Mètodes Numèrics
Pràctica 1: Tema d'errors

Albert Rodas Barceló i Marc Seguí Coll
Universitat Autònoma de Barcelona

Març 2019

1 Exercici 1

L'objectiu del primer problema consisteix en observar errors de cancel·lació quan la màquina avalua la funció de variable real $1 - \cos(x)$ si $|x| \ll 1$. Per tal de fer plausibles els resultats, es considera la funció:

$$f(x) = \begin{cases} \frac{1-\cos(x)}{x^2} & \text{si } x \neq 0, \\ 1/2 & \text{si } x = 0. \end{cases} \tag{1}$$

Aquesta és contínua a $\mathbb{R}$ ja que per la regla de l'Hôpital:

$$\lim_{x\to 0} \frac{1-\cos(x)}{x^2} = \lim_{x\to 0} \frac{\sin(x)}{2x} = \lim_{x\to 0} \frac{\cos(x)}{2} = \frac{1}{2}$$

Per tant, si s'avalua la funció a $|x| \ll 1$ el resultat és proper a $\frac{1}{2}$. Deguda la manca de precisió de la màquina, es comprovarà que per a certs $x \in \mathbb{R}$ suficientment petits, el valor que pren $f(x)$ és completament erroni. Mitjançant la igualtat $\cos(2\theta) = 1 - 2\sin^2(\theta)$, a l'apartat **(b)** s'ha emprat per a $x \neq 0$ l'expressió

$$f(x) = \frac{2\sin^2(x/2)}{x^2} \tag{2}$$

1.1 Resultats

A l'apartat **(a)**, en avaluar la funció en el punt $x_0 = 1.2 \cdot 10^{-5}$ el programa escrit en *float* (`p1as.c`) retorna $f(x_0) = 0$ i el programa en *double* (`p1ad.c`) mostra com a resulatat $f(x_0) = 0.4999997329749008$. Analitzant el que succeeix a `coss.c` i `cosd.c`, el valor de $\cos(x_0)$ en el primer cas es correspon a exactament 1 i en el cas del segon codi el resultat obtingut és 0.999999999928.

En canvi, a l'apartat **(b)** els valors resultants són $f(x_0) = 0.5$ per al cas simple (`p1bs.c`) i $f(x_0) = 0.499999999994$ per al cas en doble precisió (`p1bd.c`). Noteu que tot i haver especificat la sortida per pantalla de nombres amb 16 dígits, el nombre de xifres mostrades per pantalla es veu afectat per l'arrodoniment que realitza la màquina.

1.2 Conclusions

L'error de cancel·lació és la pèrdua de precisió a les xifres significatives degut a la resta de quantitats molt properes. En aquest cas es deu al fet que en *float* la funció $\cos(x)$ retorna el número 1 (resultat de l'arrodonoiment) per $x \leq 10^{-4}$. Aleshores la màquina retorna el valor 0 en comptes d'un valor pròxim a $1/2$ en avaluar $f(x_0)$, de manera que es passa d'una precisió de 10^{-7} a una de 10^0. En format *double* es manifesta el mateix problema per a $x \leq 10^{-8}$. Es pot veure que el nombre de condició[1] de la funció $1-\cos(x)$ és proper a 2 (nombre més gran que 1) per a $|x| \ll 1$ i per tant està mal condicionada[2].

2 Exercici 2

La determinació de les solucions x_+, x_- de l'equació de segon grau $ax^2+bx+c=0$ amb coeficients reals i $a \neq 0$ mitjançant la fórmula mal condicionada

$$x_\pm = \frac{-b \pm \sqrt{b^2-4ac}}{2a}, \tag{3}$$

provoca que hi puguin aparèixer cancel·lacions quan $b^2 \gg ac$. En particular, si $b<0$ llavors x_- sofreix errors de cancel·lació i si $b>0$ succeeix el mateix però amb x_+. Per tal d'evitar aquestes errades, es proposa una fórmula alternativa per a cada cas, multiplicant pel conjugat, si $b<0$:

$$x_- = \frac{2c}{-b+\sqrt{b^2-4ac}} \quad \text{si } b<0, \quad x_+ = \frac{-2c}{b+\sqrt{b^2-4ac}} \quad \text{si } b>0 \tag{4}$$

2.1 Resultats

Siguin $a=0.004$, $b=10500$, $c=0.02$ els coeficients de l'equació de segon grau $ax^2+bx+c=0$. El programa en *float* (`p2as.c`) retorna per pantalla els valors $x_+=0$ i $x_-=-2625000$. D'altra banda el representat en format *double* (`p2ad.c`) mostra com a solucions $x_+ = 1.904709279187955 \cdot 10^{-6}$ i $x_- = -2624999.999998095$.

Si aquestes solucions es determinen amb els codis que incorporen les fórmules de 4 (`p2bs.c` i `p2bd.c`) s'obten les solucions $x_+ = -1.90476191619382 \cdot 10^{-6}$ i $x_- = -2625000$ en precisió simple i $x_+ = 1.904761904763287 \cdot 10^{-6}$ i $x_- = -2624999.999998095$.

Considerem ara l'equació de segon grau $0.002x^2 + 10^9 x + 0.001 = 0$. A continuació s'exposen els resultats obtinguts:

Programa	x_+	x_-
`p2as.c`	0	-499999965184
`p2ad.c`	0	-500000000000
`p2bs.c`	$-9.999999960041972 \cdot 10^{-13}$	-499999965184
`p2bd.c`	-10^{-12}	-500000000000

Amb aquest exemple s'ilustra també la cancel·lació en la fórmula 3 en format *double* a la vegada que queda plausible l'efectivitat de les fórmules de 4.

[1] El nombre de condició per a una funció de variable real $g: U \subset \mathbb{R} \longrightarrow \mathbb{R}$ es defineix com $\mathcal{K}\{g(x)\} = \frac{\delta g(x)}{\delta x}$ on δ denota l'error relatiu comès en el càlcul. A la pràctica, si $g \in \mathcal{C}^1$ llavors $\mathcal{K}\{g(x)\} \approx \left|\frac{x g'(x)}{g(x)}\right|$.

[2] Si $\mathcal{K}\{g(x_0)\} > 1$ es diu que la fórmula donada per $g(x)$ està mal condicionada per a valors propers a x_0

2.2 Conclusions

En aquest exercici la situació és similar a la del primer. Pel fet de treballar amb $b^2 \gg |4ac|$, l'ordinador en fer $b^2 - 4ac$ obté un número molt proper a b^2. Aleshores, en fer l'arrel quadrada (en *float*) d'aquest número ens retorna $|b|$ (per nombres amb prou diferència d'ordre de magnitud això també passaria per *double*, cosa que es veu al darrer exemple a `p2ad.c`). Així, quan després computa la resta es produeix un error de cancel·lació i una de les solucions de l'equació que ens retorna la màquina és 0 quan és evident que no pot ser.

3 Exercici 3

Donat un conjunt de dades $X = \{x_1, \ldots, x_n\}$ es defineix la mitjana d'aquests valors i la variança com:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{5}$$

Una forma equivalent per calcular la variança és

$$s_n^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right) \tag{6}$$

Els codis emprats calculen la variança donat un conjunt de dades que s'especifica per pantalla en format *float* `p3as.c` i el mateix en precisió doble `p3ad.c`. Dits programes ho fan primer amb la definició 5 i posteriorment amb la fórmula 6. El programa `p3c.c` calcula la variança en format simple i doble d'un conjunt de N nombres generat a partir d'un nombre y introduït per pantalla i nombres aleatoris continguts en un interval $[y, y + \delta]$, on δ és un nombre establert per l'usuari.

3.1 Resultats

Programa	`p3as.c`	`p3ad.c`
s_n^2 (definició)	1	1
s_n^2 (fórmula)	0	1

`p3c.c`	*simple*	*double*
s_n^2 (definició)	1.7570396	$8.3022426 \cdot 10^{-6}$
s_n^2 (fórmula)	$-4.9929494 \cdot 10^{5}$	$2.5312753 \cdot 10^{-3}$

A la primera taula s'ha considerat el conjunt $X = \{10000, 10001, 10002\}$. Com a exemple de l'apartat **(c)**, s'ha emprat un nombre $y = 10^5$ que ha generat 10^5 nombres al seu voltant en un interval entre 10^5 i $1.01 \cdot 10^5$. Els resultats d'aquest darrer cas són els de la segona taula.

3.2 Conclusions

Com s'ha comentat a la introducció de l'exercici, la fórmula 5 provoca cancel·lacions molt greus. Això té a veure amb el fet de primer elevar al quadrat i sumar després. Al elevar al quadrat, la mantissa del nombre es fa notablement més llarga en la majoria dels casos, de manera que és més fàcil que es perdin xifres significatives ja que el format no permet guardar-les totes. Al segon exemple presentat la cancel·lació deguda a l'arrodoniment queda encara més clara, donant un valor negatiu (cosa totalment contradictòria) al cas de la segona fórmula en *float*. Inclús en precisió doble la segona fórmula difereix en 3 ordres de magnitud del valor resultant de la definició.

4 Exercici 4

Per aquest exercici es calculen les sumes parcials fins a un nombre N que es pregunta per pantalla de la sèrie

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} = 1.644934066848226\ldots \tag{7}$$

en ordre ascendent i descendent i amb precisió simple i doble.

Als programes `p4kahans.c` i `p4kahand.c` es proporciona una manera alternativa de calcular el valor d'aquestes sumes parcials mitjançant la sumació de Kahan que consisteix en aproximar el valor de l'error en cada pas per tal de corregir el resultat emprant l'expressió

$$e(x+y) = fl(fl(x - fl(x+y)) + y) \tag{8}$$

On e denota l'error al sumar $x+y$ i fl el còmput que fa l'ordinador en *float* (equivalent, amb més precisió, però *double*). Es pot veure com aquesta cota de l'error és una bona de manera de corregir la suma (S) fent $S = x + y + e(x+y)$ s'obté un valor més precís[3].

Una altra possibilitat per minimitzar l'error comès consisteix en agrupar els termes pel seu ordre de magnitud (`p4ds.c` i `p4dd.c`). Aquest mètode es beneficia del fet que es perd menys precisió al sumar números d'un ordre de magnitud semblant (com s'explica a les conclusions) i per tant, la idea al darrera és semblant a la proposta de fer el sumatori en ordre descendent.

4.1 Resultats

Es presenten els resultats per diferents valors de N i pels programes esmentats en la taula següent:

N	`p4as.c`	`p4bs.c`	`p4kahans.c`	`p4ad.c`	`p4bd.c`	`p4kahand.c`
10^5	1.644725323	1.644924045	1.644924045	1.644924067	1.644924067	1.644924067
10^6	1.644725323	1.644932985	1.644933105	1.644933067	1.644933067	1.644933067
10^7	1.644725323	1.644933939	1.644933939	1.644933967	1.644933967	1.644933967
10^8	1.644725323	1.644934058	1.644934058	1.644934058	1.644934057	1.644934057
10^9	1.644725323	1.644934058	1.644934058	1.644934058	1.644934066	1.644934066

4.2 Conclusions

El fet d'obtenir un resultat menys precís en sumar de forma ascendent és degut al fet que els últims termes a sumar tenen un ordre de magnitud molt més petit que la suma acumulada de manera que es produeix novament un error de cancel·lació. Això es pot veure tenint en compte que l'ordinador guarda els números com una mantissa i un exponent. La mantissa té més o menys extensió depenent de si treballem en format *float* o *double*, però en ambdós casos, si l'exponent entre dos números és molt diferent, les mantisses no coincidiran i en operar, el número gran quedarà inalterat. Per exemple, suposem que tenim una mantissa de 5 xifres. Aleshores

$$f(0.12345 \cdot 10^2 + 0.54321 \cdot 10^{-7}) = f(12.345000054321) = 0.12345, \tag{9}$$

on f representa tornar el resultat amb una mantissa de 5 xifres. Sumant començant pels termes més petits es redueix en gran mesura l'error comès i aquest és un fet a tenir en compte ja que es produirà per a qualsevol sèrie convergent.

[3]Per informació més detallada en el mètode i la seva demostració, veure a partir de la pàgina 70 del següent PDF `http://www.lcc.uma.es/~villa/tn/tema02.pdf`

5 Exercici 5

Donades les quantitats $a \geq b \geq c$ corresponents a la longitud dels costats d'un cert triangle ABC és coneguda la fórmula d'Heró per a calcular-ne l'àrea S

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{on} \quad p = \frac{a+b+c}{2} \tag{10}$$

Una alternativa per al càlcul de l'àrea del triangle ABC donats els seus costats és

$$S = \frac{1}{2}\sqrt{a^2c^2 - \frac{1}{4}\left(a^2 + c^2 - b^2\right)^2} \tag{11}$$

5.1 Resultats

A continuació es presenten els resultats obtinguts en calcular l'àrea dels triangles $a = 10^7, b = 10^7$ i $c = 1$ i $a' = 5.0000001$, $b' = 4.0000002$, $c' = 1.0000003$ amb la fórmula 10 (`p5as.c` i `p5ad.c`) i amb la fórmula 11 (`p5cs.c` i `p5cd.c`).

abc	`p5as.c`	`p5ad.c`	`p5cs.c`	`p5cd.c`
Àrea	0	$4.99999999 \cdot 10^6$	$5 \cdot 10^6$	$4.99999999 \cdot 10^6$

$a'b'c'$	`p5as.c`	`p5ad.c`	`p5cs.c`	`p5cd.c`
Àrea	0	$2.00000015836 \cdot 10^{-3}$	$2.183660166338 \cdot 10^{-3}$	$2.00000015868 \cdot 10^{-3}$

5.2 Conclusions

Considerem les quantitats $\tilde{a}$, $\tilde{b}$, $\tilde{c}$ les representacions en punt flotant de les longituds dels costats. Tenim que l'error relatiu entre la longitud real i la seva representació en punt flotant és una quantitat que depèn de la precisió de la màquina. Si calculem l'error relatiu que es produeix en calcular l'àrea del triangle mitjançant les dues fórmules a partirla fórmula de propagació de l'error maximal, emprant que $\varepsilon_r(a, \tilde{a}) = \varepsilon_r(b, \tilde{b}) = \varepsilon_r(c, \tilde{c}) = \varepsilon$ obtenim

$$\varepsilon_r\left(S(a,b,c)\right) = \varepsilon_r\left(S(a,b,c), S(\tilde{a},\tilde{b},\tilde{c})\right) = \varepsilon\left(\left|\frac{a\frac{\partial S(a,b,c)}{\partial a}}{S(a,b,c)}\right| + \left|\frac{b\frac{\partial S(a,b,c)}{\partial b}}{S(a,b,c)}\right| + \left|\frac{c\frac{\partial S(a,b,c)}{\partial c}}{S(a,b,c)}\right|\right) \tag{12}$$

Per veure la millora entre fórmules, amb els valors d'a, b, c dels exemples de la subsecció anterior tindrem que $\varepsilon_r\left(S(a',b',c')\right) = \varepsilon \cdot 1.375 \cdot 10^7$ i $\varepsilon_r\left(S(a,b,c)\right) = \varepsilon \cdot 10^7$ per a la fórmula 10 i $\varepsilon_r\left(S(a',b',c')\right) = \varepsilon \cdot 2.453333$ i $\varepsilon_r\left(S(a,b,c)\right) = \varepsilon \cdot 2$ per a l'igualtat 11. Aquestes dades s'han calculat amb `Sagemath` ja que els càlculs són prou farragosos.

Els resultats obtinguts són prou clars, es pot observar que en els casos que $a \approx b + c$ l'error relatiu en la primera fórmula es dispara -(produint cancel·lacions òbvies mostrades als resultats) mentre que a la segona l'error relatiu és de l'ordre de l'error relatiu entre la longitud dels costats i la seva representació en punt flotant ε, justificant així la millora prou significativa que aporta 11. Aquest error relatiu es podria implementar a la definició del nombre de condició i comprovar que efectivament la segona fórmula és més eficient en aquest cas.