Strojna pogreška kod izračunavanja vrijednosti funkcije

Petar Taler*

Sažetak. U članku se promatraju problemi koji nastaju prilikom izračunavanja vrijednosti nekih funkcija u aritmetici s pomičnim zarezom. Analizirano je nekoliko karakterističnih primjera i dan postupak kako umanjiti pogreške koje nastaju prilikom izračunavanja vrijednosti funkcija.

Ključne riječi: pogreške, vrijednosti funkcije, aritmetika s pomičnim zarezom.

Computer error in calculating function value

Abstract. Problems arising when calculating values of some floating-point arithmetic functions are considered in this paper. Several typical examples are analyzed and the procedure is given on how to reduce errors arising in the calculation of function values.
Key words: errors, function values, floating point arithmetic.

1. Uvod

Brojevi se u računalu zapisuju u unaprijed zadanom formatu koji propisuje koliko se binarnih znamenaka (bitova) koristi za prikaz broja i kako se one interpretiraju. Stoga se u računalu može prikazati samo konačan podskup skupa cijelih odnosno realnih brojeva. Ako je za prikaz cijelog broja rezervirano n bitova, onda je na taj način moguće prikazati ili sve pozitivne cijele brojeve od 0 do $2^n - 1$ ili pozitivne i negativne cijele brojeve npr. od -2^{n-1} do $2^{n-1} - 1$. Brojevi izvan tih intervala nisu prikazivi u računalu.

Prikaz realnih brojeva u računalu predstavlja složeniji problem. Oni se mogu prikazati kao *brojevi s fiksnom točkom* ili kao *brojevi s pomičnom točkom*. Drugi način je prihvaćeniji i u nastavku ćemo ga podrazumijevati. Pri tome posebnu pažnju treba obratiti na računske operacije nad brojevima s pomičnom točkom, jer se kod njih mogu pojaviti neke neočekivane pogreške.

Primjer 1. *Promotrimo izračunavanje vrijednosti funkcije* $f : \mathbb{R}\backslash\{0\} \to \mathbb{R}$

$$f(x) = \frac{1 - \cos(x)}{x^2} \tag{1}$$

*Odjel za matematiku, Sveučilište u Osijeku, Trg Ljudevita Gaja 6, HR-31000 Osijek, email: petar@mathos.hr

u okolini nule uz pretpostavku da se računanje odvija na računalu koje radi sa 7 značajnih znamenki.

Za $x_0 = 5.4 \times 10^{-4}$ zaokruživanjem na 7 značajnih znamenki izračunajmo vrijednost funkcije f. Imamo:

$$\cos x_0 = 0.9999999,$$
$$1 - \cos x_0 = 0.0000001,$$
$$f(x_0) = \frac{1-\cos x_0}{x_0^2} = \frac{10^{-7}}{2.916\times 10^{-7}} \approx 0.343.$$

Kako je $\lim_{x\to 0} f(x) = 0.5$, primjećujemo da smo strojnim računanjem dobili rezultat koji nema niti jednu točnu decimalu. Zbog čega je došlo do tolike netočnosti? Da bismo mogli odgovoriti na ovo pitanje, moramo znati kako računala rade s realnim brojevima.

2. Prikaz realnih brojeva u računalu

Računalo je stroj koji radi s određenim (konačnim) brojem znamenaka i kao takvo ne može realne brojeve egzaktno prikazati. Brojeve zapisane u računalu nazivamo *strojnim brojevima*. Definiranje skupa strojnih brojeva počinje odabirom baze b brojevnog sustava i broja znamenki t koje će se koristiti za reprezentaciju brojeva. Odabirom brojeva b i t, strojni broj se definira kao

$$s = \pm z_0.z_1z_2\cdots z_{t-1}\cdot b^e \equiv \pm(z_0 + \frac{z_1}{b} + \frac{z_2}{b^2} + \cdots + \frac{z_{t-1}}{b^{t-1}})\cdot b^e, \tag{2}$$

gdje su $(z_0, \ldots, z_{t-1})$ znamenke strojnog broja iz skupa $\{0, 1, \ldots, b-1\}$, a eksponent e cijeli broj iz segmenta $[e_{min},\ e_{max}]$. Dio $z_0.z_1z_2\cdots z_{t-1}$ nazivamo *mantisa* strojnog broja i kažemo da je ona normalizirana ako je $z_0 \neq 0$. Dakle, prikaz strojnog broja potpuno je određen četvorkom parametara (b, t, e_{min}, e_{max}). Većina današnjih računala koristi binarni brojevni sustav ($b = 2$) i u nastavku ćemo podrazumijevati upotrebu tog sustava.

Da bismo spremili normalizirani broj u računalo, podijelimo memorijsku riječ u tri dijela – polja. Kod 32-bitnih računala, riječ obično ima 32 bita (tip podatka `single`[1]), pa ćemo podjelu izvršiti na sljedeći način: 1 bit za predznak, 8 bitova za eksponent i 23 bita za mantisu. Bit za predznak je 0 ako je broj pozitivan, a 1 ako je broj negativan. Polje za eksponent ima 8 bitova i smješta se s *pomakom* od -127. Tako primjerice pohranjena vrijednost 1 znači da je eksponent $1 - 127 = -126$, dok pohranjena vrijednost 200 daje eksponent $200 - 127 = 73$. Budući su najmanji i najveći eksponent rezervirani za posebnu upotrebu, ovim načinom možemo dobiti eksponente između -126 i 127. Preostala 23 bita koriste se za smještaj decimalnog dijela mantise, jer je kod normaliziranih binarnih brojeva uvijek $b_0 = 1$, pa ga nema potrebe spremati.

Pogledajmo sada koji je najveći, odnosno najmanji broj koji se na ovaj način može zapisati u računalo. Najveći eksponent je 127, pa uzimanjem svih binarnih znamenki mantise jednakih 1 dobijemo najveću vrijednost:

$$(1 + 2^{-1} + 2^{-2} + \cdots + 2^{-23}) \times 2^{127} = (2 - 2^{-23}) \times 2^{127} \approx 3.4028 \times 10^{38}.$$

[1] jedan od tipova podataka u IEEE standardu, vidi primjerice [6]

Najmanji pozitivni cijeli broj dobijemo ako uzmemo $z_1 = z_2 = \cdots = z_3 = 0$ i najmanji eksponent, -126. Dobivena vrijednost je

$$(1 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 0 \cdot 2^{-23}) \times 2^{-126} = 2^{-126} \approx 1.1755 \times 10^{-38}.$$

3. Algoritam za pohranjivanje realnih brojeva u računalo

Neka je zadan realni broj $x = x_c + x_r$, pri čemu je x_c cijeli dio broja, a x_r decimalni dio, takav da je $0 \leq x_r < 1$. Izradimo algoritam koji računa binarnu reprezentaciju brojeva x_c i x_r. Koristit ćemo operaciju **div** za cjelobrojno dijeljenje, **mod** za dobivanje ostatka kod cjelobrojnog dijeljenja i **int** za zaokruživanje realnog broja do cijelog broja u smjeru nule. Sa l_{max} označit ćemo najveći dopušteni broj binarnih znamenaka u binarnoj mantisi.

Algoritam 1.

```
y=x_c
k=-1
ponavljati
    k=k+1
    a(k)=mod(y,2)
    y=div(y,2)
sve dok je y<>0
z=x_r
l=0
ponavljati
    l=l+1
    zz=2*z
    b(l)=int(zz)
    z=zz-b(l)
sve dok je z<>0 i l<=l_max
```

Pri tome su `a(k)` binarne znamenke cjelobrojnog dijela x_c, a `b(l)` binarne znamenke decimalnog dijela x_r broja x.

Primjer 2. *Upotrijebimo Algoritam 1 da bi izračunali binarnu reprezentaciju brojeva a)* 8.25 *i b)* 26.1 *uz* $l_{max} = 10$.

a)

$$x = 8.25 \Rightarrow x_c = 8, x_r = 0.25$$

k	0	1	2	3
a(k)	**0**	**0**	**0**	**1**
y	4	2	1	0

l	1	2
zz	0.5	1
b(l)	**0**	**1**
z	0.5	0

$$x = (1000.01)_2.$$

b)

$$x = 26.1 \Rightarrow x_c = 26, x_r = 0.1$$

k	0	1	2	3	4
a(k)	**0**	**1**	**0**	**1**	**1**
y	13	6	3	1	0

l	1	2	3	4	5	6	7	8	9	10
zz	0.2	0.4	0.8	1.6	1.2	0.4	0.8	1.6	1.2	0.4
b(l)	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**
z	0.2	0.4	0.8	0.6	0.2	0.4	0.8	0.6	0.2	0.4

$$x = (11010.0001100110)_2.$$

Primijetimo da je u slučaju *a) ponavljati - dok je* petlja za računanje decimalnog dijela prekinuta zbog ispunjenja uvjeta $z = 0$, dok je u slučaju *b)* došlo do ispunjenja uvjeta $l > l_{max}$. Dakle, u slučaju *b)* broj se ne može predstaviti sa zadanim maksimalnim brojem binarnih znamenaka.

Ako se realni broj x može bez greške pohraniti u računalu, nazivamo ga **egzaktno reprezentabilnim**. Ako broj nije egzaktno reprezentabilan, on se prije pohranjivanja u računalo mora zaokružiti. Kao što smo pokazali, broj $(26.1)_{10}$ nije egzaktno reprezentabilan jer je[2]

$$(26.1)_{10} = (11010.0\dot{0}01\dot{1})_2.$$

Nakon normalizacije dobije se

$$x = (1.10100\dot{0}01\dot{1})_2 \times 2^4.$$

Budući koristimo mantisu duljine 23 bita, broj se zaokružuje na 23 znamenke iza decimalne točke i sprema u prostor za mantisu, pa je reprezentacija tog broja u računalu

$$\hat{x} = 1.10100001100110011001100 \times 2^4.$$

4. Pogreška aproksimacije i strojni epsilon

Apsolutna pogreška aproksimacije realnog broja x strojnim brojem $\hat{x}$ definira se kao $\Delta\hat{x} := |x - \hat{x}|$, dok se relativna pogreška definira s

$$\delta\hat{x} := \frac{|x - \hat{x}|}{|x|}.$$

U radu s normaliziranim strojnim brojevima pogrešku aproksimacije je prirodno mjeriti u *jedinicama na posljednjem mjestu* - (engl. units in the last place – **ulp**). Ako je primjerica zadana baza $b = 10$ i $t = 3$ znamenke u mantisi, te ako je rezultat računanja u tom sustavu jednak $\hat{x} = 3.14 \times 10^{-2}$, dok je točan rezultat $x = 0.0312 = 3.12 \times 10^{-2}$, reći ćemo da je učinjena pogreška od 2 ulp-a.

[2] oznakom "$\dot{}$" ograđujemo dio decimalnog broja koji se ponavlja u beskonačnost, primjerice $1.2\dot{3}4\dot{5} = 1.2345345345\ldots$

Ukoliko je rezultat računanja strojnog broja najbliži točnom rezultatu, očito je da je apsolutna pogreška aproksimacije najviše 1/2 ulp-a. Relativna pogreška koja odgovara apsolutnoj pogrešci od 1/2 ulp-a naziva se **strojni epsilon** i iznosi

$$\epsilon = \frac{1}{2}\beta^{-t+1}.$$

Uočimo da u binarnom sustavu ($\beta = 2$) vrijedi

$$\epsilon = 2^{-t}.$$

5. Aritmetičke operacije sa strojnim brojevima

Aritmetičke operacije u sustavu strojnih brojeva vrše se s konačnim brojem znamenaka. Kod operacija zbrajanja i oduzimanja dvaju normaliziranih brojeva s različitim eksponentima potrebno je najprije *poravnati* eksponente operanada, a zatim izvršiti operaciju nad mantisama, koristeći ograničen broj znamenaka. Tako se manjem broju eksponent poveća, a decimalni zarez mu se pomakne udesno. Pri tome se nepovratno izgubi onoliko znamenaka koliko mjesta se pomaknuo decimalni zarez. Taj gubitak predstavlja opasan izvor grešaka pri zbrajanju i oduzimanju.

Ilustrirajmo to na primjeru zbrajanja brojeva $x = 1.32 \times 10^4$ i $y = 9.15 \times 10^1$ u sustavu s $t = 3$ znamenke u mantisi i bazom $\beta = 10$. Točan zbroj je $x + y = 1.32915 \times 10^4$ koji ispravno zaokružen na 3 znamenke daje 1.33×10^4. Pogledajmo što se događa kod strojnog zbrajanja tih brojeva u troznamenkastoj aritmetici s pomičnom točkom:

$$\begin{array}{rr} & 1.32000 \times 10^4 \\ + & 0.00915 \times 10^4 \\ \hline & 1.32000 \times 10^4 \end{array}$$

Pri tome su izgubljene sve znamenke drugog pribrojnika i nastala je relativna pogreška od približno 1.38ϵ.

Vratimo se sada na problem iz Primjera 1. Vrijednost $\cos x_0 = 0.9999999$ ima sve točne znamenke, no problem je što $1 - \cos(x_0) = 0.0000001$ u 7-znamenkastoj aritmetici s pomičnom točkom ima samo jednu značajnu znamenku. Oduzimanje $1 - \cos(x_0)$ je egzaktno, ali to oduzimanje proizvodi rezultat koji je veličine kao i pogreška u $\cos(x_0)$. Drugim riječima, oduzimanje podiže značaj prethodne greške.

Ovaj problem možemo eliminirati tako da upotrijebivši formulu

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

izraz (1) transformiramo u

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

Kada na računalu izračunamo $f(5.4 \times 10^{-4})$ pomoću ove transformirane formule, dobit ćemo 0.5, što je točan rezultat na 7 značajnih znamenki.

Primjer 3. *Promotrimo rješavanje kvadratne jednadžbe*

$$ax^2 + bx + c = 0, \quad a \neq 0, \tag{3}$$

ako je $b^2 \gg 4ac$ (njena diskriminanta je puno veća od nule).

Postoje dva rješenja jednadžbe:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{4}$$

Pretpostavimo najprije da je $b > 0$. Budući da vrijedi $b^2 \gg 4ac$, imat ćemo $\sqrt{b^2 - 4ac} \approx b$, pa se prilikom izračunavanja po apsolutnoj vrijednosti manjeg rješenja (izbor predznaka "+") iz formule (4) dobiva rješenje $x_1 = 0$. Ovaj problem možemo eliminirati tako da najprije izračunamo po apsolutnoj vrijednosti veće rješenje primjenom formule

$$x_1 = \frac{-(b + \sqrt{b^2 - 4ac})}{2a}.$$

Drugo rješenje dobije se iz Viètеove formule

$$x_1 \cdot x_2 = \frac{c}{a},$$

i iznosi

$$x_2 = \frac{c}{a \cdot x_1}. \tag{5}$$

Sve prethodno navedeno može se provjeriti na rješavanju jednadžbe

$$x^2 + 1000x + 0.1 = 0 \tag{6}$$

na računalu sa 7 značajnih znamenaka. Ako sa $\tilde{x}_1$ i $\tilde{x}_2$ označimo strojne brojeve koji reprezentiraju vrijednosti rješenja jednadžbe (6), izborom predznaka "+" u formuli (4) dobiva se

$$\tilde{x}_1 = \frac{-1000 + \sqrt{1000000 - 0.4}}{2} = \frac{-1000 + 1000}{2} = 0.$$

Ako najprije izračunamo po apsolutnoj vrijednosti veće rješenje, a potom to rješenje uvrstimo u formulu (5), dobiva se

$$\tilde{x}_2 = \frac{-1000 - \sqrt{1000000 - 0.4}}{2} = \frac{-1000 - 1000}{2} = -1000,$$
$$\tilde{x}_1 = \frac{0.1}{1 \cdot (-1000)} = 0.0001,$$

što su točna rješenja na 7 značajnih znamenaka.

Sami proanalizirajte slučaj $b < 0$ u jednadžbi (4).

Primjer 4. *Promotrimo rješavanje Heronove formule za izračunavanje površine trokuta*

$$P = \sqrt{s(s-a)(s-b)(s-c)}, \quad \textit{gdje je } s = \tfrac{a+b+c}{2}, \tag{7}$$
$$\textit{u slučaju } a \approx b \gg c. \tag{8}$$

Iz uvjeta (8) slijedi $s \approx a \approx b$ i zbog toga dolazi do nekontroliranog poništavanja nekih međurezultata u izrazu (7). Kako bi bolje razumjeli ovaj problem sa s

označimo pravu vrijednost poluzbroja opsega trokuta $(a+b+c)/2$, a sa $\tilde{s}$ strojni broj koji reprezentatira tu vrijednost. Tada vrijedi

$$\tilde{s} = s(1+\delta)$$

pri čemu je δ relativna pogreška za $\tilde{s}$. Veličinu $\tilde{s}-a$ možemo zapisati u obliku

$$\tilde{s}-a = s-a+s\cdot\delta = (s-a)(1+\delta_1), \quad \delta_1 = \frac{s}{s-a}\delta,$$

što znači da je relativna pogreška δ_1 u veličini $\tilde{s}-a$ postala $\frac{s}{s-a}$ puta veća od vrijednosti pogreške δ u $\tilde{s}$. Ako je $s-a$ puno manji od s, onda je $\frac{s}{s-a}$ veliki broj, pa relativna pogreška δ_1 postaje jako velika.

Ovaj problem može se eliminirati označavanjem stranica trokuta tako da vrijedi $a \geq b \geq c$. Naime, ako u jednakosti

$$a+b+c = 2\cdot s$$

na lijevoj i desnoj strani redom oduzimamo: $2c$, $2a$, $2b$, dobivamo:

$$\begin{aligned} a+b-c &= 2\cdot(s-c)\\ b+c-a &= 2\cdot(s-a)\\ c+a-b &= 2\cdot(s-b). \end{aligned}$$

Koristeći ove jednakosti, izraz (7) transformira se u

$$P = \frac{1}{4}\sqrt{(a+b+c)(b+c-a)(c+a-b)(a+b-c)} \tag{9}$$

i time se eliminira problem poništavanja međurezultata.

Primjerice, za vrijednosti $a = 1000$, $b = 1000.001$, $c = 0.002$ imamo $s = 1000.0015$, a prava vrijednost površine trokuta iznosi $P \approx 0.866$. Računajući sa 7 značajnih znamenaka, formula (7) daje $\tilde{P}_1 \approx 1.441$, a formula (9) $\tilde{P}_2 \approx 0.866$.

Primjer 5. *Razmotrimo izračunavanje vrijednosti funkcije* $f : \mathbb{R}^2 \to \mathbb{R}$

$$f(x,y) = x^2 - y^2 \tag{10}$$

kada je $x \approx y$. *Usporedimo pogreške koja nastaju prilikom izračunavanja funkcije (10) i funkcije*

$$f(x,y) = (x-y)\cdot(x+y). \tag{11}$$

Ako sa ϵ označimo strojni epsilon, a sa $\tilde{x}$ i $\tilde{y}$ strojne brojeve koji reprezentiraju x i y znamo da je

$$\begin{aligned} \tilde{x}-\tilde{y} &= (x-y)(1+\delta_1), \quad |\delta_1| \leq \epsilon\\ \tilde{x}+\tilde{y} &= (x+y)(1+\delta_2), \quad |\delta_2| \leq \epsilon. \end{aligned}$$

Tada imamo

$$\begin{aligned} f(\tilde{x},\tilde{y}) = (\tilde{x}-\tilde{y})\cdot(\tilde{x}+\tilde{y}) &= [(x-y)(x+y)(1+\delta_1)(1+\delta_2)](1+\delta_3)\\ &= (x-y)(x+y)(1+\delta_1)(1+\delta_2)(1+\delta_3), \end{aligned}$$

pri čemu je δ_3 relativna pogreška množenja $(\tilde{x}-\tilde{y})\cdot(\tilde{x}+\tilde{y})$. Stoga je ukupna relativna pogreška:

$$\begin{aligned}\frac{(\tilde{x}-\tilde{y})\cdot(\tilde{x}+\tilde{y})-(x^2-y^2)}{x^2-y^2} &= (1+\delta_1)(1+\delta_2)(1+\delta_3)-1\\ &= \delta_1+\delta_2+\delta_3+\delta_1\delta_2+\delta_1\delta_3+\delta_2\delta_3+\delta_1\delta_2\delta_3\\ &\approx 3(\epsilon+\epsilon^2)=3\epsilon(1+\epsilon) \text{ (uz pretpostavku } \epsilon^3\approx 0).\end{aligned}$$

S druge strane je

$$\begin{aligned}\tilde{x}\cdot\tilde{x} &= x^2(1+\delta_4), \quad |\delta_4|\le\epsilon\\ \tilde{y}\cdot\tilde{y} &= y^2(1+\delta_5), \quad |\delta_5|\le\epsilon,\end{aligned}$$

pa imamo

$$\begin{aligned}f(\tilde{x},\tilde{y})=(\tilde{x}\cdot\tilde{x})-(\tilde{y}\cdot\tilde{y}) &= [x^2(1+\delta_4)-y^2(1+\delta_5)](1+\delta_6)\\ &= [(x^2-y^2)(1+\delta_4)+(\delta_4-\delta_5)y^2](1+\delta_6),\end{aligned}$$

pri čemu je δ_6 relativna pogreška oduzimanja $(\tilde{x}\cdot\tilde{x})-(\tilde{y}\cdot\tilde{y})$. Ukupna relativna pogreška za ovaj slučaj iznosi

$$\frac{(\tilde{x}\cdot\tilde{x})-(\tilde{y}\cdot\tilde{y})-(x^2-y^2)}{(x^2-y^2)}=\delta_4+\delta_6+\delta_4\delta_6+(\delta_4-\delta_5)(1+\delta_6)\frac{y^2}{x^2-y^2}.$$

Kada je $x\approx y$, izraz $\frac{y^2}{x^2-y^2}$ postaje velik, pa zaključujemo da je formula (11) pogodnija za strojno računanje, budući da relativna pogreška kod nje ne ovisi o vrijednosti koeficijenata.

Primjerice, ako uzmemo vrijednosti $x=7500001$ i $y=7500000$, točna vrijednost funkcije (10), odnosno (11) iznosi $f(x,y)=1.5000001\times 10^7$. Računajući na računalu sa 7 značajnih znamenaka, formulom (10) dobije se $\tilde{f}_1(x,y)=2\times 10^7$, a formulom (11) $\tilde{f}_2(x,y)=1.5\times 10^7$, što je točan rezultat na 7 značajnih znamenaka.

Literatura

[1] Z. DRMAČ, *Numerička matematika i računala*, Matematičko - fizički list, **49**(1998/99), 212-219

[2] D. GOLDBERG, *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Comput. Surv., **23**(1991), 5-48

[3] V. HARI I DR., *Numerička analiza: osnovni udžbenik*, skripta PMF – Matematičkog odjela, Zagreb, 2003., `http://web.math.hr/nastava/unm/udzbenik/num_anal.pdf`

[4] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[5] M. JURAK, *Reprezentacija brojeva u računalu*, skripta PMF – Matematičkog odjela, Zagreb, `http://web.math.hr/nastava/ppm1/ppm_fpb.pdf`

[6] M. L. OVERTON, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, 2001.