Introducción al Análisis Numérico

Mario De León Urbina

mario.deleon@ucr.ac.cr

EMate

Escuela de Matemática

Universidad de Costa Rica

22 de febrero de 2024

Índice general

Motivación

En el colegio a menudo utilizamos calculadoras científicas para realizar cálculos numéricos y algebraicos. Pero, ¿se ha preguntado usted cómo es que funciona su calculadora, al menos matemáticamente hablando? Usemos la CASIO fx-570ES PLUS para mostrar lo siguiente.

- Si ponemos 10^{99} el output que aparece es 1×10^{99}, pero si ponemos 10^{100} aparece el mensaje `Math ERROR`. ¿Será que la calculadora no puede almacenar números de esta magnitud "tan grande"?

- Ahora, si ponemos 10^{-99} el output que aparece es 1×10^{-99}, pero si ponemos 10^{-100} el resultado es igual a 0. ¿Será que la calculadora no puede almacenar números de esta magnitud "tan pequeña"?

- Al digitar $1 + 10^{-9}$ obtenemos como output `1.000000001`. En cambio, si digitamos $1 + 10^{-10}$ obtenemos como output `1`. ¿No se supone que $1 + \varepsilon \neq 1$ si $\varepsilon > 0$?

- Consideremos la ecuación $x^2 + 100\,000x + 1 = 0$. Si usamos el `Mode`$\rightarrow 5 \rightarrow 3$ obtenemos que $x_1 = -1 \times 10^{-5}$ y $x_2 = -99999.99999$. Las soluciones exactas, calculadas por métodos conocidos son

  $$x_1 = \sqrt{2\,499\,999\,999} - 50\,000 \approx -0.000010000000001000000000200000000005...$$

  $$x_2 = -\sqrt{2\,499\,999\,999} - 50\,000 \approx -99999.999989999999998999999999980000...$$

  Es notable que al tener una cantidad determinada de dígitos en el output las representaciones de números irracionales en la calculadora "eliminan" dígitos de la representación exacta. Por ejemplo, si la calculadora solo mostrara 2 o 3 dígitos en la pantalla, después del punto decimal se tendrá que $x_1 = 0$, lo cual no es del todo cierto pero tampoco del todo falso. ¿Qué tan "exactas" son las soluciones calculadas para la ecuación?

- En esta calculadora no podemos tratar ecuaciones polinomiales de grado mayor o igual que 4 en general, tampoco podemos "resolver" sistemas lineales de 4×4 o superiores.

A pesar de estas y otras limitaciones, la calculadora científica es una muy buena herramienta para realizar cálculos aritméticos y algebraicos básicos. Antes del gran avance que han tenido las computadoras, para realizar cálculos se utilizaban sendas tablas o reglas de cálculo, lo cual era tedioso en muchas ocasiones.

En MathWorks encontramos una definición de qué es el Análisis Numérico:

El **Análisis Numérico** es una rama de las matemáticas que resuelve problemas continuos mediante la aproximación numérica. Implica diseñar métodos que den soluciones numéricas aproximadas pero precisas, lo cual es útil en los casos en que la solución exacta es imposible o prohibitivamente costosa de calcular. El análisis numérico también implica caracterizar la convergencia, la precisión, la estabilidad y la complejidad computacional de estos métodos.

Entonces es importante utilizar un software que permita realizar algoritmos para problemas matemáticos formulados desde distintas áreas del conocimiento. En nuestro caso utilizaremos `MATLAB`:

`MATLAB` se usa ampliamente para el análisis numérico aplicado en ingeniería, finanzas computacionales y biología computacional. Proporciona una gama de métodos numéricos para:

- Interpolación, extrapolación y regresión
- Diferenciación e integración
- Sistemas lineales de ecuaciones
- Valores propios y valores singulares
- Ecuaciones diferenciales ordinarias (EDO)
- Ecuaciones diferenciales parciales (EDP)

También puede realizar transformadas rápidas de Fourier, cuadratura, optimización y programación lineal con la familia de productos `MATLAB`. Además, puede crear e implementar sus propios métodos numéricos utilizando el soporte integrado para operaciones vectoriales y matriciales en el lenguaje `MATLAB`.

Hay muchos textos sobre los antecedentes históricos del Análisis Numérico, recomiendo la lectura de los siguientes enlaces:

- `http://math.univ-lille1.fr/~bbecker/ano/pub/2001/ano428.pdf`
- `https://www.britannica.com/science/numerical-analysis/Historical-background`

Capítulo 1

Introducción a MATLAB

Utilizaremos MATLAB para realizar algunas de nuestros cálculos y ejercicios, así como la programación de algunos métodos numéricos.

MATLAB (MATrix LABoratory) es un software comercial que se utiliza tanto en la industria como en la academia para simulaciones numéricas, obtención y análisis de datos.

El siguiente resumen está basado en Bornemann ([**born**]) y en Echevarría ([**eche**]).

1.1. Comandos generales

- Help:

  `help`
  Muestra el texto de ayuda de un comando en la consola o buscador.

- Coma y semicolon:

  "," y ";" separan comandos. Semicolon (;) suprime la impresión del output.

- Información

  `whos`
  Da información sobre las variables guardadas en la memoria.

- Medición del tiempo de ejecución:

  `tic` *línea* `toc`
  Ejecuta la *línea* de código y mide el tiempo de ejecución requerido.

- Comentarios:

`%`
Sirve para escribir `%` comentarios para líneas de código.

1.2. Forma en la que se presenta la información

Por defecto, cuando se hace un cálculo en la ventana de comandos aparece el resultado asignando a la variable `ans` (answer). Los resultados numéricos son mostrados en notación fija decimal con 4 cifras decimales si su valor absoluto está comprendido entre 10^{-3} y 10^{3}. En caso contrario son mostrados en notación científica.

Se puede modificar este comportamiento mediante el comando `format`

`format`
Cambia el formato de salida a su valor por defecto, `short`

`format short`
El formato por defecto

`format long`
muestra 15 dígitos

`format short e`
formato `short`, en coma flotante.

`format long e`
formato `long`, en coma flotante.

`format rat`
muestra los números como cociente de enteros.

1.3. Matrices

`MATLAB` identifica escalares con matrices de 1×1, y vectores columna (fila) de dimensión m con matrices $m \times 1$ ($1 \times m$).

- Asignación:

`A =` *expresión*
Asigna a la variable `A` el valor de la expresión.

- Operaciones:

`+ , -, *` son las operaciones de matrices de adición, resta y multiplicación. `^` es potencia de matrices cuadradas.

`A/B` calcula la solución X del sistema lineal $A = XB$ y `A\B` calcula la solución X de $AX = B$.

- Operaciones componente a componente:

`.*, ./, .^` son las operaciones componente a componente.

- Matriz adjunta (transpuesta):

`A'` es la matriz adjunta de `A`.

- Input para matriz:

```
A = [ 1 2 3;
4 5 6 ];
```
O también `A = [ 1 2 3; 4 5 6 ];` asigna la siguiente matriz a la variable `A`:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

- Matriz identidad:

`eye(m)` es la matriz identidad de dimensión m.

- Matriz nula :

`zeros(m,n), zeros(m)`
es la matriz de entradas nulas de dimensión $m \times n$ y $m \times m$ respectivamente.

- Matriz de unos:

`ones(m,n), ones(m)`
es la matriz de entradas iguales a 1.

- Matrices aleatorias:

`rand(m,n), rand(m), randn(m,n), randn(m)`
generan matrices con entradas aleatorias.

- Vectores indexados:

`j:k, j:s:k`
son los vectores fila `[j, j+1, ..., k-1,k]` y `[j,j+s, ..., j+m*s]` con

$$m = \left[\!\left[\frac{k-j}{s}\right]\!\right]$$

- Vectores regularmente espaciados:

`linspace(a,b,n)`
Si a y b números reales, n entero, genera un vector fila de longitud n cuyo primer elemento es a y último elemento b, y los elementos están regularmente espaciados.

- Componentes:

`A(j,k)`
es la entrada A_{jk} de la matriz `A`.

- Bloques de matrices (submatrices):

`A(j1:j2, k1:k2)`
es la submatriz de `A` en la cual las fila va de $j1 \leq j2$ y las columnas van de $k1 \leq k2$.

`A(j1:j2, :), A(:, k1:k2)`
son las submatrices construidas desde las filas $j1$ a $j2$ y de las columnas $k1$ a $k2$ respectivamente.

`A(j,:), A(:,k)`
son el vector fila $j-$ésima y el vector columna $k-$ésima de la matriz `A`.

- Matriz como vector:

  `A(:)`
  transforma la matriz `A` $\in \mathbb{R}^{m\times n}$ en un vector $v \in \mathbb{R}^{mn}$.

- Matrices triangulares:

  `tril(A), triu(A)`
  extrae la parte triangular inferior y triangular superior de la matriz `A`.

- Diagonales de/en una matriz:

  `diag(A,k)`
  extrae como vector columna la diagonal $k-$ésima de la matriz `A`. La diagonal principal es $k = 0$ y se puede usar también `diag(A)`.

  `diag(v,k), diag(v)`
  si v es un vector entonces se genera una matriz de ceros en donde la diagonal k es el vector v.

- Dimensiones de una matriz:

  `size(A,1)`
  da el número de filas

  `size(A,2)`
  da el número de columnas.

  `size(A)`
  da un vector `v = [m, n]` donde m es el número de filas y n el número de columnas de `A`.

- Dimensión de un vector:

  `length(v)`
  da la dimensión de un vector columna o fila `v`.

1.4. Funciones

A veces a las funciones de MATLAB se les llama M-funciones.

- M-Funciones:

  `function [o_1, o_2, ..., o_n] = myfunc(i_1, i_2, ..., i_m)`
  donde `i_1, i_2, ..., i_m` representan las variables de inputs y `o_1, o_2, ..., o_n` representan las variables de outputs. La función puede ser guardada en un archivo `myfunc.m` en el directorio del proyecto. La función se puede llamar en la línea de comando (o en la creación de otras funciones) como:

  ```
  [o_1, o_2, ..., o_n] = myfunc(i_1, i_2, ..., i_m)
  ```

  Las variables de salida (outputs) podrían ignorarse reemplazándolas por $\sim$,
  por ejemplo:
  `[~ , o_2] = myfunc(i_1, i_2, ..., i_m)`

- Función anónima:

  `f = @(i_1, i_2, ..., i_m)` *expresión*
  define una función que asigna el valor de la expresión con los inputs `i_1, i_2, ..., i_m`. Al llamarla se ejecuta como

  ```
  o_1 = f(i_1, i_2, ..., i_m)
  ```

1.4.1. Funciones matemáticas elementales

Función		Función	
`sqrt(x)`	raíz cuadrada	`sin(x)`	seno (radianes)
`abs(x)`	módulo	`cos(x)`	coseno (radianes)
`conj(z)`	conjugado	`tan(x)`	tangente (radianes)
`real(z)`	parte real	`cotg(x)`	cotangente (radianes)
`imag(z)`	parte imaginaria	`asin(x)`	arcoseno
`exp(x)`	exponencial	`acos(x)`	arcocoseno
`log(x)`	logaritmo natural	`atan(x)`	arcotangente
`log10(x)`	logaritmo decimal	`cosh(x)`	cos hiperbólico
`rat(x)`	aprox. racional	`sinh(x)`	seno hiperbólico
`fix(x)`	redondeo hacia 0	`tanh(x)`	tang. hiperbólico
`ceil(x)`	redondeo hacia $+\infty$	`acosh(x)`	arccos hiperbólico
`floor(x)`	redondeo hacia $-\infty$	`asinh(x)`	arcsen hiperbólico
`round(x)`	redondeo al entero más próx.	`atanh(x)`	arctan hiperbólico

1.4.2. Funciones para cálculos matriciales

Función	
`sum(v)`	suma de las entradas de v vector
`sum(A)`	suma de las entradas de A matriz
`sum(A,1)`	suma de los elementos por columnas
`sum(A,2)`	suma de los elementos por filas
`prod(v)`	producto de las entradas de v vector
`prod(A)`	producto de las entradas de A
`max(v)`	máximo elemento de v
`min(v)`	mínimo elemento de v
`norm(v, p)`	norma $\|v\|_p$ vectorial/matricial
`norm(v, Inf)`	norma $\|v\|_\infty$ vectorial/matricial
`eig(A)`	espectro de A

1.5. Flujos de control

- Condicionales:

`if` *expresión*
    instrucciones
`elseif` *expresión*
    instrucciones
`else`
    instrucciones
`end`

Las instrucciones son ejecutadas si las partes reales de todas las entradas de la (matriz evaluada) expresión son no-cero; '`elseif`' y '`else`' son opcionales y múltiples '`elseif`' pueden utilizarse.

- Ciclos `for`:

`for` *Variable* = *vector*
    instrucciones
`end`

ejecuta instrucciones varias veces donde a la variable se le asigna sucesivamente el valor de las componentes del vector.

- Ciclos `while`:

`while` *expresión*
    instrucciones
`end`

ejecuta instrucciones repetidamente hasta que la parte real de todas las entradas de las (matriz evaluada) expresiones son no-cero.

- Interrupción de ciclos/procesos:

'`break`' detiene la ejecución de un `for` o de un `while`.

'`continue`' salta a la instancia siguiente de un `for` o de un `while`.

'`return`' deja la función en curso para retornar control a la función invocada.

1.6. Funciones lógicas

- Comparaciones:

  `A == B, A ~= B, A <= B, A >= B, A < B, A > B`
  comparaciones componente a componente de matrices; 1 es para 'verdadero' y 0 para 'falso'.

- Operaciones lógicas booleanas:

  `a && b, a || b`
  son los operadores lógicos 'y' y 'o', respectivamente.

  $\sim$ *expresión*, `not` (*expresión*)
  ambas retornan la negación lógica de la expresión.

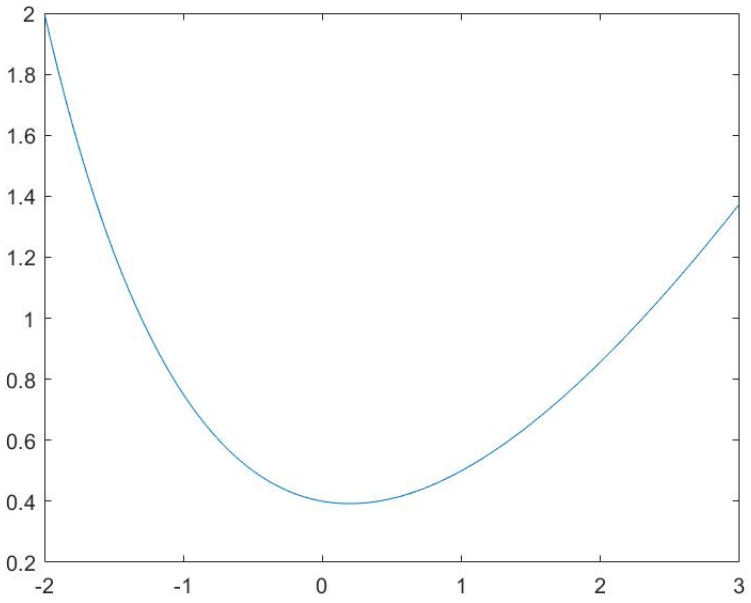
1.7. Graficación

Se muestran algunas funciones elementales de dibujo de `MATLAB`, cuyo uso permite realizar gráficas de funciones.

- Plot:

  `plot(x, y)`
  se puede utilizar junto con `linspace`. Por ejemplo, al graficar $y = \dfrac{x^2+2}{x+5}$ en el intervalo $[-2, 3]$:

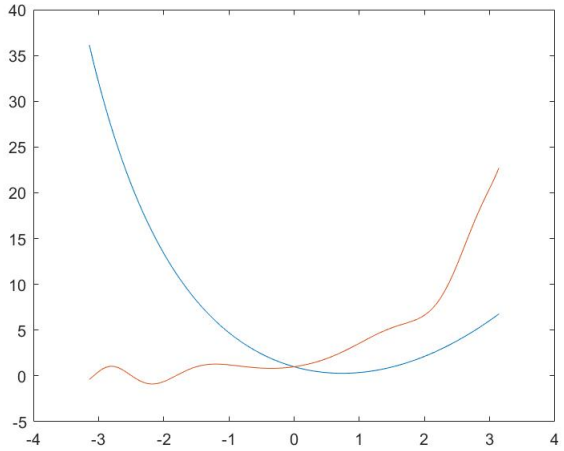
```
x = linspace(-2, 3);
y = (x.^2+2)./(x+5);
plot(x,y)
```




```
plot(x1, y1, x2, y2, x3, y3)
```

Se pueden dibujar dos o más curvas, proporcionando varios pares de vectores (abscisas y ordenadas) para cada una de las funciones en cuestión. Por ejemplo, al graficar $y = e^{-x} + x^2 - x, \quad y = e^x + \text{sen}(x^2)$:

```
x = linspace(-pi, pi);
y1 = exp(-x) + x.^2 -x;
y2 = exp(x) + sin(x.^2);
plot(x, y1, x, y2)
```



- Información de gráficas:

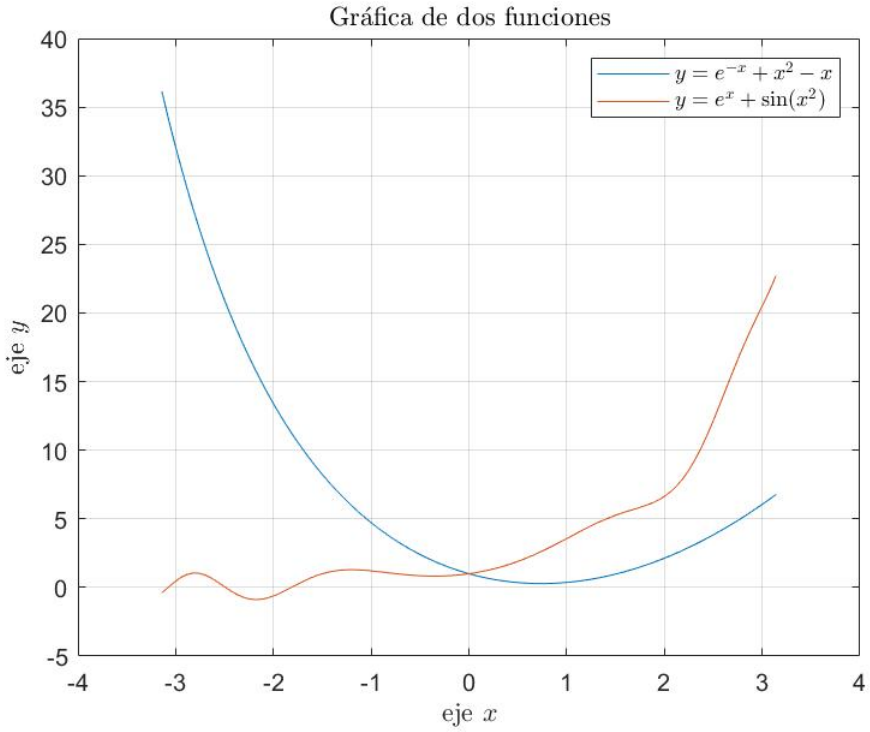
En la graficación anterior no se distingue cuál gráfica pertenece a cada criterio. Por esto se puede agregar información. Algunas funciones son:

- `title('título')`: añade título a la gráfica.
- `xlabel('eje_x')`: añade una etiqueta al eje x.
- `ylabel('eje_x')`: añade una etiqueta al eje y.
- `legend('f', 'g', ...)`: define etiquetas para las gráficas de las funciones involucradas.
- `grid`: muestra una cuadrícula.
- `text(x, y, 'texto')`: agrega texto en el lugar definido por el punto (x, y).

Al usar (`' '`) se puede utilizar el lenguaje LaTeX con ayuda de `'interpreter'`, y así digitar de forma matemática fórmulas y expresiones.
Por ejemplo, al graficar $y = e^{-x} + x^2 - x, \quad y = e^x + \text{sen}(x^2)$:

```
x = linspace(-pi, pi);
y1 = exp(-x) + x.^2 -x;
y2 = exp(x) + sin(x.^2);
plot(x, y1, x, y2), grid

% Informacion de la grafica
title('Gr\''afica de dos funciones', 'interpreter', 'latex'
   )
xlabel('eje $x$', 'interpreter', 'latex')
ylabel('eje $y$', 'interpreter', 'latex')
 l = legend('$y=e^{-x}+x^2-x$', '$y=e^x+\sin(x^2)$');
set(l, 'interpreter', 'latex')
```



1.8. Ejercicios

Se pueden agregar como ejercicios adicionales los de la página 70–72 de ECHEVARRÍA.[1]

1. Dado $x = 1$ encuentre el valor de y definido por

$$y = \left(\frac{x^2 - 3x + 2}{\frac{1}{2} + 5x}\right)^{2x+1} - \left[\frac{x^3 + \frac{1}{3}x^2 - \frac{5}{4}x + \pi}{(\frac{5}{6})^2(x-2) - \frac{1}{2x}}\right]^7$$

   Escriba en MATLAB un fragmento de código que lleve a cabo dicha evaluación.

2. Introduzca en MATLAB la matriz [2]

$$\mathbf{H} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{pmatrix}$$

   Utilice `format long` y `format rat`, los cuales puede investigar en la ayuda, para visualizar el resultado.

[1] https://personal.us.es/echevarria/documentos/APUNTESMATLAB.pdf
[2] Le puede servir saber qué es una matriz de HILBERT.

3. Utilice el operador dos puntos de `MATLAB` para crear los siguientes vectores.

   a) $\boldsymbol{w} = (-5, -4, ..., 8, 9)$

   b) $\boldsymbol{x} = (132, 129, ..., 3, 0)$

   c) $\boldsymbol{y} = \begin{pmatrix} 0.1 & 0.2 & ... & 9.9 & 10 \end{pmatrix}$

   d) $\boldsymbol{z} = \begin{pmatrix} 0 & \frac{\pi}{12} & ... & 4\pi \end{pmatrix}$

4. Dado el intervalo $I = [1, 5]$ utilice el operador dos puntos para crear un vector $\boldsymbol{x}$ que contenga n puntos igualmente espaciados de I, donde el primer punto sea 1 y el último sea 5. Luego ejecútelo con el valor de $n = 100$.

5. Introduzca en `MATLAB` la matriz

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

   y realice lo siguiente a dicha matriz:

   a) Extraiga la fila 2.

   b) Reemplace la fila 2 por el vector $(-1, -2, 0, 1)$.

   c) Reemplace la columna 3 por el vector nulo.

   d) Extraiga el bloque $\mathbf{B}$ definido por $2 \leq i \leq 3, \quad 3 \leq j \leq 4$.

   e) Reemplace el bloque $2 \leq i \leq 3$, $3 \leq j \leq 4$ de $\mathbf{A}$ por $\begin{pmatrix} -10 & -20 \\ -30 & -40 \end{pmatrix}$.

6. Considere la matriz

$$\mathbf{M} = \begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

   Utilice el manejo de bloques para construir dicha matriz en `MATLAB`.

7. Considere las siguientes matrices:

$$\mathbf{A} := \begin{pmatrix} 1 & 2 & 7 \\ 2 & 5 & -6 \\ 7 & 8 & -3 \\ -6 & -3 & 9 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 14 & 9 \\ 9 & 11 \\ 5 & 2 \end{pmatrix} \quad \text{y} \quad \mathbf{C} = \begin{pmatrix} -1 & 2 \\ 8 & 5 \\ 7 & -6 \end{pmatrix}$$

Luego, efectúe las siguientes concatenaciones:

$$\mathbf{D} = \begin{pmatrix} \mathbf{A}^{\mathtt{t}} & \mathbf{B} & \mathbf{C} \end{pmatrix}, \quad \mathbf{E} = \begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^{\mathtt{t}} & \mathbf{B} \end{pmatrix},$$

8. Escriba un fragmento de código que cree para todo $n \in \mathbb{N}$ la matriz tridiagonal

$$\mathbf{T}_n = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

9. Sea $\mathbf{A}$ matriz de $n \times n$ la cual se descompone de la forma $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, donde $\mathbf{L}$ es la parte estrictamente diagonal inferior, $\mathbf{D}$ la diagonal y $\mathbf{U}$ la parte estrictamente diagonal superior de $\mathbf{A}$. Escriba las instrucciones de MATLAB que le permitan obtener la matriz de iteración de Gauss–Seidel

$$\mathbf{B} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U},$$

para cualquier matriz $\mathbf{A}$ dada.

10. Considere la función $f : \mathbb{R} - \left\{ \frac{\sqrt[5]{48}}{2} \right\} \to \mathbb{R}$ definida por

$$f(x) = \frac{x^3(x+1)}{2x^5 - 3}$$

Escriba un fragmento de código que le permita evaluar la función f en múltiples valores de x almacenados en un vector.

11. Usando MATLAB resuelva el sistema

$$\begin{cases} x_1 + 4x_2 - 5x_3 + 7x_5 &= 7 \\ 4x_1 + 7x_2 + 8x_3 - x_4 - x_5 &= 17 \\ x_2 + 4x_3 + 5x_4 - 4x_5 &= 6 \end{cases}$$

12. Dada $\mathbf{A}$ matriz real de $n \times n$, implemente en MATLAB la función

$$f_{\mathbf{A}} = \mathbf{x}^{\mathrm{t}}\mathbf{A}\mathbf{x} + \frac{1}{2}\|\mathbf{A}\mathbf{x}\|^2$$

En particular, calcule $f_{\mathbf{A}}(\mathbf{x})$ para $x = (1, -1, -1, 1)^{\mathrm{t}}$, con $\mathbf{A} = \begin{pmatrix} 1 & 0 & -1 & 2 \\ -6 & 1 & 8 & -15 \\ -4 & 1 & 6 & -10 \\ -2 & 0 & 2 & -5 \end{pmatrix}$

13. Cree la siguiente función en un archivo .m en MATLAB:

```
function [fx] = mifun(x)
% v = mifun(x) devuelve el valor en x de la funcion
%           f(x) = 3*x^2+1 si x < -1
%           f(x) = 1-x^2 si no
fx = 3*x^2 + 1;
    if x >= -1
    fx = 1 - x^2;
    end
end
```

Describa dicha función. Luego calcule $(mifun(0))^3 + \sqrt{|mifun(-3)|}$.

14. Escriba el siguiente código en un nuevo *script*:

```
n1=input('Escribe un numero : ');
n2=input('Escribe otro numero : ');
mayor=n2;
   if (n1 > n2)
     mayor=n1;
   end
fprintf('El mayor es: %6.2f \n ',mayor);
```

Dicho código imprime en la pantalla el mayor número de dos números dados. Modifique el código y el mensaje de salida de tal forma que más bien se imprima el menor número de los dos.

15. Escriba el siguiente código en un nuevo *script*:

```
n1 = input(' Escribe un numero : ');
n2 = input(' Escribe otro numero : ');
   if (n1 < n2)
```

```
4       fprintf(' % 4i %4i \n', n1, n2)
5     else
6       fprintf(' % 4i %4i \n', n2, n1)
7     end
```

¿Qué hace dicho código?

16. Vamos a crear una M–función (función de MATLAB) tal que dados x, y reales distintos se muestre el vector ordenado $v = (\text{mín}\{x, y\}, \text{máx}\{x, y\})$. Cree la función `Orden.m` de la siguiente manera:

```
function [v] = Orden(n1, n2)
% v = Orden(n1, n2) es un vector que contiene los dos
% numeros n1 y n2 ordenados en orden creciente
%
    if n1 > n2
     v = [n2, n1];
    else
     v = [n1, n2];
    end
```

Pruebe dicha función `Orden(n1, n2)` con algunas parejas de números. ¿Qué sucede si $n1 = n2$?

17. Vamos a escribir una M-función que determin cuándo un número natural n es un cuadrado perfecto:

```
function [ ]=CuadradoPerfecto(n)
% Dicha funcion nos dice si n natural es
% cuadrado perfecto o no
m = sqrt(n);
    if floor(m) == m
     disp('es cuadrado perfecto');
    else
     disp('no es cuadrado perfecto');
    end
  end
```

Use dicha función para determinar si 225, 1000, 196, 289, 12769 son cuadrados perfectos o no.

18. Una *matriz tridiagonal* es una matriz cuadrada cuyos elementos son solo distintos de cero en la diagonal principal y las diagonales adyacentes por encima y por debajo de esta. En el caso de `MATLAB` sería que las diagonales `k = -1, 0, 1` no son nulas. Escriba el siguiente código para crear la función `Tridiag.m`:

```
function [] = Tridiag(A)
% La funcion verifica si A es o no tridiagonal
B = diag(diag(A)) + diag(diag((A,1),1) + diag(diag((A,-1),-1);
    if A == B
      disp('Es tridiagonal');
    else
      disp('no es tridiagonal');
    end
end
```

Compruebe dicha función con las siguientes matrices:

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

19. Cree una función en `MATLAB` que identifique cuándo una matriz A es simétrica o no.

20. El **Método de Horner** se usa para evaluar polinomios en una variable de manera eficiente. Dado el polinomio

$$p(x) = a_1 x^{n-1} + a_2 x^{n-2} + a_3 x^{n-3} + ... + a_{n-2} x^2 + a_{n-1} x + a_n$$

con $a_i \in \mathbb{R}$, la idea es calcular $p(x_0)$. Se puede reescribir el polinomio como

$$p(x) = x(x(...(x(a_1 x + a_2) + a_3) + ... + a_{n-2}) + a_{n-1}) + a_n$$

Entonces el método se puede definir como

$$\begin{cases} v_1 &= & a_1 \\ v_2 &= & v_1 x_0 + a_2 \\ v_3 &= & v_2 x_0 + a_3 \\ & \vdots & \\ v_n &= & v_{n-1} x_0 + a_n \end{cases}$$

donde $v_n = p(x_0)$. En MATLAB podemos implementarlo de la siguiente manera:

```
function [v] = Horner(a,x0)
% La funcion evalua x0 en el polinomio [a1 a2 ... an]
%
n=length(a);
v=a(1);
   for k=2:n
      v=v*x0 + a(k);
   end
end
```

Use dicha función para evaluar x_0 en los siguientes polinomios:

a) $p(x) = -3x^2 + 2x + 100$; $x_0 = 2$

b) $p(x) = x^4 - x^3 + 2x - 12$; $x_0 = -1.5$

c) $p(x) = 0.3x^6 - 0.03x^3 + 0.21$; $x_0 = 0.01$

d) $p(x) = x^6 - x^{11} - 8x^2 - x^9$; $x_0 = -1.1$

21. Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

La función debe emitir mensaje de error en los casos en que no se pueda calcular el área:

a) Si alguna de las longitudes recibidas es menor o igual a cero;

b) Si el radicando es negativo.

22. Escriba una M-función `function [y] = Traza(A)` que calcule la traza de una matriz cuadrada A. Debe emitir mensaje de error cuando la matriz no es cuadrada. Recuerde que la traza de $A \in \mathbb{R}^{n\times n}$ es igual a

$$\text{tr}(A) = a_{11} + a_{22} + ... + a_{nn}.$$

23. Escriba una M-función en MATLAB para la siguiente función:

$$f(x) = \begin{cases} \log_2 |x^2 + 2x - 3|, & \text{si } x < 1 \\ \frac{1}{2}\tan(3x - 2), & \text{si } x \geq 1 \end{cases}$$

24. Escriba una M-función `function Donde(x,a,b)` que reciba como argumentos de entrada $x \in \mathbb{R}$ y los extremos de un intervalo $[a, b]$, y escriba en la pantalla un mensaje indicando si $x < a$, $x \in [a, b]$ o $x > b$.

25. Escriba una M-función `function [vmax] = Mayor(v)` que reciba como argumento un vector v y devuelva el máximo entre los valores absolutos de todas sus componentes.

26. Escriba una M-función `function [amax] = MayorM(A)` que reciba como argumento una matriz A y devuelva el máximo entre los valores absolutos de todas sus componentes.

27. Programe de alguna forma la sucesión de Fibonacci con términos iniciales f_1, f_2 cualesquiera tales que pueda obtener el término $n-$ésimo f_n de dicha sucesión.

28. En `MATLAB`, un polinomio se representa mediante un vector que contiene los valores de sus coeficientes: el polinomio

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

se representa como el vector fila $c = [a_1, a_2, ..., a_n, a_{n+1}]$. Escriba una M-función `function [dc] = derpol(c)` que devuelva el vector `dc` de coeficientes de la derivada del polinomio.

Capítulo 2

Aritmética de precisión finita

2.1. Números punto flotante

Una computadora almacena una cantidad finita de bits. Esto quiere decir que tendrá representaciones aproximadas de los números reales. Para esto es importante el concepto de números máquina y representación punto flotante.

Definición 2.1 (Representación punto flotante)**.** Sea $\beta \geq 2$ una base (usualmente par), t la precisión (número de dígitos en base β). Un número **punto flotante** se representa en esta base como[1]

$$\xi = \pm(d_1.d_2...d_t)_\beta \times \beta^e = \pm m \times \beta^e$$

en donde $d_k \in \{0, 1, ..., \beta-1\}$ son los **dígitos significativos,** m **mantisa** y tiene t dígitos; $e \in \mathbb{Z}$ es el **exponente**. Si $d_1 \neq 0$ el número se dice **normalizado**. Un número punto flotante normalizado tal que $d_1 = 0$ implica entonces que $d_2 = d_3 = ... = d_t = 0$

△

Note que dicho número es igual a

$$\pm(d_1 + d_2\beta^{-1} + d_3\beta^{-2} + ... + d_t\beta^{-(t-1)}) \times \beta^e, \quad 0 \leq d_i < \beta$$

Si el exponente de dos números punto flotante es el mismo, se dice que tienen la misma *magnitud*. Los exponentes máximo y mínimo se denotan como $e_{\max}$ y $e_{\min}$, respectivamente, y se tiene que (usualmente) $e_{\min} < 0 < e_{\max}$. Entonces hay $e_{\max} - e_{\min} + 1$ posibles exponentes. El $+1$ es para el exponente cero.

[1] En otros libros de texto se utiliza la notación $\pm(0.d_1d_2...d_t)_\beta \times \beta^e$. Acá usamos la de [**Faul**] que es familiar porque se relaciona con la notación científica en base 10 usual.

De manera alternativa podemos representar ξ de manera única en la forma

$$\xi = \pm m \times \beta^{e+1-t}, \qquad m \in \{\beta^{t-1}, \beta^{t-1}+1, ..., \beta^t - 1\} \subset \mathbb{N}, \quad e \in \mathbb{Z}$$

Definición 2.2. Sean β, $t \in \mathbb{N}$, $0 \leq d_i < \beta$ para $i > 1$, $d_1 \neq 0$ y $L = e_{\min}$, $U = e_{\max}$. Se define *el conjunto normalizado de números reales de precisión finita*[2] como

$$\mathbb{F}(\beta, t, L, U) := \{0\} \cup \left\{ \xi \in \mathbb{R}^* : \xi = \pm\beta^e \times \sum_{i=1}^{t} d_i \beta^{1-i} \right\}$$

Cuando no haya ambigüedad dicho conjunto se denotará como $\mathbb{F}_{\beta,t}$.

Teorema 2.1. *La cantidad de elementos en $\mathbb{F}(\beta, t, L, U)$ es exactamente*

$$\#\mathbb{F}(\beta, t, L, U) = 2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

Además, $x_{min} = \beta^L$ y $x_{max} = (1 - \beta^{-t})\beta^{U+1}$, siendo estos elementos el valor mínimo y máximo positivos de $\mathbb{F}_{\beta,t}$, respectivamente.

Demostración: Sea $\xi = (-1)^s \times (d_1.d_2 d_3 ... d_{t-1} d_t)_\beta \times \beta^e \in \mathbb{F}_{\beta,t}$, con ξ distinto de cero. Se tiene que $s \in \{0, 1\}$, $d_1 \in \{1, 2, ..., \beta - 1\}$, $d_i \in \{0, 1, ..., \beta - 1\}$ para $i = 2 : t$, $e \in \{L, L-1, ..., 0, ..., U-1, U\}$, entonces hay $2(\beta - 1)\beta^{t-1}(U - L + 1)$ de escoger ξ. Si sumamos a esto que $\xi = 0$ también está en $\mathbb{F}_{\beta,t}$ se tiene enconces que $\#\mathbb{F}(\beta, t, L, U) = 2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$.

Para la segunda parte, se escoge $s = 0$. El valor mínimo se calcula tomando $d_1 = 1, d_2 = d_3 = ... = d_t = 0$ y $e = L$, por lo tanto, $x_{\min} = 1 \times \beta^L = \beta^L$.

Para el valor máximo se toman $d_1 = d_2 = ... = d_t = \beta - 1$, $e = U$. Luego,

$$\begin{aligned}
x_{\max} &= (\beta - 1) \times (1 + \beta^{-1} + \beta^{-2} + ... + \beta^{-(t-1)}) \times \beta^U \\
&= (\beta - 1) \times \sum_{k=0}^{t-1} \left(\tfrac{1}{\beta}\right)^k \times \beta^U \\
&= (\beta - 1) \times \frac{1 - (\frac{1}{\beta})^{t-1+1}}{1 - \frac{1}{\beta}} \times \beta^U \\
&= \cancel{(\beta - 1)} \times \frac{\beta^t - 1}{\beta^t} \cdot \frac{\beta}{\cancel{\beta - 1}} \times \beta^U \\
&= (1 - \beta^{-t}) \cdot \beta^{U+1}
\end{aligned}$$

Se concluyen los resultados. □

[2] O *machine numbers set.*

Ejemplo 2.1. Considere el conjunto, $\mathbb{F}(2,3,-1,1)$. Calcule la cantidad de elementos de dicho conjunto y elabore una lista de todos los elementos. Además represéntelos en la recta numérica.
Solución: En este caso tenemos que

$$\mathbb{F}(2,3,-1,1) = \{0\} \cup \left\{ \xi = \pm(1.b_2b_3)_2 \times 2^e,\ b_i \in \{0,1\},\ e \in \{-1,0,1\} \right\}$$

La cantidad de elementos es $\#\mathbb{F}(2,3,-1,1) = 2(1)\cdot 2^{3-1}(1-(-1)+1)+1 = 25$ de los cuales 12 de ellos son positivos y son los que calcularemos, junto con el cero. Hay $2^{3-1} = 4$ formas para escoger los dígitos b_2 y b_3, las cuales son

$$1.00, \quad 1.01, \quad 1.10, \quad 1.11$$

Los exponentes se pueden escoger de $U-L+1 = 1-(-1)+1 = 3$ formas. Entonces con $e = -1, 0, 1$ calculamos: $(1.00)_2 \times 2^e$, $(1.01)_2 \times 2^e$, $(1.10)_2 \times 2^e$, $(1.11)_2 \times 2^e$

$$\begin{array}{lll} (1.00)_2 \times 2^{-1} & (1.00)_2 \times 2^0 & (1.00)_2 \times 2^1 \\ (1.01)_2 \times 2^{-1} & (1.01)_2 \times 2^0 & (1.01)_2 \times 2^1 \\ (1.10)_2 \times 2^{-1} & (1.10)_2 \times 2^0 & (1.10)_2 \times 2^1 \\ (1.11)_2 \times 2^{-1} & (1.11)_2 \times 2^0 & (1.11)_2 \times 2^1 \end{array}$$

Expresándo la forma desarrollada y simplificando, se tiene que los números en base 10, incluyendo signos, son:

$$\begin{array}{lll} \pm 0.5 & \pm 1 & \pm 2 \\ \pm 0.625 & \pm 1.25 & \pm 2.5 \\ \pm 0.75 & \pm 1.5 & \pm 3 \\ \pm 0.875 & \pm 1.75 & \pm 3.5 \end{array}$$

Gráficamente los puntos se representan de la siguiente manera:

Ejercicio 2.1.1. Para el ejemplo 2.1 utilice `MATLAB` para crear un vector que ordene de menor a mayor los elementos de $\mathbb{F}_{\beta,t}$. Además grafique dichos elementos en la recta numérica.

Observación 2.1. Hay dos razones por las cuales un número real x no puede ser representado exactamente como un número punto flotante. La primera se ilustra con el número en base diez 0.1, el cual tiene una representación decimal finita, pero en binario su representación es $(0.0\overline{0011})_2$, luego dicho número en binario se encuentra estrictamente entre dos números punto flotante y a

la vez no es ninguno de ellos. La segunda razón es que el número $x \in \mathbb{R}$ podría ser muy grande o muy pequeño en valor absoluto y se escape del rango. A esto se le conoce como **overflow** (sobredesbordamiento) o **underflow** (subdesbordamiento) respectivamente.

Por defecto, las computadoras "atrapan" esos casos limítrofes asignando los valores $\pm\infty$ y cero sin previo aviso. ▲

2.1.1. Standard de la IEEE

La IEEE 754 reconoce los siguientes símbolos:

- $\pm\infty$, i.e. como el valor de $\pm 1/0$;
- `NaN` *not a number*, i.e. como el resultado de $0/0$ o $\infty - \infty$.

Esencialmente cada computadora desde 1985 implementa el estandard IEEE 754, el cual ofrece dos formatos binarios para la *hardware arithmethic* (por defecto `MATLAB` usa doble precisión, pero también permite el uso de precisión simple). La idea es representar un número asignando los bits de la siguiente manera:

signo	exponente	mantisa

- Precisión simple: utiliza 32 bits (4 bytes) distribuidos así: 1 bit para el signo, 8 bits para el exponente y 23 bits para la mantisa. Tiene una precisión de 24 dígitos binarios. El exponente toma valores en $[-126, 127]$ con sesgo 127.

$\pm$	$a_1a_2a_3\ldots a_8$	$b_1b_2b_3\ldots b_{23}$

Si $a_1a_2a_3\ldots a_8$ es	luego el valor representado es
$(00000000)_2 = (0)_{10}$	$\pm\,(0.b_1b_2\ldots b_{23})_2 \times 2^{-126}$
$(00000001)_2 = (1)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{-126}$
$(00000010)_2 = (2)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{-125}$
$\vdots$	$\vdots$
$(01111111)_2 = (127)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{0}$
$(10000000)_2 = (128)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{1}$
$\vdots$	$\vdots$
$(11111101)_2 = (253)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{126}$
$(11111110)_2 = (254)_{10}$	$\pm\,(1.b_1b_2\ldots b_{23})_2 \times 2^{127}$
$(11111111)_2 = (255)_{10}$	$\pm\infty$ si $b_1 = b_2 = \ldots = b_{23} = 0$, `NaN` en otro caso

El valor de e que se guarda en los bits $a_1, a_2, ..., a_8$ corresponde a $e+127$, y el épsilon máquina es $\varepsilon_m = 2^{-23}$.

- Precisión doble: utiliza 64 bits (8 bytes) distribuidos así: 1 bit para el signo, 11 bits para el exponente y 52 bits para la mantisa. Tiene una precisión de 53 dígitos binarios. El exponente toma valores en $[-1022, 1023]$ con sesgo 1023.

$\pm$	$a_1a_2a_3\ldots a_{11}$	$b_1b_2b_3\ldots b_{52}$

Si $a_1a_2a_3\ldots a_{11}$ es	luego el valor representado es
$(00000000000)_2 = (0)_{10}$	$\pm\,(0.b_1b_2\ldots b_{52})_2 \times 2^{-1022}$
$(00000000001)_2 = (1)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{-1022}$
$(00000000010)_2 = (2)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{-1021}$
$\vdots$	$\vdots$
$(01111111111)_2 = (1023)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{0}$
$(10000000000)_2 = (1024)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{1}$
$\vdots$	$\vdots$
$(11111111101)_2 = (2045)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{1022}$
$(11111111110)_2 = (2046)_{10}$	$\pm\,(1.b_1b_2\ldots b_{52})_2 \times 2^{1023}$
$(11111111111)_2 = (2047)_{10}$	$\pm\infty$ si $b_1 = b_2 = \ldots = b_{52} = 0$, `NaN` en otro caso

El valor de e que se guarda en los bits $a_1, a_2, a_3, ..., a_{11}$ corresponde a e^{1023}. Además, la precisión es $p = 53$ y el épsilon máquina es $\varepsilon_m = 2^{-52}$

Ejemplo 2.2. Determine una fórmula para convertir números en formato de precisión simple a números en base 10. Use dicha fórmula para mostrar que

0	01111100	01000000000000000000000	$= 0.15625$

Solución: precisión simple utiliza 32 bits (4 bytes) distribuidos así: 1 bit para el signo, 8 bits para el exponente y 23 bits para la mantisa. Tiene una precisión de 24 dígitos binarios. El exponente toma valores en $[-126, 127]$ con sesgo 127.

s	$a_1a_2a_3\ldots a_8$	$b_1b_2b_3\ldots b_{23}$

Sea N la representación en base 10 del número anterior, entonces se tiene que

$$N = (-1)^s \times (1.b_1b_2b_3...b_{23})_2 \times 2^{E-127}$$

donde $s \in \{0, 1\}$ es el signo, $E = (a_1a_2a_3...a_8)_2$ es el exponente sesgado. En este caso,

$$E = (01111100)_2 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = 124$$

Usando lo anterior se tiene que el número representado es

$$N = (-1)^0 \times (1.01)_2 \times 2^{124-127} = (1 + 2^{-2}) \times 2^{-3} = 0.15625$$

Ejercicio 2.1.2. Escriba la fórmula para convertir un número en precisión doble a su representación en base 10.

2.2. Aproximaciones y errores

Cuando se representa un número real x por medio de otro número real $\hat{x}$ es importante asignar una forma de "medir" qué tan "parecido" es $\hat{x}$ a x. Por esto se necesitan las nociones de errores de aproximación.

Definición 2.3. Sea ξ una aproximación de x. Se define el *error absoluto* $\varepsilon \geq 0$ como

$$\xi = x + \varepsilon,$$

y el *error relativo* $\delta \geq 0$ como

$$\xi = x(1 + \delta)$$

Observación 2.2. De lo anterior, $\varepsilon = x\delta$, o, si $x \neq 0$, $\delta = \dfrac{\varepsilon}{x}$. Además se tiene que $\varepsilon = |x - \xi|$ y $\delta = \dfrac{|x - \xi|}{|x|}$. ▲

Para una base β se tiene que el error relativo, al aproximar x por algún valor, con precisión t, está acotado de la siguiente manera:

$$|\delta| = \left|\frac{\varepsilon}{x}\right| < \frac{1/2 \times \beta^{e-t+1}}{1 \times \beta^e} = 1/2 \times \beta^{-t+1}$$

En el caso de $\beta = 10$, base 10, se tiene la siguiente definición:

Definición 2.4. Se dice que ξ aproxima a x con p dígitos significativos si p es el mayor entero no negativo para cual se cumple que $\delta < \dfrac{1}{2} \times 10^{1-p} = 5 \times 10^{-p}$.

Se observa que basta tomar

$$p = \left[\!\left[\log_{10} \frac{5}{\delta} \right]\!\right]$$

debido a lo siguiente:

$$\begin{aligned} \frac{|x-\xi|}{|x|} = \delta &< 5 \times 10^{-p} \\ \log_{10}(\delta) &< \log_{10}(5) - p \\ p &< \log_{10}(5) - \log_{10}(\delta) \\ p &< \log_{10} \frac{5}{\delta} \end{aligned}$$

Ejemplo 2.3. Considere $\pi = 3.14159265$ y la aproximación $\hat{\pi} = 3.141591$. Determine el error absoluto, error relativo y la cantidad de dígitos significativos teóricos con los que $\hat{\pi}$ aproxima a π.

Solución:

$$\varepsilon_{\text{abs}} = |\pi - \hat{\pi}| = |3.14159265 - 3.141591| = 1.65 \times 10^{-6}$$

$$\delta_{\text{rel}} = \frac{|\pi - \hat{\pi}|}{|\pi|} = \frac{|3.14159265 - 3.141591|}{|3.14159265|} = 5.2521 \times 10^{-7}$$

$$\delta_{\text{rel}} = 5.2521 \times 10^{-7} < 5 \times 10^{-6}$$

implicando esto último que $\hat{\pi}$ aproxima a π con a lo sumo $p = 6$ dígitos significativos.

Observación 2.3. En la aritmética punto flotante, el error relativo es muy apropiado porque cada número es representado con una precisión relativa similar. Cuando $x = 0$ o x es muy cercano a cero, es mejor considerar el error absoluto. ▲

2.2.1. Representaciones en $\mathbb{F}$

Supongamos que se está utilizando el sistema de precisión doble de la IEEE. Sea $\mathbb{F}$ como el conjunto de números en punto flotante del sistema de precisión doble. Considere la función $\texttt{fl} : \mathbb{R} \to \mathbb{F}$, donde a cada número real ξ se le asigna el correspondiente número en punto flotante $\texttt{fl}(\xi)$, utilizando redondeo al valor más cercano. Es decir, si $\xi_- \leqslant \xi < \xi_+$, donde $\xi_-, \xi_+ \in \mathbb{F}$ son dos números en punto flotante consecutivos, se define $\texttt{fl}(\xi) := \xi_-$ si $|\xi - \xi_-| \leqslant |\xi - \xi_+|$, o $\texttt{fl}(\xi) := \xi_+$ en caso contrario.

A continuación se presentan dos maneras de representar un número real en algún sistema $\mathbb{F}$.

Definición 2.5 (Truncamiento). Sea $x = \pm(d_1.d_2...d_t d_{t+1}...)_\beta \times \beta^e$ entonces podemos aproximar x por *truncamiento* hasta el dígito t como $\texttt{fl}_T(x) = \pm(d_1.d_2...d_t)_\beta \times \beta^e$.

Definición 2.6 (Redondeo)**.** Sea $x = \pm(d_1.d_2...d_t d_{t+1}...)_\beta \times \beta^e$ entonces podemos aproximar x por *redondeo* hasta el dígito t como

$$\mathtt{fl}_R(x) = \begin{cases} \pm(d_1.d_2...d_t))_\beta \times \beta^e & \text{si } d_{t+1} < \frac{\beta}{2}, \\ \pm(d_1.d_2...(d_t+1)_\beta \times \beta^e & \text{si } d_{t+1} \geq \frac{\beta}{2} \end{cases}$$

△

La operación de redondeo $\mathtt{fl} : \mathbb{R} \to \mathbb{F}$ mapea $\xi \in \mathbb{R}$ al número máquina más cercano $\mathtt{fl}(\xi) = \hat{\xi} \in \mathbb{F}$. Dicha operación posee las siguientes propiedades:

- Monotonía: $\xi \leq \eta \Rightarrow \mathtt{fl}(\xi) \leq \mathtt{fl}(\eta)$;
- Idempotencia: $\mathtt{fl}(\xi) = \xi$ para $\xi \in \mathbb{F}$.

Teorema 2.2 (Cota superior para el error relativo)**.** *Sea x real en el rango y $\mathtt{fl}(x)$ su representación punto flotante. Entonces se cumple que*

$$\frac{|x - \mathtt{fl}(x)|}{|x|} \leq \mu = \begin{cases} \beta^{1-t} & \textit{(truncamiento)} \\ \frac{1}{2}\beta^{1-t} & \textit{(redondeo)} \end{cases}$$

Demostración: Sin pérdida de generalidad, sea $x = d_1\beta^e + d_2\beta^{e-1} + ... + d_t\beta^{e+1-t} + d_{t+1}\beta^{e-t} + ... \in \mathbb{R}$. Para el caso del truncamiento, sea $\xi_T = \mathtt{fl}_T(x) = d_1\beta^e + d_2\beta^{e-1} + ... + d_t\beta^{e+1-t} \in \mathbb{F}(\beta, t, L, U)$. Entonces

$$\frac{|x - \xi_T|}{|x|} = \frac{|d_{t+1}\beta^{e-t} + ...|}{|d_1\beta^e + d_2\beta^{e-1} + ... + d_t\beta^{e-(t-1)} + d_{t+1}\beta^{e-t} + ...|}$$

$$\leq \frac{\beta \cdot \beta^{e-t}}{\beta^e} = \beta^{1-t}$$

En el caso del redondeo, sea $\xi_R = \mathtt{fl}_R(x) = d_1\beta^e + d_2\beta^{e-1} + ... + \hat{d}_t\beta^{e-(t-1)} \in \mathbb{F}(\beta, t, L, U)$ donde $\hat{d}_t$ está en función de d_{t+1}.

Se puede comprobar que $|x - \xi_R| \leq \frac{1}{2}\beta^{e-t+1}$, luego se tiene que

$$\frac{|x - \xi_R|}{|x|} \leq \frac{\frac{1}{2}\beta^{e-t+1}}{\beta^e} = \frac{1}{2}\beta^{1-t}$$

□

Observación 2.4. El Teorema 2.2 nos dice que redondear es mejor que truncar en cuanto a error relativo se refiere. ▲

Ejemplo 2.4. Considere el conjunto $\mathbb{F}_{10,3}$ (en donde el rango de exponentes ahorita no importa) en el cual se utiliza la técnica de redondeo. Escriba la versión flotante de los números $a = 1.23456$, $b = -0.1988$ y $c = 5062.2$ en el conjunto $\mathbb{F}_{10,3}$.

Solución: Debemos escribir la forma normalizada de los números dados, las cuales son

$$a = 1.23456 \times 10^0 \quad b = -1.988 \times 10^{-1}, \quad c = 5.0622 \times 10^3$$

El redondeo se aplica sobre la mantisa, y como la precisión es $t = 3$, se aplica la regla de "corte" al dígito $t + 1 = 4$, obteniéndose los siguientes números en el conjunto $\mathbb{F}_{10,3}$:

- $\text{fl}(a) = \text{fl}(1.23456 \times 10^0) = 1.23 \times 10^0$
- $\text{fl}(b) = \text{fl}(-1.9888 \times 10^{-1}) = -1.99 \times 10^{-1}$
- $\text{fl}(c) = \text{fl}(5.06222 \times 10^3) = 5.06 \times 10^3$

Definición 2.7. El número $\varepsilon_m \in \mathbb{R}^+$ más pequeño tal que

$$\texttt{fl}(1 + \varepsilon_m) > 1$$

se denomina *épsilon máquina* o *macheps*.

Observación 2.5. El épsilon máquina es la diferencia entre 1 y el número siguiente $x > 1$ con $x \in \mathbb{F}$ que se puede almacenar de forma exacta. Para el caso de `MATLAB` se puede calcular una aproximación para el épsilon máquina:

```
epsMach = 0.5;
while 1 + epsMach > 1
    epsMach = epsMach/2;
end

epsMach
```

lo cual nos da que `epsMach = 1.1102e-16`

Teorema 2.3. *Sea $x \in \mathbb{R}$ un número que puede ser representado de manera normal en un sistema de punto flotante con precisión t. Entonces, existe $\delta \in \mathbb{R}$ tal que*

$$fl(x) = x(1 + \delta)$$

donde $|\delta| < \varepsilon_m/2$

Teorema 2.4. *Sea $\mathbb{F}(\beta, t, L, U)$. Entonces $\varepsilon_m = \beta^{1-t}$ si la representación es por truncamiento y, $\varepsilon_m = \frac{1}{2}\beta^{1-t}$ si la representación es por redondeo.*

2.3. Operaciones aritméticas

La IEEE 754 define que las operaciones aritméticas y la operación raíz cuadrada son calculadas para números máquina por un redondeo correcto para el resultado exacto que se persigue. Si se denota la realización de una operación como $\star$ por $\hat{\star}$, entonces para $x, y \in \mathbb{F}$ se tiene que

$$x \hat{\star} y = \texttt{fl}(x \star y) \qquad [\star \in \{+, -, \cdot, \div, \sqrt{}\}]$$

Esto así, se tiene el modelo estándar para los números máquina y sus operaciones: para $x, y \in \mathbb{F}$, $\star \in \{+, -, \cdot, \div, \sqrt{}\}$ en donde $|\varepsilon| \leq \varepsilon_m$, tal que

$$x \hat{\star} y = (x \star y)(1 + \varepsilon)$$

para la realización-máquina $\hat{\star}$ de la operación $\star$.

La idea es evitar overflow o underflow. Así, las operaciones aritméticas en precisión finita se definen de la siguiente manera:

- **Suma:** $x \oplus y := \texttt{fl}(\texttt{fl}(x) + \texttt{fl}(y))$
- **Resta:** $x \ominus y := \texttt{fl}(\texttt{fl}(x) - \texttt{fl}(y))$
- **Producto:** $x \odot y := \texttt{fl}(\texttt{fl}(x) \cdot \texttt{fl}(y))$
- **División:** $x \oslash y := \texttt{fl}(\texttt{fl}(x) \div \texttt{fl}(y))$

La suma y el producto son conmutativas pero no asociativas. Tampoco se cumple la ley distributiva.

Ejemplo 2.5. Sabiendo que

$$a = 1.23456 \times 10^0 \quad b = -1.988 \times 10^{-1}, \quad c = 5.0622 \times 10^3$$

calcule los resultados siguientes, en el conjunto $\mathbb{F}_{10,3}$ con redondeo:

a) $a \oplus c$ b) $b \oslash c$

Compare sus resultados con los de la calculadora científica.

Solución:

Por el ejemplo 2.4 tenemos que $\text{fl}(a) = 1.23 \times 10^0$, $\text{fl}(b) = -1.99 \times 10^{-1}$, $\text{fl}(c) = 5.06 \times 10^3$. Entonces:

a) $a \oplus c = \text{fl}(\text{fl}(a) + \text{fl}(c)) = \text{fl}(1.23 \times 10^0 + 5.06 \times 10^3)$

$= \text{fl}(1.23 + 5060) = \text{fl}(5061.23) = \text{fl}(5.06123 \times 10^3) = 5.06 \times 10^3 = 5060$

b) $b \oslash c = \text{fl}(\text{fl}(-1.99 \times 10^{-1}) \div \text{fl}(5.06 \times 10^3))$

$= \text{fl}(-0.199 \div 5060) = \text{fl}(-3.932806 \times 10^{-5}) = -3.93 \times 10^{-5}$.

Con la calculadora los resultados son

$$\begin{aligned} a + c &= 5063.43456 \\ b \div c &= -3.9271463 \times 10^{-5} \end{aligned}$$

Note que los resultados difieren del resultado de la calculadora debido a la precisión y técnica escogida.

Ejercicio 2.3.1. Calcule los errores relativos al representar $a + c$ y $b \div c$ por $a \oplus c$ y $b \oslash c$ en el ejemplo 2.5. Además determine los dígitos de precisión en cada caso.

2.3.1. Pérdida de significancia

Hay que evitar en la medida de lo posible la cancelación, debido a que puede darse la *cancelación catastrófica*, en la cual, los dígitos más significativos se cancelan y el dígito menos significativo permanece. Esto ocurre cuando dos números muy similares de igual signo son restados entre sí (o dos con signos opuestos son sumados). Los cálculos recursivos acumulan y magnifican el error.

Considere $x_1 \oplus x_2 = \texttt{fl}(x_1) + \texttt{fl}(x_2) = x_1(1 + \delta_1) + x_2(1 + \delta_2) = x_1 + x_2 + (x_1\delta_1 + x_2\delta_2)$

$$= x_1 + x_2 + (\varepsilon_1 + \varepsilon_2)$$

Se nota cómo el error acumulado depende de los errores absolutos ε_1 y ε_2. En el peor de los casos podría ser que ambos errores absolutos tengan el mismo signo, i.e., el error absoluto en $x_1 \oplus x_2$ no es peor que $|\varepsilon_1| + |\varepsilon_2|$.

Usando el hecho de que $\texttt{fl}(-x_2) = -x_2 - \varepsilon_2$ se tiene que el error absoluto de $\texttt{fl}(x_1) - \texttt{fl}(x_2)$ no es peor que $|\varepsilon_1| + |\varepsilon_2|$.

Otro ejemplo: consideremos la ecuación cuadrática $ax^2 + bx + c = 0$ de la cual sabemos que

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Un problema surge si b^2 es mucho mayor que $|4ac|$ pues el numerador se "parece" a 0 y las raíces por tanto también "se parecen" a 0. Incluso si la diferencia de magnitudes no es muy grande, una

raíz se mantiene todavía muy pequeña. Sin pérdida de generalidad asumamos que $b > 0$ y la raíz menor está dada por

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

El problema puede ser tratado racionalizando:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

Por ejemplo, tomando $a = 1, b = 100000, c = 1$ y una precisión de 2×10^{-10}, la fórmula original da $x = -1.525878906 \times 10^{-5}$ para la raíz menor mientras que con la fórmula modificada da $x = -1.000000000 \times 10^{-5}$ la cual es la mejor que permite dicha precisión.

2.4. Ejercicios

1. Observe el video Representación de Información Digital - Cambio de base

2. Compruebe que $(\frac{11}{2})_{10} = (1.011)_2 \times 2^2$.

3. Observe el video Representación de Información Digital - Números reales en coma flotante IEEE 754

4. Sea $\xi = (257.0625)_{10}$. Transforme ξ a su representación binaria y luego a su representación en el sistema de precisión simple según el standard de la IEEE. Haga lo mismo pero representándo ξ en precisión doble.

5. Escriba explícitamente todos los elementos del conjunto $\mathbb{F}(2, 4, -2, 2)$. Luego dibújelos en la recta numérica. Calcule la cantidad total de elementos de este $\mathbb{F}$.

6. Calcule el error absoluto y relativo de las aproximaciones de ξ (busque representaciones con al menos 10 dígitos) mediante $\hat{\xi}$.

   a) $\xi = \pi, \quad \hat{\xi} = 3,1416$
   b) $\xi = \pi, \quad \hat{\xi} = \frac{22}{7}$
   c) $\xi = e, \quad \hat{\xi} = 2.8182$
   d) $\xi = e, \quad \hat{\xi} = \frac{65}{24}$

   ¿Cuántos dígitos significativos[3] hay en la representación del inciso b).

7. Considere $x = 0.134$, $y = 0.00345$. Normalice dichos números en su representación punto flotante en base $\beta = 10$, aplicando redondeo con precisión $t = 3$, y calcule $x \oplus y$. Compare con $x + y$ calculando el error relativo $\frac{|(x+y) - (x \oplus y)|}{|x+y|}$.

[3] ¿Qué significa ue una aproximación $\hat{\xi}$ tenga p dígitos significativos con respecto al valor exacto ξ?

8. Determine una fórmula para convertir números en formato de precisión simple a números en base 10. Use dicha fórmula para mostrar que

0	01111100	01000000000000000000000

$= 0.15625$

9. Sea $\mathbb{F}(\beta, t, L, U)$ conjunto de precisión finita con $|L|, |U| < \infty$.

   Demuestre que

$$x_{\min} = \beta^L, \quad x_{\max} = (1 - \beta^{-t}) \cdot \beta^{U+1}$$

   donde dichos valores son el mínimo y máximo del conjunto $\mathbb{F}$ que son positivos.

10. La función $f(x) = \sqrt{x^2 + 1} - 1$ presenta cancelación catastrófica para $x \approx 0$. Evalúela en $x = 0.01$ con aritmética exacta, y luego utilice redondeo con tres cifras ($t = 3$). ¿Cómo puede ajustarse la fórmula para evitar cancelación catastrófica?

11. Al resolver la ecuación cuadrática $ax^2 + bx + c = 0$ se pueden presentar serios problemas de cancelación catastrófica si una de las raíces es muy pequeña en comparación con la otra. A fin de evitar este problema en lugar de usar la fórmula general clásica se usan las fórmulas[4]

$$x_1 = \text{signo}\left(\frac{-b}{2a}\right)\left[\left|\frac{-b}{2a}\right| + \sqrt{\left(\frac{b}{2a}\right)^2 - \frac{c}{a}}\right]; \qquad x_2 = \frac{c/a}{x_1}$$

   Utilice los dos métodos para resolver la ecuación $x^2 - 10^5 x + 1 = 0$ usando truncamiento con 8 cifras.

12. Considere el sistema de ecuaciones lineales

$$\begin{cases} 1.2x + 2.4y &= 3.5 \\ 0.1x + 1.25y &= 0.01 \end{cases}$$

   Calcule la solución exacta usando la regla de Cramer. Luego haga lo mismo pero usando redondeo a 2 cifras. Determine el error relativo para cada componente del vector solución.

13. Considere el vector $\mathbf{x} = (1, 0.01, 0.01, ..., 0.01)^{\mathtt{t}} \in \mathbb{R}^{10^4+1}$. Calcule la norma euclídea en aritmética exacta. Luego calcule $\mathtt{fl}(\|\mathbf{x}\|_2)$ usando $\mathbb{F}(10, 2, -3, 3)$. ¿Cuántos dígitos significativos tiene la aproximación calculada de la norma?

14. Observe el video ¡Hay números subnormales! ¿Lo sabías?

[4] signo$(x) = 1$ si $x > 0$ (es positivo), signo$(0) = 0$, y signo$(x) = -1$ si $x < 0$ (es negativo).

Capítulo 3

Ecuaciones no lineales

Ecuaciones de la forma $f(x) = 0$ en la cual f es una función "suave" apropiada $f : \mathbb{R} \to \mathbb{R}$, se quiere encontrar una raíz de dicha función. Cualquier ecuación no lineal puede expresarse en dicha forma. Para ello estudiaremos algunos *métodos iterativos*: dado un valor inicial x_0 se construye una sucesión de números reales $\{x_n\}_{n=0}^{\infty} = \{x_0, x_1, x_2, ...\}$, y la idea es que si la ecuación y el método son adecuados se esperará que $\lim_{n\to+\infty} x_n = \xi$, es decir, que haya convergencia de la sucesión hacia un punto ξ.

3.1. El método de bisección

Dados $[a, b]$ y $f(a)f(b) < 0$ con f continua en dicho intervalo, por el teorema del valor intermedio (TVI) f debe tener al menos un cero. El *método de bisección* puede ser utilizado para encontrar dicho cero. Este método es *robusto*, esto es, se garantiza que converge hacia el cero y posiblemente lo haga a una velocidad de convergencia muy lenta. También es conocido como el *método de búsqueda binario*.

Calculamos el punto medio $m = \frac{a+b}{2}$ en cada paso, así como $f(m)$. Si $f(m) = 0$ se termina el proceso. Si no, se tienen dos casos: si $f(a)f(m) < 0$ se toma $m \leftarrow b$, de lo contrario si $f(m)f(b) < 0$ se toma $m \leftarrow a$. El algoritmo acaba cuando $b - a$ es suficientemente pequeño. En otras palabras, el método se detiene cuando $|b_{n+1} - a_{n+1}| < \texttt{tol}$, donde `tol` es la tolerancia especificada para el método de bisección. También puede usarse como criterio de parada $|f(m)| < \texttt{tol}$. En cualquiera de los casos es menester revisar el comportamiento de la función y elegir el criterio de parada más conveniente.

Teorema 3.1. *Suponiendo que f es continua en $[a, b]$ y $f(a)f(b) < 0$, el método de bisección genera una sucesión $\{x_n\}$ que aproxima ξ un cero de f, tal que*

$$|x_n - \xi| \leq \frac{b - a}{2^n}$$

Demostración: Sean las sucesiones $\{a_n\}, \{b_n\}$ y $\{x_n\}$ definidas de la siguiente manera, con $x_n := \frac{a_n+b_n}{2}$:

$$[a_n, b_n] := \begin{cases} [a_{n-1}, x_{n-1}], & \text{si } f(a_{n-1})f(x_{n-1}) < 0 \\ [x_{n-1}, b_{n-1}], & \text{si } f(x_{n-1})f(b_{n-1}) < 0 \end{cases}$$

Supondremos también que f no alcanza el cero en una cantidad finita de pasos en el intervalo $[a, b]$. Sea la sucesión de intervalos $I_0 := [a_0, b_0]$, $I_k := [a_k, b_k]$, entonces se tiene que $I_0 \supset I_1 \supset ... \supset I_k$ tales que longitud$(I_k) =$ longitud$(I_{k-1})/2$. Luego longitud$(I_n) = (b-a)/2^n \to 0$ cuando $n \to \infty$. Por el teorema de intervalos encajados de Cantor se tiene que existe ξ único tal que $\xi \in I_n$ para todo n. Por consiguiente,

$$|x_n - \xi| \leq b_n - a_n \leq \frac{b-a}{2^n}$$

□

Observación 3.1. En la iteración de bisección se tiene que $|e_k| = |x_k - \xi| \leq 2^{-k}(b-a)$, además si se considera $|e_k| <$ `tol` como condición de parada, entonces despejando k se obtiene que

$$\frac{b-a}{2^k} < \texttt{tol} \Rightarrow \frac{b-a}{\texttt{tol}} < 2^k \Rightarrow \log_2\left(\frac{b-a}{\texttt{tol}}\right) < k$$

así, si se desea utilizar una tolerancia tol, entonces el valor mínimo de `iterMax` para la iteración de bisección puede ser considerado de la forma $\texttt{iterMax} = \left[\!\left[\log_2\left(\frac{b-a}{\texttt{tol}}\right)\right]\!\right] + 1$

Algorithm 1 Método de bisección

Require: f continua, intervalo inicial $I_0 \leftarrow [a, b]$ con $f(a)f(b) < 0$
Ensure: Aproximación c_k de una raíz c de f en I_0

```
k ← 0;
while la iteración no haya convergido do
    c_k ← (a_k + b_k)/2;
    if c_k es una raíz then
        Retorne c_k;
    end if
    if f(a_k)f(c_k) < 0 then
        a_{k+1} ← a_k, b_{k+1} ← c_k;
    else
        a_{k+1} ← c_k, b_{k+1} ← b_k;
    end if
    k ← k + 1;
end while
```

A continuación un posible algoritmo implementado en `MATLAB`, para el cual hay que definir la función f en un archivo previo (digamos `f.m`, o también utilizando funciones anónimas):

```
function [c, iters] = biseccion(f, a, b, tol)
if f(a)*f(b) > 0
    disp('No se cumple el cambio de signo')
    return
end

iters = floor(log2((b-a)/tol)) + 1;

for k = 1:iters
    c = (a+b)/2;
    if f(c) == 0
        a = c;
        b = c;
    elseif f(a)*f(c) < 0
        b = c;
    else
        a = c;
    end

end

c = (a+b)/2;

end
```

Este algoritmo devuelve `c` la cual es una aproximación a la raíz e `iters` es la cantidad de iteraciones llevadas a cabo.

Ejemplo 3.1. Consideremos el problema físico de hallar la porción de una esfera de radio r que queda sumergida al meter la esfera en agua. Supongamos que la esfera está construida con una variedad de pino que tiene una densidad de $\rho = 0.638$ gr/cm^3 y que su radio mide $r = 10$ cm. Después de aplicar varios cálculos y el principio de Arquímedes, se obtiene la ecuación

$$\frac{\pi(2552 - 30h^2 + h^3)}{3} = 0$$

en donde h es la profundidad alcanzada por el polo sur de la esfera en el agua. Se sabe que una aproximación de las tres raíces del polinomio $p(h) = 2552 - 30h^2 + h^3$ son $y_1 = -8.17607212$, $y_2 =$

11.86150151 y $y_3 = 26.31457061$.

Explique por qué y_1 y y_3 no pueden ser soluciones al problema planteado. Además dé una función f y un intervalo adecuado para poder aproximar con el **método de bisección** la raíz y_2.

Solución: y_1 no puede darse porque la idea es sumergir la esfera, y con este dato la esfera estaría fuera del agua. y_3 no puede darse porque la esfera estaría completamente hundida, y la idea es que quede siempre parte de ella fuera del agua.

Tome $f(h) = p(h)$ en el intervalo $I_0 := [0, 20]$. Nótese que (ver tabla) $f(0) \cdot f(20) < 0$, con f continua en dicho intervalo.

.....

Completamos la tabla siguiente, donde $\boxed{h_k := \dfrac{a_k + b_k}{2}}$:

k	a	b	h	f(a)	f(b)	f(h)
0	0	20	10	2552	−1448	552
1	10	20	15	552	−1448	−823
2	10	15	12.5	552	−823	−182.38
3	10	12.5	11.25	552	−182.38	178.95
4	11.25	12.5	11.875	178.95	−182.38	−3.9082
5	11.25	11.875	11.563	178.95	−3.9082	87.065
6	11.563	11.875	11.719	87.065	−3.9082	41.452
7	11.719	11.875	11.797	41.452	−3.9082	18.739
8	11.797	11.875	11.836	18.739	−3.9082	7.4071
9	11.836	11.875	11.855	7.4071	−3.9082	1.7473
10	11.855	11.875	11.865	1.7473	−3.9082	−1.081
11	11.855	11.865	11.86	1.7473	−1.081	0.33304
12	11.86	11.865	11.863	0.33304	−1.081	−0.374

Entonces una aproximación es $h_{12} = 11.863$ .....

Ejemplo 3.2. La velocidad de descenso de un paracaidista está dada por $v = \dfrac{mg}{k}\left(1 - e^{\frac{-k}{m}t}\right)$, en donde k es el *coeficiente de arrastre*. Vamos a trabajar el caso $v = 40$ m/s, $m = 68.1$ kg, $t = 10$ s y $g = 9.8$ m/s^2.

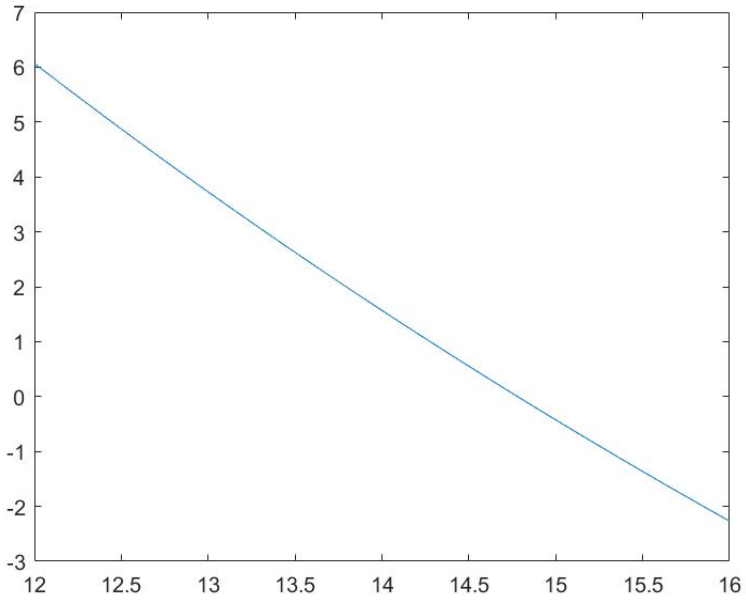
a) Determine una función adecuada $f(k)$ a partir de la fórmula anterior, tal que $f(k) = 0$. Grafique dicha función.

Solución: Una posible respuesta es la siguiente

$$f(k) = \frac{mg}{k}\left(1 - e^{\frac{-k}{m}t}\right) - v = \frac{667.38}{k}\left(1 - e^{-0.146843k}\right) - 40$$

Con `MATLAB` se puede crear una gráfica de manera simple, en el intervalo $[12, 16]$, usando

```
x = linspace(12, 16); y = (667.38*(1-exp(-0.146843*x)))./x -40; plot(x,y)
```



b) Postule un intervalo adecuado para utilizar el método de bisección y aproxime el coeficiente de arrastre a partir de $f(k) = 0$ con una tolerancia de $5 \times 10^{-3} = 0.5\,\%$.

Solución: Un intervalo posible es $I_0 = [12, 16]$ puesto que

$$f(12) = 6.067, \quad f(16) = -2.269$$

Utilizando `[T] = biseccionT2(f, 12, 16, 5e-3, 9)`

T =

1.0000	14.0000	16.0000	15.0000	−0.4248	2.0000
2.0000	14.0000	15.0000	14.5000	0.5523	1.0000
3.0000	14.5000	15.0000	14.7500	0.0590	0.5000
4.0000	14.7500	15.0000	14.8750	−0.1841	0.2500
5.0000	14.7500	14.8750	14.8125	−0.0629	0.1250
6.0000	14.7500	14.8125	14.7813	−0.0020	0.0625
7.0000	14.7500	14.7813	14.7656	0.0284	0.0313
8.0000	14.7656	14.7813	14.7734	0.0132	0.0156
9.0000	14.7734	14.7813	14.7773	0.0056	0.0078
10.0000	14.7773	14.7813	14.7793	0.0018	0.0039

De donde la aproximación buscada es `x = 14.7773`, teniéndose que

$$f(14.7773) = 0.0056$$

◇

3.2. Método de falsa posición

El método de bisección siempre escoge el punto medio de cada intervalo en cuestión. Ahora vamos a considerar una modificación del mismo. Se puede considerar el segmento de recta entre $(a, f(a))$ y $(b, f(b))$, el cual está dado por

$$\frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b)$$

De esto se tiene que $m = \dfrac{f(b)a - f(a)b}{f(b) - f(a)}$, se hace la prueba de signos y en cada paso el intervalo que contiene la raíz es actualizado. En el cálculo de m no hay peligro de pérdida de significancia debido a que $f(a)$ y $f(b)$ tienen signos opuestos.

Este método es llamado *regula falsi* o *regla de falsa posición* pues se toma una "falsa" raíz –un sustituto pues. En primera instancia este método parece superior en cuanto a convergencia[1] al de bisección al tratar de aproximar un cero de la ecuación. Asintóticamente uno de los puntos extremos de cada intervalo deberá converger a la raíz, mientras que el otro permanecerá siempre fijo. Así que solo uno de los extremos es actualizado en cada paso del método. El valor de m es un promedio ponderado de los valores de la función. El método puede ser modificado [2] para ajustar los pesos de los valores de la función en el caso en el cual el mismo punto final es repetido en un paso:

$$m = \frac{\frac{1}{2}f(b)a - f(a)b}{\frac{1}{2}f(b) - f(a)} \qquad \text{o} \qquad m = \frac{f(b)a - \frac{1}{2}f(a)b}{f(b) - \frac{1}{2}f(a)}$$

Estos ajustes garantizan convergencia superlineal.

Concretamente, el método de falsa posición es una variante del método de la secante combinado con el de bisección, donde en lugar de calcular la pendiente con los puntos $(x_k, f(x_k))$ y $(x_{k-1}, f(x_{k-1}))$ se eligen los puntos $(x_k, f(x_k))$ y $(x_\tau, f(x_\tau))$ siendo $\tau = 0, 1, ..., k-1$ el índice menor que cumple $f(x_k) \cdot f(x_\tau) < 0$. El error relativo es $|e_k| = \left|\dfrac{x_k - x_{k-1}}{x_k}\right|$.

[1] Aunque no siempre es el caso.

[2] Esta variante se conoce como *algoritmo de Illionois*.

```
function [x, k] = RegulaFalsi(f, x0, x1, tol, iterMax)
k = 1;
err = tol + 1;

while k < iterMax && err >= tol
    q = f(x1)-f(x0);
    if q == 0
        disp('Pendiente indefinida')
        return
    end
    t = x1;
    x1 = x1 - (x1-x0)/q * f(x1);
    err = abs(x1 - t)/abs(x1);
    if f(x1)*f(t) < 0
        x0 = t;
    end
    k = k+1;
end
x = x1;
end
```

3.3. Método del punto fijo

En este apartado se estudia una función φ en un intervalo $[a, b]$ que satisface ciertas condiciones, entre ellas la contractividad, para dar paso a una iteración que utiliza la misma función dada partiendo de un valor inicial x_0.

Definición 3.1 (Punto fijo)**.** Dada φ una función definida en $D \subseteq \mathbb{R}$. Se dice que $\xi \in D$ es punto fijo de φ si y solo si $\varphi(\xi) = \xi$.

Teorema 3.2 (Brower)**.** *Supongamos que φ es una función definida y continua sobre $[a, b]$ y además $\varphi([a, b]) \subseteq [a, b]$. Entonces existe $\xi \in [a, b]$ tal que $\xi = \varphi(\xi)$.*

Demostración: Si $\varphi(a) = a$ o $\varphi(b) = b$ el teorema se cumple. Supongamos que $\varphi([a, b]) \subset [a, b]$, entonces $a < \varphi(a)$ y $\varphi(b) < b$. Consideremos la función $g(x) := \varphi(x) - x$, la cual cumple que $g(a) \cdot g(b) < 0$, y por el TVI existe $\xi \in [a, b]$ tal que $g(\xi) = 0$, es decir, $\varphi(\xi) = \xi$. □

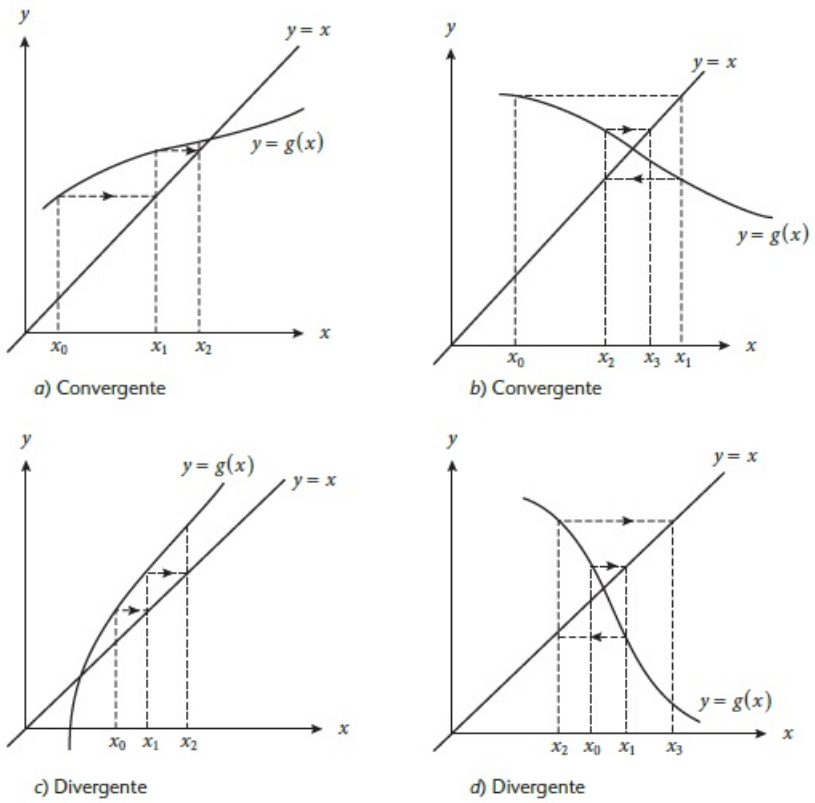


Figura 3.1: Iteración del punto fijo. Tomado de GUTIÉRREZ et alii.

Definición 3.2 (Iteración de punto fijo)**.** Sea φ definida y continua sobre $[a,b]$ y que además $\varphi([a,b]) \subseteq [a,b]$. Dado un valor inicial $x_0 \in [a,b]$ se define la sucesión

$$x_{k+1} := \varphi(x_k), \quad \text{para } k = 0, 1, 2, ...$$

la cual es llamada *iteración simple* o *método de aproximaciones sucesivas.*

Definición 3.3. Sea φ definida y continua sobre $[a,b]$. Se dice que φ es una **contracción** sobre $[a,b]$ si existe L positiva tal que $0 < L < 1$ y además

$$|\varphi(x) - \varphi(y)| \leq L|x - y|, \quad \forall x, y \in [a,b]$$

Teorema 3.3. *Suponga que φ es una función real, definida y diferenciable sobre $[a,b]$. Si $|\varphi'(x)| \leq L$ para todo $x \in [a,b]$ con $0 < L < 1$ entonces φ es una contracción sobre $[a,b]$.*

Demostración: Sean $x, t \in [a,b]$. Por el Teorema del Valor Medio $\exists c \in]a,b[$ tal que $\frac{\varphi(x)-\varphi(t)}{x-t} = \varphi'(c)$. Esto implica que

$$\left|\frac{\varphi(x) - \varphi(t)}{x - t}\right| = |\varphi'(c)| \leq L$$

$$\Rightarrow |\varphi(x) - \varphi(t)| \leq L|x - t|$$

Como $L \in]0,1[$ se concluye que φ es contractiva en $[a,b]$. □

El teorema que sigue es muy importante puesto que nos da las condiciones para determinar si una sucesión de iteración simple converge a un único punto en un subconjunto real.

Teorema 3.4. *Sea φ definida y continua sobre $[a,b]$ y que además $\varphi([a,b]) \subseteq [a,b]$ y siendo φ una contracción sobre $[a,b]$. Entonces φ tiene un único punto fijo $\xi \in [a,b]$. Además la iteración simple $x_{k+1} = \varphi(x_k)$ converge a ξ para cualquier valor $x_0 \in [a,b]$.*

Demostración: Sea $x_0 \in [a,b]$ arbitrario. Se define $x_{k+1} := \varphi(x_k)$. Entonces, como φ es contractiva,

$$|x_{k+1} - x_k| = |\varphi(x_k) - \varphi(x_{k-1})| \leq L|x_k - x_{k-1}|$$

Aplicando sucesivamente la contractividad se tiene que

$$|x_{k+1} - x_k| \leq L^k |x_1 - x_0|$$

Dados $m, n \in \mathbb{N}$ con $m > n$, utilizando desigualdad triangular y el hecho previo se tiene que

$$\begin{aligned} |x_m - x_n| &\leq |x_m - x_{m-1}| + |x_{m-1} - x_{m-2}| + ... + |x_{n+1} - x_n| \\ &\leq (L^{m-1} + L^{m-2} + ... + L^n)|x_1 - x_0| \\ &= |x_1 - x_0| \sum_{i=n}^{m-1} L^i = L^n |x_1 - x_0| \left(\frac{1 - L^{m-n-1}}{1 - L} \right) \\ &< L^n |x_1 - x_0| \cdot \frac{1}{1-L} = \frac{L^n}{1-L} |x_1 - x_0| \end{aligned}$$

porque $L \in]0,1[$ implica que $\lim_{n \to +\infty} L^{m-n-1} = 0$. Por la misma razón $\lim_{n \to +\infty} \frac{L^n}{1-L} = 0$ y por lo tanto $\{x_k\}_{k \in \mathbb{N}}$ es una sucesión de Cauchy, implicando que existe $\xi \in [a,b]$ tal que $x_n \to \xi$. Como φ es continua en $[a,b]$ se tiene que

$$\xi = \lim_{k \to +\infty} x_{k+1} = \lim_{k \to +\infty} \varphi(x_k) = \varphi(\lim_{k \to +\infty} x_k) = \varphi(\xi)$$

lo cual demuestra que ξ es un punto fijo de φ en $[a,b]$. Para mostrar la unicidad, sean ξ y η dos puntos fijos, luego

$$|\xi - \eta| = |\varphi(\xi) - \varphi(\eta)| \leq L|\xi - \eta|$$

$$\Rightarrow (1 - L)|\xi - \eta| \leq 0$$

lo cual se cumple solo si $\xi = \eta$. □

A continuación un posible algoritmo en `MATLAB`, para el cual hay que definir la función f en un archivo previo (digamos `f.m`):

Algorithm 2 Método de punto fijo

Require: f continua sobre $[a, b]$, valor inicial $c_0 \in [a, b]$
Ensure: c_k aproximación al punto fijo de f en $[a, b]$
  for $k \leftarrow 1, 2, \ldots$ **do**
    $c_k \leftarrow f(c_{k-1})$;
  end for

```
function [x, k] = puntofijo(f, x0, tol, iterMax)
    k = 0;
    x = x0;
    err = tol+1;
    while k < iterMax && err > tol
        x = f(x);
        err = abs(x-x0)/abs(x);
        x0 = x;
        k = k+1;
    end
end
```

Este algoritmo devuelve x la cual es una aproximación a la raíz y k es la cantidad de iteraciones llevadas a cabo.

Para efectos de los ejercicios tomaremos el error relativo, que es el que compararemos con la tolerancia dada: $|e_k| = \dfrac{|x_{k+1} - x_k|}{|x_{k+1}|}$

Una cota teórica para el número máximo de iteraciones necesarias que se requieren para que el método del punto fijo alcance una aproximación con exactitud de `tol` > 0 viene dada por

$$\texttt{iterMax} = \left[\!\!\left[\frac{\ln(\texttt{tol}(1-L)) - \ln|x_1 - x_0|}{\ln(L)} \right]\!\!\right] + 1$$

Quepa agregar que esta cota no es óptima.

Ejemplo 3.3. Sea $f(x) = x^3 - 2x - 5$. Muestre que la función $g(x) = \sqrt[3]{2x+5}$, en el intervalo $[2, 3]$ satisface las condiciones del teorema de unicidad del punto fijo.

Solución: En el caso de g, se obtiene de $x^3 = 2x + 5 \Rightarrow x = \sqrt[3]{2x+5}$.

- $g'(x) = \dfrac{2}{3(2x+5)^{2/3}}, \quad g''(x) = \dfrac{-8}{9(2x+5)^{5/3}}.$

  La ecuación $g'(x) = 0$ no tienen solución, así que los valores extremos de g se alcanzan en $[2, 3]$, luego, $g(2) = \sqrt[3]{9} \approx 2.0800$, $g(3) = \sqrt[3]{11} \approx 2.2239$. Esto muestra que $g([2, 3]) \subseteq [2, 3]$.

- Ahora, como $g''(x) = 0$ no tiene solución, los valores extremos de $|g'(x)|$ en $[2, 3]$ son: $|g'(2)| = \frac{2}{3\cdot 9^{2/3}} \approx 0.1540$, $|g'(3)| = \frac{2}{3\cdot 11^{2/3}} \approx 0.1347$, entonces tomando $L = |g'(2)| < 1$ se comprueba que g es contractiva en $[2, 3]$.

◇

Ejemplo 3.4. La función $f(x) = x^3 - x^2 - 4x + 5$ tiene exactamente 3 puntos fijos.

a) Calcule de manera exacta puntos fijos de f.

Solución: Se tiene que $x = f(x)$ es la ecuación a resolver. Es decir,

$$x = x^3 - x^2 - 4x + 5 \Rightarrow x^3 - x^2 - 5x + 5 = 0$$

Agrupando,

$$x^3 - x^2 - 5x + 5 = (x - 1)(x^2 - 5) = (x - 1)(x - \sqrt{5})(x + \sqrt{5})$$

Así los puntos fijos de f son $x = 1$, $x = \pm\sqrt{5}$.

b) Considere el intervalo $I = [\frac{9}{10}, \frac{11}{10}]$. Justifique si la sucesión por iteración de punto fijo converge o no a x_0 cercano a $x = 1$.

Solución: Tenemos que $f'(x) = 3x^2 - 2x - 4$ y $f''(x) = 6x - 2$ y esta última función es positiva para todo $x \in I$, así que f' es creciente y significa que los extremos relativos los alcanza en la frontera de I. Luego,

$$|f'(\tfrac{9}{10})| = \frac{337}{100} = 3.37, \qquad |f'(\tfrac{11}{10})| = \frac{257}{100} = 2.57$$

Entonces $\max\limits_{x\in I}\{|f'(x)|\} = 3.37 \geq 1$ y esto implica que f no es contractiva en I. Esencialmente, la no contractividad lleva a que al tomar cualquier $x_0 \in I - \{1\}$ implique que la sucesión $x_{n+1} = f(x_n)$ no converja a $x = 1$ que es precisamente el punto fijo que vive en I. Ahora, la sucesión converge claramente en un paso cuando $x_0 = 1$ pues $f(1) = 1$.

◇

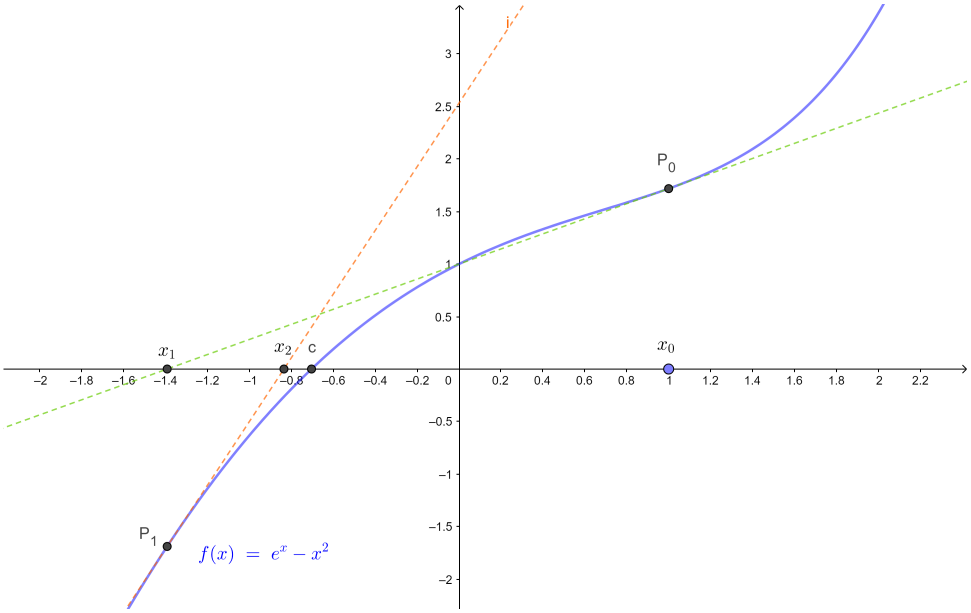
3.4. Método de Newton–Raphson

Supongamos que f es derivable en cierto conjunto en donde las operaciones a realizar estén bien definidas. Tomemos un punto $(a, f(a))$ y la pendiente de la recta que contiene a este punto es $f'(a)$. La ecuación de la recta tangente es $\ell : y - f(a) = (x - a)f'(a)$. El punto de intersección de ℓ con el eje X está dado al resolver la ecuación $0 - f(a) = (x - a)f'(a) \Rightarrow x = a - \frac{f(a)}{f'(a)}$.

Si definimos la sucesión

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Esta fórmula es conocida como *método de Newton* o *método de Newton-Raphson*



El método falla si cualquier punto de iteración es estacionario, i.e., un punto en donde la primer derivada se anula.

Teorema 3.5. *Si $f \in C^2([a,b])$ y además se satisface que*

1) $f(a) \cdot f(b) < 0$;

2) $f'(x) \neq 0$ *para todo* $x \in [a,b]$;

3) $f''(x) \geq 0$ *o* $f''(x) \leq 0$, *para todo* $x \in [a,b]$;

4) $\left|\dfrac{f(a)}{f'(a)}\right| \leq b-a$ *y* $\left|\dfrac{f(b)}{f'(b)}\right| \leq b-a$.

Entonces el método de Newton convergerá a la única solución ξ *de* $f(x) = 0$ *para cualquier* $x_0 \in [a,b]$.

A continuación un posible algoritmo en `MATLAB`, para el cual hay que definir la función f en un archivo previo (digamos `f.m`) y su derivada (digamos `df.m`):

Algorithm 3 Método de Newton-Rhapson

Require: f continua y derivable, valor inicial c_0 (cercano a la raíz c de f)
Ensure: Aproximación c_k de la raíz c de f
  for $k \leftarrow 1, 2, \ldots$ **do**
    $c_k \leftarrow c_{k-1} - \dfrac{f(c_{k-1})}{f'(c_{k-1})}$;
  end for

```
function [x, k] = NewRaph(f, df, x0, tol, iterMax )
k = 0;
x = x0;
err = tol+1;
    while k < iterMax && err >= tol
        q = df(x);
            if q==0
                disp('Se anula la derivada')
            end
        x = x-f(x)/q;
        err = abs(x0-x)/abs(x);
        x0 = x;
        k = k+1;
    end
end
```

Este algoritmo devuelve x la cual es una aproximación a la raíz y k es la cantidad de iteraciones llevadas a cabo.

Con respecto a la condición de parada del método de Newton-Raphson, se puede considerar el error absoluto o el relativo de una aproximación, es decir $|e_k| = |x_k - x_{k-1}|$ o $|e_k| = \left|\dfrac{x_k - x_{k-1}}{x_k}\right|$. Sin embargo, como se ve en el código de `MATLAB` previo se elige el error relativo.

Ejemplo 3.5. Estudios médicos han establecido que una persona que salta en bungee puede sufrir una lesión significativa de vértebra si la velocidad de caída libre excede los 36 m/s después de 4 s de caída libre. Los gerentes de una compañía de salto en bungee quieren determinar la masa crítica en bajo este criterio médico, dado el coeficiente de resistencia $c_d = 0.25$ kg/m.

Se sabe que la velocidad de la persona que salta en bungee está dada por

$$v(t) = \sqrt{\frac{gm}{c_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m}}t\right)$$

donde $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Con el método de bisección en el intervalo [140, 150] para las masas se obtuvo la aproximación $m \approx 142.74$ kg, después de 21 iteraciones.

a) Plantee la función $f(m)$ tal que el problema a resolver sea $f(m) = 0$ con la masa m como variable independiente, y además se tenga que (esto debe comprobarlo también derivando)

$$\frac{df}{dm} = \frac{1}{2}\sqrt{\frac{g}{mc_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m}}\, t\right) - \frac{g}{2m} t \, \operatorname{sech}^2\left(\sqrt{\frac{gc_d}{m}}\, t\right)$$

donde $\operatorname{sech}(x) := \dfrac{2}{e^x + e^{-x}}$.

Solución: La función es

$$f(m) = \sqrt{\frac{gm}{c_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t)$$

y al derivarla con respecto a m se obtiene

$$\frac{df}{dm} = \frac{1}{2}\sqrt{\frac{g}{mc_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m}}\, t\right) - \frac{g}{2m} t \, \operatorname{sech}^2\left(\sqrt{\frac{gc_d}{m}}\, t\right)$$

b) Considere la sucesión de Newton-Rhapson

$$m_{n+1} := m_n - \frac{f(m_n)}{f'(m_n)}$$

Utilice el inciso anterior para escribir explícitamente m_{n+1} y calcule m_1 sabiendo que $m_0 := 140$ [kg]. Use $g = 9.81$ [m/s^2]. Luego calcule el error relativo de m_1 con respecto a la aproximación dada por el método de bisección.

Solución: La sucesión es

$$m_{n+1} = m_n - \frac{\sqrt{\frac{gm_n}{c_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m_n}} t\right) - v(t)}{\frac{1}{2}\sqrt{\frac{g}{m_n c_d}} \cdot \tanh\left(\sqrt{\frac{gc_d}{m_n}}\, t\right) - \frac{g}{2m_n} t \, \operatorname{sech}^2\left(\sqrt{\frac{gc_d}{m_n}}\, t\right)}$$

Colocando los datos y calculando m_1 se tiene:

$$m_1 = 140 - \frac{\sqrt{\frac{9.81\times 140}{0.25}} \cdot \tanh\left(\sqrt{\frac{9.81 \times 0.25}{140}} \times 4\right) - 36}{\frac{1}{2}\sqrt{\frac{9.81}{140\times 0.25}} \cdot \tanh\left(\sqrt{\frac{9.81\times 0.25}{140}} \times 4\right) - \frac{9.81}{2\times 140} \times 4 \, \operatorname{sech}^2\left(\sqrt{\frac{9.81\times 0.25}{140}} \times 4\right)} \approx 142.6903 \text{ [kg]}$$

El error relativo está dado por

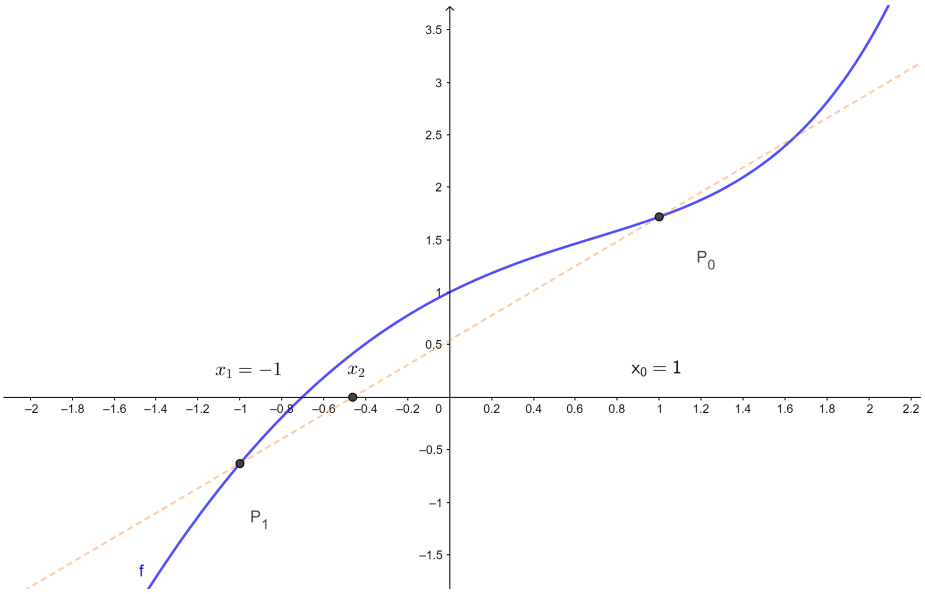
$$\varepsilon_r = \frac{|142.74 - 142.6903|}{|142.74|} \approx 3.48 \times 10^{-4}$$

◇

3.5. Método de la secante

El problema que podría presentarse en el método de Newton es que la función en cuestión no sea derivable o no se conozca explícitamente la derivada. Una alternativa es utilizar rectas secantes: tomamos al inicio $x_0 = a$, $x_1 = b$, luego

$$\begin{aligned} x_{n+1} &= \frac{f(x_n)x_{n-1} - f(x_{n-1})x_n}{f(x_n) - f(x_{n-1})} \\ &= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \end{aligned}$$



El resultado siguiente nos da garantía de la convergencia del método de la secante cuando se satisfacen las condiciones pedidas.

A continuación un posible algoritmo en `MATLAB`, para el cual hay que definir la función f en un archivo previo (digamos `f.m`) o como función anónima:

Algorithm 4 Método de la Secante

Require: f continua, valores iniciales c_0, c_1 (cercanos a la raíz)
Ensure: Aproximación c_k de la raíz c de f
for $k \leftarrow 1, 2, \ldots$ **do**
$$c_{k+1} \leftarrow c_k - \frac{c_k - c_{k-1}}{f(c_k) - f(c_{k-1})} f(c_k);$$
end for

```
function [x, k] = secante(f, x0, x1, tol, iterMax )
    k = 0;
    err = tol+1;
    while k < iterMax && err >= tol
        q = f(x1) - f(x0);
        if q==0
            disp('Pendiente indefinida')
        end
         t = x1;
         x1 = x1-f(x1)*(x1-x0)/q;
         err = abs(x0-x1)/abs(x1);
         x0 = t;
         k = k+1;
    end
    x = x1;
end
```

Este algoritmo devuelve x la cual es una aproximación a la raíz y k es la cantidad de iteraciones llevadas a cabo.

El error relativo en este método es $|e_k| = \left| \frac{x_k - x_{k-1}}{x_k} \right|$ y es el que se acota si se da una tolerancia específica.

Ejemplo 3.6. Considere la función $g(x) = x^5 + x^4 - 3$

a) Muestre que la iteración por el método de Newton–Raphson converge para cualquier $x_0 \in [1, 2]$. También calcule x_2 comenzando con $x_0 = 1$.

Solución: tenemos que $f \in C^2([1, 2])$ pues es un polinomio y además se satisface que

a) $f(a) \cdot f(b) < 0$: $f(1) = -1, \quad f(2) = 45$

b) $f'(x) \neq 0$ para todo $x \in [a, b]$: $f'(x) = 5x^4 + 4x^3$, claramente $f'(x) > 0$ para todo $x \in [1, 2]$

c) $f''(x) \geq 0$ o $f''(x) \leq 0$, para todo $x \in [a,b]$: $f''(x) = 20x^3 + 12x^2$, claramente $f''(x) \geq 0$ para todo $x \in [1,2]$

d) $\left|\dfrac{f(a)}{f'(a)}\right| \leq b-a$ y $\left|\dfrac{f(b)}{f'(b)}\right| \leq b-a$:

$$\left|\frac{f(1)}{f'(1)}\right| = \frac{1}{9} \leq 2-1 = 1, \qquad \left|\frac{f(2)}{f'(2)}\right| = \frac{45}{112} \leq 1$$

Entonces el método de Newton converge a la única solución ξ de $f(x) = 0$ para cualquier $x_0 \in [1,2]$.

Ahora, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$. Como $x_0 = 1$,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{-1}{9} = \frac{10}{9} = 1.\overline{1}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \approx 1.09345$$

b) Calcule x_4 con el método de la secante, tomando $x_0 = 1$, $x_1 = 2$.

Solución: La fórmula es la siguiente:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)$$

$$= x_n - \frac{x_n - x_{n-1}}{x_n^5 + x_n^4 - x_{n-1}^5 - x_{n-1}^4} \cdot (x_n^5 + x_n^4 - 3)$$

Luego,

$$x_2 = 2 - \frac{2-1}{2^5 + 2^4 - 1 - 1} \cdot (2^5 + 2^4 - 3) = \frac{47}{46} \approx 1.02173913,$$

$$x_3 = \frac{47}{46} - \frac{\frac{47}{46} - 2}{(\frac{47}{46})^5 + (\frac{47}{46})^4 - 2^5 - 2^4} \cdot \left((\tfrac{47}{46})^5 + (\tfrac{47}{46})^4 - 3\right) \approx 1.038756163,$$

Finalmente,

$$x_4 \approx 1.101338884$$

◇

3.5.1. Orden de convergencia

Definición 3.4. Considere una sucesión $\{x_k\}_{k=0}^{\infty}$ tal que $\lim_{k\to\infty} x_k = \xi$, con $x_k \neq \xi$ para todo k. Suponga que

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|} = \mu \geq 0$$

Entonces se dice que x_k converge a ξ

- sublinealmente, si $\mu = 1$;
- superlinealmente, si $\mu = 0$;
- linealmente, si $\mu \in]0, 1[$, y $\rho := -\log \mu$ se define como la razón de convergencia asintótica.

En el caso de convergencia superlineal, se dice que x_k converge a ξ con orden $q > 1$ si

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^q}$$

existe.

Orden de convergencia bisección

Para este caso el error $e_k := |x_k - \xi|$ no decrece de forma monótona en general, pero la convergencia es lineal porque

$$\frac{e_{k+1}}{e_k} \sim \frac{1}{2}$$

Orden de convergencia punto fijo

Si f es derivable y satisface las hipótesis adecuadas para convergencia de punto fijo, el método converge al menos linealmente, pues

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|} = \lim_{k\to\infty} \frac{|f(x_k) - f(\xi)|}{|x_k - \xi|} = |f'(\xi)| < 1$$

Si $f'(\xi) = 0$ entonces la convergencia es superlineal.

Orden de convergencia secante

Suponiendo que f tiene segunda derivada continua, sea ξ tal que $f(\xi) = 0$ y $f'(\xi) \neq 0$. Si x_0 es lo suficientemente cercano a ξ, entonces la sucesión $\{x_n\}_{n=0}^{\infty}$ generada por el método de la secante converge a ξ con un orden de convergencia $q = \dfrac{1+\sqrt{5}}{2} \approx 1.618$.

Orden de convergencia Newton

La convergencia del método de Newton es *cuadrática*.

3.6. Ejercicios

3.6.1. Método de bisección

1. Justifique por qué $2x\cos(2x) - (x-2)^2 = 0$ posee al menos un cero en [2, 3] y otro en [3, 4].
2. Encuentre la tercera iteración por el método de bisección de la función

$$f(x) = \text{sen}(x) - \cos(1 + x^2) - 1$$

en el intervalo [1.8, 2]. [No olvide que se trabaja en radianes]

3. Considere la ecuación $x = 2^{-x}$. Proporcione un intervalo que contenga la solución de dicha ecuación. Además determine el número de iteraciones necesarias para alcanzar una tolerancia de 10^{-5}.
4. Aplique la iteración de bisección a la ecuación $\text{sen}\, x = \frac{3}{4}$ en el intervalo [0.8, 0.9] y trate de determinar el cero actual con una tolerancia de 10^{-3}. [No olvide que se trabaja en radianes]
5. Encuentre una aproximación de $\sqrt{3}$ con un error menor a 10^{-4}, utilizando el método de bisección. [Sugerencia: utilice un polinomio adecuado]

3.6.2. Método del punto fijo

1. Utilice la iteración de punto fijo para determinar una aproximación de una solución de $x^4 - 3x^2 = 3$ en el intervalo [1, 2], utilizando $x_0 = 1$ y una tolerancia de 10^{-2}.
2. Considere la ecuación $8e^{\frac{-x}{5}} - 2x = 5$. Compruebe que $\varphi(x) = 4e^{\frac{-x}{5}} - \frac{5}{2}$ satisface las condiciones del teorema del punto fijo en $[0, 2]$. Use $x_0 = 1$ y obtenga una aproximación de la solución de la ecuación con una tolerancia de 10^{-2}.
3. Considere la ecuación $x^2 + 10\cos x = 0$ la cual posee dos raíces aproximadas que son 3.1619 y 1.9688. Determine la o las funciones auxiliares $\varphi(x)$ y un intervalo adecuado para que pueda utilizar las hipótesis del teorema del punto fijo de Brower. Con lo anterior, ¿cuántas iteraciones se necesitan para obtener una aproximación con una tolerancia de 10^{-4}?
4. Considere la ecuación $2x - \ln(1 + e^x) = 0$. Usando que $x = \ln(1 + e^x) - x = \varphi(x)$ muestre que φ satisface las hipótesis del teorema del punto fijo en el intervalo [0, 2]. Usando $x_0 = 1$ obtenga una aproximación de la solución de la ecuación con dos dígitos significativos.
5. Sea $f(x) = x^3 - 2x - 5$. Muestre que la función $g(x) = \sqrt[3]{2x + 5}$, en el intervalo [2, 3] satisface las condiciones del teorema de unicidad del punto fijo. Además calcule 5 iteraciones de la sucesión $x_{k+1} := g(x_k)$ con $x_0 = 3$.

3.6.3. Método de Newton–Raphson

1. Use el método de Newton para calcular una aproximación de la solución de $x^6 = x + 1$ en [1, 2] con un error relativo de 10^{-4}

2. Realice 3 iteraciones de Newton para resolver la ecuación $x^2 = 2$ y verifique que

$$\sqrt{2} \approx 1 + \frac{1}{2} - \frac{1}{12} - \frac{1}{108}$$

3. Aplique el método de Newton para calcular el cero de $f(x) = x^3 - 3x^2 2^{-x} + 3x4^{-x} - 8^{-x}$ en [0, 1].

4. Use el método de Newton para aproximar una solución de $0 = \frac{1}{2} + \frac{x^2}{4} - x \operatorname{sen} x - \frac{\cos 2x}{2}$ con la aproximación inicial $x_0 = \frac{\pi}{2}$ y con un error relativo menor a 10^{-5}.

5. Muestre que $f(x) = x^2 - x - 2$ tiene una raíz única en $[1, 3]$ a la cual converge la sucesión del método de Newton para todo $x_0 \in [1, 3]$.

3.6.4. Método de la secante

1. Aplique el método de la secante iniciando con $x_0 = 1$, $x_1 = 2$, $f(x_0) = 2$ y $f(x_1) = 1.5$. ¿Cuál es el valor de x_2?

2. Realice 4 iteraciones utilizando el método de la secante a $f(x) = x - 2\cos x$ con $x_0 = 1$, $x_1 = 1.5$ como condiciones iniciales.

3. Utilice el método de la secante para aproximar un cero de $f(x) = e^x - x^2$ utilizando 8 iteraciones y considerando que $x_0 = 1$ y $x_1 = 3$.

4. Considere la ecuación $\log(2) + \log(11 - x^2) = 2\log(5 - x)$. Use el método de la secante para aproximar las dos soluciones usando las aproximaciones iniciales $(x_0, x_1) = (0, 1)$ y $(x_0, x_1) = (2.5, 3.5)$. Calcule la solución exacta de la ecuación y determine el error absoluto y el error relativo de cada aproximación.

3.6.5. Adicionales

1. Aproxime la solución en $[-1, 0]$ de la ecuación $(x^3 + 0.1) \operatorname{sen}\left(\frac{3}{x^2+0.06}\right) = x$. Justifique la elección del método usado, indicando las ventajas que tiene sobre otros métodos para esta ecuación particular.

2. Un proyectil es lanzado con una velocidad inicial v_0 y con una inclinación α en un túnel de altura h, obtiene su máximo alcance cuando α es tal que $\text{sen}(\alpha) = \sqrt{\frac{2gh}{v_0^2}}$, donde $g \approx 9.8\ \text{m/s}^2$ corresponde a la aceleración gravitatoria terrestre. Calcule el valor de α usando el método de Newton, cuando $v_0 = 10\,\text{m/s}$ y $h = 1\,\text{m}$.

3. La velocidad de descenso de un paracaidista está dada por $v = \dfrac{mg}{k}\left(1 - e^{\frac{-kt}{m}}\right)$, en donde k es el *coeficiente de arrastre*. Vamos a trabajar el caso $v = 40$ m/s, $m = 68.1$ kg, $t = 10$ s y $g = 9.8\ \text{m/s}^2$.

   a) Determine una función adecuada $f(k)$ a partir de la fórmula anterior, tal que $f(k) = 0$. Grafique dicha función.

   b) Postule un intervalo adecuado para utilizar el método de bisección y aproxime el coeficiente de arrastre a partir de $f(k) = 0$ con una tolerancia de $5 \times 10^{-3} = 0.5\,\%$.

4. La función $f(x) = x^3 - x^2 - 4x + 5$ tiene exactamente 3 puntos fijos. Uno de ellos es $x = 1$.

   a) Calcule de manera exacta los otros dos puntos fijos.

   b) Considere el intervalo $I = [\frac{9}{10}, \frac{11}{10}]$. Justifique si la sucesión por iteración de punto fijo converge o no a x_0 cercano a $x = 1$.

5. Considere la función $f(x) = -1 - 2x + \displaystyle\int_0^x e^{t^2}\,dt$. Vamos a calcular una aproximación de uno de sus puntos críticos, que de hecho es un mínimo local. (Nota: Use truncamiento a 6 dígitos después del punto decimal: si tiene $N.1234567...$ con N entero, entonces lo escribe como $N.123456$)

   a) Muestre que la ecuación $f'(x) = 0$ posee una raíz en el intervalo $I = [0, 1]$.

   b) Verifique si la función $\varphi(x) = \frac{x+\ln 2}{x+1}$ satisface todas las condiciones del teorema de existencia y unicidad del punto fijo en el intervalo I. ¿Por qué no sirve tomar $\varphi(x) = \frac{\ln 2}{x}$?

   c) Ahora utilice el método de iteración de punto fijo con $x_0 = 0$ para determinar una aproximación con 11 iteraciones ($k = 10$) de la raíz de la ecuación $f'(x) = 0$ en I. [Sugerencia: use `MATLAB`]

   d) Determine el valor de $f(x_{10})$ para la aproximación encontrada con el método del punto fijo. Para ello puede usar el hecho de que $\displaystyle\int_0^x e^{t^2}\,dt \approx x + \frac{x^3}{3} + \frac{x^5}{10}$ cuando $x \approx 0$. ¿Cuál es el error relativo con respecto a la raíz exacta de $f'(x) = 0$?

6. Considere el polinomio $\dfrac{x}{8}(63x^4 - 70x^2 + 15)$, el cual tiene un cero en [0.6, 1] dado por $\xi = \frac{1}{21}\sqrt{245 + 14\sqrt{70}} \approx 0.9061798459$. Determine la cantidad de iteraciones máximas que se

necesitan para aproximar este cero por el método de bisección con una tolerancia de 10^{-10}. Luego utilice `MATLAB` para aproximar este cero.

7. Dada la ecuación $x\,\text{sen}(x) + x^2 = 1$, utilice el método de la secante para aproximar un cero con tolerancia 10^{-3} tal que $x_0 = 0.1$ y $x_1 = 0.9$.

8. Muestre que la ecuación $e^x = 2x^2$ tiene una raíz negativa. Use el método de Newton con un valor apropiado de x_0 y determine una aproximación de la raíz negativa con una precisión de 10^{-5}.

9. Aplique el método de Newton para determinar una aproximación de una raíz de $f(x) = x^2 \cos x - 6x \ln x - 25$ con $x_0 = 23$ como condición inicial.

10. Considere la función $f(x) = e^x - \text{sen}(x) - 4$ en $[1, 2]$. Determine

    a) Una función $\varphi(x)$ que satisfaga las condiciones del teorema del punto fijo.

    b) Deduzca la cota teórica para el número máximo de iteraciones necesarias que se requieren para que el método del punto fijo alcance una aproximación con exactitud de `tol`> 0:

$$\texttt{iterMax} = \left[\!\!\left[\frac{\ln(\texttt{tol}(1-L)) - \ln|x_1 - x_0|}{\ln(L)} \right]\!\!\right] + 1$$

    Utilizando los datos del problema dado, calcule dicha cota con la aproximación inicial $x_0 = 1$, usando $\texttt{tol} = 10^{-3}$.

    c) Aplique el método del punto fijo y determine una aproximación para la solución de $f(x) = 0$, con una exactitud de 10^{-3} en el intervalo $[1, 2]$ y con la aproximación inicial $x_0 = 1$. También calcule el error relativo en cada iteración.

    d) ¿La cantidad de iteraciones llevadas es igual a `iterMax`? Compruebe esto con `MATLAB`.

11. Considere la ecuación $f(x) = 0$, donde $f \in C^2(\mathbb{R})$, y el método de Halley definido por

$$x_k := x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1}) - \dfrac{1}{2} f''(x_{k-1}) \dfrac{f(x_{k-1})}{f'(x_{k-1})}}, \qquad k = 1, 2, 3, ...$$

    a) Muestre que el método de Halley se puede ver como una aplicación del método de Newton-Rhapson a la función $g(x) = \dfrac{f(x)}{\sqrt{f'(x)}}$, con g bien definida en un conjunto adecuado que contiene a la raíz de f.

    b) Sea $A \in \mathbb{R}^+$. Escriba de forma simplificada el método de Halley al cálculo de x_{k+1}, para una aproximación de $\sqrt{A}$, utilizando una función adecuada y tomando $x_0 := 1$. Use esto para aproximar $\sqrt{5}$. Implemente un algoritmo en `MATLAB` del método de Halley.

Capítulo 4

Sistemas lineales

Un sistema lineal no homogéneo es de la forma

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

donde $\mathbf{A} \in \mathbb{R}^{n\times n}$ y $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ con $\mathbf{b} \neq \mathbf{0}$.

Dicho sistema tiene solución única si y solo si $\mathbf{A}$ no es singular y entonces

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

El cálculo de $\mathbf{A}^{-1}$ computacionalmente hablando es costoso cuando las dimensiones de la matriz son muy grandes, así que evitaremos calcular la inversa, estudiando algoritmos menos costosos.

4.1. Sustitución *hacia atrás* y *hacia adelante*

La solución del sistema es "evidente" si A es triangular superior o inferior. Si algún elemento en la diagonal de estas últimas matrices es cero entonces A es singular.

Consideremos primero una matriz triangular superior A para el sistema $A\boldsymbol{x} = \boldsymbol{b}$:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ 0 & a_{22} & \cdots & a_{2,n-1} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

Por la última línea, $x_n = \dfrac{b_n}{a_{nn}}$. La $(n-1)-$ésima ecuación es

$$a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1}$$

$$\Rightarrow a_{n-1,n-1}x_{n-1} = b_{n-1} - a_{n-1,n}x_n$$

$$\Rightarrow x_{n-1} = \frac{1}{a_{n-1,n-1}}\left(b_{n-1} - a_{n-1,n}\cdot\frac{b_n}{a_{nn}}\right)$$

Algoritmos de sustitución

En general,

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right) \quad \text{con } i = n, n-1, ..., 2, 1.$$

lo cual se conoce como **algoritmo de sustitución hacia atrás**.

En el caso en el que la matriz A es triangular superior el **algoritmo de sustitución hacia adelante** es

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j\right) \quad \text{con } i = 1, 2, ..., n-1, n.$$

A continuación los scripts para las sutituciones expuestas.

Sustitución hacia adelante

```
function [x] = Forward(A,b)
% Resuelve el sistema triangular inferior Ax=b
% Inputs: Matriz a Triangular Inferior, b vector constantes
% Output: x solucion
[n,m]=size(A);       % Dimensiones de A
if n ~= m            % Comprobar que A sea cuadrada
    error('La matriz no es cuadrada');
end
if size(b,1) ~= n
    error('el vector de constantes no tiene dimension adecuada');
end
x = zeros(n,1);                  % x inicial
x(1)=b(1)/A(1,1);
```

```
for k=2:n
        t = 0;
        for j=1:k-1
            t = t + A(k,j) * x(j);
        end
        x(k) = (b(k) - t )/A(k , k) ;
end
end
```

Sustitución hacia atrás

```
function [x] = Backward(A,b)
% Resuelve el sistema triangular superior Ax=b
% Inputs: Matriz a Triangular Superior, b vector constantes
% Output: x solucion
[n,m]=size(A);      % Dimensiones de A
if n ~= m           % Comprobar que A sea cuadrada
    error('La matriz no es cuadrada');
end
if size(b,1) ~= n
    error('el vector de constantes no tiene dimension adecuada');
end
x = zeros(n,1);                 % x inicial
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
        t = 0;
        for j=k+1:n
            t= t + A(k,j) * x(j);
        end
        x(k) = (b(k) - t)/A(k , k) ;
end
end
```

4.2. Eliminación gaussiana y pivoteo

Dado el sistema $A\boldsymbol{x} = \boldsymbol{b}$, la solución $\boldsymbol{x}$ no cambiará si cualquiera de las siguientes operaciones se aplican a la ecuación $k-$ésima:

- $\alpha \cdot \text{Eq}_k, \quad \alpha \neq 0;$
- $\alpha \cdot \text{Eq}_i + \text{Eq}_k;$
- $\text{Eq}_k \longleftrightarrow \text{Eq}_j$

Dichas operaciones usualmente se efectúan para llevar la matriz a una triangular superior (o inferior) y esto es conocido como **eliminación gaussiana**.

En realidad dichas operaciones no son más que multiplicar la matriz ampliada $(A|b)$ por una matriz de permutación elemental.

Definición 4.1 (Matriz elemental)**.** Una matriz elemental cuadrada de orden $n \times n$ es una permutación de la matriz identidad I_n las cuales son:

- Intercambio de la fila $i-$ésima por la fila $j-$ésima: $T_{i,j} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$
- Multiplicación de la fila $i-$ésima por el escalar $\alpha \neq 0$: $D_i(\alpha) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \alpha & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$
- Suma de α veces la fila i más la fila j: $L_{ij}(\alpha) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & \alpha & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$

Si la matriz A es invertible y E_i son matrices elementales entonces un resultado nos dice que $E_1E_2\cdots E_kA = I$.

La estrategia usual es la del *pivoteo* y dependiendo de cómo se elija se puede evitar acumular errores en los cálculos.

En el pivoteo la entrada pivote debe ser distinta de cero. El pivoteo puede ser visto como un acomodo de filas (o columnas) en una matriz. El intercambio de filas es equivalente a multiplicar A por una matriz de permutación. En la práctica las entradas de la matriz raramente son movidas pues esto tiene un costo muy alto en tiempo. Por esto los algoritmos deben evitar las permutaciones. El pivoteo aumenta el costo computacional general de un algoritmo. Sin embargo, a veces el pivoteo es necesario para que el algoritmo elegido funcione del todo, e incluso otras veces puede incrementar la estabilidad numérica.

Un pivoteo exitoso requiere la comparación del tamaño relativo de un coeficiente con respecto a otros coeficientes en una misma ecuación. Esto se calcula dividiendo cada coeficiente entre el valor absoluto más grande en la fila. Esto se conoce como **pivoteo escalado**. Tenemos dos tipos de este pivoteo:

Pivoteos

- *Pivoteo parcial:* consiste en intercambiar filas de manera que

$$|a_{kk}| = \max_{j=k}^{n}\{|a_{jk}|\}, \qquad \text{para } k = 1, 2, ..., n-1$$

  al hacer ceros por debajo de la diagonal. Se toma como pivote el mayor valor absoluto de los elementos de la columna por debajo de la diagonal antes de empezar a hacer estos valores cero.

- *Pivoteo total:* consiste en intercambiar filas y columnas de la matriz de forma que

$$|a_{kk}| = \text{máx}\{|a_{ij}| : k \leq j \text{ y } k \leq i\}, \qquad \text{para } k = 1, 2, ..., n-1,$$

  es decir, se toma como pivote al coeficiente de mayor valor absoluto dentro del arreglo $[k, n] \times [k, n]$.

El pivoteo parcial es el más comúnmente utilizado debido a que su costo computacional no es tan alto.

4.3. Factorización LU

La idea es utilizar eliminación gaussiana sin pivoteo.

Para el sistema $Ax = b$ vamos a factorizar la matriz $A = LU$, donde

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ l_{n-1,1} & \cdots & l_{n-1,n-1} & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \ddots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}$$

Entonce $L(Ux) = b \Leftrightarrow Ly = b \quad \wedge \quad Ux = y$, los cuales se pueden resolver por sustitución hacia adelante y hacia atrás respectivamente.

Otras aplicaciones de la factorización LU son

- Cálculo del determinante:

$$\det A = \det L \cdot \det U = \prod_{k=1}^{n} l_{kk} \cdot \prod_{k=1}^{n} u_{kk}$$

- Prueba de no singularidad: $A = LU$ no es singular si y solo si los elementos de las diagonales de L y U son distintos de cero.
- Cálculo de inversa: $A^{-1} = U^{-1}L^{-1}$

Algoritmo para calcular la factorización LU

Denotemos las columnas de L como $\ell_1, \ell_2, ..., \ell_n$ y las filas de U como $u_1^\intercal, u_2^\intercal, ..., u_n^\intercal$. Luego

$$A = LU = \begin{pmatrix} \ell_1 & \dots & \ell_n \end{pmatrix} \begin{pmatrix} u_1^\intercal \\ \vdots \\ u_n^\intercal \end{pmatrix} = \sum_{k=1}^{n} \ell_k u_k^\intercal$$

Asumamos que A no es singular y que admite la factorización LU. Entonces los elementos de la diagonal de L son no nulos. Como $\ell_k u_k^\intercal$ se mantiene igual si ℓ_k es reemplazado por $\alpha\ell_k$ y u_k es reemplazado por $\alpha^{-1}u_k$, con $\alpha \neq 0$, entonces podemos asumir que todos los elementos de la diagonal de L son iguales a 1.

Como las primeras $k-1$ componentes de ℓ_k y u_k son ceros, cada matriz $\ell_k u_k^\intercal$ tiene ceros en las primeras $k-1$ filas y columnas. Se sigue que $u_1^\intercal$ es la primera fila de A y ℓ_1 es la primera columna

de A dividida por a_{11}, pues $l_{11} = 1$. Teniendo ℓ_1 y u_1 formamos la matriz

$$A_1 = A - \ell_1 u_1^{\mathsf{t}} = \sum_{k=2}^{n} \ell_k u_k^{\mathsf{t}}$$

La primer fila y columna de A_1 es de ceros y se sigue que u_2^{t} es la segunda fila de A_1, mientras que ℓ_2 es la segunda columna de A_1 escalada de tal forma que $l_{22} = 1$. Para generalizar el algortimo de la factorización LU sea $A_0 := A$, para todo $k = 1, ..., n$ sea u_k^{t} la $k-$ésima fila de A_{k-1} y ℓ_k la $k-$ésima columna de A_{k-1} escalada de tal forma que $l_{kk} = 1$. Entonces

$$A_k := A_{k-1} - \ell_k u_k^{\mathsf{t}}$$

Todos los elementos de las primeras k filas y columnas de A_k son ceros. Entonces se pueden usar para almacenar de la matriz original A para acumular L y U. La acumulación total LU requiere $O(n^3)$ operaciones.

LU de forma práctica

Realizamos eliminación gaussiana sin intercambio de filas ni reescalamiento, es decir, solo operaciones de la forma $\alpha_k f_j + f_i$. La idea es obtener la matriz triangular superior U por este método, y las entradas de la matriz triangular inferior unitaria L (unitaria porque $L_{ii} = 1$ para todo i), en la cual $L_{ij} = -\alpha_k$ para la operación $\alpha_k f_j + f_i$, y el resto de entradas son ceros.

Definición 4.2. Suponga que A es una matriz $n \times n$ de entradas reales, con $n \geq 2$. Sea $1 \leq k \leq n$. La *submatriz principal* de orden k de A es definida como la matriz $A^{(k)}$ de dimensiones $k \times k$ que tiene por entrada los elementos a_{ij} de A con $1 \leq i, j \leq k$.

Teorema 4.1 (Existencia de la factorización LU)**.** *Sea A matriz $n \times n$ de entradas reales. Si toda submatriz principal $A^{(k)}$ con $1 \leq k \leq n-1$ es no singular, entonces existen L triangular inferior unitaria y U triangular superior tales que $A = LU$.*

Teorema 4.2 (Unicidad de la factorización LU)**.** *Si A es no singular y además $A = LU$ como antes, entonces la factorización es única.*

La factorización LU es análoga a la eliminación gaussiana al resolver $Ax = b$. La diferencia consiste en que LU no considera b hasta el final.

Observación 4.1. Existen casos en donde la factorización LU de la matriz A no existe, pero sí la de $PA = LU$, en donde P es producto de matrices elementales de permutaciones.

Factorización LU

```
function [L,U]=FactLU(A)
% Calcula la factorizacion LU de la matriz cuadrada A
% L = matriz triangular inferior unitaria, U = matriz triangular
   superior

[n,m]=size(A);
if n~=m
    error('la matriz no es cuadrada');
end
L=eye(n);
U=A;
for k=1:n-1
    for j=k+1:n
        if abs(U(k,k))>1e-12
        L(j,k)=U(j,k)/U(k,k);
        end
        U(j,:)=U(j,:)-L(j,k)*U(k,:);
    end
end
end
```

Resolver sistema con factorización LU

```
function [L,U,x]=LUx(A,b)
[L,U]=FactLU(A);
x=Backward(U,Forward(L,b));
end
```

Ejemplo 4.1. Considere la matriz

$$A = \begin{pmatrix} 8 & 6 & -2 & 1 \\ 8 & 8 & -3 & 0 \\ -2 & 2 & -2 & 1 \\ 4 & 3 & -2 & 5 \end{pmatrix}$$

a) Justifique teóricamente si A posee una factorización LU, donde L es triangular inferior unitaria y U es triangular superior. Explique si dicha factorización es única o no.

b) Calcule paso a paso la factorización LU de la matriz A, donde L es triangular inferior unitaria y U es triangular superior.

c) Resuelva el sistema $Ax = (1, 2, 3, 4)^{\mathtt{t}}$ con los algoritmos de sustitución.

Solución:

a) Al calcular los determinantes de las submatrices principales (con `MATLAB` es ingresar `det(A(1:j,1:j))` con $j = 2, 3, 4$)

$$|A^{(1)}| = 8, \quad |A^{(2)}| = 16, \quad |A^{(3)}| = -12, \quad |A^{(4)}| = |A| = -6$$

la factorización LU existe y además es única pues $|A| \neq 0$.

b) Se realizan operaciones de fila (solamente $\alpha f_i + f_j$) para conseguir una matriz triangular superior U:

$$\begin{pmatrix} 8 & 6 & -2 & 1 \\ 8 & 8 & -3 & 0 \\ -2 & 2 & -2 & 1 \\ 4 & 3 & -2 & 5 \end{pmatrix} \xrightarrow[\frac{1}{4}f_1+f_3 \\ \frac{-1}{2}f_1+f_4]{-1f_1+f_2} \begin{pmatrix} 8 & 6 & -2 & 1 \\ 0 & 2 & -1 & -1 \\ 0 & \frac{7}{2} & \frac{-5}{2} & \frac{5}{4} \\ 0 & 0 & -1 & \frac{9}{2} \end{pmatrix} \xrightarrow{\frac{-7}{4}f_2+f_3} \begin{pmatrix} 8 & 6 & -2 & 1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & \frac{-3}{4} & 3 \\ 0 & 0 & -1 & \frac{9}{2} \end{pmatrix}$$

$$\xrightarrow{\frac{-4}{3}f_3+f_4} \begin{pmatrix} 8 & 6 & -2 & 1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & \frac{-3}{4} & 3 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} = U$$

La matriz $L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{-1}{4} & \frac{7}{4} & 1 & 0 \\ \frac{1}{2} & 0 & \frac{4}{3} & 1 \end{pmatrix}$ se construye a partir de las operaciones realizadas en donde la entrada $L_{ij} = -\alpha$ si $\alpha f_j + f_i$.

c) $Ax = LUx = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{-1}{4} & \frac{7}{4} & 1 & 0 \\ \frac{1}{2} & 0 & \frac{4}{3} & 1 \end{pmatrix} \begin{pmatrix} 8 & 6 & -2 & 1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & \frac{-3}{4} & 3 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$

Se resuelven las ecuaciones $Ly = b$, $Ux = y$:

$$\begin{cases} y_1 & = 1 \\ y_1 + y_2 & = 2 \\ \frac{-1}{4}y_1 + \frac{7}{4}y_2 + y_3 & = 3 \\ \frac{1}{2}y_1 + \quad + \frac{4}{3}y_3 + y_4 & = 4 \end{cases}, \quad \begin{cases} 8x_1 + 6x_2 - 2x_3 + 1x_4 & = y_1 \\ 2x_2 - x_3 - x_4 & = y_2 \\ \frac{-3}{4}x_3 + 3x_4 & = y_3 \\ \frac{1}{2}x_4 & = y_4 \end{cases}$$

Se puede calcular a mano o usando `[y]=Forward(L,b)` para el primer sistema y `[x]=Backward(U,Forward(L,b))` para el segundo, se tiene la solución $\boldsymbol{x} = (-3, 7, 10, 3)^T$.

Igual se puede comprobar utilizando directamente `[L,U,x]=LUx(A,b)`.

◇

4.3.1. Factorización PA = LU

Existen casos en donde no es posible obtener la factorización LU, digamos por ejemplo, cuando la entrada $a_{11} = 0$, y esto se debe a que no utilizamos algún método de pivoteo en la eliminación gaussiana. La idea ahora es utilizar pivoteo parcial.

Teorema 4.3. *Toda matriz $A \in \mathbb{R}^{n\times n}$ tiene una factorización $PA = LU$ donde P es una matriz de permutaciones de fila, L es una matriz unitaria triangular inferior donde todas sus entradas están acotadas por 1 y U es una matriz triangular superior.*

Algorithm 5 Eliminación gaussiana con pivoteo parcial

Require: Matriz $A \in \mathbb{R}^{n\times n}$
Ensure: P, L, U tales que $PA = LU$
  $U \leftarrow A, L \leftarrow I, P \leftarrow I$
  for $k \leftarrow 1 : m-1$ **do**
    Seleccione $i \geq k$ tal que $|U_{i,k}|$ es máximo;
    $U_{k,k:m} \longleftrightarrow U_{i,k:m}$;
    $L_{k,1:k-1} \longleftrightarrow L_{i,1:k-1}$;
    $P_{k,:} \longleftrightarrow P_{i,:}$;
    for $j \leftarrow k+1 : m$ **do**
      $L_{j,k} \leftarrow U_{j,k}/U_{k,k}$;
      $U_{j,k:m} \leftarrow U_{j,k:m} - L_{j,k}U_{k,k:m}$;
    end for
  end for

Se puede utilizar pivoteo total y así obtener lo que se conoce como factorización $PAQ = LU$, donde Q representa los cambios de columna que se realizan en cada paso. Aunque en la práctica se recurre al pivoteo parcial debido a los costos computacionales del pivoteo total.

Ejemplo 4.2. Muestre que la matriz $\mathbf{A} = \begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix}$ no se puede expresar en la forma $\mathbf{LU}$.

Calcule $\mathbf{P}, \mathbf{L}, \mathbf{U}$ tales que $\mathbf{PA} = \mathbf{LU}$.

Solución: Aplicando la siguiente operación

$$\begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix} \xrightarrow{-4f_1+f_2} \begin{pmatrix} 1 & 2 & 6 \\ 0 & 0 & -25 \\ -2 & 3 & 5 \end{pmatrix}$$

es notable que el elemento `A(2,2)` no podrá utilizarse como pivote para hacer ceros debajo de sí, por esto no se puede obtener la factorización $\mathbf{LU}$.

Usaremos pivoteo parcial para obtener la factorización $\mathbf{PA} = \mathbf{LU}$:

$$\begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix} \xrightarrow{f_1 \leftrightarrow f_2} \begin{pmatrix} 4 & 8 & -1 \\ 1 & 2 & 6 \\ -2 & 3 & 5 \end{pmatrix} \xrightarrow[\frac{1}{2}f_1+f_3]{-\frac{1}{4}f_1+f_2} \begin{pmatrix} 4 & 8 & -1 \\ \frac{1}{4} & 0 & \frac{25}{4} \\ \frac{-1}{2} & 7 & \frac{9}{2} \end{pmatrix}$$

$$\xrightarrow{f_2 \leftrightarrow f_3} \begin{pmatrix} 4 & 8 & -1 \\ \frac{-1}{2} & 7 & \frac{9}{2} \\ \frac{1}{4} & 0 & \frac{25}{4} \end{pmatrix} \longrightarrow \begin{pmatrix} 4 & 8 & -1 \\ \frac{-1}{2} & 7 & \frac{9}{2} \\ \frac{1}{4} & 0 & \frac{25}{4} \end{pmatrix}$$

De lo anterior se tiene que

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{f_1 \leftrightarrow f_2} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{f_2 \leftrightarrow f_3} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \mathbf{P}$$

Además,

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{-1}{2} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix} 4 & 8 & -1 \\ 0 & 7 & \frac{9}{2} \\ 0 & 0 & \frac{25}{4} \end{pmatrix}$$

Puede comprobarlo con `MATLAB`, poniendo `[L, U, P] = lu(A)`.

◇

4.4. Factorización de Cholesky

Matriz simétrica real

$A \in \mathbb{R}^{n\times n}$ es una **matriz simétrica** si $A = A^{\mathsf{t}}$.

Queremos expresar $A = LDL^{\mathsf{t}}$ donde L es triangular inferior con 1's en su diagonal y D es una matriz diagonal. Podemos escribir dicha factorización como sigue:

$$A = \begin{pmatrix} \boldsymbol{\ell}_1 & \dots & \boldsymbol{\ell}_n \end{pmatrix} \begin{pmatrix} D_{11} & & & \\ & D_{22} & & \\ & & \ddots & \\ & & & D_{nn} \end{pmatrix} \begin{pmatrix} \boldsymbol{\ell}_1^{\mathsf{t}} \\ \\ \vdots \\ \boldsymbol{\ell}_n^{\mathsf{t}} \end{pmatrix} = \sum_{k=1}^{n} D_{kk}\boldsymbol{\ell}_k\boldsymbol{\ell}_k^{\mathsf{t}}$$

Nuevamente, $\boldsymbol{\ell}_k$ denota la $k-$ésima columna de L.

Este algoritmo explota la simetría de la matriz A y requiere aproximadamente la mitad de las entradas de la matriz.

Similar a la factorización LU, sea $A_0 = A$ y para $k = 1, ..., n$ se tiene que $\boldsymbol{\ell}_k$ es la $k-$ésima columna de A_{k-1}, escalada de tal manera que $L_{kk} = 1$. Sea $D_{kk} = (A_{k-1})_{kk}$ y se calcula

$$A_k = A_{k-1} - D_{kk}\boldsymbol{\ell}_k\boldsymbol{\ell}_k^{\mathsf{t}}$$

Matriz definida positiva

$A \in \mathbb{R}^{n\times n}$ es **definida positiva** si $x^{\mathsf{t}}Ax > 0$ para todo $x \neq \mathbf{0}$.

Teorema 4.4. *Sea $A \in \mathbb{R}^{n\times n}$ simétrica. A es definida positiva si y solo si $A = LDL^{t}$ existe, donde los elementos de la diagonal de D son todos positivos.*

Definición 4.3. Defina $D^{\frac{1}{2}}$ como la matriz diagonal donde $(D^{\frac{1}{2}})_{kk} = \sqrt{D_{kk}}$. Luego $D^{\frac{1}{2}} \cdot D^{\frac{1}{2}} = D$. Para una matriz definida positiva A se tiene que

$$A = (LD^{\frac{1}{2}})(D^{\frac{1}{2}}L^{\mathsf{t}}) = (LD^{\frac{1}{2}})(LD^{\frac{1}{2}})^{\mathsf{t}}$$

Tomando $\tilde{L} := LD^{\frac{1}{2}} \Rightarrow A = \tilde{L}\tilde{L}^{\mathsf{t}}$.

Factorización de Cholesky

Si consideramos la factorización LU de una matriz simétrica A, se espera obtener una factorización en donde $U = L^t$. De esta manera se considera la posibilidad de escribir

$$A = LL^t = R^tR$$

donde L es triangular inferior y $R = L^t$. En este caso no se espera que las entradas de la diagonal de L sean iguales a 1. Dicha descomposición se denomina **factorización de Cholesky**.

Teorema 4.5. *Toda matriz $A \in \mathbb{R}^{n\times n}$ simétrica y definida positiva tiene una factorización de Cholesky $A = LL^t = R^tR$.*

4.5. Factorización QR

Supongamos que $A \in \mathbb{R}^{m\times n}$, con $m \geq n$. Si se consideran las n columnas de A, denotadas por $\mathbf{a}_j \in \mathbb{R}^m$, $1 \leq j \leq n$, se puede obtener una base ortogonal de vectores unitarios $\mathbf{q}_j \in \mathbb{R}^m$, $1 \leq j \leq n$, tales que

$$\langle \mathbf{a}_1, ..., \mathbf{a}_k \rangle = \langle \mathbf{q}_1, ..., \mathbf{q}_k \rangle \quad \text{para } k = 1, ..., n$$

Si A tiene rango completo (i.e. sus columnas son linealmente independientes), se tienen las siguientes ecuaciones

$$\begin{aligned}
\mathbf{a}_1 &= r_{11}\mathbf{q}_1 \\
\mathbf{a}_2 &= r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \\
\mathbf{a}_3 &= r_{13}\mathbf{q}_1 + r_{23}\mathbf{q}_2 + r_{33}\mathbf{q}_3 \\
&\vdots \\
\mathbf{a}_n &= r_{1n}\mathbf{q}_1 + r_{2n}\mathbf{q}_2 + r_{3n}\mathbf{q}_3 + ... + r_{nn}\mathbf{q}_n
\end{aligned}$$

Como los vectores $\mathbf{q}_j$ son ortogonales, se tiene que

$$r_{ij} = \mathbf{q}_i^t\mathbf{a}_j \quad (i \neq j), \qquad |r_{jj}| = \left\| \mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i \right\|_2$$

Luego,

$$\left(\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ & & & \end{array}\right) = \left(\begin{array}{c|c|c|c} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \\ & & & \end{array}\right) \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix}$$

es decir, $A = \hat{Q}\hat{R}$, donde $\hat{Q} \in \mathbb{R}^{m\times n}$ satisface que $\hat{Q}^t\hat{Q} = I_n$ y $\hat{R} \in \mathbb{R}^{n\times n}$ es triangular superior. Esta descomposición se denomina *factorización QR reducida*.

Si A no tiene rango completo, entonces algún $\mathbf{a}_k \in \langle \mathbf{a}_1, ..., \mathbf{a}_{k-1}\rangle$, con lo que $r_{kk} = 0$. En dicho caso $\mathbf{q}_k$ no está bien definido, pero basta con seleccionarlo de tal manera que sea ortogonal a $\langle \mathbf{q}_1, ..., \mathbf{q}_{k-1}\rangle$. Así también se puede escribir $A = \hat{Q}\hat{R}$.

Si se extiende $\{\mathbf{q}_1, ..., \mathbf{q}_n\}$ a una base de $\mathbb{R}^m$ y se denota por Q a la matriz unitaria cuyas columnas son $\{\mathbf{q}_1, ..., \mathbf{q}_m\}$, se puede escribir la *factorización QR completa* de A dada por $A = QR$, donde $Q \in \mathbb{R}^{m\times m}$ unitaria y $R \in \mathbb{R}^{m\times n}$ triangular superior (la cual se obtiene al agregar $m - n$ filas nulas al final de $\hat{R}$).

Teorema 4.6. *Toda matriz $A \in \mathbb{R}^{m\times n}$, $m \geq n$, admite una descomposición reducida $\hat{Q}\hat{R}$, donde $\hat{Q} \in \mathbb{R}^{m\times n}$, $\hat{Q}^t\hat{Q} = I_n$ y $\hat{R}$ es triangular superior. Además admite una factorización completa $A = QR$, con $Q \in \mathbb{R}^{m\times m}$ unitaria y $R \in \mathbb{R}^{m\times n}$ triangular superior.*

4.5.1. Ortogonalización de Gram-Schmidt

Este es un método de ortogonalización triangular, en el cual se multiplica a la derecha de A por matrices triangulares superiores para obtener $\hat{Q}$. Las ecuaciones anteriores se pueden reescribir como

$$\begin{aligned} \mathbf{q}_1 &= \frac{1}{r_{11}}\mathbf{a}_1 \\ \mathbf{q}_2 &= \frac{1}{r_{22}}(\mathbf{a}_2 - r_{12}\mathbf{q}_1) \\ \mathbf{q}_3 &= \frac{1}{r_{33}}(\mathbf{a}_3 - r_{13}\mathbf{q}_1 - r_{23}\mathbf{q}_2) \\ &\vdots \\ \mathbf{q}_n &= \frac{1}{r_{nn}}\left(\mathbf{a}_n - \sum_{i=1}^{n-1} r_{in}\mathbf{q}_i\right) \end{aligned}$$

el cual es conocido como el proceso de ortogonalización de Gram-Schmidt. El algoritmo previo es inestable debido a errores de redondeo. En caso de que A tenga rango completo se pueden definir

Algorithm 6 Ortogonalización de Gram-Schmidt (inestable)

Require: Matriz $A \in \mathbb{R}^{m\times n}$
Ensure: Matriz unitaria Q y triangular superior R tales que $A = QR$
  for $j \leftarrow 1 : n$ **do**
    $\mathbf{v}_j \leftarrow \mathbf{a}_j$;
    for $i \leftarrow 1 : j-1$ **do**
      $r_{ij} \leftarrow \mathbf{q}_i^{\mathsf{t}}\mathbf{a}_j$;
      $\mathbf{v}_j \leftarrow \mathbf{v}_j - r_{ij}\mathbf{q}_i$;
    end for
    $r_{jj} \leftarrow \|\mathbf{v}_j\|_2$;
    $\mathbf{q}_j \leftarrow \mathbf{v}_j / r_{jj}$;
  end for

las matrices triangulares superiores

$$R_1 = \begin{pmatrix} \frac{1}{r_{11}} & -\frac{r_{12}}{r_{11}} & -\frac{r_{13}}{r_{11}} & \cdots & -\frac{r_{1n}}{r_{11}} \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \qquad R_2 = \begin{pmatrix} 1 & & & & \\ & \frac{1}{r_{22}} & -\frac{r_{23}}{r_{22}} & \cdots & -\frac{r_{2n}}{r_{22}} \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

$$\cdots \qquad R_n = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & \frac{1}{r_{nn}} \end{pmatrix}$$

Al calcular $AR_1R_2\cdots R_n = \hat{Q} \in \mathbb{R}^{m\times n}$ cuyas columnas son los vectores $\mathbf{q}_i$. De esta manera, si se define $\hat{R} = (R_1\cdots R_n)^{-1}$ se tiene que $A = \hat{Q}\hat{R}$ que es la factorización reducida QR de A.

Algorithm 7 Ortogonalización de Gram-Schmidt (estable)

Require: Matriz $A \in \mathbb{R}^{m\times n}$
Ensure: Matriz unitaria Q y triangular superior R tales que $A = QR$
  $V \leftarrow A$
  for $i \leftarrow 1 : n$ **do**
    $r_{ii} \leftarrow \|\mathbf{v}_i\|_2$;
    $\mathbf{q}_i \leftarrow \mathbf{v}_i / r_{ii}$;
    for $j \leftarrow i+1 : n$ **do**
      $r_{ij} \leftarrow \mathbf{q}_i^{\mathsf{t}}\mathbf{v}_j$;
      $\mathbf{v}_j \leftarrow \mathbf{v}_j - r_{ij}\mathbf{q}_i$;
    end for
  end for

Lema 4.7. *El número de operaciones en aritmética de punto flotante (flops) para calcular la factorización QR de una matriz $A \in \mathbb{R}^{m\times n}$ dada por los algoritmos previos es de orden $O(2mn^2)$.*

4.5.2. Reflexiones de Householder

Ahora se van a definir matrices ortogonales $Q_1, ..., Q_n$ tales que

$$Q_n \cdots Q_1 A = \hat{R}$$

Entonces se define $\hat{Q} := (Q_n \cdots Q_1)^{\mathrm{t}}$. De esta manera se obtiene $A = \hat{Q}\hat{R}$. Dicho proceso se denomina triangularización ortogonal. A continuación se describe el método de Householder.

Definición 4.4. Sea $\boldsymbol{u} \in \mathbb{R}^n$ no nulo. La matriz $n \times n$ de la forma

$$H_n = I_n - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{t}}}{\|\boldsymbol{u}\|^2}$$

se denomina *reflexión de Householder*.

Una reflexión de Householder describe una reflexión sobre un hiperplano el cual es ortogonal al vector $\dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$ y que además contiene al origen. La matriz de Householder es simétrica y ortogonal, pues

$$\left(I - \frac{2uu^{\mathrm{t}}}{\|u\|^2}\right)^{\mathrm{t}} \left(I - \frac{2uu^{\mathrm{t}}}{\|u\|^2}\right) = I - \frac{4uu^{\mathrm{t}}}{\|u\|^2} + \frac{4u(u^{\mathrm{t}}u)u^{\mathrm{t}}}{\|u\|^4} = I$$

Con cada multiplicación de la matriz de Householder con una matriz A de dimensiones $n \times m$ lo que se desea es introducir ceros debajo de la diagonal en una columna entera. Para comenzar se construye una reflexión la cual transforma la primer columna no nula $\boldsymbol{x} \in \mathbb{R}^n$ de A en un múltiplo del vector unitario $\boldsymbol{e_1}$. Se busca elegr $u \in \mathbb{R}^n$ tal que las últimas $n-1$ entradas de

$$\left(I - \frac{2uu^{\mathrm{t}}}{\|u\|^2}\right) x = x - \frac{2u^{\mathrm{t}}x}{\|u\|^2}u \qquad (\star)$$

se anulen.

Para u se hace

$$\boldsymbol{u} = \boldsymbol{x} + \text{signo}(x_1) \cdot \|\boldsymbol{x}\| \cdot \boldsymbol{e}_1$$

Usualmente se escoge el signo de acuerdo al signo de x_1 para evitar que $\|\boldsymbol{u}\|$ sea demasiado pequeño.

Ahora se calculan matrices de Householder $\boldsymbol{\hat{H}_k}$ cuya dimensión cada vez va decreciendo. Supongamos que las primeras $k-1$ columnas han sido tansformadas tales que tengamos ceros debajo de la diagonal. Necesitamos escoger la siguiente reflexión de Householder para transformar la $k-$ésima columna tal que las primeras $k-1$ columnas permanezcan en dicha forma. Entonces las primeras $k-1$ entradas de $\boldsymbol{u}$ son ceros. Con esta escogencia las primeras $k-1$ filas y columnas de la matriz

$uu^{\mathtt{t}}$ son ceros y la submatriz $(k-1)\times(k-1)$ de la matriz de reflexión de Householder es la identidad I_k:

$$H_k = \begin{pmatrix} \boldsymbol{I_k} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\hat{H}_k} \end{pmatrix} \in \mathbb{R}^{n\times n}$$

Sea $u_k = x_k \pm \sqrt{\sum_{i=k}^{n} x_i^2}$ y $u_i = x_i$ para $i = k+1, ..., n$. Esto introduce ceros debajo de la diagonal en la $k-$ésima columna.

El resultado final despues de aplicar el algoritmo a todas las columnas de A en orden es una matriz triangular superior $R = H_n H_{n-1} \cdots H_2 H_1 A$, y $Q = H_1 H_2 \cdots H_{n-1} H_n$.

Algorithm 8 Triangularización de Householder

Require: Matriz $A \in \mathbb{R}^{m\times n}$
Ensure: Matriz unitaria Q y triangular superior R tales que $A = QR$
  for $j \leftarrow 1 : n$ **do**
    $\mathbf{x} \leftarrow A_{j:m,j}$;
    $\mathbf{v}_j \leftarrow \mathbf{x} + \text{sign}(x_1)\mathbf{e}_1$;
    $\mathbf{v}_j \leftarrow \mathbf{v}_j / \|\mathbf{v}_j\|_2$;
    $A_{j:m,j:n} \leftarrow A_{j:m,j:n} - 2\mathbf{v}_j \left(\mathbf{v}_j^{\mathtt{t}} A_{j:m,j:m}\right)$;
  end for

Lema 4.8. *El algoritmo previo requiere un orden de* $2mn^2 - \frac{2}{3}n^3$ *flops.*

En `MATLAB` la factorización QR se puede realizar poniendo `[Q,R]=qr(A)`. Igualmente, podemos utilizar el siguiente script:

Factorización QR con Householder

```
function [Q,R]=QRHouse(A)
[m,n]=size(A);
R=A; % Comienza con R = A
Q=eye(m); %  Q inicia como la identidad
for k=1:n
    x=zeros(m,1);
    x(k:m,1)=R(k:m,k);
    g=sign(x(k))*norm(x);
    v=x;
    v(k)=x(k)+g;
```

```
    s=norm(v);
    if s~=0, w=v/s; u=2*R'*w;
        R=R-w*u'; % Producto HR
        Q=Q-2*Q*w*w'; % Producto QR
    end
end
```

Resolver sistema con factorización QR

```
function [Q,R,x]=QRx(A,b)
[Q,R]=QRHouse(A);
x=Backward(R,Q'*b);
end
```

Ejemplo 4.3. Considere la matriz

$$\mathbf{A} = \begin{pmatrix} 2 & -2 & 18 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{pmatrix}$$

a) Aplicando matrices de transformación de Householder sobre $\mathbf{A}$, encuentre las matrices $\mathbf{Q}$ ortogonal y $\mathbf{R}$ triangular superior, tales que $\mathbf{A} = \mathbf{QR}$.

Solución: Los vectores y matrices respectivas son:

$$u_1 = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}, \quad u_2 = \begin{pmatrix} \frac{24}{5} \\ \frac{12}{5} \end{pmatrix}, \quad H_1 = \begin{pmatrix} \frac{-2}{3} & \frac{-2}{3} & \frac{-1}{3} \\ \frac{-2}{3} & \frac{11}{15} & \frac{-2}{15} \\ \frac{-1}{3} & \frac{-2}{15} & \frac{14}{15} \end{pmatrix}, \quad H_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{-3}{5} & \frac{-4}{5} \\ 0 & \frac{-4}{5} & \frac{3}{5} \end{pmatrix}$$

$$Q = H_1 H_2 = \begin{pmatrix} \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{-2}{3} & \frac{-1}{3} & \frac{-2}{3} \\ \frac{-1}{3} & \frac{-2}{3} & \frac{2}{3} \end{pmatrix}, \quad R = \begin{pmatrix} -3 & 0 & -12 \\ 0 & -3 & 12 \\ 0 & 0 & 6 \end{pmatrix}$$

b) Utilizando la factorización del inciso a), resuelva **utilizando el algoritmo de sustitución hacia atrás**, el sistema $\mathbf{Ax} = \mathbf{b}$ con $\mathbf{b} = (1, 0, 6)^T$.

Solución:

$$Ax = b \Longrightarrow QRx = b \Longrightarrow Rx = Q^{-1}b \Longrightarrow Rx = Q^T b$$

Lo último se debe a que Q es ortogonal. Luego se tiene el sistema triangular superior siguiente:

$$\begin{aligned} -3x_1 + 0x_2 - 12x_3 &= \tfrac{-8}{3} \\ 0x_1 - 3x_2 - 12x_3 &= \tfrac{-10}{3} \\ 0x_1 + 0x_2 + 6x_3 &= \tfrac{13}{3} \end{aligned}$$

Con el algoritmo de sustitución hacia atrás se obtiene

$$x_3 = \frac{13}{18}, \quad x_2 = 4, \quad x_1 = -2$$

Ejemplo 4.4. El siguiente problema tiene 2 incisos.

a) Considere la matriz de rotación $M(\theta) := \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{pmatrix}$ con $0 < \theta < \dfrac{\pi}{2}$.

Utilizando reflexiones de Householder, muestre que $M(\theta) = QR$, en donde $Q = \begin{pmatrix} -\cos\theta & -\operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{pmatrix}$,

y $R = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$.

. . . . . .

Solución:

$x = (\cos\theta, \operatorname{sen}\theta)' \implies \|x\|_2 = 1 \implies u = x + \operatorname{sign}(x_1)\|x\|_2 \cdot e_1 = (\cos\theta, \operatorname{sen}\theta)' + (1,0)' = (1+\cos\theta, \operatorname{sen}\theta)'$

El signo se escoge "positivo" ya que $\cos\theta > 0$ si $\theta \in \left]0, \frac{\pi}{2}\right[$.

Tenemos que

$$u'u = \|(1+\cos\theta, \operatorname{sen}\theta)'\|^2 = 1 + 2\cos\theta + \cos^2\theta + \operatorname{sen}^2\theta = 2(1+\cos\theta)$$

$$u\,u' = \begin{pmatrix} (1+\cos\theta)^2 & \operatorname{sen}\theta(1+\cos\theta) \\ \operatorname{sen}\theta(1+\cos\theta) & \operatorname{sen}^2\theta \end{pmatrix}$$

Luego,

$$H = I_2 - \frac{2}{u'u} u\,u'$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{2}{2(1+\cos\theta)} \begin{pmatrix} (1+\cos\theta)^2 & \operatorname{sen}\theta(1+\cos\theta) \\ \operatorname{sen}\theta(1+\cos\theta) & \operatorname{sen}^2\theta \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1+\cos\theta & \operatorname{sen}\theta \\ \operatorname{sen}\theta & \frac{\operatorname{sen}^2\theta}{1+\cos\theta} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1+\cos\theta & \operatorname{sen}\theta \\ \operatorname{sen}\theta & \frac{1-\cos^2\theta}{1+\cos\theta} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1+\cos\theta & \operatorname{sen}\theta \\ \operatorname{sen}\theta & 1-\cos\theta \end{pmatrix}$$

$$= \begin{pmatrix} -\cos\theta & -\operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{pmatrix} = Q$$

Entonces,

$$R = HM(\theta) = \begin{pmatrix} -\cos\theta & -\operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

b) Resuelva el sistema $M(\frac{\pi}{4}) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, utilizando la factorización QR obtenida en el inciso a), esto es: aplicando el algoritmo de sustitución hacia atrás al sistema $R \begin{pmatrix} x \\ y \end{pmatrix} = Q^{\mathtt{t}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

. . . . . .

Solución: Por la descomposición obtenida se tiene que

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}^{\mathtt{t}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Con sustitución hacia atrás, tenemos que $y = -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} = 0$, $-x = \frac{-\sqrt{2}}{2} - \frac{\sqrt{2}}{2} = -\sqrt{2}$.

Por lo tanto,

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{2} \\ 0 \end{pmatrix}$$

◇

4.6. Ejercicios

4.6.1. Factorización LU

1. Encuentre la factorización LU de las siguiente matrices:

$$\mathbf{A}_1 = \begin{pmatrix} 2 & 4 & 0 & -2 \\ 1 & -4 & 2 & 1 \\ 2 & 1 & -6 & -8 \\ -4 & -8 & 2 & 0 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} 2 & -1 & 2 & 4 \\ 1 & 1 & 4 & 3 \\ 1 & 1 & -1 & 0 \\ 2 & -1 & 2 & 3 \end{pmatrix}$$

   Utilizando la factorizaciones obtenidas, resuelva los sistemas $\mathbf{Ax} = \mathbf{b}$, donde $\mathbf{b} = (-4, 6, -10, 20)^{\mathtt{t}}$

2. Realice la descomposición LU de la matriz

$$\begin{pmatrix} 1 & 2 & -3 & 4 & -2 \\ 4 & 11 & -10 & 19 & -7 \\ 4 & 2 & -13 & 8 & -5 \\ -1 & 4 & -2 & 12 & -14 \\ 3 & 12 & 13 & 26 & 13 \end{pmatrix}$$

   ¿Es única dicha factorización? Si la respuesta es afirmativa, resuelva el sistema $\mathbf{Ax} = \mathbf{b}$, con $\mathbf{b} = (1, 0, 1, 0, 2)^{\mathtt{t}}$.

3. Considere el sistema lineal $Ax = b$ en donde

$$A = \begin{pmatrix} 1 & 1 & -1 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{pmatrix}, \qquad b = \begin{pmatrix} -4 \\ -7 \\ 13 \\ -13 \end{pmatrix}$$

   Realice sustitución hacia atrás a mano y compruebe su respuesta poniendo en la ventana de comandos `x = Backward(A,b)`

4. Considere el sistema lineal $Ax = b$ en donde

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{-7}{3} & 1 & 0 \\ 2 & \frac{-8}{3} & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ -9 \\ -6 \end{pmatrix}$$

   Realice sustitución hacia adelante a mano y compruebe su respuesta poniendo en la ventana

de comandos `x = Forward(A,b)`

5. Utilice `[L,U]=FactLU(A)` para determinar la factorización LU de

$$A = \begin{pmatrix} 1 & 2 & -3 & 4 & -2 \\ 4 & 11 & -10 & 19 & -7 \\ 4 & 2 & -13 & 8 & -5 \\ -1 & 4 & -2 & 12 & -14 \\ 3 & 12 & 13 & 26 & 13 \end{pmatrix}$$

6. Resuelva el sistema lineal utilizando factorización LU:

$$\begin{cases} -15x_1 - 6x_2 + 9x_3 &= 0 \\ 35x_1 - 4x_2 - 12x_3 &= -9 \\ -30x_1 + 36x_2 - 16x_3 &= -6 \end{cases}$$

Utilice `[L,U,x]=LUx(A,b)` y compare con sus respuestas.

7. Considere la integral indefinida

$$\int \frac{2022x - 1}{(x-1)(x^2+1)}\, dx = \int \left(\frac{a_1}{x-1} + \frac{a_2 x + a_3}{x^2+1} \right) dx$$

Vamos a calcular el vector solución exacto $(a_1, a_2, a_3)^{\mathtt{t}}$.

a) Plantee un sistema de ecuaciones lineales $\mathbf{Ax} = \mathbf{b}$ con $\mathbf{x} = (a_1, a_2, a_3)^{\mathtt{t}}$. Muestre que la matriz $\mathbf{A}$ posee factorización $\mathbf{LU}$, donde $\mathbf{L}$ es triangular inferior unitaria y $\mathbf{U}$ es triangular superior.

b) Determine la factorización $\mathbf{A} = \mathbf{LU}$, en donde $\mathbf{L}$ es triangular inferior unitaria, $\mathbf{U}$ es triangular superior. Justifique por qué dicha factorización es única.

c) Resuelva el sistema del inciso a) por medio de la factorización $\mathbf{LU}$ (haciendo sustitución hacia adelante y hacia atrás) y escriba las fracciones parciales del problema con las constantes encontradas.

4.6.2. Factorización QR con Householder

1. Sea $\mathbf{H} = \mathbf{I}_n - \dfrac{2\mathbf{u}\mathbf{u}^{\mathtt{t}}}{\mathbf{u}^{\mathtt{t}}\mathbf{u}}$ una matriz de Householder. Pruebe que (i) $\mathbf{Hu} = -\mathbf{u}$, y (ii) $\mathbf{Hv} = \mathbf{v}$ si $\mathbf{v}^{\mathtt{t}}\mathbf{u} = 0$

2. Sean $\mathbf{x} \in \mathbb{R}^{n\times 1} - \{\mathbf{0}\}$, $\mathbf{u} = \mathbf{x} - \|\mathbf{x}\|_2\mathbf{e}_1$ y $\tau = \frac{1}{2}\mathbf{u}^t\mathbf{u}$. Demuestre que

$$\left(\mathbf{I}_n \pm \frac{1}{\tau}\mathbf{u}\mathbf{u}^t\right)\mathbf{x} = \mp\|\mathbf{x}\|_2\mathbf{e}_1$$

donde $\mathbf{e}_1 = (1, 0, ..., 0)^t \in \mathbb{R}^{n\times 1}$.

3. Considere la matriz

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

Determine una factorización $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ reducida por el método de Gram-Schmidt y una factorización completa $\mathbf{A} = \mathbf{QR}$ por el método de reflexiones de Householder.

4. Dado el vector $\mathbf{u} = (1, 1, 1)^t$ y $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ calcule $\mathbf{HA}$ y $\mathbf{AH}$, donde $\mathbf{H}$ es la matriz de Householder con respecto a $\mathbf{u}$.

5. Considere la matriz

$$\mathbf{A} = \begin{pmatrix} 8 & -2 & 4 \\ 4 & 23 & 2 \\ 8 & 22 & -23 \end{pmatrix}$$

   a) Aplicando matrices de transformación de Householder sobre $\mathbf{A}$, encuentre las matrices $\mathbf{Q}$ ortogonal y $\mathbf{R}$ triangular superior, tales que $\mathbf{A} = \mathbf{QR}$.

   b) Muestre que la matriz $\mathbf{Q}$ obtenida en a) es ortogonal.

   c) Utilizando la factorización obtenida en a), resuelva el problema $\mathbf{Ax} = \mathbf{b}$, con $\mathbf{b} = (1, 2, 3)^t$.

6. Digite en la ventana de comandos lo siguiente:

```
A=[8 -2 4; 4 23 2; 8 22 -23];
format rat
[Q,R]=qr(A)
```

   ¿Coincide dicha factorización con la que realizó en el ejercicio 5.a) de Householder?

7. Guíese con el script de `[L,U,x]=LUx(A,b)` para crear `function [Q,R,x]=QRHx(A,b)` que resuelva el sistema $Ax = b$ por medio de la factorización QR de Householder, cuyos inputs

sean una matriz cuadrada A y un vector b. Compruebe su funcionamiento con el ejercicio 5.c) de Householder.

4.6.3. Más ejercicios

1. Considere la matriz

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

   a) Aplicando matrices de transformación de Householder sobre $\mathbf{A}$, encuentre las matrices $\mathbf{Q}$ ortogonal y $\mathbf{R}$ triangular superior, tales que $\mathbf{A} = \mathbf{QR}$. No redondee ni trunque los resultados, utilice valores exactos.

   b) Utilizando la factorización del inciso a), resuelva el sistema $\mathbf{Ax} = \mathbf{b}$ con $\mathbf{b} = (3, -21, -6)^{\mathtt{t}}$.

2. Elabore una M-función que tenga como input una matriz cuadrada A y que tenga como output un vector v que tenga como entradas $v_k = \det(A^{(\mathtt{k})})$, es decir, un vector que almacene los determinantes de las submatrices principales de A, incluyendo a la matriz A.

3. Utilice pivoteo parcial y pivoteo total redondeando a dos decimales, para resolver el sistema lineal

$$\begin{cases} 23.12x_1 + 7.86x_2 - 8.15x_3 &= 15.26 \\ 12.01x_1 + 2.67x_2 - 56.43x_3 &= 9.34 \\ -32.12x_1 + 10.00x_2 - 4.32x_3 &= -42.12 \end{cases}$$

   Sabiendo que que la solución exacta es $\mathbf{x} = (1, -1, 0)^{\mathtt{t}}$, calcule el valor relativo de cada solución obtenida con la norma $\|\cdot\|_\infty$

4. Determine la factorización $\mathbf{LU}$ para resolver el sistema

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

Ahora veremos el caso general. Una **matriz tridiagonal** tiene la siguiente forma:

$$\mathbf{A} = \begin{pmatrix} a_1 & c_1 & & & & & \\ b_2 & a_2 & c_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & & b_n & a_n \end{pmatrix}$$

Todos los elementos son nulos excepto las diagonales $k = -1, 0, 1$. Tales matrices pueden encontrarse en la solución de problemas de valores en la frontera para ecuaciones diferenciales ordinarias, las cuales son diagonal dominantes, es decir, $|a_k| \geq |b_k| + |c_k|$.

Se tiene entonces que

$$\mathbf{A} = \mathbf{LU} = \begin{pmatrix} 1 & & & & & & \\ \beta_2 & 1 & & & & & \\ & \ddots & \ddots & & & & \\ & & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ & & & & \beta_{n-1} & 1 & \\ & & & & & \beta_n & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 & c_1 & & & & & \\ & \alpha_2 & c_2 & & & & \\ & & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ & & & & \ddots & & \\ & & & & & \alpha_{n-1} & c_{n-1} \\ & & & & & & \alpha_n \end{pmatrix}$$

en donde

$$\begin{aligned} \alpha_1 &= a_1, \\ \beta_k &= \frac{b_k}{\alpha_{k-1}}, \quad (k = 2, 3, ..., n) \\ \alpha_k &= a_k - \beta_k c_{k-1}, \quad (k = 2, 3, ..., n) \end{aligned}$$

Al resolver $\mathbf{Ax} = \mathbf{v}$ será equivalente a resolver $\begin{cases} \mathbf{Ly} = \mathbf{v} \\ \mathbf{Ux} = \mathbf{y} \end{cases}$

En este caso, $\mathbf{Ly} = \mathbf{v}$ implica

$$\begin{cases} y_1 = v_1, \\ y_k = v_k - \beta_k y_{k-1}, \quad (k = 2, 3, ..., n) \end{cases}$$

y finalmente, $\mathbf{Ux} = \mathbf{y}$ implica

$$\begin{cases} x_n = \dfrac{y_n}{\alpha_n}, \\ \\ x_k = \dfrac{y_k - x_{k+1}c_k}{\alpha_k}, \quad (k = 2, 3, ..., n) \end{cases}$$

Utilice la información anterior para calcular las matrices $\mathbf{L}$ y $\mathbf{U}$ y a su vez calcular los vectores $\mathbf{y}$ y $\mathbf{x}$, y compare con su solución del sistema del inicio.

Capítulo 5

Métodos iterativos

Dado el sistema $A\boldsymbol{x} = \boldsymbol{b}$ con $A \in \mathbb{R}^{n\times n}$ y $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$. Resolverlo por factorización es con frecuencia muy costoso para n grande. Por esto vamos a expresar el sistema en la siguiente forma:

$$(A - B)\boldsymbol{x} = -B\boldsymbol{x} + \boldsymbol{b}$$

donde B se escoge de tal forma que $A-B$ sea no singular y el sistema $(A-B)\boldsymbol{x} = \boldsymbol{y}$ sea fácilmente resuelto para cualquier y. Un esquema iterativo simple comienza con un estimado de la solución $\boldsymbol{x^{(0)}} \in \mathbb{R}^n$, el cual puede ser arbitrario, y generar la sucesión $\boldsymbol{x^{(k)}}$, $k = 1, 2, ...$, resolviendo

$$(A - B)\boldsymbol{x^{(k+1)}} = -B\boldsymbol{x^{(k)}} + \boldsymbol{b} \qquad (\star)$$

Esta técnica es conocida como *splitting*. Si la sucesión converge a un límite, $\lim_{k\to\infty} \boldsymbol{x^{(k)}} = \hat{\boldsymbol{x}}$, entonces al tomar límites en $(\star)$ se tiene que

$$(A - B)\hat{\boldsymbol{x}} = -B\hat{\boldsymbol{x}} + \boldsymbol{b}$$

Por tanto $\hat{\boldsymbol{x}}$ es una solución de $(\star)$. ¿Cuáles son las condiciones necesarias y suficientes para la convergencia? Supongamos que A es no singular y por lo tanto el sistema tiene una única solución $\boldsymbol{x}^*$. Como $\boldsymbol{x}^*$ es solución de $Ax = b$ también satisface $(A - b)x^* = -Bx^* + b$. Sustrayendo esto último de $(\star)$ se tiene

$$(A - b)(x^{(k+1)} - x^*) = -B(x^{(k)} - x^*)$$

Denotemos $\boldsymbol{e^{(k)}} := \boldsymbol{x^{(k)}} - \boldsymbol{x}^*$, el cual es el error en la $k-$ésima iteración. Como $A - B$ es no singular, podemos escribir

$$\boldsymbol{e^{(k+1)}} = -(A - B)^{-1}B\boldsymbol{e^{(k)}}$$

La matriz $H := -(A - B)^{-1}B$ es conocida como *matriz de iteración*.

En la práctica H no se calcula. Se analizan sus propiedades teoréticamente en order de deter-

minar cuando se tiene convergencia o no.

Definición 5.1 (Radio espectral)**.** Sean $\lambda_1, ..., \lambda_n$ valores propios de una matriz H. Entonces su *radio espectral* $\rho(H)$ se define como

$$\rho(H) := \max_{i=1:n}(|\lambda_i|)$$

Recordemos que aunque H sea real sus valores propios pueden ser complejos.

Teorema 5.1.

$$\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^* \quad \textit{para todo } \boldsymbol{x}^{(0)} \quad \textit{si y solo si } \rho(H) < 1.$$

5.1. Normas de matrices. Condicionamiento

Definición 5.2 (Norma matricial)**.** Se dice que una función $|||\cdot||| : \mathbb{R}^{m\times n} \to \mathbb{R}$ es una *norma matricial* si satisface las siguientes propiedades:

1. $|||A||| \geq 0$ para todo A. También $|||A||| = 0 \Leftrightarrow A = 0$;
2. $|||\lambda A||| = |\lambda| \cdot |||A|||$ para todo $\lambda \in \mathbb{R}$;
3. $|||A + B||| \leq |||A||| + |||B|||$.

La norma es *consistente* si $|||AB||| \leq |||A|||\,|||B|||$ suponiendo bien definido AB.

Definición 5.3 (Norma de Frobenius)**.** Sea $A \in \mathbb{R}^{m\times n}$ una matriz. Entonces la norma de Frobenius se define como

$$|||A|||_F := \left(\sum_{j=1}^{n}\sum_{i=1}^{m} |a_{ij}|^2\right)^{\frac{1}{2}}$$

Se puede probar que la norma de Frobenius es consistente. Además se cumple que

- $|||I_n|||_F = \sqrt{n}$;
- $|||A|||_F^2 = \mathrm{tr}(A^T A)$

Ejercicio 5.1.1. Programe la norma de Frobenius. Compare con la norma, `norm(A, 'fro')`.

Teorema 5.2. *Sea $|||\cdot|||$ una norma matricial en $\mathbb{R}^{n\times n}$. Entonces para todo $A \in \mathbb{R}^{m\times n}$ se tiene que*

- $|||A^k||| \leq |||A|||^k$, *para todo k natural;*
- $\rho(A) \leq |||A|||$.

Teorema 5.3. *Sea $\|\cdot\|$ una norma vectorial en $\mathbb{R}^n$. La función $|||\cdot||| : \mathbb{R}^{n\times n} \to \mathbb{R}$,*

$$|||A||| := \max_{x\in\mathbb{R}^n-\{\mathbf{0}\}} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

es una norma matricial consistente, la cual es inducida por la norma vectorial dada.

Además se verifica que

- $\|Ax\| \leq |||A||| \cdot \|x\|$, para todo $x \in \mathbb{R}^n$;
- $|||I_n||| = 1$.

Algunas normas matriciales son:

- $|||A|||_\infty = \max\limits_{i=1}^{n} \sum\limits_{j=1}^{n} |a_{ij}|$
- $|||A|||_1 = \max\limits_{j=1}^{n} \sum\limits_{i=1}^{n} |a_{ij}|$
- $|||A|||_2 = \max\limits_{i=1}^{n} \sqrt{\lambda_i}$, donde los valores propios son los de la matriz A^TA.
- $|||A||| = \max_{i,j} |a_{ij}| \quad \longrightarrow$ no es consistente.

Definición 5.4 (Número de condicionamiento)**.** El número de condicionamiento de una matriz A no singular se define como

$$\kappa(A) := |||A||| \cdot |||A^{-1}|||$$

con $|||\cdot|||$ consistente. Se dice que la matriz A está **mal condicionada** si $\kappa(A) \gg 1$. En caso contrario se dice que está **bien condicionada**.

El sistema $Ax = b$ está mal condicionado si A es mal condicionada, y está bien condicionado si A es bien condicionada. △

Los métodos a continuación son de *splitting* y pueden ser usados cuando A no tenga ningún cero en la diagonal, y se expresa $A = L + D + U$ donde D es la matriz diagonal cuya diagonal es la diagonal principal de A, y L, U son matrices triangular inferior y superior respectivamente.

5.2. Método de Jacobi

Se escoge $A - B = D$ la parte diagonal de A, o en otras palabras, $B = L + U$. La iteración está dada por

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

esto es,

Método iterativo de Jacobi

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

La iteración de JACOBI es en realidad una iteración de punto fijo, en la cual se despejan los términos de la diagonal de A:

$$\begin{cases} x_1 &= \dfrac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - ... - a_{1n}x_n) \\ \vdots & \vdots \\ x_n &= \dfrac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - ... - a_{n-1,n-1}x_{n-1}) \end{cases}$$

Es decir,

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right), \qquad \text{para } i = 1, 2, ..., n$$

Como criterio de parada se tiene que $\|\boldsymbol{b} - A\boldsymbol{x}^{(k)}\| < \|\boldsymbol{b}\| \cdot \texttt{tol}$, donde se escoge $\|\cdot\|$ a convenir.

Método de Jacobi

```
function [x,k] = Jacobi(A,b,x0,tol,itermax)
[n,m]=size(A);
[~,s]=size(b);
[~,r]=size(x0);
if n~=m||s~=1||r~=1
    disp("La matriz A debe ser cuadrada y tanto b como x0 deben
    ser vectores columna")
    return
end
x = x0;
k = 0;
tol = tol*norm(b, Inf);
err = tol+1;
while k < itermax && err > tol
```

```
15      for i = 1:n
16          suma = 0;
17          for j = 1:n
18              if i~=j
19                  suma = suma + A(i,j)*x0(j);
20              end
21          end
22          x(i) = (b(i) - suma)/A(i,i);
23      end
24      err = norm(b - A*x, Inf);
25      k=k+1;
26      x0=x;
27  end
28  k = k-1;
29  end
```

5.3. Método de Gauss–Seidel

En este método se tienen dos formas:

- **Gauss–Seidel hacia adelante:** Se toma $A - B = L + D$, la parte tringular inferior de la matriz A, con lo cual $B = U$. Entonces la sucesión $\boldsymbol{x}^{(k)}$ con $k = 1, ...,$ es generada por

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

es decir,

Método iterativo de Gauss-Seidel hacia adelante

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

La iteración de Gauss–Seidel hacia adelante se expresa en el sistema como

$$\begin{cases} a_{11}x_1^{(k+1)} & = & b_1 - a_{12}x_2^{(k)} - ... - a_{1n}x_n^{(k)} \\ a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} & = & b_2 - a_{23}x_3^{(k)} - ... - a_{2n}x_n^{(k)} \\ \vdots & & \vdots \\ a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + ... + a_{nn}x_n^{(k+1)} & = & b_n \end{cases}$$

Es decir,

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \qquad \text{para } i = 1, 2, ..., n$$

- **Gauss–Seidel hacia atrás:** Se toma $A - B = D + U$, la parte tringular superior de la matriz A, con lo cual $B = L$. Entonces la sucesión $\boldsymbol{x^{(k)}}$ con $k = 1, ...,$ es generada por

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

es decir,

Método iterativo de Gauss-Seidel hacia atrás

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

Las componentes se calculan con

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k+1)} \right), \qquad \text{para } i = n, n-1, ..., 2, 1$$

Gauss–Seidel hacia adelante

```
function [x,k] = gauss_seidelF(A,b,x0,tol,itermax)
[n,m] = size(A);
[~,s] = size(b);
[~,r] = size(x0);
if m~=n || s~=1||r~=1
    disp('Dimensiones incorrectas')
    return
end
x = x0;
k = 0;
tol = tol*norm(b, Inf);
err = tol+1;
```

```
while k<itermax && err>tol
    for i = 1:n
        suma1=0;
        suma2=0;
        for j = 1:i-1
            suma1 = suma1 + A(i,j)*x(j);
        end
        for j = i+1:n
            suma2 = suma2 +A(i,j)*x0(j);
        end
        x(i) = (b(i)-suma1-suma2)/A(i,i);
    end
    k=k+1;
    err = norm(b - A*x, Inf);
    x0=x;
end
k=k-1;
end
```

Gauss–Seidel hacia atrás

```
function [x,k] = gauss_seidelB(A,b,x0,tol,itermax)
[n,m] = size(A);
[~,s] = size(b);
[~,r] = size(x0);
if m~=n || s~=1 || r~=1
    disp('Dimensiones incorrectas')
    return
end
x = x0;
k = 0;
tol = tol*norm(b, Inf);
err = tol+1;
while k<itermax && err>tol
    for i = n:-1:1
        suma1=0;
```

```
16          suma2=0;
17          for j = 1:i-1
18              suma1 = suma1 + A(i,j)*x0(j);
19          end
20          for j = i+1:n
21              suma2 = suma2 +A(i,j)*x(j);
22          end
23          x(i) = (b(i)-suma1-suma2)/A(i,i);
24      end
25      k=k+1;
26      err = norm(b - A*x, Inf);
27      x0=x;
28  end
29  k=k-1;
30  end
```

5.3.1. Algunas consideraciones

La sucesión $\boldsymbol{x}^{(k)}$ converge si el radio espectral de las matrices de iteración $H_J = -D^{-1}(L+U)$ (Jacobi), $H_{GSF} = -(L+D)^{-1}U$ (Gauss-Seidel hacia adelante) es menor que 1.

Definición 5.5. Una matriz A es denominada **estrictamente diagonal dominante por filas** si

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq 1}}^{n} |a_{ij}|, \qquad i = 1, ..., n.$$

Teorema 5.4 (Householder–John). *Si A y B son matrices reales tales que A y $A - B - B^T$ son simétricas y definidas positivas, entonces el radio espectral de $H = -(A-B)^{-1}B$ es estrictamente menor que* 1.

Teorema 5.5 (Gerschgorin). *Todos los valores propior de una matriz de $A \in \mathbb{C}^{n\times n}$ están contenidos en la unión de los discos de* Gerschgorin Γ_i, $i = 1, ..., n$, *definidos en $\mathbb{C}$ como*

$$\Gamma_i := \left\{ z \in \mathbb{C} : \ |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \right\}$$

Se sigue de este teorema que las matrices estrictamente diagonal dominantes son no singulares puesto que ninguno de los discos de Gerschgorin contienen a cero.

Teorema 5.6. *Si A es estrictamente diagonal dominante, entonces las sucesiones que salen de los métodos de Jacobi y Gauss-Seidel convergen.*

Corolario 5.7. *A continuación se tienen los siguientes resultados:*

1) Si A es simétrica y definida positiva, entonces el método de Gauss-Seidel converge.

2) Si tanto A como $2D - A$ son simétricas y definidas positivas, entonces el método de Jacobi converge.

Ejemplo 5.1. Considere el sistema

$$\begin{cases} 5x_1 - 4x_2 &= -1 \\ 4x_1 + 5x_2 &= 2 \end{cases}$$

a) Calcule el número de condición (en norma infinito matricial) de la matriz del sistema.

Solución:

$$A = \begin{pmatrix} 5 & -4 \\ 4 & 5 \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{41}\begin{pmatrix} 5 & 4 \\ -4 & 5 \end{pmatrix} \Longrightarrow \left\| \begin{pmatrix} 5 & -4 \\ 4 & 5 \end{pmatrix} \right\|_\infty = 9, \left\| \frac{1}{41}\begin{pmatrix} 5 & 4 \\ -4 & 5 \end{pmatrix} \right\|_\infty = \frac{9}{41}.$$

Entonces,

$$\kappa_\infty(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty = \frac{81}{41} \approx 1.9756$$

b) Resuelva el sistema de manera exacta. Luego utilice el método de Gauss-Seidel hacia adelante para calcular $\vec{\boldsymbol{x}}_3$, tomando $\vec{\boldsymbol{x}}_0 = (0,1)^t$.

Solución:

Multiplicando la primera ecuación por -4 se tiene que $-20x_1 + 16x_2 = 4$, y la segunda ecuación por 5 se tiene $20x_1 + 25x_2 = 10$, al sumarlas se tiene la ecuación $41x_2 = 14 \Rightarrow x_2 = \frac{14}{41} \Rightarrow x_1 = \frac{3}{41}$. Tenemos así que el vector solución es

$$\vec{\boldsymbol{x}} = \begin{pmatrix} \frac{3}{41} \\ \frac{14}{41} \end{pmatrix} \approx \begin{pmatrix} 0.07317 \\ 0.34146 \end{pmatrix}$$

El método de Gauss-Seidel hacia adelante para el sistema dado es el siguiente:

$$\begin{cases} \left(x_1^{(0)}, x_2^{(0)}\right)^t &= (0,1)^t \\ \left(x_1^{(k+1)}, x_2^{(k+1)}\right) &= \left(\frac{1}{5}(-1 + 4x_2^{(k)}), \frac{1}{5}(2 - 4x_1^{(k+1)})\right)^t, \quad k = 0, 1, \ldots \end{cases}$$

En nuestro caso calcularemos para $k = 0, 1, 2$:

- $\left(x_1^{(1)}, x_2^{(1)}\right)^t = \left(\frac{1}{5}(-1 + 4x_2^{(0)}), \frac{1}{5}(2 - 4x_1^{(1)})\right)^t = \left(\frac{3}{5}, \frac{1}{5}(2 - 4 \cdot \frac{3}{5})\right)^t = \left(\frac{3}{5}, \frac{-2}{25}\right)^t = (0.6, -0.08)^t$

- $\left(x_1^{(2)},\, x_2^{(2)}\right)^t = \left(\frac{1}{5}(-1+4x_2^{(1)}),\, \frac{1}{5}(2-4x_1^{(2)})\right)^t = \left(\frac{-33}{125},\, \frac{382}{625}\right)^t = (-0.2640,\, 0.6112)^t$
- $\left(x_1^{(3)},\, x_2^{(3)}\right)^t = \left(\frac{1}{5}(-1+4x_2^{(2)}),\, \frac{1}{5}(2-4x_1^{(3)})\right)^t = \left(\frac{903}{3125},\, \frac{2638}{15625}\right)^t = (0.2890,\, 0.1688)^t$

c) Determine el error relativo en norma ∞ al aproximar el vector solución con $\vec{\boldsymbol{x}}_3$.

Solución:

$$\delta = \frac{\|(0.07317,\, 0.34146)^t - (0.2890,\, 0.1688)^t\|_\infty}{\|(0.07317,\, 0.34146)^t\|_\infty} \approx 0.6320$$

$\diamond$

5.4. Relajación

La eficiencia de los métodos *splitting* puede ser mejorado por **relajación**. Así que antes de iterar $(A-B)\boldsymbol{x}^{(k+1)} = -B\boldsymbol{x}^{(k)} + \boldsymbol{b}$ primero calculamos $(A-B)\tilde{\boldsymbol{x}}^{(k+1)} = -B\boldsymbol{x}^{(k)} + \boldsymbol{b}$ como un valor intermedio y sea

$$\boldsymbol{x}^{(k+1)} = \omega\tilde{\boldsymbol{x}}^{(k+1)} + (1-\omega)\boldsymbol{x}^{(k)}, \qquad k = 0, 1, ...$$

en donde $\omega \in \mathbb{R}$ es llamado **parámetro de relajación**. De hecho $\omega = 1$ corresponde al método original sin relajación. El parámetro es escogido tal que el radio espectral del método relajado es pequeño. A radio espectral más pequeño más rápido converge la iteración. Sea $\boldsymbol{c} = (A-B)^{-1}\boldsymbol{b}$, la matriz de la iteración relajada H_ω se puede deducir de

$$\boldsymbol{x}^{(k+1)} = \omega\tilde{\boldsymbol{x}}^{(k+1)} + (1-\omega)\boldsymbol{x}^{(k)} = \omega H\boldsymbol{x}^{(k)} + (1-\omega)\boldsymbol{x}^{(k)} + \omega\boldsymbol{c}$$

como

$$H_\omega = \omega H + (1-\omega)I$$

Se sigue que los valores propios de H_ω y H están relacionados de la forma $\lambda_\omega = \omega\lambda + (1-\omega)\lambda$. La mejor escogencia de ω se puede obtener minimizando

$$\text{máx}\{|\omega\lambda_i + (1-\omega)|,\, i = 1, ..., n\}$$

donde $\lambda_1, ..., \lambda_n$ son los valores propios de H. Sin embargo, los valores propios de H con frecuencia son desconocidos, pero a veces se tiene información (por ejemplo, derivado del teorema de Gerschgorin), la cual hace posible la escogencia de un buen –pero no siempre óptimo– ω.

5.4.1. Métodos de Sobrerrelajación SOR

Estos consisten en utilizar relajación en los métodos de Gauss-Seidel.

SOR hacia adelante

La iteración

$$\begin{cases} \mathbf{x}^{(0)},\ \omega \in \mathbb{R} - \{0\} \ \text{ dados,} \\ \text{para } k = 0, 1, 2, ..., \ \text{hacer:} \\ \hat{x}_i^{(k+1)} = \dfrac{1}{a_{ii}} \left(b_i - \displaystyle\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \\ x_i^{(k+1)} = (1-\omega) x_i^{(k)} + \omega \hat{x}_i^{(k+1)}, \quad \forall i = 1, 2, ..., n \end{cases}$$

se conoce como **iteración de sobrerrelajación hacia adelante**.

SOR hacia atrás

La iteración

$$\begin{cases} \mathbf{x}^{(0)},\ \omega \in \mathbb{R} - \{0\} \ \text{ dados,} \\ \text{para } k = 0, 1, 2, ..., \ \text{hacer:} \\ \hat{x}_i^{(k+1)} = \dfrac{1}{a_{ii}} \left(b_i - \displaystyle\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k+1)} \right) \\ x_i^{(k+1)} = (1-\omega) x_i^{(k)} + \omega \hat{x}_i^{(k+1)}, \quad \forall i = n, n-1, ..., 1 \end{cases}$$

se conoce como **iteración de sobrerrelajación hacia atrás**.

Para el método de Jacobi se puede definir el método de relajación como

$$\mathbf{x}^{(k+1)} = (1-\omega)\mathbf{x}^{(k)} + \omega \hat{\mathbf{x}}^{(k+1)}$$

con

$$\hat{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{D}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})$$

Teorema 5.8 (Ostrowski-Reich). *Si $\mathbf{A} \in \mathbb{R}^{n\times n}$ es simétrica definida positiva y $\omega \in]0, 2[$, entonces la iteración de SOR converge hacia la solución de $\mathbf{A}\mathbf{x} = \mathbf{b}$ para todo $\mathbf{b} \in \mathbb{R}^n$ y para todo $\mathbf{x}^{(0)} \in \mathbb{R}^n$.*

Definición 5.6 (Matriz consistentemente ordenada). Sea $A \in \mathbb{R}^{n\times n}$ tal que $A = D + L + U$ y $a_{ii} \neq 0$ para todo i. Se dice que A es **consistentemente ordenada** si los valores propios de la matriz

$$\mathscr{J}(\alpha) := \alpha D^{-1} \left(L + \frac{1}{\alpha} U \right), \qquad \alpha \in \mathbb{C} - \{0\}$$

son independientes de α, es decir, si se cumple que $\sigma[\mathscr{J}(\alpha)] = \sigma[\mathscr{J}(1)]$ para todo $\alpha \in \mathbb{C} - \{0\}$.

$\omega_{\texttt{opt}}$ para matrices consistentemente ordenadas

Consideremos el sistema lineal $\mathbf{Ax} = \mathbf{b}$ para el cual se desea aproximar la solución por el método de SOR. Entonces es necesario definir un valor de ω, donde para matrices consistentemente ordenadas se tiene un valor óptimo, definido como

$$\omega_{\texttt{opt}} := \frac{2}{1 + \sqrt{1 - [\rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})]^2}}$$

en donde $D = \texttt{diag}(\texttt{diag}(\mathbf{A}))$

5.5. Método del Gradiente

Consideremos A simétrica definida positiva y el sistema $Ax = b$. La forma cuadrática

$$F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{t}} A\boldsymbol{x} - \boldsymbol{x}^{\mathsf{t}}\boldsymbol{b}, \qquad \boldsymbol{x} \in \mathbb{R}^n$$

es una función multivariable, la cual puede expresarse como

$$F(x_1, ..., x_n) = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n} b_i x_i$$

Para calcular los extremos de dicha función se hace $\nabla F(\boldsymbol{x}) = 0$. Usando la simetría de A se tiene que

$$\frac{d}{dx_k}F = \sum_{j=1}^{n} a_{kj}x_j - b_k$$

es decir, $\nabla F(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = \mathbf{0}$. Entonces encontrar extremos de F es equivalente a resolver $Ax = b$. En este caso el extremo es un mínimo puesto que A es definida positiva.

Sea $\boldsymbol{\xi}$ la solución. El mínimo de F no cambia si se agrega la constante $\frac{1}{2}\boldsymbol{\xi}^{\mathsf{t}} A\boldsymbol{\xi}$. Usando que $\boldsymbol{b} = A\boldsymbol{\xi}$ vemos que es equivalente a minimizar

$$\frac{1}{2}\boldsymbol{x}^{\mathsf{t}} A\boldsymbol{x} - \boldsymbol{x}^{\mathsf{t}} A\boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}^{\mathsf{t}} A\boldsymbol{\xi} = \frac{1}{2}(\boldsymbol{\xi} - \boldsymbol{x})^{\mathsf{t}} A(\boldsymbol{\xi} - \boldsymbol{x})$$

La última expresión puede denotarse como $\|\boldsymbol{x}\|_A := \sqrt{\boldsymbol{x}^{\mathsf{t}} A\boldsymbol{x}}$ lo cual está bien definido pues A es definida positiva. Entonces lo que se está minimizando es $\|\boldsymbol{\xi} - \boldsymbol{x}\|_A$. Cada iteración construye una aproximación $\boldsymbol{x}^{(k)}$ la cual está cercana a $\boldsymbol{\xi}$ en la norma definida por A. Como esto es equivalente a minimizar F, la sucesión construida deberá satisfacer la condición

$$F(\boldsymbol{x}^{(k+1)}) < F(\boldsymbol{x}^{(k)})$$

Esto es, el valor de F en la nueva iteración deberá ser menor que el valor previo, puesto que se está buscando el mínimo. Precisamente los métodos de JACOBI y de GAUSS–SEIDEL hacen esto.

Generalmente los métodos de descenso tienen la siguiente forma:

1. Iniciar con cualquier vector inicial $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$.

2. Para cualquier $k = 0, 1, 2, ...$, el cálculo para si la norma del gradiente $\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\| = \|\nabla F(\boldsymbol{x}^{(k)})\|$ es aceptablemente pequeño.

3. De lo contrario una *dirección de búsqueda* $\boldsymbol{d}^{(k)}$ se determina de tal manera que se satisfaga la *condición de descenso*
$$\left[\frac{d}{d\omega}F(\boldsymbol{x}^{(k)} + \omega\boldsymbol{d}^{(k)})\right]_{\omega=0} < 0$$
En otras palabras, si se camina en la dirección de búsqueda, los valores de F se hacen más pequeños.

4. El valor $\omega^{(k)} > 0$ el cual minimiza $F(\boldsymbol{x}^{(k)} + \omega\boldsymbol{d}^{(k)})$ es calculado y la nueva aproximación es
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \omega^{(k)}\boldsymbol{d}^{(k)}$$
Regresar luego al paso 2.

Después de unos cálculos al usar la definición de F y al hacer $\boldsymbol{g}^{(k)} = A\boldsymbol{x}^{(k)} - \boldsymbol{b}$ (el cual denota el gradiente $\nabla F(\boldsymbol{x}^{(k)})$, diferenciando con respecto a ω e igualando a cero se tiene que
$$\omega^{(k)} = -\frac{(\boldsymbol{d}^{(k)})^{\mathtt{t}}\boldsymbol{g}^{(k)}}{(\boldsymbol{d}^{(k)})^{\mathtt{t}}A\boldsymbol{d}^{(k)}}$$

5.6. Método del Gradiente Conjugado

Hemos visto que la postitividad definida de A puede ser utilizada para definir una norma. Esto puede extenderse para obtener un concepto similar al de ortogonalidad.

Definición 5.7. Los vectores $\boldsymbol{u}$ y $\boldsymbol{v}$ son conjugados con respecto a una matriz definida positiva A si son no nulos y satisfacen que $\boldsymbol{u}^{\mathsf{t}} A\boldsymbol{v} = 0$. Se dice entonces que $\boldsymbol{u}$ y $\boldsymbol{v}$ son $A-$ortogonales. Se usa la notación

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_A := \boldsymbol{u}^{\mathsf{t}} A\boldsymbol{v} = \langle \boldsymbol{u}, A\boldsymbol{v} \rangle = \langle A\boldsymbol{u}, \boldsymbol{v} \rangle$$

Teorema 5.9. *Sea $\{\boldsymbol{d}_0, ..., \boldsymbol{d}_{n-1}\}$ una base $A-$ortogonal y $\boldsymbol{x}_0 \in \mathbb{R}^n$ cualquiera. Entonces la sucesión*

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \omega_k \boldsymbol{d}^{(k)} \quad \text{con } \omega_k = \frac{\langle \boldsymbol{d}^{(k)}, \boldsymbol{r}^{(k)} \rangle}{\langle A\boldsymbol{d}^{(k)}, \boldsymbol{d}^{(k)} \rangle}, \quad \boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)}$$

determina la solución $\boldsymbol{\xi}$ del sistema lineal $A\boldsymbol{x} = \boldsymbol{b}$ en a lo sumo n pasos.

Lema 5.10. *Sea A una matriz simétrica definida positiva y sea $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$. Si existe un polinomio $p(t)$ de grado k tal que $p(0) = 1$ y $|p(\lambda)| \leq r$ para todo valor propio λ de A, entonces el vector $\boldsymbol{x}^{(k)}$ del método del gradiente conjugado satisface que*

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{\xi}\|_A \leq r \|\boldsymbol{x}^{(0)} - \boldsymbol{\xi}\|_A$$

donde $\boldsymbol{\xi}$ es la solución de $A\boldsymbol{x} = \boldsymbol{b}$.

5.7. Método de Newton multivariable

Conseideremos $\mathbf{g} : D \subset \mathbb{R}^n \to \mathbb{R}^n$, función continua en varias variables, donde D es un cerrado de $\mathbb{R}^n$. En concreto,

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Vamos a buscar puntos fijos de $\mathbf{g}$. Dado $\mathbf{x}^{(0)} \in D$, sea la sucesión

$$\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)}), \quad k = 0, 1, ...$$

La existencia de un único punto fijo en D se garantiza si $\mathbf{g}$ es una contracción, es decir, si existe $L \in]0, 1[$ tal que

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\|_\infty \leq L\|\mathbf{x} - \mathbf{y}\|_\infty, \qquad \text{para todo } \mathbf{x}, \mathbf{y} \in D$$

Se pueden obtener estimados locales si se conoce la norma del jacobiano $J_{\mathbf{g}}(\mathbf{x}) \in \mathbb{R}^{n\times n}$,

$$J_{\mathbf{g}}(\mathbf{x}) := \left(\frac{\partial g_i(\mathbf{x})}{\partial x_j}\right)_{1\le i,j\le n}$$

Teorema 5.11. *Suponga que* $\mathbf{g}$ *es continua en un cerrado* $D \subset \mathbb{R}^n$ *y sea* $\boldsymbol{\xi} \in D$ *un punto fijo de* $\mathbf{g}$. *Suponga que las primeras derivadas parciales son continuas en un vecindario abierto* $N(\boldsymbol{\xi}) \subset D$ *con*

$$\|J_{\mathbf{g}}(\boldsymbol{\xi})\|_\infty < 1$$

Entonces existe $\varepsilon > 0$ *tal que* $\mathbf{g}\left(\overline{B}_\varepsilon(\boldsymbol{\xi})\right) \subset \overline{B}_\varepsilon(\boldsymbol{\xi})$. *Además la sucesión* $\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)})$ *converge a* $\boldsymbol{\xi}$ *para todo* $\mathbf{x}^{(0)} \in \overline{B}_\varepsilon(\boldsymbol{\xi})$.

Iteración de Newton

Considere la iteración

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J_{\mathbf{g}}(\mathbf{x}^{(k)})\right)^{-1}\mathbf{g}(\mathbf{x}^{(k)}), \qquad k = 0, 1, ...$$

se denomina *método de Newton en* $\mathbb{R}^n$.

Como consejo, para no tener que calcular la inversa del jacobiano se resuelve en cada iteración el sistema lineal con incógnita $\mathbf{y}$:

$$J_{\mathbf{g}}(\mathbf{x}^{(k)})\mathbf{y} = \mathbf{g}(\mathbf{x}^{(k)})$$

Como criterio de parada puede utilizarse $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty < \texttt{tol}$.

Ejemplo 5.2. Dos circunferencias de radio 1 con centros en $(0, 1)$ y $(2, 1)$ son tangentes en el punto $\boldsymbol{P} = (1, 1)$.

a) Utilice el método de Newton multivariable, con 3 iteraciones, para aproximarse al punto de intersección de las circunferencias tomando como vector inicial $\vec{\boldsymbol{P}}_0 = (1, 0)^t$.

Solución:

El sistema no lineal a plantear es el siguiente:

$$\begin{cases} x^2 + (y-1)^2 &= 1 \\ (x-2)^2 + (y-1)^2 &= 1 \end{cases}$$

Luego $F\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x^2 + (y-1)^2 - 1 \\ (x-2)^2 + (y-1)^2 - 1 \end{pmatrix} \Longrightarrow J_F\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x & 2y-2 \\ 2x-4 & 2y-2 \end{pmatrix}$

$$\left[J_F\begin{pmatrix} x \\ y \end{pmatrix}\right]^{-1} = \frac{1}{8(y-1)}\begin{pmatrix} 2y-2 & 2-2y \\ 4-2x & 2x \end{pmatrix}$$

Por lo tanto, la sucesión de Newton para este problema es:

$$\begin{cases} (x_0, y_0)^t &= (0,1)^t \\ \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} &= \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \dfrac{1}{8(y_k-1)} \begin{pmatrix} 2y_k-2 & 2-2y_k \\ 4-2x_k & 2x_k \end{pmatrix} \cdot \begin{pmatrix} x_k^2+(y_k-1)^2-1 \\ (x_k-2)^2+(y_k-1)^2-1 \end{pmatrix} \end{cases}$$

Con base en esto se tiene que

$$(x_1, y_1)^t = (1,\, 0.5)^t, \quad (x_2, y_2)^t = (1,\, 0.75)^t, \quad (x_3, y_3)^t = (1,\, 0.875)^t$$

b) Calcule el error relativo $\dfrac{\|\vec{\boldsymbol{P}} - \vec{\boldsymbol{P}}_3\|_\infty}{\|\vec{\boldsymbol{P}}\|_\infty}$.

Solución:

$$\frac{\|\vec{\boldsymbol{P}} - \vec{\boldsymbol{P}}_3\|_\infty}{\|\vec{\boldsymbol{P}}\|_\infty} = \frac{\|(1,1)^t - (1,0.875)^t\|_\infty}{\|(1,1)^t\|_\infty} = 0.125$$

◇

Ejemplo 5.3. Considere el sistema no lineal

$$\begin{cases} 3x_1^2 - x_2^2 &= 0 \\ 3x_1x_2 - x_1^3 &= 1 \end{cases}$$

Utilice el método de Newton con $\vec{\boldsymbol{x}}_0 = (1,1)^t$ hasta obtener una aproximación a una solución que cumpla que $\|\vec{\boldsymbol{x}}_k - \vec{\boldsymbol{x}}_{k+1}\|_\infty < 10^{-2}$.

Solución:

$$F\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3x_1^2 - x_2^2 \\ 3x_1x_2 - x_1^3 - 1 \end{pmatrix} \Longrightarrow J_F\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 6x_1 & -2x_2 \\ 3x_2 - 3x_1^2 & 3x_1 \end{pmatrix}$$

$$\left[J_F\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right]^{-1} = \frac{1}{18x_1^2 + 6x_2^2 - 6x_1^2x_2} \begin{pmatrix} 3x_1 & 2x_2 \\ 3x_1^2 - 3x_2 & 6x_1 \end{pmatrix}$$

Por lo tanto, la sucesión de Newton para este problema es:

$$\begin{cases} (x_1^{(0)}, x_2^{(0)})^t &= (1,1)^t \\ \begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} &= \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} - \dfrac{1}{18(x_1^{(k)})^2 + 6(x_2^{(k)})^2 - 6(x_1^{(k)})^2 x_2^{(k)}} \begin{pmatrix} 3x_1^{(k)} & 2x_2^{(k)} \\ 3(x_1^{(k)})^2 - 3x_2^{(k)} & 6x_1^{(k)} \end{pmatrix} \cdot \begin{pmatrix} 3(x_1^{(k)})^2 - (x_2^{(k)})^2 \\ 3x_1^{(k)}x_2^{(k)} - (x_1^{(k)})^3 - 1 \end{pmatrix} \end{cases}$$

El cómputo es a mano, pero acá tenemos con MATLAB una manera de conseguir lo que buscamos. Primero creamos la función:

```
function X = MultiNewton(F, JF, X, maxit)
% Inputs: F campo vectorial, JF matriz jacobiana de F
%         X vector inicial X0, maxit cantidad de iteraciones
% Outputs: Aproximacion a la solucion de F(X) = 0

Y = F(X);

for k = 1:maxit
    J = JF(X);
    Q = X-(J\Y);
    Z = F(Q);

    X = Q;
    Y = Z;
end
end
```

Luego ejecutamos las siguientes líneas:

```
f = @(X) [3.*X(1).^2 - X(2).^2 ; 3.*X(1).*X(2) - X(1).^3 - 1];

Jf = @(X) [6.*X(1), -2.*X(2); 3.*X(2) - 3.*X(1).^2, 3.*X(1)];

X0 = [1 1]';
X = MultiNewton(f, Jf, X0, 1);
norma = norm(X0 - X, "inf");
tol = 1e-2;
k = 1;

while norm(X0 - X, "inf") >= tol
    X0 = X;
    X = MultiNewton(f, Jf, X0, k);
    norma = norm(X0 - X, "inf");
```

```
    k = k + 1;
end
k
X
norma
```

Obteniendo los outputs:

```
k = 4
X = [0.4595; 0.7958]
norma = 3.7715e-08
```

5.8. Ejercicios

5.8.1. Normas matriciales. Condicionamiento

1. Calcule $\|\mathbf{A}\|_F$, $\|\mathbf{A}\|_1$, $\|\mathbf{A}\|_\infty$ para las matrices

$$\mathbf{A}_1 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}, \qquad \mathbf{A}_2 = \begin{pmatrix} 4 & -1 & 7 & 2 \\ -1 & 4 & 0 & 1 \\ -7 & 0 & 4 & -5 \\ 2 & 0 & 2 & 2 \end{pmatrix}, \qquad \mathbf{A}_3 = \begin{pmatrix} 1 & 2 & -3 & 4 & -2 \\ 4 & 11 & -10 & 19 & -7 \\ 4 & 2 & -13 & 8 & -5 \\ -1 & 4 & -2 & 12 & -14 \\ 3 & 12 & 13 & 26 & 13 \end{pmatrix}$$

2. Implemente M-funciones en `MATLAB` que calculen

$$\|\mathbf{A}\|_1 := \max_{j=1:n} \left(\sum_{i=1:m} |a_{ij}| \right), \qquad \|\mathbf{A}\|_\infty := \max_{i=1:m} \left(\sum_{j=1:n} |a_{ij}| \right)$$

Utilice la matriz $\mathbf{A} = \begin{pmatrix} 1 & 2 & -3 & 4 & -2 \\ 4 & 11 & -10 & 19 & -7 \\ 4 & 2 & -13 & 8 & -5 \\ -1 & 4 & -2 & 12 & -14 \\ 3 & 12 & 13 & 26 & 13 \end{pmatrix}$ y compare sus resultados con `norm(A,1)` y `norm(A, 'inf')`, las cuales son funciones para las normas matriciales en `MATLAB`.

3. Recordemos que el *número de condicionamiento* de una matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ es

$$\kappa(\mathbf{A}) := \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

Considere el sistema

$$\begin{cases} 7x + 10y &= 1 \\ 5x + 7y &= \frac{7}{10} \end{cases}$$

Utilice $\|\cdot\|_1$ matricial para calcular $\kappa(\mathbf{A})$ y verifique que el sistema está mal condicionado.

4. Sea $\mathbf{A} \in \mathbb{R}^{n\times n}$ y $\sigma(\mathbf{A}) = \{\lambda_k\}_{k=1:n}$ el conjunto de valores propios de la matriz $\mathbf{A}^\mathtt{t}\mathbf{A}$, entonces se tiene que

$$\|\mathbf{A}\|_2 = \max_{k=1:n} \sqrt{\lambda_k}$$

Considere la matriz $\mathbf{A} = \begin{pmatrix} 3 & -2 & 3 \\ 1 & 6 & -3 \\ 1 & 2 & 1 \end{pmatrix}$. Determine $\sigma(\mathbf{A})$, $\rho(\mathbf{A})$, $\|\mathbf{A}\|_2$.

5.8.2. Métodos iterativos

1. Utilizando los métodos de Jacobi, Gauss-Seidel hacia adelante y hacia atrás, aproxime la solución de los siguientes sistemas con $\mathtt{tol} = 10^{-3}$, con norma vectorial-∞ para acotar el error relativo y utilizando $\mathbf{x}^{(0)}$ como el vector nulo.

a) $\begin{cases} 3x_1 - x_2 + x_3 &= 1 \\ 3x_1 + 6_2 + 2x_3 &= 0 \\ 3x_1 + 3x_2 + 7x_3 &= 4 \end{cases}$

b) $\begin{cases} 4x_1 + x_2 - x_3 + x_4 &= -2 \\ x_1 + 4x_2 - x_3 - x_4 &= -1 \\ -x_1 - x_2 + 5x_3 + x_4 &= 0 \\ x_1 - x_2 + x_3 + 3x_4 &= 1 \end{cases}$

2. Calcule $\omega_{\mathtt{opt}}$ de la iteración SOR para la matriz del sistema

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

Tomando $\mathbf{x}^{(0)} = (0, 0, 0, 0)^\mathtt{t}$ calcule $\mathbf{x}^{(2)}$ utilizando el método de sobrerrelajación hacia adelante.

5.8.3. Gradiente conjugado

1. Escriba un programa `GradientC` para el método del gradiente conjugado. Luego utilícelo para aproximar la solución del sistema $\mathbf{Ax} = \mathbf{b}$ donde

$$\mathbf{A} = \frac{1}{12}\begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 5 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

2. Considere el sistema

$$\begin{cases} 2x - y &= -7 \\ -x + 2y &= 14 \end{cases}$$

   Compruebe que con el método del gradiente conjugado se converge a la solución exacta en no más de 2 iteraciones.

5.8.4. Newton multivariable

1. Considere el sistema no lineal

$$\begin{cases} x^2 - 10x + y^2 + 8 &= 0 \\ xy^2 + x - 10y + 8 &= 0 \end{cases}$$

   Calcule 5 iteraciones ($k = 4$) del método de Newton usando $\mathbf{x}^{(0)} = (3, -2)^{\mathtt{t}}$. Estime el error absoluto con la norma $\|\cdot\|_\infty$.

2. Utilice la iteración de Newton para aproximar la solución del sistema

$$\begin{cases} x^2 + y^2 + z^2 &= 1 \\ 2x^2 + y^2 - 4z &= 0 \\ 3x^2 - 4y + z^2 &= 0 \end{cases}$$

   usando $\mathbf{x}^{(0)} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^{\mathtt{t}}$.

5.8.5. Adicionales

1. Modifique el script de `MATLAB` visto en clase de SOR hacia adelante para obtener el método de SOR hacia atrás.

2. Sea $\mathbf{Ax} = \mathbf{b}$ y sean D = diag(diag(A)), `L = tril(A)-diag(diag(A))`,

`U = triu(A)-diag(diag(A))`

Se tiene que los métodos de iteración en su versión matricial son los siguientes:

- JACOBI: $\mathbf{x^{(k+1)}} = \mathbf{D^{-1}}(\mathbf{b} - (\mathbf{L}+\mathbf{U})\mathbf{x^{(k)}})$
- GS HACIA ADELANTE: $\mathbf{x^{(k+1)}} = (\mathbf{L}+\mathbf{D})^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x^{(k)}})$
- GS HACIA ATRÁS: $\mathbf{x^{(k+1)}} = (\mathbf{D}+\mathbf{U})^{-1}(\mathbf{b} - \mathbf{L}\mathbf{x^{(k)}})$
- SOR HACIA ADELANTE: $\mathbf{x^{(k+1)}} = \left(\mathbf{L} + \frac{1}{\omega}\mathbf{D}\right)^{-1}\left(\mathbf{b} + \left(\left(\frac{1-\omega}{\omega}\right)\mathbf{D} - \mathbf{U}\right)\mathbf{x^{(k)}}\right)$

Para ver que estas versiones son equivalentes a las vistas en clase, compruebelo con el casos 3×3, es decir, con

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad \mathbf{b} = (b_1, b_2, b_3)^{\mathtt{t}}$$

3. En `MATLAB` se puede calcular el número de condicionamiento de una matriz $\mathbf{A}$ con `cond(A,p)`, donde p es la $p-$norma matricial. Compruebe el cálculo que obtuvo en el ejercicio **4.1.**3. Además calcule el número de condicionamiento de $\mathbf{H} =$ `hilb(n)` con $n = 10, 100, 1000, 10000$ en las normas $p = 1, 2$ e `'inf'`. Comente los resultados.

4. Se tiene que para cualquier matriz $\mathbf{A} \in \mathbb{R}^{n\times n}$ no singular se cumple que el número de condicionamiento en la $2-$norma

$$\kappa_2(\mathbf{A}) = \sqrt{\frac{\lambda_n}{\lambda_1}}$$

donde λ_1 es el menor valor propio y λ_n es el mayor valor propio de la matriz $\mathbf{A}^{\mathtt{t}}\mathbf{A}$. Utilice este resultado para calcular $\kappa_2(\mathbf{A})$ con

$$\mathbf{A} = \begin{pmatrix} 3 & -2 & 3 \\ 1 & 6 & -3 \\ 1 & 2 & 1 \end{pmatrix}$$

Capítulo 6

Interpolación

La interpolación describe el problema de encontrar una curva (llamada *interpolante*) que contiene a todos los puntos $(x_0, y_0), ..., (x_n, y_n)$ los cuales son llamados *nodos*. Existen muchas formas de interpolación. La teoría de interpolación es uan base importante para la integración numérica conocida como *cuadratura*.

Sea $\mathbb{P}_n[x]$ el espacio lineal de todos los polinomios de grado a lo sumo n. Cada $p \in \mathbb{P}_n[x]$ está definido únicamente por $n+1$ coeficientes. Esto nos da $n+1$ grados de libertad, mientras que interpolando con $x_0, ..., x_n$ dados nos da $n+1$ condiciones.

Teorema 6.1. *Sean $n+1$ puntos $(x_0, y_0), ..., (x_n, y_n)$ donde $x_i \neq x_j$ si $i \neq j$. Entonces existe un único polinomio $p \in \mathbb{P}_n[x]$ tal que $p_n(x_k) = y_k$ para $k = 0, 1, ..., n$.*

Al tener el polinomio $p_n(x) = \sum_{k=0}^{n} a_k x^k$ los coeficientes a_k se determinan resolviendo el sistema $(n+1) \times (n+1)$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

en donde la matriz de coeficientes es conocida como *matriz de Vandermonde* la cual es no singular cuando $x_i \neq x_j$ si $i \neq j$.

6.1. Interpolación de Lagrange

Definición 6.1 (Polinomios cardinales de Lagrange)**.** Los *polinomios cardinales de* Lagrange están dados por

$$L_k(x) := \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{x - x_i}{x_k - x_i}, \qquad x \in \mathbb{R}$$

Cada polinomio cardinal de Lagrange pertenece a $\mathbb{P}_n[x]$. Se puede comprobar que $L_k(x_j) = 0$ para $j \neq k$ y que $L_k(x_k) = 1$.

Polinomio interpolador de Lagrange

Definición 6.2. El polinomio interpolante está dado por

$$p(x) = \sum_{k=0}^{n} y_k L_k(x) = \sum_{k=0}^{n} y_k \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

Lema 6.2. *El polinomio interpolante es único.*

Demostración: Supongamos que $p, q \in \mathbb{P}_n(x)$ son dos polinomios interpoladores de Lagrange, luego $p - q \in \mathbb{P}_n(x)$, el cual tiene el conjunto de $n + 1$ ceros $\{x_0, x_1, .., x_n\}$. Por resultado de Álgebra se tiene que $p - q \equiv 0$, con lo cual $p(x) = q(x)$ para todo $x \in [x_0, x_n]$. □

Teorema 6.3 (Error de interpolación)**.** *Dada $f \in \mathcal{C}^{n+1}([a, b])$ y $f(x_i) = y_i$, donde $x_0, ..., x_n$ son distintos por pares. Sea $p \in \mathbb{P}_n[x]$ el polinomio interpolante. Entonces para cada $x \in [a, b]$ existe $\xi \in [a, b]$ tal que*

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i)$$

Es decir,

Cota del error Lagrange

$$|f(x) - p(x)| \leq \max_{\zeta \in [a,b]} \left| f^{(n+1)}(\zeta) \right| \cdot \frac{1}{(n+1)!} \prod_{i=0}^{n} |x - x_i|$$

6.2. Método de diferencias divididas

Otra forma para describir el polinomio interpolante fue presentado por Newton. Para ello se necesitan los siguientes conceptos.

Definición 6.3 (Diferencias divididas)**.** Dados $x_0, ..., x_n \in [a, b]$ distintos dos a dos, sea $p \in \mathbb{P}_n[x]$ polinomio interpolador de $f \in \mathcal{C}^n([a, b])$ en dichos puntos. El coeficiente de x^n en p es denominado *diferencia dividida de grado* n y denotado como $f[x_0, x_1, ..., x_n]$.

De la fórmula de Lagrange se tiene que

$$f[x_0, x_1, ..., x_n] = \sum_{k=0}^{n} f(x_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{1}{x_k - x_i}$$

Teorema 6.4. *Sean $x_0, x_1, ..., x_{k+1}$ distintos dos a dos, donde $k \geq 0$. Entonces*

$$f[x_0, x_1, ..., x_k, x_{k+1}] = \frac{f[x_1, ..., x_{k+1}] - f[x_0, ..., x_k]}{x_{k+1} - x_0}$$

Esta fórmula recursiva da una vía rápida para calcular las diferencias divididas. Esto requiere $O(n^2)$ operaciones y calcula los números $f[x_0, ..., x_i]$ para $i = 0, ..., n$.

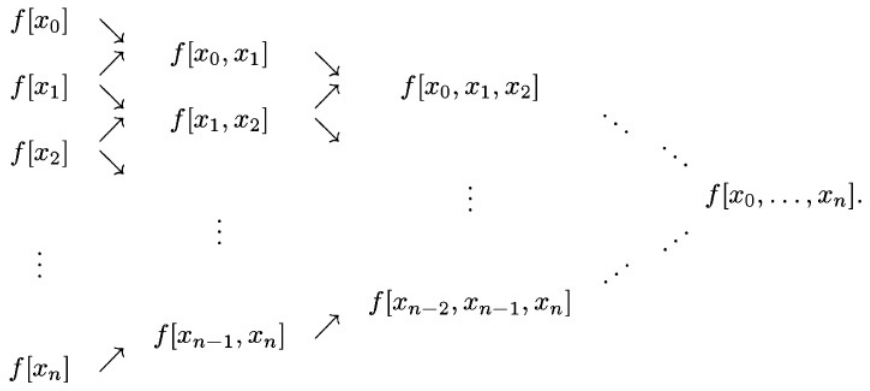


Figura 6.1: Tabla de diferencias divididas. Tomado de Faul

En general, la $m-$ésima diferencia dividida se expresa como

$$\begin{cases} f[x_i] &= f(x_i) \\ f[x_i, x_{i+1}, ..., x_{i+m}] &= \dfrac{f[x_{i+1}, x_{i+2}, ..., x_{i+m}] - f[x_i, x_{i+1}, ..., x_{i+m-1}]}{x_{i+m} - x_i} \end{cases}$$

Fórmula de interpolación de Newton

Teorema 6.5. *Sean $x_0, x_1, ..., x_n$ distintos dos a dos. El polinomio que pertenece a $\mathbb{P}_n[x]$,*

$$p_n(x) := f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, ..., x_{n-1}] \prod_{i=0}^{n-2}(x - x_i) + f[x_0, ..., x_n] \prod_{i=0}^{n-1}(x - x_i)$$

satisface $p_n(x_i) = f(x_i)$ para $i = 0, 1, ..., n$.

Ejemplo 6.1. Calcule el polinomio de interpolación de Lagrange que aproxima la función $f(x) = 2^x \cos(\pi x)$ en el intervalo $[1, 3]$, utilizando el conjunto soporte $\mathscr{S} = \{1, 2, 3\}$.

Solución: Utilizando diferencias divididas de Newton se tienen los siguientes cálculos:

x_i	$f(x_i)$	$f[x_i, x_{i+1}]$	$f[x_0, x_1, x_2]$
1	-2	6	-9
2	4	-12	
3	-8		

Luego, el polinomio de interpolación de Newton (que coincide con el de Lagrange) es

$$p_2(x) = -2 + 6(x - 1) - 9(x - 1)(x - 2) = -9x^2 + 33x - 26$$

◇

Ejemplo 6.2. Sea $f : [-2, 1] \to \mathbb{R}$ una función para la cual se sabe que

$$f'(-2) = 0, \quad f'(-1) = 1, \quad f'(0) = -2, \quad f'(1) = 0$$

a) Determine el polinomio de interpolación de Lagrange $p(x)$ para $f'(x)$ en el conjunto $\{-2, -1, 0, 1\}$.

Solución: Utilizando diferencias divididas se tiene que

x_i	$f'[x_i]$	$f'[x_i, x_{i+1}]$	$f'[x_i, x_{i+1}, x_{i+2}]$	$f'[x_0, x_1, x_2, x_3]$
-2	0	1	-2	$\frac{3}{2}$
-1	1	-3	$\frac{5}{2}$	
0	-2	2		
1	0			

Entonces $f'(x) \approx p(x) = 0 + 1(x+2) - 2(x+2)(x+1) + \frac{3}{2}(x+2)(x+1)x = \frac{3}{2}x^3 + \frac{5}{2}x^2 - 2x - 2$.

b) Si $f(0) = 4$, determine un polinomio que aproxime $f(x)$.

Solución:

Basta calcular $f(x) \approx \int p(x)\,dx = \frac{3}{8}x^4 + \frac{5}{6}x^3 - x^2 - 2x + C$.

Como $f(0) = 4 \Rightarrow C = 4$. Por lo tanto

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

◇

6.3. Interpolación de Hermite

En este método se interpolan tanto las imágenes de una función como las imágenes de su derivada.

Definición 6.4. Sea $\{x_0, x_1, ..., x_n\} \subset [a, b]$ un conjunto soporte de puntos distintos dos a dos. Sean y_i, z_i números reales con $i \in \{1, ..., n\}$. El polinomio $h_{2n+1} \in \mathbb{P}_{2n+1}$ definido por

Polinomio interpolador de Hermite

$$h_{2n+1}(x) := \sum_{i=0}^{n} [H_i(x)y_i + K_i(x)z_i]$$

donde

$$H_k(x_i) = \begin{cases} 1, & \text{si } i = k \\ 0 & \text{si } i \neq k \end{cases} \quad \text{y} \quad H_k'(x_i) = 0,$$

$$K_k(x_i) = 0 \quad \text{y} \quad K_k'(x_i) = \begin{cases} 1, & \text{si } i = k \\ 0 & \text{si } i \neq k \end{cases}$$

es denominado **polinomio de interpolación de Hermite** de grado $2n + 1$ para el conjunto $\{(x_k, y_k, z_k) : k = 0, 1, ..., n\}$.

De manera explícita se sabe que

$$H_i(x) := [L_i(x)]^2(1 - 2L_i'(x_i)(x - x_i)), \qquad K_i(x) := [L_i(x)]^2(x - x_i)$$

donde $L_i(x)$ es el polinomio cardinal de Lagrange.

Teorema 6.6. *Sea $f \in C^{2n+2}([a, b])$. Sea $h_{2n+1} \in \mathbb{P}_{2n+1}$ el polinomio de interpolación de Hermite de f en los $n + 1$ nodos distintos $x_0, ..., x_n \in [a, b]$. Entonces para todo $x \in [a, b]$ existe $\xi \in]a, b[$ tal*

que

$$f(x) = h_{2n+1}(x) + \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\prod_{j=0}^{n}(x-x_j)^2$$

También se cumple que

Cota del error Hermite

$$|f(x) - h_{2n+1}(x)| \leq \frac{\max\limits_{\zeta\in[a,b]}|f^{(2n+2)}(\zeta)|}{(2n+2)!}\prod_{j=0}^{n}(x-x_j)^2$$

Ejemplo 6.3. Calcule el polinomio de interpolación $h(x)$ de Hermite para $f(x) = x^5$ para lo nodos $(0,0)$, (a,a^5), con $a > 0$. Compruebe que $\xi = \dfrac{x+2a}{5}$ satisface la igualdad

$$f(x) - h(x) = \frac{f^{(4)}(\xi)}{24}x^2(x-a)^2$$

Solución: Partimos de que ξ satisface la igualdad dada y así obtenemos el polinomio de interpolación de Hermite:

$$h(x) = f(x) - \frac{f^{(4)}(\xi)}{24}x^2(x-a)^2$$

$$= x^5 - \frac{120\xi}{24}x^2(x-a)^2 = x^5 - (x+2a)x^2(x-a)^2$$

$$= x^5 - (x^3+2ax^2)(x^2-2ax+a^2) = 3a^2x^3 - 2a^3x^2$$

Este polinomio satisface que $h(0) = 0$, $h'(0) = 0$, $h(a) = a^5$, $h'(a) = 5a^4$. Por unicidad, este último es efectivamente el polinomio de interpolación de Hermite para la función y los nodos dados.

◇

Observación 6.1. La vía tradicional de hacer este ejercicio es la de calcular explícitamente el polinomio de interpolación de Hermite, pero en la resolución anterior se pudieron aprovechar los datos dados.

Observación 6.2. También se puede utilizar el método de diferencias divididas para Hermite, lo cual se puede visualizar en la siguiente tabla:

ω_i	$f[\omega_i]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$
$\omega_0 = x_0$	$f(\omega_0)$		
		$f'(x_0)$	
$\omega_1 = x_0$	$f(\omega_1)$		$f[\omega_0,\omega_1,\omega_2]$
		$f[\omega_1,\omega_2]$	
$\omega_2 = x_1$	$f(\omega_2)$		
		$f'(x_1)$	
$\omega_3 = x_1$	$f(\omega_3)$		$f[\omega_1,\omega_2,\omega_3]$
		$f[\omega_3,\omega_4]$	
$\omega_4 = x_2$	$f(\omega_4)$		
		$\vdots$	

con lo cual $h(x) = f(\omega_0)+f'(x_0)(x-x_0)+f[\omega_0,\omega_1,\omega_2](x-x_0)^2+f[\omega_0,\omega_1,\omega_2,\omega_3](x-x_0)^2(x-x_1)+...$

6.4. Interpolación por trazadores cúbicos

Definición 6.5. Sea $f \in C([a,b])$. Un trazador interpolante de grado k para f con soporte $\mathscr{S} = \{x_0,...,x_n\}$ es una función $S(x)$ formada por polinomios a trozos con grado a lo sumo k, la cual tiene las siguientes condiciones:

1. El polinomio de grado menor o igual que k, $s_i(x) = S(x)\big|_{[x_i,x_{i+1}]}$
2. Para todo $i = 0,...,n$, $S(x_i) = f(x_i)$.
3. $s_i^{(j)}(x_{i+1}) = s_{i+1}^{(j)}(x_{i+1})$ para $i = 0,1,...,n-2$ y $j = 0,1,...,k-1$.

Si $k = 1$ los trazadores son lineales, si $k = 2$ los trazadores son cuadráticos. Si $k = 3$ los trazadores se denominan **cúbicos**.

Trazadores cúbicos

Sean los nodos (x_i, y_i) con $\mathscr{S} = \{x_0, ..., x_n\}$. El trazador cúbico

$$S(x) = \sum_{i=0}^{n-1} \chi_{[x_i, x_{i+1}]}(x) \cdot s_i(x)$$

con $s_i(x)$ a lo más cúbico, cumple las siguientes condiciones:

1. $S(x_i) = y_i$ para $i = 0, 1, ..., n$;
2. $s_i(x_{i+1}) = s_{i+1}(x_{i+1})$ para $i = 0, 1, ..., n-2$;
3. $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$ para $i = 0, 1, ..., n-2$;
4. $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$ para $i = 0, 1, ..., n-2$.

Estas condiciones dan un total de $4n-2$ ecuaciones y se cuenta con $4n$ incógnitas al tener presente que

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + y_i$$

A las condiciones anteriores se le agraga una de las siguientes, para completar la cantidad de ecuaciones:

- **Frontera libre o natural:** $S''(x_0) = S''(x_n) = 0$.
- **Frontera sujeta:** $S'(x_0) = z_0$, $S'(x_n) = z_n$.

Haciendo $h_i := x_{i+1} - x_i$ para $i = 0, 1, ..., n-1$, $M_i := S''(x_i)$ para $i = 0, 1, ..., n$ y luego de varios cálculos se obtiene que

$$a_i = \frac{M_{i+1} - M_i}{6h_i}, \qquad b_i = \frac{M_i}{2}, \qquad c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(M_{i+1} + 2M_i)$$

para $i = 0, 1, ..., n-1$.

Dado que $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$ para $i = 0, 1, ..., n-2$ se puede comprobar que

$$h_i M_i + 2(h_i + h_{i+1})M_{i+1} + h_{i+1}M_{i+2} = u_i$$

donde

$$u_i = 6\left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}\right)$$

y se obtiene un sistema lineal de $n-1$ ecuaciones con $n+1$ incógnitas. Las dos ecuaciones restantes

dependen de la condición de frontera que se elija.

- Si se tiene frontera libre o natural entonces $M_0 = M_n = 0$, con lo cual se obtiene el sistema tridiagonal simétrico con $n-1$ ecuaciones y $n-1$ incógnitas

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix}$$

- Si se tiene frontera sujeta entonces $S'(x_0) = z_0$, $S'(x_n) = z_n$, de donde se tiene que

$$2h_0M_0 + h_0M_1 = v, \qquad h_{n-1}M_{n-1} + 2h_{n-1}M_n = w$$

con

$$v = 6\left(\frac{y_1 - y_0}{h_0} - z_0\right), \qquad w = 6\left(z_n - \frac{y_n - y_{n-1}}{h_{n-1}}\right)$$

con lo cual se obtiene el sistema tridiagonal simétrico con $n+1$ ecuaciones y $n+1$ incógnitas

$$\begin{pmatrix} 2h_0 & h_0 & & & & \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ & & & h_{n-1} & 2h_{n-1} \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} v \\ u_0 \\ \vdots \\ u_{n-2} \\ w \end{pmatrix}$$

6.5. Implementación de los métodos con MATLAB

Polinomio cardinal $L_k(x)$ de Lagrange

```
function [c]=Lk(k,x)
npts=length(x);
c=1;
for i=[1:(k-1), (k+1):npts] % i \neq k
    den = x(k) - x(i);
    c = conv(c, [1/den, - x(i)/den]);
```

```
end
end
```

Polinomio interpolador de Lagrange

```
function [c] = interlagrange(x,y)
npts=length(x);
c=0;
for k=1:npts
    c = c + Lk(k,x)*y(k);
end
end
```

Polinomios cardinales de Hermite

```
function [Hk, Kk] = HkKk(k, x)
    % Determinar el correspondiente polinomio de Lagrange
    Lx = Lk(k, x);
    % Evaluar la derivada xk
    dLx = polyval(polyder(Lx), x(k));
    % Calcular Lk^2
    Lx2 = conv(Lx, Lx);
    % Determinar el polinomio Hk
    q = [-2*dLx, 1 + 2*dLx*x(k)];
    Hk = conv(Lx2, q);
    % Deterimar el polinomio Kk
    q = [1, -x(k)];
    Kk = conv(Lx2, q);
end
```

Polinomio interpolador de Hermite

```
function [c] = InterpoladorHermite(x, y, z)
    nPts = length(x);
    c = 0;
    for k = 1 : nPts
        % Calcular polinomios de Hermite
        [Hk, Kk] = HkKk(k,x);
        % Acumular al polinomio interpolador
        c = c + Hk * y(k) + Kk * z(k);
    end
end
```

Polinomio interpolador de Lagrange con Diferencias Divididas

```
function [C, D] = newpoly(X, Y)
n = length(X);
D = zeros(n);
D(:,1) = Y';

for j = 2:n
    for k = j:n
        D(k, j) = (D(k, j-1) - D(k-1, j-1))/(X(k)-X(k-j+1));
    end
end

C = D(n,n);

for k = (n-1):-1:1
    C = conv(C, poly(X(k)));
    m = length(C);
    C(m) = C(m) + D(k,k);
end

D = [X', D];

end
```

Polinomio interpolador de Hermite con Diferencias Divididas

```
function [C, D] = HermiteNewPoly(X, Y, Z)
n = length(X);
D = zeros(2*n);

WdupliMatriz = [X; X];
W = (WdupliMatriz(:))';

fWdupliMatriz = [Y; Y];
fW = (fWdupliMatriz(:))';
D(:,1) = fW;

for j = 2:2*n
    for k = j:2*n
        D(k, j) = (D(k, j-1) - D(k-1, j-1))/(W(k)-W(k-j+1));
        if j == 2
            for r = 1:length(Z)
                D(2*r, j) = Z(r);
            end
        end
    end
end

C = D(2*n,2*n);

for k = (2*n-1):-1:1
    C = conv(C, poly(W(k)));
    m = length(C);
    C(m) = C(m) + D(k,k);
end

D = [W', D];

end
```

Trazador cúbico natural

```
function [coeficientes] = trazadorCubicoNatural(x,y)
    % Verficar que los vectores siempre sean columna
    if size(x, 1) == 1 % El vector es fila
        x = x';
    end
    if size(y, 1) == 1 % El vector es fila
        y = y';
    end
    % -----------------------------------------------
    nPts = length(x); % Cantidad de puntos
    % Calcular los valores de h
    h = zeros(nPts-1, 1);
    for i = 1 : nPts-1
        h(i) = x(i+1) - x(i);
    end
    % Calcular el vector u
    u = zeros(nPts-2, 1);
    for i = 1 : nPts-2
        u(i) = 6*( (y(i+2)-y(i+1))/h(i+1) - (y(i+1)-y(i))/h(i) );
    end
    % Calcular la matriz del trazador natural
    A = zeros(nPts-2, nPts-2);
    for i = 1 : nPts-2
        A(i,i) = 2*( h(i) + h(i+1) );
    end
    A = A + diag(h(2:nPts-2),-1) + diag(h(2:nPts-2),1);
    % Calcular los momentos
    M = A \ u;
    M = [0; M; 0]; % Agregar los momentos de los extremos
    % Calcular los coeficientes del trazador
    a = zeros(nPts-1, 1);
    b = zeros(nPts-1, 1);
    c = zeros(nPts-1, 1);
    for i = 1:nPts-1
        a(i) = ( M(i+1)-M(i) ) / ( 6*h(i) );
```

```
        b(i) = M(i) / 2;
        c(i) = (y(i+1)-y(i))/h(i) - (h(i)/6)*( M(i+1) + 2*M(i) );
    end
    % Agrupar los coeficientes en una matriz
    coeficientes = [a, b, c, y(1:end-1)];
end
```

Trazador cúbico con frontera sujeta

```
function [coeficientes] = trazadorCubicoSujeto(x,y,z0,zn)
    % Verficar que los vectores siempre sean columna
    if size(x, 1) == 1 % El vector es fila
        x = x';
    end
    if size(y, 1) == 1 % El vector es fila
        y = y';
    end
    % ------------------------------------------------
    nPts = length(x); % Cantidad de puntos
    % Calcular los valores de h
    h = zeros(nPts-1, 1);
    for i = 1 : nPts-1
        h(i) = x(i+1) - x(i);
    end
    % Calcular el vector u
    u = zeros(nPts-2, 1);
    for i = 1 : nPts-2
        u(i) = 6*( (y(i+2)-y(i+1))/h(i+1) - (y(i+1)-y(i))/h(i) );
    end
    % Agregar v y w al vector u
    u = [6*( (y(2)-y(1))/h(1) - z0 ); % v
         u;
         6*( zn - (y(nPts)-y(nPts-1))/h(nPts-1) )]; % w
    % Calcular la matriz del trazador natural
    A = zeros(nPts-2, nPts-2);
    for i = 1 : nPts-2
```

```
28          A(i,i) = 2*( h(i) + h(i+1) );
29      end
30      A = A + diag(h(2:nPts-2),-1) + diag(h(2:nPts-2),1);
31      % Extneder la matriz a la del trazador sujeto
32      f1 = [2*h(1), h(1), zeros(1, nPts-2)];
33      f2 = [zeros(1, nPts-2), h(nPts-1), 2*h(nPts-1)];
34      c1 = [h(1); zeros(nPts-3, 1)];
35      c2 = [zeros(nPts-3, 1); h(nPts-1)];
36      A = [f1; [c1, A, c2]; f2];
37      % Calcular los momentos
38      M = A \ u;
39      % Calcular los coeficientes del trazador
40      a = zeros(nPts-1, 1);
41      b = zeros(nPts-1, 1);
42      c = zeros(nPts-1, 1);
43      for i = 1:nPts-1
44          a(i) = ( M(i+1)-M(i) ) / ( 6*h(i) );
45          b(i) = M(i) / 2;
46          c(i) = (y(i+1)-y(i))/h(i) - (h(i)/6)*( M(i+1) + 2*M(i) );
47      end
48      % Agrupar los coeficientes en una matriz
49      coeficientes = [a, b, c, y(1:end-1)];
50  end
```

6.6. Ejercicios

6.6.1. Interpolación de Lagrange. Diferencias divididas. Acotación del error

1. Encuentre el polinomio de interpolación de Lagrange que aproxima la función $f(x) = 2^x \cos(\pi x)$ en el intervalo $[2, 4]$, utilizando el conjunto soporte $\mathscr{S} = \{2, 3, 4\}$.

2. Considere los nodos $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, con $x_i \neq x_j$, para todo $i \neq j$, y sea

$$p_2(x) = a_2 x^2 + a_1 x + a_0$$

Tomando las condiciones $p_2(x_k) = y_k$ para $k \in 0, 1, 2$,

a) Plantee el sistema en donde las incógnitas son a_0, a_1, a_2, de forma matricial, con la matriz de Vandermonde correspondiente.

b) Muestre que la matriz de Vandermonde del inciso a) es invertible, y calcule las incógnitas a_k en función de x_k e y_k.

c) Calcule los polinomios $L_k(x)$ y luego el polinomio de interpolación de Lagrange. Simplifique al máximo y compare los coeficientes del polinomio de interpolación de Lagrange con los coeficientes a_k calculados en b).

d) Calcule el polinomio interpolador de Newton, exprese explícitamente las diferencias divididas en función de x_k e y_k. Luego simplifique dicho polinomio y compare con el inciso b) y el c).

3. Una función $f(x)$ se aproxima por medio del polinomio interpolador $p(x)$. Use la siguiente tabla para determinar el polinomio de Lagrange por diferencias divididas:

x	$f(x)$
1.6	3
2	8
2.5	14
3.2	15
4	8
4.5	2

Además determine una aproximación de $f(3)$.

4. Considere la ecuación $f(x) = 2\,\mathrm{sen}(x) - 2x^2 + 1 = 0$.

Utilice los nodos

$$(f(1),\, 1) \quad (f(2),\, 2) \quad (f(3),\, 3) \quad (f(4),\, 4)$$

para determinar el polinomio interpolador de Lagrange $p(y)$. Estime el valor de $f^{-1}(0)$, sabiendo que f es invertible en $[1, 4]$. Con base en lo anterior, ¿cuál es un valor aproximado de la solución de $f(x) = 0$ en el intervalo $[1, 4]$?

5. Sea $f(x) = \frac{1}{x}$. Sabiendo que $f[x_0, x_1, ..., x_k] = \dfrac{(-1)^k}{x_0 x_1 \cdots x_k}$ determine el polinomio interpolador $p(x)$ utilizando $x_0 = 1, x_1 = 2, x_2 = \frac{1}{3}, x_3 = \frac{1}{5}, x_4 = \frac{1}{8}$.

6. Sea $f(x) = e^x$ en el intervalo $[1, 2]$. Determine el polinomio de interpolación de Lagrange con los puntos del conjunto $\mathscr{S} = \{1, 1.5, 2\}$. Luego encuentre una cota para el error al aproximar f por p en dicho intervalo.

7. Sea $f : \mathbb{R} \to \mathbb{R}$ una función para la cual se sabe que

$$f'(-2) = -1, \quad f'(-1) = 1, \quad f'(0) = -2, \quad f'(1) = 5$$

Determine un polinomio de interpolación para f' con la información dada. Simplifique al máximo sus resultados. Sabiendo que $f(1) = 1006$, determine una aproximación para $f(x)$.

6.6.2. Interpolación de Hermite. Acotación del error

1. Construya el polinomio de interpolación de Hermite que cumpla que $h_3(0) = 0$, $h_3(1) = 1$, $h_3'(0) = 1$ y $h_3'(1) = 0$.

2. Sea $f(x) = e^x$ en el intervalo $[1, 2]$. Determine el polinomio de interpolación de Hermite h con los puntos del conjunto $\mathscr{S} = \{1, 1.5, 2\}$. Luego encuentre una cota para el error al aproximar f por h en dicho intervalo.

3. Calcule, con detalles, el polinomio de interpolación $h(x)$ de Hermite para $f(x) = x^5$ usando $x_0 = 0$, $x_1 = a > 0$. Determine el valor exacto de $\xi \in]0, a[$ que satisface la igualdad

$$f(x) - h(x) = \frac{f^{(4)}(\xi)}{24}[\pi_2(x)]^2$$

4. Sea $f(x) = \sqrt{x}$. Utilice el polinomio de interpolación de Hermite en los puntos $\{1, 4, 9\}$ para aproximar el valor de $\sqrt{3}$. Calcule el error relativo de dicha aproximación.

5. Sea $f \in C^6([-1, 1])$ y $h \in \mathbb{P}_5[x]$ el polinomio interpolador de Hermite con $h(x_i) = f(x_i)$ y $h'(x_i) = f'(x_i)$, para $x_i \in \{-1, 0, 1\}$. Muestre que

$$\int_{-1}^{1} h(t)\, dt = \frac{7}{15}f(-1) + \frac{16}{15}f(0) + \frac{7}{15}f(1) + \frac{1}{15}f'(-1) - \frac{1}{15}f'(1)$$

Use este resultado para aproximar el valor de $\displaystyle\int_{-1}^{1} \operatorname{sen}(x^2)\, dx$.

6.6.3. Interpolación por trazadores cúbicos

1. Considere los nodos $(0, 0), (1, 1), (2, 2)$.

   a) Determine el trazador cúbico con frontera natural. Grafíquelo con `MATLAB` como función a trozos.

   b) Determine el trazador cúbico con frontera sujeta tal que $S'(0) = S'(2) = 1$. Grafíquelo con `MATLAB` como función a trozos.

c) ¿Qué puede concluir de los dos incisos anteriores?

2. Un trazador cúbico natural S en $[0, 2]$ está definido por

$$S(x) = \begin{cases} 1 + 2x - x^3, & \text{si } x \in [0, 1[\\ a(x-1)^3 + b(x-1)^2 + c(x-1) + 2, & \text{si } x \in [1, 2] \end{cases}$$

Calcule los valores de a, b, c.

3. Sea $f(x) = (3x - 2)e^x$ y el conjunto soporte $\mathscr{S} = \{-3, \frac{-3}{2}, \frac{1}{4}, 1\}$.

a) Determine el trazador cúbico con frontera natural.

b) Determine el trazador cúbico con frontera sujeta tal que $S'(-3) = \frac{-1}{2}$, $S'(1) = 10$.

c) Grafíque con `MATLAB` f y los trazadores determinados en los incisos previos, así como los nodos asociados al conjunto soporte.

Capítulo 7

Derivación e integración numérica

7.1. Diferenciación numérica

La diferenciación numérica es un problema mal condicionado en el sentido en que pequeñas perturbaciones del input puede dar significativas diferencias en el output. Pero es importante destacar que hay problemas que que requieren aproximaciones de la derivada.

Entonces se introduce el concepto de *discretización del error*, el cual es el error que ocurre cuando una función continua es aproximada por un conjunto discreto de valores. Esto es usualmente conocido como *truncamiento local del error*. Es distinto de *redondear el error*, el cual es el error inherente natural que surge al trabajar con la representación punto flotante.
Dada una función $f : \mathbb{R} \to \mathbb{R}$, la derivada se define como

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

Luego $f'(x)$ puede ser estimado escogiendo h pequeño,

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

Dicha aproximación es generalmente llamada una *diferencia cociente*. La cuestión importante aquí es cómo escoger h. Supongamos que f puede ser derivada al menos tres veces, y usando Taylor,

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + O(h^3)$$

con lo cual,

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) + O(h^2)$$

Así, el valor absoluto del error discretizado es aproximadamente $\frac{h}{2}|f''(x)|$.

7.1.1. Aproximaciones a la derivada

De la definición de derivada por límite es claro que si elegimos una sucesión $\{h_k\}$ tal que $h_k \to 0$ y calculamos el límite de la sucesión

$$D_k := \frac{f(x+h_k)-f(x)}{h_k}, \qquad \text{para } k = 1, 2, ...$$

calcularemos $f'(x)$. Como solo calcularemos una cantidad finita de términos hasta $k = N$ por ejemplo, y usaremos D_N como aproximación de la derivada, ¿qué valor de h_N hay que elegir para asegurar que D_N es una buena aproximación de $f'(x)$?

Teorema 7.1 (Fórmula centrada de orden $O(h^2)$)**.** *Supongamos que $f \in C^3([a,b])$ y que $x-h, x, x+h \in [a,b]$. Entonces*

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

Además existe $c = c(x) \in [a,b]$ tal que

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2 f^{(3)}(c)}{6}$$

Teorema 7.2 (Fórmula centrada de orden $O(h^4)$)**.** *Supongamos que $f \in C^5([a,b])$ y que $x-2h, x, x+2h \in [a,b]$. Entonces*

$$f'(x) \approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h}$$

Además existe $c = c(x) \in [a,b]$ tal que

$$f'(x) = \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} + \frac{h^4 f^{(5)}(c)}{30}$$

Supongamos que se usa un programa para calcular de manera que se puede escribir

$$f(x_0-h) = y_{-1} + e_{-1} \qquad \text{y} \qquad f(x_0+h) = y_1 + e_1$$

donde se han aproximado $f(x_0-h) \approx y_{-1}$, $f(x_0+h) \approx y_1$, siendo los errores de redondeo e_{-1} y e_1 respectivamente.

Corolario 7.3. *a) Supongamos que f verifica las hipótesis de la fórmula de dos puntos, y que se utiliza la fórmula computacional $f'(x_0) \approx \dfrac{y_1 - y_{-1}}{2h}$. Entonces el término del error de esta fórmula viene dado por*

$$f'(x_0) = \frac{y_1 - y_{-1}}{2h} + \frac{e_1 - e_{-1}}{2h} - \frac{h^2 f^{(3)}(c)}{6}$$

en donde los dos últimos términos corresponden a errores de redondeo y error de truncamiento.

b) Supongamos que f verifica las hipótesis de la fórmula de dos puntos, y que además $|e_{-1}| \leq \varepsilon$, $|e_1| \leq \varepsilon$, $M := \max\{|f^{(3)}(x)| : a \leq x \leq b\}$, entonces

$$\left| \frac{e_1 - e_{-1}}{2h} - \frac{h^2 f^{(3)}(c)}{6} \right| \leq \frac{\varepsilon}{h} + \frac{Mh^2}{6}$$

El valor de h que minimiza la expresión del lado derecho es $h = \left(\dfrac{3\varepsilon}{M}\right)^{\frac{1}{3}}$.

Ahora, sea $f(x_0 + kh) := y_k + e_k$.

Corolario 7.4. *a) Supongamos que f verifica las hipótesis de la fórmula de cuatro puntos, y que se utiliza la fórmula computacional $f'(x_0) \approx \dfrac{-y_2 + 8y_1 - 8y_{-1} + y_{-2}}{12h}$. Entonces el término del error de esta fórmula viene dado por*

$$f'(x_0) = \frac{-y_2 + 8y_1 - 8y_{-1} + y_{-2}}{12h} - \frac{e_2 - 8e_1 + 8e_{-1} - e_{-2}}{12h} + \frac{h^4 f^{(5)}(c)}{30}$$

en donde los dos últimos términos corresponden a errores de redondeo y error de truncamiento.

b) Supongamos que f verifica las hipótesis de la fórmula de cuatro puntos, y que además $|e_k| \leq \varepsilon$, $M := \max\{|f^{(5)}(x)| : a \leq x \leq b\}$, entonces

$$\left| \frac{-e_2 + 8e_1 - 8e_{-1} + e_{-2}}{12h} + \frac{h^4 f^{(5)}(c)}{30} \right| \leq \frac{3\varepsilon}{2h} + \frac{Mh^4}{30}$$

El valor de h que minimiza la expresión del lado derecho es $h = \left(\dfrac{45\varepsilon}{4M}\right)^{\frac{1}{5}}$.

7.1.2. Extrapolación de Richardson

Definimos $f_k := f(x_k) = f(x_0 + kh)$. Sea $D_0(h) := \dfrac{f(x_0+h) - f(x_0-h)}{2h}$. Entonces,

$$\begin{aligned} f'(x_0) &\approx D_0(h) + Ch^2 \\ f'(x_0) &\approx D_0(2h) + 4Ch^2 \end{aligned}$$

Luego se tiene que

$$3f'(x_0) \approx 4D_0(h) - D_0(2h) = \frac{4(f_1 - f_{-1})}{2h} - \frac{f_2 - f_{-2}}{4h}$$

con lo cual,

$$f'(x_0) \approx \frac{D_0(h) - D_0(2h)}{3} = \frac{-f_2 + 8f_1 - 8f_{-1} + f_{-2}}{12h}$$

El método para obtener una fórmula de mayor orden para aproximar $f'(x_0)$ a partir de una fórmula de orden menor se denomina **extrapolación**. Denotemos

$$D_1(h) := \frac{-f(x_0+2h) + 8f(x_0+h) - 8f(x_0-h) + f(x_0-2h)}{12h}$$

Entonces se tiene que

$$\begin{aligned} f'(x_0) &\approx D_1(h) + Ch^4 \\ f'(x_0) &\approx D_1(2h) + 16Ch^4 \end{aligned}$$

Suponiendo que $f^{(5)}(x)$ tiene signo constante y no cambia demasiado rápido, podemos suponer que $f^{(5)}(c_1) \approx f^{(5)}(c_2)$, para eliminar los términos que contienen h^4 en las fórmulas previas, y se tiene que

$$f'(x_0) \approx \frac{16D_1(h) - D_1(2h)}{15}$$

Para ir mejorando los cálculos se tiene el siguiente resultado.

Teorema 7.5 (Extrapolación de Richardson). *Supongamos que $D_{k-1}(h)$ es una aproximación de orden $O(h^{2k})$ a $f'(x_0)$ que verifica que*

$$f'(x_0) = D_{k-1}(h) + c_1h^{2k} + c_2h^{2k+2} + ...,$$

con lo que

$$f'(x_0) = D_{k-1}(2h) + 4^k c_1h^{2k} + 4^{k+1}c_2h^{2k+2} + ...$$

Entonces se puede construir la siguiente aproximación mejorada

$$f'(x_0) = D_k(h) + O(h^{2k+2}) = \frac{4^k D_{k-1}(h) - D_{k-1}(2h)}{4^k - 1} + O(h^{2k+2})$$

Derivación numérica usando extrapolación

La idea es que $f'(x) \approx D(n,n)$, en donde las aproximaciones $D(j,k)$ (con $k \leq j$) se almacenan en una matriz triangular inferior, cuya primer columna es

$$D(j,1) = \frac{f(x + 2^{-j}h) - f(x - 2^{-j}h)}{2^{1-j}h}$$

y los elementos de la fila $j-$ésima, para $j \geq 2$, son

$$D(j,k) = D(j,k-1) + \frac{D(j,k-1) - D(j-1,k-1)}{4^k - 1}, \qquad 2 \leq k \leq j$$

7.1.3. Derivadas de orden superior (con diferencias centradas)

Sea $f_k := f(x_0 + kh)$ con $k = -3:3$. Algunas fórmulas de diferencias centradas de orden $O(h^2)$ son las siguientes:

$$\begin{aligned} f'(x_0) &\approx \frac{f_1 - f_{-1}}{2h} \\ f''(x_0) &\approx \frac{f_1 - 2f_0 + f_{-1}}{h^2} \\ f'''(x_0) &\approx \frac{f_2 - 2f_1 + 2f_{-1} - f_{-2}}{2h^3} \end{aligned}$$

En el caso de orden $O(h^4)$, algunas fórmulas de diferencias centradas son las siguientes:

$$\begin{aligned} f'(x_0) &\approx \frac{-f_2 + 8f_1 - 8f_{-1} + f_{-2}}{12h} \\ f''(x_0) &\approx \frac{-f_2 + 16f_1 - 30f_0 + 16f_{-1} - f_{-2}}{12h^2} \\ f'''(x_0) &\approx \frac{-f_3 + 8f_2 - 13f_1 + 13f_{-1} - 8f_{-2} + f_{-3}}{8h^3} \end{aligned}$$

Por ejemplo, para deducir la fórmula de orden $O(h^2)$ de $f''(x)$ tenemos lo siguiente:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2 f''(x)}{2} + \frac{h^3 f'''(x)}{6} + \frac{h^4 f^{(4)}(x)}{24} + ..., \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2 f''(x)}{2} - \frac{h^3 f'''(x)}{6} + \frac{h^4 f^{(4)}(x)}{24} + ... \end{aligned}$$

$$\Rightarrow f(x+h)+f(x-h)=2f(x)+\frac{2h^2f''(x)}{2}+\frac{2h^4f^{(4)}(x)}{24}+...$$

Despejando $f''(x)$ se obtiene

$$f''(x)=\frac{f(x+h)-2f(x)+f(x-h)}{h^2}-\frac{2h^2f^{(4)}(x)}{4!}-\frac{2h^4f^{(6)}(x)}{6!}-...-\frac{2h^{2k-2}f^{(2k)}(x)}{(2k)!}-...$$

$$=\frac{f_1-2f_0+f_{-1}}{h^2}-\frac{h^2f^{(4)}(c)}{12}+O(h^2),\quad c\in[x-h,\,x+h]$$

Existen también fórmulas de diferencias no centradas, denominadas diferencias progresivas o regresivas, de orden $O(h^2)$ por ejemplo, que se deducen derivando el polinomio interpolador de Lagrange, las cuales se pueden consultar pp. 360-362 en [**mathews**].

7.1.4. Aplicación en una EDO de segundo orden

Consideremos la ecuación

$$-y''(t) + p(t)y'(t) + q(t)y(t) = f(t), \quad 0 < t < 1$$

con condiciones de frontera $y(0) = a$, $y(1) = b$ y con $p, q, f \in C([0,1])$. Consideremos la discretización uniforme de $[0,1]$ dada por $t_k := kh$, con $k = 0 : N$ y $h = \frac{1}{N}$. Utilizando las diferencias centradas de orden $O(h^2)$ de las derivadas de $y(t)$ y evaluando en $t = t_k$ para $k = 1 : N-1$, se tiene que

$$-\frac{y(t_{k+1}) - 2y(t_k) + y(t_{k-1})}{h^2} + p_k \frac{y(t_{k+1}) - y(t_{k-1})}{2h} + q_k y(t_k) + O(h^2) = f_k$$

en donde $p_k := p(t_k)$, $q_k := q(t_k)$, $f_k := f(t_k)$. Truncando los términos de orden dos, la solución de la ecuación diferencial se aproxima mediante

$$-\frac{y(t_{k+1}) - 2y(t_k) + y(t_{k-1})}{h^2} + p_k \frac{y(t_{k+1}) - y(t_{k-1})}{2h} + q_k y(t_k) = f_k, \quad k = 1 : N-1$$

con lo cual se tiene que

$$\left(-1 - \frac{h}{2}p_k\right) y_{k-1} + (2 + h^2 q_k) y_k + \left(-1 + \frac{h}{2}p_k\right) y_{k+1} = h^2 f_k$$

Haciendo $b_k := -1 - \dfrac{h}{2}p_k$, $a_k := 2 + h^2 q_k$, $c_k := -1 + \dfrac{h}{2}p_k$, las $N-1$ ecuaciones que resultan se representan en el producto matricial

$$\begin{pmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & & \\ & b_3 & a_3 & \ddots & & \\ & & \ddots & \ddots & c_{N-3} & \\ & & & b_{N-2} & a_{N-2} & c_{N-2} \\ & & & & b_{N-1} & a_{N-1} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} h^2 f_1 - b_1 a \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{N-2} \\ h^2 f_{N-1} - c_{N-1} b \end{pmatrix}$$

La matriz tridiagonal es invertible en el caso particular $q(t) > 0$ (con h suficientemente pequeño) ya que la matriz será estrictamente diagonal dominante.

7.1.5. Diferencias finitas

En general, *diferencias finitas* son expresiones matemáticas de la forma $f(x+b) - f(x+a)$. Si una diferencia finita se divide por $b-a$ entonces se obtiene una *diferencia cociente*. Tenemos los siguientes operadores con diferencias finitas:

Operadores de diferencias finitas

- **Operador esquema hacia adelante**: $\Delta_+ f(x) = f(x+h) - f(x)$
- **Operador esquema hacia atrás**: $\Delta_- f(x) = f(x) - f(x-h)$
- **Operador esquema central**: $\delta f(x) = f(x+\frac{h}{2}) - f(x-\frac{h}{2})$

Adicionalmente podemos definir:

- **Operador shift**: $Ef(x) = f(x+h)$
- **Operador promediante**: $\mu_0 f(x) = \frac{1}{2}(f(x+\frac{h}{2}) - f(x-\frac{h}{2}))$
- **Operador diferencial**: $Df(x) = \frac{d}{dx}f(x) = f'(x)$

Todos los operadores conmutan y pueden ser expresados en términos de los otros:

$$\begin{aligned} \Delta_+ &= E - I \\ \Delta_- &= I - E^{-1} \\ \delta &= E^{\frac{1}{2}} - E^{\frac{-1}{2}} \\ \mu_0 &= \frac{1}{2}(E^{\frac{1}{2}} + E^{\frac{-1}{2}}) \end{aligned}$$

Más importante, de la expansión de Taylor se tiene que

$$E = e^{hD}$$

y entonces

$$\begin{aligned} \frac{1}{h}\Delta_+ &= \frac{1}{h}(e^{hD} - I) = D + O(h) \\ \frac{1}{h}\Delta_- &= \frac{1}{h}(I - e^{-hD}) = D + O(h) \end{aligned}$$

Se sigue de lo anterior que los operadores de esquema hacia adelante y hacia atrás aproximan el operador diferencial, o en otras palabras, la primera derivada con un error de $O(h)$.

Se puede mostrar que la combinación del operador esquema central y el operador promediante

dan una mejor aproximación de la primera derivada. De hecho, se puede lograr mayor precisión haciendo lo siguiente:

$$\frac{1}{2h}(\Delta_+ + \Delta_-) = \frac{1}{2h}(e^{hD} - I + I - e^{-hD}) = D + O(h^2)$$

Derivadas de orden superior pueden ser aproximadas aplicando los operadores diferencia muchas veces. Por ejemplo, la derivada de orden n esquemas hacia adelante, atrás y central están dados por

$$\begin{aligned}
\Delta_+^n f(x) &= \sum_{i=0}^{n}(-1)^i \binom{n}{i} f(x + (n-i)h) \\
\Delta_-^n f(x) &= \sum_{i=0}^{n}(-1)^i \binom{n}{i} f(x - ih) \\
\delta^n f(x) &= \sum_{i=0}^{n}(-1)^i \binom{n}{i} f\left(x + \left(\frac{n}{2} - i\right)h\right)
\end{aligned}$$

Combinación de diferencias de orden superior pueden ser usadas para construir mejores aproximaciones. Por ejemplo,

$$\frac{1}{h}(\Delta_+ - \frac{1}{2}\Delta_+^2)f(x) = \frac{-1}{2h}(f(x+2h) - 4f(x+h) + 3f(x))$$

el cual puede escribirse como

$$\frac{-1}{2h}(E^2 - 4E + 3I) = \frac{-1}{2h}(e^{2hD} - 4e^{hD} + 3I) = D + O(h^2)$$

7.2. Integración numérica

La integración numérica es una herramienta esencial que se utiliza en ciencias e ingenierías para obtener valores aproximados de integrales definidas que no pueden calcularse de manera exacta. Por ejemplo, en termodinámica estadística, el modelo de Debye para calcular la capacidad calórica de un sólido considera la función

$$\Phi(x) := \int_0^x \frac{t^3}{e^t - 1}\, dt$$

Otro ejemplo es la función error `erf` que aparece en Probabilidad, Estadística, y en la solución de algunas ecuaciones diferenciales en derivadas parciales. Debe su nombre a que está relacionada con la distribución gaussiana de los errores en observaciones o mediciones experimentales:

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

Agreguemos un ejemplo más: las funciones de Bessel tienen diversas aplicaciones en física e ingeniería, como resolver ecuaciones diferenciales parciales con simetría cilíndrica y analizar fenómenos de propagación de ondas. Acá tenemos una de ellas:

$$J_0(x) := \frac{1}{\pi} \int_0^\pi \cos(x \operatorname{sen}(\theta))\, d\theta$$

Entremos en detalle. Vamos a considerar la evaluación numérica de las integrales de la forma

$$I = \int_a^b f(x)\, dx$$

Cuadratura

Una *regla de cuadratura* aproxima dicha integral por

$$I \approx Q_n(f) = \sum_{i=1}^{n} w_i f(x_i)$$

Los puntos x_i, $i = 1, ..., n$ son llamados las *abscisas* escogidas tal que $a \leq x_1 < ... < x_n \leq b$. Los coeficientes w_i son denominados los *pesos*. Las reglas de cuadratura son derivadas al integrar polinomios interpoladores de la función evaluada en las abcisas. Usualmente solo pesos positivos son permitidos puesto que sumar o restar términos se determina por el signo de la función en ese punto. Además esto evita pérdida de significancia.

7.2.1. La regla del punto medio y la regla del trapecio

La regla más simple surge de interpolar una función por una constante, la cual es un polinomio de grado cero, y es razonable que dicha constante tome el valor del punto medio en el intervalo $[a, b]$. Esto es conocido como *regla del punto medio*. La fórmula es

Regla del punto medio

$$I \approx (b-a) f\left(\frac{a+b}{2}\right)$$

El punto medio es la única abcisa y su peso es igual a $b - a$.

Teorema 7.6. *La regla del punto medio tiene el siguiente término de error*

$$\int_a^b f(x)\,dx = (b-a) f\left(\frac{a+b}{2}\right) + \frac{1}{24} f''(\xi)(b-a)^3$$

en donde ξ es algún punto en el intervalo $[a, b]$. Es decir,

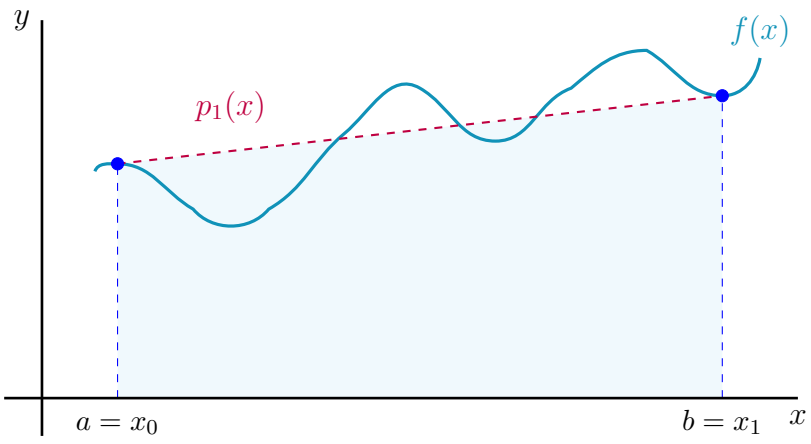
$$|I - Q_{rectángulo}(f)| \leq \frac{M_2}{24}|b-a|^3$$

donde $M_2 := \max_{z\in[a,b]} |f''(z)|$ ■

Alternativamente, podemos integrar el polinomio lineal cuyos nodos son $(a, f(a))$ y $(b, f(b))$. Esta es la *regla del trapecio* descrita por la fórmula

Regla del trapecio

$$I \approx \frac{b-a}{2}[f(a) + f(b)]$$



Algorithm 9 Cuadratura del trapecio

Require: f continua sobre $[a, b]$

Ensure: $\tilde{I}$ aproximación de $\int_a^b f(x)\,dx$

$\tilde{I} \leftarrow \frac{b-a}{2}(f(a) + f(b))$;

Teorema 7.7. *La regla del trapecio tiene el siguiente término de error*

$$\int_a^b f(x)\,dx = \frac{b-a}{2}[f(a) + f(b)] - \frac{1}{12}f''(\xi)(b-a)^3$$

en donde ξ es algún punto en el intervalo $[a, b]$, es decir,

$$|I - Q_{trapecio}(f)| \leq \frac{M_2}{12}|b-a|^3$$

donde $M_2 := \max_{z\in[a,b]} |f''(z)|$

Ejemplo 7.1. La integral $\int_0^1 e^x\,dx$ puede aproximarse por la regla del trapecio como

$$\int_0^1 e^x\,dx \approx \frac{1-0}{2}(e^0 + e^1) = \frac{1}{2}(1+e) \approx 1.85914$$

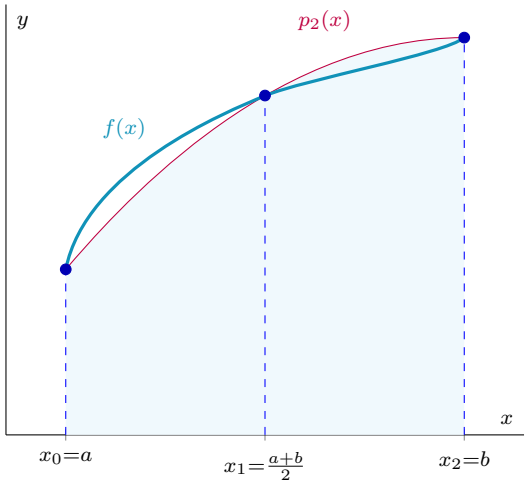
◇

7.2.2. La regla de Simpson 1/3

El término del error puede reducirse significativamente si se integra un polinomio cuadrático, que posea nodos en los puntos extremos y punto medio del intervalo. Esto se conoce como *regla de* Simpson y la fórmula es

Regla de Simpson 1/3

$$I \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$



Algorithm 10 Cuadratura de Simpson

Require: f continua sobre $[a, b]$
Ensure: $\tilde{I}$ aproximación de $\int_a^b f(x)\,dx$
$c = \frac{a+b}{2}$;
$\tilde{I} \leftarrow \frac{b-a}{6}(f(a) + 4f(c) + f(b))$;

Teorema 7.8. *La regla de Simpson tiene el siguiente término de error*

$$\int_a^b f(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{1}{90}f^{(4)}(\xi)\left(\frac{b-a}{2}\right)^5$$

en donde ξ es algún punto en el intervalo $[a, b]$. Es decir,

$$|I - Q_{Simpson}(f)| \leq \frac{M_4}{2880}|b-a|^5$$

donde $M_4 := \max_{z\in[a,b]} |f^{(4)}(z)|$

Ejemplo 7.2. La integral $\int_0^1 e^x\,dx$ puede aproximarse por la regla de Simpson como

$$\int_0^1 e^x\,dx \approx \frac{1-0}{6}(e^0 + 4e^{0.5} + e^1) \approx 1.71886$$

◇

7.2.3. Cuadraturas de Newton–Cotes

La idea de aproximar f usando polinomio interpolante en abcisas igualmente espaciadas se puede extender a n puntos. Dichas reglas con llamadas de Newton–Cotes. Se distinguen dos

tipos de reglas de Newton–Cotes: tipo *cerrado*, las cuales usan valores de la función en los puntos límite del intervalo; y los de tipo *abierto*, los cuales no usan los valores de la función en los puntos límite del intervalo.

Las reglas cerradas son

Trapecio	$(b-a)/2\,[f(a)+f(b)]$
Simpson	$(b-a)/6\,[f(a)+4f((a+b)/2)+f(b)]$
Simpson $\frac{3}{8}$	$(b-a)/8[f(a)+3f((2a+b)/3)+3f((a+2b)/3)+f(b)]$
Boole	$(b-a)/90[7f(a)+32f((3a+b)/4)+12f((a+b)/2)$ $+32f((a+3b)/4)+7f(b)]$

y sus términos de error son

Trapecio	$-(b-a)^3/12\,f''(\xi)$
Simpson	$-(b-a)^5/2880\,f^{(4)}(\xi)$
Simpson $\frac{3}{8}$	$-(b-a)^5/6480 f^{(4)}(\xi)$
Boole	$-(b-a)^7/1935360 f^{(6)}(\xi)$

Las reglas abiertas son

Punto medio	$(b-a)/2\,[f(a)+f(b)]$
sin nombre	$(b-a)/2[f((2a+b)/3)+f((a+2b)/3)]$
Milne	$(b-a)/3[2f((3a+b)/4)-f((a+b)/2)+2f((a+3b)/4)]$

y sus términos de error son

Punto medio	$(b-a)^3/24\,f''(\xi)$
sin nombre	$(b-a)^3/36 f''(\xi)$
Milne	$7(b-a)^5/23040 f^{(4)}(\xi)$

Reglas de orden superior de Newton–Cotes son raramente utilizadas, por dos razones. Primeramente, para n grandes algunos de los pesos son negativos, lo cual da inestabilidad numérica. Segundo, métodos basados en polinomios de orden alto con puntos equiespaciados tienen las mismas desventajas de los polinomios de interpolación de grados altos: ocurre el fenómeno de Runge.

7.2.4. Método de coeficientes indeterminados

Si la cuadratura calcula de manera exacta hasta cierto orden de polinomios, entonces los pesos se pueden calcular con un sistema de ecuaciones.

Por ejemplo, si tenemos la cuadratura $Q(f)=\omega_0 f(a)+\omega_1 f(b)$ en el intervalo $[a,b]$ de manera que la cuadratura es exacta para el conjunto $\mathbb{P}_1[x]$, tomando la base canónica tenemos que $\mathbb{P}_1[x]=\text{gen}\{1,\,x\}$.

Luego,

$$\begin{aligned}\int_a^b 1\,dx &= b-a = \omega_0 f(a) + \omega_1 f(b)\\ &= \omega_0 + \omega_1\end{aligned}$$

$$\begin{aligned}\int_a^b x\,dx &= \frac{b^2-a^2}{2} = \omega_0 f(a) + \omega_1 f(b)\\ &= \omega_0 a + \omega_1 b\end{aligned}$$

Resolviendo se tiene que $\omega_0 = \omega_1 = \dfrac{b-a}{2}$, la cual es la cuadratura del Trapecio sin más ni menos.

De una manera similar se puede comprobar que la cuadratura que fija los puntos $(a, f(a)), ((a+b)/2, f((a+b)/2)), (b, f(b))$ en el intervalo $[a, b]$ y que es exacta para polinomios cuadráticos es precisamente la cuadratura de Simpson 1/3.

7.2.5. Cuadraturas compuestas

Las *reglas compuestas* también son conocidas como reglas *repetidas, iteradas o extendidas*. La idea es separar la integral dada en integrales "más pequeñas" (usualmente el intervalo es equiespaciado) aplicando (usualmente) la misma regla de cuadratura en cada subintervalo y finalmente se suman los resultados.

Sea N la cantidad de subintervalos equiespaciados tales que $h = \dfrac{b-a}{N}$, entonces las reglas compuestas clásicas son las siguientes, con sus respectivos errores:

- **Punto medio compuesto:**

$$\int_a^b f(x)\,dx \approx h\sum_{i=1}^{N} f(a + (i-\tfrac{1}{2})h)$$

$$|E| = \sum_{i=1}^{N} \frac{1}{24} f''(\xi_i)h^3 \leq \frac{1}{24} \max_{\xi\in[a,b]} |f''(\xi)| N h^3 = O(h^2)$$

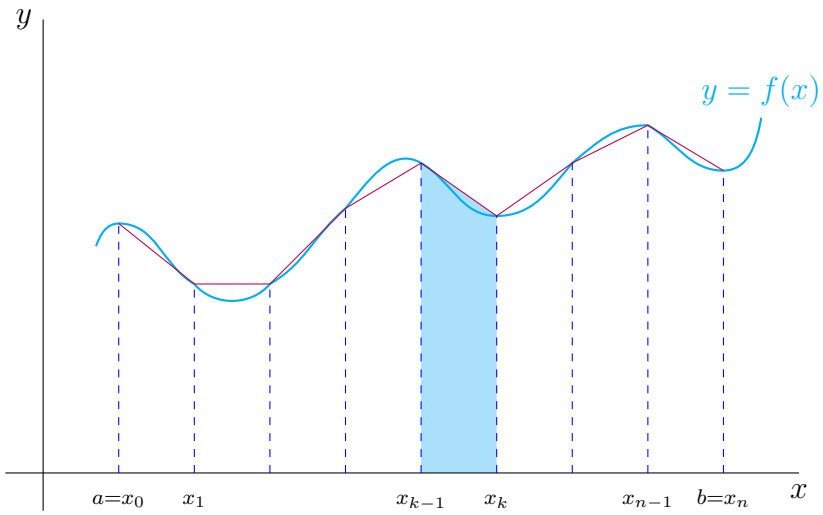
  en donde cada $\xi_i \in [a + (i-1)h, a + ih]$.

- **Trapecio compuesto:**

$$\int_a^b f(x)\,dx \approx \frac{h}{2} f(a) + h\sum_{i=1}^{N-1} f(a+ih) + \frac{h}{2} f(b)$$

$$|E| = \sum_{i=1}^{N} \frac{1}{12} f''(\xi_i)h^3 \leq \frac{1}{12} \max_{\xi\in[a,b]} |f''(\xi)| N h^3 = O(h^2)$$

en donde cada $\xi_i \in [a+(i-1)h, a+ih]$.

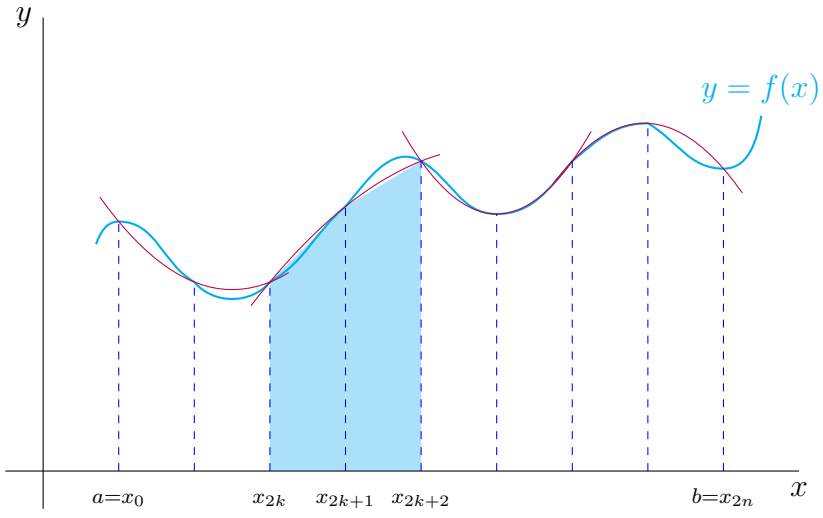


- **Simpson 1/3 compuesto:**

$$\int_a^b f(x)\,dx \approx \frac{h}{6}\left[f(a) + 2\sum_{i=1}^{N-1} f(a+ih) + 4\sum_{i=1}^{N} f(a+(i-\tfrac{1}{2})h) + f(b)\right]$$

$$|E| \leq \frac{1}{180}\max_{\xi\in[a,b]} |f^{(4)}(\xi)|h^4 = O(h^4)$$

en donde cada $\xi_i \in [a+(i-1)h, a+ih]$.



7.2.6. Reglas recursivas e integración de Romberg

Vamos a utilizar unas notaciones para la exposición de la temática. Supongamos que se divide el intervalo $[a,b]$ en M subintervalos $[x_k, x_{k+1}]$ con paso $h = \frac{b-a}{M}$ y con $x_k = a+kh$ para $k = 0:M$.

La regla compuesta del trapecio con M subintervalos se puede denotar como

$$T(f,h) := \frac{h}{2}\sum_{k=1}^{M}(f(x_{k+1}) + f(x_k))$$

o bien

$$T(f,h) = \frac{h}{2}(f_0 + f_M) + h\sum_{k=1}^{M-1} f_k$$

Ahora, supongamos que dividimos $[a,b]$ en $2M$ subintervalos $[x_k, x_{k+1}]$ con $h = \frac{b-a}{2M}$, con $x_k = a+kh$ siendo $k = 0:2M$. La regla compuesta de Simpson con $2M$ subintervalos se puede expresar de la siguiente forma:

$$S(f,h) := \frac{h}{3}\sum_{k=1}^{M}(f(x_{2k-2}) + 4f(x_{2k-1}) + f(x_{2k}))$$

o bien

$$S(f,h) = \frac{h}{3}(f_0 + f_{2M}) + \frac{2h}{3}\sum_{k=1}^{M-1} f_{2k} + \frac{4h}{3}\sum_{k=1}^{M} f_{2k-1}$$

Se mostrará cómo calcular las aproximaciones de la regla de Simpson usando combinaciones lineales especiales de las aproximaciones dadas por la regla del trapecio. Las aproximaciones mejoran su precisión al aumentar la cantidad de subintervalos. Se comienza generando con la regla del trapecio una sucesión de aproximaciones $\{T(J)\}$. Al duplicar la cantidad de subintervalos se duplica la cantidad de evaluaciones de la función ya que hay que evaluar la función en los puntos previos y en los puntos de los subintervalos previos. El siguiente teorema ayuda a eliminar las evaluaciones y sumas redundantes.

Teorema 7.9 (Reglas del trapecio sucesivas)**.** *Suponga que $J \geq 1$ y que los puntos $\{x_k := a + kh\}$ dividen $[a,b]$ en $2^J = 2M$ subintervalos del mismo tamaño $h = \frac{b-a}{2^J}$. Las reglas del trapecio compuestas $T(f,h)$ y $T(f,2h)$ verifican que*

$$T(f,h) = \frac{T(f,2h)}{2} + h\sum_{k=1}^{M} f(x_{2k-1})$$

Definición 7.1 (Sucesión de aproximaciones con la regla del trapecio)**.** Se define $T(0) = \frac{h}{2}(f(a) + f(b))$, que es la regla del trapecio con incremento $h = b - a$, y para cada $J \geq 1$, se define $T(J) = T(J) = T(f,h)$, donde $T(f,h)$ es la regla compuesta del trapecio con incremento $h = \frac{b-a}{2^J}$.

Corolario 7.10 (Regla recursiva del trapecio). *La sucesión* $\{T(J)\}$ *se define como*

$$\begin{cases} T(0) &= \dfrac{h}{2}(f(a)+f(b)), \\ \\ T(J) &= \dfrac{T(J-1)}{2} + h\displaystyle\sum_{k=1}^{M} f(x_{2k-1}), \quad J=1,2,... \end{cases}$$

siendo $h = \dfrac{b-a}{2^J}$ *y* $x_k = a + kh$.

Existe una relación importante entre la regla del trapecio y la regla de Simpson, que es la siguiente:

$$S(f,h) = \frac{4T(f,h) - T(f,2h)}{3}$$

Teorema 7.11 (Regla recursiva de Simpson). *Suponga que* $\{T(J)\}$ *es la sucesión de aproximaciones con la regla del trapecio. Si* $J \geq 1$ *y* $S(J)$ *es la aproximación dada por la regla compuesta de Simpson con* 2^J *subintervalos de* $[a,b]$, *entonces* $S(J)$ *verifica que*

$$S(J) = \frac{4T(J) - T(J-1)}{3}, \quad J=1,2,...$$

Teorema 7.12 (Regla recursiva de Boole). *Suponga que* $\{S(J)\}$ *es la sucesión de aproximaciones con la regla de Simpson. Entonces se tiene que*

$$B(J) = \frac{16S(J) - S(J-1)}{15}, \quad J=2,3,...$$

Las fórmulas recursivas de Simpson y de Boole son casos especiales del proceso conocido como método de integración de Romberg.

Lema 7.13 (Esquema de Richardson para la integración de Romberg). *Dadas dos aproximaciones* $R(2h, K-1)$ *y* $R(h, K-1)$ *de una cantidad* Q *que verifican*

$$Q = R(h,K-1) + c_1h^{2K} + c_2h^{2K+2} + ...$$

y

$$Q = R(2h,K-1) + c_1 4^K h^{2K} + c_2 4^{K+1} h^{2K+2} + ...$$

entonces se puede construir una aproximación mejor que viene dada por la fórmula

$$Q = \frac{4^K R(h,K-1) - R(2h,K-1)}{4^K - 1} + O(h^{2K+2})$$

Definición 7.2 (Esquema de integración de Romberg). Se define $\{R(J,K) : J \geq K\}_{J=0}^{\infty}$ de

fórmulas de cuadratura para aproximar $\int_a^b f(x)\,dx$ de la siguiente manera:

$$\begin{aligned} R(J,0) &= T(J), \quad \text{para } J \geq 0 \\ R(J,1) &= S(J), \quad \text{para } J \geq 1 \\ R(J,2) &= B(J), \quad \text{para } J \geq 2 \end{aligned}$$

La fórmula general para construir las mejoras es

$$R(J,K) := \frac{4^K R(J,K-1) - R(J-1,K-1)}{4^K - 1}, \quad J \geq K$$

Teorema 7.14 (Precisión de la integración de Romberg)**.** *Supongamos que* $f \in C^{2K+2}([a,b])$, *entonces*

$$\int_a^b f(x)\,dx = R(J,K) + O(h^{2K+2})$$

donde $h = \dfrac{b-a}{2^J}$.

7.2.7. Polinomios ortogonales

Sea $w :]a,b[\to \mathbb{R}^+$ función continua e integrable, la cual se denomina *función peso*. Dadas dos funciones f, g se define

$$\langle f, g \rangle_w := \int_a^b f(x)g(x)w(x)$$

Lo anterior es un producto interno definido en el espacio de funciones integrables con respecto al peso w, e induce una norma

$$\|f\| = \langle f, f \rangle_w^{1/2}$$

Definición 7.3. Sea $\{\varphi_j\}_{j=0}^{\infty}$ un conjunto de polinomios, donde cada φ_j tiene exactamente grado j. Dicho conjunto se denomina *sistema de polinomios ortogonales* si $\langle \varphi_j, \varphi_k \rangle_w = 0$ si y solo si $j \neq k$.

Se muestran algunas funciones de peso para obtener polinomios ortogonales. Considere $\alpha, \beta > -1$.

De forma general, dado cualquier peso w es posible obtener un sistema de polinomios ortogonales con respecto a w. La forma usual es utilizar el proceso de ortogonalización de Gram-Schmidt

Nombre	Notación	Intervalo	Función Peso
Legendre	P_n	$[-1,1]$	$w(x) \equiv 1$
Jacobi	$P_n^{(\alpha,\beta)}$	$]-1,1[$	$w(x) = (1-x)^\alpha (1+x)^\beta$
Chebyshev (primer tipo)	T_n	$]-1,1[$	$w(x) = (1-x^2)^{-1/2}$
Chebyshev (segundo tipo)	U_n	$[-1,1]$	$w(x) = (1-x^2)^{1/2}$
Laguerre	L_n	$[0,+\infty[$	$w(x) = e^{-x}$
Hermite	H_n	$\mathbb{R}$	$w(x) = e^{-x^2}$

Cuadro 7.1: Funciones de pesos y polinomios ortogonales.

sobre la base de polinomios canónica $\{x^j\}_{j=0}^{\infty}$:

$$\begin{aligned}
\varphi_0(x) &= 1 \\
\varphi_1(x) &= x - \frac{\langle x, \varphi_0\rangle_w}{\langle \varphi_0, \varphi_0\rangle_w}\varphi_0 \\
&\vdots \quad\quad \vdots \\
\varphi_{k+1}(x) &= x^{k+1} - \sum_{j=0}^{k} \frac{\langle x^{k+1}, \varphi_j\rangle_w}{\langle \varphi_j, \varphi_j\rangle_w}\varphi_j
\end{aligned}$$

Teorema 7.15. *Sea el sistema de polinomios ortogonales $\{\varphi_j\}_{j=0}^{\infty}$ en el intervalo $]a,b[$ con función de peso $w(x)$. Entonces φ_j con $j \geq 1$ tiene exactamente j ceros reales y distintos, los cuales se encuentran en $]a,b[$.*

7.2.8. Cuadraturas gaussianas

Como las abscisas equidistantes causan problemas en las reglas de cuadratura, el paso siguiente es escoger las abscisas $x_0, ...x_n$ tal que la fórmula será precisa para el polinomio con el mayor grado posible (que es $n+1$). Dicha regla es conocida como *cuadratura gaussiana* o *cuadratura de Gauss–Christoffel*.

Las cuadraturas gaussianas basadas en polinomios ortogonales se describen de la siguiente manera. Sean $x_0, ..., x_n$ raíces del $n+1-$ésimo polinomio ortogonal p_{n+1}. Sea L_i el $i-$ésimo polinomio cardinal de Lagrange para dichos puntos. Entonces los pesos de la cuadratura son calculados como

$$W_i = \int_a^b L_i(x) w(x)\, dx$$

y entonces se tiene que

$$\int_a^b f(x) w(x)\, dx \approx \mathscr{G}(f) := \sum_{i=1}^{n} W_i f(x_i)$$

Teorema 7.16. *La aproximación previa es exacta para polinomios de grado a lo sumo* $2n+1$.

Los pesos de la cuadratura gaussiana son positivos: consideremos $L_k^2(x)$, el cual es un polinomio de grado $2n$, y como la fórmula de cuadratura es exacta hasta ese orden se tiene que

$$0 < \int_a^b L_k^2(x)w(x)\,dx = \sum_{i=0}^{n} W_i L_k^2(x_i) = W_k$$

Cuadraturas de Gauss-Legendre

La cuadratura gaussiana más común es cuando $]a,b[=]-1,1[$ y $w(x)\equiv 1$. Los polinomios ortogonales son los llamados *polinomios de Legendre*. Para construir la regla de cuadratura, se deben determinar los ceros del $n+1-$ésimo polinomio ortogonal de Legendre y calcular los pesos asociados.

Por ejemplo, cuando $n=1$ el polinomio ortogonal de Legendre es $x^2-\frac{1}{3}$, cuyas raíces son $\pm\frac{1}{\sqrt{3}}$. Los polinomios cardinales de Lagarange asociados son $(\pm\sqrt{3}x+1)/2$ y al integrarlos en $[-1,1]$ son iguales a 1. Por tanto, la cuadratura de Gauss-Legendre de dos puntos está dada por

$$\int_{-1}^{1} f(x)\,dx \approx f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

la cual es correcta para polinomios hasta grado 3. La siguiente tabla muestra los casos $n \le 4$:

n	**2n + 1**	**Abscisas**	**Pesos**
1	3	$\pm 1/\sqrt{3}$	1
2	5	0	8/9
		$\pm\sqrt{3/5}$	5/9
3	7	$\pm\sqrt{(3-2\sqrt{6/5})/7}$	$(18+\sqrt{30})/36$
		$\pm\sqrt{(3+2\sqrt{6/5})/7}$	$(18-\sqrt{30})/36$
4	9	0	128/225
		$\pm\frac{1}{3}\sqrt{(5-2\sqrt{10/7})}$	$(322+13\sqrt{70})/900$
		$\pm\frac{1}{3}\sqrt{(5+2\sqrt{10/7})}$	$(322-13\sqrt{70})/900$

Cuadro 7.2: Abscisas y pesos para cuadraturas de Gauss-Legendre con $n \le 5$.

Polinomios de Legendre

Los polinomios de Legendre pueden calcularse de manera recursiva:

$$\begin{cases} P_0(x) &= 1, \\ P_1(x) &= x, \\ (n+1)P_{n+1} &= (2n+1)xP_n - nP_{n-1} \end{cases}$$

Con esta fórmula se tiene que

- $P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$
- $P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x$
- $P_4(x) = \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$

Teorema 7.17. *Si $f \in C^{2n+2}([a,b])$ y la integral es aproximada por una regla de cuadratura gaussiana, entonces*

$$\int_a^b f(x)w(x)\,dx - \sum_{i=1}^{n} w_i f(x_i) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\int_a^b [\pi_{n+1}(x)]^2 w(x)\,dx$$

para algún $\xi \in]a,b[$, donde $\pi_{n+1}(x) := (x-x_0)\cdots(x-x_n) \in \mathbb{P}_{n+1}[x]$.

Teorema 7.18. *Sea f una función continua en $[a,b]$ y w una función positiva, continua e integrable en $]a,b[$. Entonces*

$$\lim_{n\to\infty} \mathscr{G}(f) = \int_a^b f(x)w(x)\,dx$$

Teorema 7.19. *Supongamos que tenemos los nodos $\{x_{n,k}\}_{k=1}^n$ y los pesos $\{w_{n,k}\}_{k=1}^n$ de la cuadratura de Gauss-Legendre con n nodos, en el intervalo $[-1,1]$. Entonces, para aplicar el método de Gauss-Legendre en un intervalo $[a,b]$ se puede usar el cambio de variable*

$$t = \frac{a+b}{2} + \frac{b-a}{2}x, \quad con \quad dt = \frac{b-a}{2}\,dx$$

y la relación

$$\int_a^b f(t)\,dt = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{a+b}{2} + \frac{b-a}{2}x\right)\,dx$$

proporciona la fórmula de cuadratura

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2}\sum_{k=1}^{n} w_{n,k} f\left(\frac{a+b}{2} + \frac{b-a}{2}x_{n,k}\right)$$

7.2.9. Implementación de Trapecio y Simpson compuestas

No definiremos f en un script aparte, sino que usaremos funciones anónimas:

`nombre_funcion = @(argumentos) expresion_funcion`

Por ejemplo:

f= @(x) **exp**(−x^2)

Método del Trapecio Simple

```
function It = trapecioSimple(f, a, b)
% Inputs: funcion f, a, b puntos
% Output: aproximacion de la integral de f en [a, b]

It = (b-a)/2 * (f(a) + f(b));

end
```

Método de Simpson 1/3 simple

```
function Ism = simpsonSimple(f, a, b)
% Inputs: funcion f, a, b puntos
% Output: aproximacion de la integral de f en [a, b]

c = (a+b)/2;

Ism = (b-a)/6 * (f(a) + 4*f(c) + f(b));
end
```

Método del Trapecio Compuesto

```
function [c] = trapecioCompuesto(f,a,b,n)
% aproximar integrales usando trapecio compuesto
% Entradas:     f ---- funcion a integrar
%           [a,b] ---- intervalo de integracion
```

```
5  %               n ---- cantidad de nodos de interpolacion
6  % Salida:       c ---- valor de la integral definida aproximado
7
8  h = (b-a)/n;
9  s = 0;
10 for i = 1:n-1
11     s = s + f(a+i*h);
12 end
13 c = (h/2)*( f(a) + 2*s + f(b));
14 end
```

Método de Simpson compuesto

```
1  function [c] = ReglaCompuestaSimpson (f,a,b,n)
2  % Este programa aproxima integrales utilizando la regla compuesta
      de
3  % Simpson.
4  % Entradas:    f ---- funcion a integrar
5  %          [a,b]---- intervalo de integracion
6  %             n ---- cantidad de nodos de interpolacion
7  % Salida:      c ---- valor de la integral definida aproximado
8
9  h = (b-a)/n;
10 s = 0;
11 s1 = 0;
12
13 for j=1:(n/2)-1
14     s = s + f(a+2*j*h);
15 end
16
17 for j=1:(n/2)
18     s1 = s1 + f(a+(2*j-1)*h);
19 end
20 c = (h/3)*(f(a)+2*s+4*s1+f(b));
21 end
```

Polinomios de Legendre

```
function P = LegendrePolyn(n)
% Input: n natural >= 0
% Output: P polinomio n-esimo de Legendre

if n == 0
    P = 1;
elseif n == 1
    P= [1 0];
else
    P0 = 1;
    P1 = [1 0];

    for k = 2:n

        Q1 = conv([1 0], P1);    % x*P1(x)
        Q0 = [zeros(1, length(Q1)-length(P0)), P0];

        P = (1./k).*((2.*k-1).* Q1 - (k-1).*Q0);
        P0 = P1;
        P1 = P;
    end
end
end
```

Cuadratura de Gauss-Legendre

```
function I = QuadGaussLeg(f, n, a, b)
% Inputs: f funcion, n grado del polinomio de Legendre,
% [a, b] intervalo de integracion
% Output: aproximacion numerica de la integral de f en [a, b]
% utilizando como nodos los ceros del polinomio de Legendre

% Almacenamos los ceros del n-esimo
```

```
% polinomio de Legendre en un vector fila 'r':
r = sort(roots(LegendrePolyn(n)))';

% Almacenaremos los pesos de la cuadratura en un vector nulo 'W':
W = zeros(1, n);

% Iniciamos declarando 'I = 0':
I = 0;

for k = 1:n
    % Calculamos los pesos con los polinomios cardinales de
       Lagrange
    W(k) = diff(polyval(polyint(Lk(k,r)),[-1 1]));

    % Actualizamos 'I' usando el cambio de variable para trabajar
    % en el intervalo [a, b]:
    I = I + (b-a)./2 * (W(k).*f((b-a)./2 * r(k) + (a+b)./2));
end
end
```

Cuadratura compuesta de Gauss-Legendre

```
function I = GaussLegC(f, n, a, b, m)
% Inputs: f funcion, n grado del polinomio de Legendre,
% [a, b] intervalo de integracion, m cantidad de intervalos
% Output: aproximacion numerica de la integral de f en [a, b]
% utilizando como nodos los ceros del polinomio de Legendre

% Paso entre los nodos:
h=(b-a)/m;

% Nodos:
X = a:h:b;

% Iniciando con 'I = 0':
I = 0;
```

```
15
16 for k=1:length(X)-1
17     I = I + QuadGaussLeg(f, n, X(k), X(k+1));
18 end
19 end
```

7.3. Ejercicios

7.3.1. Derivación numérica

1. Sea $f(x) = \text{sen}(x)$, con x en radianes.

   a) Calcule aproximaciones a $f'(0.8)$ usando la fórmula de orden $O(h^2)$ y tomando $h = 0.1$, $h = 0.01$ y $h = 0.001$.

   b) Compare los valores obtenidos con $f'(0.8) = \cos(0.8)$.

   c) Calcule las cotas del error de truncamiento tomando

   $$|f^{(3)}(c)| \leq \cos(0.7) \approx 0.764842187$$

2. Sea $f(x) = \text{sen}(x)$, con x en radianes.

   a) Calcule aproximaciones a $f'(0.8)$ usando la fórmula de orden $O(h^4)$ y tomando $h = 0.1$ y $h = 0.01$.

   b) Compare los valores obtenidos con $f'(0.8) = \cos(0.8)$.

   c) Calcule las cotas del error de truncamiento tomando

   $$|f^{(5)}(c)| \leq \cos(0.6) \approx 0.825335615$$

3. *Fórmulas de derivación parcial.* La fórmula de orden $O(h^2)$ puede adaptarse para calcular las derivadas parciales

   $$\begin{aligned} f_x(x,y) &= \frac{f(x+h,y) - f(x-h,y)}{2h} + O(h^2) \\ f_y(x,y) &= \frac{f(x,y+h) - f(x,y-h)}{2h} + O(h^2) \end{aligned}$$

   Sea $f(x,y) = \dfrac{xy}{x+y}$. Calcule aproximaciones para $f_x(2,3)$ y $f_y(2,3)$ con $h = 0.1$ y $h = 0.01$. Compare los valores obtenidos con los valores exactos.

4. La distancia $D = D(t)$ recorrida por un móvil se muestra en la siguiente tabla:

t	$D(t)$
8.0	17.453
9.0	21.460
10.0	25.752
11.0	30.301
12.0	35.084

Determine la velocidad $V(10)$ mediante derivación numérica. Compare su respuesta, sabiendo que $D(t) = -70 + 7t + 70e^{\frac{-t}{10}}$.

5. Usando la fórmula de Taylor para $f(x+kh)$ con $k = -2, -1, 1, 2$, deduzca la fórmula de diferencias centradas

$$f^{(3)}(x) \approx \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3}$$

7.3.2. Cuadraturas de Newton-Cotes simples y compuestas

1. Utilice la regla del trapecio y la de Simpson para aproximar

$$I = \int_0^{\frac{\pi}{2}} \frac{dx}{1+\cos x}$$

2. Calcule una aproximación para la integral $\int_0^4 e^x\,dx$ con base en la regla

   a) del trapecio compuesta, con 4 subintervalos.

   b) de Simpson compuesta, con 4 subintervalos.

3. Considere la integral

$$I = \int_0^{\pi} \cos(x^2)\,dx \approx 0.5656935142...$$

   a) Utilice las reglas del trapecio y de Simpson para aproximar I. Determine cotas para los errores respectivos al aproximar la integral.

   b) Utilice las versiones compuestas de ambas reglas con 4 subintervalos.

   c) ¿Cuántos subintervalos (el valor de m visto en clase) se requieren en cada regla para obtener aproximaciones con una tolerancia de 10^{-3}?

4. Sea $f \in C^2(\mathbb{R})$. Considere la regla de cuadratura conocida como *cuadratura del rectángulo*:

$$\int_a^b f(t)\,dt \approx (b-a)\cdot f\left(\frac{a+b}{2}\right)$$

   a) Considere $f(x) = 1 - x^2$ en el intervalo $I = [-1, 1]$. Utilice la regla de cuadratura anterior para calcular una aproximación de $A = \int_{-1}^{1} f(t)\,dt$. Realice un dibujo preciso de la gráfica de f y represente geométricamente las áreas calculadas.

   b) Particione el intervalo I en 6 subintervalos de la misma longitud y utilice la cuadratura dada en cada subintervalo (esto es: usar la versión compuesta de la cuadratura) para

obtener una aproximación del área bajo la curva de $f(x)$. Represente gráficamente la situación.

c) Calcule el valor exacto de A y compare con la aproximación obtenida en el inciso b).

5. La regla de Simpson 3/8 compuesta es la siguiente:

$$I \approx \frac{3h}{8}\left[f(x_0) + 3\sum_{i=0}^{m-1} f(x_{3i+1}) + 3\sum_{i=0}^{m-1} f(x_{3i+2}) + 2\sum_{i=0}^{m-2} f(x_{3i+3}) + f(x_{3m})\right]$$

en donde $h = \dfrac{b-a}{3m}$. Considere $f(x) = e^{-\frac{x^2}{2}} = \exp\left(\dfrac{-x^2}{2}\right)$.

a) Calcule la aproximación de $\int_{-1}^{1} f(x)\,dx$ por medio de esta regla compuesta, usando $m = 3$.

b) Para comprobar sus cálculos modifique el código del script `ReglaCompuestaSimpson`, creando uno nuevo para la regla compuesta de Simpson 3/8.

6. Considere la integral $I = \dfrac{1}{\sqrt{2\pi}}\left[\displaystyle\int_{-1}^{0} \frac{e^{-1/2x^2}}{x^2}\,dx + \int_{-1}^{1} e^{-t^2/2}\,dt\right]$

a) Explique por qué las reglas del trapecio y de Simpson 1/3 no permiten aproximar la integral I.

b) Considere la regla de cuadratura compuesta para m subintervalos,

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2m}\left(3f(x_1) + 2\sum_{i=2}^{m-2} f(x_i) + 3f(x_{m-1})\right)$$

con $x_i := a + ih$ para $i = 0 : m$. Use dicha regla con $m = 6$ para mostrar que el valor de $I \approx 0.8651$

7.3.3. Coeficientes indeterminados

1. Considere una cuadratura en $[-1, 1]$ tal que

$$\int_{-1}^{1} f(x)\,dx \approx w_0 f(-\alpha) + w_1 f(\alpha)$$

con $\alpha \in]0, 1[$. Teniendo en cuenta que dicha cuadratura es exacta para polinomios de grado menor o igual que 3, determine α, w_0 y w_1. ¿Es posible encontrar valores para esos parámetros tal que la cuadratura sea exacta para polinomios de grado 4?

2. Dado el intervalo $[a, b]$ y el conjunto soporte $\mathscr{S} = \{a, \frac{a+b}{2}, b\}$, se tiene la siguiente regla de cuadratura:
$$\int_a^b f(x)\,dx \approx w_0 f(a) + w_1 f\left(\tfrac{a+b}{2}\right) + w_2 f(b)$$
Determine el valor de w_0, w_1, w_2 sabiendo que la regla de cuadratura es exacta para polinomios de grado menor o igual que 2. Esta es la regla de Simpson 1/3. Compruebe que también dicha regla de cuadratura es exacta para polinomios de grado 3.

3. La regla de Simpson 3/8 está dada por
$$I = \int_a^b f(x)\,dx \approx \frac{b-a}{8}\left[f(a) + 3f\left(\frac{2a+b}{3}\right) + 3f\left(\frac{a+2b}{3}\right) + f(b)\right]$$
Deduzca dicha fórmula sabiendo que dicha regla es exacta para polinomios de grado menor o igual que 3, y calculando los valores de los pesos
$$I \approx w_0 f(a) + w_1 f\left(\frac{2a+b}{3}\right) + w_2 f\left(\frac{a+2b}{3}\right) + w_3 f(b)$$

4. Determine los pesos w_0 y w_1 y los puntos x_0 y x_1 tales que la aproximación
$$\int_0^1 f(x)\,dx \approx w_0 f(x_0) + w_1 f(x_1)$$
sea exacta para polinomios hasta grado 3

7.3.4. Acotación de errores

1. Determine una cota del error al aproximar ${}_0^1 e^{-x^2}\,dx$ con las reglas del trapecio y Simpson simples.

2. Considere la integral
$$I = \int_0^\pi \cos(x^2)\,dx \approx 0.5656935142...$$
Determine cotas para los errores del trapecio y Simpson simples al aproximar la integral.

7.3.5. Cuadraturas Gaussianas

1. Observe el siguiente video: `https://youtu.be/PWApmpx7wds`

2. Considere la cuadratura gaussiana
$$\int_{-1}^1 f(x) \approx W_0 f(x_0) + W_1 f(x_1)$$

la cual es exacta para polinomios de grado menor o igual que 3. Determine los valores de W_0, W_1, x_0 y x_1. Utilice dicha cuadratura y un cambio de variable apropiado para aproximar el valor de la integral

$$\int_{-2}^{2} e^{-x^2}\,dx$$

3. Considere la base $\{1, x, x^2, x^3\}$ de $\mathbb{P}_3[x]$.

   a) Utilice el proceso de ortogonalización de Gram-Schmidt (producto interior con $w(x) = 1$ en el intervalo $]-1, 1[$) para calcular el conjunto ortogonal $\{\varphi_0, \varphi_1, \varphi_2, \varphi_3\}$ base de $\mathbb{P}_3[x]$.

   b) Determine los ceros de $\varphi_3(x)$ y establezca la fórmula de cuadratura gaussiana

   $$\int_{-1}^{1} f(x)\,dx \approx W_0 f(x_0) + W_1 f(x_1) + W_2 f(x_2)$$

   en donde x_0, x_1, x_2 son las raíces de $\varphi_3(x)$.

   c) Utilice la cuadratura del inciso b) para aproximar la integral

   $$\int_{-1}^{1} e^x \cos(x)\,dx$$

   Calcule el valor exacto de la integral (sin usar aproximaciones, recuerde que se usa integración por partes). Compare la aproximación con el valor exacto (use error relativo).

4. Investigue cómo implementar en `MATLAB` cuadraturas gaussianas.

5. Sea f integrable en $[-1, 1]$. Considere la regla de cuadratura de dos puntos de Gauss-Legendre, la cual está dada por

   $$\int_{-1}^{1} f(t)\,dt \approx f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

   Utilice la versión compuesta para calcular una aproximación de $\displaystyle\int_0^{\pi} \cos(t^2)\,dt$ con el conjunto soporte $\mathscr{S} = \{t_k\}_{k=0}^{3}$ con $t_k = \dfrac{k\pi}{3}$.

Capítulo 8

Ecuaciones Diferenciales Ordinarias

Deseamos aproximar la solución exacta de la ecuación diferencial ordinaria

$$\frac{\partial \mathbf{y}}{\partial t} = \mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \qquad t \geq 0$$

en donde $\mathbf{y} \in \mathbb{R}^n$ y la función $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ es suficientemente "buena". La ecuación es acompañada por la condición inicial $\mathbf{y}(0) = \mathbf{y}_0$.

La siguiente definición es central en el análisis de ODEs.

Definición 8.1 (Función Lipschitz continua)**.** $\mathbf{f}$ es Lipschitz continua si existe una cota $L \geq 0$ tal que

$$\|\mathbf{f}(t, \mathbf{v}) - \mathbf{f}(t, \mathbf{w})\| \leq L\|\mathbf{v} - \mathbf{w}\|, \qquad t \in [t_0, T], \qquad \mathbf{v}, \mathbf{w} \in \mathbb{R}^n.$$

La continuidad en sentido de Lipschitz significa que las pendientes de todas las rectas secantes a la función entre todos los puntos posibles $\mathbf{v}$ y $\mathbf{w}$ están acotadas por una constante positiva. Luego, una función Lipschitz continua está limitada en cuánto y qué tan rápido puede cambiar.

En la teoría de ecuaciones diferenciales, la Lipschitz continuidad es una condición central en el *Teorema de Picard–Lindelöf*, el cual garantiza la existencia y unicidad de una solución para un problema de valor inicial.

Para nuestro análisis de soluciones numéricas podemos asumir que $\mathbf{f}$ es analítica y que siempre podremos tener localmente su serie de Taylor. Queremos calcular $\mathbf{y}_{n+1} \approx \mathbf{y}(t_{n+1})$, $n = 0, 1, ...,$ desde $\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_n$, donde $t_k = t_0 + ih$ y el paso $h > 0$ es pequeño.

Teorema 8.1 (Teorema de Picard)**.** *Sea f una función real definida y continua en $R = [t_0 - a,\, t_0 + a] \times [y_0 - b,\, y_0 + b]$, sea $|f(t, y)| \leq M$ para todo $(t, y) \in R$. Suponga que f es*

Lipschitz continua en y con constante L. Entonces el problema

$$\begin{cases} y'(t) &= f(t, y(t)) \\ y(t_0) &= y_0 \end{cases}$$

tiene solución única en $I = \left[t_0 - \text{mín}\{a, \frac{b}{M}\},\ t_0 + \text{mín}\{a, \frac{b}{M}\}\right]$

Al tener $y'(t) = f(t, y(t)) \Rightarrow \displaystyle\int_{t_0}^{t} y'(u)\,du = y(t) - y(t_0) = \int_{t_0}^{t} f(u, y(u))\,du$

$$\Rightarrow y(t) = y_0 + \int_{t_0}^{t} f(u, y(u))\,du$$

Iteración de Picard

Viendo esto último como problema de punto fijo se puede establecer que

$$y_{k+1}(t) = y_0 + \int_{t_0}^{t} f(x, y_k(x))\,dx$$

la cual es conocida como la iteración de Picard, en donde $y_0(t) = y_0$.

Ahora, tomando $t_0 \leq t \leq T$ y particionando el intervalo $[t_0, T]$ tal que $t_0 < t_1 < ... < t_{n-1} < t_n = T$, en donde $t_i = t_0 + ih$ y $h = \dfrac{T - t_0}{n}$, para $i = 0, 1, ..., n$. Aplicando integración con respecto a t en $[t_k, t_{k+1}]$ se tiene que

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

Haciendo $y(t_k) \approx y_k$ se tiene que

$$y(t_{k+1}) = y_k + \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt, \qquad k = 0, 1, ..., n-1.$$

Entonces se obtiene el conjunto de puntos $\{(t_k, y_k)\}_{k=0}^{n}$ y con estos se puede usar interpolación para aproximar la solución.

8.1. Métodos de un paso

Estos consisten en calcular y_{k+1} a partir de y_k. Considerando que

$$\begin{cases} y(t_0) &= y_0 \\ y_{k+1} &= y_k + \displaystyle\int_{t_k}^{t_{k+1}} f(t, y(t))\, dt \end{cases}$$

algunos de estos métodos son los siguientes.

1. **Método de Euler hacia adelante:** Tomando $f(t, y(t)) \approx f(t_k, y_k)$ en $[t_k, t_{k+1}]$, se tiene que

   $$y_{k+1} = y_k + (t_{k+1} - t_k) f(t_k, y_k)$$

   Haciendo $h = t_{k+1} - t_k$, el método de Euler hacia adelante es

   $$y_{k+1} = y_k + h \cdot f(t_k, y_k)$$

2. **Método de Euler hacia atrás:** Similar al anterior pero haciendo $f(t, y(t)) \approx f(t_{k+1}, y_{k+1})$ en $[t_k, t_{k+1}]$, el método es

   $$y_{k+1} = y_k + h \cdot f(t_{k+1}, y_{k+1})$$

   El método de Euler hacia adelante es explícito, pues solo se van sustituyendo los valores de t_k, y_k en la fórmula, mientras que el método de Euler hacia atrás es implícito y se resuelve con algún método de ecuación no lineal.

3. **Método del trapecio implicito:** o también conocido como **método de Crank–Nicolson** y se basa en la regla del trapecio para aproximar integrales:

   $$y_{k+1} = y_k + \frac{h}{2} \left[f(t_k, y_k) + f(t_{k+1}, y_{k+1}) \right]$$

   el cual también es un método implícito.

4. **Método del trapecio explícito de Heun:** consiste en combinar el método explícito de Euler hacia adelante y el del trapecio implícito:[1]

   $$y_{k+1} = y_k + \frac{h}{2} \left[f(t_k, y_k) + f\left(t_{k+1},\ y_k + \underbrace{h f(t_k, y_k)}_{=\hat{y}_{k+1}} \right) \right]$$

[1] Esto se conoce en general como *predictor–corrector*, en donde el método explícito predice y el implícito corrige.

8.2. Métodos de Runge–Kutta

El orden de precisión del método de EULER es uno, por ello se necesitan métodos que mejoren la aproximación de dicho método. Entre estos se consideran los *métodos de* RUNGE–KUTTA los cuales evalúan puntos intermedios entre (t_k, y_k) y (t_{k+1}, y_{k+1}).

El método de Runge–Kutta (RK) de orden s tiene la siguiente expresión:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

donde $h = t_{n+1} - t_n$. Los coeficientes k_i son términos de aproximación intermedios, evaluados de forma local en f

$$k_i = f\left(t_n + hc_i,\, y_n + \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, ..., s$$

con a_{ij}, b_i, c_i coeficientes propios del esquema numérico elegido dependiendo de la regla de cuadratura usada. Los esquemas de Runge–Kutta pueden ser explícitos o implícitos, en dependencia de las constantes a_{ij} del esquema. Si la matriz (a_{ij}) es triangular inferior con $a_{ij} = 0$ para todo $j = i, ...s$, los esquemas son explícitos.

Para efectos nuestros, consideremos la familia de métodos

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f(t_k + \alpha h,\, y_k + \beta h k_1) \\ y_{k+1} &= y_k + h(ak_1 + bk_2) \end{cases}$$

en donde k_1, k_2 son **pasos intermedios**, α, β, a, b son parámetros a determinar explícitamente. Para mejorar el método de Euler se hace el siguiente esquema:

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f(t_k + \alpha h,\, y_k + \alpha h k_1) \\ y_{k+1} &= y_k + \frac{h}{2\alpha}((2\alpha - 1)k_1 + k_2) \end{cases}$$

Si en este esquema previo se escoge $\alpha = 1$ se obtiene el método del trapecio explícito:

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f(t_k + h,\, y_k + hk_1) = f(t_{k+1},\, y_k + hk_1) \\ y_{k+1} &= y_k + \frac{h}{2}(k_1 + k_2) \end{cases}$$

$$\Longrightarrow y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k)]$$

Ahora, si $\alpha = \frac{1}{2}$ se obtiene lo que se conoce como el **método de Euler modificado**, método del punto medio o también **método RK2**:

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f(t_k + \frac{h}{2},\, y_k + \frac{h}{2}k_1) \\ y_{k+1} &= y_k + hk_2 \end{cases}$$

$$\Longrightarrow y_{k+1} = y_k + hf\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}f(t_k, y_k)\right)$$

Otros métodos usuales de Runge–Kutta son:

- **Método RK3 (Runge–Kutta de tercer orden):**

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f(t_k + \frac{h}{2},\, y_k + \frac{h}{2}k_1) \\ k_3 &= f(t_k + h,\, y_k + h(2k_2 - k_1)) \\ y_{k+1} &= y_k + \frac{h}{6}(k_1 + 4k_2 + k_3) \end{cases}$$

- **Método RK4 (Runge–Kutta de cuarto orden):**

$$\begin{cases} k_1 &= f(t_k, y_k) \\ k_2 &= f\left(t_k + \frac{h}{2},\, y_k + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_k + \frac{h}{2},\, y_k + \frac{h}{2}k_2\right) \\ k_4 &= f(t_k + h,\, y_k + hk_3) \\ y_{k+1} &= y_k + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

Los coeficientes de los h's en la primera entrada y los de las h's de las k_i's en la segunda entrada se tabulan en lo que se conoce como **tablas de Butcher**.

A continuación las tablas de Butcher de RK3 y de RK4, respectivamente:

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Ejemplo 8.1. Considere el problema de Cauchy:

$$\begin{cases} y' - 3\displaystyle\int_0^t e^{-2(t-x)}y(x)\,dx &= 4e^{-2t}, \quad \text{para } 0 \le t \le 1 \\ y(0) &= 4 \end{cases}$$

a) Use la regla del trapecio para aproximar la integral $\displaystyle\int_0^t e^{-2(t-z)}y(z)\,dz$.

Solución:

$$\begin{aligned} \int_0^t e^{-2(t-z)}y(z)\,dz &\approx \frac{t-0}{2}\left(e^{-2(t-t)}y(t) + e^{-2(t-0)}y(0)\right) \\ &= \frac{t}{2}\left(y(t) + 4e^{-2t}\right) \end{aligned}$$

b) Reescriba el problema con base en la información del inciso a) y utilice el método RK3 con paso $h = 0.25$ para determinar una aproximación de $y(0.5)$.

Solución: El problema se reescribe como

$$\begin{cases} y' &= 4e^{-2t} + \dfrac{3t}{2}\left(y(t) + 4e^{-2t}\right), \quad \text{para } 0 \le t \le 1 \\ y(0) &= 4 \end{cases}$$

donde $f(t,y) := 4e^{-2t} + \dfrac{3t}{2}\left(y(t) + 4e^{-2t}\right)$

El esquema RK3 es el siguiente:

$$\begin{cases} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_i + \frac{h}{2},\, y_i + \frac{h}{2}k_1) \\ k_3 &= f(t_i + h,\, y_i + h(2k_2 - k_1)) \\ y_{i+1} &= y_i + \frac{h}{6}(k_1 + 4k_2 + k_3) \end{cases}$$

Teniendo en cuenta que $t_0 = 0$, $t_1 = 0.25$, $t_2 = 0.5$, se tiene que:

- $i = 0$: $k_1 = 4$, $k_2 = 4.5431$, $k_3 = 4.6864$, $\mathbf{y_1} = \mathbf{5.1191}$
- $i = 1$: $k_1 = 5.2556$, $k_2 = 5.8318$, $k_3 = 6.4145$, $\mathbf{y_2} = \mathbf{6.5773} \approx y(t_2) = y(0.5)$.

c) Resuelva el problema de manera exacta y compare el valor exacto $y(0.5)$ con el valor aproximado del inciso b).

Solución: Para el problema original se aplica transformada de Laplace en ambos lados de la ecuación con la condición inicial, haciendo $Y(s) := \mathcal{L}\{y(t)\}$,

$$s\mathcal{L}\{y'(t)\} - 3\mathcal{L}\{y(t) * e^{-2t}\} = \mathcal{L}\{4e^{-2t}\}$$

$$\Rightarrow sY(s) - 4 - \frac{3Y(s)}{s+2} = \frac{4}{s+2} \Longrightarrow Y(s) = \frac{4s+12}{s^2+2s-3} = \frac{4\cancel{(s+3)}}{\cancel{(s+3)}(s-1)} = \frac{4}{s-1}$$

$$\Rightarrow y(t) = \mathcal{L}^{-1}\{Y(s)\} = \mathcal{L}^{-1}\left\{\frac{4}{s-1}\right\} = 4e^t$$

Entonces $\mathbf{y(0.5) = 4e^{0.5} \approx 6.5949}$.

Tomando el error relativo para comparar los valores: $\left|\dfrac{y(0.5) - y_2}{y(0.5)}\right| \approx 0.002669$.

$\diamond$

8.3. Fórmulas multipaso

Los métodos de m pasos expresan y_{k+1} en términos de y_{k-j} con $j = 0, 1, ..., m-1$, para $m \geq 2$, en donde el cálculo también podría ser implícito. Los métodos de Runge–Kutta son esquemas de un paso, los cuales incrementan el orden de precisión del método de Euler, mediante evaluaciones de puntos intermedios. En los métodos multipasos se realizan evaluaciones de f en puntos previos.

Consideremos los **métodos multipasos lineales** para aproximar la solución al problema de valor inicial de Cauchy mediante m pasos, lo cual se expresa como

$$\sum_{j=0}^{m} \alpha_{k+1-j} \cdot y_{k+1-j} = h \sum_{j=0}^{m} \beta_{k+1-j} \cdot f(t_{k+1-j},\, y_{k+1-j})$$

en donde los α's y β's son reales. Tiene que darse que $\alpha_{k+1} \neq 0$ para que el esquema esté bien definido, además α_{k-m+1} y β_{k-m+1} no son ambos nulos para que así el esquema sea de m pasos. Si $\beta_{k+1} = 0$ el esquema es explícito.

8.3.1. Métodos de Adams-Bashfort (AB) y Adams-Moulton (AM)

Considerando la iteración genérica

$$\begin{cases} y_0 & \text{dado,} \\ y_{k+1} & = y_k + \displaystyle\int_{t_k}^{t_{k+1}} f(t, y(t))\, dt \end{cases}$$

Se utiliza alguna cuadratura de Newton–Cotes para calcular una aproximación de la integral. La función $g(t) := f(t, y(t))$ se interpola usando polinomio interpolador de Lagrange. Asúmase que

$f_k := f(t_k, y_k)$ con $k = 0, 1, ..., n$. Entonces, si al interpolar se utilizan los nodos

$$\begin{gathered}(t_{k-(m-1)}, f_{k-(m-1)})\\(t_{k-(m-2)}, f_{k-(m-2)})\\\vdots\\(t_{k-2}, f_{k-2})\\(t_{k-1}, f_{k-1})\\(t_k, f_k)\end{gathered}$$

el esquema es explícito y se denomina **método de Adams–Bashforth de $m-$pasos**. Si se agrega el nodo (t_{k+1}, f_{k+1}):

$$\begin{gathered}(t_{k-(m-1)}, f_{k-(m-1)})\\(t_{k-(m-2)}, f_{k-(m-2)})\\\vdots\\(t_{k-2}, f_{k-2})\\(t_{k-1}, f_{k-1})\\(t_k, f_k)\\(t_{k+1}, f_{k+1})\end{gathered}$$

el esquema es implícito y se denomina **método de Adams–Moulton de $m-$pasos**.

Adams-Bashforth	**Adams-Moulton**
$\begin{cases} y_0, y_1 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{2}\left[3f_k - f_{k-1}\right] \\ \text{para } k = 1, 2, \ldots, n-1 \end{cases}$	$\begin{cases} y_0, y_1 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{12}\left[5f_{k+1} + 8f_k - f_{k-1}\right] \\ \text{para } k = 1, 2, \ldots, n-1 \end{cases}$
$\begin{cases} y_0, y_1, y_2 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{12}\left[23f_k - 16f_{k-1} + 5f_{k-2}\right] \\ \text{para } k = 2, 3, \ldots, n-1 \end{cases}$	$\begin{cases} y_0, y_1, y_2 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{24}\left[9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}\right] \\ \text{para } k = 2, 3, \ldots, n-1 \end{cases}$
$\begin{cases} y_0, y_1, y_2, y_3 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{24}\left[55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}\right] \\ \text{para } k = 3, 4, \ldots, n-1 \end{cases}$	$\begin{cases} y_0, y_1, y_2, y_3 \text{ dados} \\ y_{k+1} = y_k + \frac{h}{720}\left[251f_{k+1} + 646f_k - 264f_{k-1} + 106f_{k-2} - 19f_{k-3}\right] \\ \text{para } k = 3, 4, \ldots, n-1 \end{cases}$

Figura 8.1: Métodos multipasos (2,3 y 4) de **Adams–Bashfort (AB)** y de **Adams–Moulton (AM)**. De la diapositiva de SEQUEIRA.

8.3.2. Predictor-Corrector de cuarto orden (PC4)

El esquema explícito de cuarto orden es el siguiente:

$$\begin{cases} y_0 \text{ dado,} \\ y_1, y_2, y_3 \leftarrow \text{RK4} \\ \hat{y}_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3} \leftarrow \text{AB4} \\ y_{k+1} = y_k + \frac{h}{24}[9f(t_{k+1}, \hat{y}_{k+1}) + 19f_k - 5f_{k-1} + f_{k-2}] \leftarrow \text{AM3} \end{cases}$$

8.4. Implementación de los métodos con MATLAB

Euler hacia adelante

```
function [M] = Forward_Euler(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%                y0 ---- valor inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i

h = (b-a)/N;
M(1,1) = a; M(1,2) = y0;

for i=1:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ h*f(M(i,1), M(i,2));
end
end
```

Método RK2

```
function[M] = metodoRK2(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i

h = (b-a)/N;
M(1,1) = a; M(1,2) = y0;

for i=1:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ h*f(M(i,1)+h/2, M(i,2)+h/2*f(M(i,1), M(i,2))
       );
end
end
```

Método RK3

```
function [M]= metodoRK3(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Inputs:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- valor inicial
%             N ---- numero de subintervalos
% Output:     M ---- matriz con aproximaciones y_i en los puntos
   t_i

h = (b-a)/N;
```

```
M(1,1) = a; M(1,2) = y0;

for i=1:N
    M(i+1,1) = a+i*h;
    k1 = h*f(M(i,1), M(i,2));
    k2 = h*f(M(i,1) + h/2, M(i,2) + k1/2);
    k3 = h*f(M(i+1,1), M(i,2) + 2*k2-k1);
    M(i+1,2) = M(i,2)+ (1/6)*(k1+4*k2+k3);
end
end
```

Método RK4

```
function [M]= metodoRK4(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Inputs:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- valor inicial
%             N ---- numero de subintervalos
% Output:     M ---- matriz con aproximaciones y_i en los puntos
   t_i

h = (b-a)/N;
M(1,1) = a; M(1,2) = y0;

for i=1:N
    M(i+1,1) = a+i*h;
    k1 = h*f(M(i,1), M(i,2));
    k2 = h*f(M(i,1) + h/2, M(i,2) + k1/2);
    k3 = h*f(M(i,1) + h/2, M(i,2) + k2/2);
    k4 = h*f(M(i+1,1), M(i,2) + k3);
    M(i+1,2) = M(i,2)+ (1/6)*(k1+2*k2+2*k3+k4);
end
end
```

Método AB2

```
function[M] = AB2(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Bashford de 2 pasos, utilizando
% el metodo de RK4 para aproximar las condiciones iniciales

% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = metodoRK4(f,a,b,y0,N); % se puede cambiar por RK2 o RK3

M(1,1) = a;      M(1,2) = y0;
M(2,1) = a+h;    M(2,2) = R(2,2);


for i = 2:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/2)*(3*f(M(i,1), M(i,2))-f(M(i-1,1), M(i
       -1,2)));
end
end
```

Método AB3

```
function[M] = AB3(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Bashford de 3 pasos, utilizando
% el metodo de RK para aproximar las condiciones iniciales

% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
```

```
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = metodoRK3(f,a,b,y0,N); % se puede cambiar por otro RK

M(1,1) = a;      M(1,2) = y0;
M(2,1) = a+h;    M(2,2) = R(2,2);
M(3,1) = a+2*h; M(3,2) = R(3,2);


for i = 3:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/12)*( 23*f(M(i,1), M(i,2))...
        -16*f(M(i-1,1), M(i-1,2))+5*f(M(i-2,1), M(i-2,2)));
end
end
```

Método AB4

```
function[M] = AB4(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Bashford de 4 pasos, utilizando
% el metodo de RK4 para aproximar las condiciones iniciales

% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
```

```
R = metodoRK4(f,a,b,y0,N);

M(1,1) = a;      M(1,2) = y0;
M(2,1) = a+h;    M(2,2) = R(2,2);
M(3,1) = a+2*h;  M(3,2) = R(3,2);
M(4,1) = a+3*h;  M(4,2) = R(4,2);

for i = 4:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/24)*( 55*f(M(i,1), M(i,2))...
        -59*f(M(i-1,1), M(i-1,2))+37*f(M(i-2,1), M(i-2,2))...
        -9*f(M(i-3,1), M(i-3,2)) );
end
end
```

Método AM2

```
function[M] = AM2(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Moulton de 2 pasos, utilizando
% el metodo de RK4 para aproximar las condiciones iniciales
% para predecir w_(i+1) en cada iteracion se usa RK4
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = metodoRK4(f,a,b,y0,N);

M(1,1) = a;      M(1,2) = y0;
M(2,1) = a+h;    M(2,2) = R(2,2);

for i = 2:N
```

```
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/12)*( 5*f(M(i+1,1), R(i+1,2))...
    +8*f(M(i,1), M(i,2))-f(M(i-1,1), M(i-1,2)) );
end
end
```

Método AM3

```
function[M] = AM3(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Moulton de 3 pasos, utilizando
% el metodo de RK4 para aproximar las condiciones iniciales
% para predecir w_(i+1) en cada iteracion se usa RK4
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = metodoRK4(f,a,b,y0,N);

M(1,1) = a;     M(1,2) = y0;
M(2,1) = a+h;   M(2,2) = R(2,2);
M(3,1) = a+2*h; M(3,2) = R(3,2);

for i = 3:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/24)*( 9*f(M(i+1,1), R(i+1,2))...
    +19*f(M(i,1), M(i,2))-5*f(M(i-1,1), M(i-1,2))...
        + f(M(i-2,1), M(i-2,2)) );
end
end
```

Método AM4

```
function[M] = AM4(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo de Adams-Moulton de 4 pasos, utilizando
% el metodo de RK4 para aproximar las condiciones iniciales
% para predecir w_(i+1) en cada iteracion se usa RK4
% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = metodoRK4(f,a,b,y0,N);

M(1,1) = a;     M(1,2) = y0;
M(2,1) = a+h;   M(2,2) = R(2,2);
M(3,1) = a+2*h; M(3,2) = R(3,2);
M(4,1) = a+3*h; M(4,2) = R(4,2);

for i = 4:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/720)*( 251*f(M(i+1,1), R(i+1,2))...
    +646*f(M(i,1), M(i,2))-264*f(M(i-1,1), M(i-1,2))...
        +106*f(M(i-2,1), M(i-2,2))-19*f(M(i-3,1), M(i-3,2)) );
end
end
```

Método PC4

```
function [M] = PC4(f,a,b,y0,N)
% este metodo aproxima la solucion a la ecuacion diferencial
% usando el metodo PC4 = RK4+AB4+AM3

% y' = f(t,y), a<= t <= b con condicion inicial y(a) = y0
```

```
% Entradas:  f(t,y) --- funcion
%             [a,b] --- intervalo
%             y0 ---- condicion inicial
%                 N ---- numero de subintervalos
% Salidas:        M ---- matriz con aproximaciones en los puntos
   t_i
h = (b-a)/N;
R = RK4(f,a,b,y0,N);

M(1,1) = a;      M(1,2) = y0;
M(2,1) = a+h;    M(2,2) = R(2,2);
M(3,1) = a+2*h;  M(3,2) = R(3,2);
M(4,1) = a+3*h;  M(4,2) = R(4,2);

for i = 4:N
    M(i+1,1) = a+i*h;
    M(i+1,2) = M(i,2)+ (h/24)*( 55*f(M(i,1), M(i,2))...
        -59*f(M(i-1,1), M(i-1,2))+37*f(M(i-2,1), M(i-2,2))...
        -9*f(M(i-3,1), M(i-3,2)) );

     M(i+1,2) = M(i,2)+ (h/24)*( 9*f(M(i+1,1), M(i+1,2))...
    +19*f(M(i,1), M(i,2))-5*f(M(i-1,1), M(i-1,2))...
        + f(M(i-2,1), M(i-2,2)) );
end
end
```

8.5. Ejercicios

8.5.1. Funciones tipo Lipschitz. Iteración de Picard

1. Muestre que la función $f(t,y) = y + e^{-y} + e^{-t}$ es Lipschitz continua en la variable y en el conjunto $D = \{(t,y) \in \mathbb{R}^2 : 0 \leq t \leq 1,\ 0 \leq y \leq +\infty\}$.

2. Verifique que la función $f(t,y) = e^{-t^2}\arctan(y)$ satisface la condición de Lipschitz en la segunda variable para $t \in [1, +\infty[$. Encuentre la constante de Lipschitz.

3. Considere el problema

$$y'(t) = nty, \qquad y(0) = 1$$

La iteración de Picard está dada por

$$\begin{cases} \varphi_0(t) &= y_0 \\ \varphi_{k+1} &= y_0 + \displaystyle\int_{t_0}^{t} f(x, \varphi_k(x))\, dx \end{cases}$$

Muestre que

$$\varphi_k(t) = \sum_{j=0}^{k} \frac{n^j \cdot t^{2j}}{2^j \cdot j!}$$

y concluya que

$$y(t) = \lim_{k \to +\infty} \varphi_k(t) = e^{nt^2/2}$$

8.5.2. Métodos de un paso. Runge–Kutta. Multipasos

1. Aplique el método de Euler hacia adelante para

$$\begin{cases} y' &= \cos(2t) + \operatorname{sen}(2t), \\ y(0) &= 1 \end{cases}$$

en el intervalo $[0, 1]$ utilizando 5 puntos. Determine el error relativo de cada aproximación. Mejore la aproximación aplicando el método predictor–corrector de Heun. Determine el error relativo de cada aproximación con este último método.

2. Aproxime $\int_0^1 e^{-x^2}\, dx$ utilizando el método RK3, con $h = 0.25$

3. Un proyectil de masa $m = 0.1$ kg que es lanzado verticalmente hacia arriba con una velocidad inicial de $v(0) = 8$ m/s, disminuye su velocidad por el efecto de la gravedad y además por la resistencia del aire, y se modela con la ecuación diferencial

$$m\frac{dv}{dt} = -9.8m - 0.002v \cdot |v|$$

Se sabe que dicho problema tiene solución única en $D = \{(t, v) \in [0,\, 0.9] \times [-2,\, 8]\}$. Use el método de Runge-Kutta 4, con paso $h = 0.3$, para aproximar la velocidad en $t = 0.9$

4. Considere el problema de Cauchy:

$$\begin{cases} y' - 3\displaystyle\int_0^t e^{-2(t-x)} y(x)\, dx &= 4e^{-2t}, \quad \text{para } t \in [0,\, 1] \\ y(0) &= 4 \end{cases}$$

a) Use la regla del trapecio para aproximar la integral $\int_0^t e^{-2(t-w)}y(w)\,dw$.

b) Reescriba el problema con base en la información del inciso a) y utilice el método RK4 con paso $h = 0.25$ para determinar una aproximación de $y(0.75)$.

c) Resuelva el problema de manera exacta (use transformada de Laplace) y compare el valor exacto $y(0.75)$ con el valor aproximado del inciso b).

5. Muestre que el método de Adams-Bashfort de tres pasos corresponde a

$$y_{k+1} = y_k + \frac{h}{12}\left[23f_k - 16f_{k-1} + 5f_{k-2}\right]$$

6. Encuentre una aproximación para $y(1)$, donde

$$\begin{cases} y' &= -5y + 5t^2 + 2t, \\ y(0) &= \frac{1}{2}. \end{cases}$$

utilizando el método PC4 con paso $h = 0.2$

7. Elabore un script en donde se utilice el método de predictor-corrector con el método de Euler hacia atrás y el método de Euler hacia adelante, es decir

$$\begin{cases} y_0 \text{ dado}, & \\ \hat{y}_{k+1} &= y_k + hf(t_k, y_k), \\ y_{k+1} &= y_k + hf(t_{k+1}, \hat{y}_{k+1}) \end{cases}$$

8. Implemente el método PC4 en `MATLAB` y compruebe los cálculos que realizó en el ejercicio 6.

9. Aplique el método de predictor–corrector de Heun hacia adelante para

$$\begin{cases} y' &= \cos(2t) + \operatorname{sen}(2t), \\ y(0) &= 1 \end{cases}$$

en el intervalo $[0, 1]$ utilizando 5 puntos. Grafique todo junto: los nodos y la gráfica de la función exacta en `MATLAB`, así como el polinomio interpolador de Lagrange que contiene los nodos (t_k, y_k) del método de Heun. (Este es parte del problema 1)

10. Considere el siguiente problema de valor inicial:

$$\begin{cases} y' &= \arctan(y) - te^t, \quad \text{para } -1.2 \leq t \leq 1.2 \\ y(-1.2) &= 2 \end{cases}$$

Aplique el método PC4 para aproximar la solución del problema de valor inicial con $h = 0.4$

11. Considere el problema de valor inicial:

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{y^3-2t}{y^3-1}\right) &= t^2 - \left(\dfrac{y^3-2t}{y^3-1}\right)^2, \quad \text{para } 0 \le t \le 1, \\ y(0) &= \sqrt[3]{2}. \end{cases}$$

a) Determine la función $f(t, y)$ en la ecuación diferencial anterior.

b) Utilizando un cambio de variable adecuado, muestre que el problema anterior se puede escribir de la forma:

$$\begin{cases} w' &= t^2 - w^2, \quad \text{para } 0 \le t \le 1, \\ w(0) &= 2. \end{cases}$$

c) Utilice el método de Euler modificado (RK2) con $h = 0.25$, para aproximar la solución del problema del inciso b).

d) Con los resultados obtenidos en c) encuentre una aproximación para $y(1)$.

Bibliografía

[1] Bornemann, F. *Numerical Linear Algebra. A concise introduction with* MATLAB *and Julia*, Springer, Munich, 2018.

[2] Burden, R., Faires, D. *Análisis numérico*. Cengage Learning, Novena Edición, México, 2011.

[3] Calvo, J. MA-0501 *Análisis Numérico I. Notas del curso*. Universidad de Costa Rica, 2020.

[4] Chapra, S., Canale, R. *Métodos numéricos para ingenieros*. McGraw–Hill, Sétima Edición, México, 2014.

[5] Chavarría, J. *Métodos numéricos*. Editorial Tecnológica de Costa Rica, Cartago, 2014.

[6] Echevarría, R. *Apuntes de* MATLAB *orientados a métodos numéricos elementales*. Departamento de Ecuaciones Diferenciales y Análisis Numérico, Universidad de Sevilla, 2020.

[7] Faul, A. C. *A Concise Introduction to Numerical Analysis*. CRC Press, Florida, 2016.

[8] Karris, S. *Numerical Analysis Using* MATLAB *and Excel*. Orchard Publications, Third Edition, California, 2007.

[9] Kincaid, D., Cheney, W. *Numerical Analysis. Mathematics of Scientific Computing*. AMS, Rhode Island, 2002.

[10] Mathews, J. y Fink, K. *Métodos numéricos con* MATLAB. Prentice Hall, Tercera edición, Madrid, 2000.

[11] Sastry, S. *Introductory methods of Numerical Analysis*, PHI Privated Limited, Nueva Delhi, 2006.

[12] Sequeira F. Segura E., Salas O., *Introducción a* MATLAB. Editorial Universidad de Costa Rica, 2020.

[13] Süli, E.; Mayers, D.F., *An Introduction to Numerical Analysis*, Cambridge University Press, 2006.