Predavanje 1: Uvod

4 Tipovi grešaka

Da bismo mogli procijeniti da li neki algoritam implementiran na računalu izračunava rješenje promatranog problema s dovoljnom točnošću najprije se moramo upoznati s vrstama grešaka koje se pri tom javljaju.

4.1 Greške zbog polaznih aproksimacija

Ovaj tip grešaka se često javlja kod rješavanja praktičnih problema. Ovako nastale greške se mogu podijeliti u tri klase: greške *modela*, greške *metode* i greške u *polaznim podacima*.

- **Greške modela**

  Ove greške nastaju zamjenom složenih sustava jednostavnijima koje se mogu opisati matematičkim modelima odnosno zapisima (primjer bi bio zanemarivanje utjecaja otpora zraka na gibanje u zemaljskim uvjetima). Često se već postojeći dobri modeli zamjenjuju jednostavnijima da bi se mogle primijeniti numeričke metode (npr. linearizirаnje nelinearnih sustava parcijalnih diferencijalnih jednadžbi). Greške modela se mogu javiti prilikom rješavanja problema koji su granični slučajevi (npr. aproksimiranje vrijednosti $\sin x$ s x i kada vrijednost x nije bliska nuli). Ove greške su neuklonjive, a na karisniku je ocijeniti da li primjena daje dovoljno dobre rezultate.

- **Greške metode**

  Te greške nastaju kada se beskonačni procesi zamjenjuju konačnima. Također nastaju kod računanja veličina koje su definirane pomoću limesa (derivacije, integrali, granične vrijednosti nizova,...). Velik broj numeričkih metoda za aproksimiranje funkcija i rješavanje jednadžbi je upravo ovog oblika. Greške koje nastaju zbog zamjene beskonačnog nečim konačnim obično dijelimo u dvije kategorije: greške *diskretizacije* i greške *odbacivanja*.

(i) **Greške diskretizacije**

Ove greške nastaju zamjenom continuuma konačnim diskretnim skupom točaka ili kada se beskonačno mala veličina zamijeni nekim konkretnim malim brojem. One nastaju i kada se derivacija zamijeni podijeljenom razlikom, integral kvadraturnom formulom ili diferencijalna jednadžba diferencijskom jednadžbom. Možda je najjednostavniji primjer aproksimacija funkcije definirane na intervalu $[a, b]$ funkcijom definiranom na diskretnom skupu $\{x_1, \ldots, x_n\} \subset [a, b]$.

(ii) **Greške odbacivanja**

Greške odbacivanja nastaju kada se beskonačni niz, red, umnožak, suma i sl. zamijene konačnima (tj. kada odbacimo ostatak).

- **Greške u polaznim podacima**
  One imaju izvor u mjerenjima fizičkih veličina, smještanju podataka u računalo i prethodnim računanjima. No greške mjerenja i smještanja je puno jednostavnije ocijeniti od grešaka koje nastaju usljed brojnih zaokruživanja tijekom računanja.

Grubo rečeno: diskretizacija je vezana za continuum $(\mathbb{R}, \mathbb{C})$, a odbacivanje za diskretnu (prebrojivu) beskonačnost $(\mathbb{N}, \mathbb{Z})$. Objekti koji nedostaju zbog tih zamjena tvore tip grešaka koji se zovu greške metode.

4.2 Greške zaokruživanja

Greške zaokruživanja nastaju zbog toga što računala koriste konačnu aritmetiku, ili točnije binarnu **aritmetiku s pomičnom točkom**, kod koje je unaprijed rezerviran određeni broj binarnih mjesta za eksponent i mantisu. Usljed toga se svaka računska operacija u kojoj sudjeluju dva broja izračunava s nekom malom greškom (koja može biti i nula). Tu grešku, ako nije jednaka nula, može se precizno ocijeniti, a nazivamo je **greškom zaokruživanja.** Očito, što je neki algoritam složeniji to ima više računskih operacija, a kod gotovo svake će se javiti greška zaokruživanja. Stoga se postavlja pitanje s kojom ćemo greškom dobiti traženo rješenje?

Ovim problemom se bavi **teorija grešaka zaokruživanja**, a osjetljivošću rješenja problema kojeg rješavamo na pomake u polaznim podacima bavi se **teorija perturbacije**. Njihovom usklađenom uporabom često je moguće procijeniti točnost promatranog algoritma, a ako točnost izračunatih podataka ne odstupa znatno od točnosti ulaznih, onda govorimo o **stabilnom algoritmu**.

Malo pogledat:

Ne samo iracionalni, već i mnogi racionalni brojevi umjerene veličine nemaju točnu reprezentaciju u računalu. Ako je x neki realni broj, onda njegovu *računalnu reprezentaciju* označavamo s

$$fl(x).$$

Proučavanje pokazuje da se kod svake računske operacije u računalu javlja greška. To se zapisuje u obliku

$$fl(x \circ y) = (x \circ y)(1+\varepsilon), \quad |\varepsilon| \leq u, \quad \circ \in \{+, -, *, /\},$$

pričemu je u tzv. **preciznost računanja** ili **strojni** u. Greška ovisi o operandima x, y i operaciji $\circ$, dok u ovisi o računalu (IEEE standardu). Općenito, ako računalo koristi p binarnih znamenaka u mantisi, onda vrijedi $u = 2^{-p+1}$ ili $u = 2^{-p}$ ovisno o načinu zaokruživanja u računalu.
Glavna zadaća osobe koja se bavi numeričkom matematikom jest određivanje što bolje aproksimacije rješenja u što kraćem vremenu.

4.3 Apsolutna i relativna greška

Neka je $\widehat{x}$ neka aproksimacija realnog broja x. Najkorisnije mjere za točnost broja $\widehat{x}$ kao aproksimacije broja x su:

- **apsolutna greška**

$$G_{\text{aps}}(x) = |x - \widehat{x}|$$

- **relativna greška**

$$G_{\text{rel}}(x) = \frac{|x - \widehat{x}|}{|x|}$$

koja nije definirana za $x = 0$.

Ako je x poznat ili mu se zna red veličine, onda je apsolutna greška dobra mjera udaljenosti aproksimacije od točne vrijednosti. No u praksi x često varira od vrlo velikih do vrlo malih vrijednosti, pa je primjerenija mjera relativna greška. Ona ima dodatno lijepo svojstvo da je naovisna o skaliranju,

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

Relativna greška povezana je s brojem točnih **značajnih znamenaka** neke aproksimacije. Značajne znamenke su prva netrivijalna znamenka i one koje slijede iza nje u zapisu. Npr. u broju 6.9990 imamo pet značajnih znamenaka, a u broju 0.0832 samo tri. Što znači broj točnih značajnih znamenaka vidjet ćemo kroz primjer:

$$x = 1.00000,\ \widehat{x} = 1.00499,\ \ G_{\text{rel}}(x) = 4.99 \cdot 10^{-3},$$
$$x = 9.00000,\ \widehat{x} = 8.99899,\ \ G_{\text{rel}}(x) = 1.12 \cdot 10^{-4}.$$

Evo jedne moguće definicije.

$\widehat{x}$ kao aproksimacija od x ima p točnih značajnih znamenaka ako se $\widehat{x}$ i x zaokružuju na isti broj od p značajnih znamenaka. Zaokružiti broj na p značajnih znamenaka znači zamijeniti ga s najbližim brojem koji ima p značajnih znamenaka. No prema ovoj definiciji brojevi $x = 0.9949$ i $\widehat{x} = 0.9951$ se ne slažu u dvije značajne znamenke, a slažu se u jednoj i u tri. Prema tome, definicija nije dobra.

Evo druge definicije.

$\widehat{x}$ kao aproksimacija od x ima p točnih značajnih znamenaka ako je $|x - \widehat{x}|$ manje od jedne polovine jedinice u p-toj značajnoj znamenci od x. Ova definicija implicira da se brojevi $x = 0.123$ i $\widehat{x} = 0.127$ slažu u dvije značajne znamenke, iako će mnogi misliti da se slažu u tri.

Kada se radi o vektorima greške se definiraju kao

$$G_{\text{aps}}(\boldsymbol{x}) = \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|$$

i

$$G_{\text{rel}}(\boldsymbol{x}) = \frac{\|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|},$$

a relacija

$$\frac{\|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leq \frac{1}{2}10^{-p}$$

implicira da komponente x_i za koje vrijedi $|x_i| \approx \|\boldsymbol{x}\|$ imaju približno p točnih značajnih znamenaka.
Ako želimo sve komponente vektora staviti u prvi plan onda koristimo relativne greške po komponentama, a veličinu

$$\max_i \frac{|x_i - \widehat{x}_i|}{|x_i|}$$

nazivamo **maksimalna relativna greška po komponentama**.

Treba razlikovati pojam *preciznosti* od pojma *točnosti*.

- **Točnost** se odnosi na apsolutnu i relativnu grešku kojom se aproksimira tražena veličina.
- **Preciznost** je točnost kojom se izvršavaju osnovne računske operacije, a u aritmetici pomične točke mjerimo je pomoću u. Određena je brojem bitova u reprezentaciji mantise, pa se ista riječ koristi i za taj broj bitova.

Ipak, važno je znati da preciznost *ne limitira* točnost. Naime, uvijek se (uz povečanje potrošnje računalnog vremena) uz neku danu preciznost može simulirati i veća preciznost računanja.

Predavanje 2: Aritmetika s pomičnom točkom

(Ne triba sve znat odavde samo podebljano, formule malo proučit)

1 Aritmetika s pomičnom točkom

Zapis broja -27.77 u tzv. **znanstvenoj notaciji** glasi

$$-2.777 \times 10^1,$$

pri čemu je:

- $-$ **predznak** broja,
- 2.777 **mantisa** ili razlomljeni dio broja,
- 10 **baza**,
- 1 **eksponent**.

Općenito, svaki se realni broj može na jedinstven način zapisati u obliku

$$\pm m \times 10^e, \quad 1 \leq m < 10.$$

Računala koriste sličan zapis, ali ne u bazi 10 već u bazi 2.

Npr. broj $3.625_{10} = 11.1011_2$ (zapisan u bazi 2) u znanstvenoj notaciji ima zapis

$$\begin{aligned} 11.1011_2 &= 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} \\ &= \left(1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5}\right) \cdot 2^1 \\ &= 1.11011_2 \cdot 2^1 \end{aligned}$$

Malo pogledat:

1.1 Pretvaranje decimalne u binarnu reprezentaciju

Neka broj x ima konačan binarni zapis. Tada x možemo zapisati kao sumu njegovog cijelog i razlomljenog dijela na način:

$$\begin{aligned} x &= (a_k a_{k-1} \cdots a_1 a_0 . b_1 b_2 \cdots b_{l-1} b_l)_2 = x_c + x_r \\ x_c &= (a_k a_{k-1} \cdots a_1 a_0)_2 \\ &= a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2^1 + a_0 2^0 \\ x_r &= (.b_1 b_2 \cdots b_{l-1} b_l)_2 \\ &= b_1 2^{-1} + b_2 2^{-2} + \cdots + b_{l-1} 2^{-l+1} b_l 2^{-l}. \end{aligned}$$

1.2 Reprezentacija brojeva u računalu

U programskim jezicima postoji nekoliko vrsta aritmetika koje korite posve određene tipove podataka.

- **Cjelobrojna aritmetika** koristi cjelobrojni tip podataka koji čine konačan interval u skupu cijelih brojeva $\mathbb{Z}$.
- **Realna aritmetika** koristi realni tip podataka koji čine interval s konačnim granicama u skupu racionalnih brojeva $\mathbb{Q}$.
- **Aritmetika s dvostrukom preciznošću** koristi zapis realnih brojeva kod koga je mantisa dvostruko dulja od mantise u standardnom zapisu brojeva realnog tipa.
- **Kompleksna aritmetika** koristi kompleksni tip podataka koji čine konačan podskup skupa kompleksnih brojeva $\mathbb{C}$.

Općenito, današnja računala imaju ćelije od 32 bita, pa je tomu prilagođena i aritmetika. Iznimno se koriste i računala s 64 bitnim ćelijama.

1.2.1 Reprezentacija cijelih brojeva u računalu

Pozitivni cijeli brojevi reprezentiraju se u 32 bitnoj ćeliji kao desno pozicionirani binarni brojevi. Npr., broj $(111)_{10} = (1101111)_2$ bit će smješten kao

$$0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.1.0.1.1.1.1.$$

Na taj način možemo prikazati sve brojeve od nule (32 nule) do $2^{32} - 1$ (32 jedinice). Međutim, ako moramo spremati i negativne brojeve, onda se situacija mijenja.

Prva je ideja potrošiti jedan bit za predznak što bi nam omogućilo prikaz brojeva od $-2^{31} + 1$ do $2^{31} - 1$. Ipak, gotovo sva računala koriste pametniji zapis: on se zove **drugi komplement**.

1.2.2 Reprezentacija realnih brojeva u računalu

U računalu se realni brojevi reprezentiraju u **znanstvenoj notaciji**

$$x = \pm m \times 2^e, \quad 1 \leq m < 2.$$

Dakle,

$$m = (b_0.b_1b_2 \cdots b_{p-1})_2, \quad b_0 = 1.$$

Ovakav prikaz nazivamo **normaliziranom reprezentacijom** broja x.
Na primjer,

$$(111.5)_{10} = (1101111.1)_2 = (1.1011111)_2 \times 2^6.$$

Da bismo ovako reprezentiran broj spremili u računalo 32 bitnu ćeliju razdijelimo na sljedeći način:

1. 23 bita za mantisu (za razlomljeni dio, b_0 je tzv. skriveni bit),
2. 8 bitova za eksponent (od -128 do 127),
3. 1 bit za predznak (0 za pozitivan, a 1 za negativan predznak).

Broj x nazivamo **egzaktno reprezentabilnim** u računalu ako se na opisani način može bez greške smjestiti u računalo. Ako broj nije egzaktno reprezentabilan u računalu, onda se mora prije smještanja u računalu **zaokružiti**.

1.2.3 Strojni ε, ulp i preciznost

Preciznost definiramo kao broj bitova u mantisi pri čemu se računa i skriveni bit. U opisanom sustavu je $p = 24$. Općenito, u sustavu s preciznošću p, normalizirani broj s pomičnom točkom ima oblik

$$\pm (b_0.b_1b_2 \cdots b_{p-1}b_{p-1})_2 \times 2^e. \tag{1}$$

Najmanji takav broj već od 1 je broj

$$(1.00 \cdots 01)_2 \times 2^0 = 1 + 2^{-p+1}.$$

Razmak između njega i jedinice naziva se **strojni** ε i piše se

$$\varepsilon_M = 2^{-p+1}.$$

Za x reprezentiran kao u (1) definiramo **ulp** (*unit in the last place*) kao

$$\text{ulp}\,(x) = (0.00 \cdots 01)_2 \times 2^e = 2^{-p+1} \times 2^e = \varepsilon_M \times 2^e.$$

Ako je $x > 0$ $(x < 0)$, onda je ulp(x) razmak između x i sljedećeg većeg (manjeg) reprezentabilnog broja.

1.2.4 BSP ili brojevni sustav za prikazivanje

Da bismo shvatili koje točke na realnom pravcu odgovaraju reprezentabilnim brojevima uvodimo **brojevni sustav za prikazivanje** ili kraće BSP. On se sastoji od brojeva oblika

$$\pm (b_0.b_1b_2)_2 \times 2^e, \quad e \in \{-1, 0, 1\}.$$

Očito, preciznost je $p = 3$, najveći prikazivi broj je

$$(1.11)_2 \times 2^1 = (3.5)_{10},$$

a najmanji

$$(1.00)_2 \times 2^{-1} = (0.5)_{10}.$$

Kako je desni susjed broja 1 u BSP-u broj 1.25, to je $\varepsilon_M = 0.25$. Brojevi za koje je $e = 0$ su 1, 1.25, 1.5 i 1.75. Između svih je isti razmak ulp$(x) = \epsilon_M$. Brojevi za koje je $e = 1$ su 2, 2.5, 3 i 3.5, a između svih je isti razmak ulp$(x) = 2\epsilon_M$. Brojevi za koje je $e = -1$ su 0.5, 0.625, 0.75 i 0.825, a između svih je isti razmak ulp$(x) = \epsilon_M/2$.

Dakle, općenito je razmak između nekog broja i njegovog desnog susjeda u BSP-u jednak

$$ulp\,(x) = \epsilon_M \times 2^e.$$

Uočimo da razmaci nisu ravnomjerno raspoređeni!

2 IEEE aritmetika

Ovo je standard smještanja brojeva u računalo i računalne aritmetike koji je 1985. ustanovljen između vodećih proizvođača procesora (Intel i Motorola) te znanstvenika iz Berkeley-a.

Bitni zahtjevi IEEE standarda su:

- konzistentna reprezentacija brojeva s pomičnom točkom na svim računalima koja prihvaćaju standard;
- korektno zaokruživanje kod svih računalnih operacija u svim načinima rada;
- konzistentno tretiranje izvanrednih situacija (npr. kao što je dijeljenje s nulom).

U ovom standardu je vodeća jedinica skrivena, pa je potreban poseban načini prikaza nule. Također se posebno prikazuju $+\infty$ i $-\infty$ (to je u ovom standardu isti broj!) i neki posebni izmišljeni brojevi (kao $0/0$).
Ovaj standard razlikuje dva osnovna formata: **jednostruki** i **dvostruki**.

Evo nekih važnih podataka:

- **najmanji eksponent** je -126
- **najveći eksponent** je 127
- **najmanji normalizirani broj** je

$$N_{\min} = (1.00\ldots0)_2 \times 2^{-126} \approx 1.1755 \times 10^{-38}$$

  a reprezentira se kao $\boxed{0}\boxed{00000001}\boxed{000\cdots00}$

- **najveći normalizirani broj** je

$$N_{\max} = (1.11\ldots1)_2 \times 2^{127} \approx 3.4028 \times 10^{38}$$

  a reprezentira se kao $\boxed{0}\boxed{11111110}\boxed{111\cdots11}$

- niz bitova samih jedinica u eksponencijalnom dijelu reprezentacije vodi na $\pm\infty$ ako su u mantisi same nule, a inače dobivamo oznaku NaN (not a number)

Postoje još i tzv. **subnormalni** ili **denormalizirani** brojevi kojima je, kao i nuli, skriveni bit jednak 0. Najmanji takav pozitivan broj je

$$(0.00\ldots1)_2 \times 2^{-126} = 2^{-149} < N_{\min},$$

dok je najveći

$$(0.11\ldots1)_2 \times 2^{-126} - N_{\min} - 2^{149}.$$

Subnormalni brojevi su manje točni od normaliziranih.

2.2 Dvostruki format

Kod zahtjevnijih računanja jednostruki format nije dovoljno dobar, kako zbog neizbježnih zaokruživanja, tako i zbog premalog raspona brojeva. Zato IEEE standard specificira i tzv. dvostruki format koji koristi 64 bitnu riječ

$\pm$	$a_1a_2\ldots a_{11}$	$b_1b_2\cdots b_{52}$

Koncept je isti kao kod jednostrukog formata. Neki važni podaci su:

- **najmanji eksponent** je -1022
- **najveći eksponent** je 1023
- **najmanji normalizirani broj** je

$$N_{\min} = 2^{-1022} \approx 2.225 \times 10^{-308}$$

- **najveći normalizirani broj** je

$$N_{\max} = \left(2 - 2^{-52}\right) \times 2^{1023} \approx 1.797693 \times 10^{308}$$

2.3 BSPT (Brojevni Sustav s Pomičnom Točkom)

Za realni broj x kažemo da leži u **intervalu normaliziranih brojeva** sustava s pomičnom točkom ako vrijedi

$$N_{\min} \leq x \leq N_{\max}.$$

Iz ovoga odmah slijedi da brojevi ± 0 i $\pm\infty$ nisu u tom intervalu, iako pripadaju spomenutom sustavu.

Pretpostavimo sada da je x neki realni broj koji nije reprezentabilan u sustavu brojeva s pomičnom točkom. Tada je točna barem jedna od sljedećih tvrdnji:

- x leži izvan intervala normaliziranih brojeva;
- binarna reprezentacija broja x zahtjeva više od p bitova za egzaktnu reprezentaciju.

U oba slučaja potrebno je x zamijeniti brojem iz sustava brojeva s pomičnom točkom.

Malo pogledati:

- Razmotrimo ovu drugu mogućnost. Neka je

$$x_- \leq x \leq x_+,$$

pri čemu su brojevi s pomičnom točkom x_- i x_+ brojevi skupa BPT najbliži broju x. Ako pretpostavimo da je

$$x = (1.b_1b_2\cdots b_{p-1}b_pb_{p+1}\cdots)_2 \times 2^e,$$

onda su ti brojevi upravo

$$\begin{aligned} x_- &= (1.b_1b_2\cdots b_{p-1})_2 \times 2^e \\ x_+ &= \left[(1.b_1b_2\cdots b_{p-1})_2 + (0.00\cdots 01)\right] \times 2^e, \end{aligned}$$

a razmak između x_- i x_+ je $\text{ulp}(x_-) = 2^{p-1} \times 2^e$.

- Ako je $x > N_{\max}$, onda je $x_- = N_{\max}$ i $x_+ = +\infty$.
- Ako je $0 < x < N_{\min}$, onda je $x_- = 0$ ili je x_- neki subnormalni broj, a x_+ je subnormalni broj ili $N_{\min}$.
- Ako je x negativan, onda je situacija zrcalna u odnosu na ishodište.

IEEE standard definira **korektno zaokruženu vrijednost** broja x, u oznaci **round**(x), na način koji slijedi. Ako je x broj s pomičnom točkom, onda je round$(x) = x$. Ako nije, onda se round(x) odredi na jedan od četiri načina ovisno o načinu (modu) zaokruživanja koji je aktivan. Ti načini su:

- round$(x) = x_-$ (**zaokruživanje prema dolje**),
- round$(x) = x_+$ (**zaokruživanje prema gore**),
- round$(x) = \begin{cases} x_- , & x > 0 \\ x_+ , & x < 0 \end{cases}$ (**zaokruživanje prema nuli**)
- round$(x) = \begin{cases} x_- , & |x - x_-| < |x - x_+| \\ x_+ , & |x - x_-| > |x - x_+| \end{cases}$ (**zaokruživanje prema najbližem**);

  Ako je $|x - x_-| = |x - x_+|$, onda se uzme x_- ili x_+, već prema tome je li u x_- ili u x_+ najmanje značajan bit b_{p-1} nula.

Ako je $x < 0$ i $|x| > N_{\max}$ uzima se round$(x) = -\infty$, a ako je $x > N_{\max}$ uzima se round$(x) = +\infty$.
U IEEE standardu se po defaultu uzima četvrti način.

2.4 Korektno zaokruživanje kod računskih operacija

Jedna od najvažnijih značajki IEEE standarda jest zahtjev da se prilikom izvođenja osnovnih računskih operacija rezultat dobiva kao da je izračunat točno i zatim zaokružen.

Označimo redom s $\oplus, \ominus, \otimes$ i $\oslash$ operacije $+, -, \times$ i $/$ kako su stvarno implementirane u računalu. Također označimo s $\circ$ proizvoljni element skupa $\{+, -, \times, /\}$, a s $\odot$ proizvoljni element skupa $\{\oplus, \ominus, \otimes, \oslash\}$. Za IEEE standard vrijedi

$$x \odot y = fl(x \circ y) = \text{round}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq u.$$

No naglasimo da osim za ove četiri osnovne operacije standard mora vrijediti i za unarnu operaciju drugog korijena. Aritmetika koja zadovoljava ovaj uvijet ponekad se naziva i **aritmetika s korektnim zaokruživanjem**.

2.5 Implementacija operacija u računalu

2.5.1 Zbrajanje

Napomenimo najprije da nećemo posebno razlikovati oduzimanje od zbrajanja jer je

$$x - y = x + (-y).$$

Ako operandi nemaju isti eksponent, onda se po modulu manji operand napiše u obliku nenormaliziranog broja s eksponentom većeg. To znači da mu se mantisa pomakne u desno za onoliko binarnih mjesta kolika je bila razlika u eksponentima operanada. Nakon izvršenja operacije rezultat se svede na normalizirani oblik, pri čemu se prvo mantisa opet pomakne u lijevo ili u desno (po potrebi), zaokruži, i ako je potrebno, mantisa se opet pomakne lijevo ili desno.

Ipak, u praksi je dovoljno koristiti tek tri dodatna bita, točnije dva **zaštitna** i jedan **zalijepljeni** bit. Zovemo ga zalijepljeni jer se on aktivira tek onda kada je potrebno pomaknuti mantisu za više od dva mjesta, a jednom kada se postavi više se ne mijenja. U našem

Malo pogledati:

2.5.2 Množenje i dijeljenje

Ove dvije operacije u BSPT ne zahtijevaju poravnavanje eksponenata. Ako je

$$x = m_x \times 2^{e_x}, \qquad y = m_y \times 2^{e_y},$$

onda je

$$z = x \cdot y = (m_x \cdot m_y) \times 2^{e_x + e_y},$$
$$w = x/y = (m_x / m_y) \times 2^{e_x - e_y},$$

nakon čega se primjenjuju normalizacija i pravilno zaokruživanje.
Današnji dizajneri čipova su postigli to da uz dovoljnu rezervu memorije množenje bude gotovo jednako brzo kao zbrajanje, no s dijeljenjem to nije slučaj.
Uočimo da zbog $1 \leq m_x < 2$ i $1 \leq m_y < 2$ vrijedi

$$1 \leq m_z < 4.$$

2.6 Konverzija formata

IEEE standard zahtijeva podršku za prevođenje između raznih vrsta formata. Tu spada:

- Konverzija iz kraćega u dulji format koja mora biti egzaktna.
- Konverzija iz duljega u kraći format koja zahtijeva korektno zaokruživanje.
- Konverzija iz BSPT u cjelobrojni format koja zahtijeva zaokruživanje na najbliži cijeli broj. Ako je početni broj cijeli, onda rezultat mora biti taj isti broj (osim ako nema reprezentaciju u cjelobrojnom formatu!).
- Konverzija iz cjelobrojnog formata u BSPT koja može zahtijevati zaokruživanje.
- Konverzija iz decimalnog u binarni sustav i obratno.

2.8 Prekoračenje i potkoračenje

O **prekoračenju** se govori kada je egzaktan rezultat neke operacije konačan broj, ali po modulu veći od $N_{\max}$. O **potkoračenju** se govori kada je egzaktan rezultat neke operacije broj različit od nule, ali po modulu manji od $N_{\min}$. Postupamo na sljedeći način:

- ako je nastupilo prekoračenje zaokružimo dani broj na $\pm\infty$ ili $\pm N_{\max}$ ovisno o tipu zaokruživanja;
- ako je nastupilo potkoračenje dani broj pravilno zaokružimo (rezultat je subnormalni broj, ± 0 ili $\pm N_{\min}$).

Ovaj drugi postupak je najkontraverzniji dio IEEE standarda jer daje manju relativnu točnost rezultata (to manju što je potkoračenje veće), no upravo ono dovodi do toga da u BSPT vrijedi implikacija

$$x \ominus y = 0 \Longrightarrow x = y.$$

Predavanje 3: Analiza pogrešaka

(Obratiti pozornost na **BOLDANO**)

1 Stabilnost numeričkog računanja

Sada ćemo se pozabaviti stabilnošću numeričkih algoritama, a s čime je usko povezana i pouzdanost dobivenih rješenja. Kroz primjere ćemo se upoznati s nekim nepoželjnim fenomenima koji se mogu pojaviti prilikom korištenja aritmetike računala.

1.1 Greške unazad i unaprijed

Neka je f realna funkcija jedne varijable. Pretpostavimo da je u aritmetici preciznosti u izračunata vrijednost $y = f(x)$ i da ona iznosi $\hat{y}$. Kako možemo mjeriti kvalitetu dobivenog $\hat{y}$ kao aproksimacije točnog y?

U većini slučajeva bit ćemo sretni ako postignemo neku malu relativnu grešku, no to neće uvijek biti moguće. Umjesto toga možemo se zapitati: Za koji stvari skup podataka smo zaista riješili problem? Dakle, za koji $\triangle x$ vrijedi

$$\hat{y} = f(x + \triangle x) \quad ?$$

Općenito može biti i više takvih $\triangle x$, a nas će zanimati najveći od njih, $\max|\triangle x|$ (ponekad se uzima i podijeljeno s $|x|$). Zove se **greška unatrag** ili **povratna greška**. S druge strane, apsolutna i relativna greška po funkcijskoj vrijednosti zovu se **greške unaprijed** ili jednostavno **greške**. Proces omeđivanja povratne greške izračunatog rješenja zove se **analiza povratne greške**, a motivacija za njega je dvostruka.

- **Omogućava interpretaciju grešaka zaokruživanja kao grešaka u ulaznim podacima**.
  Podaci često kriju netočnosti nastale usljed prethodnih računanja, nepreciznih rezultata mjerenja i spremanja podataka u računalo. Ako povratna greška nije veća od polaznih netočnosti, onda je dobiveno rješenje točno "do na ulaznu netočnost".

- **Reducira problem omeđivanja greške unaprijed na primjenu teorije perturbacije za dani problem**.
  Naime, teorija perturbacije je dobro poznata za većinu problema i važno je da ovisi o samom problemu, a ne o metodi koju koristimo. Kada imamo ocjenu greške unatrag rješenja promatrane metode, onda primjenom opće teorije perturbacije za dani problem lako dođemo do ocjene greške unaprijed.

Metoda za računanje vrijednosti $y = f(x)$ je **stabilna unazad** ili **povratno stabilna** ako za svaki x producira izračunati $\hat{y}$ s malom povratnom greškom, tj. ako vrijedi

$$\hat{y} = f(x + \triangle x) \qquad (1)$$

za neki mali $\triangle x$. Pri tom značenje izraza "mala" povratna greška ovisi o kontekstu.
U načelu, za dani problem može postojati više metoda rješavanja od kojih će neke biti povratno stabilne, a neke neće.
Npr. sve osnovne računske operacije u računalu zadovoljavaju relaciju (1), pa daju rezultat koji je točan za malo pomaknute ulazne podatke:

$$x \to x(1+\delta) \quad \text{i} \quad y \to y(1+\delta), \quad |\delta| \leq u.$$

Međutim, većina metoda za računanje funkcije cos ne zadovoljava relaciju (1), već nešto slabiju

$$\hat{y} + \triangle y = \cos(x + \triangle x)$$

za neke male $\triangle x$ i $\triangle y$.

Općenito, greška u rezultatu koji se zapisuje kao

$$\hat{y} + \triangle y = f(x + \triangle x), \qquad |\triangle x| \leq \xi |x|, \quad |\triangle y| \leq \eta |y| \qquad (2)$$

naziva se **miješana naprijed-nazad greška.**

Može se reći ovako:
Izračunato rješenje $\hat{y}$ jedva se razlikuje od vrijednosti $\hat{y} + \triangle y$ koja se dobije egzaktnim računom na ulaznoj vrijednosti $x + \triangle x$ koja se jedva razlikuje od točne ulazne vrijednosti x.

Algoritam je **numerički stabilan** ako je stabilan u smislu relacije (2) s malim ξ i η.
Ova definicija se uglavnom odnosi na izračunavanja u kojima su greške zaokruživanja osnovnih aritmetičkih operacija dominantni oblici grešaka. U drugim područjima numeričke analize ovaj pojam ima različita značenja.

1.2 Uvjetovanost

Odnos između greške unazad i greške unaprijed za neki dani problem u velikoj je mjeri određen **uvjetovanošću problema**, tj. osjetljivošću rješenja problema na ulazne podatke.

Pretpostavimo da je dano približno rješenje $\hat{y}$ problema $y = f(x)$ koje zadovoljava relaciju

$$\hat{y} = f(x + \triangle x).$$

Ako pretpostavimo da je funkcija f dvaput neprekidno derivabilna, onda razvoj u Taylorov red daje

$$\hat{y} - y = f(x + \triangle x) - f(x) = f'(x)\triangle x + \frac{f''(x + \Theta \triangle x)}{2!}(\triangle x)^2, \quad \Theta \in (0,1),$$

i možemo ocijeniti desnu stranu ove jednakosti.

Zbog

$$\frac{\hat{y}-y}{y}=\frac{f'(x)}{f(x)}\triangle x+\frac{f''(x+\Theta\triangle x)}{2f(x)}(\triangle x)^2$$

imamo

$$\frac{\hat{y}-y}{y}=\frac{xf'(x)}{f(x)}\frac{\triangle x}{x}+\mathcal{O}(\triangle x)^2,$$

pa veličina

$$\kappa(f)(x)=\left|\frac{xf'(x)}{f(x)}\right|$$

mjeri relativnu promjenu y za malu relativnu promjenu x.

Zato κ zovemo **uvjetovanost** funkcije f. Ako je f funkcija više varijabla, onda se u izrazu (3) umjesto apsolutne vrijednosti javlja norma.
Uvjetovanost služi za mjerenje najveće relativne promjene koja se dostiže za neku vrijednost broja x ili vektora $\boldsymbol{x}$.
Na primjer, ako je f logaritamska funkcija, onda je

$$\kappa(f)(x)=\frac{1}{|\ln(x)|},$$

pa je uvjetovanost jako velika za $x\approx 1$.

Kada se greške unatrag i unaprijed te uvjetovanost za neki problem definiraju na konzistentan način, vrijedi jednostavno pravilo:

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

Dakle, izračunato rješenje loše uvjetovanog problema može imati veliku grešku unaprijed. Zato se uvodi sljedeća definicija.

DEFINICIJA. *Ako metoda daje rješenja s greškama unaprijed koja su sličnog reda veličine kao ona koja se dobiju primjenom povratno stabilne metode, onda se za metodu kaže da je stabilna unaprijed.*

1.3 Akumulacija grešaka zaokruživanja

Rasprostranjeno je mišljenje da velike brzine modernih računala koja u svakoj sekundi izvršavaju i nekoliko milijardi računskih operacija imaju za posljedicu potencijalno velike greške u rezultatu. Na sreću, ta tvrdnja uglavnom nije istinita, a u rijetkim slučajevima kada dolazi do većih grešaka u rezultatu, kriva je jedna ili tek nekoliko podmuklih grešaka zaokruživanja.

1.4 Kraćenje

Kraćenje nastaje kada se oduzimaju dva bliska broja. To najčešće, iako ne uvijek, ima za posljedicu netočan rezultat.

Iz ovoga se vidi da ni desetoroznamenkasta aproksimacija vrijednosti $\cos(x)$ nije dovoljno točna da bi izračunata vrijednost $f(x)$ imala barem jednu točnu značajnu znamenku.

Problem je u tomu što (iako je oduzimanje egzaktno) $1-\cos(x)$ ima samo jednu značajnu znamenku, pa je rezultat iste veličine kao i greška u $\cos(x)$. Drugim riječima, **oduzimanje podiže značaj prethodne greške!**

Da bismo dobili dublji uvid u fenomen kraćenja pogledajmo oduzimanje

$$\hat{x} = \hat{a} - \hat{b}$$

u egzaktnoj aritmetici, gdje su

$$\hat{a} = a(1+\triangle_a), \quad \hat{b} = b(1+\triangle_b).$$

Članovi $\triangle_a$ i $\triangle_b$ su relativne greške koje unosimo u račun. Izraz za relativnu grešku rezultata oduzimanja daje

$$\left|\frac{x-\hat{x}}{x}\right| = \left|\frac{-a\triangle_a + b\triangle_b}{a-b}\right| \leq \max\{|\triangle_a|, |\triangle_b|\}\frac{|a|+|b|}{|a-b|}.$$

Očito je ograda za relativnu grešku visoka ako je

$$|a-b| \ll |a|+|b|,$$

a to je istina kada postoji bitno kraćenje u oduzimanju. Ova analiza pokazuje da se zbog kraćenja postojeće greške ili netočnosti u $\hat{a}$ i $\hat{b}$ povećavaju. Drugim riječima, **kraćenje dovodi prethodne greške na vidjelo**.

Ipak, postoje situacije kada kraćenje neće dovesti do loših pojava, a to su npr. sljedeće:

- ulazni podaci su točni,
- uvjetovanost je loša, pa je kraćenje nužnost,
- utjecaj kraćenja na daljnji račun nije loš (npr. kod $x+(y-z)$ ako je $x \gg y \approx z > 0$),
- kraćenje grešaka zaokruživanja.

1.5 Kako dizajnirati stabilne algoritme

Najprije naglasimo da je numerička stabilnost važnija od drugih karakteristika algoritma, kao što su npr. broj računskih operacija, paralelizacija, ušteda memorije i slično. Evo nekih općih uputa:

- izbjegavati oduzimanje bliskih brojeva koji nose greške
- minimizirati veličinu međurezultata u odnosu na konačni rezultat
- iskušavati razne formulacije istog problema
- koristiti jednostavne formule za ažuriranje tipa

$$\text{nova vrijednost} = \text{stara vrijednost} + \text{mala korekcija}$$

  ako se korekcija može izračunati na dovoljan broj značajnih znamenki
- koristiti samo dobro uvjetovane transformacije
- poduzimati mjere opreza protiv prekoračenja i potkoračenja
- koristiti što manje cijepanje formula u više programskih linija uvođenjem pomoćnih varijabla jer CPU često koristi precizniju aritmetiku za operande u registrima, dok zaokruživanje nastupa tek prilikom spremanja u memoriju.

Predavanje 4: Sustavi linearnih jednadžbi

Teorem kod LU(LR) faktorizacije

TEOREM. Neka je $A \in \mathbb{R}^{n\times n}$ i neka su determinante glavnih podmatrica $A(1:k,1:k)$ različite od nule za $k = 1, 2, \ldots, n-1$. Tada postoji donjetrokutasta matrica L s jedinicama na dijagonali i gornjetrokutasta matrica U tako da vrijedi $A = LU$. Ako faktorizacija $A = LU$ postoji i ako je matrica A regularna, onda je ova faktorizacija jedinstvena. Tada je i

$$\det A = \prod_{i=1}^{n} u_{ii}.$$

TEOREM. Neka je $A \in \mathbb{R}^{n\times n}$ proizvoljna matrica. Tada postoj permutacija P takva da Gaussove eliminacije daju LU faktorizaciju $PA = LU$ matrice PA. Matrica $L = [l_{ij}]$ je donjetrokutasta s jedinicama na dijagonali, a $U = [u_{ij}]$ je gornjetrokutasta matrica. Pri tome, ako je P umnožak od p inverzija, vrijedi

$$\det A = (-1)^p \prod_{i=1}^{n} u_{ii}.$$

Ako su matrice $P^{(k)}$ odabrane tako da vrijedi

$$\left|\left(P^{(k)}A^{(k-1)}\right)_{kk}\right| = \max_{1\le j\le n}\left|\left(P^{(k)}A^{(k-1)}\right)_{jk}\right|$$

onda je

$$\max_{1\le k\le n}\max_{1\le i,j\le n}\left|\left(L^{(k)}\right)_{ij}\right| = \max_{1\le i,j\le n}|l_{ij}| = 1.$$

U tom slučaju faktorizaciju $PA = LU$ nazivamo LU **faktorizacijom s pivotiranjem redaka**.

TEOREM. Neka je $\widetilde{\boldsymbol{x}}$ rješenje regularnog $n \times n$ sustava $A\boldsymbol{x} = \boldsymbol{b}$ dobiveno Gaussovim eliminacijama s pivotiranjem redaka u aritmetici s preciznošću ε takvom da je $2n\varepsilon < 1$. Tada postoji perturbacija ΔA matrice A za koju vrijedi

$$(A + \Delta A)\,\widetilde{\boldsymbol{x}} = \boldsymbol{b},$$

pri čemu je

$$|\Delta A| \le \frac{5n\varepsilon}{1-2n\varepsilon} P^T \left|\widetilde{L}\right| \left|\widetilde{U}\right|.$$

6 Iterativne metode

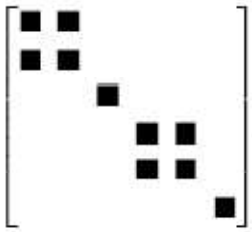
U prethodnom smo vidjeli da se rješenje linearnog sustava $A\boldsymbol{x} = \boldsymbol{b}$ općenito ne može izračunati potpuno točno: najčešće dobivamo rješenje $\widetilde{\boldsymbol{x}}$ koje zadovoljava sustav

$$(A + \delta A)\,\widetilde{\boldsymbol{x}} = \boldsymbol{b}$$

blizak polaznom sustavu. Dakle, Gaussove eliminacije ne garantiraju idealnu točnost.

Osim toga, u praksi moramo biti svjesni da je računalo ograničeno ne samo po pitanju numeričke točnosti, već i raspoloživim vremenom i memorijom. U primijenjenoj matematici najčešće se javljaju sustavi velikih dimenzija $\left(n > 10^5\right)$ kod kojih je proces Gaussovih eliminacija iz više razloga praktički neprovediv. Npr. spremanje matrice sustava s $n = 10^5$ nepoznanica zahtijeva 10^{10} memorijskih lokacija, pa već i to može predstavljati poteškoću. No važno je istaknuti da su matrice takvih sustava često **rijetko popunjene** (tj. velika većina elemenata im je jednaka nuli, a elementi koji nisu nula su obično pravilno raspoređeni).

Rijetko popunjena matrica po blokovima može izgledati npr. ovako (praznine su blokovi nula):



Kada je matrica velika i gusto popunjena ono što preostaje jest učitavanje djelova matrice iz vanjske u radnu memoriju.

No ponekad je poznat način na koji se generiraju elementi matrice A, pa nas zanima kako postupiti u takvom slučaju pod uvjetom da ne tražimo egzaktno rješenje x, već dovoljno dobru aproksimaciju $\widetilde{x}$. Stoga ima smisla konstruirati niz $x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots$ vektora iz $\mathbb{R}^n$ sa sljedećim svojstvima:

- za svaki $k \in \mathbb{N}_0$ formula za računanje $x^{(k)}$ je jednostavna;
- $x^{(k)}$ teži prema $x = A^{-1}b$ za neki k (obično je takav $k << n$).

Kako točno konstruirati takav niz ovisit će o konkretnom problemu kojeg rješavamo.

6.1 Jacobijeva metoda

Jacobijeva metoda je jedna od najjednostavnijih klasičnih iterativnih metoda za rješavanje linearnih sustava. Ideju same metode ilustrirat ćemo na jednostavnom primjeru 2×2 sustava.

Neka je dan sustav

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2, \end{aligned}$$

pri čemu je $a_{11} \neq 0$ i $a_{22} \neq 0$. Uočimo da rješenje x zadovoljava uvjete

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2) \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1). \end{aligned}$$

Te nas relacije motiviraju da neku približnu vrijednost rješenja $x^{(0)} = \begin{bmatrix} x_1^{(0)} & x_2^{(0)} \end{bmatrix}^T$ korigiramo pomoću formula

$$x_1^{(1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(0)}\right)$$
$$x_2^{(1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(0)}\right).$$

Naravno, nadamo se da je $\boldsymbol{x}^{(1)}$ bolja aproksimacija egzaktnog rješenja $\boldsymbol{x}$ nego $\boldsymbol{x}^{(0)}$. Postupak možemo nastaviti tako da pomoću $\boldsymbol{x}^{(1)}$ izračunamo na isti način $\boldsymbol{x}^{(2)}$ itd. Pitanje je pod kojim uvjetima tako dobivene iteracije teže prema rješenju $\boldsymbol{x}$?

Uočimo da vrijedi

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} 1/a_{11} & 0 \\ 0 & 1/a_{22} \end{bmatrix} \left(\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} 0 & -a_{12} \\ -a_{21} & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} \right).$$

Dakle, ako stavimo

$$A = D - N, \quad D = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix}, \quad N = \begin{bmatrix} 0 & -a_{12} \\ -a_{21} & 0 \end{bmatrix},$$

možemo jednostavno pisati

$$\boldsymbol{x}^{(k+1)} = D^{-1}\left(\boldsymbol{b} + N\boldsymbol{x}^{(k)}\right) = D^{-1}N\boldsymbol{x}^{(k)} + D^{-1}\boldsymbol{b}.$$

Upravo ovom relacijom definirana je **Jacobijeva iterativna metoda**.

PROPOZICIJA. Ako je u rastavu $A = D - N$ u nekoj matričnoj normi ispunjeno

$$\left\|D^{-1}N\right\| < 1,$$

onda za svaku početnu iteraciju $\boldsymbol{x}^{(0)}$ niz

$$\boldsymbol{x}^{(k+1)} = D^{-1}\left(\boldsymbol{b} + N\boldsymbol{x}^{(k)}\right), \quad k \in \mathbb{N}_0,$$

konvergira rješenju $\boldsymbol{x}$ sustava $A\boldsymbol{x} = \boldsymbol{b}$.

6.2 Gauss-Seidelova metoda

Vidjeli smo da se u primjeru danom za Jacobijevu metodu $x_1^{(1)}$ i $x_2^{(1)}$ računaju neovisno pomoću $x_1^{(0)}$ i $x_2^{(0)}$. No imalo bi smisla u formuli za $x_2^{(1)}$ koristiti upravo izračunatu vrijednost $x_1^{(1)}$ jer je ona vjerojatno bolja od $x_1^{(0)}$. Općenito, Jacobijevu formulu za iteraciju modificiramo tako da prilikom računanja svake komponente vektora $\boldsymbol{x}^{(k+1)}$ koristimo najsvježije izračunate vrijednosti. Npr. u slučaju $n = 4$ imali bismo

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)}\right)$$
$$x_2^{(k+1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - a_{24}x_4^{(k)}\right)$$
$$x_3^{(k+1)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - a_{34}x_4^{(k)}\right)$$
$$x_4^{(k+1)} = \frac{1}{44}\left(b_4 - a_{41}x_1^{(k+1)} - a_{42}x_2^{(k+1)} - a_{43}x_3^{(k+1)}\right).$$

U općenitom slučaju imali bismo

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right).$$

No vratimo se primjeru $n = 4$. Stavimo li

$$L = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \qquad U = - \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

vrijedi $A = L - U$, pa uz uvjet regularnosti matrice L Gauss-Seidelovu metodu možemo zapisati kao

$$\boldsymbol{x}^{(k+1)} = L^{-1} \left(\boldsymbol{b} + U \boldsymbol{x}^{(k)} \right), \quad k \in \mathbb{N}_0,$$

a kao i u analizi Jacobijeve metode imamo

$$\boldsymbol{e}^{(k)} = \left(L^{-1} U \right)^k \boldsymbol{e}^{(0)}, \quad k \in \mathbb{N}.$$

Predavanje 5: Izvrednjavanje funkcija

2 Hornerova shema

Polinomi su najjednostavnije algebarske funkcije. Možemo ih definirati nad bilo kojim prstenom R u obliku

$$p(x) = \sum_{i=0}^{n} a_i x^i, \quad n \in \mathbb{N}_0,$$

gdje su a_i koeficijenti iz prstena R, a x simbolička varijabla. Polinomi, kao simbolički objekti, imaju i sami strukturu prstena.

Međutim, polinome možemo interpretirati i kao funkcije koje se daju izvrednjavati u svim točkama x_0 prstena R uvrštavanjem x_0 umjesto simboličke varijable x. Dobiveni rezultat $p(x_0)$ je opet u R. Nas zanimaju efikasni algoritmi za računaje te vrijednosti. Složenost postupka očito ovisi o broju članova u sumi, a da broj članova ne bi bio umjetno prevelik uzimamo da je $p \neq 0$ i $a_n \neq 0$ (tj. $\partial p = n$).

TEOREM (Borodin, Muro) Za izvrednjavanje općeg polinoma n-tog stupnja potrebno je barem n aktivnih množenja (tj. množenja između a_i i x).

Napomenimo da se rezultat ovog teorema može poboljšati samo ako izvrednjavamo isti polinom u puno točaka. U tom se slučaju koeficijenti polinoma **adaptiraju** tako da bismo kasnije imali što manje operacija po svakoj pojedinoj točki. No u to nećemo ulaziti.

Zanimljivo je da je Hornerova shema optimalna za polinome stupnja ne većeg od tri čak i ako računamo vrijednost polinoma u više točaka. No pogledajmo jedan primjer adaptiranja koeficijenata za polinom stupnja 4.

TEOREM (Motzkin, Belaga) Slučajno odabrani polinom stupnja n ima vjerojatnost 0 da ga se može izvredniti za strogo manje od $\lceil (n+1)/2 \rceil$ množenja ili za strogo manje od n zbrajanja.

Posljedica ovog teorema je to da je Hornerova shema optimalna za izvrednjavanje gotovo svih polinoma.

Predavanje 6: Interpolacijski polinomi

2.1 Ortogonalni polinomi i neka njihova svojstva

Ortogonalni polinomi imaju niz dobrih svojstava zbog kojih se mogu konstruktivno primijenjivati u raznim granama numeričke matematike. Dat ćemo nekoliko teorema koji daju neka od najvažnijih svojstava ortogonalnih polinoma. Sva svojstva su direktna posljedica ortogonalnosti i ne ovise bitno o tome da li je skalarni umnožak diskretan ili kontinuiran. Mi ćemo standardno promatrati kontinuirani skalarni umnožak

$$\langle u, v \rangle = \int_a^b w(x)\, u(x)\, v(x)\, \mathrm{d}x$$

generiran nenegativnom težinskom funkcijom w na intervalu $[a, b]$.

Paralelno ćemo promatrati i diskretni skalarni umnožak

$$\langle u, v \rangle = \sum_{i=0}^{n} w_i u(x_i)\, v(x_i)$$

generiran međusobno različitim čvorovima $x_0, \ldots, x_n$ i pripadnim nenegativnim težinama $w_0, \ldots, w_n$.

Težinska funkcija w određuje ovakav sustav polinoma do na konstantni faktor u svakom od polinoma. Izbor takvog faktora zove se još i **standardizacija** ili **normalizacija**.

Analogno se definira ortogonalnost s obzirom na diskretni skalarni umnožak.

Može se pokazati da u oba slučaja u pripadnom unitarnom prostoru sigurno postoji baza ortogonalnih polinoma koju ćemo označavati s

$$\{p_n \mid n \geq 0\}.$$

Pri tom ćemo smatrati da je stupanj polinoma p_n upravo n.

TEOREM. Neka je $\{p_n \mid n \geq 0\}$ familija ortogonalnih polinoma na intervalu $[a, b]$ s nenegativnom težinskom funkcijom w. Ako je polinom f stupnja m, onda vrijedi

$$f = \sum_{n=0}^{m} \frac{\langle f, p_n \rangle}{\langle p_n, p_n \rangle} p_n.$$

TEOREM. Neka je $\{p_n \mid n \geq 0\}$ familija ortogonalnih polinoma na intervalu $[a, b]$ s nenegativnom težinskom funkcijom w. Tada svaki polinom p_n ima točno n različitih (jednostrukih) realnih nultočaka na otvorenom intervalu (a, b).

> za nultočke

2.2 Klasični ortogonalni polinomi

Samo ih nabrojat. To su:

1. **Čebiševljevi polinomi prve vrste**
2. **Čebiševljevi polinomi druge vrste**
3. **Legendreovi polinomi**
4. **Leguerreovi polinomi**
5. **Hermiteovi polinomi**

2.3 Egzistencija i jedinstvenost interpolacijskog polinoma

Sljedeći teorem nam u potpunosti rješava ključno pitanje egzistencije i jedinstvenosti rješenja problema polinomne interpolacije u njegovom najjednostavnijem obliku - kada su zadane funkcijske vrijednosti u međusobno različitim čvorovima. Takav oblik interpolacije obično zovemo **Lagrangeova interpolacija**.

TEOREM. Neka je $n \in \mathbb{N}_0$. Za zadane točke (x_k, f_k), $k = 0, 1, \ldots, n$, gdje je $x_i \neq x_j$ za $i \neq j$, postoji jedinstveni interpolacijski polinom p_n stupnja najviše n takav da vrijedi

$$p_n(x_k) = f_k, \quad k = 0, 1, \ldots, n.$$

Problemu izbora baze za prikaz interpolacijskog polinoma možemo pristupiti na tri načina.

- *"jednostavna baza, kompliciran koeficijenti"*
  U ovom slučaju baza ne ovisi o čvorovima, pa imamo osigurano brzo izvrednjavanje $p_n(x)$. No sva ovisnost o podacima ulazi u koeficijente polinoma p_n, pa je prva faza spora.
  Faza 1: $\mathcal{O}(n^3)$, prisutna nestabilnost. Faza 2: $\mathcal{O}(n)$, donekle prisutna nestabilnost.
- *"jednostavni koeficijenti, kompliciran baza"*
  U ovom slučaju koeficijenti jednostavno ovise o zadanim podacima i lako se računaju (npr. mogu čak biti određeni s $a_k = f_k$), no baza kompliciran ovisi o čvorovima. Druga faza je spora jer u svakoj točki x izvrednjavamo sve $b_k(x)$ koji i sami mogu svi biti stupnja n.
  Faza 1: $\mathcal{O}(n)$. Faza 2: Sporo izvrednjavanje.
- *"kompromis između jednostavnosti baze i koeficijenata"*
  Dopustimo da baza jednostavno ovisi o čvorovima i da koeficijenti u nekoj mjeri ovise o podacima, ali tako da dobijemo relativno jednostavne algoritme za obje faze.
  Faza 1: $\mathcal{O}(n^2)$. Faza 2: $\mathcal{O}(n)$.

2.5 Lagrangeov oblik interpolacijskog polinoma

Da bismo odredili koeficijente interpolacijskog polinoma nije nužno rješavati sustav jednadžbi. Interpolacijski polinom p_n možemo odmah napisati u tzv. Lagrangeovoj bazi $\mathcal{L} = \{l_0, l_1, \ldots, l_n\}$ prostora $\mathcal{P}_n$, pri čemu je

$$l_k(x) = \prod_{i=0,\ i\neq k}^{n} \frac{x - x_i}{x_k - x_i}, \quad k = 0, 1, \ldots, n.$$

Vidimo da su polinomi l_k stupnja n, pa je polinom p_n definiran s

$$p_n(x) = \sum_{k=0}^{n} f_k l_k(x)$$

najviše stupnja n.

Osim toga vrijedi

$$l_k(x_i) = \begin{cases} 0, & i \neq k \\ 1, & i = k \end{cases}.$$

Iz ovoga odmah slijedi da je

$$p_n(x_i) = \sum_{k=0}^{n} f_k l_k(x_i) = f_i, \quad i = 0, 1, \ldots, n.$$

Ovo je tzv. **Lagrangeov oblik interpolacijskog polinoma**.

2.6 Greška interpolacijskog polinoma

Ako su nam poznati jos neki podaci o funkciji f koju aproksimiramo, možemo napraviti i ocjenu greške interpolacijskog polinoma.

TEOREM. Pretpostavimo da funkcija f ima $(n+1)$.-u derivaciju na intervalu $[a,b]$ za neki $n \in \mathbb{N}_0$. Neka su $x_0, x_1, \ldots, x_n$ međusobno različiti interpolacijski čvorovi i neka je p_n interpolacijski polinom za funkciju f u tim čvorovima. Za bilo koju točku $x \in [a,b]$ postoji točka $\xi \in (x_{\min}, x_{\max})$, gdje je $x_{\min} = \min\{x_0, x_1, \ldots, x_n\}$ i $x_{\max} = \max\{x_0, x_1, \ldots, x_n\}$, takva da za grešku e interpolacijskog polinoma p_n vrijedi

$$e(x) = f(x) - p_n(x) = \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\xi),$$

gdje je

$$\omega(x) = \prod_{k=0}^{n} (x - x_k).$$

2.7 Newtonov interpolacijski polinom

Vidjeli smo da Lagrangeov oblik interpolacijskog polinoma nije pogodan kada želimo povećati stupanj interpolacijskog polinoma da bismo eventualno smanjili grešku zbog toga što se svi koeficijenti novog polinoma moraju računati ponovno. Postoji oblik interpolacijskog polinoma kod kojeg je puno lakše dodavati nove točke interpolacije, tj. povećavati stupanj interpolacijskog polinoma: to je tzv. **Newtonov oblik interpolacijskog polinoma**.

Oblik Newtonovog interpolacijskog polinoma:

$$\begin{aligned} p_n(x) = {} & f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ & + \cdots + (x_n - x_0) \cdots (x_n - x_{n-1}) f[x_0, x_1, \ldots, x_n]. \end{aligned}$$

Predavanje 7: Aproksimacija funkcija

1.1 Linearni splajn

Najjednostavniji interpolacijski splajn, linearni splajn S_1, određen je uvjetom interpolacije u čvorovima $\{x_0, \ldots, x_n\}$ i globalnom neprekidnošću na $[a, b]$. Lako se dobije da vrijedi

$$S_1(x) = f_i \frac{x_{i+1} - x_i}{h_i} + f_{i+1} \frac{x - x_i}{h_i} = f_i + \frac{x - x_i}{h_i}(f_{i+1} - f_i), \quad x \in [x_i, x_{i+1}]$$

za sve $i = 0, 1, \ldots, n-1$, pri čemu je $h_i = x_{i+1} - x_i$.
Kako je nalaženje samog splajna jednostavno odmah ćemo ispitati pogrešku.

LEMA. Ako je f neprekidna na $[a, b]$ i $\alpha, \beta \in \mathbb{R}$ istog predznaka, onda postoji $\xi \in [a, b]$ takav da vrijedi

$$\alpha f(a) + \beta f(b) = (\alpha + \beta) f(\xi).$$

1.2 Hermiteov kubični splajn

Hermiteov kubični splajn se razmatra na sličan način kao i Hermiteov interpolacijski polinom. Prvi je netrivijalni slučaj po djelovima kubični splajn s globalno neprekidnom derivacijom.

DEFINICIJA. Neka su u čvorovima $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ zadane vrijednosti f_i, f_i' za $i = 0, 1, \ldots, n$. Hermiteov kubični splajn je funkcija $H \in C^1[a, b]$ koja zadovoljava uvjete

1. Za svaki $i \in \{0, 1, \ldots, n-1\}$ vrijedi

$$H(x) = a_{i0} + a_{i1}(x - x_i) + a_{i2}(x - x_i)^2 + a_{i3}(x - x_i)^3, \; x \in [x_i, x_{i+1}].$$

2. Za svaki $i \in \{0, 1, \ldots, n\}$ vrijedi

$$H(x_i) = f_i, \quad DH(x_i) = f_i'.$$

1.3 Potpuni kubični splajn

Već smo spomenuli da s obzirom na zahtjeve koje postavljamo na globalnu neprekidnost prve i druge derivacije kubičnog splajna dobivamo dva različita interpolacijska splajna. U prvom slučaju, kada je prva derivacija globalno neprekidna i poklapa se u čvorovima s vrijednostima derivacije funkcije f, dobili smo Hermiteov kubični splajn. Na žalost, on je ponajviše od teorijskog značenja. U drugom slučaju, kada zahtijevamo globalnu neprekidnost druge derivacije, dobivamo splajn koji se jednostavno zove **kubični interpolacijski splajn**. Od svih splajn funkcija, kubični interpolacijski splajn je najviše korišten i najbolje izučen.

DEFINICIJA. Neka su u čvorovima $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ zadane vrijednosti f_i za $i = 0, 1, \ldots, n$. Potpuni kubični splajn je funkcija $S_3 \in C^2[a, b]$ koja zadovoljava uvjete

1. Za svaki $i \in \{0, 1, \ldots, n-1\}$ vrijedi

$$S_3(x) = a_{i0} + a_{i1}(x - x_i) + a_{i2}(x - x_i)^2 + a_{i3}(x - x_i)^3, \; x \in [x_i, x_{i+1}].$$

2. Za svaki $i \in \{0, 1, \ldots, n\}$ vrijedi $S_3(x_i) = f_i$.

Uočimo da nema interpoliranja u vrijednostima derivacije jer tražimo što veću glatkoću splajna. Dodatni uvjeti se zadaju na rubovima intervala, pa se nazivaju **rubnim uvjetima**. Ovdje navodimo neke od njih koji se često javljaju.

$(R1)$ $DS_3(a) = Df(a), \quad DS_3(b) = Df(b);$

$(R2)$ $D^2S_3(a) = 0, \quad D^2S_3(b) = 0;$

$(R3)$ $D^2S_3(a) = D^2f(a), \quad D^2S_3(b) = D^2f(b);$

$(R4)$ $DS_3(a) = Df(b), \quad D^2S_3(a) = D^2f(b).$

Prve uvjete nazivamo **potpunim** rubnim uvjetima, druge **prirodnim** rubnim uvjetima, a četvrte **periodičkim** rubnim uvjetima. Sukladno tome nazivamo i pripadne splajnove, pa razlikujemo **potpuni**, **prirodni** i **periodički** kubični splajn. Kojeg čemo kada koristiti na očigledan način ovisi o samoj funkciji koju aproksimiramo.

2 Diskretna metoda najmanjih kvadrata

Neka je funkcija f zadana na diskretnom skupu točaka $\{x_0, x_1, \ldots, x_n\}$ kojih je mnogo više nego nepoznatih parametara aproksimacijske funkcije $\varphi(x, a_0, \ldots, a_m)$. Pokušajmo funkciju φ odrediti iz uvjeta da euklidska norma (norma 2) vektora pogreške u čvorovima aproksimacije bude najmanja moguća, tj. da minimiziramo funkciju S definiranu s

$$S = \sum_{k=0}^{n} (f(x_k) - \varphi(x_k))^2 .$$

Ovu funkciju (koja je u stvari definirana kao kvadrat euklidske norme) interpretiramo kao funkciju nepoznatih parametara $a_0, \ldots, a_m$, tj.

$$S = S(a_0, \ldots, a_m) .$$

Ako je S dovoljno glatka funkcija, nužni uvjeti ekstrema su

$$\frac{\partial S}{\partial a_k} = 0, \quad k = 0, \ldots, m.$$

Pokazat ćemo da nas takav pristup vodi na tzv. **sustav normalnih jednadžbi**.

2.2 Matrična formulacija linearnog problema najmanjih kvadrata

Da bismo formirali matrični zapis linearnog problema najmanjih kvadrata moramo preimenovati nepoznanice da nam zapis ne bi previše odstupao od standardne forme. Ako je dana mreža $\{(t_1, y_1), \ldots, (t_n, y_n)\}$ i ako želimo model aproksimirati funkcijom

$$\varphi(t) = x_1\varphi_1(t) + \cdots + x_m\varphi_m(t) ,$$

onda trebamo odrediti nepoznate parametre $x_1, \ldots, x_m$ tako da vrijedi

$$\sum_{k=1}^{n} [y_k - (x_1\varphi_1(t) + \cdots + x_m\varphi_m(t))]^2 \to \min .$$

Uvedemo li oznake

$$\begin{aligned} a_{kj} &= \varphi_j(t_k), \quad k = 1, \ldots, n, \quad j = 1, \ldots, m, \\ b_k &= y_k, \quad k = 1, \ldots, n, \\ A &= [a_{kj}] \in \mathbb{R}^{n\times m}, \quad b = [b_k] \in \mathbb{R}^n, \quad x = [x_j] \in \mathbb{R}^m, \end{aligned}$$

onda nam se prethodni uvjet minimizacije svodi na traženje minimuma (po x) euklidske norme

$$\|Ax - b\|_2 .$$

TEOREM. Neka je

$$S = \{x \in \mathbb{R}^m \mid \|Ax - b\|_2 = \min\} .$$

Vrijedi: $x \in S$ ako i samo ako je ispunjena sljedeća relacija ortogonalnosti (ovdje koristimo oznaku $\mathbf{0}$ za nul-vektor)

$$A^T(b - Ax) = \mathbf{0}.$$

PROPOZICIJA. Matrica $A^T A$ je pozitivno definitna ako i samo ako su stupci matrice A linearno neovisni, tj. ako je rang matrice A jednak m.

3 Minimaks aproksimacija

Neka je f neprekidna funkcija na intervalu $[a, b] \subseteq \mathbb{R}$. Pokušajmo usporediti polinomne aproksimacije funkcije f dobivene različitim metodama i ustanoviti koja od njih daje najmanju maksimalnu pogrešku. Označimo s $\rho_n(f)$ najmanju maksimalnu pogrešku aproksimacije po svim takvim polinomima stupnja ne većeg od n, tj.

$$\rho_n(f) = \inf_{\partial p \leq n} \|f - p\|_\infty .$$

To bi značilo da ne postoji polinom stupnja ne većeg od n koji bi s manjom greškom od $\rho_n(f)$ aproksimirao funkciju f na danom intervalu. Nas, naravno, interesira za koji se polinom dostiže ta greška. Pa neka je p_n^* upravo taj polinom, tj. neka je

$$\rho_n(f) = \|f - p_n^*\|_\infty .$$

Nadalje, ako je polinom p_n^* jedinstven zanima nas kako ga možemo konstruirati. Polinom p_n^* nazivamo **minimaks aproksimacijom** funkcije f na intervalu $[a, b]$.

TEOREM. (**Čebiševljev teorem o oscilacijama grešaka**) Za danu funkciju f koja je neprekidna na $[a, b]$ i za dani $n \in \mathbb{N}_0$ postoji jedinstveni polinom p_n^* stupnja ne većeg od n za kojeg vrijedi

$$\rho_n(f) = \|f - p_n^*\|_\infty .$$

Taj polinom je karakteriziran sljedećim svojstvom: postoje barem $n+2$ točke

$$a \leq x_0 < x_1 < \cdots < x_n < x_{n+1} \leq b$$

za koje je

$$f(x_j) - p_n^*(x_j) = \sigma(-1)^j \rho_n(f), \quad j = 0, 1, \ldots, n+1,$$

pri čemu je $\sigma = \pm 1$ i ovisi samo o f i n.