Numeričke metode

prof. dr Aleksandar Cvetković prof. dr Miodrag Spalević

Univerzitet u Beogradu
Mašinski fakultet
Katedra za Matematiku
Beograd, 2013. godine

Dr Aleksandar S. Cvetković, vanredni profesor
Mašinskog fakulteta Univerziteta u Beogradu

Dr Miodrag M. Spalević, redovni profesor
Mašinskog fakulteta Univerziteta u Beogradu

Numeričke metode

Osnovni udžbenik

I izdanje

Recenzenti:
Dr Gradimir V. Milovanović, redovni profesor
Matematički institut, SANU
Dr Boško Jovanović, redovni profesor
Matematičkog fakulteta Univerziteta u Beogradu

Izdavač:
Univerzitet u Beogradu, Mašinski fakultet
Kraljice Marije 16, 11120 Beograd 35, Srbija

Za izdavača:
Dekan dr Milorad Milovančević, redovni profesor
Mašinskog fakulteta Univerziteta u Beogradu

Glavni i odgovorni urednik:
Dr Aleksandar Obradović, redovni profesor
Mašinskog fakulteta Univerziteta u Beogradu

Odobreno za štampu:
Odlukom Dekana Mašinskog fakulteta 232/13 od 21.03.2013. godine

Beograd 2013. godine
Tiraž: 400 primeraka

Štampa: Planeta print
YU ISBN 99-9999-999-9 ????

Angelini i Nikoli

Sadržaj

1 Elementi teorije grešaka **1**

1.1 Aproksimacija brojeva . . . 1

1.1.1 Rastojanje na skupu realnih brojeva . . . 1

1.1.2 Apsolutna i relativna greška . . . 2

1.2 Reprezentacija brojeva . . . 3

1.2.1 Brojevi u fiksnom i pokretnom zarezu . . . 3

1.2.2 Zaokruživanje brojeva i značajne cifre . . . 5

1.2.3 Značajne cifre u proizvoljnoj brojnoj osnovi . . . 11

1.2.4 Zapisivanje brojeva, naučna i inženjerska notacija . . . 12

1.2.5 Format realnog broja IEEE-754, tipovi single i double . . . 13

1.3 Računske operacije sa približnim brojevima . . . 21

1.3.1 Aritmetičke operacije sa približnim brojevima . . . 21

1.3.2 Implementacija aritmetičkih operacija u računskim mašinama . . . 25

1.3.3 Efekti izračunavanja u aritmetici konačne dužine . . . 28

1.3.4 Izračunavanje funkcija sa približnim vrednostima argumenata . . . 30

1.3.5 Uslovljenost izračunavanja . . . 33

2 Sistemi linearnih jednačina **36**

2.1 Norme vektora i matrica . . . 37

2.1.1 Norme vektora . . . 37

2.1.2 Norme matrica . . . 40

2.1.3 Izračunavanje normi vektora i matrica u Matlabu . . . 44

2.2 Direktni metodi . . . 46

2.2.1 Trougaoni sistemi linearnih jednačina . . . 46

2.2.2 Gaussova eliminacija . . . 49

2.2.3 LU faktorizacija . . . 54

2.2.4 Izbor glavnog elementa . . . 57

2.2.5 Matlab implementacija Gaussove eliminacije . . . 62

2.3 Uslovljenost sistema linearnih jednačina . . . 64

2.4 Iterativni metodi . . . 69

2.4.1 Primer iterativnog metoda . . . 69

2.4.2 Konstrukcija iterativnih metoda . . . 71

2.4.3 Jacobijev iterativni metod . . . 73

2.4.4 Gauss-Seidelov iterativni metod . . . 75

2.4.5 Kriterijum za prekidanje iteracija . . . 75

2.4.6 Implementacija Jacobijevog i Gauss-Seidelovog metoda 76

3 Nelinearne jednačine **79**
3.1 Nelinearne jednačine jedne promenljive 79
3.1.1 Metod polovljenja intervala (bisekcije) 79
3.1.2 Uslov za izlazak iz while petlje 82
3.1.3 Metod proste iteracije 85
3.1.4 Red konvergencije metoda proste iteracije 88
3.1.5 Newtonov metod 94
3.1.6 Newtonov metod za višestruke nule 96
3.1.7 Divergencija Newtonovog metoda 98
3.1.8 Funkcija fzero 98
3.2 Sistemi nelinearnih jednačina 102
3.2.1 Metod Newton-Kantoroviča 102
3.2.2 Faktorizacija polinoma 105
3.2.3 Gradijentni metod 107
3.2.4 Funkcija fsolve 110
3.2.5 Metod proste iteracije 117

4 Interpolacija **125**
4.1 Lagrangeova interpolacija 126
4.1.1 Numerička konstrukcija Lagrangeovog interpolanta 128
4.1.2 Greška interpolacije 133
4.1.3 Izbor interpolacionih čvorova 136
4.1.4 O rešenju problema interpolacije matričnim metodima 140
4.2 Newtonov interpolacioni polinom 143
4.2.1 Numerička konstrukcija Newtonovog interpolacionog polinoma 147
4.3 Interpolacija u ekvidistantnim tačkama 151
4.3.1 Numerička konstrukcija prvog i drugog Newtonovog interpolanta 156
4.3.2 Račun konačnih razlika 159
4.3.3 Stirlingov i Besselov interpolacioni polinom 161
4.3.4 Numerička konstrukcija Stirlingovog i Besselovog interpolanta 164
4.4 Hermiteov interpolant 166
4.4.1 Newtonova forma Hermiteovog interpolanta 169
4.4.2 Greška Hermiteovog interpolanta 171
4.4.3 Hermiteov interpolant u Lagrangeovom obliku 171
4.4.4 Numerička konstrukcija Hermiteovog interpolanta 174
4.5 Splajn 176
4.5.1 Splajn stepena 1 reda m 177
4.5.2 Kubni splajn 179
4.6 Uopšteni problem interpolacije 185
4.6.1 Čebiševljev sistem 186
4.6.2 Trigonometrijski interpolant 189

5 Aproksimacija diskretnih skupova podataka **191**
5.1 Najmanji kvadrati . . . 191
5.1.1 Postavka problema . . . 191
5.1.2 Rešenje problema . . . 192
5.1.3 Numerička konstrukcija . . . 194
5.1.4 Matlab implementacija . . . 197
5.1.5 Ekvidistantne tačke . . . 198
5.2 Najmanji kvadrati sa težinama . . . 203
5.3 Uopšteni problem najmanjih kvadrata . . . 206
5.3.1 Problemi sa trendom . . . 206
5.3.2 Uopšteni polinomi . . . 208
5.4 Linearna regresija i funkcija regress . . . 210

6 Numeričko diferenciranje i integracija **214**
6.1 Numeričko diferenciranje . . . 214
6.1.1 Numeričko diferenciranje upotrebom interpolacionog polinoma . . . 214
6.1.2 Formule za jednostrano diferenciranje . . . 216
6.1.3 Formule za dvostrano diferenciranje . . . 222
6.1.4 Uticaj tačnosti ulaznih podataka . . . 226
6.1.5 Izvodi višeg reda . . . 227
6.2 Numerička integracija . . . 235
6.2.1 Interpolacione formule . . . 238
6.2.2 Newton-Cotesove kvadraturne formule . . . 239
6.2.3 Newton-Cotesove formule sa pozitivnim težinama . . . 243
6.2.4 Uopštene kvadraturne formule . . . 245
6.2.5 Otvorene Newton-Cotesove formule . . . 248
6.2.6 Uopštene otvorene Newton-Cotesove formule . . . 250
6.2.7 Divergencija Newton-Cotesovih formula . . . 252
6.2.8 Uticaj tačnosti ulaznih podataka . . . 252
6.2.9 Problemi numeričke integracije . . . 253
6.2.10 Implementacija numeričke integracije u Matlabu . . . 254

7 Obične diferencijalne jednačine **261**
7.1 Cauchyev problem . . . 261
7.2 Eulerov metod . . . 262
7.2.1 Implicitni Eulerov metod . . . 264
7.2.2 Adaptivna integracija . . . 267
7.2.3 Apsolutna stabilnost . . . 268
7.3 Linearni višekoračni metodi . . . 270
7.3.1 Red metoda . . . 270
7.3.2 Konvergencija . . . 272
7.3.3 Konstrukcija upotrebom kvadraturnih formula . . . 275
7.3.4 Analiza greške . . . 277
7.3.5 Stabilnost . . . 278
7.3.6 Prediktor-korektor metodi . . . 281
7.3.7 Matlab implementacija Adams-Bashfort-Moultonovog metoda . . . 285
7.3.8 Metodi sa diferenciranjem unazad . . . 288

7.4 Metodi Runge-Kutta 290
7.4.1 Matlab implementacija eksplicitnih metoda Runge-Kutta 293
7.5 Sistemi običnih diferencijanih jednačina 294
7.5.1 Linearni višekoračni metodi 294
7.5.2 Metodi Runge-Kutta 298
7.5.3 Stif sistemi diferencijalnih jednačina 299
7.6 Uticaj tačnosti ulaznih podataka 302

Slike

1.1 Slika levo grafici funkcija z_x i $z_{x'}$ za $x = 1$ dok se x' menja od 10^{-10} do 10^{10}, slika desno ilustruje iste funkcije ali za $x' = 1$ i x koje se menja u opsegu 10^{-10} do 10^{10}. 9

1.2 Grafik funkcije $|a_k| = 50^k/k!$, $k = 1, \ldots, 100$, levo, i broj značajnih cifara u vrednostima $\sin((1/4 + 10^k)\pi)$, $k = 1, \ldots, 15$, desno. 30

2.1 Značajne cifre u rešenju sistema (2.4) slika levo, značajne cifre u rešenju sistema sa Hilbertovom matricom u funkciji reda matrice sa i bez izbora glavnog elementa slika desno. 68

3.1 Slika levo ilustruje koncept metoda bisekcije, slika desno predstavlja grafike funkcija $y = \cos x$ i $y = x$. 80

3.2 Slika levo ilustruje broj značajnih cifara u iteracijama u procesima $x_{k+1} = \cos x_k$, $x_0 = 0$ i $x_{k+1} = x_k(x_k^2 + 6)/(3x_k^2 + 2)$, $x_0 = 2$, slika desno ilustruje metod proste iteracije. 91

3.3 Slika levo daje geometrijsku interpretaciju Newtonovog metoda, slika desno grafici funkcija $y = 2((9 - x^2)/x)^{1/2}/3$ i $y = (16x^2 + x + 1)^{1/4}$. 96

4.1 Slika levo prikazuje grafike funkcije log i Lagrangeovog interpolanta u 5 tačaka, slika desno prikazuje značajne cifre interpolacionog polinoma funkcije $f(x) = .1$ konstruisanog u interpolacionim čvorovima $x_i = -1 + 2i/59$, $i = 0, \ldots, 59$. 130

4.2 Slika levo prikazuje grafik Lebesgueove funkcije konstruisane za interpolacione čvorove $x_i = -1 + 2i/59$, $i = 0, \ldots, 59$, slika desno prikazuje značajne cifre interpolanta za funkciju $f(x) = .1$ za ekvidisantne čvorove $x_i = -1 + 2i/59$, $i = 0, \ldots, 59$, ali sa gornjom granicom relativne greške u vrednostima funkcije 10^{-3}. 131

4.3 Slika levo prikazuje grafike funkcija $|\log x - P_4(x)|$ i $1/30 \cdot 1/160 \cdot |\omega(x)|$ za interpolacione čvorove $\{2.4, 3.7, 5.1, 7.4, 9.8\}$, slika desno prikazuje zavisnost faktora uslovljenosti Vandermondove matrice u funkciji reda. 135

4.4 Slika levo prikazuje grafik čvornog polinoma za interpolacione čvorove koji su zadati Čebiševljevim tačkama, ima ukupno 30 čvorova, slika desno prikazuje grafik čvornog polinoma za slučaj ekvidistantno izabranih interpolacionih čvorova, ukupno ima 30 čvorova. 137

4.5 Slika levo ilustruje broj značajnih cifara pri interpolaciji funkcije $\sin(17x)$ na intervalu $[-1, 1]$ u Čebiševljevim i ekvidistantnim interpolacionim čvorovima, u 30 interpolacionih čvorova, slika desno ilustruje broj značajnih cifara pri interpolaciji funkcije $\exp(17x)$ na intervalu $[-1, 1]$ u Čebiševljevim i ekvidistantnim interpolacionim čvorovima, u 100 interpolacionih čvorova. 138

4.6 Slika levo prikazuje grafik negativnog logaritma za osnovu deset Lebesgueove funkcije sa interpolacionim čvorovima u Čebiševljevim tačkama (4.4) i ekvidistantnim tačkama, sa 30 interpolacionih čvorova, slika desno prikazuje značajne cifre interpolacionih polinoma u Čebiševljevim tačkama (4.4) i ekvidistantnim tačkama funkcije sin, sa 120 interpolacionih čvorova. 140

4.7 Slika levo prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta u 60 Čebiševljevih tačaka funkcije log na intervalu $[2, 9]$, slika desno prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta u 110 Čebiševljevih tačaka funkcije log na intervalu $[2, 9]$. Oba grafika su dobijena sa vrednostima funkcije sa gornjom granicom relativne greške 10^{-3}. 148

4.8 Slika levo prikazuje grafik funkcije $\log_{10} |[x_0, \ldots, x_k; f]|$, $k = 0, \ldots, 59$, podeljenih razlika funkcije $f(x) = 1$ u ekvidistantnim tačkama na intervalu $[2, 9]$ pri čemu vrednosti funkcije imaju gornju granicu apsolutne greške 10^{-3}, slika desno ilustruje grafik iste funkcije ali za 110 ekvidistantnih interpolacionih čvorova zadatih na intervalu $[0, .2]$. 149

4.9 Slika levo prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta za 50 ekvidistantnih interpolacionih tačaka na intervalu $[2, 9]$ funkcije log sa ulaznim podacima mašinske preciznosti, slika desno prikazuje isti grafik ali sa ulaznim podacima koji imaju gornju granicu apsolutne greške 10^{-14} 151

4.10 Slika levo prikazuje značajne cifre Lagrangeovog, prvog i drugog Newtonovog interpolanta za 40 ekvidistantnih interpolacionih čvorova na intervalu $[2, 9]$ pri interpolaciji funkcije log, slika desno prikazuje isti interpolacioni problem ali sa 70 interpolacionih čvorova. 157

4.11 Slika levo ilustruje prostiranje greške u tabeli konačnih razlika za konstantnu funkciju, dok slika desno ilustruje isto ali za funkciju $\sin(5x)$, podaci u oba slučaja imaju gornju granicu apsolutne greške 10^{-5}. 158

4.12 Slika levo prikazuje broj značajnih cifara Lagrangeovog, prvog i drugog Newtonovog interpolanta za 40 ekvidistantnih interpolacionih čvorova na intervalu $[2, 9]$ pri interpolaciji funkcije log, slika desno prikazuje isti interpolacioni problem ali sa 70 interpolacionih čvorova. Podaci kojima je zadata funkcija log imaju gornju granicu apsolutne greške 10^{-5}. 158

4.13 Slika levo ilustruje mogućnost konstrukcije interpolanta koji ima lokalne ekstreme u interpolacionim čvorovima, slika desno ilustruje tri Hermiteova interpolanta sa izvodima u interpolacionim čvorovima određenim sa $f' = [0, 3, -3, 0]$, $f' = [0, 0, 0, 0]$ i $f' = [0, -3, 3, 0]$. 169

4.14 Slika levo prikazuje broj značajnih cifara Hermitevog interpolanta u Newtonovoj i Lagrangeovoj formi pri interpolaciji funkcije log na intervalu $[2, 9]$ sa 20 ekvidistantnih čvorova višestrukosti dva, slika desno ilustruje isto što i slika levo ali podaci imaju gornju granicu relativne greške 10^{-5}. 176

4.15 Slika levo ilustruje u kojoj meri algebarski interpolant može odstupati od grafika funkcije, slika desno ilustruje glatkost splajna stepena 1 reda 5. 177

4.16 Slika levo ilustruje interpolacioni kubni splajn za različite vrednosti drugog izvoda u tačkama t_0 i t_m, slika desno ilustruje interpolacioni kubni splajn za razne vrednosti prvog izvoda u tačkama t_0 i t_m. 181

4.17 Slika levo ilustruje odnos periodičnog i not-a-knot interpolacionog kubnog splajna, slika desno ilustruje broj značajnih cifara interpolanta prilikom interpolacije interpolacionim kubnim splajnom not-a-knot funkcije log na intervalu $[2, 7]$ sa 100 i 200 čvorova. 184

5.1 Slika levo ilustruje zavisnost faktora uslovljenosti matrice V^TV u funkciji broja tačaka n za stepen polinoma m uzet kao parametar, slika desno ilustruje grafik polinoma rešenja najmanjih kvadrata dobijen za skup tačaka $x_i = -1 + 2i/100$, $y_i = 2 + x_i + .25e^{3|x_i|}\sigma_i$, $i = 0, \ldots, 100$, gde su σ_i uniformno raspodeljeni slučajni brojevi u intervalu $[-1, 1]$. 195
5.2 Slika levo ilustruje zavisnost odstupanja Δ_m u funkciji stepena polinoma m, slika desno ilustruje grafike polinoma dobijenih sa aritmetikom mantise 16 i 2000 dekadnih cifara, za $n = 400$ i $m = 390$. Na slici desno, zbog preglednosti, grafik polinoma konstruisan sa 2000 cifara mantise i podaci su prikazani translirani po y-osi za vrednost 10. 196
5.3 Slika levo ilustruje zavisnost odstupanja $\log_{10} \Delta_m$ u funkciji stepena polinoma m u izlazu funkcije polyfit, slika desno ilustruje odnos veličina Δ_m kod funkcija najmanjiKvadrati i polyfit za polinome do stepena 55. 197
5.4 Slika levo ilustruje zavisnost Δ_m u funkciji m, slika desno ilustruje odnos grafika polinoma aproksimacije i tačaka u ravni za $m = 165$. Na slici desno, zbog preglednosti, grafik polinoma koji se dobija metodom izloženom u ovoj sekciji i podaci su tranlirani po y-osi za vrednost 10. 201
5.5 Slika levo ilustruje zavisnost minimuma broja značajnih cifara u vrednostima $Q_{500}^{2000}(i)$, $i = 0, \ldots, 2000$, slika desno ilustruje zavisnost minimuma broja značajnih cifara u vrednostima $Q_{n/4}^{n}(i)$, $i = 0, \ldots, n$, i $Q_{n/4}^{n}(i)$, $i = 10, \ldots, n - 10$. 202
5.6 Slika levo ilustruje zavisnost Δ_m u funkciji m za 400 tačaka u ravni, slika desno ilustruje grafik funkcije Q_{230}^{418}. 203
5.7 Slika levo ilustruje uticaj težina na rešenje problema najmanjih kvadrata, slika desno ilustruje osetljivost polinoma na promenu tezina. 205
5.8 Slika levo ilustruje podatke sa trendom $f(x) = 8/x$, trend f i podatke bez trenda, slika desno ilustruje podatke sa i bez trenda, polinom koji aproksimira podatke sa trendom p_T, polinom koji aproksimira podatke bez trenda p i funkciju $f(p(\cdot))$ koja aproksimira podatke sa trendom. 206
5.9 Slika levo ilustruje odnos trigonometrijskog polinoma kao rešenja problema najmanjih kvadrata i podataka, slika desno ilustruje zavisnost faktora uslovljenosti matrice V^TV u funkciji broja tačaka n za stepen polinoma m uzet kao parametar. 210
5.10 Slika levo ilustruje odnos opservacija i regresione linije, slika desno ilustruje odnos opservacija, regresione linije i pogrešnih opservacija (outliera). 213

6.1 Slika levo prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$ sa 15 tačaka, slika desno prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$ sa 35 tačaka. 219
6.2 Slika levo ilustruje broj značajnih cifara u funkciji reda formule kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$, slika desno prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja iste funkcije u istoj tački. Podaci imaju gornju granicu apsolutne greške 10^{-5} . . . 227

6.3 Slika levo ilustruje grafik $\log_{10}|k!S^{\ell}_{k+\ell}/(k+\ell)!|$ u funkciji od ℓ za $k = 1, 2, 5, 10$, dok slika desno prikazuje zavisnost broja značajnih cifara u funkciji reda formule pri određivanju petog izvoda za vrednosti koraka $h = .1, .01, .001, .0001$. 229

6.4 Slika levo prikazuje grafik funkcije $\log_{10}|A^n_k|$ u funkciji od k za $n = 1, 2, 5, 10$, slika desno ilustruje zavisnost broja značajnih cifara u funkciji reda formule za vrednosti koraka $h = .1, .01, .001, .0001$, pri određivanju petog izvoda funkcije sin u tački $\pi/4$. 234

6.5 Slika levo prikazuje grafik funkcije $\log_{10}(|H_k|)$, $k = 0, \ldots, 100$, dok slika desno prikazuje grafik funkcije $\mathrm{sgn}(H_k)$, $k = 0, \ldots, 100$. 242

6.6 Slika levo prikazuje broj značajnih cifara u funkciji ω pri integraciji funkcije $\omega \sin(\omega x)$ na $[0, 1]$, slika desno prikazuje broj značajnih cifara u funkciji γ pri integraciji funkcije $|x|^{-\gamma}$ na intervalima $[0, 1]$ i $[-1, 1]$. 257

6.7 Slika levo daje zavisnost broja značajnih cifara u funkciji $2g$ pri integraciji funkcije $1/(2(2g)!)|\log|x||^{2g}$ na $[0, 1]$ i $[-1, 1]$, slika desno prikazuje broj značajnih cifara u funkciji p pri integraciji funkcija f_p i $p^{-1}e^{-x/p}$ na intervalu $[0, +\infty)$. 259

7.1 Slika levo ilustruje akumulaciju greške Eulerovog metoda, slika desno ilustruje konvergenciju Eulerovog metoda sa smanjenjem koraka. U oba slučaja $h = 0$ znači egzaktno rešenje. 263

7.2 Slika levo ilustruje akumulaciju greške i konvergenciju implicitnog Eulerovog metoda, slika desno ilustruje gubitak značajnih cifara sa porastom A pri rešavanju Cauchyevog problema (7.2), za fiksnu vrednost koraka $h = .01$. 266

7.3 Slika levo prikazuje grafik funkcije $10\exp(-25(x - 1.41)^2)$ i njenog izvoda, slika desno prikazuje grafike dobijene primenom eksplicitne Eulerove metode na Cauchyev problem (7.6), sa koracima $h = .1$, $h = .05$, $h = .001$, dok je četvrti grafik sa korakom $h = .05$ ali sa metodom koji startuje od tačke $x_0 = .6$. 268

7.4 Ilustracija apsolutne stabilnosti eksplicitnog Eulerovog metoda slika levo, i implicitnog Eulerovog metoda slika desno. 269

7.5 Slika desno ilustruje divergenciju metoda (7.10), slika desno ilustruje konvergenciju linearnog višekoračnog metoda drugog reda. 274

7.6 Slika levo ilustruje nule polinoma $\pi(\cdot; Ah)$ u funkciji Ah, slika desno ilustruje koncept apsolutne i relativne stabilnosti Simpsonovog metoda za rešavanje Cauchyevog problema. 280

7.7 Slika levo ilustruje oblasti stabilnosti familije Adams-Bashfortovih metoda, dok slika desno ilustruje oblasti stabilnosti Adams-Moultonovih metoda, pod metodom I Euler podrazumevamo implicitni Eulerov metod. 281

7.8 Slika levo ilustruje dejstvo opcije MaxStep pri integraciji periodičnih problema, slika desno dolazi do istih rezultata ali varira opciju RelTol 288

7.9 Slika levo predstavlja oblasti stabilnosti metoda sa diferenciranjem unazad, dok slika desno predstavlja raspodelu nula polinoma ρ metoda sa diferenciranjem unazad. . 289

7.10 Slika levo ilustruje upotrebu eksplicitnog Eulerovog metoda za rešavanje Cauchyevog problema $z_1' = z_1 + z_2$, $z_2' = z_1 - z_2$, $z_1(0) = 1$, $z_2(0) = 1$, sa $h = 0$ su označena tačna rešenja, slika desno ilustruje rešenja sistema jednačina matematičkog klatna. 296

7.11 Slika levo ilustruje broj značajnih cifara u rešenju sistema diferencijalnih jednačina (7.21), slika desno ilustruje grafik rešenja sistema jednačina (7.21), gde je vrednost brzine normalizovana sa 200. 301

7.12 Slika levo ilustruje gubitak značajnih cifara u numeričkom rešenju Cauchyevog problema usled slabe uslovljenosti početnim uslovom, slika desno ilustruje mogućnost detekcije slabo uslovljenih Cauchyevih problema. 303

Tabele

1.1 Značenje pojedinih bitova u jednostrukoj i dvostrukoj preciznosti 14

3.1 Moguće vrednosti izlaznog parametra exitflag funkcije fzero sa opisima. 99
3.2 Polja unutar strukture output izlaznog argumenta funkcije fzero. 100
3.3 Polja strukture options funkcije fzero. 101
3.4 Vrednosti izlaznog parametra exitflag i njihovo značenje. 112
3.5 Polja strukture output i njihov opis. 112
3.6 Polja strukture options i njihova značenja, prvi deo. 113
3.7 Polja strukture options i njihova značenja, drugi deo. 114

4.1 Vrednosti funkcije log u pojedinim tačkama. 134
4.2 Podaci za konstrukciju Hermiteovog interpolanta 167
4.3 Hermiteov interpolacioni problem za polinom H_3 168

6.1 Vrednost aproksimacije izvoda $(D_n^+ \sin)(\pi/4)$, apsolutne greške Δ_n, i teorijski očekivane greške, za korak podele $h = 10^{-7}$. 218
6.2 Vrednost intregrala I_n i apsolutna greška Δ_n dobijene primenom trapezne formule, Simpsonove formule, Simponovog pravila 3/8 i Booleovog pravila. 241
6.3 Vrednost T_N, apsolutne greške Δ_N, i teorijske ocene apsolutne greške $|R_{N+1}|$ prilikom aproksimacije integrala (6.8) 247
6.4 Aproksimacija vrednosti integrala M_N uopštenim pravilom srednje tačke, i apsolutna greška aproksimacije Δ_N, prilikom računanja integrala (6.9). 251
6.5 Divergencija Newton-Cotesovih formula pri integraciji Rungeove funkcije $f(x) = 1/(1 + x^2)$. Brojevi u zagradama predstavljaju eksponente broja deset. 252

7.1 Adamsova, Nystromova i Milne-Simpsonova familija metoda, koeficijenti linearnog višekoračnog metoda β_i, $i = 0, \ldots, k$, red metoda p, asimptotska konstanta greške C_{p+1} i donja granica intervala apsolutne stabilnosti γ. 276
7.2 Konvergentni metodi familije sa diferenciranjem unazad. 289
7.3 Butcherova tabela eksplicitnog metoda Runge-Kutta. 291
7.4 Butcherova tabela levo predstavlja jednoparametarski metod drugog reda, tabela u sredini predstavlja Euler-Cauchyev metod, dok tabela desno predstavlja poboljšani Euler-Cauchyev metod. 292
7.5 Butcherova tabela Heunovog metoda tabela levo, i tabela jednog metoda trećeg reda desno. 292

7.6 Butcherova tabela metoda četvrtog reda levo, tabela desno je metod u literaturi poznat kao Gillov. 292
7.7 Butcherova tabela za Bogacki-Shampine (2,3) par eksplicitnih Runge-Kutta metoda koji se koristi u funkciji ode23. 293
7.8 Butcherova tabela za Dormand-Prince (4,5) par eksplicitnih Runge-Kutta metoda koji se koristi u funkciji ode45. 293

Predgovor

Knjiga Numeričke metode je pre svega namenjena studentima treće godine osnovnih studija Mašinskog fakulteta Univerziteta u Beogradu. Međutim, mogu je koristiti svi koji žele da se upoznaju sa predmetom numeričkih metoda. Gotovo sve izložene metode su ili eksplicitno implementirane funkcijama u `Matlab`u ili je, u slučaju da `Matlab` poseduje funkcije koje implementiraju metode, prezentovana funkcionalnost `Matlab`a.

Knjiga se sastoji iz sedam glava. Prva glava se bavi teorijom grešaka i osnovama reprezentacija brojeva u računskim mašinama. Obrađen je format IEEE-754, reprezentacija brojeva u formatima jednostruke i dvostruke preciznosti i prostiranje greške kroz osnovne aritmetičke operacije, kao i prostiranje greške prilikom izračunavanja funkcija. Posebna pažnja posvećena je pojmu značajnih cifara, jer je izlaganje u ostalim glavama bazirano na njemu. Uveden je pojam slabe uslovljenosti izračunavanja i motivisan je primerima. Efekti koji se javljaju prilikom izračunavanja u aritmetici dvostruke preciznosti su ilustrovani primerima.

Predmet izučavanja druge glave su sistemi linearnih jednačina. Uvedene su norme vektora i matrica. Obrađeni su direktni metodi za rešavanje sistema linearnih jednačina i prezentovana je funkcionalnost `Matlab`a. Pažnja je posvećena funkcijama `lu` i `linsolve`. Gaussova eliminacija i LU faktorizacije su ilustrovane nizom primera i motivisana je potreba za izborom glavnog elementa. Pojam uslovljenosti sistema linearnih jednačina je uveden detaljno i demonstrirane su osobine slabo uslovljenih sistema. Na kraju su date iterativne metode za rešavanje sistema linearnih jednačina.

Treća glava se bavi problemom rešavanjem nelinearnih jednačina. Prikazani su metodi polovljenja intervala, proste iteracije i Newtonov metod. Motivacija pojma reda konvergencije je uvedena kroz broj iteracija potrebnih za rešavanje jednačine sa zadatom tačnošću. Sistemi nelinearnih jednačina su obrađeni kroz metode Newton-Kantoroviča, gradijentnu metodu i metodu proste iteracije. Ove metode imaju mnoge modifikacije i date su u osnovnoj fomi. Ove forme su neophodne za uspešno razumevanje raznih modifikacija. Problem faktorizacije polinoma je ilustrovan primerima u `Matlab`u. Detaljno su analizirane funkcije `fzero` i `fsolve`.

Četvrta glava se bavi interpolacijom. Do detalja su obrađeni Lagrangeov i Newtonov interpolacioni polinom, sa posebnim osvrtom na Lebesgueovu funkciju, zajedno sa posledicama koje ona donosi prilikom konstrukcija interpolanata visokog stepena. Jedno poglavlje je posvećeno grešci interpolacije. Prezentovani su sledeći interpolacioni polinomi u ekvidistantnim tačkama: prvi i drugi Newtonov interpolacioni polinom, Stirlingov i Besselov. Hermiteova interpolacija, kao i zanimljiv metod konstrukcije Hermiteovog interpolanta koji se ne sreće često u literaturi su obrađeni. Posebna pažnja posvećena je konstrukciji kubnog splajna, koja je razrađena do u detalje, zajedno sa funkcijama `Matlab`a koje se mogu koristiti za konstrukciju raznih tipova kubnog splajna. Poslednje poglavlje ove glave bavi se Čebiševljevim sistemima, uopštenim problemom interpolacije i trigonometrijskom interpolacijom.

Problem najmanjih kvadrata je predmet izučavanja pete glave. Data je osnovna metoda, kao i metoda sa težinama. Primerima su objašnjeni problemi koji se javljaju prilikom konstrukcije polinoma. Data je metoda konstrukcije koja se zasniva na ortogonalnim polinomima u slučaju ekvidistantnih čvorova. Prezentovana je metoda konstrukcije problema sa trendom, kao i metoda konstrukcije koja koristi Čebiševljeve sisteme funkcija. Poslednje poglavlje ove glave bavi se metodom linearne regresije i funkcijom `regress`.

Šesta glava je posvećena metodima numeričkog diferenciranja i integracije. Numeričko diferenciranje pokriva metode jednostranog i dvostranog diferenciranja prilikom izračunavanja izvoda proizvoljnog reda. Numerička integracija sadrži Newton-Cotesove formule, kao i njihove ekstenzije u obliku uopštenih kvadraturnih formula. Funkcija `Matlab`a `integral` i problemi numeričke integracije su predmet poslednjeg poglavlja.

Sedma glava izučava integracijom Cauchyevog problema. Na primeru Eulerovih metoda ilustrovani su svi bitni koncepti numeričke integracije Cauchyevog problema. Deo poglavlja je posvećen linearnim višekoračnim metodima, metodima Runge-Kutta i metodima sa diferenciranjem unazad, kao i njihovoj implementaciji u `Matlab`u. Posebna pažnja je posvećena rešavanju Cauchyevog problema u vektorskom obliku. Ilustrovani su stif sistemi diferencijalnih jednačina, kao i zavisnost rešenja po podacima.

Knjiga kroz obilje primera i slika ilustruje numeričke osobine prezentovanih algoritama. Uz svaku metodu je data funkcija u `Matlab`u koja implementira pomenutu numeričku metodu. Ove funkcije su pisane sa ciljem da posluže kao ilustracija i ne mogu se shvatiti kao profesionalni numerički software opšte namene. Laka razumljivost i kratkoća programa su osnovni zahtevi postavljeni prilikom pisanja funkcija. Smatramo da je ovaj pristup dobar, jer omogućava brzu implementaciju kojom se rezultati prezentovani u knjizi mogu usvojiti relativno jednostavno. Iako je uslov o jednostavnosti poštovan, implementirane funkcije ilustruju sve potrebne koncepte za izradu profesionalnog softwarea za implementaciju numeričkih algoritama. Takođe, implementirane funkcije omogućavaju čitaocu razumevanje najvažnijih koncepata u izradi numeričkog softwarea, što olakšava razumevanje i ovladavanje profesionalnim numeričkim softwareom. Uvođenje čitaoca u pojedine elemente kreiranja numeričkih algoritama ide gradacijom i svaki korak je motivisan primerom.

Smatramo da je integralna prezentacija numeričkih metoda i implementacija odgovarajućih algoritama u `Matlab`u dobra prilika da čitaoci dobiju praktična znanja koja će biti u mogućnosti da primene u oblastima za koje su zainteresovani.

Autori posebnu zahvalnost duguju recenzentima koji su pažljivim čitanjem uočili i omogućili otklanjanje grešaka u rukopisu.

Niš–Beograd, 28.01.2013. godine Autori

Glava 1

Elementi teorije grešaka

1.1 Aproksimacija brojeva

1.1.1 Rastojanje na skupu realnih brojeva

Predmet našeg izlaganja su približne vrednosti i aproksimacije. Prirodno nam je potreban način da merimo kvalitet aproksimacije najelementarnijih matematičkih objekata brojeva.

Rastojanje između dva broja x i y najčešće određujemo kao apsolutnu vrednost njihove razlike

$$d(x, y) = |x - y|.$$

Rastojanje obeležavamo sa d, a argumenti x i y, funkcije d, pokazuju da određujemo rastojanje brojeva x i y. Prisetimo se geometrijske interpretacije rastojanja. Naime, ako svakoj tački prave pridružimo realan broj, onda je $d(x, y)$ dužina duži $\overline{xy}$.

Osnovne osobine ovako definisanog rastojanja brojeva su:

0. Nenegativnost, za svaka dva broja $x, y \in \mathbb{R}$, važi $d(x, y) \geq 0$.

1. $d(x, y) = 0$ ako i samo ako je $x = y$.

2. $d(x, y) = d(y, x)$.

3. za svako $x, y, z \in \mathbb{R}$, važi $d(x, y) \leq d(x, z) + d(z, y)$.

Nulta osobina je očigledna, ako se setimo interpretacije rastojanja $d(x, y)$ kao dužine duži $\overline{xy}$. Prva osobina obezbeđuje osobinu identifikacije rastojanja. Naime, ako utvrdimo da su dva objekta na rastojanju nula, možemo sa sigurnošću tvrditi da su to isti objekti. Međutim, takođe iz ove osobine zaključujemo da objekat ne može biti od sebe na nekom pozitivnom rastojanju. Druga osobina iskazuje svojstvo simetrije, i izražava prostu činjenicu da, koliko je tačka x daleko od tačke y, toliko je i tačka y daleko od tačke x. Treća osobina se naziva nejednakost trougla. Ova osobina izražava iskustvenu činjenicu da je rastojanje između tačaka x i y manje, ili najviše jednako, od zbira rastojanja između tačaka x i z i tačaka z i y. Ove osobine rastojanja su dovoljne za zasnivanje pojma rastojanja. U apstraktnijim slučajevima, nego što je pomenuti slučaj realnih brojeva, zadovoljavanje pomenute četiri osobine vodi do valjanog pojma rastojanja.

Posledica ovih osobina rastojanja je sledeća osobina

4. $|d(x,z) - d(z,y)| \leq d(x,y)$.

Ova osobina se naziva druga nejednakost trougla. Primenjena na skup realnih brojeva ova osobina postaje $||x-z| - |z-y|| \leq |x-y|$. Ako izaberemo $z = 0$, dobijamo $||x| - |y|| \leq |x-y|$.

1.1.2 Apsolutna i relativna greška

Nama je rastojanje neophodno da bismo mogli da definišemo kvalitet aproksimacije. Aproksimacija broja x brojem x' je tim bolja što su brojevi x i x' na manjem rastojanju. Dakle, veličina $d(x,x') = |x - x'|$ ima značajnu ulogu u oceni kvaliteta aproksimacije broja x brojem x'. Zato je dobila posebno ime u ovom kontekstu.

Definicija 1.1 *Neka su x i x' realni brojevi. Apsolutna greška Δ pri aproksimaciji broja x brojem x' iznosi $\Delta = |x - x'|$. Sa Δ_x označavamo apsolutnu grešku broja x.*

Na primer, apsolutna greška aproksimacije broja $x = 2$ brojem $x' = 2.4$ iznosi

$$\Delta_x = |x - x'| = |2 - 2.4| = |-.4| = .4$$

Primetimo da na osnovu definicije sledi da je apsolutna greška aproksimacije broja x brojem x', ista kao i apsolutna greška aproksimacije broja x' brojem x.

Apsolutna greška jeste merilo aproksimacije, ali je nezavisna od kvantiteta. Zamislimo sledeću situaciju. Data su vam na čuvanje dva dinara, $x = 2$, a vašem kolegi dva miliona dinara $y = 2 \cdot 10^6$. Pretpostavimo da ste novac dobili u apoenima po jedan dinar. Posle mesec dana, vlasnik dođe i prebroji kod svakog od vas, koliko dinara imate. Pretpostavimo da kod vas nađe jedan dinar, $x' = 1$, a kod vašeg kolege dva miliona minus jedan dinar, $y' = 2 \cdot 10^6 - 1$. Apsolutna greška iznosi $\Delta = |x - x'| = |y - y'| = 1$ i kod vas i kod vašeg kolege. Zaključak donet samo na osnovu apsolutne greške je da ste podjednako dobri čuvari dinara. Naravno, ovo nije dobar zaključak. Problem je što apsolutna greška ne vodi računa o količini. Da bismo mogli da vodimo računa o količini uvodimo pojam relativne greške.

Definicija 1.2 *Relativna greška r pri aproksimaciji realnog broja $x \neq 0$, realnim brojem x' iznosi $r = |(x - x')/x| = \Delta_x/|x|$. Sa r_x označavamo relativnu grešku broja x.*

Ako koristimo pojam relativne greške, za ocenu uspešnosti vas i vašeg kolege, dobijamo $r_x = |(2-1)/2| = .5$, $r_y = |(2 \cdot 10^6 - 2 \cdot 10^6 + 1)/(2 \cdot 10^6)| = .5 \cdot 10^{-6}$. Izraženo u relativnim iznosima, vidimo da je kolega mnogo manje 'grešan'.

Ovde posebno naglašavamo da prilikom aproksimacije broja nula ne možemo definisati relativnu grešku. Ovo je potpuno prirodno, jer definicija relativne greške broja nula zahteva deljenje brojem nula. Poslednja operacija nije definisana.

U mnogim slučajevima nije moguće precizno odrediti apsolutnu i relativnu grešku aproksimacije nekog broja. Međutim, sasvim je zadovoljavajuće poznavati i moguće granice ovih grešaka. Recimo, merna oprema često dolazi sa specifikacijom gornje granice relativne greške za zadati opseg vrednosti merene fizičke veličine. U ovom slučaju ne znamo kolika je učinjena relativna greška prilikom merenja, ali znamo njenu gornju granicu. Gornju granicu apsolutne i relativne greške nekog broja x takođe označavamo sa Δ_x i r_x, redom. Ako je izmerena vrednost x', merene veličine x, sa gornjim granicama apsolutne i relativne greške Δ_x i r_x, važe sledeće nejednakosti

$$|x - x'| \leq \Delta_x, \quad \frac{|x - x'|}{|x|} \leq r_x.$$

Vrlo često se relativna greška određuje koristeći približnu formulu

$$r_x \approx r'_x = |(x - x')/x'|.$$

Veličinu r'_x nazivamo približnom relativnom greškom. Razlog za upotrebu veličine r'_x nije teško shvatiti. Pod pretpostavkom da je x dovoljno blizu x', veličine r_x i r'_x su, takođe, dovoljno blizu

$$r_x - r'_x = \left|\frac{x-x'}{x}\right| - \left|\frac{x-x'}{x'}\right| = |x-x'|\frac{|x'|-|x|}{|xx'|},$$
$$|r_x - r'_x| = \frac{|x-x'|}{|xx'|}||x|-|x'|| \leq \frac{|x-x'|}{|xx'|}|x-x'| = \frac{|x-x'|^2}{|xx'|} = r_x r'_x.$$

Kao što vidimo, razlika relativnih grešaka je manja ili jednaka proizvodu stvarne i približne relativne greške. Dakle, ako su relativne greške dovoljno male, manje od jedan recimo, učinjena greška prilikom zamene relativne greške r_x sa približnom relativnom greškom r'_x je mala.

Teorema 1.1 *Neka su r_x i r'_x relativna i približna relativna greška broja x. Važi sledeća relacija*

$$|r_x - r'_x| \leq r_x r'_x.$$

Na primer, recimo da merni instrument ima gornju granicu relativne greške 5%, a da je izmerena vrednost $x' = 1.2345$. Odredimo u kom intervalu se nalazi stvarna vrednost fizičke veličine. Kako je $r_x = 5\% = .05 \approx r'_x$, možemo odrediti

$$|x - x'| \leq .05x' = .05 \cdot 1.2345 = .0617, \quad -.0617 \leq x - x' \leq .0617, \quad 1.1728 \leq x \leq 1.2962.$$

Iz ovog primera jasno zaključujemo da stvarna vrednost fizičke veličine može biti bilo koji broj u intervalu $[1.1728, 1.2962]$. Drugim rečima, možemo zaključiti da stvarna vrednost merene veličine počinje cifrom 1. Međutim, već sledeća cifra može biti 1 ili 2. O sledećoj cifri nema puno smisla ni govoriti, jer ona može biti bilo koja. Koristeći ovo razmatranje dolazimo do ideje da većina cifara u broju x' nema puno smisla. Dovoljno je pisati $x' = 1.2$ i nećemo izgubiti nikakvu informaciju o rezultatu merenja. Na sličan način i granica apsolutne greške određuje interval u kome se nalazi stvarna vrednost broja x. Naime, $|x - x'| \leq \Delta_x$ je ekvivalentno sa $x' - \Delta_x \leq x \leq x' + \Delta_x$, dakle, $x \in [x' - \Delta_x, x' + \Delta_x]$.

Razvijanjem ideje, dolazimo do zaključka da broj 'smislenih' cifara u približnom broju, na neki način, zavisi od gornje granice apsolutne ili relativne greške koja je učinjena prilikom aproksimacije. Očigledno, od značaja je samo ona grupa cifara počev od cifre krajnje desno u zapisu broja koja se ne menja kad dodamo ili oduzmemo granicu apsolutne greške. U prethodnom primeru, ta grupa cifara se svodi na jednu jedinu cifru, to je cifra 1. Ostale cifre se menjaju prilikom dodavanja ili oduzimanja gornje granice apsolutne greške.

Pretpostavimo da je u prethodnom primeru merena vrednost ponovo $x' = 1.2345$, ali da je merni instrument sa gornjom granicom relativne greške $r_x = .5\%$. Posle istovetnog izračunavanja, dobijamo interval $[1.2283, 1.2407]$. Zaključujemo da su značajne cifre, broja x', grupa cifara 1, 2.

Pojam značajne cifre je ovde dat više intuitivno. Potpuno precizno pojam je razvijen u odeljku 1.2.2.

1.2 Reprezentacija brojeva

1.2.1 Brojevi u fiksnom i pokretnom zarezu

Svi realni brojevi mogu se predstaviti koristeći razvoj po stepenima osnove nekog brojnog sistema. Mi smo navikli da koristimo osnovu 10 i grupu cifara ove brojne osnove $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

U prvom delu izlaganja mi koristimo brojnu osnovu 10. Neki realan broj, recimo $a > 0$, se opisuje pomoću dve grupe cifara, jedne konačne a_i, $i = n, n-1, \ldots, 1, 0$, i druge beskonačne a_i, $i = -1, -2, \ldots$, na sledeći način

$$a = \sum_{i=0}^{n} a_i \cdot 10^i + \sum_{i=-1}^{-\infty} a_i \cdot 10^i.$$

Cifre koje su jednake nuli se obično izostavljaju iz zapisa. Na primer, broj .005 se zapisuje na sledeći način: $5 \cdot 10^{-3}$. Ako je broj a negativan, ceo izraz se stavlja u zagradu i ispred zagrade se stavlja znak minus, na primer, $a = -(5 \cdot 10^1 + 1 \cdot 10^0 + 5 \cdot 10^{-1}) = -51.5$. Naravno, ovakav način zapisivanja brojeva je neefikasan i postoji efikasniji, to je uobičajeni pozicioni način zapisivanja brojeva. Ideja je u sledećem: broj se zapisuje kao niz cifara koji uključuje decimalnu tačku koja se stavlja između cifre koja množi 10^0 i cifre koja množi 10^{-1}. Pozicija cifre u odnosu na decimalnu tačku jednoznačno određuje eksponent osnove u razvoju po stepenima osnove. Na primer,

$$5 \cdot 10^1 + 9 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 3 \cdot 10^{-3} + 4 \cdot 10^{-4} = 59.1234,$$

dakle, decimalna tačka se nalazi između cifara 9 i 1, jer cifra 9 množi 10^0, a cifra 1 množi 10^{-1}. Dalje se cifre redaju redom prema eksponentu osnove u razvoju po stepenima osnove. Primetimo da počev od decimalne tačke na levo stepen osnove raste počev od nule, dok počev od decimalne tačke na desno stepen osnove opada počev od -1. Ovakav način zapisivanja brojeva naziva se zapisivanje brojeva u fiksnom zarezu.

Ništa se bitno ne menja ako umesto osnove 10 izaberemo bilo koju drugu osnovu. Najčešće korišćene osnove su 2 i 16. U osnovi 2 skup cifara je $\{0, 1\}$, dok u osnovi 16 skup cifara predstavlja skup $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$, gde se umesto velikih slova A, B, C, D, E i F mogu koristiti i mala slova a, b, c, d, e i f. Značenje ovih cifara je sledeće: A označava 10, B označava 11, C označava 12, D označava 13, E označava 14 i F označava 15. Posebni simboli za označavanje brojeva od 10 do 15 su neophodni da bismo mogli da jednoznačno pišemo brojeve u ovom brojnom sistemu. Na primer, broj $B1 = 11 \cdot 16^1 + 1 \cdot 16^0$ je jednoznačno određen. Kada bismo koristili simbol 11 za B, dobili bismo 111 što može da se protumači kao $11 \cdot 16^1 + 1 \cdot 16^0$ ili $1 \cdot 16^1 + 11 \cdot 16^0$.

U nekim slučajevima je neophodno naznačiti osnovu brojnog sistema u kojoj je broj zapisan. Na primer, broj 10 ako je zapisan u osnovi 10 predstavlja broj $1 \cdot 10^1 + 0 \cdot 10^0 = 10$. Međutim, ako je zapisan u osnovi 2 predstavlja broj $1 \cdot 2^1 + 0 \cdot 2^0 = 2$. Da bismo naglasili osnovu u kojoj je broj zapisan pišemo $(10)_2$, gde indeks 2 znači da je broj zapisan u osnovi 2. Ako brojna osnova nije posebno naglašena, ne podrazumeva se iz teksta, i nema je u zapisu broja, podrazumevamo osnovu 10.

Predmet našeg interesovanja u daljem tekstu su izračunavanja na računskim mašinama, tako da smo zainteresovani za predstavljanje brojeva u računskim mašinama. Pretpostavimo da želimo da realne brojeve predstavljamo u fiksnom zarezu i razmotrimo posledice. Newtonova konstanta gravitacije, u SI sistemu jedinica, iznosi $G = 6.67384 \cdot 10^{-11}\mathrm{Nm}^2/\mathrm{kg}^2$. Planckova konstanta ima još manju vrednost, i iznosi $h = 6.626068 \cdot 10^{-34}\mathrm{Js}$, dok sa druge strane Avogadrov broj iznosi $L = 6.02214129 \cdot 10^{23}\mathrm{mol}^{-1}$. Kad bismo odlučili da predstavljamo brojeve u fiksnom zarezu morali bismo, sem ako ne želimo da se odreknemo računanja sa Planckovom konstantom ili Avogadrovim brojem, da budemo u stanju da predstavimo bar 23 cifre levo od decimalne tačke i bar 34 cifre desno od decimalne tačke. Potrebno nam je, otprilike, šezdesetak dekadnih cifara za predstavljanje ovih prirodnih konstanti u fiksnom zarezu. Problem sa ovakvim predstavljanjem je i značajan višak cifara pri predstavljanju Avogadrovog broja. Imali bismo na raspolaganju svih 60 cifara, dok bismo za predstavljanje Planckove konstante imali na raspolaganju svega 6 cifara. Zbog ovih

problema brojevi se u računskim mašinama ne predstavljaju u fiksnom zarezu. Rešenje problema je nađeno u formatu pokretnog zareza.

Broj $a > 0$ u pokretnom zarezu ima sledeći zapis $a = a^* \cdot 10^e$, gde je a^* mantisa, a e je eksponent. Na primer, imamo više načina za zapisivanje broja $a = 12.3456$ u pokretnom zarezu

$$a = 12.3456 = 1.23456 \cdot 10^1 = .123456 \cdot 10^2 = 123456. \cdot 10^{-4} = 12.3456 \cdot 10^0.$$

Primetimo da je i sam zapis u fiksnom zarezu, takođe, zapis u pokretnom zarezu sa eksponentom jednakim nula. Zapis broja u pokretnom zarezu nije jednoznačan za razliku od zapisa u fiksnom zarezu, koji jeste jednoznačan[1]. Da bi zapis u pokretnom zarezu bio jednoznačan uvodi se normalizovan zapis u pokretnom zarezu. Broj $a = a^* \cdot 10^e$ zapisan u normalizovanom obliku pokretnog zareza ima mantisu u intervalu $[.1, 1)$. Uslov ograničenja intervala mogućih vrednosti mantise obezbeđuje jednoznačnost zapisa u pokretnom zarezu. Primetimo da se zapisivanje broja u pokretnom zarezu može izvršiti u bilo kojoj brojnoj osnovi. Takođe, normalizovani zapis u pokretnom zarezu se odnosi na bilo koju brojnu osnovu. Međutim, interval mogućih vrednosti mantise zavisi od brojne osnove, jer broj .1 u različitim brojnim osnovama ima različite vrednosti. Na primer, u brojnoj osnovi 16 iznosi $(.1)_{16} = 1 \cdot 16^{-1} = 1/16 = .0625$, dok u osnovi 2 iznosi $(.1)_2 = 1 \cdot 2^{-1} = 1/2 = .5$.

Za reprezentovanje broja u računskim mašinama, brojevi u pokretnom zarezu predstavljaju mnogo bolje rešenje od brojeva u fiksnom zarezu. Kao što vidimo i Planckova konstanta i Avogadrov broj su već zapisani u pokretnom zarezu. Da bismo ove prirodne konstante reprezentovali u računaru dovoljno je samo zapamtiti odgovarajuću normalizovanu vrednost mantise i eksponent. Dakle, za Planckovu konstantu dovoljno je zapamtiti $m = .6626068$ i $e = -33$, dok je za Avogadrov broj dovoljno zapamtiti $m = .60221412$ i $e = 24$. Kao što vidimo broj cifara u mantisi je približno jednak, i iznosi oko 8. Takođe, broj cifara u eksponentu je jednak i iznosi 2. Ovakva razmatranja dovode do definisanja standarda za reprezentaciju brojeva u računskim mašinama. Ideja je da fiksiramo broj cifara za predstavljanje normalizovane mantise i za predstavljanje eksponenta. Ovaj standard razmatramo u odeljku 1.2.5.

1.2.2 Zaokruživanje brojeva i značajne cifre

Neka je x realan broj i neka je x' aproksimacija broja sa gornjom granicom apsolutne greške Δ, $|x - x'| \leq \Delta$. Kako smo pokazali ranije broj x se nalazi u intervalu $[x' - \Delta, x' + \Delta]$. U ovom odeljku nam je cilj da detaljno definišemo pojam značajne cifre. Pojam smo definisali na intuitivnom nivou u odeljku 1.1.2. Pre nego što damo preciznu definiciju pređimo na problem zaokruživanja brojeva.

Pretpostavimo da imamo sledeći problem: dat nam je broj $x = 12.345678$, koji očigledno ima osam cifara, i neka je potrebno smanjiti broj cifara ovog broja na četiri. Najjednostavniji postupak bi bio da prosto zanemarimo sve cifre broja x izuzev prve četiri, dakle, da za aproksimaciju proglasimo broj $x' = 12.34$. Prilikom ovakvog postupka učinjena apsolutna greška iznosi $\Delta = |x - x'| = .005678$. Međutim, možemo izabrati i broj $x'' = 12.35$. Prilikom ove aproksimacije učinjena apsolutna greška iznosi $\Delta' = |x - x''| = .004322$. Očigledno je $\Delta' < \Delta$. Druga aproksimacija je bolja. Očigledno je potrebno prilikom odbacivanja cifara odlučiti se da li povećati za jedan poslednju cifru koja ostaje u broju ili je ostaviti sa vrednošću koju ima. Algoritam koji obezbeđuje minimizaciju apsolutne greške je sledeći:

1. ako je prva cifra koja se odbacuje manja od pet, poslednja cifra koja ostaje se ne menja,

[1] Ovo tvrđenje nije u potpunosti tačno. Na primer, brojevi 1. i $.99\dot{9}\ldots$, gde se drugi broj sastoji iz cifre 9 koja se bez prekida ponavlja, su jednaki. Međutim, ova mala nepreciznost ne menja suštinu izlaganja.

2. ako je prva cifra koja se odbacuje veća od pet, poslednja cifra koja ostaje se uvećava za jedan,

3. ako je prva cifra koja se odbacuje pet i iza nje postoji još neka cifra koja se odbacuje različita od nule, onda se poslednja cifra uvećava za jedan,

4. ako je prva cifra koja se odbacuje pet i iza nje nema nijedne cifre koja se odbacuje a koja je različita od nule, onda, ako je poslednja cifra koja ostaje neparna uvećavamo je za jedan, a ako je parna ostaje nepromenjena.

Na primer, zaokruživanje sledećih pet brojeva $x_1 = 65.4321$, $x_2 = 23.4567$, $x_3 = 12.3450001$, $x_4 = 12.335$ i $x_5 = 12.345$ na četiri cifre ilustruje svih pet slučajeva u prethodnom algoritmu. Tako dobijamo $x_1' = 65.43$, $x_2' = 23.46$, $x_3' = 12.35$, $x_4' = 12.34$ i $x_5' = 12.34$, redom.

Pravilo koje se odnosi na odbacivanje cifre pet, pravilo 4., je na prvi pogled zbunjujuće. Naime, očigledno je da u ovom slučaju možemo postupiti na koji god hoćemo način, proizvedena apsolutna greška je ista u oba slučaja. Zato ostaje nejasno zašto se parne poslednje cifre tretiraju na jedan, a neparne poslednje cifre na drugi način. Razlog je u sledećem. Ako pretpostavimo da radimo sa velikom grupom brojeva koje treba zaokruživati i ako pretpostavimo da su cifre unutar te grupe brojeva uniformno raspodeljene, možemo pretpostaviti da ćemo u polovini slučajeva izvršiti zaokruživanje ka većoj vrednosti, a u drugoj polovini nećemo povećavati poslednju cifru. Kad saberemo ova povećanja i umanjenja, usled pravila zaokruživanja, dobićemo da je očekivana vrednost dobitka (gubitka) jednaka nula. Ovo je posebno značajno ako imamo na umu da neke od aplikacija koje koriste zaokruživanje brojeva mogu da rade u bankama. Na ovaj način aplikacije pisane za rad u bankama obezbeđuju da banke nemaju porast (pad) profita nastalu usled načina zaokruživanja.

Vratimo se sada definiciji pojma značajnih cifara. Pretpostavimo da broj $x' = 12.3456$ ima gornju granicu apsolutne greške $\Delta = .0046$. Postavlja se pitanje možemo li na osnovu ovih podataka odrediti koliko cifara broja x' je značajno, odnosno, koliko cifara broja x' će biti sadržano u broju x čija je x' aproksimacija. Posmatrajmo zapis brojeva x' i x u razvijenom obliku po stepenima osnove 10. Onda je očigledno

$$\begin{aligned} x' = 12.3456 &= 1\cdot 10^1 + 2\cdot 10^0 + 3\cdot 10^{-1} + 4\cdot 10^{-2} + 5\cdot 10^{-3} + 6\cdot 10^{-4} \\ x &= a_1\cdot 10^1 + a_0\cdot 10^0 + a_{-1}\cdot 10^{-1} + a_{-2}\cdot 10^{-2} + a_{-3}\cdot 10^{-3} + \cdots, \\ |x - x'| \le \Delta &= 0\cdot 10^1 + 0\cdot 10^0 + 0\cdot 10^{-1} + 0\cdot 10^{-2} + 4\cdot 10^{-3} + 6\cdot 10^{-4}. \end{aligned}$$

Kao što vidimo na prve četiri pozicije dolazi do poništavanja cifara u razlici brojeva x' i x, zato možemo reći da broj x' aproksimira broj x sa četiri značajne cifre.

Ova razmatranja nam omogućavaju sledeću definiciju.

Definicija 1.3 (Značajne cifre) *Neka je $x' = (x')^* 10^e$, $(x')^* \in [.1, 1)$, broj zadat sa gornjom granicom apsolutne greške Δ. Broj x' ima k značajnih cifara ako i samo ako je k najveći ceo broj takav da važi*

$$\Delta < 10^{e-k}.$$

Na primer, u prethodnom slučaju imamo $x' = 12.3456 = .123456\cdot 10^2$, $e = 2$. Odavde dobijamo

$$\Delta = .0046 = .46\cdot 10^{-2} = .46\cdot 10^{2-4} < 10^{2-4}.$$

Znači broj x' ima 4 značajne cifre.

Posmatrajmo sledeći primer. Neka je $x = .2522$ i $x' = .2518$. U ovom slučaju znamo apsolutnu grešku i ona iznosi $\Delta = |x - x'| = .4\cdot 10^{-3}$. Kako je $x' = .2518\cdot 10^0$ jednostavno nalazimo da važi

$\Delta = .4 \cdot 10^{-3} < 10^{0-3}$, pa je broj x' zadat sa 3 značajne cifre. Kao što vidimo broj značajnih cifara ne mora odgovarati broju jednakih cifara na početku brojeva x i x'. Međutim, oduzimanjem jasno vidimo da je broj nula na odgovarajućim pozicijama upravo tri

$$x - x' = 0 \cdot 10^{-1} + 0 \cdot 10^{-2} + 0 \cdot 10^{-3} + 4 \cdot 10^{-4}.$$

Posmatrajmo broj $x' = 1.0001 = .10001 \cdot 10^1$, $e = 1$, koji je aproksimacija broja $x = 1$. Onda je gornja granica apsolutne greške $\Delta = |x - x'| = .0001 = .1 \cdot 10^{-3}$. Zaključujemo da važi

$$\Delta = .1 \cdot 10^{-3} = .1 \cdot 10^{1-4} < 10^{1-4}.$$

Broj x' ima četiri značajne cifre, što se jednostavno zaključuje.

Odredimo broj značajnih cifara kojima broj $x' = 10^{14}$ aproksimira broj $x = 1$. Nalazimo $\Delta = 10^{14} - 1$ i $x' = .1 \cdot 10^{15}$, $e = 15$. Zaključujemo da važi $\Delta = 10^{14} - 1 < 10^{15-1} = 10^{e-k}$, pa je broj značajnih cifara $k = 1$. Ovo deluje iznenađujuće, ali ako oduzmemo brojeve

$$|x' - x| = 0 \cdot 10^{14} + 9 \cdot 10^{13} + \cdots + 9 \cdot 10^0,$$

vidimo da se stvarno pojavi jedna nula na vodećoj poziciji.

Odredimo broj značajnih cifara prilikom aproksimacije broja $x = 1$ brojem $x' = 10^{-14}$. U ovom slučaju je $\Delta = 1 - 10^{-14}$ i $x' = .1 \cdot 10^{-13}$, $e = -13$. Odavde nalazimo $\Delta = 1 - 10^{-14} < 10^{-13-(-13)} = 10^{e-k}$, pa je broj značajnih cifara $k = -13$. Ovaj rezultat možemo protumačiti na sledeći način. Prilikom oduzimanja brojeva

$$|x - x'| = 9 \cdot 10^{-1} + \cdots + 9 \cdot 10^{-13} + 9 \cdot 10^{-14},$$

pojavi se 13 cifara na pozicijama koje su ispred prve pozicije na kojoj postoji cifra različita od nule pri zapisu broja x'.

Sve cifre broja x su značajne, što je jednostavno razumeti. Ako broj x shvatimo kao aproksimaciju broja x, onda je apsolutna greška $\Delta = |x - x| = 0 < 10^{e-k}$, za proizvoljno $k > 0$, pa su sve cifre broja x značajne. Međutim, kako je apsolutna greška pri aproksimaciji broja x brojem x' jednaka apsolutnoj grešci pri aproksimaciji broja x' brojem x, možemo se zapitati sa koliko značajnih cifara broj x aproksimira broj x' i u kakvom odnosu stoje značajne cifre brojeva x i x' shvaćene na ovaj način. Videćemo da broj značajnih cifara brojeva x i x' nije uvek isti.

Teorema 1.2 *Ako brojevi x i x' imaju isti eksponent u normalizovanom zapisu pokretnog zareza, onda je broj značajnih cifara brojeva x i x' jednak.*

Dokaz. Po uslovima teoreme važi $x = x^* \cdot 10^e$ i $x = (x')^* \cdot 10^e$. Brojevi značajnih cifara k i k' u brojevima x i x', redom, su određeni kao najveći celi brojevi koji zadovoljavaju uslove

$$\Delta = |x - x'| < 10^{e-k}, \quad \Delta = |x - x'| < 10^{e-k'}.$$

Zaključujemo da je $k = k'$. □

Ako eksponenti u normalizovanom zapisu brojeva nisu isti, broj značajnih cifara ne mora biti jednak. Na primer, $x = .1 = .1 \cdot 10^0$, $e = 0$, i $x' = .09 = .9 \cdot 10^{-1}$, $e' = -1$. Onda je $\Delta = |x - x'| = .01 = 10^{-2}$. Broj značajnih cifara broja x je $k = 1$, jer je $\Delta = 10^{-2} < 10^{0-1}$, dok broj značajnih cifara broja x' iznosi $k' = 0$, jer je $\Delta = 10^{-2} < 10^{-1-0}$.

Pri lošim aproksimacijama dolazi do značajne razlike broja značajnih cifara brojeva x i x'. Ako je $x = 1$ i $x' = 10^{-14}$ broj značajnih cifara broja x iznosi $k = 1$, dok broj značajnih cifara broja x' iznosi $k' = -13$. Ako izaberemo $x = 1$ i $x' = 10^{14}$ broj značajnih cifara broja x iznosi $k = -13$, dok broj značajnih cifara broja x' iznosi $k' = 1$.

Definicija 1.4 *Broj značajnih cifara broja x' označavamo sa $z_{x'}$.*

Pojam broja značajnih cifara broja x' usko je povezan sa gornjom granicom približne relativne greške broja x'.

Teorema 1.3 *Neka je x' broj zadat sa gornjom granicom približne relativne greške r'. Onda broj značajnih cifara broja x' zadovoljava sledeću nejednakost*

$$z_{x'} < -\log_{10} |(x')^*| - \log_{10} r',$$

gde je $(x')^* \in [.1, 1)$ normalizovana mantisa broja x'.

Dokaz. Kako gornja granica apsolutne greške broja određuje broj značajnih cifara, na osnovu

$$\Delta < 10^{e-k},$$

deljenjem ove nejednakosti sa $x' = (x')^* \cdot 10^e$, nalazimo

$$r' = \frac{\Delta}{|x'|} < \frac{10^{e-k}}{|x'|} = \frac{10^{-k}}{|(x')^*|}.$$

Ako nađemo $-\log_{10}$ ove nejednakosti, i uzimajući u obzir da je ovo opadajuća funkcija te menja smisao nejednakosti, dobijamo

$$-\log_{10} r' > -\log_{10} \frac{10^{-k}}{|(x')^*|} = z_{x'} + \log_{10} |(x')^*|,$$

a odavde sledi tvrđenje teoreme. □

Kako normalizovana mantisa uvek zadovoljava uslov $|(x')^*| \in [.1, 1)$ to važi $-\log_{10} |(x')^*| \in (0, 1]$. Zbog ovoga se često broj značajnih cifara broja određuje na osnovu formule

$$z_{x'} \approx -\log_{10} r'.$$

Broj značajnih cifara broja x, kada broj x shvatamo kao aproksimaciju broja x', određujemo na sledeći način

$$z_x \approx -\log_{10} r.$$

Razlika u broju značajnih cifara brojeva x i x' se može odrediti jednostavno

$$z_x - z_{x'} \approx -\log_{10} r - (-\log_{10} r') = -\log_{10} \frac{|x'|}{|x|}.$$

Ako su brojevi $|x|$ i $|x'|$ približno isti, ako je aproksimacija dobra, možemo tvrditi da je $z_x \approx z_{x'}$. Na primer, za $x = .2522$ i $x' = .2518$ nalazimo

$$z_x \approx -\log_{10} \frac{|.2522 - .2518|}{.2522} \approx 2.7996851, \quad z_{x'} \approx -\log_{10} \frac{|.2522 - .2518|}{.2518} \approx 2.7989957.$$

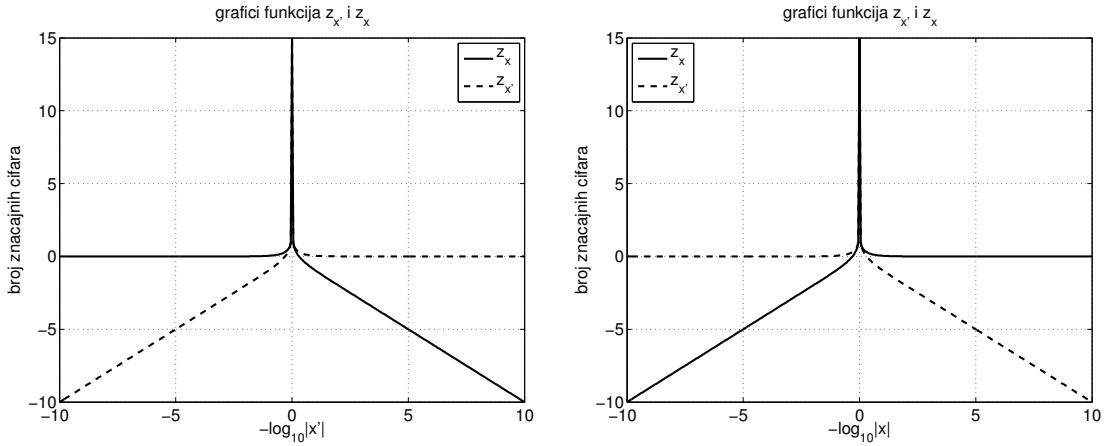
Vidimo da je razlika u broju značajnih cifara upotrebom jedne i druge formule zanemarljivo mala. Međutim, broj značajnih cifara može bitno da se razlikuje u slučaju loše aproksimacije. Izaberimo $x = 1$ i $x' = 10^{-14}$, onda nalazimo

$$z_x \approx -\log_{10} \frac{|1 - 10^{-14}|}{|1|} \approx 4.34 \cdot 10^{-15}, \quad z_{x'} \approx -\log_{10} \frac{|1 - 10^{-14}|}{|10^{-14}|} \approx -14.$$

Sa druge strane ako izaberemo $x = 1$ i $x' = 10^{14}$, dobijamo

$$z_x \approx -\log_{10} \frac{|1 - 10^{14}|}{|1|} \approx -14, \quad z_{x'} \approx -\log_{10} \frac{|1 - 10^{14}|}{|10^{14}|} \approx 4.34 \cdot 10^{-15}.$$

Odavde lako zaključujemo da je $z_x \approx z_{x'}$ ako je broj značajnih cifara pozitivan. Međutim, u slučaju da je broj značajnih cifara negativan ili približno jednak nuli funkcije z_x i $z_{x'}$ se značajno razlikuju. Naravno, funkciju z_x nije moguće uvek izračunati jer je vrednost x uglavnom nepoznata. Ako je vrednost x poznata, onda funkcija z_x ima prednost, jer nam omogućava da sagledamo red veličine relativne greške.



Slika 1.1: Slika levo grafici funkcija z_x i $z_{x'}$ za $x = 1$ dok se x' menja od 10^{-10} do 10^{10}, slika desno ilustruje iste funkcije ali za $x' = 1$ i x koje se menja u opsegu 10^{-10} do 10^{10}.

Slika 1.1, levo, prikazuje grafike funkcija z_x i $z_{x'}$ u slučaju $x = 1$ dok se x' menja od 10^{-10} do 10^{10}. Vidimo da se grafici funkcija z_x i $z_{x'}$ poklapaju ako je broj značajnih cifara pozitivan. Međutim, dolazi do velike razlike ako je aproksimacija loša. Slika 1.1, desno, prikazuje grafike funkcija z_x i $z_{x'}$ u slučaju $x' = 1$ dok se x menja u granicama od 10^{-10} do 10^{10}. Vidimo da je ovaj grafik slika u ogledalu prethodnog grafika.

Mi ćemo u knjizi kad god je moguće, što je slučaj kad nam je poznata vrednost x, crtati funkciju $z_x = -\log_{10} r$ i njen grafik ćemo zvati brojem značajnih cifara. Ovo je korektno u slučaju kad je broj značajnih cifara pozitivan, ali nije u potpunosti korektno kad su vrednosti funkcije z_x negativne ili približno jednake nuli. Međutim, na ovaj način dobijamo mogućnost da sa grafika funkcije z_x očitavamo red veličine relativne greške.

Posmatrajmo proces zaokruživanja broja $x = .123456$ na 5 cifara. Pravila zaokruživanja nalažu iz potreba minimizacije učinjene greške, da izaberemo broj $x' = .12346$. Broj x' ima apsolutnu grešku $\Delta' = |x - x'| = .4 \cdot 10^{-5}$. Kao što vidimo broj x' ima 5 značajnih cifara jer je $\Delta' < 10^{0-5}$. Posmatrajmo sada alternativu $x'' = .12345$. Ovakvim zaokruživanjem učinili smo grešku $\Delta'' = |x - x''| \approx .6 \cdot 10^{-5}$. Vidimo da i broj x'' ima 5 značajnih cifara jer je $\Delta'' \approx .6 \cdot 10^{-5} < 10^{0-5}$. Da bismo učinili razliku između ova dva načina zaokruživanja uvodimo pojam značajne cifre u užem smislu.

Definicija 1.5 (Značajne cifre u užem smislu) *Neka je broj $x' = (x')^* \cdot 10^e$, $|(x')^*| \in [.1, 1)$, zadat sa apsolutnom greškom Δ. Broj x' ima k značajnih cifara u užem smislu ako i samo ako je*

k najveći ceo broj koji zadovoljava uslov

$$\Delta \leq .5 \cdot 10^{e-k}.$$

Prethodni primer sad izgleda ovako $\Delta' = .4 \cdot 10^{-5} \leq .5 \cdot 10^{0-5}$, i broj x' ima 5 značajnih cifara u užem smislu. Za broj x'' sa druge strane, važi $\Delta'' \approx .6 \cdot 10^{-5} > .5 \cdot 10^{0-5}$, pa zaključujemo da broj x'' nema 5 značajnih cifara u užem smislu.

Gornja granica relativne greške broja određuje broj značajnih cifara u užem smislu. Ovo je relativno jednostavno razumeti. Pretpostavimo da broj $x' = (x')^* \cdot 10^e$ ima k značajnih cifara u užem smislu. Onda važi

$$\Delta \leq \frac{1}{2} \cdot 10^{e-k}.$$

Ako ovu nejednakost podelimo vrednošću $|x'| = |(x')^*|10^e$, gde je $|(x')^*| \in [.1, 1)$, dobijamo

$$r' \leq \frac{1}{2|(x')^*|} 10^{-k}.$$

Ako pomnožimo ovu nejednakost sa $2|(x')^*|$ i odredimo negativan logaritam za osnovu 10, dobijamo

$$-\log_{10}(2r') - \log_{10}|(x')^*| \geq k.$$

Vidimo da veličina $-\log_{10}(2r')$ približno određuje broj značajnih cifara k, a to je približno i eksponent broja 10 u veličini približne relativne greške.

Na osnovu ovih razmatranja možemo formulisati teoremu

Teorema 1.4 *Broj značajnih cifara u užem smislu, u oznaci $\overline{z}_{x'}$, broja x' sa približnom gornjom granicom relativne greške r', zadovoljava nejednakost*

$$-\log_{10}(2r') - \log_{10}|(x')^*| \geq \overline{z}_{x'}.$$

Za ocenu broja značajnih cifara u užem smislu broja x' obično se uzima

$$\overline{z}_{x'} \approx -\log_{10}(2r') = -.3 - \log_{10} r' \approx -.3 + z_{x'}.$$

Vidimo da se približne vrednosti broja značajnih cifara u užem smislu i broja značajnih cifara ne razlikuju bitno. U stvari u pitanju je faktor $-.3$. Zato se u praktičnim izračunavanjima za broj značajnih cifara gotovo uvek koristi ocena

$$z_{x'} \approx -\log_{10} r'.$$

Na primer, odredimo broj značajnih cifara i broj značajnih cifara u užem smislu broja $x' = .123456$ ako je broj zadat sa gornjom granicom apsolutne greške $\Delta = .0056$. Dobijamo

$$z_{x'} \approx -\log_{10} \frac{.0056}{.123456} \approx 1.34, \quad \overline{z}_{x'} \approx -\log_{10} \frac{2 \cdot .0056}{.123456} \approx 1.04.$$

Jednostavno zaključujemo da je broj značajnih cifara dva, jer $\Delta = .56 \cdot 10^{-2} < 10^{0-2}$, dok je broj značajnih cifara u užem smislu jedan, jer $\Delta = .56 \cdot 10^{-2} < .5 \cdot 10^{0-1}$. Vidimo da formule ne greše više od jedne cifre. Ovo u nekim slučajevima može biti previše, ali je za većinu aplikacija sasvim dovoljno.

1.2.3 Značajne cifre u proizvoljnoj brojnoj osnovi

U prethodnoj sekciji razmatran je broj značajnih cifara prilikom zapisivanja brojeva u osnovi 10. Jednostavno je zaključiti da je koncept univerzalan i da se prenosi na proizvoljnu brojnu osnovu. Tako na primer, za određivanje broja značajnih cifara u osnovi $b \in \mathbb{N} \setminus \{1\}$, možemo koristiti formule

$$z_{x'} \approx -\log_b r', \quad \overline{z}_{x'} \approx -\log_b(2r'). \tag{1.1}$$

Definicije broja značajnih cifara i broja značajnih cifara u užem smislu ostaju u važnosti, jedino se broj 10 zameni odgovarajućom brojnom osnovom.

Definicija 1.6 (Broj značajnih cifara u proizvoljnoj brojnoj osnovi) *Neka je $x' = (x')^* \, b^e$, $(x')^* \in [b^{-1}, 1)$, broj zadat sa gornjom granicom apsolutne greške Δ. Broj x' ima k značajnih cifara ako i samo ako je k najveći ceo broj takav da važi*

$$\Delta < b^{e-k}.$$

Neka je $x' = (x')^* \, b^e$, $(x')^* \in [b^{-1}, 1)$, broj zadat sa gornjom granicom apsolutne greške Δ. Broj x' ima k značajnih cifara u užem smislu ako i samo ako je k najveći ceo broj takav da važi

$$\Delta \leq \frac{1}{2} \, b^{e-k}.$$

Recimo, ako je $x = .46875$ i $x' = .4375$, $\Delta = |x - x'| = .3125 \cdot 10^{-1}$, pri čemu su brojevi zadati u osnovi deset, broj značajnih cifara i broj značajnih cifara u užem smislu u osnovi dva može se odrediti na sledeći način

$$\begin{aligned}
&x' = .34375 = (.111 \cdot 10^{-1})_2, \quad (x')^* = (.111)_2 \in [2^{-1}, 1), \quad e = -1, \\
&\Delta = .3125 \cdot 10^{-1} < 2^{-1-3} = .625 \cdot 10^{-1}, \quad k = 3, \\
&\Delta = .3125 \cdot 10^{-1} \leq \frac{1}{2} \, 2^{-1-3} = .3125 \cdot 10^{-1}, \quad k = 3, \\
&z_{x'} \approx -\log_2 \frac{\Delta}{x'} \approx 3.81, \quad \overline{z}_{x'} \approx -\log_2 \frac{2\Delta}{x'} \approx 2.81.
\end{aligned}$$

Prilikom određivanja broja značajnih cifara ili broja značajnih cifara u užem smislu prema definicijama, neophodno je prevođenje brojeva u odgovarajuću brojnu osnovu. Međutim, vidimo da prevođenje brojeva u drugu brojnu osnovu nije neophodno ako koristimo približne formule (1.1).

Broj značajnih cifara u osnovi dva je od važnosti, recimo, u sledećoj situaciji. Zamislimo da vršite merenje neke fizičke veličine i na izlazu senzora dobijate napon. Pretpostavimo da želite da vršite obradu rezultata merenja i da vrednost napona želite da analogno-digitalnim konvertorom prevedete u numeričke vrednosti. Postavlja se pitanje koliko bitova je potrebno upotrebiti prilikom koverzije, pod pretpostavkom da ne želimo da izgubimo informaciju sadržanu u rezultatu merenja. Odgovor je jednostavan. Broj potrebnih bitova je jednak broju značajnih cifara koju ima vrednost napona na izlazu iz senzora u osnovi dva. Kako svi senzori dolaze sa specifikacijom gornje granice relativne greške r za zadati interval merene veličine, dovoljno je samo odrediti vrednost $-\log_2 r$. Na primer, ako je gornja granica relativne greške 10^{-3}, potrebno je $-\log_2 10^{-3} \approx 10$ bitova pri analogno-digitalnoj konverziji.

1.2.4 Zapisivanje brojeva, naučna i inženjerska notacija

Prilikom računanja nema smisla raditi sa ciframa koje nisu značajne. Zamislite sledeći scenario. Dobijete da pomnožite ručno dva stocifrena broja od kojih svaki ima po šest značajnih cifara. Naravno, možete množiti stocifrene brojeve ili množiti šestocifrene brojeve. Rezultat je na kraju isti po pitanju broja značajnih cifara, ali se napor primetno razlikuje. Zato je preporučljivo prilikom zapisavanja brojeva pisati samo značajne cifre.

Prilikom zadavanja brojeva obično se zadaju samo značajne cifre u užem smislu. Ideja je u sledećem: zadavanjem samo zapisa približnog broja, a s obzirom na pravila zaokruživanja, potrebno je biti u stanju odrediti gornju granicu apsolutne greške kojom je broj zadat.

Na primer, posmatrajmo broj $x' = 1234.56$. Recimo da su sve cifre ovog broja značajne u užem smislu, onda, s obzirom na to kako se vrši zaokruživanje cifara, učinjena apsolutna greška prilikom zaokruživanja ne može biti veća od polovine stepena broja deset koji množi poslednju cifru u zapisu broja. Stepen broja deset koji množi poslednju cifru u zapisu broja je 10^{-2}, pa je gornja granica učinjene apsolutne greške $.5 \cdot 10^{-2}$. U ovom slučaju sve cifre zadatog broja su značajne. Međutim, posmatrajmo sledeći broj $x' = .0001234500$. U ovako zapisanom broju grupu nula kojom počinje broj, grupu nula krajnje levo, ne smatramo značajnom. Sve ostale cifre 1234500 smatramo značajnim, jer daju informaciju o tome kolika je učinjena apsolutna greška. Apsolutna greška na osnovu istog kriterijuma iznosi $.5 \cdot 10^{-10}$. Vodeće nule nisu značajne jer broj možemo zapisati, koristeći pokretni zarez, i na sledeći način $x' = 1.234500 \cdot 10^{-4}$. Vidimo da u ovom zapisu nule s početka zapisa broja nestaju.

Postoje slučajevi kada nule generišu i dvosmislenosti. Posmatrajmo sledeći broj $x' = 1230000$. kojim je zadata neka približna vrednost. U ovom zapisu nije jasno da li su nule značajne cifre ili služe samo za pozicioniranje decimalne tačke u zapisu u fiksnom zarezu. Ispravno zapisan broj, u slučaju da nule nisu značajne cifre, bi izgledao ovako $x' = 1.23 \cdot 10^6$. U slučaju da nule jesu značajne cifre, zapis bi izgledao ovako $x' = 1.230000 \cdot 10^6$. Pojedini autori, da bi istakli značajne cifre, koriste podvlačenje. Prethodni primer pod uslovom da su sve cifre značajne izgleda ovako $x' = \underline{1.230000} \cdot 10^6$.

Iz pomenutih razloga uvedene su naučna i inženjerska notacija.

Naučna notacija je notacija pokretnog zareza sa mantisom koja se nalazi u intervalu $[1, 10)$ i celobrojnim eksponentom. Njoj slična je inženjerska notacija. Inženjerska notacija je zapis u pokretnom zarezu gde je eksponent celobrojni umnožak broja tri. U inženjerskoj notaciji mantisa pripada intervalu $[1, 1000)$.

Napominjemo da broj zapisan bez decimalne tačke ne podleže pravilima za tumačenje učinjene apsolutne greške. Naime, takvi brojevi se smatraju apsolutno tačnim.

Na kraju pomenimo i notaciju kojom se često predstavljaju rezultati merenja. Vrednost Planckove konstante koja je rezultat trenutno najtačnijih merenja, prihvaćena 2010. godine od strane Komiteta za Podatke za Nauku i Tehnologiju (engleski Committee on Data for Science and Technology (CODATA)), može biti iskazana na sledeći način

$$h' = 6.62606957(29) \cdot 10^{-34}\text{Js}.$$

Kao što vidimo vrednost h' je u naučnoj notaciji, izuzev dve cifre u zagradi. Mi smo zainteresovani za njihovo značenje. U suštini ovi brojevi predstavljaju standardnu devijaciju obrađenih rezultata merenja, dok je sama vrednost h' srednja vrednost rezultata merenja. Sam proces merenja se shvata kao merenje slučajne veličine koja ima Gaussovu raspodelu. Možemo reći da $h' = 6.62606957 \cdot 10^{-34}$Js aproksimira Planckovu konstantu sa apsolutnom greškom $\Delta_1 = .00000029 \cdot 10^{-34}$Js sa verovatnoćom .683. Primetimo da je apsolutna greška formirana tako što poslednje dve cifre

u vrednosti h' zamenimo ciframa iz zagrade, dok ostale postavimo na vrednost nula. Pri ovoj operaciji ne menjamo eksponent. U pitanju su poslednje dve cifre broja h' jer je toliko cifara u zagradi, da je više (manje) cifara u zagradi zamenili bismo više (manje) cifara u broju h'. U ovom kontekstu se veličina Δ_1 obično naziva standardna neizvesnost (engleski standard uncertainty) rezultata merenja.

Kako je u pitanju standardna devijacija rezultata merenja, povećanje verovatnoće možemo postići umnošcima veličine Δ_1. Tako sa verovatnoćom .954 možemo tvrditi da je h' vrednost Planckove konstante sa gornjom granicom apsolutne greške $\Delta_2 = 2\Delta_1$. Sa verovatnoćom .997 može se tvrditi da je h' aproksimacija Planckove konstante sa gornjom granicom apsolutne greške $\Delta_3 = 3\Delta_1$.

U ovom slučaju možemo uzeti da je h' aproksimacija Planckove konstante sa gornjom granicom apsolutne greške Δ_3. Međutim, pomenimo da ovaj model dozvoljava sa verovatnoćom različitom od nule, mada vrlo malom, da recimo vrednost Planckove konstante ne pripada intervalu $[h' - 5\Delta_1, h' + 5\Delta_1]$. Birajući za gornju granicu apsolutne greške vrednost Δ_3, mi u stvari izvodimo izračunavanja i eventualnu analizu greške koje su ispravne sa verovatnoćom .997. Razumljivo je da ilustrovani koncept može da se primeni na sve fizičke veličine i sve rezultate merenja.

Relativna standardna neizvesnost se određuje kao količnik standardne neizvesnosti i apsolutne vrednosti srednje vrednosti merene veličine. Za konkretan primer Planckove konstante relativna standardna neizvesnost iznosi

$$\frac{\Delta_1}{|h'|} \approx 4.4 \cdot 10^{-8}.$$

Pojam relativne standardne neizvesnosti odgovara relativnoj grešci sa verovatnoćom .683.

Merena fizička veličina je tim preciznije izmerena što je manja relativna standardna neizvesnost. Trenutno najpreciznije izmerena prirodna konstanta je Rydbergova konstanta

$$R_\infty = 1.0973731568539(55) \cdot 10^7 \mathrm{m}^{-1},$$

sa relativnom standardnom neizvesnošću $5 \cdot 10^{-12}$.

1.2.5 Format realnog broja IEEE-754, tipovi single i double

Realni brojevi se u računaru predstavljaju u formatu koji je definisan standardom IEEE-754. Standard je donesen 1985. godine, a revidiran je 2008. godine. U ovom delu govorimo o osnovama standarda i izlažemo implementaciju standarda u `Matlabu`.

Nama su pre svega od interesa podaci tipa jednostruke i dvostruke preciznosti, u standardu označeni kao `binary32` i `binary64`. U `Matlabu` su reprezentovani podacima tipa `single` i `double`, redom. I jedan i drugi format za reprezentaciju realnih brojeva koriste reprezentaciju u pokretnom zarezu. Format jednostruke preciznosti zauzima tridesetdva bita ili četiri bajta memorije, dok format dvostruke preciznosti zauzima šezdesetčetiri bita ili osam bajtova memorije. Tabela 1.1 sadrži značenje pojedinih bitova u formatima jednostruke i dvostruke preciznosti.

Vodeći bitovi u oba formata predstavljaju bitove znaka, imaju vrednost 1 ako je predstavljen negativan broj i vrednost 0 ako je predstavljen pozitivan broj. Broj se, pre nego se upiše u neki od pomenutih formata, prevede u binarnu osnovu i to u zapis u pokretnom zarezu u formi $m \cdot 2^e$, gde je mantisa m u intervalu $[1, 2)$. Ideja je sledeća: kako je brojni sistem binarni i mantisa je ograničena na interval $[1, 2)$, to je prva cifra mantise izvesno 1, a to znači da prvu cifru mantise ne moramo pamtiti. Na ovaj način jedna cifra mantise ne mora biti zapisana i dobijamo uštedu od jednog bita. Mantisa normalizovana na ovaj način se u engleskoj literaturi naziva significand. Kad je broj preveden u pokretni zarez prosto se upisuju bitovi mantise na odgovarajuće pozicije

	znak	eksponent	pomeraj	mantisa
jednostruka preciznost	31	30 - 23	127 (2^7-1)	22 - 0
dvostruka preciznost	63	62 - 52	1023 $(2^{10}-1)$	51 - 0

Tabela 1.1: Značenje pojedinih bitova u jednostrukoj i dvostrukoj preciznosti

registra. Eksponent se, za razliku od mantise, upisuje sa pomerajem. Ne upisuje se binarno broj e, već se upisuje $e+127$, a u slučaju dvostruke preciznosti upisuje se $e+1023$.

Na primer, pretvorimo broj 23.7 u format jednostruke i format dvostruke preciznosti. Prvo je potrebno broj prevesti u binarni zapis. Ovo je najčešće posao koji radimo iz dva dela. Prvo pretvaramo celobrojni deo, a zatim razlomljeni deo. Celobrojni deo pretvaramo tako što delimo, celobrojno brojem dva, i pamtimo ostatke, dok ne dođemo do nule

$$\begin{aligned} 23:2 &= 11 \quad 1\\ 11:2 &= 5 \quad 1\\ 5:2 &= 2 \quad 1\\ 2:2 &= 1 \quad 0\\ 1:2 &= 0 \quad 1 \end{aligned}$$

Sad ako pročitamo ostatke unazad, dobijamo $(10111)_2 = 1\cdot 2^4 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 23$. Da bismo pretvorili razlomljeni deo, množimo razlomljeni deo brojem dva i pamtimo celobrojni deo

$$\begin{aligned} .7\cdot 2 &= 1.4, \quad 1\\ .4\cdot 2 &= \ .8, \quad 0\\ .8\cdot 2 &= 1.6, \quad 1\\ .6\cdot 2 &= 1.2, \quad 1\\ .2\cdot 2 &= \ .4, \quad 0 \end{aligned}$$

Očigledno je da se proces nadalje ponavlja. Čak, proces konverzije se nikad ne prekida. Dakle, broj .7 zapisan u binarnom brojnom sistemu izgleda ovako

$$.10110011001100110\ldots$$

Kad zapišemo rezultat konverzije zajedno dobijemo

$$10111.10110011001100110\ldots = 1.01111011001100110011001100110\ldots\cdot 10^{100}.$$

U jednostrukoj preciznosti se eksponent zapisuje sa pomerajem $127 = 2^7-1 = (1111111)_2$, dok se u dvostrukoj preciznosti eksponent zapisuje sa pomerajem $1023 = 2^{10}-1 = (1111111111)_2$. Eksponent broja u jednostrukoj preciznosti izgleda ovako $1111111+100 = 10000011$, dok eksponent broja u dvostrukoj preciznosti izgleda ovako $1111111111+100 = 10000000011$. Broj zapisan u jednostrukoj i dvostrukoj preciznosti, redom, izgleda ovako

$$0|10000011|01111011001100110011010$$
$$0|10000000011|0111101100110011001100110011001100110011001100110011001100110011$$

U formatu jednostruke preciznosti dolazi do zaokruživanja jer je prva zanemarena cifra jedinica, tako da se poslednja cifra 1 koja ostaje u zapisu uvećava za jedan.

Ovakav način zapisivanja je prilično nepregledan. Najčešće brojeve i ne zapisujemo ovako, nego grupišemo četiri binarne cifre u jednu heksadecimalnu cifru. Koristeći ovakvo grupisanje, prethodna dva broja dobijaju sledeći zapis

$$41BD999A$$
$$4037B33333333333$$

Problem sa heksadecimalnom notacijom je što polje znaka više nije samostalno. Dalje, polje eksponenta je pomešano sa poljem znaka kod dvostruke preciznosti, a kod jednostruke preciznosti i sa poljem znaka i sa poljem mantise.

Proveru rezultata možemo dobiti upotrebom funkcije `num2hex` u `Matlabu`.

```
>>num2hex(23.7)
ans =
4037b33333333333
>>num2hex(single(23.7))
ans =
41bd999a
```

Svaki broj zapisan u `Matlabu` se interno od strane interpretatora prepoznaje kao broj tipa `double`, dakle, kao broj dvostruke preciznosti. Da bismo funkciji `num2hex` prosledili argument u jednostrukoj preciznosti moramo koristiti konstruktor tipa `single`.

Kao što vidimo rezultat koji smo dobili je ispravan. Konverzija iz heksadecimalno zapisanog broja u formatu dvostruke preciznosti u broj zapisan u osnovi deset, se može postići korišćenjem funkcije `hex2num`.

```
>>hex2num('4037b33333333333')
ans =
  23.699999999999999
```

Kao što vidimo rezultat je približan. Rezultat ne može biti potpuno tačan, jer kao što smo videli konverzijom u osnovu dva zapis broja 23.7 postaje beskonačan. Zbog prethodnog, prilikom smeštanja broja 23.7 u formate jednostruke i dvostruke preciznosti napravili smo grešku odsecanja.

U vezi sa prethodnim je i pitanje određivanja granice relativne greške prilikom smeštanja realnih brojeva u formate jednostruke i dvostruke preciznosti. Možemo problem postaviti i ovako: kolike su gornje granice apsolutne greške brojeva

$$1.00000000000000000000000$$
$$1.00000000000000000000000000000000000000000000000000000$$

zadatih u binarnom sistemu. Ovo su reprezentacije broja 1 u formatu jednostruke i dvostruke preciznosti. Koristeći izložene metode, a kako i jedan i drugi format koriste algoritam zaokruživanja prilikom smeštanja, zaključujemo da su gornje granice apsolutne greške date sa $\Delta_s = 1/2 \cdot 2^{-23} = 2^{-24}$ i $\Delta_d = 1/2 \cdot 2^{-52} = 2^{-53}$, redom. Granice relativne greške koje odgovaraju ovim granicama apsolutne greške iznose $r_s = 2^{-24} \approx 5.96 \cdot 10^{-8}$ i $r_d = 2^{-53} = 1.11 \cdot 10^{-16}$, redom. Na osnovu prethodno iznesenog možemo zaključiti da format jednostruke preciznosti može predstaviti 7 dekadnih cifara

$$z_s = -\log_{10} r_s \approx 7.224719895935548,$$

dok format dvostruke preciznosti može predstaviti 16 dekadnih cifara

$$z_d = -\log_{10} r_d \approx 15.954589770191003.$$

Ako hoćemo da odredimo broj značajnih cifara u užem smislu potrebno je od ovih vrednosti oduzeti konstantu $-.3$.

Konstante koje smo prezentovali možemo dobiti u Matlabu koristeći funkciju `eps`.

```
>>eps
ans =
    2.220446049250313e-016
>>%2rd
>>2^(-52)
ans =
    2.220446049250313e-016
>>%broj znacajnih cifara u uzem smislu
>>-log10(eps)
ans =
  15.653559774527022
>>eps('single')
ans =
   1.1920929e-007
>>%2rs
>>2^(-23)
ans =
     1.192092895507812e-07
>>%broj znacajnih cifara u uzem smislu
>>-log10(eps('single'))
ans =
   6.9236898
```

Ako želimo da dobijemo vrednost gornje granice relativne greške formata jednostruke preciznosti koristimo funkciju `eps` sa argumentom `'single'`. Vrednosti `eps` i `eps('single')` predstavljaju dvostruke vrednosti gornjih granica relativnih grešaka formata dvostruke i jednostruke preciznosti, redom. Kao što vidimo direktno su povezane sa brojem značajnih cifara pojedinih formata, a imaju i sledeće značenje. Brojevi $2r_s$ i $2r_d$ su najmanji brojevi formata `single` i `double` koje treba dodati broju 1, u formatu jednostruke i dvostruke preciznosti, da bi se dobili brojevi koji su strogo veći od broja jedan.

Greška formata, kojom su zadati brojevi u jednostrukoj i dvostrukoj preciznosti, ima zanimljive posledice na izračunavanja. Posmatrajmo sledeći primer

```
>>a=4/3
a =
   1.333333333333333
>>b=a-1
b =
   0.333333333333333
>>c=3*b
c =
```

```
   1.000000000000000
>>e=1-c
e =
    2.220446049250313e-016
```

Vrednost promenljive `e` bi morala biti nula. Međutim, zbog grešaka koje nastaju prilikom smeštanja brojeva u računaru dolazi do pojave greške prilikom računanja. Mi ćemo se dalje baviti analizom načina za otklanjanje ovakvih grešaka ili barem načina za minimizaciju njihovog uticaja. Ovaj niz naredbi je korišćen od početka razvoja hardwarea za računanje sa realnim brojevima, jer je u početku svaki proizvođač imao sopstveni format realnog broja, a pomenuti niz naredbi je određivao dvostruku gornju granicu relativne greške formata. Naime, poređenjem vidimo da se dobijena veličina ne razlikuje od izlaza funkcije `eps`.

Vrednost funkcije `eps` se često naziva mašinskom preciznošću (na engleskom machine epsilon ili machine precision).

Jedna od fizičkih veličina koju je moguće meriti vrlo precizno, ako ne i najpreciznije, je vreme. Koristeći atomske satove na bazi Cezijuma, a prema metodologiji koja je razvijena 2011. godine u Nacionalnoj laboratoriji za fiziku (National Physical Laboratory, NPL) u Engleskoj, moguće je konstruisati merne instrumente koji greše manje od jedne sekunde svakih 138 miliona godina. Ovaj podatak se može pretvoriti u gornju granicu relativne greške

$$r_{\mathrm{Cs}} = \frac{1\mathrm{s}}{138 \cdot 10^6 \mathrm{god}} = \frac{1}{138 \cdot 10^6 \cdot 365 \cdot 24 \cdot 60 \cdot 60} \approx 2.3 \cdot 10^{-16}.$$

Kao što vidimo format dvostruke preciznosti je dovoljan da bez kreiranja dodatne greške primi i rezultate ovako preciznih merenja. Možemo zaključiti da je broj cifara kojima raspolaže format dvostruke preciznosti dovoljno dobar za reprezentaciju rezultata trenutno najpreciznijih merenja. Međutim, ovaj podatak će se promeniti u skoroj budućnosti, jer se očekuje konstrukcija još preciznijih mernih instrumenata za merenje vremena.

Specijalne vrednosti formata IEEE-754

Minimalna i maksimalna dozvoljena vrednost polja eksponenta u formatu dvostruke preciznosti iznose 1 i 2046. S obziom da se polje eksponenta dobija tako što se eksponentu dodaje pomeraj 1023, minimalni i maksimalni ekponent broja u formatu dvostruke preciznosti su -1022 i 1023. Vrednosti eksponenta 0 i 2047 su rezervisane za specijalne namene.

Jedna specijalna namena je očigledna. Naime, broj se smešta u format dvostruke preciznosti tako što se prvo dovede u formu pokretnog zareza pri čemu mantisa pripada intervalu $[1, 2)$. Ako je broj koji pokušavamo da prikažemo jednak nuli, nemoguće je dobiti mantisu u intervalu $[1, 2)$. Zato je za reprezentaciju broja nula potrebno upotrebiti neki specijalan način reprezentacije. Broj nula se reprezentuje tako što postavimo polje eksponenta na vrednost nula i polje mantise na vrednost nula, pri čemu je vrednost bita znaka nebitna.

```
>>num2hex(0)
ans =
0000000000000000
>>hex2num('8000000000000000')
ans =
     0
```

Drugim rečima, imamo dve ravnopravne reprezentacije broja nula, jednu sa bitom znaka jednakim nula i drugu sa bitom znaka jednakim jedan.

Ako polje mantise nije jednako nula predstavljamo takozvane subnormalne brojeve. Pretpostavimo da treba predstaviti broj $x = 1. \cdot 2^{-1025}$. Očigledno, ovaj broj je zapisan u normalizovanom obliku mantise i eksponenta. Međutim, vrednost eksponenta je isuviše mala i nije celobrojna vrednost iz intervala $[-1022, 1023]$. Imamo mogućnost da odbacimo ovaj broj i generišemo grešku prilikom pokušaja upisivanja takvog broja ili da ovakav broj prihvatimo kao regularnu vrednost sa poljem eksponenta postavljenim na nulu i jedinicom raspoređenom na ispravnu poziciju u mantisi.

```
>>num2hex(2^(-1025))
ans =
0002000000000000
```

Prevedeno u binarnu notaciju, sa podelom polja, broj izgleda ovako

$$0|00000000000|0010000000000000000000000000000000000000000000000000$$

Primetimo da je $-1025 + 1023 = -2$ pa se jedinica javlja na trećoj poziciji unutar mantise, jer je prva pozicija jedinice rezervisana za eksponent -1023, jer je $1. \cdot 2^{-1023}$ prvi broj koji nije reprezentabilan u normalizovanom obliku. Ovakvi brojevi se nazivaju brojevi sa subnormalnom mantisom. Brojevi sa subnormalnom mantisom određuju minimalni pozitivan broj koji se može reprezentovati u formatu dvostruke preciznosti. Očigledno minimalni broj u formatu dvostruke preciznosti je broj $(0000000000000001)_{16}$, koristeći osnovu dva to je broj 2^{-1074} ili prevedeno u osnovu 10, to je broj

```
>>hex2num('0000000000000001')
ans =
    4.940656458412465e-324
```

Postojanje subnormalnih brojeva ima svoju cenu. Naime, očigledno gornja granica relativne greške subnormalnih brojeva raste sa opadanjem eksponenta. Što manji eksponent to je sve veća i veća gornja granica relativne greške. Možemo grubo reći da kad god se eksponent subnormalnog broja smanji za jedan, eksponent gornje granice relativne greške takvog broja poraste za jedan. Najmanji pozitivan broj koji se može reprezentovati, dakle broj $x_{min} = 2^{-1074}$, ima gornju granicu apsolutne greške $\Delta_{min} = 1/2 \cdot 2^{-1074} = 2^{-1075}$. Odavde zaključujemo da je gornja granica relativne greške zadata sa $r_{min} = \Delta_{min}/x_{min} = 2^{-1} = .5$. Znači da ovaj broj nema nijednu značajnu cifru. Naravno, ovo je ekstremna situacija, i stvar je mnogo povoljnija za subnormalne brojeve sa većim vrednostima eksponenata. Ideja sa subnormalnim brojevima je u stvari želja da se mogućnosti hardwarea iskoriste do krajnjih granica.

Slična je situacija sa formatom jednostruke preciznosti. Polje eksponenta se može kretati u opsegu od 1 do 254, što daje moguće vrednosti eksponenta u rasponu od -126 do 127. Postoje dve moguće vrednosti reprezentacije broja nula, jedna sa bitom znaka postavljenim na vrednost jedan i druga sa bitom znaka postavljenim na vrednost nula, poljem eksponenta postavljenim na vrednost nula i poljem mantise postavljenim na vrednost nula. Takođe, postoje brojevi u subnormalnom formatu. Najmanji pozitivan broj u formatu jednostruke preciznosti je broj $(00000001)_{16}$, koristeći osnovu dva ovo je broj 2^{-149}, ili u dekadnom brojnom sistemu

```
>>single(2^(-149))
ans =
   1.4012985e-45
```

Postoje funkcije koje preračunavaju minimalnu moguću vrednost pozitivnog broja u pojedinom formatu. Za ove svrhe koristimo funkciju `realmin`.

```
>>a=realmin
a =
    2.225073858507201e-308
>>num2hex(a)
ans =
0010000000000000
>>num2hex(a/2^7)
ans =
0000200000000000
>>a/2^7
ans =
    1.738338951958751e-310
>>num2hex(a/2^20)
ans =
0000000100000000
>>a/2^20
ans =
    2.121995790965272e-314
>>a/2^100
ans =
    0
>>realmin('single')
ans =
   1.1754944e-038
```

Kao što vidimo postoje i manje vrednosti koje je moguće reprezentovati, od one koju vraća funkcija `realmin`, ali to su vrednosti sa subnormalnom mantisom. Vrednosti sa subnormalnom mantisom su one kod kojih raste gornja granica relativne greške ili, ekvivalentno, opada broj značajnih cifara u reprezentaciji broja. Iz ovog razloga funkcija `realmin` ne tretira subnormalne brojeve ravnopravno sa ostalim realnim brojevima reprezentabilnim u formatu dvostruke preciznosti. Interesantno je kad izađemo i iz opsega subnormalnih brojeva, kada dalje više nema nikakvih mogućnosti za reprezentaciju broja, `Matlab` proizvodi rezultat nula kao rezultat operacije deljenja bez ikakvog upozorenja. Da bismo dobili minimalni pozitivan broj koji se može reprezentovati u formatu jednostruke preciznosti, bez gubitka značajnih cifara, koristimo funkciju `realmin` sa argumentom `'single'`.

Opisali smo jednu ekstremnu vrednost polja eksponenta. Pozabavimo se drugom. Naime, postavlja se pitanje za šta su rezervisane vrednosti polja eksponenta 11111111 i 11111111111 u formatu jednostruke i dvostruke preciznosti, redom. Ako mantisa ima vrednost nula i polje eksponenta ima pomenute vrednosti, onda je reprezentovana beskonačnost, pozitivna ili negativna zavisno od bita znaka. Ako bit znaka ima vrednost nula reprezentovana je pozitivna beskonačnost, ako bit znaka ima vrednost jedan reprezentovana je negativna beskonačnost. Beskonačnosti se uvode na sličan način kao u matematičkoj analizi. Definiše se skup proširenih realnih brojeva, kao i operacije na ovom skupu koje obuhvataju beskonačnosti, a koje se mogu dobro definisati. Na primer, zbir realnog broja i beskonačnosti je naravno beskonačnost, proizvod realnog broja, različitog od nule, i beskonačnosti je beskonačnost i slično. Takođe, relacioni operatori veće, manje,

jednako i nejednako se definišu na proširenom skupu realnih brojeva i interpretiraju se tako što se negativna beskonačnost shvata kao najmanji element, a pozitivna beskonačnost kao najveći element skupa proširenih realnih brojeva. U `Matlabu` se koriste oznake `-inf`, `-Inf` i `inf`, `Inf` da označe negativnu i pozitivnu beskonačnost u formatu dvostruke preciznosti. U formatu jednostruke preciznosti potreban nam je konstruktor tipa `single`, na primer, `single(-inf)` i `single(inf)`.

```
>>num2hex([-inf, inf])
ans =
fff0000000000000
7ff0000000000000
>>1+inf
ans =
    Inf
>>1*inf
ans =
    Inf
>>1/0
ans =
   Inf
>>-inf<0
ans =
    1
>>inf == inf
ans =
    1
>>num2hex(realmax)
ans =
7fefffffffffffff
>>realmax*2
ans =
    Inf
>>0*inf
ans =
   NaN
```

Vidimo da su rezultati očekivani. Vrednost `realmax` određuje maksimalnu vrednost broja reprezentabilnog u formatu dvostruke preciznosti. Ako maksimalnu moguću vrednost formata dvostruke preciznosti pomnožimo brojem dva dobijamo beskonačnost kao rezultat. Da bismo dobili maksimalnu moguću vrednost koja se može reprezentovati u formatu jednostruke preciznosti pozivamo funkciju `realmax` sa argumentom `'single'`. Ono što je neočekivano, u prethodnom skriptu, je rezultat poslednje operacije. Kao što znamo proizvod nule i beskonačnosti, kao i razlika beskonačnosti istog znaka, zbir dve beskonačnosti različitog znaka, količnik dve beskonačnosti, količnik dve nule, i drugi izrazi, ne mogu biti definisani ni u skupu proširenih realnih brojeva. Za ove slučajeve rezervisana je posebna vrednost u formatima jednostruke i dvostruke preciznosti. To je vrednost `NaN` ili `nan`. Ovo je vrednost formata koja ne predstavlja realni broj (na engleskom **n**ot **a** **n**umber), i koristi se da označi rezultat operacije koji ne pripada skupu proširenih realnih brojeva. Vrednost `NaN` je reprezentovana poljem eksponenta koje ima vrednost 11111111 i 11111111111 u formatima jednostruke i dvostruke preciznosti, redom, i mantisom koja ima vrednost različitu od nule. Bit

znaka nema značaj i može imati proizvoljnu vrednost.

```
>>num2hex(NaN)
ans =
fff8000000000000
>>num2hex(single(NaN))
ans =
ffc00000
>>1+NaN
ans =
   NaN
>>0*NaN
ans =
   NaN
>>Inf+NaN
ans =
   NaN
>>1<NaN
ans =
    0
>>NaN == NaN
ans =
    0
>>NaN ~= NaN
ans =
    1
```

Karakteristika vrednosti `NaN` je da može da učestvuje u svim aritmetičkim operacijama i kao rezultat proizvodi uvek vrednost `NaN`. Takođe, ako učestvuje u izrazima sa relacionim operatorima proizvodi vrednost `false`. Čak, vrednost `NaN` nije jednaka sama sebi, dok je vrednost `NaN` različita od same sebe.

1.3 Računske operacije sa približnim brojevima

1.3.1 Aritmetičke operacije sa približnim brojevima

Cilj ovog odeljka je izučavanje granice relativne greške u rezultatu aritmetičkih operacija, kad su operandi brojevi zadati sa odgovarajućim granicama relativnih grešaka. Kako smo u prethodnim odeljcima pokazali da je pojam granice relativne greške povezan sa brojem značajnih cifara, možemo reći da proučavamo problem prostiranja značajnih cifara kroz aritmetičke operacije.

Neka su x' i y' približne vrednosti brojeva x i y, zadate sa gornjim granicama relativnih grešaka r_x i r_y. Gornja granica relativne greške operacije sabiranja se može odrediti na sledeći način

$$r_{x+y} = \frac{|x'+y'-x-y|}{|x+y|} \leq \frac{|x'-x|+|y'-y|}{|x+y|} = \frac{|x|}{|x+y|}\frac{|x'-x|}{|x|} + \frac{|y|}{|x+y|}\frac{|y'-y|}{|y|}$$
$$= \frac{|x|}{|x+y|}r_x + \frac{|y|}{|x+y|}r_y.$$

Slični rezultati se mogu pokazati za operacije množenja i deljenja:

$$\begin{aligned} r_{xy} &= \frac{|x'y' - xy|}{|xy|} = \frac{|x'y' - x'y + x'y - xy|}{|xy|} = \frac{|x'(y'-y) + (x'-x)y|}{|xy|} \\ &\leq \frac{|x'-x+x|}{|x|}\frac{|y'-y|}{|y|} + \frac{|x'-x|}{|x|} \leq r_x + \frac{|x'-x|+|x|}{|x|}r_y = r_x + r_y + r_x r_y, \\ r_{x/y} &= \frac{|x'/y' - x/y|}{|x/y|} = \frac{|x'y - xy'|}{|xy'|} = \frac{|x'y - xy + xy - xy'|}{|xy'|} = \frac{|(x'-x)y + x(y-y')|}{|xy'|} \\ &\leq \frac{|(x'-x)y|}{|xy'|} + \frac{|x(y-y')|}{|xy'|} \leq \frac{|y|}{|y'|}r_x + \frac{|y|}{|y'|}r_y = \frac{|y|}{|y'|}(r_x + r_y). \end{aligned}$$

Pod uslovom da granica relativne greške broja y zadovoljava uslov $r_y < 1$, možemo modifikovati izraz za relativnu grešku količnika. Naime, moguće je izraziti količnik y/y' koristeći relativnu grešku:

$$\frac{|y|}{|y'|} = \frac{|y|}{|y'-y+y|} \leq \frac{1}{\left|\frac{|y'-y|}{|y|} - 1\right|} = \frac{1}{1-r_y}.$$

Kombinujući dobijamo

$$r_{x/y} \leq \frac{r_x + r_y}{1 - r_y}.$$

Ove rezultate ćemo formulisati u obliku teoreme.

Teorema 1.5 *Neka su x', y' približne vrednosti brojeva x, y sa gornjim granicama relativnih grešaka r_x, r_y, redom. Gornje granice relativnih grešaka aritmetičkih operacija su zadate na sledeći način*

$$\begin{aligned} r_{x+y} &\leq \frac{|x|}{|x+y|}r_x + \frac{|y|}{|x+y|}r_y, \quad r_{xy} \leq r_x + r_y + r_x r_y, \\ r_{x/y} &\leq \frac{|y|}{|y'|}(r_x + r_y), \quad r_{x/y} \leq \frac{r_x + r_y}{1 - r_y}, \end{aligned}$$

pri čemu poslednja nejednakost važi pod uslovom $r_y < 1$.

Tvrđenje teoreme daje izraze koji opisuju 'najgori' mogući slučaj. U praksi se dešava da su relativne greške rezultata aritmetičkih operacija uglavnom manje od datih izraza. Naime, može doći do potiranja uticaja grešaka. Prilikom operacije sabiranja može se desiti da su greške suprotnog znaka. U celini uzev rezultujuća greška će biti manja od zbira apsolutnih vrednosti pojedinačnih grešaka, što naši izrazi u suštini predstavljaju.

U ekstremnom slučaju možemo zamisliti sledeću situaciju. Neka je $x = -1$, $y = 2$ i $x' = -2$, $y' = 3$. Onda je $x + y = x' + y' = 1$. Vidimo da je greška zbira jednaka 0. Međutim, naši izrazi bi dali gornju granicu relativne greške

$$r_{x+y} \leq \frac{|x|}{|x+y|}\frac{|x-x'|}{|x|} + \frac{|y|}{|x+y|}\frac{|y-y'|}{|y|} = 2.$$

Kažemo da u ovom slučaju dolazi do potiranja grešaka, jer greške deluju u suprotnom smeru.

Analizirajmo izraze koje daje teorema. Izraz za relativnu grešku operacije sabiranja je najznačajniji i njegovu analizu ostavljamo za kraj. Prvo analiziramo operaciju množenja. Upotrebljene

nejednakosti su oštre, u smislu da postoje vrednosti x, y i x', y' za koje nejednakosti postaju jednakosti. Konkretno, za operaciju množenja jednakosti važe kada su, recimo, sve vrednosti pozitivne i kada važi $x' > x$ i $y' > y$. Pod pretpostavkom da je $r_x, r_y < 1$, sabirak $r_x r_y$ možemo zanemariti u odnosu na izraz $r_x + r_y$, tako da možemo približno reći da važi $r_{xy} \approx r_x + r_y$. Ako pretpostavimo dalje da je $r_x = r_y$, dobijamo da operacija množenja povećava relativnu grešku dva puta. Na primer, pretpostavimo da množimo vrednost $x = .1$ dve stotine puta, pri čemu radimo sa vrednošću x' koja reprezentuje vrednost x u formatu dvostruke preciznosti. Kao što znamo gornja granica relativne greške iznosi $r_x = 2^{-53}$. Možemo očekivati da je rezultat množenja broja x' sa gornjom granicom relativne greške $200 \cdot r_x \approx 2.2 \cdot 10^{-14}$. Ovo razmatranje možemo ilustrovati upotrebom sledećeg skripta

```
x=0.1;
y=1;
for k=1:200
    y=y*x;
end
y
```

Rezultat izvršenja je

```
y =
    1.000000000000012e-200
```

Kao što vidimo dobijamo vrednost čija relativna greška iznosi približno $1.2 \cdot 10^{-14}$. Vidimo da naša teorijska ocena daje nešto veću vrednost relativne greške, što je posledica nekog potiranja grešaka tokom procesa množenja koje nismo uzeli u obzir. Generalno, možemo zaključiti da operacija množenja proizvodi uglavnom male greške prilikom računanja. Potrebno je izvršiti mnogo operacija množenja da bismo narušili broj značajnih cifara u rezultatu u prilično skromnim razmerama.

Slična operaciji množenja je i operacija deljenja. Prvi izraz za relativnu grešku operacije deljenja je vrlo sličan izrazu za relativnu grešku operacije množenja. Pod uslovom da je $|y/y'| \approx 1$, dobijamo $r_{x/y} \approx r_x + r_y$. Znači da analiza koju smo izveli za operaciju množenja važi i za operaciju deljenja. Možemo zaključiti da operacija deljenja ne proizvodi preveliki gubitak značajnih cifara.

Kako preveliki broj operacija množenja ili operacija deljenja uglavnom dovodi do prekoračenja formata dvostruke preciznosti, to je uticaj ovih operacija na gubitak značajnih cifara težak za posmatranje pojedinačno. Međutim, možemo ih posmatrati zajedno. Sledeći skript koristi deset hiljada operacija množenja i deljenja čiji rezultat ne prekoračuje opseg dvostruke preciznosti.

```
x=0.1;
z=0.09;
y=1;
for k=1:10^4
    if mod(k,2)==0
        y=y*x;
    else
        y=y/z;
    end
end
y
```

Rezultat izvršenja skripta je

```
y =
    6.129891723955239e+228
```

Vrednosti $x = .1$ i $z = .09$ su izabrane tako da proizvode greške u istom smeru u nejednakostima koje daju ocenu greške. Teorijska ocena je $10^4 \cdot 2^{-53} = 2.22 \cdot 10^{-12}$, dok je stvarna relativna greška u rezultatu jednaka $4.6 \cdot 10^{-13}$, a prava vrednost rezultata iznosi $6.129891723952415 \cdot 10^{228}$.

Drugi izraz za ocenu greške pri operaciji deljenja, u kome ne učestvuju vrednosti y i y', je grublji, i kao što vidimo, važi samo pod uslovom $r_y < 1$. U stvari je naveden da bismo ilustrovali da možemo dati izraz za gornju granicu relativne greške koji zavisi samo od gornjih granica relativne greške.

Pomenućemo da operacija množenja u opštem slučaju nije asocijativna. Recimo, ako izaberemo $x = y = 10^{300}$ i $z = 10^{-300}$, dobijamo

```
>>x=10^(300);y=10^(300);z=10^(-300);
>>x*(y*z)
ans =
    1.000000000000000e+300
>>(x*y)*z
ans =
    Inf
```

Očigledno je `x*(y*z)~= (x*y)*z`.

Operacija sabiranja je najzanimljivija i glavni je uzrok gotovo svih problema koji se javljaju prilikom izračunavanja. Uzrok možemo opisati na sledeći način. Zamislimo da brojevi x', y' aproksimiraju brojeve x, y sa mašinskom preciznošću. Neka je dalje $x = 1$ i $y = -1 + 10^{-k}$, gde je k neki prirodan broj. Koristeći izraz iz teoreme 1.5 dobijamo

$$r_{x+y} \leq \frac{|x|}{|x+y|} r_x + \frac{|y|}{|x+y|} r_y = \frac{1}{10^{-k}} r_x + \frac{1-10^{-k}}{10^{-k}} r_y \approx 10^k (r_x + r_y).$$

Vidimo da relativnu grešku u ulaznim podacima možemo po volji multiplikovati u izlaznim podacima.

```
>>x=1;y=-1+.00000001;
>>x+y
ans =
    1.000000005024759e-008
```

Kao što vidimo prava vrednost rezultata je `1.000000000000000e-008`. Međutim, dobili smo rezultat sa relativnom greškom

$$\left| \frac{1.000000005024759 \cdot 10^{-8} - 1 \cdot 10^{-8}}{1 \cdot 10^{-8}} \right| = 5.02 \cdot 10^{-9}.$$

Procenjena teorijska vrednost relativne greške iznosi $10^8 (r_x + r_y) = 2 \cdot 10^8 \cdot 2^{-53} \approx 2.22 \cdot 10^{-8}$. Naravno, na ovaj način možemo proizvesti gubitak onolikog broja značajnih cifara koliko želimo. Ovo je najznačajniji problem koji se javlja prilikom izračunavanja sa približnim vrednostima. Veliki deo numeričke analize i teorije algoritama se bavi upravo ovim problemom.

Operacija sabiranja takođe nije asocijativna. Na primer, posmatrajmo rezultat sledećih operacija

```
>>x=1;y=-1;z=10^(-17);
>>x+(y+z)
ans =
    0
>>(x+y)+z
ans =
    9.999999999999999e-018
```

Vidimo da rezultati nisu isti.

Postoji još jedna zanimljivost vezana za operaciju sabiranja. Posmatrajmo izraz $10^{17}+1$ i pokušajmo da predvidimo njegovu vrednost, ili pokušajmo da predvidimo vrednost promenljive y posle izvršenja sledećeg skripta

```
y=10^17;
for k=1:10^10
    y=y-1;
end
y
```

Očigledno, vrednost promenljive y bi morala iznositi $10^{17}-10^{10}$. Međutim, posle izvršenja skripta dobijamo

```
y =
    1.000000000000000e+017
```

Problem nastaje što svako pojedinačno oduzimanje broja jedan ne menja vrednost promenljive y. Razlog za ovakvo ponašanje je u sledećem. Promenljiva y ima vrednost 10^{17}, tako da poslednja cifra mantise posmatrano u dekadnom brojnom sistemu, koja može biti reprezentovana, odgovara poziciji desetica. Međutim, mi oduzimamo jedinice i računar jednostavno nema mogućnost da zabeleži promenu promenljive y, jer cifra na poziciji jedinica u vrednosti promenljive y ne postoji. Svih 10^{10} oduzimanja je uzaludno. Vrednost promenljive y ostaje nepromenjena.

U slučaju da brojevi koji se sabiraju nisu bliski po vrednosti a suprotnog znaka, ne dolazi do prevelikog gubitka značajnih cifara prilikom operacije sabiranja. U stvari, možemo pretpostaviti, bez gubitka opštosti, da su relativne greške r_x i r_y jednake i da izose 2^{-53}. U tom slučaju izraz za gornju granicu relativne greške postaje

$$r_{x+y} \leq \frac{|x|+|y|}{|x+y|} r_x.$$

Ako su promenljive x i y istog znaka, onda je $|x|+|y| = |x+y|$, pa dobijamo $r_{x+y} \leq r_x$. Ovaj rezultat sugeriše da ako se ograničimo na sabiranje brojeva istog znaka ne dolazi do povećanja relativne greške u rezultatu sabiranja. Ovaj zaključak nije u potpunosti tačan, a predmet je razmatranja u sledećoj sekciji.

1.3.2 Implementacija aritmetičkih operacija u računskim mašinama

Interesantno je da za razliku od operacija množenja i deljenja, koje relativno brzo dovode do prekoračenja opsega, operacija sabiranja može biti izvršena relativno veliki broj puta. Sledeći skript izvršava operaciju sabiranja deset milijardi puta.

```
x=0.1;
y=0;
for k=1:10^10
    y = y+x;
end
y
```

Rezultat izvršenja je

```
y =
    1.000000163124458e+009
```

Međutim, očigledno je tačna vrednost $10^{10} \cdot .1 = 10^9$. Relativna greška rezultata iznosi

$$\left| \frac{1.000000163124458 \cdot 10^9 - 10^9}{10^9} \right| \approx 1.63 \cdot 10^{-7}.$$

Kao što znamo greška ne bi trebalo da se uvaćava prilikom operacije sabiranja, ako sabiramo brojeve istog znaka. Postavlja se logično pitanje šta je uzrok pojave ovako velike relativne greške u rezultatu. Ova greška je posledica takozvane mašinske greške, koja nastaje otuda što računska mašina ne izvodi operacije sa apsolutnom tačnošću već ih izvodi sa greškom.

Ovu pojavu možemo shvatiti na sledeći način. Aritmetičke operacije množenje, deljenje i sabiranje se izvode u registrima mikroprocesora. Ovi registri imaju na raspolaganju veću količinu binarnih pozicija za reprezentaciju mantise, tako da je rezultat aritmetičkih operacija, u mikroprocesoru, reprezentovan sa većom dužinom mantise od one kojom raspolažu podaci tipa jednostruke ili dvostruke preciznosti. Međutim, posle završene aritmetičke operacije rezultat moramo smestiti u memoriju u odgovarajućem formatu jednostruke ili dvostruke preciznosti. Prilikom ovog smeštanja rezultata iz registara mikroprocesora u memoriju dolazi do zaokruživanja, pri čemu se pravi dodatna greška aritmetičke operacije. Učinjena greška aritmetičke operacije, uglavnom, ne može biti veća od gornje granice relativne greške formata koji koristimo za smeštanje podataka. Ovu grešku najčešće nazivamo mašinska greška odsecanja.

Najčešće je dovoljno da mislimo da se radi o većoj dužini mantise za jedan jedini bit. Radi ilustracije posmatrajmo sledeću sekvencu naredbi

```
>>num2hex(1)
ans =
3ff0000000000000
>>num2hex(1+2^(-52))
ans =
3ff0000000000001
>>num2hex((1+2^(-52))+2^(-53))
ans =
3ff0000000000002
>>num2hex(1-2^(-53))
ans =
3fefffffffffffff
```

Ovi primeri su jako ilustrativni. Prvo vidimo kako je reprezentovan broj jedan u formatu dvostruke preciznosti. Rezultat sledeće naredbe je reprezentacija broja $1 + 2^{-52}$. Vidimo jedinicu na poslednjoj poziciji u formatu koja odgovara poziciji 2^{-52}. Međutim, sledeća naredba proizvodi čudan

rezultat, jer kao što vidimo uvećava vrednost broja $1 + 2^{-52}$ za 2^{-53}. Ovakvo ponašanje bi bilo nemoguće da ne postoji izvršenje u registru koji ima mantisu barem za jedan bit dužu od formata dvostruke preciznosti. Po završetku operacije sabiranja brojeva $1 + 2^{-52}$ i 2^{-53}, dolazi do zaokruživanja rezultata na format dvostruke preciznosti i otuda jedinica na poziciji 2^{-51} u broju $(1 + 2^{-52}) + 2^{-53}$. Sličnom analizom objašnjavamo i poslednji primer. Napominjemo da ako promenimo vrednost 2^{-53} u, recimo, 2^{-54} aritmetičke operacije sa ovim operandom neće proizvoditi nikakve promene.

Teorema 1.5 koja uzima u obzir mašinsku grešku dobija sledeći oblik.

Teorema 1.6 *Neka su x', y' približne vrednosti brojeva x, y sa gornjim granicama relativnih grešaka r_x, r_y, redom. Gornje granice relativnih grešaka aritmetičkih operacija su zadate na sledeći način*

$$\begin{aligned}
r_{x+y} &\le \frac{|x|}{|x+y|}r_x + \frac{|y|}{|x+y|}r_y + \frac{|x'+y'|}{|x+y|}r_f \approx \frac{|x|}{|x+y|}r_x + \frac{|y|}{|x+y|}r_y + r_f,\\
r_{xy} &\le r_x + r_y + r_x r_y + \frac{|(x'y')|}{|(xy)|}r_f \approx r_x + r_y + r_x r_y + r_f,\\
r_{x/y} &\le \frac{|y|}{|y'|}(r_x + r_y) + \frac{|(x'/y')|}{|(x/y)|}r_f \approx \frac{|y|}{|y'|}(r_x + r_y) + r_f,\\
r_{x/y} &\le \frac{r_x + r_y}{1 - r_y} + \frac{|(x'/y')|}{|(x/y)|}r_f \approx \frac{r_x + r_y}{1 - r_y} + r_f,
\end{aligned}$$

gde poslednja nejednakost važi pod uslovom $r_y < 1$, a r_f je relativna greška zaokruživanja formata koja iznosi $r_f = r_d = 2^{-53}$ za format dvostruke preciznosti i $r_f = r_s = 2^{-24}$ za format jednostruke preciznosti.

Dokaz. Označimo sa $\circ$ neku aritmetičku operaciju od mogućih aritmetičkih operacija množenja, deljenja i sabiranja. Označimo brojeve koji učestvuju u aritmetičkoj operaciji sa x i y, njihove približne vrednosti sa x', y', i označimo $\overline{x' \circ y'}$ rezultat aritmetičke operacije sa mašinskom greškom. Onda je gornja granica relativne greške određena sa

$$\begin{aligned}
\left|\frac{\overline{x' \circ y'} - x \circ y}{x \circ y}\right| &= \left|\frac{\overline{x' \circ y'} - x' \circ y' + x' \circ y' - x \circ y}{x \circ y}\right|\\
&\le \left|\frac{x' \circ y'}{x \circ y}\right| \cdot \left|\frac{\overline{x' \circ y'} - x' \circ y'}{x' \circ y'}\right| + \left|\frac{x' \circ y' - x \circ y}{x \circ y}\right| = \left|\frac{x' \circ y'}{x \circ y}\right| r_f + r_\circ.
\end{aligned}$$

Sad samo ostaje da zamenimo operaciju $\circ$ sa konkretnim operacijama množenja, deljenja ili sabiranja da dobijemo tvrđenje teoreme.

Aproksimacije gornjih granica relativne greške dobijamo pod sledećim uslovima: $(x'y')/(xy) \approx 1$, $(x'/y')/(x/y) \approx 1$ i $(x' + y')/(x + y) \approx 1$. □

Koristeći ovu teoremu jednostavno je objasniti rezultate izračunavanja sa početka sekcije. Naime, ako uračunamo mašinsku grešku u rezultatu, za operaciju sabiranja nalazimo

$$r_{x+y} \le \frac{|x| + |y|}{|x + y|}r_d + r_d = 2r_d,$$

tako da procenjena teorijska vrednost gornje granice relativne greške iznosi $2^{-53} \cdot 10^{10} \approx 1.11 \cdot 10^{-6}$. Kao što vidimo ovaj rezultat se dobro poklapa sa dobijenom vrednošću.

Napomenimo na kraju da pomenuta mašinska greška ne mora da se javi. Na primer, sledeći skript ne generiše nikakvu grešku u rezultatu

```
x=.5;
y=0;
for k=1:10^8
    y = y+x;
end
y
```

Razlog za ovakvo ponašanje je što ne dolazi do greške pri reprezentovanju broja .5 u formatu dvostruke preciznosti, niti dolazi do greške prilikom izvršenja operacija sabiranja, jer su rezultat uvek brojevi koji su bez greške reprezentabilni u formatu dvostruke preciznosti. Naravno, ovakve situacije su retke i zahtevaju dodatnu analizu.

Primetimo na kraju da je teorema 1.5 sasvim dovoljna ako su nam podaci zadati sa relativnim greškama koje su mnogo veće od mašinske preciznosti.

1.3.3 Efekti izračunavanja u aritmetici konačne dužine

Iz kurseva o teoriji redova dobro je poznat razvoj u potencijalni red eksponencijalne funkcije

$$e^x = \sum_{k=0}^{+\infty} \frac{x^k}{k!}.$$

Takođe, vrlo dobro znamo da Maclaurinov red eksponencijalne funkcije uniformno konvergira na svakom zatvorenom intervalu realne prave. Iskoristićemo ove osobine Maclaurinovog reda da kreiramo algoritam za izračunavanje eksponencijalne funkcije u `Matlabu`. Jednostavno ćemo sabirati članove reda dok ne postignemo zahtevanu tačnost. Sledeća funkcija implementira algoritam.

```
function [y,err,iter]=expLose(x)
%expLose(X) izracunava vrednost funkcije exp(x)

a = 1; y = 0; iter = 0;
while abs(a)>100*eps*y
      y = y+a;
      iter=iter+1;
      a=a*x/iter;
      if iter > 300
           disp('maksimalan broj iteracija postignut');
           break;
      end;
end
y = y+a;
err=abs(a)/y;
end
```

Promenljiva `y` predstavlja sumu reda do nekog člana, preciznije promenljiva `y` predstavlja sumu reda do člana `iter`. Promenljiva `a` ima vrednost opšteg člana reda. Uslov za izlazak iz `while` petlje je ispunjen onoga trenutka, kad je sledeći član reda koji treba dodati manji od `100*eps*y`. Ovo je prilično razuman uslov, s obzirom na opisani gubitak značajnih cifara koji može nastati zbog sabiranja, množenja i deljenja.

Funkcija radi dobro ako je argument pozitivan. Evo nekoliko primera izvršenja i poređenja dobijenih rezultata sa funkcijom `exp`.

```
>>[y,err,iter]=expLose(1)
y =
     2.718281828459043
err =
     1.758271450130249e-14
iter =
    16
>>(y-exp(1))/exp(1)
ans =
   -9.802277420994503e-016
>>[y,err,iter]=expLose(100)
y =
    2.688117141816082e+043
err =
     1.669404228801942e-14
iter =
   184
>>(y-exp(100))/exp(100)
ans =
   -1.989459791956728e-014
```

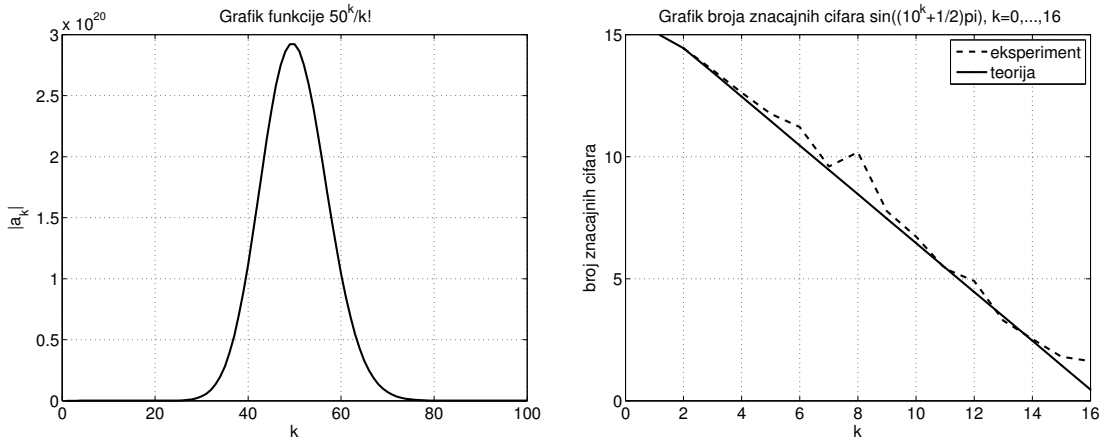
Kao što vidimo broj iteracija nije veliki, i postignuta relativna greška se uglavnom slaže sa onom koja je dobijena poređenjem sa ispravnim rezultatom. Dakle, sve je u najboljem redu. Međutim,

```
>>[y,err,iter]=expLose(-50)
y =
    1.107293338289200e+004
err =
     1.279946693620542e-14
iter =
   154
>>(y-exp(-50))/exp(-50)
ans =
    5.740989892795653e+025
>>exp(-50)
ans =
    1.928749847963918e-022
```

Ovde više ništa nije u redu. Vidimo da smo našim programom dobili da vrednost e^{-50} ima red veličine 10^4. Stvarna vrednost je reda veličine 10^{-22} kako vidimo. Primetimo da problem konvergencije ne postoji, naš program konvergira, i učinjena greška je reda veličine 10^{-14}, i postignuta je u 154 iteracije. Dakle, problem ne leži u tom delu. Predlažemo čitaocu da napravi program koji sabira po volji veliki broj članova reda i da se izvršenjem uveri da se rezultat ne menja ni kada saberemo milion ili milijardu članova reda. U stvari problem leži u formatu dvostruke preciznosti, a ne u broju članova reda.

Opšti član reda koji sabiramo ima sledeći oblik

$$a_k = \frac{(-50)^k}{k!} = (-1)^k \frac{50^k}{k!}.$$



Slika 1.2: Grafik funkcije $|a_k| = 50^k/k!$, $k = 1, \ldots, 100$, levo, i broj značajnih cifara u vrednostima $\sin((1/4 + 10^k)\pi)$, $k = 1, \ldots, 15$, desno.

Slika 1.2, levo, ilustruje funkciju $|a_k|$, $k = 1, \ldots, 100$. Kao što vidimo najveća vrednost apsolutne vrednosti člana reda $|a_k|$ iznosi oko 10^{20}. Dostiže se za vrednost $k = 50$. Kako su članovi reda prikazani u formatu dvostruke preciznosti i kako format dvostruke preciznosti ima na raspolaganju 16 dekadnih cifara, to je očigledno da član reda koji ima vrednost reda veličine 10^{20}, za cifru na poziciji deset hiljada jednostavno nema mesta u formatu dvostruke preciznosti. Setimo se da format dvostruke preciznosti ima na raspolaganju 16 dekadnih cifara i $20 - 16 = 4$. To dalje znači da su sva izračunavanja na poziciji deset hiljada pogrešna, i umesto da proizvedu nulu, koliko je stvarna vrednost cifre na toj poziciji, proizvode u našem konkretnom slučaju vrednost 1.

Ovaj vrlo jednostavan primer pokazuje sa koliko opreznosti se mora pristupiti izradi softwarea koji vrši izračunavanja. Naime, nije dovoljno da su ispunjeni uslovi konvergencije da zaključimo da program radi, kao što se vidi iz prikazanog primera.

1.3.4 Izračunavanje funkcija sa približnim vrednostima argumenata

Operacije sabiranja, množenja i deljenja su samo specijalan slučaj generalnog problema određivanja zavisnosti granice relativne greške vrednosti funkcije u funkciji relativne greške njenih argumenata. Ono što operacije sabiranja, množenja i deljenja čini specifičnim je njihova česta upotreba i relativna jednostavnost za analizu. U generalnom slučaju suočeni smo sa sledećim problemom. Data je funkcija $f : D \subset \mathbb{R}^n \to \mathbb{R}$ i date su približne vrednosti argumenata x_i', $i = 1, \ldots, n$ funkcije f, sa gornjim granicama relativnih grešaka r_i, $i = 1, \ldots, n$, odrediti relativnu grešku vrednosti funkcije f. Ovako najopštije postavljen problem teško da ima neko upotrebljivo rešenje. Zato dodatno pretpostavljamo da je funkcija f diferencijabilna i pretpostavljamo da su gornje granice apsolutnih grešaka relativno male. Ove pretpostavke dozvoljavaju upotrebu Taylorovog polinoma i osnovnog aparata matematičke analize.

Teorema 1.7 *Neka je $f : D \subset \mathbb{R}^n \to \mathbb{R}$ diferencijabilna funkcija u oblasti D, i neka su x_i', $i = 1, \ldots, n$, približne vrednosti argumenata x_i, $i = 1, \ldots, n$, funkcije f, sa gornjim granicama*

apsolutnih grešaka Δ_i, $i = 1, \ldots, n$. *Onda je*

$$|f(x) - f(x')| \leq \sum_{i=1}^{n} \left| \frac{\partial f(x')}{\partial x_i} \right| \Delta_i + \omega(x - x') \sqrt{\sum_{i=1}^{n} \Delta_i^2},$$

gde pozitivna funkcija ω *teži nuli kad njen argument teži nuli, i gde je* $x = (x_1, \ldots, x_n)$ *i* $x' = (x_1', \ldots, x_n')$.

Dokaz. Dokaz teoreme se zasniva na pojmu diferencijabilnosti. Naime, iz diferencijabilnosti funkcije f sledi

$$f(x) - f(x') = \sum_{i=1}^{n} \frac{\partial f(x')}{\partial x_i}(x_i - x_i') + \omega_1(x - x') \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2},$$

gde funkcija ω_1 teži nuli kad argument funkcije teži nuli. Ako primenimo apsolutnu vrednost na ovu jednakost i iskoristimo osobine apsolutne vrednosti, dobićemo

$$|f(x) - f(x')| \leq \sum_{i=1}^{n} \left| \frac{\partial f(x')}{\partial x_i} \right| |x_i - x_i'| + |\omega_1(x - x')| \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2}.$$

Zamenom odgovarajućih izraza za apsolutne greške, uz $\omega = |\omega_1|$, dobijamo tvrđenje teoreme. □

Pod uslovom da su granice apsolutnih grešaka male i uz pretpostavku neprekidnosti ili barem ograničenosti parcijalnih izvoda nad D, a s obzirom na činjenicu da je drugi sabirak

$$\omega(x - x') \sqrt{\sum_{i=1}^{n} \Delta_i^2}$$

u tvrđenju teoreme zanemarljiv u odnosu na Δ_i, $i = 1, \ldots, n$, možemo pojednostaviti izraz za gornju granicu apsolutne greške vrednosti funkcije

$$\Delta_f = |f(x) - f(x')| \leq \sum_{i=1}^{n} \left| \frac{\partial f(x')}{\partial x_i} \right| \Delta_i. \tag{1.2}$$

Poslednji izraz je najčešće u upotrebi.

Ako su umesto gornjih granica apsolutne greške date gornje granice relativne greške, izraz se dalje može transformisati u sledeći oblik

$$r_f' = \frac{|f(x) - f(x')|}{|f(x')|} \leq \sum_{i=1}^{n} \left| \frac{x_i'}{f(x')} \frac{\partial f(x')}{\partial x_i} \right| r_i', \tag{1.3}$$

gde smo koristili približnu vrednost gornje granice relativne greške $r_i' = \Delta_i / |x_i'|$, $i = 1, \ldots, n$, što ne unosi preveliku grešku pod uslovom $r_i r_i' < 1$, $i = 1, \ldots, n$. Izraz

$$S_{x_i}^f = \frac{x_i}{f(x)} \frac{\partial f(x)}{\partial x_i},$$

se u literaturi naziva parcijalnom osetljivošću. Razlog za ovaj naziv možemo videti iz izraza za granicu relativne greške. Naime, što je veća parcijalna osetljivost to se može očekivati veća promena relativne greške u vrednosti funkcije usled korišćenja približnih vrednosti argumenata za istu gornju granicu relativne greške argumenta.

Primetimo da ove nejednakosti treba shvatiti uslovno. Naime, ako treba aproksimirati vrednosti funkcije f u tački $x = (x_1, \ldots, x_n)$ na osnovu približnih vrednosti argumenata $x' = (x'_1, \ldots, x'_n)$, za svako $\varepsilon > 0$ postoji $\delta > 0$, tako da kad god je ispunjen uslov

$$\sqrt{\sum_{i=1}^{n}(x'_i - x_i)^2} < \delta,$$

važe nejednakosti

$$\Delta_f \leq \sum_{i=1}^{n}\left|\frac{\partial f(x')}{\partial x_i}\right|\Delta_i + \varepsilon, \quad r'_f \leq \sum_{i=1}^{n}\left|\frac{x'_i}{f(x')}\frac{\partial f(x')}{\partial x_i}\right| r'_i + \varepsilon.$$

Primenimo dobijeni izraz na operacije sabiranja, množenja i deljenja. Dobijamo sledeće ocene

$$\begin{aligned}
r'_{x+y} &= \frac{|x'+y'-x-y|}{|x'+y'|} \leq \left|\frac{x'}{x'+y'}\right| r'_x + \left|\frac{y'}{x'+y'}\right| r'_y,\\
r'_{xy} &= \frac{|x'y'-xy|}{|x'y'|} \leq \left|\frac{x'}{x'y'}y'\right| r'_x + \left|\frac{y'}{x'y'}x'\right| r'_y = r'_x + r'_y,\\
r'_{x/y} &= \frac{|x'/y'-x/y|}{|x'/y'|} \leq \left|\frac{x'}{x'/y'}1/y'\right| r'_x + \left|\frac{y'}{x'/y'}(-x'/(y')^2)\right| r'_y = r'_x + r'_y.
\end{aligned}$$

Vidimo da su dobijene nejednakosti samo prva aproksimacija nejednakosti koje smo dobili u teoremi 1.5.

Primenimo dobijeni rezultat na ocenu greške koja nastaje prilikom izračunavanja funkcije

$$f(x_1, x_2, x_3) = x_1 \sin(x_2 x_3)$$

u tački $(x'_1, x'_2, x'_3) = (1.234, 2.34567, 9.876)$. Kako je pretpostavka da su nam sve cifre zadatih brojeva značajne u užem smislu to su gornje granice apsolutnih grešaka pojedinih približnih vrednosti date sa $\Delta_{x_1} = .5 \cdot 10^{-3}$, $\Delta_{x_2} = .5 \cdot 10^{-5}$ i $\Delta_{x_3} = .5 \cdot 10^{-3}$. Nalazimo

$$\begin{aligned}
&\frac{\partial f(x')}{\partial x_1} = \sin(x'_2 x'_3) \approx -.923, \quad \frac{\partial f(x')}{\partial x_2} = x'_1 x'_3 \cos(x'_2 x'_3) \approx -4.70,\\
&\frac{\partial f(x')}{\partial x_3} = x'_1 x'_2 \cos(x'_2 x'_3) \approx -1.12.
\end{aligned}$$

Koristeći nejednakost (1.2) dobijamo

$$\begin{aligned}
|f(x) - f(x')| &\leq .923 \cdot .5 \cdot 10^{-3} + 4.70 \cdot .5 \cdot 10^{-5} + 1.12 \cdot .5 \cdot 10^{-3}\\
&\approx 1.04 \cdot 10^{-3}.
\end{aligned}$$

Da bismo iskoristili nejednakost (1.3), neophodne su nam parcijalne osetljivosti, a samim tim i vrednost funkcije, i približne vrednosti gornjih granica relativne greške promenljivih

$$f(x') = x_1' \sin(x_2' x_3') \approx -1.14, \quad S_{x_1}^f = \frac{x_1'}{f(x')} \frac{\partial f(x')}{\partial x_1} \approx 1.00,$$
$$S_{x_2}^f = \frac{x_2'}{f(x')} \frac{\partial f(x')}{\partial x_2} \approx 9.69, \quad S_{x_3}^f = \frac{x_3'}{f(x')} \frac{\partial f(x')}{\partial x_3} \approx 9.69,$$
$$r'_{x_1} = \frac{\Delta_{x_1}}{|x_1|} \approx 4.05 \cdot 10^{-4}, \quad r'_{x_2} = \frac{\Delta_{x_2}}{|x_2|} \approx 2.13 \cdot 10^{-6}, \quad r'_{x_3} = \frac{\Delta_{x_3}}{|x_3|} \approx 5.06 \cdot 10^{-5}.$$

Odavde nalazimo

$$r'_f \leq 1.00 \cdot 4.05 \cdot 10^{-4} + 9.69 \cdot 2.13 \cdot 10^{-6} + 9.69 \cdot 5.06 \cdot 10^{-5} = 9.16 \cdot 10^{-4}.$$

Naravno, kako su nejednakosti (1.2) i (1.3) jedna drugoj posledice, primećujemo da se rezultati dobro slažu $1.04 \cdot 10^{-3} \approx 1.14 \cdot 9.16 \cdot 10^{-4}$.

1.3.5 Uslovljenost izračunavanja

Nejednakost (1.3) je od značaja jer nam omogućava da zasnujemo pojam uslovljenosti izračunavanja. Pokazaćemo na primeru izračunavanja funkcije sin da se granica koju ova nejednakost predstavlja u stvarnosti dostiže.

Posmatrajmo granicu relativne greške koju dobijamo za izračunavanje funkcije sin u tačkama $x_k = \pi/4 + 10^k\pi$, $k = 1, \ldots, 15$. Primetimo da je vrednost funkcije sin u svim tačkama x_k, $k = 1, \ldots, 15$, ista i iznosi $\sin(x_k) = \sin(\pi/4) = \sqrt{2}/2$. Dobijamo

$$r'_{\sin} \leq \left| \frac{x_k}{\sin(x_k)} \cos(x_k) \right| r'_x = x_k r'_x = \left(\frac{\pi}{4} + 10^k \pi \right) r'_x.$$

Vidimo da se gornja granica relativne greške funkcije sin menja eksponencijalno sa k. Veličina r'_x iznosi 2^{-53}, jer radimo u formatu dvostruke preciznosti. Sledeći skript izračunava vrednosti funkcije sin u tačkama x_k, $k = 1, \ldots, 15$, i računa relativne greške.

```
x=pi*[2 10 10^2 10^3 10^4 10^5 10^6 10^7 10^8 10^9 10^10
10^11 10^12 10^13 10^14 10^15]+pi/4;
y=sin(x);
err=abs((y-y(1))/y(1))
```

Rezultat izvršenja je

```
err =
  Columns 1 through 3
                  0   0.000000000000004   0.000000000000028
  Columns 4 through 6
  0.000000000000239   0.000000000001767   0.000000000006138
  Columns 7 through 9
  0.000000000253577   0.000000000066009   0.000000017227233
  Columns 10 through 12
  0.000000190159670   0.000003826839798   0.000012537084124
```

```
Columns 13 through 15
0.000511756899111   0.002943354574523   0.015630803955754
Column 16
0.024143198245071
```

Slika 1.2, desno, prikazuje grafik funkcije `-log10 err`, drugim rečima, prikazuje značajne cifre u rezultatu, i grafik funkcije `-log10(x*2^(-53))`. Kao što vidimo teorijska ocena ne odstupa mnogo od dobijene eksperimentalne ocene.

Ovaj primer je jako ilustrativan, jer nam pokazuje da za neke vrednosti argumenta funkcije sin, broj značajnih cifara u vrednosti funkcije sin može biti značajno manji u odnosu na broj značajnih cifara argumenta funkcije sin. Čak, možemo zaključiti da koliki god bio broj značajnih cifara u argumentu funkcije sin, uvek postoji vrednost argumenta za koju vrednost funkcije sin nema nijednu značajnu cifru.

Međutim, situacija može biti i obrnuta. Posmatrajmo izračunavanje funkcije $f(x) = kx$, gde je $k \in \mathbb{R} \setminus \{0\}$. Ako primenimo formulu, dobićemo sledeću gornju granicu približne relativne greške

$$r'_f \leq \left| \frac{x'}{kx'} k \right| r'_x = r'_x.$$

Analizom izraza jednostavno zaključujemo da gornja granica približne relativne greške u vrednosti ne može biti veća od gornje granice približne relativne greške argumenta funkcije.

Za izračunavanje vezano za funkciju sin za velike vrednosti argumenta kažemo da je loše ili slabo uslovljeno (engleski ill-conditioned), dok za izračunavanje funkcije f kažemo da je dobro uslovljeno (engleski well-conditioned). Mera loše ili dobre uslovljenosti je faktor uslovljenosti

$$\mathrm{cond} f(x') = \left| \frac{x'}{f(x')} \frac{df(x)}{dx} \right|.$$

Što je faktor uslovljenosti veći, to je moguć veći gubitak značajnih cifara prilikom izračunavanja vrednosti funkcije, i obrnuto, što je faktor uslovljenosti manji, to je gubitak značajnih cifara prilikom izračunavanja vrednosti funkcije manji.

Koristeći faktor uslovljenosti možemo odrediti zavisnost značajnih cifara u vrednosti funkcije od broja značajnih cifara u vrednosti nezavisno promenljive. Od nejednakosti

$$r'_f \leq \mathrm{cond} f(x')\, r'_x,$$

ako odredimo negativnu vrednosti logaritma za osnovu 10, dobijamo

$$-\log_{10} r'_f \geq -\log_{10} \mathrm{cond} f(x') - \log_{10} r'_x \quad \Rightarrow \quad z_{f'} \geq -\log_{10} \mathrm{cond} f(x') + z_{x'}. \tag{1.4}$$

Kao što vidimo veličina $-\log_{10} \mathrm{cond} f(x')$ određuje donju granicu gubitka značajnih cifara u vrednosti funkcije.

Postoje i ekstremni slučajevi kada je moguće povećanje značajnih cifara u vrednosti funkcije u odnosu na broj značajnih cifara u argumentu funkcije. Uzmimo na primer funkciju $f(x) = x^{1/\alpha}$. Izračunavanjem dobijamo

$$r'_f \leq \left| \frac{x}{x^{1/\alpha}} \frac{1}{\alpha} x^{1/\alpha - 1} \right| r'_x = \frac{1}{\alpha} r'_x, \quad \mathrm{cond} f(x') = \frac{1}{\alpha}.$$

Izaberimo $\alpha = 1000$. Ako je gornja granica relativne greške u argumentu funkcije 10^{-1}, vrednost funkcije će imati gornju granicu relativne greške manju od 10^{-4}. Ilustraciju dobijamo vrlo jednostavno

```
>>y=1.1^(1/1000)
y =
   1.000095314721964
>>y-1
ans =
    9.531472196377955e-005
```

Ako iskoristimo nejednakost (1.4), dobijamo

$$z_{f'} \geq -\log_{10} \operatorname{cond} f(x') + z_{x'} = -\log_{10} \frac{1}{10^3} + 1 = 3 + 1 = 4,$$

odakle zaključujemo da je broj značajnih cifara u vrednosti funkcije najmanje četiri. Ovo je upravo ilustrovano u primeru.

Pojam dobro i loše uslovljenih izračunavanja je od izuzetne važnosti i mi ćemo se sa njim susretati kroz celu knjigu. Ovde je dat samo kao uvod koji opisuje pojam na primeru funkcije jedne realne promenljive.

Glava 2

Sistemi linearnih jednačina

Sa $\mathcal{M}_{nm}$ označavamo skup svih matrica tipa $n \times m$, drugim rečima, skup svih matrica sa n vrsta i m kolona. Ako može doći do dvosmislenosti, na primer, u slučaju $\mathcal{M}_{234}$ nije jasno da li govorimo o matricama tipa 23×4 ili matricama tipa 2×34, koristimo notaciju $\mathcal{M}_{n,m}$. Mi ćemo se baviti isključivo realnim matricama, tako da smatramo da su elementi matrica realni brojevi. Specijalan slučaj ovog skupa je skup vektora kolona $\mathcal{M}_{n,1}$ i vektora vrsta $\mathcal{M}_{1,m}$. Za skup $\mathcal{M}_{n,1}$ često koristimo notaciju $\mathbb{R}^n$. Za označavanje matrica koristimo velika slova latinice, dok za vektore koristimo mala slova latinice. Na primer,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}, \quad y = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}.$$

Za vektore kolone još koristimo i notaciju uređenih n-torki, zbog identifikacije skupova $\mathcal{M}_{n,1}$ i $\mathbb{R}^n$, tako da u slučaju $n = 3$, važi

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = (1, 2, 3).$$

Skup kvadratnih matrica $\mathcal{M}_{nn}$ označavamo jednostavno sa $\mathcal{M}_n$, i kažemo da matrice imaju red n.

Sistem linearnih jednačina u opštem slučaju ima oblik

$$Ax = b,$$

gde je $A \in \mathcal{M}_{nm}$ matrica sistema linearnih jednačina, $x \in \mathcal{M}_{m,1}$ je vektor promenljivih, dok je $b \in \mathcal{M}_{n,1}$ slobodan vektor. Sistem linearnih jednačina je kvadratni ako je matrica sistema kvadratna, drugim rečima, ako je broj jednačina sistema jednak broju promenljivih. Egzistencija rešenja sistema linearnih jednačina može se utvrditi, recimo, primenom Kronecker-Cappelijeve teoreme. Međutim, ako sistem jednačina ima rešenje sasvim je drugi problem pronaći rešenje sistema jednačina.

Najjednostavniji sistem za rešavanje je kvadratni sistem linearnih jednačina. Mi ćemo se uglavnom baviti proučavanjem kvadratnih sistema linearnih jednačina.

2.1 Norme vektora i matrica

2.1.1 Norme vektora

Da bismo mogli da govorimo o aproksimaciji vektora neophodno je, kao u slučaju realnih brojeva, imati definisano rastojanje na skupu vektora. Sa definisanim rastojanjem dobijamo mogućnost da govorimo o vektorima koji su daleko ili blizu jedan drugom, da jedan vektor više ili manje aproksimira drugi i slično. Rastojanje dva vektora $u, v \in \mathbb{R}^n$ obeležavaćemo sa $d(u, v)$ i zahtevaćemo da zadovoljava iste osobine kao rastojanje u slučaju realnih brojeva, koje smo definisali u odeljku 1.1.1.

Na skupu vektora mnogo nam je bliži pojam intenziteta vektora koji smo navikli da koristimo. Najčešći način za definisanje intenziteta vektora $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, je sledeći

$$||x||_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}.$$

U Matematici se intenzitet vektora naziva norma vektora. Mi dalje u tekstu umesto intenzitet vektora koristimo pojam norma vektora. Ovako definisana norma vektora naziva se Euklidova norma. Najvažnije osobine Euklidove norme su izložene u sledećoj lemi.

Lema 2.1 *Neka su $x, y \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$ i $|| \cdot ||_2 : \mathbb{R}^n \to \mathbb{R}_0^+$ Euklidova norma. Onda važi*

0° $||x||_2 \geq 0$,
1° $||x||_2 = 0 \Leftrightarrow x = 0$,
2° $||\lambda x||_2 = |\lambda| \, ||x||_2$,
3° $||x + y||_2 \leq ||x||_2 + ||y||_2$.

Dokaz. Osobina 0° se dokazuje trivijalno. Naime, kako je $x \in \mathbb{R}^n$, to su sve koordinate vektora x realni brojevi, pa su njihove apsolutne vrednosti nenegativni brojevi. Zbir kvadrata nenegativnih brojeva je nenegativan broj, i konačno kvadratni koren iz nenegativnog broja je nenegativan broj. Dakle, za svako $x \in \mathbb{R}^n$ važi $||x||_2 \geq 0$. Ako je $x = 0$, onda je $||x||_2 = (\sum 0^2)^{1/2} = 0^{1/2} = 0$. Ako je $||x||_2 = 0$, onda je $(\sum |x_i|^2)^{1/2} = 0$, odakle sledi da je $\sum |x_i|^2 = 0$. Kako je zbir nenegativnih brojeva, što kvadrati realnih brojeva sigurno jesu, jednak nula, to svaki broj pojedinačno mora biti jednak nula. Dakle, $|x_i|^2 = 0$, $i = 1, \ldots, n$, odakle sledi $x_i = 0$, $i = 1, \ldots, n$, što konačno daje $x = 0$. Da bismo dokazali osobinu 2° dovoljno je samo zameniti izraz λx u definiciju Euklidove norme

$$||\lambda x||_2 = ||(\lambda x_1, \ldots, \lambda x_n)||_2 = \left(\sum_{i=1}^{n} |\lambda x_i|^2 \right)^{1/2} = |\lambda| \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} = |\lambda| \, ||x||_2.$$

Osobina 3° se ne može dokazati elementarnim metodima i njen dokaz izostavljamo. □

Pojedinačno ove osobine imaju sledeće interpretacije. Prva osobina znači da norma vektora ne može biti negativna. Ovo je vrlo jednostavno za interpretaciju ako se setimo da je norma vektora drugo ime za intenzitet vektora. Intenzitet vektora, u geometrijskoj interpretaciji, predstavlja rastojanje između početne i krajnje tačke vektora, pa sasmim tim ne može imati negativnu vrednost. Druga osobina govori da nula vektor ima normu nula i da normu nula ima nula vektor. Na ovaj način je uspostavljena identifikacija nula vektora i vektora koji imaju normu nula. Treća jednakost

govori o homogenosti norme. U suštini, izražava činjenicu da ako pomnožimo koordinate vektora nekim brojem λ, onda se dužina vektora menja $|\lambda|$ puta. Poslednja nejednakost se naziva nejednakost trougla. Leva strana predstavlja dužinu vektora $x + y$, dok desna strana predstavlja zbir dužina vektora x i y. Vektori x, y i $x+y$ obrazuju trougao, pa nejednakost iskazuje osobinu stranica trougla da zbir dužina dve stranice nije manji od dužine treće.

Koristeći normu vektora moguće je definisati rastojanje dva vektora.

Lema 2.2 *Neka su vektori $u, v, w \in \mathbb{R}^n$. Onda je $d(u, v) = ||u - v||_2$ rastojanje vektora koje zadovoljava uslove*

0° $d(u, v) \geq 0$,
1° $d(u, v) = 0 \Leftrightarrow u = v$,
2° $d(u, v) = d(v, u)$,
3° $d(u, v) \leq d(u, w) + d(w, v)$.

Dokaz. $d(u, v) = ||u - v||_2 \geq 0$ što je posledica leme 2.1, osobina 0°. Ako je $d(u, v) = 0$, onda je $||u - v||_2 = 0$. Na osnovu leme 2.1 osobine 1° sledi da je $u - v = 0$, dakle, $u = v$. Ako je $u = v$, onda je $u - v = 0$, pa je $||u - v||_2 = ||0||_2 = 0$. Za dokaz osobine 2° koristimo osobinu 2° iz leme 2.1. Imamo

$$d(u, v) = ||u - v||_2 = ||(-1)(v - u)||_2 = |-1|||v - u||_2 = ||v - u||_2 = d(v, u).$$

Za dokaz osobine 3° koristimo osobinu 3° iz leme 2.1

$$d(u, v) = ||u - v||_2 = ||(u - w) + (w - v)||_2 \leq ||u - w||_2 + ||w - v||_2 = d(u, w) + d(w, v).$$

Ovim smo završili dokaz leme. □

Euklidova norma nije jedina norma koja se može definisati na skupu vektora. Postoje i drugi načini da se odredi norma vektora, a da su zadovoljene osobine norme nabrojane u lemi 2.1. Na primer, sledeći izraz

$$||u||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad p \geq 1,$$

je norma vektora. Za $p = 2$ dobijamo, već pomenutu, Euklidovu normu vektora. Ako u prethodnom izrazu za normu dozvolimo da p postane neograničeno veliki broj, dobijamo

$$||x||_{+\infty} = \max_{i=1,\ldots,n} |x_i|,$$

koju nazivamo beskonačna norma vektora.

Lema 2.3 *Norme vektora $|| \cdot ||_p$, $p \geq 1$ ili $p = +\infty$, zadovoljavaju sve osobine norme iz leme 2.1.*

Dokaz je potpuno isti kao i u slučaju Euklidove norme i ne navodimo ga.

U praksi se najčešće koriste norme za $p = 1$, $p = 2$ i $p = +\infty$. Na primer, izračunajmo ove norme za vektor $x = (1, -2, 3)$. Imamo redom

$$||x||_1 = |1| + |-2| + |3| = 6, \quad ||x||_2 = \left(|1|^2 + |-2|^2 + |3|^2\right)^{1/2} = \sqrt{14},$$
$$||x||_{+\infty} = \max\{|1|, |-2|, |3|\} = 3.$$

Jedna od primena uvedenih normi je određivanje gornje granice izraza

$$||(a_1x_1, \ldots, a_nx_n)||_1 = \sum_{i=1}^{n} |a_ix_i|.$$

Neka je $|a_\ell| = \max_{i=1,\ldots,n} |a_i|$, onda je

$$\sum_{i=1}^{n} |a_ix_i| \leq \sum_{i=1}^{n} |a_\ell| \cdot |x_i| = |a_\ell| \sum_{i=1}^{n} |x_i| = ||a||_{+\infty}||x||_1.$$

Drugim rečima, dobili smo ocene

$$||(a_1x_1, \ldots, a_nx_n)||_1 \leq ||a||_{+\infty}||x||_1, \quad ||(a_1x_1, \ldots, a_nx_n)||_1 \leq ||a||_1||x||_{+\infty}.$$

Druga ocena sledi iz činjenice da vektori a i x nastupaju u prvoj nejednakosti simetrično na levoj strani.

Svaka od opisanih normi omogućava formiranje odgovarajućeg rastojanja na prostoru vektora. Pojedinačna rastojanja označavamo simbolima d_p, $p \geq 1$ ili $p = +\infty$, i definišemo ih na sledeći način $d_p(u, v) = ||u - v||_p$, $u, v \in \mathbb{R}^n$.

Lema 2.4 *Rastojanja d_p, $p \geq 1$ ili $p = +\infty$, zadovoljavaju sve osobine rastojanja iz leme* 2.2.

Dokaz je potpuno isti kao u dokazu leme 2.2 i ne navodimo ga.

Primetimo da se norma preko rastojanja može izračunati koristeći izraz $||x||_p = d_p(x, 0)$. Norma vektora je jednaka rastojanju vektora od koordinatnog početka.

Kad imamo rastojanje možemo da govorimo o aproksimaciji vektora. Možemo da govorimo o apsolutnoj grešci vektora i relativnoj grešci vektora. Ako je x' približna vrednost vektora x, onda apsolutnu grešku i relativnu grešku definišemo sa

$$\Delta = d_p(x', x) = ||x' - x||_p, \quad r = \frac{d_p(x', x)}{d_p(x, 0)} = \frac{||x' - x||_p}{||x||_p},$$

redom. Kao u slučaju broja nula, tako i u slučaju nula vektora nema smisla određivati relativnu grešku nula vektora.

Bitna posledica dokazanih lema za rastojanje vektora, preciznije posledica prve nejednakosti trougla, je druga nejednakost trougla.

Lema 2.5 *Neka je $p \geq 1$ ili $p = +\infty$ i $u, v \in \mathbb{R}^n$, onda važi*

$$|\,||u||_p - ||v||_p| \leq ||u - v||_p.$$

Dokaz. Koristeći prvu nejednakost trougla nalazimo

$$||u||_p = ||u - v + v||_p \leq ||u - v||_p + ||v||_p, \quad ||v||_p = ||v - u + u||_p \leq ||v - u||_p + ||u||_p.$$

Odavde izračunavamo

$$||u||_p - ||v||_p \leq ||u - v||_p, \quad ||v||_p - ||u||_p \leq ||v - u||_p = ||u - v||_p.$$

Iz ove dve nejednakosti sledi tvrđenje leme. □

Norme vektora nisu međusobno nezavisne. Naime, uvedene norme vektora zadovoljavaju izvesne nejednakosti. Najvažnije su prezentovane u sledećoj lemi.

Lema 2.6 *Neka je $x \in \mathbb{R}^n$ onda važi*

1° $||x||_2 \leq ||x||_1 \leq \sqrt{n}||x||_2$,
2° $||x||_{+\infty} \leq ||x||_2 \leq \sqrt{n}||x||_{+\infty}$,
3° $||x||_{+\infty} \leq ||x||_1 \leq n||x||_{+\infty}$.

Ove nejednakosti nećemo dokazivati, međutim, uz pomoć ovih nejednakosti možemo zaključiti da ako neki vektor x' aproksimira neki vektor x u jednoj normi, on to radi u svim normama koje su od interesa. Tvrđenje da neki vektor x' aproksimira neki drugi vektor x ne zavisi od upotrebljene norme. Naravno, rastojanje između vektora će zavisiti od upotrebljene norme, ali kao što vidimo, neće se bitno razlikovati[1].

2.1.2 Norme matrica

Pored vektora, gde je pojam norme neophodan za definisanje pojma rastojanja dva vektora, u skupu matrica se rastojanje takođe uvodi koristeći pojam norme matrica. Međutim, ne želimo da uvodimo bilo kakvu normu u skupu matrica, već želimo da norma u skupu matrica bude saglasna sa odgovarajućom normom vektora. Normu na skupu matrica $\mathcal{M}_{nm}$ ćemo označavati simbolom $||\cdot||$, kao i normu vektora. Iz konteksta će biti jasno da li se misli na normu vektora ili matrica.

Definicija 2.1 *Reći ćemo da je norma matrica saglasna sa normom vektora ako i samo ako za svaku matricu $A \in \mathcal{M}_{nm}$ i svaki vektor $x \in \mathbb{R}^m$ važi $||Ax|| \leq ||A||\,||x||$.*

Kako smo definisali razne norme vektora ne možemo očekivati da jedna norma matrica može biti saglasna sa svim definisanim normama vektora. Za svaku od pomenutih normi vektora, definiše se posebna norma matrica koja je saglasna sa odgovarajućom normom vektora. Pre nego što definišemo odgovarajuće norme matrica, saglasne sa pomenutim normama vektora, pomenimo da postoji način definisanja saglasne norme matrica za proizvoljnu normu vektora.

Lema 2.7 *Neka su na prostoru vektora $\mathbb{R}^n$ i $\mathbb{R}^m$ definisane norme $||\cdot||_n$ i $||\cdot||_m$, redom. Norma na prostoru matrica $\mathcal{M}_{nm}$ određena sa*

$$||A|| = \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||Ax||_n}{||x||_m} = \sup_{x\in\mathbb{R}^m, ||x||_m=1} ||Ax||_n, \quad A \in \mathcal{M}_{nm}, \tag{2.1}$$

je saglasna sa normom vektora $||\cdot||_m$. Ovako definisana norma matrica zadovoljava sve osobine norme. Za $A, B \in \mathcal{M}_{nm}$, i $\lambda \in \mathbb{R}$, važi

0° $||A|| \geq 0$,
1° $||A|| = 0 \Leftrightarrow A = 0$,
2° $||\lambda A|| = |\lambda|\,||A||$,
3° $||A + B|| \leq ||A|| + ||B||$.

Dodatno, ako je $A, B \in \mathcal{M}_n$, važi

4° $||AB|| \leq ||A||\,||B||$. (2.2)

[1] Ovde se pre svega misli na konvergenciju niza vektora. Ako neki niz vektora $x_k \in \mathbb{R}^n$, $k \in \mathbb{N}$, konvergira u jednoj normi ka vektoru $x \in \mathbb{R}^n$, on će konvergirati u svim normama ka vektoru x. Ova osobina se naziva ekvivalentnost normi. Može se pokazati da su sve norme na skupu vektora $\mathbb{R}^n$ ekvivalentne.

Dokaz. Primetimo da po definiciji norme (2.1), za svako $x \in \mathbb{R}^m \setminus \{0\}$, važi

$$||A|| \geq \frac{||Ax||_n}{||x||_m} \geq 0.$$

Odavde neposredno sledi nejednakost 0°. Takođe, ako je $||A|| = 0$, onda je za svako $x \in \mathbb{R}^m \setminus \{0\}$ $||Ax||_n/||x||_m = 0$, pa je $||Ax||_n = 0$. Na osnovu osobina norme vektora zaključujemo da je $Ax = 0$, za svako $x \in \mathbb{R}^m$. Jedina matrica koja pomnožena svakim vektorom daje nula vektor je nula matrica. Dakle, $A = 0$. Ako je $A = 0$, onda je $||Ax||_n/||x||_m = 0/||x||_m = 0$. Zaključujemo da je $||A|| = \sup_{x\in\mathbb{R}^m} 0 = 0$. Dokazali smo osobinu 1°. Izaberimo $\lambda \in \mathbb{R}$, onda je

$$||\lambda A|| = \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||(\lambda A)x||_n}{||x||_m} = \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{|\lambda|\,||Ax||_n}{||x||_m} = |\lambda| \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||Ax||_n}{||x||_m} = |\lambda|\,||A||.$$

Izaberimo $A, B \in \mathcal{M}_{nm}$. Posle malo računanja, nalazimo

$$\begin{aligned} ||A+B|| &= \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||(A+B)x||_n}{||x||_m} \leq \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||Ax||_n + ||Bx||_n}{||x||_m} \\ &\leq \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||Ax||_n}{||x||_m} + \sup_{x\in\mathbb{R}^m\setminus\{0\}} \frac{||Bx||_n}{||x||_m} = ||A|| + ||B||. \end{aligned}$$

Kako je $||A|| \geq ||Ax||_n/||x||_m$, $x \in \mathbb{R}^m \setminus \{0\}$, to je $||Ax||_n \leq ||A||\,||x||_m$, $x \in \mathbb{R}^m$. Znači da je definisana norma matrica saglasna sa normom vektora.

Da dokažemo poslednju nejednakost, izaberimo proizvoljne $A, B \in \mathcal{M}_n$, onda je $||ABx||_n \leq ||AB||\,||x||_n$ i $||ABx||_n \leq ||A||\,||Bx||_n \leq ||A||\,||B||\,||x||_n$. Odavde dobijamo

$$||AB|| = \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{||ABx||_n}{||x||_n} \leq ||A||\,||B||,$$

što je i trebalo dokazati. □

Ovako definisana norma matrica, kao u jednakosti (2.1), naziva se još i subordinisana (potčinjena) norma matrica normama vektora $||\cdot||_n$ i $||\cdot||_m$. Ne moraju sve saglasne norme biti subordinisane. Primer je Frobeniusova norma definisana dalje u tekstu. Vrlo se jednostavno pokazuje da subordinisana norma jedinične matrice ima vrednost jedan. Naime, važi

$$||I|| = \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{||Ix||_n}{||x||_n} = \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{||x||_n}{||x||_n} = \sup_{x\in\mathbb{R}^n\setminus\{0\}} 1 = 1.$$

Subordinisana norma je uvek saglasna sa normom vektora, što se iz dokaza leme može videti.

Sa ovako definisanom normom matrica možemo definisati rastojanje na prostoru matrica $\mathcal{M}_{nm}$. Matrice $A, B \in \mathcal{M}_{nm}$ se nalaze na rastojanju $d(A, B) = ||A - B||$. Primetimo da je metod definisanja rastojanja ista kao i u slučaju vektora. Rastojanje matrica zadovoljava sve osobine koje zadovoljava i rastojanje vektora pobrojane u lemi 2.2. Takođe, moguće je definisati apsolutnu i relativnu grešku matrica. Neka je A' približna vrednost matrice A, onda su apsolutna i relativna greška date sa

$$\Delta = ||A' - A||, \quad r = \frac{||A' - A||}{||A||}.$$

Nama su od interesa norme vektora $||\cdot||_1$, $||\cdot||_2$ i $||\cdot||_{+\infty}$ i odgovarajuće subordinisane norme matrica. Ovde imamo na umu subordinisane norme matrica koje zadovoljavaju sledeće nejednakosti

$$||Ax||_p \leq ||A||\,||x||_p, \quad p = 1,\ p = 2,\ p = +\infty.$$

Lema 2.8 *Normi vektora $||\cdot||_1$ subordinisana je matrična norma $||\cdot||_1$ definisana sa*

$$||A||_1 = \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}|.$$

Normi vektora $||\cdot||_{+\infty}$ subordinisana je matrična norma $||\cdot||_{+\infty}$ definisana sa

$$||A||_{+\infty} = \max_{i=1,\dots,n} \sum_{j=1}^{m} |a_{ij}|.$$

Normi vektora $||\cdot||_2$ subordinisana je matrična norma $||\cdot||_2$ definisana sa

$$||A||_2 = \max\{\sqrt{\lambda} \mid \lambda \in \sigma(A^T A)\},$$

gde je $\sigma(A)$ skup svih sopstvenih vrednosti matrice A ili spektar matrice A. Matrična norma $||\cdot||_2$ naziva se spektralna norma matrice A.
Normi vektora $||\cdot||_2$ saglasna, ali ne i subordinisana, je matrična norma $||\cdot||_F$ definisana sa

$$||A||_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2 \right)^{1/2}.$$

Matrična norma $||\cdot||_F$ naziva se Frobeniusova norma.
Spektralni radijus $\rho(A)$ matrice $A \in \mathcal{M}_n$ definišemo kao maksimum modula svih sopstvenih vrednosti matrice A, $\rho(A) = \max\{|\lambda| \mid \lambda \in \sigma(A)\}$.
Neka je $||\cdot||$ bilo koja matrična norma, onda za svaku matricu $A \in \mathcal{M}_n$ važi $\rho(A) \leq ||A||$.

Dokaz. Neka je $x \in \mathbb{R}^m \setminus \{0\}$, onda je

$$\frac{||Ax||_1}{||x||_1} = \frac{1}{||x||_1} \sum_{i=1}^{n} \left| \sum_{j=1}^{m} a_{ij} x_j \right| \leq \frac{1}{||x||_1} \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|\,|x_j| = \frac{1}{||x||_1} \sum_{j=1}^{m} |x_j| \sum_{i=1}^{n} |a_{ij}|$$
$$\leq \frac{1}{||x||_1} \sum_{j=1}^{m} |x_j| \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}| = \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}|.$$

Očigledno je $||A||_1 \leq \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}|$. Da bismo dokazali da je u pitanju jednakost pretpostavimo da se maksimum dostiže za $j = \ell$. Dakle, neka je

$$\max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}| = \sum_{i=1}^{n} |a_{i\ell}|.$$

Ako izaberemo $x = e_\ell$, gde je e_ℓ vektor koji ima sve koordinate jednake nula izuzev ℓ-te koja ima vrednost jedan, dobijamo

$$||A||_1 \geq \frac{||Ae_\ell||_1}{||e_\ell||_1} = ||(a_{1\ell}, \dots, a_{n\ell})||_1 = \sum_{i=1}^{n} |a_{i\ell}| = \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}| \geq ||A||_1.$$

Zaključujemo da važi $||A||_1 = \max_{j=1,\dots,m} \sum_{i=1}^{n} |a_{ij}|$. Vidimo da je matrična norma $||\cdot||_1$ subordinisana normi vektora $||\cdot||_1$.

Posmatrajmo sada normu vektora $||\cdot||_{+\infty}$. U ovom slučaju, imamo

$$\frac{||Ax||_{+\infty}}{||x||_{+\infty}} = \frac{1}{||x||_{+\infty}} \max_{i=1,\dots,n} \left| \sum_{j=1}^{m} a_{ij}x_j \right| \leq \frac{1}{||x||_{+\infty}} \max_{i=1,\dots,n} \sum_{j=1}^{m} |a_{ij}|\,|x_j|$$
$$\leq \frac{1}{||x||_{+\infty}} \max_{i=1,\dots,n} \sum_{j=1}^{m} |a_{ij}| \max_{j=1,\dots,m} |x_j| = \max_{i=1,\dots,n} \sum_{j=1}^{m} |a_{ij}|.$$

Kao i u prethodnom slučaju dokazali smo da važi $||A||_{+\infty} \leq \max_{j=1,\dots,n} \sum_{j=1}^{m} |a_{ij}|$. Za dokaz obrnute nejednakosti, može se iskoristiti vektor $x = (a_{\ell,1}/|a_{\ell,1}|, \dots, a_{\ell,m}/|a_{\ell,m}|)$, gde je ℓ indeks za koji se postiže maksimum u izrazu za normu.

Dokaz, da je matrična norma $||\cdot||_2$ subordinisana normi vektora $||\cdot||_2$, se uglavnom daje koristeći osobine simetričnih matrica i skalarnih proizvoda. Međutim, dokaz nije elementaran pa ga nećemo navoditi.

Saglasnost Frobeniusove norme sa Euklidovom normom nećemo dokazivati. Pokažimo samo da Frobeniusova norma nije subordinisana nijednoj normi vektora na skupu kvadratnih matrica. Ovaj deo dokaza nije komplikovan. Dovoljno je samo odrediti normu jedinične matrice

$$||I||_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \delta_{ij}^2 \right)^{1/2} = n^{1/2} \neq 1.$$

Na osnovu osobina subordinisane norme zaključujemo da Frobeniusova norma nije subordinisana nijednoj normi vektora.

Deo tvrđenja o spektralnom radijusu je, za razliku od prethodnog dela tvrđenja, jednostavan i može se dati relativno elementarnim metodima. Naime, neka je $\lambda \in \sigma(A)$, onda postoji $x \in \mathbb{C}^n$, tako da važi $Ax = \lambda x$. Međutim,

$$|\lambda|\,||x|| = ||\lambda x|| = ||Ax|| \leq ||A||\,||x||,$$

odakle zaključujemo da važi $|\lambda| \leq ||A||$, za svako $\lambda \in \sigma(A)$, pa mora važiti i za $\rho(A) = \max|\lambda|$.[2]□

S obzirom da je matrična norma $||\cdot||_2$ subordinisana normi vektora $||\cdot||_2$, dok je matrična norma $||\cdot||_F$ samo saglasna, važi $||A||_2 \leq ||A||_F$.

Između matričnih normi se mogu uspostaviti nejednakosti koje mogu biti od koristi. Najvažnije nejednakosti su prezentovane u sledećoj lemi.

Lema 2.9 *Neka je $A \in \mathcal{M}_{nm}$, onda je:*

1° $||A||_2 \leq ||A||_F \leq \sqrt{m}||A||_2$,

2° $||A||_2 \leq \sqrt{||A||_1||A||_{+\infty}}$,

3° $\frac{1}{\sqrt{m}}||A||_{+\infty} \leq ||A||_2 \leq \sqrt{n}||A||_{+\infty}$,

4° $\frac{1}{\sqrt{n}}||A||_1 \leq ||A||_2 \leq \sqrt{m}||A||_1$.

[2]Za dokaz ovog dela tvrđenja smo koristili vektore sa kompleksnim elementima. Na početku glave sva razmatranja smo ograničili na vektore sa realnim elementima. Može se pokazati, u šta ovde nećemo ulaziti, da osobine normi koje se koriste u ovom dokazu važe i u ovom opštijem slučaju.

Ovu lemu nećemo dokazivati, ali ova lema omogućava da shvatimo da, kao i kod vektora, aproksimacija neke matrice drugom matricom ne zavisi od izabrane norme koja se koristi za određivanje relativne ili apsolutne greške[3].

Dajemo primer izračunavanja normi matrica. Neka je matrica A zadata na sledeći način

$$A = \begin{bmatrix} 1 & -2 & -3 \\ -2 & 0 & -3 \\ 1 & -3 & -2 \end{bmatrix}.$$

Odredimo $||\cdot||_1$, $||\cdot||_{+\infty}$ i $||\cdot||_F$ norme matrice A.

$$\begin{aligned} ||A||_1 &= \max\{|1|+|-2|+|1|, |-2|+|0|+|-3|, |-3|+|-3|+|-2|\} \\ &= \max\{4,5,8\} = 8, \\ ||A||_F &= \left(1^2+(-2)^2+(-3)^2+(-2)^2+0^2+(-3)^2+1^2+(-3)^2+(-2)^2\right)^{1/2} \\ &= \sqrt{41} \approx 6.403, \\ ||A||_{+\infty} &= \max\{|1|+|-2|+|-3|, |-2|+|0|+|-3|, |1|+|-3|+|-2|\} \\ &= \max\{6,5,6\} = 6. \end{aligned}$$

Da bismo odredili $||\cdot||_2$ normu moramo odrediti sosptvene vrednosti matrice

$$A^TA = \begin{bmatrix} 6 & -5 & 1 \\ -5 & 13 & 12 \\ 1 & 12 & 22 \end{bmatrix}.$$

Karakteristični polinom je

$$p(x) = \det(A^TA - xI) = -x^3 + 41x^2 - 326x + 169.$$

Nule karakterističnog polinoma su sopstvene vrednosti matrice A^TA. Dobijamo

$$x_1 \approx 30.49, \quad x_2 \approx 9.953, \quad x_3 \approx .5569.$$

Zaključujemo da važi $||A||_2 \approx \sqrt{30.49} \approx 5.522$.

Dok je norme $||\cdot||_1$, $||\cdot||_{+\infty}$ i $||\cdot||_F$ relativno jednostavno odrediti, normu $||\cdot||_2$ nije. Normu $||\cdot||_2$ je relativno jednostavno odrediti u slučaju kad je matrica A simetrična. U ovom slučaju važi

$$||A||_2 = \max\{|\lambda| \,|\, \lambda \in \sigma(A)\}.$$

Međutim, norme se relativno jednostavno određuju koristeći `Matlab`.

2.1.3 Izračunavanje normi vektora i matrica u Matlabu

Određivanje normi vektora i matrica u `Matlab`u se postiže koristeći funkciju `norm`. Funkcija prihvata dva argumenta. Prvi argument je vektor ili matrica čiju normu treba odrediti, a drugi je vrsta norme koju treba odrediti. Moguće vrednosti drugog argumenta su `1`, `2`, `inf` ili `'fro'` u slučaju kad je argument matrica.

Za prethodni primer mogli bismo da izračunamo tražene norme na sledeći način

[3]Kao i u slučaju vektora, ovde se pre svega misli da ako niz matrica $A_k \in \mathcal{M}_{nm}$, $k \in \mathbb{N}$, konvergira u nekoj normi ka matrici A, onda on konvergira u svim normama ka matrici A. Drugim rečima, kao i u slučaju vektora, sve norme na prostoru matrica su ekvivalentne.

```
>>A=[1 -2 -3; -2 0 -3; 1 -3 -2];
>>norm(A,1)
ans =
     8
>>norm(A,'fro')
ans =
   6.403124237432849
>>norm(A,2)
ans =
   5.521742294608997
>>norm(A)
ans =
   5.521742294608997
>>norm(A,inf)
ans =
     6
```

Vidimo da podrazumevana vrednost drugog argumenta iznosi 2.

U slučaju da je prvi argument vektor, drugi argument može biti bilo koji realni broj p. Za proizvoljnu realnu vrednost p drugog argumenta, funkcija izračunava vrednost izraza

$$\left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p},$$

gde je vektor x prvi argument. Naglašavamo da realna vrednost p ne mora biti veća ili jednaka jedan. Dakle prihvatljive su i, recimo, negativne vrednosti. U slučaju da je drugi argument veći ili jednak jedan funkcija računa odgovarajuću normu vektora. Moguće je da drugi argument ima vrednost `inf`, u kom slučaju funkcija računa normu vektora $||\cdot||_{+\infty}$. Ako je drugi argument `-inf`, funkcija vraća minimum apsolutnih vrednosti koordinata vektora. U slučaju vektora, vrednost drugog argumenta `'fro'` je ekvivalentna vrednosti 2. Podrazumevana vrednost drugog argumenta je 2. Evo nekoliko primera.

```
>>b=[1 -2 3];
>>norm(b,1)
ans =
     6
>>norm(b,inf)
ans =
     3
>>norm(b,'fro')
ans =
   3.741657386773941
>>norm(b,2)
ans =
   3.741657386773941
>>norm(b)
ans =
   3.741657386773941
```

```
>>norm(b,-inf)
ans =
    1
>>norm(b,-3)
ans =
   0.951174086294281
>>norm(b,0)
ans =
    inf
```

2.2 Direktni metodi

U ovom odeljku izučavamo direktne metode za rešavanje kvadratnih sistema linearnih jednačina. Sem ako drugačije nije naglašeno razmatramo sisteme jednačina sa regularnom matricom sistema. Drugim rečima, razmatramo sisteme jednačina sa jedinstvenim rešenjem.

2.2.1 Trougaoni sistemi linearnih jednačina

Gornje trougaoni sistemi linearnih jednačina imaju sledeću formu

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{nn}x_n &= b_n. \end{aligned}$$

Kao što vidimo matrica sistema je gornje trougaona, dakle, ima nule ispod glavne dijagonale

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}.$$

Donje trougaoni sistem jednačina je sličan prethodno opisanom sistemu jednačina. Razlika je što je matrica sistema donje trougaona, pa su nule raspoređene iznad glavne dijagonale.

Regularnost matrice sistema kod trougaonog sistema linearnih jednačina se svodi na uslov $a_{ii} \neq 0$, $i = 1, \dots, n$, jer je $\det(A) = \prod_{i=1}^{n} a_{ii}$.

Postoji vrlo efikasan algoritam za rešavanje ovakvog sistema linearnih jednačina. Naime, iz poslednje jednačine sistema možemo izračunati

$$x_n = \frac{b_n}{a_{nn}}.$$

Kad smo izračunali x_n, možemo iskoristiti dobijenu vrednost promenljive x_n, i dobiti novi sistem

linearnih jednačina

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,n-1}x_{n-1} &= b_1 - a_{1,n}x_n, \\ a_{22}x_2 + \cdots + a_{2,n-1}x_{n-1} &= b_2 - a_{2,n}x_n, \\ &\vdots \\ a_{n-1,n-1}x_{n-1} &= b_{n-1} - a_{n-1,n}x_n. \end{aligned}$$

Vidimo da novodobijeni sistem jednačina ima istu formu, pa opisani postupak možemo ponoviti još jednom, da dobijemo vrednost promenljive x_{n-1}. Proces ponavljamo rekurzivno sve dok ne izračunamo x_1. Ovaj metod je poznat kao zamena unazad (engleski backward substitution).

Proces možemo opisati algoritmom na sledeći način

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

Kao što vidimo dovoljne su dve `for` petlje da opišu program za rešenje gornje trougaonog sistema linearnih jednačina.

U slučaju da je sistem jednačina donje trougaoni, prvo bismo računali vrednost promenljive x_1, zatim x_2, i tako redom do promenljive x_n. Ovaj proces se naziva eliminacija unapred (engleski forward elimination). U ovom slučaju algoritam bi izgledao ovako

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

Funkcija `trougaoniSistemR` implementira algoritam zamene unazad i eliminacije unapred.

```
function [x,res]=trougaoniSistemR(A,b,smer)
%TRIOUGAONISISTEMR(A,B,SMER) resava trougaoni sistem
% linearnih jednacina sa matricom sistema A, slobodnim vektorom b,
% vrednost argumenta smer 'unapred' ili 'unazad',
% odredjuje da li je sistem donje ili gornje trougaoni, redom

    if nargin == 2
        smer = 'unapred'
    end
    n = length(b);x=zeros(1,n);
    if strcmp(smer,'unapred')
        x(1) = b(1)/A(1,1);
        for k=2:n
            x(k) = (b(k)-dot(x(1:k-1),A(k,1:k-1)))/A(k,k);
        end
    elseif strcmp(smer,'unazad')
        x(end) = [ b(n)/A(n,n) ];
        for k=n-1:-1:1
            x(k) = (b(k)-dot(x(k+1:n),A(k,k+1:n)))/A(k,k);
        end
```

```
    else
        error('nepoznat smer');
    end
    x = x';
    res = A*x-b;
end
```

Argument `smer` određuje eliminaciju unapred, ako ima vrednost `'unapred'`, ili zamenu unazad ako ima vrednost `'unazad'`. Prvi argument `A`, funkcije `trougaoniSistemR`, predstavlja matricu sistema, dok je drugi argument `b` slobodni vektor sistema linearnih jednačina. Funkcija ima dva izlazna argumenta. Izlazni argument `x` predstavlja približno rešenje sistema. Kažemo približno rešenje sistema jednačina, jer bismo rešenje sistema dobili u slučaju da izračunavanja izvodimo u aritmetici beskonačne preciznosti. Međutim, kako mi radimo uvek sa aritmetikom čija je tačnost ograničena, ako ne tačnošću ulaznih podataka onda barem mašinskom greškom, to ne možemo očekivati da veličina `res`, koja na izvestan način može da posluži za kontrolu greške, uvek ima vrednost jednaku nula.

Evo jednog jednostavnog primera koji pokazuje kako funkcija `trougaoniSistemR` može da pogreši.

```
>>[x,res]=trougaoniSistemR([1 1 0; 0 10^(-5) 1; 0 0 1],
[ 1 1+10^(-5) 1]','unazad')
x =
  -0.000000000006551
   1.000000000006551
   1.000000000000000
res =
     0
     0
     0
>>As=sym('[ 1 1 0; 0 10^(-5) 1; 0 0 1]');
>>bs=sym('[ 1; 1+10^(-5); 1]');
>>xTacno=double(As\bs)
xTacno =
     0
     1
     1
>>norm(xTacno-x,2)/norm(xTacno,2)
ans =
    6.551204023708124e-012
```

Rešenje koje dobijamo upotrebom funkcije `trouganiSistemR` je očigledno netačno, što se vidi u drugom delu gde rešenje izračunavamo egzaktno. Vidimo i da je vrednost izlaznog argumenta `res` jednaka nuli, iako je rešenje pogrešno. Učinjena greška nije velika, relativna greška je reda veličine $7 \cdot 10^{-12}$. Međutim, dovoljno je recimo promeniti sva pojavljivanja broja 10^{-5} u matrici sistema i slobodnom vektoru, na recimo 10^{-10}, i relativna greška raste na $8 \cdot 10^{-8}$. Daljim smanjenjem može se dobiti po volji velika relativna greška u rešenju. Ako smanjimo na 10^{-15} dobićemo relativnu grešku reda veličine 10^{-1}.

Da bismo na neki način bili u stanju da ocenimo 'najgori' mogući slučaj koji se može javiti prilikom rešavanja trougaonih sistema linearnih jednačina pomenućemo sledeću teoremu.

Teorema 2.1 *Neka je $A = [a_{ij}] \in \mathcal{M}_n$ donja (gornja) trougaona matrica i neka je $Ax = b$ sistem linearnih jednačina. Ako je x' rešenje sistema dobijeno, metodom eliminacije unapred (zamene unazad), na računskoj mašini koja ima mašinsku preciznost ε, onda postoji matrica $\Delta A = [\Delta a_{ij}]$ tako da važi*

$$(A + \Delta A)x' = b, \quad |\Delta a_{ij}| \leq n\varepsilon|a_{ij}| + O(\varepsilon^2).$$

Teoremu treba shvatiti na sledeći način. Ako sistem jednačina rešimo na računskoj mašini koja ima mašinsku tačnost ε, onda, dobijamo rešenje sistema koji se malo razlikuje od stvarnog sistema. Ta mala razlika je upravo matrica ΔA, a veličina te male razlike je izmerena vrednošću elemenata matrice $|\Delta a_{ij}|$, koja je ocenjena gornjom granicom apsolutne greške $n\varepsilon|a_{ij}|$. Ako radimo u formatu dvostruke preciznosti, onda je $\varepsilon = 2^{-52}$. Pretpostavimo da na raspolaganju imamo četiri giga bajta rama, to nam omogućava smeštanje matrice reda $n = \sqrt{2^{32}/2^3} \approx 2^{15} = 32768$, što daje ocenu $n\varepsilon = 2^{15} \cdot 2^{-52} \approx 7.3 \cdot 10^{-12}$. Ako se ograničimo na sisteme čije se matrice mogu reprezentovati u memoriji današnjih računara, rešavanjem sistema dobijamo, u stvari, rešenja sistema čije se matrice sistema razlikuju od originalnih za matricu, čiji su elementi zadati sa gornjom granicom relativne greške od približno $7.3 \cdot 10^{-12}$.

Primetimo da ova ocena ne znači mnogo. Na primer, bez obzira što se elementi matrica sistema koji rešavamo i sistema čije smo rešenje dobili malo razlikuju, relativna greška rešenja može biti po volji velika. Ovo je jasno pokazano u prethodnom primeru.

2.2.2 Gaussova eliminacija

Inspirisan lakoćom kojom se rešavaju trougaoni sistemi linearnih jednačina, Gauss je pronašao sistematski postupak kojim se svaki kvadratni sistem linearnih jednačina može svesti na trougaoni. Postupak se naziva Gaussova eliminacija. Ideja koju je razvio Gauss je poslužila kao osnova za formulaciju Kronecker-Cappelijeve teoreme. Nama je od najveće važnosti jer omogućava formulisanje algoritma za rešavanje sistema linearnih jednačina koristeći već postojeći algoritam za rešavanje trougaonih sistema linearnih jednačina.

Posmatrajmo sistem linearnih jednačina

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2, \\ & & & & & & & \vdots & \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n. \end{array}$$

Ideja je eliminacija promenljive x_1 iz svih jednačina izuzev iz prve, jer očigledno trougaoni sistem linearnih jednačina baš tako izgleda. Eliminaciju promenljive x_1 iz svih jednačina izuzev iz prve možemo postići koristeći samo množenje prve jednačine brojem i dodavanjem ostalim jednačinama. Na primer, ako prvu jednačinu pomnožimo sa $m_{i1} = -a_{i1}/a_{11}$ i dodamo i-toj jednačini dobijamo sledeći sistem jednačina

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1, \\ & & a_{22}^{(1)}x_2 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)}, \\ & & & & & & & \vdots & \\ & & a_{n2}^{(1)}x_2 & + & \cdots & + & a_{nn}^{(1)}x_n & = & b_n^{(1)}, \end{array}$$

gde smo sa $a_{ij}^{(1)}$ označili koeficijente koje dobijamo posle dodavanja prve jednačine pomnožene odgovarajućim faktorom

$$a_{ij}^{(1)} = a_{ij} + m_{i1}a_{ij}, \quad i, j = 2, \ldots, n.$$

Na ovaj način smo eliminisali promenljivu x_1 iz svih jednačina sistema linearnih jednačina. Ova operacija eliminacije se može opisati i matričnim množenjem. Definišemo li matricu

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ m_{21} & 1 & 0 & \dots & 0 \\ m_{31} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & 0 & 0 & \dots & 1 \end{bmatrix} = I + m_1 e_1^T,$$

gde je $m_1 = (0, m_{21}, m_{31}, \dots, m_{n1})$ i $e_1 = (1, 0, \dots, 0)$, možemo proveriti da važi

$$M_1 A = A_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix}.$$

Opisani postupak možemo primeniti na novodobijeni sistem jednačina i promenljivu x_2. Kad eliminišemo promenljivu x_2, proces nastavljamo eliminacijom promenljive x_3, i tako redom. Na kraju dobijamo trougaoni sistem linearnih jednačina. Eliminaciju promenljive x_2 dobijamo tako što faktorom $m_{i2} = -a_{i2}^{(1)}/a_{22}^{(1)}$ množimo drugu jednačinu i dodajemo je i-toj. Kao rezultat dobijamo sistem jednačina

$$\begin{array}{rcrcrcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1, \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)}, \\ & & & & a_{33}^{(2)}x_3 & + & \cdots & + & a_{3n}^{(2)}x_n & = & b_3^{(2)}, \\ & & & & \vdots & & & & & & \vdots \\ & & & & a_{n3}^{(2)}x_3 & + & \cdots & + & a_{nn}^{(2)}x_n & = & b_n^{(2)}. \end{array}$$

Dobijanje ovog sistema jednačina iz pethodnog takođe možemo opisati upotrebom matrica. Ako definišemo matricu

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & m_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & m_{n2} & 0 & \dots & 0 \end{bmatrix} = I + m_2 e_2^T,$$

gde je $m_2 = (0, 0, m_{32}, \dots, m_{n2})$ i $e_2 = (0, 1, 0, \dots, 0)$, onda je $A_2 = M_2 A_1 = M_2 M_1 A$, gde je A_2 matrica sistema jednačina sa eliminisanim promenljivama x_1 i x_2.

Nastavljajući postupak dobijamo trougaoni sistem jednačina

$$\begin{array}{rcrcrcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1, \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)}, \\ & & & & a_{33}^{(2)}x_3 & + & \cdots & + & a_{3n}^{(2)}x_n & = & b_3^{(2)}, \\ & & & & & & \ddots & & & & \vdots \\ & & & & & & & & a_{nn}^{(n-1)}x_n & = & b_n^{(n-1)}, \end{array}$$

sa trougaonom matricom sistema koju obeležavamo sa U. Matrica U se dobija od matrice sistema A množeći matricama M_i, $i = 1, \ldots, n-1$, na sledeći način

$$U = M_{n-1} \cdots M_1 A.$$

Matrice M_i, $i = 1, \ldots, n-1$, su definisane sa

$$M_i = I + m_i e_i^T,$$

gde je $m_i = (0, \ldots, 0, m_{i+1,i}, \ldots, m_{ni})$, $i = 1, \ldots, n-1$, a e_i, $i = 1, \ldots, n-1$, su vektori kolone koji su ispunjeni nulama izuzev i-te koordinate gde imaju vrednost jednaku 1. Vektori e_i, $i = 1, \ldots, n$, su poznati kao prirodna baza prostora $\mathbb{R}^n$. Koeficijente m_{ij} možemo dobiti na sledeći način

$$m_{ij} = -a_{ij}^{(j-1)} / a_{jj}^{(j-1)}, \quad i = j+1, \ldots, n, \quad j = 1, \ldots, n-1,$$

pri čemu koristimo konvenciju $a_{ij}^{(0)} = a_{ij}$, $i, j = 1, \ldots, n$.

Primetimo da se prilikom Gaussove eliminacije menja i slobodni vektor sistema b. Međutim, promena slobodnog vektora sistema prati promenu matrice sistema A i ovu promenu možemo da opišemo na isti način

$$b^{(n-1)} = M_{n-1} \cdots M_1 b.$$

Algoritam Gaussove eliminacije možemo sažeto opisati na sledeći način

$$\left.\begin{array}{l} m_{ij} = -a_{ij}^{(j-1)} / a_{jj}^{(j-1)} \\ b_i^{(j)} = b_i^{(j-1)} + m_{ij} b_j^{(j-1)} \\ a_{ik}^{(j)} = a_{ik}^{(j-1)} + m_{ij} a_{jk}^{(j-1)} \end{array}\right\} k = j+1, \ldots, n \Bigg\} i = j+1, \ldots, n \Bigg\} j = 1, \ldots, n-1.$$

Broj operacija potreban za izvršenje algoritma Gaussove eliminacije je reda veličine n^3, jer kao što vidimo sadrži tri `for` petlje.

Posmatrajmo sledeći primer linearnog sistema jednačina

$$\begin{array}{rcrcrcrcl} x_1 & + & 2x_2 & - & x_3 & + & x_4 & = & 1, \\ 2x_1 & + & 5x_2 & - & x_3 & + & 2x_4 & = & 1, \\ 3x_1 & - & x_2 & - & 2x_3 & + & x_4 & = & 2, \\ x_1 & - & x_2 & + & 3x_3 & - & 5x_4 & = & 2, \end{array}$$

i na ovom primeru ilustrujmo sve prethodne koncepte. Tradicionalno, prvu jednačinu množimo sa 2 i oduzimamo od druge, da bismo iz druge jednačine eliminisali x_1. Zatim, prvu jednačinu množimo sa 3 i oduzimamo od treće da bismo iz treće jednačine eliminisali x_1. Konačno prosto oduzmemo prvu jednačinu od četvrte da bismo u četvrtoj jednačini eliminisali x_1. Mi ćemo se poslužiti naprednijim načinom izlaganja koji koristi matričnu notaciju.

Matrica M_1, zbog $m_{21} = -a_{21}/a_{11} = -2$, $m_{31} = -a_{31}/a_{11} = -3$, $m_{41} = -a_{41}/a_{11} = -1$, izgleda ovako

$$M_1 = I + m_1 e_1^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ -3 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}.$$

Ako pomnožimo sistem jednačina matricom M_1, dobijamo

$$M_1Ax = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & -1 & 1 \\ 2 & 5 & -1 & 2 \\ 3 & -1 & -2 & 1 \\ 1 & -1 & 3 & -5 \end{bmatrix} x = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -7 & 1 & -2 \\ 0 & -3 & 4 & -6 \end{bmatrix} x$$
$$= \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = M_1 b = b^{(1)}.$$

Kao što vidimo, dobijamo sistem jednačina gde je promenljiva x_1 eliminisana iz svih jednačina izuzev iz prve. Matrica ovog sistema je u tekstu obeležena matricom A_1. Slobodni vektor sistema se menja na isti način i dobijamo vektor koji je u tekstu označen sa $b^{(1)}$.

U sledećem koraku formiramo sistem jednačina u kome je eliminisana promenljiva x_2. Matrica M_2, zbog $m_{32} = -(-7)/1 = 7$, $m_{42} = -(-3)/1 = 3$, iznosi

$$M_2 = I + m_2 e_2^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 7 \\ 3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 7 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{bmatrix}.$$

Sistem jednačina posle eliminacije x_2 postaje

$$M_2A_1x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 7 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -7 & 1 & -2 \\ 0 & -3 & 4 & -6 \end{bmatrix} x = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 8 & -2 \\ 0 & 0 & 7 & -6 \end{bmatrix} x$$
$$= \begin{bmatrix} 1 \\ -1 \\ -8 \\ -2 \end{bmatrix} = M_2 b^{(1)} = b^{(2)}.$$

Dobijamo sistem jednačina sa eliminisanom promenljivom x_2 u svim jednačinama izuzev prve i druge. U sledećem koraku eliminišemo promenljivu x_3 iz četvrte jednačine. Matrica M_3, zbog $m_{43} = -7/8$, iznosi

$$M_3 = I + m_3 e_3^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -7/8 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -7/8 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -7/8 & 1 \end{bmatrix}.$$

Eliminacija promenljive x_3 se postiže množenjem sistema jednačina matricom M_3. Dobijamo

$$M_3A_2x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -7/8 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 8 & -2 \\ 0 & 0 & 7 & -6 \end{bmatrix} x = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 8 & -2 \\ 0 & 0 & 0 & -17/4 \end{bmatrix} x$$
$$= \begin{bmatrix} 1 \\ -1 \\ -8 \\ 5 \end{bmatrix} = M_3b^{(2)} = b^{(3)}.$$

Dobijamo gornje trougaoni sistem linearnih jednačina sa matricom sistema A_3. Sistem je sada jednostavno rešiti koristeći zamenu unazad kako je opisano u prethodnom odeljku.

Rešenje sistema jednačina iznosi $17x = (5, 5, -22, 20)$.

Primetimo da regularnost matrice sistema ne garantuje valjanost uslova $a_{ii}^{(i-1)} \neq 0$, $i = 1, \ldots, n-1$. U odeljku 2.2.4 razmatraćemo kako se ovaj uslov kod regularnih matrica može obezbediti.

Funkcija koja implementira Gaussov algoritam može biti data na sledeći način

```
function [x,L,U,res]=gaussovaEliminacijaR(A,b)
%GAUSSOVAELIMINACIJAR(A,B) izracunava resenje sistema linearnih
% jednacina AX=B Gaussovim metodom eliminacije,
% izlazni argumenti resenje x, i donja trougaona matrica L
% i gornja trougaona matrica U, tako da vazi A=L*U,
% res=A*x-b sluzi za kontrolu greske

n=length(b); L=eye(n); U=A; bb=b;
for j=1:n-1
    for i=j+1:n
        L(i,j)=+U(i,j)/U(j,j);
        bb(i)=bb(i)-L(i,j)*bb(j);
        for k=j:n
            U(i,k)=U(i,k)-L(i,j)*U(j,k);
        end
    end
end
[x,res]=trougaoniSistemR(U,bb,'unazad');
res=A*x-b;
end
```

Sledeći niz komandi ilustruje primenu funkcije `guassovaEliminacijaR`[4] na rešavanje prethodnog sistema linearnih jednačina.

```
>>format long g
>>A=[1 2 -1 1; 2 5 -1 2; 3 -1 -1 1; 1 -1 3 -5]; b=[1 1 2 2]';
```

[4]Ova implementacija nije najefikasnija po pitanju upotrebljene memorije. Obično se matrice L i U smeštaju u istu matricu, jer je matrica L donje trougaona sa jedinicama na glavnoj dijagonali, dok je matrica U gornje trougaona. Ovde smo se odlučili za posebne matrice L i U iz razloga preglednosti programa.

```
>>[x,L,U,res]=gaussovaEliminacijaR(A,b)
x =
       0.65
       0.1
      -1.1
      -0.95
L =
       1                    0                    0                    0
       2                    1                    0                    0
       3                   -7                    1                    0
       1                   -3   0.777777777777778                    1
U =
       1                    2                   -1                    1
       0                    1                    1                    0
       0                    0                    9                   -2
       0                    0                    0   -4.44444444444444
res =
                           0
                           0
     -6.66133814775094e-16
     -8.88178419700125e-16
>>L*U-A
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Primetimo da funkcija vraća dve matrice, matricu L i matricu U. Matrica U je već opisana, dok je kao što jednostavno proveravamo matrica L konstruisana od vrednosti $-m_{ij}$, $i = j+1, \ldots, n$, $j = 1, \ldots, n$, gde su m_{ij}, $i = j+1, \ldots, n$, $j = 1, \ldots, n$, koeficijenti koji se koriste u algoritmu Gaussove eliminacije. Kao što vidimo matrice L i U imaju zanimljivu osobinu. Njihov proizvod je polazna matrica A. Sledeća sekcija je posvećena konstrukciji matrica L i U sa ovom osobinom.

2.2.3 LU faktorizacija

Zamislimo da je potrebno rešiti veliki broj linearnih jednačina kod kojih se menja samo slobodni vektor, dok matrica sistema ostaje nepromenjena. Na primer, ovakav slučaj imamo kad god se menja pobuda sistema dok sam opis sistema ostaje nepromenjen. U ovom slučaju nije racionalno za svaki pojedinačni sistem jednačina iznova vršiti proces eliminacije promenljivih, jer u suštini koraci koje prolazimo, izuzev za slobodni vektor, ostaju isti. Iz ovog razloga su uvedeni faktorizacioni metodi.

Lema 2.10 *Neka je m_i vektor takav da ima i-tu koordinatu jednaku nula. Matrice*

$$M_i = I + m_i e_i^T, \quad M_i^{-1} = I - m_i e_i^T,$$

su jedna drugoj inverzne.

Dokaz. Dokaz je jednostavan, i zasniva se na činjenici da je $e_i^T m_i = ||0||$, što je ispunjeno jer vektor m_i ima i-tu koordinatu jednaku nula, a vektor e_i sve sem i-te koordinate ima jednake nula. Imamo

$$\begin{aligned} M_i M_i^{-1} &= (I + m_i e_i^T)(I - m_i e_i^T) = I + m_i e_i^T - m_i e_i^T - m_i e_i^T m_i e_i^T = I - m_i (e_i^T m_i) e_i^T \\ &= I - m_i \, ||0|| \, e_i^T = I, \\ M_i^{-1} M_i &= (I - m_i e_i^T)(I + m_i e_i^T) = I - m_i e_i^T + m_i e_i^T - m_i (e_i^T m_i) e_i^T = I - m_i \, ||0|| \, e_i^T = I, \end{aligned}$$

čime smo završili dokaz leme. □

Kao što znamo, na osnovu prethodnog odeljka, Gaussovu eliminaciju možemo opisati koristeći matrice $M_i = I + m_i e_i^T$, $i = 1, \ldots, n-1$, na sledeći način

$$M_{n-1} \cdots M_1 A x = M_{n-1} \cdots M_1 b, \quad Ux = M_{n-1} \cdots M_1 b,$$

gde smo sa U označili gornju trougaonu matricu sistema koju dobijamo Gaussovom eliminacijom. Koristeći lemu 2.10, možemo polazni sistem jednačina napisati u obliku

$$M_1^{-1} \cdots M_{n-1}^{-1} U x = b, \quad LUx = b,$$

gde smo označili $L = M_1^{-1} \cdots M_{n-1}^{-1}$. Na ovaj način smo dobili faktorizaciju matrice sistema na matricu L i matricu U. Matrica U je gornje trougaona, dok je matrica L, kako ćemo pokazati, donje trougaona.

Lema 2.11 *Matrica L je donje trougaona, i važi*

$$L = I - \sum_{i=1}^{n-1} m_i e_i^T.$$

Dokaz. Koristeći lemu 2.10, znamo eksplicitni izraz za matrice M_i^{-1}, $i = 1, \ldots, n-1$. Kako znamo reprezentaciju matrice L preko matrica M_i, $i = 1, \ldots, n-1$, možemo direktno da odredimo vrednost matrice L. Izraz za matricu L je sledeći

$$L = M_1^{-1} \cdots M_{n-1}^{-1} = (I - m_1 e_1^T) \cdots (I - m_{n-1} e_{n-1}^T).$$

Pre nego što odredimo osobine matrice L, pokažimo prvo jednu osobinu matrica $m_i e_i^T$. Naime, izračunajmo proizvod $m_i e_i^T m_j e_j^T$. Za $i < j$, važi

$$m_i e_i^T m_j e_j^T = m_i \, ||0|| \, e_j^T = 0.$$

Vrednost matrice je nula jer je proizvod $e_i^T m_j = ||0||$, jer vektor m_j ima i-tu koordinatu jednaku nula, dok vektor e_i ima sve sem i-te koordinate jednake nula. Posmatrajmo poizvod koji definiše matricu L, imamo

$$\begin{aligned} L &= (I - m_1 e_1^T)(I - m_2 e_2^T) \cdots (I - m_{n-1} e_{n-1}^T) \\ &= (I - m_1 e_1^T - m_2 e_2^T + m_1 (e_1^T m_2) e_2^T)(I - m_3 e_3^T) \cdots (I - m_{n-1} e_{n-1}^T) \\ &= (I - m_1 e_1^T - m_2 e_2^T)(I - m_3 e_3^T) \cdots (I - m_{n-1} e_{n-1}^T) \\ &= (I - m_1 e_1^T - m_2 e_2^T - m_3 e_3^T - (m_1 e_1^T - m_2 e_2^T) m_3 e_3^T) \cdots (I - m_{n-1} e_{n-1}^T) \\ &= (I - m_1 e_1^T - m_2 e_2^T - m_3 e_3^T)(I - m_4 e_4^T) \cdots (I - m_{n-1} e_{n-1}^T) = \cdots \\ &= I - \sum_{i=1}^{n-1} m_i e_i^T. \end{aligned}$$

Proces nastavljamo dok ima članova u proizvodu. Očigledno se članovi u proizvodu poništavaju prilikom množenja i doprinose samo sabircima.

Na osnovu ovog izraza moguće je odrediti u potpunosti matricu L. Naime, svaki sabirak $m_i e_i^T$, što se videlo iz primera u prethodnom odeljku, doprinosi matrici L u i-toj koloni i to upravo vektorom $-m_i$, tako da matrica L ima sledeću formu

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -m_{21} & 1 & 0 & \dots & 0 \\ -m_{31} & -m_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & -m_{n2} & -m_{n3} & \dots & 1 \end{bmatrix}.$$

Vidi se da je matrica L donje trougaona. □

Na prethodno dokazanoj lemi se zasniva metod LU faktorizacije. Naime, prilikom izvođenja Gaussove eliminacije možemo umesto da izvršavamo operacije nad slobodnim vektorom operacije izvršavati samo nad matricom sistema. U tom slučaju možemo matricu sistema faktorisati u obliku $A = LU$, pri čemu polazni sistem jednačina postaje $LUx = b$, gde je matrica L donje trougaona, a matrica U gornje trougaona. Ovako faktorisan sistem linearnih jednačina može se rešiti sukcesivnim rešavanjem dva trougaona sistema linearnih jednačina

$$Ly = b, \quad Ux = y,$$

pri čemu se prvi sistem jednačina rešava eliminacijom unapred, a drugi zamenom unazad. Jednom faktorisana matrica sistema može se koristiti za rešavanje po volji velikog broja sistema linearnih jednačina kojima se samo menja slobodni vektor sistema.

U primeru iz prethodnog odeljka matrica L iznosi

$$L = M_1^{-1} M_2^{-1} M_3^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & -7 & 1 & 0 \\ 1 & -3 & 17/4 & 1 \end{bmatrix}.$$

Rešanje sistema jednačina iz prethodnog odeljka LU faktorizacijom se svodi na rešavanje dva sistema linearnih jednačina sa trougaonom matricom sistema

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & -7 & 1 & 0 \\ 1 & -3 & 17/4 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 8 & -2 \\ 0 & 0 & 0 & -17/4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

Rešenje prvog sistema je $y = (1, -1, -8, 5)$, a rešenje drugog sistema je $17x = (5, 5, -22, -20)$, kako je dato i u prethodnom odeljku.

Ovakav metod konstrukcije LU faktorizacije proizvodi jediničnu dijagonalu u matrici L. Međutim, moguće je proizvesti i jediničnu dijagonalu u matrici U. Sve što je potrebno uraditi je primeniti LU faktorizaciju nad transponovanom matricom sistema. Primenom LU faktorizacije nad matricom A^T dobijamo matrice L i U sa jediničnom dijagonalom u matrici L, međutim, $A^T = LU$ i $A = (LU)^T = U^T L^T$. Transponovanjem donja trougaona matrica postaje gornja trougaona i gornja trougaona matrica postaje donja trougaona. Tako smo dobili LU faktorizaciju matrice koja ima jediničnu dijagonalu u matrici U. Ovo pokazuje da faktorizacija na proizvod gornje i donje

trougaone matrice ne mora biti jedinstvena. Međutim, faktorizacija jeste jedinstvena ako unapred zadamo jediničnu dijagonalu u gornjoj ili donjoj trougaonoj matrici.

Primetimo da pod uslovom regularnosti matrice, kao i u slučaju Gaussove eliminacije, ne moraju biti ispunjeni uslovi $a_{ii}^{(i-1)} \neq 0$, $i = 1, \ldots, n-1$, drugim rečima, regularnost matrice ne garantuje mogućnost izvršenja algoritma LU faktorizacije. Sledeći odeljak objašnjava da svaka regularna matrica ima LU faktorizaciju i kako do nje doći.

Sledeća teorema govori o grešci koja nastaje prilikom rešavanja sistema linearnih jednačina LU faktorizacijom.

Teorema 2.2 *Neka je $Ax = b$ i neka su $L = [\ell_{ij}]$ i $U = [u_{ij}]$ matrice koje se dobijaju algoritmom Gaussove eliminacije od matrice $A = [a_{ij}]$ na računskoj mašini mašinske preciznosti ε. Ako je x' rešenje sistema $Ux' = y$, $Ly = b$, onda je $(A + \Delta A)x' = b$, gde je*

$$|\Delta a_{ij}| \leq n\varepsilon \left(3|a_{ij}| + 5 \sum_{k=1}^{\min\{i,j\}} |\ell_{ik}|\,|u_{kj}| \right) + O(\varepsilon^2), \quad \Delta A = [\Delta a_{ij}].$$

2.2.4 Izbor glavnog elementa

Veoma jednostavan primer dovoljan je da ilustruje da izvršenje Gaussove eliminacije nije uvek moguće, iako je matrica sistema jednačina regularna. Na primer, posmatrajmo sistem jednačina

$$\begin{aligned} x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

Očigledno, nemoguće je eliminisati promenljivu x_1 iz druge jednačine koristeći prvu jednačinu, jer promenljiva x_1 u prvoj jednačini uopšte ne učestvuje. Drugim rečima, koeficijent a_{11} iznosi nula.

Međutim, ako malo proučimo sistem, vrlo jednostavno zapažamo da je sistem već zapisan u trougaonoj formi. Potrebno je samo razmeniti mesta jednačinama

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 &= 1. \end{aligned}$$

Ovakva razmena mesta jednačinama naziva se izbor glavnog elementa, a zasniva se na činjenici da se rešenje sistema linearnih jednačina ne menja ako jednačine sistema razmene mesta.

U opštem slučaju možemo zamisliti sledeći scenario. Izvršavamo algoritam Gaussove eliminacije i u koraku k, koji izgleda ovako

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & \ldots & + & a_{1k}x_k & + & \cdots & + & a_{1n}x_n & = & b_1, \\ & & \ddots & & \vdots & & & & & & \vdots \\ & & & & a_{kk}^{(k-1)}x_k & + & \cdots & + & a_{kn}^{(k-1)}x_n & = & b_k^{(k-1)}, \\ & & & & \vdots & & & & & & \vdots \\ & & & & a_{nk}^{(k-1)}x_k & + & \cdots & + & a_{nn}^{(k-1)}x_n & = & b_n^{(k-1)}, \end{array}$$

pronađemo da je $a_{kk}^{(k-1)} = 0$. Regularnost matrice sistema garantuje da je bar jedan od elemenata $a_{ik}^{(k-1)}$, $i = k, \ldots, n$, različit od nule. Pretpostavimo da je to element $a_{\ell k}^{(k-1)}$, $\ell > k$. Onda je dovoljno razmeniti k-tu i ℓ-tu jednačinu i nastaviti algoritam Gaussove eliminacije.

Ilustrujmo ovo jednim primerom. Posmatrajmo sledeći sistem jednačina

$$\begin{array}{rcrcrcl} x_1 & + & 2x_2 & + & x_3 & = & 1, \\ x_1 & + & 2x_2 & + & 2x_3 & = & 2, \\ x_2 & + & x_2 & + & x_3 & = & 3. \end{array}$$

Posle prve eliminacije dobijamo

$$A_1 = M_1 A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \quad b^{(1)} = M_1 b = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}.$$

Kao što vidimo dobili smo element $a_{22}^{(1)} = 0$. Da bismo nastavili proces eliminacije potrebno je razmeniti pozicije jednačina dva i tri. Ovu razmenu možemo postići koristeći permutacione matrice. Permutaciona matrica $P_{ij} = [p_{ij}]$ koja razmenjuje vrste i i j je istog oblika kao jedinična matrica, izuzev što su elementi $p_{ii} = p_{jj} = 0$ i $p_{ij} = p_{ji} = 1$. Za naš konkretan primer permutaciona matrica koja razmenjuje drugu i treću vrstu ima oblik

$$P_{23} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Dejstvom permutacione matrice na matricu A_1 i slobodni vektor $b^{(1)}$, dobijamo

$$P_{23} A_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad P_{23} b^{(1)} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

Postoji strategija vezana za izbor glavnog elementa koja se ne sastoji samo u izbegavanju nule. Potrebu strategije ilustrujemo na sledećem primeru.

Posmatrajmo sledeći sistem linearnih jednačina

$$\begin{array}{rcrcrcr} 10x_1 & - & 7x_2 & & & = & 7, \\ -3x_1 & + & 2.099999x_2 & + & 6x_3 & = & 3.900001, \\ 5x_1 & - & x_2 & + & 5x_3 & = & 6. \end{array}$$

Vrlo je jednostavno utvrditi da je rešenje sistema $x = (0, -1, 1)$. Međutim, rešavanjem sistema koristeći funkciju `gaussovaEliminacijaR`, dobijamo

```
>>A=single([10 -7 0; -3 2.099999 6; 5 -1 5]);
>>b=single([7; 3.900001; 6]);
>>format long g
>>[x,L,U,res]=gaussovaEliminacijaR(A,b)
x =
      -0.1400001
            -1.2
       0.9999999
L =
               1               0               0
            -0.3               1               0
```

```
              0.5         -2097152                1
U =
               10               -7                0
                0   -1.192093e-06                6
                0                0    1.258292e+07
res =
                0
                0
       -0.500001

>>A-L*U
ans =
     0     0     0
     0     0     0
     0     0     0
>>xTacno=A\b
xTacno =
    1.430512e-07
      -0.9999998
       0.9999999
```

Kao što vidimo, naša LU faktorizacija je u redu. Jedino što daje nagoveštaj da nešto sa rešenjem nije u redu je vrednost `res=A*x-b`, koja se kao što vidimo primetno razlikuje od nule. Vidimo takođe da operator levog deljenja u `Matlab`u proizvodi tačan rezultat, pri čemu operator levog deljenja radi sa jednostrukom preciznošću i sistem jednačina rešava koristeći Gaussovu eliminaciju. Jedina razlika je što operator levog deljenja koristi izbor glavnog elementa prilikom LU faktorizacije, odnosno, Gaussove eliminacije.

Sva izračunavanja smo izvodili u jednostrukoj preciznosti jer se efekti ne bi mogli prepoznati u dvostrukoj preciznosti. Da biste dobili izlaz kakav je prikazan neophodno je modifikovati inicijalizaciju matrice `L` u funkciji `gaussovaEliminacijaR`. Potrebno je naredbu `L=eye(n)` zameniti naredbom `L=single(eye(n))`. Takođe, u funkciji `trougaoniSistemR` potrebno je naredbu `x=zeros(1,n)` zameniti naredbom `x=single(zeros(1,n))`. Ovim postižemo da, prilikom izvršenja, ne dolazi do pojave formata jednostruke i dvostruke preciznosti u izrazima istovremeno.

Da bismo objasnili potrebu za izborom glavnog elementa uradićemo primer detaljno.

Prvo vršimo eliminaciju promenljive x_1. Posle eliminacije promenljive x_1, sistem jednačina postaje

$$\begin{array}{rcrcrcl} 10x_1 & - & 7x_2 & & & = & 7, \\ & - & 1.192093 \cdot 10^{-6} x_2 & + & 6x_3 & = & 6.000001, \\ & & 2.5x_2 & + & 5x_3 & = & 2.5. \end{array}$$

Uočimo element na poziciji $(2,2)$. Ovaj element je relativno mali po apsolutnoj vrednosti. Prilikom sledeće eliminacije, ovaj element proizvodi množenje sa relativno velikim brojem, što se vidi u sledećem koraku.

$$\begin{array}{rcrcrcl} 10x_1 & - & 7x_2 & & & = & 7, \\ & - & 1.192093 \cdot 10^{-6} x_2 & + & 6x_3 & = & 6.000001, \\ & & & & 1.2582917 \cdot 10^{7} x_3 & = & 1.2582916 \cdot 10^{7}. \end{array}$$

Iako brojevi $b_3^{(2)}$ i $a_{33}^{(2)}$ izgledaju isto, razlikuju se tek na osmoj cifri zapisani u dekadnom brojnom sistemu, ovi brojevi nisu isti. Ako sprovedemo sva računanja u `Matlab`u, i primenimo funkciju

`num2hex` na odgovarajuće elemente matrica, dobijamo heksadecimalne cifre `4b400004` i `4b400005`, otuda je vrednost promenljive x_3 manja od 1. Do gubitka tačne vrednosti x_3 dolazi otuda što jednostruka preciznost ne može da sprovede egzaktno sabiranje $-6.000001 \cdot 2.5/1.192093 \cdot 10^6 + 2.5$, jer jednostavno format jednostruke preciznosti nema dovoljno veliku dužinu mantise. Usled ovoga dolazi do zaokruživanja i do gubitka tačnosti u vrednosti promenljive x_3. Kad smo izgubili tačnost u vrednosti promenljive x_3, prilikom računanja promenljive x_2, dobijamo

$$x_2 = \frac{6.000001 - 6 \cdot x_3}{-1.192093 \cdot 10^{-6}} = -1.2.$$

Broj $-1.192093 \cdot 10^{-6}$ je nastao oduzimanjem brojeva bliskih po vrednosti, pa je samim tim sa malim brojem značajnih cifara. Ovo opterećuje tačnost sa kojom se izračunava vrednost promenljive x_2. Vidimo da je ta tačnost izuzetno mala. Relativna greška sa kojom je određena vrednost promenljive x_2 je reda veličine 10^{-1}. Konačno, ovako velika relativna greška promenljive x_2, proizvodi veliku relativnu grešku u promenljivoj x_1.

Možemo zaključiti da treba izbegavati male elemente na glavnoj dijagonali prilikom procesa Gaussove eliminacije iz dva razloga. Prvi je ograničena dužina mantise i problemi vezani sa nemogućnošću egzaktnih izračunavanja, i drugi je velika relativna greška u tim malim vrednostima koja se kasnije prenosi u procesu rešavanja zamenom unazad.

Izbegavanje elemenata koji su mali po apsolutnoj vrednosti na glavnoj dijagonali se postiže prostom zamenom mesta jednačina. Najbolje je na glavnoj dijagonali dovoditi elemente koji su najveći po apsolutnoj vrednosti u datoj koloni. U k-tom koraku na poziciju jednačine k dovodimo jednačinu koja ima po apsolutnoj vrednosti najveći element koji množi promenljivu x_k. Ovakav izbor glavnog elementa se naziva parcijalni izbor glavnog elementa.

Kao što smo opisali proces zamene mesta jednačina je ekvivalentan množenju permutacionim matricama. Proces faktorizacije sada možemo opisati ovako

$$U = M_{n-1} P^{(n-2)} M_{n-2} \cdots P^{(1)} M_1 A,$$

gde su $P^{(i)}$, $i = 1, \ldots, n-2$, permutacione matrice koje razmeštaju vrste u matrici tako da na pozicije na glavnoj dijagonali dovode elemente koji su najveći po apsolutnoj vrednosti. Na primer, proces parcijalnog izbora glavnog elementa možemo ilustrovati na primeru gornjeg sistema jednačina:

$$A = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2.099999 & 6 \\ 5 & -1 & 5 \end{bmatrix}, \quad M_1 = \begin{bmatrix} 1 & 0 & 0 \\ .3 & 1 & 0 \\ .5 & 0 & 1 \end{bmatrix},$$

$$A_1 = M_1 A = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -1.192093 \cdot 10^{-6} & 6 \\ 0 & 2.5 & 5 \end{bmatrix}, \quad P^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

$$A_1^{(1)} = P^{(1)} A_1 = \begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & -1.192093 \cdot 10^{-6} & 6 \end{bmatrix}, M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4.768372 \cdot 10^{-7} & 1 \end{bmatrix},$$

$$U = M_2 A_1^{(1)} = \begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.000002 \end{bmatrix} = M_2 P^{(1)} M_1 A.$$

Zanimljivo je što možemo naći matricu $M_1^{(1)}$ tako da važi $M_1^{(1)} P^{(1)} = P^{(1)} M_1$. Kako su permutacione matrice regularne, matrica $M_1^{(1)}$ se jednostavno nalazi. Naime, $M_1^{(1)} = P^{(1)} M_1 P^{(1)}$, jer su

permutacione matrice same sebi inverzne.

Permutacione matrice su same sebi inverzne iz sledećeg razloga: permutaciona matrica razmenjuje i-tu vrstu k-tom, pa kad dva puta razmenimo i-tu i k-tu vrstu dobijamo početnu matricu, što odgovara množenju jediničnom matricom. Za svaku permutacionu matricu važi $P^2 = I$.

Radi ilustracije odredimo matricu $M_1^{(1)}$. Imamo

$$M_1^{(1)} = P^{(1)} M_1 P^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ .3 & 1 & 0 \\ .5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ .5 & 0 & 1 \\ .3 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ .5 & 1 & 0 \\ .3 & 0 & 1 \end{bmatrix}.$$

Množenje permutacionom matricom sa desne strane, kao što vidimo, razmenjuje kolone, jer je $AP = (P^T A^T)^T = (PA^T)^T$, što predstavlja razmenu vrsta u matrici A^T, a to je razmena kolona u matrici A.

Ovako dobijena matrica $M_1^{(1)}$ ima razmenjene dve vrste i dve kolone u odnosu na matricu M_1. Primetimo da se strukture matrica M_1 i $M_1^{(1)}$ ne razlikuju.

Prethodno razmatranje nam omogućava zapis

$$U = M_2 M_1^{(1)} P^{(1)} A, \quad LU = PA, \quad L = \left(M_2 M_1^{(1)} \right)^{-1}, \quad P = P^{(1)}.$$

Pri svemu ovome se struktura matrice L ne menja, dakle, matrica L ostaje donja trougaona, jer se struktura svih M matrica ne menja, a kao što vidimo permutacione matrice deluju samo nad matricom sistema.

U slučaju da je matrica višeg reda od trećeg, koristeći Gaussovu eliminaciju sa parcijalnim izborom glavnog elementa, na isti način se dobija faktorizacija

$$LU = PA,$$

gde je P odgovarajuća permutaciona matrica.

Postoji mogućnost totalnog izbora glavnog elementa prilikom Gaussove eliminacije. Ova procedura, u k-toj eliminaciji, prodrazumeva izbor glavnog elementa, ne samo u k-toj koloni, već među svim elementima koji se nalaze na pozicijama (i, j), $i, j = k, \ldots, n$. Rezultat totalnog izbora glavnog elementa je faktorizacija matrice sistema u obliku

$$LU = PAQ,$$

gde matrica P opisuje razmenu jednačina unutar sistema, dok matrica Q opisuje razmenu promenljivih unutar sistema jednačina. Nećemo dublje ulaziti u ovu problematiku.

Sledeća teorema govori o grešci koja nastaje prilikom rešavanja sistema linearnih jednačina LU faktorizacijom sa totalnim izborom glavnog elementa.

Teorema 2.3 *Neka je $Ax = b$ i neka su $L = [\ell_{ij}]$ i $U = [u_{ij}]$ matrice koje se dobijaju algoritmom Gaussove eliminacije sa totalnim izborom glavnog elementa od matrice $A = [a_{ij}]$ na računskoj mašini mašinske preciznosti ε. Ako je x' rešenje sistema $Ux' = y$, $Ly = b$, onda je $(A+\Delta A)x' = b$, gde je*

$$[|\Delta a_{ij}|] \leq n\varepsilon \left(3\,[|a_{ij}|] + 5P^T\,[|\ell_{ij}|]\,[|u_{ij}|]\,Q^T \right) + O(\varepsilon^2), \quad \Delta A = [\Delta a_{ij}],$$

gde izraz $C \leq D$, $C = [c_{ij}]$, $D = [d_{ij}]$, treba shvatiti kao nejednakost $c_{ij} \leq d_{ij}$ između odgovarajućih elemenata matrica C i D.

Kao što vidimo teorema je vrlo slična odgovarajućoj teoremi koja govori o grešci rešavanja linearnog sistema bez izbora glavnog elementa. Međutim, postoji bitna razlika. Razlika su permutacione matrice koje minimiziraju grešku.

2.2.5 Matlab implementacija Gaussove eliminacije

Pored toga što operator \ implementira algoritam Gaussove eliminacije, postoje i funkcije koje su specijalizovane za rešavanje sistema linearnih jednačina.

Funkcija `lu` izračunava LU faktorizaciju. Ova funkcija može biti pozvana sa više ulaznih i izlaznih argumenata.

Funkcija `lu` pozvana samo sa jednim argumentom, matricom A, i dva izlazna, vraća matrice L i U za koje važi $A = LU$. Matrica U je gornje trougaona, a matrica L je 'psihološki' donje trougaona, u smislu da je donje trougaona do na množenje permutacionom matricom. Može se zaključiti da funkcija `lu` vrši LU faktorizaciju sa parcijalnim izborom glavnog elementa.

```
>>A=[10 -7 0; -3 2.099999 6; 5 -1 5];
>>[L,U]=lu(A)
L =
           1                         0                         0
        -0.3    -4.00000000233547e-007                         1
         0.5                         1                         0
U =
          10                        -7                         0
           0                       2.5                         5
           0                         0                  6.000002
```

Vidimo da je matrica `L` donje trougaona do na razmenu druge i treće vrste.

Funkcija `lu` pozvana sa jednim ulaznim argumentom, matricom A, i tri izlazna, vraća matrice L, U i permutacionu matricu P, tako da važi $PA = LU$. Pri ovome matrice L i U su donja i gornja trougaona matrica, redom. Ovako pozvana funkcija `lu` vrši LU faktorizaciju sa parcijalnim izborom glavnog elementa.

```
>>A=[10 -7 0; -3 2.099999 6; 5 -1 5];
>>[L,U,P]=lu(A)
L =
           1                         0                         0
         0.5                         1                         0
        -0.3    -4.00000000233547e-007                         1
U =
          10                        -7                         0
           0                       2.5                         5
           0                         0                  6.000002
P =
     1     0     0
     0     0     1
     0     1     0
>> L*U-P*A
ans =
             0                         0                         0
```

```
             0                         0                         0
-4.44089209850063e-16                  0                         0
```

Kao što vidimo u ovom slučaju je matrica `L` donje trougaona, ali matrice `L` i `U` ne predstavljaju LU faktorizaciju matrice `A`, nego matrice `PA`.

Postoje i drugi načini pozivanja funkcije `lu`, ali se svi uglavnom odnose na retke matrice i mi ih nećemo obrađivati.

Funkcija `linsolve` se poziva sa dva argumenta. Jedan je matrica sistema a drugi je slobodan vektor. Funkcija rešava sistem linearnih jednačina koristeći Gaussov algoritam sa parcijalnim izborom glavnog elementa. Ako je funkcija pozvana sa jednim izlaznim argumentom vraća samo rešenje sistema linearnih jednačina. U slučaju da je funkcija pozvana sa dva izlazna argumenta, drugi argument je recipročna vrednost faktora uslovljenosti matrice ako je sistem jednačina kvadratni, ili rang matrice sistema ako je sistem pravougaoni.

```
>>A=[1 2 3; 3 1 2; 2 1 3];
>>b=[1 2 3]';
>>[x,cond]=linsolve(A,b)
x =
         0.333333333333333
         -1.66666666666667
          1.33333333333333
cond =
        0.0576923076923077
```

Ako je funkcija `linsolve` pozvana sa jednim izlaznim argumentom, a faktor uslovljenosti sistema je veliki, funkcija štampa upozorenje da je faktor uslovljenosti sistema veliki. O faktoru uslovljenosti matrice sistema linearnih jednačina biće više reči u sledećoj sekciji.

Moguće je zadati i treći argument u pozivu funkcije koji je tipa strukture. Ovaj argument daje bliže informacije o tipu sistema, što omogućava funkciji `linsolve` izbor algoritma koji upotrebljava za rešavanje sistema. Moguće je identifikovati sistem kao gornje, donje trougaoni, kao sistem jednačina sa simetričnom matricom i slično.

```
>>A=[1 2 3; 0 1 2; 0 0 1];
>>b=[1 2 3]';
>>opts.UT=true;
>>[x,cond]=linsolve(A,b,opts)
x =
     0
    -4
     3
cond =
     1
```

Ovako pozvana funkcija `linsolve` rešava gornje trougaoni sistem linearnih jednačina. Struktura `opts` sadrži i druga polja koja mogu bliže odrediti strukturu matrice sistema jednačina koji se rešava. Na primer, postavljanjem polja `LT` na vrednost `true` funkcija rešava donje trougaoni sistem linearnih jednačina, postavljanjem polja `UHESS` na vrednost `true` funkcija rešava sistem jednačina sa matricom sistema koja ima gornju Hessenbergovu formu, postavljanjem polja `SYMM` na vrednost `true` rešava se sistem jednačina sa simetričnom (Hermitskom) matricom, postavljanjem

polja `TRANSA` na vrednost `true` rešava se sistem jednačina sa transponovanom matricom od predate, postavljanjem polja `POSDEF` na vrednost `true` rešava se sistem jednačina sa pozitivno definitnom matricom sistema, postavljanjem polja `RECT` na vrednost `true` funkcija podrazumeva rešavanje sistema jednačina sa generalnom pravougaonom matricom.

2.3 Uslovljenost sistema linearnih jednačina

U ovom odeljku proučavamo koliku maksimalnu relativnu grešku u rešenju sistema linearnih jednačina unosi greška u slobodnom vektoru ili matrici sistema. Ova razmatranja omogućavaju uvođenje pojma faktora uslovljenosti matrice sistema linearnih jednačina.

Teorema 2.4 *Neka je gornja granica apsolutne greške slobodnog vektora sistema linearnih jednačina $Ax = b$ zadata sa Δb i neka važi sledeća jednakost $A(x + \Delta x) = b + \Delta b$. Onda je*

$$\frac{||\Delta x||}{||x||} \leq ||A^{-1}||\,||A|| \frac{||\Delta b||}{||b||}.$$

Proizvod normi $||A^{-1}||\,||A||$ zovemo faktor uslovljenosti (kondicioni broj, engleski condition number) matrice A i označavamo sa $k(A)$.

Dokaz. Koristeći jednakosti $Ax = b$ i $A(x + \Delta x) = b + \Delta b$ i osobine normi vektora i matrica nalazimo

$$||A||\,||x|| \geq ||Ax|| = ||b||, \quad ||\Delta x|| = ||A^{-1}\Delta b|| \leq ||A^{-1}||\,||\Delta b||.$$

Deljenjem ove dve nejednakosti nalazimo

$$\frac{||\Delta x||}{||x||} \leq ||A^{-1}||\,||A|| \frac{||\Delta b||}{||b||}.$$

Ovim je dokaz teoreme završen. □

Ova teorema nam daje procenu gornje granice relativne greške rešenja pod uslovom da znamo gornju granicu relativne greške slobodnog vektora. Faktor koji 'pojačava' gornju granicu relativne greške slobodnog vektora je faktor uslovljenosti matrice sistema.

Faktor uslovljenosti matrice za subordinisane norme je uvek veći od jedinice, jer na osnovu nejednakosti (2.2), važi

$$1 = ||I|| = ||AA^{-1}|| \leq ||A||\,||A^{-1}|| = k(A).$$

Faktor uslovljenosti simetrične matrice u $||\cdot||_2$ normi se može iskazati kao

$$k(A) = ||A||_2||A^{-1}||_2 = \frac{|\lambda_{max}|}{|\lambda_{min}|}, \tag{2.3}$$

gde su $\lambda_{max}, \lambda_{min} \in \sigma(A)$ sopstvene vrednosti matrice A sa osobinom $|\lambda_{min}| \leq |\lambda| \leq |\lambda_{max}|$, $\lambda \in \sigma(A)$. Ova osobina simetričnih matrica se često koristi za izračunavanje faktora uslovljenosti.

Da stvari mogu da postanu zaista komplikovane pokazuje sledeći jednostavan primer. Posmatrajmo sledeći sistem linearnih jednačina

$$\begin{array}{rcrcr} x_1 & + & 7x_2 & = & 8, \\ x_1 & + & 7.000001x_2 & = & 8.000001. \end{array}$$

Očigledno je rešenje sistema $x = (1, 1)$. Ako samo malo promenimo slobodan vektor sistema linearnih jednačina i posmatramo sledeći sistem

$$\begin{array}{rcrcl} x_1 & + & 7x_2 & = & 8, \\ x_1 & + & 7.000001x_2 & = & 8.000002, \end{array}$$

dobićemo rešenje $x = (-6, 2)$. Ostaje nam da proverimo valjanost teoreme. Sledeće linije naredbi u `Matlab`u proveravaju tvrđenje teoreme.

```
>>A=[1 7; 1 7.000001];
>>b=[8 8.000001]';
>>db=[0 0.000001]';
>>rx=norm([-6; 2]-[1; 1],2)/norm([1; 1],2)
rx =
    5
>>rb=norm(db,2)/norm(b,2)
rb =
    8.838834212404687e-008
>>kA=norm(inv(A),2)*norm(A,2)
kA =
    1.000000139860232e+008
>>[rx kA*rb]
ans =
   5.000000000000000   8.838835448606089
>>cond(A)
ans =
    1.000000140082561e+008
```

Kao što vidimo rezultati se slažu. Međutim, vidimo na ovom prostom primeru da je moguće pronaći matricu reda dva koja ima veliku vrednost faktora uslovljenosti. Posledica velike vrednosti faktora uslovljenosti je da jako mala relativna greška u slobodnom vektoru prouzrokuje velike relativne greške u rešenju. Ovde relativna greška u slobodnom vektoru reda veličine 10^{-8} prouzrokuje relativnu grešku 5. u rešenju.

Funkcija `cond`, kako se vidi u primeru, može se koristiti za izračunavanje faktora uslovljenosti. Funkcija `cond` može se koristiti za izračunavanje faktora uslovljenosti u normama $p = 1$, $p = 2$ i $p = +\infty$, kao i Frobeniusovoj normi, slično kao i funkcija `norm`. Primetimo da funkcija `cond` daje malo veću vrednost faktora uslovljenosti matrice `A`, što je posledica algoritma za ocenu faktora uslovljenosti u funkciji `cond` i velike vrednosti faktora uslovljenosti matrice `A`.

Funkcija `condest` procenjuje vrednost faktora uslovljenosti matrice u normi $||\cdot||_1$. Prilikom ocene ne izračunava se inverzna matrica, već se koristi ocena norme $||A^{-1}||_1$ pomoću Hagerovog algoritma ili algoritma koji su razvili Higham i Tisseur. Ovaj način izračunavanja faktora uslovljenosti je brži, ali je podložan greškama.

Moguće je u prethodnom primeru postići i više. Povećamo li broj nula, dobićemo matricu sistema sa još većim faktorom uslovljenosti. Može se dobiti matrica sa tako velikim faktorom uslovljenosti da su sva izračunavanja besmislena. Izaberemo li matricu

$$A = \begin{bmatrix} 1 & 7 \\ 1 & 7.000000000000001 \end{bmatrix}$$

dobićemo faktor uslovljenosti reda veličine $7.96 \cdot 10^{16}$. Naravno, ako radimo u formatu dvostruke preciznosti slobodan vektor mora imati relativnu grešku barem jednaku mašinskoj preciznosti, što znači da možemo očekivati da rešenje sistema jednačina neće imati nijednu značajnu cifru.

Ovo tvrđenje možemo jednostavno proveriti. Naime, izaberimo rešenje, recimo $x = (1, 1)$, a zatim konstruišimo slobodan vektor za to rešenje $b = Ax$, na kraju rešimo sistem $Ax' = b$. Odredimo relativnu grešku izabranog rešenja i dobijenog rešenja. Sledeća sekvenca naredbi opisuje proceduru u Matlabu.

```
>>A=[1 7; 1 7.000000000000001];
>>x=[1; 1];
>>b=A*x;
>>xPrim=linsolve(A,b);
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 7.930164e-018.
>>norm(xPrim-x,2)/norm(x,2)
ans =
   0.707106781186547
```

Postoji još jedna interpretacija nejednakosti u teoremi 2.4. Naime, neka je dat sistem linearnih jednačina $Ax = b$ i pretpostavimo da nas zanima kolika je gornja granica odstupanja vektora x' od rešenja sistema linearnih jednačina. Koristeći pomenutu teoremu možemo tvrditi da važi

$$\frac{||x' - x||}{||x||} \leq k(A) \frac{||Ax' - b||}{||b||}.$$

Veličina $s = b - Ax'$ se naziva ostatkom, i kao što vidimo, može se koristiti da izmeri koliko neki vektor odstupa od rešenja sistema linearnih jednačina. Međutim, ovo treba shvatiti uslovno. Naime, mala vrednost norme ostatka s znači mala odstupanja vektora x' od rešenja sistema linearnih jednačina za dobro uslovljene sisteme jednačina, dok za slabo uslovljene sisteme jednačina nema značaja. Ako je faktor uslovljenosti veliki, mala norma vektora ostatka s ne znači da je mala relativna greška u rešenju sistema linearnih jednačina. Međutim, možemo slobodno zaključiti da velika vrednost norme vektora ostatka s znači da je velika relativna greška rešenja sistema linearnih jednačina.

Slična teorema važi i u slučaju kad se posmatra greška u matrici sistema.

Teorema 2.5 *Neka je dat sistem linearni jednačina $Ax = b$ i neka važi $(A + \Delta A)(x + \Delta x) = b$. Ako je matrica $I + A^{-1}\Delta A$ regularna važi*

$$\frac{||\Delta x||}{||x||} \leq ||(I + A^{-1}\Delta A)^{-1}||\,||A^{-1}\Delta A|| \leq ||(I + A^{-1}\Delta A)^{-1}||k(A)\frac{||\Delta A||}{||A||}.$$

Pod uslovom $||\Delta A|| \leq \psi||A||$ i $\psi k(A) < 1$, važi

$$\frac{||\Delta x||}{||x||} \leq \frac{\psi k(A)}{1 - \psi k(A)}.$$

Dokaz. Koristeći date jednakosti i osobine norme nalazimo

$$(A + \Delta A)\Delta x = -\Delta A x, \quad (I + A^{-1}\Delta A)\Delta x = -A^{-1}\Delta A x,$$
$$\Delta x = -(I + A^{-1}\Delta A)^{-1}A^{-1}\Delta A x, \quad ||\Delta x|| \leq ||(I + A^{-1}\Delta A)^{-1}||\,||A^{-1}\Delta A||\,||x||,$$
$$||A^{-1}\Delta A|| \leq ||A^{-1}||\,||A||\frac{||\Delta A||}{||A||} = k(A)\frac{||\Delta A||}{||A||}.$$

Pretpostavimo li da važe date nejednakosti, nalazimo

$$A\Delta x = -\Delta A(x+\Delta x), \quad \Delta x = -A^{-1}\Delta A(x+\Delta x),$$
$$||\Delta x|| \le ||A^{-1}||\,||\Delta A||\,(||x|| + ||\Delta x||), \quad ||\Delta x|| \le \psi||A^{-1}||\,||A||\,(||x|| + ||\Delta x||),$$
$$(1 - \psi k(A))||\Delta x|| \le \psi k(A)||x||.$$

Ovim smo završili dokaz teoreme. □

Primer nam može biti ponovo sličan sistem jednačina kao za ilustraciju prethodne teoreme. Posmatrajmo sistem jednačina

$$\begin{array}{rcrcr} x_1 & + & 7x_2 & = & 8, \\ x_1 & + & 7.000001x_2 & = & 8.000002. \end{array}$$

Rešenje znamo $x = (-6, 2)$. Posmatrajmo sada sistem

$$\begin{array}{rcrcr} x_1 & + & 7x_2 & = & 8, \\ x_1 & + & 7.000002x_2 & = & 8.000002. \end{array}$$

I ovde znamo rešenje $x' = (1, 1)$. Ilustraciju teoreme možemo dati nizom naredbi u `Matlabu`.

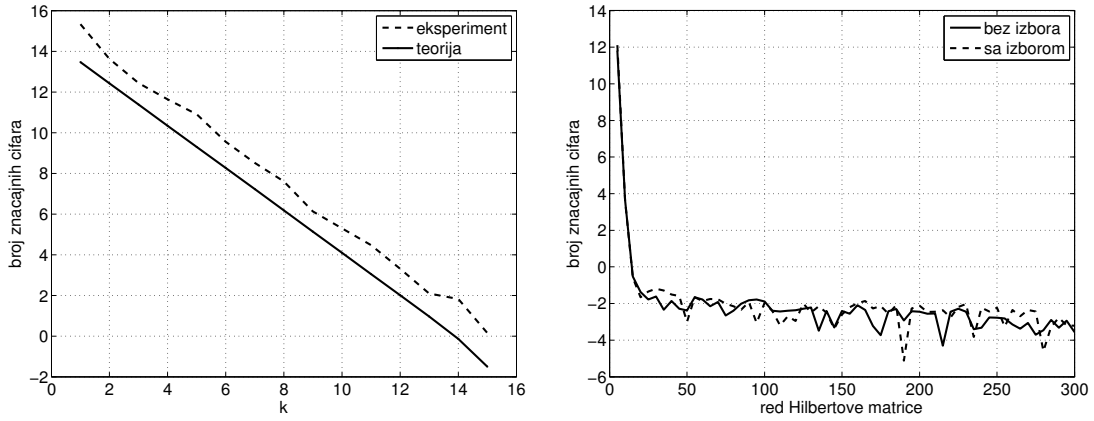
```
>>A=[1 7; 1 7.000001];
>>dA=[0 0; 0 0.000001];
>>x=[-6; 2];
>>dx=[1; 1]-x;
>>invA=inv(A);
>>n1=norm(inv(eye(2)+invA*dA),2);
>>n2=norm(invA*dA,2);
>>[norm(dx,2)/norm(x,2) n1*n2]
ans =
   1.118033988749895  25.962912012587370
>>psi=norm(dA,2)/norm(A,2)
psi =
    9.999999300000025e-008
>>kA=cond(A)
kA =
    1.000000140082561e+008
>>psi*kA
ans =
  10.000000698602246
```

Kao što vidimo, izlaz naredbe `[norm(dx,2)/norm(x,2) n1*n2]`, rezultati se dobro slažu. Međutim, iz drugog dela, vidimo da drugi oblik teoreme nije uvek moguće primeniti. Naime, čak i ako izaberemo najmanje moguće ψ, čemu treba uvek težiti, jer je u tom slučaju izraz na levoj strani nejednakosti najmanji, uslov $\psi k(A) < 1$ nije ispunjen. Međutim, druga nejednakost u teoremi jasno potencira važnost pojma faktora uslovljenosti. Zato je i data.

U slučaju kad je dopušteno variranje slobodnog vektora sistema linearnih jednačina i matrice sistema linearnih jednačina mogu se dati razne ocene za relativnu grešku sistema linearnih jednačina. Mi ćemo bez dokaza dati jednu teoremu koja opet naglašava važnost pojma faktora uslovljenosti.

Teorema 2.6 *Neka je dat sistem linearnih jednačina* $Ax = b$ *i neka važi* $(A + \Delta A)(x + \Delta x) = b + \Delta b$. *Ako je* $||\Delta A||\,||A^{-1}|| = r < 1$, *onda važi*

$$\frac{||\Delta x||}{||x||} \leq \frac{2k(A)}{1-r} \max\left\{\frac{||\Delta b||}{||b||}, \frac{||\Delta A||}{||A||}\right\}.$$



Slika 2.1: Značajne cifre u rešenju sistema (2.4) slika levo, značajne cifre u rešenju sistema sa Hilbertovom matricom u funkciji reda matrice sa i bez izbora glavnog elementa slika desno.

Sistemi linearnih jednačina sa velikim faktorom uslovljenosti se nazivaju loše ili slabo uslovljenim (engleski ill-conditioned). Sistemi linearnih jednačina sa malim faktorom uslovljenosti se nazivaju dobro uslovljenim (engleski well-conditioned). Uticaj porasta vrednosti faktora uslovljenosti se ogleda u smanjenju broja značajnih cifara rešenja.

Posmatrajmo niz sistema linearnih jednačina

$$\begin{aligned} 7x_1 &+ \frac{1}{3}x_2 &&= \frac{22}{3}, \\ 7x_1 &+ \left(\frac{1}{3} + 11^{-k}\right) x_2 &&= \frac{22}{3} + 11^{-k}, \end{aligned} \tag{2.4}$$

za $k = 1, \ldots, 15$. Rešenje svih sistema je $x = (1, 1)$. Slika 2.1, levo, prikazuje gubitak značajnih cifara u rešenju sistema, dobijenih funkcijom `linsolve`, sa promenom parametra $k = 1, \ldots, 15$. Konkretno prikazuje grafik funkcije $-\log_{10}(||x'_k - (1,1)||_2/\sqrt{2})$, gde je x'_k, rešenje sistema linearnih jednačina sa parametrom k dobijeno funkcijom `linsolve`. Kao što vidimo gubitak značajnih cifara je skoro linearan. Punom linijom je prikazan grafik funkcije $-\log_{10}(2^{-52}k(A_k)/(1 - 2^{-52}||A_k^{-1}||))$, gde je $k(A_k)$ faktor uslovljenosti matrice sistema. Ova funkcija je teorijska granica relativne greške rešenja sistema linearnih jednačina data u teoremi 2.6. Relativne greške u slobodnom vektoru i matrici sistema su reda veličine mašinske preciznosti ε, dok je norma matrice ΔA reda veličine 2ε, jer su apsolutne greške pojedinih elemenata matrice reda veličine ε. Vidimo da se teorijska ocena dobro slaže sa praktičnom.

Poznatija familija slabo uslovljenih matrica su Hilbertove matrice $H_n = [1/(i + j)]_{i,j=1}^n$, $n \in \mathbb{N}$. Ova grupa matrica se često koristi pri demonstraciji osobina slabo uslovljenih matrica. Posebna pogodnost ovih matrica je ta što su simetrične i pozitivno definitne, a ipak slabo uslovljene. Na primeru ovih matrica pokazaćemo da Gaussov metod, sa i bez izbora glavnog elementa, proizvodi

podjednako loše rezultate. Slika 2.1, desno, sadrži grafike značajnih cifara u rešenju sistema $H_n x = b$, sa i bez izbora glavnog elementa, u funkciji reda matrice n. Vektor x je vektor čije su sve koordinate jednake jedan, a vektor b se izračunava matričnim množenjem iz tog uslova, $b = H_n x$. Funkcija `hilb` konstruiše Hilbertovu matricu odgovarajućeg reda u `Matlabu`. Za rešavanje sistema linearnih jednačina bez izbora glavnog elementa koristi se funkcija `gaussovaEliminacijaR` data u poglavlju o Gaussovoj eliminaciji, dok se za rešavanje sa izborom glavnog elementa koristi funkcija `linsolve`. Skript kojim je generisan grafik je

```
x=[]; z=[]; y=[];
clear opts;
opts.SYM=true;
for i=5:5:300
    x=[x i];
    H=hilb(i);
    jedinice=ones(i,1);
    b=H*jedinice;
    sol=gaussovaEliminacijaR(H,b);
    y=[y norm(sol-jedinice,2)/norm(jedinice,2)];
    sol=linsolve(H, b, opts);
    z=[z norm(sol-jedinice,2)/norm(jedinice,2)];
end
plot(x,-log10(y),'-',x,-log10(z),'--');
legend('bez izbora','sa izborom');
xlabel('red Hilbertove matrice');
ylabel('broj znacajnih cifara');
grid on;
```

Kao što vidimo samo za matrice vrlo malog reda imamo mali gubitak značajnih cifara. Recimo do reda 10 rešenja imaju nekog smisla. Preko reda 10, rešenja imaju sve manje i manje značajnih cifara, tako da već sa redom 20 broj značajnih cifara u rešenju sistema linearnih jednačina je postao nula. Primećujemo da broj značajnih cifara ne zavisi bitno od toga da li sistem rešavamo sa ili bez izbora glavnog elementa. Prosto sistem je slabo uslovljen, i tu nema pomoći klasičnim metodima rešavanja.

2.4 Iterativni metodi

Direktni metodi pokušavaju da reše sistem jednačina egzaktno. Međutim, tačno rešenje sistema jednačina iz više razloga nije moguće odrediti. Neke od razloga smo pomenuli i vezani su za faktor uslovljenosti. Takođe, sistemi jednačina koji nastaju prilikom rešavanja parcijalnih diferencijalnih jednačina, a koji imaju veliki broj promenljivih, imaju matrice sistema koje su većinom ispunjene nulama. U ovakvim slučajevim je primena direktnih metoda za rešavanje sistema linearnih jednačina jako 'skupa' iz perspektive potrebnog procesorskog vremena. Alternativa je pronađena u iterativnim metodima.

2.4.1 Primer iterativnog metoda

Da bismo motivisali upotrebu iterativnih metoda za rešavanje sistema linearnih jednačina počećemo sa tradicionalnim metodom. Naš metod izbora je Jacobijev metod. Posmatrajmo sledeći

sistem linearnih jednačina

$$\begin{array}{rcrcrcl} 4x_1 & + & x_2 & + & x_3 & = & 1, \\ x_1 & + & 4x_2 & + & x_3 & = & 2, \\ x_1 & + & x_2 & + & 4x_3 & = & 3. \end{array} \tag{2.5}$$

Ideja se sastoji u zapisivanju sistema jednačina u sledećoj formi

$$\begin{aligned} x_1 &= \frac{1}{4}\left(1 - x_2 - x_3\right), \\ x_2 &= \frac{1}{4}\left(2 - x_1 - x_3\right), \\ x_3 &= \frac{1}{4}\left(3 - x_1 - x_2\right). \end{aligned}$$

Ako uvedemo matrice

$$B = \begin{bmatrix} 0 & -1/4 & -1/4 \\ -1/4 & 0 & -1/4 \\ -1/4 & -1/4 & 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1/4 \\ 2/4 \\ 3/4 \end{bmatrix},$$

prethodnu formu sistema linearnih jednačina možemo zapisati i ovako

$$x = Bx + \beta.$$

Polazeći od proizvoljne početne vrednosti x^0, formiramo niz vektora

$$x^{k+1} = Bx^k + \beta, \quad k \in \mathbb{N}_0,$$

gde je $x^k = (x_1^k, x_2^k, x_3^k)$, $k \in \mathbb{N}_0$, i nadamo se da ovako formiran niz konvergira ka rešenju sistema linearnih jednačina. Rešenje sistema je $x = (0, 1/3, 2/3)$, a sledeći skript konstruiše prvih 20 članova niza x^k, $k \in \mathbb{N}_0$.

```
x=[1; 1; 1];
B=[0 -1/4 -1/4; -1/4 0 -1/4; -1/4 -1/4 0];
for k=1:20
    x=[x [1/4; 2/4; 3/4]+B*x(:,end)];
end
single(x)'
```

Rezultati izvršenja su

```
ans =
                1                1                1
            -0.25                0             0.25
           0.1875              0.5           0.8125
        -0.078125             0.25         0.578125
       0.04296875            0.375        0.7070313
      -0.02050781           0.3125        0.6455078
       0.01049805          0.34375        0.6770020
     -0.005187988         0.328125        0.6614380
      0.002609253        0.3359375        0.6692657
```

-0.001300812	0.3320313	0.6653633
0.0006513596	0.3339844	0.6673174
-0.0003254414	0.3330078	0.6663411
0.0001627803	0.3334961	0.6668294
-8.137524e-005	0.3332520	0.6665853
4.069135e-005	0.3333740	0.6667073
-2.034474e-005	0.3333130	0.6666463
1.017260e-005	0.3333435	0.6666768
-5.086244e-006	0.3333282	0.6666616
2.543136e-006	0.3333359	0.6666692
-1.271565e-006	0.3333321	0.6666654
6.357832e-007	0.3333340	0.6666673

Analizom rezultata vidimo da niz vektora stvarno konvergira ka rešenju sistema linearnih jednačina. Međutim, ovo ponašanje nije pravilo. Potrebno je da sistem linearnih jednačina zadovolji izvesne uslove da bi konstruisani niz vektora konvergirao ka rešenju sistema linearnih jednačina.

Niz vektora $x^{k+1} = Bx^k + \beta$, $k \in \mathbb{N}_0$, nazivamo iterativni proces.

2.4.2 Konstrukcija iterativnih metoda

Posmatrajmo sistem linearnih jednačina $Ax = b$ i neka je $A = M - N$, pri čemu je M regularna matrica. Sistem linearnih jednačina možemo napisati u sledećem obliku

$$Mx = Nx + b, \quad x = M^{-1}Nx + M^{-1}b, \quad x = Bx + \beta, \quad B = M^{-1}N, \quad \beta = M^{-1}b.$$

Da bismo dokazali sledeću teoremu navodimo sledeću lemu bez dokaza.

Lema 2.12 *Pretpostavimo da je neka norma matrice B manja od jedan, onda je matrica $I - B$ regularna.*

Sad smo u mogućnosti da damo teoremu o konvergenciji iterativnih procesa.

Teorema 2.7 *Neka je dat sistem linearnih jednačina $x = Bx + \beta$ i neka je x^0 proizvoljni vektor. Za niz vektora $x^{k+1} = Bx^k + \beta$, $k \in \mathbb{N}_0$, važe sledeće ocene:*

1° $||x^{k+1} - x^k|| \leq ||B||\,||x^k - x^{k-1}||$,
2° $||x^{k+1} - x^k|| \leq ||B||^k||x^1 - x^0||$,
3° $||x^{k+1} - x|| \leq ||B||\,||x^k - x||$,
4° $||x^{k+1} - x|| \leq ||B||^k||x^1 - x||$,
5° $||x^k - x|| \leq ||(I - B)^{-1}B||\,||x^k - x^{k-1}||$,

pri čemu ocena 5° važi pod uslovom $||B|| < 1$.

Pod uslovom $||B|| < 1$ iterativni proces $x^{k+1} = Bx^k + \beta$, $k \in \mathbb{N}_0$, konvergira ka rešenju sistema linearnih jednačina za proizvoljan vektor x^0.

Dokaz. Kombinujući jednakosti

$$x^{k+1} = Bx^k + \beta, \quad x^k = Bx^{k-1} + \beta,$$

dobijamo

$$x^{k+1} - x^k = B(x^k - x^{k-1}), \quad ||x^{k+1} - x^k|| \leq ||B||\,||x^k - x^{k-1}||.$$

Ako ponovimo prethodno dobijeni stav više puta, nalazimo

$$||x^{k+1} - x^k|| \leq ||B||\,||x^k - x^{k-1}|| \leq ||B||^2\,||x^{k-1} - x^{k-2}|| \leq \cdots \leq ||B||^k||x^1 - x^0||.$$

Slično dobijanju prethodnih nejednakosti mogu se dobiti i nejednakosti 3° i 4°. Samo se ovoga puta kombinuju jednakosti

$$x^{k+1} = Bx^k + \beta, \quad x = Bx + \beta.$$

Kombinujući jednakosti

$$x^k = Bx^{k-1} + \beta, \quad x = Bx + \beta,$$

i pod uslovom $||B|| < 1$, uz tvrđenje leme 2.12, dobijamo

$$x^k - x = B(x^{k-1} - x) = B(x^{k-1} - x^k + x^k - x),\ (I - B)(x^k - x) = B(x^{k-1} - x^k),$$
$$x^k - x = (I - B)^{-1}B(x^{k-1} - x^k), \quad ||x^k - x|| \leq ||(I - B)^{-1}B||\,||x^k - x^{k-1}||.$$

Ako je ispunjen uslov $||B|| < 1$, koristeći lemu 2.12, nalazimo da sistem $(I - B)x = \beta$ ima jedinstveno rešenje.

Pod uslovom $||B|| < 1$, koristeći četvrti stav

$$||x^k - x|| \leq ||B||^{k-1}||x^1 - x||,$$

nalazimo da desna strana teži nuli kad k teži beskonačnosti, tako da važi

$$\lim_{k \to +\infty} ||x^k - x|| = 0.$$

Odavde zaključujemo da niz x^k, $k \in \mathbb{N}_0$, konvergira ka rešenju x sistema linearnih jednačina. □

Stav 5° iz prethodne teoreme omogućava nam da utvrdimo gornju granicu apsolutne greške x^k koristeći apsolutnu grešku x^k u odnosu na x^{k-1}. Kako vidimo konstanta proporcionalnosti je $||(I - B)^{-1}B||$. Kako mi ne znamo tačno rešenje sistema jednačina, koristeći ovu ocenu, a kako veličine $||B||$ i $||x^k - x^{k-1}||$ možemo da računamo u svakoj iteraciji, u stanju smo da ocenjujemo gornju granicu apsolutne greške iteracije prema rešenju sistema jednačina. Može se pokazati da važi

$$||(I - B)^{-1}B|| \leq \frac{||B||}{1 - ||B||},$$

pri čemu je ova ocena grublja, posebno ako je upotrebljena norma matrice B bliska vrednosti jedan.

Primetimo da se uslov $||B|| < 1$ odnosi na bilo koju normu. Ovaj uslov je dovoljan. Međutim, nije i potreban. Naime, može se desiti da neka norma nije manja od jedan, a da iterativni proces konvergira. Čak može se desiti da sve matrične norme koje smo definisali imaju vrednost veću od jedan, a da je iterativni proces konvergentan.

Da bismo donekle ilustrovali opisane pojave posmatrajmo sledeći iterativni proces

$$x^{k+1} = Bx^k + \beta, \quad B = \begin{bmatrix} .1 & .9 \\ .8 & .05 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Ako izračunamo norme matrice B, nalazimo

$$||B||_1 = \max\{|.1| + |.8|, |.9| + |.05|\} = .95,$$
$$||B||_F = \sqrt{.1^2 + .9^2 + .8^2 + .05^2} \approx 1.21,$$
$$||B||_{+\infty} = \max\{|.1| + |.9|, |.8| + |.05|\} = 1, \quad ||B||_2 \approx .94.$$

Kao što vidimo za dokaz konvergencije možemo iskoristiti norme $||\cdot||_1$ i $||\cdot||_2$. Međutim, ne mogu se koristiti norme $||\cdot||_F$ i $||\cdot||_{+\infty}$.

Još zanimljiviji primer daje iterativni proces $x^{k+1} = Bx^k + \beta$, gde je

$$B = \begin{bmatrix} 1/3 & 1/9 \\ 2 & 1/3 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1/9 \\ -4/3 \end{bmatrix}.$$

U ovom slučaju je $||B||_1 = 7/3 > 1$, $||B||_F = 2.06 > 1$, $||B||_{+\infty} = 7/3 > 1$ i $||B||_2 = 2.06 > 1$. Vidimo da o konvergenciji iterativnog procesa na osnovu normi ne možemo zaključiti ništa.

U ovakvim slučajevima može se koristiti spektralni radijus matrice. Naime, prethodna teorema može se pokazati i kad se pojam norme matrice zameni pojmom spektralnog radijusa matrice. Za pomenutu matricu B spektralni radijus iznosi $\rho(B) \approx .8$ pa je iterativni proces konvergentan. Rešenje sistema iznosi $x = (-1/3, -3)$.

Sledeća teorema daje potrebne i dovoljne uslove konvergencije.

Teorema 2.8 *Iterativni proces $x^{k+1} = Bx^k + \beta$, $k \in \mathbb{N}_0$, konvergira ka rešenju sistema linearnih jednačina $x = Bx + \beta$, pri čemu je x^0 proizvoljno, ako i samo ako je spektralni radijus $\rho(B)$ matrice B po modulu manji od jedan.*

Ova teorema daje potrebne i dovoljne uslove konvergencije. Dakle, možemo reći da iterativni proces konvergira ako i samo ako su sve sopstvene vrednosti matrice B po modulu manje od jedan. Pomenimo da ako su sve sopstvene vrednosti matrice B po modulu manje od jedan onda je matrica $I - B$ regularna, tako da sistem jednačina $(I - B)x = \beta$ ima jedinstveno rešenje $x = (I - B)^{-1}\beta$.

Pitanje od posebnog značaja je koliko značajnih cifara dobijamo po iteraciji. Na ovo pitanje možemo odgovoriti relativno jednostavno koristeći teoremu 2.7. Naime, koristeći stav 3° možemo pisati

$$r_k = \frac{||x^{k+1} - x||}{||x||} \leq ||B|| \frac{||x^k - x||}{||x||} = ||B|| r_{k-1}.$$

Odavde dobijamo $-\log_{10} r_k \geq -\log_{10} ||B|| - \log_{10} r_{k-1}$, znači po iteraciji dobijamo $-\log_{10} ||B||$ značajnih cifara u rešenju. Što je norma matrice B bliža vrednosti jedan to će prinos značajnih cifara po iteraciji biti manji. Idealno, ako je $||B|| = 0$, onda je $-\log_{10} ||B|| = +\infty$, po iteraciji dobijamo beskonačno mnogo značajnih cifara. Ovo jeste tačno, jer već u prvoj iteraciji nalazimo tačno rešenje $x^1 = 0x^0 + \beta = \beta$. Iskoristimo li stav 4° iz teoreme 2.7 možemo dobiti i sledeću ocenu

$$-\log_{10} r_{k+1} \geq -k \log_{10} ||B|| - \log_{10} r_1.$$

Iz ove nejednakosti vidimo da sa brojem iteracija broj značajnih cifara raste linearno, sa nagibom $-\log_{10} ||B||$. Zato kažemo da je red konvergencije iterativnih metoda za rešavanje sistema linearnih jednačina linearan ili jednak jedan. Umesto norme matrice B u izrazima za prinos značajnih cifara može se koristiti spektralni radijus matrice B. Izrazi sa spektralnim radijusom daju bolju procenu prinosa značajnih cifara, ali ne menjaju bitno zaključke.

U primeru iz prethodnog odeljka, sistem jednačina (2.5), imamo $||B||_2 = .5$, pa je prinos značajnih cifara po iteraciji $-\log_{10} ||B||_2 = .3$. Dakle, potrebne su otprilike tri do četiri iteracije za jednu značajnu cifru u rešenju, što se iz prikazanih rezultata i vidi.

2.4.3 Jacobijev iterativni metod

Jacobijev metod možemo da opišemo kao metod koji se formira na sledeći način. Neka matrica sistema ima faktorizaciju $A = B_1 + D + B_2$, gde je B_1 donja trougaona sa nulama na glavnoj

dijagonali, D dijagonalna matrica i B_2 gornja trougaona matrica sa nulama na glavnoj dijagonali. Jacobijev metod za rešavanje sistema jednačina $Ax = b$, ima oblik

$$x^{k+1} = Bx^k + \beta, \quad B = -D^{-1}(B_1 + B_2), \quad \beta = D^{-1}b.$$

Naravno, Jacobijev metod je moguće formirati jedino u slučaju kad je matrica D regularna, drugim rečima, kada su svi elementi na glavnoj dijagonali matrice A različiti od nule.

Na osnovu teoreme 2.8 možemo odmah formulisati teoremu koja daje potrebne i dovoljne uslove konvergencije Jacobijevog metoda za proizvoljnu vrednost x^0.

Teorema 2.9 *Jacobijev metod konvergira, za proizvoljnu vrednost x^0, ako i samo ako sva rešenja jednačine*

$$\begin{vmatrix} \lambda a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \lambda a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & \lambda a_{nn} \end{vmatrix} = 0,$$

imaju moduo manji od jedan.

Dokaz. Primenom teoreme 2.8 znamo da iterativni proces konvergira pod uslovom $\rho(B) < 1$. Međutim, $B = -D^{-1}(B_1 + B_2)$ i karakteristični polinom matrice B ima oblik

$$p(\lambda) = \det(-D^{-1}(B_1 + B_2) - \lambda I) = \det(D^{-1}) \det(B_1 + B_2 + \lambda D).$$

Kako vidimo, sopstvene vrednosti matrice B su upravo rešenja jednačine pomenute u teoremi. □

Skalarno zapisan Jacobijev metod izgleda ovako

$$x_i^{k+1} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^k - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^k + \frac{b_i}{a_{ii}}, \quad i = 1, \dots, n.$$

Za neke specijalne klase matrica može se utvrditi konvergencija Jacobijevog metoda.

Definicija 2.2 *Matrica A, reda n, je striktno dijagonalno dominantna po vrstama ako i samo ako je ispunjen uslov $|a_{ii}| > \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$, $i = 1, \dots, n$.*

Na primer, matrica

$$A = \begin{bmatrix} 3 & -1 & 1 \\ 1 & 4 & -2 \\ -3 & 3 & 9 \end{bmatrix},$$

je striktno dijagonalno dominantna po vrstama, jer važi

$$3 = |3| > |-1| + |1| = 2, \quad 4 = |4| > |1| + |-2| = 3, \quad 9 = |9| > |-3| + |3| = 6.$$

Teorema 2.10 *Ako je matrica A striktno dijagonalno dominatna po vrstama, onda Jacobijev metod za sistem linearnih jednačina $Ax = b$ konvergira.*

Na primer, za pomenutu matricu A Jacobijev metod konvergira.

Primer Jacobijevog metoda je dat u prvom odeljku ove sekcije, kad smo počeli izlaganje o iterativnim metodima.

2.4.4 Gauss-Seidelov iterativni metod

Gauss-Seidelov iterativni metod je sličan Jacobijevom metodu. Matrica sistema linearnih jednačina $Ax = b$ se faktoriše kao i u slučaju Jacobijevog metoda, $A = B_1 + D + B_2$, gde je D dijagonalna matrica, B_1 donja trougaona sa nulama na glavnoj dijagonali i B_2 gornja trougaona sa nulama na glavnoj dijagonali. Gauss-Seidelov metod možemo formulisati ovako

$$x^{k+1} = Bx^k + \beta, \quad B = -(B_1 + D)^{-1}B_2, \quad \beta = (B_1 + D)^{-1}b.$$

Na osnovu ove reprezentacije Gauss-Seidelovog metoda moguće je formulisati sledeću teoremu.

Teorema 2.11 *Gauss-Seidelov metod konvergira, za proizvoljnu vrednost x_0, ako i samo ako sva rešenja jednačine*

$$\begin{vmatrix} \lambda a_{11} & a_{12} & \dots & a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{n1} & \lambda a_{n2} & \dots & \lambda a_{nn} \end{vmatrix} = 0,$$

imaju moduo manji od jedan.

Dokaz. Kao i u slučaju Jacobijevog metoda, karakteristični polinom matrice B, posle malo računanja, može se izraziti determinantom koja je data u tvrđenju teoreme. □

Skalarani oblik Gauss-Seidelovog metoda izgleda ovako

$$x_i^{k+1} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^k + \frac{b_i}{a_{ii}}, \quad i = 1, \dots, n.$$

Jedina razlika je što Gauss-Seidelov metod prilikom izračunavanja u $(k+1)$-voj iteraciji koristi sve vrednosti već izračunate u $(k+1)$-voj iteraciji, dok ih Jacobijev metod ignoriše.

Kao i u slučaju Jacobijevog metoda, za neke specijalne klase matrica se konvergencija Gauss-Seidelovog metoda može dokazati.

Teorema 2.12 *Ako je matrica sistema linearnih jednačina dijagonalno dominantna po vrstama ili simetrična i pozitivno definitna, Gauss-Seidelov metod konvergira.*

Bitna razlika između Jacobijevog i Gauss-Seidelovog metoda je ta što Gauss-Seidelov metod konvergira za simetrične i pozitivno definitne matrice. Sistemi sa ovakvim matricama se najčešće javljaju prilikom numeričkog rešavanja parcijalnih diferencijalnih jednačina.

2.4.5 Kriterijum za prekidanje iteracija

Ovaj odeljak pokušava da odgovori na pitanje u kojoj iteraciji treba prekinuti iterativni proces. Drugim rečima, kad je postignuto rešenje sa dovoljnom tačnošću. Odgovor na ovo pitanje nije jednostavan.

Jedna od metoda bi bila praćenje norme ostatka. Iz iteracije u iteraciju pratimo veličinu $s^k = b - Ax^k$. Međutim, kako smo pokazali

$$\frac{||x^k - x||}{||x||} \leq k(A) \frac{||s^k||}{||b||},$$

tako da ispunjenje uslova $||s^k|| \leq \varepsilon||b||$, znači jedino $r_x \leq k(A)\varepsilon$. Ako je matrica sistema slabo uslovljena rešenje sistema jednačina će biti određeno sa velikom relativnom greškom.

Drugi pristup je praćenje veličine $||x^{k+1} - x^k||$. Kako smo pokazali u teoremi 2.7, stav 5°, važi ocena

$$||x^{k+1} - x|| \leq ||(I - B)^{-1}B||\,||x^{k+1} - x^k|| \leq \frac{||B||}{1 - ||B||}||x^{k+1} - x^k||,$$

tako da je praćenje ove veličine dobro u slučaju kad je norma matrice B relativno mala u odnosu na broj jedan. Što je norma matrice B bliža broju jedan to je apsolutna greška rešenja veća pri istoj apsolutnoj grešci između $(k+1)$-ve i k-te iteracije.

S obzirom na pomenute činjenice različiti se autori odlučuju za različite kriterijume za zaustavljanje iterativnog procesa. Kod većine autora preovlađuje drugi kriterijum.

2.4.6 Implementacija Jacobijevog i Gauss-Seidelovog metoda

Funkcije `jacobiR` i `gaussSeidelR` implementiraju Jacobijevu i Gauss-Seidelovu metodu rešavanja sistema linearnih jednačina.

```
function [x1,err,iter]=jacobiR(A,b,prec,maxIter)
%JACOBIR(A,B,PREC,MAXITER) resava sistem linearnih jednacina
% Ax=B koristeci Jacobijev metod
% argument PREC odredjuje relativnu gresku
% za prestanak procesa, argument MAXITER odredjuje
% gornju granicu broja iteracija iterativnog procesa
% izlazni argument X1 je resenje, ERR je relativna greska resenja
% ITER je broj iteracija koji je upotrebljen za resavanje sistema

% konstrukcija matrice B i slobodnog vektora
D = diag(diag(A)); B1 = tril(A,-1);
B2 = triu(A,1); B = -inv(D)*(B1+B2);
beta = inv(D)*b;

% inicijalizacija iterativnog procesa
x0 = 0;  x1 = beta;
err = [norm(x1-x0)/norm(x1)];
iter = 0;
while norm(x1-x0) >= prec*norm(x1)
    x0 = x1;
    x1 = B*x0+beta;
    err = [err norm(x1-x0)/norm(x1)];
    iter = iter+1;
    if iter > maxIter
        disp('Prekoracen maksimalan dozvoljen broj iteracija');
        break;
    end
end

function [x1,err,iter]=gaussSeidelR(A,b,prec,maxIter)
%GAUSSSEIDELR(A,B,PREC,MAXITER) resava sistem linearnih jednacina
```

```
% Ax=B koristeci Gauss-Seidelov metod
% argument PREC odredjuje relativnu gresku
% za prestanak procesa, argument MAXITER odredjuje
% gornju granicu broja iteracija iterativnog procesa
% izlazni argument X1 je resenje, ERR je relativna greska resenja
% ITER je broj iteracija koji je upotrebljen za resavanje sistema

% konstrukcija matrica
B1PlusD = tril(A); B2 = triu(A,1);
opts.LT = true;

% inicijalizacija iterativnog procesa
x0 = 0; x1 = b;
err = [norm(x1-x0)/norm(x1)];
iter = 0;
while norm(x1-x0) >= prec*norm(x1)
    x0 = x1;
    x1=linsolve(B1PlusD, B2*x0+b,opts);
    err = [err norm(x1-x0)/norm(x1)];
    iter = iter+1;
    if iter > maxIter
        disp('Prekoracen maksimalan dozvoljen broj iteracija');
        break;
    end
end
```

Kako je već napomenuto iterativni metodi za rešavanje sistema linearnih jednačina se koriste i za retke matrice. Primetimo da implementacija koristi matrične operacije umesto `for` petlji. Ovo je od posebnog značaja ako radimo sa retkim matricama. Ako radimo sa retkim matricama, uobičajena implementacija pomoću `for` petlji ima manju efikasnost od upotrebe matričnih operacija, jer upotreba `for` petlji podrazumeva veliki broj množenja nulom.

Funkcija `gaussSeidelR` koristi funkciju `linsolve` za rešavanje donjeg trougaonog sistema jednačina. Ovo podrazumeva rešavanje sistema linearnih jednačina u svakoj iteraciji. Drugi pristup bi bio izračunavanje inverzne matrice van `while` petlje, i zatim njeno korišćenje u `while` petlji u formi matričnog množenja. Iako primamljiv ovaj pristup se izbegava, jer izračunavanje inverzne matrice može da naruši tačnost iterativnog metoda.

Primetimo da su metodi implementirane tako da izlazna promenljiva `err` sadrži veličine $||x^{k+1} - x^k||/||x^{k+1}||$ za sve iteracije. Ovakva implementacija nije neophodna izuzev u slučaju demonstracija, što je nama sada cilj. Uobičajeno je pamtiti samo veličinu $||x^{k+1} - x^k||/||x^k||$ u poslednjoj iteraciji. Ulazni parametar `maxIter` određuje gornju granicu dozvoljenog broja iteracija za rešavanje sistema linearnih jednačina. Ako dođe do prekoračenja ovog broja iterativni proces se prekida.

```
>>A=[4 1 1; 1 4 1; 1 1 4];
>>b=[1 2 3]';
>>[x,err,iter]=jacobiR(A,b,10^(-6),100)
x =
    -7.94728407527145e-008
```

```
         0.333333253860474
         0.666666587193788
err =
  Columns 1 through 3
  1         0.71428571429         0.27263437405
  Columns 4 through 6
  0.15127000746         0.07132388275        0.036658375176
  Columns 7 through 9
  0.018070580315        0.009098714965       0.0045333504464
  Columns 10 through 12
  0.0022706579712       0.0011343309320      0.00056741468298
  Columns 13 through 15
  0.00028364500024      0.00014183808080     7.0915144652e-005
  Columns 16 through 18
  3.5458546191e-005     1.7729029620e-005     8.8645756778e-006
  Columns 19 through 21
  4.4322726219e-006     2.2161401152e-006     1.1080691065e-006
  Column 22
  5.54034791e-007
iter =
    21
```

Primer prikazuje način upotrebe funkcije `jacobiR`. Kao što vidimo, relativna greška reda veličine 10^{-6} se postiže u dvadesetprvoj iteraciji. Ovaj rezultat je očekivan, jer kao što je napomenuto, za ovaj konkretan slučaj je prinos značajnih cifara po iteraciji .3, pa je $.3 \cdot 21 = 6.3$ značajnih cifara u dvadesetprvoj iteraciji.

Glava 3

Nelinearne jednačine

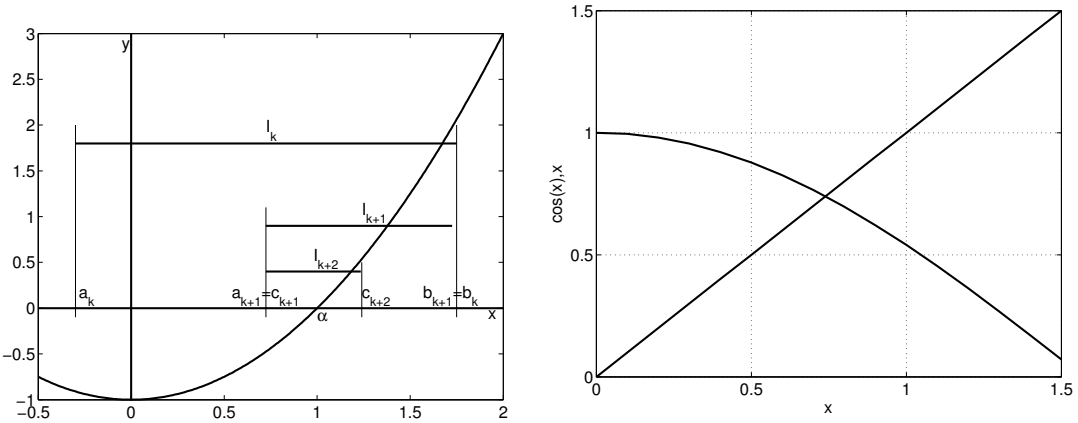
U ovoj glavi problem koji rešavamo je sledeći $f(x) = 0$. Kako funkcija f uglavnom nije linearna to se prethodni problem naziva rešavanje nelinearne jednačine. Ako posmatramo problem u najvećoj opštosti, ne možemo mnogo reći o načinu njegovog rešavanja. Međutim, kako se pokazuje u daljem tekstu, već uslov neprekidnosti funkcije f omogućava formulaciju metoda za rešavanje ovog problema metodom bisekcije. Podrazumevajući diferencijabilnost funkcije f možemo da konstruišemo i mnogo efikasniji metodi kao što je, recimo, Newtonov metod. U drugom odeljku ove glave bavimo se rešavanjem sistema nelinearnih jednačina. Prezentujemo metod Newton-Kantoroviča, gradijentni metod i metod proste iteracije.

3.1 Nelinearne jednačine jedne promenljive

3.1.1 Metod polovljenja intervala (bisekcije)

Ovaj metod je najopštiji i sve što je potrebno za primenu metoda je neprekidnost funkcije f. Pretpostavimo da je funkcija f neprekidna na intervalu $[a, b]$, što označavamo sa $f \in C[a, b]$, i neka je $f(a)f(b) < 0$. Radi jednostavnijeg izlaganja neka je $f(a) < 0$ i $f(b) > 0$. Tada slika 3.1, levo, ilustruje koncept metoda bisekcije. Zbog neprekidnosti funkcije f postoji tačka $\alpha \in [a, b]$ za koju važi $f(\alpha) = 0$.

Postupak je sledeći. Označimo sa $I_0 = [a, b]$. Izaberemo tačku c koja je na sredini intervala I_0. Razmatramo koji od intervala $[a, c]$ ili $[c, b]$ ima ista svojstva kao interval I_0. Ako je ispunjen uslov $f(a)f(c) < 0$, zaključujemo da je nula funkcije sadržana u intervalu $[a, c]$, pa ovaj interval ima ista svojstva kao interval I_0. Ako je $f(c)f(b) < 0$, onda interval $[c, b]$ ima ista svojstva kao interval I_0, pa je nula funkcije unutar intervala $[c, b]$. Kad smo zaključili u kom je intervalu nula funkcije označimo taj interval sa I_1. Primetimo da je dužina intervala I_1 dva puta manja od dužine intervala I_0. Proces nastavljamo iznova nad intervalom I_1 i dobijamo interval I_2, čija je dužina, takođe, dva puta manja od dužine intervala I_1. Ponavljajući proces dobijamo niz intervala I_k, $k = 0, \ldots, n$, sa osobinama $I_{k+1} \subset I_k$, $k = 0, \ldots, n-1$, i dužina intervala I_{k+1} je dva puta manja od dužine intervala I_k. Proces prekidamo onog trenutka kada je dužina intervala I_n manja od unapred zadate tačnosti ε. Nula funkcije f, tačka α, se nalazi unutar intervala I_n. Kako je interval I_n dužine manje od ε, to je aproksimacija vrednosti α sa apsolutnom greškom manjom od ε donja ili gornja granica intervala I_n. Ako odredimo sredinu intervala I_n, dobićemo rešenje sa gornjom granicom apsolutne greške $\varepsilon/2$.



Slika 3.1: Slika levo ilustruje koncept metoda bisekcije, slika desno predstavlja grafike funkcija $y = \cos x$ i $y = x$.

Dužina intervala I_n je određena sa $(b-a)/2^n$. Odavde možemo da procenimo broj iteracija potreban za postizanje apsolutne i relativne greške ε. Važi

$$\frac{b-a}{2^n} < \varepsilon \;\Rightarrow\; n > \log_2 \frac{b-a}{\varepsilon}, \quad \frac{b-a}{2^n|\alpha|} < \varepsilon \;\Rightarrow\; n > \log_2 \frac{b-a}{|\alpha|\varepsilon}. \tag{3.1}$$

Primetimo da potreban broj iteracija zavisi od rastojanja tačaka a i b. Što je dužina početnog intervala veća to će biti potrebno više iteracija da se postigne zadata gornja granica apsolutne greške.

Algoritam koji nalazi tačku α može se opisati na sledeći način
1° izaberemo $c = (a+b)/2$
2° ako je $f(c) = 0$ rešenje je $\alpha = c$, kraj
3° ako je $f(a)f(c) < 0$, uzimamo $b = c$
4° ako je $f(c)f(b) < 0$, uzimamo $a = c$
5° ako je $|a-b| \geq \epsilon$ vraćamo se na korak 1°
6° rešenje je $\alpha = (a+b)/2$, kraj.

Funkcija `bisekcijaR` implementira pomenuti algoritam.

```
function [c,err,iter]=bisekcijaR(fun,a,b,prec,maxIter)
%BISEKCIJAR(FUN,A,B,PREC,MAXITER) racuna nulu jednacine fun(x)=0
% na intervalu [A,B] sa relativnom greskom PREC, parametar MAXITER
% ogranicava maksimalni moguci broj iteracija, ulazni parametar FUN
% predstavlja pokazivac na funkciju cija se nula odredjuje,
% funkcija vraca vrednost nule, gresku, i broj iteracija

% ako je ispunjen uslov u if naredbi metod bisekcije ne konvergira
if fun(a)*fun(b) >= 0
    error('fun(a)*fun(b)>=0 pogresni podaci');
end

iter = 0;err = [abs(b-a)/abs(a)];c = (a+b)/2;
```

```
while abs(b-a) >= abs(a)*prec
    fc = fun(c);
    err = [err abs(a-c)/abs(c)];
    iter = iter+1;
    if fc == 0
        return;
    end
    if fun(a)*fc < 0
        b = c;
    elseif fc*fun(b) < 0
        a = c;
    else
        error('nula funkcije izgubljena');
    end
    c = (a+b)/2;
    if iter > maxIter
        disp('maksimalan broj iteracija dostignut');
        break;
    end
end
```

Ilustrujmo primenu funkcije `bisekcijaR` za nalaženje rešenja jednačine $f(x) = x - \cos x = 0$. Kako je $f(0) = 0 - \cos 0 = 0 - 1 = -1$ i $f(\pi/2) = \pi/2 - \cos \pi/2 = \pi/2$, zaključujemo da postoji barem jedna nula na intervalu $[0, \pi/2]$.

```
>>fun=@(x) x-cos(x);
>>[x,err,iter]=bisekcijaR(fun,0,pi/2,10^(-6),100)
x =
         0.739085500757726
err =
  Columns 1 through 3
     Inf                  Inf                  1
  Columns 4 through 6
     0.33333333333        0.14285714286        0.066666666667
  Columns 7 through 9
     0.033333333333       0.016666666667       0.0083333333333
  Columns 10 through 12
     0.0041666666667      0.0020790020790      0.0010384215992
  Columns 13 through 15
     0.00051894135963     0.00025947067981     0.00012973533991
  Columns 16 through 18
     6.4863462412e-005    3.2430679423e-005    1.6215076779e-005
  Columns 19 through 21
     8.1074726577e-006    4.0537363289e-006    2.0268681645e-006
  Columns 22 through 23
     1.0134340822e-006    5.0671704108e-007
iter =
    22
```

Kao što vidimo broj potrebnih iteracija je 22, a teorijska ocena broja iteracija, nejednakost (3.1), u ovom slučaju daje granicu

$$n > \log_2 \frac{\pi/2 - 0}{10^{-6} \cdot 0.74} \approx 21.02,$$

što je odlično slaganje.

Metod bisekcije konvergira ka rešenju jednačine i pod uslovom da početni interval $[a, b]$ sadrži više od jedne nule polazne funkcije. Zavisno od položaja nula u odnosu na krajnje tačke intervala dobićemo jednu od nula funkcije kao rešenje. Sledeći primer ilustruje rečeno.

```
>>fun=@(x) (x-1)*(x-2)*(x-3);
>> x=bisekcijaR(fun,0,10,10^(-14),100)
x =
          2.99999999999999
>> x=bisekcijaR(fun,-10,5,10^(-14),100)
x =
         0.999999999999994
```

Ovde je funkcija `bisekcijaR` primenjena na rešavanje jednačine $f(x) = (x-1)(x-2)(x-3) = 0$. Kad iterativni proces započnemo sa intervalom $[0, 10]$ dobijemo rešenje 3. Sa druge strane, kad proces započnemo sa intervalom $[-10, 5]$ dobijemo rešenje 1.

Pomenimo na kraju da metod bisekcije uvek konvergira pod uslovom da funkcija u nuli menja znak.

3.1.2 Uslov za izlazak iz while petlje

Interesantno je ponašanje funkcije `bisekcijaR` u slučaju kad je rešenje jednačine jednako nula. Primetimo da uslov za izlazak iz `while` petlje podrazumeva vrednost relativne greške manje od zadate vrednosti parametra `prec`. Kako za broj 0 nema smisla definisati relativnu grešku, funkcija se ponaša neočekivano ako je rešenje jednačine baš jednako nula. Na primer, primenimo funkciju na rešavanje jednačine $\sin x = 0$, u okolini tačke nula.

```
>>[x,err,iter]=bisekcijaR(@sin,-pi/3,pi/2,10^(-6),100)
maksimalan broj iteracija dostignut
x =
     1.47388897796033e-031
err =
  Columns 1 through 3
            2.5                   1.25           1.666666666667
  .
  .
  .
  Columns 100 through 102
  1.399939383997          2.332996611778          1.166498305889
iter =
   101
```

Nismo prikazali celokupan izlaz funkcije `bisekcijaR`, pri čemu mislimo na vrednost promenljive `err`. Kao što vidimo iterativni proces konvergira ka vrednosti nula. Apsolutna greška aproksimacije rešenja je mala, reda veličine 10^{-31}. Međutim, kriterijum za izlazak iz `while` petlje je

iskazan relativnom greškom koja sve vreme izračunavanja ima veliku vrednost. Ovakvo ponašanje je posledica činjenice da za broj nula ne možemo definisati relativnu grešku.

Da bismo izbegli ovakvo ponašanje funkcije obično se uslov `while` petlje menja na sledeći način

```
abs(b-a) >= max(abs(a)*prec, acc)
```

gde je `acc` još jedan ulazni parametar funkcije. Na ovaj način se izbegava pomenuto ponašanje, jer kad apsolutna greška postane manja od ulaznog parametra `acc` smatra se da je ispunjen uslov konvergencije. Naravno, ovde nije ispunjen uslov konvergencije u smislu relativne greške već apsolutne. Modifikovana funkcija `bisekcijaR` izgleda ovako

```
function [c,err,iter]=bisekcijaR(fun,a,b,prec,acc,maxIter)
%BISEKCIJAR(FUN,A,B,PREC,MAXITER) racuna nulu jednacine fun(x)=0
% na intervalu [A,B] sa relativnom greskom PREC, parametar MAXITER
% ogranicava maksimalni moguci broj iteracija, ulazni parametar FUN
% predstavlja pokazivac na funkciju cija se nula odredjuje,
% funkcija vraca vrednost nule, gresku, i broj iteracija

% ako je ispunjen uslov u if naredbi metod bisekcije ne konvergira
if fun(a)*fun(b) >= 0
    error('fun(a)*fun(b)>=0 pogresni podaci');
end

iter = 0;
c = (a+b)/2;
err = [abs(c-a)/abs(c)];
while abs(b-a) >= max(abs(a)*prec,acc)
    fc = fun(c);
    err = [err abs(a-c)/abs(c)];
    iter = iter+1;
    if fc == 0
        return;
    end
    if fun(a)*fc < 0
        b = c;
    elseif fc*fun(b) < 0
        a = c;
    else
        error('nula funkcije izgubljena');
    end
    c = (a+b)/2;
    if iter > maxIter
        disp('maksimalan broj iteracija dostignut');
        break;
    end
end
```

Ako primenimo ovako definisanu funkciju na prethodni problem dobijemo nešto izmenjeno ponašanje.

```
>>[x,err,iter]=bisekcijaR(@sin,-pi/3,pi/2,10^(-6),10^(-14),100)
x =
     -5.94818801116956e-14
err =
  Columns 1 through 2
                         5                         5
  .
  .
  .
  Column 43
           1.666873925436
iter =
    42
```

Kao što vidimo sada postižemo konvergenciju u 42 iteracije, i to zahvaljujući novom uslovu za izlazak iz `while` petlje.

Međutim, postoji i cena. Pretpostavimo da želimo da odredimo nulu funkcije

$$f(x) = (x - .5 \cdot 10^{-15})^3.$$

Ako primenimo modifikovanu funkciju `bisekcijaR` sa parametrom `acc` postavljenim na vrednost 10^{-14}, dobićemo

```
>>f=@(x) (x-.5*10^(-15))^3;
>>[x,err,iter]=bisekcijaR(f,-pi/3,pi/2,10^(-6),10^(-14),100)
x =
      9.74507187120641e-16
err =
  Columns 1 through 2
                         5                         5
  Columns 3 through 4
         1.66666666666667                         5
  .
  .
  .
  Column 49
         1.65350846438908
iter =
    48
```

Kao što vidimo nemamo nijednu značajnu cifru u rešenju. Ovo je posledica činjenice da je vrednost ulaznog parametra `acc` prevelika. Parametar `acc` treba shvatiti kao okolinu broja nula (−`acc`,`acc`) u kojoj se svi brojevi smatraju nulom. Rešenje jednačine koje pripada intervalu (−`acc`,`acc`) neće biti određeno ni sa jednom značajnom cifrom, već se njegova vrednost smatra nulom.

Ako želimo da rešenje prethodne jednačine odredimo sa zadatom relativnom greškom `prec`, moramo smanjiti vrednost parametra `acc` ispod vrednosti traženog rešenja za onoliko redova veličine koliku granicu relativne greške tražimo. Recimo, kako je kod nas rešenje reda veličine 10^{-16}, i kako zahtevamo relativnu grešku reda veličine 10^{-6}, to bi vrednost parametra `acc` morala iznositi najviše $10^{-16} \cdot 10^{-6} = 10^{-22}$. Na ovaj način dobijamo

```
>>f=@(x) (x-.5*10^(-15))^3;
>>[x,err,iter]=bisekcijaR(f,-pi/3,pi/2,10^(-6),10^(-22),100)
x =
      5.00000051625905e-16
err =
  Columns 1 through 2
                         5                         5
  Columns 3 through 4
         1.66666666666667                         5
  .
  .
  .
  Columns 69 through 70
      1.77401901441859e-05      8.87017375127884e-06
  Columns 71 through 72
      4.43506720573106e-06      2.21752868561837e-06
  Columns 73 through 74
      1.10876311345218e-06      5.54381864065171e-07
iter =
    73
```

Ovako su uglavnom implementirane sve funkcije koje se bave numeričkim određivanjem neke vrednosti iterativnim procesima. Tokom kursa videćemo da je to slučaj gotovo sa svim funkcijama koje su implementirane u `Matlab`u.

Parametar `acc` se na engleskom naziva accuracy, dok se parametar `prec` na engleskom naziva precision. Na srpskom prevod imena parametra `acc` je tačnost, dok je prevod imena parametra `prec` na srpskom preciznost. Međutim, imamo na umu da `acc` znači maksimalnu dozvoljenu apsolutnu grešku u rešenju, dok `prec` znači maksimalnu dozvoljenu relativnu grešku u rešenju.

Mi ćemo dalje u knjizi koristiti ovakav uslov za izlazak iz `while` petlje, jer je široko prihvaćen u gotovo svakom profesionalnom softwareskom paketu koji se bavi numeričkim izračunavanjima.

3.1.3 Metod proste iteracije

Jednačina $f(x) = 0$ se može na više načina zapisati i u sledećem obliku $x = \phi(x)$. Na primer, $x = x + f(x)$ ili za $\lambda \in \mathbb{R} \setminus \{0\}$, možemo pisati $x = x + \lambda f(x)$, i slično. U ovom odeljku bavimo se rešavanjem nelinearne jednačine oblika $x = \phi(x)$. Pokazaćemo da je posmatranje jednačine u ovom obliku pogodnije za formulisanje rezultata vezanih za karakteristike procesa koji se koristi za njeno rešavanje.

Preciznije, pokazaćemo da pod izvesnim uslovima niz definisan na sledeći način

$$x_{k+1} = \phi(x_k), \quad k \in \mathbb{N}_0,$$

sa pogodno izabranom početnom vrednošću x_0 konvergira ka rešenju jednačine $x = \phi(x)$. Niz $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, nazivamo iterativni proces, a metod za rešavanje jednačina nazivamo metodom proste iteracije. Funkciju ϕ obično nazivamo iterativna funkcija.

Grafički metod proste iteracije možemo prikazati kao na slici 3.2, desno. Slika prikazuje način na koji konstruisani niz konvergira ka preseku krivih $y = x$ i $y = \phi(x)$. Tačka preseka krivih je rešenje jednačine $x = \phi(x)$.

Radi ilustracije posmatrajmo jednačinu $x = \cos x$. Ovu ste jednačinu verovatno već rešili koristeći digitron. Naime, ako vam je digitron podešen za rad u radijanima, do rešenja ove jednačine možete doći jednostavnim izračunavanjem funkcije cos više puta polazeći od broja nula. Sledeći skript u `Matlabu` ilustruje opisani postupak.

```
x=[0];
for i=1:20
    x=[x cos(x(end))];
end
x

x =
  Columns 1 through 7
         0    1.0000    0.5403    0.8576    0.6543    0.7935    0.7014
  Columns 8 through 14
    0.7640    0.7221    0.7504    0.7314    0.7442    0.7356    0.7414
  Columns 15 through 21
    0.7375    0.7401    0.7384    0.7396    0.7388    0.7393    0.7389
```

Skript generiše vrednosti niza $x_{k+1} = \cos x_k$, $k \in \mathbb{N}_0$, $x_0 = 0$. Vidimo da vrednosti niza konvergiraju ka .7390 što je u stvari rešenje jednačine. Slika 3.1, desno, prikazuje grafike funkcija $y = \cos x$ i $y = x$. Tačno rešenje u formatu dvostruke preciznosti iznosi $\alpha = .739085133215161$.

Sledeća teorema ilustruje koje uslove može da zadovolji funkcija ϕ da bi ovako konstruisan niz konvergirao ka rešenju jednačine $x = \phi(x)$.

Teorema 3.1 *Neka je $\phi : [a, b] \to [a, b]$ i neka je $\phi \in C^1[a, b]$. Ako je*

$$|\phi'(x)| \leq q < 1, \quad x \in (a, b),$$

onda postoji tačno jedno $\alpha \in [a, b]$ tako da važi $\alpha = \phi(\alpha)$ i za proizvoljno $x_0 \in [a, b]$, niz $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, konvergira ka α.

Važe sledeće ocene

$$|x_{k+1} - \alpha| \leq q|x_k - \alpha|, \quad k \in \mathbb{N}_0,$$
$$\lim_{k \to +\infty} \frac{x_{k+1} - \alpha}{x_k - \alpha} = \phi'(\alpha).$$

Dokaz. U dokazu teoreme koristićemo Lagrangeovu teoremu, primenjenu na funkciju ϕ

$$\phi(x) - \phi(y) = \phi'(\psi)(x - y), \quad x, y \in [a, b],$$

gde je $\psi \in (\min\{x, y\}, \max\{x, y\})$, i direktne posledice ove teoreme i uslova koje zadovoljava funkcija ϕ

$$|\phi(x) - \phi(y)| = |\phi'(\psi)||x - y| \leq q|x - y| < |x - y|.$$

Funkcije koje zadovoljavaju ovaj uslov nazivamo kontrakcijama.

Pokazaćemo da je niz $x_{k+1} = \phi(x_k)$ Cauchyev. Primetimo prvo da važi sledeća ocena

$$|x_{n+1} - x_n| = |\phi(x_n) - \phi(x_{n-1})| \leq q|x_n - x_{n-1}| \leq \cdots \leq q^n|x_1 - x_0|.$$

Izaberimo proizvoljne $m, n \in \mathbb{N}$, i $m > n$, onda je

$$\begin{aligned}|x_m - x_n| &= |x_m - x_{m-1} + x_{m-1} - x_{m-2} + \cdots + x_{n+1} - x_n| \\ &\leq |x_m - x_{m-1}| + |x_{m-1} - x_{m-2}| + \cdots + |x_{n+1} - x_n| \\ &\leq q^{m-1}|x_1 - x_0| + q^{m-2}|x_1 - x_0| + \cdots + q^n|x_1 - x_0| \\ &= q^n(q^{m-1-n} + \cdots + 1)|x_1 - x_0| = q^n \frac{1 - q^{m-n}}{1 - q}|x_1 - x_0| \\ &\leq \frac{q^n}{1 - q}|x_1 - x_0|.\end{aligned}$$

Dovoljno je izabrati $n_0 \in \mathbb{N}$, tako da je ispunjen uslov $q^{n_0}/(1 - q)|x_1 - x_0| < \varepsilon$, da bismo pokazali da je niz Cauchyev. Možemo, recimo, odabrati

$$n_0 = \left[\frac{\log((1 - q)\varepsilon/|x_1 - x_0|)}{\log q}\right] + 1.$$

Kako je interval $[a, b]$ kompletan metrički prostor, to u prostoru $[a, b]$ svaki Cauchyev niz konvergira, pa konvergira i niz $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, $x_0 \in [a, b]$. Označimo granicu sa $\alpha \in [a, b]$. Onda je

$$\alpha = \lim_{k \to +\infty} x_{k+1} = \lim_{k \to +\infty} \phi(x_k) = \phi\left(\lim_{k \to +\infty} x_k\right) = \phi(\alpha),$$

gde smo koristili činjenicu da je funkcija ϕ neprekidna. Vidimo da je α rešenje jednačine $x = \phi(x)$.

Neka su $\alpha \in [a, b]$ i $\beta \in [a, b]$ dva rešenja jednačine. Onda važi $\alpha = \phi(\alpha)$ i $\beta = \phi(\beta)$, i

$$|\alpha - \beta| = |\phi(\alpha) - \phi(\beta)| = |\phi'(\psi)(\alpha - \beta)| \leq q|\alpha - \beta| < |\alpha - \beta|,$$

što predstavlja kontradikciju. Zaključujemo da je α jedino rešenje.

Da bismo pokazali prvu ocenu koristimo se već korišćenim tehnikama. Tako dobijamo

$$|x_{k+1} - \alpha| = |\phi(x_k) - \phi(\alpha)| \leq q|x_k - \alpha|, \quad k \in \mathbb{N}_0.$$

Da bismo pokazali drugu ocenu koristimo diferencijabilnost funkcije ϕ. Imamo

$$\phi(x_k) = \phi(\alpha) + \phi'(\alpha)(x_k - \alpha) + \omega(x_k - \alpha)(x_k - \alpha),$$

gde funkcija ω teži nuli kad njen argument teži nuli. Koristeći činjenicu da je $\phi(\alpha) = \alpha$ i $\phi(x_k) = x_{k+1}$, dobijamo

$$\frac{x_{k+1} - \alpha}{x_k - \alpha} = \phi'(\alpha) + \omega(x_k - \alpha).$$

Ako potražimo graničnu vrednost leve i desne strane, nalazimo

$$\lim_{k \to +\infty} \frac{x_{k+1} - \alpha}{x_k - \alpha} = \phi'(\alpha) + \lim_{k \to +\infty} \omega(x_k - \alpha) = \phi'(\alpha).$$

Ovim smo završili dokaz teoreme. □

Veličina $\phi'(\alpha)$ se naziva asimptotska konstanta greške, jer opisuje asimptotsko ponašanje iterativnog procesa.

Primenimo teoremu na rešavanje jednačine $x = \cos x$. Ovde je $\phi(x) = \cos x$. Kao što znamo funkcija cos pripada klasi elementarnih funkcija pa je neprekidno diferencijabilna proizvoljan broj

puta u svim tačkama u kojima je definisana, a definisana je na celoj realnoj pravoj. Ostaje da identifikujemo interval $[a,b]$ za koji važi $\cos : [a,b] \to [a,b]$ i gde je $|\sin x| \leq q < 1$, $x \in (a,b)$. Međutim, ako pogledamo grafike funkcija $y = \cos x$ i $y = x$, slika 3.1 desno, vidimo da se rešenje nalazi u okolini tačke .7, tako da možemo izabrati $a = 0$ i $b = 1$. Lako proveravamo da je $\cos a = \cos 0 = 1$ i $\cos b = \cos 1 \approx .54$, tako da je ispunjen uslov $\cos : [0,1] \to [0,1]$. Takođe, na intervalu $[0,1]$ izvod funkcije cos je ograničen, jer je $|(\cos x)'| = |-\sin x| \leq \sin 1 < .85 = q$. Na osnovu iznetog moguće je primeniti teoremu 3.1 i zaključiti da na intervalu $[0,1]$ postoji jedinstveno rešenje jednačine $x = \cos x$.

Ako primenimo prvu ocenu iz teoreme 3.1 možemo zaključiti koliki je prinos značajnih cifara u jednoj iteraciji. Naime, podelimo li pomenutu nejednakost sa $|\alpha|$ dobijamo

$$\frac{|x_{k+1} - \alpha|}{|\alpha|} \leq q \frac{|x_k - \alpha|}{|\alpha|},$$

a odavde

$$-\log_{10} \frac{|x_{k+1} - \alpha|}{|\alpha|} \geq -\log_{10} q - \log_{10} \frac{|x_k - \alpha|}{|\alpha|}.$$

Broj značajnih cifara koji dobijamo u jednoj iteraciji iznosi $-\log_{10} q \approx .075$. Potrebno nam je $1/.075 \approx 13.3$ iteracija za jednu značajnu cifru. Naravno, ovo je gornja granica. Kako vidimo iz stvarnog ponašanja procesa, proces se ponaša malo bolje.

Bolje ponašanje procesa možemo objasniti drugom ocenom. Naime, iz druge ocene dobijamo

$$\frac{|x_{k+1} - \alpha|}{|\alpha|} \approx |\phi'(\alpha)| \frac{|x_k - \alpha|}{|\alpha|} = \sin\alpha \frac{|x_k - \alpha|}{|\alpha|}.$$

Ova ocena je tim tačnija što je veća vrednost k. Sa ovom ocenom prinos značajnih cifara u jednoj iteraciji iznosi

$$-\log_{10} \sin .739085133215161 \approx .17.$$

Na osnovu ove ocene možemo zaključiti da je potrebno $1/.17 \approx 5.9$ iteracija za dobijanje jedne značajne cifre, što se otprilike i vidi na osnovu rezultata koji je generisan skriptom.

3.1.4 Red konvergencije metoda proste iteracije

Ono što je od posebnog interesa je ponašanje iterativnog procesa metoda proste iteracije u slučaju kada je ispunjen uslov $\phi'(\alpha) = 0$. U ovom slučaju prinos značajnih cifara u iteraciji iznosi $-\log_{10} 0 = +\infty$, što je naravno preterano. Postaje očigledno da je u ovom slučaju potrebna teorema koja preciznije opisuje ponašanje iterativnog procesa.

Teorema 3.2 *Neka je* $\phi : [a,b] \to [a,b]$, $\phi \in C^r[a,b]$ *i neka je*

$$|\phi'(x)| \leq q < 1, \quad x \in (a,b).$$

Postoji tačno jedno $\alpha \in (a,b)$ *tako da važi* $\alpha = \phi(\alpha)$, *i za proizvoljno* $x_0 \in [a,b]$, *niz* $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, *konvergira ka* α.

Ako je $\phi'(\alpha) = \cdots = \phi^{(r-1)}(\alpha) = 0$, *onda je*

$$\lim_{k \to +\infty} \frac{x_{k+1} - \alpha}{(x_k - \alpha)^r} = \frac{1}{r!} \phi^{(r)}(\alpha).$$

Dokaz. Dokaz egzistencije i jedinstvenosti rešenja ostaje potpuno isti kao i u teoremi 3.1, zato taj deo dokaza izostavljamo. Dajemo samo dokaz poslednje jednakosti koja je nama od značaja.

Koristeći činjenicu da je funkcija ϕ neprekidno diferencijabina reda r, možemo pisati

$$\phi(x_k) = \phi(\alpha) + \frac{\phi'(\alpha)}{1!}(x_k-\alpha) + \cdots + \frac{\phi^{(r-1)}(\alpha)}{(r-1)!}(x_k-\alpha)^{r-1} + \frac{\phi^{(r)}(\alpha)}{r!}(x_k-\alpha)^r + \omega(x_k-\alpha)(x_k-\alpha)^r,$$

gde funkcija ω teži nuli kad argument funkcije teži nuli. Koristeći činjenicu da je $\phi^{(\ell)}(\alpha) = 0$, $\ell = 1, \ldots, r-1$, $\phi(x_k) = x_{k+1}$, $\phi(\alpha) = \alpha$, nalazimo

$$x_{k+1} = \alpha + \frac{\phi^{(r)}(\alpha)}{r!}(x_k-\alpha)^r + \omega(x_k-\alpha)(x_k-\alpha)^r,$$

a odavde dobijamo

$$\lim_{k\to+\infty} \frac{x_{k+1}-\alpha}{(x_k-\alpha)^r} = \frac{1}{r!}\phi^{(r)}(\alpha) + \lim_{k\to+\infty}\omega(x_k-\alpha) = \frac{1}{r!}\phi^{(r)}(\alpha).$$

Ovim je dokaz teoreme završen. □

Evo primera jednog iterativnog procesa koji ima navedene osobine. Posmatrajmo sledeću jednačinu

$$x = \phi(x) = x\frac{x^2+6}{3x^2+2}.$$

Pored trivijalnog rešenja $x = 0$, posle kraćeg izračunavanja dobijamo da su rešenja i $x_{1,2} = \pm\sqrt{2}$. Funkcija ϕ je rastuća, što proveravamo jednostavno

$$\phi'(x) = 3\frac{(x^2-2)^2}{(3x^2+2)^2} \geq 0.$$

Vidimo da je $\phi'(\sqrt{2}) = 0$. Da bismo opravdali primenu teoreme 3.2, možemo postupiti na sledeći način. Izaberimo neku okolinu tačke $\sqrt{2}$, recimo $[\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon]$ za koju važi $|\phi'(x)| \leq 1/2$. U ovoj okolini tačke $\sqrt{2}$ funkcija $\phi(x)$ sporije raste od funkcije $y = x$, jer je izvod funkcije ϕ u toj okolini manji ili jednak $1/2$, dok je izvod funkcije $y = x$ jednak 1. Na osnovu iznetog možemo zaključiti da će biti ispunjeni uslovi $\sqrt{2}-\varepsilon \leq \phi(\sqrt{2}-\varepsilon)$ i $\phi(\sqrt{2}+\varepsilon) \leq \sqrt{2}+\varepsilon$, ili $\phi : [\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon] \to [\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon]$. Funkcija ϕ je elementarna funkcija i definisana na celoj realnoj pravoj, pa je neprekidno diferencijabilna proizvoljnog reda. Takođe, na osnovu izbora intervala $[\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon]$ zaključujemo da važi $|\phi'(x)| \leq 1/2$, $x \in [\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon]$. Na osnovu teorema 3.1 ili 3.2 zaključujemo da iterativni proces $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, konvergira ka vrednosti $\sqrt{2}$, ako je ispunjen uslov $x_0 \in [\sqrt{2}-\varepsilon, \sqrt{2}+\varepsilon]$. Za tačku x_0 možemo izabrati bilo koje rešenje nejednačine

$$\phi'(x) = 3\frac{(x^2-2)^2}{(3x^2+2)^2} \leq \frac{1}{2} \Rightarrow x^2-2 \leq \frac{1}{\sqrt{6}}(3x^2+2) \Rightarrow \left(1-\sqrt{\frac{3}{2}}\right)x^2 - 2\left(1+\frac{1}{\sqrt{6}}\right) \leq 0.$$

Zbog operacije korenovanja, koju smo primenili, poslednja nejednakost važi za $x \geq \sqrt{2}$. Međutim, kako vidimo, poslednja nejednakost je ispunjena za sve vrednosti $x \geq \sqrt{2}$, tako da za x_0 možemo izabrati recimo 2.

Kako smo već pomenuli važi $\phi'(\sqrt{2}) = 0$, ako odredimo drugi i treći izvod dobijamo

$$\phi''(x) = 96x\frac{x^2-2}{(3x^2+2)^3}, \quad \phi^{(3)}(x) = -96\frac{9x^4-36x^2+4}{(3x^2+2)^4}.$$

Odavde izračunavamo $\phi''(\sqrt{2}) = 0$ i $\phi^{(3)}(\sqrt{2}) = 3/4$. Primenom teoreme 3.2, dobijamo

$$\lim_{k\to+\infty}\frac{x_{k+1}-\alpha}{(x_k-\alpha)^3} = \frac{1}{3!}\phi^{(3)}(\alpha) = \frac{1}{8}.$$

Ocenimo sada asimptotski priraštaj značajnih cifara u jednoj iteraciji. Za dovoljno veliku vrednost k, približno važi

$$x_{k+1}-\alpha \approx \frac{1}{8}(x_k-\alpha)^3 \Rightarrow \frac{|x_{k+1}-\alpha|}{|\alpha|} \approx \frac{|\alpha|^2}{8}\frac{|x_k-\alpha|^3}{|\alpha|^3}.$$

Odavde dobijamo da je prinos značajnih cifara u jednoj iteraciji

$$\begin{aligned} -\log_{10}\frac{|x_{k+1}-\alpha|}{|\alpha|} &\approx -\log_{10}\frac{|\alpha|^2}{8} - 3\log_{10}\frac{|x_k-\alpha|}{|\alpha|}, \\ -\log_{10} r_{k+1} - (-\log_{10} r_k) &\approx -\log_{10}\frac{|\alpha|^2}{8} - 2\log_{10} r_k \approx .6 - 2\log_{10} r_k. \end{aligned} \tag{3.2}$$

Ovo je prilično zanimljiv izraz, jer pokazuje da prilikom svake iteracije prinos značajnih cifara nije linearan, već priraštaj broja značajnih cifara iznosi bar dva puta broj značajnih cifara u prethodnoj iteraciji. Zanemarićemo prvi sabirak .6 da bismo lakše sproveli analizu. Ako u prvoj iteraciji imamo 2 značajne cifre, onda će druga iteracija imati $2 + 2 \cdot 2 = 6$ značajnih cifara, sledeća iteracija će imati $6 + 2 \cdot 6 = 18$ značajnih cifara. Dalje nema smisla ni izračunavati, jer format dvostruke preciznosti podržava samo 16 dekadnih cifara.

Sledeći skript ilustruje ponašanje ovog iterativnog procesa.

```
x=[2];
for i=1:4
    x=[x x(end)*(x(end)*x(end)+6)/(3*x(end)*x(end)+2)];
end
x

x =
  Columns 1 through 3
   2.000000000000000   1.428571428571429   1.414213926776741
  Columns 4 through 5
   1.414213562373095   1.414213562373095
```

Vidimo da 16 značajnih cifara imamo već u trećoj iteraciji. Četvrta iteracija je izvedena samo da potvrdi ovu činjenicu.

Očigledno je da vrednost r iz teoreme 3.2 bitno određuje broj iteracija iterativnog procesa koji je potreban da se dostigne mašinska preciznost. Kako smo videli na primeru jednačine $x = \cos x$, gde je $r = 1$, dvadeset iteracija nam nije bilo dovoljno da dobijemo četiri značajne cifre. Sa druge strane, u slučaju jednačine $x = x(x^2+6)/(3x^2+2)$, dovoljno je bilo uraditi tri iteracije i dostići svih šesnaest značajnih cifara. Zato je broj r iz teoreme 3.2 bitna karakteristika procesa.

Definicija 3.1 *Neka funkcija ϕ zadovoljava uslove teoreme* 3.2*, i neka je dat iterativni proces $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, pri čemu niz x_k konvergira ka rešenju α jednačine $x = \phi(x)$.*

Iterativni proces ima red konvergencije r ako i samo ako je

$$\phi'(\alpha) = \cdots = \phi^{(r-1)}(\alpha) = 0, \quad \phi^{(r)}(\alpha) \neq 0.$$

Ako iterativni proces ima red konvergencije r, onda veličinu

$$\lim_{k\to+\infty} \frac{x_{k+1} - \alpha}{(x_k - \alpha)^r} = \frac{1}{r!}\phi^{(r)}(\alpha) \neq 0,$$

nazivamo asimptotska konstanta greške.

Pomenuti iterativni metod za određivanje broja $\sqrt{2}$ je samo specijalan slučaj jednog opštijeg metoda trećeg reda koji se može koristiti za određivanje c-tog korena iz pozitivnog broja a. Ovaj opšti iterativni proces izgleda ovako

$$x_{k+1} = \phi(x_k) = x_k \frac{(c-1)x_k^c + (c+1)a}{(c+1)x_k^c + (c-1)a}, \quad k \in \mathbb{N}_0. \tag{3.3}$$

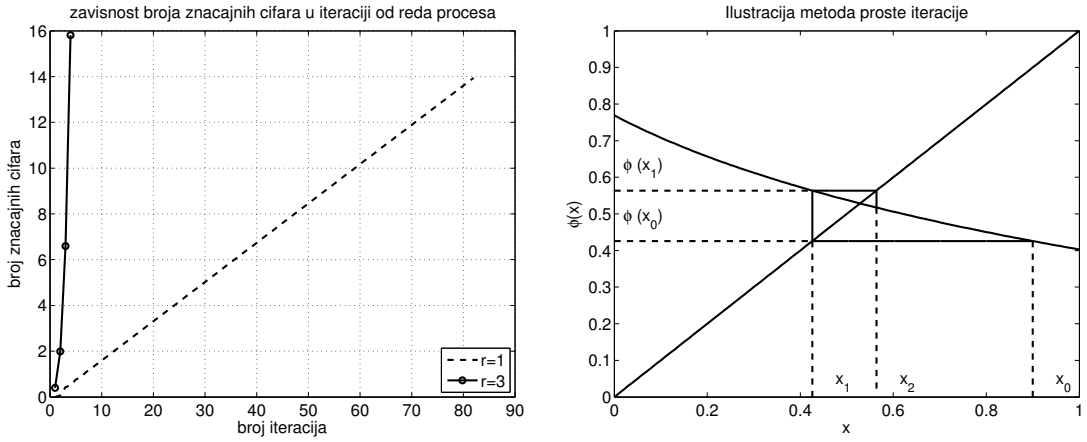
Može se pokazati da važi

$$\phi'(a^{1/c}) = \phi''(a^{1/c}) = 0, \quad \phi^{(3)}(a^{1/c}) = \frac{c^2-1}{2a^{2/c}}.$$

Jednostavno vidimo da je proces trećeg reda i da je asimptotska konstanta greške

$$\frac{1}{3!}\phi^{(3)}(a^{1/c}) = \frac{1}{12}\frac{c^2-1}{a^{2/c}}.$$

Ovaj iterativni proces možemo koristiti za izračunavanje bilo kog korena iz pozitivnog broja, pri čemu je proces vrlo efikasan.



Slika 3.2: Slika levo ilustruje broj značajnih cifara u iteracijama u procesima $x_{k+1} = \cos x_k$, $x_0 = 0$ i $x_{k+1} = x_k(x_k^2 + 6)/(3x_k^2 + 2)$, $x_0 = 2$, slika desno ilustruje metod proste iteracije.

Funkcija koja u `Matlab`u implementira metod proste iteracije može se zadati na sledeći način

```
function [x0,err,iter]=prostaIteracija(fun,x0,prec,acc,maxIter)
%PROSTAITERACIJA(FUN,X0,PREC,ACC,MAXITER) konstruise niz
% x(k+1)=FUN(x(k))
% koristeci X0 kao pocetnu vrednost, argument PREC odredjuje
% relativnu gresku pri kojoj se prekida iterativni proces,
% parametar ACC odredjuje apsolutnu gresku
% pri kojoj se prekida iterativni proces
% argument MAXITER oznacava maksimalni dozvoljen broj iteracija,
% funkcija vraca izracunatu vrednost resenja ili
% vrednost poslednjeg izracunatog elementa niza,
% drugi argument je relativna greska u svakoj iteraciji
% treci argument je broj upotrebljenih iteracija

iter = 0; err=[]; deltaX = Inf;
while abs(deltaX) >= max(prec*abs(x0), acc)
    deltaX = x0;
    x0 = fun(x0);
    deltaX = x0 - deltaX;
    iter = iter+1;
    err = [err abs(deltaX)/abs(x0)];
    if iter>maxIter
        disp('maksimalni dozvoljeni broj iteracija dostignut');
        break;
    end
end
```

Na primer, ako želimo da odredimo π-ti koren iz broja e, možemo postupiti na sledeći način

```
>>fun=@(x) x*((pi-1)*x^pi+(pi+1)*exp(1))/((pi+1)*x^pi+(pi-1)*exp(1));
>>[x,err,iter]=prostaIteracija(fun,exp(1),10^(-14),10^(-20),10)
x =
   1.374802227439359
err =
  Columns 1 through 3
   0.671613603507139   0.178783506733509   0.003423561569352
  Column 4 through  5
   0.000000029507836                   0
iter =
     5
>>exp(1/pi)
ans =
   1.374802227439359
```

Primećujemo da proces jako brzo konvergira. Dovoljno je pet iteracija. Pri tome prva i druga iteracija se koriste samo za pronalaženje dobre startne vrednosti, i u suštini, ne predstavljaju iteracije koje doprinose prinosu značajnih cifara.

Još jednu ilustraciju uticaja reda konvergencije na broj značajnih cifara ilustruje slika 3.2, levo. Na slici je prikazan grafik zavisnosti broja značajnih cifara u iterativnim procesima $x_{k+1} = \cos x_k$, $k \in \mathbb{N}_0$, $x_0 = 0$, i $x_{k+1} = x_k(x_k^2 + 6)/(3x_k^2 + 2)$, $k \in \mathbb{N}_0$, $x_0 = 2$, u funkciji broja iteracija.

Linija koja ima mnogo veći nagib odgovara drugom iterativnom procesu. Kao što vidimo broj značajnih cifara raste neuporedivo brže kad je red metoda veći. Zato se uglavnom teži konstrukciji metoda koji imaju veliki red konvergenije. Međutim, karakteristika metoda sa velikim redom konvergencije je da su iterativne funkcije uglavnom zadate prilično komplikovanim izrazima. U pojedinim slučajevima, ovi izrazi mogu postati toliko komplikovani da je 'jeftinije' primenjivati nekoliko iteracija metoda nižeg reda od jedne iteracije metoda višeg reda.

Opišimo i uticaj veličine asimptotske konstante greške na prinos značajnih cifara u metodu proste iteracije. Na osnovu formule za prinos značajnih cifara, jednačina (3.2), vidimo da proizvod asimptotske konstante greške i veličine rešenja koje određujemo ima uticaja na prinos značajnih cifara u iteraciji. Da bismo ilustrovali ovo ponašanje, primenimo iterativni proces dat jednačinom (3.3) sa $c = 1000$ za određivanje hiljaditog korena broja 10^{12}. Koristeći činjenicu da je proces reda tri, sa asimptotskom konstantom greške

$$\frac{1}{3!}\phi^{(3)}((10^{12})^{1/1000}) = \frac{1}{12}\frac{1000^2-1}{(10^{12})^{2/1000}} \approx \frac{8333.25}{10^{24/1000}},$$

možemo dati izraz za prinos značajnih cifara po iteraciji, na sledeći način

$$\begin{aligned} -\log_{10}\frac{|x_{k+1}-\alpha|}{|\alpha|} - \left(-\log_{10}\frac{|x_k-\alpha|}{|\alpha|}\right) &\approx -\log_{10}\left(\frac{8333.25}{10^{24/1000}}\alpha^2\right) - 2\log_{10}\frac{|x_k-\alpha|}{|\alpha|} \\ &\approx -3.92 - 2\log_{10}\frac{|x_k-\alpha|}{|\alpha|}. \end{aligned}$$

Kao što vidimo da bismo dobili više značajnih cifara u $(k+1)$-voj iteraciji moramo u k-toj imati bar dve značajne cifre. Ovo zaključivanje ima zanimljive posledice na ponašanje iterativnog procesa.

```
>>fun=@(x) x*(999*x^1000+1001*10^12)/(1001*x^1000+999*10^12);
>>[x,err,iter]=prostaIteracija(fun,1.01,10^(-14),10^(-20),100)
x =
          1.028016298126474
err =
  Columns 1 through 3
  0.00199800191433       0.00199800137977       0.00199799742982
  Columns 4 through 6
  0.00199796824383       0.00199775260827       0.00199616043582
  Columns 7 through 9
  0.00198445941859       0.00190134201694       0.00142827430261
  Columns 10 through 12
  0.000359912903330      3.9646130485e-006     5.19312626263e-012
  Column 13
  0
iter =
    13
```

Proces konvergencije počinje sa dve značajne cifre i vidimo da je u prvih 9 iteracija relativna greška reda veličine 10^{-3}. Aproksimacije imaju dve do tri značajne cifre. Tek od devete iteracije dolazi do ubrzanja procesa konvergencije i metod postaje reda tri. Primetimo da u jedanaestoj iteraciji imamo pet značajnih cifara, jer je $-3.92 + 3 \cdot 3 \approx 5$, gde je tri broj značajnih cifara u desetoj iteraciji.

Jednostavno je pokazati da u slučaju iterativnog procesa (3.3), broj značajnih cifara u $(k+1)$-voj iteraciji iznosi

$$-\log_{10}\frac{|x_{k+1}-\alpha|}{|\alpha|} \approx -\log_{10}\frac{c^2-1}{12} - 3\log_{10}\frac{|x_k-\alpha|}{|\alpha|}.$$

Vidimo da broj značajnih cifara zavisi od reda korena koji se određuje.

Za generalni iterativni proces reda r, možemo formulisati sledeću teoremu.

Teorema 3.3 *Neka je $x_{k+1} = \phi(x_k)$, $k \in \mathbb{N}_0$, iterativni proces reda r, gde je $\lim x_k = \alpha$. Broj značajnih cifara u $(k+1)$-voj iteraciji približno iznosi*

$$-\log_{10} r_{k+1} \approx -\log_{10}\left|\frac{1}{r!}\phi^{(r)}(\alpha)\alpha^{r-1}\right| - r\log_{10} r_k,$$

gde je $r_k = |\alpha - x_k|/|\alpha|$.

Dokaz. Koristeći teoremu 3.2 znamo da važi

$$\lim_{k\to+\infty}\frac{x_{k+1}-\alpha}{(x_k-\alpha)^r} = \frac{1}{r!}\phi^{(r)}(\alpha).$$

Za dovoljnu veliku vrednost k, možemo pisati

$$|x_{k+1}-\alpha| \approx \left|\frac{1}{r!}\phi^{(r)}(\alpha)\right| |x_k-\alpha|^r.$$

Ako podelimo ovu jednakost sa $|\alpha|$ dobijamo

$$r_{k+1} = \frac{|x_{k+1}-\alpha|}{|\alpha|} \approx \left|\frac{1}{r!}\phi^{(r)}(\alpha)\alpha^{r-1}\right| \frac{|x_k-\alpha|^r}{|\alpha|^r} = \left|\frac{1}{r!}\phi^{(r)}(\alpha)\alpha^{r-1}\right| r_k^r.$$

Ovim je dokaz teoreme završen. □

Vidimo da, ako je konstanta

$$-\log_{10}\left|\frac{1}{r!}\phi^{(r)}(\alpha)\alpha^{r-1}\right|$$

velika, procesu treba dobra startna vrednost, tj. mala relativna greška, da bi došlo do povećanja broja značajnih cifara u narednoj iteraciji.

3.1.5 Newtonov metod

Newtonov metod je jedan od najstarijih poznatih metoda za rešavanje nelinearnih jednačina funkcije jedne promenljive. Metod se primenjuje na rešavanje jednačine $f(x) = 0$, i može se opisati pomoću sledećeg iterativnog procesa

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k \in \mathbb{N}_0,$$

gde je x_0 pogodno izabrana startna vrednost. Kao što vidimo Newtonov metod može se primeniti na rešavanje jednačine $f(x) = 0$ jedino u slučaju kada je funkcija f diferencijabilna u okolini rešenja jednačine.

Generalno, Newtonov metod može se izvesti iz metoda proste iteracije koristeći iterativnu funkciju $\phi(x) = x - f(x)/f'(x)$, što neposredno proveravamo. Koristeći teoriju metoda proste iteracije možemo dati sledeću teoremu o konvergenciji Newtonovog metoda.

Teorema 3.4 *Neka je f dva puta neprekidno diferencijabilna funkcija u okolini tačke α koja predstavlja rešenje jednačine $f(x) = 0$ i neka je $f'(x) \neq 0$ u okolini tačke α. Postoji $\varepsilon > 0$ tako da za $x_0 \in [\alpha - \varepsilon, \alpha + \varepsilon]$ iterativni proces*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k \in \mathbb{N}_0,$$

konvergira ka α. Pri tome važi ocena

$$\lim_{k \to +\infty} \frac{x_{k+1} - \alpha}{(x_k - \alpha)^2} = \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)}.$$

Newtonov iterativni proces ima red konvergencije barem dva, a pod uslovom $f''(\alpha) \neq 0$ iterativni proces ima red konvergencije tačno dva.

Dokaz. Kako je $\phi(x) = x - f(x)/f'(x)$, posmatrajmo izvod funkcije ϕ. Dobijamo

$$\phi'(x) = \frac{f(x)f''(x)}{(f'(x))^2}.$$

Kako je $f(\alpha) = 0$, i kako su sve funkcije f, f' i f'' neprekidne u okolini tačke α, to postoji okolina tačke α u kojoj je $|\phi'(x)| \leq 1/2$. Neka je interval $[\alpha - \varepsilon, \alpha + \varepsilon]$ podskup ove okoline. Posmatrajmo izraz $\phi(x) = x - f(x)/f'(x)$. Kako je ovo neprekidna funkcija, $\phi(\alpha) = \alpha$ i kako je izvod funkcije ϕ manji od $1/2$ u okolini $[\alpha - \varepsilon, \alpha + \varepsilon]$ to je vrednost funkcije ϕ manja od vrednosti funkcije $y = x$ na ovom intervalu, jer ova funkcija brže raste pošto ima izvod 1. Zaključujemo da važi $\phi : [\alpha - \varepsilon, \alpha + \varepsilon] \to [\alpha - \varepsilon, \alpha + \varepsilon]$. Na osnovu teoreme 3.1 zaključujemo da postoji tačno jedno rešenje jednačine $x = \phi(x)$ na intervalu $[\alpha - \varepsilon, \alpha + \varepsilon]$. Naravno, ovo rešenje je baš tačka α. Kako je

$$\phi'(\alpha) = f(\alpha)f''(\alpha)/(f'(\alpha))^2 = 0, \quad \phi''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)},$$

zaključujemo da proces ima red konvergencije barem dva, a pod uslovom $f''(\alpha) \neq 0$ proces ima red konvergencije tačno dva. □

Newtonov metod nije istorijski dobijen metodom proste iteracije, već je izveden iz Taylorovog razvoja funkcije. Naime, razvijemo li u Taylorov polinom funkciju f u tački x, koja je u okolini tačke α, dobijamo

$$f(\alpha) = f(x) + f'(x)(\alpha - x) + \omega(\alpha - x)(\alpha - x).$$

Iskoristimo li činjenicu da je $f(\alpha) = 0$ i zanemarimo li član uz funkciju ω, posle kraćeg rešavanja po α, nalazimo

$$\alpha \approx x - \frac{f(x)}{f'(x)}.$$

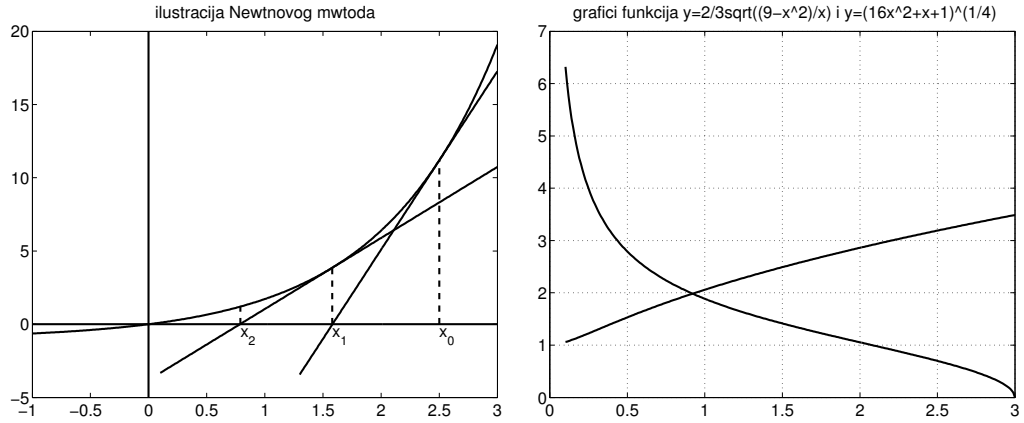
Kako na ovaj način dobijamo samo aproksimaciju rešenja jednačine možemo postupak ponoviti sa novom vrednošću x. Na taj način dobijamo Newtonov iterativni proces.

Newtonov iterativni proces ima i jednostavnu geometrijsku interpretaciju. Naime, ako posmatramo jednačinu

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \Rightarrow 0 = f(x_k) + f'(x_k)(x_{k+1} - x_k),$$

jednostavno možemo zaključiti da je njeno rešenje, tačka x_{k+1}, tačka u kojoj tangenta na grafik funkcije f, u tački $(x_k, f(x_k))$, seče x-osu. Na taj način Newtonov iterativni proces možemo

interpretirati kao niz konstrukcija tangenata na grafik funkcije i određivanja presečnih tačaka tih tangenata sa x-osom. U prvoj iteraciji postavljamo tangentu na grafik funkcije u tački $(x_0, f(x_0))$. Zatim određujemo tačku x_1 kao presek ove tangente sa x-osom. U sledećoj iteraciji postavljamo tangentu u tački $(x_1, f(x_1))$, pa tačku x_2 određujemo kao presek ove tangente i x-ose, i tako redom. Tačku x_{k+1} određujemo kao presek tangente postavljene u tački $(x_k, f(x_k))$ i x-ose. Slika 3.3, levo, prikazuje geometrijsku interpretaciju Newtonovog metoda.



Slika 3.3: Slika levo daje geometrijsku interpretaciju Newtonovog metoda, slika desno grafici funkcija $y = 2((9-x^2)/x)^{1/2}/3$ i $y = (16x^2 + x + 1)^{1/4}$.

Ilustraciju Newtonovog metoda možemo dati koristeći funkciju `prostaIteracija`. Koristićemo Newtonov metod za rešavanje jednačine $f(x) = x - \cos x = 0$. Možemo iskoristiti sledeću sekvencu naredbi.

```
>>fun=@(x) x-(x-cos(x))/(1+sin(x));
>>[x,err,iter]=prostaIteracija(fun,0,10^(-14),10^(-20),20)
x =
   0.739085133215161
err =
  Columns 1 through 3
   1.000000000000000   0.332686770857289   0.015222271275785
  Columns 4 through 6
   0.000037556601836   0.000000000230181                   0
iter =
     6
```

Kao što vidimo dolazi do znatnog ubrzavanja konvergencije u odnosu na metod proste iteracije. Ubrzanje konvergencije može se i očekivati, jer je Newtonov metod drugog reda. Iterativni proces konvergira u šest iteracija.

3.1.6 Newtonov metod za višestruke nule

Newtonov metod se može iskoristiti za rešavanje jednačina čije rešenje je istovremeno i nula prvog izvoda. Radi ilustracije pokušajmo Newtonovim metodom da rešimo jednačinu $f(x) = (x-1)^2 = 0$.

Očigledno je rešenje $\alpha = 1$. Sledeći niz komandi ilustruje ponašanje Newtonovog metoda u ovom slučaju.

```
>>fun=@(x) x-(x-1)^2/(2*(x-1));
>>[x,err,iter]=prostaIteracija(fun,1.1,10^(-14),10^(-20),100)
x =
   1.000000000000006
err =
   .
   .
   .
  Columns 37 through 39
   0.000000000000728   0.000000000000364   0.000000000000182
  Columns 40 through 42
   0.000000000000091   0.000000000000046   0.000000000000023
  Columns 43 through 44
   0.000000000000011   0.000000000000006
iter =
    44
```

Kao što vidimo proces konvergira, ali je konvergencija spora. Razlog je što Newtonov metod u ovom slučaju konvergira linearno. Ova se činjenica može dokazati.

Teorema 3.5 *Neka je* $\phi(x) = x - f(x)/f'(x)$ *i neka je* $f(\alpha) = f'(\alpha) = 0$. *Pod pretpostavkom da funkcija zadovoljava uslove teoreme* 3.2*, na intervalu* $[a, b]$*, iterativni proces* $x_{k+1} = \phi(x_k)$*,* $k \in \mathbb{N}_0$*,* $x_0 \in [a, b]$*, ima red konvergencije jedan, sa asimptotskom konstantom greške*

$$\phi'(\alpha) = 1 - \frac{1}{m},$$

gde je m *red nule* α *funkcije* f*.*

Dokaz. Kako funkcija ϕ zadovoljava uslove teoreme 3.2, dovoljno je samo utvrditi red najvišeg izvoda koji je jednak nuli u tački α. Međutim, kako funkcije f i f' imaju nulu u tački α, to možemo pretpostaviti da važi $f(x) = (x - \alpha)^m g(x)$, gde funkcija g nema vrednost nula u tački α. Koristeći ovu reprezentaciju nalazimo

$$\phi'(\alpha) = 1 - \frac{1}{m} \neq 0,$$

pod uslovom $m \neq 1$, što je ispunjeno jer nula nije reda jedan. Zaključujemo da je u ovom slučaju Newtonov metod prvog reda sa zadatom asimptotskom konstantom greške. □

Ako se zna red nule može se vratiti kvadratna konvergencija Newtonovog metoda. Na primer, neka funkcija f ima u tački α nulu reda m. Onda funkcija $g = f^{1/m}$ ima u tački α nulu prvog reda. Ako primenimo Newtonov metod na funkciju g, nalazimo metod

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} = x_k - \frac{(f(x_k))^{1/m}}{)1/m)f'(x_k)(f(x_k))^{1/m-1}} = x_k - m\frac{f(x_k)}{f'(x_k)}.$$

Primenimo li novodobijeni metod na rešavanje jednačine $f(x) = (\sin x - \sqrt{2}/2)^2 = 0$, nalazimo

```
>>fun=@(x) x-2*(sin(x)-sqrt(2)/2)^2/(2*(sin(x)-sqrt(2)/2)*cos(x));
>>[x,err,iter]=prostaIteracija(fun,.5,10^(-14),10^(-20),20)
x =
         0.785398163397448
err =
  Columns 1 through 3
     0.3416213490352        0.03265169806562      0.0004108385899208
  Columns 4 through 5
     6.626875282296e-008      1.83765381596e-015
iter =
     5
```

Kao što vidimo povratili smo brzinu konvergencije. Međutim, prilikom primene Newtonovog metoda za rešavanje ovakvih nelinearnih jednačina treba obratiti posebnu pažnju. Na primer, ako ovaj metod primenimo na rešavanje jednačine $f(x) = (x-1)^2 = 0$ u dve iteracije pronalazimo vrednost `NaN` kao rešenje. Ovo je posledica toga što proces već posle jedne iteracije konvergira u vrednost 1, pa u sledećoj iteraciji imamo izračunavanje $0/0$, što rezultira pojavom vrednosti `NaN`, koja se nezadrživo prostire kroz izračunavanja.

3.1.7 Divergencija Newtonovog metoda

Newtonov metod nije metod koji se može primeniti na sve moguće probleme rešavanja nelinearnih jednačina. Konkretno, ako primenimo Newtonov metod na rešavanje jednačine $f(x) = \mathrm{sgn}(x)\sqrt{|x|} = 0$ dobijamo ilustrativnu divergenciju. Iterativni proces izgleda ovako

$$x_{k+1} = x_k - \frac{\mathrm{sgn}(x_k)\sqrt{|x_k|}}{\frac{1}{2}\mathrm{sgn}(x_k)\frac{\mathrm{sgn}(x_k)}{\sqrt{|x_k|}}} = x_k - 2\mathrm{sgn}(x_k)|x_k| = x_k - 2x_k = -x_k.$$

Sledeća iteracija je prosto negativna vrednost prethodne. Ako počnemo od tačke $x_0 \neq 0$, niz izgleda ovako $x_k = (-1)^k x_0$, $k \in \mathbb{N}_0$. Kako vidimo konvergencije ne može biti. Ovo nije u suprotnosti sa teoremom 3.4, jer funkcija f nije dva puta neprekidno diferencijabilna, već je izvod neograničen u nuli funkcije f.

3.1.8 Funkcija fzero

Funkcija `fzero` predstavlja implementaciju, u okviru `Matlab`-a, čuvenog Dekkerovog algoritma za pronalaženje tačke na x-osi u kojoj neprekidna funkcija menja znak. Primetimo da je pojam menja znak bitan. Na primer, tačka u kojoj neprekidna funkcija menja znak jeste nula neprekidne funkcije. Međutim, postoje nule neprekidne funkcije u kojima ne dolazi do promene znaka. Funkcija $f(x) = x^2$ i tačka $\alpha = 0$ su očigledan primer. Funkcija `fzero` nije u stanju da nađe nulu pomenute funkcije. Ovo je značajno ograničenje koje treba imati na umu pri radu sa funkcijom `fzero`. Algoritam koji se koristi je kombinacija više algoritama sa različitim redovima konvergencije. Zavisno od lokalnog ponašanja funkcije čija se tačka promene znaka traži, funkcija `fzero` menja strategiju, što dovodi do promene brzine konvergencije tokom konstrukcije rešenja.

Postoji nekoliko formata sa kojima se funkcija može pozvati. Ilustrovaćemo neke od njih. Najelementarniji format je

```
x=fzero(fun,x0)
[x,fval,exitfalg]=fzero(fun,x0)
```

gde je `fun` funkcija čija se tačka promene znaka traži, tačka `x0` je početna aproksimacija rešenja, dok je izlazni argument `x` eventualno pronađeno rešenje. U drugom formatu, izlazni argument `fval` je vrednost funkcije `fun` u argumentu `x`, dok vrednost izlaznog argumenta `exitflag` opisuje stanje sa kojim je funkcija izašla. Tabela 3.1 opisuje moguće vrednosti izlaznog argumenta `exitflag`.

vrednost	opis
1	konvergencija postignuta
-1	izvršenje algoritma prekinuto od strane izlazne funkcije, izlazna funkcija je korisnički definisana funkcija čiji se pokazivač prenosi vrednošću ulaznog argumenta `options.OutpuFnc`, videti dalje u tekstu
-3	`NaN` ili `Inf` vrednost funkcije `fun` je detektovana tokom izvršenja algoritma
-4	kompleksna vrednost funkcije `fun` je detektovana tokom izvršenja algoritma
-5	konvergencija postignuta, ali je vrednost možda singularna tačka, a ne nula funkcije `fun`
-6	ne može se detektovati promena znaka funkcije `fun`

Tabela 3.1: Moguće vrednosti izlaznog parametra exitflag funkcije fzero sa opisima.

Sledeći primer ilustruje korišćenje ovog formata funkcije `fzero` za pronalaženje nule funkcije `sin` u tački 2π.

```
>>[x,fval,exitflag]=fzero(@sin,2*pi+.5)
x =
          6.28318530717959
fval =
    -2.44929359829471e-016
exitflag =
     1
>>abs(x-2*pi)/(2*pi)
ans =
         0
```

Kao što vidimo vrednost argumenta `exitflag` je postavljena na vrednost za regularan izlazak iz funkcije.

Moguće je da vrednost argumenta `x0` sadrži dve vrednosti. U ovom slučaju funkcija `fzero` očekuje da vrednosti funkcije `fun` budu različitog znaka u ovim tačkama. Ako ovaj uslov nije ispunjen funkcija izlazi i generiše poruku o grešci. Ova mogućnost postoji da bi korisnik mogao preneti funkciji `fzero` eventualno poznavanje tačaka u kojima su vrednosti funkcije `fun` različitog znaka, jer algoritam koji implementira funkcija `fzero` radi tako što za početak traži ovakve dve tačke. Na ovaj način se izvršenje funkcije `fzero` ubrzava.

Odredimo tačku $\pi/2$ promene znaka funkcije `tan`.

```
>>[x,fval,exitflag]=fzero(@tan,[pi/2-.5,pi/2+.5])
x =
           1.5707963267949
```

```
fval =
    -1.20926868691496e+015
exitflag =
    -5
```

Kao što vidimo tačka promene znaka je nađena. Međutim, ispravno, vrednost izlaznog argumenta `exitflag` iznosi -5, jer je u pitanju singularna tačka a ne nula funkcije `tan`.

Ilustrujmo šta se dešava ako funkciju primenimo na pronalaženje nule funkcije koja u svojoj nuli ne menja znak. Na primer, pokušajmo da nađemo nulu funkcije $f(x) = x^2$.

```
>>f=@(x) x^2;
>>[x,fval,exitflag]=fzero(f,.1)
Exiting fzero: aborting search for an interval containing a sign
  change because NaN or Inf function value encountered during search.
(Function value at -1.37296e+154 is Inf.)
Check function or try again with a different starting value.
x =
   NaN
fval =
   NaN
exitflag =
    -3
```

Iako je startna vrednost dovoljno dobra i funkcija ima nulu u blizini startne vrednosti, funkcija `fzero` nije u stanju da je detektuje, jer u ovoj nuli ne dolazi do promene znaka funkcije.

Postoje još dva formata za pozivanje funkcije `fzero`

```
[x,fval,exitflag]=fzero(fun,x0,options)
[x,fval,exitflag,output]=fzero(fun,x0,options)
```

Izlazni argument `output` predstavlja strukturu podataka koja sadrži polja koja su data u tabeli 3.2. Kao što vidimo ova struktura služi za bolje opisivanje algoritma kojim je pronađena tačka u kojoj funkcija menja znak. Ulazni argument `options` predstavlja strukturu podataka koja se koristi za upravljanje tokom izvršenja funkcije `fzero`. Polja unutar strukture `options` su data u tabeli 3.3.

polje	opis
`algorithm`	opisuje upotrebljeni algoritam, moguće vrednosti su `'bisection'` i `'interpolation'`
`funcCount`	broj izračunavanja funkcije upotrebljen prilikom izvršenja algoritma
`intervaliterations`	broj iteracija upotrebljen za pronalaženje intervala na kome funkcija `fun` menja znak
`iterations`	broj iteracija upotrebljen za nalaženje nule
`message`	poruka generisana prilikom završetka izvršenja funkcije

Tabela 3.2: Polja unutar strukture output izlaznog argumenta funkcije fzero.

Na primer, prilikom određivanja tačke promene znaka funkcije `tan` u tački $\pi/2$, možemo koristiti sledeći poziv

polje	opis
`Display`	vrednost `'off'` ne dozvoljava nikakvo štampanje poruka tokom izvršenja funkcije, vrednost `'iter'` štampa poruke po završetku svake iteracije, vrednost `'final'` štampa samo poruke prilikom poslednje iteracije, vrednost `'notify'`, što je podrazumevana vrednost, štampa izlaz jedino u slučaju ako nema konvergencije
`FunValCheck`	proverava da li su vrednosti objektne funkcije validne, vrednost `'on'` štampa grešku ako objektna funkcija vraća `NaN`, `Inf` ili kompleksni tip, vrednost `'off'`, koja je podrazumevana, ne štampa nikakve poruke u ovim slučajevima
`OutputFcn`	korisnički definisana funkcija koja se poziva prilikom završetka svake iteracije, naziva se i izlazna funkcija
`PlotFcns`	crta različite prikaze napretka procesa konvergencije dok se izvršava algoritam, mogu se koristiti predefinisane vrednosti, na primer `@optimplotx` koja crta vrednost aproksimacije tačke u kojoj funkcija menja znak, `@optimplotfval` crta vrednost funkcije u tačkama koje aproksimiraju vrednost tačke u kojoj funkcija menja znak, moguće je predati i neku korisnički definisanu funkciju
`TolX`	vrednost gornje granice relativne greške rešenja za prekid izvršenja funkcije

Tabela 3.3: Polja strukture options funkcije fzero.

```
>>options.PlotFcns=@optimplotx;
>>options.TolX=10^(-14);
>>[x,fval,exitflag,output]=fzero(@tan,[pi/2-.5,pi/2+.5],options)
x =
           1.5707963267949
fval =
    -1.20926868691496e+015
exitflag =
    -5
output =
    intervaliterations: 0
            iterations: 51
             funcCount: 53
             algorithm: 'bisection, interpolation'
               message: [1x215 char]
>> output.message
ans =
Current point x may be near a singular point. The interval
[1.0708, 2.0708] reduced to the requested tolerance and the
function changes sign in the interval,
```

```
but f(x) increased in magnitude as the interval reduced.
```

Kao što vidimo dobili smo rešenje u pedesetjednoj iteraciji i izračunavali smo funkciju pedesettri puta. Nismo koristili nijednu intervalnu iteraciju, jer smo zadali u startu dve tačke u kojima funkcija ima suprotne znake. Primenjen je algoritam `'bisection'` i `'interpolation'`. Po završetku algoritma dobili smo poruku čija se sadržina uglavnom odnosi na činjenicu da tačka u kojoj funkcija menja znak nije nula funkcije, već je u pitanju singularna tačka. Kako smo postavili i vrednost polja `PlotFcns` prilikom pronalaženja rešenja, možemo pratiti proces grafički.

Ovim smo samo ilustrovali mogućnosti funkcije `fzero` koja u `Matlabu` implementira trenutno najmoćniji algoritam za rešavanje nelinearnih jednačina jedne realne promenljive. Kao što se vidi iz priloženog algoritam nije svemoćan. Dakle, potrebna je prilična količina znanja da bi se sa lakoćom moglo upravljati ovim alatom.

3.2 Sistemi nelinearnih jednačina

Rešavanje sistema nelinearnih jednačina je neiscrpna tema. Velika broj naučnih radova je posvećen upravo ovoj temi. Problem je aktuelan i kako stvari stoje verovatno nikada neće biti rešen na zadovoljavajući način. Naime, uvek će postojati primene i sistemi nelinearnih jednačina koji će zahtevati nova istraživanja. Mi ćemo prikazati metod proste iteracije, metod Newton-Kantoroviča i gradijentni metod. Metod Newton-Kantoroviča je jedan od najstarijih metoda za rešavanje sistema nelinearnih jednačina. Ovaj metod je adaptacija Newtonovog metoda na sisteme nelinearnih jednačina. Ilustrovaćemo i osobine funkcije `roots` koja se koristi za faktorizaciju polinoma u `Matlabu`.

Još jedna bitna karakteristika sistema nelinearnih jednačina je nepostojanje metoda koji uvek dovodi do rešenja. U slučaju nelinearne jednačine jedne realne promenljive metod bisekcije uvek dovodi do rešenja, ako funkcija u rešenju menja znak. Kod sistema nelinearnih jednačina ne postoji metod koji uvek dovodi do rešenja. Zato je problem rešavanja sistema nelinearnih jednačina uvek aktuelan.

3.2.1 Metod Newton-Kantoroviča

U ovom odeljku rešavamo sistem jednačina

$$f_1(x_1, \ldots, x_n) = 0, \ldots, f_n(x_1, \ldots, x_n) = 0.$$

Vektor nezavisno promenljivih ćemo označavati sa $x = (x_1, \ldots, x_n)$, dok ćemo vektor funkcija označavati sa $f = (f_1, \ldots, f_n)$. Naš sistem jednačina u ovoj notaciji postaje $f(x) = 0$. Vidimo da je zapis potpuno isti i ne razlikuje se u zavisnosti od toga da li rešavamo jednu jednačinu ili sistem jednačina.

Koristeći se analogijom sa Newtonovim metodom, iterativni proces Newton-Kantoroviča možemo prikazati na sledeći način

$$x^{k+1} = x^k - W^{-1}(x^k) f(x^k), \quad k \in \mathbb{N}_0,$$

gde je W takozvana Jacobijeva matrica definisana sa

$$W(x) = \frac{Df(x)}{Dx} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}.$$

Veličina

$$d^{k+1} = -W^{-1}(x^k)f(x^k), \quad k \in \mathbb{N}_0, \tag{3.4}$$

se obično naziva korekcija metoda Newton-Kantoroviča.

U slučaju funkcije jedne promenljive Jacobijeva matrica se svodi na matricu reda jedan i to na $W(x) = [f'(x)]$. U tom slučaju se i metod Newton-Kantoroviča svodi na Newtonov metod, a inverzna matrica iznosi $[1/f'(x)]$.

Kod rešavanja sistema nelinearnih jednačina indeks koji broji iteracije pišemo gore, da bismo izbegli dvosmislenost sa indeksom koji označava promenljive, a koji pišemo dole. Tako x^2 znači vrednost u drugoj iteraciji vektora x, dok x_2 znači druga koordinata vektora x. Takođe, x_2^2 znači vrednost druge koordinate vektora x u drugoj iteraciji.

U slučaju Newtonovog metoda da bismo obezbedili konvergenciju metoda zahtevali smo da izvod funkcije f bude različit od nule u okolini rešenja. U slučaju metoda Newton-Kantoroviča zahtevamo regularnost Jacobijeve matrice u okolini rešenja. U suprotnom ne možemo garantovati konvergenciju metoda Newton-Kantoroviča.

Uslovi pod kojima metod Newton-Kantoroviča konvergira mogu se izraziti na različite načine. Jedan od ovih načina je sledeći.

Teorema 3.6 (Metod Newton-Kantoroviča) *Neka je funkcija f dva puta neprekidno diferencijabilna u oblasti $D = \{x \mid ||x - x^0||_{+\infty} \leq R\}$ i neka je*

$$s_{ij} = \sum_{k=1}^{n} \left| \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k} \right| \leq N, \quad i, j = 1, \ldots, n, \quad ||f(x^0)||_{+\infty} \leq Q,$$

$$||W^{-1}(x^0)||_{+\infty} \leq b, \quad \Delta_0 = \det W(x^0) \neq 0, \quad h = nNQb^2 \leq \frac{1}{2}.$$

Ako je

$$R \geq r = \frac{1 - \sqrt{1 - 2h}}{h} Qh,$$

iterativni proces Newton-Kantoroviča konvergira ka rešenju $\alpha \in \{x \mid ||x - x^0||_{+\infty} \leq r\}$.

Red konvergencije iterativnog procesa je najmanje dva.

U ovom slučaju red konvergencije dva iterativnog procesa znači da postoji granična vrednost

$$\lim_{k \to +\infty} \frac{||x^{k+1} - \alpha||}{||x^k - \alpha||^2} = C,$$

koja je za generički iterativni proces različita od nule. Za specijalne procese ova vrednost može biti jednaka nul. U tom slučaju iterativni proces može imati i red konvergencije veći od dva.

Teorema koju smo naveli daje dovoljne uslove konvergencije u $||\cdot||_{+\infty}$ normi. Postoje slične teoreme koje se mogu dati za ostale norme vektora. Naš cilj nije da nabrojimo sve ove teoreme. Ova teorema je ovde više data kao ilustracija. Otprilike sve teoreme koje se mogu formulisati upotrebom ostalih normi imaju sličan 'ukus'.

Metod Newton-Kantoroviča možemo implementirati na sledeći način.

```
function [x0,err,iter]=newtonKantorovichR(fun,jac,x0,prec,acc,maxIter)
%NEWTONKANTOROVICHR(FUN,JAC,X0,PREC,MAXITER) resava sistem
% nelinearnih jednacina FUN(X)=0 koristeci metod
% Newton-Kantorovica, drugi argument je Jacobijan sistema
% jednacina, treci je pocetna vrednost, cetvrti je zeljena
% relatvina greska u resenju, peti je maksimalna apsolutna
% greska u resenju, sesti je maksimalni dozvoljeni
% broj iteracija, funkcija ima tri izlazna argumenta,
% prvi je vrednost poslednje iteracije, drugi je relativna
% greska u svim iteracijama, treci je broj iteracija
% za koje resenje dostignuto

iter = 0; err=[]; deltaX = Inf;
while norm(deltaX) >= max(prec*norm(x0),acc)
    deltaX=linsolve(jac(x0),fun(x0));
    x0=x0-deltaX;
    iter = iter+1;
    err = [err norm(deltaX)/norm(x0)];
    if iter>maxIter
        disp('maksimalni dozvoljeni broj itercija dostignut');
        break;
    end
end
```

Da bismo ilustrovali metod Newton-Kantoroviča primenimo je na rešavanje sistema nelinearnih jednačina

$$f(x) = \begin{bmatrix} 9x_1^2x_2 + 4x_2^2 - 36 \\ 16x_2^2 - x_1^4 + x_2 + 1 \end{bmatrix} = 0. \tag{3.5}$$

Rešavanjem obe jednačine po x_1, dobijamo

$$x_1 = \frac{2}{3}\sqrt{\frac{9 - x_2^2}{x_2}}, \quad x_1 = (16x_2^2 + x_2 + 1)^{1/4}.$$

Ako nacrtamo ove dve funkcije, slika 3.3, desno, možemo pronaći da se krive seku u okolini tačke $(x_1, x_2) = (2, 1)$. Ovu vrednost ćemo iskoristiti kao početnu vrednost za iterativni proces Newton-Kantoroviča. Jacobijevu matricu sistema jednačina izračunavamo jednostavno

$$W(x) = \begin{bmatrix} \dfrac{\partial f_1(x)}{\partial x_1} & \dfrac{\partial f_1(x)}{\partial x_2} \\ \dfrac{\partial f_2(x)}{\partial x_1} & \dfrac{\partial f_2(x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 18x_1x_2 & 9x_1^2 + 8x_2 \\ -4x_1^3 & 32x_2 + 1 \end{bmatrix}.$$

Sledeća sekvenca naredbi ilustruje upotrebu funkcije `newtonKantorovichR`.

```
>>fun=@(x) [9*x(1)^2*x(2)+4*x(2)^2-36; 16*x(2)^2-x(1)^4+x(2)+1];
>>jac=@(x) [ 18*x(1)*x(2) 9*x(1)^2+8*x(2); -4*x(1)^3 32*x(2)+1];
>>format long
>>[x,err,iter]=newtonKantorovichR(fun,jac,[2; 1],10^(-14),10^(-20),20)
x =
   1.983708733953144
   0.920742637018965
err =
  Columns 1 through 3
   0.036064313190568   0.001056412830540   0.000000788637842
  Columns 4 through 5
   0.000000000000598   0.000000000000000
iter =
     5
```

Vidimo da proces konvergira brzo. Prinos značajnih cifara po iteraciji je veliki, jer je iterativni proces dugog reda.

Metod Newton-Kantoroviča se u literaturi, posebno u engleskom govornom području, najčešće sreće pod imenom Newton-Raphsonov metod.

3.2.2 Faktorizacija polinoma

Faktorizacija polinoma, ili ekvivalentno pronalaženje svih nula polinoma, je vrlo star problem. Međutim, interesantno je da problem još uvek nije zadovoljavajuće rešen. Problemi nastaju pre svega u činjenici da nule polinoma mogu biti višestruke, zatim, grupisane u malim diskovima kompleksne ravni i slično. Ovi problemi čine problem pronalaženja nula polinoma i danas aktuelnim.

Funkcija `roots`, koja u `Matlab`u služi za određivanje nula polinoma, pretvara problem određivanja nula polinoma u problem određivanja sopstvenih vrednosti takozvane Frobeniusove matrice. Može se pokazati da su sopstvene vrednosti Frobeniusove matrice identične skupu nula polinoma za koji je odgovarajuća Frobenuisova matrica konstruisana. Metodi za određivanje sopstvenih vrednosti matrice su jako napredovali i predstavljaju jedan od glavnih pravaca istraživanja u oblasti linearne algebre, tako da funkcija `roots` u većini slučajeva daje dobre rezultate. Jedan interesantan slučaj u kojem funkcija greši je sledeći. Pretpostavimo da treba odrediti nule polinoma $p(x) = (x-1)^6$. Vidimo da polinom p ima jednu nulu u tački 1 višestrukosti šest. Sledeći niz komandi `Matlab`a ilustruje mogućnosti funkcije `roots`.

```
>>r=roots(poly(ones(1,6)))
r =
  1.004799966335366
  1.002396863444901 + 0.004156907489679i
  1.002396863444901 - 0.004156907489679i
  0.997599991527655 + 0.004151459950429i
  0.997599991527655 - 0.004151459950429i
  0.995206323719518
>>norm(r-ones(6,1))/norm(ones(1,6))
ans =
        0.0047968392089633
>>poly(r)
```

```
ans =
  Columns 1 through 3
  1                        -6                        15
  Columns 4 through 6
  -20                      15                        -6
  Column 7
  0.999999999999998
>>poly(ones(1,6))
ans =
     1    -6    15   -20    15    -6     1
```

Kao što vidimo funkcija `roots` u stanju je da rekonstruiše nule polinoma sa dve značajne cifre. Rezultati će imati još veću grešku ako funkciju primenimo recimo na polinom $p(x) = (x-1)^{20}$. Međutim, kao što vidimo, kad pokušamo da rekonstruišemo koeficijente polinoma koristeći nule polinoma i u jednom i u drugom slučaju dobijamo isti polinom. Razlog je prosto slaba uslovljenost algoritma koji preslikava koeficijente polinoma u nule polinoma i dobra uslovljenost algoritma koji preslikava nule polinoma u koeficijente za ovaj skup podataka.

Da problem ne mora biti vezan za višestrukost nula pokazuje i sledeći Wilkinsonov primer. Posmatrajmo polinom $p(x) = \prod_{i=1}^{20}(x-i)$. Pokušajmo da primenimo funkciju `roots` na određivanje nula ovog polinoma. Dobijamo

```
>>r=roots(poly(1:20))
r =
  20.000324878110792
  18.997159988498897
  18.011221691503330
  16.971132188215869
  16.048274637499368
  14.935355597149178
  14.065272906061795
  12.949055582469070
  12.033449209209296
  10.984041246175890
  10.006059694509711
   8.998394489161083
   8.000284344046330
   6.999973480924894
   5.999999755878211
   5.000000341909170
   3.999999967630577
   3.000000001049188
   1.999999999997379
   0.999999999999841
>>norm(r-(20:-1:1)')/norm(1:20)
ans =
       0.00234227799831508
>>p=poly(r);
>>norm(poly(r)-poly(1:20))/norm(poly(1:20))
```

```
ans =
    3.21233544363623e-015
```

Kao što vidimo relativna greška je reda veličine 10^{-3}. Rezultati postaju još gori ako funkciju primenimo na određivanje nula polinoma $p(x) = \prod_{i=1}^{50}(x-i)$. I u ovom slučaju, kao u prethodnom, koeficijenti polinoma su dobijeni upotrebom oba skupa nula isti.

Korišćenjem nekih naprednijih algoritama mogu se dobiti preciznije vrednosti nula polinoma. Da bismo ilustrovali ovu činjenicu potreban nam je metod Newton-Kantoroviča primenjen na Vietove formule. Vietove formule povezuju nule polinoma i koeficijente polinoma. Ako važi

$$p(x) = \sum_{i=0}^{n} p_i x^i = p_n \prod_{i=1}^{n}(x - x_i),$$

onda je

$$\frac{p_{n-i}}{p_n} = (-1)^i \sum_{\{j_1,\ldots,j_i\}} x_{j_1} \cdots x_{j_i}, \quad i = 1, \ldots, n,$$

gde se sumiranje vrši po svim mogućim kombinacijama skupa $\{1, \ldots, n\}$ od i elemenata.

Može se pokazati, primenom naprednih metoda izračunavanja, da metod Newton-Kantoroviča primenjen na Vietove formule dobija sledeći oblik

$$x_i^{k+1} = x_i^k - \frac{p(x_i^k)}{q_{i,k}(x_i^k)}, \quad i = 1, \ldots, n, \quad k \in \mathbb{N}_0,$$

gde je p polinom čije nule određujemo, a polinom $q_{i,k}$ je određen sa

$$q_{i,k}(x) = \prod_{j=1,\ j\neq i}^{n} (x - x_j^k).$$

Ovaj algoritam se može primeniti ako imamo dovoljno dobre startne vrednosti $x^0 = (x_1^0, \ldots, x_n^0)$, i ako su nule polinoma p proste. Nije teško napisati program koji implementira opisani algoritam. Ideja je primena ovog algoritma na određivanje nula polinoma

$$p(x) = \prod_{i=1}^{20}(x - i),$$

kojim smo se bavili u prethodnom primeru. Ako iskoristimo kao startne vrednosti vrednosti koje vraća funkcija `roots(poly(1:20))`, dobićemo nule sa relativnom greškom reda veličine $3.4 \cdot 10^{-4}$, što je za red veličine bolje od relativne greške koju dobijamo upotrebom funkcije `roots`.

Naravno, ovo poređenje nije u potpunosti korektno, jer je funkcija `roots` funkcija opšte namene, dok je naš algoritam više pisan za specijalne namene. Na primer, prezentovani algoritam radi samo sa prostim nulama i moramo imati dovoljno dobre startne vrednosti.

3.2.3 Gradijentni metod

Gradijentni metod za rešavanje sistema nelinearnih jednačina polazi od ekvivalentnog problema. Naime, umesto da rešavamo sistem jednačina

$$f_1(x_1, \ldots, x_n) = 0, \ldots, f_n(x_1, \ldots, x_n) = 0,$$

rešavamo jednačinu

$$U(x) = \sum_{i=1}^{n} f_i^2(x) = (f(x), f(x)) = 0.$$

Za rešavanje ove jednačine formirajmo iterativni proces

$$x^{k+1} = x^k - \lambda_k \nabla U(x^k), \quad k \in \mathbb{N}_0,$$

gde je

$$\nabla U(x) = \text{grad} U(x) = \left(\frac{\partial U(x)}{\partial x_1}, \ldots, \frac{\partial U(x)}{\partial x_n} \right).$$

Parametar λ_k određujemo iz uslova da funkcija $S(t) = U(x^k - t\nabla U(x^k))$ u tački $t = \lambda_k$ ima minimum. Intuicija iza ovakvog određivanja korekcije se zasniva na sledećem zapažanju. Gradijent funkcije je smer u kome funkcija ima najveću promenu. Recimo, posmatrajmo funkciju $f(x_1, x_2) = x_1^2 + x_2^2$. Grafik ove funkcije je paraboloid sa centrom u koordinatnom početku. Često kažemo i da liči na gornji deo elegantne čaše za vino. Ako odredimo gradijent ove funkcije nalazimo $\nabla f = (2x_1, 2x_2)$. Vidimo da je gradijent usmeren u smeru najbržeg rasta funkcije. Ideja gradijentnog metoda je da traži minimum funkcije u smeru suprotnom od gradijenta ako je gradijent smer rasta i smeru gradijenta ako je gradijent smer opadanja funkcije.

Kako je problem nalaženja minimuma funkcije S teško rešiti egzaktno koristimo aproksimaciju. Naime, podrazumevamo da je λ_k dovoljno malo da vrednost funkcija f_i u tački $x^k - t\nabla U(x^k)$, možemo predstaviti sledećim izrazom

$$f_i(x^k - t\nabla U(x^k)) \approx f_i(x^k) - t(\nabla f_i(x^k), \nabla U(x^k)), \quad i = 1, \ldots, n.$$

Diferenciranjem funkcije S, uz učinjene pretpostavke, nalazimo

$$\begin{aligned} \frac{dS(t)}{dt} &= \frac{d}{dt} \sum_{i=1}^{n} (f_i(x^k) - t(\nabla f_i(x^k), \nabla U(x^k)))^2 \\ &= -2 \sum_{i=1}^{n} (f_i(x^k) - t(\nabla f_i(x^k), \nabla U(x^k))(\nabla f_i(x^k), \nabla U(x^k)). \end{aligned}$$

Ako izjednačimo gornji izraz sa nulom i rešimo jednačinu po t, nalazimo

$$\lambda_k = \frac{\sum_{i=1}^{n} f_i(x^k)(\nabla f_i(x^k), \nabla U(x^k))}{\sum_{i=1}^{n} (\nabla f_i(x^k), \nabla U(x^k))^2}.$$

Ovaj izraz se može dodatno pojednostaviti. Naime, važi

$$\nabla U(x) = \nabla \sum_{i=1}^{n} f_i^2(x) = 2 \sum_{i=1}^{n} f_i \nabla f_i(x) = 2W^T(x) f(x),$$

gde je W Jacobijeva matrica. Koristeći ovu jednakost možemo izraziti

$$\lambda_k = \frac{1}{2} \frac{(W_k^T f^k, W_k^T f^k)}{(W_k W_k^T f^k, W_k W_k^T f^k)},$$

gde smo radi kratkoće zapisa uveli $W_k = W(x^k)$ i $f^k = f(x^k)$. Odavde dobijamo

$$x^{k+1} = x^k - \frac{(W_k^T f^k, W_k^T f^k)}{(W_k W_k^T f^k, W_k W_k^T f^k)} W_k^T f^k, \quad k \in \mathbb{N}_0.$$

Primetimo da gradijentni metod za razliku od metoda Newton-Kntoroviča ne zahteva izračunavanje inverzne Jacobijeve matrice. Ovo predstavlja bitnu razliku, jer se gradijentni metod može primeniti i u slučajevima kad je Jacobijeva matrica singularna u rešenju sistema nelinearnih jednačina. Veličina

$$d^{k+1} = -2\lambda_k W_k^T f^k, \quad k \in \mathbb{N}_0, \tag{3.6}$$

naziva se korekcija gradijentnog metoda.

Implementacija gradijentnog metoda može se dati na sledeći način.

```
function [x0,err,iter]=gradientR(fun,jac,x0,prec,acc,maxIter)
%GRADIENTR(FUN,JAC,X0,PREC,ACC,MAXITER) resava sistem
% nelinearnih jednacina FUN(X)=0 gradijentnim metodom,
% prvi argument je funkcija, drugi je Jacobijeva matrica,
% treci je startna vrednost metoda,
% cetvrti je dozvoljena relativna greska u resenju,
% peti je dozvoljena apsolutna greska u resenju
% sesti je maksimalni dozvoljeni broj iteracija
% funkcija ima tri izlazna argumenta, prvi je vrednost
% poslednje iteracije resenja, drugi je relativna greska
% u svim iteracijama, treci je broj iteracija u kojem
% je dostignuto resenje

iter = 0; err = []; deltaX = Inf;
while norm(deltaX) >= max(prec*norm(x0), acc)
    j = jac(x0); f = fun(x0); jT = j';
    jjT = j*jT; jTf = jT*f; jjTf = jjT*f;
    lambda = .5*dot(jTf,jTf)/dot(jjTf,jjTf);
    deltaX = 2*lambda*jTf;
    x0 = x0-deltaX;
    err = [err norm(deltaX)/norm(x0)];
    iter = iter+1;
    if iter > maxIter
        disp('maksimalni broj iteracija dostignut');
        break;
    end
end
```

Primenimo gradijentni metod na rešavanje sistema jednačina (3.5).

```
>>fun=@(x) [9*x(1)^2*x(2)+4*x(2)^2-36; 16*x(2)^2-x(1)^4+x(2)+1];
>>jac=@(x) [ 18*x(1)*x(2) 9*x(1)^2+8*x(2); -4*x(1)^3 32*x(2)+1];
>> [x,err,iter]=gradientR(fun,jac,[2; 1],10^(-14),10^(-20),100)
x =
   1.983708733953143
```

```
   0.920742637018966
err =
  Columns 1 through 3
   0.035722543964026   0.004402908749225   0.000337282151915
  Columns 4 through 6
   0.000034967264896   0.000002351151598   0.000000249158165
  Columns 7 through 9
   0.000000016731596   0.000000001773355   0.000000000119084
  Columns 10 through 12
   0.000000000012622   0.000000000000847   0.000000000000090
  Column 13
   0.000000000000006
iter =
    13
```

Primetimo da je ovde broj potrebnih iteracija znatno veći u odnosu na broj iteracija kod metoda Newton-Kantoroviča, što je u skladu sa činjenicom da je gradijentni metod reda jedan.

Nećemo se baviti detaljnije uslovima pod kojima gradijentni metod konvergira. Zainteresovani čitaoci mogu naći ove uslove u literaturi na kraju knjige.

3.2.4 Funkcija fsolve

Funkcija `fsolve` koristi tri algoritma za rešavanje sistema nelinearnih jednačina. To su:

`trust-region-dogleg, trust-region-reflective, levenberg-marquardt.`

Međutim, nije moguće sve algoritme primeniti na rešavanje svih sistema nelinearnih jednačina. Na primer, algoritam `trust-region-reflective` se može primeniti jedino u slučaju kad je broj jednačina barem jednak broju promenljivih, mada može biti veći. Algoritam `trust-region-dogleg` može biti primenjen jedino na sisteme jednačina kod kojih je broj jednačina tačno jednak broju promenljivih. U generalnom slučaju se može primeniti `levenberg-marquardt` algoritam. Ako navedete kao opciju algoritam koji je nemoguće primeniti na rešavanje konkretnog sistema nelinearnih jednačina, funkcija `fsolve` bira algoritam koji je moguće primeniti i rešava sistem nelinearnih jednačina.

Algoritam `trust-region-dogleg` se zasniva na kombinaciji metoda Newton-Kantoroviča i gradijentnog metoda. Naime, vrednost korekcije d^{k+1} koja je potrebna za izračunavanje $x^{k+1} = x^k + d^{k+1}$, se dobija kao linearna kombinacija korekcije metoda Newton-Kantoroviča i korekcije gradijentnog metoda. Ovakvo izračunavanje korekcije je moguće jedino u slučaju kad je metod Newton-Kantoroviča moguće primeniti, dakle kad je Jacobijeva matrica sistema nelinearnih jednačina regularna. Ako je Jacobijeva matrica sistema singularna koristi se samo korekcija dobijena na osnovu gradijentnog metoda. Označimo korekciju metoda Newton-Kantoroviča sa d_{NK}^{k+1} i sa d_G^{k+1} korekciju gradijentnog metoda. Vrednost korekcije algoritma `trust-region-dogleg` se bira kao vrednost $d^{k+1} = d_{NK}^{k+1} + \lambda(d_G^{k+1} - d_{NK}^{k+1})$ u kojoj funkcija

$$S(\lambda) = (f^k + W_k d^{k+1}, f^k + W_k d^{k+1}) = (f^k, f^k) + 2(d^{k+1})^T W_k^T f^k + (d^{k+1})^T W_k^T W_k d^{k+1}, \quad (3.7)$$

dostiže minimum. Vrednosti korekcija u metodu Newton-Kantoroviča i gradijentnom metodu su date pomoću (3.4) i (3.6), tako da je potrebno samo odrediti parametar λ. Međutim, kao i u gradijentnom metodu, funkcija koja se minimizira je samo aproksimacija pravog aproksimacionog

problema koja je izvedena pod pretpostavkama da je korekcija d^{k+1} mala. Zato se dodatno zahteva da korekcija zadovoljava uslov $||D\,d^{k+1}|| \leq \Delta$, gde je D unapred zadata dijagonalna matrica, a Δ je parametar koji određuje oblast u kojoj važi učinjena aproksimacija (engleski trust region).

Algoritam se implementira na sledeći način:

1. odredimo d_{NK}^{k+1} i d_G^{k+1}, ako je d_{NK}^{k+1} nemoguće odrediti biramo $d_{NK}^{k+1} = 0$,
2. zatim odredimo korekciju d^{k+1} koja minimizira funkciju (3.7), ali tako da je ispunjen uslov $||D\,d^{k+1}|| \leq \Delta$,
3. proverimo uslov $f(x^k + d^{k+1}) < f(x^k)$,
4. ako je uslov ispunjen uzimamo $x^{k+1} = x^k + d^{k+1}$,
5. ako uslov nije ispunjen smanjimo Δ i počinjemo izračunavanje ispočetka.

Naravno, postoji čitava strategija kako se smanjuje Δ i pod kojim uslovima treba prekinuti proces smanjivanja Δ, dakle, pod kojim uslovima treba odustati od dalje primene algoritma.

Levenberg-Marquardt algoritam koristi drugi metod za određivanje korekcije d^{k+1}. Ovde se korekcija izračunava koristeći jednačinu

$$(W_k^T W_k + \lambda_k) d^{k+1} = -W_k^T f^k, \tag{3.8}$$

ili jednačinu

$$(W_k^T W_k + \lambda_k \text{diag} W_k^T W_k) d^{k+1} = -W_k^T f^k. \tag{3.9}$$

Ako izaberemo $\lambda_k = 0$ dobijamo korekciju koja je jednaka korekciji metoda Newton-Kantoroviča. Sa povećanjem koeficijenta λ_k dobijamo korekcije na koje sve više ima uticaja korekcija gradijentnog metoda. U graničnom slučaju kad λ_k neograničeno raste dobijamo korekciju kao kod gradijentnog metoda. Funkcija `fsolve` bira između jednačina (3.8) i (3.9) na osnovu opcije `'ScaleProblem'` čija je vrednost `'none'` za jednačinu (3.8) i `'Jacobian'` za jednačinu (3.9).

Algoritam `trust-region-reflective` se koristi uglavnom za probleme koji imaju veliki broj jednačina i promenljivih. Ovaj algoritam dalje nećemo proučavati.

Funkcija `fsolve` ima sledeći format

```
[x,fval,exitflag,output,jacobian]=fsolve(fun,x0,options)
```

Ulazni parametri `fun` i `x0` su isti kao kod funkcije `fsolve`, predstavljaju sistem jednačina koji treba rešiti i početnu vrednost rešenja. Izlazni parametar `x` predstavlja rešenje sistema nelinearnih jednačina, parametar `fval` predstavlja vrednost funkcije `fun` u izračunatom rešenju `x` sistema nelinearnih jednačina. Vrednost parametra `exitflag` opisuje način pod kojim je prekinuto izvršenje funkcije `fsolve`, sve moguće vrednosti su date u tabeli 3.4. Parametar `output` predstavlja strukturu podataka koja daje informacije o načinu na koji je izvršen algoritam. Sva polja strukture i njihov opis su dati u tabeli 3.5. Parametar `jacobian` predstavlja vrednost Jacobijeve matrice sistema u postignutom rešenju sistema nelinearnih jednačina. Ulazni parametar `options` je struktura podataka koja omogućava upravljanje izvršenjem algoritma. Najznačajnija polja strukture `options` zajedno sa opisom su data u tabelama 3.6 i 3.7.

Neke ilustracije upotrebe funkcije `fsolve` mogu biti sledeće

```
>>fun=@(x) [x'*x-1; ones(1,length(x))*x];
>>[x,fval,exitflag,output]=fsolve(fun,[1;-1])
Equation solved.
```

vrednost	opis
1	rešenje pronađeno
2	pronađena korekcija je manja od unapred zadate vrednosti
3	promena u ostatku je manja od unapred zadate vrednosti ovde se pod ostatkom podrazumeva vrednost izraza $\sum f_i^2$ u poslednjoj iteraciji
4	veličina promene u smeru pretrage je manja od unapred zadate vrednosti
0	broj iteracija je prevazišao vrednost `options.MaxIter` ili je broj izračunavanja funkcije prevazišao `options.MaxFunEvals`
−1	izlazna funkcija je prekinula izvršenje
−2	algoritam konvergira ka tački koja nije rešenje
−3	vrednost parametra Δ je postala previše mala
−4	pretraga po pravcu ne može dovoljno da smanji ostatak

Tabela 3.4: Vrednosti izlaznog parametra exitflag i njihovo značenje.

polje	opis
`iterations`	upotrebljeni broj iteracija
`funcCount`	upotrebljeni broj izračunavanja funkcije
`algorithm`	upotrebljeni algoritam
`cgiterations`	ukupan broj PCG iteracija, koristi se samo algoritmom `trust-region-reflective`
`stepsize`	poslednja korekcija, samo u slučaju Levenberg-Marquardt algoritma
`firstorderopt`	meri optimalnost prvog reda $\|\|W_k^T f^k\|\|_{+\infty}$, samo sa algoritmima `trust-region-dogleg` i `trust-region-reflective`
`message`	tekstualna poruka koja bliže opisuje izlaz

Tabela 3.5: Polja strukture output i njihov opis.

```
fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
   0.707106781187354
  -0.707106781187354
fval =
   1.0e-11 *
   0.228217444941947
                   0
```

polje	opis
`'Algorithm'`	moguće vrednosti su: `'trust-region-dogleg'`, `'trust-region-reflective'` i `'levenberg-marquardt'`
`'DerivativeCheck'`	podrazumevana vrednost je `'off'`, ako je vrednost `'on'` porede se vrednosti izvoda dobijene aproksimacijama konačnim razlikama sa vrednostima drugog izlaznog argumenta funkcije `fun`
`'Diagnostics'`	podrazumevana vrednost je `'off'`, ako je vrednost `'on'` štampa se poruka o načinu kojim je funkcija pozvana
`'DiffMaxChange'`	maksimalna dozvoljena korekcija u slučaju da koristimo Jacobijevu matricu izračunatu konačnim razlikama, podrazumevana vrednost je `inf`
`'DiffMinChange'`	minimalna dozvoljena korekcija u slučaju da koristimo Jacobijevu matricu izračunatu konačnim razlikama, podrazumevana vrednost je `0`
`'Display'`	način na koji se koristi štampanje na izlaznom uređaju vrednost `'off'` nema štampanja, podrazumevana vrednost vrednost `'iter'` štampa informacije o procesu u svakoj iteraciji vrednost `'notify'` štampa informacije o procesu ako nema konvergencije vrednost `'final'` štampa samo podatke o poslednjoj iteraciji
`'FinDiffRelStep'`	kad je za ovu opciju zadat vektor `v` kao vrednost korak prilikom izračunavanja izvoda konačnim razlikama je određen sa `h=v.*sign(x).*max(abs(x),TypicalX);` za jednostrane formule i `h=v.*max(abs(x),TypicalX);` za dvostrane formule, ako je zadata skalarna vrednost ona se tretira kao vektor, podrazumeva vrednost je `sqrt(eps)` za jednostrano diferenciranje i `eps^(1/3)` za dvostrano diferenciranje
`'FinDiffType'`	može se izabrati `'forward'` ili `'central'` koje odgovaraju formulama za jednostrano ili dvostrano diferenciranje redom, ako je zadat problem sa ograničenjima vrednost `'forward'` znači i upotrebu formule za jednostrano diferenciranje unazad
`'FunValCheck'`	podrazumevana vrednost je `'off'`, ako je postavljena na `'on'` korisnik biva obavešten u slučaju da je izračunata vrednost funkcije kompleksan broj, `Inf`, `-Inf` ili `NaN`
`'Jacobian'`	podrazumevana vrednost je `'off'`, ako je vrednost `'on'` funkcija `fun` mora da vrati kao drugi izlazni argument vrednost Jacobijeve matrice
`'MaxFunEvals'`	gornja granica dozvoljenih izračunavanja funkcije `fun` kada dođe do prekoračenja funkcija prekida dalje izvršenje podrazumevana vrednost je sto puta veća od broja promenljivih
`'MaxIter'`	gornja granica dozvoljenog broja iteracija, ako dođe do prekoračenja funkcija prekida dalje izvršenje, podrazumevana vrednost je 200
`'OutputFun'`	korisnički definisane funkcije koje se pozivaju na kraju svake iteracije, poznate i pod nazivom izlazne funkcije

Tabela 3.6: Polja strukture options i njihova značenja, prvi deo.

```
exitflag =
     1
output =
       iterations: 4
```

polje	opis
`'PlotFncs'`	omogućava iscrtavanje raznih podataka tokom izvršenja algoritma vrednost `@optimplotx` crta trenutnu vrednost rešenja vrednost `@optimplotfunccount` crta broj izračunavanja funkcije `fun` vrednost `@optimplotval` crta vrednost funkcije u rešenju vrednost `@optimplotresnorm` crta vrednost norme ostataka vrednost `@optimplotstepstize` crta vrednost korekcije vrednost `@optimplotfirstorderopt` crta meru optimalnosti prvog reda moguće je predati i korisnički definisanu funkciju
`'TolFun'`	podrazumevana vrednost je `1e-6`, a predstavlja gornju granicu vrednosti funkcije za koju se smatra da je konvergencija postignuta
`'TolX'`	podrazumevana vrednost je `1e-6`, a predstavlja gornju granicu apsolutne greške promenljive za prekid izvršenja
`'TypicalX'`	koristi se prilikom određivanja koraka za numeričko diferenciranje, podrazumevana vrednost je vektor ispunjen jedinicama tipa vektora `x0`, kod algoritma `'trust-region-dogleg'` vrednost ovog vektora se koristi kao glavna dijagonala matrice `D` prilikom kontrole oblasti $\|\|D\,d\|\| \leq \Delta$

Tabela 3.7: Polja strukture options i njihova značenja, drugi deo.

```
       funcCount: 15
       algorithm: 'trust-region-dogleg'
   firstorderopt: 3.227482070848464e-12
         message: [1x691 char]
>>output.message
ans =
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

Stopping criteria details:

Equation solved. The sum of squared function values, r = 5.208320e-24,
is less than sqrt(options.TolFun) = 1.000000e-03. The relative norm
of the gradient of r, 3.227482e-12, is less than options.
TolFun = 1.000000e-06.

Optimization Metric                                          Options
relative norm(grad r) =   3.23e-12                  TolFun =   1e-06 (default)
r =   5.21e-24                                sqrt(TolFun) =  1.0e-03 (default)
```

Ovaj niz komandi rešava sistem jednačina $(x_1)^2 + (x_2)^2 = 1$ i $x_1 + x_2 = 0$, sa početnom vrednošću $x^0 = (1, -1)$. Kao što vidimo rešenje je pronađeno, vrednost `exitflag=1`, vrednost funkcije u rešenju je mala. Struktura `output` sadrži sve informacije o toku izvršenja algoritma. Broj upotrebljenih iteracija je četiri, ukupan broj izračunavanja funkcije je 15, upotrebljeni algoritam je `trust-region-dogleg`, vrednost `firstoerderopt` je mala. Ovo je bitan detalj jer nam ukazuje

da je eventualna korekcija koju bismo dobili još jednom upotrebom metoda Newton-Kantoroviča takođe jako mala. Deo polja `message` je kao što vidimo odštampan bez da je eksplicitno pozvan. Da bismo dobili celu poruku iskoristili smo još jednu komandu `output.message`. Dobijena poruka sadrži već dobijenu informaciju o vrednosti $r^2 = ||f||_2$, koja je kao što vidimo manja od `sqrt(options.TolFun)`. Takođe, sadrži i informaciju da je vrednosti `firstorderopt`, the relative norm of the gradient u tekstu poruke, manja od vrednosti `options.FunTol`.

Pozabavimo se opcijama funkcije `fsolve`. Opcije se postavljaju upotrebom funkcije `optimset`. Opcija koja čini bitnu razliku je opcija `'Jacobian'`. Podrazumevana vrednost ove opcije je `'off'`. U ovom slučaju se vrednosti Jacobijeve matrice računa koristeći formule za numeričko diferenciranje. Implementirani algoritmi za numeričko diferenciranje su u suštini prezentovani u glavi o numeričkom diferenciranju. Karakteristika ovih algoritama je velika teškoća pri njihovoj upotrebi, tako da je velika količina opcija vezana upravo za kontrolu o ovih teškoća koje mogu nastati prilikom numeričkog diferenciranja. Ako se vrednost opcija `'Jacobijan'` postavi na vrednost `'on'` podrazumeva se da funkcija `'fun'` kao drugi izlazni argument prilikom izračunavanja vrednosti funkcije vraća i vrednost Jacobijeve matrice sistema. U ovom slučaju nema potrebe za numeričkim diferenciranjem, izuzev ako to eksplicitno zahtevamo. Na primer, za prethodni primer bi funkcija koja vraća vrednost funkcije i Jacobijeve matrice sistema izgledala ovako

```
function [ val,jac ]=testFSolve( x )

val = [ x'*x-1; ones(1,length(x))*x];
jac = [ 2*x'; ones(1,length(x)) ];

end
```

Poziv funkcije `fsolve` u ovom slučaju izgleda ovako

```
>>ops=optimset('Jacobian','on');
>>x=fsolve(@testFSolve,[1; -1],ops)
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
   0.707106781187345
  -0.707106781187345
```

Opcije `'DiffMaxChange'` i `'DiffMinChange'` omogućavaju kontrolu promene korekcije. Opcija `'DiffMaxChange'` omogućava ograničenje dopuštene korekcije sa gornje strane, u smislu da nije moguće primeniti korekciju koja može imati veću vrednost norme od vrednosti ove opcije. Sa druge strane, nije dozvoljena ni korekcija čija je norma manja od vrednosti opcije `'DiffMinChange'`. Ovo se odnosi na one probleme kod kojih se Jacobijeva matrica sistema izračunava koristeći numeričko diferenciranje. Opcija `'DiffMaxChange'` služi da onemogući velike korekcije koje su posledica problema u numeričkom diferenciranju. Dok opcija `'DiffMinChange'` služi da onemogući vrlo male korekcije koje u suštini ne popravljaju rešenje, nego samo povećavaju broj iteracija bespotrebno.

Ako postavimo vrednost opcije `'DiffMaxChange'` na vrednost `10^(-20)` dobićemo poruku da algoritam ne konvergira sa vrednošću `exitflag=-2`, jer je dozvoljena korekcija previše mala.

Opcija `'DerivativeCheck'` omogućava proveru numerički izračunatih Jacobijevih matrica u odnosu na egzaktno određenu vrednost Jacobijevih matrica koja se izračunava kao drugi argument funkcije `fun`. Ova opcija se koristi pre svega za dijagnostiku valjanosti numeričkih metoda koji se koriste za izračunavanje Jacobijevih matrica sistema. Opcija `'FinDiffRelStep'` omogućava određivanje koraka, od strane korisnika, koji se koristi prilikom numeričkih metoda diferenciranja. Određivanje koraka je od najveće važnosti kako je pokazano u glavi koja se bavi numeričkim diferenciranjem. Opcija `'FinDiffType'` omogućava izbor algoritma numeričkog diferenciranja. U suštini postoje dva metoda, metod jednostranog i metod dvostranog diferenciranja. U glavi o numeričkom diferenciranju oba metoda su prezentovana i u detalje objašnjena. Pokazano je da je metod dvostranog diferenciranja znatno stabilniji od metoda jednostranog diferenciranja. Međutim, nije uvek moguće primeniti metod dvostranog diferenciranja, pogotovu kad posmatramo probleme sa ograničenjima. U ovom slučaju, ako smo dovoljno blizu granici oblasti u kojoj tražimo rešenje, ne možemo koristiti metode dvostranog diferenciranja.

Opcija `'Diagnostics'` služi da dobijemo informacije o iterativnom procesu. Opcija `'Display'` služi da dobijemo informacije o procesu tokom iteracija.

```
>>ops=optimset('Diagnostics','on','Display','iter');
>>x=fsolve(@testFSolve,[1;-1],ops)
------------------------------------------------------------
   Diagnostic Information

Number of variables: 2

Functions
Objective:                              testFSolve
Gradient:                               finite-differencing

Algorithm selected                      trust-region-dogleg
------------------------------------------------------------
   End diagnostic information

                                          Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)             step         optimality    radius
     0          3              1                                 2              1
     1          6        0.015625         0.353553           0.188              1
     2          9     1.20563e-05        0.0589256         0.00492              1
     3         12     9.02222e-12       0.00173311        4.25e-06              1
     4         15     5.20832e-24      1.50185e-06        3.23e-12              1

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>s
```

```
x =
   0.707106781187354
  -0.707106781187354
```

Opcije `'MaxIter'` i `'MaxFunEvals'` se koriste za kontrolu broja iteracija i evaluacija prilikom izvršenja algoritma. Opcija `'FunValCheck'` postoji da bi se dala mogućnost korisniku da bude obavešten odmah po detekciji vrednosti funkcije koje ne pripadaju skupu realnih brojeva. Ovo je od značaja jer omogućava detekciju uslova koji onemogućavaju konvergenciju algoritma, tako da korisnik može da prekine izvršenje. Opcija `'PlotFcns'` je sama po sebi jasna i omogućava grafičko praćenje napredovanja algoritma. Opcije `'TolFun'` i `'TolX'` predstavljaju gornje granice apsolutnih grešaka u vrednosti funkcije i promenljive koje određuju uslov uspešne konvergencije. U suštini ako funkcija izađe sa vrednošću `exitflag=1`, onda je ili $||f||_2^2 <$`TolFun` ili $||x||_2^2 <$`TolX`.

Postoje i opcije koje su specifične za upotrebu sa algoritmom `'trust-region-reflective'` a koje se odnose na probleme sa velikim brojem promenljivih. Ove opcije se uglavnom odnose na strukturu Jacobijevih matrica sistema koje u ovom slučaju mogu biti retke matrice. Mi se ovim problemom nećemo baviti.

3.2.5 Metod proste iteracije

Pretpostavimo da treba rešiti sistem jednačina

$$x_1 = \phi_1(x_1, \ldots, x_n), \ldots, x_n = \phi_n(x_1, \ldots, x_n).$$

Ako uvedemo oznake $x = (x_1, \ldots, x_n)$, $\phi(x) = (\phi_1(x), \ldots, \phi_n(x))$, sistem jednačina možemo zapisati u vektorskom obliku

$$x = \phi(x).$$

Kao što vidimo zapis sistema jednačina se ne razlikuje od zapisa u skalarnom slučaju. Čak, pokazaćemo da metod proste iteracije koju smo detaljno obradili u slučaju skalarnih jednačina može biti primenjena i u slučaju sistema jednačina. Metod proste iteracije za sisteme nelinearnih jednačina nam je potreban za numeričko rešavanje sistema običnih diferencijalnih jednačina prediktor-korektor metodom. O ovom metodu rešavanja sistema običnih diferencijalnih jednačina će biti reči u sedmoj glavi.

U slučaju funkcija više promenljivih polazimo od definicije pojma kontrakcije.

Definicija 3.2 *Neka je $\Omega \subset \mathbb{R}^n$ ograničen i zatvoren skup i neka je $\phi : \Omega \to \Omega$. Funkcija ϕ se naziva kontrakcijom ako i samo ako postoji $q \in [0, 1)$, tako da za svako $x, y \in \Omega$ važi*

$$||\phi(x) - \phi(y)|| \leq q||x - y||.$$

U ovom kontekstu q se naziva koeficijent kontrakcije.

Izbor norme vektora u prethodnoj definiciji nije bitan. Dakle, dovoljno je da utvrdimo da je funkcija ϕ kontrakcija u bilo kojoj normi.

Primetimo da su kontrakcije neprekidne funkcije.

Lema 3.1 *Ako je $\phi : \Omega \subset \mathbb{R}^n \to \Omega$ kontrakcija, onda je $\phi \in C(\Omega)$.*

Dokaz. Neka je $q \in [0, 1)$ koeficijent kontrakcije funkcije ϕ. Izaberimo $\varepsilon > 0$ i $x, y \in \Omega$, onda važi

$$||\phi(x) - \phi(y)|| \leq q||x - y|| < \varepsilon,$$

ako je $||x - y|| < \delta$ i $\delta = \varepsilon/q$ za $q \neq 0$, a $\delta = 1$ za $q = 0$. □

Definicija kontrakcije omogućava formulaciju sledeće teoreme.

Teorema 3.7 *Neka je $\phi : \Omega \subset \mathbb{R}^n \to \Omega$, Ω konveksan skup i $\phi \in C^1(\Omega)$. Ako postoji $q \in [0,1)$ tako da za svako $x \in \Omega$ važi $||W(x)|| \leq q < 1$ onda je funkcija ϕ kontrakcija.*

Dokaz. Neka su $x, y \in \Omega$. Iz diferencijabilnosti funkcije ϕ sledi

$$||\phi(x) - \phi(y)|| \leq \sup_{t\in[0,1]} ||W(x + t(x - y))||\, ||x - y||.$$

Kako je $||W(x)|| \leq q$, $x \in \Omega$ važi

$$||\phi(x) - \phi(y)|| \leq \sup_{t\in[0,1]} ||W(x + t(x - y))||\, ||x - y|| \leq q\, ||x - y||,$$

što znači da je funkcija ϕ kontrakcija. □

Sledeća teorema je analogna teoremi 3.1 za skalarne jednačine.

Teorema 3.8 (Metod proste iteracije) *Neka je $\phi : \Omega \subset \mathbb{R}^n \to \Omega$ kontrakcija sa koeficijentom kontrakcije q i neka je $x^0 \in \Omega$. Postoji tačno jedno $\alpha \in \Omega$ tako da važi $\alpha = \phi(\alpha)$.*

Iterativni proces $x^{k+1} = \phi(x^k)$, $k \in \mathbb{N}_0$, konvergira ka α, i važi

$$\limsup_{k\to+\infty} \frac{||x^{k+1} - \alpha||}{||x^k - \alpha||} \leq q.$$

Ako je $\phi \in C^1(\Omega)$ i ako α pripada unutrašnjosti oblasti Ω, onda je

$$\limsup_{k\to+\infty} \frac{||x^{k+1} - \alpha||}{||x^k - \alpha||} \leq ||W(\alpha)||.$$

Dokaz. Jednostavno nalazimo da važi

$$||x^{n+2} - x^{n+1}|| = ||\phi(x^{n+1}) - \phi(x^n)|| \leq q||x^{n+1} - x^n||.$$

Potrebno je samo da pokažemo da je niz x^k Cauchyev. Neka je $m > n$ i $\varepsilon > 0$, onda dobijamo

$$\begin{aligned} ||x^m - x^n|| &\leq ||x^m - x^{m-1}|| + \cdots + ||x^{n+1} - x^n|| \leq (q^{m-n-1} + \cdots + 1)||x^{n+1} - x^n|| \\ &\leq \frac{1 - q^{m-n}}{1 - q} q^n ||x^1 - x^0|| \leq \frac{q^n}{1 - q}||x^1 - x^0|| < \varepsilon, \end{aligned}$$

pod uslovom $n_0 = [\log((1 - q)\varepsilon/||x_1 - x_0||)/\log q] + 1$ i $m > n > n_0$. Zaključujemo da je niz x^k, $k \in \mathbb{N}_0$, Cauchyev.

Kako je skup Ω zatvoren i ograničen to je Ω kompletan prostor, pa svaki Cauchyev niz ima graničnu vrednost. Neka je $\alpha = \lim x^k$, onda je

$$\alpha = \lim_{k\to+\infty} x^{k+1} = \lim_{k\to+\infty} \phi(x^k) = \phi\left(\lim_{k\to+\infty} x^k\right) = \phi(\alpha),$$

gde smo iskoristili činjenicu da je funkcija ϕ neprekidna.

Neka su $\alpha, \beta \in \Omega$ sa osobinom $\alpha = \phi(\alpha)$, $\beta = \phi(\beta)$. Onda

$$||\alpha - \beta|| = ||\phi(\alpha) - \phi(\beta)|| \leq q||\alpha - \beta|| < ||\alpha - \beta||,$$

što je moguće jedino u slučaju $||\alpha - \beta|| = 0$, ili $\alpha = \beta$.

Kako je $||x^{k+1}-\alpha|| \leq q||x^k-\alpha||$ nalazimo $||x^{k+1}-\alpha||/||x^k-\alpha|| \leq q$. Odavde neposredno zaključujemo

$$\limsup_{k\to+\infty} \frac{||x^{k+1}-\alpha||}{||x^k-\alpha||} \leq q.$$

Ako rešenje α pripada unutrašnjosti oblasti Ω i ako je funkcija $\phi \in C^1(\Omega)$, onda za dovoljno veliku vrednost k važi

$$||x^{k+1}-\alpha|| = ||\phi(x^k)-\phi(\alpha)|| \leq \sup_{t\in[0,1]} ||W(\alpha+t(x^k-\alpha))||\,||x^k-\alpha|| = ||W(\alpha+t_k(x^k-\alpha))||\,||x^k-\alpha||,$$

jer je W neprekidna funkcija na Ω, pa postoji $t_k \in [0,1]$ tako da važi

$$\sup_{t\in[0,1]} ||W(\alpha+t(x^k-\alpha))|| = ||W(\alpha+t_k(x^k-\alpha))||.$$

Odavde dobijamo

$$\limsup_{k\to+\infty} \frac{||x^{k+1}-\alpha||}{||x^k-\alpha||} \leq \lim_{k\to+\infty} ||W(\alpha+t_k(x^k-\alpha))|| = ||W(\alpha)||.$$

Ovim smo završili dokaz teoreme. □

Implementacija metoda proste iteracije u vektorskom obliku zahteva samo male promene implementacije metoda proste iteracije u skalarnom obliku i može se dati na sledeći način

```
function [x0,err,iter]=prostaIteracijaVektor(fun,x0,prec,acc,maxIter)
%PROSTAITERACIJAVEKTOR(FUN,X0,PREC,ACC,MAXITER) konstruise niz
% x(k+1)=FUN(x(k))
% koristeci X0 kao pocetnu vrednost, argument PREC odredjuje
% relativnu gresku pri kojoj se prekida iterativni proces,
% parametar ACC odredjuje apsolutnu gresku
% pri kojoj se prekida iterativni proces
% dok argument MAXITER oznacava maksimalni dozvoljen broj iteracija,
% funkcija vraca izracunatu vrednost resenja ili
% vrednost poslednjeg izracunatog elementa niza,
% drugi argument je relativna greska u svakoj iteraciji
% treci argument je broj upotrebljenih iteracija

iter = 0; err=[]; deltaX = Inf*ones(size(x0));
while norm(deltaX) >= max(prec*norm(x0), acc)
    deltaX = x0;
    x0 = fun(x0);
    deltaX = x0 - deltaX;
    iter = iter+1;
    err = [err norm(deltaX)/norm(x0)];
    if iter > maxIter
        disp('maksimalni dozvoljeni broj iteracija dostignut');
        break;
    end
end
```

Kao što vidimo jedina razlika je upotreba norme umesto apsolutne vrednosti.

Iterativni metodi za rešavanje sistema linearnih jednačina su specijalni slučaj metoda proste iteracije za sistem nelinearnih jednačina. Radi ilustracije rešimo sistem linearnih jednačina (2.5) Jacobijevim metodom upotrebom funkcije `prostaIteracijaVektor`. Potrebno je definisati funkciju koja izračunava funkciju ϕ koja u ovom slučaju ima oblik

$$\phi(x) = Bx + \beta, \quad B = \begin{bmatrix} 0 & -1/4 & -1/4 \\ -1/4 & 0 & -1/4 \\ -1/4 & -1/4 & 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1/4 \\ 2/4 \\ 3/4 \end{bmatrix}.$$

Upotrebom funkcije `prostaIteracijaVektor`, sekvenca naredbi koja rešava sistem linearnih jednačina je sledeća

```
>>B=[0 -1/4 -1/4; -1/4 0 -1/4; -1/4 -1/4 0];
>>beta=[1/4 2/4 3/4]';
>>fun=@(x) B*x+beta;
>>[x,err,iter]=prostaIteracijaVektor(fun,[1 1 1]',10^(-6),10^(-10),100)
x =
     -7.94728549635693e-08
         0.333333253860474
         0.666666587193802
err =
  Columns 1 through 2
                         5         0.895350709254191
  Columns 3 through 4
         0.683130051063973         0.270222308417633
  Columns 5 through 6
         0.150894978760176        0.0712820100398246
  Columns 7 through 8
        0.0366528432642735        0.0180699078710638
  Columns 9 through 10
       0.00909862973347778       0.00453333986641817
  Columns 11 through 12
       0.00227065664404848       0.00113433076647107
  Columns 13 through 14
      0.000567414662264751      0.000283644997650735
  Columns 15 through 16
      0.000141838080479733      7.09151446118008e-05
  Columns 17 through 18
      3.54585461861055e-05      1.77290296196502e-05
  Columns 19 through 20
      8.86457567775682e-06      4.43227262184312e-06
  Columns 21 through 22
      2.21614011517384e-06      1.10806910652303e-06
  Column 23
      5.54034791027385e-07
iter =
    23
```

Vidimo da nalazimo rešenje u 23 iteracije i vidimo kako opada relativna greška rešenja. Naravno, u slučaju linearnog sistema jednačina Jacobijeva matrica funkcije ϕ se poklapa sa matricom B, tako da primenjena na slučaj sistema linearnih jednačina teorema 3.8 ne tvrdi ništa novo.

Da bismo ilustrovali metodu u slučaju sistema nelinearnih jednačina možemo iskoristiti sistem nelinearnih jednačina kojim smo ilustrovali upotrebu metoda Newton-Kantoroviča. Dakle, posmatrajmo sistem jednačina

$$f(x) = \begin{bmatrix} 9x_1^2x_2 + 4x_2^2 - 36 \\ 16x_2^2 - x_1^4 + x_2 + 1 \end{bmatrix} = 0. \tag{3.10}$$

Odredimo iterativnu funkciju na sledeći način

$$\phi(x) = x - (W(x))^{-1}f(x) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \frac{9x_1^6 + 8x_1^4x_2 - 4(x_2^2 + 290x_2 + 9) + 9x_1^2(16x_2^2 - 1)}{2x_1(18x_1^4 + 16x_1^2x_2 + 9x_2(32x_2 + 1))} \\ \frac{9x_1^4x_2 + 8x_1^2(x_2^2 - 9) + 9x_2(16x_2^2 + x_2 + 1)}{18x_1^4 + 16x_1^2x_2 + 9x_2(32x_2 + 1)} \end{bmatrix}.$$

Očigledno je rešenje jednačine $f(x) = 0$ ujedno i rešenje jednačine $x = \phi(x)$ jer je

$$\phi(\alpha) = \alpha - (W(\alpha))^{-1}f(\alpha) = \alpha - (W(\alpha))^{-1}0 = \alpha.$$

Na osnovu ovog zapažanja, metod Newton-Kantoroviča možemo shvatiti kao poseban slučaj metoda proste iteracije kao i u slučaju skalarne jednačine. Problem određivanja startnih vrednosti kod metoda Newton-Kantoroviča onda predstavlja problem pronalaženja takve startne vrednosti koja se nalazi u okolini $U(\alpha)$ rešenja α jednačine $f(x) = 0$, tako da je funkcija ϕ kontrakcija na skupu $U(\alpha)$.

Iskoristimo kao startnu vrednost vektor $x^0 = (2, 0)$. Funkcija `prostaIteracijaVektor` vraća sledeće vrednosti

```
function [ y ]=phi(x)
%PHI(X) koristi se za ilustraciju metoda proste iteracije
y=x;
pom=9*(x(1))^6+8*(x(1))^4*x(2)-4*((x(2))^2+290*x(2)+9);
pom=pom+9*(x(1))^2*(16*(x(2))^2-1);
pom=pom/(2*x(1)*(18*(x(1))^4+16*(x(1))^2*x(2)+9*x(2)*(32*x(2)+1)));
y(1)=y(1)-pom;
pom=(9*(x(1))^4*x(2)+8*(x(1))^2*((x(2))^2-9)+9*x(2)*(16*(x(2))^2+x(2)+1));
pom=pom/(18*(x(1))^4+16*(x(1))^2*x(2)+9*x(2)*(32*x(2)+1));
y(2)=y(2)-pom;
end

>>[x,err,iter]=prostaIteracijaVektor(@phi,[2;1],10^(-6),10^(-10),100)
x =
          1.98370873395405
         0.920742637018026
err =
  Columns 1 through 2
        0.0360643131905682       0.00105641283053997
  Column 3
```

```
      7.88637841489313e-07
iter =
     3
```

Naravno, ovde proces konvergira u samo tri iteracije, jer je metod Newton-Kantoroviča reda dva. U opštem slučaju metod proste iteracije ima red konvergencije jedan.

Funkciju `phi` nismo morali da implementiramo, mogli smo da koristimo sintaksu kao i u drugim primerima. Međutim, to bi proizvelo probleme sa zapisom u rukopisu. Naravno, u konkretnoj aplikaciji koristili bismo anonimne funkcije.

Kao još jedna ilustracija metoda proste iteracije može nam poslužiti iterativni proces

$$x^{k+1} = \phi(x^k) = x^k - Af(x^k),$$

gde je x^0 pogodno odabrana početna vrednost, a matrica A je pogodno izabrana konstantna matrica. Neka je α rešenje jednačine $f(x) = 0$ i neka je funkcija f neprekidno diferencijabilna u okolini U tačke α. Izaberimo $x, y \in U$ tako da važi $\overline{xy} \subset U$. Funkcija ϕ je neprekidno diferencijabilna na U i važi

$$||\phi(x) - \phi(y)|| \leq \sup_{t\in[0,1]} ||W_\phi(x + t(y - x)||\,||x - y|| = \sup_{t\in[0,1]} ||I - AW(x + t(y - x))||\,||x - y||,$$

gde smo sa W_ϕ i W označili Jacobijeve matrice funkcija ϕ i f, redom. Kako je funkcija f neprekidno diferencijabilna, to je funkcija $g(x) = ||I - AW(x)||$ neprekidna u okolini U. Primetimo da za $A = W^{-1}(\alpha)$ važi $g(\alpha) = ||I - W^{-1}(\alpha)W(\alpha)|| = 0$. Zbog neprekidnosti funkcije g, u ovom slučaju postoji okolina $\Omega = \{x \,|\, ||x - \alpha|| \leq \varepsilon\}$ tačke α tako da je ispunjen uslov $g(x) \leq 1/2$, $x \in \Omega$. Ako pretpostavimo da je Jacobijeva matrica W reguralna u okolini tačke α, onda je $W^{-1}(x)$ neprekidna funkcija koeficijenata matrice $W(x)$. Tako, za matricu A u ovom slučaju možemo izabrati i matricu $W^{-1}(x)$, gde je tačka x dovoljno blizu tačke α. Za ovako izabranu matricu A možemo tvrditi da je ispunjen uslov $g(x) \leq 1/2$ za $x \in \Omega$. Onda za $x, y \in \Omega$, važi

$$||\phi(x) - \phi(y)|| \leq \frac{1}{2}\,||x - y||,$$

pa je funkcija ϕ kontrakcija. Ako je α rešenje jednačine $f(\alpha) = 0$, onda je $\alpha = \phi(\alpha)$, pa na osnovu prethodnog zaključujemo da za $x \in \Omega$, važi

$$||\phi(x) - \alpha|| = ||\phi(x) - \phi(\alpha)|| \leq \frac{1}{2}\,||x - \alpha|| \leq \frac{\varepsilon}{2} \;\Rightarrow\; \phi(x) \in \Omega.$$

Dakle, $\phi : \Omega \to \Omega$. Na osnovu teoreme 3.8, iterativni proces

$$x^{k+1} = x^k - Af(x^k), \quad x^0 \in \Omega,$$

je konvergentan i konvergira ka rešenju jednačine α.

Najčešće se bira $A = W^{-1}(x^0)$. Za ovako izabranu matricu A, ovaj metod se u literaturi sreće pod imenom modifikovani metod Newton-Kantoroviča. U skalarnom slučaju dobijamo modifikovani Newtonov metod. Primetimo da je red konvergencije modifikovanog metoda Newton-Kantoroviča jedan. Ilustrujmo metod rešavanjem našeg sistema jednačina (3.10), sa $x^0 = (1.98, .92)$. Onda je

$$W(x_0)=\begin{bmatrix} 32.7888 & 42.6436 \\ -31.049568 & 30.44 \end{bmatrix},\; W^{-1}(x_0)=\frac{1}{10}\begin{bmatrix} .13108505356144946 & -.18363793002802314 \\ .13371006190341215 & .14119978988881909 \end{bmatrix}.$$

Koristeći funkciju `prostaIteracijaVektor`, dobijamo

```
>>A=[.013108505356144946 -.018363793002802314;
.013371006190341215 .014119978988881909];
>>f=@(x) [(9*x(1)^2+4*x(2))*x(2)-36; (16*x(2)+1)*x(2)-x(1)^4+1];
>>phi=@(x) x-A*f(x);
>>[x,err,iter]=prostaIteracijaVektor(phi,[1.98;.92],10^(-6),10^(-14),100)
x =
     1.983708733953144e+00
     9.207426370189652e-01
err =
  Columns 1 through 2
     1.733197883234870e-03     4.014078059907867e-06
  Columns 3 through 4
     1.756100876954401e-08     7.114004951849324e-11
  Columns 5 through 6
     2.702047662040399e-13     1.015303907562922e-15
iter =
     6
```

Kao što vidimo potreban broj iteracija je 6. Važi i uslov konvergencije

$$g(\alpha) = ||I - W^{-1}(x_0)W(\alpha)||_2 \approx .0046.$$

Vidimo da je proces linearan, sa prinosom nešto većim od dve cifre po iteraciji. Na osnovu ocene iz teoreme 3.8, prinos značajnih cifara po iteraciji možemo dobiti i teorijski. Važi

$$r_{k+1} = \frac{||x^{k+1} - \alpha||}{||\alpha||} \approx ||I - AW(\alpha)|| \frac{||x^k - \alpha||}{||\alpha||} = ||I - AW(\alpha)||r_k.$$

Ulogu asimptotske konstante greške preuzima veličina $||I - AW(\alpha)||$. Ako odredimo negativan logaritam za osnovu deset, nalazimo

$$-\log_{10} r_{k+1} \approx -\log_{10} ||I - AW(\alpha)|| - \log_{10} r_k,$$

Zaključujemo da je prinos značajnih cifara po iteraciji konstantan i iznosi $-\log_{10} ||I - AW(\alpha)|| \approx 2.33$. Što se zaista i dešava što smo bliže tački α.

Interesantno je pitanje šta se događa sa procesom ako izaberemo $A = W^{-1}(\alpha)$. U ovom slučaju se može pokazati da proces ima red konvergencije dva. Ovu osobinu modifikovanog metoda Newton-Kantoroviča možemo ilustrovati na već korišćenom primeru.

```
>>A=[0.013033286112928644 -0.018303288138670475;
0.013358661741939616 0.014065618432261012];
>>phi=@(x) x-A*f(x);
>>[x,err,iter]=prostaIteracijaVektor(phi,[1.98;.92],10^(-14),10^(-14),100)
x =
     1.983708733953144e+00
     9.207426370189652e-01
err =
  Columns 1 through 2
     1.725796749055426e-03     3.982636322328075e-06
```

```
  Columns 3 through 4
    1.500776058833963e-11                         0
iter =
     4
```

Kao što vidimo proces konvergira velikom brzinom. Ovu osobinu procesa možemo objasniti činjenicom da je ovo praktično metod Newton-Kantoroviča, kad se vrednosti x^k nađu dovoljno blizu rešenju α, jer je W^{-1} neprekidna funkcija pa je $W^{-1}(x^k) \approx W^{-1}(\alpha)$.

Glava 4

Interpolacija

Često smo u mogućnosti da posmatramo promenu neke fizičke veličine sa promenom neke druge fizičke veličine. Na primer, možemo posmatrati promenu dnevne temperature u funkciji vremena, promenu brzine neke hemijske reakcije u funkciji koncentracije reagenasa, promenu brzine strujanja fluida duž cevi i slično. Apstraktno sve rezultate merenja možemo opisati kao skupove uređenih parova $(x_i, f(x_i))$, $i = 0, \ldots, n$, gde su x_i, $i = 0, \ldots, n$, razne vrednosti parametra koji se menja nezavisno, dok su $f(x_i)$, $i = 0, \ldots, n$, vrednosti merene veličine za zadate vrednosti parametra x_i, $i = 0, \ldots, n$. Očigledno je da tačke x_i moraju međusobno biti različite, jer dopuštanje jednakosti nekih od tačaka ne odgovara intuitivnoj postavci problema.

Ideja interpolacije je dobijanje analitičkog izraza p koji opisuje zavisnost merene veličine f u funkciji promenljive x, tako da su ispunjeni interpolacioni uslovi

$$p(x_i) = f(x_i), \quad i = 0, \ldots, n. \tag{4.1}$$

Analitički izraz p, kojeg u daljem tekstu nazivamo interpolant, koji zadovoljava interpolacione uslove možemo konstruisati na razne načine. U ovom kontekstu tačke x_i, $i = 0, \ldots, n$, nazivamo interpolacionim čvorovima i podrazumevamo da su uređene u rastućem poretku $x_0 < x_1 < \cdots < x_n$. Kažemo da interpolant zadovoljava interpolacione uslove (4.1) u interpolacionim čvorovima.

U daljem tekstu pažnja je posvećena algebarskoj interpolaciji. U ovom slučaju p je algebarski polinom najnižeg stepena koji zadovoljava interpolacione uslove (4.1).

Na primer, za skup podataka $\{(0, 1), (1, 2), (2, 3), (3, 4)\}$, algebarski polinom najnižeg stepena koji zadovoljava interpolacione uslove (4.1) je $p(x) = 1 + x$. Ovu činjenicu neposredno proveravamo

$$p(0) = 1 + 0 = 1, \quad p(1) = 1 + 1 = 2, \quad p(2) = 1 + 2 = 3, \quad p(3) = 1 + 3 = 4.$$

Stepen polinoma p ne može biti niži, jer je očigledno da polinom stepena nula ne može biti rešenje.

Egzistencija interpolanta i algoritam za njegovu konstrukciju su bitni problemi kojima se u ovoj glavi bavimo.

Analitički izraz koji predstavlja interpolant nam u izvesnom smislu može zameniti nepoznatu funkciju koju reprezentuje. Dalje razmišljanje u ovom pravcu omogućava, u izvesnom smislu, primenu klasične matematičke analize nad interpolantom i interpretiranje rezultata kao da su dobijeni upotrebom same funkcije. O ovome će posebno biti reči u glavi o numeričkom diferenciranju i integraciji.

4.1 Lagrangeova interpolacija

U ovom odeljku dokazaćemo da za svaki skup podataka problem algebarske interpolacije ima jedinstveno rešenje. Korišćenjem ideje koju je razvio Lagrange dobićemo eksplicitan izraz za reprezentaciju algebarskog interpolanta.

Podrazumevamo u celoj sekciji da je funkcija f zadata vrednostima $f(x_i)$, $i = 0, \ldots, n$, u interpolacionim čvorovima x_i, $i = 0, \ldots, n$, pri čemu su tačke međusobno različite i uređene $x_0 < \cdots < x_n$.

Kao test primer u ovom odeljku koristićemo skup podataka

$$\{(0,1),(1,2),(2,3),(3,4)\}.$$

U ovom slučaju je $n = 3$, $x_0 = 0$, $f(x_0) = 1$, $x_1 = 1$, $f(x_1) = 2$, $x_2 = 2$, $f(x_2) = 3$, $x_3 = 3$ i $f(x_3) = 4$.

Definicija 4.1 *Polinome*

$$L_k(x) = \prod_{j=0, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)} = \frac{(x - x_0)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_0)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)},$$

$k = 0, \ldots, n$, *nazivamo Lagrangeovim bazisom.*

Polinom

$$\omega(x) = \prod_{j=0}^{n}(x - x_j) = (x - x_0)\cdots(x - x_n)$$

nazivamo čvorni polinom.

Polinomi L_k, $k = 0, \ldots, n$, su izuzetno značajni pri konstrukciji algebarskog interpolanta. Zato ih i navodimo. Za naš test primer polinomi L_k, $k = 0, 1, 2, 3$, izgledaju ovako

$$\begin{aligned}
L_0(x) &= \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} = -\frac{1}{6}(x-1)(x-2)(x-3),\\
L_1(x) &= \frac{(x-0)(x-2)(x-3)}{(1-0)(1-2)(1-3)} = \frac{1}{2}(x-0)(x-2)(x-3),\\
L_2(x) &= \frac{(x-0)(x-1)(x-3)}{(2-0)(2-1)(2-3)} = -\frac{1}{2}(x-0)(x-1)(x-3),\\
L_3(x) &= \frac{(x-0)(x-1)(x-2)}{(3-0)(3-1)(3-2)} = \frac{1}{6}(x-0)(x-1)(x-2).
\end{aligned}$$

Polinome Lagrangeovog bazisa možemo pomoću čvornog polinoma izraziti na sledeći način

$$L_k(x) = \frac{\omega(x)}{(x - x_k)\omega'(x_k)}, \quad k = 0, \ldots, n.$$

Polinomi Lagrangeovog bazisa su važni zbog osobine koju iskazuje sledeća lema.

Lema 4.1 *Za $k, i = 0, \ldots, n$ važi*

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

Polinomi L_k, $k = 0, \ldots, n$, su stepena n.

Dokaz. Ako je $k = i$, onda očigledno važi

$$L_k(x_k) = \prod_{j=0, j\neq k}^{n} \frac{(x_k - x_j)}{(x_k - x_j)} = \prod_{j=0, j\neq k}^{n} 1 = 1.$$

Ako je $k \neq i$, onda je

$$L_k(x_i) = \prod_{j=0, j\neq k}^{n} \frac{(x_i - x_j)}{(x_k - x_j)} = 0,$$

jer indeks proizvoda j uzima vrednost i.

Polinom L_k je stepena n jer je proizvod n moničnih polinoma prvog stepena. □

Na našem test primeru možemo proveriti lemu vrlo jednostavno

$$L_0(0) = -\frac{1}{6}(0-1)(0-2)(0-3) = 1, \quad L_0(1) = -\frac{1}{6}(1-1)(1-2)(1-3) = 0.$$

Slično se dobija i za ostale kombinacije indeksa.

Ova osobina Lagrangeovog bazisa je dovoljna za konstrukciju algebarskog interpolanta za naš primer. Posmatrajmo polinom

$$\begin{aligned} P_3(x) &= f(0)L_0(x) + f(1)L_1(x) + f(2)L_2(x) + f(3)L_3(x) \\ &= 1\,L_0(x) + 2\,L_1(x) + 3\,L_2(x) + 4\,L_3(x). \end{aligned}$$

Vrednosti ovog polinoma u interpolacionim čvorovima iznose

$$\begin{aligned} P_3(0) &= 1\,L_0(0) + 2\,L_1(0) + 3\,L_2(0) + 4\,L_3(0) = 1\cdot 1 + 2\cdot 0 + 3\cdot 0 + 4\cdot 0 = 1 = f(0), \\ P_3(1) &= 1\,L_0(1) + 2\,L_1(1) + 3\,L_2(1) + 4\,L_3(1) = 1\cdot 0 + 2\cdot 1 + 3\cdot 0 + 4\cdot 0 = 2 = f(1), \\ P_3(2) &= 1\,L_0(2) + 2\,L_1(2) + 3\,L_2(2) + 4\,L_3(2) = 1\cdot 0 + 2\cdot 0 + 3\cdot 1 + 4\cdot 0 = 3 = f(2), \\ P_3(3) &= 1\,L_0(3) + 2\,L_1(3) + 3\,L_2(3) + 4\,L_3(3) = 1\cdot 0 + 2\cdot 0 + 3\cdot 0 + 4\cdot 1 = 4 = f(3). \end{aligned}$$

Kao što vidimo ovako konstruisan polinom P_3 zadovoljava interpolacione uslove. Da bi polinom P_3 bio interpolant dovoljno je samo pokazati da je polinom P_3 polinom najmanjeg stepena koji zadovoljava interpolacione uslove. Ovaj deo dokaza je integralni deo dokaza sledeće teoreme koja problem rešava na potpuno apstraktnom nivou.

Teorema 4.1 (Lagrangeov interpolacioni polinom) *Neka je zadata funkcija f svojim vrednostima $f(x_i)$, $i = 0, \ldots, n$, u interpolacionim čvorovima x_i, $i = 0, \ldots, n$. Polinom P_n, najnižeg stepena, koji zadovoljava interpolacione uslove*

$$P_n(x_i) = f(x_i), \quad i = 0, \ldots, n,$$

određen je sa

$$P_n(x) = \sum_{k=0}^{n} f(x_k)L_k(x) = f(x_0)L_0(x) + \cdots + f(x_n)L_n(x),$$

gde je L_k, $k = 0, \ldots, n$, Lagrangeov bazis.

Dokaz. Zadovoljenost interpolacionih uslova se proverava jednostavno. Koristeći lemu 4.1, izračunavamo

$$P_n(x_i) = f(x_0)L_0(x_i) + \cdots + f(x_i)L_i(x_i) + \cdots + f(x_n)L_n(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Polinomi L_k imaju stepen n na osnovu leme 4.1. Kako je P_n linearna kombinacija polinoma L_k, to i polinom P_n ima stepen najviše n.

Egzistenciju bar jednog polinoma stepena ne većeg od n koji zadovoljava interpolacione uslove smo utvrdili. Dokažimo da je polinom P_n jedinstven u skupu polinoma ne većeg stepena od n.

Neka polinom Q_n, ne većeg stepena od n, takođe zadovoljava interpolacione uslove $Q_n(x_i) = P_n(x_i) = f(x_i)$, $i = 0, \ldots, n$. Posmatrajmo razliku $R = Q_n - P_n$. Polinom R je razlika dva polinoma ne većeg stepena od n, pa je i sam ne većeg stepena od n i $R(x_i) = Q_n(x_i) - P_n(x_i) = f(x_i) - f(x_i) = 0$, $i = 0, \ldots, n$. Zaključujemo da je R polinom ne većeg stepena od n koji ima $n+1$ nulu. To je moguće jedino u slučaju $R = 0$. Onda je $Q_n = P_n$. Zaključujemo da je polinom P_n jedinstven u skupu polinoma ne većeg stepena od n. Samim tim je P_n polinom najnižeg stepena koji zadovoljava interpolacione uslove. □

U daljem tekstu algebarski interpolant konstruisan u $n + 1$ čvorova označavamo sa P_n. Na osnovu ove teoreme vidimo da je stepen polinoma najviše n. Ako želimo da naglasimo funkciju f korišćenjem čijih vrednosti je polinom konstruisan koristimo notaciju $P_n(\cdot; f)$.

Na osnovu teoreme 4.1, zaključujemo da je rešenje interpolacionog problema sa $n + 1$ interpolacionih čvorova jedinstveno u skupu algebarskih polinoma ne većeg stepena od n. Rešenje ne mora biti jedinstveno u skupu polinoma višeg stepena. Na primer, posmatrajmo skup podataka $\{(0, 1), (1, 2), (2, 3), (3, 4)\}$. Rešenje problema interpolacije u skupu polinoma ne većeg stepena od 3 je polinom $p(x) = 1 + x$. Ovo je jedinstveno rešenje u ovom skupu polinoma. Ako posmatramo skup polinoma ne većeg stepena od 5, možemo konstruisati beskonačno mnogo rešenja

$$p(x) = 1 + x, \quad q_\lambda(x) = (1 + x)(1 + \lambda x(x - 1)(x - 2)(x - 3)), \quad \lambda \in \mathbb{R}.$$

Direktna posledica teoreme 4.1 je sledeća lema.

Lema 4.2 *Neka je q polinom stepena ne većeg od m i neka je polinom P_n interpolacioni polinom koji zadovoljava interpolacione uslove*

$$P_n(x_i) = q(x_i), \quad i = 0, \ldots, n,$$

pri čemu je $n \geq m$. Onda je $P_n = q$. Drugim rečima, polinom q je identičan svom interpolacionom polinomu pod uslovom da je broj interpolacionih tačaka barem za jedan veći od stepena polinoma.

Dokaz. Interpolacione uslove zadovoljavaju i polinom P_n i polinom q, jer je $P_n(x_i) = q(x_i) = q(x_i)$, $i = 0, \ldots, n$, pa su polinomi P_n i q rešenja interpolacionog problema. Međutim, rešenje problema je jedinstveno u skupu polinoma ne većeg stepena od n zbog teoreme 4.1. Kako je $n \geq m$ mora biti $P_n = q$. □

4.1.1 Numerička konstrukcija Lagrangeovog interpolanta

Najefikasniji način numeričke konstrukcije algebarskog interpolanta je konstrukcija upotrebom Lagrangeovog bazisa. Sledeća funkcija implementira konstrukciju Lagrangeovog interpolanta korišćenjem Lagrangeovog bazisa.

```
function p=lagrangePoly(cvorovi,vrednosti,x)
%LAGRANGEPOLY(CVOROVI,VREDNOSTI,X) izracunava vrednost Lagrangeovog
% interpolanta u vektoru X, konstruisanog u CVOROVIMA za vrednosti
% funkcije VREDNOSTI, koristeci Lagrangeov bazis

n = length(cvorovi);
p = zeros(length(x),1);
for k = 1:n
    pro = ones(length(x),1);
    for j = [ 1:k-1 k+1:n ]
        pro = (x-cvorovi(j))./(cvorovi(k)-cvorovi(j)).*pro;
    end
    p = p+vrednosti(k)*pro;
end
```

Funkcija `lagrangePoly` izračunava vrednost Lagrangeovog polinoma u vektoru `x`, koji zbog jednostavnosti funkcije mora biti vektor kolona, ulazni argument čvorovi predstavlja interpolacione čvorove, ulazni argument `vrednosti` predstavlja vrednosti funkcije u interpolacionim čvorovima.

Koristeći funkciju `lagrangePoly` možemo nacrtati interpolant za, recimo, funkciju log zajedno sa grafikom funkcije. Slika 4.1, levo, prikazuje grafik Lagrangeovog interpolanta i grafik funkcije log dobijene sledećim skriptom.

```
x=(1:.1:10)';
c=2+7*rand(1,5);
plot(x,lagrangePoly(c,log(c),x),'--',x,log(x),'-',c,log(c),'+');
title('grafici funkcija log i Lagrangeovog interpolanta u 5 tacaka');
ylabel('y'); xlabel('x');
legend('interpolant','log','cvorovi','Location','NorthWest');
grid on;
```

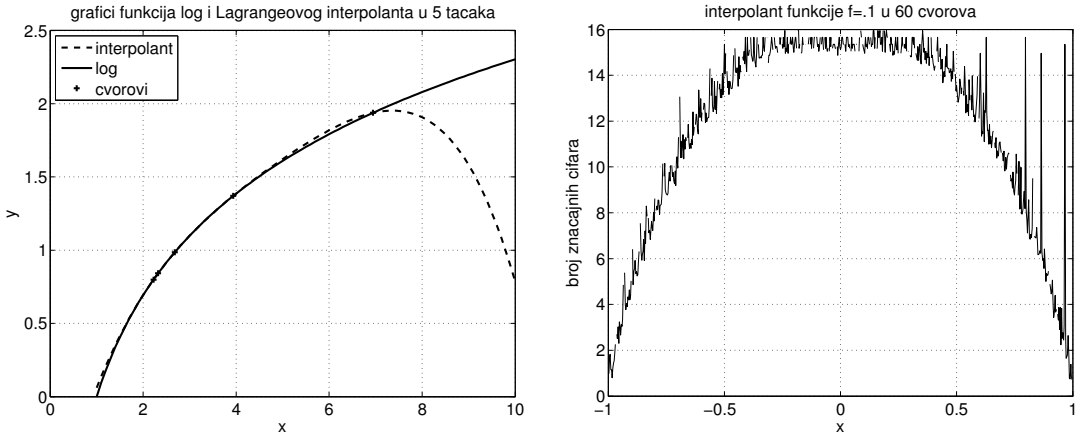
Interpolacioni čvorovi su izabrani slučajno u intervalu $(2, 9)$. Vidimo da se grafici funkcije i interpolanta dobro poklapaju u intervalu kome pripadaju interpolacioni čvorovi. Međutim, ako izađemo van intervala kome pripadaju interpolacioni čvorovi primećujemo da razlike postaju sve veće i veće.

Ako funkciju `lagrangePoly` pozovemo sa simboličkom promenljivom kao trećim argumentom dobićemo interpolacioni polinom u simboličkom obliku. Na primer, sledeće naredbe konstruišu interpolacioni polinom u simboličkom obliku za skup podataka $\{(-1, -1), (0, 2), (2, 10), (3, 35)\}$.

```
>>syms t clear;
>>p=expand(lagrangePoly([-1 0 2 3],[-1 2 10 35],t))
p =
(5*t^3)/3 - (4*t^2)/3 + 2
```

Primetimo da su brojevi u rešenju zadati u obliku razlomaka, što je posledica toga što dolazi do konverzije realnih vrednosti u razlomljene brojeve pri radu sa simboličkim izrazima.

Najčešći problem koji nastaje prilikom konstrukcije interpolanta je numeričke prirode. Najlakše se problem ilustruje konstrukcijom interpolacionog polinoma za konstantnu funkciju. Kako je konstatna funkcija polinom nultog stepena, na osnovu leme 4.2, Lagrangeov interpolacioni polinom bilo kog stepena bi morao biti jednak pomenutoj konstanti. Slika 4.1, desno, prikazuje značajne cifre interpolacionog polinoma konstruisanog za funkciju $f(x) = .1$ u 60 interpolacionih čvorova



Slika 4.1: Slika levo prikazuje grafike funkcije log i Lagrangeovog interpolanta u 5 tačaka, slika desno prikazuje značajne cifre interpolacionog polinoma funkcije $f(x) = .1$ konstruisanog u interpolacionim čvorovima $x_i = -1 + 2i/59$, $i = 0, \ldots, 59$.

koji su ekvidistantno raspoređeni na intervalu $[-1, 1]$, pri čemu je $x_0 = -1$ i $x_{59} = 1$. Skript kojim je generisana slika je

```
x=[-1:.01:1];
n=59;
c=linspace(-1,1,n+1);
plot(x,-log10(abs(lagrangePoly(c,.1*ones(1,n+1),x')/.1-1)));
title('interpolant funkcije f=.1 u 60 cvorova');
ylabel('broj znacajnih cifara'); xlabel('x');
grid on;
syms t clear;
p=expand(lagrangePoly(c,.1*ones(1,n+1),t));
p
```

Posle izvršenja promenljiva `p` ima vrednost $1/10$, dok se kao što vidimo grafik interpolacionog polinoma prilično razlikuje od grafika konstantne funkcije kad se približavamo tačkama minus jedan i jedan. Problem postaje još značajniji ako dodamo još više interpolacionih čvorova. Da bismo objasnili grešku koja nastaje prilikom konstrukcije interpolanta dajemo sledeća razmatranja.

Prilikom konstrukcije interpolacionog polinoma raspolažemo vrednostima funkcije koje su zadate sa nekom apsolutnom greškom. Ova apsolutna greška u podacima funkcije se reflektuje u vrednostima interpolanta. Sledeća teorema daje gornju granicu apsolutne greške interpolanta koja nastaje zbog greške u podacima.

Teorema 4.2 *Neka su P_n i $\tilde{P}_n$ interpolanti konstruisani za skupove padataka $\{(x_i, f(x_i)) \mid i = 0, \ldots, n\}$ i $\{(x_i, \tilde{f}(x_i) \mid i = 0, \ldots, n\}$, redom, pri čemu je*

$$\max_{i=0,\ldots,n} |f(x_i) - \tilde{f}(x_i)| \leq \varepsilon.$$

Onda važi sledeća ocena

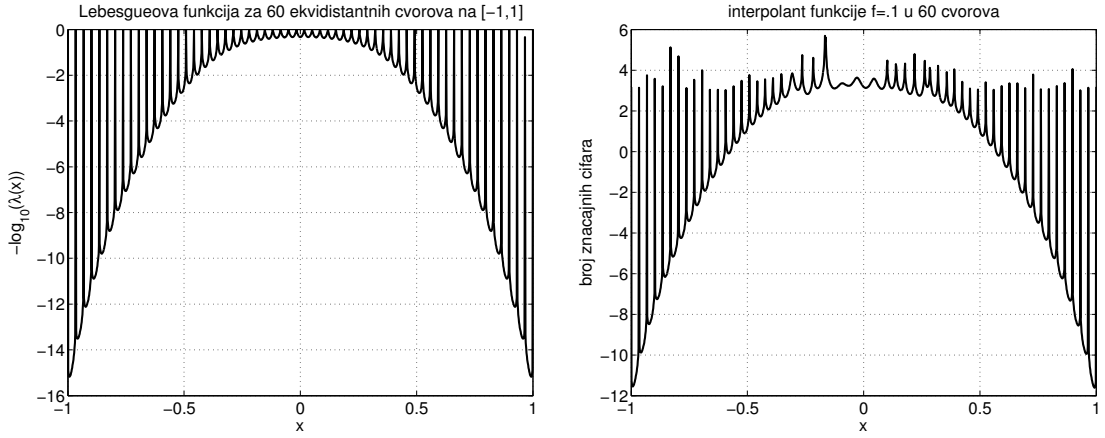
$$|\tilde{P}_n(x) - P_n(x)| \leq \varepsilon \lambda(x),$$

gde je $\lambda(x) = \sum_{k=0}^{n} |L_k(x)|$ *Lebesgueova funkcija.*

Dokaz. Koristeći teoremu 4.1, oduzimanjem eksplicitnih izraza za interpolante $\tilde{P}_n$ i P_n, nalazimo

$$\begin{aligned}\left|\tilde{P}_n(x) - P_n(x)\right| &= \left|\sum_{k=0}^{n} \tilde{f}(x_k)L_k(x) - \sum_{k=0}^{n} f(x_k)L_k(x)\right| = \left|\sum_{k=0}^{n} (\tilde{f}(x_k) - f(x_k))L_k(x)\right| \\ &\leq \sum_{k=0}^{n} |\tilde{f}(x_k) - f(x_k)|\,|L_k(x)| \leq \max_{k=0,\ldots,n} |\tilde{f}(x_k) - f(x_k)| \sum_{k=0}^{n} |L_k(x)| \\ &\leq \varepsilon\lambda(x),\end{aligned}$$

čime je dokaz teoreme završen. □



Slika 4.2: Slika levo prikazuje grafik Lebesgueove funkcije konstruisane za interpolacione čvorove $x_i = -1+2i/59$, $i = 0, \ldots, 59$, slika desno prikazuje značajne cifre interpolanta za funkciju $f(x) = .1$ za ekvidisantne čvorove $x_i = -1 + 2i/59$, $i = 0, \ldots, 59$, ali sa gornjom granicom relativne greške u vrednostima funkcije 10^{-3}.

Ova teorema ukazuje da se gornja granica apsolutne greške interpolanta ponaša kao Lebesgueova funkcija. Slika 4.2, levo, prikazuje grafik funkcije $-\log_{10}(\lambda(x))$ za interpolacione čvorove ekvidistantno raspoređene na intervalu $[-1, 1]$.

Vratimo se interpolaciji konstantne funkcije i slici 4.1, desno. Možemo tvrditi da su vrednosti funkcije date sa gornjom granicom relativne greške $2^{-53} \approx 10^{-16}$, jer je to gornja granica relativne greške koju nam obezbeđuje dvostruka preciznost. Interpolacioni polinom konstruisan za egzaktne podatke, u ovom slučaju, iznosi $P_n(x) = .1$. Primenom teoreme 4.2, dobijamo značajne cifre interpolacionog polinoma konstruisanog za skup podataka sa greškom

$$-\log_{10} |\tilde{P}_n(x)/P_n(x) - 1| \geq -\log_{10} \frac{\varepsilon}{.1} - \log_{10} \lambda(x) = 16 - \log_{10} \lambda(x).$$

Vidimo da grafik broja značajnih cifara, prikazan na slici 4.1, desno, možemo da dobijemo prostom translacijom grafika funkcije $-\log_{10} \lambda(x)$ za vrednost 16, što jednostavno proveravamo.

Vidimo da se teorijska granica dobro slaže sa granicom dobijenom eksperimentom[1]. Sa slike se vidi da je minimum broja značajnih cifara otprilike 1, koji se dostiže u blizini tačaka ± 1.

Funkcija koja implementira izračunavanje Lebesgueove funkcije može se dati na sledeći način.

```
function p=lebesgueFunction(cvorovi,x)
%LEBESGUEFUNCTIN(CVOROVI,X) izracunava Lebesgueovu funkciju
% za cvorove CVOROVI u vektoru X

n = length(cvorovi);
p = zeros(length(x),1);
for k = 1:n
    pro = ones(length(x),1);
    for j = [ 1:k-1 k+1:n ]
        pro = (x-cvorovi(j))./(cvorovi(k)-cvorovi(j)).*pro;
    end
    p = p+abs(pro);
end
end
```

Ako su podaci kojim je zadata funkcija rezultat nekog merenja, gornja granica apsolutne greške podataka je obično još veća. Obično je u ovom slučaju gornja granica relativne greške 10^{-3}. U ovom slučaju analiza ostaje u važnosti. Na primer, sledeći skript konstruiše interpolant u 60 tačaka konstantne funkcije $f(x) = .1$, na intervalu $[-1, 1]$, pri čemu su vrednosti funkcije slučajno promenjene sa gornjom granicom apsolutne greške 10^{-4}.

```
n=59;
c=linspace(-1,1,n+1);
x=-1:(1/(10*n)):1;
v=.1*ones(1,n+1)+10^(-4)*(2*rand(1,n+1)-1);
plot(x,-log10(abs(lagrangePoly(c,v,x')/.1-1)));
title('interpolant funkcije f=.1 u 60 cvorova');
ylabel('broj znacajnih cifara'); xlabel('x');
grid on;
```

Slika 4.2, desno, prikazuje grafik izvršenog skripta. Na ovoj slici se vidi još jasnije da je ovo samo grafik funkcije $-\log_{10}\lambda(x)$ transliran za približno vrednost $\log_{10}(10^{-4}/.1) = 3$. Primetimo da je relativna greška reda veličine 10^{11} što predstavlja stvarno veliku vrednost. Situacija će naravno postati još upečatljivija ako povećamo broj interpolacionih čvorova.

Lebesgueova funkcija zavisi od izbora interpolacionih čvorova. Za različite izbore interpolacionih čvorova Lebesgueova funkcija se ponaša različito. U stvari postoji specijalan izbor interpolacionih čvorova pri kojem se Lebegueova funkcija ponaša najbolje. Ovakav izbor interpolacionih čvorova će biti proučen u sekciji 4.1.3.

[1]Ovde smo upotrebili interpolaciju funkcije $f(x) = .1$. Razlog je što broj .1 prilikom reprezentacije u formatu dvostruke preciznosti ne može biti reprezentovan egzaktno. Međutim, dubljom analizom se može pokazati da su zaključci analize ispravni i za konstantne funkcije čije je vrednosti moguće egzaktno prikazati u formatu dvostruke preciznosti. Mi se ovom dubljom analizom nećemo baviti.

4.1.2 Greška interpolacije

U ovom odeljku dajemo ocenu apsolutne greške vrednosti interpolanta u odnosu na vrednost funkcije za koju je interpolant konstruisan. Pretpostavićemo da interpolacioni čvorovi pripadaju intervalu $[a, b]$.

Teorema 4.3 *Neka je $f \in C^{n+1}[a, b]$ i neka je P_n Lagrangeov interpolant konstruisan u čvorovima $x_i \in [a, b]$, $i = 0, \ldots, n$, i vrednostima $f(x_i)$, $i = 0, \ldots, n$. Za svako $x \in [a, b]$ postoji $\psi \in (\min\{x, x_0, \ldots, x_n\}, \max\{x, x_0, \ldots, x_n\}) \subset [a, b]$, tako da važi*

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\psi)}{(n+1)!}\omega(x).$$

Dokaz. Posmatrajmo funkciju

$$F(x) = f(x) - P_n(x) - K\omega(x).$$

Ova funkcija uzima vrednost nula u bar $(n + 1)$-noj tački intervala $[a, b]$. Naravno, ove tačke su interpolacioni čvorovi. Izaberimo proizvoljno $x \in (a, b)$ koje ne pripada skupu interpolacionih čvorova. Vrednost K je proizvoljna i možemo je izabrati tako da važi $F(x) = 0$. Dovoljno je uzeti

$$K = \frac{f(x) - P_n(x)}{w(x)}.$$

Sada funkcija F ima bar $n + 2$ nule na intervalu $[c, d]$, gde je $c = \min\{x, x_0, \ldots, x_n\}$, $d = \max\{x, x_0, \ldots, x_n\}$. Na osnovu Rolleove teoreme sukcesivno zaključujemo da funkcija F' ima bar $n+1$ različitu nulu na intervalu (c, d), funkcija F'' ima bar n različitih nula na intervalu (c, d), i na kraju funkcija $F^{(n+1)}$ ima bar jednu nulu na intervalu (c, d). Neka je to tačka ψ. Onda je

$$F^{(n+1)}(\psi) = f^{(n+1)}(\psi) - P_n^{(n+1)}(\psi) - K\omega^{(n+1)}(\psi) = 0.$$

Međutim, $P_n^{(n+1)} = 0$, jer je polinom P_n najviše n-tog stepena, dok je $\omega^{(n+1)} = (n + 1)!$, jer je ω polinom $(n + 1)$-vog stepena. Odavde dobijamo

$$K = \frac{f^{(n+1)}(\psi)}{(n+1)!}.$$

Kombinujući sa prethodnim izrazom za K, dobijamo tvrđenje teoreme. □

Na osnovu ove teoreme može se dati teorema koja daje ocenu gornje granice apsolutne greške uniformno na intervalu od interesa. Da bismo lakše iskazali rezultate sledećih teorema uvedimo oznaku

$$M_{n+1} = \max_{x \in [a,b]} |f^{(n+1)}(x)|.$$

Teorema 4.4 *Neka je $f \in C^{n+1}[a, b]$ i neka je P_n Lagrangeov interpolant konstruisan u čvorovima $x_i \in [a, b]$, $i = 0, \ldots, n$, i vrednostima $f(x_i)$, $i = 0, \ldots, n$. Za svako $x \in [a, b]$ važi*

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\omega(x)|.$$

Dokaz. Koristeći prethodnu teoremu i činjenicu da je

$$|f^{(n+1)}(\psi)| \leq \max_{\psi\in[a,b]} |f^{(n+1)}(\psi)| = M_{n+1}$$

dobijamo tvrđenje teoreme. □

U slučaju ekvidistantnih čvorova ova teorema se može znatno uprostiti.

Teorema 4.5 *Neka je $f \in C^{n+1}[a,b]$ i neka je P_n Lagrangeov interpolant konstruisan u čvorovima $x_i \in [a,b]$, $x_i = a + ih$, $i = 0,\ldots,n$, $h = (b-a)/n$ i vrednostima $f(x_i)$, $i = 0,\ldots,n$. Za svako $x \in [a,b]$ važi sledeća ocena*

$$|f(x) - P_n(x)| \leq \frac{h^{n+1}}{4(n+1)} M_{n+1}.$$

Ove teoreme mogu pomoći prilikom ocene greške koja se čini prilikom zamene funkcije njenim interpolacionim polinomom na nekom intervalu $[a,b]$. Na primer, ocenimo gornju granicu apsolutne greške koju činimo zamenom funkcije log, na intervalu $[2,10]$, njenim interpolacionim polinomom koji je konstruisan nad skupom podataka zadatih tabelom 4.1. Koristeći vrednosti iz tabele možemo

x_k	2.4	3.7	5.1	7.4	9.8
$\log x_k$	.875469	1.30833	1.62924	2.00148	2.28238

Tabela 4.1: Vrednosti funkcije log u pojedinim tačkama.

konstruisati Lagrangeov interpolant konstrukcijom Lagrangeovog bazisa.

$$\begin{aligned}
L_0(x) &= \frac{(x-3.7)(x-5.1)(x-7.4)(x-9.8)}{(2.4-3.7)(2.4-5.1)(2.4-7.4)(2.4-9.8)} \\
&= 7.70001\cdot 10^{-3}(x-3.7)(x-5.1)(x-7.4)(x-9.8), \\
L_1(x) &= \frac{(x-2.4)(x-5.1)(x-7.4)(x-9.8)}{(3.7-2.4)(3.7-5.1)(3.7-7.4)(3.7-9.8)} \\
&= -2.43443\cdot 10^{-2}(x-2.4)(x-5.1)(x-7.4)(x-9.8), \\
L_2(x) &= \frac{(x-2.4)(x-3.7)(x-7.4)(x-9.8)}{(5.1-2.4)(5.1-3.7)(5.1-7.4)(5.1-9.8)} \\
&= 2.44727\cdot 10^{-2}(x-2.4)(x-3.7)(x-7.4)(x-9.8), \\
L_3(x) &= \frac{(x-2.4)(x-3.7)(x-5.1)(x-9.8)}{(7.4-2.4)(7.4-3.7)(7.4-5.1)(7.4-9.8)} \\
&= -9.79240\cdot 10^{-3}(x-2.4)(x-3.7)(x-5.1)(x-9.8), \\
L_4(x) &= \frac{(x-2.4)(x-3.7)(x-5.1)(x-7.4)}{(9.8-2.4)(9.8-3.7)(9.8-5.1)(9.8-7.4)} \\
&= 1.96395\cdot 10^{-3}(x-2.4)(x-3.7)(x-5.1)(x-7.4).
\end{aligned}$$

Na osnovu ovako konstruisanog Lagrangeovog bazisa možemo konstruisati Lagrangeov interpolacioni polinom

$$\begin{aligned}P_4(x) &= \log x_0\, L_0(x) + \log x_1\, L_1(x) + \log x_2\, L_2(x) + \log x_3\, L_3(x) + \log x_4\, L_4(x)\\ &= 6.74112 \cdot 10^{-3}(x-3.7)(x-5.1)(x-7.4)(x-9.8)\\ &- 3.18504 \cdot 10^{-2}(x-2.4)(x-5.1)(x-7.4)(x-9.8)\\ &+ 3.98720 \cdot 10^{-2}(x-2.4)(x-3.7)(x-7.4)(x-9.8)\\ &- 1.95993 \cdot 10^{-2}(x-2.4)(x-3.7)(x-5.1)(x-9.8)\\ &+ 4.48247 \cdot 10^{-3}(x-2.4)(x-3.7)(x-5.1)(x-7.4).\end{aligned}$$

Gornja granica apsolutne greške, na osnovu teoreme 4.4, je određena sa

$$|\log x - P_4(x)| \leq \frac{M_5}{5!}|\omega(x)|.$$

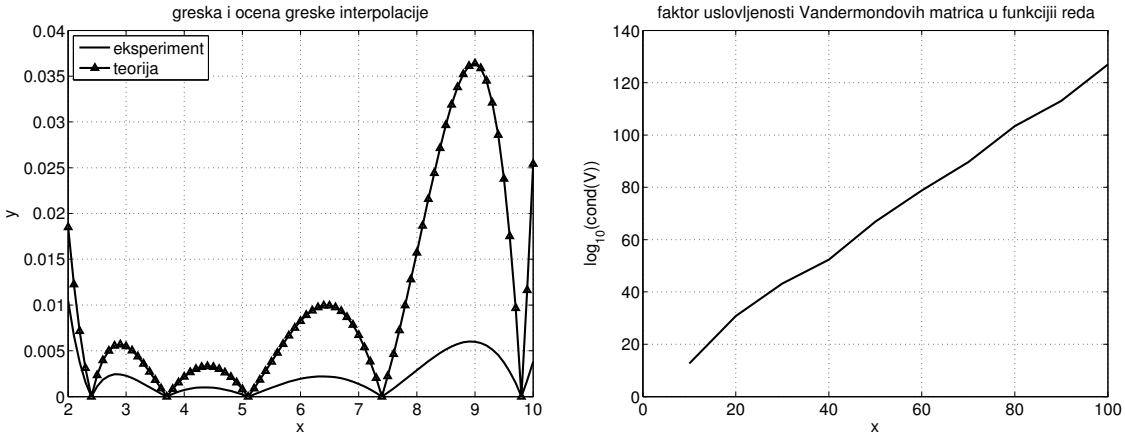
Vrednost konstante M_5 je

$$M_5 = \max_{x\in[2,10]} |(\log x)^{(5)}| = \max_{x\in[2,10]} \frac{4!}{x^5} = \frac{4!}{2^5} = \frac{3}{4}.$$

Odavde dobijamo

$$|\log x - P_4(x)| \leq \frac{1}{160}\,|(x-2.4)(x-3.7)(x-5.1)(x-7.4)(x-9.8)| = \frac{1}{160}|w(x)|.$$

Nažalost, ova ocena je prilično pesimistična. Slika 4.3, levo, prikazuje grafike funkcija $|\log x - P_4(x)|$ i $1/30 \cdot 1/160\,|\omega(x)|$. Morali smo da pomnožimo faktorom $1/30$ desnu stranu nejednakosti da bismo na jednom grafiku mogli da prikažemo grafike i leve i desne strane nejednakosti. Vidimo da je desna strana nejednakosti mnogo veća od leve.



Slika 4.3: Slika levo prikazuje grafike funkcija $|\log x - P_4(x)|$ i $1/30 \cdot 1/160 \cdot |\omega(x)|$ za interpolacione čvorove $\{2.4, 3.7, 5.1, 7.4, 9.8\}$, slika desno prikazuje zavisnost faktora uslovljenosti Vandermondove matrice u funkciji reda.

Gornja granica apsolutne greške prilikom zamene vrednosti funkcije log, u tački 5.5, vrednošću interpolacionog polinoma, iznosi

$$\frac{1}{160}|(5.5-2.4)(5.5-3.7)(5.5-7.4)(5.5-9.8)| \approx 2.85 \cdot 10^{-1}.$$

Stvarna apsolutna greška iznosi

$$|\log 5.5 - P_4(5.5)| \approx 9.6 \cdot 10^{-4}.$$

Teoreme koje govore o grešci interpolacije jasno ilustruju važnost čvornog polinoma ω kao polinoma koji određuje gornju granicu apsolutne greške interpolacije. Međutim, ponašanje polinoma ω u mnogome zavisi od izbora interpolacionih čvorova. U stvari postoji izbor interpolacionih čvorova pri kome se funkcija ω ponaša najbolje. To je tema sekcije 4.1.3.

4.1.3 Izbor interpolacionih čvorova

Nismo uvek u mogućnosti da biramo interpolacione čvorove. U stvari, to je vrlo retka privilegija. Ova sekcija služi da razvije svest da sa gledišta minimizacije greške interpolacije postoji izbor interpolacionih čvorova koji postiže optimum.

Primetimo da je interpolant uvek konstruisan na računskoj mašini i da podaci o vrednosti funkcije nikada nisu i ne mogu biti apsolutno tačni. Konstruisani interpolant, kao posledica, je uvek konstruisan nad približnim skupom podataka. Označimo ovako konstruisan interpolant sa $\tilde{P}_n$, dok sa P_n označimo interpolant konstruisan sa egzaktnim skupom podataka. Na osnovu teorema 4.2 i 4.3, grešku interpolanta možemo predstaviti na sledeći način

$$|f(x) - \tilde{P}_n(x)| \leq |f(x) - P_n(x)| + |P_n(x) - \tilde{P}_n(x)| \leq \frac{|f^{(n+1)}(\psi)|}{(n+1)!}|\omega(x)| + \varepsilon\lambda(x), \tag{4.2}$$

gde je ψ ima odgovarajuće značenje iz teoreme 4.3, dok je ε gornja granica apsolutne greške vrednosti funkcije u interpolacionim čvorovima. Funkcije λ i ω su Lebesgueova funkcija i čvorni polinom za zadati skup interpolacionih čvorova.

Ovako konstruisan izraz za grešku ima dva sabirka. Pod određenim uslovima svaki od sabiraka može imati dominantan uticaj na grešku interpolacije. Prvi sabirak uglavnom dominira kad je broj interpolacionih čvorova relativno mali, mada ovo ne mora biti tačno za sve moguće izbore interpolacionih čvorova i za sve moguće izbore funkcija koje interpoliramo. Drugi sabirak uglavnom dolazi do izražaja kad je broj interpolacionih čvorova relativno veliki. Posebno drugi sabirak dolazi do izražaja, ako interpoliramo polinom stepena m, pri čemu je $m < n$. U ovom slučaju je prvi sabirak jednak nuli, jer je odgovarajući izvod polinoma jednak nuli. Imajući ovo u vidu razmatraćemo konfiguracije interpolacionih čvorova koje minimiziraju vrednost svakog od sabiraka.

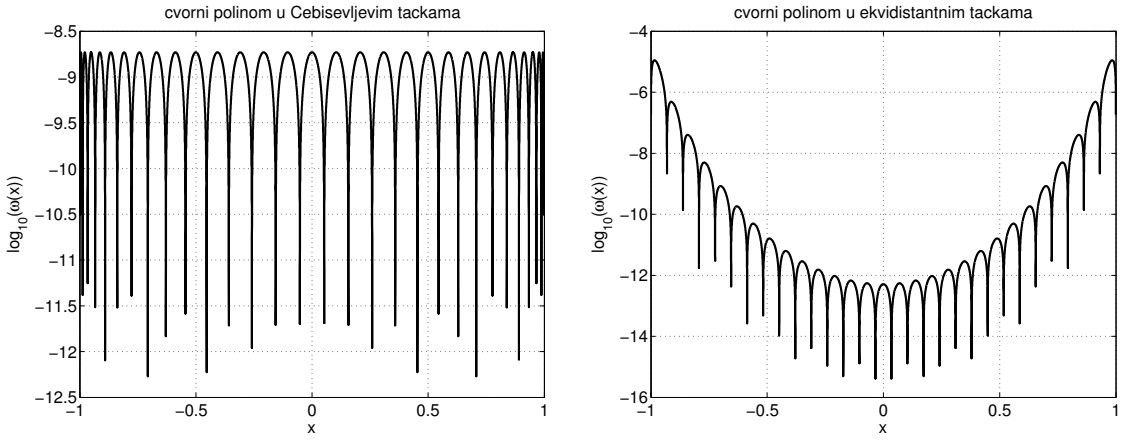
Minimizacija vrednosti čvornog polinoma

Prilikom analize izraza za grešku interpolacije postaje jasno da je povoljna konfiguracija interpolacionih čvorova ona za koju čvorni polinom ima minimalne vrednosti. Postavlja se pitanje: kako izabrati interpolacione čvorove tako da čvorni polinom ω, zadat sa

$$\omega(x) = (x - x_0) \cdots (x - x_n),$$

ima najmanju moguću vrednost maksimuma među svim moničnim polinomima stepena $n+1$ čije nule pripadaju određenom intervalu.

Ovaj problem obično posmatramo na normalizovanom intervalu $[-1,1]$, u koji lokalizujemo interpolacione čvorove. Rezultate možemo preneti na proizvoljni interval $[a,b]$ prostom linearnom transformacijom promenljive $a+(b-a)(x+1)/2$.



Slika 4.4: Slika levo prikazuje grafik čvornog polinoma za interpolacione čvorove koji su zadati Čebiševljevim tačkama, ima ukupno 30 čvorova, slika desno prikazuje grafik čvornog polinoma za slučaj ekvidistantno izabranih interpolacionih čvorova, ukupno ima 30 čvorova.

Problem dobija sledeću formulaciju: u skupu svih moničnih polinoma stepena $n+1$, čije se nule nalaze unutar intervala $[-1,1]$, naći onaj polinom čija apsolutna vrednost ima najmanji maksimum na intervalu $[-1,1]$. Ovo je čuveni Čebiševljev mini-maks problem, čije rešenje predstavlja niz Čebiševljevih polinoma prve vrste $\hat{T}_n$, $n \in \mathbb{N}_0$. Polinom $\hat{T}_n$ je stepena n. Ovaj niz polinoma ima razne interesantne osobine. Za nas je najznačajnija sledeća

$$\max_{x\in[-1,1]} |\omega(x)| \geq \max_{x\in[-1,1]} |\hat{T}_n(x)| = \begin{cases} 2^{1-n}, & n>0, \\ 1, & n=0, \end{cases}$$

gde je ω bilo koji monični polinom stepena n. Čebiševljev polinom $\hat{T}_n$, pored toga, ima sve nule unutar intervala $[-1,1]$ i nule su zadate sa

$$x_i = -\cos\frac{(2i-1)\pi}{2n}, \quad i=1,\ldots,n.$$

Na osnovu rečenog se može zaključiti da je optimalno, s obzirom na minimizaciju izraza za grešku interpolacije, izabrati interpolacione čvorove u nulama Čebiševljevog polinoma prve vrste $\hat{T}_{n+1}$.

Kako Čebiševljev polinom prve vrste $\hat{T}_n$ ima n nula, to se ove nule mogu iskoristiti za interpolacine čvorove interpolacionog problema sa n tačaka. S obzirom na to da su interpolacioni čvorovi numerisani počev od nule, a nule polinoma su numerisane počev od jedinice, ovo može da predstavlja mali problem. Zato dajemo eksplicitan izraz za $n+1$ interpolacionih čvorova u Čebiševljevim tačkama

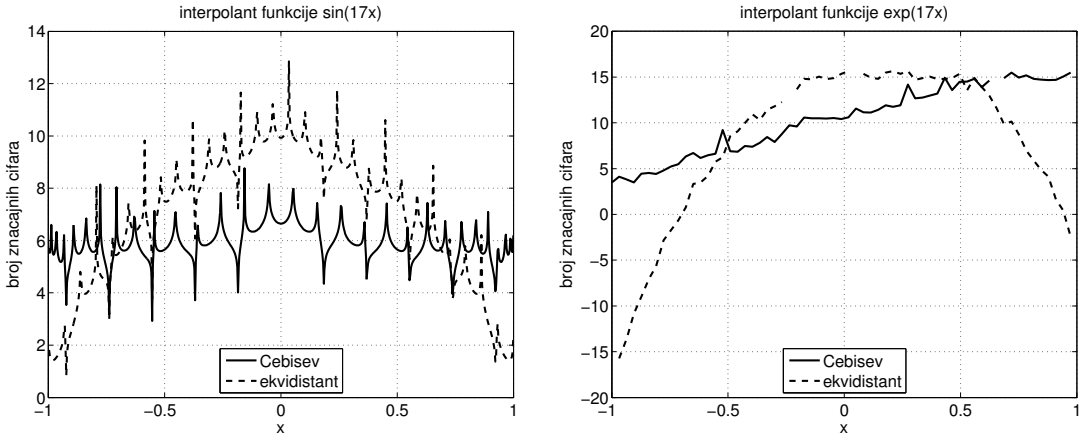
$$x_i = -\cos\frac{(2(i+1)-1)\pi}{2(n+1)} = -\cos\frac{(2i+1)\pi}{2(n+1)}, \quad i=0,\ldots,n. \tag{4.3}$$

Ako su interpolacioni čvorovi x_i, $i = 0, \ldots, n$, nule Čebiševljevog polinoma stepena $n+1$, izraz za grešku interpolacije dat u teoremi 4.4, dobija sledeći oblik

$$\max_{x\in[-1,1]} |f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \max_{x\in[-1,1]} |\hat{T}_{n+1}(x)| = \frac{M_{n+1}}{2^n(n+1)!}.$$

Slika 4.4 prikazuje grafike čvornih polinoma kada su interpolacioni čvorovi zadati u nulama Čebiševljevog polinoma prve vrste i kada su zadati ekvidistantno na intervalu $[-1, 1]$ za trideset interpolacionih čvorova. Uravnotežen grafik odgovara izboru interpolacionih čvorova u Čebiševljevim tačkama. Vidimo da maksimum u slučaju interpolacionih čvorova u Čebiševljevim tačkama ima red veličine $10^{-8.7}$, dok je maksimum u slučaju ekvidistantnih čvorova 10^{-5}. U pitanju je četiri reda veličine. Ekvidistantni čvorovi postižu bolje rezultate u unutrašnjosti intervala $[-1, 1]$, ali je cena na krajevima intervala previsoka.

Radi ilustracije, navedimo šta prethodna razlika znači u praksi. Pretpostavimo da hoćemo da interpoliramo funkciju $\sin(17x)$ na intervalu $[-1, 1]$ sa trideset interpolacionih čvorova. Slika 4.5, levo, ilustruje broj značajnih cifara interpolacionih polinoma konstruisanih u Čebiševljevim čvorovima i ekvidistantnim čvorovima.



Slika 4.5: Slika levo ilustruje broj značajnih cifara pri interpolaciji funkcije $\sin(17x)$ na intervalu $[-1, 1]$ u Čebiševljevim i ekvidistantnim interpolacionim čvorovima, u 30 interpolacionih čvorova, slika desno ilustruje broj značajnih cifara pri interpolaciji funkcije $\exp(17x)$ na intervalu $[-1, 1]$ u Čebiševljevim i ekvidistantnim interpolacionim čvorovima, u 100 interpolacionih čvorova.

Primetimo prvo da je drugi sabirak u nejednakosti (4.2) zanemarljiv, jer je ε jednako mašinskoj preciznosti. Iz ovog razloga posmatramo samo uticaj prvog sabirka, u ovom slučaju sabirka sa čvornim polinomom, čija se vrednost može proceniti na

$$\frac{17^{30}}{30!} \max_{\psi\in[-1,1]} |\cos(17\psi)|\,|\omega(x)| \leq 10^6 |\omega(x)|.$$

Gornja granica relativne greške iznosi $10^6|\omega(x)|/|\sin(17x)|$. Kao što vidimo sa slike broj značajnih cifara, pri interpolaciji u Čebiševljevim čvorovima je distribuiran ravnomerno po celom intervalu

$[-1,1]$. Preciznije, na izvestan način, ako zanemarimo varijacije funkcije $\sin(17x)$ na intervalu $[-1,1]$, možemo reći da je grafik broja značajnih cifara samo translirana verzija grafika funkcije $-\log_{10}|\omega(x)|$, gde je čvorni polinom konstruisan za Čebiševljeve čvorove. Slična je situacija i za ekvidistantne čvorove, grafik broja značajnih cifara je do na varijaciju funkcije $\sin(17x)$, samo translirana verzija grafika funkcije $-\log_{10}|\omega(x)|$, gde je čvorni polinom konstruisan u ekvidistantnim čvorovima. Pod varijacijama funkcije $\sin(17x)$ razumemo pikove koji se javljaju na graficima, a koji su posledica promena vrednosti funkcije na intervalu $[-1,1]$.

Slika 4.5, desno, ilustruje broj značajnih cifara interpolacionih polinoma pri interpolaciji funkcije e^{17x}, na intervalu $[-1,1]$, ali ovoga puta sa 100 interpolacionih čvorova.

Generalno, možemo zaključiti da Čebiševljevi interpolacioni čvorovi obezbeđuju uniformnost interpolacione greške na celom intervalu interpolacije, dok, na primer, ekvidistantni interpolacioni čvorovi postižu bolje rezultate na sredini intervala interpolacije, ali znatno gore na krajevima intervala interpolacije.

Primetimo da grafik 4.5, desno, sadrži pored dela gde je dominantan prvi izraz u interpolacionoj grešci i deo gde je dominantan deo vezan za Lebesgueovu funkciju. Ovaj deo je u blizini tačke jedan na grafiku funkcije. O uticaju ovog člana govorimo nadalje. Primetimo da ako je broj interpolacionih čvorova dovoljno veliki, onda je član vezan za Lebesgueovu funkciju dominantan, jer je faktorijel u prvom članu uglavnom dovoljno veliki da potisne doprinos prvog sabirka na levoj strani nejednakosti (4.2).

Minimizacija Lebesgueove funkcije

Drugi problem optimalnog izbora interpolacionih čvorova se odnosi na takav izbora interpolacionih čvorova koji minimizira maksimum Lebesgueove funkcije. Ovakav izbor čvorova je bitan kad se konstruiše interpolacioni polinom visokog stepena, jer je u tom slučaju drugi član u nejednakosti (4.2) dominantan. Rešenje ovog problema takođe dajemo za normalizovan interval $[-1,1]$. Rešenje za generalni interval $[a,b]$ može se pronaći linearnom transformacijom promenljive $a+(b-a)(x+1)/2$.

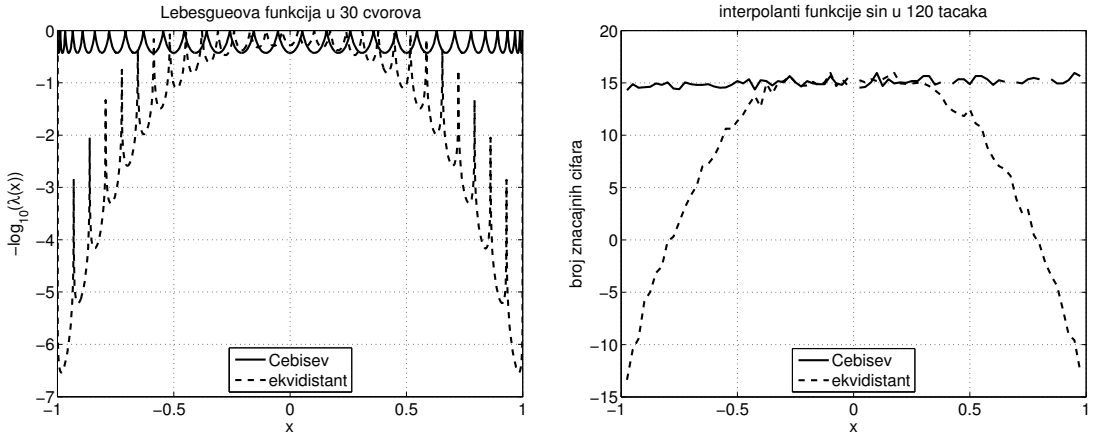
Ovaj problem još uvek nema zadovoljavajuće rešenje, ali se dobri rezultati mogu postići koristeći nule Čebiševljevih polinoma prve vrste. Međutim, još bolje rešenje se postiže kada interpolacione čvorove postavimo na sledeći način

$$x_i = -\frac{\cos\dfrac{(2i+1)\pi}{2(n+1)}}{\cos\dfrac{\pi}{2(n+1)}}, \quad i=0,\ldots,n. \tag{4.4}$$

Ovo su nule Čebiševljevih polinoma prve vrste $\hat{T}_{n+1}$ koje su linearno razmeštene, tako da nula krajnje levo i nula krajnje desno padaju u tačke -1 i 1, redom.

Slika 4.6, levo, prikazuje grafike Lebesgueovih funkcija za interpolacione čvorove postavljene u tačkama (4.4) i za ekvidistantne interpolacione čvorove. Broj interpolacionih čvorova iznosi 30. Maksimum Lebesgueove funkcije za izbor interpolacionih čvorova vezan za nule Čebiševljevih polinoma ima maksimum 2.7, dok sa interpolacionim čvorovima u ekvidistantnim tačkama ovaj maksimum ima red veličine 10^6. Interpolacioni polinomi konstruisani u jednim i drugim interpolacionim čvorovima bi se bitno razlikovali. Polinom konstruisan u interpolacionim čvorovima koje su vezane za Čebiševljeve tačke bi imao mnogo manju grešku za istu tačnost podataka.

Praktičnu ilustraciju pomenute razlike možemo dobiti jednostavno. Konstruišimo interpolacione polinome za funkciju sin na intervalu $[-1,1]$, sa interpolacionim čvorovima koji su postavljeni u tačkama (4.4) i u ekvidistantnim tačkama. Slika 4.6, desno, ilustruje broj značajnih



Slika 4.6: Slika levo prikazuje grafik negativnog logaritma za osnovu deset Lebesgueove funkcije sa interpolacionim čvorovima u Čebiševljevim tačkama (4.4) i ekvidistantnim tačkama, sa 30 interpolacionih čvorova, slika desno prikazuje značajne cifre interpolacionih polinoma u Čebiševljevim tačkama (4.4) i ekvidistantnim tačkama funkcije sin, sa 120 interpolacionih čvorova.

cifara u interpolantima koji su konstruisani u 120 tačaka. Vidimo da je interpolant konstruisan Čebiševljevim tačkama (4.4) sa gornjom granicom relativne greške 10^{-15} uniformno na intervalu $[-1, 1]$. Međutim, interpolant konstruisan u ekvidistantnim tačkama ima gornju granicu relativne greške koja nije uniformno raspodeljena na intervalu $[-1, 1]$ i vidimo da je na krajevima intervala ova greška jako velika. Primetimo da smo ocenu mogli dobiti prostom translacijom grafika $-\log_{10}\lambda(x)$ za vrednost $-\log_{10}10^{-16} = 16$, kolika je relativna greška podataka u dvostrukoj preciznosti.

4.1.4 O rešenju problema interpolacije matričnim metodima

Moguće je egzistenciju algebarskog interpolanta dokazati rešavanjem sistema linearnih jednačina. Sledeća teorema koristi metode linearne algebre da utvrdi postojanje algebarskog interpolanta.

Teorema 4.6 *Neka su x_i, $i = 0, \ldots, n$, interpolacioni čvorovi i neka su $f(x_i)$, $i = 0, \ldots, n$, vrednosti funkcije f u ovim interpolacionim čvorovima. Postoji tačno jedan polinom P_n, stepena ne većeg od n, koji zadovoljava interpolacione uslove*

$$P_n(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Dokaz. Za dokaz tvrđenja, pretpostavimo da je polinom P_n zadat na sledeći način $P_n(x) = \sum_{k=0}^{n} p_k x^k$. Interpolacioni uslovi se mogu izraziti na sledeći način

$$f(x_i) = P_n(x_i) = p_0 + p_1 x_i + \cdots + p_n x_i^n, \quad i = 0, \ldots, n.$$

Ovo je sistem linearnih jednačina po nepoznatim koeficijentima polinoma p reda $n+1$. Matrica i slobodni vektor sistema linearnih jednačina su određeni sa

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}, \quad b = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

Matrica sistema je poznata u literaturi kao Vandermondova matrica. Vrednost deteminante Vandermondove matrice, koja je poznata kao Vandermondova determinanta, iznosi

$$\det(V) = \prod_{i,j=0, i>j}^{n} (x_i - x_j) \neq 0.$$

Vrednost determinante nije jednaka nuli, jer su interpolacioni čvorovi međusobno različiti, a kako vidimo u proizvodu učestvuju samo članovi $x_i - x_j$ za $i > j$.

Prethodna činjenica znači da je matrica sistema V regularna, pa sistem ima jedinstveno rešenje za svaki slobodan vektor b. Kako je rešenje sistema linearnih jednačina vektor ispunjen koeficijentima polinoma P_n, zaključujemo da za svaki skup interpolacionih čvorova x_i, $i = 0, \dots, n$ i svaki skup vrednosti funkcije $f(x_i)$, $i = 1, \dots, n$, postoji jedinstven polinom P_n, ne većeg stepena od n, koji zadovoljava interpolacione uslove $P_n(x_i) = f(x_i)$, $i = 0, \dots, n$. □

Numerička konstrukcija interpolanta korišćenjem Vandermondovih matrica

Na osnovu teoreme 4.6 moguće je izvršiti konstrukciju interpolanta rešavanjem sistema linearnih jednačina. U ovom odeljku mi se bavimo mogućnostima konstrukcije interpolanta rešavanjem sistema linearnih jednačina.

Funkcija `vander`, `Matlab`a, konstruiše Vandermondove matrice. Prihvata jedan ulazni argument, koji predstavlja vektor interpolacionih čvorova tipa $1 \times n$ ili $n \times 1$. Funkcija vraća Vandermondovu matricu konstruisanu nad zadatim vektorom u formi

$$V = \begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & 1 \\ x_2^n & x_2^{n-1} & x_2^{n-2} & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & 1 \end{bmatrix}.$$

Konstrukcija koeficijenata interpolacionog polinoma se može izvršiti rešavanjem sistema linearnih jednačina. Primetimo da zbog forme Vandermondove matrice koju vraća funkcija `vander`, rešenje sistema linearnih jednačina sadrži koeficijente polinoma onako kako su polinomi reprezentovani u `Matlab`u. Prvi element vektora rešenja je koeficijent uz najviši stepen promenljive, dok je poslednji element vektora koeficijent polinoma koji stoji uz nulti stepen promenljive.

Na primer, za skup podataka $\{(0,1),(1,2),(2,3),(3,4)\}$, konstrukcija polinoma se može izvršiti na sledeći način

```
>>V=vander([0 1 2 3])
V =
```

```
     0     0     0     1
     1     1     1     1
     8     4     2     1
    27     9     3     1
>>b=[1; 2; 3; 4];
>>p=linsolve(V,b)
p =
     0
     0
     1
     1
```

Iako najjednostavniji za upotrebu ovaj način konstrukcije je daleko od najboljeg. Problem je veliki faktor uslovljenosti Vandermondovih matrica. Da bismo ilustrovali ovaj problem na slici 4.3, desno, je prikazan logaritam, za osnovu deset, faktora uslovljenosti Vandermondovih matrica u funkciji reda matrice. Interpolacioni čvorovi su generisani slučajno u intervalu $(0, 15)$. Kao što vidimo zavisnost logaritma faktora uslovljenosti je skoro linearna u funkciji reda matrice. Znači da faktor uslovljenosti Vandermondove matrice raste eksponencijalno sa porastom reda matrice. Drugim rečima, već za interpolant u deset tačaka možemo očekivati da su koeficijenti polinoma zadati sa vrlo malim brojem značajnih cifara.

Sledeći primer ilustruje grešku koja se dobija primenom ovog algoritma u odnosu na konstrukciju korišćenjem Lagrangeovog bazisa.

```
>>c=12*rand(1,12);
>>V=vander(c);
>>b=(1:12)';
>>p=linsolve(V,b);
>>polyval(p,.5)
ans =
  -1.327231003399312e+06
>>lagrangePoly(c,b,.5)
ans =
  -1.327231213365709e+06
>>syms t clear;
>> pp=lagrangePoly(c,b,t);
>> subs(pp,1/2)
ans =
  -1.327231213365708e+06
```

Kao što vidimo konstrukcija Vandermondovom matricom ima šest značajnih cifara, dok konstrukcija Lagrangeovim bazisom ima petnaest, ako ne i svih šesnaest, značajnih cifara.

Problem postaje posebno značajan ako su vrednosti funkcije zadate sa relativnom greškom koja je znatno veća od mašinske preciznosti. Ako je relativna greška ulaznih podataka 10^{-3} i konstruišemo polinom stepena 12, može se desiti da nećemo imati nijednu značajnu cifru u konstruisanom interpolantu.

Zbog ovakvog ponašanja faktora uslovljenosti Vandermondovih matrica, ovaj način konstrukcije interpolanta se u praksi nikada ne koristi.

4.2 Newtonov interpolacioni polinom

Problem sa interpolacionim polinomom, koji je predstavljen u Lagrangeovom obliku, se sastoji u tome što je nemoguće iskoristiti konstruisani polinom stepena n za konstrukciju polinoma stepena $n+1$. Ovaj problem može se rešiti upotrebom Newtonovog interpolacionog polinoma. Napominjemo da su Newtonov i Lagrangeov interpolacioni polinom isti za iste skupove podataka, ako zanemarimo efekte izračunavanja u dvostrukoj preciznosti.

Za zadati skup interpolacionih čvorova x_i, $i = 0, \ldots, n$, označimo sa

$$\omega_k(x) = \prod_{i=0}^{k-1}(x - x_i), \quad k \in \mathbb{N},$$

gde po definiciji uzimamo $\omega_0(x) = 1$, čvorne polinome za skup interpolacionih čvorova. Primetimo da čvorni polinomi zadovoljavaju jednakost

$$\omega_k(x_j) = \prod_{i=0}^{k-1}(x_j - x_i) = 0,$$

za $k > j$, jer u proizvodu postoji faktor $x_j - x_j$ kada je $k > j$.

Ideja Newtona je da se interpolacioni polinom zapiše u sledećem obliku

$$\begin{aligned} P_n(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1}) \\ &= \sum_{j=0}^{n} c_j \omega_j(x). \end{aligned}$$

Očigledno je da polinom ima željenu zavisnost što se tiče zavisnosti po monomima $x - x_i$, $i = 0, \ldots, n$. Ostaje još samo da utvrdimo kako treba birati koeficijente c_i, $i = 0, \ldots, n$, da bismo dobili algebarski interpolant. Primetimo da važi rekurentna relacija

$$P_n(x) = P_{n-1}(x) + c_n\omega_n(x).$$

Polinom P_n mora da zadovolji interpolacione uslove. Odavde dobijamo

$$f(x_k) = P_n(x_k) = \sum_{j=0}^{n} c_j\omega_j(x_k) = \sum_{j=0}^{k} c_j\omega_j(x_k) = P_k(x_k).$$

Na osnovu ovoga možemo odrediti koeficijente c_k na sledeći način

$$c_k = \frac{f(x_k) - \sum_{j=0}^{k-1} c_j\omega_j(x_k)}{\omega_k(x_k)} = \frac{f(x_k) - P_{k-1}(x_k)}{\omega_k(x_k)}, \quad k = 0, \ldots, n.$$

U ovoj jednačini po dogovoru uzimamo $P_{-1} = 0$.

Očigledno koeficijent c_k zavisi samo od vrednosti funkcije f u tačkama $x_0, \ldots, x_k$. Da bismo ovu činjenicu istakli usvajamo sledeće označavanje

$$c_k = [x_0, \ldots, x_k; f], \quad k = 0, \ldots, n.$$

Očigledno je $c_0 = [x_0; f] = f(x_0)$ i

$$c_1 = \frac{f(x_1) - P_0(x_1)}{\omega_1(x_1)} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Zbog poslednjeg izraza simboli $[x_0, \ldots, x_k; f]$, $k = 0, \ldots, n$, su dobili naziv podeljene razlike.

Broj interpolacionih čvorova umanjen za jedan u simbolu $[x_0, \ldots, x_k; f]$ naziva se redom podeljene razlike. Tako simbol $[x_0; f]$ ima red nula, simbol $[x_0, x_1; f]$ ima red jedan i slično.

Definicija 4.2 *Neka su x_i, $i = 0, \ldots, n$ interpolacioni čvorovi i neka je funkcija f zadata svojim vrednostima $f(x_i)$, $i = 0, \ldots, n$ u ovim interpolacionim čvorovima. Polinom P_n određen sa*

$$P_n(x) = \sum_{k=0}^{n} [x_0, \ldots, x_k; f]\omega_k(x),$$

gde je

$$[x_0, \ldots, x_k; f] = \frac{f(x_k) - P_{k-1}(x_k)}{\omega_k(x_k)}, \quad k = 0, \ldots, n,$$

naziva se Newtonov interpolacioni polinom.

Teorema 4.7 (Newtonov interpolacioni polinom) *Newtonov interpolacioni polinom zadovoljava interpolacione uslove*

$$P_n(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Drugim rečima, Newtonov interpolacioni polinom je rešenje problema algebarske interpolacije.

Dokaz. Direktnom proverom nalazimo

$$\begin{aligned} P_n(x_i) &= \sum_{k=0}^{n} [x_0, \ldots, x_k; f]\omega_k(x_i) = [x_0, \ldots, x_i; f]\omega_i(x_i) + \sum_{k=0}^{i-1} [x_0, \ldots, x_k; f]\omega_k(x_i) \\ &= \frac{f(x_i) - P_{i-1}(x_i)}{\omega_i(x_i)}\omega_i(x_i) + P_{i-1}(x_i) = f(x_i). \end{aligned}$$

Kako je polinom P_n ne većeg stepena od n, što je stepen čvornog polinoma ω_n, i kako polinom P_n zadovoljava interpolacione uslove, na osnovu jedinstvenosti algebarskog interpolanta zaključujemo da je polinom P_n algebarski interpolacioni polinom. □

Sledeća teorema daje informaciju o načinu na koji je moguće računati podeljene razlike.

Teorema 4.8 *Podeljene razlike zadovoljavaju sledeću rekurentnu relaciju*

$$[x_0, \ldots, x_k; f] = \frac{[x_1, \ldots, x_k; f] - [x_0, \ldots, x_{k-1}; f]}{x_k - x_0}.$$

Dokaz. Označimo sa Q_k, $k = 1, \ldots, n$, niz interpolacionih polinoma, stepena ne većeg od k, koji zadovoljavaju interpolacione uslove

$$Q_k(x_i) = f(x_i), \quad i = 0, \ldots, k.$$

Neka je P interpolacioni polinom, stepena ne većeg od $n-1$, koji zadovoljava interpolacione uslove

$$P(x_i) = f(x_i), \quad i = 1, \ldots, n.$$

Posmatrajmo polinom

$$R(x) = P(x) + \frac{x - x_n}{x_n - x_0}(P(x) - Q_{n-1}(x)).$$

Direktnim izračunavanjem za $i = 1, \ldots, n-1$, nalazimo

$$\begin{aligned} R(x_i) &= P(x_i) + \frac{x_i - x_n}{x_n - x_0}(P(x_i) - Q_{n-1}(x_i)) \\ &= f(x_i) + \frac{x_i - x_n}{x_n - x_0}(f(x_i) - f(x_i)) = f(x_i). \end{aligned}$$

Za $i = 0$ važi

$$R(x_0) = P(x_0) + \frac{x_0 - x_n}{x_n - x_0}(P(x_0) - Q_{n-1}(x_0)) = P(x_0) - P(x_0) + f(x_0) = f(x_0).$$

Slično, za $i = n$ važi

$$R(x_n) = P(x_n) + \frac{x_n - x_n}{x_n - x_0}(P(x_n) - Q_{n-1}(x_n)) = f(x_n).$$

Zaključujemo da je $R = Q_n$, ili

$$Q_n(x) = P(x) + \frac{x - x_n}{x_n - x_0}(P(x) - Q_{n-1}(x)).$$

Vodeći koeficijent u polinomu Q_n, prema teoremi 4.7, je podeljena razlika $[x_0, \ldots, x_n; f]$, dok je vodeći koeficijent u polinomu

$$P(x) + \frac{x - x_n}{x_n - x_0}(P(x) - Q_{n-1}(x))$$

izraz

$$\frac{[x_1, \ldots, x_n; f] - [x_0, \ldots, x_{n-1}; f]}{x_n - x_0}.$$

Ova dva koeficijenta moraju biti jednaka, pa mora da važi

$$[x_0, \ldots, x_n; f] = \frac{[x_1, \ldots, x_n; f] - [x_0, \ldots, x_{n-1}; f]}{x_n - x_0}.$$

Ovo je i trebalo dokazati. □

Ova teorema omogućava rekurzivno izračunavanje podeljenih razlika. Podeljene razlike nultog reda su vrednosti funkcije, podeljene razlike prvog reda se računaju koristeći podeljene razlike nultog reda. Podeljenje razlike reda k se računaju koristeći podeljene razlike reda $k-1$. Najčešće se podeljene razlike prikazuju u obliku tablice podeljenih razlika na sledeći način.

$$\begin{array}{lllll} x_0 & \underline{[x_0; f]} & & & \\ & & \underline{[x_0, x_1; f]} & & \\ x_1 & [x_1; f] & & \underline{[x_0, x_1, x_2; f]} & \\ & & [x_1, x_2; f] & & \underline{[x_0, x_1, x_2, x_3; f]} \\ x_2 & [x_2; f] & & [x_1, x_2, x_3; f] & \\ & & [x_2, x_3; f] & & \\ x_3 & [x_3; f] & & & \\ \vdots & \vdots & & & \end{array}$$

Primetimo da dodavanje nove tačke menja tabelu podeljenih razlika samo u smislu dodavanja nove dijagonale. Deo tabele koji je već konstruisan postaje deo nove, šire tabele. Podeljene razlike u tabeli koje su podvučene, su potrebne prilikom konstrukcije Newtonovog interpolacionog polinoma.

Koristeći skup podataka $\{(0,1),(1,2),(2,3),(3,4)\}$ formirajmo Newtonov interpolacioni polinom. Postupak je jednostavan. Dovoljno je formirati tabelu podeljenih razlika. Dobijamo

$$\begin{array}{cccccc}
k & x_k & [x_k;f] & [x_k,x_{k+1};f] & [x_k,x_{k+1},x_{k+2};f] & [x_k,x_{k+1},x_{k+2},x_{k+3};f] \\
0 & 0 & \underline{1} & & & \\
 & & & \dfrac{2-1}{1-0}=\underline{1} & & \\
1 & 1 & 2 & & \dfrac{1-1}{2-0}=\underline{0} & \\
 & & & \dfrac{3-2}{2-1}=1 & & \dfrac{0-0}{3-0}=\overline{\underline{0}} \\
2 & 2 & 3 & & \dfrac{1-1}{3-1}=\overline{0} & \\
 & & & \dfrac{4-3}{3-2}=\overline{1} & & \\
3 & 3 & \overline{4} & & &
\end{array}$$

Newtonov interpolacioni polinom za zadati skup podataka, koristeći podatke iz tabele koji su podvučeni, konstruišemo jednostavno

$$\begin{aligned}
P_3(x) &= [x_0;f]\omega_0(x)+[x_0,x_1;f]\omega_1(x)+[x_0,x_1,x_2;f]\omega_2(x)+[x_0,x_1,x_2,x_3;f]\omega_3(x) \\
&= 1+1\,(x-0)+0\,(x-0)(x-1)+0\,(x-0)(x-1)(x-2) \\
&= 1+x.
\end{aligned}$$

Kao što vidimo, a kako smo i očekivali na osnovu teoreme 4.7, dobijeni interpolacioni polinom se podudara sa interpolacionim polinomom dobijenim primenom Lagrangeove formule u sekciji 4.1 za pomenuti skup podataka.

Primetićemo, takođe, da nas ništa prilikom dokaza teoreme 4.7 ne obavezuje da koristimo uređenje interpolacionih čvorova. Teorema i rezultat ostaju u važnosti i za interpolacione čvorove koji nisu uređeni u rastućem poretku. Da bismo ilustrovali ovu činjenicu možemo izvršiti renumeraciju čvorova u prethodnom primeru. Na primer, možemo izabrati $x_0=3$, $x_1=2$, $x_2=1$ i $x_3=0$. Za ovako izabrane interpolacione čvorove, tabela konačnih razlika se dobija iz prethodne tabele ogledanjem oko horizontalne linije koja prolazi kroz sredinu tabele. Naravno, nije potrebno konstruisati novu tabelu, dovoljno je samo imati ovu osobinu na umu. Sa ovakvom renumeracijom interpolacionih čvorova, dobijamo sledeći interpolacioni polinom

$$\begin{aligned}
P_3(x) &= [x_0;f]\omega_0(x)+[x_0,x_1;f]\omega_1(x)+[x_0,x_1,x_2;f]\omega_2(x)+[x_0,x_1,x_2,x_3;f]\omega_3(x) \\
&= 4+1\,(x-3)+0\,(x-3)(x-2)+0\,(x-3)(x-2)(x-1) \\
&= 1+x.
\end{aligned}$$

Rezultat je isti, jer je interpolacioni polinom jedinstven. Vrednosti iz tabele koje smo koristili prilikom ove konstrukcije su nadvučene.

Korišćenje iste tabele podeljenih razlika je moguće, jer podeljene razlike zadovoljavaju sledeću osobinu.

Lema 4.3 *Neka su x_i, $i=0,\ldots,n$, interpolacioni čvorovi i $f(x_i)$, $i=0,\ldots,n$, vrednosti funkcije u ovim interpolacionim čvorovima. Za svako $k\in\{0,\ldots,n\}$ i svako $\ell=0,\ldots,n-k$, važi*

$$[x_k,\ldots,x_{k+\ell};f]=[x_{k+\ell},\ldots,x_k;f].$$

Dokaz. Dokaz je najjednostavnije dati indukcijom po ℓ. Formula je očigledno tačna za $\ell = 0$, jer je

$$[x_k; f] = [x_k; f].$$

Neka je formula tačna za $\ell = m$. Na osnovu teoreme 4.8, važi

$$\begin{aligned}
[x_k, \ldots, x_{k+m+1}; f] &= \frac{[x_{k+1}, \ldots, x_{k+m+1}; f] - [x_k, \ldots, x_{k+m}; f]}{x_{k+m+1} - x_k} \\
&= \frac{[x_{k+m+1}, \ldots, x_{k+1}; f] - [x_{k+m}, \ldots, x_k; f]}{x_{k+m+1} - x_k} \\
&= \frac{[x_{k+m}, \ldots, x_k; f] - [x_{k+m+1}, \ldots, x_{k+1}; f]}{x_k - x_{k+m+1}} \\
&= [x_{k+m+1}, \ldots, x_k; f].
\end{aligned}$$

Ovim je dokaz leme završen. □

Na osnovu ove leme moguće je istu tabelu podeljenih razlika shvatiti kao tabelu konstruisanu za čvorove sa obrnutom numeracijom.

Oblik Newtonovog interpolacionog polinoma omogućava sledeće elementarno zapažanje.

Lema 4.4 *Neka je p polinom ne većeg stepena od k. Za svakih $n+1$ različitih tačaka x_i, $i = 0, \ldots, n$, gde je $n > k$, važi*

$$[x_0, \ldots, x_n; p] = 0.$$

Dokaz. Izaberimo interpolacione čvorove x_i, $i = 0, \ldots, n$, i konstruišimo Newtonov interpolacioni polinom koji zadovoljava interpolacione uslove

$$P_n(x_i) = p(x_i), \quad i = 0, \ldots, n.$$

Kako je broj interpolacionih čvorova veći od stepena polinoma p, na osnovu leme 4.2, mora biti $P_n = p$. Međutim,

$$p(x) = P_n(x) = \sum_{k=0}^{n} [x_0, \ldots, x_k; p]\omega_k(x).$$

Ako je $[x_0, \ldots, x_n; p] \neq 0$, onda je stepen polinoma na desnoj strani jednak n, dok je stepen polinoma na levoj strani k, i prema uslovu teoreme je $n > k$. Ovo je kontradikcija, pa mora biti $[x_0, \ldots, x_n; p] = 0$. □

4.2.1 Numerička konstrukcija Newtonovog interpolacionog polinoma

Pri implementaciji numeričkih izračunavanja sa podeljenim razlikama u `Matlab`u od značaja je funkcija `diff`.

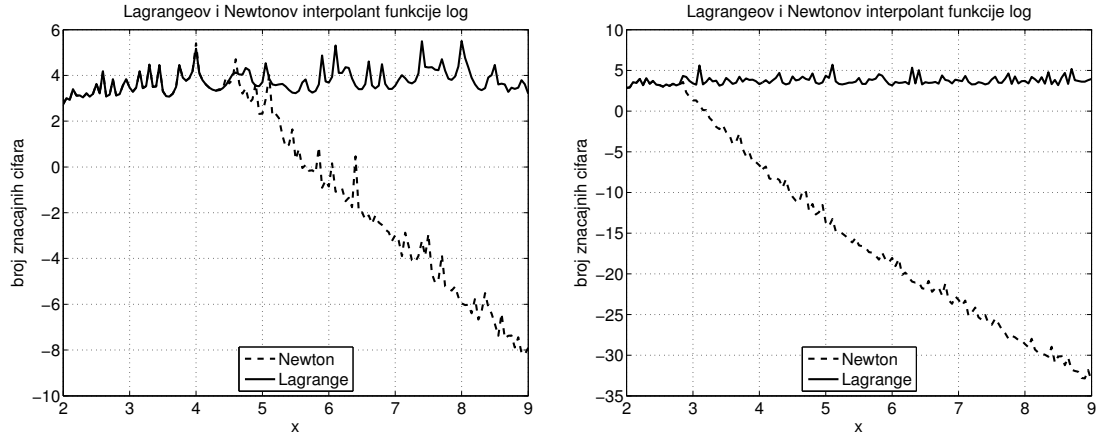
Funkcija `diff(y)` izračunava razlike susednih elemenata vektora `y`, rezultat izvršenja je vektor `[y(2)-y(1) y(3)-y(2) ... y(end)-y(end-1)]`. Sledeći primer jasno ilustruje koncept

```
>>y=(1:10).^2
y =
     1     4     9    16    25    36    49    64    81   100
>>diff(y)
ans =
     3     5     7     9    11    13    15    17    19
```

Koristeći funkciju `diff`, funkcija koja implementira konstrukciju Newtonovog interpolacionog polinoma može se dati na sledeći način.

```
function poly=newtonPoly(cvorovi,vrednosti,x)
%NEWTONPOLY(CVOROVI,VREDNOSTI,X) izracunava vrednosti Newtonovog
% interpolacionog polinoma u cvorovima CVOROVI sa vrednostima funkcije
% VREDNOSTI u argumentu X koji moze biti vektor kolona

n = length(vrednosti)-1;
podRazlike = vrednosti;
poly = podRazlike(1);
podRazlike=diff(podRazlike);
omega = 1; k = 1;
while not(isempty(podRazlike))
      omega = omega.*(x-cvorovi(k));
      for i=1:length(podRazlike)
            podRazlike(i) = podRazlike(i)/(cvorovi(i+k)-cvorovi(i));
      end
      poly = poly + podRazlike(1).*omega;
      podRazlike=diff(podRazlike);
      k = k+1;
end
end
```

Konstrukcija algebarskog interpolanta uz pomoć Newtonovog interpolacionog polinoma može da bude znatno osetljivija na greške ulaznih podataka od iste konstrukcije izvedene korišćenjem Lagrangeovog interpolacionog polinoma. Slika 4.7 ilustruje ovu činjenicu. Levi grafik prikazuje broj



Slika 4.7: Slika levo prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta u 60 Čebiševljevih tačaka funkcije log na intervalu $[2, 9]$, slika desno prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta u 110 Čebiševljevih tačaka funkcije log na intervalu $[2, 9]$. Oba grafika su dobijena sa vrednostima funkcije sa gornjom granicom relativne greške 10^{-3}.

značajnih cifara pri interpolaciji funkcije log na intervalu $[2, 9]$ sa šezdeset Čebiševljevih čvorova

Newtonovim i Lagrangeovim interpolacionim polinomom, desni grafik prikazuje istu konstrukciju ali u 110 interpolacionih čvorova. Može se primetiti da je kvalitet Newtonove interpolacije lošiji u odnosu na Lagrangeovu. Pri crtanju oba grafika podrazumevali smo da su vrednosti funkcije zadate sa gornjom granicom relativne greške 10^{-3}.

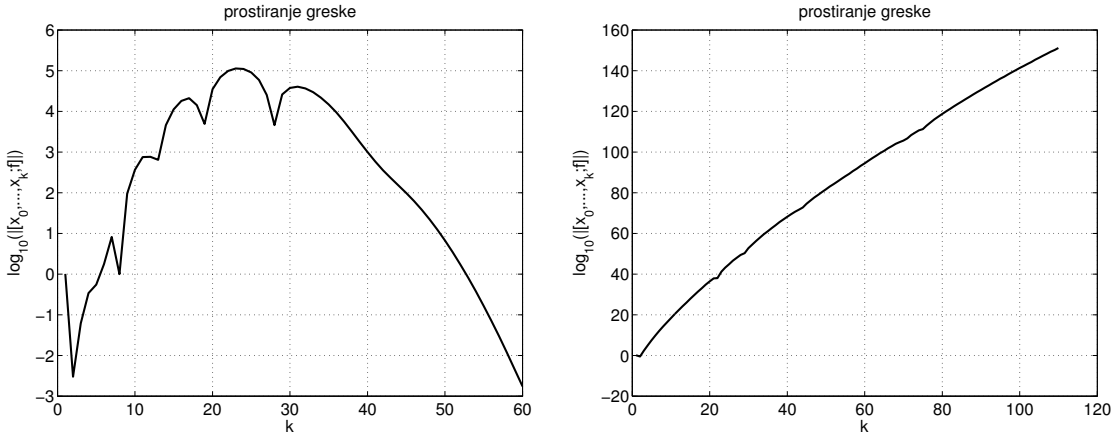
Da bismo objasnili razloge za ovakvo ponašanje Newtonovog interpolacionog polinoma, moramo se pozabaviti prostiranjem greške ulaznih podataka prilikom konstrukcije Newtonovog interpolacionog polinoma. Pretpostavimo da smo odredili podeljene razlike reda $k-1$, $[x_0,\ldots,x_{k-1};f]$ i $[x_1,\ldots,x_k;f]$, sa greškama $\varepsilon_{0,\ldots,k-1}$i $\varepsilon_{1,\ldots,k}$, redom. Odredimo grešku $\varepsilon_{0,\ldots,k}$ u podeljenoj razlici reda k. Imamo

$$\begin{aligned}[x_0,\ldots,x_k;f]+\varepsilon_{0,\ldots,k} &= \frac{[x_1,\ldots,x_k;f]+\varepsilon_{1,\ldots,k}-[x_0,\ldots,x_{k-1};f]-\varepsilon_{0,\ldots,k-1}}{x_k-x_0}\\ &= [x_0,\ldots,x_k;f]+\frac{\varepsilon_{1,\ldots,k}-\varepsilon_{0,\ldots,k-1}}{x_k-x_0},\end{aligned}$$

odakle zaključujemo da važi

$$\varepsilon_{0,\ldots,k}=\frac{\varepsilon_{1,\ldots,k}-\varepsilon_{0,\ldots,k-1}}{x_k-x_0}.$$

Drugim rečima, greške podeljenih razlika se prenose poštujući ista pravila izračunavanja kao i same podeljene razlike.



Slika 4.8: Slika levo prikazuje grafik funkcije $\log_{10}|[x_0,\ldots,x_k;f]|$, $k=0,\ldots,59$, podeljenih razlika funkcije $f(x)=1$ u ekvidistantnim tačkama na intervalu $[2,9]$ pri čemu vrednosti funkcije imaju gornju granicu apsolutne greške 10^{-3}, slika desno ilustruje grafik iste funkcije ali za 110 ekvidistantnih interpolacionih čvorova zadatih na intervalu $[0,.2]$.

Da stvari sa prostiranjem greške mogu postati zaista komplikovane, može nam ilustrovati sledeći primer. Posmatrajmo prostiranje greške prilikom konstrukcije podeljenih razlika funkcije $f(x)=1$, pri čemu su podaci kojima je zadata funkcija zadati sa gornjom granicom apsolutne greške 10^{-3}. Neka su interpolacioni čvorovi zadati ekvidistantno na intervalu $[2,7]$ i neka ih ima šezdeset. Sledeći skript ilustruje konstrukciju podeljenih razlika u ovom slučaju i generiše grafik koji je prikazan na slici 4.8, levo.

```
n=59;
```

```
c=linspace(2,7,n+1);
podRazlike=[1+(2*rand(n+1,1)-1)*10^(-3)];
podRazlike0=[podRazlike(1)];
for k=1:n
    cD=[];
    for i=1:n+1-k
        cD=[cD; c(i+k)-c(i)];
    end
    podRazlike=diff(podRazlike)./cD;
    podRazlike0=[podRazlike0 podRazlike(1)];
end
plot(log10(abs(podRazlike0))');
title('prostiranje greske');
ylabel('log_{10}(|[x_0,...,x_k;f]|)');
xlabel('k');
grid on;
```

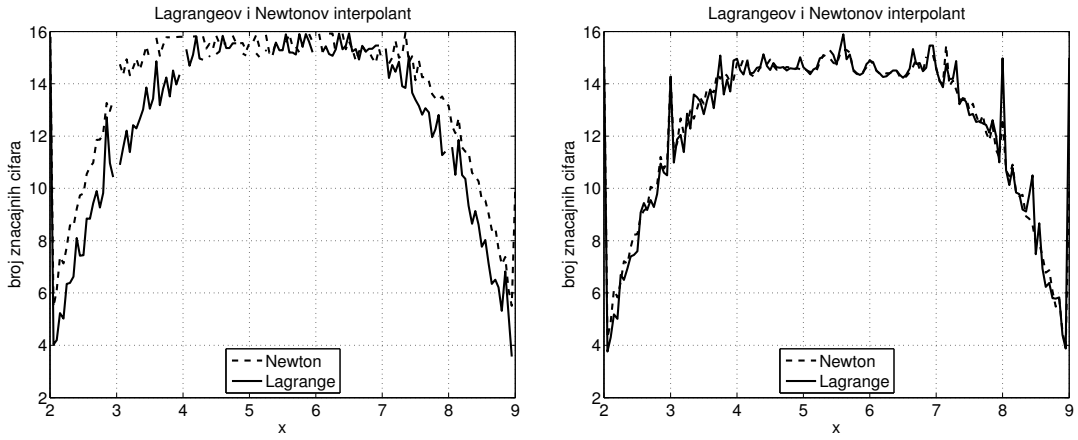
Na osnovu teoreme 4.4, znamo da podeljene razlike konstantne funkcije moraju biti jednake nuli počev od reda jedan. Međutim, kako su naši podaci sa greškom, konstruisane podeljene razlike, koje učestvuju u konstrukciji Newtonovog interpolacionog polinoma, su daleko od zahtevanog ponašanja. Pomenuta slika prikazuje grafik funkcije $\log_{10}|[x_0,\ldots,x_k;f]|$, $k=0,\ldots,59$. Kao što vidimo vrednost podeljenih razlika može biti reda veličine 10^5. Za drugačiju konfiguraciju čvorova ovaj grafik ne mora ostati u važnosti. Slika 4.8, desno, prikazuje istu funkciju za 110 ekvidistantnih interpolacionih čvorova iz intervala $[0,.2]$. Kao što vidimo ovde je greška reda veličine 10^{150}. Različito ponašanje greške u jednom i drugom slučaju je posledica toga što, u prvom slučaju, zbog dužine intervala $[2,7]$, počev od podeljene razlike reda šest prilikom računanja podeljenih razlika delimo brojem većim od jedan. U drugom slučaju, zbog dužine intervala $[0,.2]$, delimo sve vreme brojevima koji su manji od jedinice, tako da greška prilikom računanja podeljenih razlika stalno raste.

Greške koje se javljaju u vrednostima podeljenih razlika najviše se reflektuju u grešci interpolacionog polinoma u okolini tačke x_n, pod uslovom da interpolacioni čvorovi zadovoljavaju uslov $x_0 < x_1 < \cdots < x_n$. Razlog se sastoji u sledećem. Pretpostavimo da smo konstruisali Newtonove interpolacione polinome za skupove podataka $\{(x_i, f(x_i)) \,|\, i=0,\ldots,n\}$ i $\{(x_i, \tilde{f}(x_i)) \,|\, i=0,\ldots,n\}$. Neka su to polinomi P_n i $\tilde{P}_n$ redom. Razliku ovih polinoma možemo izraziti na sledeći način

$$P_n(x) - \tilde{P}_n(x) = \sum_{k=0}^{n} \left([x_0,\ldots,x_k;f] - [x_0,\ldots,x_k;\tilde{f}] \right) \omega_k(x) = \sum_{k=0}^{n} \varepsilon_{0,\ldots,k}\omega_k(x).$$

Čvorni polinomi ω_k, $k=0,\ldots,n$, će imati najveće vrednosti kada se x nalazi u okolini tačke x_n. Ovo dalje znači da će greške $\varepsilon_{0,\ldots,k}$, $k=0,\ldots,n$, imati najviše uticaja na rezultat onda kada se tačka x nalazi u okolini tačke x_n. Sa slika 4.7 se upravo i vidi ovakvo ponašanje.

Postoje slučajevi kada je Newtonov interpolacioni polinom superioran u odnosu na Lagrangeov interpolacioni polinom. Ovo se odnosi na slučaj kad je gornja granica apsolutne greške ulaznih podataka mala i kad je broj interpolacionih čvorova relativno mali. Izaberemo li 50 ekvidistantnih interpolacionih čvorova na intervalu $[2,9]$, uz pretpostavku da su ulazni podaci tačni do na mašinsku preciznost, i interpoliramo li funkciju log, dobijamo grafik značajnih cifara prikazan na slici 4.9, levo. Vidimo da je u ovom slučaju Newtonov interpolacioni polinom superioran. Međutim, ta superiornost se gubi već sa gornjom granicom apsolutne greške 10^{-14}. Kako su vrednosti funkcije



Slika 4.9: Slika levo prikazuje broj značajnih cifara Newtonovog i Lagrangeovog interpolanta za 50 ekvidistantnih interpolacionih tačaka na intervalu $[2, 9]$ funkcije log sa ulaznim podacima mašinske preciznosti, slika desno prikazuje isti grafik ali sa ulaznim podacima koji imaju gornju granicu apsolutne greške 10^{-14}

log na intervalu $[2, 9]$ relativno bliske jedinici, možemo zaključiti da je relativna greška podataka istog reda veličine kao apsolutna. Dakle, superiornost Newtonovog interpolacionog polinoma se gubi već za gornju granicu relativne greške 10^{-14}. Kako su merenja sa tako malom relativnom greškom teško izvodljiva, možemo reći da je Newtonova interpolacija inferiornija od Lagrangeove po pitanju gubitka značajnih cifara.

4.3 Interpolacija u ekvidistantnim tačkama

Ako su interpolacioni čvorovi izabrani ekvidistantno moguće je dobiti posebnu formu Newtonovog interpolacionog polinoma. Pretpostavimo da su interpolacioni čvorovi izabrani tako da važi $a = x_0 < x_1 < \cdots < x_n = b$, pri čemu je $h = x_{i+1} - x_i$, $i = 0, \ldots, n-1$. Onda možemo pisati $x_i = a + ih$, $i = 0, \ldots, n$, i $h = (b-a)/n$. Za vrednosti funkcije u ekvidistantnim tačkama koristimo posebnu notaciju $f_i = f(x_i)$, $i = 0, \ldots, n$.

Definišimo operator prednje razlike Δ i zadnje razlike ∇

$$\Delta f(x) = f(x+h) - f(x), \quad \nabla f(x) = f(x) - f(x-h).$$

Stepen operatora prednje i zadnje razlike definišemo rekurzivno pomoću

$$\Delta^{k+1} f(x) = \Delta^k f(x+h) - \Delta^k f(x), \quad \nabla^{k+1} f(x) = \nabla^k f(x) - \nabla^k f(x-h), \quad k \in \mathbb{N}.$$

Dogovorom, da bi pisanje nekih jednakosti bilo jednostavnije, uzimamo $\Delta^0 f(x) = f(x)$ i $\nabla^0 f(x) = f(x)$.

Ova dva operatora nisu nezavisna. Očigledno je

$$\Delta f(x) = f(x+h) - f(x) = f(x+h) - f(x+h-h) = \nabla f(x+h).$$

Na sličan način se može pokazati da važi

$$\Delta^k f(x) = \nabla^k f(x+kh), \quad k \in \mathbb{N}_0. \tag{4.5}$$

Pomoću ovih operatora možemo vrlo jednostavno izraziti podeljene razlike pod uslovom da su čvorovi ekvidistantni.

Lema 4.5 *Neka su interpolacioni čvorovi $x_i = x_0 + ih$, $i = 0, \ldots, n$, ekvidistantni. Za svako $k = 0, \ldots, n$, i $\ell = 0, \ldots, n-k$, važi*

$$[x_k, \ldots, x_{k+\ell}; f] = \frac{1}{\ell! h^\ell} \Delta^\ell f_k, \quad [x_{k+\ell}, \ldots, x_k; f] = \frac{1}{\ell! h^\ell} \nabla^\ell f_{k+\ell}.$$

Dokaz. Dokaz dajemo indukcijom. Prvo dajemo dokaz prve formule. Očigledno je formula tačna za $\ell = 0$, jer

$$[x_k; f] = f(x_k) = \Delta^0 f(x_k) = \frac{1}{0! h^0} \Delta^0 f_k.$$

Pretpostavimo da je formula tačna za $\ell = m$. Na osnovu rekurentne relacije koju zadovoljavaju podeljene razlike imamo

$$\begin{aligned}[x_k, \ldots, x_{k+m+1}; f] &= \frac{[x_{k+1}, \ldots, x_{k+m+1}; f] - [x_k, \ldots, x_{k+m}; f]}{x_{k+m+1} - x_k} = \frac{\frac{1}{m! h^m} \Delta^m f_{k+1} - \frac{1}{m! h^m} \Delta^m f_k}{(m+1)h} \\ &= \frac{1}{(m+1)! h^{m+1}} (\Delta^m f_{k+1} - \Delta^m f_k) = \frac{1}{(m+1)! h^{m+1}} \Delta^{m+1} f_k.\end{aligned}$$

Da bismo dokazali drugi identitet dovoljno je iskoristiti formulu (4.5). □

Dokazana lema nam omogućava da izrazimo Newtonov interpolacioni polinom u nešto izmenjenom obliku u slučaju ekvidistantnih interpolacionih čvorova.

Teorema 4.9 (Prvi Newtonov interpolacioni polinom) *Ako su čvorovi ekvidistantni $x_i = x_0 + ih$, $i = 0, \ldots, n$, Newtonov interpolacioni polinom funkcije f, sa vrednostima $f_i = f(x_i)$, $i = 0, \ldots, n$, u interpolacionim čvorovima, može se izraziti na sledeći način*

$$P_n(x) = \sum_{k=0}^{n} \binom{p}{k} \Delta^k f_0,$$

gde je $x = x_0 + ph$.

Dokaz. Ako usvojimo $x = x_0 + ph$ čvorne polinome, unutar Newtonovog interpolacionog polinoma, možemo izraziti na sledeći način

$$\omega_k(x) = \prod_{i=0}^{k-1} (x - x_i) = \prod_{i=0}^{k-1} (x_0 + ph - x_0 - ih) = \prod_{i=0}^{k-1} h(p-i) = h^k p(p-1)\cdots(p-k+1).$$

Kombinujući sa izrazom za podeljene razlike dobijenim u lemi 4.5, izraz za Newtonov interpolacioni polinom postaje

$$P_n(x) = \sum_{k=0}^{n} [x_0, \ldots, x_k; f] \omega_k(x) = \sum_{k=0}^{n} \frac{1}{k! h^k} \Delta^k f_0 h^k p(p-1)\cdots(p-k+1) = \sum_{k=0}^{n} \binom{p}{k} \Delta^k f_0.$$

Ovim je dokaz teoreme završen. □

Primetimo da smo u suštini dobili još jednu reprezentaciju algebarskog interpolanta za slučaj ekvidistantnih interpolacionih čvorova. Obično se za konstrukciju prvog Newtonovog interpolacionog polinoma koristi tabela prednjih razlika. Pre nego pređemo na konstrukciju dokažimo još jedan mogući izraz za Newtonov interpolacioni polinom u slučaju ekvidistantnih interpolacionih čvorova.

Teorema 4.10 (Drugi Newtonov interpolacioni polinom) *U slučaju ekvidistantnih čvorova $x_i = x_0 + ih$, $i = 0, \ldots, n$, Newtonov interpolacioni polinom funkcije f, sa vrednostima $f_i = f(x_i)$, $i = 0, \ldots, n$, u interpolacionim čvorovima, može se izraziti na sledeći način*

$$P_n(x) = \sum_{k=0}^{n} (-1)^k \binom{-p}{k} \nabla^k f_n,$$

gde je $x = x_n + ph$.

Dokaz. Da bismo dokazali ovaj oblik Newtonovog interpolacionog polinoma, prisetimo se da oblik Newtonovog interpolacionog polinoma ostaje u važnosti ako izvršimo renumeraciju interpolacionih čvorova. U ovom slučaju čvorni polinomi izgledaju ovako

$$\begin{aligned}\omega_k(x) &= (x - x_n) \cdots (x - x_{n-k+1}) = (x_n + ph - x_n) \cdots (x_n + ph - x_n + (k-1)h) \\ &= (-1)^k h^k (-p)(-p-1) \cdots (-p-k+1).\end{aligned}$$

Prednje razlike u ovom slučaju postaju zadnje razlike, tako da dobijamo izraz iz tvrđenja teoreme. □

Konstrukcija prvog i drugog Newtonovog interpolacionog polinoma se uglavnom izvodi koristeći tabele operatora prednje i zadnje razlike nad vrednostima funkcije. Ove tabele su slične tabelama podeljenih razlika, ali postoji bitna razlika. Naime, u tabeli prednjih razlika nema deljenja razlikama vrednosti interpolacionih čvorova. Tabela operatora prednjih razlika izgleda ovako

$$\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & \Delta f_0 & & & \\
x_1 & f_1 & & \Delta^2 f_0 & & \\
 & & \Delta f_1 & & \Delta^3 f_0 & \\
x_2 & f_2 & & \Delta^2 f_1 & & \Delta^4 f_0 \\
 & & \Delta f_2 & & \Delta^3 f_1 & \\
x_3 & f_3 & & \Delta^2 f_2 & & \\
 & & \Delta f_3 & & & \\
x_4 & f_4 & & & & \\
\vdots & \vdots & & & &
\end{array}$$

Potpuno isto izgleda i tabela operatora zadnjih razlika. Potrebno je samo primeniti jednakost 4.5 da ista tabela postane

$$\begin{array}{llllll}
\vdots & \vdots & & & & \\
x_0 & f_0 & & & & \\
 & & \nabla f_1 & & & \\
x_1 & f_1 & & \nabla^2 f_2 & & \\
 & & \nabla f_2 & & \nabla^3 f_3 & \\
x_2 & f_2 & & \nabla^2 f_3 & & \nabla^4 f_4 \\
 & & \nabla f_3 & & \nabla^3 f_4 & \\
x_3 & f_3 & & \nabla^2 f_4 & & \\
 & & \nabla f_4 & & & \\
x_4 & f_4 & & & &
\end{array}$$

Da budemo potpuno jasni, nije potrebno konstruisati novu tabelu, već je dovoljno samo na nov način protumačiti staru.

Ilustrujmo konstrukciju tabele primerom. Konstruišimo prvi i drugi Newtonov interpolacioni polinom u 5 interpolacionih čvorova za funkciju log na intervalu $[2, 9]$. S obzirom na to da konstruišemo prvi i drugi Newtonov interpolacioni polinom, interpolacioni čvorovi su zadati ekvidistantno. Odavde dobijamo $h = (9-2)/4 = 1.75$, $x_i = 2 + 1.75i$, $i = 0, \ldots, 4$. Koristeći vrednosti funkcije log sa šest značajnih cifara dobijamo sledeću tebelu.

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$
0	2	$\underline{.693147}$				
			$\underline{.628613}$			
1	3.75	1.32176		$\underline{-.245623}$		
			.382990		$\underline{.138883}$	
2	5.5	1.70475		−.106740		$\overline{\underline{-.092173}}$
			.276250		$\overline{.046710}$	
3	7.25	1.98100		$\overline{-.060030}$		
			$\overline{.216220}$			
4	9	$\overline{2.19722}$				

Podvučene su one prednje razlike koje su potrebne za konstrukciju prvog Newtonovog interpolacionog polinoma, dok su nadvučene one zadnje razlike koje su potrebne za konstrukciju drugog Newtonovog interpolacionog polinoma. Formirajmo prvi Newtonov interpolacioni polinom. Koristeći teoremu 4.9, dobijamo

$$\begin{aligned} P_4(x) &= .693147 + .628613p - .245623\frac{p(p-1)}{2} + .138883\frac{p(p-1)(p-2)}{3\cdot 2} \\ &- .092173\frac{p(p-1)(p-2)(p-3)}{4\cdot 3\cdot 2} = .693147 + .628613p - .1228115p(p-1) \\ &+ .02314717p(p-1)(p-2) - .0038405417p(p-1)(p-2)(p-3), \end{aligned}$$

gde je $p = (x-2)/1.75$. Na sličan način, koristeći teoremu 4.10, dobijamo

$$\begin{aligned} P_4(x) &= 2.19722 + (-1).21622(-p) - .06003\frac{(-p)(-p-1)}{2} + (-1).04671\frac{(-p)(-p-1)(-p-2)}{3\cdot 2} \\ &- .092173\frac{(-p)(-p-1)(-p-2)(-p-3)}{4\cdot 3\cdot 2} = 2.19722 + .21622p - .030015p(p+1) \\ &+ .007785p(p+1)(p+2) - .0038405417p(p+1)(p+2)(p+3), \end{aligned}$$

gde je $p = (x-9)/1.75$.

Pogodnost korišćenja prvog i drugog Newtonovog interpolacionog polinoma se svodi na ideju Newtonove interpolacije. Naime, moguće je vrlo jednostavno dodati interpolacioni čvor, odnosno, povećati stepen interpolacionog polinoma. Na primer, u slučaju da želimo da dodamo prethodnoj tabeli interpolacioni čvor u tački $x_5 = 10.75$, u prethodnoj tabeli potrebno je izvršiti samo

izračunavanja po jednoj dijagonali, što ilustruje sledeća tabela

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$	$\Delta^5 f_k$
0	2	$\underline{.693147}$					
			$\underline{.628613}$				
1	3.75	1.32176		$\underline{-.245623}$			
			.382990		$\underline{.138883}$		
2	5.5	1.70475		$-.106740$		$\underline{\overline{-.092173}}$	
			.276250		$\overline{.046710}$		$\underline{.066963}$
3	7.25	1.98100		$\overline{-.060030}$		$-.025210$	
			$\overline{.216220}$		.021500		
4	9	$\overline{2.19722}$		$-.038530$			
			.177690				
5	10.75	2.37491					

Kao što vidimo, da bismo konstruisali prvi Newtonov interpolacioni polinom stepena 5 dovoljno je dodati jedan član polinomu stepena 4. Važi sledeća jednakost

$$\begin{aligned} P_5(x) &= P_4(x) + .066963\frac{p(p-1)(p-2)(p-3)(p-4)}{5\cdot 4\cdot 3\cdot 2} \\ &= P_4(x) + .000558025p(p-1)(p-2)(p-3)(p-4), \end{aligned} \tag{4.6}$$

gde je $p = (x-2)/1.75$.

Međutim, kao što vidimo ne postoji veza između drugog Newtonovog interpolacionog polinoma konstruisanog za originalnu tabelu i proširenu tabelu. Veza bi postojala kad bismo tabelu proširili dodavanjem tačke .25. Ovu osobinu ilustrujemo sledećom tabelom konačnih razlika.

k	x_k	$\nabla^0 f_k$	$\nabla^1 f_k$	$\nabla^2 f_k$	$\nabla^3 f_k$	$\nabla^4 f_k$	$\nabla^5 f_k$
-1	.25	-1.38629					
			2.07944				
0	2	$\underline{.693147}$		-1.45083			
			$\underline{.628613}$		1.20521		
1	3.75	1.32176		$\underline{-.245623}$		-1.06633	
			.382990		$\underline{.138883}$		$\overline{.974157}$
2	5.5	1.70475		$-.106740$		$\underline{\overline{-.092173}}$	
			.276250		$\overline{.046710}$		
3	7.25	1.98100		$\overline{-.060030}$			
			$\overline{.216220}$				
4	9	$\overline{2.19722}$					

Na osnovu ove tabele je moguće konstruisati drugi Newtonov interpolacioni polinom stepena 5 koristeći drugi Newtonov interpolacioni polinom stepena 4. Dovoljno je dodati samo jedan član. Važi sledeća jednakost

$$\begin{aligned} P_5(x) &= P_4(x) + (-1)\,.974157\frac{(-p)(-p-1)(-p-2)(-p-3)(-p-4)}{5\cdot 4\cdot 3\cdot 2} \\ &= P_4(x) + .008117975p(p+1)(p+2)(p+3)(p+4), \end{aligned} \tag{4.7}$$

gde je $p = (x - 9)/1.75$. Takođe, dodavanjem interpolacionog čvora u tački .25 više nije moguće koristiti staru vrednost prvog Newtonovog interpolacionog polinoma za konstrukciju novog, što se iz tabele jasno vidi.

Primetimo da interpolacioni polinomi (4.6) i (4.7) nisu jednaki. Ovi polinomi su konstruisani za različite skupove interpolacionih čvorova pa samim tim ne mogu biti jednaki.

4.3.1 Numerička konstrukcija prvog i drugog Newtonovog interpolanta

Funkcije koje implementiraju numeričku konstrukciju prvog i drugog Newtonovog interpolacionog polinoma mogu biti sledeće. Za prvi Newtonov interpolacioni polinom

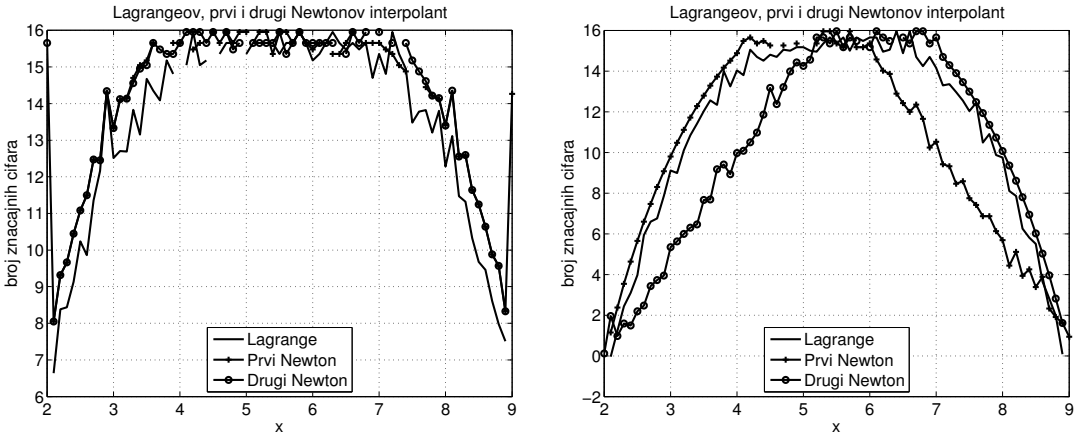
```
function poly=newtonPrviPoly(cvorovi,vrednosti,x)
%NEWTONEPRVIPOLY(CVOROVI,VREDNOSTI,X) konstruise prvi Newtonov
% interpolacioni polinom argument CVOROVI odredjuje
% interpolacione cvorove, sa vrednostima funkcije VREDNOSTI
% u tacki X

k = 1; omega = 1;
n = length(cvorovi)-1;
tabela = vrednosti;
h = (cvorovi(end)-cvorovi(1))/n;
p = (x-cvorovi(1))/h;
poly = tabela(1);
tabela = diff(tabela);
while not(isempty(tabela))
    omega = omega.*((p-k+1)/k);
    poly = poly+tabela(1).*omega;
    tabela = diff(tabela);
    k = k+1;
end
end
```

i za drugi Newtonov interpolacioni polinom

```
function poly=newtonDrugiPoly(cvorovi,vrednosti,x)
%NEWTONEDRUGIPOLY(CVOROVI,VREDNOSTI,X) konstruise drugi Newtonov
% interpolacioni polinom argument CVOROVI odredjuje
% interpolacione cvorove, sa vrednostima funkcije VREDNOSTI
% u tacki X

k = 1; omega = 1;
n = length(cvorovi)-1;
tabela = vrednosti;
h = (cvorovi(end)-cvorovi(1))/n;
p = (cvorovi(end)-x)/h;
poly = tabela(end);
tabela = diff(tabela);
minus = -1;
while not(isempty(tabela))
```

```
    omega = omega.*((p-k+1)/k);
    poly = poly+minus*tabela(end).*omega;
    tabela = diff(tabela);
    k = k+1;
    minus = -minus;
end
end
```

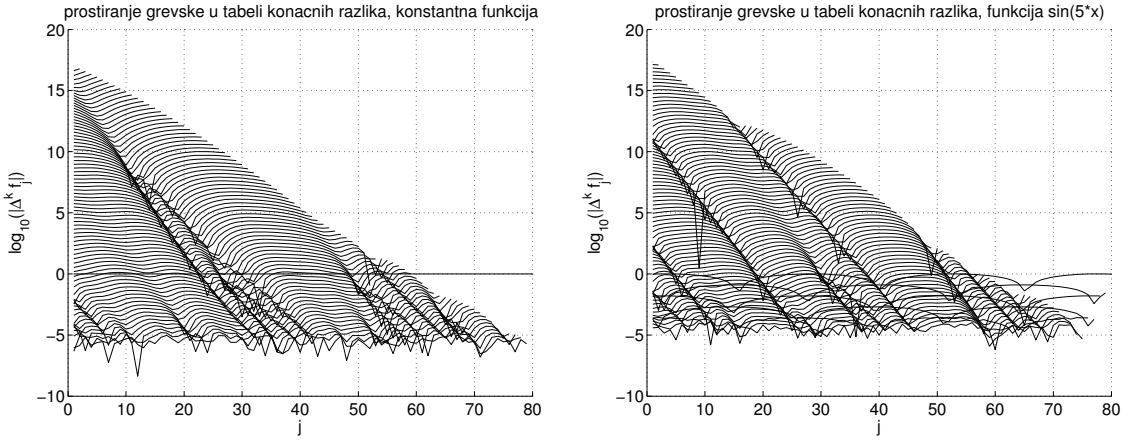
Nas pre svega u ovoj sekciji interesuje osetljivost numeričke konstrukcije interpolanta korišćenjem ovih funkcija. Slika 4.10, levo, prikazuje značajne cifre u prvom i drugom Newtonovom interpolacionom polinomu i Lagrangeovom interpolacionom polinomu za 40 ekvidistantnih interpolacionih čvorova na intervalu $[2, 9]$ pri interpolaciji funkcije log.



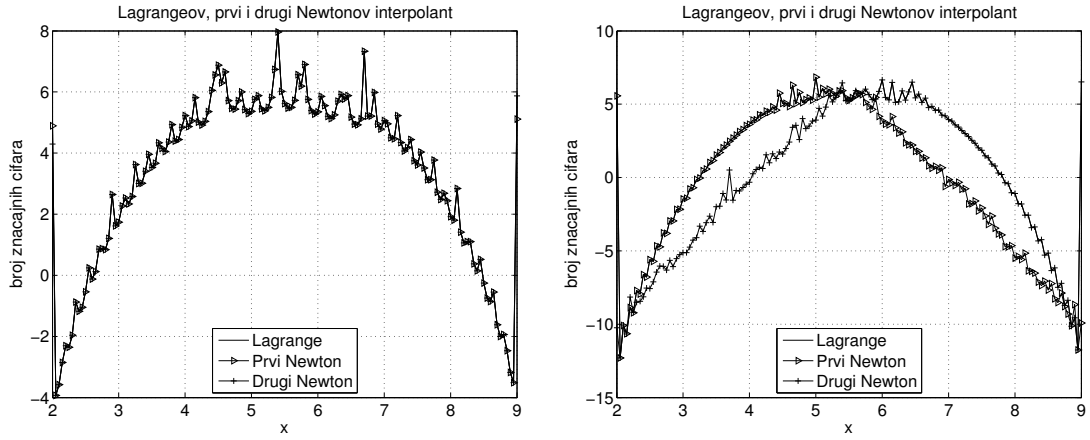
Slika 4.10: Slika levo prikazuje značajne cifre Lagrangeovog, prvog i drugog Newtonovog interpolanta za 40 ekvidistantnih interpolacionih čvorova na intervalu $[2, 9]$ pri interpolaciji funkcije log, slika desno prikazuje isti interpolacioni problem ali sa 70 interpolacionih čvorova.

Ono što je zanimljivo je da prvi i drugi Newtonov interpolacioni polinom imaju otprilike za jednu značajnu cifru više u odnosu na Lagrangeov interpolacioni polinom za 40 interpolacionih čvorova. Međutim, već sa 70 interpolacionih čvorova prvi Newtonov interpolant ima značajniju grešku u okolini tačke x_{69}, dok drugi Newtonov interpolacioni polinom pravi značajniju grešku u okolini tačke x_0 u odnosu na Lagrangeov interpolant. Slika 4.10, desno, jasno ilustruje ovo ponašanje. Ovo nije teško shvatiti, ako se setimo da je prvi Newtonov interpolacioni polinom u stvari Newtonov interpolacioni polinom konstruisan u ekvidistantnim tačkama. Ponašanje prvog Newtonovog interpolacionog polinoma je, kao što vidimo, identično ponašanju Newtonovog interpolacionog polinoma. Novina je ponašanje drugog interpolacionog polinoma. Međutim, i ovo je jednostavno shvatiti, ako se setimo da je drugi Newtonov interpolacioni polinom isto Newtonov interpolacioni polinom, samo sa obrnutim rasporedom tačaka.

Razlog za lošije ponašanje prvog i drugog Newtonovog interpolacionog polinoma je isti kao i u slučaju Newtonovog interpolacionog polinoma. Naime, u tabeli konačnih razlika dolazi do prostiranja greške. Ilustracija prostiranja greške kroz tabelu konačnih razlika je data na slici 4.11. Slika prikazuje grafik $\log_{10} |\Delta^k f_j|$, $j = 0, \ldots, n-k$, $k = 0, \ldots, n$. Tabela konačnih razlika je konstruisana za funkcije $f(x) = 1$ i $f(x) = \sin(5x)$, na intervalu $[-1, 1]$ za 80 ekvidistantnih tačaka,



Slika 4.11: Slika levo ilustruje prostiranje greške u tabeli konačnih razlika za konstantnu funkciju, dok slika desno ilustruje isto ali za funkciju $\sin(5x)$, podaci u oba slučaja imaju gornju granicu apsolutne greške 10^{-5}.



Slika 4.12: Slika levo prikazuje broj značajnih cifara Lagrangeovog, prvog i drugog Newtonovog interpolanta za 40 ekvidistantnih interpolacionih čvorova na intervalu $[2, 9]$ pri interpolaciji funkcije log, slika desno prikazuje isti interpolacioni problem ali sa 70 interpolacionih čvorova. Podaci kojima je zadata funkcija log imaju gornju granicu apsolutne greške 10^{-5}.

pri čemu su podaci zadati sa gornjom granicom apsolutne greške 10^{-5}. Vidimo da gornja granica apsolutne greške stalno raste sa redom konačne razlike. Konačna razlika reda 80 ima gornju granicu apsolutne greške reda veličine 10^{17}. Ovo ponašanje je tipično. Ovako velike greške u konačnim razlikama višeg reda proizvode velike greške u konstruisanom Newtonovom interpolantu ako je stepen polinoma relativno veliki. Međutim, za relativno mali broj interpolacionih čvorova manje od trideset, prvi i drugi Newtonov interpolant su ravnopravni ili superiorniji u odnosu na Lagrangeov interpolacioni polinom.

Pomenuta superiornost Newtonovih interpolacionih polinoma, prikazana na slici 4.10, nestaje ako dopustimo da ulazni podaci imaju manju tačnost od mašinske preciznosti. Efekat je isti kao u slučaju Newtonovog i Lagrangeovog interpolanta. Slika 4.12 sadrži ponovljene pomenute grafike, ali ovoga puta vrednosti funkcije log imaju gornju granicu apsolutne greške 10^{-5}. Kao što vidimo pomenuta superiornost Newtonovih interpolacionih polinoma nestaje.

4.3.2 Račun konačnih razlika

Da bismo mogli da damo formule za Stirlingov i Besselov interpolant u sledećem odeljku neophodno je uvođenje još nekih operatora. Pored već uvedenih operatora prednje razlike Δ i zadnje razlike ∇, uvodimo operatore

$$\mu f(x) = \frac{1}{2}\left(f\left(x+\frac{h}{2}\right)+f\left(x-\frac{h}{2}\right)\right), \quad \delta f(x) = f\left(x+\frac{h}{2}\right)-f\left(x-\frac{h}{2}\right), \quad Ef(x) = f(x+h).$$

Operator μ obično nazivamo operator usrednjavanja, dok operator δ nazivamo operator centralne razlike. Operator E se naziva operatorom pomeraja ili šiftovanja. Da bismo imali kompletan skup operatora uvodimo i jedinični operator 1, čije delovanje definišemo na sledeći način $1f(x) = f(x)$.

Kao i u slučaju operatora Δ i ∇, definišemo proizvode operatora μ i δ i odgovarajuće stepene sa

$$\delta^0 f(x) = f(x), \quad \delta^{k+1} f(x) = \delta^k f\left(x+\frac{h}{2}\right) - \delta^k f\left(x-\frac{h}{2}\right),$$
$$\mu\delta^{k+1} f(x) = \mu\delta^k f\left(x+\frac{h}{2}\right) - \mu\delta^k f\left(x-\frac{h}{2}\right), \quad k \geq 0.$$

Stepenovanje operatora E se definiše na sledeći način

$$E^m f(x) = f(x+mh),$$

pri čemu m može biti proizvoljan realan broj.

Primetimo da sve operatore možemo opisati koristeći samo operatore 1 i E. Naime, važi sledeća vrlo jednostavna lema.

Lema 4.6 *Važe sledeće jednakosti*

$$\Delta = E - 1, \quad \nabla = 1 - E^{-1}, \quad \mu = \frac{1}{2}(E^{1/2} + E^{-1/2}), \quad \delta = E^{1/2} - E^{-1/2}.$$

Svi operatori u skupu $\{1, E, \Delta, \nabla, \mu, \delta\}$ *međusobno komutiraju.*

Dokaz. Na primer, ako izaberemo operator Δ dobijamo direktno

$$\Delta f(x) = f(x+h) - f(x) = Ef(x) - 1f(x) = (E-1)f(x).$$

Za ostale jednakosti dokaze možemo dobiti na identičan način. Da bismo dokazali deo vezan za komutativnost dovoljno je primetiti da operatori E i 1 komutiraju, jer važi

$$E1f(x) = Ef(x) = f(x+h) = 1f(x+h) = 1Ef(x).$$

Kako su svi operatori iz skupa $\{1, E, \Delta, \nabla, \delta, \mu\}$ linearne kombinacije operatora 1 i E, to sledi da međusobno komutiraju, jer operatori 1 i E komutiraju. □

Sledeća lema daje vezu između novouvedenih operatora μ i δ i operatora Δ.

Lema 4.7 *Za svako $k \in \mathbb{N}$, važi*

$$\delta^{2k} f(x) = \Delta^{2k} f(x - kh), \quad \mu\delta^{2k-1} f(x) = \frac{\Delta^{2k-1}}{2}(1 + E^{-1}) f(x - (k-1)h),$$
$$\delta^{2k-1} f\left(x + \frac{h}{2}\right) = \Delta^{2k-1} f(x - (k-1)h), \quad \mu\delta^{2k} f\left(x + \frac{h}{2}\right) = \frac{\Delta^{2k}}{2}(1 + E^{-1}) f(x - (k-1)h).$$

Dokaz. Dokaz je prilično jednostavan. Posmatrajmo jednakost

$$\begin{aligned}\delta^2 f(x) &= \delta f\left(x + \frac{h}{2}\right) - \delta f\left(x - \frac{h}{2}\right) = f(x+h) - f(x) - f(x) + f(x-h) \\ &= f(x+h) - 2f(x) + f(x-h) = \Delta^2 f(x-h) = \Delta^2 E^{-1} f(x).\end{aligned}$$

Na osnovu ove jednakosti dobijamo $\delta^2 = \Delta^2 E^{-1}$. Odavde zaključujemo da važi

$$\delta^{2k} = (\delta^2)^k = (\Delta^k E^{-1})^k = \Delta^{2k} E^{-k},$$

ili ako primenimo na funkciju

$$\delta^{2k} f(x) = \Delta^{2k} E^{-k} f(x) = \Delta^{2k} f(x - kh).$$

Ovim je završen dokaz prve jednakosti.

Za dokaz druge jednakosti primetimo da važi

$$\begin{aligned}\mu\delta f(x) &= \mu f\left(x + \frac{h}{2}\right) - \mu f\left(x - \frac{h}{2}\right) = \frac{1}{2}(f(x+h) + f(x) - f(x) - f(x-h)) \\ &= \frac{1}{2}(f(x+h) - f(x) + f(x) - f(x-h)) = \frac{1}{2}(\Delta f(x) + \Delta E^{-1} f(x)) = \frac{\Delta}{2}(1 + E^{-1}) f(x).\end{aligned}$$

Koristeći ovu jednakost i dokazanu prvu jednakost, dobijamo

$$\mu\delta^{2k-1} f(x) = \mu\delta\delta^{2k-2} f(x) = \frac{\Delta}{2}(1 + E^{-1})\Delta^{2k-2} E^{-(k-1)} f(x) = \frac{\Delta^{2k-1}}{2}(1 + E^{-1}) f(x - (k-1)h),$$

čime je završen dokaz druge jednakosti.

Za dokaz treće i četvrte jednakosti možemo postupiti na sledeći način

$$\begin{aligned}\delta^{2k-1} f\left(x + \frac{h}{2}\right) &= E^{1/2}\delta\delta^{2k-2} f(x) = E^{1/2}(E^{1/2} - E^{-1/2})\Delta^{2k-2} E^{-k+1} f(x) \\ &= (E-1)\Delta^{2k-2} E^{-k+1} f(x) = \Delta^{2k-1} f(x - (k-1)h), \\ \mu\delta^{2k} f\left(x + \frac{h}{2}\right) &= \mu\delta\delta^{2k-1} f\left(x + \frac{h}{2}\right) = \frac{\Delta}{2}(1 + E^{-1})\Delta^{2k-1} f(x - (k-1)h) \\ &= \frac{\Delta^{2k}}{2}(1 + E^{-1}) f(x - (k-1)h).\end{aligned}$$

Ovim smo dokazali treću i četvrtu jednakost. □

4.3.3 Stirlingov i Besselov interpolacioni polinom

Stirlingov i Besselov interpolant su naravno identični Lagrangeovom ili Newtonovom interpolantu. Konstruišu se nad ekvidistantnim skupom interpolacionih čvorova. Jedina razlika je, kao i u slučaju Newtonovih interpolanata, pogodnost primene u pojedinim slučajevima.

Stirlingov interpolant se konstruiše za neparan broj interpolacionih čvorova. Konstrukcija se izvodi koristeći sledeću teoremu.

Teorema 4.11 (Stirlingov interpolacioni polinom) *Algebarski interpolant P_{2n} u interpolacionim čvorovima x_i, $i = -n, \ldots, 0, \ldots, n$, sa vrednostima funkcije f_i, $i = -n, \ldots, 0, \ldots, n$, u interpolacionim čvorovima, može se izraziti na sledeći način*

$$P_{2n}(x) = f_0 + p\sum_{k=1}^{n}\left(\prod_{i=1}^{k-1}\frac{p^2-i^2}{2i(2i+1)}\right)\left(\mu + \frac{p}{2k}\delta\right)\delta^{2k-1}f_0,$$

gde je $x = x_0 + ph$.

Mi nećemo dati dokaz ove teoreme. Međutim, daćemo interpretaciju veličina koje učestvuju u izrazu za interpolant. Naime, koristeći lemu 4.7, možemo prepoznati da veličine koje učestvuju u konstrukciji interpolanta možemo dobiti koristeći tablicu konačnih razlika. Koristeći lemu 4.7, izraz za Stirlingov interpolacioni polinom postaje

$$P_{2n}(x) = f_0 + \frac{p}{2}\sum_{k=1}^{n}\left(\prod_{i=1}^{k-1}\frac{p^2-i^2}{2i(2i+1)}\right)\left(\Delta^{2k-1}f_{-k+1} + \Delta^{2k-1}f_{-k} + \frac{p}{k}\Delta^{2k}f_{-k}\right),$$

gde je $x = x_0 + ph$.

Koristeći ovaj oblik Stirlingovog interpolacionog polinoma, možemo podatke neophodne za konstrukciju pročitati iz tabele prednjih razlika. Na sledećoj slici su podvučeni podaci koji su potrebni za konstrukciju polinoma P_4, $n = 2$.

$$\begin{array}{lllllll}
\vdots & \vdots & & & & & \\
x_{-2} & f_{-2} & & & & & \\
 & & \Delta f_{-2} & & & & \\
x_{-1} & f_{-1} & & \Delta^2 f_{-2} & & & \\
 & & \underline{\Delta f_{-1}} & & \underline{\Delta^3 f_{-2}} & & \\
x_0 & \underline{f_0} & & \underline{\Delta^2 f_{-1}} & & \underline{\Delta^4 f_{-2}} & \\
 & & \underline{\Delta f_0} & & \underline{\Delta^3 f_{-1}} & & \\
x_1 & f_1 & & \Delta^2 f_0 & & & \\
 & & \Delta f_1 & & & & \\
x_2 & f_2 & & & & & \\
\vdots & \vdots & & & & &
\end{array}$$

Ilustrujmo konstrukciju primerom koji koristimo kroz celu glavu. Naime, kostruišimo Stirlingov interpolant za funkciju log na intervalu $[2, 9]$ u 5 čvorova. Kako konstruišemo Stirlingov interpolant, interpolacioni čvorovi su raspoređeni ekvidistantno sa korakom $h = (9-2)/4 = 1.75$. Tabela konačnih razlika je ista kao ona koju smo već koristili prilikom konstrukcije prvog i drugog

Newtonovog interpolanta.

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$
-2	2	.693147				
			.628613			
-1	3.75	1.32176		$-.245623$		
			.382990		.138883	
0	5.5	1.70475		$-.106740$		$-.092173$
			.276250		.046710	
1	7.25	1.98100		$-.060030$		
			.216220			
2	9	2.19722				

Na osnovu ove tablice konačnih razlika, dobijamo Stirlingov interpolacioni polinom

$$\begin{aligned} P_4(x) &= f_0 + \frac{1}{2}\frac{p}{1!}\left(\Delta f_0 + \Delta f_{-1} + \frac{p}{1}\Delta^2 f_{-1}\right) + \frac{1}{2}\frac{p}{3!}(p^2-1^2)\left(\Delta^3 f_{-1} + \Delta^3 f_{-2} + \frac{p}{2}\Delta^4 f_{-2}\right) \\ &= 1.70475 + \frac{1}{2}p\,(.38299 + .27625 + p(-.10674)) \\ &+ \frac{1}{2}\frac{p}{6}(p^2-1^2)\left(.138883 + .04671 + \frac{p}{2}(-.092173)\right) = 1.70475 + p(.32962 - .05337p) \\ &+ p(p^2-1)(.0154661 - .00384054p), \end{aligned}$$

gde je $p = (x - 5.5)/1.75$.

Za Stirlingove interpolacione polinome važi sledeća rekurentna relacija

$$P_{2n+2}(x) = P_{2n}(x) + \frac{p}{2}\left(\prod_{i=1}^{n}\frac{p^2-i^2}{2i(2i+1)}\right)(\Delta^{2n+1}f_{-n} + \Delta^{2n+1}f_{-n-1} + \frac{p}{n+1}\Delta^{2n+2}f_{-n-1}).$$

Iz ove rekurentne relacije možemo jasno sagledati motiv postojanja Stirlingovog interpolanta. Vidimo da je za konstrukciju interpolanta P_{2n+2} moguće iskoristiti interpolant P_{2n}, drugim rečima, moguće je koristiti već postojeću tabelu konačnih razlika, koja je samo proširena za po jednu dijagonalu sa donje i gornje strane. Ilustrujemo to primerom konstrukcije polinoma P_6, koristeći već konstruisan polinom P_4, za funkciju log. Tabela podeljenih razlika u ovom slučaju izgleda ovako.

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$	$\Delta^5 f_k$	$\Delta^6 f_k$
-3	.25	$\overline{-1.38629}$						
			$\overline{2.07944}$					
-2	2	.693147		$\overline{-1.45084}$				
			.628613		$\overline{1.20520}$			
-1	3.75	1.32176		$-.245623$		$\overline{-1.06632}$		
			.382990		.138883		$\overline{.974145}$	
0	5.5	1.70475		$-.106740$		$-.092173$		$\overline{-.907182}$
			.276250		.046710		$\overline{.066963}$	
1	7.25	1.98100		$-.060030$		$\overline{-.025210}$		
			.216220		$\overline{.021500}$			
2	9	2.19722		$\overline{-.038530}$				
			$\overline{.177690}$					
3	10.75	$\overline{2.37491}$						

Podaci potrebni za konstrukciju polinoma P_6 su podvučeni, dok su dodati podaci u tabeli konačnih razlika nadvučeni.

Polinom P_6 izgleda ovako

$$\begin{aligned} P_6(x) &= P_4(x) + \frac{p(p^2-1)(p^2-4)}{240}(.974145 + .066963 - .907182\frac{p}{3}) \\ &= P_4(x) + p(p^2-1)(p^2-4)(.00433795 - .00125998p), \end{aligned}$$

gde je $x = x_0 + ph$.

Za Besselov interpolacioni polinom imamo sličnu situaciju. Jedina razlika između Stirlingovog i Besselovog interpolanta je što Besselov interpolant koristi paran broj interpolacionih čvorova.

Teorema 4.12 (Besselov interpolacioni polinom) *Algebarski interpolant P_{2n+1} u interpolacionim čvorovima x_i, $i = -n, \ldots, 0, \ldots, n+1$, sa vrednostima funkcije f_i, $i = -n, \ldots, 0, \ldots, n+1$, u interpolacioni čvorovima, može se izraziti na sledeći način*

$$P_{2n+1}(x) = \sum_{k=0}^{n} \left(\prod_{i=1}^{k} \frac{(p+i-1)(p-i)}{(2i-1)(2i)} \right) \left(\mu + \frac{p-1/2}{2k+1}\delta \right) \delta^{2k} f_{1/2},$$

gde je $x = x_0 + ph$.

Na sličan način kao kod Stirlingove interpolacione formule, koristeći lemu 4.7, možemo pokazati da Besselov interpolant ima sledeću formu

$$P_{2n+1}(x) = \sum_{k=0}^{n} \left(\prod_{i=1}^{k} \frac{(p+i-1)(p-i)}{(2i-1)(2i)} \right) \left(\frac{1}{2}(\Delta^{2k} f_{-k+1} + \Delta^{2k} f_{-k}) + \frac{p-1/2}{2k+1}\Delta^{2k+1} f_{-k} \right).$$

Sledeća slika ilustruje koje razlike iz tabele konačnih razlika su nam potrebne za konstrukciju polinoma P_5.

$\vdots$	$\vdots$					
x_{-2}	f_{-2}					
		Δf_{-2}				
x_{-1}	f_{-1}		$\Delta^2 f_{-2}$			
		Δf_{-1}		$\Delta^3 f_{-2}$		
x_0	$\underline{f_0}$		$\underline{\Delta^2 f_{-1}}$		$\underline{\Delta^4 f_{-2}}$	
		$\underline{\Delta f_0}$		$\underline{\Delta^3 f_{-1}}$		$\underline{\Delta^5 f_{-2}}$
x_1	$\underline{f_1}$		$\underline{\Delta^2 f_0}$		$\underline{\Delta^3 f_{-1}}$	
		Δf_1		$\Delta^3 f_0$		
x_2	f_2		$\Delta^2 f_1$			
		Δf_2				
x_3	f_3					
$\vdots$	$\vdots$					

Konstrukciju ilustrujemo našim standardnim primerom. Konstruisaćemo Besselov interpolant stepena 5 za funkciju log, sa korakom $h = 1.75$, na intervalu $[2, 10.75]$. Tabela konačnih razlika izgleda

ovako

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$	$\Delta^5 f_k$
-2	2	.693147					
			.628613				
-1	3.75	1.32176		$-.245623$			
			.382990		.138883		
0	5.5	1.70475		$-.106740$		$-.092173$	
			.276250		.046710		.066963
1	7.25	1.98100		$-.060030$		$-.025210$	
			.216220		.021500		
2	9	2.19722		$-.038530$			
			.177690				
3	10.75	2.37491					

Veličine koje su potrebne za konstrukciju polinoma su podvučene. Interpolant P_5 izgleda ovako

$$\begin{aligned}
P_5(x) &= \frac{1}{2}(f_1 + f_0) + \frac{p-1/2}{1}\Delta f_0 + \frac{p(p-1)}{1\cdot 2}\left(\frac{1}{2}(\Delta^2 f_0 + \Delta^2 f_{-1}) + \frac{p-1/2}{3}\Delta^3 f_{-1}\right) \\
&+ \frac{p(p-1)(p+1)(p-2)}{1\cdot 2\cdot 3\cdot 4}\left(\frac{1}{2}\left(\Delta^4 f_{-1} + \Delta^4 f_{-2}\right) + \frac{p-1/2}{5}\Delta^5 f_{-2}\right) \\
&= \frac{1}{2}(1.70475 + 1.98100) + .276250(p-1/2) \\
&+ \frac{p(p-1)}{2}\left(\frac{1}{2}(-.060030 - .106740) + .046710\frac{p-1/2}{3}\right) \\
&+ \frac{p(p-1)(p+1)(p-2)}{24}\left(\frac{1}{2}(-.092173 - .025210) + .066963\frac{p-1/2}{5}\right) \\
&= 1.84288 + .276250(p-1/2) + p(p-1)(-.041693 + .007785(p-1/2)) \\
&+ p(p-1)(p+1)(p-2)(-.00244548 + .000558025(p-1/2)).
\end{aligned}$$

Besselov interpolant zadovoljava sličnu rekurentnu relaciju kao i Stirlingov interpolant, naime važi sledeća rekurentna relacija

$$\begin{aligned}
P_{2n+3}(x) &= P_{2n+1}(x) \\
&+ \left(\prod_{i=1}^{n+1}\frac{(p+i-1)(p-i)}{(2i-1)(2i)}\right)\left(\frac{1}{2}(\Delta^{2(n+1)} f_{-n} + \Delta^{2(n+1)} f_{-n-1}) + \frac{p-1/2}{2n+3}\Delta^{2n+3} f_{-n-1}\right).
\end{aligned}$$

Kao i u slučaju Stirlingovog interpolanta, ova rekurentna relacija znači da nije potrebno konstruisati tabelu konačnih razlika iznova, već je dovoljno samo izračunati po jednu dijagonalu sa donje i gornje strane. Primer je gotovo isti kao u slučaju Stirlingovog interpolanta i nećemo ga navoditi.

4.3.4 Numerička konstrukcija Stirlingovog i Besselovog interpolanta

Funkcije koje implementiraju numeričku konstrukciju Stirlingovog i Besselovog interpolanta mogu biti date na sledeći način:

```
function poly=stirlingPoly(cvorovi,vrednosti,x)
%STIRLINGPOLY(CVOROVI,VREDNOSTI,X) konstruise Stirlingov
```

```
% interpolacioni polinom, argument CVOROVI odredjuje
% interpolacione cvorove, sa vrednostima funkcije VREDNOSTI
% u tacki X

k = 1; poly = 0; mul = 1;
n = (length(cvorovi)-1)/2;
tabela = vrednosti;
h = (cvorovi(end)-cvorovi(1))/(2*n);
p = (x-cvorovi(n+1))/h;
tabela = diff(tabela);
while not(isempty(tabela))
    len = length(tabela)/2;
    pom = tabela(len)+tabela(len+1);
    tabela = diff(tabela);
    pom = pom+p/k*tabela(len);
    tabela = diff(tabela);
    poly = poly+mul.*pom;
    mul = mul.*((p-k)/(2*k)).*((p+k)/(2*k+1));
    k = k+1;
end
poly = vrednosti(n+1)+(p/2).*poly;
end

function poly=besselPoly(cvorovi,vrednosti,x)
%BESSELPOLY(CVOROVI,VREDNOSTI,X) konstruise Besselov
% interpolacioni polinom, argument CVOROVI odredjuje
% interpolacione cvorove, sa vrednostima funkcije VREDNOSTI
% u tacki X

k = 0; poly = 0; mul = 1;
n = (length(cvorovi)-2)/2;
tabela = vrednosti;
h = (cvorovi(end)-cvorovi(1))/(2*n+1);
p = (x-cvorovi(n+1))/h;
while not(isempty(tabela))
    len = length(tabela)/2;
    pom = (tabela(len)+tabela(len+1))/2;
    tabela = diff(tabela);
    pom = pom+(p-.5)/(2*k+1)*tabela(len);
    poly = poly+mul.*pom;
    k = k+1;
    mul = mul.*((p+k-1)/(2*k)).*((p-k)/(2*k-1));
    tabela = diff(tabela);
end
end
```

Problemi gubitka značajnih cifara interpolanta, koji se javljaju prilikom povećanja broja interpolacionih čvorova, postoje i u ovom slučaju.

4.4 Hermiteov interpolant

Problem interpolacije se može posmatrati i opštije. Neka su dati interpolacioni čvorovi x_ℓ, $\ell = 0, \ldots, k$, zajedno sa nizom prirodnih brojeva m_ℓ, $\ell = 0, \ldots, k$. Pretpostavljaćemo da su interpolacioni čvorovi zadati u rastućem poretku $x_0 < x_1 < \cdots < x_k$. Niz brojeva m_ℓ, $\ell = 0, \ldots, k$, nazivamo višestrukostima čvorova. Za zbir višestrukosti čvorova m_ℓ, $\ell = 0, \ldots, k$, umanjen za jedan, uvodimo specijalnu oznaku $n = -1 + \sum_{\ell=0}^{k} m_\ell$.

Hermiteov interpolacioni problem možemo predstaviti na sledeći način: za zadate interpolacione čvorove x_ℓ, $\ell = 0, \ldots, k$, sa višestrukostima m_ℓ, $\ell = 0, \ldots, k$, odrediti polinom H_n, stepena n, tako da su ispunjeni uslovi

$$H_n^{(j)}(x_\ell) = f_\ell^j, \quad j = 0, \ldots, m_\ell - 1, \quad \ell = 0, \ldots, k. \tag{4.8}$$

Vrednosti f_ℓ^j, $j = 0, \ldots, m_\ell - 1$, $\ell = 0, \ldots, k$, mada mogu biti proizvoljne, često se interpretiraju kao j-ti izvodi funkcije f u tačkama x_ℓ, $f_\ell^j = f^{(j)}(x_\ell)$, $j = 0, \ldots, m_\ell - 1$, $\ell = 0, \ldots, k$.

Na primer, jedan jednostavniji slučaj Hermiteovog interpolacionog problema bi bio sledeći

$$H_2(0) = 0, \quad H_2(1) = 1, \quad H_2'(1) = 0.$$

U ovom problemu interpolacioni čvorovi su $x_0 = 0$ i $x_1 = 1$, sa višestrukostima $m_0 = 1$ i $m_1 = 2$, $n = -1 + 1 + 2 = 2$. Kako je H_2 polinom drugog stepena možemo relativno jednostavno odrediti rešenje. Naime, neka je $H_2(x) = h_0 + h_1 x + h_2 x^2$, onda je

$$0 = H_2(0) = h_0, \quad 1 = H_2(1) = h_0 + h_1 + h_2, \quad 0 = H_2'(1) = h_1 + 2h_2.$$

Kao što vidimo, dobili smo sistem linearnih jednačina po nepoznatim koeficijentima polinoma H_2. Rešenje sistema je jedinstveno i rešavanjem sistema se dobija polinom

$$H_2(x) = 2x - x^2.$$

Postavlja se pitanje da li svaki Hermiteov interpolacioni problem ima jedinstveno rešenje. Odgovor na ovo pitanje je pozitivan i dat je u sledećoj teoremi.

Teorema 4.13 *Hermiteov interpolacioni problem* (4.8) *ima jedinstveno rešenje.*

Dokaz. Dokaz se zasniva na argumentima iz linearne algebre. Naime, Hermiteov interpolacioni problem možemo da predstavimo kao sistem linearnih jednačina po nepoznatim koeficijentima Hermiteovog interpolacionog polinoma, kao u prethodnom primeru. Ako prebrojimo jednačine vidimo da ih ima upravo $n+1$, koliko ima i koeficijenata Hermiteovog interpolanta. Dakle, imamo kvadratni sistem linearnih jednačina.

Posmatrajmo odgovarajući homogeni sistem. Kod ovog Hermiteovog interpolacionog problema je $f_\ell^j = 0$, $j = 0, \ldots, m_\ell - 1$, $\ell = 0, \ldots, k$ i problem ima formu

$$H_n^{(j)}(x_\ell) = 0, \quad j = 0, \ldots, m_\ell - 1, \quad \ell = 0, \ldots, k.$$

Jedno rešenje ovog interpolacionog problema je polinom $H_n = 0$. Ovo rešenje nazivamo trivijalnim. Potražimo netrivijalno rešenje ovog homogenog sistema. Vidimo da u tački x_ℓ polinom H_n ima nulu reda m_ℓ. Pa zaključujemo da polinom H_n, mora imati $\sum_{\ell=0}^{k} m_\ell = n+1$ nula. Drugim rečima, polinom H_n mora biti stepena $n+1$. Ovo je nemoguće jer mi tražimo rešenje u skupu polinoma ne većeg stepena od n. Zaključujemo da je jedino rešenje $H_n = 0$. Znači da sistem linearnih jednačina

ima invertibilnu matricu, jer je jedino rešenje homogene jednačine nula vektor. Kako je matrica sistema invertibilna, rešenje sistema linearnih jednačina sa tom matricom kao matricom sistema je jedinstveno. □

Za razliku od dokaza egzistencije Lagrangeovog interpolanta koji je konstruktivan, dokaz egzistencije Hermiteovog interpolanta je samo egzistencijalan. Nismo dobili mogućnost konstrukcije iz dokaza, izuzev metoda konstrukcije koji se zasniva na čisto matričnim metodima. Kako je poznato ovaj način konstrukcije interpolanta je numerički nestabilan.

Postoji mogućnost konstrukcije Hermiteovog interpolanta koja koristi konstrukciju zasnovanu na konstrukciji Lagrangeovog interpolanta. Prezentovaćemo ovaj metod na jednom primeru. Konstruisaćemo Hermiteov interpolant za skup podataka zadat tabelom 4.2. Očigledno su interpola-

k	x_k	$f(x_k)$	$f'(x_k)$	$f''(x_k)$
0	0	1	0	
1	2	0	1	2
2	3	1	-1	

Tabela 4.2: Podaci za konstrukciju Hermiteovog interpolanta

cioni čvorovi $x_0 = 0$, $x_1 = 2$ i $x_2 = 3$, sa višestrukostima $m_0 = 2$, $m_1 = 3$ i $m_2 = 2$, redom. Odavde zaključujemo da tražimo polinom ne većeg stepena od $n = -1 + 2 + 3 + 2 = 6$. Potražićemo ga u obliku

$$H_6(x) = P_2(x) + \omega(x)H_3(x), \tag{4.9}$$

gde je $\omega(x) = (x-0)(x-2)(x-3)$. Ideja je u sledećem, očigledno je $H_6(x_\ell) = P_2(x_\ell)$, $\ell = 0, 1, 2$, pa polinom P_2 možemo da konstruišemo koristeći Lagrangeov interpolant. Dobijamo

$$P_2(x) = 1\frac{(x-2)(x-3)}{(0-2)(0-3)} + 0\frac{(x-0)(x-3)}{(2-0)(2-3)} + 1\frac{(x-0)(x-2)}{(3-0)(3-2)} = \frac{1}{2}x^2 - \frac{3}{2}x + 1.$$

Ostaje nam da odredimo polinom H_3. Primetimo da je stepen ovog polinoma 3, jer je stepen polinoma ω tri i stepen polinoma H_6 je šest.

Međutim, ako diferenciramo jednakost (4.9) u tački x_ℓ, $\ell = 0, 1, 2$, nalazimo

$$f'(x_\ell) = H_6'(x_\ell) = P_2'(x_\ell) + \omega'(x_\ell)H_3(x_\ell) + \omega(x_\ell)H_3'(x_\ell) = P_2'(x_\ell) + \omega'(x_\ell)H_3(x_\ell).$$

Odavde nalazimo

$$H_3(x_\ell) = \frac{f'(x_\ell) - P_2'(x_\ell)}{\omega'(x_\ell)},$$

pa možemo izračunati

$$H_3(0) = \frac{1}{4}, \quad H_3(2) = -\frac{1}{4}, \quad H_3(3) = -\frac{5}{6}.$$

Ako diferenciramo dva puta jednakost (4.9), nalazimo

$$\begin{aligned} f''(x_\ell) &= H_6''(x_\ell) = P_2''(x_\ell) + \omega''(x_\ell)H_3(x_\ell) + 2\omega'(x_\ell)H_3'(x_\ell) + \omega(x_\ell)H_3''(x_\ell) \\ &= P_2''(x_\ell) + \omega''(x_\ell)H_3(x_\ell) + 2\omega'(x_\ell)H_3'(x_\ell). \end{aligned}$$

Odavde dobijamo

$$H_3'(x_\ell) = \frac{f''(x_\ell) - P_2''(x_\ell) - \omega''(x_\ell)H_3(x_\ell)}{2\omega''(x_\ell)}.$$

Na osnovu raspoloživih podataka važi

$$H_3'(2) = \frac{f''(2) - P_2''(2) - \omega''(2)H_3(2)}{2\omega'(2)} = -\frac{3}{8}.$$

Praktično smo dobili novi Hermiteov interpolacioni problem za polinom H_3, koji je predstavljen pregledno u tabeli 4.3. Postupamo na isti način kao u prethodnom slučaju.

k	x_k	$H_3(x_k)$	$H_3'(x_k)$
0	0	1/4	
1	2	−1/4	−3/8
2	3	−5/6	

Tabela 4.3: Hermiteov interpolacioni problem za polinom H_3

$$H_3(x) = P_2(x) + \omega(x)H_0(x). \tag{4.10}$$

Lagrangeov interpolant se izračunava jednostavno

$$P_2(x) = \frac{1}{4}\frac{(x-2)(x-3)}{(0-2)(0-3)} + \left(-\frac{1}{4}\right)\frac{(x-0)(x-3)}{(2-0)(2-3)} + \left(-\frac{5}{6}\right)\frac{(x-0)(x-2)}{(3-0)(3-2)} = -\frac{1}{9}x^2 - \frac{1}{36}x + \frac{1}{4}.$$

Diferenciramo li jednakost (4.10), nalazimo

$$H_3'(2) = P_2'(2) + \omega'(2)H_0(2), \quad H_0(2) = \frac{H_3'(2) - P_2'(2)}{\omega'(2)} = -\frac{7}{144}.$$

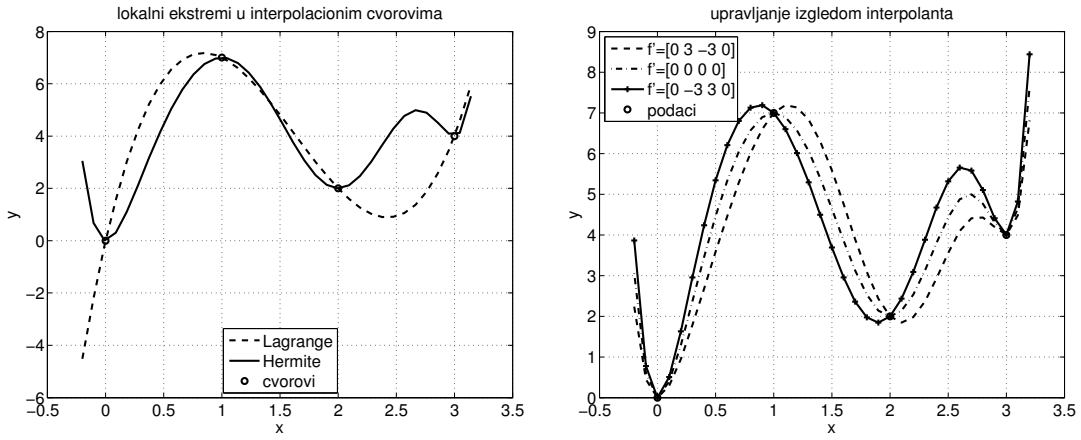
Polinom H_0 je stepena nula, pa je prema tome konstantan, i samim tim određen vrednošću u jednoj tački.

Konačno nalazimo

$$H_6(x) = \frac{1}{2}x^2 - \frac{3}{2}x + 1 + x(x-2)(x-3)\left(-\frac{1}{9}x^2 - \frac{1}{36}x + \frac{1}{4} - \frac{7}{144}x(x-2)(x-3)\right).$$

Ovaj metod je prilično neupotrebljiv kada se želi konstruisati polinom višeg stepena, ili kada se želi napisati program koji konstruiše Hermiteov interpolant. Zato se problemom konstrukcije bavimo detaljnije u narednim sekcijama.

Hermiteov interpolant postoji iz potrebe da se svi podaci koji su poznati o funkciji upotrebe pri konstrukciji interpolacionog polinoma. Međutim, postoji jedno svojstvo Hermiteovog interpolanta koje ga čini superiornim u odnosu na Lagrangeov interpolacioni polinom. Pretpostavimo da želimo da konstruišemo interpolant koji zadovoljava interpolacione uslove $\{(0,0),(1,7),(2,2),(3,4)\}$. Ako problem rešavamo Lagrangeovom interpolacijom rešenje je jednoznačno. Međutim, ako problem rešavamo Hermiteovom interpolacijom broj rešenja je beskonačan. Slika 4.13, levo, ilustruje Lagrangeov i Hermiteov interpolant za posmatrani problem, pri čemu je Hermiteov interpolant konstruisan tako da je u svim interpolacionim čvorovima vrednost prvog izvoda jednaka nula. Drugim rečima, dobili smo interpolacioni polinom koji u interpolacionim čvorovima ima lokalne ekstreme. Naravno, u ovom slučaju se Hermiteov interpolant konstruiše sa vrednostima prvog izvoda jednakim nula u svim interpolacionim čvorovima.



Slika 4.13: Slika levo ilustruje mogućnost konstrukcije interpolanta koji ima lokalne ekstreme u interpolacionim čvorovima, slika desno ilustruje tri Hermiteova interpolanta sa izvodima u interpolacionim čvorovima određenim sa $f' = [0, 3, -3, 0]$, $f' = [0, 0, 0, 0]$ i $f' = [0, -3, 3, 0]$.

Mogućnost izbora prvog izvoda omogućava uticaj na izgled grafika interpolanta. Ova ideja, kada se razvije do kraja kroz teoriju splajnova, omogućava potpunu kontrolu nad izgledom interpolanta. Na slici 4.13, desno, dati su grafici tri Hermiteova interpolanata sa različitim vrednostima prvog izvoda da bi se stekao utisak o ideji upravljanja izgledom interpolanta. Vidimo da je upravljanje izgledom moguće, a razvoj ideje vodi ka disciplini koja se obično naziva kompjutersko modelovanje.

4.4.1 Newtonova forma Hermiteovog interpolanta

Da bismo mogli da odredimo Hermiteov interpolant u formi Newtonovog interpolacionog polinoma, potrebno je modifikovati definiciju podeljenih razlika tako da je dopušteno ponavljanje čvorova. Sa ovakvom definicijom podeljenih razlika Hermiteov interpolant može se shvatiti kao Newtonov interpolacioni polinom sa interpolacionim čvorovima koji se mogu ponavljati.

Neka su x_ℓ, $\ell = 0, \ldots, k$, interpolacioni čvorovi zadati u rastućem poretku, sa višestrukostima m_ℓ, $\ell = 0, \ldots, k$. Formirajmo novi niz interpolacionih čvorova z_j, $j = 0, \ldots, n$, $n = -1 + \sum_{\ell=0}^{k} m_\ell$, na sledeći način

$$(z_0, \ldots, z_n) = (\underbrace{x_0, \ldots, x_0}_{m_0}, \underbrace{x_1, \ldots, x_1}_{m_1}, \ldots, \underbrace{x_k, \ldots, x_k}_{m_k}),$$

gde se u vektoru na desnoj strani jednakosti svaki od interpolacionih čvorova x_ℓ, $\ell = 0, \ldots, k$, javlja onoliko puta kolika je njegova višestrukost m_ℓ. Recimo, x_0 se javlja m_0 puta, x_1 se javlja m_1 puta i tako redom.

Nama je cilj da definišemo niz podeljenih razlika za niz z_j, $j = 0, \ldots, n$. Problem koji se ovde javlja je ponavljanje elemenata. Naime, koje je značenje simbola $[z_0, z_0; f]$. Koristeći ideju neprekidnosti, vrednost ovog simbola može se interpretirati na sledeći način

$$[z_0, z_0; f] = \lim_{z \to z_0} [z_0, z; f] = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0} = f'(z_0).$$

Pokazuje se da je upravo ovakva interpretacija značenja simbola $[z_0, z_0; f]$ pogodna za konstrukciju Hermiteovog interpolanta. U generalnom slučaju simbol $[z_0, \ldots, z_0; f]$, gde se z_0 ponavlja m puta,

se može interpretirati rekurzivno

$$[z_0,\dots,z_0;f]=\lim_{z\to z_0}\frac{[z_0,\dots,z_0,z;f]-[z_0,\dots,z_0;f]}{z-z_0},$$

pri čemu se u prvom simbolu $[z_0,\dots,z_0,z;f]$, na desnoj strani jednakosti, z_0 javlja $m-2$ puta, dok se u simbolu $[z_0,\dots,z_0;f]$, na desnoj strani jednakosti, javlja $m-1$ puta.

Sledeća teorema je generalizacija teoreme 4.8 na slučaju podeljenih razlika sa ponavljanjem.

Teorema 4.14 *Važi sledeća rekurentna relacija*

$$[z_0,\dots,z_m;f]=\begin{cases}\dfrac{[z_1,\dots,z_m;f]-[z_0,\dots,z_{m-1};f]}{z_m-z_0}, & z_m\neq z_0,\\ \dfrac{f^{(m)}(z_0)}{m!}, & z_m=z_0.\end{cases}$$

Koristeći pojam podeljenih razlika sa ponavljanjem, možemo formulisati Newtonovu formu Hermiteovog interpolanta.

Teorema 4.15 (Newtonova forma Hermiteovog interpolanta) *Neka su x_ℓ, $\ell=0,\dots,k$, interpolacioni čvorovi sa višestrukostima m_ℓ, $\ell=0,\dots,k$, i neka su z_j, $j=0,\dots,n$, interpolacioni čvorovi sa ponavljanjima. Newtonov interpolacioni polinom*

$$H_n(x)=\sum_{j=0}^{n}[z_0,\dots,z_j;f]\omega_j(x),\quad \omega_j(x)=\prod_{i=0}^{j-1}(x-z_j),\quad j=1,\dots,n,\quad \omega_0(x)=1,$$

je rešenje Hermiteovog interpolacionog problema

$$H_n^{(j)}(x_\ell)=f^{(j)}(x_\ell),\quad j=0,\dots,m_\ell-1,\quad \ell=0,\dots,k.$$

Ilustrujmo ovu teoremu na primeru konstrukcije Hermiteovog interpolanta za skup podataka datih u tabeli 4.2. Sledeća tabela podeljenih razlika sadrži potrebnu konstrukciju.

k	z_k	d_k^0f	d_k^1f	d_k^2f	d_k^3f	d_k^4f	d_k^5f	d_k^6f
0	0	$\underline{1}$						
			$\underline{f'(0)=0}$					
1	0	1		$\underline{-1/4}$				
			$-1/2$		$\underline{1/2}$			
2	2	0		$3/4$		$\underline{-3/16}$		
			$f'(2)=1$		$1/8$		$\underline{-1/16}$	
3	2	0		$\dfrac{f''(2)}{2}=1$		$-3/8$		$\underline{-7/144}$
			$f'(2)=1$		-1		$-5/24$	
4	2	0		0		-1		
			1		-2			
5	3	1		-2				
			$f'(3)=-1$					
6	3	1						

U ovoj tabeli, zbog prostornog ograničenja koristili smo oznake $d_k^if=[z_k,\dots,z_{k+i};f]$, $i=0,\dots,6$. U ovom slučaju interpolacioni čvorovi su $x_0=0$, $x_1=2$ i $x_2=3$, sa višestrukostima $m_0=2$,

$m_1 = 3$ i $m_2 = 2$, redom. Odgovarajući interpolacioni čvorovi sa ponavljanjem su $z_0 = z_1 = 0$, $z_2 = z_3 = z_4 = 2$ i $z_5 = z_6 = 3$. Jedino na šta treba obratiti pažnju je popunjavanje onih delova tabele gde umesto standardnog načina konstrukcije treba upisivati odgovarajuće vrednosti izvoda. Ostalo je sve potpuno isto kao i pri konstrukciji tabele podeljenih razlika koju smo već imali prilike da konstruišemo pri konstrukciji Newtonovog interpolacionog polinoma. Mesta na kojima treba upisivati vrednosti izvoda u tabeli su istaknuta.

Podaci iz tabele koji su potrebni za konstrukciju Hermiteovog interpolacionog polinoma su podvučeni. Na osnovu podataka iz tabele, dobijamo

$$H_6(x) = 1 + 0(x-0) + \left(-\frac{1}{4}\right)(x-0)^2 + \frac{1}{2}(x-0)^2(x-2) + \left(-\frac{3}{16}\right)(x-0)^2(x-2)^2 + \left(-\frac{1}{16}\right)(x-0)^2(x-2)^3 + \left(-\frac{7}{144}\right)(x-0)^2(x-2)^3(x-3).$$

4.4.2 Greška Hermiteovog interpolanta

Iz Newtonove forme Hermiteovog interpolanta može se zaključiti da je teorema koja opisuje grešku Hermiteovog interpolanta slična onoj koja opisuje grešku Lagrangeovog interpolanta. U stvari teorema je identična, ali je treba primeniti nad nizom ponovljenih interpolacionih tačaka z_ℓ, $\ell = 0, \ldots, n$.

Teorema 4.16 (Greška Hermiteovog interpolanta) *Neka su $x_\ell \in [a,b]$, $\ell = 0, \ldots, k$, interpolacioni čvorovi, višestrukosti m_ℓ, $\ell = 0, \ldots, k$, i neka je $n = -1 + \sum_{\ell=0}^{k} m_\ell$. Neka je H_n Hermiteov interpolant funkcije $f \in C^{n+1}[a,b]$, konstruisan za interpolacione uslove*

$$H_n^{(j)}(x_\ell) = f^{(j)}(x_\ell), \quad j = 0, \ldots, m_\ell - 1, \quad \ell = 0, \ldots, k.$$

Za svako $x \in [a,b]$ postoji $\psi \in (\min\{x, x_0, \ldots, x_k\}, \max\{x, x_0, \ldots, x_k\})$, tako da važi

$$f(x) - H_n(x) = \frac{f^{(n+1)}(\psi)}{(n+1)!} \prod_{\ell=0}^{k} (x - x_\ell)^{m_\ell}.$$

Obično se čvorni polinom u ovom slučaju označava sa Ω, da bi se istakla činjenica da se radi o višestrukim interpolacionim čvorovima. Međutim, niz ponovljenih interpolacionih čvorova daje formalne argumente da i u ovom slučaju čvorni polinom označavamo kao i do sada slovom ω. Analiza izraza za grešku ostaje ista kao i slučaju Lagrangeovog interpolanta.

4.4.3 Hermiteov interpolant u Lagrangeovom obliku

Pod Lagrangeovom formom Hermiteovog interpolatna podrazumevamo konstrukciju Lagrangeovog bazisa $L_{\ell\nu}$, $\nu = 0, \ldots, m_\ell - 1$, $\ell = 0, \ldots, k$, koji bi za zadate interpolacione čvorove x_ℓ, $\ell = 0, \ldots, k$, sa višestrukostima m_ℓ, $\ell = 0, \ldots, k$, zadovoljavao sledeće uslove

$$L_{\ell\nu}^{(j)}(x_\mu) = \begin{cases} 1, & \ell = \mu \text{ i } j = \nu, \\ 0, & \ell \neq \mu \text{ ili } j \neq \nu, \end{cases} \qquad j, \nu = 0, \ldots, m_\ell - 1, \quad \mu, \ell = 0, \ldots, k.$$

Nije teško proveriti da se poznavanjem ovih polinoma, Hermiteov interpolant može izraziti na sledeći način

$$H_n(x) = \sum_{\ell=0}^{k} \sum_{\nu=0}^{m_\ell - 1} f^{(\nu)}(x_\ell) L_{\ell\nu}(x).$$

Označimo sa $\Omega(x)=\prod_{\ell=0}^{k}(x-x_\ell)^{m_\ell}$ čvorni polinom Hermiteove interpolacije. Lagrangeov bazis Hermiteove interpolacije se može eksplicitno izraziti koristeći sledeću teoremu.

Teorema 4.17 (Lagrangeov bazis Hermiteove interpolacije) *Neka su x_ℓ, $\ell=0,\dots,k$, interpolacioni čvorovi višestrukosti m_ℓ, $\ell=0,\dots,k$, onda je*

$$L_{\ell\nu}(x)=\frac{1}{\nu!}\frac{\Omega(x)}{(x-x_\ell)^{m_\ell-\nu}}\sum_{j=0}^{m_\ell-\nu-1}\frac{1}{j!}\left[\frac{(x-x_\ell)^{m_\ell}}{\Omega(x)}\right]^{(j)}_{x=x_\ell}(x-x_\ell)^j,\quad \nu=0,\dots,m_\ell-1,\quad \ell=0,\dots,j,$$

Lagrangeov bazis Hermiteove interpolacije.

Dokaz. Najelementarniji dokaz može se dati primenom metoda kompleksne integracije i Cauchyeve teoreme o ostacima. Ako pomnožimo prethodnu jednakost sa $1/(\omega-x_\ell)^m$, $m=1,\dots,m_\ell$, i integralimo po zatvorenoj konturi Γ_ℓ koja obuhvata samo interpolacioni čvor x_ℓ, nalazimo

$$\frac{L_{\ell\nu}^{(m-1)}(x_\ell)}{(m-1)!}=\frac{1}{2\pi\mathrm{i}\nu!}\sum_{j=0}^{m_\ell-\nu-1}\frac{1}{j!}\left[\frac{(x-x_\ell)^{m_\ell}}{\Omega(x)}\right]^{(j)}_{x=x_\ell}\oint_{\Gamma_\ell}\frac{\Omega(\omega)}{(\omega-x_\ell)^{m_\ell}}\frac{d\omega}{(\omega-x_\ell)^{-\nu-j+m}}.$$

Nas zanimaju samo vrednosti j za koje važi $-\nu-j+m\geq 1$, jer jedino u tom slučaju podintegralna funkcija ima singularitet unutar konture Γ_ℓ. Ovaj uslov je ispunjen za vrednosti j koje zadovoljavaju nejednakost $-\nu-1+m\geq j$. Koristeći ovaj uslov, nalazimo

$$\begin{aligned}\frac{L_{\ell\nu}^{(m-1)}(x_\ell)}{(m-1)!}&=\frac{1}{\nu!}\sum_{j=0}^{\min\{m-\nu-1,m_\ell-\nu-1\}}\frac{1}{j!}\left[\frac{(x-x_\ell)^{m_\ell}}{\Omega(x)}\right]^{(j)}_{x=x_\ell}\frac{1}{(m-\nu-j-1)!}\left[\frac{\Omega(x)}{(x-x_\ell)^{m_\ell}}\right]^{(m-\nu-j-1)}_{x=x_\ell}\\&=\frac{1}{\nu!(m-\nu-1)!}\sum_{j=0}^{m-\nu-1}\binom{m-\nu-1}{j}\left[\frac{(x-x_\ell)^{m_\ell}}{\Omega(x)}\right]^{(j)}_{x=x_\ell}\left[\frac{\Omega(x)}{(x-x_\ell)^{m_\ell}}\right]^{(m-\nu-j-1)}_{x=x_\ell}\\&=\frac{1}{\nu!(m-\nu-1)!}\left[\frac{(x-x_\ell)^{m_\ell}}{\Omega(x)}\frac{\Omega(x)}{(x-x_\ell)^{m_\ell}}\right]^{(m-\nu-1)}_{x=x_\ell}=\begin{cases}0, & m\neq\nu+1,\\ \dfrac{1}{\nu!}, & m=\nu+1.\end{cases}\end{aligned}$$

Zaključujemo da važi $L_{\ell\nu}^{(m-1)}(x_\ell)=\delta_{\nu,m-1}$.

Za tačku x_j, $j\neq\ell$, vidimo da je nula reda m_j polinoma $L_{\ell\nu}$, pa mora biti ispunjen uslov $L_{\ell\nu}^{(m)}(x_j)=0$, $m=0,\dots,m_j-1$. □

Eksplicitna konstrukcija Lagrangeovog bazisa za Hermiteovu interpolaciju daje nam mogućnost konstrukcije Hermiteovog interpolacionog polinoma.

Teorema 4.18 (Lagrangeova forma Hermiteovog interpolanta) *Neka su x_ℓ, $\ell=0,\dots,k$, interpolacioni čvorovi višestrukosti m_ℓ, $\ell=0,\dots,k$, $n=-1+\sum_{\ell=0}^{k}m_\ell$. Interpolacioni problem*

$$H_n^{(\nu)}(x_\ell)=f^{(\nu)}(x_\ell),\quad \nu=0,\dots,m_\ell-1,\quad \ell=0,\dots,k,$$

ima rešenje

$$H_n(x)=\sum_{\ell=0}^{n}\sum_{\nu=0}^{m_\ell-1}f^{(\nu)}(x_\ell)L_{\ell\nu}(x),$$

gde je $L_{\ell\nu}$, $\nu=0,\dots,m_\ell-1$, $\nu=0,\dots,k$, Lagrangeov bazis Hermiteove interpolacije.

Iako je ova teorema konstruktivna, jer dobijamo eksplicitni izraz kojim je određen Hermiteov interpolant, ipak ne dobijamo izraz na osnovu koga možemo da izvršimo konstrukciju Hermiteovog interpolanta u Lagrangeovom obliku. Naime, problem predstavlja određivanje izvoda polinoma $\Omega(x)/(x-x_\ell)^{m_\ell}$, $\ell = 0, \ldots, k$.

U slučaju da svi interpolacioni čvorovi imaju višestrukost dva, može se dati još jednostavniji izraz za Hermiteov interpolant u Lagrangeovom obliku. U ovom slučaju možemo izraziti rezultat u formi Lagrangeovog bazisa iz leme 4.1 i dobiti izraz za Hermiteov interpolant koji je pogodan za numeričku konstrukciju.

Teorema 4.19 (Lagrangeova forma Hermiteovog interpolanta višestrukosti dva) *Neka su x_ℓ, $\ell = 0, \ldots, k$, interpolacioni čvorovi višestrukosti $m_\ell = 2$, $\ell = 0, \ldots, k$. Interpolacioni problem*

$$H_{2k-1}^{(j)}(x_\ell) = f^{(j)}(x_\ell), \quad j = 0, 1, \quad \ell = 0, \ldots, k,$$

ima rešenje

$$H_{2k-1}(x) = \sum_{\ell=0}^{k} f(x_\ell)(1 + 2L'_\ell(x_\ell)(x_\ell - x))L_\ell^2(x) + \sum_{\ell=0}^{k} f'(x_\ell)(x - x_\ell)L_\ell^2(x),$$

gde je L_ℓ, $\ell = 0, \ldots, k$, Lagrangeov bazis iz leme 4.1 konstruisan za interpolacione čvorove x_ℓ, $\ell = 0, \ldots, k$.

Dokaz. Sve vreme u dokazu koristimo osobine Lagrangeovog bazisa iz leme 4.1. Očigledno je

$$H_{2k-1}(x_\nu) = \sum_{\ell=0}^{k} f(x_\ell)(1 + 2L'_\ell(x_\ell)(x_\ell - x_\nu))L_\ell^2(x_\nu) + \sum_{\ell=0}^{k} f'(x_\ell)(x_\nu - x_\ell)L_\ell^2(x_\nu) = f(x_\nu),$$

za $\nu = 0, \ldots, k$.

Ako potražimo izvod polinoma H_{2k-1} nalazimo

$$\begin{aligned} H'_{2k-1}(x) = &\sum_{\ell=0}^{k} f(x_\ell)\left(-2L'_\ell(x_\ell)L_\ell^2(x) + (1 + 2L'_\ell(x_\ell)(x_\ell - x))2L'_\ell(x)L_\ell(x)\right) \\ &+ \sum_{\ell=0}^{k} f'(x_\ell)(L_\ell^2(x) + (x - x_\ell)2L'_\ell(x)L_\ell(x)). \end{aligned}$$

Ako primenimo prethodnu formulu na tačku x_ν, nalazimo

$$\begin{aligned} H'_{2k-1}(x_\nu) &= \sum_{\ell=0}^{k} f(x_\ell)\left(-2L'_\ell(x_\ell)L_\ell^2(x_\nu) + (1 + 2L'_\ell(x_\ell)(x_\ell - x_\nu))2L'_\ell(x_\nu)L_\ell(x_\nu)\right) \\ &\quad + \sum_{\ell=0}^{k} f'(x_\ell)(L_\ell^2(x_\nu) + (x_\nu - x_\ell)2L'_\ell(x_\nu)L_\ell(x_\nu)) \\ &= f(x_\nu)(-2L'_\nu(x_\nu) + 2L'_\nu(x_\nu)) + f'(x_\nu) = f'(x_\nu). \end{aligned}$$

Ovim smo završili dokaz teoreme. □

Ova teorema omogućava konstrukciju Hermiteovog interpolanta u Lagrangeovoj formi za specijalan slučaj interpolacionih podataka. Konkrektno, u slučaju interpolacionih čvorova višestrukosti dva. Teorema ima i teorijski značaj, jer se koristi pri dokazu nekih drugih važnih teorema u numeričkoj analizi.

4.4.4 Numerička konstrukcija Hermiteovog interpolanta

Metodi izloženi u prethodne dve sekcije, za konstrukciju Hermiteovog interpolanta, su povoljni za implementaciju na računskim mašinama. Na primer, konstrukcija u formi Newtonovog interpolanta može biti implementirana u Matlabu na sledeći način

```
function [ res ] = hermitePolyNewton( cv, mn, vr, x )
%HERMITEPOLY(CV,MN,VR,X) izracunava vrednost u tacki X
% Hermiteovog interpolanta u interpolacionim cvorovima CV,
% sa mnogostrukostima MN i vrednostima funkcije VR,
% vrednosti funkcije su zadate u obliku matrice
% vrste matrice izgledaju ovako f(cv(i)) f'(cv(i)) ... f^{(m)}(cv(i))
% gde je m=max(mn)

m = mn;k = -1+length(m);
n = -1+sum(m);z = [];
tabela = [];jedinice = ones(1,k+1);
for i=1:(k+1)
    z = [z cv(i)*ones(1,m(i))];
    tabela = [tabela vr(i,1)*ones(1,m(i))];
end

res = tabela(1);omega = 1;fakt = 1;
for i=1:n
    m = m-jedinice;pos = 1;
    fakt = fakt*i;j = 1;
    while j<=n+1-i
       if z(j+i)-z(j) == 0
           while m(pos)<1
               pos = pos+1;
           end
           while z(j+i)-z(j) == 0
               tabela(j) = vr(pos,i+1)/fakt;
               j = j+1;
               if j+i > length(z)
                   break;
               end
           end
           pos = pos+1;
       else
           tabela(j) = (tabela(j+1)-tabela(j))/(z(j+i)-z(j));
           j = j+1;
       end
    end
    omega = omega.*(x-z(i));
    res = res+tabela(1)*omega;
end
end
```

Konstrukcija u Lagrangeovoj formi data teoremom 4.19 može se implementirati na sledeći način

```
function [ res ] = hermitePolyLagrange2( cvorovi, vrednosti, x )
%HERMITEPOLYLAGRANGE2(CVOROVI,VREDNOSTI,X) izracunava
% vrednost Hermiteovog interpolanta za cvorove CVOROVI
% visestrukosti 2, sa vrednostima VREDNOSTI koja je matrica
% koja po vrstama ima vrednosti (f(x_i),f'(x_i))
% u tacki X koja moze biti vektor kolona

n = length(cvorovi);
m = length(x);
res = 0;
for ell = 1:n
    prod = 1;
    sum = 0;
    prodX = ones(m,1);
    for j = [1:ell-1 ell+1:n]
        sum = sum+1/(cvorovi(ell)-cvorovi(j));
        prodX = (x-cvorovi(j))/(cvorovi(ell)-cvorovi(j)).*prodX;
    end
    pom = vrednosti(ell,1)*(1+2*sum*(cvorovi(ell)-x)).*prodX.^2;
    res = res+pom+vrednosti(ell,2)*(x-cvorovi(ell)).*prodX.^2;
end
end
```

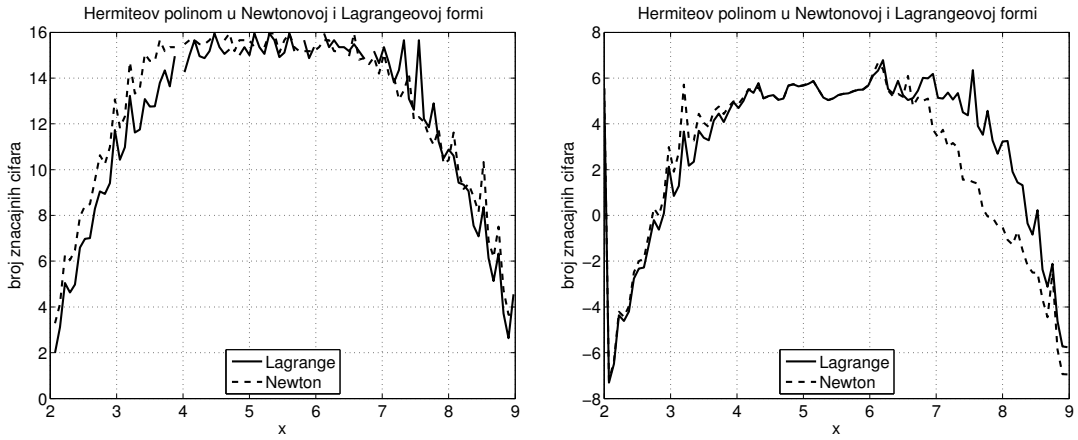
Na primer, ako želimo da konstruišemo Hermietov interpolant za interpolacione uslove zadate tabelom 4.2, funkciju `hermitePolyNewton` pozivamo na sledeći način

```
>>y=hermitePolyNewton([0 2 3],[2 3 2],[1 0 0; 0 1 2; 1 -1 2],(0:.5:3)')
y =
     1.000000000000000e+00
     5.947265625000000e-01
     2.777777777777778e-02
    -2.333984375000000e-01
                         0
     6.771918402777778e-01
     1.000000000000000e+00
```

Implementirana funkcija za konstrukciju Hermiteovog interpolanta u Lagrangeovom obliku se, recimo, može iskoristiti za konstrukciju interpolanta koji ima lokalne ekstreme u interpolacionim čvorovima. Primena funkcije izgleda ovako

```
>>y=hermitePolyLagrange2([-1 1],[1 0; -1 0],(-2:1:2)')
y =
    -1
     1
     0
    -1
     1
```

Za ove interpolacione uslove je jednostavno odrediti Hermitov interpolant koji iznosi $H_3(x) = x(x^2 - 3)/2$.



Slika 4.14: Slika levo prikazuje broj značajnih cifara Hermitevog interpolanta u Newtonovoj i Lagrangeovoj formi pri interpolaciji funkcije log na intervalu $[2, 9]$ sa 20 ekvidistantnih čvorova višestrukosti dva, slika desno ilustruje isto što i slika levo ali podaci imaju gornju granicu relativne greške 10^{-5}.

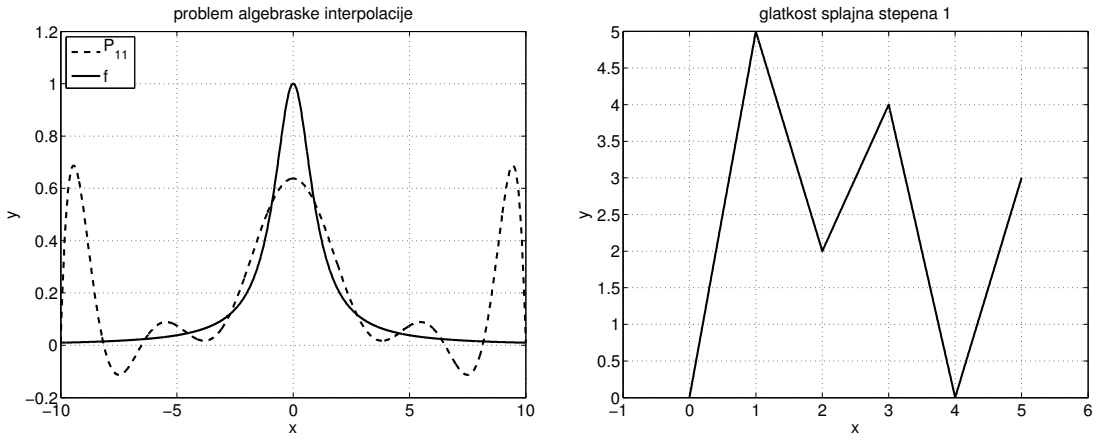
Hermiteov interpolant konstruisan ovim algoritmima ima slično ponašanje kao i interpolacioni polinomi konstruisani u ovoj glavi što se numeričkih karakteristika tiče. Prilikom konstrukcije interpolanta u Newtonovoj formi najveći gubitak značajnih cifara nastaje oko tačke z_n. Previše velika vrednost stepena polinoma dovodi do gubitka značajnih cifara, jer dolazi do efekta prostiranja greške prilikom konstrukcije tabele podeljenih razlika. Konstrukcija u Lagrangeovoj bazi dovodi do poznatih efekata gubitka značajnih cifara na krajevima intervala interpolacije.

Slika 4.14, levo, ilustruje broj značajnih cifara Hermiteovog interpolanta u Newtonovoj i Lagrangeovoj formi pri interpolaciji funkcije log na intervalu $[2, 9]$ sa ekvidistantnim interpolacionim čvorovima. Broj interpolacionih čvorova iznosi 20 i svi čvorovi imaju višestrukost dva. Kao što vidimo Newtonova forma interpolanta je superiorna u okolini tačke z_0, dok u okolini tačke z_n počinje gubitak značajnih cifara. Sa porastom stepena polinoma gubitak značajnih cifara u okolini tačke z_n je sve izraženiji. Slika 4.14, desno, ilustruje ponašanje oba oblika interpolanta Hermiteovog polinoma kad su ulazni podaci zadati sa gornjom granicom relativne greške 10^{-5}. Ponašanje je isto kao i u slučaju konstrukcije Lagrangeovog i Newtonovog interpolanta. Superiornost Newtonove forme nestaje.

4.5 Splajn

Nedostaci algebarske interpolacije mogu se najbolje ilustrovati primerom. Posmatrajmo interpolaciju funkcije $f(x) = 1/(1 + x^2)$, na intervalu $[-10, 10]$, u dvanaest ekvidistantnih interpolacionih čvorova sa $x_0 = -10$ i $x_{11} = 10$. Rezultat interpolacije, zajedno sa grafikom funkcije, je prikazan na slici 4.15, levo.

Problem je što većina korisnika očekuje da interpolant prati grafik funkcije u izvesnoj meri, pogotovu kad je grafik funkcije ovako jednostavan. U ovom slučaju, i mnogim drugim slučajevima,



Slika 4.15: Slika levo ilustruje u kojoj meri algebarski interpolant može odstupati od grafika funkcije, slika desno ilustruje glatkost splajna stepena 1 reda 5.

ne dolazi do željenog ponašanja. Ako bismo nastavili da povećavamo stepen interpolanta došlo bi do sve većeg odstupanja grafika interpolanta od grafika funkcije. Kažemo još da niz interpolacionih polinoma, konstruisanih u ekvidistantim tačkama, divergira za funkciju f. Ovaj primer je u literaturi poznat kao Rungeov primer divergencije niza interpolanata. Na ovom primeru je među prvima dokazano da niz algebarskih interpolanata ne mora da konvergira ka funkciji koju interpolira.

Postaje jasno da povećanje stepena algebarskog interpolanta ne dovodi do povećanja kvaliteta interpolacije neizostavno, u smislu povećanja sličnosti grafika interpolanta i funkcije koja se interpolira. Ovaj problem je doveo do ideje da se zadrži mali stepen polinoma, ali da se podeli interval na kome posmatramo funkciju. Pri tome još dodatno zahtevamo da interpolant ostane glatka funkcija (neprekidno diferencijabilna) dovoljno visokog reda na celom intervalu.

Rešenje ovog problema je nađeno u interpolaciji splajnom. Engleska reč spline (čita se splajn) nema odgovarajući srpski prevod, i mi je dalje u tekstu koristimo kao reč preuzetu iz engleskog jezika.

U ovom odeljku ćemo samo dotaći problem splajna. Ova oblast numeričke analize se razvila i ima veliki broj primena u raznim oblastima nauke i tehnike. Mi ćemo se pre svega baviti problemom motivacije splajna, problemom definicije splajna i problemom konstrukcije kubnog splajna.

Definicija 4.3 (Splajn stepena k reda m) *Neka su dati čvorovi t_i, $i = 0, \ldots, m$, $t_i < t_{i+1}$, $i = 0, \ldots, m-1$. Funkcija s je splajn stepena k reda m, za skup čvorova t_i, $i = 0, \ldots, m$, ako i samo ako je s polinom stepena ne većeg od k na svakom od intervala $[t_i, t_{i+1})$, $i = 0, \ldots, m-1$, i ako s ima neprekidni izvod reda $k-1$ na intervalu $[t_0, t_m]$.*

Kao što vidimo splajn ima zahtevane osobine. Na svakom od intervala $[t_i, t_{i+1})$, $i = 0, \ldots, m-1$, splajn je polinom stepena k, a gledano u celini, na intervalu $[t_0, t_m]$ splajn je neprekidno diferencijabilna funkcija reda $k-1$.

4.5.1 Splajn stepena 1 reda m

Definicija splajna deluje prilično apstraktno. Zato dajemo jednu konstrukciju splajna koja je više nego očigledna. Izaberimo $k = 1$. Za proizvoljan skup čvorova, naša definicija zahteva da

splajn s bude funkcija, koja je polinom stepena ne većeg od 1 na svakom od od intervala $[t_i, t_{i+1})$, $i = 0, \ldots, m-1$. Zaključujemo da je

$$s(x) = \begin{cases} a_0 x + b_0, & x \in [t_0, t_1), \\ \vdots & \vdots \\ a_{m-1} x + b_{m-1}, & x \in [t_{m-1}, t_m). \end{cases} \tag{4.11}$$

Dodatno, zahtevamo da funkcija ima neprekidni izvod reda $k-1=0$ na intervalu $[t_0, t_m]$. Drugim rečima zahtevamo neprekidnost funkcije s. Kako je funkcija s neprekidna na intervalima $[t_i, t_{i+1})$, $i = 0, \ldots, m-1$, jer je na ovim intervalima polinom, zaključujemo da su jedine tačke u kojima treba obezbediti neprekidnost čvorovi t_i, $i = 1, \ldots, m-1$. U ovim tačkama važi

$$\lim_{x \to t_i -} s(x) = a_{i-1} t_i + b_{i-1}, \quad \lim_{x \to t_i +} s(x) = a_i t_i + b_i.$$

Odavde zaključujemo da moraju biti ispunjene jednakosti

$$a_{i-1} t_i + b_{i-1} = a_i t_i + b_i, \quad i = 1, \ldots, m-1., \tag{4.12}$$

pri čemu neprekidnost zahteva i uslov $s(t_m) = a_{m-1} t_m + b_{m-1}$. Međutim, ovaj uslov je više stvar dogovora kako ćemo definisati funkciju s u tački t_m. Drugim rečima, ako funkcija s nije splajn, jer ovaj uslov nije ispunjen, uvek možemo promeniti definiciju funkcije s u čvoru t_m tako da funkcija s postane splajn. Zato ovaj uslov nije od nekog značaja i nećemo ga dalje navoditi.

Zaključujemo da je funkcija s splajn stepena 1, ako i samo ako je ispunjen uslov (4.12). Broj slobodnih parametara u funkciji s iznosi $2m$. Broj jednačina koje parametri treba da zadovolje, da bi funkcija s bila splajn stepena jedan, je $m-1$. Zaključujemo da splajn stepena jedan reda m ima $m+1$ slobodnih parametara. Primetimo da je broj slobodnih parametara jednak broju čvorova.

Poslednje zapažanje nam daje mogućnost da iskoristimo splajn stepena jedan za interpolaciju. Naime, možemo zahtevati da u čvorovima splajn zadovolji interpolacione uslove

$$s(t_i) = f(t_i), \quad i = 0, \ldots, m.$$

Na ovaj način čvorovi splajna postaju jednaki čvorovima interpolacionog problema, što ne mora uvek biti slučaj.

Rešenje problema interpolacije u ovom obliku je dobro poznato i dato sa

$$s(x) = \begin{cases} f(t_0) + \dfrac{f(t_1) - f(t_0)}{t_1 - t_0}(x - t_0), & x \in [t_0, t_1), \\ \vdots & \vdots \\ f(t_{m-1}) + \dfrac{f(t_m) - f(t_{m-1})}{t_m - t_{m-1}}(x - t_{m-1}), & x \in [t_{m-1}, t_m). \end{cases}$$

Ovaj oblik splajna je dobro poznata deo po deo linearna interpolacija.

Splajn stepena jedan, iako predstavlja funkciju koja je glatka na svakom od intervala (t_i, t_{i+1}), $i = 0, \ldots, m-1$, predstavlja grafik funkcije koja gubi osobine glatkosti u čvorovima. Naime, u čvorovima je funkcija samo neprekidna i njen grafik predstavlja izlomljenu krivu liniju. Ova osobina je ilustrovana na slici 4.15, desno. Prikazan je interpolant koji zadovoljava interpolacioni problem

$$s(0) = 0, \quad s(1) = 5, \quad s(2) = 2, \quad s(3) = 4, \quad s(4) = 0, \quad s(5) = 3.$$

Za glatkost višeg reda moramo pokušati konstrukciju splajna višeg stepena. Najčešće se u praksi koristi kubni splajn koji obrađujemo u sledećoj sekciji.

Funkcija `interp1` ima mogućnost konstrukcije splajna stepena 1. Format funkcije koji konstruiše splajn stepena 1 je sledeći

```
y=interp1(c,v,x,'linear')
```

gde su `c` čvorovi splajna, `v` interpolacioni uslovi u čvorovima splajna, a `x` je tačka u kojoj se računa vrednost splajna. Vrednost `y` je izračunata vrednost splajna u vektoru `x`. Slika 4.15, desno, je nacrtana koristeći ovu funkciju na sledeći način

```
>>c=[0 1 2 3 4 5];
>>v=[0 5 2 4 0 3];
>>x=-.1:.01:5.1;
>>y=interp1(c,v,x,'linear');
>>plot(x,y);
```

4.5.2 Kubni splajn

U primenama se najčešće koristi kubni splajn, tj. splajn stepena 3. Mi ćemo se detaljnije pozabaviti konstrukcijom kubnog splajna.

Na osnovu definicije splajna, splajn stepena tri predstavlja polinom stepena tri na svakom od intervala $[t_i, t_{i+1})$, $i = 0, \ldots, m-1$, uz dodatni uslov neprekidnosti drugog izvoda na intervalu $[t_0, t_m]$. Neka je $s(x) = s_i(x)$, $x \in [t_i, t_{i+1})$, $i = 0, \ldots, m-1$, Neprekidnost drugog izvoda, znači postojanje drugog izvoda pa samim tim i neprekidnost prvog izvoda funkcije s i neprekidnost funkcije s. Odavde zaključujemo da mora važiti

$$s_{i-1}(t_i) = s_i(t_i), \quad s'_{i-1}(t_i) = s'_i(t_i), \quad s''_{i-1}(t_i) = s''_i(t_i), \quad i = 1, \ldots, m-1, \tag{4.13}$$

ako želimo da funkcija s predstavlja kubni splajn.

Kako svaki od polinoma s_i, $i = 0, \ldots, m-1$, ima 4 koeficijenta, to imamo ukupno $4m$ slobodnih parametara. Ako želimo da funkcija s bude kubni splajn, funkcija s mora zadovoljiti sistem jednačina (4.13). Jednačina ima ukupno $3(m-1)$. Na osnovu ove analize nalazimo da kubni splajn ima $4m - 3(m-1) = m + 3$ slobodnih parametara.

Ako zahtevamo da kubni splajn zadovolji interpolacione uslove

$$s(t_i) = f(t_i) = f_i, \quad i = 0, \ldots, m,$$

u čvorovima, zaključujemo da nam ostaje ukupno $m + 3 - (m+1) = 2$ slobodna parametra. Interpolant konstruisan kubnim splajnom ima dva slobodna parametra i nije jednoznačan. Postoje različiti načini da se odrede ova dva parametra. Neke od njih razmatraćemo dalje u tekstu.

Označimo $h_i = t_{i+1} - t_i$, $i = 0, \ldots, m-1$ i $z_i = s''(t_i)$, $i = 0, \ldots, m$. Kako je s_i polinom trećeg stepena, to je s''_i polinom prvog stepena. U opštem slučaju možemo pisati

$$s''_i(x) = z_i + \frac{z_{i+1} - z_i}{t_{i+1} - t_i}(x - t_i) = \frac{x - t_i}{h_i} z_{i+1} - \frac{x - t_{i+1}}{h_i} z_i, \quad i = 0, \ldots, m-1.$$

Očigledno su ispunjeni uslovi neprekidnosti drugog izvoda, jer važi

$$z_i = \frac{t_i - t_{i-1}}{h_{i-1}} z_i - \frac{t_i - t_i}{h_{i-1}} z_{i-1} = s''_{i-1}(t_i) = s''_i(t_i) = \frac{t_i - t_i}{h_i} z_{i+1} - \frac{t_i - t_{i+1}}{h_i} z_i = z_i,$$

za $i = 1, \ldots, m-1$.

Ako integralimo jednom, nalazimo

$$s_i'(x) = \frac{(x-t_i)^2}{2h_i} z_{i+1} - \frac{(x-t_{i+1})^2}{2h_i} z_i + c_i, \quad i = 0, \ldots, m-1.$$

Ako integralimo još jednom, i ako malo manipulišemo izrazima, uz činjenicu da je $t_i < t_{i+1}$, $i = 0, \ldots, m-1$, možemo pisati

$$s_i(x) = \frac{(x-t_i)^3}{6h_i} z_{i+1} - \frac{(x-t_{i+1})^3}{6h_i} z_i + c_i(x-t_i) + d_i(x-t_{i+1}), \quad i = 0, \ldots, m-1.$$

Kako mi određujemo interpolacioni kubni splajn, mora važiti

$$s_i(t_i) = f_i, \quad s_i(t_{i+1}) = f_{i+1}, \quad i = 0, \ldots, m-1.$$

Iz ovih uslova dobijamo sistem linearnih jednačina

$$\frac{h_i^2 z_i}{6} - d_i h_i = f_i, \quad \frac{h_i^2 z_{i+1}}{6} + c_i h_i = f_{i+1}, \quad i = 0, \ldots, m-1,$$

odakle možemo odrediti konstante c_i, d_i, $i = 0, \ldots, m-1$, na sledeći način

$$d_i = -\frac{f_i}{h_i} + \frac{z_i h_i}{6}, \quad c_i = \frac{f_{i+1}}{h_i} - \frac{z_{i+1} h_i}{6}, \quad i = 0, \ldots, m-1.$$

Konačno možemo odrediti reprezentaciju svakog polinoma s_i, $i = 0, \ldots, m-1$, na sledeći način

$$s_i(x) = \frac{(x-t_i)^3}{6h_i} z_{i+1} - \frac{(x-t_{i+1})^3}{6h_i} z_i + \left(\frac{f_{i+1}}{h_i} - \frac{z_{i+1} h_i}{6} \right) (x-t_i) - \left(\frac{f_i}{h_i} - \frac{z_i h_i}{6} \right) (x-t_{i+1}).$$

Dobili smo reprezentaciju interpolacionog kubnog splajna s, gde još nismo obezbedili važenje neprekidnosti s'.

Ako diferenciramo funkciju s_i, nalazimo

$$s_i'(x) = \frac{(x-t_i)^2}{2h_i} z_{i+i} - \frac{(x-t_{i+1})^2}{2h_i} z_i + \frac{f_{i+1} - f_i}{h_i} - \frac{z_{i+1} - z_i}{6} h_i.$$

Odavde dobijamo da mora biti ispunjen uslov

$$s_{i-1}'(t_i) = \frac{z_i h_{i-1}}{2} + \frac{f_i - f_{i-1}}{h_{i-1}} - \frac{z_i - z_{i-1}}{6} h_{i-1} = -\frac{z_i h_i}{2} + \frac{f_{i+1} - f_i}{h_i} - \frac{z_{i+1} - z_i}{6} h_i = s_i'(t_i),$$

za $i = 1, \ldots, m-1$.

Preuređenjem, dobijamo sistem linearnih jednačina

$$\frac{h_{i-1}}{6} z_{i-1} + \frac{h_i + h_{i-1}}{3} z_i + \frac{h_i}{6} z_{i+1} = \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}, \quad i = 1, \ldots, m-1. \tag{4.14}$$

Ovo je trodijagonalni sistem linearnih jednačina, gde je broj jednačina $m-1$, dok je broj promenljivih $m+1$, tako da dve vrednosti z_i, $i = 0, \ldots, m$, možemo po volji da izaberemo.

Obično se ostavljaju za izbor vrednosti z_0 i z_m. U ovom slučaju dobijamo trodijagonalni sistem linearnih jednačina

$$\begin{bmatrix} \frac{h_1+h_0}{3} & \frac{h_1}{6} & \dots & 0 \\ \frac{h_1}{6} & \frac{h_2+h_1}{3} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{h_{m-1}+h_{m-2}}{3} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{m-1} \end{bmatrix} = \begin{bmatrix} \frac{f_2-f_1}{h_1} - \frac{f_1-f_0}{h_0} - \frac{h_0}{6} z_0 \\ \frac{f_3-f_2}{h_2} - \frac{f_2-f_1}{h_1} \\ \vdots \\ \frac{f_m-f_{m-1}}{h_{m-1}} - \frac{f_{m-1}-f_{m-2}}{h_{m-2}} - \frac{h_{m-1}}{6} z_m \end{bmatrix}.$$

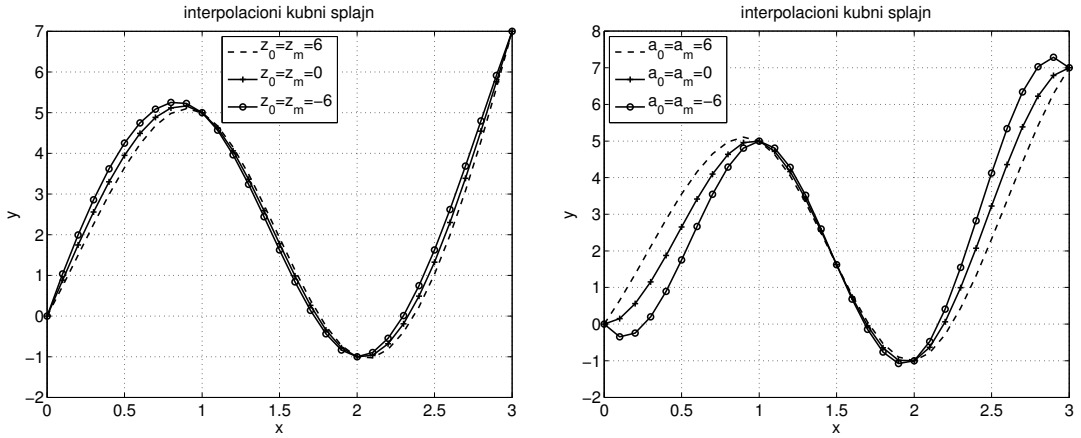
Primetimo da je matrica sistema dijagonalno dominantna, tako da sistem ima jedinstveno rešenje.

Specijalan slučaj $z_0 = z_m = 0$ naziva se prirodni interpolacioni kubni splajn.

Ovaj algoritam konstrukcije interpolacionog kubnog splajna u `Matlab`u implementira funkcija `csape`. Format funkcije koji konstruiše ovaj obik splajna izgleda ovako

```
pp=csape(c,v,'second')
```

Prvi argument `c` predstavlja čvorove splajna, drugi argument `v`, predstavlja interpolacione vrednosti u čvorovima splajna. Vektor `v` ima dva elementa više od vektora `c`. Elementi `v(1)` i `v(end)` predstavljaju vrednosti z_0 i z_m, redom.



Slika 4.16: Slika levo ilustruje interpolacioni kubni splajn za različite vrednosti drugog izvoda u tačkama t_0 i t_m, slika desno ilustruje interpolacioni kubni splajn za razne vrednosti prvog izvoda u tačkama t_0 i t_m.

Prirodni interpolacioni kubni splajn može se dobiti pozivom prethodne funkcije sa vrednostima `v(1)=0` i `v(end)=0`. Međutim, moguće je pozvati funkciju i u sledećem formatu

```
pp=csape(c,v,'variational')
```

U ovom slučaju dolazi do konstrukcije prirodnog interpolacionog kubnog splajna bez obzira na to da li vektor `v` ima istu dužinu kao vektor `c` ili nema. Izlazni argument funkcije `csape` je uvek struktura nad kojom se može primeniti funkcija `ppval` da bi se dobila konkretna vrednost splajna u tački `x`.

Na primer, slika 4.16, levo, sadrži nekoliko različitih interpolacionih kubnih splajnova, konstruisanih za iste interpolacione uslove, sa različitim vrednostima z_0 i z_m. Da bismo dobili jednu krivu sa slike postupamo na sledeći način

```
>>c=[0 1 2 3];
>>v=[6 0 5 -1 7 6];
>>x=0:.1:3;
>>y=ppval(csape(c,v,'second'),x);
>>plot(x,y)
```

Iskoristili smo funkciju `ppval` nad strukturom koju vraća funkcija `csape` kao prvim argumentom, i konkretnim vrednostima `x` u kojima želimo da odredimo vrednosti interpolacionog kubnog splajna.

Ekstremalno svojstvo prirodnog interpolacionog kubnog splajna

Neka je

$$E = \{f \mid f(t_i) = f_i, \quad i = 0, \ldots, m, \quad f \in C^2[t_0, t_m], \quad f''(t_0) = f''(t_m) = 0\}.$$

Primetimo da prirodni interpolacioni kubni splajn s, sa čvorovima u tačkama t_i, $i = 0, \ldots, m$, koji zadovoljava interpolacione uslove

$$s(t_i) = f_i, \quad i = 0, \ldots, m,$$

pripada skupu E. Ovaj splajn je jedinstven i određen konstrukcijom skupa E. Označimo ga sa s_E.

Prirodni interpolacioni kubni splajn je značajan zbog sledeće osobine.

Teorema 4.20 (Ekstremalno svojstvo prirodnog interpolacionog kubnog splajna) *Neka je $f \in E$, onda važi*

$$\int_{t_0}^{t_n} (s_E''(x))^2 dx \leq \int_{t_0}^{t_n} (f''(x))^2 dx.$$

Ova teorema ima mehaničku interpretaciju. Naime, pretpostavimo da treba postaviti uniformno elastično telo, koje zamišljamo kao materijalnu krivu, tako da prolazi kroz tačke u ravni koje su određene interpolacionim uslovima. Potencijalna energija ovog tela je određena kao integral krivine krive koju ovo telo zauzima. Kako je kriva ravna, to potencijalnu energiju možemo odrediti na sledeći način

$$E_p = C \int_{t_0}^{t_m} \frac{(y''(x))^2}{(1 + (y'(x))^2)^3} dx,$$

gde je y kriva koju zauzima telo, a C je konstanta proporcinalnosti. Na osnovu principa minimuma potencijalne energije, kriva koju zauzima telo zadovoljava sledeće ekstremalno svojstvo

$$E_p \leq \int_{t_0}^{t_m} \frac{(f''(x))^2}{(1 + (f'(x))^2)^3} dx,$$

gde je $f \in C^2[t_0, t_m]$ bilo koja funkcija koja zadovoljava interpolacione uslove $f(t_i) = f_i$, $i = 0, \ldots, m$. Ako pretpostavimo da važi $1 + (y'(x))^2 \approx 1$, vidimo da rešenje ekstremalnog problema možemo naći u obliku prirodnog interpolacionog kubnog splajna.

Istorijski se stvar ovako i razvijala. Naime, pre pojave računskih mašina se konstrukcija splajna izvodila koristeći veoma tanko elastično parče drveta koje je postavljano tako da prođe kroz interpolacione uslove. Ovaj tanak elastični komad drveta se nazivao splajn na engleskom jeziku. Danas je konstrukcija pomoću komada drveta nepotrebna, ali se ime splajn zadržalo u upotrebi.

Uslovi zadati prvim izvodom

Uslovi koji se zadaju koristeći vrednosti drugog izvoda u krajnjim tačkama su najjednostavniji za implementaciju, ali nisu jedini koji se koriste. Najčešće se koriste uslovi koji se izražavaju preko vrednosti izvoda $s'(t_0) = a_0$ i $s'(t_m) = a_m$. U ovom slučaju sistem jednačina, pored već postojećih (4.14), uključuje još dve jednačine

$$\frac{h_0}{3}z_0 + \frac{h_0}{6}z_1 = \frac{f_1 - f_0}{h_0} - a_0,$$
$$\frac{h_{m-1}}{6}z_{m-1} + \frac{h_{m-1}}{3}z_m = -\frac{f_m - f_{m-1}}{h_m} + a_m.$$

Kombinujući sa već pomenutim sistemom jednačina (4.14), vidimo da i u ovom slučaju dobijamo trodijagonalni sistem linearnih jednačina. Sistem je i daje dijagonalno dominantan, što garantuje jedinstvenost rešenja.

Funkcija `csape` se može koristiti i za konstrukciju splajna koji je upravo ilustrovan. U ovom slučaju format funkcije je sledeći

```
pp=csape(c,v,'complete')
```

Kao i u slučaju kada se uslovi zadaju za drugi izvod, očekuje se da vektor `v` ima dva elementa više od vektora `c`. U ovom slučaju se elementi `v(1)` i `v(end)` shvataju kao vrednosti a_0 i a_m. Izlazni argument `pp` je struktura podataka, pa je primena funkcije `ppval` neminovnost.

Ako vektori `c` i `v` imaju istu dužinu, funkcija konstruiše splajn koji ima vrednosti prvog izvoda u čvorovima t_0 i t_m, koje su jednake izvodima polinoma trećeg stepena koji je konstruisan koristeći prva i poslednja četiri interpolaciona podatka, redom.

Funkcija `spline` se takođe može koristiti za konstrukciju pomenutog splajna. Format funkcije je

```
y=spline(c,v,x)
```

gde je ulazni argument `c` skup interpolacionih čvorova, dok je `v` skup vrednosti u interpolacionim čvorovima, a `x` je tačka u kojoj treba izračunati interpolacioni kubni splajn. Vrednosti a_0 i a_m se zadaju tako što vektor `v` ima dve koordinate više od vektora `c`. U tom slučaju se vrednosti `v(1)` i `v(end)` interpretiraju kao a_0 i a_m, redom.

Slika 4.16, desno, ilustruje konstrukciju interpolacionog kubnog splajna za razne vrednosti izvoda u krajnjim tačkama.

U slučaju da vektori `c` i `v` imaju istu dužinu funkcija `spline` konstruiše interpolacioni kubni splajn koji se obično naziva not-a-knot na engleskom jeziku.

Interpolacioni kubni splajn not-a-knot

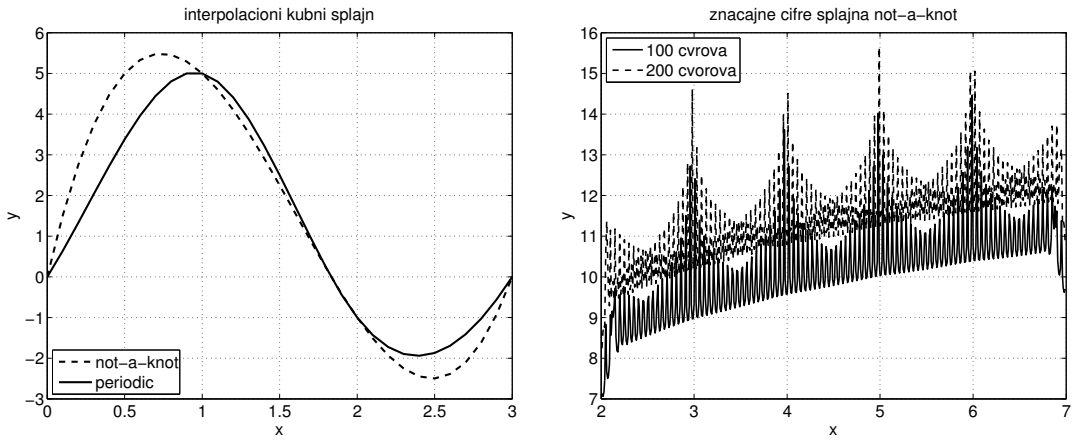
Interpolacioni kubni splajn not-a-knot se konstruiše tako što se slobodni parametri određuju iz uslova neprekidnosti trećeg izvoda splajna u tačkama t_1 i t_{m-1}. Ovaj uslov možemo izraziti sledećim sistemom linearnih jednačina

$$s_0'''(t_1) = \frac{z_1}{h_0} - \frac{z_0}{h_0} = \frac{z_2}{h_1} - \frac{z_1}{h_1} = s_1'''(t_1),$$
$$s_{m-2}'''(t_{m-1}) = \frac{z_{m-1}}{h_{m-1}} - \frac{z_{m-2}}{h_{m-2}} = \frac{z_m}{h_{m-1}} - \frac{z_{m-1}}{h_{m-1}} = s_{m-1}'''(t_{m-1}).$$

Ove dve jednačine, koje možemo da zapišemo u sledećem obliku

$$\frac{z_0}{h_0} - \left(\frac{1}{h_0} + \frac{1}{h_1}\right) z_1 + \frac{z_2}{h_1} = 0,$$
$$\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right) z_{m-1} + \frac{z_m}{h_{m-1}} = 0,$$

zajedno sa već postojećim sistemom jednačina (4.14), predstavljaju sistem linearnih jednačina koji omogućava konstrukciju interpolacionog kubnog splajna not-a-knot.



Slika 4.17: Slika levo ilustruje odnos periodičnog i not-a-knot interpolacionog kubnog splajna, slika desno ilustruje broj značajnih cifara interpolanta prilikom interpolacije interpolacionim kubnim splajnom not-a-knot funkcije log na intervalu $[2, 7]$ sa 100 i 200 čvorova.

Ovaj način konstrukcije splajna može se postići i koristeći funkciju `interp1`. Format funkcije `interp1` kojim se postiže ova konstrukcija je

```
y=interp1(c,v,x,'spline')
```

Takođe, moguće je koristiti i funkciju `spline`. Format funkcije je

```
y=spline(c,v,x)
```

ali je neophodno da vektori `c` i `v` imaju isti broj elemenata. Ako vektor `v` ima dva elmenta više, kako je već opisano, funkcija `spline` konstruiše splajn sa uslovima koji su vezani za prve izvode u čvorovima t_0 i t_m.

Konačno, moguće je koristiti i funkciju `csape`. U ovom slučaju format funkcije je

```
y=ppval(csape(c,v,'not-a-knot'),x)
```

Slika 4.17, levo, ilustruje interpolacioni kubni splajn not-a-knot.

Periodični interpolacioni kubni splajn

U nekim primenama je pogodno obezbediti periodičnost interpolacionog kubnog splajna. Periodičnost možemo postići interpolacionim uslovom $s(t_0) = s(t_m)$. Međutim, slobodne parametre splajna možemo odrediti i zahtevajući periodičnost prvog i drugog izvoda

$$s_0'(t_0) = s_{m-1}'(t_m), \quad s_0''(t_0) = s_{m-1}''(t_m).$$

U ovom slučaju, dobijamo dve dodatne jednačine određene sa

$$s_0'(t_0) = -\frac{h_0}{3}z_0 - \frac{h_0}{6}z_1 + \frac{f_1 - f_0}{h_0} = \frac{h_{m-1}}{3}z_m + \frac{h_{m-1}}{6}z_{m-1} + \frac{f_m - f_{m-1}}{h_{m-1}} = s_{m-1}'(t_m),$$
$$s_0''(t_0) = z_0 = z_m = s_{m-1}''(t_m).$$

Ove dve jednačine, zajedno sa sistemom (4.14), omogućavaju konstrukciju periodičnog splajna.

Funkcija `csape` omogućava konstrukciju periodičnog splajna. Format funkcije izgleda ovako

```
y=ppval(csape(c,v,'periodic'),x)
```

Slika 4.17, levo, ilustruje periodični interpolacioni kubni splajn.

Gubitak značajnih cifara

Za razliku od interpolacionih polinoma gde povećanje interpolacionih čvorova dovodi do smanjenja broja značajnih cifara kojim je određen interpolant, zbog raznih numeričkih efekata, posebno ako su interpolacioni čvorovi ekvidistantni, prilikom interpolacije kubnim splajnom, ovakvi efekti uglavnom nisu izraženi. Slika 4.17, desno, ilustruje broj značajnih cifara u interpolantu koji je konstruisan splajnom not-a-knot. Kao što vidimo gubitak značajnih cifara na krajevima intervala interpolacije je zanemarljivo mali u odnosu na algebarsku interpolaciju, iako smo konstruisali interpolante sa 100 i 200 ekvidistantnih interpolacionih čvorova.

Dalja analiza ovog problema prevazilazi okvire udžbenika.

4.6 Uopšteni problem interpolacije

U nekim primenama polinomi nisu dovoljni kao interpolacioni elementi. Priroda problema je takva da skup bazisnih funkcija mora biti izabran na neki drugi način. Na primer, u problemima identifikacije sistema često se koriste Müntzovi polinomi kao interpolacioni elementi.

U opštem slučaju imamo niz funkcija φ_k, $k \in \mathbb{N}_0$, koje možemo shvatiti kao zamenu za prirodnu bazu u prostoru polinoma x^k, $k \in \mathbb{N}_0$. Koristeći ovaj niz funkcija, po analogiji sa prirodnom bazom prostora polinoma, možemo formirati uopštene polinome

$$p(x) = \sum_{k=0}^{n} p_k \varphi_k(x).$$

Za ovakav uopšteni polinom reći ćemo da je stepena n, a za koeficijent p_n uz φ_n reći ćemo da je vodeći. Slično koeficijent uz φ_0 zvaćemo slobodnim članom.

Kao model može nam poslužiti sistem eksponencijalnih funkcija $\varphi_k(x) = e^{\lambda_k x}$, $k \in \mathbb{N}_0$, gde su λ_k, $k \in \mathbb{N}_0$, u opštem slučaju kompleksni brojevi. Specijalan slučaj ovog sistema funkcija je trigonometrijski sistem funkcija

$$\varphi_0(x) = 1, \quad \varphi_{2k}(x) = \cos kx, \quad \varphi_{2k-1}(x) = \sin kx, \quad k \in \mathbb{N}.$$

Takođe, Müntzov sistem funkcija $\varphi_k(x) = x^{\lambda_k}$, $k \in \mathbb{N}_0$, gde su λ_k, $k \in \mathbb{N}_0$, u opštem slučaju kompleksni brojevi.

Nas ovde pre svega zanima mogućnost rešenja uopštenog problema interpolacije. Uopšteni problem interpolacije se može formulisati na sledeći način: neka je dat skup interpolacionih čvorova x_i, $i = 0, \ldots, n$, skup vrednosti f_i, $i = 0, \ldots, n$, i niz funkcija φ_k, $k = 0, \ldots, n$, odrediti uopšteni polinom $p(x) = \sum_{k=0}^{n} p_k \varphi_k(x)$, tako da su zadovoljeni interpolacioni uslovi

$$p(x_i) = f_i, \quad i = 0, \ldots, n.$$

Ovde takođe podrazumevamo da su čvorovi različiti $x_i \neq x_j$ za $i \neq j$. Ukoliko postoji, uopšteni polinom p koji je rešenje interpolacionog problema zovemo interpolacioni polinom ili interpolant.

Jednostavno je konstruisati primer uopštenog interpolacionog problema koji nema rešenje. Na primer, ako izaberemo $x_0 = 0$, $x_1 = \pi$, $f_0 = 0$, $f_1 = 1$, $\varphi_0(x) = 1$ i $\varphi_1(x) = \sin x$ na osnovu interpolacionih uslova dobijamo

$$\begin{aligned} 0 = f_0 = p(x_0) = p(0) = p_0 \cdot 1 + p_1 \cdot \sin 0 = p_0, \\ 1 = f_1 = p(x_1) = p(\pi) = p_0 \cdot 1 + p_1 \cdot \sin \pi = p_0. \end{aligned}$$

Zaključujemo da mora važiti $0 = p_0 = 1$. Ovo je kontradikcija. Dakle, interpolant ne postoji i interpolacioni problem nema rešenje.

Međutim, ako potražimo rešenje u skupu funkcija $\varphi_0(x) = 1$ i $\varphi_1(x) = \cos x$, nalazimo

$$\begin{aligned} 0 = f_0 = p(x_0) = p(0) = p_0 \cdot 1 + p_1 \cdot \cos 0 = p_0 + p_1, \\ 1 = f_1 = p(x_1) = p(\pi) = p_0 \cdot 1 + p_1 \cdot \cos \pi = p_0 - p_1. \end{aligned}$$

Očigledno ovaj interpolacioni problem ima rešenje. Rešenje je

$$p(x) = \frac{1 - \cos x}{2}.$$

Kako je nemoguće opisati svaki pojedinačni niz funkcija to uvodimo definiciju Čebiševljevog sistema funkcija kao niza funkcija za koji interpolacioni problem ima rešenje.

4.6.1 Čebiševljev sistem

Definicija 4.4 (Čebiševljev sistem) *Niz funkcija $\varphi_k \in C[a,b]$, $k = 0, \ldots, n$, je Čebiševljev sistem na intervalu $[a,b]$ ako i samo ako je jedini uopšteni polinom*

$$p(x) = \sum_{k=0}^{n} p_k \varphi_k(x)$$

koji ima $(n+1)$-nu nulu na $[a,b]$ polinom sa koeficijentima $p_k = 0$, $k = 0, \ldots, n$.

Ovakva definicija Čebiševljevog sistema omogućava sledeću karakterizaciju.

Lema 4.8 *Niz funkcija $\varphi_k \in C[a,b]$, $k = 0, \ldots, n$, je Čebiševljev na $[a,b]$ ako i samo ako za svaki izbor međusobno različitih tačaka $x_i \in [a,b]$, $i = 0, \ldots, n$, važi*

$$\Delta(x_0, \ldots, x_n) = \begin{vmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} \neq 0.$$

Dokaz. Neka niz funkcija φ_k, $k = 0, \ldots, n$, nije Čebiševljev. Onda postoji uopšteni polinom $p = \sum p_k \varphi_k$ koji ima $(n+1)$-nu nulu na $[a, b]$. Neka su to nule x_i, $i = 0, \ldots, n$. Posmatrajmo determinantnu

$$\Delta(x_0, \ldots, x_n) = \begin{vmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix}.$$

Kako polinom p nema sve koeficijente jednake nuli, to postoji $p_k \neq 0$. Neka je to recimo p_0. Pomnožimo prvu kolonu sa p_0 i dodajmo prvoj koloni svaku k-tu kolonu pomnoženu sa p_k, $k = 1, \ldots, n$. Dobijamo

$$\begin{aligned} \Delta(x_0, \ldots, x_n) &= \frac{1}{p_0} \begin{vmatrix} p_0\varphi_0(x_0) + \cdots + p_n\varphi_n(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ p_0\varphi_0(x_1) + \cdots + p_n\varphi_n(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ p_0\varphi_0(x_n) + \cdots + p_n\varphi_n(x_n) & \varphi_1(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} \\ &= \frac{1}{p_0} \begin{vmatrix} p(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ p(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ p(x_n) & \varphi_1(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = \frac{1}{p_0} \begin{vmatrix} 0 & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ 0 & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \varphi_1(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = 0. \end{aligned}$$

Pretpostavimo da je $\Delta(x_0, \ldots, x_n) = 0$ za neki skup međusobno različitih tačaka. Onda su kolone te determinante linearno zavisne. Dakle, postoje skalari α_k, $k = 0, \ldots, n$, od kojih bar jedan nije jednak nuli, tako da važi

$$0 = \alpha_0 \begin{bmatrix} \varphi_0(x_0) \\ \varphi_0(x_1) \\ \vdots \\ \varphi_0(x_n) \end{bmatrix} + \alpha_1 \begin{bmatrix} \varphi_1(x_0) \\ \varphi_1(x_1) \\ \vdots \\ \varphi_1(x_n) \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} \varphi_n(x_0) \\ \varphi_n(x_1) \\ \vdots \\ \varphi_n(x_n) \end{bmatrix} = \begin{bmatrix} \alpha_0\varphi_0(x_0) + \cdots + \alpha_n\varphi_n(x_0) \\ \alpha_0\varphi_0(x_1) + \cdots + \alpha_n\varphi_n(x_1) \\ \vdots \\ \alpha_0\varphi_0(x_n) + \cdots + \alpha_n\varphi_n(x_n) \end{bmatrix}.$$

Dovoljno je onda izabrati $p = \sum \alpha_k \varphi_k$ da zaključimo da postoji uopšteni polinom koji ima $(n+1)$-nu nulu na intervalu $[a, b]$. Zaključujemo da niz funkcija φ_k, $k = 0, \ldots, n$, nije Čebiševljev sistem na $[a, b]$. □

Ova karakterizacija Čebiševljevih sistema nam omogućava formulaciju sledeće teoreme.

Teorema 4.21 (Uopšteni interpolacioni problem) *Neka je φ_k, $k = 0, \ldots, n$, Čebiševljev sistem funkcija na $[a, b]$ i neka su interpolacioni čvorovi $x_i \in [a, b]$, $i = 0, \ldots, n$, međusobno različiti. Interpolacioni problem*

$$p(x_i) = \sum_{k=0}^{n} p_k \varphi_k(x_i) = f_i, \quad i = 0, \ldots, n,$$

u skupu uopštenih polinoma ima jedistveno rešenje.

Dokaz. Interpolacioni problem možemo zapisati u matričnoj formi na sledeći način

$$\begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

Kako je niz funkcija φ_k, $k = 0, \ldots, n$, Čebiševljev sistem na $[a, b]$, na osnovu leme 4.8, znamo da je determinanta matrice sistema različita od nule. Sledi da sistem linearnih jednačina ima jedinstveno rešenje. □

Kao što vidimo Čebiševljevi sistemi su značajni jer omogućavaju jedinstvenost rešenja uopštenog interpolacionog problema.

Niz funkcija $\varphi_k(x) = x^k$, $k = 0, \ldots, n$, je Čebiševljev sistem na skupu realnih brojeva. Dokaz je jednostavan. Determinanta iz leme 4.8, u ovom slučaju, se svodi na Vandermondovu determinantu

$$\Delta(x_0, \ldots, x_n) = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix} = \prod_{i,j=0,\, i>j}^{n} (x_i - x_j),$$

čija vrednost nije jednaka nuli jer su interpolacioni čvorovi međusobno različiti.

Izaberimo niz funkcija $\varphi_{2k}(x) = \cos kx$, $k = 0, \ldots, n$ i $\varphi_{2k-1}(x) = \sin kx$, $k = 1, \ldots, n$. Ovaj niz funkcija je Čebiševljev sistem funkcija na intervalu $[0, 2\pi)$. Ovo se može dokazati izračunavanjem determinante

$$\Delta(x_0, \ldots, x_{2n}) = \begin{vmatrix} 1 & \cos x_0 & \sin x_0 & \cdots & \cos nx_0 & \sin nx_0 \\ 1 & \cos x_1 & \sin x_1 & \cdots & \cos nx_1 & \sin nx_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \cos x_{2n} & \sin x_{2n} & \cdots & \cos nx_{2n} & \sin nx_{2n} \end{vmatrix} = 2^{2n^2} \prod_{i,j=0,i>j}^{2n} \sin \frac{x_i - x_j}{2}, \tag{4.15}$$

čija vrednost nije jednaka nuli, jer je $0 < (x_i - x_j)/2 < \pi$, $i > j$, pa je $\sin(x_i - x_j)/2 > 0$.

Na skupu $\mathbb{R}$ niz funkcija $\varphi_k(x) = e^{\lambda_k x}$, $k = 0, \ldots, n$, $\lambda_k < \lambda_{k+1}$, $k = 0, \ldots, n-1$, je Čebiševljev sistem funkcija. Niz funkcija $\varphi_{2k}(x) = \cosh kx$, $k = 0, \ldots, n$, i $\varphi_{2k-1}(x) = \sinh kx$, $k = 1, \ldots, n$, je Čebiševljev sistem na skupu $\mathbb{R}$.

Na skupu $\mathbb{R}^+$ sledeći nizovi funkcija $\varphi_k(x) = x^{\lambda_k}$, $k = 0, \ldots, n$, $\lambda_k < \lambda_{k+1}$, $k = 0, \ldots, n-1$, i $\varphi_k(x) = (x + \lambda_k)^{-1}$, $k = 0, \ldots, n$, $0 \le \lambda_0$, $\lambda_k < \lambda_{k+1}$, $k = 0, \ldots, n-1$, su Čebiševljevi sistemi funkcija.

Postoje i drugi nizovi funkcija za koje se može pokazati da su Čebiševljevi sistemi. Mi smo ovde naveli samo najilustrativnije primere.

Navedimo još jedno elementarno svojstvo Čebiševljevog sistema funkcija.

Lema 4.9 *Neka je niz funkcija $\varphi_k \in C[a, b]$, $k = 0, \ldots, n$, Čebiševljev sistem funkcija na $[a, b]$ i neka je $[c, d] \subset [a, b]$. Onda je niz funkcija φ_k, $k = 0, \ldots, n$, Čebiševljev sistem funkcija na $[c, d]$.*

Dokaz. Ako φ_k, $k = 0, \ldots, n$, nije Čebiševljev sistem funkcija na $[c, d]$, onda za uopšteni polinom $p = \sum p_k \varphi_k$ postoje tačke $x_i \in [c, d]$, $i = 0, \ldots, n$, koje su nule uopštenog polinoma p. Međutim,

onda su $x_i \in [c,d] \subset [a,b]$, $i = 0,\ldots,n$, nule uopštenog polinoma p i na intervalu $[a,b]$, pa niz funkcija φ_k, $k = 0,\ldots,n$, nije Čebiševljev na $[a,b]$. □

Na osnovu ove leme je, recimo, niz funkcija $\varphi_k(x) = e^{\lambda_k x}$, $k = 0,\ldots,n$, $\lambda_k < \lambda_{k+1}$, $k = 0,\ldots,n-1$, Čebiševljev na svakom intervalu $[a,b] \subset \mathbb{R}$, niz funkcija $\varphi_k(x) = x^{\lambda_k}$, $k = 0,\ldots,n$, $0 \le \lambda_0$, $\lambda_k < \lambda_{k+1}$, $k = 0,\ldots,n-1$, je Čebiševljev na svakom intervalu $[a,b] \subset \mathbb{R}^+$. Slični zaključci važe i za ostale pomenute nizove funkcija.

4.6.2 Trigonometrijski interpolant

Izračunavanje determinante (4.15) omogućava formulaciju sledeće teoreme.

Teorema 4.22 (Trigonometrijska interpolacija) *Neka su $x_i \in [0,2\pi)$, $i = 0,\ldots,2n$, međusobno različiti interpolacioni čvorovi. Interpolacioni problem*

$$T_n(x_i) = \sum_{k=0}^{n} a_k \cos kx_i + b_k \sin kx_i = f_i, \quad i = 0,\ldots,2n, \tag{4.16}$$

ima jedinstveno rešenje.

Dokaz. Primetimo da je vrednost sabirka $b_0 \sin 0x = 0$, tako da ovaj sabirak ne učestvuje u sumi. Broj nepoznatih koeficijenata iznosi $2n+1$ koliko ima i interpolacionih uslova.

Niz funkcija $\varphi_{2k}(x) = \cos kx$, $k = 0,\ldots,n$, i $\varphi_{2k-1}(x) = \sin kx$, $k = 1,\ldots,n$, je Čebiševljev sistem na intervalu $[0,2\pi)$ na osnovu vrednosti determinante (4.15). Na osnovu teoreme 4.21 zaključujemo da uopšteni interpolacioni problem ima jedinstveno rešenje. □

Polinom T_n, koji je jedinstveno rešenje interpolacionog problema, naziva se trigonometrijski interpolacioni polinom. Interesantno je da polinom T_n može biti iskazan koristeći formu koja je vrlo slična Lagrangeovoj formi algebarskog interpolanta. Da bismo mogli da formulišemo ovu teoremu neopodna nam je konstrukcija Lagrangeovog bazisa trigonometrijske interpolacije.

Lema 4.10 (Lagrangeov bazis trigonometrijske interpolacije) *Trigonometrijski polinomi*

$$t_k(x) = \prod_{i=0,\, i\neq k}^{2n} \frac{\sin\dfrac{x-x_i}{2}}{\sin\dfrac{x_k-x_i}{2}}, \quad k = 0,\ldots,2n,$$

zadovoljavaju sledeći identitet

$$t_k(x_j) = \begin{cases} 0, & k \neq j, \\ 1, & k = j \end{cases}$$

Dokaz. Ako je $k = j$ onda je očigledno $t_k(x_k) = 1$. Ako je $k \neq j$, onda u proizvodu postoji član za koji važi $i = j$, što daje $\sin(x_j - x_i)/2 = 0$ i $t_k(x_j) = 0$. □

Ovde može da deluje zbunjujuća forma polinoma t_k. Naime, nije očigledno kako se polinomi t_k mogu odrediti kao uopšteni polinomi po trigonometrijskom nizu $\cos kx$, $k = 0,\ldots,n$, $\sin kx$, $k = 1,\ldots,n$. Rešenje je u transformaciji proizvoda trigonometrijskih funkcija u njihov zbir. Važe identiteti

$$\sin\alpha\sin\beta = \frac{1}{2}(\cos(\alpha-\beta) - \cos(\alpha+\beta)), \quad \sin(\alpha+\beta) = \sin\alpha\cos\beta + \sin\beta\cos\alpha$$
$$\cos\alpha\cos\beta = \frac{1}{2}(\cos(\alpha-\beta) + \cos(\alpha+\beta)), \quad \cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$
$$\sin\alpha\cos\beta = \frac{1}{2}(\sin(\alpha-\beta) + \sin(\alpha+\beta)).$$

Koristeći ove identitete, za sličaj $n = 1$, možemo izračunati

$$t_0 = A_0 \sin\frac{x-x_1}{2}\sin\frac{x-x_2}{2} = \frac{A_0}{2}\left(\cos\frac{x_2-x_1}{2} - \cos\left(x - \frac{x_1+x_2}{2}\right)\right)$$
$$= \frac{A_0}{2}\left(\cos\frac{x_2-x_1}{2} - \cos\frac{x_1+x_2}{2}\cos x - \sin\frac{x_1+x_2}{2}\sin x\right),$$

gde je $A_0 = (\sin(x_0-x_1)/2\sin(x_0-x_2)/2)^{-1}$. Kao što vidimo polinom t_0 stvarno pripada uopštenim polinomima po trigonometrijskom nizu. Na sličan način se može postupiti u generalnom slučaju kad imamo $2n+1$ interpolacionih tačaka i za proizvoljni polinom t_k, $k = 0, \ldots, 2n$.

Teorema 4.23 (Lagrangeova forma trigonometrijskog interpolanta) *Neka je u interpolacionim čvorovima $x_i \in [0, 2\pi)$, $i = 0, \ldots, 2n$, zadata funkcija f sa svojim vrednostima $f_i = f(x_i)$, $i = 0, \ldots, 2n$. Trigonometrijski interpolacioni polinom T_n koji zadovoljava interpolacione uslove*

$$T_n(x_i) = f_i, \quad i = 0, \ldots, 2n,$$

je zadat sa

$$T_n(x) = \sum_{k=0}^{2n} f_k t_k(x).$$

Dokaz. Izaberimo $x = x_i$. Na osnovu leme 4.10 zaključujemo da važi

$$T_n(x_i) = \sum_{k=0}^{2n} f_k t_k(x_i) = f_i,$$

jer su u sumi svi sabirci jednaki nuli, izuzev sabirka $k = i$. □

Nadalje se mogu posmatrati sve osobine trigonometrijskog interpolanta koje smo već posmatrali u slučaju algebarskog interpolanta. Zbog ograničenog prostora mi se u ova razmatranja nećemo upuštati.

Glava 5

Aproksimacija diskretnih skupova podataka

5.1 Najmanji kvadrati

5.1.1 Postavka problema

Posmatrajmo sledeći problem. Neka je dat skup tačaka u ravni (x_i, y_i), $i = 0, \ldots, n$, koji zadovoljava uslov jednoznačnosti $x_i \neq x_j$ za $i \neq j$. U glavi o interpolaciji mi smo se bavili problemom određivanja polinoma p koji u svakoj tački x_i ima vrednost y_i. Međutim, u većini praktičnih problema ovaj uslov je previše strog. Zamislimo situaciju u kojoj su vrednosti y_i, $i = 0, \ldots, n$, rezultati merenja. U ovom slučaju su vrednosti y_i date sa greškom i nije neophodno zahtevati da polinom p u tački x_i ima egzaktnu vrednost y_i. U većini problema dovoljno je da polinom p u tački x_i ima samo približno vrednost y_i.

Ovo nas dovodi do sledećeg problema. Pretpostavimo da imamo dva skupa tačaka u ravni (x_i, y_i), i $(x_i, p(x_i))$, $i = 0, \ldots, n$. Postavlja se pitanje: koliko je daleko jedan skup podataka od drugog. Najjednostavniji način da izmerimo ovo rastojanje je da tretiramo oba skupa podataka kao vektore

$$y = (y_0, \ldots, y_n), \quad P = (p(x_0), \ldots, p(x_n)),$$

onda rastojanje možemo odrediti kao $d(y, P) = ||y - P||$, gde za normu možemo izabrati bilo koju od uvedenih normi vektora.

Sa ovako uvedenim rastojanjem možemo postaviti i sledeći problem. Naći polinom p koji zadovoljava sledeću nejednakost

$$d(y, P) = ||y - P|| < \delta,$$

gde je δ unapred zadata tačnost. Međutim, ovako postavljen problem ima trivijalno rešenje. Naime, dovoljno je za p izabrati interpolacioni polinom koja zadovoljava interpolacione uslove $p(x_i) = y_i$, $i = 0, \ldots, n$. Zato se problem ne posmatra nad svim mogućim polinomima, nego se posmatra nad skupom polinoma ograničenog stepena.

Najčešće problem posmatramo nad skupom polinoma ne većeg stepena od m, u oznaci $\mathcal{P}_m$. Pri ovome podrazumevamo da je m manje od n, inače dobijamo trivijalno rešenje u obliku interpolacionog polinoma. U praksi je obično m mnogo manje od n. Tako da se problem može opisati

i kao problem redukcije kompleksnosti. Naime, kako je stepen polinoma ograničen, kompleksnost polinoma p je ograničena, pa možemo reći da dobijamo rešenja ograničene kompleksnosti.

Sa ovim razmatranjem naš problem dobija sledeću formulaciju: u skupu polinom $\mathcal{P}_m$ naći polinom p koji zadovoljava uslov

$$d(y, P) = ||y - P|| < \delta.$$

Međutim, ovako postavljen problem ne mora imati rešenje, jer ništa nam ne garantuje da u skupu polinoma $\mathcal{P}_m$ postoji polinom koji je od zadatog skupa podataka (x_i, y_i), $i = 0, \ldots, n$, na rastojanju manjem od δ. Ono što možemo sa sigurnošću da odredimo je polinom $p \in \mathcal{P}_m$ koji je na najmanjem rastojanju od zadatog skupa podataka (x_i, y_i), $i = 0, \ldots, n$. Problem dobija novu formulaciju: naći $p \in \mathcal{P}_m$ tako da je $d(y, P)$ najmanje. Ovako postavljen problem ima limitiranu kompleksnost polinoma p i u okviru zadate limitirane kompleksnosti pronalazi najbolje moguće rešenje.

Mi se nećemo baviti svim mogućim normama vektora. Ograničićemo se na Euklidovu normu, jer sa ovom normom problem ima analitičko rešenje. Sa uvedenim ograničenjima naš problem dobija formu: odrediti

$$p(x) = \sum_{k=0}^{m} p_k x^k$$

tako da izraz

$$d(y, P) = ||y - P||_2 = \sqrt{\sum_{i=0}^{n} (y_i - p(x_i))^2}$$

ima najmanju moguću vrednost.

Problem može da pretrpi još neke modifikacije. Funkcija stepenovanja pozitivnim brojem je rastuća, tako da se rešenje problema neće promeniti ako posmatramo problem

$$d(y, P)^2 = ||y - P||_2^2 = \sum_{i=0}^{n} (y_i - p(x_i))^2.$$

Konačno, problem možemo shvatiti kao minimizaciju funkcije

$$F_m(p_0, \ldots, p_m) = \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} p_k x_i^k \right)^2. \tag{5.1}$$

Ovo je klasična formulacija problema. Mi ćemo se zadržati na ovoj formulaciji problema i daćemo rešenje ovako formulisanog problema. Ovako formulisan problem se obično naziva problem najmanjih kvadrata, zbog kvadrata koji učestvuju u izrazu.

5.1.2 Rešenje problema

Nadalje ćemo, uz očiglednu zloupotrebu notacije, označavati sa p polinom

$$p(x) = \sum_{k=0}^{m} p_k x^k,$$

i vektor koeficijenata $p = (p_0, \ldots, p_m)$ polinoma p. Iz konteksta će biti jasno na koju veličinu mislimo.

Teorema 5.1 (Problem najmanjih kvadrata) *Neka je dat skup tačaka u ravni (x_i, y_i), $i = 0, \ldots, n$, koji zadovoljava uslov $x_i \neq x_j$, $i \neq j$. Vektor $p = (p_0, \ldots, p_m)$ za koji funkcija*

$$F_m(p) = \sum_{i=0}^{n} (y_i - p_0 - \cdots - p_m x_i^m)^2$$

dostiže minimum je rešenje sistema linearnih jednačina

$$V^T V p = V^T y, \quad V = \begin{bmatrix} 1 & x_0 & \cdots & x_0^m \\ 1 & x_1 & \cdots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^m \end{bmatrix}.$$

Sistem jednačina $V^T V p = V^T y$ ima jedinstveno rešenje

$$p = (V^T V)^{-1} V^T y.$$

Dokaz. Ovako postavljen problem je jednostavno problem određivanja minimuma funkcije F_m više promenljivih. Stacionarne tačke, tačke u kojima su moguće ekstremne vrednosti, određujemo iz uslova

$$\frac{\partial F_m(p)}{\partial p_j} = \sum_{i=0}^{n} (-2)(y_i - p_0 - \cdots - p_m x_i^m) x_i^j = 0, \quad j = 0, \ldots, m.$$

Ovo je sistem linearnih jednačina koji posle malo manipulacija jednačinama postaje

$$\sum_{i=0}^{n} x_i^j \sum_{k=0}^{m} x_i^k p_k = \sum_{i=0}^{m} x_i^j y_i, \quad j = 0, \ldots, m.$$

Ovde možemo da prepoznamo proizvode matrice slične Vandermondovoj matrici

$$V = \begin{bmatrix} 1 & x_0 & \cdots & x_0^m \\ 1 & x_1 & \cdots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^m \end{bmatrix}$$

i vektora p, y i $x^j = (x_0^j, x_1^j, \ldots, x_n^j)$, $j = 0, \ldots, m$. Tako sistem dobija formu

$$(x^j)^T V p = (x^j)^T y, \quad j = 0, \ldots, m.$$

Međutim, proizvod $(x^j)^T V$ je vektor vrsta, dok je $(x^j)^T y$ broj. Tako prethodni sistem jednačina možemo da sažmemo u formu

$$\begin{bmatrix} (x^0)^T V \\ \vdots \\ (x^m)^T V \end{bmatrix} p = \begin{bmatrix} (x^0)^T y \\ \vdots \\ (x^m)^T y \end{bmatrix},$$

a ovaj sistem jednačina možemo zapisati u formi

$$\begin{bmatrix} (x^0)^T \\ \vdots \\ (x^m)^T \end{bmatrix} V p = \begin{bmatrix} (x^0)^T \\ \vdots \\ (x^m)^T \end{bmatrix} y.$$

Matrica

$$\begin{bmatrix} (x^0)^T \\ \vdots \\ (x^m)^T \end{bmatrix}$$

je samo transponovana matrica matrice V, tako da konačno sistem dobija oblik

$$V^T V p = V^T y.$$

Ako potražimo drugi izvod funkcije F_m, nalazimo

$$\frac{\partial^2 F_m(p)}{\partial p_k \partial p_j} = 2 \sum_{i=0}^{n} x_i^k x_i^j, \quad k, j = 0, \ldots, n.$$

Odavde zaključujemo da je Hessian sistema zadat sa

$$H = \left[\frac{\partial^2 F_m(a)}{\partial p_k \partial p_j} \right] = 2V^T V.$$

Primetimo da je matrica V punog ranga, jer je submatrica

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^m \\ 1 & x_1 & \cdots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^m \end{bmatrix}$$

regularna, kao Vandermondova matrica za vrednosti $x_i \neq x_j$, $i \neq j$, što je tačno po uslovu teoreme. Činjenica da je matrica V punog ranga znači da je jedino rešenje jednačine $Vb = 0$, upravo vektor $b = 0$.

Međutim, za $b \neq 0$, jednostavno nalazimo da važi

$$(b, Hb) = b^T H^T b = b^T (2V^T V)^T b = 2b^T V^T V b = 2(Vb)^T (Vb) = 2(Vb, Vb) > 0.$$

Odavde zaključujemo da je matrica H pozitivno definitna, pa je stacionarna tačka, tačka minimuma funkcije F_m.

Kako je matrica $V^T V$ pozitivno definitna, to je jedino rešenje jednačine $V^T V b = 0$ upravo vektor $b = 0$. Međutim, onda je matrica $V^T V$ regularna, pa sistem linearnih jednačina $V^T V p = V^T y$ ima jedinstveno rešenje. □

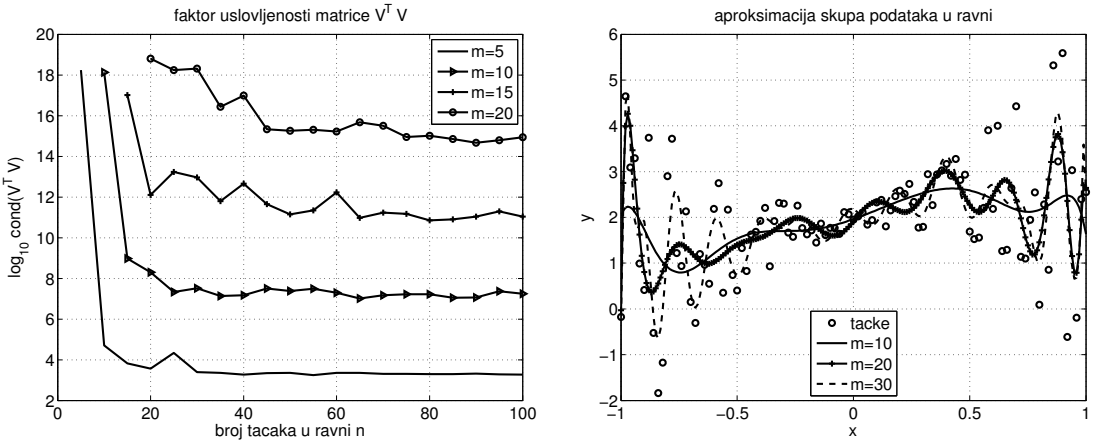
5.1.3 Numerička konstrukcija

Numerička konstrukcija rešenja problema najmanjih kvadrata je implementirana u operatoru \ matričnog deljenja. Operator \, kao što znamo, rešava matričnu jednačinu $Ax = b$ na sledeći način `A\b`, pod uslovom da je matrica A kvadratna. Međutim, ako matrica A nije kvadratna, već je tipa $n \times m$, gde je $n > m$, onda operator pronalazi rešenje metodom najmanjih kvadrata.

Zbog ove osobine operatora \ možemo vrlo jednostavno implementirati funkciju koja konstruiše rešenje problema najmanjih kvadrata.

```
function [ res,err ] = najmanjiKvadrati( m,c,v,x )
%NAJMANJIKVADRATI(M,C,V,X) konstruise resenje problema najmanjih
% kvadrata, u skupu polinom P_m, za skup tacaka u ravni
% (c(i),v(i)), i=0,...,n u tacki x, izlazni parametar res sadrzi
% koeficijente polinoma, dok err sadrzi velicinu greske
% srednje kvadratne aproksimacije

n = length(c)-1;
V = zeros(n+1,m+1);
V(:,1) = ones(n+1,1);
for j=2:(m+1)
    V(:,j) = V(:,j-1).*(c');
end
p = V\v';
res = polyval(flipud(p),x);
err = norm(v'-V*p)/sqrt(n);
end
```

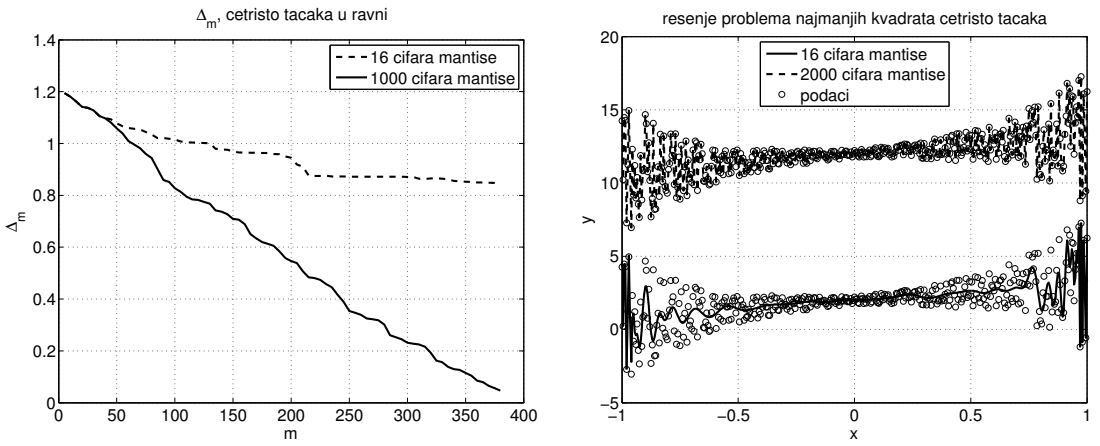
Slika 5.1, levo, ilustruje faktor uslovljenosti matrice V^TV u funkciji broja tačaka u ravni, sa stepenom polinoma p uzetim kao parametar. Kao što vidimo faktor uslovljenosti matrice uglavnom



Slika 5.1: Slika levo ilustruje zavisnost faktora uslovljenosti matrice V^TV u funkciji broja tačaka n za stepen polinoma m uzet kao parametar, slika desno ilustruje grafik polinoma rešenja najmanjih kvadrata dobijen za skup tačaka $x_i = -1 + 2i/100$, $y_i = 2 + x_i + .25e^{3|x_i|}\sigma_i$, $i = 0, \ldots, 100$, gde su σ_i uniformno raspodeljeni slučajni brojevi u intervalu $[-1, 1]$.

ne zavisi od n, kad vrednost n dostigne $2m$. Međutim, vidimo da svako povećanje stepena polinoma m za 5, otprilike povećava faktor uslovljenosti matrice V^TV za 4 reda veličine. Ovo jasno ukazuje da stepen polinoma p preko vrednosti 40 nema smisla ni koristiti, jer će u svakom slučaju koeficijenti polinoma biti izračunati sa vrlo malo značajnih cifara.

Slika 5.1, desno, ilustruje odnos rešenja problema najmanjih kvadrata stepena $m = 10, 20$ i 30, za problem koji ima 101 tačku u ravni. Vidimo da povećanje stepena polinoma dovodi do približavanja grafika polinoma tačkama koje pokušavamo da aproksimiramo. Međutim, takođe



Slika 5.2: Slika levo ilustruje zavisnost odstupanja Δ_m u funkciji stepena polinoma m, slika desno ilustruje grafike polinoma dobijenih sa aritmetikom mantise 16 i 2000 dekadnih cifara, za $n = 400$ i $m = 390$. Na slici desno, zbog preglednosti, grafik polinoma konstruisan sa 2000 cifara mantise i podaci su prikazani translirani po y-osi za vrednost 10.

možemo primetiti da već kod polinoma stepena $m = 30$ dolazi do ozbiljnog iskakanja grafika polinoma iz konveksnog omotača skupa tačaka u ravni. Dalje povećanje stepena polinoma još više potencira ovu pojavu.

Tačka u kojoj funkcija F_m dostiže minimum je rešenje problema najmanjih kvadrata. Vrednost funkcije F_m, u rešenju sistema linearnih jednačina, daje ocenu greške metoda najmanjih kvadrata u skupu polinoma $\mathcal{P}_m$. Međutim, očigledno je da nije isto koliko tačaka u ravni imamo, jer veći broj tačaka u ravni znači i povećanje sabiraka u definiciji funkcije F_m. Zato definišemo normalizovanu grešku

$$\Delta_m = \left(\frac{F_m(p)}{n} \right)^{1/2}$$

da bismo bili u stanju da poredimo greške za različite brojeve tačaka u ravni. Kako je $\mathcal{P}_{m-1} \subset \mathcal{P}_m$ možemo tvrditi da je $\Delta_{m-1} \geq \Delta_m$. Očigledno je niz Δ_m nerastući. Kako je u slučaju $m = n$ rešenje problema najmanjih kvadrata interpolacioni polinom možemo tvrditi da je $\Delta_n = 0$. Na osnovu ovih zapažanja možemo očekivati da niz Δ_m, $m = 1, \ldots, n$, opada ka nuli. Videćemo da ovakvo ponašanje postoji do neke vrednosti m, ali zbog efekata aritmetike mantise konačne dužine ovakvo ponašanje nije garantovano.

Slika 5.2, levo, ilustruje zavisnost veličine Δ_m u funkciji od stepena polinoma m. Dve krive prikazuju zavisnost veličine Δ_m sa izračunavanjima izvedenim u aritmetici mantise 16 dekadnih cifara (dvostruka preciznost) i 1000 dekadnih cifara. Kao što vidimo ako računamo sa 1000 dekadnih cifara veličina Δ_m opada ka nuli, dok u aritmetici dužine 16 dekadnih cifara Δ_m vrlo brzo postane konstantno. Ovo je posledica velikog faktora uslovljenosti matrica $V^T V$ koji onemogućava precizna izračunavanja. Grafik na slici je konstruisan za problem sa $n = 400$ tačaka u ravni.

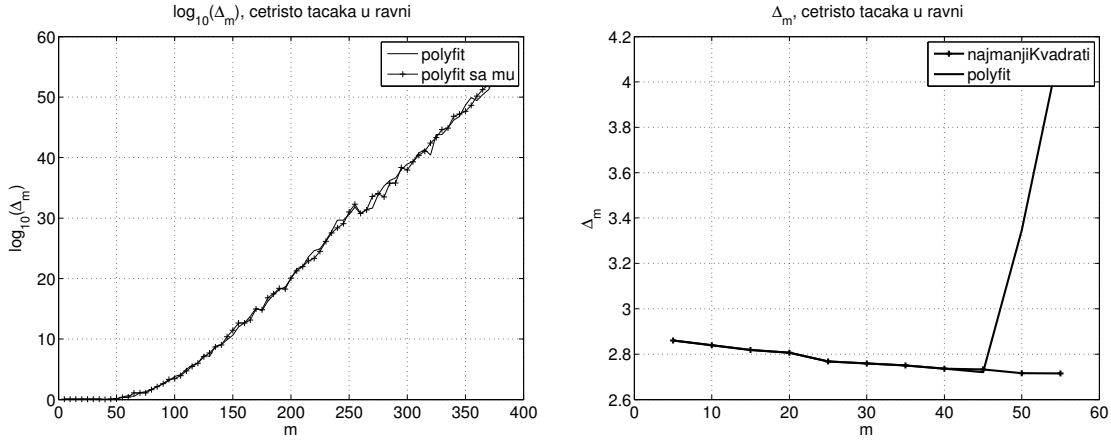
Slika 5.2, desno, ilustruje grafike polinoma i podataka za $n = 400$ tačaka u ravni i polinom stepena $m = 390$. Izračunavanja su u jednom slučaju izvedena sa mantisom od 16 dekadnih cifara (dvostruka preciznost), a u drugom sa mantisom od 2000 dekadnih cifara. Zbog preglednost, polinom konstruisan sa 2000 dekadnih cifara mantise i podaci su prikazani translirani po y-osi za

vrednost 10. Kao što se vidi sa grafika polinom konstruisan sa 2000 dekadnih cifara uspeva da prođe dovoljno blizu svake tačke u skupu podataka. Međutim, interesantno je da ovaj polinom ima vrednosti reda veličine 10^{118} za vrednosti x koje se nalaze između apscisa tačaka podataka. Grafici polinoma su nacrtani u apscisama podataka. Može se zaključiti da se povećanjem dužine mantise aritmetike u kojoj se izvode izračunavanja dobijaju tačni koeficijenti polinoma, ali su vrednosti tih polinomi uglavnom neupotrebljive u tačkama koje ne predstavljaju apscise podataka.

5.1.4 Matlab implementacija

Metod najmanjih kvadrata je u `Matlab`u implementiran u funkciji `polyfit`. Format funkcije je sledeći

```
[p,S,mu]=polyfit(c,v,m)
```



Slika 5.3: Slika levo ilustruje zavisnost odstupanja $\log_{10} \Delta_m$ u funkciji stepena polinoma m u izlazu funkcije polyfit, slika desno ilustruje odnos veličina Δ_m kod funkcija najmanjiKvadrati i polyfit za polinome do stepena 55.

Ulazni parametar `c` predstavlja apscise, a `v` ordinate tačaka podataka, ulazni parametar `m` predstavlja stepen polinoma rešenja. Izlazni parametar `p` predstavlja koeficijente polinoma rešenja, po konvenciji reprezentcije polinoma vektorima u `Matlab`u, izlazni argument `S` predstavlja strukturu podataka koja nosi informacije o rezultatu primenjenog algoritma, izlazni argument `mu` sadrži srednju vrednost vektora `v` i standardnu devijaciju. Kad je funkcija `polyfit` pozvana sa tri izlazna argumenta koeficijenti polinoma su konstruisani po promenljivoj $(x - \mu_1)/\mu_2$, gde je μ_1 srednja vrednost vektora `v` dok je μ_2 standardna devijacija vektora `v`. Algoritam rešava sistem linearnih jednačina koristeći QR faktorizaciju Vandermondove matrice za skup čvorova `c`. Polja unutar strukture `S` su: `R` koje sadrži celu matricu R iz QR faktorizacije Vandermondove matrice, `normr` koje sadrži vrednost Δ_m, polje `df` predstavlja broj stepena slobode koje ima problem. Broj stepena slobode obično iznosi broj tačaka u ravni umanjen za stepen polinoma plus jedan. Ako želimo da izračunamo vrednost rešenja u tački `x` koristimo funkciju `polyval`. Na primer, `polyval(p,x)`.

Ono što nas ovde zanima su performanse implementirane funkcije `polyfit`. Pre svega nas zanima ponašanje greške, polje `S.normr`, sa porastom kompleksnosti problema. Slika 5.3, levo, ilustruje

ponašanje veličine `S.normr` za problem najmanjh kvadrata identičan problemu koji je prikazan na slici 5.2, levo. Primećujemo da je greška mnogo redova veličine veća od one koju generiše funkcija `najmanjiKvadrati`. Greška je toliko velika da nismo bili u mogućnosti da prikažemo rezultate na jednom zajedničkom grafiku, pa smo se odlučili da prikažemo samo grafik funkcije `log10(S.normr)`. Razlog za ovakvo ponašanje je QR faktorizacija Vandermondove matrice. Ovaj postupak proizvodi greške koje se reflektuju u rešenju. Možemo zaključiti da je implementacija pomoću funkcije `najmanjiKvadrati` bolja što se tiče kvaliteta izračunavanja, mada slabije obrađuje rezultate izračunavanja. Slika 5.3, desno, ilustruje da do odstupanja dolazi za polinome pedesetog stepena.

5.1.5 Ekvidistantne tačke

Postoji mogućnost konstrukcije rešenja problema najmanjih kvadrata koristeći metod koji je numerički daleko stabilniji od prezentovanog. Algoritam podrazumeva neke teorijske koncepte koji prevazilaze okvire knjige. Međutim, u slučaju ekvidistantnih tačaka algoritam dobija posebno jednostavnu formu i ovaj slučaj ćemo prezentovati. Napominjemo da slučaj neekvidistantnih čvorova može biti rešen analogno, ali zahteva upotrebu naprednijih algoritama iz teorije ortogonalnih polinoma.

Od izuzetne važnosti nam je sledeća teorema.

Teorema 5.2 (Čebiševljevi diskretni ortogonalni polinomi) *Niz polinoma Q_k^n, $k = 0, \ldots, n$, definisan sa*

$$\lambda_{k+1} Q_{k+1}^n(x) = \left(x - \frac{n}{2}\right) Q_k^n(x) - \lambda_k Q_{k-1}^n(x), \quad Q_{-1}^n(x) = 0, \quad Q_0^n(x) = \frac{1}{\sqrt{n+1}}, \tag{5.2}$$

gde je $\lambda_k > 0$, $k = 0, \ldots, n$, zadato sa

$$\lambda_k^2 = \begin{cases} n+1, & k = 0, \\ \dfrac{1}{4} \dfrac{k^2((n+1)^2 - k^2)}{4k^2 - 1}, & k > 0, \end{cases}$$

zadovoljava sledeći identitet

$$\sum_{k=0}^{n} Q_\ell^n(k) Q_\nu^n(k) = \delta_{\ell\nu}. \tag{5.3}$$

Ovu teoremu nećemo dokazivati. Jednačina (5.2) se naziva tročlana rekurentna relacija, dok se osobina (5.3) naziva ortogonalnost niza polinoma Q_k^n, $k = 0, \ldots, n$. Sam niz polinoma se u ovom slučaju naziva niz ortonormalnih Čebiševljevih diskretnih polinoma. Na primer, za $n = 10$, prvih nekoliko članova niza polinoma su

$$Q_0^{10}(x) = \frac{1}{\sqrt{11}}, \quad Q_1^{10}(x) = \frac{1}{\sqrt{110}}(x - 5), \quad Q_2^{10}(x) = \frac{1}{\sqrt{858}} \left(x^2 - 5x + 10\right),$$
$$Q_3^{10}(x) = \frac{\sqrt{5}}{6\sqrt{858}} \left(x^3 - 15x^2 + \frac{286}{3}x - 36\right).$$

Primećujemo da je stepen polinoma Q_k^n baš jednak k. Ova osobina važi za svako $k = 0, \ldots, n$, i za svako $n \in \mathbb{N}$. Jednostavno se može dokazati indukcijom. Ova osobina je važna jer pokazuje da je skup polinoma Q_k^n, $k = 0, \ldots, n$, baza prostora $\mathcal{P}_n$.

Pretpostavimo da je skup tačaka u ravni zadat apscisama $x_i = x_0 + ih$, $i = 0, \ldots, n$. Ideja je u sledećem: umesto da tražimo aproksimaciju skupa podataka u obliku

$$F_m(p_0, \ldots, p_m) = \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} p_k x_i^k \right)^2,$$

mi aproksimaciju tražimo u obliku

$$F_m(p_0^*, \ldots, p_m^*) = \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} p_k^* Q_k^n(i) \right)^2. \tag{5.4}$$

Problemi su ekvivalentni, jer je skup polinoma Q_k^n, $k = 0, \ldots, m$, baza prostora polinoma $\mathcal{P}_m$ na isti način na koji je to i prirodna baza x^k, $k = 0, \ldots, m$. Međutim, ubrzo ćemo videti da se formulacije problema u numeričkom smislu jako razlikuju.

Teorema 5.3 *Neka je dat skup tačaka u ravni $(x_0 + ih, y_i)$, $i = 0, \ldots, n$. Funkcija*

$$F_m(p^*) = \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} p_k^* Q_k^n(i) \right)^2,$$

dostiže minimum u tački

$$p^* = (p_0^*, \ldots, p_m^*) = Q^T y, \quad Q = \begin{bmatrix} Q_0^n(0) & Q_1^n(0) & \cdots & Q_m^n(0) \\ Q_0^n(1) & Q_1^n(1) & \cdots & Q_m^n(1) \\ \vdots & \vdots & \ddots & \vdots \\ Q_0^n(n) & Q_1^n(n) & \cdots & Q_m^n(n) \end{bmatrix}.$$

Dokaz. Ako potražimo stacionarne tačke, dobijamo sistem linearnih jednačina

$$\frac{\partial F_m(p^*)}{\partial p_\ell^*} = -2 \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} p_k^* Q_k^n(i) \right) Q_\ell^n(i) = 0, \quad \sum_{k=0}^{m} p_k^* \sum_{i=0}^{n} Q_k^n(i) Q_\ell^n(i) = \sum_{i=0}^{n} Q_\ell^n(i) y_i.$$

Ako pažljivo pogledamo drugu sumu u drugoj jednakosti primetićemo da je to suma iz jednakosti (5.3). Primenom osobine ortogonalnosti niza polinoma Q_k^n, $k = 0, \ldots, n$, dobijamo

$$p_\ell^* = \sum_{k=0}^{m} p_k^* \delta_{k\ell} = \sum_{i=0}^{n} Q_\ell^n(i) y_i, \quad \ell = 0, \ldots, m.$$

Ovaj sistem jednačina zapisan u matričnoj formi izgleda baš kao u tvrđenju teoreme.

U stacionarnoj tački p^*, Hessian funkcije F_m ima vrednost

$$\frac{\partial^2 F_m(p^*)}{\partial p_k^* \partial p_\ell^*} = 2 \sum_{i=0}^{n} Q_k^n(i) Q_\ell^n(i) = 2\delta_{k\ell}, \quad k, \ell = 0, \ldots, m.$$

Odavde dobijamo $H = 2I_m$, gde je I_m jedinična matrica reda m. Zaključujemo da je Hessian pozitivno definitna matrica pa funkcija F_m dostiže minimum u stacionarnoj tački p^*. □

Ova teorema čini bitnu razliku. Ako uporedimo ovu teoremu i teoremu 5.1 vidimo da smo koristeći teoremu 5.1 morali da rešavamo sistem linearnih jednačina da bismo dobili stacionarnu

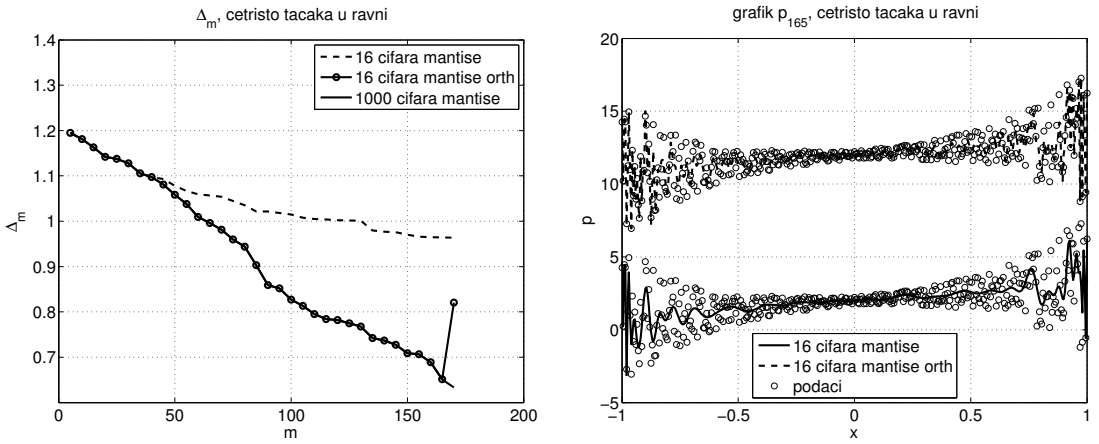
tačku funkcije F_m. Prema upravo dokazanoj teoremi nije potrebno rešavati nikakav sistem linearnih jednačina, ali je potrebno konstruisati matricu Q. Na sreću, matricu Q možemo jednostavno konstruisati, jer su njeni elementi vrednosti niza polinoma Q_k^n, $k = 0, \ldots, m$, u tačkama $0, \ldots, n$. Ove vrednosti izračunavamo koristeći tročlanu rekurentnu relaciju (5.2).

Implementacija prethodnog algoritma je sada prilično jednostavna i može biti data na sledeći način

```
function [res,err,p] = najmanjiKvadratiEkvidistant(m,a,b,n,y,x)
%NAJMANJIKVDRATIEKVIDISTANT(M,A,B,N,Y,X) izracunava polinom koji
% najbolje aproksimira skup tacaka u ravni (a+ih,y_i), i=0,...,n
% stepena m, h=(b-a)/n, u tackama x, gde je x vektor kolona
% res je vrednost polinoma u vektoru x, err je greska, a p su
% koeficijenti polinoma razvijenog po Cebisevljevim
% diskrentim ortogonalnim polinomima

q = chebyshevOrthMatrix(m,n);
p = q'*y;
err = norm(y-q*p)/sqrt(n);
q = chebyshevOrthPoly(m,a,b,n,x);
res = q*p;
end

function [ q ] = chebyshevOrthPoly( m,a,b,n,x )
%CHEBYSHEVORTHPOLY(M,A,B,N,X) izracunava vrednosti
% niza Cebisevljevih diskretnih ortogonalnih polinoma stepena 0 do m
% u vektoru x koji je vektor kolona, sa ortogonalnoscu u tackama
% a+i(b-a)/n, i=0,...,n

k = length(x); h = (b-a)/n;
t = (x-a)/h; q = [ones(k,1)/sqrt(n+1) zeros(k,m)];
gamma0 = .5*sqrt(n/3*(n+2));
q(:,2) = ((t-n/2).*q(:,1))/gamma0;
for i=2:m
    gamma1 = .5*i*sqrt((n+1-i)/(2*i-1)*(n+1+i)/(2*i+1));
    q(:,i+1) = ((t-n/2).*q(:,i)-gamma0*q(:,i-1))/gamma1;
    gamma0 = gamma1;
end
end

function [ q ] = chebyshevOrthMatrix( m,n )
%CHEBYSHEVORTHMATRIX( m,n ) konstruise matiricu Q
% sa vrstama [Q_0(i) Q_1(i) ... Q_m(i)], i=0,...,n

x = (0:n)';
q = [ones(n+1,1)/sqrt(n+1) zeros(n+1,m)];
gamma0 = .5*sqrt(n/3*(n+2));
q(:,2) = ((x-n/2).*q(:,1))/gamma0;
for i=2:m
```

```
    gamma1 = .5*i*sqrt((n+1-i)/(2*i-1)*(n+1+i)/(2*i+1));
    q(:,i+1)=((x-n/2).*q(:,i)-gamma0*q(:,i-1))/gamma1;
    gamma0 = gamma1;
end
end
```

Najjednostavnije je sve funkcije staviti u isti fajl sa imenom `najmanjiKvadratiEkvidistant.m`. Funkcija `chebyshevOrthMatrix` konstruiše matricu Q, funkcija `chebyshevOrthPoly` konstruiše vrednosti niza polinoma u konkretnoj tački `x`, dok funkcija `najmanjiKvadratiEkvidistant` konstruiše polinom p i izračunava njegove vrednosti u konkretnim tačkama koristeći prethodne dve funkcije.



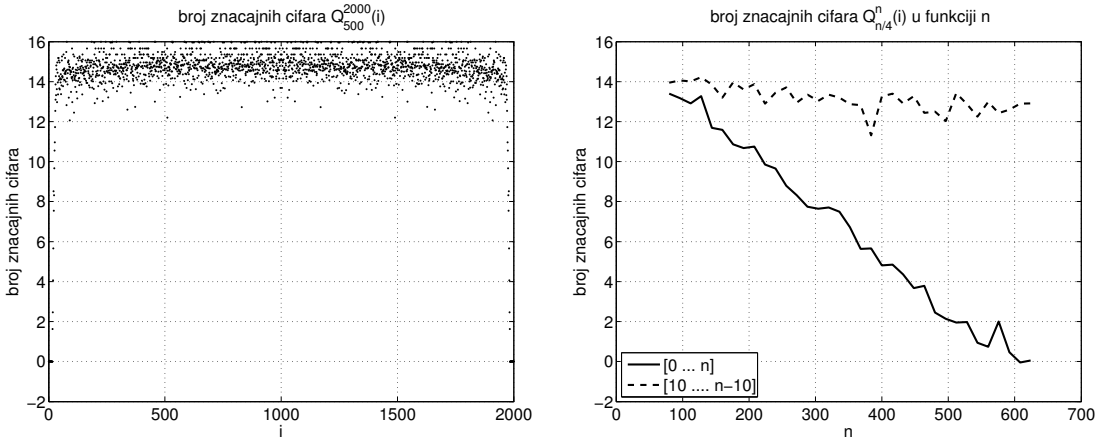
Slika 5.4: Slika levo ilustruje zavisnost Δ_m u funkciji m, slika desno ilustruje odnos grafika polinoma aproksimacije i tačaka u ravni za $m = 165$. Na slici desno, zbog preglednosti, grafik polinoma koji se dobija metodom izloženom u ovoj sekciji i podaci su tranlirani po y-osi za vrednost 10.

Sve što nam ostaje je da probamo kakve rezultate daje novi algoritam u poređenju sa starim algoritmom i u poređenju sa aritmetikom mantise proizvoljne dužine. Na osnovu dosadašnje analize je jasno da jedini problem koji može da nastane, leži u izračunavanju vrednosti niza polinoma Q_k^n, $k = 0, \ldots, m$, korišćenjem tročlane rekurentne relacije (5.2).

Slika 5.4, levo, ilustruje zavisnost Δ_m u funkciji m. Kao što vidimo to je ponovljena slika 5.2, levo, izuzev grafika funkcije Δ_m za slučaj konstrukcije novouvedenim algoritmom. Rezultati dobijeni novouvedenim algoritmom se slažu dobro sa rezultatima dobijenim upotrebom mantise dužine 1000 dekadnih cifara, do vrednosti $m = 165$. Zatim, rezultati počinju da se razilaze. Ovo je posledica toga što je za tako velike vrednosti m nemoguće precizno računati vrednosti niza polinoma Q_k^n, $k > 165$, u tačakama $i = 0, \ldots, n$. Ovo je pirodna granica upotrebljivosti algoritma. Slika 5.4, desno, ilustruje odnos dobijenih polinoma upotrebom oba algoritma sa podacima. Zbog preglednosti, podaci i grafik polinoma dobijen metodom ove sekcije prikazani su translirani po y-osi za vrednost 10. Kao što vidimo polinom dobijen algoritmom koji koristi Čebiševljev niz ortogonalnih polinoma bolje aproksimira podatke. Međutim, ponovimo još jednom da je i u ovom slučaju od izuzetnog značaja to što smo grafik polinoma crtali u tačkama $x_i = a + ih$, $i = 0, \ldots, n$. Da smo pokušali da predstavimo grafik van ovih tačaka došlo bi do značajnog iskakanja grafika polinoma van konveksnog omotača tačaka u ravni. Ovo se posebno odnosi na tačke koje se nalaze

blizu krajeva intervala ± 1.

Ilustracija problema koji nastaje prilikom izračunavanja vrednosti polinoma Q_k^n, $k = 0, \ldots, n$, u tačkama $0, \ldots, n$ prikazana je na slici 5.5. Slika levo ilustruje broj značajnih cifara u vrednostima $Q_{500}^{2000}(k)$, $k = 0, \ldots, 2000$, dok slika desno ilustruje minimum broja značajnih cifara u vrednostima $Q_{n/4}^n(i)$, $i = 0, \ldots, n$, i $Q_{n/4}^n(i)$, $i = 10, \ldots, n-10$. Kao što vidimo slaba uslovljenost izračunavanja nastaje jedino u tačkama koje su u blizini tačke 0 i tačke n.



Slika 5.5: Slika levo ilustruje zavisnost minimuma broja značajnih cifara u vrednostima $Q_{500}^{2000}(i)$, $i = 0, \ldots, 2000$, slika desno ilustruje zavisnost minimuma broja značajnih cifara u vrednostima $Q_{n/4}^n(i)$, $i = 0, \ldots, n$, i $Q_{n/4}^n(i)$, $i = 10, \ldots, n-10$.

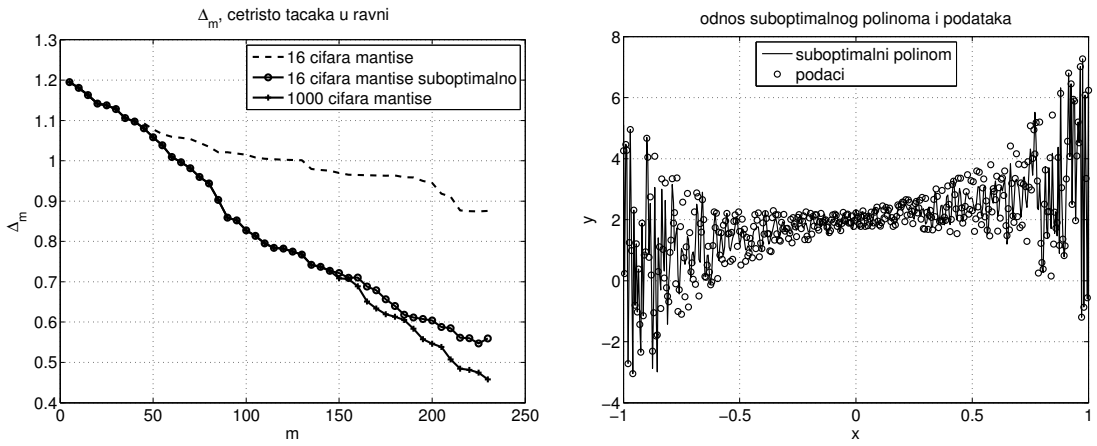
Ova razmatranja omogućavaju suboptimalno rešenje problema najmanjih kvadrata. Ideja se sastoji u eliminaciji vrednosti $Q_k^n(i)$ iz matrice Q koje su izračunate sa malim brojem značajnih cifara. Kako su to vrednosti koje su blizu krajnjim tačkama 0 i n, dovoljno je umesto originalnog problema sa tačkama $(x_0 + ih, y_i)$, $i = 0, \ldots, n$, posmatrati modifikovani problem $(x_0 - dh + ih, \tilde{y}_i)$, $i = 0, \ldots, n + 2d$, gde je

$$\tilde{y}_i = \begin{cases} 0, & i = 0, \ldots, d-1, \\ y_i, & i = d, \ldots, n+d, \\ 0, & i = n+d+1, \ldots, n+2d. \end{cases}$$

Za ovako postavljeni problem rešenje koje minimizuje funkciju F_m ima oblik

$$p^* = Q^T \tilde{y} = \overline{Q}^T y, \quad \overline{Q} = \begin{bmatrix} Q_0^{n+2d}(d) & Q_1^{n+2d}(d) & \cdots & Q_m^{n+2d}(d) \\ Q_0^{n+2d}(d+1) & Q_1^{n+2d}(d+1) & \cdots & Q_m^{n+2d}(d+1) \\ \vdots & \vdots & \ddots & \vdots \\ Q_0^{n+2d}(d+n) & Q_1^{n+2d}(d+n) & \cdots & Q_m^{n+2d}(d+n) \end{bmatrix}.$$

Primetimo da je ova konstrukcija moguća jer su odgovarajuće vrednosti ordinate tačaka modifikovanog problema jednake nuli. Naravno, ovako dobijeno rešenje neće biti rešenje originalnog problema, drugim rečima, neće biti tačka u kojoj funkcija F_m originalnog problema dostiže minimum, ali će biti omogućena konstrukcija rešenja u formatu dvostruke preciznosti koje neće preterano odstupati od optimalnog rešenja.



Slika 5.6: Slika levo ilustruje zavisnost Δ_m u funkciji m za 400 tačaka u ravni, slika desno ilustruje grafik funkcije Q_{230}^{418}.

Koristeći izloženi metod moguće je konstruisati suboptimalna rešenja još višeg stepena koristeći samo dvostruku preciznost. Slika 5.6, levo, prikazuje grafik funkcije Δ_m sa primenom ovakvog modifikovanog algoritma za već opisivan problem prikazan na prethodnim slikama. Primećujemo da smo ovim metodom u stanju da konstruišemo polinom stepena 230. Prilikom konstrukcije polinoma stepena 230 koristili smo $d = 9$. Slika 5.6, desno, prikazuje grafik polinoma Q_{230}^{418}. Primetimo da je bitno da parametar d ostane mali, inače suboptimalno rešenje može previše odstupati od stvarnog rešenja.

Prilikom konstrukcije suboptimalnog rešenja pogodno je vrednosti ordinata normalizovati na sledeći način $y_i = (y_i - \overline{y})/\sigma_y$, $i = 0, \ldots, n$, gde su $\overline{y}$ i σ_y srednja vrednost i varijansa niza y_i, $i = 0, \ldots, n$. Ovu transformaciju je pogodno koristiti jer na taj način eliminišemo preterani uticaj dodatih vrednosti niza $\tilde{y}_i$, $i = -d, \ldots, n + d$, čija je vrednost nula, a koje mogu bitno da utiču na izgled rešenja.

Prezentovana sekcija ilustruje važnost ortogonalnih polinoma u okviru numeričke analize. Veliki broj naprednih algoritama koristi ortogonalne polinome u ovom ili onom obliku.

5.2 Najmanji kvadrati sa težinama

Pretpostavimo da imate podatke koji su rezultat merenja i da za svaku pojedinačnu tačku pored apscise x_i, $i = 0, \ldots, n$, i ordinate y_i, $i = 0, \ldots, n$, imate i vrednost standardne devijacije σ_i, $i = 0, \ldots, n$, kojom su izvršena merenja. Manja vrednost standardne devijacije σ_i znači pouzdanije merenje y_i. U ovom slučaju želimo da polinom koji je rešenje problema najmanjih kvadrata prođe bliže onim tačkama koje imaju manje vrednosti standardne devijacije. Ovakvo rešenje problema najmanjih kvadrata možemo postići ako tačkama pridružimo težine $w_i > 0$, $i = 0, \ldots, n$, a funkciju F_m definišemo na sledeći način

$$F_m(p) = \sum_{i=0}^{n} w_i^2 \left(y_i - \sum_{j=0}^{m} p_j x_i^j \right)^2 .$$

U modelu sa standardnom devijacijom obično biramo $w_i = 1/\sigma_i$, $i = 0, \ldots, n$.

Teorema 5.4 (Problem najmanjih kvadrata sa težinama) *Tačka $p = (p_0, \ldots, p_m)$ u kojoj funkcija*

$$F_m(p) = \sum_{i=0}^{m} w_i^2 \left(y_i - \sum_{k=0}^{n} p_k x_i^k \right)^2$$

dostiže minimum je rešenje sistema linearnih jednačina

$$(WV)^T WVp = (WV)^T Wy, \quad W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}, \tag{5.5}$$

gde je matrica V data kao u teoremi 5.1.

Sistem linearnih jednačina (5.5) *ima jedinstveno rešenje.*

Dokaz. Postupajući kao u dokazu teoreme 5.1, dobijamo

$$\frac{\partial F_m(p)}{\partial p_j} = -2 \sum_{i=0}^{n} w_i^2 \left(y_i - \sum_{k=0}^{m} p_k x_i^k \right) x_i^j = 0.$$

Ovo je sistem linearnih jednačina

$$\sum_{i=0}^{n} (w_i x_i^j) \sum_{k=0}^{m} (w_i x_i^k) p_k = \sum_{i=0}^{n} (w_i x_i^j)(w_i y_i).$$

Ovaj sistem linearnih jednačina može se zapisati u matričnoj formi

$$(WV)^T WVp = (WV)^T Wy.$$

Ako pronađemo Hessian, dobijamo

$$\frac{\partial^2 F_m(p)}{\partial p_k \partial p_j} = 2 \sum_{i=0}^{n} w_i^2 x_i^k x_i^j.$$

Odavde možemo zaključiti da važi

$$H = \left[\frac{\partial^2 F_m(p)}{\partial p_k \partial p_j} \right] = 2(WV)^T WV.$$

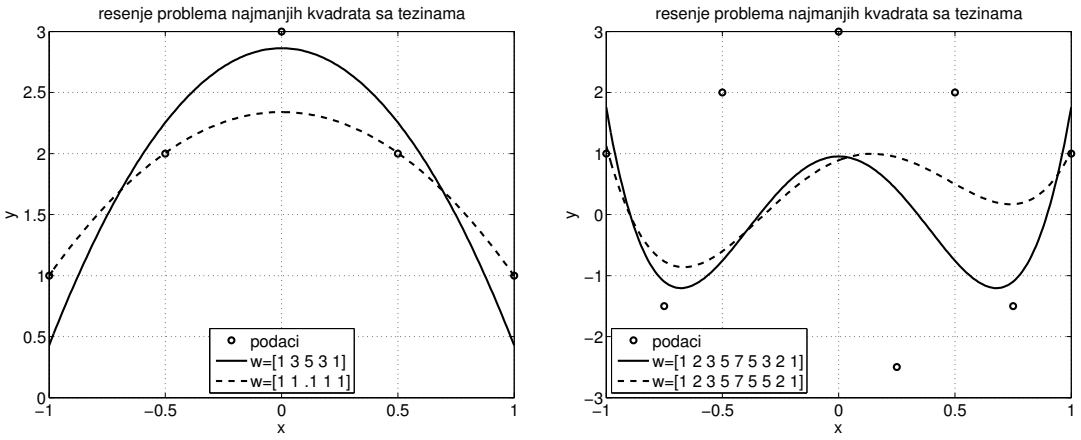
Ova matrica ima pun rang, jer matrica V ima rang m, a matrice W su regularne, jer imaju determinante koje nisu jednake nuli. Odavde zaključujemo da jednačina $Hb = 0$ ima jedno jedino rešenje $b = 0$. Međutim, za $b \neq 0$, imamo

$$(b, Hb) = b^T H^T b = 2b^T (WV)^T WVb = 2(WVb)^T (WVb) = 2(WVb, WVb) > 0.$$

Zaključujemo da je matrica H pozitivno definitna pa funkcija F_m ima minimum u svojoj stacionarnoj tački.

Sistem linearnih jednačina ima jedinstveno rešenje jer je, na osnovu prethodnog izlaganja, matrica sistema pozitivno definitna, pa samim tim invertibilna. □

Numerička konstrukcija rešenja problema najmanjih kvadrata se može postići relativno jednostavno. Potrebne su samo male modifikacije funkcije `najmanjiKvadrati`. Funkcija koja implementira ove promene može se dati na sledeći način.



Slika 5.7: Slika levo ilustruje uticaj težina na rešenje problema najmanjih kvadrata, slika desno ilustruje osetljivost polinoma na promenu tezina.

```
function [ res, err ] = najmanjiKvadratiTezine( m,c,v,w,x )
%NAJMANJIKVADRATITEZINE(M,C,V,W,X) konstruise resenje problema najmanjih
% kvadrata, u skupu polinom P_m, za skup tacaka u ravni
% (c(i),v(i)), i=0,...,n u tacki x, sa vektorom tezina w
% izlazni parametar res sadrzi koeficijente polinoma,
% dok err sadrzi velicinu greske
% srednje kvadratne aproksimacije

n = length(c)-1;
V = zeros(n+1,m+1);
V(:,1) = w';
for j=2:(m+1)
    V(:,j) = V(:,j-1).*(c');
end
p = V\((w').*(v'));
res = polyval(flipud(p),x);
err = norm((w').*(v'-V*p))/sqrt(n);
end
```

Slika 5.7, levo, ilustruje uticaj težina na rešenje problema najmanjih kvadrata. Linija crtana stilom crta je rešenje problema najmanjih kvadrata sa vektorom težina `w=[1 1 .1 1 1]`, dok je linija crtana stilom `'--'` dobijena vektorom težina `w=[1 3 5 3 1]`. Podaci su zadati sa `x=[-1 -.5 0 .5 1]` i `y=[1 2 3 2 1]`. Vidimo da povećanje težine tačke povlači grafik polinoma rešenja problema najmanjih kvadrata ka toj tački, dok ga od ostalih tačaka, sa manjim težinama, udaljava.

Međutim, treba napomenuti da je konstrukcija sa težinama jako osetljiva jer deluje globalno na izgled polinoma. Drugim rečima, promena jedne težine, naročito ako je n relativno veliko, proizvodi promenu grafika polinoma p u svim tačkama. Nemoguće je podešavati grafik polinoma tačku po tačku. Na slici 5.7, desno, prikazani su grafici rešenja problema najmanjih kvadrata sa

težinama `w=[1 2 3 5 7 5 3 2 1]` i `w=[1 2 3 5 7 5 5 2 1]`, grafici su prikazani stilovima `'--'` i crta, redom. Skup podataka je

```
x=[-1 -.75 -.5 -.25 0 .25  .5 .75  1];  y=[ 1 -1.5  2  -2.5 3 -2.5  2 -1.5 1]
```

a stepen polinoma je četiri. Vidimo da promena težine u jednoj tački, tački $x_6 = .5$, izaziva globalnu promenu u izgledu polinoma. Naravno, ovaj metod ne može da odgovori potrebama kompjuterskog modelovanja u čije svrhe bismo upotrebili Bezierove krive ili splajnove.

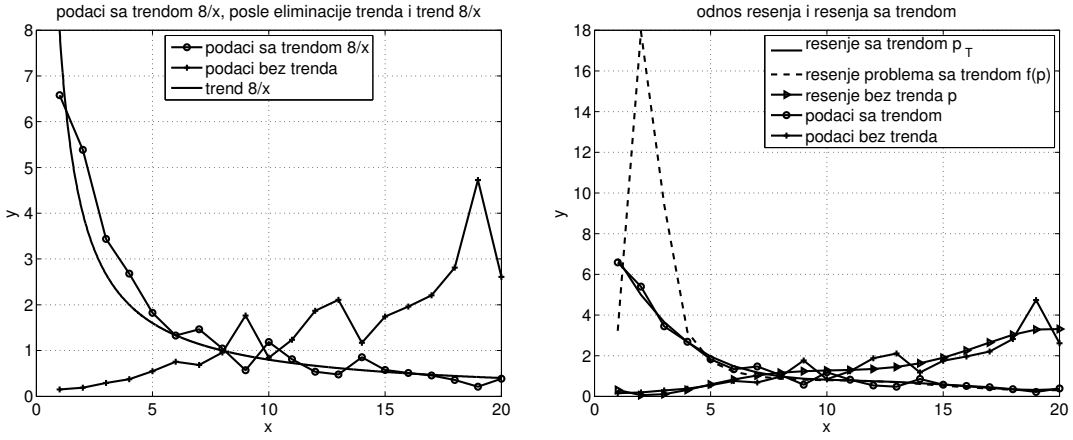
Pomenimo da je i ovde moguće koristiti ortogonalne polinome prilikom numeričke konstrukcije rešenja problema najmanjih kvadrata. Međutim, u ovom slučaju zbog proizvoljnosti težina w_i, $i = 0, \ldots, n$, nije moguće dati jednostavan izraz za tročlanu rekurentnu relaciju kao u slučaju kad su težine jednake jedan. Korišćenje ortogonalnih polinoma u ovom slučaju zahteva znanja koja prevazilaze obim ovog udžbenika.

5.3 Uopšteni problem najmanjih kvadrata

5.3.1 Problemi sa trendom

Posmatrajmo problem aproksimacije tačaka u ravni dat na slici 5.8, levo, grafik predstavljen stilom `'-o'`. Na slici možemo uočiti trend u ponašanju podataka. Naime, ako zanemarimo varijacije vidimo da se podaci ponašaju približno kao funkcija $y = f(x) = 8/x$, čiji je grafik takođe prikazan na pomenutoj slici. Ovo zapažanje sugeriše da problem aproksimacije podataka može imati najmanju kompleksnost ako podatke pokušamo da aproksimiramo funkcijom

$$p(x) = f\left(\sum_{k=0}^{m} p_k x^k\right) = \frac{8}{\sum_{k=0}^{m} p_k x^k}.$$



Slika 5.8: Slika levo ilustruje podatke sa trendom $f(x) = 8/x$, trend f i podatke bez trenda, slika desno ilustruje podatke sa i bez trenda, polinom koji aproksimira podatke sa trendom p_T, polinom koji aproksimira podatke bez trenda p i funkciju $f(p(\cdot))$ koja aproksimira podatke sa trendom.

Međutim, ako pokušamo sa ovako definisanom funkcijom da formulišemo problem najmanjih kvadrata dobijamo

$$F_m(p) = \sum_{i=0}^{n} \left(y_i - f\left(\sum_{k=0}^{m} p_k x_i^k \right) \right)^2 .$$

Ako za ovako formulisanu funkciju F_m pokušamo da odredimo stacionarne tačke dobićemo nelinearni sistem jednačina koji se teško rešava.

Ovde postoji jednostavan način da problem ostane u domenu linearnih jednačina. Naime, dovoljno je prepoznati činjenicu da funkcija $f(x) = 8/x$ ima inverznu funkciju $f^{-1}(x) = 8/x$. Ova činjenica nam omogućava reformulaciju problema u sledećem obliku

$$F_m(p) = \sum_{i=0}^{n} \left(f^{-1}(y_i) - \sum_{k=0}^{m} p_k x_i^k \right)^2 .$$

U suštini umesto da posmatramo originalni problem (x_i, y_i), $i = 0, \ldots, n$, možemo posmatrati njegovu sliku $(x_i, f^{-1}(y_i))$, $i = 0, \ldots, n$, iz koje je očigledno trend odstranjen i ostavljen je samo varijacioni deo problema.

Naravno, ništa nije ekskluzivno u vezi sa funkcijom $f(x) = 8/x$ izuzev činjenice da poseduje inverznu funkciju. Prezentovani metod se može sa uspehom primeniti sa svakim skupom podataka u kome uočavamo trend u obliku neke funkcije f koja ima inverznu funkciju f^{-1}. I više, dovoljno je da je funkcija f bijekcija samo svedena na interval $[\min\{x_i \,|\, i = 0, \ldots, n\}, \max\{x_i \,|\, i = 0, \ldots, n\}]$.

Funkciju f ćemo zvati funkcija trenda. Problem (x_i, y_i), $i = 0, \ldots, n$, ćemo zvati skup podataka sa trendom, dok ćemo $(x_i, f^{-1}(y_i))$, $i = 0, \ldots, n$, zvati skup podataka bez trenda. Rešenje problema najmanjih kvadrata skupa podataka sa trendom ćemo zvati rešenje sa trendom i označavaćemo ga sa p_T. Dakle, polinoma p_T je rešenje sledećeg problema najmanjih kvadrata

$$F_m(p_T) = \sum_{i=0}^{m} \left(y_i - \sum_{k=0}^{m} p_k^T x_i^k \right)^2 .$$

Rešenje problema najmanjih kvadrata skupa podataka bez trenda $(x_i, f^{-1}(y_i))$, $i = 0, \ldots, n$, ćemo označavati kao do sada sa p. Polinom p ćemo zvati rešenje bez trenda. Primetimo da je p_T polinom, dok je $f(p(\cdot))$ rešenje problema skupa podataka sa trendom koje nije polinom, već ima uračunatu funkciju trenda.

Na primer, za skup podataka $(1 + i, 8/(1 + i)(1 + \sigma_i))$, $i = 0, \ldots, 19$, gde su σ_i slučajni brojevi uniformno raspodeljeni na intervalu $[-.5, .5]$, koji su prikazani na slici 5.8, levo, funkcijom `najmanjiKvadrati` možemo kostruisati aproksimaciju polinomom stepena 10 sledećim skriptom

```
c=1:1:20;
v=(8./c).*(1+.5*(2*rand(1,20)-1));
[pT,err1]=najmanjiKvadrati(5,c,v,c');
[p,err2]=najmanjiKvadrati(5,c,8./v,c');
```

Ovde prvi poziv funkcije najmanji kvadrati konstruiše rešenje p_T problema najmanjih kvadrata skupa podataka sa trendom, dok drugi poziv konstruiše rešenje p problema najmanjih kvadrata skupa podataka bez trenda.

Slika 5.8, desno, ilustruje grafike skupova podataka sa i bez trenda, kao i grafike polinoma p i p_T i funkcije $f(p(\cdot)) = 8/p(\cdot)$. Primećujemo jednu važnu osobinu koja važi generalno. Naime, iako je aproksimacija u prostoru bez trenda dobra, kad se posmatra u prostoru sa trendom, dakle, kad

upoređujemo skup podataka sa trendom i funkciju $f(p(\cdot))$ može doći do velikih odstupanja. Ova osobina je ilustrovana na prikazanim graficima.

Kao funkcije trenda često se koriste i funkcije $f(x) = e^x$ i $f(x) = \log x$, jer imaju inverzne funkcije a i često se javljaju u primenama.

5.3.2 Uopšteni polinomi

Drugo uopštenje problema najmanjih kvadrata ide u pravcu napuštanja polinoma kao aproksimacionih elemenata. Naime, umesto da aproksimaciju skupa podataka tražimo u obliku polinoma, rešenje problema tražimo u obliku uopštenog polinoma zadatog nad skupom funkcija φ_k, $k = 0, \ldots, m$.

Neka je zadat skup tačaka u ravni (x_i, y_i), $i = 0, \ldots, n$, i niz funkcija φ_k, $k = 0, \ldots, m$. Aproksimaciju skupa podataka tražimo u obliku uopštenog polinoma

$$p(x) = \sum_{k=0}^{m} p_k \varphi_k(x),$$

pri čemu zahtevamo ono rešenje koje minimizira funkciju

$$F_m(p) = \sum_{i=0}^{n} w_i^2 \left(y_i - \sum_{k=0}^{m} p_k \varphi_k(x_i) \right)^2.$$

Da bismo mogli da dokažemo da rešenje problema postoji neophodno je da niz funkcija bude Čebiševljev sistem.

Teorema 5.5 (Uopšteni problem najmanjih kvadrata) *Neka je (x_i, y_i), $i = 0, \ldots, n$, $x_i \in [a, b]$, $i = 0, \ldots, n$, skup tačaka u ravni koje zadovoljavaju uslov $x_i \neq x_j$ za $i \neq j$, neka su $w_i > 0$, $i = 0, \ldots, m$, težine i neka je φ_k, $k = 0, \ldots, m$, Čebiševljev sistem funkcija na intervalu $[a, b]$. Funkcija*

$$F_m(p) = \sum_{k=0}^{n} w_i^2 \left(y_i - \sum_{k=0}^{m} p_k \varphi_k(x_i) \right)^2,$$

dostiže minimum u tački p koja zadovoljava sistem linearnih jednačina

$$(WV)^T WVp = (WV)^T Wy, \quad V = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_m(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_m(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_m(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_m(x_n) \end{bmatrix},$$

gde je matrica W kao u teoremi 5.4.

Sistem linearnih jednačina ima jedistveno rešenje $p = ((WV)^T WV)^{-1} (WV)^T Wy$.

Dokaz. Stacionarne tačke funkcije F_m nalazimo jednostavno. Važi

$$\frac{\partial F_m(p)}{\partial p_j} = -2 \sum_{i=0}^{n} w_i^2 \left(y_i - \sum_{k=0}^{m} p_k \varphi_k(x_i) \right) \varphi_j(x_i) = 0.$$

Odavde dobijamo sistem linearnih jednačina

$$(WV)^T WVp = (WV)^T Wy, \quad V = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_m(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_m(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_m(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_m(x_n) \end{bmatrix},$$

gde smo sa W označili dijagonalnu matricu koja na glavnoj dijagonali ima elemente w_i, $i = 0, \ldots, n$, redom.

Elementi matrice Hessiana iznose

$$\frac{\partial^2 F_m(p)}{\partial p_j \partial p_\nu} = 2 \sum_{i=0}^{n} w_i^2 \varphi_j(x_i) \varphi_\nu(x_i).$$

Odavde zaključujemo da važi

$$H = \left[\frac{\partial^2 F_m}{\partial p_j \partial p_\nu} \right] = 2(WV)^T WV.$$

Međutim, matrica V je punog ranga, jer je niz funkcija φ_k, $k = 0, \ldots, m$, Čebiševljev sistem. Matrica W je takođe punog ranga jer važi $w_i \neq 0$, $i = 0, \ldots, n$. Na osnovu ovoga, za $b \neq 0$ važi

$$(b, Hb) = (Hb)^T b = (2(WV)^T WVb)^T b = 2(WVb)^T (WVb) = 2(WVb, WVb) > 0,$$

jer $WVb \neq 0$, za $b \neq 0$. Zaključujemo da je matrica H pozitivno definitna, pa je stacionarna tačka tačka u kojoj se dostiže minimum.

Sistem linearnih jednačina $Hp = 2(WV)^T WVp = 2(WV)^T Wy$, ima jedinstveno rešenje jer je matrica H pozitivno definitna i samim tim invertibilna. Naravno, rešenje sistema jednačina je $p = (WV^T WV)^{-1}(WV)^T Wy$. □

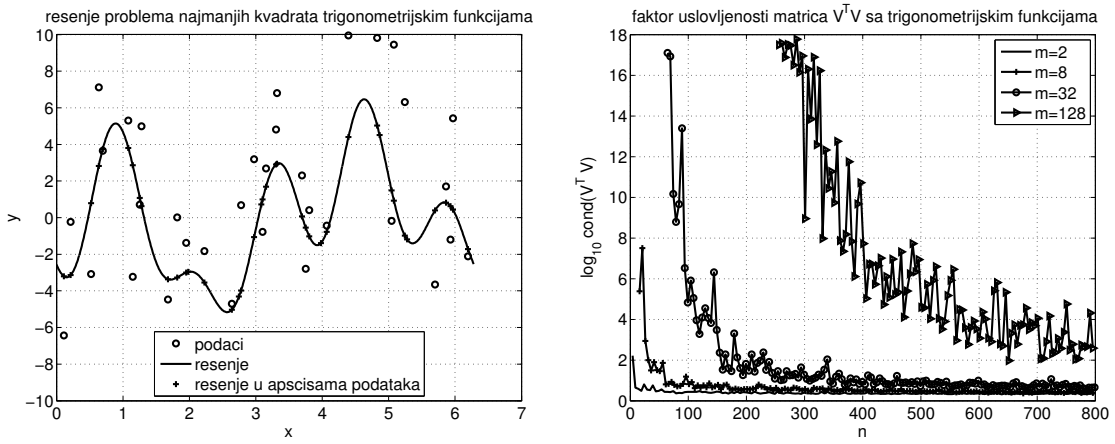
U daljem tekstu ćemo posmatrati slučaj $w_i = 1$, $i = 0, \ldots, n$.

Na primer, niz funkcija $\varphi_{2k}(x) = \cos kx$, $k = 0, \ldots, m$, i $\varphi_{2k-1}(x) = \sin kx$, $k = 1, \ldots, m$, je Čebiševljev sistem na $[0, 2\pi)$ na osnovu izračunavanja determinante 4.15. Na osnovu ovoga možemo potražiti rešenje problema najmanjih kvadrata u obliku

$$p(x) = \sum_{k=0}^{m} a_k \cos kx + b_k \sin kx.$$

Ako izaberemo slučajno raspoređene apscise u intervalu $[0, 2\pi)$ i ordinate, možemo koristiti sledeći skript za konstrukciju koeficijenata rešenja problema najmanjih kvadrata.

```
n=39; m=5;
c=(2*pi*rand(1,n+1))';
v=(10*(2*rand(1,n+1)-1))';
V=[cos(kron(0:m,c)) sin(kron(1:m,c))];
p=V\v;
x=(0:.01:2*pi)';
mat=[cos(kron(0:m,x)) sin(kron(1:m,x))];
poly=mat*p;
```



Slika 5.9: Slika levo ilustruje odnos trigonometrijskog polinoma kao rešenja problema najmanjih kvadrata i podataka, slika desno ilustruje zavisnost faktora uslovljenosti matrice V^TV u funkciji broja tačaka n za stepen polinoma m uzet kao parametar.

```
cc=sort(c);
matC=[cos(kron(0:m,cc)) sin(kron(1:m,cc))];
polyC=matC*p;
plot(c,v,'o',x,poly,'-',cc,polyC,'+');
```

Kao što vidimo upotrebljeno je 40 tačaka u ravni i upotrebljeno je 11 funkcija. Slika 5.9, levo, prikazuje sliku koja se dobija pozivom funkcije `plot` na kraju skripta. Prikazan je grafik uopštenog polinoma stilom crta i vrednosti uopštenog polinoma u apscisama podataka.

U slučaju metoda najmanjih kvadrata sa uopštenim nizom funkcija problem faktora uslovljenosti ostaje u važnosti. Međutim, za konkretan slučaj trigonometrijskih funkcija faktor uslovljenosti matrice V^TV ima zanimljvo ponašanje. Na primer, ako biramo slučajno 800 apscisa na intervalu $[0, 2\pi)$ faktor uslovljenosti matrice V^TV u funkciji n se menja kao na slici 5.9, desno. Kao što vidimo faktor uslovljenosti opada sa porastom broja tačaka n, i više, faktor uslovljenosti matrice postaje reda veličine 10.

Ovo ponašanje faktora uslovljenosti matrica V^TV sa apscisama koje su uniformno raspoređene na intervalu $[0, 2\pi)$ daju opravdanje da se svaki problem najmanjih kvadrata sa trigonometrijskim funkcijama prvo linearno preslika na interval $[0, 2\pi)$, pa zatim rešava.

5.4 Linearna regresija i funkcija regress

Metod linearne regresije je u suštini metod najmanjih kvadrata, samo što je metod najmanjih kvadrata ogrnut u koncept teorije verovatnoće i statistike.

Teorijska postavka u okviru statistike izgleda ovako. Neka je dat skup podataka (X^i, Y_i), $i = 0, \ldots, n$, gde je $X = (X_0, \ldots, X_m)$ vektor koji predstavlja skup obeležja ili prediktora od kojih zavisi posmatrana veličina Y. Veličinu Y u ovom kontekstu često nazivamo i izlazom. Skup podataka (X^i, Y_i), $i = 0, \ldots, n$, u ovom kontekstu obično nazivamo opservacijama. Vrednost prediktora X_0 je za sve opservacije obično fiksirana i iznosi jedan. Vrednost prediktora X_k u opservaciji i obeležavamo sa X_k^i.

Problem linearne regresije je onda sledeći: za zadati skup opsevacija (X^i, Y_i), $i = 0, \ldots, n$, odrediti koeficijente p_k, $k = 0, \ldots, m$, za koje funkcija

$$r(p) = \sum_{i=0}^{n} \left(Y_i - \sum_{k=0}^{m} p_k X_k^i \right)^2,$$

dostiže minimum. Funkcija r se u ovom kontekstu naziva sumom kvadrata ostataka ili reziduala, dok se veličine

$$r_i = Y_i - \sum_{k=0}^{m} p_k X_k^i, \quad i = 0, \ldots, n,$$

nazivaju rezidualima.

Bez obzira na kontekst rešenje problema je isto kao kod običnog metoda najmanjih kvadrata i može se iskazati sledećim sistemom linearnih jednačina

$$V^T V p = V^T y, \quad V = \begin{bmatrix} X_0^0 & X_1^0 & \cdots & X_m^0 \\ X_0^1 & X_1^1 & \cdots & X_m^1 \\ \vdots & \vdots & \ddots & \vdots \\ X_0^n & X_1^n & \cdots & X_m^n \end{bmatrix}.$$

Međutim, ovde očigledno nailazimo na problem. Naime, ništa ne garantuje da sistem linearnih jednačina ima jedinstveno rešenje. Čak, ništa ne garantuje da sistem jednačina ima rešenje.

`Matlab` ima implementiranu funkciju koja metodom najmanjih kvadrata rešava problem linearne regresije. To je funkcija `regress`. Ova funkcija ima sledeći format

```
[p,pint,r,rint,stats]=regress(y,x)
```

Ulazni parametar `x` predstavlja matricu prediktora, u čijoj i-toj vrsti se nalazi i-ta opservacija. Da bi funkcija mogla ispravno da radi neophodno je da prvi prediktor ima uvek vrednost jedan. Vektor kolona `y` predstavlja vrednosti izlaza u svakoj opservaciji. Funkcija ima više izlaznih parametara. Parametar `p` je rešenje problema najmanjih kvadrata, parametar `r` je vektor ostataka, dok se ostali parametri bave statističkim opisivanjem valjanosti primenjenog modela. Parametar `pint` predstavlja intervale pouzdanosti svakog pojedinačnog koeficijenta u rešenju problema najmanjih kvadrata, dok parametar `rint` predstavlja intervale pouzdanosti svakog pojedinačnog reziduala. Intervali pouzdanosti koeficijenata rešenja se koriste za određivanje pouzdanosti koeficijenata rešenja. Sa ovako definisanim formatom funkcije `regress`, interval pouzdanosti koeficijenata rešenja znači da svaka naredna konstrukcija regresionih koeficijenata sa verovatnoćom od .95 proizvodi vrednost koeficijenata rešenja unutar datog intervala pouzdanosti. Ovakva interpretacija intervala pouzdanosti je od posebnog značaja u slučaju kad se problem posmatra unutar statistike, u kom slučaju prediktori mogu biti slučajne promenljive. Slično važi i za intervale pouzdanosti reziduala. Ako neki od intervala pouzdanosti reziduala ne uključuje broj nula kažemo za odgovarajuću opservaciju da je izvan modela (engleski outlier). U suštini ako interval pouzdanosti reziduala ne sadrži vrednost nula znači da model linearne regresije nije primenljiv na zadati skup opservacija. Izlazni parametar `stats` sadrži statističke parametre koji se mogu koristiti za ocenu valjanosti modela linearne regresije. Vektor `stats` sadrži redom R^2 statistiku ili koeficijent determinisanosti, F statistiku i p vrednost, kao i varijansu greške.

Označimo sa $\overline{Y}$ srednju vrednost izlaza

$$\overline{Y} = \frac{1}{n+1} \sum_{i=0}^{n} Y_i.$$

Vrednost koeficijenta determinisanosti se izračunava

$$R^2 = 1 - \frac{\sum_{i=0}^{n} r_i^2}{\sum_{i=0}^{n}(Y_i - \overline{Y})^2},$$

i koristi se da proceni neobjasnjenu varijansu. Što je ova vrednost bliža jedinici to je manja količina neobjašnjene varijanse u izlazu Y. U upotrbi je i takozvana modifikovana vrednost koeficijenta determinisanosti

$$R^2 = 1 - \frac{n}{n-m}\frac{\sum_{i=0}^{n} r_i^2}{\sum_{i=0}^{n}(Y_i - \overline{Y})^2},$$

koja uzima u obzir broj stepena slobode.

F statistika se izračunava po formuli

$$F = \frac{n-m}{m}\frac{\sum_{i=0}^{n}(\overline{Y} - \sum_{k=0}^{m} p_k X_k^i)^2}{\sum_{i=0}^{n} r_i^2}.$$

Vrednost ovog parametra meri statistički značaj linearnog modela za dati skup opservacija u odnosu na model kod koga je $p_k = 0$, $k = 1, \ldots, m$. Pretpostavka da su svi koeficijenti $p_k = 0$, $k = 1, \ldots, m$, se obično naziva nulta hipoteza. Što veća vrednost F statistike to se sa više verodostojnosti može prihvatiti linearni model. Vrednot p je verovatnoća da se dobije dobijena vrednost F statistike pri čemu je vrednost izlaza aproksimirana konstantom.

Varijansa greške se izračunava po formuli

$$\frac{1}{n-m}\sum_{i=0}^{n} r_i^2.$$

Veća vrednost varijanse znači da model sadrži veću količinu neobjašnjene varijanse.

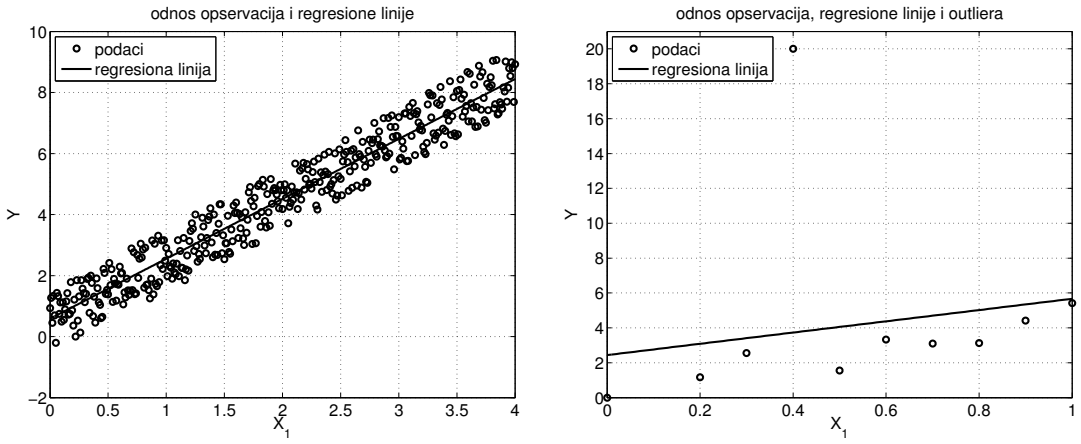
Radi ilustracije posmatrajmo problem sa dva prediktora. U ovom slučaju jedan ima fiksnu vrednost jednaku jedan, dok se drugi menja. Model je sledećeg oblika

$$Y = p_0 X_0 + p_1 X_1.$$

Izaberimo skup opservacija koji je linearna funkcija opterećena šumom koji ima uniformnu raspodelu na intervalu $[-1, 1]$. Opišimo opservacije sledećim skupom podataka $(1, .01 \cdot i, .5 + .02 \cdot i + \sigma_i)$, $i = 0, \ldots, 400$, gde su σ_i, $i = 0, \ldots, 400$, slučajne promenljive koje imaju uniformnu raspodelu sa vrednostima iz intervala $[-1, 1]$.

Jedna realizacija ovog skupa podataka je prikazana na slici 5.10, levo. Na osnovu sledećeg niza komandi možemo dobiti regresione koeficijente kao i statistike koje opisuju linearnu regresiju.

```
>>n=400;
>>x=[ones(n+1,1) .01*(0:1:n)'];
>>y=.5+2*x(:,2)+2*rand(n+1,1)-1;
>>[p,pint,r,rint,stats]=regress(y,x);
>>stats
stats =
  Columns 1 through 2
        0.944111998542444          6740.27837092228
  Columns 3 through 4
     5.05431479558852e-252         0.324312576697059
```



Slika 5.10: Slika levo ilustruje odnos opservacija i regresione linije, slika desno ilustruje odnos opservacija, regresione linije i pogrešnih opservacija (outliera).

Dajmo interpretaciju pojedinih statistika. Vrednost R^2 statistike je jako bliska jedinici. Na osnovu ove vrednosti R^2 statistike možemo reći da je 94% varijabilnosti podataka objašnjeno pomoću prediktora. Vrednost F statistike je velika i ova vrednost se može postići sa parametrima jednakim nula sa verovatnoćom reda veličine 10^{-252}. Ovako mala vrednost verovatnoće znači da je verodostojnost podataka izuzetna. Varijansa ima relativno malu vrednost.

Objasnimo još i pojam loše opservarcije (engleski outlier). Posmatrajmo primer prikazan na slici 5.10, desno. Na ovoj slici se vidi da opservacija $i = 5$ odstupa od modela. Posmatrajmo sledeći skript

```
n=10;
x=[ones(n+1,1) (0:1:n)'/10];
y=.5+4*x(:,2)+(2*rand(n+1,1)-1);
y(5)=20;
[p,pint,r,rint,stats]=regress(y,x);
```

Posle izvršenja ovog skripta, nalazimo

```
>>rint(5,:)
ans =
          14.4060532778745            17.7682552161272
```

Interval pouzdanosti reziduala opservacije $i = 5$ ne uključuje vrednost nula, što nam ukazuje da je opservacija pogrešna. Naravno, postoji i drugo objašnjenje da je model pogrešan a opservacija ispravna. Međutim, ukoliko veliki broj opservacija zadovoljava model izuzev jedne, sa većom verovatnoćom možemo tvrditi upravo suprotno, da je model ispravan a opservacija pogrešna.

Glava 6

Numeričko diferenciranje i integracija

6.1 Numeričko diferenciranje

Pod numeričkim diferenciranjem funkcije podrazumevamo diferenciranje funkcije koja je zadata skupom vrednosti nad nekim diskretnim skupom tačaka. Podrazumevaćemo da je funkcija f zadata vrednostima $f(x_k)$, $k = 0, \ldots, n$, u tačkama x_k, $k = 0, \ldots, n$, koje zadovoljavaju uslov $x_0 < x_1 < \cdots < x_n$.

6.1.1 Numeričko diferenciranje upotrebom interpolacionog polinoma

S obzirom na način kako je zadata funkcija, prirodno se nameće metod diferenciranja koji se zasniva na interpolaciji. Za funkciju f konstruišemo interpolacioni polinom P_n i vrednost izvoda funkcije dobijemo diferenciranjem polinoma

$$f'(x) \approx P_n'(x).$$

Međutim, ovde je od izuzetne važnosti ne dospeti u poziciju da je potrebno eksplicitno razviti interpolacioni polinom, jer je proces razvijanja interpolacionog polinoma iz neke baze u prirodnu bazu slabo uslovljen numerički proces tako da dolazi do gubitka značajnih cifara.

Mi ćemo u ovom odeljku obraditi izvod Lagrangeovog interpolacionog polinoma dok ćemo se izvodima ostalih interpolacionih polinoma više ili manje baviti u narednim odeljcima.

U slučaju da treba diferencirati Lagrangeov interpolacioni polinom u interpolacionim tačkama možemo eksplicitno dati izraze koji određuju izvode.

Teorema 6.1 (Izvod Lagrangeovog interpolanta) *Neka je P_n Lagrangeov interpolant konstruisan u interpolacionim čvorovima x_i, $i = 0, \ldots, n$. Onda je izvod u tački x_m, određen sa*

$$P_n'(x_m) = \sum_{\ell=0, \ell \neq m}^{n} \left(\frac{\omega'(x_m)}{\omega'(x_\ell)} f(x_\ell) + f(x_m) \right) \frac{1}{x_m - x_\ell}.$$

Dokaz. Polazeći od izraza za Lagrangeov interpolant

$$P_n(x) = \omega(x) \sum_{\ell=0}^{n} \frac{f(x_\ell)}{\omega'(x_\ell)} \frac{1}{x - x_\ell}$$

i nalaženjem izvoda u tački x_m, nalazimo

$$\begin{aligned} P_n'(x_m) &= \omega'(x_m) \sum_{\ell=0,\ell\neq m}^{n} \frac{f(x_\ell)}{\omega'(x_\ell)} \frac{1}{x_m - x_\ell} + \frac{f(x_m)}{\omega'(x_m)} ((x-x_0)\cdots(x-x_{m-1})(x-x_{m+1})\cdots(x-x_n))|'_{x=x_m} \\ &= \omega'(x_m) \sum_{\ell=0,\ell\neq m}^{n} \frac{f(x_\ell)}{\omega'(x_\ell)} \frac{1}{x_m - x_\ell} + \frac{f(x_m)}{\omega'(x_m)} \sum_{\ell=0,\ell\neq m}^{n} \frac{\omega'(x_m)}{x_m - x_\ell} \\ &= \sum_{\ell=0,\ell\neq m}^{n} \left(\frac{\omega'(x_m)}{\omega'(x_\ell)} f(x_\ell) + f(x_m) \right) \frac{1}{x_m - x_\ell}. \end{aligned}$$

Time smo završili dokaz. □

Formula je prilično jednostavna za implementaciju i može se implementirati, recimo, na sledeći način

```
function [ res ] = izvodLagrangeInterpolant( m, cvorovi, vrednosti )
%IZVODLAGRANGEINTERPOLANT(M,CVOROVI,VREDNOSTI) izracunava
% izvod u m-tom interpolacionom cvoru Lagrangeovog interpolacionog
% polinoma

xm = cvorovi(m+1); n = length(cvorovi); res = 0; prod = ones(1,n);
for ell = 1:n
    for j = [1:ell-1 ell+1:n]
        prod(ell) = prod(ell)*(cvorovi(ell)-cvorovi(j));
    end
end
for ell = [1:m m+2:n]
    prod(ell) = prod(m+1)/prod(ell)*vrednost(ell)+vrednosti(m+1);
    res = res+prod(ell)/(xm-cvorovi(ell));
end
end
```

Na primer, možemo upotrebiti implementiranu funkciju da odredimo izvod funkcije log u nekoj tački unutar intervala $(2, 7)$, koristeći deset tačaka.

```
>>c=2+5*rand(1,10);
>>v=log(c);
>>aIzvod=izvodLagrangeInterpolant(4,c,v)
aIzvod =
      0.216985380538036
>>izvod=1/c(5)
      0.216985341570706
>>abs(aIzvod-izvod)/izvod
ans =
      1.79585080354104e-07
```

Vidimo da je relativna greška reda veličine 10^{-7}. Da smo upotrebili veći broj tačaka dobili bismo bolju aproksimaciju izvoda. Već sa dvadeset tačaka može se dobiti greška reda veličine mašinske preciznosti.

Prilikom diferenciranja interpolacionog polinoma moguće je odrediti grešku koja nastaje pod uslovom da tražimo izvod u interpolacionim čvorovima.

Teorema 6.2 *Neka je P_n interpolacioni polinom funkcije f, konstruisan u interpolacionim čvorovima x_k, $k = 0, \ldots, n$. Onda važi*

$$f'(x_k) = P_n'(x_k) + \frac{1}{(n+1)!}\omega'(x_k)f^{(n+1)}(\psi), \quad \psi \in (x_0, x_n).$$

Dokaz. U stvari dokaz formule se zasniva na teoremi 4.3. Ako diferenciramo izraz

$$f(x) = P_n(x) + \frac{1}{(n+1)!}\omega(x)f^{(n+1)}(\psi), \quad \psi \in (x_0, x_n),$$

nalazimo

$$f'(x) = P_n'(x) + \frac{1}{(n+1)!}\omega'(x)f^{(n+1)}(\psi) + \frac{1}{(n+1)!}\omega(x)\frac{d}{dx}f^{(n+1)}(\psi), \quad \psi \in (x_0, x_n).$$

Iako ne znamo da odredimo izvod $(n+1)$-vog izvoda funkcije f, jer ne znamo kako se menja argument ψ u funkciji od x, srećna okolnost je da i ne moramo da znamo. Ako pretpostavimo da je $x = x_k$, onda je $\omega(x_k) = 0$, tako da nam problematični član nestaje iz jednačine i dobijamo tvrđenje teoreme. □

6.1.2 Formule za jednostrano diferenciranje

Pretpostavimo da se funkcija f može razviti u stepeni red u nekoj okolini tačke x i neka je h dovoljno malo tako da važi formula

$$f(x+h) = \sum_{k=0}^{+\infty} \frac{f^{(k)}(x)}{k!}h^k.$$

Uvedimo operator diferenciranja D, rekurzivno, na sledeći način $(Df)(x) = f'(x)$, $(D^{k+1}f)(x) = (D(D^k f))(x)$. Dogovorno još usvojimo da važi $(D^0 f)(x) = f(x)$. Onda upravo napisani razvoj u red možemo predstaviti na sledeći način

$$Ef = \sum_{k=0}^{+\infty} \frac{h^k D^k f}{k!} = \sum_{k=0}^{+\infty} \frac{h^k D^k}{k!} f = e^{hD} f.$$

Odakle dobijamo zanimljivu formulu $1 + \Delta = E = e^{hD}$. Ako logaritmujemo prethodnu jednakost nalazimo

$$D = \frac{1}{h}\log(1+\Delta) = \frac{1}{h}\sum_{k=1}^{+\infty} \frac{(-1)^{k-1}}{k}\Delta^k.$$

Ako primenimo formulu na funkciju f u tački x, nalazimo

$$(Df)(x) = f'(x) = \frac{1}{h}\sum_{k=1}^{+\infty} \frac{(-1)^{k-1}}{k}\Delta^k f(x).$$

Drugim rečima, našli smo familiju formula za numeričko diferenciranje

$$f'(x) = \frac{1}{h}\left(\Delta f(x) - \frac{1}{2}\Delta^2 f(x) + \cdots + \frac{(-1)^{k-1}}{k}\Delta^k f(x)\right) + R_k(f).$$

Radi kratkoće pisanja koristićemo notaciju

$$(D_n^+ f)(x) = \frac{1}{h}\left(\Delta f(x) - \frac{1}{2}\Delta^2 f(x) + \cdots + \frac{(-1)^{n-1}}{n}\Delta^n f(x)\right).$$

Kao što vidimo u formuli učestvuju vrednosti funkcije desno od tačke u kojoj određujemo izvod. Vidimo takođe da učestvuju vrednosti funkcije u ekvidistantnim tačkama, koje su pomerene jedna u odnosu na drugu za vrednost koraka podele h. Vrednosti koje su nam potrebne da bismo numerički aproksimirali izvod mogu se odrediti koristeći tablicu konačnih razlika, koju smo već imali prilike da koristimo i konstruišemo prilikom konstrukcije interpolacionih polinoma.

Na primer, koristeći dobijenu formulu, odredimo približno vrednost izvoda funkcije $f(x) = \sin x$ u tački $x = \pi/4$, koristeći podelu sa korakom $h = .01$.

Konstruišemo tablicu konačnih razlika

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$
0	.785398	$\underline{.707107}$				
			$\underline{.703559(-2)}$			
1	.795398	.714142		$\underline{-.714136(-4)}$		
			.696418(−2)		$\underline{-.696412(-6)}$	
2	.805398	.721107		−.721101(−4)		$\underline{0.721095(-8)}$
			.689207(−2)		−.689201(−6)	
3	.815398	.727999		−.727993(−4)		
			.681927(−2)			
4	.825398	.734818				

Na osnovu ove tablice konačnih razlika, koristeći podatke koji su podvučeni, možemo konstruisati aproksimacije sa 2, 3, 4 i 5 tačaka. Približne vrednosti izvoda su određene sa

$$f'(x) \approx (D_1^+ f)(x) = \frac{1}{h}\Delta f(x) = .\underline{70}3559,$$
$$f'(x) \approx (D_2^+ f)(x) = \frac{1}{h}\left(\Delta f(x) - \frac{1}{2}\Delta^2 f(x)\right) = .\underline{7071}30,$$
$$f'(x) \approx (D_3^+ f)(x) = \frac{1}{h}\left(\Delta f(x) - \frac{1}{2}\Delta^2 f(x) + \frac{1}{3}\Delta^3 f(x)\right) = .\underline{707107},$$
$$f'(x) \approx (D_4^+ f)(x) = \frac{1}{h}\left(\Delta f(x) - \frac{1}{2}\Delta^2 f(x) + \frac{1}{3}\Delta^3 f(x) - \frac{1}{4}\Delta^4 f(x)\right) = .\underline{707107}.$$

Kao što vidimo već formula koja sadrži četiri tačke omogućava određivanje izvoda sa šest značajnih cifara.

Sledeća teorema daje mogućnost da se oceni tačnost formule za numeričko diferenciranje.

Teorema 6.3 *Neka je $f \in C^{n+1}[x, x+nh]$, onda je*

$$f'(x) = (D_n^+ f)(x) + \frac{(-1)^n}{n+1}h^n f^{(n+1)}(\psi), \quad \psi \in (x, x+nh).$$

n	D_n^+	Δ_n	R_n
1	.707106746489217	.347(−7)	.500(−7)
2	.707106782016353	.830(−9)	.333(−14)
3	.707106781646279	.460(−9)	.250(−21)
4	.707106780536056	.650(−9)	.200(−28)
5	.707106778315610	.287(−8)	.167(−35)
6	.707106774244792	.694(−8)	.143(−42)
7	.707106766790438	.144(−7)	.125(−49)
8	.707106752912650	.283(−7)	.111(−56)
9	.707106727007446	.542(−7)	.100(−63)
10	.707106679378878	.102(−6)	.909(−71)
11	.707106593891705	.187(−6)	.833(−78)
12	.707106444011597	.337(−6)	.769(−85)
13	.707106185671239	.596(−6)	.714(−92)
14	.707105744119682	.104(−5)	.667(−99)
15	.707104989686130	.179(−5)	.625(−106)
16	.707103693223193	.309(−5)	.588(−113)
17	.707101443846037	.534(−5)	.556(−120)
18	.707097495831283	.929(−5)	.526(−127)
19	.707090482786167	.163(−4)	.500(−134)
20	.707077889248843	.289(−4)	.476(−141)
21	.707055099489346	.517(−4)	.455(−148)
22	.707013759578054	.930(−4)	.435(−155)
23	.706939133523925	.168(−3)	.417(−162)

Tabela 6.1: Vrednost aproksimacije izvoda $(D_n^+ \sin)(\pi/4)$, apsolutne greške Δ_n, i teorijski očekivane greške, za korak podele $h = 10^{-7}$.

Dokaz. Može se pokazati da se formula za diferenciranje dobija diferenciranjem prvog Newtonovog interpolacionog polinoma u čvoru x_0. Na osnovu teoreme 6.2 dovoljno je samo za ekvidistatno izabrane interpolacione čvorove x_k, $k = 0, \ldots, n$, odrediti vrednost $\omega'(x_0)$. Dobijamo

$$\omega'(x_0) = (x_0 - x_1) \cdots (x_0 - x_n) = (-h) \cdots (-nh) = (-1)^n n! h^n.$$

Time je dokaz teoreme završen. □

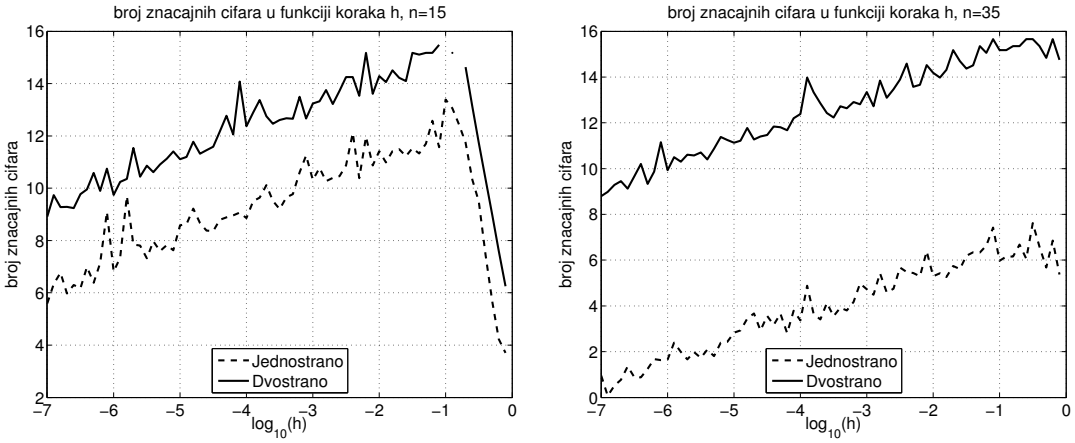
Na osnovu ocene ostatka možemo govoriti o redu formule. Naime, stepen koraka h u ostatku formule se obično naziva redom formule. Tako formula $D_n^+ f$ ima red n, a za konstrukciju formule potrebno je poznavati vrednosti funkcije u $(n+1)$-noj tački.

Kao što vidimo učinjena greška je tim manja što je manji korak podele. Idealno, greška bi bila jednaka nuli, ako je korak podele jednak nula. Zato je logičan zahtev da se primenjuju formule sa što je moguće manjim h. Sa druge strane previše mali korak dovodi do pojave oduzimanja bliskih brojeva u tabeli konačnih razlika, tako da je sa gledišta ograničene dužine mantise neprihvatljivo da korak podele bude previše mali.

Koliko problem određivanja veličine koraka h može postati ozbiljan ilustrovaćemo na već demonstriranom primeru određivanja izvoda funkcije sin u tački $\pi/4$. Tabela 6.1 sadrži aproksimaciju izvoda $(D_n^+ \sin)(\pi/4)$, apsolutnu grešku Δ_n i očekivanu teorijsku apsolutnu grešku R_n prema teoremi 6.3. Korak koji se koristi pri određivanju izvoda iznosi $h = 10^{-7}$. Kao što vidimo u početku,

za mali broj tačaka, imamo bolje aproksimacije nego za veći broj tačaka. Takođe, vidimo da su dobijene apsolutne greške u odnosu na greške koje očekujemo teorijski daleko veće. Ovo ponašanje se vrlo jednostavno objašnjava. Naime, kako je korak mali, to su vrednosti funkcije sin u tačkama x_k i x_{k+1} jako bliske po vrednosti, zato dolazi do gubitka značajnih cifara prilikom konstrukcije tabele konačnih razlika. Pojava je već opisana kod konstrukcije interpolacionih polinoma, što veći red konačne razlike koju konstruišemo to veću grešku unosimo u tabelu konačnih razlika.

Slika 6.1 levo i desno prikazuje zavisnost broja značajnih cifara u funkciji koraka h prilikom numeričkog određivanja $(D_n^+ \sin)(\pi/4)$. Slika levo prikazuje zavisnost za 15 tačaka, a slika desno za 35 tačaka. Tačnost kojom su zadati podaci funkcije je reda veličine mašinske preciznosti. Kao što vidimo sa smanjenjem koraka h dolazi do pada broja značajnih cifara, što je posledica opisanih pojava gubitka značajnih cifara prilikom oduzimanja bliskih brojeva u tablici konačnih razlika. Povećanje broja tačaka u formuli ne dovodi po povećanja tačnosti formule, jer formula sa 15 tačaka postiže 12 značajnih cifara za $h = .1$, dok formula sa 35 tačaka postiže 7 značajnih cifara za $h = .1$. Za $n = 15$, sa smanjenjem h dolazi do povećanja broja značajnih cifara, do vrednosti $h = .1$. Dalje smanjenje h dovodi do gubitka značajnih cifara.



Slika 6.1: Slika levo prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$ sa 15 tačaka, slika desno prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$ sa 35 tačaka.

Operator pomeranja E možemo izraziti i koristeći operator zadnje razlike ∇, $E^{-1} = 1 - \nabla$. Ako ovu formulu iskoristimo možemo operator diferenciranja izraziti preko operatora zadnje razlike ∇. Dobijamo

$$(1 - \nabla)^{-1} = E = e^{hD},$$

a odavde

$$D = -\frac{1}{h}\log(1 - \nabla) = \frac{1}{h}\sum_{k=1}^{+\infty}\frac{1}{k}\nabla^k.$$

Na ovaj način dobijamo familiju formula za numeričko diferenciranje koje koriste operator zadnje razlike ∇. Za razliku od formula koje koriste operator prednje razlike Δ, formule koje koriste operator zadnje razlike ∇ koriste samo vrednosti funkcije u tačkama koje se nalaze levo od tačke

u kojoj se traži izvod funkcije. Radi kratkoće pisanja uvodimo oznaku

$$(D_n^- f)(x) = \frac{1}{h}\sum_{k=1}^{n}\frac{1}{k}\nabla^k f(x). \tag{6.1}$$

Teorema koja opisuje u potpunosti ponašanje formule za diferenciranje izgleda ovako.

Teorema 6.4 *Neka je $f \in C^{n+1}[x-nh, x]$, onda je*

$$f'(x) = (D_n^- f)(x) + \frac{1}{n+1}h^n f^{(n+1)}(\psi), \quad \psi \in (x-nh, x).$$

Dokaz. Može se pokazati, direktnim diferenciranjem, da je formula za diferenciranje izvod drugog Newtonovog interpolacionog polinoma u čvoru x_n. Za određivanje ostatka se može koristiti teorema 6.2 za ekvidistatno izabrane čvorove x_k, $k = 0, \ldots, n$, pri čemu određujemo ostatak u čvoru x_n. □

Što se tiče numeričkog ponašanja, formula se ponaša potpuno istovetno kao i formula za diferenciranje sa operatorom prednje razlike i nećemo se zadržavati na tome.

Primetimo da je red formule D_n^- isti kao red formule D_n^+ i iznosi n. Takođe, za postizanje reda n potrebno je poznavati vrednosti funkcije u $(n+1)$-noj tački.

Daćemo još jedino implementaciju u `Matlab`u funkcija koje se mogu koristiti za diferenciranje sa operatorima prednje i zadnje razlike.

```
function [ res ] = izvodDesni( n, h, x, fun )
%IZVODDESNI(N,H,X,FUN) izracunava vrednost prvog izvoda
% funkcije fun u tacki X, koristeci formulu za jednostrano
% diferenciranje udesno sa korakom h i brojem tacaka N

tabela = fun(x:h:x+n*h);
tabela = diff(tabela);
res = 0; minus = 1; k = 1;
while not(isempty(tabela))
    res = res + minus*tabela(1)/k;
    minus = -minus;
    k = k+1;
    tabela = diff(tabela);
end
res = res/h;
end

function [ res ] = izvodLevi( n, h, x, fun )
%IZVODLEVI(N,H,X,FUN) izracunava vrednost prvog izvoda
% funkcije fun u tacki X, koristeci formulu za jednostrano
% diferenciranje ulevo sa korakom h i brojem tacaka N

tabela = fun(x-n*h:h:x);
tabela = diff(tabela);
res = 0; k = 1;
while not(isempty(tabela))
    res = res + tabela(end)/k;
```

```
    k = k+1;
    tabela = diff(tabela)
end
res = res/h;
end
```

Formule za numeričko diferenciranje, sa operatorom prednje razlike, se mogu predstaviti i u razvijenom obliku za malu vrednost broja tačaka n. Uz notaciju $f_k = f(x_k)$, dobijamo sledeće formule

$$f'_k = \frac{1}{h}(f_{k+1} - f_k) - \frac{h}{2}f''(\psi), \quad \psi \in (x_k, x_{k+1}),$$
$$f'_k = \frac{1}{2h}(-3f_k + 4f_{k+1} - f_{k+2}) + \frac{h^2}{3}f^{(3)}(\psi), \quad \psi \in (x_k, x_{k+2}),$$
$$f'_k = \frac{1}{6h}(-11f_k + 18f_{k+1} - 9f_{k+2} + 2f_{k+3}) - \frac{h^3}{4}f^{(4)}(\psi), \quad \psi \in (x_k, x_{k+3}).$$

Ove formule imaju značaj u praksi. Naime, postoje slučajevi kada je neophodno aproksimirati izvode funkcije na osnovu vrednosti funkcije u tačkama podele. Ovo je posebno slučaj prilikom rešavanja parcijalnih diferencijalnih jednačina. Najčešće vrednosti funkcije u tačkama podele nisu poznate već se rešava sistem linearnih jednačina kako bi se odredile vrednosti funkcije u tačkama podele. U ovom slučaju su ove formule podesne za primenu jer omogućavaju vrlo efikasno formiranje sistema linearnih jednačina.

Na sličan način se mogu razviti i formule za diferenciranje sa operatorom zadnje razlike. Ove formule izgledaju ovako

$$f'_k = \frac{1}{h}(f_k - f_{k-1}) + \frac{h}{2}f''(\psi), \quad \psi \in (x_{k-1}, x_k),$$
$$f'_k = \frac{1}{2h}(3f_k - 4f_{k-1} + f_{k-2}) + \frac{h^2}{3}f^{(3)}(\psi), \quad \psi \in (x_{k-2}, x_k),$$
$$f'_k = \frac{1}{6h}(11f_k - 18f_{k-1} + 9f_{k-2} - 2f_{k-3}) + \frac{h^3}{4}f^{(4)}(\psi), \quad \psi \in (x_{k-3}, x_k).$$

Naravno, formule imaju smisla jedino u slučaju kad je funkcija dovoljan broj puta diferencijabilna. Ove formule imaju upotrebu i pri konstrukciji metoda za numeričku integraciju običnih diferencijalnih jednačina.

Prilikom konstrukcije ovih eksplicitnih izraza obično se koristi sledeća lema.

Lema 6.1 *Za svako $n \in \mathbb{N}_0$ važi*

$$\Delta^n f(x) = \sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k} f(x+kh), \quad \nabla^n f(x) = \sum_{k=0}^{n}(-1)^{k}\binom{n}{k} f(x-kh),$$
$$f(x+nh) = \sum_{k=0}^{n}\binom{n}{k}\Delta^k f(x), \quad f(x-nh) = \sum_{k=0}^{n}(-1)^{k}\binom{n}{k}\nabla^k f(x).$$

Dokaz. Dovoljno je primeniti binomnu formulu na jednakosti $\Delta^n = (E-1)^n$, $\nabla^n = (1-E^{-1})^n$, $E^n = (1+\Delta)^n$ i $E^{-n} = (1-\nabla)^n$. □

6.1.3 Formule za dvostrano diferenciranje

U ovom odeljku nam je cilj da odredimo formule za aproksimaciju izvoda funkcije koje koriste vrednosti funkcije i sa leve i sa desne strane u odnosu na tačku u kojoj tražimo izvod funkcije. Kao i u slučaju interpolacionih polinoma, u ovom slučaju je neophodno koristiti operator centralne razlike δ i operator usrednjavanja μ.

Pretpostavimo da je funkcija f realna i analitička, dakle, može se razviti u red u okolini tačke x, i neka je h dovoljno mali broj tako da važe sledeći razvoji

$$f\left(x-\frac{h}{2}\right)=\sum_{k=0}^{+\infty}\frac{f^{(k)}(x)}{k!}\left(-\frac{h}{2}\right)^k=\sum_{k=0}^{+\infty}\frac{1}{k!}\left(-\frac{hD}{2}\right)^k f(x)=e^{-hD/2}f(x), \tag{6.2}$$

$$f\left(x+\frac{h}{2}\right)=\sum_{k=0}^{+\infty}\frac{f^{(k)}(x)}{k!}\left(\frac{h}{2}\right)^k=\sum_{k=0}^{+\infty}\frac{1}{k!}\left(\frac{hD}{2}\right)^k f(x)=e^{hD/2}f(x). \tag{6.3}$$

Oduzimenjem ovih jednakosti nalazimo

$$\delta f(x)=f\left(x+\frac{h}{2}\right)-f\left(x-\frac{h}{2}\right)=(e^{hD/2}-e^{-hD/2})f(x)$$

ili

$$\delta=e^{hD/2}-e^{-hD/2}.$$

Ovu jednakost možemo rešiti po D. Kao rešenje dobijamo

$$D=\frac{2}{h}\log\left(\frac{1}{2}\delta+\left(1+\frac{1}{4}\delta^2\right)^{1/2}\right)=\frac{1}{h}\sum_{k=0}^{+\infty}(-1)^k\frac{\prod_{\ell=1}^{k}(2\ell-1)^2}{2^{2k}(2k+1)!}\delta^{2k+1},$$

gde podrazumevamo da važi $\prod_{\ell=1}^{0}(2\ell-1)^2=1$. Odavde dobijamo familiju formula za numeričko diferenciranje koje možemo koristeći lemu 4.7 i jednakost

$$\delta^{2k+1}f(x)=\Delta^{2k+1}f\left(x-\left(k+\frac{1}{2}\right)h\right),$$

da napišemo na sledeći način

$$\begin{aligned}f'(x)&=\frac{1}{h}\sum_{k=0}^{n}(-1)^k\frac{\prod_{\ell=1}^{k}(2\ell-1)^2}{2^{2k}(2k+1)!}\delta^{2k+1}f(x)+R_n(f)\\&=\frac{1}{h}\sum_{k=0}^{n}(-1)^k\frac{\prod_{\ell=1}^{k}(2\ell-1)^2}{2^{2k}(2k+1)!}\Delta^{2k+1}f\left(x-\left(k+\frac{1}{2}\right)h\right)+R_n(f).\end{aligned}$$

Ova familija formula ima očigledan nedostatak. Naime, da bismo mogli da odredimo izvod u tački x, neophodno je da poznajemo vrednosti funkcije u tačkama $f(x-(k+1/2)h)$, $k\in\mathbb{Z}$. Ovo su tačke koje ne pripadaju podeli, već se nalaze između tačaka koje pripadaju podeli. Međutim, koristeći ovu familju formula moguće je odrediti izvod u tačkama međupodele, recimo $x+h/2$, koristeći vrednosti funkcije u tačkama podele.

Stavimo li $x:=x+h/2$ u prethodnu familiju formula nalazimo

$$f'\left(x+\frac{h}{2}\right)=\frac{1}{h}\sum_{k=0}^{n}(-1)^k\frac{\prod_{\ell=1}^{k}(2\ell-1)^2}{2^{2k}(2k+1)!}\Delta^{2k+1}f\left(x-kh\right)+R_n(f).$$

Na primer, odredimo izvod funkcije sin u tački $x = \pi/4$ sa podelom koja je konstruisana sa korakom $h = .01$. Prvo je neophodno formirati tablicu konačnih razlika. Koristimo vrednosti funkcije u tačkama $x_{-1} = \pi/4 - h/2 - h$, $x_0 = \pi/4 - h/2$, $x_1 = \pi/4 + h/2$ i $x_2 = \pi/4 + h/2 + h$

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$
-1	.770398	.696421			
			.714139(−2)		
0	.780398	.703562		−.703557(−4)	
			.707104(−2)		−.707098(−6)
1	.790398	.710633		−.710628(−4)	
			.699998(−2)		
2	.800398	.717633			

Podvučene vrednosti su one koje su nam potrebne u formuli. Na osnovu raspoloživih podataka dobijamo sledeću aproksimaciju izvoda

$$\begin{aligned}\cos(\pi/4) &\approx \frac{1}{h}\left(\Delta f(x - h/2) - \frac{1^2}{2^2 \cdot 3!}\Delta^3 f(x - h/2 - h)\right) \\ &= 10^2\left(.707104 \cdot 10^{-2} - \frac{1}{32}(-.707098 \cdot 10^{-6})\right) \approx .\underline{707106}.\end{aligned}$$

Kao što vidimo aproksimacija je prilično dobra.

Implementacija ovog metoda diferenciranja u `Matlab`u može biti sledeća

```
function [ res ] = izvodDvostraniMedjupodela( n, h, x, fun )
%IZVODDVOSTRANIMEDJUPODELA(N,H,X,FUN) numericki odredjuje vrednost
% izvoda funkcije fun u tacki x, korsiteci korak podele h, sa n
% tacaka podeli sa obe strane tacke x

tabela = fun(x-(n-1)*h-h/2:h:x+n*h-h/2);
tabela = diff(tabela);
k = 0; res = 0; koef = 1;
while not(isempty(tabela))
    res = res+koef*tabela(ceil(length(tabela)/2));
    k = k+1;
    koef = -koef*(2*k-1)*(2*k-1)/(8*k*(2*k+1));
    tabela = diff(tabela,2);
end
res = res/h;
end
```

Nedostatak prethodne formule je što se može koristiti za dobijanje izvoda u tačkama koje ne pripadaju podeli, ako koristimo vrednosti funkcije u podeli, ili što se može koristiti za dobijanje izvoda u tačkama podele, ako koristimo vrednosti funkcije koje su van podele. Da bismo ovakvo ponašanje mogli da otklonimo moramo u kombinaciji sa operatorom δ iskoristiti operator μ.

Lema 6.2 *Važi*

$$\mu^2 - \frac{1}{4}\delta^2 = 1, \quad \mu = \left(1 + \frac{1}{4}\delta^2\right)^{1/2}.$$

Dokaz. Koristeći razvoje (6.2) dobili smo reprezentaciju $\delta = e^{hD/2} - e^{-hD/2}$. Na identični način možemo dobiti $2\mu = e^{hD/2} + e^{-hD/2}$. Odavde formalno možemo da pišemo $\delta = 2\sinh(hD/2)$ i $\mu = \cosh(hD/2)$. Tvrđenje leme postaje osnovni identitet za hiperboličke funkcije. □

Koristeći već dobijenu formulu za numeričko diferenciranje zasnovanu na centralnim razlikama, možemo da pišemo

$$D = \frac{2\mu}{h}\left(1+\frac{1}{4}\delta^2\right)^{-1/2}\log\left(\frac{1}{2}\delta+\left(1+\frac{1}{4}\delta^2\right)^{1/2}\right).$$

Razvijanjem u red, i koristeći lemu 4.7, nalazimo familiju formula za numeričko diferenciranje

$$\begin{aligned}
f'(x) &= \frac{\mu}{h}\sum_{k=0}^{+\infty}(-1)^k\frac{\prod_{\ell=1}^{k}\ell^2}{(2k+1)!}\delta^{2k+1}f(x)\\
&= \frac{1}{2h}\sum_{k=0}^{+\infty}(-1)^k\frac{\prod_{\ell=1}^{k}\ell^2}{(2k+1)!}\left(\Delta^{2k+1}f(x-kh)+\Delta^{2k+1}f(x-(k+1)h)\right)\\
&= \frac{1}{2h}\sum_{k=0}^{n-1}(-1)^k\frac{\prod_{\ell=1}^{k}\ell^2}{(2k+1)!}\left(\Delta^{2k+1}f(x-kh)+\Delta^{2k+1}f(x-(k+1)h)\right)+R_n(f).
\end{aligned}$$

Ovde takođe podrazumevamo da važi $\prod_{\ell=1}^{0}\ell^2 = 1$. Iskoristimo prethodnu formulu da odredimo izvod funkcije sin u tački $\pi/4$. U ovom slučaju, za razliku od prethodne formule, koristimo vrednosti funkcije u tačkama podele da bismo odredili izvod funkcije u tački podele. Prvo konstruišimo tablicu konačnih razlika.

k	x_k	$\Delta^0 f_k$	$\Delta^1 f_k$	$\Delta^2 f_k$	$\Delta^3 f_k$	$\Delta^4 f_k$
-2	.765398	.692824				
			.717630(−2)			
-1	.775398	.700000		−.699995(−4)		
			.710631(−2)		−.710625(−6)	
0	.785398	.707107		−.707101(−4)		−.707095(−8)
			.703559(−2)		−.703554(−6)	
1	.795398	.714142		−.714136(−4)		
			.696418(−2)			
2	.805398	.721107				

Na osnovu raspoloživih podataka možemo izračunati aproksimaciju izvoda

$$\begin{aligned}
f'(x) &\approx \frac{1}{2h}\left(\Delta f(x)+\Delta f(x-h)-\frac{1^2}{3!}\left(\Delta^3 f(x-h)+\Delta^3 f(x-2h)\right)\right)\\
&= 50\left(.710631\cdot 10^{-2}+.703559\cdot 10^{-2}-\frac{1}{6}\left(-.710625\cdot 10^{-6}+(-.703554\cdot 10^{-6})\right)\right)\approx \underline{.707107}
\end{aligned}$$

Aproksimacija je prilično dobra. Implementacija ovog metoda diferenciranja u `Matlabu` može biti ovakva

```
function [ res ] = izvodDvostrano( n, h, x, fun )
%IZVODDVOSTRANO(N,H,X,FUN) implementira metod dvostranog
```

```
% numerickog diferenciranja funkcije fun, sa n tacaka
% levo i desno do tacke x u kojoj trazimo izvod sa korakom
% podele h

tabela = fun(x-n*h:h:x+n*h);
tabela = diff(tabela);
k=0; koef = 1; res = 0;
while not(isempty(tabela))
    l = length(tabela)/2;
    res = res+koef*(tabela(l)+tabela(l+1));
    k = k+1;
    koef = -koef*k/(2*(2*k+1));
    tabela = diff(tabela,2);
end
res = res/(2*h);
end
```

Prikazana formula za numeričko diferenciranje se može dati i u razvijenom obliku. Ako označimo $f_k = f(x_k)$, onda prvih nekoliko aproksimacija izvoda možemo predstaviti na sledeći način

$$f'_k = \frac{1}{2h}(f_{k+1} - f_{k-1}) - \frac{1}{6}h^2 f^{(3)}(\psi), \quad \psi \in (x_{k-1}, x_{k+1}),$$
$$f'_k = \frac{1}{12h}(-f_{k+2} + 8f_{k+1} - 8f_{k-1} + f_{k-2}) + \frac{1}{30}h^4 f^{(5)}(\psi), \quad \psi \in (x_{k-2}, x_{k+2}).$$

Formule sadrže i izraze za grešku. Izrazi imaju smisla jedino u slučaju kad su funkcije dovoljan broj puta diferencijabilne.

Primetimo da u ovom slučaju red formule iznosi $2n$, međutim, lako se može primetiti da je potrebno poznavati samo $2n$ vrednosti funkcije da bi se primenile formule. Na ovaj način formule za dvostrano diferenciranje imaju malo veću efikasnost od odgovarajućih formula za jednostrano diferenciranje. To se posebno odnosi na formule koje koriste $n = 1$. Formule za jednostrano diferenciranje koje koriste dve vrednosti funkcije za aproksimaciju izvoda imaju red 1, za razliku od formule za dvostrano diferenciranje koja ima red 2. Zato se formula za aproksimaciju izvoda dvostranim metodom u praksi vrlo često koristi, ako to uslovi dozvoljavaju.

U opštem slučaju važi sledeća teorema.

Teorema 6.5 *Ako je $f \in C^{2n+1}[x_{-n}, x_n]$, onda je*

$$f'(x) = \frac{1}{2h}\sum_{k=0}^{n-1}(-1)^k \frac{\prod_{\ell=1}^{k}\ell^2}{(2k+1)!}\left(\Delta^{2k+1}f(x-kh) + \Delta^{2k+1}f(x-(k+1)h)\right) + \frac{(-1)^n(n!)^2}{(2n+1)!}h^{2n}f^{(2n+1)}(\psi),$$

gde je $\psi \in (x_{-n}, x_n)$.

Dokaz. Može se pokazati, direktnim diferenciranjem, da je formula za dvostrano diferenciranje izvod Stirlingovog interpolacionog polinoma u čvoru x_0. Za ocenu greške može se koristiti teorema 6.2, za ekvidistantne interpolacione čvorove x_k, $k = -n, \ldots, 0, \ldots, n$, pri čemu određujemo ostatak formule u čvoru x_0. □

Sve što je rečeno vezano za probleme smanjenja koraka u slučaju jednostranog diferenciranja važi i u slučaju dvostranog diferenciranja. Ovo je posledica toga što su problemi vezani za smanjenje

koraka u suštini vezani za konstrukciju tablice konačnih razlika, pa su samim tim nezavisni od metoda koji se koristi za diferenciranje. Međutim, slika 6.1 pokazuje da je formula za dvostrano diferenciranje numerički daleko stabilnija od formule za jednostrano diferenciranje. Vidimo na slikama levo i desno da sa porastom n ne dolazi do gubitka značajnih cifara, za razliku od formula za jednostrano diferenciranje. O ovoj temi ćemo govoriti iscrpnije u narednim odeljcima.

6.1.4 Uticaj tačnosti ulaznih podataka

Tačnost ulaznih podataka je od velike važnosti za tačno određivanje vrednosti izvoda funkcije. U ovom odeljku nećemo ulaziti u detaljnu analizu načina na koji greška zavisi od tačnosti ulaznih podataka, ali ćemo ilustrovati primerom kako tačnost ulaznih podataka utiče na vrednost aproksimacije izvoda.

Ilustraciju dajemo na već posmatranom primeru određivanja izvoda funkcije sin u tački $\pi/4$. Sledeća funkcija omogućava da vrednosti funkcije sin opteretimo šumom. Ovo će nam biti osnovni model na osnovu koga ćemo posmatrati uticaj greške u ulaznim podacima.

```
function [ res ] = randSin( x,aSuma )
%RANDSIN(X) racuna vrednosti funkcije sin u tacki x
% dodajuci sum uniformno raspodeljen na intervalu [-aSuma,aSuma]

res = sin(x)+aSuma*(2*rand(size(x))-1);
end
```

Menjajući parametar `aSuma` možemo varirati tačnost ulaznih podataka.

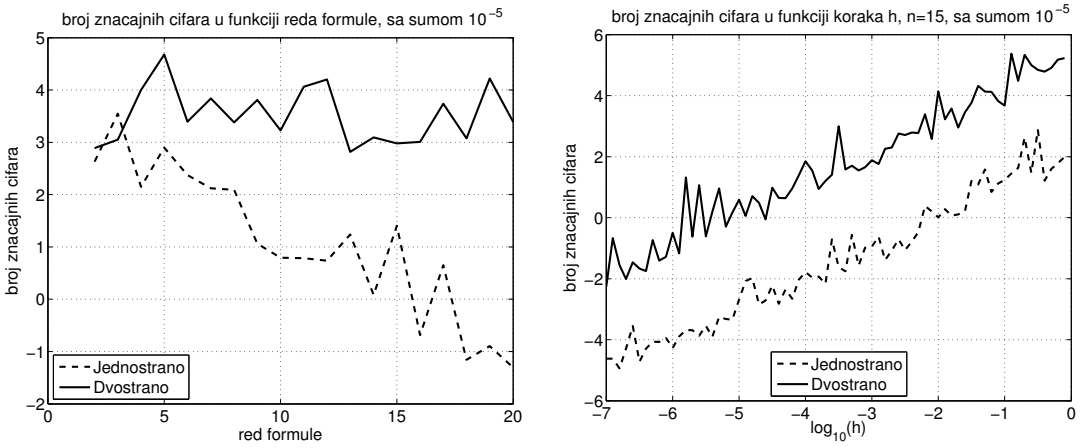
Implementirane funkcije, za tačnost ulaznih podataka 10^{-5}, mogu se pozvati na sledeći način

```
>>izvodLevi(10,.01,pi/4,@(x) randSin(x,10^(-5)))
ans =
         0.696640453370979
>>izvodDvostrano(10,.01,pi/4,@(x) randSin(x,10^(-5)))
ans =
         0.705650101906824
```

Kao što vidimo rezultati su prilično pogrešni.

Slika 6.2, levo, daje zavisnost broja značajnih cifara u funkciji reda formule u slučaju metoda za jednostrano i dvostrano diferenciranje. Kao što se vidi sa slike, sa povećanjem reda formule dolazi do pada broja značajnih cifara. Zavisnost pada broja značajnih cifara je skoro linearna kod jednostranog diferenciranja. Podaci kojima je zadata funkcija imaju gornju granicu apsolutne greške 10^{-5}. Kao što vidimo, za razliku od metoda jednostranog diferenciranja, kod metoda dvostranog diferenciranja ne dolazi do pada broja značajnih cifara sa porastom reda formule. Ova osobina formula za dvostrano diferenciranje čini bitnu razliku. Kažemo da su formule za dvostrano diferenciranje numerički stabilnije.

Slika 6.2, desno, prikazuje zavisnost broja značajnih cifara izvoda u funkciji koraka h. Slika ilustruje određivanje izvoda funkcije sin u tački $\pi/4$, sa formulom reda 15, pri čemu ulazni podaci imaju gornju granicu apsolutne greške 10^{-5}. Vidimo da je prikazani grafik samo translacija grafika 6.1, levo, za vrednost 10 koliko cifara smo izgubili u ulaznim podacima. Primetimo da nam nedostaje deo rasta broja značajnih cifara za velike vrednosti koraka h sa slike 6.1, levo. Red formule 15 je dovoljan za postizanje tačnosti 10^{-5} za velike vrednosti h, koliko formula maksimalno



Slika 6.2: Slika levo ilustruje broj značajnih cifara u funkciji reda formule kod jednostranog i dvostranog diferenciranja funkcije sin u tački $\pi/4$, slika desno prikazuje broj značajnih cifara u funkciji koraka h kod jednostranog i dvostranog diferenciranja iste funkcije u istoj tački. Podaci imaju gornju granicu apsolutne greške 10^{-5}

i postiže, tako da dalje smanjenje koraka ne može da dovede do povećanja tačnosti izvoda, jer nastupaju efekti dodatog šuma.

Naravno, superiornost metoda dvostranog diferenciranja je evidentna.

6.1.5 Izvodi višeg reda

Za određivanje izvoda višeg reda možemo se poslužiti sledećim metodom. Kako je

$$D = \frac{1}{h}\log(1+\Delta),$$

jednostavno nalazimo da važi

$$D^k = \frac{1}{h^k}(\log(1+\Delta))^k, \quad k \in \mathbb{N}.$$

Da bismo mogli da generišemo formule za numeričko diferenciranje neophodno nam je poznavanje razvoja u red funkcije $(\log(1+x))^k$, $k \in \mathbb{N}$, za malu vrednost argumenta x.

Lema 6.3 *Za svako $k \in \mathbb{N}$ i $|x| < 1$, važi*

$$\frac{(\log(1+x))^k}{k!} = \sum_{\ell=0}^{+\infty} \frac{S_{k+\ell}^k}{(k+\ell)!} x^{k+\ell},$$

gde su $S_{k+\ell}^k$, $\ell \in \mathbb{N}_0$, $k \in \mathbb{N}$, Stirlingovi brojevi koji zadovoljavaju sledeću rekurentnu relaciju

$$S_{k+\ell+1}^{k+1} = S_{k+\ell}^k - (k+\ell)S_{k+\ell}^{k+1}, \quad \ell \in \mathbb{N}, \quad k \in \mathbb{N},$$

uz početni uslov $S_{1+\ell}^1 = (-1)^{\ell-1}(\ell-1)!$, $\ell \in \mathbb{N}$, i $S_k^k = 1$, $k \in \mathbb{N}$.

Dokaz. Ako diferenciramo jednakost kojom su definisani Stirlingovi brojevi, nalazimo

$$\sum_{\ell=0}^{+\infty} \frac{S_{k+1+\ell}^{k+1}}{(k+\ell)!} x^{k+\ell} = \frac{(\log(1+x))^k}{k!} \frac{1}{1+x} = \sum_{\ell=0}^{+\infty} \frac{S_{k+\ell}^{k}}{(k+\ell)!} x^{k+\ell} \sum_{\ell=0}^{+\infty} (-1)^\ell x^\ell$$
$$= x^k \sum_{\ell=0}^{+\infty} x^\ell \sum_{\nu=0}^{\ell} (-1)^{\ell-\nu} \frac{S_{k+\nu}^{k}}{(k+\nu)!}.$$

Odavde zaključujemo da važe identiteti

$$\frac{S_{k+1+\ell}^{k+1}}{(k+\ell)!} = \sum_{\nu=0}^{\ell} (-1)^{\ell-\nu} \frac{S_{k+\nu}^{k}}{(k+\nu)!}, \quad \frac{S_{k+1+\ell-1}^{k+1}}{(k+\ell-1)!} = \sum_{\nu=0}^{\ell-1} (-1)^{\ell-1-\nu} \frac{S_{k+\nu}^{k}}{(k+\nu)!}.$$

Kombinujući ove dve jednakosti nalazimo

$$\frac{S_{k+1+\ell}^{k+1}}{(k+\ell)!} = \frac{S_{k+\ell}^{k}}{(k+\ell)!} - \sum_{\nu=0}^{\ell-1} (-1)^{\ell-1-\nu} \frac{S_{k+\nu}^{k}}{(k+\nu)!} = \frac{S_{k+\ell}^{k}}{(k+\ell)!} - \frac{S_{k+\ell}^{k+1}}{(k+\ell-1)!}.$$

Ako ovu jednakost pomnožimo sa $(k+\ell)!$ dobijamo tvrđenje leme.

Za $k = 1$, nalazimo

$$\sum_{\ell=0}^{+\infty} \frac{S_{\ell+1}^{1}}{(\ell+1)!} x^{\ell+1} = \frac{(\log(1+x))^1}{1!} = \sum_{\ell=1}^{+\infty} (-1)^{\ell-1} \frac{x^\ell}{\ell},$$

odakle dobijamo prvu grupu početnih uslova. Drugu grupu početnih uslova dobijamo iz činjenice da je koeficijent uz najniži stepen za x^k, u razvoju $(\log(1+x))^k/k!$, upravo $S_k^k/k!$ i koji mora biti jednak 1 jer je proizvod jedinica iz razvoja za $\log(1+x)$. □

Zahvaljujući ovoj lemi u stanju smo da generišemo formule za izračunavanje izvoda višeg reda. Naime, važi sledeća teorema.

Teorema 6.6 *Za svako $k \in \mathbb{N}$, važi*

$$D^k = \frac{1}{h^k} \sum_{\ell=0}^{+\infty} \frac{k!}{(k+\ell)!} S_{k+\ell}^{k} \Delta^{k+\ell},$$

gde je $S_{k+\ell+1}^{k+1} = S_{k+\ell}^{k} - (k+\ell) S_{k+\ell}^{k+1}$, $\ell \in \mathbb{N}$, $k \in \mathbb{N}$, uz početni uslov $S_{1+\ell}^{1} = (-1)^{\ell-1}(\ell-1)!$, $\ell \in \mathbb{N}$, i $S_k^k = 1$, $k \in \mathbb{N}$.

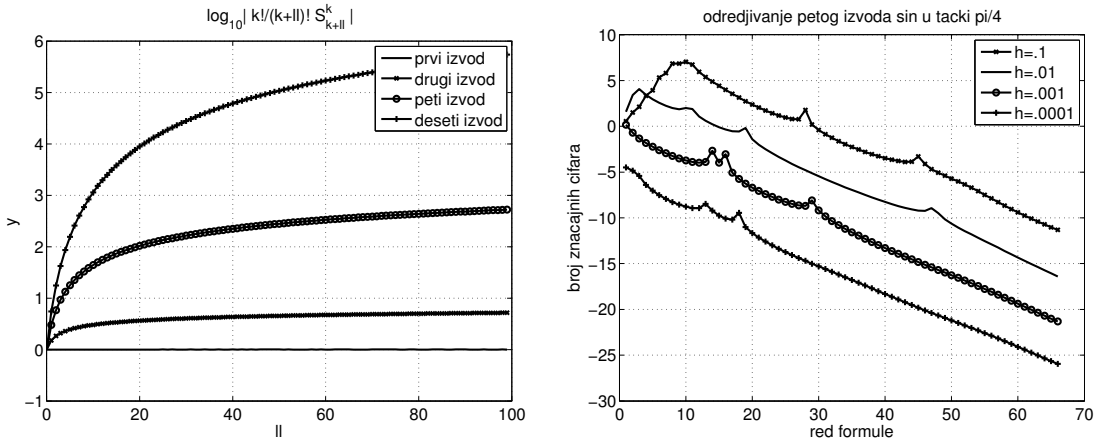
Nažalost, diferencna jednačina koja opisuje Stirlingove brojeve ne može se rešiti analitički. Međutim, moguće je za neke konkretne vrednosti k odrediti neke elemente niza Stirlingovih brojeva. Posebno, može se vrlo jednostavno napisati funkcija koja generiše niz Stirlingovih brojeva. Na primer, u Matlabu se funkcija može implementirati na sledeći način:

```
function [ res ] = stirling( mu, j )
%STIRLING(MU,J) vraca matricu ispunjenu
% Stirlingovim brojevima [a(n,m)], gde je a(n,m)=S_{m+n}^m
% m=1,...,mu, n=1,...,j
```

```
res=zeros(j,mu);
res(1,:)=1;
for ell=2:j;
    res(ell,1)=-(ell-1)*res(ell-1,1);
end
for k=2:mu
    for ell=2:j
        res(ell,k)=res(ell,k-1)-(k+ell-2)*res(ell-1,k);
    end
end
end
```

Implementacija ove funkcije omogućava konstrukciju slike 6.3, levo. Slika ilustruje grafik funkcije $\log_{10}|k!/(k+\ell)!S^k_{k+\ell}|$, kao funkcije od ℓ, a za različite vrednosti k shvaćenje kao parametar. Kao što vidimo kako k raste funkcije $|k!/(k+\ell)!S^k_{k+\ell}|$ imaju sve brži rast, drugim rečima, ako hoćemo da računamo izvod visokog reda to će sve veći uticaj imati konačne razlike visokog reda. Međutim, kao što znamo, konačne razlike visokog reda su opterećene greškama koje unose ulazni podaci i uglavnom su određene sa vrlo malo ili nijednom značajnom cifrom. Ova pojava dovodi do ograničene mogućnosti upotrebe formula za određivanje izvoda visokog reda.

Slika 6.3, desno, ilustruje opisanu pojavu. Prikazan je broj značajnih cifara u petom izvodu funkcije sin u tački $\pi/4$ u zavisnosti od broja tačaka koje se upotrebljavaju pri izračunavanju. Interesantno je da se najbolji rezultati dobijaju sa najvećom vrednošću koraka $h = .1$, dok se



Slika 6.3: Slika levo ilustruje grafik $\log_{10}|k!S^\ell_{k+\ell}/(k+\ell)!|$ u funkciji od ℓ za $k = 1, 2, 5, 10$, dok slika desno prikazuje zavisnost broja značajnih cifara u funkciji reda formule pri određivanju petog izvoda za vrednosti koraka $h = .1, .01, .001, .0001$.

najgori rezultati dobijaju sa najmanjom vrednošću koraka. Takođe, vidimo da sa povećanjem reda formule dolazi do povećanja broja značajnih cifara u rezultatu, ali samo do izvesnog reda. Dalje povećanje reda formule dovodi samo do gubitka značajnih cifara. Ova pojava je posledica ponašanja funkcije $5!S^5_{5+\ell}/(5+\ell)!$. Naime, kao što vidimo na grafiku, levo, ova funkcija je rastuća što dovodi do povećanja uticaja konačnih razlika visokog reda koje se u aritmetici mantise konačne dužine ne mogu odrediti dovoljno tačno. Grafik koji dobijamo je posledica istog fenomena kao i

pojava posmatrana u sekciji 1.3.3.

Za generalnu vrednost k može se dati izraz koju uključuje nekoliko prvih članova

$$D^k \approx \frac{1}{h^k}\left(\Delta^k - \frac{k}{2}\Delta^{k+1} + \frac{k(3k+5)}{24}\Delta^{k+2} - \frac{k(k+2)(k+3)}{48}\Delta^{k+3}\right).$$

Implementacija metoda u `Matlabu` može izgledati ovako

```
function [ res ] = izvodJednostranoK( h, fun, k )
%IZVODJEDNOSTRANOK(H,FUN,K) odredjuje K-ti izvod funkcije
% koja je zadata vrednostima FUN sa korakom H

n = length(fun); ss = stirling(k,n);
s = ss(:,end); tabela = diff(fun);
dif = [];
while not(isempty(tabela))
    dif = [dif tabela(1)];
    tabela = diff(tabela);
end
res = 0; fak = 1;
for i=k:(n-1)
    res = res+s(i-k+1)/fak*dif(i);
    fak = fak*(i+1);
end
res = res/h^k;
end
```

Sličnu teoremu je moguće formulisati i za izvode višeg reda koji koriste operator ∇. Kako je

$$D = -\frac{1}{h}\log(1-\nabla), \quad D^k = \frac{(-1)^k}{h^k}\left(\log(1-\nabla)\right)^k, \quad k \in \mathbb{N}_0,$$

na sličan način, kao u slučaju operatora prednje razlike možemo formulisati sledeću lemu.

Lema 6.4 *Za svako $k \in \mathbb{N}$ i $|x| < 1$, važi*

$$\frac{(\log(1-x))^k}{k!} = \sum_{\ell=0}^{+\infty}(-1)^{k+\ell}\frac{S_{k+\ell}^k}{(k+\ell)!}x^{k+\ell},$$

gde su $S_{k+\ell}^k$, $\ell \in \mathbb{N}_0$, $k \in \mathbb{N}$, Stirlingovi brojevi.

Dokaz. Dovoljno je samo u već postojeći razvoj funkcije $(\log(1+x))^k/k!$ uvesti smenu $x := -x$.□

Koristeći ovu lemu možemo da formulišemo teoremu o izvodu reda k koja koristi operator ∇.

Teorema 6.7 *Za svako $k \in \mathbb{N}$, važi*

$$D^k = \frac{(-1)^k}{h^k}\sum_{\ell=0}^{+\infty}(-1)^{k+\ell}\frac{k!}{(k+\ell)!}S_{k+\ell}^k\nabla^{k+\ell},$$

gde su $S_{k+\ell}^k$, $\ell \in \mathbb{N}_0$, $k \in \mathbb{N}$, Stirlingovi brojevi.

Funkcija koja implementira izračunavanje izvoda koristeći operator ∇ može se dati ovako

```
function [ res ] = izvodJednostranoNablaK( h, fun, k )
%IZVODJEDNOSTRANONABLAK(H,FUN,K) odredjuje K-ti izvod funkcije
% koja je zadata vrednostima FUN sa korakom H
% pri cemu koristi operator zadnje razlike

n = length(fun); ss = stirling(k,n);
s = ss(:,end); tabela = diff(fun);
dif = [];
while not(isempty(tabela))
    dif = [dif tabela(end)];
    tabela = diff(tabela);
end
res = 0; fak = 1;
for i=k:(n-1)
    res = res+s(i-k+1)/fak*dif(i);
    fak = -fak*(i+1);
end
res = res/h^k;
end
```

Numeričke osobine ovog načina određivanja izvoda su identične načinu određivanja izvoda korišćenjem operatora prednje razlike.

Formule za određivanje viših izvoda koje koriste centralne razlike mogu se konstruisati na sličan način. Osnovni elementi za konstrukciju su nam formule

$$D = \frac{2}{h}\log\left(\frac{1}{2}\delta + \left(1+\frac{1}{4}\delta^2\right)^{1/2}\right) = \frac{1}{h}\sum_{k=0}^{+\infty}(-1)^k\frac{\prod_{\ell=1}^{k}(2\ell-1)^2}{2^{2k}(2k+1)!}\delta^{2k+1}, \tag{6.4}$$

i formula

$$D = \frac{2\mu}{h}\left(1+\frac{1}{4}\delta^2\right)^{-1/2}\log\left(\frac{1}{2}\delta + \left(1+\frac{1}{4}\delta^2\right)^{1/2}\right) = \frac{\mu}{h}\sum_{k=0}^{+\infty}(-1)^k\frac{\prod_{\ell=1}^{k}\ell^2}{(2k+1)!}\delta^{2k+1}. \tag{6.5}$$

Pretpostavimo da treba odrediti n-ti izvod dvostranim metodom i da n ima binarni zapis u obliku $n = (a_\ell a_{\ell-1}\cdots a_1 a_0)_2$. Onda možemo pisati

$$D^n = D^{a_\ell 2^\ell + \cdots + a_0} = D^{a_\ell 2^\ell} D^{a_{\ell-1}2^{\ell-1}} \cdots D^{a_1 2^1} D^{a_0 2^0}.$$

Ovde su a_i, $i = 0, \ldots, \ell$, cifre koje pripadaju binarnom brojnom sistemu, imaju moguće vrednosti iz skupa nula ili jedan. Red kojim je reprezentovan operator D^n, možemo dobiti proizvodom redova reprezentacija operatora $D^{a_i 2^i}$, $i = 0, \ldots, \ell$. Međutim, potrebno je da na kraju red koji dobijemo ima zavisnost od operatora centralne razlike takav da sve tačke pripadaju podeli istovremeno sa tačkom u kojoj se određuje izvod. Drugim rečima, treba nam formula koja se ne ponaša kao formula (6.4), već kao formula (6.5).

Prvo primetimo da važi $D^{2^{i+1}} = D^{2^i} D^{2^i}$, $i \in \mathbb{N}_0$. Ovo zapažanje nam omogućava da efikasno određujemo razvoj u red operatora $D^{2^{i+1}}$ ako znamo razvoj u red operatora D^{2^i}. Može se pokazati

da ako pomnožimo razvoje operatora D i D zadate formulom (6.4), dobijamo formulu za drugi izvod, operator D^{2^1}, kod koje i tačka u kojoj se određuje izvod i tačke koje učestvuju u formuli pripadaju podeli. I više, može se pokazati da sve formule D^{2^i}, $i \in \mathbb{N}$, dobijene na ovaj način imaju ovo svojstvo. To znači da je dovoljno formule za operatore D^{2^i}, $i \in \mathbb{N}$, dobijati množenjem formule (6.4), dok je samo za operator D^{2^0} potrebno koristiti formulu (6.5).

Nemoguće je dobiti analitičke izraze za koeficijente u formulama. Međutim, moguće je napisati program koji generiše koeficijente u formulama

$$D^{2n} = \frac{1}{h^{2n}} \sum_{k=0}^{+\infty} A_k^{2n} \delta^{2k+2n} = \frac{1}{h^{2n}} \sum_{k=0}^{+\infty} A_k^{2n} \Delta^{2k+2n} E^{-k-n},$$

$$D^{2n+1} = \frac{\mu}{h^{2n+1}} \sum_{k=0}^{+\infty} A_k^{2n+1} \delta^{2k+2n+1} = \frac{1}{2h^{2n+1}} \sum_{k=0}^{+\infty} A_k^{2n+1} \Delta^{2k+2n+1} (1 + E^{-1}) E^{-k-n},$$

gde smo koristili lemu 4.7. Da bismo program izrazili preglednije moramo napisati četiri funkcije. Sve četiri funkcije treba staviti u jedan M fajl, sa imenom `dvostranoKoeficijenti.m`. Izgled fajla je sledeći:

```
function [ res ] = dvostranoKoeficijenti( n,brojClanova )
%DVOSTRANOKOEFICIJENTI(N,BROJCLANOVA) konstruise formulu za
% dvostrano numericko izracunavanje izvoda reda N
% sa BROJCLANOVA clanova, izlazni parametar RES sadrzi vektor
% koeficijenata formule

nBin=dec2bin(n); res=zeros(1,brojClanova); res(1)=1;
mP=koeficijentiMedjupodela(brojClanova);
for k=(length(nBin)-1):-1:1
    mP=cauchyProizvod(mP,mP);
    if nBin(k)=='1'
        res=cauchyProizvod(res,mP);
    end
end
if nBin(end)=='1'
    mP=koeficijentiIzvod(brojClanova);
    res=cauchyProizvod(res,mP);
end
end

function [ res ] = koeficijentiIzvod(brojClanova)
%KOEFICIJENTIIZVOD(BROJCLANOVA) vraca koeficijente u formuli za
% dvostrano numericko diferenciranje reda BROJCLANOVA

res=ones(1,brojClanova);
for i=2:brojClanova
    res(i)=-(i-1)/(2*(2*i-1))*res(i-1);
end
end
```

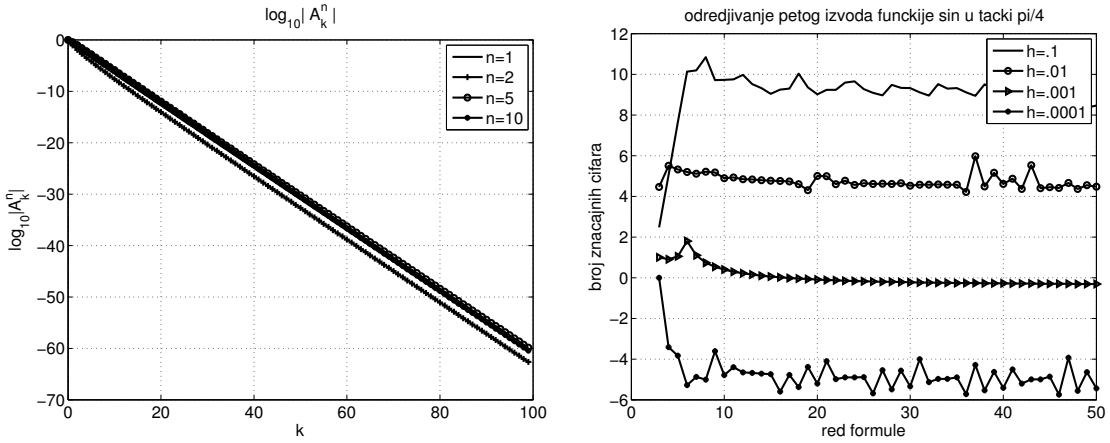
```
function [ res ] = koeficijentiMedjupodela(brojClanova)
%KOEFICIJENTIMEDJUPODELA(BROJCLANOVA) vraca koeficijente u formuli za
% dvostrano numericko diferenciranje sa medjupodelom reda BROJCLANOVA

res=ones(1,brojClanova);
for i=2:brojClanova
    res(i)=-(2*i-3)*(2*i-3)/(8*(i-1)*(2*i-1))*res(i-1);
end
end

function [ res ] = cauchyProizvod(a,b)
%CAUCHYPROIZVOD(A,B) izracunava Cauchyev proizvod vektora A i B
% vektori A i B moraju biti iste duzine

res=zeros(1,length(a));
for i=1:length(a)
    for j=1:i
        res(i)=res(i)+a(j)*b(i+1-j);
    end
end
end
```

Funkcija `cauchyProizvod` izračunava Cauchyev proizvod dva vektora, u suštini realizuje proizvod vektora shvaćenih kao koeficijente polinoma, funkcija `koeficijentiMedjupodela` prosto konstruiše koeficijente formule (6.4), funkcija `koeficijentiIzvod` vraća koeficijente formule (6.5), dok funkcija `dvostranoKoeficijenti` konstruiše koeficijente u formulama za numeričko izračunavanje izvoda D^n, gde je n prvi argument u pozivu funkcije, dok je drugi argument zahtevani red formule.

Na primer, konstruišimo formulu za izračunavanje drugog i petog izvoda reda osam. Nama su u interesu egzaktne vrednosti koeficijenata, tako da ćemo koristiti format racionalnih brojeva. Dobijamo

```
>>format rat
>>dvostranoKoeficijenti(2,8)
ans =
  Columns 1 through 4
       1            -1/12           1/90          -1/560
  Columns 5 through 8
       1/3150       -1/16632        1/84084       -1/411840
>>dvostranoKoeficijenti(5,8)
ans =
  Columns 1 through 4
       1            -1/3           13/144       -139/6048
  Columns 5 through 8
      37/6480      -27/19241       34/99087       -9/107504
```

Koeficijenti za određivanje petog izvoda nisu u potpunosti tačni, jer `Matlab` pravi grešku pri konverziji iz formata dvostruke preciznosti u racionalne brojeve. Ovo se odnosi na elemente sa indeksima 7, 8 i 9. Međutim, greška je malog reda veličine.



Slika 6.4: Slika levo prikazuje grafik funkcije $\log_{10}|A_k^n|$ u funkciji od k za $n = 1, 2, 5, 10$, slika desno ilustruje zavisnost broja značajnih cifara u funkciji reda formule za vrednosti koraka $h = .1, .01, .001, .0001$, pri određivanju petog izvoda funkcije sin u tački $\pi/4$.

Ono što možemo uočiti na osnovu ovih podataka je da, za razliku od koeficijenata u metodu jednostranog diferenciranja, koeficijenti opadaju sa porastom indeksa. Slika 6.4, levo, prikazuje zavisnost koeficijenata A_k^n, $k \in \mathbb{N}_0$, $n \in \mathbb{N}$, u formulama za dvostrano numeričko diferenciranje. Vidimo da koeficijenti opadaju eksponencijalno sa porastom indeksa k. Ovo je osobina stabilnosti formula za dvostrano diferenciranje koju formule za jednostrano diferenciranje nemaju. Ovo je glavni razlog za konstantan broj značajnih cifara sa porastom reda formule, što vidimo na slici 6.4, desno. Ovakvo ponašanje ne postoji kod formula za jednostrano diferenciranje, kao što je ilustrovano na slici 6.3, desno. Kod formula za jednostrano diferenciranje sa porastom reda formule dolazi do opadanja broja značajnih cifara, jer koeficijenti formule rastu.

Međutim, kao i kod formula za jednostrano diferenciranje sa smanjivanjem koraka h dolazi do pada broja značajnih cifara. Ovo je posledica oduzimanja bliskih brojeva i prostiranja grešaka u tabeli konačnih razlika.

Funkcija koja implementira nalaženje k-tog izvoda metodom dvostranog diferenciranja može se dati na sledeći način

```
function [ res ] = izvodDvostranoK( h, fun, k )
%IZVODDVOSTRANOK(H,FUN,K) odredjuje K-ti izvod funkcije
% koja je zadata vrednostima FUN sa korakom H

tabela = fun; tabela = diff(tabela,k);
koef = dvostranoKoeficijenti(k,length(tabela));
res = 0;
if mod(k,2) == 0
    i = 1; n = (length(tabela)+1)/2;
    while not(isempty(tabela))
        res = res+koef(i)*tabela(n);
        tabela = diff(tabela,2);
        i = i+1;
```

```
        n = n-1;
    end
else
    i = 1; n = length(tabela)/2;
    while not(isempty(tabela))
        res = res+koef(i)*(tabela(n)+tabela(n+1));
        tabela = diff(tabela,2);
        i = i+1;
        n = n-1;
    end
    res = res/2;
end
res = res/h^k;
end
```

Na kraju navedimo najjednostavniju formulu za određivanje drugog izvoda koja se često koristi u praksi

$$D^2 f(x_i) = \frac{1}{h^2}(f_{i+1} - 2f_i + f_{i-1}) - \frac{h^2}{12} f^{(4)}(\psi),$$

koja važi pod uslovom da je funkcija f četiri puta diferencijabilna na intervalu $[x_{i-1}, x_{i+1}]$, pri čemu je $\psi \in (x_{i-1}, x_{i+1})$.

Formule za dvostrano numeričko diferenciranje u razvijenom obliku mogu se dobiti koristeći sledeću lemu.

Lema 6.5 *Za svako $n \in \mathbb{N}_0$, važi*

$$\delta^{2n} f(x) = \sum_{k=0}^{2n} (-1)^k \binom{2n}{k} f(x + (n-k)h),$$

$$\mu\delta^{2n+1} f(x) = \frac{1}{2(n+1)} \sum_{k=-1}^{2n+1} (-1)^{k+1}(n-k) \binom{2n+2}{k+1} f(x + (n-k)h).$$

Dokaz. Dovoljno je primeniti binomnu formulu na sledeće identitete $\delta^{2n} = (E^{1/2} - E^{-1/2})^{2n}$ i $\mu\delta^{2n+1} = 1/2(E^{1/2} + E^{-1/2})(E^{1/2} - E^{-1/2})^{2n}$. □

6.2 Numerička integracija

U ovom poglavlju bavimo se metodima za približno izračunavanje određenih intergala

$$\int_a^b w(x) f(x) dx,$$

gde je w težinska funkcija, a f je podintegralna funkcija. Interval na kome integralimo $[a, b]$ može biti ograničen ili neograničen. U slučaju neograničenog intervala preciznije je govoriti o intervalu $[a, +\infty)$, $(-\infty, b]$ ili $(-\infty, +\infty) = \mathbb{R}$. Međutim, da bismo zadržali uniformnost notacije svesno usvajamo nepreciznost, i govorimo i u ovom slučaju o intervalu $[a, b]$. Ako drugačije nije naglašeno podrazumevaćemo da je funkcija f neprekidna, čak u većini slučajeva podrazumevamo da je funkcija f i dovoljan broj puta neprekidno diferencijabilna.

Formule za približno izračunavanje integrala uglavnom su bazirane na formuli za definiciju Riemannovog integrala, i imaju sledeći oblik

$$\int_a^b w(x)f(x)dx = \sum_{\ell=1}^{n} A_\ell f(x_\ell) + R_n(f;w),$$

gde su koeficijenti A_ℓ, $\ell = 1,\dots,n$, težinski koeficijenti ili težine, a x_ℓ, $\ell = 1,\dots,n$, su čvorovi kvadraturne formule. Podrazumevamo da su čvorovi realni, različiti i da su poređani u rastući poredak $x_1 < x_2 < \cdots < x_n$.

Iz kvadraturne formule možemo sagledati pojam težinske funkcije. Kao što vidimo težinska funkcija ne učestvuje u izrazu desno, tako da se kvadraturna formula konstruiše u odnosu na interval integracije $[a,b]$ i težinsku funkciju w. Promenom intervala integracije ili težinske funkcije menjamo težine i (ili) čvorove kvadraturne formule. Međutim, čvorovi i težine kvadraturne formule ne zavise od funkcije f, pa je za funkciju f obezbeđeno ime podintegralna funkcija.

Veličinu $R_n(f;w)$ nazivamo ostatkom kvadraturne formule. Ova veličina meri grešku koja nastaje prilikom aproksimacije integrala konačnom sumom na desnoj strani jednakosti. Ako se težinska funkcija podrazumeva prosto pišemo $R_n(f)$, ako se i podintegralna funkcija podrazumeva pišemo R_n za grešku kvadraturne formule.

Kako podintegralna funkcija f ne mora biti definisana van intervala $[a,b]$, nema puno smisla za čvorove kvadraturne formule birati tačke koje se nalaze van intervala $[a,b]$. S obzirom na ovu činjenicu podrazumevamo da su čvorovi kvadraturne formule unutar intervala $[a,b]$.

Sve formule delimo na formule otvorenog i zatvorenog tipa. Kvadraturna formula je zatvorenog tipa ako je $x_1 = a$, $x_n = b$, u slučaju da je interval ograničen. Ako je interval neograničen sa desne strane, a ograničen sa leve, onda je formula zatvorenog tipa u slučaju $x_1 = a$. Slično je za ostale kombinacije ograničenosti i neograničenosti intervala integracije. Formula je otvorena ako nije zatvorena.

Kad smo fiksirali čvorove postavlja se pitanje određivanja težina kvadraturne formule A_k, $k = 1,\dots,n$. Postoji veliki broj načina za određivanje težina u kvadraturnoj formuli. Međutim, nas zanimaju samo kvadraturne formule sa algebarskim stepenom tačnosti. Kod ovih formula težine se određuju iz uslova da je formula tačna na skupu polinoma što je moguće višeg stepena.

Na primer, posmatrajmo formulu

$$\int_{-1}^{1} f(x)dx = f(-.5) + f(.5) + R_2(f).$$

Jednostavno proveravamo da važi

$$2 = \int_{-1}^{1} 1\,dx = 1 + 1 = 2, \quad 0 = \int_{-1}^{1} x dx = -.5 + .5 = 0.$$

Odavde zaključujemo da je $R_2(1) = 0$ i $R_2(x) = 0$, pa je formula tačna na skupu polinoma ne višeg stepena od jedan. Ovaj skup polinoma označavamo sa $\mathcal{P}_1$. Međutim, za polinom x^2 dobijamo

$$\frac{2}{3} = \int_{-1}^{1} x^2 dx \neq (-.5)^2 + (.5)^2 = .25 + .25 = \frac{1}{2},$$

tako da formula nije tačna na skupu polinoma ne višeg stepena od dva, u oznaci $\mathcal{P}_2$.

Definicija 6.1 *Kvdaraturna formula*

$$\int_a^b w(x)f(x)dx = \sum_{k=1}^{n} A_k f(x_k) + R_n(f;w),$$

ima algebarski stepen tačnosti najmanje $m \in \mathbb{N}_0$ ako i samo ako za svako $p \in \mathcal{P}_m$ važi $R_n(p;w) = 0$.

Jednostavno možemo reći da težine A_k, $k = 1, \ldots, n$, kvadraturne formule određujemo iz uslova da kvadraturna formula ima maksimalni algebarski stepen tačnosti. Iz ovog uslova dobijamo sistem linearnih jednačina koji težine moraju da zadovolje

$$\mu_k = \int_a^b w(x)x^k dx = \sum_{\ell=1}^{n} A_\ell x_\ell^k, \quad k = 0, \ldots, n-1.$$

Primetimo da je matrica sistema Vandermondova matrica konstruisana za čvorove kvadraturne formule. Kako su čvorovi kvadraturne formule različiti jednostavno zaključujemo da sistem linearnih jednačina ima jedinstveno rešenje. I više, možemo zaključiti da za svaku kvadraturnu formulu sa n čvorova možemo izabrati težine A_k, $k = 1, \ldots, n$, tako da formula ima algebarski stepena tačnosti barem $n-1$. Veličine μ_k, $k \in \mathbb{N}_0$, obično nazivamo momentima.

Teorema 6.8 *Neka su $x_k \in [a,b]$, $k = 1, \ldots, n$, $x_1 < x_2 < \cdots < x_n$, čvorovi kvadraturne formule*

$$\int_a^b w(x)f(x)dx = \sum_{\ell=1}^{n} A_\ell f(x_\ell) + R_n(f;w),$$

onda kvadraturna formula ima algebarski stepen tačnosti barem $n-1$ ako su težine izabrane kao rešenja sistema linearnih jednačina

$$\mu_k = \int_a^b w(x)x^k dx = \sum_{\ell=1}^{n} A_\ell x_\ell^k, \quad k = 0, \ldots, n-1.$$

Na primer, konstruišimo kvadraturnu formulu

$$\int_0^1 \frac{1}{1+x^2} f(x)dx = A_1 f(.25) + A_2 f(.5) + A_3 f(.75) + R_3(f;w),$$

koja ima algebarski stepen tačnosti barem dva. Prema tvrđenju prethodne teoreme to je moguće. Potrebno je samo rešiti sistem linearnih jednačina

$$\begin{aligned}
\mu_0 &= \frac{\pi}{4} = \int_0^1 \frac{x^0}{1+x^2}dx = A_1 + A_2 + A_3,\\
\mu_1 &= \frac{1}{2}\log 2 = \int_0^1 \frac{x}{1+x^2}dx = \frac{1}{4}A_1 + \frac{1}{2}A_2 + \frac{3}{4}A_3,\\
\mu_2 &= 1 - \frac{\pi}{4} = \int_0^1 \frac{x^2}{1+x^2}dx = \frac{1}{16}A_1 + \frac{1}{4}A_2 + \frac{9}{16}A_3.
\end{aligned}$$

Koristeći `Matlab`, prethodni sistem linearnih jednačina možemo relativno jednostavno rešiti koristeći sledeću sekvencu naredbi

```
>>V=vander([1/4 1/2 3/4])';
>>mu=[1-pi/4 log(2)/2 pi/4]';
>>A=linsolve(V,mu)
A =
   0.607273280213032
  -0.244646431353610
   0.422771314538026
```

Sa ovako odabranim težinama smo dobili kvadraturnu formulu koja ima algebarski stepen tačnosti barem dva.

Naravno, ovakav način konstrukcije kvadraturne formule ima nedostataka. Najvažniji je taj što prilikom konstrukcije koristimo rešavanje sistema linearnih jednačina kod koga je matrica sistema Vandermondova matrica. Kako je objašnjeno u delu o interpolaciji, u opštem slučaju, faktor uslovljenosti Vandermondove matrice raste eksponencijalno sa redom matrice kao što je ilustrovano na slici 4.3, desno. Veliki faktor uslovljenosti Vandermondove matrice dovodi do ozbiljnog gubitka značajnih cifara sa povećanjem broja čvorova kvadraturne formule ili sa povećanjem stepena algebarske tačnosti. U sledećoj sekciji bavimo se preciznijim metodom za konstrukciju kvadraturne formule.

6.2.1 Interpolacione formule

Kao što je poznato interpolacioni polinom stepena ne većeg od n, polinoma stepena ne većeg od n, je upravo polazni polinom (videti lemu 4.2). Ova činjenica nam omogućava formulaciju sledeće teoreme.

Teorema 6.9 *Neka su $x_\ell \in [a,b]$, $\ell = 0,\dots,n$, $x_0 < x_1 < \cdots < x_n$, čvorovi kvadraturne formule*

$$\int_{-1}^{1} w(x)f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + R_{n+1}(f;w).$$

Formula ima algebarski stepen tačnosti barem n ako su težine izabrane na sledeći način

$$A_\ell = \int_a^b w(x)L_\ell(x)dx, \quad \ell = 0,\dots,n, \tag{6.6}$$

gde je L_ℓ, $\ell = 0,\dots,n$, Lagrangeov bazis.

Dokaz. Neka je p polinom ne većeg stepena od n. Lagrangeov interpolacioni polinom P_n, konstruisan u interpolacionim čvorovima x_i, $i = 0,\dots,n$, polinoma p identičan je polinomu p. Važi

$$p(x) = P_n(x) = \sum_{\ell=0}^{n} p(x_\ell)L_\ell(x), \quad p \in \mathcal{P}_n.$$

Ako ovu jednakost pomnožimo sa w i integralimo od a do b, nalazimo

$$\int_a^b w(x)p(x)dx = \int_a^b w(x)\left(\sum_{\ell=0}^{n} p(x_\ell)L_\ell(x)\right)dx = \sum_{\ell=0}^{n} p(x_\ell)\int_a^b w(x)L_\ell(x)dx.$$

Iz ove jednakosti zaključujemo da ako izaberemo

$$A_\ell = \int_a^b w(x)L_\ell(x)dx, \quad \ell = 0, \ldots, n,$$

formula postaje tačna za sve polinome $p \in \mathcal{P}_n$. □

Ova teorema omogućava da problem određivanja težina u kvadraturnoj formuli svedemo na problem izračunavanja integrala. Na ovaj način izbegavamo rešavanje sistema linearnih jednačina sa slabo uslovljenim matricama, ali nailazimo na novi problem. To je problem određivanja integrala sa podintegralnom funkcijom datom u obliku polinoma. Nažalost, ovaj problem nije trivijalan, i takođe je teško rešiv.

Ako izaberemo proizvoljnu funkciju umesto polinoma, interpolaciona formula postaje

$$f(x) = P_n(x) + \frac{f^{(n+1)}(\psi)}{(n+1)!}\omega(x),$$

odakle možemo zaključiti da se ostatak u kvadraturnoj formuli može oceniti na sledeći način

$$R_{n+1}(f; w) = \frac{1}{(n+1)!}\int_a^b w(x)f^{(n+1)}(\psi)\omega(x)dx.$$

Nažalost ova formula je prilično neupotrebljiva, jer je skoro nemoguće odrediti zavisnost ψ od x. Videćemo u narednim odeljcima da postoje druge mogućnosti za određivanje ostatka u kvadraturnim formulama.

6.2.2 Newton-Cotesove kvadraturne formule

Newton-Cotesove kvadraturne formule predstavljaju zatvorene interpolacione kvadraturne formule za težinsku funkciju $w = 1$, na ograničenom intervalu $[a, b]$, gde su čvorovi formule postavljeni u ekvidistantnim tačkama.

Iz postavljenih uslova možemo odrediti čvorove formule na sledeći način $x_\ell = x_0 + \ell h$, $\ell = 0, \ldots, n$, gde je $h = (b-a)/n$. Očigledno je $x_0 = a$ i $x_n = b$, što odgovara činjenici da je formula zatvorena.

Formule za težine (6.6), u ovom specijalnom slučaju, uz smenu $x = x_0 + ph$, možemo transformisati na sledeći način

$$\begin{aligned} A_\ell &= \int_a^b L_\ell(x)dx = \int_a^b \frac{(x-x_0)\cdots(x-x_{\ell-1})(x-x_{\ell+1})\cdots(x-x_n)}{(x_\ell-x_0)\cdots(x_\ell-x_{\ell-1})(x_\ell-x_{\ell+1})\cdots(x_\ell-x_n)}dx \\ &= \int_0^n \frac{(p-0)h\cdots(p-(\ell-1))h(p-(\ell+1))h\cdots(p-n)h}{(\ell-0)h\cdots(\ell-(\ell-1))h(\ell-(\ell+1))h(\ell-n)h}hdp = \frac{(-1)^{n-\ell}h}{\ell!(n-\ell)!}\int_0^n \frac{p^{(n+1)}}{p-\ell}dp, \end{aligned}$$

gde smo uveli oznaku $p^{(s)} = p(p-1)\cdots(p-s+1)$. Obično se izraz za težine izražava preko Newton-Cotesovih brojeva

$$H_\ell = \frac{(-1)^{n-\ell}}{n!n}\binom{n}{\ell}\int_0^n \frac{p^{(n+1)}}{p-\ell}dp, \quad \ell = 0, \ldots, n. \tag{6.7}$$

Formula za težine onda postaje

$$A_\ell = (b-a)H_\ell, \quad \ell = 0, \ldots, n.$$

Kao što vidimo Newton-Cotesovi brojevi su težine za interpolacione formule na intervalu dužine jedan.

Najvažnije osobine Newton-Cotesovih brojeva mogu se iskazati sledećom teoremom.

Teorema 6.10 *Newton-Cotesovi brojevi zadovoljavaju sledeće jednakosti*

$$H_{n-k} = H_k, \quad k = 0, \ldots, n, \quad \sum_{k=0}^{n} H_k = 1.$$

Dokaz. Ako u jednakost (6.7) uvedemo smenu $p := n - p$, nalazimo

$$\begin{aligned} H_\ell &= \frac{(-1)^{n-\ell}}{n!n} \binom{n}{\ell} \int_0^n \frac{(n-p)^{(n+1)}}{n-p-\ell} dp \\ &= -\frac{(-1)^{n-\ell}}{n!n} \binom{n}{\ell} \int_0^n \frac{(n-p)(n-1-p)\cdots(n-p-n-1+1)}{p-(n-\ell)} dp \\ &= \frac{(-1)^{n+2+n-\ell}}{n!n} \binom{n}{n-\ell} \int_0^n \frac{(p-n)(p-(n-1))\cdots p}{p-(n-\ell)} dp \\ &= \frac{(-1)^{n-(n-\ell)}}{n!n} \binom{n}{n-\ell} \int_0^n \frac{p^{(n+1)}}{p-(n-\ell)} dp = H_{n-\ell}. \end{aligned}$$

Primenimo li kvadraturnu formulu na integraciju funkcije $f(x) = 1$, na intervalu $[0, 1]$, nalazimo

$$1 = \int_0^1 1 dx = \sum_{k=0}^{n} H_k,$$

jer su Newton-Cotesovi brojevi težine kvadraturne formule na intervalu dužine jedan. □

Prethodna razmatranja nam omogućavaju formulaciju sledeće teoreme.

Teorema 6.11 (Newton-Cotesove kvadraturne formule) *Newton-Cotesova kvadraturna formula*

$$\int_a^b f(x)dx = (b-a) \sum_{\ell=0}^{n} H_\ell f(x_\ell) + R_{n+1}(f),$$

sa ekvidistantnim čvorovima $x_\ell = x_0 + \ell h$, $\ell = 0, \ldots, n$, $h = (b-a)/n$, $x_0 = a$, *i težinskim koeficijentima* (6.7)*, ima algebarski stepen tačnosti barem* n.

Problem sa Newton-Cotesovim kvadraturnim formulama je njihova konstrukcija. Naime, kao što vidimo da bismo mogli da konstruišemo formulu algebarskog stepena tačnosti n moramo odrediti Newton-Cotesove brojeve H_ℓ, $\ell = 0, \ldots, n$, koji su predstavljeni u obliku određenog integrala. Problem izračunavanja određenog integrala smo sveli na problem izračunavanja određenih integrala kojima su određeni Newton-Cotesovi brojevi.

Da bi ovaj problem bio koliko toliko rešen problem konstrukcije se rešava za male vrednosti n. Konkretno, za male vrednosti n pojedine Newton-Cotesove formule nose imena prema autorima koji su odredili Newton-Cotesove brojeve. Koristićemo skraćeno označavanje $f_k = f(x_k)$, $a = x_0$, $b = x_n$, $h = (b-a)/n$.

Teorema 6.12 (Trapezno pravilo) *Neka je* $f \in C^2[a, b]$*, onda je*

$$\int_a^b f(x)dx = \frac{h}{2}(f_0 + f_1) - \frac{h^3}{12} f''(\psi), \quad \psi \in (a, b).$$

Ovo je najprostija Newton-Cotesova formula koja ima samo dve tačke, $n = 1$, u ovom slučaju je $h = b - a$, $x_0 = a$ i $x_1 = b$. Kao što vidimo, greška formule opada sa trećim stepenom dužine intervala $[a, b]$. Formula se naziva trapeznim pravilom jer se površina ispod grafika funkcije aproksimira površinom trapeza čije osnovne linije leže na pravama $x = a$ i $x = b$, a čija je visina $h = b - a$.

Teorema 6.13 (Simpsonovo pravilo) *Neka je $f \in C^4[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) - \frac{h^5}{90}f^{(4)}(\psi), \quad \psi \in (a, b).$$

Simpsonovo pravilo je Newton-Cotesova formula konstruisana u tri tačke, $n = 2$. Kao što vidimo greška opada sa petim stepenom dužine intervala. Ono što je interesantno kod ove formule je da formula ima algebarski stepen tačnosti 3, iako je $n = 2$. Kako je ostatak dat u obliku četvrtog izvoda vidimo da će biti jednak nuli za sve polinome trećeg stepena. Odavde zaključujemo da je algebarski stepen tačnosti najmanje 3, iako je formula konstruisana za $n = 2$, pa po teoremi 6.11 ima garantovani algebarski stepen tačnosti 2.

Teorema 6.14 (Simpsonovo pravilo 3/8) *Neka je $f \in C^4[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3) - \frac{3h^5}{80}f^{(4)}(\psi), \quad \psi \in (a, b).$$

Kao što vidimo ovde povećanje broja tačaka ne povećava algebarski stepen tačnosti formule. Naime, stepen tačnosti formule ostaje 3.

Teorema 6.15 (Booleovo pravilo) *Neka je $f \in C^6[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) - \frac{8h^7}{945}f^{(6)}(\psi), \quad \psi \in (a, b).$$

U ovom slučaju je algebarski stepen tačnosti 5, što vidimo po tome što je greška u formuli izražena korišćenjem šestog izvoda. I u ovom slučaju je stepen tačnosti za jedan veći u odnosu na procenu koju daje teorema 6.11.

Tabela 6.2 sadrži vrednosti izračunatog određenog integrala

$$\int_0^1 \frac{dx}{1 + x^2} = \frac{\pi}{4},$$

i greške koje dobijamo upotrebom pomenutih kvadraturnih formula. Kao što vidimo greška opada

n	I_n	Δ_n
1	.75	.354(−1)
2	.7833	.210(−2)
3	.7846	.783(−3)
4	.7855	.131(−3)

Tabela 6.2: Vrednost intregrala I_n i apsolutna greška Δ_n dobijene primenom trapezne formule, Simpsonove formule, Simponovog pravila 3/8 i Booleovog pravila.

sa porastom broja tačaka u kvadraturnoj formuli. To se razume jednostavno na osnovu sledeće teoreme.

Teorema 6.16 (Greška Newton-Cotesove formule) *Neka je $n \in \mathbb{N}$ paran broj i neka je $f \in C^{n+2}[a,b]$, onda postoji $\psi \in (a,b)$ tako da važi*

$$\int_a^b f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + \frac{h^{n+3} f^{(n+2)}(\psi)}{(n+2)!} \int_0^n x \prod_{\ell=0}^{n} (x-\ell)dx.$$

Neka je $n \in \mathbb{N}$ neparan broj i neka je $f \in C^{n+1}[a,b]$, onda postoji $\psi \in (a,b)$ tako da važi

$$\int_a^b f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + \frac{h^{n+2} f^{(n+1)}(\psi)}{(n+1)!} \int_0^n \prod_{\ell=0}^{n} (x-\ell)dx.$$

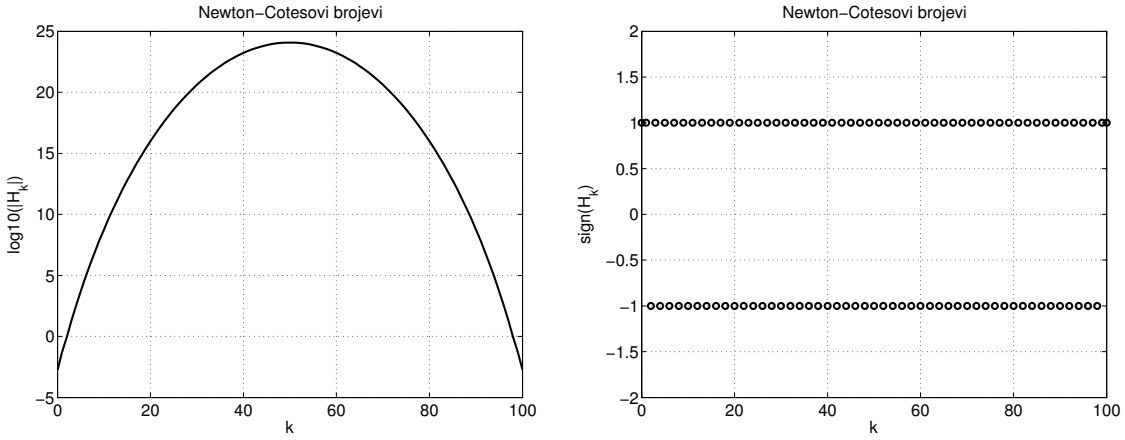
Na osnovu ove teoreme, greška kvadraturne formule u opštem slučaju ima oblik

$$R_{n+1}(f) = C_n h^m f^{(m-1)}(\psi), \quad \psi \in (a,b),$$

gde je $m = 2[n/2] + 3$, i gde smo sa $[x]$ označili celi deo broja x. Odavde zaključujemo da greška u kvadraturnoj formuli opada sa brojem tačaka na sledeći način

$$R_{n+1}(f) \approx \frac{(b-a)^n}{n^n},$$

jer je $h = (b-a)/n$. Kao što vidimo greška je sve manja što je broj tačaka u kvadraturnoj formuli veći. Nažalost, ovaj zaključak, iako teorijski potpuno tačan, nema opravdanja u primeni, jer mi sva izračunavanja izvodimo u aritmetici mantise konačne dužine.



Slika 6.5: Slika levo prikazuje grafik funkcije $\log_{10}(|H_k|)$, $k = 0, \ldots, 100$, dok slika desno prikazuje grafik funkcije $\mathrm{sgn}(H_k)$, $k = 0, \ldots, 100$.

Da bismo mogli da ilustrujemo o čemu se radi posmatraćemo Newton-Cotesovu formulu sa $n = 100$, konstruisanu za izračunavanje integrala

$$\int_0^1 f(x)dx.$$

Nećemo navoditi vrednosti Newton-Cotesovih brojeva, koji su u ovom slučaju jednaki težinama, jer je $b - a = 1 - 0 = 1$, ali ćemo prikazati zavisnost vrednosti Newton-Cotesovih brojeva u funkciji indeksa. Slika 6.5, levo, sadrži grafik funkcije $\log_{10} |H_k|$, dok ista slika, desno, sadrži samo informacije o znaku Newton-Cotesovih brojeva. Sa pomenutih slika vidimo da red veličine Newton-Cotesovih brojeva varira od 10^{25} do otprilike 10^{-3}. Vidimo da Newton-Cotesovi brojevi otprilike alternativno menjaju znak. Ovo je već dovoljna količina informacija da zaključimo da je formula neupotrebljiva u formatu dvostruke preciznosti. Na primer, ako ovako konstruisanu formulu primenimo na izračunavanje integrala

$$\int_0^1 1dx = 1,$$

možemo da očekujemo vrednost reda veličine 10^9, jer je $25 - 16 = 9$, a argumentacija je ista kao u odeljku 1.3.3.

Da bismo bili potpuno precizni, vrednost koja se dobije primenom ove kvadraturne formule na izračunavanje pomenutog integrala iznosi $1.383269227512789 \cdot 10^8$.

Ovakvo ponašanje Newton-Cotesovih brojeva, sa povećanjem stepena algebarske tačnosti, ograničava broj tačaka u formulama u primeni na relativno male vrednosti. Formule koje dopuštaju visok stepen algebarske tačnosti, a ostaju primenljive u aritmetici dvostruke preciznosti su Gauss-Christoffelove kvadraturne formule. Gauss-Christoffelove kvadraturne formule su van opsega ovog udžbenika.

6.2.3 Newton-Cotesove formule sa pozitivnim težinama

Pored već pomenutih Newton-Cotesovih formula u praksi se koriste i formule kod kojih je $n \leq 7$. Ove formule imaju dobre numeričke osobine. Naime, kod ovih formula svi Newton-Cotesovi brojevi su pozitivni. Zato ih ovde navodimo zajedno sa formulama koje određuju grešku. Takođe, dajemo implementiranu funkciju `newtonCotes` koja obuhvata sve Newton-Cotesove formule dobrih numeričkih osobina. Koristimo notaciju kao u prethodnim teoremama.

Teorema 6.17 *Neka je $f \in C^6[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{5h}{288}\left(19f_0 + 75f_1 + 50f_2 + 50f_3 + 75f_4 + 19f_5\right) - \frac{275h^7}{12096}f^{(6)}(\psi), \quad \psi \in (a, b).$$

Teorema 6.18 *Neka je $f \in C^8[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{h}{140}\left(41f_0 + 216f_1 + 27f_2 + 272f_3 + 27f_4 + 216f_5 + 41f_6\right) - \frac{9h^9}{1400}f^{(8)}(\psi),$$

gde je $\psi \in (a, b)$.

Teorema 6.19 *Neka je $f \in C^8[a, b]$, onda je*

$$\int_a^b f(x)dx = \frac{7h}{17280}\left(751f_0 + 3577f_1 + 1323f_2 + 2989f_3 + 2989f_4 + 1323f_5 + 3577f_6 + 751f_7\right) - \frac{8183h^9}{518400}f^{(8)}(\psi), \quad \psi \in (a, b).$$

Funkicija `newtonCotes`, koja implementira sve do sada pomenute Newton-Cotesove kvadraturne formule, može da se napiše u sledećem obliku

```
function [ res ] = newtonCotes( n, a, b, fun )
%NEWTONCOTES(N,A,B,FUN) racuna vrednost odredjenog integrala
% funkcije fun na intervalu (a,b) koristeci Newton-Cotesovu
% formulu u n<=4 tacaka

switch n
    case 1
        %trapezno pravilo
        h = b-a;
        res = h/2*(fun(a)+fun(b));
    case 2
        %Simpsonovo pravilo
        h = (b-a)/2;
        res = h/3*(fun(a)+4*fun(a+h)+fun(b));
    case 3
        %Simpsonovo praviloa 3/8
        h = (b-a)/3;
        res = 3*h/8*(fun(a)+3*fun(a+h)+3*fun(a+2*h)+fun(b));
    case 4
        %Booleovo pravilo
        h = (b-a)/4;
        res = 7*fun(a)+32*fun(a+h)+12*fun(a+2*h);
        res = res+32*fun(a+3*h)+7*fun(b);
        res = 2*h/45*res;
    case 5
        h = (b-a)/5;
        res = 19*fun(a)+75*fun(a+h)+50*fun(a+2*h);
        res = res + 50*fun(b-2*h)+75*fun(b-h)+19*fun(b);
        res = 5*h/288*res;
    case 6
        h = (b-a)/6;
        res = 41*fun(a)+216*fun(a+h)+27*fun(a+2*h);
        res = res + 272*fun(a+3*h);
        res = res + 27*fun(b-2*h)+216*fun(b-h)+41*fun(b);
        res = h/140*res;
    case 7
        h = (b-a)/7;
        res = 751*fun(a)+3577*fun(a+h)+1323*fun(a+2*h);
        res = res + 2989*fun(a+3*h)+2989*fun(b-3*h);
        res = res + 1323*fun(b-2*h)+3577*fun(b-h)+751*fun(b);
        res = 7*h/17280*res;
    otherwise
        disp('vrednost n nije podrzana');
        res = 0;
end
```

```
end
```

Ove formule su praktično sve što je upotrebljivo iz sveta Newton-Cotesovih formula u okviru numeričke analize. Za dalje povećanje stepena algebarske tačnosti koriste se Gauss-Christoffelove kvadraturne formule.

6.2.4 Uopštene kvadraturne formule

Kako smo pokazali u prethodnom odeljku sa povećanjem stepena algebarske tačnosti, kod Newton-Cotesovih formula ne može se postići povećanje tačnosti sa kojom računamo integral u aritmetici konačne dužine. Međutim, analizom izraza kojim je zadata greška

$$R_{n+1} \sim \left(\frac{b-a}{n}\right)^n$$

možemo zaključiti da se povećanje tačnosti može postići smanjenjem intervala na kome se vrši numerička integracija. Naravno, ovaj zahtev je nemoguće nametnuti korisniku, u smislu da je moguće izračunavati integrale samo na intervalima male dužine. Međutim, moguće je interval na kome se vrši intregracija podelite na veliki broj intervala, pa nad tim intervalima primenjivati Newton-Cotesove formule koje smo pomenuli. Ovo se u praksi i radi. Ovako dobijene formule se nazivaju uopštene kvadraturne formule.

Pretpostavimo da treba izvršiti integraciju funkcije f na intervalu $[a, b]$. Postupamo na sledeći način. Izaberemo N, broj podintervala, i iskoristimo sledeću osobinu integrala

$$\int_a^b f(x)dx = \sum_{k=0}^{N-1} \int_{c_k}^{c_{k+1}} f(x)dx,$$

gde je $c_k = a + kH$, $k = 0, \ldots, N$, $H = (b-a)/N$. Na svakom od podintervala $[c_k, c_{k+1}]$, $k = 0, \ldots, N-1$, primenimo neku Newton-Cotesovu formulu malog algebarskog stepena tačnosti, recimo sa n tačaka. Na ovaj način dobijamo da je $h = (b-a)/(Nn)$, tako da h možemo da smanjujemo po volji. Smanjenje h postižemo menjajući N, pri čemu, n može biti relativno mali broj, dok god povećavamo samo N.

Najčešće se kao Newton-Cotesova formula koristi trapezna formula. U ovom slučaju dobijamo

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=0}^{N-1} \int_{c_k}^{c_{k+1}} f(x)dx = \sum_{k=0}^{N-1} \frac{h}{2}(f(c_k) + f(c_{k+1})) + R_{N+1}(f) \\ &= h\left(\frac{1}{2}f(a) + f(a+h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right) + R_{N+1}(f).\end{aligned}$$

Kako je kod trapezne formule $n = 1$, to je $H = (b-a)/N = (b-a)/(Nn) = h$, što smo iskoristili u formuli. Obično se zbog kratkoće pisanja koristi oznaka

$$T_N(f) = h\left(\frac{1}{2}f(a) + f(a+h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right),$$

tako da uopštena trapezna formula dobija sledeću formu

$$\int_a^b f(x)dx = T_N(f) + R_{N+1}(f).$$

Sledeća teorema daje ocenu ostatka kod uopštene trapezne formule.

Teorema 6.20 (Uopštena trapezna formula) *Neka je $f \in C^2[a,b]$, onda je*

$$\int_a^b f(x)dx = T_N(f) - \frac{(b-a)^3}{12N^2} f''(\psi), \quad \psi \in (a,b).$$

Kao što vidimo greška opada sa porastom broja podintervala kao $1/N^2$. Međutim, vidimo da sa porastom broja podintervala ne raste algebarski stepen tačnosti. Formula nije tačna već za polinom stepena dva.

Ilustrujmo prethodne činjenice jednim primerom. Primenimo uopštenu trapeznu formulu na izračunavanje

$$\int_0^1 x^2 dx = \frac{1}{3}. \tag{6.8}$$

U ovom slučaju je $f(x) = x^2$ i apsolutna greška je zadata sa

$$|R_{N+1}(f)| = \frac{(1-0)^3}{12N^2} 2 = \frac{1}{6N^2}.$$

Ako želimo tačnost 10^{-3} potrebno je $N \geq \sqrt{10^3/6} \approx 12.9$, dakle, $N = 13$ podintervala. Ako želimo tačnost 10^{-6} potrebno nam je $N \geq \sqrt{10^6/6} \approx 408.24$, dakle, $N = 409$ podintervala. Ovo je samo ilustracija koja pokazuje da je za povećanje tačnosti potreban priličan napor.

Primetimo da zbog efikasnosti povećanje broja podintervala treba realizovati na taj način da se sve prethodno izračunate vrednosti funkcije koriste. Naime, posmatrajmo izraze

$$T_N(f) = h\left(\frac{1}{2}f(a) + f(a+h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right),$$
$$T_{2N}(f) = \frac{h}{2}\left(\frac{1}{2}f(a) + f\left(a+\frac{h}{2}\right) + f(a+h) + \cdots + f(b-h) + f\left(b-\frac{h}{2}\right) + \frac{1}{2}f(b)\right).$$

Očigledno je da važi

$$T_{2N}(f) = \frac{1}{2}T_N(f) + \frac{h}{2}\left(f\left(a+\frac{h}{2}\right) + f\left(a+\frac{3h}{2}\right) + \cdots + f\left(b-\frac{h}{2}\right)\right).$$

Tako se povećanje tačnosti postiže na taj način što stalno dupliramo broj podintervala i izračunavamo vrednosti funkcije samo u tačkama koje se nalaze u sredini pređašnjih podintervala.

Na prethodno opisan način je konstruisana tabela 6.3. Prikazane su aproksimacije integrala T_N, apsolutna greška Δ_N, kao i teorijska greška $|R_{N+1}|$. Vidimo da se apsolutna greška i teorijska greška potpuno slažu. Kao što vidimo konvergencija je prilično spora.

Naravno, ništa nas ne sprečava da kao Newton-Cotesovu formulu, umesto trapezne, koristimo neku formulu od onih koje smo već pomenuli. Koristeći već implementiranu funkciju `newtonCotes` možemo vrlo jednostavno implementirati uopštene Newton-Cotesove formule. Funkcija koja implementira uopštene formule mogla bi izgledati ovako

```
function [ res ] = uopstenNewtonCotes( N, n, a, b, fun )
%UOPSTENNEWTONCOTES(N,n,a,b,fun) izracunava integral
% funkcije fun na intervalu (a,b) koristeci podelu na N podintervala
% i primenjujuci na svakom podintervalu Newton-Cotesovu formulu
% tip Newton-Cotesove formule je odredjen sa n,
```

N	T_N	Δ_N	$\lvert R_{N+1}\rvert$
2	.375	.417(−1)	.417(−1)
4	.34375	.104(−1)	.104(−1)
8	.3359375	.260(−2)	.260(−2)
16	.333984375	.651(−3)	.651(−3)
32	.33349609375	.163(−3)	.163(−3)
64	.3333740234375	.406(−4)	.406(−4)
128	.333343505859375	.102(−4)	.102(−4)
256	.333335876464844	.254(−5)	.254(−5)
512	.333333969116211	.636(−6)	.636(−6)
1024	.333333492279053	.159(−6)	.159(−6)

Tabela 6.3: Vrednost T_N, apsolutne greške Δ_N, i teorijske ocene apsolutne greške $|R_{N+1}|$ prilikom aproksimacije integrala (6.8)

```
% n=1 za trapezenu formulu
% n=2 za Simpsonovu formulu
% n=3 za Simpsonovo pravilo 3/8
% n=4 za Booleovo pravilo
% n mora biti manje od 8

H = (b-a)/N; c = a:H:b; res = 0;
for i=1:N
    res = res + newtonCotes(n,c(i),c(i+1),fun);
end
end
```

Kao ilustraciju navedimo teoremu koja daje ocenu ostatka u uopštenoj Simpsonovoj formuli.

Teorema 6.21 *Neka je $f \in C^4[a,b]$, onda je*

$$\int_a^b f(x)dx = \sum_{k=0}^{N-1} \frac{h}{3}\left(f(c_k) + f(c_k+h) + f(c_{k+1})\right) - \frac{(b-a)^5}{2880N^4} f^{(4)}(\psi), \quad \psi \in (a,b),$$

gde je $h = (b-a)/(2N)$, $c_k = a + 2kh$, $k = 0, \ldots, N$.

Primetimo da je u ovom slučaju algebarski stepen tačnosti povećan u odnosu na uopštenu trapeznu formulu i iznosi 3. Ovo zapažanje znači da bi uopštena Simpsonova formula morala da izračunava bez greške (6.8). Ovo je jednostavno proveriti, jer važi

```
>> uopstenNewtonCotes(2, 3, 0, 1, f)
ans =
   0.333333333333333
```

Generalno možemo reći da ako se na podintervalima, kojih ima N, primenjuje Newton-Cotesova formula sa n tačaka greška ima oblik

$$R_{N+1}(f) = C_N \frac{(b-a)^m}{N^{m-1}} f^{(m-1)}(\psi), \quad \psi \in (a,b),$$

gde je $m = 2[n/2] + 3$.

Konačno primetimo da uopštena trapezna formula može biti primenjena i u slučaju kad intervali $[c_k, c_{k+1}]$ nemaju istu dužinu. U ovom slučaju formula dobija oblik

$$\int_a^b f(x)dx = \sum_{k=0}^{N-1} \frac{c_{k+1} - c_k}{2}(f(c_{k+1}) + f(c_k)) + R_{N+1}(f).$$

Ova formula ima slična svojstva kao i formula sa intervalima koji imaju istu dužinu.

6.2.5 Otvorene Newton-Cotesove formule

Problem sa uopštenom trapeznom formulom je činjenica da je formula zatvorenog tipa. Ova činjenica onemogućava primenu formule na nesvojstvene integrale.

Pretpostavimo da treba odrediti vrednost integrala

$$\int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}} = \pi. \tag{6.9}$$

Očigledno je da ovde nije moguće primeniti Newton-Cotesove formule niti uopštene Newton-Cotesove formule, jer je vrednost funkcije u krajnjim tačkama intervala neograničena. U ovakvim slučajevima je neophodno primenjivati formule otvorenog tipa, jer ove formule ne uključuju vrednosti funkcije u krajnjim tačkama.

Najjednostavnija formula otvorenog tipa je sledeća formula

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) + \frac{h^3}{3}f''(\psi), \quad h = \frac{b-a}{2}, \quad \psi \in (a,b).$$

Ova formula je poznata kao formula srednje tačke. Intuicija iza formule srednje tačke je sledeća. Površina ispod grafika funkcije se aproksimira površinom pravougaonika sa stranicama $b-a$ i $f((b+a)/2)$. Naravno, tačka $(b+a)/2$ je srednja tačka o kojoj se govori.

Pod otvorenim Newton-Cotesovim kvadraturnim formulama podrazumevamo interpolacione kvadraturne formule konstruisane u ekvidistantnim čvorovima $x_\ell = x_0 + \ell h$, $\ell = 0, \ldots, n$, gde je $h = (b-a)/(n+2)$, $a = x_0 - h$, $b = x_n + h$, na ograničenom intervalu $[a,b]$ sa težinskom funkcijom $w = 1$.

Težine ovih kvadraturnih formula su, prema (6.6), određene na sledeći način

$$A_\ell = \int_a^b L_\ell(x)dx = \frac{(-1)^{n-\ell}h}{\ell!(n-\ell)!}\int_{-1}^{n+1} \frac{p^{(n+1)}}{p-\ell}dp, \quad \ell = 0, \ldots, n. \tag{6.10}$$

Na osnovu razmatranja u vezi sa interpolacionim formulama otvorenog tipa možemo formulisati teoremu o algebarskom stepenu tačnosti kvadraturnih formula otvorenog tipa.

Teorema 6.22 (Otvorene Newton-Cotesove kvadraturne formule) *Otvorena Newton-Cotesova kvadraturna formula*

$$\int_a^b f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + R_{n+1}(f),$$

sa ekvidistantnim čvorovima $x_\ell = x_0 + \ell h$, $\ell = 0, \ldots, n$, $h = (b-a)/(n+2)$, $a = x_0 - h$, $b = x_n + h$ i težinskim koeficijentima (6.10) *ima algebarski stepen tačnosti barem n.*

U praksi se sreću i formule sa $n = 1$, $n = 2$ i $n = 3$, koje se mogu opisati na sledeći način

$$\int_a^b f(x)dx = \frac{3h}{2}(f(a+h) + f(b-h)) + \frac{3h^3}{4}f''(\psi), \quad h = \frac{b-a}{3},$$
$$\int_a^b f(x)dx = \frac{4h}{3}(2f(a+h) - f(a+2h) + 2f(b-h)) + \frac{14h^5}{45}f^{(4)}(\psi), \quad h = \frac{b-a}{4},$$
$$\int_a^b f(x)dx = \frac{5h}{24}(11f(a+h) + f(a+2h) + f(b-2h) + 11f(b-h)) + \frac{95h^5}{144}f^{(4)}(\psi), \quad h = \frac{b-a}{5},$$

gde je $\psi \in (a, b)$. Formule za $n = 1$ i $n = 2$ su poznate kao trapezna formula i Milneova formula, redom. Formula srednje tačke kao što vidimo ima vrednost $n = 0$.

Greška je u opštem slučaju određena sledećom teoremom.

Teorema 6.23 (Greška otvorene Newton-Cotesove formule) *Neka je $n \in \mathbb{N}$ paran broj i $f \in C^{n+2}[a, b]$, onda postoji $\psi \in (a, b)$ tako da važi*

$$\int_a^b f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + \frac{h^{n+3}f^{(n+2)}(\psi)}{(n+2)!}\int_0^n x\prod_{\ell=0}^{n}(x-\ell)dx.$$

Neka je $n \in \mathbb{N}$ neparan broj i neka je $f \in C^{n+1}[a, b]$, onda postoji $\psi \in (a, b)$ tako da važi

$$\int_a^b f(x)dx = \sum_{\ell=0}^{n} A_\ell f(x_\ell) + \frac{h^{n+2}f^{(n+1)}(\psi)}{(n+1)!}\int_0^n \prod_{\ell=0}^{n}(x-\ell)dx.$$

Vidimo da se algebarski stepen tačnosti kod otvorenih Newton-Cotesovih kvadraturnih formula ponaša identično kao kod Newton-Cotesovih formula. Algebarski stepen tačnosti formule sa n čvorova je $2[n/2] + 2$.

Pomenute formule možemo implementirati na sledeći način

```
function [ res ] = newtonCotesOtvoren( n, a, b, f )
%NEWTONCOTESOTVOREN(N,A,B,FUN) racuna vrednost odredjenog integrala
% funkcije fun ina intervalu (a,b) koristeci otvorene Newton-Cotesove
% formule u n<=3 tacaka

switch n
    case 0
        %formula srednje tacke
        res = (b-a)*f((a+b)/2);
    case 1
        %trapezna formula
        h = (b-a)/3;
        res = 3*h/2*(f(a+h)+f(b-a));
    case 2
        %Milneova formula
        h = (b-a)/4;
        res = 4*h/3*(2*f(a+h)-f(a+2*h)+2*f(b-h));
    case 3
```

```
        h = (b-a)/5;
        res = 11*f(a+h)+f(a+2*h)+f(b-2*h)+11*f(b-h);
        res = 5*h/24*res;
    otherwise
        disp('vrednost n nije podrzana');
        res = 0;
end
end
```

Na primer, formula srednje tačke ima algebarski stepen tačnosti jedan. Ovu činjenicu jednostavno proveravamo

```
>>newtonCotesOtvoren(0,0,2,@(x) 1)
ans =
     2
>>newtonCotesOtvoren(0,0,2,@(x) x)
ans =
     2
```

6.2.6 Uopštene otvorene Newton-Cotesove formule

Najjednostavnija uopštena otvorena Newton-Cotesova formula je formula koja se zasniva na kvadraturnoj formuli srednje tačke.

Pretpostavimo sada da smo podelili interval $[a, b]$ na kome integralimo na N podintervala $[c_k, c_{k+1}]$, $k = 0, \ldots, N-1$, i da smo na svakom od njih primenili pravilo srednje tačke. Dobijamo formulu

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=0}^{N-1} \int_{c_k}^{c_{k+1}} f(x)dx \\ &= h\left(f\left(a+\frac{h}{2}\right) + f\left(a+\frac{3h}{2}\right) + \cdots + f\left(a+\frac{(2N-1)h}{2}\right)\right) + R_N(f),\end{aligned}$$

gde je $h = (b-a)/N$, $c_k = a + kh$, $k = 0, \ldots, N$. Radi kratkoće pisanja obično koristimo notaciju

$$M_N(f) = h\left(f\left(a+\frac{h}{2}\right) + f\left(a+\frac{3h}{2}\right) + \cdots + f\left(a+\frac{(2N-1)h}{2}\right)\right).$$

Sledeća teorema u potpunosti opisuje uopštenu kvadarturnu formulu srednje tačke.

Teorema 6.24 (Uopštena formula srednje tačke) *Neka je $f \in C^2[a, b]$, onda je*

$$\int_a^b f(x)dx = M_N(f) + \frac{(b-a)^3}{24N^2} f''(\psi), \quad \psi \in (a, b).$$

Kao što vidimo formula ima veoma slične osobine kao i uopštena trapezna formula, jedina razlika je što je uopštena formula srednje tačke otvorenog tipa. Slično kao kod uopštene trapezne formule ima smisla samo povećavati broj intervala dva puta.

N	M_N	Δ_N
128	3.034702782443241	.107
256	3.065995200277488	.756(−1)
512	3.088131919778891	.535(−1)
1024	3.103788346912556	.378(−1)
2048	3.114860314823282	.267(−1)
4096	3.122689803265439	.189(−1)
8192	3.128226237815358	.134(−1)
16384	3.132141141320574	.945(−2)
32768	3.134909414906539	.668(−2)

Tabela 6.4: Aproksimacija vrednosti integrala M_N uopštenim pravilom srednje tačke, i apsolutna greška aproksimacije Δ_N, prilikom računanja integrala (6.9).

Tabela 6.4 sadrži vrednosti aproksimacije integrala M_n i apsolutnu grešku Δ_N, prilikom aproksimacije vrednosti integrala (6.9). Kao što vidimo konvergencija postoji, ali je jako spora. Primetimo takođe da se teorema o uopštenoj formuli srednje tačke ne može primeniti u ovom slučaju jer funkcija $1/\sqrt{1-x^2}$ nije dva puta diferencijabilna na intervalu $[-1,1]$. Na osnovu iznetih podataka možemo zaključiti da kad god povećamo broj tačaka dva puta, smanjimo apsolutnu grešku za otprilike $2/3$ puta. Prethodno zapažanje upućuje na eksperimentalnu zakonitost $\Delta_N \sim N^\gamma$, pri čemu je $2^\gamma \approx 2/3$. Odavde dobijamo $\gamma \approx -.6$, tako da je potreban veliki napor da bi se dobile makar tri značajne cifre u rezultatu. Ovakvi problemi se obično nazivaju zasićenim problemima i nisu retki.

Ako pretpostavimo da je $\Delta_N = N^{-.6}$ za postizanje tačnosti 10^{-3} potrebno nam je 10^5 podintervala.

Umesto otvorene Newton-Cotesove formule srednje tačke mogli smo koristiti i neku drugu otvorenu Newton-Cotesovu formulu. Sledeća funkcija implementira uopštene otvorene Newton-Cotesove formule.

```
function [ res ] = uopstenNewtonCotesOtvoren( N, n, a, b, fun )
%UOPSTENANEWTONCOTESOVAOTVOREN(N,n,a,b,fun) izracunava integral
% funkcije fun na intervalu (a,b) koristeci podelu na N podintervala
% i primenjujuci na svakom podintervalu otvorenu Newton-Cotesovu formulu
% tip Newton-Cotesove formule je odredjen sa n,
% n=0 za formulu srednje tacke
% n=1 za trapeznu formulu
% n=2 za Milneovu formulu
% n mora biti manje od 4

H = (b-a)/N; c = a:H:b; res = 0;
for i=1:N
    res = res + newtonCotesOtvoren(n,c(i),c(i+1),fun);
end
end
```

Greška uopštene otvorene Newton-Cotesove formule se ponaša slično kao i greška uopštene Newton-Cotesove formule. Ako koristimo otvorenu Newton-Cotesovu formulu sa $n \in \mathbb{N}_0$ tačaka,

za uopštenu formulu sa N podintervala greška se u opštem slučaju ponaša

$$R_{N+1}(f) = C_N \frac{(b-a)^m}{N^{m-1}} f^{(m-1)}(\psi), \quad \psi \in (a,b),$$

gde je $m = 2[n/2] + 3$.

Na primer, uopštena otvorena Milneova formula ima algebarski stepen tačnosti tri, što jednostavno proveravamo

```
>>uopstenNewtonCotesOtvoren(1,2,0,1,@(x) x^3)
ans =
                    0.25
```

Primetimo da smo izabrali podelu na samo jedan podinterval.

6.2.7 Divergencija Newton-Cotesovih formula

Poseban problem numeričke integracije prilikom integracije Newton-Cotesovim formulama predstavlja činjenica da formule ne konvergiraju za sve neprekidne funkcije, što je posledica činjenice da je

$$\sum_{\ell=0}^{n} |A_\ell| \to +\infty, \quad n \to +\infty.$$

Radi ilustracije posmatrajmo integraciju funkcije $f(x) = 1/(1+x^2)$ na intervalu $(-10, 10)$, dakle, iste funkcije koja nam je poslužila za ilustraciju značajnog odstupanja interpolacionog polinoma od funkcije koja se interpolira. Grafik interpolacionog polinoma i funkcije je prikazan na slici 4.15, levo.

Tabela 6.5 prikazuje relativnu grešku prilikom integracije funkcije f Newton-Cotesovim formulama. Vidimo da relativna greška raste sa porastom algebarske tačnosti formule umesto da opada. Ova pojava nije posledica gubitka značajnih cifara kako je opisano za formulu sa sto tačaka u sekciji 6.2.2. Prosto ovo je posledica činjenice da Newton-Cotesove formule ne konvergiraju za sve neprekidne funkcije.

k	2	10	18	26	34	42	50
r	.35(1)	.30(1)	.74(2)	.29(3)	.13(5)	.70(7)	.39(9)

Tabela 6.5: Divergencija Newton-Cotesovih formula pri integraciji Rungeove funkcije $f(x) = 1/(1+x^2)$. Brojevi u zagradama predstavljaju eksponente broja deset.

6.2.8 Uticaj tačnosti ulaznih podataka

U slučaju da ulazni podaci nisu tačni nego su opterećeni greškama, možemo očekivati da će doći do izvesnih grešaka prilikom izračunavanja određenog integrala korišćenjem numeričkih metoda. Međutim, kao što ćemo videti metode numeričke integracije su stabilne u smislu da skoro ne dolazi do povećanja apsolutne greške u izlaznim podacima.

Teorema 6.25 *Neka su podaci kojima je određena funkcija dati sa gornjom granicom apsolutne greške Δ. Onda je gornja granica uopštene trapezne formule i uopštene formule srednje tačke, takođe, određena sa $(b-a)\Delta$.*

Dokaz. Označimo podintegralnu funkciju sa f, a funkciju koju mi možemo da merimo sa $\tilde{f}$. Kako su podaci zadati sa gornjom granicom apsolutne greške Δ, to važi

$$|f(c_k) - \tilde{f}(c_k)| \leq \Delta, \quad k = 0, \ldots, N,$$

gde smo sa $[c_k, c_{k+1}]$, $k = 0, \ldots, N-1$, označili podintervale u podeli intervala $[a, b]$ na kome integralimo funkciju. Međutim, vrednost integrala kod uopštene trapezne formule je određena sa

$$T_N(f) = h\left(\frac{1}{2}f(c_0) + f(c_1) + \ldots + \frac{1}{2}f(c_N)\right),$$

a za funkciju $\tilde{f}$ je određena sa

$$T_N(\tilde{f}) = h\left(\frac{1}{2}\tilde{f}(c_0) + \tilde{f}(c_1) + \ldots + \frac{1}{2}\tilde{f}(c_N)\right).$$

Oduzimanjem nalazimo

$$\begin{aligned}
|T_N(f) - T_N(\tilde{f})| &= h\left|\frac{1}{2}(f-\tilde{f})(c_0) + (f-\tilde{f})(c_1) + \ldots + \frac{1}{2}(f-\tilde{f})(c_N)\right| \\
&\leq h\left(\frac{1}{2}|(f-\tilde{f})(c_0)| + |(f-\tilde{f})(c_1)| + \ldots + \frac{1}{2}|(f-\tilde{f})(c_N)|\right) \\
&\leq h\left(\frac{1}{2} + 1 + \ldots + \frac{1}{2}\right)\Delta = \frac{b-a}{N}N\Delta = (b-a)\Delta.
\end{aligned}$$

Na isti način se pokazuje nejednakost i u slučaju uopštene formule srednje tačke. □

Na osnovu teoreme o srednjoj vrednosti integrala, znamo da za neprekidnu podintegralnu funkciju možemo vrednost integrala odrediti na sledeći način

$$\int_a^b f(x)dx = (b-a)f(c), \quad c \in (a, b).$$

Koristeći ovu teoremu i dokazanu teoremu o gornjoj granici apsolutne greške možemo odrediti gornju granicu relativne greške

$$r_T = \frac{(b-a)\Delta}{(b-a)f(c)} = r_f.$$

Kao što vidimo gornja granica relativne greške se ne povećava već ostaje približno ista, pod uslovom da su vrednosti funkcije približno konstantne unutar intervala integracije, dakle, pod uslovom da možemo uzeti da je $\Delta/f(c) = r_f$.

6.2.9 Problemi numeričke integracije

Veoma je jednostavno naći funkciju koju je nemoguće integraliti koristeći današnji hardware metodima prezentovanim u ovom odeljku. Najjednostavniji primer su trigonometrijske funkcije. Posmatrajmo sledeći integral

$$\int_0^1 \sin \omega x dx = \frac{1 - \cos\omega}{\omega}, \quad \omega > 0.$$

Kakav god hardware da imate, sa kolikom god hoćete velikom vrednošću clocka ili koliko god veliki broj jezgara, uvek postoji dovoljno velika vrednost ω tako da je vrednost integrala nemoguće odrediti pomenutim metodima u razumnom vremenu.

Još jedan primer integrala koji se jako teško može izračunati prikazanim metodima su nesvojstveni integrali. Jedan primer smo već dali

$$\int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}} = \pi.$$

Ovakvi integrali, kao i integrali koji predstavljaju integrale analitičkih funkcija koje imaju singularitete u blizini intervala integracije obično pokazuju već ilustrovano svojstvo zasićenja. Dakle, konvergiraju veoma sporo sa povećanjem broja tačaka u kvadraturnoj formuli.

6.2.10 Implementacija numeričke integracije u Matlabu

Oblast numeričke integracije je vrlo dinamična oblast istraživanja. Broj funkcija implementiranih u `Matlabu`, koje se mogu koristiti za numeričku integraciju je prilično veliki. Mi ćemo ograničiti pažnju na dve `trapz` i `integral`.

Funkcija `trapz` ima više formata sa kojima može biti pozvana. Za nas su od interesa dva

```
Z=trapz(Y)
Z=trapz(X,Y)
```

U oba slučaja se podrazumeva da je `Y` vektor odmeraka funkcije koja se treba integraliti, sem što u drugom slučaju dajemo eksplicitno tačke u kojima je funkcija odmerena, vektor `X`, dok se u prvom slučaju podrazumeva da je funkcija odmerena u tačkama počev od broja 1 sa korakom 1. Funkcija vraća vrednost primene uopštene trapezne formule nad datim podacima. Pri tome, nije nužno da podaci budu odmereni u ekvidistantnim tačkama prilikom primene drugog formata.

```
>>Y=1:10;
>>trapz(Y)
ans =
          49.5
>>X=[2 2+5*sort(rand(1,20)) 7];
>>Y=log(X);
>>trapz(X,Y)
ans =
           7.2311073610266
```

Stvarna vrednost integrala funkcije log na intervalu $[2, 7]$ iznosi približno 7.24, tako da dobijamo ne tako lošu aproksimaciju.

Funkcija `integral` je mnogo ozbiljnija od funkcije `trapz`. Funkcija ima više mogućih formata, zavisno do toga da li navodimo opcije ili ne. Sve formate možemo da obuhvatimo na sledeći način

```
q=integral(fun,xmin,xmax,imeParametra,vrednostParametra)
```

Ulazni argumenti `imeParametra` i `vrednostParametra` su opcioni i ne moraju biti navedeni pri svakom pozivu. Argument `fun` je podatak tipa `function_handle` koji predstavlja pokazivač na funkciju čiji integral pokušavamo da izračunamo, dok su argumenti `xmin` i `xmax` donja i gornja granica integracije. Izlazni argument `q` predstavlja vrednost aproksimacije određenog integrala.

```
>>q=integral(@sin,0,1)
q =
```

```
   0.459697694131860
>>q=integral(@(x) exp(-x^2),-inf,inf)
q =
   1.772453850780313
```

Argument `imeParametra` može da uzme neku od sledećih vrednosti: `'AbsTol'`, u kom slučaju funkcija očekuje vrednost dozvoljene apsolutne greške kao sledeći argument u pozivu funkcije na mestu argumenta `vrednostParametra`, `'RelTol'`, u kom slučaju funkcija `integral` očekuje dozvoljenu relativnu grešku kao vrednost argumenta `vrednostParamentra`, `'ArrayValued'`, u kom slučaju funkcija očekuje logičku vrednost argumenta `vrednostParametra`, `'Waypoints'`, u kom slučaju na mestu argumenta `vrednostParametra` očekuje vektor tačaka koje mora uključiti u skup čvorova u kvadraturnoj formuli.

Podrazumevana vrednost parametra `'AbsTol'` iznosi 10^{-10}, dok podrazumevana vrednost parametra `'RelTol'` iznosi 10^{-6}. Podrazumevana vrednost parametra `'ArrayValued'` iznosi `false`, drugim rečima, podrazumeva se da je funkcija `fun` skalarna. Podrazumevana vrednost parametra `'Waypoints'` je prazan vektor. Parametar `'Waypoints'` se koristi prilikom kompleksne integracije i mi ga dalje nećemo proučavati.

Funkcija `integral` povećava broj čvorova u kvadraturnoj formuli sve dok se ne ispuni uslov

$$|q - Q| \leq \max\{\text{AbsTol}, \text{RelTol}\,|q|\},$$

gde je q trenutna aproksimacija određenog integrala, a Q je prava (nepoznata) vrednost određenog integrala.

Ilustrovaćemo dejstvo opcija `'AbsTol'` i `'RelTol'`. Na primer, odredimo koristeći funkciju `integral` vrednost integrala

$$\int_0^1 \omega \sin \omega x dx = 1 - \cos \omega.$$

Ovde je podintegralna funkcija neprekidna, čak diferencijabilna proizvoljnog reda. Međutim, sa porastom parametra ω, funkcija ima veliki broj perioda na intervalu $[0, 1]$, tako da dolazi do gubitka značajnih cifara prilikom izračunavanja kvadraturnih suma jer funkcija često menja znak. Ova pojava onemogućava izračunavanje vrednosti integrala sa zahtevanom tačnošću. Na primer, posmatrajmo sledeći skript

```
omega=10;
fun=@(x) omega*sin(omega*x);
exact=1-cos(omega);
q6=integral(fun,0,1);
q14=integral(fun,0,1,'RelTol',10^(-14));
[exact abs(q4/exact-1) abs(q14/exact-1)]
```

Rezultat izvršenja je

```
ans =
   1.839071529076453                   0                   0
```

Međutim, promenimo skript na sledeći način

```
omega=10^3;
fun=@(x) omega*sin(omega*x);
```

```
exact=1-cos(omega);
q6=integral(fun,0,1,'AbsTol',10^(-20));
q14=integral(fun,0,1,'AbsTol',10^(-20),'RelTol',10^(-14));
[exact abs(q4/exact-1) abs(q14/exact-1)]
```

Rezultat izvršenja je sada

```
ans =
   0.437620923709297   0.000000000000797   0.000000000000293
```

Prvi poziv funkcije uspeva da izračuna vrednost integrala u granicama dozvoljene relativne greške 10^{-6}, dok drugi poziv funkcije ne uspeva da izračuna vrednost integrala sa vrednošću relativne greške 10^{-14}, što se iz priloženih rezultata vidi. Naime, drugi poziv funkcije izračunava integral sa gornjom granicom relativne greške $3 \cdot 10^{-13}$. Po završetku izvršenja funkcija `integral` generiše poruku koja u suštini znači da se od postignute relativne greške $3 \cdot 10^{-13}$ bolje ne može.

Ako vrednost ω postavimo na vrednost 10^6 ni prvi ni drugi poziv funkcije ne uspevaju da postignu zahtevanu relativnu grešku, kao ni apsolutnu. Dobija se sledeći rezultat

```
ans =
   0.0632478724668553   8806.949293846   8806.94929385367
```

Kao što vidimo, ne samo da zadata relativna greška, niti apsolutna, nije dostignuta, nego je postignuta relativna greška reda veličine 10^4.

Pomenuta poruka koju vraćaju pozivi funkcije `integral` u ovim slučajevima izgleda ovako

```
Warning: Reached the limit on the maximum number
of intervals in use. Approximate bound on error
is   1.1e+05. The integral may not exist, or it
may be difficult to approximate numerically to
the requested accuracy.
> In funfun/private/integralCalc>iterateScalarValued at 372
  In funfun/private/integralCalc>vadapt at 133
  In funfun/private/integralCalc at 76
  In integral at 89
```

Ova poruka u stvari može da nam posluži za razumevanje algoritma koji funkcija koristi. Kao što vidimo funkcija obaveštava da je postignut maksimalan broj intervala koji se može koristiti. Naravno, ovo su podintervali na koje se deli interval $[0, 1]$ prilikom izračunavanja integrala. Primetimo da u spisku opcija funkcije nije omogućena promena maksimalnog broja podintervala sa kojima funkcija radi, tako da smo u ovom slučaju bespomoćni.

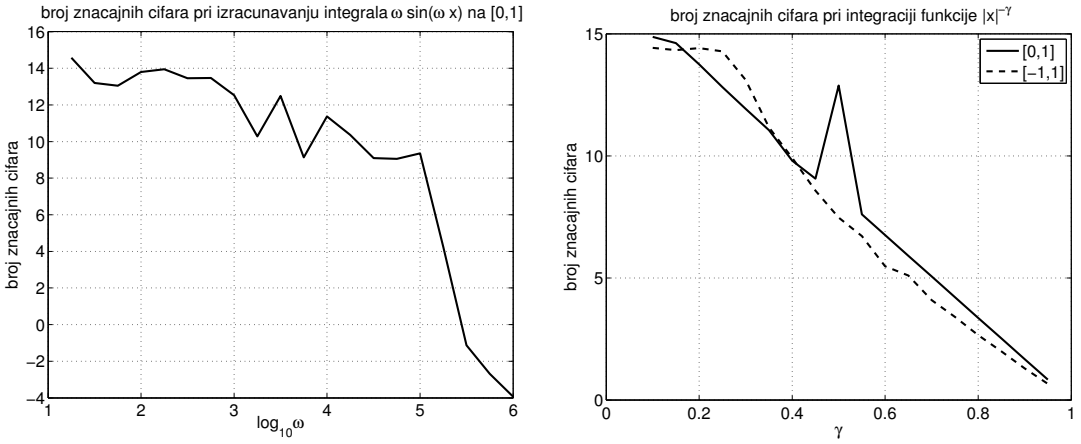
U poruci dobijamo obaveštenje da integral možda ne postoji ili ga je teško izračunati sa zahtevanom apsolutnom greškom. Mogućnost nepostojanja integrala je prilično realna mogućnost, kao što znamo nisu sve funkcije integrabilne, ali je neprimenljiva u ovom slučaju. Na primer, možemo pokušati da odredimo vrednost izraza

$$\int_0^1 \frac{dx}{x^2}.$$

Ovaj integral ne postoji, u smislu da ne postoji realan broj kome je ovaj simbol jednak. Ako bismo pokušali da izračunamo ovaj integral funkcijom `integral`, dobili bismo istu poruku, s tim što ovoga puta integral stvarno ne postoji.

U našem primeru je u pitanju druga mogućnost, prosto je nemoguće odrediti integral sa zadatom tačnošću. Postoje napredni algoritmi koji omogućavaju izračunavanje ovakvih integrala, ali su oni daleko izvan okvira ovog udžbenika.

Slika 6.6, levo, prikazuje zavisnost broja značajnih cifara od vrednosti ω koja se može dobiti upotrebom funkcije `integral`. Kao što vidimo pri vrednosti $\omega = 10^6$ nemamo nijednu značajnu cifru u izračunatoj vrednosti integrala. Drugim rečima, trenutna implementacija funkcije `integral` može uspešno da integrali signale frekvencije do vrednosti 100kHz.



Slika 6.6: Slika levo prikazuje broj značajnih cifara u funkciji ω pri integraciji funkcije $\omega \sin(\omega x)$ na $[0, 1]$, slika desno prikazuje broj značajnih cifara u funkciji γ pri integraciji funkcije $|x|^{-\gamma}$ na intervalima $[0, 1]$ i $[-1, 1]$.

Vratimo se određivanju nesvojstvenih integrala. Posmatrajmo integrale neograničenih funkcija na ograničenim intervalima. Da bi funkcija `integral` uspešno određivala vrednosti nesvojstvenih integrala neophodno je singularitet postaviti na krajeve intervala integracije. Funkcija `integral` podrazumeva da je integrand neprekidan i ograničen unutar intervala integracije. Na primer, posmatrajmo sledeći skript

```
fun=@(x) 1./sqrt(abs(x));
tacno=4;
q6=integral(fun,-1,1);
q14=integral(fun,-1,1,'RelTol',10^(-14));
[tacno abs(q6/tacno-1) abs(q14/tacno-1)]
```

Rezultat izvršenja je

```
ans =
   4.000000000000000   0.000000539891732   0.000000000032952
```

uz neizostavnu poruku o grešci, jer drugi poziv funkcije ne uspeva da izračuna vrednost integrala sa zadatom relativnom greškom. U pitanju je izračunavanje integrala

$$\int_{-1}^{1} \frac{dx}{|x|^{1/2}} = 4.$$

U ovom slučaju mi zloupotrebljavamo algoritam, jer očigledno podintergalna funkcija ima singularitet unutar intervala integracije u tački $x = 0$.

Ispravno je postaviti singularitete na krajeve intervala integracije. U našem slučaju to bi značilo izračunavanje dva integrala, koji su u ovom slučaju isti, a u generalnom slučaju ne moraju biti, tako da dobijamo

```
fun=@(x) 1./sqrt(abs(x));
tacno=4;
q6=integral(fun,-1,0)+integral(fun,0,1);
q14=integral(fun,-1,0,'AbsTol',10^(-20),'RelTol',10^(-13));
q14=q14+integral(fun,0,1,'AbsTol',10^(-20),'RelTol',10^(-13));
[tacno abs(q6/tacno-1) abs(q14/tacno-1)]
```

Rezultat izvršenja skripta je

```
ans =
   4.000000000000000   0.000000000000118   0.000000000000133
```

U ovom slučaju prvi poziv funkcije postiže relativnu grešku manju od $1.2 \cdot 10^{-13}$, dok drugi poziv funkcije dostiže relativnu grešku od $1.3 \cdot 10^{-13}$. Kao što vidimo procena postignute gornje granice relativne greške nije apsolutno tačna i funkcija izlazi a da nije stvarno postignuta zadata relativna greška 10^{-13}. Međutim, vrednosti zadate i postignute relativne greške se razlikuju relativno malo. Pomenimo i da dalje smanjenje zahtevane relativne greške ne proizvodi dalje povećanje tačnosti vrednosti integrala, jer jednostavno format dvostruke preciznosti onemogućava izračunavanje integrala sa većom preciznošću.

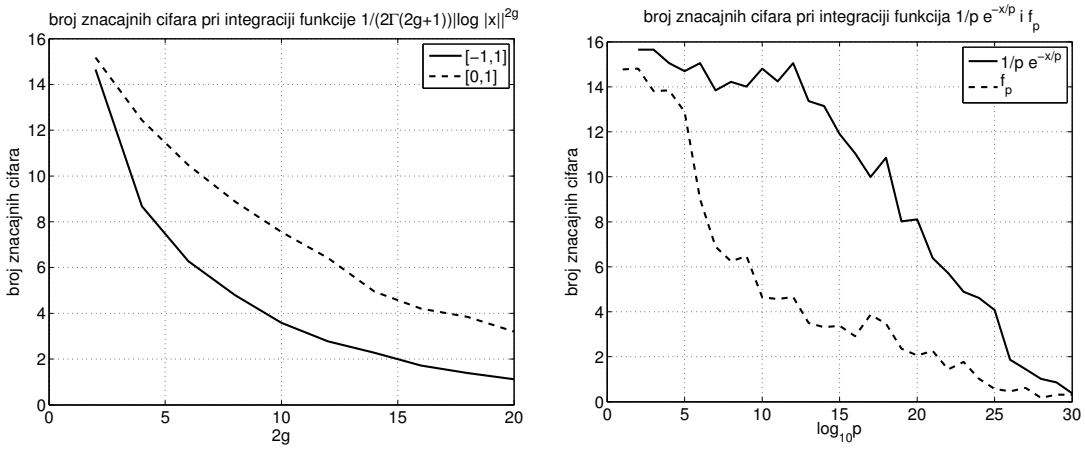
Slika 6.6, desno, ilustruje zavisnost broja značajnih cifara u vrednosti integrala u zavisnosti od parametra γ u izrazima

$$\int_{-1}^{1} \frac{dx}{|x|^\gamma} = \frac{2}{1-\gamma}, \quad \int_{0}^{1} \frac{dx}{x^\gamma} = \frac{1}{1-\gamma}.$$

Kao što vidimo, za male vrednosti γ čak je povoljnije singularitet imati unutar intervala integracije. Međutim, za vrednosti parametra γ veće od .4 vidimo da je povoljnije postaviti singularitet na kraj intervala integracije. Naglo povećanje broja značajnih cifara u okolini vrednosti $\gamma = .5$ izgleda prilično čudno i verovatno je posledica činjenice da funkcija `integral` ima ugrađenu heuristički metod za prepoznavanje singulariteta $|x|^{-1/2}$ na krajevima intervala integracije i njegovu uspešnu integraciju. Iako se na ovom primeru ne vidi da je značajno bolje postaviti singularitet na kraj intervala, na slici 6.7, levo, je prikazan broj značajnih cifara u funkciji parametra $2g$ pri integraciji

$$\int_{-1}^{1} \frac{1}{2(2g)!} |\log|x||^{2g} dx = 1, \quad \int_{0}^{1} \frac{1}{2(2g)!} |\log|x||^{2g} dx = \frac{1}{2}.$$

Kao što vidimo, ovde je broj značajnih cifara veći bar za dve kad se singularitet postavi na kraj intervala. Ako posmatramo još komplikovanije funkcije, sa većim brojem singulariteta unutar intervala integracije, možemo dobiti još veću razliku pri integraciji sa singularitetima na krajevima intervala i sa singularitetima unutar intervala integracije. Problem sa singularitetima unutar intervala integracije je što algoritam mora sam da ih pronađe i da prilagodi algoritam integracije kako bi sa dovoljnim brojem značajnih cifara odredio vrednost integrala na podintervalima u blizini singulariteta.



Slika 6.7: Slika levo daje zavisnost broja značajnih cifara u funkciji $2g$ pri integraciji funkcije $1/(2(2g)!)|\log|x||^{2g}$ na $[0,1]$ i $[-1,1]$, slika desno prikazuje broj značajnih cifara u funkciji p pri integraciji funkcija f_p i $p^{-1}e^{-x/p}$ na intervalu $[0,+\infty)$.

Ilustracija problema koji mogu nastati prilikom integracije na neograničenim intervalima se dobija jednostavno. Dovoljno je, za $p>0$, posmatrati integraciju familije funkcija

$$f_p(x)=\begin{cases}1, & x<p,\\ e^{-(x-p)}, & x\geq p,\end{cases}$$

na intervalu $[0,+\infty)$. Svaka od funkcija f_p je neprekidna na intervalu $[0,+\infty)$. Međutim, sa porastom parametra p funkcija `integral` uspeva da izračunava integral sa prihvatljivom relativnom greškom sve do vrednosti $p=10^5$. Kad se dostigne ova vrednost dobijamo poruku o grešci da je prekoračen maksimalan dozvoljen broj intervala u podeli, koju smo i do sada imali prilike da vidimo. Naime, funkcija `integral` izračunava integral tako što izračunava vrednost integrala na nekom ograničenom intervalu $[0,a]$ i prilikom izračunavanja pokušava da oceni vrednost integrala na intervalu $[a,+\infty)$. Uspešnost integracije zavisi od mogućnosti da se zanemari vrednost integrala na intervalu $[a,+\infty)$. Naravno, ovo je prilično uprošćeno rečeno ali izražava suštinu algoritma. Pri ovome postoji strategija na osnovu koje se konstruiše interval $[0,a]$. U našem slučaju za $p=10^5$, jednostavno funkcija f_p ne opada dovoljno brzo tako da je nemoguće zanemariti integral na ostatku intervala integracije za unapred zadatu vrednost pokušaja zanemarivanja. Ova situacija stvara uslove da funkcija generiše poruku o grešci.

Familija funkcija f_p služi samo kao model. Slično ponašanje bismo dobili kad bismo vršili integraciju neke funkcije koja takođe ne opada dovoljno brzo. Recimo, prilikom izračunavanja integrala

$$\int_0^{+\infty} p^{-1}e^{-x/p}dx=1.$$

Slika 6.7, desno, daje zavisnost broja značajnih cifara pri integraciji funkcija f_p i $p^{-1}e^{-x/p}$ na intervalu $[0,+\infty)$. Vidimo da je situacija prilikom integracije funkcije f_p prilično lošija. Ovo možemo pripisati nedovoljnoj glatkosti funkcije f_p i posebno već opisanoj nedovoljnoj brzini opadanja na intervalu $[0,p]$.

Pomenimo na kraju da se često prilikom korišćenja funkcije `integral` dešava da funkcija vrati beskonačnu vrednost kao rezultat integracije ako podintegralna funkcija ima singularitet na kraju intervala integracije. Na primer, sledeća sekvenca naredbi

```
>>fun=@(x) 1./x.^(6/10);
>>integral(fun,0,1,'RelTol',10^(-14))
Warning: Infinite or Not-a-Number value encountered.
> In funfun/private/integralCalc>iterateScalarValued at 349
  In funfun/private/integralCalc>vadapt at 133
  In funfun/private/integralCalc at 76
  In integral at 89
ans =
   Inf
```

proizvodi beskonačnost kao rezultat, uz poruku da je prilikom izračunavanja funkcije došlo do pojave vrednosti `inf` ili `nan`.

Pojava beskonačnosti je posledica premale dozvoljene vrednosti gornje granice relativne greške, koja najverovatnije prouzrokuje upotrebu neke zatvorene kvadraturne formule. Da bismo bili u stanju da iskoristimo do maksimuma mogućnosti funkcije `integral` možemo se poslužiti sledećim skriptom

```
q14=inf;
reltol=2^(-52);
fun=@(x) 1./x.^(6/10);
while q14==inf
   reltol=2*reltol;
   q14=integral(fun,0,1,'RelTol',reltol,'AbsTol',10^(-20));
end
[q14 reltol]
```

Ovaj skript, uz veliku količinu poruka o detekciji vrednosti `inf` ili `nan`, generiše izlaz

```
ans =
   2.499999558169475   0.000000014901161
```

Kao što vidimo, dobili smo vrednost integrala približno 2.5 sa gornjom granicom relativne greške $1.5 \cdot 10^{-8}$. Stvarna relativna greška je reda veličine $1.8 \cdot 10^{-7}$. Koristeći ovaj metod nacrtani su svi grafici vezani za upotrebu funkcije `integral`.

Glava 7

Obične diferencijalne jednačine

7.1 Cauchyev problem

U teoriji diferencijalnih jednačina Cauchyev problem se postavlja u sledećem obliku

$$y' = f(x, y), \quad y(x_0) = y_0, \tag{7.1}$$

gde je $y = y(x)$ nepoznata funkcija koju treba odrediti. Funkcija $f : D \to \mathbb{R}$ je poznata funkcija definisana nad nekim domenom $D \subset \mathbb{R}^2$. Uslov $y(x_0) = y_0$ naziva se početni uslov, i u odgovarajućim uslovima obezbeđuje jedinstvenost rešenja Cauchyevog problema.

Na primer, jedna forma Cauchyevog problema poznata iz osnovnih kurseva matematike je oblika

$$y' = Ay, \quad y(x_0) = y_0, \tag{7.2}$$

gde je $A \in \mathbb{R}$. Rešenje ovog problema je moguće dobiti u analitičkom obliku

$$y(x) = y_0 e^{A(x-x_0)}.$$

Ovaj problem će nam poslužiti za ilustraciju bitnih koncepata numeričke integracije diferencijalnih jednačina.

Da bismo mogli da znamo koji Cauchyev problem ima jedinstveno rešenje, drugim rečima, da bismo mogli da znamo koji Cauchyev problem uopšte ima smisla numerički rešavati, navodimo sledeću teoremu koja daje potrebne uslove za egzistenciju jedinstvenog rešenja Cauchyevog problema. Napominjemo da teorema koju navodimo nije jedina, niti je 'najsnažnija', u smislu da su sve postojeće teoreme koje se bave egzistencijom rešenja Cauchyevog problema njene posledice.

Teorema 7.1 (Egzistencija rešenja Cauchyevog problema) *Neka je*

$$D = \{(x, y) \,|\, |x - x_0| \leq \alpha, \quad |y - y_0| \leq \beta\}$$

domen nad kojom je funkcija f neprekidna, neka za $(x, y) \in D$ važi $|f(x, y)| \leq M$, neka za svako $(x, y_1), (x, y_2) \in D$ važi

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|, \tag{7.3}$$

i neka je $h \leq \min\{\alpha, \beta/M\}$.

Onda postoji jedinstvena funkcija $y \in C^1[x_0 - h, x_0 + h]$ tako da važi $y' = f(x, y)$, za $x \in [x_0 - h, x_0 + h]$ i $y(x_0) = y_0$.

Uslov (7.3) se naziva Lipschitzov uslov, a vrednost L se naziva Lipschitzova konstanta funkcije f.

Naš problem (7.2) zadovoljava uslove teoreme. U ovom slučaju je $f(x,y) = Ay$, funkcija f je neprekidna na $\mathbb{R}^2$. Funkcija f zadovoljava Lipschitzov uslov, jer je

$$|f(x,y_1) - f(x,y_2)| = |A||y_1 - y_2|,$$

tako da možemo uzeti $L = |A|$. Ako izaberemo

$$D = \{(x,y) \,|\, |x - x_0| \leq \alpha,\, |y - y_0| \leq \beta\},$$

onda važi

$$|f(x,y)| = |Ay| = |A||y| = |A||y - y_0 + y_0| \leq |A|(|y - y_0| + |y_0|) \leq |A|(\beta + |y_0|) = M.$$

Ako izaberemo $h \leq \min\{\alpha, \beta/M\}$, na osnovu pomenute teoreme postoji tačno jedno rešenje Cauchyevog problema (7.2). Naravno, to je upravo naše rešenje.

7.2 Eulerov metod

Eulerov metod je najstariji metod za numeričku integraciju običnih diferencijalnih jednačina. Nama je od posebnog značaja, jer će nam poslužiti za ilustraciju osnovnih koncepata neophodnih za razumevanje komplikovanijih metoda za numeričku integraciju diferencijalnih jednačina.

Ideja Eulerovog metoda polazi od razvoja u Taylorov red rešenja Cauchyevog problema. Naime, ako razvijemo u red rešenje Cauchyevog problema nalazimo

$$y(x+h) = y(x) + y'(x)h + R_2(x+h;x,y) = y(x) + hf(x,y(x)) + R_2(x+h;x,y).$$

Ako zanemarimo ostatak Taylorovog polinoma dobićemo sledeći metod

$$y(x+h) = y(x) + hf(x,y(x)), \quad y(x_0) = y_0.$$

Vidimo da je moguće odrediti vrednost $y(x+h)$, ako poznajemo vrednost $y(x)$. Kako sa sigurnošću jedino poznajemo vrednost $y(x_0)$, to možemo odrediti $y(x_0+h)$. Zatim koristeći vrednost $y(x_0+h)$, moguće je odrediti vrednost $y(x_0+2h)$, i tako redom. Zaključujemo da je moguće odrediti vrednosti $y(x_0 + kh)$, $k \in \mathbb{N}$.

Označimo $x_k = x_0 + kh$, $y_k = y(x_k)$, $f_k = f(x_k, y_k)$, $k = 0, \ldots, n$, gde je n određeno iz uslova da $x_0 + nh$ ostaje u oblasti u kojoj je garantovana jedinstvenost rešenja Cauchyevog problema. Dobijamo sledeći metod za numeričku integraciju Cauchyevog problema

$$y_{k+1} = y_k + hf_k, \quad k = 0, \ldots, n-1, \tag{7.4}$$

koju nazivamo Eulerov metod.

Metod se obično implementira na sledeći način. Pretpostavimo da želimo da nađemo rešenje na intervalu $[a,b]$, gde je $a = x_0$, onda izaberemo korak h, tako da je ispunjen uslov $n = (b-a)/h$. Naravno, ovde podrazumevamo da unutar intervala $[a,b]$ postoji jedinstveno rešenje Cauchyevog problema. Zatim konstruišemo niz y_k, $k = 0, \ldots, n$.

Implementacija metoda je prilično jednostavna.

```
function [ x,y ] = eulerMethod( x0, y0, b, h, fun )
%EULERMETHOD(X0,Y0,B,H,FUN) konstruise resenje Cauchyevog
% problema y'=fun(x,y), y(x0)=y0, sa korakom h na intervalu
% [x0,b]

x = x0:h:b-h; y = [y0];
for t = x
    y = [y y(end)+h*fun(t,y(end))];
end
x = [x x(end)+h];
end
```

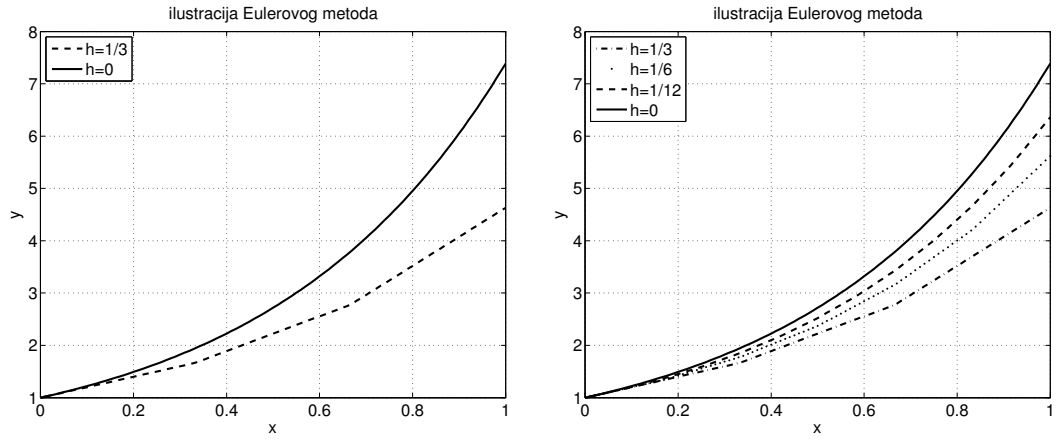
Slika 7.1 ilustruje primenu Eulerovog metoda na rešavanje Cauchyevog problema

$$y' = 2y, \quad y(0) = 1,$$

čije je rešenje $y(x) = e^{2x}$.

Slika 7.1, levo, daje ilustraciju primene Eulerovog metoda. Kao što vidimo greška se tokom konstrukcije niza y_k akumulira. Sa slike se vidi da greška raste kako se udaljavamo od početka intervala, tačke x_0. Ovo nije teško razumeti. Naime, prilikom izračunavanja y_1 mi pravimo grešku. Zatim, prilikom računanja y_2, koristimo pogrešnu vrednost y_1, što daje pogrešnu vrednost za f_1, što nas još više udaljava od tačnog rešenja, i tako redom prilikom konstrukcije svakog y_k. Na ovaj način konstruišemo niz y_k koji sve vreme akumulira grešku.

Intuitivno je jasno da greška Eulerovog metoda zavisi od koraka h, jer greška koju smo učinili pri odbacivanju ostatka Taylorovog polinoma je tim manja što je manja vrednost h. Slika 7.1, desno, ilustruje činjenicu da sa smanjenjem koraka h prilazimo sve bliže i bliže rešenju. Ono što



Slika 7.1: Slika levo ilustruje akumulaciju greške Eulerovog metoda, slika desno ilustruje konvergenciju Eulerovog metoda sa smanjenjem koraka. U oba slučaja $h = 0$ znači egzaktno rešenje.

nas trenutno zanima je kojom brzinom se približava kriva dobijena Eulerovim metodom pravom rešenju Cauchyevog problema u zavisnosti od promene koraka h. Pokazaćemo da je ta brzina linearna.

Posmatrajmo naš problem (7.2), i primenimo Eulerov metod na ovaj problem. Nalazimo da važi

$$y_{k+1} = y_k + hf_k = y_k + Ahy_k = (1 + Ah)y_k = \cdots = (1 + Ah)^{k+1}y_0.$$

Ako sad oduzmemo ovako dobijeno rešenje od tačnog rešenja, nalazimo

$$\begin{aligned} y(x_k) - y_k &= y_0 e^{A(x_k - x_0)} - y_0(1 + Ah)^k = y_0 \left(e^{kAh} - e^{k \log(1+Ah)} \right) \\ &= y_0 e^{kAh} \left(1 - e^{k(\log(1+Ah) - Ah)} \right) = y(x_k) \left(1 - e^{k(-(Ah)^2/2 + O(Ah)^3)} \right) \\ &= y(x_k) \left(1 - 1 + \frac{k}{2}(Ah)^2 + kO(Ah)^3 \right) = y(x_k) \left(\frac{x_k - x_0}{2} A^2 h + O((x_k - x_0)A^3 h^2) \right). \end{aligned}$$

Odavde možemo dobiti gornju granicu apsolutne greške

$$|y(x_k) - y_k| \leq |y_0| e^{A(x_k - x_0)} \left(\frac{x_k - x_0}{2} A^2 h + O(|(x_k - x_0)A^3 h^2|) \right) \approx |y(x_k)| \frac{x_k - x_0}{2} A^2 h,$$

i gornju granicu relativne greške

$$\frac{|y(x_k) - y_k|}{|y(x_k)|} \approx \frac{x_k - x_0}{2} A^2 h.$$

Kao što vidimo greška opada linearno sa korakom h. Zato kažemo da Eulerov metod ima linearnu konvergenciju ili da ima red jedan. Naravno, postoje metodi sa većim redom konvergencije. Ovi metodi će biti predmet izučavanja u narednim odeljcima. Primetimo da analiza greške važi jedino pod uslovom $|Ah| < 1$. Takođe, primetimo da za fiksnu vrednost koraka h greška zavisi od A. Što veća vrednost konstante A potreban je manji korak da bi se postigla ista apsolutna greška. Relativna greška u ovom slučaju menja se po zakonu A^2. Slika 7.2, desno, ilustruje ovu zavisnost. Konačno primetimo da za fiksno h i A greška raste linearno sa udaljavanjem od tačke x_0, što znači da broj značajnih cifara opada kao $-\log_{10}(x - x_0)$ kako se udaljavamo od tačke x_0. Ovaj izraz dobro opisuje gubitak značajnih cifara što se vidi na slici 7.2, desno. Vidimo da grafik zavisnosti broja značajnih cifara od x ima logaritamski izgled.

7.2.1 Implicitni Eulerov metod

Prilikom izvođenja Eulerovog metoda mogli smo postupiti i na sledeći način

$$y(x - h) = y(x) - y'(x)h + R_2(x - h; x, y) = y(x) - hf(x, y(x)) + R_2(x - h; x, y),$$

odakle dobijamo metod

$$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1}) = y_k + hf_{k+1}. \tag{7.5}$$

Primetimo da je ovaj metod komlikovaniji od pređašnjeg, jer kao što vidimo moramo rešavati nelinearnu jednačinu da bismo odredili y_{k+1}. Zato se ovaj metod naziva implicitni Eulerov metod, za razliku od pređašnjeg, koji se naziva eksplicitni Eulerov metod.

Postavlja se pitanje kako rešiti nelinearnu jednačinu (7.5). Sledeća teorema daje samo kao mogućnost korišćenje metoda proste iteracije. Ovaj način rešavanja nije najbolji, ali je za demonstracije i na ovom nivou studija metod koji nam najviše odgovara.

Teorema 7.2 *Neka funkcija f zadovoljava sledeći uslov*

$$\left|\frac{\partial f(x,y)}{\partial y}\right| \leq L$$

nad oblašću od interesa. Ako jednačina

$$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1})$$

ima rešenje, onda iterativni proces

$$y_{k+1}^{i+1} = y_k + hf(x_{k+1}, y_{k+1}^i), \quad i \in \mathbb{N}_0,$$

konvergira pod uslovom $hL < 1$, uz izbor dovoljno dobre startne vrednosti y_{k+1}^0.

Dokaz. Namera nam je da za dokaz iskoristimo teoremu 3.1. Definišimo funkciju

$$\phi(y_{k+1}) = y_k + hf(x_{k+1}, y_{k+1}).$$

Kako jednačina ima rešenje to postoji okolina rešenja, recimo interval $[a, b]$, gde funkcija ϕ zadovoljava uslov $\phi : [a, b] \to [a, b]$. Ostaje nam da pokažemo da je izvod funkcije ϕ manji od jedinice.

Nalazimo

$$\left|\frac{d\phi(y_{k+1})}{dy_{k+1}}\right| = h\left|\frac{\partial f(x_{k+1}, y_{k+1})}{\partial y_{k+1}}\right| \leq hL < 1,$$

odakle zaključujemo da je iterativni proces konvergentan, pod uslovom da možemo da obezbedimo $y_{k+1}^0 \in [a, b]$. □

Ako implicitni Eulerov metod primenimo na naš problem (7.2) nalazimo rešenje

$$y_k = \frac{y_0}{(1 - Ah)^k}.$$

Red konvergencije ovog metoda može se odrediti kao kod eksplicitnog metoda. Primenom istih metoda nalazimo

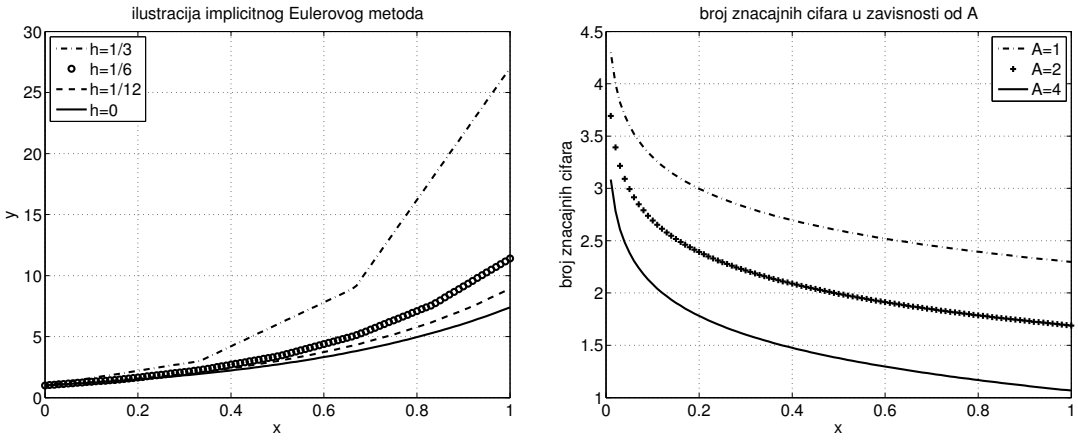
$$\begin{aligned} y(x_k) - y_k &= y_0 e^{kAh} - y_0(1 - Ah)^{-k} = y_0 e^{kAh}\left(1 - e^{-k(\log(1-Ah)+Ah)}\right) \\ &= y(x_k)\left(1 - e^{-k(-(Ah)^2/2+O(Ah)^3)}\right) = y(x_k)\left(1 - 1 - \frac{k}{2}(Ah)^2 + kO(Ah)^3\right) \\ &= y(x_k)\left(-\frac{x_k - x_0}{2}A^2h + O((x_k - x_0)A^3h^2)\right), \end{aligned}$$

a odavde dobijamo gornju granicu apsolutne i relativne greške

$$|y(x_k) - y_k| \approx |y(x_k)|\frac{x_k - x_0}{2}A^2h, \quad \frac{|y(x_k) - y_k|}{|y(x_k)|} \approx \frac{x_k - x_0}{2}A^2h.$$

Kao što vidimo red implicitnog Eulerovog metoda se ne razlikuje od reda eksplicitnog Eulerovog metoda. Međutim, primena implicitnog Eulerovog metoda zahteva rešavanje nelinearne jednačine, što predstavlja prilično veliku razliku u odnosu na eksplicitni Eulerov metod. Razlog za izučavanje implicitnog Eulerovog metoda, kao i implicitnih metoda uopšte, će biti objašnjen u narednom odeljku.

Implementacija implicitnog Eulerovog metoda je prilično jednostavna.



Slika 7.2: Slika levo ilustruje akumulaciju greške i konvergenciju implicitnog Eulerovog metoda, slika desno ilustruje gubitak značajnih cifara sa porastom A pri rešavanju Cauchyevog problema (7.2), za fiksnu vrednost koraka $h = .01$.

```
function [ x,y ] = eulerMethodImplicit( x0, y0, b, h, fun)
%EULERMETHODIMPLICIT(X0,Y0,B,H,FUN) implementira numericku
% integraciju Cauchyevog problema y'=fun(x,y), y(x0)=y0
% implicitnim Eulerovim metodom sa korakom h, do tacke b

x = x0+h:h:b; tt = x0; y = [y0];
for t = x
    ypom0 = Inf;
    ypom1 = y(end)+h*fun(tt,y(end));
    tt = t; iter = 0;
    while abs(ypom0-ypom1) > max(1e-10,1e-10*abs(ypom1))
        ypom0 = ypom1;
        ypom1 = y(end)+h*fun(t,ypom1);
        iter = iter+1;
        if iter > 100
            x = [x0 x];
            x = x(1:length(y));
            disp('previse iteracija');
            return;
        end
    end
    y = [y ypom1];
end
x = [x0 x];
end
```

U ovom slučaju je kao početna vrednost za rešavanje nelinearne jednačine uzeta vrednost koju vraća eksplicitni Eulerov metod. Ovakav način korišćenja implicitnih metoda se najčešće sreće u praksi.

Kao početna vrednost za rešavanje nelinearne jednačine u implicitnom metodu koristi se vrednost koju izračunava neki eksplicitni metod. Eksplicitni metod se u ovom slučaju naziva prediktor, dok se implicitni metod naziva korektor. Par metoda se naziva prediktor-korektor metod. Više o ovom načinu rešavanja jednačina se može naći u sekciji 7.3.6.

Slika 7.2, levo, ilustruje konvergenciju implicitnog Eulerovog metoda. Za konstrukciju grafika je korišćena prikazana funkcija a primenjena je na rešavanje Cauchyevog problema

$$y' = 2y, \quad y(0) = 1.$$

Kao što vidimo sa smanjenjem koraka h, dolazi do smanjenja greške što i očekujemo.

Slika 7.2, desno, ilustruje zavisnost greške implicitnog Eulerovog metoda u zavisnosti od parametra A, prilikom rešavanja problema (7.2). Kao što je pomenuto i u slučaju eksplicitnog Eulerovog metoda za fiksnu vrednost koraka h, relativna greška raste sa A^2. Na slici su prikazane značajne cifre u rešenju za Cauchyeve probleme (7.2) sa $A = 1$, $A = 2$ i $A = 4$. Broj značajnih cifara koje gubimo prilikom porasta parametra možemo eksplicitno da odredimo. Kako A raste dva puta to će gubitak značajnih cifara iznositi

$$-\log_{10}\left(\frac{|x_n - x_0|}{2}(2A)^2|h|\right) + \log_{10}\left(\frac{|x_n - x_0|}{2}A^2|h|\right) = -2\log_{10} 2 \approx .6$$

Prilikom povećanja veličine A dva puta, sa fiknim korakom, gubimo .6 značajnih cifara. Ovo ponašanje je jasno demonstrirano na grafiku.

7.2.2 Adaptivna integracija

Prethodni primer omogućava da se izvedu neki opšti zaključci. Ako posmatramo opšti Cauchyev problem (7.1), pod pretpostavkom da je funkcija f diferencijabilna po promenljivoj y, možemo za male vrednosti k pisati

$$f(x, y + k) \approx f(x, y) + \frac{\partial f(x, y)}{\partial y}k.$$

Lokalno, na malim intervalima, možemo svaki Cauchyev problem shvatiti kao Cauchyev problem zadat sa (7.2), gde je

$$A = \frac{\partial f(x, y)}{\partial y}.$$

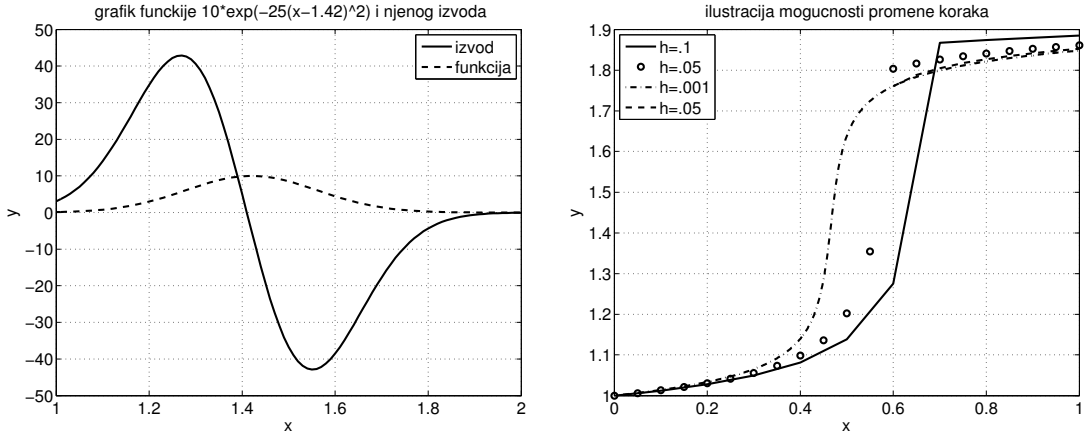
Očigledno je da ako dolazi do velikih varijacija u vrednosti parcijalnog izvoda po promenljivoj y funkcije f može doći do smanjenja broja značajnih cifara prilikom integracije unutar oblasti gde je vrednost parcijalnog izvoda najveća. Kako se greška prilikom integracije akumulira ovo smanjenje značajnih cifara dovešće do smanjenja broja značajnih cifara u rezultatu posle oblasti sa velikom vrednošću parcijalnog izvoda.

Iz ovog razloga, kad je to moguće, koristi se adaptivna integracija. Pod adaptivnom integracijom podrazumevamo promenu koraka u zavisnosti od vrednosti parcijalnog izvoda funkcije f po promenljivoj y.

Posmatrajmo sledeći Cauchyev problem

$$y' = 10\exp(-25(y - 1.42)^2), \quad y(0) = 1. \tag{7.6}$$

Da bismo shvatili zašto je baš izabran ovaj Cauchyev problem na slici 7.3, levo, je prikazan grafik funkcije $f(x, y) = 10\exp(-25(y - 1.42)^2)$ i njenog izvoda. Kao što vidimo funkcija f je pozitivna



Slika 7.3: Slika levo prikazuje grafik funkcije $10\exp(-25(x-1.41)^2)$ i njenog izvoda, slika desno prikazuje grafike dobijene primenom eksplicitne Eulerove metode na Cauchyev problem (7.6), sa koracima $h = .1$, $h = .05$, $h = .001$, dok je četvrti grafik sa korakom $h = .05$ ali sa metodom koji startuje od tačke $x_0 = .6$.

što znači da je izvod funkcije y pozitivan pa je funkcija y rastuća. Posledica ovoga je da vrednost funkcije y raste. Međutim, rast funkcije y od početne vrednosti dovodi integrator vrlo brzo u oblast gde je izvod funkcije f po promenljivoj y veliki. Ovu poslednju činjenicu vidimo sa slike. U ovoj oblasti nam je potreban mali korak da bismo zadržali tačnost prilikom integracije.

Veliki korak integracije u odnosu na vrednost izvoda funkcije f dovodi do velikih grešaka prilikom integracije. To se jasno vidi na slici 7.3, desno. Međutim, kao što vidimo, nije potrebno imati mali korak integracije sve vreme. Naime, počev od vrednosti .6 dovoljno je već ponovo početi integraciju sa korakom .05 da bi se dobili tačni rezultati.

Implicitni Eulerov metod, ako je korak preveliki, uglavnom kao rezultat daje prekoračenje dozvoljenog broja iteracija. Prevelika vrednost koraka znači da ni uslov konvergencije metoda proste iteracije u teoremi 7.2 nije zadovoljen. Naravno, kao što je napomenuto moguće je koristiti neki kvalitetniji metod za rešavanje nelinearne jednačine od metoda proste iteracije, u tom slučaju do ovog problema ne bi dolazilo.

Implementacija adaptivne metode uglavnom polazi od više integratora različitih koraka ili od numeričkog izračunavanja izvoda funkcije f. Prilikom svakog koraka upoređuju se vrednosti rezultata integratora sa većim koracima u odnosu na dobijene vrednosti integratora najmanjeg koraka. Ako se zadržala željena tačnost napredujemo ka sledećoj tački u podeli, ako nije smanjujemo korak i ponovo vršimo izračunavanja. Moguće je numerički određivati izvod funkcije f i na osnovu ove vrednosti menjati korak integracije. Pri ovakvoj implementaciji se uglavnom koriste već izračunate vrednosti funkcije f u prethodnim koracima.

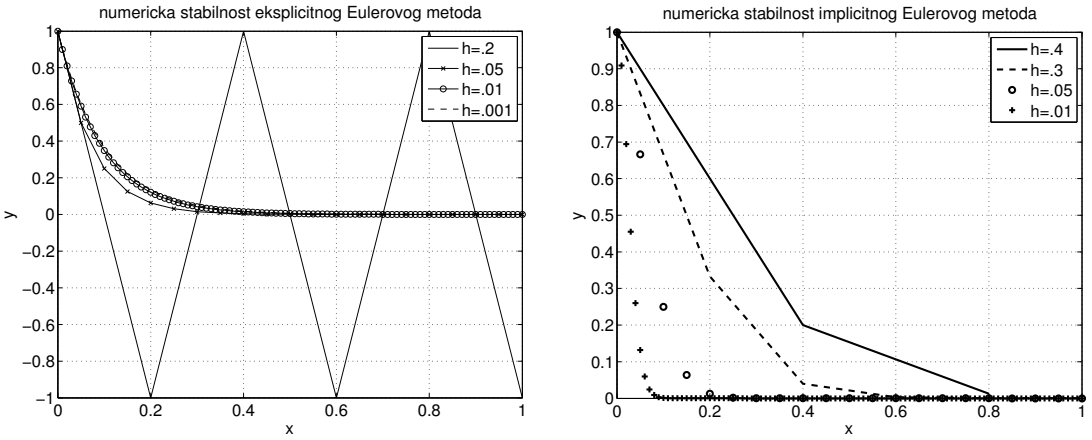
7.2.3 Apsolutna stabilnost

U ovom odeljku ćemo demonstrirati razlog zbog kojeg su implicitni metodi bolji od eksplicitnih. Pokazaćemo da je dodatni napor koji ulažemo prilikom rešavanja nelinearne jednačine isplativ.

Posmatrajmo Cauchyev problem

$$y' = -10y, \quad y(0) = 1.$$

Slika 7.4, levo, ilustruje rešenja eksplicitnim Eulerovim metodom, za razne vrednosti parametra h. Kao što vidimo za vrednost $h = .2$, dobijeno rešenje eksplicitnim Eulerovim metodom nema ništa zajedničko sa pravim rešenjem. Slika desno prikazuje rešenje istog Cauchyevog problema ali implicitnim Eulerovim metodom za razne vrednosti parametra. Za razliku od eksplicitnog Eulerovog metoda ovde ne dolazi do pojave rešenja koja se ponašaju potpuno proizvoljno u odnosu na stvarno rešenje. Da bismo objasnili dobijene slike posmatrajmo rešenja koja dobijamo upotrebom oba metoda.



Slika 7.4: Ilustracija apsolutne stabilnosti eksplicitnog Eulerovog metoda slika levo, i implicitnog Eulerovog metoda slika desno.

Posmatrajmo Cauchyev problem (7.2) uz uslov $A < 0$. Eksplicitni Eulerov metod ima rešenje $y_k = y_0(1 + Ah)^k$, dok implicitni Eulerov metod ima rešenje $y_k = y_0/(1 - Ah)^k$. Da bi se rešenje eksplicitnog Eulerovog metoda ponašalo kao pravo rešenje Cauchyevog problema, tj. da bi težilo nuli kad k raste, potrebna je ispunjenost uslova $|1+Ah| < 1$. Iz ovog sistema nejednakosti nalazimo $0 < h < 2/|A|$. Vidimo da postoji jasno određen interval, $(0, 2/|A|)$, u kome se mora nalaziti korak da bi eksplicitni Eulerov metod davao rešenja koja po smislu prate rešenje Cauchyevog problema. Ovaj interval zovemo intervalom apsolutne stabilnosti metoda. Sa druge strane, ako istu analizu sprovedemo za implicitni Eulerov metod nalazimo da mora biti ispunjena nejednakost $|1-Ah| > 1$. Iz ovog sistema nejednakosti nalazimo $Ah < 0$ ili $2 < Ah$. Druga nejednakost ne može biti ispunjena ali vidimo da je prva ispunjena za svako $h > 0$. Implicitni Eulerov metod ima interval apsolutne stabilnosti $(0, +\infty)$.

Interval apsolutne stabilnosti nosi naziv otuda što se može pokazati da u ovom intervalu ne dolazi do povećanja apsolutne greške rešenja. Naime, kod našeg Cauchyevog problema rešenje je takvo da opada sa x. Kako vidimo, kad izađemo iz intervala apsolutne stabilnosti za h dolazi do porasta apsolutne greške i u nekom trenutku naš metod više nema ništa zajedničko sa rešenjem, već predstavlja samo porast apsolutne greške.

Generalno, važi zaključak da implicitni metodi imaju značajno veći interval apsolutne stabilnosti od eksplicitnih metoda. Ova osobina posebno dolazi do izražaja ako prilikom integracije

dolazimo do oblasti gde je izvod funkcije f veliki. Eksplicitni metodi u ovoj oblasti, ako ne radimo adaptivnu promenu koraka, uglavnom nemaju nikakve veze sa rešenjem, pod uslovom da je korak h dovoljno veliki da smo izašli iz intervala apsolutne stabilnosti. Za razliku od njih implicitni metodi će zadržati vezu sa smislom ponašanja rešenja, jer je interval apsolutne stabilnosti veći.

Pomenimo da slika 7.4, desno, za velike vrednosti koraka nije dobijena upotrebom funkcije koja je prezentovana u knjizi. Razlog za ovo je što metod proste iteracije za velike vrednosti koraka ne bi konvergirao. Međutim, pri konstrukciji ozbiljnijeg softwarea koristili bismo ozbiljnije metode za rešavanje nelinearne jednačine, pa do ovog problema ne bi dolazilo.

Ovim smo praktično iscrpli demonstracione mogućnosti Eulerovog metoda. Međutim, Eulerov metod nam je poslužio da prezentujemo sve važne koncepte koji se sreću i kod metoda višeg reda.

7.3 Linearni višekoračni metodi

Eulerovi metodi su primeri najjednostavnijih linearnih višekoračnih metoda. U najopštijoj formi linearni višekoračni metod za rešavanje Cauchyevog problema (7.1), izgleda ovako

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i}, \tag{7.7}$$

gde koristimo oznake $y_k = y(x_k)$ i $f_k = f(x_k, y(x_k))$. Podrazumevamo da je $\alpha_k \neq 0$, jer u suprotnom metod ne bi imao mnogo smisla, jer jednačina ne bi sadržala eksplicitno vrednost y_{n+k}. Kako se jednačina kojom je određen metod ne menja ako je pomnožimo proizvoljnim brojem, dogovorom normalizujemo jednačinu tako da važi $\alpha_k = 1$.

Na primer, kod Eulerovih metoda je $k = 1$ i $\alpha_0 = -1$, $\alpha_1 = 1$. Kod eksplicitnog metoda je $\beta_1 = 0$ i $\beta_0 = 1$, dok je kod implicitnog $\beta_1 = 1$ i $\beta_0 = 0$.

Sve metode delimo u dve velike grupe. Eksplicitni metodi kod kojih je $\beta_k = 0$, i implicitni metodi kod kojih je $\beta_k \neq 0$.

7.3.1 Red metoda

Kompleksnost izraza kojim su zadati linearni višekoračni metodi omogućava povećanje reda metoda. Naime, kod Eulerovih metoda je $k = 1$ i metodi imaju red jedan. Od generalnog linearnog višekoračnog metoda očekujemo da sa porastom k raste red metoda. Drugim rečima, koeficijenti α_i, β_i, $i = 0, \ldots, k$, ne mogu biti proizvoljni nego se biraju tako da obezbede najveći mogući red metoda, s obzirom na druga ograničenja koja pominjemo dalje u tekstu.

Posmatrajmo izraz

$$L(y) = \sum_{i=0}^{k} (\alpha_i y(x + ih) - h\beta_i y'(x + ih))$$

koji je dobijen iz izraza (7.7), gde smo vrednost funkcije f zamenili izvodom funkcije y. Ako u izrazu $L(y)$ razvijemo u Taylorov red u tački x sve funkcije koje učestvuju u izrazu $L(y)$, dobićemo razvoj

$$L(y) = C_0 y(x) + hC_1 y'(x) + h^2 C_2 y''(x) + \cdots .$$

Sad je samo potrebno što je moguće više konstanti C_i, $i = 0, \ldots, p$, postaviti na vrednost nula koristeći koeficijente α_i, β_i, $i = 0, \ldots, k$, da bi red metoda bio što je moguće veći. Ako ispunimo uslov $C_i = 0$, $i = 0, \ldots, p$, i $C_{p+1} \neq 0$, dobijamo metod reda p.

Na primer, ako posmatramo eksplicitni Eulerov metod nalazimo

$$L(y) = y(x+h) - y(x) - hy'(x) = \sum_{i=0}^{+\infty} \frac{y^{(i)}(x)}{i!} h^i - y(x) - hy'(x) = \frac{1}{2}h^2 y^{(2)}(x) + \frac{1}{6}h^3 y^{(3)}(x) + \cdots .$$

Očigledno je $C_0 = C_1 = 0$, dok je $C_2 \neq 0$. Dobijamo zaključak koji smo već dobili: Eulerov metod je reda 1. Sličan rezultat bismo dobili za implicitni Eulerov metod.

Sa druge strane posmatrajmo metod

$$y_{n+2} - y_n = \frac{h}{3}(f_{n+2} + 4f_{n+1} + f_n). \tag{7.8}$$

Primetimo da je na desnoj strani Simpsonova formula za integraciju, pa se metod naziva Simpsonovim višekoračnim metodom. Ako sprovedemo pomenutu analizu nalazimo

$$\begin{aligned} L(y) &= y(x+2h) - y(x) - \frac{h}{3}y'(x+2h) - \frac{4h}{3}y'(x+h) - \frac{h}{3}y'(x) \\ &= \sum_{i=0}^{+\infty} \frac{y^{(i)}(x)}{i!}(2h)^i - y(x) - \frac{h}{3}\sum_{i=0}^{+\infty} \frac{y^{(i+1)}(x)}{i!}(2h)^i - \frac{4h}{3}\sum_{i=0}^{+\infty} \frac{y^{(i+1)}(x)}{i!}h^i - \frac{h}{3}y'(x) \\ &= (1-1)y(x) + h\left(2 - \frac{1}{3} - \frac{4}{3} - \frac{1}{3}\right)y'(x) + h^2\left(2 - \frac{2}{3} - \frac{4}{3}\right)y''(x) + h^3\left(\frac{4}{3} - \frac{2}{3} - \frac{2}{3}\right)y^{(3)}(x) \\ &+ h^4\left(\frac{2}{3} - \frac{4}{9} - \frac{2}{9}\right)y^{(4)}(x) + h^5\left(\frac{4}{15} - \frac{2}{9} - \frac{1}{18}\right)y^{(5)}(x) + \cdots = -\frac{h^5}{90}y^{(5)}(x) + \cdots . \end{aligned}$$

Vidimo da je Simpsonov višekoračni metod reda četiri.

Definicija 7.1 *Linearni višekoračni metod je reda p ako i samo važi Taylorov razvoj*

$$L(y) = \sum_{i=0}^{k}(\alpha_i y(x+ih) - h\beta_i y'(x+ih)) = \sum_{i=p+1}^{+\infty} h^i C_i y^{(i)}(x) = h^{p+1}C_{p+1}y^{(p+1)}(x) + \cdots ,$$

gde je $C_{p+1} \neq 0$.

Konstantu $C_{p+1} \neq 0$ metoda reda p nazivamo asimptotska konstanta greške.

Koeficijente C_i je moguće eksplicitno izraziti preko koeficijenata α_i, β_i, $i = 0, \ldots, k$, i to u sledećem obliku

$$\begin{aligned} C_0 &= \alpha_0 + \alpha_1 + \cdots + \alpha_k, \\ C_1 &= (\alpha_1 + 2\alpha_2 + \cdots + k\alpha_k) - (\beta_0 + \beta_1 + \cdots + \beta_k) = 0, \\ C_i &= \frac{1}{i!}(\alpha_1 + 2^i\alpha_2 + \cdots + k^i\alpha_k) - \frac{1}{(i-1)!}(\beta_1 + 2^{i-1}\beta_2 + \cdots + k^{i-1}\beta_k), \quad i \in \mathbb{N} \setminus \{1\}. \end{aligned} \tag{7.9}$$

Izborom k i željenog reda p dobijamo sistem linearnih jednačina koji treba da zadovolje koeficijenti linearnog višekoračnog metoda.

Vrednost $L(y)$ je u literaturi poznata kao lokalna greška odsecanja. Ime dolazi iz sledećeg razmatranja. Pretpostavimo da važi $y_{n+i} = y(x_{n+i})$, $i = 0, \ldots, k-1$. Ova pretpostavka se naziva

lokalna pretpostavka,.Greška vrednosti y_{n+k} je zadata sa

$$\begin{aligned} e_{n+k} = y(x_{n+k}) - y_{n+k} &= \sum_{i=0}^{k-1}(-\alpha_i y(x_{n+i}) + h\beta_i f(x_{n+i}, y(x_{n+i})) - \sum_{i=0}^{k-1}(-\alpha_i y_{n+i} + h\beta_i f_{n+i}) \\ &+ h\beta_k f(x_{n+k}, y(x_{n+k})) - h\beta_k f_{n+k} + L(y)(x_{n+k}) \\ &= L(y)(x_{n+k}) + h\beta_k(f(x_{n+k}, y(x_{n+k})) - f(x_{n+k}, y_{n+k})). \end{aligned}$$

Ako je metod eksplicitni, onda je $\beta_k = 0$, pa dobijamo $e_{n+k} = L(y)(x_{n+k})$, tako da izraz stvarno predstavlja lokalnu grešku odsecanja. Ako je metod implicitni, pod pretpostavkom postojanja parcijalnog izvoda funkcije f i korišćenjem Lagrangeove teoreme, nalazimo

$$e_{n+k} = L(y)(x_{n+k}) + h\beta_k \frac{\partial f(x_{n+k}, \psi_{n+k})}{\partial y}(y(x_{n+k}) - y_{n+k}) = L(y)(x_{n+k}) + h\beta_k \frac{\partial f(x_{n+k}, \psi_{n+k})}{\partial y} e_{n+k},$$

$$e_{n+k} = \frac{L(y)(x_{n+k})}{1 - h\beta_k \frac{\partial f(x_{n+k}, \psi_{n+k})}{\partial y}},$$

gde je $\psi_{n+k} \in (\min\{y_{n+k}, y(x_{n+k})\}, \max\{y_{n+k}, y(x_{n+k})\})$. Vidimo da je i u ovom slučaju greška proporcionalna lokalnoj grešci odsecanja, pod uslovom

$$h\beta_k \frac{\partial f(x_{n+k}, \psi_{n+k})}{\partial y} \neq 1.$$

Ovaj je uslov obično ispunjen ako smo u obasti gde metod ima veći broj značajnih cifara. Posebno, u primenama prediktor-korektor metoda mora biti ispunjen znatno oštriji uslov

$$\left| h\beta_k \frac{\partial f(x_{n+k}, \psi_{n+k})}{\partial y} \right| < 1,$$

ako želimo konvergenciju metoda, što je ilustrovano u teoremi 7.7.

7.3.2 Konvergencija

Da bi linearni višekoračni metod bio konvergentan, drugim rečima, da bi sa smanjenjem koraka h rešenje bilo sve bliže stvarnom rešenju diferencijalne jednačine moramo zahtevati da red metoda bude najmanje jedan. Metodi reda nula nemaju smisla, jer generalno ne konvergiraju.

Definicija 7.2 *Linearni višekoračni metod je konvergentan na intervalu $[x_0, b]$ ako i samo ako za svako $x_n \in [x_0, b]$ važi*

$$\lim_{h \to 0,\, x_n - x_0 = nh} y(x_n) - y_n = 0.$$

Primetimo da je u definiciji tačka x_n fiksirana kad h teži nuli, pa n mora neograničeno da raste.

Definicija 7.3 *Za linearni višekoračni metod kažemo da je konzistentan ako i samo ako ima red veći ili jednak jedan.*

Za opisivanje osobina linearnih višekoračnih metoda potrebni su nam prvi i drugi karakteristični polinom.

Definicija 7.4 *Prvi karakteristični polinom linearnog višekoračnog metoda* (7.7) *je polinom*

$$\rho(\psi) = \sum_{i=0}^{k} \alpha_i \psi^i,$$

dok je drugi karakteristični polinom određen sa

$$\sigma(\psi) = \sum_{i=0}^{k} \beta_i \psi^i.$$

Primetimo da važi $C_0 = \rho(1)$ i $C_1 = \rho'(1) - \sigma(1)$. Uslov konzistentnosti linearnog višekoračnog metoda se može izraziti na sledeći način

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1).$$

Konzistentnost nije dovoljna za konvergenciju linearnog višekoračnog metoda. Da bi metod bio konvergentan potrebno je i da je nula stabilan.

Definicija 7.5 *Linearni višekoračni metod je nula stabilan ako i samo ako sve nule prvog karakterističnog polinoma imaju moduo manji ili jednak jedan, pri čemu su nule sa modulom jednakim jedan proste.*

Na primer, Simpsonov višekoračni metod je nula stabilan. Kod ovog metoda je $\rho(\psi) = \psi^2 - 1$, pa su nule ± 1. Očigledno nule prvog karakterističnog polinoma imaju moduo manji ili jednak jedan i nule u tačkama $+1$ i -1, sa modulom jednakim jedan, su proste.

Sledeća teorema daje potrebne i dovoljne uslove za konvergenciju linearnih višekoračnih metoda.

Teorema 7.3 *Linearni višekoračni metod je konvergentan ako i samo ako je konzistentan i nula stabilan.*

Primer metoda koji nije nula stabilan, a jeste konzistentan, je metod

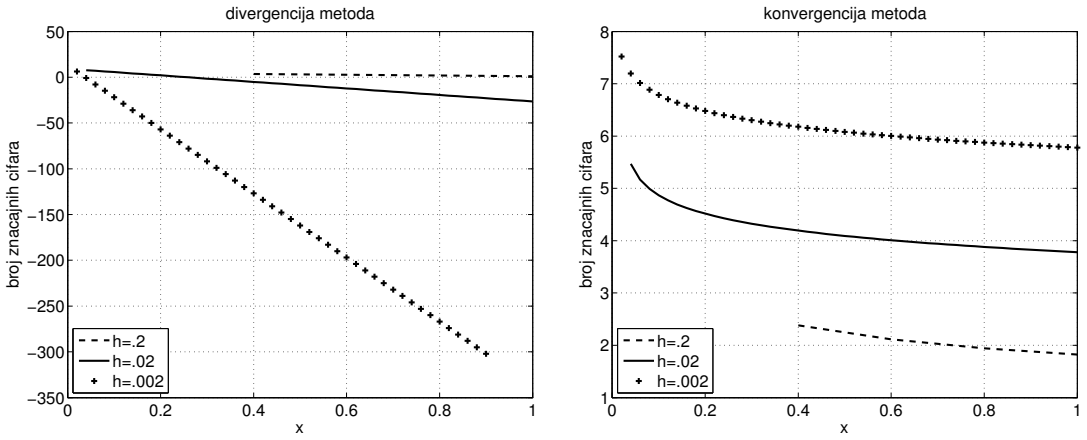
$$y_{n+2} + 4y_{n+1} - 5y_n = 2h(2f_{n+1} + f_n). \tag{7.10}$$

Ovde je $\rho(\psi) = \psi^2 + 4\psi - 5$ i $\sigma(\psi) = 4\psi + 2$. Očigledno je $\rho(1) = 0$ i $\rho'(1) = 6 = 6 = \sigma(1)$. Metod jeste konzistentan. Može se pokazati da metod ima red $p = 3$. Međutim, metod nije nula stabilan. Naime, nule prvog karakterističnog polinoma su -5 i 1. Primena ovog metoda dovodi do grešaka.

Slika 7.5, levo, prikazuje zavisnost broja značajnih cifara u rešenju sa smanjenjem koraka h. Kao što vidimo umesto da taj broj raste sa smanjenjem koraka, dešava se upravo suprotno, sa smanjenjem koraka h broj značajnih cifara u rešenju opada. Ovo je posledica divergencije metoda. Metod je primenjen na rešavanje Cauchyevog problema $y' = -y$, $y(0) = 1$.

Primetimo da je jedan važan problem prilikom konstrukcije linearnih višekoračnih metoda i određivanje startnih vrednosti. Na primeru Simpsonovog metoda možemo videti da pored vrednosti y_0, da bismo uspešno pokrenuli metod, moramo imati i vrednost y_1. Generalno, za opšti linearni višekoračni metod (7.7) za uspešnu primenu metoda moramo imati vrednosti y_i, $i = 0, \ldots, k-1$. Ovo predstavlja poseban problem u konstrukciji softwarea za linearne višekoračne metode i ovaj problem se rešava primenom različitih metoda. Konkretno, u konstrukciji primera na slici 7.5, levo, mi smo koristili egzaktne vrednosti, jer je jednostavno odrediti egzaktno rešenje.

Jedna od metoda koji može da se koristi pri konstrukciji startnih vrednosti je primena Taylorovog metoda. Naime, pod pretpostavkom da je funkcija f dovoljan broj puta diferencijabilna



Slika 7.5: Slika desno ilustruje divergenciju metoda (7.10), slika desno ilustruje konvergenciju linearnog višekoračnog metoda drugog reda.

moguće je odrediti izvode višeg reda funkcije y. Na osnovu vrednosti ovih izvoda moguće je konstruisati Taylorov polinom pomoću kojeg je moguće odrediti nedostajuće startne vrednosti za linearne višekoračne metode. Ovaj način određivanja startnih vrednosti nije jedini i daleko je od najboljeg, jer zahteva diferenciranje. Videćemo kasnije u ovoj glavi da možemo koristiti i metode Runge-Kutta za konstrukciju nedostajućih startnih vrednosti.

Slika 7.5, desno, ilustruje konvergenciju metoda drugog reda

$$y_{n+2} - y_{n+1} = \frac{h}{2}(3f_{n+1} - f_n), \tag{7.11}$$

primenjenog na isti Cauchyev problem $y'(x) = -y$, $y(0) = 1$. Na ovoj slici je prikazan broj značajnih cifara u svakom koraku. Kao što vidimo sa smanjenjem koraka broj značajnih cifara se povećava. Kod ovog metoda je $\rho(\psi) = \psi^2 - \psi$ i $\sigma(\psi) = 3/2\psi - 1/2$. Vidimo da je metod konzistentan, jer je $\rho(1) = 0$ i $\rho'(1) = 1 = 1 = \sigma(1)$. Takođe, nule polinoma ρ su 0 i 1, vidimo da su sve po modulu manje ili jednake 1, i nula koja ima moduo jednak jedan ima višestrukost jedan. Dakle, metod je konvergentan. Primetimo da sa slike vidimo da broj značajnih cifara koje dobijemo kad korak smanjimo 10 puta iznosi približno 2. Pokazaćemo, kad se bavimo analizom gornje granice greške linearnih višekoračnih metoda, da ovaj porast značajnih cifara ne iznosi slučajno dva.

U praksi se najčešće koriste familije metoda sa prvim karakterističnim polinomom oblika

$$\rho(\psi) = \psi^k - \psi^{k-1},$$
$$\rho(\psi) = \psi^k - \psi^{k-2}.$$

Eksplicitni metodi prve grupe se nazivaju Adams-Bashfortovi metodi, dok se implicitni nazivaju Adams-Moultonovi. Eksplicitni metodi druge grupe se nazivaju Nystromovi metodi, dok se implicitni nazivaju Milne-Simpsonovi.

Mi smo se sretali sa ovim metodima. Na primer, eksplicitni Eulerov metod i metod dat u jednačini (7.11) spadaju u grupu Adams-Bashfortovih metoda. Simpsonov metod (7.8) spada u grupu Milne-Simposonovih metoda. Tabela 7.1 sadrži pregled ovih familija metoda za $k \leq 4$ čija

se konstrukcija može izvršiti upotrebom kvadraturnih formula. Neki autori ubrajaju i implicitni Eulerov metod u familiju Adams-Moultonovih metoda. Naravno, razlog je izgled prvog karakterističnog polinoma. Međutim, implicitni Eulerov metod ne pripada u klasičnom smislu Adams-Moultonovoj familiji metoda. Implicitni Eulerov metod pripada familiji metoda sa diferenciranjem unazad. O ovoj familiji metoda detaljnije govorimo u sekciji 7.3.8.

Konstrukcija metoda pomenutih familija je jednostavna. Prvo fiksiramo k u jednačini metoda (7.7). Kako su koeficijenti metoda α_i, $i = 0, \ldots, k$, određeni kad se odlučimo za neku familiju metoda, ostaje nam da iz sistema jednačina (7.9) odredimo koeficijente β_i, $i = 0, \ldots, k$, i da pri tome postignemo što je moguće veći red metoda. Postoji i metod konstrukcije zasnovan na kvadraturnim formulama, izložen u sekciji 7.3.3, koji je bliži intuiciji.

Važna teorema koja može pomoći prilikom određivanja reda metoda je sledeća.

Teorema 7.4 *Neka je dat konvergentan linearni višekoračni metod* (7.7)*. Ako je k neparno metod može imati najviše red $k+1$, ako je k parno metod može imati najviše red $k+2$.*

Na primer, Simpsonov metod (7.8) ima najveći mogući red $p = 4$, s obzirom na vrednost $k = 2$.

Ovakvi posebno efikasni metodi nazivaju se optimalni. Može se pokazati da kod optimalnih metoda prvi karakteristični polinom ρ ima sve nule na jediničnom krugu. Familija Adamsovih metoda ne može biti optimalna za $k \geq 2$, dok familije Nystromovih ili Milne-Simpsonovih metoda ne mogu biti optimalne pod uslovom $k \geq 3$. Vidimo da je Simpsonov metod optimalan na osnovu vrednosti u tabeli 7.1.

Primetimo da teorema 7.4 ne znači da je nemoguće konstruisati metod većeg reda od $p = k+2$. Naime, moguće je za zadato k konstruisati eksplicitne metode koji imaju red $p = 2k-1$, ili implicitne metode koji imaju red $p = 2k$, koristeći sistem linearnih jednačina (7.9). Međutim, na osnovu tvrđenja teoreme ovi metodi neće biti konvergentni, što na osnovu teoreme 7.3 znači da neće biti nula stabilni. Naime, prvi karakteristični polinom ρ metoda koji imaju preveliki red će imati nule koje ne pripadaju jediničnom krugu.

7.3.3 Konstrukcija upotrebom kvadraturnih formula

Kao što smo videli u prikazanim primerima postoji veza između kvadraturnih formula i linearnih višekoračnih metoda. Ovu vezu je relativno jednostavno shvatiti. Diferencijalnu jednačinu $y' = f(x, y)$ je uvek moguće prikazati u formi integralne jednačine

$$y(x) - y(x - jh) = \int_{x-jh}^{x} f(x, y(x))dx.$$

Ako se ograničimo na tačke podele $x = x_n$, prethodnu jednakost možemo prikazati u sledećoj formi

$$y_{n+k} - y_{n+k-j} = \int_{x_{n+k-j}}^{x_{n+k}} f(x, y(x))dx.$$

Sve što ostaje je da zamenimo izraz za funkciju $g(x) = f(x, y(x))$ njenim interpolacionim polinomom u tačkama $x_n, \ldots, x_{n+k-1}$ za eksplicitne metode ili $x_n, \ldots, x_{n+k}$ za implicitne metode. Koristeći Lagrangeov bazis, konstruisan u pomenutim čvorovima, dobijamo eksplicitne metode

$$y_{n+k} - y_{n+k-j} = \sum_{i=0}^{k-1} f(x_{n+i}, y(x_{n+i})) \int_{x_{n+k-j}}^{x_{n+k}} L_i(x)dx = h \sum_{i=0}^{k-1} \beta_i f_{n+i}, \tag{7.12}$$

$$\beta_i = \frac{1}{h} \int_{x_{x+k-j}}^{x_{n+k}} L_i(x)dx = \frac{(-1)^{k-i-1}}{(k-1)!} \begin{pmatrix} k-1 \\ i \end{pmatrix} \int_{k-j}^{k} \frac{p^{(k)}}{p-i} dp, \quad i = 0, \ldots, k-1,$$

i odgovarajuće implicitne metode

$$y_{n+k} - y_{n+k-j} = \sum_{i=0}^{k} f(x_{n+i}, y(x_{n+i})) \int_{x_{n+k-j}}^{x_{n+k}} L_i(x)dx = h \sum_{i=0}^{k} \beta_i f_{n+i}, \tag{7.13}$$

$$\beta_i = \frac{1}{h} \int_{x_{x+k-j}}^{x_{n+k}} L_i(x)dx = \frac{(-1)^{k-i}}{k!} \begin{pmatrix} k \\ i \end{pmatrix} \int_{k-j}^{k} \frac{p^{(k+1)}}{p-i} dp, \quad i = 0, \ldots, k.$$

Ako izaberemo $j = 1$, vidimo da metodi imaju prvi karakteristični polinom upravo kao i familija Adamsovih metoda. U stvari konstrukcija familije Adamsovih metoda se upravo i izvodi na ovaj način. Ako izaberemo $j = 2$ prvi karakteristični polinom postaje kao kod familije Nystromovih ili Milne-Simpsonovih metoda. I ovi metodi se konstruišu na ovaj način. Da bismo pokazali da metodi imaju odgovarajući red možemo koristiti teoremu 4.5.

	Adams-Bashfort				Adams-Moulton			
k	1	2	3	4	1	2	3	4
β_0	1	$-1/2$	$5/12$	$-3/8$	$1/2$	$-1/12$	$1/24$	$-19/720$
β_1		$3/2$	$-4/3$	$37/24$	$1/2$	$2/3$	$-5/24$	$53/360$
β_2			$23/12$	$-59/24$		$5/12$	$19/24$	$-11/30$
β_3				$55/24$			$3/8$	$323/360$
β_4								$251/720$
p	1	2	3	4	2	3	4	5
C_{p+1}	$1/2$	$5/12$	$3/8$	$251/720$	$-1/12$	$-1/24$	$-19/720$	$-3/160$
γ	-2	-1	$-6/11$	$-3/10$	$-\infty$	-6	-3	$-90/49$
	Nystrom				Milne-Simpson			
k	1	2	3	4	1	2	3	4
β_0		0	$1/3$	$-1/3$		$1/3$	0	$-1/90$
β_1		2	$-2/3$	$4/3$		$4/3$	$1/3$	$2/45$
β_2			$7/3$	$-5/3$		$1/3$	$4/3$	$4/15$
β_3				$8/3$			$1/3$	$62/45$
β_4								$29/90$
p		2	3	4		4	4	5
C_{p+1}		$1/3$	$1/3$	$29/90$		$-1/90$	$-1/90$	$-1/90$
γ		-	-	-		-	-	-

Tabela 7.1: Adamsova, Nystromova i Milne-Simpsonova familija metoda, koeficijenti linearnog višekoračnog metoda β_i, $i = 0, \ldots, k$, red metoda p, asimptotska konstanta greške C_{p+1} i donja granica intervala apsolutne stabilnosti γ.

Teorema 7.5 *Metodi* (7.12) *imaju red konvergencije najmanje k, dok metodi* (7.13) *imaju red konvergencije najmanje $k + 1$.*

Dokaz. Ograničimo se na eksplicitne metode. Prilikom interpolacije funkcije $g(x) = f(x, y(x))$ mi pravimo grešku koja se može opisati sa

$$|g(x) - P_{k-1}(x)| \leq \frac{h^k}{4k} M_k, \quad M_k = \max_{x \in [x_{n+k-j}, x_{n+k}]} |g^{(k)}(x)|.$$

Za lokalnu grešku odsecanja nalazimo

$$|L(y)| = \left| y_{n+k} - y_{n+k-j} - \int_{x_{n+k-j}}^{x_{n+k}} P_{k-1}(x)dx \right| = \left| \int_{x_{n+k-j}}^{x_{n+k}} f(x, y(x))dx - \int_{x_{n+k-j}}^{x_{n+k}} P_{k-1}(x)dx \right|$$
$$\leq \int_{x_{n+k-j}}^{x_{n+k}} |f(x, y(x))dx - P_{k-1}(x)|\, dx \leq \frac{jh^k}{4k} M_k \int_{x_{n+k-1}}^{x_{n+k}} dx = \frac{jh^{k+1}}{4k} M_k.$$

Ovde jasno vidimo da je metod reda najmanje k.

Jedina razlika kod implicitnih metoda je da interpolacioni polinom P_k ima jedan dodatni interpolacioni čvor. U ovom smislu je stepen h u izrazu za grešku h^{k+1}, dok je izraz za granicu greške onda dat sa

$$\frac{jh^{k+2}}{4(k+1)} M_{k+1}, \quad M_{k+1} = \max_{x \in (x_{n+k-j}, x_{n+k})} |g^{(k+1)}(x)|.$$

I ovde se jasno vidi da je red metoda najmanje $k+1$. Ovim je dokaz teoreme završen. □

Jednostavno zaključujemo da je familija Adamsovih metoda konzistentna, kako je još i nula stabilna jer je $\rho(\psi) = \psi^k - \psi^{k-1}$, pomenuta familija metoda je konvergentna na osnovu teoreme 7.3 . Na osnovu teoreme 7.4 možemo zaključiti i da familija Adamsovih metoda ne može imati red konvergencije bitno veći od reda pomenutog u tvrđenju teoreme. Slična razmatranja važe i za Nystromove i Milne-Sympsonove metode koji se dobijaju za $j = 2$.

Tabela 7.1 pokazuje da u slučaju Simpsonovog metoda red metoda iznosi $k+2$, što u suštini znači da se u teoremi 7.5 ne može tvrditi da je red metoda baš jednak k ili $k+1$.

7.3.4 Analiza greške

Analiza gornje granice greške koju proizvode linearni višekoračni metodi je prilično komplikovana. Mi se nećemo detaljnije baviti ovim problemom, ali ćemo dati izraz koji predstavlja neki način na koji se može shvatiti kako se prostire greška kod linearnih višekoračnih metoda. Pod uslovom da posmatramo konvergentan eksplicitni metod čiji koeficijenti zadovoljavaju uslov

$$\alpha_i \leq 0, \quad i = 0, \ldots, k-1,$$

na primer Adams-Bashfortovi ili Nystromovi metodi, grešku u n-tom koraku možemo izraziti na sledeći način

$$|y(x_n) - y_n| \leq (\delta + (x_n - x_0)h^p GY)\, e^{LB(x_n - x_0)},$$

gde je $B = \sum_{i=0}^{k-1} |\beta_i|$, L je Lipschitzova konstanta funkcije f, dok su konstante G i Y određene na izvestan način (videti [8]), a $\delta = \max\{|y_0 - y(x_0)|, \ldots, |y_{k-1} - y(x_{k-1})|\}$. Ono što je nama bitno je ponašanje greške u funkciji h. Kao što vidimo greška se ponaša kao h^p, gde je p red metoda.

Vratimo se primeru datom na slici 7.5. Ovde je prikazana zavisnost broja značajnih cifara u funkciji h kod metoda drugog reda $p = 2$. Kao što je konstatovano, ako h smanjimo 10 puta dobijemo približno dve značajne cifre. Ovo je sada jednostavno obrazložiti, jer je očigledno

$$-\log_{10} \left(\frac{1}{10} \right)^2 = 2.$$

Ovaj izraz za grešku treba shvatiti uslovno. Naime, pri izvođenju nisu uključeni efekti konačne dužine mantise. Međutim, zaključak je ipak dovoljno dobar da bi se shvatilo da red metoda određuje način na koji se greška ponaša.

7.3.5 Stabilnost

Sad smo u mogućnosti da se detaljnije pozabavimo i pojmom apsolutne stabilnosti linearnih višekoračnih metoda. Kao i kod Eulerovih metoda, interval apsolutne stabilnosti je onaj interval (γ, δ) koji ima osobinu da ako je $Ah \in (\gamma, \delta)$, onda dolazi do opadanja apsolutne greške prilikom integracije.

Pretpostavimo da posmatramo Cauchyev problem (7.2). Onda važi

$$L(y)(x_{n+k}) = \sum_{i=0}^{k}(\alpha_i y(x_{n+i}) - h\beta_i y'(x_{n+i})) = \sum_{i=0}^{k}(\alpha_i - Ah\beta_i)y(x_{n+i}),$$
$$0 = \sum_{i=0}^{k}(\alpha_i y_{n+i} - h\beta_i f_{n+i}) = \sum_{i=0}^{k}(\alpha_i - Ah\beta_i)y_{n+i}.$$

Pretpostavimo da niz y_n izračunavamo koristeći aritmetiku mantise konačne dužine. U ovom slučaju možemo pisati

$$T_{n+k} = \sum_{i=0}^{k}(\alpha_i - Ah\beta_i)y_{n+i},$$

jer je niz y_n određen sa greškom koju modeluje veličina T_n.

Oduzimanjem ovih jednakosti nalazimo

$$L(y)(x_{n+k}) - T_{n+k} = \sum_{i=0}^{k}(\alpha_i - Ah\beta_i)(y(x_{n+i}) - y_{n+i}) = \sum_{i=0}^{k}(\alpha_i - Ah\beta_i)e_{n+i},$$

gde smo označili $e_n = y(x_n) - y_n$, $n \in \mathbb{N}_0$. Pod pretpostavkom da je broj značajnih cifara u rešenju relativno veliki možemo reći da je veličina $L(y(x_{n+k})) - T_{n+k}$ relativno mala. Zbog ove činjenice, u oblasti gde je broj značajnih cifara relativno veliki, dobru aproksimaciju možemo dobiti ako pretpostavimo da je $L(y(x_{n+k})) - T_{n+k} = C$, dakle nezavisno od n. U ovom slučaju diferencna jednačina koja određuje grešku dobija formu

$$\sum_{i=0}^{k}(\alpha_i - Ah\beta_i)e_{n+i} = C. \tag{7.14}$$

Rešenje ove linearne diferencne jednačine se dobija jednostavno. Uvedimo polinom $\pi(\psi; Ah) = \rho(\psi) - Ah\sigma(\psi)$ koji je karakteristični polinom naše diferencne jednačine. Ovde polinom π shvatamo kao polinom po ψ sa parametrom Ah. Pretpostavimo da su sve nule r_i, $i = 0, \ldots, k$, polinoma π međusobno različite, onda diferencna jednačina ima rešenje

$$e_n = \sum_{i=0}^{k} C_i r_i^n + \frac{C}{\pi(1; Ah)}.$$

Ako želimo da greška ne raste, onda mora biti ispunjen uslov $|r_i| < 1$. Ništa se bitno ne menja i ako dopustimo da je neka od nula polinoma π višestruka. Naime, neka je r_j nula višestrukosti m, onda je odgovarajući sabirak, koji odgovara nuli r_j, oblika

$$\left(\sum_{\ell=0}^{m-1} C_{j,\ell} n^\ell\right) r_j^n.$$

Kao što vidimo i ovaj sabirak teži nuli pod uslovom $|r_j| < 1$.

Na osnovu ovih razmatranja interval apsolutne stabilnosti opisujemo sledećom definicijom.

Definicija 7.6 *Ah pripada intervalu apsolutne stabilnosti linearnog višekoračnog metoda ako i samo ako sve nule polinoma $\pi(\cdot; Ah)$ leže unutar jediničnog kruga.*

Ovde nam je od značaja sledeća teorema koja opisuje interval apsolutne stabilnosti konvergentnih metoda.

Teorema 7.6 *Neka je I interval apsolutne stabilnosti konvergentnog linearnog višekoračnog metoda, onda postoji $\delta > 0$ tako da važi $I \cap (0, \delta) = \emptyset$.*

Svaki optimalni linearni višekoračni metod ima prazan interval apsolutne stabilnosti.

Drugim rečima, ako je metod konvergentan onda interval apsolutne stabilnosti ne može sadržati pozitivne vrednosti Ah bliske nuli. Iz ovog razloga interval apsolutne stabilnosti opisujemo kao skup $(\gamma, 0)$, gde je $\gamma < 0$. Za familije Adamsovih, Nystromovih i Milne-Simponovih metoda intervali apsolutne stabilnosti su dati u tabeli 7.1. Kao što vidimo iz tabele Simpsonov metod je optimalan pa je zato interval apsolutne stabilnosti prazan skup. Međutim, i ostali prikazani metodi Nystromove ili Milne-Simpsonove familije imaju prazan interval apsolutne stabilnosti.

U nekim slučajevima nije ni potrebno posmatrati apsolutnu stabilnost. Naime, dovoljno je zahtevati da se ne povećava relativna greška. Pod pojmom interval relativne stabilnosti podrazumevamo onaj interval $(0, \delta)$ koji ima osobinu da ako je $Ah \in (0, \delta)$, onda ne dolazi do povećanja relativne greške tokom integracije. Primetimo da je interval relativne stabilnosti ograničen na pozitivne realne brojeve, jer ne možemo očekivati da je relativna greška ograničena ako apsolutna greška nije, a interval apsolutne stabilnosti je podskup skupa negativnih realnih brojeva.

Diferencna jednačina (7.14), koju zadovoljava greška integracije e_n, je u suštini ista kao i diferencna jednačina koju zadovoljava niz rešenja y_n. To znači da će i rešenja biti opisana na isti način

$$e_n = \sum_{i=1}^{k} C_i r_i^n + \frac{C}{\pi(1; Ah)}, \quad y_n = \sum_{i=1}^{k} D_i r_i^k + \frac{T}{\pi(1; Ah)},$$

gde smo pretpostavili da je niz T_n konstantan i gde smo pretpostavili da su sve nule polinoma π međusobno različite. Ako je $|r_1| > |r_i|$, $i = 2, \ldots, k$, onda je relativna greška za dovoljno veliko n, određena sa

$$\frac{|e_n|}{|y_n|} = \frac{C_1 + C_2(r_2/r_1)^n + \cdots + C_k(r_k/r_1)^n + C/(r_1^n \pi(1; Ah))}{D_1 + D_2(r_2/r_1)^n + \cdots + C_k(r_k/r_1)^n + T/(r_1^n \pi(1; Ah))} \approx \frac{C_1}{D_1}.$$

Kao što vidimo relativna greška je bar u prvoj aproksimaciji ograničena. Ovde nije bilo potrebno pretpostaviti da su sve nule polinoma π međusobno različite. Dovoljno je samo pretpostaviti da važi $|r_1| > |r_i|$, $i = 2, \ldots, k$. Na osnovu ovih razmatranja interval relativne stabilnosti opisujemo sledećom definicijom.

Definicija 7.7 *Ah pripada intervalu relativne stabilnosti linearnog višekoračnog metoda ako i samo ako je nula sa najvećim modulom polinoma $\pi(\cdot; Ah)$ prosta.*

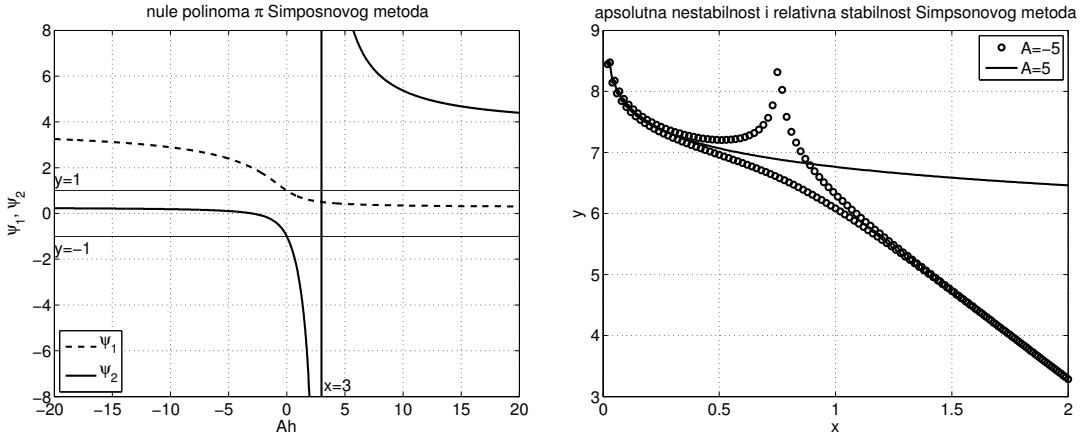
Na primer, posmatrajmo Simpsonov metod (7.8). Ovaj metod je idealan da ukažemo na razliku između pojmova apsolutne i relativne stabilnosti. Polinom π Simpsonovog metoda je

$$\pi(\psi; Ah) = \rho(\psi) - Ah\sigma(\psi) = \left(1 - \frac{Ah}{3}\right)\psi^2 + \frac{4Ah}{3}\psi - \left(1 + \frac{Ah}{3}\right).$$

Nule ovog polinoma su date izrazima

$$\psi_1 = \frac{2Ah - \sqrt{3(3+(Ah)^2)}}{Ah-3}, \quad \psi_2 = \frac{2Ah + \sqrt{3(3+(Ah)^2)}}{Ah-3}.$$

Nije jednostavno analizirati ove izraze, pa ćemo zato posmatrati grafike obe funkcije. Slika 7.6, levo, ilustruje grafike ove dve funkcije. Sa slike se vidi da ne postoji oblast na realnoj pravoj gde su obe funkcije po modulu manje od jedinice. Zaključujemo da ne postoji oblast apsolutne stabilnosti Simpsonovog metoda. Sa druge strane, takođe, sa slike vidimo da je uvek jedna nula dominantna. Drugim rečima, ne postoji tačka ili oblast u kojoj bi nule bile jednakih modula. Zaključujemo da je oblast relativne stabilnosti metoda skup pozitivnih realnih brojeva. Ovo ima velike posledice



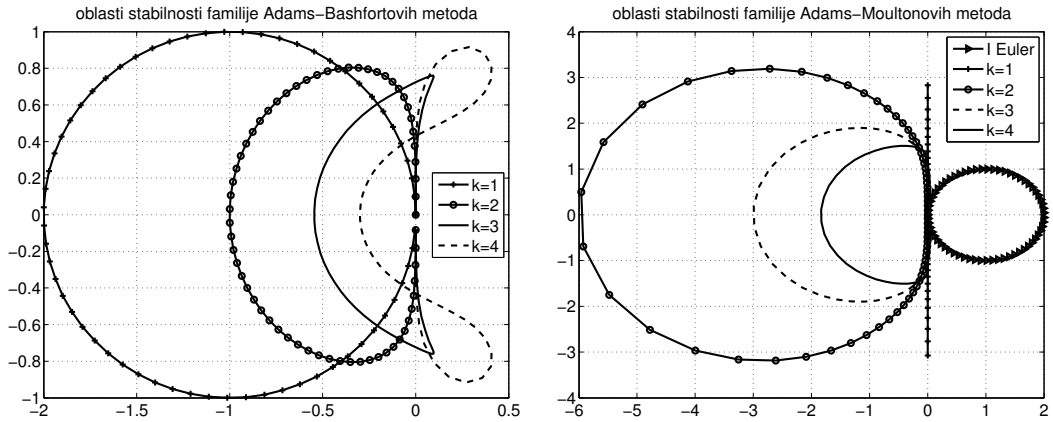
Slika 7.6: Slika levo ilustruje nule polinoma $\pi(\cdot; Ah)$ u funkciji Ah, slika desno ilustruje koncept apsolutne i relativne stabilnosti Simpsonovog metoda za rešavanje Cauchyevog problema.

na primenu Simposonovog metoda za integraciju diferencijalnih jednačina. Naime, kako apsolutna stabilnost ne postoji, to apsolutna greška nije ograničena već raste kako napredujemo. Na primer, ako primenimo Simpsonov metod na integraciju Cauchyevog problema $y' = -5y$, $y(0) = 1$, onda možemo očekivati da, kako apsolutna greška raste tokom napredovanja u intervalu integracije, gubimo značajne cifre, jer rešenje očigledno opada. Upravo se ovo i dešava, što je ilustrovano na slici 7.6, desno, grafik prikazan tačkastom linijom. Međutim, kako je metod relativno stabilan to znači da ako bude primenjen na integraciju Cauchyevog problema $y' = 5y$, $y(0) = 1$, neće doći do gubitka značajnih cifara, jer je relativna greška ograničena, a rast apsolutne greške ne može toliko da naruši tačnost rešenja, jer i samo rešenje raste. Ovo se jasno vidi na već pomenutoj slici, linija prikazana punom linijom.

Postoji jednostavan metod za određivanje intervala apsolutne stabilnosti nekog metoda. Nas zanimaju samo nule metoda koje se nalaze na jediničnom krugu, kao granični slučaj nula koje se nalaze unutar jediničnog kruga kompleksne ravni. Neku nulu polinoma π koja se nalazi na jediničnom krugu možemo odrediti kao $e^{i\theta}$, a vrednost Ah za koju se postiže ova nula možemo odrediti kao rešenje jednačine

$$\pi(e^{i\theta}; Ah) = \rho(e^{i\theta}) - Ah\sigma(e^{i\theta}) = 0, \quad Ah = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}.$$

Sve što treba uraditi je nacrtati u ravni grafik parametarski zadate krive Ah, njenog realnog i imaginarnog dela, u funkciji θ. Na ovaj način dobijamo oblast u kompleksnoj Ah ravni vrednosti Ah za koje su sve nule polinoma π po modulu manje ili jednake jedan.



Slika 7.7: Slika levo ilustruje oblasti stabilnosti familije Adams-Bashfortovih metoda, dok slika desno ilustruje oblasti stabilnosti Adams-Moultonovih metoda, pod metodom I Euler podrazumevamo implicitni Eulerov metod.

Slika 7.7, levo i desno, sadrži oblasti stabilnosti Adamsove familije metoda. Kao što vidimo vrednost γ u tabeli 7.1 možemo dobiti prostim očitavanjem sa slike najmanje vrednosti na x-osi koja pripada oblasti stabilnosti.

Posebno je interesantna oblast stabilnosti implicitnog Eulerovog metoda koji je prikazan na slici 7.7. Kao što vidimo oblast stabilnosti je ili unutrašnjost krive ili spoljašnost. Međutim, unutrašnjost sadrži interval $(0, 2)$, pa kako je implicitni Eulerov metod konvergentan, na osnovu teoreme 7.6 možemo zaključiti da je oblast stabilnosti Eulerovog metoda spoljašnost krive. Ovo je od izuzetnog značaja, jer je implicitni Eulerov metod jedinstven, od do sada razmatranih metoda, po osobini da je oblast nestabilnosti ograničena, dok je oblast stabilnosti neograničen skup kompleksne ravni. Postoji grupa metoda sa ovom osobinom koju proučavamo u sekciji 7.3.8.

Od interesa je i metod

$$y_{n+1} - y_n = \frac{h}{2}(f_{n+1} + f_n), \quad n \in \mathbb{N}. \tag{7.15}$$

Ovaj metod je u literaturi poznat kao trapezni metod ili Crank-Nicolsonov metod. Karakteristika metoda je veliki red $p = 2$ s obzirom na vrednost $k = 1$, dok je kompleksnost izračunavanja jednaka implicitnom Eulerovom metodu. Kao što vidimo i interval apsolutne stabilnosti je isti kao kod implicitnog Eulerovog metoda. Međutim, sa slike 7.7, desno, se vidi da je oblast stabilnosti Crank-Nicolsonovog metoda znatno manja u odnosu na oblast stabilnosti implicitnog Eulerovog metoda, što implicitnom Eulerovom metodu daje prednost pri integraciji sistema diferencijalnih jednačina. Posebno, implicitni Eulerov metod ima bolje osobine pri integraciji takozvanih stif sistema diferencijalnih jednačina, koje obrađujemo u sekciji 7.5.3.

7.3.6 Prediktor-korektor metodi

Problem koji nastaje prilikom korišćenja implicitnih linearnih višekoračnih metoda sastoji se u nelinearnoj jednačini. U svakom koraku metoda, da bismo odredili y_n moramo rešavati nelinearnu

jednačinu. Međutim, po ostalim karakteristikama, oblast stabilnosti i red, ovi metodi imaju prednost u odnosu na eksplicitne metode. Prediktor-korektor metodi spadaju u grupu metoda koja pokušava da pomiri ove suprotnosti.

Ideja je primena metoda proste iteracije za rešavanje nelinearne jednačine kojom je određen metod korektora.

Teorema 7.7 *Pretpostavimo da funkcija f zadovoljava sledeći uslov*

$$\left|\frac{\partial f(x,y)}{\partial y}\right| \leq L \tag{7.16}$$

nad oblašću od interesa. Ako jednačina

$$y_{n+k} = \psi_{n+k} + h\beta_k f(x_{n+k}, y_{n+k}), \quad \psi_{n+k} = \sum_{i=0}^{k-1}(h\beta_i f_{n+i} - \alpha_i y_{n+i}),$$

ima rešenje, onda iterativni proces

$$y_{n+k}^{\ell+1} = \psi_{n+k} + h\beta_k f(x_{n+k}, y_{n+k}^{\ell}), \quad \ell \in \mathbb{N}_0,$$

konvergira pod uslovom $h|\beta_k|L < 1$, uz dovoljno dobru startnu vrednost y_{n+k}^0.

Dokaz. Namera nam je da iskoristimo teoremu 3.1. Definišimo funkciju

$$\phi(y_{n+k}) = \psi_{n+k} + h\beta_k f(x_{n+k}, y_{n+k}).$$

Kako jednačina ima rešenje to postoji okolina rešenja, recimo interval $[a,b]$, gde funkcija ϕ zadovoljava uslov $\psi : [a,b] \to [a,b]$. Ostaje nam da pokažemo da je izvod funkcije ϕ manji od jedinice.

Nalazimo

$$\left|\frac{d\phi(y_{n+k})}{dy_{n+k}}\right| = h|\beta_k| \left|\frac{\partial f(x_{n+k}, y_{n+k})}{\partial y_{n+k}}\right| \leq h|\beta_k|L < 1,$$

odakle zaključujemo da je iterativni proces konvergentan, pod uslovom da možemo da obezbedimo $y_{n+k}^0 \in [a,b]$. □

Prilikom implementacije implicitnog Eulerovog metoda, kao startnu vrednost za rešavanje nelinearne jednačine koristili smo vrednost koju izračunava eksplicitni Eulerov metod. Kao metod za rešavanje nelinearne jednačine koristili smo metod proste iteracije. Kod prediktor-korektor metoda filozofija je ista. Koristimo jedan eksplicitni metod koji nazivamo prediktor i jedan implicitni metod koji nazivamo korektor. Međutim, da bi primena metoda korektora imala smisla mora biti zadovoljen uslov $h|\beta_k|L < 1$, po tvrđenju prethodne teoreme. Ako prediktor-korektor metod koristimo tako da prediktor samo konstruiše startnu vrednost za korektor, čije rešenje zatim određujemo sa unapred zadatom tačnošću, mi u suštini dobijamo metod istih karakteristika kao kod metoda korektora. Kvalitativno nova familija metoda se dobija kad broj iteracija, u metodu proste iteracije korektora ograničimo na unapred zadatu vrednost, recimo m.

Neka su metodi prediktora i korektora zadati na sledeći način

$$P: \quad \sum_{i=0}^{k} \alpha_i^P y_{n+i} = h\sum_{i=0}^{k-1} \beta_i^P f_{n+i},$$

$$C: \quad \sum_{i=0}^{k} \alpha_i^C y_{n+i} = h\sum_{i=0}^{k} \beta_i^C f_{n+i}.$$

Ovde imamo male probleme u notaciji. Naime, prediktor i korektor ne moraju imati istu vrednost k, a mi smo napisali oba metoda sa istom vrednošću k. Ovaj problem prevazilazimo tako što izaberemo $k = \max\{k_P, k_C\}$, gde su k_P i k_C odgovarajuće vrednosti metoda prediktora i korektora, a nedostajuće koeficijente α_i i β_i, u metodu u kome nedostaju, postavimo na vrednost nula. Na primer, ako koristimo par metoda

$$y_{n+1} = y_n + hf_n, \quad y_{n+2} = y_{n+1} + \frac{h}{12}(5f_{n+2} + 6f_{n+1} - f_n),$$

mi prosto izaberemo $k = 2$ i metode pišemo u obliku

$$y_{n+2} = y_{n+1} + hf_{n+1}, \quad y_{n+2} = y_{n+1} + \frac{h}{12}(5f_{n+2} + 6f_{n+1} - f_n).$$

Primetimo da sada eksplicitni Eulerov metod možemo da opišemo sa $\alpha_2 = 1$, $\alpha_1 = -1$, $\alpha_0 = 0$, $\beta_2 = 0$, $\beta_1 = 1$ i $\beta_0 = 0$. Na ovaj način dobijamo uniformnu notaciju.

Označimo sa P jednu primenu metoda prediktora i sa C jednu primenu metoda korektora, dok sa E označimo jedno izračunavanje funkcije f. Metod koji koristi m iteracija u metodu korektora možemo opisati na sledeći način $P(EC)^m$, dakle, potrebno je jedno izračunavanje prediktora, zatim m uzastopnih izračunavanja funkcije f i metoda korektora. Sistemom jednačina to možemo opisati na sledeći način

$$\begin{aligned} P: &\quad y_{n+k}^{(0)} = h\sum_{i=0}^{k-1}\beta_i^P f_{n+i} - \sum_{i=0}^{k-1}\alpha_i^P y_{n+i}, \\ \left.\begin{aligned} E: &\quad f_{n+k}^{(\ell)} = f(x_{n+k}, y_{n+k}^{(\ell)}), \\ C: &\quad y_{n+k}^{(\ell+1)} = h\textstyle\sum_{i=0}^{k-1}\beta_i^P f_{n+i} + h\beta_k f_{n+k}^{(\ell)}, \end{aligned}\right\} &\quad \ell = 0, \ldots, m-1. \end{aligned}$$

Primetimo da je kod ovog metoda poslednja vrednost y_{n+k} u kojoj je izračunata vrednost funkcije $y_{n+k}^{(m-1)}$, dok je vrednost elementa y_{n+k} koju označavamo kao rešenje zadata sa $y_{n+k}^{(m)}$. Iz ovog razloga se razmatraju i prediktor korektor metodi $P(EC)^m E$, koji se mogu opisati sledećim algoritmom

$$\begin{aligned} P: &\quad y_{n+k}^{(0)} = h\sum_{i=0}^{k-1}\beta_i^P f_{n+i} - \sum_{i=0}^{k-1}\alpha_i^P y_{n+i}, \\ \left.\begin{aligned} E: &\quad f_{n+k}^{(\ell)} = f(x_{n+k}, y_{n+k}^{(\ell)}), \\ C: &\quad y_{n+k}^{(\ell+1)} = h\textstyle\sum_{i=0}^{k-1}\beta_i^P f_{n+i} + h\beta_k f_{n+k}^{(\ell)}, \end{aligned}\right\} &\quad \ell = 0, \ldots, m-1. \\ E: &\quad f_{n+k} = f(x_{n+k}, y_{n+k}^{(m)}). \end{aligned}$$

Pokazuje se da metodi $P(EC)^m$ i $P(EC)^m E$ imaju različite numeričke karakteristike.

Karakteristike koje su nama od interesa su red metoda i oblast stabilnosti. Neka su p^P i p^C redovi metoda prediktora i korektora redom. Pod pretpostavkom $p^P = p^C - \ell$, $\ell \geq 0$, može se pokazati da metodi $P(EC)^m$ i $P(EC)^m E$ imaju red konvergencije $p = p^C$ pod uslovom $m \geq \ell$. Drugim rečima, ako izaberemo prediktor čiji je red za $\ell \geq 0$ manji od reda korektora, primena $m \geq \ell$ prostih iteracija proizvodi metod koji ima red korektora. Ovo je od značaja jer u suštini ne moramo proveravati nikakvu konvergenciju, dovoljno je samo znati razliku $\ell = r^C - r^P$ i toliko iteracija upotrebiti u metodu proste iteracije u metodu korektora da bismo zadržali red metoda korektora. Ako izaberemo baš $m = \ell$, onda metod ima isti red kao metod korektora, ali izraz za

lokalnu grešku odsecanja nije isti kao kod metoda korektora, jer se asimptotske konstante greške metoda prediktor-korektor i metoda korektora razlikuju.

Iz ovih razloga se kao metod prediktora koristi metod čiji je red manji ili jednak metodu korektora, a onda se saobrazno razlici $\ell = p^C - p^P$, bira broj iteracija $m \geq \ell$.

U praksi se najčešće bira slučaj $r^P = r^C - 1$, $\ell = 1$, i bira se metod $PECE$. U ovom slučaju oblast stabilnosti je određena polinomom

$$\pi_{PECE}(\psi; Ah) = \pi_C(\psi; Ah) + Ah\beta_k^C \pi_P(\psi; Ah).$$

`Matlab` implementacija prediktor-korektor metoda, funkcija `ode113`, upravo koristi metod $PECE$, sa Adamsovom familijom metoda.

U generalnom slučaju može se pokazati da važi

$$\begin{aligned}\pi_{P(EC)^m E}(\psi; Ah) &= \pi_C(\psi; Ah) + AhM_m \pi_P(\psi; Ah),\\ \pi_{P(EC)^m}(\psi; Ah) &= \beta_k^C \psi^{k_P} \pi_C(\psi; Ah) + AhM_m(\rho_C(\psi)\sigma_P(\psi) - \rho_P(\psi)\sigma_C(\psi)),\end{aligned}$$

gde je

$$M_m = \frac{(Ah\beta_k^C)^m(1 - Ah\beta_k^P)}{1 - (Ah\beta_k^P)^m}.$$

Primetimo da smo u polinomu $\pi_{P(EC)^m}$ iskoristili vrednost k_P kao stepen promenljive ψ. Ovo je neophodno jer bi upotreba vrednosti k mogla proizvesti grešku za $k = k_C > k_P$.

U specijalnom slučaju kad izaberemo $p = p^P = p^C$ može se upotrebiti takozvani Milneov metod za popravljanje rezultata metoda. Pod ovom pretpostavkom, uz upotrebu lokalne greške odsecanja, možemo pisati

$$\begin{aligned}y(x_{n+k}) - y_{n+k}^0 &= C_{p+1}^P h^{p+1} y^{(p+1)}(x_{n+k}) + O(h^{p+2}),\\ y(x_{n+k}) - y_{n+k}^m &= C_{p+1}^C h^{p+1} y^{(p+1)}(x_{n+k}) + O(h^{p+2}).\end{aligned}$$

Odavde oduzimanjem i sa malo manipulacije možemo dobiti ocenu

$$y(x_{n+k}) - y_{n+k}^m \approx C_{p+1}^C h^{p+1} y^{(p+1)}(x_{n+k}) \approx \frac{C_{p+1}^C}{C_{p+1}^P - C_{p+1}^C}(y_{n+k}^m - y_{n+k}^0).$$

Kako su na desnoj stani veličine koje izračunavamo u svakoj iteraciji, na osnovu ovog izraza možemo dobiti procenu lokalne greške odsecanja metoda korektora. Ovo se i koristi prilikom implementacije para prediktor-korektor metoda istog reda.

Međutim, možemo dobiti i više. Manipulacijom izrazima, može se pokazati da su vrednosti

$$\begin{aligned}\tilde{y}_{n+k}^0 &= y_{n+k}^0 + \frac{C_{p+1}^P}{C_{p+1}^P - C_{p+1}^C}(y_{n+k-1}^m - y_{n+k-1}^0),\\ \tilde{y}_{n+k}^m &= y_{n+k}^m + \frac{C_{p+1}^P}{C_{p+1}^P - C_{p+1}^C}(y_{n+k}^m - y_{n+k}^0),\end{aligned}$$

bolje aproksimacije od vrednosti y_{n+k}^0 i y_{n+k}^m. Ove korekcije nazivaju se Milneove korekcije. Prilikom implementacije para prediktor-korektor istog reda uvek je uputno koristiti Milneovu modifikaciju.

Ako označimo sa M upotrebu Milneove korekcije, onda prediktor-korektor metodi dobijaju sledeću simboličku notaciju $PM(EC)^m ME$ i $PM(EC)^m M$.

7.3.7 Matlab implementacija Adams-Bashfort-Moultonovog metoda

Funkcija `ode113` implementira Adams-Bashfortove i Admas-Moultonove metode. Pri tome se Adams-Bashfortovi metodi koriste kao prediktor metodi, dok se Adams-Moultonovi metodi koriste kao korektor metodi. Funkcija implementira metod $PECE$, pri čemu red metoda zavisi od zahtevane tačnosti.

Opšti format funkcije je

```
[x,y]=ode113(fun,[x0 b],y0)
```

Ulazni parametar `fun` predstavlja podatak tipa `function_handle` koji reprezentuje funkciju f u postavci Cauchyevog problema, ulazni parametri `x0` i `b` su početak i kraj intervala integracije na kome integralimo diferencijalnu jednačinu, dok je `y0` početni uslov Cauchyevog problema u tački `x0`.

Na primer, možemo pozvati funkciju na izvršenje na sledeći način

```
>>f=@(x,y) -y;
>>[x,y]=ode113(f,[0,1],1)
x =
                         0
       0.00790569415042095
        0.0237170824512628
        0.0553398590529466
         0.118585412256314
         0.218585412256314
         0.318585412256314
         0.418585412256314
         0.518585412256314
         0.618585412256314
         0.718585412256314
         0.818585412256314
         0.918585412256314
y =
                         1
         0.992125555849579
         0.976562032499627
         0.946163606532549
          0.88817600736786
         0.803655007898871
         0.727176867938557
         0.657976856879659
         0.595362080808059
         0.538705896500395
         0.487441257916941
         0.441055088952001
         0.399083147453228
         0.367879411024711
```

Kao što vidimo dobijemo rezultat integracije, i to x koordinate i vrednosti funkcije y.

Ako funkciju `ode113` pozovemo sa drugim argumentom `x` koji predstavlja vektor dužine veće od dva, funkcija vraća vrednosti rešenja samo u koordinatama vektora `x`. Na primer, ako su nam za prethodni problem potrebne samo vrednosti u tačkama 0, 1/3, 1/2 i 1, postupamo na sledeći način

```
[x,y]=ode113(f,[0 1/3 1/2 1],1)
x =
                          0
     3.333333333333333e-01
     5.000000000000000e-01
     1.000000000000000e+00
y =
     1.000000000000000e+00
     7.165312185881303e-01
     6.065305961154958e-01
     3.678794110247113e-01
```

U mogućnosti smo i da postavljamo izvesne parametre kojima možemo da upravljamo izvršenjem algoritma. Nas pre svega zanimaju opcije koje omogućavaju promenu parametara koji kontrolišu grešku i parametara koji kontrolišu korak prilikom integracije. Da bismo bili u mogućnosti da postavljamo vrednosti ovih opcija koristimo funkciju `odeset`. Funkcija `odeset` ima sledeći format

```
ops=odeset('imeOpcije','vrednost',...)
```

Funkcija `odeset` vraća strukturu podataka koja je ispunjena opcijama koje se zadaju parametrom `'imeOpcije'`, a čije su vrednosti zadate ulaznim parametrima `'vrednost'`. U pozivu funkcije može biti po volji veliki broj ovakvih parova.

Da bismo mogli da uspešno kontrolišemo način rada algoritma neophodna nam je opcija `'Stats'`. Postavljanjem ove opcije dobijamo izveštaje, od strane funkcije `ode113`, šta se dešava prilikom izvršenja algoritma. Sledeća sekvenca naredbi ilustruje postavljanje ove opcije i njeno delovanje na prethodnom primeru.

```
>>ops=odeset('Stats','on');
>>[x,y]=ode113(f,[0,1],1,ops);
13 successful steps
0 failed attempts
27 function evaluations
```

Kao što vidimo dobijamo izveštaj o broju uspešnih koraka, broju neuspešnih pokušaja i ukupnom broju izračunavanja vrednosti funkcije. Broj izračunavanja funkcije koji je skoro dva puta veći od broja koraka može nam omogućiti da shvatimo da se radi o paru metoda prediktoru i korektoru. Ako malo zakomplikujemo funkciju dobićemo i malo realističnije izveštaje.

```
>>f=@(x,y) 10*exp(-(5*(y-1.41))^2);
>> [x,y]=ode113(f,[0,1],1,ops);
26 successful steps
4 failed attempts
57 function evaluations
```

Kao što vidimo sada već dolazi do pojave neuspelih pokušaja, što dovodi do povećanja broja izračunavanja funkcije. Primetimo da dolazi i do smanjenja koraka u odnosu na prethodni slučaj, kad je na istom intervalu sve bilo rešeno u 13 koraka.

Opcije koje su vezane za kontrolu tačnosti su `'RelTol'`, `'AbsTol'` i `'NormControl'`. Opcije `'RelTol'` i `'AbsTol'` određuju gornju granicu dozvoljene greške pri svakom koraku, i to `'RelTol'` određuje gornju granicu relativne greške, dok `'AbsTol'` određuje prihvatljivu gornju granicu apsolutne greške. Naime, pri svakom koraku ocenjuju se vrednost greške $y(x_n) - y_n$, i ako je ispunjen uslov

$$|y(x_n) - y_n| \leq \max(\text{AbsTol}, \text{RelTol}\,|y_n|)$$

prelazi se na sledeći korak. Ako uslov nije ispunjen smanjuje se korak integracije. Vrednost `'AbsTol'` određuje praktično gornju granicu ispod koje vrednost rešenja smatramo jednakom nuli, i u tom slučaju koncept relativne greške prosto nije primenljiv. Vrednost `'AbsTol'` može imati vrednost u obliku vektora, u tom slučaju se pri svakom koraku može menjati vrednost opcije `'AbsTol'`. Ako je vrednost skalarna pri svakom koraku se primenjuje ista vrednost opcije `'AbsTol'`. Opcija `'NormControl'` primenjuje donekle različit način kontrole greške. Naime, umesto datog izraza koristi se

$$||(y(x_0), \ldots, y(x_n)) - (y_0, \ldots, y_n)|| \leq \max(\text{AbsTol}, \text{RelTol}\,||(y_0, \ldots, y_n)||),$$

gde se kao norma koristi neka od vektorskih normi. Ovaj način kontrole greške se ne preporučuje jer je način bez norme mnogo strožiji.

Podrazumevana vrednost opcije `'AbsTol'` iznosi 10^{-6}, opcije `'RelTol'` iznosi 10^{-3}, dok je podrazumevana vrednost opcije `'NormCotrol'` postavljena na `'off'`.

Postavljanje opcija možemo da ilustrujemo rešavanjem primera.

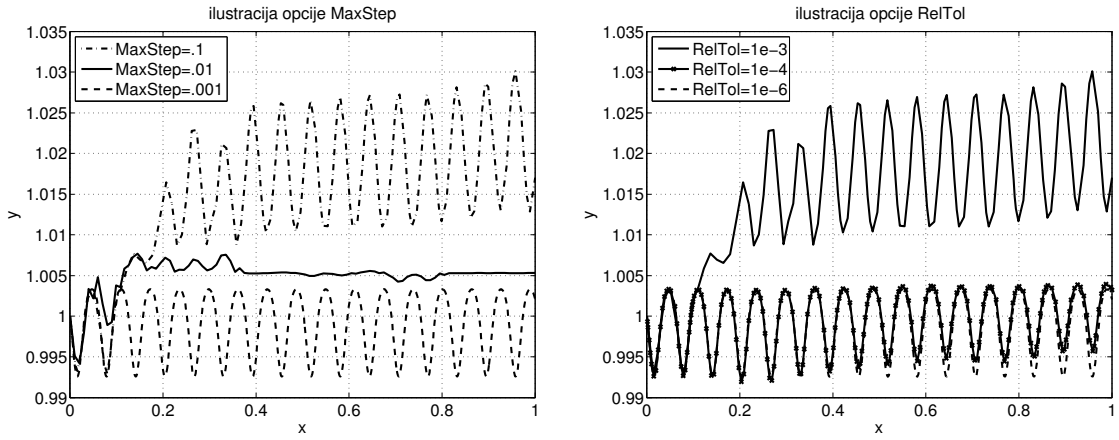
```
>>ops=odeset('Stats','on','AbsTol',1e-7,'RelTol',1e-10);
>>f=@(x,y) 10*exp(-(5*(y-1.41))^2);
>>[x,y]=ode113(f,[0 1],1,ops);
113 successful steps
7 failed attempts
234 function evaluations
```

Opcija `'InitialStep'` omogućava postavljanje inicijalnog koraka prilikom integracije. Ako vrednost nije postavljena funkcija `ode113` sama određuje korak sa kojim počinje integraciju, a koji određuje na osnovu procene vrednosti prvog izvoda u početnoj tački rešavanja. Međutim, u nekim slučajevima je pogodno izabrati ovu vrednost. Pogotovu ako je vrednost prvog izvoda mala u početnoj tački, a ubrzo iza nje dolazi do značajnog povećanja. U ovim slučajevima može da se desi da dođe, zbog velike vrednosti inicijalnog koraka, do preskakanja oblasti koja je od interesa.

Opcija `'MaxStep'` određuje maksimalnu dozvoljenu vrednost koraka prilikom integracije. Ovo je posebno značajno kada su koeficijenti jednačine periodične funkcije. Pod izvesnim uslovima, ako maksimalna vrednost koraka nije ograničena, može se desiti da vrednost koraka bude tolika da se preskaču čitave periode. Da se ovako nešto ne bi dešavalo postoji mogućnost limitiranja maksimalnog koraka integracije koja predstavlja izlaz iz ove situacije. Podrazumevana vrednost ove opcije iznosi $.1(b - x_0)$.

Kao lepa ilustracija upotrebe opcije `'MaxStep'` može nam poslužiti Cauchyev problem

$$y' = \cos(100x)\sin(100y), \quad y(0) = 1.$$



Slika 7.8: Slika levo ilustruje dejstvo opcije MaxStep pri integraciji periodičnih problema, slika desno dolazi do istih rezultata ali varira opciju RelTol

Ovaj problem ima eksplicitno rešenje i ono iznosi

$$y(x) = \frac{1}{50}\left(\arctan\left(\tan(50)e^{\sin(100x)}\right) + 16\pi\right).$$

Slika 7.8, levo, ilustruje integraciju sa različitim vrednostima opcije `'MaxStep'`. Interesantno je da za vrednost opcije .01 dolazi do gorih rezultata nego za vrednost .1 koja je podrazumevana. Rezultati su gori u smislu da se pri vrednosti .1 barem vidi nekakva periodičnost, koja se za vrednost opcije .01 i ne vidi. Naime, pri vrednosti opcije .01 dolazi tačno do preskakanja celih perioda tako da se informacija o periodičnosti gubi.

Do tačnih rezultata se može doći i postavljanjem opcije `'RelTol'`. U ovom slučaju je malo 'jeftinije' doći do tačnih rezultata upotrebom opcije `'RelTol'` nego postavljanjem opcije `'MaxStep'`, jer je potrebno oko pedeset izračunavanja funkcije f manje. Međutim, na ukupan broj izračunavanja funkcije f, koji iznosi oko hiljadu, ovo nije preterano velika ušteda.

Opcija `'MaxStep'` je 'skupa' i treba je štedljivo koristiti. Posebno je ne treba koristiti u slučaju kad je potrebno u izlazu generisati veći broj tačaka u podeli. Za ove namene postoji opcija `'Refine'` koja će u postignuto rešenje umetnuti dodatne tačke tako da se dodatne tačke u rešenju umeću bez preteranog zauzimanja procesorskog vremena. Podrazumevana vrednost ove opcije je jedan i treba je postaviti na neku celobrojnu vrednost koja je veća od jedan da bi svaki interval integracije dobio dodatne umetnute tačke.

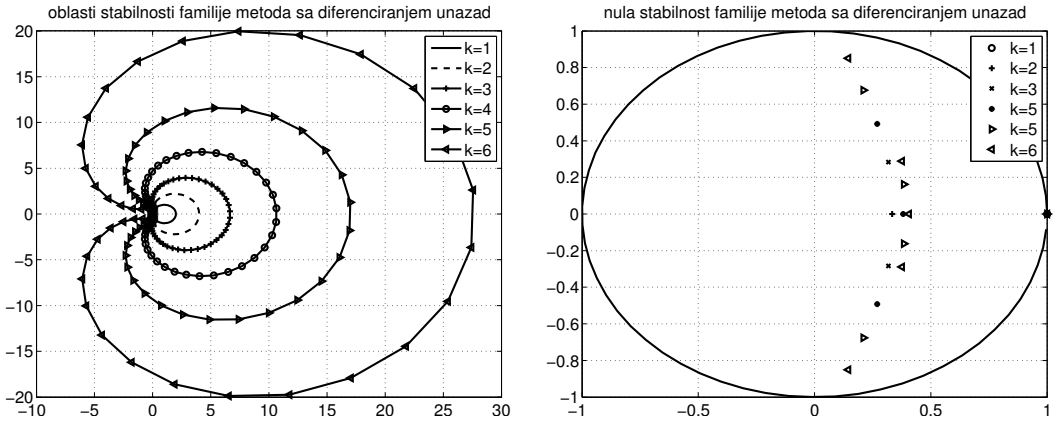
7.3.8 Metodi sa diferenciranjem unazad

Metodi sa diferenciranjem unazad (engleski backward differentiation formulas) su grupa metoda koja se karakteriše visokim redom i širokom oblašću stabilnosti. Videćemo da implicitni Eulerov metod prirodno pripada ovoj klasi metoda.

Pođimo od Cauchyevog problema (7.1) i zamenimo vrednost izvoda formulom za diferenciranje

k	1	2	3	4	5	6
α_0	-1	$1/3$	$-2/11$	$3/25$	$-12/137$	$10/147$
α_1	1	$-4/3$	$9/11$	$-16/25$	$75/137$	$-24/49$
α_2		1	$-18/11$	$36/25$	$-200/137$	$75/49$
α_3			1	$-48/25$	$300/137$	$-400/147$
α_4				1	$-300/137$	$150/49$
α_5					1	$-120/49$
α_6						1
β_k	1	$2/3$	$6/11$	$12/25$	$60/137$	$20/49$
p	1	2	3	4	5	6
C_{p+1}	$-1/2$	$-2/9$	$-3/22$	$-12/125$	$-10/137$	$-20/343$
γ	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Tabela 7.2: Konvergentni metodi familije sa diferenciranjem unazad.



Slika 7.9: Slika levo predstavlja oblasti stabilnosti metoda sa diferenciranjem unazad, dok slika desno predstavlja raspodelu nula polinoma ρ metoda sa diferenciranjem unazad.

unazad (6.1). Dobijamo familiju metoda

$$\frac{1}{h}\sum_{i=1}^{k}\frac{1}{i}\nabla^i y(x) = f(x,y),$$

koja posle malo preuređivanja postaje

$$\sum_{i=1}^{k}\frac{a}{i}\nabla^i y_{n+k} = haf_{n+k}, \quad a = \left(\sum_{i=1}^{k}\frac{1}{i}\right)^{-1}, \tag{7.17}$$

gde smo pomnožili jednačinu sa a, da bismo dobili normalizovanu formu linearnog višekoračnog metoda.

Prvih šest metoda ove familije prikazani su u tabeli 7.2.

Konzistenciju metoda možemo pokazati jednostavno koristeći teoremu 6.4.

Teorema 7.8 *Red metoda* (7.17) *iznosi* k.

Dokaz. Ovo je direktna posledica teoreme 6.4, jer

$$L(y(x)) = \sum_{i=1}^{k} \frac{a}{i} \nabla^i y(x) - hay'(x) = -\frac{ah^{k+1}}{k+1} f^{(k+1)}(\psi), \quad \psi \in (x-kh, x),$$

čime smo završili dokaz teoreme. □

Međutim, može se pokazati da su samo metodi familije prikazani u tabeli 7.2 nula stabilni. Metodi višeg reda ove familije koji se mogu konstruisati nisu nula stabilni, pa samim tim ni konvergentni. Slika 7.9, desno, ilustruje raspodelu nula prvog karakterističnog polinoma metoda prikazanih u tabeli 7.2.

Razlog zbog kog se ovi metodi proučavaju leži u oblasti stabilnosti. Slika 7.9, levo, prikazuje oblasti stabilnosti metoda prikazanih u tabeli 7.2.

Funkcija `ode15s` implementira ovu grupu metoda. Koristi se za rešavanje stif sistema običnih diferencijalnih jednačina, koje proučavamo u sekciji 7.5.3.

7.4 Metodi Runge-Kutta

Metod Runge-Kutta ima oblik

$$y_{n+1} = y_n + h\Phi(x_n, y_n, h), \tag{7.18}$$

gde je h korak integracije, a funkcija Φ je podesno izabrana tako da obezbeđuje konvergenciju metoda.

Pod konvergencijom metoda Runge-Kutta podrazumevamo iste uslove kao u definiciji konvergencije linearnih višekoračnih metoda.

Kao i kod linearnih višekoračnih metoda, prvo govorimo o redu metoda.

Definicija 7.8 *Metod* (7.18) *ima red* p *ako i samo ako*

$$y(x+h) - y(x) - h\Phi(x, y(x), h) = \sum_{i=p+1}^{+\infty} C_i h^i g_i(x) = C_{p+1} h^{p+1} g_{p+1}(x) + \cdots .$$

Definicija 7.9 *Metod* (7.18) *je konzistentan ako i samo ima red veći od nula.*

Sledeća teorema, pod izvesnim uslovima, pokazuje da je konzistentnost metoda potrebna i dovoljna za konvergenciju metoda.

Teorema 7.9 *Neka je*

$$D = \{(x, y, h) \,|\, x \in [x_0, b], \quad y \in \mathbb{R}, \quad h \in [0, h_0]\}$$

i neka je funkcija Φ *neprekidna nad* D *i neka je za svako* $(x, y_1, h), (x, y_2, h) \in D$ *ispunjen uslov*

$$|\Phi(x, y_1, h) - \Phi(x, y_2, h)| \leq M|y_1 - y_2|.$$

Metod (7.18) *je konvergentan ako i samo ako je konzistentan.*

Mi se nećemo baviti metodima konstrukcije metoda Runge-Kutta. Metodi konstrukcije zahtevaju previše prostora i priličnu količinu ispisanih stranica. Radije, daćemo pregled nekih osnovnih metoda koji se koriste. U opštem slučaju funkcija Φ ima sledeći oblik

$$\Phi(x, y, h) = \sum_{i=1}^{s} b_i k_i,$$

gde su b_i, $i = 1, \ldots, s$ brojevi dok su k_i, $i = 1, \ldots, s$, funkcije koje su određene na sledeći način

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_i &= f\left(x_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right), \quad i = 2, \ldots, s, \\ c_i &= \sum_{j=1}^{i-1} a_{ij}, \quad i = 2, \ldots, s, \end{aligned}$$

gde su opet a_{ij}, $j = 1, \ldots, i-1$, $i = 2, \ldots, s$, brojevi. Ovo je opšti oblik eksplicitnih metoda. Kod implicitnih metoda funkcije k_i, $i = 1, \ldots, s$, su zadate na sledeći način

$$k_i = f\left(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, \ldots, s.$$

Kao što vidimo dobijamo nelinearni sistem jednačina za čije rešavanje je potrebno upotrebiti neku od metoda za rešavanje nelinearnih jednačina. Mi se nećemo baviti implicitnim metodima Runge-Kutta, iako implicitni metodi imaju prednost u odnosu na eksplicitne, što se tiče intervala apsolutne stabilnosti.

Zbog načina na koji su zadati, sve eksplicitne metode Runge-Kutta možemo opisati pomoću Butcherove tabele koja ima formu prezentovanu u tabeli 7.3. Kao što vidimo zbir elemenata a_{ij}

c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_s	a_{s1}	$a_{s,2}$	$\cdots$	$a_{s,s-1}$	
	b_1	b_2	$\cdots$	b_{s-1}	b_s

Tabela 7.3: Butcherova tabela eksplicitnog metoda Runge-Kutta.

po vrstama Butcherove tabele mora odgovarati elementima c_i u vrsti.

Sad možemo vrlo jednostavno koristeći Butcherove tabele opisati klasične metode Runge-Kutta koji imaju pretežno istorijsku vrednost.

Prvi je metod kod koga je $s = 2$, ovde dobijamo jedno parametarsku familiju metoda reda dva koja se može opisati Butcherovom tabelom 7.4. Za specijalne vrednosti parametra $\alpha = 1/2$ i $\alpha = 1$, dobijaju se Euler-Cauchyev i poboljšani Euler-Cauchyev metod, takođe prikazini u tabeli.

Ako izaberemo $s = 3$, dobijamo metode trećeg reda koji su prikazni u tabeli 7.5. Tabela levo predstavlja Heunov metod, dok tabela desno ima istorijski značaj jer je korišćena za ručna izračunavanja.

α	α	
	$(2\alpha-1)/(2\alpha)$	$1/(2\alpha)$

$1/2$	$1/2$	
	0	1

1	1	
	$1/2$	$1/2$

Tabela 7.4: Butcherova tabela levo predstavlja jednoparametarski metod drugog reda, tabela u sredini predstavlja Euler-Cauchyev metod, dok tabela desno predstavlja poboljšani Euler-Cauchyev metod.

$1/3$	$1/3$		
$2/3$	0	$2/3$	
	$1/4$	0	$3/4$

$1/2$	$1/2$		
1	-1	2	
	$1/6$	$2/3$	$1/6$

Tabela 7.5: Butcherova tabela Heunovog metoda tabela levo, i tabela jednog metoda trećeg reda desno.

Izaberemo li $s = 4$, možemo dobiti metode reda četiri, od kojih su dva najviše korišćena data u tabeli 7.6. Metod u tabeli levo se najčešće koristi, dok je Gillov metod, prikazan u tabeli desno, pogodan za integraciju sistema diferencijalnih jednačina.

$1/2$	$1/2$			
$1/2$	0	$2/3$		
1	0	0	1	
	$1/6$	$1/3$	$1/3$	$1/6$

$1/2$	$1/2$			
$1/2$	$(\sqrt{2}-1)/2$	$1-\sqrt{2}/2$		
1	0	$-\sqrt{2}/2$	$1+\sqrt{2}/2$	
	$1/6$	$(2-\sqrt{2})/6$	$(2+\sqrt{2})/6$	$1/6$

Tabela 7.6: Butcherova tabela metoda četvrtog reda levo, tabela desno je metod u literaturi poznat kao Gillov.

Metodi Runge-Kutta imaju slično ponašanje što se tiče greške. Naime, pod vrlo pojednostavljenim uslovima može se tvrditi da je gornja granica apsolutne greške u n-tom koraku određena sa

$$|y(x_n) - y_n| \leq \frac{Kh^p}{L}\left(e^{L(x_n - x_0)} - 1\right),$$

gde je L Lipschitzova konstanta funkcije f, dok je K izvesna konstanta. Dakle, red metoda, kao i kod linearnih višekoračnih metoda, određuje način na koji greška metoda zavisi od koraka h.

Interval apsolutne stabilnosti prikazanih metoda je interval oblika $(\alpha, 0)$. Za $s = 2$, interval apsolutne stabilnosti je $(-2, 0)$, dok je za $s = 3$, interval $(-2.51, 0)$, i za $s = 4$ ovaj interval iznosi $(-2.78, 0)$.

Za razliku od linearnih višekoračnih metoda, metodi Runge-Kutta ne zahtevaju poznavanje dodatnih startnih vrednosti izuzev vrednosti koja je zadata prilikom postavljanja Cauchyevog problema. Ovo je prednost metoda Runge-Kutta u odnosu na linearne višekoračne metode. Međutim, ovi metodi zahtevaju barem s izračunavanja funkcije f po svakom koraku integracije h, dok linearni višekoračni metodi nemaju ovakav zahtev. Konačno, linearni višekoračni metodi uglavnom koriste metode Runge-Kutta odgovarajućeg reda za izračunavanje nedostajućih startnih vrednosti.

7.4.1 Matlab implementacija eksplicitnih metoda Runge-Kutta

Funkcije `ode23` i `ode45` su `Matlab` implementacije eksplicitnih metoda Runge-Kutta. Funkcija `ode23` implementira dva metoda jedan drugog i jedan trećeg reda, dok funkcija `ode45` implementira, takođe dva metoda, jedan četvrtog i jedan petog reda. Ideja sa implementacijom dva metoda se sastoji u mogućnosti procene greške i adaptaciji koraka u slučaju da greška postaje nedopustivo velika. Kod obe funkcije eksplicitne metode višeg reda možemo opisati na sledeći način

$$y_{n+1} = y_n + h\sum_{k=1}^{s} b_i k_i, \quad k_i = f\left(x_n + c_i h, y_n + h\sum_{j=1}^{i-1} a_{ij}k_j\right), \quad i = 1, \ldots, s,$$

dok metod nižeg reda ima formu

$$y^*_{n+1} = y_n + h\sum_{i=1}^{s} b^*_i k_i.$$

Kao što vidimo metodi višeg i nižeg reda se razlikuju jedino u koeficijentima b_i i b^*_i, $i = 1, \ldots, s$. Par implementiran u funkciji `ode23` se naziva Bogacki-Shampine (2,3) par i ima parametar $s = 4$, dok su koeficijenti c_i, a_{ij}, b_i i b^*_i, dati u proširenoj Butcherovoj tabeli 7.7. Tabela 7.8 sadrži Butcherovu tabelu para Dormand-Prince (4,5) koji se nalazi implementiran u funkciji `ode45`. Dormand-Prince (4,5) par ima parametar $s = 7$. Butcherove tabele parova su iste kao i Butcherove tabele

1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	7/24	1/4	1/3	1/8

Tabela 7.7: Butcherova tabela za Bogacki-Shampine (2,3) par eksplicitnih Runge-Kutta metoda koji se koristi u funkciji ode23.

1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	64448/6561	−212/729			
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40
	35/384	0	500/113	125/192	−2187/6784	11/84	0

Tabela 7.8: Butcherova tabela za Dormand-Prince (4,5) par eksplicitnih Runge-Kutta metoda koji se koristi u funkciji ode45.

običnih metoda Runge-Kutta, jedina razlika je što poslednja vrsta sadrži izraz za koeficijente b^*_i, $i = 1, \ldots, s$, kojih kod običnih metoda nema.

Greška u $(n+1)$-vom koraku se određuje jednostavno

$$y_{n+1} - y^*_{n+1} = h\sum_{i=1}^{s}(b_i - b^*_i)k_i.$$

Ako je greška manja od unapred zadate vrednosti korak se ne smanjuje, ako nije dolazi do smanjenja koraka. U ovome se sastoji adaptibilnost metoda.

Praktično jedina razlika između funkcija `ode23` i `ode45` je veći red metoda kod funkcije `ode45`. Naravno, veći red metoda znači i nešto komplikovanije formule za izračunavanja u jednom koraku. Međutim, kao što vidimo ovo nije preterano veliki problem, jer se izračunavanja najviše dupliraju.

Pozivanje funkcija je isto kao kod funkcije `ode113`. Opcije od interesa su iste kao kod funkcije `ode113`. Pomenimo samo da kod svih metoda integracije diferencijalnih jednačina opcija `'Refine'` ima podrazumevanu vrednost jedan, izuzev kod funkcije `ode45`. Ovde zbog izuzetno visokog reda metoda moguće je koristiti veliku vrednost koraka, pa je podrazumevana vrednost opcije `'Refine'` jednaka 4.

7.5 Sistemi običnih diferencijanih jednačina

Cauchyev problem za sistem običnih diferencijalnih jednačina ima sledeću formu

$$z' = F(x, z), \quad z(x_0) = z_0, \tag{7.19}$$

gde je $z = (z_1, \ldots, z_n)$ vektor funkcija koje tražimo, $F(x, z) = (f_1(x, z), \ldots, f_n(x, z))$ je vektorska funkcija, dok je $z(x_0) = z_0 = (z_{0,1}, \ldots, z_{0,m})$ početni uslov.

Sisteme običnih diferencijalnih jednačina proučavamo jer su značajni sami po sebi, ali takođe omogućavaju i rešavanje inicijalnih problema višeg reda. Na primer, posmatrajmo problem reda m sa realnim i skalarnim funkcijama

$$y^{(m)} = f(x, y, y', \ldots, y^{(m-1)}), \quad y^{(\ell)}(x_0) = y_0^{\ell}, \quad \ell = 0, \ldots, m-1.$$

Ako uvedemo smene $z_\ell = y^{(\ell-1)}$, $\ell = 1, \ldots, m$, dobijamo sledeći sistem običnih diferencijalnih jednačina

$$z_\ell' = z_{\ell+1}, \quad \ell = 1, \ldots, m-1, \quad z_m' = f(x, z_2, \ldots, z_m), \quad z_\ell(x_0) = y_0^{\ell-1}.$$

Ovaj sistem jednačina možemo zapisati u formi Cauchyevog problema u vektorskom obliku

$$z' = F(x, z), \quad z(x_0) = z_0,$$

gde je $z = (z_1, \ldots, z_m)$, $F(x, z) = (z_1, \ldots, z_{m-1}, f(x, z))$ i $z_0 = (y_0^0, \ldots, y_0^{m-1})$.

U narednim odeljcima objasnićemo kako se rezultati koje smo dobili za skalarni Cauchyev problem bez problema prenose na Cauchyev problem u vektorskom obliku. Videćemo da je prelaz sa skalarnih numeričkih metoda na vektorske gotovo trivijalan.

7.5.1 Linearni višekoračni metodi

Linearni višekoračni metod za rešavanje Cauchyevog problema (7.19) u vektorskom obliku ima isti oblik kao u slučaju skalarnih funkcija

$$\sum_{i=0}^{k} \alpha_i z_{n+i} = h \sum_{i=0}^{k} \beta_{n+i} F_{n+i}.$$

Sve veličine imaju isto značanje kao i u skalarnom slučaju. Jedino na šta treba obratiti pažnju je činjenica da su veličine z i F vektori.

Svi linearni višekoračni metodi se prenose direktno na Cauchyev problem u vektorskom obliku. Na primer, vrlo mala modifikacija programa za eksplicitni Eulerov metod omogućava primenu eksplicitnog Eulerovog metoda na vektorske funkcije.

```
function [ x,z ] = eulerMethodVector( x0, z0, b, h, fun )
%EULERMETHODVECTOR(X0,Z0,B,H,FUN) konstruise resenje Cauchyevog
% problema z'=fun(x,z), z(x0)=z0, sa korakom h na intervalu
% [x0,b]

x = x0:h:b-h;
z = [z0];
for t = x
    z = [z z(:,end)+h*fun(t,z(:,end))];
end
x = [x x(end)+h];
end
```

Kao što vidimo elemente niza rešenja tretiramo kao matricu, i rešenja u koraku n smeštamo u n-tu kolonu matrice. Na primer, integraciju Cauchyevog problema

$$z' = Az, \quad A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad z(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{7.20}$$

možemo uraditi na sledeći način

```
>>fun=@(x,z) [z(1)+z(2); z(1)-z(2)];
>>[x,z]=eulerMethodVector(0,[1; 1],2,.01,fun);
```

Rešenje prethodnog sistema jednačina je prikazano na slici 7.10, levo. Kao što vidimo rešenja dobijena eksplicitnim Eulerovim metodom se ponašaju kao i u slučaju skalarne jednačine. Sa porastom rastojanja od početnog uslova dolazi do pada broja značajnih cifara u rešenju, jer dolazi do akumulacije greške.

Prilikom proučavanja sistema diferencijalnih jednačina koristi se linearna jednačina koja ima sledeću formu

$$z' = Az, \quad z(x_0) = z_0,$$

gde je A matrica sa konstantnim elementima. Ovaj sistem linearnih jednačina se može rešiti eksplicitno pod pretpostavkom da je matrica A proste strukture. Ovo je slučaj kada sopstveni vektori matrice A formiraju bazu prostora $\mathbb{R}^m$, gde je m broj jednačina i u isto vreme broj nepoznatih funkcija. U ovom slučaju matrica A može da se napiše u sledećoj formi

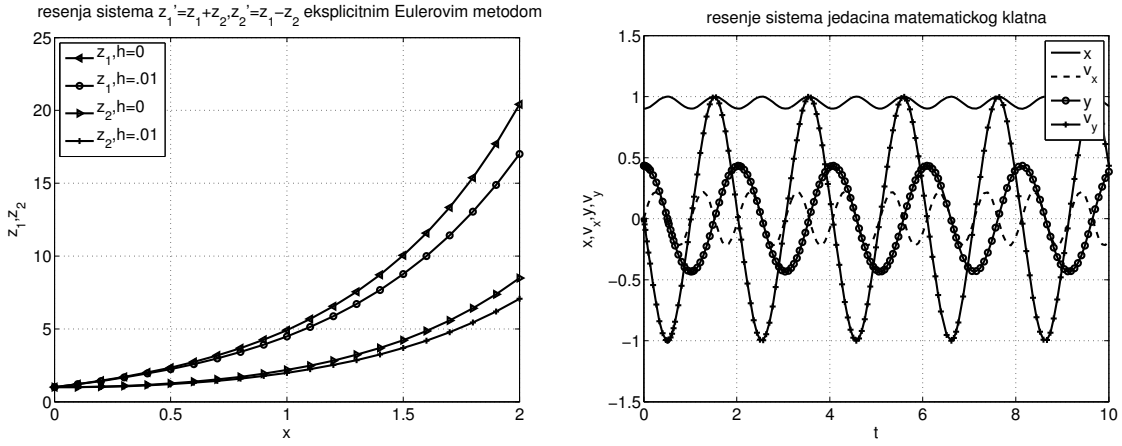
$$A = VDV^{-1},$$

gde je $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ matrica koja na glavnoj dijagonali ima sopstvene vrednosti λ_i, $i = 1, \ldots, m$, matrice A, a kolone matrice V su sopstveni vektori matrice A. U ovom slučaju rešenje sistema jednačina dobija formu

$$z = V\mathrm{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_m t})V^{-1}z_0.$$

Na primer, za Cauchyev problem (7.20), matrica A je proste strukture i egzaktno rešenje je

$$z = \begin{bmatrix} 1-\sqrt{2} & 1+\sqrt{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{-\sqrt{2}t} & 0 \\ 0 & e^{\sqrt{2}t} \end{bmatrix} \begin{bmatrix} 1/(1-\sqrt{2}) & 1/(1+\sqrt{2}) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$= \frac{1}{2} \begin{bmatrix} (1-\sqrt{2})e^{-\sqrt{2}t} + (1+\sqrt{2})e^{\sqrt{2}t} \\ e^{-\sqrt{2}t} + e^{\sqrt{2}t} \end{bmatrix}.$$



Slika 7.10: Slika levo ilustruje upotrebu eksplicitnog Eulerovog metoda za rešavanje Cauchyevog problema $z_1' = z_1 + z_2$, $z_2' = z_1 - z_2$, $z_1(0) = 1$, $z_2(0) = 1$, sa $h = 0$ su označena tačna rešenja, slika desno ilustruje rešenja sistema jednačina matematičkog klatna.

Svi koncepti koje smo uveli i ilustrovali u prethodnim sekcijama kad smo se bavili numeričkim metodima za rešavanje Cauchyevog problema u skalarnom obliku ostaju na snazi. Na primer, sva diskusija vezana za red metoda i ponašanje greške u vezi sa tim ostaje na snazi. Kao model problem, prilikom razmatranja sistema običnih diferencijalnih jednačina koristimo sistem običnih diferencijalnih jednačina sledećeg oblika

$$z' = Az, \quad z(x_0) = z_0,$$

gde je A matrica sa konstantnim elementima. Tako ulogu skalarne veličine Ah, koju smo posmatrali pri definiciji apsolutne stabilnosti skalarnih metoda, sada preuzima veličina Ah, gde je A matrica. Pod apsolutnom stabilnošću metoda, kao i do sada, podrazumevaćemo uslove pod kojima ne dolazi do porasta apsolutne greške rešenja. Međutim, očigledno je da ne možemo intervalu apsolutne stabilnosti pridati nikakvu interpretaciju ako veličinu Ah posmatramo kao matricu. Jedina smislena interpretacija je zamena matrice A njenim sopstvenim vrednostima.

Definicija 7.10 *Neka su $\lambda_i \in \mathbb{C}$ sopstvene vrednosti matrice A i neka su prvi i drugi karakteristični polinomi linearnog višekoračnog metoda ρ i σ, redom. Linearni višekoračni metod je apsolutno stabilan ako i samo ako za svaku sopstvenu vrednost λ_i polinom $\pi(\psi; \lambda_i h) = \rho(\psi) - \lambda_i h \sigma(\psi)$ ima sve nule po modulu manje od jedan.*

Ova definicija jasno pokazuje važnost oblasti stabilnosti linearnog višekoračnog metoda. Što je oblast stabilnosti linearnog višekoračnog metoda širi skup kompleksne ravni, to će za širu klasu matrica linearni višekoračni metod biti apsolutno stabilan. Metodi sa diferenciranjem unazad najpodesniji su za rešavanje sistema običnih diferencijalnih jednačina, jer imaju najširu oblast stabilnosti. Najbolji metod iz ove perspektive je implicitni Eulerov metod, međutim, red implicitnog Eulerovog metoda je mali i iznosi jedan.

Pojam relativne stabilnosti se takođe definiše koristeći sopstvene vrednosti matrice.

Definicija 7.11 *Neka su $\lambda_i \in \mathbb{C}$ sopstvene vrednosti matrice A i neka su prvi i drugi karakteristični polinomi linearnog višekoračnog metoda ρ i σ, redom. Linearni višekoračni metod je relativno stabilan ako i samo ako za svaku sopstvenu vrednost polinom $\pi(\psi, \lambda_i h) = \rho(\psi) - \lambda_i h\sigma(\psi)$ ima prostu nulu koja je najveća po modulu.*

Najčešća primena izloženih metoda za numeričku integraciju običnih diferencijalnih jednačina podrazumeva da je nezavisno promenljiva vreme. Zato se za Cauchyev problem kaže da predstavlja problem sa inicijalnom vrednošću. Evolucija sistema u vremenu je obično opisana zakonima dinamike, od kojih je najznačajniji drugi Newtonov zakon. Na primer, ako hoćemo da opišemo kretanje klatna, koje ima dužinu L i masu m, na osnovu drugog Newtonovog zakona dobijamo diferencijalnu jednačinu

$$m\frac{d^2\vec{r}}{dt^2} = \vec{Q} + \vec{T}, \quad \vec{r}(0) = \vec{r}_0, \quad \frac{d\vec{r}(0)}{dt} = \vec{v}_0,$$

gde smo sa $\vec{r}$ označili vektor položaja centra mase klatna, sa $\vec{Q}$ odgovarajuću gravitacionu silu i sa $\vec{T}$ silu zatezanja kanapa, za koji pretpostavljamo da je mase nula. Na osnovu učinjenih pretpostavki celokupna masa klatna je sadržana u kugli na kraju, pa je centar mase neka tačka u kugli. Pretpostavićemo da je kugla na kraju mala, idealno materijalna tačka. Vektori $\vec{r}_0$ i $\vec{v}_0$ predstavljaju inicijalni položaj i brzinu klatna. Letimičnim pogledom možemo utvrditi da je ovo jednačina drugog reda, i da je u pitanju sistem diferencijalnih jednačina, a ne skalarna jednačina. Već iz ovog prostog primera je jasno da su nam potrebni metodi za numeričku integraciju sistema diferencijalnih jednačina.

Vektorsku jednačinu kretanja klatna možemo da napišemo kao sistem dve skalarne jednačine

$$m\frac{d^2x}{dt^2} = Q_x + T_x, \quad m\frac{d^2y}{dt^2} = Q_y + T_y,$$

gde su Q_x, T_x i Q_y, T_y projekcije odgovarajućih sila na koordinatne ose.[1] Ako koordinatni sistem postavimo u tačku vešanja klatna, i ako x-osu usmerimo prema centru zemlje, dok y-osu usmerimo paralelno površini zemlje u smislu koordinatnog sistema desne orijentacije, dobijamo $Q_y = 0$. Ako sa θ označimo ugao koji zaklapa klatno sa pozitivnim delom x-ose, dobijamo

$$T_x = -||\vec{T}|| \cos\theta, \quad T_y = -||\vec{T}|| \sin\theta.$$

Sve vreme kretanja klatna važi identitet $x^2 + y^2 = L^2$. Ako ovu jednakost diferenciramo dva puta nalazimo

$$\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + x\frac{d^2x}{dt^2} + y\frac{d^2y}{dt^2} = v^2 + x\frac{d^2x}{dt^2} + y\frac{d^2y}{dt^2} = 0.$$

[1] Namerno smo izbegli upotrebu polarnih koordinata da bismo dobili što komplikovaniji sistem diferencijalnih jednačina.

Iz ove jednačine možemo da odredimo intenzitet sile zatezanja $||\vec{T}||$. Posle malo manipulacija jednačinama dobijamo

$$||\vec{T}|| = m\frac{v^2 + xg}{x\cos\theta + y\sin\theta} = m\frac{v^2 + xg}{L},$$

gde smo iskoristili $\cos\theta = x/L$ i $\sin\theta = y/L$. Na osnovu ovih jednačina sistem diferencijalnih jednačina postaje

$$\frac{d^2x}{dt^2} = g - \left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + xg\right)\frac{x}{L^2}, \quad \frac{d^2y}{dt^2} = -\left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + xg\right)\frac{y}{L^2}.$$

Svaku od ovih jednačina možemo predstaviti kao sistem jednačina prvog reda.

Ako uvedemo oznake $z_1 = x$, $z_2 = x'$, $z_3 = y$ i $z_4 = y'$, dobijamo Cauchyev problem

$$\begin{aligned}
z_1' &= z_2, \quad z_1(0) = x_0,\\
z_2' &= g - (z_2^2 + z_4^2 + gz_1)\frac{z_1}{L^2}, \quad z_2(0) = v_{x,0},\\
z_3' &= z_4, \quad z_3(0) = y_0,\\
z_4' &= -(z_2^2 + z_4^2 + gz_1)\frac{z_3}{L^2}, \quad z_4(0) = v_{y,0},
\end{aligned}$$

koji se može zapisati u vektorskoj formi

$$z' = F(x, z), \quad z(0) = z_0,$$

gde je $z = (z_1, z_2, z_3, z_4)$, $F(x, z) = (z_2, g - (z_2^2 + z_4^2 + gz_1)z_1/L^2, z_4, -(z_2^2 + z_4^2 + gz_1)z_3/L^2)$ i $z_0 = (x_0, v_{x,0}, y_0, v_{y,0})$.

Na primer, integracija prethodnog sistema jednačina, za klatno dužine jednog metra, upotrebom funkcije `ode113`, izgleda ovako

```
g=9.80665;
theta=pi/7;
fun=@(t,z) [z(2); g-(z(2)*z(2)+z(4)*z(4)+g*z(1))*z(1); z(4);
        -(z(2)*z(2)+z(4)*z(4)+g*z(1))*z(3)];
[t,z]=ode113(fun,[0 10],[cos(theta) 0 sin(theta) 0]);
```

Slika 7.10, desno, sadrži grafike rešenja sistema jednačina. Na slici su date normalizovane vrednosti brzine $v_x := v_x/\sqrt{2g(1-\cos\theta)}$ i $v_y := v_y/\sqrt{2g(1-\cos\theta)}$, gde je $\sqrt{2g(1-\cos\theta)}$ maksimalna vrednost intenziteta vektora brzine na osnovu zakona o održanju energije.

7.5.2 Metodi Runge-Kutta

Kao i linearni višekoračni metodi i metodi Runge-Kutta se mogu primeniti direktno na rešavanje sistema običnih diferencijalnih jednačina. Funkcije `ode23` i `ode45` se mogu koristiti za integraciju sistema običnih diferencijalnih jednačina metodom Runge-Kutta. Sve osobine koje smo posmatrali u skalarnom slučaju važe i u slučaju sistema diferencijalnih jednačina.

U ovom slučaju eksplicitne metode Runge-Kutta možemo opisati sledećim jednačinama

$$\begin{aligned}\Phi(x,z,h) &= \sum_{i=1}^{s} b_i k_i,\\ k_1 &= F(x_n, z_n),\\ k_i &= F\left(x_n + c_i h, z_n + h\sum_{j=1}^{i-1} a_{ij}k_j\right), \quad i = 2,\ldots,s,\\ c_i &= \sum_{j=1}^{i-1} a_{ij}, \quad i = 2,\ldots,s.\end{aligned}$$

Veličine k_i, $i = 1,\ldots,s$ su u ovom slučaju vektori, dok su c_i, $i = 2,\ldots,s$, i a_{ij}, $i = 1,\ldots,s$, $j = 1,\ldots,s$, i dalje brojevi. Na ovaj način svi prikazani metodi Runge-Kutta imaju primenu i na rešavanje sistema diferencijalnih jednačina u nepromenjenom obliku.

7.5.3 Stif sistemi diferencijalnih jednačina

U literaturi na srpskom jeziku još ne postoji opšte prihvaćen prevod pojma stif sistema diferencijalnih jednačina. Mi ćemo prihvatiti englesku reč stif, piše se stiff, i dalje je u tekstu koristimo kao reč preuzetu iz engleskog jezika. U matematičkoj literaturi još uvek ne postoji precizna definicija stif sistema diferencijalnih jednačina. Međutim, efekat se prepoznaje relativno jednostavno.

Pretpostavimo da treba rešiti diferencijalnu jednačinu koja opisuje oscilacije tela, mase m, zakačenog za oprugu krutosti k, čije se kretanje odvija u sredini koja ima koeficijent trenja μ. Na osnovu drugog Newtonovog zakona možemo napisati diferencijalnu jednačinu

$$m\frac{d^2x}{dt^2} = -\mu\frac{dx}{dt} - kx.$$

Ako podelimo jednačinu sa m, i uz smene $z_1 = x$, $z_2 = v$, $2b = \mu/m$ i $c = k/m$ možemo dobiti sistem diferencijalnih jednačina koji opisuje kretanje

$$z_1' = z_2, \quad z_2' = -cz_1 - 2bz_2. \tag{7.21}$$

Na ovom sistemu jednačina objasnićemo šta pojam stif sistem diferencijalnih jednačina znači.

Primetimo da je sistem jednačina linearan i da se može opisati na sledeći način

$$z' = Az, \quad A = \begin{bmatrix} 0 & 1 \\ -c & -2b \end{bmatrix}.$$

Koristeći ovu formu sistema linearnih jednačina možemo eksplicitno odrediti rešenja

$$x = z_1 = C_1e^{\lambda_1 t} + C_2e^{\lambda_2 t}, \quad v = z_2 = \lambda_1 C_1 e^{\lambda_1 t} + \lambda_2 C_2 e^{\lambda_2 t},$$

gde su $\lambda_1 = -b + \sqrt{b^2 - c}$ i $\lambda_2 = -b - \sqrt{b^2 - c}$ sopstvene vrednosti matrice A.

Radi ilustracije izaberimo $b = 100$, $c = 1$, $x_0 = 5$, $v_0 = 0$, $z_1(0) = 5$ i $z_2(0) = 0$. Sistem jednačina ćemo rešiti upotrebom funkcija `ode113` i `ode15s`. Sledeći skript rešava pomenuti sistem upotrebom ove dve funkcije

```
b=100; c=1; pos=[5 0];
fun=@(x,z) [z(2); -c*z(1)-2*b*z(2)];
disp('ode15s');
ops=odeset('RelTol',10^(-10),'Stats','on');
[xS,zS]=ode15s(fun,[0 10],pos,ops);
disp('ode113');
ops=odeset('RelTol',10^(-10),'Stats','on');
[xLVM,zLVM]=ode113(fun,[0 10],pos,ops);
```

Po izvršenju skripta dobijemo izlaz

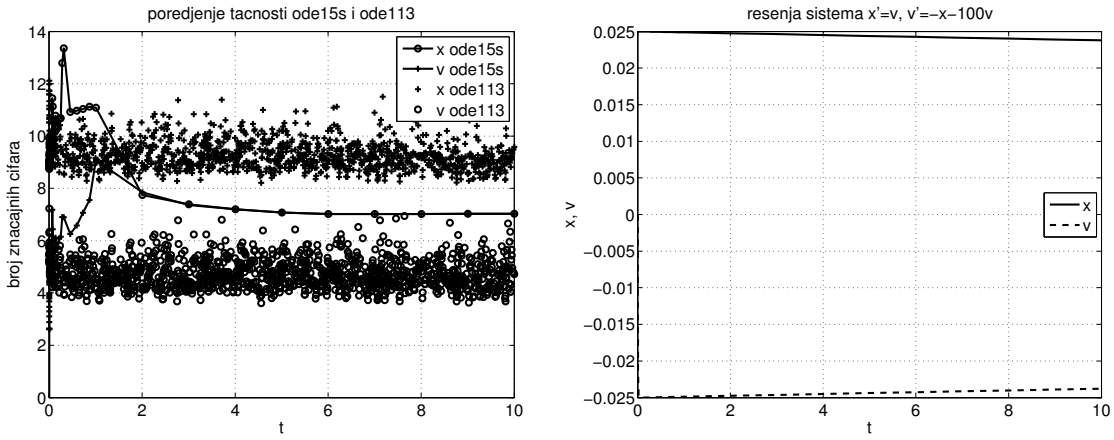
```
ode15s
73 successful steps
0 failed attempts
95 function evaluations
1 partial derivatives
15 LU decompositions
91 solutions of linear systems
ode113
1271 successful steps
177 failed attempts
2720 function evaluations
```

Možemo utvrditi da je funkcija `ode15s` znatno efikasnija. Vidimo da je potrebno svega 73 koraka i 95 izračunavanja funkcije da bi se dobilo rešenje sistema jednačina. Funkcija `ode113` mora da uradi 1271 korak i da izračuna funkciju 2720 puta. Slika 7.11, levo, ilustruje da se ovako velike razlike ne dobijaju zbog veće tačnosti funkcije `ode113`. Vidimo da brzina v nije određena sa 10 značajnih cifara, što je posledica činjenice da je vrednost opcije `AbsTol` podrazumevana i iznosi 10^{-6}. Slika 7.11, desno, ilustruje zavisnost položaja i brzine od vremena za posmatrani interval. Vrednosti brzine su prikazane normalizovane vrednošću 200. Kao što vidimo rešenja jednačine su prilično jednostavne funkcije, tako da do velikog broja koraka kod funkcije `ode113` ne dolazi ni zbog veoma kompleksne zavisnosti od vremena. Razlog znatno veće efikasnosti funkcije `ode15s` leži u znatno široj oblasti stabilnosti metoda sa diferenciranjem unazad.

Sopstvene vrednosti matrice sistema u ovom slučaju imaju vrednosti

$$\lambda_1 = -0.00500012500624791, \quad \lambda_2 = -199.994999874994,$$

što znači da je jedna sopstvena vrednost vrlo velika a druga relativno mala. Metod za numeričku integraciju sistema diferencijalnih jednačina će biti apsolutno stabilan ako je korak integracije odabran tako da su veličine $\lambda_1 h$ i $\lambda_2 h$ unutar oblasti apsolutne stabilnosti. Slika 7.7 prikazuje oblast apsolutne stabilnosti familije Adamsovih metoda koje implementira funkcija `ode113`. Vidimo da korak integracije mora biti odabran tako da važi $\lambda_2 h > -6$. Za grupu metoda sa diferenciranjem unazad sa slike 7.9, levo, vidimo da oblast apsolutne stabilnosti obuhvata sve negativne realne brojeve, tako da sopstvene vrednosti matrice A ne nameću nikakvo ograničenje na veličinu koraka integracije. Ove osobine omogućavaju relativno veliku vrednost koraka integracije kod funkcije `ode15s` i onemogućavaju izbor velike vrednosti koraka integracije kod funkcije `ode113`. Tako je funkcija `ode113` prinuđena da radi sa malim koracima ako želi da zadrži zahtevanu tačnost. Naravno, pojava je tim izraženija što je vrednost b veća.



Slika 7.11: Slika levo ilustruje broj značajnih cifara u rešenju sistema diferencijalnih jednačina (7.21), slika desno ilustruje grafik rešenja sistema jednačina (7.21), gde je vrednost brzine normalizovana sa 200.

U posmatranom sistemu diferencijalnih jednačina problem možemo da opišemo kroz veličinu količnika $|\lambda_2/\lambda_1|$. Naime, da rešenje samo obuhvata sopstvenu vrednost λ_1 mogli bismo da integralimo sa velikom vrednošću h, međutim, prisustvo sopstvene vrednosti λ_2 onemogućava izbor velikog koraka h kod Adamsove familije metoda. Mera neefikasnosti Adamsove familije metoda se onda može izraziti kao količnik po modulu najveće i najmanje sopstvene vrednosti, jer taj količnik meri sa kolikom vrednošću koraka bismo mogli da integralimo, a sa kolikom vrednošću smo primorani da integralimo.

Za sistem diferencijalnih jednačina $z' = Az$, $z(x_0) = z_0$, mera koja određuje stifness sistema se uobičajeno opisuje kao količnik $|\lambda_m(A)/\lambda_1(A)|$, gde su $\lambda_m(A)$ i $\lambda_1(A)$ po modulu najveća i najmanja sopstvena vrednost matrice A, redom. U slučaju sistema jednačina $z' = F(x, z)$, $z(x_0) = z_0$, stifness sistema se opisuje lokalno koristeći sopstvene vrednosti Jacobijeve matrice funkcije $F(x, z)$. Tako dolazimo do pojma oblasti u kojoj se sistem diferencijalnih jednačina ponaša kao stif sistem.

Važno je primetiti da stif sistemi mogu biti rešeni Adamsovom familijom metoda. Međutim, vreme potrebno za rešavanje u nekim slučajevima može postati neprihvatljivo dugo. Zato su metodi sa diferenciranjem unazad značajni u primenama. Primetimo i da odgovor na izložene probleme još uvek nema zadovoljavajuće rešenje, jer metodi sa diferenciranjem unazad postoje samo do reda 6, kako je prikazano u tabeli 7.9. Metodi višeg reda za stif sisteme diferencijalnih jednačina još uvek nisu pronađeni.

Metodi sa diferenciranjem unazad su implicitni metodi, tako da je neophodno rešavati sistem nelinearnih jednačina za određivanje člana niza y_n. Iz ovog razloga funkcija `ode15s` podržava opciju `'Jacobian'`. Opcija `'Jacobian'` ima vrednost podatka tipa `function_handle` koji predstavlja funkciju koja izračunava Jacobijevu matricu. Ulazni argumenti funkcije moraju biti nezavisno promenljiva i vektor promenljivih.

Pomenimo da `Matlab` implementira i funkciju `ode23s` koja koristi metode diferenciranja unazad nižeg reda prilikom rešavanja sistema običnih diferencijalnih jednačina. Ova funkcija se obično koristi kad je potrebno brzo proveriti da li neki sistem običnih direferencijalnih jednačina ima stif ponašanje. Takođe, postoje i funkcije `ode23t` i `ode23tb` koje su niskog reda, a mogu se koristiti

za stif sisteme jednačina.

Pomenimo da se metodi Runge-Kutta, primenjeni na stif sisteme diferencijalnih jednačina, ponašaju gore od Adamsove familije metoda, jer imaju još manje oblasti stabilnosti.

7.6 Uticaj tačnosti ulaznih podataka

Posmatrajmo Cauchyev problem

$$z' = F(x, z), \quad z(x_0) = z_0.$$

U ovom poglavlju dajemo odgovor na pitanje kako se tačnost ulaznih podataka, pre svega početnog uslova $z(x_0) = z_0$, reflektuje na tačnost rešenja Cauchyevog problema u nekom trenutku $x > x_0$.

Posmatrajmo Cauchyev problem $y' = 10(y + x)$, $y(0) = -1/10$. Opšte rešenje diferencijalne jednačine $y' = 10(y + x)$, koja se može zapisati kao $y' - 10y = 10x$, možemo dobiti relativno jednostavno. Naime, kako je poznato iz osnovnih kurseva matematike, opšte rešenje diferencijalne jednačine je određeno sa

$$y = e^{-\int(-10)dx}\left(C + \int (10x)e^{\int(-10)dx}dx\right) = -x - \frac{1}{10} + Ce^{10x}.$$

Partikularno rešenje koje zadovoljava početni uslov $y(0) = -1/10$, dakle, rešenje Cauchyevog problema, iznosi

$$y = -x - \frac{1}{10}.$$

Slika 7.12, levo, sadrži zavisnost broja značajnih cifara rešenja diferencijalne jednačine od x, koja je dobijena upotrebom funkcije `ode113`. Različiti grafici odgovaraju različitim tačnostima ulaznih podataka. Kao što vidimo podaci imaju relativne greške 10^{-5}, 10^{-10} i 2^{-53}. Skript koji je korišćen za izračunavanje izgleda ovako

```
fun=@(x,y) 10*(y+x);
ops=odeset('RelTol',10^(-12),'AbsTol',10^(-20));
[x,y]=ode113(fun,[0 5],-1/10,ops);
[x10,y10]=ode113(fun,[0 5],-1/10*(1+10^(-10)),ops);
[x5,y5]=ode113(fun,[0 5],-1/10*(1+10^(-5)),ops);
```

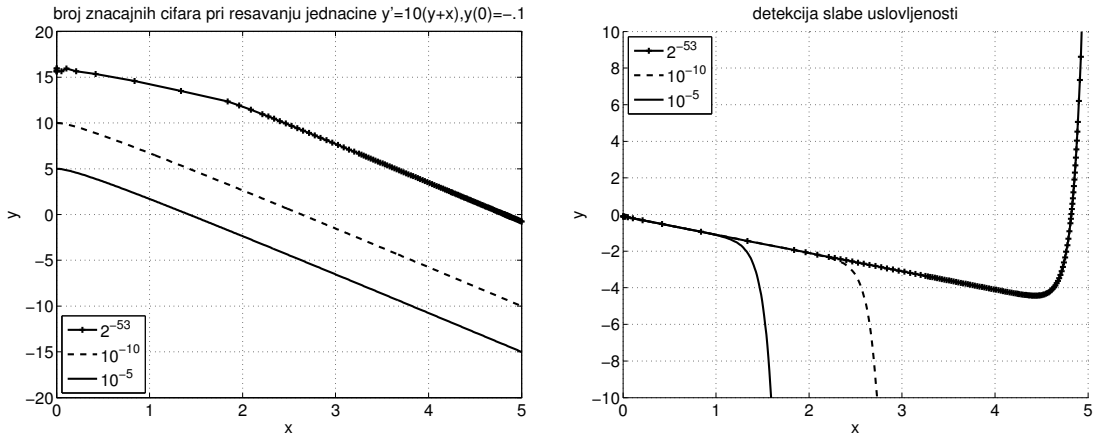
Zahtevana relativna greška je velika, tako da gubitak značajnih cifara nije posledica činjenice da smo izračunavanja izvršavali sa premalom vrednošću opcije `RelTol`. Mi intuitivno očekujemo da broj značajnih cifara u rešenju jednačine ne opada sa porastom x, barem ne ovako drastično i barem ne za ovako jednostavne diferencijalne jednačine. Međutim, postoji valjani razlog za ovakav gubitak značajnih cifara.

Pretpostavimo da je početna vrednost $y(0)$ zadata sa greškom i da umesto vrednosti $-1/10$ iznosi $-1/10 + r$. Onda je konstanta C u opštem rešenju diferencijalne jednačine određena iz jednačine

$$-\frac{1}{10} + C = -\frac{1}{10} + r.$$

Dakle, vrednost konstante C iznosi r, pa je opšte rešenje Cauchyevog problema određeno sa

$$y = -x - \frac{1}{10} + re^{10x}.$$



Slika 7.12: Slika levo ilustruje gubitak značajnih cifara u numeričkom rešenju Cauchyevog problema usled slabe uslovljenosti početnim uslovom, slika desno ilustruje mogućnost detekcije slabo uslovljenih Cauchyevih problema.

Kao što vidimo pored pravog rešenja $-x - 1/10$ javlja se i takozvano parazitno rešenje re^{10x}. Kako se udaljavamo od koordinatnog početka, bez obzira koliko mala vrednost za r bila, u jednom trenutku će parazitno rešenje preovladati, jer pravo rešenje mnogo sporije raste. Vrednost x_c, kad pravo rešenje prestaje da dominira i kad parazitno rešenje postaje dominantno, možemo odrediti iz jednačine

$$x_c + \frac{1}{10} = |r|e^{10x_c}.$$

Odavde dobijamo

$$x_c = \phi(x_c) = \frac{1}{10}\log\frac{10x_c+1}{10} - \frac{1}{10}\log|r|.$$

Ako pretpostavimo da važi $|r| \leq 10^{-3}$, jednostavno zaključujemo da važi $\phi : [.5, 5] \to [.5, 5]$, jer je za $|r| = 10^{-3}$, $\phi(.5) \approx .87 > .5$, $\phi(5) \approx 1.08 < 4$ i za $|r| = 2^{-54}$, $\phi(.5) \approx 3.92 > .5$ i $\phi(5) = 4.14 < 5$ i $\phi'(x_c) = 1/(10x_c + 1) > 0$. Takođe je $|\phi'(x_c) \leq 1/6 = q < 1$ za $x_c \in [.5, 5]$. Jednačinu možemo rešiti metodom proste iteracije. Za $r = .1 \cdot 10^{-5}$, $r = .1 \cdot 10^{-10}$ i $|r| = .1 \cdot 2^{-54}$ tačke su određene sa $x_{c,5} \approx 1.42$, $x_{c,10} \approx 2.63$ i $x_{c,2^{-54}} \approx 4.12$. Slika 7.12, desno, jasno ilustruje da se grafici rešenja sa različitim tačnostima ulaznih podataka razilaze upravo u ovim tačkama. Vidimo i da je greška odsecanja broja .1 u dvostrukoj preciznosti pozitivna, dok je za brojeve $.1\cdot(1+10^{-5})$ i $.1\cdot(1+10^{-10})$ ova greška negativna. Ovo se da zaključiti na osnovu znaka parazitnog rešenja kad x pređe vrednost tačke x_c.

Slika 7.12, desno, daje ideju kako detektovati Cauchyeve probleme sa slabom uslovljenošću. Naime, dovoljno je Cauchyev problem rešiti tri puta, sa različitim greškama u početnom uslovu. Ako je Cauchyev problem slabo uslovljen, grafici rešenja sa manjim tačnostima početnih uslova će u različitim tačkama početi da odstupaju od rešenja koje je određeno sa najmanjom greškom u vrednosti početnog uslova.

Primetimo jednu vrlo bitnu osobinu posmatranog Cauchyevog problema. To je zavisnost između tačke x_c i r. Naime, ako želimo da numeričko rešenje ima smisla do tačke x_c ulazni podaci moraju imati gornju granicu apsolutne greške $(x_c + 1/10)e^{-10x_c}$. Primetimo da je to osobina Cauchyevog problema i da nema nikakve veze sa primenjenim numeričkim algoritmom.

Naravno, u prethodnom primeru za ilustraciju je izabrana najgora moguća početna vrednost. Za ostale početne vrednosti situacija bi bila mnogo povoljnija i ne bi dolazilo da ovolikog gubitka značjanih cifara.

Veza izmedju tačnosti početnih uslova i tačnosti izračunate vrednosti se u najopštijem slučaju može opisati sledećom nejednakošću

$$||y(x) - \tilde{y}(x)|| \leq ||y(x_0) - \tilde{y}(x_0)||e^{L(x-x_0)},$$

gde su y i $\tilde{y}$ dva rešenja istog Cauchyevog problema sa različitim početnim uslovima $y(x_0)$ i $\tilde{y}(x_0)$, a L je Lipschitzova konstanta funkcije f. U slučaju da posmatramo sistem diferencijalnih jednačina Lipschitzova konstanta i uslov funkcije F se definišu analogno korišćenjem norme

$$||F(x, y) - F(x, \tilde{y})|| \leq L||y - \tilde{y}||.$$

Dalja razmatranja u ovom smeru vode ka teoriji stabilnosti i posebno teoriji haosa.

Literatura

[1] Matlab Programming Fundamentals, version 2012a, The MathWorks, Inc, 2012.

[2] Aleksandar S. Cvetković, Slobodan Lj. Radojević, Matlab I, Mašinski fakultet, Univerzitet u Beogradu, 2012.

[3] Nickolas J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 2002.

[4] Gene H. Golub, Charles F. Van Loan, Matrix Computation, JHU Press, 1996.

[5] Boško S. Jovanović, Numerička analiza, Matematički fakultet, Beograd, 2003.

[6] Gradimir V. Milovanović, Numerička analiza, prvi deo, Naučna knjiga, Beograd, 1991.

[7] Gradimir V. Milovanović, Numerička analiza, drugi deo, Naučna knjiga, Beograd, 1991.

[8] Gradimir V. Milovanović, Numerička analiza, treći deo, Naučna knjiga, Beograd, 1991.

[9] Cleve Moler, Experiments with Matlab, MathWorks, 2011.

[10] Miodrag Spalević, Miroslav Pranić, Numeričke metode, Prirodno-matematički fakultet u Kragujevcu, 2007.

[11] Alfio Quarteroni, Fausto Saleri, Scientific Computin with Matlab, Springer-Verlag, Berlin Heidelberg, 2003.

Indeks

-Inf, 20, 45, 113
    -inf, 20
Čebiševljev sistem, 186
    algebarski, 188
    eksponencijalni, 188
    hiperbolički, 188
    karakterizacija, 186
    Müntzov, 188
Čebiševljevi diskretni ortogonalni polinomi, 198
Čebiševljevi polinomi prve vrste, 137

A/D konverzija, 11
accuracy, 85
algoritam Higham i Tisseur, 65
apsolutna greška
    izračunavanje funkcija, 31
    matrica, 41
    na osnovu zapisa broja, 12
    realnih brojeva, 2
    vektora, 39
Avogadrov broj, 5

backward differentiation formulas, 288
backward substitution, 47
Besselov interpolacioni polinom, 161, 163
    implementacija, 165
    inkrementalna konstrukcija, 164
besselPoly, 165
bisekcijaR, 80–85

Cauchyev problem, 261, 302
    egzistencija rešenja, 262
    linearan, 261
    vektorska forma, 294
      linearan, 295
      linearan, analitičko rešenje, 296
    višeg reda, 294
cauchyProizvod, 233
chebyshevOrthMatrix, 201
chebyshevOrthPoly, 201
cond, 65
    'fro', 65
    1, 65
    2, 65
    inf, 65
condest, 65
condition number, 64
csape, 181–185
    complete, 183
    not-a-knot, 184
    periodic, 185
    second, 181, 182
    variational, 181

dec2bin, 232
diff, 147, 148, 150, 156, 157, 164, 165, 220, 221, 223, 225, 230, 231, 234, 235
disp, 28, 76, 77, 81, 84, 92, 104, 109, 120, 245, 250, 266, 300
dobra uslovljenost, 34, 35, 66, 68
double, 13, 15, 16, 48
double precision, 49
drugi Newtonov interpolacioni polinom, 153, 154
    inkrementalna konstrukcija, 156
    osetljivost na promenu ulaznih podataka, 157
dvostranoKoeficijenti, 232–234
dvostruka preciznost, 13, 15–17, 19, 20, 23, 26–30, 33, 66
    beskonačnost, 19, 20
    bit znaka, 13
    broj značajnih cifara, 16
    brojevi sa subnormalnom mantisom, 18
    eksponent, 14
    eksponent pomeraj, 14

eps, 16
mantisa, 13
najmanji pozitivan reprezentabilan broj, 18, 19
najveći pozitivan reprezentabilan broj, 20
relativna greška, 15

eps, 16, 17, 28, 113
eps('single'), 16
error, 47, 80, 81, 83, 84
eulerMethod, 262
eulerMethodImplicit, 265
eulerMethodVector, 295
Eulerov metod
apsolutna stabilnost, 269
interval, 269
eksplicitni, 262, 267, 274
greška, 264
implementacija, 262
vektorska forma, 295
vektorska forma, implementacija, 295
implicitni, 264, 275, 281
greška, 265
implementacija, 265
prosta iteracija, 265
prediktor-korektor, 267
expand, 129, 130
expLose, 28

faktor uslovljenosti, 33, 34, 63–69, 135, 142, 195, 196, 210, 238
dobra uslovljenost, 34, 35, 66, 68
ill-conditioned, 34, 68
konstrukcije kvadraturnih formula, 238
matrice, 63–68, 238
cond, 65
condest, 65
norma, 1, 65
simetrične, 64
subordinisane norme, 64
promena baze polinoma, 106
slaba uslovljenost, 34, 35, 66, 68, 106, 202, 303
well-conditioned, 34, 68
značajne cifre, 34
faktorizacija polinoma, 105
fiksni zarez, 4
format, 53, 234
long g, 53
rat, 234
forward elimination, 47
Frobeniusova matrica, 105
fsolve, 110, 111, 115
algorithm
levenberg-marquardt, 110, 111, 113
levenberg-marquardt, korekcija, 111
trust-region-dogleg, 110, 112–114
trust-region-dogleg, algoritam, 111
trust-region-dogleg, korekcija, 110
trust-region-reflective, 110, 111, 113, 117
exitflag, 111, 116, 117
jacobian, 111
optimset, 115
options, 111
Algorithm, 113
DerivativeCheck, 113, 116
Diagnostics, 113, 116
DiffMaxChange, 115
DiffMinChange, 113, 115
Display, 113, 116
FinDiffRelStep, 113, 116
FinDiffType, 113
FunTol, 115
FunValCheck, 113, 117
Jacobian, 113, 115
MaxFunEvals, 113, 117
MaxIter, 113, 117
OutputFun, 113
PlotFcns, 117
PlotFncs, 114
ScaleProblem, 111
TolFun, 114, 117
TolX, 114, 117
TypicalX, 114
output, 111
algorithm, 111
cgiterations, 111
firstorderopt, 111, 115
funcCount, 111
iterations, 111
message, 111, 114, 115
function handle, 254, 285, 301
fzero, 98–100, 102, 111
exitflag, 99, 100

options, 100
Display, 100
FunValCheck, 100
OutputFcn, 100
PlotFcns, 100
TolX, 100
output, 100
algorithm, 100
funcCount, 100
intervaliterations, 100
iterations, 100
message, 100, 102

Gauss-Christoffelove formule, 243, 245
Gaussova raspodela, 12
gaussovaEliminacijaR, 53, 58, 59, 69
gaussSeidelR, 76, 77
err, 77
maxIter, 77
gradijentni metod, 108, 110
implementacija, 109
iterativni proces, 109
korekcija, 109, 110
gradijentR, 109

Hagerov algoritam, 65
Hermiteov interpolacioni polinom, 166
čvorovi, 166
višestrukost, 166
čvorovi sa ponavljanjem, 169
greška, 171
implementacija
Lagrangeova forma, višestrukost čvorova dva, 175
Newtonova forma, 174
Lagrangeova forma, 172, 173
višestrukost čvorova, dva, 173
Newtonova forma, 170
podeljene razlike sa ponavljanjem, 170
hermitePolyNewton, 174
hex2num, 15, 17, 18
hilb, 69
Hilbertova matrica, 68, 69

IEEE-754, 13, 17
binary32, 13
binary64, 13
ne asocijativnost množenja, 24
ne asocijativnost sabiranja, 25
specijalne vrednosti, 17
ill-conditioned, 34, 68
inženjerska notacija, 12
Inf, 20, 99–101, 113, 260
inf, 20
integral, 254–258, 260
neograničeni integrand, 257
neograničeni interval integracije, 259
opcije, 254, 255
AbsTol, 255, 256, 258, 260
ArrayValued, 255
RelTol, 255–258, 260
Waypoints, 255
oscilatorni integrand, 255
interp1, 179, 184
linear, 179
splajn, 184
interpolacija
čvorni polinom, 136
algebarska, 125
algebarski
matrični metodi, 140
Besselova, 163
deo po deo linearna, 178
Hermiteova, 166
Lagrangeov bazis, 171, 172
Lagrangeov bazis, 126–128, 134, 135, 142, 173, 176, 238, 275
Lagrangeova, 126
Newtonova, 143
čvorni polinom, 143
ekvidistantni čvorovi, 152, 153
splajn, 178
Stirlingova, 161
trigonometrijska, 189
Lagrangeov bazis, 189
uopštena, 185, 187
interpolacioni čvorovi, 125
Čebiševljevi, 138
Čebiševljevi modifikovani, 139
Hermiteovi, 166
splajn, 178
trigonometrijski, 189
uopšteni polinomi, 186
interpolacioni polinom, 125
interpolant, 125

interpolant, 125
  algebarski, 125
    matrični metodi, 140
  Besselov, 161
    implementacija, 165
  drugi Newtonov, 153
    implementacija, 156
  greška, 133
  Hermiteov, 166
    Lagrangeova forma, 172, 173
    Newtonova forma, 170
  Lagrangeov, 126, 127
    Čebiševljevi čvorovi, 138
    implementacija, 128
    Lebesgueova funkcija, 131
    ukupna greška, 136
  Newtonov, 143, 144
    implementacija, 148
  prvi Newtonov, 152
    implementacija, 156
  Stirlingov, 161
    implementacija, 164
  trigonometrijski, 189
  uopšteni, 185, 187
iterativni metodi
  acc, 83, 84
  faktorizacija polinoma, 105
  fzero, 98
  Matlab implementacija, 98
  maxIter, 84
  Newtonov metod, 94
  polovljenje intervala, 79
  prec, 82, 84
  prosta iteracija, 85
  sistemi jednačina
    fsolve, 110
    gradijentni metod, 108
    Matlab implementacija, 110
    Newton-Kantorovič, 102
    prosta iteracija, 117
  uslov za prekid iteracija, 83
iterativni proces, 71–73, 82, 83, 85, 88, 89, 91–98, 103–105, 109, 118, 265, 282
izvodDesni, 220
izvodDvostraniMedjupodela, 223
izvodDvostrano, 224, 226
izvodDvostranoK, 234
izvodJednostranoK, 230
izvodJednostranoNablaK, 231
izvodLagrangeInterpolant, 215
izvodLevi, 221, 226

Jacobijeva matrica, 103, 104, 108, 110, 111, 113, 115–117, 121, 301
jacobiR, 76, 78
jednostruka preciznost, 13–16, 20, 26, 27
  beskonačnost, 20
  bit znaka, 13
  broj značajnih cifara, 15
  brojevi sa subnormalnom mantisom, 18
  eksponent, 14
  eksponent pomeraj, 14
  eps('single'), 16
  mantisa, 13
  najmanji pozitivan reprezentabilan broj, 18, 19
  najveći pozitivan reprezentabilan broj, 20
  relativna greška, 15

koeficijentiIzvod, 233
koeficijentiMedjupodela, 232
kompleksnost, 192, 197, 206, 281
kontrakcija, 86, 117, 118, 121
kron, 210

Lagrangeov interpolacioni polinom, 127–129, 134, 135, 142, 143, 146, 148–150, 157–159, 161, 167, 168, 171, 176, 214, 238
  Čebiševljevi čvorovi, 138, 148
  implementacija, 128
  izvod, 214
  Lebesgueova funkcija, 131
    minimizacija, 139
    implementacija, 132
    modifikovani Čebiševljevi čvorovi, 139
  osetljivost na tačnost ulaznih podataka, 131
  ukupna greška, 136
lagrangePoly, 128–130, 132, 142
lebesgueFunction, 132
Lebesgueova funkcija, 131, 132, 136
  implementacija, 132
  minimizacija, 139
  modifikovani Čebiševljevi čvorovi, 139

linearna regresija, 210
F statistika, 211, 212
p vrednost, 211
interval pouzdanosti ostatka, reziduala, 211
interval pouzdanosti parametra, 211
izlaz, 210
koeficijent determinisanosti, R^2 statistika, 211, 212
neobjašnjena varijansa, 212
opservacija, 210, 211
ostatak, rezidual, 211
outlier, 211
prediktor, 210, 211
problem, 211
rešenje, 211
srednja vrednost izlaza, 212
suma kvadrata ostataka, reziduala, 211
varijansa greške, 211, 212
linearni višekoračni metod, 270
Adams-Bashfort, 274
Adams-Moulton, 274
Crank-Nicolson, 281
Adamsova familija, 274, 302
konstrukcija upotrebom kvadraturnih formula, 276
Matlab implementacija, 285
ode113, 285
pregled, 276
analiza greške, 277
apsolutna stabilnost, 279
asimptotska konstanta greške, 271, 276, 284
definicija konvergencije, 272
drugi karakteristični polinom, 273
interval apsolutne stabilnosti, 279
interval relativne stabilnosti, 279
karakterizacija konvergentnih metoda, 273
konzistencija, 272
lokalna greška odsecanja, 271
metodi sa diferenciranjem unazad, 288
Milne-Simpson, 274
konstrukcija upotrebom kvadraturnih formula, 276
pregled, 276
nula stabilnost, 273
Nystrom, 274
konstrukcija upotrebom kvadraturnih formula, 276
pregled, 276
oblast apsolutne stabilnosti, 281
optimalan, 275
prediktor-korektor, 272, 282
$P(EC)^m$, 283
$P(EC)^mE$, 283
$PM(EC)^mEM$, 284
$PM(EC)^mM$, 284
asimptotska konstanta greške, 284
Matlab implementacija, 285
Milneova modifikacija, 284
oblast stabilnosti, 284
prosta iteracija, 282
red, 284
prvi karakteristični polinom, 273
red, 271
relativna stabilnost, 279
Simpsonov metod, 271
vektorska forma, 294
apsolutna stabilnost, 296
apsolutna stabilnost, oblast apsolutne stabilnosti, 296
relativna stabilnost, 297
linsolve, 63, 66, 68, 69, 77, 104, 142, 237
opcije, 63
LT, 63, 77
POSDEF, 64
RECT, 64
SYM, 69
SYMM, 63
TRANSA, 64
UHESS, 63
UT, 63
linspace, 130, 132, 149
Lipschitz
konstanta, 262, 277, 292
uslov, 261, 262
lu, 62, 63

mašinska greška odsecanja, 26, 27, 48
mašinska preciznost, 17, 24, 28, 49, 57, 62, 66, 68, 90, 138, 150, 151, 159, 215, 219
machine epsilon, 17
machine precision, 17

metod polovljena intervala, 79
algoritam, 80
broj iteracija, 80
implementacija, 80
metod proste iteracije, 85, 94
asimptotska konstanta greške, 88
implementacija, 91
iterativna funkcija, 85
iterativni proces, 85
konvergencija, 86
prinos značajnih cifara po iteraciji, 88, 90, 94
red konvergencije, 89, 91
sistemi jednačina, 117
implementacija, 119
konvergencija, 118
metod Runge-Kutta, 290, 302
Bogacki-Shampine, 293
Butcherova tabela, 291
definicija konvergencije, 290
Dormand-Prince, 293
Euler-Cauchyev, 291
poboljšani, 291
Gillov, 292
greška, 292
Heunov, 291
interval stabilnosti, 292
konvergencija, 291
konzistencija, 290
Matlab implementacija, 294
ode23, 293
ode45, 293
red, 290
vektorska forma, 299
metodi sa diferenciranjem unazad, 289
konzistencija, 290
nula stabilnost, 290
ode15s, 290
pregled, 288
red, 290

najmanji kvadrati
ekvidistantne apscise, 199
rešenje, 199
suboptimalno rešenje, 202, 203
faktor uslovljenosti, 195
implementacija, 194
Matlab implementacija, 197
normalizovana greška, 196, 201, 202
polyfit, 197
postavka problema, 192
problem sa trendom, 206, 207
linearizacija, 207
rešenje, 193
sa težinama, 203
implementacija, 204
osetljivost konstrukcije, 204
problem, 203
rešenje, 203
uopšteni polinomi, 208
faktor uslovljenosti, 209, 210
problem, 208
rešenje, 208
najmanjiKvadrati, 204, 207
najmanjiKvadratiEkvidistant, 200, 201
najmanjiKvadratiTezine, 204
NaN, 20, 21, 98–101, 113, 260
nan, 20
naučna notacija, 12
Newton-Cotesove formule, 239–248, 252
$n = 5$, 243
$n = 6$, 243
$n = 7$, 244
Booleovo pravilo, 241
divergencija, 252
greška, 242
implementacija, 244
Newton-Cotesovi brojevi, 239, 240, 242
Simpsonovo pravilo, 241
Simpsonovo pravilo 3/8, 241
težine, 239
trapezno pravilo, 241
uopštene, 245
greška, 247
implementacija, 246
Simpsonova formula, 247
Simpsonova formula, greška, 247
trapezna formula, 245, 253
trapezna formula, greška, 246
trapezna formula, nejednaki intervali, 248
trapezna formula, operator T_N, 246
uticaj tačnosti ulaznih podataka, 252
Newton-Kantorovič, 102, 107, 110

implementacija, 104
iterativni proces, 103–105
korekcija, 103, 110
modifikovan metod, 122
red konvergencije, 103
uslovi konvergencije, 103
newtonCotes, 243, 244, 246, 247
newtonCotesOtvoren, 250, 251
newtonDrugiPoly, 156
newtonKantorovichR, 104
Newtonov interpolacioni polinom, 143–147, 149–153, 157–159, 161, 169–171, 174, 176
analiza greške, 150
implementacija, 148
podeljene razlike, 144
ekvidistantni čvorovi, 152
prostiranje greške, 149
tabela podeljenih razlika, 145
Newtonov metod, 94–98, 102, 103
geometrijska interpretacija, 95
ilustracija divergencije, 98
iterativna funkcija, 94
iterativni proces, 94, 95, 97
modifikovan, 122
red konvergencije, 95
višestruke nule, 97
kvadratna konvergencija, 97
linearna konvergencija, 97
Newtonova konstanta gravitacije, 5
newtonPoly, 148
newtonPrviPoly, 156
norm, 44, 65, 77, 105, 107, 195, 205
'fro', 44, 45
-inf, 45
1, 44
2, 44, 45, 48, 65–67, 69
inf, 44, 45
norma
intenzitet vektora, 37
matrica
+∞, 42, 44, 73
1, 42, 44, 73
2, 42, 44, 73
Euklidova, 43
Frobeniusova, 41–44, 73
homogenost, 40
Matlab implementacija, 44
multiplikativnost, 40
nenegativnost, 40
nula, 40
prva nejednakost trougla, 40
saglasna normi vektora, 40
spektralna, 42
subordinisana normi vektora, 40, 41
vektora, 191
+∞, 38
1, 38
2, 38
druga nejednakost trougla, 39
ekvivalentnost, 40
Euklidova, 37, 38, 192
homogenost, 38
Matlab implementacija, 45
nenegativnost, 37
nula, 37
p, 38
prva nejednakost trougla, 38
not a number, 20
num2hex, 15, 17–21, 26, 60
numerička integracija, 235
čvorovi, 236
interpolacione formule, 238
kvadraturna formula, 236
algebarski stepen tačnosti, 237
kvadraturne formule
interpolacione, 238
Newton-Cotesove, 240
otvorene Newton-Cotesove, 248
uopštene Newton-Cotesove, 245
uopštene otvorene Newton-Cotesove, 250
Matlab implementacija, 254
momenti, 237
problemi, 253
težine, 236
numeričko diferenciranje, 214
dvostrano, 222–225
greška, 225
implementacija, 224
izvod u tačkama međupodele, 222
izvod u tačkama međupodele, implementacija, 223
izvod višeg reda, 232, 234
izvod višeg reda, implementacija, 234
razvijena forma, 225, 235

jednostrano, 216, 217, 219–221, 228
operator Δ, 217
operator ∇, 220
Lagrangeov interpolacioni polinom, 214
greška, 216
implementacija, 215
operator Δ, 217
greška, 217
implementacija, 220
izvod višeg reda, 228
izvod višeg reda, implementacija, 230
razvijene formule, 221
operator ∇, 220
greška, 220
implementacija, 221
izvod višeg reda, 230
izvod višeg reda, implementacija, 231
razvijene formule, 221

ode113, 284–287, 294, 298–300, 302
opcije, 286
AbsTol, 287
InitialStep, 287
MaxStep, 287, 288
NormControl, 287
Refine, 288
RelTol, 287, 288
Stats, 286
ode15s, 290, 299–301
opcije
Jacobian, 301
ode23, 293, 294, 298
opcije, 294
ode23s, 301
ode23t, 301
ode23tb, 301
ode45, 293, 294, 298
opcije, 294
Refine, 294
odeset, 286, 287
operator 1, 159
operator 1, E, Δ, ∇, μ, δ
komutacija, 159
operator \, 62, 194
operator Δ, 151–155, 159–164, 216, 217, 221–225, 227, 228, 230, 232
stepen, 151
operator δ, 159–161, 163, 222–224, 231, 232, 235
stepen, 159
operator μ, 159, 160, 163, 222–224, 231, 235
operator ∇, 151, 153, 155, 159, 219–221, 230, 231
stepen, 151
operator D, 216, 219, 222, 224, 227, 228, 230–233, 235
stepen, 216
operator D_n^+, 217–220
operator D_n^-, 220
operator E, 159, 160, 216, 219, 221, 232, 235
stepen, 159
operator M_N, 250, 251
operator T_N, 245, 246, 253
optimset, 115, 116
otvorene Newton-Cotesove formule, 248
$n = 3$, 249
formula srednje tačke, 248
greška, 249
implementacija, 249
Milneova formula, 249
težine, 248
trapezna formula, 249
uopštene, 250
formula srednje tačke, 250
formula srednje tačke, greška, 250
formula srednje tačke, operator M_N, 250
greška, 252
implementacija, 251
otvorenNewtonCotes, 249

Planckova konstanta, 5, 12, 13
plot, 69, 129, 130, 132, 150, 179, 182, 210
podeljene razlike, 144–146, 149, 150, 152, 169
sa ponavljanjem, 169, 170
pokretni zarez, 5, 13
eksponent, 5
mantisa, 5
normalizovan zapis, 5
poly, 105–107
polyfit, 197
polyval, 142, 195, 197, 205
ppval, 181–185
precision, 85

prostaIteracija, 91, 92
prostaIteracijaVektor, 119
prvi Newtonov interpolacioni polinom, 152, 154
  inkrementalna konstrukcija, 155
  osetljivost na promenu ulaznih podataka, 157

rand, 129, 132, 142, 149, 207, 210, 213, 215, 226, 254
randSin, 226
rastojanje
  matrica, 41
    identifikacija, 41
    korišćenjem norme, 41
    nenegativnost, 41
    prva nejednakost trougla, 41
    simetričnost, 41
  realnih brojeva, 1
    druga nejednakost trougla, 2
    identifikacija, 1
    osobine, 1
    prva nejednakost trougla, 1
    simetrija, 1
  vektora
    $+\infty$, 39
    1, 39
    Euklidovo, 38
    identifikacija, 38
    korišćenjem norme, 38
    nenegativnost, 38
    p, 39
    prva nejednakost trougla, 38
    simetričnost, 38
realmax, 20
realmax('single'), 20
realmin, 19
realmin('single'), 19
regress, 211, 213
relativna greška
  aritmetičke operacije, 22
  faktor uslovljenosti, 34
  izračunavanje funkcija, 31
  mašinska greška odsecanja, 27
  matrica, 41
  merenje vremena atomskim satovima, 17
  na osnovu zapisa broja, 12
  osetljivost, 32
  približna, 3
  realnih brojeva, 2
  vektora, 39
relativna standardna neizvesnost, 13
roots, 102, 105–107
Rydbergova konstanta, 13

significand, 13
single, 13, 15, 16, 18–20, 59
single(-inf), 20
single(inf), 20
sistem linearnih jednačina
  backward substitution, 47
  dobro uslovljen, 68
  donje trougaoni
    algoritam, 47
    implementacija, 47
  faktor uslovljenosti, 63, 64, 66, 67, 69
  forward elimination, 47
  Gauss-Seidelov metod, 75
    dominantna po vrstama, 75
    implementacija, 77
    simetrična i pozitivno definitna, 75
    uslov konvergencije, 75
  Gaussova eliminacija, 49
    algoritam, 51
    greška, 57
    greška sa izborom glavnog elementa, 62
    implementacija, 53
    izbor glavnog elementa, 57
    Matlab implementacija, 63
    permutacione matrice, 58
  gornje trougaoni, 46
    algoritam, 47
    implementacija, 47
  ill-conditioned, 68
  iterativni metodi, 71
    broj cifara po iteraciji, 73
    Gauss-Seidelov metod, 75
    iterativni proces, 71–73
    Jacobijev metod, 74
    kriterijum za prekidanje konvergencije, 75
    spektralni radijus, 73
  Jacobijev metod, 74

dominantnost po vrstama, 74
implementacija, 76
uslovi konvergencije, 74
LU faktorizacija, 54
algoritam, 56
greška, 57
Matlab implementacija, 62
permutacione matrice, 60, 61
totalni izbor glavnog elementa, 61
slabo uslovljen, 68
trougaoni
granica greške, 49
well-conditioned, 68
slaba uslovljenost, 34, 35, 66, 68, 106, 202, 303
spektralni radijus, 42
splajn, 177
kubni, 179
interpolacioni uslovi, 179
kompletni, 183
not-a-knot, 183
periodični, 185
prirodni interpolacioni, 181
prirodni interpolacioni, ekstremalno svojstvo, 182
sistem jednačina, 180
uslovi, 179
stepena 1, reda m, 178
spline, 183, 184
standard uncertainty, 13
standardna neizvesnost, 13
stif sistem diferencijalnih jednačina, 299
mera, 301
stirling, 228, 230, 231
Stirlingov interpolacioni polinom, 161
implementacija, 164
inkrementalna konstrukcija, 162
Stirlingovi brojevi, 227–231
implementacija, 228
stirlingPoly, 164
subs, 142
sym, 48
syms, 129, 130, 142

tabela podeljenih razlika, 145, 146
sa ponavljanjem, 170
tablica konačnih razlika, 153–155, 161–164, 217–219, 223, 224
operator ∇, 153, 155
prostiranje greške, 157
testFSolve, 115
trapz, 254
trougaoniSistemR, 47, 48
smer, 48
'unapred', 48
'unazad', 48

uopšteni interpolacioni polinom, 185, 187
Čebiševljev sistem, 186
eksponencijalni, 188
eksponencijalni bazis, 185
hiperbolički, 188
Müntzov, 188
Müntzov bazis, 186
trigonometrijski, 188, 189
Lagrangeov bazis, 189
Lagrangeova forma, 190
trigonometrijski bazis, 185
uslovi, 186
uopstenNewtonCotes, 246, 247
uopstenNewtonCotesOtvoren, 251, 252

vander, 141, 142, 237
Vandermondova matrica, 141, 142, 193, 194, 197, 198, 237, 238
determinanta, 188
faktor uslovljenosti, 135, 142, 238
Vietove formule, 107

well-conditioned, 34, 68
Wilkinsonov polinom, 106

zaokruživanje brojeva, 6
pravila, 6
prilikom smeštanja u memoriju, 26
značajne cifre, 3, 6, 8, 10, 302
$\log_{10}$, 8
A/D konverzija, 11
aritmetičke operacije, 22
faktor uslovljenosti, 34
gubitak prilikom oduzimanja bliskih brojeva, 24
mašinska greška odsecanja, 27
proizvoljna brojna osnova, 11
značajne cifre u užem smislu, 10
$\log_{10}$, 10