Mètodes Numèrics
Pràctica 1: Errors

Raquel García, Arnau Mas

11 de Març 2018

Problema 1

Considerem la funció

$$f(x) = \begin{cases} \dfrac{1-\cos x}{x^2} & \text{si } x = 0 \\ \dfrac{1}{2} & \text{si } x \neq 0 \end{cases}$$

Primer observem que f està definida i és continua a tot $\mathbb{R}$. Això és clar per $x \neq 0$. I per $x = 0$ fem servir que $\cos x \sim 1 - \frac{1}{2}x^2$ quan $x \to 0$. I per tant

$$\lim_{x\to 0} \frac{1-\cos x}{x^2} = \frac{1}{2}.$$

Hem de comprovar que per tot $x \in \mathbb{R}^\times$ es compleix $0 \leq f(x) < \frac{1}{2}$. Veure que f és positiva a tot arreu és senzill tenint en compte que per tot $x \in \mathbb{R}$ es té $1 - \cos x \geq 1 - 1 = 0$ i que $x^2 \geq 0$ per $x \in \mathbb{R}$. Per veure la fita superior procedim en dues parts. Primer observem que per tot $x \in \mathbb{R}^\times$ es té

$$\frac{1-\cos x}{x^2} \leq \left|\frac{1-\cos x}{x^2}\right| < \frac{1+|\cos x|}{x^2} < \frac{2}{x^2}.$$

Ara bé, aquesta fita només ens és útil per $|x| > 2$ ja que aleshores es té

$$\frac{1-\cos x}{x^2} < \frac{2}{2^2} = \frac{1}{2}$$

ja que $\frac{1}{x^2}$ és estrictament decreixent per $x > 0$ i estrictament creixent per $x < 0$. Per demostrar la fita prop del zero farem ús del teorema de Taylor amb la forma de Lagrange per l'error. Primer trobem una desigualtat equivalent a la desigualtat que volem veure:

$$\begin{aligned} \frac{1-\cos x}{x^2} < \frac{1}{2} &\iff -\cos x < \frac{1}{2}x^2 - 1 \\ &\iff \cos x > 1 - \frac{1}{2}x^2. \end{aligned} \tag{$*$}$$

El desenvolupament de Taylor fins a ordre 2 de $\cos x$ al voltant de 0 és $1 - \frac{x^2}{2}$. Si considerem $x \in [-2, 2]$ el teorema de Taylor ens garanteix que existeix $0 < a < x$ o $x < a < 0$ segons si $x > 0$ o $x < 0$ tal que

$$\cos x = 1 - \frac{1}{2}x^2 + \frac{\sin a}{3!}x^3.$$

Si substituïm a $(*)$ trobem que hem de veure que

$$\frac{\sin a}{3!}x^3 > 0$$

per $x \in [-2, 2]$. Si $x > 0$ tenim que $a < x < 2$. Com que $0 < 2 < \pi$ tenim $\sin a > 0$ i tenim la desigualtat que volem ja que $x^3 > 0$ si $x > 0$. En canvi, si $x < 0$ tenim $x^3 < 0$. Però en aquest cas $-2 < x < a < 0$. I per tant, com que ara $-\pi < a < 0$ es compleix que $\sin a < 0$ i per tant també tenim la desigualtat que voliem.

Els programes `prob1b_fl.c` i `prob1b_do.c` calculen f amb precisió simple i doble respectivament. Si avaluem al punt indicat, $x_0 = 1.2 \times 10^{-5}$ trobem que el programa amb precisó simple retorna 0 fins a 8 xifres, mentre que el programa amb precisió doble retorna un valor molt proper a $\frac{1}{2}$, concretament $0.499\,999\,7$, arrodonint fins a 7 xifres decimals. Ja hem observat que f és continua —i de fet de classe $\mathcal{C}^\infty$—, per tant, com que x_0 és proper a 0, és raonable pensar que $f(x_0)$ hauria de ser molt proper a $\frac{1}{2}$. Això és el que ens dóna el programa en precisió doble.

Si fem servir que $1 - \cos x = 2\sin(x/2)^2$ podem reescriure f com

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

per $x \neq 0$. Si implementem això en codi —tal i com es fa en els programes `prob1c_fl.c` i `prob1c_do.c`— veiem que ara obtenim el resultat correcte tant en precisió doble com en simple.

L'error que apareix en la primera implementació de f és un error de representació. Si calculem $\cos x_0$ en precisió doble trobem que el resultat difereix de 1 a l'onzena xifra decimal. Quan representem aquest valor en precisió simple obtenim 1 degut a l'arrodoniment. Per tant obtenim 0 com a resultat de $f(x_0)$ en precisió simple. En canvi, quan executem la segona implementació, el càlcul $2\sin(x_0/2)^2$ retorna $x_0^2/4$ quan el representem en precisió simple, de manera que obtenim el resultat esperat.

Problema 2

Considerem l'equació quadràtica

$$ax^2 + bx + c = 0 \qquad a \neq 0$$

que té per arrels

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{i} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Suposem que $a > 0$ i $b^2 > 4ac$, per tant hi ha dues arrels diferents i són reals.

Volem demostrar que quan fem el càlcul numèric alguna de les arrels estarà contaminada per error de cancelació si $b^2 \gg 4ac$. En aquest cas:

$$\sqrt{b^2 - 4ac} \approx |b| \tag{1}$$

Si $b > 0$ aleshores $|b| = b$ i podem veure que l'arrel que patirà cancelació serà x_1, ja que si considerem la suma del numerador $-b + \sqrt{b^2 - 4ac}$ i calculem la propagació de la fita de l'error relatiu procedent de la representació en punt flotant i de les operacions prèvies:

$$\varepsilon_r(-b + \sqrt{b^2 - 4ac}) = \left|\frac{-b}{-b + \sqrt{b^2 - 4ac}}\right| \varepsilon_r(-b) + \left|\frac{\sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}}\right| \varepsilon_r(\sqrt{b^2 - 4ac})$$

veiem que el denominador dels coeficients de propagació tendeix a zero:

$$-b + \sqrt{b^2 - 4ac} \approx -b + b \to 0$$

per (1), i com que el numerador està fitat per $b > 0$, aleshores l'error relatiu es dispara, és a dir, hi haurà cancelació. També observem que el càlcul de x_2 serà segur, ja que els coeficients de propagació de l'error relatiu a l'efectuar la suma del numerador estaran fitats per 1/2:

$$\frac{|b|}{|-b - \sqrt{b^2 - 4ac}|} \leq \frac{|b|}{|-2b|} = \frac{1}{2}.$$

Si $b < 0$ tindrem $|b| = -b$, i pel mateix raonament l'arrel que ara patirà cancelació serà x_2, i calcular x_1 serà segur.

Per a pensar una manera alternativa de calcular les arrels evitant l'error de cancelació, podem tenir en compte que tot polinomi $ax^2 + bx + c$ amb arrels x_1 i x_2 es pot escriure de la següent manera:

$$a(x - x_1)(x - x_2) = ax^2 - a(x_1 + x_2)x + ax_1x_2 \tag{2}$$

i per tant $b = -a(x_1 + x_2)$ i $c = ax_1x_2$. Ja hem vist que tant si $b > 0$ com $b < 0$, és possible calcular x_2 o x_1 de manera segura respectivament. Per tant en cada cas podem avaluar una de les arrels amb la fórmula estàndard obtenint un resultat

prou exacte. Per a calcular l'arrel que patiria cancelació, considerant (2) veiem que coneixent l'arrel segura podem trobar l'altra fent:

$$x_1 = \frac{c}{ax_2} \quad \text{o} \quad x_2 = \frac{c}{ax_1}$$

segons sigui x_2 ó x_1 l'arrel segura, ja que el càlcul de productes i divisions és segur des del punt de vista numèric.

Els programes `prob2a_fl.c` i `prob2a_do.c` avaluen les arrels amb la fórmula estàndard amb precisió simple i doble respectivament. I els programes `prob2b_fl.c` i `prob2b_do.c` les evaluen amb el procediment descrit anteriorment amb precisió simple i doble respectivament.

Considerem el següent polinomi

$$x^2 - (10^5 + 10^{-5})x + 1 = (x - 10^5)(x - 10^{-5}),$$

que sabem que té per arrels $x_1 = 10^5$ i $x_2 = 10^{-5}$. El programa `prob2a_fl.c` ens retorna $x_1 = 10^5$ i $x_2 = 0$. Veiem, doncs, que l'arrel segura x_1 s'ha calculat correctament, però que al càlcul x_2 hi ha hagut una cancelació. El programa `prob2a_do.c` ens retorna $x_1 = 10^5$ i $x_2 = 10^{-5}$, que és el resultat correcte. Per tant el càlcul amb la fórmula només s'ha vist afectat per cancelació en precisió simple. Els programes `prob2b_fl.c` i `prob2b_do.c` ens retornen ambdós $x_1 = 10^5$ i $x_2 = 10^{-5}$, per tant amb aquest mètode s'elimina l'error de cancelació tant en precisió simple com en doble.

Considerem ara el següent polinomi:

$$x^2 + (3 \times 10^8 + 2 \times 10^{-8})x + 6 = (x + 3 \times 10^8)(x + 2 \times 10^{-8})$$

que sabem que té per arrels $x_1 = -2 \times 10^{-8}$ i $x_2 = -3 \times 10^8$. Els programes `prob2b_do.c` i `prob2b_fl.c` retornen ambdós el resultat correcte. En canvi el programa `prob2a_do.c` retorna $x_1 = -2.98 \times 10^{-8}$ i $x_2 = -3 \times 10^8$, i el programa `prob2a_fl.c` retorna $x_1 = 2.546\,619\,8 \times 10^{-1}$ i $x_2 = -3 \times 10^8$. Per tant observem que amb la fórmula de les arrels per a polinomis de segon grau, el càlcul de l'arrel no segura és totalment erroni en precisió simple, i també apareixen errors en precisió doble, deguts a la pèrdua de xifres significatives.

Problema 3

El programa `prob3.c` conté funcions que calculen la variança d'una mostra en un i en dos bucles, en precisió simple i doble. Els programes `prob3a1_fl.c` i `prob3a1_do.c` fan servir les funcions de `prob3.c` per llegir dades d'un fitxer i calcular-ne la variança mostral en un sol bucle, en precisió simple i doble respectivament. Els programes `prob3a2_fl.c` i `prob3a2_do.c` fan el mateix però fent servir dos bucles. Si executem cada un d'aquests quatre programes passant com a mostra $(10\,000, 10\,001, 10\,002)$ trobem que tots els programes calculen correctament la variança com a 1, tret de `prob3a1_fl.c`, que dóna el resultat com a 0.

Si fem un anàlisi més acurat de com s'està fent el càlcul amb un sol bucle descobrim que hi ha un error de cancelació. Si, per $i \leq n$ posem $x_i = x + \delta_i$ amb $|\delta_i| < \delta$ tenim que la mitjana dels x_i és de l'ordre de x i per tant $x_i - \bar{x}$ és de l'ordre de δ. Així, donada la definició de la variança, s_n^2.

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2, \tag{3}$$

trobem que és de l'ordre de δ^2. Si en canvi fem el càlcul amb un sol bucle, és a dir, fent servir la següent expressió equivalent per a la variància:

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} x_i^2 - \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} x_i \right)^2 \tag{4}$$

observem que tant el primer terme com el segon terme són de l'ordre de x^2. Posem que $x \sim 10^k$ i $\delta \sim 10^{-l}$. Aleshores estem dient que els dos termes de (4) són d'ordre 10^{2k} i que la seva diferència és d'ordre 10^{-2l}, per tant han de coincidir en les $2(k+l)$ primeres xifres. Si aquest nombre de xifres és més gran que el nombre de xifres que tenim disponible en la precisió que estiguem treballant, aleshores el resultat que obtinguem fent el càlcul amb un bucle distarà del resultat correcte.

Si estem treballant en precisió simple tenim 7 xifres significatives disponibles. Per tant, si $2(k+l) > 7$ hauriem d'observar cancelació fent el càlcul amb un sol bucle. Per a comprovar això, el programa `prob3c.c` genera una llista de n nombres que no disten més de δ d'un nombre donat x. És a dir, genera n nombres uniformement distribuïts a l'interval $[x - \delta, x + \delta]$. Si triem $x = 10^3$ i $n = 500$ aleshores amb $\delta = 10^{-1}$ ja hauriem de veure cancelació en precisió simple. I en efecte, el resultat del càlcul és aproximadament -0.45, valor que no té sentit. Això és perque hem perdut totes les xifres significatives. Si $\delta = 5$ aleshores obtenim un resultat que discrepa del resultat correcte en un 10%, mentre que amb $\delta = 10$, la discrepància és d'un 2%.

Per observar cancelació en precisió doble hem de tenir $2(k+l) > 15$ ja que en precisió doble tenim 15 xifres significatives. Si $x = 1000$, $n = 500$ i $\delta = 10^{-5}$ ja tenim cancelació i el càlcul amb un sol bucle ens dóna un resultat negatiu. Si $\delta = 10^{-4}$ hi ha una discrepància del 16% respecte el valor correcte, mentre que si $\delta = 10^{-3}$, aquesta discrepància passa a ser del 0.3%.

Problema 4

Considerem la sèrie

$$S = \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6} \approx 1.644\,934\,066\,848\,226$$

Els programes `prob4a_fl.c` i `prob4a_do.c` calculen la suma dels termes de la sèrie fins al terme especificat en ordre decreixent, és a dir:

$$S_n = \left(... \left(\left(\left(1 + \frac{1}{2^2} \right) + \frac{1}{3^2} \right) + ... \right) + \frac{1}{(n-1)^2} \right) + \frac{1}{n^2}$$

en precisió simple i doble respectivament. En canvi els programes `prob4b_fl.c` i `prob4b_do.c` calculen les sumes parcials en ordre creixent, és a dir:

$$S_n = 1 + \left(\frac{1}{2^2} + \left(\frac{1}{3^2} + \left(... + \left(\frac{1}{(n-1)^2} + \frac{1}{n^2} \right) ... \right) \right) \right)$$

en precisió simple i doble respectivament.

La següent taula presenta les discrepàncies entre el valor de la sèrie S i les sumes parcials fins al terme n-èssim, calculades amb cadascun dels programes:

n	`prob4a_fl.c`	`prob4a_do.c`	`prob4b_fl.c`	`prob4b_do.c`
10^5	$2.0874 \cdot 10^{-4}$	$9.9999499841 \cdot 10^{-6}$	$1.002 \cdot 10^{-5}$	$9.9999500001 \cdot 10^{-6}$
10^6	$2.0874 \cdot 10^{-4}$	$9.999994564 \cdot 10^{-7}$	$1.08 \cdot 10^{-6}$	$9.999995001 \cdot 10^{-7}$
10^7	$2.0874 \cdot 10^{-4}$	$1.000009668 \cdot 10^{-7}$	$1.3 \cdot 10^{-7}$	$9.99999950 \cdot 10^{-8}$
10^8	$2.0874 \cdot 10^{-4}$	$9.0136514 \cdot 10^{-9}$	$1 \cdot 10^{-8}$	$9.9999999 \cdot 10^{-9}$
10^9	$2.0874 \cdot 10^{-4}$	$9.0136514 \cdot 10^{-9}$	$1 \cdot 10^{-8}$	$1.0000001 \cdot 10^{-9}$

Observem que el resultat de `prob4a_fl.c` no millora tot i afegir termes —de fet deixa de canviar a partir del terme 10^4—. I el mateix succeeix amb `prob4a_do.c` a partir del terme 10^8. En canvi els resultats dels programes `prob4b_fl.c` i `prob4b_do.c` si que van millorant com més termes anem sumant.

La causa dels estancaments dels programes que sumen en ordre decreixent és el nombre limitat de xifres significatives en la representació en punt flotant. En precisió simple hi ha 7 xifres significatives disponibles. El terme 10^4 de la sèrie contribueix 10^{-8}. Com que S és almenys 1, la diferència de xifres significatives és 8 i per tant els termes següents ja no contribueixen. En precisió doble tenim 15 xifres significatives, de manera que l'estancament té lloc a partir del terme 10^8, que contribueix 10^{-16}.

Els programes `prob4d1_fl.c` i `prob4d2_fl.c` proporcionen maneres alternatives de realitzar la suma de termes de S. No s'aconsegueix millorar el resultat que

proporciona sumar en ordre creixent, però sí que s'aconsegueix que la precisió en el resultat sigui independent de l'ordre de suma.

El programa `prob4d1_fl.c` agrupa els termes de la sèrie segons el seu ordre de magnitud en binari. Aleshores suma cada un d'aquests conjunts i finalment suma el resultat de cada un d'aquests conjunts. En detall, primer agrupa els termes en N conjunts S_k

$$A_k = \left\{ i \in \mathbb{N} \;\middle|\; \frac{1}{2^{k+1}} \leq \frac{1}{i^2} < \frac{1}{2^k} \right\}.$$

Seguidament realitza les sumes

$$S_k = \sum_{i \in A_k} \frac{1}{i^2}$$

i acaba fent la suma

$$S \approx \sum_{i=1}^{N} S_k. \tag{5}$$

D'aquesta manera, la suma final a (5), tot i ser en ordre decreixent[1] no es veu tant afectada per la pèrdua de precisió ja que consta de menys termes i amb diferències menors entre ells.

El programa `prob4d2_fl.c` utilitza l'algoritme de suma de Kahan, que és capaç de recuperar l'error comés al sumar cada terme, i el té en compte al sumar el terme següent —per a més detalls veure l'arxiu del programa—, aconseguint d'aquesta manera compensar l'error.

En la taula següent es mostren les discrepàncies dels resultats d'aquests dos programes respecte al valor exacte:

n	`prob4d1_fl.c`	`prob4d2_do.c`
10^5	$1.014 \cdot 10^{-5}$	$1.002 \cdot 10^{-5}$
10^6	$1.08 \cdot 10^{-6}$	$9.6 \cdot 10^{-7}$
10^7	$2.5 \cdot 10^{-7}$	$1.3 \cdot 10^{-7}$
10^8	$2.5 \cdot 10^{-7}$	$1 \cdot 10^{-8}$

Observem que amb `prob4d1_fl.c` tot i sumar en ordre decreixent, sumar els termes de magnitud similar de manera intermitja contribueix a alentir l'estancament del resultat: ara s'estanca partir de 10^7 termes, en comptes de 10^4.

Per un altra banda, `prob4d2_fl.c` ens dóna el mateix resultat que sumant en ordre creixent, tot i que la suma l'hem fet en ordre decreixent. Per tant l'algoritme de suma de Kahan ha compensat eficientment els errors comesos sense necessitat d'alterar l'ordre de sumació.

[1] Si suméssim de més petit a més gran, aquest algoritme en realitat estaria fent el mateix que fa `prob4b_fl.c` però de manera més complicada