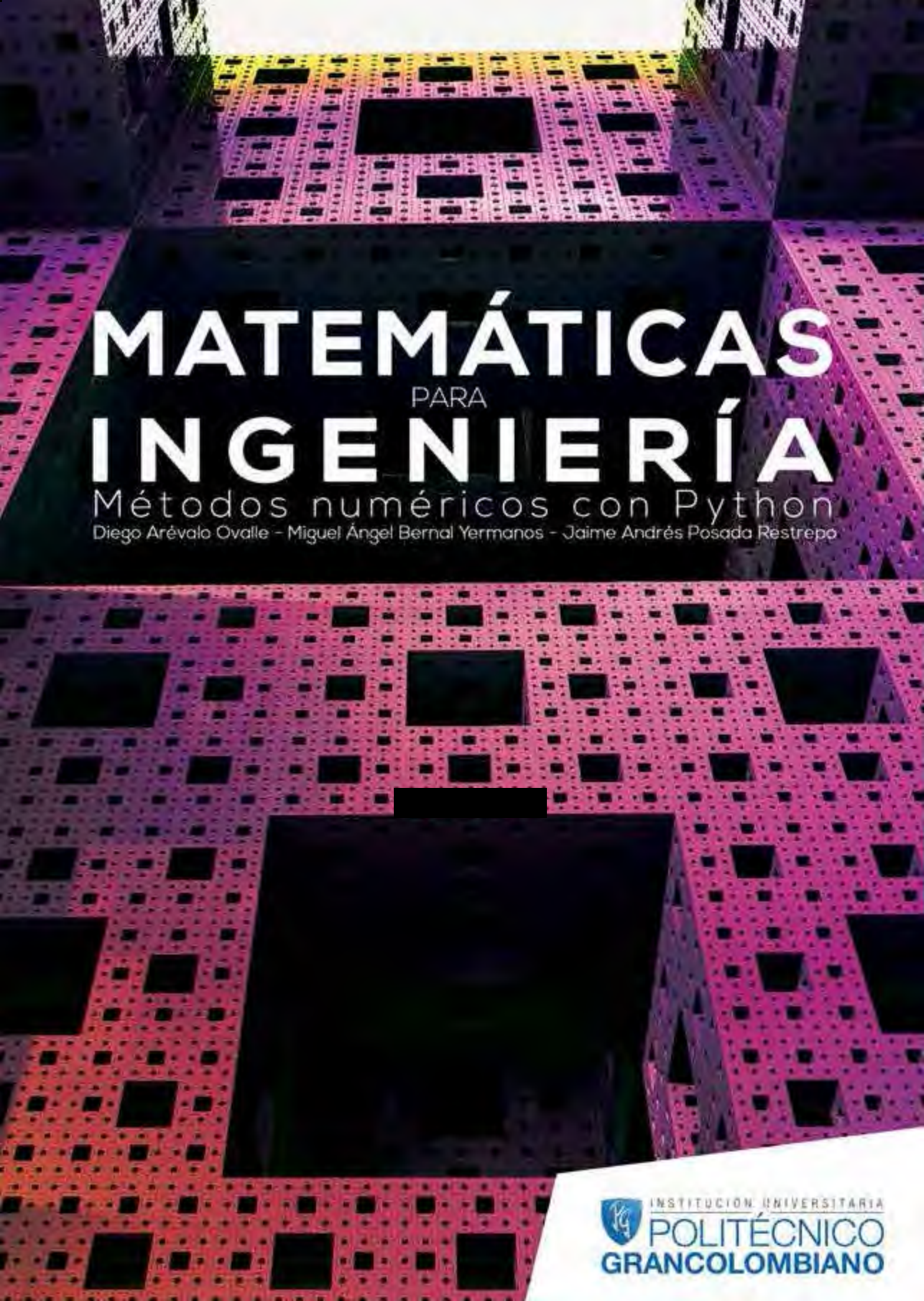

MATEMÁTICAS
PARA
INGENIERÍA
Métodos numéricos con Python
Diego Arévalo Ovalle - Miguel Ángel Bernal Yermanos - Jaime Andrés Posada Restrepo
INSTITUCIÓN UNIVERSITARIA
POLITÉCNICO
GRANCOLOMBIANO

# Matemáticas para Ingeniería.
# Métodos numéricos con `Python`

# Matemáticas para Ingeniería.

# Métodos numéricos con `Python`

Diego Arévalo Ovalle

Miguel Ángel Bernal Yermanos

Jaime Andrés Posada Restrepo

INSTITUCIÓN UNIVERSITARIA
POLITÉCNICO
GRANCOLOMBIANO

Bogotá, marzo de 2017

INSTITUCIÓN UNIVERSITARIA
POLITÉCNICO
**GRANCOLOMBIANO**



Matemáticas para Ingeniería. Métodos numéricos con `Python`.
ISBN: 978–958–8721–56–9
E ISBN: 978–958–8721–55–2

Editorial Politécnico Grancolombiano
Av. Caracas # 63–55, Piso 4
PBX: 7455555 Ext. 1171
`editorial@poligran.edu.co`

Marzo de 2017
Bogotá Colombia

Institución Universitaria Politécnico Grancolombiano
Facultad de Ingeniería y Ciencias Básicas

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

**Coordinador editorial**
Eduardo Norman Acevedo

**Analista de producción editorial**
Paulo Mora Noguera

**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
Matemáticas para Ingeniería: Métodos numéricos con `Python`. / Diego Arévalo Ovalle, Miguel Ángel Bernal Yermanos y Jaime Andrés Posada Restrepo; coordinador editorial, Eduardo Norman Acevedo. – Bogotá D.C.: Editorial Politécnico Grancolombiano, 2017.

143 p.: il.; 17 × 24 cm.
Incluye referencias bibliográficas.
ISBN: 978–958–8721–56–9
E ISBN: 978–958–8721–55–2

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? eybooks.com

Arévalo Ovalle, D., Bernal Yermanos, M. A., & Posada Restrepo, J. A. (2017), Matemáticas para Ingeniería. Métodos numéricos con `Python`, Bogotá: Editorial Politécnico Grancolombiano.

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

# 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.

El capítulo 1 expone los principales procedimientos computacionales para la solución de ecuaciones de una variable, describiendo sus ventajas y desventajas computacionales.

El capítulo 2 presenta las herramientas básicas para la estimación de nuevos datos a partir de un conjunto de puntos. Se abordan dos enfoques de interpolación: exacta y ajuste de mínimos cuadrados.

El capítulo 3 comprende los algoritmos de factorización $LU$, Jacobi y Gauss-Seidel para la solución de sistemas de ecuaciones lineales.

El capítulo 4 se encuentra dedicado a los métodos numéricos para la solucíon de ecuaciones diferenciales, iniciando con el método de Euler y finalizando con el método RK4.

Al final del libro, el lector puede encontrar todos los algoritmos presentados a lo largo del texto desarrollados en `Python`, un lenguaje de programación flexible y que responde a las necesidades didácticas del curso.

Para terminar, deseamos expresar nuestros agradecimientos a los alumnos y compañeros que a través de sus inquietudes, permitieron la construcción de los contenidos de este documento.

# Índice general

**1. Ecuaciones de una variable** **11**

1.1. Introducción . . . . . 11

1.2. Método de bisección . . . . . 13

1.2.1. Criterios de parada . . . . . 16

1.2.2. Método de *regula falsi* . . . . . 16

1.3. Método de Newton-Raphson . . . . . 18

1.3.1. Método de la secante . . . . . 20

1.4. Método de punto fijo . . . . . 22

1.5. Orden de un método de iteración . . . . . 28

1.5.1. Orden de convergencia en el método de punto fijo . . . . . 29

1.5.2. Orden del método de Newton-Raphson . . . . . 30

**2. Interpolación** **33**

2.1. Ajuste exacto . . . . . 34

2.1.1. Polinomio de interpolación de Lagrange . . . . . 34

2.1.2. Polinomio de interpolación de Newton . . . . . 38

2.1.3. Trazadores cúbicos *(Cubic Splines)* . . . . . 45

2.1.4. Trazadores cúbicos . . . . . 46

2.2. Ajuste por mínimos cuadrados . . . . . 50

2.2.1. Errores . . . . . 51

2.2.2. Funciones de ajuste . . . . . 51

2.2.3. Polinomios de mínimos cuadrados . . . . . 51

2.2.4. Ajuste exponencial . . . . . 55

**3. Sistemas de ecuaciones** **59**

3.1. Métodos directos . . . . . 59

3.1.1. Factorización *LU* . . . . . 59
3.2. Métodos iterativos . . . . . 67
3.2.1. Normas vectoriales . . . . . 67
3.2.2. Normas matriciales . . . . . 69
3.2.3. Métodos iterativos en la solución de sistemas de ecuaciones lineales . . . . . 71

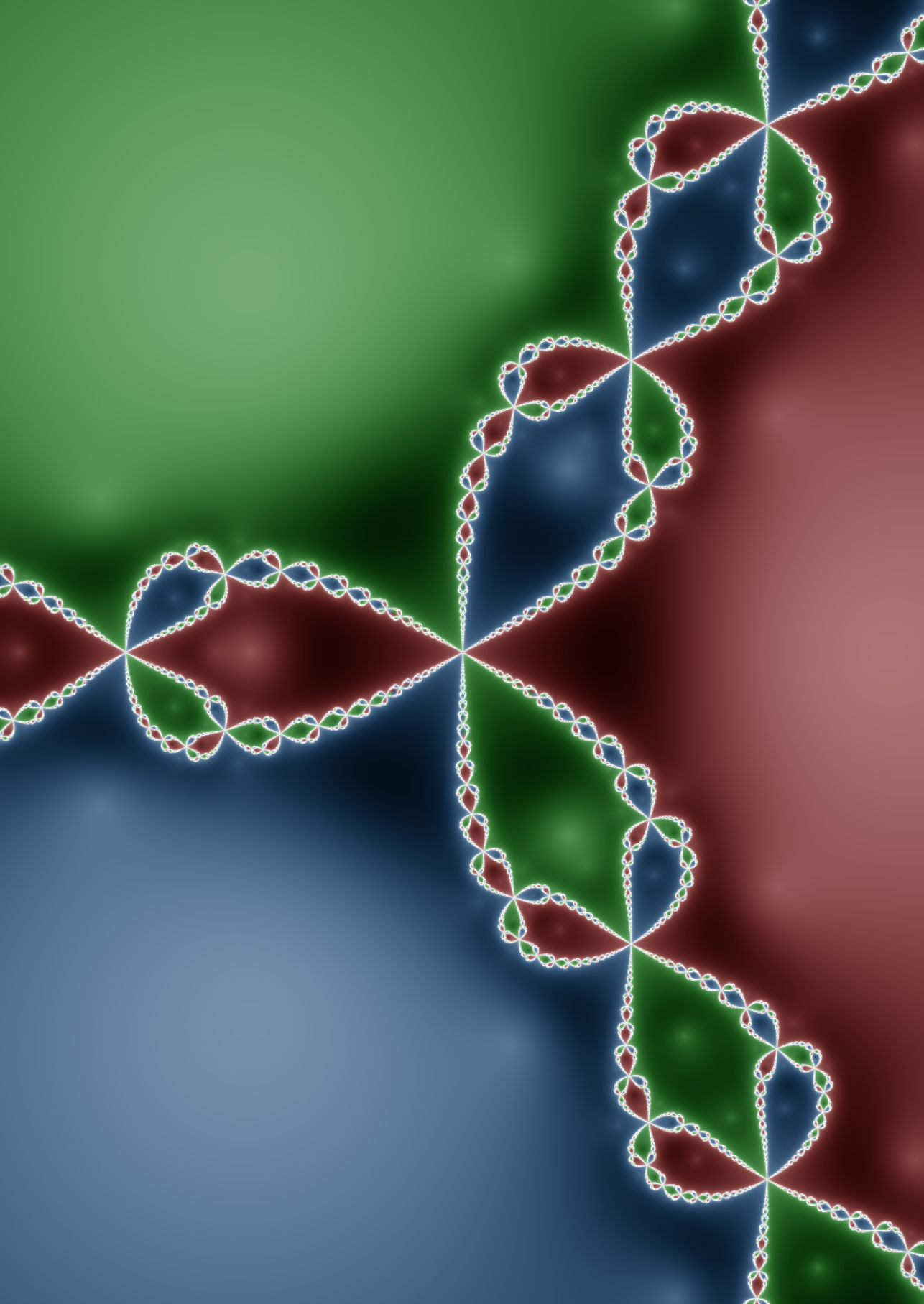
**4. Ecuaciones diferenciales** **83**
4.1. Problemas de valor inicial . . . . . 84
4.1.1. Método de Euler . . . . . 84
4.1.2. Orden del método de Euler . . . . . 88
4.1.3. Método de Verlet . . . . . 89
4.1.4. Error del método de Verlet . . . . . 90
4.1.5. Métodos de Runge-Kutta orden 2 . . . . . 93
4.1.6. Método de Runge-Kutta orden 4 (RK4) . . . . . 97

**A. Tutorial de `Python`** **103**
A.1. Generalidades . . . . . 103
A.2. Consola interactiva . . . . . 103
A.3. Funciones . . . . . 104
A.4. Estructuras básicas de control . . . . . 104
A.4.1. `if` . . . . . 104
A.4.2. `while` . . . . . 105
A.4.3. `for` . . . . . 105
A.5. Ejemplos . . . . . 105
A.5.1. Factorial . . . . . 105
A.5.2. GCD . . . . . 106
A.5.3. ¿Es palíndromo? . . . . . 106
A.5.4. `Numpy` . . . . . 106

**B. Compendio de algoritmos** **109**
B.1. Ecuaciones de una variable . . . . . 109
B.1.1. Bisección . . . . . 109
B.1.2. *Regula falsi* . . . . . 112
B.1.3. Newton . . . . . 115
B.1.4. Secante . . . . . 117

B.1.5. Punto fijo . . . . . 119
B.2. Interpolación . . . . . 121
B.2.1. Lagrange . . . . . 121
B.2.2. Diferencias divididas de Newton . . . . . 122
B.2.3. Trazadores cúbicos . . . . . 124
B.2.4. Recta de ajuste mínimos cuadrados . . . . . 126
B.3. Sistemas de ecuaciones . . . . . 128
B.3.1. *LU* . . . . . 128
B.3.2. Jacobi . . . . . 131
B.3.3. Gauss-Seidel . . . . . 134
B.4. Ecuaciones diferenciales . . . . . 137
B.4.1. Euler . . . . . 137
B.4.2. Verlet . . . . . 139
B.4.3. RK4 . . . . . 141

**Bibliografía** **145**



# Capítulo 1

# Ecuaciones de una variable

## 1.1. Introducción

Uno de los problemas más antiguos, y que con mayor frecuencia se debe resolver en matemáticas aplicadas, es encontrar ceros de funciones, es decir, dada la función $f(x)$, 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 1.** 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:

i. Sumar a ambos lados de la igualdad 4:

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

que lleva a:

$$3x = 4$$

ii. 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:

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

$\diamondsuit$

**Ejemplo 2.** 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:

i. Se suma 6 a ambos lados de la ecuación:

$$x^2 - x = 6$$

ii. Se completa el trinomio cuadrado perfecto:

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

iii. Se factoriza el lado izquierdo de la igualdad:

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

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

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

v. Se resuelve la ecuación anterior para obtener:

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

Finalmente, se obtiene las dos soluciones (raíces) de esta ecuación, que son:

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

♢

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

i) $x - \cos x = 0$

ii) $e^{-x} - \sin x = 0$

iii) $x - \tan x = 0$

iv) $e^{-x} - x = 0$

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 1.1, el punto de corte es el cero que se busca.

Surge la pregunta ¿qué se puede hacer?, y es en este momento cuando entran los métodos numéricos a resolver estos interrogantes. Aunque varias técnicas van a permitir solucionar estas preguntas, sus respuestas no serán exactas, contrario a las soluciones de los ejemplos 1 y 2. 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 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 una aproximación de la solución del problema general de este capítulo.

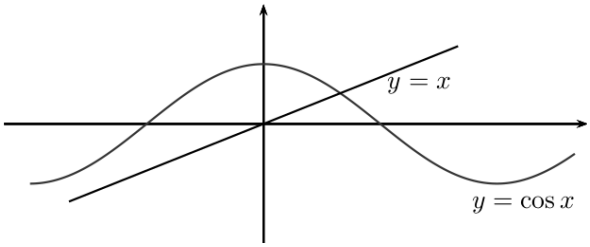



Figura 1.1: la coordenada $x$ del punto de corte de las gráficas de las dos funciones es el cero de la función $f(x) = x - \cos x$.

## 1.2. Método de bisección

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

**Teorema 1** (Teorema de valor intermedio)**.** *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 $f(a)f(b) < 0$, entonces 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 del intervalo (por esto se llama método de bisección) $[a_0, b_0]$, $c_0 = \frac{a_0+b_0}{2}$.
3. a. Si $f(c_0) = 0$ entonces $c_0$ es un cero de la función y por lo tanto una solución del problema.
   b. Si $f(a_0)f(c_0) > 0$ quiere decir que existe un cero de la función en el intervalo $(c_0, b_0)$, entonces seleccionar este intervalo y definir $a_1 = c$ y $b_1 = b_0$.
   c. Si $f(a_0)f(c_0) < 0$ quiere decir que existe un cero de la función en el intervalo $(a_0, c_0)$, entonces seleccionar este intervalo y definir $a_1 = a$ y $b_1 = c_0$.

   **Nota:** observar que ninguno de los tres casos mencionados arriba se pueden 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 $(c_0, b_0)$, por el teorema del valor intermedio, esto no descarta que pueda existir otra raíz en el intervalo $(a_0, c_0)$. ¿Por qué?
4. Si $f(c_0) \neq 0$, repetir los pasos 2, 3 y 4 para el nuevo intervalo $[a_1, b_1]$

A continuación se demuestra la convergencia del método y en la figura 1.2 se muestra en forma gráfica algunos de los pasos en su ejecución.

**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

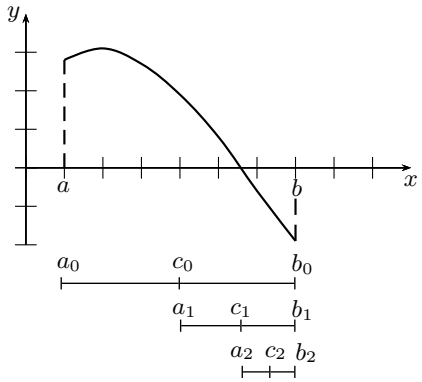



Figura 1.2: un ejemplo del método de bisección.

a un cuarto de la medida del intervalo inicial; 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:

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

Entonces $0 \leq |c_n - c| \leq \frac{1}{2^n}|b_0 - a_0|$. Si ahora se toma el límite cuando el número $n$ de iteraciones tiende al infinito, se tiene que $\lim_{n\to\infty} 0 \leq \lim_{n\to\infty} |c_n - c| \leq \lim_{n\to\infty} \frac{1}{2^n}|b_0 - a_0|$ en caso de existir los límites. Notar que $|b_0 - a_0|$ es una constante, dado que es la medida del intervalo inicial y dado que no cambia, se tiene que $\lim_{n\to\infty} \frac{1}{2^n}|b_0 - a_0| = |b_0 - a_0| \lim_{n\to\infty} \frac{1}{2^n} = 0$. También es claro que $\lim_{n\to\infty} 0 = 0$, y por tanto $\lim_{n\to\infty} |c_n - c|$ existe y es igual a 0. Luego se tiene que $\{c_n\}_{n=0}^{\infty}$ converge a $c$. □

Como se puede observar, el procedimiento encierra a un cero de la función en un sub-intervalo que en cada iteración es "más pequeño" que el anterior. Dado que $|c_n - c| \leq \frac{1}{2^n}|b_0 - a_0|$ 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 $\varepsilon$

real positivo, entonces es suficiente que $\frac{1}{2^n}|b_0 - a_0| \leq \varepsilon$. Se sigue entonces:

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

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

**Ejemplo 3.** Determinar un cero de la función $f(x) = -\frac{1}{10}x^2 + 3$ utilizando el método de bisección.

***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 $c_0 = \frac{a_0 + b_0}{2} = \frac{1+7}{2} = 4$. 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 1.1 presenta los resultados.

| $n$ | $a_n$ | $b_n$ | $c_n$ | $f(a_n)$ | $f(c_n)$ |
|---|---|---|---|---|---|
| 0 | 1 | 7 | 4 | 2,9 | 1,4 |
| 1 | 4 | 7 | 5,5 | 1,4 | -0,025 |
| 2 | 4 | 5,5 | 4,75 | 1,4 | 0,74375 |
| 3 | 4,75 | 5,5 | 5,125 | 0,74375 | 0,3734375 |
| 4 | 5,125 | 5,5 | 5,3125 | 0,3734375 | 0,177734375 |
| 5 | 5,3125 | 5,5 | 5,40625 | 0,177734375 | 0,077246094 |
| 6 | 5,40625 | 5,5 | 5,453125 | 0,077246094 | 0,026342773 |
| 7 | 5,453125 | 5,5 | 5,4765625 | 0,026342773 | 0,000726318 |
| 8 | 5,4765625 | 5,5 | 5,48828125 | 0,000726318 | -0,012123108 |
| 9 | 5,4765625 | 5,48828125 | 5,482421875 | 0,000726318 | -0,005694962 |
| 10 | 5,4765625 | 5,482421875 | 5,479492188 | 0,000726318 | -0,002483463 |
| 11 | 5,4765625 | 5,479492188 | 5,478027344 | 0,000726318 | -0,000878358 |
| 12 | 5,4765625 | 5,478027344 | 5,477294922 | 0,000726318 | -7,59661E-05 |
| 13 | 5,4765625 | 5,477294922 | 5,476928711 | 0,000726318 | 0,00032519 |
| 14 | 5,476928711 | 5,477294922 | 5,477111816 | 0,00032519 | 0,000124615 |
| 15 | 5,477111816 | 5,477294922 | 5,477203369 | 0,000124615 | 2,43253E-05 |
| 16 | 5,477203369 | 5,477294922 | 5,477249146 | 2,43253E-05 | -2,58202E-05 |

Cuadro 1.1: solución de la ecuación $-\frac{1}{10}x^2 + 3 = 0$ con el método de bisección.

Se tiene entonces que $c_{16} = 5,477249146$ es una aproximación de un cero de la función con un error de

$$|c_{16} - c| \leq |b_{16} - a_{16}| = 0,000091552$$

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

**Ejemplo 4.** 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 $\varepsilon = 10^{-3}$ si $a_0 = 0,5$ y $b_0 = 1$.

***Solución:***

$$\begin{aligned} n &= \left\lceil \frac{\ln|b_0 - a_0| - \ln\varepsilon}{\ln 2} \right\rceil \\ &= \left\lceil \frac{\ln 0,5 - \ln 10^{-3}}{\ln 2} \right\rceil \\ &= \lceil 12,28771238 \rceil \\ &= 13 \end{aligned}$$

Por lo tanto se necesitan 13 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.

### 1.2.1. 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 $|c_N - c| < \frac{\varepsilon}{2}$ para una tolerancia $\varepsilon > 0$, es necesario generar $c_1, c_2, c_3, \ldots$ hasta que se cumpla una de las siguientes condiciones:

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

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

**Ejemplo 5.** Sea $c_n = \left(1 + \frac{1}{n}\right)^n$, determinar $N \in \mathbb{N}$ tal que $|c_N - c_{N-1}| < 10^{-4}$

***Solución:*** se genera $c_1 = 2, c_2 = 2,25, \ldots$ y se evalúa $|c_N - c_{N-1}|$, concluyendo que para $N = 369$ se tiene $|c_{369} - c_{368}| = |2,714607646 - 2,714597687| < 10^{-4}$. ♢

### 1.2.2. 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 ve en la figura 1.3.

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 $(b_n, f(n))$ es $y = \frac{f(b_n)-f(a_n)}{b_n - a_n}(x - a_n) + f(a_n)$ y su corte con el eje $x$, $a_n - \frac{f(a_n)(b_n - a_n)}{f(b_n)-f(a_n)}$, define el punto $c_n$, que se espera sea una mejor aproximación.

**Ejemplo 6.** Determinar un cero de la función $f(x) = -\frac{1}{10}x^2 + 3$ con un error inferior a $\varepsilon = 9,8 \times 10^{-4}$, 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 $c_0 = a_0 - \frac{f(a_0)(b_0 - a_0)}{f(b_0)-f(a_0)} = 4,625$ y $f(c_0) = 0,8609375$. Por lo tanto $f(a_0)f(c_0) > 0$,

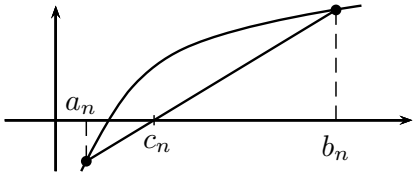



Figura 1.3: método de *regula falsi*.

| $n$ | $a_n$ | $b_n$ | $c_n$ | $f(a_n)$ | $f(c_n)$ |
|---|---|---|---|---|---|
| 0 | 1 | 7 | 4,625 | 2,9 | 0,8609375 |
| 1 | 4,625 | 7 | 5,365591398 | 0,8609375 | 0,121042895 |
| 2 | 5,365591398 | 7 | 5,463478261 | 0,121042895 | 0,015040529 |
| 3 | 5,463478261 | 7 | 5,475545943 | 0,015040529 | 0,001839663 |
| 4 | 5,475545943 | 7 | 5,477020558 | 0,001839663 | 0,000224581 |
| 5 | 5,477020558 | 7 | 5,477200553 | 0,000224581 | 2,74097E-05 |
| 6 | 5,477200553 | 7 | 5,477222521 | 2,74097E-05 | 3,34521E-06 |

Cuadro 1.2: solución de la ecuación $-\frac{1}{10}x^2 + 3 = 0$ con el método de *regula falsi*.

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 1.2 presenta los resultados.

Luego, utilizando el criterio de parada, $c_6$ es una aproximación de la solución de la ecuación $-\frac{1}{10}x^2+3 = 0$ con un error inferior a $\varepsilon = 9,8 \times 10^{-4}$. ♢

**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 formula 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 1

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

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 $[-1, 0]$

   *b)* $g(x) = 2^x(x - 6) - x$ en $[-5, 5]$

   *c)* $h(x) = \sin(3x) - \cos(2x) - 1$ en $[-8, -1]$

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

   *e)* $x^{-2} - \tan x$ en $[3, 4]$

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

3. Aplicar el método de bisección para la función $f(x) = \frac{1}{x}$, 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 $[1,3]$

   *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 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 los ceros en este intervalo con precisión hasta la tercera cifra decimal.

7. Con ayuda de un programa graficador, 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.

## 1.3. 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 1.4) 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 $p_1$.

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 $x = p_0 - \frac{f(p_0)}{f'(p_0)}$. 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 el sentido de cantidad de iteraciones.

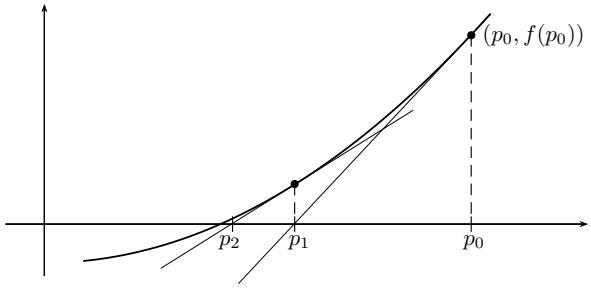



Figura 1.4: método de Newton-Raphson.

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

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \tag{1.1}$$

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

**Ejemplo 7.** Determinar una solución de la ecuación $e^{-x} - \sin x = 0$ con una precisión de $\varepsilon = 10^{-3}$ 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, $p_2 = p_1 - \frac{f(p_1)}{f'(p_1)} = 0,587795322$, observando los resultados del cuadro 1.3 y utilizando el criterio de parada, se obtiene que $p_4 = 0,588532744$ es una aproximación de la solución de la ecuación $e^{-x} - \sin x = 0$ con una precisión de $\varepsilon = 10^{-3}$.

| $n$ | $p_n$ |
|---|---|
| 0 | 0,2 |
| 1 | 0,544708885 |
| 2 | 0,587795322 |
| 3 | 0,588532526 |
| 4 | 0,588532744 |

Cuadro 1.3: resultados del método de Newton-Raphson.

♢

### 1.3.1. 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 aritméticas. 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 que el método de Newton-Raphson sea convergente, 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 1.5) 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 $p_2$.

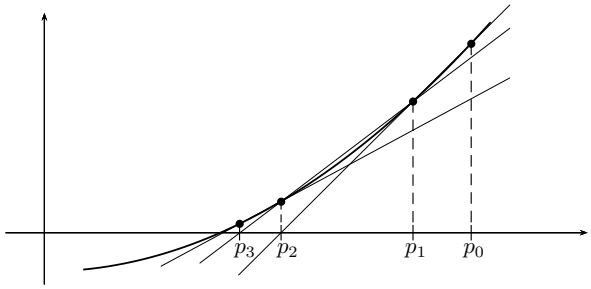



Figura 1.5: método de la secante.

**Ejemplo 8.** Aplicar el método de la secante para obtener una solución de la ecuación $x - 0,5\tan(x) = 0$ con una precisión de $\varepsilon = 10^{-4}$.

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

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} = 1 - \frac{0,221296138(1 - 1.2)}{0,221296138 - (-0,086075811)} = 1,14399241$$

| $n$ | $p_n$ |
|---|---|
| 0 | 1,2 |
| 1 | 1 |
| 2 | 1,14399241 |
| 3 | 1,180243146 |
| 4 | 1,164484057 |
| 5 | 1,165508347 |
| 6 | 1,165561378 |
| 7 | 1,165561185 |

Cuadro 1.4: resultados del método de la secante.

Los resultados del método se presentan en el cuadro 1.4.

Luego $p_7 = 1,165561185$ es una aproximación de la solución a la ecuación $x - 0,5\tan(x) = 0$. ♢

## Ejercicios 2

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)* $e^{-2}x^2 + 2e^{-1}x = -1$ en $[-4, -2]$

   *d)* $x^2\cos^2(x) + \cos^2(x) - 2x^3\cos(x) - 2x\cos(x) + x^4 + x^2$ en $[-1, 1]$

3. **[Ejemplos clásicos]** Utilizar el método de Newton-Raphson para aproximar la solución de cada ecuación desde el punto $p_0$. Determinar qué ocurre y justificar.

   *a)* $\tan^{-1}(x) = 0$, $p_0 = 1,39174520027073$.

   *b)* $-x^4 + 6x^2 + 11$, $p_0 = 1$.

   *c)* $1 - 2e^{-x^2} = 0$, $p_0 = 0,83255461$.

   *d)* $\sin x - e^{-x}$, $p_0 = 3,5$.

4. Usar el método de Newton-Raphson para aproximar $\sqrt[3]{25}$ con cuatro 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 $f(x) = \ln(x)$, $p_0 = 2,8$ y $p_1 = 2,64$. 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 $p_0 = 2,8$.

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 $p_0 = 2$. Comparar (en cantidad de iteraciones) el resultado con el obtenido al aplicar el método de la secante con $p_0 = 2$ y $p_1 = 1,9$.

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 $n \to \infty$, $x_n \to A$.

10. Dos números $x$ y $y$, al multiplicarse dan 45. Si $\sqrt{x}\,(y + 12,648001) = 25$, con el método de Newton-Raphson encontrar los dos números con precisión de cuatro cifras decimales.

## 1.4. 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 $n \geq 0$

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

**Ejemplo 9.** La función $g(x) = x$, tiene infinitos puntos fijos, dado que para cualquier valor de $c$, $g(c) = c$.

**Ejemplo 10.** La función $g(x) = x^2 + 6x + 6$ tiene dos puntos fijos, $x = -2$ y $x = -3$. ¿Por qué?

***Solución:*** observar:

$g(-2) = (-2)^2 + 6(-2) + 6 = 4 - 12 + 6 = -2$, luego $g(-2) = -2$.

$g(-3) = 9 - 18 + 6 = -3$, luego $g(-3) = -3$. ¿Tendrá más puntos fijos? ♢

¿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 10, 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 1.6):

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.*

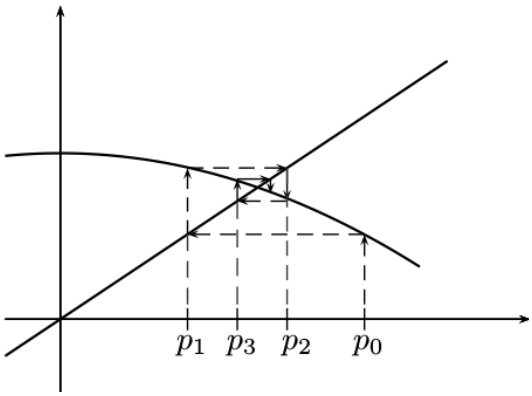



Figura 1.6: método de punto fijo.

**Ejemplo 11.** 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 $\varepsilon = 10^{-4}$.

***Solución:*** seleccionar un punto inicial $p_0$, en este caso $p_0 = 1.5$. Calcular $p_1 = g(p_0) = \sqrt{\dfrac{10}{p_0+4}} = 1,348399725$, y dado que $|p_0 - p_1| > 10^{-4}$, se repite el proceso. Calcular $p_2 = g(p_1) = 1.367376372$, y ya que $|p_1 - p_2| > 10^{-4}$ se sigue iterando. Los resultados se resumen en el cuadro 1.5.

| $n$ | $p_n$ |
|---|---|
| 0 | 1,5 |
| 1 | 1,348399725 |
| 2 | 1,367376372 |
| 3 | 1,364957015 |
| 4 | 1,365264748 |
| 5 | 1,365225594 |

Cuadro 1.5: resultados del método de punto fijo.

Luego $p_5 = 1,365225594$ es una aproximación a la solución de la ecuación $x = \sqrt{\dfrac{10}{x+4}}$ con la precisión deseada. ♢

En este punto es necesario presentar condiciones suficientes para garantizar que el método de punto fijo sea convergente.

**Teorema 4** (**Teorema de punto fijo**). *Sean $g : [a,b] \to [a,b]$ una función continua, $s \in [a,b]$ tal que $g(s) = s$ y $|g'(x)| \leq \alpha < 1$, para toda $x \in (a,b)$. 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 $\zeta$ 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

$$|g(\zeta)| = \frac{|g(p_0) - g(s)|}{|p_0 - s|} \leq \alpha$$

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

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 una fracción[2] de la distancia entre $p_0$ y $s$. Se repite el razonamiento, pero empezando con $p_1$ y $s$, lo cual lleva a:

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

luego

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

[2]Estrictamente no es necesario que sea una fracción, pues se sabe que hay muchos irracionales entre cualquier par de números, en particular entre 0 y 1, que son los valores posibles de $\alpha$. Para nuestros propósitos se puede mantener la imagen de fracción, para dar la idea que se encoge la distancia entre $p_0$ y $s$.

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 $\alpha$ es un número entre 0 y 1 se tiene que $\lim_{n\to\infty} \alpha^n = 0$ y por tanto luego:

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

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

Para ilustrar cómo actúa el método de punto fijo y su relación con la ecuación $f(x) = 0$ se analizará un par de situaciones.

**Ejemplo 12.** Encontar una aproximación con cuatro decimales correctas a una solución de $x^3 + 4x^2 - 10 = 0$.

***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 $g(x) = \sqrt{\dfrac{10}{x+4}}$ cumple las hipótesis del teorema 4 en algún intervalo, se invita al lector a revisar que el intervalo $[1, 2]$ efectivamente es adecuado. Al seleccionar $p_0 \in [1, 2]$, en este caso $p_0 = 1,5$, se obtienen los resultados del cuadro 1.5 presentado en el ejemplo 11. ♢

**Ejemplo 13.** Encontrar una aproximación con cuatro decimales correctas 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 $p_0 = 0$.

| $i$ | $p_i$ | $\lvert p_i - p_{i-1}\rvert$ |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | 1 |
| 2 | 0,540302306 | 0,459697694 |
| 3 | 0,857553216 | 0,317250910 |
| 4 | 0,654289790 | 0,203263425 |
| 5 | 0,793480359 | 0,139190568 |
| 6 | 0,701368774 | 0,092111585 |
| 7 | 0,763959683 | 0,062590909 |
| ⋮ | ⋮ | ⋮ |
| 26 | 0,739071365 | 3,42066E-05 |
| 27 | 0,739094407 | 2,30421E-05 |
| 28 | 0,739078886 | 1,55214E-05 |

Cuadro 1.6: solución de la ecuación $x = \cos x$.

4. Aplicar la iteración $p_{n+1} = g(p_n)$.

En el cuadro 1.6 se incluyen los valores de la sucesión $p_1, p_2, p_3, \ldots, p_{28}$, y adicionalmente se presenta la medida del error en cada paso.

**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 y el programa podría caer en un ciclo infinito intentando alcanzar un cero al cual nunca va a llegar. 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 valor de la función $|g(p_n)| < \epsilon_1$, o como el que se presenta en la última columna del cuadro 1.6, es decir $|p_{i+1} - p_i| < \epsilon_2$. ♢

**Ejemplo 14.** 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 $g(x) = x^3 - 1$.
2. Encontrar el intervalo donde $g(x)$ cumple las hipótesis del teorema 4. En este caso, $[1, 2]$ es un intervalo tal que $g([1, 2]) \subseteq [1, 2]$ y además existe $p \in [1, 2]$ punto fijo de $g$.
3. Elegir $p_0$ dentro del intervalo. Para este caso tomamos $p_0 = 0$.
4. Aplicamos $p_{n+1} = g(p_n)$ obteniendo los resultados del cuadro 1.7.

En este momento es mejor detener el proceso, puesto que la sucesión no tiende a ningún número en particular. Por el contrario, tiende a $\infty$, 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. Lo anterior, en algunas ocasiones exige una búsqueda exhaustiva, requiriendo aplicar otros métodos menos eficientes.

## Ejercicios 3

| $i$ | $p_i$ |
|---|---|
| 0 | 1,5 |
| 1 | 2,375 |
| 2 | 12,39648438 |
| 3 | 1904,002772 |
| 4 | 6902441413 |
| 5 | 3,28858E+29 |
| 6 | 3,55651E+88 |
| 7 | 4,4986E+265 |

Cuadro 1.7: solución de la ecuación $x = x^3 - 1$.

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)* $xe^{-2x} + \sin(2x + 1) = 0$ con $-0.5 \leq x \leq 0$

   *d)* $0.5\sin(e^{-2x}) - x = 0$

   *e)* $x^3 - x - 10 = 0$

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

2. Dados los siguientes cuatro esquemas de punto fijo para obtener una solución de la ecuación $x^3 - 4x^2 + 10 = 0$, clasificar por la rapidez de convergencia y 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 por 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 $\varepsilon = 10^{-8}$. Comparar los resultados. *Sugerencia:* usar $g(x) = \dfrac{e^{-3x} - x^2e^{-x} + xe^{-2x} - x^3}{2e^{-2x} - 2x^2}$.

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

6. 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 $g(x) = \dfrac{2x - 1}{x^2 - 2x + 2}$.

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

8. Considerar la fracción continua:
$$x = \frac{1}{1 + \frac{1}{1+\frac{1}{1+\cdots}}}.$$
Con razonamientos de punto fijo, demostrar que tiende a $\phi - 1$.

## 1.5. Orden de un método de iteración

Cuando se utiliza un método iterativo, como los desarrollados en las secciones anteriores, es útil conocer la "*velocidad*" con que converge, constituyendo un criterio para determinar la eficiencia de un procedimiento en la solución de un problema particular. La velocidad de un método iterativo no debe entenderse como el *tiempo* necesario para alcanzar una precisión deseada, en cambio, se refiere a la proporción en que un método converge, representando en algunos casos la cantidad de cifras decimales que se alcanzan en cada iteración. Para entender este concepto y desarrollarlo en algunos casos especiales, es necesario presentar algunas definiciones.

**Definición 2.** *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 $\lambda$ y $\alpha$ 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 $\alpha$ y constante de error asintótica $\lambda$.*

**Ejemplo 15.** La sucesión $\{\frac{1}{n}\}_{n=0}^{\infty}$ converge a 0 con orden 1.

***Solución:*** Para comprobar que la sucesión $p_n = \frac{1}{n}$ converge a 0 con orden 1 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,

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

y por tanto la convergencia de $\{\frac{1}{n}\}$ es de orden 1. ♢

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 puede esperar, entre más alto sea el orden de un método, éste converge más rápido. Aunque en ocasiones lo anterior no ocurre, es un buen criterio para clasificar los métodos iterativos.

En el caso de métodos iterativos del esquema $p_n = g(p_{n-1})$, se estudia su orden de convergencia a través de la sucesión $\{p_n\}_{n=0}^{\infty}$ y se afirma que el método converge a la solución $p = g(p)$ con orden $\alpha$.

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 tienen una convergencia lineal como se verá a continuación.

### 1.5.1. 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.

**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 1.4 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 $\xi_n$ 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 $\xi_n$ 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 a $p$. □

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 $g(x), g'(x), g''(x), \ldots$ en $p$, entonces se puede expandir $g(x)$ en serie de Taylor alrededor de $p$:

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

Al evaluar la serie en $p_n$, que está "cerca" de $p$ se obtiene

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

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 enésima iteración como $e_n = p_n - p$, lleva a

$$e_{n+1} = g'(s)(p_n - p) + \frac{g''(p)}{2!}(p_n - p)^2 + \frac{g'''(p)}{3!}(p_n - p)^3 + \cdots \tag{1.2}$$

Si $|e_n| < 1$ (nuestro criterio para "cerca" o "buena aproximación"), entonces la magnitud de $|e_n^2|$ es más pequeña y la de $|e_n^3|$ aun más pequeña y así sucesivamente. En estas circunstancias, si $g'(p) \neq 0$, entonces el primer término domina el error y por lo tanto $e_{n+1} \approx e_n$ y el método es de orden 1.

Si $g'(p) = 0$ y $g''(p) \neq 0$, el segundo término de la serie 1.2 es el que domina el error y por lo tanto $e_{n+1} \approx e_n^2$ y el método es de orden 2. 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 $e_{n+1} \approx e_n^3$.

### 1.5.2. 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

$$\begin{cases} p_0 \\ p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)} \quad \text{si } n \geq 0 \end{cases}$$

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

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

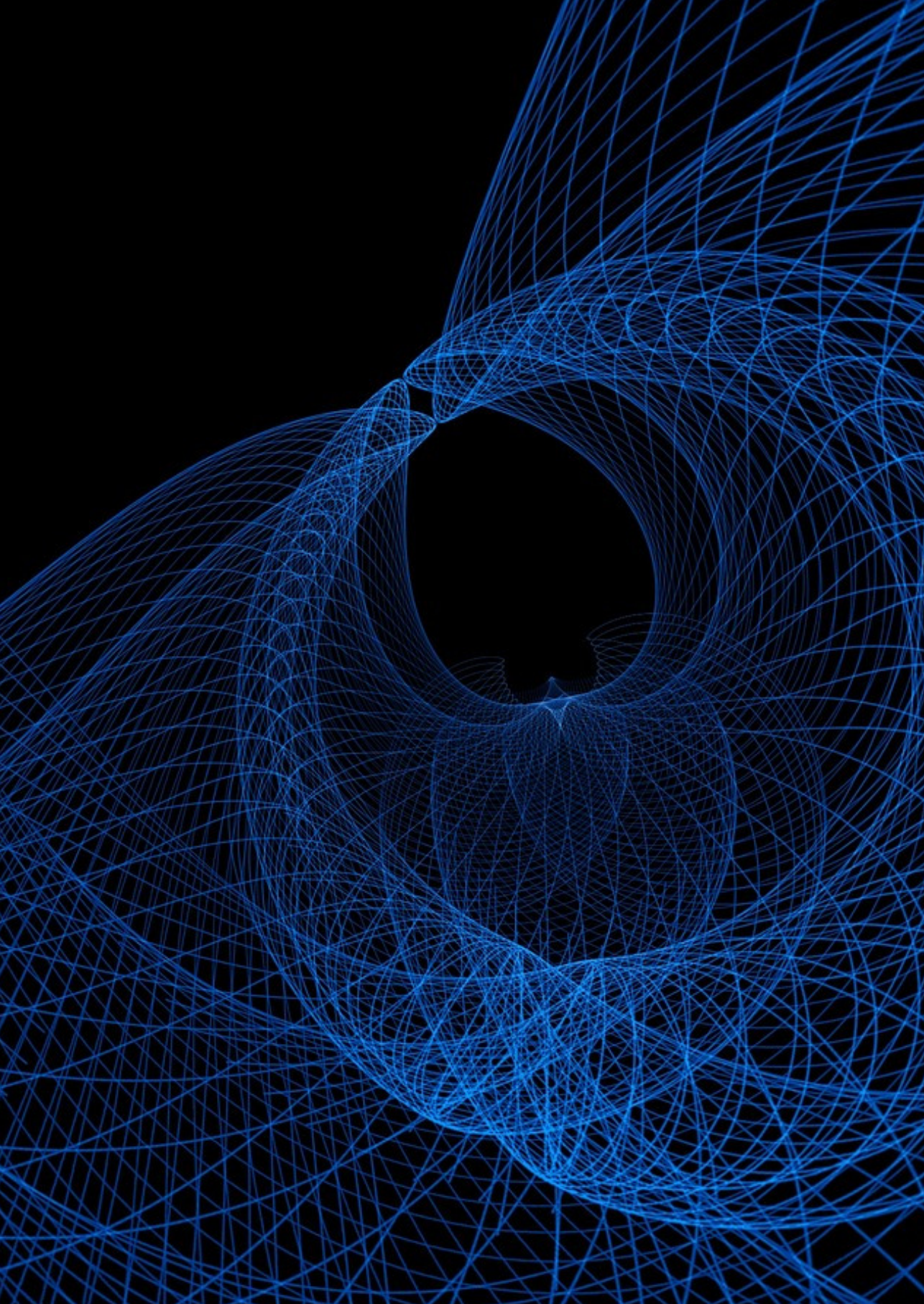
Luego el método parece ser de orden 2, pero todavía no es posible asegurar lo anterior hasta verificar el valor de la segunda derivada:

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

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.



# Capítulo 2

# Interpolación

El problema de *interpolar* aparece con mucha frecuencia en el trabajo de ciencias e ingeniería, donde los experimentos arrojan datos numéricos y a su vez estos datos se puede representar en tablas o en forma gráfica. Para analizar el comportamiento del sistema no basta con observar los datos obtenidos en el experimento (figura 2.1), y en cambio se debe intentar "ajustar" o aproximar una curva (función), que permita *predecir* valores al interior del rango, es decir, en puntos para los cuales no se dispone de información, y a su vez sirva como modelo del comportamiento del sistema.

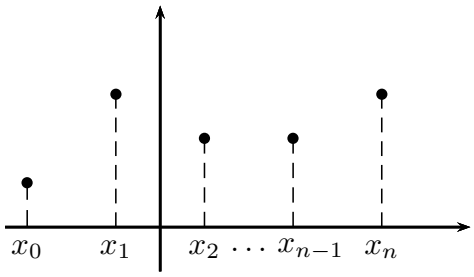



Figura 2.1: problema de interpolación.

Se puede ajustar funciones (de interpolación) a una lista de datos siguiendo dos criterios: ajuste exacto y ajuste por mínimos cuadrados (figura 2.2). Decidir cuál tipo de ajuste se utiliza depende de las necesidades de la investigación, origen de los datos y posibilidades computacionales.

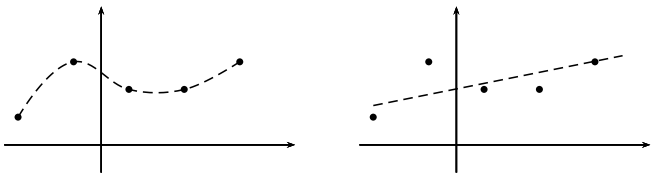


Figura 2.2: tipos de interpolación: ajuste exacto (izquierda) y ajuste por mínimos cuadrados (derecha).

## 2.1. Ajuste exacto

Una de las técnicas de interpolación (exacta) para un conjunto de datos $(x_0, y_0)$, $(x_1, y_1)$, $(x_2, y_2)$,$\cdots$, $(x_n, y_n)$ es la *interpolación polinomial*, donde se busca un polinomio $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ 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.

### 2.1.1. 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 $(x_0, y_0)$ y que se anule en $x_1$, $x_2$, ..., $x_n$, es decir,

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

  ¿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 $x_3, x_4, \cdots, x_n$. 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 1. 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 $x_2$,$\cdots$, en $x_n$, por la misma razón que $\tilde{P}_0(x_j)$ y cuando se evalúa en 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 $P_1(x)$:

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

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

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 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*:

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

En notación de sumatoria:

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

El grado de $P_n(x)$ es $n$ o menor, debido a que cada término es de a lo más grado $n$.

**Ejemplo 16.** 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:**

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

Operando y simplificando:

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

**Segundo polinomio:**

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

Operando y simplificando:

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

**Tercer polinomio:**

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

Operando y simplificando:

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

**Cuarto polinomio:**

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

Operando y simplificando:

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

Una vez calculados todos los polinomios de Lagrange, 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:

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

de donde, al simplificar se obtiene finalmente:

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

A primera vista parece un polinomio de grado 3, pero si se expanden los productos y se suman términos semejantes, podría ocurrir que el coeficiente del término cúbico sea cero. Sin embargo, existe una forma conveniente de saber el coeficiente del término cúbico de este ejemplo. Basta con sumar los coeficientes de cada uno de los cuatro términos del polinomio de interpolación, es decir:

$$-\frac{1}{3}-\frac{1}{2}-\frac{1}{2}-\frac{1}{3}=-\frac{5}{3},$$

de donde es ahora posible asegurar que se trata de un polinomio cúbico. ♢

**Nota:** este procedimiento también se puede aplicar al caso general, es decir, cuando se calcula el polinomio de Lagrange de una lista de $n+1$ puntos. En este caso, el polinomio de interpolación tiene $n+1$ términos cada uno con $n$ factores lineales. Al sumar los coeficientes de estos términos, se calcula el coeficiente del término de grado $n$.

## Ejercicios 4

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. 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.

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

5. 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 2.71828 y 1.

6. Sea $P(x)$ el polinomio de Lagrange que interpola los nodos $(x_0,y_0)$, $(x_1,y_1)$; $Q(x)$ 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 $i\neq j$.

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

8. Determinar la cantidad de divisiones y multiplicaciones necesarias para construir el polinomio de interpolación de Lagrange de 3 puntos.

9. ¿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 $i \neq j$?

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

### 2.1.2. 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 este, se construye un polinomio de grado uno que pasa por otro punto de la lista, y desde estos dos últimos puntos, se construye un polinomio de grado 2 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 2.3 se grafican algunos puntos de la lista.

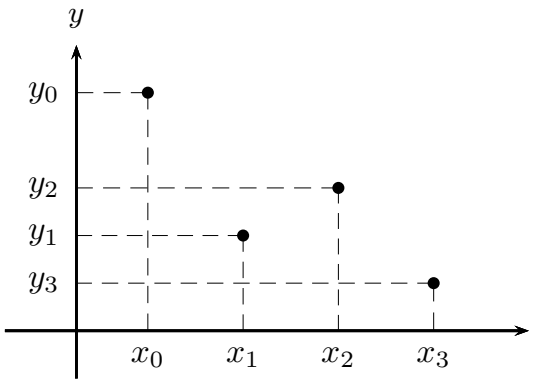



Figura 2.3: polinomio de interpolación de Newton.

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 2.4.

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 1 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:

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

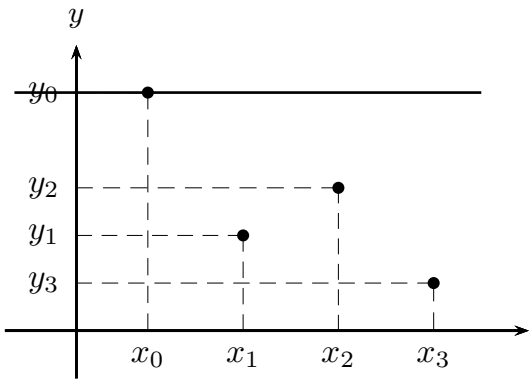



Figura 2.4: polinomio de interpolación de Newton.

expresión conocida como diferencia dividida de orden 1, y denotada como:

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

Para unificar la notación, se llama a $y_0 = f[x_0]$ como diferencia dividida de orden cero, y el polinomio de interpolación en esta 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 2.5. De igual forma, se construye un polinomio que capture otro punto más, es decir, que pase

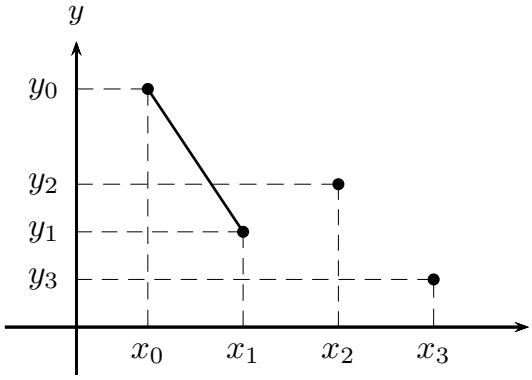



Figura 2.5: polinomio de interpolación de Newton.

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: $P_2(x_2) = f[x_0] + f[x_0, x_1](x_2 - x_0) + c_2(x_2 - x_0)(x - x_1) = y_2$. Al despejar y ordenar adecuadamente:

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

Notar que en el numerador hay diferencias divididas de orden 1 y escribiendo en notación de diferencias, se concluye:

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

expresión llamada diferencia divida de orden 2. El polinomio de interpolación en notación de diferencias divididas quedaría:

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

La gráfica de este nuevo resultado se presenta en la figura 2.6.

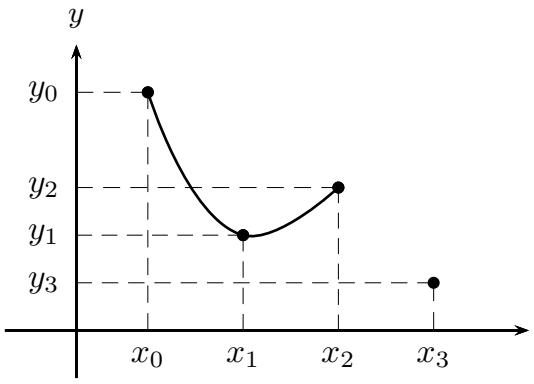



Figura 2.6: polinomio de interpolación de Newton.

Similar al procedimiento anterior, se puede inducir la forma para el polinomio de grado 3 que interpola cuatro puntos:

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

donde:

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

expandiendo las diferencias dividas de orden 2

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

y expandiendo las de diferencias de primer orden, finalmente el coeficiente $c_2$ se reduce a:

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

En la figura 2.7 se ilustra el comportamiento del polinomio resultante.

**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.

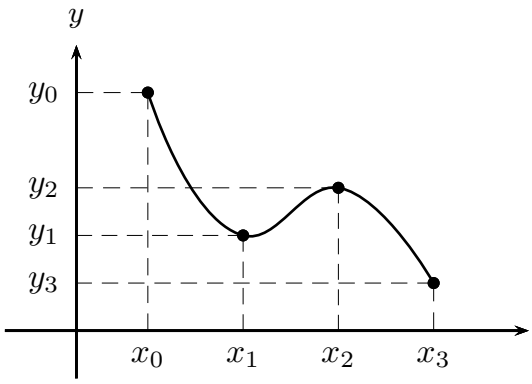



Figura 2.7: polinomio de interpolación de Newton.

| $i$ | $x_i$ | $y_i$ |
|---|---|---|
| 0 | -1 | 2 |
| 1 | 0 | -1 |
| 2 | 1 | 1 |
| 3 | 2 | -2 |

**Ejemplo 17.** Encontrar el polinomio de interpolación de Newton que ajusta los datos $(-1, 2)$, $(0, -1)$, $(1, 1)$ y $(2, -2)$.

***Solución:*** al organizar los puntos en una tabla se obtiene:

Ahora, se deben calcular las diferencias divididas de orden $0, 1, 2$ y 3. Las diferencias de orden cero $f[x_i]$, corresponden a la columna $y_i$, de donde:

| $i$ | $x_i$ | $\Delta^0$ |
|---|---|---|
| 0 | -1 | 2 |
| 1 | 0 | -1 |
| 2 | 1 | 1 |
| 3 | 2 | -2 |

Para determinar las diferencias de orden uno, $\Delta^1$, se debe calcular $f[x_{i-1}, x_i] = \frac{f[x_i]-f[x_{i-1}]}{x_i-x_{i-1}}$ para $i = 1, 2, 3$:

| $i$ | $x_i$ | $\Delta^0$ | $\Delta^1$ |
|---|---|---|---|
| 0 | -1 | 2 | |
| 1 | 0 | -1 | $f[x_0, x_1]$ |
| 2 | 1 | 1 | $f[x_1, x_2]$ |
| 3 | 2 | -2 | $f[x_2, x_3]$ |

Calculando cada $f[x_{i-1}, x_i]$ y recordando que $f[x_j]$ corresponde a $y_j$, se obtiene:

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

Tomando los valores correspondientes, se concluye:

| $i$ | $x_i$ | $\Delta^0$ | $\Delta^1$ |
|---|---|---|---|
| 0 | -1 | 2 | |
| 1 | 0 | -1 | $\frac{-1-2}{0-(-1)} = -3$ |
| 2 | 1 | 1 | $\frac{1-(-1)}{1-0} = 2$ |
| 3 | 2 | -2 | $\frac{-2-1}{2-1} = -3$ |

Ahora, se agrega una columna con las diferencias de orden dos, $\Delta^2$, que contiene los valores $f[x_{i-2}, x_{i-1}, x_i]$ para $i = 2, 3$.

| $i$ | $x_i$ | $\Delta^0$ | $\Delta^1$ | $\Delta^2$ |
|---|---|---|---|---|
| 0 | -1 | 2 | | |
| 1 | 0 | -1 | -3 | |
| 2 | 1 | 1 | 2 | $f[x_0, x_1, x_2]$ |
| 3 | 2 | -2 | -3 | $f[x_1, x_2, x_3]$ |

Como $f[x_{i-2}, x_{i-1}, x_i] = \frac{f[x_{i-1}, x_i] - f[x_{i-2}, x_{i-1}]}{x_i - x_{i-2}}$:

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

Dado que $f[x_{i-1}, x_i]$ está ubicado en la columna $\Delta^1$, entonces:

| $i$ | $x_i$ | $\Delta^0$ | $\Delta^1$ | $\Delta^2$ |
|---|---|---|---|---|
| 0 | -1 | 2 | | |
| 1 | 0 | -1 | -3 | |
| 2 | 1 | 1 | 2 | $\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{2-(-3)}{1-(-1)} = \frac{5}{2}$ |
| 3 | 2 | -2 | -3 | $\frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} = \frac{-3-2}{2-0} = \frac{-5}{2}$ |

Por último, se construye la columna $\Delta^3$, que contiene a $f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}$

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

Como $f[x_1, x_2, x_3]$ y $f[x_0, x_1, x_2]$ están en la columna $\Delta^2$, se determina que

| $i$ | $x_i$ | $y_i$ | $\Delta^1$ | $\Delta^2$ | $\Delta^3$ |
|---|---|---|---|---|---|
| 0 | -1 | 2 | | | |
| 1 | 0 | -1 | -3 | | |
| 2 | 1 | 1 | 2 | 5/2 | |
| 3 | 2 | -2 | -3 | -5/2 | $\frac{f[x_1,x_2,x_3]-f[x_1,x_2,x_3]}{x_3-x_0} = \frac{-5}{3}$ |

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 de la tabla de diferencias divididas. Por lo tanto, al reemplazar los correspondientes valores de la diagonal, se concluye que $P(x) = 2 - 3(x + 1) + \frac{5}{2}(x + 1)x - \frac{5}{3}(x + 1)x(x - 1)$. La correspondiente gráfica se presenta en figura 2.8. ♢

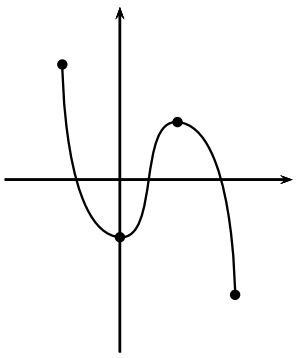


Figura 2.8: Polinomio de Newton para los datos $(-1, 2)$, $(0, -1)$, $(1, 1)$ y $(2, -2)$.

Cuando se tienen $n + 1$ puntos, el polinomio se puede escribir:

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

$$\cdots + f[x_0,\ldots,x_n](x-x_0)\cdots(x-x_{n-1}).$$

Notar que el último término tiene $n$ factores lineales, de donde el polinomio de interpolación es de a lo más grado $n$. Una conexión entre el polinomio de interpolación de Lagrange y el polinomio de Newton se establece en el siguiente teorema.

**Teorema 6** (Teorema de interpolación)**.** *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 de grado $n$ o menor $p$ y $q$ diferentes que interpolan los $n+1$ puntos, es decir:

$p(x_0) = y_0$, al igual $q(x_0) = y_0$.

$p(x_1) = y_1$, al igual $q(x_1) = y_1$.

$p(x_2) = y_2$, al igual $q(x_2) = y_2$, así sucesivamente hasta:

$p(x_n) = y_n$, al igual $q(x_n) = y_n$.

Ahora se construye el polinomio $r(x)$ como $r(x) = p(x) - q(x)$. Al evaluar el nuevo polinomio en $x_0, x_1, ..., x_n$ se obtiene:

$$\left.\begin{aligned} r(x_0) = p(x_0) - q(x_0) = y_0 - y_0 = 0 \\ r(x_1) = p(x_1) - q(x_1) = y_1 - y_1 = 0 \\ r(x_2) = p(x_2) - q(x_2) = y_2 - y_2 = 0 \\ \vdots \\ r(x_n) = p(x_n) - q(x_n) = y_n - y_n = 0 \end{aligned}\right\} n+1 \text{ raíces}$$

Observar que al sumar dos polinomios de grado $n$ (como $p$ y $q$), se espera un nuevo 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, lo que lleva a: $r(x) = 0 = p(x) - q(x)$, y por lo tanto se concluye que $p(x) = q(x)$, demostrando así la unicidad del polinomio de interpolación. □

**Nota:** los polinomios de interpolación de Lagrange y de Newton son iguales. La diferencia radica en la construcción. Si para los puntos $(x_0, y_0)$, $(x_1, y_1)$, $(x_2, y_2), \cdots, (x_n, y_n)$ se construyeran los polinomios de interpolación de Lagrange y Newton y cada uno se simplificara totalmente, entonces se observaría el mismo polinomio.

## Ejercicios 5

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. Construir el interpolador de Newton en los puntos $(-\frac{\pi}{2}, \sin(-\frac{\pi}{2}))$, $(\frac{\pi}{6}, \sin(\frac{\pi}{6}))$, $(\frac{\pi}{4}, \sin(\frac{\pi}{4}))$, $(\frac{\pi}{2}, \sin(\frac{\pi}{2}))$ y estimar la imagen de $x = 0$.

4. 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 $x_4 = 2$.

5. 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 $(5, 1)$.

6. Construir el interpolador de Newton de un conjunto de datos, si se conoce que la tabla de diferencias divididas es:

| $x$ | $y$ | $\Delta^1$ | $\Delta^2$ | $\Delta^3$ |
|---|---|---|---|---|
| -1 | 4 | | | |
| 2 | ? | 1 | | |
| 1 | ? | ? | -2 | |
| -2 | ? | ? | ? | -3 |

7. Completar la siguiente tabla de diferencias divididas:

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

8. ¿Es correcto afirmar que, al intercambiar el orden de los datos, entonces el interpolador de Newton es diferente?

9. Calcular la cantidad total de multiplicaciones y divisiones necesarias para construir el interpolador de Newton sobre un conjunto de tres datos.

10. 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 rango de los datos?

### 2.1.3. Trazadores cúbicos *(Cubic Splines)*

Un posible problema que pueden presentar los polinomios de interpolación de Lagrange y Newton es que cuando tienen que interpolar muchos puntos el grado del polinomio puede ser alto, y adicionalmente se presenten fuertes oscilaciones en puntos muy cercanos.

Una solución es ordenar los puntos, 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 (trazadores de grado cero o *splines* de grado cero), pero lo anterior no es aceptable si se desea captar el contorno de una figura o describir una función continua.

La siguiente opción es polinomios de grado uno en cada subintervalo, lo que garantiza continuidad de la función de *spline*. 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, como es el caso en la descripción del movimiento de un automóvil, donde la función aceleración es continua, es decir, el auto no cambia de un valor de aceleración a otro sin pasar por estados intermedios.

### 2.1.4. Trazadores cúbicos

Dada una lista *ordenada* de $n+1$ puntos que cumplen las hipótesis de interpolación[1], se desea construir una función a trozos

$$S(x) = \begin{cases} s_0(x) & \forall x \in [x_0, x_1] \\ s_1(x) & \forall x \in [x_1, x_2] \\ s_2(x) & \forall x \in [x_2, x_3] \\ \quad \vdots & \\ s_{n-1}(x) & \forall x \in [x_{n-1}, x_n] \end{cases} \tag{2.1}$$

donde los $s_i(x)$ son 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 $(n-1)$ debemos asegurar que los polinomios que se encuentran, lleven la misma dirección, es decir que cumpla: $s'_{i-1}(x_i) = s'_i(x_i)$. Esta condición provee $n-1$ ecuaciones.

3. **Condición de la segunda derivada**: también para nodos internos $(n-1)$ 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:

| | | |
|---|---|---|
| | $2n$ | 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.

#### 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:

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

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

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

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

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

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

por lo tanto, si $s_{i+1}'(x_{i+1}) = s_i'(x_{i+1})$ para $i = 0, 2, \ldots, n-2$ entonces:

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

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 $S''(x)$:

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

si $s_{i+1}''(x_{i+1}) = s_i''(x_{i+1})$ para $i = 0, 2, \ldots, n-2$ 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:

$$a_i = y_i \tag{2.2}$$
$$a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1} \tag{2.3}$$
$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \tag{2.4}$$
$$c_{i+1} = c_i + 3d_i h_i \tag{2.5}$$

Como se indicó al inicio de esta sección, el conjunto anterior contiene $4n-2$ ecuaciones. Es necesario hallar $4n$ coeficientes, y por tanto para equilibrar el sistema de ecuaciones, se adicionan las siguientes condiciones (denominadas *condiciones de frontera libre o natural*):

$$s_0''(x_0) = 0$$
$$s_{n-1}''(x_n) = 0$$

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

- Si se despeja $d_i$ de la ecuación (2.5), se concluye que $d_i = \frac{(c_{i+1}-c_i)}{3}$.
- Sustituyendo $d_i$ en las ecuaciones (2.3) y (2.4) se obtiene:

$$a_{i+1} = a_i + b_i h_i + \frac{h_i^2}{3}(2c_i + c_{i+1}) \tag{2.6}$$
$$b_{i+1} = b_i + h_i(c_i + c_{i+1}) \tag{2.7}$$

- Despejando $b_i$ de la ecuación (2.6) y reemplazando adecuadamente en (2.7), se obtiene:

$$h_{i-1}c_{i-1} + 2(h_{i-1}+h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1}-a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) \tag{2.8}$$

La ecuación (2.8) solamente relaciona los coeficientes $c_i$, lo que representa un sistema mas pequeño que el original. Por lo tanto, para construir el trazador cúbico de los puntos $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$, primero se debe resolver el sistema:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \frac{3}{h_2}(a_3-a_2) - \frac{3}{h_1}(a_2-a_1) \\ \\ \\ \vdots \\ \\ \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{pmatrix} \tag{2.9}$$

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

**Ejemplo 18.** Dados los datos $(-1,2)$, $(0,-1)$, $(2,2)$, $(3,2)$ y $(7,-1)$, construir su trazador cúbico libre asociado.

***Solución:*** aunque no es el caso, lo primero que se debe hacer es ordenar de menor a mayor los puntos por la coordenada $x$. Una vez ordenados, se disponen en una tabla y se calculan los elementos que conformarán el sistema de ecuaciones (2.9).

| $i$ | $x_i$ | $a_i$ | $h_i$ | $\frac{3}{h_{i-1}}(a_i - a_{i-1}) - \frac{3}{h_{i-2}}(a_{i-1} - a_{i-2})$ |
|---|---|---|---|---|
| 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 | | |

Por lo tanto, 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:

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

y al utilizar las relaciones $d_i = \frac{(c_{i+1} - c_i)}{3}$ y $b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1})$ 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 6

1. Construir el trazador cúbico para los datos $(0,1), (1,1), (2,7)$.

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

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

   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 $s'''(x)$?

## 2.2. 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 "toque" ningún punto, pero que pase "cerca" de cada uno de ellos como se muestra en la figura 2.9.

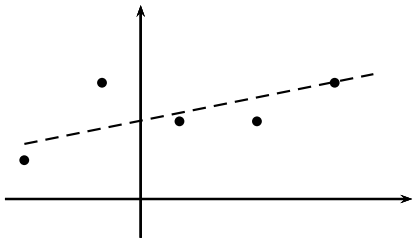


Figura 2.9: ajuste por mínimos cuadrados.

### 2.2.1. Errores

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

**Error relativo**: suponer que se tienen $n+1$ puntos y que estos se ajustan con la función $\bar{f}(x)$. El error relativo que se comete con la función de ajuste es $E_r = \sum_{i=0}^{n}(y_i - \bar{f}(x_i))$. Un problema en esta situación es de compensación, es decir, pueden existir errores grandes que al sumarlos con otros de igual magnitud pero de signo contrario, se cancelen y pareciera no ser entonces tan crítica la situación.

**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 en un punto lo hace difícil de tratar.

**Error cuadrático**: esta manera de medir el error tiene la virtud de no presentar problemas de diferenciabilidad y se define como: $E_c = \sum_{i=0}^{n}(y_i - \bar{f}(x_i))^2$.

### 2.2.2. 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*.

### 2.2.3. 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 $\{1, x\}$, obteniéndose polinomios de grado 1 (rectas). Ahora, la idea es entonces determinar una recta $a_0 + a_1 x$ que pase cerca de los puntos $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. Si se observa el error cuadrático $E$ entre la curva $\bar{f}(x) = a_0 + a_1 x$ y la lista de puntos:

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

se puede determinar que dependiendo del valor de los parámetros $a_0$ y $a_1$, la curva de ajuste comete mayor o menor error, es decir, el error depende de estos valores, por lo tanto, el error cuadrático se puede analizar como la función de dos variables

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

Así, determinar que valores $a_0$ y $a_1$ satisfacen que $E$ sea mínimo, se traduce en determinar un mínimo de la función $E(a_0, a_1)$. Notar que $E(a_0, a_1)$ es una función cuadrática, y su gráfica corresponde a un paraboloide. Luego, se tiene un único punto crítico y corresponde a su mínimo. Para calcular el punto crítico, se debe determinar cuando el gradiente de la función es el vector nulo, y así:

$$\nabla E(a_0, a_1) = \left\langle \frac{\partial E(a_0, a_1)}{\partial a_0}, \frac{\partial E(a_0, a_1)}{\partial a_1} \right\rangle$$

de donde se obtienen las dos ecuaciones (normales):

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

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

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

Al derivar término a término

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

se realiza la derivada

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

y se factoriza:

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

para obtener:

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

Por propiedades de la sumatoria se tiene que

$$\sum_{i=0}^{n} y_i - \sum_{i=0}^{n} a_0 - \sum_{i=0}^{n} a_1 x_i = 0$$

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 de $a_1$, se llega a:

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

La matriz aumentada de este sistema lineal es:

$$\left[\begin{array}{cc|c} \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} y_i x_i \\ \sum_{i=0}^{n} x_i & n+1 & \sum_{i=0}^{n} y_i \end{array}\right]$$

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

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

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

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

**Ejemplo 19.** 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 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, $\bar{f}(x) = a_0 + a_1x + a_2x^2$, el error es

$$E(a_0, a_1, a_2) = \sum_{i=0}^{n} \left(y_i - a_0 - a_1x_i - a_2x_i^2\right)^2.$$

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:

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

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

$$a_2 \sum_{i=0}^{n} x_i^4 + a_1 \sum_{i=0}^{n} x_i^2 + a_0(n+1) = \sum_{i=0}^{n} y_i x_i^2.$$

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 3, 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 $n$.

### 2.2.4. 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 $E(a, b) = \sum(y_i - ax_i^b)^2$. Aunque al calcular las ecuaciones normales rápidamente se observa que no es un sistema lineal, un buen intento por obtener un "sistema lineal" es aplicar el logaritmo de la función de ajuste y por las propiedades de estos, se obtiene:

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

Si se realizan los cambios $\bar{F}(x) = \log \bar{f}(x)$, $X_i = \log x_i$, $a_0 = \log a$ y $a_1 = b$, se tiene el caso de la recta de mínimos cuadrados

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

## Ejercicios 7

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

| $x_i$ | $y_i$ |
|---|---|
| 1 | 10 |
| 2 | 100 |
| 3 | 10000 |
| 4 | 100000 |
| 5 | 1000000 |

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

9. 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?

10. Si se conoce que la nota promedio en el curso de métodos numéricos durante los últimos años esta dada por:

| Año | Nota promedio |
|---|---|
| 2007 | 2.8 |
| 2008 | 2.8 |
| 2009 | 2.6 |
| 2010 | 3.1 |
| 2011 | 3.8 |
| 2014 | 3.1 |
| 2016 | 2.9 |

estimar, uitlizando la recta de mínimos cuadrados, la nota promedio en el año 2015.


7
6

# Capítulo 3

# Sistemas de ecuaciones

En dos de los problemas estudiados en sesiones anteriores se ha necesitado solucionar sistemas de ecuaciones lineales. En el caso de los trazadores cúbicos, en el que es necesario solucionar un sistema tridiagonal con el fin de conocer los $z_i$ 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 dicho número puede ser del orden de 1000 o 10000 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.

## 3.1. 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.

### 3.1.1. Factorización $LU$

Primero que todo, es de anotar que sistemas 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 examina en detalle dichas situaciones.

**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

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

y se puede entonces escribir la solución genérica:

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

**Nota:** observar que el sistema tiene solución única si $d_{ii} \neq 0$ para todo $i$. Luego basta asegurar que $\det D \neq 0$, donde $D$ es la matriz diagonal y su determinante se calcula como $\det D = \prod_{i=1}^{n} d_{ii}$.

**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 $x_2$:

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

y 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

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

y lo anterior 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 $i = 1,\ 2,\ 3, \ldots, n$. Observar que para $i = 1$ la suma va desde 1 hasta 0, lo que se interpretará como 0, 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 y única si $l_{ii} \neq 0$ para toda $i = 1,\ 2,\ 3, \ldots, n$. 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

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

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 8

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

**Factorización** *LU*

Aunque para sistemas $n \times n$ no triangulares el problema no es aparentemente fácil de resolver, si la matriz de coeficientes $A$ se puede expresar como el producto de una triangular inferior $L$ con una triangular superior $U$, es decir $A = LU$, entonces el siguiente algoritmo permite reducir el problema a las técnicas de la sección 3.1.1.

$$1)\ \ Lz = b$$

$$2)\ \ Ux = z$$

Notar que en la primera línea el algoritmo de sustitución progresiva permite calcular completamente el vector $z$, mientras que en el segundo renglón, una vez es conocido $z$, con sustitución regresiva se puede

calcular completamente el vector $x$ y el problema quedaría solucionado. Es entonces 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 20.** Determinar una factorización $LU$ de la matriz

$$\begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

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

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

Luego $l_{11}u_{11} = 2$, si se define que $l_{11} = 1$, entonces $u_{11} = 2$. Por otro lado $l_{11}u_{12} = -1$ y $l_{11}u_{13} = 1$, y por tanto $u_{12} = -1$ y $u_{13} = 1$.

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

Ahora, $l_{21}u_{11} = 3$ y $l_{31}u_{11} = 3$, de donde al despejar se tiene $l_{21} = \frac{3}{2}$ y $l_{31} = \frac{3}{2}$.

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & l_{22} & 0 \\ \frac{3}{2} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

Continuando con el elemento $a_{22}$, se tiene $\frac{3}{2}(-1) + l_{22}u_{22} = 3$, y si se define $l_{22} = 1$, se obtiene $u_{22} = 3 + \frac{3}{2} = \frac{9}{2}$. Adicionalmente $\frac{3}{2}(-1) + l_{32}u_{22} = 3$ y $\frac{3}{2}(1) + l_{22}u_{23} = 9$, de donde se concluye que $l_{32} = \frac{9}{2}$ y $u_{23} = \frac{15}{2}$.

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{3}{2} & \frac{9}{2} & l_{33} \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & \frac{9}{2} & \frac{15}{2} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

Por último, $\frac{3}{2}(1) + \frac{9}{2}(\frac{15}{2}) + l_{33}u_{33} = 5$, y si se define $l_{33} = 1$, se obtiene $u_{33} = 5 - \frac{3}{2} - \frac{135}{4} = -\frac{121}{4}$.

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{3}{2} & \frac{9}{2} & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & \frac{9}{2} & \frac{15}{2} \\ 0 & 0 & -\frac{121}{4} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

Así,

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{3}{2} & \frac{9}{2} & 1 \end{pmatrix} \qquad U = \begin{pmatrix} 2 & -1 & 1 \\ 0 & \frac{9}{2} & \frac{15}{2} \\ 0 & 0 & -\frac{121}{4} \end{pmatrix}$$

♢

El anterior procedimiento se puede resumir de la siguiente manera informal:

**Primer paso.** Calcular los elementos $l_{ii}$ y $u_{ii}$. Multiplicar la fila $i$-ésima de la matriz $L$ y la columna $i$-ésima de la matriz $U$ e igualar el resultado con $a_{ii}$. En este paso es necesario definir un valor particular para $l_{ii}$ o $u_{ii}$; en el ejemplo se tiene $l_{ii} = 1$.

**Segundo paso.** Calcular todos los elementos de la columna $i$ de la matriz $L$.

**Tercer paso.** Calcular todos los elementos de la fila $i$ de la matriz $U$.

Como se puede observar, en cada ciclo del algoritmo se avanza una columna de la matriz $L$ desde la izquierda, y se avanza en una fila de la matriz $U$ desde la primera. De esta forma, cuando se va a calcular el ciclo $r$, se conocen completamente todos los elementos $l_{ij}$ para $i$ desde 1 hasta $n$, y $j$ desde 1 hasta $r-1$ y también $u_{ij}$ donde $i$ va desde 1 hasta $r-1$ y $j$ desde 1 hasta $n$. De manera detallada los pasos son:

**Primer paso.**

$$a_{rr} = \sum_{k=1}^{r} l_{ir} u_{rj}$$

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

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

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. Elecciones famosas:

a) Factorización de Doolittle: $l_{ii} = 1$ para $i = 1,\ 2, \ldots, n$.

b) Factorización de Crout: $u_{ii} = 1$ para $i = 1,\ 2, \ldots, n$.

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

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

en cambio, al trabajar con la factorización de Crout:

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

**Segundo paso.** Cálculo de la columna $r$ de $L$. El siguiente elemento a calcular es $l_{r+1r}$, después $l_{r+2,r}$, hasta finalizar con $l_{n,r}$. 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.** Cálculo de la fila $r$ de $U$. 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 la $L$ desde la primera columna hasta la columna $r$, al igual que todos los elementos de la $U$ 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 21.** 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 $Ax = b$, donde $b = (-1, -6, -13)^t$.

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

- $Lz = b$
- $Ux = z$

Luego, se debe solucionar en primer lugar el problema $Lz = b$, es decir

$$\begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -2 & 4 & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_{31} \end{pmatrix} = \begin{pmatrix} -1 \\ -6 \\ -13 \end{pmatrix}$$

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 $z = (-1, -3, -3)^t$. Ahora, se debe solucionar el problema $Ux = z$:

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

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 $x = (0, 1, -1)^t$ como la solución al problema $Ax = b$ inicial. ♢

En el ejemplo anterior, aunque los cálculos utilizados en la sustitución progresiva y regresiva son de un costo computacional relativamente "barato", es en el cálculo de la factorización $LU$ donde el anterior método puede no ser tan eficiente. Por tanto, la utilización de este método para solucionar sistemas de ecuaciones es útil cuando se conoce de antemano una factorización de la matriz de coeficientes, y no es parte de nuestro trabajo calcularla, dado que en caso contrario los costos computacionales son equivalentes a emplear un método tradicional como eliminación de Gauss-Jordan.

## Ejercicios 9

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

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

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

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

   *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 $Ax = b$ donde $b = (1, 0, 1)^t$ y $A$ corresponde a las siguientes matrices.

$$a)\ \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix}\begin{pmatrix} 4 & 3 & 2 \\ 0 & 2 & 6 \\ 0 & 0 & 3 \end{pmatrix} \qquad b)\ \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -7 & 9 & 1 \end{pmatrix}\begin{pmatrix} 5 & 0 & -3 \\ 0 & 2 & 1 \\ 0 & 0 & -4 \end{pmatrix}$$

4. Si se conoce que $LU$ es una factorización de la matriz $A$ y $\det(U) = 0$, demostrar que si el problema $Ax = b$ tiene solución, entonces existe $b_i$ tal que $b_i = 0$.

5. Determinar la factorización $LU$ de las siguientes matrices tomando como $U = L^t$. Este hecho es conocido como descomposición de Cholesky.

$$a)\ \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \qquad c)\ \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$$

$$b)\ \begin{pmatrix} 2 & -1 & 0 \\ -1 & 4 & 2 \\ 0 & 2 & 2 \end{pmatrix} \qquad d)\ \begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix}$$

## 3.2. Métodos iterativos

Cuando se tiene un sistema de ecuaciones $Ax = b$ y se desea determinar el valor de $x$ que lo satisface, existen diferentes técnicas, como el método de Gauss-Jordan, matriz inversa, regla de Cramer, factorización $LU$, etc. que permiten encontrar de manera exacta este valor, aunque con un costo computacional alto.

Ahora, si para determinados problemas es suficiente obtener un $\widetilde{x}$ que no satisface de manera exacta el sistema $Ax = b$, pero el resultado de $A\widetilde{x}$ está "cerca" de $b$ y el costo de calcular $\widetilde{x}$ es menor al de calcular el valor exacto $x$, entonces se desearía construir $\widetilde{x}$. En esta sección se desarrollan algunos métodos para calcular $\widetilde{x}$.

Si se observa que la solución del problema $Ax = b$ corresponde a la solución de la ecuación[1] $f(x) = 0$ donde $f(x) = Ax - b$, entonces aunque surge la intención de utilizar los métodos descritos en el primer capitulo del libro, es necesario primero revisar algunos conceptos de la teoría de matrices.

### 3.2.1. Normas vectoriales

En el curso de álgebra lineal se define la norma de un vector $u = (x, y, z)$ como:

$$\|u\|_2 = \sqrt{x^2 + y^2 + z^2}$$

y se denomina *norma euclidiana*. Una de las aplicaciones de la norma euclidiana es la definición de *distancia euclidiana* entre dos vectores $u$ y $v$:

$$d_2(u, v) = \|u - v\|_2$$

**Ejemplo 22.** Determinar la distancia entre los vectores $u = (1, -1, 2)$ y $v = (2, -3, -6)$.

[1] Notar que 0 se refiere al vector cero y que $f$ es una función vectorial, es decir, $f : \mathbb{R}^n \to \mathbb{R}^n$.

***Solución:***

$$\begin{aligned} d_2(u,v) &= \|u-v\|_2 \\ &= \|(1,-1,2)-(2,-3,-6)\|_2 \\ &= \|(-1,2,8)\|_2 \\ &= \sqrt{(-1)^2+2^2+8^2} \\ &= \sqrt{1+4+64} = \sqrt{69} \end{aligned}$$

♢

En la siguiente definición se busca generalizar la idea de norma euclidiana y construir nuevas "distancias" sobre los vectores.

**Definición 3.** *Una* ***norma vectorial*** *en $\mathbb{R}^n$ es una función $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ tal que:*

*i)* $\|x\| \geq 0$ *para todo* $x \in \mathbb{R}^n$

*ii)* $\|x\| = 0$ *si y sólo si* $x = 0$

*iii)* $\|\alpha x\| = |\alpha|\|x\|$ *para todo* $x \in \mathbb{R}^n$ *y* $\alpha \in \mathbb{R}$

*iv)* $\|x+y\| \leq \|x\| + \|y\|$ *para todo* $x, y \in \mathbb{R}^n$ *(Desigualdad triangular)*

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 $l_\infty$, que presentan ventajas computacionales para procedimientos iterativos.

**Definición 4.** *Sea $x = (x_1, x_2, \ldots, x_n)$ un vector en $\mathbb{R}^n$, entonces:*

- *La norma $l_1$ del vector $x$ es:* $\|x\|_1 = \sum_{i=1}^{n} |x_i|$
- *La norma $l_\infty$ del vector $x$ es:* $\|x\|_\infty = \max\limits_{1\leq i\leq 1} |x_i|$

**Ejemplo 23.** Sea $x = (-1, 2, -5, 3)$ entonces:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| = |-1| + |2| + |-5| + |3| = 1 + 2 + 5 + 3 = 11$$

$$\|x\|_\infty = \max_{1\leq i\leq 1} |x_i| = \max\{|-1|, |2|, |-5|, |3|\} = 5$$

A partir de las normas $l_1$ y $l_\infty$ se construyen las distancias $d_1(u,v) = \|u-v\|_1$ y $d_\infty(u,v) = \|u-v\|$.

Dado que en esta sección se desea desarrollar métodos *iterativos* para aproximar la solución del sistema $Ax = b$, en los cuales se parte de un vector inicial (semilla) $x^{(0)}$ y se construyen vectores $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$ los cuales se "aproximan" al vector $x$ solución de la ecuación $Ax = b$, es necesario definir cuando una sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ de vectores de $\mathbb{R}^n$ *convergen* a un vector $x$.

**Definición 5.** *Sea $\{x^{(k)}\}_{k=0}^{\infty}$ una sucesión de vectores de $\mathbb{R}^n$ y $x$ un vector de $\mathbb{R}^n$. Entonces la sucesión converge a $x$ respecto a la norma $\|\cdot\|$ si dado $\epsilon > 0$, existe un entero positivo $N$ tal que*

$$\|x^{(k)} - x\| < \epsilon \qquad \text{para todo } k \geq N$$

Desde un aspecto computacional, la definición anterior no es la mejor, por lo tanto se presenta el siguiente teorema.

**Teorema 7.** *La sucesión* $\{x^{(k)}\}_{k=0}^{\infty}$ *converge a* $x$ *respecto a la norma* $l_\infty$ *si y solo si* $\lim_{k\to\infty} x_i^{(k)} = x_i$ *para cada* $i = 1, 2, \ldots, n$.

Se debe anotar que el teorema 7 indica una manera de comprobar si una sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ converge a un vector $x$ con *relación* a la norma $l_\infty$ y no con relación a cualquier norma.

**Ejemplo 24.** Demostrar que la sucesión $x^{(k)} = \left(e^{-k}, 5 + \frac{1}{k}, \frac{-3\sin(k)}{k}\right)$ converge a el vector $(0, 5, 0)$ con relación a la norma $l_\infty$.

***Solución:*** utilizando el teorema 7 es suficiente demostrar que

- $\lim\limits_{k\to\infty} e^{-k} = 0$
- $\lim\limits_{k\to\infty} 5 + \frac{1}{k} = 5$
- $\lim\limits_{k\to\infty} \frac{-3\sin(k)}{k} = 0$

En este caso,

- $\lim\limits_{k\to\infty} e^{-k} = \lim\limits_{k\to\infty} \frac{1}{e^k} = 0$,
- $\lim\limits_{k\to\infty} 5 + \frac{1}{k} = \lim\limits_{k\to\infty} 5 + \lim\limits_{k\to\infty} \frac{1}{k} = 5 + 0 = 5$ y
- $\lim\limits_{k\to\infty} \frac{-3\sin(k)}{k} = -3 \lim\limits_{k\to\infty} \frac{\sin(k)}{k} = -3(0) = 0$,

con lo cual se demuestra que la sucesión $\{x^{(k)}\}_{k=1}^{\infty}$ converge a el vector $(0, 5, 0)$ con relación a la norma $l_\infty$. ♢

Puede demostrarse el siguiente resultado: *"Si una sucesión* $\{x^{(k)}\}_{k=1}^{\infty}$ *converge a* $x$ *con relación a una norma en* $\mathbb{R}^n$*, entonces converge a* $x$ *con relación a cualquier otra norma de* $\mathbb{R}^n$*"*. Este hecho permite concluir que si una sucesión $\{x^{(k)}\}_{k=1}^{\infty}$ converge a $x$ con relación a la norma $l_\infty$, entonces converge a $x$ con relación a la norma euclidiana y la norma $l_1$.

### 3.2.2. Normas matriciales

En la construcción y estudio de los métodos iterativos para aproximar una solución del sistema de ecuaciones $Ax = b$ es necesario definir la distancia entre matrices, donde una de las maneras más efectivas es definir una norma sobre espacios de matrices.

**Definición 6.** *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*

*i)* $\|A\| \geq 0$ *para toda* $A \in \mathcal{M}_n(\mathbb{R})$

*ii)* $\|A\| = 0$ *si y solo si* $A = 0$

*iii)* $\|\alpha A\| = |\alpha|\|A\|$ *para toda* $A \in \mathcal{M}_n(\mathbb{R})$ *y* $\alpha \in \mathbb{R}$

*iv)* $\|A + B\| \leq \|A\| + \|B\|$ *para toda* $A, B \in \mathcal{M}_n(\mathbb{R})$

*v)* $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ *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 $l_\infty$.

**Definición 7.** *Si* $A = [a_{ij}]_{n\times n}$ *es una matriz, entonces:*

- $\|A\|_\infty = \max\limits_{1\leq i\leq n} \left\{ \sum\limits_{j=1}^{n} |a_{ij}| \right\}$
- $\|A\|_1 = \max\limits_{1\leq j\leq n} \left\{ \sum\limits_{i=1}^{n} |a_{ij}| \right\}$

**Ejemplo 25.** Si

$$A = \begin{pmatrix} -5 & 3 & 1 \\ 3 & 7 & 8 \\ 3 & 8 & 1 \end{pmatrix}$$

calcular $\|A\|_\infty$.

***Solución:*** dado que $\|A\|_\infty = \max\limits_{1\leq i\leq n} \left\{ \sum\limits_{j=1}^{n} |a_{ij}| \right\}$, entonces

$$\|A\|_\infty = \max\left\{ \sum_{j=1}^{n} |a_{1j}|, \sum_{j=1}^{n} |a_{2j}|, \sum_{j=1}^{n} |a_{3j}| \right\}$$

De otro lado, $\sum_{j=1}^{n} |a_{1j}| = |-5| + |3| + |1| = 9$, $\sum_{j=1}^{n} |a_{2j}| = |3| + |7| + |8| = 18$ y $\sum_{j=1}^{n} |a_{3j}| = |3| + |8| + |1| = 12$, y por tanto $\|A\|_\infty = \max\{9, 18, 12\} = 18$. ♢

**Ejemplo 26.** Si

$$A = \begin{pmatrix} -5 & 3 & 1 \\ 3 & 7 & 8 \\ 3 & -3 & 6 \end{pmatrix}$$

calcular $\|A\|_1$.

***Solución:*** dado que $\|A\|_\infty = \max\limits_{1\leq j\leq n} \left\{ \sum\limits_{i=1}^{n} |a_{ij}| \right\}$, entonces

$$\|A\|_\infty = \max\left\{ \sum_{i=1}^{n} |a_{i1}|, \sum_{i=1}^{n} |a_{i2}|, \sum_{i=1}^{n} |a_{i3}| \right\}$$

pero $\sum_{i=1}^{n} |a_{i1}| = |-5|+|3|+|3| = 11$, $\sum_{i=1}^{n} |a_{i2}| = |3|+|7|+|-3| = 13$ y $\sum_{i=1}^{n} |a_{i3}| = |1|+|8|+|6| = 15$, de donde $\|A\|_1 = \max\{11, 13, 15\} = 15$. ♢

## Ejercicios 10

1. Determinar la norma euclidiana, $l_\infty$, y $l_1$ para los siguientes vectores.

*a)* $u = (-1, 2, -1)$
*b)* $v = (-5, 3, 1)$
*c)* $w = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, \frac{1}{128})$
*d)* $z = (e^{-1}, e^{2}, e^{-3}, \ldots, e^{-7})$

2. Dados $u = (\sqrt{2}, e^{-1}, \pi, 1)$ y $w = (1.4, 0.5, 3.14, 0.9999)$, determinar $d_2(u, w)$, $d_\infty(u, w)$ y $d_1(u, w)$.

3. Demostrar que las sucesiones $\{x^{(k)}\}_{k=1}^{\infty}$ convergen a el vector $(-1, 0, e, 1)$ con relación a la norma $l_\infty$.

*a)* $x^k = \left(-1, \frac{1}{k}, e, \frac{k}{k+1}\right)$
*b)* $x^k = \left(\frac{k^2}{1-k^2}, \frac{\cos(k)}{k}, (1+\frac{1}{n})^n, \frac{x+1}{x}\right)$
*c)* $x^k = \left(-1 + e^{-k}, \frac{-1}{k^2}, e, 1 + \frac{x}{x^3+1}\right)$
*d)* $x^k = \left(-1 + \frac{x}{x^4-1}, 0, e, 1\right)$

4. 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)* $\begin{pmatrix} 2 & -1 & 4 & 3 \\ -1 & 8 & 5 & -6 \\ 4 & 5 & 3 & -8 \\ -11 & 15 & 13 & 1 \end{pmatrix}$

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

### 3.2.3. Métodos iterativos en la solución de sistemas de ecuaciones lineales

Como se menciona con anterioridad, los métodos iterativos en la solución de sistemas de ecuaciones $Ax = b$ construyen una sucesión $\{x^k\}_{k=0}^{\infty}$ 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 $x^{(k+1)} = Tx^{(k)} + c$, donde $T$ es una matriz fija y $c$ un vector. En lo que sigue se presentan dos métodos iterativos clásicos, el método de Jacobi y el método de Gauss-Seidel.

#### Método de Jacobi

El método de Jacobi se basa en el método del punto fijo, y para entenderlo se analiza el siguiente ejemplo.

**Ejemplo 27.** Sea $Ax = b$ el sistema

$$\begin{aligned} 5x_1 - x_2 + x_3 + x_4 &= 5 \\ x_1 + 4x_2 + x_4 &= 2 \\ 3x_1 - x_2 + 6x_3 &= -3 \\ x_2 - x_3 + 3x_4 &= 4 \end{aligned}$$

Si de la primera ecuación se despeja $x_1$, de la segunda ecuación se despeja $x_2$, y así sucesivamente, se obtiene:

$$\begin{aligned} x_1 &= \frac{5 + x_2 - x_3 - x_4}{5} \\ x_2 &= \frac{2 - x_1 - x_4}{4} \\ x_3 &= \frac{-3 - 3x_1 + x_2}{6} \\ x_4 &= \frac{4 - x_2 + x_3}{3} \end{aligned} \tag{3.1}$$

Tomando ahora $x^{(0)} = (-1, 5, 3, 2)$ y sustituyendo en el lado derecho del sistema 3.1 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 $x^{(1)} = \left(1, \frac{1}{4}, \frac{5}{6}, \frac{2}{3}\right)$, y repitiendo el proceso ahora con $x^{(1)}$ se desprende que:

$$\begin{aligned} x_1^{(2)} &= \frac{5 + \frac{1}{4} - \frac{5}{6} - \frac{2}{3}}{5} = \frac{3}{4} \\ x_2^{(2)} &= \frac{2 - 1 - \frac{2}{3}}{4} = \frac{1}{12} \\ x_3^{(2)} &= \frac{-3 - 3 + \frac{1}{4}}{6} = -\frac{23}{24} \\ x_4^{(2)} &= \frac{4 - \frac{1}{4} + \frac{5}{6}}{3} = \frac{55}{36} \end{aligned}$$

de donde $x^{(2)} = \left(\frac{3}{4}, \frac{1}{12}, \frac{23}{24}, \frac{55}{36}\right)$. En el cuadro 3.1 se encuentran las primeras diez iteraciones del método.

Se puede observar que $\|x^{(9)} - x^{(10)}\|_\infty = \text{máx}\{|0,9995 - 0,9997|, |-0,0001 + 0,0000|, |-0,9993 + 0,9997|, |1,0002 - 1,0002|\} = 0,0004$ y en cada iteración esta distancia disminuye. Si se resuelve el sistema con el método Gauss-Jordan, se obtiene que la solución es $x = (1, 0, -1, 1)$ 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 $Ax = b$ se puede resumir, de manera muy simple, en los siguientes pasos:

| $k$ | 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 3.1: iteraciones del método de Jacobi.

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 $x^{(0)}$ inicial.
3. Sustituir el vector seleccionado en las ecuaciones del paso 1 y llamar a este resultado $x^{(1)}$.
4. Repetir el paso 3 con $x^{(1)}$.

Aunque en la descripción anterior no se hace referencia a cuándo detener el proceso, para lo anterior se debe establecer un valor $\epsilon > 0$ (tolerancia) y considerar como criterio de parada $\|x^{(k)} - x^{(k-1)}\|_\infty < \varepsilon$, de donde el proceso termina cuando la distancia entre los vectores $x^{(k)}$ y $x^{(k-1)}$ es menor a $\varepsilon$.

**Ejemplo 28.** 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 $\varepsilon = 10^{-3}$.

***Solución:*** los pasos son:

**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 $x^{(0)}$. En este caso $x^{(0)} = (1, -2, 3)$.

**Paso 3.** Sustituir $x^{(0)}$ en las ecuaciones del paso 1.

$$\begin{aligned} x_1^{(1)} &= \frac{-3 - 2x_2^{(0)} + x_3^{(0)}}{4} = \frac{-3 - 2(-2) + 3}{4} = 1 \\ x_2^{(1)} &= \frac{-9 + 2x_1^{(0)} - 2x_3^{(0)}}{5} = \frac{-9 + 2(1) - 2(3)}{5} = -\frac{13}{5} \\ x_3^{(1)} &= \frac{10 + 6x_1^{(0)} - 3x_2^{(0)}}{7} = \frac{10 + 6(1) - 3(-2)}{7} = \frac{22}{7} \end{aligned}$$

luego $x^{(1)} = \left(1, \frac{13}{5}, \frac{22}{7}\right)$ y $\|x^{(0)} - x^{(1)}\| = 0.6$. Dado que la distancia no es menor a $10^{-3}$, continuar el proceso.

**Paso 4.** Repetir el paso 3, ahora con $x^{(1)}$.

$$x_1^{(2)} = \frac{-3 - 2x_2^{(1)} + x_3^{(1)}}{4} = \frac{-3 - 2(\frac{13}{5}) + \frac{22}{7}}{4} = 1.3357$$
$$x_2^{(2)} = \frac{-9 + 2x_1^{(1)} - 2x_3^{(1)}}{5} = \frac{-9 + 2(1) - 2(\frac{22}{7})}{5} = -2.6571$$
$$x_3^{(2)} = \frac{10 + 6x_1^{(1)} - 3x_2^{(1)}}{7} = \frac{10 + 6(1) - 3(\frac{13}{5})}{7} = 3.4$$

luego $x^{(2)} = (1.3357, -2.6571, 3.4)$ y $\|x^{(1)} - x^{(2)}\| = 0.3357$. Dado que la distancia no es menor a $10^{-3}$, continuar el proceso.

En el cuadro 3.2 se encuentran las primeras 19 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 | 6,8e-5 |

Cuadro 3.2: iteraciones del método de Jacobi.

Dado que $d_\infty\left(x^{(18)}, x^{(19)}\right) = 6.8\text{e-}05 < 10^{-3}$, entonces el proceso se detiene y la aproximación de la solución es $x^{(19)} = (1.6249, -2.7499, 3.9998)$. Se puede verificar que la solución exacta es $x = (1.625, -2.75, 4)$. ♢

### 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 $x^{(k-1)}$ 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 $Ax = b$. 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 29.** Utilizar el método de Gauss-Seidel resolver el siguiente sistema de ecuaciones.

$$\begin{aligned} 5x_1 - x_2 + 3x_3 &= 1 \\ -x_1 - 5x_2 + x_4 &= -9 \\ 2x_2 + 8x_3 - 2x_4 &= -4 \\ 4x_1 - x_3 - 6x_4 &= 9 \end{aligned}$$

***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 $x^{(0)}$. En este caso $x^{(0)} = (0, 0, 0, 0)$.

**Paso 3.** Calcular $x^{(1)}$.

$$x_1^{(1)} = \frac{1 + x_2^{(0)} - 3x_3^{(0)}}{5} = \frac{1 + 0 - 3(0)}{5} = \frac{1}{5}$$
$$x_2^{(1)} = \frac{-9 + x_1^{(1)} - x_4^{(0)}}{-7} = \frac{-9 + \frac{1}{5} - 0}{-7} = \frac{44}{35}$$
$$x_3^{(1)} = \frac{-4 - 2x_2^{(1)} + 2x_4^{(0)}}{8} = \frac{-4 - 2(\frac{44}{35}) + 2(0)}{8} = -\frac{57}{70}$$
$$x_4^{(1)} = \frac{9 - 4x_1^{(1)} - x_3^{(1)}}{-6} = \frac{-9 - 4(\frac{1}{5}) - (-\frac{57}{70})}{-6} = -\frac{631}{420}$$

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 $x_i^{(1)}$, se utiliza la información más reciente.

**Paso 4.** Calcular $x^{(2)}$.

Dado que $x^{(1)} = (\frac{1}{5}, \frac{44}{35}, -\frac{57}{70}, -\frac{631}{420}) = (0.2, 1.2571, -0.8142, -1.5023)$, 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 $x^{(2)} = (0.94, 0.9368, -1.1097, -1.0582)$

En el cuadro 3.3 se encuentran las primeras seis iteraciones del método. Se puede verificar que la solución exacta corresponde a $x = (1, 1, -1, -1)$ ♢

| $k$ | 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 3.3: 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 $x^{(k)}$. El método de Jacobi emplea únicamente la información contenida en el vector $x^{(k-1)}$, mientras que el método de Gauss-Seidel utiliza la información más reciente al usar las componentes de $x^{(k)}$ que ya calculadas. Al igual que el método de Jacobi, el método de Gauss-Seidel se establece como criterio de parada $\|x^{(k-1)} - x^{(k)}\| < \varepsilon$, para $\varepsilon > 0$ fijo durante la ejecución del método.

**Ejemplo 30.** 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 $\varepsilon = 10^{-3}$.

***Solución:*** los pasos son:

**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 $x^{(0)}$. En este caso $x^{(0)} = (1, -2, 3)$.

**Paso 3.** Sustituir $x^{(0)}$ en las ecuaciones del paso 1.

$$\begin{aligned} x_1^{(1)} &= \frac{-3 - 2x_2^{(0)} + x_3^{(0)}}{4} = \frac{-3 - 2(-2) + 3}{4} = 1 \\ x_2^{(1)} &= \frac{-9 + 2x_1^{(1)} - 2x_3^{(0)}}{5} = \frac{-9 + 2(1) - 2(3)}{5} = -2,6 \\ x_3^{(1)} &= \frac{10 + 6x_1^{(1)} - 3x_2^{(1)}}{7} = \frac{10 + 6(1) - 3(-2.6)}{7} = 3,4 \end{aligned}$$

luego $x^{(1)} = (1, -2.6, 3.4)$ y $\|x^{(0)} - x^{(1)}\| = 0,6$. Dado que la distancia no es menor a $10^{-3}$, entonces sigue el proceso.

**Paso 4.** Repetir el paso 3, ahora con $x^{(1)}$.

$$x_1^{(2)} = \frac{-3 - 2x_2^{(1)} + x_3^{(1)}}{4} = \frac{-3 - 2(1) + 3.4}{4} = 1.4$$
$$x_2^{(2)} = \frac{-9 + 2x_1^{(2)} - 2x_3^{(1)}}{5} = \frac{-9 + 2(1.4) - 2(3.4)}{5} = -2.6$$
$$x_3^{(2)} = \frac{10 + 6x_1^{(2)} - 3x_2^{(2)}}{7} = \frac{10 + 6(1.4) - 3(-2.6)}{7} = 3.7428$$

luego $x^{(2)} = (1.4, -2.6, 3.7428)$ y $\|x^{(1)} - x^{(2)}\| = 0.4$. Por lo tanto, la distancia entre $x^{(1)}$ y $x^{(2)}$ no es menor a $10^{-3}$, entonces sigue el proceso.

En el cuadro 3.4 se encuentran las primeras 14 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 | 5,63e-05 |

Cuadro 3.4: iteraciones del método de Gauss-Seidel.

Como $d_\infty\left(x^{(13)}, x^{(14)}\right) = 5.63\text{e-}05 < 10^{-3}$, entonces el proceso se detiene y la aproximación de la solución es $x^{(14)} = (1.6249, -2.7499, 3.9999)$. ♢

**Nota:** si el anterior ejemplo se desarrolla con el método de Jacobi, son necesarias 19 iteraciones. Se observa entonces como el método de Gauss-Seidel presenta una mejor convergencia en término de iteraciones.

## Ejercicios 11

1. Obtener $x^{(3)}$ en el método de Jacobi en la solución de los siguientes sistemas lineales. Usar $x^{(0)} = (0, 0, 0)$.

*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 $\varepsilon = 10^{-4}$, resolver los ejercicios del numeral 1 utilizando el método de Jacobi.

4. Repetir el ejercicio anterior utilizando el método de Gauss-Seidel.

**Convergencia de los métodos iterativos de Jacobi y Gauss-Seidel**

En esta parte se analizan condiciones para asegurar la convergencia de los métodos iterativos de la forma $x^{(k)} = Tx^{(k-1)} + c$. Antes de presentar estas condiciones, es necesario revisar algunos conceptos.

**Definición 8.** *Sea $A$ una matriz $n \times n$ entonces $\rho(A)$, denominado el radio espectral, corresponde a:*

$$\rho(A) = \text{máx}\{|\lambda| \mid \lambda \text{ es valor propio de } A\}$$

**Definición 9.** *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 31.** Dadas las matrices

$$A = \begin{pmatrix} -6 & 2 & 3 \\ -3 & 7 & 1 \\ 4 & 5 & 10 \end{pmatrix} \quad \text{y} \quad B = \begin{pmatrix} -2 & 1 & 0 \\ 5 & 7 & -3 \\ 9 & -1 & 10 \end{pmatrix}$$

$A$ 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 estos elementos es posible presentar los siguientes resultados.

**Teorema 8.** *Dado cualquier $x^{(0)}$ vector de $\mathbb{R}^n$, la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ generada por el esquema recursivo*

$$x^{(k)} = Tx^{(k-1)} + c$$

*converge a la única solución del problema $x = Tx + c$ 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 $x^{(k)} = Tx^{(k-1)} + c$ con $T$ una matriz fija y $c$ 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 $x^{(k)} = Tx^{(k-1)} + c$.

Si $Ax = b$ es un sistema de ecuaciones lineales con la siguiente matriz de coeficientes $A$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

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 $Ax = b$ se transforma en el sistema $(D - L - U)x = b$ lo que implica que

$$Dx = (L + U)x + b$$

Ahora, si $D^{-1}$ existe[2] entonces multiplicando ambos lados de la anterior ecuación se obtiene:

$$x = D^{-1}(L + U)x + D^{-1}b$$

que a su vez, se transforma en el esquema iterativo

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b$$

Si se define $D^{-1}(L+U) = J$ y $c = D^{-1}b$, entonces el anterior esquema corresponde a la forma matricial del método de Jacobi:

$$x^{(k)} = Jx^{(k-1)} + c \tag{3.2}$$

Ahora, si la ecuación

$$(D - L - U)x = b$$

se despeja de la siguiente manera

$$(D - L)x = Ux + b$$

y existe $(D - L)^{-1}$, entonces

$$x = (D - L)^{-1}Ux + (D - L)^{-1}b$$

lo cual origina el método matricial de Gauss-Seidel

$$x^{(k)} = Gx^{(k-1)} + s \tag{3.3}$$

donde $G = (D - L)^{-1}U$ y $s = (D - L)^{-1}b$. 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 $x^{(k)} = Tx^{(k-1)} + c$, 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 $x^{(0)}$, los método de Jacobi y Gauss-Seidel son convergentes a la única solución del sistema $Ax = b$.*

## Ejercicios 12

1. Dado el sistema de ecuaciones $Ax = b$, 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 $c$ usados en el método de Jacobi.
2. Dado el sistema de ecuaciones $Ax = b$, 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 $s$ usados en el método de Gauss-Seidel.
3. Determinar si las siguientes matrices son de diagonal estrictamente dominante.

[2] $D^{-1}$ para una matriz diagonal $D$ existe si y sólo si $a_{ii} \neq 0$ para $i = 1, 2, \ldots, n$.

*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 $x^{(0)}$.

*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 $\epsilon = 10^{-4}$.

$$\begin{aligned} 5x + 3y - z &= 2 \\ -x + 2y - 4z &= -2 \\ 2x - 4y - z &= -1 \end{aligned}$$

7. Resolver el sistema lineal dado por medio del método de Gauss-Seidel con $\epsilon = 10^{-4}$.

$$\begin{aligned} 2x + 5y + z &= -2 \\ 3x - y + z &= 2 \\ -x - y + 3z &= 0 \end{aligned}$$

8. Dada la matriz $A = \begin{pmatrix} \alpha & 1 & 0 \\ \beta & 3 & 2 \\ 1 & -1 & 3 \end{pmatrix}$, encontrar los valores de $\alpha$ y $\beta$ de forma tal que la matriz sea de diagonal estrictamente dominante.


LAMPE

# Capítulo 4



# Ecuaciones diferenciales

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 conocemos que la velocidad de un automóvil (figura 4.1) está 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 que permite modelar esta situación es

$$\begin{aligned} v = \frac{dx(t)}{dt} &= 2 - 0.2t \\ x(0) &= 50 \end{aligned} \tag{4.1}$$

donde $x(t)$ es la función de posición del automóvil.

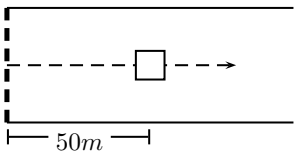



Figura 4.1: una representación de la situación del automóvil.

La ecuación 4.1 se denomina *ecuación diferencial* y su solución, a diferencia de las ecuaciones del capítulo 1, corresponde a una función. Por ejemplo, en la ecuación diferencial 4.1, la solución es la función $x(t) = 50 + 2t - 0.1t^2$, dado que $\frac{dx(t)}{dt} = 2 - 0.2t$ y $x(0) = 50$.

Al tener la solución de una ecuación diferencial es posible dar respuesta a inquietudes particulares. En el modelo del automóvil la solución a ¿qué posición tiene el automóvil a los 3 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*, es decir, la función que satisface la ecuación. En este capítulo se presentan algunos métodos para construir una *aproximación* de las *imágenes* de la función en valores particulares. Así, para el modelo anterior, se construye una aproximación de $x(t_i)$ para algunos $t_i \in [0, 3]$.

## 4.1. 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 32.** Las siguientes ecuaciones 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

$$\frac{dx}{dt} = \frac{x}{t} \qquad 1 \leq t \leq 2$$

sujeta a:

$$x(1) = 1$$

Como se mencionó al principio, dar solución a este tipo de ecuaciones significa encontrar una función $x(t)$ tal que $\frac{dx(t)}{dt} = f(t,x)$ y $x(a) = \alpha$. Aunque existen diferentes técnicas para determinar la función $x(t)$, dichas técnicas están fuera de los alcances de este libro. A cambio, se presentan métodos para aproximar $x(t_i)$ para $t_i \in [a,b]$.

### 4.1.1. Método de Euler

Se debe recordar que el problema a solucionar en este capítulo es:

Sea:

$$\frac{dy}{dt} = f(t,y) \qquad a \leq t \leq b$$
$$\text{sujeto a:}$$
$$y(a) = \alpha$$

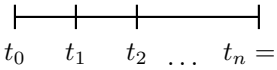
un problema de valor inicial. Para $t_i \in [a,b]$ obtener una aproximación de $y(t_i)$, donde $y(t)$ es la solución de la ecuación diferencial.

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 sus 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 $\xi$ tal que*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2$$

Ahora, la idea del método de Euler, consiste en construir una serie de puntos $(t_0, y_0)$, $(t_1, y_1), \ldots (t_n, y_n)$ tales que $t_n = b$ y $y_n$ sea la aproximación de $y(b)$.




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

- $t_i \in [a,b]$
- $y(a) = \alpha$

entonces una selección de los $t_i$ podrían ser valores que pertenezcan al intervalo $[a,b]$ tales que $t_i < t_{i+1}$, $t_0 = a$, $t_n = b$ y además la distancia entre $t_i$ y $t_{i+1}$ constante[1].

**Ejemplo 33.** En el intervalo $[1,3]$ seleccionar de manera uniforme cinco valores tales que $x_0 = 1$ y $x_5 = 3$.

[1] Este tipo de selección se denomina *uniforme*.

***Solución:*** dado que se conoce que el primer valor es $t_0 = 1$ y el último valor es $t_{10} = 3$, entonces para construir los siguientes puntos se utiliza la formula

$$t_i = t_0 + ih \qquad \text{con } i = 1, 2 \ldots, N-1$$

donde $h = \frac{b-a}{N}$, $N$ corresponde con la cantidad de puntos a construir y $a, b$ son los extremos del intervalo. En este caso $a = 1$, $b = 3$, $N = 5$ y por tanto $h = \frac{3-1}{5} = 0.4$ y

$$t_i = 1 + 0.4i$$

con lo cual se construyen los puntos ♢

| $t_i$ |
|---|
| 1 |
| 1.4 |
| 1.8 |
| 2.2 |
| 2.6 |
| 3 |

Luego de seleccionar $t_i$ de manera uniforme, la siguiente pregunta es ¿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 $y(t)$ cumple 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 $y'(t) = \frac{dy}{dt} = f(t, y)$ entonces

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))\,(t_{i+1} - t_i) + \frac{y''(\xi)}{2}(t_{i+1} - t_i)^2$$

Ahora, $t_{i+1} - t_i = h$ pues los $t_i$ fueron seleccionados de manera uniforme,

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))\,h + \frac{y''(\xi)}{2}(t_{i+1} - t_i)^2$$

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 $y_i$ que se están buscando. Como $y_0$ es la imagen de $t_0 = a$ y $y(a) = \alpha$, entonces $y_0 = \alpha$. Así, el método de Euler corresponde a:

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + f(t_i, y_i)h \end{aligned} \tag{4.2}$$

**Ejemplo 34.** Utilizar el método de Euler aproximar la solución al problema de valor inicial

$$y' = (2 - y)t \qquad 1 \leq t \leq 3$$

sujeto a:

$$y(1) = -5$$

***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 se desean 10 puntos, luego $h = \frac{b-a}{N} = \frac{3-1}{10} = 0.2$ y por lo tanto

| $t_i$ | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 | 2.2 | 2.4 | 2.6 | 2.8 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Paso 2.** Calcular los $y_i$. Para esto se utiliza el esquema recursivo definido en la ecuación 4.2 que corresponde, en este caso, a

$$y_0 = -5$$
$$y_{i+1} = y_i + 0.2[(2 - y_i)t_i]$$

luego

| $t_i$ | $y_i$ |
|---|---|
| 1 | -5 |
| 1.2 | -3.6 |
| 1.4 | -2.256 |
| 1.6 | -1.0643 |
| 1.8 | -0.083 |
| 2 | 0.6664 |
| 2.2 | 1.1998 |
| 2.4 | 1.5519 |
| 2.6 | 1.7669 |
| 2.8 | 1.8881 |
| 3 | 1.9507 |

♢

Se debe anotar que la cantidad de $t_i$ no es determinada y corresponde a posibilidades computacionales, pues para obtener buenos resultados se debe seleccionar una gran cantidad de puntos.

**Ejemplo 35.** Aplicar el método de Euler 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 5 puntos, luego $h = \frac{b-a}{N} = \frac{3-0}{5} = 0.6$, y por tanto

| $t_i$ | 0 | 0.6 | 1.2 | 1.8 | 2.4 | 3 |
|---|---|---|---|---|---|---|

**Paso 2.** Calcular los $y_i$ utilizando el esquema recursivo 4.2, se tiene

$$y_0 = 50$$
$$y_{i+1} = y_i + 0.6[(2 - t_i)]$$

luego

| $t_i$ | $y_i$ |
|---|---|
| 0 | 50 |
| 0.6 | 51.2 |
| 1.2 | 52.328 |
| 1.8 | 53.384 |
| 2.4 | 54.368 |
| 3 | 55.28 |

Ahora, si se utilizan 10 puntos, entonces

| $t_i$ | $y_i$ |
|---|---|
| 0 | 50 |
| 0.3 | 50.6 |
| 0.6 | 51.182 |
| 0.9 | 51.746 |
| 1.2 | 52.292 |
| 1.5 | 52.82 |
| 1.8 | 53.33 |
| 2.1 | 53.822 |
| 2.4 | 54.296 |
| 2.7 | 54.752 |
| 3 | 55.19 |

de donde una aproximación de $y(3)$ es 55.28 en el primer caso, y 55.19 en el segundo caso, que corresponde a un valor más cercano al real $y(3) = 55.1$. ♢

### 4.1.2. Orden del método de Euler

El método de Euler se obtuvo de la expansión en series de la función alrededor de $x = 0$ (serie de Maclaurin). En el caso más general, si la función se conoce completamente en cierto valor $x = x_0$, la expansión en series de potencia (serie de Taylor) es:

$$f(x_0 + \Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 + \frac{f'''(x_0)}{3!}\Delta x^3 + \cdots,$$

donde se supone que $\Delta x$ es un paso muy pequeño adelante del valor $x_0$ (de manera práctica $|\Delta x| \ll 1$). Como se pudo observar en un ejemplo de la sección pasada, si $\Delta x$ es pequeño, su cuadrado lo es más aún, y por lo tanto el método comete un error "más pequeño".

Pero entre más pequeño sea el intervalo, mayor es el gasto computacional y más cercano se está del épsilon de la máquina, que entre otros errores puede inducir a divisiones por cero. Por lo tanto, hay que balancear entre el error que se comete y tamaño del paso.

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 ese "$\frac{f''(\xi)}{2!} * \Delta x^2$" es el error que se comente 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:

$$\mathrm{E}_{\mathrm{total}} = (\text{Error de cada paso}) \times N,$$

donde el número de pasos $N$, es la longitud del intervalo de trabajo, $|b-a|$, divida entre el paso $\Delta x$, y por tanto:

$$\mathrm{E}_{\mathrm{total}} \approx \frac{f''(\xi)}{2!} * \Delta x^2 * \frac{|b-a|}{\Delta x}.$$

Al cancelar y reunir todas las constantes en una, se obtiene

$$\mathrm{E}_{\mathrm{total}} \approx \mathrm{Cte} * \Delta x.$$

Debido a que el error total es proporcional a $\Delta x$, se dice que el método de Euler es de orden 1. En lo sucesivo se construirán métodos que mejoren la precisión, de orden mayor, sin sacrificar el tamaño del paso, es decir, sin tener que tomar $\Delta x$ muy pequeños, que obliguen a dar muchos pasos y/o que expongan el cálculo por estar muy cerca al épsilon de la máquina.

### 4.1.3. Método de Verlet

La expresión de este método se obtiene a partir de dos expansiones de Taylor, una hacia adelante y otra hacia atrás,

$$f(x_0 + \Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 + \frac{f'''(x_0)}{3!}\Delta x^3 + \frac{f^{(4)}(x_0)}{4!}\Delta x^4 \cdots,$$

$$f(x_0 - \Delta x) = f(x_0) - f'(x_0)\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 - \frac{f'''(x_0)}{3!}\Delta x^3 + \frac{f^{(4)}(x_0)}{4!}\Delta x^4 \cdots,$$

al sumar,

$$f(x_0 + \Delta x) + f(x_0 - \Delta x) = 2f(x_0) + f''(x_0)\Delta x^2 + 2\frac{f^{(4)}(x_0)}{4!}\Delta x^4 \cdots$$

Al despejar,

$$f(x_0 + \Delta x) = 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)\Delta x^2 + 2\frac{f^{(4)}(x_0)}{4!}\Delta x^4 \cdots$$

Cortando está última en el tercer término,

$$f(x_0 + \Delta x) \approx 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)\Delta x^2,$$

da origen a la conocida fórmula recursiva, llamada método de Verlet.

$$y_{i+1} = 2y_i - y_{i-1} + y_i'' h^2.$$

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 de el método para solucionar el problema $y'' = F(y)$ con condiciones iniciales $y_0 = \alpha$ y $y_0' = \beta$:

$$\begin{aligned} y_0 &= \alpha \\ y_{-1} &= y_0 - \beta h + \frac{1}{2} F(y_0) h^2 \quad \text{(arranque)} \\ y_{i+1} &= 2y_i - y_{i-1} + F(y_i) h^2 \end{aligned} \tag{4.3}$$

**Ejemplo 36.** 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:

$$\begin{aligned} y'' &= -9.8 \qquad 0 \le t \le 4.5 \\ y(0) &= 100 \\ y'(0) &= 0 \end{aligned}$$

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 figura 4.2 resume los resultados numéricos, y también expone las diferencias entre los métodos de Verlet, Euler y la solución analítica al problema.

♢

### 4.1.4. Error del método de Verlet

Para analizar el error, se debe observar que antes de cortar la serie, el error lo domina el término $\Delta x^4$:

$$f(x_0 + \Delta x) \approx 2f(x_0) - f(x_0 - \Delta x) + f''(x_0)\Delta x^2, + \frac{f^{(iv)}(\xi)}{4!} * \Delta x^4,$$

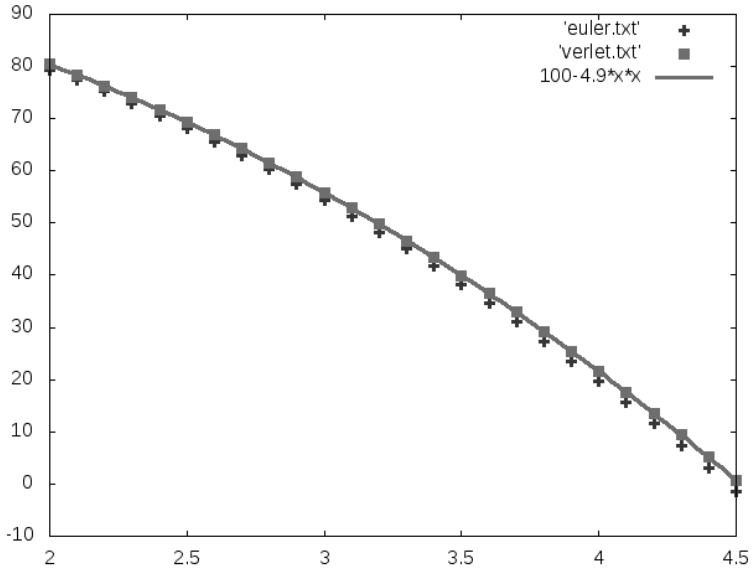



Figura 4.2: aproximación de la caída libre de un cuerpo.

luego en cada paso, el método de Verlet, comete un error de orden $\Delta x^4$. El error total cuando se han dado $N$ pasos, es:

$$\mathrm{E}_{\mathrm{total}} \approx \frac{f^{(iv)}(\xi)}{4!} * \Delta x^4 * \frac{|b-a|}{\Delta x},$$

cancelando y reuniendo todas las constantes en una,

$$\mathrm{E}_{\mathrm{total}} \approx \mathrm{Cte} * \Delta x^3,$$

se concluye que método de Verlet es de orden 3.

## Ejercicios 13

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)* $y' = -y, \quad 0 \le t \le 1, \quad y(0) = 5$ con $N = 10$.

   *b)* $y' = -2y + t, \quad 1 \le t \le 3, \quad y(1) = 2$ con $N = 20$.

   *c)* $y' = \frac{y}{t}, \quad -3 \le t \le -1, \quad y(-3) = 2$ con $N = 100$.

   *d)* $y' = \cos(2t) + \sin(3t), \quad 0 \le t \le 1, \quad y(0) = 2$ con $h = 0.0001$.

5. Dado el problema de valor inicial

$$y' = \frac{2}{t}y + e^{-t} \qquad 2 \leq t \leq 4$$

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 $y(2.05)$, $y(0.1)$ y $y(3.975)$.

6. Dado el problema de valor inicial

$$y' = -2ty \qquad 0 \leq t \leq 1$$

sujeto a:

$$y(0) = 1$$

   si se conoce el valor correcto de $y(1) = e^{-1}$, determinar cuántos puntos son necesarios en el método de Euler para obtener una aproximación con un error inferior a $10^{-2}$.

7. Si se conoce que la velocidad de un auto esta dada por $(5 - 0.4t)$ $\frac{\text{km}}{\text{s}}$ y su posición inicial es $x(0) = 30$ km, utilizar el método de Euler para aproximar la posición del auto a los 5 segundos.

8. En el circuito que se muestra en la figura 4.3, 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 $v_C(1)$.

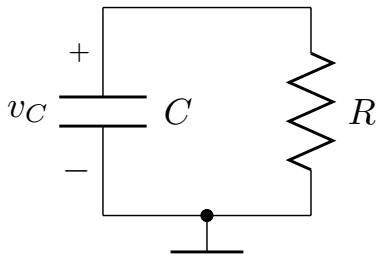



Figura 4.3: circuito RC.

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*) $y'' = y, \quad 0 \leq t \leq 1, \quad y(0) = 1, y'(0) = 1$ con $N = 10$.

   *b*) $y'' = -y, \quad 0 \leq t \leq 1, \quad y(0) = 1, y'(0) = 0$ con $N = 10$.

   *c*) $y'' = 4y, \quad 0 \leq t \leq 2, \quad y(0) = 0, y'(0) = 14$ con $N = 20$.

   *d*) $y'' = -9y, \quad 0 \leq t \leq 2, \quad y(0) = 1, y'(0) = 6$ con $N = 20$.

### 4.1.5. Métodos de Runge-Kutta orden 2

En la búsqueda de métodos más precisos en la solución de problemas de valor inicial, los métodos de Runge-Kutta ofrecen la mejor alternativa. Estos métodos a diferencia de otros, como sucede con los métodos de Taylor, 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:

$$y(t) = y(t_i) + y'(t_i)(t - t_i) + \frac{y''(t_i)}{2!}(t - t_i)^2 + \frac{1}{3!}y'''(t_i)(t - t_i)^3 + \cdots$$

Ahora, al evaluar en $t_i + h$, se obtiene:

$$y(t_i + h) = y(t_i) + y'(t_i)(t_i + h - t_i) + \frac{1}{2!}y''(t_i)(t_i + h - t_i)^2 + \frac{1}{3!}y'''(t_i)(t_i + h - t_i)^3 + \cdots$$

y al simplificar se concluye que

$$y(t_i + h) = y(t_i) + h\ y'(t_i) + \frac{h^2}{2!}y''(t_i) + \frac{h^3}{3!}y'''(t_i) + \cdots \tag{4.4}$$

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 2.

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 $\xi$ entre $x_0$ y $x$, y $\mu$ entre $y_0$ y $y$ tales que

$$g(x, y) = g(x_0, y_0) + \frac{\partial g}{\partial x}\Big|_{(x_0,y_0)} \cdot (x - x_0) + \frac{\partial g}{\partial y}\Big|_{(x_0,y_0)} \cdot (y - y_0) + R_2(x, y)$$

donde

$$R_2(x, y) = \frac{1}{2!}\sum_{j=0}^{2}\binom{2}{j}\frac{\partial^2 g}{\partial x^{2-j}\partial y^j}\Big|_{(\xi,\mu)} \cdot (x - x_0)^{2-j}(y - y_0)^j$$

A la función $P_1(x, y) = g(x_0, y_0) + \frac{\partial g}{\partial x}\Big|_{(x_0,y_0)} \cdot (x - x_0) + \frac{\partial g}{\partial y}\Big|_{(x_0,y_0)} \cdot (y - y_0)$ se le denomina *polinomio de Taylor de orden 1* de la función $g(x, y)$ alrededor del punto $(x_0, y_0)$.

**Ejemplo 37.** Dada $g(x, y) = \cos(x) + \sin(y)$, $(x_0, y_0) = (\frac{\pi}{3}, \frac{\pi}{6})$, entonces su polinomio de Taylor de orden 1 es:

$$\begin{aligned} P_1(x, y) &= g\left(\frac{\pi}{3}, \frac{\pi}{6}\right) + \frac{\partial g}{\partial x}\Big|_{(\frac{\pi}{3},\frac{\pi}{6})} \cdot \left(x - \frac{\pi}{3}\right) + \frac{\partial g}{\partial y}\Big|_{(\frac{\pi}{3},\frac{\pi}{6})} \cdot \left(y - \frac{\pi}{6}\right) \\ &= 1 - \frac{\sqrt{3}}{2}\left(x - \frac{\pi}{3}\right) + \frac{\sqrt{3}}{2}\left(y - \frac{\pi}{6}\right) \end{aligned}$$

Ahora, si de la serie 4.4 se toman los tres primeros términos, entonces:

$$y(t_i + h) \approx y(t_i) + h\ y'(t_i) + \frac{h^2}{2}y''(t_i)$$

Dado que $y' = f(t, y)$ en un problema de valor inicial, 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 $f'(t, y) = \frac{\partial f}{\partial t}\frac{dt}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$, se tiene que:

$$y(t_i + h) \approx y(t_i) + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{\partial f}{\partial y}\Big|_{(t_i,y_i)} \cdot \frac{dy}{dt}\Big|_{t_i}\right)$$

lo cual es equivalente a

$$\begin{aligned} y(t_i + h) &\approx y(t_i) + h\left(f(t_i, y_i) + \frac{h}{2}\left(\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{\partial f}{\partial y}\Big|_{(t_i,y_i)} \cdot \frac{dy}{dt}\Big|_{t_i}\right)\right) \\ &= y(t_i) + h\left(f(t_i, y_i) + \tfrac{h}{2}\left(\tfrac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \tfrac{\partial f}{\partial y}\Big|_{(t_i,y_i)} \cdot f(t_i, y_i)\right)\right) \end{aligned} \tag{4.5}$$

Si se aproxima $f(t_i, y_i) + \frac{h}{2}\frac{\partial f}{\partial t}\big|_{(t_i,y_i)} + \frac{h}{2}\frac{\partial f}{\partial y}\big|_{(t_i,y_i)} \cdot f(t_i, y_i)$ con la serie de Taylor de orden 1 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 $a_1 = 1$, $\alpha_1 = \frac{h}{2}$ y $\beta_1 = \frac{h}{2} f(t_i, y_i)$ y por tanto

$$f(t_i, y_i) + \frac{h}{2}\frac{\partial f}{\partial t}\Big|_{(t_i,y_i)} + \frac{h}{2}\frac{\partial f}{\partial y}\Big|_{(t_i,y_i)} \cdot f(t_i, y_i) \approx f\left(t_i + \frac{h}{2}, y_1 + \frac{h}{2} f(t_i, y_i)\right)$$

Al reemplazar en la ecuación 4.5 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:**

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + hf\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} f(t_i, y_i)\right) \end{aligned} \tag{4.6}$$

**Ejemplo 38.** Utilizando el método del punto medio, aproximar la solución del problema de valor inicial

$$\begin{aligned} y' &= -2ty \qquad 0 \le t \le 1 \\ &\text{sujeto a:} \\ y(0) &= 1 \end{aligned}$$

***Solución:*** para utilizar el método, se aplican los siguientes pasos:

**Paso 1.** Seleccionar $t_i$ de manera uniforme, para este ejemplo se seleccionan 5 puntos, luego $h = \frac{b-a}{N} = \frac{1-0}{5} = 0.2$ y por tanto

| $t_i$ | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 |
|---|---|---|---|---|---|---|

**Paso 2.** Calcular los $y_i$ utilizando el esquema recursivo definido en 4.6, 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 4.1 se presentan los resultados del método y al comparar con la solución real (cuadro 4.2) del problema de valor inicial, se evidencia la precisión del método.

| $t_i$ | $y_i$ |
|---|---|
| 0 | 1 |
| 0.2 | 0.96 |
| 0.4 | 0.8494 |
| 0.6 | 0.6931 |
| 0.8 | 0.5223 |
| 1 | 0.3643 |

Cuadro 4.1: aproximación del problema $y' = -2ty$.

| $t_i$ | $y_i$ |
|---|---|
| 0 | 1 |
| 0.2 | 0.9607 |
| 0.4 | 0.8521 |
| 0.6 | 0.6976 |
| 0.8 | 0.5272 |
| 1 | 0.3678 |

Cuadro 4.2: solución real 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 2 con las ideas desarrolladas en la deducción del método del punto medio, entre los cuales se tienen los siguientes.

**Método modificado de Euler:**

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ y_{i+1} &= y_i + \frac{1}{2}\left(k_1 + hf\left(t_{i+1}, y_i + k_1\right)\right) \end{aligned}$$

**Método de Heun:**

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_i, y_i) \\ y_{i+1} &= y_i + \frac{1}{4}\left(k_1 + 3hf\left(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_1\right)\right) \end{aligned}$$

**Ejemplo 39.** Utilizar el método modificado de Euler y el de Heun con $N = 5$, para aproximar soluciones del problema de valor inicial

$$\begin{aligned} y' &= -3t^2(y+1) \qquad 1 \le t \le 2 \\ &\text{sujeto a:} \\ y(1) &= -2 \end{aligned}$$

cuya solución real es $y(t) = -e^{-t^3+1} - 1$.

***Solución:*** dado que $N = 5$ entonces $h = 0.2$. En el cuadro 4.3 se resumen los resultados.

| $t_i$ | $y(t_i)$ | Método modificado de Euler | Error | Método de Heun | Error |
|---|---|---|---|---|---|
| 1 | -2 | -2 | 0 | -2 | 0 |
| 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 | 0.02103 | -1.0692 | 0.0613 |
| 2 | -1.0009 | -1.0457 | 0.0448 | -1.0701 | 0.0692 |

Cuadro 4.3: 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 14

1. Utilizar el método del punto medio para aproximar la solución de los siguientes problemas de valor inicial.

   *a)* $y' = -2t + yt, \quad 1 \le t \le 2, \quad y(1) = -3$; con $N = 5$

   *b)* $y' = -2e^{-ty}, \quad 0 \le t \le 3, \quad y(0) = 1$; con $N = 10$

*c)* $y' = -2\cos(2t) + \sin(y), \quad -2 \leq t \leq -1, \quad y(-1) = 0$; con $N = 10$

*d)* $y' = 1 + y/t, \quad -2.5 \leq t \leq -0.5, \quad y(-2.5) = 1$; con $N = 10$

*e)* $y' = 5t^2 - 3t + e^{-t}, \quad 2 \leq t \leq 4, \quad y(2) = 2$; con $N = 15$

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)* Utilizar los resultados del numeral anterior para construir el interpolador de mínimos cuadrados y aproximar $y(4.2367)$, $y(4.56982)$ y $y(4.12563)$.

*c)* Si conoce que la solución exacta es $y(t) = \frac{5}{65 - 2t^{5/2}}$, 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 de dicho problema de valor inicial.

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.

### 4.1.6. Método de Runge-Kutta orden 4 (RK4)

El método de Runge-Kutta de orden 4 (RK4) es uno de los métodos más utilizados en la práctica. Su deducción sigue las ideas de los métodos de Runge-Kutta orden 2 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 4:**

$$
\begin{aligned}
y_0 &= \alpha \\
k_1 &= hf(t_i, y_i) \\
k_2 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right) \\
k_3 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right) \\
k_4 &= hf(t_{i+1}, y_i + k_3) \\
y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

**Ejemplo 40.** Utilizar el método de Runge-Kutta de orden 4 para aproximar la solución del siguiente problema de valor inicial.

$$
\begin{aligned}
y' &= -3(t^2 - 1)(y + ty) \qquad 0 \leq t \leq 1 \\
&\text{sujeto a:} \\
y(0) &= -2
\end{aligned}
$$

con $N = 10$.

***Solución:*** como $N = 10$ se tiene que $h = 0.2$. En el cuadro 4.4 se resumen los resultados donde $y(t_i)$ representa el valor real. ♢

| $t_i$ | $y(t_i)$ | RK4 | Error |
|---|---|---|---|
| 0 | -2 | -2 | 0 |
| 0.1 | -2.10585459 | -2.105854583 | 7.04491E-09 |
| 0.2 | -2.227013791 | -2.227013731 | 5.93929E-08 |
| 0.3 | -2.369294038 | -2.369293871 | 1.66495E-07 |
| 0.4 | -2.539123488 | -2.539123149 | 3.3929E-07 |
| 0.5 | -2.743606354 | -2.743605763 | 5.9041E-07 |
| 0.6 | -2.990594002 | -2.990593068 | 9.34419E-07 |
| 0.7 | -3.288763537 | -3.288762149 | 1.38808E-06 |
| 0.8 | -3.647704642 | -3.647702672 | 1.97065E-06 |
| 0.9 | -4.078015556 | -4.078012852 | 2.70422E-06 |
| 1 | -4.591409142 | -4.591405528 | 3.61412E-06 |

Cuadro 4.4: solución aproximada del problema $y' = -3(t^2 - 1)(y + ty)$.

Aunque los resultados del cuadro 4.4 dan muestra de la precisión del método de Runge-Kutta orden 4 (RK4), un elemento no tan positivo, es la cantidad de evaluaciones de la función $f(t, y)$ que utiliza este 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 2 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 15

1. Utilizar el método de Runge-Kutta de orden 4 para aproximar la solución de las siguientes problemas de valor inicial.

   *a)* $y' = -2t + yt, \quad 1 \leq t \leq 2, \quad y(1) = -3$; con $N = 5$

   *b)* $y' = -2e^{-ty}, \quad 0 \leq t \leq 3, \quad y(0) = 1$; con $N = 10$

   *c)* $y' = -2\cos(2t) + \sin(y), \quad -2 \leq t \leq -1, \quad y(-1) = 0$; con $N = 10$

   *d)* $y' = 1 + y/t, \quad -2.5 \leq t \leq -0.5, \quad y(-2.5) = 1$; con $N = 10$

   *e)* $y' = 5t^2 - 3t + e^{-t}, \quad 2 \leq t \leq 4, \quad y(2) = 2$; con $N = 15$

   Comparar los resultados con el uso del método de Heun.

2. 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

eybooks.com

## 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 del código, que permite un expresión rica en pocas líneas de código, y también es usado ampliamente por la comunidad. 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 en la consola interactiva (a explorar en más detalle en un instante), se obtiene, como es de esperar:

Salida

```
Hola Mundo
```

## A.2. Consola interactiva

Al ser Python de naturaleza interpretada, las instrucciones escritas se van ejecutando una a una en la consola interactiva. Dicha consola indica con >>> que se encuentra lista para recibir un comando de entrada, mientras que ... indica la continuación de una línea.

```
>>> x = 12
>>> y = 17
>>> z = 2*x+y**3
>>> z
4937
>>> 7**52
88124787089723195184393736687912818113311201L
>>> 1/8
```

```
0
>>> 1.0/8
0.125
>>> from math import *
>>> sqrt(2)
1.4142135623730951
>>> exp(1)
2.718281828459045
>>> pi
3.141592653589793
```

## A.3. Funciones

La construcción de las funciones se hacen 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 una indentación* (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, la indentación es fundamental en `Python`.

```
1 def␣f(entrada):
2 ␣␣␣␣return␣entrada**3
3 print␣f(7)
```

Salida

```
343
```

Sobre el tema de asignación de variables y otros identificadores, las siguientes palabras son reservadas por `Python`:

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

## A.4. Estructuras básicas de control

### A.4.1. `if`

```
1 if␣x␣==␣y:
2 ␣␣␣␣print␣"x␣es␣igual␣a␣y"
3 else:
4 ␣␣␣␣if␣x␣<␣y:
```

```
    print " x es menor a y "
else:
    print " x es mayor a y "
```

### A.4.2. while

```
i = 1
while i < 5:
    print i*6 + 1
    i += 1
```

Salida

```
7
13
19
25
```

### A.4.3. for

```
for i in range(5):
    print i*i + 1
```

Salida

```
1
2
5
10
17
```

## A.5. Ejemplos

### A.5.1. 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.5.2. 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.5.3. ¿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.5.4. Numpy

Numpy es un poderoso módulo de Python para cálculo científico matricial. Brinda a Python de una funcionalidad semejante a la de MATLAB.

```
import numpy as np
from numpy.linalg import solve, inv
```

```
3
4  a=np.array([[-1.1, 2.5], [1.3, 4.2]])  # crear matriz
5  print a
6  print a.transpose()  # transpuesta
7  print inv(a)  # inversa
8  b=np.array([2, -3])
9  print b
10 s=solve(a, b)  # solucionar el sistema de ecuaciones
11 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 16

1. Calcular el valor de la expresión $\ln\big(4+\cos\left(\frac{\pi}{4}\right)\big)+\sqrt{\arctan\left(7+\frac{\pi}{3}\right)}$
2. Diseñar una función que imprima los $n$ primeros números triangulares, esto es, números de la forma $\dfrac{n(n+1)}{2}$.
3. Dado un entero $n$, diseñar una función que retorne la suma de los dígitos de $n$.
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 algoritmos

## B.1. Ecuaciones de una variable

### B.1.1. Bisección

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 *


def pol(x):
    return x**3+4*x**2-10  # retorna pol(x) = x^3 + 4x^2 - 10


```

```
29 def trig(x):
30     return x*cos(x-1)-sin(x)  # retorna trig(x) = x cos(x − 1) − sin(x)
31
32
33 def pote(x):  # retorna pote(x) = 7^x − 13
34     return pow(7.0, x)-13
35
36
37 def bisec(f, a, b, tol, n):  # Método de bisección
38     i = 1
39     while i <= n:
40         p = a+(b-a)/2.0
41         print "Iter = ", "%03d" % i, "; p =", "%.14f" % p
42         if abs(f(p)) <= 1e-15 or (b-a)/2.0 < tol:
43             return p
44         i += 1
45         if f(a)*f(p) > 0:
46             a = p
47         else:
48             b = p
49     print "Iteraciones agotadas: Error!"
50     return
51
52 # pol(x), a = 1.0, b = 2.0, Tol = 10^−8, N_0 = 100
53 print "\n"+r"-- Bisecci\'on funci\'on pol(x): "+"\n"
54 bisec(pol, 1.0, 2.0, 1e-8, 100)
55
56 # trig(x), a = 4.0, b = 6.0, Tol = 10^−8, N_0 = 100
57 print "\n"+r"-- Bisecci\'on funci\'on trig(x): "+"\n"
58 bisec(trig, 4.0, 6.0, 1e-8, 100)
59
60 # pote(x), a = 0, b = 2.0, Tol = 10^−8, N_0 = 100
61 print "\n"+r"-- Bisecci\'on funci\'on pote(x): "+"\n"
62 bisec(pote, 0.0, 2.0, 1e-8, 100)
```

Salida

```
-- Bisecci\'on funci\'on pol(x):

Iter =  001 ; p = 1.50000000000000
Iter =  002 ; p = 1.25000000000000
Iter =  003 ; p = 1.37500000000000
Iter =  004 ; p = 1.31250000000000
Iter =  005 ; p = 1.34375000000000
Iter =  006 ; p = 1.35937500000000
Iter =  007 ; p = 1.36718750000000
Iter =  008 ; p = 1.36328125000000
Iter =  009 ; p = 1.36523437500000
Iter =  010 ; p = 1.36425781250000
```

```
Iter =  011 ; p = 1.36474609375000
Iter =  012 ; p = 1.36499023437500
Iter =  013 ; p = 1.36511230468750
Iter =  014 ; p = 1.36517333984375
Iter =  015 ; p = 1.36520385742188
Iter =  016 ; p = 1.36521911621094
Iter =  017 ; p = 1.36522674560547
Iter =  018 ; p = 1.36523056030273
Iter =  019 ; p = 1.36522865295410
Iter =  020 ; p = 1.36522960662842
Iter =  021 ; p = 1.36523008346558
Iter =  022 ; p = 1.36522984504700
Iter =  023 ; p = 1.36522996425629
Iter =  024 ; p = 1.36523002386093
Iter =  025 ; p = 1.36522999405861
Iter =  026 ; p = 1.36523000895977
Iter =  027 ; p = 1.36523001641035

-- Bisecci\'on funci\'on trig(x):

Iter =  001 ; p = 5.00000000000000
Iter =  002 ; p = 5.50000000000000
Iter =  003 ; p = 5.75000000000000
Iter =  004 ; p = 5.62500000000000
Iter =  005 ; p = 5.56250000000000
Iter =  006 ; p = 5.59375000000000
Iter =  007 ; p = 5.60937500000000
Iter =  008 ; p = 5.60156250000000
Iter =  009 ; p = 5.59765625000000
Iter =  010 ; p = 5.59960937500000
Iter =  011 ; p = 5.59863281250000
Iter =  012 ; p = 5.59912109375000
Iter =  013 ; p = 5.59936523437500
Iter =  014 ; p = 5.59924316406250
Iter =  015 ; p = 5.59930419921875
Iter =  016 ; p = 5.59933471679688
Iter =  017 ; p = 5.59931945800781
Iter =  018 ; p = 5.59931182861328
Iter =  019 ; p = 5.59931564331055
Iter =  020 ; p = 5.59931373596191
Iter =  021 ; p = 5.59931278228760
Iter =  022 ; p = 5.59931325912476
Iter =  023 ; p = 5.59931302070618
Iter =  024 ; p = 5.59931313991547
Iter =  025 ; p = 5.59931308031082
Iter =  026 ; p = 5.59931305050850
Iter =  027 ; p = 5.59931303560734
Iter =  028 ; p = 5.59931302815676
```

```
-- Bisecci\'on funci\'on pote(x):

Iter =  001 ; p = 1.00000000000000
Iter =  002 ; p = 1.50000000000000
Iter =  003 ; p = 1.25000000000000
Iter =  004 ; p = 1.37500000000000
Iter =  005 ; p = 1.31250000000000
Iter =  006 ; p = 1.34375000000000
Iter =  007 ; p = 1.32812500000000
Iter =  008 ; p = 1.32031250000000
Iter =  009 ; p = 1.31640625000000
Iter =  010 ; p = 1.31835937500000
Iter =  011 ; p = 1.31738281250000
Iter =  012 ; p = 1.31787109375000
Iter =  013 ; p = 1.31811523437500
Iter =  014 ; p = 1.31823730468750
Iter =  015 ; p = 1.31817626953125
Iter =  016 ; p = 1.31814575195312
Iter =  017 ; p = 1.31813049316406
Iter =  018 ; p = 1.31812286376953
Iter =  019 ; p = 1.31812667846680
Iter =  020 ; p = 1.31812477111816
Iter =  021 ; p = 1.31812381744385
Iter =  022 ; p = 1.31812334060669
Iter =  023 ; p = 1.31812310218811
Iter =  024 ; p = 1.31812322139740
Iter =  025 ; p = 1.31812328100204
Iter =  026 ; p = 1.31812325119972
Iter =  027 ; p = 1.31812323629856
Iter =  028 ; p = 1.31812322884798
```

### B.1.2. *Regula falsi*

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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.
```

```
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 regula falsi y algunos casos de salida.
21
22 from math import *
23
24
25 def pol(x):
26     return x**3+4*x**2-10  # retorna pol(x) = x^3 + 4x^2 - 10
27
28
29 def trig(x):
30     return x*cos(x-1)-sin(x)  # retorna trig(x) = x cos(x-1) - sin(x)
31
32
33 def pote(x):  # retorna pote(x) = 7^x - 13
34     return pow(7.0, x)-13
35
36
37 def regula(f, p0, p1, tol, n):  # Método de Regula falsi
38     i = 2
39     while i <= n:
40         q0 = f(p0)
41         q1 = f(p1)
42         p = p1-(q1*(p1-p0))/(q1-q0)
43         print "Iter = ", "%03d" % i, "; p =", "%.14f" % p
44         if abs(p-p1) < tol:
45             return p
46         i += 1
47         q = f(p)
48         if q*q1 < 0:
49             p0 = p1
50             q0 = q1
51         p1 = p
52         q1 = q
53     print "Iteraciones agotadas: Error!"
54     return
55
56 # pol(x), a = 1.0, b = 2.0, Tol = 10^-8, N_0 = 100
57 print "\n"+r"-- Regula falsi funci\'on pol(x): "+"\n"
58 regula(pol, 1.0, 2.0, 1e-8, 100)
59
60 # trig(x), a = 4.0, b = 6.0, Tol = 10^-8, N_0 = 100
61 print "\n"+r"-- Regula falsi funci\'on trig(x): "+"\n"
62 regula(trig, 4.0, 6.0, 1e-8, 100)
63
64 # pote(x), a = 0, b = 2.0, Tol = 10^-8, N_0 = 100
```

```
65 print "\n"+r"-- Regula falsi funci\'on pote(x): "+"\n"
66 regula(pote, 0.0, 2.0, 1e-8, 100)
```

Salida

```
-- Regula falsi funci\'on pol(x):

Iter =  002 ; p = 1.26315789473684
Iter =  003 ; p = 1.33882783882784
Iter =  004 ; p = 1.35854634182478
Iter =  005 ; p = 1.36354744004209
Iter =  006 ; p = 1.36480703182678
Iter =  007 ; p = 1.36512371788438
Iter =  008 ; p = 1.36520330366260
Iter =  009 ; p = 1.36522330198554
Iter =  010 ; p = 1.36522832702552
Iter =  011 ; p = 1.36522958967385
Iter =  012 ; p = 1.36522990694057
Iter =  013 ; p = 1.36522998666042
Iter =  014 ; p = 1.36523000669168
Iter =  015 ; p = 1.36523001172495

-- Regula falsi funci\'on trig(x):

Iter =  002 ; p = 5.23565737472245
Iter =  003 ; p = 5.56947741050970
Iter =  004 ; p = 5.59762303531170
Iter =  005 ; p = 5.59922074987343
Iter =  006 ; p = 5.59930799603642
Iter =  007 ; p = 5.59931274963277
Iter =  008 ; p = 5.59931300860022
Iter =  009 ; p = 5.59931302270821
Iter =  010 ; p = 5.59931302347678

-- Regula falsi funci\'on pote(x):

Iter =  002 ; p = 0.50000000000000
Iter =  003 ; p = 0.83505824110381
Iter =  004 ; p = 1.04516978342356
Iter =  005 ; p = 1.16884708035964
Iter =  006 ; p = 1.23819700824382
Iter =  007 ; p = 1.27586210147702
Iter =  008 ; p = 1.29593375501013
Iter =  009 ; p = 1.30651664223152
Iter =  010 ; p = 1.31206443941295
Iter =  011 ; p = 1.31496381829887
Iter =  012 ; p = 1.31647664069380
Iter =  013 ; p = 1.31726532550939
Iter =  014 ; p = 1.31767631153884
```

```
Iter =  015 ; p = 1.31789042821980
Iter =  016 ; p = 1.31800196593592
Iter =  017 ; p = 1.31806006455273
Iter =  018 ; p = 1.31809032641627
Iter =  019 ; p = 1.31810608866450
Iter =  020 ; p = 1.31811429854456
Iter =  021 ; p = 1.31811857469975
Iter =  022 ; p = 1.31812080195024
Iter =  023 ; p = 1.31812196202005
Iter =  024 ; p = 1.31812256624534
Iter =  025 ; p = 1.31812288095750
Iter =  026 ; p = 1.31812304487605
Iter =  027 ; p = 1.31812313025338
Iter =  028 ; p = 1.31812317472235
Iter =  029 ; p = 1.31812319788411
Iter =  030 ; p = 1.31812320994797
Iter =  031 ; p = 1.31812321623145
```

### B.1.3. Newton

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 Newton y algunos casos de salida.

from math import *


def expo(x):  # retorna expo(x) = x^2 + e^{-2x} - 2xe^{-x}
    return x**2+exp(-2*x)-2*x*exp(-x)


```

```
29 def expoprima(x):  # retorna expoprima(x) = d/dx expo(x)
30     return 2*x-2*exp(-2*x)-2*exp(-x)+2*x*exp(-x)
31
32
33 def trig(x):  # retorna trig(x) = cos(x) - x
34     return cos(x)-x
35
36
37 def trigprima(x):  # retorna trigprima(x) = d/dx trig(x)
38     return -sin(x)-1
39
40
41 def newton(f, fprima, p0, tol, n):  # Método de Newton
42     i = 1
43     while i <= n:
44         p = p0-f(p0)/fprima(p0)
45         print "Iter = ", "%03d" % i, "; p =", "%.14f" % p
46         if abs(p-p0) < tol:
47             return p
48         p0 = p
49         i += 1
50     print "Iteraciones agotadas: Error!"
51     return
52
53 # trig(x), trigprima(x), p0 = π/4, Tol = 10^-10, N0 = 100
54 print "\n"+r"-- Newton funci\'on trig(x): "+"\n"
55 newton(trig, trigprima, pi/4, 1e-10, 100)
56
57 # expo(x), expoprima(x), p0 = 4.0, Tol = 10^-8, N0 = 100
58 print "\n"+r"-- Newton funci\'on expo(x): "+"\n"
59 newton(expo, expoprima, 4.0, 1e-8, 100)
```

Salida

```
-- Newton funci\'on trig(x):

Iter =  001 ; p = 0.73953613351524
Iter =  002 ; p = 0.73908517810601
Iter =  003 ; p = 0.73908513321516
Iter =  004 ; p = 0.73908513321516

-- Newton funci\'on expo(x):

Iter =  001 ; p = 2.04496552490523
Iter =  002 ; p = 1.19690154878521
Iter =  003 ; p = 0.85332087193837
Iter =  004 ; p = 0.70348791030730
Iter =  005 ; p = 0.63370473523551
```

```
Iter =  006 ; p = 0.60003137481868
Iter =  007 ; p = 0.58349045862567
Iter =  008 ; p = 0.57529281785962
Iter =  009 ; p = 0.57121206025870
Iter =  010 ; p = 0.56917617940458
Iter =  011 ; p = 0.56815936124237
Iter =  012 ; p = 0.56765123244958
Iter =  013 ; p = 0.56739723809056
Iter =  014 ; p = 0.56727025841597
Iter =  015 ; p = 0.56720677295439
Iter =  016 ; p = 0.56717503131751
Iter =  017 ; p = 0.56715916077200
Iter =  018 ; p = 0.56715122556949
Iter =  019 ; p = 0.56714725798531
Iter =  020 ; p = 0.56714527419961
Iter =  021 ; p = 0.56714428230315
Iter =  022 ; p = 0.56714378635442
Iter =  023 ; p = 0.56714353837910
Iter =  024 ; p = 0.56714341450390
Iter =  025 ; p = 0.56714335239325
Iter =  026 ; p = 0.56714332139724
Iter =  027 ; p = 0.56714330535002
Iter =  028 ; p = 0.56714329778561
```

### B.1.4. Secante

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 *
```

```
23
24
25 def trig(x):  # retorna trig(x) = sin(2/x)
26     return sin(2/x)
27
28
29 def pol(x):  # retorna pol(x) = x^3 - 2
30     return x**3-2
31
32
33 def secante(f, p0, p1, tol, n):  # Método de la secante
34     i = 2
35     while i <= n:
36         p = p1-(f(p1)*(p1-p0))/(f(p1)-f(p0))
37         print "Iter = ", "%03d" % i, "; p =", "%.14f" % p
38         if abs(p-p1) < tol:
39             return p
40         p0 = p1
41         p1 = p
42         i += 1
43     print "Iteraciones agotadas: Error!"
44     return
45
46 # pol(x), p0 = -3.0, p1 = 3.0,Tol = 10^-10, N0 = 100
47 print "\n"+r"-- Secante funci\'on pol(x): "+"\n"
48 secante(pol, -3.0, 3.0, 1e-10, 100)
49
50 # trig(x), p0 = 1.1, p1 = 0.8,Tol = 10^-8, N0 = 100
51 print "\n"+r"-- Secante funci\'on trig(x): "+"\n"
52 secante(trig, 1.1, 0.8, 1e-8, 100)
```

Salida

```
-- Secante funci\'on pol(x):

Iter =  002 ; p = 0.22222222222222
Iter =  003 ; p = 0.42693773824651
Iter =  004 ; p = 6.31355877988707
Iter =  005 ; p = 0.47191278930293
Iter =  006 ; p = 0.51591568078189
Iter =  007 ; p = 3.05938543642494
Iter =  008 ; p = 0.68216111845434
Iter =  009 ; p = 0.82340822908250
Iter =  010 ; p = 1.66897585774860
Iter =  011 ; p = 1.12142575196627
Iter =  012 ; p = 1.22112631430500
Iter =  013 ; p = 1.26462114511073
Iter =  014 ; p = 1.25977368664182
Iter =  015 ; p = 1.25992050148297
```

```
Iter =  016 ; p = 1.25992104995902
Iter =  017 ; p = 1.25992104989487

-- Secante funci\'on trig(x):

Iter =  002 ; p = 0.31616955314644
Iter =  003 ; p = 0.27916396598709
Iter =  004 ; p = 0.31832835636724
Iter =  005 ; p = 0.31830985629667
Iter =  006 ; p = 0.31830988618553
Iter =  007 ; p = 0.31830988618379
```

### B.1.5. Punto fijo

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 *


def pote(x):  # retorna pote(x) = 2^{-x}
    return pow(2.0, -x)


def pol(x):  # retorna pol(x) = (x^2-1)/3
    return (x**2-1)/3


def puntofijo(f, p0, tol, n):  # Método del punto fijo
    i = 1
```

```
35      while i <= n:
36          p = f(p0)
37          print "Iter = ", "%03d" % i, "; p =", "%.14f" % p
38          if abs(p-p0) < tol:
39              return p
40          p0 = p
41          i += 1
42      print "Iteraciones agotadas: Error!"
43      return
44
45  # pol(x), p_0 = 0.9, Tol = 10^-10, N_0 = 100
46  print "\n"+r"-- Punto fijo funci\'on pol(x): "+"\n"
47  puntofijo(pol, 0.9, 1e-10, 100)
48
49  # pote(x), p_0 = 0.5, Tol = 10^-8, N_0 = 100
50  print "\n"+r"-- Punto fijo funci\'on pote(x): "+"\n"
51  puntofijo(pote, 0.5, 1e-8, 100)
```

Salida

```
-- Punto fijo funci\'on pol(x):

Iter =  001 ; p = -0.06333333333333
Iter =  002 ; p = -0.33199629629630
Iter =  003 ; p = -0.29659281974851
Iter =  004 ; p = -0.30401089975788
Iter =  005 ; p = -0.30252579094280
Iter =  006 ; p = -0.30282604860481
Iter =  007 ; p = -0.30276546142880
Iter =  008 ; p = -0.30277769178860
Iter =  009 ; p = -0.30277522311839
Iter =  010 ; p = -0.30277572142187
Iter =  011 ; p = -0.30277562083916
Iter =  012 ; p = -0.30277564114182
Iter =  013 ; p = -0.30277563704372
Iter =  014 ; p = -0.30277563787092
Iter =  015 ; p = -0.30277563770395
Iter =  016 ; p = -0.30277563773766

-- Punto fijo funci\'on pote(x):

Iter =  001 ; p = 0.70710678118655
Iter =  002 ; p = 0.61254732653607
Iter =  003 ; p = 0.65404086004207
Iter =  004 ; p = 0.63549784581337
Iter =  005 ; p = 0.64371864172287
Iter =  006 ; p = 0.64006102117724
Iter =  007 ; p = 0.64168580704300
Iter =  008 ; p = 0.64096353717796
```

```
Iter =  009 ; p = 0.64128450906659
Iter =  010 ; p = 0.64114185147174
Iter =  011 ; p = 0.64120525244986
Iter =  012 ; p = 0.64117707452884
Iter =  013 ; p = 0.64118959776687
Iter =  014 ; p = 0.64118403197862
Iter =  015 ; p = 0.64118650561396
Iter =  016 ; p = 0.64118540624078
Iter =  017 ; p = 0.64118589484183
Iter =  018 ; p = 0.64118567768987
Iter =  019 ; p = 0.64118577420003
Iter =  020 ; p = 0.64118573130743
Iter =  021 ; p = 0.64118575037045
Iter =  022 ; p = 0.64118574189816
```

## B.2. Interpolación

### B.2.1. Lagrange

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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):

    def L(k, x):  # pol L_k(x) = \prod_{i \neq k} (x-x_i)/(x_k-x_i)
```

```
28          out = 1.0
29          for i, p in enumerate(datos):
30              if i != k:
31                  out *= (x-p[0])/(datos[k][0]-p[0])
32          return out
33
34      def P(x):  # polinomio P(x) = sum_k f(x_k)L_k(x)
35          lag = 0.0
36          for k, p in enumerate(datos):
37              lag += p[1]*L(k, x)
38          return lag
39
40      return P
41
42  # datos para f(x) = 1/x con x_0 = 2, x_1 = 2.75 y x_2 = 4
43  datosf = [(2.0, 1.0/2.0), (11.0/4.0, 4.0/11.0), (4.0, 1.0/4.0)]
44  Pf = LagrangePol(datosf)
45  print "\n"+r"-- Polinomio de Lagrange en x=3: "+"\n"
46  print Pf(3)
47
48  # 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
49  datosg = [(1.0, 0.1411), (1.3, -0.6878), (1.6, -0.9962),
50            (1.9, -0.5507), (2.2, 0.3115)]
51  Pg = LagrangePol(datosg)
52  print "\n"+r"-- Polinomio de Lagrange en x=1.5: "+"\n"
53  print 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

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # ----------------------------------------------------------------------
5  # Compendio de algoritmos.
6  # Matemáticas para Ingeniería. Métodos numéricos con Python.
7  # Copyright (C) 2017  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
```

```
# 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 Newton y algunos casos de salida.

from math import *
from pprint import pprint


def NewtonPol(datos):
    n = len(datos)-1
    F = [[0 for x in datos] for x in datos]  # crear tabla nula

    for i, p in enumerate(datos):  # condiciones iniciales
        F[i][0] = p[1]

    for i in range(1, n+1):  # tabla de diferencias divididas
        for j in range(1, i+1):
            F[i][j] = (F[i][j-1]-F[i-1][j-1])/(datos[i][0]-datos[i-j][0])

    def L(k, x):  # polinomio L_k(x) = \prod_{i <= k} (x - x_i)
        out = 1.0
        for i, p in enumerate(datos):
            if i <= k:
                out *= (x-p[0])
        return out

    def P(x):  # P(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, ..., x_k] L_{k-1}(x)
        newt = 0.0
        for i in range(1, n+1):
            newt += F[i][i]*L(i-1, x)
        return newt + F[0][0]

    return F, P

datost = [(-1.0, 3.0), (0.0, -4.0), (1.0, 5.0), (2.0, -6.0)]
T, P = NewtonPol(datost)
print "\nTabla de diferencias divididas:"
pprint(T)
print "\nEvaluar el polinomio en x=0"
print P(0.0)
```

```
58
59 datosf = [(2.0, 1.0/2.0), (11.0/4.0, 4.0/11.0), (4.0, 1.0/4.0)]
60 T, P = NewtonPol(datosf)
61 print "\nTabla de diferencias divididas:"
62 pprint(T)
63 print "\nEvaluar el polinomio en x=3"
64 print P(3.0)
```

Salida

```
Tabla de diferencias divididas:
[[3.0, 0, 0, 0],
 [-4.0, -7.0, 0, 0],
 [5.0, 9.0, 8.0, 0],
 [-6.0, -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
0.329545454545
```

### B.2.3. Trazadores cúbicos

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # ------------------------------------------------------------------------
5  # Compendio de algoritmos.
6  # Matemáticas para Ingeniería. Métodos numéricos con Python.
7  # Copyright (C) 2017  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
```

```
# Implementaci\'on de trazadores c\'ubicos naturales y algunos casos de salida.

from math import *


def CubicSplines(datos):
    n = len(datos)-1
    # Inicializar vectores auxiliares
    A = [x[1] for x in datos]
    X = [x[0] for x in datos]
    H = [0.0 for x in range(n)]
    B = [0.0 for x in range(n+1)]
    C = [0.0 for x in range(n+1)]
    D = [0.0 for x in range(n+1)]
    alpha = [0.0 for x in range(n)]
    mu = [0.0 for x in range(n+1)]
    l = [1.0 for x in range(n+1)]
    z = [0.0 for x in range(n+1)]

    # Crear vector H
    for i in range(n):
        H[i] = X[i+1]-X[i]

    # Crear vector alpha
    for i in range(1, n):
        alpha[i] = (3.0/H[i])*(A[i+1]-A[i])-(3.0/H[i-1])*(A[i]-A[i-1])

    # Solucionar sistema tridiagonal
    for i in range(1, n):
        l[i] = 2.0*(X[i+1]-X[i-1])-H[i-1]*mu[i-1]
        mu[i] = float(H[i])/l[i]
        z[i] = (alpha[i]-H[i-1]*z[i-1])/float(l[i])

    # Solucionar sistema tridiagonal
    for j in range(n-1, -1, -1):
        C[j] = z[j]-mu[j]*C[j+1]
        B[j] = (A[j+1]-A[j])/float(H[j])-H[j]*(C[j+1]+2*C[j])/3.0
        D[j] = (C[j+1]-C[j])/(3.0*H[j])

    # Retornar vectores A, B, C, D
    return A[:-1], B[:-1], C[:-1], D[:-1]

# Datos de prueba (1,2), (2,3), (3,5)
datosPrueba = [(1.0, 2.0), (2.0, 3.0), (3.0, 5.0)]
a, b, c, d = CubicSplines(datosPrueba)
print "\nVectores de coeficientes:"
print "A =", a
print "B =", b
print "C =", c
```

```
69 print "D =", d
70
71 # Datos de prueba (0,1), (1,e), (2,e^2) y (3,e^3)
72 datosPrueba = [(0.0, exp(0.0)), (1.0, exp(1.0)),
73                (2.0, exp(2.0)), (3.0, exp(3.0))]
74 a, b, c, d = CubicSplines(datosPrueba)
75 print "\nVectores de coeficientes:"
76 print "A =", a
77 print "B =", b
78 print "C =", c
79 print "D =", d
```

Salida

```
Vectores de coeficientes:
A = [2.0, 3.0]
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

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # ------------------------------------------------------------------------
5  # Compendio de algoritmos.
6  # Matemáticas para Ingeniería. Métodos numéricos con Python.
7  # Copyright (C) 2017  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 recta de mínimos cuadrados.
```

```
21
22 from math import *
23
24
25 def RectaMinSq(datos):
26     X = sum([p[0] for p in datos])
27     Y = sum([p[1] for p in datos])
28     XX = sum([(p[0])**2 for p in datos])
29     XY = sum([p[0]*p[1] for p in datos])
30     m = len(datos)
31
32     def P(x):  # recta de mínimos cuadrados
33         a0 = float(Y*XX-X*XY)/float(m*XX-X**2)
34         a1 = float(m*XY-X*Y)/float(m*XX-X**2)
35         return a0+a1*x
36     return P
37
38
39 def ErrorSq(f, datos):  # Error cuadrático
40     E = sum([(p[1]-f(p[0]))**2 for p in datos])
41     return E
42
43 # datos de prueba
44 datos = [(-1.0, 2.0), (0.0, -1.0), (1.0, 1.0), (2.0, -2.0)]
45 f = RectaMinSq(datos)
46 print "Evaluar en x=1:"
47 print f(1.0)
48 print r"Error cuadr\'atico:"
49 print ErrorSq(f, datos)  # calcular error cuadrático
50 print "\n"
51
52 # datos de prueba
53 datos = [(1.0, 1.3), (2.0, 3.5), (3.0, 4.2), (4.0, 5.0), (5.0, 7.0),
54          (6.0, 8.8), (7.0, 10.1), (8.0, 12.5), (9.0, 13.0), (10.0, 15.6)]
55 print "Evaluar en x=1:"
56 print f(1.0)
57 print r"Error cuadr\'atico:"
58 print ErrorSq(f, datos)  # calcular error cuadr\'atico
59 print "\n"
```

Salida

```
Evaluar en x=1:
-0.5
Error cuadr\'atico:
5.0


Evaluar en x=1:
-0.5
```

```
Error cuadr\'atico:
2249.94
```

---

## B.3. Sistemas de ecuaciones

### B.3.1. *LU*

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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):
    n = len(A)
    # crear matrices nulas
    L = [[0.0 for x in range(n)] for x in range(n)]
    U = [[0.0 for x in range(n)] for x in range(n)]

    # Doolittle
    L[0][0] = 1.0
    U[0][0] = float(A[0][0])

    if abs(L[0][0]*U[0][0]) <= 1e-15:
        print "Imposible descomponer"
        return
```

```
39
40      for j in range(1, n):
41          U[0][j] = A[0][j]/L[0][0]
42          L[j][0] = A[j][0]/U[0][0]
43
44      for i in range(1, n-1):
45          L[i][i] = 1.0
46          s = sum([L[i][k]*U[k][i] for k in range(i)])
47          U[i][i] = float(A[i][i])-float(s)
48
49          if abs(L[i][i]*U[i][i]) <= 1e-15:
50              print "Imposible descomponer"
51              return
52
53          for j in range(i+1, n):
54              s1 = sum([L[i][k]*U[k][j] for k in range(i)])
55              s2 = sum([L[j][k]*U[k][i] for k in range(i)])
56              U[i][j] = float(A[i][j])-float(s1)
57              L[j][i] = (float(A[j][i])-float(s2))/float(U[i][i])
58
59      L[n-1][n-1] = 1.0
60      s3 = sum([L[n-1][k]*U[k][n-1] for k in range(n)])
61      U[n-1][n-1] = float(A[n-1][n-1])-float(s3)
62
63      if abs(L[n-1][n-1]*U[n-1][n-1]) <= 1e-15:
64          print "Imposible descomponer"
65          return
66
67      print "Matriz L: \n"
68      pprint(L)
69      print
70      print "Matriz U: \n"
71      pprint(U)
72
73
74  A = [[4, 3], [6, 3]]
75  print "Matriz A: \n"
76  pprint(A)
77  print
78  lu(A)
79  print "---------"
80
81  A = [[0, 1], [1, 1]]
82  print "Matriz A: \n"
83  pprint(A)
84  print
85  lu(A)
86  print "---------"
87
```

```
88  A = [[3, 1, 6], [-6, 0, -16], [0, 8, -17]]
89  print "Matriz A:\n"
90  pprint(A)
91  print
92  lu(A)
93  print "---------"
94
95  A = [[1, 2, 3], [2, 4, 5], [1, 3, 4]]
96  print "Matriz A:\n"
97  pprint(A)
98  print
99  lu(A)
100 print "---------"
```

Salida

```
Matriz A:

[[4, 3], [6, 3]]

Matriz L:

[[1.0, 0.0], [1.5, 1.0]]

Matriz U:

[[4.0, 3.0], [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, 0.0, 0.0], [-2.0, 1.0, 0.0], [0.0, 4.0, 1.0]]

Matriz U:

[[3.0, 1.0, 6.0], [0.0, 2.0, -4.0], [0.0, 0.0, -1.0]]
---------
Matriz A:

[[1, 2, 3], [2, 4, 5], [1, 3, 4]]
```

```
Imposible descomponer
---------
```

### B.3.2. Jacobi

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ------------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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):
    return max([abs(x[i]-y[i]) for i in range(len(x))])


def Jacobi(A, b, x0, TOL, MAX):
    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
            s = sum([A[i][j]*x0[j] for j in range(n) if j != i])
            x[i] = (b[i]-float(s))/float(A[i][i])
        pprint(x)
        if distinf(x, x0) < TOL:
            print r"Soluci\'on encontrada"
```

```
44              return
45          k += 1
46          for i in range(n):
47              x0[i] = x[i]
48      print "Iteraciones agotadas"
49      return
50
51  A = [[2, 1], [5, 7]]
52  b = [11, 13]
53  x0 = [1, 1]
54  print "Matriz A: \n"
55  pprint(A)
56  print
57  print "Vector b: \n"
58  pprint(b)
59  print
60  print "Semilla x0: \n"
61  pprint(x0)
62  print "\n"+r"Iteraci\'on de Jacobi"
63  # TOL = 10^-5, MAX = 50
64  Jacobi(A, b, x0, 1e-5, 50)
65
66
67  A = [[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
68  b = [6, 25, -11]
69  x0 = [0, 0, 0]
70  print "\n"+"Matriz A: \n"
71  pprint(A)
72  print
73  print "Vector b: \n"
74  pprint(b)
75  print
76  print "Semilla x0: \n"
77  pprint(x0)
78  print "\n"+r"Iteraci\'on de Jacobi"
79  # TOL = 10^-10, MAX = 50
80  Jacobi(A, b, x0, 1e-10, 50)
```

Salida

```
Matriz A:

[[2, 1], [5, 7]]

Vector b:

[11, 13]

Semilla x0:
```

```
[1, 1]

Iteraci\'on 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]
Soluci\'on encontrada

Matriz A:

[[10, -1, 2], [-1, 11, -1], [2, -1, 10]]

Vector b:

[6, 25, -11]

Semilla x0:

[0, 0, 0]

Iteraci\'on 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, -1.081730769195627]
[1.0432692307598361, 2.269230769237108, -1.0817307692400526]
Soluci\'on encontrada
```

### B.3.3. Gauss-Seidel

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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):
    return max([abs(x[i]-y[i]) for i in range(len(x))])


```

```
30 def GaussSeidel(A, b, x0, TOL, MAX):
31     n = len(A)
32     x = [0.0 for x in range(n)]
33     k = 1
34     while k <= MAX:
35         for i in range(n):
36             if abs(A[i][i]) <= 1e-15:
37                 print "Imposible iterar"
38                 return
39             s1 = sum([A[i][j]*x[j] for j in range(i)])
40             s2 = sum([A[i][j]*x0[j] for j in range(i+1, n)])
41             x[i] = (b[i]-float(s1)-float(s2))/float(A[i][i])
42         pprint(x)
43         if distinf(x, x0) < TOL:
44             print r"Soluci\'on encontrada"
45             return
46         k += 1
47         for i in range(n):
48             x0[i] = x[i]
49     print "Iteraciones agotadas"
50     return
51
52 A = [[2, 1], [5, 7]]
53 b = [11, 13]
54 x0 = [1, 1]
55 print "Matriz A: \n"
56 pprint(A)
57 print
58 print "Vector b: \n"
59 pprint(b)
60 print
61 print "Semilla x0: \n"
62 pprint(x0)
63 print "\n"+r"Iteraci\'on de Gauss-Seidel"
64 # TOL = 10^-5, MAX = 50
65 GaussSeidel(A, b, x0, 1e-5, 50)
66
67
68 A = [[10, -1, 2], [-1, 11, -1], [2, -1, 10]]
69 b = [6, 25, -11]
70 x0 = [0, 0, 0]
71 print "\n"+"Matriz A: \n"
72 pprint(A)
73 print
74 print "Vector b: \n"
75 pprint(b)
76 print
77 print "Semilla x0: \n"
78 pprint(x0)
```

```
79 print "\n"+r"Iteraci\'on de Gauss-Seidel"
80 # TOL = 10^{-10}, MAX = 50
81 GaussSeidel(A, b, x0, 1e-10, 50)
```

Salida

```
Matriz A:

[[2, 1], [5, 7]]

Vector b:

[11, 13]

Semilla x0:

[1, 1]

Iteraci\'on 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]
Soluci\'on encontrada

Matriz A:

[[10, -1, 2], [-1, 11, -1], [2, -1, 10]]

Vector b:

[6, 25, -11]

Semilla x0:

[0, 0, 0]

Iteraci\'on 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]
Soluci\'on encontrada
```

## B.4. Ecuaciones diferenciales

### B.4.1. Euler

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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.

from math import *


def test1(t, y):  # dy/dt = y - t^2 + 1
    return y-t**2+1


def test2(t, y):  # dy/dt = 2 - e^(-4t) - 2y
    return 2.0-exp(-4*t)-2*y


def Euler(a, b, y0, f, N):
```

```
34      h = (b-a)/float(N)
35      t = a
36      w = y0
37      print t, "-->", w
38      for i in range(1, N+1):
39          w = w+h*f(t, w)
40          t = a+i*h
41          print t, "-->", w
42
43  # dy/dt = y - t^2 + 1, a = 0, b = 2, y0 = 0.5, N = 10
44  print "\n"+r"M\'etodo de Euler"+"\n"
45  Euler(0, 2, 0.5, test1, 10)
46
47  # dy/dt = 2 - e^(-4t) - 2y, a = 0, b = 1, y0 = 1, N = 20
48  print "\n"+r"M\'etodo de Euler"+"\n"
49  Euler(0, 1, 1, test2, 20)
```

Salida

```
M\'etodo de Euler

0 --> 0.5
0.2 --> 0.8
0.4 --> 1.152
0.6 --> 1.5504
0.8 --> 1.98848
1.0 --> 2.458176
1.2 --> 2.9498112
1.4 --> 3.45177344
1.6 --> 3.950128128
1.8 --> 4.4281537536
2.0 --> 4.86578450432

M\'etodo de Euler

0 --> 1
0.05 --> 0.95
0.1 --> 0.914063462346
0.15 --> 0.88914111381
0.2 --> 0.872786420624
0.25 --> 0.863041330356
0.3 --> 0.858343225262
0.35 --> 0.85744919214
0.4 --> 0.859374424729
0.45 --> 0.863342156356
0.5 --> 0.868742996309
0.55 --> 0.875101932517
0.6 --> 0.882051581347
0.65 --> 0.889310525548
```

```
0.7 --> 0.896665794082
0.75 --> 0.903958711543
0.8 --> 0.91107348697
0.85 --> 0.917928028074
0.9 --> 0.924466561769
0.95 --> 0.93065371947
1.0 --> 0.93646980893
```

### B.4.2. Verlet

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 Verlet y algunos casos de salida.

from math import *


def test1(x):  # d^2x/dt^2 = x
    return x


def test2(x):  # d^2x/dt^2 = -x
    return -x


def Verlet(a, b, x0, v0, f, N):
    h = (b-a)/float(N)
    p0 = x0
    p1 = p0 + v0*h + 0.5*f(p0)*h**2
    print a, "-->", p0
    for i in range(1, N+1):
```

```
39          p = 2.0*p1-p0+f(p1)*h**2
40          print a+i*h, "-->", p1
41          p0 = p1
42          p1 = p
43
44  # d^2x/dt^2 = x, a = 0, b = 1, x_0 = 1, v_0 = 1, N = 20
45  print "\n"+r"M\'etodo de Verlet"+"\n"
46  Verlet(0, 1.0, 1.0, 1.0, test1, 20)
47
48  # d^2x/dt^2 = -x, a = 0, b = 1, x_0 = 1, v_0 = 0, N = 20
49  print "\n"+r"M\'etodo de Verlet"+"\n"
50  Verlet(0, 1.0, 1.0, 0, test2, 20)
```

Salida

```
M\'etodo de Verlet

0 --> 1.0
0.05 --> 1.05125
0.1 --> 1.105128125
0.15 --> 1.16176907031
0.2 --> 1.2213144383
0.25 --> 1.28391309238
0.3 --> 1.3497215292
0.35 --> 1.41890426984
0.4 --> 1.49163427115
0.45 --> 1.56809335814
0.5 --> 1.64847267853
0.55 --> 1.73297318061
0.6 --> 1.82180611564
0.65 --> 1.91519356597
0.7 --> 2.0133690002
0.75 --> 2.11657785694
0.8 --> 2.22507815832
0.85 --> 2.3391411551
0.9 --> 2.45905200476
0.95 --> 2.58511048444
1.0 --> 2.71763174033

M\'etodo de Verlet

0 --> 1.0
0.05 --> 0.99875
0.1 --> 0.995003125
0.15 --> 0.988768742188
0.2 --> 0.98006243752
0.25 --> 0.968905976758
0.3 --> 0.955327251054
0.35 --> 0.939360207223
```

```
0.4 --> 0.921044762873
0.45 --> 0.900426706617
0.5 --> 0.877557583594
0.55 --> 0.852494566612
0.6 --> 0.825300313213
0.65 --> 0.796042809032
0.7 --> 0.764795197827
0.75 --> 0.731635598629
0.8 --> 0.696646910433
0.85 --> 0.659916604962
0.9 --> 0.621536507978
0.95 --> 0.581602569724
1.0 --> 0.540214625046
```

### B.4.3. RK4

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ----------------------------------------------------------------------
# Compendio de algoritmos.
# Matemáticas para Ingeniería. Métodos numéricos con Python.
# Copyright (C) 2017  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 *


def test1(t, y):  # dy/dt = y - t^2 + 1
    return y-t**2+1


def test2(t, y):  # dy/dt = 2 - e^(-4t) - 2y
    return 2.0-exp(-4*t)-2*y


```

```
33 def RK4(a, b, y0, f, N):
34     h = (b-a)/float(N)
35     t = a
36     w = y0
37     print t, "-->", w
38 
39     for i in range(1, N+1):
40         k1 = h*f(t, w)
41         k2 = h*f(t+h/2.0, w+k1/2.0)
42         k3 = h*f(t+h/2.0, w+k2/2.0)
43         k4 = h*f(t+h, w+k3)
44         w = w+(k1+2.0*k2+2.0*k3+k4)/6.0
45         t = a+i*h
46         print t, "-->", w
47 
48 # dy/dt = y - t^2 + 1, a = 0, b = 2, y0 = 0.5, N = 10
49 print "\n"+r"M\'etodo RK4"+"\n"
50 RK4(0, 2, 0.5, test1, 10)
51 
52 # dy/dt = 2 - e^(-4t) - 2y, a = 0, b = 1, y0 = 1, N = 20
53 print "\n"+r"M\'etodo RK4"+"\n"
54 RK4(0, 1, 1, test2, 20)
```

Salida

```
M\'etodo RK4

0 --> 0.5
0.2 --> 0.829293333333
0.4 --> 1.21407621067
0.6 --> 1.64892201704
0.8 --> 2.12720268495
1.0 --> 2.64082269273
1.2 --> 3.17989417023
1.4 --> 3.73234007285
1.6 --> 4.28340949832
1.8 --> 4.81508569458
2.0 --> 5.30536300069

M\'etodo RK4

0 --> 1
0.05 --> 0.956946773927
0.1 --> 0.925794826349
0.15 --> 0.903996935703
0.2 --> 0.88950471587
0.25 --> 0.880674661873
0.3 --> 0.876191562614
0.35 --> 0.875006100539
```

```
0.4 --> 0.876284037659
0.45 --> 0.879364861493
0.5 --> 0.883728152457
0.55 --> 0.888966251604
0.6 --> 0.894762067259
0.65 --> 0.900871071482
0.7 --> 0.907106710988
0.75 --> 0.91332859923
0.8 --> 0.919432972496
0.85 --> 0.925344987868
0.9 --> 0.931012518523
0.95 --> 0.93640116533
1.0 --> 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