UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Ivan Petković

ANALIZA PROCESNIH I RAČUNSKIH ITERACIJA PRIMENOM SAVREMENIH RAČUNARSKIH ARITMETIKA

Doktorska disertacija

NIŠ 2011

Sadržaj

Uvod

Tema disertacije – analiza procesnih i računskih iteracija primenom savremenih računarskih aritmetika – pripada oblasti računarskih nauka. U širem smislu računarske nauke se bave izučavanjem teorijskih osnova složenih izračunavanja i obradom informacija, kao i praktičnim tehnikama za njihovu implementaciju i primenu u računarskim sistemima. Osnovni cilj je razvijanje algoritamskih procesa koji kreiraju, opisuju, realiziju i transformišu informacije ili modeluju kompleksne sisteme najrazličitijih tipova u cilju analize ili rešavanja postavljenog problema. Po svojoj sušini računarske nauke su bliže primenjenoj matematici nego drugim naučnim disciplinama, ali se u današnje vreme interdisciplinarnih nauka nemonovno prepliću sa svim oblastima koje su, u većoj ili manjoj meri, povezane sa računarskim sistemima, informacionim tehnologijama, softverskim inženjerstvom, programskim jezicima, itd.

Savremeni razvoj tehnologije doveo je naglog razvoja ne samo inženjerskih disciplina, već i fizike, hemije, informacionih tehnologija, ekonomije, medicine, kriptologije, društveno-humanističkim nauka, itd. Rešavanje nekih suptilnih i zahtevnih problema koji su se javili u ovim oblastima nije bilo moguće bez daljeg napretka softvera i hardvera u računarskim sistemima. To se naročito odnosi na: povećanje kapaciteta memorija, brzine rada procesora (tj. brzine izvršenja operacija), preciznosti računarske aritmetike i kontrole rezultata. Problem memorijskog prostora i brzine rada procesora u dobroj meri je rešen ili se rešava putem hardvera i raspoložive tehnologije uz pomoć sve sofisticiranijih algoritama. Preciznost rezultata i njihova kontrola postižu se softverski uvođenjem aritmetike višestruke preciznosti i intervalne aritmetike, koje se prema Brent-Zimermanovoj terminologiji [19] nazivaju savremenim (ili naprednim) računarskim aritmetikama.

Savremene računarske aritmetike omogućile su razvoj algoritama za rešavanje složenih problema kao što su brzina izračunavanja i kontrola tačnosti rezultata, optimizacija karakteristika najrazličitijih tipova procesa uključujući projektovanje softvera i hardvera, optimizacija transporta, modeliranje inženjerskih projekata, realizacija numeričkih algoritama, itd. Ovo su veoma važne i aktuelne istraživačke teme sa stanovišta primene na kojima se intenzivno radi u svetu. Veliki deo pomenutih istraživačkih tema diskutovan je u nedavno publikovanim knjigama [19], [82] i [103], kao i u brojnim radovima citiranim u ovim knjigama.

Teme razmatrane u ovoj doktorskoj disertaciji odnose se na neke od nabrojanih procesa i to na procese iterativnog karaktera. Ovo su procesi kod kojih se globalni proces, zbog svoje složenosti, modelira i rasčlanjuje na više potprocesa, tzv. taskova, a zatim se odgovarajući model analizira korak po korak, tj. iterativno. Istraživanja u disertaciji bila su usmerena na sledeća dva procesa iterativnog karaktera:

(I) Analiza matričnih modela procesnih iteracija;

(II) Numerički algoritmi iterativnog tipa i računska efikasnost.

Cilj doktorske disertacije je realizacija pomenutih tema pomoću aritmetike višestruke preciznosti i intervalne aritmetike, pri čemu su glavni pravci istraživanja bili usmereni na sledeće oblasti:

– Izrada softvera za dijametarsku formu aritmetike diskova koja daje optimalnu inkluziju rezultata u kompleksnoj aritmetici.

– Matrični modeli procesnih iteracija i njihova analiza.

– Računska efikasnost numeričkih algoritama iterativnog tipa u režimu dinamičke preciznosti korišćene aritmetike višestruke preciznosti.

U disertaciji je izloženo više originalnih rezultata koji su nastali kao rezultat trogodišnjeg rada na pomenutim temama. Neki od dobijenih rezultata su do sada publikovani u internacionalnim časopisima za računarske nauke i primenjenu matematiku (*J. Computational and Applied Mathematics, Applied Mathematics and Computation, Reliable Computing, International J. Computer Mathematics*) ili su saopšteni na međunarodnim konferencijama ICCAM-2006, Luven (Belgija), SCAN-2008, El Paso (SAD) i IEEE International Conference on Electronics and Information Engineering, 2010, Kjoto (Japan)) i štampani u odgovarajućim zbornicima ili časopisima.

Disertacija se sastoji iz sledećih poglavlja:

Uvod;

1. Savremene računarske aritmetike;
2. Realna intervalna aritmetika i intervalna D-aritmetika;
3. Matrični modeli procesnih iteracija;
4. Računska efikasnost numeričkih algoritama;
5. Zaključak;

Literatura.

Prvo poglavlje je uvodnog karaktera i u njemu su uglavnom predstavljene karaktersitike novog standarda IEEE 754-2008 aritmetike sa pokretnom tačkom, sa posebnim osvrtom na pravilno izvršavanje različitih modova zaokruživanja - glavnog preduslova tačnog izračunavanja. Ovo poglavlje sadrži pregled aktuelnih programskih paketa (*Mathematica, Maple,...*) koji rade sa aritmetikom višestruke preciznosti i izvršavaju vrlo složena izračunavanja i simulacije.

U drugom poglavlju glavni deo istraživanja je posvećen intervalnoj aritmetici i intervalnim matricama neophodnim za analizu matričnih modela procesnih iteracija, razmatranim u Poglavlju 3. Razlog za upotrebu intervalne aritmetike je mogućnost kontrole tačnosti rezultata i rad sa približnim podacima za koje se jedino zna da leže u nekim intervalima. U ovu svrhu realizovan je softver u C #, koji omogućuje rad sa dijametarskom aritmetikom i izračunavanje elementarnih kompleksnih funkcija. U Poglavlju 2 dat je i kratak opis softvera INTLAB za intervalnu aritmetiku (realnu i kompleksnu sa centriranim diskovima). Ovaj softver je takodje korišćen u 3. poglavlju.

Pri rešavanju globalnih problema različitog tipa (projektovanje i razvoj novih proizvoda, tehnička rešenja, razvoj hardvera ili softvera, itd.) koji se sastoje od velikog broja međusobno zavisnih taskova, veoma dobri rezultati postignuti su korišćenjem modela koji su krajem dvadesetog veka razvili S. Epinger i R. Smit, profesori sa Masačusetskog instituta za tehnologiju (MIT, Boston). Njihov pristup je iterativne prirode, a karakteristike i međusobna

zavisnost taskova opisuju se nenegativnom, nerazloživom matricom A strukture procesne iteracije. Jedna od glavnih tema u disertaciji posvećena je poboljšanju karakteristika razvojno-projektnog procesa zasnovanog na Epinger-Smitovom modelu. U disertaciji je razrađena metodologija za utvrđivanje optimalnog redosleda taskova razvojnog moda koji se posmatra. Rangiranje se vrši preko koordinata sopstvenog vektora koji odgovara spektralnom radijusu $\rho(A)$ matrice strukture procesne iteracije. Ovaj pristup je elegantniji i precezniji od Epinger-Smitove karakterizacije koja se oslanja na grubu kvalitativnu procenu sopstvenih vrednosti matrice $(I - A)^{-1}$.

Druga važna tema u okviru 3. poglavlja odnosi se na originalni pristup u analizi modela procesne iteracije koji koristi intervalnu linearnu algebru i radi sa realnim intervalima i intervalnim matricama umesto realnih brojeva i realnih matrica. Na ovaj način a) postiže se veći stepen slobode u kvantitativnoj proceni taskova, b) dopušteno je prisustvo (do izvesne mere, definisano dužinom intervala) „neodredjenih" veličina.

U 4. poglavlju disertacije razmatrana je efikasnost algoritama iterativne prirode, pre svega iterativnih procesa za rešavanje nelinearnih i algebarskih jednačina, sa posebnim osvrtom na upotrebu aritmetike promenljive (dinamičke) preciznosti. Pri ovoj analizi koristi se računska cena $C(b)$ pojedinih osnovnih aritmetičkih operacija u funkciji broja b upotrebljenih bitova koji definišu preciznost aritmetike višestruke preciznosti. U disertaciji je predloženo korišćenje dinamičke preciznosti (u smislu promenljive preciznosti upotrebljene aritmetike) u različitim iterativnim koracima u cilju povećanja računske efikasnosti numeričkih algoritama za rešavanje nelinearnih jednačina. Za opisani slučaj u disertaciji se predlaže procena računske efikasnosti iterativnog algoritma preko indeksa efikasnosti u režimu dinamičke preciznosti. Nova formula za indeks efikasnosti iskorišćena je za upoređenje i rangiranje postojećih iterativnih metoda za istovremeno nalaženje svih nula polinoma.

Spisak direktno korišćene ili citirane literature, koji sadrži 159 referenci, dat na na kraju disertacije.

U disertaciji je kraj definicija, primera i napomena označavan simbolom ▲.

*

* *

Koristim priliku da se zahvalim svima koji su na bilo koji način doprineli izradi ove disertacije.

Posebno se zahvaljujem svom mentoru dr Mileni Stanković, redovnom profesoru Elektronskog fakulteta u Nišu, koja mi je pružila dragocenu pomoć ne samo pri izradi disertacije nego i kao rukovodilac globalnog projekta pri realizaciji dela projekta o matričnim modelima procesnih iteracija. Takođe se zahvaljujem i dr Stivenu Epingeru, profesoru Masačusetskog instituta za tehnologiju (MIT) u Bostonu, na delotvornim diskusijama o matričnim modelima procesnih iteracija, dr Zigfridu Rumpu, profesoru Tehničkog univerziteta u Hamburgu na savetima pri korišćenju softvera INTLAB čiji je on kreator i dr Polu Zimermanu sa Nacionalnog instituta za istraživanja u informatici i automatici (INRIA) u Nansiju (Francuska) na korisnim sugestijama pri primeni aritmetike višestruke preciznosti.

Oznake

U disertaciji su korišćene sledeće oznake:

- $\mathbb{R}$ – skup realnih brojeva
- $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$ – prošireni skup realnih brojeva
- $\mathbb{C}$ – skup kompleksnih brojeva
- APT – aritmetika sa pokretnom tačkom
- AVP – aritmetika višestruke preciznosti
- β – osnova brojnog sistema u APT
- $e_{\min}, e_{\max}$ – minimalni i maksimalni eksponent PT brojeva
- PT broj x – broj u aritmetici sa pokretnom tačkom mantise m, $x = (-1)^s \cdot m \cdot \beta^e,\ s \in \{0, 1\},\ e_{\min} \leq e \leq e_{\max}$
- p-preciznost APT – broj značajnih cifara mantise $m = \sum_{i=1}^{p} d_i \beta^{-i}$, $d_i \in \{0, 1, \ldots, \beta - 1\}$ PT broja
- $S(\beta, p, e_{\min}, e_{\max})$ – PT sistem sa pokretnom tačkom osnove β, preciznosti p i opsega eksponenta $[e_{\min}, e_{\max}]$
- $\overline{S}(\beta, p, e_{\min}, e_{\max}) = S(\beta, p, e_{\min}, e_{\max}) \cup \{-\infty\} \cup \{+\infty\}$ – prošireni PT sistem sa pokretnom tačkom
- $\square$ – operator zaokruživanja, $\square\, x$ je PT broj nastao zaokruživanjem realnog broja x
- $\triangle x$ – uzlazno zaokruživanje broja x (na najbliži veći PT broj)

- $\triangledown x$ – silazno zaokruživanje broja x (na najbliži manji PT broj)
- $\bigwedge_{a\in M} P(a)$ – osobina $P(a)$ je tačna za sve elemente $a \in M$
- $RD(x) := \max\{y \in \overline{S} \ : \ y \leq x\}$, silazno direktno zaokruživanje
- $RG(x) := \min\{y \in \overline{S} \ : \ y \geq x\}$, uzlazno direktno zaokruživanje
- $RN(x) := \begin{cases} \triangledown x & \text{ako je } x \geq 0 \\ \triangle x & \text{ako je } x < 0 \end{cases}$, zaokruživanje prema nuli
- $RB(x) := y, \ y \in \overline{S}, \ |y - x| \to \min$ zaokruživanje na najbliži PT broj
- SPD – *spojen pomnoži-i-dodaj* operator ($SPD(a, x, b) = a \cdot x + b$)
- `NaN` (Not a Number) – nedefinisan rezultat u APT
- $\mathrm{exp1m}(x) = \exp(x) - 1$
- $f = O(g)$ ili $f \sim Cg$ – f i g su veličine istog reda, tj. $\dfrac{f}{g} \to C$, $C \neq 0$ je konstanta
- $[a, b]$ - realan interval sa donjom granicom a i gornjom granicom b, $a \leq b$
- $I(\mathbb{R})$ - skup realnih intervala
- $Z = \{c; r\} = (x, y, r)$ – disk (kružni kompleksan interval) sa centrom u tački $c = x + iy$ i poluprečnikom r
- $K(\mathbb{C})$ – skup diskova u kompleksnoj ravni
- Z^{-1} – tačna inverzija diska Z
- Z^{I_c} – centrirana inverzija diska Z
- $I_c(f(Z))$ – centralna forma zatvorene oblasti $f(Z) = \{f(z) : z \in Z\}$
- $I_d(f(Z))$ – dijametarska forma zatvorene oblasti $f(Z) = \{f(z) : z \in Z\}$
- $A_1 + iA_2, \ A_1, A_2 \in I(\mathbb{R})$ – kompleksan interval u obliku pravougaonika sa strana paralelnim koordinatnim osama
- $\boldsymbol{A} = (A_{ij})$ intervalna matrica čiji su elementi realni intervali A_{ij}

- $\dot{\boldsymbol{B}} = (b_{ij})$ - „tačkasta" matrica čiji su elementi realni brojevi b_{ij}
- $\|A\|$ – norma realne matrice A
- $\|\boldsymbol{A}\| = \max\limits_{\dot{\boldsymbol{A}} \in \boldsymbol{A}} \|\dot{\boldsymbol{A}}\|$ – norma intervalne matrice $\boldsymbol{A}$
- $\boldsymbol{A}^k = \boldsymbol{A}^{k-1} \cdot \boldsymbol{A}$ – k-ti stepen intervalne matrice $\boldsymbol{A}$, $\boldsymbol{A}^0 = \dot{\boldsymbol{I}}$
- $M_{nm}(\mathbb{R})$ – skup tačkastih matrica koje imaju n vrsta i m kolona
- $\lambda_1, \ldots, \lambda_n$ – sopstvene vrednosti regularne kvadratne matrice $n \times n$
- $\rho(A) = \max\limits_{1 \le i \le n} |\lambda_i|$ – spektralni radijus matrice A
- $f^{(k)}(x_k)$ - k-ti izvod funkcije f u tački x_k
- $P(z) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ – algebarski polinom n-tog stepena
- q – red konvergencije iterativnog metoda $x_{k+1} = \phi(x_k)$ definisan pomoću $\dfrac{|\phi(x_k) - \alpha|}{|x_k - \alpha|^q} \to C \neq 0$, α je nula funkcije i C konstanta
- θ – broj novih informacija po iteraciji
- $E_1(q, \theta) = \dfrac{q}{\theta}$ – indeks efikasnosti po Ostrovskom
- $E_2(q, \theta) = q^{1/\theta}$ – indeks efikasnosti po Traubu
- $E_3(q, \theta) = \dfrac{\log q}{\theta}$ – indeks efikasnosti po Brentu
- $w_s = w_s(b) = b$ – težinska funkcija za sabiranje i oduzimanje
- $w_m = w_m(b) = b \log(b) \log\log(b)$ – težinska funkcija za množenje
- $w_d = w_d(b) = 3.5b \log(b) \log\log(b)$ – težinska funkcija za deljenje
- W_s, W_m, W_d – težinske funkcije u režimu dinamičke preciznosti
- $t_b(X)$ - vreme izvršenja operacije X u aritmetici preciznosti b bitova

1

Savremene računarske aritmetike

Ovo poglavlje je uvodnog karaktera i u njemu su uglavnom predstavljene karakteristike novog standarda IEEE 754-2008 aritmetike sa pokretnom tačkom, usvojenog 2008. godine. Posebno je opisano jedno od osnovnih poboljšanja ovog standarda, strogo definisano *pravilno* zaokruživanje brojeva, veoma važna karakteristika savremene računarske aritmetike. Pravilno izvršavanje različitih modova zaokruživanja je od izuzetne važnosti pri korišćenju aritmetike višestruke preciznosti i realizaciji tzv. intervalne aritmetike. S obzirom da se u disertaciji koristi aritmetika višestruke preciznosti, u prvom poglavlju je ukratko dat pregled aktuelnih programskih paketa koji rade sa aritmetikom višestruke preciznosti, izvršavaju vrlo složena izračunavanja i simulacije, realizuju mnogobrojne algoritme iz oblasti numeričke analize sa skoro proizvoljno velikom tačnošću i grafički predstavljaju dobijene rezultate.

1.1 Aritmetika sa pokretnom tačkom

Aritmetika sa pokretnom tačkom (skraćeno APT) počela je da se koristi ranih četrdesetih i pedesetih godina prošlog veka. APT je najrasprostranjeniji način za aproksimaciju realnih brojeva za numerička izračunavanja na modernim računarima. Napori velikog broja naučnika u oblasti kompjuterskih nauka i matematike rezultirali su uvođenjem APT sa IEEE 754-1985 standardom za osnovu 2. Aritmetika sa IEEE 754-1985 standardom imala je više revizija i poboljšanja tokom godina, što je dovelo do nove verzije

IEEE 754-2008 standarda u junu 2008. Može se reći da ova APT sa 2008-standardom na veoma efikasan način simulira beskonačan kontinualan skup (realnih brojeva) pomoću konačnog diskretnog skupa („mašinskih brojeva"), pri čemu je učinjen vrlo zadovoljavajući kompromis između brzine, tačnosti, dinamičkog opsega, jednostavnog korišćenja i implemetacije, zauzeća memorijskog prostora, itd., što su retko kada kompatibilni zahtevi.

Predstavljanje brojeva u aritmetici sa pokretnom tačkom

Opšte prihvaćeno predstavljanje broja x u aritmetici sa pokretnom tačkom (APT) sa bazom $\beta > 1$ je

$$x = (-1)^s \cdot m \cdot \beta^e, \tag{1.1}$$

gde je $(-1)^s$ ($s \in \{0, 1\}$) *znak*, $m = \sum_{i=1}^{p} d_i \beta^{-i}$, $d_i \in \{0, 1, \ldots, \beta - 1\}$ je *mantisa* i ceo broj e je *eksponent* broja x koji se nalazi u intervalu $[e_{\min}, e_{\max}]$. Prirodan broj p u sumi definiše *preciznost* broja x, što znači da mantisa m sadrži najviše p značajnih cifara u bazi β. Skup $S = S(\beta, p, e_{min}, e_{\max})$ se zove *sistem sa pokretnom tačkom* (PT sistem).

Termini *tačnost* i *preciznost* se često nepravilno upotrebljavaju ili im se zamenjuje značenje te je zbog toga potrebno napraviti jasnu razliku između njih. *Tačnost* se odnosi na apsolutnu ili relativnu grešku približnog rešenja nekog problema ili približnog rezultata nekog izračunavanja. *Preciznost* je definisana brojem značajnih (tačnih) cifara sa kojim se osnovne aritmetičke operacije $+, -, *, /$ izvršavaju. Na primer, tačnost može biti mnogo lošija nego preciznost upotrebljene aritmetike pri rešavanju sistema linearnih jednačina. Važno je naglasiti da se tražena tačnost ponekad ne može postići povećanjem preciznosti (zabluda naročito prisutna u inženjerskim disciplinama) jer je struktura rešavanog problema takva da dolazi do efekata koji ne omogućavaju povećanje preciznosti (na primer, gubitak značajnih cifara). Ilustrativan je Primer 1.2 gde se tačan rezultat (pa čak ni relativno dobra aproksimacija) ne može dobiti čak ni sa znatno uvećanom preciznošću upotrebljene artimetike. Više detalja dato je u knjizi [67, str. 7].

Drugi način da se izrazi PT broj x je pomoću uređenog para (M, e) tako da je

$$x = M \cdot \beta^{e-p+1}, \tag{1.2}$$

gde je M ceo broj manji ili jednak $\beta^p - 1$ dok je e ceo broj takav da je $\varepsilon_{\min} \leq e \leq e_{\max}$. Kao i ranije, e je eksponent, a p preciznost PT sistema. Veza između m i M data je sa $m = |M| \cdot \beta^{1-p}$.

Predstavljanje PT brojeva u obliku (M, e) nije uvek jedinstveno. Na primer, ako je $\beta = 10$ i $p = 3$, tada se broj 17 može predstaviti ili kao 17×10^0 ili kao 170×10^{-1}, jer oba broja 17 i 170 su manja od $\beta^p = 1000$. Skup ovih ekvivalentnih reprezentacija zove se *kohorta*. Da bi se dobilo jedinstveno predstavljanje PT brojeva vrši se *normalizacija* birajući onaj oblik za koji je eksponent minimalan (ali i dalje jednak ili veći od e_{min}). Ovako predstavljeni brojevi nazivaju se *normalizovanim brojevima*. Da bi PT broj bio normalizovan, potrebno je da važi $\beta^{p-1} \leq |M| \leq \beta^p$. Na primer, broj $x = -3.25$ može se predstaviti u dekadnom sistemu preciznosti $p = 4$ kao

$$x = (-1)^1 \cdot 0.325 \cdot 10^1 \quad \text{pomoću (1.1), ili} \quad x = -325 \cdot 10^{-2} \quad \text{pomoću (1.2).}$$

Normalizovano predstavljanje PT brojeva omogućuje jednostavnije izraze za granice greške, jednostavniju implementaciju i uštedu od jednog bita u sistemu sa osnovom 2.

Važan specijalan slučaj je $m = 0$ koji predstavlja nulu. U ovom slučaju znak s i eksponent e nisu važni i mogu poslužiti za kodiranje drugih informacija.

Za $m \neq 0$ uvode se još nekoliko pojmova:

- Ako je $\beta^{-1} \leq m < 1$, tada je $\beta^{e-1} \leq |x| < \beta^e$. U ovom slučaju m je celobrojni faktor od β^{-p} a broj β^{e-p} određuje *jedinicu na poslednjem mestu* broja x i označava se sa $\text{ulp}(x)$ (ulp dolazi od *unit in the last place*). Jedan *ulp* normalizovanog PT broja $y = \pm\beta^e \times .d_1 d_2 \ldots d_p$ je

  $$\text{ulp}(y) = \beta^e \times .00 \ldots 01 = \beta^{e-p}.$$

  Na primer, $x = 3.1416$ sa bazom $\beta = 10$ se kodira pomoću $m = 0.31416$ i $e = 1$.

- Ako je $1 \leq m < \beta$, tada je $\beta^e \leq |x| < \beta^{e+1}$ i $\text{ulp}(x) = \beta^{e+1-p}$. S bazom $\beta = 10$ broj $x = 3.1416$ se kodira pomoću $m = 3.1416$ i $e = 0$.

- Veličina $u = \frac{1}{2}\beta^{1-p}$ se zove *jedinica zaokruživanja* i često se koristi u analizi greške zaokruživanja radeći sa PT sistemom S.

Svaki element skupa S je racionalan broj sa jedinstvenom reprezentacijom. Broj elemenata skupa $S(\beta, p, e_{min}, e_{\max})$ je $2(\beta - 1)\beta^{p-1}(e_{\max} - e_{\min} +$

$1) + 1$. Najveći PT broj u $S(\beta, p, e_{min}, e_{\max})$ je

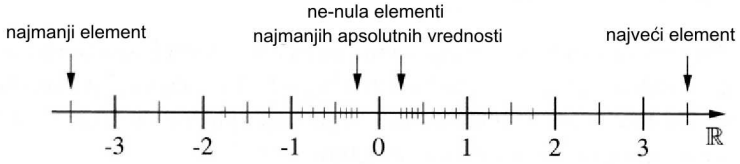
$$B := +0.\underbrace{(\beta-1)(\beta-1)\cdot(\beta-1)}_{p-cifara}\beta^{e_{\max}}.$$

Najmanji broj u S je $-B$. Najmanji nenula PT brojevi u S sa najmanjom apsolutnom vrednošću su

$$-0.100\cdots0\cdot\beta^{e_{\min}}, \quad +0.100\cdots0\cdot\beta^{e_{\min}}.$$

PT brojevi u S nisu ravnomerno raspoređeni između brojeva datih sa (1.1) na intervalu $[-B, +B]$. Gustina opada sa porastom eksponenta. Ilustracije radi, koristimo primer PT sistema od 33 elementa koji odgovara izboru $\beta = 2,\ p = 3,\ e_{\min} = -1,\ e_{\max} = 2$. Slika 1.1 prikazuje PT sistem $S(2, 3, -1, 2)$. Postoji relativno velike praznine oko nule koje ne sadrže PT brojeve.

e	2^e	m_1	m_2	m_3	m_4
-1	$\frac{1}{2}$	0.100	0.101	0.110	0.111
0	1	0.100	0.101	0.110	0.111
1	2	0.100	0.101	0.110	0.111
2	4	0.100	0.101	0.110	0.111

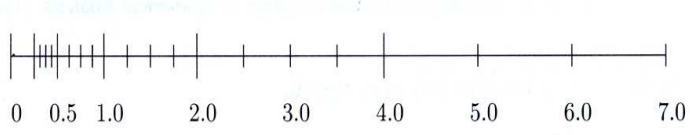


Slika 1.1 Karakteristični razmaci PT sistema

Kao drugi primer posmatrajmo PT sistem $S(2, 3, -1, 3)$. Nenegativni PT brojevi su

$$0,\ 0.25,\ 0.3125,\ 0.3750,\ 0.4375,\ 0.5,\ 0.625,\ 0.750,\ 0.875,$$
$$1.0,\ 1.25,\ 1.50,\ 1.75,\ 2.0,\ 2.5,\ 3.0,\ 3.5,\ 4.0,\ 5.0,\ 6.0,\ 7.0.$$

Ovi brojevu su predstavljeni na slici 1.2.



Slika 1.2 Brojevi PT sistema $S(2, 3, -1, 3)$

Razmak između PT brojeva može se pogodno okarakteristati pomoću termina *mašinske preciznosti* ϵ_M koja predstavlja rastojanje od 1.0 do sledećeg većeg PT broja. Očigledno je $\epsilon_M = \beta^{1-p}$, i ovo je razmak PT brojeva na intervalu $[1, \beta]$. Razmak za proizvoljan PT broj $x \in S$ dat je sledećom teoremom.

Teorema 1.1. *Razmak između normalizovanog PT broja x i susednog normalizovanog broja iznosi najmanje $\beta^{-1}\epsilon_M|x|$, a najviše $\epsilon_M|x|$ (izuzev ako je x ili susedni broj nula).* ▲

Dokaz se može naći u [67].

Zbog istorijskog značaja dajemo tabelu sa APT parametrima za neke računare i standardne aritmetike.

Računar i preciznost aritmetike	β	p	$e_{\min}$	$e_{\max}$	u
Cray-1, jednostruka	2	48	−8192	8191	4×10^{-15}
Cray-1, dvostruka	2	96	−8192	8191	1×10^{-29}
DEC VAX G format, dvostruka	2	53	−1023	1023	1×10^{-16}
DEC VAX D format, dvostruka	2	56	−127	127	1×10^{-17}
HP 28 i 48G kalkulatori	10	12	−499	499	5×10^{-12}
IBM 3090, jednostruka	16	6	−64	63	5×10^{-7}
IBM 3090, dvostruka	16	14	−64	63	1×10^{-16}
IBM 3090, proširena	16	28	−64	63	2×10^{-33}
IEEE, jednostruka	2	24	−125	128	6×10^{-8}
IEEE, dvostruka	2	53	−1021	1024	1×10^{-16}
IEEE, proširena	2	64	−16381	16384	5×10^{-20}

Tabela 1.1 Parametri PT aritmetike

PT sistem S može se proširiti uključujući *subnormalizovane brojeve* (takođe poznati i pod imenom *denormalizovani brojevi*) koji u notaciji (1.1) imaju oblik

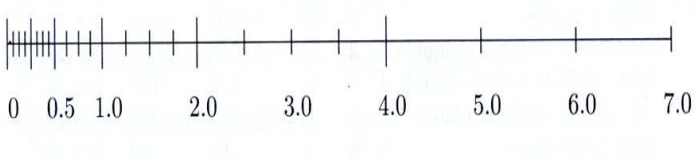
$$x = \pm m \cdot \beta^{e_{\min}}, \quad 0 < m < 1.$$

Subnormalizovani brojevi imaju manje značajnih cifara nego normalizovani PT brojevi. Najmanji pozitivan normalizovan PT broj je $\lambda = \beta^{e_{\min}-1}$ dok je najmanji pozitivan subnormalizovan broj $\mu = \beta^{e_{\min}-p} = \lambda\epsilon_M$. Subnormalizovani brojevi se uvode da bi popunili prazninu između 0 i λ i podjednako su raspoređeni na istom razmaku kao PT brojevi između λ i $\beta\lambda$, tj.

$\lambda\epsilon_M = \beta^{e_{\min}-p}$. Na primer, za sistem $S(2,3,-1,3)$ koji je ranije pomenut imamo $\lambda = 2^{-2}$ i $\mu = 2^{-4}$ tako da su subnormalizovani brojevi za ovaj sistem

$$0.0625,\ 0.125,\ 0.1875.$$

Uzimajući ove brojeve i PT brojeve sa slike 1.2, dobija se kompletan brojni sistem prikazan na slici 1.3.



Slika 1.3 Brojevi PT sistema $S(2,3,-1,3)$ i dodati subnormalizovani brojevi

Većina PT implementacija koriste bazu $\beta = 2$ ili stepen od 2, jer je ovakav način pogodan i efikasan na binarnim računarima. Za bazu β koja nije stepen broja 2 moguća su sledeća dva izbora:

- smestiti mantisu u bazi β, ili opštije, u bazi β^k za neko $k \geq 1$. Svaka cifra u bazi β^k zahteva $\lceil k \log_2 \beta \rceil$ bita. Za $\beta = 10$ i $k = 1$ ovo je „binarno kodirana decimalna cifra" ili BCD kodiranje (binary coded decimal): svaka decimalna cifra je predstavljena pomoću 4 bita, pri čemu se na memoriji gubi oko 17% (jer je $\log_2(10)/4 \approx 0.83$). Nešto kompaktniji izbor je baza 10^3, gde su 3 decimalne cifre smeštene pomoću 10 bita umesto 12 bita sa BCD formatom. Pri ovome se na memoriji gubi samo 0.34% (jer je $\log_2(1000)/10 \approx 0.9966$).

- smestiti mantisu binarno. Ova ideja je korišćena u Intelovom binarno-celobrojnom kodiranju decimalne cifre ili BID kodiranju (binary-integer decimal) u novom standardu IEEE 754-2008.

Nedostatak binarnog kodiranja je u tome što prilikom sabiranja dva broja proizvoljne preciznosti, nije lako detektovati da li je (celobrojna) mantisa premašila maksimalnu vrednost $\beta^p - 1$.

Razmatrajući opseg vrednosti eksponenta, mogle bi se uzeti celobrojne vrednosti proizvoljno velike preciznosti. Ovo bi bila značajna prednost jer se prekoračenja opsega ne bi javljala. Međutim, u najvećem broju primena, izbor eksponenta kodiranog sa 32 bita je više nego dovoljno. Ovo nam

omogućuje predstavljanje vrednosti približno do $10^{646\,456\,993}$ za $\beta = 2$. Rezultati koji su veći od ove granice najverovatnije ukazuju na grešku u algoritmu ili softverskoj/hardverskoj implementaciji. Osim toga, korišćenje proizvoljno velike preciznosti rešilo bi samo neke ekstremalne probleme koji se retko javljaju u praksi, ali bi se kod najčešće primenjivanih algoritama smanjila računarska efikasnost i usporila njihova realizacija.

Zbog navedenih okolnosti, u praksi eksponent e ima limitiran opseg $e_{\min} \leq e \leq e_{\max}$. Kaže se da neki broj ima PT *reprezentaciju* ako se može predstaviti u obliku $(-1)^s \cdot m \cdot \beta^e$, gde je $e_{\min} \leq e \leq e_{\max}$. S obzirom da se najčešće uzima da je $\beta^{-1} \leq m < 1$, najmanji pozitivni PT broj koji se može predstaviti je $\beta^{e_{\min}-1}$, dok je najveći $\beta^{e_{\max}}(1-\beta^{-p})$. Druge konvencije daju drugačije opsege eksponenta. Na primer, u slučaju dvostruke preciznosti, zvane `binary64` u IEEE-754-2008 standardu, imamo da je $e_{\min} = -1022$, $e_{\max} = 1023$ za mantisu u intervalu $[1, 2)$ i $e_{\min} = -1074$, $e_{\max} = 971$ za celobrojnu mantisu u intervalu $[2^{52}, 2^{53})$. U opštem slučaju, vrednosti $e_{\min}$ i $e_{\max}$ se biraju da budu simetrične: $e_{\min} \approx -e_{\max}$. Jedan od razloga je da vrednosti recipročne funkcije $1/x$ upadnu u opseg PT brojeva. U IEEE 754-2008 standardu bira se $e_{\min} = -e_{\max}$ ili $e_{\min} = 1 - e_{\max}$. Na ovaj način, ako je x normalizovan broj, tada $1/x$ nikad ne daje potkoračenje, videti [103, str. 26] za detalje.

Napomenimo da je IEEE aritmetika jedan zatvoren sistem: svaka aritmetička operacija daje rezultat, bilo da je matematički očekivan ili ne, ili poruke u izuzetnim situacijama koje su navedene u tabeli 1.2. Radeći sa ograničenim opsegom eksponenta, u nekim slučajevima su neophodne poruke ili upozorenja koja se odnose na veoma velike i veoma male vrednosti. Vrlo male vrednosti se predstavljaju nulom, koja je specijalan broj u smislu da je njegova mantisa $m = 0$. Za vrlo velike vrednosti prirodno je da se uvedu dve specijalne vrednosti $-\infty$ i $+\infty$, koje kodiraju velike brojeve koji se ne mogu predstaviti. S obzirom na ove vrednosti, takođe je prirodno imati dve nule, -0 i $+0$, na primer $1/(-\infty) = -0$ i $1/(+\infty) = +0$, kao što je slučaj sa IEEE 754 standardom. Druga mogućnost je jedna beskonačnost ∞ i jedna nula 0, izostavljajući znak u oba slučaja.

Dodatna specijalna vrednost je *Not a Number* (NaN), koja predstavlja ili neinicijalizovanu vrednost ili rezultat *ilegalne* operacije kao što je $\sqrt{-1}$ (u realnoj aritmetici) ili $(+\infty) - (+\infty)$, $0 \times \infty$, itd. U ranijim softverskim aplikacijama pri pojavi prekoračenja opsega ili deljenja nulom, dalje izvršavanje programa se prekidalo. Međutim, generalna strategija IEEE standarda je da u ovim slučajevima pošalje poruku o ilegalnim operacijama i nastavi sa

izračunavanjem, mada to ponekad vodi u „klopku." Na ovaj način omogućeno je programeru da odloži neke testove i odluke za neki pogodniji moment kasnije u programu.

Poruke upozorenja	Primer	Izlazni rezultat
Ilegalna operacija	$0/0,\ 0\times\infty,\ \sqrt{-1},\ \log(+0)$	NaN (Not a Number)
Prekoračenje	Eksponent veći od $e_{\max}$	$\pm\infty$
Deljenje nulom	Konačan broj/0	$\pm\infty$
Potkoračenje	Eksponent manji od $e_{\min}$	Subnormalizovani brojevi
Netačnost	Uvek kada je $RB(x*y)\neq x*y$	*Pravilno* zaokružen broj*

* Pravilno zaokruživanje je dato Definicijom 1.1.

Tabela 1.2 Poruke upozorenja u IEEE aritmetici i odgovarajući izlazni rezultati

Kodiranje PT broja $x=(-1)^s\cdot m\cdot\beta^e$ je postupak kojim se vrednosti $s,\ m,\ e$ smeštaju u računar. Za normalizovane brojeve u APT sa bazom 2, tj., $2^{p-1}\le m<2^p$ gde p označava preciznost, *vodeća cifra* mantise je uvek 1, što znači da ona ne mora da bude kodirana u memoriju (podrazumeva se implicitno). Na ovaj način se dobija jedan bit više. Ovaj, tzv. *implicitni* ili *skriveni vodeći bit*, se koristi u IEEE 754 standardu. Na primer, format sa dvostrukom preciznošću ima bit za znak, polje eksponenta zauzima 11 bita a mantisa 53 bita, pri čemu se memoriše samo 52 bita (ušteda na vodećem bitu), što ukupno čini 64 bita.

znak	eksponent	mantisa
(1 bit)	(11 bita)	(52 bita, plus implicitni vodeći bit)

U aritmetici visoke preciznosti gde se koristi format od 64 ili više bitova ušteda od jednog bita nije od suštinskog značaja, tako da je pogodnije vršiti kodiranje cele mantise. Pomenimo, takođe, da implicitni bit nije moguć u bazi $\beta>2$ jer za normalan broj prva značajna cifra mantise može uzeti vrednosti od 1 do $\beta-1$.

Navedimo jedan interesantan rezultat koji se odnosu na distribuciju vodeće cifre mantise u PT sistemu sa osnovom β. Još 1938. godine Benford je primetio da se vodeće cifre mantisa u logaritamskim tablicama *ne javljaju ravnomerno*. Brent [11] je pokazao da je njihova distribucija logaritamska i približno data formulom

$$\log_\beta\Big(1+\frac{1}{k}\Big),$$

gde je β baza PT sistema a $k \in \{1, \ldots, \beta - 1\}$ predstavlja vodeći cifru. Pojavljivanje cifara $1, \ldots, 9$ kao vodećih cifara mantise u PT sistemu sa osnovom 10 dato je u tabeli 1.3 procentualno.

1	2	3	4	5	6	7	8	9
30.1	17.6	12.5	9.7	7.9	6.7	5.8	5.1	4.6

Tabela 1.3 Pojavljivanje cifara $1, \ldots, 9$ kao vodećih cifara matice (u %)

1.2 Zaokruživanje – preduslov tačnog izračunavanja

Postoje tri glavna izvora grešaka pri numeričkom izračunavanju: zaokruživanje, netačnost podataka i odsecanje. Greške usled zaokruživanja nastaju kao posledica rada u aritmetici konačne preciznosti. Greške usled netačnosti ulaznih podataka se uvek javljaju pri rešavanju praktičnih problema. Uglavnom nastaju pri merenju fizičkih veličina, pri smeštaju podataka u računarima (greške zaokruživanja), ili kad su podaci dobijeni iz prethodnog izračunavanja.

Greške odsecanja ili greške diskretizacije se često sreću u u numeričkoj analizi. Mnogi standardni numerički metodi (na primer, trapezno pravilo za približnu integraciju, Ojlerov metod za rešavanje diferencijalne jednačine, Njutnov metod za nelinearne jednačine) mogu se izvesti uzimajući konačno mnogo članova Tejlorovog reda. Izostavljeni (zanemareni) članovi dovode do greške odsecanja.

Jedan mogući način da se izbegnu greške usled zaokruživanja i odsecanja je korišćenje simboličkih manipulacija i izračunavanja (obično u aritmetici višestruke preciznosti) kakva su moguća u programskim paketima *Maple* ili *Mathematica*. Ipak, ovaj pristup je ograničen na probleme koji nisu suviše zahtevni.

Neka je M neki skup, $a \in M$ i $P(a)$ je neko logičko tvrđenje. Definišimo kvantifikator $\bigwedge$ na sledeći način:

$$\bigwedge_{a \in M} P(a) \quad \text{znači:} \quad P(a) \text{ je tačno za sve elemente } a \in M.$$

Uvedimo sledeće skupove:

$$\begin{aligned}\overline{\mathbb{R}} : &= \mathbb{R} \cup \{-\infty\} \cup \{+\infty\},\\ \overline{S} : &= \overline{S}(\beta, n, e_{\min}, e_{\max}) := S(\beta, n, e_{\min}, e_{\max}) \cup \{-\infty\} \cup \{+\infty\}.\end{aligned}$$

Sada smo u mogućnosti da definišemo *zaokruživanje brojeva* iz skupa $\overline{\mathbb{R}}$ u $\overline{S}$ kao transformaciju $\square : \overline{\mathbb{R}} \to \overline{S}$, gde je sa $\square$ označen mod zaokruživanja. Na primer, pri izračunavanju $a + b$, gde su a i b PT brojevi, rezultat koji daje APT je $\square(a + b)$.

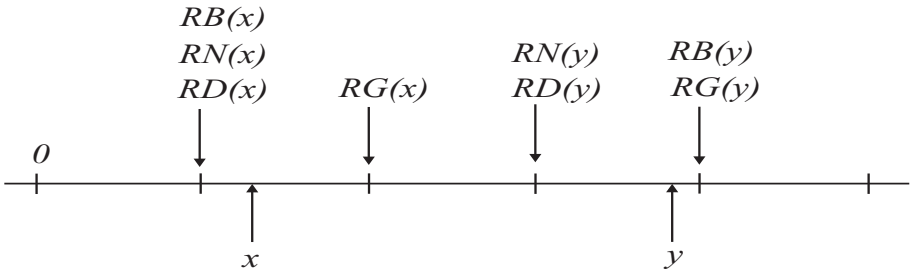
Očigledno je

$$\bigwedge_{x \in \overline{S}} \square x = x,$$

tj. zaokruživanje PT broja daje sam taj broj. U opštem slučaju, tj. ako $x \notin \overline{S}$, postoji više načina da se realan broj x zaokruži. Posmatraćemo sledeće tipove zaokruživanja:

$$\begin{aligned}
RD(x) : &= \bigwedge_{x \in \overline{\mathbb{R}}} \triangledown x := \max\{y \in \overline{S} \ : \ y \le x\}, \quad \text{silazno direktno zaokruživanje,}\\
RG(x) : &= \bigwedge_{x \in \overline{\mathbb{R}}} \triangle x := \min\{y \in \overline{S} \ : \ y \ge x\}, \quad \text{uzlazno direktno zaokruživanje,}\\
RN(x) : &= \bigwedge_{x \in \overline{\mathbb{R}}} \square_0 x := \begin{cases} \triangledown x & \text{ako je } x \ge 0 \\ \triangle x & \text{ako je } x < 0 \end{cases}, \quad \text{zaokruživanje prema nuli,}\\
RB(x) : &= \bigwedge_{x \in \overline{\mathbb{R}}} \square_N x := y, \ y \in \overline{S}, \ |y - x| \to \min, \quad \text{zaokr. na najbliži PT broj.}
\end{aligned}$$

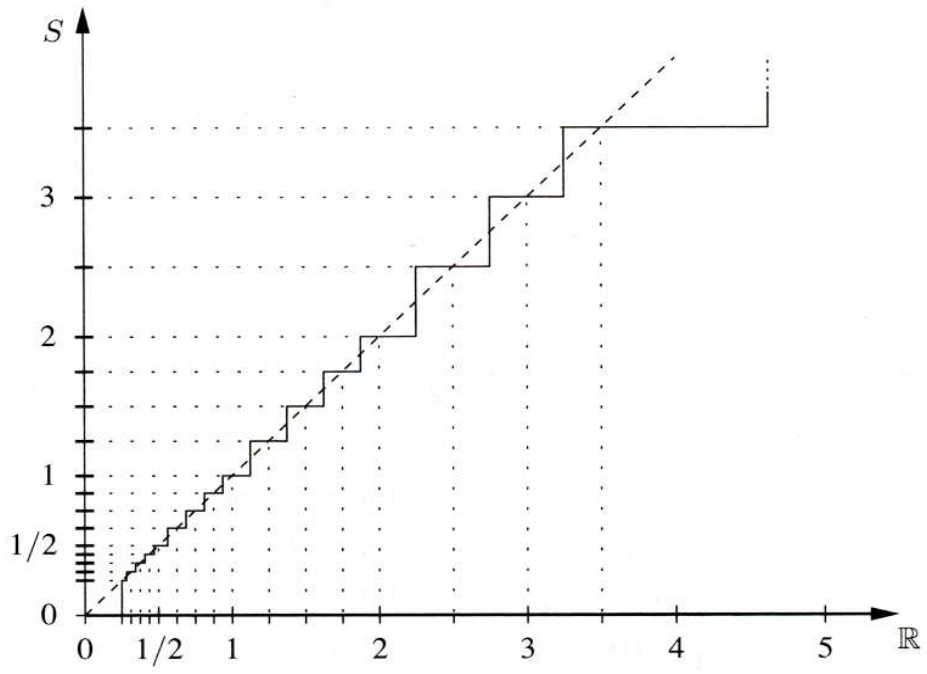
Drugo slovo u oznakama RD, RG, RN, RB je opisnog karaktera i redom znači D–dole, G–gore, N–nula, B-bliži. Slika 1.4 ilustruje ova četiri tipa zaokruživanja.



Slika 1.4 Četiri tipa zaokruživanje podrazumevajući da su x i y pozitivni brojevi

U slučaju da je x tačno na polovini između dva PT broja, pravilo o zaokruživanju mora biti definisano. Najčešće se koristi *zaokruživanje ka najbližem parnom*: između dva susedna PT broja x se zaokružuje na onaj čija se mantisa završava parnom cifrom. Postoji takođe pravilo zaokruživanja *prema većem*, podrazumevajući zaokruživanje prema broju sa mantisom čija je poslednja cifra veća. Oba pravila su ugrađena u IEEE standard 754-2008. Na primer, broj 0.53873225 u dekadnom sistemu preciznosti $p = 7$ se može zaokružiti na jedan od dva susedna PT broja 0.5387322 i 0.5387323; prema prvom pravilu (*parna cifra*) se zaokružuje na 0.5387322, a prema drugom pravilu (*veća cifra*) na 0.5387323.

Dijagram za zaokruživanje na najbliži PT broj za PT sistem $S(2, 3, -1, 2)$ prikazan je na slici 1.5.



Slika 1.5 Zaokruživanje na najbliži PT broj

Jedna alternativna definicija za $RB(x)$ je najbliži PT broj y koji zadovoljava $|y| \leq |x|$. Ova operacija se naziva *odsecanje*. Aritmetika sa ovim tipom zaokruživanja bila je korišćena kod IBM/370 PT sistema. Razlika između odsecanja i zaokruživanja (ka najbližem) postala je više nego očigledna, skoro dramatična, na primeru indeksa Vankuverske berze osamdesetih godi-

na dvadesetog veka. Kurs na berzi u januaru 1982. bio je ocenjen indeksom 1000. U novembru 1983. indeks se smanjio na 520 iako je berza poslovala uspešno. Šta se dogodilo? Indeks je bio zapisivan koristeći tri decimalna mesta. Otkriveno je da je kompjuterski program pri izračunavanju koristio odsecanje umesto zaokruživanje. S obzirom da je indeks bio preračunavan više hiljada puta na dan, svaki put sa negativnom greškom, negativan uticaj uveden odsecanjem postao je značajan. Posle novog preračunavanja koristeći zaokruživanje, indeks se skoro udvostručio.

Sledećom definicijom uvodi se tzv. *pravilno zaokruživanje* funkcije.

Definicija 1.1. Neka su $a, b, \ldots$ PT brojevi, f realna funkcija jedne ili više promenljivih, $p \geq 1$ prirodan broj i neka RB definiše operaciju zaokruživanja ka najbližem PT broju. Kažemo da je c *pravilno zaokruživanje* od $f(a, b, \ldots)$ i pišemo $c = RB_p(f(a, b, \ldots))$ ako je c broj sa pokretnom tačkom najbliži broju $f(a, b, \ldots)$ sa preciznošću p dobijen prema datom modu zaokruživanja. U slučaju da je više brojeva na istom rastojanju od $f(a, b, \ldots)$, mora se definisati mod zaokruživanja (prema *parnom* ili *najvećem*). ▲

Definicija 1.2. Neka $RB(x)$ označava PT broj najbliži datom broju $x \in \mathbb{R}$, tj., $x \to RB(x)$ definiše zaokruživanje ka najbližem PT broju. Kažemo da je $RB(x)$ *prekoračenje* opsega ako je $|RB(x)| > \max\{|y| : y \in S\}$ i *potkoračenje*[1] opsega ako je $0 < |RB(x)| < \min\{|y| : 0 \neq y \in S\}$. ▲

Primer 1.1. Neka je baza $\beta = 10$ sa preciznošću $p = 2$, $e_{\max} = 3$ i posmatrajmo sabiranje brojeva $x = 0.9234 \cdot 10^3$ i $y = 0.7656 \cdot 10^2$. Tačna suma je $0.99996 \cdot 10^3$. Sa *zaokruživanjem prema nuli* dobija se $RN(y) = 0.0765 \cdot 10^3$ pa je suma $0.9999 \cdot 10^3$, što se može predstaviti kao PT broj sa datim parametrima i ovde se ne javlja prekoračenje. Ako se primeni *zaokruživanje ka najbližem* dobija se $RB(y) = 0.0766 \cdot 10^3$ te se suma $x + y$ zaokružuje na $0.1000 \cdot 10^4$. U ovom slučaju eksponent 4 prelazi opseg $e_{\max} = 3$ tako da dobijamo $+\infty$ kao rezultat, sa prekoračenjem. U ovom modelu prekoračenje zavisi ne samo od operanda već i od moda zaokruživanja. ▲

Ako $x \in \mathbb{R}$ leži u opsegu PT brojeva (tj. $x \in [-B, B]$), tada se može pokazati da je

$$RB(x) = x(1 + \delta), \quad |\delta| < u. \tag{1.3}$$

[1] Na engl. *overflow* i *underflow*, termini koji se često koriste i kod nas.

U cilju analize greške zaokruživanja kod numeričkih algoritama, najčesće se usvaja sledeći model koji polazi od (1.3) Aritmetika sa IEEE 754-1985 standardom imala je više revizija i poboljšanja tokom godina, što je dovelo do nove verzije IEEE 754-2008 standarda u junu 2008. Može se reći da ova APT sa 2008-standardom na veoma efikasan način simulira beskonačan kontinualan skup (realnih brojeva) pomoću konačnog diskretnog skupa („mašinskih brojeva"), pri čemu je učinjen vrlo zadovoljavajući kompromis između brzine, tačnosti, dinamičkog opsega, jednostavnog korišćenja i implemetacije, zauzeća memorijskog prostora, itd., što su retko kada kompatibilni zahtevi:

$$RB(x * y) = (x * y)(1 + \delta), \quad |\delta| \leq u, \quad * \in \{+, -, \cdot, /\}. \tag{1.4}$$

Takođe se podrazumeva da model (1.4) važi i za kvadratni koren.

Skalarni proizvod vektora $x = (x_1, \ldots, x_r)$ i $y = (y_1, \ldots, y_r)$, definisan na konvencionalan način kao

$$x \cdot y = \sum_{i=1}^{r} x_i y_i,$$

se veoma često sreće u primenama tako da moderni standardi APT predviđaju uvođenje ove operacije kao *pete operacije* pored sabiranja, oduzimanja, množenja i deljenja, o čemu će kasnije još biti reči. Osobine skalarnog proizvoda u prisustvu greške zaokruživanja u običnoj i intervalnoj aritmetici detaljno su razmatrani u knjigama [67], [82] i [103]. Važan rezultat koji se odnosi na skalarni proizvod u prisustvu greške zaokruživanja dat je sledećom teoremom.

Teorema 1.2. (Kulisch [82]) *Neka je* $\overline{S} = \overline{S}(\beta, p, e_{\min}, e_{\max})$ *PT sistem u kome je pomoću* $\square : \overline{\mathbb{R}} \to \overline{S}$ *definisano zaokruživanje. Tada je*

$$\square\left(\sum_{i=1}^{r} x_i y_i\right) = (1 - \epsilon)\sum_{i=1}^{r} x_i y_i = (x \cdot y)(1 - \epsilon), \quad |\epsilon| < \epsilon^*,$$

gde je

$$\epsilon^* = \begin{cases} \frac{1}{2}\beta^{1-p} & \text{za } \square = RB \\ \beta^{1-p} & \text{za } \square \neq RB \end{cases}$$

i RB označava zaokruživanje ka najbližem PT broju. ▲

Koristeći ovu teoremu i definiciju $a \boxdot b = \square(a \cdot b)$ za $a, b \in \overline{S}$, izvodi se sledeća granica greške za skalarni proizvod u prisustvu greške zaokruživanja

$$\left| \sum_{i=1}^{r} x_i y_i - \sum_{i=1}^{r} x_i \boxdot y_i \right| \leq \frac{r\epsilon^*}{1 - r\epsilon^*} \sum_{i=1}^{r} |x_i y_i|.$$

Ova ocena greške važi sve dok je $0 \leq r\epsilon^* < 1$, tj. ako je $\epsilon^* < 1/r$. Na primer, za dvostruku preciznost u binarnom kodu IEEE standarda 754 je $\epsilon^* = 2^{-53}$, dakle $r < 9 \cdot 10^{15}$, što znači da se može raditi sa vektorima vrlo velikih dužina.

Analiza greške kod algoritama primenjenih u kompleksnoj aritmetici koristi model za osnovne aritmetičke operacije u APT podrazumevajući realnu aritmetiku. Za kompleksne brojeve $x = a + ib$ i $y = c + id$ izračunava se

$$\begin{aligned} x \pm y &= a + c \pm i(b + d), \\ x \cdot y &= ac - bd + i(ad + bc), \\ x/y &= \frac{ac + bd}{c^2 + d^2} + i\frac{bc - ad}{c^2 + d^2}. \end{aligned} \tag{1.5}$$

Neka je, kao i ranije, u jedinica zaokruživanja, i neka je $\gamma_k := \dfrac{ku}{1 - ku}$. Sledeća teorema je dokazana u [67]:

Teorema 1.3. *Za $x, y \in \mathbb{C}$ osnovne operacije izračunate na osnovu* (1.5) *sa standardnim modelom* (1.4) *zadovoljavaju*

$$\begin{aligned} RB(x \pm y) &= (x + y)(1 + \delta), & |\delta| \leq u, \\ RB(xy) &= x \cdot y(1 + \delta), & |\delta| \leq \sqrt{2}\,\gamma_2, \\ RB(x/y) &= (x/y)(1 + \delta), & |\delta| \leq \sqrt{2}\,\gamma_4. \end{aligned}$$

Mada su performanse APT godinama usavršavane tako da APT može da reši najveći broj savremenih problema zahvaljujući vrlo sofisticiranim i adaptivnim algoritmima, moćnim bibliotekama, vrlo naprednom hardveru i IEEE standardu, ponekad je vrlo teško dati procenu dobijenog rezultata. Dobijeno rešenje ili rezultat izračunavanja može biti zadovoljavajući, nedovoljno tačan ili potpuno pogrešan. Ponekad je teško ustanoviti koji od ova tri slučaja se desio. Da bismo ilustrovali ove nedostatke, u nastavku dajemo dva primera koji ukazuju na pomenute teškoće i probleme pri realizaciji aritmetike sa pokretnom tačkom.

Primer 1.2. Ovaj originalno konstruisan primer u duhu Kahanovih rekurentnih relacija [102] efektno prikazuje uticaj greške zaokruživanja na tačnost rezultata izračunavanja. Pri izračunavanju vrednosti x_k u APT dvostruke preciznosti (oko 16 značajnih decimalnih cifara) pomoću rekurentne relacije

$$\begin{cases} x_0 = 1, \\ x_1 = -5, \\ x_{k+1} = 207 - \dfrac{1412}{x_k} + \dfrac{2400}{x_{k-1}x_k}, \end{cases} \tag{1.6}$$

posle izvesnog broja iterativnih koraka dobijamo da se x_k približava vrednosti 200. Međutim, koristeći se metodama za rešavanje diferencnih jednačina dolazimo do opšteg rešenja date rekurentne relacije u obliku

$$x_k = \frac{200^{k+1}a + 4^{k+1}b + 3^{k+1}}{200^k a + 4^k b + 3^k},$$

gde su a i b proizvoljne konstante. Za date početne vrednosti x_0 i x_1 dobija se $a = 0,\ b = -2/3$, tako da se gornja formula svodi na jednostavan oblik

$$x_k = \frac{-\frac{2}{3}\cdot 4^{k+1} + 3^{k+1}}{-\frac{2}{3}\cdot 4^k + 3^k} = 4 - \frac{1}{1 - \frac{2}{3}\left(\frac{4}{3}\right)^k}. \tag{1.7}$$

Iz poslednje formule je jasno da x_k konvergira ka 4 kad k raste neograničeno.

Pogrešan rezultat ($x_k \to 200$) koji se dobija u aritmetici dvostruke preciznosti je posledica greške zaokruživanja pri izračunavanju. Naime, u ovom postupku ne dobija se (teorijska) vrednost $a = 0$ već broj η koji je reda mašinske preciznosti 10^{-16}. Na taj način, umesto formule (1.7) imamo

$$\hat{x}_k \approx \frac{200^{k+1}\eta - \frac{2}{3}\cdot 4^{k+1} + 3^{k+1}}{200^k\eta - \frac{2}{3}\cdot 4^k + 3^k}. \tag{1.8}$$

Bez obzira što je veličina η vrlo mala, poslednja relacija vrlo brzo konvergira ka 200. Čak i korišćenje aritmetike višestruke preciznosti primenjeno na (1.8) samo odlaže konvergenciju ka broju 200.

k	Izračunata vrednost	Tačna vrednost
2	9.400000000000000	9.400000000000000
3	5.7234042553191489	5.7234042553191489
4	4.903345724907063	4.903345724907063
7	4.250352959133056	4.250352959133056
10	4.0922638222296479	4.0922638222296296
15	4.020460574181692	4.020455220324120
19	32.2526502744188142	4.006382906979505
20	179.162468609660354	4.0047795533734821
21	199.534220359450101	4.003580386867579
25	199.999999925113958	4.001130090850963
29	199.999999999999988	4.000357291792170

Tabela 1.4 Članovi niza $\{x_k\}$ izračunati u aritmetici dvostruke tačnosti

Rezultati izračunavanja korišćenjem aritmetike dvostruke preciznosti (oko 16 značajnih decimalnih cifara) prema formulama (1.6) i (1.7) dati su u tabeli 1.4. Primetimo da se čak i tačna vrednost (prema formuli (1.7)) ne može dobiti sa više od 3 značajne cifre mantise. ▲

Primer 1.3. Rekurzivni postupak za izračunavanje nekog izraza može dovesti do rezultata nezadovoljavajuće tačnosti ako se startuje sa nedovoljno tačnom početnom vrednošću jer se greška akumulira i „širi" tokom rekurzije. Posmatrajmo, na primer, izračunavanje određenog integrala

$$I_n = \frac{1}{e}\int_0^1 x^n e^x dx$$

pomoću rekurzivne formule

$$I_n = 1 - nI_{n-1} \quad (n = 1, 2, \ldots).$$

Polazeći od vrednosti $I_0 = 1 - 1/e \approx 0.6321205588$ sa 10 tačnih cifara, dobijamo

$$I_7 = 0.\underline{1123836}480$$

sa 6 tačnih cifara. Da bismo dobili I_{14} sa tačnošću 10^{-3} potrebno je startovati čak sa 16 tačnih cifara za I_0!

Objašenje leži u činjenici da je proces definisan gornjom rekurentnom relacijom **nestabilan**. Greška od δ jedinica u I_0 daje gresku od δ jedinica pri izračunavanju I_1. Ova greška se multiplicira faktorom 2 pri izračunavanju I_2, faktorom 3 za I_3 itd. To znači da je greška pri izračunavanju I_n reda $n!\delta$. Za I_7 i $\delta = 10^{-10}$ (razmatran slučaj) dobija se greška reda $7!\delta \approx 5 \times 10^{-7}$. Prema tome, usled nestabilnosti rekurzivnog procesa greške dovode do pogrešnih rezultata za I_n. ▲

Problem zaokruživanja u APT je naročito izražen kod izvršavanja osnovnih aritmetičkih operacija i pri izračunavanju elementarnih funkcija, te zbog toga dajemo kraću diskusiju koja se odnosi na ovaj problem. Elementarne funkcije su matematičke funkcije koje se najčešće pojavljuju u praksi, a da pritom imaju formu koja se ne može dalje uprostiti. U IEEE 754-2008 standardu to su sledeće funkcije:

$$
\begin{aligned}
&e^x,\ e^x-1,\ 2^x,\ 2^x-1,\ 10^x,\ 10^x-1,\\
&\ln(x),\ \log_2(x),\ \log_{10}(x),\ \ln(1+x),\ \log_{10}(1+x),\\
&\sqrt{x^2+y^2},\ 1/\sqrt{x},\ (1+x)^n,\ x^n,\ x^{1/n}\ (n \text{ je ceo broj}),\\
&\sin(x),\ \cos(x),\ \tan(x),\ \sin(\pi x),\ \cos(\pi x),\\
&\arctan(x)/\pi,\ \arctan(y/x)/\pi,\ \arcsin(x),\ \arccos(x),\\
&\sinh(x),\ \cosh(x),\ \tanh(x),\ \sinh^{-1}(x),\ \cosh^{-1}(x),\ \tanh^{-1}(x).
\end{aligned}
$$

S obzirom da se ove funkcije veoma često pojavljuju u naučnim istraživanjima i mnogim praktičnim primenama, kao i u slučaju realizacija osnovnih aritmetičkih operacija, potrebno je da one budu izračunate što brže i sa što je moguće većom preciznošću.

Izračunavanje elementarnih funkcija ugrađeno je u specijalne potprograme koji su ili deo softvera (kod računara u ranijem periodu) ili hardvera (kod modernih računarskih sistema). U većini slučajeva za njihovo izračunavanje koriste se razvoji u stepeni red, na primer,

$$
\begin{aligned}
e^x &= \sum_{k=0}^{+\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots;\\
\ln(1+x) &= \sum_{k=0}^{+\infty} \frac{(-1)^k x^{k+1}}{k+1} = x - \frac{x}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \ (-1 < x \leq 1);\\
\sin x &= \sum_{k=0}^{+\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots .
\end{aligned}
$$

U primeni stepenih redova može doći do ekstremno loših rezultata. Radi ilustracije, razmotrimo stepeni red za sinus

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \quad . \tag{1.9}$$

Teorijski, ovaj red važi za svako konačno x koje pripada intervalu konvergencije i greška odsecanja koja se dobija prekidom sabiranja alternativnog reda posle konačno mnogo članova je manja po apsolutnoj vrednosti od prvog zanemarenog člana. Ovo tvrđenje bi bilo tačno kada bi postojao način za čuvanje beskonačno mnogo cifara u svakom aritmetičkom rezultatu. Međutim, u praksi koristimo računarsku aritmetiku ograničene preciznosti koja prouzrokuje da stepeni red (1.9) postaje potpuno beskoristan za veliku vrednost argumenta x.

Primer 1.4. Primenom formule (1.9) nalazimo da je

$$\sin 1830^\circ = 639408$$

pri čemu je pri izračunavanju argument x bio izražen u radijanima, $1830^\circ = 31.9395142$ radijana. Ovako besmisleni rezultati su posledica zaokruživanja brojeva pri izračunavanju. Naime, greške zaokruživanja koje se pojavljuju pri primeni aritmetike ograničene preciznosti uzrokuju gubitak značajnih cifara u međurezultatima (u pravcu sabiranja). Čak i korišćenje aritmetike dvostruke preciznosti (16 značajnih cifara) ne može rešiti problem zaokruživanja; dobija se $\sin 2550^\circ = 158.59$ i $\sin 2910^\circ = 20217.9$. U praksi, gornji problem se rešava svođenjem ugla na ugao manji od 2π i primenom osobine periodičnosti sinus funkcije.

Zbog navedenih problema u praksi se koriste kombinovane tehnike, od kojih pominjemo one najvažnije. Napomenimo da su ove tehnike od specijalnog interesa pri izračunavanju elementarnih funkciju u aritmetici višestruke preciznosti (AVP).

1. Redukcija argumenta
2. Izračunavanje pomoću stepenog reda
3. Njutnov metod (ili iterativni metod višeg reda)
4. Aritmetičko-geometrijska sredina
5. Binarno „razlaganje”

Ovi metodi detaljno su razmatrani u knjizi [19, Pogl. 4].

1.3 Celobrojna aritmetika

Specijalan slučaj računarske aritmetike je *celobrojna aritmetika* (od engl. *integer arithmetic*), koja operiše sa celim brojevima. Ova aritmetika je vrlo korisna ne samo pri izračunavanju i manipulaciji sa celim brojevima, već i kao osnova za konstrukciju aritmetike višestruke preciznosti (na primer, GNU MP aritmetika). Izaberimo za *bazu* prirodan broj $\beta > 1$. Prirodan broj A se predstavlja pomoću niza od n cifara a_i (reč dužine n) kao razvoj u bazi β:

$$A = a_{n-1}\beta^{n-1} + \cdots + a_1\beta + a_0,$$

gde je $0 \leq a_i \leq \beta - 1$, podrazumevajući da je $a_{n-1} \neq 0$. S obzirom da je baza β najčešće nepromenljiva, jedino dužina n i cifre $(a_i)_{0 \leq i \leq n-1}$ treba da budu smešteni u memoriju.

Pri izvođenju operacija u bilo kojoj računarskoj aritmetici vrlo značajna karakteristika je *aritmetička složenost* ili *cena* operacija, koja se izražava brojem izvršenih mašinskih instrukcija, ili ekvivalentno (sa tačnošću do konstantnog faktora) *vremenom izvršenja* na procesoru.

Označimo sa $M(b)$ vreme potrebno za množenje dva broja od b-bitova u celobrojnoj aritmetici. $M(b)$ može takođe da označava potreban broj mašinskih instrukcija. Uobičajeno množenje $A \cdot B$ kakvo se uči u školi (zvaćemo ga *osnovnim množenjem*) množenjem prvog broja A ciframa drugog broja i pomeranjem za jednu poziciju, ima cenu $M(b) = O(b^2)$.[2]

Sabiranje i oduzimanje dva cela broja koji se sastoje od b-bitova ima cenu od $O(b)$ i to je uglavnom zanemarljivo u odnosu na cenu množenja $M(b)$. Ipak, i dalje se vrše napori u cilju smanjenja konstantnog faktora koji implicitno figuriše u ceni $O(b)$ sabiranja ili oduzimanja. Drugim rečima, ako je cena sabiranja/oduzimanja $c_a b$, treba smanjiti c_a što je moguće više. Ovo smanjenje može se izvršiti samo do izvesne granice jer konstantni faktor zavisi od karakteristika upotrebljenog procesora.

Osnovno množenje sa cenom $M(b) = O(b^2)$ suviše je skupo i ne koristi se u modernim računarima. Umesto toga primenjuje se tzv. Karacubin algoritam [76] za množenje celih brojeva (takođe pogodan i za množenje polinoma). Osnovna ideja ovog algoritma je svođenje množenja dužine n na tri množenja dužine $n/2$, plus izvesne dodatne operacije sa cenom $O(n)$.

[2] $\alpha = O(\beta)$ označava da su brojevi α i β istog reda veličine.

Cena Karacubinog algoritma je $K(n) = O(n^\alpha)$, gde je $\alpha = \log_2 3 \approx 1.585$. Dokaz ovog tvrđenja kao i sam algoritam dati su u knjizi [19].

Karacubina ideja dovela je do generalisanog Tom-Kukovog (Toom-Cook) r-metoda za množenje, nazvanog po autorima A. Toma [138] i S. Kuka [31]. Neka su brojevi koji se množe predstavljeni kao

$$a_0 + a_1 x + \cdots + a_{r-1} x^{r-1}$$

i

$$b_0 + b_1 x + \cdots + b_{r-1} x^{r-1}$$

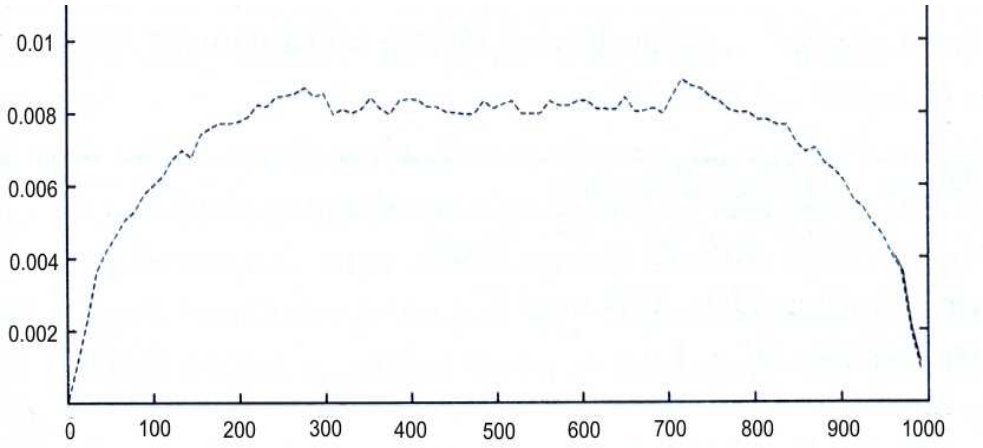
sa $x = \beta^k$ i $k = \lceil n/r \rceil$ ($\lceil x \rceil$ označava najmanji ceo broj m takav da je $m \geq x$). S obzirom da je njihov proizvod $C(x)$ polinom reda $2r-2$, dovoljno je izračunati $C(x)$ u $2r-1$ različitih tačaka da bismo ponovo dobili $C(x)$, a time i $C(\beta^k)$. Ako je r optimalno izabrano, Tom-Kukovo množenje brojeva od n reči troši vreme $n^{1+O(1/\sqrt{\log n})}$. Dakle, Tom-Kukov r-metod svodi množenje brojeva od n-reči na $2r-1$ množenja od oko n/r reči. Slučaj $r = 2$ odgovara Karacubinom algoritmu. Slučaj $r = 3$ je poznat kao Tom-Kukov 3-metod ili jednostavno Tom-Kukov metod. „Brzo" množenje je takođe razmatrano u radu [49].

Deljenje je skuplja operacija u odnosu na množenje i u cilju smanjenja njene aritmetičke složenosti razvijen je veći broj algoritama (za više detalja videti knjigu [19]). Deljenje brojeva A/B se često realizuje kao kombinacija množenja i inverzije $1/B$, tj. $A/B = A \cdot (1/B)$, pri čemu su posebni napori usmereni na optimizaciju cene inverzije $1/B$. Ako za izračunavanja f i g u funkciji broja bitova b uvedemo oznaku $f(b) \sim g(b)$ koja znači da $f(b)/g(b) \to 1$ kada $b \to \infty$, tada se može dokazati da je cena deljenja $D(b)$:

– $D(b) \sim 2M(b)$ za Karacubino množenje;

– $D(b) \sim 2.63M(b)$ za Tom-Kukov metod.

Gornji podaci se odnose na slučaj kada se operacije vrše nad brojevima iste dužine. Ako imamo brojeve od n i m reči, pri čemu je $n \leq m$, u tom slučaju se mora odabrati posebna strategija za sprovođenje operacija (često zvanih *nebalansirane operacije*), koja zavisi od razlike brojeva m i n. Na slici 1.6 dato je vreme izvršavanja nebalansiranog množenja kada je $n = x$ i $m = 1000 - x$. Kao što se i moglo očekivati, vreme množenja broja od x-reči sa brojem od $(n-x)$-reči je približno isto vremenu za množenje brojeva od $n - x$ i n reči.



Slika 1.6 Množenje brojeva od x reči i brojeva od $n - x$ reči
u GNU MP koristeći procesor Core 2 na 2.83GHz

1.4 Aritmetika višestruke preciznosti

U ovom poglavlju dajemo detaljan pregled realizacija osnovnih aritmetičkih operacija i izračunavanja elementarnih funcija u aritmetici višestruke preciznosti (skraćeno AVP), uključujući i neke originalne priloge. Razmatranja su prvenstveno usmerena na aritmetiku višestruke preciznosti. Pod „visokom preciznošću" podrazumevamo aritmetiku koja radi sa više značajnih cifara nego standardna IEEE 754 aritmetika sa pokretnom tačkom; tipično, broj bitova b koji definiše preciznost (tzv. b-preciznost) je reda 100 ili čak 1000. Računska efikasnost i algoritmi za izvršavanje osnovnih aritmetičkih operacija i izračunavanje elementarnih funcija su ukratko analizirani u Odeljku 4.4 uzimajući vreme izvršenja kao glavni parametar.

Pri oceni računske cene algoritama u b-preciznosti, usvajaju se izvesne pretpostavke za dovoljno veliko b. Na primer, u izvesnom smislu ignorišemo „konstantne" faktore koji su ograničeni kada $b \to \infty$ pa tako, umesto cene sabiranja Cb, gde je C konstanta takva da je $C = O(1)$, pišemo $O(b)$. Mada su konstantni faktori od praktičnog značaja i treba da budu što manji, oni zavise od tipa softverske implementacije, teorijske analize računske cene (koja često pretpostavlja izvesne uslove) kao i upotrebljenog procesora. Ipak, radeći u AVP, vrednost ovih konstantnih faktora samo malo utiče na konačnu računsku cenu kada se radi sa dovoljno velikom b-preciznošću.

Mada je aritmetika sa pokretnom tačkom u dvostrukoj preciznosti (oko 16 značajnih decimalnih cifara) dovoljno dobra za najveći broj praktičnih primena, aritmetika u četvorostrukoj preciznosti , ili kraće q-preciznosti[3] (oko 33–34 značajne decimalne cifre), postaje sve više značajna u mnogim naučnoistraživačkim disciplinama. Pomenimo fiziku visoke energije (proučavanje interakcije čestica), nelinearne procese i simulacije, procesiranje signala (kompresija/dekompresija), teoriju brojeva (gde se koristi kombinacija simboličkog i numeričkog izračunavanja), „eksperimentalnu matematiku" [4] [5], [10] višestruku integraciju singularnih funkcija, numeričke algoritme vrlo visoke tačnosti [9], [88], [93] kao i komercijane aplikacije poput CAD modeliranja metodom konačnih elemenata, 3-D grafičke aplikacije visokog kvaliteta, statistike i kriptografije za kodiranje uređaja za obezbeđenje i zaštitu. Pomenimo da je u upotrebi, mada ređe, i aritmetika osmostruke preciznosti, tzv. *qd*-aritmetika (od engl. *quad-double*)[5] koja radi sa oko 67 značajnih decimalnih cifara.

Izračunavanja sa visokom preciznošću ne daju samo rezultate veće preciznosti, već u slučaju velikog broja oduzimanja bliskih brojeva ili grešaka zaokruživanja obezbeđuju tačnije rezultate u poređenju sa aritmetikom niže preciznosti. Štaviše, korišćenje AVP poboljšava konvergenciju mnogih iterativnih algoritama smanjujući na taj način broj iteracija, kao što se može videti sa slike 1.8 gde je grafički prikazan broj iteracija i tačnost aproksimacija pri primeni algoritma za rešavanje sistema velikog broja linearnih jednačina primenjujući aritmetike različite preciznosti. Sa slike 1.8 se vidi da q-aritmetika daje aproksimacije velike tačnosti (reda 10^{-25}) uz korišćenje znatno manjeg broja iteracija.

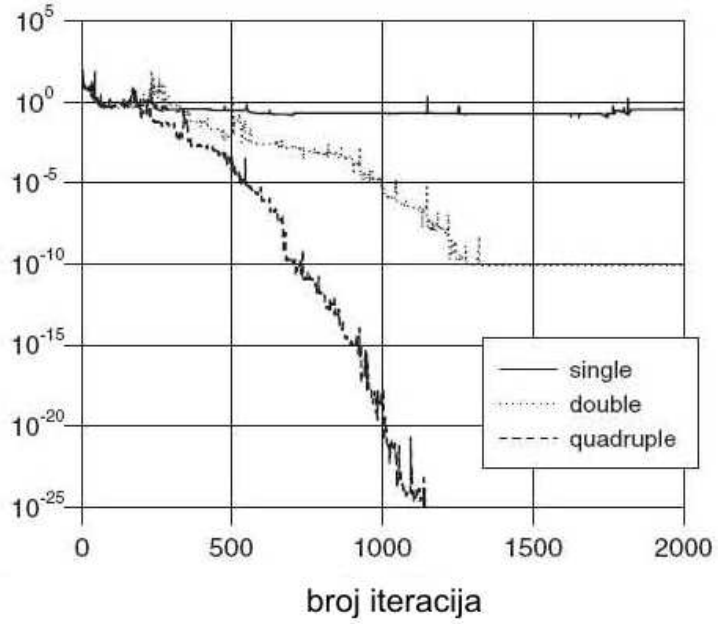
Govoreći o konvergenciji algoritama za izračunavanje ili rešavanje jednačina, treba dodati da iterativan pristup sukcesivno poboljšava tačnost aproksimacije traženog rezultata izračunavanja ili rešenja jednačine iz koraka u korak. Ukoliko se želi velika preciznost aproksimacije, neophodno je primeniti AVP.

---

[3] Od engl. *quadruple*. Ponekad se koristi i termin *dd*-preciznost (od *double-double*) kao što je naznačeno na slici 1.7.

[4] Pod ovim se podrazumeva korišćenje moderne kompjuterske tehnologije kao aktivnog alata u matematičkim istraživanjima. Tipičan primer su algoritmi za izračunavanje broja π, osnove prirodnog logaritma e, Eulerove konstante γ, nula Rimanove zeta funkcije, kao i testiranje algoritama vrlo brze konvergencije i preciznosti.

[5] U literaturi na engleskom jeziku koristi se i termin *octuple precision* (osmostruka preciznost).



Slika 1.7 q-preciznost dovodi do brže konvergencije metoda za rešavanje sistema velikog broja jednačina

Bez korišćenja AVP često se u nekim komplikovanim izračunavanjima može doći do pogrešnih rezultata ili zaključaka. Ilustrativan je sledeći primer dat u radu [5].

Primer 1.5. Određeni integral

$$I_p = \int_0^\infty \cos(2x) \prod_{n=1}^{p} \cos\Big(\frac{x}{n}\Big) dx,$$

gde je p velika vrednost, tipično $p > 100$, je teško izračunati zbog oscilatornog ponašanja proizvoda $\prod_{n\geq 1} \cos(x/n)$ čak i u aritmetici višestruke preciznosti. Korišćenjem moćnog tzv. *tanh-sinh* kvadraturnog pravila za numeričku integraciju (takođe poznata pod imenom *dvostruka eksponencijalna formula*, videti [135]) dobija se rezultat (sa 60 prvih cifara)

0.392699081698724154807830422909937860524645434187231595926812...

Na prvi pogled vrednost integrala je $\pi/8$, što nije tačno jer je

$$\frac{\pi}{8} = \underline{0.392699081698724154807830422909937860524617492188822762186 8}...$$

Dakle, poslednje dve aproksimacije se razlikuju za 7.407×10^{-43}. ▲

Primer 1.6. Broj $y = e^{\pi\sqrt{163}}$ izračunat je u APT preciznosti p za nekoliko vrednosti p. Dobijene vrednosti date su u tabeli 1.5.

p	y
10	262537412600000000
15	262537412640769000
20	262537412640768744.00
25	262537412640768744.0000000
30	262537412640768743.999999999999

Tabela 1.5 Izračunate vrednosti izraza $e^{\pi\sqrt{163}}$.

Na prvi pogled izgleda da je $e^{\pi\sqrt{163}}$ ceo broj, sa cifrom 4 ispred decimalne tačke. Da je tako, to bi izgledalo potpuno neverovatno jer pri izračunavanju učestvuju dva transendentna broja e i π i iracionalan broj $\sqrt{163}$?! Međutim, ništa se ne može zaključiti o poslednjoj cifri ispred decimalne tačke dok se ne upotrebi još veća preciznost pri izračunavanju. Tako dobijamo

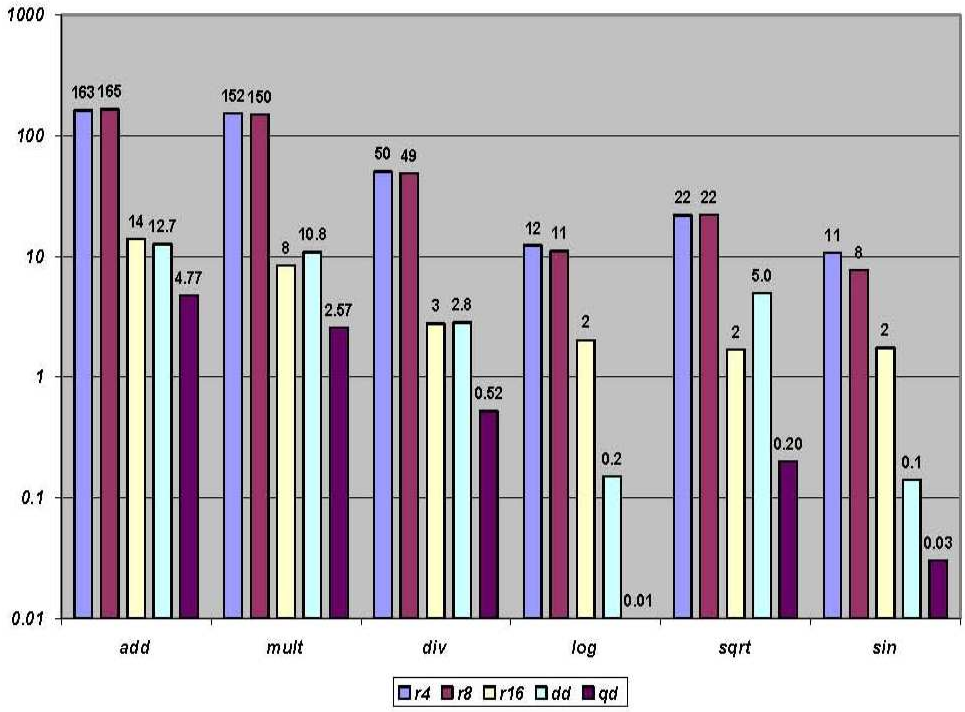
p	y
35	262537412640768743.99999999999925007
40	262537412640768743.9999999999992500725972

Prema tome, tražena cifra je 3 dok je iza decimalne tačke čak 12 cifara 9. $e^{\pi\sqrt{163}}$ nije ceo broj. ▲

Nabrojane dobre karakteristike AVP, od kojih je znatno uvećana preciznost na prvom mestu, imaju za posledicu sporije izvršavanje osnovnih aritmetičkih operacija (2–3 puta) a naročito sporije izračunavanje elementarnih funkcija (čak do 10 puta). Faktori usporenja su bar dva puta veći kod *qd*-aritmetike u odnosu na *q*-aritmetiku. Poboljšanja hardvera i softvara i korišćenje sofisticiranih numeričkih algoritama, na kojima se neprestano radi, stalno dovode do značajnog smanjenja navedenih faktora usporenja. Kao jedan ilustrativan primer poboljšanja hardvera pomenimo realizaciju specijalnog koprocesora za AVP nazvanog HAPPY (od High Arithmetic Precision

Processor Yoke) realizovanog na Institutu za visoke energije (KEK) u Cukubi (Japan) zajedničkim radom japanskih i francuskih istraživača. Detalji o ovom koprocesoru mogu se videti u radu [48]. Drugi primer je IBM-ov procesor za kriptografiju, vrlo zahtevno polje istraživanja i primene koje zahteva rad čak sa 1024 bita. Konačno, pomenimo i činjenicu da preciznost, osim više vremena za izvršenje operacija i izračunavanja funkcija i više memorije, zahteva i posebne tehnike koje treba da prevaziđu probleme koji se javljaju zbog ograničenog formata (oblasti eksponenta) pri korišćenju standardnog hardvera i softvera.

Na slici 1.8 date su karakteristike osnovnih aritmetičkih operacija i nekih elementarnih funkcija u aritmetikama različitih preciznosti za procesor Pentium M 2.8 GHz (Fedora core 3). rn označava da je korišćena aritmetika preciznosti od n decimalnih cifara ($n = 4, 8, 16$ na sl. 1.9), dok su dd i qd oznake za q(četvorostruku)-preciznost i osmostruku preciznost.



Slika 1.8 Karakteristike operacija i funkcija za Pentium M 2.8 GHz (Fedora core 3) izražene u Mflop/s

U današnje vreme IEEE 754-1985 i od skora 754-2008 postali su standard u računarstvu sa vrlo širokim primenama u najrazličitijim disciplinama i naučnim istraživanjima. Programi koji koriste ovaj format i operacije specificirane pomoću IEEE 754 instrukcija ponašaju se istovetno na različitim računarskim konfiguracijama, podrazumevajući da su procesori, operativni sistemi i kompajleri dovoljno adaptivni sa ovim standardom. Istraživači su u mogućnosti da konstruišu algoritme i dokazuju njihovu efikasnost i tačnost koristeći IEEE 754 operacije i nove pristupe i tehnike kao što su intervalna aritmetika, proširena APT, izračunavanje elementarnih funkcija sa informacijom o greški zaokruživanja, itd. Uprkos nekim kritikama koje dolaze kako od onih koji se bave hardverom (komplikovana implementacija u čipovima) tako i od kreatora sofvera (spori programi), IEEE 754 standard je nesumnjivo doveo do velikog napretka sa gledišta tačnosti, brzine i univerzalnog korišćenja.[6]

Međutim, IEEE 754 standard zahteva fiksne formate, posebno jednostruku i dvostruku preciznost sa 24, odnosno 53 bita mantise. Zbog toga je još u ranijoj fazi razvoja računarskih aritmetika ulagan napor za konstrukcijom softverskog oruđa za aritmetiku *višestruke preciznosti*, počev od već istorijskog Brentovog MP paketa [13], [14] pa do poznatih softvera Maple [25], GMP [52], CLN [56], PARI/GP [7], itd. Mada je tačnost koja se postiže ovim AVP prilično zadovoljavajuća, njihovi glavni nedostataci su nedovoljno jasna semantika, problem *pravilnog* zaokruživanja, kao i nestabilnost algoritama.[7]

Ovaj poslednji nedostatak se ne može ukloniti čak ni povećanjem preciznosti upotrebljene AVP. Zbog toga je sasvim razumljivo da se stalno radi na usavršavanju i najmanjih delova algoritama, bilo za izvršavanje osnovnih operacija, bilo za izračunavanje elementarnih i specijalnih funkcija. U ovom poslu zajednički učestvuju eksperti koji se bave računarskim naukama i matematičari koji razvijaju numeričke postupke. Napomenimo da je veliki doprinos na ovom polju dao profesor Kalifornijskog univerziteta u Los Anđelesu (poznatiji kao UCLA) Miloš Ercegovac (videti [41], [42], [148]), koji je diplomirao na Elektrotehničkom fakultetu u Beogradu 1965. godine i karijeru nastavio u SAD.

[6] Efikasno korišćenje nezavisno od tipa računara na kome je standard primenjen.

[7] Male promene u algoritmu mogu proizvesti znatne promene u krajnjem rezultatu.

Implementacija aritmetike višestruke preciznosti

U suštini postoje dva tipa realizacije AVP u zavisnosti od toga na koji način se predstavljaju brojevi sa pokretnom tačkom: primena celobrojne (intedžer) aritmetike i višecifarski format zasnovan na korišćenju simboličkog jezika (Mathematica, Maple, Form). Razmatrajući AVP, možemo razlikovati dve klase korišćenog softvera; biblioteke i interaktivne programe. Ova druga klasa često obezbeđuje i druge funkcionalnosti; ovo je slučaj sa programskim algebra-sistemima kao što su Maple [25], Mathematica [152], Magma, ili PARI/GP. Većina biblioteka sa AVP obezbeđuje četiri osnovne operacije $(+, -, \times, \div)$, ali one se razlikuju počev od korišćenog programskog jezika (Fortran, C, C++) pa do interno korišćene osnove $(2, 2^{32}, 2^{64}, 10, 10^4, 10^9)$, biblioteke matematičkih funkcija, efikasnosti algoritama i načina implementacije. Jedan od najvećih napora pri usavršavanju ovih i sličnih programa usmeren je na kontrolu osnovnih aritmetičkih operacija, pre svega pravilnog zaokruživanja. Najveći napredak je učinjen kod programskih paketa Maple i Mathematica. Veliki doprinos u pravcu pravilnog zaokruživana učinila je grupa francuskih naučnika sa instituta LORIA (Vije-de-Nansi) koja je razvila binarnu biblioteku za aritmetiku višestruke preciznosti sa pravilnim zakruživanjem, kraće nazvanu MPFR [47]. MPFR koristi programski jezik C i GNU MP paket.

Kao što je pomenuto, jedan od načina realizacije aritmetike višestruke preciznosti je korišćenje niza brojeva sa pokretnom tačkom (PT brojevi). Broj u aritmetici višestruke preciznosti x, skraćeno VP broj, je suma komponenata ovog niza. On se predstavlja u kompletnom registru dovoljne dužine kao promenljiva sa fiksnom tačkom tipa `long real`,

$$x = \sum_{i=1}^{n} x_i, \tag{1.10}$$

pri čemu se n naziva dužinom reprezentacije (1.10), x_i i-tom komponentom od x, a x *kompletnom promenljivom*, skraćeno *cv* (od engl. *complete variable*). Pod kompletnim registrom podrazumeva se registar u koji može da se smesti ceo broj u formatu sa fiksnom tačkom i da pritom dozvoli akumulaciju međurezultata bez gubitaka ijednog bita za čitavu proceduru koja je predviđena. Primera radi, u slučaju IEEE standarda sa dvostrukom preciznošcu kompletan registar se sastoji od oko 500 bajtova.

Komponente x_i su PT brojevi i mogu biti promenljive dužine. Preporučljivo je da apsolutne vrednosti komponenata u (1.10) budu poređane u niz prema svojim apsolutnim vrednostima, dakle, $x_1| > |x_2| > \cdots > |x_n|$ i da se eksponenti dve susedne komponente x_i, x_{i+1} razlikuju bar za m, gde je m dužina mantise. U tom slučaju kažemo da se mantise ne preklapaju i da je x predstavljen sa *optimalnom preciznošću* od oko $n \cdot m$ cifara mantise. Napomenimo da preklapanje mantisa može dovesti do gubitka preciznosti ([82]).

Drugi pristup pri realizaciji aritmetike proizvoljne preciznosti je predstavljanje PT brojeva kao celih brojeva (intedžera) proizvoljne „dužine" i odgovarajućeg eksponenta u obliku

$$x = (-1)^{s_x} \cdot m_x \cdot 2^{e_x},$$

gde je mantisa m_x intedžer čija je „dužina" (u bitovima) bliska preciznosti p_x broja x, $s_x \in \{0, 1\}$ i eksponent e_x je ceo broj. Celobrojna aritmetika višestruke preciznosti se, zatim, primenjuje nad mantisama vodeći računa o eksponentima. Na ovaj način se, u suštini, celobrojna aritmetika proizvoljne preciznosti koristi za implementaciju operacija APT, koje se zatim primenjuju pri izračunavanju elementarnih i specijalnih funkcija. Za više detalja videti Odeljak 14.4 u knjizi [103]. Ovaj prilaz je primenjen u ranim danima programiranjima kada je Brent realizovao svoje MP biblioteke u Fortranu [13], [14]. Kasnije su Bailey i njegovi saradnici koristili opisani prilaz za paket MP realizovan u ARPREC-u [6]. Nedavno realizovana MPFR biblioteka koristila je isti princip [47].

S obzirom na veliku kompleksnost AVP, istraživanja na poboljšanju karakteristika AVP se nastavljaju nesmanjenim tempom, kao što je i predvideo R. Brent [15]:

"Projekat koji će uvek biti aktuelan je implemetacija algoritama u višestrukoj precizosti za nove klase specijalnih funkcija, i poboljšanja efikasnosti onih algoritama u višestrukoj preciznosti koji su već implementirani."

Programski paketi za aritmetiku višestruke preciznosti

Mada su se paketi sa aritmetikom višestruke preciznosti pojavili još šezdesetih godina prošlog veka, Brentov paket potprograma MP u Fortranu 66 za realizaciju aritmetičkih operacija i elementarnih funkcija višestruke preciznosti (AVP) iz 1978. godine ([13], [14]) je prvi uspešan paket u kome su

otklonjeni nedostaci prethodnih programa kao što su: zavisnost od računara na kome se primenjuje paket, korišćenje fiksne umesto aritmetike sa pokretnom tačkom, ograničena preciznost (gubitak značajnih cifara), odsustvo potprograma za izračunavanje elementarnih i specijalnih funkcija (ln, exp, sin, Beselove funkcije, itd.) i konstanti (π, γ). MP je radio sa nizovima celobrojnih brojeva od t cifara sa osnovom β, gde je $t \geq 2$ i $\beta \geq 2$, i operisao nad njim koristeći celobrojnu (intedžersku) aritmetiku. Sastojao se od 101 potprograma i 4 glavna programa a interfejs[8] između samog paketa i Fortranovog prekompajlera Augment je opisan u radu [17]. Detalji kao i liste ovih potprograma mogu se naći u radovima [13] i [14].

Kasnije je Bailey [4] realizovao svoju AVP u paketu nazvanom MPFUN, koji se sastojao od oko 10000 linija u Fortranu 77 u okviru 87 potprograma. U MPFUN-u *vp* broj je vektor PT brojeva u jednostrukoj preciznosti predstavljenih sa osnovom 2^{24} (za IEEE aritmetiku). Osnovne aritmetičke operacije i standardne elementarne funkcije su izvođene sa preciznošću do 1000 značajnih cifara. Po prvi put su primenjeni iterativni metodi za rešavanje nelinearnih jednačina pri realizaciji deljenja i nalaženju kvadratnog korena i korišćena brza Furijeova transformacija pri množenju.

U nastavku dajemo listu nekih softverskih paketa i biblioteka koji rade sa aritmetikom višestruke ili proizvoljne preciznosti. Napominjemo da ovo nije definitivna lista, već samo pregled najčešće korišćenih sofvera u ovom momentu. Intenzivan rad na poboljšanju postojećih i kreiranju novih softverskih paketa sa AVP stalno menja efikasnost i rang ovih paketa. Neke od najvažnijih karakteristika navedenih sofverskih paketa opisane su u radovima i priručnicima za upotrebu [39], [47] i [57].

- **GNU MP (GMP)** (`http://gmplib.org`) spada među najnaprednije biblioteke za aritmetiku proizvoljne preciznosti i napisan je u jeziku C. GNU je kompletan operativni sistem sličan Unix-u koji se besplatno distribuira. Glavni koordinator projekta od 1991.

- **CLN** (Class Library for Numbers) (`http://www.ginac.de/CLN/`) je biblioteka za efikasno izračunavanje i radi sa svim vrstama brojeva sa proizvoljnom preciznošću. Napisana je u C++ a distribuira se pod

[8] Ovaj termin je odomaćen u računarskim naukama i označava interakciju između komponenti, bilo hardverskih (npr, grafička kartica), bilo softverskih (npr. internet pretraživači), preko ulaznog/izlaznog sistema i pridruženog protokola.

GNU licencom. Njen tvorac je Bruno Haible, a sada je održava Richard Kreckel.

- **MPFR** (`http://www.mpfr.org`) je biblioteka binarnih brojeva sa pokretnom tačkom višestruke preciznosti, koja se distribuira pod licencom GNU LGPL (Lesser General Public Licence). Ovaj sofver koristi većinu instrukcija i ideja ugrađenih u IEEE 754 standard i primenjuje ih na aritmetiku proizvoljne preciznosti tako što obezbeđuje *pravilno zaokruživanje* i *signalne instrukcije.*

- **MPC** (`http://www.multiprecision.org/`) je biblioteka programskog jezika C za aritmetiku koja koristi kompleksne brojeve sa proizvoljno velikom preciznošću i *pravilnim zaokruživanjem.*

- **MPFI** (`http://mpfi.gforge.inria.fr/`) je paket za intervalnu aritmetiku sa pokretnom tačkom proizvoljne preciznosti, zasnovan na MPFR. Glavna prednost ovog softvera je obezbeđivanje precizne granice greške pri izračunavanjima ili rešavanju jednačina uz pomoć intervalne aritmetike.

- **ARPREC** (`http://crd.lbl.gov~dhbailey/mpdist/`) je programski paket za APT višestruke preciznosti koji su razvili David Bailey i saradnici u C++/Fortran-u. Uključuje interaktivnu AVP *The Experimental Mathematician's Toolkit.*

Pomenimo i nekoliko programskih paketa numeričke algebre[9] opšte namene koji uključuju aritmetiku višestruke ili proizvoljne preciznosti. *Mathematica* je najpopularniji i najviše korišćen softverski paket za numeričku algebru i matematiku u Srbiji. Korisne informacije i primene mogu se naći u knjigama na srpskom [65] i [81]. Najefikasniji i naviše korišćeni programski paketi numeričke algebre koji rade sa aritmetikom višestruke preciznosti dati su u tabeli 1.6.

[9]U engleskom jeziku se koristi termin *computational algebra*, ali smatramo da je termin numerička algebra više u duhu srpskog jezika nego računska algebra, koji više podseća na aritmetiku.

Program	Kreator	Početak razvoja	Prva javna realizacija	Poslednja verzija
Magma	University of Sidney	1990	1993	2011(2.17)
Maple	University of Waterloo	1980	1984	2010(14)
Mathematica	Wolfram Research	1986	1988	2011(8.01)
PARI/GG	Henri Cohen +	1985	1990	2011(2.5.0)
Sage	William Stein	2005	2005	2011(4.7)
Axiom	Tim Daly	1971	2002	2011
MathEclipse/Symia	Axel Kramer	2002	2002	2007
Maxima	MIT Project MAC	1967	1998	2011(5.24)
SymPy	Ondrej Čertik	2006	2007	2011(0.7.1)
Xcas	Bernard Parisse	2004	2008	2010(0.8.1)
Yacas	Ayal Pinkus +	1998	–	2007(1.2.2)

Tabela 1.6 Karakteristike programskih paketa u AVP

1.5 Standard IEEE 754-2008 – poboljšanja i perspektive

IEEE 754-1985 standard predstavljao je veliki napredak i bio je implementiran na mnogim platformama od naučnog i komercijalnog značaja. Ipak, samo 15 godina nakon njegove distribucije, javila se jasna potreba za njegovom revizijom.

- neke osobine je trebalo standardizovati: na primer, četvorostruku („quad") preciznost (tj. 128-bitnu binarnu reprezentaciju) i novi tzv. SPOJEN POMNOŽI-I-DODAJ operator, kraće SPD.[10]

- Posle 1985. godina razvijeni su mnogi napredni algoritmi koji su omogućavali izračunavanja koja su ranije bila veoma komplikovana i praktično ili neizvodljiva ili vrlo „skupa." Tipični primeri su konverzija brojeva iz binarnog u dekadni sistem sa izlaznim rezultatom velike tačnosti i biblioteke elementarnih funkcija (uključujući transendentne) sa pravilnim zaokruživanjem i velikom tačnošću.

- Postojale su izvesne nejasnoće u IEEE 754-1985 standardu. Na primer, nije bilo jasno u kom unutrašnjem formatu je trebalo predstavljati implicitne međuvrednosti i promenljive.

[10]Poznat kao FMA operator (od engl. fused muliply-and-add).

Radeći na standarizaciji i poboljšanju karakteristika PT sistema, od kojih su neke navedene, IEEE 754-1985 standard bio je revidiran u periodu od 2000. do 2006. i prihvaćen juna 2008. Pri svemu tome moralo se voditi računa da hardver i dalje podržava stari IEEE 754-1985 standard. Revidirani standard radi sa osnovama 2 ili 10 i sa graničnim opsezima ekponenata vezanim realacijom $e_{\min} = 1 - e_{\max}$ za oba formata. Glavni parametri binarnog i dekadnog formata su dati u tabelama 1.7 i 1.8.

Parametar	binarni 16	binarni 32	binarni 64	binarni 128
p	11	24	53	113
$e_{\min}$	+15	+127	+1023	+16383
$e_{\max}$	−14	−126	−1022	−16382

Tabela 1.7 Glavni parametri binarnih promenljivih formata za IEEE 754-2008 standard

Parametar	dekadni 32	dekadni 64	dekadni 128
p	7	16	34
$e_{\min}$	+96	+384	+6144
$e_{\max}$	−95	−383	−6143

Tabela 1.8 Glavni parametri dekadnih promenljivih formata za IEEE 754-2008 standard

Jedno od suštinskih poboljšanja IEEE 754-2008 standarda je rad sa dve vrste formata:

- *promenljivi formati*, gde je kodiranje potpuno specificirano i koje dozvoljava razmenu podataka između platformi;
- *proširen* i *proširiv* format preciznosti, čije kodiranje nije specificirano, ali je u saglasnosti sa promenljivim formatom.

Proširen format preciznosti omogućuje povećanje preciznosti u odnosnu na osnovne formate prikazane u tabelama 1.9 i 1.10. Osnovna ideja je da se na ovaj način međurezultati izračunavaju sa većom preciznošću a da finalni rezultat bude dat u osnovnim formatima. Ovi formati rade sa nizovima[11] od

[11]Konačan niz ili reč (engl. *string/word*), u opštem slučaju *string* može označavati niz simbola, slova, brojeva.

256 i 1024 bita. Proširena preciznost omogućuje da krajnji rezultati budu generalno tačniji u odnosu na one koji bi se dobili korišćenjem samo osnovnih formata, a takođe da elimiše pojavu rezultata van opsega PT sistema (potkoračenje i prekoračenje).

Proširiv format preciznosti je format kod koga su preciznost i opseg definisani kontrolnim programom ili korisnikom. Dovoljno je u programskom jeziku koji podržava ovaj format specificirati preciznost p i $e_{\max}$ (pri čemu je $e_{\min} = 1 - e_{\max}$).

SPD (SPOJENA POMNOŽI-I-DODAJ) instrukcija omogućuje izračunavanje izraza $a \cdot b + c$, gde su a, b i c PT brojevi, sa samo jednim zaokruživanjem na kraju, tj., ona izračunava $\square(a \cdot b + c)$, gde je $\square$ mod zaokruživanja. SPD instrukcija omogućuje brže i tačnije izračunavanje skalarnog proizvoda, matričnog množenja i polinoma. Ona je takođe korisna u konstrukciji algoritama za *pravilno* zaokruživanje za operacije deljenje, korenovanje i za redukciju oblasti argumenta pri izračunavanju trigonometrijskih funkcija, videti [84]. SPD je uvedena 1990. godine na IBM RS/6000 procesoru a kasnije na mnogim komercijalnim procesorima opšte namene kao što su IBM Power PC, HP PA-8000 i HP/Intel Itanium. SPD je uključena u novi IEEE 754-2008 standard za aritmetiku sa pokretnom tačkom.

IEEE 754-2008 standard zahteva da elementarne funkcije, date na početku Odeljka 1.4, budu *pravilno* zaokružene. IEEE 754-2008 standard takođe zahteva *pravilno* zaokruživanje za pet osnovnih aritmetičkih operacija: sabiranje, oduzimanje, množenje, deljenje i kvadratni koren. Detalji primene PT operacija dati su na slici 1.9.

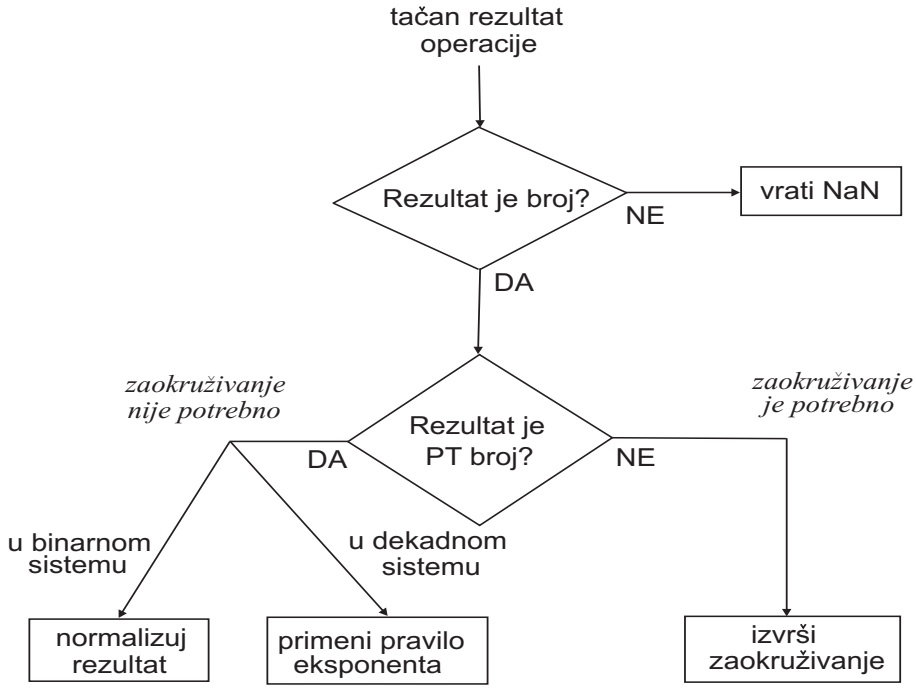
- Ako je rezultat nedefinisan, vraća se poruka NaN (Not a Number).
- Ako je rezultat broj, tada
  - Ako se ovaj rezultat može predstaviti kao PT broj, zaokruživanje nije potrebno. Međutim, zbog različitog predstavljanja brojeva u PT sistemu sa osnovom 2 i 10, potrebno je izvršiti dodatni posao:
    * U binarnom sistemu postoji samo jedna ispravna reprezentacija i to je ona sa najmanjim mogućim eksponentom (normalizacija).

    * U dekadnom sistemu postoji nekoliko načina predstavljanja brojeva koji se nazivaju *kohorte*. Standard precizno definiše koji član kohorte treba da bude izabran. Za svaku operaciju je definisan tzv. *prioritetni eksponent* kao funkcija ulaznih podataka operacije, a izlazni rezultat je onaj član kohorte čiji eksponent je najbliži prioritetnom eksponentu.
  - Ako se tačan rezultat ne može predstaviti u obliku PT broja, on mora biti zaokružen na PT broj. Ako taj PT broj ima više reprezentacija, u binarnom ili dekadnom sistemu, bira se ona reprezentacija koja ima najmanji mogući eksponent.

- Pri sabiranju, oduzimanju ili množenju dva PT broja, ili kada se sprovodi SPOJENA POMNOŽI-I-DODAJ (SPD) operacija, rezultat beskonačne preciznosti je uvek konačan i može biti tačno izračunat primenom postupka prikazanog na slici 1.10.

- Kada se sprovodi deljenje ili izračunavanje kvadratnog korena ili elementarne funkcije, tačan rezultat može imati beskonačan broj cifara; jednostavan primer je deljenje 1.0/3.0. U ovakvim slučajevima neophodno je primeniti konačno izračunavanje uz korišćenje postupka zaokruživanja. Za deljenje se može izračunati količnik u konačnoj preciznosti, a zatim se primenjuje odgovarajući postupak zaokruživanja na ostatak pri deljenju. Slično, za kvadratni koren može se izračunati $y \approx \sqrt{x}$, a zatim se primenjuje zaokruživanje na $x - y^2$.

Za izvršavanje osnovnih aritmetičkih operacija u binarnom kodu, IEEE standard, znatno poboljšan 1985. godine, davao je veoma dobre rezultate i, kao takav, bio je široko prihvaćen i korišćen u skoro svim procesorima tog vremena. Međutim, računarska tehnologija dramatično je napredovala od 1985. Brzina izvršavanja aritmetičkih operacija uvećavala se od megaflopsa (10^6 PT operacija u sekundi) do gigaflopsa (10^9) i teraflopsa (10^{12}) i u današnje vreme približava se petaflopsu (10^{15}). Poboljšanja nisu vršena samo u pogledu brzine već i kvalitativno. U vreme megaflops računara konvencionalna analiza greške bila je sastavni deo analize skoro svakog numeričkog algoritma. Danas je za giga i teraflops računare analiza greške ne samo skupa već često i nemoguća. Na primer, ako bi brojevi dobijeni u toku jednočasovnog izračunavanja bili štampani dvostrano u $A4$ formatu, stvorila bi se gomila papira koja bi dosegla visinu jednaku rastojanju od Zemlje do Sunca.



Slika 1.9 Implementacija PT operacija

Ogroman broj operacija izvršenih u jedinici vremena logično je doveo do pitanja da li dobijeni rezultati na računaru zaista rešavaju postavljeni problem, tj. da li je dobijeni rezultat tačan (i kako ga proveriti). Jedini način da se odgovori na ovo pitanje i izvrši provera rezultata je korišćenje samog računara. Ustvari, moć i kvalitet računara ne smeju se posmatrati samo na osnovu brzine izvršavanja operacija, već se mora uzeti u obzir da li se traženi rezultat može dobiti sa garantovanom tačnošću do 5, 10 ili 15 decimalnih cifara. Ako se pitanje postavi na ovaj način, to samo može dovesti do daljeg poboljšanja performansi računara. Matematički metodi već danas su dovoljno razvijeni da mogu da odgovore na postavljena pitanja i zahteve u najvećem broju problema, ali u ovom momentu još uvek nisu u dovoljnoj meri implementirani u softver i hardver savremenih računara.

U poslednjih 20 do 30 godina razvijen je veliki broj matematičkih metoda koji dozvoljavaju računaru da *sam* proveri tačnost dobijenog rezultata. Ovi metodi daju intervale koji sadrži rezultate koji se traže i često koriste iterativni tehniku u cilju smanjenja dužina ovih rezultujućih intervala. Ovakav pristup doveo je na prirodan način do potrebe za uvođenjem intervalne arit-

metike kao računarske aritmetike, o čemu će biti reči u Poglavlju 2. Pomenuti *samoproveravajući* metodi u najvećem broju praktičnih problema ne zahtevaju aritmetiku velike preciznosti; dvostruka preciznost je uglavnom sasvim dovoljna.

Osim intervalne aritmetike, neophodne za kontrolu izračunatog rezultata, potrebne su i sledeće dve veoma važne aritmetičke osobine:

(I) **brza hardverska podrška intervalnoj aritmetici;**

(II) **brza i tačna „pomnoži-i-dodaj" operacija, ili, što je ekvivalentno, tačan skalarni proizvod.**

Hardverska realizacija intervalne aritmetike (svojstvo (I)) još uvek nije dovoljno brza; izračunavanje u intervalnoj aritmetici je 3 do 5 puta sporije nego u slučaju obične PT aritmetike. Razlog u velikoj meri leži u sprovođenju postupka zaokruživanja kod komercijalnih procesora, koje se vrši na početku izračunavanja. S druge strane, realizacija intervalne aritmetike je zasnovana na izračunavanju donje granice putem *silaznog zaokruživanja* (bit manje) i gornje granice putem *uzlaznog zaokruživanja* (bit više), i ovo je potrebno raditi u svakoj operaciji, što nije slučaj kod procesora koji su danas u upotrebi.

U vezi sa svojstvom (II), da bi se dobile što uže granice rešenja ili rezultata izračunavanja, intervalna aritmetika mora da bude kombinovana sa tehnikama (najčešće iterativnim „rafiniranjem") koje zahtevaju tačnu SPOJEN-POMNOŽI-I-DODAJ (SPD) instrukciju, ili, što je ekvivalentno, tačan skalarni proizvod. Ovo se realizuje korišćenjem dvostruke dužine u fiksnom dovoljno širokom (u memorijskom smislu) registru. Na ovaj način je potpuno eliminisana greška odsecanja. Brza i tačna SPD instrukcija omogućuje brzu aritmetiku višestruke preciznosti. Broj u višestrukoj preciznosti predstavljen je kao uređena grupa PT brojeva u regustru dovoljne dužine. Njegova vrednost je zbir njegovih komponenata. Sabiranje i oduzimanje se jednostavno izvodi u ovako dizajniranom registru, a množenje je suma parcijalnih proizvoda PT brojeva.

Tradicionalna analiza greške numeričkih algoritama je zasnovana na oceni grešaka pojedinih aritmetičkih operacija i prostiranju ovih grešaka kroz manje ili više složen algoritam. Problemi koji se danas rešavaju pomoću računara zahtevaju ogroman broj operacija koje moderan hardver izvršava veoma brzo, nekoliko biliona PT operacija u sekundi. U ovom slučaju pomenuta analiza

greške je bespredmetna jer korisnik nije u mogućnosti da prati i kontroliše ogroman broj međurezultata.

Usled navedenih razloga aritmetika sa pokretnom tačkom mora biti tako dizajnirana (softverski i hardverski) da isključi bilo kakve greške počev od jednostavnih izračunavanja pa sve do vrlo složenih algoritmima. Jedan od efikasnih načina koji je danas aktuelan je uvođenje *pete* aritmetičke operacije, tačnog skalarnog proizvoda realizovanog preko SPD instrukcije. Postoje dva razloga za to: 1) skalarni proizvod se veoma često sreće kod numeričkih algoritama tako da se može smatrati centralnom i fundamentalnom „operacijom" numeričke analize; 2) skalarni proizvod se može uvek izračunati potpuno tačno bez velikog „tehničkog" napora pri realizaciji hardvera. Korišćenje skalanog proizvoda ima sledeće prednosti:

- međurezultati ne moraju da se smeštaju u memoriju računara;

- zaokruživanje i normalizacija međurezultata ne mora da se vrši;

- *prekoračenje* i *potkoračenje* se ne javlja u međurezultatima;

- gubutak značajnih cifara pri oduzimanju bliskih brojeva je izbegnut;

- analiza greške je irelevantna;

- omogućuje realizaciju vrlo brze aritmetike višestruke preciznosti ili aritmetike sa promenljivom preciznošću;

- rezultat ne zavisi od reda po kome se sabirci dodaju;

- ubrzanje konvergencije nekih numeričkih algoritama;

- dobijeni rezultat je uvek tačan ako se koriste PT brojevi.

U matematici postoji velika razlika da li je izračunavanje tačno za veliki broj (ili za najveći broj) ulaznih podataka ili za *sve* podatke! To isto važi i za neke sofisticirane probleme u tehničkim disciplinama kada i najmanja greška može dovesti do katastrofalnih posledica. Za probleme sa perturbovanim (netačnim) podacima (dobijenim, na primer, merenjem), intervalna aritmetika je pogodno oruđe jer radi sa intervalima koji izumaju u obzir sve greške. S druge strane, tačan skalarni proizvod proširuje zahteve IEEE standarda u pogledu tačnosti i to ne samo pri radu sa realnim brojevima, već i sa kompleksnim brojevima, vektorima, matricama i njihovim intervalnim proširenjima. Ukratko, dok intervalna aritmetika omogućuje izračunavanja i rešavanje problema sa garantovanim granicama greške, tačan skalarni proizvod donosi brzinu, dinamičku preciznost i tačnost. Kombinacija

intervalne aritmetike i APT sa tačnim skalarnim proizvodom kao petom operacijom je ono što je potrebno savremenom računaru.

Zbog navedenih osobina, intervalna aritmetika i brza i tačna hardverska podrška moraju biti dodate konvencionalnoj PT aritmetici u najskorijoj budućnosti kao neophodna proširenja i poboljšanja. Umesto da računari budu samo sredstva za vrlo brza izračunavanja, oni treba da postanu važan naučni instrument. Premda je potrebno više računarskog vremena i napora da bi se dobili verifikovani rezultati (sa kontrolisanim granicama greške), pouzdanost i tačnost izračunavanja su od primarnog značaja kada se radi o kritičnim aplikacijama. Na primer, vrlo ozbiljni, čak tragični incidenti se mogu javiti ako su sopstvene frekvencije moćnog električnog generatora pogrešno izračunate, ili je nuklearna eksplozija nekorektno simulirana. Vrlo je svež primer Londonskog milenijumskog pešačkog mosta koji je juna 2000. zatvoren samo dva dana posle svečanog otvaranja zbog mogućnosti rušenja usled vibracija koje izazivaju pešaci dovodeći do rezonantog ponašanja. Sa matematičkog gledišta, rezonantne frekvencije se opisuje sopstvenim vrednostima. U slučaju Milenijumskog mosta najmanja sopstvena vrednost modela je pogrešno izračunata i proglašena nulom a time i zanemarena, što je moglo da ima kobne posledice. Slični slučajevi sa katastrofalnim posledicama su rušenje Takomi mosta u američkoj saveznoj državi Vašington (usled rezonanse izazvane vetrom) i Brajton mosta u Engleskoj (zbog rezonanse izazvane marširanjem vojske).

2

Realna intervalna aritmetika i intervalna D-aritmetika

Ovo poglavlje je posvećeno intervalnoj aritmetici i intervalnim matricama. Razlog za njihovu upotrebu je mogućnost kontrole tačnosti rezultata i rad sa približnim podacima za koje se jedino zna da leže u nekim intervalima. Takav model upravo je primenjen u 3. poglavlju disertacije u delu koji se odnosi na matrične modele procesnih iteracija. Najbolji rezultati postižu se primenom kompleksne intervalne aritmetike u dijametarskoj formi. Zbog toga je u Odeljku 2.8 realizovan softver (u C #) koji omogućuje rad sa dijametarskom aritmetikom i izračunava elementarne kompleksne funkcije u dijametarskoj aritmetici. U ovom poglavlju dat je i kratak opis softvera INTLAB za intervalnu aritmetiku (realnu i kompleksnu sa centriranim diskovima). Razvijeni softver za dijametarsku aritmetiku kao originalni doprinos i INTLAB su kombinovano korišćeni u realizaciji primera u Odeljku 3.7.

2.1 Razvoj i značaj intervalne aritmetike

Korišćenje intervala i intervalnih metoda je novijeg datuma i pojavilo se početkom šezdesetih godina dvadestog veka sa pojavom računara. Diskretna priroda kompjuterske aritmetike nameće potrebu procene preciznosti podataka i greške zaokruživanja. Kako je praktično svaka realna brojna veličina u računaru prikazana približno pomoću najbližeg mašinskog broja (tj. pripada nekom intervalu čiji je poluprečnik jednak kompjuterskoj preciznosti), uvođenje intervalne aritmetike kao kompjuterske aritmetike potpuno je pri-

rodno. Ovo takođe ima svoje opravdanje i u činjenici da mnoge veličine ne znamo precizno već samo u okviru preciznosti s kojom one mogu biti izmerene ili izračunate.

U cilju kontrole grešaka zaokruživanja američki matematičar Ramon Mur,[1] radeći na Lokidovom svemirskom programu u Palo Altu (Kalifornija, SAD), uveo je intervalnu aritmetiku i razvio prve metode koji rade sa intervalima [99]. Dalji razvoj intervalne analize, kako se u početku zvala ova nova oblast primenjene matematike i računarskih nauka, usledio je posle pojave Murove monografije *Interval Analysis* 1966. godine [97], napisane na osnovu rezultata njegovih istraživanja i doktorske disertacije odbranjene na Stenfordu 1963. godine [96]. Murova nova monografija [98] iz 1979. godine donela je nove ideje i ubzala razvoj intervalne matematike. K. Nikel (Univerzitet u Frajburgu, Nemačka) i U. Kuliš (Univerzitet u Karlsrueu) i njihovi saradnici stvorili su neku vrstu „škola intervale matematike" u ovim univerzitetskim centrima, što je dovelo do velike ekspanzije ove discipline.

Pogrešno je intervalnu aritmetiku shvatiti samo kao prostu zamenu realnih brojeva realnim intervalima. U nekim, mada retkim situacijama, koristi se i ovaj prilaz mada u najvećem broju slučajeva daje kao finalni rezultat suviše širok interval od malog praktičnog značaja. Pravi pristup je razvoj intervalnih metoda koji daju dovoljno uske intervale pri rešavanju problema koji se mogu razdvojiti na dve klase: probleme sa „netačnim" i „tačnim" podacima. Prva klasa sadrži problem rešavanja jednačine čiji podaci variraju unutar izvesnih intervala, pri čemu je cilj nalaženje što je moguće užeg intervala koji sadrži traženo rešenje. Primer ovakve vrste problema je rešavanje sistema linearnih jednačina čiji koeficijenti mogu da variraju unutar intervala. U klasi problema gde su dati tačni podaci, intervalna analiza uglavnom služi da razvije metode koji generišu konvergentne nizove intervala koji u toku iterativnog procesa 1) postaju dovoljno mali i 2) sadrže tačno rešenje. Na ovaj način se dobija informacija o gornjoj granici greške sa kojom je izračunata aproksimacija.

Kontrola tačnosti rezultata je najznačajnija prednost intervalne aritmetike koja do izražaja dolazi naročito ako se radi u aritmetici višestruke preciznosti jer u tom slučaju klasičnu analizu greške nije moguće sprovesti. Ova *osobina inkluzivnosti* dovela je do široke primene intervalne aritmetike u najrazličitijim oblastima teorije i prakse, kao što su numerička analiza, opti-

[1] Ramon Moore (1929–), Sakramento (SAD).

mizacija, simulacija procesa sa promenljivim parametrima, inženjerske discipline, proučavanje stabilnosti sistema, finansije, geodetska istraživanja, opis haotičnih fenomena, pa čak i proračun svemirskih letova. U poslednje vreme intervalna matematika i samoproveravajući algoritmi koriste se u dokazivanju teorema pomoću računara (computer assisted proofs). Pomenimo samo dokaz čuvene Keplerove hipoteze stare blizu 400 godina koju je 2006. godine dokazao Tomas Hejlz (Hales) sa Univerziteta u Pitsburgu korišćenjem metoda globalne optimizacije, linearnog programiranja i intervalne aritmetike. Uopšte, primena intervalne aritmetike uvek je dobrodošla pri rešavanju problema u kojima se javljaju neizvesne promenljive (koje, na primer, leže u nekim intervalima), pri proračunima u prisustvu greške zaokruživanja, a takođe i u onim slučajevima gde je bitna informacija o gornjoj granici greške sa kojim je pronađeno neko rešenje ili rezultat izračunavanja.

Nedostatak intervalne aritmetike u ranijem periodu ogledao se u sporom izvršavanju intervalnih operacija u poređenju sa APT. Međutim, razvoj novih metoda je omogućio skoro u potpunosti prevazilaženje ovog problema. To je postignuto korišćenjem tačne POMNOŽI-I-DODAJ instrukcije ili, što je ekvivalentno, tačnog skalarnog proizvoda, videti [82]. Ako se POMNOŽI-I-DODAJ instrukcija implementira hardverski, tada intervalna aritmetika postaje još brža. Podaci se mogu memorisati i transferisati kao PT brojevi u dvostrukoj preciznosti. Generalno, intervalna aritmetika obezbeđuje kontrolu granica grešaka u izračunavanjima (garanciju tačnog rezultata pomoću inkluzivnih intervala koji sadrže rezultat), dok POMNOŽI-I-DODAJ instrukcija donosi tačnost i višestruku (dinamičku) preciznost.

2.2 Realna intervalna aritmetika

U ovom odeljku najpre će biti razmotrena realna intervalna aritmetika, koja je osnova za intervalne aritmetike implementirane na računarima u jednostrukoj i višestrukoj preciznosti. Ova aritmetika takođe služi za kompleksnu intervalnu aritmetiku koja radi sa pravougaonicima u kompleksnoj ravni kao i za kreiranje biblioteka intervalnih operacija i intervalnih funkcija, kao što je **Intlab** opisan ukratko u odeljku 2.4 i korišćen u 3. poglavlju za matrične modele procesnih iteracija.

Glavni doprinos u ovom poglavlju je softver za optimalnu (dijametarsku) kružnu kompleksnu aritmetiku koja radi sa diskovima. Programi su napisani

u programskom jeziku C++ na osnovu formula koje su date u odeljku 2.7 i Prilogu A.

Definicija 2.1. Podskup skupa realnih brojeva $\mathbb{R}$ oblika

$$A = [a_1, a_2] = \{x \,:\, a_1 \leq x \leq a_2,\ a_1, a_2 \in \mathbb{R}\}$$

zove se zatvoren *realan interval.* Skup svih zatvorenih realnih intervala označavamo sa $I(\mathbb{R})$. U degenerativnom slučaju kada je $a_1 = a_2$, ovaj interval se zove *tačka-interval.* ▲

Definicija 2.2. Dva intervala $A = [a_1, a_2]$ i $B = [b_1, b_2]$ su jednaka ako su jednaka u skupovnom smislu. Na osnovu toga sledi da je

$$A = B \Leftrightarrow a_1 = b_1 \wedge a_2 = b_2. \text{ ▲}$$

Ako je $*$ jedna od binarnih operacija $+, -, \cdot, :$ na skupu realnih brojeva, osnovne intervalne aritmetičke operacije na skupu $I(\mathbb{R})$ definišu se pomoću

$$A * B = \{x = a * b \,:\, a \in A \wedge b \in B\}, \quad A, B \in I(\mathbb{R}), \tag{2.1}$$

podrazumevajući da $0 \notin B$ u slučaju deljenja.

Neka su $A = [a_1, a_2]$ i $B = [b_1, b_2]$ dva realna intervala. Osnovne intervalne operacije između dva intervala mogu se eksplicitno izračunati na sledeći način:

$$\begin{aligned}
&A + B = [a_1 + b_1, a_2 + b_2],\\
&A - B = [a_1 - b_2, a_2 - b_1],\\
&A \cdot B = [\min\{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}, \max\{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}],\\
&A : B = [a_1, a_2] \cdot [1/b_2, 1/b_1] \quad \text{ako } 0 \notin B.
\end{aligned}$$

Operacije oduzimanja i deljenja u skupu realnih intervala $I(\mathbb{R})$ nemaju inverzni element za razliku od skupa realnih brojeva $\mathbb{R}$. Na primer,

$$\begin{aligned}
&[1, 3] - [1, 3] = [-2, 2],\\
&[3, 4] : [3, 4] = [3/4, 4/3].
\end{aligned}$$

Lako se pokazuje da u opštem slučaju važi $0 \in A - A$ i $1 \in A : A$.

Distributivni zakon u opštem slučaju ne važi za intervalnu aritmetiku. Umesto toga, važi tzv. *subdistributivni zakon* izražen na sledeći način

$$A(B+C) \subseteq AB + AC \quad \text{za } A, B, C \in I(\mathbb{R}).$$

Sabiranje i množenje u $I(\mathbb{R})$ su komutativne i asocijativne operacije. Jedna od fundamentalnih osobina intervalne aritmetike je *inkluzivna izotonost*, data u sledećoj teoremi:

Teorema 2.1. *Neka* $A_k, B_k \in I(\mathbb{R})$, $k = 1, 2$, *i neka* $* \in \{+, -, \cdot, :\}$. *Tada važi*

$$A_k \subseteq B_k \ (k = 1, 2) \ \Rightarrow \ A_1 * A_2 \subseteq B_1 * B_2. \ \blacktriangle$$

Dokaz se može naći u knjizi [2, Pogl. 1]. Opštije, ako je $F(X_1, \ldots, Z_n)$ *racionalan* izraz sa intervalnim promenljivama $X_1, \ldots, X_n$, tada je

$$X_1' \subseteq X_1, \ldots, X_n' \subseteq X_n \quad \Rightarrow \quad F(X_1', \ldots, X_n') \subseteq F(X_1, \ldots, X_n)$$

za svaki skup intervala za koji su intervalne aritmetičke operacije u F definisane (videti Teoremu 3.1 u [97]).

Deljenje nulom je nedozvoljena („invalid") operacija kojom prilikom se prekida proces izračunavanja, pri čemu softver primenjen na računaru signalizira deljenje nulom. Međutim, pokazalo se da u nekim primenama intervalne matematike deljenje intervalom koji sadrži nulu ne mora da bude razlog za prekid izračunavanja. Tipičan primer je proširenI Njutnov intervalni metod za nalaženje intervala koji sadrži traženu nulu funkcije. Ovo se postiže dodefinisanjem operacije deljenje uključujući i slučaj deljenja nula-intervalom.

Polazeći od definicije osnovnih intervalnih operacija (2.1) deljenje u $I(\mathbb{R})$ se definiše sa

$$A/B := \{a/b \ : \ a \in A \wedge b \in B\}. \tag{2.2}$$

Količnik a/b je definisan kao inverzna operacija množenja, tj., kao rešenje jednačine $b \cdot x = a$. Odavde, (2.2) se može napisati u obliku

$$A/B := \{x \ : \ bx = a \wedge a \in A \wedge b \in B\}. \tag{2.3}$$

Za $0 \notin B$ relacije (2.2) i (2.3) su ekvivalentne. Dok u $\mathbb{R}$ deljenje nulom nije definisano, relacija (2.3) dozvoljava interpretaciju rezultata za $0 \in B$.

Neka su $A = [a_1, a_2]$ i $B = [b_1, b_2]$ intervali takvi da $0 \in B$, tada se pomoću (2.2) „degenerativno" deljenje definiše na sledeći način,

$$(A/B)_0 := A \cdot \left(\frac{1}{B}\right)_0, \tag{2.4}$$

gde simbol „0" označava da imenilac može biti interval koji sadrži nulu, pri čemu je recipročna vrednost definisana sa

$$\left(\frac{1}{B}\right)_0 := \begin{cases} \left[\frac{1}{b_2}, \frac{1}{b_1}\right] & \text{ako } 0 \notin B, \\ \left[\frac{1}{b_2}, +\infty\right] & \text{ako je } b_1 = 0, \\ \left[-\infty, \frac{1}{b_1}\right] & \text{ako je } b_2 = 0, \\ \left[-\infty, \frac{1}{b_1}\right] \bigcup \left[\frac{1}{b_2}, +\infty\right] & \text{ako je } b_1 b_2 < 0, \text{ tj. } 0 \in B. \end{cases} \tag{2.5}$$

Intervalna arimetika u kojoj je deljenje definisano pomoću (2.4)–(2.5) naziva se *proširenom intervalnom aritmetikom*. Pojava unije dva intervala kod proširenog intervalnog Njutnovog metoda jednostavno znači da su izolovane *dve* nule funkcije i proces nalaženja njihovih inkluzivnih intervala treba nastaviti u okviru pronađenih oblasti. Prošireni intervalni Njutnov metod snabdeven proširenom intervalnom aritmetikom postaje globalno konvergentan, za razliku od običnog Njutnovog metoda koji se prekida u slučaju deljenja nulom. Više detalja o proširenom Njutnovom intervalnom metodu može se naći u knjizi [82, Pogl. 9].

Proizvoljan interval $A = [a_1, a_2]$, gde su a_1 i a_2 proizvoljni realni brojevi, ne može se uvek predstaviti na računaru ako a_1 i a_2 nisu PT brojevi. Zbog toga se vrši zaokruživanje broja a_1 *ka manjem PT broju*, u oznaci $\triangledown a_1$, i zaokruživanje broja a_2 *ka većem PT broju*, u oznaci $\triangle a_2$. Ovde simboli $\triangledown$ i $\triangle$ ukazuju na zaokruživanje ka manjem, odnosno zaokruživanje ka većem PT broju. Na ovaj način dobija se *zaokružujući interval* $\square A = [\triangledown a_1, \triangle a_2]$ koji uvek sadrži interval A, tj. $\square A \supseteq A$. Tako predstavljen interval koristi se u algebarskim operacijama uz primenu pomenutih modova zaokruživanja. Na primer,

Sabiranje: $C := [\triangledown a_1 \boxplus \triangle a_2, \triangledown a_2 \boxplus \triangle b_2]$.

Oduzimanje: $C := [\triangledown a_1 \boxminus \triangle b_2, \triangledown a_2 \boxminus \triangle b_1]$.

Ovde simboli $\triangledown$, $\triangle$, $\triangledown$, $\triangle$ ukazuju da se operacije sabiranja i oduzimanja vrše sa odgovarajućim modovima zaokruživanja. Operacije množenja i deljenja u zaokružujućoj aritmetici date su u [82].

Intervalna aritmetika višestruke preciznosti koristi predstavljanje intervala preko niza skalara - PT brojeva i intervala $I = [i_d, i_g]$ sa donjom i gornjom granicom i_d i i_g aritmetike sa pokretnom tačkom. Tada se element

$$X = \sum_{j=1}^{n} x_j + I \tag{2.6}$$

zove *interval dužine* n, x_j su *komponente* od X i I je *intervalna komponenta*. U gornjoj sumi dozvoljeno je da n bude 0 i tada je $X = I$ upravo interval sa granicama standardne aritmetike sa pokretnom tačkom. Kao i u APT višestruke preciznosti, u reprezentaciji (2.6) poželjno je da se komponente ne preklapaju jer se time sprečava gubitak značajnih cifara. Osnovne operacije sa intervalima oblika (2.6) navedene su u knjizi [82].

Primer 2.1. Izračunajmo vrednost izraza $x = 1 - (1/3) \times 3$. U realnoj („tačnoj") aritmetici je $x = 0$. U APT sa n značajnih decimalnih cifara dobija se

$$x(n) = 1 - 0.\overbrace{333\ldots33}^{n} \times 3 = 1 - 0.\overbrace{999\ldots99}^{n} = 0.\overbrace{000\ldots0}^{n-1}1 > 0.$$

U zaokružujućoj intervalnoj aritmetici dobija se interval

$$\begin{aligned} X(n) &= 1 - \left[\, 0.\overbrace{333\ldots33}^{n},\ 0.\overbrace{333\ldots3}^{n-1}4 \,\right] \times 3 \\ &= 1 - \left[\, 0.\overbrace{999\ldots99}^{n},\ 1.\overbrace{000\ldots0}^{n-1}2 \,\right] \\ &= \left[\, -0.\overbrace{000\ldots0}^{n-1}1,\ 0.\overbrace{000\ldots0}^{n-1}1 \,\right], \end{aligned}$$

koji sadrži tačan rezultat $x = 0$ za svako $n = 1, 2, \ldots$. ▲

2.3 Intervalne matrice

U ovom odeljku pomenućemo intervalne realne matrice i osnovne operacije sa intervalnim matricama koje su potrebne u Poglavlju 3. Matrice sa elementima iz skupa $I(\mathbb{R})$ označene su velikim masnim slovima $\boldsymbol{A}, \boldsymbol{B}, \ldots$ ili pomoću

$(A_{ij}), (B_{ij}), \ldots$; skup svih ovakvih matrica reda $n \times m$ je $M_{nm}(I(\mathbb{R}))$. Intervalni vektori označeni su malim masnim slovima $\boldsymbol{x}, \boldsymbol{y}, \ldots$. Konačno, tačkaste matrice koje pripadaju skupu $M_{nm}(\mathbb{R})$, biće označene pomoću $\dot{\boldsymbol{A}}, \dot{\boldsymbol{B}}, \ldots$, korišćenjem tačke ispod slova da se označi njihov karakter. Tačka se takođe koristi za označavanje realnih vektora.

Neka je $* \in \{+, -, \cdot, :\}$ jedna od uobičajenih binarnih operacija sabiranja, oduzimanja, množenja i deljenja kako u skupu realnih brojeva, tako i na skupu intervala $I(\mathbb{R})$ i intervalnih matrica $M_{nm}(I(\mathbb{R}))$. Za $\boldsymbol{A} = (A_{ij})$, $\boldsymbol{B} = (B_{ij})$, i $X \in I(\mathbb{R})$ matrične operacije $+, -, \cdot$ se formalno definišu kao u realnom slučaju:

$$\begin{aligned} \boldsymbol{A} \pm \boldsymbol{B} : &= (A_{ij} + B_{ij}), \\ \boldsymbol{A} \cdot \boldsymbol{B} : &= \Big(\sum_s A_{is} B_{sj}\Big), \\ X \cdot \boldsymbol{A} : &= (X \cdot A_{ij}), \end{aligned}$$

gde se pretpostavlja da $\boldsymbol{A}$ i $\boldsymbol{B}$ uvek imaju odgovarajući broj vrsta i kolona. Neka je intervalna matrica $\boldsymbol{A}$ iz skupa $M_{nn}(I(\mathbb{R}))$ i označimo odgovarajuću jediničnu matricu sa $\dot{\boldsymbol{I}} = \operatorname{diag}\{1, \ldots, 1\}$. Tada se stepeni $\boldsymbol{A}^k$ matrice $\boldsymbol{A}$ definišu pomoću

$$\boldsymbol{A}^0 = \dot{\boldsymbol{I}}, \quad \boldsymbol{A}^k = \boldsymbol{A}^{k-1} \cdot \boldsymbol{A} \ \ (k = 1, 2, \ldots).$$

Za realni interval $A = [\underline{a}, \overline{a}]$, definišemo *širinu* intervala

$$d(A) := \overline{a} - \underline{a}$$

i *apsolutnu vrednost*

$$|A| := \max\{|\overline{a}|, |\underline{a}|\}.$$

Definicija 2.3. Intervalnoj matrici $\boldsymbol{A} = (A_{ij})$ pridružujemo sledeće realne matrice:

a) *matricu širina* $d(\boldsymbol{A}) := (d(A_{ij}))$;

b) *matricu sredina* $\operatorname{mid}(\boldsymbol{A}) := (\operatorname{mid}(A_{ij}))$;

c) *matricu apsolutnih vrednosti* $|\boldsymbol{A}| := (|A_{ij}|)$. ▲

Definicija 2.4. Pod *inverznom matricom* intervalne matrice $\boldsymbol{A}$ podrazumevamo skup

$$\boldsymbol{A}^{-1} = \{A^{-1} \,:\, A \in \boldsymbol{A}\}. \ ▲$$

Teorema 2.2. *Za intervalne matrice* $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ *važi*

$$\begin{aligned}
&\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A} \quad \text{(komutativnost)},\\
&\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})=(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C} \quad \text{(asocijativnost)},\\
&A+\dot{\boldsymbol{O}}=\dot{\boldsymbol{O}}+\boldsymbol{A}=\boldsymbol{A} \quad (\dot{\boldsymbol{O}}-\text{nula-matrica}),\\
&\boldsymbol{A}\dot{\boldsymbol{I}}=\dot{\boldsymbol{I}}\boldsymbol{A}=\boldsymbol{A} \quad (\dot{\boldsymbol{I}}-\text{jedinična matrica}),\\
&(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{C}\subseteq\boldsymbol{A}\boldsymbol{C}+\boldsymbol{B}\boldsymbol{C} \quad \text{(subdistributivnost)},\\
&(\boldsymbol{A}+\boldsymbol{B})\dot{\boldsymbol{C}}=\boldsymbol{A}\dot{\boldsymbol{C}}+\boldsymbol{B}\dot{\boldsymbol{C}},\\
&\boldsymbol{A}(\dot{\boldsymbol{B}}\dot{\boldsymbol{C}})\subseteq(\boldsymbol{A}\dot{\boldsymbol{B}})\dot{\boldsymbol{C}}. \ \blacktriangle
\end{aligned}$$

Definicija 2.5. *Matrična norma* intervalne matrice $\boldsymbol{A}$ definiše se kao

$$\|\boldsymbol{A}\| := \max_{\dot{\boldsymbol{A}}\in\boldsymbol{A}} \|\dot{\boldsymbol{A}}\|,$$

gde je $\|\cdot\|$ proizvoljna matrična norma. ▲

U upotrebi je najčešće „maksimum suma kolona" norma $\|\cdot\|_\infty$:

$$\|\boldsymbol{A}\|_\infty := \max_{\dot{\boldsymbol{A}}\in\boldsymbol{A}} \|\dot{\boldsymbol{A}}\|_\infty = \max_i \Big|\sum_j |A_{ij}|\Big|.$$

Lako se pokazuje da važi implikacija

$$\boldsymbol{B}\subseteq\boldsymbol{A} \implies \|\boldsymbol{B}\|\le\|\boldsymbol{A}\|.$$

Teorema 2.3. *Važe sledeće relacije:*

$$\begin{aligned}
&\boldsymbol{B}\subseteq\boldsymbol{A} \implies |\boldsymbol{B}|\le|\boldsymbol{A}|,\\
&|\boldsymbol{A}+\boldsymbol{B}|\le|\boldsymbol{A}|+|\boldsymbol{B}|,\\
&|\lambda\boldsymbol{A}|=|\boldsymbol{A}\lambda|=|\lambda||\boldsymbol{A}| \quad (\lambda\in\mathbb{R}),\\
&|\boldsymbol{A}\boldsymbol{B}|\le|\boldsymbol{A}|\,|\boldsymbol{B}|,\\
&\dot{\boldsymbol{O}}\in\boldsymbol{A} \implies |\boldsymbol{A}|\le d(\boldsymbol{A})\le 2|\boldsymbol{A}|. \ \blacktriangle
\end{aligned}$$

2.4 INTLAB - softver za intervalnu aritmetiku

INTLAB je efikasan softver za intervalnu aritmetiku koji radi sa realnim i kompleksnim intervalima, intervalnim vektorima i intervalnim matricama. Razvio ga je Dr Zigfrid M. Rump, profesor Tehničkog univerziteta Hamburg-Harburg (Institute for Reliable Computing).

INTLAB koncept je zasnovan na dva principa. Prvi je razvoj brzih intervalnih biblioteka koje koriste isti kod za različite računarske arhitekture. Izabrane su biblioteke koje koriste bazične algoritme linearne algebre, tzv. Basic Linear Algebra Subroutines (kraće BLAS) koje se lako implemeniraju u skoro svim programskim jezicima. Drugi koncept je interaktivno programsko okruženje za jednostavno korišćenje intervalnih operacija. Z. Rump je izabrao programski paket MATLAB koji dozvoljava programiranje intervalnih algoritama na način koji je vrlo sličan pseudo-kodu korišćenom u naučnim publikacijama. Efikasan i pouzdan rad INTLAB-a postignut je korišćenjem IEEE 754 aritmetičkog standarda jer računarska aritmetika zadovoljava ovaj standard. Na ovaj način omogućeno je i stalno (po potrebi) uključivanje različitih modova zaokruživanja. Ovo uključivanje modova zaokruživanja u INTLAB-u se vrši pomoću instrukcije `setround(i)`, gde $i \in \{-1, 0, 1\}$ odgovara zaokruživanju *prema manjem* (-1), *ka najbližem* (0) i *prema većem* (1) PT broju.

Realni intervali u INTLAB-u se smeštaju preko infimuma (donja granica) i supremuma (gornja granica) naredbom `infsup`, dok se kompleksni intervali predstavljaju preko sredine (centra) i poluprečnika naredbom `midrad`.

Donja granica, gornja granica, sredina i poluprečnik intervala `x` dobijaju se pomoću naredbi `inf(x), sup(x), mid(x), rad(x)`. Intervalni vektori i intervalne matrice se unose na sličan način, pri čemu su argumenti vektori i matrice odgovarajućih dimenzija.

INTLAB koristi novu MATLAB funkciju `intval` (interval value) za unošenje intervalne promenljive ili za promenu proste promenljive u interval sa garantovanim granicama (koristeći odgovarajući mod zaokruživanja). Ova funkcija se odnosi i na realan i na kompleksan interval i ima jednu od najvažnijih uloga u verifikacionim algoritmima.[2] Pri korišćenju ove naredbe treba voditi računa o tome da li su brojevi na koje se primenjuje `intval` PT

[2]Pod verifikacionim algoritmima podrazumevaju se algoritmi koji produkuju interval koji sadrži tačan rezultat.

brojevi ili ne. Ako se broj a ne može predstaviti u binarnoj APT (na primer, $a = 0.1$), tada se umesto naredbe

```
>> x = intval(a)
```

koriste naredbe

```
>> x = intval('a')
intval x
```

Apostrofi u gornjoj naredbi označavaju da prvo treba primeniti konverziju. Na primer, za $a = 0.1$ stroge granice intervala mogu se dobiti koristeći `intval` zajedno sa konverzijom u binarni broj na sledeći način:

```
>> x=intval('0.1')
intval x =
[0.09999999999999, 0.10000000000001]
>> rad(x)
ans=
1.387778780781446e-017
```

Rezultujući interval ima poluprečnik koji nije nula i ovaj interval sadrži tačnu vrednost 0.1.

Intervalna matrica se može zadati na dva načina:

1) Ako su `Ad` i `Ag` matrice čiji su elementi donje i gornje granice elemenata intervalne matrice `A`, tada je `A=infsup(Ad,Ag)`. Na primer,

```
>> Ad = [2 -2;-1 2]
>> Ag = [4 1;2 4]
>> A = infsup(Ad,Ag)
intval A =
[ 2.0000, 4.0000] [-2.0000, 1.0000]
[-1.0000, 2.0000] [ 2.0000, 4.0000]
```

2) Intervalni elementi matrice formiraju se naredbom `infsup`:

```
>> A = [infsup(2,4) infsup(-2,1); infsup(-1,2) infsup(2,4)]
intval A =
[ 2.0000, 4.0000] [-2.0000, 1.0000]
[-1.0000, 2.0000] [ 2.0000, 4.0000]
```

Kao i u slučaju realnog vektora, za matricu `A` se matrice donjih granica, gornjih granica, sredina i poluprečnika formiraju naredbama `inf(A),sup(A)`, `mid(A),rad(A)`.

Inverzna matrica zadate matrice `A` dobija se jednostavnom naredbom

```
B=inv(A)
```

Pri izračunavanju inverzne intervalne matrice uzimaju se u obzir greške zaokruživanja (koristeći modove zakruživanja zadate pomoću `setround(i)`) tako da rezultujuća matrica `B` sadrži tačnu inverziju od A (ne uvek optimalnu). Ako intervalna matrica `A` sadrži singularnu matricu, inverzna intervalna matrica ne postoji i rezultat je matrica čiji su elementi simboli `NaN` (Not a Number) koji signaliziraju nedozvoljenu operaciju.

Detalji o INTLAB sofveru mogu se naći u fundamentalnom Rumpovom radu [119] i doktorskoj disertaciji (štampanoj u [59]) odbranjenoj na Univerzitetu u Mančesteru 2002. godine, u kojoj su date neke primene intervalnih metoda realizovane u INTLAB-u.

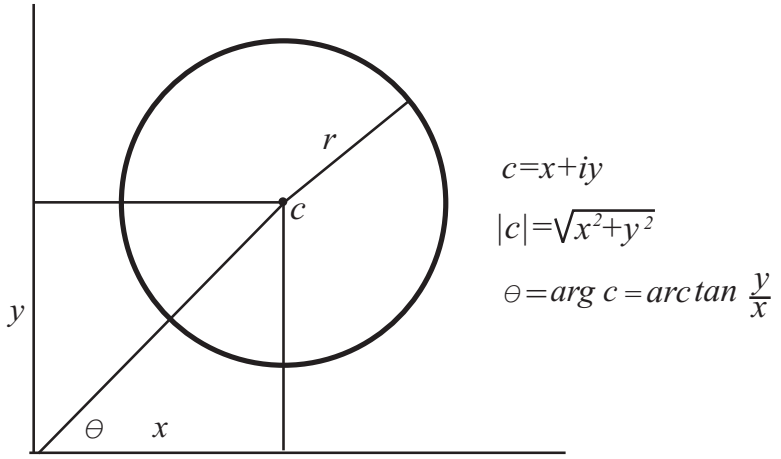
2.5 Kompleksna intervalna aritmetika

U radu s kompleksnim veličinama nepouzdane preciznosti potrebno je uvesti pojam kompleksnih intervala i razviti posebnu kompleksnu intervalnu aritmetiku. Postoji više različitih pristupa za konstrukciju kompleksnih intervalnih aritmetika, od kojih su najpoznatije aritmetika pravougaonika i aritmetika diskova, poznata takođe i pod nazivom *kružna intervalna aritmetika*. Aritmetika pravougaonika je zasnovana na realnoj intervalnoj aritmetici i kompleksan interval A u obliku pravougaonika sa stranama paralelnim koordinatnim osama predstavlja se u obliku $A = A_1 + iA_2$, gde su A_1 i A_2 realni intervali. Osnovne operacije se, zatim, definišu koristeći osnovna pravila za operacije u običnoj kompleksnoj aritmetici.

U ovom poglavlju razmatraćemo kružnu kompleksnu intervalnu aritmetiku i razviti odgovarajući paket programa za rad sa kompleksnim intervalima – diskovima. Kružna kompleksna intervalna aritmetika zasnovana je na inkluzivnoj aproksimaciji kompleksnih veličina pomoću kružnih diskova u kompleksnoj ravni, tj. takvim aproksimacijama gde se umesto kompleksnog broja z radi sa diskom Z koji sadrži z.

Disk Z u kompleksnoj ravni sa centrom u tački $z = x+iy$ i poluprečnikom r, predstavićemo pomoću centra c i poluprečnika r kao $Z = \{c; r\}$, slika 2.1. S obzirom da se kompleksan broj $z = x + iy$ može tretirati kao uređen par realnih brojeva (x, y), disk Z se može predstaviti pomoću uređene trojke

realnih brojeva $Z = (x, y, r)$. Skup svih kružnih intervala označava se sa $K(\mathbb{C})$.



Slika 2.1 Predstavljanje diska u kompleksnoj ravni

Operacije u kružnoj intervalnoj kompleksnoj aritmetici uveli su Gargantini i Henrici u radu [50]. Neka su $Z = \{c; r\}$, $Z_1 = \{c_1; r_1\}$ i $Z_2 = \{c_2; r_2\}$ kružne zatvorene oblasti (diskovi). Osnovne operacije kružne aritmetike definišu se na sledeći način:

$$\begin{aligned} \{c_1; r_1\} \pm \{c_2; r_2\} &= \{c_1 \pm c_2; r_1 + r_2\}, \\ \{c_1; r_1\} \cdot \{c_2; r_2\} &= \{c_1 c_2; |c_1| r_2 + |c_2| r_1 + r_1 r_2\} \\ &\supseteq \{z_1 \cdot z_2 \ : \ z_1 \in Z_1 \wedge z_2 \in Z_2\}. \end{aligned}$$

Inverzija diska Z koji ne sadrži 0 nalazi se pomoću Mebijusove transformacije diska Z pomoću preslikavanja $f(z) = 1/z$, tj.,

$$Z^{-1} = \{c; r\}^{-1} = \left\{ \frac{1}{z} \ : \ z \in Z \right\} = \frac{\{\bar{c}; r\}}{|c|^2 - r^2} \quad (|c| > r, \text{ tj., } 0 \notin Z). \tag{2.7}$$

Kako je

$$\{c_1; r_1\} \pm \{c_2; r_2\} = \{z_1 + z_2 \ : \ z_1 \in Z_1 \wedge z_2 \in Z_2\}, \quad \{c; r\}^{-1} = \left\{ \frac{1}{z} \ : \ z \in Z \right\},$$

zaključujemo da su sabiranje diskova i inverzija diska tačne operacije, ali da množenje diskova nije tačna operacija i da se pri množenju diskova dobija uvećani disk koji sadrži tačnu oblast (koja u opštem slučaju nije disk).

Osim tačne inverzije Z^{-1} često se koristi i tzv. *centirana inverzija* Z^{I_c} definisana pomoću

$$Z^{I_c} = \{c; r\}^{I_c} := \left\{ \frac{1}{c}; \frac{r}{|c|(|c| - r)} \right\} \supseteq Z^{-1} \quad (0 \notin Z). \tag{2.8}$$

Koristeći formule za množenje diskova i inverziju diskova, deljenje se definiše kao

$$Z_1 : Z_2 = Z_1 \cdot Z_2^{-1} \text{ ili } Z_1 : Z_2 = Z_1 \cdot Z_2^{I_c} \quad (0 \notin Z_2).$$

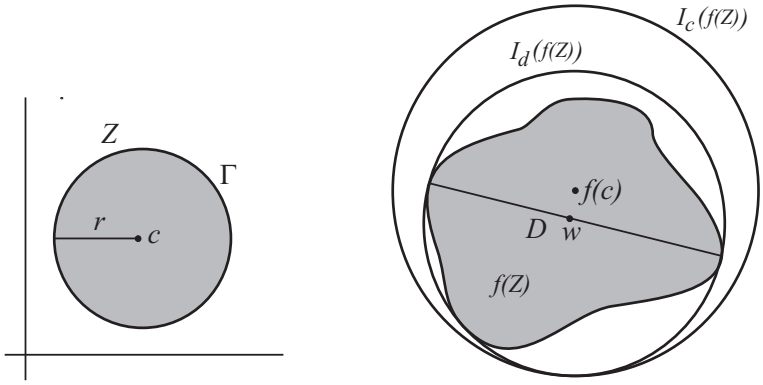
Napomena 2.1. Pokazuje se da u teoriji intervalnih iterativnih metoda za simultanu inkluziju nula polinoma centrirana inverzija (2.8) ima prednost nad tačnom inverzijom (2.7). Naime, detaljna analiza data u radu [114] pokazuje da je konvergencija poluprečnika diskova–argumenata kod primene intervalnih metoda koji rade sa diskovima brža ukoliko su centri ovih diskova bliži traženim nulama polinoma.

2.6 I-aproksimacije

Neka je f kompleksna funkcija kompleksne promenljive z koja je definisana za svako $z \in Z$, gde je $Z = \{c; r\}$ disk sa centrom u c i poluprečnikom r. Izračunavanje vrednosti funkcije $f(z)$ za neki argument z_0 vrši se pomoću aritmetike konačne preciznosti. Stoga se vrednost $f(z_0)$ može izračunati i predstaviti samo u konačnoj preciznosti kao zaokružena vrednost. Imajući u vidu i greške koje se javljaju pri izračunavanju međurezultata, jasno je da se dobija aproksimacija tačne vrednosti sa greškom nepoznate veličine. Za procenu i analizu načinjene greške moraju se razviti posebni postupci. Ponekad je moguće analitički odrediti granice greške ali to je uglavnom vrlo složen i težak problem.

Suštinski pomak u prevazilaženju ovog problema predstavlja korišćenje intervalne matematike zbog mogućnosti *automatske kontrole greške*. Najpre predstavljamo kompleksan broj z_0 pomoću kruga Z_0 koji ga sadrži, a zatim vršimo aproksimaciju oblasti $f(Z_0) = \{f(z) \mid z \in Z_0\}$ pomoću diska W, koji ovu oblast sadrži (odnosno, važi inkluzija $f(Z_0) \subset W$). Korišćenjem intervalne matematike razvijeni su algoritmi za što efikasniju kružnu aproksimaciju oblasti $f(Z_0)$, pomoću kojih se dobija disk W što je moguće manjeg

poluprečnika (koji istovremeno predstavlja gornju granicu greške ove aproksimacije).



Slika 2.2 Dijametarski $I_d(f(Z))$ i centrirani inkluzivni disk $I_c(f(Z))$ za oblast $f(Z)$

Svaki disk W koji ispunjava uslov da je $f(Z) \subseteq W$, zove se inkluzivni disk oblasti $f(Z)$ ili **I-aproksimacija**. Jasno je da ovakva aproksimacija nije jedinstvena. Smatramo da je neki disk W utoliko bolja I-aproksimacija oblasti $f(Z)$ ukoliko je manji njegov poluprečnik. U tom smislu, *najbolja I-aproksimacija* je onaj disk čiji je poluprečnik najmanji, odnosno čija je površina najmanja. Označimo takav disk sa $V = \{\zeta; R\}$. Dijametar diska V može u najboljem slučaju da bude jednak dijametru oblasti $f(Z)$, odnosno, važiće

$$2R = D = \mathrm{diam} f(Z) = \max_{z_1, z_2 \in Z} |f(z_1) - f(z_2)|.$$

Kako je oblast $f(Z)$ zatvorena, pri nalaženju dijametra dovoljno je uzeti tačke sa konture γ diska Z, tj.

$$D = \max_{z_1, z_2 \in \gamma} |f(z_1) - f(z_2)|. \tag{2.9}$$

Ovakav disk, ukoliko postoji, zove se *dijametarski inkluzivni disk* oblasti $f(Z)$ ili, kraće, *D-forma* i biće označen kao

$$V_d = I_d(f(Z)) = \{w; D/2\},$$

gde su centar w i poluprečnik $R = D/2$ određeni pomoću

$$w = \frac{f(z_1) + f(z_2)}{2}, \quad R = \frac{D}{2} = \frac{|f(z_1) - f(z_2)|}{2},$$

pri čemu je z_1, z_2 bilo koji par tačaka sa konture γ koji ispunjava relaciju (2.9). Aritmetika koja radi sa dijametarskim diskovima skraćeno se zove D-aritmetika.

Problem nalaženja dijametarskog diska za neku oblast ne samo što je težak zadatak (imajući na umu rešavanje složenih ekstremalnih problema pri nalaženju dijametra) već ponekad ni nema rešenje. U slučajevima kada ili nije moguće dokazati postojanje dijametarskog diska ili on ne postoji za datu oblast, koriste se drugi oblici inkluzivnih aproksimacija. Najpogodnija I-aproksimacija je takozvana *centralna* ili *C-forma*, koja predstavlja inkluzivni disk $I_c(f(Z))$ čiji je centar $f(c)$ slika centra domena $Z = \{c; r\}$. Drugim rečima, C-forma neke oblasti $f(Z)$, gde je $Z = \{c; r\}$, ima oblik

$$V = I_c(f(Z)) = \{f(c); R\}.$$

Primer dijametarskog i centriranog diska za oblast $f(Z)$ prikazan je na Slici 2.2.

Centrirani diskovi $I_c(f(Z))$ za standardne elementarne kompleksne funkcije dati su u donjoj listi. Za multiformne funkcije $\operatorname{Ln} z$ i $z^{1/n}$ koriste se unije diskova u C-formi.

1. $f(z) = z^n, \quad I_c(Z^n) = \{c^n; (|c| + r)^n - |c|^n\};$

2. $f(z) = e^z, \quad I_c(e^Z) = \{e^c; |e^c|(e^r - 1)\};$

3. $f(z) = \sin z,$
$I_c(\sin Z) = \{\sin c; \operatorname{sh} r(\operatorname{ch}^2 y - \sin^2 z)^{1/2} + (\operatorname{ch} r - 1)(\operatorname{ch}^2 y - \cos^2 x)^{1/2}\};$

4. $f(z) = \cos z,$
$I_c(\cos Z) = \{\cos c; \operatorname{sh} r(\operatorname{ch}^2 y - \cos^2 z)^{1/2} + (\operatorname{ch} r - 1)(\operatorname{ch}^2 y - \sin^2 x)^{1/2}\};$

5. $f(z) = \operatorname{Ln} z, \ 0 \notin Z,$
$I_c(\operatorname{Ln} Z) = \{\operatorname{Ln} c; -\ln(1 - r/|c|)\} = \bigcup_{k \in \mathbb{Z}} \{\ln c + 2k\pi i; -\ln(1 - r/|c|)\};$

6. $f(z) = z^{1/n}, \ 0 \notin Z,$
$I_c(Z^{1/n}) = \{c^{1/n}; |c|^{1/n} - (|c| - r)^{1/n}\}$
$$= \bigcup_{m=0}^{n-1} \left\{ |c|^{1/n} \exp\Big(i\frac{\theta + 2m\pi}{n}\Big); |c|^{1/n} - (|c| - r)^{1/n} \right\}.$$

Oznaka $\mathbb{Z}$ označava skup celih brojeva.

2.7 Osnovne operacije u D-aritmetici

Neka su $Z_1 = (x_1, y_1, r_1)$ i $Z_2 = (x_2, y_2, r_2)$ dva diska. Sabiranje i oduzimanje diskova u D-aritmetici definiše se na sledeći način:

SABIRANJE:

$$Z_1 + Z_2 = (x_1 + x_2, y_1 + y_2, r_1 + r_2)$$

ODUZIMANJE:

$$Z_1 - Z_2 = (x_1 - x_2, y_1 - y_2, r_1 + r_2)$$

Za ove operacije važi

$$Z_1 + Z_2 = \{z_1 + z_2 : z_1 \in Z_1, z_2 \in Z_2\}, \quad Z_1 - Z_2 = \{z_1 - z_2 : z_1 \in Z_1, z_2 \in Z_2\},$$

što znači da su ove operacije *tačne*, a samim tim su i rezultujući diskovi dijametarski.

INVERZIJA:

$$Z^{-1} = \Big(\frac{x}{x^2 + y^2 - r^2}, \frac{-y}{x^2 + y^2 - r^2}, \frac{r}{x^2 + y^2 - r^2}\Big) \tag{2.10}$$

Pre izvršavanja ove operacije treba proveriti da li disk Z sadrži nulu, tj. da li je $x^2 + y^2 \leq r^2$. Ako je to slučaj, sledi poruka:
`operacija je nedefinisana - disk sadrži nulu`.

Napomenimo da se inverzni disk definisan formulom (2.10) dobija primenom Mebijusove transformacije $w = 1/z$ na disk $Z = \{c; r\}$ $(c = x + iy)$. Kao što je poznato, ova transformacija preslikava krug $Z = \{c; r\}$ na krug $\{1/z : z \in Z\}$, koji je definisan formulom (2.10). Prema tome, Z^{-1} je *tačna operacija* koja daje dijametarski disk za oblast $\{1/z : z \in Z\}$, koja se u ovom slučaju poklapa sa diskom Z^{-1} datim pomoću (2.10).

MNOŽENJE:

U svojoj doktorskoj disertaciji (štampanoj u [20]) Börsken je dokazao da oblast $\{z_1 \cdot z_2 \ : \ z_1 \in Z_1, z_2 \in Z_2\}$ nije disk u opštem slučaju, ali da se ova oblast može uvek okružiti diskom čiji je prečnik jednak dijametru ove oblasti

(videti sliku 2.3). Uvedimo kvadrate modula η_1 i η_2 centara c_1 i c_2 diskova Z_1 i Z_2,

$$\eta_1 = |c_1|^2 = x_1^2 + y_1^2, \quad \eta_2 = |c_2|^2 = x_2^2 + y_2^2.$$

Tada je množenje u D-aritmetici definisano na sledeći način:

$$\begin{aligned} Z_1 \cdot Z_2 : \; = \; & \Big(x_1x_2 - y_1y_2(1+\tau), x_1y_2 - y_1x_2(1+\tau), \Big[3\eta_1\eta_2\tau^2 + \\ & 2\big(\eta_1\eta_2 + \eta_1 r_2^2 \quad + \eta_2 r_1^2\big)\tau + \eta_1 r_2^2 + \eta_2 r_1^2 + r_1^2 r_2^2 \Big]^{1/2} \Big), \end{aligned} \tag{2.11}$$

gde je τ pozitivan realan koren polinoma

$$P(t) = 2\eta_1\eta_2 t^3 + \big(\eta_1\eta_2 + \eta_1 r_2^2 + \eta_2 r_1^2\big)t^2 - r_1^2 r_2^2.$$

Ako je ovaj polinom stepena < 2, tada je $\tau = 0$.

Neka je

$$p_1 = r_1/|c_1| = r_1/\sqrt{x_1^2 + y_1^2}, \quad p_2 = r_2/|c_2| = r_2/\sqrt{x_2^2 + y_2^2}.$$

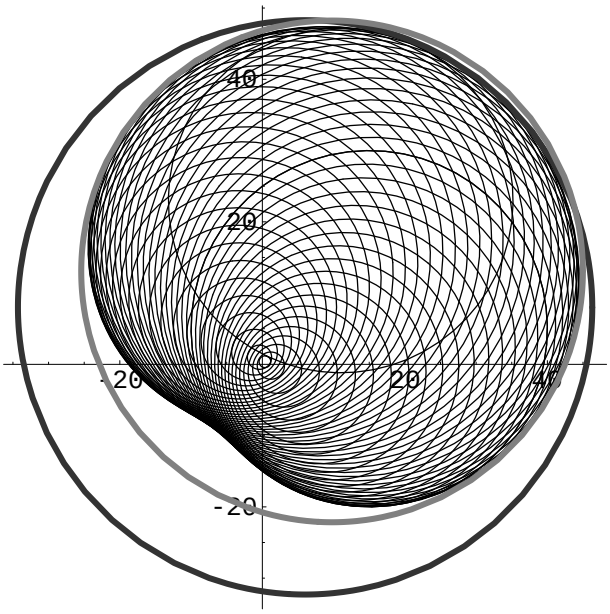
Deleći $P(t)$ sa $|c_1^2 c_2^2| = \eta_1\eta_2$, dobijamo polinom

$$G(t) = 2t^3 + (1 + p_2^2 + p_1^2)t^2 - p_1^2 p_2^2$$

koji ima iste nule kao $P(t)$. S obzirom da je polinom $G(t)$ rastuća funkcija za $t \geq 0$ i da je

$$G(0) = -p_1^2 p_2^2 < 0, \; G(p_1) > 0, \; G(p_2) > 0,$$

za nalaženje korena τ pogodno je upotrebiti metod sečice polazeći od intervala $[0, \min\{p_1, p_2\}]$. Iterativni proces se prekida kada bude ispunjen uslov $|G(t_k)| < \varepsilon$ (recimo, $\varepsilon = 10^{-10}$) pri čemu se uzima $\tau = t_k$.



Slika 2.3 Oblast vrednosti $\{z_1 z_2 \ : \ z_1 \in Z_1, z_2 \in Z_2\}$ i inkluzivni diskovi $Z_1 \cdot Z_2$ u C- (veći disk) i D-formi (manji disk)

DELJENJE:

Koristeći inverziju diska datu formulom (2.10) i množenje definisano sa (2.11), deljenje u D-aritmetici se defiše sa

$$Z_1 : Z_2 := Z_1 \cdot Z_2^{-1}.$$

Disk Z_2 ne sme da sadrži nulu, zato se najpre proverava $x_2^2 + y_2^2 - r_2^2 > 0$. Ako ovaj uslov ne važi, šalje se poruka:

`deljenje nulom, operacija deljenje nije definisana`.

D-FORMA ELEMENTARNIH KOMPLEKSNIH FUNKCIJA

Pri rešavanju problema koji koriste kompleksnu aritmetiku uz pomoć digitalnih računara, od posebnog značaja su elementarne funkcije: stepena z^n, korenska $z^{1/n}$, logaritamska $\ln z$, eksponencijalna e^z, trigonometrijske funkcije $\sin z$ i $\cos z$ i inverzne trigonometrijske funkcije $\tan^{-1} z$, $\cot^{-1} z$, $\arcsin z$, $\arccos z$, hiperboličke funkcije $\sinh z$, $\cosh z$ i inverzne hiperboličke funkcije

$\sinh^{-1}z$, $\cosh^{-1}z$, $\tanh^{-1}z$ i $\coth^{-1}z$. Ove funkcije zovu se bibliotečke (subroutine library) funkcije. Za neke od ovih funkcija moguće je odrediti najbolje inkluzivne aproksimacije u obliku dijametarskih diskova. U Dodatku 1 je dat pregled rezultata dobijenih u [20] i [115] za logaritamsku, eksponencijalnu, stepenu, korensku i još neke elementarne funkcije. Za trigonometrijske i hiperboličke funkcije nije dokazano postojanje dijametarske forme pa je iskorišćena optimalna centralna forma. Na osnovu tih rezultata sačinjena je baza kompleksnih intervalnih elementarnih funkcija i razvijen odgovarajući softver dat u Odeljku 2.8 i Dodatku 2.

2.8 Softver za kompleksnu kružnu intervalnu aritmetiku

Softver za kružnu intervalnu artimetiku je urađen u programskom jeziku C# korišćenjem Microsoft Visual Studio 2008 okruženja. Sastoji se iz nekoliko celina:

- Korisnički interfejs
- Paket potrebnih matematičkih operacija i struktura (Base)
- Paket za izračunavanja u kružnoj intervalnoj aritmetici

Korisnički interfejs se sastoji iz dijaloga i obrazaca gde korisnik navodi parametre izračunavanja i gde može videti rezultate. Rezultati se trenutno prikazuju preko matematičkih izraza, a u planu je takođe i grafički prikaz.

Base paket sadrži nekoliko klasa i interfejsa. Klasa **Complex** omogućava rad sa kompleksnim brojevima. Implementirani su svi relevantni operatori kao i razne korisne funkcije:

- Izračunavanje ugla polarnih koordinata u radijanima ili stepenima
- Izračunavanje apsolutne vrednosti kompleksnog broja
- Konjugovanje kompleksnog broja
- Konvertovanje iz polarnih koordinata

Kompleksan broj se može kreirati na više načina, npr:

– učitavanjem realnog i imaginarnog dela

```
Complex broj = new Complex(2.3, -3.4);
```

– učitavanjem apsolutne vrednosti i ugla u radijanima

```
Complex broj = new Complex(5, 1.2,
Complex.ConstructorMode.Polar);
```

Nakon kreiranja kompleksnog broja, njegovo korišćenje je vrlo jednostavno:

```
Complex broj1 = new Complex(5, 1.2,
Complex.ConstructorMode.Polar);
Complex broj2 = new (-3.2, 4.2);
Complex broj3;
// konjugovanje broja 1
broj3 = broj1.Conjugation();
// izračunavanje ugla polarnih koordinata
double ugao = broj3.Arg();
// upotreba raznih operatora
broj3 = broj2 * broj1;
broj3 += broj1;
```

Klasa **Polinom** sadrži sve potrebne funkcije za definisanje samog polinoma (npr. dodavanje koeficijenata), nalaženje nula polinoma kao i izračunavanje vrednosti polinoma za zadati argument.

Polinom se zadahe pomoću niza kompleksnih brojeva (objekata klase **Complex**) pridruženih odgovarajućim stepenima polinoma. Nalaženje nula polinoma se vrši pomoću funkcije `FindRoots`. Jedan od prototipa ove funkcije je sledećeg oblika:

```
public List<Complex> FindRoots(IRootMethod method,
IApproximationFinder approxFinder)
```

```
namespace Base.RootMethods
{
  class Newton :    IRootMethod
{     const int MAX_INTERATIONS = Constants.DefaultNbIterations;

    // default error is 10^(-8)
    double _errorBound = Constants.DefaultErrorBound;

    public double ErrorBound
    {
      get {
        return _errorBound;
      }
      set {
        _errorBound = value;
      }
    }

    public List<Complex> FindRoots(Polynomial p,
    Complex approximation)
    {
      List<Complex> roots = new List<Complex>();
      int counter = 0;
      double t = approximation.Abs();
      double pValue = p.Calculate(t).Real;
      while (counter < MAX_INTERATIONS && pValue > ErrorBound)
      {
        t -= pValue / p.Derivate().Calculate(t).Real;
        Debug.WriteLine(t);
        // get value P(tk) for new tk
        pValue = p.Calculate(t).Real;
      }
      // TODO: where are other roots?
            roots.Add(new Complex(t));
      return roots;
    }
    //public double FindRoots(
  }
}
```

Početna aproksimacija kao i metod kojim će se nalaziti nule su navedeni kao parametri funkcije. Svi metodi za nalaženje nula se realizuju kao klase koje implementiraju interfejs IRootMethod. Ukoliko se ne navede neki specifičan metod, biće korišćen Njutnov metod. Navedeni interfejs deklariše kriterijum zaustavljanja iterativnog procesa `ErrorBound` definisan zahtevanom tačnošću (greška aproksimacije), kao i samu funkciju za nalaženje aproksimacije nule funkcije. Primer implementacije Njutnovog metoda dat je programom na prethodnoj strani.

Prednost ovakvog pristupa je što se nule polinoma mogu nalaziti pomoću proizvoljnog metoda i proizvoljne aproksimacije. Oni nisu deo polinoma i predstavljaju klasičan primer primene **Strategy** šablona projektovanja (design pattern).

Načini nalaženja aproksimacije mogu biti različiti, takođe realizovani kao klase koje implementiraju interfejs IApproximationFinder. Te klase su deo namespace-a `RootApproximations`, gde je i standardna klasa pod nazivom DefaultApproximation.

Celokupna prethodno navedena softverska infrastruktura je neophodna radi implementacije klase **Disk**, kojom se predstavljaju diskovi u kružnoj intervalnoj aritmetici. Objekat klase **Disk** se instancira navođenjem centra (objekat klase **Complex**) i poluprečnika diska. Na primer:

```
Disk d = new Disk (new Complex(2.3, 4.5), 3.55);
```

Klasa **Disk** sadrži operatore *, /, - , + kao i sledeće javne funkcije:

- `Inversion` - funkcija za inverziju diska
- `LnId` - dijamatarska inkluzija za Ln Z
- `BilinearTransformation` - izračunavanje bilinearne transformacije
- `Exp` - dijametarska inkluzija za e^Z
- `Pow` - dijamatarska inkluzija za Z^n
- `Root` - dijametarska inkluzija za izračunavanje $Z^{1/n}$
- `Atan` - dijametarska inkluzija za $\tan^{-1} Z$
- `ACot` - dijametarska inkluzija za $\cot^{-1} Z$

- `Atanh` - dijametarska inkluzija za $\tanh^{-1} Z$
- `Acoth` - dijametarska inkluzija za $\coth^{-1} Z$
- `SinIc` - izračunavanje centrirane inkluzije $I_c(\sin Z)$
- `CosIc` - izračunavanje centrirane inkluzije $I_c(\cos Z)$
- `SinhIc` - izračunavanje centrirane inkluzije $I_c(\sinh Z)$
- `CoshIc` - izračunavanje centrirane inkluzije $I_c(\cosh Z)$

Kodovi za napred navedene operacije u D-aritmetici i dijametarske diskove za većinu elementarnih funkcija dati su u Dodatku B.

3

Matrični modeli procesnih iteracija

U ovom poglavlju razmatrane su razvojno-projektne procesne iteracije koje se u praksi javljaju pri rešavanju problema globalnog tipa. Ovaj tip problema je sastavljen od velikog broja povezanih taskova između kojih često postoje vrlo složene međusobne zavisnosti. Pokazalo se da je modeliranje pomoću matrice strukture procesne iteracije i rešavanje u više sukcesivnih koraka (iteracija) veoma efikasan način za rešavanje problema ove vrste. U ovom poglavlju data su dva originalna pristupa za analizu karakteristika razvojno-projektnog procesa zasnovanog na Epinger-Smitovom modelu [128], [129]. Pokazuje se da se rangiranje taskova i brzina procesne iteracije može izvršiti na jednostavan i efikasan način korišćenjem sopstvenih vrednosti i sopstvenih vektora nenegativne matrice koja modelira zadati globalni problem rastavljen na međusobno zavisne taskove. U drugom delu ovog poglavlja izložen je originalni pristup u analizi modela procesne iteracije koji koristi intervalnu linearnu algebru i radi sa realnim intervalima i intervalnim matricama umesto realnih brojeva i realnih matrica. Na ovaj način postiže se veći stepen slobode u kvantitativnoj proceni taskova i kontrola tačnosti rezultata.

3.1 Modeli procesnih iteracija

Projektovanje i razvoj novih proizvoda, bilo da se radi o industrijskim proizvodima, tehničkom rešenju, hardveru ili softveru, često sadrži vrlo kompleksan skup međusobnih relacija između velikog broja povezanih problema. Ta

zavisnost dovela je do potrebe za rešavanjem globalnog problema u jednom velikom projektu korak po korak, tj. do *iterativnog pristupa* koji uključuje različite međusobno zavisne zadatke (taskove) u razvoju novog proizvoda. Modeli koji se razvijaju ili projektuju pomoću iterativnog postupka obezbeđuju suštinsko shvatanje efekata složenih međurelacija u okviru razvojnog procesa.

Proizvodne firme danas se suočavaju sa ogromnim pritiskom da poboljšaju produktivnost u razvoju i realizaciji proizvoda. Posebna pažnja posvećena je vremenu neophodnom za razvoj novog proizvoda ili redizajniranju nekog postojećeg. Jedan pristup za poboljšanje karakteristika proizvoda je zasnovan na činjenici da su razvoj i projektovanje proizvoda često veoma proceduralni i sadrže ponavljanja (iteriranja); stoga se ovaj proces iterativnog karaktera može modelirati u velikoj meri na isti način kao što bismo to uradili za proizvodni proces koji želimo da poboljšamo. Ovu iterativnu proceduru zvaćemo *razvojno-projektna procesna iteracija* ili kraće *procesna iteracija* usvajajući terminologiju koju su uveli istraživači sa Masečusetskog instituta za tehnologiju (MIT, Boston) pre svih Smith i Eppinger (videti, npr., [128], [129]). Projektno-razvojni proces pomoću koga se razvijaju raznorodne tehničke procedure (metodi) za rešavanje datog problema treba shvatiti u širem smislu. Nekada je to iteriranje taskova pri realizaciji gotovog proizvoda. Klark i Fujimoto [27] opisuju slučajeve gde konstruktori razmenjuju bitne tehničke informacije i pomoću toga poboljšavaju proces proizvodnje, dok Whitney [150] opisuje procesnu iteraciju kao interakciju između projektno-razvojnih aktivnosti.

Višegodišnja proučavanja opisanih procesa iterativnog karaktera u kojima učestvuje veliki broj međusobno zavisnih faktora pokazala su da se najbolji rezultati postižu pomoću modela koji koristi matricu strukture procesne iteracije. Ovaj model se u literaturu najčešće naziva DSM (design structure matrix) modelem. Ovaj model ima veoma široku primenu i ovde pominjemo samo neke od oblasti primena:

– Projektovanje u inženjerskim disciplinama [86], [61], [3], [118], [35], [125], [126], [54], [23];

– Proučavanje i modelovanje zavisnih struktura [149], [89], [95], [72], [137], [130], [143];

– Organizacija i optimizacija projektovanja i proizvodnje [24], [116], [94], [38], [85], [51], [127], [147];

– Informacione nauke i inženjerstvo [153], [154], [146];

– Menadžment i e-biznis [55], [136], [62], [33], [45], [87], [26], [64], [158], [63];

– Evolucioni algoritmi [117], [157];

Procesne iteracije su veoma uobičajene u razvoju proizvoda tako da je njihovo izučavanje i pravilna interpretacija njihove strukture od izuzetnog značaja za ubrzanje i poboljšanje projektno-razvojnog procesa (Kline [79]). U ovom poglavlju date su precizne interpretacije matrične strukture modela procesne iteracije opisane u radu [129] korišćenjem sopstvenih vrednosti i sopstvenih vektora odgovarajuće matrice (Odeljci 4.3–4.5). U drugom delu poglavlja uveden je jedan nov pristup u analizi ovog modela koji koristi intervalnu linearnu algebru i radi sa realnim intervalima umesto realnih brojeva (Odeljci 4.6 i 4.7), prezentovan na Internacionalnoj konferenciji u Kjotu 2010. godine i opisan u radu [109]. Tim postupkom postignut je veći stepen slobode u kvantitativnoj proceni taskova i dopušteno je prisustvo (do izvesne mere, definisano dužinom intervala) „neodređenih" veličina. Na primer, ako neki task učestvuje u celokupnom razvojnom procesu sa otprilike 10%, tada je rad sa intervalom (0.09,0.11) (koji predstavlja procenu od 9% do najviše 11%) svrsishodniji nego rad sa fiksnim brojem 0.1 (10%). Pri korišćenju intervalnog modela razmatranog u ovom radu umesto realnih matrica razmatramo intervalne matrice strukture procesa (IMSP) i intervalne matrice transformacije rada (IMTR) koje koristimo za modeliranje bilo procesa proizvodnje, bilo projektovanja ili razvoja proizvoda. Oba pristupa ilustrovana su numeričkim primerima u Odeljcima 4.5 (realne matrice i vektori) i 4.7 (intervalne matrice i vektori).

Analitičkom analizom karakteristične strukture matričnog modela može se steći uvid u pojedinačni doprinos ili uticaj svakog taska (od kojih se neki mogu ponavljati sa umanjenim ili uvećanim intenzitetom) u svakom iterativnom koraku, pri čemu se smatra da je model uspešan ako sve aktivnosti (taskovi) konvergiraju posle izvesnog broja iterativnih koraka. Jasno je da je model utoliko bolji ukoliko je broj iteracija manji. Na ovaj način u mogućnosti smo da analiziramo globalni proces preko pojedinačnih grupa taskova i odredimo koji podskupovi taskova zahtevaju najveći deo uvećanog rada tokom izvršenja iterativnog procesa.

3.2 Matrična linearna algebra

Model procesne iteracije koristi matricu strukture procesa (MSP) koja mora biti nenegativna i nerazloživa, o čemu će više biti reči u Odeljku 4.3. Pojam nerazložive matrice i osnovni elementi matrične linearne algebre dati su u ovom odeljku. Više detalja može se naći u knjizi [144]. U prvom delu ovog poglavlja matrice ćemo označavati velikim slovima a vektore malim masnim slovima.

Matricu čiji su elementi realni brojevi zvaćemo realna ili tačkasta matrica, da bismo je razlikovali od intervalne matrice razmatrane u Odeljcima 4.6 i 4.7. Skup tačkastih matrica koje imaju n vrsta i m kolona označavaćemo sa $M_{nm}(\mathbb{R})$. Realnu matricu

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & & a_{2m} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nm} \end{bmatrix}$$

označavaćemo kraće sa $A = \big(a_{ij}\big)_{n\times m}$ ili sa $A = \big(a_{ij}\big)$ ako je jasno o kojim dimenzijama se radi.

Matricu koja na glavnoj dijagonali ima elemente $d_1, d_2, \cdots, d_n$, dok su svi elementi van ove dijagonale jednaki 0, zvaćemo dijagonalna matrica i označavaćemo je na sledeći način

$$\begin{bmatrix} d_1 & & & \\ & d_2 & & 0 \\ & & \ddots & \\ & 0 & & \\ & & & d_n \end{bmatrix} = \operatorname{diag}\{d_1, d_2, \ldots, d_n\}.$$

Jedinična matrica $I = \operatorname{diag}\{1, 1, \ldots, 1\}$ je specijalni slučaj dijagonalne matrice koja po glavnoj dijagonali ima sve jedinice.

Vektor $\boldsymbol{x}$ koji ima n elemenata je specijalni slučaj kolona-matrice dimenzije $n \times 1$, tj., to je vektor-kolona

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [x_1 \; x_2 \; \cdots \; x_n]^T,$$

gde je T oznaka za transponovanu matricu. Takođe ćemo koristiti i zapisivanje preko uređene n-torke, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$.

U nastavku dajemo nekoliko definicija koje se odnose na sopstvene vrednosti i sopstvene vektore matrica.

Definicija 3.1. Nenula vektor $\boldsymbol{x}$ zove se *sopstveni vektor* date realne ili kompleksne matrice A reda $n \times n$ ako postoji skalar $\lambda \in \mathbb{C}$ takav da važi

$$A\boldsymbol{x} = \lambda\boldsymbol{x}.$$

Broj λ zove se *sopstvena vrednost* matrice A koja odgovara sopstvenom vektoru $\boldsymbol{x}$. ▲

Definicija 3.2. Ako je A kvadratna matrica, polinom $P(\lambda) = \det(A - \lambda I)$ zove se *karakteristični polinom,* a odgovarajuća jednačina $P(\lambda) = 0$ njena *karakteristična jednačina.* Rešenja ove jednačine $\lambda_1, \ldots, \lambda_n$ su sopstvene vrednosti matrice A. ▲

Definicija 3.3. Neka je $A = (a_{ij})$ matrica reda $n \times n$ sa sopstvenim vrednostima $\lambda_1, \ldots, \lambda_n$. Tada je

$$\rho(A) := \max_{1 \le i \le n} |\lambda_i|$$

spektralni radijus matrice A. ▲

Geometrijski, ako se sve sopstvene vrednosti λ_i matrice A predstave u kompleksnoj ravni, tada je $\rho(A)$ radijus najmanjeg diska $|z| \le R$ sa centrom u koordinatnom početku koji sadrži sve sopstvene vrednosti matrice A.

Teorema 3.1. *Neka su $\lambda_1, \ldots, \lambda_n$ sopstvene vrednosti matrice A reda $n \times n$. Tada su $F(\lambda_1), \ldots, F(\lambda_n)$ sopstvene vrednosti matrice $F(A)$, gde je $x \mapsto F(x)$ algebarski polinom.* ▲

U nastavku se definišu pojmovi nenegativne i nerazložive matrice.

Definicija 3.4. Neka su $A = (a_{ij})$ i $B = (b_{ij})$ dve realne matrice reda $n \times r$. Kaže se da je $A \ge B$ ako je $a_{ij} \ge b_{ij}$ za svako $1 \le i \le n,\ 1 \le j \le r$. Ako je O nula matrica i $A \ge O$ (> 0), kažemo da je A *nenegativna (pozitivna) matrica.* Konačno, ako je $B = (b_{ij})$ proizvoljna realna ili kompleksna matrica reda $n \times r$, tada $|B|$ označava nenegativnu matricu sa elementima $|b_{ij}|$. ▲

Definicija 3.5. Za matricu $A = (a_{ij})$ reda $n \times n$ $(n \geq 2,)$ kaže se da je *nerazloživa* (nesvodljiva, nesparena) ako i samo ako za svaki par indeksa $i, j \in \{1, \ldots, n\}$ postoji niz nenula elemenata matrice A oblika

$$a_{ii_1}, a_{i_1 i_2}, \ldots, a_{i_m j} \quad (m \geq n). \ \blacktriangle$$

Ova definicija može se preformulisati na sledeći način: matrica A je nerazloživa ako za svaka dva indeksa i i j postoji ceo broj $m \geq 0$ i niz celih brojeva $i_1, i_2, \ldots, i_m$ tako da je proizvod $a_{ii_1} a_{i_1 i_2} \cdots a_{i_m j} \neq 0$.

Koncept nerazloživosti može se izraziti pomoću matrice permutacije.

Definicija 3.6. Za matricu $A = (a_{ij})_{n \times n}$ $(n \geq 2)$ kaže se da je razloživa (svodljiva) ukoliko postoji permutaciona matrica P reda $n \times n$ takva da važi

$$PAP^T = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

gde je A_{11} submatrica reda $r \times r$, a A_{22} je submatrica reda $(n-r) \times (n-r)$. U suprotnom kažemo da je matrica A nerazloživa (nesvodljiva). $\blacktriangle$

Da bismo razjasnili motivaciju za nazive matrica uvedene u Definiciji 3.6, posmatrajmo jedan linearni dinamički sistem. Varijacija vektora stanja (n-dimenzionalni vektor) linearnog dinamičkog sistema sa diskretnim vremenom, ima oblik

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k,$$

gde je A konstantna razloživa matrica. Tada, s obzirom na osobine blok-matrica, ovaj sistem može da se napiše u obliku

$$\begin{aligned} \boldsymbol{y}_{k+1} &= A_{11}\boldsymbol{y}_k + A_{12}\boldsymbol{z}_k \\ \boldsymbol{z}_{k+1} &= A_{22}\boldsymbol{z}_k, \end{aligned}$$

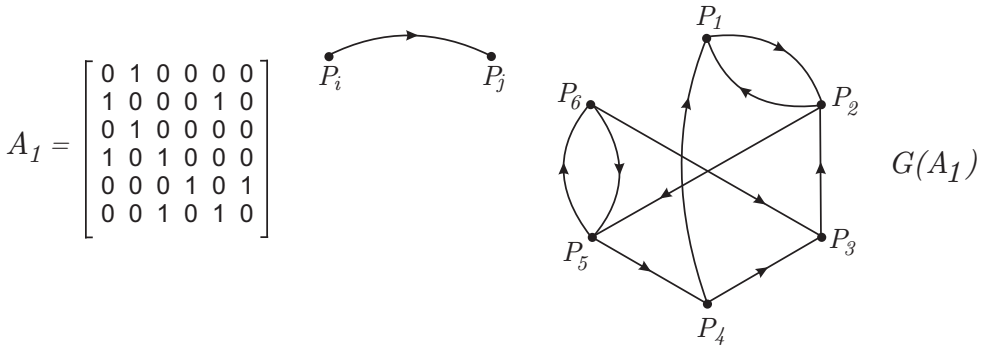
gde $\boldsymbol{y}$ sadrži prvih r komponenti vektora $\boldsymbol{x}$ a $\boldsymbol{z}$ sadrži poslednjih $n - r$ komponenti originalnog vektora $\boldsymbol{x}$. Stoga, rešavanjem druge jednačine po $\boldsymbol{z}$, unošenjem tog rešenja u prvu jednačinu i njenim rešavanjem po $\boldsymbol{y}$, polazni sistem je *razložen* na dva jednostavnija sistema matričnih jednačina *nižeg* reda. Inženjeri i fizičari koriste termin da je sistem parcijalno *sparen.* Prema tome, reći da je matrica A nerazloživa, znači da se sistem ne može razložiti; mora se tretirati u celosti da bi se izučilo njegovo ponašanje.

Koncept nerazloživosti može se veoma korisno interpretirati geometrijski pomoću teorije grafova. Neka je $A = (a_{ij})$ matrica reda $n \times n$, i neka je dat skup od n različitih tačaka $P_1, P_2, \ldots, P_n$ u ravni, koje ćemo zvati *čvorovi*. Za svaki nenula element a_{ij} matrice, čvor P_i povezujemo s čvorom P_j pomoću *orijentisanog luka* $\overrightarrow{P_iP_j}$, sa smerom od P_i ka P_j, kao što je prikazano na slici 3.1b. Na ovaj način svakoj matrici A reda $n \times n$ može se pridružiti jedan *konačan usmeren graf* $G(A)$. Na primer, matrica A_1 (slika 3.1a) ima usmeren graf prikazan na slici 3.1c.

Definicija 3.7. Usmeren graf sa n čvorova zove se *strogo povezan* ako za svaki uređen par čvorova (P_i, P_j), gde je $1 \leq i, j \leq n$, postoji *usmeren graf* koji se sastoji od usmerenih lukova

$$\overrightarrow{P_iP_{k_1}}, \overrightarrow{P_{k_1}P_{k_2}}, \ldots, \overrightarrow{P_{k_{r-1}}P_{k_r=j}},$$

koji se nadovezuju jedan na drugi i koji spajaju P_i sa P_j. ▲



a) b) c)

Slika 3.1 Usmeren graf matrice A_1.

Ekvivalencija matrične osobine nerazloživosti iz Definicije 3.5 i koncepta strogo povezanog grafa data je sledećom teoremom.

Teorema 3.2. (Varga [144]) *Matrica A reda $n \times n$ je nerazloživa ako i samo ako je njen usmeren graf $G(A)$ strogo povezan.* ▲

Pri ispitivanju konvergencije iterativnih procesa važnu ulogu imaju norme vektora i norme matrica. U ovom poglavlju koristićemo sledeće norme ma-

trice $A = (a_{ij})_{n\times n}$:

$$1^\circ \qquad \|A\|_1 = \max_j\Big(\sum_{i=1}^{n} |a_{ij}|\Big),$$

$$2^\circ \qquad \|A\|_2 = \Big(\sum_{i,j=1}^{n} |a_{ij}|^2\Big)^{1/2},$$

$$3^\circ \qquad \|A\|_\infty = \max_i\Big(\sum_{j=1}^{n} |a_{ij}|\Big).$$

Pri razmatranju konvergencije procesnih iteracija ključnu ulogu ima sledeća teorema.

Teorema 3.3. (Perron-Frobenius) *Neka je A nenegativna i nerazloživa matrica reda $n \times n$. Tada*

1) *A ima pozitivnu realnu sopstvenu vrednost koja je jednaka njenom spektralnom radijusu.*

2) *Radijusu $\rho(A)$ odgovara pozitivan sopstveni vektor.*

3) *$\rho(A)$ se povećava ukoliko se poveća vrednost bilo kog elementa matrice A.*

4) *$\rho(A)$ je prosta (jednostruka) sopstvena vrednost matrice A.* ▲

Definicije i teoreme, date u nastavku, odnose se na konvergenciju matričnih nizova i redova, kao i na pojam konvergentne matrice.

Definiciaj 3.7. Ako je $A^{(0)} = (a_{ij}^{(0)})$, $A^{(1)} = (a_{ij}^{(1)}), \ldots$ beskonačan niz matrica reda $n \times n$, kaže se da taj niz *konvergira* ka matrici $A = (a_{ij})$ reda $n \times n$ ukoliko

$$\lim_{m\to\infty} a_{ij}^{(m)} = a_{ij} \quad \text{za svako } 1 \le i, j \le n.$$

Slično, pod konvergencijom beskonačnog reda $\sum_{m=0}^{\infty} B^{(m)}$ matrica $B^{(m)} = (b_{ij}^{(m)})$ ka matrici $B = (b_{ij})$, podrazumeva se

$$\lim_{N\to\infty} \sum_{m=0}^{N} b_{ij}^{(m)} = b_{ij} \quad \text{za svako } 1 \le i, j \le n. \ ▲$$

Definicija 3.8. Neka je A matrica reda $n \times n$. Kaže se da je A *konvergentna* (*ka nuli*) ako niz matrica $A, A^2, A^3, \ldots$ konvergira ka nula-matrici O, a da je *divergentna* u suprotnom. ▲

Teorema 3.4. *Matrica A reda $n \times n$ je konvergentna ako i samo ako je $\rho(A) < 1$.* ▲

Teorema 3.5. *Ako je A proizvoljna realna ili kompleksna matrica reda $n \times n$ sa $\rho(A) < 1$, tada je $I - A$ regularna i važi*

$$(I - A)^{-1} = I + A + A^2 + A^3 + \cdots ,$$

tj., zbir na desnoj strani konvergira ka $(I - A)^{-1}$.

Brzina konvergencije procesnih iteracija razmatranih u Odeljku 3.3 može se oceniti koristeći pojam usrednjene brzine konvergencije koji je uveo Varga u [144] posmatrajući generalisani iterativni metod

$$\boldsymbol{x}^{(k+1)} = A\boldsymbol{x}^{(k)} + \boldsymbol{g} \quad (k = 0, 1, \ldots).$$

Definicija 3.9. Neka su A i B dve matrice reda $n \times n$. Ako je $\|A^k\| < 1$ za neki prirodan broj k, tada se

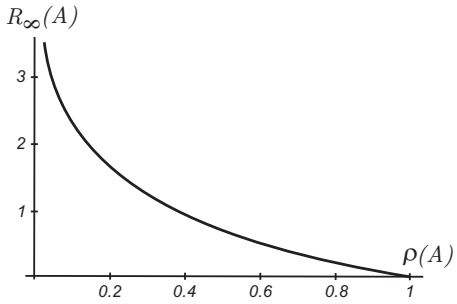
$$R(A^k) := -\ln\left[(\|A^k\|)^{1/k}\right] = \frac{-\ln \|A^k\|}{k}$$

zove *prosečna brzina konvergencije* za k iteracija matrice A. Ako je $R(A^k) < R(B^k)$, tada je B *iterativno brža*, za k iteracija, nego A. ▲

Teorema 3.6. (Varga [144, str. 73]). *Neka je A konvergentna matrica reda $n \times n$. Tada prosečna brzina konvergencije $R(A^k)$ za k iteracija zadovoljava relaciju*

$$\lim_{k\to\infty} R(A^k) = -\ln \rho(A) =: R_\infty(A). \quad ▲$$

Varga [144] je nazvao $R_\infty(A)$ *asimptotskom brzinom konvergencije*, dok je Young [156] koristio naziv *brzina konvergencije*. Zavisnost asimptotske brzine konvergencije $R_\infty(A)$ od spektralnog radijusa $\rho(A)$ data je na slici 3.2. Očigledno je da je konvergencija utoliko sporija ukoliko je spektralni radijus posmatrane matrice bliži 1.



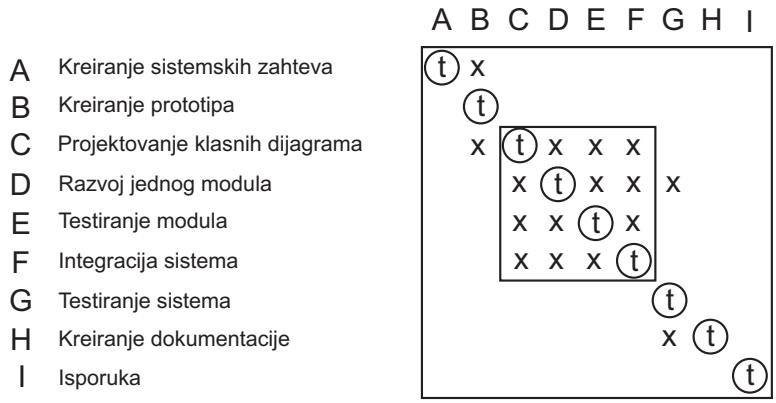
Slika 3.2 Zavisnost asimptotske brzine konvergencije $R_\infty(A)$ od $\rho(A)$

3.3 Model procesne iteracije

Modeli procesne iteracije mogu obezbediti koristan uvid u iterativni proces projektovanja. Iterativni model koji je ovde prikazan daje informacije o tome koje aktivnosti u složenom i povezanom procesu mogu najviše doprineti poboljšanju projektno-razvojnog procesa. Rešenja koja daje ovaj model mogu uključiti dodatne izvore informacija, restruktuiranje procesa, obezbediti nove inženjerske alate, redefinisati probleme, ograničiti ciljeve razvojnih zahteva, i sl. Matrica strukture procesa predstavlja osnovu naše formalne analize i biće ukratko opisana. (Za detaljniji opis videti [131] i [40]). Strategija MSP metoda sastoji se u tome da se razvojni projekat podeli na individualne taskove, a zatim se međusobne relacije između ovih taskova analiziraju sa ciljem da se identifikuje karakteristična struktura projekta. Izučavanje relacija između pojedinačnih taskova može poboljšati celokupan razvojni proces i omogućiti analizu alternativne razvojne strategije (von Hipel [145]). Primer na kome ćemo ilustrovati navedeni pristup odnosi se na razvoj softvera. Taskovi $A, B, \ldots, I$ su dati u sledećoj listi:

A Kreiranje sistemskih zahteva
B Kreiranje prototipa
C Projektovanje klasnih dijagrama
D Razvoj jednog modula
E Testiranje modula
F Integracija sistema
G Testiranje sistema
H Kreiranje dokumentacije
I Isporuka

Specifični razvojni taskovi su aranžirani u kvadratnu matricu A gde je svaka vrsta i njoj odgovarajuća kolona pridružena jednom od taskova. Ovu matricu zovemo *matrica strukture procesa*, skraćeno MSP. Duž svake vrste oznake "x" ukazuju od kojih drugih taskova dati task zahteva ulazne podatke koji predstavljaju meru njihove zavisnosti i izražavaju se u procentima. Na primer, ako ova mera zavisnosti iznosi 15%, tada je $x = 0.15$; u opštem slučaju, $x \in (0, 1)$. Oznake "x" duž svake kolone j ukazuju koji drugi taskovi dobijaju ulazne podatke od taska j. Za dijagonalne elemente MSP usvaja se $x = 0$ jer task ne može da zavisi od svog sopstvenog završetka. Uvođenjem dijagonalne matrice T sa dijagonalnim elementima koji označavaju vremena potrebna za izvršavanje pojedinih taskova (elementi "t" na glavnoj dijagonali, sl. 3.3), superpozicijom $A + T = W$ dobija se *matrica transformacije rada*. Na primer, na slici 3.3 prikazana je matrica transformacije rada koja opisuje uprošćenu proceduru razvoja softvera. U ovom procesu task C zahteva ulaz od taskova B, D, E i F, task A zahteva ulaz samo od od taska B, a task B ne zahteva nikakve ulaze da bi započeo. Submatrica 4×4 sa slike 3.3 opisuje izdvojenu grupu taskova, tzv. *procesni mod*. Ovi taskovi su dovoljno složeni i međusobno povezani da je neophodan iterativni postupak da bi se taskovi završili. MSP se može koristiti za identifikaciju prioritetnih taskova, njihovo rangiranje i za identifikaciju problematičnih taskova i koraka u razvojnom procesu.



Slika 3.3 Matrični model procesne iteracije – razvoj softvera

U našoj analizi pretpostavićemo da su taskovi i njihove međusobne relacije u razvojnom projektu poznate i nepromenljive tokom izvršenja projekta. Ovo

je logična pretpostavka jer proizvođač radi na razvojnom projektu u oblasti u kojoj je u značajnom stepenu već afirmisan. Pretpostavka nije dovoljno utemeljena za nove ili vrlo napredne tehnologije.

Razvojno vreme (dato dijagonalnim elementima matrice na slici 3.3) je značajna mera u upravljanju projektno-razvojnim procesom. Zahtevna, a samim tim i složena, procesna iteracija je glavni izvor uvećanog vremena razvoja. Korišćenje razvojnog vremena taskova datih pomoću MSP daje mogućnost dobre procene ukupnog trajanja projekta. Na taj način se može izvršiti identifikacija onih taskova u razvojnom procesu koji najviše povećavaju ukupno vreme iteracije.

Za modeliranje procesne iteracije koristimo nerazloživu matricu strukture projektovanja sa nenegativnim numeričkim elementima. Ova matrica zove se matrica transformacije rada (MTR)

$$W = \begin{bmatrix} t_1 & a_{12} & \cdots & a_{1n} \\ a_{21} & t_2 & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & t_n \end{bmatrix} = T + A,$$

gde je $T = \operatorname{diag}\{t_1, t_2, \ldots, t_n\}$ i

$$A = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}.$$

Postoje dve vrste informacija u matrici transformacije rada W. Dijagonalni elementi matrice T (slika 3.4b) predstavljaju vreme potrebno za završetak svakog taska tokom prve iteracije. Vandijagonalni elementi matrice sturkture projektovanja A (slika 3.4a) predstavljaju stepen zavisnosti između taskova, što utiče na prenos rada (ili dorade) među iteracijama. Preciznije, svaki element a_{ij} matrice A označava da jedna jedinica rada na projektnom tasku j proizvodi a_{ij} jedinica dodatnog rada (dorade) na projektnom tasku i. Dijagonalni elementi matrice A jednaki su nuli.

(a) Međusobna zavisnost taskova (b) Vremena izvršavanja taskova

Slika 3.4 Matrica transformacije rada

U našoj analizi pretpostavljamo da svaki task zahteva određenu količinu dodatnog rada na drugim taskovima. Dodatni rad (prepravka) je neophodno ponavljanje rada na tasku zbog prvobitnog neuspeha prouzrokovanog neadekvatnim informacijama (pretpostavkama). Prepravka stoga predstavlja evolutivno rešenje jer obuhvata modifikovanu informaciju.

Model matrice transformacije rada (MTR) podrazumeva da važe sledeće pretpostavke koje nam omogućuju analizu primenom linearne algebre za MTR i MSP:

1) Svi taskovi se izvršavaju u svakom koraku – potpuno paralelna iteracija.

2) Model je napravljen tako da je matrica strukture procesa (MSP) nenegativna i nerazloživa.

3) Dodatni rad koji se vrši je funkcija rada izvršenog u prethodnom iterativnom koraku.

4) Parametri transformacije rada u matrici ne menjaju se s vremenom.

Ove pretpostavke detaljno su diskutovane u [128].

Prva pretpostavka, da se svi upareni taskovi (taskovi koji pripadaju istoj grupi u okviru razmatranog globalnog projektnog moda) izvršavaju u svakom koraku, je idealizacija koja je učinjena na osnovu praktičnog iskustva u mnogim razvojnim projektima. Ovo podrazumeva da se tim izvršilaca ne menja i da svi rade na skupu međusobno povezanih delova razvoja. Ova situacija je uobičajena u inženjerskoj praksi. Druga pretpostavka omogućuje da se procesne iteracije mogu okončati u nekoliko iterativnih koraka, tj. da je iterativni projektno-razvojni proces konvergentan.

Treća i četvrta pretpostavka da je dodatni rad linearna funkcija rada u prethodnim koracima i da količine dodatnog rada ne variraju sa vremenom su pogodne sa matematičke tačke gledišta ali nisu uvek u potpunosti ispunjene u praksi. Naime, u praksi se količina vremena po iteraciji zapravo smanjuje (projektno-razvojne procesne iteracije konvergiraju s vremenom). Dalje, ako je broj iteracija relativno mali (projektno-razvojni proces je stabilan), veći deo dodatnog rada je završen u prvih nekoliko iteracija. S obzirom na ove argumente, treća i četvrta pretpostavka su prihvatljive.

Da bismo opisali model procesne iteracije, najpre ćemo uvesti pojam radnog vektora $\boldsymbol{u}_k$. To je n-dimenzionalni vektor, gde je n broj sparenih projektnih taskova koje treba obaviti. Svaki element radnog vektora sadrži količinu rada (izraženu u procentima) koji treba obaviti na svakom tasku nakon k-te iteracije. Početni radni vektor $\boldsymbol{u}_0 = (1, 1, \ldots, 1)$ je vektor jedinica, što ukazuje na činjenicu da na početku iterativnog procesa treba obaviti celokupan rad na svakom tasku, odnosno, preostali rad na svim taskovima je 100%.

Tokom svakog iterativnog koraka obavlja se celokupan rad na svim projektnim taskovima. Međutim, rad na nekom tasku i prouzrokovaće potrebu za ponovnim radom (prepravkama) na svim ostalim taskovima čija će količina zavisiti od informacije koja se odnosi na posmatrani task i. Matrica strukture procesa $A = (a_{ij})$ sadrži informaciju o ovoj zavisnosti. Svaki od elemenata a_{ij} matrice A označava da izvršenje jedne jedinice rada na tasku j prouzrokuje a_{ij} jedinica dorade na tasku i. Stoga matrica A definiše nivo zavisnosti između taskova (slika 3.4a). Na primer, $a_{ij} = 0.5$ označava visok nivo zavisnosti dok $a_{ij} = 0.05$ predstavlja slabu zavisnost.

Nakon prve iteracije, preostala količina rada iznosi $\boldsymbol{u}_1 = A\boldsymbol{u}_0$. Posle druge iteracije preostali rad je $\boldsymbol{u}_2 = A \cdot A\boldsymbol{u}_0 = A^2\boldsymbol{u}_0$. Posle k-te iteracije, preostali rad je $\boldsymbol{u}_k = A^k\boldsymbol{u}_0$. Svaki iterativni korak prouzrokuje promenu u radnom vektoru prema formuli

$$\boldsymbol{u}_k = A\boldsymbol{u}_{k-1} = A^k\boldsymbol{u}_0. \tag{3.1}$$

Primetimo da ovaj model ima analogiju sa dinamičkim linearnim sistemima.

Zbir svih vektora rada daje *vektor ukupnog rada* $\boldsymbol{u}$, koji sadrži informaciju o tome koliko je ukupno rađeno na svakom od taskova za vreme svih m iterativnih koraka u projektno-razvojnom procesu:

$$\boldsymbol{u} = \sum_{k=0}^{m} \boldsymbol{u}_k = \sum_{k=0}^{m} A^k\boldsymbol{u}_0 = \left(\sum_{k=0}^{m} A^k\right)\boldsymbol{u}_0. \tag{3.2}$$

Izlazna veličina $\boldsymbol{u}$ ovog modela stoga je izražena u jedinicama početne količine rada izvršene na svakom od taskova u prvom iterativnom koraku. Na primer, ako i-ti element vektora $\boldsymbol{u}$ iznosi 1.6, to znači da će biti potrebno 60% dorade na tasku i u sledećim iterativnim koracima. Ako je T dijagonalna matrica čiji elementi predstavljaju vremena izvršenja taskova (slika 3.4b), tada množenjem vektora rada $\boldsymbol{u}$ matricom T, dobijamo vektor $T\boldsymbol{u}$ koji sadrži ukupnu količinu vremena neophodnu za izvršenje svakog od taskova tokom m iterativnim koraka.

Pretpostavimo da matrica strukture procesa A ima linearno nezavisne sopstvene vektore $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ koji odgovaraju sopstvenim vrednostima λ_1, $\lambda_2, \ldots, \lambda_n$ matrice A. Tada je odgovarajuća matrica sopstvenih vektora $S = (\boldsymbol{v}_i)$ invertibilna a matrica A može se razložiti na sledeći način

$$A = SLS^{-1},$$

gde je $L = \operatorname{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ dijagonalna matrica sopstvenih vrednosti matrice A. Na osnovu toga sledi da je $A^k = SL^kS^{-1}$ tako da se iz (3.2) dobija vektor ukupnog rada $\boldsymbol{u}$, izražen kao

$$\boldsymbol{u} = S\left(\sum_{k=0}^{m} L^k\right)S^{-1}\boldsymbol{u}_0. \tag{3.3}$$

Na osnovu Teoreme 3.4 sledi da matrična geometrijska progresija $I + A + A^2 + \cdots$ konvergira ako su sve sopstvene vrednosti matrice A manje od 1 po apsolutnoj vrednosti, odnosno, ako je spektralni radijus $\rho(A)$ manji od 1. Isto važi za geometrijsku progresiju $I + L + L^2 + \cdots$, s obzirom da je $\rho(A) = \rho(L)$. Shodno uslovima 1)–4) iz postavke modela matrice transformacije rada, na osnovu Teoreme 3.4 sledi da iterativni proces projektovanja konvergira, odnosno, vektor ukupnog rada $\boldsymbol{u}$ ostaje ograničen kad se m uvećava ka beskonačnosti. U tom slučaju dobijamo

$$\lim_{m\to\infty}\sum_{k=0}^{m} L^k = (I - L)^{-1},$$

tako da na osnovu (3.3) sledi

$$\boldsymbol{u} = S(I - L)^{-1}S^{-1}\boldsymbol{u}_0. \tag{3.4}$$

Interpretacija sopstvenih vrednosti i sopstvenih vektora za modele procesnih iteracija je slična analizi sopstvene strukture koja se koristi za ispitivanje dinamičkih karakteristika nekog fizičkog sistema. U opisu linearnih dinamičkih sistema sa diskretnim vremenom svaka sopstvena vrednost odgovara brzini konvergencije jednog od modova sistema (prirodna frekvencija koja određuje raspad ili oscilaciju moda). Sopstveni vektori određuju prirodu i oblike moda prirodnog kretanja, kvantifikujući učešće svake od promenljivih stanja u svakom modu. Korišćenje ove analize u cilju uočavanja primarnog ponašanja jednog složenog sistema takođe se koristi i u drugim oblastima.

Pre diskusije o konvergenciji niza radnih vekora $\{\boldsymbol{u}_k\}$ definisanih formulom (3.1), napomenimo da sopstvena vrednost veća od 1 odgovara razvojnom procesu gde izvršenje jedne jedinice rada u nekom tasku za vreme nekog iterativnog koraka prouzrokuje više od jedne jedinice rada za taj task u sledećem koraku. Takav sistem je nestabilan i vektor $\boldsymbol{u}$ neće konvergirati već će se njegovi elementi beskonačno uvećavati sa porastom broja iteracija m.

Razvojni proces koji ne konvergira bio bi, na primer, onaj gde ne postoji ostvarivo tehničko rešenje za date specifikacije, ili gde projektanti nisu voljni na kompromis radi postizanja rešenja. Ako razvojni proces ne konvergira bilo bi adekvatno napustiti projekat ili prilagoditi specifikacije i restruktuirati problem tako da proces postane stabilan.

Posmatrajmo izdvojeni projektni mod sa matricom transformacije rada W i matricom strukture procesa A. Razmotrićemo kako se karakteristike ovog projektnog moda mogu interpretirati pomoću sopstvenih vrednosti i sopstvenih vektora matrice A. Kao što je napomenuto ranije, sopstvene vrednosti i sopstveni vektori matrice A određuju brzinu i prirodu konvergencije razvojnog procesa. Apsolutna vrednost svake sopstvene vrednosti matrice A određuje geometrijsku brzinu konvergencije jednog od projektnih modova (Strang [133]). Sopstveni vektor koji odgovara nekoj sopstvenoj vrednosti definiše relativan doprinos svakog od različitih taskova intenzitetu rada.

Pri definisanju matrice strukture rada A rečeno je da je ona nenegativna i nerazloživa. Na osnovu Peron-Frobenijusove teoreme (Teorema 3.2) to znači da postoji tačno jedna pozitivna sopstvena vrednost koja je najveća po modulu (jednaka spektralnom radijusu $\rho(A)$, videti Definiciju 3.4) i da njoj odgovara sopstveni vektor čiji su elementi pozitivni. Pretpostavljajući da je ova sopstvena vrednost manja od 1, može se reći da ona dominantno utiče na brzinu konvergencije niza $\{A^k\}$, a time i niza radnih vektora $\{\boldsymbol{u}_k\}$

(na osnovu (3.1)). Na osnovu Teoreme 3.5 i slike 3.2 sledi da što je njena vrednost veća (ali ne prelazi 1), to je konvergencija sporija tako da takvom projektnom modu treba posvetiti posebnu pažnju kod rekonstrukcije modela. Kasnije ćemo pokazati da ukoliko su vrednosti elemenata u odgovarajućem pozitivnom sopstvenom vektoru veće, to je značajniji uticaj tog elementa (i odgovarajućeg taska) na taj mod.

U nastavku ćemo izvršiti analizu značenja sopstvenih vrednosti kada su one predstavljene negativnim i kompleksnim brojevima.

Smith i Eppinger [128] polaze od izraza za vektor ukupnog rada (3.4) i posmatraju izraz $(I-L)^{-1} = \text{diag}\,\{1/(1-\lambda_1), \ldots, 1/(1-\lambda_n)\}$ kao dijagonalnu *težinsku matricu*. Oni koriste dijagonalne elemente ove matrice da procene (i čak rangiraju) nivo uticaja projektno-razvojnih taskova, ali bez dublje analize koja bi dovela do zaključaka. Štaviše, oni koriste izraze dobijene u graničnom procesu iako se radi o konačnom iterativnom procesu koji se u praksi završava u nekoliko koraka. Njihovi finalni zaključci su ipak korektni zahvaljujući činjenici da je funkcija $\lambda \mapsto 1/(1-\lambda)$ rastuća na intervalu (0,1). Na ovaj način, najveća vrednost, recimo λ_m (koja ima dominantan uticaj) dovodi to toga da je odgovarajuća vrednost $1/(1-\lambda_m)$ takođe najveća. U nastavku će biti izvršena analiza i dat dublji uvid u interpretaciju sopstvenih vrednosti.

Neka su $\lambda_1, \ldots, \lambda_n$ sopstvene vrednosti neke matrice A reda $n \times n$ i neka su $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ njima odgovarajući sopstveni vektori. Pretpostavimo da su ovi vektori linearno nezavisni, što znači da mogu predstavljati bazu u $\mathbb{R}^n$. Tada proizvoljan nenula vektor $\boldsymbol{u}_0$ može da se predstavi u obliku

$$\boldsymbol{u}_0 = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i, \tag{3.5}$$

gde su α_i skalari koji zavise od početnih uslova. Konkretno, u našem slučaju $\boldsymbol{u}_0 = (1, \ldots, 1)$ je početni vektor rada a skalari α_i se određuju iz sistema linearnih jednačina koji se dobija iz (3.5) zamenom $\boldsymbol{u}_0 = (1, \ldots, 1)$.

Posmatrajmo procesnu iteraciju u obliku

$$\boldsymbol{u}_k = A\boldsymbol{u}_{k-1} = A^k \boldsymbol{u}_0 \quad (\boldsymbol{u}_0 = (1, \ldots, 1)). \tag{3.6}$$

Kako je $A\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$ (videti Definiciju 3.2), na osnovu (3.5) i (3.6) nalazimo

$$\boldsymbol{u}_k = \sum_{i=1}^{n} \alpha_i A^k \boldsymbol{v}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k \boldsymbol{v}_i. \tag{3.7}$$

Tada je vektor ukupnog rada $\boldsymbol{u}$ dat pomoću

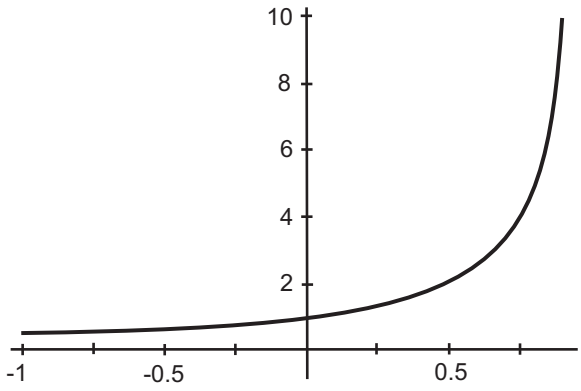
$$\boldsymbol{u} = \sum_{k=0}^{m} \boldsymbol{u}_k = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \sum_{k=0}^{m} \lambda_i^k$$

Pretpostavimo da m teži ka beskonačnosti i da je $|\lambda_i| < 1$. Tada je

$$\boldsymbol{u} = \sum_{i=1}^{n} \frac{\alpha_i}{1-\lambda_i} \boldsymbol{v}_i. \tag{3.8}$$

Iz relacije (3.8) vidi se da faktori $1/(1-\lambda_i)$ imaju direktan uticaj na vektor ukupnog rada $\boldsymbol{u}$. Međutim, relacija (3.8) je previše složena da bi omogućila rangiranje projektno-razvojnih taskova prema vrednostima $1/(1-\lambda_i)$. S obzirom na (3.4) i težinsku matricu $(I-L)^{-1}$, koja ima sopstvene vrednosti $1/(1-\lambda_1), \ldots, 1/(1-\lambda_n)$ (prema Teoremi 3.1), Smith i Eppinger [129] su usvojili rangiranje na osnovu pojednostavljenog izraza $1/(1-\mathrm{Re}\,(\lambda_i))$.

Za interpretaciju realnih sopstvenih vrednosti dovoljno je primetiti da je funkcija $1/(1-\lambda)$ strogo rastuća na $(-1, 1)$. Grafik ove funcije prikazan je na slici 3.5. Na osnovu grafika zaključujemo da taskovi s većom pozitivnom sopstvenom vrednošću više doprinose ukupnom radu nego taskovi s manjom ili negativnom sopstvenom vrednošću. Stoga, kad razmatramo koji su taskovi značajniji, treba uzeti u obzir samo one realne sopstvene vrednosti koje su pozitivne i velike.



Slika 3.5 Grafik funkcije $1/(1-\lambda),\ \lambda \in \mathbb{R}$

Što se tiče kompleksnih sopstvenih vrednosti, Smit i Epinger posmatraju apsolutnu vrednost izraza

$$\frac{1}{1-\lambda}, \quad \lambda = \alpha + i\beta$$

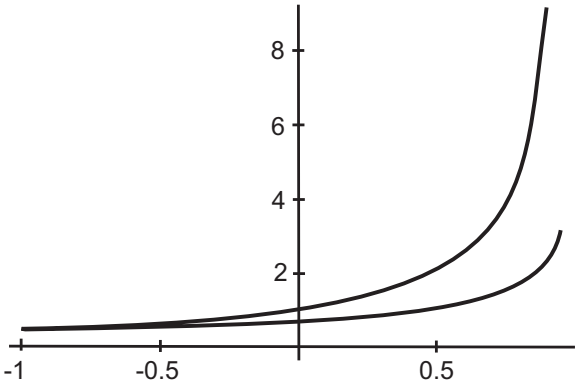
i za $\beta \neq 0$ i $\alpha^2 + \beta^2 < 1$ nalaze gornju i donju granicu:

$$\left|\frac{1}{1-(\alpha+i\beta)}\right| = \frac{1}{\sqrt{(1-\alpha)^2+\beta^2}} < \frac{1}{1-\alpha} \tag{3.9}$$

i

$$\left|\frac{1}{1-(\alpha+i\beta)}\right| > \frac{1}{\sqrt{2-2\alpha}}, \tag{3.10}$$

respektivno. Grafički prikaz ovih granica dat je na slici 3.6.



Slika 3.6 Granice apsolutnih vrednosti izraza $|1/(1-\lambda)|$, $\lambda \in \mathbb{C}$

Na osnovu (3.9) i (3.10) i slika 3.5 i 3.6 može se zaključiti da realni deo kompleksne sopstvene vrednosti daje granice apsolutne vrednosti izraza $1/(1-\lambda)$ koje odgovaraju toj sopstvenoj vrednosti. Takođe vidimo da kompleksna sopstvena vrednost sa negativnim realnim delom neće značajno doprinositi sumi, i stoga se može zanemariti. Na osnovu ovih argumenata Smith i Eppinger [128] zaključuju da je dovoljno razmatrati samo one sopstvene vrednosti sa relativno velikom realnom komponentom, bez obzira da li su realne ili kompleksne.

U nastavku dajemo precizniju analizu uticaja sopstvenih vrednosti na brzinu konvergencije procesne iteracije. Pretpostavimo da je λ_1 dominantna

sopstvena vrednost matrice strukture procesa A. Neka je $\lambda_{2,3}$ par konjugovano kompleksnih sopstvenih vrednosti λ i $\bar{\lambda}$ kojima odgovaraju sopstveni vektori $\boldsymbol{v}$ i $\bar{\boldsymbol{v}}$. Tada iz (3.7) nalazimo

$$\boldsymbol{u}_k = \alpha_1 \lambda_1^k \Big(\boldsymbol{v}_1 + \frac{\alpha}{\alpha_1} \Big(\frac{\lambda}{\lambda_1} \Big)^k \boldsymbol{v} + \frac{\bar{\alpha}}{\alpha_1} \Big(\frac{\bar{\lambda}}{\lambda_1} \Big)^k \bar{\boldsymbol{v}} + \sum_{i=4}^{n} \frac{\alpha_i}{\alpha_1} \Big(\frac{\lambda_i}{\lambda_1} \Big)^k \boldsymbol{v}_i \Big). \qquad (3.11)$$

Ako $|\lambda|$ nije blisko dominantnoj sopstvenoj vrednosti $|\lambda_1|$, tada iz (3.11) sledi

$$\boldsymbol{u}_k = \alpha_1 \lambda_1^k \Big(\boldsymbol{v}_1 + \boldsymbol{\varepsilon}_k \Big)$$

gde $\boldsymbol{\varepsilon}_k \to 0$ kada $k \to \infty$. Prema tome, izraženo dominantan uticaj na radni vektor $\boldsymbol{u}_k$ ima dominantna sopstvena vrednost λ_1. U suprotnom, ako je $|\lambda|$ blisko $|\lambda_1|$, količnici $\big(\frac{\lambda}{\lambda_1}\big)^k$ i $\big(\frac{\bar{\lambda}}{\lambda_1}\big)^k$ u (3.11) težiće ka 0 vrlo sporo.

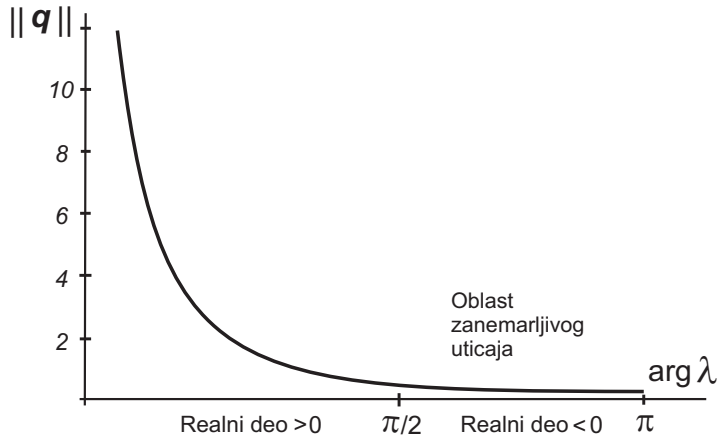
Vratimo se relaciji (3.8) i tumačenju kompleksnih sopstvenih vrednosti. Kompleksne sopstvene vrednosti javljaju se kao konjugovano kompleksni parovi pa se njihov uticaj mora anlizirati zajedno, tj. u parovima. Posmatrajamo par

$$\frac{\alpha \boldsymbol{v}}{1-\lambda} + \frac{\bar{\alpha} \bar{\boldsymbol{v}}}{1-\bar{\lambda}}$$

u (3.8), pri čemu su indeksi izostavljeni. Neka je $\boldsymbol{v} = (c_1, \ldots, c_n)$. Tada nalazimo

$$\boldsymbol{q} := \frac{\alpha}{1-\lambda}\boldsymbol{v} + \frac{\bar{\alpha}}{1-\bar{\lambda}}\bar{\boldsymbol{v}} = \left(2\mathrm{Re}\left(\frac{\alpha}{1-\lambda} c_1 \right), \ldots, 2\mathrm{Re}\left(\frac{\alpha}{1-\lambda} c_n \right) \right).$$

Analiza norme $\|\boldsymbol{q}\|$ izvršena je za različite vrednosti $|\lambda|$ (< 1) i za argument $\theta = \arg\lambda$ iz intervala $[0, \pi/2]$ ($\mathrm{Re}\,\lambda > 0$) i $[\pi/2, \pi]$ ($\mathrm{Re}\,\lambda < 0$). Osim od $|\lambda|$ i $\arg\lambda$, vrednost $\|\boldsymbol{q}\|$ zavisi od komponenata sopstvenog vektora $\boldsymbol{v}$ na jedan složen način. Testiranjem većeg broja numeričkih primera, može se zaključiti da tipična kriva zavisnosti $\|\boldsymbol{q}\| = f(|\lambda|, \arg\lambda)$ ima oblik prikazan na slici 3.7. Primećujemo da kompleksna sopstvena vrednost ima značajan uticaj na vektor ukupnog rada $\boldsymbol{u}$ jedino ako joj je apsolutna vrednost bliska 1 a njen realan deo je pozitivan. U slučaju kompleksne sopstvene vrednosti sa negativnim realnim delom, njen uticaj je zanemarljiv čak i u slučaju velike apsolutne vrednosti (ali manje od 1).



Slika 3.7

Pozitivne realne sopstvene vrednosti odgovaraju neoscilatornim razvojnim modovima. Negativne i kompleksne sopstvene vrednosti opisuju prigušene oscilacije. Oscilatorni razvojni modovi ukazuju da se rad ne smanjuje za sve taskove u modu za isti iznos, već da će se rad pomerati od taska do taska za vreme iterativnog procesa. Promene u količini rada između odvojenih radnih vektora $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ nisu tako značajne kao ukupna količina izvršenog rada.

3.4 Rangiranje taskova

Smith i Eppinger [129] razmatraju tri metoda za rangiranje projektno-razvojnih taskova. Prvi je korišćenje apsolutnih vrednosti elemenata matrice $(I - L)^{-1}$. Drugi je na osnovu članova $(I - L)^{-1}S^{-1}u_0$. Treći je na osnovu toga koliko svaki task zaista učestvuje u ukupnom radnom vektoru $\boldsymbol{u}$. Na osnovu diskusije izložene u radu [129] oni sugerišu rangiranje taskova pomoću izraza $1/(1-Re(\lambda))$ (prvi metod). Međutim, rangiranje taskova na osnovu vrednosti $1/(1 - \mathrm{Re}\,(\lambda))$ nije dovoljno dobar metod ne samo zato što $1/(1 - \mathrm{Re}\,(\lambda))$ predstavlja samo gornju granicu za $|1/(1-\lambda)|$ (uzgred, ne dovoljno oštru), već zbog činjenice da, osim dominantnog uticaja sopstvene vrednosti sa najvećom apsolutnom vrednošću, o uticaju ostalih sopstvenih vrednosti je teško govoriti zbog izuzetno složenih relacija oblika (3.8).

Da bismo došli do dobro obrazloženog odgovora na pitanje kako rangirati taskove, najpre ukratko izlažemo Kinerov metod dat u radu [77].

Neka su $P_1, \ldots, P_n$ učesnici u igri, inženjerskom dizajnu, takmičenju ili nekom drugom događaju čiji ishod zavisi od uticaja P-ova, pretpostavljajući da postoje interakcije između učesnika. Uzmimo kao primer da su $P_1, \ldots, P_n$ učesnici u igri. Neka su a_{ij} nenegativni brojevi koji zavise od ishoda igre između učesnika i i učesnika j podeljenog brojem igara m_i koje će odigrati učesnik i. Matrica $A = (a_{ij}/m_i)$ često se zove *matrica prednosti.* Zadatak se sastoji u tome da se nađe dobra šema rangiranja koja će dati, manje ili više objektivno, redosled učesnika uzimajući u obzir i ishode i međusobni odnos snaga između učesnika.

Kao što je pokazano u [77], postoje različite šeme rangiranja koje mogu otkloniti neke, ali ne sve, subjektivne činioce. Ove šeme mogu dati različite odgovore o tome koja je informacija najvažnija od svih koje se koriste u šemi. Ovde će biti predstavljen jednostavan direktan metod koji je predložen u [77]. Ispitujemo snagu/kvalitet r_j učesnika j i formiramo vektor rangiranja $\boldsymbol{r} = (r_1, \ldots, r_n)$. Komponente vektora rangiranja $\boldsymbol{r}$ određuju redosled učesnika na rang listi.

Najpre definišemo učinak/doprinos učesnika i kao

$$s_i = \frac{1}{m_i} \sum_{j=1}^{n} a_{ij} r_j.$$

Kao što je predloženo u [77], rang učesnika trebalo bi da bude proporcionalan njegovom učinku, odnosno,

$$A\boldsymbol{r} = \lambda \boldsymbol{r}.$$

Drugim rečima, vektor rangiranja $\boldsymbol{r}$ je pozitivan sopstveni vektor nenegativne matrice A, shodno Definiciji 3.1.

Vratimo se sada problemu rangiranja taskova. Kako je matrica strukture procesa A nenegativna i nerazloživa, prema Perron-Frobeniusovoj teoremi (Teorema 3.2) matrica A ima jednu pozitivnu sopstvenu vrednost sa najvećom apsolutnom vrednošću (jednaku spektralnom radijusu $\rho(A)$) i odgovarajući pozitivan vektor. Ovaj pozitivan vektor, koji smo označili sa $\boldsymbol{r}$, upravo daje rangiranje taskova.

Izvodimo sledeći zaključak: najveća sopstvena vrednost, recimo λ_m, ima dominantnu ulogu u geometrijskoj brzini konvergencije procesne iteracije. i-ta koordinata odgovarajućeg sopstvenog vektora $\boldsymbol{v}_m$, koji odgovara sopstvenoj vrednosti λ_m $(= \rho(A))$, karakteriše relativan doprinos projektnog

taska i vektoru ukupnog rada, odnosno, projektnom modu. Što je veći element u pozitivnom sopstvenom vektoru $\boldsymbol{v}_m$, tim značajnije taj element (task) doprinosi tom modu. Ostali projektni taskovi manje su očigledni. Prema tome, rangiranjem koordinata odgovarajućeg sopstvenog vektora vršimo rangiranje doprinosa svakog od taskova. U tom procesu vršimo identifikaciju kontrolnih osobina svakog taska. Ovi zaključci ilustrovani su u primeru 3.1.

Za izračunavanje sopstvene vrednosti $\boldsymbol{r}$ može se primeniti metod stepenovanja (videti, [132], [144]), koji se zasniva na graničnom procesu

$$\lim_{k\to\infty} \frac{A^k \boldsymbol{r}_0}{|A^k \boldsymbol{r}_0|} = \boldsymbol{r}$$

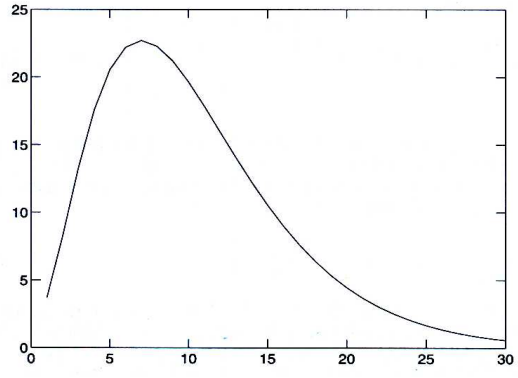
za proizvoljan nenegativan vektor $\boldsymbol{r}_0$.

U tačnoj aritmetici (aritmetici beskonačne preciznosti), granično ponašanje stepena matrice $A \in \mathbb{C}^{n\times n}$ je definisano pomoću sopstvenih vrednosti matrice A. Ako je spektalni radijus $\rho(A)$ manji od 1 tada $A^k \to 0$ kada $k \to \infty$, a ako je $\rho(A) > 1$ tada $\|A^k\| \to \infty$ kada $k \to \infty$. U prvom slučaju kaže se da je A *konvergentna matrica*. Dok spektralni radijus određuje asimptotsku brzinu rasta/opadanja matričnih stepena, norma ukazuje na početno ponašanje stepena. Rezultat Horna i Johnsona [71] da je $\rho(A) = \lim_{k\to\infty} \|A\|^{1/k}$ za bilo koju normu potvrđuje ulogu spektralnog radijusa u graničnom slučaju.

S obzirom da je izračunavanje norme $\|A\|$ jednostavnije nego izračunavanje spektralnog radijusa matrice A, postavlja se pitanje da li se pri ispitivanju konvergencije matričnog reda može koristiti norma umesto spektralnog radijusa. Odgovor je porvrdan, ali samo u slučaju da je $\|A\| < 1$. Kao što je poznato, s obzirom da je $\rho(A) \leq \|A\|$ može se desiti da je $\rho(A) < 1$ i da je matrica A konvergentna a da je pritom $\|A\| > 1$. U ovom slučaju javlja se tzv. *efekat grbe* date sa $\max_k \|A^k\|$ koja može biti i veoma velika. Slika 3.8 pokazuje grbu za 3×3 gornju trougaonu matricu A datu sa

$$A = \begin{bmatrix} 3/4 & 2 & 2 \\ 0 & 3/4 & 2 \\ 0 & 0 & 3/4 \end{bmatrix}.$$

Za ovu matricu je $\rho(A) = 0.75 < 1$ i matrica A je konvergentna, dok je $\|A\| \approx 3.7$. Napomenimo da za matrice kod kojih je $\rho(A) \leq 1$ i $\|A\| < 1$ (tzv. *normalne matrice*) ovaj efekat se ne javlja.



Slika 3.8 Norma $\|A^k\|_2$ (y-osa) kao funkcija eksponenta k (x-osa)

U slučaju izračunavanja u aritmetici konačne preciznosti (npr. u APT), zbog pojave greške zaokruživanja uslov za konvergenciju se mora modifikovati. U slučaju jedne široke klase matrica, dovoljan uslov za konvergenciju stepene matrica $n \times n$ dat je sa

$$\rho(A) < \frac{1}{1+\gamma_{n+2}}, \quad \gamma_n = \frac{n\varepsilon}{1-n\varepsilon},$$

gde je ε mašinska preciznost upotrebljene aritmetike, tj. *jedinica zaokruživanja*. Međutim, u opštem slučaju $\rho(|A|)$ može nadmašiti $\rho(A)$ i poslednji uslov ne važi. U ovom opštem slučaju neophodno je pribaviti dodatne informacije o matrici. Dovoljni uslovi za konvergenciju u prilično složenom obliku, zasnovani na Žordanovoj kanoničnoj formi, razmatrani su u knjizi [67].

3.5 Numerički primer

Primer 3.1. Kao ilustraciju prethodne interpretacije sopstvenih vrednosti i sopstvenih vektora, posmatrajmo 4×4 matricu transformacije rada

$$W = \begin{bmatrix} 12 & 0.2 & 0.1 & 0.2 \\ 0.25 & 15 & 0.35 & 0.1 \\ 0.2 & 0.25 & 17 & 0.4 \\ 0.2 & 0.1 & 0.2 & 11 \end{bmatrix}.$$

Odavde nalazimo da je matrica strukture procesa

$$A = \begin{bmatrix} 0 & 0.2 & 0.1 & 0.2 \\ 0.25 & 0 & 0.35 & 0.1 \\ 0.2 & 0.25 & 0 & 0.4 \\ 0.2 & 0.1 & 0.2 & 0 \end{bmatrix},$$

dok su vremena potrebna za završetak svakog taska tokom prve iteracije data dijagonalnom matricom $T = \text{diag}\{12, 15, 17, 11\}$. Ovo je kvantitativna verzija uparenih blokova (taskovi C-F) u matrici ravoja softvera sa slike 3.3. Taskovi u ovoj matrici su redom: Projektovanje klasnih dijagrama, Razvoj jednog modula, Testiranje modula i Integracija sistema. Numeričke vrednosti elemenata matrice A uzete su na sledeći način: ako se projektovanje klasnih dijagrama vrši ponovo od početka, tada 25% rada na razvoju modula mora ponovo biti urađeno (ulaz u vrsti 2, kolona 1 je 0.25), i tako dalje.

Matrice sopstvenih vrednosti $L = \text{diag}\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ i sopstvenih vektora $S = (\boldsymbol{v}_i)$ su

$$L = \begin{bmatrix} 0.632001 & & & \\ & -0.385109 & & \\ & & -0.123446 + 0.0400809i & \\ & & & -0.123446 - 0.0400809i \end{bmatrix}$$

i

$$S = \begin{bmatrix} 0.400286 & 0.361247 & 0.626106 & 0.626106 \\ 0.557020 & -0.67131 & 0.0185799 + 0.497679i & 0.0185799 - 0.497679i \\ 0.603925 & 0.568816 & -0.460882 - 0.078238i & -0.460882 + 0.078238i \\ 0.405924 & -0.308695 & -0.174591 - 0.333085i & -0.174591 + 0.333085i \end{bmatrix}.$$

Četiri sopstvena vektora $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ su kolone matrice S. Najveća sopstvena vrednost $\lambda_1 = 0.632$ dominantno utiče na brzinu konvergencije procesne iteracije, tj. ona određuje projektni task koji najsporije konvergira. Njoj odgovarajući sopstveni vektor $\boldsymbol{v}_1$ (prva kolona matrice S) je pozitivan (što i sledi na osnovu Peron-Frobenijusove teoreme). Apsolutne vrednosti njegovih elemenata su veće nego apsolutne vrednosti elemenata drugih sopstvenih vektora.

Na osnovu vrednosti elemenata sopstvenog vektora $\boldsymbol{v}_1$ zaključujemo da su dva srednja taska najuticajnija u smislu da zahtevaju najviše rada. U ovo

se možemo uveriti i izračunavanjem radnih vektora u nekoliko prvih iteracija:

$$\boldsymbol{u}_0 = \begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ \boldsymbol{u}_1 = \begin{bmatrix}0.5\\0.7\\0.85\\0.5\end{bmatrix},\ \boldsymbol{u}_2 = \begin{bmatrix}0.325\\0.4725\\0.475\\0.33\end{bmatrix},\ \boldsymbol{u}_3 = \begin{bmatrix}0.21\\0.2815\\0.319125\\0.20725\end{bmatrix},\ \boldsymbol{u}_4 = \begin{bmatrix}0.129663\\0.184919\\0.195275\\0.133975\end{bmatrix}.$$

Rad izvršen u prvom i poslednjem tasku manji je od dva srednja u svim iterativnim koracima.

Isti zaključak može se izvesti izračunavanjem ukupnog rada koji se izvrši za vreme iterativnog procesa. Vektor ukupnog rada $\boldsymbol{u}$ dat je kolona vektorom

$$\boldsymbol{u} = S(I-L)^{-1}S^{-1}\boldsymbol{u}_0 = \begin{bmatrix}2.38997\\2.95118\\3.17938\\2.40899\end{bmatrix}.$$

Vidimo da je više rada izvršeno pomoću srednja dva taska za vreme iterativnog procesa, kao što je već zaključeno na osnovu sopstvenih vrednosti i sopstvenih vektora.

Konačno, nalazimo vektor ukupnih vremena

$$T\boldsymbol{u} = \begin{bmatrix}28.6796\\44.2677\\54.0495\\26.4989\end{bmatrix}. \blacktriangle$$

3.6 Intervalna linearna algebra

Na početku ovog poglavlja je rečeno da se veći stepen slobode u kvantitativnoj proceni taskova može postići korišćenjem intervala umesto realnih brojeva u odgovarajućem modelu datom preko intervalne matrice strukture procesa $\boldsymbol{A} = (A_{ij})$, gde su elementi A_{ij} realni intervali. Na primer, ako neki task učestvuje u celokupnom razvojnom procesu sa otprilike 10%, onda je rad sa intervalom (0.09,0.11) (koji predstavlja procenu od najmanje 9% do najviše 11%) svrsishodniji nego rad sa fiksnim brojem 0.1 (10%).

Sada ćemo dati kratak pregled intervalnih operacija koje će biti korišćene u daljem radu. Za oznake i više detalja videti [2] i [91]. Mi ćemo se ovde

ograničiti na realne intervale i realne intervalne matrice, mada ćemo uzeti kružne diskove za kompleksne intervale kada budemo razmatrali kompleksne sopstvene vrednosti matrice. Najvažnije definicije intervalnih matrica i osnovne matrične relacije date su u Poglavlju 2.

Slično Definicijama 3.5 i 3.8, uvodimo pojam nerazložive i konvergentne matrice i dajemo odgovarajuće teoreme. Više detalja može se naći u radu [92].

Definicija 3.10. Intervalna matrica $\boldsymbol{A}$ zove se nerazloživa ako je realna matrica $|\boldsymbol{A}|$ nerazloživa. ▲

Definicija 3.11. Za matricu $\boldsymbol{A}$ kaže se da konvergira (ka nuli) ako i samo ako niz $\left\{\boldsymbol{A}^k\right\}$, gde je $\boldsymbol{A}^k = (A_{ij}^{(k)}) = ([\underline{a}_{ij}^{(k)}, \overline{a}_{ij}^{(k)}])$, konvergira ka nula matrici $\boldsymbol{O}$, odnosno, ako je

$$\lim_{k\to\infty} \underline{a}_{ij}^{(k)} = 0, \ \lim_{k\to\infty} \overline{a}_{ij}^{(k)} = 0. \ ▲$$

Teorema 3.6. (Mayer [91]) *Neka je* $\boldsymbol{A} \in M_{nn}(I(\mathbb{R}))$. *Matrica* $\boldsymbol{A}$ *je konvergentna ako je* $\rho(|\boldsymbol{A}|) < 1$. ▲

Teorema 3.7. (Mayer [91]) *Neka je* $\boldsymbol{A} \in M_{nn}(I(\mathbb{R}))$ *nerazloživa i neka je* $d(\boldsymbol{A}) \neq \boldsymbol{O}$. *Tada je matrica* $\boldsymbol{A}$ *konvergentna ako i samo ako je* $\rho(|\boldsymbol{A}|) < 1$. ▲

Pojam konvergencije niza intervalnih matrica i konvergencije geometrijskog matričnog reda uvodi se analogno kao za tačkaste matrice.

Definicija 3.12. Neka su $\boldsymbol{C}^{(k)} := ([\underline{c}_{ij}^{(k)}, \overline{c}_{ij}^{(k)}])$ i $\boldsymbol{C} := ([\underline{c}_{ij}, \overline{c}_{ij}])$ intervalne matrice i neka važi

$$\lim_{k\to\infty} \underline{c}_{ij}^{(k)} = \underline{c}_{ij}, \ \lim_{k\to\infty} \overline{c}_{ij}^{(k)} = \overline{c}_{ij}.$$

Tada kažemo da $\{\boldsymbol{C}^{(k)}\}$ konvergira (ka $\boldsymbol{C}$). ▲

Definicija 3.13. Neka je $\boldsymbol{C}^{(k)} \in M_{nn}(I(\mathbb{R}))$. Za matrični red $\sum_{k=0}^{\infty} \boldsymbol{C}^{(k)}$ kaže se da je konvergentan ako i samo ako niz parcijalnih suma $\boldsymbol{E}_m := \sum_{k=0}^{m} \boldsymbol{C}^{(k)}$ $(m = 0, 1, \ldots)$ konvergira. ▲

Teorema 3.8. (Mayer [92]) *Neka je* $\boldsymbol{A} \in M_{nn}(I(\mathbb{R}))$. *Geometrijski red* $\sum_{k=0}^{\infty} \boldsymbol{A}^k$ *je konvergentan ako* $\lim_{k\to\infty} \boldsymbol{A}^k = \boldsymbol{O}$, *odnosno, ako je* $\rho(|\boldsymbol{A}|) < 1$. ▲

3.7 Intervalne matrice strukture procesa

Isto kao u Odeljku 4.3 za tačkaste matrice, pretpostavimo da je početni radni vektor dat u obliku jediničnog vektora $\boldsymbol{u}_0 = (1, 1, 1, 1)$, i neka je $\boldsymbol{A} = (A_{ij})$ intervalna matrica strukture procesa. Model projektno-razvojnog procesa je takav da je intervalna matrica $|\boldsymbol{A}|$ nerazloživa, što shodno Definiciji 3.10 znači da je tačkasta matrica $|\boldsymbol{A}|$ nerazloživa. Svaki iterativni korak proizvodi promenu u intervalnom vektoru rada na osnovu

$$\boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{u}_{k-1} = \boldsymbol{A}^k\boldsymbol{u}_0.$$

Zbir svih intervalnih vektora rada obrazuje intervalni vektor ukupnog rada

$$\boldsymbol{u} = \sum_{k=0}^{m} \boldsymbol{A}^k\boldsymbol{u}_0 = \left(\sum_{k=0}^{m} \boldsymbol{A}^k\right)\boldsymbol{u}_0. \tag{3.12}$$

Ako je $\boldsymbol{T} = \mathrm{diag}\{[\underline{t}_1, \bar{t}_1], \ldots, [\underline{t}_n, \bar{t}_n]\}$ dijagonalna intervalna matrica vremena izvršavanja taskova, tada je $\boldsymbol{A}\boldsymbol{u}$ intervalni vektor koji daje ukupno vreme neophodno za izvršenje svakog od taskova tokom m iterativnih koraka.

Potrebno je ispitati uslove pod kojima zbir intervalnih matrica $I + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots + \boldsymbol{A}^m$ u (3.12) postaje ograničen i konvergira ka nekoj fiksnoj intervalnoj matrici kada broj iteracija m raste neograničeno. Pre svega, intervalna matrica $\boldsymbol{A}$ mora da bude konvergentna, videti Definiciju 3.11. U tom slučaju, na osnovu Teorema 3.7 i 3.8 sledi da geometrijski red $I + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots +$ konvergira ako i samo ako je $\rho(|\boldsymbol{A}|) < 1$. Tada je

$$\sum_{k=0}^{m} \boldsymbol{A}^k \subset \sum_{k=0}^{\infty} \boldsymbol{A}^k = (\boldsymbol{I} - \boldsymbol{A})^{-1}.$$

Odavde, za intervalni vektor ukupnog rada važi

$$\boldsymbol{u} = \left(\sum_{k=0}^{m} \boldsymbol{A}^k\right)\boldsymbol{u}_0 \subset (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{u}_0. \tag{3.13}$$

Karakterizacija taskova koji učestvuju u projektno-razvojnom modelu preko sopstvenih vrednosti i sopstvenih vektora intervalne matrice $\boldsymbol{A}$ vrši se na isti način kao u slučaju tačkaste MSP, s tim što se sada radi sa sopstvenim vrednostima u obliku intervala i sopstvenim vektorima u obliku intervalnih vektora. Ovo je ilustrovano u primeru 3.2.

Pre razmatranja konkretnog numeričkog primera, izložićemo jedan jednostavan algoritam za nalaženje inverzne intervalne matrice, koji se može pogodno upotrebiti u (3.13). Ovaj algoritam je zasnovan na rezultatima Hansena [58] i Herzbergera i Bethkea [66].

Neka je $\boldsymbol{X}$ kvadratna intervalna matrica čiju inverzuju tražimo, i neka je $\operatorname{mid}(\boldsymbol{X})$ matrica sredina koja ima inverznu matricu $(\operatorname{mid}(\boldsymbol{X}))^{-1}$. Definišimo intervalnu matricu

$$\boldsymbol{E} = \dot{\boldsymbol{I}} - \boldsymbol{X} \cdot (\operatorname{mid}(\boldsymbol{X}))^{-1} = [-1, 1] \cdot |\boldsymbol{E}|. \tag{3.14}$$

Tada, pod uslovom da je $\| \, |\boldsymbol{E}| \, \| < 1$, za svako $\nu \geq 0$ važi inkluzija (Hansen [58])

$$\boldsymbol{X}^{-1} \subseteq (\operatorname{mid}(\boldsymbol{X}))^{-1} \left(\sum_{j=0}^{\nu} \boldsymbol{E}^j + \boldsymbol{R}^{(\nu)} \right) =: \boldsymbol{Y}^{(0)}, \quad \boldsymbol{E}^0 = \dot{\boldsymbol{I}}, \tag{3.15}$$

gde je

$$\boldsymbol{R}^{(\nu)} = ([-r^{(\nu)}, r^{(\nu)}]), \quad r^{(\nu)} = \frac{\| \, |\boldsymbol{E}| \, \|^{\nu+1}}{1 - \| \, |\boldsymbol{E}| \, \|}. \tag{3.16}$$

S obzirom da izračunavanje stepena intervalnih matrica $\boldsymbol{E}$ zahteva dosta numeričkih operacija, Herzberger i Bethke [66] su izveli sledeću inkluziju

$$\boldsymbol{E}^{\nu+1} = [-1, 1]|\boldsymbol{E}|^{\nu+1} \subseteq [-1, 1] \cdot (\| \, |\boldsymbol{E}| \, \|^{\nu+1}) \tag{3.17}$$

koja se može upotrebiti u (3.15).

Za nalaženje inkluzivne intervalne matrice koja sadrži datu inverznu intervalnu matricu postoje brojne varijante iterativnog intervalnog Šulcovog metoda [2]. Jedna od jednostavnijih varijanti definisana je sledećim iterativnim postupkom:

Modifikovan intervalni Šulcov metod: Polazeći od $\boldsymbol{Y}^{(0)}$, gde je $\boldsymbol{Y}^{(0)}$ intervalna matrica određena pomoću Hansenove inkluzivne formule (3.15), izračunavamo niz inkluzivnih intervalnih matrica pomoću iterativne formule

$$\boldsymbol{Y}^{(k+1)} = (\operatorname{mid}(\boldsymbol{X}))^{-1} + \boldsymbol{Y}^{(k)} \cdot \boldsymbol{E}, \quad (k = 0, 1, \ldots) \tag{3.18}$$

koji zadovoljava „lanac" inkluzija (videti [66])

$$\boldsymbol{Y}^{(0)} \supseteq \boldsymbol{Y}^{(1)} \supseteq \cdots \supseteq \boldsymbol{Y}^{(k)} \supseteq \boldsymbol{X}^{-1}.$$

Iterativni metod (3.18) konvergira pod uslovom $\| \, |\boldsymbol{E}| \, \| < 1$.

Primer 3.2. Neka je intervalna matrica strukture modela zadata intervalnom matricom

$$\boldsymbol{A} = \begin{bmatrix} 0 & [0.24, 0.26] & [0.29, 0.31] & [0.09, 0.11] \\ [0.19, 0.21] & 0 & 0 & [0.24, 0.26] \\ 0 & [0.19, 0.21] & 0 & [0.29, 0.31] \\ [0.29, 0.31] & [0.09, 0.11] & [0.19, 0.21] & 0 \end{bmatrix}$$

i neka je

$$\boldsymbol{T} = \text{diag}\{[22, 24], [12, 13], [15, 17], [19, 20]\}$$

intervalna dijagonalna matrica vremena izvršavanja pojedinih taskova. Na primer, interval $A_{12} = [0.24, 0.26]$ znači da ako projektovanje klasnih dijagrama (C) treba da se realiziju u potpunosti, tada na razvoju jednog modula (D) mora biti urađeno 24 do 26% rada.

Koristeći programski paket INTLAB(videti [119] i Poglavlje 2) i operacija, osobina intervalne aritmetike kao i softver razvijen u Poglavlju 2, našli smo da sopstvene vrednosti pripadaju intervalima $\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4$, dok su odgovarajući intervalni sopstveni vektori dati vektorima $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_2, \boldsymbol{v}_4$. Ovi intervali i intervalni vektori prikazani su u tabeli 3.1.

Iz tabele 3.1 vidimo da je dominantna sopstvena vrednost sadržana u intervalu

$$\Lambda_1 = [0.52301924009798, 0.58114053016347].$$

Ovaj interval ima dominantan uticaj na brzinu konvergencije zbira intervalnih matrica $\boldsymbol{I}+\boldsymbol{A}+\boldsymbol{A}^2+\boldsymbol{A}^3+\cdots$ definišući na taj način najsporiji mod. To znači da je većina posla u iterativnom procesu opisana ovim primarnim razvojnim modom. S obzirom da je $|\Lambda_1| = 0.581... < 1$, geometrijski red $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \boldsymbol{A}^3 + \cdots$ je konvergentan (na osnovu Teoreme 3.8).

Intervalnoj sopstvenoj vrednosti Λ_1 odgovara pozitivan intervalni sopstveni vektor $\boldsymbol{v}_1$. Na osnovu vrednosti $\boldsymbol{v}_1$ zaključujemo da prvi i četvrti task dominiraju, tj. oni zahtevaju više rada nego drugi i treći task. Ovo potvrđuju

i vrednosti radnih intervalnih vektora izračunatih za prvih pet iteracija,

$$\boldsymbol{u}_1 = \begin{bmatrix} [0.62, 0.68] \\ [0.43, 0.47] \\ [0.48, 0.52] \\ [0.57, 0.63] \end{bmatrix}, \quad \boldsymbol{u}_2 = \begin{bmatrix} [0.294, 0.353] \\ [0.255, 0.307] \\ [0.247, 0.295] \\ [0.310, 0.372] \end{bmatrix}, \quad \boldsymbol{u}_3 = \begin{bmatrix} [0.1606, 0.2117] \\ [0.1301, 0.1707] \\ [0.1382, 0.1796] \\ [0.1550, 0.2048] \end{bmatrix},$$

$$\boldsymbol{u}_4 = \begin{bmatrix} [0.0853, 0.1226] \\ [0.0677, 0.0977] \\ [0.0697, 0.0993] \\ [0.0845, 0.1221] \end{bmatrix}, \quad \boldsymbol{u}_5 = \begin{bmatrix} [0.0441, 0.0696] \\ [0.0364, 0.0575] \\ [0.0374, 0.0584] \\ [0.0440, 0.0696] \end{bmatrix}.$$

$\Lambda_1 = [0.52301924009798, 0.58114053016347]$
$\boldsymbol{v}_1$
$[0.54738574910929, 0.54738574910930]$
$[0.40530246418811, 0.48320004678584]$
$[0.41236410795740, 0.49979479009649]$
$[0.49655775290886, 0.58972250530368]$

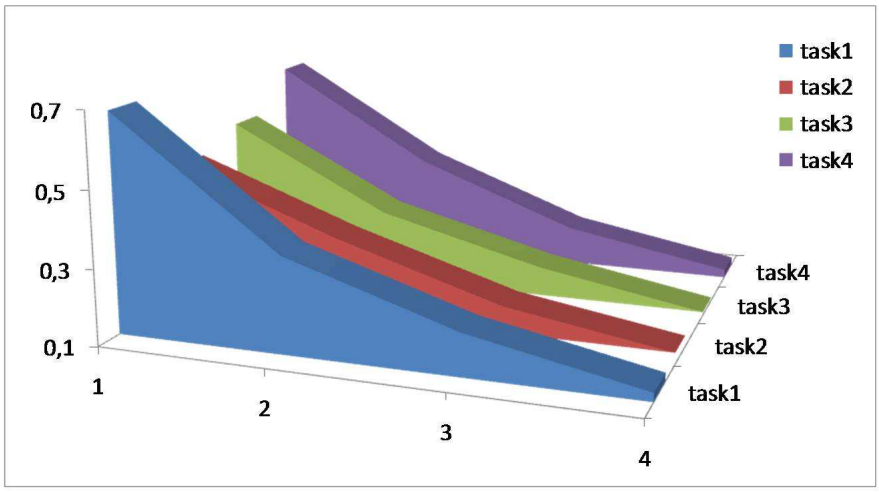
$\Lambda_2 = \{-0.24101922047039 + 0.21895037422036\,i, 0.03673486072572\}$
$\boldsymbol{v}_2$
$[-0.54357862004773, -0.54357862004771]$
$\{0.47325704096327 - 0.05569722569723\,i, 0.12496058713142\}$
$\{0.03319997099298 - 0.50636666369940\,i, 0.09783032840514\}$
$\{0.02738643729559 + 0.46817563256495\,i, 0.10342721556151\}$

$\Lambda_3 = \{-0.24101922047039 - 0.21895037422036\,i, 0.03673486072572\}$
$\boldsymbol{v}_3$
$[-0.54357862004773, -0.54357862004771]$
$\{0.47325704096327 + 0.05569722569723\,i, 0.12496058713142\}$
$\{0.03319997099298 + 0.50636666369940\,i, 0.09783032840514\}$
$\{0.02738643729559 - 0.46817563256495\,i, 0.10342721556151\}$

$\Lambda_4 = [-0.10170498388582, -0.03837790449406]$
$\boldsymbol{v}_4$
$[-0.31566698617679, -0.08070970080049]$
$[-0.73458918895239, -0.73458918895238]$
$[0.45026763870841, 0.62368537786722]$
$[0.30210950227359, 0.42660534875258]$

Tabela 3.1 Intervalne sopstvene vrednosti i intervalni sopstveni vektori

Rad izvršen u drugom i trećem tasku manji je od prvog i četvrtog u svim iterativnim koracima. Ovo je prikazano pomoću dijagrama na slici 3.9.



Slika 3.9 Intervalni radni vektori u prve četiri iteracije

Da bismo ocenili vektor ukupnog rada, primenićemo formulu (3.13), pri čemu ćemo inverziju intervalne matrice $\boldsymbol{I} - \boldsymbol{A}$ odrediti postupkom (3.15)–(3.18).

Neka je $\boldsymbol{X} = \boldsymbol{I} - \boldsymbol{A}$. Tada izračunavamo

$$\boldsymbol{X} = \boldsymbol{I} - \boldsymbol{A} = \begin{bmatrix} [1,1] & [-0.26,-0.24] & [-0.31,-0.29] & [-0.11,-0.09] \\ [-0.21,-0.19] & [1,1] & [0,0] & [-0.26,-0.24] \\ [0,0] & [-0.21,-0.19] & [1,1] & [-0.31,-0.29] \\ [-0.31,-0.29] & [-0.11,-0.09] & [-0.21,-0.19] & [1,1] \end{bmatrix}.$$

Na osnovu (3.14) izračunavamo intervalnu matricu $\boldsymbol{E}$ i nalazimo odgovarajuću realnu matricu

$$|\boldsymbol{E}| = \begin{bmatrix} 0.00975 & 0.01790 & 0.01704 & 0.02056 \\ 0.016200 & 0.00723 & 0.00803 & 0.01584 \\ 0.00776 & 0.01466 & 0.00555 & 0.01611 \\ 0.01733 & 0.01902 & 0.01755 & 0.01175 \end{bmatrix}.$$

Kako je $\| \, |\boldsymbol{E}| \, \| = 0.06565 < 1$, na osnovu (3.15) važi inkluzija

$$\boldsymbol{X}^{-1} = (\boldsymbol{I} - \boldsymbol{A})^{-1} \subseteq (\operatorname{mid}(\boldsymbol{X}))^{-1} \left(\sum_{j=0}^{\nu} \boldsymbol{E}^j + \boldsymbol{R}^{(\nu)} \right) =: \boldsymbol{Y}^{(0)}, \quad \boldsymbol{E}^0 = \boldsymbol{I},$$

za svako $\nu \geq 0$. Osim toga, obezbeđena je konvergencija intervalnog Šulcovog metoda (3.18). Za $\nu = 2$ izračunali smo

$$r_2 = \frac{\| \, |\boldsymbol{E}| \, \|^3}{1 - \| \, |\boldsymbol{E}| \, \|} = 0.0003$$

i formirali odgovarajuću intervalnu matricu $\boldsymbol{R}^{(2)}$ saglasno formuli (3.16). Na osnovu (3.15) i (3.17) dobija se

$$\begin{aligned} \boldsymbol{X}^{-1} &\subseteq (\operatorname{mid}(\boldsymbol{X}))^{-1} \cdot (\boldsymbol{I} + \boldsymbol{E} + \boldsymbol{E}^2 + \boldsymbol{E}^{(2)}) \\ &\subseteq (\operatorname{mid}(\boldsymbol{X}))^{-1} \cdot (\boldsymbol{I} + [-1, 1] \cdot ((\| \, |\boldsymbol{E}| \, \|) + (\| \, |\boldsymbol{E}| \, \|)^2 + \boldsymbol{R}^{(2)}) =: \boldsymbol{Y}_0, \end{aligned}$$

pri čemu je za konkretne vrednosti intervalna matrica $\boldsymbol{Y}^{(0)}$ data sa

$$\boldsymbol{Y}^{(0)} = \begin{bmatrix} [1.042, 1.356] & [0.270, 0.586] & [0.279, 0.595] & [0.204, 0.519] \\ [0.218, 0.491] & [1.032, 1.305] & [0.052, 0.324] & [0.251, 0.524] \\ [0.068, 0.347] & [0.193, 0.473] & [1.018, 1.298] & [0.314, 0.594] \\ [0.286, 0.596] & [0.161, 0.471] & [0.231, 0.541] & [1.087, 1.397] \end{bmatrix}.$$

Zatim je primenjen intervalni Šulcov iterativni postupak (3.18). U ovom iterativnom postupku kao kriterijum zaustavljanja primenili smo uslov

$$\|d(\boldsymbol{Y}^{(k)} - \boldsymbol{Y}^{(k-1)})\|_2 < \varepsilon = 10^{-5}, \tag{3.19}$$

gde $d(\boldsymbol{Z})$ označava realnu matricu širina pridruženu intervalnoj matrici $\boldsymbol{Z}$ (videti Odeljak 2.3), dok je norma definisana kao u Odeljku 3.2. Uslov (3.19) praktično znači da se iterativni proces prekida kada razlika između elemenata-intervala dve inkluzivne matrice u uzastopnim iterativnim koracima postane dovoljno mala. U konkretnom slučaju ($\varepsilon = 10^{-5}$), granice intervala koji predstavljaju elemente iterativne matrice $\boldsymbol{Y}^{(k)}$ se posle četvrte iteracije menjaju tek na petoj ili šestoj decimali, tako da je razumno prekinuti proces – veća tačnost za procesne iteracije nije potrebna. Norma $\|d(\boldsymbol{Y}^{(k)} - \boldsymbol{Y}^{(k-1)})\|_2$ data je u tabeli 3.2.

k	1	2	3	4	5
$\|d(\boldsymbol{Y}^{(k)} - \boldsymbol{Y}^{(k-1)})\|_2$	0.323	0.052	2.9×10^{-3}	1.62×10^{-4}	9.09×10^{-6}

Tabela 3.2

Za inkluzivnu matricu intervalne matrice $(\boldsymbol{I} - \boldsymbol{A})^{-1}$ $(= \boldsymbol{X}^{-1})$ usvojili smo matricu $\boldsymbol{Y}^{(5)}$ koja zadovoljava uslov (3.19), a dobijena je iterativnim postupkom (3.18),

$$(\boldsymbol{I} - \boldsymbol{A})^{-1} \subset \boldsymbol{Y}^{(5)}$$
$$= \begin{bmatrix} [1.1593, 1.2187] & [0.3784, 0.4572] & [0.3931, 0.4609] & [0.3071, 0.3958] \\ [0.3139, 0.3772] & [1.1339, 1.1863] & [0.1550, 0.2037] & [0.3437, 0.4131] \\ [0.1730, 0.2237] & [0.2909, 0.3567] & [1.1269, 1.1702] & [0.4106, 0.4801] \\ [0.3956, 0.4663] & [0.2651, 0.3471] & [0.3405, 0.4110] & [1.1957, 1.2689] \end{bmatrix}.$$

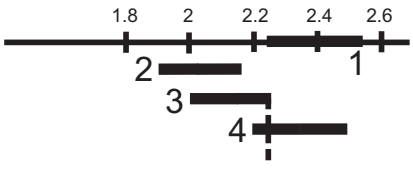
Na osnovu ovog procenili smo intervalni vektor ukupnog rada

$$\boldsymbol{u} \subset (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{u}_0 = \begin{bmatrix} [2.2377, 2.5327] \\ [1.9465, 2.1802] \\ [2.0016, 2.2308] \\ [2.1968, 2.4934] \end{bmatrix}.$$

Vektor ukupnog vremena $\boldsymbol{Tu}$ sadržan je u intervalu

$$\boldsymbol{Tu} \subset \boldsymbol{T}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{u}_0 = \begin{bmatrix} [49.229, 60.785] \\ [23.358, 28.343] \\ [30.024, 37.924] \\ [41.739, 49.868] \end{bmatrix}.$$

Posmatrajući elemente vektora $\boldsymbol{u}$ može se videti da prvi i četvrti task zahtevaju više rada, kao što je i bilo zaključeno na osnovu uvida u vrednosti pojedinačnih vektora rada $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ i intervalnog sopstvenog vektora $\boldsymbol{v}_1$. Primetimo da se intervali koji definišu treći i četvrti task seku jednim delom (slika 3.10), što je rezultat rada sa intervalima umesto sa realnim brojevima, ali i samog modela gde se treći task po svom uticaju približava četvrtom tasku. ▲



Slika 3.10 Intervalni elementi vektora ukupnog rada

4

Računska efikasnost numeričkih algoritama

Ovo poglavlje bavi se računskom efikasnošću numeričkih algoritama, pre svega algoritmima za rešavanje nelinearnih i algebarskih jednačina. Kao sastavni deo analize efikasnosti u slučaju primene aritmetike višestruke preciznosti, u prvom delu poglavlja su razmatrani iterativni procesi za realizaciju osnovnih operacija i računske cene aritmetičkih operacija kao funkcije preciznosti primenjene aritmetike izražene preko broja bitova koji definišu ovu preciznost. Originalni doprinos ovoj oblasti predstavlja analiza efikasnosti iterativnih metoda za rešavanje nelinearnih jednačina kada se aritmetika višestruke preciznosti koristi sa promenljivim brojem značajnih cifara u iterativnim koracima. Za opisani mod realizacije iterativnih metoda, u ovom poglavlju se predlaže nova formula za ocenu računske efikasnosti iterativnih postupaka preko indeksa efikasnosti. Nova formula koristi težine proporcionalne broju bitova u svakoj iteraciji koji definiše preciznost upotrebljene aritmetike. Ova formula iskorišćena je za upoređenje i rangiranje postojećih iterativnih metoda za istovremeno nalaženje svih nula polinoma u režimu dinamičke preciznosti.

4.1 Analiza efikasnosti algoritama

Analiza efikasnosti algoritama predstavlja centralnu oblast u računarskim naukama jer je izbor algoritma za rešavanje određenog problema često od

presudnog značaja. Tekst-procesori, internet pretračivači, ATMovi[1], video igre i sistemi održavanja života veoma zavise od efikasnih algoritama. Da bismo odabrali najefikasnije algoritme za rešavanje datog problema, potrebno je oceniti njihove prednosti i mane i na osnovu dobijenih parametara izvršiti upoređivanje algoritama i njihovo rangiranje. *Teorija efikasnosti numeričkih algoritama* je oblast računarskih nauka koja se bavi uvođenjem i razvojem postupaka za upoređivanje efikasnosti različitih metoda rešavanja datog problema. Primetimo da ovde koristimo pojam *metodi rešavanja* a ne *programi*. Važno je naglasiti da se analiza bavi prevashodno uočavanjem značajnih poboljšanja u efikasnosti algoritama - poboljšanja koje se najčešće mogu dobiti korišćenjem usavršenih metoda rešavanja, a ređe preko vispremih trikova u programiranju.

Teorija efikasnosti numeričkih algoritama kao naučna disciplina pripada ne samo računarskim naukama, već i primenjenoj matematici. Primetimo da su najveći doprinos ovoj oblasti dali naučnici koji su se bavili (ili se još uvek bave) računarskim naukama, kao što su Knut, Traub, Brent i Kung. Interesantno je da su sva četvorica stekla svoje doktorate iz matematike. Ovo je jedan od najočiglednijih primera vrlo bliskih veza između primenjene matematike i računarskih nauka.

Pri konstrukciji algoritama značajno je kako efikasno korišćenje vremena izvršenja algoritma tako i memorijskog prostora. Međutim, ekspanzija memorijskih kapaciteta računara znatno je redukovala značaj prostorne (memorijske) efikasnosti. Stoga ćemo se usredsrediti na vremensku efikasnost. Razmotrićemo problem upoređivanja vremenske efikasnosti dva algoritma koji rešavaju isti problem. Jedan moguć pristup upoređivanja je da se ta dva algoritma implementiraju u C++ (na primer) a zatim izmere vremena njihovog izvršenja. Međutim, postoje tri osnovne teškoće u ovom pristupu:

1. Kako su algoritmi kodirani? Da li je jedan algoritam brži od drugog zato što je bolje isprogramiran? Ne bi trebalo da upoređujemo implementacije već same algoritme. Implementacije su osetljive na faktore kao što je stil programiranja, što zamagljuje problem.

2. Koji računar treba koristiti? Jedino bi bilo ispravno koristiti isti računar za oba programa. Međutim, čak i tada pojedine operacije koje

[1]ATM (Asynchronous Transfer Mode) je komutaciona tehnika bazirana na ćelijama koja koristi asinhroni vremenski multipleks.

jedan algoritam koristi mogu biti brže ili sporije nego druge - a to može biti upravo suprotno na nekom drugom računaru.

3. Koje baze podataka treba programi da koriste? Uvek postoji opasnost da ćemo izabrati neke delove problema koje jedan od algoritama izvršava netipično brzo. Na primer, uporedimo sekvencijalnu pretragu i binarnu pretragu uređenog niza. Ukoliko je potrebno da tražimo najmanji element u tom nizu, tada će sekvencijalna pretraga pronaći traženi član mnogo brže od binarne pretrage.

Da bi se ove teškoće prevazišle, eksperti u oblasti računarskih nauka koriste matematičke metode da bi sproveli analizu algoritma nezavisno od specifične implementacije, korišćenog računara ili baze podataka. Analize algoritama često su zasnovane na prebrojavanju značajnih operacija u određenom rešenju, uzimajući u obzir vreme izvršenja ovih operacija.

Kao primer, posmatrajmo izračunavanje vremena neophodnog za izvršenje dela programa koji se sastoji od umetnutih petlji:

```
for (i=1; i<=n; ++i)
  for (j=1; j<=i; ++j)
    for (k=0; k<5; ++k)
      Task T;
```

Ako task T zahteva t jedinica vremena, unutrašnja petlja po k zahteva $5t$ jedinica vremena. Uzimajući u obzir i- i j-petlju, ukupno vreme iznosi $5t \cdot n(n+1)/2$ jedinica vremena. Ovaj primer prikazuje vremenske zahteve nekog algoritma kao funkciju dimenzije problema n.

Način merenja dimenzije problema zavisi od aplikacije. Vreme za izvršenje algoritma za pretraživanje zavisi od veličine niza koji pretražujemo. Najznačajnija stvar koju treba uočiti je brzina kojom se uvećava vreme izvršenja algoritma u funkciji dimenzije problema. Zaključak u smislu: *„Algoritam A zahteva vreme proporcionalno $f(n)$"*, omogućava da se algoritam A uporedi s drugim algoritmom B koji zahteva $g(n)$ vremenskih jedinica. Za algoritam A se kaže da je reda $f(n)$, što označavamo kao $O(f(n))$; $f(n)$ zove se funkcija stope rasta datog algoritma. S obzirom da ova oznaka koristi veliko slovo O, zvaćemo je *veliko-O*. Napomenimo da se *veliko-O*, poznato kao Landauov simbol, koristi za upoređivanje absolutnih vrednosti realnih veličina. Na primer, $x = O(1)$ znači da je $|x|$ reda konstante. Gausov algoritam eliminacije za rešavanje sistema od n linearnih jednačina zahteva $(4n^3+9n^2-7n)/6$

operacija sabiranja, oduzimanja i množenja, što se za veliko n obično izražava kao $O(n^3)$. Godine 1969. V. Štrasen [134] je razvio algoritam za rešavanje sistema lineranih jednačina koji koristi množenje i inverziju matrice smanjujući broj operacija na $n^{\log_2 7}$ (videti (4.1)). Ako je $M(n)$ broj izračunavanja pri množenju matrica dimenzija $n \times n$ tada važi $M(n) = O(n^3)$, dakle, taj broj je srazmeran kubu dimenzije matrice. itd.

Ako problem čija je dimenzija n zahteva vreme koje je direktno proporcionalno n, kažemo da je problem reda $O(n)$ ili, kraće, reda n. Ako je vreme izvršenja pri rešavanju problema direktno proporcionalno n^2, problem je reda $O(n^2)$, i tako dalje. Preciznija definicija reda algoritma može se iskazati na sledeći način:

Algoritam A je reda $f(n)$, u oznaci $O(f(n))$, ako postoji konstanta c i broj n_0 tako da A ne zahteva više od $cf(n)$ jedinica vremena da reši problem dimenzije $n \geq n_0$.

Složeni algoritam A_n može kombinovati više pojedinačnih jednostavnijih algoritama B_k reda n^k. Na taj način, vreme izvršenja ovog složenog algoritma može se izraziti kao linearna kombinacija (vremenske) efikasnosti pojedinih algoritama, tj. u obliku polinoma po n :

$$t(A_n) = a_r n^r + a_{r-1} n^{r-1} + \cdots + a_1 n + a_0.$$

U ovom slučaju se kaže da je algoritam *polinomijalnog tipa*. Druga vrsta algoritama su algoritmi *eksponencijalnog tipa* čije je vreme izvršavanja proporcionalno eksponencijalnoj funkciji k^n. Lako se pokazuje da za dovoljno veliko n, $k > 1$ i fiksno r važi

$$a_r n^r + a_{r-1} n^{r-1} + \cdots + a_1 n + a_0 < k^n.$$

U praksi treba izbegavati algoritme eksponencijalnog tipa; u najvećem broju slučajeva čak i vrlo moćni računari ne mogu da realizuju ove algoritme za razumno vreme. Primer algoritma sa ovom cenom je izračunavanje vrednosti determinante reda $n \times n$ po definiciji, što podrazumeva razvijanje determinante po elementima jedne vrste ili kolone i izračunavanje subdeterminanti. Broj operacija (sabiranje + množenje) je $S(n) + M(n) \approx n \cdot n!$. Korišćenjem Stirlingove formule, za veliko n dobija se procena

$$n! \approx n^n \sqrt{2\pi n}\, e^{-n}$$

tako da je

$$S(n) + M(n) \approx \sqrt{2\pi} n^{n+3/2} e^{-n} \approx \sqrt{2\pi} e^{3/2} k^{n+3/2}, \quad (k = n/e).$$

Uzimajući da računar može da izvrši bilion (10^{12}) operacija u sekundi, računanje determinante 30-reda zahteva približno $1.5 \cdot 10^{14}$ godina. Ovo vreme je duže nego pretpostavljena starost Svemira. Eksponencijalno vreme javlja se i kod čuvenog Hamiltonovog problema (1856.), poznatog i pod imenom problem trgovačkog putnika: *Da li postoji put u ravanskom grafu koji prolazi kroz sve čvorove grafa jednom i samo jednom?* „Metod iscrpljivanja" koji se sasoji od ispitivanja svih mogućih puteva u grafu sa n spojnica je reda 2^n. Još uvek nije pronađen algoritam za rešavanje ovog problema.

Analiza algoritama treba da uzme u obzir bitna svojstva samog algoritma, njegovo granično ponašanje, preciznost i druge osobine, od kojih su neke navedene u nastavku.

- **Analiza najgoreg slučaja.** Neki algoritam može zahtevati različita vremena za rešavanje različitih problema iste dimenzije. Na primer, vreme potrebno nekom algoritmu da pretraži n fajlova zavisi od prirode i dužine fajlova. Obično se posmatra maksimalna količina vremena koja je potrebna nekom algoritmu za rešavanje problema reda n, što znači da uzimamo najgori mogući slučaj. Interesantno je napomenuti da se taktike najgoreg slučaja često koriste u analizama vojnih strategija. Konstrukcije građevinskih objekata takođe razmatraju najgori slučaj.[2]

- **Preciznost**. Poželjno je dobiti što precizniju, odnosno, najmanju moguću gornju granicu za $O(f(n))$.

- **Jednostavnost**. Generalno ćemo smatrati da je $f(n)$ „jednostavno" ako se sastoji samo od jednog člana a koeficijent uz njega je jednak 1. Tako se za pomenuti primer ukupnog broja izračunavanja kod

[2]Jedan od naučnika koji se bave analizom algoritama, potrebu za analizom najgoreg slučaja duhovito je obrazložio da to treba raditi zbog Marfijevog zakona: *„Ako postoje dva ili više mogućih ishoda nekog događaja, i ako jedan od njih dovodi do katastrofe, onda će se taj ishod i desiti."* Edward A. Murphy je bio jedan od inženjera koji su učestvovali u eksperimentu Američkih vojnih snaga sa raketnim sankama pedesetih godina dvadesetog veka. 16 senzora trebalo je pričvrstiti na telo ispitanika i svaki senzor se mogao prilepiti na dva načina. Desilo se da je svih 16 senzora bilo pričvršćeno pogrešno (verovatnoća za to je $1/2^{16} \approx 1.5 \times 10^{-5}$). Posle ovog eksperimenta Murphy je izrekao svoju čuvenu rečenicu.

primene Gausovog metoda eliminacije pri rešavanju sistema od n linearnih jednačina, za dovoljno veliko n uzima da je taj broj $O(n^3)$ umesto $(4n^3 + 9n^2 - 7n)/6$.

4.2 Iterativna kompleksnost algoritama

Računska kompleksnost[3] u smislu efikasnosti primene nekog algoritma na računaru je mera računske cene neophodne da bi se taj algoritam uspešno realizovao. Računska cena može se izraziti na različite načine i u različitim jedinicama, na primer, kao procesorsko vreme (pri raznim izračunavanjima ili rešavanju jednačina), broj upoređivanja (algoritmi sortiranja), veličina memorije (kod linearnih sistema sa velikim brojem jednačina) ili kao broj aritmetičkih operacija (kod matričnog množenja). Napomenimo da se računskoj kompleksnosti algoritama, kao jednoj od bazičnih oblasti računarskih nauka, i danas poklanja velika pažnja i stalno razvijaju novi metodi za analizu efikasnosti algoritama. U ovim istraživanjima ravnopravno učestvuju eksperti koji se bave računarskim naukama i primenjenom matematikom. Navedimo dva renomirana časopisa koje se bave teorijom računske kompleksnosti, *Journal of Complexity* (Elsevier) i *Journal of Complexity* (Birkhäuser-Springer).

U računarskim naukama se računska kompleksnost algoritama definiše na više načina u zavisnosti od usvojenog modela i korišćenih parametara. Mi ćemo razmatrati samo one tipove računske kompleksnosti kod kojih je broj aritmetičkih operacija glavni parametar i samo one algoritme koji rešavaju probleme sa konačnom računskom cenom (tzv. *algebarska računska kompleksnost*). Tipičan primer su problemi koji zavise od izvesne dimenzije n. Na primer, ako je P_n problem množenja dve $n \times n$ matrice i ako se cena svake aritmetičke operacije normalizuje na 1, tada je računska kompleksnost $komp(P_n)$ u granicama datim nejednakostima

$$O(n^2) \leq komp(P_n) \leq O(n^{\log_2 7}). \tag{4.1}$$

Tačna rešenja većine problema u inženjerskim disciplinama i primenjenoj nauci ne mogu se dobiti sa uloženom konačnom cenom, čak i u slučaju korišćenja aritmetike beskonačne preciznosti. Čak i kada se neki problemi mogu rešeti tzv. direktnim metodima (npr. sistem linearnih jednačina

[3]Pojam *računska efikasnost* po prvi put se pojavio u radu Harmanisa i Stearnsa [60] godine 1964.

Kramerovim pravilom pomoću determinanti, ili kubna jednačina), često je efikasnije koristiti iterativni pristup.

Algoritmi koji se javljaju u primenjenoj matematici, računarskim naukama i inženjerskim disciplinama su često iterativne prirode i u tom slučaju odgovarajuća analiza efikasnosti pripada oblasti *iterativne računske kompleksnosti* o kojoj će uglavnom biti reči u ovom poglavlju.

Uobičajeno je da se algebarska računska kompleksnost vezuje za algebarske (ili kombinatorne) procese, a analitička računska kompleksnost za analitičke (ili neprekidne) procese. Prvi značajniji rezultati u Teoriji algebarske računske kompleksnosti odnosili su se na množenje brojeva ([31], Schönhage i Strassen [123]), množenje matrica (Winograd [151], Strassen [134], Hopcroft i Kerr [69]), izračunavanju polinoma (Winograd [151]), planarnosti grafova (Hopcroft i Tarjan [70]). Prvi rezultati u Teoriji analitičke računske kompleksnosti pojavili su se šezdesetih godina dvadesetog veka u radovima Trauba [139], [140], [141], [142], Brenta [12], Cohena [29], Feldsteina i Firestouna [44], Jarratta [74], [75] i Kinga [78].

Veliki broj raznovrsnih algoritama analizira se primenom iterativne računske kompleksnosti. U ovom poglavlju naše razmatranje biće usmereno jedino ka iterativnim procesima za rešavanje nelinearnih jednačina primenom iterativnih postupaka. Iako predstavlja mali deo Numeričke analize, ova oblast je dovoljno široka i značajna da joj je posvećena velika pažnja tokom poslednjih pedeset godina (videti Collatz [30]). Uprkos intenzivnom izučavanju ostalo je još dosta nedovoljno izučenih problema, od kojih su neki nastali kao posledica vrlo brzog razvoja računara i korišćenja aritmetike višestruke preciznosti (skraćeno AVP).

Brzina konvergencije iterativnih metoda

Posmatrajmo neki algoritam za rešavanje izvesnog problema P koji ima rešenje α. To može da bude rezultat nekog izračunavanja ili rešenje jednačine (nelinearne jednačine oblika $f(x) = 0$, uključujući i sistem nelineranih jednačina, sistem linearnih algebarskih jednačina, diferencijalne ili integralne jednačine). Posmatrajmo algoritme koji su po svojoj strukturi iterativne prirode, tj., polazeći od početnog rešenja x_0 u svakom narednom koraku nalazimo aproksimaciju x_k $(k = 1, 2, \ldots)$ ovog rešenja koristeći iterativnu formulu (koja ne mora isključivo da se odnosi na rešavanje nelinearnih jednačina,

već se mogu posmatrati i druge klase problema iterativne prirode)

$$x_{k+1} = \psi(x_k) \quad (k = 0, 1, \ldots), \tag{4.2}$$

sve do tražene tačnosti τ definisane pomoću $|x_k - \alpha| < \tau$. Na primer, ako tražimo jednostruku preciznost, biramo $\tau = 10^{-8}$, ako želimo dvostruku preciznost onda je $\tau = 10^{-16}$. Za najveći broj praktičnih problema dvostruka preciznost je sasvim zadovoljavajuća. Ako je izabrana dovoljno dobra početna aproksimacija i dobar algoritam, tada se tačnost dobijenih aproksimacija $x_1, x_2, x_3, \ldots$ povećava iz koraka u korak smanjujući grešku aproksimacije $\varepsilon_k = |x_k - \alpha|$ sve do ispunjenja tražene tačnosti $\varepsilon_k < \tau$ kada se iterativni proces prekida. U ovom slučaju kažemo da algoritam *konvergira* ka traženom rešenju. Brzinu konvergencije q određujemo na sledeći način.

Ako postoje realan broj q i konstanta C različita od 0, takvi da

$$\frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^q} = \frac{|\phi(x_k) - \alpha|}{|x_k - \alpha|^q} \to C \quad \text{kada } k \to \infty, \tag{4.3}$$

tada se q zove ***red konvergencije*** *iterativnog niza $\{x_k\}$ definisanog iterativnom funkcijom* (4.2).

Ova formula ima teorijski karakter iz dva razloga:

1) u praksi se primenjuje samo konačan broj iterativnih koraka;

2) rešenje (rezultat izračunavanja) α u najvećem broju slučajeva nije poznato.

S obzirom na ograničenje 1), u praksi se često radi sa *računskim redom konvergencije*. Pretpostavimo da nam je poznato rešenje jednačine α i neka su x_{k-1}, x_k i x_{k+1} tri uzastopne aproksimacije ovog rešenja dobijene u iterativnom procesu reda q. Ako logaritmujemo približne relacije

$$\frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^q} \approx C, \quad \frac{|x_k - \alpha|}{|x_{k-1} - \alpha|^q} \approx C$$

i eliminišemo $\log C$, dobijamo računski red konvergencije

$$q_\alpha \approx \frac{\log(|x_{k+1} - \alpha|/|x_k - \alpha|)}{\log(|x_k - \alpha|/|x_{k-1} - \alpha|)}. \tag{4.4}$$

Ova približna vrednost uglavnom dobro aproksimira teorijski red konvergencije q, pod uslovom da nema „patološkog" ponašanja iterativnog metoda

(spora konvergencija u početku iterativnog procesa, „oscilovanje" aproksimacija i slično).

Umesto formule (4.4), korišćenjem aproksimacije $f(x) \approx (x_k - \alpha)g(\alpha)$ dolazimo do formule za računski red konvergencije od praktičnog značaja

$$q_f \approx \frac{\log(|f(x_{k+1})|/|f(x_k)|)}{\log(|f(x_k)|/|f(x_k)|)}. \tag{4.5}$$

Upotrebu formula (4.4) i (4.5) demonstriraćemo na sledećem primeru.

Primer 4.1. Za približno izračunavanje nule funkcije $f(x)$ često se koristi Njutnov iterativni method

$$x_{k+1} = \frac{f(x_k)}{f'(x_k)} \quad (k = 0, 1, \ldots) \tag{4.6}$$

startujući sa približnom početnom vrednoču x_0. Odredimo nulu funkcije

$$f(x) = e^{x^2+3x+2}(x^5 + x + 1) + 1$$

Njutnovim metodom (4.6) u intervalu $[-2, 0]$ polazeći od $x_0 = -1.5$. Korišćenjem aritmetike dvostruke preciznosti (oko 16 značajnih decimalnih cifara), pet iteracija daje sledeće rezultate:

k	x_k	$\lvert x_k - \alpha\rvert$	$\lvert f(x_k)\rvert$
1	-1.241198115598584	0.241198115598584	1.6539861
2	-1.064133327641759	0.064133327641759	0.3454287
3	$-1.\underline{00}4646549752278$	0.004646549752278	0.0233522
4	$-1.\underline{000}023819032346$	0.000023819032346	0.0001191
5	$-1.\underline{000000000}624091$	0.000000000624091	3.12×10^{-9}

Nula funkcije f, koja je testirana u intervalu $[-2, 0]$, je $\alpha = -1$. Korišćenjem formule (4.4) dobija se računski red konvergencije $q_\alpha = 2.00055$, dok formula (4.5), koja je od većeg praktičnog značaja jer ne zahteva poznavanje vrednosti tražene nule, je $q_f = 1.99862$. Napomenimo da su formule (4.4) i (4.5) bile primenjene za poslednje tri iteracije. S obzirom da Njutnov metod ima red konvergencije 2, dobijene vrednosti za q_α i q_f su vrlo dobre procene reda konvergencije. ▲

4.3 Računska cena osnovnih aritmetičkih operacija

Neke od osnovnih operacija u računarskim aritmetikama novijeg datuma često se realizuju koristeći brze iterativne procese za rešavanje nelinearnih jednačina. Analize i rezultati koji slede izvedeni su za binarnu osnovu ($\beta = 2$), ali najveći broj rezultata važi za proizvoljnu bazu. Ako se radi u aritmetici preciznosti od b-bitova i početnim vrednostima sa modulom bliskim 1, tada je cilj dobiti njihove aproksimacije sa tačnošću reda 2^{-b}.

Kao što je već ranije rečeno, Njutnov iterativni metod (4.6) je vrlo koristan metod pri realizaciji aritmetike proizvoljne preciznosti. Njutnov metod konvergira kvadratno, što praktično znači da se broj tačnih cifara pronađene aproksamicije duplira u odnosu na prethodnu aproksimaciju. Većina modernih procesora implementira hardverski samo operacije sabiranja i množenja. Deljenje i korenovanje (kvadratni koren) su mikrokodirani koristeći ili Njutnov metod ako postoji FMA[4] instrukcija ili SRT algoritam.[5]

U nastavku ćemo prikazati realizaciju nekih osnovnih aritmetičkih operacija u aritmetici proizvoljne preciznosti primenom Njutnovog metoda.

Izračunavanje inverznog korena

Izračunavanje inverznog korena $1/\sqrt[m]{y}$ lako se sprovodi primenom Njutnovog metoda (4.6). Primenimo ovaj metod na funkciju

$$f(x) = y - x^{-m},$$

gde je m celobrojna pozitivna konstanta. Kako je $f'(x) = mx^{-(m+1)}$, Njutnov metod (4.6) u ovom slučaju daje iterativnu formulu

$$x_{k+1} = x_k + x_k(1 - x_k^m y)/m. \tag{4.7}$$

Niz aproksimacija $\{x_k\}$ konvergira ka nuli $\zeta = y^{-1/m}$ ako je početna aproksimacija x_0 dovoljno bliska nuli ζ. Posmatrajući formulu (4.7) uočavamo vrlo

[4]FMA - od *fused multiply-add* - objedinjeno množenje i sabiranje.

[5]SRT algoritam za deljenje nosi naziv po autorima Sweeney-u, Robertson-u i Tocher-u. Ovaj algoritam koristi lukap tabelu.

povoljnu okolnost da (4.7) ne sadrži deljenje, izuzev deljenja[6] celobrojnom konstantom m.

U specijalnom slučaju možemo lako izračunati recipročnu vrednost $1/y$ (slučaj $m = 1$) pomoću (4.7). Formula (4.7) tada dobija jednostavan oblik

$$x_{k+1} = x_k + x_k(1 - x_k y). \tag{4.8}$$

Posmatrajući iterativnu formulu (4.8) nameće se pitanje zašto umesto nje ne koristimo jednostavniju formulu

$$x_{k+1} = x_k(2 - x_k y) \tag{4.9}$$

koja se direktno dobija iz (4.8). Mada matematički ekvivalentne, formule (4.8) i (4.9) nisu ekvivalentne sa gledišta računske efikasnosti. Da bismo ovo pokazali, pretpostavimo da u formuli (4.8) x_k aproksimira $1/y$ sa tačnošću od $b/2$ bitova, tj.,

$$x_k = \frac{1}{y} + \alpha 2^{-b/2}, \quad \text{ili} \quad 1 - x_k y = O(2^{-b/2}),$$

gde je $\alpha = O(1)$ izvesna konstanta. Tada je

$$\begin{aligned} x_{k+1} &= \frac{1}{y} + \alpha 2^{-b/2} + \Big(\frac{1}{y} + \alpha 2^{-b/2}\Big)\Big(1 - \Big(\frac{1}{y} + \alpha 2^{-b/2}\Big)y\Big) \\ &= \frac{1}{y} + \alpha 2^{-b/2} + \Big(\frac{1}{y} + \alpha 2^{-b/2}\Big)\Big(-\alpha y 2^{-b/2}\Big) = \frac{1}{y} - \alpha^2 y 2^{-b}, \end{aligned}$$

odakle je

$$x_{k+1} - \frac{1}{y} = O(2^{-b}).$$

Dakle, proizvod veličina x_k i $1 - x_k y$ može se izračunati sa preciznošću od $b/2$ bitova da bi se dobila aproksimacija x_{k+1} sa preciznošću od b bitova. S druge strane, u jednostavnijoj formuli (4.9) imamo da je $2 - x_k y = 1 + O(2^{-b/2})$, odakle sledi da proizvod veličina x_k i $2 - x_k y$ mora biti izračunat sa punom preciznošću od b bitova da bi se dobila aproksimacija x_{k+1} sa preciznošću od b bitova.

[6]Konstanta $c_m = 1/m$ izračuna se na početku iterativnog procesa tako da se u nastavku vrši množenje sa c_m.

Izračunavanje kvadratnog korena

Stavljajući $m = 2$ u (4.7) dobijamo iterativnu formulu

$$x_{k+1} = x_k + x_k(1 - x_k^2 y)/2 \tag{4.10}$$

za generisanje niza aproksimacija $\{x_k\}$ koji konvergira ka $1/\sqrt{y}$ za dovoljno blisku početnu aproksimaciju x_0.

Gornja formula za $y^{-1/2}$ može se iskoristiti za efikasno (u smislu male računske cene) izračunavanje korena $\sqrt{y}$ uz pomoć samo jednog množenja,

$$\sqrt{y} = y \times y^{-1/2}.$$

Primećujemo da se i u ovom slučaju ne javlja deljenje, izuzev konstantom 2. Iterativna formula (4.10) je efikasnija od formule koja se dobija ako se Njutnov metod direktno primeni na funkciju $f(x) = x^2 - y$, što daje

$$x_{k+1} = \frac{1}{2}\Big(x_k + \frac{y}{x_k}\Big). \tag{4.11}$$

Ovo je poznata Heronova formula izvedena pre 2000 godina.[7] Nedostatak ove formule je što ona zahteva deljenje sa x_k, koje je uvek „skuplje" (u smislu računske efikasnosti, odnosno utrošenog procesorskog vremena).

Izračunavanja recipročne vrednosti $1/y$ i recipročnog kvadratnog korena $1/\sqrt{y}$ nisu od velikog interesa sama po sebi. Njihov značaj ogleda se u primeni za realizaciju operacije ***deljenja dva broja*** b/a i ***kvadratnog korena***

$$\sqrt{a} = a \times 1/\sqrt{a}.$$

Pri ovome se, najpre, iterativnim putem (primenom Njutnove iteracije (4.7)) određuje dovoljno dobra aproksimacija x_k za $1/a$, a zatim se količnik b/a nalazi kao $b/a \approx b \times x_k$. Slično, aproksimacija x_k izračunata iterativno pomoću (4.10) množi se sa a da bi se dobio koren $\sqrt{a} \approx a \times x_k$. Primećujemo da je u oba slučaja na ovaj način izbegnuta „skupa" operacija deljenje.

[7] Ovu formulu izveo je Heron Aleksandrijski, koji je živeo u I veku. Međutim, prema podacima sa Vavilonske ploče stare oko 4000 godina koja se čuva na Jejlskom univerzitetu (SAD), postupak (opisan rečima) ekvivalentan formuli (4.11) bio je poznat vavilonskim matematičarima.

U finalnom koraku rezulat deljenja $b/a \approx b \times x_k$ i kvadratnog korena $\sqrt{a} \approx a \times x_k$ dobijaju se finalnim množenjem. U slučaju aritmetike standardne pa u dvostruke preciznosti računska efikasnost je zadovoljavajuća. Međutim, u slučaju AVP postoji problem računske cene jer se finalna množenja u oba slučaja moraju izvršiti sa punom preciznošću. Dok se pri iterativnom procesu, naročito u početku, može koristiti aritmetika niže preciznosti, poslednji korak koji se sastoji u množenju $b/a \approx b \times x_k$ ili $\sqrt{a} \approx a \times x_k$, je veoma skup.

Da bi se izbegao ovaj nedostatak, množenje treba izvršiti u okviru iterativnog procesa: tako poslednja iteracija za deljenje postaje

$$y_{k+1} = y_k + x_k(b - ay_k),$$

gde je $y_k = bx_k$, dok je u slučaju kvadratnog korena

$$y_{k+1} = y_k + \frac{x_k}{2}(a - y_k^2), \quad y_k = ax_k.$$

Iterativni metodi visokog reda konvergencije

Kao što je napomenuto, red konvergencije Njutnovog metoda je dva. Postoji veliki broj iterativnih metoda za rešavanje nelinearnih jednačina sa višim redom konvergencije. U nastavku dajemo jedan originalni pristup za generisanje formula visokog reda. Uvedimo oznake

$$u = u(x) = \frac{f(x)}{f'(x)}, \quad B_k(x) = \frac{f^{(k)}(x)}{k!f'(x)}, \quad I_2(x) = x - u(x).$$

Tada je Njutnov metod definisan iterativnom formulom

$$x_{k+1} = I_2(x_k) \quad (k = 0, 1, \ldots)$$

polazeći od neke početne aproksimacije x_0. U svojoj knjizi [141] iz 1964. Traub je izveo diferencijalno-diferencnu jednačinu oblika

$$I_{n+1}(x) = I_n(x) - \frac{u(x)}{n} I_n'(x) \quad (n \geq 2) \tag{4.12}$$

koja, polazeći od Njutnove iterativne formule $I_2(x)$, generiše iterativne metode višeg reda. Red konvergencije metoda (4.12) je $n + 1$. Na primer, za $n = 2$ dobija se iterativni metod trećeg reda (poznat kao Čebiševljev metod)

$$I_3(x) = I_2(x) - \frac{u(x)}{k} I_2'(x) = I_2(x) - B_2(x)u(x)^2. \tag{4.13}$$

Stavljajući $n = 3$ u (4.12) i koristeći $I_3(x)$, dobija se metod četvrtog reda

$$I_4(x) = I_3(x) - (2B_2(x)^2 - B_3(x))u(x)^3,$$

itd.

Posmatrajmo *Čebiševljevu iterativnu formulu*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f(x_k)^2 f''(x_k)}{2f'(x_k)^3} \tag{4.14}$$

koja se dobija iz (4.13) i primenimo je za izračunavanje recipročne vrednosti koristeći funkciju $f(x) = y - 1/x$. Dobijamo

$$x_{k+1} = x_k(1 - x_k y) + x_k(1 - x_k y)^2 = x_k(1 + \varepsilon_k + \varepsilon_k^2), \tag{4.15}$$

gde smo stavili $\varepsilon_k = 1 - x_k y$ i pretpostavili da je $|\varepsilon_k| \leq 1$. Oblik formule (4.15) sugeriše da je metod n-tog reda oblika

$$x_{k+1} = x_k(1 + \varepsilon_k + \varepsilon_k^2 + \cdots + \varepsilon_k^{n-1}) \ (= I_n(x_k)). \tag{4.16}$$

Tačnost formule (4.16) dokazuje se indukcijom.

Kao sledeći važan primer, posmatrajmo iterativne metode višeg reda generisane pomoću Traubove relacije za izračunavanje e^y. Očigledno je da je e^y koren funkcije $f(x) = y - \ln x$, tako da primenom Njutnove formule na ovu funkciju dobijamo iteraciju

$$x_{k+1} = x_k + x_k(y - \ln x_k).$$

Traubova relacija za posmatranu funkciju generiše Čebiševljev metod trećeg reda u obliku

$$x_{k+1} = x_k + x_k(y - \ln x_k) + \frac{1}{2}x_k(y - \ln x_k)^2.$$

S obzirom na činjenicu da je $(\ln x)' = 1/x$, lako se dolazi do slične formule kao u slučaju recipročne vrednosti, gde treba staviti $\varepsilon_k = y - \ln x_k = \ln(x/x_k)$. Koristeći razvoj u stepeni red za eksponencijalnu funkciju, dobija se

$$\frac{x}{x_k} = \exp \varepsilon_k = \sum_{r=0}^{\infty} \frac{\varepsilon_k^r}{r!}.$$

Prekidajući ovaj red posle n članova, dobija se Tejlorov polinom za izračunavanje aproksimacije za e^y,

$$x_{k+1} = x_k\left(\sum_{r=0}^{n-1}\frac{\varepsilon_k^r}{r!}\right).$$

Kao što smo videli, Njutnov metod primenjen za izračunavanje inverznog kvadratnog korena $1/\sqrt{y}$ daje iterativni proces (4.10). Umesto ove iteracije može se primeniti Čebiševljev metod (4.14) na funkciju $f(x) = y - 1/x^2$ koji daje iteraciju sa redom konvergencije tri,

$$x_{k+1} = x_k + \frac{1}{2}x_k\left(\varepsilon_k + \frac{3}{4}\varepsilon_k^2\right), \quad \varepsilon_k = 1 - yx_k^2. \tag{4.17}$$

Čebiševljev metod (4.14) primenjen na funkciju $f(x) = y - 1/x$ daje iteraciju trećeg reda

$$x_{k+1} = x_k + x_k(\varepsilon_k + \varepsilon_k^2), \quad \varepsilon_k = 1 - yx_k \tag{4.18}$$

koja konvergira ka recipročnoj vrednosti $1/y$.

Važno je napomenuti da je pri upoređivanju različitih metoda za realizaciju osnovnih aritmetičkih operacija i za izračunavanje elementarnih funkcija iterativnim postupcima (ili, opštije, pri rešavanju nelinearnih jednačina u AVP) zgodno pretpostaviti da se osnovne aritmetičke operacije (sabiranje, množenje, itd.) izvršavaju u fiksnoj preciznosti. Međutim, kod primene raznih algoritama za izračunavanje ili rešavanje jednačina sa stanovišta računske efikasnosti poželjno je koristiti AVP promenljive preciznosti. Mnogo efikasnije je koristiti aritmetike niže preciznosti u početnim iteracijama jer tada aproksimacije imaju samo nekoliko tačnih cifara, i povećavati preciznost u toku iterativnog procesa, da bi finalni rezultat bio dobijen sa visokom preciznošću. Na ovaj način se smanjuje vreme izvršavanja i time povećava računska efikasnost primenjenog metoda.

Zahtevani broj značajnih decimalnih cifara se zadaje na početku procesa izračunavanja ili rešavanja jednačine. Kod iterativnog izračunavanja primenom metoda $x_{k+1} = g(x_k)$ sa redom konvergencije q, gde je broj tačnih cifara aproksimacije q puta veći u odnosu na aproksimaciju iz prethodne iteracije (kod Njutnovog metoda je $q = 2$), radi se sa promenljivom (*dinamičkom*) preciznošću. Sledeći kratak primer napisan je u programskom paketu *Mathematica*:

Data je iterativna funkcija g,
početna aproksimacija $x = x_0$ i
početna preciznost od $P = 8$ značajnih cifara

```
P=8; For[k=0, k<10, k++, x1=SetAccuracy[g[x],P*(k+1)]; x=x1]
```

Naredbom `SetAccuracy[g[x],P*(k+1)]` za vreme iterativnog procesa kada k uzima vrednosti $0, 1, 2, \ldots$ nova aproksimacija `x1` izračunava se sa preciznošću redom 8, 16, 24, $\ldots$ $(= P \cdot (k+1))$.

Računska cena osnovnih operacija

U nastavku ćemo razmatrati računsku cenu osnovnih aritmetičkih operacija u AVP u funkciji broja bitova b koji definiše preciznost primenjene aritmetike. Pritom ćemo koristiti ranije izvedene formule iterativne prirode.

Neka je A operacija u AVP sa b-preciznošću i neka $t_b(A)$ označava vreme za izvršenje operacije A uzimajući najgori slučaj i pretpostavljajući da je rezultat dobijen sa relativnom tačnošću od najviše $2^{-b}c$, gde je c nezavisno od b. Da bismo naglasili da se radi o operacijama u aritmetici višestruke preciznosti, ove operacije zvaćemo *vp* (višestruka preciznost) operacije. Sa D, I, M, Q, R, S označavamo operacije deljenje, recipročnu vrednost, množenje, inverzno korenovanje $(1/\sqrt{a}\,)$, korenovanje $(\sqrt{a}\,)$ i kvadriranje. Koristimo notaciju $f \sim g$ da označimo da je $\lim_{b\to\infty} f(b)/g(b) = 1$, a $f = O(g)$ znači da je $|f(b)| \leq Kg(b)$ za neku konstantu K i dovoljno veliko b.

Hopcroft [68] je uveo sledeću definiciju linearne zavisnosti između operacija:

Definicija 4.1. Operacija A je *linearno svodljiva* na operaciju B (u oznaci $A \leq B$) ako postoji pozitivna konstanta K takva da je

$$t_b(A) \leq Kt_b(B)$$

za dovoljno veliko b. A je *linearno ekvivalentna* operaciji B (u oznaci $A \equiv B$) ako je istovremeno $A \leq B$ i $B \leq A$. ▲

Na osnovu prethodne diskusije prizilazi da je $t_b(M) \geq t_b(A)$, te na osnovu Definicije 4.1 sledi da je $M \geq A$. S druge strane je $M \equiv S$ jer je kvadriranje specijalan slučaj množenja.

Dobro je poznato da *vp* operacija sabiranja A (takođe i oduzimanja) ima cenu $t_b(A) = O(b)$. U slučaju klasičnog množenja je $t_b(M) = O(b^2)$.

Karacuba i Ofman [76] su koristeći Karacubin algoritam za množenje u slučaju dovoljno velikog b, dobili vreme

$$t_b(M) = O\big(b^{\log_2 3}\big) = O(b^{1.58\ldots}).$$

Kasnije su Schönhage i Strassen [123] pokazali da se ovo vreme može još smanjiti (podrazumevajući opet dovoljuno veliko b) na

$$t_b(M) = O\big(b\,\log(b)\,\log\log(b)\big) \tag{4.19}$$

primenjujući algoritam koji koristi Furijeovu brzu transformaciju. Knuth [80] je izložio hipotezu da je Šenage-Štrasenov algoritam optimalan, ali strog dokaz nije još dat. U teoriji kompleksnosti radi se sa opštijom ocenom

$$M(b) \sim cb\big[\log(b)\big]^{\beta}\big[\log\log(b)\big]^{\gamma},$$

odakle se pokazuje da važe uslovi

$$M(\alpha b) \sim \alpha M(b), \quad \text{tj.} \quad \lim_{b\to\infty} \frac{M(\alpha b)}{\alpha M(b)} = 1. \tag{4.20}$$

Posmatrajmo APT preciznosti b-bitova. Neka je $M(b)$ broj operacija za množenje dva prirodna broja u intervalu $[0, 2^b]$. Tada se na osnovu (4.20) može dokazati sledeće tvrđenje.

Lema 4.1. *Ako je* $0 < \alpha < 1$, $M(b) = 0$ *za* $b < 1$, *i* $c_1 < \frac{1}{1-\alpha} < c_2$, *tada je*

$$c_1 M(b) < \sum_{k=0}^{\infty} M(\alpha^k b) < c_2 M(b)$$

za dovoljno veliko b. ▲

U nastavku ukratko dajemo relacije između *vp* operacija D, I, M, Q, R, S. U radu [16] dokazano je da ako se neka od ovih operacija (recimo Y) može realizovati sa preciznošću b za vreme $t_b(Y)$, tada se druge operacije iz navedenog skupa mogu izvršiti sa preciznošću b za vreme $Kt_b(Y)$, gde je K multiplikativna konstanta.

Definicija 4.2. C_{XY} je minimalna konstanta takva da, za proizvoljno pozitivno ε i dovoljno veliko b, operacija X se može realizovati sa preciznošću

b u vremenu $(C_{XY}+\varepsilon)t_b(Y)$ ako se operacija Y može izvršiti u vremenu $t_b(Y)$, gde je $X, Y \in \{D, I, M, Q, R, S\}$. ▲

Sledeće nejednakosti odmah slede iz Definicije 4.2,

$$C_{XY}C_{YZ} \leq C_{XZ}$$

i

$$C_{XY}C_{YX} \geq C_{XX} = 1.$$

Konstante C_{XY} nije lako odrediti jer one zavise od više parametara, od kojih je najvažniji algoritam koji se primenjuje. Na primer, nije svejedno da li se koristi Njutnov metod (4.6) drugog reda ili Čebiševljev metod (4.14) trećeg reda, ili da li se koristi klasičan način množenja sa cenom $t_n(M) = O\big(n^2\big)$ ili Šenage-Štrasenov algoritam sa cenom $t_b(M) = O\big(b \log(b) \log\log(b)\big)$. Ipak, da bi se došlo do ocena ovih konstanti, autori su u svojim radovima uvodili izvesne dodatne uslove. Na primer, Brent [16] uvodi uslov

$$t_{\alpha b}(Y) \leq (\alpha + \varepsilon)t_b(Y), \tag{4.21}$$

za proizvoljne pozitivne vrednosti α i ε i koristi uslov

$$\lim_{b\to\infty} \frac{t_b(M)}{b} = \infty, \tag{4.22}$$

što proizilazi iz Šenage-Štrasenovog algoritma za množenje u AVP, videti radove [32], [101] i [100]. Sa pretpostavkama (4.20), (4.21) i (4.22) formiramo tabelu 4.1 sa gornjim granicama za konstante C_{XY}.

	$X = D$	I	M	Q	R	S
$Y = M$	3.5	3	1	4.5	5.5	1

Tabela 4.1 Gornje granice za C_{XY}

Izračunavanje gornjih granica ilustrovaćemo na tri primera koja se najčešće javljaju u praksi. Razmatrane granice su oivičene kvadratom u tabeli 4.1.

Primer 4.2. Pokažimo, najpre, da je

$$C_{IM} \leq 3,$$

ili, što je ekvivalentno,

$$I(b) \sim 3M(b). \tag{4.23}$$

Podsećamo da je iterativna formula za nalaženje recipročne vrednosti $1/a$, dobijena primenom Njutnovog metoda na funciju $f(x) = a - 1/x$, data pomoću

$$x_{k+1} = x_k + x_k\varepsilon_k,$$

gde je $\varepsilon_k = 1 - ax_k$ (videti (4.8)). S obzirom da je red konvergencije Njutnovog metoda jednak dva, to je

$$\varepsilon_{k+1} = \varepsilon_k^2, \quad \varepsilon_k = (\varepsilon_0)^{2^k}.$$

Ako pretpostavimo (bez gubitka opštosti) da radimo u binarnoj aritmetici i da je $\varepsilon_0 = 2^{-1}$, željena tačnost 2^{-b} dobiće se posle m iteracija, tj. važi

$$\varepsilon_m = 2^{-b} = \left(\varepsilon_0\right)^{2^m} = \left(2^{-1}\right)^{2^m} = 2^{-2^m},$$

odakle posle logaritmovanja dobijamo da je potreban broj iteracija jednak $m \sim \log_2 b$.

Ako je $\varepsilon_k = O(2^{-b/2})$, iz relacije $\varepsilon_{k+1} = \varepsilon_k^2$ zaključujemo da je $\varepsilon_{k+1} = O(2^{-b})$ tako da je u poslednjoj iteraciji dovoljno izvršiti množenje brojeva x_k i $\varepsilon_k = 1 - ax_k$ koristeći preciznost $b/2$ bitova, mada ax_k mora biti izračunato sa preciznošću b.

Prema tome, vreme potrebno za poslednju iteraciju je

$$M(b) + M(b/2) + O(n).$$

Dodati član $O(n)$ odnosi se na potrebna sabiranja i oduzimanja čija je računska cena $O(b)$ i ovaj član je asimptotski zanemarljiv[8] u poređenju sa vremenom za množenje.

Vreme za pretposlednju iteraciju je

$$M(b/2) + M(b/4) + O(b/2)$$

jer ova iteracija jedino treba da izračuna aproksimaciju preciznosti $O(2^{-b/2})$. Na ovaj način nastavljamo sa analizom prema prvoj iteraciji. Koristeći (4.20)

[8]Dakle, pretpostavlja se da je b dovoljno veliko.

u obliku $M(2^{-r}b) \sim 2^{-r}M(b)$ i Lemu 4.1, nalazimo da je ukupno vreme za izračunavanje recipročne vrednosti dato sa

$$\begin{aligned} I(b) &\sim \Big(M(b) + M(\tfrac{b}{2})\Big) + \Big(M(\tfrac{b}{2}) + M(\tfrac{b}{4})\Big) + \Big(M(\tfrac{b}{4}) + M(\tfrac{b}{8})\Big) + \cdots \\ &\sim \Big(M(b) + 2^{-1}M(b) + 2^{-2}M(b) + \cdots\Big) + \\ &\quad \Big(2^{-1}M(b) + 2^{-2}M(b) + 2^{-3}M(b) + \cdots\Big) \\ &\sim \Big(1 + \frac{1}{2}\Big)\Big(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots\Big)M(b) \\ &= \frac{3}{2} \cdot \frac{1}{1 - \frac{1}{2}}M(b) = 3M(b). \ \blacktriangle \end{aligned}$$

Primer 4.3. U ovom primeru pokazaćemo da je

$$C_{DM} \leq 3.5,$$

ili, što je ekvivalentno,

$$D(b) \sim 3.5M(b).$$

Za realizaciju operacije deljenja c/d iskoristićemo Njutnovu iteraciju (4.8) kojom se nalazi aproksimacija x_k recipročne vrednosti $1/d$. Ako $x_k \to 1/d$ kao u (4.8), tada $y_k \to c/d$. Takođe, ako x_k zadovoljava iterativnu formulu (4.8), tada y_k zadovoljava relaciju

$$y_{k+1} = y_k + x_k(c - dy_k). \tag{4.24}$$

Pretpostavimo da se x_k izračunava iterativno po formuli (4.8) sa preciznošću od $b/2$ bitova. Na osnovu (4.23) sledi da je za ovo izračunavanje potrebno vreme

$$I(b/2) \sim 3M(b/2) \sim \frac{3}{2}M(b).$$

Ako u (4.24) izračunavamo proizvod $y_k = c \cdot x_k$ sa preciznošću $b/2$, potrebno je vreme $M(b/2)$. Kao i ranije, proizvod brojeva x_k i $c - dy_k$ može se izračunati sa preciznošću $b/2$ a korekcija $c - dy_k$ sa preciznošću b da bismo dobili y_{k+1} u b-preciznosti. Ukupno vreme dobijamo sabiranjem

$$D(b) \sim \underbrace{3M(b/2)}_{x_k} + \underbrace{M(b/2)}_{y_k} + \underbrace{M(b)}_{d-cy_k} + \underbrace{M(b/2)}_{x_k(c-dy_k)} \sim 3.5M(b). \ \blacktriangle$$

Primer 4.4. Da bismo aproksimirali $1/\sqrt{a}$, koristićemo iteraciju sa redom konvergencije tri,

$$x_{k+1} = x_k + \frac{1}{2}x_k\Big(\varepsilon_k + \frac{3}{4}\varepsilon_k^2\Big), \tag{4.25}$$

gde je

$$\varepsilon_k = 1 - ax_k^2.$$

Iterativni metod (4.25) se dobija primenom Čebiševljevog metoda (4.14) na funkciju $f(x) = a - 1/x^2$.

U poslednjoj iteraciji potrebno je izračunati ax_k^2 sa preciznošću b, ε_k^2 sa preciznošću $b/3$, i proizvod $x_k(\varepsilon_k + \frac{3}{4}\varepsilon_k^2)$ sa preciznošću $2b/3$. S obzirom da je metod reda 3, na sličan način kao u primerima 2 i 3 dobijamo

$$C_{QM} \leq \Big(1 + \frac{1}{3} + \frac{2}{3}\Big)\Big(1 + \frac{1}{3} + \frac{1}{3^2} + \cdots\Big) = \frac{9}{2}. \tag{4.26}$$

Na osnovu ovog i relacije $\sqrt{a} = a \times 1/\sqrt{a}$ sledi direktno da je

$$C_{RM} \leq C_{QM} + M(b) = 4.5.$$

Napomenimo da je granica data u (4.26) preciznija nego granica $C_{QM} \leq 5$ koja se dobija iz iterativnog procesa (primenjujući Njutnov metod na $f(x) = a - 1/x^2$)

$$x_{k+1} = x_k + \frac{1}{2}x_k\varepsilon_k. \quad \blacktriangle$$

Operacije sa kompleksnim brojevima (višestruka preciznost)

Radeći sa kompleksnim brojevima podrazumevamo da preciznost od b bitova kompleksnog broja $z = x + iy$ znači da svaki od realnih brojeva u paru (x, y) ima preciznost b. Kao i ranije, pod preciznošću b neke operacije podrazumeva se da ona daje rezultat čija apsolutna vrednost ima relativnu grešku $O(2^{-b})$.

Kompleksno množenje

S obzirom da je $z = (s + it)(u + iv) = (su - tv) + i(su + tv)$, sledi da kompleksno množenje može biti izračunato pomoću 4 realna množenja i 2 realna sabiranja (svuda u ovom radu, kada se radi o računskoj ceni, ne

pravimo razliku između operacija sabiranja i oduzimanja). Međutim, u AVP može se zgodno iskoristiti jednostavna ideja iz rada Karacube i Ofmana [76] da bi se izračunavanje svelo na 3 realna množenja i 6 sabiranja;

1) Izračunati $p_1 = su,\ p_3 = tv,\ p_3 = (s+t)(u+v)$;

2) Rezultat kompleksnog množenja dat je pomoću

$$z = (s+it)(u+iv) = p_1 - p_2 + i(p_3 - p_1 - p_2).$$

Prikazani metod množenja kompleksnih brojeva često se u literaturi zove „***3M metod***" jer zahteva 3 realna množenja.

S obzirom da je $|s+t| \leq \sqrt{2}|s+it|$ i $|u+v| \leq \sqrt{2}|u+iv|$, imamo

$$|(s+t)(u+v)| \leq 2|z|.$$

Prema tome, sve greške zaokruživanja su reda $2^{-b}|z|$ ili manje, i izračunati proizvod kompleksnih brojeva ima relativnu grešku $O(2^{-b})$. Vreme za izvršenje 6 sabiranja u prikazanom množenju u AVP je asimptotski zanemarljivo u poređenju sa 3 množenja tako da se kompleksno množenje kompleksnih brojeva preciznosti b može izvršiti u vremenu $\sim 3M(b)$, dakle u vremenu približno jednakom vremenu za 3 realna množenja preciznosti b. 3M množenje ima svrhe samo u onim situacijama kada je množenje (realnih brojeva) znatno skuplja operacija od sabiranja, na primer, u aritmetikama višestruke preciznosti.

S obzirom da je kod 3M metoda komutativnost irelevantna, 3M metod se može efikasno upotrebiti pri množenju kompleksnih matrica. Neka je $A = A_1 + iA_2$ i $B = B_1 + iB_2$, gde su A_j, B_j realne matrice dimenzije $n \times n$, i definišimo $C = C_1 + iC_2 = AB$. Tada se C može izračunati pomoću tri matrična množenja kao

$$\begin{aligned} T_1 &= A_1B_1, \quad T_2 = A_2B_2, \\ C_1 &= T_1 - T_2, \\ C_2 &= (A_1 + A_2)(B_1 + B_2) - T_1 - T_2. \end{aligned}$$

Ovo izračunavanje zahteva $3n^3$ skalarnih množenja i $3n^3 + 2n^2$ skalarnih sabiranja. Klasično množenje matrica po formuli $C = A_1B_1 - A_2B_2 + i(A_1B_2 + A_2B_1)$ zahteva $4n^3 - 2n^2$ sabiranja. Za (recimo) $n \geq 30$ ušteda iznosi oko 25%.

Kompleksno kvadriranje

Kako je $(x+iy)^2 = (x-y)(x+y) + 2ixy$, kompleksno kvadriranje može se izračunati sa 2 realna množenja i 2 realna sabiranja, dakle u vremenu $\sim 2M(b)$ (opet uzimamo da je vreme za sabiranje u AVP asimptotski zanemarljivo u poređenju sa množenjem).

Kompleksno deljenje

Koristeći 3M metod za kompleksno množenje i isti algoritam za deljenje kao u realnom slučaju, može se pokazati da kompleksno deljenje zahteva vreme $\sim 12M(b)$. Međutim, proces se može ubrzati ako se koristi identitet

$$\frac{s+it}{u+iv} = (u^2+v^2)^{-1} \cdot [(s+it)(u-iv)],$$

svodeći problem na jedno kompleksno množenje ($\sim 3M(b)$), 4 realna množenja ($4M(b)$), jedno nalaženje recipročne vrednosti ($I(b) \sim 3M(b)$) i nekoliko sabiranja. Ovo daje vreme $\sim 10M(b)$. U specijalnom slučaju dobijamo vreme $\sim 7M(b)$ za kompleksnu recipročnu vrednost.

Kompleksni kvadratni koren

Ako primenimo Njutnovu formulu na kompleksnu funkciju $f(z) = z^2 - a$ (a je kompleksan broj) dobija se iterativni proces

$$z_{k+1} = z_k + \Big(\frac{a - z_k^2}{2z_k}\Big).$$

U poslednjoj iteraciji treba primeniti jedno kompleksno kvadriranje preciznosti b (što zahteva vreme $\sim 2M(b)$) i jedno kompleksno deljenje preciznosti $b/2$ (zahteva vreme $\sim 10M(b/2)$). S obzirom na kvadratnu konvergenciju Njutnovog metoda, kao i ranije izračunavamo da je vreme za kompleksni kvadratni koren srazmerno

$$\begin{aligned}\Big(2M(b) + 10M(b/2)\Big)\Big(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots\Big) &\sim \Big(2M(b) + \frac{10}{2}M(b)\Big) \cdot \frac{1}{1-\frac{1}{2}} \\ &= 7M(b) \cdot 2 = 14M(b).\end{aligned}$$

Napomenimo da je veliki doprinos poboljšanju efikasnosti operacija deljenje i korenovanje kompleksnih brojeva dao M. Ercegovac [43].

4.4 Efikasnost iterativnih metoda

Efikasnost nekog algoritma u savremenim uslovima koji podrazumevaju primenu računara izražava se na jednostavan i intuitivan, prirodan način koji je speopšte prisutan pri oceni kvaliteta nekog proizvoda:

$$\text{Efikasnost} = \frac{\text{kvalitet (pozitivne osobine)}}{\text{cena realizacije}}.$$

Cena realizacije pri korišćenju računara je obično izražena vremenom potrebnim za realizaciju algoritma, pri čemu se podrazumeva da algoritam treba da ostvari očekivane performanse, recimo brzinu rasta izvesne veličine, broj traženih informacija ili tačnost aproksimacije. U ovom poglavlju bavićemo se jedino algoritmima koji vrše izvesna izračunavanja ili rešavaju jednačine do tražene tačnosti. Jasno je da je algoritam utoliko efikasniji ukoliko je rezultat izračunavanja tačniji, a utrošeno vreme računara pri ovoj realizaciji manje.

Poslednji intuitivan zahtev doveo je do nekoliko definicija ocena efikasnosti algoritama kada su u pitanju algoritmi za iterativno rešavanje nelinearnih jednačina oblika $f(x) = 0$ (podrazumevajući tu i sisteme linearnih jednačina). S obzirom da se brzina algoritma izražava preko reda konvergencije, Ostrovski [106] je uveo meru računske efikasnosti definišući *indeks efikasnosti*

$$E_1(q, d) = \frac{q}{d}, \tag{4.27}$$

gde je q red konvergencije posmatranog iterativnog metoda, a d broj novih informacija zahtevanih u jednoj iteraciji. To je obično broj novih funkcijskih izračunavanja u jednoj ili više tačaka, pri čemu se osim funkcije f ubrajaju i njeni izvodi f', f'', ..., ukoliko se koriste. *Količina novih informacija* ili *računska cena* d može da uključi i dodatne operacije, kao što su izračunavanje suma i proizvoda, operacije korenovanja, itd. S obzirom da se radi o raznorodnim informacijama, pomenuti brojevi novih izračunavanja se uzimaju sa izvesnim težinama. Na primer, ako su $d_\oplus$, $d_\ominus$, $d_\otimes$ i $d_\oslash$ brojevi osnovnih operacija sabiranja, oduzimanja, množenja i deljenja, svaki od ovih brojeva se množi izvesnom težinom koja je proporcionalna vremenu izvršenja posmatrane operacije i tako dobijena suma predstavlja računsku cenu d. O ovome će biti više reči kasnije.

Jedna alternativna definicija efikasnosti je data formulom

$$E_2(q, d) = q^{1/d}. \tag{4.28}$$

Ova formula može se preformulisati na sledeći način koristeći logaritmovanje:

$$E_3(q, d) = \frac{\log q}{d}. \tag{4.29}$$

Indeksi efikasnosti definisani pomoću funkcija E_1, E_2 i E_3, dati formulama (4.27), (4.28) i (4.29), su monotono rastuće funkcije po argumentu q (red konvergencije) i monotono opadajuće funkcije po argumentu d (računska cena). To znači da se suštinski ne razlikuju i da daju isti poredak metoda koje želimo međusobno da uporedimo. Formula (4.29) može se izvesti polazeći od Brentove definicije efikasnosti iterativnog metoda datog u radu [12] za iterativne metode za rešavanje sistema nelinearnih jednačina.

U nastavku ćemo uvesti jedan nov pristip za izračunavanje efikasnosti algoritama za istovremeno nalaženje svih nula polinoma. Ranije su se težine koje se dodeljuju operacijama pri izračunavanju indeksa efikasnosti izražavale preko vremena izvršavanja ovih operacija. Ovaj način nije bio pogodan jer su težine zavisile od upotrebljenog procesora i drugog harverskog oruđa. S obzirom na veliki broj modela procesora sa različitim tehničkim karakteristikama i teškoćama pri merenju vremena izvršavanja ovih operacija (koje postaje sve kraće), ovakve „vremenske" težine nisu prikladne za upotrebu.

Indeks efikasnosti koji ćemo uvesti uzima u obzir težine osnovnih aritmetičkih operacija izražene preko preciznosti upotrebljene aritmetike. Na ovaj način postiže se f-nezavisnost, tj. sva izračunavanja, uključujući funkciju f i njihove izvode, zatim sume, proizvode i elementarne operacije koje ih povezuju, svode se na osnovne aritmetičke operacije. Umesto „vremenskih" težina uvodimo težine koje zavise od preciznosti upotrebljene aritmetike izražene u bitovima. Našu pažnju u ovom poglavlju usmerićemo prvenstveno na aritmetiku višestruke preciznosti (AVP). Napominjemo da je u poslednje vreme publikovan veliki broj radova koji se odnose na vreme izračunavanja osnovnih operacija i elementarnih funkcija u AVP. Najveći deo ovih rezultata može se naći u knjizi [19] Brenta i Zimmermanna.

Uvedimo skraćenice:

n – red polinoma;

b – preciznost binarne aritmetike (u bitovima);

q – red konvergencije;

r – ukupan broj iteracija da bi se dostigla tačnost 2^{-b};

S_n – broj operacija sabiranja i oduzimanja (po iteraciji) u zavisnosti od reda polinoma n;

M_n – broj operacija množenja (po iteraciji) u zavisnosti od reda polinoma n;

D_n – broj operacija deljenja (po iteraciji) u zavisnosti od reda polinoma n;

$w_s(b) = C_s b$ – težinska funkcija za sabiranje i oduzimanje;

$w_m(b) = C_m b \log(b) \log\log(b)$ – težinska funkcija za množenje;

$w_d(b) = C_d 3.5 b \log(b) \log\log(b)$ – težinska funkcija za deljenje.

$C_s,\ C_m,\ C_d$ – konstante koje zavise od arhitekture računara, karakteristika upotrebljenog procesora i drugih hardverskih alata.

Vrednosti konstanti $C_s,\ C_m$ i C_d su zavisne od upotrebljenog hardvera i arhitekture računara. Obično se uzima da je $C_d = C_m$. Radi jednostavnije analize, ali i zbog neznatnih varijacija koje one unose budući da su reda konstante (tj., $C_s = O(1),\ C_m = C_d = O(1)$), usvojićemo da je $C_s = C_m = C_d$. Ovim postupkom dolazimo do jednostavnijih izraza za težinske funkcije osnovnih aritmetičkih operacija sa kojima ćemo nadalje raditi:

$$w_s = w_s(b) = b \quad \text{– težinska funkcija za sabiranje i oduzimanje;} \tag{4.30}$$

$$w_m = w_m(b) = b \log(b) \log\log(b) \quad \text{– težinska funkcija za množenje;} \tag{4.31}$$

$$w_d = w_d(b) = 3.5 b \log(b) \log\log(b) \quad \text{– težinska funkcija za deljenje.} \tag{4.32}$$

Primetimo da smo pri definisanju težina iskoristili rezultate iz Odeljka 4.3.

Zbog lakšeg operisanja podacima koji su u slučaju AVP uglavnom izraženi u bitovima, pri analizi efikasnosti koja sledi koristićemo binarnu aritmetiku. Pod pretpostavkom da je početna aproksimacija određena sa greškom 2^{-1}, očekuje se da iterativni metod sa redom konvergencije q u prvoj iteraciji izračuna aproksimaciju sa q tačnih binarnih cifara, u drugoj iteraciji sa q^2 tačnih decimala, itd. To znači da je najracionalnije u prvoj iteraciji koristiti AVP (u binarnoj aritmetici) sa q bitova ($b = q$), u drugoj iteraciji sa q^2 bitova, itd. Na primer, u slučaju množenja *težinska funkcija* u prvoj iteraciji jednaka je

$$w_{m,1} = b \log(b) \log\log(b) = q \log(q) \log\log(q),$$

u drugoj iteraciji

$$w_{m,2} = q^2 \log(q^2) \log\log(q^2),$$

u j-toj iteraciji

$$w_{m,j} = q^j \log(q^j) \log\log(q^j).$$

Zbog toga je *totalna težina* $W_m = w_{m,1} + \cdots + w_{m,r}$. Slična analiza važi za sabiranje i deljenje.

Na osnovu gornje diskusije, totalna računska težina pojedinih operacija, koja je jednaka sumi pojedinih težina, jednaka je

$$W = \sum_{\lambda=1}^{r} w_t(b^\lambda) \quad (w_t \in \{w_s, w_m, w_d\}). \tag{4.33}$$

S obzirom na (4.30)–(4.33), za pojedine operacije imamo sledeće totalne težine:

za sabiranje:

$$W_s = q + q^2 + \cdot + q^r = q \cdot \frac{q^r - 1}{q - 1};$$

za množenje:

$$W_m = \sum_{\lambda=1}^{r} w_m(b^\lambda) = \sum_{\lambda=1}^{r} \lambda q^\lambda \log(q) \Big(\log(\lambda) + \log\log(q) \Big);$$

za deljenje:

$$W_d = \sum_{\lambda=1}^{r} 3.5 w_m(b^\lambda) = 3.5 W_m.$$

Na osnovu gornjih rezultata nalazimo da je ukupna računska cena

$$d = S_n W_s + M_n W_m + D_n W_d. \tag{4.34}$$

Ukupan broj iteracija r potreban da bi se postigla „binarna preciznost" od 2^{-b} izračunava se iz relacije

$$(2^{-1})^b = 2^{-q^r}$$

(jer je pretpostavljena tačnost početne aproksimacije 2^{-1}), odakle je

$$r = \frac{\log b}{\log q}. \tag{4.35}$$

S obzirom da broj iteracija r treba da bude ceo broj, strogo posmatrano bilo bi potrebno da uzmemo celobrojni deo izraza na desnoj strani formule (4.35) i eventualno ga povećamo za 1. Međutim, s obzirom da ne izračunavamo već samo procenjujemo indeks efikasnosti i imajući u vidu da smo već vršili aproksimacije konstanti C_s, C_m i C_d, činimo samo nebitnu grešku radeći sa veličinom $r = \log b/(\log q)$ umesto sa celim brojem. Indeks efikasnosti izračunavamo koristeći (4.34) i (4.35),

$$E = \frac{1}{rd} = \frac{\log q}{\log b\Big(S_n W_s + M_n W_m + D_n W_d\Big)}. \tag{4.36}$$

Dobijeni izraz za indeks efikasnosti se suštinski ne razlikuje od formule (4.29), odnosno Brentovog izraza. Ipak, primetimo da je poslednja formula (4.36) mnogo preciznija od pomenutih jer uzima u obzir kompleksnost operacija (izraženu težinama). Ova formula je povezana sa jednom racionalnom primenom AVP gde se koristi dinamička preciznost (u smislu promenljive preciznosti) u različitim iterativnim koracima koristeći naredbu tipa

$$x_{k+1} = \texttt{Preciznost[}\phi(x_k), P_k\texttt{]}$$

gde P_k označava broj značajnih cifara u k-toj iteraciji a x_k aproksimaciju nule funkcije u k-toj iteraciji. Na primer, u programskom paketu *Matematica* opisnu naredbu `Precision` treba zamenitu naredbom `SetAccuracy`. Na ovaj način se znatno štedi u ukupnom vremenu izvršavanja algoritma. Zaista, bilo bi krajnje neracionalno izračunavati aproksimaciju nule funkcije u, recimo, drugom iterativnom koraku sa 32 značajne decimalne cifre ako se u ovoj iteraciji ne očekuje tačnost veća od 4 značajne cifre.

Formula za indeks efikasnosti (4.36) sadrži kao parametre red konvergencije q, broj maksimalne preciznosti b (u bitovima) i stepen polinoma n. U donjim tabelama date su vrednosti težina W_s, W_m i W_d za $q = 2, 3, 4, 5, 6$ i $b = 64, 128, 256$.

Totalne težine za sabiranje i oduzimanje W_s

$b\downarrow \ q\rightarrow$	2	3	4	5	6
64	126	120	84	155	258
128	254	363	340	780	258
256	510	1092	340	780	1554

Totalne težine za množenje W_m

$b \downarrow \ q \rightarrow$	2	3	4	5	6
64	576	649	426	1048	2124
128	1557	2923	2858	8540	2124
256	3988	11984	2858	8540	20417

Totalne težine za deljenje W_d

$b \downarrow \ q \rightarrow$	2	3	4	5	6
64	2016	2271	1492	2669	7433
128	5449	10230	10003	29893	7433
256	13959	41946	10003	29893	71460

Kao jednu informaciju koja je istorijskog karaktera, dajemo (normalizovane) težine osnovnih operacija (zasnovanih na vremenu izvršenja operacija) na digitalnim računarima koji su predstavljali vrhunsku tehnologiju osamdesetih godina dvadesetog veka.

digitalni računari	w_a	w_s	w_m	w_d
HONEYWELL DPS 6/92	1.00	1.00	3.00	5.62
VAX 11/780	1.00	1.00	1.50	4.25
IBM 4341	1.00	1.00	1.50	12.37
CRAY X-MP/2	1.00	1.00	1.17	2.33

Primena matričnog metoda u konstrukciji algoritama

U nastavku ćemo pokazati kako se matrični metod opisan u Odeljku 3.4 može upotrebiti pri konstrukciji numeričkog algoritma za rešavanje nelinearnih jednačina. U ovom slučaju taskovi od kojih se sastoji model (algoritam) su zahtevi i željene karakteristike koje treba da zadovolji algoritam koji se konstruiše. Zahtevi mogu biti različiti i sa različitim težinama, što zavisi od namene ovog algoritma. Pre svega, poželjno je da iterativni metod koji se razvija uvek konvergira ka rešenju za početne aproksimacije u što širem intervalu. Neki korisnici traže brlo brz algoritam sa realtivno malom tačnošću aproksimacija rešenja (recimo, 3 do 4 značajne cifre, za industrijske potrebe), drugi zahtevaju vrlo veliku preciznost (30 i više tačnih decimala, na primer u fizici visokih energija, i znatno više u Teoriji brojeva i tzv. Eksperimentalnoj matematici). U nekim slučajevima veoma je važno znati grešku

aproksimacije (recimo u vojnoj industriji, navigaciji) tako da algoritam mora da koristi intervalnu aritmetiku, itd. Ilustracije radi, dajemo model jednog algoritma i taskove, od kojih su neki već pomenuti. Kao i u Poglavlju 3, i ovde u konkretnom modelu X treba zameniti nekim brojem iz intervala [0,1], u zavisnosti od stepena međusobne zavisnosti.

Taskovi	A	B	C	D	E	F
A tačnost aproksimacije	0	X	X	X	X	X
B vreme izvršavanja algoritma	X	0	X	X	X	X
C granica greške aproksimacije	X	X	0	X	X	X
D dostupnost softvera	X	X	X	0	X	X
E garantovana konvergencija	X	X	X	X	0	X
F memorija računara	X	X	X	X	X	0

Dostupnost softvera podrazumeva da korisnik uglavnom na raspolaganju ima standardnu aritmetiku najviše dvostruke preciznosti. Ukoliko se žele aproksimacije vrlo velike tačnosti, potrebno je imati specijalne programske pakete kao što su *Mathematica, Maple, PARI, GNU MP* koji rade sa aritmetikom višestruke preciznosti. Ovo za posledicu ima ne samo uvećane finansijske troškove za nabavku softvera, već i povećanje vremena izvršenja algoritma, što menja vrednost elemenata u gornjoj šemi. To znači da postoji „jaka" zavisnost izmedju taskova A, B i C. Isti slučaj se javlja ako se traži gornja granica greške aprokcimacije (task C), tada treba obezbediti softver koji radi sa intervalnom aritmetikom tako da se javlja povezanost taskova C i D.

Vreme izvršenja algoritma ne predstavlja problem kod iterativnih postupaka za nalaženje jedne nule funkcije ili nula polinoma, ali je ozbiljan problem pri rešavanju sistema nelinearnih jednačina kada je dimenzija sistema velika. Primera radi, u problemima optimizacije javljaju se matrice reda 1000×1000, pa i većeg reda. U ovom slučaju su taskovi B i F u dobroj meri međusobno zavisni.

4.5 Efikasnost metoda za simultanu aproksimaciju nula polinoma

Formule za indeks efikasnosti (4.27), (4.28) i (4.29) se jednostavno koriste kod metoda za nalaženje jedne nule funkcije i upoređivanje ovih metoda

je lako. Međutim, u slučaju metoda za istovremeno nalaženje svih nula polinoma, problem je znatno složeniji jer, osim izračunavanja polinoma i njegovih izvoda, potrebno je izračunati i izvesne sume ili proizvode koji zavise od aproksimacija $x_1, \ldots, x_n$ nula $\zeta_1, \ldots, \zeta_n$ polinoma

$$P(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$

stepena n. Ove sume i proizvodi su najčešće oblika

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{(z_i - z_j)^k} \quad \text{i} \quad \prod_{\substack{j=1 \\ j \neq i}}^{n} (z_i - z_j)$$

i njihova računska cena je približno ista kao i jedno izračunavanje polinoma po Hornerovoj šemi

$$\texttt{s=a}_n\texttt{; for (k=1, k<=n, ++k, s=s*x+a}_{n-k}\texttt{); P(x)=s}$$

Jedini način da se na ispravan način izračuna indeks efikasnosti je kombinacija računskih cena izračunavanja polinoma i suma, odnosno proizvoda. Pod ispravnim načinom podrazumevamo takav pristup koji daje indeks efikasnosti proporcionalan ukupnom računskom vremenu računara potrebnom da se sve aproksimacije izračunaju sa datom preciznošću. S obzirom na složenost ovog pristupa, u literaturi je bilo vrlo malo pokušaja za određivanjem indeksa efikasnosti metoda za istovremeno određivanje svih nula polinoma, a time i za njihovo upoređivanje i rangiranje. Istraživanja poslednjih godina donela su rezultate i u ovoj oblasti publikovane u radovima [108], [109], [112] i [113]. Kao što je već rečeno u Odeljku 4.4, posebna teškoća koja se javlja pri izračunavanju indeksa efikasnosti u slučaju ovog tipa metoda je zavisnost od tipa korišćenog procesora (odnosno računara) čije procesorsko vreme direktno utiče na težinske funkcije koje učestvuju u formuli (4.36). S obzirom na stalni porast brzine izvršavanja operacija sa napretkom tehnologije, ova zavisnost je nepoželjna. Zbog toga je u 4. odeljku izložen bolji pristip korišćenjem težinskih funkicija koje direktno zavise od preciznosti izražene u bitovima primenjene aritmetike.

Kao i u slučaju metoda za određivanje jedne nule funkcije, i kod metoda za istovremeno nalaženje nula polinoma postavlja se pitanje optimalne računske efikasnosti. S obzirom na pomenutu složenost formule za indeks efikasnosti u ovom slučaju, koja je rezultat kombinovanog izračunavanja polinoma i njegovih izvoda i izvesnih suma i proizvoda, matematički pristip nije moguć.

Pod matematičkim pristupom ovde podrazumevamo nalaženje optimalnih parametara koji daju maksimum indeksu efikasnosti primenom matematičke teorije optimizacije. Umesto toga, rangiranje simultanih metoda vrši se poređenjem indeksa efikasnosti razmatranih metoda kao funkcije stepena polinoma n za izabranu preciznost (izraženu u bitovima). Najčešće posmatramo 64-bitnu, 128-bitnu i 256-bitnu arhitekturu.

Napomena 4.1. U našim istraživanjima, programi za realizaciju simultanih metoda za nule polinoma izvršavani su u programskom paketu *Mathematica* koristeći aritmetiku višestruke preciznosti. Napomenimo da ovaj programski paket (kao i veći broj drugih programskih paketa) koristi softver GNU MP (ili kraće GMP) za realizaciju aritmetike višetruke preciznosti. Razvoj softvera GMP je inicirao Granlund [52], i on se neprestano usavršava slobodnim online pristupom GNU sajtu putem mnogobrojnih predloga novih algoritama i nihovih analiza i testiranja od strane eksperata iz oblasti računarskih nauka i primenjene matematike na čijem je čelu Pol Zimerman sa Nacionalnog instituta za istraživanja u informatici i automatici (INRIA) u Nansiju (Francuska). ▲

Napomena 4.2. Kao što je napomenuto u [48], izračunavanja u AVP ne produkuju samo preciznije rezultate, već daju tačne rezultate u svim situacijama u kojima dolazi do znatnog gubitka značajnih cifara usled oduzimanja ili greške zaokruživanja. Štaviše, AVP ubrzava konvergenciju izvesnih klasa iterativnih metoda, kao što je opisano u [48]. ▲

Da bismo došli do odgovora koji je simultani metod najefikasniji, morali smo da pođemo od trenutno važeće rang liste najefikasnijih simultanih metoda. To su metodi zasnovani na Erlihovim iteracijama i u nastavku će ukratno biti opisan ovaj postupak.

Neka je

$$P(z) = \prod_{j=1}^{n} (z - \zeta_j)$$

moničan polinom[9] stepena n koji ima realne ili kompleksne nule $\zeta_1, ..., \zeta_n$. Označimo sa $h(z) = P(z)/P'(z)$ korekciju koja se javlja kod Njutnovog

[9]Pod moničnim polinomom podrazumevamo algebarski polinom $P(z) = z^n + a_1 z^{n-1} + \cdots + a_n$ čiji je vodeći koefficijent jednak 1.

metoda $\hat{z} = z - h(z)$. Koristeći logaritamski izvod

$$\frac{1}{h(z)} = \frac{P'(z)}{P(z)} = \Big[\frac{d}{dz}\log P(z)\Big]^{-1} = \Big(\sum_{j=1}^{n}\frac{1}{z-\zeta_j}\Big)^{-1}, \tag{4.37}$$

izdvajamo član $z - \zeta_i$ iz (4.37) i dobijamo relaciju

$$\zeta_i = z - \frac{h(z)}{1 - h(z)\sum\limits_{\substack{j=1\\ j\neq i}}^{n}\frac{1}{z-\zeta_j}} \quad (i = 1, \ldots, n). \tag{4.38}$$

Ova relacija je osnova za konstrukciju iterativnih metoda za istovremeno nalaženje nula polinoma.

Neka su $z_1, \ldots, z_n$ aproksimacije nula $\zeta_1, \ldots, \zeta_n$. Zamenjujući aproksimacije $z_1, \ldots, z_n$ umesto nula $\zeta_1, \ldots, \zeta_n$ na desnoj strani relacije (4.38) i uzimajući $z = z_i$, Erlihov metod trećeg reda za simultano određivanje prostih nula polinoma P sledi iz (4.38)

$$\hat{z}_i = z_i - \frac{h(z_i)}{1 - h(z_i)\sum\limits_{\substack{j=1\\ j\neq i}}^{n}\frac{1}{z_i-z_j}} \quad (i = 1, \ldots, n) \tag{4.39}$$

(videti Ehrlich [36]). Ako su aproksimacije $z_1, \ldots, z_n$ dovoljno dobre, tada je $\hat{z}_i$ nova poboljšana aproksimacija nule ζ_i. Iterativni metod (4.39) su takođe razmatrali Börsch-Supan [21] i Aberth [1].

Imajući u vidu opisanu konstrukciju Erlihovog metoda (4.39) i strukturu relacije (4.38), primećujemo da ako $z_j \to \zeta_j$ $(j \in \{1, \ldots, n\} \setminus \{i\})$, tada $\hat{z}_i \to \zeta_i$. Drugim rečima, bolje aproksimacije nula ζ_j generisaće metod koji je brži od Erlihovog metoda (4.39). Nurejn [104] je iskoristio ovu činjenicu i zamenio Njutnovu aproksimaciju $z_j - h(z_j)$ u (4.38) (umesto z_j) da izvede metod četvrtog reda

$$\hat{z}_i = z_i - \frac{h(z_i)}{1 - h(z_i)\sum\limits_{\substack{j=1\\ j\neq i}}^{n}\frac{1}{z_i-z_j+h(z_j)}} \quad (i = 1, \ldots, n). \tag{4.40}$$

Kao što je pokazano u [111], Nurejnov metod (4.40) je najefikasniji u klasi iterativnih simultanih metoda konstruisanih na bazi relacije nepokretne tačke. Zaista, ubrzanje reda konvergencije od 3 na 4 je dobijeno koristeći zanemarljiv broj dodatnih numeričkih operacija (n oduzimanja po iteraciji) jer su korišćene već prethodno izračunate korekcije $h(z_j)$.

Naš cilj je povećanje računske efikasnosti Erlihovog metoda (4.39) i Nurejnovog metoda (4.40). U ovu svrhu primenićemo pogodne korekcije izvedene iz metoda koji su brži nego Njutnov metod. Dvotačkasti metod Ostrovskog [106] i Džeretov dvotačkasti metod [74]

$$y = z - \frac{2}{3}\frac{f(z)}{f'(z)}, \quad \hat{z} = \psi(z) := z - \frac{1}{2}\frac{f(z)}{f'(z)} - \frac{f(z)}{3f'(y) - f'(z)} \tag{4.41}$$

su pravi kandidati. Kao što je pokazano u radu [74], red konvergencije ovog metoda je četiri ako je nula ζ od f prosta, to jest,

$$\psi(z) - \zeta = \mathcal{O}_m\big((z-\zeta)^4\big), \tag{4.42}$$

gde oznaka $z_1 = \mathcal{O}_m(z_2)$ ili $z_1 \overset{\text{mag}}{\sim} z_2$ označava da dva realna ili kompleksna broja z_1 i z_2 imaju module istog reda.

U nastavku ćemo podrazumevati da je funkcija f u (4.41) algebarski polinom, to jest, $f(z) \equiv P(z)$. Takođe, uvodimo sledeće aproksimacije:

$z_N = z - h(z)$ (Njutnova aproksimacija);

$z_J = z - c(z) = z - \frac{1}{2}h(z) - \dfrac{P(z)}{3P'(y) - P'(z)}, \;\; y = z - \frac{2}{3}h(z)$

(Džeretova aproksimacija).

(Index J stoji zbog autora Jarratta (Džereta)). Džeretovu korekciju označićemo sa

$$c(z) = \frac{1}{2}h(z) + \frac{P(z)}{3P'(y) - P'(z)}. \tag{4.43}$$

Uzimajući u obzir uvedene aproksimacije, kombinujemo Erlihov i Džeretov metod i izvodimo sledeći iterativni metod za simultanu aproksimaciju nula polinoma

$$z_i^{(m+1)} = z_i^{(m)} - \frac{h(z_i^{(m)})}{1 - h(z_i^{(m)}) \displaystyle\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{z_i^{(m)} - z_j^{(m)} + c(z_j^{(m)})}} \tag{4.44}$$

za $i = 1, \ldots, n$ i $m = 0, 1, 2, \ldots$, gde je Džeretova korekcija $c(z)$ data formulom (4.43).

Red konvergencije predloženog Erlih-Džeretovog metoda (4.44) je razmatran u sledećoj teoremi.

Teorema 4.1. *Neka su $z_1^{(0)}, \ldots, z_n^{(0)}$ početne aproksimacije dovoljno bliske nulama $\zeta_1, \ldots, \zeta_n$ polinoma P, tada je red konvergencije Erlih-Džeretovog metoda* (4.44) *jednak šest.*

Dokaz. Jednostavnosti radi, izostavljamo indeks iteracije m u dokazu i koristimo simbol $\widehat{\ }$ da označimo veličine u narednoj iteraciji. Uvedimo skraćenice

$$u_i = z_i - \zeta_i, \quad \hat{u}_i = \hat{z}_i - \zeta_i,$$
$$a_{ij} = z_i - z_j + c(z_j), \quad t_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{z_j - c(z_j) - \zeta_j}{(z_i - \zeta_j) a_{ij}}.$$

Umajući u vidu (4.37), polazimo od (4.44) i dobijamo

$$\begin{aligned} \hat{u}_i = \hat{z}_i - \zeta_i &= z_i - \zeta_i - \frac{1}{\dfrac{1}{u_i} + \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{z_i - \zeta_j} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{a_{ij}}} \\ &= u_i - \frac{u_i}{1 - u_i \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{z_j - c(z_j) - \zeta_j}{(z_i - \zeta_j) a_{ij}}} \\ &= u_i - \frac{u_i}{1 - u_i t_i}. \end{aligned}$$

Odavde je

$$\hat{u}_i = \frac{-u_i^2 t_i}{1 - u_i t_i}. \tag{4.45}$$

Neka je $u \in \{u_1, \ldots, u_n\}$ i neka je $\hat{u} \in \{\hat{u}_1, \ldots, \hat{u}_n\}$ greška sa najvećim modulom ali takva da je $u_i = \mathcal{O}_m(u)$ i $\hat{u}_i = \mathcal{O}_m(\hat{u})$ za svako $i \in \{1, \ldots, n\}$ (drugim rečima, sve greške imaju module istog reda). Za dovoljno bliske aproksimacije imenioci $(z_i - \zeta_j)a_{ij}$ u izrazu za t_i su ograničeni i teže ka $(\zeta_i -$

$\zeta_j)^2$ $(i \neq j)$. S obzirom na ovu činjenicu, iz (4.42) vidimo da je $t_i = \mathcal{O}_m(u^4)$ a iz (4.45) dobijamo

$$\hat{u} = \mathcal{O}_m(u^6),$$

dokazujući na ovaj način da je red konvergencije kombinovanog Erlih-Džeretovog metoda (4.44) jednak šest. □

Sada ćemo oceniti računsku efikasnost novog Erlih-Džeretovog metoda (4.44) i uporediti ovaj metod sa Erlihovim metodom (4.39) i Nurejnovim metodom (4.40).

Tabela 4.2 daje broj osnovnih aritmetičkih operacija neophodnih u primeni metoda (4.39), (4.40) i (4.44) kao funciju stepena polinoma n, podrazumevajući da su testirani realni polinomi sa realnim nulama (u realnoj aritmetici). $A(n)$, $S(n)$, $M(n)$ i $D(n)$ označavaju redom broj operacija sabiranja, oduzimanja, množenja i deljenja. U slučaju kompleksnih nula ocena računske efikasnosti je komplikovanija, ali je redosled razmatranih metoda koji sledi na osnovu vrednosti indeksa efikasnosti nepromenjen.

metodi	$A(n)$	$S(n)$	$M(n)$	$D(n)$
Erlihov metod (4.39)	$3n^2 - 3n$	$n^2 + n$	$2n^2$	$n^2 + n$
Nurejnov metod (4.40)	$3n^2 - 3n$	$n^2 + 2n$	$2n^2$	$n^2 + n$
novi metod (4.44)	$4n^2 - 4n$	$n^2 + 5n$	$3n^2 + 2n$	$n^2 + 2n$

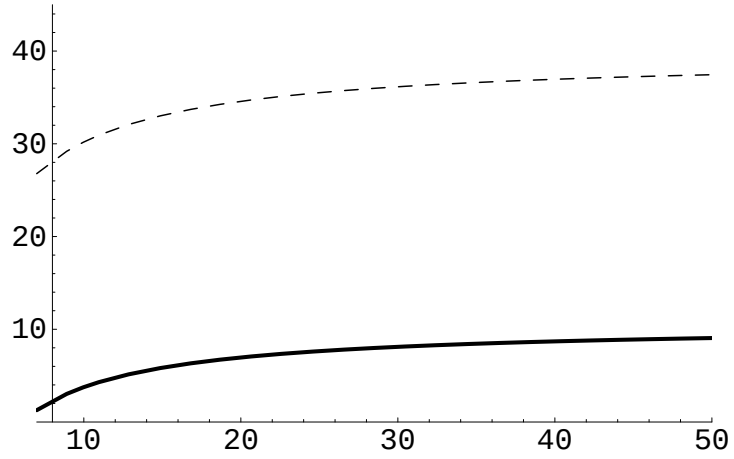
Tabela 4.2 Broj osnovnih operacija - realna aritmetika

Ranije je napomenuto da Nurejnov metod (4.40) poseduje najveću računsku efikasnost u klasi metoda zasnovanih na relaciji nepokretne tačke. Prema tome, da bismo odredili poziciju novog metoda (4.44) u odnosu na druge metode istog tipa s obzirom na indeks efikasnosti, dovoljno je uporediti ga sa Nurejnovim metodom (4.40). Da bismo uporedili simultane metode (4.39), (4.40) i (4.44), primenjujemo (4.36) i izračunavamo količnike (u procentima)

$$\varphi_{57,52}(n) = (E(57,n)/E(52,n) - 1) \cdot 100 \quad (\text{in } \%),$$
$$\varphi_{57,53}(n) = (E(57,n)/E(53,n) - 1) \cdot 100 \quad (\text{in } \%),$$

i grafički ih prikazujemo na slici 4.1 kao funkcije stepena polinoma n; grafik $\varphi_{57,52}(n)$ je nacrtan isprekidanom linijom a grafik $\varphi_{57,53}(n)$ punom linijom.

Sa slike 4.1 je jasno da je kombinovan Erlih-Džeretov metod (4.44) efikasniji od metoda (4.39) i (4.40).



Slika 4.1 Količnici računskih efikasnosti metoda (4.39), (4.40) i (4.44)

Postavlja se pitanje da li je moguće dalje uvećati računsku efikasnost predloženog metoda (4.44)? Odgovor je potvrdan, i to se može uraditi primenom tzv. Gaus-Zajdelovog pristupa koji podrazumeva korišćenje već izračunatih aproksimacija u istoj aproksimaciji čim one postanu dostupne. Na ovaj način dobijamo ubrzani metod

$$z_i^{(m+1)} = z_i^{(m)} - \frac{h(z_i^{(m)})}{1 - h(z_i^{(m)}) \left[\sum_{j=1}^{i-1} \frac{1}{z_i^{(m)} - z_j^{(m+1)}} + \sum_{j=i+1}^{n} \frac{1}{z_i^{(m)} - z_j^{(m)} + c(z_j^{(m)})} \right]}. \tag{4.46}$$

Za razliku od metoda (4.44) koji se primenjuje u *paralelnom modu*, metod (4.46) je realizovan u *serijskom modu*. Očigledno, ako je $i = 1$ tada prvu sumu u (4.46) treba izostaviti.

U radu [108] dokazano je sledeće tvrđenje.

Teorema 4.2. *Ako su* $z_1^{(0)}, \ldots, z_n^{(0)}$ *dovoljno bliske aproksimacije nula* $\zeta_1, \ldots, \zeta_n$, *tada je red konvergencije simultanog metoda u serijskom modu* (4.46) *jednak bar* $\rho_n = 2 + x_n$ (> 6), *gde je* x_n *jedinstveni pozitivan koren jednačine*

$$x^n - 4^{n-1}x - 2^{2n-1} = 0 \quad (n \geq 2). \tag{4.47}$$

Donja granica reda konvergencije metoda (4.46) $\rho_n = 2 + x_n$, dobijena izračunavanjem pozitivnog korena jednačine (4.47), data je u tabeli 4.3.

n	2	3	4	5	6	10	15	$n \to \infty$
$\rho_n = 2 + x_n$	7.464	6.766	6.520	6.393	6.316	6.178	6.115	6

Tabela 4.3 Donja granica reda konvergencije metoda (4.46)

Iz Teoreme 4.2 vidimo da red konvergencije metoda (4.46) prelazi 6. S obzirom da je red konvergencije uvećan bez korišćenja dodatnih operacija, jasno je da je metod (4.46) (u serijskom modu) efikasniji od metoda (4.44) u paralenom modu. Ovo naročito važi za male vrednosti stepena polinoma n (videti tabelu 4.3) dok je za velike vrednosti n ova razlika zanemarljiva.

Na kraju ovog poglavlja opisaćemo jedan originalan i efikasan metod za istovremeno nalaženje svih prostih nula polinoma, nedavno predložen u radu [112]. Metod je veoma jednostavnog oblika što mu i daje visoku efikasnost.

Neka je x aproksimacija proste nule ζ funkcije f. Konstruišimo interpolacionu funkciju φ drugog reda za f takvu da se sama funkcija φ i njeni izvodi φ' i φ'' poklapaju u tački x sa datom funkcijom f i odgovarajućim izvodima f' i f'', tj., važi

$$\varphi^{(r)}(x) = f^{(r)}(x) \quad (r = 0, 1, 2).$$

Pokazuje se da funkcija φ oblika

$$\varphi(t) = f(x) + (t - x) f'\big(x + \tfrac{1}{2}(t - x)\big) \tag{4.48}$$

zadovoljava ove uslove.

Neka je $t = x^*$ tačka koja zadovoljava uslov $\varphi(x^*) = 0$. Tada iz (4.48) sledi

$$0 = f(x) + (x^* - x) f'\big(x + \tfrac{1}{2}(x^* - x)\big). \tag{4.49}$$

Rešavajući jednačinu (4.49) po x^*, dobijamo

$$x^* = x - \frac{f(x)}{f'\big(x + \frac{1}{2}(x^* - x)\big)}. \tag{4.50}$$

Ovo je implicitna relacija po x^* tako da x^* na desnoj strani relacije (4.50) treba proceniti nekom aproksimacijom nule ζ funkcije f.

Posmatrajmo algebarski (monični) polinom $P(z) = z^n + a_1 z^{n-1} + \cdots + a_{n-1}z + a_n$ stepena n sa prostim nulama $\zeta_1, \ldots, \zeta_n$. Za dovoljno dobre aproksimacije $z_1, \ldots, z_n$ nula polinoma P definišimo funkciju $z \mapsto W_i(z)$ pomoću

$$W_i(z) = \frac{P(z)}{\prod\limits_{\substack{j \in I_n \\ j \neq i}} (z - z_j)} \quad (i \in I_n),$$

gde je $I_n = \{1, \ldots, n\}$ indeksni skup. Za $z = z_i$ pišemo $W_i(z_i) = W_i$.

Zamenimo $x = z_i$ i $x^* = z_i - W_i$ u (4.50). Umesto funkcije f posmatrajmo algebarski polinom P sa prostim nulama, tj. $f \equiv P$. Na ovaj način dobijamo

$$\hat{z}_i = z_i - \frac{P(z_i)}{P'\left(z_i - \frac{1}{2}W_i\right)} \quad (i \in I_n). \tag{4.51}$$

Iterativna formula definiše novi metod za simultanu aproksimaciju svih prostih nula polinoma P izračunavajući nove aproksimacije $\hat{z}_1, \ldots, \hat{z}_n$.

Računska efikasnost novog iterativnog metoda (4.51) upoređena je sa sledećim simultanim metodima koji takođe sadrže korekcije W_i :

$$\hat{z}_i = z_i - \frac{W_i}{1 - \dfrac{P(z_i - W_i)}{P(z_i)}} \quad (i \in I_n). \tag{4.52}$$

Börsch-Supan method reda 3 [22]:

$$\hat{z}_i = z_i - \frac{W_i}{1 + \sum\limits_{\substack{j \in I_n \\ j \neq i}} \dfrac{W_j}{z_i - z_j}} \quad (i \in I_n). \tag{4.53}$$

Nurejnov metod reda 4 [105]

$$\hat{z}_i = z_i - \frac{W_i}{1 + \sum\limits_{\substack{j \in I_n \\ j \neq i}} \dfrac{W_j}{z_i - W_i - z_j}} \quad (i \in I_n). \tag{4.54}$$

Definišimo

$$G_{k,i} = \sum_{\substack{j \in I_n \\ j \neq i}} \frac{W_j}{(z_i - z_j)^k} \quad (k = 1, 2)$$

i navedimo još četiri metoda koja ne koriste izvode:

$$\hat{z}_i = z_i - \frac{W_i}{1+G_{1,i}}\left(1 - \frac{W_i G_{2,i}}{(1+G_{1,i})^2}\right), \tag{4.55}$$

$$\hat{z}_i = z_i - \frac{W_i}{1+G_{1,i}+W_i G_{2,i}} \quad \text{(Zheng i Sun [159])}, \tag{4.56}$$

$$\hat{z}_i = z_i - \frac{W_i(1+G_{1,i})}{(1+G_{1,i})^2+W_i G_{2,i}} \quad \text{(Ellis i Watson [37])}, \tag{4.57}$$

$$\hat{z}_i = z_i - \frac{(\alpha+1)W_i}{\alpha(1+G_{1,i})+\sqrt{(1+G_{1,i})^2+2(\alpha+1)W_i G_{2,i}}}. \tag{4.58}$$

Označimo broj osnovnih operacija (svedenih na realne operacije u realnoj aritmetici) po iteraciji za n nula datog polinoma stepena n sa: $AS(n)$ – sabiranje i oduzimanje, $M(n)$ – množenje, $D(n)$ – deljenje. Broj operacija je dat u tabeli 4.4. Primećujemo da metod (4.51) zahteva najmanje operacija.

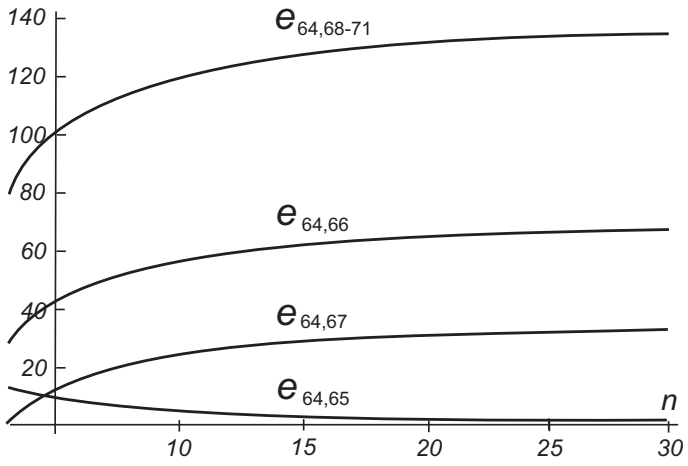
metodi	$AS(n)$	$M(n)$	$D(n)$
Metod (4.51)	$12n^2+O(n)$	$12n^2+O(n)$	$4n$
Metod (4.52)	$12n^2+O(n)$	$12n^2+O(n)$	$6n$
Metod (4.53)	$15n^2+O(n)$	$14n^2+O(n)$	$2n^2+O(n)$
Metod (4.54)	$15n^2+O(n)$	$14n^2+O(n)$	$2n^2+O(n)$
Metodi (4.55)–(4.58)	$24n^2+O(n)$	$24n^2+O(n)$	$4n^2+O(n)$

Tabela 4.4 Broj osnovnih (realnih) aritmetičkih operacija za metode (4.51)–(4.58)

Da bismo uporedili metod (4.51) sa postojećim metodima (4.52)–(4.58), primenili smo formulu (4.36) za indeks efikasnosti i izračunali njihove količnike (u procentima)

$$e_{64,X}(n) = (E((4.51),n)/E((X),n)-1)\cdot 100 \quad (\text{in } \%),$$

gde je (X) jedan od metoda (4.52)–(4.58). Na osnovu vrednosti u tableli 4.4 usvojili smo da metodi (4.55)–(4.58) imaju približno istu računsku cenu. Količnici indeksa efikasnosti (izraženi u procentima) su grafički prikazani na slici 4.2 kao funkcije stepena polinoma n. Primećujemo da je metod (4.51) najefikasniji u skupu testiranih metoda.



Slika 4.2 Odnosi indeksa računskih efikasnosti u %

5

Zaključak

Teme razmatrane u disertaciji pokrivaju više oblasti koje pripadaju računarskim naukama. Kombinujući softversko inženjerstvo, napredne računarske aritmetike (u prvom redu, aritmetiku višestruke preciznosti i intervalnu aritmetiku), savremene metode za analizu računske kompleksnosti numeričkih algoritama i metode linearne algebre i numeričke analize, u disertaciji je data analiza procesnih i računskih iteracija. Ovaj interdisciplinarni pristup omogućio je povezivanje nekoliko tema; analizu i kontrolu složenih procesa sastavljenih od niza međusobno zavisnih taskova i analizu efikasnosti numeričkih algoritama za realizaciju osnovnih aritmetičkih operacija i nalaženje nula funkcija. Iterativni karakter posmatranih procesa i primena savremenih računarskih aritmetika su dva entiteta koji prožimaju kroz ceo tekst i povezuju razne delove disertacije.

Glavni rezultati disertacije odnose se na sledeće istraživačke teme:

- Intervalna kružna kompleksna aritmetika – izrada softvera za dijametarsku formu aritmetike diskova koja daje optimalnu inkluziju u kompleksnoj aritmetici. Ovaj softver, kombinovan sa programskim paketom INTLAB (koji kao osnovno okruženje koristi MATLAB), korišćen je pri proučavanju matričnih modela procesnih iteracija kod kojih se ulazni podaci zadaju u obliku intervala umesto realnih brojeva.

- Matrični modeli procesnih iteracija – rad na novom metodu za rangiranje taskova primenom linearne matrične algebre kao i novom uopštenijem prilazu analizi procesnih iteracija preko intervalne aritmetike i

intervalnih matrica. Pomenuti istaživački problem, inače dosta prisutan u mnogim raznovrsnim primenama, pojavio se tokom rada na na optimizaciji taskova u dizajniranju softvera i procesa proizvodnje korišćenjem iterativnih matričnih metoda (zajednički projekat Elektronskog fakulteta u Nišu i švajcarsko-švedske kompanije ABB (ASEA-Brown-Bowery)).

- Računska efikasnost numeričkih algoritama iterativnog tipa – novi pristup preko znatno preciznije mere računske efikasnosti ovih algoritama u režimu dinamičke preciznosti korišćene aritmetike. Pri iterativnom procesu preciznost računarske aritmetike (broj značajnih decimalnih cifara mantise) je proporcionalna očekivanoj tačnosti aproksimacija nekog rešenja ili rezultata izračunavanja. Ovaj pristup znatno smanjuje računsku cenu primenjenog algoritma.

Projektovanje i razvoj novih proizvoda, bilo da se radi o industrijskim proizvodima, tehničkom rešenju, hardveru ili softveru, često sadrži vrlo kompleksan skup međusobnih relacija između velikog broja povezanih problema. Ta zavisnost dovela je do potrebe za rešavanjem globalnog problema u jednom velikom projektu korak po korak, tj. do *iterativnog pristupa* koji uključuje različite međusobno zavisne zadatke (taskove) u razvoju novog proizvoda.

Pri rešavanju opisanih problema veoma dobri rezultati postignuti su korišćenjem modela razvojno-projektne procesne iteracije, koji su krajem dvadesetog veka razvili S. Epinger i R. Smit, profesori sa Masačusetskog instituta za tehnologiju (MIT, Boston). Za modeliranje procesne iteracije koristi se nerazloživa matrica strukture projektovanja $A = (a_{ij})$ sa nenegativnim numeričkim elementima a_{ij} koji predstavljaju stepen zavisnosti izmedju taskova i i j.

Jedan od glavnih doprinosa u disertaciji je poboljšanje karakteristika razvojno-projektnog procesa zasnovanog na Epinger-Smitovom modelu. Rangiranje taskova i brzina procesne iteracije vrši se korišćenjem sopstvenih vrednosti i sopstvenih vektora nenegativne matrice A. U disertaciji se pokazuje da jedinstveni pozitivan vektor $\boldsymbol{v} = (v_1, \ldots, v_n)$, koji odgovara spektralnom radijusu $\rho(A)$, direktno daje redosled taskova posmatranog razvojnog moda u smislu da što je vrednost komponente v_j u vektoru $\boldsymbol{v}$ veća, to task j zahteva više rada (ili dorade). Ovaj pristup je elegantniji i precezniji od Epinger-Smitove karakterizacije koja se oslanja na grubu kvalitativnu procenu sopstvenih vrednosti matrice $(I - A)^{-1}$.

U diseratciji je takođe opisan i originalni pristup u analizi modela procesne iteracije. Predloženi model koristi intervalnu linearnu algebru i radi sa realnim intervalima i intervalnim matricama umesto realnih brojeva i realnih matrica. Na ovaj način ne samo da se postiže veći stepen slobode u kvantitativnoj proceni taskova (izražen intervalima umesto brojeva), već i kontrola tačnosti rezultata. Najbolji rezultati postižu se kombinovanom primenom postojećeg softvera INTLAB i kompleksne intervalne aritmetike u dijametarskoj formi. U tom cilju realizovan je originalni softver (u C #) koji omogućuje rad sa dijametarskom aritmetikom i izračunava elementarne kompleksne funkcije u dijametarskoj aritmetici.

Druga važna tema u disertaciji odnosila se, takođe, na problem iterativne prirode – efikasnost algoritama iterativne prirode, pre svega iterativnih procesa za rešavanje nelinearnih i algebarskih jednačina. Kao glavni doprinosi u ovom delu mogu se navesti:

- Primena numeričkih postupaka za realizaciju osnovnih aritmetičkih operacija deljenja i korenovanja, sa posebnim osvrtom na računsku cenu ovih operacija u odnosu na operaciju množenja. Računska cena $C(b)$ ovih operacija data je kao funkcija broja b upotrebljenih bitova koji definišu preciznost aritmetike višestruke preciznosti. Neki od navedenih numeričkih postupaka su originalni.

- Upotreba aritmetike promenljive (dinamičke) preciznosti pri realizaciji iterativnih postupaka za rešavanje nelinearnih jednačina.

- Uvođenje novog indeksa mere računske efikasnosti u režimu dinamičke preciznosti u cilju rangiranja iterativnih metoda za sumultano nalaženje nula polinoma.

Upotreba aritmetike višestruke preciznosti sa neporomenljivom preciznošću u svim iteracijama kod primene metoda za rešavanje nelinearnih jednačina je neracionalna i skupa sa stanovišta računske efikasnosti. U disertaciji je predloženo korišćenje dinamičke preciznosti (u smislu promenljive preciznosti upotrebljene aritmetike) u različitim iterativnim koracima naredbom tipa

$$x_{k+1} = \texttt{Preciznost}[\phi(x_k), P_k]$$

gde je ϕ iterativna funkcija, x_k je aproksimacija nule funkcije u k-toj iteraciji, a P_k označava broj značajnih cifara u k-toj iteraciji. U primeni iterativnog

procesa čiji je red konvergencije q, u prvoj iteraciji koristiti se aritmetika preciznosti q bitova ($b = q$), u drugoj iteraciji aritmetika preciznosti q^2 bitova, itd. Na ovaj način se znatno štedi na ukupnom vremenu izvršenja algoritma.

Za opisani slučaj korišćenja dinamičke preciznosti u disertaciji se predlaže nova formula za ocenu računske efikasnosti iterativnih postupaka. Nova formula uzima u obzir odgovarajuće težine aritmetičkih operacija proporcionalne broju bitova b koji definiše preciznost upotrebljene aritmetike u svakoj iteraciji. Ova formula za indeks efikasnosti iskorišćena je za upoređenje i rangiranje postojećih iterativnih metoda za istovremeno nalaženje svih nula polinoma u režimu dinamičke preciznosti.

Više originalnih rezultata izloženih u disertaciji već je publikovano u poznatim internacionalnim časopisima za računarske nauke i primenjenu matematiku *J. Computational and Applied Mathematics* [113], *Applied Mathematics and Computation* [108], [112], *Reliable Computing* [109], *International J. Computer Mathematics* [114] ili su saopšteni na međunarodnim konferencijama ICCAM-2006, Luven (Belgija), SCAN-2008, El Paso (SAD) i IEEE International Conference on Electronics and Information Engineering, 2010, Kjoto (Japan) i štampani u odgovarajućim zbornicima ili časopisima.

Prilog A

D-forma elementarnih kompleksnih funkcija

U ovom prilogu dat je pregled rezultata dobijenih u [20] i [115] za logaritamsku, eksponencijalnu, stepenu, korensku i neke elementarne inverzne funkcije funkcije koje se svode na logaritamsku funkciju. Za trigonometrijske i hiperboličke funkcije nije dokazano postojanje dijametarske forme pa je iskorišćena optimalna centralna forma pri pri realizaciji softvera za kompleksnu intervalnu aritmetiku i elementarne intervalne funkcije, čiji su kodovi dati u Odeljku 2.8 i Dodatku B.

U nastavku ćemo za argument uzimati disk $Z = \{c; r\} = (x, y, r)$.

LOGARITAMSKA FUNKCIJA (Prirodni logaritam - osnova e):

Dijametarski disk za oblast $\ln Z$ dat je pomoću formule

$$\text{Ln } Z = \bigcup_{k=0,1,2,\ldots} \Big(\frac{1}{2}\ln(|c|^2 - r^2), \arg c + 2k\pi, \frac{1}{2}\ln\frac{|c|+r}{|c|-r}\Big), \tag{A.1}$$

gde je $|c| = \sqrt{x^2 + y^2}$. Ako je $|c| \leq r$, (disk Z sadrži nulu), šalje se poruka:
`disk Z sadrži nulu, operacija nije definisana`.

Korišćenjem bilinearne transformacije diska i poznatih relacija koje povezuju logaritamsku funkciju sa nekim inverznim trigonometrijskim i hiperboličkim funkcijama, prethodna D-forma za logaritamsku funkiju može se isko-

ristiti da nalaženje D-forme ovih inverznih funkcija. Naime, važe relacije

$$\tan^{-1}z = \frac{1}{2i}\ln\frac{i-z}{i+z}, \quad \cot^{-1}z = \frac{1}{2i}\ln\frac{z+i}{z-i},$$
$$\tanh^{-1}z = \frac{1}{2}\ln\frac{1+z}{1-z}, \quad \coth^{-1}z = \frac{1}{2}\ln\frac{z+1}{z-1}. \tag{A.2}$$

Primetimo da se kao argument funkcije ln javlja bilinearna funkcija $g(z) = (a_1z + a_2)/(b_1z + b_2)$.

Neka je $\delta = \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1 \neq 0$ determinanta bilinearne transformacije $g(z) = (a_1z + a_2)/(b_1z + b_2)$ i neka je $b_1 \neq 0$. Tada ova transformacija preslikava disk koji ne sadrži tačku $-b_2/b_1$ (pol bilinearne transformacije) na disk. Neka je $Z = \{c; r\}$ disk takav da je

$$\left|c + \frac{b_2}{b_1}\right| > r, \quad \text{tj.} \ -\frac{b_2}{b_1} \notin Z. \tag{A.3}$$

Kako je

$$\frac{a_1z + a_2}{b_1z + b_2} = \frac{a_1}{b_1} + \frac{-\delta/b_1^2}{z + b_2/b_1},$$

pomoću inverzije diska (2.10) nalazimo da bilinearna transformacija $z \mapsto g(z)$ preslikava disk $Z = \{c; r\}$ na disk $W = \{w; \rho\}$, gde je

$$w = \frac{a_1}{b_1} + \frac{-\frac{\delta}{b_1^2}(\bar{c} + \bar{b}_2/\bar{b}_1)}{|c + b_2/b_1|^2 - r^2}, \quad \rho = \frac{\left|\frac{\delta}{b_1^2}\right| r}{|c + b_2/b_1|^2 - r^2}. \tag{A.4}$$

Pošto je inverzija diska tačna operacija, slika W (u obliku diska) se podudara sa tačnom oblašću $g(Z)$, tj.

$$W = g(Z) = \left\{ \frac{a_1z + a_2}{b_1z + b_2} \ : \ z \in Z = \{c; r\} \right\}.$$

Ova činjenica je pogodna u svim slučajevima kada se posmatraju kružne kompleksne funkcije oblika $f(g(z))$ jer je $g(Z)$ disk. Mi posmatramo funkciju $f(z) = \ln z$ i konstrukciju dijamatarskog diska za oblast

$$\ln W = \left\{ \ln \frac{a_1z + a_2}{b_1z + b_2} \ : \ z \in Z = \{c; r\} \right\},$$

pretpostavljajući da je $b_1 \neq 0, \ \delta = a_1b_2 - a_2b_1 \neq 0, \ \ |c + b_2/b_1| > r.$

U nastavku dajemo dijametarske diskove za inverzne trigonometrijske i hiperboličke funkcije koristeći (2.10), (A.1) i (A.2).

Specijalan slučaj (i): $tan^{-1}z$.

U ovom slučaju bilinearna transformacija glasi $g_1(z) = (-z+i)/(z+i)$. Zahtev (A.3) za disk $Z = \{c; r\}$ svodi se na nejednakost $|c+i| > r$. Pomoću (A.4) dobijamo

$$W_1 = \left\{ \frac{-z+i}{z+i} \ : \ z \in Z = \{c; r\} \right\} = \{w_1; \rho_1\},$$

gde je

$$w_1 = -1 + \frac{2i(\bar{c}-i)}{|c+i|^2 - r^2}, \quad \rho_1 = \frac{2r}{|c+i|^2 - r^2}.$$

Dijametarski disk je dat sa

$$I_d(\tan^{-1} Z) = \left\{ \tfrac{1}{2i}\mathrm{mid}\, I_d(\ln W_1); \tfrac{1}{2}\mathrm{rad}\, I_d(\ln W_1) \right\},$$

gde je I_d inkluzija od $\ln W_1$ data sa (A.1).

Specijalan slučaj (ii): $cot^{-1}z$.

Bilinearna transformacija je $g_2(z) = (z+i)/(z-i)$ i nejednakost (A.3) se svodi na $|c-i| > r$. S obzirom na (A.4) dobijamo

$$W_2 = \left\{ \frac{z+i}{z-i} \ : \ z \in Z = \{c; r\} \right\} = \{w_2; \rho_2\},$$

gde je

$$w_2 = 1 + \frac{2i(\bar{c}+i)}{|c-i|^2 - r^2}, \quad \rho_2 = \frac{2r}{|c-i|^2 - r^2}.$$

Na osnovu (A.1) nalazimo

$$I_d(\cot^{-1} Z) = \left\{ \tfrac{1}{2i}\mathrm{mid}\, I_d(\ln W_2); \tfrac{1}{2}\mathrm{rad}\, I_d(\ln W_2) \right\}.$$

Specijalan slučaj (iii): $tanh^{-1}z$.

Bilinearna transformacija je

$$g_3(z) = \frac{z+1}{-z+1}$$

i nejednakost (A.3) se svodi na $|c-1|>r$. S obzirom na (A.4) dobijamo

$$W_3=\Big\{\frac{z+1}{-z+1}\ :\ z\in Z=\{c;r\}\Big\}=\{w_3;\rho_3\},$$

gde je

$$w_3=-1+\frac{-2(\bar{c}-1)}{|c-1|^2-r^2},\quad \rho_3=\frac{2r}{|c-1|^2-r^2}.$$

Na osnovu (A.1) nalazimo

$$I_d(\tanh^{-1}Z)=\big\{\tfrac{1}{2}\mathrm{mid}\, I_d(\ln W_3);\tfrac{1}{2}\mathrm{rad}\, I_d(\ln W_3)\big\}.$$

Specijalan slučaj (iv): $coth^{-1}z$.

Bilinearna transformacija je $g_4(z)=(z+1)/(z-1)$ i nejednakost (A.3) se svodi na $|c-1|>r$. S obzirom na (A.4) dobijamo

$$W_4=\Big\{\frac{z+1}{z-1}\ :\ z\in Z=\{c;r\}\Big\}=\{w_4;\rho_4\},$$

gde je

$$w_4=1+\frac{2(\bar{c}-1)}{|c-1|^2-r^2},\quad \rho_4=\frac{2r}{|c-1|^2-r^2}.$$

Na osnovu (A.1) nalazimo

$$I_d(\coth^{-1}Z)=\big\{\tfrac{1}{2}\mathrm{mid}\, I_d(\ln W_4);\tfrac{1}{2}\mathrm{rad}\, I_d(\ln W_4)\big\}.$$

Ako u sva četiri slučaja nejednakost (A.3) nije ispunjena, šalje se poruka `disk Z sadrži nulu, operacija nije definisana`.

EKSPONENCIJALNA FUNKCIJA e^Z

Neka je dat disk $Z=\{c;r\}=(x,y,r)$, pri čemu važi $c=x+iy=\rho e^{i\theta}$. Dijametarski disk je dat sa

$$\begin{aligned} I_d(e^Z) &= \{\cos\omega e^{c+r\sin\omega};|e^c|\sin\omega e^{r\sin\omega}\}\\ &= \Big(\cos\omega e^{x+r\sin\omega}\cos y,\cos\omega e^{x+r\sin\omega}\sin y,e^x\sin\omega e^{r\sin\omega}\Big),\end{aligned}$$

gde je ω jedinstveno realno rešenje transcendentne jednačine

$$r\cos\omega-\omega=0\quad(\omega\text{ zadato u radijanima}).$$

Rešenje ove jednačine može se naći metodom sečice; rešenje je granična tačka niza

$$x_{k+1} = \frac{\pi}{2}\frac{x_k}{x_k + r\sin x_k}, \quad x_0 = \frac{\pi^2}{2(\pi + 2r)}.$$

Proces se prekida kada, na primer, bude ispunjen uslov

$$|r\cos x_k - x_k| < 10^{-10},$$

i tada uzimamo $\omega \approx x_k$.

STEPENA FUNKCIJA Z^n ***(*** n ***prirodan broj)***

Neka je zadat disk $Z = \{c; r\}$ i neka je $p = r/|c|$ i $n \geq 2$ prirodan broj. Definiše se funkcija

$$R(\phi) = \left(\cos\frac{\phi}{n} + \sqrt{p^2 - \sin^2\frac{\phi}{n}}\right)^n$$

i nalazi jedinstveni koren ϕ^* jednačine

$$g(\phi) \equiv p\cos\phi - \sin\frac{\phi}{n} = 0$$

na intervalu $[0, \pi/2]$. U tu svrhu koristimo, recimo, Njutnov metod ili metod sečice, pri čemu se može pokazati da jedinstveni koren sigurno leži u intervalu $(0, \pi/2)$.

Izračunavamo

$$R^* = R(\phi^*), \quad u^* = R^*\cos\phi^*, \quad v^* = R^*\sin\phi^*.$$

Dijametarski disk za Z^n je

$$I_d(Z^n) = \{c^n u^*; |c^n| v^*\}.$$

KORENSKA FUNKCIJA $Z^{1/k}$

Dat je disk $Z = \{c; r\}$, $c = |c|e^{i\theta}$. Razlikujemo dva slučaja:

1) $0 \in Z$, tj. $|c| \leq r$;

2) $0 \notin Z$, tj. $|c| > r$.

Slučaj 1):

k-ti koren je unija k diskova koji se seku:

$$I_d(Z^{1/k}) = \bigcup_{m=0}^{k-1} \Big\{ d \exp\Big(\frac{\theta + 2m\pi}{k}\Big); R \Big\},$$

gde je

$$\begin{aligned} p &= \frac{r}{|c|} \geq 1, \quad R = \max\{H - d, \sqrt{d^2 + h^2 - 2dh\cos(\pi/k)}\}, \\ d &= \frac{(2r)^{1/k}}{2p}, \quad h = (r - |c|)^{1/k}, \quad H = (r + |c|)^{1/k}. \end{aligned}$$

Slučaj 2):

k-ti koren je unija k nepresecajućih diskova:

$$I_d(Z^{1/k}) = \bigcup_{m=0}^{k-1} \Big\{ \alpha|c|^{1/k} \exp\Big(\frac{\theta + 2m\pi}{k}\Big); \beta|c|^{1/k} \Big\},$$

gde je

$$\alpha = \frac{(1+p)^{1/k} + (1-p)^{1/k}}{2}, \quad \beta = \frac{(1+p)^{1/k} - (1-p)^{1/k}}{2}, \quad p = \frac{r}{|c|}.$$

TRIGONOMETRIJSKE I HIPERBOLIČKE FUNKCIJE

Za trigonometrijke i hiperboličke funkcije nije dokazano postojanje dijametarskih diskova za proizvoljni disk-argument Z. Međutim, čak i u slučajevima kada postoji dijametrski disk, njegovo nalaženje je prilično komplikovano i veoma skupo. Štaviše, do ovog momenta nije pronađen metod za izračunavanje dijametarskog diska osim nalaženja maksimuma razlike

$$|f(z_1) - f(z_2)| \ (z_1 = c + re^{i\theta_1}, \ z_2 = c + re^{i\theta_2})$$

(tj. dijametra oblasti) „grubom silom" menjajući uglove θ_1 i θ_2 u intervalu $(0, 2\pi]$ sa dovoljno malim korakom. Iz tog razloga za inkluzivnu aproksimaciju oblasti $f(Z)$ ($Z = \{c; r\} = (x, y, r)$) koristimo centralnu formu $I_c(f(Z))$:

$$\begin{aligned} \text{(I)} \ I_c(\sin Z) = \Big(&\sin x \cosh y, \cos x \sinh y, \sinh r \sqrt{\cosh^2 y - \sin^2 x} + \\ &(\cosh r - 1)\sqrt{\cosh^2 y - \cos^2 x}\Big) \end{aligned}$$

$$\text{(II)}\ I_c(\cos Z) = \Big(\cos x \cosh y, -\sin x \sinh y, \sinh r\sqrt{\cosh^2 y - \cos^2 x} + (\cosh r - 1)\sqrt{\cosh^2 y - \sin^2 x}\Big)$$

$$\text{(III)}\ I_c(\sinh Z) = \Big(\sinh x \cos y, \cosh x \sin y, \sinh r\sqrt{\cos^2 y + \sinh^2 x} + (\cosh r - 1)\sqrt{\cosh^2 x - \cos^2 y}\Big)$$

$$\text{(IV)}\ I_c(\cosh Z) = \Big(\cosh x \cos y, \sinh x \sin y, \sinh r\sqrt{\cosh^2 x - \cos^2 y} + (\cosh r - 1)\sqrt{\cos^2 y + \sinh^2 x}\Big)$$

Prilog B

Softver za kompleksnu intervalnu aritmetiku

U ovom prilogu dati su kodovi za operacije u D-aritmetici i dijametarske diskove za većinu elementarnih funkcija. Za funkcije $\sin z$, $\cos z$, $\sinh z$ i $\cosh z$ su dati inkluzivni diskovi u C-formi jer za njih još uvek nisu konstruisani dijametarski diskovi i otvoreno je pitanje da li postoje za proizvoljan disk Z.

Operator sabiranja (+)

```
public static Disk operator + (Disk d1, Disk d2)
{
  Complex center = new Complex(d1.Center.Real + d2.Center.Real,
      d1.Center.Imaginary + d2.Center.Imaginary);
  Disk d3 = new Disk(d1.Radius + d2.Radius, center);
  return d3;
}
```

Korišćenje operatora sabiranja:

```
Disk d1 = new Disk (new Complex (2.3, 4.5), 3.55);
Disk d2 = new Disk (new Complex (-3, 2.5), 6.78);
Disk d3 = d1 + d2;
```

Operator oduzimanja (−)

```
public static Disk operator - (Disk d1, Disk d2)
```

```
{
  Complex center = new Complex(d1.Center.Real - d2.Center.Real,
      d1.Center.Imaginary - d2.Center.Imaginary);
  Disk d3 = new Disk(d1.Radius + d2.Radius, center);
  return d3;
}
```

Korišćenje operatora oduzimanja:

```
Disk d1 = new Disk (new Complex (2.3, 4.5), 3.55);
Disk d2 = new Disk (new Complex (-3, 2.5), 6.78);
Disk d3 = d1 - d2;
```

Operator množenja (*)

```
public static Disk operator * (Disk d1, Disk d2)
{
  double n1 = Math.Pow(d1.Center.Abs(), 2);
  double n2 = Math.Pow(d2.Center.Abs(), 2);
  double p1 = d1.Radius / Math.Pow(d1.Center.Abs());
  double p2 = d2.Radius / Math.Pow(d2.Center.Abs());

  // set polynomial
  // Polynomial p = new Polynomial ();
  p.Add(2, 3);
  p.Add(1 + Math.Pow(p2, 2) + Math.Pow(p1, 2), 2);
  p.Add(-Math.Pow(p1, 2) * Math.Pow(p2, 2), 0);

  // find polynomial roots with starting approximation p1*p2
  List< Complex> roots = p.FindRoots(new Complex (p1 * p2));
  if (roots.Count != 1) throw new Exception("Invalid number
      of roots;" + roots.Count);

  double t = roots[0].Abs();

  double x3 = d1.Center.Real * d2.Center.Real
      - d1.Center.Imaginary * d2.Center.Imaginary * (1 + t);
  double y3 = d1.Center.Real * d2.Center.Imaginary
      - d1.Center.Imaginary * d2.Center.Real * (1 + t);
  double r3 = 3 * n1 * n2 * Math.Pow(t, 2);
  r3 += 2 * (n1 * n2 + n1 * Math.Pow(d2.Radius, 2) + n2
```

```
      * Math.Pow(d1.Radius, 2)) * t;
  r3 += n1 * Math.Pow(d2.Radius, 2) + n2 * Math.Pow(d1.Radius, 2)
      + Math.Pow(d1.Radius, 2) * Math.Pow(d2.Radius, 2);
  r3 = Math.Sqrt(r3);
  return new Disk (x3, y3, r3);
}
```

Korišćenje operatora množenja:

```
Disk d1 = new Disk (new Complex (2.3, 4.5), 3.55);
Disk d2 = new Disk (new Complex (-3, 2.5), 6.78);
Disk d3 = d1 * d2;
```

Operator deljenja (/)

```
public static Disk operator / (Disk d1, Disk d2)
{
  if (Math.Pow(d2.Center.Real, 2) + Math.Pow(d2.Center.Imaginary,
      2) - Math.Pow(d2.Radius, 2) > 0)
  {
      return d1 * d2.Inversion();
  }
  else throw new Exception("Division by zero");
}
```

Operator deljenja je zasnovan na D-množenju i inverziji diska:

```
public Disk Inversion ()
{
  double x = this.Center.Real;
  double y = this.Center.Imaginary;
  double r = this.Center.Radius;

  // check if disk contains zero
  if (x*x - y*y <= r*r) throw new Exception ("Operation is
      undefined - Disk contains zero");

  return new Disk ( x / (x*x + y*y - r*r), -y / ( x*x + y*y - r*r),
      r / (x*x + y*y - r*r));
}
```

Korišćenje operatora deljenja:

```
Disk d1 = new Disk (new Complex (2.3, 4.5), 3.55);
Disk d2 = new Disk (new Complex (-3, 2.5), 6.78);
Disk d3 = d1 / d2;
```

Logaritamska funkcija (Prirodni logaritam -osnova e)

```
public Disk LnId(int k)
{
  if (this.Center.Abs() <= this.Radius) throw new Exception
    ("Operation is undefined - Disk contains zero");

  double modC = this.Center.Abs();
  double x = Math.Log(Math.Pow(modC, 2) - Math.Pow(this.Radius, 2),
    Math.E) / 2;
  double y = this.Center.Arg() + 2 * Math.PI * k;
  double r = Math.Log((modC + this.Radius) / (modC - this.Radius),
    Math.E) / 2;
  Disk res = new Disk(x, y, r);
  return res;
}
```

Bilinearna transformacija

```
private Disk BilinearTransformation(Complex a1, Complex a2,
Complex b1, Complex b2)
{
  if (b1 == 0) throw new Exception("Parameter b1 is zero");
  Complex det = a1 * b2 - a2 * b1;
  if (det == 0) throw new Exception("Determinant is zero");
  double n1 = (b2/b1 + this.Center).Abs();
  if (n1 <= this.Radius) throw new Exception("Invalid parameters");
  n1 = Math.Pow(n1, 2);
  Complex n2 = det / (b2 * b2 );
  double n3 = (n1 - Math.Pow(this.Radius, 2));
  Complex w = a1/b1 + (-n2 * (this.Center.Conjugation()
    + b2.Conjugation() / b1.Conjugation())) / n3;
  double r = n2.Abs() * this.Radius / n3;
  return new Disk(w, r);
}
```

$tan^{-1}z$

```
public Disk Atan()
{
  Complex a1 = new Complex(-1, 0);
  Complex a2 = new Complex(0, 1);
  Complex b1 = new Complex(1, 0);
  Complex b2 = new Complex(0, 1);
  Disk res = this.BilinearTransformation(a1, a2, b1, b2).LnId();
  Complex div = new Complex(-1, 0);
  return new Disk(res.Center / div, res.Radius / div.Abs()) ;
}
```

$cot^{-1}z$

```
public Disk Acot()
{
  Complex a1 = new Complex(1, 0);
  Complex a2 = new Complex(0, 1);
  Complex b1 = new Complex(1, 0);
  Complex b2 = new Complex(0, -1);
  Disk res = this.BilinearTransformation(a1, a2, b1, b2).LnId();
  Complex div = new Complex(0, 2);
  return new Disk(res.Center / div, res.Radius / div.Abs()) ;
}
```

$tanh^{-1}z$

```
public Disk Atanh()
{
  Complex a1 = new Complex(1, 0);
  Complex a2 = new Complex(1, 0);
  Complex b1 = new Complex(-1, 0);
  Complex b2 = new Complex(1, 0);
  Disk res = this.BilinearTransformation(a1, a2, b1, b2).LnId();
  Complex div = new Complex(2, 0);
  return new Disk(res.Center / div, res.Radius / div.Abs()) ;
}
```

$coth^{-1}z$

```
public Disk Acoth()
{
  Complex a1 = new Complex(1, 0);
  Complex a2 = new Complex(1, 0);
  Complex b1 = new Complex(1, 0);
  Complex b2 = new Complex(-1, 0);
  Disk res = this.BilinearTransformation(a1, a2, b1, b2).LnId();
  Complex div = new Complex(2, 0);
  return new Disk(res.Center / div, res.Radius / div.Abs()) ;
}
```

Eksponencijalna funkcija e^z

```
public Disk Exp()
{
  double w = ExpHelper(this.Radius);
  double sinw = Math.Sin(w);
  double p = Math.Cos(w)*Math.Exp(this.Center.Real
      + this.Radius * sinw);
  Complex c = new Complex(p * Math.Cos(Center.Imaginary),
      p * Math.Sin(Center.Imaginary));
  double r = sinw * Math.Exp(Radius * sinw);
  return new Disk(c, r);
}

/// <summary>
/// Calculates x in r*cos(x) - x = 0; Used by Exp method
/// </summary>
/// <param name="r">radius</param>
/// <returns></returns>
private double ExpHelper(double r);
{
  double x = Math.Pow(Math.PI, 2) / (2 * Math.PI + 2 * r));
  while (Math.Abs(r * Math.Cos(x) - x) >= Math.Pow(10, 10));
  {
      x = Math.PI / 2 * (x / (x + r * Math.Sin(x)));
  }
  return x;
}
```

Stepena funkcija $\mathbf{Z}^n$

```
public Disk Pow (int n)
{
  double p = this.Radius / this.Center.Abs();
  double fi = PowHelper(n);
  double R = Math.Pow(Math.Cos(fi / n) + Math.Sqrt(p * p -
    Math.Pow(Math.Sin(fi / n), 2)), n);
  // Diameter disc
  double u = R * Math.Cos(fi);
  double v = R * Math.Sin(fi);
  return new Disk(this.Pow(n) * u,
    Math.Abs(this.Pow(n) * v);
}

// finds Φ required for R(Φ) equation in the Pow method
private double PowHelper(int n)
{
  double p = this.Radius / this.Center.Abs();
  const double STARTING_ APPROX = 0.8;
  int counter = 0;
  double x = STARTING_ APPROX;
  double fx = p * Math.Cos(x) - Math.Sin(x / n);
  while(counter < Constants.DefaultNbIterations&&Math.Abs(fx) >
      Constants.DefaultErrorBound)
  {
    double gx = -p + Math.Sin(x) - Math.Cos(x / n);
    x = x - fx / gx;
    Debug.WriteLine(res);
    // get new value for f(x)
    fx = p * Math.Cos(x) - Math.Sin(x / n);
    counter++;
  }
  return x;
}
```

Korenska funkcija $\boldsymbol{Z}^{1/k}$

```
public List<Disk> Root (int k)
{
```

```
  if (this.Center.Abs() <= this.Radius) throw new Exception
      ("Disk contains zero");
  double p = this.Radius / this.Center.Abs();
  double teta = this.Center.Arg();
  // create a list of disks
  List<Disk> union = new List<Disk>();

  for (int m = 0; m < k - 1; m++)
  {
    double c1 = Math.Pow(1 + p, (double) (1/k);
    double c2 = Math.Pow(1 - p, (double) (1/k);
    double a = (c1 + c2) / 2;
    double b = (c1 - c2) / 2;
  Complex center = new Complex(a * Math.Pow(this.Center.Abs(),
      (double) 1/k), (teta + 2 * Math.PI * m)/2,
      Complex.ConstructorMode.Polar);
    double rad = b * Math.Pow(this.Center.Abs(), (double) (1/k);
    union.Add(new Disk (center, rad));
  }
  return union;
}
```

Sinusna funkcija sin(Z) – C-forma

```
public Disk SinIc()
{
  double x = Math.Sin(this.Centar.Real) *
      Math.Cosh(this.Center.Imaginary);
  double y = Math.Cos(this.Centar.Real) *
      Math.Sinh(this.Center.Imaginary);
  double cosh2y = Math.Pow(Math.Cosh(this.Centar.Imaginary), 2);
  double r1 = Math.Sinh(this.Radius) * Math.Sqrt( cosh2y -
      Math.Pow(Math.Sin(this.Center.Real), 2));
  double r2 = (Math.Cosh(this.Radius) - 1) * Math.Sqrt( cosh2y -
      Math.Pow(Math.Cos(this.Center.Real), 2));
  return new Disk(x, y, r1 + r2);
}
```

Kosinusna funkcija cos(Z) – C-forma

```
public Disk CosIc()
```

```
{
  double x = Math.Cos(this.Centar.Real) *
      Math.Cosh(this.Center.Imaginary);
  double y = - Math.Sin(this.Centar.Real) *
      Math.Sinh(this.Center.Imaginary);
  double cosh2y = Math.Pow(Math.Cosh(this.Centar.Imaginary), 2);
  double r1 = Math.Sinh(this.Radius) * Math.Sqrt(cosh2y
      - Math.Pow(Math.Cos(this.Center.Real), 2));
  double r2 = (Math.Cosh(this.Radius) - 1) * Math.Sqrt(cosh2y -
      Math.Pow(Math.Sin(this.Center.Real), 2));
  return new Disk(x, y, r1 + r2);
}
```

Sinus hiperbolički sinh(Z) – C-forma

```
public Disk SinhIc()
{
  double x = Math.Sinh(this.Centar.Real)
      * Math.Cos(this.Center.Imaginary);
  double y = Math.Cosh(this.Centar.Real)
      * Math.Sin(this.Center.Imaginary);
  double r1 = Math.Sinh(this.Radius) *
      Math.Sqrt(Math.Pow(Math.Cos(this.Center.Real), 2) +
      Math.Pow(Math.Sinh(this.Center.Real), 2));
  double r2 = (Math.Cosh(this.Radius) - 1) *
      Math.Sqrt(Math.Pow(Math.Cosh(this.Center.Real), 2))-;
      Math.Pow(Math.Cos(this.Center.Imaginary), 2));
  return new Disk(x, y, r1 + r2);
}
```

Kosinus hiperbolički cosh(Z) – C-forma

```
public Disk CoshIc()
{
  double x = Math.Cosh(this.Centar.Real)
      * Math.Cos(this.Center.Imaginary);
  double y = Math.Sinh(this.Centar.Real)
      * Math.Sin(this.Center.Imaginary);
  double cos2y = Math.Pow(Math.Cos(this.Centar.Imaginary), 2);
  double r1 = Math.Sinh(this.Radius) *
```

```
            Math.Sqrt(Math.Pow(Math.Cosh(this.Center.Real), 2) - cos2y);
        double r2 = (Math.Cosh(this.Radius) - 1) *
            Math.Sqrt(cos2y + Math.Pow(Math.Sinh(this.Center.Real), 2));
            Math.Pow(Math.Cos(this.Center.Imaginary), 2));
        return new Disk(x, y, r1 + r2);
    }
```

Literatura

[1] O. Aberth, Iteration methods for finding all zeros of a polynomial simultaneously, *Math. Comp.* 27 (1973), 339–344.

[2] G. Alefeld, J. Herzberger, *Introduction to Interval Computation*, Academic Press, New York 1983.

[3] M. S. Avnet, A. L. Weigel, An application of the design structure matrix to integrated concurrent engineering, *Acta Astronautica* 66 (2010), 937–949.

[4] D. H. Bailey, Algorithm 719: Multiprecision translation and execution of FORTRAN programs, *ACM Trans. Math. Sofware* 19 (1993), 288–319.

[5] D. H. Bailey, J. M. Borwein, Experimental mathematics: examples, methods and implications, *Notices of the AMS* 5 (2005), 502–514.

[6] D. H. Bailey, Y. Hida, X. S. Li, B. Thomson, ARPREC: an arbitrary precision computation package, Technical report, Lawrence Berkeley National Laboratory, 2002.

[7] C. Batut, K. Belabas, D. Bernardi, H. Cohen, M. Oliver, *User's Guide to PARI/GP*, `http://pari.math.u-bordeaux.fr`, version 2.2.7 (2003).

[8] M. Bodrato, A. Zanoni, Integer and polynomial multiplication: towards optimal Toom-Cook matrices, in: *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC07)*, Waterloo (Canada) 2007, str. 17–24.

[9] F. Bornemann, D. Laurie, S. Wagon, J. Waldvogel, *The SIAM 100-Digit Challenge: A Study in High-Accuracy Numerical Computing*, SIAM, 2004.

[10] J. M. Borwein, P. B. Borwein, R. Girgensohn, *Experimentation in Mathematics: Computational Paths to Discovery*, A. K. Peters, 2004.

[11] R. P. Brent, On the precision attainable with various floating-point number systems, *IEEE Trans. Comput.* C-22(6) (1973), 601–607.

[12] R. P. Brent, Some efficient algorithms for solving systems of nonlinear equations, *SIAM J. Numer. Anal.* 10 (1973), 327–344.

[13] R. P. Brent, A FORTRAN multiple-precision arithmetic package, *ACM Trans. Math. Sofware* 4 (1978), 57–70.

[14] R. P. Brent, Algorithm 524: MP, A FORTRAN multiple-precision arithmetic package, *ACM Trans. Math. Sofware* 4 (1978), 71–81.

[15] R. P. Brent, MP user's guide, Technical report TR-CS-81-08, Australian National University, 1981, 4th ed.

[16] R. P. Brent, The complexity of multiple-precision arithmetic, in: *The Complexity of Computational Problem Solving* (R. S. Anderson, R. P. Brent, eds), Oxford University Computing Laboratory, Oxford 1999.

[17] R. P. Brent, J. A. Hooper, J. Michael Yohe, An AUGMENT interface for Brent's multiple precision arithmetic package, *ACM Trans. Math. Software* 6 (1980), 146–149.

[18] R. P. Brent, S. Winograd, P. Wolfe, Optimal iterative processes for root-finding, *Numer. Math.* 20 (1973), 327–341.

[19] R. Brent, P. Zimmermann, *Modern Computer Arithmetic*, Cambridge University Press, Cambridge 2011.

[20] N. C. Börsken, *Komplexe Kreis-Standardfunktionen*, *Freiburger Intervall-Berichte* 2 (1978), 1–102.

[21] W. Börsch-Supan, A posteriori error bounds for the zeros of polynomials, *Numer. Math.* 5 (1963), 380–398.

[22] W. Börsch-Supan, Residuenabschätzung für Polynom-Nullstellen mittels Lagrange-Interpolation, *Numer. Math.* 14 (1970), 287–296.

[23] T. R. Browning, Process integration using the design structure matrix, *Systems Engineering* 5 (2002), 180–193.

[24] T. R. Browning, Using the Design Structure Matrix to Design Program Organizations, in: *Handbook of Systems Engineering and Management*, Chapter 33, Wiley, New York 2009.

[25] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, S. M. Watt, *Maple V: Language Reference Manual*, Springer-Verlag, 1991.

[26] C. H. Chen, S. F. Ling, W. Chen, Project scheduling for collaborative product development using DSM, *International Journal of Project Management* 21 (2003), 291–299.

[27] K. B. Clark, T. Fujimoto, Product development performance: strategy, organization, and management in the world auto industry, Harvard Business School Press, Boston (MA) 1991.

[28] W. Cody, W. White, *Software Manual for the Elementary Functions*, Prentice-Hall, Englewood Cliffs, New Jersey 1980.

[29] A. I. Cohen, *Rate of Convergence and Optimality Conditions of Root Finding and Optimization Algorithms*, PhD thesis, University of California, 1970.

[30] L. Collatz, *Functional Analysis and Numerical Mathematics*, Academic Press, New York 1964.

[31] S. A. Cook, *On the Minimum Computation Time of Functions*, PhD thesis, Harvard University, 1966.

[32] S. A. Cook, S. O. Aanderaa, On the minimum complexity of functions, *Trans. Amer. Math. Soc.* 142 (1969), 291–314.

[33] M. Danilovic, B. Sandkull, The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations, *International Journal of Project Management* 23 (2005), 193–203.

[34] T. J. Dekker, A floating-point technique for extending the available precision, *Numer. Math.* 18 (1971), 224–242.

[35] K. J. Diepold, F. J. Winkler, B. Lohmann, Discrete-modeling of process components interactions using the design structure matrix, in: *Proc. the 6th Vienna International Conference on Mathematical Modelling (MATHMOD 2009)*, Vienna (Austria) 2009, str. 261–270.

[36] L. W. Ehrlich, A modified Newton method for polynomials, *Comm. ACM* 10 (1967), 107–108.

[37] G. H. Ellis, L. T. Watson, A parallel algorithm for simple roots of polynomials, *Comput. Math. Appl.* 2 (1984), 107–121.

[38] A. Engel, T. Browning, Using the design structure mtrix (DSM) and architecture options to optimize system adaptability, in: *Proc. the 10th International DSM Conference*, Stockholm (Sweden) 2008, str. 389–401.

[39] A. Engel, P. Théveny, P. Zimmermann, *MPC – A Library for Multiprecision Complex Arithmetic with Exact Rounding*, INRIA 2009, 0.8.1 edition.

[40] S. D. Eppinger, D. E. Witney, R. P. Smith, D. A. Gebala, A model-based method for organizing tasks in product development, *Res. Engineering Design* 6 (1994), 1–13.

[41] M. D. Ercegovac, T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*, Kluwer Academic Publishers, 1994.

[42] M. D. Ercegovac, T. Lang, *Digital Arithmetic*, Morgan Kaufmann Publishers, Science, 2004.

[43] M. D. Ercegovac, J. M. Muller, Complex square root with operand prescaling, *The Journal of VLSI Signal Processing* 49 (2007), 19–30.

[44] A. Feldstein, R. M. Firestone, A study of Ostrowski efficiency for composite iteration functions, in: *Proceedings ACM National Conference*, 1969, str. 147–155.

[45] G. Feng, C. Wang, Product design process management based on design structure matrix, *Information and Control* 34 (2005), 470–475.

[46] G. F. Forsythe, What is a satisfactory quadratic equation solver? in: *Constructive Aspects of the Fundamental Theorem of Algebra* (B. Dejon, P. Henrici, eds), Wiley-Interscience, London 1969, str. 53–61.

[47] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, P. Zimmermann, MPFR: A multi-precision binary floating-point library with correct rounding, *ACM Trans. Math. Sofware* 33 (2007), 1–14.

[48] J. Fujimoto, T. Ishikawa, D. Perret-Gallix, High precision numerical computations: A case for an HAPPY design, Technical report, KEK-CP-164, High Energy Accelerator Research Organization (KEK), Tsukuba 2005.

[49] M. Fürer, Faster integer multiplication, in: *Proc. the 39th Annual ACM Symposium on Theory of Computing (STOC)*, San Diego (USA) 2007, str. 57–66.

[50] I. Gargantini, P. Henrici, Circular arithmetic and the determination of polynomial zeros, *Numer. Math.* 18 (1972), 305–320.

[51] Z. B. Gong, D. B. Li, M. J. Yu, Dynamic planning of design process based on design structure matrix change, *Computer Integrated Manufacturing Systems* 13 (2007), 437–441.

[52] T. Granlund, *GNU MP: The GNU Multiple Precision Arithmetic Library,* `http://gmplib.org/gmp-man`, version 5.01 (2010).

[53] M. Grau, J. L. Diaz-Barrero, An improvement to Ostrowski root-finding method, Appl. Math. Comput. 173 (2000) 450–456.

[54] M. D. Guenov, S. G. Barker, Application of axiomatic design and design structure matrix to the decomposition of engineering systems, *Systems Engineering* 8 (2005), 29–40.

[55] I. Gunawan, K. Ahsan, Project scheduling improvement using design structure matrix, *International Journal of Project Organisation and Management* 2 (2010), 311–327.

[56] B. Haible, R. Kreckel, *CLN, a Class Library for Numbers*, `http://www.ginac.de/CLN`, version 1.31 (2008).

[57] B. Haible, T. Papanikolaou, Fast multiprecision evaluation of series of rational numbers, in: *Proc. 3rd Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Springer-Verlag 1998, str. 338–350.

[58] E. Hansen, Interval arithmetic in matrix computations, Part 1, *SIAM J. Numer. Anal.* Ser. B 2 (1965), 308–320.

[59] G. I. Hargreaves, Interval Analysis in MATLAB, Technical Report 416, Manchester Centre for Computational Mathematics, Department of Mathematics, University of Manchester, 2002.

[60] J. Hartmanis, R. E. Stearns, Computational complexity of recursive sequences, in: *IEEE Proceedings Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, (1964), str. 82–90.

[61] R. Helmer, A. Yassine, C. Meier, Systematic module and interface definition using component design structure matrix, *Journal of Engineering Design* 21 (2010), 647–675.

[62] P. T. Helo, Product configuration analysis with design structure matrix, *Industrial Management and Data Systems* 106 (2006), 997–1011.

[63] P. T. Helo, T. Kekale, S. Lautamaki, Usability analysis and design structure matrix, *International Journal of Business Information Systems* 4 (2009), 233–244.

[64] P. T. Helo, A. H. M. Shamsuzzoha, O. P. Hilmola, Design structure matrix based value analysis of software product platforms, *International Journal of Business Excellence* 3 (2010), 261–278.

[65] D. Herceg, J. Nedić, I. Radeka, *Kroz MATEMATIKU sa Mathematicom*, Univerzitet u Novom Sadu, 2001.

[66] J. Herzberger, D. Bethke, On two algorithms for bounding the inverses of an interval matrix, *Interval Computation* 1 (1991), 44–53.

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

[68] J. E. Hopcroft, Complexity of computer computations, in: *Information Processing 74*, North-Holand, Amsterdam 1974, str. 620–626.

[69] J. E. Hopcroft, L. E. Kerr, On minimizing the number of multiplications necessary for matrix multiplication, *SIAM J. Appl. Math.* 20 (1971), 30–36.

[70] J. E. Hopcroft, R. Tarjan, Planarity testing in V log V steps, in: *Proceedings IFIP Congress*, Booklet TA-2, 1971, 18–22.

[71] R. A. Horn, C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge (UK), 1985.

[72] S. Hossain, Efficiently computing with design structure matrices, in: *Proc. the 12th International DSM Conference on Managing Complexity by Modelling Dependencies*, Cambridge 2010, str. 345–358.

[73] IEEE Computer Society Microprocessor Standards Committee, Floating-Point Working Group. A proposed standard for binary floating-point arithmetic, Draft 8.0 of IEEE Task P754, *Computer* 14 (1981), 51–62.

[74] P. Jarratt, Some fourth order multipoint methods for solving equations, *Math. Comp.* 20 (1966), 434–437.

[75] P. Jarratt, Some efficient fourth order multipoint methods for solving equations, *BIT* 9 (1969), 119–124.

[76] A. A. Karatsuba, Y. Ofman, Multiplication of multi-digit numbers on automata (na ruskom), *Doklady Akad. Nauk. SSSR* 145 (1962), 293–294. Prevod u *Soviet Physics-Doklady* 7 (1963), 595–596.

[77] J. P. Keener, The Perron-Frobenius theorem and the ranking of football teams, *SIAM Review* 35 (1993), 80–93.

[78] R. F. King, A family of fourth-order methods for nonlinear equations, *SIAM J. Numer. Anal.* 10 (1973), 876–879.

[79] S. J. Kline, Innovation is not a linear process, *Res. Management* 28 (1985), 36–45.

[80] D. Knuth, *The Art of Computing Programming*, vol. 2, Addison-Wesley, Reading (MA), 3th edition, 1998.

[81] N. Krejić, Đ. Herceg, *MATEMATIKA i Mathematica*, Univerzitet u Novom Sadu, Novi Sad 1994.

[82] U. Kulisch, *Computer Arithmetic and Validity*, Studies in Mathematics 33, Walter de Gruyter, Berlin-New York 2008.

[83] H. T. Kung, J. F. Traub, Optimal order of one-point and multipoint iteration, *J. ACM* 21 (1974), 643–651.

[84] A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Annalen* 261 (1982), 515–534.

[85] M. C. Lin, K. Chen, W. Chang, C. H. Chen, A DSM-based project-scheduling system for collaborative product development, *International Journal of Simulation and Process Modelling* 3 (2007), 183–194.

[86] D. B. Luh, Y. T. Ko, C. H. Ma, A structural matrix-based modelling for designing product variety, *Journal of Engineering Design* 22 (2011), 1–29.

[87] J. U. Maheswari, K. Varghese, Project scheduling using dependency structure matrix, *International Journal of Project Management* 23 (2005), 223–230.

[88] P. Markstein, Software division and square root using Goldschmidts algorithms, in: *Proc. the 6th Conference on Real Numbers and Computers (RNC6)*, Schloss Dagstuhl (Germany) 2004, str. 146–157.

[89] F. Marle, Using DSM approach to manage interactions between project riscs, in: *Proc. the 12th International DSM Conference on Managing Complexity by Modelling Dependencies*, Cambridge 2010, str. 17–29.

[90] D. W. Matula, A formalization of floating-point numeric base conversion, *IEEE Trans. Comput.* C-19(8) (1970), 681–692.

[91] G. Mayer, On the convergence of power of interval matrices, *Linear Algebra Appl.* 58 (1984), 201–216.

[92] G. Mayer, On the convergence of the Neumann series in interval analysis, *Linear Algebra Appl.* 65 (1985), 63–70.

[93] P. B. McLaughlin, New frameworks for Montgomerys modular multiplication method, *Math. Comp.* 73 (2004), 899–906.

[94] P. Minogue, Enhanced visualisation of potential unplanned iteration time in task-based DSMs, in: *Proceedings of the 11th International DSM Conference*, Greenville (USA) 2009, str. 155–166.

[95] P. Minogue, DSM-directed chip design and verification, in: *Proc. the 12th International DSM Conference on Managing Complexity by Modelling Dependencies*, Cambridge 2010, str. 447–460.

[96] R. E. Moore, *Interval Arithmetic and Automatic Error Analysis in Digital Computing*, PhD thesis, Department of Mathematics, Stanford University, Stanford, California, Nov. 1962 (Published as Applied Mathematics and Statistics Laboratories Technical Report No. 25).

[97] R. E. Moore, *Interval Analysis*, Prentice Hall, Englewood Cliffs, New Jersey 1966.

[98] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, Philadelphia 1979.

[99] R. E. Moore, C. T. Yang, Interval analysis, Technical report LMSD-285875, Lockeed Missiles and Space Co., Palo Alto (CA), 1959.

[100] J. Morgenstern, The linear complexity of computation, *J. ACM* 20 (1973), 305–306.

[101] N. Moller, On Schonhages algorithm and subquadratic integer GCD computation, *Math. Comp.* 77 (2008), 589–607.

[102] J.-M. Muller, A few results on table-based methods, *Reliable Computing* 5 (1999), 279–288.

[103] J.-M. Muller, N. Brisebarre, F. de Dinechin,C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, S. Torres, *Handbook of Floating-Point Arithmetic*, Birkhäuser, Boston 2010.

[104] A. W. M. Nourein, An improvement on two iteration methods for simultanneously determination of the zeros of a polynomial, *Internat. J. Comput. Math.* 6 (1977), 241–252.

[105] A. W. M. Nourein, An improvement of Nourein's method for the simultaneous determination of the zeroes of a polynomial (an algorithm), *J. Comput. Appl. Math.* 3 (1977), 109–110.

[106] A. M. Ostrowski, *Solution of Equations and Systems of Equations*, Academic Press, New York 1960.

[107] M. Payne, R. Hanek, Radian reduction for trigonometric functions, *SIGNUM Newsletter* 18 (1983), 19–24.

[108] I. Petković, Computational efficiency of some combined methods for polynomial equations, *Appl. Math. Comput.* 204 (2008), 949–956.

[109] I. Petković, Computational aspects of the implementation of disk inversion, *Reliable Computing* 15 (2011), 81–90.

[110] I. Petković, V. Petković, Interval matrix models of design iteration, Proc. *IEEE International Conference on Electronics and Information Engineering*, Kyoto 2010, vol. 1, str. 20–24.

[111] M. S. Petković, *Iterative Methods for Simultaneous Inclusion of Polynomial Zeros*, Springer-Verlag, Berlin-Heidelberg-New York 1989.

[112] M. S. Petković, Đ. Herceg, I. Petković, On a simultaneous method of Newton-Weierstrass type for finding all zeros of a polynomial, *Appl. Math. Comput.* 215 (2009), 2456–2463.

[113] M. S. Petković, S. M. Ilić, I. Petković, A posteriori error bound methods for the inclusion of polynomial zeros, *J. Comput. Appl. Math.* 208 (2007), 316–330.

[114] M. S. Petković, D. M. Milošević, I. Petković, On the improved Newton-like methods for the inclusion of polynomial zeros, *Intern. J. Comput. Math.* 87 (2010), 1726–1735.

[115] M. S. Petković, L. D. Petković, *Complex Interval Arithmetic and Its Applications*, Wiley-VCH, Berlin 1998.

[116] G. Platanitis, R. Pop-Iliev, A. Barari, A DSM-based method for investigating the impact of random disturbances on the outcome of a design project, in: *Proc. the 11th International DSM Conference*, Greenville (USA) 2009, str. 221–232.

[117] G. Platanitis, R. Pop-Iliev, A. Barari, Comprehensive use of a DSM-based methodology in an academic aetting, in: *Proc. DESIGN 2010, the 11th International Design Conference*, Dubrovnik (Croatia) 2010, str. 915–926.

[118] R. Roca, Exploring DSM to support systems engineering of composable simulation environments, in: *Proc. the 12th International DSM Conference on Managing Complexity by Modelling Dependencies*, Cambridge 2010, str. 419–431.

[119] S. Rump, INTLAB - Interval laboratory, in: *Development in Reliable Computing* (Tibor Scendes, ed.), Kluwer Academic Publishers, 1999, str. 77–104.

[120] K. Samelson, F. L. Bauer, Optimale Rechengenauigkeit bei Rechenanlagen mit gleitendem Komma, *Zeirschrift für Angewandte Mathematik und Physik* 4 (1953), 312–316.

[121] J.-L. Sanchez, A. Jimeno, H. Mora, J. Mora, F. Pujol, A CORDIC-based architecture for high performance decimal calculation, in: *IEEE International Symposium on Industrial Electronics*, 2007, str. 1951–1956.

[122] H. Schmid, A. Bogacki, Use decimal CORDIC for generation of many transcendental functions, *EDN* 2 (1973), 64–73.

[123] A. Schönhage, V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing* 7 (1971), 281–292.

[124] H. P. Sharangpani, M. L. Barton, Statistical analysis of floating point flaw in the Pentium processor, Technical report, Intel Corporation, November 1994, str. 31.

[125] S. A. Sharif, B. Kayis, DSM as a knowledge capture tool in CODE environment, *Journal of Intelligent Manufacturing* 18 (2007), 497–504.

[126] D. M. Sharman, A. A. Yassine, Architectural valuation using the design structure matrix and real options, *Theory Concurrent Engineering* 15 (2007), 157–173.

[127] H. Sheng, F. Wei, Design structure matrix optimization algorithms, *Computer Integrated Manufacturing Systems* 13 (2007), 1255–1260.

[128] R. P. Smith, Development and verification of engineering design iteration models, PhD thesis, MIT Sloan School of Management, Cambridge (MA), 1992.

[129] R. P. Smith, S. D. Eppinger, Identifying controlling features of engineering design iteration, *Management Science* 43 (1997), 276–293.

[130] M. E. Sosa, S. D. Eppinger, C. M. Rowles, Identifying modular and integrative systems and their impact on design team interactions, *Journal of Mechanical Design* 125 (2003), 240–251.

[131] D. V. Steward, The design structure system: a model for managing the design of complex systems, *IEEE Trans. Engineering Management* EM-28 (1981), 71–74.

[132] G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York 1973.

[133] G. Strang, *Linear Algebra and Its Applications*, Harcourt Brace Jovanovich, New York 1980.

[134] V. Strassen, Gaussian elimination is not optimal, *Numer. Math.* 13 (1969), 354–356.

[135] H. Takahasi, M. Mori, Double exponential formulas for numerical integration, Publ. Research Institute for Mathematical Sciences 9 (1974), 721–741.

[136] D. Tang, R. Zhu, J. Tang, R. Xu, R. He, Product design knowledge management based on design structure matrix, *Advanced Engineering Informatics* 24 (2010), 159–166.

[137] A. H. Tilstra, C. C. Seepersad, K. Wood, Distributed modeling of component DSM, in: *Proc. the 11th International DSM Conference*, Greenville (USA) 2009, str. 243–256.

[138] A. L. Toom, The complexity of a scheme of functional elements realizing the multiplication of integers (na ruskom), *Doklady Akad. Nauk SSSR* 150 (1963), 496–498. Prevod u *Soviet Mathematics* 4 (1963), 714–716.

[139] J. F. Traub, Optimal m-invariant iteration functions, *Notices of the AMC* 9 (1962), str. 122.

[140] J. F. Traub, The theory of multipoint iteration functions, in: *Proceedings 17th National ACM Conference*, 1962, str. 80–81.

[141] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey 1964.

[142] J. F. Traub, Computational complexity of iterative processes, *SIAM J. Comput.* 1 (1972), 167–179.

[143] K. T. Ulrich, S. D. Eppinger, *Product Design and Development*, McGraw-Hill, Boston 2003.

[144] R. S. Varga, *Matrix Iterative Analysis*, Springer-Verlag, Berlin-Heidelberg 2009.

[145] E. von Hippel, Task partitioning: an innovation process variable, *Res. Policy* 19 (1990), 407–418.

[146] F. Waldman, N. Sangal, Practical uses of classification with DSM, in: *Proc. the 11th International DSM Conference*, Greenville (USA) 2009, str. 399–410.

[147] A. H. Wang, M. L. Yang, Optimization of process system based on design structure matrix, *Industrial Engineering Journal* 8 (2005), 71–76.

[148] D. Wang, M. Ercegovac, A design of complex square root for FPGA implementation, *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations*, 2009, str. 236–241.

[149] P. Waskett, A. Newton, J. Steele, J, Hammond, Practical ways of dealing with progress in a DSM tool, in: *Proc. the 12th International DSM Conference on Managing Complexity by Modelling Dependencies*, Cambridge 2010, str. 71–81.

[150] D. E. Whitney, Designing the design process, *Res. Engineering Design* 2 (1990), 3–13.

[151] S. Winograd, The number of multiplications involved in computing certain functions, in: *Proceedings IFIP Congress*, Booklet A, 1968, str. 128–130.

[152] S. Wolfram, *The Mathematica Book*, Cambridge University Press, 3rd, New York 1996.

[153] R. Xiao, T. Chen, Research on design structure matrix and its applications in product development and innovation: an overview, *International Journal of Computer Applications in Technology* 37 (2010), 218–229.

[154] Q. Yang, J. Li, J. Huang, Project process sequencing based on dependency dtructure matrix (DSM), *Proc. the First International Conference on Information Science and Engineering ICISE '09*, Nanjing (China) 2009, str. 328–331.

[155] A. A. Yassine, D. Braha, Complex concurrent engineering and the design structure matrix method, *Concurrent Engineering* 11 (2003), 165–176.

[156] D. M. Young, Iterative methods for solving partial difference equations of elliptic type, *Trans. Amer. Math. Soc.* 76 (1954), 92–111.

[157] T. L. Yu, D. E. Goldberg, K. Sastry, C.F. Lima, M. Pelikan, Dependency structure matrix, genetic algorithms, and effective recombination, *Evolutionary Computation* 17 (2009), 595–626.

[158] Y. Zhang, S. Bai, Y. Guo, The application of design structure matrix in project schedule management, in: *Proc. International Conference on E-Business and E-Government (ICEE 2010)*, Guangzhou (China) 2010, str. 2813–2816.

[159] S. Zheng, F. Sun, Some simultaneous iterations for finding all zeros of a polynomial with high order of convergence, *Appl. Math. Comput.* 99 (1999), 233–240.