Mètodes Numèrics — Grau de Matemàtiques
Errors

Lluís Alsedà

adaptat dels *Apunts de Mètodes Numèrics* de Josep Maria Mondelo, 2009

Departament de Matemàtiques
Universitat Autònoma de Barcelona

http://www.mat.uab.cat/∼alseda

febrer 2016 (versió 1.1.1)

Continguts

Introducció

A qualsevol càlcul numèric és imprescindible controlar l'efecte dels errors. N'hem de considerar de tres tipus:

1. *Errors en les dades d'entrada*, deguts a mesuraments incorrectes o a la finitud de la seva representació a l'ordinador.
2. *Errors a les operacions*, deguts a que emprem aritmètica de punt flotant finita.
3. La major part de mètodes que veurem durant el curs no produeixen la solució exacta del problema que aborden, ni tan sols en el supòsit que poguéssim representar exactament les dades d'entrada i no es produïssin errors en les operacions. Els errors deguts a aquest fet es coneixen com a *errors de truncament*.

En aquest tema tractarem els dos primers tipus d'error. Dels errors de truncament, que són específics de cada mètode, ens ocuparem més endavant.

Representació en punt flotant

El següent teorema garanteix l'existència teòrica de representació (infinita) en punt flotant de qualsevol número real.

Teorema (Representació en punt flotant en base b)

Fixem $b \in \mathbb{N}$, $b \geq 2$. Tot $x \in \mathbb{R}$, $x \neq 0$ pot ser representat de la forma

$$x = s\Big(\sum_{i=1}^{\infty} \alpha_i b^{-i}\Big) b^q, \tag{1}$$

amb $s \in \{-1, 1\}$, $q \in \mathbb{Z}$ i $\alpha_i \in \{0, 1, \ldots, b-1\}$. A més, la representació anterior és única si $\alpha_1 \neq 0$ i els α_i no són tots $b-1$ d'una posició en endavant, i.e.,

$$\forall i_0 \in \mathbb{N} \quad \exists i \geq i_0 \quad : \quad \alpha_i \neq b-1.$$

Representació en punt flotant (cont.)

Escriurem (1) abreujadament com

$$x = s(0.\alpha_1\alpha_2\alpha_3\ldots)_b b^q. \tag{2}$$

El subíndex b dels parèntesis indica que els dígits $\alpha_1\alpha_2\alpha_3\ldots$ es troben en base b.

Exemple (representacions normalitzades)

- $x = -315.487 = -0.315487 \times 10^3$,
- $x = 0.00343434\ldots = +0.\widehat{34} \times 10^{-2}$,
- $x = \frac{100}{3} = +0.333333\cdots \times 10^2 = +0.\widehat{3} \times 10^2$,
- $x = \sqrt{2} = +0.14142135\ldots \times 10^1$ (la representació és infinita però no periòdica).

Exemple (pas a binari)

Suposem $x = (0.1)_{10}$, $b = 2$. Per passar a binari un nombre que només té part decimal, només cal multiplicar-lo per dos, la part entera dóna el primer dígit, quedar-se amb la part decimal, multiplicar-la per dos, la part entera dóna el segon dígit, quedar-se amb la part decimal i així successivament.

$$0.1 \times 2 = 0.2 \longrightarrow \boxed{0}$$
$$0.2 \times 2 = 0.4 \longrightarrow \boxed{0}$$
$$0.4 \times 2 = 0.8 \longrightarrow \boxed{0}$$
$$0.8 \times 2 = 1.6 \longrightarrow \boxed{1}$$
$$0.6 \times 2 = 1.2 \longrightarrow \boxed{1}$$

Com que la part decimal de 1.2 és 0.2, que ja ha sortit, es tornarà a repetir el grup 0011 "ad infinitum". Per tant, la representació binària de 0.1 és

$$(0.1)_{10} = (0.00\widehat{0011})_2 = (0.\widehat{1100})_2 \times 2^{-3}.$$

Noteu que, tot i que la representació decimal de 0.1 és finita, la binària no ho és pas.

Representació en punt flotant (cont.)

La representació numèrica més emprada en ordinadors i calculadores és la coneguda com a

Definició: *Representació en punt flotant*

És la versió finita de la representació (2). En aquesta representació, tot nombre x consta de

- el *signe*, s,
- la *mantissa*, que només consta d'un nombre finit de dígits, $\delta_1, \ldots, \delta_t$, expressats en base b, i
- l'*exponent*, q, que està limitat a un rang prefixat,

$$q_{min} \leq q \leq q_{max}.$$

Representació en punt flotant (cont.)

D'acord amb la notació introduïda a (1), un nombre representat en punt flotant s'escriu

$$x = s(0.\delta_1\delta_2\ldots\delta_t)_b b^q \tag{3}$$

(el subíndex b als parèntesis indica que la base és b).

Abreujadament,

$$x = smb^q,$$

amb

$$m = (0.\delta_1\delta_2\ldots\delta_t)_b = \delta_1 b^{-1} + \delta_2 b^{-2} + \cdots + \delta_t b^{-t}.$$

Direm que un nombre en punt flotant x donat per (3) està *normalitzat* si $\delta_1 \neq 0$ (i.e., $m \geq 1/b$). Llevat de casos excepcionals, considerarem sempre nombres normalitzats.

Representació en punt flotant (cont.)

Les bases més habituals són $b = 2$ (*binari*), $b = 8$ (*octal*) i $b = 16$ (*hexadecimal*), emprades per la pràctica totalitat d'ordinadors, i $b = 10$, emprada per moltes calculadores. Tant el nombre de dígits, t, com el rang d'exponents, q_{min}, q_{max}, es trien segons un compromís entre la quantitat de nombres que volem representar i la quantitat de memòria que cada representació volem que ocupi.

Exemple: Paràmetres d'alguns formats de punt flotant existents

Format	b	t	q_{min}	q_{max}	bits
IEEE simple	2	24	-126	127	32
IEEE doble	2	53	-1022	1023	64
HP–48 real	10	12	-498	500	64
HP–48 real estès	10	15	-49998	50000	84

Representació en punt flotant (cont.)

Observació

La major part d'ordinadors implementen els formats IEEE (Institute of Electric and Electronic Engineering) i, quan és així, els tipus `float` i `double` del llenguatge C corresponen a les aritmètiques IEEE–simple i doble, respectivament.

Observació

Com a exemple de formats de punt flotant de calculadora, s'han pres els de la HP–48. El format real és el que es fa servir a nivell d'usuari, mentre que el real estès només el fa servir internament el sistema operatiu de la calculadora per a operacions intermèdies.

Representació en punt flotant (cont.)

Observació

De fet, fer operacions intermèdies amb precisió superior és una pràctica molt comú, que garanteix que l'únic error que es comet a les operacions en precisions estàndard és del de representació en punt flotant, com suposarem més endavant.

Per exemple, els processadors de la família Intel i386 (8088, 8086, 80286, 80386, 80486 i totes les variants de Pentium de 32 bits) tenen un tipus "real temporal" de 10 bytes (supera lleugerament la precisió IEEE doble).

Els processadors de tipus RISC (com ara les famílies HP PA–RISC i Sun SPARC), emprats en estacions de treball per càlcul científic, acostumen a treballar internament en precisió quàdruple (16 bytes).

IEEE 754 single-precision binary floating-point format: `binary32` (extret de la wikipedia)

The IEEE 754 standard specifies a `binary32` as having:

- **Sign bit:** 1 bit
- **Exponent width:** 8 bits
- **Significant precision:** 24 bits (23 explicitly stored)

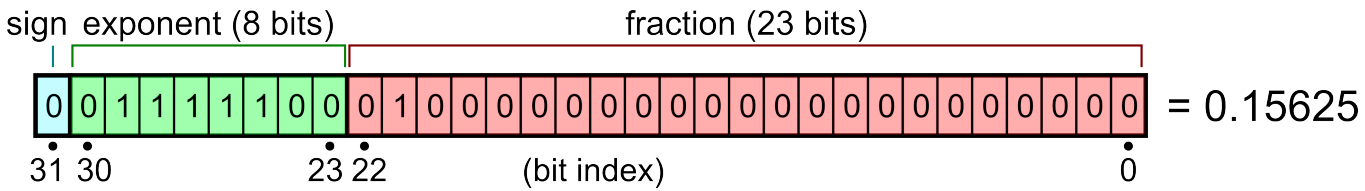
This gives from 6 to 9 significant decimal digits precision (if a decimal string with at most 6 significant decimal is converted to IEEE 754 single precision and then converted back to the same number of significant decimal, then the final string should match the original; and if an IEEE 754 single precision is converted to a decimal string with at least 9 significant decimal and then converted back to single, then the final number must match the original).

As with single-precision floating-point format, it lacks precision on integer numbers when compared with an integer format of the same size.

IEEE 754 single-precision binary floating-point format 1/8

IEEE 754 single-precision binary floating-point format: binary32 (extret de la wikipedia) (cont.)

- **Sign bit** determines the sign of the number, which is the sign of the significant as well.
- **Exponent** is an *8 bit unsigned integer* from 0 to *255*. The exponent value used in the arithmetic is the exponent shifted by a *bias of 127*. Thus, an exponent value of 127 represents the actual zero (i.e. for 2^{e-127} to be one, e must be 127).
- The **true significant** includes *23 fraction bits* to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros. Thus only *23 fraction bits of the significant appear in the memory* format but the total precision is 24 bits (equivalent to $\log_{10}(224) \approx 7.225$ decimal digits). The bits are laid out as follows:



IEEE 754 single-precision binary floating-point format 2/8

IEEE 754 single-precision binary floating-point format: binary32 (extret de la wikipedia) (cont.)

The *single-precision binary floating-point exponent* e is encoded using an offset-binary representation, with

$$\text{Exponent bias} = 7F_{\text{H}} = 127$$

Thus, the *true exponent* is

$$e - 7F_{\text{H}} = e - 127.$$

Moreover,

- $q_{min} = 01_{\text{H}} - 7F_{\text{H}} = -126$,
- $q_{max} = FE_{\text{H}} - 7F_{\text{H}} = 127$ $(FE_{\text{H}} = 254)$.

IEEE 754 single-precision binary floating-point format: `binary32` (extret de la wikipedia) (cont.)

The real value assumed by a given 32 bit `binary32` data with a given biased exponent e (the 8 bit unsigned integer) and a *23 bit fraction* is

$$(-1)^{\text{sign}}(1.b_{22}b_{21}...b_0)_2 \times 2^{e-127}$$

where, more precisely, we have

$$\text{value} = (-1)^{\text{sign}} \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}.$$

Note that *the number is positive if sign* $= 0$ *and negative otherwise*.

IEEE 754 single-precision binary floating-point format: `binary32` (extret de la wikipedia) (cont.)

Example from the figure

- sign $= 0$
- $1 + \sum_{i=1}^{23} b_{23-i} 2^{-i} = 1 + 2^{-2} = 1.25$
- biased exponent $e = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = 124.$
- $2^{(e-127)} = 2^{124-127} = 2^{-3}$

thus:

$$\text{value} = 1.25 \times 2^{-3} = 0.15625.$$

IEEE 754 single-precision binary floating-point format: binary32 (extret de la wikipedia) (cont.)

Exemple: representació de $\mathrm{fl}_A(0.1)$ en bin32

A un exemple anterior hem vist

$$(0.1)_{10} = (0.\widehat{1100})_2 \times 2^{-3} = (1.1001\widehat{1100})_2 \times 2^{-4}$$

Llavors,

$s = 0$ (és un número positiu)

$e = (-4) + 127 = 123$ (en binari) i

$m = 10011001100110011001101$ — la mantissa és directament $1001\widehat{1100}$ truncada a 23 dígits amb arrodoniment (el darrer bit — 1 en comptes de 0 — prové de l'arrodoniment).

Ara cal passar l'exponent a binari.

Per passar un número natural a binari, hem de dividir per dos; el reste dóna el primer dígit (de dreta a esquerra); ens quedem amb el quocient i iterem el procediment fins que obtinguem quocient zero.

IEEE 754 single-precision binary floating-point format: binary32 (extret de la wikipedia) (cont.)

Exemple: representació de $\mathrm{fl}_A(0.1)$ en bin32 (cont.)

En el nostre cas,

$$\begin{aligned}
123 \div 2 &= 61 \xrightarrow{\text{reste}} \boxed{1} \\
61 \div 2 &= 30 \longrightarrow \boxed{1} \\
30 \div 2 &= 15 \longrightarrow \boxed{0} \\
15 \div 2 &= 7 \longrightarrow \boxed{1} \\
7 \div 2 &= 3 \longrightarrow \boxed{1} \\
3 \div 2 &= 1 \longrightarrow \boxed{1} \\
1 \div 2 &= 0 \longrightarrow \boxed{1},
\end{aligned}$$

d'on $123 = (1111011)_2$.

Per tant, $\mathrm{fl}_A(0.1)$ en format IEEE simple queda

0	0	1	1	1	1	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1

IEEE 754 single-precision binary floating-point format: binary32 (extret de la wikipedia) (cont.)

Encoding of the IEEE 754 single-precision binary floating-point format: The *stored exponents* 00_H and FF_H are interpreted specially

Exponent	Significant: zero	Significant: non-zero	Equation
00_H	zero, -0	denormal numbers	$(-1)^s \times 2^{-126} \times 0.m$
$01_H, \ldots, FE_H$	normalized value		$(-1)^s \times 2^{e-127} \times 1.m$
FF_H	$\pm\infty$	NaN (quiet, signalling)	

where s denotes the `sign-bit`, m denotes the `significant-bits` and e denotes the `exponent-bits`.

- The minimum positive (denormal) value is $2^{-149} \approx 1.4 \times 10^{-45}$.
- The minimum positive normal value is $2^{-126} \approx 1.18 \times 10^{-38}$.
- The maximum representable value is $(2 - 2^{-23}) \times 2^{127} \approx 3.4 \times 10^{38}$.

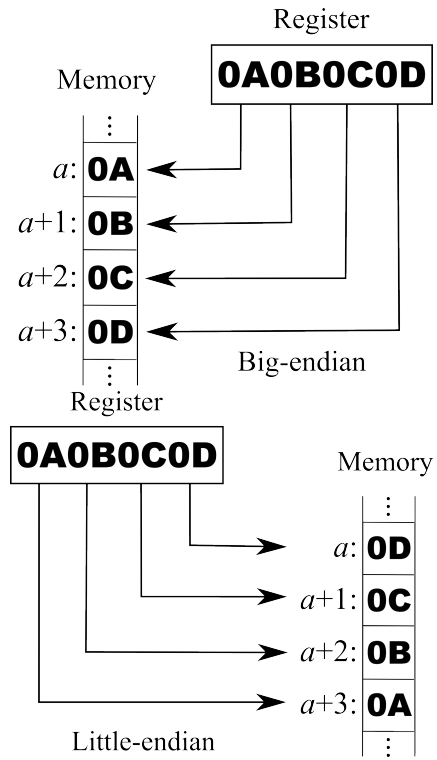
IEEE 754 single-precision binary floating-point format 8/8

Endianness (extret de la wikipedia)

The terms *endian* and *endianness* refer to the convention used to interpret the bytes making up a data word when those bytes are stored in computer memory. Memory commonly stores binary data by organizing it *linearly* into 8-bit units called *bytes*. When reading or writing a data word consisting of multiple such units, the order of the bytes stored in memory determines the interpretation of the data word.

Big-endian systems store the most significant byte of a word in the smallest address and the least significant byte is stored in the largest address.

Little-endian systems store the least significant byte in the smallest address.



Endianness 1/2

Endianness (extret de la wikipedia) (cont.)

The ubiquitous x86 of today use little-endian storage for all types of data (integer, floating point, BCD). However, there have been a few historical machines where floating point numbers were represented in big-endian form while integers were represented in little-endian form.

The widespread IEEE 754 floating point standard does not specify endianness. Theoretically, this means that even standard IEEE floating point data written by one machine might not be readable by another. However, on modern standard computers (i.e., implementing IEEE 754), one may in practice safely assume that the endianness is the same for floating point numbers as for integers, making the conversion straightforward regardless of data type.

So, to make low level manipulations, it may be necessary to know the endianness of the system.

Endianness 2/2

IEEE 754 double-precision binary floating-point format: `binary64` (extret de la wikipedia)

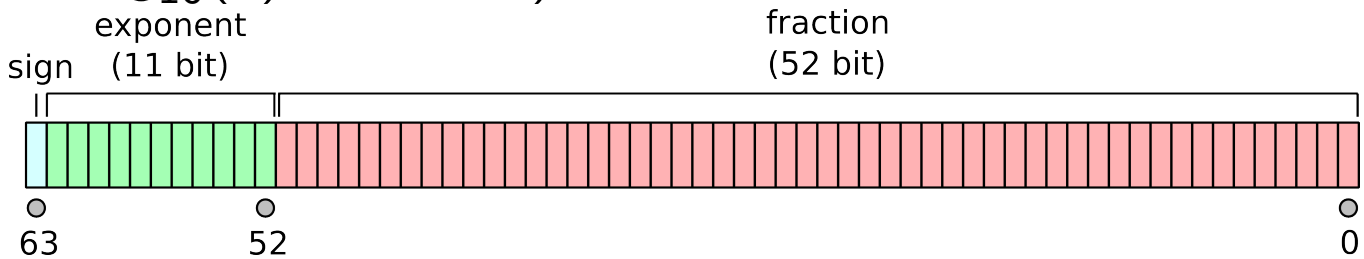
Double-precision binary floating-point is a commonly used format on PCs, due to its wider range over single-precision floating point, in spite of its performance and bandwidth cost. It is commonly known simply as double. The IEEE 754 standard specifies a `binary64` as having:

- *Sign bit:* 1 bit
- *Exponent width:* 11 bits
- *Significant precision:* 53 bits (52 explicitly stored)

This gives 15–17 significant decimal digits precision. If a decimal string with at most 15 significant digits is converted to IEEE 754 double precision representation and then converted back to a string with the same number of significant digits, then the final string should match the original. If an IEEE 754 double precision is converted to a decimal string with at least 17 significant digits and then converted back to double, then the final number must match the original.

IEEE 754 double-precision binary floating-point format: binary64 (extret de la wikipedia) (cont.)

- *Sign bit* determines the sign of the number, which is the sign of the significant as well.
- *Exponent* is an *11 bit unsigned integer* from 0 to *2047*. The exponent value used in the arithmetic is the exponent shifted by a *bias of 1023*. Thus, an exponent value of 1023 represents the actual zero (i.e. for 2^{e-1023} to be one, e must be 2023).
- The *true significant* includes *52 fraction bits* to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros. Thus only *52 fraction bits of the significant appear in the memory* format but the total precision is 53 bits (approximately 16 decimal digits, $53 \log_{10}(2) \approx 15.955$). The bits are laid out as follows:



IEEE 754 double-precision binary floating-point format 2/8

IEEE 754 double-precision binary floating-point format: binary64 (extret de la wikipedia) (cont.)

The *double-precision binary floating-point exponent* e is encoded using an offset-binary representation, with

$$\text{Exponent bias} = 3FF_{\text{H}} = 1023$$

Thus, the *true exponent* is

$$e - 3FF_{\text{H}} = e - 1023.$$

Moreover,

- $q_{min} = 01_{\text{H}} - 3FF_{\text{H}} = -1022$,
- $q_{max} = 7FE_{\text{H}} - 3FF_{\text{H}} = 1023$ $(7FE_{\text{H}} = 2046)$.

IEEE 754 double-precision binary floating-point format: `binary64` (extret de la wikipedia) (cont.)

The real value assumed by a given 64-bit double-precision datum with a given biased exponent e and a *52-bit fraction* is

$$(-1)^{\text{sign}}(1.b_{51}b_{50}...b_0)_2 \times 2^{e-1023}$$

where more precisely we have

$$\text{value} = (-1)^{\text{sign}} \left(1 + \sum_{i=1}^{52} b_{52-i}2^{-i}\right) \times 2^{e-1023}$$

- The spacing as a fraction of the numbers in the range from 2^n to 2^{n+1} is 2^{n-52}.
- The maximum relative rounding error when rounding a number to the nearest representable one (the *machine epsilon*) is therefore 2^{-53}.

IEEE 754 double-precision binary floating-point format: binary64 (extret de la wikipedia) (cont.)

Example: $\texttt{3FD5 5555 5555 5555}_{\text{H}}$

- Sign $= 0$
- Exponent $= \texttt{3FD}_{\text{H}} = 1021$
- Significant $= 5\ 5555\ 5555\ 5555_{\text{H}}$
- $15\ 5555\ 5555\ 5555_{\text{H}} = 16^{13} + 5\sum_{i=0}^{12} 16^i = 6004799503160661$
- Value $= 2^{(\text{Exponent}-1023)} \times 1.\text{Significant}$
  $= 2^{-2} \times \left(15\ 5555\ 5555\ 5555_{\text{H}} \times 2^{-52}\right)$
  $= 2^{-54} \times 6004799503160661$
  $= 0.333333333333333314829616256247390$
  $99293947219848632812 5$
  $\approx 1/3$

Remark

By default, $1/3$ rounds down, instead of up like single precision, because of the odd number of bits in the significant.

IEEE 754 double-precision binary floating-point format 5/8

IEEE 754 double-precision binary floating-point format: `binary64` (extret de la wikipedia) (cont.)

Encoding of the IEEE 754 double-precision binary floating-point format: The *stored exponents* 000_H and $7FF_H$ are interpreted specially:

Exponent	Significant: zero	Significant: non-zero	Equation
000_H	zero, -0	denormal numbers	$(-1)^s \times 2^{-1022} \times 0.m$
$001_H, \ldots, 7FE_H$	normalized value		$(-1)^s \times 2^{e-1023} \times 1.m$
$7FF_H$	$\pm\infty$	`NaN` (quiet, signalling)	

where s denotes the `sign-bit`, m denotes the `significant-bits` and e denotes the `exponent-bits`.

IEEE 754 double-precision binary floating-point format: binary64 (extret de la wikipedia) (cont.)

Several examples

$\texttt{3FF0 0000 0000 0000}_{\text{H}} = 1$

$(\texttt{3FF}_{\text{H}} = 001111111111_2;\ \texttt{s} = +;\ \texttt{e} = \texttt{3FF}_{\text{H}} = 3 * 16^2 + 15 * 16 + 15 = 1023)$

$\texttt{3FF0 0000 0000 0001}_{\text{H}} \approx 1.0000000000000002$ (smallest number > 1)

$\texttt{3FF0 0000 0000 0002}_{\text{H}} \approx 1.0000000000000004$

$\texttt{4000 0000 0000 0000}_{\text{H}} = 2$

$(400_{\text{H}} = 010000000000_2;\ \texttt{s} = +;\ ;\ \texttt{e} = 400_{\text{H}} = 4 * 16^2 = 2^{10} = 1024)$

$\texttt{C000 0000 0000 0000}_{\text{H}} = -2$

$(\texttt{C00}_{\text{H}} = 110000000000_2;\ \texttt{s} = -;\ \texttt{e} = 400_{\text{H}} = 2^{10} = 1024)$

IEEE 754 double-precision binary floating-point format: binary64 (extret de la wikipedia) (cont.)

Special values

Min subnormal positive double:

$$0000\ 0000\ 0000\ 0001_H = 2^{-1022-52} = 2^{-1074} \approx 4.940656458412465 \times 10^{-324}$$

Max subnormal double:

$$000F\ FFFF\ FFFF\ FFFF_H = \left(1 - 2^{-52}\right) \times 2^{-1022} \approx 2.225073858507201 \times 10^{-308}$$

Min normal positive double:

$$0010\ 0000\ 0000\ 0000_H = 2^{1-1023} \approx 2.225073858507201 \times 10^{-308}$$

Max Double:

$$7FEF\ FFFF\ FFFF\ FFFF_H = (1 + (1 - 2^{-52})) \times 2^{2046-1023} \approx 1.797693134862315 \times 10^{308}$$

Zero and ∞:

$$0000\ 0000\ 0000\ 0000_H = 0 \qquad 8000\ 0000\ 0000\ 0000_H = -0$$

$$7FF0\ 0000\ 0000\ 0000_H = \infty \qquad FFF0\ 0000\ 0000\ 0000_H = -\infty$$

IEEE 754 double-precision binary floating-point format 8/8

Friki-exercici: format punt flotant i programació de baix nivell

Exercici (per lliurament suplementari)

Fer un programa que faci les següents accions:

1. digui per pantalla, mitjançant un sol `if` si som en un sistema little-endian o big-endian;
2. donat un nombre normal `double` (per exemple $\frac{1}{333}$) tregui per pantalla l'exponent normalitzat (de 2 no de 10) del nombre;
3. donat un nombre enter positiu el guardi mitjançant un *cast (forçament de tipus)* a una variable `double` (el resultat és normal o subnormal?) i usant aquesta representació tregui per pantalla la part entera del seu logaritme en base 2 (òbviament no es pot usar la funció logaritme; és massa lent);
4. donat un nombre normal `double` treure per pantalla el nombre multiplicat i dividit per -2 (sense fer cap producte ni cap divisió; per simplificar es pot suposar que el seu exponent en base 2 és mes gran que -1022 i més petit que 1023).

Indicació: Useu una `union` amb una `struct` amb *bit fields* per a descompondre el nombre en els seus elements del format punt flotant i els *bitwise operators*.

Punt flotant: truncament i arrodoniment

D'acord amb l'estructura de la representació en punt flotant, és clar que el conjunt de nombres reals que es poden representar exactament és finit. Concretament,

Definició

Denotarem el *conjunt de nombres representables exactament en aritmètica de punt flotant de t dígits, en base b i amb exponent entre q_{min} i q_{max}* per $F(b, t, q_{min}, q_{max})$. És el conjunt format per la unió de $\{0\}$ i

$$\{\pm(0.\delta_1 \dots \delta_t)_b b^q, \delta_i \in \{0, 1, \dots, b-1\}, \delta_1 \neq 0, q_{min} \leq q \leq q_{max}\}$$

(noteu que només considerem nombres normalitzats). Els nombres de $F(b, t, q_{min}, q_{max})$ també es coneixen com *nombres màquina*.

Donat $x \in \mathbb{R}$, si $x \notin F(b, t, q_{min}, q_{max})$ no el podrem representar exactament, sinó que haurem d'operar amb una aproximació seva $\widetilde{x} \in F(b, t, q_{min}, q_{max})$. Aquesta aproximació es pot trobar per truncament o arrodoniment.

Punt flotant: truncament i arrodoniment (cont.)

Sigui $x \in \mathbb{R}$ amb representació en base b donada per $x = s\left(\sum_{i=1}^{\infty} \alpha_i b^{-i}\right) b^q$, d'acord amb el *Teorema de la representació en punt flotant en base b*, i suposem $q_{min} \leq q \leq q_{max}$.

- La *representació en punt flotant per truncament* de x a $\mathrm{fl}_T(x) \in F(b, t, q_{min}, q_{max})$ es defineix com:

$$\mathrm{fl}_T(x) = s(0.\alpha_1\alpha_2 \ldots \alpha_t) b^q,$$

és a dir $\delta_i = \alpha_i$ per $i = 1, 2, \ldots, t$.

Punt flotant: truncament i arrodoniment (cont.)

- La *representació en punt flotant per arrodoniment* de x a $\mathrm{fl}_A(x) \in F(b, t, q_{min}, q_{max})$ es defineix com:

$$\mathrm{fl}_A(x) = \begin{cases} s\big(0.\alpha_1 \ldots \alpha_t\big)\, b^q & \text{si } 0 \leq \alpha_{t+1} < b/2 \\ s\big(0.\alpha_1 \ldots \alpha_{t-1}(\alpha_t + 1)\big)\, b^q & \text{si } b/2 \leq \alpha_{t+1} \leq b-1 \end{cases}$$

Noteu que s'han de tenir en compte els acarrejos[1].

Exemple (Per $t = 4$ i $b = 10$)

- $\mathrm{fl}_A(0.10004) = 0.1000 \times 10^0$,
- $\mathrm{fl}_A(0.10005) = 0.1001 \times 10^0$,
- $\mathrm{fl}_A(0.99999) = 0.1000 \times 10^1$.

[1] Estrictament, també s'hauria de tenir en compte que un nombre amb $q = q_{max}$ que arrodoneixi cap a amunt pot produir un overflow.

Error absolut i relatiu

A continuació, volem quantificar l'error que es produeix en representar un nombre qualsevol $x \in \mathbb{R}$ per truncament o arrodoniment. Abans d'això hem de formalitzar el concepte d'error.

Definició

Si $x \in \mathbb{R}$ i $\widetilde{x}$ és una aproximació de x, definim l'*error absolut de $\widetilde{x}$ com a aproximació de x*, $e_a(\widetilde{x}, x)$, per

$$e_a(\widetilde{x}, x) = \widetilde{x} - x,$$

i, si $x \neq 0$, definim l'*error relatiu de $\widetilde{x}$ com a aproximació de x*, $e_r(\widetilde{x}, x)$, per

$$e_r(\widetilde{x}, x) = \frac{\widetilde{x} - x}{x} = \frac{e_a(\widetilde{x}, x)}{x}.$$

Error absolut i relatiu (cont.)

Notem que l'error relatiu no està definit si $x = 0$. Quan x sigui desconegut, prendrem

$$e_r(\widetilde{x}, x) \simeq \frac{\widetilde{x} - x}{\widetilde{x}}.$$

Ometrem el segon argument de e_a i e_r quan sigui clar a partir del context (i, per tant, denotarem $e_a(\widetilde{x})$ o $e_r(\widetilde{x})$).

En general només coneixem fites dels errors (absolut i relatiu):

- En el cas de l'error absolut, un número $\varepsilon_a(\widetilde{x}) > 0$ tal que $|e_a(\widetilde{x})| \leq \varepsilon_a(\widetilde{x})$.
  Notarem $x = \widetilde{x} \pm e_a(\widetilde{x})$ o bé $x = \widetilde{x} + e$ amb $|e| \leq \varepsilon_a(\widetilde{x})$.
- En el cas de l'error relatiu, un número $\varepsilon_r(\widetilde{x})$ tal que $|e_r(\widetilde{x})| \leq \varepsilon_r(\widetilde{x})$.
  Notarem $\widetilde{x} = x(1 \pm e_r(\widetilde{x}))$ o bé $\widetilde{x} = x(1 + \delta)$ amb $|\delta| \leq \varepsilon_r(\widetilde{x})$.

Error absolut i relatiu 2/4

Error absolut i relatiu (cont.)

Exemple

$x = 2.100 \pm 5 \times 10^{-4}$ vol dir que aproximem x per $\widetilde{x} = 2.100$ però que x està entre

$$2.100 - 5 \times 10^{-4} = 2.0995 \quad \text{i} \quad 2.100 + 5 \times 10^{-4} = 2.1005.$$

Amb les notacions anteriors, $\varepsilon_a(x) = 5 \times 10^{-4}$ i

$$\varepsilon_r(x) \simeq \frac{\varepsilon_a(x)}{\widetilde{x}} = \frac{5 \times 10^{-4}}{2.100} = 2.38 \times 10^{-4}.$$

Error absolut i relatiu 3/4

Error absolut i relatiu (cont.)

Observació

Sempre que tinguem una igualtat del tipus

$$A = B(1 + \delta)$$

es pot llegir com *l'error relatiu de A com a aproximació de B és* δ, i.e. $e_r(A, B) = \delta$. Això serà d'utilitat més endavant per fitar la propagació de l'error a les operacions.

Xifres significatives

Habitualment escrivim els números en base 10. Una manera còmoda de referir-nos a la precisió d'un càlcul és mitjançant els conceptes de *nombre de decimals correctes* i el *nombre de xifres significatives*.

Definició

Sigui $\widetilde{x}$ una aproximació de x. Si $|e_a(\widetilde{x})| \leq \frac{1}{2}10^{-t_a}$, direm que $\widetilde{x}$ *té* t_a *decimals correctes*. Si $x = sm10^q$, amb $0.1 \leq m < 1$, $\widetilde{x} = s\widetilde{m}10^q$, i

$$t_r = \max\{i \in \mathbb{Z} : |\widetilde{m} - m| \leq \frac{1}{2}10^{-i}\},$$

aleshores direm que $\widetilde{x}$ *té* t_r *xifres significatives*.

Xifres significatives (cont.)

El nombre de xifres significatives està relacionat amb l'error relatiu, d'acord amb el següent

Lema

En les notacions de la definició anterior, es compleix

(a) $|e_r(\widetilde{x})| \leq 5 \times 10^{-t_r}$,

(b) $|\widetilde{m} - m| \leq |e_r(\widetilde{x})|$.

Demostració

És directa de les definicions. □

Xifres significatives 2/3

Xifres significatives (cont.)

Quan escrivim el resultat d'un càlcul, és habitual escriure només tants dígits a la dreta del punt decimal com decimals correctes, o bé tants dígits com xifres significatives (en aquest cas sense comptar zeros al davant).

Exemple

Considerem la següent taula:

	decimals correctes	xifres significatives
$0.001234 \pm \frac{1}{2} \times 10^{-5}$	5	3
$50.789 \pm \frac{1}{2} \times 10^{-3}$	3	5

En la primera entrada, l'última xifra no es ni decimal correcte ni xifra significativa. En la segona, tots els decimals són correctes i totes les xifres són significatives.

Xifres significatives 3/3

Proposició

Sigui $x \in \mathbb{R}$, $x \neq 0$ amb representació en punt flotant infinita i normalitzada $x = s(0.\alpha_1\alpha_2\ldots)_b b^q, \quad \alpha_1 \neq 0$. L'error absolut en la seva representació en punt flotant en base b i t dígits satisfà les següents fites:

$$\begin{aligned} |e_a(\mathrm{fl}_T(x), x)| = |\mathrm{fl}_T(x) - x| &\leq b^{q-t} \\ |e_a(\mathrm{fl}_A(x), x)| = |\mathrm{fl}_A(x) - x| &\leq \frac{1}{2} b^{q-t}, \end{aligned}$$

Les fites per l'error relatiu són:

$$\begin{aligned} |e_r(\mathrm{fl}_T(x), x)| = \left|\frac{\mathrm{fl}_T(x) - x}{x}\right| &\leq b^{1-t}, \\ |e_r(\mathrm{fl}_A(x), x)| = \left|\frac{\mathrm{fl}_A(x) - x}{x}\right| &\leq \frac{1}{2} b^{1-t}. \end{aligned}$$

Errors de representació. Demostració

Sigui $x = s(0.\alpha_1\alpha_2\ldots)b^q$. La fita per $e_a\big(\mathrm{fl}_T(x)\big)$ se segueix directament de la definició de truncament.
Per la fita de $e_r\big(\mathrm{fl}_T(x)\big)$ fem

$$\left|\frac{\mathrm{fl}_T(x)-x}{x}\right| \leq \frac{b^{q-t}}{(0.\alpha_1\alpha_2\ldots\alpha_t\ldots)_b\, b^q} = \frac{b^{-t}}{(0.\alpha_1\alpha_2\ldots\alpha_t\ldots)_b} \leq b^{1-t},$$

on la darrera desigualtat és deguda al fet que
$0 < (0.\alpha_1\alpha_2\ldots\alpha_t\ldots)_b \leq 1$.

La fita de l'error absolut pel cas d'arrodoniment s'ha de fer en dos casos:

Cas 1: Si $0 \leq \alpha_{t+1} < b/2$, aleshores

$$\begin{aligned} |\mathrm{fl}_A(x)-x| &= |x| - |\mathrm{fl}_A(x)| = (0.\underbrace{0\cdots 0}_{t}\alpha_{t+1}\alpha_{t+2}\ldots)_b\, b^q \\ &= (0.\alpha_{t+1}\alpha_{t+2}\ldots)_b\, b^{q-t} \leq \frac{1}{2}b^{q-t}. \end{aligned}$$

Errors de representació. Demostració 1/2

Errors de representació. Demostració (cont.)

Cas 2: Si $b/2 \leq \alpha_{t+1} \leq b-1$, aleshores:

$$
\begin{aligned}
&|\,\mathrm{fl}_A(x) - x| \\
&\quad = |\,\mathrm{fl}_A(x)| - |x| \\
&\quad = \big(0.\alpha_1 \ldots (\alpha_t + 1)\big)_b b^q - \big(0.\alpha_1 \ldots \alpha_t \alpha_{t+1} \ldots\big)_b b^q \\
&\quad = b^q \Big((0.\alpha_1 \ldots \alpha_t)_b + b^{-t} - (0.\alpha_1 \ldots \alpha_t)_b - (0.0 \ldots 0 \alpha_{t+1} \alpha_{t+2} \ldots)_b \Big) \\
&\quad = b^q \Big(b b^{-(t+1)} - (0.0 \ldots 0 \alpha_{t+1})_b - (0.0 \ldots 00 \alpha_{t+1} \alpha_{t+2} \ldots)_b \Big) \\
&\quad \leq b^q \underbrace{(b - \alpha_{t+1})}_{\leq b/2} b^{-(t+1)} \quad \leq \quad \frac{1}{2} b^{q-t}.
\end{aligned}
$$

La fita de l'error relatiu per arrodoniment es dedueix amb el mateix argument que en el cas de $e_r\big(\mathrm{fl}_T(x)\big)$. □

Errors de representació. Demostració 2/2

Èpsilon de la màquina

La quantitat b^{1-t} o $\frac{1}{2}b^{1-t}$ s'anomena la *precisió* de la màquina (segons que aquesta talli o arrodoneixi). Un concepte relacionat és el que definim tot seguit.

Definició

Es coneix com a *èpsilon de la màquina* el nombre positiu ϵ més petit que sumat a 1 dóna diferent de 1:

$$\epsilon = \min\{\varepsilon > 0 : \mathrm{fl}(1+\varepsilon) \neq 1\}.$$

L'èpsilon de la màquina i la precisió resulten ser iguals.

Proposició

Per a una màquina que trunca, l'èpsilon màquina és b^{1-t}, mentre que per una màquina que arrodoneix és $\frac{1}{2}b^{1-t}$.

Demostració

És un exercici. □

Aritmètica de punt flotant

Suposem que tenim una màquina amb precisió ε. És fàcil comprovar que les operacions aritmètiques $(+,-,\cdot,/)$ entre nombres màquina no necessàriament donen un nombre màquina. Per tant, no podem reproduir exactament les operacions aritmètiques i ens hem de conformar amb substituts aproximats, que anomenem *operacions en punt flotant* i denotarem per $\oplus$, $\ominus$, $\otimes$, $\oslash$.

Per simplificar, suposarem que les operacions en punt flotant només cometen error en la representació del resultat, això és,

$$x \oplus y = \mathrm{fl}(\mathrm{fl}(x) + \mathrm{fl}(y)) \qquad x \otimes y = \mathrm{fl}(\mathrm{fl}(x) \cdot \mathrm{fl}(y))$$
$$x \ominus y = \mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y)) \qquad x \oslash y = \mathrm{fl}(\mathrm{fl}(x) / \mathrm{fl}(y))$$

(de fet, això és cert a la pràctica totalitat d'ordinadors i calculadores). Això implica que, cada cop que efectuem una operació en punt flotant, cometem error. A mida que el càlcul avança, introduïm nous errors i propaguem els errors de càlculs anteriors.

Aritmètica de punt flotant (cont.)

La notació $x \oplus y$, $x \ominus y$, etc., no és estàndard. És més habitual denotar les operacions aritmètiques amb fl:

$$\begin{array}{llll} \mathrm{fl}(x+y) & := & x \oplus y, & \mathrm{fl}(x-y) := x \ominus y, \\ \mathrm{fl}(xy) & := & x \otimes y, & \mathrm{fl}(x/y) := x \oslash y. \end{array}$$

Usant aquesta notació, x i y poden ser directament nombres o bé expressions. En aquest darrer cas, les regles anteriors s'apliquen recurrentment fins a acabar tenint nombres com a arguments de tots els fl. En veurem exemples més avall.

Aritmètica de punt flotant 2/2

Exemple 1: fitació l'error en el càlcul de xy

Si suposem que ni x ni y són necessàriament nombres màquina, el que calcularem realment és $\mathrm{fl}(x) \otimes \mathrm{fl}(y)$. Com abans, anomenem ε l'èpsilon–màquina. Per a certs $\delta_1, \delta_2, \delta_3$ verificant $|\delta_1|, |\delta_2|, |\delta_3| \leq \varepsilon$, tindrem

$$\begin{aligned}
\mathrm{fl}(x) &= x(1+\delta_1), \qquad \mathrm{fl}(y) = y(1+\delta_2), \\
\mathrm{fl}(x) \otimes \mathrm{fl}(y) &= \mathrm{fl}(x)\,\mathrm{fl}(y)(1+\delta_3) \\
&= xy(1+\delta_1)(1+\delta_2)(1+\delta_3) \\
&= xy(1+\delta_1+\delta_2+\delta_1\delta_2)(1+\delta_3) \\
&\approx xy(1+\delta_1+\delta_2)(1+\delta_3) \\
&= xy(1+\delta_1+\delta_2+\delta_3+\delta_3(\delta_1+\delta_2)) \\
&\approx xy(1+\delta_1+\delta_2+\delta_3)
\end{aligned}$$

on $\approx$ denota que hem eliminat sumands que són producte de dos o més errors. Aquest procés l'anomenarem *eliminar termes d'ordre superior* i el farem de manera sistemàtica. Moltes vegades escriurem $=$ en comptes de $\approx$.

Exemple 1: fitació l'error en el càlcul de xy 1/2

Exemple 1: fitació l'error en el càlcul de xy (cont.)

D'acord amb l'anterior,

$$e_r(\mathrm{fl}(x) \otimes \mathrm{fl}(y)) \approx \delta_1 + \delta_2 + \delta_3,$$

d'on

$$|e_r(\mathrm{fl}(x) \otimes \mathrm{fl}(y))| \leq 3\varepsilon,$$

o, escrit de forma equivalent,

$$\varepsilon_r(\mathrm{fl}(x) \otimes \mathrm{fl}(y)) = 3\varepsilon.$$

Per tant, també tenim

$$\varepsilon_a(\mathrm{fl}(x) \otimes \mathrm{fl}(y)) = 3|xy|\varepsilon.$$

Si x, y fossin nombres màquina, tindríem $\delta_1 = \delta_2 = 0$ i l'error relatiu estaria fitat per ε.

Exemple 1: fitació l'error en el càlcul de xy 2/2

Exemple 2: error en el càlcul de $(a_1 \oplus a_2) \oplus a_3$, on a_1, a_2, a_3 són números màquina

Calculem $e_r((a_1 \oplus a_2) \oplus a_3)$. Aquest cop fem anar la notació fl:

$$
\begin{aligned}
&(a_1 \oplus a_2) \oplus a_3 \\
&= \mathrm{fl}\big((a_1 + a_2) + a_3\big) = \big(\mathrm{fl}(a_1 + a_2) + \mathrm{fl}(a_3)\big)(1+\delta_2) \\
&= \big(\mathrm{fl}(a_1 + a_2) + a_3\big)(1+\delta_2) \\
&= \Big(\big(\mathrm{fl}(a_1) + \mathrm{fl}(a_2)\big)(1+\delta_1) + a_3\Big)(1+\delta_2) \\
&= \big((a_1 + a_2)(1+\delta_1) + a_3\big)(1+\delta_2) \\
&= \big(a_1 + a_2 + a_3 + \delta_1(a_1 + a_2)\big)(1+\delta_2) \\
&= a_1 + a_2 + a_3 + \delta_1(a_1 + a_2) + \delta_2(a_1 + a_2 + a_3) + \delta_1\delta_2(a_1 + a_2) \\
&\approx a_1 + a_2 + a_3 + \delta_1(a_1 + a_2) + \delta_2(a_1 + a_2 + a_3) \\
&= (a_1 + a_2 + a_3)\Big(1 + \delta_1 \frac{a_1 + a_2}{a_1 + a_2 + a_3} + \delta_2\Big),
\end{aligned}
$$

essent $|\delta_1|, |\delta_2| \leq \varepsilon$.

Exemple 2: error en el càlcul de $(a_1 \oplus a_2) \oplus a_3$, on a_1, a_2, a_3 són números màquina 1/2

Exemple 2: error en el càlcul de $(a_1 \oplus a_2) \oplus a_3$, on a_1, a_2, a_3 són números màquina (cont.)

Per tant,

$$e_r\big((a_1 \oplus a_2) \oplus a_3\big) = \delta_1 \frac{a_1 + a_2}{a_1 + a_2 + a_3} + \delta_2. \tag{4}$$

Notem que, si $\delta_1 \neq 0$ i $|a_1 + a_2| \gg |a_1 + a_2 + a_3|$, l'error relatiu es dispara. Una de les coses que se segueix d'això és que l'ordre de sumació és important, i per tant la suma en punt flotant no és associativa.

Exemple 3: la suma en punt flotant no és associativa

Treballem amb $b = 10$, $t = 8$ i definim

$$\begin{aligned} a &:= 0.23371258 \times 10^{-4}, \\ b &:= 0.33678429 \times 10^{2}, \\ c &:= -0.33677811 \times 10^{2}. \end{aligned}$$

Aleshores,

$$\begin{aligned} (a \oplus b) \oplus c &= 0.33678452 \times 10^{2} - 0.33677811 \times 10^{2} \\ &= 0.64100000 \times 10^{-3}, \\ (b \oplus c) \oplus a &= 0.61800000 \times 10^{-3} + 0.23371258 \times 10^{-4} \\ &= 0.64137126 \times 10^{-3}. \end{aligned}$$

El resultat exacte és $a + b + c = 0.641371258 \times 10^{-3}$.

Exemple 3: la suma en punt flotant no és associativa (cont.)

Mirem quin valor pren el factor d'amplificació en cada cas.

	a_1	a_2	a_3	$\left\lvert \frac{a_1+a_2}{a_1+a_2+a_3} \right\rvert$
Cas 1	a	b	c	5.25×10^4
Cas 2	b	c	a	0.964

Noteu que es prediu molt bé la pèrdua de dígits en el cas 1.

El que està passant és que s'està donant el procés conegut com a *cancel·lació:* quan, en el cas 1, sumem $a \oplus b$ i c, estem restant quantitats molt properes i en fer-ho perdem 5 dígits (feu la resta a mà i veureu que obteniu 5 zeros al començament).

Exemple 3: la suma en punt flotant no és associativa 2/3

Exemple 3: la suma en punt flotant no és associativa (cont.)

La conclusió que s'ha de treure de l'exemple anterior és que les cancel·lacions no són perilloses en quant a introduir errors en les operacions, però són fatals en quant a propagar errors acumulats, com tornarem a veure a la següent secció.

Observació

Si sou suspicaços, us haureu adonat que en el cas 2 també hi ha una cancel·lació (en la primera suma que es fa), però en aquest cas no dispara l'error, tal com prediu (4). La raó és que estem suposant que a, b, c són números màquina, i en fer $b \oplus c$ de fet estem obtenint la suma exacta $b + c$. Per tant aquesta cancel·lació no està introduint cap error.

Propagació d'errors

En aquesta secció estudiarem com es propaguen els errors suposant que fem les operacions de manera exacta. Tot i que no és veritat mai, suposar les operacions exactes és del tot raonable quan l'error en les dades d'entrada és uns quants ordres de magnitud superior a l'èpsilon–màquina.

Començarem amb una proposició que fita la propagació d'errors en fer operacions aritmètiques, suposant que aquestes s'efectuen de manera exacta. Abans d'això, recordarem la fórmula de Taylor en diverses variables.

Proposició (Fórmula de Taylor en diverses variables)

Sigui $f : U \subset \mathbb{R}^n \to \mathbb{R}$, $a \in U$, U *obert estrellat respecte de* a, $f \in C^k(U)$. *Aleshores, per tot* h *t.q.* $a + h \in U$,

$$\begin{aligned} f(a+h) &= f(a) + \sum_{i=1}^{n} \partial_{x_i} f(a) h_i + \frac{1}{2!} \sum_{i,j=1}^{n} \partial^2_{x_j x_i} f(a) h_i h_j \\ &\quad + \frac{1}{(k-1)!} \sum_{i_1,\ldots,i_{k-1}=1}^{n} \partial^{k-1}_{x_{i_{k-1}},\ldots,x_{i_1}} f(a) h_{i_1} \ldots h_{i_{k-1}} \\ &\quad + \frac{1}{k!} \sum_{i_1,\ldots,i_k}^{n} \partial^{k}_{x_{i_k},\ldots,x_{i_1}} f(a+\xi h) h_{i_1} \ldots h_{i_k}, \end{aligned}$$

on $\xi \in (0,1)$ *depèn de* h.

Propagació d'errors 2/4

Propagació d'errors (cont.)

Proposició (fórmula de propagació de l'error maximal)

Sigui $f : U \subset \mathbb{R}^n \to \mathbb{R}$, $x = (x_1, \ldots, x_n) \in U$, U *obert estrellat respecte de* x, $\widetilde{x}_i$ *aproximació de* x_i *per* $i = 1, \ldots, n$, $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_n) \in U$. *Aleshores, menyspreant productes d'almenys dos errors,*

(a) $$e_a(f(\widetilde{x}_1, \ldots, \widetilde{x}_n), f(x_1, \ldots, x_n)) = \sum_{i=1}^{n} \partial_{x_i} f(x) e_a(\widetilde{x}_i, x_i),$$

(b) $$e_r(f(\widetilde{x}_1, \ldots, \widetilde{x}_n), f(x_1, \ldots, x_n)) = \sum_{i=1}^{n} \frac{x_i \partial_{x_i} f(x)}{f(x)} e_r(\widetilde{x}_i, x_i).$$

Per a les corresponents fites, tenim

(c) $$\varepsilon_a(f(\widetilde{x}_1, \ldots, \widetilde{x}_n), f(x_1, \ldots, x_n)) = \sum_{i=1}^{n} |\partial_{x_i} f(x)| \varepsilon_a(\widetilde{x}_i, x_i),$$

(d) $$\varepsilon_r(f(\widetilde{x}_1, \ldots, \widetilde{x}_n), f(x_1, \ldots, x_n)) = \sum_{i=1}^{n} \left| \frac{x_i \partial_{x_i} f(x)}{f(x)} \right| \varepsilon_r(\widetilde{x}_i, x_i).$$

Demostració

Si considerem la fórmula de Taylor amb resta d'ordre 2,

$$f(a+h) = f(a) + \sum_{i=1}^{n} \partial_{x_i} f(a) h_i + \frac{1}{2!} \sum_{i,j=1}^{n} \partial^2_{x_j,x_i} f(a+\xi h) h_i h_j,$$

i menyspreem la resta d'ordre 2, només cal prendre $a = x$ i $h = \widetilde{x} - x$ per obtenir

$$f(\widetilde{x}) - f(x) \approx \sum_{i=1}^{n} \partial_{x_i} f(x) h_i,$$

que és l'apartat (a). L'apartat (b) s'obté usant la definició d'error relatiu. Els altres apartats s'obtenen prenent valors absoluts i usant la desigualtat triangular. □

Exemple: Volum de l'esfera

Volem calcular

$$V = \frac{1}{6}\pi d^3 \quad \text{amb} \quad \pi = 3.14 \pm 0.0016 \text{ i } d = 3.7 \pm 0.05\text{cm}.$$

Podem fer directament

$$\frac{(3.14 - 0.0016)[(3.7 - 0.05)10^{-2}]^3}{6} \leq V \leq \frac{(3.14 + 0.0016)[(3.7 + 0.05)10^{-2}]^3}{6},$$

que ens dona fites del resultat exacte.

Exemple: Volum de l'esfera 1/2

Exemple: Volum de l'esfera (cont.)

Per altra banda, per la fórmula de propagació d'errors:

$$
\begin{aligned}
|\Delta V| &\lessapprox \left|\frac{d^3}{6}\right| |\Delta\pi| + \left|\frac{\pi d^2}{2}\right| |\Delta d| \\
&= \left|\frac{(3.7 \cdot 10^{-2})^3}{6}\right| \cdot 0.0016 + \left|\frac{3.14 \cdot (3.7 \cdot 10^{-2})^2}{2}\right| \cdot 0.05 \cdot 10^{-2} \\
&\approx 1.088 \cdot 10^{-6}.
\end{aligned}
$$

Llavors,

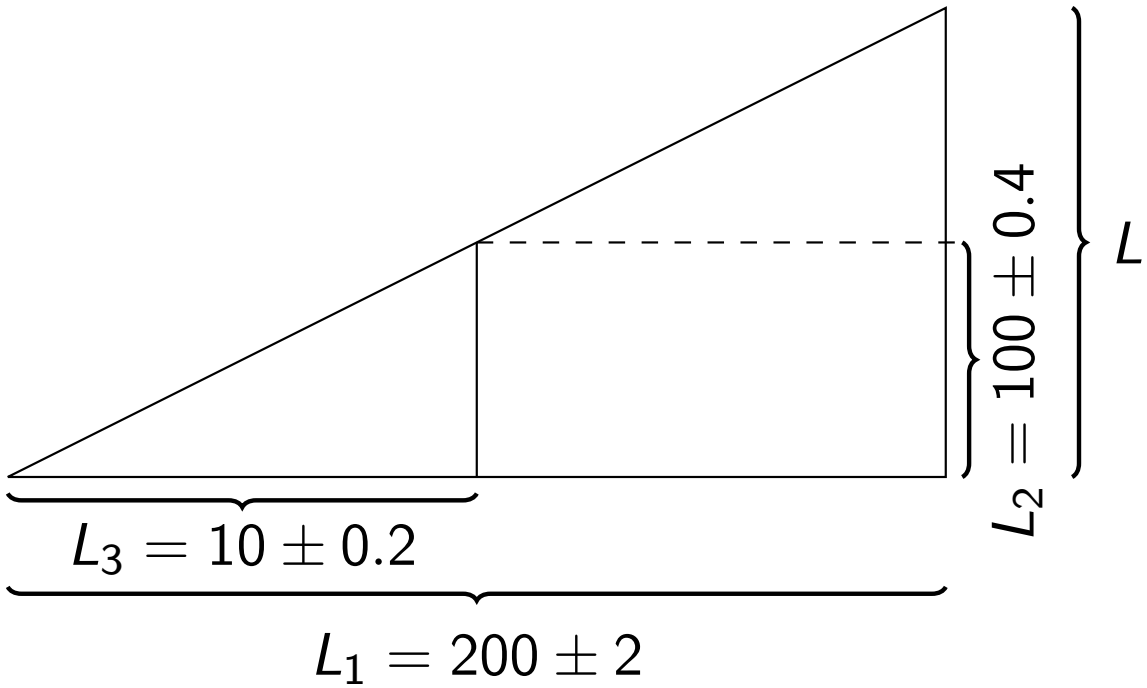
$$
V = \frac{3.14 \cdot (3.7 \cdot 10^{-2})^3}{6} \pm 1.088 \cdot 10^{-6} = (26.508 \pm 1.088) \cdot 10^{-6}, \text{ i}
$$

$$
\varepsilon_r(V) = \frac{|\Delta V|}{|V|} \approx 0.04105.
$$

Exemple: Volum de l'esfera 2/2

Exemple: Càlcul del costat d'un triangle L

Volem calcular la mida L d'un costat del següent triangle:



Pel Teorema de Tales tenim:

$$\frac{L}{L_1} = \frac{L_2}{L_3} \Longrightarrow L = L(L_1, L_2, L_3) = \frac{L_1 L_2}{L_3}.$$

Exemple: Càlcul del costat d'un triangle L 1/2

Exemple: Càlcul del costat d'un triangle L (cont.)

Com abans podem fer afitar el resultat exacte:

$$\frac{198 \cdot 99.6}{10.2} = 1933.41 \leq L \leq \frac{202 \cdot 100.4}{9.8} = 2069.47$$

i

$$L = \frac{200 \cdot 100}{10} = 2000.$$

La fórmula de propagació d'errors ens dona:

$$|\Delta L| \lessapprox \left|\frac{L_2}{L_3}\right| |\Delta L_1| + \left|\frac{L_1}{L_3}\right| |\Delta L_2| + \left|\frac{L_1 L_2}{L_3^2}\right| |\Delta L_3| \approx 68.$$

Llavors,

$$L = 2000 \pm 68, \text{ i}$$
$$\varepsilon_r(L) = 0.034.$$

Exemple: Càlcul del costat d'un triangle L 2/2

Una altra manera de calcular l'error relatiu

Lema

$$\Delta \log |f(x)| \approx \varepsilon(f(x)).$$

Demostració

$$\Delta f(x) = f(x + \Delta x) - f(x) \approx \sum_{i=1}^{n} \partial_{x_i} f(x) \Delta x_i$$

$$\varepsilon(f(x)) = \frac{\Delta f(x)}{f(x)} \approx \sum_{i=1}^{n} \frac{\partial_{x_i} f(x)}{f(x)} \Delta x_i, \text{ i}$$

$$\begin{aligned} \Delta \log |f(x)| &= \log |f(x + \Delta x)| - \log |f(x)| \\ &\approx \sum_{i=1}^{n} \partial_{x_i} \log |f(x)| \, \Delta x_i = \sum_{i=1}^{n} \frac{\partial_{x_i} f(x)}{f(x)} \Delta x_i. \end{aligned}$$

Exemple re-visitat: Càlcul del costat d'un triangle L usant el lema anterior

$$\log|L| = \log|L_1| + \log|L_2| - \log|L_3|$$

i

$$|\varepsilon(L)| \approx |\Delta \log|L|| \lesssim \sum_{i=1}^{3} \left|\frac{1}{L_i}\right| \Delta L_i = \frac{2}{200} + \frac{0.4}{100} + \frac{0.2}{10} = 0.034.$$

Error absolut i relatiu de les operacions elementals

En el cas de les operacions elementals obtenim:

Proposició

Suposem que $\widetilde{x}$ és una aproximació de x i que $\widetilde{y}$ és una aproximació de y.

(a) $e_a(\widetilde{x} \pm \widetilde{y}, x \pm y) = e_a(\widetilde{x}, x) \pm e_a(\widetilde{y}, y)$,

(b) $e_r(\widetilde{x}\widetilde{y}, xy) = e_r(\widetilde{x}, x) + e_r(\widetilde{y}, y)$,

(c) $e_r(\widetilde{x}/\widetilde{y}, x/y) = e_r(\widetilde{x}, x) - e_r(\widetilde{y}, y)$.

Per a les corresponents fites,

(d) $\varepsilon_a(\widetilde{x} \pm \widetilde{y}, x \pm y) = \varepsilon_a(\widetilde{x}, x) + \varepsilon_a(\widetilde{y}, y)$,

(e) $\varepsilon_r(\widetilde{x}\widetilde{y}, xy) = \varepsilon_r(\widetilde{x}, x) + \varepsilon_r(\widetilde{y}, y)$,

(f) $\varepsilon_r(\widetilde{x}/\widetilde{y}, x/y) = \varepsilon_r(\widetilde{x}, x) + \varepsilon_r(\widetilde{y}, y)$.

Demostració

Fem, per exemple, (c) i (f). Prenent $f(x,y) = x/y$ i aplicant la *fórmula de propagació dels errors maximals*,

$$\begin{aligned} e_r(\widetilde{x}/\widetilde{y}) &= e_r(f(\widetilde{x},\widetilde{y}), f(x,y)) \\ &= \frac{x\partial_x f(x,y)}{f(x,y)} e_r(\widetilde{x},x) + \frac{y\partial_y f(x,y)}{f(x,y)} e_r(\widetilde{y},y) \\ &= \frac{x(1/y)}{x/y} e_r(\widetilde{x},x) + \frac{y(-x/y^2)}{x/y} e_r(\widetilde{y},y) \\ &= e_r(\widetilde{x},x) - e_r(\widetilde{y},y). \end{aligned}$$

Prenent valors absoluts, tenim

$$|e_r(\widetilde{x}/\widetilde{y}, x/y)| \leq |e_r(\widetilde{x},x)| + |e_r(\widetilde{y},y)|,$$

d'on $$\varepsilon_r(\widetilde{x}/\widetilde{y}) = \varepsilon_r(\widetilde{x},x) + \varepsilon_r(\widetilde{y},y).$$

La resta d'apartats es demostren de manera similar. □

Error absolut i relatiu de les operacions elementals 2/2

Exemple: error en el càlcul de $x_1 x_2^2$

Suposant que les operacions es fan exactament, anem a determinar l'error efectuat en el càlcul de $x_1 x_2^2$, essent

$$\begin{cases} x_1 = 2.0 \pm 0.1, \\ x_2 = 3.0 \pm 0.2. \end{cases}$$

Traduint això al formalisme en què treballem, tenim

$$\varepsilon_a(\widetilde{x}_1, x_1) = 0.1 \implies \varepsilon_r(\widetilde{x}_1, x_1) = \frac{\varepsilon_a(\widetilde{x}_1, x_1)}{|\widetilde{x}_1|} = 0.05$$

$$\varepsilon_a(\widetilde{x}_2, x_2) = 0.2 \implies \varepsilon_r(\widetilde{x}_2, x_2) = \frac{\varepsilon_a(\widetilde{x}_2, x_2)}{|\widetilde{x}_2|} = 0.0667$$

Fent servir la proposició anterior apartat (e), obtenim

$$\begin{aligned} \varepsilon_r(\widetilde{x}_1 \widetilde{x}_2^2, x_1 x_2^2) &= \varepsilon_r(\widetilde{x}_1, x_1) + \varepsilon_r(\widetilde{x}_2^2, x_2^2) \\ &= \varepsilon_r(\widetilde{x}_1, x_1) + \varepsilon_r(\widetilde{x}_2, x_2) + \varepsilon_r(\widetilde{x}_2, x_2) \\ &= \varepsilon_r(\widetilde{x}_1, x_1) + 2\varepsilon_r(\widetilde{x}_2, x_2) \\ &= 0.05 + 2 \times 0.0667 = 0.183. \end{aligned}$$

Exemple: error en el càlcul de $x_1 x_2^2$ (cont.)

Passem a error absolut,

$$\varepsilon_a(\widetilde{x}_1\widetilde{x}_2^2, x_1x_2^2) = |\widetilde{x}_1\widetilde{x}_2^2|\varepsilon_r(\widetilde{x}_1\widetilde{x}_2^2, x_1x_2^2) = 3.30,$$

i, per tant, podem escriure,

$$x_1x_2^2 = 18 \pm 3.30.$$

Exemple: error en el càlcul de $x_1 x_2^2$ 2/2

Esmorteïment de l'error i cancel·lacions

Durant un càlcul llarg, en la major part de situacions estarem interessats en l'evolució de l'error relatiu, donat que està directament relacionat amb el nombre de xifres significatives que tenim al llarg del càlcul. Des d'aquest punt de vista, el producte i la divisió són operacions "segures". Cada cop que multipliquem o dividim dues quantitats afectades d'error, l'error relatiu del resultat "només" és la suma dels errors de les dades.

No passa el mateix amb sumes i restes. A partir dels resultats anteriors amb $f(x, y) = x + y$, s'obté:

$$\varepsilon_r(\widetilde{x} + \widetilde{y}, x + y) = \left|\frac{x}{x+y}\right| \varepsilon_r(\widetilde{x}, x) + \left|\frac{y}{x+y}\right| \varepsilon_r(\widetilde{y}, y). \qquad (5)$$

Esmorteïment de l'error i cancel·lacions (cont.)

D'aquesta fórmula, podem deduir:

- La suma d'operands del mateix signe també és "segura", al igual que el producte i la divisió.
- Si $|x| \ll |y|$,
$$\frac{|x|}{|x+y|} \leq \frac{|x|}{|y|-|x|} \ll 1$$
i, per tant, encara que $\varepsilon_r(\widetilde{x}, x)$ sigui gros, si tant el factor $|x|/|x+y|$ com $\varepsilon_r(\widetilde{y}, y)$ són prou petits, $\varepsilon_r(\widetilde{x}+\widetilde{y}, x+y)$ serà petit. D'això se'n diu *esmorteïment de l'error*, degut a que l'error relatiu al resultat és més petit que el màxim dels errors relatius dels arguments.
- Si x, y són de signes diferents, almenys un dels factors
$$\frac{|x|}{|x+y|}, \quad \frac{|y|}{|x+y|}$$
serà > 1 i l'error relatiu que porta multiplicant es veurà amplificat. Aquesta amplificació és dràstica si $x \approx -y$, és a dir, quan hi ha *cancel·lacions*, tal com hem comprovat abans.

Esmorteïment de l'error i cancel·lacions 2/2

Exemple: una cancel·lació pot eliminar tota la precisió que teníem i un esmorteïment la pot recuperar

Definim

$$\begin{aligned} a &= 0.326\,724 \pm 10^{-7} \implies \varepsilon_r(\widetilde{a}, a) = 3.06 \times 10^{-7} \\ b &= -0.326\,725 \pm 10^{-7} \implies \varepsilon_r(\widetilde{b}, b) = 3.06 \times 10^{-7} \\ c &= 0.248\,763 \pm 10^{-7} \implies \varepsilon_r(\widetilde{c}, c) = 4.02 \times 10^{-7} \end{aligned}$$

Suposem que fem les següents dues sumes de manera exacta:

$$\widetilde{d} = \widetilde{a} + \widetilde{b}, \qquad \widetilde{e} = \widetilde{d} + \widetilde{c}.$$

Trobem la fita de l'error relatiu de la primera suma:

$$\begin{aligned} \varepsilon_r(\widetilde{d}, d) &= \varepsilon_r(\widetilde{a} + \widetilde{b}, a + b) = \left|\frac{\widetilde{a}}{\widetilde{a} + \widetilde{b}}\right| \varepsilon_r(\widetilde{a}, a) + \left|\frac{\widetilde{b}}{\widetilde{a} + \widetilde{b}}\right| \varepsilon_r(\widetilde{b}, b) \\ &= 327\,000 \times 3.06 \times 10^{-7} + 327\,000 \times 3.06 \times 10^{-7} \\ &= 0.200. \end{aligned}$$

Exemple: cancel·lació i esmorteïment que es compensen 1/2

Exemple: una cancel·lació pot eliminar tota la precisió que teníem i un esmorteïment la pot recuperar (cont.)

L'error relatiu s'ha disparat. De fet, l'error absolut ve donat per

$$\varepsilon_a(\widetilde{a}+\widetilde{b}, a+b) = \varepsilon_a(\widetilde{a}, a) + \varepsilon_a(\widetilde{b}, b) = 2\times 10^{-7},$$

d'on $$a+b = -0.000\,001 \pm 2\times 10^{-7}$$

i és clar que a $\widetilde{a}+\widetilde{b}$ hem perdut 5 xifres significatives.

Continuem operant: en fer $\widetilde{e} = \widetilde{d}+\widetilde{c}$, podem fitar l'error relatiu per

$$\begin{aligned}
\varepsilon_r(\widetilde{e}, e) &= \varepsilon_r(\widetilde{d}+\widetilde{c}, d+c) = \left|\frac{\widetilde{d}}{\widetilde{d}+\widetilde{c}}\right|\varepsilon_r(\widetilde{d}, d) + \left|\frac{\widetilde{c}}{\widetilde{d}+\widetilde{c}}\right|\varepsilon_r(\widetilde{c}, c)\\
&= 4.01\times 10^{-6}\times 0.200 + 1.00\times 4.02\times 10^{-7}\\
&= 1.20\times 10^{-6}.
\end{aligned}$$

Al resultat final tenim error relatiu petit, malgrat haver tingut una cancel·lació fatal enmig del càlcul. L'esmorteïment l'ha compensada.

Exemple: cancel·lació i esmorteïment que es compensen 2/2

Propagació de l'error a través de l'aplicació d'una funció

Suposem que la funció s'avalua exactament. La *fórmula de propagació dels errors maximals* permet fitar la propagació de l'error a través de l'aplicació d'una funció qualsevol. El seu ús més freqüent és per a funcions d'una variable $\varphi : \mathbb{R} \to \mathbb{R}$, en la forma següent:

$$\begin{aligned} e_a(\varphi(\widetilde{x}), \varphi(x)) &= \varphi'(x) e_a(\widetilde{x}, x), \\ e_r(\varphi(\widetilde{x}), \varphi(x)) &= \frac{x\varphi'(x)}{\varphi(x)} e_r(\widetilde{x}, x), \\ \varepsilon_a(\varphi(\widetilde{x}), \varphi(x)) &= |\varphi'(x)| \varepsilon_a(\widetilde{x}, x), \\ \varepsilon_r(\varphi(\widetilde{x}), \varphi(x)) &= \frac{|x||\varphi'(x)|}{|\varphi(x)|} \varepsilon_r(\widetilde{x}, x). \end{aligned}$$

Propagació de l'error a través de l'aplicació d'una funció (cont.)

El nombre

$$\frac{|x||\varphi'(x)|}{|\varphi(x)|}$$

és el factor d'amplificació (o reducció) d'una fita de l'error relatiu d'una quantitat x en aplicar-li una funció qualsevol $\varphi(x)$ (que suposem que s'avalua exactament). És per això que es coneix com a *coeficient de propagació*.

Una aplicació immediata de la proposició anterior és la classificació d'expressions matemàticament equivalents des del punt de vista de la propagació d'errors. Vegem-ne un exemple.

Propagació de l'error a través de l'aplicació d'una funció 2/2

Exemple: Càlcul de $(3-2\sqrt{3})^4$ amb $\sqrt{3}\approx 1.73205$

Definim $x := \sqrt{3}$, $\widetilde{x} = 1.73205$, de manera que

$$\varepsilon_r(\widetilde{x}, x) = \frac{\varepsilon_a(\widetilde{x}, x)}{|\widetilde{x}|} = 4.66 \times 10^{-7}.$$

Una possibilitat per trobar $a = (3-2\sqrt{3})^4$ és fer $a = \varphi_1(x) := (3-2x)^4$. Suposant que poguéssim avaluar φ_1 exactament, obtindríem $\widetilde{a} = \varphi_1(\widetilde{x})$. Trobem la corresponent fita de l'error relatiu:

$$\begin{aligned}
\varepsilon_r(\varphi_1(\widetilde{x}), \varphi_1(x)) &= \left|\frac{\varphi_1'(\widetilde{x})\widetilde{x}}{\varphi_1(\widetilde{x})}\right| \varepsilon_r(\widetilde{x}, x) \\
&= \left|\frac{4(3-2\widetilde{x})^3(-2)\widetilde{x}}{(3-2\widetilde{x})^4}\right| \varepsilon_r(\widetilde{x}, x) \\
&= 29.86 \times 4.66 \times 10^{-7} = 1.39 \times 10^{-5}.
\end{aligned}$$

El coeficient de propagació corresponent a aquest càlcul és 29.86.

Exemple: Càlcul de $(3-2\sqrt{3})^4$ amb $\sqrt{3}\approx 1.73205$ (cont.)

Pensant en millorar la propagació de l'error, podem pensar a fer $a=(3-2\sqrt{3})^4=(21-12\sqrt{3})^2$ i avaluar, per tant, $a=(21-12x)^2=:\varphi_2(x)$. Obtenim la corresponent fita de l'error relatiu,

$$\begin{aligned}\varepsilon_r(\varphi_2(\widetilde{x}),\varphi_2(x)) &= \left|\frac{\varphi_2'(\widetilde{x})\widetilde{x}}{\varphi_2(\widetilde{x})}\right|\varepsilon_r(\widetilde{x},x)\\ &= \left|\frac{2(21-12\widetilde{x})(-12)\widetilde{x}}{(21-12\widetilde{x})^2}\right|\varepsilon_r(\widetilde{x},x)\\ &= 193.00\times 4.66\times 10^{-7} \;=\; 8.99\times 10^{-5},\end{aligned}$$

que surt pitjor que el cas anterior. El factor de propagació és, en aquest cas, 193.00.

Exemple: Càlcul de $(3-2\sqrt{3})^4$ amb $\sqrt{3}\approx 1.73205$ (cont.)

Una altra possibilitat és $a=(21-12\sqrt{3})^2=873-504x=:\varphi_3(x)$. La fita, en aquest cas, és

$$\begin{aligned}\varepsilon_r(\varphi_3(\widetilde{x}),\varphi_3(x)) &= \left|\frac{\varphi_3'(\widetilde{x})\widetilde{x}}{\varphi_3(\widetilde{x})}\right|\varepsilon_r(\widetilde{x},x)\\ &= \left|\frac{-504\widetilde{x}}{873-504\widetilde{x}}\right|\varepsilon_r(\widetilde{x},x)\\ &= 18\,817\times 4.66\times 10^{-7} \;=\; 8.77\times 10^{-3},\end{aligned}$$

que és molt pitjor (factor de propagació 18 817). De fet, en aquest cas, la fórmula de propagació de l'error maximal està detectant el fent que, en avaluar $873-504x$, es produeix una cancel·lació.

Exemple: Càlcul de $(3-2\sqrt{3})^4$ amb $\sqrt{3}\approx 1.73205$ (cont.)

En canvi, si "desracionalitzem",

$$a=(873-504\sqrt{3})\frac{873+504\sqrt{3}}{873+504\sqrt{3}}=\frac{9}{97+56x}=:\varphi_4(x),$$

i la fita de l'error relatiu queda,

$$\begin{aligned}\varepsilon_r(\varphi_4(\widetilde{x}),\varphi_4(x)) &= \left|\frac{\varphi_4'(\widetilde{x})\widetilde{x}}{\varphi_4(\widetilde{x})}\right|\varepsilon_r(\widetilde{x},x)\\ &= \left|\frac{-56\widetilde{x}}{97+56\widetilde{x}}\right|\varepsilon_r(\widetilde{x},x)\\ &= 0.50\times 4.66\times 10^{-7} \;=\; 2.33\times 10^{-7},\end{aligned}$$

amb factor de propagació 0.50, que és el millor amb diferència.

Per tant, des del punt de vista numèric, la millor fórmula per a avaluar $(3-2\sqrt{3})^4$ és $9/(97+56\sqrt{3})$. De fet, aquest fet és independent del nombre de decimals que es prenguin a l'aproximació de $\sqrt{3}$.

Exemple: $\cos(\cos(\sqrt{x}))$

Volem calcular la propagació de l'error al avaluar $\cos(\cos(\sqrt{x}))$ a $x \in [0,1]$ suposant que

- els errors de representació són ε,
- els errors al producte són 3ε,
- els errors a l'$\sqrt{\cdot}$ són 3ε, i
- els errors al $\cos(\cdot)$ són 5ε.

Tenim: $\mathrm{fl}(x) = x(1+\varepsilon_1)$. Llavors,

$$\begin{aligned}
\mathrm{fl}\left(\sqrt{x}\right) &= \sqrt{x(1+\varepsilon_1)}(1+3\varepsilon_2) = \sqrt{x}(1+\varepsilon_1)^{1/2}(1+3\varepsilon_2) \\
&= \sqrt{x}\left(1+\frac{\varepsilon_1}{2}+\mathcal{O}(\varepsilon_1^2)\right)(1+3\varepsilon_2) \\
&= \sqrt{x}\left(1+3\varepsilon_2+\frac{\varepsilon_1}{2}+\frac{3}{2}\varepsilon_1\varepsilon_2+\mathcal{O}(\varepsilon_1^2)+\mathcal{O}(\varepsilon_1^2\cdot\varepsilon_2)\right) \\
&\approx \sqrt{x}\left(1+\frac{7}{2}\varepsilon_3\right),
\end{aligned}$$

amb $\max\{|\varepsilon_1|, |\varepsilon_2|\} = |\varepsilon_3| \leq \varepsilon$.

Exemple: $\cos(\cos(\sqrt{x}))$ 1/3

Exemple: $\cos(\cos(\sqrt{x}))$ (cont.)

Seguidament,

$$\begin{aligned}\mathrm{fl}\left(\cos\sqrt{x}\right) &\approx \cos\left(\sqrt{x}(1+\frac{7}{2}\varepsilon_3)\right)(1+5\varepsilon_4)\\ &= \left(\cos\sqrt{x} - (\sin\sqrt{x})\sqrt{x}\frac{7}{2}\varepsilon_3 + \mathcal{O}\left(\left(\sqrt{x}\frac{7}{2}\varepsilon_3\right)^2\right)\right)(1+5\varepsilon_4)\\ &\lessapprox \cos\sqrt{x} + \varphi(x)\varepsilon_5,\end{aligned}$$

amb

$$\varphi(x) = 5\left|\cos\sqrt{x}\right| + \frac{7}{2}\left|\sin\sqrt{x}\right|\sqrt{x}.$$

Nota:: A la primera igualtat, hem usat

$$\cos(y+h) = \cos(y) - sin(y)h + \mathcal{O}(h^2).$$

Exemple: $\cos(\cos(\sqrt{x}))$ 2/3

Exemple: $\cos(\cos(\sqrt{x}))$ (cont.)

Finalment,

$$\begin{aligned}
\mathrm{fl}\left(\cos(\cos\sqrt{x})\right) &\lessapprox \cos\left(\cos\sqrt{x}+\varphi(x)\varepsilon_5\right)(1+5\varepsilon_6)\\
&= \left(\cos(\cos\sqrt{x})-\sin(\cos\sqrt{x})\varphi(x)\varepsilon_5+\mathcal{O}(2)\right)(1+5\varepsilon_6)\\
&\lessapprox \cos(\cos\sqrt{x})+\left[\left|\sin(\cos\sqrt{x})\varphi(x)\right|+5\left|\cos(\cos\sqrt{x})\right|\right]\varepsilon_7\\
&= \cos(\cos\sqrt{x})\left[1+\left(\left|\tan(\cos\sqrt{x})\varphi(x)\right|+5\right)\varepsilon_7\right]
\end{aligned}$$

on $\mathcal{O}(2)$ denota $\mathcal{O}\left((\varphi(x)\varepsilon_5)^2\right)$.

Com que $0 \le x \le 1$, tenim: $\left|\sqrt{x}\right| \le 1$, $\left|\cos\sqrt{x}\right| \le 1$, i $\left|\sin\sqrt{x}\right| \le 1$. Per tant,

$$\left|\tan(\cos\sqrt{x})\varphi(x)\right|+5 \le \left|\tan(1)(5+\tfrac{7}{2})+5\right| \approx 18.24 < 19.$$

Resumint: $\mathrm{fl}\left(\cos(\cos\sqrt{x})\right) \lessapprox \cos(\cos(\sqrt{x}))(1+19\varepsilon)$,
i l'error relatiu acumulat és 19ε.

Exemple: $\cos(\cos(\sqrt{x}))$ 3/3

Exemple combinant els errors a les operacions i la propagació d'errors a les dades

Volem fitar l'error relatiu comès en avaluar

$$f(x) = \sqrt{3 + (\ln x)^2}$$

tenint en compte els errors a les operacions. Suposem que els errors relatius comesos en la representació de nombres, operacions aritmètiques, arrels quadrades i logaritmes estan fitats per ε, 2ε, 3ε i 5ε, respectivament. A més, suposem que en mesurar x cometem un error relatiu fitat per 4ε.

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

Si desenvolupem com fèiem a la secció anterior, quan consideràvem errors a les operacions obtenim:

$$
\begin{aligned}
&\mathrm{fl}(\sqrt{3+(\ln x)^2}) \\
&= \sqrt{\mathrm{fl}(3+(\ln x)^2)}(1+\delta_5) \\
&= \sqrt{\left(3+\mathrm{fl}((\ln x)^2)\right)(1+\delta_4)}(1+\delta_5) \\
&= \sqrt{\left(3+(\mathrm{fl}(\ln x))^2(1+\delta_3)\right)(1+\delta_4)}(1+\delta_5) \\
&= \sqrt{\left[3+\Big((\ln \mathrm{fl}(x))(1+\delta_2)\Big)^2(1+\delta_3)\right](1+\delta_4)}(1+\delta_5) \\
&= \sqrt{\left[3+\Big((\ln(x(1+\delta_0)(1+\delta_1)))(1+\delta_2)\Big)^2(1+\delta_3)\right](1+\delta_4)}(1+\delta_5)
\end{aligned}
$$

on

$$
\begin{array}{llll}
|\delta_0| \leq 4\varepsilon & \text{(mesura)}, & |\delta_3| \leq 2\varepsilon & \text{(producte)}, \\
|\delta_1| \leq \varepsilon & \text{(representació)}, & |\delta_4| \leq 2\varepsilon & \text{(suma)}, \\
|\delta_2| \leq 5\varepsilon & \text{(ln)}, & |\delta_5| \leq 3\varepsilon & (\sqrt{\ }).
\end{array}
$$

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

Per a acabar, hauríem de trobar δ tal que

$$\begin{aligned}&\sqrt{3+(\ln x)^2})(1+\delta)\\ &= \sqrt{\left[3+\Big((\ln(x(1+\delta_0)(1+\delta_1)))(1+\delta_2)\Big)^2(1+\delta_3)\right](1+\delta_4)(1+\delta_5)},\end{aligned}$$

hauríem de fitar $|\delta|$ en termes de les fites de les $|\delta_i|$, i aquesta fita seria la resposta.

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

Una estratègia més senzilla és acumular aproximacions dels errors a primer ordre usant la *fórmula de propagació de l'error maximal*. Trenquem el càlcul de $f(x)$ en operacions elementals i considerem els valors exactes i aproximats a cada etapa:

$$
\begin{array}{lcllcll}
\widetilde{x}_0 & := & \text{mesura}(x), & x_0 & := & x, & \\
\widetilde{x}_1 & := & \text{fl}(\widetilde{x}_0), & x_1 & := & x_0 & = x, \\
\widetilde{x}_2 & := & \text{fl}(\ln \widetilde{x}_1), & x_2 & := & \ln x_1 & = \ln x, \\
\widetilde{x}_3 & := & \text{fl}(\widetilde{x}_2^2), & x_3 & := & x_2{}^2 & = (\ln x)^2 \\
\widetilde{x}_4 & := & \text{fl}(3 + \widetilde{x}_3), & x_4 & := & 3 + x_3 & = 3 + (\ln x)^2, \\
\widetilde{x}_5 & := & \text{fl}(\sqrt{\widetilde{x}_4}), & x_5 & := & \sqrt{x_4} & = \sqrt{3 + (\ln x)^2}.
\end{array}
$$

Se'ns demana donar un valor per $\varepsilon_r(\widetilde{x}_5, x_5)$. Anem a trobar progressivament: $\varepsilon_r(\widetilde{x}_0, x_0)$, $\varepsilon_r(\widetilde{x}_1, x_1)$, $\varepsilon_r(\widetilde{x}_2, x_2)$,...

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

- Sigui $\delta_0 = e_r(\text{mesura}(x))$, d'on $\text{mesura}(x) = x(1+\delta_0)$, per a $|\delta_0| \leq 4\varepsilon$. Aleshores,

$$\varepsilon_r(\widetilde{x}_0, x_0) \quad = \quad \varepsilon_r\big(x(1+\delta_0), x\big) \quad = \quad 4\varepsilon.$$

- Sigui $\delta_1 = e_r(\text{fl}(\widetilde{x}_0), x_0)$, d'on $\text{fl}(\widetilde{x}_0) = \widetilde{x}_0(1+\delta_1)$, per a $|\delta_1| \leq \varepsilon$. Aleshores

$$\begin{aligned} \varepsilon_r(\widetilde{x}_1, x_1) \quad &= \quad \varepsilon_r(\widetilde{x}_0(1+\delta_1), x_0). \\ &= \quad \varepsilon_r(\widetilde{x}_0, x_0) + \varepsilon_r(1+\delta_1, 1) \quad = \quad 4\varepsilon + \varepsilon \quad = \quad 5\varepsilon. \end{aligned}$$

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

- Sigui $\delta_2 = e_r(\mathrm{fl}(\ln \widetilde{x}_1), \ln \widetilde{x}_1)$, d'on $\mathrm{fl}(\ln \widetilde{x}_1) = \ln \widetilde{x}_1(1+\delta_2)$, per a $|\delta_2| \leq 5\varepsilon$. Aleshores,

$$\begin{aligned}
\varepsilon_r(\widetilde{x}_2, x_2) &= \varepsilon_r\big(\mathrm{fl}(\ln \widetilde{x}_1), \ln x_1\big) \\
&= \varepsilon_r\big(\ln \widetilde{x}_1(1+\delta_2), \ln x_1\big) \\
&= \varepsilon_r(\ln \widetilde{x}_1, \ln x_1) + \varepsilon_r(1+\delta_2, 1) \qquad (6) \\
&= \frac{|x_1|\frac{1}{|x_1|}}{|\ln x_1|}\varepsilon_r(\widetilde{x}_1, x_1) + 5\varepsilon \;=\; \Big(\frac{1}{|\ln x|} + 1\Big)5\varepsilon.
\end{aligned}$$

Noteu que, a (6), el primer terme correspon a la propagació de l'error acumulat ($\varepsilon_r(\widetilde{x}_1, x_1)$), mentre que el segon terme és el nou error introduït degut a l'avaluació del logaritme en punt flotant.

Exemple combinant els errors a les operacions i la propagació d'errors a les dades (cont.)

- Sigui $\delta_3 = e_r(\mathrm{fl}(\widetilde{x}_2^2, \widetilde{x}_2^2))$, d'on $\mathrm{fl}(\widetilde{x}_2^2) = \widetilde{x}_2^2(1+\delta_3)$, $|\delta_3| \leq 2\varepsilon$.

Aleshores,

$$\begin{aligned}
\varepsilon_r(\widetilde{x}_3, x_3) &= \varepsilon_r(\mathrm{fl}(\widetilde{x}_2^2), x_2{}^2) = \varepsilon_r(\widetilde{x}_2^2(1+\delta_3), x_2^2) \\
&= \varepsilon_r(\widetilde{x}_2^2, x_2^2) + \varepsilon_r(1+\delta_3, 1) \qquad (7) \\
&= \frac{|x_2|2|x_2|}{|x_2|^2}\varepsilon_r(\widetilde{x}_2, x_2) + 2\varepsilon = 2\Big(\frac{1}{|\ln x|}+1\Big)5\varepsilon + 2\varepsilon \\
&= \frac{12|\ln x| + 10}{|\ln x|}\varepsilon.
\end{aligned}$$

A (7) el primer terme correspon a la propagació de l'error acumulat ($\varepsilon_r(\widetilde{x}_2, x_2)$), i el segon és l'error introduït pel producte.

Si continuem el procés, acabem amb

$$\varepsilon_r(\widetilde{x}_5, x_5) = \frac{12 + 5|\ln x| + 10|\ln x|^2}{3 + |\ln x|^2}\varepsilon,$$

que és el què se'ns demanava.

Algorismes estables i inestables. Problemes mal condicionats

Definició

Direm que un algorisme és *numèricament inestable* quan petites pertorbacions dels resultats inicials produeixen una gran diferència en el resultat final. Això el farà inservible des del punt de vista numèric, donat que amplificarà de manera sistemàtica els errors de les dades inicials.

Exemple d'algorisme numèricament inestable

Considerem la família d'integrals

$$E_n = \int_0^1 x^n e^{x-1} dx.$$

Més endavant veurem mètodes per a aproximar integrals definides, però ara volem fer una cosa més senzilla i més eficient: integrant per parts, es veu de seguida que

$$E_n = 1 - nE_{n-1},$$

i és immediat calcular que

$$E_0 = \int_0^1 e^{x-1} = 1 - e^{-1}.$$

Això suggereix el següent procediment per trobar E_n:

$E_0 := 1 - e^{-1}$
$\forall i = 1 \div n$
  $E_i := 1 - iE_{i-1}$

Exemple d'algorisme numèricament inestable (cont.)

Suposem que volem trobar E_9. Prenem una calculadora i treballem amb 6 xifres decimals:

n	E_n	n	E_n
0	0.632121	5	0.145480
1	0.367879	6	0.127120
2	0.264242	7	0.110160
3	0.207274	8	0.118720
4	0.170904	9	-0.0684800

Obtenim E_9 negatiu, mentre que hauria de ser positiu, donat que estem integrant coses positives. Per tant, E_9 no té cap xifra significativa!

Exemple d'algorisme numèricament inestable 2/6

Exemple d'algorisme numèricament inestable (cont.)

Ara farem una anàlisi de l'error que ens permetrà predir aquest comportament per després corregir-lo. Suposem que les operacions són exactes (o dit altrament, donat que el nombre d'operacions que fem és de l'ordre de desenes, els errors introduïts són menyspreables al costat dels errors que estem observant), l'única font d'error és l'error de representació d'E_0, que és

$$e_0 := 4.41 \times 10^{-7},$$

de manera que no comencem amb E_0 sinó amb $\widetilde{E}_0 = E_0 + e_0$, i llavors anem fent $\widetilde{E}_i = 1 - i\widetilde{E}_{i-1}$. Si anomenem e_i l'error en el càlcul d'$\widetilde{E}_i$, tenim inductivament

$$e_i = \widetilde{E}_i - E_i = 1 - i\widetilde{E}_{i-1} - (1 - iE_{i-1}) = -i(\widetilde{E}_{i-1} - E_{i-1}) = -ie_{i-1}.$$

Observem que a cada pas l'error es multiplica per i.

Exemple d'algorisme numèricament inestable (cont.)

Així,

$$e_n = -ne_{n-1} = n(n-1)e_{n-2} = \cdots = (-1)^n n! e_0$$

d'on $e_9 = 9!e_0 = 3.63 \times 10^5 \times 4.41 \times 10^{-7} = 1.61 \times 10^{-1}$, que fa que l'error relatiu d'$\widetilde{E}_9$ sigui de l'ordre de la unitat i per tant perdem totes les xifres significatives.

Per a arreglar-lo, observem que, si iterem cap a enrere, l'error disminueix en comptes d'augmentar. En efecte, la recurrència endarrere ens queda:

$$E_{i-1} = \frac{1 - E_i}{i},$$

i, per les integrals aproximades,

$$\widetilde{E}_{i-1} = \frac{1 - \widetilde{E}_i}{i}.$$

Exemple d'algorisme numèricament inestable (cont.)

Llavors, l'error a la integral i ens queda:

$$e_{i-1} = \widetilde{E}_{i-1} - E_{i-1} = \frac{1-\widetilde{E}_i}{i} - \frac{1-E_i}{i} = -\frac{e_i}{i}.$$

Com que $\widetilde{E}_n = E_n + e_n$, l'error a la integral n, que és la que volem calcular, ens queda:

$$e_n = -\frac{e_{n+1}}{n+1} = \frac{e_{n+2}}{(n+1)(n+2)} = \cdots = \frac{(-1)^k n!}{(n+k)!} e_{n+k}.$$

Per tant, per a calcular E_n amb precisió, n'hi ha prou d'escollir $\widetilde{E}_{n+k}$ (i per tant e_{n+k}) de manera que, en iterar enrere k vegades, el factor $\frac{n!}{(n+k)!}$ hagi eliminat e_{n+k}.

Exemple d'algorisme numèricament inestable (cont.)

Per a fer això, observem que $E_{n+k} \leq E_{n+k-1} \leq \cdots \leq E_0$ d'on, si prenem $\widetilde{E}_{n+k} = 0$, tindrem $|e_{n+k}| = \left|\widetilde{E}_{n+k} - E_{n+k}\right| = |E_{n+k}| \leq E_0$, i, per tant,

$$|e_n| \leq \frac{n!}{(n+k)!}(1 - e^{-1}).$$

Així, per trobar E_9 amb 6 dígits, podem prendre $k = 7$, donat que

$$|e_9| \leq \frac{9!}{(9+7)!}(1 - e^{-1}) = 1.096 \times 10^{-8}.$$

Els iterats donen:

n	$\widetilde{E}_n$	n	$\widetilde{E}_n$
16	0	12	0.0717720
15	0.0625000	11	0.0773523
14	0.0625000	10	0.0838771
13	0.0669643	9	0.0916123

El valor d'E_9 amb 11 decimals correctes és 0.09161229299.

Exemple d'algorisme numèricament inestable 6/6

Problemes ben i mal condicionats

Com hem vist a l'exemple anterior, quan tenim un algorisme numèricament inestable per resoldre un determinat problema, hem de mirar de modificar-lo per desfer-nos de la inestabilitat.

Pot passar que, per a un determinat problema, petites variacions en les dades d'entrada produeixin grans variacions en els resultats finals, independentment de l'algorisme emprat en la seva resolució. En aquest cas parlarem de *problemes mal condicionats*.

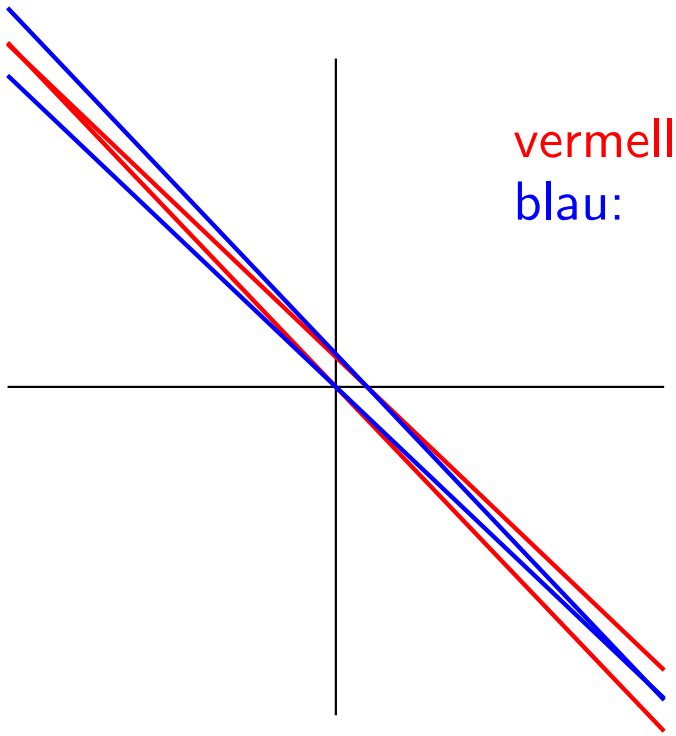
A continuació veurem un exemple senzill de problema mal condicionat.

Un sistema lineal mal condicionat

Considerem el sistema

$$\left.\begin{array}{rcrcl} x & + & \alpha y & = & 1 \\ \alpha x & + & y & = & 0 \end{array}\right\}$$

que té per solució $x(\alpha) = \frac{1}{1-\alpha^2}$ $y(\alpha) = -\frac{\alpha}{1-\alpha^2}$,
amb $\alpha = 1 \pm 10^{-3}$.



vermell: $\alpha = 1 + 10^{-3}$
blau: $\alpha = 1 - 10^{-3}$

Un sistema lineal mal condicionat 1/2

Un sistema lineal mal condicionat (cont.)

Anem a calcular el *nombre de condició* de x:

$$\Delta x \lesssim \left| \frac{2\alpha}{(1-\alpha^2)^2} \right| \Delta\alpha.$$

Per $\alpha = 1 + 10^{-3}$ tenim

$$\text{nombre de condició} = \left| \frac{2\alpha}{(1-\alpha^2)^2} \right| \approx \frac{2}{4 \cdot 10^{-6}} = \frac{1}{2} 10^6.$$

És a dir, els errors s'amplifiquen en un factor de 500,000. Això justifica que, amb variacions de $2 \cdot 10^{-3}$ a alpha, la solució del sistema passa de $x \approx -500$ a $x \approx 500$.

En canvi, si en lloc de voler calcular la solució del sistema volem calcular $S(\alpha) = x(\alpha) + y(\alpha) = \frac{1}{1+\alpha}$:

$$\Delta S \lesssim \left| \frac{1}{(1+\alpha)^2} \right| \Delta\alpha \approx \frac{1}{2} \Delta\alpha,$$

que és un càlcul molt més fiable.

Un sistema lineal mal condicionat 2/2

Exemple (Wilkinson, 1963)

Considerem el polinomi

$$\begin{aligned} p(x) &:= (x-1)(x-2)\dots(x-19)(x-20) \\ &= x^{20} + a_1 x^{19} + a_2 x^{18} + a_3 x^{17} + \cdots + a_{20}, \end{aligned}$$

essent

$$\begin{aligned} a_1 &= -210, & a_5 &= -1\,672\,280\,820, \\ a_2 &= 20\,615, & &\vdots \\ a_3 &= -1\,256\,850, & a_{20} &= 20! \\ a_4 &= 53\,327\,946, & & \end{aligned}$$

té per arrels $1, 2, \dots, 20$. Pertorbem-lo lleugerament i considerem

$$p(x) + 2^{-23} x^{19}.$$

Exemple (Wilkinson, 1963) 1/5

Exemple (Wilkinson, 1963) (cont.)

Aquest nou polinomi té arrels

$$\begin{aligned}
&1.00000, \quad 6.00001, \quad 10.09527 \pm 0.64350i,\\
&2.00000, \quad 6.99970, \quad 11.79363 \pm 1.65233i,\\
&3.00000, \quad 8.00727, \quad 13.99236 \pm 2.51883i,\\
&4.00000, \quad 8.91725, \quad 16.73074 \pm 2.81262i,\\
&5.00000, \quad 20.84691, \quad 19.50244 \pm 1.94033i.
\end{aligned}$$

Veiem que un petit canvi ($2^{-23} \simeq 1.192 \times 10^{-7}$) en el coeficient a_1 ha causat que 10 zeros passin a ser complexos i dos d'ells estiguin allunyats 2.8 unitats de l'eix real.

El problema no és d'inestabilitat numèrica de l'algorisme emprat per trobar els zeros (que no és el cas), sinó de mal-condicionament del problema.

Exemple (Wilkinson, 1963) 2/5

Exemple (Wilkinson, 1963) (cont.)

Escrivim

$$p(x, a_1) := x^{20} + a_1 x^{19} + \cdots + a_{19} x + a_{20}.$$

L'equació

$$p(x, a_1) = 0$$

defineix x implícitament com a funció d'a_1. La seva derivada, dx/da_1, ens donarà la variació de les arrels de $p(x)$ respecte del coeficient a_1. Per a trobar-les, derivem implícitament

$$0 = \frac{d}{da_1} p(x, a_1) = \frac{\partial p}{\partial x}\frac{dx}{da_1} + \frac{\partial p}{\partial a_1},$$

d'on

$$\left.\frac{dx}{da_1}\right|_{a_1=-210} = \left.\frac{-\partial p/\partial a_1}{\partial p/\partial x}\right|_{a_1=-210} = \frac{-x^{19}}{\sum_{i=1}^{20}\prod_{j=1, j\neq i}^{20}(x-j)}.$$

Exemple (Wilkinson, 1963) 3/5

Exemple (Wilkinson, 1963) (cont.)

La sensibilitat de cada arrel respecte d'a_1 s'obté avaluant la quantitat anterior a cada arrel, i.e.,

$$\left.\frac{\partial x}{\partial a_1}\right|_{a_1=-210, x=i} = \frac{-i^{19}}{\sum_{i=1}^{20}\prod_{j=1, j\neq i}^{20}(i-j)}.$$

Els valors són:

i	$\partial x/\partial a_1\vert_{x=i}$	i	$\partial x/\partial a_1\vert_{x=i}$	i	$\partial x/\partial a_1\vert_{x=i}$	i	$\partial x/\partial a_1\vert_{x=i}$
1	-8.2×10^{-18}	6	5.8×10^{-1}	11	-4.6×10^{7}	16	2.4×10^{9}
2	8.2×10^{-11}	7	-2.5×10^{3}	12	2.0×10^{8}	17	-1.9×10^{9}
3	-1.6×10^{-6}	8	6.0×10^{4}	13	-6.1×10^{8}	18	1.0×10^{9}
4	2.2×10^{-3}	9	-8.3×10^{5}	14	1.3×10^{9}	19	-3.1×10^{8}
5	-6.1×10^{-1}	10	7.6×10^{6}	15	-2.1×10^{9}	20	4.3×10^{7}

És clar que, llevat de les primeres arrels, totes són molt sensibles a petites variacions d'a_1.

Exemple (Wilkinson, 1963) 4/5

Exemple (Wilkinson, 1963) (cont.)

D'això se segueix que, tot i que $p(x)$ tingui arrels enteres ben fàcils, si proveu de determinar-les numèricament, els errors de representació dels coeficients (i.e., $e_r\big(\mathrm{fl}(a_1)\big), \ldots, e_r\big(\mathrm{fl}(a_{20})\big)$) introduiran errors considerables en les arrels. Si us voleu convèncer, proveu de fer

Maple

```
expand(mul((x-i),i=1..20));
p:=evalf(%);
solve(p=0,x);
```

per diversos valors de `Digits`.

Exemple (Wilkinson, 1963) 5/5

Errors

$$
\begin{aligned}
a &= b \\
a^2 &= ab \\
a^2 - b^2 &= ab - b^2 \\
(a+b)(a-b) &= b(a-b) \\
a + b &= b \\
2b &= b \\
2 &= 1
\end{aligned}
$$

Why do programmers always mix up Halloween and Christmas?

Because Oct 31 = Dec 25

Hi ha 10 tipus de persones en aquest món: els que entenen el binari i els que no.