

# ERRORES

---

## Indice

1. Errores
  2. Clases de errores
  3. Números en coma flotante
  4. Aritmética del punto flotante
    - 4.1. Errores
    - 4.2. Operaciones en punto flotante
    - 4.3. Problemas con operaciones en punto flotante
  5. Algoritmos o métodos numéricos
    - 5.1. Convergencia y velocidad de convergencia
    - 5.2. Estabilidad numérica
    - 5.3. Coste operativo y eficiencia
  6. Conclusiones
- 

## 1 Errores

**Definición 1.1.** Si  $p^*$  es una aproximación de  $p$ , el *error absoluto* es  $\Delta p = |p - p^*|$ , y el *error relativo* es  $\epsilon_p = \frac{|p-p^*|}{|p|}$ , siempre que  $p \neq 0$ .

Normalmente no se conoce  $p$  y, por tanto, tampoco se conocerá con exactitud el error absoluto (ni el relativo) de tomar  $p^*$  como una aproximación de  $p$ . Por tanto, se pretende encontrar cotas superiores de esos errores. Cuanto más pequeñas sean esas cotas superiores, mejor será la aproximación.

**Definición 1.2.** Se dice que el número  $p^*$  *aproxima a*  $p \neq 0$  con  $t$  *dígitos* (o *cifras*) *significativos* (exactos), si  $t$  es el mayor entero no negativo para el cual  $\frac{|p-p^*|}{|p|} < 5 \cdot 10^{-t}$  o sea,  $5 \cdot 10^{-t}$  es una *cota superior del error relativo*.

**Ejemplo 1.3.** Cota superior de  $|p - p^*|$  para distintos valores de  $p$  tomando cuatro dígitos significativos para  $p^*$ :

$p$	0.1	0.5	100	1000
$\max p - p^* $	0.00005	0.00025	0.05	0.5

Así para  $p = 0.1$  se tiene:

$$\frac{|0.1 - p^*|}{|0.1|} < 5 \cdot 10^{-4}$$

luego,  $\max|p - p^*| < 0.1 \cdot 5 \cdot 10^{-4}$  y así,  $0.1 - 0.00005 < p^* < 0.1 + 0.00005$ , o sea:  $0.09995 < p^* < 0.10005$ .

Y para  $p = 1000$ , se tiene  $999.5 < p^* < 1000.5$ .

## 2 Clases de errores

### A) Errores en las entradas. Propagación.

Los errores en las entradas o datos iniciales suelen ser consecuencia del hecho de que nuestros datos provienen de un experimento y son inherentes a la imperfección de las medidas físicas; o bien, se producen al tomar (de forma voluntaria) un número limitado de dígitos para los datos de entrada.

A la hora de valorar el resultado obtenido en cualquier método numérico es importante conocer la magnitud de dichos errores y cómo se propagan.

#### Propagación del error

Sea  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$  la función que nos define el cálculo a realizar, donde  $f$  viene dada por  $y_i = f_i(x_1, x_2, \dots, x_n)$ , para  $i = 1, \dots, m$ . Supongamos que  $f$  es de clase  $\mathcal{C}^1(D)$ . Sea  $\Delta x = |x - x^*|$ , de forma que  $\Delta x_i = |x_i - x_i^*|$ . Si reemplazamos los datos de entrada  $x$  por los datos aproximados  $x^*$ , el resultado de nuestro cálculo será  $y^* = f(x^*)$ , en lugar de  $y = f(x)$ .

Aplicando Taylor, se tiene:

$$|y_i - y_i^*| = |f_i(x) - f_i(x^*)| \approx \sum_{j=1}^n |x_j - x_j^*| \cdot \left| \frac{\partial f_i(x)}{\partial x_j} \right|.$$

Así, una forma de aproximar el efecto que tienen los datos de entrada en la salida (en términos de errores absolutos) es:

$$\Delta y_i \approx \sum_{j=1}^n \left| \frac{\partial f_i(x)}{\partial x_j} \right| \cdot \Delta x_j.$$

Y, en terminos de errores relativos:

$$\epsilon_{y_i} = \left| \frac{\Delta y_i}{y_i} \right| \approx \sum_{j=1}^n \left| \frac{x_j}{f_i(x)} \frac{\partial f_i(x)}{\partial x_j} \right| \cdot \epsilon_{x_j}$$

Al valor de  $\left| \frac{x_j}{f_i(x)} \frac{\partial f_i(x)}{\partial x_j} \right|$  se le llama *número de condición*.

Si aparece algún número de condición grande, decimos que estamos ante un *problema mal condicionado*, en otro caso hablamos de un *problema bien condicionado*.

## B) Errores de truncamiento o discretización

Los errores de truncamiento o discretización provienen, por ejemplo, de la sustitución de una expresión continua por otra discreta (por ejemplo al aproximar la derivada de  $f$  por una expresión en diferencias),

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

Usando el desarrollo de Taylor de  $f$ :

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2!}f''(x_0)h^2 + \frac{1}{3!}f'''(x_0)h^3 + \dots$$

De donde,

$$\frac{f(x_0 + h) - f(x_0)}{h} = f'(x_0) + \frac{1}{2!}f''(x_0)h + \frac{1}{3!}f'''(x_0)h^2 + \dots$$

Así el error cometido es

$$\frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) = \frac{1}{2!}f''(x_0)h + \frac{1}{3!}f'''(x_0)h^2 + \dots = O(h)$$

Otro caso donde aparecen errores de truncamiento es al aproximar un proceso infinito por uno finito (por ejemplo, truncando los términos de una serie). Sea  $(a_n)$  una sucesión de términos no negativos, monótona decreciente y con  $\lim_{n \rightarrow \infty} a_n = 0$ , por el criterio de Leibnitz, sabemos que si

$$S = \sum_{n=1}^{+\infty} (-1)^n a_n \quad \text{y} \quad S_k = \sum_{n=1}^k (-1)^n a_n$$

se tiene:

$$\Delta S = |S - S_k| \leq a_{k+1}.$$

## C) Errores de redondeo

Si trabajamos con ordenadores, los números reales se representan mediante aproximaciones a dichos números (con un número finito de dígitos), lo cual produce un *Error de redondeo*.

### 3 Números en coma flotante

Todo número real se representa en el ordenador con un número finito de dígitos. Así  $\sqrt{3}$  no tiene una representación exacta. Se representa mediante una aproximación.

Casi siempre la representación y la aritmética computacional son satisfactorias y pasan inadvertidas.

$\mathbb{R}$  se representa mediante un conjunto finito de números racionales que se representan como *números en punto flotante* o *números máquina*.

Un número en punto flotante tiene la forma  $\pm m \cdot b^c$ , donde  $b^{-1} \leq m < 1$  ó  $1 \leq m < b$ .

En un número de punto flotante podemos distinguir:

- Signo
- Base ( $b$ )
- Exponente o característica ( $c$ )
- Mantisa o parte fraccionaria ( $m$ )

Por ejemplo, en notación científica normalizada en el sistema decimal (*números máquina decimales*)  $x = \pm r \cdot 10^n$  donde  $\frac{1}{10} \leq r < 1$  (ó  $1 \leq r < 10$ ) (si  $x \neq 0$ ) y  $n \in \mathbb{Z}$ . Así:

$$\begin{aligned}732.5051 &= 0.7325051 \cdot 10^3 (= 7.325051 \cdot 10^2) \\ -0.005612 &= -0.5612 \cdot 10^{-2} (= 5.612 \cdot 10^{-3})\end{aligned}$$

En notación científica en el sistema binario  $x = \pm q \cdot 2^m$  donde  $\frac{1}{2} \leq q < 1$  (ó  $1 \leq q < 2$ ) (si  $x \neq 0$ ) y  $m \in \mathbb{Z}$ . Así:

$$10011.101001_{(2)} = 0.10011101001_{(2)} \cdot 2^5 (= 1.0011101001_{(2)} \cdot 2^4)$$

**Ejemplo 3.1.** En 1985, el IEEE (Institute for Electrical and Electronic Engineers, <http://standards.ieee.org>) publicó la *Norma de la aritmética binaria de punto flotante 754* (estándar IEEE 754 de doble precisión).

Se representa con 64 bits y consta de:

- 1 bit de signo (0 positivo, 1 negativo) ( $s$ )
- 11 bits de exponente ( $c$ ) en exceso 1023  $\Rightarrow c - 1023$  es el exponente
- 52 bits de mantisa ( $f$ ) que se utiliza en la forma  $(1 + f)$ .
- la base es 2.

Se obtiene

$$(-1)^s \cdot 2^{c-1023} \cdot (1 + f)$$

Tiene entre 15 y 16 dígitos decimales de precisión y un rango aproximado entre  $10^{-308}$  y  $10^{308}$ .

## 4 Aritmética del punto flotante

Por simplicidad, en lo que sigue, supongamos números máquina decimales:

$$\pm 0.d_1d_2\dots d_k \cdot 10^n \text{ con } 1 \leq d_1 \leq 9, 0 \leq d_i \leq 9; i = 2, \dots, k$$

Se puede normalizar cualquier número real positivo  $y$  para convertirlo en

$$y = 0.d_1d_2\dots d_kd_{k+1}d_{k+2}\dots \cdot 10^n$$

Si  $y$  está en el rango de la máquina puede ponerse en forma de punto flotante,  $fl(y)$ , tomando  $k$  dígitos decimales. Hay dos maneras de elegir esos  $k$  dígitos decimales: *corte* y *redondeo*.

- En el corte simplemente se eliminan todos los dígitos a partir del  $d_{k+1}$ .
- En el redondeo se agrega  $5 \cdot 10^{n-(k+1)}$  a  $y$  y luego se realiza el corte. O sea,

si  $d_{k+1} \geq 5$  agregamos 1 a  $d_k$  y cortamos,

si  $d_{k+1} < 5$  simplemente cortamos.

**Ejemplo 4.1.** El número  $\pi$  es un número irracional, luego tiene una expresión decimal infinita de la forma  $\pi = 3.14159265\dots$ . En forma decimal normalizada es

$$\pi = 0.314159265\dots \cdot 10^1$$

El número  $\pi$  en punto flotante con cinco dígitos y corte se representa por

$$fl(\pi) = 0.31415 \cdot 10^1 = 3.1415$$

Con redondeo será

$$fl(\pi) = (0.31415 + 0.00001) \cdot 10^1 = 3.1416$$

El error cometido al reemplazar un número por su forma en punto flotante recibe el nombre de *error de redondeo* independientemente de si se ha aplicado el método de corte o de redondeo.

### 4.1 Errores

**Proposición 4.2.** *El error absoluto en la representación de un número  $p \neq 0$  en punto flotante con  $t$  decimales, viene dado por:*

$$|p - fl(p)| \leq 5 \cdot |p| \cdot 10^{-t} \cdot k$$

con  $k = 2$  si es por corte y  $k = 1$  si es por redondeo.

*Demostración.* i) Caso de corte:

$$\begin{aligned}
 & |p - fl(p)| = \\
 & = | \pm 10^q \cdot (0.d_1d_2 \dots d_t d_{t+1} \dots) - (\pm 10^q \cdot (0.d_1d_2 \dots d_t)) | = \\
 & = 10^{q-t} \cdot (0.d_{t+1}d_{t+2} \dots) = \frac{10^{q-t} \cdot (0.d_{t+1}d_{t+2} \dots)}{10^q \cdot (0.d_1d_2 \dots)} |p| = \\
 & = 10^{-t} \cdot \frac{0.d_{t+1}d_{t+2} \dots}{0.d_1d_2 \dots} \cdot |p| \leq \\
 & \leq 10^{-t} \cdot \frac{1}{0.1} \cdot |p| = 10^{-t} \cdot |p| \cdot 5 \cdot k \\
 & \quad (\text{con } k = 2)
 \end{aligned}$$

ii) Caso de redondeo:

$$\begin{aligned}
 & |p - fl(p)| \leq 10^q \cdot 10^{-t} \cdot \frac{1}{2} = \\
 & = 10^{q-t} \cdot \frac{1}{2} \cdot \frac{|p|}{|p|} = 10^q \cdot 10^{-t} \cdot \frac{1}{2} \cdot \frac{|p|}{10^q \cdot (0.d_1d_2 \dots)} = \\
 & = \frac{1}{2} \cdot 10^{-t} \cdot \frac{|p|}{(0.d_1d_2 \dots)} \leq \frac{1}{2} \cdot 10^{-t} \cdot \frac{|p|}{0.1} = \\
 & 0.5 \cdot 10 \cdot 10^{-t} \cdot |p| = 5 \cdot 10^{-t} \cdot |p| \cdot k \\
 & \quad (\text{con } k = 1)
 \end{aligned}$$

□

**Corolario 4.3.** *El error relativo en la representación de un número  $p \neq 0$  en punto otante con  $t$  decimales viene dado por:*

$$\frac{|p - fl(p)|}{|p|} \leq 5 \cdot 10^{-t} \cdot k$$

(con  $k = 2$  si es por corte y  $k = 1$  si es por redondeo). Luego, si es por redondeo,  $fl(p)$  aproxima a  $p$  con  $t$  dígitos significativos.

## 4.2 Operaciones en punto flotante

Representamos las operaciones en la computadora mediante:  $\oplus$ ,  $\ominus$ ,  $\otimes$ ,  $\oslash$ . Supondremos una aritmética de dígitos finitos (equivale a realizar operaciones exactas sobre representaciones de punto otante, y luego convertir el resultado exacto en su representación de punto flotante), viene definida por:

$$x \oplus y = fl(fl(x) + fl(y))$$

$$x \ominus y = fl(fl(x) - fl(y))$$

$$x \otimes y = fl(fl(x) \times fl(y))$$

$$x \oslash y = fl(fl(x)/fl(y))$$

**Ejemplo 4.4.** Sean  $x = 1/3$  e  $y = 5/7$ . En corte a cinco dígitos tenemos

$$fl(x) = 0.33333 \cdot 10^0; \quad fl(y) = 0.71428 \cdot 10^0$$

<i>Oper.</i>	<i>Resultado</i>	<i>Error abs.</i>	<i>Error rel.</i>
$x \oplus y$	$0.10476 \cdot 10^1$	$0.190 \cdot 10^{-5}$	$0.182 \cdot 10^{-4}$
$x \ominus y$	$-0.38095 \cdot 10^0$	$0.238 \cdot 10^{-5}$	$0.625 \cdot 10^{-5}$
$x \otimes y$	$0.23809 \cdot 10^0$	$0.524 \cdot 10^{-5}$	$0.220 \cdot 10^{-4}$
$x \oslash y$	$0.46666 \cdot 10^0$	$0.667 \cdot 10^{-5}$	$0.143 \cdot 10^{-4}$

### 4.3 Problemas con operaciones en punto flotante

Comencemos con algunos ejemplos de operaciones en coma flotante, para luego indicar algunos problemas que se observan en los ejemplos

$$\begin{aligned}
 y = 5/7 & \Rightarrow fl(y) = 0.71428 \cdot 10^0 \\
 u = 0.714251 & \Rightarrow fl(u) = 0.71425 \cdot 10^0 \\
 v = 98765.9 & \Rightarrow fl(v) = 0.98765 \cdot 10^5 \\
 w = 0.111111 \cdot 10^{-4} & \Rightarrow fl(w) = 0.11111 \cdot 10^{-4}
 \end{aligned}$$

<i>Oper.</i>	<i>Error abs.</i>	<i>Error rel.</i>
$y \ominus u$	$0.472 \cdot 10^{-5}$	0.136
$(y \ominus u) \oslash w$	0.425	0.136
$(y \ominus u) \otimes v$	0.466	0.136
$u \oplus v$	$0.162 \cdot 10^1$	$0.164 \cdot 10^{-4}$
$y \ominus w$	0.779	$0.122 \cdot 10^{-4}$

#### Problemas:

- La división por un número muy pequeño (o la multiplicación por un número muy grande) da un error absoluto muy grande. Si  $p$  lo representamos por  $p^* + \epsilon$ , y calculamos  $p/d$ , donde  $d$  es pequeño, se obtendrá un error aproximado de  $\epsilon/d$  y, si  $d$  es muy pequeño, el cociente será muy grande.
- La sustracción de números casi iguales da errores relativos muy grandes: Sean  $x, y$  (con  $x > y$ ) tales que (con  $k$  dígitos):

$$fl(x) = 0.d_1d_2 \dots d_p \cdot \alpha_{p+1} \dots \alpha_k \cdot 10^n$$

$$fl(y) = 0.d_1d_2 \dots d_p \cdot \beta_{p+1} \dots \beta_k \cdot 10^n$$

$$fl(fl(x) - fl(y)) = 0.\gamma_{p+1} \dots \gamma_k \cdot 10^{n-p}$$

donde  $0.\alpha_{p+1} \dots \alpha_k - \beta_{p+1} \dots \beta_k = 0.\gamma_{p+1} \dots \gamma_k$ . Así  $x \ominus y$  tendrá  $k - p$  dígitos.

La pérdida de dígitos es peligrosa si se quiere mantener un error relativo pequeño. Recordemos que  $p^*$  aproxima a  $p$  con  $t$  dígitos significativos si:

$$\frac{|p - p^*|}{|p|} < 5 \cdot 10^{-t}$$

- La propagación de estos errores al resto de cálculos.

Podemos evitar estos problemas:

- minimizando el número de operaciones,
- ordenando adecuadamente las operaciones,
- replanteando el problema en otros términos.

**Ejemplo 4.5.** Resolución de  $x^2 + 62.10 \cdot x + 1 = 0$ .

Las raíces aproximadas son

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = -0.0161072$$

$$x_2 = -\frac{-b - \sqrt{b^2 - 4ac}}{2a} = -62.08390$$

Como  $b^2$  es mucho mayor que  $4ac$  en el cálculo de  $x_1$  el numerador tiene una resta de números casi iguales.

Trabajando con redondeo a cuatro dígitos tenemos que

$$\sqrt{b^2 - 4ac} = \sqrt{3856 - 4} = 62.07$$

$$fl(x_1) = \frac{-62.10 + 62.06}{2} = -0.15 \cdot 10^{-1}$$

y se obtiene un error relativo grande

$$\frac{|-0.0161072 + 0.015|}{|0.0161072|} \approx 6.9 \cdot 10^{-1}$$

El cálculo de  $x_2$  no presenta ese problema (no aparece la resta).

*Solución:* hacer desaparecer la resta en el cálculo de  $x_1$ .

$$x_1 = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})} = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

$$fl(x_1) = -0.1611 \cdot 10^{-1}$$

y el error relativo es ahora:

$$\frac{|-0.0161072 + 0.1611 \cdot 10^{-1}|}{|0.0161072|} \approx 2 \cdot 10^{-4}$$

En el caso de que  $b$  fuera negativo las cosas ocurrirían exactamente al contrario. Para  $x_1$  obtendríamos una buena aproximación y para  $x_2$  habría que considerar la segunda opción.

**Ejemplo 4.6.** Evaluación del polinomio  $P(x) = x^3 - 6.1x^2 + 3.2x + 1.5$  en el punto  $x = 4.71$  usando aritmética de tres dígitos.

	$x$	$x^2$	$x^3$	$6.1x^2$	$3.2x$
<i>Exacto</i>	4.71	22.1841	104.487	135.323	15.072
<i>Corte</i>	4.71	22.1	104.	135.	15.0
<i>Redondeo</i>	4.71	22.2	104.	135.	15.1

Exacto :  $P(4.71) = 104.487 - 135.323 + 15.072 + 1.5 = -14.2639$

Corte a 3 dígitos:  $P(4.71) = ((104. - 135.) + 15.0) + 1.5 = -14.5$

Redondeo a 3 dígitos:  $f(4.71) = ((104. - 135.) + 15.1) + 1.5 = -14.4$

Errores relativos (con 3 dígitos):

En corte:  $\frac{-14.2639 - (-14.5)}{-14.2639} \approx 0.017.$

En redondeo:  $\frac{-14.2639 - (-14.4)}{-14.2639} \approx 0.00962.$

*Solución:* evaluar  $P(x)$  en forma *anidada*:  $P(x) = ((x - 6.1)x + 3.2)x + 1.5.$

Así la evaluación de  $P(x)$  en 4.71 queda:

En corte:

$$P(4.71) = -14.2$$

con un error relativo:

$$\frac{-14.2639 + 14.2}{-14.2639} \approx 0.0045$$

En redondeo:

$$P(4.71) = -14.3$$

con un error relativo:

$$\frac{-14.2639 + 14.3}{-14.2639} \approx 0.0026$$

## 5 Algoritmos o métodos numéricos

Un algoritmo es un procedimiento que describe, sin ninguna ambigüedad, una sucesión finita de pasos a realizar en un orden específico. Si estos pasos consisten en operaciones aritmético-lógicas conducentes a la solución numérica de un cierto problema, el algoritmo será llamado numérico o también *método numérico*.

Hay dos tipos de métodos: métodos directos y métodos iterativos.

### A) Métodos directos

Son aquellos que, al menos teóricamente y supuesta la ausencia de errores, en un número finito de pasos conducen a la solución exacta del problema. Por ejemplo: la regla de Cramer para la solución de un sistema lineal compatible determinado.

### B) Métodos iterativos

Son métodos que utilizan sucesiones, que nos proporcionan aproximaciones cada vez mejores de la solución buscada.

Por ejemplo, usemos el desarrollo en serie del número  $e$  para aproximar su valor.

$$e = \sum_{n=0}^{+\infty} \frac{1}{n!}$$

se tiene:

$$S_0 = 1, S_1 = 1 + 1, S_2 = 1 + 1 + \frac{1}{2!}, S_n = 1 + 1 + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

que proporcionarán aproximaciones a  $e$ , cuanto mayor sea  $n$ .

### 5.1 Convergencia y velocidad de convergencia

**Definición 5.1.** La aplicación de un método iterativo conduce a una sucesión  $(s_n)$  de aproximaciones de la solución  $s$  buscada. Se dice que el método es *convergente* si dicha sucesión es convergente a la solución del problema sobre el que se aplica.

**Definición 5.2.** La *velocidad u orden de convergencia* de un método iterativo convergente  $((s_n) \rightarrow s)$  es el mayor número real  $q$ , tal que existe el límite

$$\lim_{n \rightarrow \infty} \frac{|s_{n+1} - s|}{|s_n - s|^q} = C \neq 0$$

No es necesario que tal  $q$  exista, y si existe, no necesariamente es un entero.

- Si  $q = 1$ , decimos que la convergencia es *lineal*.
- Si  $q = 2$ , decimos que la convergencia es *cuadrática*.
- Si  $q > 1$ , decimos que la convergencia es *superlineal*.

A la constante  $C$  se le suele llamar *constante de error asintótico*.

## 5.2 Estabilidad numérica

Diremos que un método numérico es **inestable** cuando pequeños errores en algunas de sus etapas generan, a lo largo del resto del proceso, errores que degradan seriamente la exactitud de los resultados del cálculo en su conjunto.

Sea por ejemplo,

$$I_n = \int_0^1 x^n \cdot e^{x-1} dx \quad \forall n \in \mathbb{N}$$

Aplicando integración por partes, se llega a :

$$I_n = 1 - nI_{n-1},$$

para  $n = 2, 3, \dots$ , siendo  $I_1 = 1/e$ .

Si aplicamos la fórmula recurrente anterior y tomando 6 cifras en la representación numérica, se tiene:

$$\begin{array}{lll} I_1 \approx 0.367879 & I_2 \approx 0.264242 & I_3 \approx 0.207274 \\ I_4 \approx 0.170904 & I_5 \approx 0.145480 & I_6 \approx 0.127120 \\ I_7 \approx 0.110160 & I_8 \approx 0.118720 & I_9 \approx -0.0684800?? \end{array}$$

El valor de  $I_9$  no puede ser correcto puesto que para todo  $n$ ,  $I_n$  expresa el área de la región del primer cuadrante delimitada por la función  $x^n e^{x-1}$ , para valores de  $x$  entre 0 y 1. Esta función es positiva en  $[0, 1]$ , luego  $I_n \geq 0$ .

## 5.3 Coste operativo y eficiencia

El número de operaciones elementales que un método necesita en su aplicación se denomina coste operativo y supone una medida de lo que se denomina complejidad del método. Para cada problema suele haber varios métodos posibles, ¿cómo elegir entre ellos?

Una posibilidad sería: elegir el método que diese menos errores. Hay otra posibilidad mejor: quedarse con el método que, dando errores dentro de unos límites predeterminados, necesite el menor trabajo.

Este equilibrio entre precisión y coste operativo se llama *eficiencia*, y diremos que un método es más eficiente que otro si, dando errores parecidos es menos costoso, o que siendo parecido de costoso es más preciso.

## 6 Conclusiones

- Los cálculos numéricos pueden ser inexactos.
- El *error final* de un proceso de cálculo efectivo es fruto de la acumulación de distintos tipos de errores. Unos iniciales (de entrada, redondeo de los datos, truncadura del problema, ...) y otros generados a lo largo del proceso.

- El aumento de la *precisión* (aumento de las unidades de memoria para almacenar los números), en general reduce el error final. No obstante, a veces conviene sacrificar precisión frente a economía de tiempo o de recursos. (Un equilibrio entre precisión suficiente y coste adecuado se le suele llamar *eficiencia*).
- Hay operaciones o procesos de cálculo que propagan fuertemente los errores de redondeo, operaciones o algoritmos inestables (inestabilidad numérica).
- Hay problemas que tienen una naturaleza que los hace especialmente sensibles a la variación de los datos, problemas mal condicionados. Para cierto tipo de problemas se puede definir una medida de esta sensibilidad mediante un número de condición.

**Lectura recomendada:** Capítulo 4 del texto: I. Martín Llorente, V.M. Pérez García: *Cálculo Numérico para computación en Ciencia e Ingeniería*, ed. Síntesis.