MÈTODES NUMÈRICS

Grau en Matemàtiques i itinerari doble Matemàtiques–Física, 2023–2024

Pràctica 1: recurrències numèricament estables i inestables

1 Objectiu

L'objectiu d'aquesta pràctica és experimentar amb recurrències numèricament estables i inestables. Concretament, experimentarem amb recurrències per al càlcul del sinus i cosinus dels múltiples d'un angle, és a dir, $\{\cos(jx)\}_{j=0}^{k}$ i $\{\sin(jx)\}_{j=0}^{k}$. Aquest és un càlcul freqüent quan, per exemple, es treballa amb sèries de Fourier.

2 Metodologia

Suposem que volem avaluar les quantitats

$$c_j := \cos(jx), \qquad s_j := \sin(jx),$$

per a $x \in \mathbb{R}$ i $j = 0, 1, 2, \ldots, k$. Una primera manera és fer servir les funcions sin i cos del llenguatge de programació, C en el nostre cas. Fer-ho així correspondria a implementar l'algorisme 2.1 tal com està escrit.

Algorisme 2.1 (càlcul directe de $\{c_j\}_{j=0}^{k}$, $\{s_j\}_{j=0}^{k}$)
Donats $k \in \mathbb{N}$, $x \in \mathbb{R}$ fem:

$c_0 := 1, \quad s_0 := 0$
$\forall j = 1, 2, \ldots, k$
$\quad c_j := \cos(jx)$
$\quad s_j := \sin(jx)$

Com que treballem en C, cada crida a la funció cos o sin es tradueix en una instrucció en codi màquina, perquè les funcions trigonomètriques estan implementades per hardware. Comprovareu en aquesta pràctica que va bastant ràpid (on bastant vol dir més aviat molt). Un avantatge d'aquest enfoc és que no acumulem error d'arrodoniment, perquè cada c_j i cada s_j s'obté mitjançant dues operacions elementals: un producte i un cosinus o sinus.

No obstant, podem guanyar una mica de velocitat si fem servir recurrències trigonomètriques. Si fem això, els c_{j+1}, s_{j+1} es calculen a partir dels c_j, s_j mitjançant unes poques operacions aritmètiques, que són més ràpides que avaluar sin o cos. Una primera opció és fer servir les fórmules del sinus i el cosinus de la suma:

$$\begin{cases} \cos\big((m+1)x\big) + \cos\big((m-1)x\big) = 2\cos x \cos(mx) \\ \sin\big((m+1)x\big) + \sin\big((m-1)x\big) = 2\cos x \sin(mx) \end{cases}$$
$$\Longrightarrow \begin{cases} \cos\big((m+1)x\big) = 2\cos x \cos(mx) - \cos\big((m-1)x\big) \\ \sin\big((m+1)x\big) = 2\cos x \sin(mx) - \sin\big((m-1)x\big) \end{cases}. \tag{1}$$

D'aquesta recurrència se segueix directament l'algorisme 2.2.

Algorisme 2.2 (Càlcul de $\{c_j\}_{j=0}^k$, $\{s_j\}_{j=0}^k$ mitjançant la recurrència (1))
Donats $k \in \mathbb{N}$, $x \in \mathbb{R}$, fem:

$$
\begin{aligned}
&c_0 := 1, \quad s_0 := 0\\
&c_1 := \cos x, \quad s_1 := \sin x\\
&\forall j = 1, 2, \ldots, k-1\\
&\qquad c_{j+1} := 2c_1c_j - c_{j-1},\\
&\qquad s_{j+1} := 2c_1s_j - s_{j-1}
\end{aligned}
$$

Però ara no és cert que cada c_j, s_j es calculi mitjançant poques operacions elementals: com que cada parella c_j, s_j depèn de les parelles anteriors, per tal d'obtenir c_k, s_k es fan moltes operacions si k es gran. Per tant, ens arrisquem a tenir inestabilitat numèrica. De fet, és el cas.

Per tal de considerar l'efecte dels errors d'arrodoniment, modelem la seva propagació de la següent manera: considerem que l'algorisme 2.2 fa les operacions de manera exacta però treballa a partir d'una aproximació $\tilde{c}$ de $c := \cos x$ amb error relatiu e fitat per l'èpsilon de la màquina ($|e| \leq \varepsilon$). Vist així, l'algorisme avalua de manera exacta les funcions

$$\varphi_1(c) := \cos(k \arccos c), \quad \varphi_2(c) := \sin(k \arccos c) \tag{2}$$

sobre l'aproximació $\tilde{c}$.[1] A [1, 2] p. 23 es demostra que, a primer ordre, els errors absoluts de $\varphi_1(\tilde{c})$, $\varphi_2(\tilde{c})$ son, respectivament,

$$e\frac{\cos x}{\sin x}k\sin(kx), \quad -e\frac{\cos x}{\sin x}k\cos(kx).$$

Observeu que, per a x petita (p.ex. 0.001) i k gran (p.ex. $k = 1000$), aquests errors es poden disparar. La recurrència és per tant numèricament inestable.

Una recurrència alternativa surt de fer servir les fórmules del sinus i el cosinus de la suma directament sobre $\cos\big((m+1)x\big)$, $\sin\big((m+1)x\big)$:

$$
\begin{aligned}
\cos\big((m+1)x\big) &= \cos(mx)\cos x - \sin(mx)\sin x,\\
\sin\big((m+1)x\big) &= \sin(mx)\cos x + \cos(mx)\sin x.
\end{aligned}
\tag{3}
$$

D'aquí se segueix immediatament l'algorisme 2.3.

[1] En el cas de φ_2, estem entenent que $\sin x = \pm\sqrt{1-c^2}$ s'avalua de manera exacta a partir de c.

Algorisme 2.3 (Càlcul de $\{c_j\}_{j=0}^k$, $\{s_j\}_{j=0}^k$ mitjançant (3))
Donats $k \in \mathbb{N}$, $x \in \mathbb{R}$, fem

$c_0 := 1$, $s_0 := 0$
$c_1 := \cos x$, $s_1 := \sin x$
$\forall j = 1, 2, \ldots, k-1$
$\quad c_{j+1} := c_1 c_j - s_1 s_j$
$\quad s_{j+1} := s_1 c_j + c_1 s_j$

De manera anàloga a com hem fet abans, modelem la propagació dels errors d'arrodoniment suposant que l'algorisme 2.3 avalua exactament la funció

$$\varphi\begin{pmatrix} c \\ s \end{pmatrix} := \begin{pmatrix} \cos(k \arccos c) \\ \sin(k \arccos s) \end{pmatrix}$$

a valors aproximats $\tilde{c}$, $\tilde{s}$ de $c := \cos x$, $s := \sin x$ que tenen errors relatius e_c, e_s, que satisfan $|e_c|, |e_s| \leq \varepsilon$. A [1, 2] p. 24 es demostra que (a primer ordre) l'error absolut de $\varphi(\tilde{c}, \tilde{s})$ és

$$e_c k \cos(kx) \begin{pmatrix} \cos\big((k-1)x\big) \\ \sin\big((k-1)x\big) \end{pmatrix} + e_s k \sin(kx) \begin{pmatrix} -\sin\big((k-1)x\big) \\ \cos\big((k-1)x\big) \end{pmatrix}.$$

Com que $\cos(kx)$, $\sin(kx)$ són de l'ordre de la unitat, si p.ex. $k = 1000$ podem perdre tres ordres de magnitud. Aquesta recurrència és millor que la de l'algorisme 2.2 de del punt de vista numèric, però encara és numèricament inestable.

Un truc astut per tal d'obtenir una recurrència amb millors propietats numèriques és considerar les diferències entre cosinus i sinus consecutius:

$$\begin{aligned} \delta^c_{m+1} &:= \cos\big((m+1)x\big) - \cos(mx) \\ &= 2(\cos x - 1)\cos(mx) - \sin x \sin(mx) - \cos x \cos mx + \cos mx \\ &= -4\big(\sin\frac{x}{2}\big)^2 \cos(mx) + \Big(\cos(mx) - \cos\big((m-1)x\big)\Big), \\ \delta^s_{m+1} &:= \sin\big((m+1)x\big) - \sin(mx) \\ &= 2(\cos x - 1)\sin(mx) - \sin x \cos(mx) - \cos x \sin mx + \sin mx \\ &= -4\big(\sin\frac{x}{2}\big)^2 \sin(mx) + \Big(\sin(mx) - \sin\big((m-1)x\big)\Big). \end{aligned} \tag{4}$$

Aquestes expressions donen lloc al següent algorisme:

Algorisme 2.4 (Càlcul de $\{c_j\}_{j=0}^k$, $\{s_k\}_{j=0}^k$ mitjançant (4))
Donats $k \in \mathbb{N}$, $x \in \mathbb{R}$ fem:

$$\begin{aligned}
&\delta_1^c := -2\big(\sin\tfrac{x}{2}\big)^2, \quad t := 2\delta_1^c \\
&\delta_1^s := (-1)^{\lfloor x/\pi \rfloor}\sqrt{-\delta_1^c(2+\delta_1^c)} \\
&c_0 := 1, \quad s_0 := 0 \\
&\forall j = 1, 2, \ldots, k \\
&\qquad c_j := c_{j-1} + \delta_j^c, \quad \delta_{j+1}^c := tc_j + \delta_j^c \\
&\qquad s_j := s_{j-1} + \delta_j^s, \quad \delta_{j+1}^s := ts_j + \delta_j^s
\end{aligned}$$

Novament com abans, estimem la propagació de l'error d'arrodoniment de l'algorisme 2.4 considerant que opera exactament a partir d'una aproximació $\tilde{s}$ de $s := \sin(x/2)$. Vist així, l'algorisme 2.4 avalua les funcions

$$\psi_1(s) := \cos\big(2k \arcsin s\big), \quad \psi_2(x) := \sin\big(2k \arcsin s\big)$$

sobre l'aproximació $\tilde{s}$ que considerem que té error relatiu e amb $|e| \leq \varepsilon$. A [1, 2] p. 25 es prova que els errors absoluts de $\psi_1(\tilde{s})$, $\psi_2(\tilde{s})$ són, respectivament,

$$\big(-2k \tan\frac{x}{2} \sin(kx)\big)e, \quad \big(2k \tan\frac{x}{2} \cos(kx)\big)e.$$

Observeu que ara, si $k|x| \approx 1$, els errors anteriors es mantenen de l'ordre de l'èpsilon màquina. Aquesta darrera recurrència és, per tant, numèricament estable.

3 Etapes de desenvolupament i experiments

1.– Escriviu funcions amb prototipus

```
void trigrec0 (int k, double x, double c[], double s[]);
void trigrec1 (int k, double x, double c[], double s[]);
void trigrec2 (int k, double x, double c[], double s[]);
void trigrec3 (int k, double x, double c[], double s[]);
```

Cadascuna d'aquestes funcions ha de calcular els valors $\{\cos(jx)\}_{j=0}^k$, $\{\sin(jx)\}_{j=0}^k$, on k és l'argument `k` i x és l'argument `x`, de manera que el valor $\cos(jx)$ ha de quedar guardat dins `c[j]` i el valor $\sin(jx)$ ha de quedar guardat dins `s[j]`. Els vectors `c[]`, `s[]` han de ser de $k+1$ components cadascun i han d'haver estat allocatats prèviament a ser passats a aquestes funcions.

Per al càlcul de $\{\cos(jx)\}_{j=0}^k$, $\{\sin(jx)\}_{j=0}^k$, la funció `trigrec0()` ha de fer servir l'algorisme 2.1, `trigrec1()` ha de fer servir l'algorisme 2.2, `trigrec2()` ha de fer servir l'algorisme 2.3 i `trigrec3()` ha de fer servir l'algorisme 2.4.

2.– Escriviu un programa principal `trigrec_escr.c` que respongui a la crida

```
./trigrec_escr k x
```

i que bolqui línies de la forma

$$c_j^{(0)} \quad c_j^{(1)} \quad c_j^{(2)} \quad c_j^{(3)} \quad s_j^{(0)} \quad s_j^{(1)} \quad s_j^{(2)} \quad s_j^{(3)}$$

per $j = 0, \ldots, k$, on els $\{c_j^{(0)}\}_{j=0}^k, \{s_j^{(0)}\}_{j=0}^k$ estan calculats amb `trigrec0()`, els $\{c_j^{(1)}\}_{j=0}^k, \{s_j^{(1)}\}_{j=0}^k$ estan calculats amb `trigrec1()`, i així successivament. Feu-lo servir per tal de validar les funcions `trigrec0()`, `trigrec1()`, `trigrec2()`, `trigrec3()`.

3.– Escriviu un programa principal `trigrec_escr_err` que respongui a la crida

```
./trigrec_temps_err k x
```

on `k` i `x` tenen el mateix significat que al punt anterior. Com al punt anterior, aquest programa ha de cridar les funcions `trigrec0()`, `trigrec1()`, `trigrec2()`, `trigrec3()` per tal de calcular els valors $\{c_j^{(i)}\}_{j=0}^k, \{s_j^{(i)}\}_{j=0}^k$, $i = 0, 1, 2, 3$ del punt anterior. Després de cridar cada funció, `trigrec_temps_err` ha d'escriure el seu temps de càlcul (que podeu obtenir mitjançant la funció `temps()` del fitxer `temps.c` que trobareu a la web). Després de calcular aquests valors, ha d'estimar els errors comesos per les funcions `trigrec1()`, `trigrec2()`, `trigrec3()` mitjançant els valors

$$\Delta_c^i := \max_{0\le j\le k} |c_j^{(i)} - c_j^{(0)}|, \qquad \Delta_s^i := \max_{0\le j\le k} |s_j^{(i)} - s_j^{(0)}|$$

per a $i = 1, 2, 3$. Feu que el programa escrigui les línies

`errc1` Δ_c^1 `errc2` Δ_c^2 `errc3` Δ_c^3
`errs1` Δ_s^1 `errs2` Δ_s^2 `errs3` Δ_s^3

on els Δ_s^i, Δ_c^i han d'estar substituits pels seus valors numèrics.

4.– Experimenteu amb el programa `trigrec_temps_err` amb els valors de k i x de la següent taula:

k	l
100	0.0123
1000	0.00123
10000	0.000123
100000	0.0000123
1000000	0.00000123
10000000	0.000000123

Expliqueu els errors $\{\Delta_c^i, \Delta_s^i\}_{i=1}^3$ obtinguts, en termes de l'anàlisi de la secció 2.

5.– Compteu el nombre d'operacions que fan els algorismes 2.3 i 2.4 com a funció de k. Feu experiments per tal d'estimar el nombre d'operacions per segon que fan els algorismes 2.3 i 2.4 al vostre ordinador. A partir d'això, estimeu a quantes operacions aritmètiques equival (en termes del temps d'execució) avaluar les funcions sin i cos a l'algorisme 2.1.

4 Lliurament

Heu de lliurar aquesta pràctica a través del campus virtual, dins el termini que consti al lliurament. Els retards es penalitzaran multiplicant la qualificació per $0.9^{\#\text{dies retard}}$. Heu de pujar un únic fitxer `NIU.zip`, on `NIU` és el vostre NIU, que contingui els següents fitxers:

- Un fitxer de biblioteca `trigrec.c`, amb el codi de les funcions `trigrec0()`, `trigrec1()`, `trigrec2()`, `trigrec3()`.
- Un fitxer de capçalera `trigrec.h`, amb els prototipus de les funcions `trigrec0()`, `trigrec1()`, `trigrec2()`, `trigrec3()`.
- Un fitxer `trigrec_escr.c` amb l'utilitat `trigrec_escr`, que s'ha de poder generar amb

  ```
  make trigrec_escr
  ```

  a partir del fitxer `Makefile` que trobareu a la web.
- Un fitxer `trigrec_temps_err.c` amb l'utilitat `trigrec_temps_err`, que s'ha de poder generar amb

  ```
  make trigrec_temps_err
  ```

  a partir del fitxer `Makefile` que trobareu a la web.
- Un fitxer `memoria.pdf` amb l'informe de la pràctica.

Referències

[1] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer, Berlin, second edition, 1993.

[2] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer, Berlin, third edition, 2002.