

Apunts de Mètodes Numèrics

Grau en Matemàtiques UAB, 2025–2026

Josep Maria Mondelo
Departament de Matemàtiques UAB

11 de maig de 2026

Capítol 1

Anàlisi de l'error

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 fem 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 capítol tractarem els dos primers tipus d'error. Dels errors de truncament, que són específics de cada mètode, ens n'ocuparem en capítols posteriors.

1.1 Errors de representació

1.1.1 Formats de punt flotant

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

Teorema 1.1.1 *Fixem $b \in \mathbb{N}$, $b \geq 2$. Tot $x \in \mathbb{R}$, $x \neq 0$ pot ser representat de la forma*

$$x = s \left(\sum_{i=1}^{\infty} \alpha_i b^{-i} \right) b^q, \quad (1.1)$$

amb $s \in \{-1, 1\}$, $q \in \mathbb{Z}$ i $\alpha_i \in \{0, 1, \dots, b-1\}$. 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, és a dir,

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

Escriurem (1.1) abreviadament com

$$x = s(0.\alpha_1\alpha_2\alpha_3\dots)_b b^q. \quad (1.2)$$

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

Exemple 1.1.2 (representacions normalitzades)

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

▽

Exemple 1.1.3 Suposem $x = (0.1)_{10}$, $b = 2$. Per passar a binari un nombre que només té part decimal, només cal multiplicar-lo per dos, prendre la part entera com el primer dígit, quedar-se amb la part decimal, multiplicar-la per dos, prendre la part entera com el segon dígit, quedar-se amb la part decimal i així successivament.

$$\begin{array}{rcl}
 0.1 \times 2 & = & 0.2 \longrightarrow \boxed{0} \\
 0.2 \times 2 & = & 0.4 \longrightarrow \boxed{0} \\
 0.4 \times 2 & = & 0.8 \longrightarrow \boxed{0} \\
 0.8 \times 2 & = & 1.6 \longrightarrow \boxed{1} \\
 0.6 \times 2 & = & 1.2 \longrightarrow \boxed{1}
 \end{array}$$

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

$$(0.1)_{10} = (0.000\widehat{11})_2 = (0.\widehat{1100})_2 \times 2^{-3}.$$

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

▽

La representació numèrica més emprada en ordinadors i calculadores és la coneguda com a *representació en punt flotant*, que és la versió finita de la representació (1.2). En aquesta representació, tot nombre x consta de

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

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

D'acord amb la notació introduïda a (1.1), un nombre representat en punt flotant s'escriu

$$x = s(0.\delta_1\delta_2 \dots \delta_t)_b b^q \quad (1.3)$$

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

$$x = smb^q,$$

amb

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

Direm que un nombre en punt flotant x donat per (1.3) està **normalitzat** si $\delta_1 \neq 0$ (i.e., $m \geq 1/b$). Llevat de casos excepcionals, considerarem sempre nombres normalitzats.

Les bases més habituals són $b = 2$, emprada per la pràctica totalitat d'ordinadors, i $b = 10$, emprada per moltes calculadores. Tant el nombre de dígit, 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.

A la taula 1.1 es mostren els paràmetres d'alguns formats de punt flotant existents. La major part d'ordinadors implementen els formats IEEE i, quan és així, els tipus `float` i `double` del llenguatge C corresponen a les aritmètiques IEEE-simple i doble, respectivament. Com a exemple de formats de punt flotant de calculadora, es mostren els de les calculadores HP series 48,49,50. 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. 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 standard és del de representació en punt flotant, com suposarem més endavant.

Format	b	t	q_{min}	q_{max}	bits
IEEE simple	2	24	-125	128	32
IEEE doble	2	53	-1021	1024	64
HP real	10	12	-498	500	64
HP real estès	10	15	-49998	50000	84

Taula 1.1: Paràmetres d'alguns formats de punt flotant.

IEEE simple	s (1)	$e = q + 126$ (8)	Mantissa (23)
IEEE doble	s (1)	$e = q + 1022$ (11)	Mantissa (52)
HP real	s (1)	Mantissa (12)	q (3)
HP real estès	s (1)	Mantissa (15)	q (5)

Figura 1.1: Distribució a la memòria del signe, mantissa i exponent per als formats de punt flotant de la taula 1.1. El bit de signe (nibble, per HP) és 0 per nombres positius i 1 per nombres negatius.

A la figura 1.1 es mostra la distribució a la memòria del signe, mantissa i exponent per als formats de la taula 1.1. Per als formats IEEE, que són binaris, s'indica entre parèntesis la longitud de cada camp en bits. Per als formats HP-48, que són decimals, s'indica la longitud en *nibbles* (grups de 4 bits). Cada *nibble* es fa servir per representar un dígit decimal¹. Noteu que es perden els números del 10 (1010 en binari) al 15 (1111). Noteu també que, per als formats IEEE, la longitud de la mantissa és $t - 1$ (t de la taula 1.1). El motiu és que el primer dígit sempre és 1 i ja no es guarda.

¹Usar quatre bits per representar un dígit decimal és una codificació standard que es coneix com a BCD (Binary-Coded Decimal).

Observació 1.1.4 El camp e del format IEEE simple de la figura 1.1 compleix $e_{q_{\min}} \leq e \leq e_{q_{\max}}$, on

$$e_{q_{\min}} = q_{q_{\min}} + 126 = 1, \quad e_{q_{\max}} = q_{q_{\max}} + 126 = 254.$$

Noteu que no es fan servir els valors $e = 0$ i $e = 255$, tot i que són representables amb 8 bits. El motiu és que $e = 255$ s'utilitza per representar l'infinit (resultat d'*overflow*) i una classe de números anomenats NaN (Not a Number), que es fan servir com a resultats d'operacions invàlides (com dividir per zero o voler calcular l'arrel d'un número negatiu). El cas $e = 0$ s'usa per representar números no normalitzats (resultat d'*underflows*).

El format IEEE doble és totalment anàleg. En aquest cas no s'usen ni $e = 2047$ (infinits i NaN) ni $e = 0$ (números no normalitzats). ∇

1.1.2 Representació en punt flotant

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 1.1.5 Denotarem el conjunt de nombres representables exactament en aritmètica de punt flotant de t dígit, en base b i amb exponent entre q_{\min} i q_{\max} per $F(b, t, q_{\min}, q_{\max})$. És a dir,

$$\begin{aligned} F(b, t, q_{\min}, q_{\max}) \\ = \{0\} \cup \{\pm(0.\delta_1 \dots \delta_t)_b b^q, \delta_i \in \{0, 1, \dots, b-1\}, \delta_1 \neq 0, q_{\min} \leq q \leq q_{\max}\} \end{aligned}$$

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

Donat un nombre $x \notin F(b, t, q_{\min}, q_{\max})$, l'ordinador no pot operar amb aquest nombre. Ho ha de fer amb una aproximació seva $\tilde{x} \in F(b, t, q_{\min}, q_{\max})$, que s'obté a partir de x per truncament o arrodoniment.

Definició 1.1.6 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 1.1.1, i suposem $q_{\min} \leq q \leq q_{\max}$.

- Anomenarem **representació en punt flotant per truncament** de x a $fl_T(x) \in F(b, t, q_{\min}, q_{\max})$ definit per

$$fl_T(x) = s(0.\alpha_1 \alpha_2 \dots \alpha_t) b^q,$$

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

- Anomenarem **representació en punt flotant per arrodoniment** de x a $fl_A(x) \in F(b, t, q_{\min}, q_{\max})$ definit per

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

Noteu que, si $\alpha_t = b - 1$, $s(0.\alpha_1 \dots \alpha_{t-1}(\alpha_t + 1))b^q$ encara és de $F(b, t, q_{\min}, q_{\max})$ però la seva mantissa no és $(0.\alpha_1 \dots \alpha_{t-1}(\alpha_t + 1))$, i potser el seu exponent no és q .²

Exemple 1.1.7 Per $t = 4$, $b = 10$

- $fl_A(0.10004) = 0.1000 \times 10^0$,
- $fl_A(0.10005) = 0.1001 \times 10^0$,
- $fl_A(0.99999) = 0.1000 \times 10^1$.

▽

Exemple 1.1.8 Anem a veure com es representa dins la memòria $fl_A(0.1)$ en l'aritmètica IEEE-simple. A l'exemple 1.1.3 hem vist

$$(0.1)_{10} = (0.\widehat{1100})_2 \times 2^{-3}. \quad (1.4)$$

Llavors, d'acord amb la figura 1.1, $s = \boxed{0}$ (és un número positiu), $e = q + 126 = 123$ però en binari, i la mantissa surt directament de (1.4). Per passar un número natural a binari, dividim per dos, llavors la resta de la divisió és el primer dígit binari per la dreta, ens quedem amb el quocient, dividim per dos, la resta de la divisió és el segon dígit binari per la dreta, ens quedem amb el quocient, i així successivament fins que obtinguem quocient zero. En el nostre cas,

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

d'on $123 = (1111011)_2$. Per tant, $fl_A(0.1)$ en format IEEE simple queda

$$\boxed{0} \parallel \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \parallel \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{1}$$

Recordeu que el primer dígit de la mantissa sempre és 1 i no es representa. Noteu també que l'últim bit prové de l'arrodoniment.

Noteu que, degut al fet que hem truncat, $fl_T(0.1) \neq 0.1$, o sigui, **0.1 no es representa exactament en IEEE-simple** (ni en cap altra aritmètica de punt flotant binària). ▽

Exemple 1.1.9 Podem comprovar efectivament que 0.1 es representa a la memòria tal com hem deduït a l'exemple anterior. Si agrupem els bits de quatre en quatre, obtenim la representació hexadecimal 3DCCCCD. Considereu el següent programa en C.

²De fet, si $\alpha_i = b - 1$ per $i = 1, \dots, t$ i $q = q_{\max}$, llavors $s(0.\alpha_1 \dots \alpha_{t-1}(\alpha_t + 1))b^q$ no és número màquina perquè produeix un overflow. Per simplicitat, ignorarem aquest cas.

```
#include <stdio.h>

int main (void) {
    float x=0.1;
    fwrite(&x,sizeof(float),1,stdout);
    return 0;
}
```

Si guardeu l'executable corresponent com `bolca_float`, la comanda

```
./bolca_float | xxd
```

us permetrà obtenir un bolcat hexadecimal de la memòria ocupada per la variable `x`. ∇

1.1.3 Notacions per quantificar l'error

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, formalitzem el concepte d'error.

Definició 1.1.10 Si $x \in \mathbb{R}$ i \tilde{x} és una aproximació de x , definim l'**error absolut de \tilde{x} com a aproximació de x** , $e_a(\tilde{x}, x)$, per

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

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

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

Notem que l'error relatiu no està definit si $x = 0$. Quan x sigui desconegut, prendrem $e_r(\tilde{x}, x) \simeq \frac{\tilde{x} - x}{\tilde{x}}$. Quan sigui clar a partir del context, ometrem el segon argument de e_a i e_r i escriurem $e_a(\tilde{x})$ o $e_r(\tilde{x})$.

En general no coneixem aquests errors exactament sinó només una fita d'aquests:

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

Observació 1.1.11 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 ”, és a dir, $e_r(A, B) = \delta$. Això serà d'utilitat per fitar la propagació de l'error a les operacions més endavant. ∇

Per exemple, $x = 2.100 \pm 5 \times 10^{-4}$ vol dir que aproximem x per $\tilde{x} = 2.100$ però que x està entre $2.100 - 5 \times 10^{-4} = 2.0995$ i $2.100 + 5 \times 10^{-4} = 2.1005$. Amb les notacions anteriors, $\varepsilon_a(\tilde{x}) = 5 \times 10^{-4}$ i

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

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

El nombre de xifres significatives i l'error relatiu estan relacionats d'acord amb el següent

Lema 1.1.13 *En les notacions de la definició anterior, es compleix*

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

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

Prova: Exercici. □

Quan es mostra un resultat numèric, és desitjable que tots els decimals que s'escriuen siguin correctes, i que totes les xifres que s'escriuen sense comptar zeros al davant siguin significatives.

Exemple 1.1.14 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 correcta ni xifra significativa. En la segona, tots els decimals són correctes i totes les xifres són significatives. ▽

1.1.4 Fites de l'error de representació

Tornant als errors de representació, tenim la següent

Proposició 1.1.15 *Sigui $x \in \mathbb{R}$, $x \neq 0$ amb representació en punt flotant infinita i normalitzada donada pel teorema 1.1.1,*

$$x = s(0.\alpha_1\alpha_2\dots)_b b^q, \quad \alpha_1 \neq 0.$$

Suposem b parell. L'error absolut de la seva representació en punt flotant en base b i t dígitos satisfà les següents fites:

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

Les fites per l'error relatiu són:

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

Prova: Sigui $x = s(0.\alpha_1\alpha_2\dots)b^q$. La fita per $e_a(fl_T(x))$ se segueix directament de la definició de truncament. Per la fita de $e_r(fl_T(x))$ fem

$$\left| \frac{fl_T(x) - x}{x} \right| \leq \frac{b^{q-t}}{(0.\alpha_1\alpha_2\dots\alpha_t\dots)_b b^q} = \frac{b^{-t}}{(0.\alpha_1\alpha_2\dots\alpha_t\dots)_b} \leq b^{1-t}, \quad (1.5)$$

on la darrera desigualtat és deguda al fet que $b^{-1} \leq (0.\alpha_1\alpha_2\dots\alpha_t\dots)_b$ (la representació és normalitzada).

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

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

$$\begin{aligned} |fl_A(x) - x| &= |x| - |fl_A(x)| = (0.\underbrace{0\dots 0}_t \alpha_{t+1}\alpha_{t+2}\dots)_b b^q \\ &= (0.\alpha_{t+1}\alpha_{t+2}\dots)_b b^{q-t} \leq \frac{1}{2}b^{q-t}. \end{aligned}$$

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

$$\begin{aligned} |fl_A(x) - x| &= ||fl_A(x)| - |x|| \\ &= |(0.\alpha_1\dots(\alpha_t+1))_b b^q - (0.\alpha_1\dots\alpha_t\alpha_{t+1}\dots)_b b^q| \\ &= b^q \left((0.\alpha_1\dots\alpha_t)_b + b^{-t} - (0.\alpha_1\dots\alpha_t)_b - (0.0\overset{(t)}{.}0\alpha_{t+1}\alpha_{t+2}\dots)_b \right) \\ &= b^q \left(bb^{-(t+1)} - (0.0\overset{(t)}{.}0\alpha_{t+1}\alpha_{t+2}\dots)_b \right) \\ &\leq b^q \underbrace{(b - \alpha_{t+1})}_{\leq b/2} b^{-(t+1)} \leq \frac{1}{2}b^{q-t}. \end{aligned}$$

La fita de l'error relatiu per arrodoniment es dedueix amb el mateix argument que a (1.5). \square

A partir d'ara considerarem **només representació en punt flotant per arrodoniment**. És a dir, serà sempre $fl = fl_A$.

La fita de l'error relatiu de representació motiva la següent definició.

Definició 1.1.16 *A una aritmètica de punt flotant $F(b, t, q_{min}, q_{max})$, la quantitat $\frac{1}{2}b^{1-t}$ s'anomena precisió de la màquina.*

1.2 Errors a les operacions

1.2.1 Aritmètica de punt flotant

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

En la pràctica totalitat d'ordinadors i calculadores, l'únic error que comenten les operacions en punt flotant és el de la representació del resultat. És a dir, si x i y són nombres màquina,

$$\begin{aligned} x \oplus y &= fl(x + y) & x \otimes y &= fl(x \cdot y) \\ x \ominus y &= fl(x - y) & x \oslash y &= fl(x/y) \end{aligned}$$

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.

La notació $x \oplus y$, $x \ominus y$, $x \otimes y$, $x \oslash y$, no és standard. És més habitual denotar $fl(x + y)$, $fl(x - y)$, $fl(x \cdot y)$, $fl(x/y)$. De fet, aquesta darrera notació s'estén al cas en què x, y no són nombres màquina:

$$\begin{aligned} fl(x + y) &:= fl(fl(x) + fl(y)), & fl(x - y) &:= fl(fl(x) - fl(y)), \\ fl(xy) &:= fl(fl(x) \cdot fl(y)), & fl(x/y) &:= fl(fl(x)/fl(y)). \end{aligned}$$

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

El fet que considerem que l'error de les operacions és el de la representació del resultat fa que puguem obtenir la precisió de la màquina a partir de l'aritmètica de punt flotant.

Definició 1.2.1 *Es coneix com a èpsilon de la màquina el nombre positiu ϵ més petit que sumat a 1 dona diferent de 1:*

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

L'èpsilon màquina és precisament la precisió de la màquina.

Proposició 1.2.2 *Per a una màquina que trunca, l'èpsilon màquina és b^{1-t} , mentre que per una màquina que arrodoneix és $\frac{1}{2}b^{1-t}$.*

Prova: Es proposa com a exercici. □

Nota 1.2.3 La definició 1.2.1 és la definició clàssica d'èpsilon màquina de l'anàlisi numèrica. No obstant, n'existeix una altra definició que està molt estesa, segons la qual l'èpsilon màquina és el nombre que s'ha de sumar a 1 per a obtenir el següent nombre dins la representació en punt flotant. D'acord amb les nostres definicions, aquest increment és b^{1-t} , donat que $1-t$ és la potència de b que correspon a l'últim dígit de la mantissa de 1. Tant C com Octave segueixen aquesta altra definició. A Octave ho podeu comprovar mitjançant la comanda

`eps`

que retorna $b^{1-t} = 2^{-52}$, donat que $t = 53$ al format IEEE-doble. A C, els èpsilon màquina dels formats de punt flotant `float` i `double` són les constants `FLT_EPSILON` i `DBL_EPSILON` definides a `float.h`.

D'acord amb la proposició 1.2.2, l'èpsilon màquina de la definició 1.2.1 per la precisió doble, és a dir, el nombre més petit que sumat a 1 dona diferent de 1, és $\frac{1}{2}b^{1-t} = 2^{-53}$. Si ho intenteu comprovar amb Octave, sembla que no sigui veritat:

```
1==1+2*(-53)
```

El motiu és que l'arrodoniment de l'aritmètica IEEE-doble fa servir una *regla de desempat cap als parells*: si s'ha d'arrodonir a t dígit un nombre del qual el bit $t + 1$ és 1 però els bits del $t + 2$ cap a endavant són tots zero, s'arrodoneix cap al nombre que té el bit t parell, és a dir, que té el bit t zero. Això vol dir que si el bit t és 0 (sota el supòsit de que tots els bits del $t + 2$ en endavant son zero), s'arrodoneix cap a avall, mentre que si el bit t és 1 s'arrodoneix cap a amunt. Ho podeu comprovar modificant l'últim bit del nostre candidat a èpsilon màquina:

```
1==1+(2*(-53)+2*(-105))
```

Comproveu que el parèntesi que envolta la segona suma és essencial: la suma en punt flotant no és associativa. ∇

1.2.2 Propagació dels errors a les operacions

A continuació estudiem l'efecte de la propagació d'errors a les operacions mitjançant exemples bàsics.

Exemple 1.2.4 Com a primer exemple d'estudi de la propagació d'errors en les operacions, anem a afitar l'error comès en el càlcul de xy . Si suposem que ni x ni y són necessàriament nombres màquina, el que calcularem realment és $\text{fl}(x) \otimes \text{fl}(y)$. 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

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

En l'anterior cadena d'igualtats, cada cop que hem escrit un \approx ha estat perquè 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 . D'acord amb l'anterior,

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

d'on

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

o, escrit de forma equivalent,

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

Per tant, també tenim

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

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

Exemple 1.2.5 Anem a estudiar l'error degut a les operacions en el càlcul de $(a_1 \oplus a_2) \oplus a_3$, suposant que 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 :

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

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

$$e_r((a_1 \oplus a_2) \oplus a_3) = \delta_1 \frac{a_1 + a_2}{a_1 + a_2 + a_3} + \delta_2. \quad (1.6)$$

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.

Vegem-ne un exemple. 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 $a + b + c = 0.641371258 \times 10^{-3}$.

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

	a_1	a_2	a_3	$\left \frac{a_1 + a_2}{a_1 + a_2 + a_3} \right $
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 dígit 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 dígit (feu la resta a mà i veureu que obteniu 5 zeros al començament).

Segurament us heu 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 (1.6). 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. ∇

La conclusió que s'ha de treure de l'exemple anterior és que les cancel·lacions no són peril·loeses 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.

Exemple 1.2.6 A l'exemple 1.1.8 hem vist que 0.1 no es representa exactament a cap aritmètica binària. El següent codi Octave permet comprovar l'acumulació d'error en operar amb 0.1:

```
format long g
a=0 ; for i=1:100 a+=0.1; end; a
```

Observeu que l'error acumulat és superior a l'èpsilon-màquina de l'aritmètica IEEE-doble.

El motiu pel qual les calculadores treballen en base 10 és evitar l'acumulació d'error amb quantitats amb les que treballem exactament quan calculem en base 10 amb poques xifres, com és el cas d'aquest exemple. El codi equivalent per una calculadora HP-48,49,50,

```
0 1 100 START .1 + NEXT
```

no acumula cap error. ∇

1.3 Propagació dels errors a les dades

Comencem per estudiar 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.

A continuació anem a provar 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 1.3.1 (Fórmula de Taylor en diverses variables) *Sigui $f : U \subset \mathbb{R}^n \rightarrow \mathbb{R}$, $a \in U$, U obert estrellat respecte de a , $f \in C^{k+1}(U)$. Aleshores, per tot h t.q. $a + h \in U$,*

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

on $\xi \in (0, 1)$ depèn de h .

Proposició 1.3.2 (fórmula de propagació de l'error maximal) *Sigui $f : U \subset \mathbb{R}^n \rightarrow \mathbb{R}$, $x = (x_1, \dots, x_n) \in U$, U obert estrellat respecte de x , \tilde{x}_i aproximació de x_i per $i = 1, \dots, n$, $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n) \in U$. Aleshores, menyspreant productes d'almenys dos errors,*

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

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

Per a les corresponents fites, tenim

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

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

Prova: Si considerem la fórmula de Taylor (proposició 1.3.1) amb resta d'ordre 2,

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

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

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

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. \square

Proposició 1.3.3 *Suposem que \tilde{x} és una aproximació de x i que \tilde{y} és una aproximació de y .*

$$(a) \quad e_a(\tilde{x} \pm \tilde{y}, x \pm y) = e_a(\tilde{x}, x) \pm e_a(\tilde{y}, y).$$

$$(b) \quad \text{Per a } x, y \neq 0, \quad e_r(\tilde{x}\tilde{y}, xy) = e_r(\tilde{x}, x) + e_r(\tilde{y}, y).$$

$$(c) \quad \text{Per a } x, y \neq 0, \quad e_r(\tilde{x}/\tilde{y}, x/y) = e_r(\tilde{x}, x) - e_r(\tilde{y}, y).$$

Per a les corresponents fites,

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

$$(e) \quad \text{Per a } x, y \neq 0, \quad \varepsilon_r(\tilde{x}\tilde{y}, xy) = \varepsilon_r(\tilde{x}, x) + \varepsilon_r(\tilde{y}, y).$$

$$(f) \quad \text{Per a } x, y \neq 0, \quad \varepsilon_r(\tilde{x}/\tilde{y}, x/y) = \varepsilon_r(\tilde{x}, x) + \varepsilon_r(\tilde{y}, y).$$

També tenim

$$(g) \quad e_a(\tilde{x}\tilde{y}, xy) = y e_a(\tilde{x}, x) + x e_a(\tilde{y}, y).$$

$$(h) \quad \varepsilon_a(\tilde{x}\tilde{y}, xy) = |y| \varepsilon_a(\tilde{x}, x) + |x| \varepsilon_a(\tilde{y}, y).$$

$$(i) \quad \text{Per a } y \neq 0, \quad e_a(\tilde{x}/\tilde{y}, x/y) = (1/y) e_a(\tilde{x}, x) - (x/y^2) e_a(\tilde{y}, y).$$

(j) Per a $y \neq 0$, $\varepsilon_a(\tilde{x}/\tilde{y}, x/y) = (1/|y|)\varepsilon_a(\tilde{x}, x) + (|x|/y^2)\varepsilon_a(\tilde{y}, y)$.

Prova: Fem, per exemple, (c) i (f). Prenent $f(x, y) = x/y$ i aplicant la proposició 1.3.2,

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

Prenent valors absoluts, tenim

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

d'on

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

La resta d'apartats es demostren de manera similar. \square

Nota 1.3.4 Per a $x, y \neq 0$, els apartats (g), (h), (i), (j) es poden obtenir a partir dels apartats (b), (c), (e), (f) passant a error relatiu. Per exemple,

$$\begin{aligned} \varepsilon_a(\tilde{x}\tilde{y}, xy) &= |xy|\varepsilon_r(\tilde{x}\tilde{y}, xy) = |xy|(\varepsilon_r(\tilde{x}, x) + \varepsilon_r(\tilde{y}, y)) \\ &= |xy|(\varepsilon_a(\tilde{x}, x)/|x| + \varepsilon_a(\tilde{y}, y)/|y|). \end{aligned}$$

d'on s'obté (h). No obstant, aquesta prova no és vàl·lida per a $x = 0$ o $y = 0$. Tal com es diu a la demostració de la proposició 1.3.3, es pot fer una prova directa d'aquests apartats a partir de la proposició 1.3.2. ∇

Nota 1.3.5 Alguns dels apartats de la proposició 1.3.3 admeten una prova directa senzilla sense necessitat de recórrer a la proposició 1.3.2. Per exemple, l'apartat (g) es pot provar com segueix:

$$\begin{aligned} \tilde{x}\tilde{y} &= (x + e_a(\tilde{x}, x))(y + e_a(\tilde{y}, y)) \\ &= xy + xe_a(\tilde{y}, y) + ye_a(\tilde{x}, x) + e_a(\tilde{x}, x)e_a(\tilde{y}, y). \end{aligned}$$

Menyspreant el producte de dos errors, obtenim

$$e_a(\tilde{x}\tilde{y}, xy) = \tilde{x}\tilde{y} - xy = ye_a(\tilde{x}, x) + xe_a(\tilde{y}, y),$$

tal com volíem veure. ∇

Exemple 1.3.6 Suposant que les operacions es fan exactament, anem a determinar l'error efectuat en el càlcul de $x_1x_2^2$, 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

$$\begin{aligned}\varepsilon_a(\tilde{x}_1, x_1) = 0.1 &\implies \varepsilon_r(\tilde{x}_1, x_1) = \frac{\varepsilon_a(\tilde{x}_1, x_1)}{|\tilde{x}_1|} = 0.05 \\ \varepsilon_a(\tilde{x}_2, x_2) = 0.2 &\implies \varepsilon_r(\tilde{x}_2, x_2) = \frac{\varepsilon_a(\tilde{x}_2, x_2)}{|\tilde{x}_2|} = 0.0667\end{aligned}$$

Fent servir la proposició 1.3.3, apartat (e), obtenim

$$\begin{aligned}\varepsilon_r(\tilde{x}_1\tilde{x}_2^2, x_1x_2^2) &= \varepsilon_r(\tilde{x}_1, x_1) + \varepsilon_r(\tilde{x}_2^2, x_2^2) \\ &= \varepsilon_r(\tilde{x}_1, x_1) + \varepsilon_r(\tilde{x}_2, x_2) + \varepsilon_r(\tilde{x}_2, x_2) \\ &= \varepsilon_r(\tilde{x}_1, x_1) + 2\varepsilon_r(\tilde{x}_2, x_2) \\ &= 0.05 + 2 \times 0.0667 = 0.183.\end{aligned}$$

Passem a error absolut,

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

i, per tant, podem escriure,

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

▽

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 de la proposició 1.3.3, o bé directament de la proposició 1.3.2 amb $f(x, y) = x + y$, s'obté

$$\varepsilon_r(\tilde{x} + \tilde{y}, x + y) = \left| \frac{x}{x + y} \right| \varepsilon_r(\tilde{x}, x) + \left| \frac{y}{x + y} \right| \varepsilon_r(\tilde{y}, y). \quad (1.7)$$

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 $\varepsilon_r(\tilde{x}, x)$ sigui gros, si tant el factor $|x|/|x + y|$ com $\varepsilon_r(\tilde{y}, y)$ són prou petits, $\varepsilon_r(\tilde{x} + \tilde{y}, x + y)$ 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 multiplicat es veurà amplificat. Aquesta amplificació és dràstica si $x \approx -y$, és a dir, quan hi ha **cancel·lacions**, tal com hem comprovat a l'exemple 1.2.5.

Exemple 1.3.7 Anem a veure un exemple de com una cancel·lació pot eliminar tota la precisió que teníem i un esmorteïment la pot recuperar. Definim

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

Suposem que fem les següents dues sumes de manera exacta:

$$\tilde{d} = \tilde{a} + \tilde{b}, \quad \tilde{e} = \tilde{d} + \tilde{c}.$$

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

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

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

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

d'on

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

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

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

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

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

A continuació veurem un resultat per fitar la propagació de l'error a través de l'aplicació d'una funció qualsevol, suposant que aquesta funció s'avalua exactament.

La fórmula de propagació dels errors maximals (proposició 1.3.2) 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 $\varphi : \mathbb{R} \rightarrow \mathbb{R}$, en la forma següent:

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

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.

Exemple 1.3.8 Suposem que volem calcular $a = (3 - 2\sqrt{3})^4$, prenent $\sqrt{3} \approx 1.73205$. Definim $x := \sqrt{3}$, $\tilde{x} = 1.73205$, de manera que

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

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

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

El coeficient de propagació corresponent a aquest càlcul és 29.86.

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,

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

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

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

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

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 fet que, en avaluar $873 - 504x$, es produeix una cancel·lació.

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,

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

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 $\sqrt{3}$.

El següent codi Octave mostra el valor d'aquestes quatre expressions en precisió doble:

```
a=(3-2*sqrt(3))*4 ; b=(21-12*sqrt(3))*2 ; ...
c=873-504*sqrt(3) ; d=9/(97+56*sqrt(3))
format long e
[ a; b; c; d ]
```

D'acord amb la nostra anàlisi, el valor més precís és l'últim. Observeu la diferencia entre els dos darrers valors. ∇

1.4 Propagació d'errors en dades i operacions

Combinant l'estudi de la propagació dels errors a les operacions i de la propagació d'errors a les dades, es poden tenir en compte els dos efectes simultàniament. Vegem-ne un exemple.

Exemple 1.4.1 Tenim una quantitat \tilde{x} amb un error relatiu de mesura fitat per E . Si x és la corresponent quantitat exacta, suposem $x, \tilde{x} > 0$. Volem avaluar l'expressió

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

a un ordinador fictici que representa amb error relatiu fitat per ε , fa operacions aritmètiques amb error relatiu fitat per 2ε , fa arrels quadrades amb error relatiu fitat per 3ε i fa logaritmes neperians amb error relatiu fitat per 5ε . Volem trobar una fita de l'error relatiu de

$$fl\left(\sqrt{3 + (\ln \tilde{x})^2}\right) \quad \text{com a aproximació de} \quad \sqrt{3 + (\ln x)^2}$$

tenint en compte errors a les operacions.

Si “expandim fl ”, tal com hem fet fins ara, obtenim

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

on

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

Per a acabar, hauríem de trobar δ tal que

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

(menyspreant errors de segon ordre), hauríem de fitar $|\delta|$ en termes de les fites de les $|\delta_i|$, i aquesta fita seria la resposta. No és que sigui impossible, però seria extremadament tediós.

Una estratègia més senzilla i sistemàtica és acumular aproximacions dels errors a primer ordre usant la fórmula de propagació de l'error maximal (proposició 1.3.2). Trenquem el càlcul de $f(x)$ en operacions elementals i considerem els valors exactes i aproximats a cada etapa:

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

Obtindrem progressivament $\varepsilon_r(\tilde{x}_0, x_0)$, $\varepsilon_r(\tilde{x}_1, x_1)$, $\varepsilon_r(\tilde{x}_2, x_2)$, $\varepsilon_r(\tilde{x}_3, x_3)$, $\varepsilon_r(\tilde{x}_4, x_4)$, $\varepsilon_r(\tilde{x}_5, x_5)$. A cada pas tindrem en compte l'error introduït per l'operació i la propagació de l'error acumulat.

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

$$\varepsilon_r(\tilde{x}_0, x_0) = \varepsilon_r(x(1 + \delta_0), x) = E.$$

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

$$\begin{aligned} \varepsilon_r(\tilde{x}_1, x_1) &= \varepsilon_r(\tilde{x}_0(1 + \delta_1), x_0) = \varepsilon_r(\tilde{x}_0(1 + \delta_1), x_0 \cdot 1) \\ &\stackrel{\text{prop. 1.3.3}}{=} \varepsilon_r(\tilde{x}_0, x_0) + \varepsilon_r(1 + \delta_1, 1) = E + \varepsilon. \end{aligned}$$

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

$$\begin{aligned} \varepsilon_r(\tilde{x}_2, x_2) &= \varepsilon_r(f(\ln \tilde{x}_1), \ln x_1) \\ &= \varepsilon_r(\ln \tilde{x}_1(1 + \delta_2), \ln x_1) \\ &= \varepsilon_r(\ln \tilde{x}_1, \ln x_1) + \varepsilon_r(1 + \delta_2, 1) \\ &\stackrel{\text{prop. 1.3.2}}{=} \frac{|x_1|^{\frac{1}{|\ln x_1|}}}{|\ln x_1|} \varepsilon_r(\tilde{x}_1, x_1) + 5\varepsilon = \frac{E + \varepsilon}{|\ln x|} + 5\varepsilon. \end{aligned} \tag{1.8}$$

Noteu que, a (1.8), el primer terme correspon a la propagació de l'error acumulat ($\varepsilon_r(\tilde{x}_1, x_1)$), mentre que el segon terme és el nou error introduït degut a l'avaluació del logaritme en punt flotant.

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

$$\begin{aligned}
 \varepsilon_r(\tilde{x}_3, x_3) &= \varepsilon_r(f(\tilde{x}_2^2), x_2^2) = \varepsilon_r(\tilde{x}_2^2(1 + \delta_3), x_2^2) \\
 &= \varepsilon_r(\tilde{x}_2^2, x_2^2) + \varepsilon_r(1 + \delta_3, 1) \\
 &= \frac{|x_2|2|x_2|}{|x_2|^2} \varepsilon_r(\tilde{x}_2, x_2) + 2\varepsilon = 2\left(\frac{E + \varepsilon}{|\ln x|} + 5\varepsilon\right) + 2\varepsilon \\
 &= \frac{2}{|\ln x|}(E + \varepsilon) + 12\varepsilon.
 \end{aligned} \tag{1.9}$$

Novament, a (1.9) el primer terme correspon a la propagació de l'error acumulat ($\varepsilon_r(\tilde{x}_2, x_2)$), i el segon és l'error introduït pel producte en punt flotant.

Si continuem el procés, acabem amb

$$\frac{(E + \varepsilon)|\ln x| + 6\varepsilon|\ln x|^2}{3 + |\ln x|^2} + 4\varepsilon,$$

d'on, si separem els termes que multipliquen E dels que multipliquen ε , obtenim

$$E \frac{|\ln x|}{3 + |\ln x|^2} + \varepsilon \left(4 + \frac{|\ln x| + 6|\ln x|^2}{3 + |\ln x|^2}\right). \tag{1.10}$$

Sense voler fer una anàlisi detallada, observeu que, per a valors “moderadament grans” i “moderadament petits” de $|x|$, les expressions

$$\frac{|\ln x|}{3 + |\ln x|^2}, \quad 4 + \frac{|\ln x| + 6|\ln x|^2}{3 + |\ln x|^2}$$

no són gaire grans. D'aquí se segueix que, només que ε sigui uns quants ordres de magnitud menor que E , l'error degut a les operacions (segon terme) es fa menyspreable al costat de la propagació de l'error a les dades.³ Aquesta és una situació del tot habitual, i per això, en general, “ens podem creure” els càlculs fets en calculadores i ordinadors.

Observeu també que, si haguéssim fet el càlcul de la propagació de l'error de mesura suposant les operacions exactes, hauríem definit

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

i hauríem calculat

$$\varepsilon_r(\varphi(\tilde{x}), \varphi(x)) = \frac{|\ln x|}{3 + |\ln x|^2} \varepsilon_r(\tilde{x}, x),$$

que és el primer terme de (1.10). ▽

Nota 1.4.2 Al final de l'exemple anterior hem vist que, si volem estudiar la propagació de l'error relatiu en \tilde{x} , que suposem fitat per E , a través de l'expressió $\sqrt{3 + (\ln x)^2}$, obtenim que aquest error està fitat per

$$\frac{|\ln x|}{3 + |\ln x|^2} E.$$

³Observeu que diem “l'error” però sempre estem parlant de fites!

Noteu que aquesta expressió val zero si $x = 1$. Això ens podria portar a pensar que quan $x = 1$ no hi ha propagació de l'error, cosa que és impossible, perquè la funció $\sqrt{3 + (\ln x)^2}$ no és constant en un entorn de $x = 1$.

Una reflexió sobre aquest fet us portarà a descobrir què està passant: la funció $f(x) = \sqrt{3 + (\ln x)^2}$ satisfà $f'(1) = 0$. Això vol dir que l'efecte dels errors és de segon ordre i la nostra anàlisi, en menysprear els errors de segon ordre, no els detecta. Per fitar $\varepsilon_r(\varphi(\tilde{x}), \varphi(x))$ en el cas que $x = 1$ només cal expandir per Taylor fins a ordre 2 i menysprear la resta de Lagrange, d'ordre 3. ∇

1.5 Algorismes estables i inestables. Problemes mal condicionats.

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. Vegem-ne un exemple.

Exemple 1.5.1 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 :

$$\begin{aligned} E_0 &:= 1 - e^{-1} \\ \forall i = 1, 2, \dots, n \\ E_i &:= 1 - iE_{i-1} \end{aligned}$$

Suposem que volem trobar E_{20} . Amb el següent codi Octave, generem els valors $\{E_i\}_{i=0}^{20}$ sobre el vector fila `ee[]` mitjançant la recurrència anterior.

```
n=20;
ee(1)=1-exp(-1);
for i=1:n ee(i+1)=1-i*ee(i); end
transpose(ee)
```

Obtenim E_{20} negatiu (de fet, E_{18} ja ho és), mentre que hauria de ser positiu, donat que l'integrand de la definició de E_n és positiu. Per tant, E_{20} no té cap xifra significativa (i E_{18}) tampoc. Si avaluem numèricament les integrals⁴ mitjançant la funció `quad()` i comparem amb els valors obtinguts mitjançant la recurrència, podem observar la propagació de l'error.

⁴Al darrer tema veurem mètodes d'integració numèrica

```
format long e
for i=0:n eee(i+1)=quad(@x x**i*exp(x-1),0,1); end
[ transpose(ee) transpose(eee) transpose(ee)-transpose(eee) ]
```

Fem una anàlisi de l'error simplificada suposant que, en l'avaluació de la recurrència anterior, les operacions són exactes i només es propaga l'error de representació de E_0 , que està fitat per l'èpsilon-màquina de la precisió doble, que és $\frac{1}{2}b^{1-t} = 2^{-53} = 1.1102 \cdot 10^{-16}$. Per tant, al el nostre model de propagació d'errors, comencem amb $\tilde{E}_0 = E_0 + e_0$ i obtenim recurrentment $\tilde{E}_i = 1 - i\tilde{E}_{i-1}$. Si anomenem e_i l'error en el càlcul d' \tilde{E}_i , es verifica

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

Això vol dir que a cada pas l'error es multiplica per i . Així,

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

d'on $e_{20} = 20!e_0 \approx 2.4329 \cdot 10^{18} \times 1.1102 \cdot 10^{-16} = 270.11$. Com que de l'expressió de E_n se segueix que $|E_n| < 1$, aquest model d'error no garanteix cap xifra significativa per E_{20} .

Tot i això, aquesta recurrència es pot aprofitar. Observem que, si iterem cap a enrere, l'error disminueix en comptes d'augmentar. En efecte: si tenim

$$\tilde{E}_n = E_n + e_n,$$

llavors, com que

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

i per les integrals aproximades usarem la mateixa recurrència,

$$\tilde{E}_{i-1} = \frac{1 - \tilde{E}_i}{i}, \quad (1.11)$$

tindrem

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

d'on

$$e_n = -\frac{e_{n+1}}{n+1} = \frac{e_{n+2}}{(n+1)(n+2)} = \dots = \frac{(-1)^k n!}{(n+k)!} e_{n+k}. \quad (1.12)$$

Si, suposant que volem trobar E_n , podem escollir \tilde{E}_{n+k} de manera que, en iterar enrere k vegades, el factor $\frac{n!}{(n+k)!}$ hagi eliminat e_{n+k} , ja ho tindrem.

Per a fer això, observem

$$|E_n| = \left| \int_0^1 x^n e^{x-1} dx \right| \leq \int_0^1 |x^n e^{x-1}| dx \leq \int_0^1 |x^n| dx = \int_0^1 x^n dx = \frac{1}{n+1},$$

d'on, si prenem $\tilde{E}_{n+k} = 0$, tindrem $|e_{n+k}| = |\tilde{E}_{n+k} - E_{n+k}| = |E_{n+k}| \leq \frac{1}{n+k+1}$, i, per tant, de (1.12),

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

Emprem això per trobar E_{20} . Mitjançant la funció


```
erre=@(n,k) factorial(n)/factorial(n+k+1);
```

i provant valors de k , obtenim que la primera k que fa que $n!/(n+k+1)!$ sigui menor que l'èpsilon-màquina de la precisió doble és $k = 11$. Iterem cap a enrere des de $\tilde{E}_{31} := 0$ fins a \tilde{E}_{20} mitjançant

```
n=20; k=11; ee(n+k+1)=0; for i=n+k:-1:n+1 ee(i)=(1-ee(i+1))/i; end
```

El valor \tilde{E}_{20} el trobem dins $ee(21)$. Comparant amb el valor de la integral avaluada numèricament que hem calculat abans,

```
format long e
eee(n+1)
ee(n+1)
```

observem que totes les xifres són correctes. ▽

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**.

Exemple 1.5.2 Un sistema lineal mal condicionat: el sistema

$$\left. \begin{array}{rcl} 2.0000x & + & 0.6667y = 2.6667 \\ 1.0000x & + & 0.3333y = 1.3333 \end{array} \right\}$$

té per solució $x = 1, y = 1$. Modifiquem lleugerament un dels coeficients:

$$\left. \begin{array}{rcl} 2.0000x & + & 0.666\boxed{5}y = 2.6667 \\ 1.0000x & + & 0.3333y = 1.3333 \end{array} \right\}$$

Aquest sistema té solució $x = 1.6666, y = -1.0000$. I el sistema

$$\left. \begin{array}{rcl} 2.0000x & + & 0.666\boxed{6}y = 2.666\boxed{6} \\ 1.0000x & + & 0.3333y = 1.3333 \end{array} \right\}$$

té infinites solucions, $x = 1.3333 - 0.3333z, y = z$. ▽

Exemple 1.5.3 (Wilkinson, 1963) Considerem el polinomi

$$\begin{aligned} p(x) &:= (x-1)(x-2)\dots(x-19)(x-20) \\ &= x^{20} + a_1x^{19} + a_2x^{18} + a_3x^{17} + \dots + a_{20}, \end{aligned}$$

on

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

tal com podeu comprovar a Octave mitjançant `poly(1:20)`. Per construcció, aquest polinomi té per arrels $1, 2, \dots, 20$. Pertorbem-lo lleugerament i considerem

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

A Octave, podeu trobar numèricament les arrels del polinomi pertorbat mitjançant

```
roots( poly(1:20)+2**-23*eye(21)(2,:) )
```

Veiem que un petit canvi ($2^{-23} \simeq 1.192 \times 10^{-7}$) en el coeficient a_1 ha causat que diversos zeros passin a ser complexos i alguns d'ells estiguin allunyats diverses 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. Escrivim

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

L'equació

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

defineix x implícitament com a funció de 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

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

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	$\partial x / \partial a_1 _{x=i}$	i	$\partial x / \partial a_1 _{x=i}$	i	$\partial x / \partial a_1 _{x=i}$	i	$\partial x / \partial a_1 _{x=i}$
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 .

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., $e_r(f(a_1)), \dots, e_r(f(a_{20}))$) introduiran errors considerables en les arrels. Aquest l'observareu si proveu de trobar numèricament les arrels del polinomi a partir dels seus coeficients, per exemple mitjançant

```
roots(poly(1:20))
```

a Octave.

▽

Capítol 2

Àlgebra lineal numèrica

L'objectiu d'aquest capítol és estudiar mètodes per la resolució de sistemes d'equacions lineals en un ordinador, com ara

$$\left. \begin{array}{rclcl} x_1 + & x_2 & & + 3x_4 = & 4 \\ 2x_1 + & x_2 - & x_3 + & x_4 = & 1 \\ 3x_1 - & x_2 - & x_3 + 2x_4 = & -3 \\ -x_1 + 2x_2 + 2x_3 - & x_4 = & 4 \end{array} \right\},$$

que, matricialment, s'escriu $Ax = b$ essent

$$A = \begin{pmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 2 & -1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 1 \\ -3 \\ 4 \end{pmatrix},$$

on A és la **matriu de coeficients**, x és el **vector d'incògnites** i b és el **terme independent**.

Els mètodes que veurem pertanyen a la família de **mètodes directes** (per resoldre sistemes lineals). Anomenem mètode directe tot aquell que, llevat dels errors d'arrodoniment, construeix la solució exacta en un nombre finit de passos.

En tots els mètodes que descriurem suposarem que el sistema lineal té solució única, és a dir, que $\det A \neq 0$. Es diu en aquest cas que A és **regular**, o **no singular**. Determinar numèricament si una matriu és regular o no i quina és la dimensió del seu nucli és un problema que no és fàcil d'automatitzar. Es pot abordar amb mètodes com la descomposició en valors singulars (*singular value decomposition*, o SVD).

2.1 Sistemes lineals senzills

2.1.1 Matriu diagonal

Suposem que tenim un sistema lineal $Ax = b$ amb A matriu diagonal $n \times n$ no singular, és a dir

$$\begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

amb $a_{1,1}, \dots, a_{n,n} \neq 0$. En aquest cas, podem fer, simplement

$$\forall i = 1, 2, \dots, n$$

$$x_i = b_i / a_{i,i}$$

i, en acabar, haurem efectuat un total de n divisions.

2.1.2 Matriu triangular superior

Un sistema lineal no singular de la forma

$$\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n-1} & u_{1,n} \\ & u_{2,2} & \cdots & u_{2,n-1} & u_{2,n} \\ & & \ddots & \vdots & \vdots \\ & & & u_{n-1,n-1} & u_{n-1,n} \\ & & & & u_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

se resol fent **substitució endarrere**.

Algorisme 2.1.1 (*substitució endarrere*)

$$\forall i = n, n-1, \dots, 1$$

$$x_i = \frac{b_i - \sum_{j=i+1}^n u_{i,j} x_j}{u_{i,i}}$$

Comptem el nombre d'operacions:

- Divisions: n (una per cada valor de i).
- Productes: fem un producte per cada terme del sumatori.

$$\sum_{i=1}^n \sum_{j=i+1}^n 1 = \sum_{i=1}^n (n-i) = \frac{(n-1)n}{2}.$$

Recordeu la fórmula de la suma d'una progressió aritmètica: primer terme més últim, multiplicat pel nombre de termes i dividit per dos.

- Sumes i restes: si tenim en compte la resta juntament amb les sumes, fem una suma o resta per cada terme del sumatori. Per tant, el nombre de sumes i restes és el mateix que el de productes.

Si sumem totes les operacions, obtenim

$$\begin{aligned} \#\{\text{divisions}\} + \#\{\text{productes}\} + \#\{\text{sumes i restes}\} \\ &= n + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} \\ &= n^2. \end{aligned}$$

2.1.3 Matriu triangular inferior

Un sistema lineal no singular de la forma

$$\begin{pmatrix} l_{1,1} & & & & \\ l_{2,1} & l_{2,2} & & & \\ \vdots & \vdots & \ddots & & \\ l_{n-1,1} & l_{n-1,2} & \cdots & l_{n-1,n-1} & \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n-1} & l_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

se resol fent *substitució endavant*.

Exercici: *Formuleu algorísmicament la substitució endavant i compteu el nombre d'operacions necessari per dur-la a terme.*

2.2 Mètode de Gauss

Per sistemes lineals amb matriu de coeficients general i no singular farem servir el mètode de Gauss, que ja coneixeu. En aquesta secció en deduirem una formulació algorísmica.

Pensant en descriure un algorisme general, desenvolupem el mètode de Gauss per resoldre un sistema lineal general 3×3 $Ax = b$, amb

$$A =: A^{(1)} =: \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & a_{2,3}^{(1)} \\ a_{3,1}^{(1)} & a_{3,2}^{(1)} & a_{3,3}^{(1)} \end{pmatrix}, \quad b =: b_1 =: \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{pmatrix}.$$

El primer pas del mètode de Gauss és

$$\begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \left| & b_1^{(1)} \right. \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & a_{2,3}^{(1)} & \left| & b_2^{(1)} \right. \\ a_{3,1}^{(1)} & a_{3,2}^{(1)} & a_{3,3}^{(1)} & \left| & b_3^{(1)} \right. \end{pmatrix} \xrightarrow{\substack{F_2 \leftarrow F_2 - (a_{2,1}^{(1)}/a_{1,1}^{(1)})F_1 \\ F_3 \leftarrow F_3 - (a_{3,1}^{(1)}/a_{1,1}^{(1)})F_1}} \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \left| & b_1^{(1)} \right. \\ 0 & a_{2,2}^{(1)} - m_{2,1}a_{1,2}^{(1)} & a_{2,3}^{(1)} - m_{2,1}a_{1,3}^{(1)} & \left| & b_2^{(1)} - m_{2,1}b_1^{(1)} \right. \\ 0 & a_{3,2}^{(1)} - m_{3,1}a_{1,2}^{(1)} & a_{3,3}^{(1)} - m_{3,1}a_{1,3}^{(1)} & \left| & b_3^{(1)} - m_{3,1}b_1^{(1)} \right. \end{pmatrix},$$

on hem definit $m_{2,1} = a_{2,1}^{(1)}/a_{1,1}^{(1)}$ i $m_{3,1} = a_{3,1}^{(1)}/a_{1,1}^{(1)}$ (són els **multiplicadors** del pas 1 de l'eliminació gaussiana). Anomenem $A^{(2)}$ aquesta última matriu i fem un pas més:

$$\begin{aligned} A^{(2)} &=: \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \left| & b_1^{(1)} \right. \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \left| & b_2^{(2)} \right. \\ 0 & a_{3,2}^{(2)} & a_{3,3}^{(2)} & \left| & b_3^{(2)} \right. \end{pmatrix} \\ &\xrightarrow{F_3 \leftarrow F_3 - (a_{3,2}^{(2)}/a_{2,2}^{(2)})F_2} \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \left| & b_1^{(1)} \right. \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \left| & b_2^{(2)} \right. \\ 0 & 0 & a_{3,3}^{(2)} - m_{3,2}a_{2,3}^{(2)} & \left| & b_3^{(2)} - m_{3,2}b_2^{(2)} \right. \end{pmatrix} \\ &=: \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \left| & b_1^{(1)} \right. \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \left| & b_2^{(2)} \right. \\ 0 & 0 & a_{3,3}^{(3)} & \left| & b_3^{(3)} \right. \end{pmatrix} =: A^{(3)}, \end{aligned}$$

on hem definit $m_{3,2} = a_{3,2}^{(2)}/a_{2,2}^{(2)}$. El sistema lineal original $Ax = b$ és equivalent a

$$\left. \begin{aligned} a_{1,1}^{(1)}x_1 + a_{1,2}^{(1)}x_2 + a_{1,3}^{(1)}x_3 &= b_1^{(1)} \\ a_{2,2}^{(2)}x_2 + a_{2,3}^{(2)}x_3 &= b_2^{(2)} \\ a_{3,3}^{(3)}x_3 &= b_3^{(3)} \end{aligned} \right\},$$

que es pot resoldre mitjançant substitució endarrere (algorisme 2.1.1).

El procediment general el podem formalitzar en el següent

Algorisme 2.2.1 (*mètode de Gauss*)

$$\begin{array}{ll} \forall k = 1, 2, \dots, n-1 & (pas) \\ \quad \forall i = k+1, k+2, \dots, n & (fila) \\ \quad \quad m_{i,k} := a_{i,k}^{(k)}/a_{k,k}^{(k)} & (multiplicador, a_{k,k}^{(k)} \text{ es diu } \mathbf{pivot}) \\ \quad \forall j = k+1, k+2, \dots, n & (columna) \\ \quad \quad a_{i,j}^{(k+1)} = a_{i,j}^{(k)} - m_{i,k}a_{k,j}^{(k)} \\ \quad b_i^{(k+1)} = b_i^{(k)} - m_{i,k}b_k^{(k)} & (terme independent) \end{array}$$

Més en general, suposem que resollem ahora p sistemes lineals amb matriu de coeficients coincidents,

$$Ax_1 = b_1, Ax_2 = b_2, \dots, Ax_p = b_p.$$

En aquest cas, el mètode de Gauss es duu a terme mitjançant la reducció simultània de tots els termes independents. La matriu inicial és

$$\left(\begin{array}{ccc|ccc} a_{1,1}^{(1)} & \dots & a_{1,n}^{(1)} & b_{1,1}^{(1)} & \dots & b_{1,p}^{(1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{n,1}^{(1)} & \dots & a_{n,n}^{(1)} & b_{n,1}^{(1)} & \dots & b_{n,p}^{(1)} \end{array} \right),$$

i la matriu final

$$\left(\begin{array}{ccc|ccc} a_{1,1}^{(1)} & \dots & a_{1,n}^{(1)} & b_{1,1}^{(1)} & \dots & b_{1,p}^{(1)} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & a_{n,n}^{(n)} & b_{n,1}^{(n)} & \dots & b_{n,p}^{(n)} \end{array} \right).$$

El pas d'una a l'altra es duu a terme mitjançant el següent

Algorisme 2.2.2 (*mètode de Gauss per resoldre p sistemes simultàniament*)

$$\begin{array}{ll} \forall k = 1, 2, \dots, n-1 & (pas) \\ \quad \forall i = k+1, k+2, \dots, n & (fila) \\ \quad \quad m_{i,k} := a_{i,k}^{(k)}/a_{k,k}^{(k)} & (multiplicador, a_{k,k}^{(k)} \text{ pivot}) \\ \quad \forall j = k+1, k+2, \dots, n & (columna de la matriu de coeficients) \\ \quad \quad a_{i,j}^{(k+1)} = a_{i,j}^{(k)} - m_{i,k}a_{k,j}^{(k)} \\ \quad \forall j = 1, 2, \dots, p & (columnes dels termes independents) \\ \quad \quad b_{i,j}^{(k+1)} = b_{i,j}^{(k)} - m_{i,k}b_{k,j}^{(k)} \end{array}$$

Comptem el nombre d'operacions d'aquesta darrera versió.

- Productes i restes: els podem comptar simultàniament, donat que, per cada valor de j als dos bucles més interns, efectuem tant un producte com una resta.

$$\begin{aligned}
& \sum_{k=1}^{n-1} \sum_{i=k+1}^n \left(\underbrace{\sum_{j=k+1}^n 1}_{\text{tros matriu}} + \underbrace{\sum_{j=1}^p 1}_{\text{tros termind}} \right) \\
&= \sum_{k=1}^{n-1} \sum_{i=k+1}^n \underbrace{(n-k+p)}_{\text{no depèn de } i} = \sum_{k=1}^{n-1} (n-k)(n-k+p) \\
&= \sum_{k=1}^{n-1} (n-k)^2 + \sum_{k=1}^{n-1} (n-k)p = \sum_{j=1}^{n-1} j^2 + p \sum_{j=1}^{n-1} j \\
&= \frac{(n-1)n(2(n-1)+1)}{6} + p \frac{n(n-1)}{2} \\
&= \frac{2n^3 - (3-3p)n^2 + (1-3p)n}{6}.
\end{aligned}$$

Recordeu que hi ha fórmules per sumar algunes potències d'enters consecutius, com ara

- (a) $1 + 2 + \dots + n = n(n+1)/2$,
- (b) $1^2 + 2^2 + \dots + n^2 = n(n+1)(2n+1)/6$,
- (c) $1^3 + 2^3 + \dots + n^3 = (n(n+1)/2)^2 = (1 + 2 + \dots + n)^2$.

- Divisions:

$$\sum_{k=1}^{n-1} \sum_{i=k+1}^n 1 = \sum_{k=1}^{n-1} (n-k) = \frac{(n-1+1)(n-1)}{2} = \frac{n^2 - n}{2}.$$

El nombre total d'operacions és

$$\begin{aligned}
\#\{\text{ops}\} &= 2\#\{\text{restes, productes}\} + \#\{\text{divisions}\} \\
&= \frac{4n^3 - (6-6p)n^2 + (2-6p)n}{6} + \frac{3n^2 - 3n}{6} \\
&= \frac{4n^3 - (3-6p)n^2 + (-1-6p)n}{6}.
\end{aligned}$$

Si $p \ll n$, llavors

$$\#\{\text{ops}\} = \frac{2n^3}{3} + O(n^2).$$

Nota: Molts llibres consideren fer una suma i un producte com una operació. Amb aquest conveni, el nombre d'operacions del mètode de Gauss és $n^3/3 + O(n^2)$.

2.3 Estratègies de pivotatge

Tal com hem descrit l'eliminació gaussiana, si un dels $a_{k,k}^{(k)}$ esdevé zero enmig del procediment no podem continuar. Com que $\det A \neq 0$, sabem que existirà $i > k$ tal que $a_{i,k}^{(k)} \neq 0$. Si

intercanviem les files i i k , que equival a haver començat amb una matriu de coeficients amb aquestes files intercanviades, podrem continuar. D'aquest procés se'n diu **pivotar files**.

Amb aquest procediment (o sigui, intercanviant files només si el pivot surt zero), podem resoldre qualsevol sistema lineal no singular suposant que treballem amb precisió infinita (a mà o amb un manipulador simbòlic).¹ Amb aritmètica finita, no té sentit preguntar-se si $a_{k,k}^{(k)}$ és igual a zero, donat que pot ser que teòricament sigui zero però a la pràctica no degut als errors d'arrodoniment, o viceversa. Podem considerar que $a_{k,k}^{(k)} = 0$ si $|a_{k,k}^{(k)}|$ es troba per sota certa tolerància, però llavors hi ha el problema de triar la tolerància.

A més d'això, pivotar només quan tenim certa certesa que $a_{k,k} = 0$ pot introduir inestabilitat numèrica, com podem veure al següent

Exemple 2.3.1 Volem resoldre amb aritmètica de punt flotant amb $b = 10$, $t = 4$, el sistema

$$\left. \begin{array}{rrc} x_1 & & + x_3 = 1 \\ x_1 + 0.0001x_2 + 2x_3 & = & 2 \\ x_1 + & x_2 + & x_3 = 0 \end{array} \right\}.$$

Apliquem el mètode de Gauss.

$$\begin{aligned} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 1 & 0.0001 & 2 & 2 \\ 1 & 1 & 1 & 0 \end{array} \right) & \xrightarrow{\substack{F_2 \leftarrow F_2 - F_1 \\ F_3 \leftarrow F_3 - F_1}} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 1 & 0 & -1 \end{array} \right) \\ & \xrightarrow{F_3 \leftarrow F_3 - 10^4 F_2} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 0 & -10000 & \boxed{-10000} \end{array} \right) \end{aligned}$$

(requadrem els números que s'han hagut d'arrodonir degut a l'aritmètica de 4 dígit). Si ara resollem fent substitució enrere, obtenim

$$\begin{aligned} x_3 &= -10000/(-10000) = 1, \\ x_2 &= (1 - x_3)/0.0001 = 0, \\ x_1 &= 1 - x_3 = 0. \end{aligned}$$

Si haguéssim aplicat el mètode de Gauss sense cometre errors d'arrodoniment, a l'últim pas haguéssim obtingut

$$\left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 0 & -10000 & -10001 \end{array} \right),$$

d'on la solució exacta és

$$\begin{aligned} x_3 &= -10001/-10000 = 1.0001, \\ x_2 &= (1 - x_3)/0.0001 = -1, \\ x_1 &= 1 - x_3 = -0.0001. \end{aligned}$$

¹Us podeu preguntar: per què no treballem sempre amb precisió infinita? D'una banda no tindria sentit fer-ho en el cas en que treballem amb dades experimentals, perquè les dades experimentals tenen error. D'altra banda, treballant amb precisió IEEE-doble un programa vostre en un llenguatge compilat com C o FORTRAN pot arribar a fer un nombre d'operacions per segon igual al 20% de la freqüència del processador. Un parell d'experiments us convencerà que això és impensable amb un manipulador simbòlic.

En particular, a l' x_2 trobat amb 4 dígits **no tenim cap xifra significativa**. ∇

Canviem d'estratègia: a cada pas d'eliminació gaussiana, prenem com a pivot el màxim element en valor absolut de la columna. D'aquesta estratègia se'n diu *pivotatge maximal per columnes*.

Exemple 2.3.1 (Continuació) Si ho fem en el sistema anterior, el primer pas de Gauss queda igual i el segon és

$$\begin{aligned} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 1 & 0 & -1 \end{array} \right) & \xrightarrow{F_2 \leftrightarrow F_3} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0.0001 & 1 & 1 \end{array} \right) \\ & \xrightarrow{F_3 \leftarrow F_3 - 10^{-4}F_2} \left(\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & \boxed{1.000} \end{array} \right) \end{aligned}$$

d'on, fent substitució enrere, obtenim

$$\begin{aligned} x_3 &= 1.000/1 = 1, \\ x_2 &= -1/1 = -1, \\ x_1 &= 1 - x_3 = 0. \end{aligned}$$

En aquesta solució aproximada amb 4 dígits, totes les components tenen totes les xifres significatives. ∇

No obstant, també es poden trobar exemples en els quals el pivotatge maximal per columnes també fracassa.

Exercici: Resoleu, amb aritmètica en punt flotant amb $b = 10$ i $t = 4$, el sistema

$$\left. \begin{aligned} x_1 &+ x_3 = 1 \\ 10000x_1 + x_2 + 20000x_3 &= 20000 \\ x_1 + x_2 + x_3 &= 0 \end{aligned} \right\},$$

i comproveu que s'obté el mateix que en resoldre el sistema de l'exemple 2.3.1 sense fer pivotatge

El problema en aquest últim exemple és que la matriu del sistema està *desequilibrada*, és a dir, entre els seus coeficients n'hi ha d'ordre de magnitud molt diferent. Existeixen *mètodes per equilibrar matrius*, que fan que el pivotatge maximal per columnes vagi bé en la pràctica totalitat dels casos. En matrius equilibrades sense problemes de divisors petits, genèricament el pivotatge maximal per columnes dona millors resultats que no pivotar. Es poden trobar exemples en els que no pivotar dona menys error que pivotar, però en aquests casos pivotar no significa un increment significatiu de l'error.

Formalitzem el mètode de Gauss amb pivotatge maximal per columnes en el següent

Algorisme 2.3.2 (*Mètode de Gauss amb pivotatge maximal per columnes*)

$$\begin{aligned} \forall k = 1, 2, \dots, n-1 & \quad (pas) \\ \text{sigui } i_{\max} \in \{k, k+1, \dots, n\} \text{ t.q. } |a_{i_{\max}, k}^{(k)}| &= \max_{i=k, k+1, \dots, n} |a_{i, k}^{(k)}| \\ \text{intercanviem les files } k \text{ i } i_{\max} \text{ de } A^{(k)} & \\ \forall i = k+1, k+2, \dots, n & \quad (fila) \end{aligned}$$

$$\begin{aligned}
m_{i,k} &:= a_{i,k}^{(k)} / a_{k,k}^{(k)} && (\text{multiplicador, } a_{k,k}^{(k)} \text{ pivot}) \\
\forall j &= k+1, k+2, \dots, n && (\text{columna de la matriu de coeficients}) \\
a_{i,j}^{(k+1)} &= a_{i,j}^{(k)} - m_{i,k} a_{k,j}^{(k)} \\
\forall j &= 1, 2, \dots, p && (\text{columnes dels termes independents}) \\
b_{i,j}^{(k+1)} &= b_{i,j}^{(k)} - m_{i,k} b_{k,j}^{(k)}
\end{aligned}$$

En comptes de fer pivotatge maximal per columnes, existeix la possibilitat d'efectuar *pivotatge complet*, que es pot descriure mitjançant el següent

Algorisme 2.3.3 (*Mètode de Gauss amb pivotatge complet*)

$$\begin{aligned}
\forall k &= 1, 2, \dots, n-1 && (\text{pas}) \\
&\text{sigui } i_{\max}, j_{\max} \in \{k, k+1, \dots, n\} \text{ t.q. } |a_{i_{\max}, j_{\max}}^{(k)}| = \max_{i,j=k,k+1,\dots,n} |a_{i,j}^{(k)}| \\
&\text{intercanviem les files } k \text{ i } i_{\max} \text{ de } A^{(k)} \\
&\text{intercanviem les columnes } k \text{ i } j_{\max} \text{ de } A^{(k)} \\
\forall i &= k+1, k+2, \dots, n && (\text{fila}) \\
m_{i,k} &:= a_{i,k}^{(k)} / a_{k,k}^{(k)} && (\text{multiplicador, } a_{k,k}^{(k)} \text{ pivot}) \\
\forall j &= k+1, k+2, \dots, n && (\text{columna matriu de coeficients}) \\
a_{i,j}^{(k+1)} &= a_{i,j}^{(k)} - m_{i,k} a_{k,j}^{(k)} \\
\forall j &= 1, 2, \dots, p && (\text{columna terme independent}) \\
b_{i,j}^{(k+1)} &= b_{i,j}^{(k)} - m_{i,k} b_{k,j}^{(k)}
\end{aligned}$$

Notem que, si fem pivotatge complet, intercanviar columnes implica reordenar les incògnites, i això ho hem d'“enregistrar” d'alguna manera per, en acabar, saber a quina incògnita es refereix cada columna (no ho hem especificat a l'algorisme 2.3.3).

Existeixen estudis de propagació dels errors d'arrodoniment que donen fites millors per al pivotatge complet que per al pivotatge maximal per columnes. No obstant, l'experiència pràctica demostra que aquesta millora teòrica no compensa la complicació addicional de reordenar incògnites i el temps de còmput addicional requerit per la cerca del pivot.

2.4 Descomposició LU

Definició 2.4.1 *Sigui A matriu no singular (és a dir, $\det A \neq 0$). Anomenarem descomposició LU de A tota descomposició*

$$A = LU$$

amb L, U matrius $n \times n$ de la forma

$$L = \begin{pmatrix} 1 & & & & \\ l_{2,1} & 1 & & & \\ l_{3,1} & l_{3,2} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ & & u_{3,3} & \dots & u_{3,n} \\ & & & \ddots & \vdots \\ & & & & u_{n,n} \end{pmatrix},$$

és a dir, L triangular inferior amb 1's a la diagonal i U triangular superior.

Una de les utilitats d'aquest tipus de descomposició és que, si tenim $A = LU$, podem resoldre un sistema lineal $Ax = b$ en dues etapes:

- trobem $y \in \mathbb{R}^n$ tal que $Ly = b$,
- trobem $x \in \mathbb{R}^n$ tal que $Ux = y$.

Cadascun d'aquests sistemes és triangular i requereix n^2 operacions (de fet $Ly = b$ requereix n operacions menys, degut als elements 1 de la diagonal de L). Així, un cop tenim la descomposició LU , per resoldre diversos sistemes lineals amb la mateixa matriu de coeficients,

$$Ax_1 = b_1, Ax_2 = b_2, Ax_3 = b_3, \dots, \quad (2.1)$$

podem resoldre cadascun en $2n^2$ operacions ($2n^2 - n$, de fet), contra les $2n^3/3 + O(n^2)$ de fer tota l'eliminació gaussiana.

De fet, ja hem vist com resoldre diversos sistemes lineals alhora amb la mateixa matriu de coeficients mitjançant una única eliminació gaussiana: es tracta de fer servir un dels algorismes 2.2.2, 2.3.2 o 2.3.3 amb $p > 1$. En aquest cas, fer servir la descomposició LU no suposa cap avantatge. Però, per a poder fer servir un dels algorismes 2.2.2, 2.3.2 o 2.3.3 amb $p > 1$, cal conèixer tots vectors b_1, b_2, \dots de (2.1). Hi ha situacions en les que el vector b_{i+1} no es coneix fins que no s'ha resolt el sistema $Ax_i = b_i$. Vegeu el mètode de la potència desplaçada a la secció 2.7.3.

Podem obtenir la descomposició LU com a producte de l'eliminació gaussiana, tal com mostra la següent

Proposició 2.4.2 *Si A matriu $n \times n$ no singular.*

- (a) *Si existeix la descomposició LU d' A , és única.*
- (b) *Si es pot dur a terme l'eliminació gaussiana de A sense pivotatges (o sigui, si es pot completar l'algorisme 2.2.2 amb $p = 0$), llavors existeix la descomposició LU d' A , i és*

$$L = \begin{pmatrix} 1 & & & \\ m_{2,1} & 1 & & \\ \vdots & \vdots & \ddots & \\ m_{n,1} & m_{n,2} & \dots & 1 \end{pmatrix}, \quad U = A^{(n)} = \begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \dots & a_{1,n}^{(1)} \\ & a_{2,2}^{(2)} & \dots & a_{2,n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{n,n}^{(n)} \end{pmatrix} \quad (2.2)$$

Prova: Vegem primer (a). Suposem $A = L_1U_1 = L_2U_2$, on L_1, L_2 són triangulars inferiors amb 1's a la diagonal, i U_1, U_2 són triangulars superiors. Aleshores tenim

$$L_1U_1 = L_2U_2.$$

Multiplicant per la dreta per U_1^{-1} als dos costats de la igualtat, obtenim

$$L_1 = L_2U_2U_1^{-1},$$

i multiplicant per l'esquerra per L_2^{-1} a les dues bandes de la igualtat, obtenim

$$L_2^{-1}L_1 = U_2U_1^{-1}. \quad (2.3)$$

Ara hem de tenir en compte que

- El producte dues matrius triangulars superiors és una matriu triangular superior.
- La inversa d'una matriu triangular superior no singular és una matriu triangular superior.
- El producte de dues matrius triangulars inferiors amb 1's a la diagonal és una matriu triangular inferior amb 1's a la diagonal.
- La inversa d'una matriu triangular inferior amb 1's a la diagonal és una matriu triangular inferior amb 1's a la diagonal.

Exercici: *Demostreu totes aquestes afirmacions.*

Aleshores, podem representar esquemàticament la igualtat (2.3) així:

$$\begin{pmatrix} 1 & & & \\ * & 1 & & \\ \vdots & \vdots & \ddots & \\ * & * & \dots & 1 \end{pmatrix} = \begin{pmatrix} \times & \square & \dots & \square \\ & \times & \dots & \square \\ & & \ddots & \vdots \\ & & & \times \end{pmatrix}.$$

D'aquí es dedueix que els símbols \times són 1 necessàriament, i tant els símbols $*$ com \square són necessàriament zero. Per tant,

$$L_2^{-1}L_1 = U_2U_1^{-1} = I,$$

d'on $L_1 = L_2$ i $U_1 = U_2$.

Anem a provar (b). Suposem que portem a terme l'algorisme 2.2.2 amb $p = 0$. Podem escriure matricialment el pas k així:

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -m_{n,k} & & 1 \end{pmatrix} \begin{pmatrix} a_{1,1}^{(1)} & \dots & a_{1,k}^{(1)} & a_{1,k+1}^{(1)} & \dots & a_{1,n}^{(1)} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \dots & a_{k,n}^{(k)} \\ & & a_{k+1,k}^{(k)} & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ & & \vdots & \vdots & & \vdots \\ & & a_{n,k}^{(k)} & a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{pmatrix} = \begin{pmatrix} a_{1,1}^{(1)} & \dots & a_{1,k}^{(1)} & a_{1,k+1}^{(1)} & \dots & a_{1,n}^{(1)} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \dots & a_{k,n}^{(k)} \\ & & 0 & a_{k+1,k+1}^{(k+1)} & \dots & a_{k+1,n}^{(k+1)} \\ & & \vdots & \vdots & & \vdots \\ & & 0 & a_{n,k+1}^{(k+1)} & \dots & a_{n,n}^{(k+1)} \end{pmatrix},$$

que denotarem abreujadament com

$$F_k A^{(k)} = A^{(k+1)}.$$

Iterant la relació anterior, obtenim

$$U := A^{(n)} = F_{n-1}F_{n-2} \dots F_2F_1A^{(1)},$$

que és triangular superior. Aïllant $A^{(1)}$ de la igualtat anterior, obtenim

$$F_1^{-1}F_2^{-1}\dots F_{n-2}^{-1}F_{n-1}^{-1}U = A^{(1)} = A.$$

Per a acabar, només hem de veure que, si definim

$$L := F_1^{-1}F_2^{-1}\dots F_{n-2}^{-1}F_{n-1}^{-1},$$

es verifica la primera igualtat de (2.2).

Per a això, escrivim F_k en forma més manipulable. Considerem els vectors

$$m_k := \begin{pmatrix} 0 \\ \vdots (k) \\ 0 \\ m_{k+1,k} \\ m_{k+2,k} \\ \vdots \\ m_{n,k} \end{pmatrix}, \quad e_k := \begin{pmatrix} 0 \\ \vdots (k-1) \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

(e_k és el k -èsim vector de la base canònica). Aleshores

$$\begin{aligned} m_k e_k^\top &= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ m_{k+1,k} \\ \vdots \\ m_{n,k} \end{pmatrix} (0 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 0) \\ &= \begin{pmatrix} 0 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & & & \\ & & & 0 & & & \\ & & & m_{k+1,k} & 0 & & \\ & & & \vdots & & \ddots & \\ & & & m_{n,k} & & & 0 \end{pmatrix} \end{aligned} \quad (2.4)$$

d'on

$$F_k = I - m_k e_k^\top.$$

Anem a comprovar que

$$F_k^{-1} = I + m_k e_k^\top.$$

En efecte,

$$\begin{aligned} (I + m_k e_k^\top)(I - m_k e_k^\top) &= I - m_k e_k^\top + m_k e_k^\top - m_k e_k^\top m_k e_k^\top \\ &= I - m_k (e_k^\top m_k) e_k^\top = I, \end{aligned}$$

donat que

$$e_k^\top m_k = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ m_{k+1,k} \\ m_{k+2,k} \\ \vdots \\ m_{n,k} \end{pmatrix} = 0.$$

Aleshores

$$\begin{aligned} L &= F_1^{-1} F_2^{-1} \dots F_{n-1}^{-1} \\ &= (I + m_1 e_1^\top)(I + m_2 e_2^\top) \dots (I + m_{n-1} e_{n-1}^\top) \\ &= I + m_1 e_1^\top + m_2 e_2^\top + \dots + m_{n-1} e_{n-1}^\top. \end{aligned}$$

Anem a justificar l'última igualtat. El resultat del producte $(I + m_1 e_1^\top) \dots (I + m_{n-1} e_{n-1}^\top)$ s'obté sumant tots els monomis que s'obtenen d'agafar un terme de cada parèntesi i multiplicar. Hi ha tres possibilitats:

- Agafar I a tots els parèntesis, que només dona el monomi I .
- Agafar I a tots els parèntesis llevat d'un. D'aquesta manera s'obtenen els monomis

$$m_1 e_1^\top, m_2 e_2^\top, \dots, m_{n-1} e_{n-1}^\top.$$

- Agafar almenys dos $m_i e_i^\top$. D'aquesta manera s'obtenen monomis de la forma

$$\dots m_i e_i^\top m_j e_j^\top \dots = \dots m_i (e_i^\top m_j) e_j^\top \dots$$

per a $i < j$. Tots aquests monomis són zero, donat que

$$e_i^\top m_j = \underbrace{(0 \dots 0 1)}_i \begin{pmatrix} 0 \\ \vdots (j) \\ 0 \\ m_{j+1,j} \\ \vdots \\ m_{n,j} \end{pmatrix} \stackrel{j \geq i}{=} 0$$

Finalment, usant (2.4) amb $k = i$ per a $i = 1, \dots, n$, obtenim

$$\begin{aligned} L &= I + m_1 e_1^\top + m_2 e_2^\top + \dots + m_{n-1} e_{n-1}^\top \\ &= \begin{pmatrix} 1 & & & \\ m_{2,1} & 1 & & \\ \vdots & \vdots & \ddots & \\ m_{n,1} & m_{n,2} & \dots & 1 \end{pmatrix}, \end{aligned}$$

tal com volíem veure. □

A la pràctica, la descomposició LU s'implementa usant l'algorisme 2.2.2 amb $p = 0$, però s'aprofiten les posicions zero de la matriu de coeficients per guardar els multiplicadors. Així, començant amb $A^{(1)} = A$, a l'inici del pas k tindrem

$$\begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \dots & a_{1,k}^{(1)} & a_{1,k+1}^{(1)} & \dots & a_{1,n}^{(1)} \\ m_{2,1} & a_{2,2}^{(2)} & \dots & a_{2,k}^{(2)} & a_{2,k+1}^{(2)} & \dots & a_{2,n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ m_{k,1} & m_{k,2} & \dots & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \dots & a_{k,n}^{(k)} \\ m_{k+1,1} & m_{k+1,2} & \dots & a_{k+1,k}^{(k)} & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ m_{n,1} & m_{n,2} & \dots & a_{n,k}^{(k)} & a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{pmatrix},$$

mentre que, un cop finalitzat el pas k , tindrem

$$\begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \dots & a_{1,k}^{(1)} & a_{1,k+1}^{(1)} & \dots & a_{1,n}^{(1)} \\ m_{2,1} & a_{2,2}^{(2)} & \dots & a_{2,k}^{(2)} & a_{2,k+1}^{(2)} & \dots & a_{2,n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ m_{k,1} & m_{k,2} & \dots & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \dots & a_{k,n}^{(k)} \\ m_{k+1,1} & m_{k+1,2} & \dots & m_{k+1,k} & a_{k+1,k+1}^{(k+1)} & \dots & a_{k+1,n}^{(k+1)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,k} & a_{n,k+1}^{(k+1)} & \dots & a_{n,n}^{(k+1)} \end{pmatrix}$$

En acabar (al final del pas $n - 1$), tindrem

$$\begin{pmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \dots & \dots & a_{1,n}^{(1)} \\ m_{2,1} & a_{2,2}^{(2)} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n-1}^{(n-1)} & a_{n-1,n}^{(n-1)} \\ m_{n,1} & \dots & \dots & m_{n,n-1} & a_{n,n}^{(n)} \end{pmatrix},$$

de manera que a la part de sota de la diagonal tenim L , i la diagonal juntament amb la part de sobre formen U .

Exercici: *Descriviu en forma algorísmica la resolució d'un sistema lineal $Ax = b$ un cop es té la descomposició LU . Recordeu que cal resoldre dos sistemes,*

$$Ly = b, \quad Ux = y,$$

el primer dels quals és triangular inferior amb 1's a la diagonal i el segon és triangular superior.

2.4.1 Descomposició LU amb pivotatge maximal per columnes

Tal com hem plantejat la descomposició LU , presenta dos inconvenients

- Existeixen matrius no singulars sense descomposició LU .
- Tot i que la descomposició LU existeixi, la seva obtenció pot ser numèricament inestable.

Ens preguntem llavors si el mètode de Gauss amb pivotatge maximal per columnes (algorisme 2.3.2) dóna algun tipus de descomposició LU . La resposta és afirmativa, i aquesta descomposició no és la de la matriu inicial sinó la d'una permutació de les seves files. Abans d'enunciar-ho en forma de proposició, introduïm les matrius de permutació.

Definició 2.4.3 *Direm que una matriu $n \times n$, P , és una matriu de permutació si les seves files (o, equivalentment, les seves columnes) són una reordenació de les de la identitat.*

La utilitat de les matrius de permutació consisteix en que multiplicades per l'esquerra reordenen files, mentre que multiplicades per la dreta reordenen columnes. Esquemàticament, suposem que

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} e_3 \\ e_1 \\ e_2 \end{bmatrix} \end{pmatrix} = \left(\begin{bmatrix} e_2 \\ e_3 \\ e_1 \end{bmatrix} \right),$$

on e_1, e_2, e_3 són els vectors de la base canònica de \mathbb{R}^3 , i els usem indistintament com a files o columnes. Aleshores, si f_1, f_2, f_3 són vectors fila qualssevol,

$$P \begin{pmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} f_3 \\ f_1 \\ f_2 \end{bmatrix} \end{pmatrix}.$$

i, si c_1, c_2, c_3 són vectors columna qualssevol,

$$\left(\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \right) P = \left(\begin{bmatrix} c_2 \\ c_3 \\ c_1 \end{bmatrix} \right),$$

Proposició 2.4.4 *Sigui A matriu $n \times n$ amb $\det A \neq 0$. Aleshores existeix P matriu de permutació i existeixen L, U de la forma*

$$L = \begin{pmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n,1} & \dots & l_{n,n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ & u_{2,2} & \ddots & \vdots \\ & & \ddots & u_{n-1,n} \\ & & & u_{n,n} \end{pmatrix},$$

$$P = \begin{pmatrix} \begin{bmatrix} e_{s_1} \\ \vdots \\ e_{s_n} \end{bmatrix} \end{pmatrix},$$

on $\begin{bmatrix} e_{s_i} \end{bmatrix}$ representa una fila amb el s_i -èsim vector de la base canònica, tals que

$$PA = LU.$$

A més, P, L, U es poden trobar amb el següent algorisme:

$A^{(1)} := A$
 $\forall i = 1, 2, \dots, n$
 $s_i := i$
 $\forall k = 1, 2, \dots, n-1$
 $A^{(k)} =: (a_{i,j})_{i,j=1,2,\dots,n}$
sigui i_{max} t.q. $|a_{i_{max},k}| = \max_{i=k,k+1,\dots,n} |a_{i,k}|$
intercanviem files i_{max} i k de $A^{(k)}$ (TOTA la fila).
intercanviem s_k i $s_{i_{max}}$
 $\forall i = k+1, k+2, \dots, n$
 $a_{i,k} \leftarrow a_{i,k}/a_{k,k}$
 $\forall j = k+1, k+2, \dots, n$
 $a_{i,j} \leftarrow a_{i,j} - a_{i,k}a_{k,j}$
 $A^{(k+1)} := (a_{i,j})_{i,j=1,2,\dots,n}$

Prova: No la fem. La podeu trobar a Golub & Van Loan: *Matrix Computations*, 3a edició. És el teorema 3.4.1. \square

Il·lustrem l'aplicació d'aquesta proposició amb un exemple.

Exemple 2.4.5 Anem a fer la descomposició LU amb pivotatge de la següent matriu.

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

Comencem amb $s = (1, 2, 3)$.

$$\begin{aligned}
A^{(1)} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 6 \end{pmatrix} &\xrightarrow[s=(3,2,1)]{F_1 \leftrightarrow F_3} \begin{pmatrix} 3 & 4 & 6 \\ 2 & 3 & 4 \\ 1 & 2 & 3 \end{pmatrix} \\
&\xrightarrow{F_2 \leftarrow F_2 - (2/3)F_1, F_3 \leftarrow F_3 - (1/3)F_1} \begin{pmatrix} 3 & 4 & 6 \\ \boxed{2/3} & 1/3 & 0 \\ \boxed{1/3} & 2/3 & 1 \end{pmatrix} =: A^{(2)} \\
&\xrightarrow[s=(3,1,2)]{F_2 \leftrightarrow F_3} \begin{pmatrix} 3 & 4 & 6 \\ 1/3 & 2/3 & 1 \\ 2/3 & 1/3 & 0 \end{pmatrix} \\
&\xrightarrow{F_3 \leftarrow F_3 - (1/2)F_2} \begin{pmatrix} 3 & 4 & 6 \\ 1/3 & 2/3 & 1 \\ 2/3 & \boxed{1/2} & -1/2 \end{pmatrix} =: A^{(3)}
\end{aligned}$$

Com que, en acabar, $s = (3, 1, 2)$, tenim

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 2/3 & 1/2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 4 & 6 \\ 0 & 2/3 & 1 \\ 0 & 0 & -1/2 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Comproveu que, efectivament, $PA = LU$. ∇

2.5 Càlcul de determinants i inverses de matrius

Primer de tot, s'ha de remarcar que el determinant no és una quantitat gaire rellevant numèricament. En particular, NO és una bona mesura de “com a prop de singular” es troba una matriu. És molt fàcil trobar matrius gairebé singulars amb determinant molt gran, i també matrius “fortament” regulars amb determinant molt petit.

Un cop dit això, per calcular el determinant es pot fer servir qualsevol dels algorismes que hem vist, donat que tots estan basats en l'eliminació gaussiana, i l'eliminació gaussiana és una transformació que preserva el determinant (llevat del signe, que canvia cada cop que intercanviem files). Així, per exemple, per trobar el determinant amb l'algorisme 2.3.2 (amb $p = 0$), s'han de comptar el nombre de cops que s'intercanvien files, i, en acabar, el determinant és

$$(-1)^{\#\text{intercanvis}} a_{1,1}^{(1)} a_{2,2}^{(2)} \dots a_{n,n}^{(n)}.$$

Respecte d'invertir matrius, primer remarcuem que s'ha d'evitar sempre que es pot. Un dels casos més freqüents en què sembla que s'ha d'invertir una matriu però no cal és per fer el producte

$$M^{-1}v,$$

on M és una matriu $n \times n$ i $v \in \mathbb{R}^n$. En una situació com aquesta NO s'ha d'invertir M i multiplicar per v , sinó que s'ha de resoldre el sistema

$$Mx = v.$$

Llavors, x és el vector desitjat.

Un cop dit això, per invertir una matriu $n \times n$ A , s'han de resoldre n sistemes lineals,

$$Ax_1 = e_1, Ax_2 = e_2, \dots, Ax_n = e_n,$$

on $e_i \in \mathbb{R}^n$ és l' i -èsim vector de la base canònica. Aleshores,

$$A^{-1} = \left(\begin{bmatrix} x_1 \end{bmatrix} \begin{bmatrix} x_2 \end{bmatrix} \dots \begin{bmatrix} x_n \end{bmatrix} \right).$$

La resolució dels sistemes es pot fer de diverses maneres, per exemple:

- Fent reducció de Gauss simultània de n sistemes lineals (algorisme 2.3.2 amb $p = n$).
- Fent la descomposició LU de A i resolent després els n sistemes lineals mitjançant substitució endavant i substitució endarrere.

2.6 Propagació de l'error en la solució de sistemes lineals

En aquest capítol hem estudiat mètodes directes per la solució de sistemes lineals. Per definició, aquests mètodes no presenten errors de truncament. Sí que estan afectats pels errors d'arrodoniment, estudiats al capítol anterior, i també propaguen els errors en les dades d'entrada. Existeix una tècnica, coneguda com *anàlisi de l'error cap endarrere* (backward error analysis), segons la qual el resultat corresponent a un seguit d'operacions en punt flotant es pot obtenir

aplicant el mateix algorisme suposant operacions exactes a dades d'entrada lleugerament pertorbades. Gràcies a això, l'estudi de la propagació d'errors a les dades d'entrada en la resolució de sistemes lineals permet estudiar també la propagació dels errors d'arrodoniment.

En aquesta secció volem tractar la propagació d'errors en la solució de sistemes lineals mitjançant el concepte de *nombre de condició*. Per a això cal poder mesurar tant vectors com matrius, de manera que necessitem el concepte de norma.

Definició 2.6.1 (norma vectorial) Una norma vectorial sobre \mathbb{C}^n és una aplicació $\|\cdot\| : \mathbb{C}^n \rightarrow \mathbb{R}$ que satisfà, per a $x, y \in \mathbb{C}^n$ i $\alpha \in \mathbb{C}$ arbitraris:

- (a) $\|x\| \geq 0$,
- (b) $\|x\| = 0$ si i només si $x = 0$,
- (c) $\|\alpha x\| = |\alpha| \|x\|$,
- (d) $\|x + y\| \leq \|x\| + \|y\|$ (desigualtat triangular).

Algunes normes habituals són, per a $x = (x_1, \dots, x_n) \in \mathbb{C}^n$,

- $\|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \dots + |x_n|^2}$ (norma euclidiana).
- $\|x\|_\infty = \max\{|x_1|, |x_2|, \dots, |x_n|\}$ (norma del màxim).
- $\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$.

Definició 2.6.2 (norma matricial) Denotem per $\mathbb{C}^{n \times n}$ el conjunt de matrius $n \times n$ a coeficients complexos. Una norma matricial és una aplicació $\|\cdot\| : \mathbb{C}^{n \times n} \rightarrow \mathbb{R}$ que és una norma vectorial sobre \mathbb{C}^{n^2} mitjançant la identificació de $\mathbb{C}^{n \times n}$ i \mathbb{C}^{n^2} .

Definició 2.6.3 (norma matricial multiplicativa) Una norma matricial és multiplicativa si satisfà $\|AB\| \leq \|A\| \|B\|$, per a $A, B \in \mathbb{C}^{n \times n}$ arbitràries.

Definició 2.6.4 (norma matricial subordinada a norma vectorial) Donada una norma vectorial $\|\cdot\|$, la norma matricial subordinada, $\|\cdot\|_m$, es defineix mitjançant

$$\|A\|_m := \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|Ax\|}{\|x\|} = \max_{\substack{x \in \mathbb{C}^n \\ \|x\|=1}} \|Ax\|.$$

Habitualment denotarem una norma matricial subordinada a una de vectorial igual que la norma vectorial, i, per tant, escriurem

$$\|A\| = \max_{\substack{x \in \mathbb{C}^n \\ \|x\|=1}} \|Ax\|.$$

Definició 2.6.5 (radi espectral) Per a $A \in \mathbb{C}^{n \times n}$, el radi espectral de A , que es denota per $\rho(A)$, és el mòdul màxim dels seus valors propis, és a dir

$$\rho(A) = \max\{|\lambda| : \lambda \in \text{Spec } A\}.$$

Proposició 2.6.6 Les normes matricials associades a les normes vectorials habituals es poden calcular mitjançant les expressions:

$$(a) \|A\|_2 = \sqrt{\rho(A^H A)}, \text{ on } A^H := \overline{A}^\top.$$

$$(b) \|A\|_\infty = \max_{1 \leq i \leq n} \left\{ \sum_{j=1}^n |a_{ij}| \right\},$$

$$(c) \|A\|_1 = \max_{1 \leq j \leq n} \left\{ \sum_{i=1}^n |a_{ij}| \right\},$$

on $A \in \mathbb{C}^{n \times n}$.

Definició 2.6.7 (nombre de condició) Donada una norma matricial $\|\cdot\|$ subordinada a una norma vectorial, per a $A \in \mathbb{C}^{n \times n}$ arbitrària es defineix el seu nombre de condició en aquesta norma per

$$\mu(A) = \|A\| \|A^{-1}\|.$$

Per a les normes habituals, denotarem

$$\mu_2(A) = \|A\|_2 \|A^{-1}\|_2, \quad \mu_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty, \quad \mu_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

El fet que el nombre de condició controla la propagació d'errors mitjançant la resolució de sistemes lineals queda establert per la següent proposició.

Proposició 2.6.8 Sigui $A \in \mathbb{C}^{n \times n}$ no singular ($\det A \neq 0$), $b \in \mathbb{C}^n$ i denotem per x la solució del sistema $Ax = b$. Considerem $\Delta b \in \mathbb{C}^n$ una pertorbació del terme independent i denotem per $x + \Delta x$ la solució del sistema pertorbat, és a dir, $A(x + \Delta x) = b + \Delta b$. Es compleixen les estimacions

$$(a) \|\Delta x\| \leq \|A^{-1}\| \|\Delta b\|,$$

$$(b) \frac{\|\Delta x\|}{\|x\|} \leq \mu(A) \frac{\|\Delta b\|}{\|b\|},$$

on les normes matricial i vectorial que hi intervenen són associades.

Prova: Com que $Ax + A\Delta x = A(x + \Delta x) = b + \Delta b$ i sabem que $Ax = b$, obtenim que $A\Delta x = \Delta b$, d'on $\Delta x = A^{-1}\Delta b$ i, de la definició de norma matricial subordinada a una de vectorial,

$$\|\Delta x\| \leq \|A^{-1}\| \|\Delta b\|. \quad (2.5)$$

Per a provar la segona desigualtat, ens cal fitar $\|x\|$ inferiorment. Ho fem a partir de $Ax = b$ i la definició de norma subordinada:

$$b = Ax \Rightarrow \|b\| \leq \|A\| \|x\| \Rightarrow \|x\| \geq \|b\| / \|A\|.$$

Fent servir això, de (2.5) obtenim

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \|\Delta b\|}{\|x\|} \leq \frac{\|A^{-1}\| \|\Delta b\| \|A\|}{\|b\|} = \mu(A) \frac{\|\Delta b\|}{\|b\|},$$

com volíem veure. □

Les matrius que tenen nombre de condició molt gran s'anomenen *matrius mal condicionades*. Quan resollem sistemes lineals, el nombre de dígit amb què treballem determina quins són els nombres de condició més grans amb què podem obtenir solucions amb sentit.

Exemple 2.6.9 Les matrius de Hilbert constitueixen una família molt coneguda de matrius mal condicionades. La matriu de Hilbert $n \times n$ es defineix com

$$H_n = \left(\frac{1}{i+j-1} \right)_{1 \leq i,j \leq n}.$$

Per exemple,

$$H_2 = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}.$$

En Octave, la matriu de Hilbert $n \times n$ s'obté mitjançant la comanda `hilb(n)`. Podeu tabular els nombres de condició en $\|\cdot\|_\infty$ de les matrius de Hilbert per a $n = 1, 2, \dots, 10$ mitjançant

```
for i=1:10; cndh(i)=cond(hilb(i),Inf); end; transpose(cndh)
```

Suposem que volem resoldre un sistema lineal $Ax = b$ en IEEE-doble, on A és una matriu de Hilbert. D'acord amb l'anàlisi de l'error cap endarrere, l'efecte dels errors d'arrodoniment es pot tenir en compte a través d'una pertorbació adequada dels coeficients del sistema i el terme independent. No calculem aquestes pertorbacions. No obstant, per fer-nos una idea del seu efecte, anem a suposar que són de l'ordre de l'èpsilon-màquina, i, per simplificar, suposarem que només afecten el terme independent. Aleshores, d'acord amb la proposició 2.6.8, hem d'esperar un error en $\|\cdot\|_\infty$ igual a $\mu_\infty(H_n)\epsilon$. En Octave, podem avaluar aquesta estimació de l'error així:

```
cond(hilb(n),Inf)*eps/2
```

Amb el següent codi, podem veure quin és l'error que es comet en resoldre el sistema $H_n x = b$ per a $b = H_n(1, 1, \dots, 1)^\top$:

```
n=10; A=hilb(n); xe=ones(n,1); b=A*xe; x=A\b; norm(xe-x,Inf)
```

Podeu observar com l'error és inferior al predit, però és igual de dràstic: perdem casi totes les xifres. Això no passa amb matrius "habituals", tal com podeu comprovar si substituïu `hilb(n)` per `rand(n)` a la línia anterior. ∇

2.7 Vectors i valors propis

Recordeu que, per a una matriu $A \in \mathbb{C}^{n \times n}$, es diu que $\lambda \in \mathbb{C}$ és **valor propi** de A sí existeix $v \in \mathbb{C}^n$, $v \neq 0$ tal que $Av = \lambda v$. En aquest cas, v s'anomena **vector propi** de valor propi λ . Denotarem

$$\lambda \in \text{Spec } A, \quad v \in V_\lambda(A).$$

2.7.1 Teorema de Gershgorin

És un resultat que permet obtenir molt fàcilment estimacions dels valors propis d'una matriu en termes dels seus coeficients.

Teorema 2.7.1 (Gershgorin) *Tota matriu $A = (a_{i,j})_{1 \leq i,j \leq n} \in \mathbb{C}^{n \times n}$ compleix*

$$\text{Spec } A \subset \bigcup_{i=1}^n \{\mu \in \mathbb{C} : |\mu - a_{i,i}| \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{i,j}|\}.$$

Dit en paraules: els valors propis de A es troben a la unió dels discos amb centres als elements de la diagonal i radis la suma dels valors absoluts dels elements de la resta de la columna.

La demostració la podeu trobar per exemple al llibre de Stoer & Bulirsch.

Nota 2.7.2 Com que $\text{Spec } A^\top = \text{Spec } A$, el teorema de Gershgorin també es pot aplicar per files. ∇

Exemple 2.7.3 Considerem la matriu

$$A = \begin{pmatrix} 1 & 0.1 & -0.1 \\ 0 & 2 & 0.4 \\ -0.2 & 0 & 3 \end{pmatrix}$$

Aleshores els valors propis de A estan inclosos a la següent unió de discos:

$$\{\mu : |\mu - 1| < 0.2\} \cup \{\mu : |\mu - 2| < 0.01\} \cup \{\mu : |\mu - 3| < 0.5\}.$$

∇

Al llibre de Stoer & Bulirsch també trobareu la prova del següent corol·lari:

Corol·lari 2.7.4 *Si la unió de k dels discos de Gershgorin és disjunta amb la unió dels $n - k$ discos restants, llavors la primera unió conté exactament k valors propis (comptant multiplicitats), mentre que la segona en conté $n - k$ (també comptant multiplicitat).*

A l'exemple 2.7.3, com que els tres discos són disjunts, aplicant el corol·lari deduïm que cada disc conté exactament un valor propi.

2.7.2 Mètode de la potència

Es tracta d'una estratègia senzilla per trobar el valor propi dominant d'una matriu $A \in \mathbb{C}^{n \times n}$, juntament amb un vector propi corresponent.

Teorema 2.7.5 (Mètode de la potència amb quocients de Rayleigh) *Suposem $A \in \mathbb{C}^{n \times n}$, $\text{Spec } A = \{\lambda_1, \dots, \lambda_n\} \subset \mathbb{C}$, i*

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|.$$

Sigui $\{u_1, \dots, u_n\}$ una base de vectors propis associada. Prenem un vector $x^{(0)}$ inicial

$$x^{(0)} = a_1 u_1 + a_2 u_2 + \dots + a_n u_n,$$

i suposem $a_1 \neq 0$. Li apliquem A iterativament i generem la successió

$$x^{(k+1)} := Ax^{(k)}, \quad k = 0, 1, 2, \dots$$

Considerem la successió de quocients de Rayleigh,

$$\sigma_k = \frac{x^{(k)\top} A x^{(k)}}{x^{(k)\top} x^{(k)}}.$$

Aleshores

(a) La successió de quocients de Rayleigh satisfà

$$\sigma_k = \lambda_1 + O\left(\left|\frac{\lambda_1}{\lambda_2}\right|^k\right),$$

i, en particular, $\lim_k \sigma_k = \lambda_1$.

(b) Si A és real simètrica, es pot prendre $\{u_1, \dots, u_n\} \subset \mathbb{R}^n$ base ortonormal i llavors

$$\sigma_k = \lambda_1 + O\left(\left|\frac{\lambda_1}{\lambda_2}\right|^{2k}\right).$$

Prova: Sabem que

$$\begin{aligned} x^{(k)} &= A^k x^{(0)} = A^k (a_1 u_1 + \dots + a_n u_n) \\ &= a_1 \lambda_1^k u_1 + \dots + a_n \lambda_n^k u_n. \end{aligned}$$

Calculem els quocients de Rayleigh:

$$\begin{aligned} \sigma_k &= \frac{x^{(k)\top} x^{(k+1)}}{x^{(k)\top} x^{(k)}} = \frac{(\sum_i a_i \lambda_i^k u_i^\top) (\sum_j a_j \lambda_j^{k+1} u_j)}{(\sum_i a_i \lambda_i^k u_i^\top) (\sum_j a_j \lambda_j^k u_j)} = \frac{\sum_{i,j} a_i a_j \lambda_i^k \lambda_j^{k+1} u_i^\top u_j}{\sum_{i,j} a_i a_j \lambda_i^k \lambda_j^k u_i^\top u_j} \\ &= \frac{a_1^2 \lambda_1^k \lambda_1^{k+1} u_1^\top u_1 + \sum_{(i,j) \neq (1,1)} a_i a_j \lambda_i^k \lambda_j^{k+1} u_i^\top u_j}{a_1^2 \lambda_1^k \lambda_1^{k+1} u_1^\top u_1 + \sum_{(i,j) \neq (1,1)} a_i a_j \lambda_i^k \lambda_j^k u_i^\top u_j} \\ &= \frac{\lambda_1^k \lambda_1^{k+1}}{\lambda_1^k \lambda_1^k} \cdot \frac{a_1^2 u_1^\top u_1 + \sum_{(i,j) \neq (1,1)} a_i a_j \frac{\lambda_i^k \lambda_j^{k+1}}{\lambda_1^k \lambda_1^{k+1}} u_i^\top u_j}{a_1^2 u_1^\top u_1 + \sum_{(i,j) \neq (1,1)} a_i a_j \frac{\lambda_i^k \lambda_j^k}{\lambda_1^k \lambda_1^k} u_i^\top u_j}. \end{aligned}$$

A la darrera expressió, la primera fracció és igual a λ_1 . A la segona fracció, com que per tot $(i, j) \neq (1, 1)$ tenim

$$\left| \frac{\lambda_i^k \lambda_j^{k+1}}{\lambda_1^k \lambda_1^{k+1}} \right| \leq \left| \frac{\lambda_2}{\lambda_1} \right|^k \quad \text{i} \quad \left| \frac{\lambda_i^k \lambda_j^k}{\lambda_1^k \lambda_1^k} \right| \leq \left| \frac{\lambda_2}{\lambda_1} \right|^k,$$

els dos sumatoris són $O(|\lambda_2/\lambda_1|^k)$. En conseqüència, la segona fracció és $1 + O(|\lambda_2/\lambda_1|^k)$, i, per tant, tenim (a).

Ara veiem (b). Com que la base $\{u_i\}_{i=1}^n \subset \mathbb{R}^n$ és ortonormal, tenim $u_i^\top u_j = 0$ si $i \neq j$, i llavors

$$\begin{aligned} \sigma_k &= \frac{\lambda_1^k \lambda_1^{k+1}}{\lambda_1^k \lambda_1^k} \cdot \frac{a_1^2 u_1^\top u_1 + \sum_{j=2}^n a_j^2 \frac{\lambda_j^{2k+1}}{\lambda_1^{2k+1}} \|u_j\|_2^2}{a_1^2 u_1^\top u_1 + \sum_{j=2}^n a_j^2 \frac{\lambda_j^{2k}}{\lambda_1^{2k}} \|u_j\|_2^2} \\ &= \lambda_1 \left(1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right) \right). \end{aligned}$$

□

Observació 2.7.6 Notem que:

1. El quocient de Rayleig és independent de reescalats. Per a veure-ho, considerem una successió del mètode de la potència $\{x^{(k)}\}_k$ tal com hem vist i una de reescalada $\{y^{(k)}\}_k$. Les definim per

$$\begin{aligned} x^{(0)} \text{ inicial, } & x^{(k+1)} = Ax^{(k)}, \quad k = 0, 1, 2, \dots \\ y^{(0)} \text{ inicial, } & y^{(k+1)} = \alpha_k Ay^{(k)}, \quad k = 0, 1, 2, \dots \end{aligned}$$

Aleshores $y^{(k)} = (\prod_{j=0}^k \alpha_j) x^{(k)} := \beta_k x^{(k)}$, i tenim

$$\frac{y^{(k)\top} Ay^{(k)}}{y^{(k)\top} \beta_k y^{(k)}} = \frac{\beta_k x^{(k)\top} A \beta_k x^{(k)}}{\beta_k y^{(k)\top} \beta_k y^{(k)}} = \sigma_k.$$

2. Com a conseqüència, podem normalitzar cada iterat del mètode de la potència. Així evitem overflows, en cas que $|\lambda| > 1$, o underflows, en cas que $|\lambda| < 1$. Podem fer:

$$\begin{aligned} & x^{(0)} \text{ inicial} \\ & y^{(0)} := x^{(0)} / \|x^{(0)}\| \\ & \forall k = 0, 1, 2, \dots \\ & \quad \sigma_k = y^{(k)\top} Ay^{(k)} \\ & \quad y^{(k+1)} = Ay^{(k)} / \|Ay^{(k)}\| \end{aligned}$$

Observeu que, com que normalitzem, el denominador al quocient de Rayleigh és 1.

3. Suposant, amb la notació de la demostració, que

$$x^{(0)} = a_1 u_1 + \dots + a_n u_n,$$

on $\{u_i\}_{i=1}^n$ és una base de vectors propis i $a_1 \neq 0$, es pot veure que

$$y^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|} = \frac{a_1 u_1}{\|a_1 u_1\|} + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right),$$

i, per tant, $y^{(k)}$ convergeix a un vector propi de valor propi λ_1 .

4. Si tenim la (poc probable) mala sort que triem un $x^{(0)}$ amb $a_1 = 0$, l'error d'arrodoniment farà que, al cap d'uns iterats, $a_1 \neq 0$. Així que, a efectes pràctics, podem triar u_1 qualsevol.

▽

2.7.3 Mètode de la potència inversa desplaçada

Es tracta d'una estratègia per trobar el valor propi d'una matriu que és més proper a un valor prefixat.

Sigui $A \in \mathbb{C}^{n \times n}$, i sigui $\mu \in \mathbb{C}$, $\mu \notin \text{Spec } A$ tal que el valor propi de A més proper a μ és únic. És a dir, suposem que existeix $\lambda \in \text{Spec } A$ tal que

$$|\lambda - \mu| < |\xi - \mu| \quad \forall \xi \in \text{Spec } A : \xi \neq \lambda. \quad (2.6)$$

Considerem la matriu $(A - \mu I)^{-1}$. Els seus valors propis són

$$\text{Spec}(A - \mu I)^{-1} = \{(\xi - \mu)^{-1} : \xi \in \text{Spec } A\}.$$

Si considerem $\Lambda := (\lambda - \mu)^{-1}$, de (2.6) tenim que Λ és el valor propi dominant de $(A - \mu I)^{-1}$. Per tant, si apliquem el mètode de la potència a $(A - \mu I)^{-1}$, trobarem Λ , i, a partir d'ell, obtindrem el valor de A més proper a μ mitjançant

$$\lambda = \Lambda^{-1} + \mu.$$

Observació 2.7.7 La descomposició LU és especialment adequada per a implementar el mètode de la potència inversa desplaçada: donat $x^{(0)} \in \mathbb{C}^n$ vector inicial, volem calcular

$$x^{(k+1)} := (A - \mu I)^{-1} x^{(k)}, \quad k = 0, 1, 2, \dots$$

fins a tenir convergència. Si trobem L triangular inferior amb uns a la diagonal i U triangular superior amb $A - \mu I = LU$, llavors podem trobar $x^{(k+1)}$ a partir de $x^{(k)}$ mitjançant

$$L(Ux^{(k+1)}) = x^{(k)},$$

que se resol mitjançant una substitució endavant seguida d'una substitució endarrere (vegeu les seccions 2.1.2 i 2.1.3). ∇

Exercici: *Comproveu que l'estratègia anterior suposa menys operacions que calcular $(A - \mu I)^{-1}$ i després anar multiplicant per aquesta matriu, tot i que la inversa només es calculi una vegada.*

Capítol 3

Solució numèrica d'equacions no lineals

En aquest capítol considerem equacions del tipus $f(x) = 0$ on f és una funció real de variable real. També considerem equacions del tipus $x = g(x)$.

Anomenem **arrel** o **solució** de l'equació $f(x) = 0$ un valor α tal que $f(\alpha) = 0$. Es diu també que α és un **zero** de f . En el cas d'una equació del tipus $x = g(x)$, anomenem **punt fix** de g un valor α tal que $g(\alpha) = \alpha$. L'objectiu d'aquest capítol és desenvolupar mètodes numèrics que ens permetin obtenir aproximacions numèriques d'arrels o punts fixos.

3.1 Mètodes de la bisecció i de Newton

3.1.1 Mètode de la bisecció

Suposem que tenim $f : [a, b] \rightarrow \mathbb{R}$ contínua i volem trobar α amb $f(\alpha) = 0$. Si $f(a)f(b) \leq 0$ (i.e., $f(a)$ i $f(b)$ tenen signes diferents), el teorema de Bolzano ens assegura que llavors existeix $\alpha \in [a, b]$ amb $f(\alpha) = 0$. L'aplicació recursiva del teorema de Bolzano a subdivisions de l'interval original dóna lloc al mètode de la bisecció, que podem descriure com segueix.

Algorisme 3.1.1 (mètode de la bisecció)

```
 $a_1 := a, b_1 := b, I_1 := [a_1, b_1]$   
 $\forall i = 1, 2, 3, \dots$   
     $p_i := \frac{a_i + b_i}{2}$   
    si  $f(a_i)f(p_i) \leq 0$   
         $a_{i+1} := a_i, b_{i+1} := p_i$   
    si no  
         $a_{i+1} := p_i, b_{i+1} := b_i$   
     $I_{i+1} := [a_{i+1}, b_{i+1}]$ 
```

El criteri d'aturada es discuteix a l'observació 3.1.3, punt (c).

Anem a provar que aquest algorisme aproxima arbitràriament una arrel de f .

Proposició 3.1.2 *Sigui $f : [a, b] \rightarrow \mathbb{R}$ contínua amb $f(a)f(b) \leq 0$ i suposem que apliquem l'algorisme 3.1.1. Llavors, $\forall i = 1, 2, 3, \dots$,*

$$\begin{aligned} \exists \alpha \in I_i : f(\alpha) = 0 \\ |p_i - \alpha| \leq \frac{b - a}{2^i} \end{aligned} \tag{3.1}$$

Prova: Anem a veure primer per inducció que, $\forall i$,

$$f(a_i)f(b_i) \leq 0 \quad \text{i} \quad b_i - a_i = \frac{b-a}{2^{i-1}}. \quad (3.2)$$

El cas $i = 1$ se segueix directament de les hipòtesis. Suposem (3.2) cert per i qualsevol i provem-ho per $i + 1$. Si $f(a_i)f(p_i) \leq 0$, prendrem $a_{i+1} = a_i$, $b_{i+1} = p_i$ i tindrem $f(a_{i+1})f(b_{i+1}) \leq 0$. Si no, s'ha de complir $f(p_i)f(b_i) \leq 0$, donat que, si fos $f(p_i)f(b_i) > 0$, com que estem suposant $f(a_i)f(p_i) > 0$ tindríem $f(a_i)f(b_i) > 0$, contràriament a la hipòtesi d'inducció. Per tant $f(p_i)f(b_i) \leq 0$, i, com que en aquest cas prenem $a_{i+1} = p_i$, $b_{i+1} = b_i$, tenim $f(a_{i+1})f(b_{i+1}) \leq 0$. Respecte de la segona igualtat de (3.2)

$$b_{i+1} - a_{i+1} = \frac{b_i - a_i}{2} \stackrel{(3.2)}{=} \frac{1}{2} \frac{b-a}{2^{i-1}} = \frac{b-a}{2^i},$$

i això finalitza la inducció.

Per a acabar, per la primera igualtat de (3.2), el teorema de Bolzano ens assegura que existeix $\alpha \in I_i$ amb $f(\alpha) = 0$. A més,

$$|p_i - \alpha| \leq \frac{b_i - a_i}{2} \stackrel{(3.2)}{=} \frac{1}{2} \frac{b-a}{2^{i-1}} = \frac{b-a}{2^i},$$

tal com volíem veure. □

Observació 3.1.3 La desigualtat (3.1)

(a) Ens diu que, si suposem p_i aproximació de α ,

$$|e_a(p_i, \alpha)| \leq \frac{b-a}{2^i}.$$

(b) En cas que α sigui l'única arrel de f a $[a, b]$, demostra convergència de la successió $\{p_i\}_i$ cap a α .

(c) Ens permet estimar el nombre d'iterats necessaris a l'algorisme 3.1.1 per a assolir una certa precisió. ▽

Exemple 3.1.4 Sigui $f(x) = \cos x - x$. Donat que $f(0) = 1 > 0$ i $f(\frac{\pi}{2}) = -\frac{\pi}{2} < 0$, existeix $\alpha \in (0, \frac{\pi}{2})$ amb $f(\alpha) = 0$. Anem a veure quantes iteracions cal fer per assegurar que l'error sigui més petit que 10^{-6} :

$$|p_n - \alpha| \leq \frac{b-a}{2^n} = \frac{\pi}{2^{n+1}} < 10^{-6}.$$

Per resoldre aquesta inequació utilitzarem la funció logaritme i el seu creixement:

$$\ln\left(\frac{\pi}{2^{n+1}}\right) = \ln(\pi) - (n+1)\ln(2) < -6\ln(10).$$

Per tant: $6\ln(10) + \ln(\pi) < (n+1)\ln 2$ que dona $n > 20.5830$, és a dir, és suficient prendre $n = 21$. ▽

3.1.2 Mètode de Newton

El mètode de Newton permet aproximar arrels d'equacions de la forma $f(x) = 0$. Anem a deduir-lo de dues maneres. Suposem que $f : [a, b] \rightarrow \mathbb{R}$ és de classe C^2 i que existeix $\alpha \in (a, b)$ amb $f(\alpha) = 0$.

Dedució analítica. Suposem que x és una aproximació de α , i expandim f per Taylor fins a ordre 1 al voltant de x :

$$\begin{aligned} 0 = f(\alpha) &= f(x) + f'(x)(\alpha - x) + O((\alpha - x)^2) \\ &\approx f(x) + f'(x)(\alpha - x), \end{aligned}$$

d'on

$$\alpha - x \approx -\frac{f(x)}{f'(x)}.$$

Així, $x - f(x)/f'(x) \approx x + (\alpha - x) = \alpha$ i esperem que $x - f(x)/f'(x)$ sigui millor aproximació de α que x . Això suggereix el següent algorisme iteratiu:

Algorisme 3.1.5 (mètode de Newton) *Escollim x_0 aproximació inicial de α , triem ε tolerància (fita de $|e_a(x_n, \alpha)|$), n_{\max} nombre màxim d'iterats que permetrem i fem*

$\forall i = 0, 1, \dots, n_{\max} - 1,$

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

 si $|x_{i+1} - x_i| \leq \varepsilon$, tornem x_{i+1} , STOP
 error (no convergència)

Dedució geomètrica. Donat x_i aproximació de α , prenem la recta tangent a f a x_i ,

$$y = f(x_i) + f'(x_i)(x - x_i),$$

la tallem amb l'eix x (fem $y = 0$ a dalt),

$$x = x_i - \frac{f(x_i)}{f'(x_i)}$$

i prenem $x_{i+1} := x$ com a aproximació millorada (vegeu la figura 3.1 (a)).

Òbviament les dues deduccions donen la mateixa fórmula, donat que l'aproximació de Taylor d'ordre 1 de f al voltant de x no és més que la recta tangent a la gràfica de f al punt $(x, f(x))$.

El mètode de Newton és molt més ràpid que el mètode de bisecció si $f'(\alpha) \neq 0$ (i.e., si α és una arrel simple). Veurem més endavant que, quan som a prop de l'arrel, a cada iterat aproximadament es duplica nombre de decimals correctes. En aquestes condicions $|x_{i+1} - x_i| \approx |\alpha - x_i|$, i és per això que a l'algorisme 3.1.5 la condició $|x_{i+1} - x_i| < \varepsilon$ és part del criteri d'aturada.

No obstant, quan som lluny de l'arrel el mètode de Newton pot convergir molt lentament (o no convergir), i és per això que a l'algorisme 3.1.5 es limita el nombre màxim d'iterats.

Exemple 3.1.6 Suposem que volem trobar un zero de $f(x) = e^x - 2$, sigui α (que ja sabem que és $\ln 2$). Si prenem $a = 0$ i $b = 1$, com que $f(a)f(b) < 0$, el teorema de Bolzano ens assegura que tenim una arrel $\alpha \in [a, b]$.

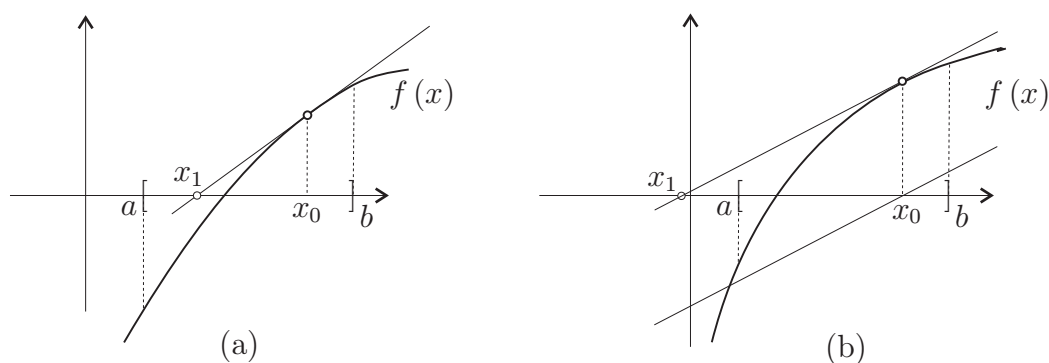


Figura 3.1: Representació gràfica d'un iterat del mètode de Newton.

Fem bisecció:

i	p_i	i	p_i
1	0.5	30	0.6931 4717 9670
2	0.75	31	0.6931 4718 0135
3	0.625	32	0.6931 4718 0370
4	0.6875	33	0.6931 4718 0485
\vdots	\vdots	\vdots	\vdots
10	0.6923 8281 2500	35	0.6931 4718 0570
11	0.6928 7019 3750	36	0.6931 4718 0555
\vdots	\vdots	37	0.6931 4718 0560
20	0.6931 4670 5625	38	ja no canvia
21	0.6931 4718 2460		

Fem Newton, començant igual:

i	x_i
0	0.5
1	0.7130 6131 9425
2	0.6933 4415 7318
3	0.6931 4719 9958
4	0.6931 4719 0558
5	ja no canvia

Observem que Newton és molt més ràpid. Ara prenem $a = -9$, $b = 1$ ($\Rightarrow f(a)f(b) < 0$) i

fem bisecció:

i	p_i	i	p_i
1	-4	21	0.6931 5052 0320
2	-1.5	\vdots	\vdots
3	-0.25	30	0.6931 5717 6870
4	0.375	31	0.6931 5718 1530
5	0.6875	\vdots	\vdots
\vdots	\vdots	39	0.6931 5718 0570
10	0.6972 5652 5	40	0.6931 5718 0560
11	0.6923 8281 25	41	0.6931 5718 0555
\vdots	\vdots	42	0.6931 5718 0560
20	0.6931 5528 8670	43	ja no canvia

mentre que, si fem Newton començant igual,

i	x_i	i	x_i
0	-4	103	2.3213 8700 863
1	104.1963 0006 6	\vdots	\vdots
2	103.1963 0006 6	107	0.6936 4539 4385
3	102.1963 0006 6	108	0.6931 4730 4647
\vdots	\vdots	109	0.6931 4718 0562
		110	ja no canvia

Observem que Newton precisa més iterats que bisecció! El motiu és que a $x = -4$ la gràfica de $f(x)$ és molt plana ($f'(-4) = e^{-4} \approx 0.0183$), de manera que la tangent per -4 envia el primer iterat al 104.1963. A partir d'aquí, i fins que s'apropa a l'arrel, baixa d'un en un perquè

$$x_{n+1} = x_n - \frac{e^{x_n} - 2}{e^{x_n}} = x_n - 1 + \frac{2}{e^{x_n}} \approx x_n - 1,$$

donat que x_n és gros. Un cop som a prop de l'arrel, la convergència torna a ser ràpida, com en el cas anterior. La convergència ràpida del mètode de Newton només està garantida quan el punt inicial és prou a prop de l'arrel. Més endavant quantificarem aquesta afirmació. ∇

3.2 Mètodes de punt fix

Un **punt fix** per a una funció g definida a $[a, b]$ és un nombre $\alpha \in [a, b]$ tal que $g(\alpha) = \alpha$. En aquesta secció ens dedicarem a trobar solucions a problemes de punt fix (i.e., equacions de la forma $x = g(x)$) i a estudiar la connexió entre aquests i els problemes de trobar zeros de funcions. Per això passarem de l'equació $f(x) = 0$ a $g(x) = x$, i estudiarem quina és la forma més interessant des del punt de vista numèric. Per exemple, si hem de resoldre $f(x) = x^3 - x^2 + 1 = 0$ podem passar a resoldre $g_1(x) = x^3 - x^2 + x + 1 = x$ o bé $g_2(x) = \pm\sqrt{x^3 + 1} = x$ o bé $g_3(x) = \sqrt[3]{x^2 - 1} = x$.

Una equació de la forma $x = g(x)$ suggereix de manera natural la iteració:

$$\begin{aligned} &x_0 \text{ aproximació inicial} \\ &\forall i = 0, 1, 2, \dots \\ &\quad x_{i+1} = g(x_i). \end{aligned}$$

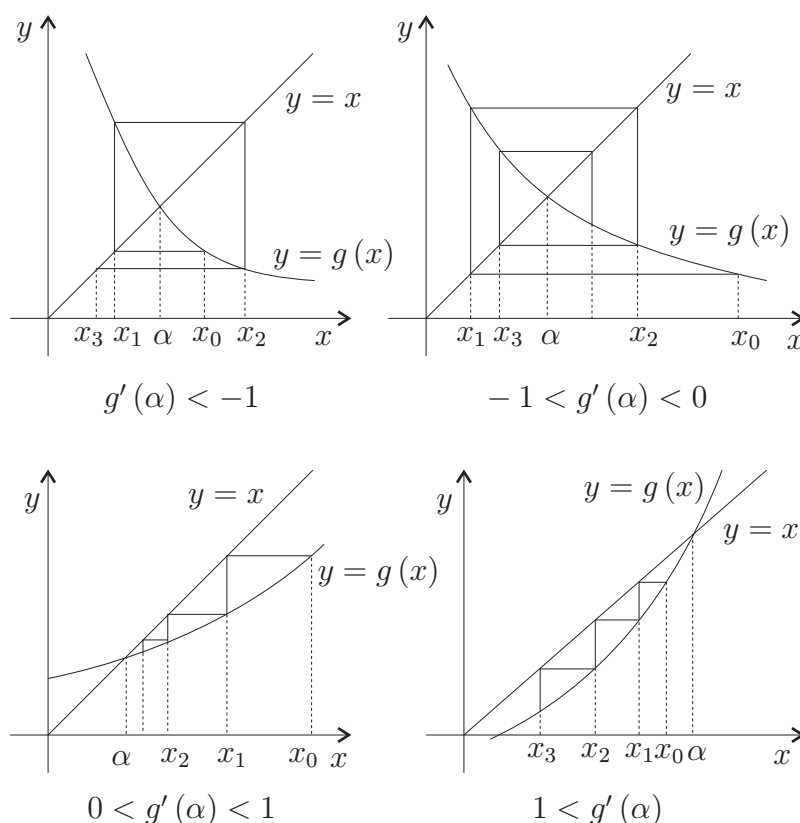


Figura 3.2: Tipus de comportaments de $x_{k+1} = g(x_k)$ al voltant d'un punt fix α (i.e. $g(\alpha) = \alpha$) amb $g'(\alpha) \neq 1$. D'esquerra a dreta i dalt a baix: teranyina divergent, teranyina convergent, escala convergent i escala divergent.

Ens preguntem: la successió d'iterats generada, convergeix? Si sí, a on? Sota quines condicions? A quina velocitat?

La pregunta “a on” és la més fàcil de respondre.

Lema 3.2.1 *Sigui $g : [a, b] \rightarrow \mathbb{R}$ contínua, prenem $x_0 \in [a, b]$, $x_{i+1} = g(x_i)$, suposem $\{x_n\}_{n \in \mathbb{N}} \subseteq [a, b]$ i que existeix $\lim_{n \rightarrow \infty} x_n =: \alpha$ ($\Rightarrow \alpha \in [a, b]$). Aleshores $g(\alpha) = \alpha$.*

Prova: Tenim

$$g(\alpha) = g\left(\lim_{n \rightarrow \infty} x_n\right) = \lim_{n \rightarrow \infty} g(x_n) = \lim_{n \rightarrow \infty} x_{n+1} = \alpha,$$

on, a la segona igualtat, hem usat que $\{x_n\}_n \subseteq [a, b]$ i que g és contínua a $[a, b]$. □

Les iteracions d'un mètode de punt fix es poden representar en una gràfica amb les corbes $y = g(x)$ i $y = x$ (veure figura 3.2). Donat x_0 , obtenim $g(x_0)$ “pujant” a la gràfica de g , però el volem representar sobre l'eix de les x per poder tornar a iterar. Per a això, ens desplace des de $(x_0, g(x_0))$ fins a $y = x$ a través d'una recta horitzontal i “baixem” el valor $g(x_0)$ a l'eix x per obtenir x_1 . Fem el mateix amb x_1 , i així successivament.

De la figura 3.2 se segueix que el “cas crític” és $|g'(x)| = 1$: tenim convergència per $|g'(x)| < 1$ i divergència per $|g'(x)| > 1$. De fet, tenim el següent teorema.

Teorema 3.2.2 (del punt fix) *Sigui $g : [a, b] \rightarrow \mathbb{R}$ contínua a $[a, b]$, derivable a (a, b) , tal que*

$$(i) \quad g([a, b]) \subset [a, b],$$

$$(ii) \quad |g'(x)| \leq \lambda \quad \forall x \in (a, b), \text{ per a certa } \lambda < 1.$$

Aleshores:

$$(a) \quad g \text{ té un punt fix i només un } \alpha : g(\alpha) = \alpha.$$

$$(b) \quad \text{Per a tot } x_0 \in [a, b], \text{ la successió } x_{n+1} = g(x_n) \text{ és convergent i té límit } \alpha.$$

$$(c) \quad \text{L'error comès en prendre } x_n \text{ com a valor aproximat de } \alpha \text{ satisfà les següents desigualtats:}$$

$$|x_n - \alpha| \leq \frac{\lambda^n}{1 - \lambda} |x_1 - x_0|, \quad (3.3)$$

$$|x_n - \alpha| \leq \frac{\lambda}{1 - \lambda} |x_n - x_{n-1}|. \quad (3.4)$$

Prova: La condició $g([a, b]) \subset [a, b]$ implica $g(a) \geq a$ i $g(b) \leq b$. Si $g(a) = a$ o $g(b) = b$ ja tenim el punt fix. Si no, $g(a) > a$ i $g(b) < b$ i considerem $f(x) = g(x) - x$. La funció f és contínua sobre $[a, b]$, $f(a) > 0$, $f(b) < 0$ i pel teorema de Bolzano tenim que existeix $\alpha \in (a, b)$ amb $f(\alpha) = 0$, o equivalentment, $g(\alpha) = \alpha$.

Per a veure la unicitat aplicarem el teorema del valor mig. Suposem $\alpha_1, \alpha_2 \in [a, b]$, $\alpha_1 < \alpha_2$ són punts fixos per g . Pel teorema del valor mig, existeix $\xi \in (\alpha_1, \alpha_2)$ tal que $g(\alpha_2) - g(\alpha_1) = g'(\xi)(\alpha_2 - \alpha_1)$. Aleshores,

$$|\alpha_2 - \alpha_1| = |g(\alpha_2) - g(\alpha_1)| = |g'(\xi)| |\alpha_2 - \alpha_1| \leq \lambda |\alpha_2 - \alpha_1| < |\alpha_2 - \alpha_1|,$$

fet que contradiu la hipòtesi que α_1, α_2 siguin punts fixos diferents.

Per veure (b) aplicarem novament el teorema del valor mig.

$$|x_n - \alpha| = |g(x_{n-1}) - g(\alpha)| = |g'(\xi_{n-1})| \cdot |x_{n-1} - \alpha| \leq \lambda \cdot |x_{n-1} - \alpha|.$$

Aplicant aquesta desigualtat inductivament:

$$0 \leq |x_n - \alpha| \leq \lambda |x_{n-1} - \alpha| \leq \lambda^2 |x_{n-2} - \alpha| \leq \dots \leq \lambda^n |x_0 - \alpha|.$$

Ara prenent límits i usant $0 < \lambda < 1$:

$$0 \leq \lim_{n \rightarrow \infty} |x_n - \alpha| \leq |x_0 - \alpha| \cdot \lim_{n \rightarrow \infty} \lambda^n = 0$$

és a dir, $\lim_{n \rightarrow \infty} |x_n - \alpha| = 0$.

Per a provar (3.3):

$$|x_{n+1} - x_n| = |g(x_n) - g(x_{n-1})| \leq \lambda |x_n - x_{n-1}| \leq \lambda^2 |x_{n-1} - x_{n-2}| \leq \dots \leq \lambda^n |x_1 - x_0|$$

Si $m > n$:

$$\begin{aligned} |x_m - x_n| &\leq |x_m - x_{m-1}| + |x_{m-1} - x_{m-2}| + \dots + |x_{n+1} - x_n| \leq \\ &\leq \lambda^{m-1} |x_1 - x_0| + \lambda^{m-2} |x_1 - x_0| + \dots + \lambda^n |x_1 - x_0| = \\ &= \lambda^n |x_1 - x_0| (1 + \lambda + \dots + \lambda^{m-n-1}) \leq \lambda^n |x_1 - x_0| \sum_{i=0}^{\infty} \lambda^i. \end{aligned}$$

Però $\sum_{i=0}^{\infty} \lambda^i$ és una sèrie geomètrica de raó $0 < \lambda < 1$. Aquesta sèrie convergeix a $\frac{1}{1-\lambda}$ i per tant

$$0 \leq |x_m - x_n| \leq \frac{\lambda^n}{1-\lambda} |x_1 - x_0|.$$

Prenent límits quan m tendeix a infinit obtenim

$$|\alpha - x_n| \leq \frac{\lambda^n}{1-\lambda} |x_1 - x_0|.$$

La prova de (3.4) és semblant: per $m \geq n$,

$$\begin{aligned} |x_m - x_n| &\leq |x_m - x_{m-1}| + \cdots + |x_{n+1} - x_n| \\ &\leq \lambda^{m-n} |x_n - x_{n-1}| + \cdots + \lambda |x_n - x_{n-1}| \\ &= (\lambda + \cdots + \lambda^{m-n}) |x_n - x_{n-1}| \\ &\leq \left(\sum_{i=1}^{\infty} \lambda^i \right) |x_n - x_{n-1}| \leq \frac{\lambda}{1-\lambda} |x_n - x_{n-1}|, \end{aligned}$$

i (3.4) s'obté fent $m \rightarrow \infty$. □

Observació 3.2.3 La fita (3.3) és una *estimació a priori*, donat que es pot avaluar abans de dur a terme el procediment iteratiu. Ens permet predir el nombre d'iterats necessaris per a assolir una tolerància prefixada.

Concretament, si volem trobar l'arrel α amb error absolut més petit que ε , imposarem

$$\frac{\lambda^n}{1-\lambda} |x_1 - x_0| \leq \varepsilon \iff n \geq \frac{\log(\varepsilon(1-\lambda)/|x_1 - x_0|)}{\log \lambda},$$

per al què és suficient prendre

$$n = \text{ceil}\left(\frac{\log(\varepsilon(1-\lambda)/|x_1 - x_0|)}{\log \lambda}\right), \quad (3.5)$$

on $\text{ceil}(x)$ és el mínim enter n verificant $n \geq x$, com en llenguatge C. ▽

Exemple 3.2.4 Suposem que volem resoldre $f(x) = 0$, on

$$f(x) = e^x - 2x - 1,$$

que podem reescriure com $x = g(x)$, on

$$g(x) = \ln(2x + 1).$$

Com que g és creixent, $g(1) = \ln 3 \approx 1.1 > 1$ i $g(2) = \ln 5 \approx 1.6 < 2$, tenim $g([1, 2]) \subset [1, 2]$. La derivada, $g'(x) = 2/(2x + 1)$ és decreixent, d'on, per a $x \in [1, 2]$

$$|g'(x)| \leq |g'(1)| = \frac{2}{3} =: \lambda < 1,$$

i per tant, segons el teorema 3.2.2, l'esquema iteratiu $x_{k+1} = g(x_k)$ és convergent a l'única arrel α . Si prenem $x_0 = 1.5$, llavors

$$x_1 = g(x_0) \approx 1.3863 \implies |x_1 - x_0| \approx 0.1137.$$

Si volem trobar α amb error absolut fitat per $\varepsilon := 10^{-5}$, d'acord amb (3.5) tindrem prou amb fer n iterats, essent

$$n = \text{ceil}\left(\frac{\log(10^{-5}(1 - 2/3)) - \log 0.1137}{\log 2/3}\right) \approx \text{ceil } 25.74 = 26.$$

Iterant amb 12 xifres decimals (només n'escrivim 6), surt

n	x_n	n	x_n	n	x_n	n	x_n
0	1.5	7	1.26052	14	1.25651	21	1.25643
1	1.38629	8	1.25876	15	1.25648	22	1.25643
2	1.32776	9	1.25775	16	1.25646	23	1.25643
3	1.29624	10	1.25718	17	1.25645	24	1.25643
4	1.27884	11	1.25686	18	1.25644	25	1.25643
5	1.26911	12	1.25668	19	1.25644	26	1.25643
6	1.26362	13	1.25657	20	1.25643		

Observem que la successió d'iterats s'estabilitza a l'iterat 20, abans del previst. ∇

Observació 3.2.5 La fita (3.4) es una *estimació a posteriori*, donat que, per poder avaluar-la, s'han de fer tots els iterats fins a x_n . Ens permet decidir quan ens hem d'aturar. Notem que, si $\lambda > 0.5$, llavors $|x_n - x_{n-1}| < \varepsilon$ (el criteri que fem servir al mètode de Newton) no és un criteri d'aturada acurat, donat que $\lambda/(1 - \lambda) > 1$ i no podem assegurar que $|x_n - \alpha| < \varepsilon$. ∇

Exemple 3.2.6 A l'exemple 3.2.4 observàvem que els iterats s'estabilitzaven abans del previst, i semblava que l'iterat 20 ja satisfia la tolerància demanada.

Anem a fer servir l'estimació a posteriori (3.4) amb una petita trampa: substituïm λ per $|g'(x_n)|$. Això està justificat pel fet que, per cada n , podríem aplicar el teorema del punt fix amb l'interval

$$I_n := [\alpha - |x_n - \alpha|, \alpha + |x_n - \alpha|].$$

Anomenem $\lambda_n := \sup_{x \in I_n} |g'(x)|$ el valor de λ que hauríem d'emprar al teorema del punt fix. Com que $\lim_{n \rightarrow \infty} \lambda_n = |g'(\alpha)| = \lim_{n \rightarrow \infty} |g'(x_n)|$, podem considerar $\lambda_n \approx |g'(x_n)|$. Així, a cada iterat calculem també la següent aproximació de la fita a posteriori:

$$f_n := \frac{|g'(x_n)|}{1 - |g'(x_n)|} |x_n - x_{n-1}|.$$

Els resultats són

n	x_n	f_n	n	x_n	f_n	n	x_n	f_n
0	1.5	— — —	9	1.25775	1.32×10^{-3}	18	1.25644	8.30×10^{-6}
1	1.38629	1.28×10^{-1}	10	1.25718	7.52×10^{-4}	19	1.25644	4.73×10^{-6}
2	1.32776	7.07×10^{-2}	11	1.25686	4.28×10^{-4}	20	1.25643	2.69×10^{-6}
3	1.29624	3.96×10^{-2}	12	1.25668	2.44×10^{-4}	21	1.25643	1.53×10^{-6}
4	1.27884	2.23×10^{-2}	13	1.25657	1.39×10^{-4}	22	1.25643	8.72×10^{-7}
5	1.26911	1.27×10^{-2}	14	1.25651	7.90×10^{-5}	23	1.25643	4.97×10^{-7}
6	1.26362	7.18×10^{-3}	15	1.25648	4.50×10^{-5}	24	1.25643	2.83×10^{-7}
7	1.26052	4.08×10^{-3}	16	1.25646	2.56×10^{-5}	25	1.25643	1.61×10^{-7}
8	1.25876	2.32×10^{-3}	17	1.25645	1.46×10^{-5}	26	1.25643	9.17×10^{-8}

D'acord amb això, amb la tolerància $\varepsilon = 10^{-5}$ triada abans, ens podríem aturar a l'iterat 18. ∇

Exemple 3.2.7 Suposem que volem trobar un zero de

$$f(x) = x - \sqrt{1 + x^2} + 0.05$$

i considerem la iteració simple $x_{k+1} = g(x_k)$ amb

$$g(x) = \sqrt{1 + x^2} - 0.05.$$

La solució de $x = g(x)$ es pot trobar exactament, i és

$$\alpha = 9.975.$$

Suposem que prèviament hem localitzat l'arrel entre 9 i 10, i prenem com a λ una fita de $|g'(x)|$ a $[9, 10]$, i.e.,

$$\lambda := |g'(10)| \approx 0.9950372.$$

Suposem que volem trobar l'arrel amb tolerància $\varepsilon = 10^{-5}$, i iterem a partir de $x_0 = 9$:

n	x_{n-1}	x_n	$ x_n - x_{n-1} $	$\frac{\lambda}{1-\lambda} x_n - x_{n-1} $	$ x_n - \alpha $
1	9.0000 000	9.0053 851	0.0053 851	1.0797 135	0.9699 149
2	9.0053 851	9.0107 374	0.0053 522	1.0731 136	0.9642 626
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
300	9.7726 149	9.7736 450	0.0010 301	0.2013 550	0.2013 550
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
500	9.9015 003	9.9018 696	0.0003 693	0.0731 304	0.0731 304
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
758	9.9548 754	9.9549 760	0.0001 006	0.0201 648	0.0200 240
759	9.9549 760	9.9550 761	0.0001 001	0.0200 639	0.0199 239
760	9.9550 761	9.9551 756	0.0000 996	0.0198 244	0.0198 244
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1500	9.9745 084	9.9745 108	0.0000 025	0.0004 916	0.0004 892
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1817	9.9748 992	9.9748 997	0.0000 005	0.0001 008	0.0001 003
1818	9.9748 997	9.9749 002	0.0000 005	0.0001 002	0.0000 998
1819	9.9749 002	9.9749 007	0.0000 005	0.0000 997	0.0000 993

Si féssim servir $|x_n - x_{n-1}| < \varepsilon$ com a criteri d'aturada, ens aturariem a l'iterat 760, per al qual la diferència entre x_n i l'arrel (darrera columna) és més gran que 0.01. En canvi, si fem servir $\frac{\lambda}{1-\lambda}|x_n - x_{n-1}| < \varepsilon$ com a criteri d'aturada, ens aturem a l'iterat 1819 i tenim la precisió que volíem. ∇

Observació 3.2.8 Si, a més de les hipòtesis del teorema anterior, $g'(x) \neq 0$, $\forall x \in (a, b)$ aleshores la successió $x_n \rightarrow \alpha$ però $x_n \neq \alpha$ per a tota n , llevat del cas $x_0 = \alpha$.

Ho provem per reducció a l'absurd: si k és el primer subíndex tal que $x_k = \alpha$, aleshores $x_{k-1} \neq \alpha = x_k$, $g(x_k) = x_k$ i

$$0 = |g(x_k) - g(x_{k-1})| = |g'(\xi_k)| \cdot |x_k - x_{k-1}|$$

ens porta a una contradicció. Així, malgrat que x_n està tan a prop de α com volguem, mai arribarà a ser igual a α . ∇

3.3 Ordre de convergència

Es tracta de quantificar la velocitat de convergència: per a una successió que convergeix, com de ràpid va cap al límit?

Desenvoluparem la teoria en una variable. Finalitzarem la secció amb comentaris sobre com generalitza la teoria a diverses variables.

Definició 3.3.1 Sigui $\{x_n\}_n \subset \mathbb{R}$ successió amb límit α que satisfà $x_n \neq \alpha \forall n$.

(a) Direm que la convergència és **d'ordre almenys p** (almenys lineal per $p = 1$, almenys quadràtica per $p = 2$, etc.) si $\exists K > 0, \exists n_0 \in \mathbb{N} : \forall n \geq n_0$

$$|x_{n+1} - \alpha| \leq K|x_n - \alpha|^p.$$

En el cas lineal ($p = 1$) demanem, a més, $K < 1$.

(b) Si $\exists C > 0$ tal que

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = C \quad (\neq \infty),$$

amb $C < 1$ si $p = 1$ direm que la successió té **ordre de convergència (exactament) p** . La quantitat C s'anomena **constant asimptòtica de l'error**.

Observació 3.3.2 Que $x_n \xrightarrow{n \rightarrow \infty} \alpha$ amb ordre almenys p implica que $|x_{n+1} - \alpha| = O(|x_n - \alpha|^p)$, però amb la restricció addicional per la constant K en el cas $p = 1$. ∇

Exercici: Demostreu que si $x_n \xrightarrow{n \rightarrow \infty} \alpha$ amb ordre p , llavors $x_n \xrightarrow{n \rightarrow \infty} \alpha$ amb ordre almenys p . Doneu un exemple de que el recíproc no és cert.

Exemple 3.3.3 Calculem l'ordre de convergència d'algunes successions:

(a) La successió $x_n := (1/2)^n$ convergeix a 0. Té ordre 1:

$$\lim_{n \rightarrow \infty} \frac{|(1/2)^{n+1} - 0|}{|(1/2)^n - 0|} = \lim_{n \rightarrow \infty} \frac{1}{2} = \frac{1}{2},$$

i per tant la constant asimptòtica de l'error és $1/2$.

(b) La successió $x_n := \frac{1}{2}(\frac{1}{3})^{2^n}$ convergeix a zero. Té ordre 2:

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - 0|}{|x_n - 0|^2} = \lim_{n \rightarrow \infty} \frac{\frac{1}{2}(\frac{1}{3})^{2^{n+1}}}{(\frac{1}{2}(\frac{1}{3})^{2^n})^2} = \lim_{n \rightarrow \infty} 2 \frac{(\frac{1}{3})^{2^{n+1}}}{(\frac{1}{3})^{2^{n+1}}} = 2,$$

i, per tant, la constant asimptòtica de l'error és 2.



L'exemple anterior suggereix com generar sistemàticament una successió amb qualsevol ordre de convergència. Però no és aquest el nostre objectiu, sinó calcular la velocitat de convergència de mètodes iteratius.

Definició 3.3.4 *Sigui $g : [a, b] \rightarrow \mathbb{R}$, $\alpha = g(\alpha)$ punt fix. Direm que el mètode iteratiu*

$$\begin{aligned} &x_0 \text{ aproximació inicial} \\ &\forall n = 0, 1, 2, \dots \\ &\quad x_{n+1} = g(x_n) \end{aligned}$$

té ordre (almenys, exactament) p si existeix $\varepsilon > 0$ tal que $\forall x_0 \in (\alpha - \varepsilon, \alpha + \varepsilon)$, $x_0 \neq \alpha$, es compleix que $x_{n+1} = g(x_n) \xrightarrow{n} \alpha$ amb ordre de convergència (almenys, exactament) p .

Dedicarem la resta d'aquesta secció al càlcul de l'ordre de diversos mètodes. L'eina fonamental serà la proposició 3.3.6. Per a poder-la demostrar necessitem abans un lema tècnic. El segon apartat d'aquest lema es pot considerar una demostració simplificada del teorema de Newton-Kantorovich.

Lema 3.3.5 *Per a $\delta > 0$, suposem $g : (\alpha - \delta, \alpha + \delta) \rightarrow \mathbb{R}$, i que existeix $K > 0$ tal que*

$$|g(x) - \alpha| \leq K|x - \alpha|^p$$

si $|x - \alpha| < \delta$, amb $K < 1$ si $p < 1$. Aleshores

(a) *Si $p = 1$, g dóna lloc a un mètode d'ordre almenys 1, i $x_{n+1} = g(x_n)$ convergeix per a $|x_0 - \alpha| < \delta$. A més, tenim l'estimació*

$$|x_n - \alpha| \leq K^n |x_0 - \alpha|.$$

(b) *Si $p > 1$, g dóna lloc a un mètode d'ordre almenys p , i $x_{n+1} = g(x_n)$ convergeix per a $|x_0 - \alpha| < \min(K^{-1/(p-1)}, \delta)$. A més, tenim l'estimació*

$$|x_n - \alpha| \leq K^{\frac{-1}{p-1}} (K^{\frac{1}{p-1}} |x_0 - \alpha|)^{p^n}. \quad (3.6)$$

Prova:

(a) Donat $x \in (\alpha - \delta, \alpha + \delta)$, tenim

$$|g(x) - \alpha| \leq K|x - \alpha| \leq |x - \alpha|,$$

d'on $g(x) \in (\alpha - \delta, \alpha + \delta)$. Per tant, si $x_0 \in (\alpha - \delta, \alpha + \delta)$ i definim $x_{n+1} = g(x_n)$, tenim $\{x_n\}_n \subseteq (\alpha - \delta, \alpha + \delta)$ i

$$|x_n - \alpha| \leq K|x_{n-1} - \alpha| \leq K^2|x_{n-2} - \alpha| \leq \dots \leq K^n|x_0 - \alpha| \xrightarrow{n \rightarrow \infty} 0$$

(donat que $K < 1$), d'on $x_n \rightarrow \alpha$ amb ordre almenys 1 (per la primera desigualtat).

(b) Si $|x - \alpha| < K^{-1/(p-1)} \iff |x - \alpha|^{p-1} < K^{-1}$,

$$|g(x) - \alpha| \leq K|x - \alpha|^p = (K|x - \alpha|^{p-1})|x - \alpha| < |x - \alpha|,$$

d'on, si x_0 és t.q. $|x_0 - \alpha| < \min(\delta, K^{-1/(p-1)})$ i $x_{n+1} = g(x_n)$, per inducció tindrem $|x_n - \alpha| < \min(\delta, K^{-1/(p-1)}) \forall n$. En particular, $\{x_n\}_n \subset (\alpha - \delta, \alpha + \delta)$. A més,

$$\begin{aligned} |x_n - \alpha| &\leq K|x_{n-1} - \alpha|^p \\ &\leq K(K|x_{n-2} - \alpha|^p)^p = K^{1+p}|x_{n-2} - \alpha|^{p^2} \\ &\leq K^{1+p}(K|x_{n-3} - \alpha|^p)^{p^2} = K^{1+p+p^2}|x_{n-3} - \alpha|^{p^3} \\ &\leq \dots \leq K^{1+p+\dots+p^{n-1}}|x_0 - \alpha|^{p^n} = K^{\frac{p^n-1}{p-1}}|x_0 - \alpha|^{p^n} \\ &= K^{\frac{-1}{p-1}}(K^{\frac{1}{p-1}}|x_0 - \alpha|)^{p^n} \xrightarrow{n \rightarrow \infty} 0. \end{aligned}$$

d'on $x_n \xrightarrow{n} \alpha$ amb ordre almenys p , donat que $|x_0 - \alpha| < K^{\frac{-1}{p-1}}$.

□

Proposició 3.3.6 *Segui $g : [a, b] \rightarrow \mathbb{R}$ de classe C^p i suposem que $\alpha \in (a, b)$ és t.q. $g(\alpha) = \alpha$.*

(a) *Si es compleix¹*

$$g'(\alpha) = g''(\alpha) = \dots = g^{(p-1)}(\alpha) = 0$$

amb $|g'(\alpha)| < 1$ si $p = 1$, llavors la funció d'iteració g dona lloc a un mètode iteratiu d'ordre almenys p per trobar α .

(b) *Si $g^{(p)}(\alpha) \neq 0$, amb $|g^{(p)}(\alpha)| < 1$ per $p = 1$, llavors l'ordre del mètode iteratiu donat per g és exactament p , amb constant asimptòtica de l'error*

$$C = \frac{|g^{(p)}(\alpha)|}{p!}.$$

Prova:

(a) Per Taylor,

$$\begin{aligned} g(x) &= g(\alpha) + g'(\alpha)(x - \alpha) + \dots + \frac{g^{(p-1)}(\alpha)}{(p-1)!}(x - \alpha)^{p-1} + \frac{g^{(p)}(\xi_x)}{p!}(x - \alpha)^p \\ &= \alpha + \frac{g^{(p)}(\xi_x)}{p!}(x - \alpha)^p. \end{aligned}$$

Aleshores:

- Si $p = 1$, escollim K t.q. $|g'(\alpha)| < K < 1$ i, per continuïtat de g' , existeix $\delta > 0$ tal que $|g'(\xi)| \leq K \forall \xi \in (\alpha - \delta, \alpha + \delta)$.
- Si $p > 1$, prenem δ tal que $(\alpha - \delta, \alpha + \delta) \subset [a, b]$ i $K = \max_{|\xi - \alpha| \leq \delta} \frac{|g^{(p)}(\xi)|}{p!}$ (existeix per continuïtat de $|g^{(p)}|/p!$).

¹Per a $p = 1$ s'entén que no estem demanant cap condició sobre les derivades, perquè la derivada $p - 1$ no existeix.

Per la proposició anterior, tenim que g dona lloc a un mètode d'ordre almenys p .

- (b) Com que ara necessitem $x_n \neq \alpha \forall n$, tornem a triar δ . Sigui $\varepsilon > 0$, $\varepsilon < |g^{(p)}(\alpha)|/p!$ i, si $p = 1$, demanarem també $\varepsilon < 1 - |g^{(p)}(\alpha)|/p!$. Sigui δ la donada per la continuïtat de $|g^{(p)}|/p!$ a α . Aleshores, si $|x - \alpha| < \delta$,

$$0 < \frac{|g^{(p)}(\alpha)|}{p!} - \varepsilon \leq \frac{|g^{(p)}(x)|}{p!} \leq \frac{|g^{(p)}(\alpha)|}{p!} + \varepsilon,$$

o, definint $\mu := \frac{|g^{(p)}(\alpha)|}{p!} - \varepsilon$, $K := \frac{|g^{(p)}(\alpha)|}{p!} + \varepsilon$,

$$0 < \mu \leq \frac{|g^{(p)}(x)|}{p!} \leq K,$$

i notem que $K < 1$ si $p = 1$. D'aquí, tenim

$$|x - \alpha| < \delta \implies |g(x) - \alpha| \leq K|x - \alpha|^p,$$

amb $K < 1$ si $p = 1$. Tornem a tenir pel lema 3.3.5 que g dona lloc a un mètode d'ordre almenys p , però amb més condicions.

Ara triem $x_0 \neq \alpha$ tal que $|x_0 - \alpha| < \delta$, si $p = 1$, o bé $|x_0 - \alpha| < \min(K^{-1/(p-1)}, \delta)$, si $p > 1$. Pel lema 3.3.5 tenim que, $x_{n+1} = g(x_n) \xrightarrow{n} \alpha$ amb ordre almenys p . Com que $|x_{n+1} - \alpha| = |g(x_n) - \alpha| \geq \mu|x_n - \alpha| \forall n$ i $x_0 \neq \alpha$, per inducció tindrem $x_n \neq \alpha \forall n$. Finalment

$$\frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = \frac{|g^{(p)}(\xi_{x_n})|}{p!} \xrightarrow{n \rightarrow \infty} \frac{|g^{(p)}(\alpha)|}{p!} =: C,$$

donat que $\xi_{x_n} \in \langle x_n, \alpha \rangle \Rightarrow \xi_{x_n} \xrightarrow{n} \alpha$. Per tant hem vist $x_n \xrightarrow{n} \alpha$ amb ordre exactament p i constant asimptòtica de l'error C .

□

Corol·lari 3.3.7 (Ordre de convergència del mètode de Newton) *Sigui $f : [a, b] \rightarrow \mathbb{R}$ de classe C^3 , $f(\alpha) = 0$, $f'(\alpha) \neq 0$ (i.e., α zero simple de f), $f''(\alpha) \neq 0$. Aleshores el mètode de Newton per trobar α com a zero de $f(x) = 0$ és quadràtic, amb constant asimptòtica de l'error*

$$C = \frac{1}{2} \left| \frac{f''(\alpha)}{f'(\alpha)} \right| \quad (3.7)$$

Prova: El mètode de Newton es pot considerar com una iteració de punt fix amb funció d'iteració

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

d'on

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

d'on $g'(\alpha) = 0$ i per tant el mètode de Newton és almenys quadràtic. Per comprovar que és quadràtic (amb les nostres hipòtesis), trobem g'' :

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

d'on $g''(\alpha) = f''(\alpha)/f'(\alpha) \neq 0$, i, per tant, per la proposició 3.3.6, l'ordre és 2 amb la constant asimptòtica de l'error de (3.7). \square

Observació 3.3.8

- (1) Si, en les hipòtesis del corol·lari anterior, canviem $f''(\alpha) \neq 0$ per $f''(\alpha) = 0$, llavors la convergència del mètode de Newton és almenys cúbica (exemple: $f(x) = \sin x$).
- (2) Si $f'(\alpha) = 0$, $f''(\alpha) \neq 0$ (i.e., α arrel doble de $f(x) = 0$), i $f'(x) \neq 0$ a un entorn de α , llavors, si tenim convergència, és lineal:

$$\begin{aligned} x_{n+1} - \alpha &= x_n - \frac{f(x_n)}{f'(x_n)} - \alpha = \frac{(x_n - \alpha)f'(x_n) - f(x_n)}{f'(x_n)} \\ &= \frac{(x_n - \alpha)f''(\xi_n)(x_n - \alpha) - \frac{1}{2}f''(\eta_n)(x_n - \alpha)^2}{f''(\xi_n)(x_n - \alpha)} \\ &= \frac{f''(\xi_n) - \frac{1}{2}f''(\eta_n)}{f''(\xi_n)}(x_n - \alpha) \end{aligned}$$

on $\xi_n, \eta_n \in \langle x_n, \alpha \rangle$ són punts intermedis que provenen de l'aplicació de la fórmula de Taylor amb resta. Un argument semblant al de la demostració de l'observació 3.2.8 prova que $x_n \neq \alpha \forall n$. Un cop tenim això,

$$\frac{|x_{n+1} - \alpha|}{|x_n - \alpha|} = \left| \frac{f''(\xi_n) - \frac{1}{2}f''(\eta_n)}{f''(\xi_n)} \right| \xrightarrow{n} \frac{1}{2} \neq 0,$$

donat que, pel principi del sandwich, $\xi_n, \eta_n \xrightarrow{n} \alpha$ i f'' és contínua.

De fet, a partir d'això es pot provar convergència, considerant $g(x) = x - f(x)/f'(x)$.²

De fet, es pot arreglar per recuperar la convergència quadràtica. En les hipòtesis esmentades, podem considerar el **mètode de Newton modificat per arrels dobles**:

x_0 aproximació inicial

$\forall n = 0, 1, 2, \dots$

$$x_{n+1} = x_n - 2 \frac{f(x_n)}{f'(x_n)}$$

²Els mateixos càlculs provenen $|g(x) - \alpha|/|x - \alpha| \xrightarrow{x \rightarrow \alpha} \frac{1}{2}$, i, per definició de límit, es veu $|g(x) - \alpha| \leq K|x - \alpha|$ amb $K < 1$ a un entorn de α . Llavors, per la proposició 3.3.5, g dona lloc a un mètode almenys lineal. La proposició 3.3.5 no ens assegura que l'ordre és exactament 1. Això ho hem vist ara.

Vegem-ho:

$$\begin{aligned}
 x_{n+1} - \alpha &= \frac{(x_n - \alpha)f'(x_n) - 2f(x_n)}{f'(x_n)} \\
 &= \frac{1}{f''(\mu_n)(x_n - \alpha)} \left((x_n - \alpha)(f''(\alpha)(x_n - \alpha) + \frac{1}{2}f'''(\xi_n)(x_n - \alpha)^2) \right. \\
 &\quad \left. - 2\left(\frac{1}{2}f''(\alpha)(x_n - \alpha)^2 + \frac{1}{6}f'''(\eta_n)(x_n - \alpha)^3\right) \right) \\
 &= \frac{\frac{1}{2}f'''(\xi_n)(x_n - \alpha)^3 - \frac{1}{3}f'''(\eta_n)(x_n - \alpha)^3}{f''(\mu_n)(x_n - \alpha)} \\
 &= \frac{\frac{1}{2}f'''(\xi_n) - \frac{1}{3}f'''(\eta_n)}{f''(\mu_n)}(x_n - \alpha)^2.
 \end{aligned}$$

Per tant (novament es pot veure $x_n \neq \alpha \forall n$),

$$\frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^2} = \left| \frac{\frac{1}{2}f'''(\xi_n) - \frac{1}{3}f'''(\eta_n)}{f''(\mu_n)} \right| \xrightarrow{n} \frac{1}{6} \left| \frac{f'''(\alpha)}{f''(\alpha)} \right|,$$

d'on el mètode de Newton modificat serà almenys quadràtic.

Estrictament, el càlcul que hem fet només prova convergència quadràtica cas que $x_n \xrightarrow{n} \alpha$. Com abans, càlculs anàlegs considerant $g(x) = x - f(x)/f'(x)$ permeten provar convergència mitjançant la proposició 3.3.5.

▽

Podria semblar que estimacions com (3.6) són purament acadèmiques, donat que a la pràctica no coneixem $|x_0 - \alpha|$. Hi ha situacions interessants, però, on es pot fitar.

Exemple 3.3.9 Imaginem-nos que estem implementant l'aritmètica de punt flotant d'un processador, que ja tenim fetes la suma, el producte i la divisió i volem fer l'arrel quadrada. Podem emprar el mètode de Newton. Anem a usar l'estimació (3.6) per a provar la convergència del mètode de Newton, a partir d'una aproximació inicial adequada, en un nombre d'iterats "raonable" i independent del número del que busquem l'arrel quadrada.

Suposem que volem trobar \sqrt{c} , on $c = mb^q > 0$, amb m mantissa normalitzada ($m \geq b^{-1}$). Això és equivalent a trobar el zero positiu de $f(x) = x^2 - c$. Els iterats del mètode de Newton són

$$x_{n+1} = g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)} = \frac{1}{2} \left(x_n + \frac{c}{x_n} \right). \quad (3.8)$$

Una aproximació inicial de l'arrel que podem trobar sense fer cap operació és

$$x_0 = b^{q/2}$$

donat que, si q és parell la mantissa és trivial, i si no, només cal tenir emmagatzemada la mantissa de $b^{1/2}$. Aleshores

$$\alpha = \sqrt{c} = \sqrt{m} b^{q/2} < b^{q/2} = x_0,$$

d'on $x_0 > \alpha$. A més,

$$\forall x > \alpha, \quad g(x) - \alpha = \frac{g''(\xi_x)}{2}(x - \alpha) > 0 \implies g(x) > \alpha,$$

donat que $g''(\xi) = c/\xi^3 > 0$ si $\xi > 0$. A més, com que $g''(\xi)$ és decreixent a $[\alpha, \infty)$, a l'estimació (3.6) podem prendre

$$K = \max_{\xi \in [\alpha, \infty)} \frac{g''(\xi)}{2} = \frac{c}{2\alpha^3} = \frac{1}{2\sqrt{c}}.$$

Ara, d'acord amb (3.6),

$$|x_n - \alpha| \leq K^{-1}(K|x_0 - \alpha|)^{2^n}, \quad (3.9)$$

però, per tenir convergència, cal que $K|x_0 - \alpha| < 1$:

$$\begin{aligned} K|x_0 - \alpha| &= \frac{1}{2\sqrt{c}}(x_0 - \sqrt{c}) = \frac{1}{2m^{1/2}b^{q/2}}(b^{q/2} - m^{1/2}b^{q/2}) \\ &= \frac{1 - m^{1/2}}{2m^{1/2}} \leq \frac{1 - b^{-1/2}}{2b^{-1/2}} = \frac{\sqrt{b} - 1}{2}. \end{aligned}$$

Per $b = 2$ tenim $(\sqrt{b} - 1)/2 \approx 0.207$ i per tant tindrem convergència. Per a $b = 10$ tindriem $(\sqrt{b} - 1)/2 > 1$, i per tant no podríem assegurar convergència (necessitariem una condició inicial x_0 més acurada).

Suposem que estem implementant l'aritmètica IEEE doble, per la qual $b = 2$. Substituint el valor de $K|x_0 - \alpha|$ en (3.9), obtenim

$$\frac{|x_n - \sqrt{c}|}{\sqrt{c}} \leq 2 \left(\frac{\sqrt{b} - 1}{2} \right)^{2^n},$$

d'on, si volem $|e_r(x_n, \sqrt{c})| < \varepsilon$, on $\varepsilon > 10^{-16}$ és l'èpsilon-màquina de l'aritmètica IEEE doble, només cal demanar

$$n > \frac{1}{\log_{10} 2} \log_{10} \frac{-16 - \log_{10} 2}{\log_{10} ((\sqrt{b} - 1)/2)} \approx 4.5.$$

Per tant, per a qualsevol número màquina $c = mb^q$, prenent $x_0 = b^{q/2}$ i fent 5 iterats de Newton obtenim $fl(\sqrt{c})$ amb error relatiu més petit que l'èpsilon-màquina.³ Cada iterat de Newton (3.8) només consta d'una divisió i una suma (no comptem la divisió per dos perquè, en una aritmètica binària, només cal decrementar l'exponent en una unitat). ∇

³Estem suposant que no cometem errors a les operacions. Aquesta suposició és realista si les operacions a nivell intern es fan amb una aritmètica més precisa que la IEEE doble, cosa que passa a la pràctica totalitat dels processadors actuals.

Capítol 4

Interpolació Polinòmica

Suposem que tenim una família de funcions $\mathbb{R} \rightarrow \mathbb{R}$ que depenen de $n + 1$ paràmetres,

$$\Phi(x; a_0, \dots, a_n)$$

on a_0, \dots, a_n són els paràmetres. Suposem també que tenim $n + 1$ punts $\{(x_i, y_i)\}_{i=0}^n$. El **problema d'interpolació** per Φ , $\{(x_i, y_i)\}_{i=0}^n$ consisteix a determinar a_0, \dots, a_n perquè

$$\Phi(x_i; a_0, \dots, a_n) = y_i, \quad i = 0, 1, \dots, n.$$

Els $\{(x_i, y_i)\}_{i=0}^n$ es diuen **punts de suport**, els $\{x_i\}_{i=0}^n$ **abscisses de suport** i els $\{y_i\}_{i=0}^n$ **ordenades de suport**.

Es diu que el problema d'interpolació és **lineal** si Φ depèn linealment dels paràmetres,

$$\Phi(x; a_0, \dots, a_n) = a_0\Phi_0(x) + a_1\Phi_1(x) + \dots + a_n\Phi_n(x),$$

per exemple, la **interpolació polinomial**,

$$\Phi(x; a_0, \dots, a_n) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n,$$

o la **interpolació trigonomètrica**,

$$\Phi(x; a_0, \dots, a_n) = a_0 + a_1e^{ix} + a_2e^{i2x} + \dots + a_ne^{inx}.$$

Exemples d'interpolació no lineal són la **interpolació racional**,

$$\Phi(x; a_0, \dots, a_n, b_0, \dots, b_m) = \frac{a_0 + a_1x + \dots + a_nx^n}{b_0 + b_1x + \dots + b_mx^m},$$

i la **interpolació exponencial**,

$$\Phi(x; a_0, \dots, a_n, \lambda_0, \dots, \lambda_n) = a_0e^{\lambda_0x} + a_1e^{\lambda_1x} + \dots + a_ne^{\lambda_nx}.$$

Abans de l'aparició de les calculadores, la principal aplicació de la interpolació era obtenir, a partir de taules, valors no tabulats (de funcions trigonomètriques, logarítmiques, distribucions de probabilitat, ...). En l'actualitat, la interpolació continua essent una eina fonamental per l'aproximació de funcions, que és el fonament d'altres mètodes numèrics, alguns dels quals els veurem durant el curs.

En aquest capítol només tractarem la interpolació polinomial.

4.1 Interpolació de Lagrange

4.1.1 Existència i unicitat

Denotem per Π_n el conjunt de polinomis de grau $\leq n$. El problema d'interpolació de Lagrange consisteix a, donats $\{(x_i, y_i)\}_{i=0}^n \subset \mathbb{R}^2$, amb $x_i \neq x_j$ per $i \neq j$, trobar $P \in \Pi_n$ tal que

$$P(x_i) = y_i, \quad i = 0, 1, \dots, n. \quad (4.1)$$

Proposició 4.1.1 *El problema d'interpolació de Lagrange (4.1) té solució única.*

Prova: Per a veure'n l'existència de solució, anem a construir-la explícitament. Definim

$$L_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}.$$

Els polinomis $L_0, \dots, L_n \in \Pi_n$ s'anomenen *polinomis bàsics de Lagrange*, i verifiquen

$$L_i(x_j) = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{si } i \neq j. \end{cases}$$

Aleshores

$$P(x) = y_0 L_0(x) + y_1 L_1(x) + \dots + y_n L_n(x) \in \Pi_n$$

verifica $P(x_i) = y_i$, per $i = 0, 1, \dots, n$.

Per veure'n la unicitat, suposem que $P, Q \in \Pi_n$ verifiquen $P(x_i) = Q(x_i) = y_i$ per $i = 0, 1, \dots, n$. Aleshores $P - Q \in \Pi_n$ té $n + 1$ zeros diferents, x_0, \dots, x_n . Pel teorema fonamental de l'àlgebra, necessàriament ha de ser el polinomi zero, i per tant $P = Q$. \square

Els polinomis bàsics de Lagrange donen un mètode de càlcul del polinomi interpolador. Tot i que és interessant des del punt de vista teòric, altres mètodes que veurem més endavant són més adequats per càlculs efectius. No obstant això, la fórmula de Lagrange pot ser interessant en situacions en les quals s'han de resoldre molts problemes d'interpolació per les mateixes abscisses de suport.

Prova: (una altra) Anem a imposar la condició d'interpolació (4.1) a un polinomi genèric

$$P(x) = a_0 + a_1 x + \dots + a_n x^n.$$

Obtenim

$$\left. \begin{array}{cccccc} a_0 & + & a_1 x_0 & + & a_2 x_0^2 & + \dots + & a_n x_0^n & = & y_0 \\ a_0 & + & a_1 x_1 & + & a_2 x_1^2 & + \dots + & a_n x_1^n & = & y_1 \\ & & & & & & \vdots & & \\ a_0 & + & a_1 x_n & + & a_2 x_n^2 & + \dots + & a_n x_n^n & = & y_n \end{array} \right\},$$

que és un sistema lineal $(n + 1) \times (n + 1)$. Tindrà solució única si i només si el determinant de la seva matriu de coeficients és no nul. El determinant esmentat és

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{vmatrix} = \prod_{i>j} (x_i - x_j) = \prod_{j=0}^{n-1} \prod_{i=j+1}^n (x_i - x_j),$$

que es diu determinant de Vandermonde. \square

4.1.2 Algorisme de Neville

Considerem els punts de suport $\{(x_i, f_i)\}_{i=0}^n$ i, per a $k \leq n$, $\{i_0, i_1, \dots, i_k\} \subset \{0, 1, 2, \dots, n\}$, denotem per

$$P_{i_0, i_1, \dots, i_k} \in \Pi_k \quad (4.2)$$

el polinomi que resol el problema d'interpolació

$$P_{i_0, i_1, \dots, i_k}(x_{i_j}) = f_{i_j}, \quad j = 0, 1, \dots, k. \quad (4.3)$$

Exemple 4.1.2 Suposem que treballem amb punts de suport $\{(0, 1), (1, 3), (3, 2)\}$. Amb les notacions anteriors $n = 2$, $(x_0, f_0) = (0, 1)$, $(x_1, f_1) = (1, 3)$, $(x_2, f_2) = (3, 2)$. Aleshores

- $P_0(x)$ és el polinomi de grau zero que satisfà $P_0(x_0) = f_0$, i.e. $P_0(0) = 1$. Amb les notacions anteriors, $P_0 = P_{i_0}$, amb $i_0 = 0$.
- $P_{0,2}(x)$ és el polinomi de grau ≤ 1 que satisfà $P_{0,2}(x_0) = f_0$ i $P_{0,2}(x_2) = f_2$, és a dir, $P_{0,2}(0) = 1$, $P_{0,2}(3) = 2$. Amb les notacions anteriors, $P_{0,2} = P_{i_0, i_2}$ amb $i_0 = 0$, $i_2 = 2$.
- $P_{0,1,2}(x)$ és el polinomi de grau ≤ 2 que satisfà $P_{0,1,2}(0) = 1$, $P_{0,1,2}(1) = 3$, $P_{0,1,2}(3) = 2$.

▽

Vegem la recurrència que dóna lloc a l'algorisme de Neville en forma de proposició.

Proposició 4.1.3 *Se satisfà*

$$P_i(x) = f_i, \quad (4.4)$$

$$P_{i_0, i_1, \dots, i_k}(x) = \frac{(x - x_{i_0})P_{i_1, i_2, \dots, i_k}(x) - (x - x_{i_k})P_{i_0, i_1, \dots, i_{k-1}}(x)}{x_{i_k} - x_{i_0}}. \quad (4.5)$$

Prova: La igualtat (4.4) és trivial. Per veure (4.5), denotem per R la part dreta de la igualtat i provem que satisfà (4.3) i (4.2), llavors, per unicitat de la interpolació de Lagrange, tindrem $R = P_{i_0, i_1, \dots, i_k}$. És clar que

$$\begin{aligned} R(x_{i_0}) &= P_{i_0, \dots, i_{k-1}}(x_{i_0}) = f_{i_0}, \\ R(x_{i_k}) &= P_{i_1, \dots, i_k}(x_{i_k}) = f_{i_k}, \end{aligned}$$

degut a les propietats d'interpolació de $P_{i_0, \dots, i_{k-1}}$ i P_{i_1, \dots, i_k} . Finalment

$$R(x_{i_j}) = \frac{(x_{i_j} - x_{i_0})f_{i_j} - (x_{i_j} - x_{i_k})f_{i_k}}{x_{i_k} - x_{i_0}} = f_{i_j}, \quad j = 1, 2, \dots, k-1,$$

com volíem veure. □

Donat x , suposem que volem avaluar $P(x)$, on P és el polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f_i)\}_{i=0}^n$. Segons les notacions anteriors, $P(x) = P_{0,1,\dots,n}(x)$.

L'organització dels càlculs de la recurrència (4.4), (4.5) en la següent taula

	$k = 0$	$k = 1$	$k = 2$
x_0	$f_0 = P_0(x)$		
		$P_{0,1}(x)$	
x_1	$f_1 = P_1(x)$		$P_{0,1,2}(x)$
		$P_{1,2}(x)$	\ddots
x_2	$f_2 = P_2(x)$		
\vdots	\vdots		

que s'omple columna a columna d'esquerra a dreta, es coneix com a algorisme de Neville.

Exemple 4.1.4 Considerem els punts de suport $\{(x_i, f_i)\}_{i=0}^2$ donats per la següent taula:

i	0	1	2
x_i	0	1	3
f_i	1	3	2

Volem avaluar el polinomi interpolador a $x = 2$, és a dir, volem trobar $P_{0,1,2}(2)$. La taula de l'algorisme de Neville és

	$k = 0$	$k = 1$	$k = 2$
$x_0 = 0$	$f_0 = P_0(2) = 1$		
		$P_{0,1}(2) = 5$	
$x_1 = 1$	$f_1 = P_1(2) = 3$		$P_{0,1,2}(2) = 10/3$
		$P_{1,2}(2) = 5/2$	
$x_2 = 3$	$f_2 = P_2(2) = 2$		

on

$$\begin{aligned}
 P_{01}(2) &= \frac{(x - x_0)P_1(x) - (x - x_1)P_0(x)}{x_1 - x_0} \Bigg|_{x=2} = \frac{(2 - 0)3 - (2 - 1)1}{1 - 0} = 5 \\
 P_{12}(2) &= \frac{(x - x_1)P_2(x) - (x - x_2)P_1(x)}{x_2 - x_1} \Bigg|_{x=2} = \frac{(2 - 1)2 - (2 - 3)3}{3 - 1} = \frac{5}{2} \\
 P_{012}(2) &= \frac{(x - x_0)P_{12}(x) - (x - x_2)P_{01}(x)}{x_2 - x_0} \Bigg|_{x=2} = \frac{(2 - 0)\frac{5}{2} - (2 - 3)5}{3 - 0} = \frac{10}{3}
 \end{aligned}$$

▽

4.1.3 Mètode de les diferències dividides de Newton

El mètode de Neville és adequat per a avaluar el polinomi interpolador a un punt una única vegada, però és ineficient per a avaluar-lo diverses vegades. Per a això, el que voldrem és tenir l'expressió general del polinomi interpolador i avaluar-la tantes vegades com calgui. D'això ens ocuparem en aquesta secció.

Suposem que volem trobar $P_n \in \Pi_n$ polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f_i)\}_{i=0}^n$. Si l'escrivim en la forma

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \quad (4.6)$$

Aquesta formulació permet avaluar el polinomi interpolador mitjançant l'esquema de Horner, per exemple

$$P_3(x) = a_0 + (x - x_0) \left[a_1 + (x - x_1) [a_2 + (x - x_2) a_3] \right]. \quad (4.7)$$

Si imposem la condició d'interpolació a (4.6), obtenim

$$\begin{cases} f_0 = P_n(x_0) = a_0 \\ f_1 = P_n(x_1) = a_0 + a_1(x_1 - x_0) \\ \vdots \\ f_n = P_n(x_n) = a_0 + a_1(x_n - x_0) + \dots + a_n(x_n - x_0) \dots (x_n - x_{n-1}) \end{cases}$$

Aquest és un sistema d'equacions lineal $(n+1) \times (n+1)$ triangular que es pot resoldre recurrentment: $a_0 = f_0$, $a_1 = (f_1 - a_0)/(x_1 - x_0)$, $a_2 = (f_2 - a_0 - a_1(x_2 - x_0))/(x_2 - x_0)(x_2 - x_1)$, ... En particular, té solució única, i aquesta és una altra demostració d'existència i unicitat del polinomi interpolador.

Recuperem les notacions de l'algorisme de Neville: per a $\{i_0, \dots, i_k\} \subset \{0, \dots, n\}$ denotem per P_{i_0, \dots, i_k} el polinomi de grau $\leq k$ que verifica $P_{i_0, \dots, i_k}(x_{i_j}) = f_{i_j}$ per $j = 0, 1, \dots, k$.

Observem que $P_{i_0, \dots, i_k} - P_{i_0, \dots, i_{k-1}}$ és un polinomi de grau k que té zeros $x_{i_0}, \dots, x_{i_{k-1}}$. Per tant, existeix una única constant, que anomenarem f_{i_0, \dots, i_k} , tal que

$$P_{i_0, \dots, i_k}(x) - P_{i_0, \dots, i_{k-1}}(x) = f_{i_0, \dots, i_k}(x - x_{i_0}) \dots (x - x_{i_{k-1}}).$$

Aleshores,

$$\begin{aligned} P_{i_0, \dots, i_k}(x) &= P_{i_0, \dots, i_{k-1}}(x) + f_{i_0, \dots, i_k}(x - x_{i_0}) \dots (x - x_{i_{k-1}}) \\ &= P_{i_0, \dots, i_{k-2}}(x) + f_{i_0, \dots, i_{k-1}}(x - x_{i_0}) \dots (x - x_{i_{k-2}}) \\ &\quad \vdots \\ &\quad + f_{i_0, \dots, i_k}(x - x_{i_0}) \dots (x - x_{i_{k-1}}) \\ &= P_{i_0}(x) + f_{i_0, i_1}(x - x_{i_0}) + \\ &\quad + f_{i_0, i_1, i_2}(x - x_{i_0})(x - x_{i_1}) + \dots + \\ &\quad + f_{i_0, i_1, \dots, i_k}(x - x_{i_0})(x - x_{i_1}) \dots (x - x_{i_{k-1}}) \end{aligned} \quad (4.8)$$

Observem que f_{i_0, i_1, \dots, i_k} és el coeficient del terme de grau més gran (o sigui, del terme de grau k) de P_{i_0, i_1, \dots, i_k} . Per la proposició 4.1.3, sabem que

$$P_{i_0, \dots, i_k}(x) = \frac{(x - x_{i_0})P_{i_1, \dots, i_k}(x) - (x - x_{i_k})P_{i_0, \dots, i_{k-1}}(x)}{x_{i_k} - x_{i_0}},$$

d'on

$$f_{i_0, \dots, i_k} = \frac{f_{i_1, \dots, i_k} - f_{i_0, \dots, i_{k-1}}}{x_{i_k} - x_{i_0}}. \quad (4.9)$$

La quantitat f_{i_0, \dots, i_k} l'anomenarem *diferència dividida d'ordre k* aplicada als arguments x_{i_0}, \dots, x_{i_k} .
Més formalment:

Definició 4.1.5 Sigui $f : \mathbb{R} \rightarrow \mathbb{R}$ funció, $\{x_i\}_{i=0}^k \subset \mathbb{R}$ diferents dos a dos. Definim la diferència dividida d'ordre k de f aplicada als arguments x_0, \dots, x_k , que denotarem per $f[x_0, \dots, x_k]$, com el coeficient de grau més gran del polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f(x_i))\}_{i=0}^k$.

Proposició 4.1.6 (recurrència de les diferències dividides) Per $x \in \mathbb{R}$, se satisfà

$$f[x] = f(x),$$

i, per $n \in \mathbb{N}$, $x_0, \dots, x_n \subset \mathbb{R}$ diferents dos a dos,

$$f[x_0, \dots, x_n] = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0}. \quad (4.10)$$

Prova: La primer igualtat és òbvia. La segona se segueix de (4.9) prenent $k = n, i_0 = 0, i_1 = 1, \dots, i_n = n$. \square

Proposició 4.1.7 El polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f(x_i))\}_{i=0}^n$ és

$$\begin{aligned} P_n(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &\quad + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}). \end{aligned}$$

Prova: Tenint en compte que $P_{i_0}(x) = f_{i_0}$, se segueix de (4.8) prenent $k = n, i_0 = 0, i_1 = 1, \dots, i_n = n$. \square

El càlcul del polinomi interpolador de Lagrange mitjançant les diferències dividides de Newton se sol organitzar en una taula semblant a la de l'algorisme de Neville, tal com s'il·lustra en el següent

Exemple 4.1.8 Volem trobar el polinomi interpolador de Lagrange amb els punts de suport de l'exemple 4.1.4: $\{(x_i, f_i)\}_{i=0}^2$ donats per

i	0	1	2
x_i	0	1	3
f_i	1	3	2

La taula de diferències dividides és

$x_0 = 0$	$f[x_0] = 1$		
		$f[x_0, x_1] = \frac{3-1}{1-0} = 2$	
$x_1 = 1$	$f[x_1] = 3$		$f[x_0, x_1, x_2] = \frac{-1/2-2}{3-0} = -\frac{5}{6}$
		$f[x_1, x_2] = \frac{2-3}{3-1} = -\frac{1}{2}$	
$x_2 = 3$	$f[x_2] = 2$		

que s'omple per columnes, d'esquerra a dreta (observeu com es tradueix la recurrència (4.10) sobre la taula). El polinomi interpolador és, d'acord amb la proposició 4.1.7,

$$\begin{aligned} P(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &= 1 + 2x - \frac{5}{6}x(x - 1). \end{aligned}$$

Noteu que els coeficients que entren al polinomi interpolador són els de la part de dalt de la taula de diferències dividides.

Anem a trobar el polinomi interpolador per Neville:

$x_0 = 0$	$P_0(x) = f_0 = 1$	
		$P_{0,1}(x) = 2x + 1$
$x_1 = 1$	$P_1(x) = f_1 = 3$	$P_{0,1,2} = -\frac{5}{6}x^2 + \frac{17}{6}x + 1$
		$P_{1,2}(x) = -\frac{x}{2} + \frac{7}{2}$
$x_2 = 3$	$P_2(x) = f_2 = 2$	

on els càlculs intermitjos per completar la taula són

$$\begin{aligned}
 P_{0,1}(x) &= \frac{(x - x_0)P_1(x) - (x - x_1)P_0(x)}{x_1 - x_0} = \frac{(x - 0)3 - (x - 1)1}{1} \\
 &= 2x + 1 \\
 P_{1,2}(x) &= \frac{(x - x_1)P_2(x) - (x - x_2)P_1(x)}{x_2 - x_1} = -\frac{x}{2} + \frac{7}{2} \\
 P_{0,1,2}(x) &= \frac{(x - x_0)P_{1,2}(x) - (x - x_2)P_{0,1}(x)}{x_2 - x_0} = -\frac{5}{6}x^2 + \frac{17}{6}x + 1
 \end{aligned}$$

El polinomi interpolador és, per tant,

$$P(x) = -\frac{5}{6}x^2 + \frac{17}{6}x + 1,$$

que coincideix amb el que havíem calculat amb diferències dividides (expandiu-lo). A més,

$$\begin{aligned}
 f[x_0, x_1] &= 2 = \text{coeficient de grau màxim de } P_{0,1}, \\
 f[x_1, x_2] &= -1/2 = \text{coeficient de grau màxim de } P_{1,2}, \\
 f[x_0, x_1, x_2] &= -5/6 = \text{coeficient de grau màxim de } P_{0,1,2},
 \end{aligned}$$

d'acord amb la definició 4.1.5.

Tot i que aquí ho hem fet amb finalitats il·lustratives, noteu que no és eficient trobar l'expressió general del interpolador pel mètode de Neville, donat que requereix manipulació simbòlica. El mètode de Neville és adequat per a avaluar el polinomi interpolador a un punt una única vegada. ∇

4.1.4 Comparació entre els diversos mètodes

Hem vist tres mètodes de càlcul del polinomi interpolador de Lagrange: usar polinomis bàsics de Lagrange (demostració del teorema 4.1.1), l'algorisme de de Neville (secció 4.1.2) i les diferències dividides de Newton (secció 4.1.3).

Si hem d'avaluar el polinomi interpolador de Lagrange $P \in \Pi_n$ corresponent a determinats punts de suport $\{(x_i, f_i)\}_{i=0}^n$ una única vegada (per exemple, només necessitem $P(3)$), la millor opció és usar l'algorisme de Neville, donat que ens estalvia construir explícitament $P(x)$.

Si hem d'avaluar P a diversos punts, llavors és més eficient construir explícitament $P(x)$ mitjançant les diferències dividides de Newton, i llavors avaluar-lo tants cops com calgui (aprofitant l'esquema de Horner (4.7)).

L'expansió en polinomis bàsics de Lagrange és d'interès principalment teòric. No obstant, hi ha una situació pràctica en la que pot ser interessant, i és quan hem de treballar amb diversos polinomis interpoladors amb les mateixes abscisses de suport. Concretament, si definim

$$L_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)},$$

llavors el polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f_i)\}_{i=0}^n$ és

$$P(x) = f_0 L_0(x) + f_1 L_1(x) + \dots + f_n L_n(x).$$

Si ara necessitem el polinomi interpolador amb punts de suport $\{(x_i, g_i)\}_{i=0}^n$ (o sigui, només canviem les ordenades de suport), el podem escriure com

$$Q(x) = g_0 L_0(x) + g_1 L_1(x) + \dots + g_n L_n(x),$$

on les $L_i(x)$ són les mateixes d'abans.

4.1.5 Error a la interpolació de Lagrange

Teorema 4.1.9 *Sigui $f : [a, b] \rightarrow \mathbb{R}$ de classe C^{n+1} , $\{x_i\}_{i=0}^n \subset [a, b]$ diferents dos a dos, $P \in \Pi_n$ polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f(x_i))\}_{i=0}^n$. Aleshores, $\forall x \in [a, b]$,*

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \omega_n(x),$$

on $\omega(x) = (x - x_0)(x - x_1) \dots (x - x_n)$ i $\xi(x) \in \langle x_0, \dots, x_n, x \rangle$ és una funció desconeguda de x .

Prova: Fixem $x \in [a, b]$. Si $x = x_i$ per $i \in \{0, \dots, n\}$, aleshores podem escollir $\xi(x)$. Suposem $x \notin \{x_0, \dots, x_n\}$.

Considerem la funció

$$F(z) := f(z) - P(z) - \omega(z)S(x),$$

on

$$S(x) = \frac{f(x) - P(x)}{\omega(x)}$$

(recordeu que x està fixada, i noteu que $S(x)$ és constant respecte de z).

És immediat comprovar que F té $n+2$ zeros (respecte de z): x_0, \dots, x_n, x . Ordenem-los i canviem-los de nom: siguin

$$\{\xi_0^{(0)}, \xi_1^{(0)}, \dots, \xi_{n+1}^{(0)}\} := \{x_0, \dots, x_n, x\}$$

amb $\xi_0^{(0)} < \xi_1^{(0)} < \dots < \xi_n^{(0)} < \xi_{n+1}^{(0)}$. Apliquem el teorema de Rolle $n+1$ cops:

$$\begin{array}{lll} & F \text{ té } n+2 \text{ zeros: } & \{\xi_i^{(0)}\}_{i=0}^{n+1} \\ \xRightarrow{\text{Rolle}} & F' \text{ té } n+1 \text{ zeros: } & \{\xi_i^{(1)}\}_{i=1}^{n+1}, \quad \xi_i^{(1)} \in (\xi_{i-1}^{(0)}, \xi_i^{(0)}) \\ \xRightarrow{\text{Rolle}} & F'' \text{ té } n \text{ zeros: } & \{\xi_i^{(2)}\}_{i=2}^{n+1}, \quad \xi_i^{(2)} \in (\xi_{i-1}^{(1)}, \xi_i^{(1)}) \\ & \vdots & \\ \xRightarrow{\text{Rolle}} & F^{(n)} \text{ té } 2 \text{ zeros: } & \{\xi_i^{(n)}\}_{i=n}^{n+1}, \quad \xi_i^{(n)} \in (\xi_{i-1}^{(n-1)}, \xi_i^{(n-1)}) \\ \xRightarrow{\text{Rolle}} & F^{(n+1)} \text{ té } 1 \text{ zero: } & \{\xi_i^{(n+1)}\}_{i=n+1}^{n+1}, \quad \xi_i^{(n+1)} \in (\xi_{i-1}^{(n)}, \xi_i^{(n)}) \end{array}$$

Lavors tenim

$$0 = F^{(n+1)}(\xi_{n+1}^{(n+1)}) = f^{(n+1)}(\xi_{n+1}^{(n+1)}) - (n+1)!S(x),$$

donat que P és un polinomi de grau n i la seva derivada $n+1$ és zero, i com que $\omega(z)$ és un polinomi mònic de grau $n+1$, la seva derivada $n+1$ és $(n+1)!$. Aïllant $S(x)$, tenim

$$\frac{f^{(n+1)}(\xi_{n+1}^{(n+1)})}{(n+1)!} = S(x) = \frac{f(x) - P(x)}{\omega(x)},$$

i només cal multiplicar per $\omega(x)$. Noteu que $\xi_{n+1}^{(n+1)}$ depèn de x , donat que x surt a la llista de punts amb la que comencem a aplicar Rolle. \square

Observació 4.1.10

- 1.— No tenim cap control sobre $\xi(x)$, de manera que la fitació més fina de l'error d'interpolació que podem fer és

$$|f(x) - P(x)| \leq \frac{\max_{y \in [a,b]} |f^{(n+1)}(y)|}{(n+1)!} |\omega(x)|$$

(on $[a, b]$ el prenem tan petit com puguem).

- 2.— Per a abscisses equiespaiades, la funció $\omega(x)$ és simètrica respecte de $(x_0 + x_n)/2$ per n senar, antisimètrica respecte del mateix punt per n parell. La podeu dibuixar amb Octave mitjançant

```
x=linspace(0,1,333);
y=prod(repmat(x,n+1,1)-diag(0:n)*ones(n+1,333)/n);
plot(x,y)
```

(proveu diversos valors de n , i vegeu la figura 4.1).

- 3.— Observeu que $|\omega(x)|$ és més petita al mig que als extrems. Per això convé que les abscisses d'interpolació estiguin distribuïdes el més simètricament possible respecte del punt on interpolem.
- 4.— La funció $|\omega(x)|$ creix molt ràpidament fora de $\langle x_0, \dots, x_n \rangle$. Això ens diu que extrapolar (avaluar el polinomi interpolador fora de l'interval generat per les abscisses d'interpolació, i.e., fora de $\langle x_0, \dots, x_n \rangle$) és perillós i s'ha de fer amb compte.

∇

Observació 4.1.11 Una altra expressió per l'error d'interpolació és

$$f(x) - P(x) = f[x_0, \dots, x_n, x] \omega(x), \quad (4.11)$$

donat que

$$\begin{aligned} f(x) &= \underbrace{f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1})}_{=P(x)} \\ &\quad + f[x_0, \dots, x_n, x](x - x_0) \dots (x - x_{n-1})(x - x_n) \end{aligned}$$

i aquesta igualtat se segueix del fet que la part de la dreta és l'expansió en diferències dividides del polinomi interpolador als punts $\{(x_0, f(x_0)), \dots, (x_n, f(x_n)), (x, f(x))\}$. ∇

D'acord amb aquesta observació, tenim el següent

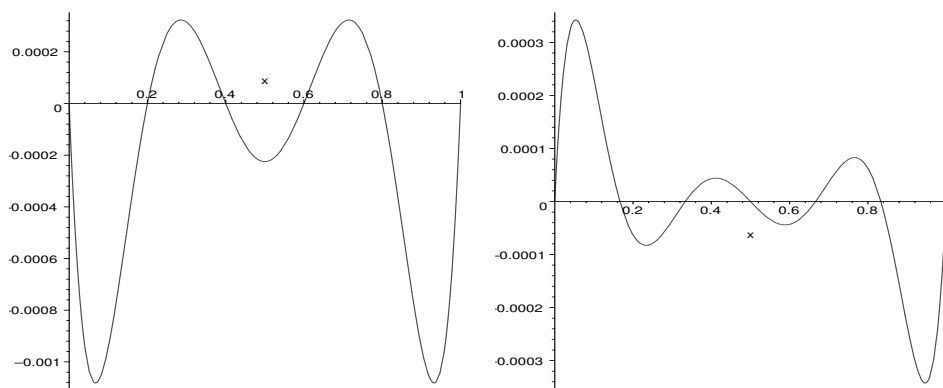


Figura 4.1: Representació gràfica de $\omega(x) = (x - x_0) \dots (x - x_n)$ amb abscisses equiespaiades a $[0, 1]$ ($x_i = i/n$, $i = 0, 1, \dots, n$), per $n = 5$ (esquerra) i $n = 6$ (dreta).

Corol·lari 4.1.12 Sigui $f : [a, b] \rightarrow \mathbb{R}$ de classe C^{n+1} , $\{x_0, \dots, x_n\} \subset [a, b]$ diferents dos a dos, $x \in [a, b]$, $x \notin \{x_0, \dots, x_n\}$. Aleshores existeix $\xi_x \in \langle x_0, \dots, x_n, x \rangle$ tal que

$$f[x_0, \dots, x_n, x] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

Prova: Sigui $P(x)$ el polinomi interpolador als punts $\{(x_0, f(x_0)), \dots, (x_n, f(x_n))\}$. L'error d'interpolació és, d'acord amb el teorema 4.1.9 i l'observació 4.1.11,

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega(x) = f[x_0, \dots, x_n, x] \omega(x),$$

i, com que $x \notin \{x_i\}_{i=0}^n$, podem dividir per $\omega(x)$ i tenim la igualtat que volíem. \square

Observació 4.1.13 Noteu que el que hem provat equival a dir que per $\{x_i\}_{i=0}^n$ diferents dos a dos existeix $\xi \in \langle x_0, \dots, x_n \rangle$ tal que

$$f[x_0, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

Noteu també que l'ordre de la derivada és el nombre d'arguments menys un. ∇

Proposició 4.1.14 (simetria de les diferències dividides) Siguin $\{x_i\}_{i=0}^n \subset [a, b]$ diferents dos a dos, $f : \mathbb{R} \rightarrow \mathbb{R}$ funció, $\sigma \in S_n$ permutació. Aleshores

$$f[x_0, \dots, x_n] = f[x_{\sigma(0)}, \dots, x_{\sigma(n)}].$$

Prova: En les notacions de la proposició de la recurrència de l'algorisme de Neville, $f[x_0, \dots, x_n]$ és el coeficient de grau màxim de $P_{0, \dots, n}$, i $f[x_{\sigma(0)}, \dots, x_{\sigma(n)}]$ és el coeficient de grau màxim de $P_{\sigma(0), \dots, \sigma(n)}$. Però $P_{0, \dots, n}$ i $P_{\sigma(0), \dots, \sigma(n)}$ resolen el mateix problema d'interpolació de Lagrange, i, per unicatat de solució del problema d'interpolació de Lagrange, han de ser iguals. \square

4.2 Interpolació d'Hermite

4.2.1 Existència i unicitat

En aquesta secció volem trobar polinomis dels quals, a més de prescriure'n valors, volem prescriure valors de les seves derivades.

Concretament, donats

- $\{x_i\}_{i=0}^m \subset \mathbb{R}$, verificant $x_0 < \dots < x_m$ (abscisses d'interpolació),
- $\{n_i\}_{i=0}^m \subset \mathbb{N}$ (ordres màxims de derivació decrementats en una unitat),
- $\{y_i^{(k)}\}_{i=0,1,\dots,m,k=0,1,\dots,n_i-1} \subset \mathbb{R}$ (ordenades d'interpolació),

cerquem $P_n \in \Pi_n$ amb

$$n_0 + n_1 + \dots + n_m = n + 1$$

tal que

$$P_n^{(k)}(x_i) = y_i^{(k)}, \quad i = 0, 1, \dots, m, \quad k = 0, 1, \dots, n_i - 1.$$

Exemple 4.2.1 Suposem que cerquem $P_5 \in \Pi_5$ verificant

$$\begin{aligned} P_5(1) &= 1, & P_5(2) &= 4, & P_5(3) &= 6, \\ P_5'(1) &= 2, & P_5'(2) &= 5, \\ P_5''(1) &= 3. \end{aligned}$$

Llavors, en les notacions anteriors,

$$\begin{aligned} x_0 &= 1, & x_1 &= 2, & x_2 &= 3, \\ n_0 &= 3, & n_1 &= 2, & n_2 &= 1, \\ y_0^{(0)} &= 1, & y_1^{(0)} &= 4, & y_2^{(0)} &= 6, \\ y_0^{(1)} &= 2, & y_1^{(1)} &= 5, \\ y_0^{(2)} &= 3. \end{aligned}$$

▽

Proposició 4.2.2 *El problema d'interpolació d'Hermite té solució única.*

Prova: Vegem primer la unicitat. Siguin $P_1, P_2 \in \Pi_n$ solució del problema d'interpolació d'Hermite. Aleshores

$$P_1^{(k)}(x_i) = P_2^{(k)}(x_i) = y_i^{(k)}, \quad i = 0, 1, \dots, m, \quad k = 0, 1, \dots, n_i - 1.$$

Aleshores el polinomi $Q := P_1 - P_2$ té $n_0 + n_1 + \dots + n_m = n + 1$ zeros comptant multiplicitats. Com que és de grau $\leq n$, pel teorema fonamental de l'Àlgebra necessàriament $Q = 0$.

Per veure'n l'existència, considerem un polinomi general $P(x) = c_0 + c_1x + \dots + c_nx^n$ de grau $\leq n$. $P(x)$ serà solució del problema d'interpolació d'Hermite si

$$P^{(k)}(x_i) = y_i^{(k)}, \quad i = 0, 1, \dots, m, \quad k = 0, 1, \dots, n_i - 1.$$

Aquest és un sistema d'equacions lineals en c_0, \dots, c_n , de $n_0 + n_1 + \dots + n_m = n + 1$ equacions i $n + 1$ incògnites. Com que sabem que té solució única, la seva matriu de coeficients necessàriament és no singular, i per tant té solució. □

4.2.2 Càlcul del polinomi interpolador d'Hermite

Es pot fer mitjançant les diferències dividides de Newton, de manera completament anàloga a la interpolació de Lagrange.

Denotem mitjançant

$$\begin{aligned} & (\xi_0, f_0), \dots, (\xi_{n_0-1}, f_{n_0-1}), \\ & (\xi_{n_0}, f_{n_0}), \dots, (\xi_{n_0+n_1-1}, f_{n_0+n_1-1}), \\ & \vdots \\ & (\xi_{n_0+\dots+n_{m-1}}, f_{n_0+\dots+n_{m-1}}), \dots, (\xi_{n_0+\dots+n_m-1}, f_{n_0+\dots+n_m-1}) \end{aligned}$$

els parells

$$\begin{aligned} & (x_0, y_0^{(0)}), \dots, (x_0, y_0^{(n_0-1)}), \\ & (x_1, y_1^{(0)}), \dots, (x_1, y_1^{(n_1-1)}), \\ & \vdots \\ & (x_m, y_m^{(0)}), \dots, (x_m, y_m^{(n_m-1)}), \end{aligned}$$

(recordeu que $n_0 + \dots + n_m - 1 = n$). Com que $x_0 < x_1 < \dots < x_m$, tenim $\xi_0 \leq \xi_1 \leq \dots \leq \xi_n$. El polinomi interpolador d'Hermite és

$$\begin{aligned} P_n(x) = & f[\xi_0] + f[\xi_0, \xi_1](x - \xi_0) + f[\xi_0, \xi_1, \xi_2](x - \xi_0)(x - \xi_1) \\ & + \dots + f[\xi_0, \dots, \xi_n](x - \xi_0)(x - \xi_1) \dots (x - \xi_{n-1}) \end{aligned}$$

Per tal que l'expressió anterior tingui sentit, hem de poder calcular diferències dividides amb arguments repetits. Observeu que la definició que hem adoptat de diferències dividides (definició 4.1.5) no té sentit si els arguments no són diferents dos a dos. Les proposicions 4.1.14, 4.1.6 i l'observació 4.1.13 ens permeten donar sentit a diferències dividides amb arguments repetits de la següent forma:¹

$$f[\xi_i, \dots, \xi_{i+j}] = \begin{cases} \frac{f[\xi_{i+1}, \dots, \xi_{i+j}] - f[\xi_i, \dots, \xi_{i+j-1}]}{\xi_{i+j} - \xi_i} & \text{si } \xi_i \neq \xi_{i+j}, \\ y_i^{(j)} / j!, & \text{si } \xi_i = \xi_{i+j}, \text{ per } l \text{ t.q. } x_l = \xi_i. \end{cases}$$

Noteu que si $\xi_i = \xi_{i+j}$, llavors $\xi_i = \xi_{i+1} = \xi_{i+2} = \dots = \xi_{i+j}$, i, per l'observació 4.1.13

$$f[\xi_i, \dots, \xi_i] = \frac{f^{(j)}(\xi_i)}{j!}.$$

Exemple 4.2.3 Continuem l'exemple 4.2.1: siguin $\{x_i\}_{i=0}^2$, $\{n_i\}_{i=0}^2$ i $\{y_i^{(j)}\}_{i=0,1,2, j=0,1,\dots,n_i-1}$ donats per les taules

i	0	1	2
x_i	1	2	3
n_i	3	2	1

 $y_i^{(j)} :$

$j \setminus i$	0	1	2
0	1	4	6
1	2	5	
2	3		

¹Podeu trobar a la pàgina 250 de l'edició de Dover de *Analysis of Numerical Methods* de Isaacson & Keller una expressió en termes d'integrals iterades per a les diferències dividides amb arguments possiblement repetits, que es pot considerar una definició intrínseca de les diferències dividides quan $f^{(n)}$ és contínua.

Amb les notacions anteriors,

$$\begin{aligned} \xi_0 = x_0, \quad \xi_1 = x_0, \quad \xi_2 = x_0, \quad \xi_3 = x_1, \quad \xi_4 = x_1, \quad \xi_5 = x_2, \\ f_0 = y_0^{(0)}, \quad f_1 = y_0^{(1)}, \quad f_2 = y_0^{(2)}, \quad f_3 = y_1^{(0)}, \quad f_4 = y_1^{(1)}, \quad f_5 = y_2^{(0)}. \end{aligned}$$

Els càlculs es poden organitzar així:

x_0	$f[x_0]$
	$f[x_0, x_0]$
x_0	$f[x_0] \quad f[x_0, x_0, x_0]$
	$f[x_0, x_0] \quad f[x_0, x_0, x_0, x_1]$
x_0	$f[x_0] \quad f[x_0, x_0, x_1] \quad f[x_0, x_0, x_0, x_1, x_1]$
	$f[x_0, x_1] \quad f[x_0, x_0, x_1, x_1] \quad f[x_0, x_0, x_0, x_1, x_1, x_2]$
x_1	$f[x_1] \quad f[x_0, x_1, x_1] \quad f[x_0, x_0, x_1, x_1, x_2]$
	$f[x_1, x_1] \quad f[x_0, x_1, x_1, x_2]$
x_1	$f[x_1] \quad f[x_1, x_1, x_2]$
	$f[x_1, x_2]$
x_2	$f[x_2]$

on

$$\begin{aligned} f[x_0] &= y_0^{(0)}, & f[x_1] &= y_1^{(0)}, & f[x_2] &= y_2^{(0)}, \\ f[x_0, x_0] &= y_0^{(1)}/1!, & f[x_1, x_1] &= y_1^{(1)}/1!, & f[x_0, x_0, x_0] &= y_0^{(2)}/2!, \end{aligned}$$

i els altres es calculen com a la proposició 4.1.6, per exemple,

$$\begin{aligned} f[x_0, x_0, x_1] &= \frac{f[x_0, x_1] - f[x_0, x_0]}{x_1 - x_0}, \\ f[x_0, x_0, x_0, x_1, x_1] &= \frac{f[x_0, x_0, x_1, x_1] - f[x_0, x_0, x_0, x_1]}{x_1 - x_0}. \end{aligned}$$

Concretament,

1	1					
		2				
1	1		3/2			
		2		-1/2		
1	1		1		3/2	
		3		1		-13/8
2	4		2		-7/4	
		5		-5/2		
2	4		-3			
		2				
3	6					

i, per tant, el polinomi interpolador d'Hermite buscat és

$$\begin{aligned} P_5(x) &= f[x_0] + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_0](x - x_0)^2 \\ &\quad + f[x_0, x_0, x_0, x_1](x - x_0)^3 + f[x_0, x_0, x_0, x_1, x_1](x - x_0)^3(x - x_1) \\ &\quad + f[x_0, x_0, x_0, x_1, x_1, x_2](x - x_0)^3(x - x_1)^2 \\ &= 1 + 2(x - 1) + \frac{3}{2}(x - 1)^2 - \frac{1}{2}(x - 1)^3 + \frac{3}{2}(x - 1)^3(x - 2) \\ &\quad - \frac{13}{8}(x - 1)^3(x - 2)^2. \end{aligned}$$

Comproveu que $P^{(j)}(x_i) = y_i^{(j)}$.

▽

4.2.3 Error a la interpolació d'hermite

Tenim un resultat anàleg al del cas de Lagrange.

Teorema 4.2.4 *Sigui $f : [a, b] \rightarrow \mathbb{R}$ de classe C^{n+1} , $\{x_0, \dots, x_m\} \subset [a, b]$, $x_0 < x_1 < \dots < x_m$, $\{n_0, \dots, n_m\} \subset \mathbb{N}$, $n_0 + \dots + n_m = n + 1$. Sigui P_n el polinomi interpolador d'Hermite corresponent, i.e.,*

$$P_n^{(k)}(x_i) = f^{(k)}(x_i), \quad i = 0, 1, \dots, m, \quad k = 0, 1, \dots, n_i - 1.$$

Aleshores, per tot $x \in [a, b]$ existeix $\xi_x \in \langle x_0, \dots, x_m, x \rangle$ tal que

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} (x - x_0)^{n_0} (x - x_1)^{n_1} \dots (x - x_m)^{n_m}.$$

Prova: Només fem un cas particular (en general és molt pesat). Suposem $m = 2$, $n_0 = 2$, $n_1 = 1$, $n_2 = 3$, d'on $n = 5$.

Fixem $x \in [a, b]$. Si $x = x_i$ per a algun i , ens serveix qualsevol ξ_x . Suposem $x \neq x_i$ per $i = 0, 1, \dots, m$ i definim

$$F(x) := f(x) - P_5(x) - (x - x_0)^2 (x - x_1) (x - x_2)^3 S(x),$$

on

$$S(x) = \frac{f(x) - P_5(x)}{(x - x_0)^2 (x - x_1) (x - x_2)^3},$$

(noteu que S no depen de z).

Per simplificar la notació, particularitzem encara més i suposem que $x \in [x_1, x_2]$. Aleshores, degut a les propietats de interpolació de P_5 , F té zeros x_0, x_1, x_2, x . Escrivim-los ordenats i repetits amb multiplicitats, i apliquem el teorema de Rolle sis vegades:

$$\begin{array}{llllllll} F \text{ s'anul·la a} & x_0 & x_0 & x_1 & x & x_2 & x_2 & x_2 \\ \Rightarrow F' \text{ s'anul·la a} & & \xi_2^{(1)} & \xi_3^{(1)} & \xi_4^{(1)} & & x_2 & x_2 \\ \Rightarrow F'' \text{ s'anul·la a} & & \xi_2^{(2)} & \xi_3^{(2)} & \xi_4^{(2)} & \xi_5^{(2)} & & x_2 \\ \Rightarrow F''' \text{ s'anul·la a} & & & \xi_3^{(3)} & \xi_4^{(3)} & \xi_5^{(3)} & \xi_6^{(3)} & \\ \Rightarrow F^{(4)} \text{ s'anul·la a} & & & & \xi_4^{(4)} & \xi_5^{(4)} & \xi_6^{(4)} & \\ \Rightarrow F^{(5)} \text{ s'anul·la a} & & & & & \xi_5^{(5)} & \xi_6^{(5)} & \\ \Rightarrow F^{(6)} \text{ s'anul·la a} & & & & & & \xi_6^{(6)} & \end{array}$$

on l'element de cada fila està inclòs a l'interval obert format per dos elements més propers de la fila anterior. Definim $\xi_x := \xi_6^{(6)} \in \langle x_0, x_1, x_2, x \rangle$, i tenim

$$0 = F^{(6)}(\xi_x) = f^{(6)}(\xi_x) - 6!S(x),$$

d'on

$$\frac{f(x) - P_5(x)}{(x - x_0)^2 (x - x_1) (x - x_2)^3} = S(x) = \frac{f^{(6)}(\xi_x)}{6!}$$

i només cal multiplicar per $(x - x_0)^2 (x - x_1) (x - x_2)^3$.

□

Observació 4.2.5

- 1.– El problema d'interpolació de Lagrange es recupera prenent $n_0 = n_1 = \dots = n_m = 1$.
- 2.– Per $m = 1$, el polinomi interpolador d'Hermite és el polinomi de Taylor, donat que $n_0 = m + 1$,

$$P_n(x) = f[x_0] + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_0](x - x_0)^2 + \dots + f[x_0, \dots, x_0]^{n+1}(x - x_0)^n,$$

i, per l'observació 4.1.13,

$$f[x_0, \dots, x_0]^{j+1} = \frac{f^{(j)}(x_0)}{j!}.$$

De fet, l'error d'interpolació d'Hermite és, en aquest cas, la resta de Lagrange del polinomi de Taylor:

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)^{n+1}.$$

▽

4.3 Interpolació per splines

Comencem aquesta secció fent paleses algunes limitacions de la interpolació de Lagrange, que fan necessari considerar altres tipus d'interpolació.

Suposem que volem aproximar $f : [a, b] \rightarrow \mathbb{R}$. Prenem $a = x_0 < x_1 < \dots < x_n = b$ i considerem $P_n(x)$ polinomi interpolador de Lagrange a $\{(x_i, f(x_i))\}_{i=0}^n$. Pot semblar que pujant el grau podem millorar arbitràriament l'aproximació, és a dir, $\forall x \in [a, b], P_n(x) \xrightarrow{n \rightarrow \infty} f(x)$. Això no és cert en general.

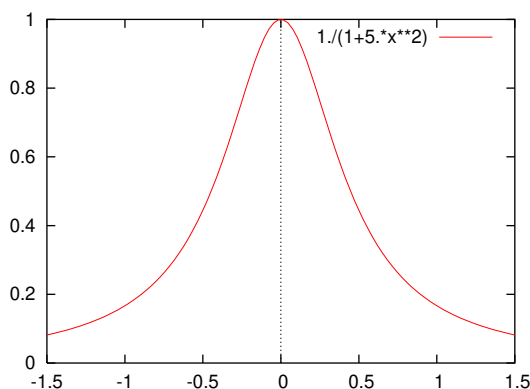


Figura 4.2: Gràfica de $f(x) = 1/(1 + (5x)^2)$.

Exemple 4.3.1 [Runge, 1901] Considerem la funció

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

que és de classe $C^\infty(\mathbb{R})$. Per $n \in \mathbb{N}$, prenem $\{x_i\}_{i=0}^n$ equiespaiats a $[-1, 1]$:

$$x_i = -1 + i \frac{2}{n}, \quad i = 0, \dots, n,$$

i denotem per P_n el polinomi interpolador de Lagrange amb punts de suport $\{(x_i, f(x_i))\}_{i=0}^n$.

Es pot demostrar analíticament que $P_n(x) \not\rightarrow f(x)$ per $n \rightarrow \infty$ en un entorn de $x = 1$ i $x = -1$. A la pràctica, s'observen grans oscil·lacions del polinomi interpolador al voltant de 1 i -1. ∇

Voldríem un procediment interpolador que

- Mantingués una certa regularitat globalment.
- Permetés millorar l'aproximació arbitràriament en augmentar el número de nodes.

Una manera natural d'aconseguir el segon punt seria usar no un polinomi interpolador de grau elevat, sinó diversos polinomis interpoladors de grau més petit. Fent això no podem garantir que la funció aproximadora sigui diferenciable a tot arreu, perquè és polinomial a trossos. Si volem tenir regularitat globalment, cal imposar que els polinomis empalmin de manera suau: voldrem que, als punts de contacte, les derivades dels polinomis coincideixin fins a un cert ordre. Això porta de manera natural a la noció de spline.

Definició 4.3.2 Donat un interval tancat $[a, b]$, direm que $\Delta := \{x_i\}_{i=0}^n$ n'és una partició si $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$. Els punts x_i s'anomenen nodes.

Definició 4.3.3 Un spline de grau p associat a la partició $\Delta := \{x_i\}_{i=0}^n$ d'un interval tancat $[a, b]$ és una funció $s : [a, b] \rightarrow \mathbb{R}$ de classe C^{p-1} verificant

$$s|_{[x_i, x_{i+1}]} \in \Pi_p, \quad \forall i = 0, \dots, n-1.$$

Denotarem per $S_p(\Delta)$ el conjunt d'splines de grau p associats a Δ .

A continuació desenvolupem la teoria necessària per al càlcul de splines cúbics.

Definició 4.3.4 Siguin $\{(x_i, y_i)\}_{i=0}^n \subset \mathbb{R}^2$ amb $x_0 < x_1 < \dots < x_n$. Anomenarem spline cúbic interpolador amb punts de suport $\{(x_i, y_i)\}_{i=0}^n \subset \mathbb{R}^2$ tot $s \in S_3(\{x_i\}_{i=0}^n)$ verificant

$$s(x_i) = y_i, \quad i = 0, \dots, n.$$

Exemple 4.3.5 Als punts de suport $\{(x_i, y_i)\}_{i=0}^2$ donats per la taula

i	0	1	2
x_i	0	1	2
y_i	1	0	3

correspon l'spline cúbic interpolador

$$s(x) = \begin{cases} s_0(x) = 1 - 2x + x^3, & \text{si } x \in [0, 1], \\ s_1(x) = 3 - 8x + 6x^2 - x^3, & \text{si } x \in [1, 2]. \end{cases}$$

Podeu comprovar que:

$$\begin{aligned} s_0(0) &= 1, & s_0(1) &= s_1(1) = 0, & s_1(2) &= 3, \\ s'_0(1) &= s'_1(1) = 1 \\ s''_0(1) &= s''_1(1) = 6 \end{aligned}$$

▽

Anem a determinar condicions d'existència i unicitat de splines cúbics interpoladors.

Fixem $\{(x_i, y_i)\}_{i=0}^n$ punts de suport amb $x_0 < x_1 < \dots < x_n$. Per a $s \in S_3(\{x_i\}_{i=0}^n)$ qualsevol, denotem

$$s_i := s|_{[x_i, x_{i+1}]} =: a_{i,0} + a_{i,1}x + a_{i,2}x^2 + a_{i,3}x^3.$$

Tenim un total de $4n$ coeficients a determinar. Comptem el nombre de condicions:

- **Regularitat:** per a que $s \in C^2([x_0, x_n])$, cal demanar

$$\left. \begin{aligned} s_{i-1}(x_i) &= s_i(x_i) \\ s'_{i-1}(x_i) &= s'_i(x_i) \\ s''_{i-1}(x_i) &= s''_i(x_i) \end{aligned} \right\} \quad i = 1, \dots, n-1,$$

que dóna un total de $3(n-1)$ condicions.

- **Interpolació:** cal

$$s(x_i) = y_i, \quad i = 0, \dots, n,$$

que són $n+1$ condicions.

En total tenim $3(n-1) + n+1 = 4n-2$ condicions. Per igualar el nombre de coeficients a determinar, ens calen 2 condicions. Se sol triar entre:

- *condicions d'extrem d'Hermite:* per $y'_0, y'_n \in \mathbb{R}$ donats, demanem

$$s'(x_0) = y'_0, \quad s'(x_n) = y'_n, \quad (4.12)$$

- *condicions d'extrem naturals:* demanem

$$s''(x_0) = s''(x_n) = 0, \quad (4.13)$$

- *condicions d'extrem periòdiques:* demanem

$$s'(x_0) = s'(x_n), \quad s''(x_0) = s''(x_n). \quad (4.14)$$

Noteu que, perquè això tingui sentit, **cal que** $y_0 = y_n$!!

Els corresponent splines es diuen *d'Hermite*, *naturals* o *periòdics*, respectivament.

Noteu que totes les condicions que hem demanat són lineals en els coeficients, i com que tenim tantes equacions com coeficients a determinar, l'existència de solució equival a la unicitat. Anem a trobar un algorisme de càlcul, del qual se'n seguirà l'existència de solució (i per tant la unicitat).

Definició 4.3.6 Per a $s \in S_3(\{x_i\}_{i=0}^n)$, definim els seus moments per

$$M_i := s''(x_i).$$

Anem a veure que, si s és spline interpolador d'Hermite, natural o periòdic, els seus moments el determinen completament. Recordem que havíem definit $s_i := s|_{[x_i, x_{i+1}]} \in \Pi_3$. Per tant, $s_i'' \in \Pi_1$, d'on

$$\begin{aligned} s_i''(x) &= M_i \frac{x - x_{i+1}}{x_i - x_{i+1}} + M_{i+1} \frac{x - x_i}{x_{i+1} - x_i} \\ &= M_i \frac{x_{i+1} - x}{h_{i+1}} + M_{i+1} \frac{x - x_i}{h_{i+1}}, \end{aligned}$$

on hem denotat

$$h_i = x_i - x_{i-1}, \quad i = 1, \dots, n.$$

(noteu que hem emprat polinomis bàsics de Lagrange). Integrant dos cops, tenim

$$\begin{aligned} s_i'(x) &= -M_i \frac{(x_{i+1} - x)^2}{2h_{i+1}} + M_{i+1} \frac{(x - x_i)^2}{2h_{i+1}} + C_i, \\ s_i(x) &= M_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + M_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + C_i(x - x_i) + \tilde{C}_i, \end{aligned} \quad (4.15)$$

on C_i, \tilde{C}_i són constants d'integració que es dedueixen de les condicions d'interpolació:

$$\begin{aligned} y_i = s_i(x_i) &\implies \tilde{C}_i = y_i - M_i \frac{h_{i+1}^2}{6}, \\ y_{i+1} = s_i(x_{i+1}) &\implies C_i = \frac{y_{i+1} - y_i}{h_{i+1}} - (M_{i+1} - M_i) \frac{h_{i+1}}{6}. \end{aligned} \quad (4.16)$$

Les expresions anteriors ja permetrien avaluar $s_i(x)$ per i arbitrària (donats els moments), però és més pràctic (per estabilitat numèrica i perquè permet usar l'esquema de Horner) usar per a cada s_i el polinomi de Taylor de grau 4,²

$$s_i = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3,$$

on

$$\begin{aligned} \alpha_i &= s_i(x_i) = y_i, \\ \beta_i &= s_i'(x_i) = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2M_i + M_{i+1}}{6} h_{i+1}, \\ \gamma_i &= \frac{1}{2} s_i''(x_i) = \frac{M_i}{2}, \\ \delta_i &= \frac{1}{6} s_i'''(x_i) = \frac{M_{i+1} - M_i}{6h_{i+1}}. \end{aligned}$$

(recordeu que $s_i''(x_i) = M_i$ per definició dels moments).

Anem a deduir equacions per als moments imposant continuïtat de la primera derivada: per $i = 1, \dots, n - 1$, volem

$$s'_{i-1}(x_i) = s'_i(x_i).$$

²que, per unicitat de l'interpolació, coincideix amb s_i .

Usant (4.15) i (4.16), obtenim

$$\begin{aligned} M_i \frac{h_i}{2} + \frac{y_i - y_{i-1}}{h_i} - (M_i - M_{i-1}) \frac{h_i}{6} \\ = -M_i \frac{h_{i+1}}{2} + \frac{y_{i+1} - y_i}{h_{i+1}} - (M_{i+1} - M_i) \frac{h_{i+1}}{6} \end{aligned} \quad (4.17)$$

d'on

$$M_{i-1} \frac{h_i}{6} + M_i \frac{h_i + h_{i+1}}{3} + M_{i+1} \frac{h_{i+1}}{6} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}$$

i, multiplicant per $6/(h_i + h_{i+1})$, obtenim

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad i = 1, \dots, n-1, \quad (4.18)$$

essent

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right)$$

Com que cal determinar M_0, \dots, M_n (total $n+1$ coeficients), ens calen 2 equacions més. Volem que tinguin la mateixa forma que (4.18).

Cas d'Hermite. Hem d'imposar

$$y'_0 = s'_0(x_0) = -M_0 \frac{h_1}{2} + \frac{y_1 - y_0}{h_1} - (M_1 - M_0) \frac{h_1}{6},$$

d'on

$$M_0 \frac{h_1}{3} + M_1 \frac{h_1}{6} = \frac{y_1 - y_0}{h_1} - y'_0,$$

que, multiplicant per $6/h_1$, equival a

$$2M_0 + \lambda_0 M_1 = d_0, \quad (4.19)$$

amb

$$\lambda_0 = 1, \quad d_0 = \frac{6}{h_1} \left(\frac{y_1 - y_0}{h_1} - y'_0 \right).$$

També volem

$$y'_n = s'_{n-1}(x_n) = M_n \frac{h_n}{2} + \frac{y_n - y_{n-1}}{h_n} - (M_n - M_{n-1}) \frac{h_n}{6},$$

d'on

$$M_{n-1} \frac{h_n}{6} + M_n \frac{h_n}{3} = y'_n - \frac{y_n - y_{n-1}}{h_n},$$

que, multiplicant per $6/h_n$, equival a

$$\mu_n M_{n-1} + 2M_n = d_n, \quad (4.20)$$

amb

$$\mu_n = 1, \quad d_n = \frac{6}{h_n} \left(y'_n - \frac{y_n - y_{n-1}}{h_n} \right).$$

Cas natural. En aquest cas, volem

$$\begin{aligned} 0 = s''(x_0) = M_0 &\iff 2M_0 + \lambda_0 M_1 = d_0, \quad \text{amb } \lambda_0 = d_0 = 0 \\ 0 = s''(x_n) = M_n &\iff \mu_n M_{n-1} + 2M_n = d_n, \quad \text{amb } \mu_n = d_n = 0 \end{aligned} \quad (4.21)$$

Ajuntant (4.18) amb (4.19) i (4.20), i (4.18) amb (4.21), hem provat

Proposició 4.3.7 *Siguin $\{(x_i, y_i)\}_{i=0}^n \subset \mathbb{R}^2$ punts de suport, $x_0 < \dots < x_n$. Sigui $s \in S_3(\{x_i\}_{i=0}^n)$ el corresponent spline cúbic interpolador, amb condicions d'extrem o bé d'Hermite ($s'(x_0) = y'_0$, $s'(x_n) = y'_n$, per $y'_0, y'_n \in \mathbb{R}$ donats) o bé naturals ($s''(x_0) = s''(x_n) = 0$). Aleshores, els moments de l'spline satisfan el següent sistema d'equacions*

$$\begin{pmatrix} 2 & \lambda_0 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

on, per $i = 1, \dots, n-1$,

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right)$$

i, a més,

- en el cas hermític,

$$\begin{aligned} \lambda_0 &= 1, \quad d_0 = \frac{6}{h_1} \left(\frac{y_1 - y_0}{h_1} - y'_0 \right), \\ \mu_n &= 1, \quad d_n = \frac{6}{h_n} \left(y'_n - \frac{y_n - y_{n-1}}{h_n} \right), \end{aligned}$$

- en el cas natural,

$$\lambda_0 = d_0 = \mu_n = d_n = 0.$$

Observació 4.3.8 Òbviament, a la pràctica, en el cas natural podem posar $M_0 = M_n = 0$ i prescindir de la primera i darrera equacions. ∇

Cas periòdic. Una de les condicions de periodicitat és

$$s''(x_0) = s''(x_n) \iff M_0 = M_n,$$

i per tant només cal determinar M_1, \dots, M_n . Així, a les equacions (4.18) només cal afegir-ne una.

Imposem $s'_0(x_0) = s'_{n-1}(x_n)$:

$$\begin{aligned} -M_0 \frac{h_1}{2} + \frac{y_1 - y_0}{h_1} - (M_1 - M_0) \frac{h_1}{6} \\ = M_n \frac{h_n}{2} + \frac{y_n - y_{n-1}}{h_n} - (M_n - M_{n-1}) \frac{h_n}{6} \end{aligned}$$

Com que $M_0 = M_n$ i $y_0 = y_n$, si definim $y_{n+1} := y_1$, $h_{n+1} := h_1$, $M_{n+1} := M_1$ (correspon a estendre l'spline per periodicitat), l'equació anterior és l'equació (4.17) per $i = n$. El sistema d'equacions queda, llavors,

$$\left. \begin{array}{ccccccc} \mu_1 M_0 & + & 2M_1 & + & \lambda_1 M_2 & = & d_1 \\ \mu_2 M_1 & + & 2M_2 & + & \lambda_2 M_3 & = & d_2 \\ & & & & \vdots & & \\ \mu_{n-1} M_{n-2} & + & 2M_{n-1} & + & \lambda_{n-1} M_n & = & d_{n-1} \\ \mu_n M_{n-1} & + & 2M_n & + & \lambda_n M_{n+1} & = & d_n \end{array} \right\},$$

i, com que $M_0 = M_n$ i $M_{n+1} = M_1$, hem provat

Proposició 4.3.9 *Siguin $\{(x_i, y_i)\}_{i=0}^n \subset \mathbb{R}$ punts de suport amb $x_0 < \dots < x_n$, $y_0 = y_n$, i sigui $s \in S_3(\{x_i\}_{i=0}^n)$ el corresponent spline interpolador periòdic. Aleshores els seus moments satisfan*

$$\begin{pmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix},$$

on

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right)$$

i s'enten $y_{n+1} = y_1$, $h_{n+1} = h_1$.

Exemple 4.3.10 Anem a determinar l'spline cúbic interpolador amb punts de suport donats per la següent taula:

i	0	1	2	3	4
x_i	1	2	3	4	5
y_i	2	4	3	1	2

Primer trobem els moments, que satisfan el sistema d'equacions

$$\begin{pmatrix} 2 & \lambda_0 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \mu_2 & 2 & \lambda_2 & \\ & & \mu_3 & 2 & \lambda_3 \\ & & & \mu_4 & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}$$

Com que sabem $M_0 = M_4 = 0$ (l'spline és natural), podem eliminar la primera i última equacions i la primera i última incògnites, de manera que ens queda

$$\begin{pmatrix} 2 & \lambda_1 \\ \mu_2 & 2 & \lambda_2 \\ & \mu_3 & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

on

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right),$$

que surten (observeu que $h_i = 1 \forall i = 1, \dots, 4$)

i	λ_i	μ_i	d_i
1	1/2		-9
2	1/2	1/2	-3
3		1/2	9

Resolem el sistema per Gauss:

$$\left(\begin{array}{ccc|c} 2 & 1/2 & 0 & -9 \\ 1/2 & 2 & 1/2 & -3 \\ 0 & 1/2 & 2 & 9 \end{array} \right) \longrightarrow \left(\begin{array}{ccc|c} 2 & 1/2 & 0 & -9 \\ 0 & 15/8 & 1/2 & -3/4 \\ 0 & 1/2 & 2 & 9 \end{array} \right)$$

$$\longrightarrow \left(\begin{array}{ccc|c} 2 & 1/2 & 0 & -9 \\ 0 & 15/8 & 1/2 & -3/4 \\ 0 & 0 & 28/15 & 46/15 \end{array} \right)$$

d'on

$$M_3 = \frac{46/5}{28/15} = \frac{69}{14}$$

$$M_2 = \frac{-3/4 - (1/2)(69/14)}{15/8} = -\frac{12}{7}$$

$$M_1 = \frac{-9 - (1/2)(-12/7)}{2} = -\frac{57}{14}$$

Ara, per trobar s_0, s_1, s_2, s_3 , usem

$$s_i(x) = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3,$$

on

$$\alpha_i = y_i, \quad \gamma_i = \frac{M_i}{2},$$

$$\beta_i = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2M_i + M_{i+1}}{6}h_{i+1}, \quad \delta_i = \frac{M_{i+1} - M_i}{6h_{i+1}}.$$

Els $\alpha_i, \beta_i, \gamma_i, \delta_i$ surten

i	0	1	2	3
α_i	2	4	3	1
β_i	75/28	9/14	-9/4	-9/14
γ_i	0	-57/28	-6/7	69/28
δ_i	-19/28	11/28	31/28	-23/28

i, per tant, l'spline és:

$$s(x) = \begin{cases} s_0(x) = 2 + \frac{75}{28}(x-1) - \frac{19}{28}(x-1)^3, & x \in [1, 2], \\ s_1(x) = 4 + \frac{9}{14}(x-2) - \frac{57}{28}(x-2)^2 + \frac{11}{28}(x-2)^3, & x \in [2, 3], \\ s_2(x) = 3 - \frac{9}{4}(x-3) - \frac{6}{7}(x-3)^2 + \frac{31}{28}(x-3)^3, & x \in [3, 4], \\ s_3(x) = 1 - \frac{9}{14}(x-4) + \frac{69}{28}(x-4)^2 - \frac{23}{28}(x-4)^3, & x \in [4, 5] \end{cases}$$

▽