SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivona Varnica

TOČNOST SUMACIJSKIH ALGORITAMA

Diplomski rad

Voditelj rada:
doc. dr. sc. Nela Bosner

Zagreb, veljača 2020.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvljujem se svojoj mentorici, doc. dr. sc. Neli Bosner, na strpljenju, pomoći i vodstvu pri izradi ovog diplomskog rada.
Želim se zahvaliti i svojim profesoricama iz matematike iz osnovne i srednje škole. Hvala im na prenesenom znanju i na strpljenju i savjetima na natjecanjima, zbog vas sam i zavoljela matematiku.
Hvala svim prijateljima i kolegama s kojima sam dijelila sve radosti i muke kroz studentske dane, zajedno smo uspjeli.
Posebno hvala baki i teti koje su me uvijek motivirale i vjerovale u mene.
I naravno, veliko hvala mami, tati i sestri što su sve ovo proživljavali sa mnom, bez vaše ljubavi i podrške ovo ne bi bilo moguće.

Sadržaj

Sadržaj **iv**

Uvod **2**

1 Konačna aritmetika računala **3**
1.1 Mjere za grešku 3
1.2 Tipovi grešaka 4
1.3 Stabilnost numeričkih algoritama 5
1.4 Binarna aritmetika s pomičnom točkom 7
1.5 Skalarni produkt 8

2 Analiza grešaka osnovnih metoda sumacija **12**
2.1 Osnovne metode sumacije 12
2.2 Analiza grešaka 14
2.3 Statističke procjene točnosti 19

3 Kompenzirano sumiranje **20**
3.1 Opis metode 20
3.2 Kahanov algoritam 21
3.3 Priestov algoritam 23

4 Ostale sumacijske metode **25**
4.1 Akumulacijska sumacija 25
4.2 Destilacijski algoritam 27
4.3 Izbor metode 30

5 Primjeri u MATLAB-u **32**
5.1 Eulerova metoda 32
5.2 Numerički eksperimenti 33
5.3 MDS 36

Bibliografija **38**

Uvod

”I do hate sums.
There is no greater mistake than to call arithmetic an exact science.
There are . . . hidden laws of Number
which it requires a mind like mine to percive.
For instance, if you add a sum from the bottom up,
and then again from the top down,
the result is always different.”

Autor ovog citata je gospođa La Touche. Smatra da se aritmetiku ne može zvati egzaktnom znanošću, jer različiti redoslijed sumiranja daje različite rezultate, a početni sumandi su jednaki. Iako se to čini nelogično i krši sva pravila matematike, istina je. Aritmetika u kojoj se to događa zove se aritmetika konačne preciznosti na računalu. Računalo je jedan ograničen stroj, jer ima ograničenu količinu memorijskog prostora gdje se pohranjuju polazni podaci, međurezultati i rezultati računanja. Umjesto skupa realnih brojeva, koristimo njegovu aproksimaciju pomoću konačno mnogo prikazivih brojeva, takozvanih brojeva s pomičnom točkom. Slijedi da računske operacije ne možemo izvršavati točno i rezultat ne možemo po volji dobro aproksimirati. Javljaju se mnoge greške, koje ćemo kasnije spomenuti.

U ovom radu se bavimo raznim sumacijama brojeva s pomičnom točkom, koje su prisutne po svuda u znanstvenom računanju. Pojavljuju se kod izvrednjavanja skalarnih produkata, srednjih vrijednosti, varijanci, normi, i raznih nelinearnih funkcija. Postoji mnogo algoritama za računanje sumacija, koji koriste razne redoslijede sumanada. Neki algoritmi su fokusirani na visoku točnost izračunatog rezultata, dok su drugi pogodni za brzo i efikasno paralelno izvršavanje. U ovoj radnji ćemo analizirati greške nekih sumacijskih algoritama, kao i neke opće smjernice kod odabira pogodnog algoritma u pojedinim slučajevima.

Glavna tema ovog rada je točnost algoritama koje koristimo pri sumiranju brojeva s pomičnom točkom, u aritmetici konačne preciznosti na računalu. U prvom poglavlju ćemo opisati konačnu aritmetiku računala. Objasnit ćemo pojmove poput apsolutne i relativne greške, greške unaprijed i unazad, uvjetovanost. Definirat ćemo standardni model aritme-

tike konačne preciznosti i primijenit ćemo ga na operaciju skalarnog produkta. U drugom poglavlju ćemo definirati tri osnovne metode sumacije. Analizirat ćemo njihove greške i koristit ćemo statističku procjenu točnosti kako bismo usporedili te metode. U trećem poglavlju ćemo opisati još jednu metodu, metodu kompenziranog sumiranja, koja je točnija od prethodnih. Uz pomoć Kahanovog i Priestovog algoritma, detaljnije ćemo analizirati tu metodu. U četvrtom poglavlju ćemo uvesti još dvije sumacijske metode, akumulacijsku sumaciju i sumaciju preko destilacijskog algoritma. Na kraju tog poglavlja ćemo izabrati, ako je to moguće, koja je metoda najbolja. Na kraju rada, u petom poglavlju, svu prethodnu teoriju ćemo ilustrirati konkretnim primjerima izrađenim u MATLAB-u.

Poglavlje 1

Konačna aritmetika računala

1.1 Mjere za grešku

Neka je x realan broj. Označimo sa $\hat{x}$ aproksimaciju broja x, dobivenu nekom numeričkom metodom. Grešku aproksimacije zapisujemo kao $x - \hat{x}$. Budući da greška može biti i pozitivna i negativna, koristimo *apsolutnu grešku* kao jednu od mjera za točnost.

Apsolutnu grešku definiramo sljedećom formulom:

$$G_a(\hat{x}) = |x - \hat{x}|.$$

U praksi je bolje koristiti *relativnu grešku*, koju definiramo na sljedeći način:

$$G_r(\hat{x}) = \frac{|x - \hat{x}|}{|x|},$$

gdje je x različit od nule.

Relativna greška se može zapisati i kao:

$$G_r(\hat{x}) = |\rho|, \quad \hat{x} = x(1 + \rho).$$

Ona mjeri koliko se faktor $(1+\rho)$ razlikuje od 1 po apsolutnoj vrijednosti. Takva jednadžba se koristi u analizi grešaka u artimetici računala.

Apsolutnu grešku koristimo za mjerenje udaljenosti aproksimacije od točne vrijednosti. U praksi, međutim, može postojati vrlo velika razlika između x-eva pa koristimo relativnu grešku, jer ona ima svojstvo nezavisnosti o skaliranju:

$$\frac{|\alpha x - \alpha \hat{x}|}{|\alpha x|} = \frac{|\alpha| \cdot |x - \hat{x}|}{|\alpha| \cdot |x|} = \frac{|x - \hat{x}|}{|x|}, \quad \alpha \in \mathbb{R}.$$

Zbog toga se relativna greška često izražava u postocima(%) ili u promilima(‰).

Relativna greška i broj *točnih značajnih znamenki* su povezani. Relativna greška mjeri relativnu točnost s obzirom na veličinu od x, a broj točnih značajnih znamenki možemo izračunati na sljedeći način.

Prva značajna znamenka u nekom broju je prva znamenka u tom broju koja je različita od nule, gledajući slijeva nadesno. Druga značajna znamenka je sljedeća znamenka broja, uključujući nulu samo u slučaju kada iza nje slijedi znamenka različita od nule. Nule na kraju broja se zanemaruju osim ako nisu iza decimalne točke. Primjerice, u broju 7.9530 ima pet značajnih znamenki, a u broju 0.0795 ih ima tri.

Kažemo da je t broj točnih značajnih znamenki ako se x i $\hat{x}$ slažu u t značajnih znamenki. Primjerice, $t = 2$ za:

$$x = 1.0000, \quad \hat{x} = 1.0497, \quad G_r = 4.97 \cdot 10^{-2}$$
$$x = 9.0000, \quad \hat{x} = 8.9799, \quad G_r = 2.23 \cdot 10^{-3}.$$

Relativne greške se u ovom slučaju razlikuju za faktor 22 dok je broj točnih značajnih znamenki 2, jer već za 3 znamenke se razlikuju.

Jedna moguća definicija broja točnih značajnih znamenki je da aproksimacija $\hat{x}$ od x ima t točnih značajnih znamenki ako se $\hat{x}$ i x zaokružuju na isti broj od t značajnih znamenki što znači da ih treba zamijeniti s najbližim brojem koji ima t značajnih znamenki. Problem se javlja kad je recimo $x = 0.9949$ i $\hat{x} = 0.9953$. Brojevi se ne slažu u dvije značajne znamenke, ali se slažu u jednoj i u tri značajne znamenke.

Druga moguća definicija je da aproksimacija $\hat{x}$ od x ima t točnih značajnih znamenki ako je $|x - \hat{x}|$ manja od jedne polovine jedinice u t-toj značajnoj znamenci od x. Iz ove definicije slijedi da se primjerice brojevi 0.124 i 0.125 slažu u dvije značajne znamenke, iako je mnogima logičnije da se slažu u manje od dvije značajne znamenke.

Budući da postoji više definicija broja točnih značajnih znamenki, relativna greška je preciznija mjera računanja točnosti. Unatoč tome, broj točnih značajnih znamenki omogućuje koristan način za razmišljanje o točnosti aproksimacije.

1.2 Tipovi grešaka

Postoje dva tipa grešaka: *greške zbog polaznih aproksimacija* i *greške zakruživanja*.

S obzirom na izvore grešaka, tipovi grešaka su:

- greške modela
- greške u ulaznim podacima
- greške metoda za rješavanje problema
- greške aritmetike računala

Prva tri tipa pripadaju greškama zbog polaznih aproksimacija dok greške aritmetike računala pripadaju greškama zaokruživanja.

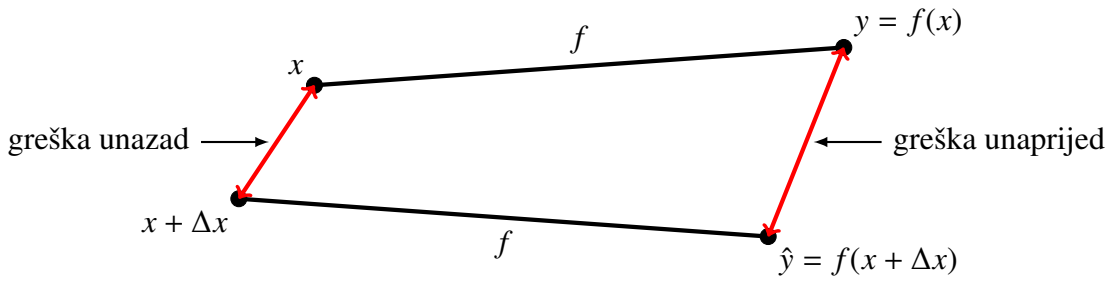
Općenito, greške zbog polaznih aproksimacija pronalazimo kod problema iz prakse. Kod grešaka modela, realan problem svodimo na matematički, pri čemu se vrše razna pojednostavljenja. Te greške se ne mogu ukloniti pa trebamo procijeniti jesu li rezultati dobiveni tom metodom zadovoljavajući. Greške u ulaznim podacima se pojavljuju zbog podataka koji se ne mogu točno izmjeriti ili ih nema smisla točno mjeriti. Greške metode za rješavanje problema se pojavljuju kad se beskonačni procesi moraju zaustaviti u konačnosti. Razlikujemo greške diskretizacije, koje nastaju prilikom zamjene neprebrojivog skupa konačnim diskretnim skupom točaka i greške odbacivanja, koje nastaju rezanjem beskonačnog niza ili reda koji odbacujemo, to jest prilikom zamjene beskonačnog diskretnog skupa konačnim skupom.

Greške zaokruživanja nastaju u prikazu brojeva u računalu i aritmetici računala. Računala koriste aritmetiku konačne preciznosti, gdje je unaprijed rezerviran broj binarnih mjesta za dijelove broja, eksponent i mantisu. Stoga svaka računska operacija daje neku grešku zaokruživanja. Greške aritmetike računala, kao jedna od vrsta grešaka računala, nastaju zbog približnog računanja.

1.3 Stabilnost numeričkih algoritama

Kažemo da je neki algoritam stabilan ako je točnost izračunatih podataka približno jednaka točnosti ulaznih podataka.

Neka je g realna funkcija realne varijable i neka je $\hat{y}$ aproksimacija od $y = g(x)$, izračunata u aritmetici preciznosti u. Aritmetika zadane preciznosti će se detaljnije objasniti u potpoglavlju 1.4. Promotrimo koliko je dobra aproksimacija. Ako je $G_r(\hat{y}) \approx \epsilon$, za ϵ dovoljno mali, tada je relativna greška mala. Međutim, to nije lako postići. Promatramo Δx za koji vrijedi $\hat{y} = g(x + \Delta x)$. Zanima nas najveća vrijednost od Δx.



Slika 1.1: Greške unaprijed i unazad

Greška unazad (ili povratna greška) je:

$$\frac{|\Delta x|}{|x|}.$$

Greška unaprijed (ili greška) je apsolutna ili relativna greška od $\hat{y}$. Na slici 1.1, koja je preuzeta iz [2], je prikazan odnos greške unaprijed i unazad. Pri određivanju rješenja, računamo i povratnu grešku rješenja, za koju tražimo neku ogradu. Taj postupak ograničavanja zovemo *povratna analiza greške*. Vrijednost funkcije $y = g(x)$ nalazimo metodom koja je povratno stabilna ako vrijedi $\hat{y} = g(x + \Delta x)$, za malo Δx. *Miješana naprijed-nazad greška* se dobiva iz sljedećeg rezultata:

$$\hat{y} + \Delta y = g(x + \Delta x), \;\; |\Delta y| \leq \xi |y|, \;\; |\Delta x| \leq \eta |x|, \tag{1.1}$$

uz male η i ξ. Razlika između izračunatog rješenja $\hat{y}$ i $\hat{y} + \Delta y$ je minimalna ako su η i ξ dovoljno mali. Možemo reći da je neki algoritam numerički stabilan ako je stabilan u smislu relacije (1.1). Greška unaprijed i greška unazad su povezane pojmom *uvjetovanosti*, koja označava osjetljivost problema na ulazne podatke. Ako pretpostavimo da je funkcija g dva puta neprekidno derivabilna, tada, uz korištenje Taylorovog reda, slijedi:

$$\Delta y = g(x + \Delta x) - g(x) \quad = g'(x)\Delta x + \frac{g''(x + \theta \Delta x)}{2!}(\Delta x)^2, \quad \theta \in (0, 1).$$

Za male vrijednosti Δx, vrijedi:

$$\Delta y = g'(x)\Delta x + O\left((\Delta x)^2\right).$$

Podijelimo obje strane jednadžbe sa y, odnosno sa $g(x)$:

$$\frac{\Delta y}{y} = \frac{x g'(x)}{g(x)} \frac{\Delta x}{x} + O\left(\left(\frac{\Delta x}{x}\right)^2\right),$$

gdje definiramo:

$$(\text{uvj}\, g)(x) := \left| \frac{x g'(x)}{g(x)} \right|$$

kao relativnu uvjetovanost funkcije g.

Kad je sve dobro definirano, vrijedi pravilo:

$$\text{greška unaprijed} \lesssim \text{uvjetovanost} \times \text{greška unazad}.$$

Tu je vidljiv utjecaj faktora uvjetovanosti problema, to jest čak i kad je greška unazad mala, ako je problem loše uvjetovan, greška unaprijed će biti velika.

Definicija 1.3.1. *Ako metoda daje rješenja s greškama unaprijed, koja su sličnog reda veličine kao ona koja se dobiju primjenom stabilne metode unazad, onda tu metodu nazivamo stabilnom unaprijed.*

Iz definicije stabilnosti unaprijed, unazad stabilna metoda je stabilna i unaprijed kad je problem dobro uvjetovan, a obrat ne mora vrijediti. Primjerice Cramerovo pravilo za 2×2 sustave je stabilno unaprijed, a nije unazad.

1.4 Binarna aritmetika s pomičnom točkom

Budući da u računalo možemo spremiti konačan broj podataka, to jest memorija računala je ograničena, umjesto realnih brojeva, koristimo njihovu aproksimaciju. Računalo koristi konačan skup brojeva koji aproksimiraju skup realnih brojeva, to jest koristimo konačan podskup skupa $\mathbb{R}$. Označimo taj podskup sa F. Znanstveni zapis broja $x \in F$ je:

$$x = \pm m \cdot \beta^{e-t},$$

gdje je β baza brojevnog sustava koje koristi računalo, t preciznost i e eksponent. Mantisa ili signifikand m je cijeli broj takav da vrijedi $0 \leq m \leq \beta^t - 1$. Pretpostavimo da vrijedi $e_{min} \leq e \leq e_{max}$. Računala koja imaju IEEE standard, umjesto baze β, koriste bazu 2. Raspon brojeva koji su prikazivi u računalu je:

$$\beta^{e_{min}-1} \leq |x| \leq \beta^{e_{max}} \left(1 - \beta^{-t}\right).$$

Neka je $fl(x)$ reprezentacija realnog broja x u računalu. Točnije, ako je x realan broj koji se nalazi unutar raspona brojeva koji su prikazivi u računalu, onda se umjesto x sprema zaokruženi prikazivi broj $fl(x)$. Pri tome nastaje greška zaokruživanja koja se zove jedinična greška zaokruživanja. Standardna oznaka je u, a relativna greška koja je napravljena tim zaokruživanjem je ρ.

Teorem 1.4.1. *Ako se $x \in \mathbb{R}$ nalazi u F, tada $fl(x) = x(1+\rho)$, $|\rho| < u$.*

Dokaz. Dokaz teorema je dan u [4]. □

Aritmetika u kojoj se odvija prikaz realnih brojeva s bazom 2 u računalu zove se *binarna aritmetika brojeva s pomičnom točkom*.

Standardni model aritmetike konačne preciznosti je:

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\rho),\ |\rho| \leq u \tag{1.2}$$

gdje op označava operacije iz skupa $\{+, -, *, /\}$, a x i y su reprezentabilni brojevi iz F.

Model se može modificirati. Jedna od modifikacija je:

$$fl(x \text{ op } y) = \frac{x \text{ op } y}{1+\delta}, \quad |\delta| \leq u. \tag{1.3}$$

Ova ocjena relativne greške odgovara idealnom izvršavanju aritmetičkih operacije, to jest, aritmetičke operacije se prvo izvrše pa tek onda se taj rezultat zaokružuje. Međutim to nije stvarno tako, pogotovo kod dijeljenja, jer njegov egzaktni rezultat može imati beskonačan binarni prikaz. Zbog toga je važno da je ocjena relativne greške zaokruživanja ista, a način na koji se računa nije bitan.

Ako su $x \ \ op \ y$ u dozvoljenom rasponu, tada $fl(x \ \ op \ y)$ ima malu relativnu grešku s obzirom na $x \ \ op \ y$.

Postoji i model u artimetici konačne preciznosti koji je jednostavniji za korištenje, ali ima slabije uvjete. To je takozvani *model bez sigurnosne znamenke*:

$$\begin{aligned} fl(x \pm y) &= x(1+\alpha) \pm y(1+\beta), \quad |\alpha|, |\beta| \leq u \\ fl(x \circ y) &= (x \circ y)(1+\delta), \quad |\delta| \leq u, \quad \text{op } = *, / \end{aligned} \tag{1.4}$$

1.5 Skalarni produkt

Standardni model aritmetike konačne preciznosti, koji smo definirali u prethodnom poglavlju, primjenjujemo na osnovne matrične operacije poput skalarnog produkta. Također ćemo ilustrirati grešku unaprijed i unazad.

Definicija 1.5.1. *Neka su x i y proizvoljni vektori u $\mathbb{R}^n$. Skalarni produkt vektora x i y je broj*

$$x \cdot y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n = x^\tau y.$$

Lema 1.5.2. *Ako $|\delta_i| \leq u$ i $\rho_i = \pm 1$ za $i = 1, 2, \ldots n$, i $nu < 1$, tada*

$$\prod_{i=1}^{n} (1+\delta_i)^{\rho_i} = 1 + \theta_n,$$

gdje

$$|\theta_n| \leq \frac{nu}{1-nu} =: \gamma_n.$$

Dokaz. Koristimo matematičku indukciju kako bismo dokazali tvrdnju. Korak indukcije ima dva slučaja. Prvi slučaj je za $\rho_n = 1$. Tada vrijedi:

$$\begin{aligned}
\prod_{i=1}^{n}(1+\delta_i) &= (1+\delta_n)(1+\theta_{n-1}) = 1+\theta_n \\
\theta_n &= \delta_n + (1+\delta_n)\theta_{n-1} \\
|\theta_n| &\leq u + (1+u)\frac{(n-1)u}{1-(n-1)u} \\
&= \frac{u(1-(n-1)u) + (1+u)(n-1)u}{1-(n-1)u} \\
&= \frac{nu}{1-(n-1)u} \leq \gamma_n.
\end{aligned}$$

Drugi slučaj, za $\rho_n = -1$, se pokaže na sličan način:

$$\begin{aligned}
\prod_{i=1}^{n}(1+\delta_i) &= (1+\delta_n)^{-1}(1+\theta_{n-1}) = 1+\theta_n \\
\theta_n &= -\frac{\delta_n}{1+\delta_n} + \frac{1}{1+\delta_n}\theta_{n-1} \\
|\theta_n| &\leq \frac{u}{1-u} + \frac{1}{1-u}\frac{(n-1)u}{1-(n-1)u} \\
&= \frac{u(1-(n-1)u) + (n-1)u}{(1-u)(1-(n-1)u)} \\
&= \frac{nu-(n-1)u^2}{1-nu+(n-1)u^2} \leq \gamma_n.
\end{aligned}$$

□

Označimo sa $s_n = x^\tau y$ skalarni produkt, prema definiciji 1.5.1. Neka je $s_i = x_1y_1 + x_2y_2 + \ldots + x_iy_i$ i-ta parcijalna suma. Koristeći (1.2) i pretpostavku $1+\delta_i = 1 \pm \delta,\ i = 1, 2, \ldots, n$, dobivamo:

$$\hat{s}_n = x_1y_1(1\pm\delta)^n + x_2y_2(1\pm\delta)^n + x_3y_3(1\pm\delta)^{n-1} + \cdots + x_ny_n(1\pm\delta)^2 \tag{1.5}$$

Detaljan raspis se nalazi u [4, str.62].

Primjenjujući prethodnu lemu na (1.5), dobivamo sljedeće:

$$\hat{s}_n = x_1y_1(1+\theta_n) + x_2y_2(1+\theta_n') + x_3y_3(1+\theta_{n-1}) + \cdots + x_ny_n(1+\theta_2).$$

To je rezultat greške unazad i tvrdi da je skalarni produkt jedinstven za perturbirani skup podataka $x_1, x_2, \ldots, x_n, y_1(1+\theta_n), y_2(1+\theta_n'), \ldots, y_n(1+\theta_2)$. Rezultat možemo prikazati

vektorski:

$$fl(x^\tau y) = (x + \Delta x)^\tau y = x^\tau (y + \Delta y), \quad |\Delta x| \leq \gamma_n |x|, \quad |\Delta y| \leq \gamma_n |y|.$$

Ograda za grešku unaprijed slijedi i dana je sljedećom formulom:

$$|x^\tau y - fl(x^\tau y)| \leq \gamma_n \sum_{i=1}^{n} |x_i y_i| = \gamma_n |x|^\tau |y|. \tag{1.6}$$

Ako $y = x$, promatramo sumu kvadrata x_τ, dobivena je velika relativna točnost. Općenito, velika relativna točnost nije garantirana za $|x^\tau y| \ll |x|^\tau |y|$.

U mnogim kompliciranim analizama, baziranim na lemi 1.5.2, potrebno je manipulirati sa članovima $1 + \theta_k$ i γ_k. Nužna pravila koja se koriste daje sljedeća lema.

Lema 1.5.3. *Za proizvoljan cijeli broj k, neka je θ_k veličina takva da $|\theta_k| \leq \gamma_k = \frac{ku}{1-ku}$. Tada vrijede sljedeće relacije:*

$$\begin{aligned}
(1 + \theta_k)\left(1 + \theta_j\right) &= 1 + \theta_{k+j} \\
\frac{1 + \theta_k}{1 + \theta_j} &= \begin{cases} 1 + \theta_{k+j}, & j \leq k \\ 1 + \theta_{k+2j}, & j > k \end{cases} \\
\gamma_k \gamma_j &\leq \gamma_{\min(k,j)}, \ za\ max\ (j,k)u \leq 1/2 \\
i\gamma_k &\leq \gamma_{ik} \\
\gamma_k + u &\leq \gamma_{k+1} \\
\gamma_k + \gamma_j + \gamma_k \gamma_j &\leq \gamma_{k+j}.
\end{aligned}$$

Dokaz. Dokaz leme je dan u [4]. □

Sljedeća lema daje još jednu moguću ogradu za grešku.

Lema 1.5.4. *Ako je $|\delta_i| \leq u$, za $i = 1, 2, \ldots, n$ i $nu \leq 0.01$, tada vrijedi sljedeće:*

$$\prod_{i=1}^{n} (1 + \delta_i) = 1 + \eta_n,$$

gdje je $|\eta| \leq 1.01nu$.

Dokaz.

$$|\eta_n| = \left| \prod_{i=1}^{n} [(1 + \delta_i) - 1] \right| \leq (1 + u)^n - 1.$$

Znamo da vrijedi $1 + x \leq \exp x$, za $x \geq 0$ pa slijedi $(1 + u)^n < \exp nu$. Supstitucijom u prethodnu nejednadžbu i razvojem u red, dobivamo:

$$\begin{aligned} |\eta_n| \leq (1+u)^n - 1 &< nu + \frac{(nu)^2}{2!} + \frac{(nu)^s}{3!} + \cdots \\ &< nu\left(1 + \frac{nu}{2} + \left(\frac{nu}{2}\right)^2 + \left(\frac{nu}{2}\right)^3 + \cdots\right) \\ &= nu\frac{1}{1 - nu/2} \\ &\leq nu\frac{1}{0.995} < 1.01nu \end{aligned}$$

□

Ova lema daje jaču ogradu nego lema 1.5.2, jer vrijedi sljedeće:

$$|\theta_n| \leq \frac{nu}{1 - nu} < \frac{nu}{0.99} = 1.0101\ldots nu.$$

Također, ogradu (1.6) možemo zapisati ovako:

$$\left|x^T y - fl\left(x^T y\right)\right| \leq 1.01nu|x|^T|y|.$$

Poglavlje 2

Analiza grešaka osnovnih metoda sumacija

2.1 Osnovne metode sumacije

Sume brojeva s pomičnom točkom se često koriste u znanstvenom računanju. Pojavljuju se u računanju skalarnih produkata, varijanci, normi, očekivanja, nelinearnih funkcija. Zbog svoje sveprisutnosti, razlikujemo više vrsta sumacija.

Označimo sumu sa s. Tada je suma n brojeva s pomičnom točkom dana sa:

$$s_n = x_1 + x_2 + \cdots + x_n = \sum_{i=1}^{n} x_i,$$

gdje su $x_i = fl(x_i),\ 1 \leq i \leq n$.

Prva metoda sumacije je **rekurzivna sumacija**. Prikazujemo ju sljedećim algoritmom:

$$\begin{aligned} &s = 0 \\ &\text{for } i = 1 : n \\ &\quad s = s + x_i \\ &\text{end} \end{aligned}.$$

Specifični poretci brojeva su rastući i padajući. To su redom:

$$\begin{aligned} |x_1| \leq |x_2| \leq \cdots \leq |x_n|, \\ |x_1| \geq |x_2| \geq \cdots \geq |x_n|. \end{aligned}$$

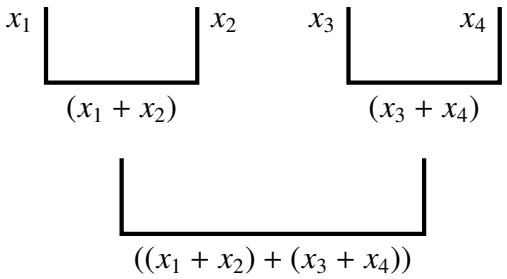
Sumacija po parovima ili **kaskadna sumacija** je druga metoda sumacije. Kako i sam naziv kaže, brojevi se sumiraju u parovima. Pravilo po kojem se parovi sumiraju je sljedeće:

$$y_i = x_{2i-1} + x_{2i},$$

gdje je $i = 1, 2, \ldots \left\lfloor \frac{n}{2} \right\rfloor$. Ako je n neparan, onda je posljednji član sume jednak $y_{\frac{n+1}{2}} = x_n$. Taj postupak se ponavlja rekurzivno za $y_i,\ i = 1, 2, \ldots \left\lceil \frac{n+1}{2} \right\rceil$. Primjerice, za $n = 4$, suma je jednaka:

$$s_4 = ((x_1 + x_2) + (x_3 + x_4)).$$

Primjer sumiranja se vidi na slici 2.1.

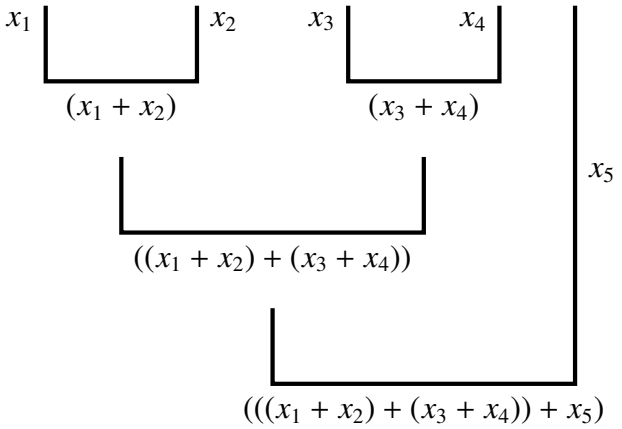


Slika 2.1: Sumacija po parovima za $n = 4$

Za $n = 5$, suma je jednaka:

$$s_5 = (((x_1 + x_2) + (x_3 + x_4)) + x_5).$$

Primjer sumiranja se vidi na slici 2.2.



Slika 2.2: Sumacija po parovima $n = 5$

U trećoj metodi sumacije, **sumacija umetanjem**, brojevi su poredani uzlazno, to jest vrijedi:

$$|x_1| \leq |x_2| \leq \cdots \leq |x_n|\,.$$

Prvo se sumiraju x_1 i x_2. Njihova suma se sprema u niz $x_2, x_3, \ldots, x_n$ tako da taj niz bude rastući. Zatim se sumiraju x_2 i x_3 i spremaju se u niz $x_3, x_4, \ldots, x_n$, koji također mora biti rastući. Ponavljanjem postupka, dobiva se konačna suma. Metodu možemo pokazati na sljedećem primjeru:

$$1 \quad 2 \quad 5 \quad 9 \rightarrow \mathbf{3} \quad 5 \quad 9 \rightarrow \mathbf{8} \quad 9 \rightarrow 17$$

Ova metoda se u nekim slučajevima može svesti na prethodne dvije metode pa je nećemo puno promatrati. Ako $x_i = 3^{i-1}$, tada je metoda umetanja ekvivalentna rekurzivnoj metodi, i slijedi:

$$1 \quad 3 \quad 9 \rightarrow \mathbf{4} \quad 9 \rightarrow 15$$

Ako vrijedi $1 \leq x_1 < x_2 < \ldots < x_n \leq 2$, umetanje se izvodi na kraju niza. Tada je metoda ekvivalentna sumaciji po parovima, za n koji je potencija broja 2. Na primjer, ako $0 < \epsilon < \frac{1}{2}$, tada slijedi:

$$1, 1+\epsilon, 1+2\epsilon, 1+3\epsilon \rightarrow 1+2\epsilon, 1+3\epsilon, \mathbf{2}+\epsilon \rightarrow 2+\epsilon, \mathbf{2}+\mathbf{5}\epsilon \rightarrow 4+6\epsilon.$$

2.2 Analiza grešaka

Umjesto da analiziramo svaku od prethodne tri metode sumacije posebno, konstruirat ćemo općeniti algoritam svih metoda. Svaka metoda je poseban slučaj tog algoritma. Algoritam je dan na sljedeći način:

Ulazni podaci: $x_1, x_2, \ldots, x_n$
Izlazni podaci: $s_n = \sum_{i=1}^{n} x_i$

Algorithm 1

$s = \{x_1, x_2, \ldots, x_n\}$
while $x, y \in s$ **do**
  Izbriši x, y iz s
  i dodaj $x + y$ u s.
end while
Preostalom elementu od s pridruži naziv s_n.

U algoritmu sudjeluje n pribrojnika i $n-1$ operacija zbrajanja. Zbog toga se petlja mora izvršiti $n-1$ puta.

Provjerimo jesu li prethodne metode stvarno posebni slučajevi danog algoritma. Kod rekurzivne sumacije, u svakom koraku uzmimo da je x jednak sumi koja je nastala u prethodnom koraku. Sumacija u parovima dobiva se pomoću $\lceil \log_2 n \rceil$ grupa izvršavanja petlje. U svakoj grupi svi članovi od s su složeni u parove, a parovi se zbroje. Metoda umetanja je

po svojoj definiciji jedan od slučajeva algoritma, jer suma dva broja zamjenjuje jedan broj tijekom postupka sumiranja.

Analizirajmo sada greške. Označimo rezultat dobiven i-tom iteracijom petlje sa $t_i = x_{i_1} + y_{i_1}$. Neka je $\hat{t}_i$ odgovarajući broj s pomičnom točkom. Uvrstimo u (1.3), $fl(x \text{ op } y) = \hat{t}$ i op = +, i raspišimo po elementima:

$$\hat{t}_i = \frac{x_{i_1} + y_{i_1}}{1 + \delta_i}, \quad |\delta_i| \leq u, \quad i = 1, 2, \ldots, n-1.$$

Lokalna greška koja pritom nastaje je $\delta_i \hat{t}_i$, a ukupna greška je suma svih lokalnih grešaka. Kako je sumiranje linearno, suma lokalnih grešaka se jednostavno izračuna i iznosi:

$$G_n := s_n - \hat{s}_n = \sum_{i=1}^{n-1} \delta_i \hat{t}_i. \tag{2.1}$$

Zbog uvjeta $|\delta_i| \leq u$, najmanja moguća ograda te greške je:

$$|G_n| \leq u \sum_{i=1}^{n-1} \left|\hat{t}_i\right|. \tag{2.2}$$

Također, možemo ograditi $\hat{t}_i$ sa $\left|\hat{t}_i\right| \leq \sum_{j=1}^{n} |x_j| + O(u)$, za svaki i. Uvrštavanjem te ograde u (2.2), dobivamo slabiju ogradu:

$$|G_n| \leq (n-1)u \sum_{i=1}^{n} |x_i| + O\left(u^2\right). \tag{2.3}$$

Takvo ograničenje predstavlja ogradu greške unaprijed. Greška unazad pokazuje sljedeće:

$$\hat{s}_n = \sum_{i=1}^{n} x_i(1 + \eta_i), \ |\eta_i| \leq \gamma_{n-1}, \tag{2.4}$$

gdje se x_i pojavljuje u ne više od $n-1$ sumiranja.

Iz (2.1) i (2.2), zaključujemo da kod odabiranja metode sumacije s ciljem dobivanja što točnije sume s_n, treba izabrati onu koja daje što manje apsolutne vrijednosti međurezultata, to jest, cilj je minimizirati apsolutnu vrijednosti suma t_i. Međutim, to je teško postići, jer minimizacija granice u (2.2) je vrlo zahtjevna za rješavanje.

Promotrimo detaljnije rekurzivnu sumaciju. Označimo $\hat{s}_1 = s_1$. Prema (1.2), slijedi:

$$\hat{s}_2 = f\ell\,(x_1 + x_2) = (x_1 + x_2)\,(1 + \epsilon_1) = x_1\,(1 + \epsilon_1) + x_2\,(1 + \epsilon_1)\,, \quad |\epsilon_1| \leq u.$$

Ponavljajući prethodni postupak, dobivamo:

$$\begin{aligned}\hat{s}_n =& f\ell\left(\hat{s}_{n-1} + x_n\right)\\ =& \left(\hat{s}_{n-1} + x_n\right)\left(1+\epsilon_{n-1}\right)\\ =& x_1\left(1+\epsilon_1\right)\left(1+\epsilon_2\right)\cdots\left(1+\epsilon_{n-1}\right) + x_2\left(1+\epsilon_1\right)\left(1+\epsilon_2\right)\cdots\left(1+\epsilon_{n-1}\right)\\ &+ x_3\left(1+\epsilon_2\right)\cdots\left(1+\epsilon_{n-1}\right) + \cdots + x_{n-1}\left(1+\epsilon_{n-2}\right)\left(1+\epsilon_{n-1}\right) + x_n\left(1+\epsilon_{n-1}\right),\end{aligned}$$

za svaki $|\epsilon_i| \leq u$. Supstituirajmo sljedeće:

$$\begin{aligned}1+\eta_1 &= \left(1+\epsilon_1\right)\left(1+\xi_2\right)\cdots\left(1+\epsilon_{n-1}\right)\\ 1+\eta_2 &= \left(1+\epsilon_1\right)\left(1+\epsilon_2\right)\cdots\left(1+\epsilon_{n-1}\right)\\ 1+\eta_3 &= \left(1+\epsilon_2\right)\cdots\left(1+\epsilon_{n-1}\right)\\ &\vdots\\ 1+\eta_{n-1} &= \left(1+\epsilon_{n-2}\right)\left(1+\epsilon_{n-1}\right)\\ 1+\eta_n &= 1+\epsilon_{n-1}\end{aligned} \quad .$$

Dobivamo (2.4), pri čemu η_i ocjenjujemo pomoću leme 1.5.2, ili još bolje sa lemom 1.5.4:

$$|\eta_i| \leq 1.01 \cdot \begin{cases} (n-1)u, & i = 1\\ (n-i+1)u, & 2 \leq i \leq n\end{cases}$$

Iz prethodne nejednakosti i (2.4) slijedi da je rekurzivni algoritam stabilan unazad, jer su sve ocjene za relativne greške perturbiranih ulaznih podataka manje od nu. Provjerimo da je algoritam stabilan unaprijed.

Oduzmimo egzaktnu sumu s_n od nove sume $\hat{s_n}$ kako bi dobili grešku:

$$\hat{s}_{\mathrm{n}} - s_{\mathrm{n}} = \sum_{i=1}^{n} x_i \cdot \eta_i.$$

Ocijenimo apsolutnu grešku:

$$\begin{aligned}|\hat{s}_n - s_n| &\leq \sum_{i=1}^{n} |x_i| \cdot |\eta_i|\\ &\leq \max_i \{|\eta_i|\} \sum_{i=1}^{n} |x_i|\\ &\leq \left(\sum_{i=1}^{n} |x_i|\right) \cdot 1.01(n-1)\mathrm{u}.\end{aligned} \tag{2.5}$$

Jednakost se postiže za $x_i > 0$ i $\eta_i = \eta_j$, za svaki $i, j = 1, 2, \ldots, n$, $i \neq j$.

Ocijenimo relativnu grešku od $\hat{s_n}$ kao aproksimaciju od s_n:

$$\frac{|\hat{s}_\mathrm{n} - s_\mathrm{n}|}{|s_\mathrm{n}|} \leq \mathrm{uvj}\,(s_\mathrm{n})\, 1.01(\mathrm{n}-1)u,$$

$$\mathrm{uvj}\,(s_n) = \frac{|x_1| + |x_2| + \cdots + |x_n|}{|x_1 + x_2 + \cdots + x_n|}. \tag{2.6}$$

Broj uvjetovanosti od s_n, uvj (s_n), može biti proizvoljno velik pa ne vrijedi stabilnost unaprijed. Ako svi x_i imaju isti predznak, onda uvj $(s_n) = 1$ i greška unaprijed je mala pa je algoritam stabilan unaprijed.

Umjesto sumiranja redom od x_1 do x_n, redoslijed sumiranja može biti proizvoljan. Ocjene grešaka će biti iste i vrijedit će (2.4), gdje x_1 i x_2 zamijenimo sa sumandima koji se prvo zbrajaju, x_3 zamijenimo sa onim sumandom koji se sljedeći dodaje toj sumi, i tako redom.

Članovi sume mogu bit različitih predznaka i različitog redoslijeda. Pretpostavimo da su svi članovi istog predznaka. Postavlja se pitanje kakav redoslijed sumacije je potreban da bi relativna greška bila najmanja. Uočimo da je traženje najmanje relativne greške ekvivalentno traženju najmanje apsolutne greške, jer s_n ne ovisi o redoslijedu sumiranja. Nađimo redoslijed indeksa $k_1, k_2, \ldots, k_n$ koji daje najbolju ocjenu za $\left|x_{k_1}\eta_1 + x_{k_2}\eta_2 + \cdots + x_{k_{n-1}}\eta_{n-1}\right|$. Ako je niz $|x_1|, |x_2|, \ldots, |x_n|$ nenegativan, najbolji redoslijed rekurzivne sumacije je rastući, u smislu da je ograda greške unaprijed najmanja. Promotrimo padajući redoslijed. Ako su svi brojevi u sumi pozitivni, ograda (2.2) može biti i veća nego kod rastućeg redoslijeda. Pogledajmo sljedeći primjer iz [4].

Neka je $n = 4$ i $x = \left[\ 1, \quad M, \quad 2M, \quad -3M\ \right]$, gdje je M broj s pomičnom točkom takav da vrijedi:

$$fl(1 + M) = M,\ M > \frac{1}{u}.$$

S rastućim poretkom dobivamo sumu:

$$\hat{s}_n = fl(1 + M + 2M - 3M) = 0$$

S padajućim poretkom dobivamo sumu:

$$\hat{s}_n = fl(-3M + 2M + M + 1) = 1$$

Razlikujemo još i "Psum" poredak, koji minimizira redom $|x_1|, |\hat{s}_2|, \ldots, |\hat{s}_{n-1}|$. "Psum" poretkom dobivamo sumu:

$$\hat{s}_n = fl(1 + M - 3M + 2M) = 0$$

Iz primjera se vidi da jedino padajući redoslijed daje točno rješenje i nema grešaka zaokruživanja, dok rastući i "Psum" poredak imaju relativnu grešku 1. Razlog tome je što

se poslije katastrofalnog kraćenja, to jest zbrajanja brojeva različitih predznaka koji su po apsolutnoj vrijednosti vrlo bliski, dodaje 1 i time se zadržava važna informacija.

Supstituirajmo $\mu = \sum_{i=2}^{n} |\hat{s}_i|$ u relaciju (2.2) koristeći $x = \begin{bmatrix} 1, & M, & 2M, & -3M \end{bmatrix}$. Dobivamo redom vrijednosti $\mu = 4M$ za rastući redoslijed, $\mu = 3M$ za "Psum" redoslijed, $\mu = M + 1$, za padajući redoslijed. Slijedi da padajući redoslijed daje najtočnije rješenje, ali ogradu koju daje je dosta pesimistična, jer nema zaokruživanja grešaka u ovom slučaju.

Zaključujemo da kad vrijedi $\left|\sum_{i=1}^{n} x_i\right| \ll \sum_{i=1}^{n} |x_i|$, to jest kad se pojavljuje katastrofalno kraćenje u sumi, padajući redoslijed daje najtočniji rezultat, ali ne tako dobru ogradu za grešku, jer u ovom slučaju nema grešaka zaokruživanja. Numerički primjer ćemo opisati u zadnjem poglavlju (vidi tablicu 5.1).

Sumiranje po parovima daje najmanju ocjenu greške, ali na računalu zahtijeva cijelo polje dodatnih lokacija u memoriji. U tom algoritmu svaki sumand sudjeluje u najviše $\log_2(n)$ zbrajanja. Pokažimo da će i ograda (2.4), za svaki η_i, biti proporcionalna tom broju.

Pretpostavimo da vrijedi $n = 2^r, r \geq 2$. Za razliku od rekurzivne sumacije, svaki pribrojnik se sumira točno $\log_2(n)$ puta. U prvom sumiranju brojeva $x_i, i = 1, 2, \ldots, n$ dobivamo sljedeće:

$$z_1 = x_1 + x_2, \quad z_2 = x_3 + x_4, \quad \ldots, \quad z_{m_1} = x_{n-1} + x_n, \quad m_1 = \frac{n}{2} = 2^{r-1}.$$

U drugom dobivamo:

$$w_1 = z_1 + z_2, \quad w_2 = z_3 + z_4, \quad \ldots, \quad w_{m_2} = z_{m_1-1} + z_{m_1}, \quad m_2 = \frac{m_1}{2} = 2^{r-2}.$$

Nakon $r - 1$ koraka, dobivamo dva člana, koja zbrojimo u r-tom koraku i spremimo u s_n. Sumu $\hat{s_n}$ onda možemo zapisati kao:

$$\hat{s}_n = \sum_{i=1}^{n} x_i \prod_{r=1}^{\log_2 n} \left(1 + \delta_r^{(i)}\right), \quad \left|\delta_r^{(i)}\right| \leq u.$$

Slijedi da je ograda dana sa:

$$|\hat{s}_n - s_n| \leq \gamma_{\log_2 n} \sum_{i=1}^{n} |x_i| \tag{2.7}$$

Kako je ograda proporcionalna sa $\log_2(n)$, a ne sa n, ona je manja od ograde (2.3). Slijedi da je ovo najjača ograda za početni općeniti algoritam.

Metoda umetanja pokušava minimizirati članove $\left|\hat{t}_1\right|, \ldots, \left|\hat{t}_{n-1}\right|$ iz (2.2). Kad su svi x_i nenegativni, onda metoda minimizira tu granicu u svim slučajevima početnog algoritma.

2.3 Statističke procjene točnosti

Ograda za zaokruživanje grešaka može biti jako pesimistična, jer je ona dana za najgori mogući slučaj. Alternativno, možemo statistički procijeniti greške kako bi usporedili sumacijske metode.

Robertazzi i Schwartz su proveli statističku analizu za tri osnovne sumacijske metode, za slučaj nenegativnih x_i. Pretpostavili su da su relativne greške u sumaciji brojeva s pomičnom točkom statistički neovisne, sa očekivanjem nula i standardnom devijacijom σ^2. Za nenegativne x_i, promatramo uniformnu distribuciju na intervalu $[0, 2\mu]$ i eksponencijalnu distribuciju sa očekivanjem μ. Robertazzi i Schwartz su, uz mnogo pojednostavljivanja pretpostavki, procijenili očekivanje kvadratne greške, to jest varijance od apsolutne greške za izračunate sume:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

Točnije, za sume dobivene rekurzivnom sumacijom sa proizvoljnim, rastućim ili padajućim redoslijedom sumiranja, za sumacijom umetanjem i sumacijom po parovima sa rastućim redoslijedom. Rezultati za sumacije n brojeva su prikazani u tablici 2.1, čiji su podaci preuzeti iz [4].

Distrib.	Rastući	Proizvoljan	Padajući	Umetanjem	Po parovima
Unif$(0, 2\mu)$	$0.20\mu^2n^3\sigma^2$	$0.33\mu^2n^3\sigma^2$	$0.53\mu^2n^3\sigma^2$	$2.6\mu^2n^2\sigma^2$	$2.7\mu^2n^2\sigma^2$
Exp(μ)	$0.13\mu^2n^3\sigma^2$	$0.33\mu^2n^3\sigma^2$	$0.63\mu^2n^3\sigma^2$	$2.6\mu^2n^2\sigma^2$	$4.0\mu^2n^2\sigma^2$

Tablica 2.1: Očekivanje kvadratnih grešaka za nenegativne x_i

Statističkim procjenama točnosti zaključujemo sljedeće:

- Za rekurzivnu sumaciju, redoslijed sumiranja utječe samo na konstantu u očekivanju kvadratne greške. Rastući redoslijed ima najmanju konstantu, a padajući najveću. Budući da su x_i nenegativni, takvo rangiranje je dano sa ogradom greške (2.5).

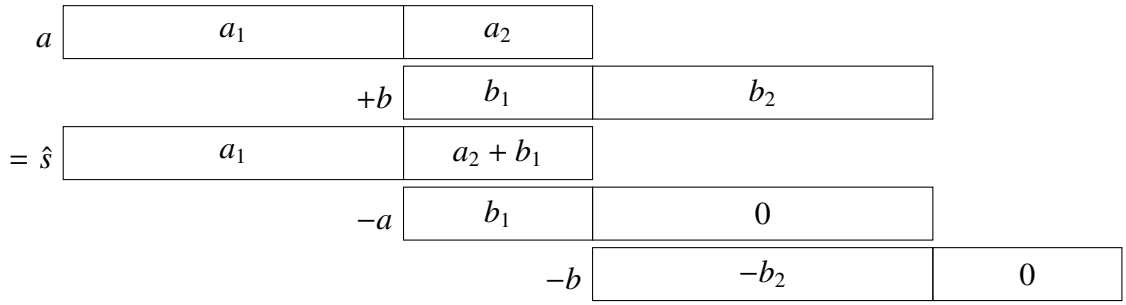
- Sumacije umetanjem i po parovima imaju očekivanje kvadratne greške proporcionalno sa n^2, a ne n^3 kao kod rekurzivne sumacije. Sumacija umetanjem također ima manju konstantu nego sumacija po parovima. To je konzistentno analizi grešaka, gdje za nenegativne x_i, metoda umetanjem zadovoljava ogradu greške ne veću od sumacije po parovima. Razlog tome je taj što sumacija umetanjem ponekad minimizira članove $|\hat{t}_1|, |\hat{t}_2|, \ldots, |\hat{t}_{n-1}|$ u ogradi (2.2). Kad su x_i nenegativni, metoda umetanjem minimizira ogradu koja vrijedi u svim slučajevima algoritma 1 pa tako i kad se radi o sumaciji po parovima.

Poglavlje 3

Kompenzirano sumiranje

3.1 Opis metode

Metoda sumacije koju još nismo spomenuli je **kompenzirana sumacija**. Poistovjećujemo ju s rekurzivnom metodom sumacije, koja sadrži i korektivni sumand koji doprinosi smanjenju greške zaokruživanja. Metodu možemo koristiti kad god se traži točna suma i kad koristimo najprecizniju aritmetiku u računalu.



Slika 3.1: Dobivanje greške zaokruživanja

Grešku zaokruživanja u sumaciji dva broja je najbolje procijenio Kahan. Način na koji je to procijenio se najbolje može objasniti pomoću slike 3.1, preuzete iz [4].

Neka su $a = fl(a)$ i $b = fl(b)$ brojevi s pomičnom točkom te neka vrijedi $|a| \geq |b|$. Označimo sumu sa $\hat{s} = fl(a+b)$. Pravokutnici prikazuju mantise brojeva a i b. Pri zbrajanju tih mantisa, mantisa broja koji je po apsolutnoj vrijednosti manji se pomakne u desno za broj bitova koji odgovara razlici eksponenata od a i b. Zbroj se zaokruži na onoliko bitova koliko mantisa dopušta. Ako pritom dođe do prekoračenja vrijednosti, mantisa se normalizira i eksponent se podesi. Zaključujemo da je greška zaokruživanja, koju smo

definirali sa e, dobivena na sljedeći način:

$$\begin{aligned} e &:= -[((a+b)-a)-b] \\ &= (a-\hat{s})+b. \end{aligned}$$

Prava greška, koja se izračuna na računalu, je dobra aproksimacija greške $(a+b)-\hat{s}$. Označimo izračunatu grešku sa $\hat{e}$. Za standardni način zaokruživanja u IEEE aritmetici, može se pokazati da vrijedi sljedeće:

$$a+b=\hat{s}+\hat{e}. \tag{3.1}$$

Iz toga slijedi da je $\hat{e}$ prava greška.

Ovaj rezultat vrijedi samo za bazu 2. Također, nema smisla računati $fl(\hat{s}+\hat{e})$, jer je $\hat{s}$ već najbolja reprezentacija sume $a+b$ u aritmetici s pomičnom točkom. Iz toga slijedi da je greška $(a+b)-\hat{s}$ broj s pomičnom točkom.

3.2 Kahanov algoritam

Kahanov algoritam u svakom koraku rekurzivnog sumacijskog algoritma, koristi korekciju e. Prvo se izračuna parcijalna suma i odmah nakon toga i njena korekcija. Ta korekcija se u sljedećem koraku dodaje novom članu koji ulazi u algoritam i to prije nego se taj član doda prethodnoj parcijalnoj sumi. Stoga, ideja algoritma je zabilježiti greške zaokruživanja i pohraniti ih natrag u algoritam. Algoritam je sljedeći.

Ulazni podaci: $x_1, x_2, \ldots, x_n$

Izlazni podaci: $s = \sum_{i=1}^{n} x_i$

Algorithm 2 Kahanov algoritam

```
s = 0
e = 0
for i = 1 to n do
   pom = s
   y = x_i + e
   s = pom + y
   e = (pom − s) + y
end for
```

Parcijalna suma, s_n, je gotovo uvijek veća od člana koji joj se dodaje pa korekcija e, koja je inače mala, može mijenjati ulazne podatka, ali ne može mijenjati parcijalnu sumu, s_{i-1}, koja je već najbolja aproksimacija od $s_{i-2}+x_{i-1}$.

Metoda ima dva nedostatka. Prvo, $\hat{e}$ nije nužno točna korekcija, jer je $a + b = \hat{s} + \hat{e}$ baziran na pretpostavci $|a| \geq |b|$. Drugo, $y = x_i + e$ ne izvršava se točno u algoritmu. Unatoč tome, korištenje korekcija doprinosi poboljšanoj ogradi greške.

Knuth je pokazao da η_i iz (2.4) za kompenziranu sumaciju zadovoljava sljedeću nejednadžu:

$$|\eta_i| \leq 2u + O\left(nu^2\right). \tag{3.2}$$

To je gotovo idealna ograda za grešku unazad. Kahan je pokazao da to vrijedi uz još jaču ogradu:

$$|\eta_i| \leq 2u + \dot{O}\left((n - i + 1)u^2\right).$$

U dokazu tvrdnji, obojica koriste standardni model aritmetike konačne preciznosti, (1.2), indukciju i neke algebarske manipulacije.

Pripadna ograda za grešku unaprijed je dana sa:

$$|G_n| \leq \left(2u + O\left(nu^2\right)\right) \sum_{i=1}^{n} |x_i| \tag{3.3}$$

Sve dok je $nu \leq 1$, konstanta u ogradi za grešku unaprijed ne ovisi o n. Zbog toga je ova ograda puno bolja od ograde kod rekurzivne sumacije, (2.3), i sumacije po parovima, (2.7). Osim, naravno, kad vrijedi $\sum_{i=1}^{n} |x_i| \gg \left|\sum_{i=1}^{n} x_i\right|$, tada kompenzirana sumacija ne garantira malu relativnu grešku.

Kompenziranu sumaciju su istraživali i mnogi drugi. Neki od njih su umjesto dodavanja svake korekcije posebno u sumu, svaku korekciju akumulirali posebno po rekurzivnoj sumaciji, i onda je ta ukupna korekcija dodana u sumu. Kielbasiński i Neumaier su pokazali da je ograda za grešku od računanja sume (2.4) sljedeća:

$$|\eta_i| \leq 2u + n^2u^2, \; nu \leq 0.1.$$

Ova ograda je slabija od ograde (3.2), jer ima dodatan faktor n. Za $n^2u \leq 0.1$, $|\eta_i| \leq 2.1u$. Jankowski, Smoktunowicz i Woźniakowski su pokazali da za takozvanu podijeli pa vladaj implementaciju kompenzirane sumacije, područje vrijednosti od n, za koje vrijedi $|\eta_i| \leq cu$, se može proširiti, uz moguće mali rast konstante c. Niti korekcijska formula (3.1) niti rezultat (3.2) za kompenziranu sumaciju ne vrijede za model aritmetike konačne preciznosti bez sigurnosne znamenke, definiran u (1.4). Također, Kahan je konstruirao primjer gdje kompenzirana sumacija nije uspjela postići (3.3) na određenom Crayovom stroju, ali to se jako rijetko događa.

Kahan je modificirao algoritam 2 tako što je naredbu $e = (pom - s) + y$, zamijenio sljedećim naredbama:

$$\begin{aligned} &f = 0 \\ &\text{if sign } (pom) = \text{sign}(y), f = (0.46 * s - s) + s, \text{ end} \\ &e = ((pom - f) - (s - f)) + y \end{aligned}$$

U knjizi [5] je pokazao da modificirani algoritam zadovoljava (3.2), i to za sve strojeve iz Sjeverne Amerike sa pomičnim hardverom. Također je objasnio konstantu 0.46, iako to može biti bilo koji broj između 0.25 i 0.50. I činjenicu da dokaz zahtjeva poznavanje dizajna poznatih strojeva, što znači da ovaj algoritam i nije neki napredak u računalnoj znanosti.

Viten'ko je pokazao da u modelu bez sigurnosne znamenke, (1.4), sumacijska metoda s optimalnom ogradom greške unaprijed, je sumacija po parovima. Kako Kahan koristi svojstva aritmetike konačne preciznosti van tog modela, ovaj zaključak ne proturječi Kahanovom rezultatu.

Dobar primjer koji pokazuje benefite kompenzirane sumacije je Eulerova metoda za obične diferencijalne jednadžbe, koju ćemo detaljnije opisati u zadnjem poglavlju.

3.3 Priestov algoritam

Priest je izveo novi algoritam iz Kahanovog algoritma. Kompenziranoj sumaciji je dodao dvije dodatne primjene procesa korekcije. Zbog toga se naziva Priestov algoritam ili dvostruka kompenzirana sumacija. Umjesto 4 zbrajanja po koraku algoritma, zbrajanje se koristi 10 puta u jednom koraku. Algoritam je istovjetan simulaciji aritmetike dvostruke preciznosti sa aritmetikom jednostruke preciznosti. Također, algoritam zahtjeva padajući poredak pribrojnika što otklanja potrebu za dodatnim logičkim testovima. Algoritam je sljedeći:

Ulazni podaci: $x_1, x_2, \ldots, x_n$
Izlazni podaci: $s_n = \sum_{i=1}^{n} x_i$

Algorithm 3 Priestov algoritam

Sortiraj $x_i,\ i = 1, 2, \ldots, n$, tako da vrijedi $|x_1| \geq |x_2| \geq \cdots \geq |x_n|$.
$s_1 = x_1$
$c_1 = 0$
for $k = 2$ **to** n **do**
  $y_k = c_{k-1} + x_k$
  $u_k = x_k - (y_k - c_{k-1})$
  $t_k = y_k + s_{k-1}$
  $v_k = y_k - (t_k - s_{k-1})$
  $z_k = u_k + v_k$
  $s_k = t_k + zk$
  $c_k = z_k - (s_k - t_k)$
end for

Priest je analizirao algoritam za t-znamenkastu aritmetiku baze β koja zadovoljava određene pretpostavke, to jest pretpostavke koje su zadovoljene u IEEE aritmetici. Pokazao je da ako vrijedi $n \leq \beta^{t-3}$, onda izračunata suma s_n zadovoljava sljedeće:

$$|s_n - \hat{s}_n| \leq 2u\,|s_n|\,.$$

To znači da je izračunata suma zapravo točna do pune preciznosti. Dokaz te tvrdnje dan je u [7, 65.str.].

Poglavlje 4

Ostale sumacijske metode

4.1 Akumulacijska sumacija

Mnogo algoritama zahtjeva računanje sume $s_n = \sum_{i=1}^{n},\ n \geq 3$, gdje su svi $x_i,\ i = 1, 2, \ldots, n$ brojevi s pomičnom točkom. U praksi je potrebno računati aproksimativnu sumu s_n, gdje se javljaju greške zaokruživanja. Wilkinsonova ograda greške (vidi [6]) za sumu akumuliranu u akumulatoru jednostruke preciznosti ne garantira malu relativnu grešku. Čak i ako je suma akumulirana u akumulatoru veće preciznosti, i dalje nije siguran radi li se o maloj relativnoj greški. Lehmer je pokazao da je velika relativna greška u akumulacijskoj sumi često rezultat katastrofalnog kraćenja. To se događa kada je srednja parcijalna suma puno veća od konačne sume. Tada jedna ili više sumacija rezultira gubitkom značajnih znamenki na čije se mjesto kasnije stavljaju nule. Kahan je otkrio da ta velika kraćenja nisu razlog nastajanju grešaka već velike srednje sume u sustavu brojeva s pomičnom točkom, s danom preciznošću.

Stoga se velike relativne greške mogu dogoditi i bez katastrofalnog kraćenja. Primjerice u velikim sumacijama, za $n > 3$, gdje su srednje sume puno veće od individualnih sumanada, ali ne veće od konačne sume. Ova vrsta grešaka se događa u numeričkoj integraciji pri korištenju velikog broja intervala. Zbog zaokruživanja brojeva, dolazi do gubitka značajnih znamenaka s desnog kraja sumanada. Što je suma veća, greška odbacivanja, to jest ograničenje broja znamenki desno od decimalne točke, raste. Stoga veći broj intervala daje manje točnu sumu.

Wolfe je predložio tehniku izbjegavanja greške odbacivanja, spomenute u poglavlju 1.2, pomoću programiranja. Zasebne varijable je postavio kao akumulatore koji drže parcijalne sume iz različitih intervala. Njegova tehnika se jednostavno programira i zahtjeva mali broj lokacija za pohranjivanje suma. Uvodi dodatne lokacije, takozvane **kaskadne akumulatore**, $s_i,\ i = 1, 2, \ldots, n$, po čemu je nazvana metoda. Bilo bi dobro kad bi se kaskadni akumulatori uveli u logički dizajn hardvera kako bi se sumiranje izvodilo uz pomoć

skupa kaskadnih akumulatora umjesto sa akumulatorima jednostruke preciznosti. Primjer možete vidjeti u [10].

Sumiranje se vrši u akumulatoru koji je na najnižoj razini, sve dok ne dođe do prekoračenja, takozvanog overflowa. Tada se suma doda sljedećem većem akumulatoru, koji je inicijaliziran na vrijednost broja koji mu se dodaje, a stari se resetira na nulu. Greške odbacivanja su na ovaj način eliminirane.

Jedna od modifikacija je opisana u [8]. Algoritam je puno jednostavniji i fleksibilniji, jer uključuje i negativne sumande. Greška odbacivanja, koja proizlazi iz sume koja je puno veća od individualnih sumanada, može biti savladana ako imamo redove akumulatora umjesto samo jednog. Vrijednost sume u svakom od tih akumulatora je restringirana tako da upadne u neki od intervala.

Pokažimo to na primjeru. Pretpostavimo da želimo zbrojiti elemente $a_1, a_2, \ldots, a_n$, za veliki n i elemente reda 1. Raspon za svaki akumulator određujemo postavljenjem gornje granice. Ako je $n < 1000$, onda postoje četiri akumulatora, s_1, s_2, s_3, s_4, sa gornjim granicama $10, 100, 1000, 10{,}000$. Tehnika dodavanja novog elementa u sumu je sljedeća. Prvo se provjeri bi li dodavanjem tog elementa u sumu s_1 prekoračila dozvoljenu granicu, to jest $s_1 \geq 10$. Ako da, s_1 se doda s_2, i s_1 se postavlja na vrijednost elementa kojeg dodajemo. Nakon toga s_1 dodajemo s_2 ako s_2 neće prekoračiti granicu, inače ponavljamo isti postupak. Važno je da svaki novi element treba biti dodan u akumulator koji pokriva dio raspona u kojem leži.

Broj akumulatora i rasponi koje pokrivaju, mogu se mijenjati i to bez reprogramiranja. Oba faktora ovise o problemu i karakteristikama računala. U većini slučajeva, poznat je prosječan konačan rezultat pa lako odredimo gornju granicu, a donju granicu u najgorem slučaju možemo vidjeti po elementima.

Prikažimo te intervale. Akumulatori su označeni sa $s_i,\ i = 1, 2, 3, 4, \ldots$ i svaki od njih drži sume u pojedinom intervalu. Primjerice, intervali su sljedeći, i $c(s_i),\ i = 1, 2, 3, 4, \ldots$ označavaju sadržaj od $s_i,\ i = 1, 2, 3, 4, \ldots$:

$$\begin{gathered} 10.0000 \leq |c(s_1)| \leq 99.9999 \\ 100.000 \leq |c(s_2)| \leq 999.999 \\ 1000.00 \leq |c(s_3)| \leq 9999.99 \\ 10000.0 \leq |c(s_4)| \leq 99999.9 \\ \vdots \end{gathered}$$

Svi sumandi se nalaze u nekom od intervala tako da srednja suma tih sumanada upada u određeni interval. Budući da su srednje sume ograničene, zaključujemo da po Wolfovoj tehnici, srednja suma nikad neće biti puno veća od sumanada. Ipak, katastrofalna kraćenja se mogu dogoditi. Wolf nije komentirao metodu kojom bi sumirao akumulatore na kraju, iako je u primjeru koristio rastući redoslijed. Njegova tehnika je dobra za određene probleme, ali ne garantira da konačan rezultat ima malu relativnu grešku.

Zbog toga Malcom modificira Wolfov algoritam. Akumulacija sume brojeva s pomičnom točkom se izvršava na računalu u aritmetici sa t znamenaka i bazom β te eksponentima u rasponu od $-m$ do M. Algoritam služi za točno sumiranje n t-znamenkastih brojeva s pomičnom točkom, i dan je u koracima:

Neka su l i η pozitivni brojevi. Pretpostavimo da postoji $\eta + 1$ akumulator, čiji su sadržaji označeni sa $a_0, a_1, \ldots, a_\eta$.

1. Postavi svaki akumulator na nulu.

2. Svaki od n brojeva podijeli na q dijelova, formirajući $q \cdot n$ t-znamenkastih brojeva s pomičnom točkom:

$$a_{i1} + a_{i2} + \ldots + a_{iq} = x_i$$

i svaki a_{ij} ima svojstvo da su posljednjih l znamenaka jednake nuli. Broj znamenaka za koji mantisa od nekog a_{ij} može biti pomaknuta u desno prije nego dođe do gubitka neke značajne znamenke, mora biti veći ili jednak l koji je strogo veći od nule.

3. Svaki a_{ij} dodaj k-tom akumulatoru, jednom od η pomoćnih t-znamenkastih akumulatora, gdje je k određen sa:

$$vk \leq \text{eksponent}(a_{ij}) \leq vk + v - 1,$$
$$v = (M + m + 1)/(\eta + 1)$$

4. Sumiraj akumulatore po padajućem redoslijedu.

Ukupno je izvršeno $q \cdot n + \eta - 1$ sumacija sa t-znamenkastim brojevima s pomičnom točkom.

Malcom daje detaljnu analizu grešaka kako bi pokazao da njegova metoda postiže malu relativnu grešku. Nedostatak njegovog algoritma je taj što je jako ovisan o stroju. U zadnjem koraku, akumulatori su sumirani po metodi rekurzivne sumacije kako bi redoslijed apsolutnih vrijednosti sumanada bio padajući. Posebno u ovoj metodi, sumacija s padajućim redoslijedom sprječava teške gubitke značajnih znamenki i garantira malu relativnu grešku. U slučaju sumacije s rastućim redoslijedom, dogodila bi se katastrofalna kraćenja i čak bi suma u nekim slučajevima bila jednaka nuli. Detaljnije o prednostima padajućeg redoslijeda i analizi grešaka pogledajte u [6].

4.2 Destilacijski algoritam

Destilacijski algoritam koristi samo standardnu aritmetiku konačne preciznosti i ne oslanja se na bazu korištenu u aritmetičkom modelu, na arhitekturu strojeva ili na korištenje akumulatora. Algoritam je primjenjiv na sve skupove podataka, a posebno na one koji nisu dobro uvjetovani, to jest za koje standardne metode ne vrijede zbog akumulacije greške zaokruživanja i izlaganja kraćenju.

Destilacijske algoritme je prvi opisao Kahan. Za dane brojeve x_i, $i = 1, 2, \ldots, n$, algoritmi iterativno konstruiraju brojeve s pomičnom točkom, $x_1^{(k)}, \ldots, x_n^{(k)}$, tako da vrijedi $\sum_{i=1}^{n} x_i^{(k)} = \sum_{i=1}^{n} x_i$, koja završava kad $x_n^{(k)}$ aproksimira sumu $\sum_{i=1}^{n} x_i$ s relativnom greškom koja iznosi najviše $u = \frac{1}{2}\beta^{1-t}$. Kahan tvrdi da takvi algoritmi imaju prosječno vrijeme izvršavanja reda najmanje $n \log(n)$.

Prvo ćemo opisati **deflacijski algoritam**, koji je jedna vrsta destilacijskog algoritma. Više nije potrebno promatrati greške ni ogradu greške, jer se pokazalo da točna računanja proizvode nove podatke čija je suma ista kao suma originalnih podataka. Jedino je broj uvjetovanosti kod deflacijskog algoritma, definiran u (2.6), manji nego kod originalnih podataka.

Kad svi sumandi imaju jednak predznak, ovaj algoritam koristi metodu kompenzirane sumacije. Kad sumandi imaju suprotan predznak, algoritam izvrši deflaciju ta dva broja, koju ćemo opisati kasnije, iterativno sve dok suma pozitivnih brojeva, s_+, i suma negativnih brojeva, s_-, ne zadovoljavaju sljedeće:

$$|fl\,(fl\,(s_+ - s_-)\,/\,fl\,(s_+ + s_-))| = 1 \tag{4.1}$$

Tada je suma dobro uvjetovana.

Jedna deflacija je poseban slučaj kompenzacije, a to je zbrajanje brojeva suprotnog predznaka:

$$\hat{s} = fl(a + b), \quad \hat{e} = fl(fl(a - \hat{s}) + b) \tag{4.2}$$

gdje $|a| \geq |b|$ i $ab \leq 0$. Tada radimo supstituciju: $a = \hat{s}$ i $b = \hat{e}$.

Deflacijski algoritam možemo prikazati po koracima:

1. Sortiraj brojeve: $|x_1| \geq |x_2| \geq |x_3| \geq \cdots \geq |x_m|$
2. Označi prvi par susjednih brojeva sa suprotnim predznakom sa a i b.
3. Izbaci a i b iz niza.
4. Izvrši deflaciju a i b kako bi dobili dva manja broja.
5. Stavi ta dva broja u niz.
6. Sortiraj brojeve kao i u koraku 1. Nova dva broja se mogu staviti na odgovarajuće mjesto u sortiranom nizu tako da i novi niz bude sortiran.
7. Ponavljaj 2. sve dok ne ostanu brojevi s istim predznakom.
8. Iskoristi kompenziranu sumaciju za sumiranje preostalih brojeva.

U [7] je pokazano da je deflacija (4.2) uvijek točna, bez obzira na bazu β, i iznosi $a+b = \hat{s}+\hat{e}$. To znači da su $\hat{s} = a$ i $\hat{e} = b$, kada $|a| \gg |b|$, što rezultira beskonačnom petljom. Kako bi to izbjegao, Anderson koristi redukcijski algoritam, koji reducira a na brojeve w i v, i vrijedi $fl(w + v) = w + v = a$ i v je puno manji od w. Nakon par reduciranja, v je dovoljno mala vrijednost da napravi deflaciju tako da $\hat{s} \neq v$ i $\hat{e} \neq b$. Kad je (4.1) zadovoljena, možemo iskoristiti Kahanovu kompenziranu sumaciju kako bi zbrojili ostatke. Kako je svaki korak točan, to jest imamo skup podataka sa brojem uvjetovanosti uvj $= 1$, i nova

suma je jednaka originalnoj, greška se pojavljuje tek u zadnjem koraku. U koraku 8., koristi se kompenzirana sumacija za sumiranje novog skupa podataka, koja proizvodi relativnu ogradu greške, $2u$. Zbog toga i zbog poglavlja 3.2 je ograda greške za ovu metodu jako mala, sve dok vrijedi $nu \leq 1$. Relativna ograda greške je neovisna o broju uvjetovanosti originalnih podataka, a konačna suma je i stabilna unaprijed i unazad.

Broj operacija deflacijskog algoritma je $O(n^2) + O(n \log n)$, gdje prvi dio pripada broju deflacija, a drugi broju operacija za sortiranje. Umjesto početnog sortiranja, koje troši puno vremena, pogotovo ako ima puno podataka, trebamo novi modificirani algoritam.

Anderson uvodi novi destilacijski algoritam za sumaciju brojeva s pomičnom točkom, zvan **modificirana deflacija**. Novi deflacijski algoritam možemo prikazati po koracima:

1. Podaci su raspoređeni u dva skupa: pozitivni brojevi, P, i negativni, N.

2. U oba skupa izbaci zadnji broj i izvrši deflaciju ta dva broja, kako bismo proizveli sumu ili razliku, s, i grešku, e. Vrijednost s je vraćena na kraj odgovarajućeg skupa, a e je odbačena u inicijalni prazan skup ϵ, takozvani skup malih članova.

3. Ponavljaj 2. sve dok se jedan od skupova ne isprazni.

4. Podaci iz skupa ϵ su raspoređeni u odgovarajuće skupove, ovisno o tome jesu li pozitivni ili negativni, i podaci u svakom skupu posebno su sumirani pomoću kompenzirane sumacije. Dobivamo s_+ i s_-. Broj uvjetovanosti sumacije preostalih članova je $\text{uvj} = |(s_+ - s_-) / (s_+ + s_-)|$. Ako uvj nije 1, ponovi cijeli proces od 2. koraka na sljedeći način. Nakon što su podaci iz skupa ϵ raspoređeni u odgovarajuće skupove, P i N, prijeđi u 2. korak i ponovi cijeli postupak do kraja.

5. Konačna suma se izračuna pomoću kompenzirane sumacije.

Iz 5. vidimo da su brojevi iz skupa ϵ puno manji od najvećeg broja iz skupova P i N. Znači da su ti mali ne toliko značajni brojevi brzo odbačeni, dok relativno veliki preostali brojevi ostaju u skupovima kako bi se izvršila deflacija među njima.

Novi deflacijski algoritam je reda veličine $O(n)$, iako je Kahan rekao da su destilacijski algoritmi reda veličine najmanje $O(n \log n)$. Razlog tome je što ovaj algoritam završava kada $fl(\text{uvj}) = 1$, što je puno ranije nego standardni destilacijski algoritam. Ostaci se tada zbroje pomoću kompenzirane sumacije.

Sve u svemu, novi deflacijski algoritam za sumaciju brojeva s pomičnom točkom je točan i dovoljno brz da se može koristiti u praksi. Algoritam koristi deflaciju na način da iterativno manipulira sumandima bez ukidanja značajnih znamenki, kako bi eliminirao podatke koji nisu dobro uvjetovani. Nakon eliminacije tih podataka, možemo iskoristiti neku od standardnih metoda sumacija kako bi dobili konačnu sumu. Destilacijska metoda je stabilna unazad, i za razliku od ostalih sumacijskih metoda, proizvodi sumu koja ima malu relativnu grešku, bez obzira na originalne podatke.

Autori još nekih poznatijih algoritama su Pichat, Bohlender, Leuprecht i Oberaigner, kao i analize grešaka destilacijskih algoritma, detaljnije su opisani u [1], [7] i [11].

4.3 Izbor metode

Kao što smo dosad vidjeli, postoji mnogo sumacijskih metoda za brojeve s pomičnom točkom. Za svaku metodu, greška može jako varirati s podacima, i to ovisno o ogradi greške. Što je ograda greške manja, greška među podacima može manje varirati. Zbog toga ne možemo izabrati najbolju metodu, u smislu točnosti. Iako, numerički eksperimenti su pokazali da za bilo koje dvije metode sumacije, moguće je naći podatke za koje je jedna od tih dviju metoda točnija. Unatoč tome, neki od zaključaka su sljedeći:

- Za većinu metoda, greške su, u najgorem slučaju, proporcionalne sa n. Metode za koje to ne vrijedi su sumacija po parovima sa konstantnom greške, $\log_2(n)$, i kompenzirana sumacija sa konstantnom greškom reda 1. Slijedi da za velike n, te dvije metode su bolje od ostalih.

- Ako svi x_i, $i = 1, 2, \ldots, n$ imaju isti predznak, tada sve metode imaju relativnu grešku najviše nu, a kompenzirana sumacija garantira i malu relativnu grešku, i to sve dok $nu \leq 1$. Ako su članovi u rekurzivnoj sumaciji istog predznaka, onda je bolji rastući redoslijed od padajućeg. Razlog tome je što rastući poredak ima najmanju ogradu greške, (2.5). Metoda sumacije umetanjem minimizira tu ogradu u svim slučajevima algoritma 1. Slijedi da sumacija umetanjem ima najmanju ogradu od svih metoda osim kompenzirane sumacije.

- Za sume sa katastrofalnim kraćenjem, to jest gdje vrijedi $\sum_{i=1}^{n} |x_i| \gg \left|\sum_{i=1}^{n} x_i\right|$, preferiramo rekurzivnu sumaciju sa padajućim redoslijedom. Međutim, to ne garantira najveću točnost.

- Promotrimo rekurzivnu sumaciju u višoj preciznosti. Ako $s_n = \sum_{i=1}^{n} x_i$ izračunamo preko rekurzivne sumacije s dvostrukom preciznošću i onda zaokružimo u jednostrukoj preciznosti, ograda za grešku je $|s_n - \hat{s}_n| \leq u\,|\hat{s}_n| + nu^2 \sum_{i=1}^{n} |x_i|$, gdje je u jedinična greška zaokruživanja za jednostruku preciznost, a u^2 za dvostruku. Relativna greška reda u je garantirana ako vrijedi $nu \sum_{i=1}^{n} |x_i| \leq |s_n|$. Priest je u [7] pokazao da ako su x_i u padajućem poretku prije nego su sumirani u dvostrukoj preciznosti, onda vrijedi $|s_n - \hat{s}_n| \leq 2u\,|s_n|$ samo ako $n \leq \beta^{t-3}$, za aritmetiku sa t znamenki i bazom β, zadovoljava određene pretpostavke. Stoga se isplati uvesti padajući poredak ako postoji puno kraćenja u sumi. Alternativa za preciznije računanje je dvostruko kompenzirana sumacija, koja garantira malu relativnu grešku u izračunatoj sumi.

Uzevši u obzir trošak i način na koji su podaci generirani, možemo istaknuti pojedine metode. Rekurzivna sumacija u prirodnom poretku, sumacija po parovima i kompenzirana sumacija mogu biti implementirane u $O(n)$ operacija, za općenite x_i, $i = 1, 2, \ldots, n$. Druge metode imaju puno veći trošak, jer zahtjevaju dodatna traženja i sortiranja. Štoviše, u

primjeni kao na primjer numeričko rješenje obične diferencijalne jednadžbe, gdje x_k nije poznat sve dok suma prethodnih članova nije formirana, sortiranje i traženje mogu biti nemogući. Također, rekurzivna sumacija u većoj preciznosti može imati manji trošak od onih u jednostrukoj preciznosti.

Poglavlje 5

Primjeri u MATLAB-u

5.1 Eulerova metoda

Uzmimo primjer koji će nam pokazati dobre strane kompenzirane sumacije. Iskoristimo Eulerovu metodu za obične diferencijalne jednadžbe kod rješavanja inicijalnog problema $y\prime = f(x, y)$, za dani $y(a)$. Eulerovu metodu možemo zapisati rekurzijom:

$$y_{k+1} = y_k + hf_k,\ y_0 = y(a).$$

Dobivene vrijednosti y_k su aproksimacije rješenja diferencijalne jednadžbe u točkama x_k.

Dana je sljedeća jednadžba:

$$y\prime = -y,\ y(0) = 1,$$

na intervalu $[0, 1]$, koristeći n koraka Eulerove metode, za n između 10 i 10^8.

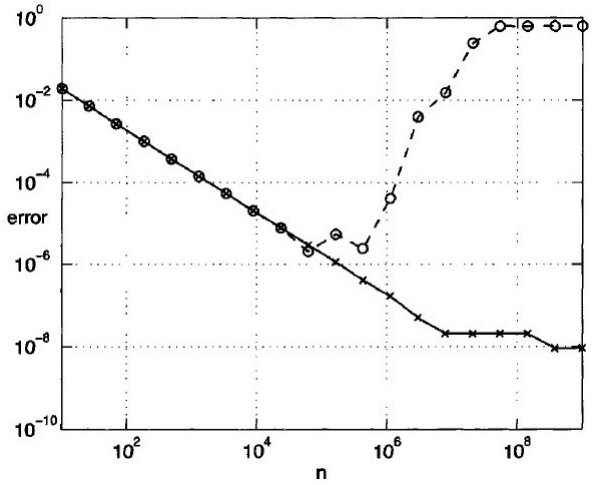
Zbog ekvidistantne mreže, u kompenziranoj sumaciji (vidi algoritam 2) zamijenimo $x = x + h$ sa sljedećom jednadžbom:

$$x_k = x_0 + k \cdot h.$$

A $y = y + h \cdot f(x, y)$ zamijenimo sa sljedećom jednadžbom, gdje $cy = 0$ na početku:

$$\begin{aligned} dy &= h \cdot f(x_k, y) + cy \\ y_{novi} &= y + dy \\ cy &= (y - y_{novi}) + dy \\ y &= y_{novi}. \end{aligned}$$

Slika 5.1, preuzeta iz [4], prikazuje greške $e_n = |y(1) - \hat{y_n}|$, gdje je $\hat{y_n}$ izračunata aproksimacija od $y(1)$. Os x predstavlja vrijednosti n, a os y greške.



Slika 5.1: Greške $|y(1) - \hat{y}_n|$ za Eulerovu metodu, gdje "x" označava bez, a "$\circ$" sa kompenziranom sumacijom

Budući da Eulerova metoda ima globalnu grešku reda h, krivulja greške bi trebala biti približno jednaka ravnoj liniji. Međutim, nakon $n = 20{,}000$, greške e_n naglo rastu. Uzrok tome je utjecaj greške zaokruživanja. Kako su greške e_n iz kompenzirane sumacije mnogo manje pod utjecajem grešaka zaokruživanja, ne rastu s porastom od n. Na primjer, za 10^8, e_n je otprilike 10 puta veći nego kod egzaktne aritmetike.

Trošak uvođenja kompenzirane sumacije u rješavanje običnih diferencijalnih jednadžbi je skoro zanemariv ako je funkcija f skupa za računanje. Dok s druge strane, benefiti koje donosi su vidljivi samo u slučaju ogromnog broja integracijskih koraka. Zbog toga su takvi koraci poduzeti u uređajima u astronomiji, gdje zaokruživanje može utjecati na sposobnost praćenja planetarnih orbita. Istraživači u astonomiji koriste kompenziranu sumaciju i ostale tehnike kako bi se borili sa zaokruživanjem.

5.2 Numerički eksperimenti

U ovom poglavlju ćemo opisati numeričke eksperimente koji će nam dati bolji uvid u točnost sumacijskih metoda. Eksperimenti se provode u MATLAB-u, koji koristi IEEE standard aritmetike dvostruke preciznosti s jediničnom greškom zaokuživanja $u \approx 1.1 \times 10^{-16}$.

Prvo ćemo pokazati kako se koja metoda ponaša na četiri različite klase podataka $\{x_i,\ i = 1, 2, \ldots, n\}$. U tim testovima ćemo simulirati aritmetiku jednostruke preciznosti s jediničnom greškom zaokruživanja $u_1 = 2^{-23} \approx 1.2 \times 10^{-7}$ tako da zaokružujemo rezultat

svake aritmetičke operacije na 23 značajna bita. Sumiramo brojeve jednostruke preciznosti x_i rekurzivnom sumacijom u dvostruku preciznost kako bi aproksimirali s_n. U svim testovima vrijedi $nu \sum_{i=1}^{n} |x_i| < u_1 |s_n|$ pa ograda greške, (2.3), garantira da je ta aproksimacija točna u jednostrukoj preciznosti.

Promatramo rezultate za originalnu rekurzivnu sumaciju, rastuću, padajuću i Psum, zatim za metodu umetanjem, metodu $+/-$, gdje su s_+ i s_- dobiveni rekurzivnom sumacijom sa rastućim redoslijedom (vidi [3]), te za sumaciju po parovima sa rastućim redoslijedom i kompenziranom sumacijom. U prvom redu tablice bilježimo relativnu grešku $|\hat{s_n} - s_n|/|s_n|$, zajedno s informacijom koja oblikuje ograde. U drugom retku imamo $t = \sum_{k=1}^{n-1} \left|\hat{t}_k\right|$, iz (2.2), za sve metode osim kompenzirane sumacije. U trećem retku je omjer $r = |\hat{s_n} - s_n|/ (u_1 \sum_{i=1}^{n} |x_i|)$, koji je po analizi grešaka, ograđen sa n za rekurzivnu metodu, metodu umetanjem i metodu $+/-$, a sa $\log_2(n)$ za sumaciju po parovima i s 2 za kompenziranu sumaciju. Vrijednosti t i r otkrivaju koliko su najjača i najslabija ograda greške blizu. Primjeri su uzeti iz [3].

- U prvom primjeru, uzmimo x_i koji predstavlja i-ti član Taylorovog reda od razvoja funkcije e^{-x} oko ishodišta i $x = 2\pi$. Ovaj red je klasičan primjer katastrofalnog kraćenja. Rezultati za $n = 64$ su dani u tablici 5.1.

	Original	Rastući	Padajući	Psum	Parovi	Umetanje	$+/-$	Kompenz.
$G_r(s_n)$	$5.11e-4$	$2.27e-3$	$1.85e-7$	$2.27e-3$	$1.41e-4$	$2.27e-3$	$1.86e-2$	$5.11e-4$
t	$2.68e2$	$2.97e2$	$2.97e2$	$2.85e2$	$2.68e1$	$2.97e2$	$1.34e3$	
r	$1.49e-2$	$6.64e-2$	$5.40e-6$	$6.64e-2$	$4.13e-3$	$6.64e-2$	$5.44e-1$	$1.49e-2$

Tablica 5.1: $x_i,\ i = 1, 2, \ldots, n$, za $n = 64$, iz razvoja Taylorovog reda od $e^{-2\pi}$

  U ovom primjeru najbolju točnost, odnosno najmanju relativnu grešku, daje rekurzivna sumacija s padajućim redoslijedom. Zbog $\left|\sum_{i=1}^{n} x_i\right| / \sum_{i=1}^{n} |x_i| = 3.48e-6$, postoje neka kraćenja u sumi. Također, padajući redoslijed omogućava najmanjim članovima po apsolutnoj vrijednosti da sudjeluju čitavi u izračunatoj sumi dok su u drugim metodama najčešće zanemareni među većim. Ograda greške nije najbolja za padajući redoslijed, jer je t skoro jednak za sve redoslijede rekurzivne sumacije. Kompenzirana sumacija nije nešto bolja od originalne rekurzivne metode, a $+/-$ metoda daje jednu manje točno značajnu znamenku, za razliku od svih ostalih metoda, kao što je predviđeno t vrijednostima.

- U drugom primjeru, x_i su proizvoljni brojevi iz standarne normalne distribucije, sa očekivanjem 0 i varijancom 1. Rezultati za $n = 2048$ su prikazani u tablici 5.2, a za $n = 4096$, u tablici 5.3.

  Zbog $\left|\sum_{i=1}^{n} x_i\right| / \sum_{i=1}^{n} |x_i| = 8.08e-3$, za $n = 2048$, i $\left|\sum_{i=1}^{n} x_i\right| / \sum_{i=1}^{n} |x_i| = 3.48e-3$, za $n = 4096$, postoje neka kraćenja u obe sume, ali ne toliko puno koliko u prvom

	Original	Rastući	Padajući	Psum	Parovi	Umetanje	+/−	Kompenz.
$G_r(s_n)$	$7.47e-6$	$3.32e-6$	$7.17e-6$	$6.82e-8$	$6.60e-7$	$5.12e-7$	$1.20e-4$	$2.28e-7$
t	$3.06e4$	$1.53e4$	$2.65e4$	$8.26e2$	$2.87e3$	$2.32e3$	$4.80e5$	
r	$5.06e-1$	$2.25e-1$	$4.86e-1$	$4.62e-3$	$4.47e-2$	$3.47e-2$	$8.15e0$	$1.54e-2$

Tablica 5.2: $x_i,\ i = 1, 2, \ldots, n$, za $n = 2048$, iz $N(0, 1)$

	Original	Rastući	Padajući	Psum	Parovi	Umetanje	+/−	Kompenz.
$G_r(s_n)$	$8.06e-6$	$1.04e-5$	$1.84e-6$	$2.66e-8$	$1.38e-7$	$6.87e-7$	$3.68e-4$	$1.92e-7$
t	$6.69e4$	$4.74e4$	$4.28e4$	$1.68e3$	$5.70e3$	$5.38e3$	$2.02e6$	
r	$2.35e-1$	$3.04e-1$	$5.38e-2$	$7.76e-4$	$4.04e-3$	$2.00e-2$	$1.07e1$	$5.59e-3$

Tablica 5.3: $x_i,\ i = 1, 2, \ldots, n$, za $n = 4096$, iz $N(0, 1)$

primjeru. U ovom primjeru je najbolja rekurzivna metoda sa Psum redoslijedom, a +/− metoda najgora, što se vidi po t vrijednostima.

U sljedeća dva primjera, x_i su pozitivni pa je za svaku metodu garantirana relativna greška ne veća od $f(n)u,\ f(n) \leq n$, koja ovisi o metodi. Također, za pozitivne x_i, rekurzivna metoda sa rastućim redoslijedom, sa Psum redoslijedom i +/− metoda su jednake.

- U trećem primjeru, $x_i = 1/i^2$. Promotrimo kako greške variraju s obzirom na n za rekurzivnu sumaciju s padajućim i rastućim redoslijedom. Rezultati za $n = 500, 1000, \ldots, 5000$ su dani u tablici 5.4.

n	500	1000	2000	3000	4000	5000
Rastući	$1.04e-7$	$1.01e-7$	$1.74e-8$	$5.22e-8$	$1.36e-7$	$3.90e-8$
Padajući	$3.31e-7$	$6.24e-7$	$5.64e-6$	$2.30e-5$	$2.77e-5$	$5.81e-5$

Tablica 5.4: $x_i = 1/i^2,\ i = 1, 2, \ldots, n$

Padajući redoslijed rekurzivne sumacije daje puno manju točnost nego rastući, kad je n velik. To je očekivano zbog analize grešaka u poglavlju 2.2.

- U četvrtom primjeru, x_i su ekvidistantni na intervalu $[1, 2]$. Za $n \leq 4096$, ne vidi se velika razlika između rastućeg i padajućeg redoslijeda rekurzivne metode. To je očekivano, jer x_i malo variraju u danom intervalu. Za poprilično velike n, u tablici 5.5, sumacija po parovima nadmašuje rekurzivnu sumaciju, a sumacija umetanjem je ekvivalentna sumaciji po parovima. Greške za kompenziranu sumaciju su nule, za svaki n koji je isproban.

n		Rastući	Padajući	Parovi	Kompenz.
2048	$G_r(s_n)$	$2.86e-6$	$3.86e-5$	$1.59e-7$	0.00
	t	$2.80e6$	$3.50e6$	$3.38e4$	
	r	$2.40e1$	$3.24e2$	$1.33e0$	0.00
4096	$G_r(s_n)$	$3.35e-5$	$2.18e-5$	$1.59e-7$	0.00
	t	$1.12e7$	$1.40e7$	$7.37e4$	
	r	$2.81e2$	$1.83e2$	$1.33e0$	0.00

Tablica 5.5: $x_i,\ i = 1, 2, \ldots, n$ ekvidistantni na intervalu $[1, 2]$

5.3 MDS

U ovom poglavlju koristimo implementaciju metode MDS u MATLAB-u kako bismo usporedili sumacijske metode koje smo spomenuli u prethodnim poglavljima. **MDS** ili **Multidirektno pretraživanje** je jedna od metoda direktnog pretraživanja koji služe za bezuvjetnu optimizaciju. Osnovna ideja ovog algoritma je izvođenje konkurentnih pretraživanja u više smjerova. Takva pretraživanja nisu međusobno ovisna pa se tražena informacija može izračunati u paralelnom računanju. Detaljnije o ovoj metodi možete naći u [9].

Klasičan problem bezuvjetne optimizacije je locirati točku maksimuma (ili minimuma) zadane funkcije $f : \mathbb{R}^n \to \mathbb{R}$. Metoda optimizacije smije koristiti samo funkcijske vrijednosti $f(x)$ u pojedinim točkama $x \in \mathbb{R}^n$, a ne smije koristiti derivacije od f, jer one ne moraju postojati u nekim točkama. Dakle, metoda zapravo pronalazi točku s najvećom (najmanjom) vrijednošću funkcije.

Uzmimo primjere iz [3]. Primijenimo metodu na f, koja je relativna greška sume izračunate u aritmetici jednostruke preciznosti, pomoću rekurzivne sumacije s rastućim redoslijedom. Neka je $n = 3$ i $x_0 = \left[\frac{1}{3}, \frac{2}{3}, 1\right]$. Nakon 280 evaluacija funkcije, korištenjem ove metode, pronađena je točka maksimuma $x = [4.975987\ -2.495094\ -2.480894]$, $f(x) = 1.0$. Pripadne sume su:

$$\hat{s_3} = -9.5367 \times 10^{-7},\ s_3 = -4.7684 \times 10^{-7}.$$

Vektor x ima elemente zaokružene na 7 značajnih znamenki, a sume na 5 značajnih znamenki.

Neka je sad funkcija f definirana kao relativna greška za kompenziranu sumaciju. Za $x_0 = \left[-\frac{1}{3}, 0, \frac{2}{3}\right]$, nakon 166 evaluacija funkcije, podaci su sljedeći:

$$x = [-0.8308306\ -0.7626623\ 1.593493],\ f(x) = 1.0,$$
$$\hat{s_3} = 2.3842 \times 10^{-7},\ s_3 = 1.1921 \times 10^{-7}.$$

Ako su elementi od x razmješteni pomoću rastućeg redoslijeda, $f(x)$ je tada 1.0, i $f(x) = 0$ u slučaju padajućeg redoslijeda.

Ovo su tipični primjeri za metodu MDS, koja pronalazi podatke za koje proizvoljna metoda sumacije ne daje točno značajne znamenke u sumi.

Također, maksimum može poslužiti i za uspoređivanje dviju različitih metoda tako da f definiramo kao omjer grešaka tih dviju metoda. Možemo usporediti rekurzivnu sumaciju s rastućim redoslijedom, sa kompenziranom sumacijom. Približavajući se podacima za koje greška koja oblikuje nazivnik od f, je nula, omjeri su jako veliki. Na isti način se mogu usporediti i ostale metode.

Bibliografija

[1] I. J. Anderson, *A distillation algorithm for floating-point summation*, SIAM J. Sci. Comput **20** (1999), 1797–1806.

[2] Z. Drmač i i ostali, *Numerička analiza (skripta)*, (2003), `https://web.math.pmf.unizg.hr/~singer/num_mat/num_anal.pdf`.

[3] N. J. Higham, *The Accuracy of Floating Point Summation*, Department of Mathematics, University of Manchester (1993), 783–799.

[4] Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second edition., SIAM, Philadelphia, 2002.

[5] W. Kahan, *Implementation of Algorithms*, NTIS, 1973.

[6] Michael M. Malcom, *On Accurate Floating-Point Summation*, Comm. ACM **14** (1971), 731–736.

[7] Douglas M. Priest, *On Properties of Floating Point Aritmetics: Numerical Stability and the Cost of Accurate Computations*, Department of Mathematics, University of California (1992).

[8] D. R. Ross, *Reducing Truncation Errors Using Cascading Accumulators*, Comm. ACM **8** (1965), 32–33.

[9] V.J. Torczon, *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*, Department of Mathematical Sciences, Rice University, Houston (1989).

[10] Jack M. Wolfe, *Reducing Truncation Errors by Programming*, Comm. ACM **7** (1964), 355–356.

[11] Yong Kang Zhu, Jun Hai Yong i Guo Qin Zheng, *New Distillation Algorithm for Floating-Point Summation*, SIAM J. Sci. Comput **26** (2006), 2066–2078.

Sažetak

Glavna tema ovog rada je točnost algoritama koje koristimo pri sumiranju brojeva s pomičnom točkom, u aritmetici konačne preciznosti na računalu. Na početku rada smo opisali konačnu aritmetiku računala i neke njene osnovne pojmove na kojima se temelji ostatak rada. Zatim smo definirali tri osnovne metode sumacije. Koristeći analizu grešaka i statističke procjene točnosti, usporedili smo metode. Nakon toga smo uveli još jednu točniju vrstu sumacije, takozvano kompenzirano sumiranje. Uz pomoć Kahanovog i Priestovog algoritma, detaljnije smo analizirali i tu metodu. Između svih tih metoda, kao i nekih dodatnih metoda, nismo mogli izdvojiti onu najbolju, u smislu točnosti za baš sve moguće ulazne podatke. Ipak, numerički eksperimenti u MATLAB-u su pokazali da za bilo koje dvije metode sumacije, moguće je naći podatke za koje je jedna od tih dviju metoda točnija.

Summary

The main subject of this thesis is accuracy of summation algorithms in finite precision arithmetic on a computer. At the beginning, we have described the finite arithmetic of a computer and some of the basic concepts that the rest of the thesis is based on. Then we define three main summation methods. By using the error analysis and statistical estimates of accuracy, we compare the methods. After that, we introduced a more accurate type of summation, so-called compensated summation. With the help of Kahan's and Priest's algorithms, we described in detail that method. Among all these methods and some additional methods, we could not single out the best one, in terms of accuracy for every possible choice of input data. However, numerical experiments in MATLAB show that, given any two summation methods, data can be found for which either method is more accurate than the other.

Životopis

Ivona Varnica rođena je u Zagrebu dana 07.09.1995. Svoje djetinjstvo je provela u Šibeniku, gdje je pohađala Osnovne škole Fausta Vrančića (prva dva razreda) i Tina Ujevića (ostatak). Nakon toga je upisala Gimnaziju Antuna Vrančića, opći smjer, koju završava s odličnim uspjehom. Iako matematika nije presudila za upis u srednju školu, za fakultet je. Godine 2014. upisuje Prirodoslovno-matematički fakultet u Zagrebu, preddiplomski studij Matematike, inženjerski smjer. Nakon stjecanja titule Sveučilišne prvostupnice matematike, godine 2017. upisuje diplomski studij Financijska i poslovna matematika na istom fakultetu, kojeg završava ovim radom.