Sadržaj

POGLAVLJE 1

Pogreške u postupcima analize

1.1 Uzroci pogrešaka

Prilikom bilo kakvog postupka računanja, odnosno primjene nekog algoritma na računalu, moguća je pojava pogreške. Sama pogreška može biti posljedica sljedećih uzroka, tj. biti jedne od sljedećih vrsta:

- pogreška modela,
- pogreška u mjerenjima (ulaznim podacima),
- pogreška metode rješavanja te
- pogreška aritmetike na računalu.

Pogreške modela posljedica su nedostataka modela koji ne opisuje vjerno modeliranu pojavu. Ova vrsta pogreške ovisi o području primjene i mogućnostima raspoloživih modela. Uzrok ove vrste pogreške najčešće je svjesno zanemarivanje nekih svojstava promatrane pojave u cilju pojednostavljenja modeliranja. Primjerice, modeli elektroničkih elemenata (poluvodiča, tranzistora i dr.) samo približno opisuju njihovo ponašanje; u analizi matematičkog njihala, sinus kuta otklona aproksimira se iznosom tog kuta (što vrijedi za male kutove); sustavi nelinearnih diferencijalnih jednadžbi se linearizirаju, itd. Ovu vrstu pogreške procjenjuju eksperti u konkretnom području primjene, na kojima je zadatak razvoja ili odabira prikladnog modela.

Pogreške u ulaznim podacima posljedica su nemogućnosti (ili nepostojanja potrebe) provedbe točnih mjerenja. U većini slučajeva, male pogreške u postupku mjerenja uzrokovat će i relativno male pogreške u rezultatu algoritma. Međutim, u nekim problemima mala promjena (perturbacija) ulaznih podataka može dovesti do velikih promjena u rezultatima, pa u tom slučaju govorimo o nestabilnim ili loše uvjetovanim (engl. *ill-conditioned*) problemima. Neke primjere ovakvih problema upoznat ćemo u sljedećim poglavljima.

Pogreške metode rješavanja posljedica su prilagodbe numeričkih postupaka uvjetima njihove primjene, odnosno aproksimacijama koje se javljaju u postupku računanja. Primjerice, svaki postupak koji uključuje beskonačni broj elemenata ili operacija, u praksi se mora ograničiti na neku konačnu vrijednost. Ova aproksimacija obuhvaća sve postupke temeljene na limesu ili na konvergenciji niza približnih rješenja prema krajnjem rješenju. Ove pogreške uobičajeno se dijele na pogreške diskretizacije i pogreške odbacivanja.

Pogreške diskretizacije nastaju zamjenom kontinuiranih vrijednosti konačnim diskretnim skupom vrijednosti, odnosno zamjenom beskonačno male veličine nekom konačnom malom vrijednošću. Tipični primjer ove vrste pogreške je aproksimacija derivacije funkcije njenom diferencijom, pri čemu je moguće odabrati različite razine aproksimacije. Pogreške odbacivanja posljedica su "rezanja" beskonačnog niza na neki konačan broj članova, kako bi se rezultat mogao odrediti u konačnom vremenu. Ova vrsta pogreške

ovisi o odabranom numeričkom postupku i obično je njena procjena unaprijed poznata za određeni postupak.

Konačno, pogreške aritmetike na računalu posljedica su činjenice da se ulazne vrijednosti i međurezultati u postupku računanja na računalu ne mogu točno zapisati. Iako je u velikom broju slučajeva ova vrsta pogreške relativno mala u odnosu na prethodne i često se zanemaruje, u primjeni je moguće da zbog ove vrste pogreške konačni rezultat postane praktično neuporabljiv. U ostatku ovog poglavlja pozabavit ćemo se određivanjem utjecaja ove vrste pogreške te načinima njenog umanjivanja.

1.2 Prikaz brojeva s pomičnom točkom

Prikaz brojeva s pomičnom točkom definiran je ovisno o iznosu baze β, preciznosti p te uz zadani raspon eksponenta $[e_{min}, e_{max}]$. Na temelju zadanih parametara, vrijednost zapisana s pomičnom točkom je:

$$\pm \underbrace{\left(d_0 + d_1\beta^{-1} + d_2\beta^{-2} + \ldots + d_{p-1}\beta^{-(p-1)}\right)}_{signifikand} \cdot \beta^e,$$

gdje je e vrijednost eksponenta, a d_i su znamenke signifikanda. Uz zadane parametre prikaza, postoji

- β^p različitih vrijednosti signifikanada,
- $e_{max} - e_{min} + 1$ različitih vrijednosti eksponenata te
- dvije vrijednosti predznaka.

Skup svih mogućih vrijednosti koje se mogu prikazati u ovom zapisu definiramo kao skup brojeva s pomičnom točkom, A. *Normirani* broj (normirani signifikand) s pomičnom točkom ima znamenku najveće težine različitu od nule, odnosno $d_0 \neq 0$.

Primjer 1.1

Broj 0.1 zapisuje se u skupu A ovisno o parametrima prikaza na ove načine:

- uz $\beta = 10$ i $p = 3$, normalizirani prikaz je 1.00×10^{-1};
- uz $\beta = 2$ i $p = 24$, normalizirani prikaz je $1.10011001100110011001101 \times 2^{-4}$ (primijetimo kako se vrijednost 0.1 ne može točno zapisati u binarnoj bazi).

♣

1.2.1 Prikaz realnih vrijednosti po standardu IEEE-754

U računalima koristimo prikaz po standardu IEEE-754, koji definira način zapisivanja vrijednosti s pomičnom točkom. Prva inačica standarda datira iz 1985. godine, dok je najnovija inačica objavljena 2019. godine [**IEEE-754**]. Prikaz po ovom standardu ima sljedeća svojstva:

- baza iznosi $\beta = 2$.
- predznak se zapisuje u najljeviji bit, tako da vrijednost 0 odgovara predznaku $+$, a vrijednost 1 predznaku $-$.
- sljedećih k bitova služi za zapisivanje *posmaknutog eksponenta* s posmakom $a = 2^{k-1} - 1$. Vrijednost eksponenta e dobiva se tako da se k bitova posmaknutog eksponenta čita kao binarni broj $b \in \left[0, 2^k - 1\right]$ te od te vrijednosti oduzima posmak, odnosno $e = b - a = b - \left(2^{k-1} - 1\right)$. Posebno se tretiraju slučajevi u kojima su svi bitovi posmaknutog eksponenta jednaki 1 (tj. $b = 2^k - 1$) te kada su svi bitovi jednaki 0.

Tablica 1.1: IEEE prikaz jednostruke preciznosti

posmaknuti eksponent	frakcija f	dekodirana vrijednost	
$00000000_2 = 0$	$f = 0$	$(-1)^z \times 0$	pozitivna ili negativna nula
$00000000_2 = 0$	$f \neq 0$	$(-1)^z \times 2^{-126} \times (0.f)$	denormirani signifikand
$0 < b < 255$	$f = 0$ ili $f \neq 0$	$(-1)^z \times 2^{b-127} \times (1.f)$	normalne vrijednosti
$11111111_2 = 255$	$f = 0$	$(-1)^z \times \infty$	pozitivno ili negativno beskonačno
$11111111_2 = 255$	$f \neq 0$	*NaN*	nije broj

- posljednjih $p-1$ bitova koristi se za zapis signifikanda od p bitova. Naime, budući da normirani signifikand ima prvu znamenku različitu od nule, u binarnom zapisu ta je znamenka jednaka 1, pa se ne zapisuje. Stoga je vrijednost p bitova signifikanda S definirana kao $S = 1.f$, gdje je f oznaka za razlomljeni dio signifikanda od $p-1$ bitova koji se naziva *frakcija* ili *mantisa*. Iznimno se (ovisno o vrijednosti eksponenta) signifikand dekodira kao $S = 0.f$.

Od mnogih formata koje definira standard IEEE, najčešći u uporabi su format jednostruke preciznosti (engl. *single precision*) uz 32 bita, te dvostruke preciznosti (engl. *double precision*) uz 64 bita. Zapis u jednostrukoj preciznosti sadrži 1 bit za predznak z, $k = 8$ bitova za eksponent e te 23 bita za frakciju f (što daje signifikand od $p = 24$ bita), a pregledno je prikazan u tablici 1.1. U ovom formatu eksponent je posmaknut za $a = 2^{k-1} - 1 = 2^7 - 1 = 127$; drugim riječima, binarni broj zapisan u polju eksponenta smanjuje se za 127 kako bi se odredila vrijednost eksponenta. Kako je navedeno, posebno se tretiraju slučajevi u kojima je u polju eksponenta zapisan niz nula odnosno niz jedinica; izuzevši ta dva slučaja, vrijednost eksponenta u ovom zapisu poprima vrijednosti u intervalu $[-126, 127]$, odnosno moguće je prikazati vrijednosti s eksponentima u rasponu od 2^{-126} do 2^{127}.

Signifikand se u većini slučajeva dekodira kao $1.f$; iznimka je područje *denormiranog* (subnormiranog) signifikanda; u ovom slučaju, dekodirana vrijednost dobiva se uz najmanji mogući eksponent (2^{-126}) te uz signifikand oblika $0.f$. Zapis u kojemu su svi bitovi eksponenta jednaki 1, a frakcija je različita od nule, koristi se za označavanje nedefiniranih vrijednosti s oznakom *NaN* (engl. *not a number*). Ove vrijednosti dobivaju se kao rezultati nedefiniranih operacija, primjerice $\frac{0}{0}$, $\frac{\infty}{\infty}$, $\infty + (-\infty)$, $\sqrt{-1}$ i drugih.

Koji je opseg vrijednosti koje se mogu predstaviti u ovom prikazu? Najmanja apsolutna vrijednost (različita od nule) koju je moguće zapisati izražava se uz pomoć denormiranog signifikanda i iznosi $2^{-126} \times 0.000\cdots000_2 = 2^{-126} \times 2^{-23} = 2^{-149} \doteq 1.4 \times 10^{-45}$. Najmanja vrijednost normiranog signifikanda iznosi $2^{-126} \times 1.0_2 \doteq 1.2 \times 10^{-38}$, dok se najveća apsolutna vrijednost (manja od beskonačno) dobiva kao $2^{127} \times 1.111\cdots111_2 = 2^{127} \times (2 - 2^{-23}) \doteq 3.4 \times 10^{38}$.

Na analogni način definira se i prikaz dvostruke preciznosti, čije su vrijednosti vidljive u tablici 1.2.

Primjer 1.2

Pretpostavimo jednostavnu inačicu zapisa po uzoru na IEEE standard uz ukupno 8 bitova, od kojih su:

- 1 bit za predznak z,
- 3 bita za eksponent te

Tablica 1.2: IEEE prikaz dvostruke preciznosti

posmaknuti eksponent	frakcija f	dekodirana vrijednost	
0	$f = 0$	$(-1)^z \times 0$	pozitivna ili negativna nula
0	$f \neq 0$	$(-1)^z \times 2^{-1022} \times (0.f)$	denormirani signifikand
$0 < b < 2047$	$f = 0$ ili $f \neq 0$	$(-1)^z \times 2^{b-1023} \times (1.f)$	normalne vrijednosti
2047	$f = 0$	$(-1)^z \times \infty$	pozitivno ili negativno beskonačno
2047	$f \neq 0$	*NaN*	nije broj

- 4 bita za frakciju.

U ovom formatu posmak eksponenta je $a = 2^2 - 1 = 3$, a moguće je prikazati vrijednosti eksponenta u intervalu $[-2, 3]$. U tablici 1.3 nalazi se prikaz nekih vrijednosti u tom zapisu te dekodirane vrijednosti u domeni realnih brojeva (prikazane su samo pozitivne vrijednosti). Najmanja apsolutna vrijednost (osim nule) koja se može zapisati uz ove parametre iznosi $2^{-2} \times 0.0001_2$, odnosno $2^{-6} = 0.015625$ i zapisana je uporabom denormiranog signifikanda. Prva vrijednost koja koristi normirani signifikand iznosi $2^{-2} \times 1.0_2 = 0.25$. Najveća vrijednost, osim beskonačno, koja se može definirati u ovom zapisu je $2^3 \times 1.1111_2 = 15.5$.

Korisno je uočiti kako je raspodjela vrijednosti koje možemo prikazati ovim zapisom izrazito neujednačena; u denormiranom rasponu susjedne vrijednosti udaljene su za $0.015625 = 2^{-6}$, dok za najveći eksponent razmaci iznose $0.25 = 2^{-2}$. Ovo je posljedica toga da, za svaku moguću vrijednost eksponenta, možemo prikazati jednak broj vrijednosti, koji je u ovom slučaju 16 (uz četiri bita frakcije).

Koliku pogrešku možemo dobiti preslikavanjem proizvoljnih realnih vrijednosti u ovaj zapis? Želimo li prikazati vrijednost 8.25, u ovom zapisu to je moguće jedino uz pomoć vrijednosti 8.0 ili 8.5; u oba slučaja pogreška iznosi 0.25, a izraženo postotkom pogreška je $0.25/8.25 = 3\%$. Želimo li primjerice prikazati vrijednost 0.2578125, to ćemo moći uporabom najbliže moguće vrijednosti iz zadanog prikaza, 0.25. Time činimo pogrešku iznosa 0.00782125; no izražena u postocima, ta pogreška je $0.0078125/0.2578125$, odnosno također 3%.

♣

1.3 Pogreške zaokruživanja

Veličina skupa brojeva s pomičnom točkom A je ograničena, pa je opravdano pitati kako se beskonačni skup realnih brojeva $\mathbb{R}$ preslikava u A? Neka je $x \in \mathbb{R}$ *točna vrijednost*, a $rd\,(x) \in A$ neka je *strojna vrijednost* s pomičnom točkom. Operacija rd predstavlja preslikavanje $rd : \mathbb{R} \to A$ skupa realnih brojeva u skup brojeva s pomičnom točkom.

Prilikom preslikavanja, pravilno zaokruživanje obavlja se tako da je pogreška manja od polovice vrijednosti zadnjeg prikazanog mjesta (zadnje prikazane znamenke):

$$|rd\,(x) - x| \leq \frac{1}{2} \text{ vrijednosti zadnjeg prikazanog mjesta.}$$

Tablica 1.3: Primjer prikaza s 3 bita za eksponent i 4 bita za frakciju

z	b	f	vrijednost	
0	000	0000	$(+1) \times 0$	
0	000	0001	$(+1) \times 2^{-2} \times 0.0625$	$= 0.015625$
0	000	0010	$(+1) \times 2^{-2} \times 0.125$	$= 0.03125$
⋮				
0	000	1111	$(+1) \times 2^{-2} \times 0.9375$	$= 0.234375$
0	001	0000	$(+1) \times 2^{-2} \times 1.0$	$= 0.25$
0	001	0001	$(+1) \times 2^{-2} \times 1.0625$	$= 0.265625$
⋮				
0	001	1111	$(+1) \times 2^{-2} \times 1.9375$	$= 0.48425$
⋮				
0	110	0000	$(+1) \times 2^{3} \times 1.0$	$= 8.0$
0	110	0001	$(+1) \times 2^{3} \times 1.0625$	$= 8.5$
⋮				
0	110	1111	$(+1) \times 2^{3} \times 1.9375$	$= 15.5$
0	111	0000	$(+1) \times \infty$	
0	111	0001	*NaN*	
⋮				
0	111	1111	*NaN*	

Primjerice, broj x zapisan u dekadnom zapisu može se predstaviti kao $x = a \times 10^e$, gdje su vrijednosti eksponenta ograničene $e_{\min} \leq e \leq e_{\max}$, a signifikand je prikazan nizom znamenki:

$$|a| = d_0 d_1 d_2 \ldots d_{p-1} d_p d_{p+1} \ldots$$

U dekadnom zapisu vrijedi $0 \leq d_i \leq 9$ te $d_0 \neq 0$. Strojna vrijednost $rd\,(x)$ se uz p znamenki može zapisati kao $rd\,(x) = sign\,(x) \cdot a' \cdot 10^e$, gdje je

$$a' = \begin{cases} d_0 d_1 d_2 \cdots d_{p-1} & \text{za } 0 \leqslant d_p \leqslant 4, \\ d_0 d_1 d_2 \cdots d_{p-1} + 10^{-(p-1)} & \text{za } d_p \geqslant 5 \end{cases}$$

Ukoliko je broj zadan u binarnom zapisu, $x = a \cdot 2^e$, tada za znamenke signifikanda vrijedi $0 \leq d_i \leq 1$, uz $d_0 \neq 0$. Strojna vrijednost je tada $rd\,(x) = sign\,(x) \cdot a' \cdot 2^e$, gdje je

$$a' = \begin{cases} d_0 d_1 d_2 \cdots d_{p-1} & \text{za } d_p = 0, \\ d_0 d_1 d_2 \cdots d_{p-1} + 2^{-(p-1)} & \text{za } d_p = 1 \end{cases}$$

1.3.1 Apsolutna pogreška

Apsolutna pogreška definira se kao apsolutna razlika točne vrijednosti iz skupa $\mathbb{R}$ i strojne vrijednosti:

$$|rd\,(x) - x| \tag{1.1}$$

Posebno je važno odrediti *maksimalnu apsolutnu pogrešku* koja može nastati preslikavanjem u skup A; za točnu vrijednost prikazanu u bazi β, uz pravilno zaokruživanje, vrijedi

$$|rd\,(x) - x| \leq \underbrace{0.00\ldots0}_{p} d' \cdot \beta^e = \frac{\beta}{2} \cdot \beta^{-p} \cdot \beta^e = \frac{1}{2} \underbrace{\beta^{-(p-1)} \cdot \beta^e}_{ulp},$$

gdje je $d' = \dfrac{\beta}{2}$ vrijednost prve znamenke koja se ne može zapisati. Izraz $\beta^{-(p-1)} \cdot \beta^e$ zapisuje se kraće kao *ulp* (engl. *units in the last place*), odnosno jedinica zadnjeg mjesta. Prema tome, uz pravilno

zaokruživanje, maksimalna apsolutna pogreška iznosi $\frac{1}{2}\ ulp$. U binarnoj bazi ($\beta = 2$), najveća apsolutna pogreška iznosi 2^{e-p}.

1.3.2 Relativna pogreška

Relativna pogreška definira se kao omjer apsolutne pogreške i stvarne vrijednosti:

$$\varepsilon = \left| \frac{rd\,(x) - x}{x} \right| \tag{1.2}$$

Najveću relativnu pogrešku možemo odrediti uz uvrštenje najmanje i najveće zadane vrijednosti x, tj.

$$x_{\min} = \underbrace{1.00\ldots0}_{p} \times \beta^e = \beta^e,$$

$$x_{\max} = \delta.\delta\delta\ldots\delta \times \beta \doteq \beta \cdot \beta^e,$$

gdje je $\delta = \beta - 1$, najveća znamenka u dotičnom prikazu. Tada vrijedi:

$$\begin{aligned} \frac{\frac{\beta}{2} \cdot \beta^{-p} \cdot \beta^e}{\beta \cdot \beta^e} \leq & \left| \frac{rd\,(x) - x}{x} \right| \leq \frac{\frac{\beta}{2} \cdot \beta^{-p} \cdot \beta^e}{\beta^e} \\ \frac{1}{2}\beta^{-p} \leq & \left| \frac{rd\,(x) - x}{x} \right| \leq \frac{\beta}{2} \cdot \beta^{-p} \\ \frac{1}{2}\beta^{-p} \leq & \left| \frac{rd\,(x) - x}{x} \right| \leq \frac{1}{2} \cdot \beta^{-(p-1)} \end{aligned}$$

Vrijednost s desne strane gornjeg izraza predstavlja maksimalnu relativnu pogrešku i naziva se i *strojna preciznost* odnosno *strojni epsilon*:

$$eps = \frac{1}{2} \cdot \beta^{-(p-1)} \tag{1.3}$$

Uz dekadni zapis ($\beta = 10$), strojna preciznost iznosi $eps = 5 \cdot 10^{-p}$, a uz binarni zapis svodi se na $eps = 2^{-p}$. U IEEE prikazu jednostruke preciznosti, strojna preciznost je $2^{-24} \doteq 5.96 \cdot 10^{-8}$, a u dvostrukoj preciznosti $2^{-53} \doteq 1.11 \cdot 10^{-16}$. U primjeru 1.2 (vidi tablicu **??**), uz zadane parametre najveća relativna pogreška iznosi $eps = 2^{-p} = 2^{-5} = 0.03125 \doteq 3\%$, što je upravo vrijednost koju smo dobili prilikom preslikavanja. Uočimo da se parametar p odnosi na ukupni broj bitova signifikanda, a ne samo na broj bitova frakcije (koji je za jedan manji).

Primjer 1.3

Pretpostavimo zadanu vrijednost u dekadnoj bazi $x = 0.0314159$ uz raspoloživi broj znamenki za prikaz $p = 3$. Strojna vrijednost je tada $rd\,(x) = 3.14 \times 10^{-2}$, a apsolutna pogreška iznosi $|rd\,(x) - x| = 0.0000159$. Uz zadane parametre vrijednost jedinice zadnjeg mjesta iznosi

$$ulp = \beta^{-(p-1)} \cdot \beta^e = 10^{-2} \times 10^{-2} = 10^{-4}$$

Kod ispravno zaokruženih brojeva, apsolutna pogreška ne može biti veća od $\frac{1}{2}\ ulp$:

$$|rd\,(x) - x| = 0.159\ ulp < 0.5\ ulp$$

Relativna pogreška iznosi

$$\left| \frac{rd\,(x) - x}{x} \right| = \frac{0.0000159}{0.0314159} \doteq 0.0005$$

Kako je strojna preciznost $eps = 5 \times 10^{-p} = 5 \times 10^{-3} = 0.005$, relativna pogreška se može

napisati kao

$$\left|\frac{rd\,(x)-x}{x}\right| = 0.1\ eps$$

♣

Može se zaključiti kako preslikavanje $\mathbb{R} \to A$ unosi relativnu pogrešku koja je uvijek manja od eps. Drugim riječima, $\forall x \in \mathbb{R}$ preslikavanje $rd : \mathbb{R} \to A$ daje

$$rd\,(x) = x\,(1+\varepsilon)\,,\ \text{gdje je}\ |\varepsilon| \leq eps \tag{1.4}$$

Primjer 1.4

Zadana je vrijednost $x = 12.35$ uz $\beta = 10$ i $p = 3$. Uz ove parametre vrijednost jedinice zadnjeg mjesta je $ulp = 10^{-2} \times 10^{-1} = 10^{-1}$, a strojna preciznost iznosi $eps = 5 \times 10^{-3} = 0.005$. Pravilno zaokružena vrijednost je $rd\,(x) = 1.24 \times 10^{1}$, a apsolutna i relativna pogreška iznose:

$$|rd\,(x) - x| = 0.05 = 0.5\ ulp$$
$$\left|\frac{rd\,(x)-x}{x}\right| = \frac{0.05}{12.35} = 0.004048583 \doteq 0.8\ eps$$

Apsolutna i relativna pogreška prilikom zaokruživanja su na taj način uvijek ograničene; međutim, ukoliko se nad ulaznim vrijednostima izvode dodatne računske operacije, granica apsolutne pogreške ne mora biti jednaka.

Primjer uz računsku operaciju: ([**Goldberg:1991:CSK:103162.103163**]) **TODO**: dodati pojasnjenje primjera

$y = 8x = 98.8$
$rd\,(y) = rd\,(8x) = 9.88 \times 10^{-1}$
$rd\,(\tilde{y}) = 8 \cdot rd\,(x) = 9.92 \times 10^{-1}$
$|rd\,(\tilde{y}) - y| = 0.4 = 4\ ulp$
$\left|\frac{rd\,(\tilde{y})-y}{y}\right| = \frac{0.4}{98.8} = 0.0040485 \doteq 0.8\ eps$

♣

1.4 Aritmetika brojeva s pomičnom točkom

Za brojeve s pomičnom točkom, elemente skupa A, osnovne računske operacije $+$, $-$, $\cdot$, $/$ ostvarene na računalu "ne vrijede potpuno". Primjerice, jednakost

$$x + y = x$$

ne znači da je $y = 0$, već da je

$$|y| < eps\,|x|\,,\ \text{gdje su}\ x, y \in A$$

Na taj se način može i na drugi način definirati strojna preciznost:

$$eps = \min\{g \in A \mid 1 + g > 1,\ g > 0\}$$

Drugim riječima, eps je jednak najmanjem pozitivnom broju g za koji je

$$1 + g > 1$$

Isto tako, zakoni asocijativnosti i distributivnosti ne mogu se izravno primijeniti, što je vidljivo u sljedećem primjeru.

Primjer 1.5

Uz parametre zapisa broja s pomičnom točkom $\beta = 10$ te $p = 8$, zadana su tri pribrojnika:

$$a = 2.3371258 \times 10^{-5}$$
$$b = 3.3678429 \times 10^{1}$$
$$c = -3.3677811 \times 10^{1}$$

U ovom prikazu strojna preciznost eps iznosi 5×10^{-8}. Točan rezultat, ako se operacije provode s proizvoljnim brojem znamenki i na kraju zaokružuju, je $a+b+c = 6.41371258 \times 10^{-4}$. Međutim, uporaba samo p znamenki tijekom zbrajanja može dovesti do različitih rezultata, ovisno o redoslijedu pribrojnika. Ukoliko se njihov zbroj računa kao $a + (b + c)$, dobivamo:

$b + c$:

$$\begin{array}{r} 3.3678429 \times 10^{1} \\ -3.3677811 \times 10^{1} \\ \hline 0.0000618 \times 10^{1} = 6.1800000 \times 10^{-4} \end{array}$$

$a + (b + c)$:

$$\begin{array}{r} 0.2337125 \times 10^{-4} \\ 6.1800000 \times 10^{-4} \\ \hline 6.4137126 \times 10^{-4} \end{array}$$

U ovom slučaju, relativna pogreška rezultata je 3.12×10^{-9}, odnosno $0.0624\ eps$. S druge strane, ako se zbroj računa kao $(a + b) + c$, dobija se:

$a + b$:

$$\begin{array}{r} 0.0000023 \times 10^{1} \\ 3.3678429 \times 10^{1} \\ \hline 3.3678452 \times 10^{1} \end{array}$$

$(a + b) + c$:

$$\begin{array}{r} 3.3678452 \times 10^{1} \\ -3.3677811 \times 10^{1} \\ \hline 0.0000641 \times 10^{1} = 6.4100000 \times 10^{-4} \end{array}$$

Prilikom ovog redoslijeda zbrajanja, dobivamo relativnu pogrešku od 5.79×10^{-4}, odnosno čak $11580\ eps$! Što bi se dogodilo uz treći mogući redoslijed, $(a + c)+b$? Nažalost, došli bismo do jednakog rezultata kao i u prethodnom slučaju, uz jednaku relativnu pogrešku. Logično je zapitati se, kako možemo *unaprijed* odrediti redoslijed operacija uz koji postižemo minimalnu pogrešku? U slučaju zbrajanja, na ovo pitanje srećom postoji učinkovit odgovor koji će biti prikazan kasnije. ♣

Iz prethodnog primjera vidljivo je kako oduzimanje dvaju brojeva $x, y \in A$ istog predznaka može dovesti do *poništavanja vodećih znamenki*. Primjerice, uz

$$x = 3.15876 \times 10^{2}$$
$$y = 3.14289 \times 10^{2}$$

operacija oduzimanja daje

$$x - y = 0.01587 \times 10^{2}$$
$$= 1.587 \times 10^{0}$$

To samo po sebi nije opasno, ali se može pokazati da dovodi do velikih pogrešaka ako su x i y već predhodno izračunati s pogreškama. Prilikom oduzimanja, ako su operandi točni (tj. njihova relativna pogreška je blizu nuli), tada je poništavanje vodećih znamenki benigno. S druge strane, ako su operandi dobiveni prethodnim operacijama s postojećom pogreškom ($\epsilon \neq 0$), tada nastaje maligno ili katastrofalno poništavanje (engl. *catastrophic cancellation*). Opravdano je provjeriti kolika pogreška može nastati nakon samo jedne operacije oduzimanja, o čemu govori sljedeći teorem.

Teorem 1.1

Ako se u sustavu brojeva s pomičnim zarezom s parametrima β i p prilikom oduzimanja dvaju brojeva koristi p znamenki, relativna pogreška rezultata može biti velika β-1. ♡

Dokaz Relativna pogreška iznosa $\beta - 1$ pojavljuje se za dekadski sustav (uz $\beta = 10$) prilikom oduzimanja u sljedećem slučaju:

$$x = 1.00\ldots0$$

$$y = 0.\underbrace{99\ldots9}_{p}$$

Točan rezultat (izračunat s proizvoljim brojem znamenki) iznosi:

$$x - y = 0.\underbrace{00\ldots1}_{p} = 10^{-p}$$

S druge strane, izračun u sustavu brojeva s pomičnom točkom uz p znamenki daje:

$$x = 1.\overbrace{00\ldots0}^{p} \times 10^{0}$$

$$y = 9.99\ldots9 \times 10^{-1}$$

$$x = 1.00\ldots0 \times 10^{0}$$

$$y = 0.99\ldots9 \times 10^{0}$$

$$x - y = 0.00\ldots1 \times 10^{0} = 1.00\ldots0 \times 10^{-(p-1)}$$

Prema tome, relativna pogreška iznosi:

$$\left|\frac{10^{-p} - 10^{-(p-1)}}{10^{-p}}\right| = \left|\frac{10^{-p}\,(1-10)}{10^{-p}}\right| = 9$$

Na jednak način se primjer može poopćiti za bilo koji β, pa se u općenitom slučaju dobiva:

$$\left|\frac{\beta^{-p} - \beta^{-(p-1)}}{\beta^{-p}}\right| = \left|\frac{\beta^{-p}\,(1-10)}{\beta^{-p}}\right| = \beta - 1$$

Uz $\beta = 2$, relativna pogreška može biti velika $\beta - 1 = 2 - 1 = 1$, tj. 100%! Drugim riječima, to znači da je apsolutna pogreška jednaka rezultatu.

Iako koristimo p znamenki za zapis rezultata i operanada, u praksi se u ostvarenju računskih operacija mogu koristiti dodatne znamenke, samo za međurezultate. Ove znamenke nazivaju se *zaštitne znamenke* (engl. *guard digits*) i danas su u uporabi u svim ostvarenjima aritmetičkih operacija na računalu. Zaštitne znamenke su potrebne samo prilikom izvođenja operacija s pomičnom točkom u procesoru, no nije ih potrebno dodavati registrima procesora. Iako se u praksi može koristiti veći broj zaštitnih znamenki, sljedeći teorem pokazuje koliko samo jedna dodatna znamenka ima utjecaj na ograničavanje pogreške rezultata.

Teorem 1.2

Ako su x i y pozitivni brojevi s pomičnom točkom u prikazu s parametrima β i p, te ako se oduzimanje obavlja s $p+1$ znamenkom (uz dodanu tzv. zaštitnu znamenku, engl. guard digit) tada je relativna pogreška u rezultatu manja od

$$\beta^{-p}\left(1+\frac{\beta}{2}\right) = \left(1+\frac{\beta}{2}\right) eps \leq 2\, eps.$$

♡

Dokaz Pretpostavimo bez smanjenja općenitosti da vrijedi $x > y$ (vrijednosti x i y se mogu i zamijeniti). Neka su x i y predstavljeni kao

$$\begin{aligned} x &= x_0.x_1x_2\ldots x_{p-1} \times \beta^{e_x} \\ y &= y_0.y_1y_2\ldots y_{p-1} \times \beta^{e_y} \end{aligned}$$

Tada postoje sljedeće mogućnosti s obzirom na iznose eksponenata e_x i e_y:

- uz $e_x = e_y$, rezultat je točan (nema gubitka znamenki).
- uz $e_y = e_x - 1$, oduzimanje sa zaštitnom znamenkom daje rezultat koji je zaokružen na vrijednost iz skupa A s pogreškom manjom od eps:

$$\begin{gathered} \overbrace{x_0.x_1x_2\ldots x_{p-1}0}^{p+1} \times \beta^{e_x} \\ 0.y_0y_1\ldots y_{p-2}\ \underbrace{y_{p-1}}_{zaštitna} \times \beta^{e_x} \end{gathered}$$

- uz $e_y = e_x - k$, gdje je $p \geq k > 1$, nakon svođenja na istu vrijednost eksponenta dobiva se:

$$\begin{aligned} x &= x_0.x_1\ldots x_{p-1} \\ y &= y_0.y_1\ldots y_{p-1} \times \beta^{-k} \end{aligned}$$

										$\beta^{-(p-1)}$	β^{-p}	β^{-p-1}		β^{-p-k}
$x=$	x_0	.	x_1	x_2	$\ldots$	x_k	x_{k+1}	$\ldots$	x_{p-1}	0	0	$\ldots$	0
$y=$	0	.	0	0	$\ldots$	y_0	y_1	$\ldots$	y_{p-k-1}	y_{p-k}	y_{p-k+1}	$\ldots$	y_{p-1}
$\overline{y}=$	0	.	0	0	$\ldots$	y_0	y_1	$\ldots$	y_{p-k-1}	y_{p-k}	0	$\ldots$	0

Razlika između točne vrijednosti y i posmaknute vrijednosti $\overline{y}$ uz zaštitni bit je

$$\begin{aligned} y - \overline{y} &= y_{p-k+1}\cdot\beta^{-p-1} + y_{p-k+2}\cdot\beta^{-p-2} + \ldots + y_{p-1}\cdot\beta^{-p-k} \leq \\ &\leq \underbrace{(\beta-1)}_{\text{najveća znamenka}} \left(\beta^{-p-1} + \beta^{-p-2} + \ldots + \beta^{-p-k}\right) = \\ &= \beta^{-p} - \beta^{-p-k} = \beta^{-p}\left(1-\beta^{-k}\right) < \beta^{-p} \end{aligned}$$

Točna vrijednost razlike je $x-y$, dok se izračunata vrijednost $x-\overline{y}$ još i zaokružuje na p znamenaka, što unosi dodatnu pogrešku $|\delta| \leq eps = \frac{\beta}{2}\beta^{-p}$. Ukupna apsolutna pogreška tada iznosi:

$$|(x-y) - (x-\overline{y} \pm \delta)| = |y - \overline{y} + \delta|$$

Relativna pogreška je stoga:

$$\left|\frac{y-\bar{y}+\delta}{x-y}\right|$$

Za slučaj kada je $x - y \geq 1$ dobija se:

$$\left|\frac{y-\bar{y}+\delta}{x-y}\right| \leq \left|\frac{x-\bar{y}+\delta}{1}\right| \leq \frac{\beta^{-p}+\frac{\beta}{2}\beta^{-p}}{1} = \beta^{-p}\left(1+\frac{\beta}{2}\right)$$

Može se pokazati [**Goldberg:1991:CSK:103162.103163**] da navedeno vrijedi i za $x - y < 1$.

U standardu IEEE-754, osim zapisa vrijednosti propisana su i svojstva računskih operacija koje se izvode na računalu. Budući da većina standardnih svojstava aritmetičkih operacija (poput asocijativnosti zbrajanja) ne vrijedi za računalnu aritmetiku, kako je moguće za proizvoljni niz operacija (proizvoljni algoritam) procijeniti ukupnu pogrešku rezultata? Iako je za konkretne izvedbe računskih operacija moguće definirati alternativna prilagođena svojstva, ona su zbog složenosti praktično neupotrebljiva za analizu algoritama s imalo većim brojem operacija. Zbog toga, standardom je propisano samo da rezultat računske operacije mora biti *ispravno zaokružen*: drugim riječima, rezultat operacije na računalu bit će jednak onom kojemu bismo dobili da se operacija izvodi s proizvoljim (dovoljnim) brojem zaštitnih znamenki, a rezultat zaokružuje na najbližu vrijednost iz skupa A (naravno, ako je točan rezultat u opsegu prikaza skupa A). Ako su x i y brojevi iz skupa A, a $\circ$ predstavlja neku osnovnu računsku operaciju izvedenu na računalu, tada će za ispravno zaokružene operacije vrijediti

$$rd\,(x \circ y) = (x \circ y)\,(1+\epsilon)\,,\ |\epsilon| \leq eps. \tag{1.5}$$

Primjerice, ako s $\oplus$ označimo operaciju zbrajanja provedenu na računalu, tada se za rezultat te operacije može napisati

$$x \oplus y = (x+y)\,(1+\epsilon)\,,\ |\epsilon| \leq eps.$$

Na ovaj način moguće je operaciju po operaciju analizirati tijek proizvoljnog algoritma te, s obzirom na definirani niz koraka, odabrati onu inačicu koja će rezultirati minimalnom mogućom pogreškom, o čemu je riječ u idućem odjeljku.

1.5 Rasprostiranje pogrešaka

Kako je vidljivo iz primjera u prethodnom odjeljku, moguće je da dva matematički ekvivalentna postupka, npr:

$$a + (b + c)$$
$$(a + b) + c$$

ne moraju dati jednake rezultate. Prema tome, uz $a, b, c \in A$, sljedeća dva algoritma nisu ekvivalentna:

$x := b + c;$ $\qquad$ $x := a + b;$
$y := a + x;$ $\qquad$ $y := x + c;$

Primjer 1.6

Rješavanje kvadratne jednadžbe $ax^2 + bx + c = 0$
Do rješenja se može doći sljedećim izrazima:

$$(A) \qquad x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$
$$(B) \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Neka u ovom slučaju vrijedi:

$$p = 3 \qquad a = 1.22$$
$$\beta = 10 \qquad b = 3.34$$
$$c = 2.28$$

Točna vrijednost diskriminante $b^2 - 4ac$ iznosi 0.0292; međutim, u zadanom zapisu b^2 se zaokružuje na 11.2, $4ac$ se zaokružuje na 11.1, te je

$$b^2 - 4ac = 0.1$$

Ako je $b^2 \gg 4ac$, računanje diskriminante nije opasno; no ako je $\sqrt{b^2 - 4ac} \doteq |b|$, tada u izrazima (A) i (B) može doći do poništavanja. No do rješenja kvadratne jednadžbe može se doći i sljedećim izrazima:

$$(C) \qquad x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} = \frac{2c}{-b - \sqrt{b^2 - 4ac}}$$

$$(D) \qquad x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

Postupak izračunavanja se tako može načiniti robusnim ako se:

- uz $b > 0$ koristi (C) i (B);
- uz $b < 0$ koristi (A) i (D).

♣

Za potrebe analize, uvedimo sljedeće oznake; neka su ulazne vrijednosti algoritma predstavljene vektorom

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

a izlazne vrijednosti kao

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Svaki algoritam se tada može prikazati kao preslikavanje $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{x})$, odnosno $y_j = \varphi_j(x_1, x_2, \ldots, x_n), j \in [1, m]$. No algoritam se može odviti u slijedu koraka s međurezultatima u *i*-tom koraku:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_{n_i}^{(i)} \end{bmatrix} \in D_i,\ D_i \subseteq \mathbb{R}^{n_i}$$

$$\varphi^{(i)} : D_i \to \mathbb{R}^{n_{i+1}}$$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{\varphi}^{(i)} \left(\boldsymbol{x}^{(i)} \right)$$

Slijed koraka algoritma $i = 0, 1, \ldots, r$ daje

$$\begin{aligned} \varphi &= \varphi^{(r)} \circ \varphi^{(r-1)} \circ \ldots \circ \varphi^{(0)} \\ D_0 &= D,\ D_{r+1} \subseteq \mathbb{R}^n \\ \varphi^{(i)} &: D_i \to D_{i+1} \end{aligned}$$

Primjer 1.7

Neka je zadatak izračunati zbroj tri pribrojnika: $\varphi(a, b, c) = a + b + c$, te neka su dostupna dva trivijalna algoritma:

$$\begin{array}{ll} I & \eta := a + b; \\ & y := \eta + c; \end{array}$$

$$\begin{array}{ll} II & \eta := b + c; \\ & y := a + \eta; \end{array}$$

Uz pretpostavku o ispravnom zaokruživanju (1.5), svaka operacija zbrajanja može biti prikazana kao umnožak točne vrijednosti i perturbacije: $(a + b)(1 + \epsilon)$, gdje je iznos pogreške ograničen s $|\epsilon| \leq eps$. Prema tome, analiza algoritma I može se provesti na ovaj način:

$$\begin{aligned} \tilde{\eta} =& a \oplus b = (a + b)(1 + \epsilon_1) \\ \tilde{y} =& \tilde{\eta} \oplus c = (\tilde{\eta} + c)(1 + \epsilon_2) \\ =& [(a + b)(1 + \epsilon_1) + c](1 + \epsilon_2) \\ =& (a + b + c)\left[1 + \frac{a + b}{a + b + c}\epsilon_1 (1 + \epsilon_2) + \epsilon_2\right] \end{aligned}$$

Relativna pogreška rezultata je

$$\begin{aligned}\epsilon_y =& \frac{\tilde{y}-y}{y}\\ =& \frac{a+b}{a+b+c}\epsilon_1(1+\epsilon_2)+\epsilon_2\\ \doteq& \frac{a+b}{a+b+c}\epsilon_1+\epsilon_2\\ \leq& \left(1+\frac{a+b}{a+b+c}\right)eps\end{aligned}$$

Analognom analizom algoritma II dolazi se do sljedeće pogreške:

$$\epsilon_y \leq \left(1+\frac{b+c}{a+b+c}\right)eps$$

Iz ove analize vidljivo je kako će konačna pogreška ovisiti o omjeru zbroja prva dva pribrojnika i cjelokupnog rezultata. Primijenimo li ovu procjenu pogreške na pribrojnike iz primjera 1.5 ($a = 2.3371258 \times 10^{-5}$, $b = 3.3678429 \times 10^{1}$, $c = -3.3677811 \times 10^{1}$), vidljivo je kako je redoslijed operacija $a+(b+c)$ bolji od redoslijeda $(a+b)+c$, budući da je $|a+b| \gg |a+b+c|$. ♣

Prilikom zbrajanja većeg broja pribrojnika, najjednostavniji postupak dodavat će pribrojnike u jednu pomoćnu varijablu (suma) onim redom kako su zadani. Trenutna vrijednost sume može postati većeg reda veličine nego sljedeći pribrojnik, pa dolazi do poništavanja znamenki niže težine manjih pribrojnika. Može se pokazati kako bi ovaj postupak bio robusniji ako sve pribrojnike prvo uredimo po veličini te potom zbrojimo od najmanjega prema najvećemu. Međutim, ovakav način uvodi dodatnu složenost postupka uređivanja, te zahtjeva pristup svim pribrojnicima odjednom (dok u stvarnim uvjetima pribrojnici mogu postajati poznati jedan po jedan).

Umjesto toga, uputno je koristiti neki od algoritama posebno razvijenih za ovu namjenu, kao što je Kahanov algoritam. Tijek Kahanovog algoritma opisan je sljedećim pseudokodom:

Algorithm 1 Kahanov algoritam

```
S = X[1];
C = 0;
za j = 2 do N čini
    Y = X[j] - C;
    T = S + Y;
    C = (T - S) - Y;
    S = T;
```

Ideja algoritma je čuvanje dijela izgubljenih znamenki (zbog eventualne razlike u redu veličine) u posebnoj varijabli (`C`), te nadomještanje te informacije pri svakom sljedećem zbrajanju. Pretpostavimo kako je varijabla `C` na početku jednaka nuli; tada će vrijednost sljedećeg pribrojnika biti pohranjena u `Y`. U retku 5, ako je trenutna vrijednost varijable `S` puno veća od `Y`, donji dio znamenki manje težine pribrojnika `Y` izgubit će se prilikom računanja nove sume, `T`. U retku 6, prvo se od nove sume `T` oduzima prethodna suma `S`, čime razlika (`T-S`) postaje jednaka onom dijelu znamenki veće težine `Y` koji su registrirani prilikom zbrajanja. No potom se od toga iznosa oduzima vrijednost pribrojnika `Y`, čime se u varijablu `C` pohranjuje vrijednost onih znamenki koje su bile izgubljene prilikom dodavanja sumi. Na početku svake sljedeće iteracije, u retku 4 ta se korekcija oduzima od sljedećeg pribrojnika; ako su pribrojnici otprilike

Tablica 1.4: Prikaz učinkovitosti Kahanovog algoritma

	uniformno iz $[0,1]$	uniformno iz $[-1,1]$	$\mathcal{N}(0,1)$
pogreška: jednostruka preciznost	5.153125	5.160864	5.177447
pogreška: Kahanov algoritam	0	0.000016	0.000024

jednakog reda veličine, ova će se korekcija zabilježiti i na taj način pridonijeti smanjenju ukupne pogreške.

Osim Kahanovog algoritma, u uporabi su i novije inačice dostupne u odgovarajućim numeričkim bibliotekama; u svakom slučaju preporuča se koristiti ovakav postupak umjesto naivnog pristupa.

Primjer 1.8

Isprobajmo učinkovitost prethodnog algoritma na zbrajanju većeg broja slučajno odabranih pribrojnika. Zbrajanje će biti provedeno jednostrukom preciznošću uporabom naivnim postupkom te Kahanovim algoritmom, dok će "točna" (referentna) vrijednost zbroja biti dobivena zbrajanjem u dvostrukoj preciznosti. U primjeru ćemo koristiti niz od 10^7 slučajno generiranih brojeva; ukoliko je niz brojeva generiran po različitim raspodjelama, zbrajanjem se dobivaju prosječne pogreške navedene u tablici 1.4. ♣

Primjer 1.9

Česta je potreba računanja razlike kvadrata, odnosno $y = a^2 - b^2$. Analizirajmo sljedeće algoritme:

$$
\begin{aligned}
I \qquad & \eta_1 := a \cdot a; \\
& \eta_2 := b \cdot b; \\
& y := \eta_1 - \eta_2;
\end{aligned}
$$

$$
\begin{aligned}
II \qquad & \eta_1 := a + b; \\
& \eta_2 := a - b; \\
& y := \eta_1 \cdot \eta_2;
\end{aligned}
$$

Pretpostavimo da su $\oplus$, $\ominus$ i $\otimes$ oznake za operacije zbrajanja, oduzimanja i množenja u artimetici s pomičnom točkom. Tada za prvi algoritam dobivamo

$$
\begin{aligned}
\tilde{\eta_1} &= a \oplus b = (a+b)(1+\epsilon_1) \\
\tilde{\eta_2} &= a \ominus b = (a-b)(1+\epsilon_2) \\
\tilde{y} &= \tilde{\eta_1} \otimes \tilde{\eta_2} = (\tilde{\eta_1} + \tilde{\eta_2})(1+\epsilon_3)
\end{aligned}
$$

gdje je $|\epsilon_1, \epsilon_2, \epsilon_3| \leq eps$. Konačan rezultat je tada

$$
\tilde{y} = (a+b)(a-b)(1+\epsilon_2)(1+\epsilon_2)(1+\epsilon_3),
$$

dok je relativna pogreška prvog algoritma

$$\epsilon_I = \frac{\tilde{y} - y}{y} = (1 + \epsilon_2)(1 + \epsilon_2)(1 + \epsilon_3) - 1.$$

Nakon množenja i uz zanemarenje viših potencija pogrešaka, dobivamo

$$\epsilon_I \doteq \epsilon_1 + \epsilon_2 + \epsilon_3 \leq 3 \cdot eps.$$

Za drugi algoritam može se provesti analogni postupak, te uz zanemarivanje viših potencija pogrešaka dobivamo sljedeću procjenu:

$$\epsilon_{II} \doteq (\epsilon_1 - \epsilon_2)(1 + \epsilon_3) \frac{b^2}{a^2 - b^2}.$$

U ovom slučaju, nije moguće dosljedno ograničiti iznos ukupne pogreške: ako su vrijednosti a i b bliske, vrijednost omjera $b^2/\left(a^2 - b^2\right)$ može biti proizvoljno velika. Iz ovoga je vidljivo kako algoritam I ima bolja svojstva u odnosu na algoritam II.

♣

1.6 Faktor pojačanja pogreške

Pretpostavimo da je, zbog prethodnih pogrešaka, $\tilde{\boldsymbol{x}}$ aproksimacija ulaznih vrijednosti $\boldsymbol{x}$. Tada je $\triangle \boldsymbol{x} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$, odnosno $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \triangle \boldsymbol{x}$. Gledano po komponentama, $\triangle x_i = \tilde{x_i} - x_i$ je apsolutna pogreška *i*-te ulazne vrijednosti. Relativna pogreška može se definirati kao

$$\epsilon_{x_i} = \frac{\tilde{x_i} - x_i}{x_i} = \frac{\triangle x_i}{x_i},\ x_i \neq 0.$$

Ako je na taj način definirana pogreška u ulaznim vrijednostima, koliko iznosi pogreška u rezultatu ϵ_y, u ovisnosti o pogreškama na ulazu, čak i uz *savršeno točne* računske operacije (ne nužno izvedene na računalu)? Razvojem funkcije y_j u red oko točke $\boldsymbol{x}$ i odbacivanjem viših članova dobiva se

$$y_j(\tilde{\boldsymbol{x}}) = y_j(\boldsymbol{x} + \triangle \boldsymbol{x}) \doteq y_j(\boldsymbol{x}) + \left(\boldsymbol{\nabla y_j}\right)^t \cdot \triangle \boldsymbol{x} + \ldots,$$

gdje je

$$\boldsymbol{\nabla y_j} = \begin{bmatrix} \frac{\partial \varphi_j(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial \varphi_j(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial \varphi_j(\boldsymbol{x})}{\partial x_n} \end{bmatrix}$$

Apsolutna pogreška rezultata funkcije y_j je

$$\triangle y_j = y_j(\boldsymbol{x} + \triangle \boldsymbol{x}) - y_j(\boldsymbol{x}) \doteq \left(\boldsymbol{\nabla y_j}\right)^t \cdot \triangle \boldsymbol{x} = \sum_{i=1}^{n} \frac{\partial \varphi_j}{\partial x_i} \triangle x_i$$

Ako se svaki pribrojnik pomnoži s $\frac{x_i}{x_i}$ te desna strana s $\frac{y_j}{\varphi_j(\boldsymbol{x})}$:

$$\triangle y_j = \left(\sum_{i=1}^{n} \frac{\partial \varphi_j}{\partial x_i} \cdot \triangle x_i \cdot \frac{x_i}{x_i}\right) \frac{y_j}{\varphi_j(\boldsymbol{x})}$$

$$\frac{\triangle y_j}{y_j} = \sum_{i=1}^{n} \frac{x_i}{\varphi_j(\boldsymbol{x})} \cdot \frac{\partial \varphi_j}{\partial x_i} \cdot \frac{\triangle x_i}{x_i}$$

$$\epsilon_{y_j} \doteq \sum_{i=1}^{n} \frac{x_i}{\varphi_j(\boldsymbol{x})} \cdot \frac{\partial \varphi_j}{\partial x_i} \cdot \epsilon_{x_i}$$

Izraz koji povezuje relativnu pogrešku ulazne vrijednosti s relativnom pogreškom rezultata naziva se **faktor pojačanja pogreške**:

$$S_{y_j}^{x_i} = \frac{x_i}{\varphi_j(\boldsymbol{x})} \cdot \frac{\partial \varphi_j}{\partial x_i} \tag{1.6}$$

Faktor pojačanja pogreške može se odrediti za osnovne operacije, pa uz $x \neq 0, y \neq 0$ vrijede sljedeći izrazi:

- za $\varphi(x,y) = x \cdot y$, vrijedi $\epsilon_{xy} \doteq \epsilon_x + \epsilon_y$

jer je $S_{xy}^x = \dfrac{x}{\varphi(x,y)} \cdot \dfrac{\partial \varphi(x,y)}{\partial x} = \dfrac{x}{x \cdot y} \cdot y = 1$, te $S_{xy}^y = \dfrac{y}{\varphi(x,y)} \cdot \dfrac{\partial \varphi(x,y)}{\partial y} = \dfrac{y}{x \cdot y} \cdot x = 1$

- za $\varphi(x,y) = x/y$, vrijedi $\epsilon_{x/y} \doteq \epsilon_x - \epsilon_y$

uz $S_{x/y}^x = \dfrac{x}{x/y} \cdot \dfrac{1}{y} = 1$, te $S_{x/y}^y = \dfrac{y}{x/y} \cdot \left(-\dfrac{x}{y^2}\right) = -1$

- za $\varphi(x,y) = x \pm y$, uz $x \pm y \doteq 0$ vrijedi $\epsilon_{x \pm y} \doteq \dfrac{x}{x \pm y}\epsilon_x \pm \dfrac{y}{x \pm y}\epsilon_y$

uz $S_{x \pm y}^x = \dfrac{x}{x \pm y} \cdot 1$, te $S_{x \pm y}^y = \dfrac{y}{x \pm y} \cdot (\pm 1)$

- za $\varphi(x) = \sqrt{x}$, vrijedi $\epsilon_{\sqrt{x}} \doteq \dfrac{1}{2}\epsilon_x$

uz $S_{\sqrt{x}}^x = \dfrac{x}{\sqrt{x}} \cdot \dfrac{1}{2}x^{-\frac{1}{2}} = \dfrac{1}{2}$.

Opisani faktor pojačanja pogreške pokazuje kako će pogreška u ulaznim podacima utjecati na pogrešku u rezultatu, uz pretpostavku o savršenoj izvedbi pojedine računske operacije. Naravno, ne zaboravimo da u stvarnim uvjetima rada na računalu i sama izvedba operacije unosi dodatnu pogrešku.

Primjer 1.10

Prisjetimo se računanja razlike kvadrata $y = a^2 - b^2$ iz primjera 1.9, za koje su navedena dva algoritma. U prvom algoritmu, koji razliku kvadrata računa kao $a^2 - b^2$, pretpostavimo da ϵ_{a^2} i ϵ_{b^2} predstavljaju relativne pogreške prilikom operacija množenja. Tada će ukupna pogreška rezultata biti pojačana pogreškom oduzimanja, odnosno

$$\epsilon_{a^2-b^2} \doteq \frac{a^2}{a^2-b^2}\epsilon_{a^2} + \frac{b^2}{a^2-b^2}\epsilon_{b^2}.$$

Može se vidjeti da ukupna pogreška nije ograničena i može biti proizvoljno povećana, budući da ovisi o omjeru pojedinog argumenta i njihove razlike. S druge strane, ako se razlika kvadrata računa kao $(a+b)(a-b)$, uz $\epsilon_{(a+b)}$ i $\epsilon_{(a-b)}$ kao relativne pogreške argumenata prije množenja, ukupna pogreška može se opisati s

$$\epsilon_{(a+b)(a-b)} \doteq \epsilon_{(a+b)} + \epsilon_{(a-b)}.$$

U ovom je slučaju onemogućeno prozvoljno pojačavanje pogreške iz prvog koraka algoritma, što predstavlja bolju opciju. Iz ovog primjera vidimo kako smo na drugi način došli do jednakog zaključka kao i analizom provedenom u primjeru 1.9. ♣

1.7 Zaključne napomene

Prethodni primjeri pokazuju kako je u ostvarenju mnogih postupaka na računalu potrebno imati na umu ograničenja prikaza i pogreške zaokruživanja. Nažalost, postoje primjeri iz stvarnog života u kojima je zbog numeričke pogreške došlo do velikih materijalnih šteta ili čak stradavanja ljudi. Srećom, postojeće biblioteke za numeričke postupke danas su većinom ostvarene uzevši u obzir navedena ograničenja, te je preporučeno kad god je moguće koristiti ugrađenu funkcionalnost takvih biblioteka. Kao nekoliko konkretnih primjera, preporuča se koristiti Kahanov (ili sličan dostupni) algoritam za zbrajanje većeg broja pribrojnika, ugrađene funkcije za računanje korijena kvadratne jednadžbe, za računanje hipotenuze i sličnih često korištenih elemenata. Također, moguće je pojedine funkcije implementirati u dvije razine preciznosti (s različitim brojem korištenih znamenki, npr. uz 64 i 128 bita), te usporediti njihovo ponašanje za zadane ulazne podatke; pri tome, uočljiva razlika u rezultatima može upućivati na veliku ovisnost pogreške o ulaznim podacima.

Iako se čini kako je problematika prikaza realnih vrijednosti na računalu odavno riješena, i danas je moguće pronaći načine prikaza koji će biti učinkovitiji u određenom području primjene. U dubokom učenju, optimiranje parametara modela može trajati izuzetno dugo i uzrokovati velik utrošak procesorskog vremena s jedne strane, dok se s druge strane u nizu velikog broja operacija mogu nagomilati nezanemarive pogreške. Stoga je grupa istraživača definirala prilagodljivi format prikaza realnih brojeva, temeljen na postojećem standardu, koji omogućava veću preciznost u području vrijednosti koje se većinom koriste za prikaz težina veza unutar neuronske mreže. Uz primjenu ovog načina kodiranja, točnost prilagodljivog 32-bitnog zapisa usporediva je s točnošću postojećeg 64-bitnog zapisa u standardnom formatu, što samo pokazuje koliki utjecaj prikaz vrijednosti može imati u stvarnoj primjeni uz veliku količinu računanja. **https://spectrum.ieee.org/floating-point-numbers-posits-processor**