Métodos numéricos con Python

DIEGO ARÉVALO OVALLE, MIGUÉL ÁNGEL BERNAL YERMANOS, JAIME ANDRÉS POSADA RESTREPO

POLI
POLITÉCNICO GRANCOLOMBIANO
INSTITUCIÓN UNIVERSITARIA

Métodos numéricos con Python

Métodos numéricos con Python

Diego Arévalo Ovalle

Miguel Ángel Bernal Yermanos

Jaime Andrés Posada Restrepo

Bogotá, octubre de 2021

© Institución Universitaria Politécnico Grancolombiano

Métodos numéricos con `Python`.
ISBN: 978–958–5142–97–8
E-ISBN: 978–958–5142–98–5
DIGITAL ISBN: 978–958–5142–99–2

Editorial Politécnico Grancolombiano
PBX: 7455555 Ext. 1171
`editorial@poligran.edu.co`

Octubre de 2021
Bogotá, Colombia

Institución Universitaria Politécnico Grancolombiano

Autores
Diego Arévalo Ovalle
Miguel Ángel Bernal Yermanos
Jaime Andrés Posada Restrepo

Director editorial
Eduardo Norman Acevedo

Analista de producción editorial
Carlos Eduardo Daza Orozco

Corrección de estilo
Hernán Darío Cadena

Armada electrónica
Jaime Andrés Posada Restrepo

Impresión y encuadernación
Xpress Estudio Gráfico y Digital

Impreso y hecho en Colombia
Printed in Colombia

Arévalo Ovalle, Diego
Métodos numéricos con `Python`. / Diego Arévalo Ovalle, Miguel Ángel Bernal Yermanos y Jaime Andrés Posada Restrepo; director editorial, Eduardo Norman Acevedo. – Bogotá D.C.: Editorial Politécnico Grancolombiano, 2021.

158 p.: il.; 17 × 24 cm.
Incluye referencias bibliográficas.
ISBN: 978–958–5142–97–8

1. INGENIERÍA – ANÁLISIS NUMÉRICO. – 2. MATEMÁTICAS PARA INGENIEROS – 3. PYTHON (LENGUAJE DE PROGRAMACIÓN DE COMPUTADORES). – 4. INTERPOLACIÓN (MATEMÁTICAS). – 5. ECUACIONES DIFERENCIALES. – 6. ALGORITMOS. – 7. ANÁLISIS MATÉMATICO

511 A678m 21 Ed.

Sistema Nacional de Bibliotecas – SISNAB
Institución Universitaria Politécnico Grancolombiano

¿Cómo citar este título?

Arévalo Ovalle, D., Bernal Yermanos, M. A., & Posada Restrepo, J. A. (2021), Métodos numéricos con `Python`, Bogotá: Editorial Politécnico Grancolombiano.

La Editorial del Politécnico Grancolombiano pertenece a la Asociación de Editoriales Universitarias de Colombia, ASEUC.

El contenido de esta publicación se puede citar o reproducir con propósitos académicos siempre y cuando se dé cuenta de la fuente o procedencia. Las opiniones expresadas son responsabilidad exclusiva del autor.

Prefacio

Este texto es el resultado de nuestra labor como orientadores del curso de métodos numéricos en la *Institución Universitaria Politécnico Grancolombiano* durante varios años. Aunque inicialmente solamente se disponía de notas de clase construidas de manera informal para los cursos, con el tiempo surgió la necesidad de consolidarlas en un sólo documento.

El libro contiene métodos computacionales para resolver problemas esenciales en el campo de ingeniería o matemática aplicada. En cada método se pretende dar al lector una visión de su esencia, necesidad, ventajas y desventajas. En algunos casos, alejándose de presentaciones rigurosas, pero sin dejar de ser correctas. El objetivo final es proporcionar los elementos necesarios para la aplicación adecuada de los algoritmos.

Adicionalmente, se tiene material sobre medidas de error, orden de convergencia, notación O y un capítulo dedicado a diferenciación e integración numérica. Se presenta la extrapolación de Richardson y esquemas de integración numérica entre los que se tienen trapecios y Simpson, entre otros.

Por otro lado, al momento de escribir una versión preliminar del texto, se utilizó `Python 2` para la implementación de los algoritmos. Dado el retiro de dicho lenguaje el primero de enero de 2020, los algoritmos del texto se han migrado completamente de `Python 2` a `Python 3`.

También se han corregido errores tipográficos, de notación, y la exposición de algunos temas se ha modificado para transmitir más efectivamente sus contenidos.

Los requisitos para acceder al material no se han alterado. Solamente se asume familiaridad del lector con temas de cálculo y álgebra lineal, entre los cuales se encuentran: teorema del valor intermedio, teorema del valor medio, teorema de Taylor, teorema fundamental del cálculo, eliminación gaussiana y regla de Cramer, entre otros.

Por último, agradecemos nuevamente a los estudiantes y compañeros que nos han brindado apoyo ya sea con peticiones, observaciones, quejas, reclamos o halagos acerca de los contenidos y recursos del texto y su correspondiente exposición en clase.

Índice general

1. Errores y aproximaciones **13**

1.1. Introducción 13

1.2. Cifras significativas 14

1.3. Error de truncamiento 16

1.4. Error de redondeo 19

1.5. Orden de convergencia 19

1.6. Notación O 20

2. Búsqueda de raíces **23**

2.1. Introducción 23

2.2. Método de bisección 25

2.2.1. Criterios de parada 29

2.2.2. Método de *regula falsi* 29

2.3. Método de Newton-Raphson 32

2.3.1. Método de la secante 33

2.4. Método de punto fijo 36

2.5. Orden de convergencia en el método de punto fijo 43

2.6. Orden del método de Newton-Raphson 45

3. Interpolación **47**

3.1. Introducción 47

3.2. Ajuste exacto 48

3.2.1. Polinomio de interpolación de Lagrange 48

3.2.2. Polinomio de interpolación de Newton 52
3.2.3. Trazadores cúbicos 60
3.3. Ajuste por mínimos cuadrados 65
3.3.1. Errores 65
3.3.2. Funciones de ajuste 66
3.3.3. Polinomios de mínimos cuadrados 67
3.3.4. Ajuste exponencial 70

4. Diferenciación e integración numérica 73
4.1. Introducción 73
4.2. Aproximaciones a la derivada 74
4.3. Extrapolación de Richardson 76
4.4. Aproximaciones a la integral definida 78
4.5. Regla de los trapecios 79
4.6. Regla de Simpson 81

5. Sistemas de ecuaciones 85
5.1. Introducción 85
5.2. Métodos directos 85
5.2.1. Factorización *LU* 89
5.3. Métodos iterativos 93
5.3.1. Normas vectoriales 94
5.3.2. Normas matriciales 96
5.3.3. Solución de sistemas de ecuaciones 98
5.3.4. Método de Jacobi 99
5.3.5. Método de Gauss-Seidel 102
5.3.6. Convergencia métodos de Jacobi y Gauss-Seidel 106

6. Ecuaciones diferenciales 111
6.1. Introducción 111
6.2. Problemas de valor inicial 112
6.2.1. Método de Euler 113

6.2.2. Orden del método de Euler 116
6.2.3. Método de Verlet 118
6.2.4. Error del método de Verlet 119
6.2.5. Métodos de Runge-Kutta orden dos 121
6.2.6. Método de Runge-Kutta orden cuatro (RK4) 127

A. Tutorial de `Python` 131

A.1. Generalidades 131
A.2. Funciones 131
A.3. Estructuras básicas de control 132
A.3.1. `if` 132
A.3.2. `while` 132
A.3.3. `for` 132
A.4. Ejemplos 133
A.4.1. Factorial 133
A.4.2. GCD 133
A.4.3. ¿Es palíndromo? 134
A.4.4. NumPy 134

B. Compendio de programas 137

B.1. Búsqueda de raíces 137
B.1.1. Bisección 137
B.1.2. *Regula falsi* 140
B.1.3. Newton 143
B.1.4. Secante 146
B.1.5. Punto fijo 148
B.2. Interpolación 151
B.2.1. Lagrange 151
B.2.2. Diferencias divididas de Newton 152
B.2.3. Trazadores cúbicos 154
B.2.4. Recta de ajuste mínimos cuadrados 157

B.3. Diferenciación e integración numérica 159
B.3.1. Extrapolación de Richardson 159
B.3.2. Regla del trapecio 160
B.3.3. Regla de Simpson 162
B.4. Sistemas de ecuaciones 164
B.4.1. *LU* 164
B.4.2. Jacobi 166
B.4.3. Gauss-Seidel 170
B.5. Ecuaciones diferenciales 173
B.5.1. Euler 173
B.5.2. Verlet 175
B.5.3. RK4 178

Bibliografía **181**

Ulevellen (de enige echte!!!)

Capítulo 1

Errores y aproximaciones

1.1. Introducción

Las cantidades numéricas que se trabajan en ingeniería y ciencias, provienen de dos fuentes principalmente, de mediciones y obtenidas por procesos de cálculos matemáticos.

En la primera fuente, se debe considerar que un proceso de medición nunca arroja el valor verdadero de la medida, pues las mediciones se hacen con instrumentos que se encuentran limitados por la división más pequeña de la unidad de medida en la que vienen graduados, obteniendo entonces una aproximación al valor verdadero hasta la incertidumbre del instrumento.

En el segundo caso, las operaciones aritméticas usualmente se ejecutan en un computador o una calculadora, y la limitación de espacio de estos dispositivos lleva a que algunas cifras no se puedan representar correctamente. En dicho caso, lo máximo que se puede esperar es una aproximación de los valores verdaderos. La situación se agrava si estas aproximaciones se siguen utilizando en posteriores cálculos. De otro lado, muchas veces resulta imposible[1] siquiera expresar en términos algebraicos soluciones a ciertas ecuaciones, inclusive polinomiales. En esta situación, no quedaría otro remedio que buscar aproximaciones.

Con lo anterior, es claro que el trabajo de ingeniería y ciencias se encuentra inevitablemente sujeto a error, y por tanto el tema de cuantificación de errores es de atención prioritaria en dichas áreas. Hay dos maneras clásicas de medir errores, mediante el *error absoluto* y el *error relativo*. Para definir lo anterior, suponer que $\bar{x}$ es una aproximación al valor verdadero x. El error absoluto es $\varepsilon = |x - \bar{x}|$, mientras que el error relativo es $\varepsilon_r = \dfrac{|x - \bar{x}|}{|x|}$ si $x \neq 0$, donde es necesario resaltar que el error absoluto puede ser algunas veces engañoso, pues no tiene en consideración el tamaño del valor, como si es el caso del error relativo.

[1] Es de resaltar el teorema de Abel-Ruffini al respecto.

Como se verá en posteriores capítulos, aunque no siempre es posible contar con el valor verdadero de una magnitud, con los procedimientos a trabajar se pueden ir encontrando aproximaciones que en cada paso resulten mejores, y por tanto, la distancia entre una aproximación y la anterior es cada vez menor. Lo anterior se puede ir visualizando a través del *error aproximado*, definido como $\varepsilon_{ap} = |x_{\text{actual}} - x_{\text{anterior}}|$. En la sección 1.2 se usa el concepto de error verdadero para formalizar lo que se entenderá como cifras significativas. De igual manera, en este capítulo, se incluye material que permite analizar si unas aproximaciones sucesivas resultantes de un método son convergentes y cuantificar la rapidez con que lo hacen. También será posible analizar comportamientos asintóticos de funciones y caracterizar términos de errores.

1.2. Cifras significativas

Considerar una región rectangular que tiene 20.5 cm de largo por 14.3 cm de ancho. Se requiere la medida del área de la región en centímetros cuadrados.

La solución del problema parece inmediata, ya que basta con multiplicar las dos mediciones para obtener 293.15 cm^2. Sin embargo, se encuentra en duda la validez de dicho resultado, pues no se ha considerado que las cantidades en ingeniería, aparte del valor de la medición, tienen implícita la incertidumbre asociada al instrumento de medición.

Volviendo al problema inicial, al considerar el valor 20.5 cm, se deduce que el instrumento con que fue tomada la medida tiene graduación en centímetros, y que cada centímetro está dividido en diez partes, por lo que cualquier medida con el instrumento, tendrá una precisión de a lo más una décima de centímetro.

En la figura 1.1, diferentes personas reportarían sin dudar los primeros dos dígitos, a saber el dos y el cero, que en este contexto se llamarán *dígitos confiables*. El tercero podría tener libre interpretación, pues algunas personas podrían decir que se trata de un seis, y otros de un cinco. En esta situación es preferible dejar la medida hasta la raya de división que parezca más cercana, en este caso la del cinco. Se tendría entonces la tranquilidad que la medición tiene una incertidumbre de una décima de centímetro, y se entenderá que la medida de 20.5 cm es en realidad una cifra entre 20.4 cm y 20.6 cm con el tercer dígito llamado *dígito de incertidumbre*. La cantidad de dígitos confiables más otro con incertidumbre, son las *cifras significativas* de una medición. Con lo anterior, las medidas 20.5 y 14.3 cm, tienen cada una tres cifras significativas.

Nota: también es posible dejar el cinco como una cifra confiable y aventurarse a tomar la mitad de la décima de centímetro, división más pequeña, como el dígito con incertidumbre. No es extraño entonces que alguien pueda reportar la misma medida como 20.55 cm. Lo anterior dejaría tres dígitos confiables y un último con incertidumbre. Sin embargo, para efectos prácticos, en este texto se trabaja como se indica con anterioridad.

Volviendo al ejemplo, notar que, al multiplicar las dos medidas para calcular el área, se obtiene un resultado (293.15 cm^2) con cinco cifras, lo que lleva a preguntarse ¿cómo es posible que trabajando con medidas que tienen tres cifras significativas, se logre una con

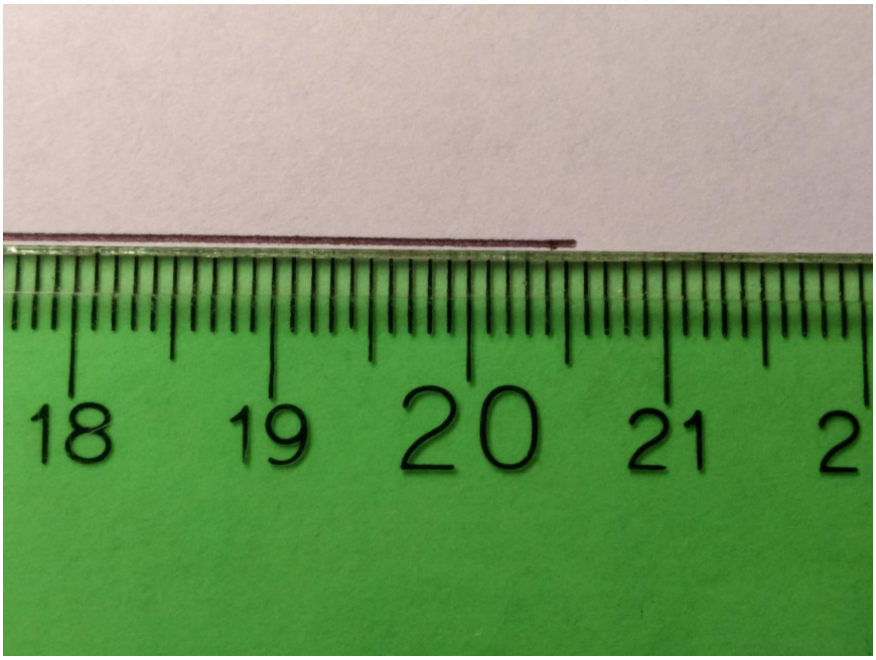


Figura 1.1: medida tomada con una regla casera graduada en centímetros y cada centímetro divido en diez partes.

cinco cifras significativas? A continuación, se tienen reglas que orientan lo que se debe hacer cuando se hacen operaciones con cifras significativas, y que responden al anterior interrogante.

Regla 1: cuando se multipliquen o dividan cifras significativas, el resultado debe tener tantas cifras como el factor que tenga menos cifras significativas.

Regla 2: cuando se sumen o resten cifras significativas, el resultado debe tener tantas cifras como el término que tenga menos cifras significativas.

Por lo tanto, de acuerdo a la primera regla, el resultado correcto del área es 293 cm^2. Es normal que la última cifra se redondee observando la cifra de la derecha. En este caso, como a la derecha del tres hay un uno, entonces sigue como tres, si hubiera un cinco o mayor, el tres se eleva en una unidad y quedaría como un cuatro.

De acuerdo a los instrumentos de medida, las características del sistema o propósitos de la información, el ingeniero debe hacer un estimativo del número de cifras significativas que requieren sus cálculos. Con lo anterior, es posible tener una cota del error absoluto ($\varepsilon = |x - \bar{x}|$) con base en el número n de cifras significativas mediante la expresión

$$\varepsilon = \left(0.5 \times 10^{2-n}\right) \%.$$

Ejercicios 1

1. Calcular el volumen en pies cúbicos de una habitación de 2.6 m de profundidad, 3.8 m de ancho, y 2.6 m de altura.

2. Se toma el pulso de una persona y se observa que 110 pulsaciones tardan 115.6 segundos. Calcular la duración de un pulso o latido.

3. Un caracol se mueve con una rapidez de 5.12 furlong por quincena. Considerando que un furlong equivale a 220 yardas, que un metro son 1.094 yardas y que una quincena son 14 días, expresar la rapidez en metros sobre segundos.

4. Una región rectangular mide 20.5 cm de largo por 14.3 cm de ancho. Calcular su perímetro.

5. Un cilindro de aluminio tiene un diámetro de 2.13 cm y 15.3 cm de longitud. Calcular a), su volumen en centímetros cúbicos, b), su masa en gramos (consultar la densidad del aluminio en una tabla) y c), la superficie total en centímetros cuadrados.

1.3. Error de truncamiento

Es conocido que números como π, e, $\sqrt{2}$ tienen un desarrollo decimal infinito y no periódico, y por tanto, es muy difícil para la mayoría de las personas retener en su memoria tan solo unas cuantas cifras decimales de ellos. Sin embargo, hay formas metódicas de obtener aproximaciones tan buenas como se quieran a estos números. Una de las más famosas es a través del uso de series de Taylor. Para ello, si se conoce una función f y todas sus derivadas en un punto c, entonces el valor de la función en otro punto x, se puede obtener usando la serie de potencias

$$f(x) = f(c) + f'(c)\,(x-c) + \frac{f''(c)}{2!}\,(x-c)^2 + \frac{f'''(c)}{3!}\,(x-c)^3 + \cdots, \qquad (1.1)$$

donde se requiere aparte de la existencia de las derivadas en el intervalo abierto entre x y c, también de la continuidad de f y sus derivadas en el intervalo cerrado entre x y c.

Si se empieza a evaluar el primer término de la serie, luego el segundo, el tercero y así sucesivamente, es posible que en situaciones prácticas llegue el momento en que se debe abandonar el cálculo de términos y por tanto truncar la serie. En este momento es deseable contar con una aproximación que pueda brindar el error cometido hasta el momento. Al cortar la serie en el término $n+1$, se logra una aproximación al valor de la función en x dada por el *polinomio de Taylor* $P_n(x)$ de orden n:

$$P_n(x) = f(c) + f'(c)\,(x-c) + \frac{f''(c)}{2!}\,(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}\,(x-c)^n.$$

Notar que, si se corta la serie en el primer término, se tiene la aproximación a orden cero, en el segundo, aproximación de primer orden y así sucesivamente. Lo interesante es que al hacer aproximaciones con la serie de Taylor, hay un indicio del error que se comete, pues se entenderá dicho error como proveniente de los términos que no se tuvieron en cuenta. En dicho caso, el error estaría dado por:

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-c)^{n+1} + \frac{f^{(n+2)}(c)}{(n+2)!}(x-c)^{n+2} + \cdots$$

Usando el teorema del valor medio, es posible reescribir la expresión anterior como

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}, \tag{1.2}$$

donde ξ se encuentra entre x y c. El resultado 1.2 se conoce como *error de truncamiento*, o término de error en forma de Lagrange. Se debe considerar que, aunque se garantiza la existencia de ξ, no se tiene, en general, una forma explícita de encontrar dicho valor. Con los elementos anteriores, la serie de Taylor 1.1 se suele escribir como

$$f(x) = P_n(x) + R_n(x). \tag{1.3}$$

Nota: en otras situaciones o contextos, es también conveniente trabajar con la *forma integral* del error de truncamiento o término de error, dada por

$$R_n(x) = \int_c^x \frac{f^{(n+1)}(t)}{n!}(x-t)^n dt. \tag{1.4}$$

Ejemplo 1. Calcular la serie de Taylor de $f(x) = \sin x$ alrededor de $c = 0$, el polinomio de orden dos, orden tres, y los correspondientes errores de truncamiento.

Solución: se comienza calculando los valores de la función y sus derivadas en $x = 0$:

- $f(0) = \sin x\Big|_{x=0} = 0.$
- $f'(0) = \cos x\Big|_{x=0} = 1.$
- $f''(0) = -\sin x\Big|_{x=0} = 0.$
- $f'''(0) = -\cos x\Big|_{x=0} = -1.$
- $f^{(4)}(0) = \sin x\Big|_{x=0} = 0.$

Para $f^{(5)}(0)$ en adelante, se tienen repeticiones de valores. Al reemplazar en la expresión 1.1 se obtiene la serie de Taylor de $\sin x$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{(2n+1)}}{(2n+1)!}.$$

Para obtener la aproximación de segundo orden, se corta la serie en el tercer término. Luego

$$P_2(x) = x.$$

Notar que el primer y tercer término de la serie valen cero y que el correspondiente término de error es

$$R_2(x) = -\frac{\cos \xi}{6} x^3,$$

donde ξ se encuentra entre cero y x. De forma similar, la aproximación de tercer orden es $P_3(x) = x - \dfrac{x^3}{6}$ y el término de error es $R_3(x) = \dfrac{\sin \xi}{24} x^4$ donde ξ, al igual que antes, se encuentra entre cero y x. ♢

Ejercicios 2

1. Calcular las series de Taylor alrededor de $c = 0$ para las siguientes funciones:

   a) $f(x) = \ln(x + 1)$.
   b) $f(x) = \tan^{-1} x$.
   c) $f(x) = \cos x$.
   d) $f(x) = e^x$.
   e) $f(x) = \ln(1 - x)$.
   f) $f(x) = \dfrac{1}{1 - x}$.
   g) $f(x) = \sinh x$.
   h) $f(x) = \cosh x$.
   i) $f(x) = x^2 e^{-3}$.
   j) $f(x) = x^5 - x^3 + 1$

2. Calcular la serie de Taylor alrededor de $c = 3$ para $f(x) = \dfrac{1}{x^2}$.

3. Calcular la serie de Taylor alrededor de $c = 2$ para $f(x) = -\ln x$.

4. Calcular la serie de Taylor alrededor de $c = 5$ para $f(x) = x^5 - 4x^3 + x^2 + x - 1$.

5. Obtener una aproximación a π usando el polinomio de tercer orden de Taylor de arcotangente y obtener el correspondiente error de truncamiento. Calcular la serie alrededor de $c = 0$.

6. Obtener un polinomio de grado tres que aproxime a e y calcular el error de truncamiento. Usar un desarrollo en serie de e^x alrededor de $c = 0$.

7. Considerar la función $f(x) = \dfrac{1}{1+x^2}$. Obtener el polinomio de cuarto orden para estimar el valor de $f(-0.25)$ y calcular el error de truncamiento. Calcular la serie alrededor de $c = 0$.

8. Obtener una aproximación a $\sqrt{2}$ con el polinomio de Taylor de cuarto orden de la función $f(x) = \sqrt{1+x}$. Calcular la serie alrededor de $c = \dfrac{1}{2}$.

1.4. Error de redondeo

Como se mencionó en la sección 1.3, hay aparición de error por la dificultad de las máquinas para representar ciertos números reales, particularmente números irracionales. En computación, debido a la forma en que los computadores almacenan los números reales usando *representación de punto flotante*, pueden ocurrir errores adicionales, en este caso de *redondeo*, debido a la misma naturaleza de dicha representación en que se usan aproximaciones para representar números reales.

Con las consideraciones anteriores, se define el error total como la suma del error de truncamiento con el error de redondeo, de donde en la práctica en ciencias e ingeniería, sería natural el querer minimizar el error total. Lo anterior puede llevar a una paradoja, pues si se quiere más precisión en las cifras, de acuerdo al error de truncamiento, se debe tomar más términos en las series de Taylor. Pero esto hace que se tenga que hacer más operaciones con números que ya son aproximaciones en las máquinas de cómputo, lo que lleva a que el error de redondeo pueda ser importante.

Al parecer, tener más operaciones tiende a incrementar el error de redondeo, pero menos operaciones lleva a mayor error de truncamiento. Por lo tanto, en el trabajo en ciencias e ingeniería, se debe balancear y encontrar un punto de equilibrio en el que los procedimientos lleven a resultados lo suficientemente satisfactorios, de tal forma que se puedan calcular con el menor número de pasos u operaciones posibles sin sacrificar la precisión y exactitud de los resultados.

1.5. Orden de convergencia

Cuando se utiliza un método iterativo, como los desarrollados en el capítulo 2, es útil conocer la velocidad de convergencia, constituyendo lo anterior un criterio para determinar la eficiencia de un procedimiento en la solución de un problema particular. Para entender este concepto y desarrollarlo en algunos casos especiales, es necesario presentar algunas definiciones.

Definición 1. *Sea $\{p_n\}_{n=0}^{\infty}$ una sucesión convergente, p el límite de la sucesión, $p_n \neq p$ para todo $n \in \mathbb{N}$. Si existen λ y α números reales positivos tales que*

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda,$$

entonces se dice que $\{p_n\}_{n=0}^{\infty}$ *es una sucesión que converge a* p *con orden* α *y constante de error asintótico* λ.

Ejemplo 2. La sucesión $\left\{\frac{1}{n}\right\}_{n=0}^{\infty}$ converge a cero con orden uno.

Solución: para comprobar que la sucesión $p_n = \frac{1}{n}$ converge a 0 con orden uno es necesario demostrar que el límite

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^1}$$

existe y es un número real positivo. En este caso,

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^1} = \lim_{n\to\infty} \frac{\left|\frac{1}{n+1} - 0\right|}{\left|\frac{1}{n} - 0\right|^1} = \lim_{n\to\infty} \frac{n}{n+1} = 1$$

y por tanto la convergencia de $\left\{\frac{1}{n}\right\}_{n=0}^{\infty}$ es de orden uno. ♢

Si el orden de convergencia de una sucesión es $\alpha = 1$, entonces se dirá que la sucesión es linealmente convergente, mientras que en el caso $\alpha = 2$, se habla de convergencia cuadrática. Como se podría esperar, entre más alto sea el orden de un método, debería presentarse una convergencia más rápida. Aunque las sucesiones de convergencia cuadrática alcanzan con mayor rapidez una precisión deseada en comparación con sucesiones de orden lineal, la mayoría de esquemas iterativos del capítulo 2 tienen una convergencia lineal como se verá en su momento.

1.6. Notación O

La notación O es usada ampliamente para describir el comportamiento asintótico de funciones o sucesiones. En el contexto de los métodos numéricos, se usa para describir términos de error como se verá en el capítulo 4. La notación O también es usada en el contexto de las ciencias de la computación para clasificar algoritmos de acuerdo a su complejidad. A continuación, se tiene la definición formal.

Definición 2. *Sean* $f(x)$ *y* $g(x)$ *funciones reales. Decimos que* $f(x) = O(g(x))$ *cuando* $x \to \infty$ *si existe un* x_0 *y una constante* M *tal que* $|f(x)| \leq Mg(x)$ *para* $x \geq x_0$. *Decimos entonces que* $|f|$ *se encuentra dominada asintóticamente por arriba mediante la función* g.

Esta definición también se puede adaptar cuando $x \to a$, *en cuyo caso decimos que* $f(x) = O(g(x))$ *cuando* $x \to a$ *si existe un* δ *y una constante* M *tal que* $|f(x)| \leq Mg(x)$ *para* $0 < |x - a| < \delta$.

En el capítulo 4 se usará el caso cuando $x \to 0$, donde la notación será simplemente $f(x) = O(g(x))$, omitiendo la parte de $x \to 0$.

Ejemplo 3. $e^x - 1 - x = O\big(x^2\big)$ o equivalentemente $e^x = 1 + x + O\big(x^2\big)$

Solución: para comprobar que $e^x - 1 - x = O\big(x^2\big)$, se usa la expansión en serie de Taylor $e^x = 1 + x + e^{\xi}\dfrac{x^2}{2}$ donde ξ se encuentra entre 0 y x. En dicho caso, $|e^x - 1 - x| = \dfrac{e^{\xi}}{2}x^2$, y dado que $x \to 0$, se puede asumir que $e^{\xi} \leq 1$ y por tanto $|e^x - 1 - x| \leq \dfrac{1}{2}x^2$ para x suficientemente pequeño, de donde se sigue el resultado. ♢

Ejemplo 4. Usando argumentos similares al anterior, se tiene que $\sin x = x + O\big(x^3\big)$, hecho que se usa frecuentemente en la solución al problema del péndulo simple.

Capítulo 2

Búsqueda de raíces

2.1. Introducción

Uno de los problemas más antiguos, y que con mayor frecuencia se debe resolver en matemáticas puras y aplicadas, consiste en encontrar ceros de funciones, es decir, para una función $f(x)$ dada, encontrar un valor c tal que $f(c) = 0$. En los cursos de cálculo el procedimiento estándar para resolver esta situación es despejar la incógnita.

Ejemplo 5. Resolver $3x - 4 = 0$.

Solución: el problema propone buscar el valor de x, que al ser reemplazado en la ecuación resulte igual a cero. Aplicando las propiedades de la igualdad, se puede proceder de la siguiente forma:

1. Sumar a ambos lados de la igualdad 4:

$$3x - 4 + 4 = 0 + 4$$

   que lleva a:

$$3x = 4$$

2. Multiplicar ambos lados de la ecuación por $\frac{1}{3}$:

$$\frac{3x}{3} = \frac{4}{3}$$

El anterior procedimiento permite despejar la incógnita y lleva a la respuesta final

$$x = \frac{4}{3}.$$

$\diamondsuit$

Ejemplo 6. Encontrar las soluciones de $x^2 - x - 6 = 0$.

Solución: de nuevo, aunque el procedimiento es intentar despejar la incógnita, ahora no parece tan evidente como en el ejemplo anterior:

1. Se suma seis a ambos lados de la ecuación:

$$x^2 - x = 6$$

2. Se completa el trinomio cuadrado perfecto:

$$x^2 - x + \frac{1}{4} = 6 + \frac{1}{4}$$

3. Se factoriza el lado izquierdo de la igualdad:

$$\left(x - \frac{1}{2}\right)^2 = \frac{25}{4}$$

4. Se calcula la raíz cuadrada en ambos lados de la igualdad:

$$\left|x - \frac{1}{2}\right| = \sqrt{\frac{25}{4}}$$

5. Se resuelve la ecuación anterior para obtener:

$$x = \frac{1}{2} \pm \frac{5}{2}$$

Finalmente, se obtienen las dos soluciones (raíces o ceros) de la ecuación, a saber:

$$x = 3 \quad \text{y} \quad x = -2.$$

♢

Un problema presente, es que hay muchas ecuaciones en ciencias e ingeniería, en las cuales no es posible aplicar un procedimiento para despejar la incógnita, como es el caso de:

- $x - \cos x = 0$
- $x - \tan x = 0$
- $e^{-x} - \sin x = 0$
- $e^{-x} - x = 0$

Cualquier intento por despejar la incógnita en las ecuaciones mencionadas arriba fracasa, aunque intuitivamente parece haber respuesta. Por ejemplo, si para la primera ecuación se hace la gráfica de las funciones x y $\cos x$, como se ilustra en la figura 2.1, el punto de corte es el cero que se busca.

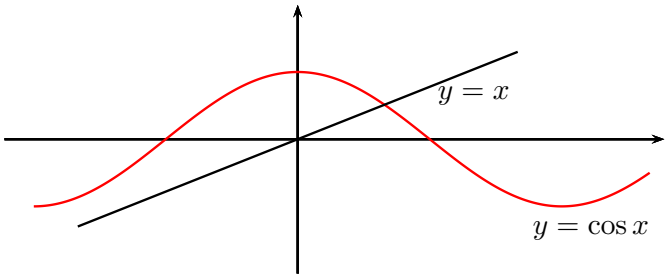


Figura 2.1: la coordenada x del punto de corte de las gráficas de las dos funciones es el cero de la función $f(x) = x - \cos x$.

Surge entonces la pregunta ¿qué se puede hacer? Es en este momento cuando entran los métodos numéricos a resolver estos interrogantes. Aunque varias técnicas van a permitir solucionar estas preguntas, sus respuestas podrían no ser exactas, contrario a las soluciones de los ejemplos 5 y 6. Lo que brindarán los métodos numéricos son aproximaciones a los resultados correctos, en principio tan precisas como se desee.

Para tener claridad del propósito del presente capítulo, se enuncia a continuación el problema que se pretende resolver.

Problema: dada la función $f(x)$, encontrar un valor $c \in \mathbb{R}$ tal que $f(c) = 0$.

En lo que sigue, se exponen diferentes métodos para determinar aproximaciones a la solución del problema general del capítulo.

2.2. Método de bisección

El método de bisección se basa en el teorema del valor intermedio, que en una de sus versiones establece:

Teorema 1. *Si $f : [a, b] \to \mathbb{R}$ es una función continua y $f(a)f(b) < 0$, entonces existe $c \in (a, b)$ tal que $f(c) = 0$.*

Ahora, si $f : [a, b] \to \mathbb{R}$ es una función continua y $f(a)f(b) < 0$, por el teorema del valor intermedio se conoce que existe al menos una solución de la ecuación $f(x) = 0$, y es posible aplicar el siguiente procedimiento, denominado *método de bisección*, para determinar dicha solución.

1. Definir $a_0 = a$ y $b_0 = b$.
2. Calcular el punto medio c_0 del intervalo $[a_0, b_0]$, es decir, $c_0 = \dfrac{a_0 + b_0}{2}$.
   a) Si $f(c_0) = 0$. entonces c_0 es un cero de la función, y por lo tanto una solución al problema. Terminar el procedimiento.

b) Si $f(a_0)f(c_0) > 0$, entonces existe un cero de la función en el intervalo (c_0, b_0). Seleccionar este intervalo y definir $a_1 = c_0$ y $b_1 = b_0$.

c) Si $f(a_0)f(c_0) < 0$, existe un cero de la función en el intervalo (a_0, c_0). Seleccionar este intervalo y definir $a_1 = a_0$ y $b_1 = c_0$.

Nota: observar que ninguno de los tres casos mencionados arriba se puede dar simultáneamente. Además, si la situación es la del caso *b*), donde se tiene seguridad que hay un cero de la función en el intervalo (c_0, b_0), esto no descarta que pueda existir otra raíz en el intervalo (a_0, c_0).

3. Repetir el procedimiento con el nuevo intervalo $[a_1, b_1]$.

A continuación, se demuestra la convergencia del método y en la figura 2.2 se muestra en forma gráfica algunos de los pasos de su ejecución.

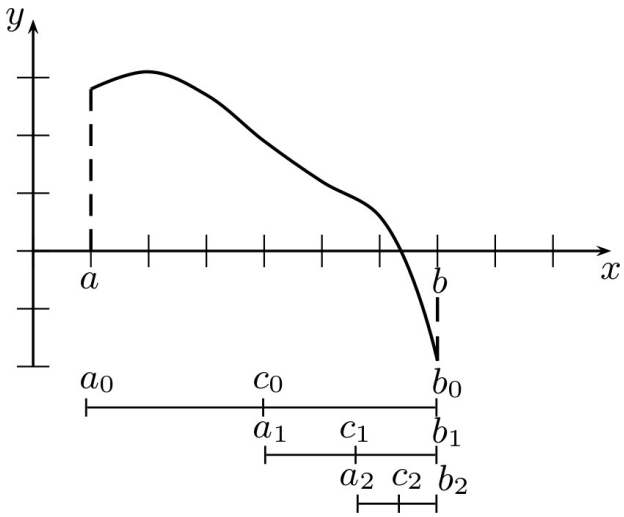


Figura 2.2: un ejemplo del método de bisección.

Teorema 2. *Si se cumplen las hipótesis del teorema del valor intermedio, entonces el método de bisección converge a un cero de $f(x)$ en el intervalo $[a, b]$.*

Demostración. Notar que en cada iteración la medida del intervalo se divide a la mitad. Así, en la primera iteración, la medida del intervalo $[a_1, b_1]$ es la mitad del intervalo inicial. En la segunda iteración, la medida del intervalo $[a_2, b_2]$ es la mitad de la medida del intervalo $[a_1, b_1]$, y por lo tanto corresponde a un cuarto de la medida del intervalo inicial. Similarmente, en la tercera iteración la medida del intervalo $[a_3, b_3]$ es un octavo de la

medida del intervalo inicial, etc. Expresando lo anterior de manera formal se tiene:

$$
\begin{aligned}
|b_1 - a_1| &= \frac{1}{2}|b_0 - a_0| = \frac{1}{2^1}|b_0 - a_0| && \text{Primera iteración} \\
|b_2 - a_2| &= \frac{1}{2}|b_1 - a_1| = \frac{1}{4}|b_0 - a_0| = \frac{1}{2^2}|b_0 - a_0| && \text{Segunda iteración} \\
|b_3 - a_3| &= \frac{1}{2}|b_2 - a_2| = \frac{1}{8}|b_0 - a_0| = \frac{1}{2^3}|b_0 - a_0| && \text{Tercera iteración} \\
&\vdots \\
|b_n - a_n| &= \frac{1}{2^n}|b_0 - a_0| && n\text{-ésima iteración.}
\end{aligned}
$$

Ahora, en cada intervalo $[a_n, b_n]$ existe $c \in \mathbb{R}$ tal que $f(c) = 0$, y por lo tanto, para c_n punto medio de a_n y b_n se tiene:

$$|c_n - c| < \frac{1}{2}|b_n - a_n| = \frac{1}{2^{n+1}}|b_0 - a_0|, \quad \text{para cada } n \in \mathbb{N}$$

Entonces $0 \leq |c_n - c| \leq \dfrac{1}{2^{n+1}}|b_0 - a_0|$. Si ahora se toma el límite cuando el número n de iteraciones tiende al infinito, se tiene que $\lim\limits_{n\to\infty} |c_n - c| \leq \lim\limits_{n\to\infty} \dfrac{1}{2^{n+1}}|b_0 - a_0|$ en caso de existir los límites. Notar que $|b_0 - a_0|$ es una constante, ya que es la medida del intervalo inicial, luego $\lim\limits_{n\to\infty} \dfrac{1}{2^{n+1}}|b_0 - a_0| = |b_0 - a_0| \lim\limits_{n\to\infty} \dfrac{1}{2^{n+1}} = 0$. Se tiene entonces que $\lim\limits_{n\to\infty} |c_n - c|$ existe y es igual a 0, de donde se desprende que $\{c_n\}_{n=0}^{\infty}$ es convergente a c. □

Como se puede observar, el procedimiento encierra a un cero de la función en un subintervalo que en cada iteración es más pequeño que el anterior. Dado que $|c_n - c| \leq \dfrac{1}{2^{n+1}}|b_0 - a_0|$, es posible determinar el número de iteraciones necesarias para obtener una aproximación de un cero de la función tan cercana como se desee. Para lo anterior, observar que si se requiere $|c_n - c| \leq \varepsilon$ para algún $n \in \mathbb{N}$ y ε real positivo, es suficiente que $\dfrac{1}{2^n}|b_0 - a_0| \leq \varepsilon$. Se sigue entonces que

$$
\begin{aligned}
&\frac{1}{2^n}|b_0 - a_0| \leq \varepsilon \\
&\Rightarrow \frac{|b_0 - a_0|}{\varepsilon} \leq 2^n \\
&\Rightarrow \ln\left(\frac{|b_0 - a_0|}{\varepsilon}\right) \leq n \ln 2 \\
&\Rightarrow \frac{\ln|b_0 - a_0| - \ln\varepsilon}{\ln 2} \leq n,
\end{aligned}
$$

y por tanto $n = \left\lceil \dfrac{\ln|b_0 - a_0| - \ln\varepsilon}{\ln 2} \right\rceil$, donde $\lceil x \rceil$ representa la función techo, es tal que $|c_n - c| \leq \varepsilon$.

Ejemplo 7. Determinar un cero de la función $f(x) = -\dfrac{1}{10}x^2 + 3$ utilizando el método de bisección y garantizando un error inferior a $\varepsilon = 10^{-4}$.

Solución: lo primero es determinar dos valores a_0 y b_0 tales que $f(a_0)f(b_0) < 0$. En este caso, se escoge $a_0 = 1$ y $b_0 = 7$ y luego se calcula $c_0 = \dfrac{a_0 + b_0}{2} = \dfrac{1+7}{2} = 4$. Entonces $f(c_0) = 1.4$ y $f(a_0)f(c_0) > 0$, lo que indica que existe un cero de la función en el intervalo $[c_0, b_0]$. Se define $a_1 = c_0$, $b_1 = b_0$ y se repite el proceso. El cuadro 2.1 presenta los resultados.

n	c_n	$\lvert a_n - b_n \rvert$
0	4.000000000000	6.000000000000
1	5.500000000000	3.000000000000
2	4.750000000000	1.500000000000
3	5.125000000000	0.750000000000
4	5.312500000000	0.375000000000
5	5.406250000000	0.187500000000
6	5.453125000000	0.093750000000
7	5.476562500000	0.046875000000
8	5.488281250000	0.023437500000
9	5.482421875000	0.011718750000
10	5.479492187500	0.005859375000
11	5.478027343750	0.002929687500
12	5.477294921875	0.001464843750
13	5.476928710938	0.000732421875
14	5.477111816406	0.000366210938
15	5.477203369141	0.000183105469
16	5.477249145508	0.000091552734

Cuadro 2.1: solución de la ecuación $-\dfrac{1}{10}x^2 + 3 = 0$ con el método de bisección.

Se tiene entonces que $c_{16} = 5.477249145508$ es una aproximación de un cero de la función con la cota superior de error.

$$|c_{16} - c| \leq |b_{16} - a_{16}| = 0.000091552734$$

y por tanto la aproximación es correcta al menos en cuatro cifras decimales. ♢

Ejemplo 8. Determinar la cantidad de iteraciones, en el método de bisección, necesarias para obtener una aproximación de la solución de la ecuación $x - \cos x = 0$ con un error inferior a $\varepsilon = 10^{-4}$ si $a_0 = 0.5$ y $b_0 = 1$.

Solución:

$$\begin{aligned} n &= \left\lceil \frac{\ln|b_0 - a_0| - \ln\varepsilon}{\ln 2} \right\rceil \\ &= \left\lceil \frac{\ln 0.5 - \ln 10^{-4}}{\ln 2} \right\rceil \\ &= \lceil 12.28771238 \rceil \\ &= 13 \end{aligned}$$

Por lo tanto, se necesitan trece iteraciones en el método de bisección para alcanzar la precisión deseada. ♢

En este punto, es necesario mencionar algunos criterios para detener un proceso iterativo como el método de bisección.

2.2.1. Criterios de parada

Si $\{c_n\}_{n=0}^{\infty}$ es una sucesión convergente a un valor c, es decir $\lim_{n\to\infty} c_n = c$, entonces si se desea determinar un $N \in \mathbb{N}$ tal que $|c_N - c| < \dfrac{\varepsilon}{2}$ para una tolerancia $\varepsilon > 0$, es necesario generar $c_1, c_2, c_3, \ldots$ hasta que se cumpla una de las siguientes condiciones:

$$\begin{aligned} &|c_n - c_{n-1}| < \varepsilon, \\ &\frac{|c_n - c_{n-1}|}{|c_n|} < \varepsilon, \quad c_n \neq 0. \end{aligned}$$

El criterio $|c_n - c_{n-1}| < \varepsilon$ se utilizará como criterio de parada para los métodos iterativos de este capítulo.

Ejemplo 9. Sea $c_n = \left(1 + \dfrac{1}{n}\right)^n$, determinar $N \in \mathbb{N}$ tal que $|c_N - c_{N-1}| < 10^{-6}$

Solución: se genera $c_1 = 2, c_2 = 2.25, \ldots$ y se evalúa $|c_N - c_{N-1}|$, concluyendo que para $N = 368$ se tiene $|c_{368} - c_{367}| = |2.71459768679726 - 2.7145876732699| < 10^{-6}$. ♢

2.2.2. Método de *regula falsi*

Este método es un intento por aumentar la rapidez del método de bisección. Los algoritmos solo se diferencian en el punto del intervalo que calculan. Mientras en bisección es el punto medio, en *regula falsi* es el corte con el eje x de la recta que une los puntos extremos de la gráfica definida en el intervalo $[a_n, b_n]$, como se muestra en la figura 2.3.

Si a_n y b_n son los extremos del intervalo, entonces la ecuación de la recta que contiene los puntos $(a_n, f(a_n))$ y $(b_n, f(b_n))$ es $y = \dfrac{f(b_n) - f(a_n)}{b_n - a_n}(x - a_n) + f(a_n)$. En dicho caso, el

corte con el eje x es $a_n - \dfrac{f(a_n)(b_n - a_n)}{f(b_n) - f(a_n)}$, y lo anterior define el punto c_n que se espera brinde una mejor aproximación.

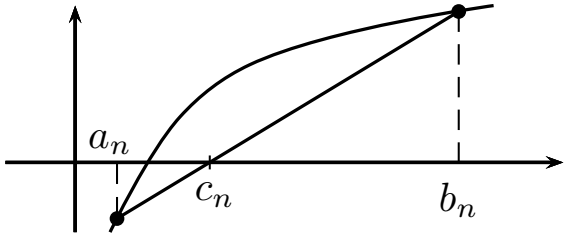


Figura 2.3: método de *regula falsi*.

Ejemplo 10. Determinar un cero de la función $f(x) = -\dfrac{1}{10}x^2 + 3$ con un error inferior a $\varepsilon = 10^{-4}$, utilizando el método de *regula falsi*.

Solución: se determinan dos valores a_0 y b_0 tales que $f(a_0)f(b_0) < 0$. En este caso se escoge $a_0 = 1$ y $b_0 = 7$, y se calcula $c_0 = a_0 - \dfrac{f(a_0)(b_0 - a_0)}{f(b_0) - f(a_0)} = 4.625$ y $f(c_0) = 0.8609375$. Por lo tanto $f(a_0)f(c_0) > 0$, lo que indica que existe un cero de la función en el intervalo $[c_0, b_0]$. Se define $a_1 = c_0$, $b_1 = b_0$ y se repite el proceso. El cuadro 2.2 presenta los resultados.

n	c_n
0	4.625000000000
1	5.365591397849
2	5.463478260870
3	5.475545942929
4	5.477020557904
5	5.477200553464
6	5.477222521303

Cuadro 2.2: solución de la ecuación $-\dfrac{1}{10}x^2 + 3 = 0$ con el método de *regula falsi*.

Luego, utilizando el criterio de parada, c_6 es una aproximación de la solución de la ecuación $-\dfrac{1}{10}x^2 + 3 = 0$ con un error inferior a $\varepsilon = 10^{-4}$. ♢

Observación: las condiciones necesarias para asegurar la convergencia del método de *regula falsi* corresponden a las condiciones del método de bisección. Aunque no es posible aplicar la fórmula del método de bisección para calcular la cantidad de iteraciones necesarias en su ejecución, se espera lograr la precisión establecida en un menor número de iteraciones.

Ejercicios 3

1. Utilizar el método de bisección para obtener c_5, con $f(x) = e^{-x-0.7} - x - 0.7$ en el intervalo $[-1, 0]$.

2. Utilizar el método de bisección para aproximar un cero de la función con una precisión de 10^{-5} dentro del intervalo indicado:

   a) $f(x) = \cos(e^x) + x$ en $[-2, 0]$.

   b) $f(x) = 2^x(x - 6) - x$ en $[-3, -1]$.

   c) $f(x) = \sin(3x) - \cos(2x) - 1$ en $[-3, 5]$.

   d) $f(x) = \dfrac{e^x}{x - 3} + 2x$ en $[1, 2]$.

   e) $f(x) = x^{-2} - \tan x$ en $[3, 4]$.

   f) $f(x) = x^3 - 4x\cos x + (2\sin x)^2 - 3$, en los intervalos $[-2, -1]$, $[-1, 0]$ y $[1, 2]$.

3. Aplicar el método de bisección para la función $f(x) = \dfrac{1}{x}$, con una precisión de 10^{-7} en el intervalo $[-1, 1]$. ¿Qué sucede?

4. Utilizar el método de *regula falsi* para aproximar un cero de la función $f(x) = xe^{-2x} + x + 1$ con una precisión de 10^{-6}.

5. Utilizar el método de bisección y *regula falsi* para aproximar un cero de cada función con una precisión de 10^{-6}. Comparar el número de iteraciones necesarias en cada método.

   a) $f(x) = (x - 1)^{4.5} - 5(x - 1) - 0.1$ en $[2, 3]$.

   b) $g(x) = x\ln(x + 1) - 2$ en $[0, 2]$.

6. Construir una tabla de datos desde -1 hasta 2, en pasos de 0.1 para detectar cambios de signo e identificar intervalos donde ocurran ceros de la función

$$f(x) = \frac{x^5 - 3x^3 - 8x^2 - x - 4}{x^3 + 2x^2 + x + 6}.$$

   Con el método de *regula falsi*, encontrar dichos ceros con precisión hasta la tercera cifra decimal.

7. Con ayuda de algún *software* de cómputo científico, construir la gráfica de

$$f(x) = |x| - \cos x$$

   en el intervalo $[-4, 4]$. Con el método de bisección, encontrar los ceros con una precisión de cuatro cifras decimales.

2.3. Método de Newton-Raphson

Este método se basa en elementos del cálculo diferencial y su interpretación gráfica. Es uno de los métodos más eficientes[1] para determinar, con una precisión deseada, una aproximación de la solución de la ecuación $f(x) = 0$.

Gráficamente (ver figura 2.4), se puede interpretar el método de Newton de la siguiente manera:

1. Seleccionar un punto inicial (semilla) p_0.
2. Calcular la ecuación de la recta tangente a la curva $f(x)$ que pasa por el punto $(p_0, f(p_0))$
3. Determinar el corte con el eje x de la recta tangente y nombrar como p_1.
4. Si $f(p_1) \neq 0$ repetir los pasos $2, 3$ y 4 para p_1.

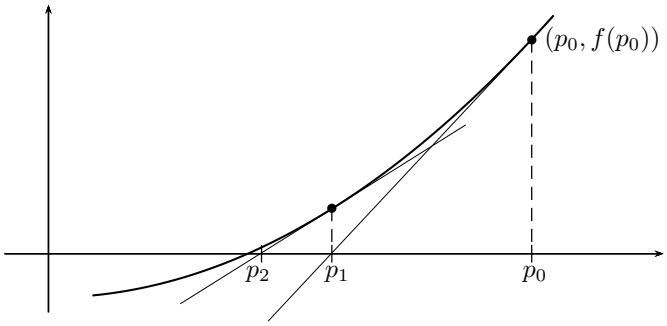


Figura 2.4: método de Newton-Raphson.

Ahora, la ecuación de la recta tangente a la curva $y = f(x)$ que pasa por el punto $(p_0, f(p_0))$ tiene como pendiente $m = f'(p_0)$ y corresponde a

$$y = f'(p_0)x + [f(p_0) - f'(p_0)p_0]$$

para calcular el cero de esta recta se reemplaza y por cero

$$0 = f'(p_0)x + f(p_0) - f'(p_0)p_0$$

y al despejar x, se tiene $x = p_0 - \dfrac{f(p_0)}{f'(p_0)}$. Dicho valor de x se nombra como p_1 y por tanto:

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}$$

[1] En cuanto a cantidad de iteraciones se refiere.

Si se repite el proceso, pero ahora con el punto $(p_1, f(p_1))$, se obtiene:

$$p_2 = p_1 - \frac{f(p_1)}{f'(p_1)}.$$

Repitiendo ahora con $(p_2, f(p_2))$

$$p_3 = p_2 - \frac{f(p_2)}{f'(p_2)},$$

y finalmente, en la n-ésima iteración

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \tag{2.1}$$

que es la fórmula de iteración correspondiente al método de Newton-Raphson.

Ejemplo 11. Determinar una solución de la ecuación $e^{-x} - \sin x = 0$ con una precisión de $\varepsilon = 10^{-6}$ usando el método de Newton-Raphson.

Solución: si se realiza la gráfica de la función $f(x) = e^{-x} - \sin x$, se observa que la función posee varios ceros y uno de ellos está en el intervalo $[0, 1]$. Por lo tanto, seleccionando $p_0 = 0.2$ se espera que el método de Newton-Raphson sea convergente. Por otro lado, $f'(x) = -e^{-x} - \cos x$, de donde se tiene:

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)} = 0.2 - \frac{e^{-0.2} - \sin 0.2}{-e^{-0.2} - \cos 0.2} = 0.544708885$$

Ahora, $p_2 = p_1 - \dfrac{f(p_1)}{f'(p_1)} = 0.587795322484$, observando los resultados del cuadro 2.3 y utilizando el criterio de parada, se obtiene que $p_4 = 0.588532743982$ es una aproximación de la solución de la ecuación $e^{-x} - \sin x = 0$ con una precisión de $\varepsilon = 10^{-6}$. ♢

n	p_n
0	0.200000000000
1	0.544708885000
2	0.587795322484
3	0.588532526471
4	0.588532743982

Cuadro 2.3: resultados del método de Newton-Raphson.

2.3.1. Método de la secante

Aunque el método de Newton-Raphson es uno de los más eficientes para determinar una solución de la ecuación $f(x) = 0$, la necesidad de conocer $f'(x)$ representa una de sus

mayores debilidades, pues el cálculo de la derivada requiere y representa más operaciones por realizar. Una manera de evitar el problema de calcular la derivada, es recordar que

$$f'(p_{n-1}) = \lim_{x \to p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}}$$

y dado que p_{n-2} se encuentra cerca de p_{n-1} en caso de ser convergente el método de Newton-Raphson, entonces es posible aproximar $f'(p_{n-1})$ por

$$f'(p_{n-1}) \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}}.$$

Al sustituir en la fórmula de iteración de Newton-Raphson, se obtiene

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}.$$

La anterior fórmula de iteración se denomina *método de la secante*. Es de anotar, que para utilizar el anterior método, son necesarias dos aproximaciones iniciales (semillas) p_0 y p_1. Una interpretación geométrica (figura 2.5) del método de la secante es la siguiente:

1. Seleccionar dos puntos iniciales p_0 y p_1.
2. Calcular la ecuación de la recta que pasa por los puntos $(p_0, f(p_0))$ y $(p_1, f(p_1))$.
3. Determinar el corte con el eje x de la anterior recta y nombrarlo como p_2.
4. Si $f(p_2) \neq 0$, repetir los pasos $2, 3$ y 4 para p_1 y p_2.

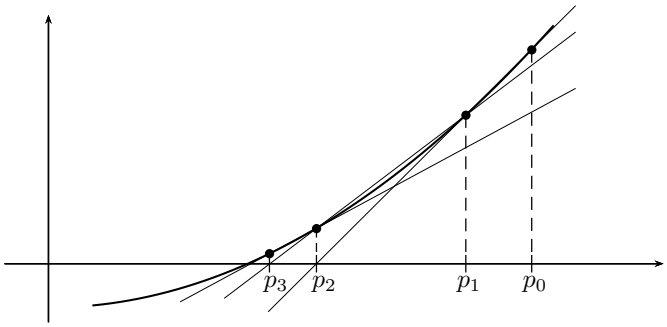


Figura 2.5: método de la secante.

Ejemplo 12. Aplicar el método de la secante para obtener una solución de la ecuación $x - 0.5\tan(x) = 0$ con una precisión de $\varepsilon = 10^{-6}$ en el intervalo $[1, 2]$.

Solución: tomando como puntos iniciales $p_0 = 1.2$ y $p_1 = 1$, dado que la gráfica de $f(x) = x - 0,5\tan(x)$ así lo indica, se obtiene

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} = 1 - \frac{0.221296137673(1 - 1.2)}{0.221296137673 - (-0.086075811063)} = 1.143992409576.$$

Al continuar las iteraciones, se tiene que $p_7 = 1.165561185173$ es la aproximación deseada a la solución de la ecuación $x - 0.5\tan(x) = 0$. Los resultados del método se presentan en el cuadro 2.4. ♢

n	p_n
0	1.200000000000
1	1.000000000000
2	1.143992409576
3	1.180243146455
4	1.164484056894
5	1.165508347221
6	1.165561377918
7	1.165561185173

Cuadro 2.4: resultados del método de la secante.

Ejercicios 4

1. Determinar p_3 en el método de Newton-Raphson al aplicarlo en la solución de la ecuación $e^{-x} - x = 0$ si $p_0 = -1$.

2. Usar el método de Newton-Raphson para obtener una aproximación de las soluciones de los siguientes problemas con una precisión de $\varepsilon = 10^{-7}$.

   a) $4x^2 - 4xe^{-2x} + e^{-4x} = 0$ en $[0, 1]$.
   b) $3x^2 + \ln(x)(2x + \ln(x)) = 2x^2$ en $[0, 1]$.
   c) $e^{-2}x^2 + 2e^{-1}x - 1 = 0$ en $[0, 2]$.

3. Utilizar el método de Newton-Raphson para aproximar la solución de cada ecuación desde el punto p_0 y con una precisión de $\varepsilon = 10^{-8}$.

   a) $\tan^{-1}(x) = 0$, $p_0 = 1.2$.
   b) $-x^4 + 6x^2 + 1 = 0$, $p_0 = 3$.
   c) $1 - 2e^{-x^2} = 0$, $p_0 = 1.5$.
   d) $\sin x - e^{-x} = 0$, $p_0 = 2.5$.

4. Usar el método de Newton-Raphson para aproximar $\sqrt[3]{25}$ con seis cifras decimales correctas. *Ayuda:* considerar $f(x) = x^3 - 25$.

5. Usar el método de la secante para aproximar un cero (con una precisión de $\varepsilon = 10^{-6}$) de la función $h(x) = x^3 - 3x + 1$ en el intervalo $[1, 2]$.

6. Sea $f(x) = \ln(x)$, $p_0 = 1.3$ y $p_1 = 1.5$. Utilizar el método de la secante para aproximar un cero de la función con una precisión $\varepsilon = 10^{-4}$. Comparar con el resultado de aplicar el método de Newton-Raphson con punto inicial $p_0 = 1.4$.

7. Si $f(x) = e^{-2x} - 2x + 1$, utilizar el método de Newton-Raphson para aproximar una solución de la ecuación $f(x) = 0$ con una precisión de $\varepsilon = 10^{-8}$ si $p_0 = 1.5$. Comparar (en cantidad de iteraciones) el resultado con el obtenido al aplicar el método de la secante con $p_0 = 1.6$ y $p_1 = 1.7$.

8. Identificar el propósito de la siguiente fórmula de iteración obtenida con el método de Newton-Raphson:
$$x_{n+1} = 2x_n - x_n^2 R.$$
Para lo anterior, considerar que cuando $n \to \infty$, $x_n \to A$.

9. Identificar el propósito de la siguiente fórmula de iteración obtenida con el método de Newton-Raphson:
$$x_{n+1} = 0.5\left(x_n + \frac{19}{x_n}\right).$$
Para lo anterior, considerar que cuando $n \to \infty$, $x_n \to A$.

2.4. Método de punto fijo

Desde la antigua Babilonia es conocido el siguiente método para calcular $\sqrt{A}$:

1. Seleccionar x_0 cercano a $\sqrt{A}$.
2. Calcular $x_{n+1} = \frac{1}{2}\left(x_n + \frac{A}{x_n}\right)$ para $n > 0$.

Este método es eficiente y uno de los mejores ejemplos de la técnica que se desarrolla en esta sección para aproximar soluciones de ecuaciones. Aunque existen muchas técnicas numéricas que dan solución a la ecuación $f(x) = 0$, tal vez una de las más famosas es el llamado *método del punto fijo*, un procedimiento para aproximar la solución de la ecuación $x = g(x)$. Para comprender este método, es necesario dar algunas definiciones y ejemplos.

Definición 3. *Un punto fijo de una función $g(x)$ es un valor c tal que $g(c) = c$.*

Ejemplo 13. La función $g(x) = x$, tiene infinitos puntos fijos, dado que para cualquier valor de c, $g(c) = c$.

Ejemplo 14. Mostrar que la función $g(x) = x^2 + 6x + 6$ tiene dos puntos fijos, $x = -2$ y $x = -3$.

Solución: se tiene que $g(-2) = (-2)^2 + 6(-2) + 6 = 4 - 12 + 6 = -2$, luego $g(-2) = -2$. Similarmente $g(-3) = 9 - 18 + 6 = -3$ y por tanto $g(-3) = -3$. ♢

¿Por qué es relevante encontrar puntos fijos? La respuesta radica en que los problemas de encontrar ceros de funciones y encontrar puntos fijos de funciones están estrechamente relacionados. Por ejemplo, para calcular los puntos fijos de la función del ejemplo 14, se puede plantear la ecuación, $g(x) = x$, es decir $x^2 + 6x + 6 = x$, que organizando términos lleva a la ecuación $x^2 + 5x + 6 = 0$. Luego el problema de buscar el punto fijo de $g(x)$ se convirtió en el de buscar ceros de una nueva función, en este caso de $f(x) = x^2 + 5x + 6$. Para generalizar y resumir, se construyó una función $f(x)$ a partir de la función $g(x)$, de la manera $f(x) = g(x) - x$, de tal forma que si la función $g(x)$ tiene un punto fijo, c, este punto fijo de $g(x)$ es un cero de $f(x)$, dado que $f(c) = g(c) - c$ y como $g(c) = c$, entonces $f(c) = c - c = 0$.

El problema inverso también es válido. Si la función $f(x)$ tiene un cero en p, entonces podemos construir una nueva función $g(x)$ a partir de $f(x)$, en la cual el cero de $f(x)$ sea un punto fijo de $g(x)$. Se puede verificar lo anterior tomando, por ejemplo, a $g(x) = x + f(x)$. Notar que si se evalúa $g(x)$ en p se obtiene $g(p) = p + f(p)$, y como $f(p) = 0$, entonces $g(p) = p + 0 = p$ y por lo tanto el cero de $f(x)$ es un punto fijo de $g(x)$. Entonces, como seguramente ya se intuye, la idea general es cambiar el problema $f(x) = 0$ por el problema $g(x) = x$.

El siguiente teorema establece condiciones suficientes para la existencia de un punto fijo.

Teorema 3. *Sean $[a, b] \subseteq \mathbb{R}$ y $g : [a, b] \to [a, b]$ una función continua. Entonces g tiene un punto fijo en $[a, b]$.*

Demostración. Si $g(a) = a$ o $g(b) = b$, entonces un punto fijo de g es un extremo. En caso contrario, se tiene $g(a) > a$ y $g(b) < b$. Dado que $f(x) = g(x) - x$ es una función continua en $[a, b]$ y $f(a) = g(a) - a > 0$ y $f(b) = g(b) - b < 0$, por el teorema del valor intermedio se sigue que f tiene un cero en $[a, b]$, es decir, existe $p \in [a, b]$ tal que $f(p) = 0$, lo que implica que $g(p) - p = 0$, o de manera equivalente $g(p) = p$. □

Ahora, para calcular una aproximación de un punto fijo de una función g es posible aplicar el siguiente procedimiento (ver figura 2.6):

1. Seleccionar un punto inicial p_0.
2. Calcular $g(p_0)$ y nombrarlo como p_1.
3. Si $g(p_1) \neq p_1$, repetir los pasos 2 y 3 para p_1.

Este procedimiento recibe el nombre de *método de punto fijo.*

Ejemplo 15. Utilizar el método de punto fijo para obtener una aproximación a la solución de la ecuación $x = \sqrt{\dfrac{10}{x+4}}$ con una precisión de $\varepsilon = 10^{-7}$.

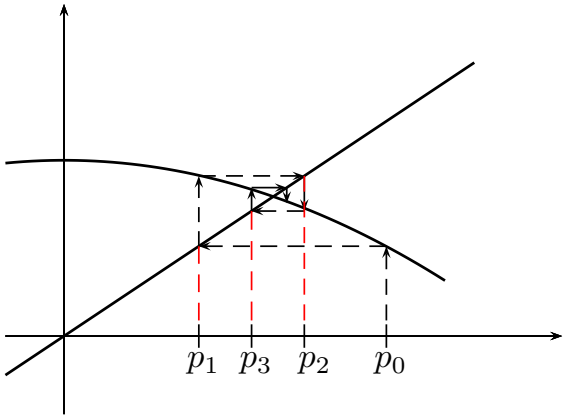


Figura 2.6: método de punto fijo.

Solución: seleccionar un punto inicial p_0, en este caso $p_0 = 1.5$. Calcular $p_1 = g(p_0) = \sqrt{\dfrac{10}{p_0 + 4}} = 1.348399724926$, y dado que $|p_0 - p_1| > 10^{-7}$, se repite el proceso. Calcular $p_2 = g(p_1) = 1.367376371991$, y ya que $|p_1 - p_2| > 10^{-7}$ se sigue iterando. Los resultados se resumen en el cuadro 2.5, de donde $p_8 = 1.365230022516$ es una aproximación a la solución de la ecuación $x = \sqrt{\dfrac{10}{x + 4}}$ con la precisión deseada. ♢

n	p_n
0	1.500000000000
1	1.348399724926
2	1.367376371991
3	1.364957015402
4	1.365264748113
5	1.365225594161
6	1.365230575673
7	1.365229941878
8	1.365230022516

Cuadro 2.5: resultados del método de punto fijo.

En este punto es necesario presentar condiciones suficientes para garantizar que el método de punto fijo sea convergente. El siguiente teorema, conocido como *teorema de punto fijo* es un resultado fundamental al respecto.

Teorema 4. *Sean $g : [a, b] \to [a, b]$ una función continua, $s \in [a, b]$ tal que $g(s) = s$ y $|g'(x)| \leq \alpha < 1$, donde $x \in (a, b)$ y α es constante. Sea $p_0 \in [a, b]$. Entonces la sucesión $\{p_n\}_{n=0}^{\infty}$ dada por $p_{n+1} = g(p_n)$ es tal que $p_n \to s$ cuando $n \to \infty$.*

Demostración. Como $g(x)$ es continua en $[a, b]$ y tanto p_0 como s, se encuentran en $[a, b]$, entonces el teorema del valor medio para derivadas asegura la existencia de ζ tal que:

$$g'(\zeta) = \frac{g(p_0) - g(s)}{p_0 - s}.$$

Tomando el valor absoluto y observando las hipótesis del teorema de punto fijo,

$$|g'(\zeta)| = \left| \frac{g(p_0) - g(s)}{p_0 - s} \right| \leq \alpha < 1.$$

Al aplicar propiedades del valor absoluto se tiene

$$|g(p_0) - g(s)| \leq \alpha |p_0 - s|.$$

Como $g(s) = s$ y $g(p_0) = p_1$, al reemplazar se sigue que:

$$|p_1 - s| \leq \alpha |p_0 - s|.$$

Lo cual significa que la distancia entre p_1 y s es menor que la distancia entre p_0 y s. Ahora se repite el razonamiento, pero empezando con p_1 y s, lo cual lleva a:

$$|p_2 - s| \leq \alpha |p_1 - s|,$$

luego

$$|p_2 - s| \leq \alpha |p_1 - s| \leq \alpha(\alpha |p_0 - s|) = \alpha^2 |p_0 - s|,$$

por lo tanto

$$|p_2 - s| \leq \alpha^2 |p_0 - s|,$$

y continuando el proceso hasta la n-ésima iteración, se tiene:

$$|p_n - s| \leq \alpha^n |p_0 - s|.$$

Si se toma el límite cuando n tiende a infinito:

$$\lim_{n \to \infty} |p_n - s| \leq \lim_{n \to \infty} \alpha^n |p_0 - s| = |p_0 - s| \lim_{n \to \infty} \alpha^n$$

y como α es tal que $0 \leq \alpha < 1$, se tiene que $\lim_{n \to \infty} \alpha^n = 0$ y por tanto:

$$\lim_{n \to \infty} |p_n - s| = 0,$$

lo que significa que siempre que n tienda a infinito, p_n tiende a s. □

Para ilustrar la forma en que opera el método de punto fijo y su relación con la ecuación $f(x) = 0$, se analizará un par de situaciones.

Ejemplo 16. Encontar una solución de $x^3 + 4x^2 - 10 = 0$ con una precisión de $\varepsilon = 10^{-7}$.

Solución: como el método de punto fijo encuentra aproximaciones a la solución de la ecuación $x = g(x)$, lo primero que se debe realizar es convertir el problema $f(x) = 0$ en un problema $x = g(x)$, para lo cual es suficiente despejar una x:

$$\begin{aligned} x^3 + 4x^2 - 10 &= 0 \\ x^3 + 4x^2 &= 10 \\ x^2(x+4) &= 10 \\ x^2 &= \frac{10}{x+4} \\ x &= \sqrt{\frac{10}{x+4}} \end{aligned}$$

Ahora, una forma de garantizar que el método sea convergente, es verificar que $g(x) = \sqrt{\frac{10}{x+4}}$ cumple las hipótesis del teorema 4 en algún intervalo, en este caso $[1, 2]$. Al seleccionar $p_0 = 1.5$ se obtienen los resultados del cuadro 2.5 presentado en el ejemplo 15. ♢

Ejemplo 17. Encontrar una aproximación con tres decimales correctos a la solución de $\cos x - x = 0$.

Solución: repitiendo los pasos del ejemplo anterior, se tiene:

1. Despejar una x. Lo más simple es $x = \cos x$ y por tanto $g(x) = \cos x$.
2. Para garantizar que el método de punto fijo sea convergente, se busca el intervalo $[a, b]$ para satisfacer las hipótesis del teorema 4, en este caso $[0, 1]$.
3. Seleccionar p_0 en el intervalo, en este ejemplo $p_0 = 0$.
4. Aplicar la iteración $p_{n+1} = g(p_n)$.

En el cuadro 2.6 se incluyen los valores de la sucesión $p_1, p_2, p_3, \ldots, p_{24}$, donde en p_{24} ya se ha alcanzado la precisión deseada. ♢

Observación: se debe apreciar que una implementación de método de punto fijo debe contemplar un número máximo de iteraciones, debido a que no todas las fórmulas son convergentes. Además, en caso de convergencia, se deben estimar otros criterios que detengan el programa en el momento de alcanzar la precisión deseada, como es el caso del criterio de parada $|p_{i+1} - p_i| < \varepsilon$ de la sección 2.2.1.

Ejemplo 18. Usando una fórmula de punto fijo, encontrar el valor de un cero de $f(x) = x^3 - x - 1$ preciso hasta la cuarta cifra decimal.

Solución:

1. Despejar x, en este caso, $x = x^3 - 1$ y por tanto $g(x) = x^3 - 1$.

i	p_i
1	1.000000000000
2	0.540302305868
3	0.857553215846
4	0.654289790498
5	0.793480358743
⋮	⋮
19	0.739303892397
20	0.738937756715
21	0.739184399771
22	0.739018262427
23	0.739130176530
24	0.739054790747

Cuadro 2.6: solución de la ecuación $x = \cos x$.

2. Elegir p_0, en este caso $p_0 = 1.5$, pues la evidencia gráfica respalda la existencia de un cero en el intervalo $[1, 2]$.

3. Aplicar $p_{n+1} = g(p_n)$ obteniendo los resultados del cuadro 2.7.

i	p_i
0	1.500000000000
1	2.375000000000
2	12.39648437500
3	1904.002772234
4	6902441412.889
5	3.2885783**E**+29
6	3.5565143**E**+88
7	4.498561**E**+265

Cuadro 2.7: solución de la ecuación $f(x) = x^3 - x - 1 = 0$. Método divergente.

En este momento es mejor detener el proceso, puesto que la sucesión no tiende a ningún número en particular. Por el contrario, tiende a infinito, y por tanto la sucesión es divergente. ♢

El ejemplo anterior presenta una de las dificultades del método de punto fijo para aproximar una solución de la ecuación $f(x) = 0$: la necesidad de transformar $f(x) = 0$ en un problema del formato $x = g(x)$, donde $g(x)$ cumpla las hipótesis del teorema 4. A continuación se tiene un ejemplo al respecto.

Ejemplo 19. Usando una fórmula de punto fijo, encontrar el valor de un cero de $f(x) = x^3 - x - 1$ preciso hasta la cuarta cifra decimal.

Solución: al despejar x, en este caso, $x = x^3 - 1$ y usar $g(x) = x^3 - 1$, el ejemplo anterior muestra que se tiene una sucesión divergente. En cambio, si se usa el despeje alternativo

$$x = \sqrt{\frac{x+1}{x}}$$

válido cuando $x > 0$, se tiene un problema equivalente y convergente según los resultados del cuadro 2.8, donde se aplica el método de punto fijo a $g(x) = \sqrt{\dfrac{x+1}{x}}$ con $p_0 = 0.9$. $\diamondsuit$

i	p_i
0	0.900000000000
1	1.452966314514
2	1.299325671882
3	1.330274402908
4	1.323527334417
5	1.324974241803
6	1.324662845115
7	1.324729811194
8	1.324715407719

Cuadro 2.8: solución de la ecuación $f(x) = x^3 - x - 1 = 0$. Método convergente.

Ejercicios 5

1. Solucionar las siguientes ecuaciones utilizando el método de punto fijo con una precisión de $\varepsilon = 10^{-8}$.

   a) $x - e^{-x} = 0$.

   b) $x + 4e^{-2x} - 4\ln(x) = 0$.

   c) $0.5\sin(e^{-2x}) - x = 0$.

   d) $e^{-x} + \ln(x+8) - x^2 = 0$.

2. Dados los siguientes esquemas de punto fijo para obtener una solución de la ecuación $x^3 - 4x^2 + 10 = 0$, clasificar por la rapidez de convergencia. Suponer que $p_0 = -1.5$.

   a) $p_n = \sqrt{\dfrac{-10}{p_{n-1} - 4}}$

   b) $p_n = \dfrac{8p_{n-1} - 10}{p_{n-1}^2 - 4p_{n-1} + 8}$

   c) $p_n = p_{n-1} - \dfrac{p_{n-1}^3 - 4p_{n-1}^2 + 10}{3p_{n-1}^2 - 8p_{n-1}}$

   d) $p_n = \dfrac{\sqrt[3]{4p_{n-1}^2 - 10} + p_{n-1}}{2}$

3. Utilizar el método de punto fijo y el método de Newton-Raphson para determinar una solución de la ecuación $3e^{-x} - 2x + \ln(x) = 0$ con una precisión de $\varepsilon = 10^{-8}$. Comparar la cantidad de iteraciones necesarias en cada método.

4. Utilizar el método de punto fijo y el método de Newton-Raphson para determinar un cero de la función $f(x) = x^3 - 3x^2e^{-x} + 3xe^{-2x} - e^{-3x}$ con una precisión de $\varepsilon = 10^{-8}$.

5. Utilizar el método de punto fijo para determinar $1/9$ al utilizar únicamente sumas y multiplicaciones. *Sugerencia:* considerar $f(x) = 9x - 1$.

6. Utilizar el método de punto fijo y el método de la secante para determinar una solución de la ecuación $x^3 - xe^{-x} = 0$ con una precisión de $\varepsilon = 10^{-6}$. Comparar la cantidad de iteraciones necesarias en cada método.

7. Si se tiene que el siguiente esquema de punto fijo es convergente, determinar su valor límite:
$$\begin{cases} p_0 = 0.8 \\ p_n = \dfrac{2p_{n-1} - 1}{p_{n-1}^2 - 2p_{n-1} + 2} \end{cases}$$
Sugerencia: considerar $g(x) = \dfrac{2x - 1}{x^2 - 2x + 2}$.

8. Considerar la expresión:
$$x = \sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}}.$$
Demostrar con razonamientos de punto fijo, que tiende a la razón áurea $\phi = \frac{1+\sqrt{5}}{2}$.

9. Considerar la fracción continua:
$$x = \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \cdots}}}.$$
Con razonamientos de punto fijo, demostrar que tiende a $\phi - 1$, donde ϕ es como en el ejemplo anterior.

2.5. Orden de convergencia en el método de punto fijo

En esta parte se estudiará el orden de convergencia de los esquemas de punto fijo, y se observará que, en la mayoría de los casos, la convergencia del método es lineal de acuerdo a lo expuesto en la sección 1.5.

Teorema 5. *Sea $g : [a, b] \to [a, b]$ una función continua, $g'(x)$ continua en (a, b) y $k < 1$ constante positiva con*
$$|g'(x)| \leq k \quad \text{para todo } x \in (a, b).$$
Si $g'(p) \neq 0$, para p punto fijo de g en $[a, b]$, entonces para cualquier número p_0 en $[a, b]$ la sucesión
$$p_n = g(p_{n-1}) \quad n \geq 1$$
converge linealmente a p.

Demostración. Con los resultados de la sección 2.4 se puede comprobar que la sucesión $p_n = g(p_{n-1})$ converge a p. Dado que $g'(x)$ existe, entonces por el teorema del valor medio, para cada $n \geq 1$ existe ξ_n tal que

$$\frac{g(p_n) - g(p)}{p_n - p} = g'(\xi_n)$$

luego $p_{n+1} - p = g(p_n) - g(p) = g'(\xi_n)(p_n - p)$, donde ξ_n esta entre p_n y p. Ahora, como $\{p_n\}_{n=0}^{\infty}$ converge a p, entonces $\{\xi_n\}_{n=1}^{\infty}$ converge a p, y por la continuidad de $g'(x)$ se tiene que

$$\lim_{n\to\infty} g'(\xi_n) = g'(p).$$

Por lo tanto

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} = \lim_{n\to\infty} |g'(\xi_n)| = |g'(p)|$$

y por hipótesis, $g'(p) \neq 0$, de donde se concluye que $\{p_n\}_{n=0}^{\infty}$ converge linealmente al punto p. □

Aunque el teorema anterior demuestra el comportamiento de los esquemas de punto fijo, existen casos donde la convergencia del método de punto fijo es cuadrática, y para lo anterior, hay que observar lo siguiente:

Si p es un punto fijo de $g(x)$ y se conoce $g(x)$, $g'(x)$, $g''(x)$, ... en p, entonces se puede expandir $g(x)$ en serie de Taylor alrededor de p

$$g(x) = g(p) + g'(p)(x - p) + \frac{g''(p)}{2!}(x - p)^2 + \frac{g'''(p)}{3!}(x - p)^3 + \cdots,$$

y al evaluar la serie en p_n se obtiene

$$g(p_n) = g(p) + g'(p)(p_n - p) + \frac{g''(p)}{2!}(p_n - p)^2 + \frac{g'''(p)}{3!}(p_n - p)^3 + \cdots.$$

Ahora, dado que $g(p) = p$, ya que es punto fijo, y también que $g(p_n) = p_{n+1}$, se desprende que

$$p_{n+1} = p + g'(p)(p_n - p) + \frac{g''(p)}{2!}(p_n - p)^2 + \frac{g'''(p)}{3!}(p_n - p)^3 + \cdots$$

al pasar p al lado izquierdo

$$p_{n+1} - p = g'(p)(p_n - p) + \frac{g''(p)}{2!}(p_n - p)^2 + \frac{g'''(p)}{3!}(p_n - p)^3 + \cdots$$

lo cual, si se define el error en la iteración n como $e_n = p_n - p$, llevaría a:

$$e_{n+1} = g'(p)e_n + \frac{g''(p)}{2!}e_n^2 + \frac{g'''(p)}{3!}e_n^3 + \cdots \tag{2.2}$$

Si $|e_n| < 1$ entonces la magnitud de $|e_n^2|$ es más pequeña y la de $|e_n^3|$ aún más y así sucesivamente. En estas circunstancias, si $g'(p) \neq 0$, entonces el primer término domina el error, es decir $e_{n+1} \approx e_n$, y en este caso el método sería de primer orden.

Si $g'(p) = 0$ y $g''(p) \neq 0$, el segundo término de la serie 2.2 es el que domina el error y por lo tanto, $e_{n+1} \approx e_n^2$ y el método sería de segundo orden. Ahora, si $g'(p) = g''(p) = 0$ y $g'''(p) \neq 0$, entonces es el tercer término el que domina y el método sería de tercer orden, puesto que $e_{n+1} \approx e_n^3$.

2.6. Orden del método de Newton-Raphson

Para $f(x)$ una función continua, con $f(c) = 0$ para $c \in \mathbb{R}$ y $f'(c) \neq 0$, si se aplica el método de Newton-Raphson para aproximar c, se tiene que

$$\begin{cases} p_0 \\ p_{n+1} = p_n - \dfrac{f(p_n)}{f'(p_n)} & \text{si } n > 0 \end{cases}$$

Si se observa de manera detallada el método de Newton-Raphson, se puede determinar que su forma corresponde a un esquema de punto fijo, donde $g(x) = x - \dfrac{f(x)}{f'(x)}$. Entonces se podría asegurar que la convergencia es lineal, pero

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

cuyo valor en la raíz c de $f(x)$, que es el punto fijo de $g(x)$, es:

$$g'(c) = 1 - \frac{(f'(c))^2 - f(c)f''(c)}{(f'(c))^2} = \frac{f(c)f''(c)}{(f'(c))^2} = 0.$$

Luego el método parece ser de orden dos, pero todavía no es posible asegurar lo anterior hasta verificar el valor de la segunda derivada:

$$g''(x) = \frac{(f'(x)f''(x) + f(x)f'''(x))(f'(x))^2 - 2f'(x)f''(x)f(x)f''(x)}{(f'(x))^4}.$$

Al evaluar en c se desprende que

$$g''(c) = \frac{(f'(c)f''(c) + f(c)f'''(c))(f'(c))^2 - 2f'(c)f''(c)f(c)f''(c)}{(f'(c))^4} = \frac{f''(c)}{f'(c)},$$

de donde si $f''(c) \neq 0$, entonces $g''(c) \neq 0$ y el método de Newton-Raphson sería de segundo orden.

Voigtländer
Landschaft
Gruppe
Porträt
Anastigmat Voigtar
Voigtländer Braunschweig
Voigtländer
BRILLANT

Capítulo 3

Interpolación

3.1. Introducción

El problema de *interpolar* aparece con mucha frecuencia en el trabajo en ciencias e ingeniería, donde los experimentos arrojan datos numéricos y a su vez estos datos se pueden representar en tablas o en forma gráfica. Para analizar el comportamiento del sistema, no basta con observar los datos obtenidos en el experimento (figura 3.1), y en cambio se debe intentar ajustar o aproximar una curva (función) que permita *predecir* valores en puntos para los cuales no se dispone de información, y a su vez sirva como modelo del comportamiento del sistema.

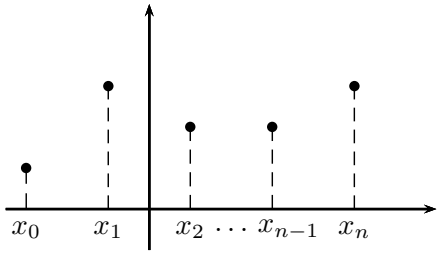


Figura 3.1: problema de interpolación.

Se pueden ajustar funciones de interpolación a una lista de datos siguiendo dos criterios: ajuste exacto y ajuste por mínimos cuadrados (figura 3.2). Decidir el tipo de ajuste a utilizar, depende de las necesidades de la investigación u origen de los datos.

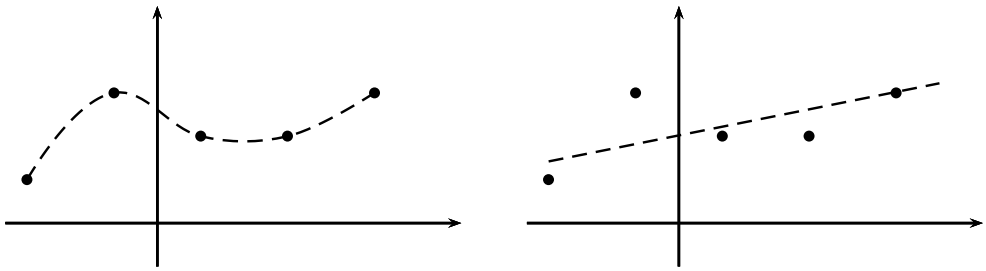


Figura 3.2: tipos de interpolación: ajuste exacto (izquierda) y ajuste por mínimos cuadrados (derecha).

3.2. Ajuste exacto

Una de las técnicas de interpolación (exacta) para un conjunto de datos (x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) es la *interpolación polinomial*, donde se busca un polinomio $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ de *menor grado posible* que contenga a todos los datos conocidos, es decir, $p(x_i) = y_i$ para todo $i = 0, 1, \ldots, n$. Por ejemplo, para los datos $(1, -1), (2, -4)$ y $(3, -9)$, el interpolador polinomial es $p(x) = -x^2$ dado que no existe un polinomio de menor grado que contenga a estos puntos. A continuación, se presentarán dos métodos para construir un polinomio de interpolación.

3.2.1. Polinomio de interpolación de Lagrange

Suponer que se tienen $n + 1$ puntos (x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) con $x_i \neq x_j$, siempre que $i \neq j$. Para construir un polinomio de interpolación, se puede realizar lo siguiente:

- Determinar un polinomio $\ell_0(x)$ que pase por $(x_0, 1)$ y que se anule en x_1, x_2,..., x_n, es decir,

$$\ell_0(x_j) = \begin{cases} 1 & \text{si } j = 0 \\ 0 & \text{si } j \neq 0 \end{cases}, \qquad \text{para } j = 0, 1, 2, \ldots, n.$$

  ¿Cómo construir $\ell_0(x)$? La primera opción es:

$$\tilde{P}_0(x) = (x - x_1)(x - x_2)(x - x_3) \cdots (x - x_n)$$

  Notar que si se evalúa en x_1,

$$\tilde{P}_0(x_1) = (x_1 - x_1)(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_n) = 0,$$

  que se anula porque el primer factor es cero. Ahora, al evaluar en x_2:

$$\tilde{P}_0(x_2) = (x_2 - x_1)(x_2 - x_2)(x_2 - x_3) \cdots (x_2 - x_n) = 0,$$

lo cual también se anula, dado que el segundo factor es cero. Similarmente, el polinomio se anula en $x_3, x_4, \ldots, x_n$. Ahora, ¿cuánto vale este polinomio en x_0?

$$\tilde{P}_0(x_0) = (x_0 - x_1)(x_0 - x_2)(x_0 - x_3)\cdots(x_0 - x_n) \neq 0,$$

Aunque es posible concluir que el valor es no nulo, no es posible asegurar que su valor sea uno. Como un segundo intento se tiene:

$$\tilde{P}_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)\cdots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)\cdots(x_0 - x_n)}.$$

Notar que este nuevo polinomio se anula en x_1, en x_2, ..., x_n, por la misma razón de antes, y cuando se evalúa en x_0 se obtiene:

$$\tilde{P}_0(x_0) = \frac{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)\cdots(x_0 - x_n)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)\cdots(x_0 - x_n)} = 1,$$

con lo cual $\tilde{P}_0(x)$ es el polinomio $\ell_0(x)$ que se deseaba construir.

El comportamiento de este polinomio en los x_i es muy particular y por esto tiene un nombre especial, es uno de los llamados *polinomios de Lagrange* definido como:

$$\ell_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)\cdots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)\cdots(x_0 - x_n)},$$

y se comporta de la siguiente manera:

$$\ell_0(x_j) = \begin{cases} 1 & \text{si } j = 0 \\ 0 & \text{si } j \neq 0 \end{cases}, \qquad \text{para } j = 0, 1, 2, \cdots, n.$$

Notar que este polinomio tiene n factores lineales en el numerador, y por tanto $\ell_0(x)$ es de grado n. También, por comodidad, se puede expresar en la siguiente forma:

$$\ell_0(x) = \prod_{\substack{j=0 \\ j\neq 0}}^{n} \frac{(x - x_j)}{(x_0 - x_j)}$$

Si a partir de $\ell_0(x)$ se define $P_0(x) = y_0\ell_0(x)$ entonces:

$$P_0(x) = \begin{cases} y_0 & \text{si } x = x_0 \\ 0 & \text{si } x = x_j, \quad j = 1, 2, 3, \ldots, n \end{cases}$$

- De igual forma se construye $P_1(x)$:

$$P_1(x) = y_1\ell_1(x) = y_1 \prod_{\substack{j=0 \\ j\neq 1}}^{n} \frac{(x - x_j)}{(x_1 - x_j)},$$

o en forma equivalente

$$P_1(x) = y_1 \frac{(x-x_0)(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)},$$

el cual se comporta como:

$$P_1(x) = \begin{cases} y_1 & \text{si } x = x_1 \\ 0 & \text{si } x = x_j, \quad j = 0, 2, 3, \ldots, n \end{cases}$$

- Siguiendo este proceso se construyen P_2, P_3,$\cdots$, P_n donde para cada $i = 0, 1, 2, \ldots, n$ el polinomio $P_i(x)$ satisface:

$$P_i(x_j) = \begin{cases} y_i & \text{si } j = i \\ 0 & \text{si } j \neq i \end{cases}, \qquad \text{para } j = 0, 1, 2, \cdots, n.$$

Se concluye que el polinomio $P(x)$ que interpola todos los $n+1$ puntos es la suma de los P_i, con lo cual se tiene la construcción del *polinomio de interpolación de Lagrange*:

$$P(x) = P_0(x) + P_1(x) + \cdots + P_n(x) = y_0\ell_0(x) + y_1\ell_1(x) + \cdots + y_n\ell_n(x),$$

que toma la siguiente forma usando notación de sumas y productos:

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

El grado de $P(x)$ es a lo más n, pues se tiene una suma de términos de grado n.

Ejemplo 20. Encontrar el polinomio de interpolación de Lagrange que ajusta los datos $(-1, 2)$, $(0, -1)$, $(1, 1)$ y $(2, -2)$.

Solución: se puede verificar fácilmente que $x_i \neq x_j$ si $i \neq j$. Nombrando a $(-1, 2)$ como (x_0, y_0), $(0, -1)$ como (x_1, y_1) y así sucesivamente, se construyen los polinomios de Lagrange como sigue.

Primer polinomio:

$$\ell_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)}.$$

Al operar y simplificar

$$\ell_0(x) = \frac{x(x-1)(x-2)}{-6}.$$

Segundo polinomio:

$$\ell_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)}.$$

Al operar y simplificar

$$\ell_1(x) = \frac{(x^2-1)(x-2)}{2}.$$

Tercer polinomio:

$$\ell_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x+1)(x-0)(x-2)}{(1+1)(1-0)(1-2)}.$$

Al operar y simplificar

$$\ell_2(x) = \frac{x(x+1)(x-2)}{-2}.$$

Cuarto polinomio:

$$\ell_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x+1)(x-0)(x-1)}{(2+1)(2-0)(2-1)}.$$

Al operar y simplificar

$$\ell_3(x) = \frac{x(x^2-1)}{6}.$$

Al calcular todos los polinomios anteriores, se procede a ensamblar el polinomio de interpolación de Lagrange con la ecuación $P(x) = y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x)$. Al reemplazar los correspondientes valores se obtiene

$$P(x) = 2\frac{x(x-1)(x-2)}{-6} + (-1)\frac{(x^2-1)(x-2)}{2} + 1\frac{x(x+1)(x-2)}{-2} + (-2)\frac{x(x^2-1)}{6},$$

de donde, al simplificar se obtiene finalmente

$$P(x) = -\frac{x(x-1)(x-2)}{3} - \frac{(x^2-1)(x-2)}{2} - \frac{x(x+1)(x-2)}{2} - \frac{x(x^2-1)}{3}.$$

♢

Ejercicios 6

1. Construir el interpolador de Lagrange para aproximar $f(0)$ si $f(-2) = -1$, $f(-1) = 0$, $f(1) = 2$, $f(2) = 3$.

2. Construir el interpolador de Lagrange para aproximar $f(8.4)$ si $f(8.0) = 1.25$, $f(8.2) = 1.76$, $f(8.3) = 1.46$, $f(8.5) = 1.75$.

3. Determinar el polinomio de interpolación de la función $f(x) = x^3 - 1$ en los puntos $x_0 = -1$, $x_1 = 0$, $x_2 = 1$ y $x_3 = 2$.

4. Si $f(x) = \cos^{-1}(x)$, $x_0 = 0$, $x_1 = 0.5$, $x_2 = 0.8$ y $x_3 = 1$, utilizar el interpolador de Lagrange para aproximar $f(0.65)$ y comparar con el valor real.

5. Utilizar un interpolador de Lagrange para obtener una aproximación de $\sqrt{7}$ con la función $f(x) = 7^x$ y los puntos $x_0 = -2$, $x_1 = -1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 2$.

6. Dados los siguientes datos:

x_i	y_i
0.5	−0.69314
0.8	−0.22314
1.2	0.18232
1.4	0.33647
1.6	0.47000
1.8	0.58778
2.0	0.69314

   utilizar el interpolador de Lagrange para estimar la imagen de 0.9 y 1.7.

7. Sea $P(x)$ el polinomio de Lagrange que interpola los puntos (x_0, y_0), (x_1, y_1) y $Q(x)$ el polinomio de Lagrange que interpola los nodos (x_2, y_2) y (x_3, y_3). Construir un interpolador, basado en $P(x)$ y $Q(x)$ para (x_0, y_0), (x_1, y_1), (x_2, y_2) y (x_3, y_3) asumiendo que $x_i \neq x_j$ para todo $i \neq j$.

8. Determinar el coeficiente de x^3 en el polinomio de interpolación de Lagrange para los datos $(1, 1)$, $(2, 2)$, $(3, 3)$ y $(4, 5)$.

9. Determinar la cantidad de divisiones y multiplicaciones necesarias para construir el polinomio de interpolación de Lagrange de tres puntos.

10. ¿Qué sucede en el proceso de construcción del polinomio de Lagrange, si para un conjunto de datos $(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)$ se tiene que $x_i = x_j$ para algún $i \neq j$?

3.2.2. Polinomio de interpolación de Newton

El polinomio de interpolación de Lagrange puede presentar los siguientes inconvenientes:

1. Si el número de puntos a interpolar es grande, el grado del polinomio resultante puede ser alto y presentar fuertes oscilaciones.

2. Agregar o quitar un punto, implica hacer de nuevo todo el cálculo.

En la propuesta de Newton, primero se hace pasar un polinomio de grado cero por uno de los puntos, y desde el anterior se construye un polinomio de grado uno que pasa por otro punto de la lista. Desde los dos últimos puntos, se construye un polinomio de grado dos

que pasa por un tercer punto de la lista y así sucesivamente. De esta manera, se respeta el trabajo anterior, y no se tienen dificultades para agregar puntos a la lista. A continuación se describe el proceso en forma detallada para los tres primeros puntos. En la figura 3.3 se muestran algunos puntos de la lista, donde $y_i = f(x_i)$ para cierta función f.

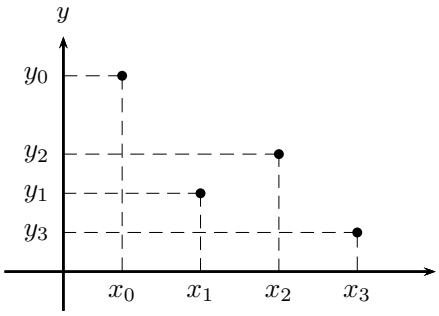


Figura 3.3: polinomio de interpolación de Newton.

El polinomio $P_0(x)$ se construye de manera que pase exactamente por el punto (x_0, y_0), de donde la propuesta más simple es $P(x) = P_0(x) = y_0$, cuya gráfica se presenta en la figura 3.4.

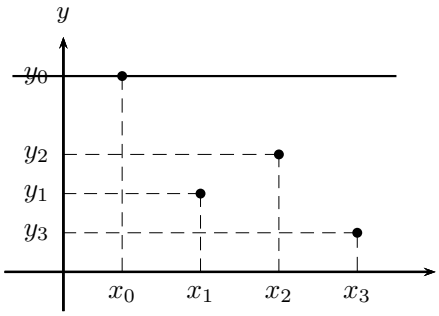


Figura 3.4: polinomio de interpolación de Newton. Caso constante.

Ahora se construye un nuevo polinomio que pase por (x_1, y_1), pero que parta de $P_0(x)$. Este nuevo polinomio de grado uno tiene la forma $P_1(x) = P_0(x) + c_1(x - x_0)$. Notar que este nuevo polinomio pasa por (x_0, y_0) (lo asegura el factor lineal) y se quiere también pase por (x_1, y_1), de donde se debe ajustar el valor de la constante c_1.

Como $P_1(x)$ debe interpolar a (x_1, y_1), al evaluar en x_1 debe dar como resultado y_1 y por tanto: $P_1(x_1) = y_0 + c_1(x_1 - x_0) = y_1$. Al despejar c_1 se obtiene:

$$c_1 = \frac{y_1 - y_0}{x_1 - x_0},$$

expresión conocida como *diferencia dividida de orden uno* y se denota como:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}.$$

Para unificar la notación, decimos que $y_0 = f[x_0]$ es una *diferencia dividida de orden cero*, y el polinomio de interpolación con dicha notación sería:

$$P_1(x) = f[x_0] + f[x_0, x_1](x - x_0).$$

Notar que se utilizó el trabajo hecho con el primer punto. En esta situación, la gráfica se presenta en la figura 3.5. De igual forma, se construye un polinomio que capture otro punto

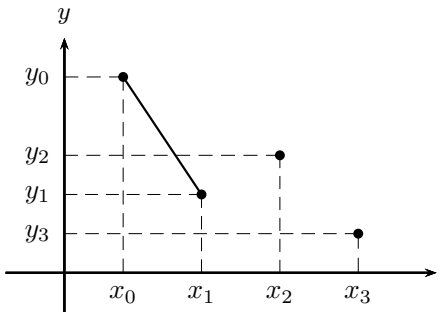


Figura 3.5: polinomio de interpolación de Newton. Primer grado.

más, es decir, que pase por (x_0, y_0), por (x_1, y_1) y por (x_2, y_2) sin perder el trabajo que se ha hecho hasta el momento. Dado $P_2(x) = f[x_0] + f[x_0, x_1](x - x_0) + c_2(x - x_0)(x - x_1)$, notar que al evaluar en x_0, los dos últimos términos se anulan, quedando solamente $P_0(x)$, cuyo valor es y_0. Igualmente, al evaluar en x_1, el último término se anula, quedando el polinomio $P_1(x)$ ya calculado, cuyo valor es y_1 cuando se evalúa en x_1.

Para encontrar el coeficiente c_2, se debe tener en cuenta que, evaluar $P_2(x)$ en x_2 resulte en y_2. Al hacer lo anterior, se obtiene: $P_2(x_2) = f[x_0] + f[x_0, x_1](x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2$. Al despejar y ordenar adecuadamente se obtiene

$$c_2 = \frac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}.$$

Notar que en el numerador hay diferencias divididas de orden uno y escribiendo en notación de diferencias, se tiene

$$c_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0},$$

expresión llamada *diferencia dividida de orden dos*. El polinomio de interpolación en notación de diferencias divididas quedaría

$$P_2(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

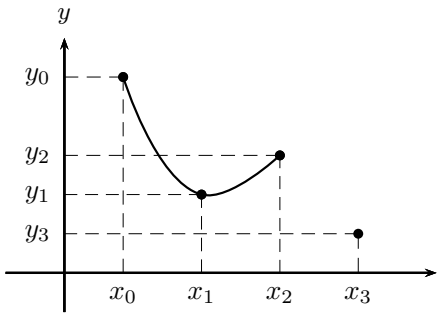


Figura 3.6: polinomio de interpolación de Newton. Segundo grado.

La gráfica de este nuevo resultado se presenta en la figura 3.6.

Similar al procedimiento anterior, se puede inducir la forma para el polinomio de grado tres que interpola cuatro puntos cuyo comportamiento se ilustra en la figura 3.7.

$$P_3(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2)$$

donde:

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}.$$

Al expandir las diferencias divididas de orden dos, se tiene

$$f[x_0, x_1, x_2, x_3] = \frac{\dfrac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} - \dfrac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}}{x_3 - x_0},$$

y expandiendo a diferencias de primer orden, finalmente el coeficiente c_3 se reduce a:

$$f[x_0, x_1, x_2, x_3] = \frac{\dfrac{\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1} - \dfrac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}}{x_3 - x_0}.$$

Nota: en este instante, puede existir la sensación que el cálculo de los coeficientes c_i es una tarea dispendiosa y susceptible de errores, pero si los datos se organizan en una tabla, el cálculo de estos coeficientes es relativamente rápido. Para ver lo anterior, se presenta el siguiente ejemplo.

Ejemplo 21. Encontrar el polinomio de interpolación de Newton que ajusta los datos $(-1, 2)$, $(0, -1)$, $(1, 1)$ y $(2, -2)$.

Solución: se deben calcular las diferencias divididas desde orden cero hasta orden tres que también se denotan Δ^0, Δ^1, Δ^2 y Δ^3 en otros contextos. Las diferencias de orden cero $f[x_i]$, corresponden a los valores y_i de los puntos y se tienen en el cuadro 3.1.

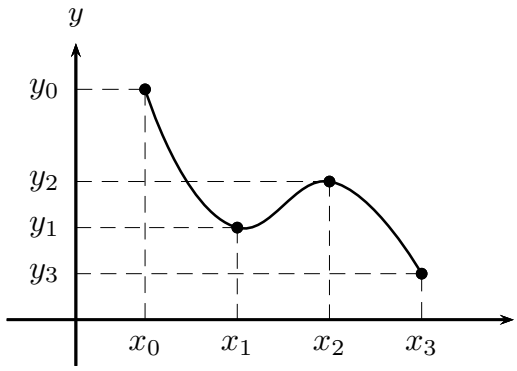


Figura 3.7: polinomio de interpolación de Newton. Tercer grado.

i	x_i	Δ^0
0	-1	2
1	0	-1
2	1	1
3	2	-2

Cuadro 3.1: cálculo de diferencias divididas de orden cero.

Para determinar las diferencias de orden uno, se debe calcular $f[x_{i-1}, x_i] = \dfrac{f[x_i] - f[x_{i-1}]}{x_i - x_{i-1}}$ para $i = 1, 2, 3$. Calculando cada $f[x_{i-1}, x_i]$ y recordando que $f[x_j]$ corresponde a y_j, se obtienen los resultados del cuadro 3.2.

i	x_i	Δ^0	Δ^1
0	-1	2	
1	0	-1	$f[x_0, x_1] = \dfrac{y_1 - y_0}{x_1 - x_0}$
2	1	1	$f[x_1, x_2] = \dfrac{y_2 - y_1}{x_2 - x_1}$
3	2	-2	$f[x_2, x_3] = \dfrac{y_3 - y_2}{x_3 - x_2}$

Cuadro 3.2: cálculo de diferencias divididas de orden uno.

Al usar los valores numéricos correspondientes, se obtienen los resultados del cuadro 3.3. Ahora, se agrega una columna con las diferencias de orden dos, que contiene los valores $f[x_{i-2}, x_{i-1}, x_i]$ para $i = 2, 3$. Como $f[x_{i-2}, x_{i-1}, x_i] = \dfrac{f[x_{i-1}, x_i] - f[x_{i-2}, x_{i-1}]}{x_i - x_{i-2}}$, se tienen entonces los resultados simbólicos en el cuadro 3.4 y los numéricos en el 3.5. Por

i	x_i	Δ^0	Δ^1
0	-1	2	
1	0	-1	-3
2	1	1	2
3	2	-2	-3

Cuadro 3.3: cálculo de diferencias divididas de orden uno.

último, se construye la columna Δ^3, que contiene a los valores de $f[x_0, x_1, x_2, x_3] = \dfrac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}$. Los resultados se encuentran en los cuadros 3.6 y 3.7.

i	x_i	Δ^0	Δ^1	Δ^2
0	-1	2		
1	0	-1	-3	
2	1	1	2	$f[x_0, x_1, x_2] = \dfrac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$
3	2	-2	-3	$f[x_1, x_2, x_3] = \dfrac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}$

Cuadro 3.4: cálculo de diferencias divididas de orden dos.

i	x_i	Δ^0	Δ^1	Δ^2
0	-1	2		
1	0	-1	-3	
2	1	1	2	$\frac{5}{2}$
3	2	-2	-3	$-\frac{5}{2}$

Cuadro 3.5: cálculo de diferencias divididas de orden dos.

Una vez calculadas todas las diferencias divididas, se procede a escribir el polinomio de interpolación:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2).$$

Notar que los valores $f[x_0], f[x_0, x_1], f[x_0, x_1, x_2], f[x_0, x_1, x_2, x_3]$ corresponden a la diagonal del cuadro 3.7. Al reemplazar los correspondientes valores, se concluye que $P(x) = 2 - 3(x + 1) + \frac{5}{2}(x + 1)x - \frac{5}{3}(x + 1)x(x - 1)$. ♢

i	x_i	Δ^0	Δ^1	Δ^2	Δ^3
0	-1	2			
1	0	-1	-3		
2	1	1	2	$\frac{5}{2}$	
3	2	-2	-3	$-\frac{5}{2}$	$f[x_0,x_1,x_2,x_3] = \dfrac{f[x_1,x_2,x_3] - f[x_0,x_1,x_2]}{x_3 - x_0}$

Cuadro 3.6: cálculo de diferencias divididas de tercer orden.

i	x_i	y_i	Δ^1	Δ^2	Δ^3
0	-1	2			
1	0	-1	-3		
2	1	1	2	$\frac{5}{2}$	
3	2	-2	-3	$-\frac{5}{2}$	$-\frac{5}{3}$

Cuadro 3.7: cálculo de diferencias divididas de tercer orden.

Ahora, de manera general, cuando se tienen $n+1$ puntos, el polinomio se puede escribir como

$$\begin{aligned} P(x) = f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) \\ + f[x_0,x_1,x_2,x_3](x-x_0)(x-x_1)(x-x_2) \\ \vdots \\ + f[x_0,\ldots,x_n](x-x_0)\cdots(x-x_{n-1}), \end{aligned}$$

donde, en general, se tiene la siguiente expresión para una diferencia dividida de orden k:

$$f[x_i,x_{i+1},\ldots,x_{i+k}] = \frac{f[x_{i+1},x_{i+2},\ldots,x_{i+k}] - f[x_i,x_{i+1},\ldots,x_{i+k-1}]}{x_{i+k} - x_i}.$$

Notar que el último término de $P(x)$ tiene n factores lineales, de donde el polinomio de interpolación es de grado a lo más n. Para concluir, el siguiente teorema muestra la unicidad del polinomio interpolación, y por tanto la equivalencia entre los polinomios de Lagrange y Newton.

Teorema 6. *Dados $n+1$ puntos (x_0,y_0), (x_1,y_1), (x_2,y_2), $\cdots$, (x_n,y_n) con $x_i \neq x_j$, siempre que $i \neq j$, si $p(x)$ es un polinomio de grado menor o igual a n tal que $p(x_i) = y_i$ para todo $i = 0,1,\ldots,n$, entonces $p(x)$ es único.*

Demostración. Para la prueba de unicidad se supone que existen dos polinomios p y q diferentes de grado n o menor, que interpolan los $n+1$ puntos, es decir, $p(x_i) = y_i$, al igual que $q(x_i) = y_i$, para $i = 0, \ldots, n$. Sea $r(x)$ definido como $r(x) = p(x) - q(x)$. Al evaluar el polinomio r en cada uno de los $n+1$ valores x_i, se tiene

$$r(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0.$$

Ahora, al sumar dos polinomios de grado n, se tiene un polinomio de grado n o menor. Por otro lado, un polinomio de grado n tiene a lo más n raíces reales. En estas circunstancias, la situación presente solo es posible si r es el polinomio cero, y por tanto se concluye que $p(x) = q(x)$. □

Ejercicios 7

1. Construir el interpolador de Newton para aproximar $f(8.4)$, si $f(8.0) = 1.25$, $f(8.2) = 1.76$, $f(8.3) = 1.46$, $f(8.5) = 1.75$.
2. Si $f(x) = \cos^{-1}(x)$, $x_0 = 0$, $x_1 = 0.5$, $x_2 = 0.8$, $x_3 = 1$ utilizar el interpolador de Newton para aproximar $f(0.65)$ y comparar con el valor real.
3. Utilizar un interpolador de Newton para obtener una aproximación de $\sqrt{7}$ con la función $f(x) = 7^x$ y los nodos $x_0 = -2$, $x_1 = -1$, $x_2 = 0$, $x_3 = 1$ y $x_4 = 2$.
4. Si el polinomio de interpolación de Newton para los datos $(1, y_0)$, $(2, y_1)$, $(3, y_2)$, $(4, y_3)$ es $p(x) = -1 + 1(x-1) - 1(x-1)(x-2) + \frac{2}{3}(x-1)(x-2)(x-3)$, determinar el polinomio de interpolación para los datos $(1, y_0)$, $(2, y_1)$, $(3, y_2)$, $(4, y_3)$ y $(5, 1)$.
5. Construir el interpolador de Newton de un conjunto de datos, si se conoce que la tabla de diferencias divididas es:

x	y	Δ^1	Δ^2	Δ^3
-1	4			
2	?	1		
1	?	?	-2	
-2	?	?	?	-3

6. Completar la siguiente tabla de diferencias divididas:

x	y	Δ^1	Δ^2	Δ^3
-1	?			
2	4	1		
?	2	?	?	
-2	-1	1	$\frac{1}{4}$	?

7. Si $P(x)$ es el interpolador de Newton de un conjunto de datos, $Q(x)$ el interpolador de Lagrange sobre el mismo conjunto de datos entonces, ¿cuál es el valor de $P(z) - Q(z)$, con z un valor en el conjunto de datos?

3.2.3. Trazadores cúbicos

Un posible problema que pueden presentar los polinomios de interpolación de Lagrange y Newton, es la posibilidad que el grado del polinomio sea alto, y adicionalmente se presenten fuertes oscilaciones en puntos muy cercanos.

Una solución es ordenar los puntos x_i, y en cada subintervalo $[x_i, x_{i+1}]$, usar un polinomio de grado lo más bajo posible. La opción más inmediata es hacer uso de polinomios de grado cero, pero lo anterior no es aceptable si se desea tener continuidad en la función resultante. La siguiente opción es polinomios de grado uno, lo que garantiza continuidad de la función. Aunque esta opción no es mala, el hecho que la primera derivada de la curva no sea continua, impide describir eventos físicos que requieren continuidad hasta la segunda derivada. La opción indicada es usar polinomios de tercer grado, lo que conlleva a la técnica de los *trazadores cúbicos*.

Para ello, dada una lista *ordenada* de $n+1$ puntos (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) que cumplen las hipótesis de interpolación[1], se desea construir una función a trozos

$$S(x) = \begin{cases} s_0(x) & \forall x \in [x_0, x_1] \\ s_1(x) & \forall x \in [x_1, x_2] \\ s_2(x) & \forall x \in [x_2, x_3] \\ \quad \vdots & \\ s_{n-1}(x) & \forall x \in [x_{n-1}, x_n] \end{cases} \tag{3.1}$$

donde los $s_i(x)$ son n polinomios cúbicos y satisfacen las siguientes condiciones:

1. **Condición de interpolación:** cada polinomio interpola dos puntos, es decir $s_i(x_i) = y_i$, y también $s_i(x_{i+1}) = y_{i+1}$. Como son n polinomios, entonces esta condición ofrece $2n$ ecuaciones. Notar que está implícito que $s_{i-1}(x_i) = s_i(x_i) = y_i$, asegurando la continuidad del trazador.

2. **Condición de la primera derivada:** en los puntos internos $(n-1)$ se debe asegurar que los polinomios que se encuentran, lleven la misma dirección, es decir que cumplan $s'_{i-1}(x_i) = s'_i(x_i)$. Esta condición provee $n-1$ ecuaciones.

3. **Condición de la segunda derivada**: también para nodos internos es conveniente asegurar que los polinomios que se encuentren tengan la misma concavidad en los nodos, es decir $s''_{i-1}(x_i) = s''_i(x_i)$. Esta condición, al igual que la anterior, provee $n-1$ ecuaciones.

Sumando el número de ecuaciones obtenidas de las condiciones mencionadas con anterioridad, se obtiene:

[1] Es decir, $x_i \neq x_j$ siempre que $i \neq j$.

	$2n$	condición de interpolación
	$n-1$	condición de dirección
$+$	$n-1$	condición de concavidad
total	$4n-2$	ecuaciones

Como cada polinomio $s_i(x)$ es de la forma $a_i + b_i x + c_i x^2 + d_i x^3$, entonces es necesario calcular $4n$ coeficientes para construir el trazador $S(x)$, pero se disponen de $4n-2$ ecuaciones, y por tanto faltan dos condiciones que ayuden a equilibrar el sistema. Más adelante se indicará una solución a esta situación. Luego de establecer la forma y condiciones que debe satisfacer el trazador cúbico, el paso siguiente es realizar su construcción.

Para facilitar la construcción del trazador cúbico, se seleccionan polinomios cúbicos $s_i(x)$ de la forma $a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3$, con lo cual:

$$S(x) = \begin{cases} s_0(x) = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3 & \forall x \in [x_0, x_1] \\ s_1(x) = a_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 & \forall x \in [x_1, x_2] \\ s_2(x) = a_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 & \forall x \in [x_2, x_3] \\ \quad \vdots \\ s_{n-1}(x) = a_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3 & \forall x \in [x_{n-1}, x_n] \end{cases}$$

Ahora, aplicando la condición de interpolación en cada polinomio, se obtiene:

$$s_i(x_i) = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3 = y_i$$
$$s_i(x_{i+1}) = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = y_{i+1}.$$

Ahora, de la primera ecuación se concluye que $a_i = y_i$ para todo $i = 0, 1, 2, \ldots, n-1$, y si se define $h_i = x_{i+1} - x_i$ y se sustituye en la segunda ecuación, se obtiene la relación

$$a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1}.$$

Para aplicar ahora la condición de la primera derivada, se debe determinar $S'(x)$, que en este caso corresponde a

$$S'(x) = \begin{cases} s_0'(x) = b_0 + 2c_0(x-x_0) + 3d_0(x-x_0)^2 & \forall x \in [x_0, x_1] \\ s_1'(x) = b_1 + 2c_1(x-x_1) + 3d_1(x-x_1)^2 & \forall x \in [x_1, x_2] \\ s_2'(x) = b_2 + 2c_2(x-x_2) + 3d_2(x-x_2)^2 & \forall x \in [x_2, x_3] \\ \quad \vdots \\ s_{n-1}'(x) = b_{n-1} + 2c_{n-1}(x-x_{n-1}) + 3d_{n-1}(x-x_{n-1})^2 & \forall x \in [x_{n-1}, x_n] \end{cases}$$

Asumiendo entonces que $s_{i+1}'(x_{i+1}) = s_i'(x_{i+1})$ para $i = 0, 1, \ldots, n-2$, se tiene que

$$b_{i+1} + 2c_{i+1}(x_{i+1} - x_{i+1}) + 3d_{i+1}(x_{i+1} - x_{i+1})^2 = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2,$$

de donde al simplificar y sustituir $x_{i+1} - x_i$ por h_i se concluye que

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2.$$

Para la condición de la segunda derivada, se debe calcular $S''(x)$:

$$S''(x) = \begin{cases} s_0''(x) = 2c_0 + 6d_0(x - x_0) & \forall x \in [x_0, x_1] \\ s_1''(x) = 2c_1 + 6d_1(x - x_1) & \forall x \in [x_1, x_2] \\ s_2''(x) = 2c_2 + 6d_2(x - x_2) & \forall x \in [x_2, x_3] \\ \quad \vdots & \\ s_{n-1}''(x) = 2c_{n-1} + 6d_{n-1}(x - x_{n-1}) & \forall x \in [x_{n-1}, x_n] \end{cases}$$

Asumiendo entonces que $s_{i+1}''(x_{i+1}) = s_i''(x_{i+1})$ para $i = 0, 1, \ldots, n-2$, se tiene entonces

$$c_{i+1} = c_i + 3d_i h_i.$$

En resumen, para que el trazador cúbico $S(x)$ cumpla las condiciones deseadas, los coeficientes de los polinomios deben satisfacer las siguientes ecuaciones.

$$a_i = y_i \tag{3.2}$$

$$a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1} \tag{3.3}$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \tag{3.4}$$

$$c_{i+1} = c_i + 3d_i h_i \tag{3.5}$$

Como se indicó al inicio de esta sección, el conjunto anterior contiene $4n - 2$ ecuaciones. Para hallar $4n$ coeficientes, y por tanto equilibrar el sistema de ecuaciones, se adicionan las condiciones $s_0''(x_0) = 0$ y $s_{n-1}''(x_n) = 0$, denominadas *condiciones de frontera libre o natural*, las cuales son equivalentes a $c_0 = 0$ y $c_n = c_{n-1} + 3d_{n-1}h_{n-1} = 0$.

Ahora, con el objetivo de reducir el sistema de ecuaciones, se realizan las siguientes consideraciones.

- Despejar d_i en la ecuación 3.5 y concluir que $d_i = \dfrac{(c_{i+1} - c_i)}{3h_i}$ para $i = 0, 1, \ldots, n-2$.
- Sustituir d_i en las ecuaciones 3.3 y 3.4 para obtener

$$a_{i+1} = a_i + b_i h_i + \frac{h_i^2}{3}(2c_i + c_{i+1}) \tag{3.6}$$

$$b_{i+1} = b_i + h_i(c_i + c_{i+1}). \tag{3.7}$$

- Disminuir en una unidad el índice de 3.7

$$b_i = b_{i-1} + h_{i-1}(c_{i-1} + c_i). \tag{3.8}$$

- Al despejar b_i de la ecuación 3.6 y reemplazar adecuadamente en 3.8 se obtiene la siguiente relación válida para $i = 1, 2, \ldots, n-1$.

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) \tag{3.9}$$

La ecuación (3.9) relaciona los coeficientes c_i en un sistema de $n+1$ ecuaciones con $n+1$ incógnitas. Para construir el trazador cúbico natural de los puntos (x_0, y_0), (x_1, y_1),..., (x_n, y_n), primero se debe resolver el sistema

$$A\mathbf{c} = \mathbf{s} \tag{3.10}$$

donde:

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix}$$

$$\mathbf{s} = \begin{pmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \frac{3}{h_2}(a_3-a_2) - \frac{3}{h_1}(a_2-a_1) \\ \vdots \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{pmatrix}$$

Para terminar, es necesario calcular el valor de b_i y d_i utilizando las ecuaciones $d_i = \dfrac{(c_{i+1}-c_i)}{3h_i}$ y $b_i = \dfrac{1}{h_i}(a_{i+1}-a_i) - \dfrac{h_i}{3}(2c_i + c_{i+1})$. El siguiente ejemplo presenta en un caso concreto la construcción un trazador cúbico natural.

Ejemplo 22. Dados los datos $(-1,2)$, $(0,-1)$, $(2,2)$, $(3,2)$ y $(7,-1)$, construir su trazador cúbico natural asociado.

Solución: lo primero que se debe hacer es ordenar de menor a mayor los puntos por la coordenada x. Una vez ordenados, se disponen en el cuadro 3.8 y se calculan los elementos que conformarán el sistema de ecuaciones 3.10.

i	x_i	a_i	h_i	$\frac{3}{h_{i-1}}(a_i - a_{i-1}) - \frac{3}{h_{i-2}}(a_{i-1} - a_{i-2})$
0	-1	2	1	
1	0	-1	2	13.5
2	2	2	1	-4.5
3	3	2	4	-2.25
4	7	-1		

Cuadro 3.8: cálculo de un trazador cúbico.

Dado lo anterior, el sistema para determinar los coeficientes c_i es

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 6 & 2 & 0 & 0 \\ 0 & 2 & 6 & 1 & 0 \\ 0 & 0 & 1 & 10 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 13.5 \\ -4.5 \\ -2.25 \\ 0 \end{pmatrix},$$

de donde se concluye que

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 2.8089 \\ -1.6767 \\ -0.0573 \\ 0 \end{pmatrix}.$$

Al utilizar las relaciones $d_i = \dfrac{(c_{i+1} - c_i)}{3h_i}$ y $b_i = \dfrac{1}{h_i}(a_{i+1} - a_i) - \dfrac{h_i}{3}(2c_i + c_{i+1})$ se obtiene

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 0.9363 \\ -0.7476 \\ 0.5398 \\ 0.0047 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} -3.9363 \\ -1.1273 \\ 1.1369 \\ -0.5971 \\ 0 \end{pmatrix}$$

Conociendo los coeficientes de los polinomios $s_i(x)$, se construye el trazador cúbico natural.

$$S(x) = \begin{cases} s_0(x) = 2 - 3.9363(x+1) + 0.9363(x+1)^3 & \text{si } x \in [-1,0] \\ s_1(x) = -1 - 1.1273x + 2.8089x^2 - 0.7476x^3 & \text{si } x \in [0,2] \\ s_2(x) = 2 + 1.1369(x-2) - 1.6767(x-2)^2 + 0.5398(x-2)^3 & \text{si } x \in [2,3] \\ s_3(x) = 2 - 0.5971(x-3) - 0.0573(x-3)^2 + 0.0047(x-3)^3 & \text{si } x \in [3,7] \end{cases}$$

♢

Ejercicios 8

1. Construir el trazador cúbico natural para los datos $(0,1)$, $(1,1)$ y $(2,7)$.
2. Para la función $f(x) = \ln(x)$, construir el trazador cúbico de los nodos $x_0 = 1$, $x_1 = 2$, $x_2 = 3$. Luego, estimar la imagen de $x = e$.
3. Construir el trazador cúbico natural para estimar $f(8.4)$ si $f(8.0) = 1.25$, $f(8.2) = 1.76$, $f(8.3) = 1.46$ y $f(8.5) = 1.75$.
4. Si $f(x) = \cos^{-1}(x)$, $x_0 = 0$, $x_1 = 0.5$, $x_2 = 0.8$, $x_3 = 1$, utilizar el trazador cúbico natural para aproximar $f(0.65)$ y comparar con el valor real.
5. Utilizar un trazador cúbico natural para obtener una aproximación de $\sqrt{7}$ con la función $f(x) = 7^x$ y los nodos $x_0 = -2$, $x_1 = -1$, $x_2 = 0$, $x_3 = 1$ y $x_4 = 2$.
6. Dado el trazador cúbico natural

$$s(x) = \begin{cases} s_0(x) = 1 + 2x - x^3 & 0 \le x \le 1 \\ s_1(x) = 2 + b(x-1) + c(x-1)^2 + d(x-1)^3 & 1 \le x \le 2 \end{cases}$$

   determinar los valores de b, c y d.
7. Construir el trazador cúbico natural de los datos $(1,1), (-1,-1), (2,8), (-2,-8)$. ¿Tiene algo particular este trazador?
8. Dados los puntos $(1,1), (2,2), (3,1), (4,2)$, construir el trazador cúbico natural asociado y estimar el valor de la derivada en $x = 3$.
9. Dados los puntos $(1,1), (2,2), (3,1), (4,2)$, construir el trazador cúbico natural asociado y determinar el valor máximo del interpolador en el intervalo $[1,4]$.
10. Dado un trazador cúbico natural $s(x)$, ¿a qué equivale $s'''(x)$?

3.3. Ajuste por mínimos cuadrados

Si lo que se desea es marcar una tendencia, los polinomios de ajuste exacto no son los más adecuados. Es mejor buscar una curva más simple, que tal vez no contiene la totalidad de los puntos, pero que pasa cerca de cada uno de ellos como se muestra en la figura 3.8.

3.3.1. Errores

En la búsqueda de encontrar una curva cercana a una serie de puntos, es necesario definir una manera de *medir* el error que se comete al seleccionar una función $\bar{f}(x)$. Existen varias posibilidades.

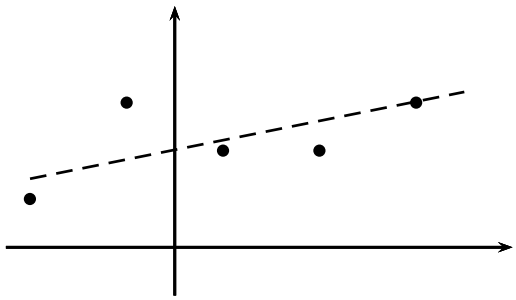


Figura 3.8: ajuste por mínimos cuadrados.

Error relativo: suponer que se tienen $n+1$ puntos puntos (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) y que estos se ajustan con la función $\bar{f}(x)$. El error relativo que se comete con la función de ajuste es $E_r = \sum_{i=0}^{n} \left(y_i - \bar{f}(x_i)\right)$. Un problema en esta situación es de compensación, pues pueden existir errores grandes que al sumarlos con otros de igual magnitud pero de signo contrario, se cancelen y pareciera que se tiene un pequeño o inclusive nulo error.

Error absoluto: para evitar que el error se compense, se toma el valor absoluto y el error se puede calcular como $E_a = \sum_{i=0}^{n} \left|y_i - \bar{f}(x_i)\right|$. Este error presenta dos problemas. Primero, puede ser que una curva ajuste bien una lista de puntos, pero la suma de errores pequeños en cada punto puede finalmente arrojar un error grande, y segundo, el hecho que la función valor absoluto presente problemas de diferenciabilidad.

Error cuadrático: esta manera de medir el error tiene la virtud de no presentar problemas de diferenciabilidad y se define como $E_c = \sum_{i=0}^{n} \left(y_i - \bar{f}(x_i)\right)^2$. Además, para valores $|y_i - \bar{f}(x_i)| < 1$ se tiene $\left(y_i - \bar{f}(x_i)\right)^2 \leq |y_i - \bar{f}(x_i)|$, y por tanto errores "pequeños" se transforman en valores inclusive menores.

3.3.2. Funciones de ajuste

En la selección de la función de ajuste $\bar{f}(x)$, normalmente se escoge una combinación lineal de familias de funciones *base*. Algunas de las familias más utilizadas en la práctica incluyen:

Monomios: $\{1, x, x^2, \ldots\}$.

Exponenciales: $\{1, e^{\pm x}, e^{\pm 2x}, \ldots\}$.

Exponenciales complejas: $\{1, e^{\pm ix}, e^{\pm 2ix}, \ldots\}$.

Funciones seno y coseno: $\{1, \cos x, \cos 2x, \ldots; \sin x, \sin 2x, \ldots\}$.

Cada uno de estos conjuntos se selecciona según la naturaleza de los datos a trabajar. En la próxima sección se estudian funciones de ajuste $\bar{f}(x)$ que son combinaciones lineales de la familia de los *monomios*.

3.3.3. Polinomios de mínimos cuadrados

Cuando se usa la familia de los monomios $\{1, x, x^2, \ldots\}$, combinaciones lineales de elementos de esta base producen un polinomio, y el caso más simple es cuando se usan los primeros dos elementos de la base, obteniéndose polinomios de grado uno. Ahora, la idea entonces es encontrar una recta $a_0 + a_1 x$ que pase cerca de los puntos (x_0, y_0), (x_1, y_1),..., (x_n, y_n). Si se observa el error cuadrático E entre la curva $\bar{f}(x) = a_0 + a_1 x$ y la lista de puntos, dicha curva presenta un error cuadrático dado por la función de dos variables

$$E(a_0, a_1) = \sum_{i=0}^{n} (y_i - a_0 - a_1 x_i)^2.$$

Ahora la intención es hacer mínimo dicho error, así que hay que buscar dicho valor usando las técnicas usuales del cálculo vectorial. Para tal efecto, notar que $E(a_0, a_1)$ es una función cuadrática y su gráfica corresponde a un paraboloide que se abre hacia arriba. Luego, tiene un único punto crítico y corresponde a su mínimo absoluto. Para calcular el punto crítico, se debe determinar cuando el gradiente

$$\nabla E = \frac{\partial E}{\partial a_0}\mathbf{i} + \frac{\partial E}{\partial a_1}\mathbf{j}$$

de la función E es el vector nulo. En tal caso se obtienen las dos ecuaciones normales

$$\frac{\partial E}{\partial a_0} = 0 \qquad \frac{\partial E}{\partial a_1} = 0.$$

A continuación se presenta el cálculo detallado de la primera ecuación.

$$\frac{\partial E}{\partial a_0} = \frac{\partial}{\partial a_0}\left(\sum_{i=0}^{n} (y_i - a_0 - a_1 x_i)^2\right) = 0.$$

Al intercambiar la derivada parcial con la sumatoria

$$\sum_{i=0}^{n} \frac{\partial}{\partial a_0}(y_i - a_0 - a_1 x_i)^2 = 0,$$

y al realizar la derivada se tiene

$$\sum_{i=0}^{n} 2(y_i - a_0 - a_1 x_i)(-1) = 0.$$

Ahora, al factorizar

$$(-2)\left\{\sum_{i=0}^{n}(y_i - a_0 - a_1x_i)\right\} = 0,$$

se obtiene

$$\sum_{i=0}^{n}(y_i - a_0 - a_1x_i) = 0.$$

Por propiedades de la sumatoria se tiene que

$$\sum_{i=0}^{n} y_i - \sum_{i=0}^{n} a_0 - \sum_{i=0}^{n} a_1x_i = 0,$$

de donde se desprende que

$$a_1\sum_{i=0}^{n} x_i + a_0(n+1) = \sum_{i=0}^{n} y_i,$$

la cual corresponde a la primera ecuación normal.

Razonando de la misma manera, pero derivando E respecto a a_1, se llega a

$$a_1\sum_{i=0}^{n} x_i^2 + a_0\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_ix_i.$$

Al tener las dos ecuaciones normales, se llega a un sistema de ecuaciones cuya matriz aumentada es

$$\left(\begin{array}{cc|c} \sum\limits_{i=0}^{n} x_i^2 & \sum\limits_{i=0}^{n} x_i & \sum\limits_{i=0}^{n} y_ix_i \\ \\ \sum\limits_{i=0}^{n} x_i & n+1 & \sum\limits_{i=0}^{n} y_i \end{array}\right).$$

El sistema tiene solución única. Al usar la regla de Cramer se desprende que

$$a_1 = \frac{\begin{vmatrix} \sum\limits_{i=0}^{n} y_ix_i & \sum\limits_{i=0}^{n} x_i \\ \\ \sum\limits_{i=0}^{n} y_i & n+1 \end{vmatrix}}{\begin{vmatrix} \sum\limits_{i=0}^{n} x_i^2 & \sum\limits_{i=0}^{n} x_i \\ \\ \sum\limits_{i=0}^{n} x_i & n+1 \end{vmatrix}} = \frac{(n+1)\sum\limits_{i=0}^{n} y_ix_i - \sum\limits_{i=0}^{n} x_i\sum\limits_{i=0}^{n} y_i}{(n+1)\sum\limits_{i=0}^{n} x_i^2 - \left(\sum\limits_{i=0}^{n} x_i\right)^2} \tag{3.11}$$

$$a_0 = \frac{\begin{vmatrix} \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} y_i x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} y_i \end{vmatrix}}{\begin{vmatrix} \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & n+1 \end{vmatrix}} = \frac{\sum_{i=0}^{n} x_i^2 \sum_{i=0}^{n} y_i - \sum_{i=0}^{n} x_i \sum_{i=0}^{n} y_i x_i}{(n+1)\sum_{i=0}^{n} x_i^2 - \left(\sum_{i=0}^{n} x_i\right)^2} \tag{3.12}$$

Finalmente, estos son los valores de a_0 y a_1 que hacen mínimo el error cuadrático.

Ejemplo 23. Encontrar los valores de a_0 y a_1 que hacen mínimo el error cuadrático cuando se aproxima la lista $(-1, 2)$, $(0, -1)$, $(1, 1)$ y $(2, -2)$ con un polinomio de grado uno.

Solución: se organizan los datos en la siguiente tabla y se suma por columnas.

i	x_i	y_i	x_i^2	$y_i x_i$
0	-1	2	1	-2
1	0	-1	0	0
2	1	1	1	1
3	2	-2	4	-4
$\sum$	2	0	6	-5

Al reemplazar en las ecuaciones 3.11 y 3.12 para a_0 y a_1:

$$a_1 = \frac{(4)(-5) - (2)(0)}{(4)(6) - (2)^2} = \frac{-20}{20} = -1.0,$$

$$a_0 = \frac{(6)(0) - (2)(-5)}{20} = \frac{10}{20} = 0.5.$$

Por lo tanto, la función de ajuste es:

$$\bar{f}(x) = 0.5 - 1.0x.$$

♢

Cuando la función de ajuste es un polinomio cuadrático, $\bar{f}(x) = a_0 + a_1 x + a_2 x^2$, el error es

$$E(a_0, a_1, a_2) = \sum_{i=0}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)^2.$$

Ahora el error depende de las constantes a_0, a_1 y a_2 que se elijan. Para encontrar el valor de aquellas que minimizan el error cuadrático, se iguala el gradiente al vector nulo y lo anterior da origen a tres ecuaciones normales:

$$a_2 \sum_{i=0}^{n} x_i^4 + a_1 \sum_{i=0}^{n} x_i^3 + a_0 \sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} y_i x_i^2,$$

$$a_2 \sum_{i=0}^{n} x_i^3 + a_1 \sum_{i=0}^{n} x_i^2 + a_0 \sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i x_i,$$

$$a_2 \sum_{i=0}^{n} x_i^2 + a_1 \sum_{i=0}^{n} x_i + a_0(n+1) = \sum_{i=0}^{n} y_i,$$

con su correspondiente sistema lineal de ecuaciones en notación de matriz aumentada

$$\left(\begin{array}{ccc|c} \sum_{i=0}^{n} x_i^4 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} y_i x_i^2 \\ \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} y_i x_i \\ \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i & n+1 & \sum_{i=0}^{n} y_i \end{array}\right).$$

Si se observa bien esta matriz, se puede notar que contiene al caso de la recta de mínimos cuadrados, es decir, al sistema del caso anterior. Es más, esta nueva matriz se puede construir agregando una fila a la izquierda y una columna arriba de la matriz aumentada del caso anterior teniendo en cuenta que los exponentes de x_i crecen hacia la izquierda y hacia arriba. Sucede igual cuando la función de ajuste es un polinomio de grado tres, es decir, a la última matriz solo basta agregar una columna a la izquierda y una fila arriba aumentando los exponentes hacia arriba y hacia la izquierda. De esta forma se puede generalizar para polinomios de grado n.

3.3.4. Ajuste exponencial

En algunos casos, para los datos $\{(x_i, y_i)\}_{i=0}^{n}$ es necesario construir una función de ajuste de la forma $\bar{f}(x) = ax^b$. Es necesario encontrar los valores de a y b que hacen mínimo el error cuadrático $E(a,b) = \sum_{i=0}^{n} \left(y_i - ax_i^b\right)^2$. Aunque al calcular las ecuaciones normales rápidamente se observa que es un sistema no lineal, un buen intento para obtener un sistema lineal es aplicar el logaritmo natural a la función de ajuste. En dicho caso, por las propiedades de los logaritmos se obtiene

$$\ln \bar{f}(x) = \ln a + b \ln x.$$

Si se realizan los cambios $\bar{F}(x) = \ln \bar{f}(x)$, $X_i = \ln x_i$, $a_0 = \ln a$ y $a_1 = b$, se tiene una reducción del problema exponencial al caso anterior de la recta de mínimos cuadrados

$$\bar{F}(X) = a_0 + a_1 X.$$

Ejercicios 9

1. Determinar la recta de mínimos cuadrados que corresponde a los datos $(-1, 2)$, $(-2, 3)$, $(1, 2.5)$ y $(-3, 0)$ y su error cuadrático.

2. Determinar la recta de mínimos cuadrados que corresponde a los datos $(10, \ln(10))$, $(100, \ln(100))$, $(1000, \ln(1000))$ y $(10000, \ln(10000))$. Luego estimar la imagen de $x = 5000$ y comparar con el valor real.

3. Para los datos $(1, -1), (2, 1), (3, -1), (4, 1), (5, -1)$:

   - Ubicar en los puntos en el plano y dibujar la recta que se considere es la más cercana a los datos.
   - Determinar la recta de mínimos cuadrados para los puntos dados.

   ¿Concuerda la recta de mínimos cuadrados con la esperada?

4. ¿Cuál es la recta de mínimos cuadrados para los datos $(1, 2)$, $(1, -1)$?

5. Determinar el polinomio cuadrático de mínimos cuadrados que corresponde a los datos $(-1, 2)$, $(-2, 3)$, $(1, 2.5)$ y $(-3, 0)$.

6. Para los siguientes datos, determinar la función $\bar{f}(x) = ax^b$ de mínimos cuadrados.

x_i	y_i
0.03	24.8
0.05	12.3
0.07	6.25
0.09	3.12
0.1	0.75

7. Determinar el error cuadrático al aproximar $(-1, 2.5)$, $(0.5, -2)$ y $(2, -0.2)$ con la función $\bar{f}(x) = \dfrac{1}{3} + \dfrac{5}{3}x^{2/3}$.

8. Si se conoce que el error cuadrático de aproximar $(1, 2.5)$, $(2, 3.5)$, $(-1.5, 0)$ y $(-2, -0.5)$ con una recta es cero, ¿qué se puede decir acerca de los datos?

Capítulo 4

Diferenciación e integración numérica

4.1. Introducción

Tanto en ciencias como en ingeniería, e incluso en otras áreas como la economía, dinámica social, etc., se estudian sistemas que cambian y por tanto existe el interés de entender la manera en que ocurren esos cambios. En la formación básica de ingeniería y ciencias, la matemática del cambio se aborda formalmente en los cursos de cálculo diferencial e integral. Por ejemplo, si se tiene una función $f(x)$ que tiene un comportamiento como el de la figura 4.1, se puede observar que el valor de la función disminuye cuando la variable x cambia desde x_0 hasta x_1.

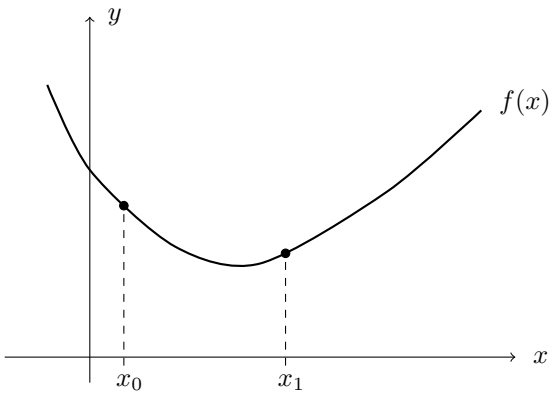


Figura 4.1: cambios en la función $f(x)$.

El cambio en el valor de la función se encuentra dado por $\Delta y = f(x_1) - f(x_0)$. El valor medio de este cambio es

$$\bar{f} = \frac{\Delta y}{\Delta x} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Si se toma $\Delta x = x_1 - x_0$, entonces $x_1 = x_0 + \Delta x$ y la ecuación anterior se transforma en

$$\bar{f} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

Ahora, si Δx se hace tender a cero, o en forma equivalente $x_1 \to x_0$, se llega a la expresión que define la derivada de la función $f(x)$ en x_0:

$$f'(x_0) = \frac{df}{dx}\Big|_{x_0} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}. \tag{4.1}$$

4.2. Aproximaciones a la derivada

Si el conocimiento de la función se tiene mediante una tabla de datos, la definición de la ecuación 4.1 no se puede aplicar. Sin embargo, si los puntos x_i y x_{i+1} están cerca y por lo tanto Δx es pequeño, es posible calcular la primera derivada en x_i con la aproximación:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{\Delta x}. \tag{4.2}$$

Aunque puede ser que la ecuación 4.2 sea una excelente aproximación, no hay conocimiento del error que se comete y esto podría ser una desventaja. Una mejor opción es suponer que la función $f(x)$ y todas sus derivadas se conocen en x_i y por lo tanto al expandir en serie de Taylor alrededor de x_i, el valor de $f(x_{i+1})$ se obtiene mediante la expresión

$$f(x_{i+1}) = f(x_i) + f'(x_i)\,(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}\,(x_{i+1} - x_i)^2 + \cdots \tag{4.3}$$

Al despejar $f'(x_i)$ se tiene que

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(x_i)}{2!}h - \cdots.$$

donde $h = x_{i+1} - x_i$. De lo anterior, se desprende que se puede aproximar la primera derivada con

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h},$$

cometiendo el error

$$E = \frac{f''(x_i)}{2!}h + \cdots.$$

Ahora, notar que si el paso h es pequeño, el primer término domina el error E, y por tanto se tiene que $E = O(h)$ de acuerdo a la notación introducida en la sección 1.6.

Observación: se puede argumentar que la última expresión para aproximar la primera derivada es idéntica a 4.2, sin embargo, la última brinda información sobre el error cometido y por tanto representa una mejora.

Con el fin de unificar la notación, la primera derivada en diferencias hacia adelante es

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}, \tag{4.4}$$

con error $O(h)$. Ahora, si la serie de Taylor 4.3, se hubiera evaluada en x_{i-1}, se tendría en dicho caso

$$f(x_{i-1}) = f(x_i) + f'(x_i)\,(x_{i-1} - x_i) + \frac{f''(x_i)}{2!}\,(x_{i-1} - x_i)^2 + \cdots$$

Al hacer $h = x_i - x_{i-1}$, se obtiene

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \cdots \tag{4.5}$$

Nuevamente, al despejar $f'(x_i)$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \frac{f''(x_i)}{2!}h - \cdots,$$

expresión que también es válida para aproximar la primera derivada y como se puede observar, con término de error $O(h)$. Como se considera el punto inmediatamente anterior, entonces, la expresión

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}, \tag{4.6}$$

se conoce como primera derivada en diferencias hacia atrás.

Ejemplo 24. Dada una función $f(x)$, se conoce la información dada por la tabla 4.1. Calcular $f'(x_2)$ usando tanto diferencias hacia adelante como hacia atrás.

i	x_i	$f(x_i)$
0	0.2	0.45
1	0.3	0.57
2	0.4	0.71
3	0.5	0.88
4	0.6	1.08

Cuadro 4.1: información de $f(x)$.

Solución: notar que $h = 0.1$. Usando diferencias hacia adelante, $f'(x_2) = \dfrac{f(x_3) - f(x_2)}{h}$, de donde $f'(x_2) = \dfrac{0.88 - 0.71}{0.1} = 1.7$. Similarmente, con diferencias hacia atrás, $f'(x_2) = \dfrac{f(x_2) - f(x_1)}{h}$, de donde se concluye que $f'(x_2) = \dfrac{0.71 - 0.57}{0.1} = 1.4$. ♢

En los dos casos vistos hasta ahora, diferencias hacia adelante y diferencias hacia atrás, se ha insistido que el error cometido es proporcional a la magnitud de h y por tanto es de esperarse que si h es pequeño, entonces el error también lo será. El problema es que h no se puede hacer tan pequeño en términos prácticos, pues valores cercanos a cero pueden afectar la precisión de la máquina, o presentarse errores de división por cero. Lo anterior lleva a buscar fórmulas o métodos que puedan lograr una precisión alta sin tener que utilizar un paso muy pequeño.

Para mejorar la anterior situación, se puede observar que si h es pequeño, (por ejemplo, menor que uno) entonces h^2 lo será aún más y h^3 inclusive más. De acuerdo a lo anterior, si se encuentra una fórmula en el que el término de error sea proporcional a una potencia alta de h, seguro se tendrá una manera más confiable de aproximar la primera derivada.

Considerar ahora que las series 4.3 y 4.5

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \cdots$$
$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 + \cdots$$

se restan. Notar que algunos términos se cancelan y por tanto

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + O\big(h^3\big)\,.$$

Al despejar $f'(x_i)$ se obtiene

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O\big(h^2\big)\,, \tag{4.7}$$

con lo que se obtiene una fórmula con error $O\big(h^2\big)$, que de acuerdo a lo argumentado anteriormente, ofrece una mejor aproximación a la primera derivada sin tener que tomar h peligrosamente pequeños. La expresión 4.7, que frecuentemente también se escribe como

$$f'(x_i) = \frac{f(x_i + h) - f(x_i - h)}{2h}, \tag{4.8}$$

se conoce como primera derivada en diferencias centradas. Notar que para un paso h, si se reduce a la mitad, el error se reduce en una cuarta parte, mientras que en los casos de error $O(h)$, la redución sería también a la mitad.

4.3. Extrapolación de Richardson

En el cálculo de la derivada, las fórmulas de la sección 4.2 aunque pueden dar una buena aproximación, dependen del valor de h elevado a cierta potencia n. Como se argumentó en su momento, entre mayor sea el valor de n, se reduce la necesidad de un paso h tan pequeño y con ello los riesgos inherentes asociados a una eventual división por cero.

Por otro lado, cuando se trabaja con una tabla de datos, el valor real V_r de la derivada en general no es posible saberlo, de donde se tendría un error real ε_r definido como el valor real V_r menos el error aproximado V_a:

$$\varepsilon_r = V_r - V_a.$$

En el caso de la primera derivada en diferencias centradas (ecuación 4.8), el error es proporcional al cuadrado del paso h y por tanto

$$V_r = (V_a)_h + Ch^2, \tag{4.9}$$

donde $(V_a)_h$ significa usar un paso h en la ecuación 4.8 y C es una constante desconocida. Si el paso se reduce a la mitad, entonces:

$$V_r = (V_a)_{h/2} + C\left(\frac{h}{2}\right)^2.$$

Notar que la constante C es igual en las últimas dos ecuaciones, ya que esta se calcula con los términos restantes de la serie de Taylor centrada en x_i. También es igual el valor real, pues se quiere calcular la primera derivada en x_i. Dado lo anterior, y al multiplicar por cuatro, la última ecuación se transforma en

$$4V_r = 4(V_a)_{h/2} + Ch^2. \tag{4.10}$$

Al restar la ecuación 4.10 de la ecuación 4.9 se tiene

$$-3V_r = (V_a)_h - 4(V_a)_{h/2},$$

de donde al despejar el valor real V_r, se tiene finalmente que

$$V_r = \frac{4}{3}(V_a)_{h/2} - \frac{1}{3}(V_a)_h.$$

La primera derivada $f'(x_i)$ se puede aproximar entonces como

$$f'(x_i) = \frac{4}{3}(f'_{\text{centrada}})_{h/2} - \frac{1}{3}(f'_{\text{centrada}})_h, \tag{4.11}$$

expresión conocida como *extrapolación de Richardson*, donde $(f'_{\text{centrada}})_{h/2}$, se interpreta como la primera derivada en diferencias centradas con paso $h/2$. Lo interesante de la extrapolación de Richardson, es que se puede demostrar que se obtiene un término de error $O(h^4)$.

Ejercicios 10

1. Usando diferencias hacia adelante y hacia atrás, completar el cuadro 4.2.
2. Usando diferencias centradas, completar el cuadro 4.3.

i	x_i	$f(x_i)$	$f'(x_i)$
0	0.4	1.45	
1	0.5	2.57	
2	0.6	3.71	
3	0.7	4.88	
4	0.8	6.12	

Cuadro 4.2: información de $f(x)$.

i	x_i	$f(x_i)$	$f'(x_i)$
0	-0.2	-1.23	
1	0.0	-2.34	
2	0.2	-0.34	
3	0.4	2.46	
4	0.6	4.35	

Cuadro 4.3: información de $f(x)$.

3. Para el intervalo $[a, b]$, función $f(x)$ y entero positivo N dados, partir uniformente el intervalo en nodos x_i, con lo cual $x_i = a + ih$ para $i = 0, \ldots, N$ donde $h = \dfrac{b-a}{N}$. A continuación calcular $(f'_{\text{centrada}})_h$ y $(f'_{\text{centrada}})_{h/2}$ en los nodos x_i. Calcular finalmente $f'(x_i)$ usando extrapolación de Richardson.

   a) $f(x) = e^x$, $N = 10$ en $[0, 1]$.

   b) $f(x) = \sin x - x$, $N = 10$ en $[0, \pi]$.

   c) $f(x) = x^4 - x^3 + 1$, $N = 20$ en $[2, 6]$.

   d) $f(x) = \ln x + 2x$, $N = 30$ en $[3, 7]$.

   e) $f(x) = \tan x - x^2 + 1$, $N = 20$ en $[-1, 0]$.

4. En las mismas situaciones del ejercicio anterior, calcular los valores reales de $f'(x_i)$ y comparar con los obtenidos usando extrapolación de Richardson.

4.4. Aproximaciones a la integral definida

En el curso de cálculo integral, es usual tratar el problema del área bajo la función $f(x)$ mediante el cálculo de sumas de Riemann. Para ello, suponer que el intervalo $[a, b]$ se ha dividido en subintervalos $[x_{i-1}, x_i]$, donde $x_0 = a$ y $x_N = b$. Si x_i^* es un punto en el intervalo $[x_{i-1}, x_i]$, entonces la suma de Riemann $\displaystyle\sum_{i=1}^{N} f(x_i^*)\,\Delta x_i$, donde $\Delta x_i = x_i - x_{i-1}$,

es una aproximación al área bajo la curva como se muestra en la figura 4.2. En dicho caso, la integral de Riemann de f sobre el intervalo $[a, b]$ se define como

$$\int_a^b f(x)\,dx = \lim_{\text{máx}\,\Delta x_i \to 0} \sum_{i=1}^{N} f(x_i^*)\,\Delta x_i, \tag{4.12}$$

siempre que dicho límite exista. Dado que hacer integrales definidas de la anterior manera no es práctico, el teorema fundamental del cálculo brinda una forma conveniente de hacer las cuentas. Para ello, es necesario calcular primitivas o antiderivadas. El problema es que no todas las funciones tienen primitiva en términos elementales[1]. Basta considerar el conocido caso de la función $f(x) = e^{-x^2}$. Dado lo anterior, es necesario desarrollar métodos que brinden aproximaciones al problema del área bajo la curva. En la secciones 4.5 y 4.6 se exponen algunos de ellos.

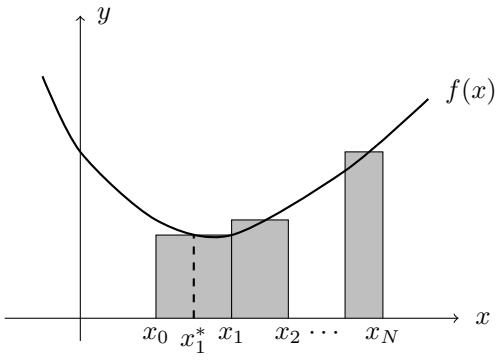


Figura 4.2: sumas de Riemann.

4.5. Regla de los trapecios

La regla de los trapecios se obtiene al aproximar el área bajo la gráfica de $f(x)$ en un intervalo $[a, b]$ mediante un trapecio como se observa en la figura 4.3. Notar que el área sombreada en dicha figura corresponde a $\dfrac{f(a) + f(b)}{2}(b-a)$, y por tanto en dicho intervalo se tiene que

$$\int_a^b f(x)\,dx \approx \frac{f(a) + f(b)}{2}(b - a). \tag{4.13}$$

Ahora, si el intervalo $[a, b]$ se divide en N subintervalos $[x_i, x_{i+1}]$, donde $x_0 = a$ y $x_N = b$,

[1] Al respecto se puede consultar el teorema de Liouville del álgebra diferencial, o también el algoritmo de Risch de integración indefinida.

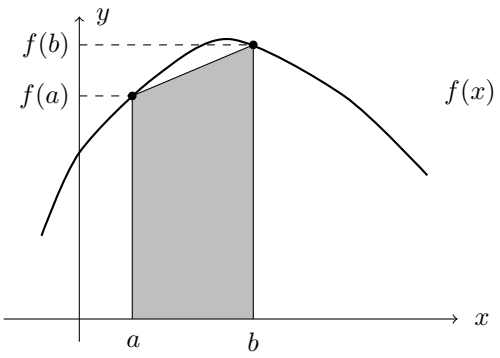


Figura 4.3: regla de los trapecios.

el resultado 4.13 se puede aplicar en cada intervalo $[x_i, x_{i+1}]$ para obtener

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} \frac{f(x_{i-1}) + f(x_i)}{2} \Delta x_i, \tag{4.14}$$

donde $\Delta x_i = x_{i+1} - x_i$. La expresión 4.14 se conoce como *regla del trapecio compuesta*. Ahora, si la partición del intervalo es uniforme, esto es, con todos los subintervalos de igual longitud h, entonces la expresión 4.14 se simplifica en

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{N-1}) + f(x_N)\right], \tag{4.15}$$

la cual se usa en la práctica, pues requiere de menos evaluaciones de la función f que el resultado 4.14.

Sobre el término de error, en el caso de funciones dos veces derivables con continuidad en el intervalo $[a, b]$, se tiene que existe $\xi \in (a, b)$ tal que:

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{N-1} f(x_i) + f(x_N)\right] - \frac{(b-a)}{12} h^2 f''(\xi). \tag{4.16}$$

Ejemplo 25. Estimar un valor de N que permita una aproximación al valor exacto de $\displaystyle\int_{-1}^{1} e^{-x^2}\,dx$ con una precisión menor a $\varepsilon = 10^{-6}$ al usar la regla del trapecio compuesta.

Solución: si $f(x) = e^{-x^2}$, entonces $f''(x) = e^{-x^2}\left(4x^2 - 2\right)$. Haciendo un análisis de máximos y mínimos, se tiene que el máximo global de $f(x)$ es $4/e^{3/2}$ y se alcanza en $x = \pm\sqrt{\frac{3}{2}}$. Similarmente el mínimo global es -2 y se alcanza en $x = 0$. Dado lo anterior, se tiene la siguiente estimación para el término de error:

$$\left|\frac{(b-a)}{12} h^2 f''(\xi)\right| = \frac{1-(-1)}{12} h^2 \left|f''(\xi)\right| \leq \frac{h^2}{3}.$$

Si se desea alcanzar la precisión dada, se tiene que $\frac{h^2}{3} < 10^{-6}$, o en forma equivalente $\frac{4}{3N^2} < 10^{-6}$ pues $h = \frac{2}{N}$. De lo anterior se sigue que $N > \sqrt{\frac{4 \cdot 10^6}{3}} \approx 1154.7$, y por tanto se requiere de $N = 1155$ para obtener la precisión de $\varepsilon = 10^{-6}$. ♢

4.6. Regla de Simpson

En la regla del trapecio, para aproximar el área bajo la curva, se usa interpolación lineal a trozos. Un mejora natural sería usar interpolación de grado superior, y de hecho, si se usan polinomios cuadráticos, se llega a un método conocido como la *regla de Simpson.*

Si se tiene una función f y puntos x_0, x_1 y x_2 (figura 4.4) tales que $h = x_2 - x_1 = x_1 - x_0$, entonces para calcular el área sombreada es necesario integrar el polinomio interpolante

$$P(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

discutido en la sección 3.2.1.

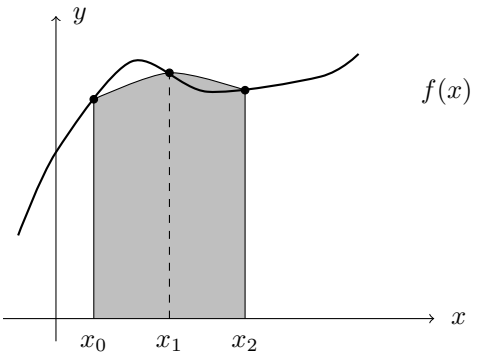


Figura 4.4: regla de Simpson

Para comenzar, dado que $h = x_2 - x_1 = x_1 - x_0$, el polinomio $P(x)$ se puede escribir como

$$P(x) = \frac{f(x_0)}{2h^2}(x-x_1)(x-x_2) - \frac{f(x_1)}{h^2}(x-x_0)(x-x_2) + \frac{f(x_2)}{2h^2}(x-x_0)(x-x_1),$$

de donde, para calcular $\int_{x_0}^{x_2} P(x)\,dx$, es necesario calcular las integrales

$$\int_{x_0}^{x_2} (x-x_1)(x-x_2)\,dx,$$

$$\int_{x_0}^{x_2} (x - x_0)(x - x_2)\, dx,$$

$$\int_{x_0}^{x_2} (x - x_0)(x - x_1)\, dx.$$

Al hacer el cambio de variable $u = x - x_1$, dichas integrales se transforman en

$$\int_{-h}^{h} u(u - h)\, du,$$

$$\int_{-h}^{h} (u + h)(u - h)\, du,$$

$$\int_{-h}^{h} (u + h)u\, du,$$

las cuales tienen valor de $\frac{2h^3}{3}$, $-\frac{4h^3}{3}$ y $\frac{2h^3}{3}$ respectivamente. Se concluye entonces que

$$\int_{x_0}^{x_2} P(x)\, dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right],$$

y por tanto

$$\int_{x_0}^{x_2} f(x)\, dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]. \tag{4.17}$$

Si el intervalo $[a, b]$ se divide en N subintervalos $[x_i, x_{i+1}]$, donde $x_0 = a$ y $x_N = b$ y N es par, el resultado 4.17 se puede aplicar en cada par de subintervalos consecutivos para llegar al siguiente método conocido como *regla de Simpson compuesta*:

$$\int_a^b f(x)\, dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{\frac{N}{2}-1} f(x_{2i}) + 4\sum_{i=1}^{\frac{N}{2}} f(x_{2i-1}) + f(x_N)\right]. \tag{4.18}$$

Sobre el término de error, en el caso de funciones cuatro veces derivables con continuidad en el intervalo $[a, b]$, se tiene que existe $\xi \in (a, b)$ tal que:

$$\int_a^b f(x)\, dx = \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{\frac{N}{2}-1} f(x_{2i}) + 4\sum_{i=1}^{\frac{N}{2}} f(x_{2i-1}) + f(x_N)\right] - \frac{(b-a)}{180}h^4 f^{(4)}(\xi). \tag{4.19}$$

Ejemplo 26. Estimar un valor de N que permita una aproximación al valor exacto de $\int_{-1}^{1} e^{-x^2}\, dx$ con una precisión menor a $\varepsilon = 10^{-6}$ al usar la regla de Simpson compuesta.

Solución: si $f(x) = e^{-x^2}$, entonces $f^{(4)}(x) = 4e^{-x^2}\left(4x^4 - 12x^2 + 3\right)$ y $\left|f^{(4)}(\xi)\right| \leq 12$ con razonamientos similares al ejemplo 25. Dado lo anterior, se tiene la siguiente estimación para el término de error:

$$\left|\frac{(b-a)}{180}h^4 f^{(4)}(\xi)\right| = \frac{1-(-1)}{180}h^4\left|f^{(4)}(\xi)\right| \leq \frac{2h^4}{15}.$$

Si se desea alcanzar la precisión dada, se tiene que $\dfrac{2h^4}{15} < 10^{-6}$, o en forma equivalente $\dfrac{32}{15N^4} < 10^{-6}$ pues $h = \dfrac{2}{N}$. De lo anterior se sigue que $N > \sqrt[4]{\dfrac{32 \cdot 10^6}{15}} \approx 37.905$, y por tanto se requiere de $N = 38$ para obtener la precisión de $\varepsilon = 10^{-6}$. Notar la buena mejora con respecto al ejemplo 25. ♢

Ejercicios 11

1. Aproximar las siguientes integrales usando la regla del trapecio compuesta con $N = 40$.

   a) $\displaystyle\int_0^1 e^{-2x+5}\,dx$ *d)* $\displaystyle\int_{-2}^2 \frac{1}{1+x^2}\,dx$

   b) $\displaystyle\int_2^6 x\ln(x)\,dx$ *e)* $\displaystyle\int_{10}^{15}\left(1+\sqrt{x^2+1}\right)dx$

   c) $\displaystyle\int_0^\pi \cos^2(x)\,dx$ *f)* $\displaystyle\int_0^4 \sin\left(e^x\right)\,dx$

2. Aproximar las siguientes integrales usando la regla de Simpson compuesta con $N = 10$.

   a) $\displaystyle\int_0^1 e^{-2x+5}\,dx$ *d)* $\displaystyle\int_{-2}^2 \frac{1}{1+x^2}\,dx$

   b) $\displaystyle\int_2^6 x\ln(x)\,dx$ *e)* $\displaystyle\int_{10}^{15}\left(1+\sqrt{x^2+1}\right)dx$

   c) $\displaystyle\int_0^\pi \cos^2(x)\,dx$ *f)* $\displaystyle\int_0^4 \sin\left(e^x\right)\,dx$

3. Estimar un valor de N que permita una aproximación al valor exacto de $\displaystyle\int_0^{\frac{\pi}{2}} \cos x\,dx$ con una precisión menor a $\varepsilon = 10^{-8}$ al usar la regla del trapecio compuesta.

4. Estimar un valor de N que permita una aproximación al valor exacto de $\displaystyle\int_1^2 e^x \sin x\,dx$ con una precisión menor a $\varepsilon = 10^{-8}$ al usar la regla de Simpson compuesta.

7
6

Capítulo 5

Sistemas de ecuaciones

5.1. Introducción

En dos de los problemas estudiados en sesiones anteriores se ha necesitado solucionar sistemas de ecuaciones lineales. En el caso de los trazadores cúbicos, es necesario solucionar un sistema tridiagonal con el fin de conocer los c_i y en el cual la dimensión del sistema depende del número de puntos a interpolar. También en el caso de los polinomios de mínimos cuadrados, donde la dimensión del sistema depende del grado del polinomio que se escoja para aproximar la lista de puntos o la función.

En ambos casos, los sistemas de ecuaciones pueden ser de dimensión alta, como es el caso de aplicaciones de computación gráfica, donde el número puede ser del orden de miles de puntos. Por lo tanto, el problema de solucionar sistemas de ecuaciones, desde el punto de vista computacional, es relevante y de extensa aplicación. En general, los métodos que resuelven el anterior problema se pueden clasificar en: *métodos directos* y *métodos iterativos*, estos últimos inspirados en la iteración de punto fijo.

5.2. Métodos directos

Aunque el más famoso de los métodos directos es la eliminación gaussiana, es poco práctico por la cantidad de operaciones que se requieren para reducir a forma escalonada una matriz. Otra clase de métodos muy usados corresponde a factorizaciones, y aunque existen varios tipos de ellas, en este capítulo se aborda la factorización LU, como mecanismo para escribir algoritmos muy eficientes para resolver sistemas tipo banda (como el tridiagonal) o sistemas con muchos ceros.

Sistemas de ecuaciones que involucren matrices triangulares, ya sean inferiores (L, *lower*) o superiores (U, *upper*) son particularmente fáciles de resolver, al igual que sistemas que

involucren matrices diagonales. A continuación se examinan en detalle dichas situaciones, como preámbulo de la factorización LU que servirá para abordar problemas más generales.

Matrices diagonales. Considerar el sistema de ecuaciones:

$$\begin{pmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & d_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

Al resolver fila por fila se obtiene

$$x_1 = \frac{b_1}{d_{11}}, \quad x_2 = \frac{b_2}{d_{22}}, \quad \ldots \quad x_n = \frac{b_n}{d_{nn}}$$

y se puede entonces escribir la solución genérica:

$$x_i \leftarrow \frac{b_i}{d_{ii}} \quad \text{para} \quad i = 1, 2, \ldots, n$$

Nota: observar que el sistema tiene solución única si $d_{ii} \neq 0$ para todo i, lo cual es equivalente a que $\det D \neq 0$, donde D es la matriz diagonal y $\det D = \prod_{i=1}^{n} d_{ii}$.

Matriz triangular inferior L. Para una matriz triangular inferior de orden $n \times n$, definida como $l_{ij} = 0$, *si* $j > i$, considerar el sistema:

$$\begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

Con la primera ecuación del sistema, primera fila, es posible calcular el valor de x_1.

$$l_{11}x_1 = b_1$$

Al despejar se tiene que

$$x_1 = \frac{b_1}{l_{11}}$$

y con dicho valor de x_1 se procede a calcular el valor de x_2 con la segunda fila del sistema

$$l_{21}x_1 + l_{22}x_2 = b_2.$$

Se obtiene el siguiente valor de x_2

$$x_2 = \frac{b_2 - l_{21}x_1}{l_{22}}.$$

Ahora que se conocen los valores de x_1 y x_2, se procede a calcular x_3 con la tercera fila del sistema

$$l_{31}x_1 + l_{32}x_2 + l_{33}x_3 = b_3$$

Como resultado se tiene

$$x_3 = \frac{b_3 - l_{31}x_1 - l_{32}x_2}{l_{33}},$$

lo cual se puede escribir en forma alternativa como

$$x_3 = \frac{b_3 - \sum_{r=1}^{2} l_{3r}x_r}{l_{33}}.$$

Al seguir este razonamiento, se obtiene para x_i

$$x_i = \frac{b_i - \sum_{r=1}^{i-1} l_{ir}x_r}{l_{ii}},$$

donde la anterior expresión es válida para $i = 1, 2, \ldots n$. Observar que para $i = 1$ la suma va desde uno hasta cero, lo que se interpretará como cero, reduciendo el valor de x_1 a b_1/l_{11}.

Un algoritmo basado en la última ecuación se conoce como *sustitución progresiva*, ya que al conocer la primera incógnita se encuentra la segunda, y con la primera y segunda es posible encontrar la tercera y así sucesivamente. De nuevo, notar que el sistema tiene solución única si $l_{ii} \neq 0$ para toda $i = 1, 2, \ldots, n$. Recordar también que $\det L = \prod_{i=1}^{n} l_{ii}$.

Matriz triangular superior U. Recordar que una matriz es triangular superior si $u_{ij} = 0$ para $i > j$. Un sistema con una matriz triangular superior U tiene la forma:

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

En este caso, si se comienza con la primera fila, se obtiene una ecuación con n incógnitas, y por tanto es adecuado empezar por la última fila, en la cual la ecuación solamente contiene a x_n.

$$u_{nn}x_n = b_n$$

Al despejar:

$$x_n = \frac{b_n}{u_{nn}}$$

y ahora, en *forma regresiva*, como es conocido el valor de x_n, se procede a calcular el de x_{n-1}, con la penúltima fila,

$$u_{n-1n-1}x_{n-1} + u_{n-1n}x_n = b_{n-1}$$

donde se obtiene

$$x_{n-1} = \frac{b_{n-1} - u_{n-1n}x_n}{u_{n-1n-1}}.$$

Al subir una fila, y con los valores de x_n y x_{n-1}, se calcula x_{n-2}, con la antepenúltima fila

$$u_{n-2n-2}x_{n-2} + u_{n-2n-1}x_{n-1} + u_{n-2n}x_n = b_{n-2}$$

para obtener

$$x_{n-2} = \frac{b_{n-2} - u_{n-2n-1}x_{n-1} - u_{n-2n}x_n}{u_{n-2n-2}}$$

que se puede escribir también como

$$x_{n-2} = \frac{b_{n-2} - \sum_{r=n-1}^{n} u_{n-2r}x_r}{u_{n-2n-2}}$$

Al seguir el mismo razonamiento, se calcula x_i como:

$$x_i = \frac{b_i - \sum_{r=i+1}^{n} u_{ir}x_r}{u_{ii}}$$

para $i = n, n-1, n-2, \ldots, 1$. Un algoritmo basado en la anterior fórmula se llama *sustitución regresiva*. Notar que cuando $i = n$, la suma va de $n+1$ hasta n, lo cual se asume como cero, y por tanto existe la reducción a $x_n = b_n/u_{nn}$. Recordar además que $\det U = \prod_{i=1}^{n} u_{ii}$.

Ejercicios 12

1. Resolver los siguientes sistemas de ecuaciones utilizando las técnicas descritas anteriormente.

   a) $\begin{pmatrix} 15 & 0 & 0 \\ 0 & -7 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ 9 \end{pmatrix}$

   b) $\begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 2 & -8 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -5 \\ 7 \\ 3 \end{pmatrix}$

   c) $\begin{pmatrix} 5 & 1 & 7 \\ 0 & -9 & 13 \\ 0 & 0 & 11 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ -3 \\ 121 \end{pmatrix}$

$$d)\ \begin{pmatrix} 7 & 0 & 0 & 0 \\ -2 & 5 & 0 & 0 \\ 6 & 9 & -4 & 0 \\ 1 & 3 & 5 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 7 \\ -2 \\ -3 \\ 0 \end{pmatrix}$$

$$e)\ \begin{pmatrix} 1 & -2 & 2 & 2 \\ 0 & -1 & 0 & 3 \\ 0 & 0 & -4 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 3 \\ 4 \end{pmatrix}$$

5.2.1. Factorización LU

Aunque para sistemas $n \times n$ no triangulares el problema es aparentemente complicado de resolver, si la matriz de coeficientes A se puede expresar como el producto de una matriz triangular inferior L con una matriz triangular superior U, es decir $A = LU$, entonces el siguiente par de sistemas brindan solución al sistema general $A\mathbf{x} = \mathbf{b}$.

$$L\mathbf{z} = \mathbf{b}$$

$$U\mathbf{x} = \mathbf{z}$$

Notar que para el primer sistema el algoritmo de sustitución progresiva permite calcular completamente el vector $\mathbf{z}$, mientras que en el segundo, una vez es conocido $\mathbf{z}$, con sustitución regresiva se puede calcular completamente el vector $\mathbf{x}$ y el problema quedaría entonces solucionado. Dado lo anterior, es relevante tratar de factorizar una matriz A en forma LU. A través del siguiente ejemplo se observará un procedimiento para calcular una factorización LU.

Ejemplo 27. Determinar una factorización LU de la matriz

$$\begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}$$

Solución: se desea encontrar una matriz triangular inferior L y una matriz triangular superior U tal que

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}.$$

En dicho caso $l_{11}u_{11} = 1$, y si se define que $l_{11} = 1$, entonces $u_{11} = 1$. Por otro lado $l_{11}u_{12} = -6$ y $l_{11}u_{13} = -3$, y por tanto $u_{12} = -6$ y $u_{13} = -3$ como se tiene a continuación.

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & -6 & -3 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}$$

Ahora, $l_{21}u_{11} = -6$ y $l_{31}u_{11} = -8$, de donde al despejar se tiene $l_{21} = -6$ y $l_{31} = -8$.

$$\begin{pmatrix} 1 & 0 & 0 \\ -6 & l_{22} & 0 \\ -8 & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & -6 & -3 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}$$

Continuando con el elemento u_{22}, se tiene $(-6)\cdot(-6) + l_{22}u_{22} = 37$, y si se define $l_{22} = 1$, se obtiene $u_{22} = 37 - 36 = 1$. También $(-8)\cdot(-6) + l_{32}u_{22} = 40$ y $(-6)\cdot(-3) + l_{22}u_{23} = 23$, de donde se concluye que $l_{32} = -8$ y $u_{23} = 5$.

$$\begin{pmatrix} 1 & 0 & 0 \\ -6 & 1 & 0 \\ -8 & -8 & l_{33} \end{pmatrix} \begin{pmatrix} 1 & -6 & -3 \\ 0 & 1 & 5 \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}$$

Por último, $(-8)\cdot(-3) + (-8)\cdot 5 + l_{33}u_{33} = -15$, y si se define $l_{33} = 1$, se obtiene $u_{33} = -15 + 40 - 24 = 1$.

$$\begin{pmatrix} 1 & 0 & 0 \\ -6 & 1 & 0 \\ -8 & -8 & 1 \end{pmatrix} \begin{pmatrix} 1 & -6 & -3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -6 & -3 \\ -6 & 37 & 23 \\ -8 & 40 & -15 \end{pmatrix}$$

Así,

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -6 & 1 & 0 \\ -8 & -8 & 1 \end{pmatrix} \qquad U = \begin{pmatrix} 1 & -6 & -3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

♢

El anterior procedimiento se puede resumir de la siguiente manera.

Primer paso. Calcular los elementos l_{rr} y u_{rr}. En este paso es necesario definir un valor particular para l_{rr} o u_{rr}. Para ello, dado que

$$a_{rr} = \sum_{k=1}^{r} l_{rk}u_{kr},$$

notar que a excepción del último término de la suma, los demás son conocidos, y al separar el último se tiene

$$a_{rr} = \sum_{k=1}^{r-1} l_{rk}u_{kr} + l_{rr}u_{rr}.$$

En este momento se tienen dos incógnitas, y una posible solución a la anterior situación, es asumir un valor arbitrario (no nulo) de una de ellas. A continuación dos formas clásicas de elegir.

1. Factorización de Doolittle: $l_{rr} = 1$ para $r = 1, 2, \ldots, n$.
2. Factorización de Crout: $u_{rr} = 1$ para $r = 1, 2, \ldots, n$.

Cualquiera de las anteriores da lugar a encontrar la otra incógnita. Por ejemplo, si elige la factorización de Doolittle, entonces

$$u_{rr} = a_{rr} - \sum_{k=1}^{r-1} l_{rk} u_{kr} \quad \text{para } r = 1, 2, \ldots, n,$$

mientras que al trabajar con la factorización de Crout,

$$l_{rr} = a_{rr} - \sum_{k=1}^{r-1} l_{rk} u_{kr} \quad \text{para } r = 1, 2, \ldots, n.$$

Segundo paso. Calcular todos los elementos de la columna r de la matriz L. El siguiente elemento a calcular es l_{r+1r}, después l_{r+2r}, hasta finalizar con l_{nr}. Para lo anterior, observar que

$$a_{ir} = \sum_{k=1}^{r} l_{ik} u_{kr},$$

si se separa el último término, el que contiene la incógnita, se tiene que

$$a_{ir} = \sum_{k=1}^{r-1} l_{ik} u_{kr} + l_{ir} u_{rr},$$

y al despejar se tiene finalmente

$$l_{ir} = \frac{a_{ir} - \sum_{k=1}^{r-1} l_{ik} u_{kr}}{u_{rr}} \quad \text{para } i = r+1, r+2, \ldots, n$$

Tercer paso. Calcular todos los elementos de la fila r de la matriz U. Así,

$$a_{rj} = \sum_{k=1}^{r} l_{rk} u_{kj}$$

donde al separar último término de la suma

$$a_{rj} = \sum_{k=1}^{r-1} l_{rk} u_{kj} + l_{rr} u_{rj}$$

se despeja u_{rj} para obtener finalmente

$$u_{rj} = \frac{a_{rj} - \sum_{k=1}^{r-1} l_{rk} u_{kj}}{l_{rr}} \quad \text{para } j = r+1, r+2, \ldots, n$$

Terminado este ciclo se conocen todos los elementos de L desde la primera columna hasta la columna r, al igual que todos los elementos de U desde la primera fila hasta la fila r. En el siguiente ciclo, se busca la fila y columna $r+1$ de las matrices U y L respectivamente.

Ahora se observará cómo utilizar la factorización LU de una matriz A para solucionar un sistema de ecuaciones.

Ejemplo 28. Sea

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

resolver $A\mathbf{x} = \mathbf{b}$, donde $\mathbf{b} = (-1, -6, -13)^t$.

Solución: se debe recordar que si desea resolver un problema de la forma $LU\mathbf{x} = \mathbf{b}$, donde L es una matriz triangular inferior y U es una matriz triangular superior, entonces es necesario resolver los problemas:

$$L\mathbf{z} = \mathbf{b}$$

$$U\mathbf{x} = \mathbf{z}$$

Luego, se debe solucionar en primer lugar el problema $L\mathbf{z} = \mathbf{b}$, es decir

$$\begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -2 & 4 & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -6 \\ -13 \end{pmatrix}$$

donde al utilizar sustitución progresiva se obtiene:

$$\begin{aligned} z_1 &= -1 \\ z_2 &= -6 - 3(z_1) = -6 - 3(-1) = -3 \\ z_3 &= -13 + 2z_1 - 4z_2 = -13 + 2(-1) - 4(-3) = -3 \end{aligned}$$

Luego $\mathbf{z} = (-1, -3, -3)^t$. Ahora, se debe solucionar el problema $U\mathbf{x} = \mathbf{z}$

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

en el que se emplea sustitución regresiva

$$\begin{aligned} x_3 &= \frac{-3}{3} = -1 \\ x_2 &= \frac{-3 - x_3}{-2} = \frac{-3 + 1}{-2} = 1 \\ x_1 &= \frac{-1 + x_2}{5} = 0 \end{aligned}$$

para obtener $\mathbf{x} = (0, 1, -1)^t$ como la solución al problema $A\mathbf{x} = \mathbf{b}$ inicial. Para terminar, notar que aunque los cálculos utilizados en la sustitución progresiva y regresiva son de un costo computacional relativamente económico, es en el cálculo de la factorización LU donde puede haber costos elevados de tipo computacional similares a métodos tradicionales como eliminación de Gauss-Jordan. ♢

Ejercicios 13

1. Determinar la factorización LU de las siguientes matrices definiendo $u_{ii} = 1$.

   a) $\begin{pmatrix} -1 & -2 & 1 \\ 2 & 7 & -8 \\ 0 & 1 & 3 \end{pmatrix}$ *c)* $\begin{pmatrix} 9 & 0 & 0 \\ 2 & 12 & 0 \\ 3 & 1 & 3 \end{pmatrix}$

   b) $\begin{pmatrix} -2 & 10 & -4 \\ 5 & -24 & 13 \\ 3 & -13 & 12 \end{pmatrix}$ *d)* $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix}$

2. Repetir el ejercicio anterior tomando $l_{ii} = 1$.

3. Solucionar el sistema $A\mathbf{x} = \mathbf{b}$ donde $\mathbf{b} = (1, 0, 1)^t$ y A corresponde a los siguientes productos de matrices.

   a) $\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix} \begin{pmatrix} 4 & 3 & 2 \\ 0 & 2 & 6 \\ 0 & 0 & 3 \end{pmatrix}$ *b)* $\begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -7 & 9 & 1 \end{pmatrix} \begin{pmatrix} 5 & 0 & -3 \\ 0 & 2 & 1 \\ 0 & 0 & -4 \end{pmatrix}$

4. Determinar la factorización LU de las siguientes matrices haciendo $U = L^t$. Este hecho es conocido como *descomposición de Cholesky.*

   a) $\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$ *c)* $\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$

   b) $\begin{pmatrix} 2 & -1 & 0 \\ -1 & 4 & 2 \\ 0 & 2 & 2 \end{pmatrix}$ *d)* $\begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$

5.3. Métodos iterativos

Cuando se tiene un sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ y se desea determinar el valor de $\mathbf{x}$ que lo satisface, existen diferentes técnicas, como lo son Gauss-Jordan, matriz inversa, regla de Cramer, factorización LU, etc., que permiten encontrar de manera *exacta* la solución, quizá con costos computacionales elevados.

Ahora, si para determinados problemas es suficiente obtener un $\widetilde{\mathbf{x}}$ que no satisface de manera exacta el sistema $A\mathbf{x} = \mathbf{b}$, pero el resultado de $A\widetilde{\mathbf{x}}$ se *aproxima* a $\mathbf{b}$ y el costo de calcular $\widetilde{\mathbf{x}}$ es menor al de calcular el valor exacto $\mathbf{x}$, entonces sería preferible construir $\widetilde{\mathbf{x}}$.

Si se observa que la solución del problema $A\mathbf{x} = \mathbf{b}$ corresponde a la solución de la ecuación[1] $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ donde $\mathbf{F}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$, sería natural entonces tratar de utilizar los métodos descritos en el capítulo 2 del libro, pues se estaría buscando un cero de cierta función. Dada la naturaleza vectorial del problema, es necesario primero revisar algunos conceptos propios de la teoría de vectores y matrices.

5.3.1. Normas vectoriales

En el curso de álgebra lineal se define la norma de un vector $\mathbf{u} = (x, y, z)$ como:

$$\|\mathbf{u}\|_2 = \sqrt{x^2 + y^2 + z^2}$$

y se denomina *norma euclidiana*. Una de las aplicaciones de la norma euclidiana es la definición de *distancia euclidiana* entre dos vectores $\mathbf{u}$ y $\mathbf{v}$:

$$d_2(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2$$

Ejemplo 29. Determinar la distancia entre los vectores $\mathbf{u} = (1, -1, 2)$ y $\mathbf{v} = (2, -3, -6)$.

Solución:

$$\begin{aligned} d_2(\mathbf{u}, \mathbf{v}) &= \|\mathbf{u} - \mathbf{v}\|_2 \\ &= \|(1, -1, 2) - (2, -3, -6)\|_2 \\ &= \|(-1, 2, 8)\|_2 \\ &= \sqrt{(-1)^2 + 2^2 + 8^2} \\ &= \sqrt{1 + 4 + 64} = \sqrt{69} \end{aligned}$$

♢

En la siguiente definición se busca generalizar la idea de norma euclidiana y construir nuevas distancias sobre los vectores.

Definición 4. *Una* ***norma vectorial*** *en $\mathbb{R}^n$ es una función $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ tal que:*

1. $\|\mathbf{x}\| \geq 0$ *para todo* $\mathbf{x} \in \mathbb{R}^n$
2. $\|\mathbf{x}\| = 0$ *si y sólo si* $\mathbf{x} = \mathbf{0}$
3. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ *para todo* $\mathbf{x} \in \mathbb{R}^n$ *y* $\alpha \in \mathbb{R}$
4. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *para todo* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *(desigualdad triangular)*

[1] Notar que $\mathbf{0}$ se refiere al vector nulo y $\mathbf{F}$ es una función vectorial $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ para cierto n.

Se puede verificar que la norma euclidiana cumple con las anteriores propiedades. Otras normas que se pueden definir en $\mathbb{R}^n$ son l_1 y l_∞, que presentan ventajas computacionales para procedimientos iterativos.

Definición 5. *Sea* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *un vector en* $\mathbb{R}^n$*, entonces:*

- *La norma* l_1 *del vector* $\mathbf{x}$ *es:* $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$
- *La norma* l_∞ *del vector* $\mathbf{x}$ *es:* $\|\mathbf{x}\|_\infty = \max\limits_{1\leq i\leq n} |x_i|$

Ejemplo 30. Sea $\mathbf{x} = (-1, 2, -5, 3)$ entonces:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| = |-1| + |2| + |-5| + |3| = 1 + 2 + 5 + 3 = 11$$

$$\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n} |x_i| = \max\{|-1|, |2|, |-5|, |3|\} = 5$$

A partir de las normas l_1 y l_∞ se construyen las distancias $d_1(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_1$ y $d_\infty(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_\infty$.

Dado que en este capítulo se desea desarrollar métodos *iterativos* para aproximar la solución del sistema $A\mathbf{x} = \mathbf{b}$, en los cuales se parte de un vector inicial (semilla) $\mathbf{x}^{(0)}$ y se construyen vectores $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}$ los cuales se aproximan al vector $\mathbf{x}$ solución de la ecuación $A\mathbf{x} = \mathbf{b}$, es necesario definir cuando una sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ de vectores en $\mathbb{R}^n$ *convergen* a un vector $\mathbf{x}$.

Definición 6. *Sea* $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ *una sucesión de vectores de* $\mathbb{R}^n$ *y* $\mathbf{x}$ *un vector de* $\mathbb{R}^n$*. Entonces la sucesión converge a* $\mathbf{x}$ *respecto a la norma* $\|\cdot\|$ *si dado* $\varepsilon > 0$*, existe un entero positivo* N *tal que*

$$\left\|\mathbf{x}^{(k)} - \mathbf{x}\right\| < \varepsilon \qquad \text{para todo } k \geq N$$

Desde un aspecto computacional, la definición anterior no es la mejor, por lo tanto se presenta el siguiente teorema.

Teorema 7. *La sucesión* $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ *converge a* $\mathbf{x}$ *respecto a la norma* l_∞ *si y solo si* $\lim\limits_{k\to\infty} x_i^{(k)} = x_i$ *para cada* $i = 1, 2, \ldots, n$.

Es de anotar, que el teorema 7 indica una manera de comprobar si una sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ converge a un vector $\mathbf{x}$ con relación a la norma l_∞ y no con relación a cualquier norma.

Ejemplo 31. Demostrar que la sucesión $\mathbf{x}^{(k)} = \left(e^{-k}, 5 + \dfrac{1}{k}, \dfrac{-3\sin(k)}{k}\right)$ converge a el vector $(0, 5, 0)$ con relación a la norma l_∞.

Solución: utilizando el teorema 7 es suficiente demostrar que

- $\lim_{k\to\infty} e^{-k} = 0$
- $\lim_{k\to\infty} 5 + \frac{1}{k} = 5$
- $\lim_{k\to\infty} \frac{-3\sin(k)}{k} = 0$

En este caso,

- $\lim_{k\to\infty} e^{-k} = \lim_{k\to\infty} \frac{1}{e^k} = 0,$
- $\lim_{k\to\infty} 5 + \frac{1}{k} = \lim_{k\to\infty} 5 + \lim_{k\to\infty} \frac{1}{k} = 5 + 0 = 5$ y
- $\lim_{k\to\infty} \frac{-3\sin(k)}{k} = -3 \lim_{k\to\infty} \frac{\sin(k)}{k} = -3(0) = 0,$

con lo cual se demuestra que la sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=1}^{\infty}$ converge a el vector $(0, 5, 0)$ con relación a la norma l_∞. ♢

5.3.2. Normas matriciales

En la construcción y estudio de los métodos iterativos para aproximar una solución del sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ es necesario definir la distancia entre matrices, donde una de las maneras más efectivas es definir una norma sobre espacios de matrices.

Definición 7. *Una* norma matricial *es una función* $\|\cdot\| : \mathcal{M}_n(\mathbb{R}) \to \mathbb{R}$ *del conjunto de matrices de tamaño* $n \times n$ *en los números reales tal que*

1. $\|A\| \geq 0$ *para toda* $A \in \mathcal{M}_n(\mathbb{R})$
2. $\|A\| = 0$ *si y solo si* $A = 0$
3. $\|\alpha A\| = |\alpha|\|A\|$ *para toda* $A \in \mathcal{M}_n(\mathbb{R})$ *y* $\alpha \in \mathbb{R}$
4. $\|A + B\| \leq \|A\| + \|B\|$ *para toda* $A, B \in \mathcal{M}_n(\mathbb{R})$
5. $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ *para toda* $A, B \in \mathcal{M}_n(\mathbb{R})$

la distancia con relación a la norma matricial $\|\cdot\|$ entre las matrices $A_{n\times n}$ y $B_{n\times n}$ se define como $\|A - B\|$.

Es posible demostrar que toda norma vectorial sobre $\mathbb{R}^n$ induce una norma matricial sobre $\mathcal{M}_n(\mathbb{R})$. En la siguiente definición se presenta las normas matriciales inducidas por las normas vectoriales l_1 y l_∞.

Definición 8. *Si $A = [a_{ij}]_{n\times n}$ es una matriz, entonces:*

- $\|A\|_\infty = \max\limits_{1\le i\le n}\left\{\sum\limits_{j=1}^{n}|a_{ij}|\right\}$
- $\|A\|_1 = \max\limits_{1\le j\le n}\left\{\sum\limits_{i=1}^{n}|a_{ij}|\right\}$

Ejemplo 32. Si

$$A = \begin{pmatrix} -5 & 3 & 1 \\ 3 & 7 & 8 \\ 3 & 8 & 1 \end{pmatrix}$$

calcular $\|A\|_\infty$.

Solución: dado que $\|A\|_\infty = \max\limits_{1\le i\le 3}\left\{\sum\limits_{j=1}^{3}|a_{ij}|\right\}$, entonces

$$\|A\|_\infty = \max\left\{\sum_{j=1}^{n}|a_{1j}|, \sum_{j=1}^{3}|a_{2j}|, \sum_{j=1}^{3}|a_{3j}|\right\}$$

De otro lado, $\sum\limits_{j=1}^{3}|a_{1j}| = |-5|+|3|+|1| = 9$, $\sum\limits_{j=1}^{3}|a_{2j}| = |3|+|7|+|8| = 18$ y $\sum\limits_{j=1}^{3}|a_{3j}| = |3|+|8|+|1| = 12$, y por tanto $\|A\|_\infty = \max\{9, 18, 12\} = 18$. ♢

Ejemplo 33. Si

$$A = \begin{pmatrix} -5 & 3 & 1 \\ 3 & 7 & 8 \\ 3 & -3 & 6 \end{pmatrix}$$

calcular $\|A\|_1$.

Solución: dado que $\|A\|_1 = \max\limits_{1\le j\le 3}\left\{\sum\limits_{i=1}^{3}|a_{ij}|\right\}$, entonces

$$\|A\|_1 = \max\left\{\sum_{i=1}^{3}|a_{i1}|, \sum_{i=1}^{3}|a_{i2}|, \sum_{i=1}^{3}|a_{i3}|\right\}$$

pero $\sum\limits_{i=1}^{3}|a_{i1}| = |-5|+|3|+|3| = 11$, $\sum\limits_{i=1}^{3}|a_{i2}| = |3|+|7|+|-3| = 13$ y $\sum\limits_{i=1}^{3}|a_{i3}| = |1|+|8|+|6| = 15$, de donde $\|A\|_1 = \max\{11, 13, 15\} = 15$. ♢

Ejercicios 14

1. Determinar la norma euclidiana, la norma l_∞ y la norma l_1 para los vectores **u**, **v**, **w** y **z** que se encuentran a continuación.

a) $\mathbf{u} = (-1, 2, -1)$

b) $\mathbf{v} = (-5, 3, 1)$

c) $\mathbf{w} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, \frac{1}{128}\right)$

d) $\mathbf{z} = \left(e^{-1}, e^{2}, e^{-3}, \ldots, e^{-7}\right)$

2. Demostrar que las sucesiones $\left\{\mathbf{x}^{(k)}\right\}_{k=1}^{\infty}$ convergen a el vector $(-1, 0, e, 1)$ con relación a la norma l_∞.

a) $\mathbf{x}^{(k)} = \left(-1, \frac{1}{k}, e, \frac{k}{k+1}\right)$

b) $\mathbf{x}^{(k)} = \left(-1 + e^{-k}, \frac{-1}{k^2}, e, 1 + \frac{k}{x^k + 1}\right)$

c) $\mathbf{x}^{(k)} = \left(-1 + \frac{k}{k^4 - 1}, 0, e, 1\right)$

d) $\mathbf{x}^{(k)} = \left(-1, \frac{\cos(k)}{k}, \left(1 + \frac{1}{k}\right)^k, 1\right)$

3. Determinar la norma matricial $\|\cdot\|_\infty$ y la norma matricial $\|\cdot\|_1$ para las siguientes matrices.

a) $\begin{pmatrix} 1 & 0 & 0 \\ -3 & 7 & 0 \\ 3 & 5 & 1 \end{pmatrix}$

b) $\begin{pmatrix} -5 & 3 & 8 \\ -3 & 7 & 4 \\ 0 & 0 & 1 \end{pmatrix}$

c) $\begin{pmatrix} 15 & 13 & 8 \\ 3 & 27 & -4 \\ -2 & 10 & 11 \end{pmatrix}$

d) $\begin{pmatrix} 2 & -1 & 4 & 3 \\ -1 & 8 & 5 & -6 \\ 4 & 5 & 3 & -8 \\ -11 & 15 & 13 & 1 \end{pmatrix}$

4. Dada una matriz cuadrada A de $n \times n$, la norma de Frobenius se define como:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}$$

calcular la norma de Frobenius para la matriz $A = \begin{pmatrix} 2 & 0 & -1 \\ -1 & 3 & 1 \\ 2 & 0 & 3 \end{pmatrix}$.

5.3.3. Solución de sistemas de ecuaciones

Como se menciona con anterioridad, los métodos iterativos en la solución de sistemas de ecuaciones $A\mathbf{x} = \mathbf{b}$ construyen una sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ convergente a la solución del sistema. Es de anotar que los métodos que se presentan en esta sección son de un esquema recursivo, es decir $\mathbf{x}^{(k+1)} = T\mathbf{x}^{(k)} + \mathbf{c}$, donde T es una matriz fija y $\mathbf{c}$ un vector. En lo que sigue se presentan dos métodos iterativos clásicos, el método de Jacobi y el método de Gauss-Seidel.

5.3.4. Método de Jacobi

El método de Jacobi se basa en el método del punto fijo, y para entenderlo se analiza el siguiente ejemplo.

Ejemplo 34. Encontrar la solución del siguiente sistema de ecuaciones.

$$\begin{aligned} 5x_1 - x_2 + x_3 + x_4 &= 5 \\ x_1 + 4x_2 + x_4 &= 2 \\ 3x_1 - x_2 + 6x_3 &= -3 \\ x_2 - x_3 + 3x_4 &= 4 \end{aligned}$$

Solución: si de la primera ecuación se despeja x_1, de la segunda ecuación se despeja x_2, y así sucesivamente, se obtiene:

$$\begin{aligned} x_1 &= \frac{5 + x_2 - x_3 - x_4}{5} \\ x_2 &= \frac{2 - x_1 - x_4}{4} \\ x_3 &= \frac{-3 - 3x_1 + x_2}{6} \\ x_4 &= \frac{4 - x_2 + x_3}{3} \end{aligned} \tag{5.1}$$

Tomando ahora $\mathbf{x}^{(0)} = (-1, 5, 3, 2)$ y sustituyendo en el lado derecho del sistema 5.1 se obtiene

$$\begin{aligned} x_1^{(1)} &= \frac{5 + 5 - 3 - 2}{5} = 1 \\ x_2^{(1)} &= \frac{2 + 1 - 2}{4} = \frac{1}{4} \\ x_3^{(1)} &= \frac{-3 - 3(-1) + 5}{6} = \frac{5}{6} \\ x_4^{(1)} &= \frac{4 - 5 + 3}{3} = \frac{2}{3} \end{aligned}$$

luego $\mathbf{x}^{(1)} = \left(1, \frac{1}{4}, \frac{5}{6}, \frac{2}{3}\right)$, y repitiendo el proceso ahora con $\mathbf{x}^{(1)}$ se desprende que:

$$x_1^{(2)} = \frac{5 + \frac{1}{4} - \frac{5}{6} - \frac{2}{3}}{5} = \frac{3}{4}$$
$$x_2^{(2)} = \frac{2 - 1 - \frac{2}{3}}{4} = \frac{1}{12}$$
$$x_3^{(2)} = \frac{-3 - 3 + \frac{1}{4}}{6} = -\frac{23}{24}$$
$$x_4^{(2)} = \frac{4 - \frac{1}{4} + \frac{5}{6}}{3} = \frac{55}{36}$$

de donde $\mathbf{x}^{(2)} = \left(\frac{3}{4}, \frac{1}{12}, \frac{23}{24}, \frac{55}{36}\right)$. En el cuadro 5.1 se encuentran las primeras diez iteraciones del método.

k	0	1	2	3	4	5	6	7	8	9	10
$x_1^{(k)}$	−1	1	0,75	0,9027	0,9611	0,9842	0,9930	0,9971	0,9987	0,9995	0,9997
$x_2^{(k)}$	5	0,25	0,0833	−0,0694	0,0277	−0,0076	0,0031	−0,0009	0,0004	−0,0001	0,0000
$x_3^{(k)}$	3	0,8333	−0,9583	−0,8611	−0,9629	−0,9759	−0,9934	−0,9959	−0,9987	−0,9993	−0,9997
$x_4^{(k)}$	2	0,6666	1,5277	0,9861	1,0694	1,0030	1,0105	1,0011	1,0016	1,0002	1,0002

Cuadro 5.1: iteraciones del método de Jacobi.

Se puede observar que $\|\mathbf{x}^{(9)} - \mathbf{x}^{(10)}\|_\infty = \text{máx}\{|0,9995 - 0,9997|, |-0,0001 + 0,0000|, |-0,9993 + 0,9997|, |1,0002 - 1,0002|\} = 0,0004$ y en cada iteración esta distancia disminuye. Si se resuelve el sistema con el método Gauss-Jordan, se obtiene que la solución es $\mathbf{x} = (1, 0, -1, 1)$ y por lo tanto se tiene que la sucesión generada por el método de Jacobi es convergente a la solución del sistema. ♢

El método de Jacobi para solucionar un sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ se puede resumir, de manera muy simple, en los siguientes pasos:

1. Despejar de cada una de las ecuaciones una de las variables (no despejar la misma variable en dos ecuaciones distintas).
2. Seleccionar un vector $\mathbf{x}^{(0)}$ inicial (semilla).
3. Sustituir el vector seleccionado en las ecuaciones del primer paso y llamar a este resultado $\mathbf{x}^{(1)}$.
4. Repetir el tercer paso con $\mathbf{x}^{(1)}$.

Aunque en la descripción anterior no se hace referencia a la forma de detener el proceso, para lo anterior se debe establecer un valor $\varepsilon > 0$ (tolerancia) y considerar como criterio

de parada $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_\infty < \varepsilon$, de donde el proceso termina cuando la distancia entre los vectores $\mathbf{x}^{(k)}$ y $\mathbf{x}^{(k-1)}$ es menor a ε.

Ejemplo 35. Utilizar el método de Jacobi obtener una aproximación de la solución del sistema

$$\begin{aligned} 4x_1 + 2x_2 - x_3 &= -3 \\ -2x_1 + 5x_2 + 2x_3 &= -9 \\ -6x_1 + 3x_2 + 7x_3 &= 10 \end{aligned}$$

con una tolerancia de $\varepsilon = 10^{-4}$.

Solución: a continuación se tiene los pasos necesarios.

Paso 1. Despejar una de las variables de cada ecuación.

$$\begin{aligned} x_1 &= \frac{-3 - 2x_2 + x_3}{4} \\ x_2 &= \frac{-9 + 2x_1 - 2x_3}{5} \\ x_3 &= \frac{10 + 6x_1 - 3x_2}{7} \end{aligned}$$

Paso 2. Seleccionar $\mathbf{x}^{(0)}$. En este caso $\mathbf{x}^{(0)} = (1, -2, 3)$.

Paso 3. Sustituir $\mathbf{x}^{(0)}$ en las ecuaciones del primer paso.

$$\begin{aligned} x_1^{(1)} &= \frac{-3 - 2x_2^{(0)} + x_3^{(0)}}{4} = \frac{-3 - 2(-2) + 3}{4} = 1 \\ x_2^{(1)} &= \frac{-9 + 2x_1^{(0)} - 2x_3^{(0)}}{5} = \frac{-9 + 2(1) - 2(3)}{5} = -\frac{13}{5} \\ x_3^{(1)} &= \frac{10 + 6x_1^{(0)} - 3x_2^{(0)}}{7} = \frac{10 + 6(1) - 3(-2)}{7} = \frac{22}{7} \end{aligned}$$

luego $\mathbf{x}^{(1)} = \left(1, \frac{13}{5}, \frac{22}{7}\right)$ y $\|\mathbf{x}^{(0)} - \mathbf{x}^{(1)}\|_\infty = 0.6$. Dado que la distancia no es menor a 10^{-4}, continuar el proceso.

Paso 4. Repetir el tercer paso, ahora con $\mathbf{x}^{(1)}$.

$$\begin{aligned} x_1^{(2)} &= \frac{-3 - 2x_2^{(1)} + x_3^{(1)}}{4} = \frac{-3 - 2\frac{13}{5} + \frac{22}{7}}{4} = 1.3357 \\ x_2^{(2)} &= \frac{-9 + 2x_1^{(1)} - 2x_3^{(1)}}{5} = \frac{-9 + 2(1) - 2\frac{22}{7}}{5} = -2.6571 \\ x_3^{(2)} &= \frac{10 + 6x_1^{(1)} - 3x_2^{(1)}}{7} = \frac{10 + 6(1) - 3\frac{13}{5}}{7} = 3.4 \end{aligned}$$

luego $\mathbf{x}^{(2)} = (1.3357, -2.6571, 3.4)$ y $\|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\|_\infty = 0.3357$. Dado que la distancia no es menor a 10^{-4}, continuar el proceso. En el cuadro 5.2 se encuentran las primeras diecinueve iteraciones del método.

k	0	1	2	3	4	$\cdots$	16	17	18	19
$x_1^{(k)}$	1	1	1.3357	1.4285	1.4909	$\cdots$	1.6247	1.6248	1.6248	1.6249
$x_2^{(k)}$	-2	-2.6	-2.657	-2.6257	-2.7134	$\cdots$	-2.7498	-2.7499	-2.7499	-2.7499
$x_3^{(k)}$	3	3.1428	3.4	3.7122	3.7783	$\cdots$	3.9995	3.9997	3.9998	3.9998
$d_\infty\left(x^{(k-1)}, x^{(k)}\right)$		0.6	0.3357	0.3122	0.0877	$\cdots$	0.0002	0.0001	0.0001	6.8**E**-5

Cuadro 5.2: iteraciones del método de Jacobi.

Dado que $d_\infty\left(\mathbf{x}^{(18)}, \mathbf{x}^{(19)}\right) = 6.8 \times 10^{-5} < 10^{-4}$, entonces el proceso se detiene y la aproximación de la solución es $\mathbf{x}^{(19)} = (1.6249, -2.7499, 3.9998)$. Se puede verificar que la solución exacta es $x = (1.625, -2.75, 4)$. ♢

5.3.5. Método de Gauss-Seidel

Al examinar con detalle el método de Jacobi, se observa que al momento de calcular $x_i^{(k)}$, con $i > 1$, se utilizan todas las componentes del vector $\mathbf{x}^{(k-1)}$ y para ese instante, se han calculado las componentes $x_j^{(k)}$, $j = 1, 2 \ldots, i-1$, las cuales suponen una mejor aproximación de las componentes x_j del vector solución del sistema $A\mathbf{x} = \mathbf{b}$. Por lo tanto, una idea para acelerar el proceso, es calcular $x_i^{(k)}$ utilizando $x_j^{(k)}$, con $j = 1, 2 \ldots, i-1$, y $x_r^{(k-1)}$ con $r = i+1, i+2, \ldots, n$. El método descrito anteriormente se denomina *método de Gauss-Seidel* y se ilustra en el siguiente ejemplo.

Ejemplo 36. Utilizar el método de Gauss-Seidel resolver el siguiente sistema de ecuaciones.

$$\begin{aligned} 5x_1 - x_2 + 3x_3 &= 1 \\ -x_1 - 7x_2 + x_4 &= -9 \\ 2x_2 + 8x_3 - 2x_4 &= -4 \\ 4x_1 + x_3 - 6x_4 &= 9 \end{aligned}$$

Solución: para aplicar el método de Gauss-Seidel son necesarios los siguientes pasos.

Paso 1. Despejar una variable de cada ecuación.

$$x_1 = \frac{1 + x_2 - 3x_3}{5}$$
$$x_2 = \frac{-9 + x_1 - x_4}{-7}$$
$$x_3 = \frac{-4 - 2x_2 + 2x_4}{8}$$
$$x_4 = \frac{9 - 4x_1 - x_3}{-6}$$

Paso 2. Seleccionar $\mathbf{x}^{(0)}$. En este caso $\mathbf{x}^{(0)} = (0, 0, 0, 0)$.

Paso 3. Calcular $\mathbf{x}^{(1)}$.

$$x_1^{(1)} = \frac{1 + x_2^{(0)} - 3x_3^{(0)}}{5} = \frac{1 + 0 - 3(0)}{5} = \frac{1}{5}$$
$$x_2^{(1)} = \frac{-9 + x_1^{(1)} - x_4^{(0)}}{-7} = \frac{-9 + \frac{1}{5} - 0}{-7} = \frac{44}{35}$$
$$x_3^{(1)} = \frac{-4 - 2x_2^{(1)} + 2x_4^{(0)}}{8} = \frac{-4 - 2\frac{44}{35} + 2(0)}{8} = -\frac{57}{70}$$
$$x_4^{(1)} = \frac{9 - 4x_1^{(1)} - x_3^{(1)}}{-6} = \frac{-9 - 4\frac{1}{5} - -\frac{57}{70}}{-6} = -\frac{631}{420}$$

Ahora, se debe observar lo siguiente:

- Para calcular $x_2^{(1)}$ se ha utilizado la componente $x_1^{(1)} = \frac{1}{5}$, calculada anteriormente y *no* la componente $x_1^{(0)} = 0$.
- Para calcular $x_3^{(1)}$ se ha utilizado la componente $x_2^{(1)} = \frac{44}{55}$, calculada anteriormente y *no* la componente $x_2^{(0)} = 0$.
- Para calcular $x_3^{(1)}$ se han utilizado las componentes $x_1^{(1)} = \frac{1}{5}$ y $x_3^{(1)} = -\frac{55}{70}$, calculadas anteriormente y *no* las componentes $x_1^{(0)} = 0$ y $x_3^{(0)} = 0$.

De donde, para calcular $\mathbf{x}^{(1)}$, siempre se ha utilizado la información más reciente.

Paso 4. Calcular $\mathbf{x}^{(2)}$.

Dado que $\mathbf{x}^{(1)} = \left(\frac{1}{5}, \frac{44}{35}, -\frac{57}{70}, -\frac{631}{420}\right) = (0.2, 1.2571, -0.8142, -1.5023)$, entonces:

$$x_1^{(2)} = \frac{1 + x_2^{(1)} - 3x_3^{(2)}}{5} = \frac{1 + 0.2 - 3(1.2571)}{5} = 0.94$$
$$x_2^{(2)} = \frac{-9 + x_1^{(2)} - x_4^{(1)}}{-7} = \frac{-9 + 0.94 - (-1.5023)}{-7} = 0.9368$$
$$x_3^{(2)} = \frac{-4 - 2x_2^{(2)} + 2x_4^{(1)}}{8} = \frac{-4 - 2(0.9368) + 2(-1.5023)}{8} = -1.1097$$
$$x_4^{(2)} = \frac{9 - 4x_1^{(2)} - x_3^{(2)}}{-6} = \frac{-9 - 4(0.94) - (-1.1097)}{-6} = -1.0582$$

por lo tanto $\mathbf{x}^{(2)} = (0.94, 0.9368, -1.1097, -1.0582)$

En el cuadro 5.3 se encuentran las primeras seis iteraciones del método. Se puede verificar que la solución exacta corresponde a $\mathbf{x} = (1, 1, -1, -1)$ ♢

k	0	1	2	3	4	5	6
$x_1^{(k)}$	0	0.2	0.94	1.0532	1.0031	0.9964	0.9998
$x_2^{(k)}$	0	1.2571	0.9368	0.9840	1.0043	1.0009	0.9997
$x_3^{(k)}$	0	−0.8142	−1.1097	−1.0105	−0.9926	−0.9994	−1.0004
$x_4^{(k)}$	0	−1.5023	−1.058	−0.9662	−0.9966	−1.0022	−1.0001

Cuadro 5.3: iteraciones del método de Gasuss-Seidel.

Se debe resaltar que la diferencia entre el método de Jacobi y Gasuss-Seidel radica en la información que utilizan para calcular $\mathbf{x}^{(k)}$. El método de Jacobi emplea únicamente la información contenida en el vector $\mathbf{x}^{(k-1)}$, mientras que el método de Gauss-Seidel utiliza la información más reciente al usar las componentes de $\mathbf{x}^{(k)}$ que ya calculadas. Al igual que el método de Jacobi, el método de Gauss-Seidel se establece como criterio de parada $\|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k)}\|_\infty < \varepsilon$, para $\varepsilon > 0$ fijo durante la ejecución del método.

Ejemplo 37. Utilizar el método de Gauss-Seidel obtener una aproximación de la solución del siguiente sistema.

$$\begin{aligned} 4x_1 + 2x_2 - x_3 &= -3 \\ -2x_1 + 5x_2 + 2x_3 &= -9 \\ -6x_1 + 3x_2 + 7x_3 &= 10 \end{aligned}$$

con una tolerancia de $\varepsilon = 10^{-4}$.

Solución: para aplicar el método de Gauss-Seidel son necesarios los siguientes pasos.

Paso 1. Despejar una de las variables de cada ecuación.

$$x_1 = \frac{-3 - 2x_2 + x_3}{4}$$
$$x_2 = \frac{-9 + 2x_1 - 2x_3}{5}$$
$$x_3 = \frac{10 + 6x_1 - 3x_2}{7}$$

Paso 2. Seleccionar $\mathbf{x}^{(0)}$. En este caso $\mathbf{x}^{(0)} = (1, -2, 3)$.

Paso 3. Sustituir $x^{(0)}$ en las ecuaciones del primer paso.

$$x_1^{(1)} = \frac{-3 - 2x_2^{(0)} + x_3^{(0)}}{4} = \frac{-3 - 2(-2) + 3}{4} = 1$$
$$x_2^{(1)} = \frac{-9 + 2x_1^{(1)} - 2x_3^{(0)}}{5} = \frac{-9 + 2(1) - 2(3)}{5} = -2.6$$
$$x_3^{(1)} = \frac{10 + 6x_1^{(1)} - 3x_2^{(1)}}{7} = \frac{10 + 6(1) - 3(-2.6)}{7} = 3.4$$

luego $\mathbf{x}^{(1)} = (1, -2.6, 3.4)$ y $\|\mathbf{x}^{(0)} - \mathbf{x}^{(1)}\|_\infty = 0.6$. Dado que la distancia no es menor a 10^{-4}, entonces sigue el proceso.

Paso 4. Repetir el tercer paso, ahora con $\mathbf{x}^{(1)}$.

$$x_1^{(2)} = \frac{-3 - 2x_2^{(1)} + x_3^{(1)}}{4} = \frac{-3 - 2(1) + 3.4}{4} = 1.4$$
$$x_2^{(2)} = \frac{-9 + 2x_1^{(2)} - 2x_3^{(1)}}{5} = \frac{-9 + 2(1.4) - 2(3.4)}{5} = -2.6$$
$$x_3^{(2)} = \frac{10 + 6x_1^{(2)} - 3x_2^{(2)}}{7} = \frac{10 + 6(1.4) - 3(-2.6)}{7} = 3.7428$$

luego $\mathbf{x}^{(2)} = (1.4, -2.6, 3.7428)$ y $\|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\|_\infty = 0.4$. Por lo tanto, la distancia entre $\mathbf{x}^{(1)}$ y $\mathbf{x}^{(2)}$ no es menor a 10^{-4}, entonces sigue el proceso. En el cuadro 5.4 se encuentran las primeras catorce iteraciones del método.

k	0	1	2	3	4	$\cdots$	11	12	13	14
$x_1^{(k)}$	1	1	1.4	1.4857	1.5665	$\cdots$	1.6245	1.6247	1.6248	1.6249
$x_2^{(k)}$	−2	−2.6	−2.6	−2.7028	−2.7175	$\cdots$	−2.7498	−2.7499	−2.7499	−2.7499
$x_3^{(k)}$	3	3.4	3.7428	3.8604	3.9359	$\cdots$	3.9995	3.9997	3.9998	3.9999
$d_\infty\left(x^{(k-1)}, x^{(k)}\right)$		0.6	0.4	0.1175	0.0808	$\cdots$	0.0004	0.0002	0.0001	5.63**E**−05

Cuadro 5.4: iteraciones del método de Gauss-Seidel.

Como $d_\infty\left(\mathbf{x}^{(13)}, \mathbf{x}^{(14)}\right) = 5.63 \times 10^{-5} < 10^{-4}$, entonces el proceso se detiene y la aproximación de la solución es $\mathbf{x}^{(14)} = (1.6249, -2.7499, 3.9999)$. ♢

Nota: si el anterior ejemplo se desarrolla con el método de Jacobi, son necesarias diecinueve iteraciones. Se observa entonces como el método de Gauss-Seidel presenta una convergencia más rápida.

Ejercicios 15

1. Obtener $\mathbf{x}^{(3)}$ en el método de Jacobi en la solución de los siguientes sistemas lineales. Usar $\mathbf{x}^{(0)} = (0, 0, 0)$.

   a)
   $$\begin{aligned} 6x_1 + 3x_2 - x_3 &= 8 \\ 2x_1 + 4x_2 + x_3 &= 7 \\ x_1 - 3x_2 - 5x_3 &= -7 \end{aligned}$$

   b)
   $$\begin{aligned} -4x_1 + x_2 + 2x_3 &= -3 \\ x_1 + 5x_2 + 3x_3 &= -1 \\ x_1 + x_2 - 3x_3 &= -3 \end{aligned}$$

   c)
   $$\begin{aligned} 3x_1 + 2x_3 &= 6 \\ x_1 + 3x_2 &= 3 \\ x_1 + 5x_2 + 7x_3 &= 19 \end{aligned}$$

   d)
   $$\begin{aligned} 4x_1 + x_2 + x_3 - x_4 &= 3 \\ x_1 + 5x_2 - x_3 + x_4 &= 2 \\ -2x_1 - x_2 + 7x_3 + 2x_4 &= -9 \\ x_1 - x_2 - 2x_3 + 6x_4 &= 3 \end{aligned}$$

2. Repetir el ejercicio anterior utilizando el método de Gauss-Seidel.
3. Si se define una tolerancia de $\varepsilon = 10^{-5}$, resolver los ejercicios del primer numeral utilizando el método de Jacobi.
4. Repetir el ejercicio anterior utilizando el método de Gauss-Seidel.

5.3.6. Convergencia métodos de Jacobi y Gauss-Seidel

En esta parte se analizan condiciones para asegurar la convergencia de los métodos iterativos de la forma $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$. Antes de presentar estas condiciones, es necesario revisar algunos conceptos.

Definición 9. *Sea A una matriz $n \times n$. $\rho(A)$, denominado el radio espectral, corresponde a:*

$$\rho(A) = \text{máx}\{|\lambda| \mid \lambda \text{ es valor propio de } A\}$$

Definición 10. *Sea A una matriz $n \times n$. A es de diagonal estrictamente dominante si*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

para $i = 1, 2, \ldots, n$.

Ejemplo 38. Dadas las matrices

$$A = \begin{pmatrix} -6 & 2 & 3 \\ -3 & 7 & 1 \\ 4 & 5 & 10 \end{pmatrix} \quad \text{y} \quad B = \begin{pmatrix} -2 & 1 & 0 \\ 5 & 7 & -3 \\ 9 & -1 & 10 \end{pmatrix},$$

determinar si son de diagonal estrictamente dominante.

Solución: A es de diagonal estrictamente dominante dado que $|-6| > |2| + |3|$, $|7| > |-3| + |1|$ y $|10| > |4| + |5|$. De otro lado, B no es de diagonal estrictamente dominante, ya que en la segunda fila el valor absoluto del elemento diagonal $|7|$ no es mayor a la suma de los valores absolutos de los elementos restantes de la fila, a saber, $|5| + |-3| = 8$. ♢

Con los anteriores elementos es posible presentar los siguientes resultados.

Teorema 8. *Dado cualquier* $\mathbf{x}^{(0)}$ *vector de* $\mathbb{R}^n$*, la sucesión* $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ *generada por el esquema recursivo*

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$$

converge a la única solución del problema $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ *si y solo si* $\rho(A) < 1$.

El teorema 8 brinda condiciones necesarias y suficientes para garantizar la convergencia de un método iterativo de la forma $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$ con T una matriz fija y $\mathbf{c}$ un vector. Antes de dar condiciones *suficientes* para garantizar la convergencia de los métodos de Jacobi y Gauss-Seidel, es necesario presentar estos métodos como esquemas recursivos de la forma $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$.

Si $A\mathbf{x} = \mathbf{b}$ es un sistema de ecuaciones lineales con la siguiente matriz de coeficientes A

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

y si se divide A de la siguiente manera

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} - \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ -a_{21} & 0 & \cdots & 0 & 0 \\ -a_{31} & -a_{32} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & -a_{nn-1} & 0 \end{pmatrix} - \begin{pmatrix} 0 & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ 0 & 0 & -a_{23} & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & -a_{n-1n} \\ 0 & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

entonces $A = D - L - U$ donde D es matriz diagonal, U es una matriz triangular superior y L es una matriz triangular inferior. En este caso, el sistema $A\mathbf{x} = \mathbf{b}$ se transforma en el sistema $(D - L - U)\mathbf{x} = \mathbf{b}$ lo que implica que

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b}$$

Ahora, si D^{-1} existe[2] entonces multiplicando ambos lados de la anterior ecuación se obtiene

$$\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b},$$

que a su vez, se transforma en el esquema iterativo

$$\mathbf{x}^{(k)} = D^{-1}(L + U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}.$$

[2] D^{-1} para una matriz diagonal D existe si y sólo si $a_{ii} \neq 0$ para $i = 1, 2, \ldots, n$.

Si se define $D^{-1}(L+U) = J$ y $\mathbf{c} = D^{-1}\mathbf{b}$, entonces el anterior esquema corresponde a la forma matricial del método de Jacobi:

$$\mathbf{x}^{(k)} = J\mathbf{x}^{(k-1)} + \mathbf{c} \tag{5.2}$$

Ahora, si la ecuación

$$(D - L - U)\mathbf{x} = \mathbf{b}$$

se despeja de la siguiente manera

$$(D - L)\mathbf{x} = U\mathbf{x} + \mathbf{b}$$

y existe $(D-L)^{-1}$, entonces

$$\mathbf{x} = (D-L)^{-1}U\mathbf{x} + (D-L)^{-1}\mathbf{b},$$

lo cual origina el método matricial de Gauss-Seidel

$$\mathbf{x}^{(k)} = G\mathbf{x}^{(k-1)} + \mathbf{s} \tag{5.3}$$

donde $G = (D-L)^{-1}U$ y $\mathbf{s} = (D-L)^{-1}\mathbf{b}$. Es de anotar que $D - L$ es una matriz triangular inferior, y por tanto su inversa existe si los elementos de la diagonal no son nulos, en este caso si $a_{ii} \neq 0$ para $i = 1, 2, \ldots, n$.

Luego de presentar los métodos de Jacobi y Gauss-Seidel como esquemas de la forma $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$, a continuación se tiene una condición suficiente para garantizar su convergencia.

Teorema 9. *Si A es una matriz de diagonal estrictamente dominante, entonces para cualquier semilla $\mathbf{x}^{(0)}$, los método de Jacobi y Gauss-Seidel son convergentes a la única solución del sistema $A\mathbf{x} = \mathbf{b}$.*

Ejercicios 16

1. Dado el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$, donde $A = \begin{pmatrix} 2 & -1 \\ 1 & 4 \end{pmatrix}$ y $b = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, calcular la matriz J y el vector $\mathbf{c}$ usados en el método de Jacobi.

2. Dado el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$, donde $A = \begin{pmatrix} 5 & 1 \\ -1 & -4 \end{pmatrix}$ y $b = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, calcular la matriz G y el vector $\mathbf{s}$ usados en el método de Gauss-Seidel.

3. Determinar si las siguientes matrices son de diagonal estrictamente dominante.

   a) $\begin{pmatrix} -2 & 1 & 0 \\ 5 & 7 & -3 \\ 9 & -1 & 10 \end{pmatrix}$ *b)* $\begin{pmatrix} 6 & 3 & -2 \\ 8 & 15 & -4 \\ -3 & -4 & 5 \end{pmatrix}$

c) $\begin{pmatrix} -3 & 1 & -1 \\ 4 & 5 & -1 \\ 0 & -4 & 5 \end{pmatrix}$ *d)* $\begin{pmatrix} 4 & 1 & 0 & 1 \\ 3 & 6 & 1 & 1 \\ -3 & -1 & -9 & 1 \\ 0 & 1 & -2 & 4 \end{pmatrix}$

4. Descomponer la matriz A en la forma $D - L - U$, donde D es una matriz diagonal, U una matriz triangular superior y L es una matriz triangular inferior.

a) $\begin{pmatrix} 5 & 7 & -3 \\ -4 & 8 & 11 \\ 9 & 3 & 16 \end{pmatrix}$ *b)* $\begin{pmatrix} 4 & 0 & -1 & 4 \\ 3 & 0 & 5 & 3 \\ 13 & 0 & 15 & -2 \\ 57 & 0 & 28 & 0 \end{pmatrix}$

5. Determinar para los siguientes sistemas, si el método de Jacobi (o Gauss-Seidel) es convergente desde cualquier vector inicial $\mathbf{x}^{(0)}$.

a)
$$\begin{aligned} -15x_1 + 3x_2 - 6x_3 &= 3 \\ 8x_1 - 4x_2 + x_3 &= 6 \\ 9x_1 - x_2 + 3x_3 &= 5 \end{aligned}$$

b)
$$\begin{aligned} 4x_1 + x_2 + x_3 &= -5 \\ 3x_1 + 6x_2 - x_3 - x_4 &= 0 \\ -3x_1 + 2x_2 - 9x_3 &= 1 \\ x_2 + 4x - 3 - 7x_4 &= 6 \end{aligned}$$

6. Resolver el sistema lineal dado por medio del método de Jacobi con $\varepsilon = 10^{-4}$.

$$\begin{aligned} 5x + 3y - z &= 2 \\ -x + 2y - 4z &= -2 \\ 2x - 4y - z &= -1 \end{aligned}$$

7. Resolver el sistema lineal dado por medio del método de Gauss-Seidel con $\varepsilon = 10^{-4}$.

$$\begin{aligned} 2x + 5y + z &= -2 \\ 3x - y + z &= 2 \\ -x - y + 3z &= 0 \end{aligned}$$

8. Dada la matriz $A = \begin{pmatrix} \alpha & 1 & 0 \\ \beta & 3 & 2 \\ 1 & -1 & 3 \end{pmatrix}$, encontrar los valores de α y β de forma tal que la matriz sea de diagonal estrictamente dominante.

LAMPE

Capítulo 6

Ecuaciones diferenciales

6.1. Introducción

Cuando se estudian ciertos problemas en ingeniería y física, los modelos matemáticos que allí surgen, corresponden a *ecuaciones* donde se relaciona la *variación* de una magnitud con las condiciones internas, externas e iniciales del sistema. Por ejemplo, si se conoce que la velocidad v de un automóvil (figura 6.1) se encuentra dada por $(2 - 0.2t)$ m/s, donde t representa el tiempo, y la posición inicial del automóvil es 50 m con relación a un punto fijo, entonces una ecuación diferencial y una condición inicial que permiten modelar esta situación son

$$\begin{aligned} v = \frac{dx}{dt} &= 2 - 0.2t \\ x(0) &= 50, \end{aligned} \tag{6.1}$$

donde $x(t)$ es la función de posición del automóvil.

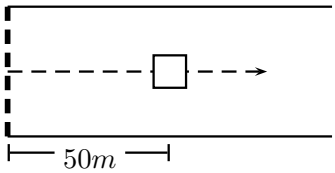


Figura 6.1: una representación de la situación del automóvil.

Una problema del tipo 6.1, se denomina *problema de valor inicial*, donde se tiene una ecuación diferencial y una condición inicial. La solución, a diferencia de las ecuaciones del capítulo 2, corresponde a una función. Por ejemplo, en la situación 6.1, la solución es la función $x(t) = 50 + 2t - 0.1t^2$, dado que $\dfrac{dx}{dt} = 2 - 0.2t$ y $x(0) = 50$.

Al tener la solución del problema, es posible dar respuesta a inquietudes particulares. Por ejemplo, en el modelo del automóvil la respuesta al interrogante ¿qué posición tiene el automóvil a los tres segundos?, corresponde a calcular $x(3) = 50 + 2(3) - 0.1(3)^2 = 55.1$.

En un curso de ecuaciones diferenciales se aprenden diferentes técnicas para determinar la solución *exacta* de la función que satisface el problema de valor inicial. En este capítulo se presentan algunos métodos para construir una *aproximación* a las *imágenes* de la función en valores particulares. Así, para el modelo anterior, se construirá una aproximación de $x(t_i)$ para algunos $t_i \in [0, 3]$.

6.2. Ecuaciones diferenciales ordinarias de primer orden con valor en la frontera

Un problema de valor inicial se puede enunciar como una ecuación diferencial

$$\frac{dx}{dt} = f(t, x), \qquad a \leq t \leq b$$

sujeta a la condición inicial

$$x(a) = \alpha$$

Ejemplo 39. Las siguientes corresponden a problemas de valor inicial.

- La ecuación diferencial

$$\frac{dx}{dt} = -2x + t \qquad 1 \leq t \leq 2$$

sujeta a:

$$x(1) = 5$$

- La ecuación diferencial

$$\frac{dx}{dt} = x^2 + 2xt + t^2 \qquad -5 \leq t \leq -3$$

sujeta a:

$$x(-5) = 8$$

- La ecuación diferencial

$$\frac{dx}{dt} = -t^2 + 1 \qquad 3 \leq t \leq 7$$

sujeta a:

$$x(3) = -2$$

- La ecuación diferencial

$$\begin{aligned} \frac{dx}{dt} &= \frac{x}{t} \qquad 1 \leq t \leq 2 \\ \text{sujeta a:} & \\ x(1) &= 1 \end{aligned}$$

Como se mencionó al principio, dar solución a este tipo de ecuaciones significa encontrar una función $x(t)$ tal que $\frac{dx}{dt} = f(t, x)$ y $x(a) = \alpha$. Aunque existen diferentes técnicas para determinar la función $x(t)$, dichas técnicas están fuera de los alcances de este libro. En su lugar, se presentan métodos para aproximar $x(t_i)$ para $t_i \in [a, b]$.

6.2.1. Método de Euler

Se debe recordar que el problema a solucionar en este capítulo es:

Sea

$$\begin{aligned} \frac{dy}{dt} &= f(t, y) \qquad a \leq t \leq b \\ \text{sujeto a:} & \\ y(a) &= \alpha \end{aligned}$$

un problema de valor inicial. Para $t_i \in [a, b]$, obtener una aproximación de $y(t_i)$, donde $y(t)$ es la solución de la ecuación diferencial.

Aunque el método de Euler es una de la técnicas numéricas más sencillas para aproximar $y(t_i)$, en la práctica no es tan común, dados algunos problemas computacionales que se analizarán más adelante. En el siguiente teorema, que corresponde a una versión particular del teorema de Taylor, se presentan los elementos necesarios para construir el método de Euler.

Teorema 10. *Sea $f : [a, b] \to \mathbb{R}$ una función tal que $f'(x)$ existe, es continua y $f''(x)$ existe en (a, b). Entonces, para $x, x_0 \in [a, b]$ existe ξ entre x y x_0 tal que*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2.$$

La idea del método de Euler consiste entonces en construir una serie de puntos (t_0, y_0), $(t_1, y_1), \ldots (t_n, y_n)$ tales que $t_0 = a$, $t_n = b$ y y_n sea una aproximación a $y(b)$. La primera inquietud a resolver es ¿cúales son los t_i que se deben escoger? Para dar respuesta, recordar

que al tener un problema de valor inicial

$$\frac{dy}{dt} = f(t,y) \qquad a \leq t \leq b$$
$$\text{sujeto a:}$$
$$y(a) = \alpha,$$

si se conoce que $t_i \in [a,b]$ y $y(a) = \alpha$, entonces una selección natural de los t_i, podrían ser valores t_i en el intervalo $[a,b]$ tales que $t_i < t_{i+1}$, $t_0 = a$, $t_n = b$ y además la distancia entre t_i y t_{i+1} es constante[1].

Ejemplo 40. En el intervalo $[1,3]$ seleccionar de manera uniforme seis valores tales que $t_0 = 1$ y $t_5 = 3$.

Solución: dado que se conoce que el primer valor es $t_0 = 1$ y el último valor es $t_5 = 3$, entonces para construir los siguientes puntos se utiliza la fórmula

$$t_i = t_0 + hi \qquad \text{con } i = 1, 2 \ldots, N$$

donde $h = \dfrac{b-a}{N}$, N es número de subintervalos a construir y a, b son los extremos del intervalo. En este caso $a = 1$, $b = 3$, $N = 5$ y por tanto $h = \dfrac{3-1}{5} = 0.4$,

$$t_i = 1 + 0.4i,$$

con lo cual se obtienen los puntos $t_0 = 1$, $t_1 = 1.4$, $t_2 = 1.8$, $t_3 = 2.2$, $t_4 = 2.6$ y $t_5 = 3$. $\diamondsuit$

Luego de seleccionar t_i de manera uniforme, la siguiente pregunta sería ¿qué valor corresponde a y_i para cada t_i? Para lo anterior, suponer que $y(t)$ es la solución del problema de valor inicial

$$\frac{dy}{dt} = f(t,y) \qquad a \leq t \leq b$$
$$\text{sujeto a:}$$
$$y(a) = \alpha$$

y adicionalmente se cumplen las hipótesis del teorema 10. Entonces para t_i y t_{i+1} se tiene

$$y(t_{i+1}) = y(t_i) + y'(t_i)(t_{i+1} - t_i) + \frac{y''(\xi)}{2}(t_{i+1} - t_i)^2$$

y dado que $y'(t) = \dfrac{dy}{dt} = f(t,y)$, entonces

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))(t_{i+1} - t_i) + \frac{y''(\xi)}{2}(t_{i+1} - t_i)^2$$

[1]Este tipo de selección se denomina *uniforme*.

Ahora, $t_{i+1} - t_i = h$ pues los t_i fueron seleccionados de manera uniforme, en cuyo caso

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))h + \frac{y''(\xi)}{2}h^2,$$

y por tanto

$$y(t_{i+1}) \approx y(t_i) + f(t_i, y(t_i))h.$$

Si se toma $y(t_i) = y_i$, entonces

$$y_{i+1} \approx y_i + hf(t_i, y_i),$$

que corresponde a una relación recursiva entre los y_i buscados. Como y_0 es la imagen de $t_0 = a$ y $y(a) = \alpha$, entonces $y_0 = \alpha$. Así, el método de Euler corresponde al esquema:

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + f(t_i, y_i)h \end{aligned} \tag{6.2}$$

Ejemplo 41. Utilizar el método de Euler con $N = 10$ para aproximar la solución al problema de valor inicial

$$\begin{aligned} y' &= (2 - y)t \qquad 1 \leq t \leq 3 \\ &\text{sujeto a:} \\ y(1) &= -5 \end{aligned}$$

Solución: para utilizar el método de Euler, se ejecutan los siguientes pasos.

Paso 1. Seleccionar t_i de manera uniforme. En este caso el intervalo es $[1, 3]$ y $N = 10$, luego $h = \dfrac{b-a}{N} = \dfrac{3-1}{10} = 0.2$ y por tanto se tienen los resultados del cuadro 6.1.

t_i	1	1.2	1.4	1.6	1.8	2	2.2	2.4	2.6	2.8	3

Cuadro 6.1: método de Euler.

Paso 2. Calcular los y_i. Para esto se utiliza el esquema recursivo definido en la ecuación 6.2 que corresponde, en este caso, a

$$\begin{aligned} y_0 &= -5 \\ y_{i+1} &= y_i + 0.2\left[(2 - y_i)t_i\right], \end{aligned}$$

donde los resultados numéricos se encuentran en el cuadro 6.2.

$\diamondsuit$

t_i	y_i
1.0	−5.000000000000
1.2	−3.600000000000
1.4	−2.256000000000
1.6	−1.064320000000
1.8	−0.083737600000
2.0	0.666407936000
2.2	1.199844761600
2.4	1.551913066496
2.6	1.766994794578
2.8	1.888157501397
3.0	1.950789300615

Cuadro 6.2: método de Euler.

Ejemplo 42. Aplicar el método de Euler con $N = 5$ para aproximar la solución del problema de valor inicial

$$y' = 2 - 0.2t \qquad 0 \leq t \leq 3$$

sujeta a:

$$y(0) = 50$$

Solución: al ejecutar los pasos descritos anteriormente se obtiene:

Paso 1. Seleccionar t_i de manera uniforme, en este caso cinco puntos. Luego $h = 0.6$ y por tanto $t_0 = 0$, $t_1 = 0.6$, $t_2 = 1.2$, $t_3 = 1.8$, $t_4 = 2.4$ y $t_5 = 3$.

Paso 2. Calcular los y_i utilizando el esquema recursivo 6.2, se tiene

$$y_0 = 50$$
$$y_{i+1} = y_i + 0.6[(2 - t_i)]$$

con los resultados del cuadro 6.3. Ahora, si se utilizan diez puntos, se tienen los resultados del cuadro 6.4. Una aproximación a $y(3)$ es 55.28 en el primer caso y 55.19 en el segundo caso, lo que corresponde a un valor más cercano al real $y(3) = 55.1$.

♢

6.2.2. Orden del método de Euler

El método de Euler se obtuvo de la expansión en serie de potencia de la función alrededor de $x = 0$. En el caso más general, si la función se conoce completamente en cierto valor $x = x_0$, la expansión en serie de Taylor es

$$f(x_0 + \Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{f''(x_0)}{2!}(\Delta x)^2 + \frac{f'''(x_0)}{3!}(\Delta x)^3 + \cdots,$$

t_i	y_i
0.0	50.000
0.6	51.200
1.2	52.328
1.8	53.384
2.4	54.368
3.0	55.280

Cuadro 6.3: método de Euler.

t_i	y_i
0.0	50.000
0.3	50.600
0.6	51.182
0.9	51.746
1.2	52.292
1.5	52.820
1.8	53.330
2.1	53.822
2.4	54.296
2.7	54.752
3.0	55.190

Cuadro 6.4: método de Euler.

donde se supone que Δx es un paso muy pequeño, o de manera práctica $|\Delta x| \ll 1$. Como se pudo observar en la sección 2.5, si Δx es pequeño, su cuadrado lo es más aún, y por lo tanto el método comete un error más pequeño. En el método de Euler, la serie se corta en el segundo término, y por tanto el error lo domina el tercer término

$$f(x_0 + \Delta x) \approx f(x_0) + f'(x_0)\Delta x + \frac{f''(\xi)}{2!}(\Delta x)^2,$$

donde el término $\dfrac{f''(\xi)}{2!}(\Delta x)^2$ es el error que se comete en cada paso. Así, el método de Euler comete un error de orden $(\Delta x)^2$ en cada paso. El error total cuando se han dado N pasos es

$$E_{\text{total}} = (\text{Error de cada paso}) \times N,$$

donde N es la longitud del intervalo $[a, b]$ dividida entre el valor del paso Δx. En dicho caso

$$E_{\text{total}} = \frac{f''(\xi)}{2!}(\Delta x)^2 \frac{|b-a|}{\Delta x},$$

de donde al cancelar y reunir todas las constantes, se obtiene

$$E_{\text{total}} = K\Delta x,$$

donde K es una constante, o en forma equivalente $E_{\text{total}} = O(\Delta x)$ al usar la notación de la sección 1.6. En lo sucesivo se construirán métodos que mejoren la precisión sin sacrificar el tamaño del paso.

6.2.3. Método de Verlet

La expresión de este método se obtiene a partir de dos expansiones de Taylor, una hacia adelante y otra hacia atrás, a saber:

$$f(x_0 + \Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{f''(x_0)}{2!}(\Delta x)^2 + \frac{f'''(x_0)}{3!}(\Delta x)^3 + \frac{f^{(4)}(x_0)}{4!}(\Delta x)^4 \cdots,$$

$$f(x_0 - \Delta x) = f(x_0) - f'(x_0)\Delta x + \frac{f''(x_0)}{2!}(\Delta x)^2 - \frac{f'''(x_0)}{3!}(\Delta x)^3 + \frac{f^{(4)}(x_0)}{4!}(\Delta x)^4 \cdots,$$

Al sumar dichas series se tiene que

$$f(x_0 + \Delta x) + f(x_0 - \Delta x) = 2f(x_0) + f''(x_0)(\Delta x)^2 + 2\frac{f^{(4)}(x_0)}{4!}(\Delta x)^4 \cdots$$

de donde al despejar $f(x_0 + \Delta x)$, se tiene que

$$f(x_0 + \Delta x) = 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)(\Delta x)^2 + 2\frac{f^{(4)}(x_0)}{4!}(\Delta x)^4 \cdots.$$

Al truncar la última expresión en el tercer término

$$f(x_0 + \Delta x) = 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)(\Delta x)^2,$$

se tiene la siguiente fórmula recursiva, conocida como método de Verlet:

$$y_{i+1} = 2y_i - y_{i-1} + y_i'' h^2. \tag{6.3}$$

Esta fórmula es de amplia aplicación en física, en la cual la primera derivada de la posición respecto al tiempo corresponde a la velocidad y la segunda a la aceleración. Observar que la fórmula recursiva no depende de la primera derivada (velocidad), pero sí de la segunda derivada, la aceleración. La anterior situación abarca una gran cantidad de problemas físicos en los que la evolución temporal de la posición no depende del valor de la velocidad. Aunque a simple vista el método de Verlet demuestra sus ventajas, tiene un pequeño problema: es necesario el dato y_{-1} no disponible para calcular la primera entrada de la tabla. Dado lo anterior, es necesario implementar un arranque. A partir de la expansión de Taylor hacia atrás

$$y_{-1} = y_0 - y_0' h + \frac{1}{2} y_0'' h^2$$

se completa la construcción del método para solucionar el problema $y'' = F(y)$ con condiciones iniciales $y_0 = \alpha$ y $y_0' = \beta$ y paso h:

$$\begin{aligned} y_0 &= \alpha \\ y_{-1} &= y_0 - \beta h + \frac{1}{2} F(y_0)\, h^2 \quad \text{(arranque)} \\ y_{i+1} &= 2y_i - y_{i-1} + F(y_i)\, h^2 \end{aligned} \tag{6.4}$$

Ejemplo 43. Aplicar el método de Verlet para aproximar la caída libre de un cuerpo que es soltado desde una altura de 100 m. Comparar los resultados con la solución analítica y con la aproximación de Euler.

Solución: la caída libre se enuncia como el problema de valor inicial:

$$\begin{aligned} y'' &= -9.8 \qquad 0 \leq t \leq 4.5 \\ y(0) &= 100 \\ y'(0) &= 0 \end{aligned}$$

Al aplicar el método Verlet se obtiene el esquema iterativo

$$\begin{aligned} y_0 &= 100 \\ y_{-1} &= 100 - 4.9h^2 \quad \text{(arranque)} \\ y_{i+1} &= 2y_i - y_{i-1} - 4.9h^2 \end{aligned}$$

donde la tabla 6.5 resume los resultados numéricos, y también expone las diferencias entre los métodos de Verlet, Euler y la solución analítica al problema.

t_i	y_i Verlet	y_i Euler	y_i exacto
0.00	100.000	100.000	100.000
0.45	99.008	98.016	99.008
0.90	96.031	94.047	96.031
1.35	91.070	88.093	91.070
1.80	84.124	80.155	84.124
2.25	75.194	70.233	75.194
2.70	64.279	58.326	64.279
3.15	51.380	44.434	51.380
3.60	36.496	28.558	36.496
4.05	19.628	10.698	19.628
4.50	0.775	−9.147	0.775

Cuadro 6.5: comparación entre algunos métodos.

♢

6.2.4. Error del método de Verlet

Para analizar el error, se debe observar que antes de cortar la serie, el error lo domina el término $(\Delta x)^4$, pues

$$f(x_0 + \Delta x) \approx 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)(\Delta x)^2 + 2\frac{f^{(4)}(\xi)}{4!}(\Delta x)^4.$$

En cada paso, el método de Verlet comete entonces un error de orden $(\Delta x)^4$. El error total cuando se han dado N pasos, es:

$$E_{\text{total}} = 2\frac{f^{(4)}(\xi)}{4!}(\Delta x)^4 \frac{|b-a|}{\Delta x},$$

cancelando y reuniendo todas las constantes, se tiene

$$E_{\text{total}} = K(\Delta x)^3,$$

donde K es una constante, o en forma equivalente $E_{\text{total}} = O\left((\Delta x)^3\right)$. Se concluye entonces que método de Verlet es de tercer orden.

Ejercicios 17

1. Encontrar la solución general de la ecuación diferencial lineal de primer orden $y' + \frac{2}{t}y = \frac{\cos t}{t}$.
2. Encontrar la solución general de la ecuación diferencial lineal de primer orden $ty' + (t+1)y = t$.
3. Encontrar la solución general de la ecuación diferencial lineal de primer orden: $t^3y' + 4t^2y = e^{-t}$.
4. Utilizar el método de Euler para aproximar la solución de los siguientes problemas de valor inicial.
   a) $y' = -y$, $0 \le t \le 1$, $y(0) = 5$ y $N = 10$.
   b) $y' = -2y + t$, $1 \le t \le 3$, $y(1) = 2$ y $N = 20$.
   c) $y' = \frac{y}{t}$, $-3 \le t \le -1$, $y(-3) = 2$ y $N = 100$.
   d) $y' = \cos(2t) + \sin(3t)$, $0 \le t \le 1$, $y(0) = 2$ y $h = 0.0001$.
5. Dado el problema de valor inicial

$$y' = \frac{2}{t}y + e^{-t} \qquad 2 \le t \le 4$$

sujeto a:

$$y(2) = 0$$

   a) Utilizar el método de Euler con $N = 10$ para aproximar su solución.
   b) Con los resultados del numeral *a)*, construir el interpolador de Newton y aproximar $y(2.5)$, $y(3.7)$ y $y(3.9)$.
6. Dado el problema de valor inicial

$$y' = -2ty \qquad 0 \le t \le 1$$

sujeto a:

$$y(0) = 1,$$

si se conoce el valor correcto $y(1) = e^{-1}$, determinar el número de puntos necesarios en el método de Euler para obtener una aproximación con un error inferior a $\varepsilon = 10^{-4}$.

7. Si se conoce que la velocidad de un auto es $(5 - 0.4t)$ m/s y su posición inicial es $x(0) = 30$ m, utilizar el método de Euler para aproximar la posición del auto después de cinco segundos.

8. En el circuito que se muestra en la figura 6.2, el voltaje v_C a través del condensador se encuentra dado por $C\dfrac{dv_C}{dt} + \dfrac{v_C}{R} = 0$. Si $C = 10^{-6}$ F, $R = 10^6\ \Omega$ y se tiene la condición inicial $v_C(0) = 5$, usar el método de Euler con $N = 10$ para aproximar $v_C(1)$.

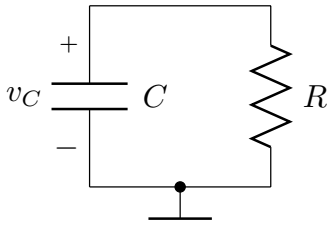


Figura 6.2: circuito RC.

9. Encontrar la solución general de la ecuación diferencial $y'' + 2y' - 3y = 0$.

10. Encontrar la solución general de la ecuación diferencial $y'' + 2y' + y = 0$.

11. Encontrar la solución general de la ecuación diferencial $4y'' + 12y' + 9y = 0$.

12. Utilizar el método de Verlet para aproximar la solución de los siguientes problemas de valor inicial.

    a) $y'' = y$, $0 \le t \le 1$, $y(0) = 1$, $y'(0) = 1$ y $N = 10$.
    b) $y'' = -y$, $0 \le t \le 1$, $y(0) = 1$, $y'(0) = 0$ y $N = 10$.
    c) $y'' = 4y$, $0 \le t \le 2$, $y(0) = 0$, $y'(0) = 14$ y $N = 20$.
    d) $y'' = -9y$, $0 \le t \le 2$, $y(0) = 1$, $y'(0) = 6$ y $N = 20$.

6.2.5. Métodos de Runge-Kutta orden dos

En la búsqueda de métodos más precisos en la solución de problemas de valor inicial, los métodos de Runge-Kutta ofrecen una mejor alternativa. Estos métodos evitan el cálculo de derivadas, lo cual permite su fácil implementación.

Para entender el origen de los métodos de Runge-Kutta observar la serie de Taylor:

$$y(t) = y(t_i) + y'(t_i)(t - t_i) + \frac{y''(t_i)}{2!}(t - t_i)^2 + \frac{y'''(t_i)}{3!}(t - t_i)^3 + \cdots$$

Ahora, al evaluar en $t_i + h$, se obtiene:

$$y(t_i + h) = y(t_i) + y'(t_i)(t_i + h - t_i) + \frac{1}{2!}y''(t_i)(t_i + h - t_i)^2 + \frac{y'''(t_i)}{3!}(t_i + h - t_i)^3 + \cdots$$

y al simplificar se concluye que

$$y(t_i + h) = y(t_i) + h\ y'(t_i) + \frac{h^2}{2!}y''(t_i) + \frac{h^3}{3!}y'''(t_i) + \cdots . \tag{6.5}$$

Se podría pensar que entre más términos de la serie se tomen, la aproximación de $y(t_i + h)$ es más exacta, y aunque lo anterior es cierto, no es aplicable por la dificultad de calcular las derivadas. En los métodos de Runge-Kutta, la idea es determinar expresiones que *aproximen* los primeros términos de la serie de Taylor y que su cálculo no conlleve el manejo de derivadas. Para comprender esta idea, observar la siguiente deducción de un método de Runge-Kutta de orden dos.

Sea $g(x, y)$ una función de dos variables. Si existen todas sus derivadas parciales de orden menor o igual a dos y son continuas en un dominio $D = \{(x, y) \mid a \leq x \leq b, c \leq y \leq d\}$, entonces para $(x_0, y_0) \in D$ y $(x, y) \in D$ existen ξ entre x_0 y x, y μ entre y_0 y y tales que

$$g(x, y) = g(x_0, y_0) + \frac{\partial g}{\partial x}\Big|_{(x_0,y_0)}(x - x_0) + \frac{\partial g}{\partial y}\Big|_{(x_0,y_0)}(y - y_0) + R_2(x, y)$$

donde

$$R_2(x, y) = \frac{1}{2!}\sum_{j=0}^{2}\binom{2}{j}\frac{\partial^2 g}{\partial x^{2-j}\partial y^j}\Big|_{(\xi,\mu)}(x - x_0)^{2-j}(y - y_0)^j$$

A la función $P_1(x, y) = g(x_0, y_0) + \frac{\partial g}{\partial x}\Big|_{(x_0,y_0)}(x - x_0) + \frac{\partial g}{\partial y}\Big|_{(x_0,y_0)}(y - y_0)$ se le denomina *polinomio de Taylor de orden uno* de la función $g(x, y)$ alrededor del punto (x_0, y_0).

Ejemplo 44. Dada $g(x, y) = \cos(x) + \sin(y)$, $(x_0, y_0) = \left(\frac{\pi}{3}, \frac{\pi}{6}\right)$, entonces su polinomio de Taylor de orden uno es:

$$\begin{aligned} P_1(x, y) &= g\left(\frac{\pi}{3}, \frac{\pi}{6}\right) + \frac{\partial g}{\partial x}\Big|_{\left(\frac{\pi}{3},\frac{\pi}{6}\right)}\left(x - \frac{\pi}{3}\right) + \frac{\partial g}{\partial y}\Big|_{\left(\frac{\pi}{3},\frac{\pi}{6}\right)}\left(y - \frac{\pi}{6}\right) \\ &= 1 - \frac{\sqrt{3}}{2}\left(x - \frac{\pi}{3}\right) + \frac{\sqrt{3}}{2}\left(y - \frac{\pi}{6}\right) \end{aligned}$$

Ahora, volviendo a la discusión anterior, si de la serie 6.5 se toman los tres primeros términos, entonces

$$y(t_i + h) \approx y(t_i) + h\ y'(t_i) + \frac{h^2}{2}y''(t_i).$$

Ahora, dado que $y' = f(t, y)$ en el problema de valor inicial descrito en este capítulo, al reemplazar se tiene que

$$y(t_i + h) \approx y(t_i) + hf(t_i, y_i) + \frac{h^2}{2}f'(t_i, y_i),$$

y al usar que (regla de la cadena) $f'(t,y) = \dfrac{\partial f}{\partial t}\dfrac{dt}{dt} + \dfrac{\partial f}{\partial y}\dfrac{dy}{dt}$, se tiene que

$$y(t_i + h) \approx y(t_i) + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{\partial f}{\partial y}\Big|_{(t_i,y_i)}\frac{dy}{dt}\Big|_{t_i}\right),$$

lo cual es equivalente a

$$\begin{aligned} y(t_i + h) &\approx y(t_i) + h\left(f(t_i, y_i) + \frac{h}{2}\left(\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{\partial f}{\partial y}\Big|_{(t_i,y_i)}\frac{dy}{dt}\Big|_{t_i}\right)\right) \\ &= y(t_i) + h\left(f(t_i, y_i) + \frac{h}{2}\left(\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{\partial f}{\partial y}\Big|_{(t_i,y_i)} f(t_i, y_i)\right)\right). \end{aligned} \tag{6.6}$$

Si ahora se aproxima $f(t_i, y_i) + \dfrac{h}{2}\dfrac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \dfrac{h}{2}\dfrac{\partial f}{\partial y}\Big|_{(t_i,y_i)} f(t_i, y_i)$ con la serie de Taylor de orden uno de la función $a_1 f(t_i + \alpha_1, y_i + \beta_1)$ se tiene

$$P_1(t_i, y_i) = a_1 f(t_i, y_i) + a_1\alpha_1\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + a_1\beta_1\frac{\partial f}{\partial y}\Big|_{(t_i,y_i)}$$

de donde se concluye que si $a_1 = 1$, $\alpha_1 = \dfrac{h}{2}$ y $\beta_1 = \dfrac{h}{2}f(t_i, y_i)$, entonces

$$f(t_i, y_i) + \frac{h}{2}\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{h}{2}\frac{\partial f}{\partial y}\Big|_{(t_i,y_i)} f(t_i, y_i) \approx f\left(t_i + \frac{h}{2}, y_1 + \frac{h}{2}f(t_i, y_i)\right)$$

Finalmente, al reemplazar en la ecuación 6.6 se tiene

$$y(t_i + h) \approx y(t_i) + hf\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)\right),$$

lo cual origina el siguiente método para aproximar la solución de un problema de valor inicial.

Método del punto medio:

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + hf\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)\right) \end{aligned} \tag{6.7}$$

Ejemplo 45. Utilizando el método del punto medio, aproximar la solución del problema de valor inicial. Usar $N = 5$.

$$\begin{aligned} y' &= -2ty \qquad 0 \le t \le 1 \\ &\text{sujeto a:} \\ y(0) &= 1 \end{aligned}$$

Solución: para utilizar el método, se aplican los siguientes pasos:

Paso 1. Seleccionar t_i de manera uniforme, para este ejemplo se seleccionan cinco puntos, luego $h = \dfrac{b-a}{N} = \dfrac{1-0}{5} = 0.2$ y por tanto $t_0 = 0$, $t_1 = 0.2$, $t_2 = 0.4$, $t_3 = 0.6$, $t_4 = 0.8$ y $t_5 = 1$.

Paso 2. Calcular los y_i utilizando el esquema recursivo definido en 6.7, que para propósitos de cálculo se puede expresar como:

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ y_{i+1} &= y_i + hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right) \end{aligned}$$

y que corresponde para este caso a

$$\begin{aligned} y_0 &= 1 \\ k_1 &= h\left(-2t_i y_i\right) \\ y_{i+1} &= y_i + h\left(-2\left(t_i + \frac{h}{2}\right)\left(y_i + \frac{1}{2}k_1\right)\right) \end{aligned}$$

En el cuadro 6.6 se presentan los resultados del método y al comparar con la solución real (cuadro 6.7) del problema de valor inicial, se evidencia la precisión del método.

♢

t_i	y_i
0.0	1.0000
0.2	0.9600
0.4	0.8494
0.6	0.6931
0.8	0.5223
1.0	0.3643

Cuadro 6.6: aproximación del problema $y' = -2ty$.

Este método, en comparación con el método de Euler, presenta una mejor aproximación de la solución de un problema de valor inicial. Es posible obtener otros métodos de Runge-Kutta orden dos con las ideas desarrolladas en la deducción del método del punto medio, entre los cuales se tienen los siguientes.

Método modificado de Euler:

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ y_{i+1} &= y_i + \frac{1}{2}\left(k_1 + hf\left(t_{i+1}, y_i + k_1\right)\right) \end{aligned}$$

t_i	y_i
0.0	1.0000
0.2	0.9607
0.4	0.8521
0.6	0.6976
0.8	0.5272
1.0	0.3678

Cuadro 6.7: solución real del problema $y' = -2ty$.

Método de Heun (también conocido como método de Ralston):

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ y_{i+1} &= y_i + \frac{1}{4}\left(k_1 + 3hf\left(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_1\right)\right) \end{aligned}$$

Ejemplo 46. Utilizar el método modificado de Euler y el de Heun con $N = 5$, para aproximar soluciones del problema de valor inicial

$$y' = -3t^2(y+1) \qquad 1 \leq t \leq 2$$

sujeto a:

$$y(1) = -2$$

cuya solución real es $y(t) = -e^{-t^3+1} - 1$.

Solución: dado que $N = 5$ entonces $h = 0.2$. En el cuadro 6.8 se resumen los resultados.

t_i	$y(t_i)$	método modificado de Euler	error	método de Heun	error
1.0	−2.0000	−2.0000	0.0000	−2.0000	0.0000
1.2	−1.4828	−1.0088	0.4740	−1.5032	0.0203
1.4	−1.1748	−0.9953	0.1794	−1.2238	0.0490
1.6	−1.0452	−1.0058	0.0393	−1.1068	0.0616
1.8	−1.0079	−0.9869	0.0210	−1.0692	0.0613
2.0	−1.0009	−1.0457	0.0448	−1.0701	0.0692

Cuadro 6.8: solución del problema $y' = -3t^2(y+1)$.

♢

Se debe anotar que, en el ejemplo anterior, conocer la solución exacta solamente permite medir el error entre los métodos. En la práctica se asume que la solución exacta es desconocida y no se desea *conocer* sino *aproximar*.

Ejercicios 18

1. Utilizar el método del punto medio para aproximar la solución de los siguientes problemas de valor inicial.

   a) $y' = -2t + yt$, $1 \leq t \leq 2$, $y(1) = -3$ y $N = 5$.

   b) $y' = -2e^{-ty}$, $0 \leq t \leq 3$, $y(0) = 1$ y $N = 10$.

   c) $y' = -2\cos(2t) + \sin(y)$, $-2 \leq t \leq -1$, $y(-1) = 0$ y $N = 10$.

   d) $y' = 1 + y/t$, $-2.5 \leq t \leq -0.5$, $y(-2.5) = 1$ y $N = 10$.

   e) $y' = 5t^2 - 3t + e^{-t}$, $2 \leq t \leq 4$, $y(2) = 2$ y $N = 15$.

2. Repetir el primer ejercicio aplicando el método modificado de Euler.

3. Repetir el primer ejercicio aplicando el método de Heun.

4. Sea

$$\begin{aligned} y' &= y^2 t^{3/2} \qquad 4 \leq t \leq 5 \\ \text{sujeto a:} & \\ y(4) &= 5 \end{aligned}$$

   un problema de valor inicial.

   a) Aproximar la solución utilizando $h = 0.001$ y el método de Heun.

   b) Si conoce que la solución exacta es $y(t) = \dfrac{5}{65 - 2t^{5/2}}$, comparar los resultados reales con los aproximados en el numeral anterior.

5. Sea

$$\begin{aligned} y' &= -y + t + 1 \qquad 0 \leq t \leq 1 \\ \text{sujeto a:} & \\ y(0) &= 1 \end{aligned}$$

   un problema de valor inicial. Para cualquier elección de h, mostrar que los métodos de Euler y Heun dan las mismas aproximaciones a la solución del problema.

6. Utilizar algún método de esta sección para aproximar

$$\int_0^1 \sin\left(\sqrt{x}\right) dx$$

   Sugerencia: Definir $y(t) = \displaystyle\int_0^t \sin\left(\sqrt{x}\right) dx$ y utilizar el teorema fundamental del cálculo para construir un problema de valor inicial.

6.2.6. Método de Runge-Kutta orden cuatro (RK4)

El método de Runge-Kutta de orden cuatro (RK4) es uno de los métodos más utilizados en la práctica. Su deducción sigue las ideas de los métodos de Runge-Kutta orden dos y como los anteriores métodos, es un método recursivo y su cálculo no implica utilizar información adicional como ocurre en los métodos de Taylor. Su esquema se encuentra a continuación.

Método de Runge-Kutta de orden cuatro:

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ k_2 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right) \\ k_3 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right) \\ k_4 &= hf(t_i + h, y_i + k_3) \\ y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

Ejemplo 47. Utilizar el método de Runge-Kutta de orden cuatro para aproximar la solución del siguiente problema de valor inicial.

$$y' = y - t^2 + 1 \qquad 0 \leq t \leq 2$$

sujeto a:

$$y(0) = 0.5$$

con $N = 10$.

Solución: como $N = 10$ se tiene que $h = 0.2$. En el cuadro 6.9 se incluyen los resultados numéricos del método y se comparan con la solución exacta dada por $y(t) = (t+1)^2 - 0.5e^t$. ♢

Aunque los resultados del cuadro 6.9 dan muestra de la precisión del método de Runge-Kutta orden cuatro (RK4), un elemento a considerar es la cantidad de evaluaciones de la función $f(t, y)$ que utiliza el método en comparación con los métodos de Euler, modificado de Euler, punto medio y Heun.

Otra característica del método RK4, radica en que sí en un método de Runge-Kutta orden dos se utiliza un tamaño de paso h, entonces en el método RK4 es suficiente utilizar un tamaño de paso igual a $2h$ para alcanzar una mejor precisión. Lo anterior, de manera muy informal, brinda un equilibrio entre la cantidad de evaluaciones de la función $f(t, y)$ entre los diferentes métodos.

Ejercicios 19

1. Utilizar el método de Runge-Kutta de orden cuatro para aproximar la solución de las siguientes problemas de valor inicial.

t_i	y_i RK4	y_i exacto
0.00	0.5000000000	0.5000000000
0.20	0.8292933333	0.8292986209
0.40	1.2140762107	1.2140876512
0.60	1.6489220170	1.6489405998
0.80	2.1272026849	2.1272295358
1.00	2.6408226927	2.6408590858
1.20	3.1798941702	3.1799415386
1.40	3.7323400729	3.7324000166
1.60	4.2834094983	4.2834837878
1.80	4.8150856946	4.8151762678
2.00	5.3053630007	5.3054719505

Cuadro 6.9: soluciones al problema $y' = y - t^2 + 1$.

a) $y' = -2t + yt$, $1 \leq t \leq 2$, $y(1) = -3$ y $N = 5$.

b) $y' = -2e^{-ty}$, $0 \leq t \leq 3$, $y(0) = 1$ y $N = 10$.

c) $y' = -2\cos(2t) + \sin(y)$, $-2 \leq t \leq -1$, $y(-1) = 0$ y $N = 10$.

d) $y' = 1 + y/t$, $-2.5 \leq t \leq -0.5$, $y(-2.5) = 1$ y $N = 10$.

e) $y' = 5t^2 - 3t + e^{-t}$, $2 \leq t \leq 4$, $y(2) = 2$ y $N = 15$.

2. Repetir el anterior ejercicio usando método de Heun. Comparar con el método de Runge-Kutta de orden cuatro.

3. Sea

$$y' = y^2 t^{3/2} \qquad 4 \leq t \leq 5$$

sujeto a:

$$y(4) = 5$$

un problema de valor inicial.

a) Aproximar la solución utilizando $h = 0.05$ y el método de Heun.

b) Aproximar la solución utilizando $h = 0.1$ y el método RK4. Comparar con los resultados del numeral anterior.

Apéndices

Apéndice A

Tutorial de Python

A.1. Generalidades

Python es un lenguaje de programación de alto nivel, multi-paradigma y multi-propósito. Es un lenguaje interpretado donde se enfatiza la legibilidad para construir expresiones efectivas en pocas líneas de código. Estos últimos aspectos se pueden evidenciar con el tradicional programa "Hola Mundo", que utiliza solamente una línea de código en Python:

```
1 print("Hola Mundo")
```

Al ejecutar el anterior código, se obtiene, como es de esperar:

Salida

```
Hola Mundo
```

A.2. Funciones

La construcción de las funciones se hace con la palabra clave `def`, seguida del nombre de la función y sus argumentos entre paréntesis. La construcción termina con `:` y en la siguiente línea, *con un sangrado* (por convención de cuatro espacios en blanco) se prosigue al bloque donde se encuentra el cuerpo de la función. Como se verá más adelante en las estructuras de control, el sangrado es fundamental en Python.

```
1 def␣f(entrada):
2 ␣␣␣␣return␣entrada**3
3 print(f(7))
```

Salida

```
343
```

A.3. Estructuras básicas de control

A.3.1. if

```
x = 5
y = 8
if x == y:
    print("x es igual a y")
elif x < y:
    print("x es menor a y")
else:
    print("x es mayor a y")
```

Salida

```
x es menor a y
```

A.3.2. while

```
i = 1
while i < 5:
    print(6*i + 1)
    i += 1
```

Salida

```
7
13
19
25
```

A.3.3. for

```
for i in range(4):
    print(i*i + 3)
```

Salida

```
3
4
```

```
7
12
```

A.4. Ejemplos

A.4.1. Factorial

```
def␣factorial(n):
␣␣␣␣if␣n␣==␣0:
␣␣␣␣␣␣␣␣return␣1
␣␣␣␣else:
␣␣␣␣␣␣␣␣return␣n*factorial(n-1)

print(factorial(5))
print(factorial(10))
```

Salida

```
120
3628800
```

A.4.2. GCD

```
def␣gcd(a,␣b):
␣␣␣␣if␣a␣==␣b:
␣␣␣␣␣␣␣␣return␣a
␣␣␣␣if␣b␣<␣a:
␣␣␣␣␣␣␣␣return␣gcd(a-b,␣b)
␣␣␣␣if␣b␣>␣a:
␣␣␣␣␣␣␣␣return␣gcd(a,␣b-a)

print(gcd(75,␣60))
print(gcd(7,␣13))
print(gcd(1024,␣888))
```

Salida

```
15
1
8
```

A.4.3. ¿Es palíndromo?

```
def espalin(palabra):
    if len(palabra) < 2:
        return True
    elif palabra[0] != palabra[-1]:
        return False
    else:
        return espalin(palabra[1:-1])

print(espalin("python"))
print(espalin("no"))
print(espalin("y"))
print(espalin("anilina"))
```

Salida

```
False
False
True
True
```

A.4.4. NumPy

`numpy` es un poderoso módulo de `Python` para cálculo científico con arreglos de datos multidimensionales. Brinda a `Python` de una gran funcionalidad y es ampliamente usado por la comunidad científica. Recientemente ha contribuido a la generación de la primera imagen de un agujero negro y también en la detección de ondas gravitacionales.

```
import numpy as np
from numpy.linalg import solve, inv

a = np.array([[-1.1, 2.5], [1.3, 4.2]])  # matriz forma (2,2)
print(a)
print(a.T)  # transpuesta
print(inv(a))  # inversa

b = np.array([[2], [-3]]) # vector forma (2,1)
print(b)
s = solve(a, b)  # solucionar el sistema de ecuaciones
print(s)
```

Salida

```
[[-1.1  2.5]
 [ 1.3  4.2]]
```

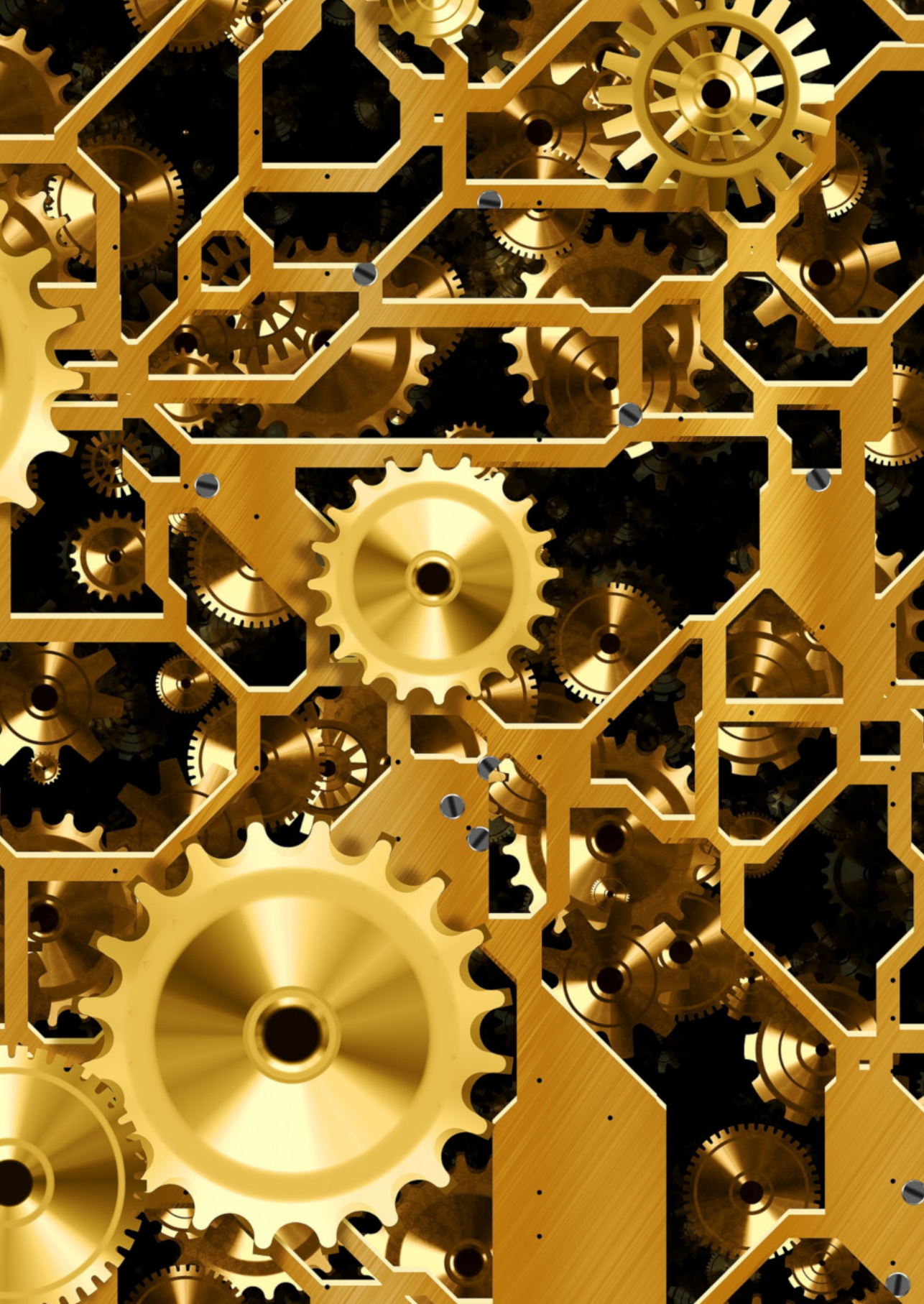
```
[[-1.1  1.3]
 [ 2.5  4.2]]
[[-0.53367217  0.31766201]
 [ 0.16518424  0.13977128]]
[[ 2]
 [-3]]
[[-2.02033037]
 [-0.08894536]]
```

Ejercicios 20

1. Calcular el valor de la expresión $\ln\left(4+\cos\left(\frac{\pi}{4}\right)\right)+\sqrt{\arctan\left(7+\frac{\pi}{3}\right)}$
2. Diseñar una función que imprima los n primeros números triangulares, esto es, números de la forma $\dfrac{i(i+1)}{2}$ donde i es un entero positivo.
3. Dado un entero positivo n, diseñar una función que retorne la suma de sus dígitos.
4. Encontrar la solución del sistema de ecuaciones

$$\begin{aligned} 2x+y-z &= 2 \\ 2x-y+2z &= -1 \\ x+y-z &= 3 \end{aligned}$$

5. Dada la función $f(n)=\begin{cases} n/2 & \text{si } n \text{ es par} \\ 3n+1 & \text{si } n \text{ es impar} \end{cases}$, diseñar un procedimiento que retorne el i-ésimo término de la sucesión $a_i=\begin{cases} m & \text{para } i=0 \\ f(a_{i-1}) & \text{para } i>0 \end{cases}$, donde m es un entero positivo arbitrario.



Apéndice B

Compendio de programas

B.1. Búsqueda de raíces

B.1.1. Bisección

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ------------------------------------------------------------------------

# Implementación del método de bisección y algunos casos de salida.

from math import *


```

```
25 def pol(x):
26     """Función de prueba"""
27     return x**3 + 4*x**2 - 10  # retorna pol(x) = x^3 + 4x^2 - 10
28
29
30 def trig(x):
31     """Función de prueba"""
32     return x*cos(x-1) - sin(x)  # retorna trig(x) = x cos(x - 1) - sin(x)
33
34
35 def bisec(f, a, b, tol, n):
36     """
37     Implementación método de bisección
38     Entradas:
39     f -- función
40     a -- inicio intervalo
41     b -- fin intervalo
42     tol -- tolerancia
43     n -- número máximo de iteraciones
44
45     Salida:
46     p aproximación a cero de f
47     None en caso de iteraciones agotadas
48     """
49     i = 1
50     while i <= n:
51         p = a + (b - a)/2
52         print("i = {0:<2}, p = {1:.12f}".format(i, p))
53         if abs(f(p)) <= 1e-15 or (b - a)/2 < tol:
54             return p
55         i += 1
56         if f(a)*f(p) > 0:
57             a = p
58         else:
59             b = p
60     print("Iteraciones agotadas: Error!")
61     return None
62
63
64 # pol(x), a = 1, b = 2, TOL = 10^-8, N_0 = 100
65 print("Bisección función pol(x):")
66 bisec(pol, 1, 2, 1e-8, 100)
67
68 # trig(x), a = 4, b = 6, TOL = 10^-8, N_0 = 100
69 print("Bisección función trig(x):")
```

```
70 bisec(trig, 4, 6, 1e-8, 100)
```

Salida

```
Bisección función pol(x):
i = 1 , p = 1.500000000000
i = 2 , p = 1.250000000000
i = 3 , p = 1.375000000000
i = 4 , p = 1.312500000000
i = 5 , p = 1.343750000000
i = 6 , p = 1.359375000000
i = 7 , p = 1.367187500000
i = 8 , p = 1.363281250000
i = 9 , p = 1.365234375000
i = 10, p = 1.364257812500
i = 11, p = 1.364746093750
i = 12, p = 1.364990234375
i = 13, p = 1.365112304688
i = 14, p = 1.365173339844
i = 15, p = 1.365203857422
i = 16, p = 1.365219116211
i = 17, p = 1.365226745605
i = 18, p = 1.365230560303
i = 19, p = 1.365228652954
i = 20, p = 1.365229606628
i = 21, p = 1.365230083466
i = 22, p = 1.365229845047
i = 23, p = 1.365229964256
i = 24, p = 1.365230023861
i = 25, p = 1.365229994059
i = 26, p = 1.365230008960
i = 27, p = 1.365230016410
Bisección función trig(x):
i = 1 , p = 5.000000000000
i = 2 , p = 5.500000000000
i = 3 , p = 5.750000000000
i = 4 , p = 5.625000000000
i = 5 , p = 5.562500000000
i = 6 , p = 5.593750000000
i = 7 , p = 5.609375000000
i = 8 , p = 5.601562500000
i = 9 , p = 5.597656250000
i = 10, p = 5.599609375000
i = 11, p = 5.598632812500
i = 12, p = 5.599121093750
i = 13, p = 5.599365234375
```

```
i = 14, p = 5.599243164062
i = 15, p = 5.599304199219
i = 16, p = 5.599334716797
i = 17, p = 5.599319458008
i = 18, p = 5.599311828613
i = 19, p = 5.599315643311
i = 20, p = 5.599313735962
i = 21, p = 5.599312782288
i = 22, p = 5.599313259125
i = 23, p = 5.599313020706
i = 24, p = 5.599313139915
i = 25, p = 5.599313080311
i = 26, p = 5.599313050508
i = 27, p = 5.599313035607
i = 28, p = 5.599313028157
```

B.1.2. *Regula falsi*

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación del método de regula falsi y algunos casos de salida.

from math import *


def pol(x):
    """Función de prueba"""
```

```
27      return x**3 + 4*x**2 - 10  # retorna pol(x) = x^3 + 4x^2 - 10
28
29
30  def trig(x):
31      """Función de prueba"""
32      return x*cos(x-1) - sin(x)  # retorna trig(x) = x cos(x - 1) - sin(x)
33
34
35  def pote(x):
36      """Función de prueba"""
37      return pow(7, x) - 13  # retorna pote(x) = 7^x - 13
38
39
40  def regula(f, p0, p1, tol, n):
41      """
42      Implementación método de regula falsi
43      Entradas:
44      f -- función
45      p0 -- aproximación inicial
46      p1 -- aproximación inicial
47      tol -- tolerancia
48      n -- número máximo de iteraciones
49
50      Salida:
51      p aproximación a cero de f
52      None en caso de iteraciones agotadas
53      """
54      i = 0
55      while i <= n:
56          q0 = f(p0)
57          q1 = f(p1)
58          p = p1-(q1*(p1 - p0))/(q1 - q0)
59          print("Iter = {0:<2}, p = {1:.12f}".format(i, p))
60          if abs(p - p1) < tol:
61              return p
62          i += 1
63          q = f(p)
64          if q*q1 < 0:
65              p0 = p1
66              q0 = q1
67          p1 = p
68          q1 = q
69      print("Iteraciones agotadas: Error!")
70      return None
71
```

```
72
73 # pol(x), a = 1.0, b = 2.0, TOL = 10^-8, N_0 = 100
74 print("Regula falsi función pol(x):")
75 regula(pol, 1, 2, 1e-8, 100)
76
77 # trig(x), a = 4.0, b = 6.0, TOL = 10^-8, N_0 = 100
78 print("Regula falsi función trig(x):")
79 regula(trig, 4, 6, 1e-8, 100)
80
81 # pote(x), a = 0, b = 2.0, TOL = 10^-8, N_0 = 100
82 print("Regula falsi función pote(x):")
83 regula(pote, 0, 2, 1e-8, 100)
```

Salida

```
Regula falsi función pol(x):
Iter = 0 , p = 1.263157894737
Iter = 1 , p = 1.338827838828
Iter = 2 , p = 1.358546341825
Iter = 3 , p = 1.363547440042
Iter = 4 , p = 1.364807031827
Iter = 5 , p = 1.365123717884
Iter = 6 , p = 1.365203303663
Iter = 7 , p = 1.365223301986
Iter = 8 , p = 1.365228327026
Iter = 9 , p = 1.365229589674
Iter = 10, p = 1.365229906941
Iter = 11, p = 1.365229986660
Iter = 12, p = 1.365230006692
Iter = 13, p = 1.365230011725
Regula falsi función trig(x):
Iter = 0 , p = 5.235657374722
Iter = 1 , p = 5.569477410510
Iter = 2 , p = 5.597623035312
Iter = 3 , p = 5.599220749873
Iter = 4 , p = 5.599307996036
Iter = 5 , p = 5.599312749633
Iter = 6 , p = 5.599313008600
Iter = 7 , p = 5.599313022708
Iter = 8 , p = 5.599313023477
Regula falsi función pote(x):
Iter = 0 , p = 0.500000000000
Iter = 1 , p = 0.835058241104
Iter = 2 , p = 1.045169783424
Iter = 3 , p = 1.168847080360
Iter = 4 , p = 1.238197008244
```

```
Iter = 5 , p = 1.275862101477
Iter = 6 , p = 1.295933755010
Iter = 7 , p = 1.306516642232
Iter = 8 , p = 1.312064439413
Iter = 9 , p = 1.314963818299
Iter = 10, p = 1.316476640694
Iter = 11, p = 1.317265325509
Iter = 12, p = 1.317676311539
Iter = 13, p = 1.317890428220
Iter = 14, p = 1.318001965936
Iter = 15, p = 1.318060064553
Iter = 16, p = 1.318090326416
Iter = 17, p = 1.318106088665
Iter = 18, p = 1.318114298545
Iter = 19, p = 1.318118574700
Iter = 20, p = 1.318120801950
Iter = 21, p = 1.318121962020
Iter = 22, p = 1.318122566245
Iter = 23, p = 1.318122880958
Iter = 24, p = 1.318123044876
Iter = 25, p = 1.318123130253
Iter = 26, p = 1.318123174722
Iter = 27, p = 1.318123197884
Iter = 28, p = 1.318123209948
Iter = 29, p = 1.318123216231
```

B.1.3. Newton

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
```

```
17  # along with this program.  If not, see <http://www.gnu.org/licenses/>
18  # ----------------------------------------------------------------------
19
20  # Implementación del método de Newton y algunos casos de salida.
21
22  from math import *
23
24
25  def expo(x):
26      """Función de prueba"""
27      return x**2 + exp(-2*x) - 2*x*exp(-x)
28      # retorna expo(x) = x^2 + e^{-2x} - 2xe^{-x}
29
30
31  def expoprima(x):
32      """Derivada función de prueba"""
33      return 2*x - 2*exp(-2*x) - 2*exp(-x) + 2*x*exp(-x)
34      # retorna expoprima(x) = d/dx expo(x)
35
36
37  def trig(x):
38      """Función de prueba"""
39      return cos(x) - x  # retorna trig(x) = cos(x) - x
40
41
42  def trigprima(x):
43      """Derivada función de prueba"""
44      return -sin(x) - 1  # retorna trigprima(x) = d/dx trig(x)
45
46
47  def newton(f, fprima, p0, tol, n):
48      """
49      Implementación método de Newton
50      Entradas:
51      f -- función
52      fprima -- derivada función f
53      p0 -- aproximación inicial
54      tol -- tolerancia
55      n -- número máximo de iteraciones
56
57      Salida:
58      p aproximación a cero de f
59      None en caso de iteraciones agotadas
```

```
    """
    i = 1
    while i <= n:
        p = p0 - f(p0)/fprima(p0)
        print("Iter = {0:<2}, p = {1:.12f}".format(i, p))
        if abs(p - p0) < tol:
            return p
        p0 = p
        i += 1
    print("Iteraciones agotadas: Error!")
    return None


# trig(x), trigprima(x), p0 = π/4, TOL = 10^-8, N0 = 100
print("Newton función trig(x):")
newton(trig, trigprima, pi/4, 1e-8, 100)

# expo(x), expoprima(x), p0 = 4.0, TOL = 10^-8, N0 = 100
print("Newton función expo(x):")
newton(expo, expoprima, 4, 1e-8, 100)
```

Salida

```
Newton función trig(x):
Iter = 1 , p = 0.739536133515
Iter = 2 , p = 0.739085178106
Iter = 3 , p = 0.739085133215
Iter = 4 , p = 0.739085133215
Newton función expo(x):
Iter = 1 , p = 2.044965524905
Iter = 2 , p = 1.196901548785
Iter = 3 , p = 0.853320871938
Iter = 4 , p = 0.703487910307
Iter = 5 , p = 0.633704735236
Iter = 6 , p = 0.600031374819
Iter = 7 , p = 0.583490458626
Iter = 8 , p = 0.575292817860
Iter = 9 , p = 0.571212060259
Iter = 10, p = 0.569176179405
Iter = 11, p = 0.568159361242
Iter = 12, p = 0.567651232450
Iter = 13, p = 0.567397238091
Iter = 14, p = 0.567270258416
Iter = 15, p = 0.567206772954
Iter = 16, p = 0.567175031318
Iter = 17, p = 0.567159160772
```

```
Iter = 18, p = 0.567151225569
Iter = 19, p = 0.567147257985
Iter = 20, p = 0.567145274200
Iter = 21, p = 0.567144282303
Iter = 22, p = 0.567143786354
Iter = 23, p = 0.567143538379
Iter = 24, p = 0.567143414504
Iter = 25, p = 0.567143352393
Iter = 26, p = 0.567143321397
Iter = 27, p = 0.567143305350
Iter = 28, p = 0.567143297786
```

B.1.4. Secante

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación del método de la secante y algunos casos de salida.

from math import *


def trig(x):
    """Función de prueba"""
    return sin(2/x)  # retorna trig(x) = sin(2/x)


def pol(x):
```

```
31      """Función de prueba"""
32      return x**3 - 2  # retorna pol(x) = x^3 - 2
33
34
35  def secante(f, p0, p1, tol, n):
36      """
37      Implementación método de la secante
38      Entradas:
39      f -- función
40      p0 -- aproximación inicial
41      p1 -- aproximación inicial
42      tol -- tolerancia
43      n -- número máximo de iteraciones
44
45      Salida:
46      p aproximación a cero de f
47      None en caso de iteraciones agotadas
48      """
49      i = 2
50      while i <= n:
51          p = p1 - (f(p1)*(p1 - p0))/(f(p1) - f(p0))
52          print("Iter = {0:<2}, p = {1:.12f}".format(i, p))
53          if abs(p - p1) < tol:
54              return p
55          p0 = p1
56          p1 = p
57          i += 1
58      print("Iteraciones agotadas: Error!")
59      return None
60
61
62  # pol(x), p0 = -3.0, p1 = 3.0, TOL = 10^-8, N0 = 100
63  print("Secante función pol(x):")
64  secante(pol, -3, 3, 1e-8, 100)
65
66  # trig(x), p0 = 1.1, p1 = 0.8, TOL = 10^-8, N0 = 100
67  print("Secante función trig(x):")
68  secante(trig, 1.1, 0.8, 1e-8, 100)
```

Salida

```
Secante función pol(x):
Iter = 2 , p = 0.222222222222
Iter = 3 , p = 0.426937738247
Iter = 4 , p = 6.313558779887
Iter = 5 , p = 0.471912789303
```

```
Iter = 6 , p = 0.515915680782
Iter = 7 , p = 3.059385436425
Iter = 8 , p = 0.682161118454
Iter = 9 , p = 0.823408229082
Iter = 10, p = 1.668975857749
Iter = 11, p = 1.121425751966
Iter = 12, p = 1.221126314305
Iter = 13, p = 1.264621145111
Iter = 14, p = 1.259773686642
Iter = 15, p = 1.259920501483
Iter = 16, p = 1.259921049959
Iter = 17, p = 1.259921049895
Secante función trig(x):
Iter = 2 , p = 0.316169553146
Iter = 3 , p = 0.279163965987
Iter = 4 , p = 0.318328356367
Iter = 5 , p = 0.318309856297
Iter = 6 , p = 0.318309886186
Iter = 7 , p = 0.318309886184
```

B.1.5. Punto fijo

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ---------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ---------------------------------------------------------------------

# Implementación del método del punto fijo y algunos casos de salida.

from math import *
```

```
23
24
25 def pote(x):
26     """Función de prueba"""
27     return pow(2, -x)  # retorna pote(x) = 2^{-x}
28
29
30 def pol(x):
31     """Función de prueba"""
32     return (x**2 - 1)/3  # retorna pol(x) = (x^2-1)/3
33
34
35 def puntofijo(f, p0, tol, n):
36     """
37     Implementación método de punto fijo
38     Entradas:
39     f -- función
40     p0 -- aproximación inicial
41     tol -- tolerancia
42     n -- número máximo de iteraciones
43
44     Salida:
45     p aproximación a punto fijo de f
46     None en caso de iteraciones agotadas
47     """
48     i = 1
49     while i <= n:
50         p = f(p0)
51         print("Iter = {0:<2}, p = {1:.12f}".format(i, p))
52         if abs(p - p0) < tol:
53             return p
54         p0 = p
55         i += 1
56     print("Iteraciones agotadas: Error!")
57     return None
58
59
60 # pol(x), p_0 = 0.9, TOL = 10^{-8}, N_0 = 100
61 print("Punto fijo función pol(x):")
62 puntofijo(pol, 0.9, 1e-8, 100)
63
64 # pote(x), p_0 = 0.5, TOL = 10^{-8}, N_0 = 100
65 print("Punto fijo función pote(x):")
66 puntofijo(pote, 0.5, 1e-8, 100)
```

Salida

```
Punto fijo función pol(x):
Iter = 1 , p = -0.063333333333
Iter = 2 , p = -0.331996296296
Iter = 3 , p = -0.296592819749
Iter = 4 , p = -0.304010899758
Iter = 5 , p = -0.302525790943
Iter = 6 , p = -0.302826048605
Iter = 7 , p = -0.302765461429
Iter = 8 , p = -0.302777691789
Iter = 9 , p = -0.302775223118
Iter = 10, p = -0.302775721422
Iter = 11, p = -0.302775620839
Iter = 12, p = -0.302775641142
Iter = 13, p = -0.302775637044
Punto fijo función pote(x):
Iter = 1 , p = 0.707106781187
Iter = 2 , p = 0.612547326536
Iter = 3 , p = 0.654040860042
Iter = 4 , p = 0.635497845813
Iter = 5 , p = 0.643718641723
Iter = 6 , p = 0.640061021177
Iter = 7 , p = 0.641685807043
Iter = 8 , p = 0.640963537178
Iter = 9 , p = 0.641284509067
Iter = 10, p = 0.641141851472
Iter = 11, p = 0.641205252450
Iter = 12, p = 0.641177074529
Iter = 13, p = 0.641189597767
Iter = 14, p = 0.641184031979
Iter = 15, p = 0.641186505614
Iter = 16, p = 0.641185406241
Iter = 17, p = 0.641185894842
Iter = 18, p = 0.641185677690
Iter = 19, p = 0.641185774200
Iter = 20, p = 0.641185731307
Iter = 21, p = 0.641185750370
Iter = 22, p = 0.641185741898
```

B.2. Interpolación

B.2.1. Lagrange

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ------------------------------------------------------------------------

# Implementación del interpolador de Lagrange y algunos casos de salida.

from math import *


def LagrangePol(datos):
    """
    Implementación del interpolador de Lagrange
    Entradas:
    datos -- lista de puntos (x, y) en el plano

    Salida:
    P -- función de interpolación
    """

    def L(k, x):
        """Implementación funciones L_k(x)"""
        # pol L_k(x) = \prod_{i≠k} (x-x_i)/(x_k-x_i)
        out = 1
        for i, p in enumerate(datos):
            if i != k:
```

```
                out *= (x - p[0])/(datos[k][0] - p[0])
        return out

    def P(x):
        """Implementación polinomio P(x)"""
        # polinomio P(x) = \sum_k f(x_k)L_k(x)
        lag = 0
        for k, p in enumerate(datos):
            lag += p[1]*L(k, x)
        return lag

    return P


# datos para f(x) = 1/x con x_0 = 2, x_1 = 2.75 y x_2 = 4
datosf = [(2, 1/2), (11/4, 4/11), (4, 1/4)]
Pf = LagrangePol(datosf)
print("Polinomio de Lagrange en x = 3:")
print("{0:.12f}".format(Pf(3)))

# datos g(x) = sin(3x), x_0 = 1, x_1 = 1.3, x_2 = 1.6, x_3 = 1.9 y x_4 = 2.2
datosg = [(1, 0.1411), (1.3, -0.6878), (1.6, -0.9962),
          (1.9, -0.5507), (2.2, 0.3115)]
Pg = LagrangePol(datosg)
print("Polinomio de Lagrange en x = 1.5:")
print("{0:.12f}".format(Pg(1.5)))
```

Salida

```
Polinomio de Lagrange en x = 3:
0.329545454545
Polinomio de Lagrange en x = 1.5:
-0.977381481481
```

B.2.2. Diferencias divididas de Newton

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
```

```
9  # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation, either version 3 of the License, or
11 # (at your option) any later version.
12 # This program is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 # GNU General Public License for more details.
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>
18 # ----------------------------------------------------------------------
19
20 # Implementación del interpolador de Newton y algunos casos de salida.
21
22 from math import *
23 from pprint import pprint
24
25
26 def NewtonPol(dat):
27     """
28     Implementación del interpolador de Newton
29     Entradas:
30     dat -- lista de puntos (x, y) en el plano
31
32     Salidas:
33     F -- tabla de diferencias divididas
34     P -- función de interpolación
35     """
36     n = len(dat)
37     F = [[0 for x in dat] for x in dat]  # crear tabla nula
38
39     for i, p in enumerate(dat):  # condiciones iniciales
40         F[i][0] = p[1]
41
42     for i in range(1, n):  # tabla de diferencias divididas
43         for j in range(1, i+1):
44             F[i][j] = (F[i][j-1]-F[i-1][j-1])/(dat[i][0]-dat[i-j][0])
45
46     def L(k, x):
47         """Implementación funciones L_k(x)"""
48         # polinomio L_k(x) = \prod_{i<=k} (x - x_i)
49         out = 1
50         for i, p in enumerate(dat):
51             if i <= k:
52                 out *= (x - p[0])
```

```
53          return out
54
55      def P(x):
56          """Implementación polinomio P(x)"""
57          # P(x) = f[x_0] + sum_{k=1}^{n} f[x_0, x_1, ..., x_k] L_{k-1}(x)
58          newt = 0
59          for i in range(1, n):
60              newt += F[i][i]*L(i-1, x)
61          return newt + F[0][0]
62
63      return F, P
64
65
66  datost = [(-1, 3), (0, -4), (1, 5), (2, -6)]
67  T, P = NewtonPol(datost)
68  print("Tabla de diferencias divididas:")
69  pprint(T)
70  print("Evaluar el polinomio en x = 0:")
71  print(P(0))
72
73  datosf = [(2, 1/2), (11/4, 4/11), (4, 1/4)]
74  T, P = NewtonPol(datosf)
75  print("Tabla de diferencias divididas:")
76  pprint(T)
77  print("Evaluar el polinomio en x = 3:")
78  print(P(3))
```

Salida

```
Tabla de diferencias divididas:
[[3, 0, 0, 0], [-4, -7.0, 0, 0], [5, 9.0, 8.0, 0], [-6, -11.0, -10.0, -6.0]]
Evaluar el polinomio en x = 0:
-4.0
Tabla de diferencias divididas:
[[0.5, 0, 0],
 [0.36363636363636365, -0.1818181818181818, 0],
 [0.25, -0.09090909090909091, 0.04545454545454544]]
Evaluar el polinomio en x = 3:
0.3295454545454546
```

B.2.3. Trazadores cúbicos

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
```

```
4  # ------------------------------------------------------------------------
5  # Compendio de programas.
6  # Matemáticas para Ingeniería. Métodos numéricos con Python.
7  # Copyright (C) 2020 Los autores del texto.
8  # This program is free software: you can redistribute it and/or modify
9  # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation, either version 3 of the License, or
11 # (at your option) any later version.
12 # This program is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 # GNU General Public License for more details.
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>
18 # ------------------------------------------------------------------------
19
20 # Trazadores cúbicos naturales y algunos casos de salida.
21
22 from math import *
23
24
25 def CubicSplines(datos):
26     """
27     Implementación trazadores cúbicos
28     Entradas:
29     datos -- lista de puntos (x, y) en el plano ordenados por x
30
31     Salidas:
32     a -- vector de coeficientes (constantes)
33     b -- vector de coeficientes (lineales)
34     c -- vector de coeficientes (cuadráticos)
35     d -- vector de coeficientes (cúbicos)
36     """
37     n = len(datos)-1
38     # Inicializar vectores auxiliares
39     A = [x[1] for x in datos]
40     X = [x[0] for x in datos]
41     H = [0.0 for x in range(n)]
42     B = [0.0 for x in range(n+1)]
43     C = [0.0 for x in range(n+1)]
44     D = [0.0 for x in range(n+1)]
45     alpha = [0.0 for x in range(n)]
46     mu = [0.0 for x in range(n+1)]
47     lo = [1.0 for x in range(n+1)]
48     z = [0.0 for x in range(n+1)]
```

```
49
50      # Crear vector H
51      for i in range(n):
52          H[i] = X[i+1]-X[i]
53
54      # Crear vector α
55      for i in range(1, n):
56          alpha[i] = (3/H[i])*(A[i+1]-A[i])-(3/H[i-1])*(A[i]-A[i-1])
57
58      # Solucionar sistema tridiagonal
59      for i in range(1, n):
60          lo[i] = 2*(X[i+1]-X[i-1])-H[i-1]*mu[i-1]
61          mu[i] = H[i]/lo[i]
62          z[i] = (alpha[i]-H[i-1]*z[i-1])/lo[i]
63
64      # Solucionar sistema tridiagonal
65      for j in range(n-1, -1, -1):
66          C[j] = z[j]-mu[j]*C[j+1]
67          B[j] = (A[j+1]-A[j])/(H[j])-H[j]*(C[j+1]+2*C[j])/3
68          D[j] = (C[j+1]-C[j])/(3*H[j])
69
70      # Retornar vectores A, B, C, D
71      return A[:-1], B[:-1], C[:-1], D[:-1]
72
73
74  # Datos de prueba (1,2), (2,3), (3,5)
75  datosPrueba = [(1, 2), (2, 3), (3, 5)]
76  a, b, c, d = CubicSplines(datosPrueba)
77  print("Vectores de coeficientes:")
78  print("A =", a)
79  print("B =", b)
80  print("C =", c)
81  print("D =", d)
82
83  # Datos de prueba (0,1), (1,e), (2,e^2) y (3,e^3)
84  datosPrueba = [(0, exp(0)), (1, exp(1)),
85                 (2, exp(2)), (3, exp(3))]
86  a, b, c, d = CubicSplines(datosPrueba)
87  print("Vectores de coeficientes:")
88  print("A =", a)
89  print("B =", b)
90  print("C =", c)
91  print("D =", d)
```

Salida

```
Vectores de coeficientes:
A = [2, 3]
B = [0.75, 1.5]
C = [0.0, 0.75]
D = [0.25, -0.25]
Vectores de coeficientes:
A = [1.0, 2.718281828459045, 7.38905609893065]
B = [1.465997614174723, 2.222850257027689, 8.809769654506473]
C = [0.0, 0.756852642852966, 5.830066754625817]
D = [0.252284214284322, 1.6910713705909506, -1.9433555848752724]
```

B.2.4. Recta de ajuste mínimos cuadrados

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ------------------------------------------------------------------------

# Implementación de la recta de mínimos cuadrados.

from math import *


def RectaMinSq(datos):
    """
    Implementación recta de mínimos cuadrados
    Entradas:
    datos -- lista de puntos (x, y) en el plano

```

```
31      Salida:
32      P -- recta de mínimos cuadrados
33      """
34      X = sum([p[0] for p in datos])
35      Y = sum([p[1] for p in datos])
36      XX = sum([(p[0])**2 for p in datos])
37      XY = sum([p[0]*p[1] for p in datos])
38      m = len(datos)
39
40      def P(x):
41          """Recta de mínimos cuadrados"""
42          a0 = (Y*XX - X*XY)/(m*XX - X**2)
43          a1 = (m*XY - X*Y)/(m*XX - X**2)
44          return a0 + a1*x
45
46      return P
47
48
49  def ErrorSq(f, datos):
50      """Calcular error cuadrático"""
51      E = sum([(p[1] - f(p[0]))**2 for p in datos])
52      return E
53
54
55  # datos de prueba
56  datos = [(-1, 2), (0, -1), (1, 1), (2, -2)]
57  f = RectaMinSq(datos)
58  print("Recta de ajuste. Evaluar en x = 1:")
59  print("{0:.10f}".format(f(1.0)))
60
61  # datos de prueba
62  datos = [(1.0, 1.3), (2.0, 3.5), (3.0, 4.2), (4.0, 5.0), (5.0, 7.0),
63           (6.0, 8.8), (7.0, 10.1), (8.0, 12.5), (9.0, 13.0)]
64  f = RectaMinSq(datos)
65  print("Recta de ajuste. Evaluar en x = 1:")
66  print("{0:.10f}".format(f(1.0)))
```

Salida

```
Recta de ajuste. Evaluar en x = 1:
-0.5000000000
Recta de ajuste. Evaluar en x = 1:
1.3066666667
```

B.3. Diferenciación e integración numérica

B.3.1. Extrapolación de Richardson

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación extrapolación de Richardson y algunos casos de salida.

from math import *


def pol(x):
    """Función de prueba"""
    return x**3 + 4*x**2 - 10  # retorna pol(x) = x^3 + 4x^2 - 10


def trig(x):
    """Función de prueba"""
    return x*cos(x-1) - sin(x)  # retorna trig(x) = x cos(x - 1) - sin(x)


def dercentrada(f, x, h):
    """Retorna diferencias centradas"""
    return (f(x + h) - f(x - h))/(2*h)


def richardson(f, x, h):
```

```
    """
    Implementación extrapolación de Richardson
    Entradas:
    f -- función
    x -- punto
    h -- paso

    Salida:
    d -- aproximación a la derivada
    """
    d = (4/3)*dercentrada(f, x, h/2)-(1/3)*dercentrada(f, x, h)
    return d


# pol(x), x = 1.5, h = 0.1
print("Derivada función pol(x):")
print("{0:.12f}".format(richardson(pol, 1.5, 0.1)))

# trig(x), x = 4, h = 0.2
print("Derivada función trig(x):")
print("{0:.12f}".format(richardson(trig, 4, 0.2)))
```

Salida

```
Derivada función pol(x):
18.750000000000
Derivada función trig(x):
-0.900812732439
```

B.3.2. Regla del trapecio

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
15 # GNU General Public License for more details.
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>
18 # ----------------------------------------------------------------------
19
20 # Implementación de la regla del trapecio y algunos casos de salida.
21
22 from math import *
23
24
25 def pol(x):
26     """Función de prueba"""
27     return x**3 + 4*x**2 - 10  # retorna pol(x) = x^3 + 4x^2 - 10
28
29
30 def trig(x):
31     """Función de prueba"""
32     return x*cos(x-1) - sin(x)  # retorna trig(x) = x cos(x - 1) - sin(x)
33
34
35 def trapecio(f, a, b, n):
36     """
37     Implementación regla del trapecio
38     Entradas:
39     f -- función
40     a -- inicio intervalo
41     b -- fin intervalo
42     n -- número de pasos
43
44     Salida:
45     abc -- aproximación área bajo la curva
46     """
47     h = (b - a)/n
48     acum = 0
49     for j in range(1, n):
50         acum += 2*f(a + h*j)
51     abc = (h/2)*(f(a) + acum + f(b))
52     return abc
53
54
55 # pol(x), a = 1, b = 2, N = 10
56 print("\nÁrea bajo la curva pol(x):\n")
57 print("{0:.12f}".format(trapecio(pol, 1, 2, 10)))
58
59 # trig(x), a = 4, b = 6, N = 20
```

```
60 print("\nÁrea bajo la curva trig(x):\n")
61 print("{0:.12f}".format(trapecio(trig, 4, 6, 20)))
```

Salida

```
Área bajo la curva pol(x):

3.097500000000

Área bajo la curva trig(x):

-3.425574350843
```

B.3.3. Regla de Simpson

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # ------------------------------------------------------------------------
5  # Compendio de programas.
6  # Matemáticas para Ingeniería. Métodos numéricos con Python.
7  # Copyright (C) 2020 Los autores del texto.
8  # This program is free software: you can redistribute it and/or modify
9  # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation, either version 3 of the License, or
11 # (at your option) any later version.
12 # This program is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 # GNU General Public License for more details.
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>
18 # ------------------------------------------------------------------------
19
20 # Implementación de la regla de Simpson y algunos casos de salida.
21
22 from math import *
23
24
25 def pol(x):
26     """Función de prueba"""
27     return x**3 + 4*x**2 - 10  # retorna pol(x) = x^3 + 4x^2 - 10
28
29
```

```
30 def trig(x):
31     """Función de prueba"""
32     return x*cos(x-1) - sin(x)  # retorna trig(x) = x cos(x − 1) − sin(x)
33
34
35 def simpson(f, a, b, n):
36     """
37     Implementación regla de Simpson
38     Entradas:
39     f -- función
40     a -- inicio intervalo
41     b -- fin intervalo
42     n -- número de pasos (par)
43
44     Salida:
45     abc -- aproximación área bajo la curva
46     """
47     h = (b - a)/n
48     oddsum = 0
49     evensum = 0
50     for j in range(1, n):
51         x = a + h*j
52         if j % 2 == 0:
53             evensum += 2*f(x)
54         else:
55             oddsum += 4*f(x)
56     abc = (h/3)*(f(a) + evensum + oddsum + f(b))
57     return abc
58
59
60 # pol(x), a = 1, b = 2, N = 10
61 print("Área bajo la curva pol(x):")
62 print("{0:.12f}".format(simpson(pol, 1, 2, 10)))
63
64 # trig(x), a = 4, b = 6, N = 20
65 print("Área bajo la curva trig(x):")
66 print("{0:.12f}".format(simpson(trig, 4, 6, 20)))
```

Salida

```
Área bajo la curva pol(x):
3.083333333333
Área bajo la curva trig(x):
-3.430561834182
```

B.4. Sistemas de ecuaciones

B.4.1. *LU*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación del método LU y algunos casos de salida.

from math import *
from pprint import pprint


def lu(A):
    """
    Implementación del método LU
    Entradas:
    A -- matriz cuadrada

    Salidas:
    L, U -- matrices de la descomposición
    None -- en caso de no ser posible la descomposición
    """
    n = len(A)
    # crear matrices nulas
    L = [[0 for x in range(n)] for x in range(n)]
    U = [[0 for x in range(n)] for x in range(n)]

```

```
41      # Doolittle
42      L[0][0] = 1
43      U[0][0] = A[0][0]
44
45      if abs(L[0][0]*U[0][0]) <= 1e-15:
46          print("Imposible descomponer")
47          return None
48
49      for j in range(1, n):
50          U[0][j] = A[0][j]/L[0][0]
51          L[j][0] = A[j][0]/U[0][0]
52
53      for i in range(1, n-1):
54          L[i][i] = 1
55          s = sum([L[i][k]*U[k][i] for k in range(i)])
56          U[i][i] = A[i][i] - s
57
58          if abs(L[i][i]*U[i][i]) <= 1e-15:
59              print("Imposible descomponer")
60              return None
61
62          for j in range(i+1, n):
63              s1 = sum([L[i][k]*U[k][j] for k in range(i)])
64              s2 = sum([L[j][k]*U[k][i] for k in range(i)])
65              U[i][j] = A[i][j] - s1
66              L[j][i] = (A[j][i] - s2)/U[i][i]
67
68      L[n-1][n-1] = 1.0
69      s3 = sum([L[n-1][k]*U[k][n-1] for k in range(n)])
70      U[n-1][n-1] = A[n-1][n-1] - s3
71
72      if abs(L[n-1][n-1]*U[n-1][n-1]) <= 1e-15:
73          print("Imposible descomponer")
74          return None
75
76      print("Matriz L:")
77      pprint(L)
78      print("Matriz U:")
79      pprint(U)
80      return L, U
81
82
83  A = [[4, 3], [6, 3]]
84  print("Matriz A:")
85  pprint(A)
```

```
lu(A)

A = [[0, 1], [1, 1]]
print("Matriz A:")
pprint(A)
lu(A)

A = [[3, 1, 6], [-6, 0, -16], [0, 8, -17]]
print("Matriz A:")
pprint(A)
lu(A)

A = [[1, 2, 3], [2, 4, 5], [1, 3, 4]]
print("Matriz A:")
pprint(A)
lu(A)
```

Salida

```
Matriz A:
[[4, 3], [6, 3]]
Matriz L:
[[1, 0], [1.5, 1.0]]
Matriz U:
[[4, 3.0], [0, -1.5]]
Matriz A:
[[0, 1], [1, 1]]
Imposible descomponer
Matriz A:
[[3, 1, 6], [-6, 0, -16], [0, 8, -17]]
Matriz L:
[[1, 0, 0], [-2.0, 1, 0], [0.0, 4.0, 1.0]]
Matriz U:
[[3, 1.0, 6.0], [0, 2.0, -4.0], [0, 0, -1.0]]
Matriz A:
[[1, 2, 3], [2, 4, 5], [1, 3, 4]]
Imposible descomponer
```

B.4.2. Jacobi

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
```

```
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ------------------------------------------------------------------------

# Implementación del método de Jacobi y algunos casos de salida.

from math import *
from pprint import pprint


def distinf(x, y):
    """Implementación distancia dada por la norma infinito"""
    return max([abs(x[i] - y[i]) for i in range(len(x))])


def Jacobi(A, b, x0, TOL, MAX):
    """
    Implementación del método de Jacobi
    Entradas:
    A -- matriz cuadrada
    b -- vector
    x0 -- aproximación inicial
    TOL -- tolerancia
    MAX -- número máximo de iteraciones

    Salida:
    x -- aproximación a solución del sistema Ax = b
    None -- en caso de agotar las iteraciones o presentar errores
    """
    n = len(A)
    x = [0.0 for x in range(n)]
    k = 1
    while k <= MAX:
        for i in range(n):
            if abs(A[i][i]) <= 1e-15:
```

```
                print("Imposible iterar")
                return None
            s = sum([A[i][j]*x0[j] for j in range(n) if j != i])
            x[i] = (b[i] - s)/A[i][i]
        pprint(x)
        if distinf(x, x0) < TOL:
            print(r"Solución encontrada")
            return x
        k += 1
        for i in range(n):
            x0[i] = x[i]
    print("Iteraciones agotadas")
    return None


A = [[2, 1], [5, 7]]
b = [11, 13]
x0 = [1, 1]
print("Matriz A:")
pprint(A)
print("Vector b:")
pprint(b)
print("Semilla x0:")
pprint(x0)
print("Iteración de Jacobi")
# TOL = 10^-5, MAX = 50
Jacobi(A, b, x0, 1e-5, 50)


A = [[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
b = [6, 25, -11]
x0 = [0, 0, 0]
print("Matriz A:")
pprint(A)
print("Vector b:")
pprint(b)
print("Semilla x0:")
pprint(x0)
print("Iteración de Jacobi")
# TOL = 10^-10, MAX = 50
Jacobi(A, b, x0, 1e-10, 50)
```

Salida

```
Matriz A:
[[2, 1], [5, 7]]
```

```
Vector b:
[11, 13]
Semilla x0:
[1, 1]
Iteración de Jacobi
[5.0, 1.1428571428571428]
[4.928571428571429, -1.7142857142857142]
[6.357142857142857, -1.6632653061224494]
[6.331632653061225, -2.683673469387755]
[6.841836734693878, -2.6654518950437316]
[6.832725947521865, -3.0298833819241984]
[7.014941690962099, -3.0233756768013325]
[7.0116878384006665, -3.1535297792586428]
[7.076764889629321, -3.151205598857619]
[7.075602799428809, -3.197689206878086]
[7.098844603439043, -3.1968591424491493]
[7.098429571224575, -3.213460431027888]
[7.106730215513944, -3.2131639794461244]
[7.106581989723062, -3.2190930110813887]
[7.109546505540695, -3.2189871355164734]
[7.1094935677582365, -3.221104646814782]
[7.110552323407391, -3.2210668341130266]
[7.110533417056513, -3.221823088148137]
[7.110911544074068, -3.221809583611795]
[7.110904791805897, -3.22207967433862]
[7.11103983716931, -3.2220748512899258]
[7.111037425644962, -3.2221713122637925]
[7.1110856561318965, -3.2221695897464024]
[7.111084794873201, -3.222204040094212]
[7.111102020047106, -3.22220342490943]
[7.111101712454715, -3.2222157286050757]
[7.111107864302538, -3.2222155088962245]
Solución encontrada
Matriz A:
[[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
Vector b:
[6, 25, -11]
Semilla x0:
[0, 0, 0]
Iteración de Jacobi
[0.6, 2.272727272727273, -1.1]
[1.0472727272727274, 2.227272727272727, -0.9927272727272728]
[1.0212727272727273, 2.277685950413223, -1.0867272727272728]
[1.0451140495867768, 2.266776859504132, -1.0764859504132231]
[1.0419748760330578, 2.2698752817430505, -1.0823451239669422]
```

```
[1.0434565529676934, 2.2690572501878283, -1.0814074470323065]
[1.043187214425244, 2.2692771914486713, -1.0817855855747558]
[1.0432848362598182, 2.269218329895499, -1.0817097237401818]
[1.0432637777375864, 2.269234101138149, -1.0817351342624137]
[1.0432704369662977, 2.269229876679561, -1.0817293454337025]
[1.0432688567546966, 2.269231008321145, -1.0817310997253036]
[1.0432693207771753, 2.26923070518449, -1.0817306705188248]
[1.043269204622214, 2.269230786387123, -1.0817307936369862]
[1.0432692373661094, 2.2692307646350205, -1.0817307622857304]
[1.0432692289206482, 2.2692307704618524, -1.08173077100972]
[1.043269231248129, 2.2692307689009934, -1.0817307687379443]
[1.0432692306376883, 2.269230769319108, -1.0817307693595264]
[1.043269230803816, 2.269230769207106, -1.08173076919562 7]
[1.0432692307598361, 2.269230769237108, -1.0817307692400526]
Solución encontrada
```

B.4.3. Gauss-Seidel

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación del método de Gauss-Seidel y algunos casos de salida.

from math import *
from pprint import pprint


def distinf(x, y):
```

```
27      """Implementación distancia dada por la norma infinito"""
28      return max([abs(x[i]-y[i]) for i in range(len(x))])
29
30
31  def GaussSeidel(A, b, x0, TOL, MAX):
32      """
33      Implementación del método de Gauss-Seidel
34      Entradas:
35      A -- matriz cuadrada
36      b -- vector
37      x0 -- aproximación inicial
38      TOL -- tolerancia
39      MAX -- número máximo de iteraciones
40
41      Salida:
42      x -- aproximación a solución del sistema Ax = b
43      None -- en caso de agotar las iteraciones o presentar errores
44      """
45      n = len(A)
46      x = [0.0 for x in range(n)]
47      k = 1
48      while k <= MAX:
49          for i in range(n):
50              if abs(A[i][i]) <= 1e-15:
51                  print("Imposible iterar")
52                  return None
53              s1 = sum([A[i][j]*x[j] for j in range(i)])
54              s2 = sum([A[i][j]*x0[j] for j in range(i+1, n)])
55              x[i] = (b[i] - s1 - s2)/A[i][i]
56          pprint(x)
57          if distinf(x, x0) < TOL:
58              print(r"Solución encontrada")
59              return x
60          k += 1
61          for i in range(n):
62              x0[i] = x[i]
63      print("Iteraciones agotadas")
64      return None
65
66
67  A = [[2, 1], [5, 7]]
68  b = [11, 13]
69  x0 = [1, 1]
70  print("Matriz A:")
71  pprint(A)
```

```
print("Vector b:")
pprint(b)
print("Semilla x0:")
pprint(x0)
print("Iteración de Gauss-Seidel")
# TOL = 10^-5, MAX = 50
GaussSeidel(A, b, x0, 1e-5, 50)


A = [[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
b = [6, 25, -11]
x0 = [0, 0, 0]
print("Matriz A:")
pprint(A)
print("Vector b:")
pprint(b)
print("Semilla x0:")
pprint(x0)
print("Iteración de Gauss-Seidel")
# TOL = 10^-10, MAX = 50
GaussSeidel(A, b, x0, 1e-10, 50)
```

Salida

```
Matriz A:
[[2, 1], [5, 7]]
Vector b:
[11, 13]
Semilla x0:
[1, 1]
Iteración de Gauss-Seidel
[5.0, -1.7142857142857142]
[6.357142857142857, -2.683673469387755]
[6.841836734693878, -3.0298833819241984]
[7.014941690962099, -3.1535297792586428]
[7.076764889629321, -3.197689206878086]
[7.098844603439043, -3.213460431027888]
[7.106730215513944, -3.2190930110813887]
[7.109546505540695, -3.221104646814782]
[7.110552323407391, -3.221823088148137]
[7.110911544074068, -3.22207967433862]
[7.11103983716931, -3.2221713122637925]
[7.1110856561318965, -3.222204040094212]
[7.111102020047106, -3.2222157286050757]
[7.111107864302538, -3.2222199030732415]
Solución encontrada
```

```
Matriz A:
[[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
Vector b:
[6, 25, -11]
Semilla x0:
[0, 0, 0]
Iteración de Gauss-Seidel
[0.6, 2.3272727272727276, -0.9872727272727273]
[1.0301818181818183, 2.276628099173554, -1.0783735537190082]
[1.043337520661157, 2.2695421788129226, -1.081713286250939]
[1.0432968751314802, 2.2692348717164132, -1.0817358878546546]
[1.0432706647425722, 2.269230434262538, -1.0817310895222607]
[1.043269261330706, 2.269230742891677, -1.0817307779769734]
[1.0432692298845623, 2.2692307683552353, -1.0817307691413889]
[1.0432692306638014, 2.2692307692293103, -1.0817307692098292]
[1.043269230764897, 2.2692307692322786, -1.0817307692297515]
[1.043269230769178, 2.269230769230857, -1.0817307692307498]
Solución encontrada
```

B.5. Ecuaciones diferenciales

B.5.1. Euler

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ----------------------------------------------------------------------

# Implementación del método de Euler y algunos casos de salida.
```

```
21
22 from math import *
23
24
25 def test1(t, y):
26     """Función de prueba"""
27     return y - t**2 + 1   # y' = y - t^2 + 1
28
29
30 def test2(t, y):
31     """Función de prueba"""
32     return 2 - exp(-4*t) - 2*y   # y' = 2 - e^{-4t} - 2y
33
34
35 def Euler(a, b, y0, f, N):
36     """
37     Implementación método de Euler
38     Entradas:
39     a -- inicio intervalo
40     b -- fin intervalo
41     y0 -- aproximación inicial
42     f -- función
43     N -- pasos
44
45     Salida:
46     w -- aproximación final
47     """
48     h = (b - a)/N
49     t = a
50     w = y0
51     print("t0  = {0:.2f}, w0  = {1:.12f}".format(t, w))
52     for i in range(1, N+1):
53         w = w + h*f(t, w)
54         t = a + i*h
55         print("t{0:<2} = {1:.2f}, w{0:<2} = {2:.12f}".format(i, t, w))
56     return w
57
58
59 # dy/dt = y - t^2 + 1, a = 0, b = 2, y0 = 0.5, N = 10
60 print("Método de Euler:")
61 Euler(0, 2, 0.5, test1, 10)
62
63 # dy/dt = 2 - e^{-4t} - 2y, a = 0, b = 1, y0 = 1, N = 20
64 print("Método de Euler:")
65 Euler(0, 1, 1, test2, 20)
```

Salida

```
Método de Euler:
t0  = 0.00, w0  = 0.500000000000
t1  = 0.20, w1  = 0.800000000000
t2  = 0.40, w2  = 1.152000000000
t3  = 0.60, w3  = 1.550400000000
t4  = 0.80, w4  = 1.988480000000
t5  = 1.00, w5  = 2.458176000000
t6  = 1.20, w6  = 2.949811200000
t7  = 1.40, w7  = 3.451773440000
t8  = 1.60, w8  = 3.950128128000
t9  = 1.80, w9  = 4.428153753600
t10 = 2.00, w10 = 4.865784504320
Método de Euler:
t0  = 0.00, w0  = 1.000000000000
t1  = 0.05, w1  = 0.950000000000
t2  = 0.10, w2  = 0.914063462346
t3  = 0.15, w3  = 0.889141113810
t4  = 0.20, w4  = 0.872786420624
t5  = 0.25, w5  = 0.863041330356
t6  = 0.30, w6  = 0.858343225262
t7  = 0.35, w7  = 0.857449192140
t8  = 0.40, w8  = 0.859374424729
t9  = 0.45, w9  = 0.863342156356
t10 = 0.50, w10 = 0.868742996309
t11 = 0.55, w11 = 0.875101932517
t12 = 0.60, w12 = 0.882051581347
t13 = 0.65, w13 = 0.889310525548
t14 = 0.70, w14 = 0.896665794082
t15 = 0.75, w15 = 0.903958711543
t16 = 0.80, w16 = 0.911073486970
t17 = 0.85, w17 = 0.917928028074
t18 = 0.90, w18 = 0.924466561769
t19 = 0.95, w19 = 0.930653719470
t20 = 1.00, w20 = 0.936469808930
```

B.5.2. Verlet

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
```

```
7  # Copyright (C) 2020 Los autores del texto.
8  # This program is free software: you can redistribute it and/or modify
9  # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation, either version 3 of the License, or
11 # (at your option) any later version.
12 # This program is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 # GNU General Public License for more details.
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>
18 # ----------------------------------------------------------------------
19
20 # Implementación del método de Verlet y algunos casos de salida.
21
22 from math import *
23
24
25 def test1(x):  # y'' = x
26     """Función de prueba"""
27     return x
28
29
30 def test2(x):  # y'' = -x
31     """Función de prueba"""
32     return -x
33
34
35 def Verlet(a, b, x0, v0, f, N):
36     """
37     Implementación método de Verlet
38     Entradas:
39     a -- inicio intervalo
40     b -- fin intervalo
41     x0 -- aproximación inicial
42     v0 -- aproximación inicial
43     f -- función
44     N -- pasos
45
46     Salida:
47     p1 -- aproximación final
48     """
49     h = (b - a)/N
50     p0 = x0
51     p1 = p0 + v0*h + 0.5*f(p0)*h**2
```

```
52      print("a0  = {0:0.2f}, p0  = {1:0.12f}".format(a, p0))
53      for i in range(1, N+1):
54          p = 2*p1 - p0 + f(p1)*h**2
55          s = a + i*h
56          print("a{0:<2} = {1:0.2f}, p{0:<2} = {2:.12f}".format(i, s, p1))
57          p0 = p1
58          p1 = p
59      return p1
60
61
62  # d^2x/dt^2 = x, a = 0, b = 1, x_0 = 1, v_0 = 1, N = 20
63  print("Método de Verlet:")
64  Verlet(0, 1, 1, 1, test1, 20)
65
66  # d^2x/dt^2 = -x, a = 0, b = 1, x_0 = 1, v_0 = 0, N = 20
67  print("Método de Verlet:")
68  Verlet(0, 1, 1, 0, test2, 20)
```

Salida

```
Método de Verlet:
a0  = 0.00, p0  = 1.000000000000
a1  = 0.05, p1  = 1.051250000000
a2  = 0.10, p2  = 1.105128125000
a3  = 0.15, p3  = 1.161769070313
a4  = 0.20, p4  = 1.221314438301
a5  = 0.25, p5  = 1.283913092385
a6  = 0.30, p6  = 1.349721529200
a7  = 0.35, p7  = 1.418904269838
a8  = 0.40, p8  = 1.491634271150
a9  = 0.45, p9  = 1.568093358141
a10 = 0.50, p10 = 1.648472678527
a11 = 0.55, p11 = 1.732973180609
a12 = 0.60, p12 = 1.821806115642
a13 = 0.65, p13 = 1.915193565965
a14 = 0.70, p14 = 2.013369000203
a15 = 0.75, p15 = 2.116577856941
a16 = 0.80, p16 = 2.225078158322
a17 = 0.85, p17 = 2.339141155098
a18 = 0.90, p18 = 2.459052004762
a19 = 0.95, p19 = 2.585110484438
a20 = 1.00, p20 = 2.717631740325
Método de Verlet:
a0  = 0.00, p0  = 1.000000000000
a1  = 0.05, p1  = 0.998750000000
a2  = 0.10, p2  = 0.995003125000
```

```
a3  = 0.15, p3  = 0.988768742188
a4  = 0.20, p4  = 0.980062437520
a5  = 0.25, p5  = 0.968905976758
a6  = 0.30, p6  = 0.955327251054
a7  = 0.35, p7  = 0.939360207223
a8  = 0.40, p8  = 0.921044762873
a9  = 0.45, p9  = 0.900426706617
a10 = 0.50, p10 = 0.877557583594
a11 = 0.55, p11 = 0.852494566612
a12 = 0.60, p12 = 0.825300313213
a13 = 0.65, p13 = 0.796042809032
a14 = 0.70, p14 = 0.764795197827
a15 = 0.75, p15 = 0.731635598629
a16 = 0.80, p16 = 0.696646910433
a17 = 0.85, p17 = 0.659916604962
a18 = 0.90, p18 = 0.621536507978
a19 = 0.95, p19 = 0.581602569724
a20 = 1.00, p20 = 0.540214625046
```

B.5.3. RK4

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de programas.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2020 Los autores del texto.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>
# ------------------------------------------------------------------------

# Implementación del método RK4 y algunos casos de salida.

from math import *

```

```
24
25 def test1(t, y):  # y' = y - t^2 + 1
26     """Función de prueba"""
27     return y - t**2 + 1
28
29
30 def test2(t, y):  # y' = 2 - e^{-4t} - 2y
31     """Función de prueba"""
32     return 2 - exp(-4*t) - 2*y
33
34
35 def RK4(a, b, y0, f, N):
36     """
37     Implementación método RK4
38     Entradas:
39     a -- inicio intervalo
40     b -- fin intervalo
41     y0 -- aproximación inicial
42     f -- función
43     N -- pasos
44
45     Salida:
46     w -- aproximación final
47     """
48     h = (b - a)/N
49     t = a
50     w = y0
51     print("t0  = {0:.2f}, w0  = {1:.12f}".format(t, w))
52
53     for i in range(1, N+1):
54         k1 = h*f(t, w)
55         k2 = h*f(t + h/2, w + k1/2)
56         k3 = h*f(t + h/2, w + k2/2)
57         k4 = h*f(t + h, w + k3)
58         w = w + (k1 + 2*k2 + 2*k3 + k4)/6
59         t = a + i*h
60         print("t{0:<2} = {1:.2f}, w{0:<2} = {2:.12f}".format(i, t, w))
61     return w
62
63
64 # dy/dt = y - t^2 + 1, a = 0, b = 2, y_0 = 0.5, N = 10
65 print("Método RK4:")
66 RK4(0, 2, 0.5, test1, 10)
67
68 # dy/dt = 2 - e^{-4t} - 2y, a = 0, b = 1, y_0 = 1, N = 20
```

```
69 print("Método RK4:")
70 RK4(0, 1, 1, test2, 20)
```

Salida

```
Método RK4:
t0  = 0.00, w0  = 0.500000000000
t1  = 0.20, w1  = 0.829293333333
t2  = 0.40, w2  = 1.214076210667
t3  = 0.60, w3  = 1.648922017042
t4  = 0.80, w4  = 2.127202684948
t5  = 1.00, w5  = 2.640822692729
t6  = 1.20, w6  = 3.179894170232
t7  = 1.40, w7  = 3.732340072855
t8  = 1.60, w8  = 4.283409498318
t9  = 1.80, w9  = 4.815085694579
t10 = 2.00, w10 = 5.305363000693
Método RK4:
t0  = 0.00, w0  = 1.000000000000
t1  = 0.05, w1  = 0.956946773927
t2  = 0.10, w2  = 0.925794826349
t3  = 0.15, w3  = 0.903996935703
t4  = 0.20, w4  = 0.889504715870
t5  = 0.25, w5  = 0.880674661873
t6  = 0.30, w6  = 0.876191562614
t7  = 0.35, w7  = 0.875006100539
t8  = 0.40, w8  = 0.876284037659
t9  = 0.45, w9  = 0.879364861493
t10 = 0.50, w10 = 0.883728152457
t11 = 0.55, w11 = 0.888966251604
t12 = 0.60, w12 = 0.894762067259
t13 = 0.65, w13 = 0.900871071482
t14 = 0.70, w14 = 0.907106710988
t15 = 0.75, w15 = 0.913328599230
t16 = 0.80, w16 = 0.919432972496
t17 = 0.85, w17 = 0.925344987868
t18 = 0.90, w18 = 0.931012518523
t19 = 0.95, w19 = 0.936401165330
t20 = 1.00, w20 = 0.941490255536
```

Bibliografía

[1] R. Burden. *Análisis Numérico*. Grupo Editorial Iberoamericano. 2002.

[2] W.E Boyce, R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley. 2008.

[3] A.B. Downey. *Think Python*. Green Tea Press. 2012

[4] H. Th. Jongen, K. Meer, E. Triesch. *Optimization Theory*, Springer. 2004

[5] D. Kincaid, W. Cheney. *Análisis Numérico*. Addison Wesley. 1994

[6] E. Kreyszig. *Matemáticas avanzadas para Ingeniería*. Limusa Wiley. 2003.

[7] L. Leithold. *Álgebra y Trigonometría con Geometría Analítica*. Editorial Harla. 1987

[8] I. Mantilla Prada. *Análisis Numérico*. Universidad Nacional de Colombia. 2004

[9] H.M. Mora Escobar. *Introducción a* `C` *y a Métodos Numéricos*. Universidad Nacional de Colombia. 2004

[10] S. Nakamura. *Métodos Numéricos aplicados con* Software. Prentice Hall. 1998

[11] A. Nieves, F.C. Domínguez. *Métodos Numéricos aplicados a la Ingeniería*. Grupo Editorial Patria. 2014

[12] M. Pilgrim. *Dive Into Python*. Apress. 2004

El libro contiene métodos computacionales para resolver problemas esenciales en el campo de ingeniería o matemática aplicada. En cada método se pretende dar al lector una visión de su esencia, necesidad, ventajas y desventajas. En algunos casos, alejándose de presentaciones rigurosas, pero sin dejar de ser correctas. El objetivo final es proporcionar los elementos necesarios para la aplicación adecuada de los algoritmos.

ISBN: 978-958-5142-97-8