Uvod u računarstvo
6. predavanje

Saša Singer

singer@math.hr

web.math.hr/~singer

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Prikaz realnih brojeva — “floating–point” standard:
  - osnovni oblik “floating–point” prikaza — mantisa i eksponent,
  - greške zaokruživanja u prikazu,
  - pojam “jedinične greške zaokruživanja”,
  - IEEE standard — tipovi: single, double, extended,
- Greške zaokruživanja u aritmetici realnih brojeva:
  - greške zaokruživanja osnovnih aritmetičkih operacija
  - opasno ili “katastrofalno” kraćenje,
  - “širenje” grešaka zaokruživanja, stabilni i nestabilni algoritmi,

Sadržaj predavanja (nastavak)

- Primjeri širenja grešaka zaokruživanja i izbjegavanja nestabilnosti:
  - parcijalne sume harmonijskog reda,
  - korijeni kvadratne jednadžbe.

Uvod u prikaz realnih brojeva

Kako pohraniti "jako velike" ili "jako male brojeve?
Recimo (dekadski pisano):

$$67800000000.0 \qquad 0.000002078$$

Koristimo tzv. znanstvenu notaciju u kojoj

- prvo pišemo vodeće značajne znamenke broja,
- a zatim pišemo faktor koji ima oblik baza na odgovarajući eksponent, tj. potenciju baze.

Uz dogovor da vodeći dio bude između 1 i 10 (strogo ispod), to izgleda ovako:

$$6.78 \cdot 10^{10} \qquad 2.078 \cdot 10^{-6}.$$

Prikaz realnih brojeva

U računalu se binarni zapis realnog broja pohranjuje u znanstvenom formatu:

$$broj = mantisa \cdot 2^{eksponent}.$$

Mantisa se uobičajeno (postoje iznimke!) pohranjuje u tzv. normaliziranom obliku, tj.

$$1 \leq mantisa < (10)_2.$$

I za pohranu mantise i za pohranu eksponenta rezervirano je konačno mnogo binarnih znamenki. Posljedice:

- prikaziv je samo neki raspon realnih brojeva,
- niti svi brojevi unutar prikazivog raspona nisu prikazivi (mantisa predugačka) $\Longrightarrow$ zaokruživanje.

Prikaz realnih brojeva (nastavak)

Primjer: Znanstveni prikaz binarnih brojeva:

$$1010.11 = 1.01011 \cdot 2^{3}$$
$$0.0001011011 = 1.01011 \cdot 2^{-4}$$

Primijetite da se vodeća jedinica u normaliziranom obliku ne mora pamtiti (ako je broj $\neq 0$).

- Taj bit se može upotrijebiti za pamćenje dodatne znamenke mantise.

Tada se vodeća jedinica zove skriveni bit (engl. hidden bit) — jer se ne pamti.

Ipak ovo je samo pojednostavljeni prikaz realnih brojeva.

Stvarni prikaz realnih brojeva

Najznačajnija promjena obzirom na pojednostavljeni prikaz:

- eksponent se prikazuje u “zamaskiranoj” ili “pomaknutoj” formi (engl. biased form).

To znači da se stvarnom eksponentu se dodaje konstanta takva da je “pomaknuti” eksponent uvijek pozitivan.

Ta konstanta ovisi o broju bitova za eksponent i bira se tako da je prikaziva recipročna vrijednost najmanjeg normaliziranog broja.

Takav “pomaknuti” eksponent naziva se karakteristika, a normaliziranu mantisu neki zovu i signifikand.

Oznake

Oznake:

- Crveno — duljina odgovarajućeg polja (u bitovima), bitove brojimo od 0 zdesna nalijevo (kao i obično),
- p — predznak: 0 za pozitivan broj, 1 za negativan broj,
- k — karakteristika,
- m — mantisa (signifikand).
- Najznačajniji bit u odgovarajućem polju je najljeviji.
- Najmanje značajan bit u odgovarajućem polju je najdesniji.

Stvarni prikaz tipa single

"Najkraći" realni tip je tzv. realni broj jednostruke točnosti — u C-u poznat kao `float`.
On ima sljedeća svojstva:

- duljina: 4 byte-a (32 bita), podijeljen u tri polja.

1	8	23
p	k	m

- u mantisi se ne pamti vodeća jedinica ako je broj normaliziran,
- stvarni eksponent broja e, $e \in \{-126, \ldots, 127\}$,
- karakteristika $k = e + 127$, tako da je $k \in \{1, \ldots, 254\}$,
- karakteristike $k = 0$ i $k = 255$ koriste se za "posebna stanja".

Stvarni prikaz tipa single (nastavak)

Primjer: Broj $(10.25)_{10}$ prikažite kao broj u jednostrukoj točnosti.

$$
\begin{aligned}
(10.25)_{10} &= \left(10 + \frac{1}{4}\right)_{10} = (10 + 2^{-2})_{10} \\
&= (1010.01)_2 = {\color{red}1}.01001 \cdot 2^3.
\end{aligned}
$$

Prema tome je:

$$
\begin{aligned}
p &= 0 \\
k &= e + 127 = (130)_{10} = (2^7 + 2^1)_{10} = 1000\ 0010 \\
m &= 0100\ 1000\ 0000\ 0000\ 0000\ 000
\end{aligned}
$$

Prikazi nule: $k = 0$, $m = 0$

Realni broj nula ima dva prikaza: mantisa i karakteristika su joj nula, a predznak može biti 0 "pozitivna nula" ili 1 "negativna nula".

Ta dva prikaza nule su:

$$+0 = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$$

$$-0 = 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$$

Smatra se da su vrijednosti ta dva broja jednake (kad se uspoređuju).

Denormalizirani brojevi: $k = 0$, $m \neq 0$

Ako je $k = 0$, a postoji barem jedan znak mantise koji nije nula, onda se kao eksponent uzima -126. Mantisa takvog broja nije normalizirana i počinje s $0.m$.

Takvi brojevi zovu se denormalizirani brojevi.

Primjer: Kako izgleda prikaz realnog broja

$$0.000\ 0000\ 0000\ 0000\ 0000\ 1011 \cdot 2^{-126}?$$

Rješenje:

$$\begin{aligned} p &= 0 \\ k &= 0000\ 0000 \\ m &= 000\ 0000\ 0000\ 0000\ 0000\ 1011 \end{aligned}$$

Plus i minus beskonačno: $k = 255$, $m = 0$

Ako je $k = 255$, a mantisa jednaka 0, onda

- $p = 0$ — prikaz $+\infty$, skraćena oznaka `+Inf`,
- $p = 1$ — prikaz $-\infty$, skraćena oznaka `-Inf`.

Primjer: Prikaz broja $+\infty$ ($-\infty$) je

$$p = 0 \qquad (p = 1)$$

$$k = 1111\ 1111$$

$$m = 000\ 0000\ 0000\ 0000\ 0000\ 0000$$

Nije broj: $k = 255$, $m \neq 0$

Ako je $k = 255$ i postoji bar jedan bit mantise različit od nule, onda je to signal da se radi o pogrešci (recimo dijeljenje s nulom, vađenje drugog korijena iz negativnog broja i sl.)

Tada se takva pogreška kodira znakom za Not a Number ili, skraćeno, s `NaN`:

$$p = 0$$

$$k = 1111\ 1111$$

$$m = 000\ 0000\ 0000\ 0101\ 0000\ 0000$$

Greške zaokruživanja

Postoje realni brojevi koje ne možemo egzaktno spremiti u računalo, čak i kad su unutar prikazivog raspona brojeva. Takvi brojevi imaju predugačku mantisu.

Primjer: Realni broj

$$a = 1.0001\ 0000\ 1000\ 0011\ 1001\ 0111$$

ne može se spremiti u realni broj jednostruke preciznosti `float` u C-u koji ima $23 + 1$ znakova mantise, jer ima 25 znakova mantise.
Procesor tada pronalazi dva najbliža prikaziva susjeda broju a takva da vrijedi

$$b < a < c.$$

Greške zaokruživanja (nastavak)

U našem primjeru je:

$$a = 1.0001\ 0000\ 1000\ 0011\ 1001\ 0111$$

$$b = 1.0001\ 0000\ 1000\ 0011\ 1001\ 011$$

$$c = 1.0001\ 0000\ 1000\ 0011\ 1001\ 100$$

Nakon toga, zaokružuje rezultat. Zaokruživanje može biti:

- prema najbližem broju (standardno, engl. default za IA-32 procesore) – ako su dva susjeda jednako udaljena od a, izabire parni od ta dva broja,
- prema dolje, tj. prema $-\infty$,
- prema gore, tj. prema ∞,
- prema nuli, tj. odbacivanjem "viška" znamenki.

Greške zaokruživanja (nastavak)

Standardno zaokruživanje u našem primjeru:

$$a = 1.0001\ 0000\ 1000\ 0011\ 1001\ 0111\textcolor{red}{1}$$

$$b = 1.0001\ 0000\ 1000\ 0011\ 1001\ \textcolor{red}{011}$$

$$c = 1.0001\ 0000\ 1000\ 0011\ 1001\ \textcolor{red}{100}$$

Ovdje su b i c jednako udaljeni od a, pa je zaokruženi a jednak c.

Jedinična greška zaokruživanja

Dakle, ako je $x \in \mathbb{R}$ unutar raspona brojeva prikazivih u računalu, onda se umjesto x sprema zaokruženi broj prikazivi $f\ell(x)$.
Time smo napravili grešku zaokruživanja $\leq \frac{1}{2}$ “zadnjeg bita” mantise i taj broj se zove

- jedinična greška zaokruživanja (engl. unit roundoff).

Standardna oznaka je u. Za `float` je $u = 2^{-24} \approx 5.96 \cdot 10^{-8}$.
Vrijedi

$$f\ell(x) = (1 + \varepsilon)x, \quad |\varepsilon| \leq u,$$

gdje je ε relativna greška napravljena tim zaokruživanjem.
Dakle, imamo vrlo malu relativnu grešku.

Prikaz brojeva jednostruke točnosti — sažetak

IEEE tip single = `float` u C-u:

1	8	23
p	k	m

- Vrijednost broja je

$$v = \begin{cases} (-1)^p * 2^{(k-127)} * (1.m) & \text{ako je } 0 < k < 255, \\ (-1)^p * 2^{(-126)} * (0.m) & \text{ako je } k = 0 \text{ i } m \neq 0, \\ (-1)^p * 0 & \text{ako je } k = 0 \text{ i } m = 0, \\ (-1)^p * \mathrm{Inf} & \text{ako je } k = 255 \text{ i } m = 0, \\ \mathrm{NaN} & \text{ako je } k = 255 \text{ i } m \neq 0. \end{cases}$$

Raspon tipa float

Najveći prikazivi pozitivni broj je $\texttt{FLT\_MAX} \approx 3.40282 \cdot 10^{38}$

$$p = 0$$

$$k = 1111\ 1110$$

$$m = 111\ 1111\ 1111\ 1111\ 1111\ 1111$$

Najmanji prikazivi normalizirani pozitivni broj je $\texttt{FLT\_MIN} \approx 1.17549 \cdot 10^{-38}$

$$p = 0$$

$$k = 0000\ 0001$$

$$m = 000\ 0000\ 0000\ 0000\ 0000\ 0000$$

Raspon tipa float

Najmanji prikazivi denormalizirani pozitivni broj je $\approx 1.4013 \cdot 10^{-45}$

$$
\begin{aligned}
p &= 0 \\
k &= 0000\ 0001 \\
m &= 000\ 0000\ 0000\ 0000\ 0000\ 0001
\end{aligned}
$$

Stvarni prikaz tipa double

"Srednji" realni tip je tzv. realni broj dvostruke točnosti — u C-u poznat kao `double`.
On ima sljedeća svojstva:

- Duljina: 8 byte-a (64 bita), podijeljen u tri polja.

1	11	52
p	k	m

- u mantisi se ne pamti vodeća jedinica ako je broj normaliziran,
- stvarni eksponent broja e, $e \in \{-1022, \dots, 1023\}$,
- karakteristika $k = e + 1023$, tako da je $k \in \{1, \dots, 2046\}$,
- karakteristike $k = 0$ i $k = 2047$ — "posebna stanja".

Prikaz brojeva dvostruke točnosti — sažetak

IEEE tip double = `double` u C-u:

1	11	52
p	k	m

- Vrijednost broja je

$$v = \begin{cases} (-1)^p * 2^{(k-1023)} * (1.m) & \text{ako je } 0 < k < 2047, \\ (-1)^p * 2^{(-1022)} * (0.m) & \text{ako je } k = 0 \text{ i } m \neq 0, \\ (-1)^p * 0 & \text{ako je } k = 0 \text{ i } m = 0, \\ (-1)^p * \mathrm{Inf} & \text{ako je } k = 2047 \text{ i } m = 0, \\ \mathrm{NaN} & \text{ako je } k = 2047 \text{ i } m \neq 0. \end{cases}$$

Tip extended

Stvarno računanje (na IA–32) se obično radi u “proširenoj” točnosti — u C-u možda dohvatljiv kao `long double`.
On ima sljedeća svojstva:

- Duljina: 10 byte-a (80 bita), podijeljen u četiri polja.

1	15	1	63
p	k	i	m

- u mantisi se pamti vodeći bit i mantise,
- stvarni eksponent broja e, $e \in \{-16382, \ldots, 16383\}$,
- karakteristika $k = e + 16383$, tako da je $k \in \{1, \ldots, 32766\}$,
- karakteristike $k = 0$ i $k = 32767$ — “posebna stanja”.

Prikaz brojeva proširene točnosti — sažetak

IEEE tip extended:

1	15	1	63
p	k	i	m

- Vrijednost broja je

$$v = \begin{cases} (-1)^p * 2^{(k-16383)} * (i.m) & \text{ako je } 0 \leq k < 32767, \\ (-1)^p * \mathrm{Inf} & \text{ako je } k = 32767 \text{ i } m = 0, \\ \mathrm{NaN} & \text{ako je } e = 32767 \text{ i } m \neq 0. \end{cases}$$

Aritmetika računala

Aritmetika računala nije egzaktna. Za aritmetičku operaciju $\circ$, gdje je $\circ$ jedna od operacija $+$, $-$, $*$, $/$, zahtijeva se samo da ima malu relativnu grešku, tj.

$$fl(x \circ y) = (1 + \varepsilon)(x \circ y),$$

pri čemu je fl rezultat operacije dobiven računalom, a ε mali pozitivan broj.
Za aritmetiku računala ne vrijedi:

- asocijativnost zbrajanja i množenja,
- distributivnost množenja prema zbrajanju.

Jedino vrijedi:

- komutativnost za zbrajanje i množenje.

Primjer neasocijativnosti zbrajanja

Primjer. Asocijativnost zbrajanja u računalu ne vrijedi.

Znamo (odn. uskoro ćete znati) da je tzv. harmonijski red

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{i} + \cdots$$

divergentan, tj. suma mu je “beskonačna”.

No, nitko nas ne spriječava da računamo konačne početne komade ovog reda, tj. njegove parcijalne sume

$$S_n := 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}.$$

A kojim redom zbrajamo? (Zbrajanje je binarna operacija!)

Primjer neasocijativnosti zbrajanja (nastavak)

U realnim brojevima je potpuno svejedno kojim poretkom zbrajanja računamo ovu sumu, jer vrijedi asocijativnost.

$$a + (b + c) = (a + b) + c = a + b + c.$$

Uostalom, sam zapis izraza bez zagrada

$$S_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}$$

već "podrazumijeva" asocijativnost. U suprotnom, morali bismo zagradama naglasiti poredak operacija.

Ovdje imamo točno $n - 1$ binarnih operacija zbrajanja, i možemo ih napraviti kojim redom hoćemo.

Primjer neasocijativnosti zbrajanja (nastavak)

Drugim riječima, u prethodni izraz za S_n

- možemo rasporediti zagrade na bilo koji način, samo da svi plusevi budu "binarni", tj. zbrajaju dva objekta, a objekt je broj ili (podizraz u zagradama).

Na pr., zbrajanju "unaprijed" odgovara raspored zagrada

$$S_{n,1} := \left(\cdots \left(\left(1 + \frac{1}{2} \right) + \frac{1}{3} \right) + \cdots + \frac{1}{n-1} \right) + \frac{1}{n},$$

a zbrajanju "unatrag" odgovara raspored zagrada

$$S_{n,2} := 1 + \left(\frac{1}{2} + \left(\frac{1}{3} + \cdots + \left(\frac{1}{n-1} + \frac{1}{n} \right) \cdots \right) \right).$$

Primjer neasocijativnosti zbrajanja (nastavak)

Koliko takvih rasporeda zagrada ima — bit će napravljeno u Kombinatorici. Bitno je da svi daju isti rezultat.

Komutativnost nam uopće ne treba. Ako i nju iskoristimo, dobivamo još puno više načina za računanje ove sume, i svi, naravno, opet daju isti rezultat.

Izračunajmo aritmetikom računala navedene dvije sume

- $S_{n,1}$ — unaprijed, i
- $S_{n,2}$ — unatrag,

za $n = 1\,000\,000$, u tri standardne IEEE točnosti single, double i extended. Preciznije, koristimo ova tri tipa za prikaz brojeva, uz pripadne artimetike za računanje.

Primjer neasocijativnosti zbrajanja (nastavak)

Uz skraćene oznake S_1 i S_2 za varijable u kojima zbrajamo pripadne sume, odgovarajući algoritmi za zbrajanje su:

- unaprijed

$$
\begin{aligned}
S_1 &:= 1, \\
S_1 &:= S_1 + \frac{1}{i}, \quad i = 2, \ldots, n,
\end{aligned}
$$

- unatrag

$$
\begin{aligned}
S_2 &:= \frac{1}{n}, \\
S_2 &:= \frac{1}{i} + S_2, \quad i = n-1, \ldots, 1.
\end{aligned}
$$

Dakle, zaista ne koristimo komutativnost zbrajanja.

Primjer neasocijativnosti zbrajanja (nastavak)

Dobiveni rezultati za sume S_1, S_2 i pripadne relativne greške su:

tip i suma	vrijednost	rel. greška
single S_1	14.3573579788208008	$2.45740E{-}0003$
single S_2	14.3926515579223633	$5.22243E{-}0006$
double S_1	14.3927267228647810	$6.54899E{-}0014$
double S_2	14.3927267228657545	$-2.14449E{-}0015$
extended S_1	14.3927267228657234	$1.91639E{-}0017$
extended S_2	14.3927267228657236	$-1.08475E{-}0018$

Slovo E u brojevima zadnjeg stupca znači "puta 10 na", pa je, na pr., $-1.08475E{-}0018 = -1.08475 \times 10^{-18}$.

Primjer neasocijativnosti zbrajanja (nastavak)

Izračunate vrijednosti S_1 i S_2 su različite (u sve tri točnosti). Dakle, zbrajanje brojeva u aritmetici računala očito nije asocijativno.

Primijetite da, u sve tri točnosti, zbrajanje unatrag S_2 daje nešto točniji rezultat. To nije slučajno.
Svi brojevi koje zbrajamo su istog predznaka pa zbroj stalno raste, bez obzira na poredak zbrajanja.

- Kad zbrajamo unatrag — od manjih brojeva prema većim, zbroj se pomalo "nakuplja".
- Obratno, kad zbrajamo unaprijed — od velikih brojeva prema manjim, zbroj puno brže naraste. Onda mali dodani član jedva utječe na rezultat (tj. dobar dio znamenki pribrojnika nema utjecaj na sumu).

Primjer katastrofalnog kraćenja

Zakruživanjem ulaznih podataka dolazi do male relativne greške. Kako ona može utjecati na konačan rezultat?

Primjer. Uzmimo realnu aritmetiku "računala" u bazi 10. Za mantisu (značajni dio) imamo $t = 4$ dekadske znamenke, a za eksponent $s = 2$ znamenke (što nije bitno). Neka je

$$x = 8.8866 = 8.8866 \times 10^0,$$
$$y = 8.8844 = 8.8844 \times 10^0.$$

Umjesto brojeva x i y (koji nisu prikazivi), u "memoriju" spremamo brojeve $f\ell(x)$ i $f\ell(y)$, pravilno zaokružene na $t = 4$ znamenke

$$f\ell(x) = 8.887 \times 10^0,$$
$$f\ell(y) = 8.884 \times 10^0.$$

Primjer katastrofalnog kraćenja (nastavak)

Ovim zaokruživanjem smo napravili malu relativnu grešku (ovdje je $u = 5 \times 10^{-5}$).

Razliku $f\ell(x) - f\ell(y)$ računamo tako da izjednačimo eksponente (što već jesu), oduzmemo značajne dijelove (mantise), pa normaliziramo

$$\begin{aligned} f\ell(x) - f\ell(y) &= 8.887 \times 10^0 - 8.884 \times 10^0 \\ &= 0.003 \times 10^0 = 3.??? \times 10^{-3}. \end{aligned}$$

Kod normalizacije, zbog pomaka "ulijevo", pojavljuju se

- $? =$ znamenke koje više ne možemo restaurirati (ta informacija se izgubila).

Što sad?

Primjer katastrofalnog kraćenja (nastavak)

Računalo radi isto što bismo i mi napravili:

- na ta mjesta ? upisuje 0.

Razlog: da rezultat bude točan, ako su ulazni brojevi točni. Dakle, ovo oduzimanje je egzaktno i u aritmetici računala.

Konačni rezultat je $f\ell(x) - f\ell(y) = 3.000 \times 10^{-3}$.

Pravi rezultat je

$$\begin{aligned} x - y &= 8.8866 \times 10^0 - 8.8844 \times 10^0 \\ &= 0.0022 \times 10^0 = 2.2 \times 10^{-3}. \end{aligned}$$

Već prva značajna znamenka u $f\ell(x) - f\ell(y)$ je pogrešna, a relativna greška ogromna! Uočite da je ta znamenka (3) ujedno i jedina koja nam je ostala — sve ostalo se skratilo!

Primjer katastrofalnog kraćenja (nastavak)

Prava katastrofa se događa ako $3.??? \times 10^{-3}$ uđe u naredna zbrajanja (oduzimanja), a onda se skrati i ta trojka!

Uočite da je oduzimanje $f\ell(x) - f\ell(y)$ bilo egzaktno (a egzaktno je i u aritmetici računala), ali rezultat je pogrešan.

Krivac, očito, nije oduzimanje (kad je egzaktno).

- Uzrok su polazne greške u operandima.

Ako njih nema, tj. ako su operandi egzaktni,

- i dalje (naravno) dolazi do kraćenja,
- ali je rezultat (uglavnom, a po IEEE standardu sigurno) egzaktan,

pa se ovo kraćenje zove benigno kraćenje.

Kvadratna jednadžba

Uzmimo da treba riješiti (realnu) kvadratnu jednadžbu

$$ax^2 + bx + c = 0,$$

gdje su a, b i c zadani i $a \neq 0$.

Matematički gledano, problem je lagan: imamo 2 rješenja

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

Numerički gledano, problem je mnogo izazovniji:

- ni uspješno računanje po ovoj formuli,
- ni točnost izračunatih korijena,

ne možemo uzeti "zdravo za gotovo".

Kvadratna jednadžba (nastavak)

Primjer: $x^2 - 56x + 1 = 0$. U aritmetici s 5 decimala dobijemo

$$x_1 = \frac{56 - \sqrt{3132}}{2} = \frac{56 - 55.964}{2} = 0.018000,$$

$$x_2 = \frac{56 + \sqrt{3132}}{2} = \frac{56 + 55.964}{2} = 55.982.$$

Točna rješenja su

$$x_1 = 0.0178628\ldots \quad \text{i} \quad x_2 = 55.982137\ldots\,.$$

Manji od ova dva korijena ima samo dvije točne znamenke (kraćenje).

Kvadratna jednadžba (popravak)

Prvo izračunamo većeg po apsolutnoj vrijednosti, po formuli

$$x_2 = \frac{-(b + \operatorname{sign}(b)\sqrt{b^2 - 4ac})}{2a},$$

a manjeg po apsolutnoj vrijednosti, izračunamo iz

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

(Vieta), tj.

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

Opasnog kraćenja za x_1 više nema!

Kvadratna jednadžba (nastavak)

Ovo je bila samo jedna, od (barem) tri "opasne" točke za računanje. Preostale dvije su:

- "kvadriranje" pod korijenom — mogućnost za overflow. Rješenje — "skaliranjem".
- oduzimanje u diskriminanti (kraćenje) — nema jednostavnog rješenja.
- To je odraz nestabilnosti problema, jer tad imamo dva bliska korijena koji su osjetljivi na male perturbacije koeficijenata (na pr. pomak c "gore–dolje").