T(f)) = E_n^T(f) - \hat{E}_n^T(f) = O(h^4),
$$

ésto es, $CT_n(f)$ es una aproximación $O(h^4)$ de $I(f)$.

6.2.3 Regla de Simpson: Fórmulas Básica y Compuesta

Utilizamos ahora un polinomio de interpolación cuadrático para aproximar $f(x)$ en (6.32). Sea $p_2(x)$ el polinomio de grado (a lo más) dos que interpola a $f(x)$ en

a, $c = (a+b)/2$, b. Este polinomio se puede escribir mediante la representación de Lagrange como:

$$\begin{aligned} p_2(x) &= \frac{(x-c)(x-b)}{(a-c)(a-b)}f(a) + \frac{(x-a)(x-b)}{(c-a)(c-b)}f(c) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(b), \\ &\equiv \ell_1(x)f(a) + \ell_2(x)f(c) + \ell_3(x)f(b). \end{aligned} \tag{6.52}$$

Tenemos ahora que

$$\begin{aligned} I(f) &\approx \int_a^b p_2(x)\,dx = f(a)\int_a^b \ell_1(x)\,dx \\ &\quad + f(c)\int_a^b \ell_2(x)\,dx + f(b)\int_a^b \ell_3(x)\,dx. \end{aligned} \tag{6.53}$$

Si escribimos $h = (b-a)/2$, y hacemos el cambio de variable $u = x - a$, tenemos que

$$\begin{aligned} \int_a^b \ell_1(x)\,dx &= \int_a^b \frac{(x-c)(x-b)}{(a-c)(a-b)}dx = \frac{1}{2h^2}\int_a^{a+2h}(x-c)(x-b)dx, \\ &= \frac{1}{2h^2}\int_0^{2h}(u-h)(u-2h)du = \frac{h}{3}. \end{aligned} \tag{6.54}$$

De forma similar se obtiene que

$$\int_a^b \ell_2(x)\,dx = \frac{4h}{3}, \quad \int_a^b \ell_3(x)\,dx = \frac{h}{3}. \tag{6.55}$$

Tenemos entonces que la aproximación (6.53) es equivalente a:

$$I(f) \approx \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right). \tag{6.56}$$

Este fórmula tiene la propiedad de ser exacta para polinomios de grado tres o menos[4] (Ejercicio 6.15).

Por un argumento similar al que hicimos para el método del trapezoide, tenemos que si n es un entero par (¿por qué?) entonces

$$I(f) = \sum_{j=1}^{n/2}\int_{x_{2(j-1)}}^{x_{2j}} f(x)\,dx, \tag{6.57}$$

[4]Claramente la fórmula debe ser exacta para polinomios de grado dos lo que le daría a la fórmula compuesta un orden de convergencia $O(h^3)$. Pero al ser la fórmula básica exacta para polinomios de grado tres, se obtiene un grado más de aproximabilidad en la fórmula compuesta siendo ésta de $O(h^4)$.

donde los $\{x_j\}$ están dados por (6.37). Usando la fórmula (6.56) podemos aproximar

$$\int_{x_{2(j-1)}}^{x_{2j}} f(x)\,dx \approx \frac{h}{3}(f(x_{2(j-1)}) + 4f(x_{2j-1}) + f(x_{2j})). \tag{6.58}$$

Ahora

$$\begin{aligned} I(f) \approx S_n(f) &\equiv \frac{h}{3}\sum_{j=1}^{n/2}(f(x_{2(j-1)}) + 4f(x_{2j-1}) + f(x_{2j})), \\ &= \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \\ &\quad \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)). \end{aligned} \tag{6.59}$$

Esta fórmula se conoce como la *regla (compuesta) de Simpson* para aproximar a $I(f)$.

Ejemplo 6.15. Usando la regla de Simpson con $n = 2$ y $n = 4$ aproximamos:

$$\int_1^2 \frac{dx}{x},$$

cuyo valor exacto es $\ln(2) \approx 0.6931472$ correcto al número de cifras mostradas. Para $n = 2$ tenemos que $h = (2-1)/2 = 0.5$, $x_0 = 1$, $x_1 = 1.5$, $x_2 = 2$. Ahora

$$\ln(2) \approx \frac{0.5}{3}\left(\frac{1}{1} + 4\,\frac{1}{1.5} + \frac{1}{2}\right) = 0.69444.$$

Con $n = 4$ tenemos $h = (2-1)/4 = 0.25$, $x_0 = 1$, $x_1 = 1.25$, $x_2 = 1.5$, $x_3 = 1.75$, $x_4 = 2$, de modo que

$$\ln(2) \approx \frac{0.25}{3}\left(\frac{1}{1} + 4\,\frac{1}{1.25} + 2\,\frac{1}{1.5} + 4\,\frac{1}{1.75} + \frac{1}{2}\right) = 0.693254.$$

□

Ejemplo 6.16. Consideramos nuevamente el problema de aproximar el integral:

$$\int_0^1 \mathrm{e}^x \operatorname{sen} x\,\mathrm{d}x,$$

cuyo valor exacto a seis cifras es 0.909331. MATLAB no tiene una rutina `simp` equivalente a `trapz`. (No obstante, tiene una mejor llamada `quad`.) La subrutina `quad` utiliza una regla de Simpson adaptativa donde el valor de h se ajusta para que el error en la aproximación satisfaga una tolerancia especificada por el usuario.

También MATLAB tiene la subrutina `quad8` que al igual que `quad` usa un método adaptativo pero con una fórmula de aproximación de grado mayor. En lugar de usar estas rutinas que hacen las comparaciones entre métodos un tanto complicadas, implementamos nuestra versión de `simp` equivalente a `trapz`:

```
function q=simp(x,y);
n=length(x)-1;
if (n/2)~=floor(n/2)
  disp('n tiene que ser par');
  break;
end
h=x(2)-x(1);
v=2*ones(n+1,1);
v(2:2:n)=v(2:2:n)+2;
v(1)=1;v(n+1)=1;
q=(h/3)*y*v;
```

Esta subrutina implementa una forma vectorizada del método de Simpson que ejecuta eficientemente en MATLAB. Note que se requiere que n sea par. Recuerde también que en MATLAB los indices de los arreglos corren empezando en uno. El mismo programa del Ejemplo 6.12 lo podemos usar aquí pero reemplazando la llamada a `trapz` por `simp`. Se obtuvieron los siguientes resultados:

n	$S_n(f)$	$e_n = I(f) - S_n(f)$	e_n/e_{2n}
2	0.908185	0.0011454	—
4	0.909254	7.71398e-005	14.8484
8	0.909326	4.90556e-006	15.725
16	0.90933	3.07905e-007	15.9321
32	0.909331	1.92644e-008	15.9831
64	0.909331	1.20435e-009	15.9958
128	0.909331	7.52768e-011	15.9989
256	0.909331	4.70501e-012	15.9993
512	0.909331	2.94098e-013	15.9981
1024	0.909331	1.77636e-014	16.5562

Estos resultados confirman claramente la convergencia de la regla de Simpson en este ejemplo particular. Podemos ver que cada vez que se duplica la n, lo cuál equivale a dividir la h entre dos, el error disminuye por un factor de 16 aproximadamente (última columna de la tabla). Esto es característico de convergencia $O(h^4)$ lo cual confirmaremos de forma rigurosa en la proxima sección. □

6.2.4 Regla de Simpson: Análisis de Convergencia

Examinamos ahora en detalles el error de aproximación en la fórmula:

$$S_n(f) = \frac{h}{3}\sum_{j=1}^{n/2}(f(x_{2(j-1)}) + 4f(x_{2j-1}) + f(x_{2j})). \tag{6.60}$$

Procedemos de forma diferente al análisis del método del trapezoide para así aprovechar que la fórmula básica de Simpson es exacta para polinomios de grado a lo más tres (Ejercicio 6.15). Esto es:

$$S_1(x^k) = I(x^k) = \int_a^b x^k \,\mathrm{d}x, \quad k = 0, 1, 2, 3. \tag{6.61}$$

Usando el Teorema de Taylor (Teorema 1.2), suponiendo que $f \in C^4[a, b]$, podemos escribir que:

$$f(x) = p_3(x) + R_3(x) = \sum_{k=0}^{3}\frac{f^{(k)}(a)}{k!}(x-a)^k + \frac{1}{3!}\int_a^x f^{(4)}(\xi)(x-\xi)^3 \,\mathrm{d}\xi,$$

$x \in [a, b]$. Usando que $I(\cdot), S_1(\cdot)$ son operadores lineales[5] tenemos que

$$I(f) = I(p_3) + I(R_3), \quad S_1(f) = S_1(p_3) + S_1(R_3).$$

Usando ahora la propiedad (6.61) y la linealidad de $I(\cdot), S_1(\cdot)$, podemos concluir que $S_1(p_3) = I(p_3)$. De aquí que

$$E_1^S(f) = I(f) - S_1(f) = I(R_3) - S_1(R_3).$$

Note que intercambiando los límites de integración tenemos que,

$$\begin{aligned} I(R_3) &= \frac{1}{3!}\int_a^b \int_a^x f^{(4)}(\xi)(x-\xi)^3 \,\mathrm{d}\xi \,\mathrm{d}x, \\ &= \frac{1}{3!}\int_a^b f^{(4)}(\xi) \int_\xi^b (x-\xi)^3 \,\mathrm{d}x \,\mathrm{d}\xi, \\ &= \frac{1}{4!}\int_a^b f^{(4)}(\xi)(b-\xi)^4 \,\mathrm{d}\xi. \end{aligned}$$

[5]Esto es, $I(\alpha f + \beta g) = \alpha I(f) + \beta I(g)$ para cualesquiera funciones $f, g \in C[a, b]$ y $\alpha, \beta \in \mathbb{R}$, con una propiedad similar para $S_1(\cdot)$.

En adición, poniendo $c = (a+b)/2$, tenemos que

$$S_1(R_3) = \frac{b-a}{36}\left(4\int_a^c f^{(4)}(\xi)(c-\xi)^3\,\mathrm{d}\xi + \int_a^b f^{(4)}(\xi)(b-\xi)^3\,\mathrm{d}\xi\right).$$

Combinando los últimos dos resultados tenemos que

$$\begin{aligned} E_1^S(f) &= \int_a^c f^{(4)}(\xi)\left(\frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}\left(4(c-\xi)^3 + (b-\xi)^3\right)\right)\mathrm{d}\xi \\ &\quad + \int_c^b f^{(4)}(\xi)\left(\frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}(b-\xi)^3\right)\mathrm{d}\xi. \end{aligned}$$

Usando técnicas elementales de cálculo se puede verificar que las funciones que aparecen multiplicando a $f^{(4)}(\xi)$ en éstos integrales, son negativas en los correspondientes intervalos especificados por los límites de integración (Ejercicio 6.22). Así que podemos aplicar el teorema del valor medio para integrales (Teorema A.2) para obtener que

$$\begin{aligned} E_1^S(f) &= f^{(4)}(c_1)\int_a^c \left(\frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}\left(4(c-\xi)^3 + (b-\xi)^3\right)\right)\mathrm{d}\xi \\ &\quad + f^{(4)}(c_2)\int_c^b \left(\frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}(b-\xi)^3\right)\mathrm{d}\xi, \end{aligned}$$

donde $c_1 \in [a,c]$ y $c_2 \in [c,b]$. Usando nuevamente el Ejercicio 6.22 y el teorema del valor intermedio para funciones (Teorema A.3) tenemos que

$$\begin{aligned} E_1^S(f) &= -\frac{(b-a)^5}{5760}\left(f^{(4)}(c_1) + f^{(4)}(c_2)\right), \\ &= -\frac{(b-a)^5}{2880}f^{(4)}(c), \end{aligned} \tag{6.62}$$

donde $c \in [a,b]$.

El estimado del error (6.62) es para la fórmula básica de Simpson. Para la fórmula compuesta tenemos usando (6.57) que:

$$\begin{aligned} E_n^S(f) &= I(f) - S_n(f), \\ &= \sum_{j=1}^{n/2}\left[\int_{x_{2(j-1)}}^{x_{2j}} f(x)\,\mathrm{d}x - \frac{h}{3}(f(x_{2(j-1)}) + 4f(x_{2j-1}) + f(x_{2j}))\right]. \end{aligned}$$

Pero usando (6.62) tenemos que

$$\int_{x_{2(j-1)}}^{x_{2j}} f(x)\,\mathrm{d}x - \frac{h}{3}(f(x_{2(j-1)}) + 4f(x_{2j-1}) + f(x_{2j})) = -\frac{(2h)^5}{2880}f^{(4)}(c_j),$$

n	$S_n(f)$	$e_n = I(f) - S_n(f)$	e_n/e_{2n}
2	0.255983	-0.0413812	—
4	0.229101	-0.0144994	2.854
8	0.219703	-0.00510087	2.84253
16	0.216401	-0.00179875	2.83579
32	0.215237	-0.000635109	2.83219
64	0.214826	-0.000224394	2.83032
128	0.214681	-7.93087e-005	2.82938
256	0.21463	-2.80351e-005	2.8289
512	0.214612	-9.91107e-006	2.82867
1024	0.214605	-3.50394e-006	2.82855

Tabla 6.3: El método de Simpson aplicado a la función del Ejemplo 6.13.

donde $c_j \in [x_{2(j-1)}, x_{2j}]$. Así que

$$\begin{aligned} E_n^S(f) &= -\frac{(2h)^5}{2880}\sum_{j=1}^{n/2} f^{(4)}(c_j) = -\frac{h^4}{180}(2h)\sum_{j=1}^{n/2} f^{(4)}(c_j), \\ &= -\frac{h^4}{180}(b-a)\frac{2}{n}\sum_{j=1}^{n/2} f^{(4)}(c_j) = -\frac{h^4}{180}(b-a)f^{(4)}(c), \end{aligned} \tag{6.63}$$

donde $c \in [a, b]$, lo cuál se conoce como la fórmula exacta del error. De la fórmula del error exacto se obtiene que el método tiene un orden de convergencia $O(h^4)$.

Al igual que para el método del trapezoide, si la función f no es suficientemente diferenciable ($f \in C^4$ para las fórmulas (6.63), (6.64)), el orden de convergencia del método se reduce considerablemente. Los resultados de la aplicación del método de Simpson a la primera función del Ejemplo 6.13 se muestran en la Tabla 6.3. Note que las aproximaciones convergen al valor exacto pero con un orden de convergencia aproximadamente igual al del método del trapezoide.

Fórmula Asintótica del Error

Trabajando igual que en el método del trapezoide, se obtiene que la fórmula asintótica del error está dada por:

$$\hat{E}_n^S(f) = -\frac{h^4}{180}(f'''(b) - f'''(a)). \tag{6.64}$$

Si $f \in C^8[a,b]$ se puede verificar que

$$E_n^S(f) = \hat{E}_n^S(f) + O(h^8). \tag{6.65}$$

Usando ahora la fórmula asintótica del error podemos ver que los cocientes $E_n^S(f)/E_{2n}^S(f)$ son aproximadamente 16 para n suficientemente grande.

La fórmula corregida de Simpson está dada por:

$$CS_n(f) = S_n(f) + \hat{E}_n^S(f), \tag{6.66}$$

lo cuál es una aproximación $O(h^8)$ de $I(f)$.

6.2.5 Extrapolación de Richardson y Reglas de Cuadratura Gaussiana

Vamos ahora a estudiar dos procedimientos o técnicas para obtener fórmulas de integración numérica de orden arbitrario. Una de estas técnicas es la extrapolación de Richardson que es un proceso comúnmente utilizado en análisis numérico para acelerar la convergencia de sucesiones convergentes. La otra técnica son las reglas de cuadratura gaussiana que producen fórmulas de alto grado utilizando puntos distribuidos en forma no uniforme en el intervalo de integración.

Extrapolación de Richardson

Denotamos por I_n cualquier fórmula numérica para aproximar $I(f)$, e.g., la fórmula del Trapezoide o la regla de Simpson, usando n subdivisiones del intervalo [a,b]. La correspondiente fórmula asintótica del método nos garantiza que para alguna constante C:

$$I(f) - I_n = Ch^p + O(h^{2p}), \tag{6.67}$$

donde p es el orden de convergencia del método y $h = (b-a)/n$. Por ejemplo, ésta es la fórmula (6.50) para el método del trapezoide donde $p = 2$, y (6.65) en el método de Simpson donde $p = 4$. Podemos escribir ahora que

$$I(f) - I_{2n} = C\left(\frac{h}{2}\right)^p + O(h^{2p}). \tag{6.68}$$

Despejando para C en (6.67) y sustituyendo en (6.68) tenemos que

$$\begin{aligned} I(f) - I_{2n} &= \frac{1}{2^p}\left(\frac{I(f) - I_n}{h^p} + O(h^p)\right) h^p + O(h^{2p}), \\ &= \frac{1}{2^p}\left(I(f) - I_n\right) + O(h^{2p}) \end{aligned}$$

Si ahora en esta ecuación despejamos para $I(f)$, obtenemos que

$$I(f) = \frac{1}{2^p - 1}(2^p I_{2n} - I_n) + O(h^{2p}). \tag{6.69}$$

La fórmula

$$R_{2n} = \frac{1}{2^p - 1}(2^p I_{2n} - I_n), \tag{6.70}$$

se conoce como la *fórmula de extrapolación de Richardson*. De (6.69) vemos que ésta fórmula es $O(h^{2p})$, es decir, el orden de convergencia de la fórmula R_{2n} es el doble del de la formula original I_n.

Ejemplo 6.17. Considere el problema de aproximar el integral

$$\int_{\pi/4}^{\pi/2} \frac{\mathrm{d}x}{1 + \cos x},$$

cuyo valor exacto es $2 - \sqrt{2} = 0.585786\ldots$. El siguiente programa en MATLAB implementa el método de extrapolación de Richardson para la regla del trapezoide:

```
n=2;
x=linspace(pi/4,pi/2,3);
y=1./(1+cos(x));
iaproxn=trapz(x,y);
for i=2:5
    n=2*n;
    x=linspace(pi/4,pi/2,n+1);
    y=1./(1+cos(x));
    iaprox2n=trapz(x,y);
    richard=(4*iaprox2n-iaproxn)/3;
    disp(['n=' num2str(n) ', iaprox2n=' num2str(iaprox2n,6)...
    ',richard=' num2str(richard,6)])
    iaproxn=iaprox2n;
end
```

Los resultados obtenidos fueron como sigue:

n	I_{2n}	R_{2n}
4	0.588211	0.585821
8	0.586394	0.585789
16	0.585938	0.585787
32	0.585824	0.585786

Aquí podemos ver que ya para $n = 32$ la fórmula de Richardson tiene seis cifras correctas. Para el método del trapezoide la fórmula de Richardson es de orden $O(h^4)$ y no requiere derivadas de la función $f(x)$ en comparación con la fórmula corregida. □

El programa que se utilizó en el ejemplo anterior es un tanto ineficiente pues repite muchas evaluaciones de la función $f(x)$ cada ves que duplica el número de subdivisiones. Para la regla del trapezoide, note que con $h = (b-a)/n$, tenemos que

$$\begin{aligned} T_{2n}(f) &= \frac{h}{4}\left(f(x_0) + f(x_{2n})\right) + \frac{h}{2}\sum_{k=1}^{2n-1} f(x_k), \\ &= \frac{h}{4}\left(f(x_0) + f(x_{2n})\right) + \frac{h}{2}\sum_{k=1}^{n-1} f(x_{2k}) + \frac{h}{2}\sum_{k=1}^{n} f(x_{2k-1}), \\ &= \frac{1}{2}T_n(f) + \frac{h}{2}\sum_{k=1}^{n} f(x_{2k-1}). \end{aligned} \tag{6.71}$$

Note pues que al duplicar el número de subdivisiones, solo hay que realizar n evaluaciones nuevas las cuales corresponden a los puntos medios de los intervalos en la fórmula de tamaño n.

La fórmula de extrapolación de Richardson puede utilizarse ahora en forma recursiva generando fórmulas de orden cada vez más alto (el orden se duplica en cada nivel de la recursión). La fórmula que resulta de este procedimiento recursivo se conoce como la *fórmula de integración numérica de Romberg* ([4]).

Reglas de Cuadratura Gaussiana

Consideramos por el momento integrales de la forma

$$I(f) = \int_{-1}^{1} f(x)\,dx. \tag{6.72}$$

Note que si el integral está dado en un intervalo arbitrario $[a, b]$, entonces usando el cambio de variables

$$x = \frac{1}{2}(a + b + (b-a)t), \quad -1 \le t \le 1, \tag{6.73}$$

tenemos que

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b+a+t(b-a)}{2}\right)\,dt, \tag{6.74}$$

lo cuál nos da una integral en $[-1,1]$. Así que sin perdida de generalidad podemos asumir que el integral está dado en $[-1,1]$.

Sean $x_1, x_2, \ldots, x_n$ puntos (no necesariamente uniformemente distribuidos) en $[-1,1]$ y $w_1, w_2, \ldots, w_n$ números llamados los *pesos* (*weights*). Los puntos $\{x_j\}$ y los pesos $\{w_j\}$ se determinan de modo que la fórmula de integración numérica

$$I_n(f) = \sum_{j=1}^{n} w_j f(x_j), \tag{6.75}$$

sea exacta para polinomios de grado a lo más $2n-1$, i.e., $I_n(p) = I(p)$ para todo polinomio p de grado a lo más $2n-1$. Como I_n, I son *operadores lineales*, basta verificar que

$$I_n(x^i) = I(x^i), \quad 0 \leq i \leq 2n-1. \tag{6.76}$$

Usando (6.75), éstas condiciones son equivalentes al sistema:

$$\sum_{j=1}^{n} w_j x_j^i = \frac{1-(-1)^{i+1}}{i+1}, \quad 0 \leq i \leq 2n-1. \tag{6.77}$$

Esto es un sistema nolineal de $2n$ ecuaciones en $2n$ desconocidas (las x_j y las w_j). Este sistema se puede resolver numéricamente usando el método de Newton para sistemas nolineales. Pero en lugar de proceder de esta forma, utilizamos el hecho de que se puede demostrar ([2], [6]) que los x_j son los ceros del n–ésimo *polinomio de Legendre* $L_n(x)$. Estos polinomios se definen por la recursión:

$$\begin{gathered}(n+1)L_{n+1}(x) - (2n+1)xL_n(x) + nL_{n-1}(x) = 0, \quad n \geq 1, \\ L_0(x) = 1, \quad L_1(x) = x.\end{gathered} \tag{6.78}$$

En particular para el caso $n=2$, tenemos que $L_2(x) = (3/2)x^2 - (1/2)$ cuyos ceros son $\pm 1/\sqrt{3}$. También

$$L_3(x) = \frac{1}{2}(5x^3 - 3x), \quad L_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3), \tag{6.79}$$

de donde podemos obtener los x_j para las fórmulas de los casos $n = 3, 4$ respectivamente. Teniendo los x_j podemos ahora calcular los w_j resolviendo un sistema lineal de ecuaciones.

En el caso $n=1$, el sistema (6.77) reduce a $w_1 = 2$, $w_1 x_1 = 0$. De modo que $w_1 = 2$ y $x_1 = 0$. Tenemos pues la fórmula numérica $I_1(f) = 2f(0)$ lo cuál se conoce como la *fórmula del punto medio*.

Para $n = 2$, el sistema (6.77) reduce a:

$$\begin{cases} w_1 + w_2 &= 2, \\ x_1 w_1 + x_2 w_2 &= 0, \\ x_1^2 w_1 + x_2^2 w_2 &= 2/3, \\ x_1^3 w_1 + x_2^3 w_2 &= 0. \end{cases} \tag{6.80}$$

Como x_1, x_2 son las raíces de $L_2(x) = (3/2)x^2 - (1/2)$, tenemos que

$$x_1 = -\frac{1}{\sqrt{3}}, \quad x_2 = \frac{1}{\sqrt{3}}. \tag{6.81}$$

Sustituyendo estos valores en las primeras dos ecuaciones de (6.80) y resolviendo para los w_j, obtenemos que $w_1 = w_2 = 1$. Así que nuestra fórmula numérica en el caso $n = 2$ lee como sigue:

$$I_2(f) = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right). \tag{6.82}$$

Usando la fórmula básica (6.75) podemos construir una correspondiente fórmula compuesta. Hacemos esto para la fórmula básica (6.82) del caso $n = 2$. Recuerde que con $h = (b-a)/m$ y $x_j = a + jh$, $0 \le j \le m$, podemos escribir que

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{j=1}^{m} \int_{x_{j-1}}^{x_j} f(x)\,\mathrm{d}x.$$

Usando (6.74) y (6.82) tenemos que

$$\begin{aligned} \int_{x_{j-1}}^{x_j} f(x)\,\mathrm{d}x &= \frac{h}{2}\int_{-1}^{1} f\left(x_{j-1/2} + \frac{h}{2}t\right)\,\mathrm{d}t, \\ &\approx \frac{h}{2}\left[f\left(x_{j-1/2} - \frac{h}{2\sqrt{3}})\right) + f\left(x_{j-1/2} + \frac{h}{2\sqrt{3}})\right)\right], \end{aligned}$$

donde $x_{j-1/2} = x_{j-1} + h/2 = x_j - h/2$. La fórmula compuesta queda dada ahora por:

$$G_{m,2} = \frac{h}{2}\sum_{j=1}^{m}\left[f\left(x_{j-1/2} - \frac{h}{2\sqrt{3}})\right) + f\left(x_{j-1/2} + \frac{h}{2\sqrt{3}})\right)\right]. \tag{6.83}$$

Ejemplo 6.18. Aproximamos

$$\int_1^3 \frac{dx}{\sqrt{x^3+1}},$$

usando la regla de cuadratura gaussiana compuesta (6.83). En este caso no tenemos el valor exacto del integral pero podemos estimar el error relativo en las aproximaciones con la expresión $\left|G_{m,2} - G_{m/2,2}\right| / |G_{m,2}|$. El siguiente programa en MATLAB usa la fórmula (6.83) y el estimador del error relativo:

```
r=1/(2*sqrt(3));
m=2;
Gold=0;
for i=1:10
   index=[1:m];
   h=2/m;
   xm=1+(index-0.5)*h;
   xl=xm-r*h;
   xr=xm+r*h;
   yl=1./sqrt(xl.^3+1);
   yr=1./sqrt(xr.^3+1);
   Gm2=0.5*h*sum(yl+yr);
   erel=abs(Gm2-Gold)/abs(Gm2);
   disp(['m=' num2str(m) ', Gm2=' num2str(Gm2,6)...
         ', Error Relativo=' num2str(erel)])
   Gold=Gm2;
   m=2*m;
end
```

Los resultados se muestran a continuación:

m	$G_{m,2}$	$\left\|G_{m,2} - G_{2,m/2}\right\| / \|G_{m,2}\|$
2	0.743311	—
4	0.743093	0.0002937
8	0.743071	2.954e-005
16	0.74307	1.9303e-006
32	0.743069	1.2105e-007

De acuerdo al estimador del error relativo, la aproximación con $m = 32$ tiene aproximadamente seis cifras correctas. □

6.2.6 Integración Multiple

El cálculo numérico de integrales en dos o más variables es mucho más complejo que el caso de integrales en una dimensión. Esto, primeramente, por la gran variedad de regiones de integración en dos a más dimensiones. (En una dimensión

solo hay que considerar intervalos.) Además, si n es el número de subdivisiones por dimensión en la fórmula numérica, asumiendo la misma cantidad en todas las dimensiones, entonces el total de evaluaciones que requiere la fórmula numérica será de $O(n^d)$, donde d es el número total de dimensiones. Cuando d es grande, entonces tenemos un problema altamente computacional.

Los métodos numéricos para integrales multiples fluctúan entre aquellos basados en polinomios de interpolación en varias variables hasta los llamados métodos Monte Carlo de carácter estadístico. El enfoque que utilizaremos aquí es uno más simple en donde aplicamos en cada dimensión una de las fórmulas discutidas para una dimensión. Por ejemplo, podemos usar en cada dimensión la regla del trapezoide, o la de Simpson, etc., o combinaciones de éstas. Discutimos únicamente el caso de dos variables ya que las generalizaciones de las fórmulas que presentamos a más variables son directas. También sólo discutimos el uso de la regla del Trapezoide en cada dimensión y dejamos para los ejercicios algunas de las otras variantes.

Sea f una función suficientemente diferenciable en la región

$$R = \{(x, y) \,:\, a \leq x \leq b, \quad c \leq y \leq d\}\,. \tag{6.84}$$

Consideramos el integral

$$I(f) = \int_a^b \left[\int_c^d f(x, y)\, \mathrm{d}y \right] \mathrm{d}x. \tag{6.85}$$

Sean $n, m \geq 1$ y defina

$$h = \frac{b-a}{n}, \quad k = \frac{d-c}{m},$$

y las particiones

$$x_i = a + ih, \quad 0 \leq i \leq n, \quad y_j = c + jk, \quad 0 \leq j \leq m,$$

de $[a, b]$ y $[c, d]$ respectivamente. Entonces $R = \cup_{i,j} R_{ij}$ donde

$$R_{ij} = \{(x, y) \,:\, x_{i-1} \leq x \leq x_i, \quad y_{j-1} \leq y \leq y_j\}\,, \tag{6.86}$$

con $1 \leq i \leq n$, $1 \leq j \leq m$. Usando la fórmula (6.40) podemos escribir para cualquier $x \in [a, b]$ que

$$\int_c^d f(x, y)\, \mathrm{d}y \approx \frac{k}{2} \left[f(x, y_0) + f(x, y_m) \right] + k \sum_{j=1}^{m-1} f(x, y_j) = T_m(f(x, \cdot)).$$

Aplicando la fórmula (6.40) pero en la variable x tenemos ahora que:

$$I(f) \quad \approx \quad \int_a^b T_m(f(x, \cdot))\, \mathrm{d}x,$$

$$\begin{aligned} &\approx \frac{h}{2}\left[T_m(f(x_0,\cdot)) + T_m(f(x_n,\cdot))\right] \\ &\quad + h\sum_{i=1}^{n-1} T_m(f(x_i,\cdot)) \equiv T_{n,m}(f). \end{aligned} \tag{6.87}$$

Se puede verificar ([11], [6]) que la fórmula (6.87) es una aproximación $O(h^2+k^2)$ de $I(f)$. Una fórmula equivalente a (6.87) se puede obtener intercambio los papeles de la x y la y. (Vea el Ejercicio 6.31.) La fórmula (6.87) se puede implementar relativamente fácil en MATLAB aprovechando que la función `trapz` puede operar en matrices. La siguiente función en MATLAB calcula la fómula (6.87):

```
function y=trapz2(f,a,b,c,d,n,m)
h=(b-a)/n;
k=(d-c)/m;
x=linspace(a,b,n+1);
y=linspace(c,d,m+1);
[X,Y]=meshgrid(x,y);
Z=feval(f,X,Y);
y=h*k*trapz(trapz(Z));
```

Ejemplo 6.19. Consideramos el problema de aproximar el integral

$$\int_0^2 \left[\int_{-1}^0 (x^2y^2 + x)\,\mathrm{d}y\right]\,\mathrm{d}x,$$

cuyo valor exacto es $26/9 = 2.88888\ldots$. Usamos la función `trapz2` con $n = m$. Los resultados se muestran a continuación donde $e_n = I(f) - T_{n,n}(f)$:

n	$T_{n,n}(f)$	e_n/e_{2n}
2	3.125	—
4	2.9453	4.1846
8	2.9028	4.0467
16	2.8924	4.0117
32	2.8898	4.0029
64	2.8891	4.0007
128	2.8889	4.0002
256	2.8889	4

Podemos ver claramente una convergencia de orden dos en este caso. □

Consideramos ahora el caso de regiones de la forma:

$$G = \{(x,y) \,:\, a \le x \le b, \quad \phi_1(x) \le y \le \phi_2(x)\}\,, \tag{6.88}$$

donde ϕ_1, ϕ_2 son funciones continuas en $[a, b]$ con $\phi_1 \leq \phi_2$ y el integral[6]

$$I_G(f) = \int_a^b \left[\int_{\phi_1(x)}^{\phi_2(x)} f(x, y)\, \mathrm{d}y \right] \mathrm{d}x. \tag{6.89}$$

Para $x \in [a, b]$ fijo, consideramos el cambio de variables:

$$y = (\phi_2(x) - \phi_1(x))w + \phi_1(x), \quad 0 \leq w \leq 1.$$

Note que $\mathrm{d}y = (\phi_2(x) - \phi_1(x))\mathrm{d}w$ y podemos escribir (6.89) como

$$I_G(f) = \int_a^b \left[\int_0^1 (\phi_2(x) - \phi_1(x)) f\left(x, (\phi_2(x) - \phi_1(x))w + \phi_1(x)\right)\, \mathrm{d}w \right] \mathrm{d}x,$$

lo cual es un integral del tipo (6.85), i.e., con límites contantes.

Ejemplo 6.20. Considere el integral

$$\int_0^{\pi/2} \left[\int_0^x (x^3 y + \cos x)\, \mathrm{d}y \right] \mathrm{d}x,$$

cuyo valor exacto es $1.82260\ldots$. Con el cambio de variables $y = xw$ donde $0 \leq w \leq 1$, podemos transformar el integral a:

$$\int_0^{\pi/2} \left[\int_0^1 x(x^4 w + \cos x)\, \mathrm{d}w \right] \mathrm{d}x.$$

Este integral lo trabajamos ahora con la función `trapz2` con la cual obtenemos los siguientes resultados:

n	$T_{n,n}(f)$	e_n/e_{2n}
2	2.4312	—
4	1.9826	3.805
8	1.8631	3.9519
16	1.8328	3.988
32	1.8251	3.997
64	1.8232	3.9993
128	1.8228	3.9998
256	1.8226	4

□

[6] Suponemos que $\phi_1(x) = \phi_2(x)$ solo en un número finito de puntos $x \in [a, b]$.

6.3 Ejercicios

Ejercicio 6.1. Utilice las fórmulas correspondientes para aproximar la primera y segunda derivada de la función $f(x) = x^2\cos(x)$ en $x = 1$ y para $h = 0.1, 0.01$.

Ejercicio 6.2. Utilizando las fórmulas de orden dos para aproximar la primera y segunda derivada, aproxime las correspondientes derivadas de la función $f(x) = \arctan(x^2 - x + 1)$ en $x = 1$ y para $h = 0.1, 0.01$.

Ejercicio 6.3. Usando los datos de la siguiente tabla, aproxime $f''(0.5)$ para $h = 0.2, 0.1$:

x	$f(x)$	x	$f(x)$
0.3	7.3891	0.6	7.6141
0.4	7.4633	0.7	7.6906
0.5	7.5383		

Ejercicio 6.4. Usando el Teorema de Taylor verifique las fórmulas (6.8) y (6.11).

Ejercicio 6.5. Para la fórmula $D_h^{(2)}f(x)$ repita un proceso similar al del Ejemplo 6.1 donde h se disminuye hasta que el error en la fórmula empieza a aumentar.

Ejercicio 6.6. Suponga que los valores de una cierta función f son dados en x_0, x_1, x_2. Sea p_2 el polinomio de grado dos que interpola a f en los puntos dados. Use p_2 para construir una fórmula para aproximar $f'(c)$ donde $c = (x_0 + x_1)/2$.

Ejercicio 6.7. Usando el spline cúbico natural que interpola a los datos $f(-2) = 3$, $f(-0.5) = -1$, $f(1) = 1$,$f(3) = 5$, halle una aproximación de $f'(0)$ y $f''(0)$. **Ayuda:** Construya el spline en MATLAB y utilice la función `unmkpp` para extraer los coeficientes del polinomio correspondiente al intervalo $[-0.5, 1]$. (Vea el Ejemplo 5.12.)

Ejercicio 6.8. Para una función f con tres derivadas continuas, verifique que:

$$f'(x) = \frac{4f(x+h) - f(x+2h) - 3f(x)}{2h} + O(h^2).$$

Esto nos da la siguiente formula de orden dos para aproximar $f'(x)$:

$$D_h^+ f(x) = \frac{4f(x+h) - f(x+2h) - 3f(x)}{2h}.$$

Esta fórmula es útil en la solución de problemas de frontera con condiciones de frontera que envuelven la derivada de la función desconocida.

Ejercicio 6.9. Sea f una función suficientemente diferenciable en el intervalo $[a, a+h]$, y $Q(x)$ el polinomio cúbico tal que:

$$Q(a) = f(a), \quad Q'(a) = f'(a), \quad Q(a+h) = f(a+h), \quad Q'(a+h) = f'(a+h).$$

(Puede utilizar la fórmula de $Q(x)$ que aparece en el Ejercicio 5.20, luego de remplazar x_i, x_{i+1} con $a, a+h$ respectivamente.) Para $x \in [a, a+h]$ aproximamos $f'(x)$ con $Q'(x)$. Halle un estimado del error en dicha aproximación en términos de h.

Ejercicio 6.10. Considere el *problema de frontera*:

$$\begin{cases} \theta''(s) - \lambda \cos\theta(s) = 0, \quad 0 < s < 1, \\ \theta(0) = 0, \quad \theta(1) = 0. \end{cases}$$

Diseñe un método numérico para resolver éste problema que utilice la formula (6.10) para aproximar la segunda derivada con respecto a s. Resuelva el sistema de ecuaciones que resulta usando el método de Newton. (Vea la Sección 4.7.) Trace las soluciones calculadas para los valores $0, \pm 1, \pm 2$ del parámetro λ.

Ejercicio 6.11. Para una función $u(r, \theta)$ periodica en θ, considere el *problema de frontera*:

$$\begin{cases} \dfrac{\partial^2 u}{\partial r^2} + \dfrac{1}{r}\dfrac{\partial u}{\partial r} + \dfrac{1}{r^2}\dfrac{\partial^2 u}{\partial \theta^2} = 0, \quad 0 < r < 1, \quad \theta \in \mathbb{R}, \\ \\ u(1, \theta) = \operatorname{sen}(\theta)\cos(\theta). \end{cases}$$

Diseñe un método numérico para resolver éste problema que utilice las formulas (6.7) y (6.10) para aproximar las derivadas en r, y diferenciación espectral para aproximar la derivada en la variable θ. (Vea el Ejemplo 6.10.) Trace una gráfica de la solución calculada.

Ejercicio 6.12. Utilice la regla de Simpson aproximar el integral $\int_1^3 \ln(x)\,\mathrm{d}x$ para $n = 4, 8$ subdivisiones.

Ejercicio 6.13. Usando el método del trapecio, aproxime el área de la región que se muestra en la Figura 6.5. Todas las medidas en el diagrama están en "pies".

Ejercicio 6.14. Usando las reglas del trapezoide y de Simpson y los programas descritos en el capítulo, aproxime el siguiente integral:

$$\int_0^1 \frac{\mathrm{d}x}{x^2+1}.$$

El valor exacto de este integral es $\pi/4$. Use ésto para generar una tabla con las aproximaciones y los errores (exactos) y estime el orden de convergencia.

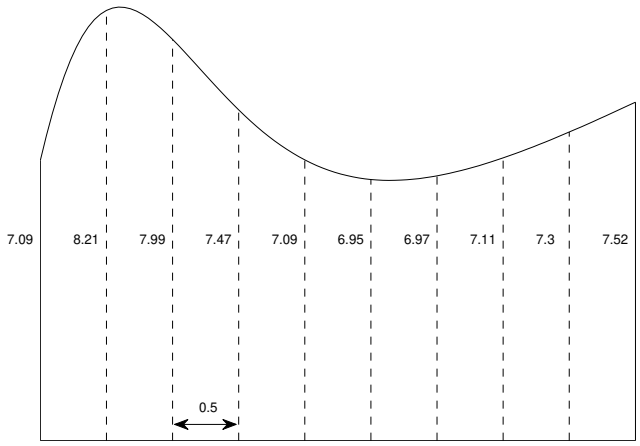


Figura 6.5: Región para el Ejercicio 6.13.

Ejercicio 6.15. Verifique que las reglas básicas del trapezoide y de Simpson dadas por:

$$T_1(f) = \frac{b-a}{2}\left(f(a)+f(b)\right), \quad S_1(f) = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right),$$

son exactas para polinomios de grado a lo más uno y tres respectivamente. Esto es, si $\phi_k(x) = x^k$, $k = 0, 1, 2, 3$, demuestre que

$$\begin{aligned} T_1(\phi_k) &= \int_a^b \phi_k(x)\,\mathrm{d}x, \quad k = 0, 1, \\ S_1(\phi_k) &= \int_a^b \phi_k(x)\,\mathrm{d}x, \quad k = 0, 1, 2, 3. \end{aligned}$$

Ejercicio 6.16. La regla del punto medio se puede usar para obtener la siguiente aproximación de $I(f)$:

$$I(f) = \int_a^b f(x)\,\mathrm{d}x \approx (b-a) f\left(\frac{a+b}{2}\right).$$

Usando esta fórmula diseñe una fórmula compuesta llamada la *regla (compuesta) del punto medio*. Use esta fórmula para aproximar la integral del Ejercicio 6.14 y estime el orden de convergencia de la misma.

Ejercicio 6.17. Utilizando un polinomio cúbico de Hermite para interpolar a $f(x)$ en $[a, b]$, desarrolle una fórmula para aproximar a $I(f)$. Obtenga la fórmula compuesta correspondiente.

Ejercicio 6.18. Repita los cálculos del Ejemplo 6.14 pero para el método de Simpson.

Ejercicio 6.19. Para la fórmulas básica y compuesta de la regla del punto medio del Ejercicio 6.16, haga un análisis de errores y derive las fórmulas exactas y asintóticas del error y la fórmula corregida del método.

Ejercicio 6.20. Si la regla del trapezoide se utiliza para aproximar $\int_2^5 \operatorname{sen}(x)\,dx$ con $h = 0.01$, halle una cota para el error en dicha aproximación y un estimado asintótico del mismo. Usando el estimador asintótico del error determine el número de subdivisiones necesarias n para que el error sea aproximadamente 10^{-8}.

Ejercicio 6.21. Utilice la regla del trapezoide con $n = 4$ para aproximar $\int_0^4 dx/(1+x^2)$. Usando el estimador asintótico del error determine el número de subdivisiones necesarias n para que el error sea aproximadamente 10^{-12}.

Ejercicio 6.22. Para $a < b$ y $c = (a+b)/2$ defina las funciones

$$\begin{aligned} F_1(\xi) &= \frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}\left(4(c-\xi)^3 + (b-\xi)^3\right), \quad \xi \in [a,c], \\ F_2(\xi) &= \frac{1}{4!}(b-\xi)^4 - \frac{b-a}{36}(b-\xi)^3, \quad \xi \in [c,b]. \end{aligned}$$

Verifique que $F_1, F_2 \leq 0$ en sus respectivos intervalos de definición y que

$$\int_a^c F_1(\xi)\,\mathrm{d}\xi = \int_c^b F_2(\xi)\,\mathrm{d}\xi = -\frac{(b-a)^5}{5760}.$$

Ejercicio 6.23. Si los puntos $\{x_0, x_1, \ldots, x_n\}$ no están uniformemente distribuidos en $[a,b]$, es posible generalizar la fórmula del trapezoide en este caso como sigue: sean $h_i = x_i - x_{i-1}$, $1 \leq i \leq n$. Entonces definimos

$$T_n(f) = \frac{1}{2}\sum_{i=1}^{n} h_i(f(x_{i-1}) + f(x_i)).$$

Verifique la siguiente fórmula para el error exacto de este método

$$\int_a^b f(x)\,\mathrm{d}x = T_n(f) - \frac{h^2}{12}(b-a)f''(\xi),$$

donde $\xi \in [a,b]$, $h \in [\min_i h_i, \max_i h_i]$.

Ejercicio 6.24. Trabaje el problema del Ejemplo 6.17 pero con la fórmula de extrapolación de Richardson que usa la regla de Simpson. ¿Cuál es el orden de convergencia de la fórmula de Richardson en este caso?

Ejercicio 6.25. Escriba una versión más eficiente del programa del Ejemplo 6.17 basado en la fórmula (6.71). Derive una fórmula equivalente a (6.71) para la fórmula de extrapolación de Richardson que usa la regla de Simpson.

Ejercicio 6.26. Utilizando las expresiones para L_3 y L_4 dadas anteriormente y la subrutina `roots` de MATLAB, calcule los x_j para las fórmulas de cuadratura gaussiana con $n = 3, 4$. Usando los x_j calculados determine usando MATLAB los pesos w_j correspondientes.

Ejercicio 6.27. Usando los resultados obtenidos en el texto para los x_j y w_j en los casos $n = 1, 2$ y los casos $n = 3, 4$ del Ejercicio 6.26, escriba una subrutina en MATLAB con secuencia de llamada `compQG(fname,a,b,n,m)` que aproxime el integral de la función con nombre `fname` en el intervalo $[a, b]$ aplicando una regla de cuadratura de n puntos ($1 \leq n \leq 4$) en cada uno de m subintervalos de $[a, b]$ del mismo largo.

Ejercicio 6.28. Considere el problema

$$I(f) = \int_0^{2h} f(x)\,\mathrm{d}x.$$

a) Construya el polinomio $p_1(x)$ que interpola a $f(x)$ en $x = 0$ y $x = h$.

b) Usando la fórmula de la primera parte, diseñe una fórmula de integración numérica para aproximar $I(f)$.

Ejercicio 6.29. Considere el problema de aproximar integrales de la forma

$$I(f) = \int_0^1 \sqrt{x} f(x)\,\mathrm{d}x.$$

Construya una fórmula para aproximar $I(f)$ de la forma $I_1(f) = w_1 f(x_1)$ que sea exacta para polinomios de grado menor o igual a uno.

Ejercicio 6.30. Considere el problema de resolver la *ecuación integral*

$$z(s) + 4\int_0^1 \cos(st) z^2(t)\,\mathrm{d}t = 4, \quad 0 \leq s \leq 1,$$

para la función desconocida $z(\cdot)$. Usando una regla de cuadratura apropiada (vea el Cápitulo (6)) para discretizar el integral en esta ecuación, obtenga un sistema de ecuaciones nolineales cuya solución es una aproximación de $z(\cdot)$. Resuelva el sistema resultante usando el Método de Newton.

Ejercicio 6.31. Verifique que la fórmula (6.87) se puede escribir también como

$$T_{n,m}(f) = \frac{k}{2}\left[T_n(f(\cdot, y_0)) + T_n(f(\cdot, y_m))\right] + k\sum_{j=1}^{m-1} T_n(f(\cdot, y_j)).$$

Ejercicio 6.32. Obtenga una fórmula similar a (6.87) para aproximar el integral

$$\int_a^b \left[\int_c^d \left[\int_e^f f(x, y, z)\,\mathrm{d}z\right]\,\mathrm{d}y\right]\,\mathrm{d}x.$$

Ejercicio 6.33. Derive una fórmula para aproximar (6.85) basada en la regla de Simpson (6.59). Escriba una función en MATLAB que implemente dicha fórmula.

Ejercicio 6.34. Usando la función `trapz2`, genere unas tablas de aproximaciones para valores de $n = m = 2, 4, 8, \ldots, 256$ para los siguientes integrales:

a) $\displaystyle\int_0^1 \left[\int_0^3 \sqrt{x^2 + y}\,\mathrm{d}y\right]\,\mathrm{d}x.$

b) $\displaystyle\int_1^3 \left[\int_0^x \frac{2}{x^2 + y^2}\,\mathrm{d}y\right]\,\mathrm{d}x.$

Aproxime los errores relativos sucesivos en dichos cálculos usando un estimador similar al del Ejemplo 6.18.

Capítulo 7

Solución Numérica de Ecuaciones Diferenciales Ordinarias

Las ecuaciones diferenciales se utilizan frecuentemente para modelar situaciones físicas en las ciencias naturales, ingeniería, y otras disciplinas, donde hay envueltas razones de cambio de una o varias funciones desconocidas con respecto a una o varias variables independientes. Estos modelos varían desde los más sencillos que envuelven una sola ecuación diferencial para una función desconocida (*ecuación escalar*), hasta otros más complejos que envuelven *sistemas de ecuaciones* diferenciales acopladas para varias funciones desconocidas. Por ejemplo, la ley de enfriamiento de Newton y las leyes mecánicas que rigen el movimiento de los cuerpos, al ponerse en términos matemáticos, dán lugar a ecuaciones diferenciales.

Usualmente una ecuación diferencial viene o está acompañada de una condición adicional que especifica el estado del sistema en un tiempo o posición inicial. Esta condición adicional se llama la *condición inicial* y junto con la ecuación diferencial forman lo que se conoce como el *problema de valor inicial*. Por lo general, la solución exacta de un problema de valor inicial es imposible o difícil de obtener de forma explícita. Por tal razón los métodos numéricos se utilizan para aproximar dichas soluciones.

Comenzaremos el capitulo discutiendo los métodos numéricos para ecuaciones escalares y luego generalizamos estos conceptos al caso de sistemas de ecuaciones. En las últimas dos secciones del capitulo, discutimos dos aplicaciones importantes de los problemas de valor inicial: en los métodos de *tiro al blanco* para la solución de problemas de frontera, y en los *métodos homotópicos* para la solución de sistemas de ecuaciones nolineales.

7.1 Problema de Valor Inicial y el Método de Euler

Considere el *problema de valor inicial* para la función (desconocida) $y(t)$ descrito por:

$$\begin{cases} y'(t) = f(t, y(t)), \quad t_0 < t < b, \\ y(t_0) = y_0, \end{cases} \tag{7.1}$$

donde $f : \mathbb{R}^2 \to \mathbb{R}$ es una función conocida. Defina para $n \geq 1$ el *largo de paso* h y la *partición* (uniforme) (t_j) del intervalo $[t_0, b]$ por[1]:

$$h = \frac{b - t_0}{n}, \quad t_j = t_0 + jh, \quad 0 \leq j \leq n. \tag{7.2}$$

Para cualquier $j \geq 0$, tenemos por el Teorema de Taylor (Teorema 1.2) y usando la ecuación diferencial en (7.1), que podemos escribir:

$$\begin{aligned} y(t_{j+1}) &= y(t_j) + hy'(t_j) + \frac{1}{2}h^2 y''(\xi_j), \\ &= y(t_j) + hf(t_j, y(t_j)) + \frac{1}{2}h^2 y''(\xi_j). \end{aligned} \tag{7.3}$$

Eliminando los términos $O(h^2)$ obtenemos la aproximación:

$$y(t_{j+1}) \approx y(t_j) + hf(t_j, y(t_j)), \quad 0 \leq j \leq n. \tag{7.4}$$

Denotamos ahora por y_j una aproximación cualquiera de $y(t_j)$. Entonces motivado por la aproximación de arriba definimos las aproximaciones (y_j) por la recursión:

$$y_{j+1} = y_j + hf(t_j, y_j), \quad 0 \leq j \leq n, \tag{7.5}$$

lo que se conoce como el *método de Euler*.

La cantidad

$$T_{j+1} = \frac{1}{2}h^2 y''(\xi_j), \tag{7.6}$$

la cual descartamos en la serie de Taylor (7.3) para obtener las aproximaciones (7.4), se llama el *error de truncación o error local* del método de Euler y está íntimamente relacionada con la convergencia del método. De hecho si definimos los *errores absolutos* por $e_j = y(t_j) - y_j$, entonces tomando la diferencia entre la expansión de Taylor (7.3) y la fórmula (7.5) del método de Euler se obtiene que

$$e_{j+1} = e_j + h[f(t_j, y(t_j)) - f(t_j, y_j)] + T_{j+1}. \tag{7.7}$$

[1]En MATLAB la instrucción `linspace(a,b,n+1)` genera una partición uniforme del intervalo `[a,b]` con `n` subintervalos.

Suponiendo ahora que f satisface una *condición de Lipschitz uniforme en* t, i.e.,

$$|f(t,y) - f(t,z)| \leq L|y - z|, \quad \forall\, t, y, z, \tag{7.8}$$

para alguna constante L no–negativa, entonces se puede verificar (Ejercicio 7.12), de la recursión (7.7) para los errores, que:

$$|e_j| \leq \left[\frac{\mathrm{e}^{L(b-t_0)} - 1}{2L}\right] h \max_{[a,b]} |y''(t)|, \quad 0 \leq j \leq n. \tag{7.9}$$

Esto demuestra que el método de Euler tiene un orden de convergencia global $O(h)$.

La programación en MATLAB del método de Euler es relativamente simple. Hacemos ésto mediante una subrutina llamada `feulern` que recibe en la secuencia de llamada el nombre (una cadena de caracteres) de la subrutina que calcula la función f, y los datos t_0, b, y_0, n. Esta subrutina devuelve dos vectores con las (t_j) y las (y_j) aproximadas. Veamos:

```
function [tvals,yvals]=feulern(f,tspan,y0,n)
t0=tspan(1);
b=tspan(2);
h=(b-t0)/n;
yvals=zeros(n+1,1);
tvals=linspace(t0,b,n+1)';
yvals(1)=y0;
for i=2:n+1
    yvals(i)=yvals(i-1)+h*feval(f,tvals(i-1),yvals(i-1));
end
```

Note el uso de la función `feval` de MATLAB para evaluar la función f, la cuál está especificada por el nombre del archivo que contiene las instrucciones para calcular f. Usamos ahora esta subrutina en el siguiente ejemplo.

Ejemplo 7.1. Considere el problema de valor inicial

$$\begin{cases} y'(t) = \dfrac{5y(t)}{t+1} - y(t), \quad 0 < t < 4, \\ \\ y(0) = 1. \end{cases}$$

En este problemas sabemos que la solución exacta es $y(t) = (t+1)^5 \exp(-t)$. La siguiente función en MATLAB evalúa el lado derecho de la ecuación diferencial:

```
function f=etest(t,y);
f=5*y/(t+1)-y;
```

Ahora podemos resolver numericamente éste problema con la función `feulern` de arriba. Tomando $n = 20$, podemos gráficar la solución numérica junto con la exacta para comparar los resultados, usando las siguientes instrucciones en MATLAB:

```
[t,y]=feulern('etest',[0,4],1,20);
yy=(t+1).^5.*exp(-t);
plot(t,y,'kx',t,yy,'k')
xlabel('t');ylabel('y');
legend('Solucion Aproximada','Solucion Exacta')
```

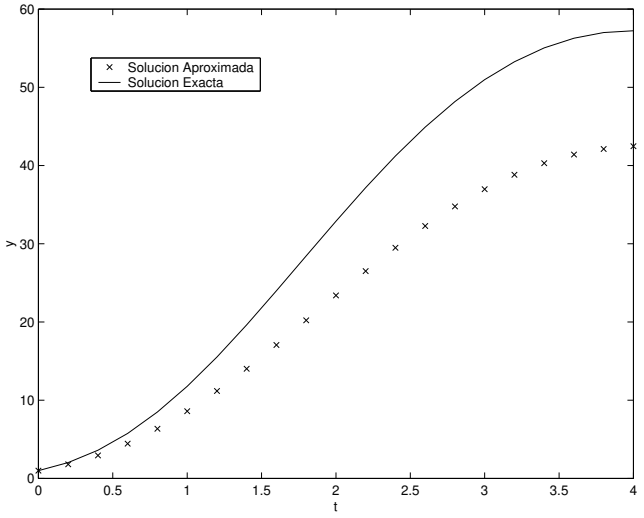


Figura 7.1: Solución exacta vs la calculada por el método de Euler en el Ejemplo 7.1.

Las solución numérica se ilustra en la Figura 7.1 con el símbolo "x". Note que las aproximaciones numéricas no coinciden con la solución exacta y que el error aumenta según aumenta la t. Esto es típico de los métodos numéricos que usan un largo de paso fijo y no contradice el estimado del error (7.9). De hecho según (7.9), el error puede crecer hasta exponencialmente con respecto al largo del intervalo. Para controlar el error, una primera alternativa[2] es disminuir la h, i.e., aumentar

[2]Otra mejor alternativa es variar o ajustar la h en cada paso dependiendo de un estimado del error local en dicho paso. Los métodos de largo de paso variable se discuten en la Sección (7.4).

la n. Para este ejemplo mostramos los resultados para la aproximación en $t = 4$ al disminuir h sucesivamente. Note que $y(4) = 57.2364$ correcto al número de cifras mostradas. Obtuvimos lo siguiente:

n	y_n	$\|y(4) - y_n\|$
20	42.4723	14.7640
40	48.3445	8.89186
80	52.2842	4.95215
160	54.6108	2.62556
320	55.8827	1.35365
640	56.5488	0.687527
1280	56.8899	0.346503
2560	57.0624	0.173945

Vemos aquí que definitivamente la aproximación mejora según aumenta la n pero la convergencia es bastante lenta. De hecho la aproximación numérica tiene apenas un error relativo de 3×10^{-3} para $n = 2560$, es decir $h = 4 \times 10^{-4}$. □

El ejemplo anterior muestra que aunque el método de Euler es convergente según la h tiende a cero, la convergencia del método es lenta ($O(h)$). Esto obliga a tomar una h excesivamente pequeña para obtener un error satisfactorio en las aproximaciones. Por otro lado, al usar un h demasiado pequeño en los cálculos, podemos tener acumulación de errores debido a la aritmética finita, un fenómeno similar al que observamos en la diferenciación numérica. Este problema lo podemos controlar (no eliminar) si utilizamos métodos con un orden de convergencia más alto, como los llamados métodos `Runge--Kutta` que discutiremos más adelante.

7.2 Sistemas de Ecuaciones

El método de Euler se puede generalizar al caso de sistemas de ecuaciones diferenciales. La teoría de convergencia global que discutimos en el caso de una ecuación, aplica palabra por palabra al caso de sistemas. En la subrutina `feulern` descrita antes solo hay que añadir una instrucción para determinar el tamaño del sistema. Hicimos un cambio también para que en lugar de n, la subrutina recibe la h. La subrutina que llamamos ahora `feulerh` queda como sigue:

```
function [tvals,yvals]=feulerh(f,tspan,y0,h)
t0=tspan(1);
b=tspan(2);
n=floor((b-t0)/h)+1;
m=length(y0);
```

```
yvals=zeros(n+1,m);
tvals=linspace(t0,b,n+1)';
yvals(1,:)=y0;
for i=2:n+1
   yvals(i,:)=yvals(i-1,:)+h*feval(f,tvals(i-1),yvals(i-1,:))';
end
```

Esta subrutina puede ser usada exactamente como antes para el caso de una ecuación. El arreglo `tvals` es un vector columna con los valores $t_0, t_1, \ldots, t_n$, mientras que `yvals` es un arreglo donde la i–esima fila contiene el vector que aproxima a

$$y(t_i) = (y_1(t_i), y_2(t_i), \ldots, y_m(t_i)).$$

En forma de tabla, el par `[tvals,yvals]` se vería de la forma siguiente[3]:

`tvals`	`yvals`			
t_0	$y_1(t_0)$	$y_2(t_0)$	$\cdots$	$y_m(t_0)$
t_1	$y_1(t_1)$	$y_2(t_1)$	$\cdots$	$y_m(t_1)$
t_2	$y_1(t_2)$	$y_2(t_2)$	$\cdots$	$y_m(t_2)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_n	$y_1(t_n)$	$y_2(t_n)$	$\cdots$	$y_m(t_n)$

Veamos un ejemplo numérico de un sistema de ecuaciones y como lo resolvemos con `feulerh`.

Ejemplo 7.2. Las siguientes ecuaciones describen un modelo simplicado del corazón, donde $x(t)$ representa el largo de una cierta fibra o músculo del corazón, y $s(t)$ representa un estimulo (eléctrico) aplicado:

$$\begin{cases} x'(t) = \mu(-s(t) - \frac{1}{3}x(t)^3 + px(t)), \\ s'(t) = x(t)/\mu. \end{cases}$$

Aquí μ y p son parámetros del modelo. El lado derecho del sistema lo evaluamos mediante la siguiente subrutina en MATLAB:

```
function f=heart(t,y)
%  y(1) representa x(t) y y(2) representa s(t)
global mheart pheart
f=zeros(2,1);
f(1)=mheart*(-y(2)-y(1)^3/3+pheart*y(1));
f(2)=y(1)/mheart;
```

[3]La tabla que genera la función `feulerh` no contiene los $y_j(t_i)$ exactos, pero si las aproximaciones a éstos.

Note el uso de la instrucción[4] que declara las variables `mheart` y `pheart` como variables globales las cuales son accesibles por cualquier rutina o programa con una instrucción `global` igual. Usamos las condiciones iniciales $x(0) = 0$, $s(0) = -1$ y los valores de $\mu = 0.5$ y $p = 1$. Ahora aproximamos en el intervalo $[0, 10]$ y graficamos la solución con la siguiente secuencia de instrucciones en MATLAB:

```
global mheart pheart
mheart=0.5;
pheart=1;
[t,y]=feulerh('heart',[0,10],[0,-1],0.1);
plot(t,y(:,1),'k-',t,y(:,2),'k.')
xlabel('t');ylabel('y');
legend('x(t)','s(t)')
```

Asi generamos la Figura 7.2. No entraremos en detalles sobre la interpretación

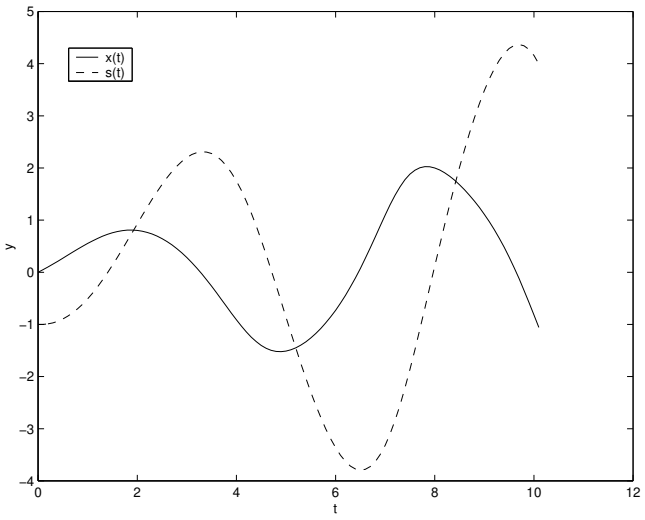


Figura 7.2: Solución por el método de Euler de un modelo del corazón.

física de estas gráficas pero si mencionamos que el método de Euler es efectivo en este problema ya que las soluciones no varían muy rápidamente en el intervalo en cuestión. □

[4]El uso de la instrucción `global` en general no es aconsejable ya que dichas variables podrian interferir con otras de nombres similares en otras subrutinas. Una forma más segura de pasar parámetros a una función es a través de su secuencia de llamada.

Para resolver ecuaciones diferenciales de orden mayor de uno, hacemos primero un cambio de variables para así convertir la ecuación dada a un sistema de primer orden. Luego usamos el método de Euler para sistemas según descrito antes.

Ejemplo 7.3. Considere el problema de valor inicial para la siguiente ecuación de orden dos:

$$\begin{cases} 2x''(t) + 4[x'(t)]^2 - 4x(t) = \cos(x(t)), \\ x(0) = 2, \quad x'(0) = 10. \end{cases}$$

Haciendo la sustitución o cambio de variables $x_1(t) = x(t)$, $x_2(t) = x'(t)$, tenemos que éste problema es equivalente al siguiente sistema de primer orden:

$$\begin{cases} x_1'(t) = x_2(t), \\ x_2'(t) = -2x_2^2(t) + 2x_1(t) + \frac{1}{2}\cos(x_1(t)), \\ x_1(0) = 2, \quad x_2(0) = 10. \end{cases}$$

Este sistema puede ser resuelto ahora en forma similar al que vimos en el Ejemplo 7.2. Primero escribimos la subrutina que evalúa el lado derecho del sistema:

```
function f=sist7_3(t,x)
f=zeros(2,1);
f(1)=x(2);
f(2)=-2*x(2)^2+2*x(1)+0.5*cos(x(1));
```

Ahora aproximamos y trazamos (vea la Figura 7.3) la gráfica de $x(t) = x_1(t)$ en el intervalo $[0, 1]$ usando las instrucciones:

```
[t,x]=feulerh('sist7_3',[0,1],[2,10],0.01);
plot(t,x(:,1),'k')
xlabel('t');ylabel('x(t)');
```

□

7.3 Métodos Runge–Kutta

En esta sección vamos a estudiar una familia de métodos con un orden de convergencia mayor que el del método de Euler. Un análisis más profundo del método de Euler muestra que según el método progresa, éste va generando aproximaciones a lo largo de la tangente de una cierta curva que está *cerca* a la curva desconocida o buscada. Los métodos Runge–Kutta extienden esta idea geométrica al utilizar varias derivadas o tangentes intermedias, en lugar de solo una, para así obtener una mejor aproximación de la función desconocida. Los métodos Runge–Kutta más simples se obtienen usando dos de éstas derivadas intermedias.

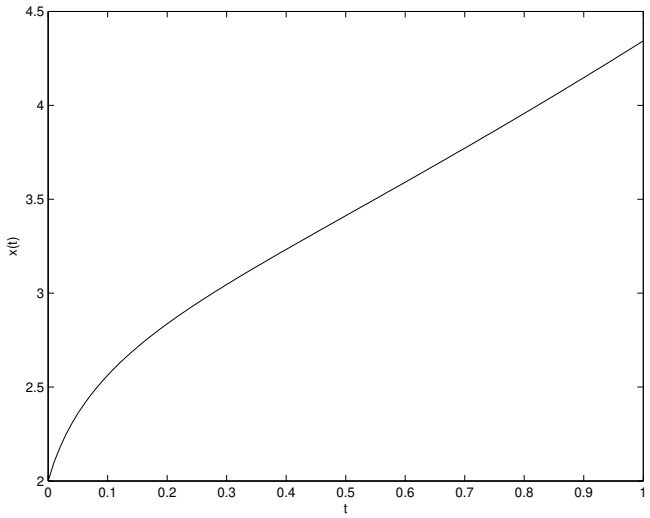


Figura 7.3: Solución numérica por el método de Euler para el problema de valor inicial del Ejemplo 7.3.

7.3.1 Métodos Runge–Kutta de dos Evaluaciones

Siguiendo la idea de las derivadas intemedias que mencionamos antes, busacamos métodos o fórmulas numéricas de la forma:

$$y_{j+1} = y_j + h[\gamma_1 f(t_j, y_j) + \gamma_2 f(t_j + \alpha h, y_j + \beta h f(t_j, y_j))], \quad j \geq 0. \tag{7.10}$$

Note que aunque en la fórmula se perciben tres f, el método envuelve solo dos evaluaciones ya que dos de estas f tienen los mismos argumentos. La idea ahora es determinar los parámetros $\alpha, \beta, \gamma_1, \gamma_2$ de modo que el método tenga orden de convergencia lo más alto posible. Un análisis del error local de ésta fórmula basado en el Teorema de Taylor muestra ([2], [6]) que el orden más alto que puede tener ésta fórmula es dos y que ésto puede ocurrir si y solo si:

$$\gamma_1 + \gamma_2 = 1, \quad \alpha = \beta = \frac{1}{2\gamma_2}, \quad \gamma_2 \neq 0. \tag{7.11}$$

Es decir si $\alpha, \beta, \gamma_1, \gamma_2$ cumplen con éstas condiciones, entonces $e_j = y(t_j) - y_j = O(h^2)$ para toda j. Los siguientes dos casos especiales de éstas fórmulas, son importantes:

Método de Heun: Aquí se toma $\gamma_2 = 1/2$ de modo que el método reduce a:

$$y_{j+1} = y_j + \frac{h}{2}[f(t_j, y_j) + f(t_j + h, y_j + h f(t_j, y_j))], \quad j \geq 0. \tag{7.12}$$

Para propósito de hacer cálculos es mejor escribir ésta fórmula como:

$$\begin{cases} y^*_{j+1} = y_j + hf(t_j, y_j), \\ y_{j+1} = y_j + \frac{h}{2}[f(t_j, y_j) + f(t_j + h, y^*_{j+1})], \quad j \geq 0. \end{cases} \tag{7.13}$$

Método del Punto Medio: Aquí se toma $\gamma_2 = 1$ de modo que el método reduce a:

$$y_{j+1} = y_j + hf\left(t_j + \frac{h}{2}, y_j + \frac{h}{2}f(t_j, y_j)\right), \quad j \geq 0, \tag{7.14}$$

el cuál para propósito de hacer cálculos, es mejor escribirlo como:

$$\begin{cases} y^*_{j+1} = y_j + \frac{h}{2}f(t_j, y_j), \\ y_{j+1} = y_j + hf\left(t_j + \frac{h}{2}, y^*_{j+1}\right), \quad j \geq 0. \end{cases} \tag{7.15}$$

Veamos ahora una implementación en MATLAB del método de Heun. Note que en el ciclo `for` hay exactamente dos evaluaciones de f:

```
function [tvals,yvals]=heun(f,tspan,y0,h)
t0=tspan(1);
b=tspan(2);
n=floor((b-t0)/h)+1;
m=length(y0);
yvals=zeros(n+1,m);
tvals=linspace(t0,b,n+1)';
yvals(1,:)=y0;
h2=h/2;
for i=2:n+1
   k1=feval(f,tvals(i-1), yvals(i-1,:))';
   yvals(i,:)=yvals(i-1,:)+h*k1;
   yvals(i,:)=yvals(i-1,:)+h2*(k1+feval(f,tvals(i),yvals(i,:)');
end
```

Ejemplo 7.4. Consideramos ahora las ecuaciones diferenciales que se obtienen de las leyes de Newton aplicadas al problema de dos cuerpos. Suponemos que uno de los cuerpos es mucho más masivo que el otro de modo que su movimiento es descartable, e.g., la tierra y un satélite. Suponemos también que el movimiento es en un plano. Como la fuerza gravitacional es inversamente proporcional a la distancia entre los cuerpos, tenemos tomando todas las constantes envueltas como uno, que la posición $(x(t), y(t))$ del cuerpo pequeño está dada por el sistema de

ecuaciones diferenciales:

$$\begin{cases} x''(t) = -\dfrac{x(t)}{(x(t)^2 + y(t)^2)^{3/2}}, \\ \\ y''(t) = -\dfrac{y(t)}{(x(t)^2 + y(t)^2)^{3/2}}. \end{cases}$$

Tomamos como condiciones iniciales $x(0) = 0.4, x'(0) = 0, y(0) = 0.1, y'(0) = 2$. Este sistema de orden dos lo podemos convertir a uno de primer orden usando la sustitución:

$$u_1(t) = x(t), \quad u_2(t) = x'(t) \quad , \quad u_3(t) = y(t), \quad u_4(t) = y'(t).$$

El sistema resultante de orden uno es:

$$\begin{cases} u_1'(t) = u_2(t), \\ u_2'(t) = -u_1(t)/(u_1(t)^2 + u_3(t)^2)^{3/2}, \\ u_3'(t) = u_4(t), \\ u_4'(t) = -u_3(t)/(u_1(t)^2 + u_3(t)^2)^{3/2}, \end{cases}$$

donde $u_1(0) = 0.4, u_2(0) = 0, u_3(0) = 0.1, u_4(0) = 2$. Definimos ahora la siguiente subrutina en MATLAB que evalúa el lado derecho de éste sistema:

```
function f=satelite(t,u)
f=zeros(4,1);
denom=(u(1)^2+u(3)^2)^1.5;
f(1)=u(2);
f(2)=-u(1)/denom;
f(3)=u(4);
f(4)=-u(3)/denom;
```

Ahora calculamos y trazamos (ver Figura 7.4) la solución del problema de valor inicial con la siguiente secuencia de instrucciones en MATLAB. Note que trazamos el conjunto de puntos $(x(t), y(t))$, que representa la trayectoria o orbita del cuerpo más peqeño, para los valores de t generados en lugar de $(t, x(t))$ y $(t, y(t))$. Tenemos entonces:

```
[t,y]=heun('satelite',[0,10],[0.4,0,0.1,2],0.01);
plot(y(:,1),y(:,3),'k')
xlabel('x');
ylabel('y');
```

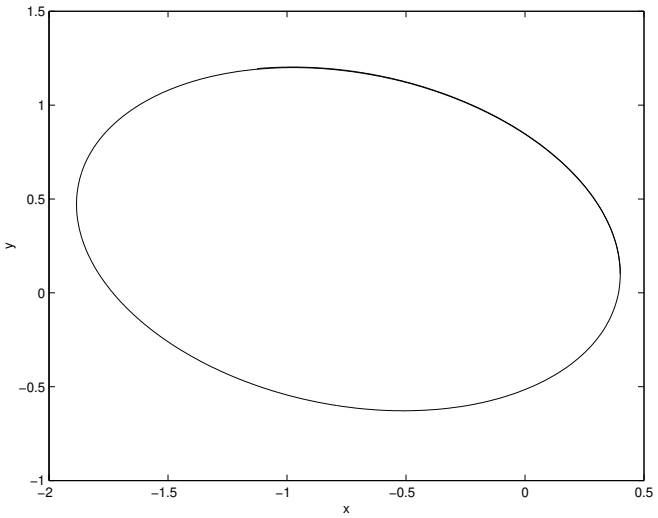


Figura 7.4: Solución particular del problema de dos cuerpos por el método de Heun.

Note que la curva luce efectivamente como una elipse, aunque ésto se acentúa en la gráfica porque los ejes tienen unidades de largo distintas. □

En el siguiente ejemplo vamos a ilustrar como se puede aproximar la solución de un problema de valor inicial en el caso que los coeficientes de dicha ecuación se conocen solo en un numero finito de puntos.

Ejemplo 7.5. Considere el siguiente problema de valor inicial:

$$\begin{cases} y'(t) + f(t)y(t) = g(t), & 0 < t < 5, \\ y(0) = 1, \end{cases}$$

donde solo tenemos la siguiente información de las funciones f, g:

t	0	1	2	3	4	5
$f(t)$	-3	-3	-1	3	9	17

t	0	0.7143	1.4286	2.1429	2.8571	3.5714	4.2857	5
$g(t)$	-0.7422	1.3434	2.7722	2.8458	1.5281	-0.5366	-2.3390	-2.9979

El problema de valor inicial lo resolvemos ahora interpolando las funciones f, g usando splines cúbicos. En lugar de la subrutina `heun` descrita en esta sección, utilizamos la función `ode45` de MATLAB la cual se describe en la próxima sección. Primero definimos los vectores con los datos de f, g:

```
tf=[0,1,2,3,4,5];
f_data=[-3,-3,-1,3,9,17];
tg=[0,0.7143,1.4286,2.1429,2.8571,3.5714,4.2857,5];
g_data=[-0.7422,1.3434,2.7722,2.8458,1.5281,-0.5366,-2.3390,-2.9979];
```

Luego calculamos los splines cúbicos para estos datos:

```
f=spline(tf,f_data);
g=spline(tg,g_data);
```

Ahora efectuamos la llamada a la subrutina `ode45` para resolver el problema de valor inicial. Note la forma de pasar a `ode45` la información de los splines de f, g:

```
lado_der=@(t,y) RHS(t,y,f,g);
[t,y]=ode45('lado_der',[0,5],1);
plot(t,y)
xlabel('t');ylabel('y');
```

La función `RHS` está dada por:

```
function yp=RHS(t,y,f,g)
ft=ppval(f,t);
gt=ppval(g,t);
yp=-ft*y+gt;
```

En la Figura 7.5 se muestra la aproximación de $y(t)$ obtenida. □

7.3.2 Métodos Runge–Kutta de más de dos Evaluaciones

Aunque los métodos de Heun y del punto medio generan soluciones con un grado de presición usualmente razonable, nos interesa estudiar métodos de orden aún más alto, que no necesiten una h muy pequeña para un buen grado de aproximación. Vimos que un método Runge–Kutta de dos evaluaciones intermedias genera un método de orden dos. Es razonable pensar que con tres o cuatro evaluaciones intermedias podemos obtener métodos Runge–Kutta de ordenes tres y cuatro respectivamente, lo cuál es el caso. No obstante, para métodos Runge–Kutta de más de cuatro evaluaciones, el orden de convergencia no aumenta linealmente con el número de evaluaciones.

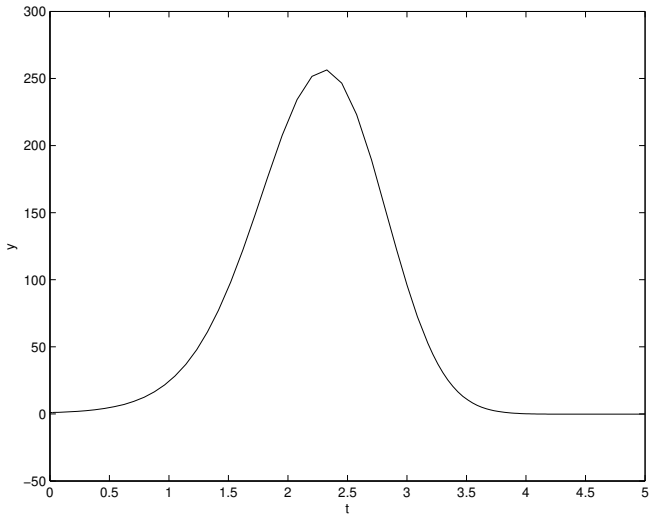


Figura 7.5: Solución calculada del problema de valor inicial del Ejemplo 7.5 con datos de coeficientes incompletos.

Un ejemplo de un método Runge–Kutta de orden cuatro de cuatro evaluaciones es el llamado *método Runge–Kutta clásico* definido por las fórmulas:

$$\begin{cases} v_1 = f(t_j, y_j), \quad v_2 = f\left(t_j + \frac{h}{2}, y_j + \frac{h}{2}v_1\right), \\ v_3 = f\left(t_j + \frac{h}{2}, y_j + \frac{h}{2}v_2\right), \quad v_4 = f(t_j + h, y_j + hv_3), \\ y_{j+1} = y_j + \frac{h}{6}[v_1 + 2v_2 + 2v_3 + v_4], \quad j \geq 0. \end{cases} \tag{7.16}$$

Esta fórmulas se derivan de forma similar a como trabajamos con la ecuación (7.10). Esto es, se plantea una ecuación similar a (7.10), pero con cuatro evaluaciones de f y con más parámetros. Luego se buscan condiciones en los parámetros que resulten en un método de grado lo más alto posible. El método (7.16) se obtiene como un caso especial de éstas condiciones. Este procedimiento es tedioso pero a la vez un tanto mecánico.

El programado estándar de MATLAB incluye dos subrutinas intrínsecas para la solución de problemas de valor inicial: `ode23` y `ode45`. `ode23` utiliza una combinación de un método Runge–Kutta de orden dos con otro de orden tres. La combinación de éstos métodos permite que se pueda calcular un estimado del error en la aproximación numérica en cada paso, de modo que la subrutina puede ajustar el largo de paso h en forma dinámica para mantener el error global menor

de una tolerancia especificada por el usuario. (Vea la Sección (7.4)). Estos métodos se llaman *adaptativos* o de *largo de paso variable*. La subrutina `ode45` es similar a `ode23` pero utiliza una combinación de métodos de ordenes cuatro y cinco, de cinco y seis evaluaciones respectivamente. La secuencia de llamada de ambas rutinas es idéntica y solo requiere del nombre de la función f (en forma de una cadena de caracteres), los tiempos inicial y final, la condición inicial, y otros parámetros que son opcionales. Por ejemplo un cómputo similar al del Ejemplo 7.4, lo podemos realizar reemplazando la llamada a `heun` con:

```
[t,y]=ode23('satelite',[0,10],[0.4,0,0.1,2]);
```

Note que no es necesario especificar la h ya que la subrutina `ode23` es adaptativa y ajusta la h según progesa el cómputo para así lograr una cierta precisión predeterminada.

7.4 Predicción y Control del Error - Métodos de un Paso

El método de Euler y los de Runge–Kutta son casos especiales de los llamados *métodos de un paso*. Un método de un paso es uno que produce una aproximación en el paso $j+1$ utilizando únicamente información del paso j. La forma general de un método de un paso está dada por:

$$y_{j+1} = y_j + hF(t_j, y_j, h), \quad j \geq 0. \tag{7.17}$$

Por ejemplo

- el método de Euler se obtiene si tomamos $F(t, y, h) = f(t, y)$ en (7.17);
- el método de Heun se obtiene tomando

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

El *error de truncación o error local en el paso* j del método (7.17) se define por

$$T_{j+1} = y(t_{j+1}) - y(t_j) - hF(t_j, y(t_j), h), \quad j \geq 0. \tag{7.18}$$

Note que T_{j+1} es la cantidad que hay que sumarle al método numérico para obtener el valor exacto en t_{j+1} partiendo de datos exactos en t_j. Para el método de Euler la ecuación (7.6) nos dice que $T_{j+1} = O(h^2)$ para éste método, mientras que para

los métodos Runge–Kutta de dos evaluaciones se obtiene que $T_{j+1} = O(h^3)$. La cantidad

$$\frac{T_{j+1}}{h}, \tag{7.19}$$

se conoce como el *error local por unidad de paso*.

El error que en realidad nos interesa estimar o controlar es el *error absoluto o global en el paso* j el cual se define por

$$e_j = y(t_j) - y_j, \quad j \geq 0. \tag{7.20}$$

Se puede demostrar ([8], [10]) que si el error local por unidad de paso satisface

$$\left| \frac{T_{j+1}}{t_{j+1} - t_j} \right| \leq \epsilon, \quad j \geq 0, \tag{7.21}$$

donde $\epsilon > 0$ es un número dado, entonces existe una constante C tal que

$$|e_j| \leq C\epsilon, \quad j \geq 0, \tag{7.22}$$

i.e., se logra un control cualitatívo del error global (7.20). Note que en (7.21) se utiliza una subdivisión no uniforme del intervalo $[t_0, b]$, i.e., la h varía de paso a paso y se ajusta para poder cumplir con (7.21).

Nos concentramos ahora en que la desigualdad (7.21) se cumpla o satisfaga en cada paso. Para ésto necesitamos un método efectivo para estimar el error de truncación T_{j+1}. El procedimiento que describimos a continuación está basado en una combinación de dos métodos de un paso cuyos errores locales tienen ordenes que difieren solo por uno. Esto es, suponemos que tenemos dos métodos

$$\begin{aligned} y_{j+1} &= y_j + hF_1(t_j, y_j, h), \quad T^1_{j+1} = O(h^{p+1}), \\ \hat{y}_{j+1} &= y_j + hF_2(t_j, y_j, h), \quad T^2_{j+1} = O(h^{p+2}). \end{aligned} \tag{7.23}$$

Se puede demostrar ahora ([8], [10]) que

$$T^1_{j+1} = \hat{y}_{j+1} - y_{j+1} + O(h^{p+2}), \tag{7.24}$$

lo que nos lleva al estimador

$$T^1_{j+1} \approx \hat{y}_{j+1} - y_{j+1}. \tag{7.25}$$

Para que el estimador (7.25) sea uno práctico desde el punto de vista computacional, debemos tener que $F_2 = F_1 +$ corrección de modo que el calculo de $\hat{y}_{j+1}$ no sea mucho más costoso que el de y_{j+1}, usualmente una evaluación mas de f.

k	a_k	c_k	d_k	b_{k0}	b_{k1}	b_{k2}	b_{k3}	b_{k4}
0	—	$\frac{25}{216}$	$\frac{16}{135}$	—	—	—	—	—
1	$\frac{1}{4}$	0	0	$\frac{1}{4}$	—	—	—	—
2	$\frac{3}{8}$	$\frac{1408}{2565}$	$\frac{6656}{12825}$	$\frac{3}{32}$	$\frac{9}{32}$	—	—	—
3	$\frac{12}{13}$	$\frac{2197}{4104}$	$\frac{28561}{56430}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$	—	—
4	1	$-\frac{1}{5}$	$-\frac{9}{50}$	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	—
5	$\frac{1}{2}$	—	$\frac{2}{55}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$

Tabla 7.1: Coeficientes del Método Runge–Kutta–Fehlberg.

Las fórmulas más conocidas para lograr ésto son las de *Runge–Kutta–Fehlberg* en donde

$$F_1(t,y,h) = \sum_{k=0}^{4} c_k v_k, \quad F_2(t,y,h) = \sum_{k=0}^{5} d_k v_k, \tag{7.26}$$

donde

$$\begin{cases} v_0 &= f(t_j, y_j), \\ v_k &= f\left(t_j + a_k h, y_j + h\sum_{m=0}^{k-1} b_{km} v_m\right), \quad 1 \le k \le 5, \end{cases} \tag{7.27}$$

y los coeficientes están dados en la Tabla 7.1. Estas ideas las recojemos en el siguiente pseudo algoritmo para predecir y controlar el error local por unidad de paso:

Algoritmo 7.6. Método para la predicción y control del error local por unidad de paso:

1. Calcule $y_{j+1}, \hat{y}_{j+1}$ mediante las fórmulas (7.26), (7.27) y la Tabla 7.1.
2. Si $|(\hat{y}_{j+1} - y_{j+1})/(t_{j+1} - t_j)| < \epsilon$, entonces aceptamos $\hat{y}_{j+1}$ y proseguimos. De lo contrario se reduce $h = t_{j+1} - t_j$ y se repite el paso anterior.

Note que en el segundo paso aceptamos $\hat{y}_{j+1}$ en lugar de y_{j+1} ya que ésta representa un valor más preciso. La función `ode45` de MATLAB emplea las fórmulas (7.26), (7.27), la Tabla 7.1, y una versión más sofisticada del pseudo algoritmo descrito antes.

7.5 Métodos Multipaso

Los métodos de un paso descritos hasta ahora tienen la desventaja de que requieren muchas evaluaciones por iteración de la función f que aparece en (7.1). Por otro

lado son bien eficientes y prácticos para la predicción y control del error según descrito en la sección anterior. Los *métodos multipaso* utilizan la información de varios de los pasos anteriores para predecir en t_{j+1}. De esta forma se pueden obtener métodos con orden de convergencia arbitrario y que requieren en general una evaluación de la función f por paso (excepto por la etapa inicial). Por otro lado estos métodos son un tanto rígidos para la implementación de estrategias para el control del error global.

Las familias de métodos multipaso más conocidas son las de *Adams–Bashforth* y *Adams–Moulton*. Ambas familias de métodos se derivan a partir de la siguiente formulación del problema (7.1). Si integramos la ecuación diferencial en (7.1) en el intervalo $[t_j, t_{j+1}]$ obtenemos la *ecuación integral*

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(t, y(t))\, dt. \tag{7.28}$$

Los métodos de Adams–Bashforth y Adams–Moulton se obtienen aproximando la integral de arriba con polinomios que interpolan a $f(t, y(t))$ en una de las siguientes formas:

- en los métodos Adams–Bashforth de orden $p+1$ de $p+1$ pasos, interpolamos a $f(t, y(t))$ en los puntos $\{t_j, t_{j-1}, t_{j-2}, \ldots, t_{j-p}\}$;
- en los métodos Adams–Moulton de orden $p+1$ de p pasos, interpolamos a $f(t, y(t))$ en los puntos $\{t_{j+1}, t_j, t_{j-1}, \ldots, t_{j-(p-1)}\}$.

Vamos a discutir para ambas familias de métodos el caso particular de interpolación lineal.

7.5.1 Método Adams–Bashforth de Orden Dos

Para fijar ideas, considere el problema de aproximar

$$\int_{t_j}^{t_{j+1}} g(t)\, dt, \tag{7.29}$$

para una cierta función $g(t)$ usando un polinomio lineal que interpola a $g(t)$ en $\{t_j, t_{j-1}\}$. Este polinomio está dado por

$$p_1(t) = \frac{1}{h}[g(t_{j-1})(t_j - t) + g(t_j)(t - t_{j-1})], \tag{7.30}$$

donde $h = t_j - t_{j-1}$. Sabemos por el Teorema 5.5 y el Ejercicio 5.8 que

$$g(t) = p_1(t) + (t - t_{j-1})(t - t_j) g[t_{j-1}, t_j, t]. \tag{7.31}$$

Sustituyendo este resultado en (7.29) y usando que

$$\int_{t_j}^{t_{j+1}} p_1(t)\,dt = \frac{h}{2}[3g(t_j) - g(t_{j-1})], \tag{7.32}$$

tenemos que

$$\int_{t_j}^{t_{j+1}} g(t)\,dt = \frac{h}{2}[3g(t_j) - g(t_{j-1})] + \int_{t_j}^{t_{j+1}} (t-t_{j-1})(t-t_j)g[t_{j-1},t_j,t]\,dt. \tag{7.33}$$

Usando el Teorema A.2 y la propiedad (5.16) podemos escribir que

$$\begin{aligned}
&\int_{t_j}^{t_{j+1}} (t-t_{j-1})(t-t_j)g[t_{j-1},t_j,t]\,dt \\
&\quad = g[t_{j-1},t_j,c_j]\int_{t_j}^{t_{j+1}} (t-t_{j-1})(t-t_j)\,dt, \\
&\quad = \frac{1}{2}g''(\xi_j)\int_{t_j}^{t_{j+1}} (t-t_{j-1})(t-t_j)\,dt,
\end{aligned} \tag{7.34}$$

donde c_j está en $[t_j, t_{j+1}]$ y ξ_j está entre t_{j-1}, t_j, c_j. Con $u = t - t_{j-1}$ tenemos que

$$\int_{t_j}^{t_{j+1}} (t-t_{j-1})(t-t_j)\,dt = \int_h^{2h} u(u-h)\,du = \frac{5}{6}h^3. \tag{7.35}$$

Combinando (7.33), (7.34), y (7.35) obtenemos que

$$\int_{t_j}^{t_{j+1}} g(t)\,dt = \frac{h}{2}[3g(t_j) - g(t_{j-1})] + \frac{5}{12}g''(\xi_j)h^3. \tag{7.36}$$

Usando esta fórmula en (7.28) podemos escribir que

$$y(t_{j+1}) = y(t_j) + \frac{h}{2}[3f(t_j,y(t_j)) - f(t_{j-1},y(t_{j-1}))] + \frac{5}{12}y'''(\xi_j)h^3. \tag{7.37}$$

Si eliminamos el termino $O(h^3)$ de esta expresión, obtenemos la recursión que define el *método de Adams–Bashforth de orden dos*:

$$y_{j+1} = y_j + \frac{h}{2}[3f(t_j,y_j) - f(t_{j-1},y_{j-1})], \quad j \geq 1, \tag{7.38}$$

con error local dado por

$$T_{j+1} = \frac{5}{12}y'''(\xi_j)h^3. \tag{7.39}$$

El método (7.38) es un ejemplo de un método de dos pasos ya que usa información en t_{j-1} y t_j para la aproximación en t_{j+1}. Note que para iniciar las iteraciones se necesitan dos valores y_0 (que es dado) y y_1. Para calcular y_1 podemos proceder de dos formas:

- podemos usar el método de Euler para escribir $y_1 = y_0 + hf(t_0, y_0)$. Como ésto corresponde a un solo paso del método de Euler y el error local de este método es $O(h^2)$, entonces $y(t_1) - y_1 = O(h^2)$ si $y_0 = y(t_0)$.

- Podemos usar cualquier método Runge–Kutta de orden dos para generar y_1.

No importa como generemos y_0, y_1, lo importante es que $y(t_0) - y_0$, $y(t_1) - y_1$ sean ambos $O(h^2)$. De esta forma logramos un error global $O(h^2)$ en el método (7.38).

Es importante observar que aunque en la fórmula (7.38) aparecen dos f, luego de la etapa inicial descrita en el párrafo anterior, el método conlleva una sola evaluación por paso ya que la evaluación en t_{j-1} se arrastra del paso anterior. Esto es típico de los métodos multipaso y contrasta con los métodos Runge–Kutta que requieren varias evaluaciones de f por paso, e.g., (7.10), (7.16), (7.27).

El proceso que usamos aquí para generar la fórmula del método de Adams–Bashforth de dos evaluaciones generaliza en forma natural para construir métodos de $p + 1$ pasos con orden $p + 1$. (Ver [26]).

7.5.2 Método Adams–Moulton de Orden Dos

Vamos ahora a aproximar (7.29) con un polinomio que interpola a $g(t)$ en $\{t_j, t_{j+1}\}$. Este polinomio está dado por:

$$p_1(t) = \frac{1}{h}[g(t_j)(t_{j+1} - t) + g(t_{j+1})(t - t_j)]. \tag{7.40}$$

Imitando el proceso para el método Adams–Bashforth de orden dos ó observando que la aproximación de (7.29) basada en (7.40) corresponde a la regla del Trapezoide para aproximar integrales discutida en el Capítulo (6), tenemos que

$$\int_{t_j}^{t_{j+1}} g(t)\,dt = \frac{h}{2}[g(t_j) + g(t_{j+1})] - \frac{1}{12}g''(\xi_j)h^3. \tag{7.41}$$

Usando esta fórmula en (7.28) obtenemos que

$$y(t_{j+1}) = y(t_j) + \frac{h}{2}[f(t_j, y(t_j)) + f(t_{j+1}, y(t_{j+1}))] - \frac{1}{12}y'''(\xi_j)h^3, \tag{7.42}$$

de donde obtenemos la fórmula para el método de *Adams–Moulton de orden dos*

$$y_{j+1} = y_j + \frac{h}{2}[f(t_j, y_j) + f(t_{j+1}, y_{j+1})], \quad j \geq 0. \tag{7.43}$$

A este método también se le conoce como el *método del Trapezoide*. Note que (7.43) es un método de un paso ya que usa información en t_j para predecir en

t_{j+1}. Los métodos Adams–Moulton de orden mayor usan más información de pasos anteriores.

Es importante observar que la fórmula (7.43) envuelve y_{j+1} en ambos lados de la ecuación. Esto es, si definimos

$$r(z) \equiv y_j + \frac{h}{2}[f(t_j, y_j) + f(t_{j+1}, z)], \tag{7.44}$$

entonces y_{j+1} en (7.43) es una solución de la ecuación nolineal $z = r(z)$. Los métodos para resolver (7.1) con esta característica se llaman *implícitos*. Todos los métodos Adams–Moulton son implícitos mientras que el de Euler, los de Runge–Kutta, y los Adams–Bashforth son *explícitos*. Para resolver $z = r(z)$ podemos emplear cualquiera de los métodos discutidos en el Capítulo (4). Pero usualmente lo que se usa es una iteración de punto fijo:

$$\begin{cases} y_{j+1}^{(i+1)} = r(y_{j+1}^{(i)}), \quad i \geq 0, \\ y_{j+1}^{(0)} \ \text{ dado.} \end{cases} \tag{7.45}$$

Se puede demostrar ([11]) que si h es suficientemente pequeña en (7.44), entonces las iteraciones (7.45) convergen a y_{j+1}. No obstante, en la practica lo que se hace es que el $y_{j+1}^{(0)}$ se genera con un método explícito de orden a lo mínimo uno menos que el implícito, y en (7.45) se hace una sola iteración, ya que se puede demostrar que las iteraciones adicionales no mejoran el orden de la aproximación. Esta combinación de un método explícito con otro implícito se conoce como un *método predictor–corrector*.

Ejemplo 7.7. Si para el método (7.43) usamos el método de Euler como predictor, obtenemos el método predictor corrector que coincide con las fórmulas (7.13). Si por el contrario usamos como predictor el método Adams–Bashforth (7.38) obtenemos el método predictor corrector definido por las fórmulas

$$\begin{cases} y_{j+1}^{(0)} &= y_j + \frac{h}{2}[3f(t_j, y_j) - f(t_{j-1}, y_{j-1})], \\ y_{j+1} &= y_j + \frac{h}{2}[f(t_j, y_j) + f(t_{j+1}, y_{j+1}^{(0)})], \quad j \geq 1. \end{cases} \tag{7.46}$$

□

Los métodos predictor–corrector envuelven dos evaluaciones de f por paso: una para el predictor y otra para el corrector. En adición, estos métodos poseen unas propiedades de *estabilidad* bien favorables por lo que son frecuentemente usados. Para más detalles sobre los métodos Adams–Bashforth y Adams–Moulton, la programación efectiva de estos métodos, y la predicción y control del error para ellos, consulte [26].

7.6 Problemas de Frontera

Un *problema de frontera (de segundo orden) de tipo Dirichlet* para una función $y(t)$ en un intervalo $[a, b]$ está dado por:

$$\begin{cases} y''(t) = f(t, y(t), y'(t)), \quad a < t < b, \\ y(a) = \alpha, \quad y(b) = \beta, \end{cases} \tag{7.47}$$

donde α, β son números dados. Una técnica común para resolver este tipo de problema es el *método de tiro al blanco*. Para resultados sobre la existencia y unicidad de soluciones del problema (7.47) y las propiedades de convergencia del método de tiro al blanco se puede consultar [10], [12].

Veamos en que consiste el método de tiro al blanco. Sea $y(t, \gamma)$ la solución del problema de valor inicial:

$$\begin{cases} y''(t) = f(t, y(t), y'(t)), \quad a < t < b, \\ y(a) = \alpha, \quad y'(a) = \gamma. \end{cases} \tag{7.48}$$

La idea del método de tiro al blanco es hallar un valor de γ tal que $y(b, \gamma) = \beta$. Esto es, queremos hallar una raíz γ^* de la ecuación $y(b, \gamma) - \beta = 0$. Si definimos la función:

$$g(\gamma) \equiv y(b, \gamma) - \beta, \tag{7.49}$$

entonces podemos aproximar una raíz γ^* de g con el método de la bisección, o el de la secante, o el de Newton. Note que cada evaluación de la función g requiere de la solución del problema de valor inicial (7.48). Este problema de valor inicial intermedio se puede resolver numéricamente ya sea con el método de Euler, Runge–Kutta, o un método multipaso.

Ejemplo 7.8. Considere el problema de frontera

$$\begin{cases} y''(t) = \frac{1}{8}(32 + 2t^3 - y(t)y'(t)), \quad 1 < t < 3, \\ \\ y(1) = 17, \quad y(3) = \frac{43}{3}, \end{cases} \tag{7.50}$$

cuya solución exacta es $y(t) = t^2 + 16/t$. Sea $y(t, \gamma)$ la solución del problema de valor inicial

$$\begin{cases} y''(t) = \frac{1}{8}(32 + 2t^3 - y(t)y'(t)), \quad 1 < t < 3, \\ \\ y(1) = 17, \quad y'(1) = \gamma. \end{cases} \tag{7.51}$$

Queremos buscar una raíz de la ecuación escalar

$$g(\gamma) \equiv y(3, \gamma) - \frac{43}{3} = 0.$$

Para calcular $g(\gamma)$ convertimos (7.51) a un sistema de orden uno. Con la sustitución $y_1(t) = y(t)$, $y_2(t) = y'(t)$ tenemos que

$$\begin{cases} y_1'(t) = y_2(t), \\ \\ y_2'(t) = \frac{1}{8}(32 + 2t^3 - y_1(t)y_2(t)), \end{cases} \tag{7.52}$$

donde $y_1(1) = 17$, $y_2(1) = \gamma$. El lado derecho de este sistema lo evaluamos con la siguiente función en MATLAB:

```
function yp=sistema(t,y);
yp=zeros(2,1);
yp(1)=y(2);
yp(2)=(32+2*t^3-y(1)*y(2))/8;
```

Ahora la función $g(\gamma)$ se evalúa con:

```
function g=gfunc(gam);
y0=zeros(2,1);
t0=1.0;
tfinal=3.0;
y0(1)=17.0;
y0(2)=gam;
[t,y] = ode45('sistema',[t0,tfinal],y0);
[m,n]=size(y);
g=y(m,1)-43/3;
```

Finalmente usamos el código para el método de la bisección que presentamos en la Sección 4.1 para resolver la ecuación $g(\gamma) = 0$, y luego generamos y graficamos la solución aproximada del problema de frontera:

```
tm=mbisect('gfunc',-15,-13,1.0e-8,50);
t0=1.0;
tfinal=3.0;
y0(1)=17.0;
y0(2)=tm;
[t,y] = ode45('sistema',[t0,tfinal],y0);
x=1:.1:3;
yexacta=x.^2+16./x;
plot(t,y(:,1),'k',x,yexacta,'k+')
xlabel('t');ylabel('y')
legend('Solucion numerica','Solucion exacta')
```

La raíz calculada de g fue de -13.9998 aproximadamente. La gráfica de la solución calculada por el programa se muestra en la Figura 7.6. □

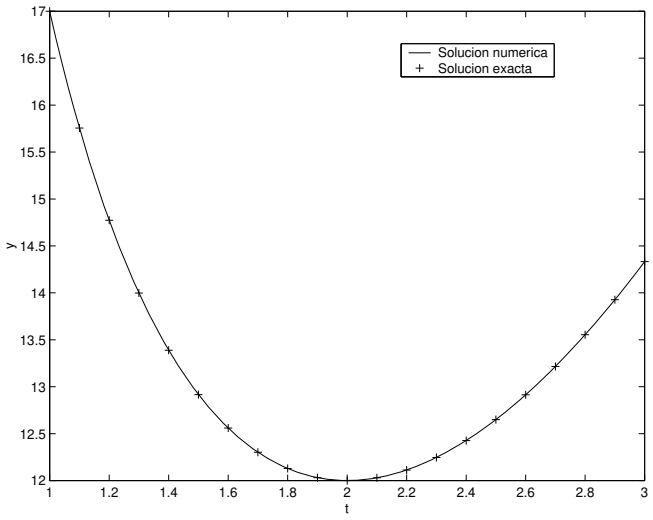


Figura 7.6: Trazado de la solución calculada por el método de tiro al blanco en el Ejemplo 7.8.

7.7 Métodos Homotópicos para la Solución de Ecuaciones Nolineales

En esta sección discutiremos el uso de *métodos homotópicos* para la solución de ecuaciones polinomiales. Estos métodos se pueden generalizar a sistemas polinomiales y a sistemas nolineales en general.

Los métodos iterativos como el de Newton o el de la secante, tienen dos problemas básicos:

- la selección del punto o aproximación inicial;
- el cómputo de todas las raíces o soluciones del problema en cuestión.

Como veremos en esta sección, los métodos homotópicos para ecuaciones polinomiales no tienen ninguno de estos problemas o limitaciones.

Suponga que se desea calcular todas las raíces (algunas pueden ser complejas) de un cierto polinomio $p(z)$. En el método homotópico se construye primeramente un polinomio auxiliar $e(z)$ para el cual la ecuación $e(z) = 0$ es "fácil" de resolver. Luego las raíces de $e(z)$ se *deforman continuamente* hasta obtener las de $p(z)$. Esta deformación continua la llevamos a cabo mediante la *homotopía*:

$$h(z,t) = (1-t)e(z) + tp(z), \quad 0 \leq t \leq 1. \tag{7.53}$$

Note que

$$h(z,0) = e(z), \quad h(z,1) = p(z). \tag{7.54}$$

El siguiente teorema caracteriza la estructura del conjunto solución de la ecuación $h(z,t) = 0$. En particular tenemos condiciones que garantizan que podemos resolver esta ecuación para z como función de t.

Teorema 7.9 ([14], [15])**.** *Sea $p(z)$ un polinomio de grado n y defina $e(z) = az^n + b$. Entonces existe un conjunto $W \subset \mathbb{C} \times \mathbb{C}$ abierto y denso tal que si $(a,b) \in W$, las n raíces de $e(z) = 0$ se pueden continuar por medio de curvas $z(t)$ continuamente diferenciables, definidas en $[0,1]$, y que satisfacen $h(z(t),t) = 0$ para toda $t \in [0,1]$.*

Dado que las curvas que se mencionan en el teorema son continuamente diferenciables, podemos diferenciar la ecuación $h(z(t),t) = 0$ con respecto a t, obteniendo así la ecuación diferencial:

$$h_z(z(t),t)z'(t) + h_t(z(t),t) = 0. \tag{7.55}$$

Si $\alpha_1, \alpha_2, \ldots, \alpha_n$ son las n raíces de $e(z)$, las cuales se pueden calcular mediante el Teorema de Demoivre (vea, e.g., [19]), entonces tenemos las condiciones iniciales

$$z(0) = \alpha_k, \quad 1 \leq k \leq n. \tag{7.56}$$

Estas condiciones, en conjunto con (7.55), definen n problemas de valor inicial para las n curvas soluciones de $h(z(t),t) = 0$ que unen las soluciones de $e(z) = 0$ a las de $p(z) = 0$. Esta *continuación* la podemos llevar a cabo con un paquete computacional para resolver problemas de valor inicial, e.g., `ode45`. Los métodos homotópicos se pueden generalizar a sistemas de polinomios ([14], [15]). Como el problema de buscar los autovalores de una matriz es equivalente a un cierto sistema de polinomial, los métodos homotópicos permiten el computo de todos los autovalores de una matriz, incluso cuando estos son reales, se pueden calcular de forma ordenada de menor a mayor.

Ejemplo 7.10. Vamos a diseñar un método homotópico para la solución de la ecuación

$$z^3 + z - 1 = 0.$$

Usamos como el polinomio fácil de resolver $e(z) = z^3 + 1$. Las raíces de $e(z) = 0$ las podemos calcular usando el Teorema de Demoivre:

$$\alpha_k = \exp\left(\left(\frac{\pi}{3} + \frac{2\pi}{3}k\right)i\right), \quad k = 0,1,2, \tag{7.57}$$

donde $i^2 = -1$. La homotopía (7.53) en este caso está dada por

$$h(z,t) = (1-t)(z^3+1) + t(z^3+z+1),$$

y la ecuación diferencial (7.55) en conjunto con las condiciones iniciales (7.56) toman la forma:

$$z'(t) = \frac{2 - z(t)}{3z(t)^2 + t}, \quad z(0) = \alpha_k. \tag{7.58}$$

El lado derecho de esta ecuación diferencial lo implementamos en MATLAB con la función:

```
function yp=ediff(t,y);
yp(1)=(2-y(1))/(3*y(1)^2+t);
```

Esta función la usamos ahora con `ode45` para resolver el problema de valor inicial (7.58) con los valores iniciales (7.57) mediante el siguiente programa en MATLAB:

```
i=sqrt(-1);
n=3; r=1; theta=pi;
raices=zeros(n,2);
figure(1);clf;
hold on
t0=0; tfinal=1;
for k=0:n-1
    y0=r^(1/n)*exp((theta+2*pi*k)*i/n);
    [t,y] = ode45('ediff',[t0,tfinal],y0);
    [m l]=size(y);
    plot(real(y),imag(y),'k',real(y(m)),imag(y(m)),'k+')
    raices(k+1,1)=t(m);
    raices(k+1,2)=y(m);
end
legend('Curvas de Continuacion','Raices Calculadas')
xlabel('Eje Real');
ylabel('Eje Imaginario');
hold off
disp('Soluciones de la ecuacion z^3+z-1=0:');
raices
```

Las raíces calculadas fueron $-0.3412 \pm 1.1615i$, 0.6823. Estas se ilustran con el símbolo + junto con las curvas soluciones en la Figura 7.7. □

7.8 Ejercicios

Ejercicio 7.1. La ecuación diferencial que modela el proceso de desintegración de un material radioactívo esta dada por:

$$x'(t) = -kx(t), \quad x(0) = x_0,$$

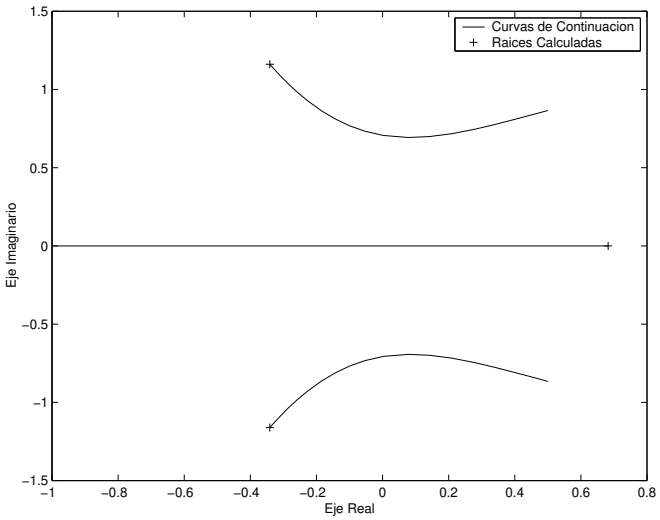


Figura 7.7: Método Homotópico para $z^3 + z - 1 = 0$.

donde k es una constante característica del isótopo radioactívo. Para $x_0 = 50$ y $k = 0.05$ resuelva este problema de valor inicial en el intervalo $[0, 10]$ con $h = 1, 0.1, 0.01$. Compare sus resultados con la solución exacta que es $x(t) = 50\exp(-0.05t)$.

Ejercicio 7.2. Para el problema de valor inicial

$$\begin{cases} y'(t) = \dfrac{2}{t}\, y(t), \quad t > 1, \\ \\ y(1) = 2, \end{cases}$$

verifique que las iteraciones del método de Euler están dadas por:

$$y_{i+1} = \frac{t_{i+2}}{t_i}\, y_i, \quad i \geq 0.$$

Ejercicio 7.3. Para el problema de valor inicial

$$\begin{cases} y'(t) = \dfrac{1}{1+t^2} - 2y(t)^2, \quad 1 < t < 4, \\ \\ y(1) = 2, \end{cases}$$

halle una aproximación de la solución exacta en $t = 1.4$ usando $h = 0.1$ y el Método de Heun (cf. (7.12)).

Ejercicio 7.4. Convierta la ecuación diferencial de orden dos dada por:

$$tx''(t) - x'(t) - 8t^3x(t)^3 = 0,$$

a un sistema de orden uno. Resuelva el sistema resultante en el intervalo $[1, 4]$ si las condiciones iniciales son $x(1) = 1/2$, $x'(1) = -1/2$. La solución exacta en este problema es $x(t) = 1/(1 + t^2)$.

Ejercicio 7.5. Resuelva el siguiente problema de valor inicial usando las subrutinas `ode23` o `ode45`:

$$\begin{cases} x''(t) = 2y'(t) + x(t) - \dfrac{\mu_*(x(t)+\mu)}{r_1^3} - \dfrac{\mu(x(t)-\mu_*)}{r_2^3}, \\ \\ y''(t) = -2x'(t) + y(t) - \dfrac{\mu_* y(t)}{r_1^3} - \dfrac{\mu y(t)}{r_2^3}, \end{cases}$$

donde $x(0) = 1.2$, $x'(0) = 0$, $y(0) = 0$, $y'(0) = -1.0493575$, $\mu = 1/82.45$, $\mu_* = 1 - \mu$, y

$$r_1 = \sqrt{(x(t)+\mu)^2 + y(t)^2}, \quad r_2 = \sqrt{(x(t)-\mu_*)^2 + y(t)^2}.$$

Examine los efectos en los cálculos de la tolerancia especificada por el usuario. Trace las soluciones calculadas como funciones de t y en el plano xy.

Ejercicio 7.6. Las ecuaciones de Lorenz están dadas por el siguiente sistema de ecuaciones diferenciales:

$$\begin{cases} x'(t) &= s(y(t) - x(t)), \\ y'(t) &= rx(t) - y(t) - x(t)z(t), \\ z'(t) &= x(t)y(t) - bz(t), \end{cases}$$

donde s, r, b son parámetros del sistema. Usando `ode45` con una tolerancia de 0.000005 resuelva las ecuaciones de Lorenz para $s = 10$, $r = 126.52$, $b = 8/3$, y con las condiciones iniciales $x(0) = -7.69$, $y(0) = -15.61$, $z(0) = 90.39$. Trace las soluciones calculadas como funciones de t. Trace las soluciones $x(t)$, $z(t)$ en el plano xz. Para los valores de s, r, b dados el sistema de Lorenz exhibe lo que se conoce como comportamiento *caótico*.

Ejercicio 7.7. Considere la ecuación para un péndulo de largo uno dada por:

$$\theta''(t) = -k\,\theta'(t) - g\ \mathrm{sen}(\theta(t)), \quad t > 0,$$

donde $g = 32.2$ pies/seg^2 y $k > 0$. Resuelva ésta ecuación numéricamente para distintos valores de $\theta(0)$, $\theta'(0)$, y k. Haga gráficas de t vs $\theta(t)$, t vs $\theta'(t)$, y de $\theta(t)$ vs $\theta'(t)$. Compare las soluciones calculadas con valores distintos de k. ¿Es o no el movimiento del péndulo periódico en el tiempo? ¿Qué pasa cuando la k es "pequeña" o cuando es "grande"?

Ejercicio 7.8. El problema de valor inicial:

$$\begin{cases} x'(t) = x(t)(1 - y(t)), & x(0) = 5, \\ y'(t) = y(t)(0.75x(t) - 1.5), & y(0) = 2, \end{cases}$$

representa un modelo de presa–depredador donde $x(t)$ es el número (en alguna unidad, digamos miles) de animales presas y $y(t)$ es el número de depredadores en un tiempo t. Resuelva el sistema hasta $t = 1$ y trace las funciones $x(t)$ y $y(t)$ en el mismo sistema de coordenadas.

Ejercicio 7.9. Considere el movimiento de una partícula de masa m que cae verticalmente bajo la influencia de la gravedad g. Suponga que una fuerza de fricción $p(v)$, que depende en la velocidad $v(t)$ de la partícula, actúa en contra del movimiento de la masa. Entonces la velocidad $v(t)$ satisface la ecuación diferencial:

$$mv'(t) = -mg + p(v(t)), \quad t > 0, \quad v(0) \text{ dado},$$

donde suponemos que la dirección positiva es hacia arriba. Sean $m = 1$ kg, $g = 9.8$ m/sec^2 , y $v(0) = 0$. Resuelva la ecuación diferencial para $0 \leq t \leq 20$ y para las siguientes funciones $p(v)$:

a) $p(v) = -0.1v$, el cual es positivo para un cuerpo que cae;

b) $p(v) = 0.1v^2$

Calcule las soluciones con al menos tres dígitos de exactitud. Trace las funciones $v(t)$ y compare las soluciones.

Ejercicio 7.10. Suponga que una cierta subrutina en MATLAB con secuencia de llamada `ode(f,[t0,b],y0,tol)`, calcula una aproximación de la solución del problema de valor inicial (7.1) en el intervalo $[t_0, b]$ con un error `tol`. Para lograr calcular la solución con un error aproximado `tol`, la subrutina ajusta el largo de paso según progresa. Suponga por el contrario que usted desea calcular la solución en intervalos de t del mismo largo. Escriba un programa en MATLAB que utilizando la subrutina `ode` aproxime la solución del problema de valor inicial en intervalos de largo h exactamente.

Ejercicio 7.11. Considere el problema de valor inicial

$$\begin{cases} x''(t) = x(t)^2 - y(t) + e^t, & x(0) = 0 \ , \ x'(0) = 0, \\ y''(t) = x(t) - y(t)^2 - e^t, & y(0) = 1 \ , \ y'(0) = -2. \end{cases}$$

a) Mediante una sustitución apropiada convierta el sistema de orden dos dado a uno equivalente de primer orden.

b) Escriba las instrucciones necesarias en MATLAB para resolver el sistema de orden uno en el intervalo $[0, 2]$ con una tolerancia de 10^{-6}, incluyendo la gráfica de y vs x, usando la función `ode45`.

Ejercicio 7.12. Suponga que para algunas constantes $K, M \geq 0$, la sucesión (α_j) satisface que

$$\alpha_{j+1} \leq K\alpha_j + M, \quad j \geq 0.$$

Demuestre usando inducción matemática que

$$\alpha_j \leq K^j \alpha_0 + M\left(\frac{K^j - 1}{K - 1}\right), \quad j \geq 1.$$

Combinando este resultado con (7.6)–(7.8), demuestre la desigualdad (7.9). **Ayuda**: Use que $1 + x \leq \mathrm{e}^x$ para $x \geq 0$.

Ejercicio 7.13. En el método de Heun (cf. (7.12)), el error de truncación T_{j+1} en el paso $j + 1$ está dado por:

$$T_{j+1} = y(t_{j+1}) - y(t_j) - \frac{h}{2}[f(t_j, y(t_j)) + f(t_{j+1}, y(t_j) + hf(t_j, y(t_j))].$$

Usando el Teorema de Taylor verifique que $T_{j+1} = O(h^3)$. **Ayuda:** Use la ecuación diferencial $y'(t) = f(t, y(t))$ y el Teorema de Taylor para funciones de dos variables.

Ejercicio 7.14. Imitando el proceso ilustrado en el texto para obtener el método de Adams–Bashforth de dos pasos y orden dos (fórmula (7.38)), construya un método de Adams–Bashforth de tres pasos y orden tres.

Ejercicio 7.15. Derive la fórmula para el método de Adams–Moulton de dos pasos y orden tres.

Ejercicio 7.16. Considere el siguiente problema de frontera:

$$\begin{cases} y''(t) = t - 2y^3(t), \quad 0.4 < t < 0.8, \\ y(0.4) = 0.3, \quad y(0.8) = 0.7. \end{cases}$$

Resuelva este problema mediante un método de tiro al blanco.

Ejercicio 7.17. Resuelva el siguiente problema de frontera con un método de tiro al blanco:

$$\begin{cases} y''(t) = \dfrac{1 + (y'(t))^2}{1 + y(t)}, \quad 0 < t < 5, \\ \\ y(0) = 1, \quad y(5) = 10. \end{cases}$$

Este problema modela el paso de un rayo de luz a través de un medio con índice de refracción variable.

Ejercicio 7.18. Considere el siguiente problema de frontera:

$$\begin{cases} y''(t) = -\mathrm{e}^{ty(t)} - \operatorname{sen}(y'(t)), & 1 < t < 2, \\ y(1) = 0, \quad y(2) = 0. \end{cases}$$

Resuelva este problema mediante un método de tiro al blanco.

Ejercicio 7.19. Escriba un programa en MATLAB para calcular todas las soluciones de la ecuación $32x^6 - 48x^4 + 18x^2 - 1 = 0$ usando un método homotópico. Modifique el programa dado en el texto.

Capítulo 8

El Método de Elementos Finitos

El método de *elementos finitos*, FEM por sus siglas en inglés, es un procedimiento para aproximar soluciones de problemas de frontera. La idea básica del método es la de aproximar la solución del problema de frontera con una función en un cierto subespacio de funciones V de dimensión finita. Este concepto es uno bastante general[1] por lo que dependiendo de como se seleccione el subespacio y su base, se pueden obtener muchos de los métodos clásicos de solución de problemas de frontera.

En su forma abstracta, un problema de frontera lineal se puede representar por una ecuación de la forma:

$$D(u) = f, \quad u \in H, \quad f \in Z, \tag{8.1}$$

donde $D : H \longrightarrow Z$ es un operador lineal, la función desconocida es u, y f representa una función o dato conocido. Los conjuntos H, Z representan espacios de funciones "regulares" (*smooth*) los cuales discutiremos en la próxima sección.

Si $\mathcal{H}, \mathcal{Z}$, con $H \subset \mathcal{H}$, $Z \subset \mathcal{Z}$, son espacios no tan "regulares" como H, Z, entonces la *forma débil* del problema de frontera de arriba es: dado $f \in \mathcal{Z}$, buscar una función $u \in \mathcal{H}$ tal que:

$$B(u, v) = L(v), \quad \forall v \in \mathcal{H}, \tag{8.2}$$

donde $B(\cdot, \cdot)$ es una forma *bilinear* asociada o determinada por el operador D, y $L(\cdot)$ es un *funcional lineal* en $\mathcal{Z}$ generado por f. Una propiedad que debe tener

[1] De hecho en el problema de cuadrados mínimos para un sistema lineal sobre–determinado, el lado derecho del sistema se aproxima con elementos o vectores del espacio columna de la matriz en cuestión.

la forma débil, es que si la solución u del problema débil (8.2) pertenece a H, entonces u debe ser solución del problema original (8.1).

Sea $V_h \subset \mathcal{H}$ un subespacio de dimensión $n \geq 1$. La discretización del método de FEM del problema (8.2) es: hallar una función $u_h \in V_h$ tal que:

$$B(u_h, v) = L(v), \quad \forall v \in V_h, \tag{8.3}$$

Si $\{v_1, v_2, \ldots, v_n\}$ es una base de V_h, entonces es suficiente para que (8.3) se cumpla que:

$$B(u_h, v_i) = L(v_i), \quad i = 1, \ldots, n. \tag{8.4}$$

Si ahora escribimos

$$u_h = \sum_{k=1}^{n} \alpha_k v_k, \tag{8.5}$$

entonces (8.4) es equivalente al sistema de ecuaciones lineales:

$$K\boldsymbol{\alpha} = \mathbf{f}, \tag{8.6}$$

donde

$$K = (B(v_k, v_i))_{k,i=1,\ldots,n}, \quad \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^t, \quad \mathbf{f} = (L(v_i))_{i=1,\ldots,n}. \tag{8.7}$$

Con la solución $\boldsymbol{\alpha}$ del sistema (8.6) se obtienen los coeficientes en la expanción (8.5) obteniendo así la aproximación u_h de FEM de la solución u del problema de frontera original. Para una amplia selección de espacios $\mathcal{H}, \mathcal{Z}$ y subespacios $V_h \subset \mathcal{H}$, se obtiene que (u_h) converge a u en $\mathcal{H}$ según $\dim V_h \longrightarrow \infty$.

Aunque en la discusión anterior nos hemos limitado a problemas lineales, el método de FEM se puede utilizar en problemas nolineales también. En tal caso el sistema (8.6) seria entonces uno nolineal y habría que utilizar un método apropiado, e.g., el método de Newton, para aproximar el vector de coeficientes $\boldsymbol{\alpha}$.

8.1 Espacios de funciones

Para $\Omega \subset \mathbb{R}^m$, denotamos por $C(\Omega)$ el conjunto de todas las funciones $u : \Omega \longrightarrow \mathbb{R}$ continuas en Ω. Para $k \geq 1$ decimos que u pertenece a $C^k(\Omega)$ si u y todas sus derivadas parciales de orden menor o igual que k, son continuas en Ω. Esto es:

$$C^k(\Omega) = \left\{ u : \Omega \longrightarrow \mathbb{R} \mid \mathrm{D}^{\mathbf{i}} u \in C(\Omega), \quad |\mathbf{i}| \leq k \right\},$$

donde $\mathbf{i} = (i_1, i_2, \ldots, i_m)$ es un *multi–índice* de largo m por lo que $\mathrm{D}^{\mathbf{i}} u$ representa cualquier derivada parcial de orden $|\mathbf{i}| = i_1 + i_2 + \cdot + i_m$. Identificamos a $C^0(\Omega)$ con $C(\Omega)$. Note que $C^k(\Omega) \subset C(\Omega)$ y la inclusión es propia para $k \geq 1$.

Definimos $L_2(\Omega)$ como el conjunto de funciones $u : \Omega \longrightarrow \mathbb{R}$ tal que[2]:

$$\int_\Omega |u|^2 \, \mathrm{d}\mathbf{x} < \infty.$$

La *norma* o tamaño de una función $u \in L_2(\Omega)$ se define por:

$$\|u\|_0 = \left[\int_\Omega |u|^2 \, \mathrm{d}\mathbf{x} \right]^{\frac{1}{2}}.$$

Note que si Ω es cerrado y acotado, entonces $C(\Omega)$ pertenece $L_2(\Omega)$. En general ésta inclusión es propia.

Definimos el tamaño o norma de una función $u \in C^k(\Omega)$ por:

$$\|u\|_k = \left[\sum_{|\mathbf{i}| \leq k} \int_\Omega \left| \mathrm{D}^{\mathbf{i}} u \right|^2 \, \mathrm{d}\mathbf{x} \right]^{\frac{1}{2}}. \tag{8.8}$$

El conjunto $H^k(\Omega)$, denotado también por $W^{k,2}(\Omega)$, se define como el *completamiento* de $C^k(\Omega)$ en la norma $\|\cdot\|_k$. Esto es $H^k(\Omega)$ consiste de funciones u con $\|u\|_k < \infty$ y tal que existe una sucesión de funciones (u_l) en $C^k(\Omega)$ que converge a u en la norma $\|\cdot\|_k$. El espacio $H^k(\Omega)$ es un ejemplo de un *espacio de Sobolev* y es un *espacio de Hilbert* con producto interior

$$\langle u, v \rangle_k = \sum_{|\mathbf{i}| \leq k} \int_\Omega \mathrm{D}^{\mathbf{i}} u \, \mathrm{D}^{\mathbf{i}} v \, \mathrm{d}\mathbf{x}.$$

Los subespacios de dimensión finita V_h que utilizaremos en (8.3), son subespacios de $H^k(\Omega)$ asociados a una *triangularización* de Ω. Una triangulación se obtiene luego de sobreponer una especie de "rejilla" sobre Ω, sub–dividiendo éste conjunto en sub–conjuntos más pequeños llamados *elementos*. Los subespacios V_h que se utilizan más comúnmente, consisten de funciones que son polinomios por pedazos, esto es, la restricción de cada función en V_h a cada elemento de la triangulación, es un polinomio. Dependiendo de la regularidad de las funciones en V_h, se obtienen diferentes métodos de FEM con distintos grados de aproximabilidad (cf. [33]).

Aunque no entraremos en los detalles técnicos sobre el siguiente resultado, si la frontera de Ω es suficientemente regular, es posible asignar valores a funciones

[2]La integral en la definición de $L_2(\Omega)$ es el sentido de *Lebesgue*. La integral de Riemann y la de Lebesgue coinciden para funciones que son integrables en el sentido de Riemann, pero hay funciones integrables en el sentido de Lebesgue que no lo son en el sentido de la integral de Riemann.

de $H^k(\Omega)$ en la frontera $\partial\Omega$ de Ω. Así que en esta presentación hablaremos ocasionalmente de los valores de éstas funciones en la frontera, en particular cuando especifiquemos condiciones de frontera en el problema original (8.1), sin necesariamente entrar en los detalles técnicos requeridos que justifican dichas discusiones. Por ejemplo denotaremos por $H_0^k(\Omega)$ al subespacio de $H^k(\Omega)$ de funciones que son cero sobre $\partial\Omega$.

8.2 Soluciones débiles y estimados apriori

En esta sección vamos a presentar los resultados más básicos sobre la existencia de soluciones del problema débil (8.2), y los estimados sobre la diferencia entre esta solución y la del problema de FEM (8.3). Para más detalles sobre los resultados de esta sección y otros más, referimos a [33].

Sea $\mathcal{H}$ un espacio vectorial con producto interior $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ y norma

$$\|u\|_{\mathcal{H}} = \sqrt{\langle u, u\rangle_{\mathcal{H}}}, \quad \forall u \in \mathcal{H}.$$

Suponemos que $\mathcal{H}$ es un espacio de Hilbert.[3]

La función $B : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ es una *forma bilineal* si B es lineal en cada argumento, con el otro argumento fijo. Vamos a suponer que B en (8.2) ó (8.3) tiene las siguientes propiedades:

B1: $B(u, v) = B(v, u)$ para todo $u, v \in \mathcal{H}$.

B2: Existe un $\alpha > 0$ tal que $B(u, u) \geq \alpha \|u\|_{\mathcal{H}}$, para todo $u \in \mathcal{H}$.

B3: Existe $C > 0$ tal que $|B(u, v)| \leq C \|u\|_{\mathcal{H}} \|v\|_{\mathcal{H}}$, para todo $u, v \in \mathcal{H}$.

Para el funcional lineal $L : \mathcal{H} \longrightarrow \mathbb{R}$ suponemos que

L1: existe $K > 0$ tal que $|L(v)| \leq K \|v\|_{\mathcal{H}}$, para todo $v \in \mathcal{H}$.

Tenemos ahora el siguiente resultado que garantiza la existencia y unicidad de soluciones para el problema (8.2).

Teorema 8.1 (Lax–Milgram)**.** *Suponga que $B : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ es una forma bilineal y que $L : \mathcal{H} \longrightarrow \mathbb{R}$ es un funcional lineal que satisfacen B1–B3 y L1. Entonces el problema débil* (8.2) *tiene una solución u la cuál es única y $\|u\|_{\mathcal{H}} \leq \frac{K}{\alpha}$.*

[3]Decimos que $\mathcal{H}$ es un espacio de Hilbert si $\mathcal{H}$ es *completo*, esto es, toda sucesión de Cauchy en $\mathcal{H}$ con respecto a la norma $\|\cdot\|_{\mathcal{H}}$, tiene un limite en $\mathcal{H}$.

Remplazando $\mathcal{H}$ con V_h, el Teorema 8.1 igualmente garantiza la existencia de una única solución $u_h \in V_h$ del problema (8.3). Tenemos entonces que

$$\begin{aligned} B(u,v) &= L(v), \quad \forall v \in \mathcal{H}, \\ B(u_h,v) &= L(v), \quad \forall v \in V_h, \end{aligned}$$

lo que implica que

$$B(u-u_h,v) = 0, \quad \forall v \in V_h. \tag{8.9}$$

Esto se conoce como la *propiedad de ortogonalidad del error* $e = u - u_h$. Usando esto podemos ahora verificar el siguiente resultado.

Teorema 8.2 (Estimado apriori)**.** *Sean u y u_h las soluciones de* (8.2) *y* (8.3) *respectivamente. Suponga que B1–B3 y L1 se cumplen. Entonces*

$$\|u-u_h\|_{\mathcal{H}} \leq \left(1+\frac{C}{\alpha}\right) \inf_{v\in V_h} \|u-v\|_{\mathcal{H}}.$$

Demostración: Para $v \in V_h$ tenemos que

$$\begin{aligned} B(u_h-v,u_h-v) &= B(u_h-v,u_h-v) + B(u-v,u_h-v) - B(u-v,u_h-v), \\ &= B(u-v,u_h-v) + B(u_h-u,u_h-v). \end{aligned}$$

Pero como $u_h - v \in V_h$, tenemos por la propiedad de ortogonalidad que $B(u_h - u, u_h - v) = 0$. Llegamos entonces a que

$$B(u_h-v,u_h-v) = B(u-v,u_h-v).$$

Esto combinado con B2 y B3 implica que

$$\|u_h-v\|_{\mathcal{H}} \leq \frac{C}{\alpha}\|u-v\|_{\mathcal{H}}.$$

Pero por la desigualdad del triangulo,

$$\|u-u_h\|_{\mathcal{H}} \leq \|u-v\|_{\mathcal{H}} + \|v-u_h\|_{\mathcal{H}},$$

que combinado con la desigualdad anterior nos da que

$$\|u-u_h\|_{\mathcal{H}} \leq \left(1+\frac{C}{\alpha}\right)\|u-v\|_{\mathcal{H}}.$$

El resultado del teorema es consecuencia de ésto ya que $v \in V_h$ es arbitrario. □

Tenemos entonces que el problema de estimar el error $u - u_h$ se reduce a determinar el tamaño o orden de

$$\inf_{v\in V_h} \|u-v\|_{\mathcal{H}}.$$

Esta cantidad mide cuan bien se puede aproximar a una función cualquiera $u \in \mathcal{H}$, por funciones en V_h, y se conoce o llama el *grado de aproximabilidad* del FEM. Para estimar este ínfimo es necesario especificar los espacios $\mathcal{H}$ y V_h. Para los problemas de frontera que discutiremos en las próximas secciones, es suficiente tomar a $\mathcal{H} = H^1(\Omega)$. Si $K = \{K_j\}$ representa una triangularización de Ω, entonces definimos a $V_h = \mathbb{P}_m$, donde

$$\mathbb{P}_m = \left\{ v \in C(\Omega) \,:\, v|_{K_j} \text{ es polinomio de grado m, } \forall K_j \in K \right\}.$$

Se puede verificar que $\mathbb{P}_m$ es un subespacio de $H^1(\Omega)$. Si definimos la semi–norma

$$|u|_k = \left[\sum_{|\mathbf{i}|=k} \int_\Omega \left| \mathrm{D}^{\mathbf{i}} u \right|^2 \mathrm{d}\mathbf{x} \right]^{\frac{1}{2}}, \tag{8.10}$$

podemos entonces enunciar el siguiente resultado sobre el grado de aproximabilidad en este caso (cf. [5]). La h en el siguiente teorema representa el diámetro máximo de cualquier elemento K_j de la triangularización K.

Teorema 8.3 (Bramble y Hilbert)**.** *Sea $\Omega \subset \mathbb{R}^d$, $d \leq 3$. Bajo ciertas condiciones adicionales en $\partial\Omega$, existe una constante C tal que para cualquier $u \in H^{m+1}(\Omega) \cap H_0^1(\Omega)$,*

$$\inf_{v \in \mathbb{P}_m} |u - v|_k \leq C h^{m+1-k} |u|_{m+1}, \quad k = 0, 1.$$

8.3 Problemas lineales

Vamos a ilustrar los aspectos esenciales del FEM con dos problemas lineales en una y dos dimensiones.

8.3.1 Problema en una dimensión

El primero de éstos problemas es un caso en que la función desconocida depende de una sola variable. El problema de frontera consiste de:

$$-w''(x) + g(x)w(x) = F(x), \quad a < x < b, \tag{8.11a}$$

$$w(a) = \alpha, \qquad w(b) = \beta. \tag{8.11b}$$

Aquí g, F son funciones dadas, digamos en $C(a,b)$, con $g(x) > 0$ en (a,b), y $\alpha, \beta \in \mathbb{R}$ son números dados. La función w desconocida es un elemento de $C^2[a,b]$. Si definimos

$$q(x) = \alpha + \left(\frac{\beta - \alpha}{b - a} \right) (x - a), \tag{8.12}$$

y ponemos $u = w - q$, entonces u es solución del problema de frontera:

$$\begin{aligned} -u''(x) + g(x)u(x) &= f(x), \quad a < x < b, && \text{(8.13a)} \\ u(a) = 0, \qquad & u(b) = 0, && \text{(8.13b)} \end{aligned}$$

donde $f = F - qg$.

Considere el problema reducido (8.13). Si multiplicamos la ecuación (8.13a) por una función $v \in C^2[a,b]$ tal que $v(a) = v(b) = 0$, integramos en ambos lados sobre (a,b), y usamos la formula de integración por partes, obtenemos que:

$$\int_a^b \big(u'(x)v'(x) + g(x)u(x)v(x)\big)\,\mathrm{d}x = \int_a^b f(x)v(x)\,\mathrm{d}x.$$

Si defimimos:

$$\begin{aligned} B(u,v) &= \int_a^b \big(u'(x)v'(x) + g(x)u(x)v(x)\big)\,\mathrm{d}x, && \text{(8.14)} \\ L(v) &= \int_a^b f(x)v(x)\,\mathrm{d}x, && \text{(8.15)} \end{aligned}$$

entonces la forma débil del problema de frontera (8.13) es: hallar $u \in H_0^1(a,b)$ tal que

$$B(u,v) = L(v), \quad \forall v \in H_0^1(a,b). \tag{8.16}$$

Es fácil verificar que $B(\cdot,\cdot)$ y $L(\cdot)$ satisfacen las condiciones del Teorema 8.1 por lo que este problema tiene una solución única $u \in H_0^1(a,b)$. De (8.16) obtenemos que

$$\int_a^b u'(x)v'(x)\,\mathrm{d}x = -\int_a^b \left[g(x)u(x) - f(x)\right] v(x)\,\mathrm{d}x, \quad \forall v \in C_0^\infty(a,b),$$

lo que implica que $gu - f$ es la derivada débil de u'. Si $f \in L_2(a,b)$, entonces $qu - f \in L_2(a,b)$, por lo que $u \in H_0^2(a,b)$. Este tipo de argumento de "retro–alimentación", denominado en el idioma inglés como *bootstrap*, se utiliza para determinar la regularidad de la solución del problema a partir de la regularidad de los datos.

Note que las condiciones de frontera (8.13b) forman parte del espacio de funciones $H_0^1(a,b)$ y por esto se les refiere como *condiciones de frontera esenciales*. Es fácil ver que si $u \in C^2[a,b] \cap H_0^1(a,b)$ es solución de (8.16), entonces u es solución del problema de frontera (8.13).

Vamos a describir ahora un FEM para resolver (8.16). Primero introducimos una partición $a = x_0 < x_1 < \cdots < x_n = b$ del intervalo $[a,b]$. Luego definimos las

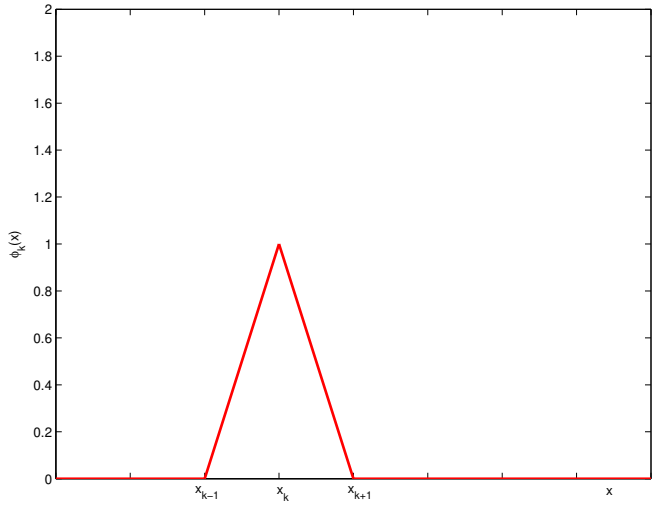


Figura 8.1: Funciones base tipo "sombrero" que son polinomios lineales por pedazos.

funciones base $(\phi_k)_{k=1,\ldots,n-1}$ que son polinomios lineales por pedazos tales que

$$\phi_k(x_j) = \begin{cases} 1, & j = k, \\ 0, & j \neq k. \end{cases}, \quad 1 \leq k \leq n-1.$$

De hecho podemos escribir que:

$$\phi_k(x) = \begin{cases} \dfrac{x - x_{k-1}}{x_k - x_{k-1}}, & x_{k-1} \leq x \leq x_k, \\ \dfrac{x_{k+1} - x}{x_{k+1} - x_k}, & x_k \leq x \leq x_{k+1}, \\ 0, & \text{en el resto.} \end{cases} \tag{8.17}$$

(Vea la Figura (8.1).) El subespacio V_h de $H_0^1(a,b)$ está dado por:

$$V_h = \left\{ v \in C(a,b) \,:\, v = \sum_{k=1}^{n-1} \alpha_k \phi_k \right\}. \tag{8.18}$$

Note que V_h tiene dimensión $n-1$. El problema reducido o lo mismo que la discretización de FEM de (8.16), está dado por: hallar $u_h \in V_h$ tal que

$$B(u_h, v) = L(v), \quad \forall v \in V_h. \tag{8.19}$$

Nuevamente, por el Teorema 8.1, este problema tiene una solución $u_h \in V_h$ que es única. Combinado el estimado apriori del Teorema 8.2 con el resultado del Teorema 8.3, podemos concluir (cf. (8.10)) que para una constante $C > 0$,

$$|u - u_h|_k \leq Ch^{2-k} |u|_2 , \quad k = 0, 1. \tag{8.20}$$

Es fácil ver que para la base (8.17) la matriz K en (8.6) es simétrica y tri–diagonal. De hecho con $h_k = x_k - x_{k-1}$, $1 \leq k \leq n$, tenemos que

$$\begin{aligned}
K_{kk} &= \frac{h_k + h_{k+1}}{h_k h_{k+1}} + \int_{x_{k-1}}^{x_{k+1}} g(x)\phi_k^2(x)\,\mathrm{d}x, \quad k = 1, \ldots, n-1, && \text{(8.21a)}\\
K_{k,k-1} &= -\frac{1}{h_k} + \int_{x_{k-1}}^{x_k} g(x)\phi_{k-1}(x)\phi_k(x)\,\mathrm{d}x, \quad k = 2, \ldots, n-1, && \text{(8.21b)}\\
K_{k,k+1} &= K_{k+1,k}, \quad k = 1, \ldots, n-2. && \text{(8.21c)}
\end{aligned}$$

El lado derecho $\mathbf{f}$ del sistema (8.6) está dado por:

$$f_k = \int_{x_{k-1}}^{x_{k+1}} f(x)\phi_k(x)\,\mathrm{d}x, \quad k = 1, \ldots, n-1. \tag{8.22}$$

Los valores de los integrales en (8.21) y (8.22) dependen de la forma particular de las funciones g y f. Estos integrales en general hay que aproximarlos también. Por ejemplo podemos usar las aproximaciones:

$$\begin{aligned}
\int_{x_{k-1}}^{x_{k+1}} g(x)\phi_k^2(x)\,\mathrm{d}x &\approx g(x_k) \int_{x_{k-1}}^{x_{k+1}} \phi_k^2(x)\,\mathrm{d}x\\
&= \frac{h_k + h_{k+1}}{3}\, g(x_k), && \text{(8.23a)}\\
\int_{x_{k-1}}^{x_k} g(x)\phi_{k-1}(x)\phi_k(x)\,\mathrm{d}x &\approx g(x_{k-1}) \int_{x_{k-1}}^{x_k} \phi_{k-1}(x)\phi_k(x)\,\mathrm{d}x\\
&= \frac{h_k}{6}\, g(x_{k-1}), && \text{(8.23b)}\\
\int_{x_{k-1}}^{x_{k+1}} f(x)\phi_k(x)\,\mathrm{d}x &\approx f(x_k) \int_{x_{k-1}}^{x_{k+1}} \phi_k(x)\,\mathrm{d}x\\
&= \frac{h_k + h_{k+1}}{2}\, f(x_k). && \text{(8.23c)}
\end{aligned}$$

Las ecuaciones (8.21), (8.23a), y (8.23b) las podemos recoger en el siguiente programa en MATLAB que calcula la matriz K:

```
function K=stiffM(g_func,x)
n=length(x)-1;
```

```
h=diff(x);
g_coef=feval(g_func,x(2:n));
diag=sparse(1:n-1,1:n-1,...
    (h(1:n-1)+h(2:n))./(h(1:n-1).*h(2:n))+...
    (1/3)*(h(1:n-1)+h(2:n)).*g_coef,n-1,n-1);
ldiag=sparse(2:n-1,1:n-2,-1./h(2:n-1)+...
    (1/6)*h(2:n-1).*g_coef(1:n-2),n-1,n-1);
K=ldiag+diag+ldiag';
```

Note el uso de la instrucción `sparse` al definir K para así aprovechar la estructura tri–diagonal de la matriz K. Esto tiene como consecuencia que el sistema lineal (8.6) se pueda resolver en $O(n)$ operaciones. El argumento `g_func` es el nombre de una subrutina que calcula la función coeficiente $g(x)$.

De igual forma podemos recoger la ecuación (8.22) y la aproximación (8.23c) en la siguiente rutina para evaluar el lado derecho $\boldsymbol{b}$, donde `f_func` es el nombre de una subrutina que evalúa la función $f(x)$, el lado de la ecuación diferencial:

```
function F=rhside(f_func,x)
n=length(x)-1;
h=diff(x);
f_coef=feval(f_func,x(2:n));
F=0.5*((h(1:n-1)+h(2:n))).*f_coef;
```

Ejemplo 8.4. Considere el problema de frontera:

$$\begin{aligned} -u''(x)+u(x) &= 1, \quad 0<x<1, \\ u(0)=0, &\qquad u(1)=0. \end{aligned}$$

Usando las funciones:

```
function z=coeficiente_g(y)
z=ones(size(y));

function z=ladoderecho_f(y)
z=ones(size(y));
```

podemos aproximar la solución de este problema con las siguientes instrucciones en MATLAB:

```
a=0;
b=1;
n=160;
```

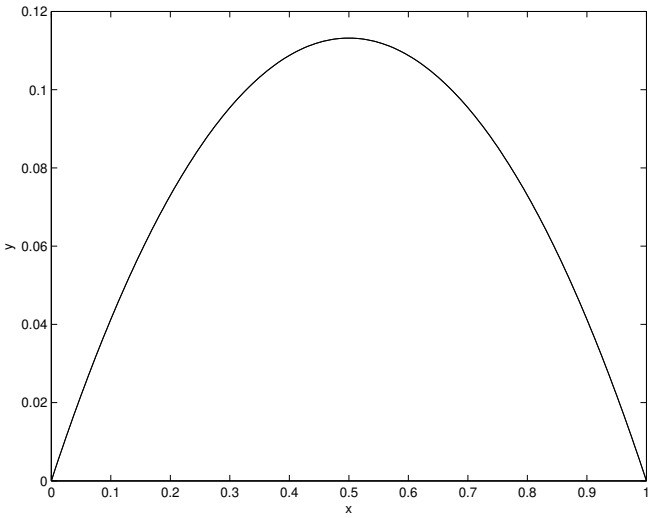


Figura 8.2: Solución aproximada del problema de frontera del Ejemplo (8.4).

```
x=linspace(a,b,n+1);
K=stiffM(@coeficiente_g,x);
F=rhside(@ladoderecho_f,x);
alfa=K\F';
plot(x,[0,alfa',0])
```

En la Figura (8.2) mostramos una gráfica de la solución aproximada.

En este ejemplo tenemos la solución exacta de modo que podemos calcular el error exacto en la aproximación. De hecho, la solución exacta del problema de frontera es:

$$u(x) = 1 - \left(\frac{\mathrm{e}^x + \mathrm{e}^{1-x}}{1 + \mathrm{e}} \right).$$

Para un valor de n dado, siendo n el numero de subintervalos en la partición, definimos

$$e_n = \max \left\{ |u(x_k) - u_h(x_k)| \, : \, k = 1, \ldots, n-1 \right\}. \tag{8.24}$$

En la siguiente tabla mostramos el comportamiento del error e_n según la n se duplica:

n	e_n	e_n/e_{n-1}
10	8.5485e-005	—
20	2.1351e-005	4.0038
40	5.3365e-006	4.0009
80	1.3341e-006	4.0002
160	3.3351e-007	4.0001

Estos resultados indican que el método de FEM basado en los espacios (8.17), (8.18) y las formulas (8.21), (8.22), (8.23), tiene orden de convergencia $O(n^{-2})$ en la norma (8.24) para este problema. Esto es esencialmente el resultado (8.20) con $k = 0$. Utilizando otros espacios con más regularidad que (8.17), (8.18), e.g., polinomios cuadráticos por pedazos, es posible diseñar métodos de FEM con orden de convergencia más alto. □

8.3.2 Problemas en dos dimensiones

Consideramos ahora un problema modelo donde la función desconocida u depende de dos variables (x, y):

$$-\Delta u(x,y) + g(x,y)u(x,y) = f(x,y), \quad (x,y) \in \Omega, \tag{8.25a}$$
$$u(x,y) = 0, \qquad (x,y) \in \partial\Omega. \tag{8.25b}$$

Aquí

$$\Delta u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \quad \Omega = (a,b) \times (c,d),$$
$$\partial\Omega = \{(x,y) \,:\, x = a, b, \quad c \le y \le d\} \cup \{(x,y) \,:\, y = c, d, \quad a \le x \le b\}.$$

El conjunto $\bar{\Omega} = \Omega \cup \partial\Omega$ se llama la *clausura* de Ω. Suponemos que $g \in C(\Omega)$ y que $g > 0$ en Ω.

La (primera) identidad de Green dice que si $u, v \in C^2(\bar{\Omega})$, entonces:

$$\int_\Omega v\,\Delta u \,\mathrm{d}x\mathrm{d}y = \int_{\partial\Omega} u \frac{\partial v}{\partial \mathbf{n}} \,\mathrm{d}s - \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x\mathrm{d}y, \tag{8.26}$$

donde $\mathbf{n}$ es la normal unitaria (exterior) a $\partial\Omega$ y

$$\frac{\partial v}{\partial \mathbf{n}} = \nabla v \cdot \mathbf{n},$$

es la *derivada direccional* de v en la dirección de $\mathbf{n}$. Si multiplicamos (8.25a) por $v \in C^2(\bar{\Omega})$ tal que $v = 0$ en $\partial\Omega$, y usamos la identidad de Green (8.26), tenemos que:

$$\int_\Omega \left[\nabla u(x,y) \cdot \nabla v(x,y) + g(x,y)u(x,y)v(x,y)\right] \,\mathrm{d}x\mathrm{d}y = \int_\Omega f(x,y)v(x,y) \,\mathrm{d}x\mathrm{d}y.$$

Si definimos

$$B(u,v) = \int_\Omega \left[\nabla u(x,y)\cdot\nabla v(x,y) + g(x,y)u(x,y)v(x,y)\right]\,\mathrm{d}x\mathrm{d}y, \quad (8.27)$$

$$L(v) = \int_\Omega f(x,y)v(x,y)\,\mathrm{d}x\mathrm{d}y, \quad (8.28)$$

entonces la forma débil del problema de frontera (8.25) es: hallar $u \in H_0^1(\Omega)$ tal que

$$B(u,v) = L(v), \quad \forall v \in H_0^1(\Omega). \quad (8.29)$$

Nuevamente, se puede verificar que las condiciones del Teorema 8.1 se cumplen por lo que este problema tiene una solución única $u \in H_0^1(\Omega)$. Un argumento tipo "bootstrap" muestra que si $f \in L_2(\Omega)$, entonces $u \in H_0^2(\Omega)$. Se puede verificar que si $u \in C^2(\bar{\Omega}) \cap H_0^1(\Omega)$ es solución de (8.29), entonces u es solución del problema de frontera (8.25).

Para describir un método de FEM para el problema (8.29), primero introducimos particiones $a = x_0 < x_1 < \cdots < x_n = b$, $c = y_0 < y_1 < \cdots < y_m = d$ de $[a,b]$ y $[c,d]$ respectivamente. Esto induce una partición de Ω en sub–rectángulos de la forma:

$$\begin{aligned}\Omega &= \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\Omega_{ij},\\ \Omega_{ij} &= [x_{i-1},x_i]\times[y_{j-1},y_j], \quad 1\le i\le n, \quad 1\le j\le m.\end{aligned}$$

Sean $h_i = x_i - x_{i-1}$, $i = 1,\ldots,n$, y $\delta_j = y_j - y_{j-1}$, $j = 1,\ldots,m$. Para cada punto (interior) de la partición, esto es, (x_i,y_j), $1 \le i \le n-1$, $1 \le j \le m-1$, definimos la función base (tipo *sombrero*) $\phi_{ij}(x,y)$ por:

$$\phi_{ij}(x,y) = \begin{cases} \dfrac{1}{h_{i+1}\delta_j}(x_{i+1}-x)(y-y_{j-1}) & , \quad x_i \le x \le x_{i+1}, \quad y_{j-1} \le y \le y_j,\\ \dfrac{1}{h_{i+1}\delta_{j+1}}(x_{i+1}-x)(y_{j+1}-y), & x_i \le x \le x_{i+1}, \quad y_j \le y \le y_{j+1},\\ \dfrac{1}{h_i\delta_{j+1}}(x-x_{i-1})(y_{j+1}-y), & x_{i-1} \le x \le x_i, \quad y_j \le y \le y_{j+1},\\ \dfrac{1}{h_i\delta_j}(x-x_{i-1})(y-y_{j-1}), & x_{i-1} \le x \le x_i, \quad y_{j-1} \le y \le y_j,\\ 0, & \text{en el resto.}\end{cases}$$

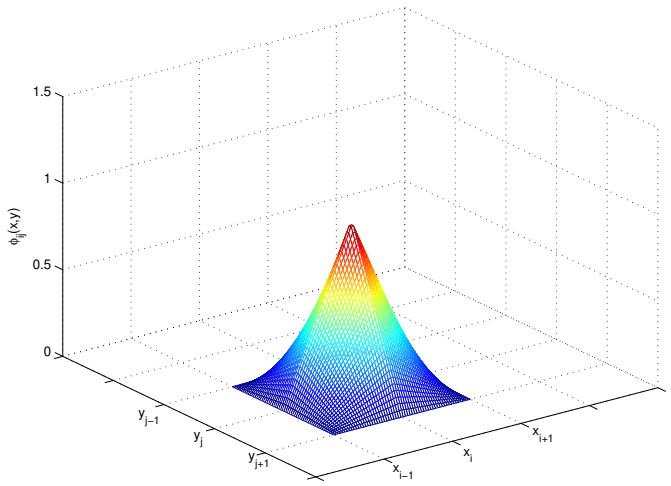


Figura 8.3: Función base ϕ_{ij} en el problema modelo de dos dimensiones.

(Vea la Figura (8.3).) Note que

$$\phi_{i,j}(x_k, y_l) = \begin{cases} 1, & \text{si } k = i \quad \text{y} \quad l = j, \\ 0, & \text{si } k \neq i \quad \text{ó} \quad l \neq j. \end{cases}$$

El subespacio V_h de $H_0^1(\Omega)$ está dado por:

$$V_h = \left\{ v \in C(\Omega) \, : \, v = \sum_{i=1}^{n-1} \sum_{j=1}^{m-1} \alpha_{ij} \phi_{ij} \right\}. \tag{8.30}$$

Note que V_h tiene dimensión $N = (n-1)(m-1)$. La discretización de FEM de (8.29) esta dada en este caso por: hallar $u_h \in V_h$ tal que

$$B(u_h, v) = L(v), \quad \forall v \in V_h. \tag{8.31}$$

Por el Teorema 8.1, este problema tiene una solución $u_h \in V_h$ que es única. El resultado del Teorema 8.3 no aplica en este caso ya que la rejilla o triangulación de Ω no es con triángulos. No obstante se puede verificar que en el caso de la rejilla con rectángulos, el siguiente estimado se cumple

$$\inf_{v \in V_h} \|u - v\|_1 \leq Ch \, |u|_2 \, .$$

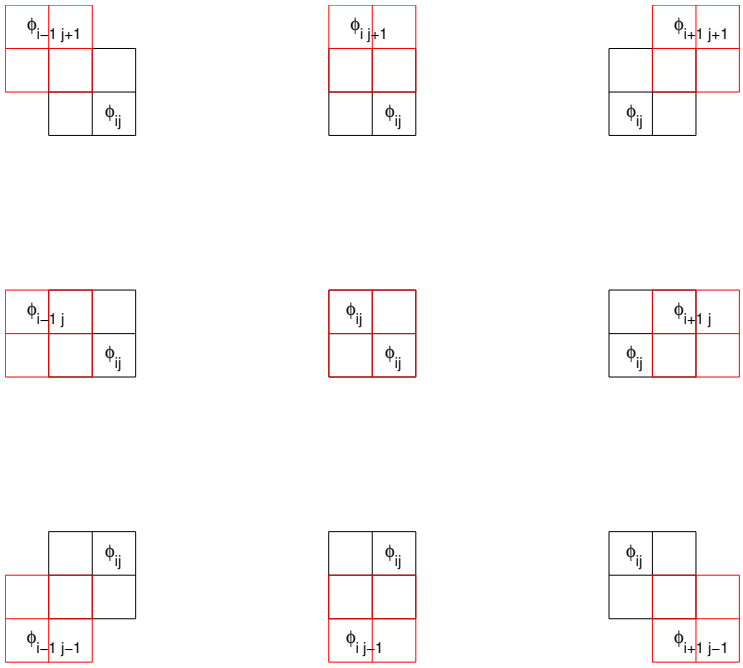


Figura 8.4: Diagrama de solapamiento de las funciones ϕ_{ij}, ϕ_{kl} para $|i-k| \leq 1$ y $|j-l| \leq 1$.

Combinado esto con el estimado apriori del Teorema 8.2, podemos concluir que para una constante $C > 0$,

$$\|u - u_h\|_1 \leq Ch\,|u|_2\,. \tag{8.32}$$

Para montar la matriz K, necesitamos calcular los integrales:

$$I_{ijkl} = \int_\Omega \nabla\phi_{ij}(x,y) \cdot \nabla\phi_{kl}(x,y)\,\mathrm{d}x\mathrm{d}y. \tag{8.33}$$

Como el soporte de ϕ_{ij} está en $\Omega_{ij} \cup \Omega_{i+1\,j} \cup \Omega_{i\,j+1} \cup \Omega_{i+1\,j+1}$, estos integrales son distintos de cero únicamente cuando $|i-k| \leq 1$ y $|j-l| \leq 1$. (Vea la Figura (8.4).) Usando las simetrías de las funciones (ϕ_{ij}) vemos que es suficiente calcular los casos:

$$(k,l) = (i-1, j+1), (i, j+1), (i+1, j+1), (i-1, j), (i, j).$$

Un cálculo elemental pero un tanto extenso nos muestra que:

$$
\begin{aligned}
I_{ij\,i-1\,j+1} &= -\frac{1}{6}\left(\frac{\delta_{j+1}}{h_i}+\frac{h_i}{\delta_{j+1}}\right), \quad 2\le i\le n-1,\ 1\le j\le m-2,\\
I_{ij\,i\,j+1} &= \frac{\delta_{j+1}}{6}\left(\frac{1}{h_i}+\frac{1}{h_{i+1}}\right)-\left(\frac{h_i+h_{i+1}}{3\delta_{j+1}}\right), \quad 1\le i\le n-1,\ 1\le j\le m-2,\\
I_{ij\,i+1\,j+1} &= -\frac{1}{6}\left(\frac{\delta_{j+1}}{h_{i+1}}+\frac{h_{i+1}}{\delta_{j+1}}\right), \quad 1\le i\le n-2,\ 1\le j\le m-2,\\
I_{ij\,i-1\,j} &= \frac{h_i}{6}\left(\frac{1}{\delta_j}+\frac{1}{\delta_{j+1}}\right)-\left(\frac{\delta_j+\delta_{j+1}}{3h_i}\right), \quad 2\le i\le n-1,\ 1\le j\le m-1,\\
I_{ij\,ij} &= \frac{1}{3}\left(\frac{h_{i+1}}{\delta_{j+1}}+\frac{\delta_{j+1}}{h_{i+1}}+\frac{h_i}{\delta_{j+1}}+\frac{\delta_{j+1}}{h_i}+\frac{h_{i+1}}{\delta_j}+\frac{\delta_j}{h_{i+1}}+\frac{h_i}{\delta_j}+\frac{\delta_j}{h_i}\right),\\
&\qquad 1\le i\le n-1,\ 1\le j\le m-1.
\end{aligned}
$$

Las entradas de la matriz K están dadas ahora por:

$$
\begin{aligned}
K_{ij\,i-1\,j+1} &= I_{ij\,i-1\,j+1}\\
&\quad +\int_{x_{i-1}}^{x_i}\int_{y_j}^{y_{j+1}} g(x,y)\phi_{ij}(x,y)\phi_{i-1\,j+1}(x,y)\,\mathrm{d}x\mathrm{d}y, \qquad (8.34a)\\
K_{iji\,j+1} &= I_{iji\,j+1}\\
&\quad +\int_{x_{i-1}}^{x_{i+1}}\int_{y_j}^{y_{j+1}} g(x,y)\phi_{ij}(x,y)\phi_{i\,j+1}(x,y)\,\mathrm{d}x\mathrm{d}y, \qquad (8.34b)\\
K_{ij\,i+1\,j+1} &= I_{ij\,i+1\,j+1}\\
&\quad +\int_{x_i}^{x_{i+1}}\int_{y_j}^{y_{j+1}} g(x,y)\phi_{ij}(x,y)\phi_{i+1\,j+1}(x,y)\,\mathrm{d}x\mathrm{d}y, \qquad (8.34c)\\
K_{ij\,i-1\,j} &= I_{ij\,i-1\,j}\\
&\quad +\int_{x_{i-1}}^{x_i}\int_{y_{j-1}}^{y_{j+1}} g(x,y)\phi_{ij}(x,y)\phi_{i-1\,j}(x,y)\,\mathrm{d}x\mathrm{d}y, \qquad (8.34d)\\
K_{ijij} &= I_{ijij}+\int_{x_{i-1}}^{x_{i+1}}\int_{y_{j-1}}^{y_{j+1}} g(x,y)\phi_{ij}^2(x,y)\,\mathrm{d}x\mathrm{d}y, \qquad (8.34e)\\
K_{ij\,i+1\,j} &= K_{i+1\,jij}, \quad 1\le i\le n-2,\ 1\le j\le m-1, \qquad (8.34f)\\
K_{ij\,i-1\,j-1} &= K_{i-1\,j-1\,ij}, \quad 2\le i\le n-1,\ 2\le j\le m-1, \qquad (8.34g)\\
K_{iji\,j-1} &= K_{i\,j-1\,ij}, \quad 1\le i\le n-1,\ 2\le j\le m-1, \qquad (8.34h)\\
K_{ij\,i+1\,j-1} &= K_{i+1\,j-1\,ij}, \quad 1\le i\le n-2,\ 2\le j\le m-1, \qquad (8.34i)
\end{aligned}
$$

donde en las ultimas cuatro ecuaciones usamos que $K_{ijkl}=K_{klij}$. El lado derecho $\mathbf{f}$ del sistema (8.6) está dado por:

$$
f_{ij}=\int_{x_{i-1}}^{x_{i+1}}\int_{y_{j-1}}^{y_{j+1}} f(x,y)\phi_{ij}(x,y)\,\mathrm{d}x\mathrm{d}y, \quad 1\le i\le n-1, \quad 1\le j\le m-1. \qquad (8.35)
$$

Los valores de las integrales en (8.34) y (8.35) dependen de las funciones g y f y en general hay que aproximarlos. Estaremos usando las siguientes aproximaciones:

$$\int_{x_{i-1}}^{x_i}\int_{y_j}^{y_{j+1}} g\phi_{ij}\phi_{i-1\,j+1}\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{36}\,h_i\delta_{j+1}\,g(x_i,y_j), \tag{8.36a}$$

$$\int_{x_{i-1}}^{x_{i+1}}\int_{y_j}^{y_{j+1}} g\phi_{ij}\phi_{i\,j+1}\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{18}\,(h_i+h_{i+1})\delta_{j+1}\,g(x_i,y_j), \tag{8.36b}$$

$$\int_{x_i}^{x_{i+1}}\int_{y_j}^{y_{j+1}} g\phi_{ij}\phi_{i+1\,j+1}\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{36}\,h_{i+1}\delta_{j+1}\,g(x_i,y_j), \tag{8.36c}$$

$$\int_{x_{i-1}}^{x_i}\int_{y_{j-1}}^{y_{j+1}} g\phi_{ij}\phi_{i-1\,j}\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{18}\,h_i(\delta_j+\delta_{j+1})g(x_i,y_j), \tag{8.36d}$$

$$\int_{x_{i-1}}^{x_{i+1}}\int_{y_{j-1}}^{y_{j+1}} g\phi_{ij}^2\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{9}\,(h_i+h_{i+1})(\delta_j+\delta_{j+1})g(x_i,y_j), \tag{8.36e}$$

$$\int_{x_{i-1}}^{x_{i+1}}\int_{y_{j-1}}^{y_{j+1}} f\phi_{ij}\,\mathrm{d}x\mathrm{d}y \approx \frac{1}{4}\,(h_i+h_{i+1})(\delta_j+\delta_{j+1})f(x_i,y_j). \tag{8.36f}$$

Si $u_h=\sum_{i=1}^{n-1}\sum_{j=1}^{m-1}\alpha_{ij}\phi_{ij}$ es la solución del problema reducido (8.31), y definimos los vectores

$$\boldsymbol{\alpha}_j=(\alpha_{1j},\alpha_{2j},\ldots,\alpha_{n-1\,j})^t,\quad \mathbf{f}_j=(f_{1j},f_{2j},\ldots,f_{n-1\,j})^t,\quad j=1,\ldots,m-1,$$

y las matrices tridiagonales de tamaño $(n-1)\times(n-1)$:

$$A_j=\begin{bmatrix} K_{1j1j} & K_{1j2j} & 0 & \cdots & 0\\ K_{2j1j} & K_{2j2j} & K_{2j3j} & \cdots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \cdots & K_{n-2\,j\,n-3\,j} & K_{n-2\,j\,n-2\,j} & K_{n-2\,j\,n-1\,j}\\ 0 & \cdots & 0 & K_{n-1\,j\,n-2\,j} & K_{n-1\,j\,n-1\,j}\end{bmatrix},$$

para $j=1,\ldots,m-1$,

$$B_{j-1}=\begin{bmatrix} K_{1j\,1\,j-1} & K_{1j\,2,j-1} & 0 & \cdots & 0\\ K_{2j\,1\,j-1} & K_{2j\,2\,j-1} & K_{2j\,3\,j-1} & \cdots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \cdots & K_{n-2\,j\,n-3\,j-1} & K_{n-2\,j\,n-2\,j-1} & K_{n-2\,j\,n-1\,j-1}\\ 0 & \cdots & 0 & K_{n-1\,j\,n-2\,j-1} & K_{n-1\,j\,n-1\,j-1}\end{bmatrix},$$

para $j = 2, \ldots, m-1$, y

$$C_{j+1} = \begin{bmatrix} K_{1j\,1\,j+1} & K_{1j\,2,j+1} & 0 & \cdots & 0 \\ K_{2j\,1\,j+1} & K_{2j\,2\,j+1} & K_{2j\,3\,j+1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & K_{n-2\,j\,n-3\,j+1} & K_{n-2\,j\,n-2\,j+1} & K_{n-2\,j\,n-1\,j+1} \\ 0 & \cdots & 0 & K_{n-1\,j\,n-2\,j+1} & K_{n-1\,j\,n-1\,j+1} \end{bmatrix},$$

para $j = 1, \ldots, m-2$, entonces el sistema (8.6) para el problema reducido (8.31) se puede escribir en forma de bloques como:

$$\begin{aligned} A_1\boldsymbol{\alpha}_1 + C_2\boldsymbol{\alpha}_2 &= \mathbf{f}_1, && \text{(8.37a)} \\ B_{j-1}\boldsymbol{\alpha}_{j-1} + A_j\boldsymbol{\alpha}_j + C_{j+1}\boldsymbol{\alpha}_{j+1} &= \mathbf{f}_j, \quad j = 2, \ldots, m-2, && \text{(8.37b)} \\ B_{m-2}\boldsymbol{\alpha}_{m-2} + A_{m-1}\boldsymbol{\alpha}_{m-1} &= \mathbf{f}_{m-1}, && \text{(8.37c)} \end{aligned}$$

o lo mismo, en la forma matricial (8.6) donde

$$K = \begin{bmatrix} A_1 & C_2 & 0 & \cdots & 0 \\ B_1 & A_2 & C_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & B_{m-3} & A_{m-2} & C_{m-1} \\ 0 & \cdots & 0 & B_{m-2} & A_{m-1} \end{bmatrix}, \qquad \text{(8.38a)}$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_{m-2} \\ \boldsymbol{\alpha}_{m-1} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{m-2} \\ \mathbf{f}_{m-1} \end{bmatrix}. \qquad \text{(8.38b)}$$

Ejemplo 8.5. Consideramos el caso especial de (8.25) donde:

$$\begin{aligned} \Omega &= [0,1] \times [0,2], \\ g(x,y) &= 100x + y^3, \\ f(x,y) &= 2\pi^2 \sin(\pi x^2) * \sin(\pi y). \end{aligned} \qquad \text{(8.39)}$$

Podemos escribir un programa en MATLAB para resolver el sistema (8.6), (8.38). En la Figura (8.5) mostramos los resultados para $n = 40$, $m = 40$. Note que en este caso, el sistema (8.6), (8.38) tiene matriz de coeficientes de dimensión 1521×1521 pero con muchas entradas cero. Al momento de resolver el sistema (8.6), (8.38) utilizamos las ecuaciones (8.37) para aprovechar la estructura de ceros de la matriz de coeficientes. □

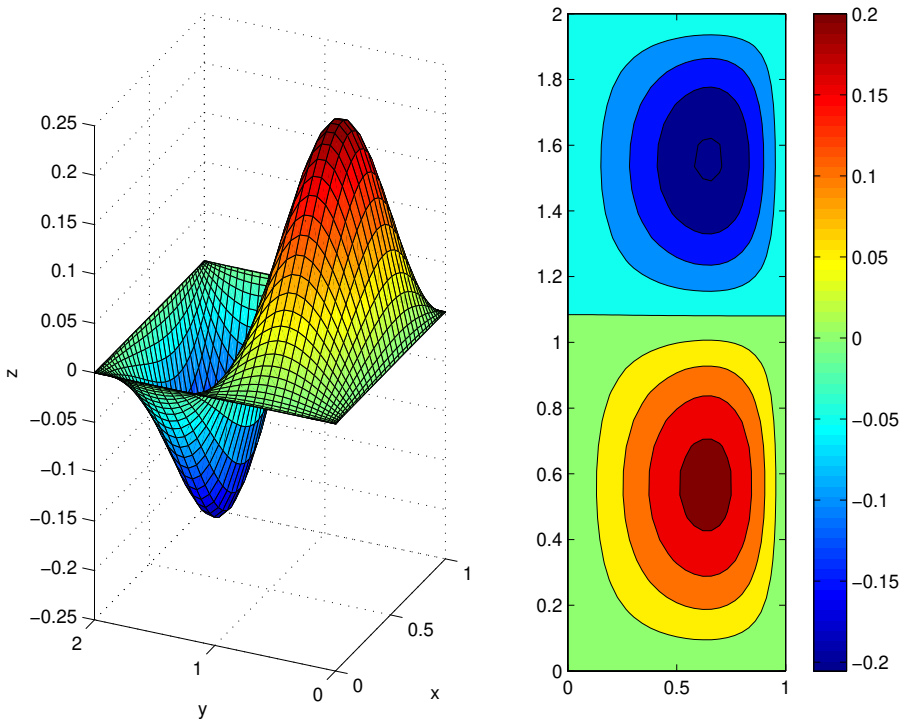


Figura 8.5: Solución aproximada del problema (8.25), (8.39) obtenida resolviendo el sistema (8.6), (8.38).

8.4 Problemas Nolineales

El método de elementos finitos puede utilizarse para resolver problemas de frontera nolineales. Estudiaremos problemas que pueden obtenerse a partir de un principio variacional. Esto es, suponemos que dado un conjunto $\Omega \subset \mathbb{R}^2$, y funciones $W : \Omega \times \mathbb{R} \times \mathbb{R}^2 \longrightarrow \mathbb{R}$ y $f : \Omega \longrightarrow \mathbb{R}$, queremos hallar una función $u : \Omega \longrightarrow \mathbb{R}$ que minimice la integral

$$I(u) = \int_\Omega [W(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})) - f(\mathbf{x})u(\mathbf{x})] \, \mathrm{d}\mathbf{x}, \tag{8.40}$$

donde $\mathbf{x} = (x, y)$. En adición se podría requerir que la función u satisfaga algún tipo de condición de frontera como la de tipo Dirichlet donde $u = g$ sobre $\partial\Omega$, siendo g otra función dada o especificada, o algún tipo de condición de periodicidad.

Si $u, v \in C^1(\bar{\Omega})$, entonces[4] la *primera variación de* (8.40) *en* u *en la dirección de* v es:

$$\begin{aligned}\delta I(u)[v] &\equiv \frac{\mathrm{d}}{\mathrm{d}\varepsilon} I(u+\varepsilon v)\Big|_{\varepsilon=0} = \int_\Omega \left[\frac{\partial W}{\partial u}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}))v(\mathbf{x})\right. \\ &\quad \left. + \frac{\partial W}{\partial \mathbf{p}}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})) \cdot \nabla v(\mathbf{x}) - f(\mathbf{x})v(\mathbf{x})\right] \mathrm{d}\mathbf{x}, \end{aligned} \tag{8.41}$$

donde los argumentos de W se etiquetan por $(\mathbf{x}, u, \mathbf{p})$. Si u es un mínimo (local) de (8.40), entonces es necesario que

$$\delta I(u)[v] = 0, \quad \forall\, v \in C^1(\bar{\Omega}). \tag{8.42}$$

Nuevamente, suponiendo que $u \in C^1(\bar{\Omega})$, usando una variación de la identidad de Green, tenemos que u satisface la ecuación diferencial de Euler–Lagrange:

$$-\mathrm{div}\left[\frac{\partial W}{\partial \mathbf{p}}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}))\right] + \frac{\partial W}{\partial u}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{8.43}$$

Esta ecuación diferencial que es en general nolineal en u, junto con la condición de frontera sobre u, forman el problema de frontera asociado al problema de minimizar el funcional (8.40).

Si $u, v, w \in C^1(\bar{\Omega})$, entonces la *segunda variación de* (8.40) *en* u *en las direcciones de* v, w es:

$$\begin{aligned}\delta^2 I(u)[v]\cdot w &\equiv \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\delta I(u+\varepsilon w)[v]\Big|_{\varepsilon=0} \\ &= \int_\Omega \left[\frac{\partial^2 W}{\partial u^2}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}))v(\mathbf{x})w(\mathbf{x})\right. \\ &\quad + \frac{\partial^2 W}{\partial u \partial \mathbf{p}}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})) \cdot (w(\mathbf{x})\nabla v(\mathbf{x}) + v(\mathbf{x})\nabla w(\mathbf{x})) \\ &\quad \left. + \left(\frac{\partial^2 W}{\partial \mathbf{p}^2}(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})) \cdot \nabla v(\mathbf{x})\right) \cdot \nabla w(\mathbf{x})\right] \mathrm{d}\mathbf{x}. \end{aligned} \tag{8.44}$$

Si u cumple con la condición necesaria de primer orden (8.42), entonces es necesario que:

$$\delta^2 I(u)[v]\cdot v \geq 0, \quad \forall\, v \in C^1(\bar{\Omega}). \tag{8.45}$$

No obstante, es bien conocido que esta condición no es suficiente para un mínimo (local) de (8.40).

[4] Aquí $v = 0$ sobre $\partial\Omega$ para la condición de frontera tipo Dirichlet en u, o tiene la misma periodicidad de u si este es el tipo de condición de frontera. En cualquier caso, suponemos que los términos de frontera en el computo de la primera variación, son cero.

Denotamos por

$$L(u)[v] = \delta I(u)[v], \tag{8.46}$$
$$B(u)[v,w] = \delta^2 I(u)[v] \cdot w. \tag{8.47}$$

Note que para cada función u fija, $L(u)[\cdot]$ es un funcional lineal y $B(u)[\cdot,\cdot]$ es una forma bilineal.

Vamos ahora a describir un método de elementos finitos para aproximar un mínimo de (8.40). Este método se basa en el método de Newton para aproximar mínimos de funciones de varias variables. Recuerde que si $F : \mathbb{R}^n \longrightarrow \mathbb{R}$, y queremos aproximar una solución de:

$$\min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}),$$

entonces el método de Newton está dado por la iteración:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - F''(\mathbf{x}_k)^{-1}\nabla F(\mathbf{x}_k), \quad k = 0, 1, 2, \ldots, \tag{8.48}$$

donde $\mathbf{x}_0$ es una aproximación inicial del mínimo buscado. Aquí $F''(\mathbf{x})$ es la matriz *Hessiana* de las segundas derivadas parciales de F. Esta matriz es simétrica si $F \in C^2(\mathbb{R}^n)$.

Para el problema de minimizar (8.40), identificamos a F con I en el método de Newton (8.48), $\nabla F(\mathbf{x})$ con $L(u)[\cdot]$, y $F''(\mathbf{x})$ con $B(u)[\cdot,\cdot]$. Escribimos $H = H_0^1(\Omega)$ si las condiciones de frontera en u son de tipo Dirichlet, y si son periódicas, entonces $H = H^1(\Omega)$ con la misma periodicidad requerida en u. El procedimiento queda como sigue:

1) Sea u_0 una aproximación inicial del mínimo de (8.40).

2) Para $k = 0, 1, 2, \ldots,$

   i) Calcule $w_k \in H$ tal que

   $$B(u_k)[w_k, v] = L(u_k)[v], \quad \forall\, v \in H.$$

   ii) Ponga $u_{k+1} = u_k - w_k$.

3) Repita el paso (2) hasta que el tamaño de w_k sea suficientemente pequeño.

En la implantación de este algoritmo estaremos utilizando el programado de dominio público FreeFem++ (cf. [9]) para elementos finitos. Este paquete además de que automatiza los procedimientos de creaciones de particiones, espacios de funciones para las aproximaciones, y definiciones de formas débiles y lados derechos,

permite trabajar con dominios más generales que los dominios rectangulares que hemos estado considerando hasta ahora. La codificación en FreeFem++ se asemeja mucho al procedimiento matemático de construir la forma débil del problema de frontera en cuestión. Para ilustrar algunos de los aspectos de este paquete, vamos a mostrar como se trabajaría el problema del Ejemplo 8.5 utilizando FreeFem++. El siguiente programa en FreeFem++ produce esencialmente los mismos resultados que vimos en el Ejemplo 8.5:

```
mesh Th = square(40,40,[x,2*y]);
fespace Vh(Th,P1);
Vh u,v;
func g=100*x+y^3;
func f= 2*pi^2*sin(pi*x^2)*sin(pi*y);
solve Poisson(u,v,solver=LU)=
          int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v)+g*u*v)
        - int2d(Th)( f*v)+on(1,2,3,4,u=0);
plot(u);
```

Las instrucciones son prácticamente auto explicativas: la primera instrucción crea una partición 40×40 de $[0,1] \times [0,2]$ llamada `Th`; la segunda instrucción define a `Vh` como el espacio de funciones continuas y de grado uno por pedazos; la tercera instrucción define a `u,v` como funciones en `Vh`; la cuarta y quinta instrucciones definen las funciones "exactas" `g` y `f`; la sexta instrucción resuelve (8.31) para `u` que representa u_h; y la última instrucción traza la solución calculada.

Vamos ahora a trabajar un caso especial del problema de minimizar (8.40), ó lo mismo, calcular una solución de la ecuación diferencial (8.43).

Ejemplo 8.6. Considere el caso en que

$$W(\mathbf{x}, u, \mathbf{p}) = \frac{1}{2}\, k(\mathbf{x}, u)\, \mathbf{p} \cdot \mathbf{p}.$$

para alguna función $k : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}$. Entonces (8.41), (8.44), (8.46), (8.47) reducen a:

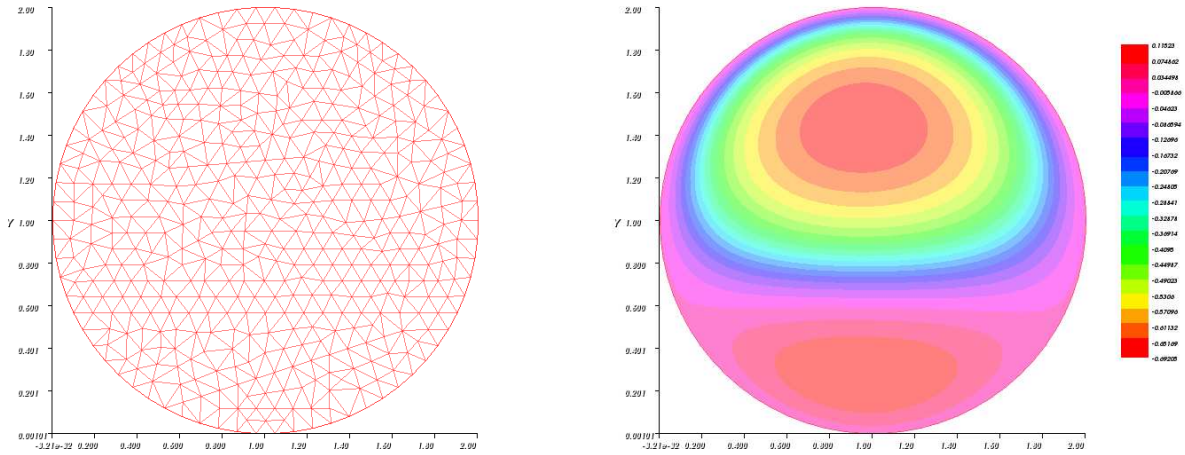
$$\begin{aligned} L(u)[v] &= \int_\Omega \left[k\, \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) + \frac{1}{2}\, k'\, [\nabla u(\mathbf{x}) \cdot \nabla u(\mathbf{x})]\, v(\mathbf{x}) - f(\mathbf{x}) v(\mathbf{x}) \right] \mathrm{d}\mathbf{x}, \\ B(u)[v,w] &= \int_\Omega \Big[k'\, \nabla u(\mathbf{x}) \cdot (w(\mathbf{x}) \nabla v(\mathbf{x}) + v(\mathbf{x}) \nabla w(\mathbf{x})) \\ &\quad + \frac{1}{2}\, k''\, [\nabla u(\mathbf{x}) \cdot \nabla u(\mathbf{x})]\, v(\mathbf{x}) w(\mathbf{x}) + k\, \nabla w(\mathbf{x}) \cdot \nabla v(\mathbf{x}) \Big] \mathrm{d}\mathbf{x}, \end{aligned}$$

donde las derivadas de k son con respecto a u y los argumentos de k y sus derivadas son $(\mathbf{x}, u(\mathbf{x}))$. Tomamos ahora Ω como el disco de radio uno y con centro en $(1,1)$, y las funciones:

$$\begin{aligned} k(\mathbf{x},u) &\equiv k(u) = u^2 + 0.5u + 0.1, \\ f(x,y) &= \sin\left(x+\frac{\pi}{4}\right)\cos\left(y+\frac{\pi}{4}\right), \quad (x,y)\in\Omega, \end{aligned}$$

y condición de frontera $u = 0$ sobre $\partial\Omega$. Con el siguiente código de FreeFem++ calculamos una solución mínima aproximada del funcional (8.40):

```
macro k(u) (u^2+0.5*u+0.1)//
macro kp(u) (2*u+0.5)//
macro kpp(u) (2.0)//

real tolerancia=1.0e-5;
int itermax=100;

border C(t=0,2*pi){x=1+cos(t); y=1+sin(t);};
mesh Th=buildmesh(C(70));
plot(Th,wait=true,ps="Th.eps");

fespace Vh(Th,P2);
Vh u,v,w;

func f=sin(x+pi/4.)*cos(y+pi/4.);

u=0;
problem Nlin(w,v,solver=LU)=int2d(Th)
        (kp(u)*(dx(u)*(w*dx(v)+v*dx(w))+dy(u)*(w*dy(v)+v*dy(w)))+
        0.5*kpp(u)*(dx(u)*dx(u)+dy(u)*dy(u))*v*w+
        k(u)*(dx(w)*dx(v)+dy(w)*dy(v)))-
        int2d(Th) (k(u)*(dx(u)*dx(v)+dy(u)*dy(v))+
        0.5*kp(u)*(dx(u)*dx(u)+dy(u)*dy(u))*v-f*v)+on(C,w=0);
for (int i=0;i<itermax;i++) {
        Nlin;
        u[]-=(1-0.50^(i+1))*w[];
        real residual=sqrt(w[]'*w[]);
        cout<<"Loop="<<i<<",  Residual="<<residual<<endl;
        if (residual<tolerancia) break;
}
plot(u,wait=false,fill=true,value=true,ps="soln.eps");
```



(a) Mallado o partición de Ω.

(b) Aproximación calculada.

Figura 8.6: Mallado y solución calculada del problema de minimizar (8.40) utilizando el método de Newton combinado con otro de elementos finitos.

El programa es una implementación del algoritmo descrito anteriormente basado en el método de Newton y que utiliza el método de elementos finitos para resolver el sistema lineal del paso (2i) en cada iteración. La instrucción `problem` se usa para definir la diferencia $B(u_k)[w_k, v] - L(u_k)[v]$ en el paso (2i) con el nombre `Nlin`. El problema del paso (2i) se resuelve al invocar el nombre `Nlin` dentro del ciclo que marca las iteraciones del método de Newton. El mallado o partición de Ω se muestra en la Figura 8.6a y la solución calculada en la Figura 8.6b. □

8.5 Ejercicios

Ejercicio 8.1. ¿Para cuales valores de k las funciones $f(x) = \operatorname{sen}(x)$, $g(x) = |x|$, y $h(x) = \operatorname{sgn}(x)$ pertenecen a $H^k(-1, 1)$? **Nota:** La función sgn está definida por

$$\operatorname{sgn}(x) = \begin{cases} 1 & , \quad x \geq 0, \\ -1 & , \quad x < 0. \end{cases}$$

Ejercicio 8.2. ¿Para cuales valores de k y α la función

$$f(x) = \frac{1}{\|x\|^{\alpha}},$$

pertenece a $H^k(\Omega)$ donde $\Omega = \{x \in \mathbb{R}^d \, : \, \|x\| < 1\}$, $1 \leq d \leq 3$?

Ejercicio 8.3. Sea Ω un disco en $\mathbb{R}^2$ con centro en el origen y radio $\frac{1}{e}$. Verifique que la función

$$u(x,y) = \ln\left[-\ln\left(\sqrt{x^2+y^2}\right)\right],$$

pertenece a $H_0^1(\Omega)$ pero no es continua en $(x,y) = (0,0)$.

Ejercicio 8.4. Determine la formulación débil, esto es, forma bilineal, funcional lineal, y los espacios de funciones, para los siguientes problemas de frontera.

a) Para $f \in L_2(\Omega)$, $g \in H^1(\Omega)$ y κ una función acotada, encuentre u tal que

$$\begin{cases} -\text{div}\ (\kappa \nabla u) = f & , \quad \text{en } \Omega, \\ u = g & , \quad \text{sobre } \partial\Omega. \end{cases}$$

b) Para $f \in L_2(\Omega)$, $g \in C^2(\partial\Omega)$, y $\beta > 0$ un numero real, encuentre u tal que

$$\begin{cases} -\Delta u = f & , \quad \text{en } \Omega, \\ \frac{\partial u}{\partial \mathbf{n}} + \beta u = g & , \quad \text{sobre } \partial\Omega. \end{cases}$$

Aquí $\mathbf{n}$ es la normal unitaria a $\partial\Omega$ y $\frac{\partial u}{\partial \mathbf{n}} = \nabla u \cdot \mathbf{n}$.

Ejercicio 8.5. Considere el problema de frontera

$$\begin{aligned} -\nu u''(x) + \beta u'(x) &= f(x), \quad 0 < x < 1, \\ u(0) = 0, \qquad & u(1) = 0, \end{aligned}$$

donde ν y β son números reales positivos. Determine la formulación débil de este problema. Modifique los programa de la Sección 8.3.1 y resuelva el problema con el método de FEM resultante. Trabaje primero con ν fijo y valores de β que aumenten progresivamente, y luego con β fijo y valores de ν progresivamente pequeños. Tome nota del comportamiento de la solución en cada caso.

Ejercicio 8.6. Utilizando las instrucciones `border` y `buldmesh` de FreeFemm++ podemos generar y triangular regiones bien complicadas en el plano. Por ejemplo, las siguientes instrucciones generan una región cuadrada con lados de largo uno, con un hueco circular con centro $(0.5, 0.5)$ y radio 0.1:

```
real r=0.1;
int nold=cout.precision(16);

border C1(t=0,1)x=t; y=0;label=1;
border C2(t=0,1)x=1; y=t;label=2;
border C3(t=0,1)x=1-t; y=1;label=3;
```

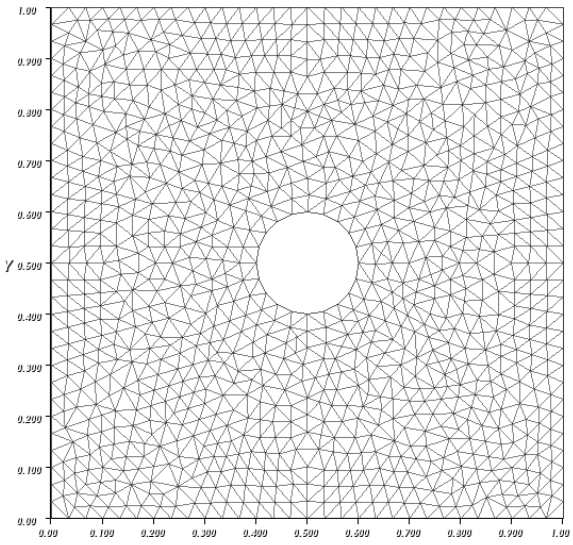


Figura 8.7: Región con un hueco para el Ejerecicio 8.6.

```
border C4(t=0,1)x=0; y=1-t;label=4;
border C5(t=0,2*pi) x=0.5+r*cos(t); y=0.5+r*sin(t);label=5;;

mesh Th = buildmesh (C1(30)+C2(30)+C3(30)+C4(30)+C5(-20));
plot(Th);
```

Escriba un programa en FreeFem++ para aproximar una solución del problema

$$\begin{cases} -\Delta u = f & , \text{ en } \Omega, \\ u = g_1 & , \text{ sobre } \Gamma_1, \\ u = g_2 & , \text{ sobre } \Gamma_2, \end{cases}$$

donde Γ_1 es el borde o frontera del cuadrado y Γ_2 es el borde del circulo interno. Utilice

$$f(x,y) = \text{ sen}\,(5x+y), \quad g_1 = 0, \quad g_2 = x.$$

Ejercicio 8.7. Si en el Ejercicio 8.6 cambiamos la instrucción que crea la triangulación a:

```
mesh Th = buildmesh (C1(30)+C2(30)+C3(30)+C4(30)+C5(20));
```

entonces la región que resulta no tiene huecos pero esta "dividida" en dos regiones: la del Ejercicio 8.6 que llamaremos Ω_1, unión con la región del circulo interno que

llamaremos Ω_2. Parecería que solo tenemos un cuadrado, que es así, pero con dos subregiones bien definidas. Esto lo podemos utilizar para resolver problemas con coeficientes que estan definidos de formas diferentes en cada subregión. Usando esto escriba un programa en FreeFem++ para aproximar una solución del problema

$$\begin{cases} -\text{div}\ (\kappa \nabla u) = f & , \quad \text{en } \Omega, \\ u = g & , \quad \text{sobre } \partial\Omega, \end{cases}$$

donde para algunas constantes c_1, c_2, tenemos que

$$\kappa(x,y) = \begin{cases} c_1 & , \quad (x,y) \in \Omega_1, \\ c_2 & , \quad (x,y) \in \Omega_2. \end{cases}$$

Para definir κ en FreeFem++ necesitará las siguientes instrucciones (suponiendo que $c_1 = 100$ y $c_2 = -10$):

```
fespace Vhk(Th,P0);
Vhk reg=region, kappa;

int Omega1=reg(0.25,0.25);
int Omega2=reg(0.5,0.5);
real c1=100.0, c2=-10.0;

kappa=(region==Omega1)*c1+(region==Omega2)*c2;
```

Apéndice A

Resultados Básicos del Cálculo

En este apéndice vamos a repasar algunos resultados del calculo elemental que utilizaremos frecuentemente en nuestras discusiones. El primer resultado básico es el *Teorema del Valor Medio* para funciones. En palabras y en el contexto físico del movimiento de una partícula, este teorema establece que en cualquier intervalo finito de tiempo, la velocidad promedio de la partícula en el intervalo es igual a la velocidad instantánea en algún instante de tiempo.

Teorema A.1 (del Valor Medio para funciones)**.** *Sea f una función continua en $[a, b]$, diferenciable en (a, b), donde a, b son finitos. Entonces existe un número $c \in (a, b)$ tal que*

$$f(b) - f(a) = f'(c)(b - a). \tag{A.1}$$

También tenemos una versión del Teorema del valor medio para integrales:

Teorema A.2 (del Valor Medio para Integrales)**.** *Suponga que f y g son funciones continuas en el intervalo acotado $[a, b]$ y que $g(x) \geq 0$ para toda $x \in [a, b]$. Entonces existe un punto $c \in [a, b]$ tal que*

$$\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx. \tag{A.2}$$

En el caso de que $g \equiv 1$, el resultado se puede interpretar como que el área bajo la curva $f(x)$ en el intervalo $[a, b]$ es igual al área del rectángulo con base $[a, b]$ y altura $f(c)$.

El próximo teorema expone de forma rigurosa la idea intuitiva de que, si una función continua cambia de signo en un intervalo $[a, b]$, entonces en algún punto del intervalo cruzó el eje de x.

Teorema A.3 (del Valor Intermedio)**.** *Sea f una función continua en $[a, b]$ y C un número estrictamente entre $f(a)$ y $f(b)$. Entonces existe un número $c \in (a, b)$ tal que $f(c) = C$. En particular si $f(a)f(b) < 0$, entonces existe $c \in (a, b)$ tal que $f(c) = 0$.*

Bibliografía

[1] T. M. Apostol. *Mathematical Analysis: A Modern Approach to Advanced Calculus*. Addison-Wesley, Reading, 2nd edition, 1974.

[2] K. Atkinson. *An Introduction to Numerical Analysis*. John Wiley and Sons, Inc., New York, 2nd edition, 1989.

[3] K. Atkinson. *Elementary Numerical Analysis*. John Wiley and Sons, Inc., New York, 2nd edition, 1993.

[4] F. Bauer, H. Rutishauser, and E. Stiefel. New aspects in numerical quadrature. In *Experimental Arithmetic, High Speed Computing, and Mathematics*, pages 199–218. Amer. Math. Soc., Providence, R. I., 1963.

[5] J. H. Bramble and S. R. Hilbert. Estimation of linear functionals on sobolev spaces with application to fourier transforms and spline interpolation. *SIAM J. Numer. Anal.*, 7:112–124, 1970.

[6] R. Burden and J. Faires. *Numerical Analysis*. Prindle, Weber, and Schmidt, Boston, 4th edition, 1989.

[7] A. Edelman. The mathematics of the pentium division bug. *SIAM Review*, 39(1):54–67, 1997.

[8] C. W. Gear. *Numerical Initial–Value Problems in Ordinary Differential Equations*. Englewood Cliffs, New Jersey, 1971.

[9] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3–4):251–265, 2012.

[10] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley, New York, 1962.

[11] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*. John Wiley and Sons, New York, 1966.

[12] H. B. Keller. *Numerical Methods for Two Point Boundary Value Problems*. Dover Publications, Inc., New York, 1992.

[13] S.J. Leon. *Linear Algebra with Applications*. Prentice Hall, New Jersey, 5th edition, 1998.

[14] T. Y. Li and T. Sauer. Regularity results for solving systems of polynomials by homotopy method. *Numer. Math.*, 50:283–289, 1987.

[15] T. Y. Li and J. A. Yorke. The random product homotopy and deficient polinomial systems. *Numer. Math.*, 51:481–500, 1987.

[16] C. Moler. Floating points. *MATLAB News and Notes: Cleve's Corner*, Fall 1996.

[17] C. Moler. A tale of two numbers. *MATLAB News and Notes: Cleve's Corner*, Winter 1995.

[18] D. E. Muller. A method for solving algebraic equations using an automatic computer. *Mathematical Tables and Other Aids to Computation*, 10:208–215, 1956.

[19] M. A. Munen and J. P. Yizze. *Precalculus: Functions and Graphs*. Worth Publishers, Inc., new York, fourth edition, 1985.

[20] Y. Nievergelt. Rounding errors to knock your stocks off. *Mathematics Magazine*, 73(1), 2000.

[21] J. M. Ortega. *Numerical Analysis: A Second Course*. Classics in Applied Mathematics. SIAM, Philadelphia, 1990.

[22] M. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM, Philadelphia, 2001.

[23] W. Rheinboldt. *Methods for Solving Systems of Nonlinear Equations*. SIAM, Philadelphia, 1974.

[24] K. Ross. *Elementary Analysis: The Theory of Calculus*. Springer-Verlag, New York, 1980.

[25] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 3rd edition, 1976.

[26] L. Shampine and M. Gordon. *Computer Solution of Ordinary Differential Equations*. Freeman, San Francisco, 1975.

[27] R. Skeel. Roundoff error cripples patriot missile. *SIAM News*, 25(4):11, 1992.

[28] G. Stewart. *Introduction to Matrix Computations*. Academic Press, New York, 1973.

[29] J.R. Taylor. *An introduction to error analysis: the study of uncertainties in physical measurements*. University Science Books, 2nd edition, 1997.

[30] J. F. Traub. *Iterative Methods for the Solution of Equations*. Prentice-Hall, Englewood Cliffs, N. J., 1964.

[31] L. Trefethen. *Spectral Methods in MATLAB*. SIAM, Philadelphia, 2000.

[32] D. Weber-Wulff. Rounding error changes parliament makeup. *The Risks Digest*, 13(37), 1992.

[33] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, New York, 6th edition, 2005.

Índice

Índice de Figuras