Desanka P. Radunović
Aleksandar B. Samardžić
Filip M. Marić

N U M E R I Č K E M E T O D E

Zbirka zadataka
kroz C, Fortran i Matlab

AKADEMSKA MISAO
Beograd, 2005

Predgovor

Knjiga predstavlja drugo, izmenjeno i dopunjeno izdanje iste zbirke koju je pod ovim naslovom prvi autor objavio 2000. godine. S obzirom da je prvo izdanje rasprodato, iskoristili smo priliku da dopunimo zbirku detaljnije urađenim primerima koji bi čitaocu pojasnili primenu metoda, uključimo detaljno dokumentovan programski kod za pojedine metode, i, svakako, ispravimo greške uočene u prethodnom izdanju.

Zbirka sadrži odabrane rešene zadatke sa vežbi i ispita iz predmeta "Uvod u numeričku matematiku" i "Numeričke metode", koji se predaju na redovnim studijama na Matematičkom fakultetu u Beogradu. Namenjena je u prvom redu studentima Matematičkog fakulteta, ali i studentima drugih fakulteta koji izučavaju Numeričku matematiku, kao i stručnjacima koji se u svakodnevnom radu njome koriste. Da bi zadovoljila potrebe ovako širokog kruga korisnika, pored teorijskih zadataka dat je i veliki broj detaljno rešenih numeričkih primera, kao i za niz metoda programski kod napisan u jezicima C ili Fortran 90.

Zbirka je u sadržaju i oznakama saglasna sa udžbenikom prvog autora *Numeričke metode* ([21]) i knjigom *Numerička analiza* Boška Jovanovića i prvog autora ([13]), koje predstavljaju teorijsku osnovu ovoj zbirci. Izloženi materijal je podeljen u jedanaest poglavlja, prema oblastima numeričke matematike kojoj zadaci pripadaju, i obuhvata sve osnovne oblasti ove grane matematike.

Prvo poglavlje je uvodnog karaktera i objašnjava način izlaganja materijala kroz prezentaciju Gauss-ove metode za rešavanje sistema linearnih jednačina sa trodijagonalnom matricom.

Sledeća tri poglavlja su posvećena problemima aproksimacije. U drugom poglavlju su obrađeni različiti vidovi interpolacije, Lagrange-ova, Hermite-ova i interpolacija splajnovima, a pomenute su i racionalna i trigonometrijska interpolacija. Takođe je ilustrovana primena interpolacionih polinoma za rešavanje problema inverzne interpolacije i numeričkog diferenciranja. Primena interpolacionih polinoma u približnom izračunavanju integrala data je u trećem poglavlju. Primerima su ilustrovane primene Newton–Cotes-ovih formula (trapezna, Simpson-ova,...) i kvadraturnih formula Gauss-ovog tipa. Posebna pažnja je posvećena oceni greške ovih algoritama. Četvrto poglavlje je posvećeno aproksimaciji funkcija, i to srednjekvadratnoj i ravnomernoj aproksimaciji, metodi najmanjih kvadrata i Fourierovoj transformaciji, posebno FFT-u.

Metode linearne algebre su obrađene u petom i šestom poglavlju. Za rešavanje sistema linearnih jednačina, izračunavanje determinanti i inverznih matrica koriš-

ćeni su razne varijante Gauss-ove metode, LU i Cholesky dekompozicija. Posebna pažnja je posvećena problemu numeričke stabilnosti. Osim direktnih, obrađene su i iterativne metode sa posebnim osvrtom na tačnost dobijenog rešenja. Metode za rešavanje problema sopstvenih vrednosti matrica izdvojene su u posebno poglavlje, šesto. Obuhvaćene su kako metode za rešavanje potpunog problema (Givens-ova, Jacobi-jeva, Householder-ova metoda i LR i QR algoritam), tako i metode za rešavanje delimičnog problema sopstvenih vrednosti.

Sedmo poglavlje je posvećeno rešavanju nelinearnih jednačina i sistema metodom proste iteracije (Jacobi-jevom) i Newton-ovom metodom. Obrađena je i metoda Bairstow-a za rešavanje algebarskih jednačina.

Poslednja četiri poglavlja su posvećena metodama za rešavanje običnih i parcijalnih diferencijalnih jednačina, kao i integralnih jednačina. Zbog svoje različite prirode, metode za rešavanje Cauchy-jevih problema i metode za rešavanje graničnih problema su razdvojene u posebna poglavlja. U osmom poglavlju su za rešavanje Cauchy-jevih problema korišćene aproksimativne metode, metode tipa Runge–Kutta i prediktor–korektor metode. Posebna pažnja je posvećena tačnosti. Metode za rešavanje graničnih problema za obične diferencijalne jednačine, a to su metode konačnih razlika i varijacione metode, su ilustrovane u devetom poglavlju. U poslednja dva poglavlja knjige dati su samo elementarni primeri rešavanja integralnih i parcijalnih jednačina, saglasno programu pomenutih predmeta.

U okviru svakog poglavlja dat je programski kod bar jedne od metoda sa iscrpnim objašnjenjem. S obzirom da je deo tog materijala pisan u programskom jeziku *Fortran90*, koji je vrlo mnogo korišćen u numeričkom modeliranju, u dodatku je dat kratak prikaz ovog programskog jezika.

Beograd, septembra 2002.

D. P. Radunović
A. B. Samardžić

Predgovor trećem izdanju

Knjiga koju imate pred sobom je nastala doradom osnovnog teksta koji je Desanka Radunović objavila 2000. godine u vidu Zbirke zadataka iz numeričkih metoda. S obzirom na dinamičan razvoj ove oblasti matematike, uslovljen i intenzivnim razvojem informacionih tehnologija, drugo izdanje objavljeno 2002. godine je obogaćeno programima napisanim na jezicima C i Fortran 90, koji implementiraju obrađene metode. Ove procedure, kao i dodatak posvećen programskom jeziku Fortran 90 napisao je Aleksandar Samardžić.

Drugo izdanje zbirke je rasprodato, i tako se ukazala prilika da sadržaj dopunimo prikazom rešavanja problema softverskim paketom Matlab, koji predstavlja nezaobilazni profesionalni alat za numerička i naučna izračunavanja, i poseduje veliki broj implementiranih numeričkih metoda. Za metode obrađene u knjizi za koje ne postoje gotove implementacije, prikazano je kako bi se one mogle implementirati u Matlab-u. Velike mogućnosti ovog paketa su iskorišćene da na dobro odabranim primerima ilustruju probleme koji se mogu javiti pri rešavanju realnih modela (velikih dimenzija), a grafički prikazi dobijenih rezultata bi trebalo značajno da pomognu u razumevanju korišćenih numeričkih metoda. Ovaj deo teksta, kao i dodatak posvećen programskom paketu Matlab napisao je Filip Marić.

Autori zahvaljuju svim studentima koji su tokom prethodnih godina, prateći kurs iz predmeta "Numericke metode" na Matematickom fakultetu Univerziteta u Beogradu, svojim primedbama i korisnim sugestijama doprineli kvalitetu ovoga teksta.

Beograd, septembra 2005.

D. P. Radunović[1]
A. B. Samardžić[2]
F. M. Marić [3]

[1] dradun@matf.bg.ac.yu
[2] asamardzic@matf.bg.ac.yu
[3] filip@matf.bg.ac.yu

Sadržaj

Slike

1

Uvod

U uvodnom poglavlju biće prezentiran način izlaganja u ovoj zbirci zadataka na primeru problema rešavanja trodijagonalnog sistema jednačina. Naglasak zbirke treba da bude na pisanju programa, odn. preciznije procedura, koje implementiraju pojedine numeričke metode. Pisanje koda za ovaj domen problema podrazumeva prolazak kroz sve one faze koje su standardne za bilo kakvo programiranje, a to su analiza problema, identifikovanje ulaza i izlaza procedure, implementacija tela procedure i na kraju testiranje napisanog koda.

Da bi se uspešno prošlo kroz sve nabrojane faze, neophodno je dobro poznavanje numeričke metode koja se želi implementirati. Zato će na početku razmatranja svake metode, ili eventualno grupe metoda, biti navedene osnovne formule koje se na tu metodu odnose. Ovaj segment teksta će biti istaknut okvirom. Na primeru metode koja rešava trodijagonalni sistem linearnih jednačina, taj deo teksta bi imao sledeći oblik:

Trodijagonalni sistem jednačina je sistem jednačina oblika

$$\begin{aligned} c_0x_0 + b_0x_1 &= d_0, \\ a_ix_{i-1} + c_ix_i + b_ix_{i+1} &= d_i, \qquad i = 1, \ldots, n-2, \\ a_{n-1}x_{n-2} + c_{n-1}x_{n-1} &= d_{n-1} \end{aligned}$$

Ako je $c_0 \neq 0$, rešenje trodijagonalnog sistema se određuje rekurentnim formulama

$$x_{n-1} = \frac{d_{n-1} - a_{n-1}\beta_{n-1}}{c_{n-1} + a_{n-1}\alpha_{n-1}}, \qquad x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}, \quad i = n-2, \ldots, 0,$$

$$\begin{aligned} \alpha_1 &= -\frac{b_0}{c_0}, & \alpha_i &= -\frac{b_{i-1}}{c_{i-1} + a_{i-1}\alpha_{i-1}} \\ \beta_1 &= \frac{d_0}{c_0}, & \beta_i &= \frac{d_{i-1} - a_{i-1}\beta_{i-1}}{c_{i-1} + a_{i-1}\alpha_{i-1}}, \end{aligned} \qquad i = 2, \ldots, n-1.$$

Faza analize problema treba da počne pažljivim razmatranjem ovih formula; takođe, već iz formula se može uočiti šta su ulazi odn. izlazi u datoj metodi. Međutim, ključni preduslov pisanju koda koji implementira neku numeričku metodu jeste i da se nekoliko zadataka koji se odnose na tu metodu reše ručno. Na ovaj način se upoznaje sa koracima od kojih se sastoji metoda, čime se uočava dekompozicija na potprobleme problema pisanja tela procedure koja implementira metodu. Dalje, kao važan rezultat ručnog rešavanja određenog broja zadataka ostaju međurezultati i finalni rezultati za date probleme, koji su veoma korisni za testiranje i debagovanje procedure koja implementira metodu. Zato će za svaku od metoda koje budu razmatrane u ovoj zbirci biti dato nekoliko zadataka u kojima je dati problem rešen ručno korišćenjem odgovarajuće metode. Na primeru rešavanja trodijagonalnog sistema jednačina, jedan takav zadatak sa rešenjem bi imao oblik:

1.1 *Rešiti trodijagonalni sistem jednačina*

$$\begin{array}{rcrcrcrcr} 3x_0 & + & 4x_1 & & & & & = & 11 \\ x_0 & - & 2x_1 & + & 3x_2 & & & = & 6 \\ & & 2x_1 & - & x_2 & - & x_3 & = & -3 \\ & & & & x_2 & - & 2x_3 & = & -5 \end{array}$$

Rešenje: Na osnovu formula za rešavanje trodijagonalnog sistema jednačina dobijaju se parametri

$$\alpha_1 = -\frac{4}{3}, \qquad \alpha_2 = -\frac{3}{-\frac{4}{3}-2} = \frac{9}{10}, \qquad \alpha_3 = -\frac{-1}{2\cdot\frac{9}{10}-1} = \frac{5}{4},$$

$$\beta_1 = \frac{11}{3}, \qquad \beta_2 = \frac{6-\frac{11}{3}}{-\frac{4}{3}-2} = -\frac{7}{10}, \qquad \beta_3 = \frac{-3-2\left(-\frac{7}{10}\right)}{2\cdot\frac{9}{10}-1} = -2,$$

pa je

$$x_3 = 4, \qquad x_2 = 3, \qquad x_1 = 2, \qquad x_0 = 1.$$

Za određeni broj metoda u zbirci, pored računskih zadataka, biće urađen i jedan ili više teorijskih zadataka koji produbljuju formule kojima je opisana metoda ili izvode neke pomoćne rezultate koji će biti od koristi za računske zadatke.

Kada se završi sa računskim zadacima za određenu metodu, može se pristupiti pisanju koda koji implementira metodu. Za većinu obrađenih metoda urađena je odgovarajuća procedura, pri čemu je za implementaciju u prvom delu zbirke korišćen programski jezik *C*, a u drugom delu zbirke *F*ORTRAN 90[1]. Na kraju većine poglavlja, biće dat kratak osvrt na opisane numeričke metode kroz sistem *M*ATLAB, koji, kao profesionalni alat za numerička i naučna izračunavanja, poseduje veliki broj implementiranih numeričkih metoda. Naglasak u izlaganju će biti stavljen, pre svega, na prikazivanje gotovih rutina i rešavanje računskih zadataka njihovim korišćenjem.

[1]ovi programski jezici su odabrani zato što se najčešće koriste za numeričko programiranje

Za programiranje pod *C*-om dostupna je standardna literatura ([14]), dok za upoznavanje sa osnovama programiranja na *FORTRAN*-u treba konsultovati dodatak A ove zbirke. Sve procedure su objedinjene u biblioteku koja je nazvana `libnumerics` i koja je dostupna na *Web* strani predmeta ([27]). Biblioteka pored procedura koje implementiraju pojedinačne metode sadrži prateći `Makefile`, kao i posebne test programe sa odgovarajućim test primerima za svaku metodu; u ovoj zbirci biće predstavljen samo test program za metodu za rešavanje sistema trodijagonalnih jednačina, dok će za ostale metode biti prezentirane samo procedure koje implementiraju metode. Čitava biblioteka, dakle i sav kod koji će biti prikazan u ovoj zbirci, pisana je isključivo sa *UNIX* okruženjem na umu i pomoću *GNU* programerskih alata; za uputstva o korišćenju ovih alata dostupna je odgovarajuća literatura ([23]).

C U slučaju posmatranog problema rešavanja trodijagonalnih sistema jednačina tokom faze analize problema može se zaključiti da se implementacija metode sastoji od prostog prevođenja datih formula u odgovarajući kod uz direktno rešavanje slučaja $n = 1$, radi izbegavanja referenciranja α_0 i β_0. Kao ulazi u metodu mogu se uočiti dimenzija sistema i, svakako, koeficijenti sistema; radi uštede prostora bolje je da koeficijenti budu predstavljeni kao četiri vektora (vektor koeficijenata na glavnoj dijagonali i dve do-dijagonale, kao i vektor koeficijenata na desnoj strani jednačina) nego da se u memoriji drži čitava matrica sistema. Izlaz iz metode je vektor rešenja sistema. S obzirom da na *C* jeziku procedure ne mogu da vraćaju vektore, odgovarajuću proceduru treba deklarisati tako da vraća `void` tip, a vektor u koji će biti smeštena rešenja treba preneti kao dodatni argument procedure. Na osnovu svega ovoga, može se napisati deklaracija procedure koja će implementirati metodu (proceduru ćemo nazvati `tridiag()`, a trudićemo se da kroz dalji tekst identifikatorima u kodu dosledno dajemo deskriptivna imena na engleskom jeziku):

```
void
tridiag (int n, double *a, double *b, double *c, double *d, double *x);
```

Ovde je `n` dimenzija sistema, a `a`, `b`, `c` i `d` vektori odgovarajućih koeficijenata sistema, dok je `x` vektor u koji procedura `tridiag()` treba da smesti rešenja sistema. U kodu koji će biti prezentiran u ovoj zbirci, za predstavljanje veličina koje ulaze u izračunavanje dosledno je korišćen realni tip dvostruke tačnosti, iz prostog razloga što je za tip aplikacije kakva je numerička analiza tačnost rezultata na prvom mestu.

Sama implementacija procedure `tridiag()` sastoji se, kako je gore već pomenuto, od prevođenja formula koje opisuju metodu u odgovarajuće izraze; pored ovoga, na početku svake procedure dodaćemo odgovarajuću proveru validnosti argumenata procedure što će u ovoj biblioteci biti urađeno prostim `assert()` pozivima[2]. Na taj način, telo procedure `tridiag()` ima sledeći oblik (za programski kod će kroz ovu zbirku takođe biti korišćen `poseban font`):

[2] u realnim uslovima bi svakako trebalo osmisliti neki robustniji način reagovanja na eventualne greške u argumentima od nasilnog prekidanja programa, što se postiže ovom funkcijom

```
#include <stdlib.h>
#include <assert.h>

/* Funkcija tridiag() resava trodijagonalni sistem jednacina oblika:
  a[i]*x[i-1]+c[i]*x[i]+b[i]*x[i+1]=d[i]
gde je n>0, i=0,...,n-1, a[0]==0, b[n-1]==0 i c[i]!=0. Argumenti funkcije su:
  n - dimenzija sistema
  a, b, c, d - polja sa koeficijentima sistema jednacina
  x - polje u koje ce biti smesteno resenje sistema
Pretpostavlja se da su sva polja alocirana izvan funkcije tridiag(). */
void
tridiag (int n, double *a, double *b, double *c, double *d, double *x)
{
  double *alpha, *beta;     /* Polja pomocnih koeficijenata. */
  int i;            /* Brojac u petljama. */

  /* Proverava se da li su ispunjeni uslovi sistema. */
  assert (n > 0);
  assert (a[0] == 0 && b[n - 1] == 0);
  for (i = 0; i < n; i++)
    assert (c[i] != 0);

  /* Resava se specijalni slucaj za n jednako 1. */
  if (n == 1) {
    x[0] = d[0] / c[0];
    return;
  }

  /* Alociraju se polja pomocnih koeficijenata. */
  alpha = (double *) malloc ((n + 1) * sizeof (double));
  beta = (double *) malloc ((n + 1) * sizeof (double));

  /* Racunaju se vrednosti pomocnih koeficijenata. */
  alpha[1] = -b[0] / c[0];
  beta[1] = d[0] / c[0];
  for (i = 2; i < n; i++)
    {
      alpha[i] = -b[i - 1] / (a[i - 1] * alpha[i - 1] + c[i - 1]);
      beta[i] =
        (d[i - 1] - a[i - 1] * beta[i - 1]) / (a[i - 1] * alpha[i - 1] +
                                               c[i - 1]);
    }

  /* Racunaju se resenja sistema. */
  x[n - 1] = (d[n - 1] - a[n - 1] * beta[n - 1]) / (a[n - 1] * alpha[n - 1] +
                                                   c[n - 1]);
  for (i = n - 2; i >= 0; i--)
    x[i] = alpha[i + 1] * x[i + 1] + beta[i + 1];

  /* Oslobadjaju se polja pomocnih koeficijenata. */
  free (alpha);
  free (beta);
}
```

Nastojalo se, u skladu sa elementarnim programerskim pravilima, da komentara bude sto više, a posebno da svaka procedura ima komentar u zaglavlju koji naznačava šta ona tačno radi i koji su joj ulazi i izlazi. Dalje, da deklaraciju svake

promenljive prati komentar koji opisuje čemu služi ta promenljiva. Na kraju, da svaki netrivijalni segment koda bude propraćen bar jednom rečenicom komentara koja pojašnjava na šta se isti odnosi.

Treba uočiti kako prilikom operacija deljenja nije proveravano da li je deljenik jednak 0. Razlog je što se pretpostavlja da se izvračunavanja u pokretnom zarezu izvršavaju prema IEEE 754 standardu, koji predviđa postojanje vrednosti za predstavljanje rezultata takvih operacija, kao i nastavak izračunavanja nad takvim vrednostima. Ovakav pristup je dosledno sproveden i u svim ostalim primerima u ovom tekstu.

Nakon što je gore prikazanom procedurom `tridiag()` implementirana metoda za rešavanje trodijagonalnog sistema jednačina, potrebno je napisani kod testirati. U tu svrhu, može se napisati jedan mali program koji, recimo, čita postavku problema sa standardnog ulaza, potom poziva proceduru `tridiag()` da reši problem, i na kraju ispisuje rešenja na standardni izlaz. U ovom slučaju, taj program bi bio oblika:

```
#include <stdio.h>
#include <stdlib.h>

/* Program testira resavanje trodijagonalnog sistema jednacina. Program cita
postavku problema sa standardnog ulaza i ispisuje rezultate na standardni
izlaz. Na ulazu se prvo unosi dimenzija sistema, a zatim red po red
koeficijenti jednacina sistema a[i], b[i], c[i] i d[i]. Na izlazu se u prvi
red upisuje dimenzija sistema, a zatim red po red resenja sistema. Program
nema ugradjenu proveru sintaksnih i semantickih gresaka pri unosu problema. */
int
main ()
{
  int n;            /* Dimenzija sistema. */
  double *a, *b, *c, *d;    /* Polja sa koeficijentima sistema. */
  double *x;            /* Polje sa resenjem sistema. */
  int i;            /* Brojac u petljama. */

  /* Ucitava se dimenzija sistema. */
  scanf ("%d", &n);

  /* Alociraju se polja sa koeficijentima sistema. */
  a = (double *) malloc (n * sizeof (double));
  b = (double *) malloc (n * sizeof (double));
  c = (double *) malloc (n * sizeof (double));
  d = (double *) malloc (n * sizeof (double));

  /* Ucitavaju se koeficijenti sistema. */
  for (i = 0; i < n; i++)
    scanf ("%lf%lf%lf%lf", &a[i], &b[i], &c[i], &d[i]);

  /* Alocira se polje za resenje sistema. */
  x = (double *) malloc (n * sizeof (double));

  /* Resava se sistem. */
  tridiag (n, a, b, c, d, x);

  /* Oslobadjaju se polja sa koeficijentima sistema. */
  free (a);
```

```
  free (b);
  free (c);
  free (d);

  /* Ispisuje se resenje sistema. */
  printf ("%d\n", n);
  for (i = 0; i < n; i++)
    printf ("%g\n", x[i]);

  /* Oslobadja se polje sa resenjem sistema. */
  free (x);

  exit (EXIT_SUCCESS);
}
```

Struktura test programa je jednostavna i detaljno pojašnjena komentarima, a u uvodnom komentaru je preciziran i format ulaznih odn. izlaznih podataka. Test program ne proverava unesene podatke u smislu sintaksnih (na primer, ako se na ulazu umesto broja koji predstavlja dimenziju sistema pojavi nešto što nije broj, recimo `aaa`) odnosno semantičkih (na primer, ako se na ulazu za dimenziju sistema pojavi `-5`) grešaka u ulaznom formatu. Jednostavnosti radi, pretpostavka je kroz sve test programe u biblioteci da će ulazni podaci biti ispravni; u slučaju pisanja realnih programa treba se naravno znatno detaljnije pozabaviti onim delom validacije unosa koji je neophodno uraditi radi pravilnog učitavanja ulaznih podataka [3].

Test programi ovakvog oblika su pogodni za neinteraktivno testiranje većim brojem test primera. Takođe se dobro uklapaju u *UNIX* filozofiju pisanja programa koji obrađuju podatke sa standardnog ulaza i ispisuju rezultate na standardni izlaz, koji se onda mogu kombinovati tako da se pomoću više ovakvih malih programa rešava neki veći problem. Na primer, ako je zadatak odrediti normu vektora rešenja nekog sistema trodijagonalnih jednačina, mnogo je bolje napisati posebne programe koji rešavaju trodijagonalni sistem odn. računaju normu, a potom njihovim kombinovanjem (tačnije preusmeravanjem izlaza prvog programa na ulaz drugog) rešiti polazni problem, nego pisati jedan veliki program koji rešava čitav problem.

Po uspešnom prevođenju ovog sada već malog projekta, treba pristupiti testiranju sa određenim brojem test primera. Tako se može pripremiti fajl u gore opisanom ulaznom formatu koji odgovara zadatku 1,

```
4
0 4 3 11
1 3 -2 6
2 -1 -1 -3
1 0 -2 -5
```

i zatim pokrenuti gornji program sa standardnim ulazom preusmerenim iz ovog fajla. Ukoliko nema grešaka u proceduri koja implementira metodu, na standardnom izlazu će biti ispisan tačan rezultat (opet u gore opisanom formatu):

[3] sve procedure koje sačinjavaju biblioteku inače rigorozno proveravaju ulazne podatke u smislu zadovoljavanja početnih uslova numeričke metode koju rešavaju

```
4
1
2
3
4
```

Ukoliko rezultat nije identičan onome što je dobijeno ručnim rešavanjem datog problema, onda se pomoću debagera i korišćenjem međurezultata dobijenih ručnim rešavanjem lako može ustanoviti izvor greške u kodu.

MATLAB Jasno je da nije moguće očekivati da MATLAB raspolaže gotovim implementacijama svih metoda koje se u okviru ove zbirke pominju. U nekim slučajevima, u nedostatku već gotove implementacije određene numeričke metode, biće prikazano kako bi se ona mogla implementirati. Na ovaj korak smo se odlučili kako bismo prikazali način kodiranja specifičan za sistem MATLAB i skrenuli pažnju na razlike u stilu kodiranja između MATLAB-a i ostalih programskih jezika koji se u zbirci pominju. Većina rešenja je data u obliku MATLAB skriptova, a ne u obliku funkcija. Osnovni razlog za to je što nismo želeli da opterećujemo zbirku opsežnim proverama korektnosti argumenata funkcija, kao ni neophodnom standardnom dokumentacijom koje implementacija MATLAB funkcija podrazumeva. Sa druge strane, trudili smo se da skriptovi budu napisani na način koji omogućava njihovo jednostavno prilagođavanje i eventualno pretvaranje u funkcije.

Prilikom izbora zadataka koji su rešavani korišćenjem MATLAB-a, trudili smo se da oni budu ilustrativni i da ukažu na probleme do kojih može doći prilikom numeričkog rešavanja zadataka, a koji se možda ne mogu uočiti prilikom ručnog rešavanja zadataka kod kojih je dimenzija problema relativno mala. Trudili smo se da većinu zadataka i grafički ilustrujemo u nadi da će ovo doprineti još boljem sagledavanju pomenutih problema. Podstičemo čitaoce da samostalno eksperimentišu menjanjem ulaznih parametara skriptova koji su u zbirci dati i rešavanjem većeg broja zadataka istom metodom. Osmatranje grafičkog prikaza dobijenih rezultata bi trebalo značajno da pomogne razumevanju korišćenih numeričkih metoda.

Svako poglavlje o MATLAB-u počinje rezimeom funkcija koje su za to poglavlje karakteristične. U slučaju rešavanja trodijagonalnog sistema, rezime bi bio:

Rešavanje sistema $Ax = b$	`x = A\b`
Rešavanje sistema $xA = b$	`x = b/A`
Efikasno čuvanje retkih matrica	`R = sparse(A);`
Kreiranje pune matrice na osnovu retke	`A = full(R);`
Retke dijagonalne matrice	`R = spdiags(cols, diagnums, m, n)`

Nakon rezimea, sledi opis funkcionalnosti kojima MATLAB raspolaže, a koje su relevantne za rešavanje zadataka iz oblasti koja se u poglavlju obrađuje. Na primer, za rešavanje sistema linearnih jednačina, MATLAB raspolaže ugrađenim operatorima `\` i `/`. Sistem oblika $A\mathbf{x} = \mathbf{b}$, gde je A kvadratna matrica, a $\mathbf{x}$ i $\mathbf{b}$ vektori kolone, se rešava komandom `x = A\b`. Sistem oblika $\mathbf{x}A = \mathbf{b}$, gde su $\mathbf{x}$ i $\mathbf{b}$ vektori vrste, se rešava komandom `x = b/A`. Mnemotehničko pravilo koje olakšava pamćenje operatora koji se koriste za rešavanje sistema je sledeće: rešenje prvog sistema je

$x = A^{-1}b$, a drugog je $x = bA^{-1}$. Ovo asocira na deljenje matricom A, pri čemu se u prvom slučaju ona nalazi levo od vektora **b**, a u drugom desno od **b**. Pošto je u oba slučaja `A` u ,,imeniocu" ono se mora naći ,,ispod" `b`.

Osnovna karakteristika trodijagonalnih sistema je postojanje velikog broja nula u matrici sistema. Kako bi poboljšao efikasnost pri radu sa matricama koje imaju veliki broj nula, MATLAB implementira posebnu strukturu retkih matrica (*sparse matrices*). Obična matrica se u retku reprezentaciju prevodi korišćenjem funkcije `sparse`, dok se retka matrica prevodi u uobičajenu, punu, reprezentaciju korišćenjem funkcije `full`. Za kreiranje retkih dijagonalnih matrica, MATLAB nudi funkciju `spdiags`. Oblik korišćenja ove funkcije koji se može koristiti pri rešavanju trodijagonalnog sistema je `A = spdiags(cols, diagnums, m, n)`. Argument `cols` je matrica čije su kolone vektori koji se postavljaju na dijagonale rezultujuće matrice. Argument `diagnums` sadrži oznake dijagonala na koje se ti vektori postavljaju, pri čemu se koristi konvencija da se glavna dijagonala označava nulom, dijagonale ispod glavne, redom, negativnim brojevima, a dijagonale iznad glavne, redom, pozitivnim brojevima. Argumenti `m` i `n` označavaju dimenziju rezultujuće matrice. Ukoliko je vektor kolona koji se postavlja na dijagonalu iznad glavne duži od dijagonale, ignorišu se njegovi početni elementi. Ukoliko se postavljanje ovakvog vektora vrši na dijagonalu ispod glavne, ignorišu se njegovi krajnji elementi.

Većina ugrađenih funkcija i operatora, uključujući i operatore za rešavanje sistema jednačina, su unapred prilagođeni radu sa retkim matricama.

Nakon ovoga, sledi nekoliko zadataka koji ilustruju korišćenje upravo opisanih funkcija. Kao primer, korišćenjem MATLAB-a rešićemo sistem zadat zadatkom 1.1.

Pošto je sistem trodijagonalni, kreiraće se retka matrica sistema, a zatim se iskoristiti uobičajeni operator za rešavanje sistema. Ovaj operator je prilagođen radu sa retkim matricama, tako da će rešavanje sistema biti prilično efikasno.

```
A = spdiags([1 2 1 0; 3 -2 -1 -2; 0 4 3 -1]', [-1 0 1], 4, 4);
b = [11 6 -3 -5]';
x = A\b
```

Način izlaganja numeričkih metoda koji je u ovom poglavlju demonstriran na primeru rešavanja trodijagonalnog sistema jednačina, biće dosledno sproveden kroz ostatak teksta. Naravno, tokom čitanja treba stalno imati na umu da se puni efekat u smislu ovladavanja tehnikom programskog rešavanja problema numeričke matematike može postići samo ukoliko se uvek računski zadaci i programerska implementacija metode prvo pokušaju uraditi samostalno.

2

Interpolacija

Funkcija $g(x)$ interpoliše zadatu funkciju $f(x)$ ako je

$$g(x_k) = f(x_k), \qquad k = 0, \ldots, n.$$

Tačke $x_0, \ldots, x_n$ nazivaju se čvorovi interpolacije. Ako je funkcija $g(x)$ polinom stepena n, ona se naziva interpolacioni polinom funkcije $f(x)$.

2.1 Lagrange-ov polinom

Lagrange-ov interpolacioni polinom stepena n funkcije $f(x)$ određen čvorovima interpolacije $x_0, \ldots, x_n$, za $x_i \neq x_j$, je

$$L_n(x) = \sum_{i=0}^{n} \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right) f(x_i) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_i)\, \omega'_{n+1}(x_i)} f(x_i),$$

gde je

$$\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$$

Greška interpolacije funkcije je

$$R_n(x) = f(x) - L_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)\, \omega_{n+1}(x), \qquad \xi \in [y_1, y_2],$$

gde je $y_1 = \min(x_0, \ldots, x_n)$, $y_2 = \max(x_0, \ldots, x_n)$, i ocenjuje se izrazom

$$|R_n(x)| \leq \frac{1}{(n+1)!} M_{n+1} |\omega_{n+1}(x)|, \qquad M_{n+1} = \max_{[y_1, y_2]} |f^{(n+1)}|$$

2.1 *Ako je*

$$l_k(x) = \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{x - x_j}{x_k - x_j}, \qquad k = 0, \ldots, n,$$

dokazati da je

$$\sum_{k=0}^{n} l_k(x) \equiv 1.$$

Rešenje: Zbog jedinstvenosti Lagrange-ovog interpolacionog polinoma sledi da je za svaki polinom $P(x)$ stepena ne većeg od n

$$P(x) \equiv \sum_{k=0}^{n} l_k(x) P(x_k).$$

Tvrđenje sledi kada se stavi da je $P(x) \equiv 1$.

2.2 *Napisati Lagrange-ov interpolacioni polinom n-tog stepena ($n \geq 2$) za funkciju*

$$f(x) \,:\, x \longrightarrow 1 + \sum_{k=0}^{n} \frac{x_k}{x - x_k} \frac{\omega(x)}{\omega'(x_k)},$$

gde je

$$\omega(x) = \prod_{i=0}^{n} (x - x_i), \qquad x_i < x_j \quad za \quad i < j.$$

Rešenje: Lagrange-ov interpolacioni polinom funkcije koja je sama polinom je identičan funkciji ukoliko je stepen polinoma veći ili jednak stepenu funkcije. Stoga je

$$\sum_{k=0}^{n} \frac{x_k}{x - x_k} \frac{\omega(x)}{\omega'(x_k)} \equiv x,$$

pa je $f(x) = 1 + x$ i njen interpolacioni polinom je

$$L_n(x) = \sum_{k=0}^{n} \frac{1 + x_k}{x - x_k} \frac{\omega(x)}{\omega'(x_k)}.$$

2.3 *Konstruisati Lagrange-ov interpolacioni polinom za funkciju $f(x) = \sqrt{x}$, ako su čvorovi interpolacije $x_0 = 100$, $x_1 = 121$ i $x_2 = 144$. Izračunati vrednost dobijenog polinoma u tački $X = 115$ i oceniti gresku rezultata koristeći formulu za grešku interpolacije. Uporediti sa stvarnom greškom.*

Rešenje: Zadata tabela

x	100	121	144
$f(x)$	10	11	12

definiše Lagrange-ov interpolacioni polinom drugog stepena

$$L_2(x) = \frac{(x-121)(x-144)}{(100-121)(100-144)}\,10 + \frac{(x-100)(x-144)}{(121-100)(121-144)}\,11 + \frac{(x-100)(x-121)}{(144-100)(144-121)}\,12 \;=\; -\frac{2}{21252}\,x^2 + \frac{1454}{21252}\,x + \frac{87120}{21252}.$$

Greška interpolacije je

$$|f(115) - L_2(115)| \le \frac{1}{3!}M_3|(115-100)(115-121)(115-144)| = 0.16\cdot 10^{-2} \approx 10^{-3},$$

jer je $M_3 = \max_{[100,144]} |f'''(x)| = 0.375 \cdot 10^{-5}$.

S obzirom na različiti red veličine koeficijenata polinoma $L_2(x)$, može se očekivati velika greška rezultata ako koeficijenti polinoma nisu izračunati sa zadovoljavajućom tačnošću, tj. ako nisu dati sa dovoljnim brojem decimala. Određujemo dopuštenu apsolutnu grešku Δ koeficijenata polinoma, tako da je računska greška rezultata $\Delta\big(L_2(115)\big) = 10^{-4}$ (da bi ona bila zanemarljiva u odnosu na grešku metode),

$$\Delta\, 115^2 + \Delta\, 115 + \Delta = 10^{-4} \quad \Longrightarrow \quad \Delta = 0.75 \cdot 10^{-8}.$$

Dakle, koeficijente polinoma treba izračunati bar na osam decimala (tada je apsolutna greška tih brojeva $0.5 \cdot 10^{-8}$),

$$L_2(x) = -0.00009411x^2 + 0.06842171x + 4.09937888.$$

Rezultat zapisan samo sigurnim ciframa, s obzirom na procenjenu tačnost, je

$$L_2(115) = 10.723270\cdots = 10.723 \qquad \sqrt{115} = 10.72380\ldots$$

2.4 *Neka su ekvidistantnom tabelom sa korakom h date vrednosti funkcije $f(x)$ zaokrugljene na r decimala. Linearnom interpolacijom se pomoću tabele računa $f(\overline{x})$. Neka su apscise u tabeli date tačno, apscisa $\overline{x}$ je zaokrugljena na s decimala, a izračunata vrednost $f(\overline{x})$ zaokrugljena na t decimala. Ako je $f''(x)$ neprekidna funkcija na tabeliranom intervalu a δ totalna greška interpolacije, pokazati da je*

$$|\delta| \le \frac{1}{8}M_2h^2 + 5M_1 \cdot 10^{-s-1} + 5 \cdot 10^{-r-1} + 5 \cdot 10^{-t-1},$$

gde je

$$M_1 = \max_{[x_0,x_n]} |f'(x)|, \qquad M_2 = \max_{[x_0,x_n]} |f''(x)|.$$

Rešenje: Interpolacioni polinom prvog stepena funkcije $f(x)$ određen čvorovima x_k i x_{k+1} u tački $\overline{x}$ je

$$L_1(\overline{x}) = \frac{\overline{x} - x_{k+1}}{x_k - x_{k+1}}\, f(x_k) + \frac{\overline{x} - x_k}{x_{k+1} - x_k}\, f(x_{k+1}).$$

Greška metode (linearne interpolacije) se ocenjuje izrazom

$$|R_1(\overline{x})| = \frac{|f''(\xi)|}{2}\, |(\overline{x} - x_k)(\overline{x} - x_{k+1})| \le \frac{h^2}{8}\, M_2.$$

Ocenimo sada računsku grešku. Vrednosti funkcije f su date sa apsolutnom greškom $A(f) = 5 \cdot 10^{-r-1}$, a $A(\overline{x}) = 5 \cdot 10^{-s-1}$. Linearna ocena apsolutne greške vrednosti polinoma L_1 u tački $\overline{x}$ je onda

$$\begin{aligned} A(L_1(\overline{x})) &= \left|\frac{\partial L_1(\overline{x})}{\partial \overline{x}}\right| A(\overline{x}) + \left|\frac{\partial L_1(\overline{x})}{\partial f_k}\right| A(f) + \left|\frac{\partial L_1(\overline{x})}{\partial f_{k+1}}\right| A(f) \\ &= \left|\frac{f_{k+1} - f_k}{h}\right| A(\overline{x}) + \left(\frac{x_{k+1} - \overline{x}}{h} + \frac{\overline{x} - x_k}{h}\right) A(f) = \frac{|\Delta f_k|}{h}\, A(\overline{x}) + A(f) \end{aligned}$$

Iz veze konačne razlike i izvoda funkcije $\Delta f_k = h\, f'(\xi)$, $\xi \in [x_k, x_{k+1}]$ sledi da je

$$A(L_1(\overline{x})) = |f'(\xi)| A(\overline{x}) + A(f) \le 5M_1 \cdot 10^{-s-1} + 5 \cdot 10^{-r-1}.$$

Totalna greška rezultata linearne interpolacije je zbir greške interpolacije, računske greške i greške zaokrugljivanja konačnog rezultata,

$$\delta = |R_1(\overline{x})| + A(L_1(\overline{x})) + 5 \cdot 10^{-t-1},$$

što dokazuje tvrđenje zadatka.

2.5 *Odrediti u intervalu* $(1, 1.2)$ *tačke* x_0 *i* x_1 *tako da linearna interpolacija za funkciju* $f(x) = \ln x$ *kroz tačke* 1, x_0, x_1 *i* 1.2 *ima grešku ne veću od* $5 \cdot 10^{-4}$.

Rešenje: Greška Lagrange-ovog interpolacionog polinoma prvog stepena za funkciju $\ln x$ konstruisanog kroz tačke a i b je

$$R_1(x) = -\frac{1}{2\xi^2}(x - a)(x - b), \qquad \xi,\, x \in (a, b).$$

Kako je $1/\xi^2 \le 1/a^2$ i $|(x - a)(x - b)| \le (b - a)^2/4$, to je

$$|R_1(x)| \le \frac{1}{8}\left(\frac{b}{a} - 1\right)^2.$$

Iz zahteva da je $|R_1(x)| \le 5 \cdot 10^{-4}$ sledi da je $b/a \le 1.06325$. U prvom intervalu je $a = 1$, pa je $b = x_0 = 1.06325$. x_0 je početak sledećeg intervala, a njegov kraj je $x_1 = 1.06325 \cdot x_0 = 1.13049$. Sledeći čvor interpolacije bi bio $1.06325 \cdot x_1 = 1.20199$, što je veće od 1.2. Stoga je i u intervalu $(x_1, 1.2)$ tačnost zadovoljena, pa je

$$x_0 = 1.06325, \qquad x_1 = 1.13049.$$

2.6 *Neka je funkcija*

$$f(x) = \int_0^x \sin^2 t \, dt$$

tabelirana na intervalu $x \in [0,1]$ *u ravnomerno raspoređenim tačkama sa korakom* h. *Pod pretpostavkom da su vrednosti funkcije uzete sa dovoljno sigurnih cifara da se može zanemariti greška zaokrugljivanja, odrediti* α *kao funkciju od* h *i* ε *tako da greška linearne interpolacije ne premašuje datu vrednost* ε *na intervalu* $[\alpha, 1]$. *Kakvo* h *treba da bude da bi se postigla tačnost* 0.005 *za* $\alpha = 0.1$*?*

Rešenje: Kako je $f'(x) = \sin^2 x$ i $f''(x) = \sin(2x)$, to je

$$\max_{[\alpha,1]} |f''(x)| = \begin{cases} 1, & \alpha \in [0, \pi/4] \\ \sin(2\alpha), & \alpha \in [\pi/4, 1] \end{cases}$$

jer je $\sin(2x)$ rastuća funkcija kada $x \in [0, \pi/4]$ i opadajuća kada $x \in [\pi/4, 1]$. Stoga je greška linearne interpolacije

$$|R_1(x)| \le \begin{cases} h^2/8, & \alpha \in [0, \pi/4] \\ (h^2/8)\sin(2\alpha) & \alpha \in [\pi/4, 1] \end{cases} \qquad x \in [\alpha, 1].$$

U slučaju kada greška interpolacije R_1 zavisi od α, greška će biti manja od ε ako je

$$\alpha \ge \frac{1}{2} \arcsin \frac{8\varepsilon}{h^2}.$$

Kada je $\alpha = 0.1$ greška ne zavisi od α i biće manja od 0.005 ako je

$$h^2/8 \le 0.005, \qquad \text{tj.} \qquad h \le 0.2.$$

2.7 *Ako uzmemo nule Čebišev-ljevog polinoma* $T_{n+1}(x) = \cos((n+1)\arccos x)$ *za čvorove* x_k *u Lagrange-ovoj interpolacionoj formuli*

$$L_n(x) = \sum_{k=0}^{n} l_k(x) f(x_k), \qquad l_k(x) = \prod_{\substack{i=0 \\ i \ne k}}^{n} \frac{x - x_i}{x_k - x_i},$$

dokazati da je

$$l_k(x) = \frac{(-1)^k \sqrt{1 - x_k^2}}{n+1} \frac{T_{n+1}(x)}{x - x_k},$$

i oceniti grešku interpolacije.

Rešenje: Ako su čvorovi interpolacije nule Čebišev-ljevog polinoma $T_{n+1}(x)$,

$$x_k = \cos \frac{2k+1}{2(n+1)} \pi, \qquad k = 0, \ldots, n,$$

onda je

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i) = 2^{1-(n+1)}T_{n+1}(x).$$

Kako je

$$T'_{n+1}(x) = \Big(\cos\big((n+1)\arccos x\big)\Big)' = \frac{n+1}{\sqrt{1-x^2}}\,\sin\big((n+1)\arccos x\big),$$

i

$$T'_{n+1}(x_k) = \frac{n+1}{\sqrt{1-x_k^2}}\,\sin(2k+1)\frac{\pi}{2} = \frac{(-1)^k(n+1)}{\sqrt{1-x_k^2}},$$

tvrđenje neposredno sledi jer je

$$l_k(x) = \frac{\omega_{n+1}(x)}{(x-x_k)\omega'_{n+1}(x_k)} = \frac{T_{n+1}(x)}{(x-x_k)T'_{n+1}(x_k)}.$$

Greška interpolacije je

$$\begin{aligned}|f(x) - L_n(x)| &\le \frac{1}{(n+1)!}\max_{[x_0,x_n]}|f^{(n+1)}(\xi)|\max_{[x_0,x_n]}|2^{-n}T_{n+1}(\xi)|\\ &= \frac{1}{2^n(n+1)!}\max_{[x_0,x_n]}|f^{(n+1)}(\xi)|.\end{aligned}$$

2.8 *Korišćenjem interpolacionog polinoma drugog stepena na intervalu* $[-1,1]$ *izračunati* $e^{0.9}$ *ako su*

a) čvorovi ravnomerno raspoređeni,

b) čvorovi nule odgovarajućeg Čebišev-ljevog polinoma.

Oceniti grešku interpolacije u oba slučaja.

Rešenje: Polinom drugog stepena određen je vrednostima funkcije u tri tačke.
a) U slučaju ravnomernog rasporeda čvorova, čvorovi su $x_0 = -1$, $x_1 = 0$ i $x_2 = 1$,

x	-1	0	1
$f(x)$	0.367879	1.000000	2.718282

Tražena vrednost je

$$\begin{aligned}L_2(0.9) &= \frac{0.9(-0.1)}{(-1)(-2)}\,0.367879 + \frac{1.9(-0.1)}{-1}\,1.000000 + \frac{1.9(0.9)}{2}\,2.718282\\ &= 2.497576,\end{aligned}$$

izračunata sa greškom

$$|\,e^{0.9} - L_2(0.9)| \le \frac{e}{3!}\,|1.9 * 0.9 * (-0.1)| = 0.8 \cdot 10^{-1}.$$

Zadržavajući samo sigurne cifre, rezultat je

$$e^{0.9} = 2.50 \pm 0.08,$$

pri čemu je 5 sigurna cifra u širem smislu.

b) Ako su čvorovi interpolacije nule Čebišev-ljevog polinoma $T_2(x) = \cos(2 \arccos x)$, tabela je

x	-0.866025	0	0.866025
$f(x)$	0.420620	1	2.377443

Tražena vrednost je

$$\overline{L}_2(0.9) = 2.447749,$$

izračunata sa greškom

$$|\, e^{0.9} - \overline{L}_2(0.9)| \leq \frac{M_3}{3!} |1.766025 * 0.9 * 0.033975)| = 0.22 \cdot 10^{-1},$$

gde je $M_3 = \max_{[x_0, x_2]} |\, e^x| = e^{x_2} = 2.377443$.

Zadržavajući samo sigurne cifre, rezultat je

$$e^{0.9} = 2.45 \pm 0.02,$$

pri čemu je 4 sigurna cifra u užem smislu. Tačna vrednost je $e^{0.9} = 2.4596....$

2.9 *a) Odrediti minimalni stepen n interpolacionog polinoma $P_n(x)$ koji na intervalu $[0, 1]$ aproksimira funkciju $f(x) = e^{-x}$ sa tačnošću 10^{-6}, ako su čvorovi interpolacije nule Čebišev-ljevog polinoma. Čebišev-ljev polinom stepena n na intervalu $[-1, 1]$ je polinom $T_n(x) = \cos(n \arccos x)$.*
b) Ako se koristi deo po deo linearna interpolacija sa čvorovima $x_i = i/n$, $i = 0, \ldots, n$, odrediti minimalnu vrednost za n koja obezbeđuje tačnost aproksimacije 10^{-6}.

Rešenje: a) Čebišev-ljev polinom $T_n(t) = \cos(n \arccos t)$ je definisan za $t \in [-1, 1]$ i nule su mu tačke $t_k = \cos((2k + 1)\pi/(2n))$, $k = 0, \ldots, n - 1$. Linearnom smenom $t = 2x-1$ ovaj polinom definišemo na intervalu $[0, 1]$, $T_n^{[0,1]}(x) = T_n(2x-1)$, a nule su mu $x_k = (1 + t_k)/2$. Normiranjem ovog polinoma (množenjem recipročnom vrednošću koeficijenta uz najviši stepen) dobijamo polinom $\overline{T}_n^{[0,1]}(x) = 2^{1-2n} T_n^{[0,1]}(x)$.

Interpolacioni polinom stepena n određen je vrednostima u $(n + 1)$-om čvoru interpolacije. Ako su čvorovi interpolacije nule Čebišev-ljevog polinoma stepena $n + 1$ na intervalu $[0, 1]$, $x_i = \frac{1}{2}\big(1 + \cos \frac{(2i+1)\pi}{2(n+1)}\big)$, $i = 0, \ldots, n$, polinom

$$\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i) \equiv \overline{T}_{n+1}^{[0,1]}(x) \equiv 2^{-1-2n} T_{n+1}(2x - 1),$$

Čebišev-ljev polinom $T_n(x) = \cos(n \arccos x)$ ograničen je sa $|T_n(x)| \leq 1$. Kako je

$$\max_{x\in[0,1]} |f^{(k)}(x)| = 1, \quad \text{i} \quad \max_{x\in[0,1]} |\overline{T}^{[0,1]}_{n+1}(x)| = 2^{-1-2n},$$

greška interpolacije polinomom stepena n sa ovim izborom čvorova je

$$|f(x) - p_n(x)| \leq \frac{1}{(n+1)!}\,\frac{1}{2^{2n+1}},$$

i zadovoljava zadatu tačnost za $n \geq 5$.
b) Kako je

$$|f(x) - p_1^i(x)| \leq \frac{1}{2} \max_{x\in[\frac{i-1}{n},\frac{i}{n}]} \left|(x - \frac{i-1}{n})(x - \frac{i}{n})\right| \leq \frac{1}{8n^2},$$

minimalni broj podintervala za postizanje zahtevane tačnosti je $n = 354$.

2.10 *Neka je* $L_n(x) = \sum_{k=0}^{n} l_k(x)\, f(x_k)$ *Lagrange-ov interpolacioni polinom stepena* n *funkcije* $f(x)$ *i*

$$\overline{h} = \max_{0\leq k\leq n-1}(x_{k+1} - x_k), \qquad \underline{h} = \min_{0\leq k\leq n-1}(x_{k+1} - x_k).$$

Dokazati da važi ocena

$$\sum_{k=0}^{n} |l_k(x)| \leq 2^n \left(\frac{\overline{h}}{\underline{h}}\right)^n.$$

Rešenje: S obzirom na pretpostavku da je $x_{i+1} - x_i \geq \underline{h}$ sledi da je za $i > j$

$$x_i - x_j = (x_i - x_{i-1}) + \cdots + (x_{j+1} - x_j) \geq (i-j)\underline{h},$$

pa je

$$|(x_k - x_0)\ldots(x_k - x_{k-1})\,(x_k - x_{k+1})\ldots(x_k - x_n)| \geq k!\,(n-k)!\,\underline{h}^n.$$

Dalje, pretpostavimo da $x \in [x_{j-1}, x_j]$, pa je $x_0 < \cdots < x_{j-2} < x_{j-1} < x < x_j < x_{j+1} < \cdots < x_n$ odakle sledi da je

$$|x - x_{j-1}| \leq \overline{h}, \qquad |x - x_{j-2}| \leq 2\overline{h}, \qquad \cdots \qquad |x - x_0| \leq j\overline{h},$$
$$|x - x_j| \leq \overline{h}, \qquad |x - x_{j+1}| \leq 2\overline{h}, \qquad \cdots \qquad |x - x_n| \leq (n-j+1)\overline{h},$$

tj.

$$|(x - x_0)\ldots(x - x_{k-1})\,(x - x_{k+1})\ldots(x - x_n)| \leq j!\,(n-j+1)!\,\overline{h}^n.$$

U slučaju da je $j = 1$ ili $j = n$ je

$$|(x - x_0)\ldots(x - x_{k-1})\,(x - x_{k+1})\ldots(x - x_n)| \leq n!\overline{h}^n,$$

što je i maksimalna vrednost ovog proizvoda bilo gde da se u intervalu $[x_0, x_n]$ nalazi argument x.

Konačno je

$$|l_k(x)| = \left| \frac{(x_k - x_0)\dots(x_k - x_{k-1})\,(x_k - x_{k+1})\dots(x_k - x_n)}{(x - x_0)\dots(x - x_{k-1})\,(x - x_{k+1})\dots(x - x_n)} \right| \le \frac{n!}{k!(n-k)!}\,\frac{\overline{h}^n}{\underline{h}^n},$$

pa je

$$\sum_{k=0}^{n} |l_k(x)| \le \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} \left(\frac{\overline{h}}{\underline{h}}\right)^n = \left(\frac{\overline{h}}{\underline{h}}\right)^n \sum_{k=0}^{n} \binom{n}{k} = 2^n \left(\frac{\overline{h}}{\underline{h}}\right)^n.$$

2.11 *Korišćenjem Lagrange-ove interpolacione formule dokazati da je*

$$\sum_{j=-k}^{k} (-1)^j \binom{2k}{k+j} f(x+jh) = (-1)^k h^{2k} f^{(2k)}(t), \qquad x - kh \le t \le x + kh.$$

Rešenje: Lagrange-ov interpolacioni polinom za funkciju $f(x)$ određen čvorovima $x - kh, \dots, x - h,\ x + h, \dots, x + kh$ je

$$L_{2k-1}(x) = \sum_{\substack{j=-k \\ j\neq 0}}^{k} \left(\prod_{\substack{i=-k \\ i\neq 0,j}}^{k} \big(x - (x+ih)\big) \right) \Bigg/ \left(\prod_{\substack{i=-k \\ i\neq 0,j}}^{k} \big((x+jh) - (x+ih)\big) \right) f(x+jh)$$

tj.

$$L_{2k-1}(x) = -\sum_{\substack{j=-k \\ j\neq 0}}^{k} \frac{(-1)^j (k!)^2}{(k+j)!(k-j)!}\, f(x+jh)$$

Greška interpolacije je

$$f(x) - L_{2k-1}(x) = \frac{f^{(2k)}(t)}{(2k)!} \prod_{\substack{i=-k \\ i\neq 0}}^{k} \big(x - (x+ih)\big), \qquad x - kh \le t \le x + kh,$$

pa je, uzimajući u obzir izvedeni izraz za $L_{2k-1}(x)$,

$$f(x) + \sum_{\substack{j=-k \\ j\neq 0}}^{k} \frac{(-1)^j (k!)^2}{(k+j)!(k-j)!}\, f(x+jh) = \frac{(-1)^k (k!)^2}{(2k)!}\, h^{2k}\, f^{(2k)}(t).$$

Množenjem poslednjeg izraza sa $(2k)!/(k!)^2$ sledi tvrđenje.

2.12 *Neka su* $W_k = \exp\frac{2\pi k}{n+1}\imath$, $k = 0, \ldots, n$, *koreni* $(n+1)$*-og reda jedinice,* $W_k^{n+1} = 1$. *Koristeći Lagrange-ov interpolacioni polinom određen čvorovima* W_k, *dokazati da je za proizvoljan polinom* $P_n(x) = a_0 + a_1 x + \cdots + a_n x^n$ *stepena* n

$$(1) \qquad a_0 = \frac{1}{n+1}\sum_{k=0}^{n} P_n(W_k),$$

$$(2) \qquad a_n = \frac{1}{n+1}\sum_{k=0}^{n} W_k\, P_n(W_k).$$

Na osnovu prethodnog dokazati da je za svako $m = 1, \ldots, n$

$$(3) \qquad \sum_{k=0}^{n} W_k^m = 0.$$

Rešenje: Lagrange-ov interpolacioni polinom za funkciju $P_n(x)$ određen čvorovima W_k je

$$L_n(x) = \sum_{k=0}^{n} \frac{\omega_{n+1}(x)}{(x - W_k)\omega'_{n+1}(W_k)}\, P_n(W_k),$$

gde je $\omega_{n+1}(x) = \prod_{k=0}^{n}(x - W_k) = x^{n+1} - 1$ i $\omega'_{n+1}(W_k) = (n+1)W_k^n = (n+1)/W_k$ jer je $W_k^{n+1} = 1$. Funkcija koja se interpoliše $P_n(x)$ je takođe polinom stepena n, pa zbog jedinstvenosti polinoma stepena n određenog svojim vrednostima u $(n+1)$-oj različitoj tački sledi da je $P_n(x) \equiv L_n(x)$. Stoga je

$$P_n(x) = \frac{x^{n+1} - 1}{n+1}\sum_{k=0}^{n} \frac{W_k}{x - W_k}\, P_n(W_k).$$

Za $x = 0$ je

$$a_0 = P_n(0) = \frac{1}{n+1}\sum_{k=0}^{n} P_n(W_k),$$

a kada $x \to \infty$

$$a_n = \lim_{x\to\infty}\frac{P_n(x)}{x^n} = \frac{1}{n+1}\lim_{x\to\infty}\sum_{k=0}^{n}\frac{x^{n+1}-1}{x^n(x - W_k)}\, W_k\, P_n(W_k)$$

$$= \frac{1}{n+1}\sum_{k=0}^{n} W_k\, P_n(W_k).$$

Sada je na osnovu izraza (1)

$$(n+1)a_0 = \sum_{k=0}^{n} P_n(W_k) = (n+1)a_0 + a_1\sum_{k=0}^{n} W_k + a_2\sum_{k=0}^{n} W_k^2 + \cdots + a_n\sum_{k=0}^{n} W_k^n,$$

tj.

$$a_1 \sum_{k=0}^{n} W_k + a_2 \sum_{k=0}^{n} W_k^2 + \cdots + a_n \sum_{k=0}^{n} W_k^n = 0.$$

Kako je $P_n(x)$ proizvoljan polinom, poslednja jednakost važi za svaki izbor koeficijenata $a_0, \ldots, a_n$, a to je moguće samo ako je

$$\sum_{k=0}^{n} W_k^m = 0, \qquad m = 1, \ldots, n.$$

MATLAB

Lagrange-ov polinom:	`p = polyfit(x, y, n)` stepen n je za jedan manji od dužine vektora x i y

Lagrange-ov interpolacioni polinom je direktno moguće konstruisati korišćenjem funkcije `p = polyfit(x, y, n)`. Argumentima `x` i `y` je predstavljena tabela interpolacije, dok je `n` stepen rezultujućeg polinoma. Funkcija vrši interpolaciju u slučaju da je `n` za jedan manji od dimenzije vektora `x`, odnosno `y`. Prikažimo korišćenje ove funkcije kroz sledeći zadatak.

2.13 *Korišćenjem* MATLAB-a, *izvršiti interpolaciju funkcije* $\frac{1}{1+25x^2}$ *na intervalu* $[-1, 1]$ *Lagrange-ovim interpolacionim polinomom stepena 9.*

a) Interpolaciju izvšiti na ekvidistantnoj mreži.

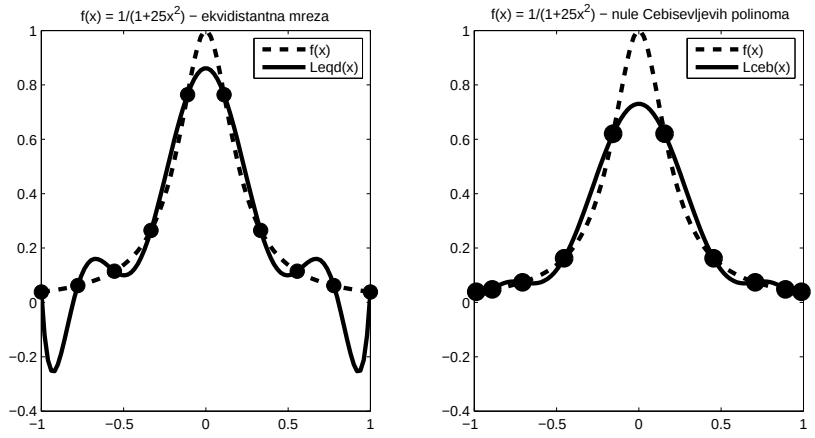
b) Za čvorove interpolacije uzeti nule Čebišev-ljevih polinoma.

Rešenje:

```
% Funkcija koja se interpolise
f = inline('1./(1+25*x.^2)');
% Broj cvorova interpolacije
n = 10;
% Mreza za iscrtavanje
X = linspace(-1,1);
% Ekvidistantna mreza
xeqd = linspace(-1, 1, n);
% Mreza nula Cebisev-ljevih polinoma
xceb = cos((2*(n-1:-1:0)+1) / (2*n)*pi);
% Izracunavamo vrednosti funkcije u tackama obe mreze
yeqd = feval(f, xeqd);
yceb = feval(f, xceb);
% Vrsimo interpolaciju
Leqd = polyfit(xeqd, yeqd, n-1);
Lceb = polyfit(xceb, yceb, n-1);

% Vrsimo iscrtavanje funkcije, polinoma i cvorova mreze
```

```
% za slucaj interpolacije na ekvidistantnoj mrezi
subplot(1,2,1); grid on;
plot(X, f(X), 'b', X, polyval(Leqd, X), 'r', xeqd, yeqd, 'ro')
title('f(x) = 1/(1+25*x^2) - ekvidistantna mreza');
legend('f(x)', 'Leqd(x)');

% Vrsimo iscrtavanje funkcije, polinoma i cvorova mreze
% za slucaj interpolacije na Cebisev-ljevoj mrezi
subplot(1, 2, 2); grid on;
plot(X, f(X), 'b', X, polyval(Lceb, X), 'r', xceb, yceb, 'ro')
title('f(x) = 1/(1+25*x^2) - nule Cebisevljevih polinoma');
legend('f(x)','Lceb(x)');
```

Rezultat rada skripta je prikazan na slici 2.1.



Slika 2.1: Lagrange-ova interpolacija funkcije $\frac{1}{1+25x^2}$. (a) ekvidistantna mreža, (b) nule Čebišev-ljevih polinoma

Primetimo da urađeni primer pokazuje kako se oscilovanje interpolacionog polinoma i greška mogu značajno smanjiti ukoliko se za interpolaciju koriste upravo nule Čebišev-ljevih polinoma.

Na ovom mestu navodimo kompletan kod funkcije koja vrši Lagrange-ovu interpolaciju. Cilj je da kroz primer prikažemo način kodiranja koji je poželjan za sistem MATLAB.

```
% INTERP1LAGRANGE interpolacija Lagranzevim interpolacionim polinomom
% L = interp1Lagrange(x,y)
% Tablica interpolacije mora da sadrzi bar jedan cvor i svi cvorovi moraju biti
% razliciti
%
% Argumenti funkcije su :
```

```
%   x - n-dimenzioni vektor kojim se zadaju cvorovi interpolacije
%   y - n-dimenzioni vektor kojim se zadaju vrednosti funkcije u cvorovima
% Funkcija vraca :
%   L - koeficijente Lagrangeovog interpolacionog polinoma

function L=interp1lagrange(x,y)

% Provera korektnosti argumenata
if nargin~=2 | min(size(x))~=1 | min(size(y))~=1 | length(x)~=length(y)
    error('Ocekivana su 2 argumenta koja predstavljaju tablicu interpolacije')
end

if (~all(finite(x)) | ~all(finite(y)) | ~all(isreal(x)) | ~all(isreal(y)))
    error('Tablica interpolacije mora da sadrzi iskljucivo konacne, realne brojeve')
end

if (length(x)~=length(unique(x)))
    error('Svi cvorovi interpolacije moraju biti razliciti')
end

% Interpolacioni polinom gradimo direktno na osnovu definicione
% konstrukcije. Rezultujuci polinom predstavlja zbir polinoma pi koji
% imaju nule u svim cvorovima interpolacije osim u xi u kojem imaju
% vrednost yi.

% Broj cvorova interpolacije
n = length(x);

% vektor L predstavlja koeficijente Lagranzevog i.p.
L=zeros(1, n);

% Prolazimo kroz sve cvorove interpolacije
for i=1:n

    % p je polinom koji ima nule u x1,...,xi-1,xi+1,...,xn
    % tj. (x-x1)...(x-xi-1)(x-xi+1)...(x-xn)
    p=poly(x(1:n ~= i));

    % na rezultujuci polinom dodajemo polinom pi =  p*yi/p(xi)
    L= L + p*y(i)/polyval(p,x(i));
end
```

Kao što se vidi, većina koda odlazi na standardnu dokumentaciju. Zatim sledi blok u kome se vrši iscrpna provera korektnosti argumenata funkcije. Sama implementacija metode za interpolaciju se intenzivno bazira na funkcionalnosti već gotovih rutina za rad sa polinomima i na logičkom indeksiranju vektora. Ključno mesto u konstrukciji interpolacionog polinoma je izgradnja polinoma $p = \prod_{j \neq i}(x - x_j)$. Ovi polinomi se grade koristeći funkciju `poly` kojoj se kao argumenti predaju vektori koji sadrži tačke x_j za $j \neq i$. Za formiranje ovakvih vektora korišćeno je logičko indeksiranje odnosno konstrukcija `x(1:n ~= i)`. Izraz `1:n` gradi vektor celih brojeva od `1` do `n`. Poređenjem sa `i` dobija se logički vektor koji ima vrednost netačno isključivo na poziciji `i`. Dobijeni polinomi u tački x_i imaju vrednost `polyval(p, x(i))`. Pošto želimo da ta vrednost bude `y(i)`, polinome normiramo deljenjem sa `polyval(p, x(i))` i množenjem sa `y(i)`.

2.2 Newton-ov polinom sa podeljenim razlikama

Podeljene razlike:

$$f[x_k] = f(x_k) \qquad \text{reda } 0$$

$$f[x_k, x_{k+1}] = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} \qquad \text{reda } 1$$

$$f[x_k, x_{k+1}, x_{k+2}] = \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k} \qquad \text{reda } 2$$

$$\dots \qquad \dots \qquad \dots \qquad \dots$$

$$f[x_k, \dots, x_{k+n}] = \frac{f[x_{k+1}, \dots, x_{k+n}] - f[x_k, \dots, x_{k+n-1}]}{x_{k+n} - x_k} \qquad \text{reda } n$$

$$f[x_0, \dots, x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}$$

Newton-ov interpolacioni polinom sa podeljenim razlikama je

$$L_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)\,(x - x_1) + \cdots + f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1})$$

Greška interpolacije izražena pomoću podeljenih razlika

$$R_n(x) = f[x, x_0, \dots, x_n]\omega_{n+1}(x)$$

2.14 *Neka je* $L_n(x) = \sum_{k=0}^{n} l_k(x)\, f(x_k)$ *Lagrange-ov interpolacioni polinom stepena* n *funkcije* $f(x)$. *Dokazati da je*

$$l_0(x) = 1 + \frac{x - x_0}{x_0 - x_1} + \frac{(x - x_0)(x - x_1)}{(x_0 - x_1)(x_0 - x_2)} + \cdots + \frac{(x - x_0) \dots (x - x_{n-1})}{(x_0 - x_1) \dots (x_0 - x_n)}.$$

Rešenje: Funkcija $l_0(x)$ je polinom n-tog stepena takav da je $l_0(x_0) = 1$ i $l_0(x_i) = 0$, $i > 0$. Podeljene razlike ove funkcije su

$$l_0[x_0, x_1] = \frac{l_0(x_1) - l_0(x_0)}{x_1 - x_0} = \frac{1}{x_0 - x_1}$$

$$l_0[x_0, x_1, x_2] = \frac{1}{x_2 - x_0} \left(\frac{l_0(x_2) - l_0(x_1)}{x_2 - x_1} - \frac{l_0(x_1) - l_0(x_0)}{x_1 - x_0} \right) = \frac{1}{(x_0 - x_1)(x_0 - x_2)}$$

$$\dots \qquad \dots \qquad \dots$$

$$l_0[x_0, \dots, x_n] = \sum_{i=0}^{n} \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{1}{x_i - x_j} \right) l_0(x_i) = \prod_{j=1}^{n} \frac{1}{x_0 - x_j}$$

Newton-ov interpolacioni polinom stepena n za funkciju $l_0(x)$ identičan je samoj funkciji jer je polinom stepena n sa $(n+1)$-im uslovom interpolacije jedinstveno određen. Stoga je

$$l_0(x) = l_0(x_0) + (x - x_0)l_0[x_0, x_1] + \cdots + (x - x_0)\dots(x - x_{n-1})l_0[x_0, \dots, x_n],$$

odakle sledi tvrđenje.

2.15 *Izračunati $f(1.16)$ za funkciju datu tabelom*

x	1.1275	1.1503	1.1735	1.1972
$f(x)$	0.11971	0.13957	0.15931	0.17902

Rešenje: Traženu vrednost ćemo izračunati pomoću Newton-ovog interpolacionog polinoma sa podeljenim razlikama

$$L_3(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + (x - x_0)(x - x_1)(x - x_2)f[x_0, x_1, x_2, x_3]$$

određenog sledećom tabelom podeljenih razlika

x	f	$f[1]$	$f[2]$	$f[3]$
1.1275	0.11971			
		0.871053		
1.1503	0.13957		−0.438935	
		0.850862		0.418978
1.1735	0.15931		−0.409732	
		0.831646		
1.1972	0.17902			

Red podeljene razlike je naznačen brojem u uglastoj zagradi u zaglavlju tabele, a čvorovi kojima je generisana određeni su položajem vrednosti podeljene razlike u tabeli.

Koeficijenti polinoma su prve podeljene razlike u svakoj koloni tabele. Tako se dobija da je

$$f(1.16) \approx L_3(1.16) = 0.14788.$$

2.16 *Dokazati da se greška interpolacije funkcije $f(x) = 1/(1+x)$ Lagrange-ovim interpolacionim polinomom $L_n(x)$ određenim čvorovima $x_k = k$, $k = 0, \dots, n$, može oceniti izrazom*

$$\left| \frac{1}{1+x} - L_n(x) \right| \leq \frac{1}{(n+1)!} |x(x-1)\dots(x-n)|.$$

Rešenje: Dokažimo indukcijom da je za datu funkciju

$$f[x_0,\dots,x_m]=\frac{(-1)^m}{(x_0+1)\dots(x_m+1)}.$$

Za $m=1$ je

$$f[x_0,x_1]=\frac{1}{x_1-x_0}\left(\frac{1}{x_1+1}-\frac{1}{x_0+1}\right)=\frac{-1}{(x_0+1)\,(x_1+1)}.$$

Neka tvrđenje važi za $m=n$,

$$f[x_0,\dots,x_n]=\frac{(-1)^n}{(x_0+1)\dots(x_n+1)}.$$

Dokažimo da važi za $m=n+1$:

$$\begin{aligned}f[x_0,\dots,x_{n+1}]&=\frac{1}{x_{n+1}-x_0}\big(f[x_1,\dots,x_{n+1}]-f[x_0,\dots,x_n]\big)\\&=\frac{1}{x_{n+1}-x_0}\left(\frac{(-1)^n}{(x_1+1)\dots(x_{n+1}+1)}-\frac{(-1)^n}{(x_0+1)\dots(x_n+1)}\right)\\&=\frac{(-1)^{n+1}}{(x_0+1)\dots(x_{n+1}+1)}.\end{aligned}$$

Ako sada u formuli za grešku Lagrange-ovog interpolacionog polinoma

$$R_n(x)=f(x)-L_n(x)=f[x_0,\dots,x_n,x]\,(x-x_0)\dots(x-x_n)$$

zamenimo $x_k=k$, $k=0,\dots,n$, i iskoristimo izvedeni izraz za podeljene razlike reda $n+1$ u kome je $x_{n+1}=x$, dobijamo da je

$$R_n(x)=\frac{(-1)^{n+1}}{(n+1)!(x+1)}\,x(x-1)(x-2)...(x-n).$$

Kako je još $\max_{[0,n]}1/(x+1)=1$ sledi da je

$$|R_n(x)|\le\frac{1}{(n+1)!}\,|x(x-1)...(x-n)|,$$

čime je dokazano tvrđenje zadatka.

2.17 *Ako je $f(x)=1/(a-x)$ dokazati da je*

$$f[x_0,x_1,\dots,x_n]=\frac{1}{(a-x_0)(a-x_1)\dots(a-x_n)}$$

i, na osnovu toga,

$$\frac{1}{a-x}=\frac{1}{a-x_0}+\frac{x-x_0}{(a-x_0)(a-x_1)}+\dots+\frac{(x-x_0)\dots(x-x_{n-1})}{(a-x_0)\dots(a-x_n)}+R(x),$$

gde je

$$R(x) = \frac{\omega_{n+1}(x)}{(a-x)\omega_{n+1}(a)}, \qquad \omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i).$$

Rešenje: Matematičkom indukcijom se neposredno dokazuje izraz za podeljenu razliku. Zamenom dobijenog izraza u Newton-ov interpolacioni polinom sa podeljenim razlikama i izraz za grešku,

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

dobija se traženi izraz za funkciju $1/(a - x)$.

2.18 *Dokazati da je za funkciju $f(x) = 1/(x-h)$, gde je h kompleksan broj, greška Lagrange-ove interpolacije polinomom $L_n(x)$ stepena n*

$$\frac{1}{x-h} - L_n(x) = \frac{1}{x-h}\,\frac{x-x_0}{h-x_0}\cdots\frac{x-x_n}{h-x_n}.$$

Rešenje: Tvrđenje se dokazuje kao u prethodnom zadatku.

2.19 *Ako je $f(x) = u(x)\,v(x)$, dokazati da je*

$$f[x_0, x_1] = u[x_0]\,v[x_0, x_1] + u[x_0, x_1]\,v[x_1],$$

i, uopšte,

$$f[x_0, \ldots, x_n] = \sum_{k=0}^{n} u[x_0, \ldots, x_k]\,v[x_k, \ldots, x_n].$$

Rešenje: Zadatak ćemo dokazati matematičkom indukcijom. Za $n = 1$ tvrđenje sledi iz definicije podeljene razlike prvog reda,

$$\begin{aligned} f[x_0, x_1] &= \frac{1}{x_1 - x_0}\big(u(x_1)v(x_1) - u(x_0)v(x_0)\big) \\ &= \frac{u(x_1) - u(x_0)}{x_1 - x_0}\,v[x_1] + u[x_0]\,\frac{v(x_1) - v(x_0)}{x_1 - x_0}. \end{aligned}$$

Pretpostavimo da tvrđenje važi za $n = m$,

$$f[x_0, \ldots, x_m] = \sum_{k=0}^{m} u[x_0, \ldots, x_k]\,v[x_k, \ldots, x_m].$$

Dokažimo da važi za $n = m + 1$. Prema definiciji podeljene razlike reda $m + 1$ je

$$f[x_0, \ldots, x_{m+1}] = \frac{1}{x_{m+1} - x_0}\big(f[x_1, \ldots, x_{m+1}] - f[x_0, \ldots, x_m]\big),$$

te je, uzimajući u obzir indukcijsku hipotezu,

$$f[x_0,\dots,x_{m+1}] = \frac{1}{x_{m+1}-x_0}\Bigg(\sum_{k=1}^{m+1} u[x_1,\dots,x_k]\,v[x_k,\dots,x_{m+1}] - \sum_{k=0}^{m} u[x_0,\dots,x_k]\,v[x_k,\dots,x_m]\Bigg)$$

$$= \frac{1}{x_{m+1}-x_0}\Bigg(\sum_{k=0}^{m} u[x_1,\dots,x_{k+1}]\,v[x_{k+1},\dots,x_{m+1}] - \sum_{k=0}^{m} u[x_0,\dots,x_k]\,v[x_k,\dots,x_m]\Bigg)$$

Koristeći vezu

$$u[x_1,\dots,x_{k+1}] = (x_{k+1}-x_0)\,u[x_0,\dots,x_{k+1}] + u[x_0,\dots,x_k]$$

dalje dobijamo da je

$$f[x_0,\dots,x_{m+1}] = \frac{1}{x_{m+1}-x_0}\Bigg(\sum_{k=0}^{m}(x_{k+1}-x_0)\,u[x_0,\dots,x_{k+1}]\,v[x_{k+1},\dots,x_{m+1}] + \sum_{k=0}^{m}(x_{m+1}-x_k)\,u[x_0,\dots,x_k]\,v[x_k,\dots,x_{m+1}]\Bigg)$$

tj.

$$f[x_0,\dots,x_{m+1}] = \frac{1}{x_{m+1}-x_0}\Bigg(\sum_{k=1}^{m+1}(x_k-x_0)\,u[x_0,\dots,x_k]\,v[x_k,\dots,x_{m+1}] + \sum_{k=0}^{m}(x_{m+1}-x_k)\,u[x_0,\dots,x_k]\,v[x_k,\dots,x_{m+1}]\Bigg)$$

Tvrđenje neposredno sledi posle sabiranja odgovarajućih sabiraka u poslednje dve sume.

2.20 *Ako je* $x_k = x_0 + kh$, $k = 0,\dots,n$, *dokazati da je*

$$f[x_0,\dots,x_n] = \frac{1}{h^n n!}\sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k} f(x_k).$$

Rešenje: Tvrđenje neposredno sledi iz izraza za podeljene razlike

$$f[x_0,\dots,x_n] = \sum_{k=0}^{n}\Bigg(\prod_{\substack{i=0\\ i\neq k}}^{n}(x_k-x_i)\Bigg)^{-1} f(x_k),$$

i

$$\left(\prod_{\substack{i=0\\ i\neq k}}^{n}(x_k-x_i)\right)^{-1}=\frac{1}{k!h^k}\,\frac{(-1)^{n-k}}{(n-k)!h^{n-k}}=\frac{(-1)^{n-k}}{n!h^n}\binom{n}{k}.$$

Tvrđenje se može dokazati i korišćenjem veze podeljene i konačne razlike

$$f[x_0,\ldots,x_n]=\frac{\Delta^n f_0}{n!h^n}$$

i izraza

$$\Delta^n f_0=\sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k}f(x_k).$$

2.21 *Ako je* $f'(x)=\frac{df}{dx}$ *dokazati da je* $\frac{d}{dx}f[x_0,x]\neq f'[x_0,x]$, *osim ako je* $f(x)$ *linearna funkcija.*

Rešenje: Kako je prema definiciji podeljene razlike

$$\frac{d}{dx}f[x_0,x]=\frac{d}{dx}\left(\frac{f(x)-f(x_0)}{x-x_0}\right)=\frac{1}{x-x_0}\big(f'(x)-f[x_0,x]\big)$$

i

$$f'[x_0,x]=\frac{1}{x-x_0}\big(f'(x)-f'(x_0)\big)$$

očigledno je da je u opštem slučaju $f[x_0,x]\neq f'(x_0)$. Za linearnu funkciju $f(x)=ax+b$ je $f[x_0,x]=f'(x_0)=a$. Time je tvrđenje dokazano.

2.22 *Ako je* $f(x)=(ax+b)/(cx+d)$, *izračunati* $f[x,y]$, $f[x,x,y]$ *i* $f[x,x,y,y]$.

Rešenje: Prema definiciji je

$$f[x,y]=\frac{1}{y-x}\left(\frac{ay+b}{cy+d}-\frac{ax+b}{cx+d}\right)=\frac{ad-bc}{(cx+d)(cy+d)}.$$

Iz veze podeljene razlike i izvoda funkcije

$$f[\underbrace{x,\ldots,x}_{n+1}]=\frac{1}{n!}\,f^{(n)}(x)$$

sledi da je

$$f[x,x,y]=\frac{\partial}{\partial x}f[x,y]=\frac{(bc-ad)c}{(cx+d)^2(cy+d)}$$

i

$$f[x,x,y,y]=\frac{\partial}{\partial y}f[x,x,y]=\frac{(ad-bc)c^2}{(cx+d)^2(cy+d)^2}.$$

MATLAB

Konačne razlike
```
dy = diff(y)
```

Podeljene razlike
```
differences = y;
for i=2:n
   differences = diff(differences) ./ (x(i:n)-x(1:n-i+1));
end
```

2.23 *Korišćenjem* MATLAB-a, *konstruisati potrebne podeljene razlike* $f[x_1, \ldots, x_i]$ *i Newton-ov interpolacioni polinom sa podeljenim razlikama stepena tri za funkciju zadatu tabelom*

x	0	0.3	0.7	1	1.3	1.7	2	2.3	2.7	3
$f(x)$	0	0.3	0.65	0.84	0.96	1	0.9	0.75	0.43	0.14

Rešenje: Za početak, potrebno je izgraditi odgovarajuće vrednosti tabele podeljenih razlika. Pošto je stepen polinoma tri, dovoljno je razmatrati samo četiri početna čvora. Za računanje proizvoljne kolone tabele podeljenih razlika potrebno je poznavati isključivo vrednosti prethodne kolone, te nije potrebno pamtiti celokupnu tabelu podeljenih razlika. Prilikom implementacije, čuva se samo vektor `differences`, koji sadrži redom kolone tabele podeljenih razlika. Za konstrukciju polinoma, dovoljno je znati samo podeljene razlike oblika $f[x_1, x_2, \ldots, x_i]$ tj. samo prve elemente svake od kolona tabele. Odabrano je da se prve vrednosti svake kolone tabele podeljenih razlika (koje su nam potrebne za izgradnju polinoma) čuvaju na početku vektora `differences`. Preciznije, u svakom koraku i pretpostavljamo da vektor `differences` ima oblik

$$(f[x_1], f[x_1, x_2], \ldots, f[x_1, x_2, \ldots, x_{i-1}],$$
$$f[x_1, x_2, \ldots, x_i], f[x_2, x_3, \ldots, x_{i+1}], \ldots, f[x_{n-i+1}, x_1, \ldots, x_n])$$

Pri izvođenju sledećeg koraka prvih i elemenata ovog vektora ostaju nepromenjeni, dok se podeljene razlike $f[x_2, x_3, \ldots, x_{i+1}], \ldots, f[x_{n-i+1}, x_1, \ldots, x_n]$ zamenjuju razlikama $f[x_1, x_2, \ldots, x_j], \ldots, f[x_{n-j+1}, x_1, \ldots, x_n]$, za $j = i + 1$.
Kada su konstruisane podeljene razlike, potrebno je izgraditi polinom

$$P = f(x_1) + f[x_1, x_2](x - x_1) + \ldots + f[x_1, \ldots, x_n](x - x_1)(x - x_2)(x - x_{n-1})$$

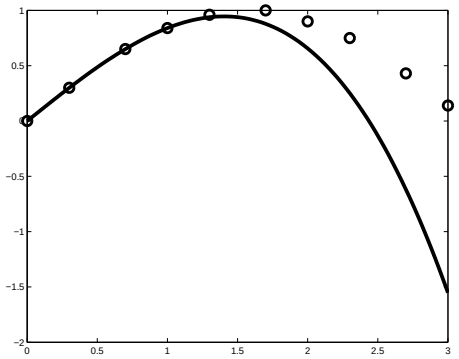
Ovaj polinom gradimo postupkom sličnim Horner-ovoj šemi

$$\begin{aligned} P_{n-1} &= f[x_1, x_2, \ldots, x_n] \\ P_{n-2} &= (x - x_{n-1})P_{n-1} + f[x_1, x_2, \ldots, x_{n-1}] \\ &\ldots \\ P_1 &= (x - x_2)P_2 + f[x_1, x_2] \\ P &= (x - x_1)P_1 + f(x_1) \end{aligned}$$

Ovo govori da ćemo polinom P izgraditi počevši od polinoma P_{n-1}, množeći u svakom koraku tekući polinom polinomom $(x - x_i)$ i dodajući odgovarajuću podeljenu razliku $f[x_1, x_2, \ldots, x_i]$.

```
% Tablica interpolacije
x = [0 0.3 0.7 1 1.3 1.7 2 2.3 2.7 3];
y = [0 0.3 0.65 0.84 0.96 1 0.9 0.75 0.43 0.14];

% Broj potrebnih cvorova za konstrukciju polinoma treceg stepena
n = 3 + 1;

differences = y(1:n);

for i=2:n
    % differences(i:n) postaje sledeca kolona tablice podeljenih razlika
    % pri cemu differences(1:i-1) zadrzava, redom, prve elemente prethodnih
    % kolona tablice podeljenih razlika
    differences(i:n) = (differences(i:n)-differences(i-1:n-1)) ./ ...
                       (x(i:n)-x(1:n-i+1))
end

% Prilikom konstruisanja polinoma P, koristimo postupak slican Hornerovoj semi.
P = differences(n);
for i = n-1:-1:1
    %   P = P*(x-xi) + f[x1,...,xi]
    P = conv(P, [1 -x(i)]);
    P(end) = P(end) + differences(i);
end

% Mreza za crtanje
X = linspace(x(1), x(end));
% Iscrtavamo cvorove i rezultat
plot(x, y, 'o', X, polyval(P, X));
```

Dobijeno rešenje, prikazano na slici 2.2, pokazuje kako kvalitet interpolacije za mali stepen polinoma opada sa udaljavanjem od intervala određenog čvorovima interpolacije. Povećavanjem stepena interpolacioni polinom se približava Lagrangeovom polinomu.



Slika 2.2: Newton-ov interpolacioni polinom

2.3 Polinomi sa ekvidistantnim čvorovima

Ravnomerno raspoređeni (ekvidistantni) čvorovi, sa korakom h, su

$$x_k = x_0 + kh, \qquad k = 0, \pm 1, \pm 2, \ldots, \qquad\qquad x = x_0 + qh, \quad |q| < 1$$

Konačne razlike se definišu rekurentnim formulama

$$\begin{aligned}
&\Delta f_k = f(x_{k+1}) - f(x_k) && \text{reda } 1\\
&\Delta^2 f_k = \Delta f_{k+1} - \Delta f_k = f(x_{k+2}) - 2f(x_{k+1}) + f(x_k) && \text{reda } 2\\
&\cdots \quad \cdots \quad \cdots\\
&\Delta^n f_k = \Delta^{n-1} f_{k+1} - \Delta^{n-1} f_k = \sum_{j=0}^{n} (-1)^j \binom{n}{j} f(x_{k+n-j}) && \text{reda } n
\end{aligned}$$

Veza podeljenih razlika, konačnih razlika i izvoda funkcije je

$$f[x_k, \ldots, x_{k+n}] = \frac{\Delta^n f_k}{h^n n!} = \frac{f^{(n)}(\xi)}{n!}, \qquad \xi \in [x_k, x_{k+n}]$$

Newton-ov polinom za interpolaciju unapred

$$L_n(x_0 + qh) = f(x_0) + \sum_{k=1}^{n} \frac{q(q-1)\ldots(q-k+1)}{k!} \Delta^k f_0$$

Newton-ov polinom za interpolaciju unazad

$$L_n(x_0 + qh) = f(x_0) + \sum_{k=1}^{n} \frac{q(q+1)\ldots(q+k-1)}{k!} \Delta^k f_{-k}$$

Stirling-ov interpolacioni polinom

$$\begin{aligned}
L_{2n+1}(x_0 + qh) = f_0 &+ q\frac{\Delta f_{-1} + \Delta f_0}{2} + \frac{q^2}{2!}\Delta^2 f_{-1}\\
&+ \cdots + \frac{q(q^2-1)\ldots(q^2-n^2)}{(2n+1)!} \, \frac{\Delta^{2n+1} f_{-(n+1)} + \Delta^{2n+1} f_{-n}}{2}
\end{aligned}$$

Bessel-ov interpolacioni polinom

$$\begin{aligned}
L_{2n+1}(x_0 + qh) = \frac{f_0 + f_1}{2} &+ (q - 0.5)\Delta f_0 + \frac{q(q-1)}{2!} \, \frac{\Delta^2 f_{-1} + \Delta^2 f_0}{2}\\
&+ \cdots + \frac{q(q^2-1)\ldots(q^2-(n-1)^2)(q-n)(q-0.5)}{(2n+1)!} \Delta^{2n+1} f_{-n}
\end{aligned}$$

2.24 *Neka je $f(x)$ funkcija takva da je $f(kh) = 0$, $k = \pm 1, \pm 2, \ldots$, i $f(0) = 1$. Dokazati da je Newton-ov interpolacioni polinom te funkcije*

$$L_n(x) = 1 - \frac{x}{1!h} + \frac{x(x-h)}{2!h^2} + \cdots + (-1)^n \frac{x(x-h)\ldots(x-(n-1)h)}{n!h^n}$$

Rešenje: Tvrđenje neposredno sledi kada se uzme da je $x_0 = 0$, tj. $q = x/h$ u formuli

$$\Delta^k f_0 = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} f(x_i) = (-1)^k,$$

2.25 *Integral verovatnoće*

$$f(x) = \sqrt{\frac{2}{\pi}} \int_0^x e^{-t^2/2}\, dt$$

dat je sledećom tabelom

x	1.00	1.05	1.10	1.15	1.20	1.25
$f(x)$	0.682689	0.706282	0.728668	0.749856	0.769861	0.788700

Izračunati $f(1.235)$ i oceniti grešku rezultata.

Rešenje: Tabela konačnih razlika je

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$
1.00	0.682689			
		0.023593		
1.05	0.706282		−0.001207	
		0.022386		0.000009
1.10	0.728668		−0.001198	
		0.021188		0.000015
1.15	0.749856		−0.001183	
		0.020005		<u>0.000017</u>
1.20	0.769861		<u>−0.001166</u>	
		<u>0.018839</u>		
1.25	<u>0.788700</u>			

Tačka u kojoj se traži približna vrednost integrala verovatnoće pripada intervalu određenom sa poslednja dva čvora u tabeli. Stoga ćemo koristiti Newton-ov polinom za interpolaciju unazad

$$L_3(x_0 + qh) = f_0 + q\Delta f_{-1} + \frac{q(q+1)}{2}\Delta^2 f_{-2} + \frac{q(q+1)(q+2)}{6}\Delta^3 f_{-3},$$

gde je $q = (x - x_0)/h$. Stavljajući da je $x_0 = 1.25$, $x = 1.235$ i $h = 0.05$ i koristeći podvučene konačne razlike u tabeli, dobijamo da je

$$f(1.235) \approx L_3(1.235) = 0.783170.$$

Greška interpolacije se grubo može oceniti tako što se u Lagrange-ovom izrazu za grešku izvod funkcije aproksimira srednjom vrednošću odgovarajuće konačne razlike,

$$|R_3(x)| = |\frac{1}{24}q(q+1)(q+2)(q+3)h^4 f^{(4)}(\xi)|$$
$$\approx |\frac{1}{24}q(q+1)(q+2)(q+3)\Delta^4 f_{sr}| = 1.6 * 10^{-7}$$

2.26 *Dat je deo dijagonalne tabele za funkciju* $f(x) = \sin x$

x	$f(x)$	$\Delta^2 f$	$\Delta^4 f$
0.9	0.78333	-0.03123	0.00125
1.1	0.89121	-0.03553	0.00141
1.3	0.96356	-0.03842	0.00158

Izračunati $f(1)$ *sa tačnošću* $5 \cdot 10^{-5}$.

Rešenje: Kako $1 \in (0.9, 1.1)$, uzećemo da je $x_0 = 0.9$. Tada je $q = 0.5$, pa je greška interpolacije funkcije $f(x)$ u tački $x = 1$ Bessel-ovim interpolacionim polinomom trećeg stepena

$$|R_3(1)| \leq \frac{1}{4!} \max |f^{(4)}| \, |0.5(0.5^2 - 1)(0.5 - 2)| \, 0.2^4 \leq 4 \cdot 10^{-5},$$

pri čemu je uzeto da je $\max |f^{(4)}(x)| = 1$. Dakle, Bessel-ovim interpolacionim polinomom $L_3(x)$ možemo izračunati sa zadovoljavajućom tačnošću traženu vrednost. Kako je $q = 0.5$, za izračunavanje su nam potrebne samo konačne razlike parnog reda koje su date u tabeli. Primenom formule

$$L_3(x_0 + qh) = \frac{1}{2}(f_0 + f_1) + (q - 0.5)\Delta f_0$$
$$+ \frac{q(q-1)}{2!} \frac{\Delta^2 f_{-1} + \Delta^2 f_0}{2} + \frac{q(q-0.5)(q-1)}{3!}\Delta^3 f_{-1}$$

dobijamo rezultat sa traženom tačnošću

$$f(1) \approx L_3(1) = 0.8415.$$

2.27 *Dokazati da greška kvadratne interpolacije, određene ravnomerno raspoređenim čvorovima* x_0, x_1 *i* x_2 *sa korakom* h, *nije veća od*

$$|R_2(x)| \leq \frac{h^3}{9\sqrt{3}} M_3, \qquad M_3 = \max_{[x_0, x_2]} |f'''(x)|.$$

Takođe pokazati da se maksimalna greška može očekivati na rastojanju $h/\sqrt{3}$ *od centralne apscise.*

Rešenje: Greška kvadratne interpolacije je

$$R_2(x) = \frac{1}{3!} f'''(\xi)\, \omega_3(x), \qquad \xi \in [x_0, x_2],$$

gde je $\omega_3(x) = (x - x_0)(x - x_1)(x - x_2)$. Smenom $x = x_1 + qh$ polinom $\omega_3(x)$ se može napisati u obliku $\omega(x_1 + qh) = h^3 q(q^2 - 1)$. Očigledno je da on dostiže ekstremne vrednosti

$$\omega(x_e) = \pm \frac{2}{3\sqrt{3}} h^3 \qquad \text{za} \qquad x_e = x_1 \pm \frac{h}{\sqrt{3}}.$$

2.28 *Neka je* $e^x/(x+2) \le f^{(4)}(x) \le e^x/(x+1)$ *i* $f'''(1) = 0.76432$. *Sa kojim korakom* h *treba tabelirati funkciju* $f(x)$ *na intervalu* $[1, 2]$ *tako da tabela dozvoljava kvadratnu interpolaciju sa tačnošću* $\varepsilon = 5 \cdot 10^{-7}$.

Rešenje: Prema prethodnom zadatku greška kvadratne interpolacije određene ravnomerno raspoređenim čvorovima se ocenjuje izrazom

$$|R_2(x)| \le \frac{h^3}{9\sqrt{3}}\, M_3, \qquad\qquad M_3 = \max_{[x_0, x_2]} |f'''(x)|.$$

Da bismo odredili konstantu M_3 pođimo od teoreme o srednjoj vrednosti

$$f'''(x) = f'''(1) + (x-1) f^{(4)}(\xi), \qquad 1 \le \xi \le x \le 2,$$

koja, koristeći pretpostavku zadatka, daje ocenu

$$f'''(1) + (x-1)\frac{e^\xi}{\xi + 2} \le f'''(x) \le f'''(1) + (x-1)\frac{e^\xi}{\xi + 1}, \qquad \text{tj.} \qquad M_3 = 3.23.$$

Zamenom u izraz za ocenu greške dobijamo uslov za h

$$|R_2(x)| \le \frac{3.23}{9\sqrt{3}}\, h^3 \le 5 \cdot 10^{-7} \qquad \text{za} \qquad h \le 0.013.$$

2.29 *Koliki treba da bude korak* h *u tabeli za funkciju* $f(x) = x^{1/3}$ *da bi na intervalu* $[1, 1000]$ *greška: a) linearne, b) kvadratne interpolacije, bila manja od* $5 * 10^{-4}$. *Analizirati dve mogućnosti*

i) Na celom intervalu $[1, 1000]$ *koristi se isti korak* h;

ii) Na intervalu $[1, 100]$ *koristi se korak* h_1, *a na intervalu* $[100, 1000]$ *korak* h_2.

Rešenje: a) Greška linearne interpolacije na intervalu $[x_k, x_{k+1}]$ se ocenjuje izrazom

$$|R_1(x)| \le \frac{1}{2}\frac{(x_{k+1} - x_k)^2}{4} M_2, \qquad M_2 = \max_{[x_k, x_{k+1}]} |f''(x)|.$$

(i) $M_2 = \max_{[1,1000]} |2x^{-5/3}/9| = 2/9$ pa je

$$|R_1| \le \frac{1}{2}\frac{2}{9}\frac{h^2}{4} \le 5 \cdot 10^{-4} \qquad \text{za} \qquad h \le 0.13.$$

(ii) U intervalu $[1, 100]$ ostaje ista procena koraka $h_1 \le 0.13$, a u intervalu $[100, 1000]$ je $M_2 = 2 \cdot 100^{-5/3}/9$, pa je $h_2 \le 6.23$.

b) Greška kvadratne interpolacije na intervalu $[x_k, x_{k+2}]$ se ocenjuje, prema prethodnom zadatku, izrazom

$$|R_2| \le \frac{h^3}{9\sqrt{3}} M_3, \qquad M_3 = \max_{[x_k, x_{k+2}]} |f'''(x)|.$$

(i) $M_3 = \max_{[1,1000]} |10x^{-8/3}/27| = 10/27$ pa je

$$|R_2| \le \frac{10}{27}\frac{1}{9\sqrt{3}} h^3 \le 5 \cdot 10^{-4} \qquad \text{za} \qquad h \le 0.27.$$

(ii) U intervalu $[1, 100]$ ostaje ista procena koraka $h_1 \le 0.27$, a u intervalu $[100, 1000]$ je $M_3 = 10 \cdot 100^{-8/3}/27$, pa je $h_2 \le 16.55$.

2.30 *Ako je Δ oznaka za operator konačne razlike unapred, $\Delta f_k = f_{k+1} - f_k$, a δ oznaka za operator centralne konačne razlike $\delta f_k = f_{k+1/2} - f_{k-1/2}$, dokazati da važi*

$$\Delta = \frac{1}{2}\delta^2 + \delta\sqrt{1 + \frac{1}{4}\delta^2}.$$

Rešenje: Transformišimo prvo dati izraz kvadriranjem

$$\big(\Delta - \frac{1}{2}\delta^2\big)^2 = \delta^2\big(1 + \frac{1}{4}\delta^2\big), \qquad \text{tj.} \qquad \Delta^2 = \delta^2 + \Delta\,\delta^2 = (1 + \Delta)\,\delta^2.$$

Sada je

$$\begin{aligned}(1 + \Delta)\delta^2 f_k &= (1 + \Delta)\,(f_{k+1} - 2f_k + f_{k-1}) \\ &= f_{k+1} - 2f_k + f_{k-1} + f_{k+2} - f_{k+1} - 2(f_{k+1} - f_k) + f_k - f_{k-1} \\ &= \Delta^2 f_k.\end{aligned}$$

2.31 *Neka je $h = 1/n$, $x_i = ih$, $i = 0, \ldots, n$, i neka je*

$$\phi(t) = \begin{cases} 1 - |t|, & |t| \le 1, \\ 0, & |t| > 1 \end{cases}$$

Dokazati da je za svako $f(x) \in \mathcal{C}^2(0, 1)$

$$\Delta^2 f_{k-1} = f(x_{k+1}) - 2f(x_k) + f(x_{k-1}) = \int_0^1 \phi_k(t) f''(t)\,dt,$$

gde je $\phi_k(t) = h\phi\big((t - x_k)/h\big)$.

Rešenje: Funkcija $\phi_k(t)$ je

$$\phi_k(t) = \begin{cases} h - x_k + t, & t \in [x_{k-1}, x_k], \\ h + x_k - t, & t \in [x_k, x_{k+1}] \\ 0, & t \notin [x_{k-1}, x_{k+1}] \end{cases}$$

te je

$$\int_0^1 \phi_k(t) f''(t)\, dt = \int_{x_{k-1}}^{x_k} (h - x_k + t) f''(t)\, dt + \int_{x_k}^{x_{k+1}} (h + x_k - t) f''(t)\, dt,$$

što integracijom neposredno daje tvrđenje zadatka.

2.4 Drugi vidovi interpolacije

U ovom odeljku su samo informativno, kroz primere, date interpolacije trigonometrijskim i racionalnim funkcijama, inverzna interpolacija i interpolacija funkcija više promenljivih.

2.32 *Interpolacijom periodične funkcije $f(x)$ trigonometrijskim polinomom*

$$Q(x) = c_0 + c_1 \cos(2x) + c_2 \sin(2x),$$

koji zadovoljava uslove interpolacije $Q(x_k) = f(x_k)$, $k = 0, 1, 2$, izvesti Hermite-ovu formulu za trigonometrijsku interpolaciju.

Rešenje: Uslovi interpolacije određuju sistem linearnih jednačina po nepoznatim koeficijentima polinoma,

$$c_0 + c_1 \cos(2x_k) + c_2 \sin(2x_k) = f(x_k), \qquad k = 0, 1, 2.$$

Determinanta sistema je

$$D = \begin{vmatrix} 1 & \cos(2x_0) & \sin(2x_0) \\ 1 & \cos(2x_1) & \sin(2x_1) \\ 1 & \cos(2x_2) & \sin(2x_2) \end{vmatrix} = 4 \sin(x_1 - x_0) \sin(x_2 - x_0) \sin(x_2 - x_1) \neq 0$$

za $0 \leq x_0 < x_1 < x_2 < \pi$, te on ima netrivijalno rešenje. Ako ovom sistemu dodamo trigonometrijski polinom napisan u proizvoljnoj tački x različitoj od čvorova, dobijamo homogeni sistem jednačina

$$\begin{aligned} -Q(x) + c_0 + c_1 \cos(2x) + c_2 \sin(2x) &= 0, \\ -f(x_k) + c_0 + c_1 \cos(2x_k) + c_2 \sin(2x_k) &= 0, \qquad k = 0, 1, 2, \end{aligned}$$

čije je rešenje $(-1, c_0, c_1, c_2)^\top$ netrivijalno. Stoga determinanta sistema mora biti jednaka nuli,

$$\begin{vmatrix} Q(x) & 1 & \cos(2x) & \sin(2x) \\ f(x_0) & 1 & \cos(2x_0) & \sin(2x_0) \\ f(x_1) & 1 & \cos(2x_1) & \sin(2x_1) \\ f(x_2) & 1 & \cos(2x_2) & \sin(2x_2) \end{vmatrix} = 0.$$

Razvojem ove determinante po elementima prve kolone

$$Q(x)\begin{vmatrix} 1 & \cos(2x_0) & \sin(2x_0) \\ 1 & \cos(2x_1) & \sin(2x_1) \\ 1 & \cos(2x_2) & \sin(2x_2) \end{vmatrix} - f(x_0)\begin{vmatrix} 1 & \cos(2x) & \sin(2x) \\ 1 & \cos(2x_1) & \sin(2x_1) \\ 1 & \cos(2x_2) & \sin(2x_2) \end{vmatrix}$$

$$+f(x_1)\begin{vmatrix} 1 & \cos(2x) & \sin(2x) \\ 1 & \cos(2x_0) & \sin(2x_0) \\ 1 & \cos(2x_2) & \sin(2x_2) \end{vmatrix} - f(x_2)\begin{vmatrix} 1 & \cos(2x) & \sin(2x) \\ 1 & \cos(2x_0) & \sin(2x_0) \\ 1 & \cos(2x_1) & \sin(2x_1) \end{vmatrix} = 0,$$

i uzimajući u obzir izraz za determinantu D, dobijamo Hermite-ov trigonometrijski interpolacioni polinom (ovde je $n = 2$)

$$Q(x) = \sum_{k=0}^{n} \left(\prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{\sin(x - x_i)}{\sin(x_k - x_i)} \right) f(x_k).$$

2.33 *Interpolisati funkciju $f(x) = e^x$ racionalnom funkcijom $r(x) = (x+a)/(bx+c)$ takvom da je $r(k) = f(k)$, $k = 0, 1, 2$, i izračunati grešku $\max_{[0,2]} |f(x) - r(x)|$.*

Rešenje: Uslovi interpolacije određuju sistem linearnih jednačina po nepoznatim koeficijentima

$$\frac{a}{c} = 1, \qquad \frac{1+a}{b+c} = e, \qquad \frac{2+a}{2b+c} = e^2,$$

čije rešenje je $a = c = 2/(e-1)$, $b = -1/e$. Tražena racionalna funkcija i greška interpolacije su

$$r(x) = e\,\frac{2 + (e-1)x}{2e - (e-1)x}, \qquad \max_{[0,2]} |f(x) - r(x)| = |f(1.69) - r(1.69)| \leq 0.16.$$

2.34 *Funkcija je zadata tabelom*

x	10	15	17	20
$f(x)$	3	7	11	17

Odrediti argument X za koji je $f(X) = 10$.

Rešenje: Zadata tabela definiše Lagrange-ov interpolacioni polinom trećeg stepena inverzne funkcije funkciji $f(x)$. Ako, radi pojednostavljenja, uvedemo oznake $y = f(x)$ i $y_i = f(x_i)$, $i = 0, \ldots, 3$, taj polinom je

$$L_3(y) = \sum_{i=0}^{3} \left(\prod_{\substack{j=0 \\ j \neq i}}^{3} \frac{y - y_j}{y_i - y_j} \right) x_i.$$

Vrednost ovog polinoma u tački $Y = f(X) = 10$ je

$$\begin{aligned} L_3(x) = {} & \frac{(10-7)(10-11)(10-17)}{(3-7)(3-11)(3-17)}\,10 + \frac{(10-3)(10-11)(10-17)}{(7-3)(7-11)(7-17)}\,15 \\ & + \frac{(10-3)(10-7)(10-17)}{(11-3)(11-7)(11-17)}\,17 + \frac{(10-3)(10-7)(10-11)}{(17-3)(17-7)(17-11)}\,20 \;=\; 16.64. \end{aligned}$$

Dakle, $f(16.64) \approx 10$.

2.35 *Inverznom interpolacijom odrediti sa tačnošću* 10^{-4} *vrednost argumenta* x *za koju je* $f(x) = 0.99800$, *ako je funkcija data tabelom*

x	1.40	1.50	1.60	1.70	1.80
$f(x)$	0.98545	0.99749	0.99957	0.99166	0.97385

Rešenje: Formirajmo tabelu konačnih razlika funkcije $f(x)$

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$
1.40	0.98545			
		0.01204		
1.50	0.99749		−0.00996	
		0.00208		−0.00003
1.60	0.99957		−0.00999	
		−0.00791		0.00009
1.70	0.99166		−0.00990	
		−0.01781		
1.80	0.97385			

U okolini tačke $x = 1.60$ je maksimum funkcije $f(x)$, te postoje dve tačke takve da je $f(\xi_1) = f(\xi_2) = 0.99800$, pri čemu $\xi_1 \in [1.50, 1.60]$ i $\xi_2 \in [1.60, 1.70]$. Odredićemo ih kao rešenja jednačine $L_2(x) = 0.99800$, gde je $L_2(x)$ Stirling-ov interpolacioni polinom drugog stepena napisan u odnosu na čvor $x_0 = 1.60$. Rezultat će biti zadovoljavajuće tačnosti, jer je greška interpolacije ovim polinomom

$$|R_2(x_0 + qh)| \approx \left| \frac{q(q^2-1)}{6}\,\Delta^3 f_{sr} \right| \leq 10^{-5},$$

gde je $x = x_0 + qh = 1.60 + 0.1q$. Jednačina $L_2(x_0 + qh) = 0.99800$ se svodi na kvadratnu jednačinu $q^2 + 0.58358q - 0.31431 = 0$, čija rešenja određuju tražene tačke

$$\xi_1 = 1.6 - 0.1 \cdot 0.92381 = \underline{1.5076}, \qquad \xi_2 = 1.6 + 0.1 \cdot 0.34023 = \underline{1.6340}.$$

2.36 *Dat je deo dijagonalne tabele konačnih razlika za funkciju* $f(x)$

x	$f(x)$	$\Delta^2 f$
0.7	3.1519	0.0146
0.8	3.2176	0.0165

Izračunati što tačnije argument x *za koji je* $f(x) = 3.1942$.

Rešenje: Da bismo mogli napisati interpolacioni polinom maksimalnog stepena, koristeći definiciju konačnih razlika $\Delta^{k+1} f(x) = \Delta^k f(x+h) - \Delta^k f(x)$, dopunimo tabelu konačnih razlika funkcije $f(x)$

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$
0.6	3.1008			
		0.0511		
0.7	3.1519		0.0146	
		0.0657		0.0019
0.8	3.2176		0.0165	
		0.0822		
0.9	3.2998			

Zadata vrednost funkcije je u intervalu $[f(0.7), f(0.8)]$, pa ćemo koristiti Bessel-ov interpolacioni polinom

$$L_3(x_0 + qh) = \frac{3.1519 + 3.2176}{2} + (q - 0.5)0.0657 + \frac{q(q-1)}{2}\,\frac{0.0146 + 0.0165}{2} + \frac{q(q-0.5)(q-1)}{6}0.0019,$$

gde je $x_0 = 0.7$ i $q = (x - x_0)/h$. Tražena vrednost argumenta je približno rešenje jednačine $L_3(x) = 3.1942$. Metodom iteracije

$$q^{(n+1)} = 0.64384 - q^{(n)}(q^{(n)} - 1)0.11842 - q^{(n)}(q^{(n)} - 0.5)(q^{(n)} - 1)0.00487, \quad n = 0, \ldots, 3,$$

sa početnom aproksimacijom $q^{(0)} = 0.64384$, dobija se rešenje $q = 0.6702$. Tražena tačka je

$$x = 0.7 + 0.1 \cdot 0.6702 = 0.76702.$$

2.37 *Inverznom interpolacijom izračunati sa tačnošću* 10^{-4} *rešenje jednačine*

$$e^x = 30.$$

Rešenje: Definišimo funkciju $f(x) = e^x - 30$, i rešenja date jednačine su nule funkcije $f(x)$. Kako je $f'(x) = e^x > 0$ za svako x, a $f(-\infty) < 0$ i $f(\infty) > 0$, to $f(x)$ ima jednu realnu nulu i ona se nalazi u intervalu [3.4, 3.5]. Aproksimirajmo $f(x)$ u okolini nule Newton-ovim interpolacionim polinomom drugog stepena. Greška interpolacije je

$$|R_2(x)| = \frac{h^3}{3!}|f'''(\xi)|\,|q(q-1)(q-2)| \le \frac{33.12}{6}\,\frac{2\sqrt{3}}{9}\,h^3 \approx 2.2h^3,$$

jer je

$$\max_{x\in[3.4,3.5]} |f'''(x)| = e^{3.5} = 33.12, \qquad \max_{q\in[0,1]} |q(q-1)(q-2)| = \frac{2\sqrt{3}}{9}.$$

Iz uslova tačnosti $2.2h^3 < 10^{-4}$, sledi da treba da bude $h < 0.036$. Stoga formirajmo tabelu za funkciju f(x) sa korakom $h = 0.03$

x	f	Δf	$\Delta^2 f$
3.40	−0.03590		
		0.91254	
3.43	0.87664		0.02780
		0.94034	
3.46	1.81698		

na osnovu koje dobijamo da je Newtonov interpolacioni polinom

$$L_2(x_0 + q\,h) = -0.03590 + 0.91254q + \frac{1}{2}0.02780q(q-1),$$

gde je $q = (x-3.40)/0.03$. Njegova nula $q^* = 0.03992$, koja pripada intervalu $(0, 1)$, određuje rešenje jednačine sa traženom tačnošću

$$x^* = 3.40 + 0.03 \cdot 0.03992 = 3.40120 \qquad \longrightarrow \qquad x^* = 3.4012.$$

2.38 *Ako je* $y = f(x)$ *i* $f'(x) \neq 0$ *za* $x \in [x_0, x_1]$, *pokazati da je greška pri linearnoj inverznoj interpolaciji pomoću tačaka* (x_0, y_0) *i* (x_1, y_1) *data izrazom*

$$R_1(y) = -(y - y_0)\,(y - y_1)\,\frac{f''(\xi)}{2{f'}^3(\xi)}, \qquad x_0 < \xi < x_1,$$

ako funkcija $f''(x)/{f'}^3(x)$ *postoji i neprekidna je na intervalu* $[x_0, x_1]$. *Pokazati takođe da važe ocene*

$$|R_1| \le \frac{(y_1 - y_0)^2}{8}\,K, \qquad |R_1| \le \frac{h^2}{8}M_1^2K, \qquad |R_1| \le \frac{h^2}{8}\left(\frac{M_1}{m_1}\right)^2\frac{M_2}{m_1},$$

gde je $h = x_1 - x_0$, $|f''(x)/{f'}^3(x)| \le K$ *i* $m_1 \le |f'(x)| \le M_1$, $|f''(x)| \le M_2$ *za* $x \in [x_0, x_1]$.

Rešenje: Neka je $x = g(y)$, gde je $g(y)$ jednoznačno određena inverzna funkcija funkciji $f(x)$ na intervalu $[x_0, x_1]$. Dalje, neka je $L_1(y)$ interpolacioni polinom prvog stepena funkcije $g(y)$ određen čvorovima (y_0, x_0) i (y_1, x_1),

$$L_1(y) = \frac{y - y_1}{y_0 - y_1} x_0 + \frac{y - y_0}{y_1 - y_0} x_1$$

i $R_1(y)$ odgovarajuća greška interpolacije

$$(*) \qquad R_1(y) = \frac{g''(\eta)}{2} (y - y_0)(y - y_1), \qquad \eta = f(\xi) \in [y_0, y_1].$$

Diferenciranjem jednačine $x = g(y)$ po x dobijamo da je $1 = g'(y)\, y'(x)$, tj. $g'(y) = 1/f'(x)$. Ponovnim diferenciranjem poslednje relacije dobijamo da je $g''(y)\, y'(x) = -f''(x)/f'^2(x)$, odnosno $g''(y) = -f''(x)/f'^3(x)$. Zamenom vrednosti $g''(\eta) = -f''(\xi)/f'^3(\xi)$ u izrazu $(*)$ sledi prvo tvrđenje.

Koristeći dokazano tvrđenje i pretpostavke zadatka, dobijamo da je

$$|R_1(y)| \le \frac{K}{2} \left(\frac{y_1 - y_0}{2} \right)^2 = \frac{K}{8} (y_1 - y_0)^2.$$

Dalje, kako je $y_1 - y_0 = f(x_1) - f(x_0) = f'(\theta)(x_1 - x_0) = h\, f'(\theta)$, na osnovu poslednje ocene sledi da je

$$|R_1(y)| \le \frac{K}{8} h^2 M_1^2.$$

Konačno, s obzirom da je $|f''(x)/f'^3(x)| \le M_2/m_1^3$, sledi

$$|R_1(y)| \le \frac{1}{2} \frac{M_2}{m_1^3} \left(\frac{y_1 - y_0}{2} \right)^2 \le \frac{h^2}{8} M_1^2 \frac{M_2}{m_1^3},$$

čime je dokazana i poslednja ocena greške linearne inverzne interpolacije.

2.39 *Eliptički integral prve vrste je funkcija*

$$f(\phi, \alpha) = \int_0^{\phi} \frac{d\psi}{\sqrt{1 - \sin^2 \alpha \sin^2 \psi}}.$$

Izračunati

$$\int_0^{\arcsin \frac{12}{13}} \frac{d\psi}{\sqrt{1 - 0.78 \sin^2 \psi}},$$

ako je podintegralna funkcija zadata sledećom tabelom:

	$\alpha_0 = 62°$	$\alpha_1 = 63°$	$\alpha_2 = 64°$
$\phi_0 = 67°$	1.4220753	1.4302236	1.4384298
$\phi_1 = 68°$	1.4522494	1.4609653	1.4697532
$\phi_2 = 69°$	1.4828598	1.4921728	1.5015826

Rešenje: Pošto su čvorovi interpolacije ekvidistantni

$$x_i - x_{i-1} = h, \qquad y_j - y_{j-1} = k,$$

koristićemo interpolacioni polinom sa konačnim razlikama

$$\begin{aligned} L_n(x_0 + ph, y_0 + qk) = f(x_0, y_0) + p\Delta^{1+0} f(x_0, y_0) + q\Delta^{0+1} f(x_0, y_0) + \\ + \frac{1}{2!} p(p-1)\Delta^{2+0} f(x_0, y_0) + \frac{1}{2!} 2pq\Delta^{1+1} f(x_0, y_0) \\ + \frac{1}{2!} q(q-1)\Delta^{0+2} f(x_0, y_0) + \dots \end{aligned}$$

gde je $p = \frac{x-x_0}{h}$ i $q = \frac{y-y_0}{k}$, a konačne razlike se računaju prema sledećim formulama:

$$\begin{aligned} \Delta^{1+0} f(x_i, y_j) &= f(x_{i+1}, y_j) - f(x_i, y_j) \\ \Delta^{0+1} f(x_i, y_j) &= f(x_i, y_{j+1}) - f(x_i, y_j) \\ \Delta^{2+0} f(x_i, y_j) &= \Delta^{1+0} f(x_{i+1}, y_j) - \Delta^{1+0} f(x_i, y_j) \\ \Delta^{1+1} f(x_i, y_j) &= \Delta^{0+1} f(x_{i+1}, y_j) - \Delta^{0+1} f(x_i, y_j) \\ &= \Delta^{1+0} f(x_i, y_{j+1}) - \Delta^{1+0} f(x_i, y_j) \\ \Delta^{0+2} f(x_i, y_j) &= \Delta^{0+1} f(x_i, y_{j+1}) - \Delta^{0+1} f(x_i, y_j) \end{aligned}$$

Prvo određujemo vrednosti argumenata α i ϕ za koje se računa vrednost integrala:

$$\begin{aligned} \sin^2 \alpha = 0.78 \quad &\Rightarrow \quad \alpha = 62.0279^\circ \\ \phi = \arcsin \tfrac{12}{13} \quad &\Rightarrow \quad \phi = 67.3801^\circ \end{aligned}$$

U datoj tabeli su koraci $h = 1$ i $k = 1$, pa je $p = \frac{\alpha-\alpha_0}{h} = 0.0279$ i $q = \frac{\phi-\phi_0}{k} = 0.3801$. Konačne razlike koje je moguće izračunati na osnovu zadatih podataka su

$$\begin{aligned} &\Delta^{1+0} f(\alpha_0, \phi_0) = f(\alpha_1, \phi_0) - f(\alpha_0, \phi_0) = 0.0081483 \\ &\Delta^{0+1} f(\alpha_0, \phi_0) = f(\alpha_0, \phi_1) - f(\alpha_0, \phi_0) = 0.0201741 \\ &\Delta^{2+0} f(\alpha_0, \phi_0) = f(\alpha_2, \phi_0) - 2f(\alpha_1, \phi_0) + f(\alpha_0, \phi_0) = 0.0000579 \\ &\Delta^{1+1} f(\alpha_0, \phi_0) = (f(\alpha_1, \phi_1) - f(\alpha_1, \phi_0)) - (f(\alpha_0, \phi_1) - f(\alpha_0, \phi_0)) = 0.0005676 \\ &\Delta^{2+0} f(\alpha_0, \phi_0) = f(\alpha_0, \phi_2) - 2f(\alpha_0, \phi_1) + f(\alpha_0, \phi_0) = 0.0004354 \end{aligned}$$

te je približno tražena vrednost funkcije

$$\begin{aligned} f(62.0279^\circ, 67.3801^\circ) &\approx L_2(62.0279^\circ, 67.3801^\circ) \\ &= 1.4220753 + 0.0279 * 0.0081483 + 0.3801 * 0.0201741 + \\ &+ \frac{1}{2} 0.0279\,(0.0279 - 1)\, 0.0000579 + 0.0279 * 0.3801 * 0.0005676 + \\ &+ \frac{1}{2} 0.3801\,(0.3801 - 1)\, 0.0004354 = 1.4337269. \end{aligned}$$

MATLAB

Višedimenziona interpolacija:	`Z = interp2 (x, y, z, X, Y)` `Z = griddata (x, y, z, X, Y)`

MATLAB poseduje dva osnovna vida dvodimenzione interpolacije. Osnovna razlika između njih je postojanje (odnosno nepostojanje) pretpostavke o uređenosti skupa tačaka koji predstavlja "tabelu interpolacije".

U prvom slučaju pretpostavlja se da je poznat skup tačaka koje predstavljaju vrednosti funkcije $z = f(x, y)$ na diskretnoj, pravouganoj, mreži određenoj podelom itervala $[a_x, b_x] \times [a_y, b_y]$. U tom slučaju, za interpolaciju se koristi funkcija `Z = interp2(x, y, z, X, Y)`. Matrice `x`, `y` i `z` definišu pomenuti skup tačaka, dok su matricama `X` i `Y` određene tačke ravni u kojima se funkcija $f(x, y)$ približno računa interpolacijom. Matrice `X` i `Y` najčešće definišu finiju pravougaonu podelu intervala $[a_x, b_x] \times [a_y, b_y]$ nego matrice `x` i `y`. Funkcija `interp2` vraća matricu `Z`, koja sadrži interpolisane vrednosti polazne funkcije na finijoj mreži.

```
% Ugradjena funkcija peaks tabelirana na 10x10 mrezi intervala [-3, 3]x[-3, 3]
[x,y,z] = peaks(10);

% Gradimo finiju 25x25 mrezu intervala [-3, 3]x[-3, 3]
[X, Y] = meshgrid(linspace(-3, 3, 25), linspace(-3, 3, 25));

%Vrsimo interpolaciju
Z = interp2(x, y, z, X, Y, 'cubic');

%Iscrtavamo polaznu i profinjenu funkciju jednu iznad druge
mesh(x, y, z-5), hold on, mesh(X, Y, Z+5);
```

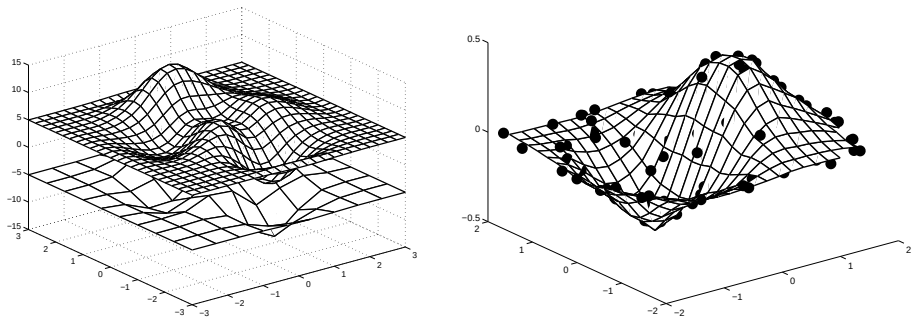
Rezultat rada skripta je prikazan na slici 2.3 (a).
Slično kao funkcija `interp1` i funkcija `interp2` vrši interpolaciju funkcijama koje su deo-po-deo polinomi. Dodatni parametar funkcije može biti `'nearest'`, `'linear'` i `'cubic'`, odnosno `'spline'` i ovim se dodatno kontroliše način interpolacije.

Ukoliko tačke koje koje predstavljaju vrednosti funkcije $z = f(x, y)$ nisu organizovane u pravougaonu mrežu, za interpolaciju je potrebno koristiti funkciju `Z = griddata(x, y, z, X, Y)`. U ovom slučaju se pretpostavlja da su `x`, `y` i `z` tri vektora iste dužine koji predstavljaju čvorove interpolacije, tj. tačke u kojima su poznate vrednosti funkcije $f(x, y)$. `X` i `Y` su matrice koje predstavljaju pravougaonu mrežu intervala na kojem se vrši interpolacija, odnosno tačke u kojima se vrednost funkcije $f(x, y)$ računa interpolacijom. Funkcija `griddata` vraća matricu `Z` koja predstavlja interpolisane vrednosti funkcije u ovim tačkama. I funkcija `griddata` može imati dodatni parametar koji kontroliše vrstu interpolacije, slično kao i funkcija `interp2`. Na primer,

```
% Gradimo sto slucajnih tacaka iz intervala [-2, 2]x[-2, 2]
x = rand(100,1)*4-2;
y = rand(100,1)*4-2;

% Izracunavamo vrednosti funkcije u tackama
z = x.*exp(-x.^2-y.^2);
```

```
% Gradimo (finiju) pravouganu podelu intervala [-2, 2]x[-2, 2]
[X, Y] = meshgrid(linspace(-2, 2, 20), linspace(-2, 2, 20));

% Vrsimo interpolaciju
Z = griddata(x, y, z, X, Y, 'cubic');

% Iscrtavamo polazne tacke i interpoliranu povrs
hold on, mesh(X, Y, Z), plot3(x,y,z,'o');
```

Rezultat rada skripta je prikazan na slici 2.3 (b).



Slika 2.3: Levo: primena funkcije `interp2`, Desno: primena funkcije `griddata`

2.5 Hermite-ov polinom

Polinom $P_n(x)$ stepena n, $n = \sum_{i=0}^{m} n_i - 1$, koji zadovoljava uslove interpolacije

$$P_n^{(k)}(x_i) = f^{(k)}(x_i), \qquad k = 0, \ldots, n_i - 1, \quad i = 0, \ldots, m,$$

naziva se Hermite-ov interpolacioni polinom. Može da se predstavi u sledećem obliku

$$\begin{aligned} P_n(x) = {} & f(x_0) + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_0](x - x_0)^2 + \cdots \\ & + f[\underbrace{x_0, \ldots, x_0}_{n_0 \text{ puta}}](x - x_0)^{n_0 - 1} + f[\underbrace{x_0, \ldots, x_0}_{n_0 \text{ puta}}, x_1](x - x_0)^{n_0} \\ & + f[\underbrace{x_0, \ldots, x_0}_{n_0 \text{ puta}}, x_1, x_1](x - x_0)^{n_0}(x - x_1) + \cdots \\ & + f[\underbrace{x_0, \ldots, x_0}_{n_0 \text{ puta}}, \ldots, \underbrace{x_m, \ldots, x_m}_{n_m \text{ puta}}](x - x_0)^{n_0} \ldots (x - x_m)^{n_m - 1} \end{aligned}$$

gde je $f[\underbrace{x, \ldots, x}_{(k+1) \text{ puta}}] = \frac{1}{k!} f^{(k)}(x)$.

2.40 *Odrediti Hermite-ov interpolacioni polinom funkcije $f(x)$ zadate tabelom*

x	$f(x)$	$f'(x)$	$f''(x)$
-1	4	-11	$*$
0	1	1	6
1	6	13	$*$

Rešenje: Zadatak ćemo, za razliku od standardnog postupka koji će biti primenjen u narednim zadacima, rešiti korišćenjem Lagrange-ovog interpolacionog polinoma. S obzirom da je funkcija zadata u tri čvora, Hermite-ov interpolacioni polinom šestog stepena definisan zadatom tabelom može se predstaviti u sledećem obliku

$$H_6(x) = L_2(x) + \omega_3(x)\, H_3(x),$$

gde je $L_2(x) = 4x^2 + x + 1$ Lagrange-ov interpolacioni polinom drugog stepena funkcije $f(x)$ određen zadatom tabelom, $\omega_3(x) = x(x^2 - 1)$ polinom trećeg stepena definisan čvorovima interpolacije, a $H_3(x)$ je za sada nepoznati polinom trećeg stepena. Ovako definisani polinom $H_6(x)$ očigledno zadovoljava uslove interpolacije $H_6(x_k) = f(x_k)$, $x_k = -1, 0, 1$. Da bi i preostali uslovi bili zadovoljeni, potrebno je da je

$$H_6'(x_k) = f'(x_k), \quad x_k = -1, 0, 1, \qquad\qquad H_6''(0) = f''(0),$$

što daje uslove interpolacije za polinom $H_3(x)$. Ti uslovi su dati tabelom

x	$H_3(x)$	$H_3'(x)$
-1	-2	$*$
0	0	1
1	2	$*$

Polinom $H_3(x)$ ćemo takođe predstaviti u obliku

$$H_3(x) = \overline{L}_2(x) + \omega_3(x)\, H_0,$$

gde je sada $\overline{L}_2(x) = x(x-1) + x(x+1) = 2x$, a $H_0 = \text{const}$ koju određujemo iz uslova $H_3'(0) = 1$. Tako dobijamo da je $H_0 = 1$, pa je $H_3(x) = x^3 + x$, i konačno

$$H_6(x) = x^6 + 3x^2 + x + 1.$$

2.41 *Odrediti Hermite-ov interpolacioni polinom za funkciju zadatu sledećom tabelom:*

x	$f(x)$	$f'(x)$	$f''(x)$
0	1	0	2
1	-1	0	$*$
2	0	$*$	$*$

Rešenje: Stepen polinoma je $n = (3 + 2 + 1) - 1 = 5$. Formiramo tabelu podeljenih razlika na sledeći način:

x_i	f	$f[1]$	$f[2]$	$f[3]$	$f[4]$	$f[5]$
0	1					
		$\frac{f'(0)}{1!} = 0$				
0	1		$\frac{f''(0)}{2!} = 1$			
		$\frac{f'(0)}{1!} = 0$		$\frac{-2-1}{1-0} = -3$		
0	1		$\frac{-2-0}{1-0} = -2$		$\frac{4-(-3)}{1-0} = 7$	
		$\frac{-1-1}{1} = -2$		$\frac{2-(-2)}{1-0} = 4$		$\frac{-\frac{9}{4}-7}{2-0} = -\frac{37}{8}$
1	-1		$\frac{0-(-2)}{1-0} = 2$		$\frac{-\frac{1}{2}-4}{2-0} = -\frac{9}{4}$	
		$\frac{f'(1)}{1!} = 0$		$\frac{1-2}{2-0} = -\frac{1}{2}$		
1	-1		$\frac{1-0}{2-1} = 1$			
		$\frac{0-(-1)}{2-1} = 1$				
2	0					

Prvi brojevi u svakoj koloni tabele podeljenih razlika predstavljaju koeficijente Hermite-ovog interpolacionog polinoma, koji glasi:

$$\begin{aligned} H_5(x) &= 1 + x^2 - 3x^3 + 7x^3(x-1) - \tfrac{37}{8}x^3(x-1)^2 \\ &= 1 + x^2 - 14.625x^3 + 16.25x^4 - 4.625x^5 \end{aligned}$$

2.42 *Napisati Hermite-ov interpolacioni polinom za funkciju datu tabelom*

x	$f(x)$	$f'(x)$
-1	-1	0
0	0	0
1	1	0

Rešenje: $H_5(x) = 0.5x^3\,(5 - 3x^2)$.

2.43 *Napisati Hermite-ov interpolacioni polinom za funkciju datu tabelom*

x	$f(x)$	$f'(x)$	$f''(x)$
-1	5	-3	-32
0	1	0	$*$
1	7	17	52

Rešenje: Zadatom tabelom je određen Hermite-ov interpolacioni polinom sedmog stepena. Tabela podeljenih razlika je formirana tako da je pri računanju podeljenih razlika nad $(n+1)$-ostrukim čvorom uzeta u obzir veza $f[\underbrace{x, ..., x}_{n}] = f^{(n)}(x)/n!$.

Prvi brojevi u svakoj koloni tabele podeljenih razlika koja sledi, predstavljaju koeficijente Hermite-ovog interpolacionog polinoma $H_7(x)$,

x	f	$f[1]$	$f[2]$	$f[3]$	$f[4]$	$f[5]$	$f[6]$	$f[7]$
-1	5							
		-3						
-1	5		-16					
		-3		15				
-1	5		-1		-10			
		-4		5		4		
0	1		4		-2		-1	
		0		1		2		1
0	1		6		2		1	
		6		5		4		
1	7		11		10			
		17		15				
1	7		26					
		17						
1	7							

Traženi polinom je

$$H_7(x) = 5 - 3(x+1) - 16(x+1)^2 + 15(x+1)^3 - 10x(x+1)^3 + 4x^2(x+1)^3 - x^2(x+1)^3(x-1) + x^2(x+1)^3(x-1)^2 = x^7 + 5x^2 + 1.$$

C Prilikom implementacije procedure koja računa Hermite-ov interpolacioni polinom važno je odlučiti se za strukturu podataka kojom će biti predstavljena tabela interpolacije. Ako bi se u kodu dosledno preslikao izgled tabele koji se dobija kada se zadatak rešava ručno, onda bi se tabela mogla predstaviti dvodimenzionim poljem koje bi imalo dosta praznih elemenata. Umesto toga, može se uočiti da se svaka kolona tabele podeljenih razlika računa samo na osnovu sadržaja prethodne kolone, tako da je za čuvanje tabele podeljenih razlika dovoljno odvojiti jedno polje dužine jednake dimenziji problema. Štaviše, ukoliko se vrednosti nove kolone računaju počev od poslednje i pritom smeštaju u polje počev od poslednjeg elementa, postiže se da na početku polja bivaju sačuvane vrednosti prvih elemenata svake kolone, a to su upravo vrednosti koje su potrebne za izračunavanje koeficijenata interpolacionog polinoma. Ovo polje je predstavljeno promenljivom `array`. S obzirom da pojedinim čvorovima odgovara više redova tabele interpolacije, biće nam potreban i jedan pomoćni vektor čiji svaki element odgovara jednom redu tabele interpolacije, a vrednost elementa predstavlja indeks odgovarajućeg čvora interpolacije u vektoru koji sadrži vrednosti tih čvorova. Ovaj vektor je predstavljen promenljivom `index`. Sa ovako uvedenim lokalnim promenljivim može se napisati kompaktan kod koji popunjava tabelu podeljenih razlika tako što za svaki njen element ili izračuna odgovarajuću podeljenu razliku ili primeni formulu koja daje vezu podeljenih razlika sa izvodom funkcije. Kada se na kraju na gore opisani

način u promenljivoj `array` formira polje prvih elemenata kolona tabele podeljenih razlika, preostaje da se izračuanju koeficijenti interpolacionog polinoma. Formula za izračunavanje ovih koeficijenata se transformiše kako je opisano u odgovarajućem komentaru niže u kodu da bi računanje bilo efikasnije. Sledi procedura koja implementira metodu (treba pomenuti i da ista procedura može da se koristi i za Lagrange-ovu interpolaciju, u kom slučaju se prosto na ulazu zadaje $n_i = 1$ za svako i):

```
#include <assert.h>
#include <stdlib.h>
#include <string.h>

/*
 * Funkcija hermite() izracunava Hermite-ov interpolacioni polinom za
 * funkciju zadatu tablicom. Tablica mora da zadovoljava uslove da je
 * broj redova tabele interpolacije veci ili jednak 1, da su cvorovi
 * interpolacije rastuci, da je broj vrednosti zadatih za svaki cvor
 * veci ili jednak 1, te da je ukupan broj zadatih vrednosti veci ili
 * jednak 2.  Argumenti funkcije su:
 *   m - broj redova tabele interpolacije
 *   x - polje sa  cvorovima interpolacije
 *   n - polje kojim je za svaki cvor interpolacije zadat broj poznatih
 *       prvih izvoda funkcije u datom cvoru
 *   f - tabela sa vrednostima izvoda funkcije
 *   c - polje u koje ce biti smesteni izracunati koeficijenti
 *       interpolacionog polinoma
 * Pretpostavlja se da su sva polja alocirana izvan funkcije hermite().
 */
void
hermite(int m, double *x, int *n, double **f, double *c)
{
        int             N;      /* Broj koeficijenata interpolacionog
                                 * polinoma (tj. stepen polinoma je N-1. */
        int            *index;  /* Polje indeksa cvorova interpolacije
                                 * koji odgovaraju pojedinim redovima
                                 * tabele podeljenih razlika. */
        double         *array;  /* Tekuca kolona tabele podeljenih
                                 * razlika, istovremeno i skup prvih
                                 * elemenata u svakoj koloni ove tabele. */
        int             den;    /* Pomocna promenljiva za racunanje
                                 * faktorijela. */
        int             i,
                        j,
                        k;      /* Indeksi u petljama. */

        /* Izracunava se broj koeficijenata interpolacionog polinoma. */
        N = 0;
        for (i = 0; i < m; i++)
                N += n[i];

        /* Proverava se da li su ulazni parametri validni. */
        assert(m >= 1);
        for (i = 0; i < m - 1; i++)
                assert(x[i] < x[i + 1]);
        for (i = 0; i < m; i++)
                assert(n[i] >= 1);
```

```
        assert(N >= 2);

        /* Alocira se i popunjava polje indeksa cvorova koji odgovaraju
         * pojedinim redovima tabele podeljenih razlika. */
        index = (int *) malloc(N * sizeof(int));
        for (k = 0, i = 0; i < m; i++)
                for (j = 0; j < n[i]; j++, k++)
                        index[k] = i;

        /* Alocira se memorija za tekucu kolonu tabele podeljenih razlika
         * i inicijalizuje se ta kolona na vrednosti funkcije zadate u
         * odgovarajucim cvorovima interpolacije. */
        array = (double *) malloc(N * sizeof(double));
        for (i = 0; i < N; i++)
                array[i] = f[index[i]][0];

        /* Izracunava se kolonu po kolonu tabela podeljenih razlika, pri
         * cemu se elementi kolone smestaju prema kraju odgovarajuceg
         * polja, tako da se u prvom delu polja formira niz prvih
         * elemenata u kolonama tabele podeljenih razlika, koji su upravo
         * potrebni za izracunavanje koeficijenata interpolacionog
         * polinoma. U svakom koraku promenljiva den jednaka je
         * faktorijelu broja kolone. Ukoliko su krajnji cvorovi
         * interpolacije u datom koraku jednaki, za izracunavanje se
         * koristi vrednost izvoda iz tablice izvoda; u suprotnom se
         * koristi uobicajena rekurzivna formula za racunanje podeljene
         * razlike. */
        for (j = 1, den = 1; j < N; j++, den *= j)
                for (i = N - 1; i >= j; i--)
                        if (index[i] == index[i - j])
                                array[i] = f[index[i]][j] / den;
                        else
                                array[i] =
                                        (array[i] - array[i - 1]) /
                                        (x[index[i]] - x[index[i - j]]);

        /* Izracunavaju se koeficijenti interpolacionog polinoma.
         * Interpolacioni polinom je odredjen izrazom:
         * array[0]+array[1]*(x-x[0])+...+
           array[N-1]*(x-x[0])^n[0]*(x-x[1])^n[1]*...*(x-x[m-2])^(n[m-2])*(x-x[m-1])^(n[m-1]-1)
         * Ovaj izraz se izracunava tako sto se gornja formula transformise na sledeci nacin:
         * array[0]+(x-x[0])*(array[1]+...*(array[N-1]+(x-x[index[N-2]])*array[N-1])) */
        memset(c, 0, N * sizeof(double));
        c[0] = array[N - 1];
        for (j = N - 1; j > 0; j--) {
                for (k = N - j; k > 0; k--)
                        c[k] = c[k - 1] - c[k] * x[index[j - 1]];
                c[0] = array[j - 1] - c[0] * x[index[j - 1]];
        }

        /* Oslobadja se memorija zauzeta pomocnim promenljivima. */
        free(index);
        free(array);
}
```

2.44 *Odrediti Hermite-ov interpolacioni polinom koji u čvoru x_0 ima vrednosti izvoda do reda n jednake odgovarajućim vrednostima izvoda funkcije $f(x)$, tj.*

$$P^{(k)}(x_0) = f^{(k)}(x_0) \qquad k = 0, 1, \ldots, n.$$

Rešenje: U ovom slučaju je broj čvorova jedan ($m = 0$), a broj uslova u tom čvoru $n_0 = n + 1$. Izraz za Hermite-ov interpolacioni polinom se može izvesti iz izraza za Newton-ov interpolacioni polinom sa podeljenim razlikama,

$$\begin{aligned} P_n(x) = f[x_0] + f[x_0, x_0 + \varepsilon](x - x_0) + \ldots \\ + f[x_0, \ldots, x_0 + n\varepsilon](x - x_0) \ldots (x - (x_0 + (n-1)\varepsilon)). \end{aligned}$$

kada se pusti da $\varepsilon \to 0$,

$$P_n(x) = f[x_0] + f[x_0, x_0](x - x_0) + \ldots + f[\underbrace{x_0, \ldots, x_0}_{n+1 \ puta}](x - x_0)^n$$

Na osnovu veze podeljenih razlika i izvoda, sledi da je:

$$P_n(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Dakle, očigledno je da je Hermite-ov interpolacioni polinom za date uslove identičan Taylor-ovom polinomu funkcije $f(x)$ u tački x_0.

2.45 *Pokazati da se greška Hermite-ovog interpolacionog polinoma*

$$\begin{aligned} H_3(x) = f(x_0) + (x - x_0)\, f[x_0, x_0] + (x - x_0)^2 f[x_0, x_0, x_1] \\ + (x - x_0)^2 (x - x_1)\, f[x_0, x_0, x_1, x_1] \end{aligned}$$

konstruisanog za funkciju $f(x) \in \mathcal{C}^4[x_0, x_1]$ može oceniti izrazom

$$|R_3(x)| \leq \frac{(x_1 - x_0)^4}{384} \max_{[x_0, x_1]} |f^{(4)}(\xi)|.$$

Rešenje: Greška interpolacije se može izraziti u Lagrange-ovom obliku

$$R_3(x) = f(x) - H_3(x) = \frac{1}{4!} f^{(4)}(\xi)\omega_4(x),$$

gde je $\xi \in [x_0, x_1]$ i $\omega_4(x) = (x - x_0)^2(x - x_1)^2$. Kako je

$$\max_{[x_0, x_1]} |\omega_4(x)| = |\omega_4\big((x_0 + x_1)/2\big)| = \left(\frac{x_1 - x_0}{2}\right)^4,$$

tvrđenje neposredno sledi.

2.46 *Odrediti polinom najnižeg stepena $H_m(x)$ takav da je*

$$H_m(x_k) = f(x_k), \qquad H'_m(x_k) = f'(x_k), \qquad za \quad x_0 < x_1 < \cdots < x_n.$$

Rešenje: Traženi polinom je Hermite-ov interpolacioni polinom koji možemo napisati pomoću podeljenih razlika, a zatim podeljene razlike izraziti pomoću zadatih vrednosti funkcije i njenog prvog izvoda u čvorovima (na način kako je to rađeno u prethodnim zadacima).

Ovde ćemo eksplicitno napisati zapis ovog polinoma pomoću datih vrednosti. Napišimo ga u obliku

$$H_m(x) = \sum_{i=0}^{n} g_i(x) f_i + \sum_{i=0}^{n} h_i(x) f'_i,$$

gde je $f_i = f(x_i)$, $f'_i = f'(x_i)$, $i = 0, \ldots, n,$. $2(n+1)$ uslova interpolacije jednoznačno određuju polinom $m = 2n+1$-og stepena. Stoga su $g_i(x)$ i $h_i(x)$ za svako i polinomi stepena $2n+1$ koje ćemo odrediti iz uslova interpolacije

$$(*) \qquad \begin{array}{lll} H_m(x_j) = f_j & \to & g_i(x_j) = \delta_{ij}, \quad h_i(x_j) = 0, \\ H'_m(x_j) = f'_j & \to & g'_i(x_j) = 0, \quad h'_i(x_j) = \delta_{ij}, \end{array} \qquad i, j = 0, \ldots, n,$$

gde je $\delta = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$ Kroneckerov δ-simbol. Ako definišemo sledeće polinome stepena n

$$l_i(x) = \frac{(x-x_0)\ldots(x-x_{i-1})\,(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})\,(x_i-x_{i+1})\ldots(x_i-x_n)}, \qquad i = 0, \ldots, n,$$

s obzirom na uslove interpolacije $(*)$ i činjenicu da je $l_i(x_j) = \delta_{ij}$ nepoznati polinomi će biti oblika

$$g_i(x) = r_i(x)\, l_i^2(x), \qquad h_i(x) = s_i(x) l_i^2(x).$$

Nepoznati činioci $r_i(x)$ i $s_i(x)$ su polinomi prvog stepena (da bi g_i i h_i bili polinomi $2n+1$ stepena) i određuju se iz preostala četiri uslova interpolacije

$$g_i(x_i) = 1, \qquad g'_i(x_i) = 0, \qquad h_i(x_i) = 0, \qquad h'_i(x_i) = 1.$$

Poslednji uslovi se svode na uslove

$$r_i(x_i) = 1, \qquad r'_i(x_i) + 2l'_i(x_i) = 0, \qquad s_i(x_i) = 0, \qquad s'_i(x_i) = 1,$$

koji određuju linearne činioce u obliku

$$r_i(x) = 1 - 2(x - x_i) l'_i(x_i), \qquad s_i(x) = x - x_i.$$

Zamenom u početni izraz dobijamo da je traženi interpolacioni polinom

$$H_{2n+1}(x) = \sum_{i=0}^{n} \Big(f_i + (x - x_i)\big(f'_i - 2l'_i(x_i) f_i\big) \Big) l_i^2(x).$$

2.47 *Izvesti formulu za Hermite-ov interpolacioni polinom i grešku*

$$f(x) = H_3(x) + R_3(x),$$

$$H_3(x_0+qh) = (1+2q)(1-q)^2 f_0 + (3-2q)q^2 f_1 + q(1-q)^2 h f_0' + q^2(1-q)h f_1'$$

$$R_3(x_0+qh) = \frac{h^4}{4!} f^{(4)}(\xi) q^2(1-q)^2, \qquad x_0 < \xi < x_0 + h.$$

gde je $x = x_0 + qh$, $0 \le q \le 1$.

Rešenje: Zadajući vrednosti funkcije i njenog izvoda u dva čvora x_0 i $x_0 + h$, prema prethodnom zadatku imamo da je

$$l_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{x_1 - x}{h}, \qquad l_0'(x) = -\frac{1}{h},$$

$$l_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - x_0}{h}, \qquad l_1'(x) = \frac{1}{h}.$$

Uvedimo smenu $x - x_0 = qh$, $x_1 - x = (1-q)h$, pa izraz za $H_3(x_0+qh)$ neposredno sledi. Izraz za grešku sledi iz Lagrange-ovog oblika greške interpolacije u kome je $\omega_4(x) = (x-x_0)^2(x-x_1)^2$, tj. $\omega_4(x_0+qh) = h^4 q^2(1-q)^2$.

2.6 Interpolacioni splajn

Definisan je podelom Δ intervala interpolacije $[a, b]$

$$\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}, \qquad h_i = x_i - x_{i-1}.$$

Analitički izraz za kubni splajn na intervalu $[x_i, x_{i+1}]$ je

$$S_\Delta(f; x) = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3,$$

gde su koeficijenti

$$\alpha_i = f(x_i) = f_i, \qquad \beta_i = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{h_{i+1}}{6}(2M_i + M_{i+1}),$$

$$\gamma = \frac{1}{2}M_i, \qquad \delta_i = \frac{1}{6h_{i+1}}(M_{i+1} - M_i)$$

Momenti $M_i = S_\Delta''(f; x_i)$ su rešenja trodijagonalnog sistema linearnih jednačina

$$\mu_i M_{i-1} + 2M_i + \nu_i M_{i+1} = \lambda_i, \qquad i = 1, \ldots, n-1,$$

dopunjenog zadatim graničnim uslovima. Ovde su korišćene oznake:

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \qquad \nu_i = 1 - \mu_i, \qquad \lambda_i = 6f[x_{i-1}, x_i, x_{i+1}].$$

2.48 *Konstruisati kubni splajn za funkciju zadatu sledećom tabelom:*

x_i	0	0.2	0.5
$f(x_i)$	1.3225	1.5479	1.7783

pri čemu je $f''(0) = 0.4723$ *i* $f''(0.5) = 1.2342$.

Rešenje: Interpolacioni splajn je određen podelom $\Delta = \{0, 0.2, 0.5\}$, tako da postoji samo jedan unutrašnji čvor u kome jednačina za momente glasi

$$\mu_1 M_0 + 2M_1 + \nu_1 M_2 = \lambda_1,$$

gde je

$$\mu_1 = \frac{0.2}{0.2 + 0.3} = 0.4 \qquad \nu_1 = 1 - 0.4 = 0.6$$

$$\lambda_1 = 6\frac{1}{0.5}\left(\frac{1.7783 - 1.5479}{0.3} - \frac{1.5479 - 1.3225}{0.2}\right) = -4.308.$$

Pošto su dodatni uslovi $M_0 = 0.4723$ i $M_2 = 1.2342$, sledi da je $M_1 = -2.6187$. Na osnovu toga su koeficijenti splajna

$x \in$	α_i	β_i	γ_i	δ_i
$[0, 0.2]$	1.3225	1.1828	0.2362	-2.5758
$[0.2, 0.5]$	1.5479	0.9682	-1.3094	2.1405

odnosno, jednačina splajna je

$$S_\Delta(f; x) = \begin{cases} 1.3225 + 1.1828x + 0.2362x^2 - 2.5758x^3 & x \in [0, 0.2] \\ 1.5479 + 0.9682(x - 0.2) - 1.3094(x - 0.2)^2 & \\ \qquad + 2.1405(x - 0.2)^3 & x \in [0.2, 0.5] \end{cases}$$

2.49 *Odrediti na intervalu* $[0, 2]$ *kubni splajn* $S(x)$ *zadat tabelom*

x	0	1	2
$S(x)$	0	0.3679	0.1353

i uslovima na granici $S'(0) = 0.5$ *i* $S'(2) = -0.2030$.

Rešenje: Koeficijenti kubnog splajna dati u reprezentaciji splajna u prethodnom zadatku se mogu odrediti i pomoću momenata $M_k = S''(x_k)$. Momenti su rešenja sistema linearnih jednačina

$$\begin{aligned} \frac{h_1}{3}M_0 + \frac{h_1}{6}M_1 &= \frac{S(x_1) - S(x_0)}{h_1} - S'(x_0) \\ \mu_1 M_0 + 2M_1 + \nu_1 M_2 &= \lambda_1 \\ \frac{h_2}{6}M_1 + \frac{h_2}{3}M_2 &= S'(x_2) - \frac{S(x_2) - S(x_1)}{h_2} \end{aligned}$$

gde je

$$h_1 = h_2 = 1, \qquad \mu_1 = \nu_1 = 0.5, \qquad \lambda_1 = 6S[x_0, x_1, x_2] = -1.8015.$$

Rešenja ovog sistema su

$$M_0 = 0.1530, \qquad M_1 = -1.0985, \qquad M_2 = 0.6381,$$

pa je traženi splajn

$$S(x) = \begin{cases} 0.5x + 0.0765x^2 - 0.2086x^3, & x \in [0,1] \\ 0.3679 + 0.0272(x-1) - 0.5493(x-1)^2 + 0.2894(x-1)^3, & x \in [1,2] \end{cases}$$

2.50 *Konstruisati kubni splajn za funkciju $f(x)$ koja zadovoljava uslove*

$$f(0) = 1, \qquad f(1) = 2, \qquad f(2) = 3, \qquad f'(0) + f(0) = 1, \qquad f'(2) + f(2) = 2.$$

Rešenje: Tačkama $x_0 = 0$, $x_1 = 1$ i $x_2 = 2$ interval $[0, 2]$ je podeljen na dva podintervala dužine $h = 1$. Kada $x \in [x_k, x_{k+1}]$ splajn je oblika

$$S(f; x) = \alpha_k + \beta_k(x - x_k) + \gamma_k(x - x_k)^2 + \delta_k(x - x_k)^3, \qquad k = 0, 1.$$

Korišćenjem pet zadatih uslova, pri čemu se uslov $f(1) = 2$ koristi dva puta (za oba intervala), kao i dva uslova neprekidnosti prvog i drugog izvoda splajna u unutrašnjem čvoru x_1, dobijamo sistem od osam linearnih jednačina po osam nepoznatih koeficijenata α_k, β_k, γ_k, δ_k, $k = 0, 1$ čije rešenje je

$$\begin{aligned} &\alpha_0 = 1, \qquad \beta_0 = 0, \qquad \gamma_0 = 5/4, \qquad \delta_0 = -1/4, \\ &\alpha_1 = 2, \qquad \beta_1 = 7/4, \qquad \gamma_1 = 1/2, \qquad \delta_1 = -5/4. \end{aligned}$$

Kada uvrstimo dobijene vrednosti koeficijenata i numeričke vrednosti za x_k, $k = 0, 1$, u izraz za splajn dobijamo da je

$$S(f; x) = \begin{cases} 1 + 5x^2/4 - x^3/4, & x \in [0,1] \\ 2 - 3x + 17x^2/4 - 5x^3/4, & x \in [1,2] \end{cases}$$

2.51 *Izračunati $f(1.16)$ pomoću prirodnog kubnog interpolacionog splajna funkcije date tabelom*

x	1.1275	1.1503	1.1735	1.1972
$f(x)$	0.11971	0.13957	0.15931	0.17902

Rešenje: Kako je splajn prirodni, momenti u krajnjim tačkama intervala su jednaki nuli, $M_0 = M_3 = 0$. Momenti u unutrašnjim čvorovima se određuju kao rešenja sistema linearnih jednačina

$$\mu_k M_{k-1} + 2M_k + \nu_k M_{k+1} = \lambda_k, \qquad k = 1, 2,$$

pri čemu je

$$h_1 = 0.0228, \qquad h_2 = 0.0232, \qquad h_3 = 0.0237,$$
$$\mu_1 = 0.49565, \qquad \nu_1 = 0.50435, \qquad \mu_2 = 0.49467, \qquad \nu_2 = 0.50533,$$
$$\lambda_1 = 6f[x_0, x_1, x_2] = -2.633610, \qquad \lambda_2 = 6f[x_1, x_2, x_3] = -2.458392.$$

Momenti su

$$M_0 = 0, \qquad M_1 = -1.073808, \qquad M_2 = -0.963606, \qquad M_3 = 0,$$

pa je reprezentacija splajna u intervalu $[1.1503, 1.1735]$ kome pripada tačka $x = 1.16$

$$S(f;x) = 0.13957 + 0.86289(x-1.1503) - 0.53690(x-1.1503)^2 + 0.79168(x-1.1503)^3,$$

i tražena vrednost

$$f(1.16) \approx S(f; 1.16) = 0.14789.$$

2.52 *Periodična funkcija $f(x)$ sa periodom jednakim 3 data je sledećom tabelom:*

x_i	0	1	2	3
$f(x_i)$	1	3	2	1

Odrediti odgovarajući interpolacioni kubni splajn.

Rešenje: U ovom slučaju za svako i važi $h_i = 1$ i $\mu_i = \nu_i = 0.5$, dok je

$$\lambda_i = 6\,\frac{1}{2}\big((f(x_{i+1}) - 2f(x_i) + f(x_{i-1})\big) = \begin{cases} 3(2-6+1) = -9 & i = 1 \\ 2(1-4+3) = 0 & i = 2 \\ 3(3-2+2) = 9 & i = 3 \end{cases}$$

S obzirom da je zbog periodičnosti funkcije i splajna $M_3 = M_0$, sistem jednačina za momente je

$$\begin{cases} 0.5M_0 + 2M_1 + 0.5M_2 &= -9, \\ 0.5M_1 + 2M_2 + 0.5M_0 &= 0, \\ 0.5M_2 + 2M_0 + 0.5M_1 &= 9, \end{cases}$$

odakle je $M_0 = 6$, $M_1 = -6$ i $M_2 = 0$. Sada se mogu izračunati koeficijenti splajna

x	α_i	β_i	γ_i	δ_i
$[0,1]$	1	1	3	-2
$[1,2]$	3	1	-3	1
$[2,3]$	2	-2	0	1

pa jednačina splajna glasi

$$S_\Delta(f;x) = \begin{cases} 1 + x + 3x^2 - 2x^3 & x \in [0,1] \\ 3 + (x-1) - 3(x-1)^2 + (x-1)^3 & x \in [1,2] \\ 2 - 2(x-2) + (x-2)^3 & x \in [2,3] \end{cases}$$

C Računanje koeficijenata splajn interpolacije se razlikuje po načinu na koji su data dva dodatna uslova, ali se bazično svodi na postavljanje i rešavanje trodijagonalnog sistema jednačina po momentima. Implementiraćemo proceduru koja računa ove koeficijente za slučaj kada su vrednosti momenata u krajnjim tačkama eksplicitno date (to je dakle generalizacija slučaja prirodnog splajna). Implementacija je jednostavna: postavlja se sistem jednačina po momentima u unutrašnjim čvorovima, a zatim se koristeći poznate vrednosti momenata u krajnjim čvorovima menjaju jednačine koje se odnose na dva unutrašnja čvora susedna krajnjim čvorovima i na taj način se formira trodijagonalni sistem jednačina po momentima u unutrašnjim čvorovima. Preostaje da se iskoristi procedura `tridiag()` implementirana u prvom poglavlju da se reši ovaj sistem i da se onda na osnovu vrednosti momenata izračunaju koeficijenti splajnova. Treba uočiti kako su radi efikasnosti na početku procedure formirana polja sa dužinama intervala odn. podeljenim razlikama prvog reda - ove veličine se koriste na više mesta u kodu tako da ih ima smisla predračunati. Sledi procedura koja implementira metodu:

```
#include <assert.h>
#include <stdlib.h>

extern void     tridiag(int n, double *a, double *b, double *c, double *d,
                        double *x);

/*
 * Funkcija spline() racuna koeficijente kubicnog interpolacionog
 * splajna na osnovu zadate tabele interpolacije i vrednosti momenata u
 * krajnjim cvorovima. Tabela mora sadrzati najmanje 3 cvora i cvorovi
 * moraju biti zadati u rastucem poretku. Argumenti funkcije su:
 *   n - broj podintervala  interpolacije (tj. broj cvorova je n+1)
 *   x - polje sa cvorovima interpolacije
 *   f - polje sa vrednostima funkcije u cvorovima interpolacije
 *   Ma, Mb - vrednosti momenata na granicama intervala
 *   alpha beta, gamma, delta - polja u koja ce biti smesteni
 *                              koeficijenti
 * Pretpostavlja se da su sva polja alocirana izvan funkcije spline().
 */
void
spline(int n, double *x, double *f, double Ma, double Mb, double *alpha,
       double *beta, double *gamma, double *delta)
{
        double          *M;      /* Polje sa vrednostima momenata u
                                  * cvorovima interpolacije. */
        double          *h;      /* Polje sa duzinama intervala. */
        double          *diffs;  /* Polje sa podeljenim razlikama prvog
                                  * reda. */
        double          *mu,
                        *nu,
                        *lambda; /* Polja sa koeficijentima sistema
                                  * jednacina za momente. */
        double          *twos;   /* Polje sa dvojkama koje figurisu u
                                  * jednacinama za momente. */
        int              i;      /* Brojac u petljama. */

        /* Provjerava se validnost ulaznih parametara. */
        assert(n >= 2);
```

```
for (i = 0; i < n; i++)
        assert(x[i] < x[i + 1]);

/* Alociraju se polja za duzine intervala i podeljene razlike
 * prvog reda. */
h = (double *) malloc((n + 1) * sizeof(double));
diffs = (double *) malloc((n + 1) * sizeof(double));

/* Izracunavaju se duzine intervala i podeljene razlike prvog
 * reda. */
for (i = 1; i <= n; i++) {
        h[i] = x[i] - x[i - 1];
        diffs[i] = (f[i] - f[i - 1]) / h[i];
}

/* Alociraju se polja za koeficijente sistema jednacina za
 * momente. */
mu = (double *) malloc(n * sizeof(double));
twos = (double *) malloc(n * sizeof(double));
nu = (double *) malloc(n * sizeof(double));
lambda = (double *) malloc(n * sizeof(double));

/* Izracunavaju se koeficijenti sistema jednacina za momente. Pri
 * racunanju koeficijenata na desnoj strani sistema koriste se
 * vrednosti podeljenih razlika prvog reda. */
for (i = 1; i < n; i++) {
        mu[i] = h[i] / (h[i] + h[i + 1]);
        twos[i] = 2;
        nu[i] = 1 - mu[i];
        lambda[i] =
            6 * (diffs[i + 1] - diffs[i]) / (h[i] + h[i + 1]);
}

/* Vrsi se korekcija u prvoj i poslednjoj jednacini u skladu sa
 * poznatim vrednostima momenata M[0] i M[n]. */
lambda[1] -= mu[1] * Ma;
mu[1] = 0;
lambda[n - 1] -= nu[n - 1] * Mb;
nu[n - 1] = 0;

/* Alocira se polje za momente i postavljaju se poznate vrednosti
 * momenata. */
M = (double *) malloc((n + 1) * sizeof(double));
M[0] = Ma;
M[n] = Mb;

/* Resava se sistem jednacina za momente. */
tridiag(n - 1, mu + 1, nu + 1, twos + 1, lambda + 1, M + 1);

/* Oslobadjaju se polja koeficijenata sistema jednacina za
 * momente. */
free(mu);
free(nu);
free(lambda);
free(twos);

/* Na osnovu poznatih momenata, izracunavaju se vrednosti
```

```
         * koeficijenata splajnova na svakom intervalu podele. */
        for (i = 0; i < n; i++) {
                alpha[i] = f[i];
                beta[i] =
                    diffs[i + 1] - h[i + 1] / 6 * (2 * M[i] + M[i + 1]);
                gamma[i] = M[i] / 2;
                delta[i] = 1 / (6 * h[i + 1]) * (M[i + 1] - M[i]);
        }

        /* Oslobadjaju se polja sa duzinama intervala, podeljenim
         * razlikama prvog reda i sa momentima. */
        free(h);
        free(diffs);
        free(M);
}
```

2.53 *Neka je $f(x) \in C^2[0,1]$ periodična funkcija. Ako je $S_\Delta(f;x)$ interpolacioni kubni splajn funkcije $f(x)$ određen podelom $\Delta = \{0 = x_0 < x_1 < \cdots < x_n = 1\}$, dokazati da je*

$$\max_{1\le k\le n} |S''_\Delta(f;x_k)| \le 6 \max_{1\le k\le n} |f[x_{k-1}, x_k, x_{k+1}]|$$

Rešenje: Momenti $M_k = S''_\Delta(f;x_k)$ su rešenja sistema jednačina

$$\mu_k M_{k-1} + 2M_k + \nu_k M_{k+1} = \lambda_k, \qquad k = 1, \ldots, n,$$

gde je $h_k = x_k - x_{k-1}$ i

$$\nu = \frac{h_{k+1}}{h_k + h_{k+1}}, \qquad \mu_k = 1 - \nu_k, \qquad \lambda_k = 6f[x_{k-1}, x_k, x_{k+1}].$$

Neka je $|M_r| = \max_{1\le k\le n} |M_k|$. Tada, kako je $0 < \mu_k, \nu_k < 1$ za svako k, važi ocena

$$\max_{1\le k\le n} |\lambda_k| \ge |\lambda_r| \ge 2|M_r| - \mu_r|M_{r-1}| - \nu_r|M_{r+1}| \ge (2 - \mu_r - \nu_r)|M_r| = |M_r|.$$

što je i trebalo dokazati.

2.54 *Neka su podelom $\Delta = \{x_i = x_0 + ih,\ i = 0, \ldots, n,\ h = const\}$ definisani kubni splajnovi $S_2, \ldots, S_{n-2}$ takvi da je*

$$S_j(x) \equiv 0 \quad x \notin [x_{j-2}, x_{j+2}], \qquad S_j(x_j) = 1 \qquad 2 \le j \le n-2.$$

Dokazati egzistenciju i jedinostvenost ovih tzv. B-splajnova i napisati njihov analitički izraz.

Rešenje: Zbog glatkosti splajna $S_j(x)$ je

$$S''_j(x_k) = M_k = 0, \qquad k = 0, \ldots, j-2, j+2, \ldots, n,$$

te je sistem linearnih jednačina po momentima M_k ovog splajna

$$(*) \qquad 0.5M_{k-1} + 2M_k + 0.5M_{k+1} = 6S_j[x_{k-1}, x_k, x_{k+1}], \qquad k = j-1, j, j+1.$$

Iz uslova glatkosti $S_j'(x_{j-2}) = S_j'(x_{j+2}) = 0$ sledi da je

$$M_{j-1} = \frac{6}{h^2} S_j(x_{j-1}), \qquad M_{j+1} = \frac{6}{h^2} S_j(x_{j+1}),$$

što zamenom u prvu i treću jednačinu sistema $(*)$ daje

$$S_j(x_{j-1}) = S_j(x_{j+1}) = \frac{1}{6} - \frac{h^2}{36} M_j, \qquad M_{j-1} = M_{j+1} = \frac{1}{h^2} - \frac{1}{6} M_j,$$

pri čemu je uzeto u obzir da je $S_j(x_j) = 1$. Zamenom dobijenih izraza u drugoj jednačini sistema $(*)$, dobijamo M_j, pa je

$$M_j = -\frac{3}{h^2}, \qquad M_{j-1} = M_{j+1} = \frac{3}{2h^2},$$
$$S_j(x_j) = 1, \qquad S_j(x_{j-1}) = S_j(x_{j+1}) = \frac{1}{4}.$$

Sistem $(*)$ ima jedinstveno rešenje, što znači da splajn $S_j(x)$ postoji i da je jedinstveno određen, a njegov analitički izraz je za svako $2 \le j \le n-2$

$$S_j(x) = \begin{cases} \frac{1}{4h^3}(x - x_{j-2})^3, & [x_{j-2}, x_{j-1}] \\ -\frac{3}{4h^3}(x - x_{j-1})^3 + \frac{3}{4h^2}(x - x_{j-1})^2 + \frac{3}{4h}(x - x_{j-1}) + \frac{1}{4}, & [x_{j-1}, x_j] \\ \frac{3}{4h^3}(x - x_j)^3 - \frac{3}{2h^2}(x - x_j)^2 + 1, & [x_j, x_{j+1}] \\ -\frac{1}{4h^3}(x - x_{j+1})^3 + \frac{3}{4h^2}(x - x_{j+1})^2 - \frac{3}{4h}(x - x_{j+1}) + \frac{1}{4}, & [x_{j+1}, x_{j+2}] \end{cases}$$

MATLAB

Deo-po-deo polinomijalne funkcije:	`pp, y = ppval(pp, x)`
Interpolacija:	`Y = interp1(x, y, X, type)`
	`type = 'nearest' | 'linear' | 'cubic'`
Kubni splajn:	`pp = spline(x, y)`
Uslovi na krajevima:	`pp = csape(x, y, type)`
	`type = 'parametric' | 'second' | 'complete'`

MATLAB direktno podržava neke metode interpolacije deo-po-deo polinomijalnim funkcijama, odnosno splajnovima. Funkcija `Y = interp1(x, y, X, type)` vrši jednodimenzionalnu interpolaciju. Vektorima `x` i `y` se zadaje tabela interpolacije, dok vektor `X` predstavlja tačke u kojima se izračunava vrednost rezultujuće interpolacione funkcije. Poslednji parametar `type` predstavlja tip interpolacije i to:

`nearest` - vrši se interpolacija deo-po-deo konstantnom funkcijom (stepenasta interpolacija)

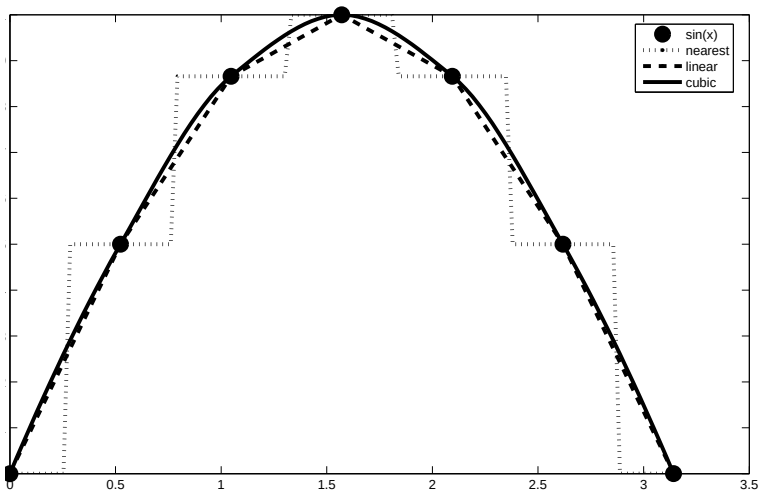
`linear` - vrši se interpolacija deo-po-deo linearnom funkcijom (linearnim splajnom)

`cubic` - vrši se interpolacija deo-po-deo kubnom funkcijom (kubnim splajnom). U ovom slučaju se automatski poziva funkcija `spline` koja će u daljem tekstu biti detaljnije opisana.

Kroz sledeći skript, dat je primer korišćenja funkcije `interp1`.

```
% Interpolisemo sin(x) na [0, pi] sa 7 ekvidistantnih cvorova
x = linspace(0, pi, 7);
y = sin(x);

% Mreza za iscrtavanje
X = linspace(0, pi);

% Vrsimo 3 vsrte interpolacije
Y_n = interp1(x, y, X, 'nearest');
Y_l = interp1(x, y, X, 'linear');
Y_c = interp1(x, y, X, 'cubic');

% Prikazujemo rezultate
plot(x, y, 'o', X, Y_n, X, Y_l, X, Y_c);
legend('sin(x)', 'nearest', 'linear', 'cubic');
```

Rezultat rada skripta je prikazan na slici 2.4.



Slika 2.4: Funkcija `interp1`

MATLAB poseduje napredne mogućnosti rada sa splajnovima i sa funkcijama koje su deo-po-deo polinomijalne. Većina ovih funkcija pripadaju modulu za rad sa splajnovima *(spline toolbox)*, međutim, i osnovni MATLAB paket poseduje izvesne mogućnosti *kubne splajn interpolacije*.

Sve deo-po-deo polinomijalne funkcije se u MATLAB-u predstavljaju posebnom strukturom podataka koja se naziva `pp` *(piecewise polynomial)*. Vrednosti ovakvih funkcija se izračunavaju korišćenjem funkcije `ppval`.

Funkcija `spline`, koja pripada osnovnom MATLAB paketu, za datu tabelu interpolacije vraća `pp` strukturu koja predstavlja konstruisani kubni splajn. Tabela interpolacije je predstavljena vektorima `x` i `y`. Ukoliko vektor `y` ima dve vrednosti više od vektora `x`, prva i poslednja vrednost se uzimaju za vrednosti izvoda funkcije na krajevima. U suprotnom, koriste se takozvani *not-a-knot* uslovi na krajevima. Ukoliko se kao treći argument funkcije `spline` navede vektor tačaka `X`, funkcija `spline` vraća vrednosti izračunatog splajna upravo u tačkama vektora `X`, tako da se ovim eksplicitno korišćenje `pp` strukture može u potpunosti zaobići.

Kao što je napomenuto, napredne mogućnosti rada sa splajnovima su date kroz funkcije *spline toolbox*-a. Iz širokog spektra mogućnosti ovog modula izdvojićemo samo neke.

Funkcija `csape(x, y, endcond)` vrši interpolaciju kubnim splajnovima uz mogućnost zadavanja različite vrste uslova na krajevima. Navodimo neke od vrednosti parametra `endcond`.

`'periodic'`	vrši se konstrukcija periodičnog splajna
`'second'`	navode se drugi izvodi splajna na krajevima intervala interpolacije. Podrazumevano je da su vrednosti ovih izvoda 0, odnosno da se konstruiše prirodni splajn. Eksplicitne vrednosti drugih izvoda je moguće navesti proširivanjem vektora `y`, slično kao što je slučaj i kod funkcije `spline`.
`'complete'`	Navode se prvi izvodi splajna, uz proširivanje vektora `y`.

2.55 *Korišćenjem* MATLAB-*a konstruisati kubni interpolacioni splajn* $S(x)$ *za tabelarno zadatu funkciju*

x	1	2	3	4
$f(x)$	0	1	2	0

pri čemu je splajn:

i) prirodni,

ii) periodični,

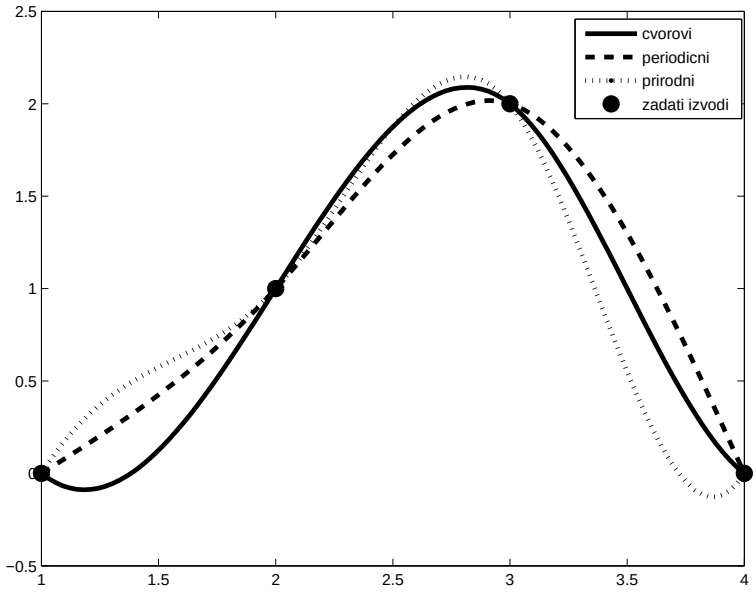
iii) sa zadatim izvodom na granici, $S'(1) = 2,\ S'(4) = 2.$

Izračunati vrednost periodičnog splajna u tački $x = 2.5$*, a zatim grafički prikazati sve splajnove.*

Rešenje:

```
% Cvorovi interpolacije
x = [1 2 3 4]; y = [0 1 2 0];

% Konstruisemo splajnove
spl_per = csape(x, y, 'periodic');          %periodicni
spl_nat = csape(x, y, 'second');            %prirodni
spl_es  = csape(x, [2 y 2], 'complete');    %zadati izvodi

% Vrednost periodicnog splajna u 2.5
ppval(spl_per, 2.5)

% Crtanje splajnova na [1, 4]
xx = linspace(1, 4);
plot(xx, ppval(spl_per, xx), ...
     xx, ppval(spl_nat, xx), ...
     xx, ppval(spl_es, xx), ...
     x, y, 'o')
legend('cvorovi', 'periodicni', 'prirodni', 'zadati izvodi')
```

Rezultat rada skripta je prikazan na slici 2.5.



Slika 2.5: Kubni splajnovi (`csape`)

2.7 Numeričko diferenciranje

Za aproksimaciju izvoda funkcije se obično koristi izvod njenog interpolacionog polinoma

$$f^{(k)}(x) = L_n^{(k)} + R_n^{(k)},$$

gde je $L_n(x)$ je interpolacioni polinom i $R_n(x)$ greška interpolacije. Greška numeričkog diferenciranja se može oceniti sledećim izrazom

$$|R_n^{(k)}(x)| \leq \sum_{j=0}^{k} \frac{k!}{(k-j)!(n+j+1)!} \max_{[y_1,y_2]} |f^{(n+j+1)}(\xi)|\, |\omega_{n+1}^{(k-j)}(x)|,$$

za $y_1 = \min(x, x_0, \ldots, x_n)$ i $y_2 = \max(x, x_0, \ldots, x_n)$.

2.56 *Sa tačnošću* 10^{-4} *odrediti koordinate tačke ekstrema funkcije date tabelom*

x	1.3	1.4	1.5	1.6	1.7	1.8
$f(x)$	1.2431	1.1486	1.2095	1.4606	1.9391	2.6846

Rešenje: Apscisa tačke ekstrema je nula prvog izvoda funkcije, koji ćemo aproksimirati prvim izvodom interpolacionog polinoma. Napišimo prvo tabelu konačnih razlika

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$
1.3	1.2431				
		-0.0945			
1.4	1.1486		0.1554		
		0.0609		0.0348	
1.5	1.2095		0.1902		0.0024
		0.2511		0.0372	
1.6	1.4606		0.2274		0.0024
		0.4785		0.0396	
1.7	1.9391		0.2670		
		0.7455			
1.8	2.6846				

S obzirom da u okolini čvora 1.4 konačna razlika prvog reda menja znak, funkcija dostiže ekstremnu vrednost u okolini te tačke. Radi smanjenja numeričke greške ćemo, stoga, za početni čvor Newton-ovog interpolacionog polinoma za interpolaciju unapred izabrati $x_0 = 1.4$ i uvesti smenu $q = (x - x_0)/h$, gde je $h = 0.1$ korak tabele.

Newton-ov polinom četvrtog stepena je

$$L_4(x_0+qh) = f_0 + q\,\Delta f_0 + \frac{1}{2!}q(q-1)\Delta^2 f_0 + \frac{1}{3!}q(q-1)(q-2)\Delta^3 f_0 + \frac{1}{4!}q(q-1)(q-2)(q-3)\Delta^4 f_0,$$

gde su $\Delta^k f_i$ podvučene vrednosti u tabeli. Metodom iteracije nalazimo da je $q_e = 0.142$ rešenje jednačine $L_4'(x_0+qh) = 0$. Stoga su koordinate tačke ekstrema funkcije $f(x)$ približno

$$x_e = x_0 + q_e h = 1.4 + 0.142 \cdot 0.1 = 1.4142, \qquad f(x_e) \approx L_4(1.4142) = 1.1470.$$

Napomena: Ako se dobije da je $q_e < 0$, tražena tačka je van intervala $[x_0, x_n]$. Polinom je korišćen za ekstrapolaciju, pa će numerička greška biti mnogo veća. Stoga je potrebno ponoviti postupak tako što ćemo aproksimirati funkciju $f(x)$ polinomom napisanim u odnosu na čvor $x_0 = 1.3$.

2.57 *Korišćenjem formula za numeričko diferenciranje sa konačnim razlikama do četvrtog reda, odrediti koeficijente diferencijalne jednačine*

$$u''(x) + a\,u'(x) + b\,u(x) = x, \qquad a,\, b = const,$$

koju zadovoljava funkcija $u(x)$ data tabelom

x	0.6	0.7	0.8	0.9	1.0	1.1
$u(x)$	1.164642	1.344218	1.517356	1.683327	1.841471	1.991207

Rešenje: Formiramo tabelu konačnih razlika funkcije $u(x)$

x	u	Δu	$\Delta^2 u$	$\Delta^3 u$	$\Delta^4 u$
0.6	1.164642				
		0.179576			
0.7	1.344218		−0.006438		
		0.173138		−0.000729	
0.8	1.517356		−0.007167		0.000069
		0.165971		−0.000660	
0.9	1.683327		−0.007827		0.000079
		0.158144		−0.000581	
1.0	1.841471		−0.008408		
		0.149736			
1.1	1.991207				

Aproksimiramo funkciju $u(x)$ Bessel-ovim interpolacionim polinomom četvrtog stepena napisanim u odnosu na čvor $x_0 = 0.8$

$$L_4(x_0+qh) = 0.000003q^4 - 0.000112q^3 - 0.003587q^2 + 0.169667q + 1.517359,$$

gde je $q = (x - x_0)/h = 10(x - 0.8)$, a njene izvode odgovarajućim izvodima ovog polinoma. Zamenom u diferencijalnoj jednačini funkcije $u(x)$ i njenih izvoda sa L_4, L_4' i L_4'', a argumenta x sa $x = 0.8 + 0.1q$, dobijamo identitet po q

$$\begin{aligned}0.000003b\,q^4 + (0.00012a - 0.000112b)q^3 - (0.00336a + 0.003587b - 0.0036)q^2 \\ -(0.07174a - 0.169667b + 0.0672)q + (1.69667a + 1.51359b - 0.7174) \\ = 0.1q + 0.8.\end{aligned}$$

Izjednačavanjem koeficijenata uz odgovarajuće stepene promenljive q na levoj i desnoj strani identiteta, dobijamo preodređeni sistem linearnih jednačina po koeficijentima a i b. Njegovo rešenje metodom najmanjih kvadrata je $a = 0$ i $b = 1$, te je diferencijalna jednačina koju zadovoljava data funkcija

$$u''(x) + u(x) = x.$$

2.58 *a) Dokazati da je za svako x i za svako n*

$$\lim_{h\to 0}\left(\frac{1}{h}\sum_{k=1}^{n}\frac{(-1)^{k+1}}{k}\,\Delta^k f(x)\right) = f'(x).$$

b) Pomoću ovog izraza izračunati $f'(1)$ ako je $f(x) = \arctan(e^{x^2})$ sa tačnošću $5 * 10^{-4}$.

Rešenje: Iz poznate veze izvoda i konačne razlike funkcije

$$\Delta^k f(x) = h^k f^{(k)}(\xi_k), \qquad \xi_k \in (x, x + kh),$$

sledi da je za $n = 1$

$$\lim_{h\to 0}\frac{1}{h}\,\Delta f(x) = \lim_{h\to 0}\frac{1}{h}\,hf'(\xi_1) = f'(x), \qquad \xi_1 \in [x, x + h]$$

jer $\xi_1 \to x$ kada dužina intervala kome pripada ova tačka teži nuli. Kada je $n > 1$ imamo da je za $\xi_k \in [x, x + kh]$

$$\begin{aligned}\lim_{h\to 0}\left(\frac{1}{h}\sum_{k=1}^{n}\frac{(-1)^{k+1}}{k}\,\Delta^k f(x)\right) &= \lim_{h\to 0}\left(\frac{1}{h}\sum_{k=1}^{n}\frac{(-1)^{k+1}}{k}\,h^k f^{(k)}(\xi_k)\right) \\ = \lim_{h\to 0}\left(f'(\xi_1) + \sum_{k=2}^{n}\frac{(-1)^{k+1}}{k}\,h^{k-1} f^{(k)}(\xi_k)\right) &= \lim_{h\to 0} f'(\xi_1) = f'(x)\end{aligned}$$

na osnovu dokaza za $n = 1$.

b) Tabelirajmo funkciju $f(x)$ sa korakom $h = 0.1$ počev od tačke $x = 1$

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$
1.0	1.21828					
		0.06271				
1.1	1.28099		−0.00554			
		0.05717		−0.00146		
1.2	1.33816		−0.00700		0.00082	
		0.05017		−0.00064		−0.00014
1.3	1.38833		−0.00764		0.00068	
		0.04253		0.00004		
1.4	1.43086		−0.00760			
		0.03493				
1.5	1.46579					

Greška formule za $n = 4$ se može oceniti izrazom

$$|f'(1) - 10\sum_{k=1}^{4} \frac{(-1)^{k+1}}{k}\,\Delta^k f(1)| = \frac{h^4}{5}\,|f^{(5)}(\xi)| \approx \frac{1}{5h}\,|\Delta^5 f| = 2.8 * 10^{-4},$$

pa će suma prva četiri člana reda aproksimirati $f'(1)$ sa zadovoljavajućom tačnošću,

$$\left(\arctan\left(e^{x^2}\right)\right)'_{x=1} = 0.648.$$

2.59 *Neka je* $\max_{[a,b]} |f^{(k)}(x)| \le M_k$, $k = 0, 1, \ldots$. *Sa kojim korakom* h *treba tabelirati funkciju* $f(x)$ *na intervalu* $[a, b]$ *tako da ukupna greška diferenciranja po formuli*

$$f''(x) \approx L''(x) = \frac{1}{12h^2}\big(-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)\big)$$

bude minimalna.

Rešenje: Razvojem u Taylorov red u okolini tačke x veličina $f(x \pm kh)$, $k = 1, 2$, dobijamo da je greška formule

$$|R_i| = |f''(x) - L''(x)| \le \frac{h^4}{54}\,M_6.$$

Pod pretpostavkom da su sve vrednosti funkcije date sa apsolutnom greškom ε, računska greška formule je

$$|R_r| \le \frac{1}{12h^2}\,64\varepsilon = \frac{16}{3}\,\frac{\varepsilon}{h^2}.$$

Ukupna greška rezultata određenog datom formulom nije veća od

$$|R| = |R_i + R_r| \le \frac{M_6}{54}\,h^4 + \frac{16}{3}\,\frac{\varepsilon}{h^2}.$$

Dobijena ocena je funkcija koraka h, i ona dostiže minimalnu vrednost za

$$h_o = \sqrt[6]{144\varepsilon/M_6}, \qquad |R| \le 1.11\sqrt[3]{\varepsilon^2 M_6}.$$

Dakle, poslednjim izrazima su date optimalna vrednost koraka koja daje minimalnu ukupnu grešku i procena te greške.

2.60 *Funkcija*

$$f(x) = \int_0^x \arctan t^2\, dt$$

tabelirana je na intervalu $[0,1]$ *sa tačnošću* $\varepsilon = 5 * 10^{-7}$. *Za koji korak* h *se dostiže maksimalna tačnost formulom za približno diferenciranje*

$$f'(x) \approx L'(x) = \frac{1}{2h}\big(-3f(x) + 4f(x+h) - f(x+2h)\big).$$

Rešenje: Grešku formule ocenjujemo razvojem u Taylorov red u okolini tačke x veličina $f(x+h)$ i $f(x+2h)$,

$$|R_i| = |f'(x) - L'(x)| \le h^2 M_3, \qquad M_3 = \max_{[0,1]} |f'''(\xi)|,$$

a računska greška je

$$|R_r| \le \frac{1}{2h}\, 8\varepsilon.$$

Ukupna greška je

$$|R| \le |R_i| + |R_r| = h^2 M_3 + 4\varepsilon/h.$$

Ona je minimalna kada je

$$h = h_0 = \sqrt[3]{\frac{2\varepsilon}{M_3}}.$$

Ocenimo još veličinu M_3. Kako je

$$f^{(4)}(x) = -\frac{8x^3(5-2x^4)}{(1+x^4)^3} < 0$$

na posmatranom intervalu, funkcija $f'''(x)$ je opadajuća,

$$f'''(x) = 2\frac{1-3x^4}{(1+x^4)^2}, \qquad \text{i} \qquad \max_{[0,1]} |f'''(x)| = f'''(0) = 2.$$

Zamenom u izvedeni izraz za h_o dobijamo da su u ovom zadatku optimalna vrednost koraka i greška

$$h_o = 0.8 * 10^{-2}, \qquad |R| \le 3.8 * 10^{-4}.$$

2.61 *Dokazati da je za proizvoljno* $x \in [x_0, x_{k+1}]$

$$f^{(k)}(x) - L_{k+1}^{(k)}(x) = O(h^2),$$

gde je $L_{k+1}(x)$ *interpolacioni polinom funkcije* $f(x) \in \mathcal{C}^{2k+2}[x_0, x_{k+1}]$ *generisan čvorovima* $x_0, \ldots, x_{k+1}$ *i* $x_i = x_0 + ih$, $i = 0, \ldots, k+1$.

Takođe pokazati da postoje dve tačke,

$$\zeta_{1/2} = \frac{1}{k+2}\left(\sum_{i=0}^{k+1} x_i \pm \frac{1}{\sqrt{k+1}}\sqrt{\sum_{i=0}^{k}\sum_{j=i+1}^{k+1}(x_i - x_j)^2}\right)$$

u kojima je greška reda $O(h^3)$, *tj.*

$$f^{(k)}(\zeta_i) - L_{k+1}^{(k)}(\zeta_i) = O(h^3), \qquad i = 1, 2.$$

Rešenje: Greška numeričkog diferenciranja je

$$R_{k+1}^{(k)}(x) = f^{(k)}(x) - L_{k+1}^{(k)}(x) = \big(f(x) - L_{k+1}(x)\big)^{(k)}$$

$$= \sum_{j=0}^{k} \frac{k!}{(k-j)!}\,\frac{f^{(k+j+2)}(\xi)}{(k+j+2)!}\,\omega_{k+2}^{(k-j)}(x),$$

gde je

$$\xi \in [\min(x, x_0, \ldots, x_{k+1}), \max(x, x_0, \ldots, x_{k+1})], \qquad \omega_{k+2}(x) = \prod_{i=0}^{k+1}(x - x_i).$$

Kako je $f^{(k+2+j)}(x)$ ograničena veličina koja ne zavisi od h, a $\omega_{k+2}^{(k-j)}(x) = O(h^{2+j})$ za svako j, to je

$$R_{k+1}^{(k)}(x) = O(h^2),$$

jer najveću težinu u grešci ima sabirak za $j = 0$. On će se anulirati, tj. greška će biti $O(h^3)$, u tački ζ u kojoj je $\omega_{k+2}^{(k)}(\zeta) = 0$. Ako je

$$\omega_{k+2}(x) = x^{k+2} + a_1 x^{k+1} + a_2 x^k + \cdots + a_{k+2},$$

onda je

$$\omega_{k+2}^{(k)}(x) = k!\left(\frac{1}{2}(k+2)(k+1)x^2 + (k+1)a_1 x + a_2\right).$$

Nule ovog polinoma drugog stepena su tražene tačke

$$\zeta_{1/2} = \frac{1}{k+2}\left(-a_1 \pm \frac{1}{\sqrt{k+1}}\sqrt{(k+1)a_1^2 - 2(k+2)a_2}\right).$$

Pošto su čvorovi interpolacije $x_0, \ldots, x_{k+1}$ nule polinoma $\omega_{k+2}(x)$, na osnovu Vieteovih formula je

$$a_1 = -\sum_{i=0}^{k+1} x_i, \qquad a_2 = \sum_{i=0}^{k} \sum_{j=i+1}^{k+1} x_i x_j.$$

Stoga je potkorena veličina u izrazu za $\zeta_{1/2}$

$$\begin{aligned}(k+1)a_1^2 - 2(k+2)a_2 &= (k+1)\left(\sum_{i=0}^{k+1} x_i\right)^2 - 2(k+2)\sum_{i=0}^{k}\sum_{j=i+1}^{k+1} x_i x_j \\ &= (k+1)\sum_{i=0}^{k+1} x_i^2 - 2\sum_{i=0}^{k}\sum_{j=i+1}^{k+1} x_i x_j = \sum_{i=0}^{k}\sum_{j=i+1}^{k+1}(x_i - x_j)^2\end{aligned}$$

što konačno daje traženi izraz za tačke u kojima se postiže povišena tačnost.

3

Numerička integracija

Kvadraturnom formulom S se određuje približna vrednost integrala sa greškom R,

$$\int_a^b p(x)f(x)\,dx = S + R,$$

gde je težinska funkcija $p(x) > 0$ neprekidna na (a, b).

3.1 *Sa tačnošću* $5 \cdot 10^{-4}$ *izračunati*

$$I_n = \int_0^1 \frac{x^n}{x+7}\,dx, \qquad n = 0, \ldots, 8.$$

Rešenje: Očigledno je da je

$$I_n = \int_0^1 x^{n-1}\,dx - 7I_{n-1} = \frac{1}{n} - 7I_{n-1}, \qquad n = 1, \ldots 8,$$

$$I_0 = \int_0^1 \frac{1}{x+7}\,dx = \ln\frac{8}{7}.$$

Označimo sa $A(I_n)$ linearnu apsolutnu grešku približnog broja I_n. Kako je apsolutna greška zbira ili razlike jednaka zbiru apsolutnih grešaka argumenata, sledi da je

$$A(I_n) = A\big(\frac{1}{n}\big) + 7A(I_{n-1}) = A\big(\frac{1}{n}\big) + 7A\big(\frac{1}{n-1}\big) + 7^2A(I_{n-2}) = \ldots$$
$$= \sum_{k=1}^{n} 7^{n-k}A\big(\frac{1}{k}\big) + 7^nA(I_0).$$

Brojevi 1, 1/2, 1/4, 1/5, 1/8 se mogu tačno izračunati (za njih je $A(1/k) = 0$), a pretpostavimo da su brojevi 1/3, 1/6, 1/7 i $A(I_0)$ izračunati sa istom apsolutnom

greškom Δ. Da bi poslednja u nizu traženih vrednosti integrala I_8, za koju je računska greška očigledno najveća, bila izračunata sa traženom tačnošću trebalo bi da je

$$A(I_8) = (7^5 + 7^2 + 7)\Delta + 7^8\Delta = 5781664\Delta \leq 5 \cdot 10^{-4}, \qquad \text{tj.} \qquad \Delta \leq 8.6 \cdot 10^{-11}.$$

Dakle, svi približni brojevi moraju biti računati na 11 decimala da bi svi integrali bili izračunati sa tri sigurne decimale.

3.1 Newton–Cotes-ove kvadraturne formule

Dobijaju se aproksimacijom podintegralne funkcije $f(x)$ interpolacionim polinomom $L_n(x)$, a greška je jednaka integralu greške interpolacije $r_n(x)$,

$$S_n = \int_a^b p(x)L_n(x)\,dx, \qquad R_n = \int_a^b p(x)r_n(x)\,dx.$$

Greška se ocenjuje izrazom

$$|R_n| \leq \frac{1}{(n+1)!}M_{n+1}\int_a^b p(x)|\omega_{n+1}(x)|\,dx, \qquad M_{n+1} = \max_{[a,b]}|f^{(n+1)}(x)|.$$

Red greške kvadraturne formule je jednak p ako je kvadraturna formula tačna za proizvoljan polinom stepena p, a postoji polinom stepena $p+1$ za koga ona nije tačna.

Trapezna kvadraturna formula $(n = 1)$

$$S_1 = \frac{h}{2}\left(f(a) + 2\sum_{k=1}^{m-1} f(x_k) + f(b)\right)$$

gde je težinska funkcija $p(x) \equiv 1$ i $h = (b-a)/m$, $x_k = a + kh$. Greška kvadraturne formule R ocenjuje se izrazom

$$|R_1| \leq \frac{b-a}{12}\,h^2 \max_{[a,b]}|f''(x)|$$

Red greške je $p = 2$.

Simpson-ova kvadraturna formula $(n = 2)$

$$S_2 = \frac{h}{3}\left(f(a) + 4\sum_{k=1}^{m} f(x_{2k-1}) + 2\sum_{k=1}^{m-1} f(x_{2k}) + f(b)\right),$$

gde je težinska funkcija $p(x) \equiv 1$ i $h = (b-a)/(2m)$, $x_k = a + kh$.Greška kvadraturne formule R ocenjuje se izrazom

$$|R_2| \leq \frac{b-a}{180}\, h^4 \max_{[a,b]} |f^{(4)}(x)|.$$

Red greške je $p = 4$.

Runge-ov kriterijum za približnu ocenu greške kvadraturne formule

$$R \approx \frac{S^{(h)} - S^{(2h)}}{2^p - 1}$$

gde je $S^{(h)}$ približna vrednost izračunata kvadraturnom formulom sa korakom h, a $S^{(2h)}$ sa korakom $2h$. Parametar p je red greške kvadraturne formule.

3.2 *Odrediti Newton–Cotes-ove kvadraturne formule zatvorenog tipa sa ekvidistantnim čvorovima oblika*

$$\int_0^1 p(x)f(x)\,dx = \sum_{k=0}^{n} c_k f(x_k) + R,$$

za $n = 1$ i $n = 2$, ako je težinska funkcija a) $p(x) = \sqrt{x}$, b) $p(x) = 1/\sqrt{x}$.

Rešenje: Kako su formule zatvorenog tipa, granice intervala integracije moraju biti čvorovi kvadraturne formule, te je $x_k = k/n$, $k = 0, \ldots, n$. Koeficijenti c_k se određuju iz uslova da formula bude tačna, tj. da je greška $R = 0$, za polinome $f(x) = x^k, k = 0, \ldots n$. Tako se dobijaju formule

$$(a) \qquad n = 1 \qquad \int_0^1 \sqrt{x} f(x)\,dx = \frac{2}{15}\left(2f(0) + 3f(1)\right) + R,$$

$$n = 2 \qquad \int_0^1 \sqrt{x} f(x)\,dx = \frac{2}{105}\left(2f(0) + 24f(\frac{1}{2}) + 9f(1)\right) + R,$$

$$(b) \qquad n = 1 \qquad \int_0^1 \frac{1}{\sqrt{x}} f(x)\,dx = \frac{2}{3}\left(2f(0) + f(1)\right) + R,$$

$$n = 2 \qquad \int_0^1 \frac{1}{\sqrt{x}} f(x)\,dx = \frac{2}{15}\left(6f(0) + 8f(\frac{1}{2}) + f(1)\right) + R,$$

Red greške formula je n.

3.3 *Odrediti koeficijente c_k, $k = 0, 1, 2$, i grešku R kvadraturne formule*

$$\int_0^2 xf(x)\,dx = c_0 f(0) + c_1 f(1) + c_2 f(2) + R$$

tako da red greške bude dva.

Rešenje: Funkciju $f(x)$ aproksimirajmo Lagrange-ovim interpolacionim polinomom drugog stepena $L_2(x)$ određenog čvorovima kvadraturne formule $x_k = k$, $k = 0, 1, 2$,

$$f(x) = L_2(x) + r(x) = \sum_{k=0}^{2} \left(\prod_{\substack{j=0 \\ j \neq k}}^{2} \frac{x - x_j}{x_k - x_j} \right) f(x_k) + \frac{1}{3!} f'''\big(\xi(x)\big)\omega_3(x),$$

gde je $\omega_3(x) = \prod_{k=0}^{2}(x - x_k)$. Zamenom u integralu funkcije $f(x)$ prethodnim izrazom dobijamo da je

$$\int_0^2 x f(x)\, dx = \frac{4}{3} f(1) + \frac{2}{3} f(2) + \frac{1}{6} \int_0^2 x\omega_3(x) f'''\big(\xi(x)\big)\, dx.$$

3.4 *Izvesti kvadraturnu formulu*

$$\int_{x_0}^{x_1} f(x)\, dx = (x_1 - x_0) f(x_0) + \frac{1}{2}(x_1 - x_0)^2 f[x_0, x_1] - \frac{1}{6}(x_1 - x_0)^3 f[x_0, x_1, x_2] + R,$$

gde je

$$R = \int_{x_0}^{x_1} (x - x_0)(x - x_1)(x - x_2) f[x, x_0, x_1, x_2]\, dx.$$

Rešenje: Aproksimacijom podintegralne funkcije Newton-ovim interpolacionim polinomom drugog stepena određenim čvorovima x_0, x_1 i x_2,

$$\begin{aligned} f(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ + (x - x_0)(x - x_1)(x - x_2) f[x, x_0, x_1, x_2] \end{aligned}$$

neposredno dobijamo traženu formulu.

3.5 *Odrediti koeficijente c_k, $k = 0, 1, 2$, kao funkcije od α tako da je formula*

$$\int_{-1}^{1} f(x)\, dx = c_0 f(-\alpha) + c_1 f(0) + c_2 f(\alpha) + R, \qquad 0 < \alpha \leq 1,$$

tačna za sve polinome stepena tri i nižeg od tri. Pokazati da je red greške formule pet za $\alpha = \sqrt{3/5}$.

Rešenje: Da bi formula bila tačna za sve polinome stepena manjeg ili jednakog tri, treba da bude tačna za bazisne polinome x^k, $k = 0, \ldots, 3$. Ovaj uslov daje sistem linearnih jednačina

$$\int_{-1}^{1} x^k\, dx = c_0(-\alpha)^k + c_1 0^k + c_2 \alpha^k, \qquad k = 0, 1, 2,$$

čije rešenje je $c_0 = c_2 = 1/(3\alpha^2)$ i $c_1 = 2(1 - 1/(3\alpha^2))$. Tražena formula je

$$\int_{-1}^{1} f(x)\,dx = \frac{1}{3\alpha^2}\big(f(-\alpha) + 2(3\alpha^2 - 1)f(0) + f(\alpha)\big) + R,$$

i ona je za svako α tačna i za $f(x) = x^3$. Za $f(x) = x^4$ je

$$R = \int_{-1}^{1} x^4\,dx - \frac{1}{3\alpha^2}(\alpha^4 + \alpha^4) = \frac{2}{3}\left(\frac{3}{5} - \alpha^2\right) = 0 \quad \text{za} \quad \alpha = \sqrt{\frac{3}{5}}.$$

Kako je za $f(x) = x^5$ greška $R = 0$ za svako α, a za $f(x) = x^6$ greška $R \neq 0$ za $\alpha = \sqrt{\frac{3}{5}}$, sledi da je izvedena kvadraturna formula reda tri, osim za $\alpha = \sqrt{\frac{3}{5}}$, kada je reda pet.

Napomena: Za neke vrednosti parametra α dobijaju se dobro poznate formule: $\alpha = 1$ Osnovna Simpson-ova formula

$$\int_{-1}^{1} f(x)\,dx = \frac{1}{3}\big(f(-1) + 4f(0) + f(1)\big) + R$$

$\alpha = \sqrt{3/5}$ Gauss-ova formula sa tri čvora

$$\int_{-1}^{1} f(x)\,dx = \frac{1}{9}\left(5f\big(-\sqrt{\frac{3}{5}}\big) + 8f(0) + 5f\big(\sqrt{\frac{3}{5}}\big)\right) + R.$$

3.6 *Izvesti kvadraturnu formulu oblika*

$$\int_{0}^{1} f(x)\,dx = c_1 f\big(\frac{1}{4}\big) + c_2 f\big(\frac{1}{2}\big) + c_3 f\big(\frac{3}{4}\big) + R$$

i oceniti grešku R formule. Pomoću dobijene formule izracunati

$$\int_{\pi/4}^{\pi/2} \frac{\sin x}{x}\,dx.$$

Rešenje: Koeficijente c_k, $k = 1, 2, 3$ određujemo iz uslova da formula bude tačna za polinome što je moguće višeg stepena. Kako je broj neodređenih parametara u formuli tri, formula treba da bude tačna za polinome $f(x) = x^k$, $k = 0, 1, 2$. Ovi uslovi daju sistem linearnih jednačina, čije rešenje je $c_1 = c_3 = 2/3$ i $c_2 = -1/3$. Stoga je tražena formula

$$\int_{0}^{1} f(x)\,dx = \frac{1}{3}\left(2f\big(\frac{1}{4}\big) - f\big(\frac{1}{2}\big) + 2f\big(\frac{3}{4}\big)\right) + R.$$

Ocenimo i grešku R formule. Ovo je formula Newton–Cotes-ovog tipa sa tri čvora, pri čemu čvorovima koji su simetrični u odnosu na sredinu intervala odgovaraju

jednaki koeficijenti. Stoga je red greške formule uvećan za jedan, tj. formula je tačna i za polinome trećeg stepena, pa je

$$R = \frac{1}{4!}\int_0^1 f^{(4)}(\xi)(x-1/4)(x-1/2)^2(x-3/4)\,dx.$$

Ako označimo sa $M_4 = \max_{[0,1]} |f^{(4)}(x)|$, sledi da je

$$|R| \le \frac{M_4}{24}\int_0^1 |(x-1/4)(x-1/2)^2(x-3/4)|\,dx \le 0.00033 M_4.$$

Za izračunavanje datog integrala izvedenom formulom, prethodno smenom $x = (t+1)\pi/4$ transformišemo interval $[\pi/4, \pi/2]$ u $[0,1]$,

$$\int_{\pi/4}^{\pi/2} \frac{\sin x}{x}\,dx = \int_0^1 \frac{\sin\big((t+1)\pi/4\big)}{t+1}\,dt \approx 0.61178.$$

3.7 *Trapeznom kvadraturnom formulom sa tačnošću* $5\cdot 10^{-5}$ *izračunati približnu vrednost integrala*

$$\int_0^{\pi/2} \sqrt{1-0.25\sin^2 x}\,dx.$$

Rešenje: Približna vrednost integrala izračunata trapeznom formulom je

$$S_1 = 1.467459 \quad \text{za} \quad h_1 = \frac{\pi}{4}, \qquad S_2 = 1.467462 \quad \text{za} \quad h_2 = \frac{\pi}{8},$$

a procena greške Runge-ovim kriterijumom je

$$\frac{S_2 - S_1}{3} = 10^{-6}$$

Stoga je sa traženom tačnošću

$$\int_0^{\pi/2} \sqrt{1-0.25\sin^2 x}\,dx = 1.46746.$$

C Pri implemenentaciji integracije trapeznom formulom primenjeno je rešenje sa adaptivnim brojem podintervala da bi bila postignuta tražena tačnost. Naime, u svakom koraku se računa vrednost integrala sa dvostruko većim brojem podintervala i postupak se zaustavlja kada razlika vrednosti izračunatih u dve uzastopne iteracije po apsolutnoj vrednosti postane manja od date tačnosti. Kao zaštita od slučajeva kada metoda ne konvergira, očekuje se od korisnika da specificira i maksimalni dozvoljeni broj podintervala. Ukoliko se ne postigne zadata tačnost sa maksimalno dozvoljenim brojem podintervala, iterativni postupak se takođe zaustavlja.

Jedan složen korak u proceduri je izračunavanje vrednosti funkcije za date vrednosti nezavisne promenljive. Ova funkcionalnost je potrebna velikom broju metoda,

naročito metodama iz drugog dela ove zbirke. Većina numeričkih biblioteka, npr. biblioteka koja prati *Numerical Recipes* ([20]), zahteva od korisnika da u ovakvim slučajevima sami napišu proceduru koja izračunava vrednost date funkcije. Ovde je umesto toga napisana posebna biblioteka koja implementira takvu funkcionalnost; biblioteka očekuje da joj se prenese string koji predstavlja funkciju i onda na osnovu toga može da izračunava vrednosti funkcije za razne vrednosti promenljivih ili čak da analitički odredi izvod funkcije po nekoj od promenljivih. Ovakav pristup je za nijansu sporiji, ali zato daleko fleksibilniji od prvopomenutog. Biblioteka koja implementira ovu funkcionalnost za potrebe biblioteke `libnumerics` je nazvana `libmahteval` i dostupna je sa *Web* strane ([26]). Biblioteka `libmatheval` se sastoji od malog broja procedura i koristi se dosta intuitivno, ali sadrži detaljnu dokumentaciju tako da se čitalac upućuje da konsultuje istu u slučajevima kada korišćenje nekih procedure iz te biblioteke u kodu predstavljenom u ovoj zbirci ne bude jasno. Treba uočiti kako je u proceduri za integraciju trapeznom formulom u svakoj iteraciji korišćen rezultat prethodne iteracije kako bi se minimizovao broj poziva funkcija iz biblioteke `libmatheval`.

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <matheval.h>

/*
 * Funkcija trapeze() izracunava integral funkcije na datom intervalu
 * koriscenjem trapezne formule. Iterativni postupak unutar funkcije traje
 * dok se postigne zadata tacnost ili dok se ne prekoraci zadati maksimalni
 * broj iteracija. Argumenti funkcije su:
 *   function - string koji predstavlja  funkciju
 *   a, b - granice intervala na kome se racuna integral
 *   epsilon - zeljena tacnost
 *   max_iterations - maksimalni dozvoljeni broj podintervala
 * Funkcija vraca izracunatu vrednost integrala. Argumenti funkcije
 * moraju da zadovolje uslov da je funkcija sintaksno ispravno zadata
 * (za nacin zadavanja funkcije videti dokumentaciju libmatheval
 * biblioteke), te da duzina intervala i tacnost budu pozitivni, a
 * maksimalni broj podintervala broj veci ili jednak 2.
 */
double
trapeze(char *function, double a, double b, double epsilon,
        int max_subintervals)
{
        double          value;  /* Rezultat izracunavanja. */
        double          prev_value;     /* Vrednost integrala u prethodnoj
                                         * iteraciji. */
        void           *evaluator;      /* Evaluator za izracunavanje
                                         * vrednosti funkcije. */
        int             n;      /* Broj podintervala u tekucoj iteraciji. */
        int             i;      /* Brojac u petljama. */

        /* Kreira se evaluator za izracunavanje vrednosti funkcije. */
        evaluator = evaluator_create(function);

        /* Proveravaju se ulazni podaci. */
        assert(evaluator != NULL);
```

```
        assert(a < b);
        assert(epsilon > 0);
        assert(max_subintervals >= 2);

        /* Izracunava se pocetna vrednost integrala. */
        value =
            (b - a) * (evaluator_evaluate_x(evaluator, b) +
                       evaluator_evaluate_x(evaluator, a)) / 2;

        /* U svakoj iteraciji duplira se broj podintervala i izracunava
         * vrednost integrala prema trapeznoj formuli. Pritom, za
         * izracunavanje integrala u svakoj iteraciji zaracunavaju se samo
         * vrednosti funkcije u novododatim cvorovima, da bi se na kraju u
         * vrednost integrala zaracunala i vrednost iz prethodne iteracije,
         * u kojoj su akumulirane vrednosti funkcije u zajednickim
         * cvorovima. Izracunavanje se prekida ako je postignuta zeljena
         * tacnost ili ako je prekoracen maksimalni broj podintervala. */
        for (n = 2; n <= max_subintervals; n <<= 1) {
                prev_value = value;
                value = 0;
                for (i = 1; i < n; i += 2)
                        value +=
                            2 * evaluator_evaluate_x(evaluator,
                                                     a + i * (b - a) / n);
                value *= ((b - a) / n) / 2;
                value += prev_value / 2;
                if (fabs(value - prev_value) < epsilon)
                        break;
        }

        /* Brise se evaluator. */
        evaluator_destroy(evaluator);

        /* Vraca se izracunata vrednost integrala. */
        return value;
}
```

3.8 *Ako* $f \in \mathcal{C}^1[-1,1]$ *izvesti kvadraturnu formulu*

$$\int_{-1}^{1} \sqrt{1-x^2} f(x)\,dx = \frac{\pi}{n} \sum_{k=1}^{n} f(\cos\frac{k\pi}{n}) \sin^2\frac{k\pi}{n} + R,$$

i proceniti grešku R *formule.*

Rešenje: Smenom $x = \cos t$ dati integral transformišemo u integral

$$\int_{-1}^{1} \sqrt{1-x^2} f(x)\,dx = \int_{0}^{\pi} f(\cos t)\sin^2 t\,dt,$$

koji aproksimiramo trapeznom formulom sa korakom $h = \pi/n$ i dobijamo traženu formulu. Greška kvadraturne formule R je određena greškom trapezne kvadraturne formule

$$|R| \le \frac{\pi}{12}\frac{\pi^2}{n^2} \max_{[0,\pi]} |(f(\cos x)\sin^2 x)''|.$$

3.9 *Simpson-ovom formulom izračunati sa tačnošću* $5 \cdot 10^{-6}$ *približnu vrednost integrala*

$$\int_1^2 (1 + \ln x)\, dx.$$

Rešenje: Za funkciju $f(x) = 1 + \ln x$ je $f^{(4)}(x) = -6x^{-4}$, pa se greška izračunavanja datog integrala Simpson-ovom formulom ocenjuje sa $|R| \leq 6h^4/180$. Ta greška će biti manja od dozvoljene greške ako izaberemo korak $h \leq 0.11$. Vrednost za h koja ispunjava ovaj uslov, a pogodna je za Simpson-ovu formulu je $h = 0.1$. Računajući sa ovim korakom Simpson-ovom formulom, dobijamo da je

$$\int_1^2 (1 + \ln x)\, dx = 1.38629 \pm 0.00001.$$

3.10 *Simpson-ovom formulom izračunati*

$$\int_{0.2}^{1.0} f(x)\, dx$$

i oceniti grešku rezultata, ako je funkcija $f(x)$ *zadata sledećom tabelom*

x	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
f	1.3117	1.4843	1.6075	1.7033	1.7818	1.8482	1.9058	1.9551	1.9953

Rešenje: Približna vrednost integrala izračunata Simpson-ovom formulom je

$$S_1 = 1.39492 \quad \text{za} \quad h_1 = 0.2, \qquad S_2 = 1.39536 \quad \text{za} \quad h_2 = 0.1.$$

Runge-ovim kriterijumom se dobija procena greške

$$\frac{S_2 - S_1}{15} = 3 \cdot 10^{-5}$$

te je

$$\int_{0.2}^{1.0} f(x)\, dx = 1.3954 \pm 0.00003.$$

3.11 *Neka je*

$$I(y) = \int_0^1 \arctan\left(\exp\left(x^2 + y^2 + y\right)\right) dx.$$

Naći $\min_{[0,1]} I(y)$ *sa tačnošću* 10^{-5}.

Rešenje: Tačka minimuma funkcije $I(y)$ je nula prvog izvoda ove funkcije,

$$I'(y) = (2y + 1) \int_0^1 \frac{\exp\left(x^2 + y^2 + y\right)}{1 + \exp\left(2(x^2 + y^2 + y)\right)}\, dx, \qquad I'(-0.5) = 0.$$

pa je

$$\min_{[0,1]} I(y) = I(-0.5) = \int_0^1 \arctan\left(\exp\left(x^2 - 0.25\right)\right) dx.$$

Integral ćemo izračunati Simpson-ovom formulom, a tačnost proveriti Runge-ovim kriterijumom. Tako dobijamo

$$\begin{aligned} h_1 &= 1/2 \qquad S_1 = 0.822129, \\ h_2 &= 1/4 \qquad S_2 = 0.823951, \\ h_3 &= 1/8 \qquad S_3 = 0.824039. \end{aligned}$$

Na osnovu Runge-ovog kriterijuma je tačnost postignuta, jer je $|S_3 - S_2|/15 = 6 \cdot 10^{-6}$, te je

$$\min_{[0,1]} I(y) = I(-0.5) = 0.82404 \pm 0.000006.$$

3.12 *Naći sa tačnošću* 10^{-4} *maksimum funkcije*

$$I(y) = \int_0^y \frac{\sin x}{x(2\pi - x)}\, dx, \qquad y > 0.$$

Rešenje: Kako je

$$I'(y) = \frac{\sin y}{y(2\pi - y)}, \qquad I'(\pi) = 0 \quad \text{i} \quad I''(\pi) < 0,$$

to je

$$\max_{y>0} I(y) = I(\pi) = \int_0^\pi \frac{\sin x}{x(2\pi - x)}\, dx.$$

Simpson-ovom formulom dobijaju se vrednosti

$$\begin{aligned} h_1 &= \pi/2 \qquad S_1 = 0.36627, \\ h_2 &= \pi/4 \qquad S_2 = 0.36392, \\ h_3 &= \pi/8 \qquad S_3 = 0.36379, \end{aligned}$$

te je na osnovu Runge-ovog kriterijuma $|S_3 - S_2|/15 = 10^{-5}$ tačnost postignuta, i

$$\max_{y>0} I(y) = I(\pi) = 0.3638.$$

3.13 *Dužina* U *elipse* $\{(x, y) \,|\, x^2/a^2 + y^2 = 1\}$ *izražava se formulom*

$$U = 4 \int_0^{\pi/2} \sqrt{1 - (1 - a^2) \sin^2 t}\, dt.$$

Izračunati sa tačnošću 10^{-3} *dužinu elipse čija je poluosa* $a = 2$.

Rešenje: Simpson-ovom formulom za $h = \pi/16$ uz proveru postignute tačnosti Runge-ovim kriterijumom, dobija se da je dužina elipse

$$U = 9.688.$$

3.14 *Odrediti sa tačnošću* 10^{-3} *najmanje pozitivno rešenje jednačine*

$$\int_1^x \frac{t}{\sin t}\, dt = 2.$$

Rešenje: Analizirajmo ponašanje funkcije

$$f(x) = \int_1^x \frac{t}{\sin t}\, dt - 2.$$

Kada $x \in (0, 1]$, $f(x) < 0$ jer je u tom intervalu $t/\sin t > 0$. Osnovnom Simpson-ovom formulom nalazimo da je $f(2) = -0.43$ i $f(3) = 8.42$. Kako je $f'(x) = x/\sin x > 0$ za $x \in (0, 3]$ sledi da je najmanje pozitivno rešenje x^* date jednačine u intervalu $[2, 3]$. Preciznije, Simpson-ovom formulom nalazimo sa tačnošću 10^{-4} da je $f(2.16) = -0.0527$ i $f(2.2) = 0.0539$. Interpolišemo funkciju $f(x)$ u ovom intervalu polinomom prvog stepena

$$L_1(x) = \frac{x - 2.2}{2.16 - 2.2}\, f(2.16) + \frac{x - 2.16}{2.2 - 2.16}\, f(2.2)$$

i približno rešenje jednačine $f(x) = 0$ određujemo kao rešenje jednačine $L_1(x) = 0$,

$$x^* = 2.180.$$

Greška linearne interpolacije je

$$|R_1(x)| \leq \frac{(2.2 - 2.16)^2}{8} \max_{[2.16.2.2]} |f''(\xi)| = 6.6 \cdot 10^{-4},$$

te je ovo rešenje zadovoljavajuće tačnosti.

3.15 *Sa tačnošću* $5 \cdot 10^{-4}$ *izračunati*

$$\int_0^1 \frac{\sin x}{\sqrt{x}}\, dx.$$

Rešenje: Simpson-ovom kvadraturnom formulom dobijaju se za različite vrednosti koraka h vrednosti integrala

$$\begin{aligned} h_1 &= 1/2 \qquad S_1 = 0.592252, \\ h_2 &= 1/4 \qquad S_2 = 0.610423, \\ h_3 &= 1/8 \qquad S_3 = 0.616952. \end{aligned}$$

Izračunavanje je prvo izvršeno za najveću dopuštenu vrednost koraka h, a zatim je svaki put vrednost integrala računata sa korakom koji je upola manji od prethodnog. Kako je $(S_3 - S_2)/15 = 0.435 \cdot 10^{-3}$, što je manje od dozvoljene greške, na osnovu Runge-ovog kriterijuma je približna vrednost integrala sa traženom tačnošću

$$\int_0^1 \frac{\sin x}{\sqrt{x}}\, dx = 0.617.$$

3.16 *a) Izvesti kvadraturnu formulu oblika*

$$\int_0^{2\pi} f(x) \sin x\, dx = c_0 f(0) + c_1 f(\pi) + c_2 f(2\pi) + R.$$

b) Deleći interval integracije na deset jednakih delova izračunati

$$\int_0^{2\pi} e^{x^2} \sin(10x)\, dx$$

koristeći dobijenu formulu kao i Simpson-ovu kvadraturnu formulu. Objasniti nastalu razliku.

Rešenje: a) Tri koeficijenta u formuli određujemo tako da formula bude tačna za proizvoljan polinom stepena ne većeg od dva, tj. da je

$$\int_0^{2\pi} x^k \sin x\, dx = c_0 0^k + c_1 \pi^k + c_2 (2\pi)^k, \qquad k = 0, 1, 2,$$

što daje formulu

$$\int_0^{2\pi} f(x) \sin x\, dx = f(0) - f(2\pi) + R.$$

b) Da bismo primenili izvedenu formulu na dati integral moramo ga prethodno transformisati

$$\begin{aligned}\int_0^{2\pi} e^{x^2} \sin(10x)\, dx &= \sum_{n=1}^{10} \int_{2(n-1)\pi/10}^{2n\pi/10} e^{x^2} \sin(10x)\, dx \\ &= \frac{1}{10} \sum_{n=1}^{10} \int_0^{2\pi} e^{\left(0.01(t+2(n-1)\pi)^2\right)} \sin t\, dt,\end{aligned}$$

pri čemu je korišćena smena $x = t/10 + 2(n-1)\pi/10$. Sada se primenom izvedene formule dobija da je

$$\begin{aligned}\int_0^{2\pi} e^{x^2} \sin(10x)\, dx &\approx \frac{1}{10} \sum_{n=1}^{10} \left(e^{\left(0.01(2(n-1)\pi)^2\right)} - e^{\left(0.01(2n\pi)^2\right)} \right) \\ &= \frac{1}{10} \left(1 - e^{(4\pi^2)}\right) = -1.4 \cdot 10^{16}.\end{aligned}$$

Ako se primeni Simpson-ova formula sa korakom $h = 2\pi/10$, podintegralna funkcija je u svim čvorovima kvadraturne formule jednaka nuli,

$$e^{x_k^2} \sin(10x_k) = e^{(2k\pi/10)^2} \sin(2k\pi) = 0,$$

pa je i približna vrednost integrala izračunata pomoću ove formule takođe jednaka nuli.

3.2 Kvadraturne formule Gauss-ovog tipa

Za razliku od Newton–Cotes-ovih kvadraturnih formula, kod ovih formula

$$S = \frac{b-a}{2} \sum_{k=1}^{n} c_k f(x_k)$$

čvorovi x_k, $k = 1, \ldots, n$, nisu unapred poznati. Oni su nule polinoma $Q_n(x)$ koji pripada skupu polinoma ortogonalnih u odnosu na težinsku funkciju $p(x)$,

$$(Q_n, Q_m) = \int_a^b p(x) Q_n(x) Q_m(x)\, dx = 0, \qquad n \neq m$$

Ortogonalni polinomi su određeni rekurentnom formulom

$$Q_0(x) \equiv 1$$
$$Q_{k+1}(x) = \left(x - \frac{(xQ_k, Q_k)}{(Q_k, Q_k)} \right) Q_k(x) - \frac{(Q_k, Q_k)}{(Q_{k-1}, Q_{k-1})} Q_{k-1}(x), \quad k = 0, 1, \ldots,$$

pri čemu je drugi sabirak nula za $k = 0$. Često korišćeni polinomi ortogonalni na intervalu $[-1, 1]$ su

$$L_n(x) = \frac{1}{2^n\, n!} \frac{d^n}{dx^n}\big((x^2-1)^n\big), \qquad p(x) \equiv 1, \qquad \text{Legendre-ov polinom}$$

$$T_n(x) = \cos(n \arccos x), \qquad p(x) = \frac{1}{\sqrt{1-x^2}}, \qquad \text{Čebišev-ljev polinom}$$

Koeficijenti kvadraturne formule c_k određeni su rešenjem sistema linearnih jednačina

$$\frac{b-a}{2} \sum_{k=1}^{n} c_k x_k^j = \int_a^b p(x) x^j\, dx, \qquad j = 0, \ldots, n-1.$$

Greška kvadraturnih formula Gauss-ovog tipa R ocenjuje se izrazom

$$|R| \leq \frac{1}{(2n)!} \max_{[a,b]} |f^{(2n)}(\xi)| \int_a^b p(x) Q_n^2(x)\, dx$$

3.17 *Sa tačnošću $\varepsilon = 10^{-2}$ izračunati*

$$\int_0^1 \frac{\cos x}{\sqrt{x}}\,dx.$$

Rešenje: Za razliku od prethodnog zadatka Simpson-ova kvadraturna formula se ne može direktno primeniti, jer podintegralna funkcija ima singularitet u levom kraju intervala integracije. Da bi se ova formula primenila, potrebno je podeliti interval integracije na dva dela

$$\int_0^1 \frac{\cos x}{\sqrt{x}}\,dx = \int_0^\delta \frac{\cos x}{\sqrt{x}}\,dx + \int_\delta^1 \frac{\cos x}{\sqrt{x}}\,dx.$$

Pošto je dati integral konvergentan, δ se može izabrati dovoljno malo da vrednost prvog integrala bude manja od $\varepsilon/2$,

$$\int_0^\delta \frac{\cos x}{\sqrt{x}}\,dx \le \int_0^\delta \frac{1}{\sqrt{x}}\,dx = 2\sqrt{\delta} \le \frac{1}{2}\varepsilon \qquad \longrightarrow \qquad \delta \le \frac{\varepsilon^2}{16}.$$

Sada se Simpson-ovom ili nekom drugom formulom Newton–Cotes-ovog tipa drugi integral može izračunati sa tačnošću $\varepsilon/2$.

Formulama Newton–Cotes-ovog tipa se ovaj integral može izračunati i ako se prethodno transformiše smenom $x = t^2$ u regularni integral $\int_0^1 2\cos t^2\,dt$.

Treći način da se izračuna dati integral je da se primeni kvadraturna formula Gauss-ovog tipa sa težinskom funkcijom $x^{-1/2}$. Ta formula je

$$\int_0^1 \frac{\cos x}{\sqrt{x}}\,dx = 1.3043 f(0.1156) + 0.6957 f(0.7416) + R,$$

$$|R| \le 5 \cdot 10^{-4} \max_{[0,1]} |f^{(4)}(x)|.$$

Primenom ma kog od pomenutih algoritama, sa traženom tačnošću dobija se da je

$$\int_0^1 \frac{\cos x}{\sqrt{x}}\,dx = 1.81.$$

3.18 *Gauss-ovom formulom sa tri čvora izračunati približnu vrednosti integrala*

$$\int_{-1}^1 \cosh x \cos x\,dx$$

i oceniti grešku rezultata.

Rešenje: Čvorovi tražene kvadraturne formule su nule Legendre-ovog polinoma trećeg stepena, jer su ovi polinomi ortogonalni na intervalu $[-1, 1]$ sa težinskom funkcijom $p(x) = 1$. Stoga je Gauss-ova kvadraturna formula sa tri čvora

$$\int_{-1}^{1} f(x)\,dx = \frac{1}{9}\left(5f\big(-\sqrt{\frac{3}{5}}\big) + 8f(0) + 5f\big(\sqrt{\frac{3}{5}}\big)\right) + R.$$

Greška je

$$R = \frac{1}{6!}\int_{-1}^{1} f^{(6)}(\xi)x^2(x^2 - \frac{3}{5})^2\,dx,$$

i može se oceniti sa

$$|R| \le \frac{1}{6!}\max_{[-1,1]}|f^{(6)}(x)|\int_{-1}^{1} x^2(x^2 - \frac{3}{5})^2\,dx = \frac{M_6}{15750},$$

gde je $M_6 = \max_{[-1,1]}|f^{(6)}(x)|$. Kako je u ovom zadatku

$$M_6 = \max_{[-1,1]}|(\cosh x\cos x)^{(6)}| = \max_{[-1,1]}|8\sinh x\sin x| = 8\sinh 1\sin 1 = 7.94,$$

primenom navedenih formula dobijamo da je

$$\int_{-1}^{1}\cosh x\cos x\,dx = 1.9334 \pm 0.0005.$$

3.19 *Gauss-ovom formulom sa tri čvora izračunati*

$$\int_0^1 \exp\big(\cos(2\pi x)\big)\,dx.$$

Računati na tri decimale.

Rešenje: Da bismo primenili formulu iz prethodnog zadatka, prethodno ćemo transformisati interval integracije $(0,1)$ u interval $(-1,1)$ smenom $x = (1+t)/2$,

$$\int_0^1 f(x)\,dx = \frac{1}{2}\int_{-1}^{1} f\big(\frac{1+t}{2}\big)\,dt \approx 1.351$$

3.20 *Napisati formulu Gauss-ovog tipa za izračunavanje približne vrednosti integrala*

$$\int_0^\infty f(x)\cos x\,dx.$$

Rešenje: Integral predstavimo zbirom

$$\int_0^\infty f(x)\cos x\,dx = \sum_{k=0}^{\infty}\int_{k\pi}^{(k+1)\pi} f(x)\cos x\,dx.$$

Smenom $x = \pi(t+2k+1)/2$ svaki od ovih integrala se svodi na integral na intervalu $[-1, 1]$,

$$\int_{k\pi}^{(k+1)\pi} f(x)\cos x\,dx = (-1)^{k-1}\frac{\pi}{2}\int_{-1}^{1} f\big((t+2k+1)\frac{\pi}{2}\big)\sin\frac{\pi t}{2}\,dt.$$

Polazni integral se onda može aproksimirati parcijalnom sumom reda

$$\int_0^{\infty} f(x)\cos x\,dx = \frac{\pi}{2}\sum_{k=0}^{\infty}(-1)^{k-1}\int_{-1}^{1} f\big((t+2k+1)\frac{\pi}{2}\big)\sin\frac{\pi t}{2}\,dt,$$

čiji je svaki sabirak približno izračunat npr. formulom Gauss-ovog tipa sa težinskom funkcijom $\sin\frac{\pi t}{2}$.

3.21 *Izvesti kvadraturnu formulu*

$$\int_0^1 \frac{f(x)}{\sqrt{x(1-x)}}\,dx = \frac{\pi}{n}\sum_{k=1}^{n} f\big(\cos^2\frac{(2k-1)\pi}{4n}\big) + R,$$

i oceniti grešku R.

Rešenje: Smenom $x = (1+t)/2$ interval $[0, 1]$ preslikavamo u interval $[-1, 1]$ i integral postaje

$$\int_0^1 \frac{f(x)}{\sqrt{x(1-x)}}\,dx = \int_{-1}^{1} \frac{f\big((1+t)/2\big)}{\sqrt{1-t^2}}\,dt.$$

S obzirom da se traži kvadraturna formula za funkciju $f(x)$, funkcija $1/\sqrt{1-t^2}$ je težinska funkcija. Na intervalu $[-1, 1]$ u odnosu na ovu težinsku funkciju ortogonalni su Čebišev-ljevi polinomi. Stoga je tražena formula kvadraturna formula Gauss-ovog tipa sa čvorovima koji su nule Čebišev-ljevog polinoma $T_n(t)$ i naziva se Čebišev-ljeva kvadraturna formula. Ona je oblika

$$\int_{-1}^{1} \frac{f(t)}{\sqrt{1-t^2}}\,dt = \frac{\pi}{n}\sum_{k=1}^{n} f\big(\cos\frac{(2k-1)\pi}{2n}\big) + R.$$

Primenjujući ovu formulu za odgovarajući argument funkcije f konačno dobijamo da je

$$\begin{aligned}\int_0^1 \frac{f(x)}{\sqrt{x(1-x)}}\,dx &= \frac{\pi}{n}\sum_{k=1}^{n} f\left(\frac{1}{2}(1+\cos\frac{(2k-1)\pi}{2n})\right) + R\\ &= \frac{\pi}{n}\sum_{k=1}^{n} f\big(\cos^2\frac{(2k-1)\pi}{4n}\big) + R\end{aligned}$$

Greška R je određena izrazom za grešku kvadraturnih formula Gauss-ovog tipa

$$|R| \le \frac{1}{(2n)!} \max_{[a,b]} |f^{(2n)}(x)| \int_a^b \frac{\omega_n^2(t)}{\sqrt{1-t^2}}\, dt$$

U slučaju Čebišev-ljeve kvadraturne formule je

$$\omega_n(t) = \overline{T}_n(t) = 2^{1-n} T_n(t)$$

pa je zbog toga što je

$$(T_n, T_n) = \int_{-1}^{1} \frac{T_n^2(t)}{\sqrt{1-t^2}}\, dt = \frac{\pi}{2}, \qquad (n > 0)$$

tražena ocena greške

$$|R| \le \frac{\pi}{2^{2n-1}(2n)!} \max_{[0,1]} |f^{(2n)}(x)|.$$

3.22 *Sa tačnošću* $2 \cdot 10^{-2}$ *izračunati približnu vrednost integrala*

$$\int_{-1}^{1} \frac{\cos x}{\sqrt{1-x^2}}\, dx.$$

Rešenje: Integral ćemo izračunati Čebišev-ljevom kvadraturnom formulom jer su Čebišev-ljevi polinomi ortogonalni na intervalu $[-1, 1]$ u odnosu na težinsku funkciju $1/\sqrt{1-x^2}$ (vidi prethodni zadatak). Pokušajmo sa formulom sa dva čvora, $n = 2$. Greška te formule se ocenjuje izrazom

$$|R| \le \frac{\pi}{192} \max_{[-1,1]} |f^{(4)}(x)|,$$

što u ovom zadatku, s obzirom da je $\max_{[-1,1]} |(\cos x)^{(4)}| \le 1$, daje $|R| \le 0.016$. Dakle, sa zahtevanom tačnošću je

$$\int_{-1}^{1} \frac{\cos x}{\sqrt{1-x^2}}\, dx = \frac{\pi}{2}\left(\cos\frac{-1}{\sqrt{2}} + \cos\frac{1}{\sqrt{2}}\right) + R = 2.39 \pm 0.02.$$

3.23 *Korišćenjem numeričke integracije dokazati da je*

$$\int_0^1 \frac{\sin x}{\sqrt{1-x^2}}\, dx < \int_0^1 \frac{\cos x}{\sqrt{1-x^2}}\, dx.$$

Rešenje: Čebišev-ljevom kvadraturnom formulom za $n = 3$ sa tačnošću $|R| \le 1.4 \cdot 10^{-4}$ izračunati su integrali

$$\int_{-1}^{1} \frac{|\sin x|}{\sqrt{1-x^2}}\, dx = 2 \cdot 0.7977 + R, \qquad \int_{-1}^{1} \frac{\cos x}{\sqrt{1-x^2}}\, dx = 2 \cdot 1.2020 + R,$$

odakle neposredno sledi tvrđenje.

C Implementacija metode integracije Gauss-ovim kvadraturnim formulama sa nulama u korenima odgovarajućeg Čebišev-ljevog polinoma prosto sledi odgovarajuće formule. Za nule Čebišev-ljevog polinoma korišćen je direktni izraz, bez optimizacija pri izračunavanju kosinusa oblika $\cos\frac{(2i+1)\pi}{2n}$ o kojima bi eventualno moglo da se razmisli.

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <matheval.h>

/*
 * Funkcija chebyshev() izracunava integral na intervalu [-1,1] sa
 * podintegralnom funkcijom oblika f(x)/sqrt(1-x^2) koriscenjem
 * kvadraturne formule Gauss-ovog tipa sa cvorovima koji odgovaraju nulama
 * odgovarajuceg Cebisevljevog polinoma. Argumenti funkcije su:
 *   function - string koji predstavlja funkciju f(x)
 *   n - broj cvorova kvadraturne formule
 * Funkcija vraca izracunatu vrednost integrala. Argumenti funkcije
 * moraju da zadovolje uslov da je funkcija f(x) sintaksno ispravno
 * zadata (za nacin zadavanja funkcije videti dokumentaciju libmatheval
 * biblioteke), te da broj cvorova bude pozitivan.
 */
double
chebyshev(char *function, int n)
{
        const double    pi = 3.14159265358979323846;    /* Broj pi. */
        double          value;  /* Rezultat izracunavanja. */
        void           *evaluator;      /* Evaluator za izracunavanje
                                         * vrednosti funkcije. */
        int             i;      /* Brojac u petljama. */

        /* Kreira se evaluator za izracunavanje vrednosti funkcije. */
        evaluator = evaluator_create(function);

        /* Proveravaju se ulazni podaci. */
        assert(evaluator != NULL);
        assert(n > 0);

        /* Izracunava se vrednost integrala. */
        value = 0;
        for (i = 0; i < n; i++)
                value +=
                    evaluator_evaluate_x(evaluator,
                                         cos((2 * i + 1) * pi / (2 * n)));
        value *= pi / n;

        /* Brise se evaluator. */
        evaluator_destroy(evaluator);

        /* Vraca se izracunata vrednost integrala. */
        return value;
}
```

3.24 *Neka je* $|f^{(k)}(x)| \leq 1$, $k = 1, 2, \ldots$. *Funkciju* $f(x)$ *možemo izračunati najviše u tri tačke. Naći formulu pomoću koje možemo sa tačnošću* $1.5 \cdot 10^{-4}$ *izračunati*

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\, dx.$$

Rešenje: S obzirom na oblik težinske funkcije $p(x) = (1-x^2)^{-1/2}$ koristićemo Čebišev-ljevu kvadraturnu formulu

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\, dt = \frac{\pi}{3}\left(f\big(-\frac{\sqrt{3}}{2}\big) + f(0) + f\big(\frac{\sqrt{3}}{2}\big)\right) + R.$$

Kako su čvorovi formule nule Čebišev-ljevog polinoma $T_3(x)$, greška formule je (zadatak 21.)

$$|R| \leq \frac{1}{6!} \max_{[-1,1]} |f^{(6)}(x)| \int_{-1}^{1} \frac{2^{-4} T_3^2(x)}{\sqrt{1-x^2}}\, dx = \frac{1}{720}\,\frac{\pi}{32} = 1.4 \cdot 10^{-4},$$

što zadovoljava postavljeni uslov.

3.25 *Pokazati da formula Gauss-ovog tipa*

$$\int_0^{\pi} f(x)\sin(nx)\, dx = c_1 f(x_1) + c_2 f(x_2) + R,$$

gde je n *prirodan broj, ima parametre*

$$x_{1/2} = \frac{\pi}{2} \mp \sqrt{\frac{\pi^2}{4} - \frac{6}{n^2}}, \qquad c_1 = -c_2 = \frac{\pi}{n\sqrt{\pi^2 - 24/n^2}}, \qquad \textit{za } n \textit{ parno},$$

$$x_{1/2} = \frac{\pi}{2} \mp \sqrt{\frac{\pi^2}{4} - \frac{2}{n^2}}, \qquad c_1 = c_2 = \frac{1}{n} \qquad \textit{za } n \textit{ neparno}.$$

Takođe pokazati da je red greške formule četiri kada je n *parno, a tri kada je* n *neparno.*

Rešenje: Čvorovi kvadraturne formule su nule polinoma $\omega(x) = x^2 + ax + b$ koji zadovoljava uslove

$$\int_0^{\pi} \omega(x) x^k \sin(nx)\, dx = 0, \qquad k = 0, 1.$$

Parcijalnom integracijom nalazimo da je

$$\int_0^\pi \sin(nx)\,dx = \frac{1-(-1)^n}{n},$$
$$\int_0^\pi x\sin(nx)\,dx = \frac{(-1)^{n+1}\pi}{n},$$
$$\int_0^\pi x^2\sin(nx)\,dx = \frac{1}{n^3}\left((-1)^{n+1}n^2\pi^2 + 2(-1)^n - 2\right),$$
$$\int_0^\pi x^3\sin(nx)\,dx = \frac{(-1)^{n+1}\pi}{n^3}\big(n^2\pi^2 - 6\big),$$

te su koeficijenti polinoma $\omega(x)$ rešenja sistema linearnih jednačina

n parno

$$a\frac{\pi}{n} = -\frac{\pi^2}{n}$$
$$a\frac{\pi^2}{n} + b\frac{\pi}{n} = \frac{\pi}{n^3}(6 - n^2\pi^2)$$

n neparno

$$a\frac{\pi}{n} + b\frac{2}{n} = \frac{1}{n^3}(4 - n^2\pi^2)$$
$$\frac{a}{n^3}(n^2\pi^2 - 4) + b\frac{\pi}{n} = \frac{\pi}{n^3}(6 - n^2\pi^2)$$

Stoga su polinom $\omega(x)$ i njegove nule

$$\omega(x) = x^2 - \pi x + \frac{6}{n^2} \quad \text{i} \quad x_{1/2} = \frac{\pi}{2} \mp \sqrt{\frac{\pi^2}{4} - \frac{6}{n^2}}, \quad \text{za } n \text{ parno},$$

$$\omega(x) = x^2 - \pi x + \frac{2}{n^2} \quad \text{i} \quad x_{1/2} = \frac{\pi}{2} \mp \sqrt{\frac{\pi^2}{4} - \frac{2}{n^2}}, \quad \text{za } n \text{ neparno}.$$

Dakle, pokazano je da su čvorovi kvadraturne formule upravo navedenog oblika. Još treba odrediti koeficijente c_1 i c_2. Iz uslova da formula važi tačno za funkcije $f(x) = 1$ i $f(x) = x$ dobijamo sistem linearnih jednačina po koeficijentima c_i,

$$\begin{aligned} c_1 + c_2 &= 0 \\ c_1x_1 + c_2x_2 &= -\frac{\pi}{n} \end{aligned} \quad \text{za } n \text{ parno}, \qquad \begin{aligned} c_1 + c_2 &= \frac{2}{n} \\ c_1x_1 + c_2x_2 &= \frac{\pi}{n} \end{aligned} \quad \text{za } n \text{ neparno},$$

čija rešenja su upravo data izrazima datim formulacijom zadatka.

Još je ostalo da analiziramo red greške kvadraturne formule. To je formula Gauss-ovog tipa sa dva čvora pa je tačna za sve polinome najviše stepena tri. Ako se formula primeni na funkciju $f(x) = x^4$, dobija se da je

$$R = \int_0^\pi x^4\sin(nx)\,dx - c_1x_1^4 - c_2x_2^4$$
$$= \begin{cases} 0, & n \text{ parno} \\ (-1)^{n+1}\pi(n^4\pi^4 - 20n^2\pi^2 + 120)/n^5, & n \text{ neparno} \end{cases}$$

Dakle, $R = 0$ kada je n parno i $R \neq 0$ kada je n neparno. Za $f(x) = x^5$ je $R \neq 0$ i za n parno. Stoga je red greške formule četiri kada je n parno i tri kada je n neparno.

3.26 *a) Ako je poznato da je*

$$\int_0^{\pi/2} \frac{\sin x}{x}\, dx = 1.37076,$$

izvesti formulu za približnu integraciju Gaussovog tipa

$$\int_0^{\pi/2} \frac{\sin x}{x} f(x)\, dx = c_1 f(x_1) + c_2 f(x_2) + R.$$

b) Proceniti grešku R dobijene formule.
c) Izračunati pomoću te formule

$$\int_0^{\pi} \frac{\sin x}{x}\, dx,$$

i proceniti grešku.

Rešenje: a) Čvorovi kvadraturne formule su nule polinoma $\omega_2(x) = x^2 + ax + b$ koji pripada skupu polinoma ortogonalnih na intervalu integracije u odnosu na težinsku funkciju $\sin x/x$, tj. koji zadovoljava uslove

$$\int_0^{\pi/2} \frac{\sin x}{x} \omega_2(x) x^k\, dx = 0, \qquad k = 0, 1.$$

Ovim uslovima je određen sistem linearnih jednačina po koeficijentima a i b polinoma $\omega_2(x)$, čije rešenje je $a = -1.52349$, $b = 0.38190$. Sada su čvorovi kvadraturne formule, a to su nule ovoga polinoma, $x_1 = 0.31637$, $x_2 = 1.20712$. Koeficijente formule određujemo iz uslova da formula važi tačno za funkcije $f(x) = 1$ i $f(x) = x$,

$$\int_0^{\pi/2} \frac{\sin x}{x} x^k\, dx = c_1 x_1^k + c_2 x_2^k, \qquad k = 0, 1,$$

što određuje sistem linearnih jednačina po c_1 i c_2. Njegovo rešenje je

$$c_1 = 0.73497, \qquad c_2 = 0.63579,$$

pa je tražena kvadraturna formula

$$\int_0^{\pi/2} \frac{\sin x}{x} f(x)\, dx = 0.73497 f(0.31637) + 0.63579 f(1.20712) + R.$$

b) Pošto je to formula Gauss-ovog tipa, greška se ocenjuje izrazom

$$|R| \leq \frac{1}{4!} \max_{[0,\pi/2]} |f^{(4)}(x)| \int_0^{\pi/2} \frac{\sin x}{x} \omega_2^2(x)\, dx = 2 \cdot 10^{-3} \max_{[0,\pi/2]} |f^{(4)}(x)|.$$

c) Transformišimo dati integral u integral na koji se može primeniti izvedena kvadraturna formula,

$$\int_0^{\pi} \frac{\sin x}{x}\,dx = \int_0^{\pi/2} \frac{\sin(2t)}{2t} 2\,dt = 2\int_0^{\pi/2} \frac{\sin t}{t}\cos t\,dt.$$

Dakle, kvadraturna formula se primenjuje na funkciju $f(x) = \cos x$, pa je

$$\int_0^{\pi} \frac{\sin x}{x}\,dx = 2\big(0.73497\cos(0.31637) + 0.63579\cos(1.20712)\big) + R,$$

gde se greška R prema izvedenoj formuli ocenjuje sa $|R| \le 2\cdot 10^{-3}$. Dakle,

$$\int_0^{\pi} \frac{\sin x}{x}\,dx = 1.849 + R, \qquad |R| \le 2\cdot 10^{-3}.$$

3.27 *Izvesti kvadraturnu formulu oblika*

$$\int_0^{\infty} e^{-x} f(x)\,dx = c_1 f(x_1) + c_2 f(x_2) + R$$

i oceniti red greške formule.

Rešenje: Na intervalu $[0,\infty]$ sa težinskom funkcijom e^{-x} ortogonalni su Laguerre-ovi polinomi. Stoga je tražena kvadraturna formula Gauss-ovog tipa sa čvorovima koji su nule Laguerre-ovog polinoma drugog stepena

$$L_2(x) = 0.5x^2 - 2x + 1, \qquad x_{1/2} = 2 \mp \sqrt{2}.$$

Koeficijente c_1 i c_2 određujemo iz uslova da je formula tačna za fukcije $f(x) = x^k$, $k = 0, 1$. Tako se dobija da je $c_{1/2} = (2 \pm \sqrt{2})/4$, pa je tražena formula

$$\int_0^{\infty} e^{-x} f(x)\,dx = \frac{2+\sqrt{2}}{4} f\big(2-\sqrt{2}\big) + \frac{2-\sqrt{2}}{4} f\big(2+\sqrt{2}\big) + R.$$

Formula je Gauss-ovog tipa te je tačna za polinome stepena manjeg ili jednakog tri.

3.28 *Na intervalu* $[0,1]$ *integrali oblika*

$$\int_0^1 f(x)\sqrt{\frac{x}{a}}\,dx, \qquad a > 0,$$

se mogu izračunati približno po formuli

$$c_0 f(0) + c_1 f(1) + f(\xi), \qquad \xi \in (0,1).$$

Za koje vrednosti parametra a *je moguće primeniti ovu formulu tako da ona bude tačna za polinome drugog stepena. Izračunati* c_0, c_1 *i* ξ *u funkciji od* a.

Rešenje: Iz uslova

$$\int_0^1 x^k \sqrt{\frac{x}{a}}\, dx = c_0 0^k + c_1 + \xi^k, \qquad k = 0, 1, 2,$$

dobijamo sistem jednačina po parametrima formule

$$c_0 + c_1 = \frac{2}{3\sqrt{a}} - 1, \qquad c_1 + \xi = \frac{2}{5\sqrt{a}}, \qquad c_1 + \xi^2 = \frac{2}{7\sqrt{a}}.$$

Njegovo rešenje je

$$\xi_{1/2} = \frac{1}{2} \pm \sqrt{A}, \qquad c_0 = \frac{4}{15\sqrt{a}} - \frac{1}{2} \pm \sqrt{A}, \qquad c_1 = \frac{2}{5\sqrt{a}} - \frac{1}{2} \mp \sqrt{A}$$

i definisano je za

$$A = \frac{1}{4} - \frac{4}{35\sqrt{a}} \geq 0 \qquad \text{tj.} \qquad a \geq 0.21.$$

3.29 *Odrediti prirodan broj n i ostale parametre kvadraturne formule*

$$\int_0^1 f(x)\, dx = c_0 f(x_0) + c_1 f^{(n)}(x_0) + R$$

tako da ona bude tačna za sve polinome stepena ne većeg od dva.

Rešenje: Postavimo uslove da formula važi tačno za bazisne polinome x^k, $k = 0, 1, 2$, koji generišu proizvoljan polinom stepena ne većeg od dva,

$$\int_0^1 x^k\, dx = c_0 x_0^k + c_1 k(k-1)\ldots(k-n+1)x_0^{k-n}, \qquad k = 0, 1, 2.$$

Za $\underline{n = 0}$ dobija se sistem

$$c_0 + c_1 = 0, \qquad c_0 x_0 + c_1 x_0 = 1/2, \qquad c_0 x_0^2 + c_1 x_0^2 = 1/3,$$

koji nema rešenje.

Za $\underline{n = 1}$ sistem jednačina po parametrima je

$$c_0 = 1, \qquad c_0 x_0 + c_1 = 1/2, \qquad c_0 x_0^2 + 2c_1 x_0 = 1/3,$$

koji takođe nema rešenje.

Za $\underline{n = 2}$ sistem jednačina po parametrima je

$$c_0 = 1, \qquad c_0 x_0 = 1/2, \qquad c_0 x_0^2 + 2c_1 = 1/3,$$

čije rešenje je $c_0 = 1$, $c_1 = 1/24$, $x_0 = 1/2$.

Za $\underline{n > 2}$ sistem jednačina po parametrima je

$$c_0 = 1, \qquad c_0 x_0 = 1/2, \qquad c_0 x_0^2 = 1/3,$$

i on nema rešenja.

Dakle, tražena kvadraturna formula je

$$\int_0^1 f(x)\,dx = f\big(\frac{1}{2}\big) + \frac{1}{24} f''\big(\frac{1}{2}\big) + R.$$

Red greške formule je 3 jer je $R(x^3) = 0$, a $R(x^4) \neq 0$.

3.30 *Ako $f \in \mathcal{C}^2[-1,1]$, odrediti koeficijente kvadraturne formule*

$$\int_{-1}^1 f(x)\,dx = f(-1) + f(1) + c_1\big(f'(-1) + f'(1)\big) + c_2\big(f''(-1) + f''(1)\big) + R.$$

Rešenje: Postavljamo uslov da formula važi tačno za polinome što je moguće višeg stepena. Za svaku vrednost koeficijenata c_1 i c_2 ona je tačna za polinome $f(x) = x^k$, $k = 0, 1, 3, 5$, a tačna je i za polinome $f(x) = x^k$, $k = 2, 4$, ako je $c_1 = 3/5$ i $c_2 = 4/15$. Dakle, tražena kvadraturna formula ima red greške pet i oblika je

$$\int_{-1}^1 f(x)\,dx = f(-1) + f(1) + \frac{3}{5}\big(f'(-1) + f'(1)\big) + \frac{4}{15}\big(f''(-1) + f''(1)\big) + R.$$

MATLAB

Simpson-ova integracija :	`i = quad(f, a, b, tol)`
Trapezna integracija :	`i = trapz(x, y)`
Gauss-Lobatto-va integracija :	`i = quadl(f, a, b, tol)`
Simbolička integracija :	`i = int(f, a, b)`
Dvodimenziona integracija :	`i = dblquad(f, xmin, xmax, ymin, ymax, tol)`

MATLAB poseduje dve osnovne funkcije koje vrše numeričko izračunavanje određenog integrala $\int_a^b f(x)dx$, `quad` i `quadl`. Obe funkcije zahtevaju da granice a i b budu konačne i da integral nema singulariteta na intervalu $[a, b]$.

Funkcija `quad` integral računa adaptivnom primenom Simpsonove kvadraturne formule koja je tačna za polinome do trećeg stepena. Osnovni oblik upotrebe ove funkcije je `q = quad(f,a,b,tol)`, gde je `f` funkcija koja se integrali, `a` i `b` su granice itegracije, dok je `tol` apsolutna greška sa kojom seintegral računa. Ukoliko se parametar `tol` izostavi, podrazumeva se vrednost 10^{-6}. Podintegralnu funkciju `fun` je moguće zadati na uobičajene načine (imenom m-datoteke, analitičkim izrazom, inline objektom ili zadavanjem function-handle-a). Od funkcije `f` se očekuje da bude vektorizovana tj. da joj argument, osim skalara, može biti i vektor u kom

slučaju se vrednosti funkcije `f` u njegovim tačkama izračunavaju pokoordinatno. Za vektorizaciju, moguće je koristiti pomoćnu funkciju `vectorize` koja priprema funkciju zadatu analitičkim izrazom za ovakav tip izračunavanja.

Integracija trapeznom kvadraturnom formulom se vrši korišćenjem funkcije `q = trapz(x, y)` pri čemu su `x` i `y` vektori koji predstavljaju tabelirane vrednosti podintegralne funkcije f.

Funkcija `qaudl` koristi Lobatto-vu formulu Gauss-ovog tipa koja dobre rezultate daje kod glatkih podintegralnih funkcija i velikih tačnosti izračunavanja. Način korišćenja funkcije `quadl` je identičan korišćenju funkcije `quad`.

Podsetimo da, pored pomenutih funkcija za numeričku integraciju, MATLAB, odnosno modul za simbolička izračunavanja (symbolic toolbox), poseduje i funkciju `int` za analitičko izračunavanje integrala.

3.31 *Fresnel-ova spirala je data preko parametarskih jednačina*

$$x(t) = \int_0^t \cos(u^2)du, \quad y(t) = \int_0^t \sin(u^2)du, \quad t \in [-4\pi, 4\pi]$$

Korišćenjem MATLAB*-a skicirati Fresnel-ovu spiralu.*

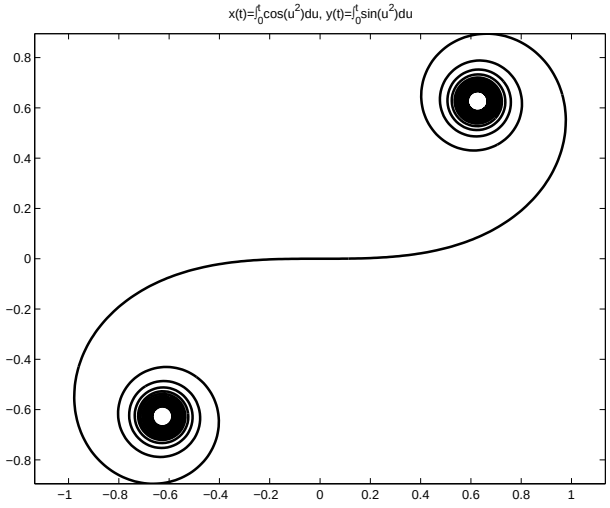
Rešenje: Kako bismo smanjili račun, primetimo da su obe koordinatne funkcije $x(t)$ i $y(t)$ neparne, tako da ćemo njihove vrednosti računati samo na ekvidistantnoj mreži t_i itervala $[0, 4\pi]$. Dalje, primetimo da je

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} \cos(u^2)du, \quad y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} \sin(u^2)du$$

Ovo znači da je dovoljno izračunati Fresnel-ove integrale na svim intervalima oblika $[t_i, t_{i+1}]$, a zatim funkcije $x(t)$ i $y(t)$ izračunati kumulativnom sumacijom dobijenih vrednosti.

```
% Mreza tacaka u kojima se racuna vrednost koordinatnih funkcija
t = linspace(0, 4*pi, n);

% Prealokacija
x = zeros(1, n); y = zeros(1, n);

% Izracunavamo Fresnel-ove integrale na [ti, ti+1]
for i = 1:n-1
    x(i) = quadl('cos(x.^2)', t(i), t(i+1), 1e-3);
    y(i) = quadl('sin(x.^2)', t(i), t(i+1), 1e-3);
end

% Kumulativna sumacija
x = cumsum(x); y = cumsum(y);

% Iscrtavanje uz koriscenje neparnosti
plot([-x(end:-1:1) 0 x],[-y(end:-1:1) 0 y]), axis equal
```

Rezultat rada skripta je prikazan na slici 3.2.



Slika 3.1: Fresnel-ova spirala

Dvostruki integral $\int_{ymin}^{ymax}\int_{xmin}^{xmax} f(x,y)dxdy$ je moguće proceniti korišćenjem funkcije `q = dblquad(f, xmin, xmax, ymin, ymax, tol)`. Podintegralna funkcija `f` je funkcija dve promenljive kojoj se prosleđuju vektor `x` i skalar `y` i koja izračunava vektor vrednosti funkcije u tačkama (x_i, y). Funkcija `dblquad` se zasniva na uzastopnom primenjivanju funkciju `quad`.

3.32 *Koristeći* MATLAB *izračunati*

$$\int_4^6 \int_0^1 (y^2 e^x + x\cos y)dxdy$$

Rešenje:

```
dblquad('y^2*exp(x)+x*cos(y)', 0, 1, 4, 6)
```

Iako MATLAB direktno ne podržava rad sa ortogonalnim polinomima, njihova konstrukcija se jednostavno implementira.

3.33 *Napisati* MATLAB *funkciju* `legendre_poly(n, a, b)` *koja izračunava sve ortogonalne Legendre-ove polinome do stepena* n *na intervalu* $[a, b]$.

Pošto je u primenama često potrebno koristiti sve polinome do određenog stepena, mnogo je efikasnije napraviti funkciju koja izračunava sve polinome do datog stepena, nego funkciju koja računa jedan polinom datog stepena. Razlog ovome je

činjenica da se Legendre-ovi polinomi na intervalu $[-1, 1]$ izračunavaju korišćenjem rekurentne formule:

$$L_0(x) = 1, \quad L_1(x) = x, \quad L_i(x) = \frac{(2i-1)x}{i} L_{i-1}(x) - \frac{i-1}{i} L_{i-2}(x), \quad i = 2, \ldots, n$$

Ovo znači da je neophodno konstruisati sve polinome stepena manjeg od n da bi se konstruisao polinom L_n stepena n. Kolekciju Legendre-ovih polinoma čuvamo u okviru cell array-a koji funkcija `legendre_poly` vraća. Pošto indeksiranje u MATLAB-u počinje od 1 polinom L_i će biti predstavljen sa `L{i+1}`.

Polinomi koji su ortogonalni na proizvoljnom intervalu $[a, b]$ se mogu dobiti od polinoma ortogonalnih na $[-1, 1]$ uvođenjem smene $t = \frac{2x-b-a}{b-a}$. Ovo se svodi na izračunavanje vrednosti polinoma L_i u tački t, što se najefikasnije izvodi Hornerovom shemom.

```
function L = legendre_poly(n, a, b)
% LEGENDRE_POLY  - racuna sve Lezandrove polinome do stepena n
% Polinomi imaju osobinu da je Ln(1)=1
% Funkcija vraca cell array L, tako da je L{i}=Li-1

L{1} = 1;        % L0 = 1
L{2} = [1 0];    % L1 = x

% Ostali polinomi se racunaju na osnovu rekurentne formule
%  Li(x)=(2i-1)/i x Li-1(x) - (i-1)/i Li-2(x)
for i=2:n
    L{i+1}=((2*i-1)/i)*[L{i} 0]-[0 0 (i-1)/i*L{i-1}];
end

% U slucaju da je naveden i interval razlicit od podrazumevanog [-1 1]
% vrsimo preslikavanje polinoma na dati interval [a, b]
if (nargin == 3)
    for i = 2:n+1
         L{i} = map_to_interval(L{i}, a, b);
    end
end

% Pomocna funkcija koja preslikava polinom P sa [-1, 1] na [a, b]
% uvodjenjem smene t = (2x-b-a)/(b-a)
% Hornerovom shemom se izracunava vrednost polinoma P u tacki t
function Pab = map_to_interval(P, a, b)
Pab = P(1);
for j=2:length(P)
    Pab = (2*[Pab 0]-(b+a)*[0 Pab])/(b-a);
    Pab(end) = Pab(end) + P(j);
end
```

Naglasimo da se na sličan način mogu dobiti i Čebiševljevi polinomi koji su ortogonalni u odnosu na težinsku funkciju $\frac{1}{\sqrt{1-x^2}}$. U ovom slučaju, primenjuje se rekurentna formula

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_i(x) = 2x\,T_{i-1}(x) - T_{i-2}(x), \quad i = 2, \ldots, n$$

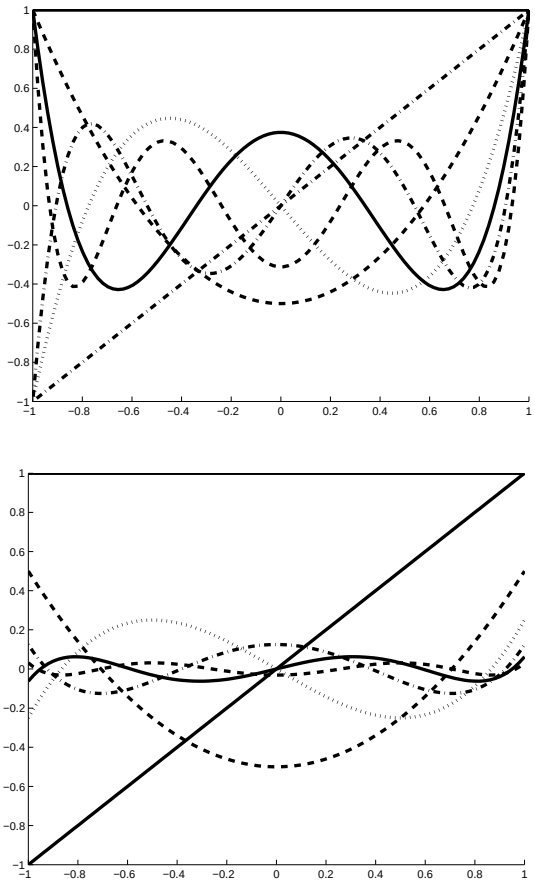
Jedina suštinska razlika u kodu u odnosu na prethodnu funkciju je petlja

```
for i=2:n
    T{i+1}=2*[T{i} 0]-[0 0 T{i-1}];
end
```

Čebisevljevi polinomi se mogu konstruisati i direktno na osnovu formule $T_n(x) = \cos(n \arccos(x))$. Mogu se grafički prikazati sledećim komandama

```
L = legendre_poly(6);
X = linspace(-1, 1);
colors = 'rgbcmyk';
hold on
for i = 1 : 7
    plot(X, polyval(L{i}, X), colors(i));
end
```

Na slici 3.2 su prikazani Legendre-ovi (`L{i}`) i monični Čebiševljevi polinomi (`T{i}/C{i}(1)`).



Slika 3.2: Legendre-ovi i Čebišev-ljevi polinomi

4

Aproksimacija funkcija

Neka je f element linearnog normiranog prostora $\mathcal{R}$ i neka su $g_1, \ldots, g_n$ dati linearno nezavisni elementi tog prostora. $Q_0 = \sum_{i=1}^{n} c_i^\circ g_i$ je element najbolje aproksimacije za f određen elementima $g_1, \ldots, g_n$ ako je

$$E_n(f) = \left\| f - \sum_{i=1}^{n} c_i^\circ g_i \right\| = \inf_{c_1,\ldots,c_n} \left\| f - \sum_{i=1}^{n} c_i g_i \right\|.$$

4.1 *Pri merenju neke veličine y dobijene su vrednosti a_k, $k = 1, \ldots, n$. Odrediti vrednost za y za koju je suma kvadrata relativnih grešaka minimalna.*

Rešenje: Zadatak se svodi na minimizaciju po y funkcionala

$$S(y) = \sum_{k=1}^{n} \left(\frac{a_k - y}{y} \right)^2.$$

Minimum funkcionala $S(y)$ se postiže u tački u kojoj je $\frac{dS}{dy} = 0$, tj.

$$y = \left(\sum_{k=1}^{n} a_k^2 \right) \Big/ \left(\sum_{k=1}^{n} a_k \right) \qquad \text{za } \sum_{k=1}^{n} a_k \neq 0.$$

4.2 *a) Dato je n tačaka u ravni $P_k = (x_k, y_k)$, $k = 1, \ldots, n$. P_0 je još jedna tačka ravni. Dokazati da postoji prava $y = c_0 + c_1 x$, koja prolazi kroz tačku P_0, takva da je*

$$(*) \qquad \max_{1 \le k \le n} |y_k - (c_0 + c_1 x_k)| = \textit{min.}$$

b) Naći pravu koja prolazi kroz tačku $P_0 = (0, 1)$ i zadovoljava uslov $(*)$, ako je $P_1 = (1, 3)$, $P_2 = (2, 5)$ i $P_3 = (3, 2)$.

Rešenje: a) Pošto tražena prava prolazi kroz tačku P_0, nepoznati parametri moraju zadovoljavati sledeću vezu

$$y_0 = c_0 + c_1 x_0, \qquad \text{tj.} \qquad c_0 = y_0 - c_1 x_0,$$

te se uslov $(*)$ svodi na uslov

$$\max_{1\le k\le n} |(y_k - y_0) - c_1(x_k - x_0)| = \min.$$

Vektori $\mathbf{y} = (y_1 - y_0, \ldots, y_n - y_0)^\top$ i $\mathbf{x} = (x_1 - x_0, \ldots, x_n - x_0)^\top$ su elementi prostora $\mathcal{R}^n$, te se problem svodi na nalaženje elementa najbolje aproksimacije oblika $c\mathbf{x}$ za dati vektor $\mathbf{y}$ u smislu uniformne norme $\|\mathbf{z}\| = \max_{1\le k\le n} |z_k|$.

S obzirom da u svakom normiranom prostoru postoji element najbolje aproksimacije, to postoji konstanta c_1 takva da je $\|\mathbf{y} - c_1\mathbf{x}\| = \min$, tj. prava $y = c_0 + c_1 x$, koja prolazi kroz tačku P_0 i zadovoljava uslov $(*)$.

Napomena: Prostor $\mathcal{R}^n$ nije strogo normiran u smislu uniformne norme, te se ne može garantovati jedinstvenost rešenja.

b) Za dati skup podataka je, u oznakama uvedenim pod a), $\mathbf{x} = (1, 2, 3)^\top$ i $\mathbf{y} = (2, 4, 1)^\top$, a uslov $(*)$ je

$$\|\mathbf{y} - c_1\mathbf{x}\| = \max_{c_1}\{|2 - c_1|, |4 - 2c_1|, |1 - 3c_1|\} = \min.$$

Minimum se dostiže za $c_1 = 1$ i jednak je 2. Stoga su tražena prava i greška aproksimacije

$$y = x + 1, \qquad \max_{1\le k\le 3} |y_k - (x_k + 1)| = 2.$$

4.3 *Naći najbolju aproksimaciju vektora* $\mathbf{y} = [3,\ 5,\ 2]^\top \in \mathcal{R}^3$ *vektorima* $\mathbf{x}_1 = [1,\ 0,\ 0]^\top$ *i* $\mathbf{x}_2 = [0,\ 1,\ 0]^\top$ *ako je rastojanje definisano*

$$(a) \quad \textit{euklidskom} \qquad \|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{3} v_i^2},$$

$$(b) \quad \textit{apsolutnom} \qquad \|\mathbf{v}\|_1 = \sum_{i=1}^{3} |v_i|,$$

$$(c) \quad \textit{uniformnom} \qquad \|\mathbf{v}\|_\infty = \max_{1\le i\le 3} |v_i|$$

normom.

Rešenje: Odrediti najbolju aproksimaciju u ovom slučaju znači naći linearnu kombinaciju vektora $\mathbf{x}_1$ i $\mathbf{x}_2$ oblika $Q^0 = c_1^0\mathbf{x}_1 + c_2^0\mathbf{x}_2$ takvu da je rastojanje između tako dobijenog vektora i vektora $\mathbf{y}$ miminalno u odnosu na datu normu, tj. naći vrednosti koeficijenata c_1^0 i c_2^0 takve da važi

$$E = \|\mathbf{y} - Q^0\| = \inf_{c_1, c_2} \|\mathbf{y} - (c_1\mathbf{x}_1 + c_2\mathbf{x}_2)\|.$$

Ovde je:

$$\mathbf{y} - Q = \begin{bmatrix} 3 \\ 5 \\ 2 \end{bmatrix} - c_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - c_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 - c_0 \\ 5 - c_1 \\ 2 \end{bmatrix}$$

(a) U slučaju euklidske norme traži se

$$E = \inf_{c_1,c2} \|\mathbf{y} - Q\|_2 = \inf_{c_1,c2} \sqrt{(3 - c_1)^2 + (5 - c_2)^2 + 2^2}.$$

S obzirom da je $(3 - c_1)^2 \geq 0$ i $(5 - c_2)^2 \geq 0$, to je

$$E = 2 \quad \text{za} \quad c_1^0 = 3,\, c_2^0 = 5 \qquad \Longrightarrow \qquad Q^0 = 3\mathbf{x}_1 + 5\mathbf{x}_2.$$

(b) U slučaju apsolutne norme traži se

$$E = \inf_{c_1,c1} \|\mathbf{y} - Q\|_1 = \inf_{c_1,c2} (|3 - c_1| + |5 - c_2| + |2|).$$

Rešenje je isto kao u prethodnom slučaju, $E = 2$, $c_1^0 = 3$, $c_2^0 = 5$.
(c) U slučaju uniformne norme traži se

$$E = \inf_{c_1,c2} \|y - Q\|_\infty = \inf_{c_1,c2} \big(\max(|3 - c_1|, |5 - c_2|, 2)\,\big).$$

Za $|3 - c_1| \leq 2$ i $|5 - c_2| \leq 2$ greška dostiže minimum $E = 2$. Dakle, u ovom slučaju, najbolja aproksimacija nije jednoznačno određena, već je $1 \leq c_1^0 \leq 5$ i $3 \leq c_2^0 \leq 7$.

4.4 *Neka je u prostoru linearnih funkcija $f(x) = c_1 x + c_0$ norma definisana kao $\|f\| = |f(0)| + |f(1)|$. Dokazati da ovo jeste norma, a zatim odrediti konstantu c koja najbolje aproksimira funkciju $f(x) = x$.*

Rešenje: Osobine norme su

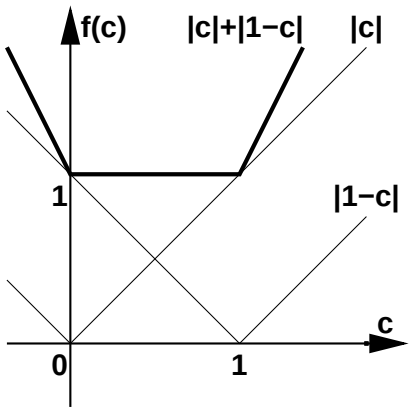
$$\|f\| \geq 0, \quad \|f\| = 0 \Leftrightarrow f = 0, \qquad \|\alpha f\| = |\alpha| \|f\|, \qquad \|f + g\| \leq \|f\| + \|g\|.$$

U ovom primeru je očigledno da je $\|f\| \geq 0$. Dalje,

$$\begin{aligned}
&f = 0 \quad \Rightarrow \quad |f(0)| + |f(1)| = 0 \quad \Rightarrow \quad \|f\| = 0 \\
&\|f\| = 0 \quad \Rightarrow \quad |f(0)| + |f(1)| = 0 \quad \Rightarrow \quad f(0) = 0,\ f(1) = 0 \quad \Rightarrow \quad f = 0 \\
&\|\alpha f\| = |\alpha f(0)| + |\alpha f(1)| = |\alpha||f(0)| + |\alpha||f(1)| = |\alpha|(|f(0)| + |f(1)|) = |\alpha| \|f\| \\
&\|f + g\| = |(f + g)(0)| + |(f + g)(1)| = |f(0) + g(0)| + |f(1) + g(1)| \\
&\qquad \leq |f(0)| + |g(0)| + |f(1)| + |g(1)| = \|f\| + \|g\|
\end{aligned}$$

što znači da je zadatim izrazom zaista definisana norma. U ovom zadatku se traži da se funkcija $f(x) = x$ aproksimira funkcijom $Q(x) = c$, tj. traži se takva vrednost konstante $Q^0(x) = c^0$ da važi

$$\|x - c_0\| = \inf_c \|x - c\| = \inf_c (|-c| + |1 - c|) = \inf_c (|c| + |1 - c|).$$



Slika 4.1: Grafik funkcije $|c| + |1 - c|$.

Grafik funkcije $|c| + |1 - c|$ ima oblik kao na slici 4.1.

Dakle, za bilo koju konstantu c iz intervala $[0, 1]$ greška najbolje aproksimacije jednaka je 1, što znači da najbolja aproksimacija nije jednoznačno određena. Prostor linearnih funkcija sa ovako definisanom normom nije strogo normiran. Naime, prostor je strogo normiran ako važi:

$$\|f + g\| = \|f\| + \|g\| \Leftrightarrow f = \lambda g$$

Kontraprimer u datom prostoru predstavljaju recimo funkcije $f(x) = 1$ i $g(x) = x$, jer je $\|f + g\| = \|f\| + \|g\| = 3$, iako je $f \neq \lambda g$.

4.1 Srednjekvadratna aproksimacija

Srednjekvadratna aproksimacija funkcije $f(x)$ na intervalu $[a, b]$ je generalisani polinom

$$Q_n(x) = \sum_{k=0}^{n} c_k g_k(x),$$

gde su $g_k(x)$, $k = 0, \ldots, n$, date linearno nezavisne funkcije, a koeficijenti c_k rešenja sistema linearnih jednačina

$$\sum_{k=0}^{n} c_k (g_k, g_j) = (f, g_j), \quad j = 0, \ldots, n.$$

Skalarni proizvod i odgovarajuća norma definisani su izrazima

$$(f,g)=\int_a^b p(x)f(x)g(x)\,dx, \qquad \|f\|=\sqrt{(f,f)}.$$

$p(x)>0$ je data neprekidna funkcija i naziva se težinska funkcija.

Ako srednjekvadratnu aproksimaciju određujemo u prostoru polinoma ortogonalnih u odnosu na težinsku funkciju $p(x)$, matrica sistema linearnih jednačina po koeficijentima c_k je dijagonalna, te su koeficijenti

$$c_j=\frac{(f,g_j)}{(g_j,g_j)}, \quad j=0,\dots,n, \qquad \text{za} \qquad (g_k,g_j)=0 \qquad k\neq j$$

Greška srednjekvadratne aproksimacije je

$$E_n(f)=\|f-Q_n\|$$

4.5 *Naći najbolju srednjekvadratnu aproksimaciju polinomom drugog stepena funkcije* $f(x)=\sqrt{x}$ *na intervalu* $[0,1]$.

Rešenje: Pošto se traži aproksimacija polinomom drugog stepena, za bazne funkcije se uzimaju $g_0(x)=1$, $g_1(x)=x$ i $g_2(x)=x^2$ i element najbolje aproksimacije ima oblik $p_2(x)=c_0+c_1x+c_2x^2$, Koeficijenti c_0, c_1, c_2 se određuju kao rešenja sistema jednačina:

$$\sum_{k=0}^{2} c_i(g_k,g_j)=(f,g_j), \quad j=0,1,2.$$

Ako vrednost težinske funkcije $p(x)$ nije eksplicitno data, uzećemo da je $p(x)\equiv 1$, pa je

$$\begin{aligned}(g_k,g_j) &= \textstyle\int_0^1 x^k x^j dx=\frac{x^{k+j+1}}{k+j+1}\Big|_0^1=\frac{1}{k+j+1}\\ (f,g_j) &= \textstyle\int_0^1 \sqrt{x}x^j dx=\int_0^1 x^{\frac12}x^j dx=\frac{x^{j+\frac32}}{j+\frac32}\Big|_0^1=\frac{1}{j+\frac32}=\frac{2}{2j+3}\end{aligned}$$

i dalje:

$$\begin{array}{lll} c_0\cdot 1+c_1\cdot\frac12+c_2\cdot\frac13=\frac23 & & c_0=\frac{6}{35}\\ c_0\cdot\frac12+c_1\cdot\frac13+c_2\cdot\frac14=\frac25 & \Longrightarrow & c_1=\frac{48}{35}\\ c_0\cdot\frac13+c_1\cdot\frac14+c_2\cdot\frac15=\frac27 & & c_2=-\frac47\end{array}$$

Element najbolje aproksimacije je $p_2(x)=\frac{6}{35}+\frac{48}{35}x-\frac47x^2$.

Treba primetiti da je matrica sistema jednačina Hilbertova matrica,

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix}$$

te je sistem loše uslovljen, tj. male promene ulaznih parametara dovode do velikih promena rešenja.

4.6 *Odrediti polinom drugog stepena $p_2(x)$ najbolje srednjekvadratne aproksimacije funkcije $f(x) = |x-1|$ na intervalu $[0,2]$. Izračunati grešku aproksimacije.*

Rešenje: Aproksimaciju tražimo u obliku

$$p_2(x) = \sum_{k=0}^{2} c_k g_k(x), \qquad g_k(x) = x^k, \quad k = 0,1,2.$$

Koeficijente c_k određujemo kao rešenje sistema jednačina

$$\sum_{k=0}^{2} c_k (g_k, g_j) = (f, g_j), \quad j = 0,1,2, \qquad \text{gde je} \quad (f,g) = \int_0^2 f(x)g(x)\,dx,$$

tj.

$$2c_0 + 2c_1 + \frac{8}{3}c_2 = 1, \qquad 2c_0 + \frac{8}{3}c_1 + 4c_2 = 1, \qquad \frac{8}{3}c_0 + 4c_1 + \frac{32}{5}c_2 = \frac{3}{2}.$$

Tako se dobija aproksimacija

$$p_2(x) = \frac{9}{8} - \frac{15}{8}x + \frac{15}{16}x^2 = \frac{15}{16}\left(\frac{1}{5} + (x-1)^2\right).$$

Greška aproksimacije je

$$\|f - p_2\|^2 = \int_0^2 \big(f(x) - p_2(x)\big)^2 dx = 0.01, \quad \text{tj.} \quad \|f - p_2\| = 0.1.$$

4.7 *U prostoru $\mathcal{C}(0,1)$ definisan je skalarni proizvod*

$$(f,g) = \int_0^1 x(1-x)f(x)g(x)\,dx$$

i odgovarajuća norma $\|f\| = \sqrt{(f,f)}$. Neka je $\mathcal{P}$ potprostor polinoma drugog stepena. Odrediti polinom $p_2(x) \in \mathcal{P}$ takav da je

$$\|e^x - p_2(x)\| = \min_{p \in \mathcal{P}} \|e^x - p(x)\|$$

Koeficijente polinoma izračunati sa tačnošću 10^{-3}.

Rešenje: Polinom $p_2(x) = c_0 + c_1 x + c_2 x^2$ je element najbolje srednjekvadratne aproksimacije funkcije e^x, te su njegovi koeficijenti rešenja sistema

$$\sum_{k=0}^{2} c_k (x^k, x^j) = (e^x, x^j), \qquad j = 0, 1, 2,$$

tj.

$$\begin{aligned} 10c_0 + \ 5c_1 + \ 3c_2 &= 16.9031 \\ 5c_0 + \ 3c_1 + \ 2c_2 &= \ 9.2907 \\ 21c_0 + 14c_1 + 10c_2 &= 41.5380 \end{aligned} \quad .$$

Rešenja su $c_0 = 1.0186$, $c_1 = 0.8411$, $c_2 = 0.8372$, i traženi polinom je

$$p_2(x) = 1.019 + 0.841x + 0.837x^2.$$

4.8 *Naći najbolju srednjekvadratnu aproksimaciju polinomom trećeg stepena funkcije* $f(x) = 3^x$ *na intervalu* $[-1, 1]$.

Rešenje: Problem se može rešavati na uobičajenom bazisu $g_k(x) = x^k$, $k = 0, \ldots, 3$. Tako se dobija:

$$\begin{aligned} (g_k, g_j) &= \int_{-1}^{1} x^k x^j dx = \left.\frac{x^{k+j+1}}{k+j+1}\right|_{-1}^{1} = \frac{1-(-1)^{k+j+1}}{k+j+1} \\ (f, g_0) &= \int_{-1}^{1} 3^x dx = 2.4273 \\ (f, g_1) &= \int_{-1}^{1} x 3^x dx = 0.8246 \\ (f, g_2) &= \int_{-1}^{1} x^2 3^x dx = 0.9260 \\ (f, g_3) &= \int_{-1}^{1} x^3 3^x dx = 0.5052 \end{aligned}$$

i dalje:

$$\begin{array}{rrrr} 2\,c_0 & + \frac{2}{3}\,c_2 & = 2.4273 & \\ & \frac{2}{3}\,c_1 + \frac{2}{5}\,c_3 & = 0.8246 & \\ \frac{2}{3}\,c_0 & + \frac{2}{5} c_2 & = 0.9260 & \\ & \frac{2}{5}\,c_1 + \frac{2}{7}\,c_3 & = 0.5052 & \end{array} \quad \Longrightarrow \quad \begin{array}{l} c_0 = 0.9940 \\ c_1 = 1.1000 \\ c_2 = 0.6576 \\ c_3 = 0.2335 \end{array}$$

Polinom najbolje aproksimacije je $p_3(x) = 0.9940 + 1.1000x + 0.6576x^2 + 0.2335x^3$.

Međutim, kako je sistem jednačina za izračunavanje koeficijenata c_k loše uslovljen, bolje je za bazis odabrati familiju polinoma ortogonalnih na $[-1, 1]$ u odnosu na težinsku funkciju $p(x) = 1$, tj. skup Legendre-ovih polinoma (vidi §3.2). U tom slučaju je:

$$\begin{aligned} (f, L_0) &= \textstyle\int_{-1}^{1} 3^x dx = 2.4273 && \Rightarrow \quad c_0 = \tfrac{1}{2}(f, L_0) = 1.2136 \\ (f, L_1) &= \textstyle\int_{-1}^{1} x 3^x dx = 0.8246 && \Rightarrow \quad c_1 = \tfrac{3}{2}(f, L_1) = 1.2369 \\ (f, L_2) &= \textstyle\int_{-1}^{1} \frac{1}{2}(3x^2 - 1) 3^x dx = 0.1753 && \Rightarrow \quad c_2 = \tfrac{5}{2}(f, L_2) = 0.4384 \\ (f, L_3) &= \textstyle\int_{-1}^{1} \frac{1}{2}(5x^3 - 3x) 3^x dx = 0.0261 && \Rightarrow \quad c_3 = \tfrac{7}{2}(f, L_3) = 0.0913 \end{aligned}$$

jer je $(L_j\,,\,L_j) = 2/(2j+1)$. Element najbolje aproksimacije je

$$\begin{aligned} p_3(x) &= 1.2136 + 1.2369x + 0.4304\tfrac{1}{2}(3x^2-1) + 0.0913\tfrac{1}{2}(5x^3-3x) \\ &= 0.9945 + 1.0968x + 0.6576x^2 + 0.2336x^3 \end{aligned}$$

4.9 *Naći najbolju srednjekvadratnu aproksimaciju polinomom petog stepena funkcije* $f(x) = |x|$ *na intervalu* $[-1,1]$ *u slučajevima kada je težinska funkcija jednaka* $p(x) = 1$ *i* $p(x) = 1/\sqrt{1-x^2}$.

Rešenje: U prvom slučaju ćemo za bazis izabrati sistem Legendre-ovih polinoma, koji su ortogonalni u odnosu na težinsku funkciju $p(x) = 1$. Treba uočiti da su Legendre-ovi polinomi parnog indeksa parne, a neparnog indeksa neparne funkcije, pa važi

$$c_{2k+1} = \int_{-1}^{1} |x| L_{2k+1} dx = 0 \qquad c_{2k} = \int_{-1}^{1} |x| L_{2k} dx = 2\int_{0}^{1} x L_{2k} dx$$

Stoga je:

$$\begin{aligned} c_0 &= 2\cdot\tfrac{2\cdot 0+1}{2}\textstyle\int_0^1 x dx = 0.5 \\ c_2 &= 2\cdot\tfrac{2\cdot 2+1}{2}\textstyle\int_0^1 x\tfrac{1}{2}(3x^2-1)dx = 0.625 \\ c_4 &= 2\cdot\tfrac{2\cdot 4+1}{2}\textstyle\int_0^1 x\tfrac{1}{8}(35x^4-30x^2+3)dx = -0.1875 \end{aligned}$$

pa je element najbolje aproksimacije

$$p_5^{(1)}(x) = 0.5 + 0.625\cdot\frac{1}{2}(3x^2-1) - 0.1875\cdot\frac{1}{8}(35x^4-30x^2+3).$$

U slučaju težinske funkcije $p(x) = 1/\sqrt{1-x^2}$, koriste se Čebišev-ljevi polinomi. Takođe važi da su ovi polinomi parnog indeksa parne, a neparnog indeksa neparne funkcije, pa je

$$c_{2k+1} = \int_{-1}^{1} \frac{|x| T_{2k+1}}{\sqrt{1-x^2}} dx = 0 \qquad c_{2k} = \int_{-1}^{1} \frac{|x| T_{2k}}{\sqrt{1-x^2}} dx = 2\int_{0}^{1} \frac{x T_{2k}}{\sqrt{1-x^2}} dx$$

Dalje je

$$\begin{aligned} c_0 &= 2\cdot\tfrac{1}{\pi}\textstyle\int_0^1 \frac{x}{\sqrt{1-x^2}}dx = 0.6366 \\ c_2 &= 2\cdot\tfrac{2}{\pi}\textstyle\int_0^1 \frac{x(2x^2-1)}{\sqrt{1-x^2}}dx = 0.4244 \\ c_4 &= 2\cdot\tfrac{2}{\pi}\textstyle\int_0^1 \frac{x(8x^4-8x^2+1)}{\sqrt{1-x^2}}dx = -0.0849 \end{aligned}$$

pa je element najbolje aproksimacije

$$p_5^{(2)}(x) = 0.6366 + 0.4244(2x^2-1) - 0.0849(8x^4-8x^2+1).$$

Ako se izračunaju vrednosti $p_5^{(1)}(1)$ i $p_5^{(2)}(1)$ i uporede sa vrednošću funkcije u toj tački, može se uočiti uloga težinske funkcije – aproksimacija Čebišev-ljevim polinomima (za koju je težinska funkcija u toj tački veća) je bolja.

4.10 *U prostoru* $\mathcal{L} = \{f \mid \int_0^\infty e^{-x}|f(x)|^2\,dx < \infty\}$ *definisani su skalarni proizvod i norma*

$$(f,g) = \int_0^\infty e^{-x} f(x)g(x)\,dx, \qquad \|f\| = \sqrt{(f,f)}.$$

Neka je

$$f(x) = \begin{cases} e^{x-1}\sin x, & 0 \le x \le 1 \\ 0, & x > 1 \end{cases}.$$

Naći element $p_2(x)$ *iz potprostora* $\mathcal{P}$ *svih polinoma drugog stepena, takav da je*

$$\|f(x) - p_2(x)\| = \min_{p\in\mathcal{P}} \|f(x) - p(x)\|.$$

Računati sa tačnošću 10^{-4}.

Rešenje: Rešenje je polinom najbolje srednjekvadratne aproksimacije te se, kao u prethodnom zadatku, koeficijenti određuju kao rešenja sistema jednačina

$$\begin{array}{rcl} c_0 + c_1 + 2c_2 = 0.16911 & & c_0 = \ \ 0.21602 \\ c_0 + 2c_1 + 6c_2 = 0.11079 & \Longrightarrow & c_1 = -0.035507 \\ 2c_0 + 6c_1 + 24c_2 = 0.082127 & & c_2 = -0.005703 \end{array}$$

Traženi polinom je

$$p_2(x) = 0.21602 - 0.035507x - 0.005703x^2,$$

a greška aproksimacije

$$\|f(x) - p_2(x)\| = 0.12.$$

4.11 *Neka je* $\mathcal{X} \subseteq \mathcal{C}[0,2]$ *prostor sa skalarnim proizvodom*

$$(*) \qquad (f,g) = \int_0^2 \frac{f(x)g(x)}{\sqrt{x(2-x)}}\,dx$$

i odgovarajućom normom, a $\mathcal{P}$ *potprostor polinoma trećeg stepena. Naći*

$$\inf_{p\in\mathcal{P}} \|\sqrt{x} - p(x)\|.$$

Rešenje: U integralu $(*)$ uvedimo smenu $x = 1 - t$, pa je

$$(f,g) = \int_{-1}^1 \frac{f(1-t)g(1-t)}{\sqrt{1-t^2}}\,dt, \qquad \|f\|^2 = \int_{-1}^1 \frac{[f(1-t)]^2}{\sqrt{1-t^2}}\,dt.$$

S obzirom da je težinska funkcija $1/\sqrt{1-t^2}$, tražićemo polinom najbolje srednjekvadratne aproksimacije funkcije $\sqrt{1-t}$ u obliku linearne kombinacije Čebiševljevih polinoma zaključno sa polinomom trećeg stepena:

$$q_3(t) = \sum_{k=0}^{3} c_k T_k(t).$$

Zbog ortogonalnosti Čebišev-ljevih polinoma u odnosu na dati skalarni proizvod koeficijenti c_k su određeni sistemom linearnih jednačina sa dijagonalnom matricom

$$c_j(T_j, T_j) = (\sqrt{1-t}, T_j), \qquad j = 0, 1, 2, 3.$$

Rešenja sistema su

$$c_0 = \frac{2\sqrt{2}}{\pi}, \quad c_1 = -\frac{4\sqrt{2}}{3\pi}, \quad c_2 = -\frac{4\sqrt{2}}{15\pi}, \quad c_2 = -\frac{4\sqrt{2}}{35\pi},$$

pa je polinom najbolje aproksimacije

$$q_3(t) = \frac{2\sqrt{2}}{\pi}\left(1 - 2\big(\frac{t}{3} + \frac{2t^2-1}{15} + \frac{4t^3-3t}{35}\big)\right),$$

tj.

$$p_3(x) = q_3(1-x) = \frac{2\sqrt{2}}{105\pi}(15 + 180x - 100x^2 + 24x^3).$$

Greška aproksimacije je

$$\|\sqrt{x} - p_3(x)\| = \|\sqrt{1-t} - \sum_{k=0}^{3} c_k T_k(t)\| = 0.05.$$

4.12 *Neka je*

$$p_n(x) = \sum_{k=0}^{n} c_k T_k(x)$$

polinom najbolje srednjekvadratne aproksimacije funkcije $f \in \mathcal{C}^1[-1,1]$ određen Čebišev-ljevim polinomima $T_k(x)$ na intervalu $[-1,1]$. Pokazati da se koeficijenti c_k računaju, za $k \geq 1$, po formuli

$$c_k = \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)\cos k\theta \, d\theta,$$

i da važi ocena

$$|c_k| \leq C(f)/k,$$

gde je C konstanta koja zavisi samo od f.

Rešenje: Kako je

$$c_k = \frac{(f, T_k)}{(T_k, T_k)}, \qquad \text{gde je} \qquad (f,g) = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\, dx,$$

uvođenjem smene $x = \cos\theta$ i izračunavanjem integrala dobija se izraz za c_k.

Parcijalnom integracijom integrala u izrazu za c_k dobija se da je

$$c_k = \frac{2}{\pi}\frac{1}{k}\int_0^{\pi} f'(\cos\theta)\sin\theta\sin k\theta \, d\theta,$$

pa je

$$|c_k| \le \frac{2}{\pi}\,\frac{1}{k}\max_{[-1,1]}|f'(x)|\int_0^{\pi}|\sin k\theta|\,d\theta.$$

Kako je

$$\int_0^{\pi}|\sin k\theta|\,d\theta = \frac{1}{k}\int_0^{k\pi}|\sin t|\,dt = \frac{1}{k}\sum_{j=1}^{k}\int_{(j-1)\pi}^{j\pi}(-1)^{j-1}\sin t\,dt$$

$$= \frac{1}{k}\sum_{j=1}^{k}(-1)^{j-1}\,2(-1)^{j-1} = 2,$$

to je

$$|c_k| \le \frac{4}{\pi}\,\frac{1}{k}\max_{[-1,1]}|f'(x)|.$$

Ako se uvede oznaka $C(f) = \frac{4}{\pi}\max_{[-1,1]}|f'(x)|$, dobija se ocena za c_k.

4.13 *Data je funkcija* $f(x) = \arccos x$, $x \in [-1,1]$. *Odrediti najmanji prirodan broj* m *i polinom* $p_m(x)$ *stepena* m *najbolje srednjekvadratne aproksimacije za funkciju* $f(x)$ *na intervalu* $[-1,1]$, *tako da je*

$$\|f - p_m\| = \left(\int_{-1}^{1}\frac{(f(x)-p_m(x))^2}{\sqrt{1-x^2}}\,dx\right)^{\frac{1}{2}} \le 0.1.$$

Rešenje: U odnosu na skalarni proizvod

$$(f,g) = \int_{-1}^{1}\frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx$$

ortogonalni su Čebišev-ljevi polinomi $T_k(x) = \cos(k\arccos x)$, te ćemo polinom $p_m(x)$ tražiti u obliku

$$p_m(x) = \sum_{k=0}^{m} c_k T_k(x), \qquad c_k = \frac{(f,T_k)}{(T_k,T_k)}\,.$$

Kako je

$$(f,T_k) = \begin{cases}((-1)^k-1)/k^2, & k\neq 0\\ \pi^2/2, & k=0\end{cases}, \qquad (T_k,T_k) = \begin{cases}\pi/2, & k\neq 0\\ \pi, & k=0\end{cases},$$

to je $c_0 = \pi/2$, $c_1 = -4/\pi$, $c_2 = 0$, $c_3 = -4/(9\pi), \ldots$. Stepen m polinoma $p_m(x)$ koji zadovoljava zahtevanu tačnost je tri, jer je

$$\|f-p_0\| = 1.9, \qquad \|f-p_1\| = \|f-p_2\| = 0.19, \qquad \|f-p_3\| = 0.077 < 0.1,$$

a traženi polinom je

$$p_3(x) = \frac{\pi}{2} - \frac{4}{\pi}x - \frac{4}{9\pi}(4x^3-3x).$$

4.14 *U prostoru funkcija* $\mathcal{C}^1(0,\pi)$ *definisani su skalarni proizvod i norma*

$$(f,g)=\int_0^\pi \big(f(x)g(x)+f'(x)g'(x)\big)\,dx, \qquad \|f\|=\sqrt{(f,f)}.$$

Naći polinom $p_n(x)\in\mathcal{P}$ *takav da je*

$$\|\sin x-p_n(x)\|=\inf_{p\in\mathcal{P}}\|\sin x-p(x)\|,$$

gde je $\mathcal{P}$ *skup polinoma stepena ne većeg od a) jedan, b) dva.*

Rešenje: Traženi polinomi najbolje srednjekvadratne aproksimacije su

$$(a)\quad p_1(x)=\frac{2}{\pi}, \qquad (b)\quad p_2(x)=\frac{12}{\pi^2+60}\left(\pi+\frac{5}{\pi^3}\,(12-\pi^2)\,x(\pi-x)\right).$$

4.15 *U prostoru* $\mathcal{C}^1(0,1)$ *definisan je skalarni proizvod i norma*

$$(f,g)=\int_0^1 f'(x)g'(x)\,dx+f(0)g(0), \qquad \|f\|=\sqrt{(f,f)}.$$

Izračunati $\min_{(c_0,c_1,c_2)\in\mathcal{R}^3}\|e^x-c_0-c_1x-c_2x^2\|$.

Rešenje: Traženi polinom je polinom najbolje srednjekvadratne aproksimacije

$$p_2(x)=1+0.8731x+0.8452x^2,$$

pa je

$$\min_{(c_0,c_1,c_2)\in\mathcal{R}^3}\|e^x-c_0-c_1x-c_2x^2\|=\|e^x-p_2(x)\|=0.0628.$$

4.16 *Odrediti*

$$E=\min_{c_{-1},c_0,c_1}\int_{-\infty}^{\infty}\Big(e^{-|x|}-c_{-1}g_{-1}(x)-c_0g_0(x)-c_1g_1(x)\Big)^2dx$$

ako je

$$g_k(x)=\begin{cases} x-k+1, & x\in[k-1,k]\\ -x+k+1, & x\in[k,k+1]\\ 0\quad , & x\neq[k-1,k+1]\end{cases}$$

Rešenje: Veličina E predstavlja grešku srednjekvadratne aproksimacije funkcije $e^{-|x|}$ "krov" funkcijama $g_k(x)$, $k=-1,0,1$ na intervalu $[-2,2]$, s obzirom da su van tog intervala sve tri bazisne funkcije identički jednake nuli. Kao rešenje sistema jednačina

$$\sum_{k=-1}^{1}c_k^\circ(g_k,g_j)=(e^{-|x|},g_j), \qquad j=-1,0,1,$$

gde je $(f,g) = \int_{-\infty}^{\infty} f(x)g(x)\,dx$, dobija se da je $c_{-1}^{\circ} = c_1^{\circ} = 0.36967$, $c_0^{\circ} = 0.91881$, pa je

$$E = \int_{-\infty}^{\infty} \left(e^{-|x|} - c_{-1}^{\circ} g_{-1}(x) - c_0^{\circ} g_0(x) - c_1^{\circ} g_1(x)\right)^2 dx = 0.02855$$

C Implementacija aproksimacije funkcije polinomima biće predstavljena na primeru Čebišev-ljevih polinoma. Za izračunavanje odgovarajućeg Čebišev-ljevog polinoma biće korišćena direktna formula $T_n(x) = \cos(n \arccos x)$, tako da je implementacija prilično jednostavna i sastoji se u prostom prevođenju izraza za izračunavanje datog koeficijenta najbolje aproksimacije u kod. Jedini problem pri implementaciji jeste računanje integrala. S obzirom da je numerička integracija tema za sebe, u sklopu biblioteke `libnumerics` implementirana je posebna metoda za numeričko računanje integrala za potrebe metode aproksimacije Čebišev-ljevim polinomima. U datom slučaju se, pošto je podintegralna funkcija singularna u krajnjim tačkama intervala, moraju koristiti kvadraturne formule Gauss-ovog tipa, a prirodan izbor je formula sa čvorovima u nulama Čebišev-ljevog polinoma. Procedura koja se odnosi na ovu metodu integracije data je u prethodnom poglavlju.

Sada može biti prezentirana procedura koji implementira aproksimaciju Čebišev-ljevim polinomima. Kako je već pomenuto gore, kod je vrlo jednostavan i za njegovo razumevanje bi odgovarajući komentari trebali da budu više nego dovoljni. Jedini detalj koga vredi pojasniti jeste fiksiranje broja podintervala za integraciju. Tačnost Gauss-ove kvadraturne formule za numeričku integraciju sa čvorovima u nulama Čebišev-ljevog polinoma je utoliko veća ukoliko je broj čvorova veći, ali takođe zavisi i od maksimuma izvoda reda dva puta većeg od broja čvorova (a broj podintervala je jednak broju čvorova). S obzirom da se generalno ne zna oblik funkcije koja se aproksimira, pa prema tome ni maksimum odgovarajućeg izvoda, najbolje što se može uraditi jeste da se fiksira broj podintervala na razumno visoku vrednost. Sledi procedura koja implementira aproksimaciju Čebišev-ljevim polinomima:

```
#include <assert.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <matheval.h>

extern double   chebyshev(char *function, int n);

/*
 * Funkcija approximation() izracunava koeficijente najbolje aproksimacije
 * Cebisevljevim polinomima za datu funkciju na intervalu [-1,1].
 * Argumenti funkcije su:
 *   function - string koji predstavlja funkciju
 *   n - red polinoma najbolje aproksimacije
 *   c0 - polje u koje se smestaju koeficijenti najbolje aproksimacije
 * Argumenti funkcije moraju da zadovolje uslov da je funkcija sintaksno
 * ispravno zadata (za nacin zadavanja funkcije videti dokumentaciju
 * libmatheval biblioteke), te da je red polinoma najbolje aproksimacije
```

```
 * veci ili jednak 0.
 */
void
approximation(char *function, int n, double *c0)
{
        const double    pi = 3.14159265358979323846;    /* Konstanta pi. */
        const int       nodes_count = 100;      /* Broj podintervala za
                                                   * racunanje integrala. */
        void           *evaluator;      /* Evaluator za datu funkciju. */
        int             digits; /* Broj cifara reda polinoma najbolje
                                 * aproksimacije. */
        int             saved_n;        /* Kopija reda polinoma najbolje
                                         * aproksimacije. */
        char           *subint; /* Tekstualna predstava podintegralne
                                 * funkcije u tekucoj iteraciji. */
        int             i;      /* Brojac u petljama. */

        /* Proveravaju se ulazni podaci (za funkciju tako sto se pokusa
         * kreirati odgovarajuci evaluator). */
        evaluator = evaluator_create(function);
        assert(evaluator != NULL);
        evaluator_destroy(evaluator);
        assert(n >= 0);

        /* Odredjuje se broj cifara reda polinoma najbolje aproksimacije. */
        saved_n = n;
        for (digits = 0; n > 0; n /= 10, digits++);
        n = saved_n;

        /* Alocira se polje karaktera dovoljne velicine da primi
         * podintegralnu funkciju. */
        subint =
            (char *) malloc((4 + digits + 13 + strlen(function) + 13 + 1) *
                            sizeof(char));

        for (i = 0; i <= n; i++) {
                /* Kreira se podintegralna funkcija za racunanje tekuceg
                 * koeficijenta polinoma najbolje aproksimacije. */
                sprintf(subint, "cos(%d*acos(x))*(%s)/sqrt(1-x^2)", i,
                        function);

                /* Izracunava se tekuci koeficijent. */
                c0[i] =
                    chebyshev(subint,
                              nodes_count) / ((i == 0) ? pi : pi / 2);
        }

        /* Oslobadja se string koriscen za podintegralnu funkciju. */
        free(subint);
}
```

MATLAB

4.17 *Korišćenjem* MATLAB-*a aproksimirati funkciju* e^x *na intervalu* $[0, 1]$ *polinomom stepena* $n = 10$ *u odnosu na*

a) kanonsku bazu polinoma ($1, x, x^2, \ldots$*)*

b) bazu Legendre-ovih polinoma

Rešenje:
a) Napišimo skript koji će biti malo opštiji od traženog, u smislu da će biti moguće menjati funkciju f koja se aproksimira, interval $[a, b]$ na kome se aproksimacija vrši i stepen polinoma n. Rešenje se zasniva na postavljanju i rešavanju sistema jednačina:

$$\begin{pmatrix} \int_a^b x^0x^0dx & \int_a^b x^0x^1dx & \ldots & \int_a^b x^0x^ndx \\ \int_a^b x^1x^0dx & \int_a^b x^1x^1dx & \ldots & \int_a^b x^0x^ndx \\ \ldots & & & \\ \int_a^b x^nx^0dx & \int_a^b x^nx^1dx & \ldots & \int_a^b x^nx^ndx \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \ldots \\ c_n \end{pmatrix} = \begin{pmatrix} \int_a^b f(x)x^0dx \\ \int_a^b f(x)x^1dx \\ \ldots \\ \int_a^b f(x)x^n \end{pmatrix}$$

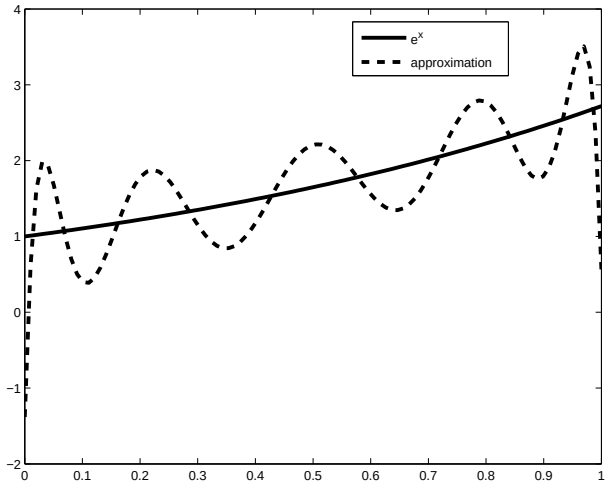
Integrale matrice sistema je moguće unapred izračunati. Ukoliko indeksiranje matrice prilagodimo MATLAB-u, važi da je

$$g_{ij} = \int_a^b x^{i-1}x^{j-1} = \int_a^b x^{i+1-2} = b^{i+j-1} - a^{i+j-1}$$

Vektor slobodnih članova zavisi od funkcije f i u narednoj implementaciji će za izračunavanje odgovarajućih integrala biti korišćena funkcija `quad`.

```
function approx_cannonical
% Postavljamo interval, funkciju i stepen polinoma
a = 0; b = 1;
fun = inline('exp(x)'); n = 10;

% Gradimo matricu G koja predstavlja proizvode <gi,gj>
[i, j] = meshgrid(1:n+1, 1:n+1);
G = (b.^(i+j-1) - a.^(i+j-1)) ./ (i+j-1);

% Gradimo vektor fg koji predstavlja proizvode <f,gj>
fg = zeros(n+1,1);  %prealokacija
for j=1:n+1
    fg(j) = quad(['(' fun ').*x.^' num2str(j-1)], a, b);
end

% Resavamo sistem i obrcemo vetkor tako da odgovara standardnom
% zapisu matlab polinoma
approx_poly = fliplr((G\fg)');

% Iscrtavamo rezultat
X = linspace(a, b);
plot(X, feval(inline(fun), X), X, polyval(approx_poly, X),':');
legend(fun, 'approximation');
```

Rezultat aproksimacije polinomom stepena 10 je prikazan na slici 4.2. Primetimo veliku grešku do koje je došlo zbog nestabilnosti sistema koji je rešavan. Naime, uslovljenost matrice G je približno $5 \cdot 10^{14}$, tako da mala greška do koje je došlo prilikom numeričkog određivanja vektora slobodnih članova izuzetno utiče na krajnji rezultat.



Slika 4.2: Aproksimacija funkcije e^x na intervalu $[0, 1]$ polinomom stepena 10 u odnosu na kanonsku bazu polinoma 1, x, x^2, ...

b) Opisani problem rešavamo korišćenjem ortogonalnih Legendre-ovih polinoma, što ukida potrebu za rešavanjem sistema jednačina. Za konstrukciju Legendre-ovih polinoma korišćena je funkcija `legendre_poly` čija je implementacija opisana u poglavlju o numeričkoj integraciji.

```
function approx_legendre
% Postavljamo interval, funkciju i stepen polinoma
a = 0; b = 1;
fun = inline('exp(x)'); n = 10;

% Konstruisimo bazne polinome
% L je cell array koji sadrzi Lezandrove polinome L{i}=Li-1 na intervalu [a, b]
L = legendre_poly(n+1, a, b);

% koeficijenti aproksimacionog polinoma
approx_poly = zeros(1,n+1); %prealokacija

for i=1:n+1
    % approx_poly += (2/(b-a))*(2i-1)/2)*<f,Li>*Li
    approx_poly = approx_poly + ...
                  [zeros(1,n-i+1) ...
                   (2*i-1)/(b-a) * quad(@Lf,a,b,[],[],fun,L{i}) * L{i}];
end

% iscrtavamo rezultat
X = linspace(a, b);
plot(X, feval(fun, X), X, polyval(approx_poly, X));

% Pomocna funkcija potrebna za racunanje integrala
% Funkcija racuna vrednost proizvoda funcije f i
% odgovarajuceg Legendreovog polinoma l u datoj tacki x
```

```
function y = Lf(x, f, l)
    % l(x)*f(x)
    y = polyval(l, x).*feval(f, x);
```

Primetimo način da se podintegralnoj funkciji `Lf` proslede dodatni parametri. Naime, koristi se poziv funkcije `quad(@Lf,a,b,[],[],fun,Li)`. Opšti oblik ovakvog poziva je `quad(fun,a,b,tol,trace,p1,p2,...)`. Prazne matrice kod parametara `tol` i `trace` označavaju da se koriste podrazumevane vrednosti, dok je kroz parametre `p1`, `p2`, ... moguće podintegralnoj funkciji `fun` proslediti dodatne parametre.

4.2 Metoda najmanjih kvadrata

Metoda najmanjih kvadrata je diskretni analogon metode srednjekvadratne aproksimacije. Skalarni proizvod je određen sumom

$$(f,g) = \sum_{i=0}^{m} p_i f(x_i) g(x_i)$$

Tačke x_i, $i = 0, \ldots, m$, su date tačke i njihov broj je $m \geq n$ (n je stepen generalisanog polinoma). Brojevi $p_i > 0$ su težinski koeficijenti.

4.18 *Trigonometrijskim polinomom* $Q(x) = c_0 + c_1 \cos x$ *aproksimirati funkciju* $f(x) = 1 - \frac{x^2}{\pi^2}$ *metodom najmanjih kvadrata na skupu tačaka* 0, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$ *i* π.

Rešenje: U ovom primeru je $g_0(x) = 1$ i $g_1(x) = \cos(x)$, a koeficijenti najbolje aproksimacije su određeni sistemom jednačina

$$\sum_{k=0}^{1} c_k (g_k, gj) = (f, g_j) \quad j = 0, 1,$$

gde je skalarni proizvod definisan izrazom

$$(f,g) = f(0)g(0) + f(\frac{\pi}{3})g(\frac{\pi}{3}) + f(\frac{\pi}{2})g(\frac{\pi}{2}) + f(\frac{2\pi}{3})g(\frac{2\pi}{3}) + f(\pi)g(\pi).$$

Na taj način dobija se sistem jednačina

$$\begin{aligned}
(1\cdot 1 + 1\cdot 1 + 1\cdot 1 + 1\cdot 1 + 1\cdot 1)c_0^0 + (1\cdot 1 + 1\cdot\frac{1}{2} + 1\cdot 0 - 1\cdot\frac{1}{2} - 1\cdot 1)c_1^0 = \\
= 1\cdot 1 + (1 - \frac{(\frac{\pi}{3})^2}{\pi^2})\cdot 1 + (1 - \frac{(\frac{\pi}{2})^2}{\pi^2})\cdot 1 + (1 - \frac{(2\frac{\pi}{3})^2}{\pi^2})\cdot 1 + (1 - \frac{\pi^2}{\pi^2})\cdot 1 \\
(1\cdot 1 + 1\cdot\frac{1}{2} + 1\cdot 0 - 1\cdot\frac{1}{2} - 1\cdot 1)c_0^0 + (1\cdot 1 + \frac{1}{2}\cdot\frac{1}{2} + 0\cdot 0 + \frac{1}{2}\cdot\frac{1}{2} + 1\cdot 1)c_1^0 = \\
= 1\cdot 1 + (1 - \frac{(\frac{\pi}{3})^2}{\pi^2})\cdot\frac{1}{2} + (1 - \frac{(\frac{\pi}{2})^2}{\pi^2})\cdot 0 - (1 - \frac{(2\frac{\pi}{3})^2}{\pi^2})\cdot\frac{1}{2} - (1 - \frac{\pi^2}{\pi^2})\cdot 1
\end{aligned}$$

čije rešenje je $c_0 = \frac{23}{36} = 0.6389$ i $c_1 = \frac{7}{15} = 0.4667$. Najbolja aproksimacija funkcije $f(x)$ je $Q(x) = 0.6389 + 0.4667 \cos x$.

4.19 *Odrediti najbolju srednjekvadratnu aproksimaciju oblika* $q(x) = c_1 e^x + c_2 e^{-x}$ *funkcije* $f(x)$ *date tabelom*

x	0	0.5	1.0	1.5	2.0	2.5
$f(x)$	5.02	5.21	6.49	9.54	16.02	24.53

Rešenje: Aproksimacija funkcije $f(x)$ traži se u obliku linearne kombinacije funkcija $g_1(x) = e^x$ i $g_2(x) = e^{-x}$. Metodom najmanjih kvadrata koeficijenti c_k, $k = 1, 2$, određuju se kao rešenja sistema

$$\sum_{k=1}^{2} c_k (g_k, g_j) = (f, g_j), \qquad j = 1, 2,$$

pri čemu je skalarni proizvod definisan sumom

$$(f, g) = \sum_{k=0}^{5} f(x_k) g(x_k).$$

Tačke x_k su vrednosti argumenta zadate u tabeli, te kada se izračunaju naznačeni skalarni proizvodi, dobija se sledeći sistem linearnih jednačina:

$$234.20418 c_1 + 6 c_2 = 491.21606$$
$$6 c_1 + 1.57806 c_2 = 16.87784$$

Njegovo rešenje je $c_1 = 2.02016$, $c_2 = 3.01439$, i tražena aproksimacija je

$$q(x) = 2.0202 e^x + 3.0144 e^{-x}.$$

4.20 *Odrediti najbolju srednjekvadratnu aproksimaciju pravom funkcije* $f(x)$ *date tabelom*

x	12	14	16	18	20	22
$f(x)$	8.85	8.31	7.67	7.01	6.39	5.68

Rešenje: Aproksimacija funkcije $f(x)$ traži se u obliku $p_1(x) = c_0 + c_1 x$. Metodom najmanjih kvadrata dobija se sistem linearnih jednačina po koeficijentima c_k, $k = 0, 1$,

$$c_0 \sum_{j=0}^{5} 1 + c_1 \sum_{j=0}^{5} x_j = \sum_{j=0}^{5} f(x_j), \qquad c_0 \sum_{j=0}^{5} x_j + c_1 \sum_{j=0}^{5} x_j^2 = \sum_{j=0}^{5} x_j f(x_j),$$

pa je tražena aproksimacija

$$p_1(x) = 12.7268 - 0.3181 x.$$

4.21 *Metodom najmanjih kvadrata rešiti sistem jednačina*

$$\begin{aligned} x_0 + x_1 &= 3 \\ x_0 + 3x_1 &= 7 \\ 2x_0 - x_1 &= 0.2 \\ 3x_0 + x_1 &= 5 \end{aligned}$$

Rešenje: Dati (preodređeni) sistem jednačina se može zapisati u matričnom obliku kao $A\mathbf{x} = \mathbf{b}$, gde je

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 2 & -1 \\ 3 & 1 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 7 \\ 0.2 \\ 5 \end{pmatrix}$$

Problem se sada može posmatrati i na sledeći način – vektor $\mathbf{b}$ predstavlja zadate vrednosti fukcije f u nekim tačkama, a vektori kolona matrice A predstavljaju zadate vrednosti funkcija g_0 i g_1 u istim tačkama. Rešenje se tako svodi na nalaženje keoficijenata najbolje aproksimacije funkcije f funkcijama g_0 i g_1 metodom najmanjih kvadrata. Sistem jednačina za nalaženje koeficijenata najbolje aproksimacije x_i glasi

$$\sum_{i=0}^{1} x_i(g_i, g_j) = (f, g_j) \quad j = 0, 1,$$

gde je

$$(f, g) = \sum_{k=0}^{3} f_k g_k$$

odnosno, u ovom slučaju,

$$\begin{aligned} (1\cdot 1 + 1\cdot 1 + 2\cdot 2 + 3\cdot 3)x_0 + (1\cdot 1 + 1\cdot 3 - 2\cdot 1 + 3\cdot 1)x_1 \\ = 3\cdot 1 + 7\cdot 1 + 0.2\cdot 2 + 5\cdot 3 \\ (1\cdot 1 + 1\cdot 3 - 2\cdot 1 + 3\cdot 1)x_0 + (1\cdot 1 + 3\cdot 3 + 1\cdot 1 + 1\cdot 1)x_1 \\ = 3\cdot 1 + 7\cdot 3 - 0.2\cdot 1 + 5\cdot 1 \end{aligned}$$

Sređivanjem ovih izraza dobijamo traženo rešenje

$$\begin{aligned} 15x_0 + 5x_1 &= 25.4 \\ 5x_0 + 12x_1 &= 28.8 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} x_0 &= 1.04 \\ x_1 &= 1.97 \end{aligned}$$

MATLAB

Metoda najmanjih kvadrata :	`p = polyfit(x, y, n)`
Rešavanje preodređenog sistema $Ax = b$:	`x = A\b`

Aproksimacija algebarskim polinomima metodom najmanjih kvadrata je u MATLAB-u direktno implementirana kroz funkciju `p = polyfit(x, y, n)`. Argumenti funkcije su vektori `x` i `y` koji predstavljaju poznate vrednosti funkcije koja se aproksimira i stepen aproksimacionog polinoma `n`. Funkcija vraća koeficijente dobijenog polinoma `p`.

4.22 *Korišćenjem* MATLAB-*a metodom najmanjih kvadrata, polinomom trećeg stepena aproksimirati funkciju zadatu tabelom*

1	2	3	4	5	6
−1	3	2	−3	2	1

Rešenje:

```
% Tablica aproksimacije
x = [1 2 3 4 5 6];
y = [-1 3 2 -3 2 1];

% Vrsimo aproksimaciju polinomom treceg stepena
p = polyfit(x, y, 3)

% Iscrtavamo rezultat
xx = linspace(1, 6);
plot(x, y, 'o', xx, polyval(p, xx));
```

4.23 *Korišćenjem* MATLAB-*a pronaći pravu koja najbolje srednjekvadratno aproksimira skup tačaka zadat tabelom*

x_i	1	2	3	4	5	6	7	8	9	10
y_i	3.81	7.73	9.41	15.18	15.86	19.11	23.07	25.06	27.90	30.17

Rešenje: Zadatak ćemo uraditi na tri načina. Ukoliko pravu tražimo u obliku $y = kx + n$ i postavimo uslov da prava prolazi kroz sve date tačke, dobija se preodređeni sistem jednačina

$$y_i = kx_i + n, \qquad i = 1, \ldots, 10$$

Pokažimo opšti metod rešavanja preodređenih sistema jednačina. Neka je sistem zapisan matrično

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \ldots & & & \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}$$

odnosno

$$A\mathbf{x} = \mathbf{b} \quad \text{tj.} \quad (\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m})\,\mathbf{x} = \mathbf{b} \quad \text{tj.} \quad x_1\mathbf{a_1} + x_2\mathbf{a_2} + \ldots + x_m\mathbf{a_m} = \mathbf{b}$$

Posmatrajmo ovaj problem kao nalaženje elementa najbolje aproksimacije vektora $\mathbf{b}$ preko vektora $\mathbf{a_i}$, $i = 1, \ldots, m$ u Hilbertovom prostoru $\mathbb{R}^n$. Na osnovu teoreme o najboljoj aproksimaciji u Hilbertovim prostorima potrebno je rešiti sistem

$$\begin{pmatrix} (\mathbf{a_1}, \mathbf{a_1}) & (\mathbf{a_1}, \mathbf{a_2}) & \ldots & (\mathbf{a_1}, \mathbf{a_m}) \\ (\mathbf{a_2}, \mathbf{a_1}) & (\mathbf{a_2}, \mathbf{a_2}) & \ldots & (\mathbf{a_2}, \mathbf{a_m}) \\ \ldots & & & \\ (\mathbf{a_m}, \mathbf{a_1}) & (\mathbf{a_1}, \mathbf{a_2}) & \ldots & (\mathbf{a_m}, \mathbf{a_m}) \end{pmatrix} \mathbf{x} = \begin{matrix} (\mathbf{a_1}, \mathbf{b}) \\ (\mathbf{a_1}, \mathbf{b}) \\ \ldots \\ (\mathbf{a_m}, \mathbf{b}) \end{matrix}$$

Ovo je upravo sistem koji se dobija kada se i leva i desna strana polaznog sistema pomnože transponovanom matricom sistema

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Naredni skript upravo navedenom metodom rešava zadatak

```
x = 1:10;
y = [3.81 7.73 9.41 15.18 15.86 19.11 23.07 25.06 27.90 30.17];

% Gradimo matricu sistema i vektor slobodnih clanova
A = [x' ones(size(x'))];
b = y';

% Gradimo pridruzeni regularni sistem i resavamo ga
coeffs = (A'*A)\(A'*b)

% Citamo i iscrtavamo resenje
k = coeffs(1); n = coeffs(2);

X = linspace(1, 10);
plot(x, y, 'o', X, k*X+n);
```

Pokretanjem skripta dobija se da je tražena prava $y = 2.9309x + 1.61$.
Mana ovog rešenja je da se množenjem transponovanom matricom sistema, kvadratno uvećava uslovljenost matrice i novi sistem je znatno nestabilniji od polaznog.

Drugo rešenje koje ćemo prikazati se zasniva na činjenici da MATLAB-ov operator \ za rešavanje sistema preodređene sisteme rešava tako da je dobijeno rešenje upravo najbolje u srednje-kvadratnom smislu. Ovo znači da je u prethodnom skriptu liniju

```
coeffs = (A'*A)\(A'*b)
```

bilo moguće zameniti linijom

```
coeffs = A\b
```

Operator \ se zasniva na znatno sofisticiranijim metodama od množenja transponovanom matricom tako da se njegova upotreba, svakako, preporučuje.

Treće, najkraće, rešenje se zasniva na korišćenju funkcije `polyfit` koja direktno vrši srednje kvadratnu aproksimaciju polinomom datog stepena.

```
x = 1:10;
y = [3.81 7.73 9.41 15.18 15.86 19.11 23.07 25.06 27.90 30.17];

%Interpoliramo tacke pravom
coeffs = polyfit(x, y, 1);

% Citamo i iscrtavamo resenje
k = coeffs(1); n = coeffs(2);

X = linspace(1, 10);
plot(x, y, 'o', X, k*X+n);
```

4.24 *Korišćenjem* MATLAB-*a pronaći kružnicu koja najbolje srednje-kvadratno aproksimira tačke*

x_i	2.9504	2.4611	1.5954	0.4624	−0.8108	−1.1109	...
y_i	−1.0180	0.2064	1.0038	0.7603	−0.0168	−1.3182	...
		...	−0.4580	0.8271	1.8606	2.6619	
		...	−2.4192	−2.7511	−2.8295	−1.9809	

Rešenje: Posmatrajmo jednačinu kružnice sa centrom u tački (a, b) i poluprečnikom r,

$$(x-a)^2 + (y-b)^2 = r^2.$$

Ukoliko želimo da kružnica sadrži sve navedene tačke (x_i, y_i) dobijamo preodređeni sistem jednačina, koji nije linearan po nepoznatim veličinama a, b i r. Međutim, prezapisivanjem jednačina sistema u obliku

$$x^2 - 2ax + a^2 + y^2 - 2by + b^2 = r^2,$$

odnosno

$$2ax + 2by + r^2 - a^2 - b^2 = x^2 + y^2,$$

i uvođenjem smene $c = r^2 - a^2 - b^2$, dobijamo preodređeni linearni sistem po nepoznatim a, b i c,

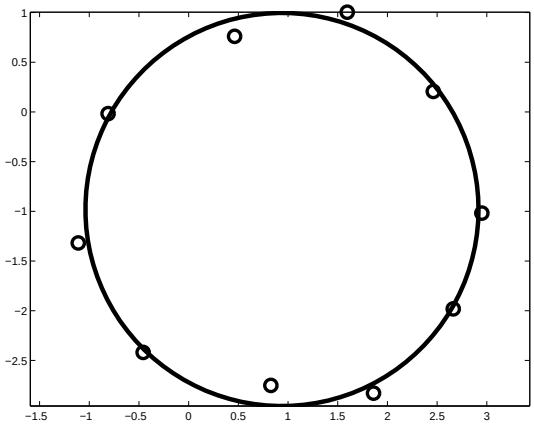
$$2x_i a + 2y_i b + c = x_i^2 + y_i^2, \quad i = 1, \ldots, 10,$$

koji se rešava na ranije opisan način.

```
% Polazni podaci
x = [2.9504 2.4611  1.5954  0.4624  -0.8108  -1.1109  -0.4580  0.8271  1.8606  2.6619];
y = [-1.0180  0.2064  1.0038  0.7603  -0.0168  -1.3182 -2.4192  -2.7511  -2.8295  -1.9809];

% Gradimo preodredjeni sistem
A = [2*x' 2*y' ones(size(x))'];
b = x'.^2 + y'.^2;

% Resavamo ga
coeffs = A\b;

% Izdvajamo centar i racunamo poluprecnik kruznice
a = coeffs(1), b = coeffs(2), c = coeffs(3);
```

```
r = sqrt(c+a^2+b^2)

% Iscrtavamo polazne tacke i kruznicu
X = linspace(0, 2*pi);
plot(x, y, 'ro', a + r*cos(X), b + r*sin(X), 'b'), axis equal;
```

Rezultat rada skripta je prikazan na slici 4.3



Slika 4.3: Srednjekvadratna aproksimacija tačaka kružnicom

4.3 Fourier-ov red

Fourier-ov red funkcije $f(x)$ po trigonometrijskim funkcijama je

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx)$$

$$a_k = (f, \cos kx) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx\, dx, \qquad b_k = (f, \sin kx) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx\, dx$$

ili, u kompleksnom obliku,

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{-\imath kx}, \qquad c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{\imath kx}\, dx$$

Parseval-ova jednakost

$$(f, f) = \left(\frac{a_0}{2}\right)^2 + \sum_{k=1}^{\infty}(a_k^2 + b_k^2)$$

4.25 *Korišćenjem Fourier-ovog reda za funkciju* $f(x) = \cos \alpha x$, α *nije ceo broj, po sistemu funkcija* $\{\cos nx\}$ *na intervalu* $[-\pi, \pi]$, *dokazati da je*

$$\sum_{n=1}^{\infty} \frac{2\alpha}{n^2 - \alpha^2} = \frac{1}{\alpha} - \pi \cot \pi\alpha.$$

Rešenje: Da bismo dokazali tvrđenje, pođimo od Fourier-ovog razvoja

$$\cos \alpha x = \sum_{n=0}^{\infty} a_n \cos nx, \qquad a_n = \frac{2}{\pi} \int_0^{\pi} \cos \alpha x \cos nx \, dx.$$

Koeficijenti a_n su

$$a_0 = \frac{1}{\alpha\pi} \sin \alpha\pi, \qquad a_n = \frac{(-1)^n}{\pi} \frac{2\alpha}{\alpha^2 - n^2} \sin \alpha\pi, \quad n = 1, 2, \ldots,$$

te je

$$\cos \alpha x = \frac{1}{\alpha\pi} \sin \alpha\pi + \sum_{n=1}^{\infty} \frac{(-1)^n}{\pi} \frac{2\alpha}{\alpha^2 - n^2} \sin \alpha\pi \cos nx.$$

Stavimo u poslednjem izrazu da je $x = \pi$, i množenjem sa $\pi / \sin \alpha\pi$ dobijamo traženi izraz.

4.26 *a)Odrediti Fourier-ov red step funkcije*

$$f(x) = \begin{cases} -1, & x \in [-\pi, 0), \\ 1, & x \in (0, \pi] \end{cases}.$$

b)Korišćenjem rezultata pod a), dokazati da je

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + \frac{(-1)^n}{2n+1} + \cdots = \frac{\pi}{4}.$$

c)Korišćenjem rezultata dobijenog pod a), dokazati da je i

$$1 + \frac{1}{9} + \frac{1}{25} + \cdots + \frac{1}{(2n+1)^2} + \cdots = \frac{\pi^2}{8}.$$

Rešenje: a) Koeficijenti Fourier-ovog reda koji množe funkcije $\cos nx$ su $a_n = 0$ $n = 0, 1, \ldots$, jer je funkcija $f(x)$ neparna. Koeficijenti kojima se množe funkcije $\sin nx$ su $b_n = 2(1 - (-1)^n)/(n\pi)$, $n = 1, 2, \ldots$, te je

$$(*) \qquad f(x) = \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{1}{2n+1} \sin (2n+1)x.$$

b) Zamenom $x = \pi/2$ u $(*)$ imamo da je

$$f(\frac{\pi}{2}) = 1 = \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1},$$

odakle sledi tvrđenje.

c) Sistem funkcija $\{\sin kx\}_k$ postaje ortonormiran na intervalu $[-\pi,\pi]$ deljenjem bazisnih funkcija njihovom normom $\sqrt{\pi}$. Fourier-ov red step funkcije po ovom ortonormiranom sistemu je

$$f(x)=\frac{4}{\pi}\sum_{n=0}^{\infty}\frac{\sqrt{\pi}}{2n+1}\,\frac{\sin(2n+1)x}{\sqrt{\pi}},$$

i Fourier-ovi koeficijenti su jednaki $\overline{a}_k=0$ i $\overline{b}_k=\frac{\sqrt{\pi}}{2n+1}$. Na osnovu Parseval-ove jednakosti tvrđenje neposredno sledi, jer je

$$(f,f)=\int_{-\pi}^{\pi}dx=2\pi,\qquad (\frac{\overline{a}_0}{2})^2+\sum_{k=1}^{\infty}(\overline{a}_k^2+\overline{b}_k^2)=\frac{16}{\pi}\sum_{k=0}^{\infty}\frac{1}{(2k+1)^2}.$$

4.27 *a) Odrediti Fourier-ov red funkcije $f(x)=|\sin x|$ na intervalu $(-\pi,\pi)$.*
b) Korišćenjem rezultata pod a) dokazati da je

$$\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)(2n+3)}=\frac{\pi}{4}-\frac{1}{2}.$$

c) Korišćenjem rezultata pod a) dokazati da je

$$\sum_{n=0}^{\infty}\frac{1}{(2n+1)(2n+3)}=\frac{1}{2}.$$

Rešenje: a) Funkcija je parna te je njen Fourier-ov red

$$(*)\qquad |\sin x|=\frac{2}{\pi}\left(1-2\sum_{n=1}^{\infty}\frac{1}{4n^2-1}\cos 2nx\right).$$

b) Tvrđenje neposredno sledi zamenom $x=\pi/2$ u $(*)$.

c) Tvrđenje neposredno sledi zamenom $x=\pi$ u $(*)$.

4.28 *Koristeći Fourier-ov red funkcije $f(x)=x$ po sistemu funkcija $\{\sin nx\}_1^{\infty}$ na intervalu $[0,\pi]$, dokazati da je*

$$\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{\pi^2}{6}.$$

Rešenje: Fourier-ov red funkcije $f(x)=x$ je

$$x=\sum_{n=1}^{\infty}b_n\sin nx=-2\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\sin nx,$$

jer je

$$b_n = \frac{2}{\pi}\int_0^\pi x\sin nx\,dx = 2\frac{(-1)^{n+1}}{n}.$$

Primenom Parseval-ove jednakosti

$$\frac{2\pi^2}{3} = (x,x) = \sum_{n=1}^{\infty} b_n^2 = 4\sum_{n=1}^{\infty}\frac{1}{n^2}$$

tvrđenje neposredno sledi.

4.29 *Dokazati da je*

$$\int_{-\pi}^{\pi}\big(f(x)-Q(x)\big)^2\,dx = \int_{-\pi}^{\pi} f^2(x)\,dx - \pi\left(\frac{1}{2}a_0^2 + \sum_{k=1}^{n}(a_k^2+b_k^2)\right),$$

gde je

$$Q(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n}(a_k\cos kx + b_k\sin kx),$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx\,dx, \qquad b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx\,dx,$$

polinom najbolje srednjekvadratne aproksimacije funkcije $f(x)$ po sistemu funkcija $\cos kx$, $\sin kx$, $k = 0,\dots,n$.

Rešenje: Zbog ortogonalnosti ovog sistema funkcija

$$(\sin mx,\sin nx) = \begin{cases} 0, & m\neq n,\\ \pi, & m=n\end{cases}, \qquad (\cos mx,\cos nx) = \begin{cases} 0, & m\neq n,\\ \pi, & m=n\neq 0,\\ 2\pi, & m=n=0\end{cases}$$

$$(\sin mx,\cos nx) = 0,$$

sledi da je

$$\begin{aligned}\int_{-\pi}^{\pi} f(x)Q(x)\,dx &= \frac{1}{2}a_0\int_{-\pi}^{\pi} f(x)\,dx\\ &\quad + \sum_{k=1}^{n}\left(a_k\int_{-\pi}^{\pi} f(x)\cos kx\,dx + b_k\int_{-\pi}^{\pi} f(x)\sin kx\,dx\right)\\ &= \pi\left(\frac{1}{2}a_0^2 + \sum_{k=1}^{n}(a_k^2+b_k^2)\right)\end{aligned}$$

i

$$\int_{-\pi}^{\pi} Q^2(x)\,dx = \frac{1}{4}a_0^2\int_{-\pi}^{\pi} dx + \sum_{k=1}^{n}\left(a_k^2\int_{-\pi}^{\pi}\cos^2 kx\,dx + b_k^2\int_{-\pi}^{\pi}\sin^2 kx\,dx\right) = \pi\left(\frac{1}{2}a_0^2 + \sum_{k=1}^{n}(a_k^2+b_k^2)\right)$$

Sada tvrđenje neposredno sledi zamenom dobijenih izraza u izrazu

$$\int_{-\pi}^{\pi}\big(f(x)-Q(x)\big)^2\,dx = \int_{-\pi}^{\pi} f^2(x) - 2\int_{-\pi}^{\pi} f(x)\,Q(x)\,dx + \int_{-\pi}^{\pi} Q^2(x)\,dx.$$

4.30 *Naći Fourier-ov red funkcije $f(x)=e^x$ na intervalu $[-\pi,\pi]$.*

Rešenje: Koeficijenti Fourier-ovog reda ove funkcije su

$$\begin{aligned} c_k &= \tfrac{1}{2\pi}\textstyle\int_{-\pi}^{\pi} e^x e^{\imath kx}dx = \tfrac{1}{2\pi}\tfrac{1}{1+\imath k}e^{(1+\imath k)x}\Big|_{-\pi}^{\pi} \\ &= \tfrac{1}{2\pi}\tfrac{1}{1+\imath k}(e^{\pi}(-1)^k - e^{-\pi}(-1)^k) = \\ &= \tfrac{1}{2\pi}\tfrac{(-1)^k}{1+\imath k}(e^{\pi}-e^{-\pi}) = \tfrac{(-1)^k}{1+\imath k}\,\tfrac{\sinh\pi}{\pi} \end{aligned}$$

Dakle, Fourier-ov red funkcije e^x na intervalu $[-\pi,\pi]$ je

$$f(x) = \frac{\sinh\pi}{\pi}\sum_{k=-\infty}^{\infty}\frac{(-1)^k}{1+\imath k}e^{-\imath kx}$$

4.31 *Na intervalu $[-\pi,\pi]$ naći Fourier-ov red Dirac-ove δ funkcije:*

$$\delta(x) = \begin{cases} \infty & x=0 \\ 0 & x\neq 0 \end{cases}$$

Rešenje: Funkcija $\delta(x)$ (tzv. Dirac-ov impuls) je generalisana funkcija i definisana je svojim dejstvom na druge funkcije,

$$\int_{-\infty}^{\infty}\delta(x-a)f(x)dx = f(a) \qquad \overset{f(x)\equiv 1}{\longrightarrow} \qquad \int_{-\infty}^{\infty}\delta(x-a)dx = 1$$

Koeficijenti Fourier-ovog reda ove funkcije su

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi}\delta(x)e^{\imath kx}dx = \frac{1}{2\pi}\int_{-\infty}^{\infty}\delta(x)e^{\imath kx}dx = \frac{1}{2\pi}\,e^0 = \frac{1}{2\pi}\,,$$

pa je njen Fourier-ov red

$$f(x) = \frac{1}{2\pi}\sum_{k=-\infty}^{\infty}e^{-\imath kx}$$

4.32 *a) Naći norme vektora* $\mathbf{u} = (1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots)^\top$ *i* $\mathbf{v} = (1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}, \dots)^\top$ *u Hilbertovom prostoru* l *i proveriti Schwartz-ovu nejednakost*

$$(\mathbf{u}, \mathbf{v})^2 \leq (\mathbf{u}, \mathbf{u})\,(\mathbf{v}, \mathbf{v}).$$

b)Za funkcije

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{2^n} e^{\imath n x} \qquad i \qquad g(x) = \sum_{n=0}^{\infty} \frac{1}{3^n} e^{\imath n x},$$

koristeći rezultate pod a), dokazati nejednakost

$$|(f, g)|^2 \leq (f, f)\,(g, g) \qquad gde\ je \quad (f, g) = \int_{-\pi}^{\pi} f(x)\overline{g}(x)\,dx.$$

Rešenje: a) Kako je u prostoru l skalarni proizvod $(\mathbf{u}, \mathbf{v}) = \mathbf{v}^\top \mathbf{u}$, to je

$$(\mathbf{u}, \mathbf{u}) = \sum_{n=0}^{\infty} \left(\frac{1}{2^n}\right)^2 = \sum_{n=0}^{\infty} \frac{1}{4^n} = \frac{4}{3}, \qquad (\mathbf{v}, \mathbf{v}) = \sum_{n=0}^{\infty} \left(\frac{1}{3^n}\right)^2 = \sum_{n=0}^{\infty} \frac{1}{9^n} = \frac{9}{8},$$

$$(\mathbf{u}, \mathbf{v}) = \sum_{n=0}^{\infty} \frac{1}{2^n}\,\frac{1}{3^n} = \sum_{n=0}^{\infty} \frac{1}{6^n} = \frac{6}{5},$$

odakle tvrđenje neposredno sledi.

b) Sistem funkcija $\{e^{\imath n x}\}$ na intervalu $[-\pi, \pi]$ je ortonormiran

$$(e^{\imath k x}, e^{\imath n x}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{\imath k x} e^{-\imath n x}\,dx = \begin{cases} 1, & k = n \\ 0, & k \neq n \end{cases}.$$

Stoga, na osnovu dokazanog pod a), tvrđenje neposredno sledi jer je

$$(f, f) = \sum_{k=0}^{\infty} \sum_{n=0}^{\infty} \frac{1}{2^k}\,\frac{1}{2^n} (e^{\imath k x}, e^{\imath n x}) = \frac{4}{3}$$

$$(g, g) = \sum_{k=0}^{\infty} \sum_{n=0}^{\infty} \frac{1}{3^k}\,\frac{1}{3^n} (e^{\imath k x}, e^{\imath n x}) = \frac{9}{8},$$

a takođe i

$$(f, g) = \sum_{k=0}^{\infty} \sum_{n=0}^{\infty} \frac{1}{2^k}\,\frac{1}{3^n} (e^{\imath k x}, e^{\imath n x}) = \frac{6}{5}.$$

4.33 *Odrediti kompleksni oblik Fourier-ovog reda sa periodom $2T$ za funkcije*

(a) $f(t) = 1$, (b) $f(t) = e^{at}$,

(c) $f(t) = \cos t$, (d) $f(t) = \begin{cases} -1, & -T < t < 0, \\ 1 & 0 < t < T \end{cases}$.

Rešenje: Prvo ćemo smenom $t = Tx/\pi$ preslikati interval $[-T, T]$ na interval $[-\pi, \pi]$ i definisati $f(t) = f(Tx/\pi) = F(x)$. Za 2π-periodičnu funkciju $F(x)$ Fourier-ov red u kompleksnom obliku je

$$F(x) \sim \sum_{k=-\infty}^{\infty} c_k e^{-\imath kx}, \qquad c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(x) e^{\imath kx} dx.$$

Vraćajući se u poslednjem izrazu na staru promenljivu t dobijamo izraz za Fourier-ov red funkcije $f(t)$ na proizvoljnom intervalu $[-T, T]$,

$$f(t) \sim \sum_{k=-\infty}^{\infty} c_k e^{(-\imath k\pi t/T)}, \qquad c_k = \frac{1}{2T} \int_{-T}^{T} f(t) e^{(\imath k\pi t/T)}\, dt.$$

Sada se direktno korišćenjem poslednje formule nalaze traženi redovi

(a) $f(t) \equiv 1$ jer je $c_0 = 1, \quad c_k = 0,\ k = \pm 1, \pm 2, \ldots,$

(b) $$f(t) \sim \sinh(aT) \sum_{k=-\infty}^{\infty} \frac{(-1)^k}{aT + \imath k\pi} e^{(-\imath k\pi t/T)}$$

(c) $$f(t) \sim T \sin T \sum_{k=-\infty}^{\infty} \frac{(-1)^k}{T^2 - k^2\pi^2} e^{(-\imath k\pi t/T)}$$

(d) $$f(t) \sim \frac{2\imath}{\pi} \sum_{k=-\infty}^{\infty} \frac{1}{2k+1} e^{(-\imath(2k+1)\pi t/T)}$$

4.34 *Napisati Fourier-ov red na intervalu $(0, T)$ u odnosu na bazisne funkcije $\left\{\sin \frac{n\pi t}{T}\right\}_1^\infty$ za sledeće funkcije*

(a) $f(t) = \begin{cases} t(T-t), & t > 0 \\ 0, & t < 0 \end{cases}$ (b) $f(t) = \begin{cases} T - t, & t > T/2 \\ t, & 0 < t < T/2 \\ 0, & t < 0 \end{cases}$

(c) $f(t) = \begin{cases} 0, & t > T/2 \\ 1, & t < T/2 \end{cases}$ (d) $f(t) = \sin \dfrac{\pi t}{2T}$,

(e) $f(t) = \begin{cases} \sin(\pi t/T), & 0 < t < T/2 \\ 0, & \textit{inače} \end{cases}$ (f) $f(t) = \begin{cases} 1/\varepsilon, & 0 < t < \varepsilon < T, \\ 0, & \textit{inače} \end{cases}$

Rešenje: Traženi red je srednjekvadratna aproksimacija funkcije $f(t)$ po zadatom sistemu funkcija

$$f(t) \sim \sum_{k=1}^{\infty} c_k \sin \frac{k\pi t}{T}.$$

Kako je sistem funkcija ortonormiran na intervalu $[0, T]$

$$\left(\sin \frac{j\pi t}{T}, \sin \frac{k\pi t}{T}\right) = \frac{2}{T}\int_0^T \sin \frac{j\pi t}{T} \sin \frac{k\pi t}{T}\, dt = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}$$

koeficijenti c_k se određuju kao rešenja sistema linearnih jednačina sa dijagonalnom matricom

$$c_k = \frac{(f, \sin k\pi t/T)}{(\sin k\pi t/T, \sin k\pi t/T)} = \frac{2}{T}\int_0^T f(t) \sin \frac{k\pi t}{T}\, dt = \frac{2}{\pi}\int_0^{\pi} f\big(\frac{T}{\pi}x\big) \sin kx\, dx$$

Tako se dobijaju rešenja

(a) $$f(t) \sim \frac{8T^2}{\pi^3} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^3} \sin \frac{(2k+1)\pi t}{T}$$

(b) $$f(t) \sim \frac{4T}{\pi^2} \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^2} \sin \frac{(2k+1)\pi t}{T}$$

(c) $$f(t) \sim \frac{2}{\pi}\left(\sum_{k=0}^{\infty} \frac{1}{2k+1} \sin \frac{(2k+1)\pi t}{T} + \sum_{k=1}^{\infty} \frac{1}{2k+1} \sin \frac{2(2k+1)\pi t}{T}\right)$$

(d) $$f(t) \sim \frac{1}{\pi} \sum_{k=1}^{\infty} \frac{(-1)^{k+1} 8k}{4k^2 - 1} \sin \frac{k\pi t}{T}$$

(e) $$f(t) \sim \frac{1}{2} \sin \frac{\pi t}{T} + \frac{1}{\pi} \sum_{k=1}^{\infty} \frac{(-1)^{k+1} 4k}{4k^2 - 1} \sin \frac{2k\pi t}{T}$$

(f) $$f(t) \sim \frac{4}{\varepsilon\pi} \sum_{k=1}^{\infty} \frac{1}{k} \sin^2 \frac{k\varepsilon\pi}{2T} \sin \frac{k\pi t}{T}$$

4.4 Diskretna Fourier-ova transformacija

Diskretna Fourier-ova transformacija je algoritam kojim se n-dimenzioni vektor $\mathbf{c} = (c_0, \ldots, c_{n-1})^\top$ koeficijenata parcijalne sume Fourier-ovog reda određuje tako da je u ekvidistantnim tačkama $\frac{2\pi j}{n}$, $j = 0, \ldots, n-1$, intervala $[0, 2\pi]$ ova parcijalna suma jednaka vrednostima funkcije f u tim tačkama:

$$f(x_j) = \sum_{k=0}^{n-1} c_k e^{-\imath k \frac{2\pi j}{n}}, \qquad j = 0, \ldots, n-1.$$

Ako označimo sa $f_j = f(x_j)$, vektor $\mathbf{f} = (f_0, \ldots, f_{n-1})^T$ i $W_n = e^{\imath \frac{2\pi}{n}} = \sqrt[n]{e^{\imath 2\pi}}$, dobijamo formule koje definišu *diskretnu* i *inverznu diskretnu* Fourier-ovu transformaciju,

$$\mathbf{c} = \frac{1}{n} F_n \mathbf{f}, \qquad \mathbf{f} = F_n^* \mathbf{c}.$$

$F_n = \{W_n^{jk}\}$, $j, k = 0, \ldots, n-1$, je Fourier-ova matrica dimenzije n.

Brza Fourier-ova transformacija (FFT) je algoritam za efikasno računanje diskretne Fourier-ove transformacije: n-dimenzioni vektor $\overline{\mathbf{c}} = F_n \mathbf{f}$ određuje se pomoću dva $m = n/2$-dimenziona vektora $\overline{\mathbf{c}}^e$ i $\overline{\mathbf{c}}^o$,

$$\overline{\mathbf{c}}^e = F_m \mathbf{f}^e, \qquad \overline{\mathbf{c}}^o = F_m \mathbf{f}^o,$$

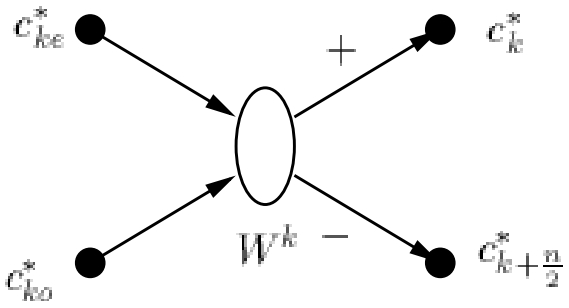
gde je $\mathbf{f}^e = (f_0, f_2, \ldots, f_{n-2})^\top$ i $\mathbf{f}^o = (f_1, f_3, \ldots, f_{n-1})^\top$. Komponente vektora $\overline{\mathbf{c}}$ se računaju po formulama

$$\overline{c}_j = \overline{c}_j^e + W_n^j \overline{c}_j^o, \quad \overline{c}_{m+j} = \overline{c}_j^e - W_n^j \overline{c}_j^o, \qquad j = 0, \ldots, m-1.$$

Vektor $\overline{\mathbf{c}}$ treba još pomnožiti sa $\frac{1}{n}$ da bi predstavljao diskretnu Fourierovu transformaciju vektora $\mathbf{f}$.

Dakle, ukoliko su poznati koeficijenti diskretne Fourier-ove transformacije vektora parnih, odn. neparnih elemenata polaznog vektora, moguće je kombinovanjem odgovarajućih elemenata izračunati koeficijente Fourier-ove transformacije polaznog vektora. Pritom, to kombinovanje se može predstaviti tzv. "butterfly" strukturom koja je prikazana na slici 4.4.

Ukoliko je npr. $n = 8$, tada se koeficijenti $\overline{c}_k$ Fourier-ove transformacije vektora $\mathbf{f} = [f_0\ f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7]^T$ mogu izračunati na osnovu vektora $\overline{\mathbf{c}}^{\mathbf{e}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^e} = [f_0\ f_2\ f_4\ f_6]^T$ i vektora $\overline{\mathbf{c}}^{\mathbf{o}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^o} = [f_1\ f_3\ f_5\ f_7]^T$. Analogno, vektor $\overline{\mathbf{c}}^{\mathbf{e}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^e}$ može se izračunati na osnovu vektora $\overline{\mathbf{c}}^{\mathbf{ee}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^{ee}} = [f_0\ f_4]^T$ i vektora $\overline{\mathbf{c}}^{\mathbf{oe}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^{oe}} = [f_2\ f_6]^T$, dok se vektor $\overline{\mathbf{c}}^{\mathbf{o}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f^o}$ može izračunati na osnovu vek-



Slika 4.4: Butterfly struktura.

tora $\overline{\mathbf{c}}^{\mathbf{eo}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f}^{\mathbf{eo}} = [f_1\ f_5]^T$ i vektora $\overline{\mathbf{c}}^{\mathbf{oo}}$ koeficijenata Fourier-ove transformacije vektora $\mathbf{f}^{\mathbf{oe}} = [f_3\ f_7]^T$. Dalje, na isti način svaki od vektora $\overline{\mathbf{c}}^{\mathbf{ee}}$, $\overline{\mathbf{c}}^{\mathbf{oe}}$, $\overline{\mathbf{c}}^{\mathbf{eo}}$ i $\overline{\mathbf{c}}^{\mathbf{oo}}$ se može izračunati kao kombinacija Fourier-ovih transformacija vektora $\overline{\mathbf{c}}^{\mathbf{eee}}$ i $\overline{\mathbf{c}}^{\mathbf{oee}}$, $\overline{\mathbf{c}}^{\mathbf{eoe}}$ i $\overline{\mathbf{c}}^{\mathbf{ooe}}$, $\overline{\mathbf{c}}^{\mathbf{eeo}}$ i $\overline{\mathbf{c}}^{\mathbf{oeo}}$, te $\overline{\mathbf{c}}^{\mathbf{eoo}}$ i $\overline{\mathbf{c}}^{\mathbf{ooo}}$, koje su zapravo dužine 1 i jednake su odgovarajućim elementima polaznog vektora $\mathbf{f}$, respektivno f_0, f_4, f_2, f_6, f_1, f_5, f_3 i f_7. Čitava šema izračunavanja se može predstaviti slikom 4.5.

Ovakva metoda izračunavanja diskretne Fourier-ove transformacije ima kompleksnost $O(n \cdot \log(n))$ (za razliku od direktnog izračunavanja čija je kompleksnost $O(n^2)$) i označava se kao brza Fourier-ova transformacija (FFT). Pokazuje se da se početni redosled elemenata formira tako što se svaki element vektora $\mathbf{f}$ postavi na mesto koje je određeno njegovim indeksom $ee\ldots o$. Pri tome se indeks prevede u binarni sistem smenom $e = 1$ i $o = 0$ i pročita obrnutim redom (tj. bit po bit zdesna ulevo).

4.35 *Odrediti diskretnu Fourier-ovu transformaciju vektora* $\mathbf{f} = [1\ 0\ 1\ 0]^T$.

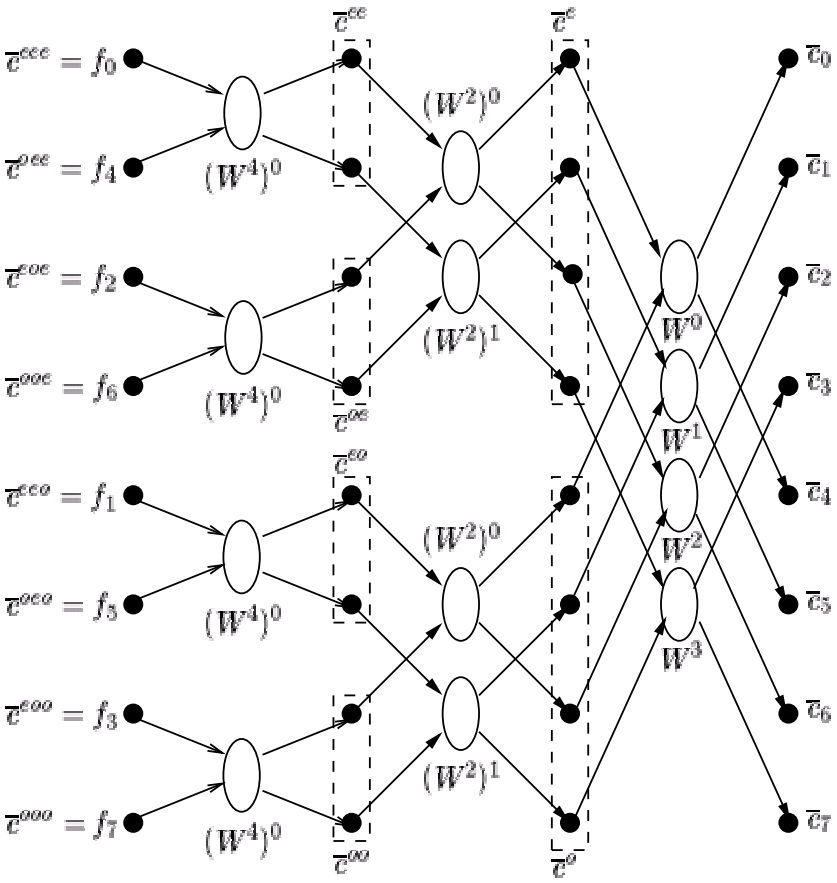
Rešenje: Ovde je:

$$n = 4 \Rightarrow W = e^{i\frac{2\pi}{4}} = e^{i\frac{\pi}{2}} = i$$

i dalje:

$$\begin{aligned}
k = 0: &\quad c_0 = \tfrac{1}{4}(f_0W^0 + f_1W^0 + f_2W^0 + f_3W^0) = \tfrac{1}{4}(1 + 0 + 1 + 0) = \tfrac{1}{2}\\
k = 1: &\quad c_1 = \tfrac{1}{4}(f_0W^0 + f_1W^1 + f_2W^2 + f_3W^3) = \tfrac{1}{4}(1 + 0 - 1 + 0) = 0\\
k = 2: &\quad c_2 = \tfrac{1}{4}(f_0W^0 + f_1W^2 + f_2W^4 + f_3W^6) = \tfrac{1}{4}(1 + 0 + 1 + 0) = \tfrac{1}{2}\\
k = 3: &\quad c_3 = \tfrac{1}{4}(f_0W^0 + f_1W^3 + f_2W^6 + f_3W^9) = \tfrac{1}{4}(1 + 0 - 1 + 0) = 0
\end{aligned}$$

Dakle, diskretna Fourier-ova transformacija datog vektora je $\mathbf{c} = [\frac{1}{2}\ 0\ \frac{1}{2}\ 0]^T$.



Slika 4.5: Šema izračunavanja kod brze Fourier-ove transformacije.

4.36 *Dokazati da je Fourier-ova transformacija diskretne konvolucije n-dimenzionih vektora* $\mathbf{f}$ *i* $\mathbf{g}$ *jednaka proizvodu koordinatu po koordinatu njihovih diskretnih Fourier-ovih transformacija pomnoženih sa* n.

Rešenje: Neka je $\mathbf{f} = \begin{pmatrix} f_0 & \dots & f_{n-1} \end{pmatrix}^\top$ i $\mathbf{g} = \begin{pmatrix} g_0 & \dots & g_{n-1} \end{pmatrix}^\top$; tada je diskretna konvolucija $\mathbf{f} * \mathbf{g}$ vektor dužine n čija je k-ta koordinata jednaka sumi svih proizvoda koordinata vektora $\mathbf{f}$ i $\mathbf{g}$ čija je suma indeksa jednaka k ili $n+k$:

$$\mathbf{f}*\mathbf{g} = \begin{pmatrix} f_0g_0 & + & f_{n-1}g_1 & + & \dots & + & f_1g_{n-1} \\ f_1g_0 & + & f_0g_1 & + & \dots & + & f_2g_{n-1} \\ \dots & & & & & & \\ f_{n-1}g_0 & + & f_{n-2}g_1 & + & \dots & + & f_0g_{n-1} \end{pmatrix} \begin{matrix} \leftarrow \text{zbir indeksa je 0 ili } n \\ \leftarrow \text{zbir in. 1 ili } n+1 \\ \\ \leftarrow \text{zbir in. } n-1 \text{ ili } 2n-1 \end{matrix}$$

Alternativno, diskretna konvolucija se može predstaviti kao proizvod Toeplitz-ove

ciklične matrice vektora $\mathbf{f}$ sa vektorom $\mathbf{g}$:

$$\mathbf{f} * \mathbf{g} = \mathbf{C}\mathbf{g} = \begin{pmatrix} f_0 & f_{n-1} & \dots & f_1 \\ f_1 & f_0 & \dots & f_2 \\ \dots & & & \\ f_{n-1} & f_{n-2} & \dots & f_0 \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \dots \\ g_{n-1} \end{pmatrix}$$

Neka je $\mathbf{F}$ Fourier-ova matrica:

$$\mathbf{F} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & W & \dots & W^{n-1} \\ \dots & & & \\ 1 & W^{n-1} & \dots & W^{(n-1)^2} \end{pmatrix}$$

Tada je $\mathbf{c^f} = \frac{1}{n}\mathbf{F}\mathbf{f}$ Fourier-ova transformacija vektora $\mathbf{f}$ i $\mathbf{c^g} = \frac{1}{n}\mathbf{F}\mathbf{g}$ Fourier-ova transformacija vektora $\mathbf{g}$. Stav koji treba dokazati je $\mathbf{c^{f*g}} = n(\mathbf{c^f c^g})$ gde je sa $(\mathbf{c^f c^g})$ označen proizvod vektora $\mathbf{c^f}$ i $\mathbf{c^g}$ koordinatu po koordinatu. Neka je $\mathbf{F}^*$ konjugovana Fourier-ova matrica:

$$\mathbf{F}^* = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \overline{W} & \dots & \overline{W}^{n-1} \\ \dots & & & \\ 1 & \overline{W}^{n-1} & \dots & \overline{W}^{(n-1)^2} \end{pmatrix}$$

Tada je k-ta kolona ove matrice vektor:

$$\overline{\mathbf{w}}_\mathbf{k} = \begin{pmatrix} 1 \\ \overline{W}^k \\ \dots \\ \overline{W}^{(n-1)k} \end{pmatrix}$$

koji je sopstveni vektor ciklične matrice $\mathbf{C}$, tj. važi $\mathbf{C}\overline{\mathbf{w}}_\mathbf{k} = \lambda_k \overline{\mathbf{w}}_\mathbf{k}$. Naime:

$$\mathbf{C}\overline{\mathbf{w}}_\mathbf{k} = \begin{pmatrix} f_0 & + & f_{n-1}\overline{W}^k & + & \dots & + & f_1\overline{W}^{(n-1)k} \\ f_1 & + & f_0\overline{W}^k & + & \dots & + & f_2\overline{W}^{(n-1)k} \\ \dots & & & & & & \\ f_{n-1} & + & f_{n-2}\overline{W}^k & + & \dots & + & f_0\overline{W}^{(n-1)k} \end{pmatrix}$$

Izdvajanjem ipred sume $\overline{W}^{jk}$ ispred j-tog elementa (za svako j) gornje matrice, množenjem sa $W^{nk} = 1$ i daljim sređivanjem dobija se

$$\mathbf{C}\overline{\mathbf{w}}_\mathbf{k} = (f_0 + f_1 W^k + \ldots + f_{n-1} W^{(n-1)k}) \begin{pmatrix} 1 \\ \overline{W}^k \\ \dots \\ \overline{W}^{(n-1)k} \end{pmatrix} = \lambda_k \overline{\mathbf{w}}_\mathbf{k}$$

Pri tome je $\lambda_k = \sum_{j=0}^{n-1} f_j W^{kj} = nc_k^f$, gde je c_k^f k-ta komponenta Fourier-ove transformacije vektora $\mathbf{f}$. Ako se uvede matrica

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & \lambda_{n-1} \end{pmatrix} = \begin{pmatrix} nc_0^f & 0 & \dots & 0 \\ 0 & nc_1^f & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & nc_{n-1}^f \end{pmatrix}$$

može se pisati

$$\mathbf{C}\mathbf{F}^* = \mathbf{F}^*\mathbf{\Lambda} \Rightarrow \mathbf{C}\mathbf{F}^*\mathbf{F} = \mathbf{F}^*\mathbf{\Lambda}\mathbf{F}.$$

Kako je $\mathbf{F}^*\mathbf{F} = n\mathbf{I}$, to je dalje

$$\mathbf{C} = \frac{1}{n}\mathbf{F}^*\mathbf{\Lambda}\mathbf{F},$$

odnosno

$$\mathbf{f} * \mathbf{g} = \mathbf{C}\mathbf{g} = \frac{1}{n}\mathbf{F}^*\mathbf{\Lambda}\mathbf{F}\mathbf{g} = \mathbf{F}^*\mathbf{\Lambda}\frac{1}{n}\mathbf{F}\mathbf{g} = \mathbf{F}^*\mathbf{\Lambda}\mathbf{c}^{\mathbf{g}} = \mathbf{F}^* n(\mathbf{c}^{\mathbf{f}}\mathbf{c}^{\mathbf{g}}).$$

Na desnoj strani je inverzna Fourier-ova transformacija vektora $n(\mathbf{c}^{\mathbf{f}}\mathbf{c}^{\mathbf{g}})$, što je upravo trebalo dokazati.

4.37 *Korišćenjem diskretne Fourier-ove transformacije odrediti proizvod polinoma* $1 + x$ *i* $2 + 3x + x^2$.

Rešenje: Proizvod dva polinoma se može odrediti pomoću diskretne konvolucije. Naime, proizvod polinoma je

$$f(x)\, g(x) = \Big(\sum_{i=0}^{n_f} f_i x^i\Big)\, \Big(\sum_{i=0}^{n_g} g_i x^i\Big) = f_0 g_0 + (f_0 g_1 + f_1 g_0)x + (f_0 g_2 + f_1 g_1 + f_2 g_0)x^2 + \dots,$$

Ako su vektori koeficijenata polinoma dopunjeni nulama do dimenzije koja odgovara stepenu polinoma proizvoda $n = n(f) + n(g) + 1$,

$$\mathbf{f} = [f_0\ f_1\ \dots\ f_{n_f}\ 0 \dots 0]^\top \qquad \mathbf{g} = [g_0\ g_1\ \dots\ g_{n_g}\ 0 \dots 0]^\top,$$

onda su koeficijenti polinoma proizvoda elementi vektora diskretne konvolucije vektora $\mathbf{f}$ i $\mathbf{g}$. Diskretna konvolucija se, pak, efikasno može izračunati pomoću diskretne Fourier-ove transformacije, što je pokazano u prethodnom zadatku.

U datom primeru je $n_f = 1$ i $n_g = 2$, pa je $n = 4$, i $\mathbf{f} = [1\ 1\ 0\ 0]^\top$, $\mathbf{g} = [2\ 3\ 1\ 0]^\top$ i $W = e^{\imath\frac{2\pi}{4}} = \imath$. Fourier-ove transformacije ovih vektora su

$$\mathbf{c}^{\mathbf{f}} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \imath & -1 & -\imath \\ 1 & -1 & 1 & -1 \\ 1 & -\imath & -1 & \imath \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{4}(1+\imath) \\ 0 \\ \frac{1}{4}(1-\imath) \end{pmatrix}$$

$$\mathbf{c^g} = \frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & \imath & -1 & -\imath\\ 1 & -1 & 1 & -1\\ 1 & -\imath & -1 & \imath\end{pmatrix}\begin{pmatrix}2\\3\\1\\0\end{pmatrix} = \begin{pmatrix}\frac{3}{2}\\ \frac{1}{4}(1+3\imath)\\ 0\\ \frac{1}{4}(1-3\imath)\end{pmatrix}$$

Proizvod dobijenih vektora koordinatu po koordinatu pomnožen sa $n = 4$ je vektor

$$n(\mathbf{c^f c^g}) = \begin{pmatrix}3\\ -\frac{1}{2}+\imath\\ 0\\ \frac{1}{2}-\imath\end{pmatrix}$$

Inverzna Fourier-ova transformacija tog vektora, koja predstavlja konvoluciju vektora $\mathbf{f}$ i $\mathbf{g}$, je vektor

$$\mathbf{f} * \mathbf{g} = \begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -\imath & -1 & \imath\\ 1 & -1 & 1 & -1\\ 1 & \imath & -1 & -\imath\end{pmatrix}\begin{pmatrix}3\\ -\frac{1}{2}+\imath\\ 0\\ \frac{1}{2}-\imath\end{pmatrix} = \begin{pmatrix}2\\5\\4\\1\end{pmatrix}$$

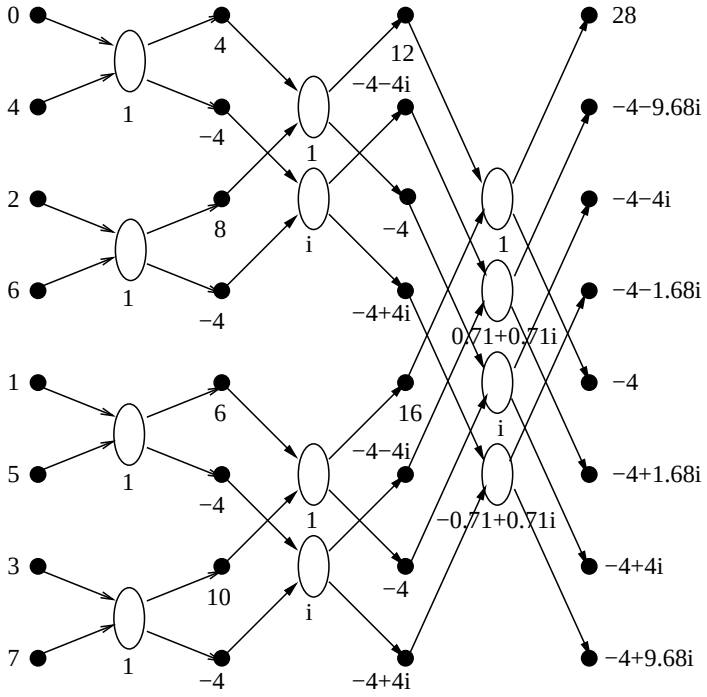
Dakle, proizvod datih polinoma je $2 + 5x + 4x^2 + x^3$.

4.38 *Metodom brze Fourier-ove transformacije odrediti diskretnu Fourier-ovu transformaciju vektora* $\mathbf{f} = \begin{pmatrix}0 & 1 & 2 & 3 & 4 & 5 & 6 & 7\end{pmatrix}^\top$.

Rešenje: Nalaženje koeficijenata $\overline{c}_k$ predstavljeno je na slici 4.6. Kada se dobijeni koeficijenti $\overline{c}_k$ podele sa dimenzijom $n = 8$, dobija se diskretna Fourier-ova transformacija datog vektora

$$\begin{aligned}\mathbf{c} = (&3.5 \quad -0.5-1.21\imath \quad -0.5-0.5\imath \quad -0.5-0.21\imath\\ &-0.5 \quad -0.5+0.21\imath \quad -0.5+0.5\imath \quad -0.5+1.21\imath)^\top.\end{aligned}$$

C Implementacija brze Fourier-ove transformacije je svakako najkompleksnija od svih implementacija numeričkih metoda koje su prezentirane u ovoj zbirci. Sa druge strane, to je verovatno metoda sa kojom će se čitalac često susretati u praksi, pa zato vredi uložiti poseban napor da se njena implementacija razume. Implementacija koristi dve pomoćne procedure. Prva od njih, procedura `shuffle()`, implementira obrtanje potrebnog broja bitova u indeksu datog elementa ulaznog vektora da bi se odredio indeks na koji treba smestiti taj vektor pre započinjanja postupka brze Fourier-ove transformacije. Ova procedura koristi proste bitske operacije da bi okrenula dati broj za specificirani broj bitova i potom vraća preokrenuti indeks. Druga procedura, `complex_powers()`, računa sve potrebne stepene kompleksnog broja W. Ako se analizira FFT algoritam, a pogotovu slika 4.5, može se videti da se u algoritmu koriste svi stepeni broja W do $\frac{n}{2} - 1$, pri čemu se neki od



Slika 4.6: Izračunavanje diskretne Fourier-ove transformacije za dati vektor.

njih koriste više puta. Stoga ima smisla unapred izračunati sve potrebne stepene u jednu tabelu i onda tokom FFT algoritma samo konsultovati ovu tabelu kada neki stepen broja W zatreba prilikom izračunavanja. Procedure `complex_powers()` formira tabelu, odn. preciznije izračunava i smešta u dato polje prvih n stepeni datog kompleksnog broja. Treba uočiti da je, opštosti radi, argument ove procedure koji određuje koliko stepeni datog kompleksnog broja treba izračunati nazvan `n`, da bi kasnije prilikom poziva ove procedure iz `fft()` procedure kao stvarni argument bilo preneto $\frac{n}{2}$, gde je n dužina ulaznog vektora. Treba, takođe, uočiti kako su unutar procedure `complex_powers()` korišćene trigonometrijske transformacije kako bi se minimizovao broj skupih poziva procedura `sin()` odn. `cos()`.

Na kraju sledi implementacije procedure `fft()`. Dužina niza ovoj proceduri se prenosi kao stepen broja 2, tako da se na taj način izbegavaju problemi sa nizovima čija dužina nije stepen dvojke. Procedura radi bez ikakvih pomoćnih polja, tj. na početku procedure se elementi ulaznog vektora prepišu u izlazni vektor, pri čemu se izvrši permutovanje ovih elemenata u skladu sa obrtanjem indeksa,i onda se svaki korak algoritma sprovodi nad izlaznim vektorom. Svaki korak u spoljašnjoj petlji algoritma predstavlja jednu kolonu na slici 4.5, dok unutrašnja petlja algoritma predstavlja jedno "butterfly" izračunavanje. Srednja petlja služi da se prođe kroz sve parove u jednoj koloni sa slike 4.5. U unutrašnjoj petlji izračunavanja su

uređena tako da se minimizuje broj potrebnih računskih operacija. Jedna procedura implementira i direktnu i inverznu brzu Fourier-ovu transformaciju; fleg koji je prvi argument procedure određuje koja je od njih u pitanju, a jedina razlika je znak veličine W, kao i to što kod direktne Fourier-ove transformacije treba na kraju svaki element izlaznog vektora podeliti sa dužinom vektora. Treba pomenuti i to da se algoritam obično primenjuje nad velikim brojem vektora, u kom slučaju bi svakako trebalo dalje raditi na izbacivanju iz procedure `fft()` svega onog što se može jednom unapred izračunati, a kasnije iz odgovarajućih tabela koristiti. Tako bi se u realnim uslovima verovatno permutacije indeksa unapred izračunale kao što je to urađeno sa stepenima veličine W, a takođe bi se posebno izračunali stepeni W za slučaj direktne odn. inverzne transformacije i to pre prvog poziva procedure `fft()`. Moguća varijacija rešenja je i da se umesto posebnih polja za realni odn. imaginarni deo ulaznog odn. izlaznog vektora definiše struktura koji će predstavljati kompleksan broj[1] i da se onda proceduri `fft()` prenesu polja veličina ovog tipa.

Sledi kod koji implementira metodu:

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>

/*
 * Funkcija shuffle() transformise dati broj u broj koji se dobija kada
 * se odredjen broj bitova sa desne strane procita naopacke. Parametri
 * funkcije su:
 *   n - broj koji treba transformisati
 *   p - broj bitova koje treba procitati naopacke radi opisane
 *       transformacije
 * Dati broj mora biti u opsegu [0,2^p), a broj bitova koji se citaju
 * naopacke mora biti u opsegu [1,15].
 */
static int
shuffle(int n, int p)
{
        int             result; /* Rezultat transformacije. */
        int             k;      /* Brojac u petljama. */

        /* Proverava se validnost argumenata. */
        assert(n >= 0 && n < 1 << p);
        assert(p >= 1 && p <= 15);

        /* Inicijalizuje se vrednost rezultata na zadnji bit broja n i
         * potom za svaki preostali bit koji treba obrnuti... */
        for (result = n & 1, k = 1; k < p; k++) {
                /* ...dati broj se siftuje udesno, rezultat ulevo i na
                 * kraj rezultata se stavlja trenutno poslednji bit datog
                 * broja. */
                n >>= 1;
                result <<= 1;
                result |= n & 1;
```

[1] alternativno, ako se koristi C prevodilac koji podržava C99 standard, može se upotrebiti ugrađeni tip za kompleksne brojeve koji je predviđen ovim standardom; GNU C prevodilac već ima ugrađenu podršku za ovaj aspekt standarda

```
        }

        return result;
}

/*
 * Funkcija complex_powers() izracunava stepene zadatog kompleksnog
 * broja sa jedinicnim modulom. Parametri funkcije su:
 *   theta - argument (fazni ugao) koji odredjuje kompleksan broj
 *   n - broj stepeni koje treba izracunati
 *   re, im - polja u koja ce biti smestene realne, odnosno imaginarne
 *            komponente stepena kompleksnog broja
 * Broj stepeni koje treba izracunati mora biti veci od 0.
 */
static void
complex_powers(double theta, int n, double *re, double *im)
{
        int             k;      /* Brojac u petlji. */

        /* Proverava se validnost broja stepeni koje treba izracunati. */
        assert(n > 0);

        /* Racuna se nulti stepen i ako je samo njega trebalo izracunati
         * izlazi se iz funkcije. */
        re[0] = 1, im[0] = 0;
        if (n == 1)
                return;

        /* Racunaju se preostali stepeni datog komplesknog broja. Pritom
         * se koriste trigonometrijske relacije o kosinusu, odn. sinusu
         * zbira. */
        re[1] = cos(theta);
        im[1] = sin(theta);
        for (k = 2; k < n; k++) {
                re[k] = re[k - 1] * re[1] - im[k - 1] * im[1];
                im[k] = im[k - 1] * re[1] + re[k - 1] * im[1];
        }
}

/*
 * Funkcija fft() racuna Fourier-ovu transformaciju vektora FFT
 * algoritmom.  Argumenti funkcije su:
 *   direct_fft - fleg koji odredjuje da li se racuna direktna (kad je
 *                fleg razlicit od 0) ili inverzna (kad je fleg jednak
 *                0) Fourier-ova transformacija
 *   p - duzina vektora cija se Fourier-ova transformacija izracunava
 *       izrazena kao stepen broja 2 (tj.  stvarna duzina vektora
 *       jednaka je 2^p)
 *   re_in, im_in - polja sa realnim, odnosno imaginarnim delovima
 *                  vektora cija se Fourier-ova transformacija racuna
 *   re_out, im_out - polja u koja ce biti smesteni realni, ondosno
 *                  imaginarni delovi izracunate Fourier-ove
 *                  transformacije ulaznog vektora
 * Vrednost argumenta p mora biti u opsegu [1,15].
 */
void
fft(int direct_fft, int p,
```

```
    double *re_in, double *im_in, double *re_out, double *im_out)
{
        const double    pi = 3.14159265358979323846;    /* Broj pi. */
        int             n;      /* Duzina vektora cija se Fourier-ova
                                 * transformacija izracunava. */
        int             shuffled;       /* Obrnuta vrednost tekuceg
                                         * indeksa. */
        double         *w_re,
                       *w_im;   /* Realna, ondosno imaginarna komponenta
                                 * vektora stepena velicine W. */
        int             w_pow;  /* Stepen velicine W u tekucem koraku
                                 * algoritma. */
        int             step;   /* Rastojanje medju elementima leptira. */
        int             lo,
                        hi;     /* Indeksi elemenata leptira. */
        int             w_curr; /* Stepen velicine W za dati leptir. */
        double          re,
                        im;     /* Promene vrednosti elemenata leptira. */
        int             k;      /* Indeks u petljama. */

        /* Proverava se validnost prvog argumenta funkcije. */
        assert(p >= 1 && p <= 15);

        /* Izracunava se duzina vektora. */
        n = 1 << p;

        /* Kopira se ulazni vektor u izlazni vektor, pri cemu se kopiranje
         * vrsi u elemenat sa obrnutim indeksom. */
        for (k = 0; k < n; k++) {
                shuffled = shuffle(k, p);
                re_out[k] = re_in[shuffled];
                im_out[k] = im_in[shuffled];
        }

        /* Alociraju se polja za stepene velicine W i izracunavaju ti
         * stepeni. */
        w_re = (double *) malloc((n / 2) * sizeof(double));
        w_im = (double *) malloc((n / 2) * sizeof(double));
        complex_powers((direct_fft ? 2 * pi / n : -2 * pi / n), n / 2,
                       w_re, w_im);

        /* U svakom koraku algoritma... */
        for (step = 1, w_pow = n / 2; step < n; step *= 2, w_pow /= 2)
                /* ...za sve parove elemenata izlaznog vektora... */
                for (lo = 0, hi = step; hi < n; lo += step, hi += step)
                        for (k = 0, w_curr = 0; k < step;
                             k++, lo++, hi++, w_curr += w_pow) {
                                /* ...izracunavaju se nove vrednosti
                                 * elemenata para. */
                                re = re_out[hi] * w_re[w_curr] -
                                    im_out[hi] * w_im[w_curr];
                                im = im_out[hi] * w_re[w_curr] +
                                    re_out[hi] * w_im[w_curr];
                                re_out[hi] = re_out[lo] - re;
                                im_out[hi] = im_out[lo] - im;
                                re_out[lo] += re;
                                im_out[lo] += im;
```

```
                }

        /* Oslobadjaju se polja sa komponentama vektora stepena velicine
         * W. */
        free(w_re);
        free(w_im);

        /* Ako je u pitanju direktna Fourier-ova transformacija, svi
         * elementi izlaznog vektora se mnoze sa 1/n. */
        if (direct_fft) {
                double          mul = 1.0 / n;  /* Broj sa kojim se mnoze
                                                 * elementi vektora. */
                for (k = 0; k < n; k++)
                        re_out[k] *= mul, im_out[k] *= mul;
        }
}
```

MATLAB

Diskretna Fourier-ova transformacija :	direktna `f = ifft(c)`, inverzna `c = fft(f)`
Diskretna konvolucija:	`c = conv(a, b)`
Digitalni filtri:	`y = filter(b, a, x)`

MATLAB već raspolaže gotovim rutinama za računanje direktne i inverzne diskretne Fourier-ove transformacije koje, naravno, koriste algoritam brze Fourier-ove transformacije. Funkcije koje ovo rade su `fft` i `ifft`. Bitno je napomenuti da je ovde korišćena tzv. inženjerska nomenklatura u kojoj se pod direktnom transformacijom podrazumeva ono što je u ovoj knjizi nazivano inverznom transformacijom i obrnuto.

Diskretnu konvoluciju dva vektora moguće je izračunati korišćenjem funkcije `conv`.

4.39 *Napisati* MATLAB *funkciju koja množi dva "velika cela broja" koja su zadata vektorima svojih cifara. Prilikom realizovanja množenja koristiti FFT i njenu vezu sa diskretnom konvolucijom.*

Rešenje: Neka vektori `x = [`x_1`,...,`x_n`]` i `y = [`y_1`,..., `y_m`]` sadrže cifre brojeva koji se množe, odnosno neka su celi brojevi

$$\begin{aligned} x &= x_1 10^{n-1} + x_2 10^{n-2} + \ldots + x_n 10^0, \\ y &= y_1 10^{m-1} + y_2 10^{m-2} + \ldots + y_n 10^0 \end{aligned}$$

Posmatrajmo polinome

$$\begin{aligned} \bar{x}(t) &= x_1 t^{n-1} + x_2 t^{n-2} + \ldots + x_n t^0, \\ \bar{y}(t) &= y_1 t^{m-1} + y_2 t^{m-2} + \ldots + y_m t^0. \end{aligned}$$

Proizvod ova dva polinoma se može efikasno izračunati diskretnom konvolucijom vektora `x` i `y`. Vrednost proizvoda polinoma $\bar{x}$ i $\bar{y}$ u tački 10 jeste traženi broj z — proizvod brojeva x i y

$$z = \bar{z}_1 10^{m+n-2} + \bar{z}_2 10^{m+n-3} \ldots + \bar{z}_{m+n-1} 10^0.$$

Međutim, koeficijenti ovog polinoma nisu cifre ovog broja. Normalizacijom tj. prezapisivanjem prethodne sume tako da koeficijenti $\bar{z}_i, i = 1, \ldots, n+m$ budu u intervalu $[0, 9]$ dobijaju se tražene cifre. Naime, svaki od brojeva $\bar{z}_i$ se može na jedinstven način predstaviti u obliku

$$\bar{z}_i = 10q_i + r_i, \quad 0 \leq r_i < 10$$

pa se njemu odgovarajući član prethodne sume može zapisati kao

$$\bar{z}_i 10^{m+n-1-i} = (10q_i + r_i)10^{m+n-1-i} = 10^{m+n-i} q_i + r_i 10^{m+n-1-i}$$

Na osnovu ove veze, izvode se rekurentne formule

$$\begin{aligned} q_{m+n} &= 0, \\ z_i &= (\bar{z}_i + q_{i+1}) \texttt{ mod } 10, \\ q_i &= (\bar{z}_i + q_{i+1}) \texttt{ div } 10, \qquad i = m+n-1, \ldots, 1, \\ z_0 &= q_1 \end{aligned}$$

i na osnovu njih, kod se jednostavno piše.

```
function z = fft_multiply(x, y)
%Duzine ulaznih vektora
n = length(x); m = length(y);
%Duzina rezultujuceg vektora
l = n + m - 1;

%Vektori se prosiruju nulama
x = [x zeros(1, l-n)]
y = [y zeros(1, l-m)]

%Vrsi se konvolucija
z = round(ifft(fft(x).*fft(y)))

%Normalizuju se cifre rezultata
q = 0;
for i = l:-1:1
    tmp  = z(i) + q;
    z(i) = mod(tmp, 10);
    q    = floor(tmp/10);
end
if (q ~= 0)
    z = [q z];
end
```

4.40 *Korišćenjem* MATLAB-*a generisati sekundu zvuka amplitude 1.2 i frekvencije 440Hz, semplovanog frekvencijom od 44100Hz. Prikazati gustinu energijskog spektra dobijenog signala (Frequency Power Spectrum).*

Rešenje: Zvuk je predstavljen sinusoidnim talasom.

```
SamplingFreq = 44100;   %frekvencija odabiranja
Freq = 440;             %frekvencija zvuka
Ampl = 1.2;             %amplituda zvuka
T = 1;                  %trajanje zvuka

%Generisanje zvucnog talasa
SampleDuration = 1/SamplingFreq;
t = 0 : SampleDuration : T-SampleDuration;
x = Ampl * sin(2*pi*Freq*t);

%Pustanje zvuka
sound(x, SamplingFreq);

%Prelazak u frekvencijski domen
X = fft(x);
N = length(X);

%Iscrtavanje energijskog spektra
stem(0:SamplingFreq/2, 2*abs(X(1:N/2+1))/N);
xlabel('Frequency [Hz]'); ylabel('Energy content');
```

Zbog simetrije dobijenog vektora Fourierovih koeficijenata, iscrtavaju se moduli samo prve polovine njegovih elemenata. Drugi način za prikazivanje energijskog spektra snage signala je korišćenje ugrađene komande `specgram`.

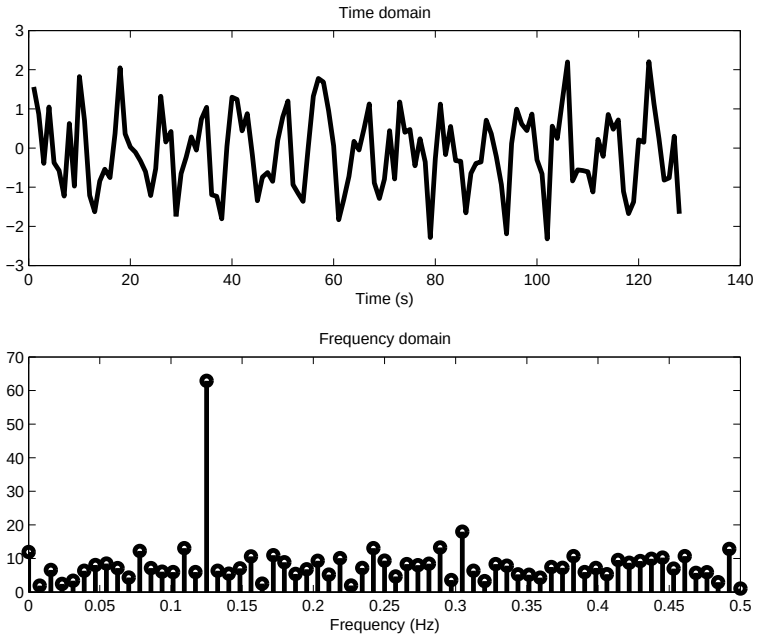
4.41 *Signal frekvencije* $0.125Hz$ *semplovan frekvencijom od* $1Hz$ *je izložen dejstvu šuma. Korišćenjem* MATLAB *digitalnih filtara izvršiti čišćenje signala od šuma.*

Rešenje: Prvi korak je generisanje šumovitog signala dodavanjem belog šuma na polazni signal.

```
%128 semplova pri frekvenciji 1Hz
t = 1 : 128;
%frekvencija signala
f = 0.125;

x = sin(2*pi*f*t) + ...     %signal
    0.75 * randn(size(t));   %sum
```

Sa dobijene slike u vremenskom domenu nije moguće očitati frekvenciju signala. Prelaskom u frekvencijski domen dobija se se jasnija slika. Sa dobijenog energijskog spektra moguće je očitati frekvenciju od $0.125Hz$ (slika 4.4).

```
% Prelazak u frekvencijski domen
X = fft(x);

% Raspon frekvencija
```



Slika 4.7: a) Vremenski domen signala b) Frekvencijski domen signala

```
Freq  = 1/128*(0:64);
% Energija
Power = abs(X);

% Nalazimo frekvenciju sa najvecom snagom
[MaxPower, MaxIndex] = max(Power);
Freq(MaxIndex)
```

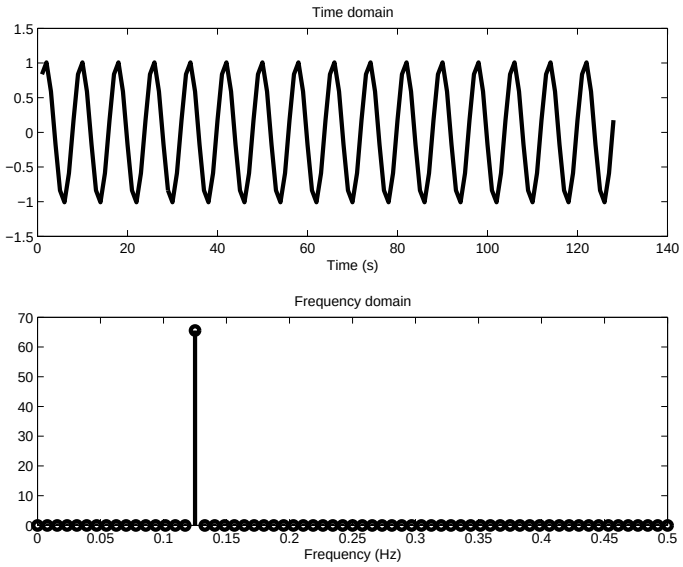
Filtriranje će biti urađeno na tri načina.

(i) Ručno ćemo ukloniti sve elemente vektora Fourier-ovih koeficijenata koji imaju manju snagu od određenog praga. Pošto se signal sastoji samo od jedne frekvencije, anuliraćemo sve one Fourier-ove koeficijente čija je snaga manja od maksimalne snage koja se javlja u Fourierovom spektru. Rezultat filtriranja je prikazan na slici 4.4.

```
X1 = fft(x);
X1(abs(X1)<max(abs(X1))) = 0;
x1 = real(ifft(X1));
```

(ii) Korišćenjem jednostavnog nisko-frekvencijskog (*low-pass*) filtra suzbićemo sve visoke frekvencije u signalu. Jedan primer takvog filtra je lokalno usrednjavanje signala definisano u vremenskom domenu sa

$$y(t) = 0.25(x(t) + x(t-1) + x(t-2) + x(t-3))$$



Slika 4.8: Manualno filtriranje u frekvencijskom domenu

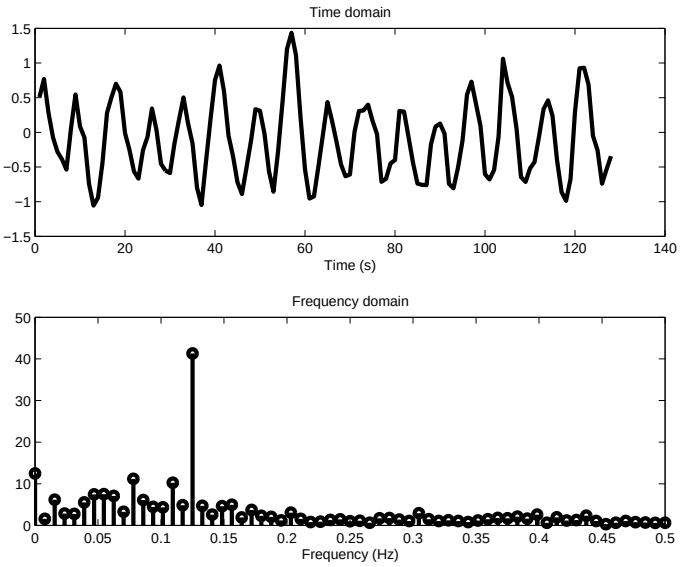
Ovaj filtar se najefikasnije primenjuje ukoliko se posmatra kao diskretna konvolucija vektora `x` i vektora `[0.25 0.25 0.25 0.25]` koja se dalje izvodi na standardni način, (prelaskom u frekvencijski domen). Rezultat filtriranja je prikazan na slici 4.4. Primetimo da primena ovog filtra eliminiše komponente signala visoke frekvencije, dok zadržava niskofrekvencijske komponente.

```
x2 = conv(x, [0.25 0.25 0.25 0.25]);
X2 = fft(x2);
```

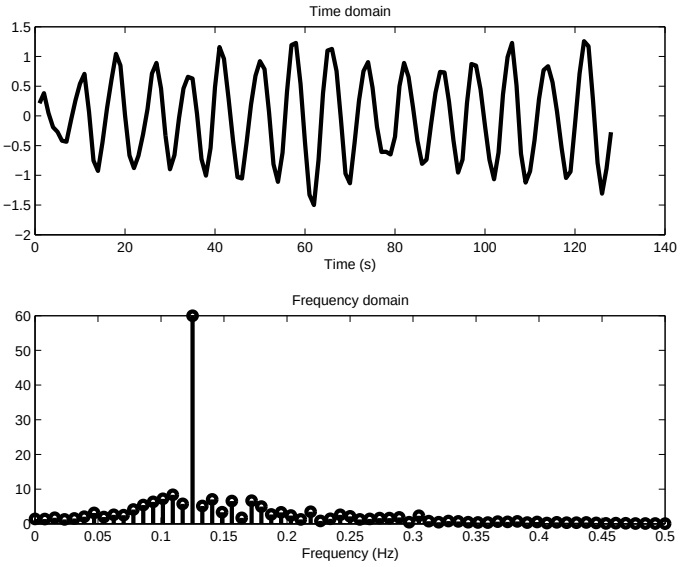
(iii) Treći, najsofisticiraniji način filtriranja koristi tzv. Butterworth-ov opsežni (*band-pass*) filtar, koji suzbija sve frekvencije signala van određenog opsega.

```
[b, a] = butter(1, [0.2 0.3], 'bandpass')
x3 = filter(b, a, x);
X3 = fft(x3);
```

Filtar je izgrađen specijalnom funkcijom `butter`, a interval `[0.2 0.3]` se odnosi na 20% do 30% polaznog frekvencijskog opsega, tj. zadržavaju se frekvencije iz `[0.1 0.15]`. Filtriranje se dalje vrši korišćenjem ugrađene funkcije `filter`. Rezultat filtriranja je prikazan na slici 4.4.



Slika 4.9: Nisko-frekvencijsko filtriranje



Slika 4.10: Opsežno filtriranje

4.5 Ravnomerna aproksimacija

Ravnomerna aproksimacija neprekidne funkcije na intervalu $[a,b]$ je generalisani polinom $Q_n(x)$ takav da je

$$E_n(f) = \|f - Q_n\| = \inf_{a_0,\ldots,a_n} \|f - \sum_{k=0}^{n} a_k g_k\|, \qquad \|f\| = \sup_{[a,b]} |f(x)|$$

Ako je $Q_n(x)$ algebarski polinom i ako su poznate tačke Čebišev-ljeve alternanse x_k, $k = 0,\ldots,n+1$, koeficijenti polinoma i greška najbolje ravnomerne aproksimacije $E_n(f)$ se mogu odrediti kao rešenja sistema jednačina

$$f(x_k) - Q_n(x_k) = \alpha(-1)^k E_n(f), \qquad k = 0,\ldots,n+1,$$

gde je $\alpha = 1$ ili $\alpha = -1$ istovremeno za sve k. Znak α se određuje tako da je izračunato $E_n(f) \geq 0$.

Polinomi Čebišev-a prve vrste

$$T_n(x) = \cos(n \arccos x), \qquad x \in [-1,1]$$

Zadovoljavaju rekurentnu relaciju

$$T_0(x) = 1, \qquad T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1,2,\ldots,$$

odakle sledi još jedan izraz za Čebišev-ljev polinom

$$T_n(x) = \frac{1}{2}\left((x+\sqrt{x^2-1})^n + (x-\sqrt{x^2-1})^n\right), \qquad n = 0,1,2,\ldots.$$

Ortogonalnost Čebišev-ljevih polinoma

$$(T_n, T_m) = \int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} 0, & n \neq m \\ \pi/2, & n = m \neq 0 \\ \pi, & n = m = 0 \end{cases}$$

Polinomi najmanjeg odstupanja od nule (sa koeficijentom 1 uz najviši stepen)

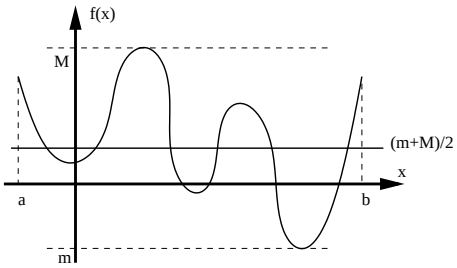
$$\overline{T}_n(x) = 2^{1-n} T_n(x), \qquad \max_{[-1,1]} |\overline{T}_n(x)| = 2^{1-n}$$

4.42 *Odrediti polinom $p_0(x)$ nultog stepena najbolje ravnomerne aproksimacije funkcije $f(x) \in C[a,b]$.*

Rešenje: Polinom nultog stepena najbolje ravnomerne aproksimacije jeste prava $p_0(x) = c_0$ paralelna x-osi. Stoga se traži konstanta c_0 takva da je

$$\|f(x) - c_0\| = \inf_c \|f(x) - c\| = \inf_c \big(\sup_{x\in[a,b]} |f(x) - c| \big).$$

Neka je $m = \min_{[a,b]} f(x)$ i $M = \max_{[a,b]} f(x)$. Tražena konstanta je $c_0 = \frac{m+M}{2}$, tj. polinom nultog stepena najbolje ravnomerne aproksimacije je $p_0(x) \equiv \frac{m+M}{2}$ (vidi sliku 4.11)



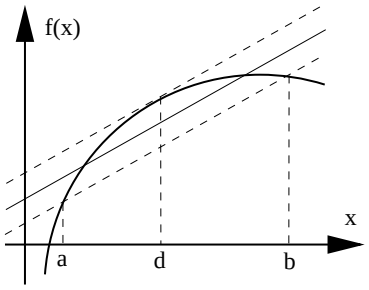
Slika 4.11: Polinom nultog stepena najbolje ravnomerne aproksimacije.

4.43 *Odrediti polinom prvog stepena $p_1(x)$ najbolje ravnomerne aproksimacije konkavne funkcije $f(x)$ na intervalu $[a,b]$.*

Rešenje: Kao u prethodnom zadatku, mogu se konstruisati dve uzajamno paralelne prave između kojih se nalazi data funkcija. Pri tome, rastojanje tih pravih treba da bude što manje. Te prave su sečica kroz tačke $(a, f(a))$ i $(b, f(b))$ i tangenta na krivu paralelna sečici. Polinom najbolje ravnomerne aproksimacije prvog stepena je prava koja prolazi sredinom rastojanja između sečice i tangente, i paralelna je sa njima (slika4.12).

Zadatak se može rešiti i korišćenjem tačaka Čebišev-ljeve alternanse. U ovom slučaju, pošto je funkcija konkavna, te tačke su krajnje tačke intervala $x_0 = a$ i $x_2 = b$, i tačka $x_1 = d$ u kojoj funkcija $|f - p_1|$ dostiže maksimalnu vrednost. Sistem jednačina kojim su određeni koeficijenti polinoma najbolje aproksimacije $p_1(x) = c_0 + c_1 x$ i greška L glasi:

$$\begin{array}{lcl} f(a) - (c_0 + c_1 a) & = & \alpha L \\ f(d) - (c_0 + c_1 d) & = & -\alpha L \\ f(b) - (c_0 + c_1 b) & = & \alpha L \\ \big(f(x) - (c_0 + c_1 x)\big)'_{x=d} & = & 0 \end{array}$$



Slika 4.12: Ravnomerna aproksimacija pravom konkavne funkcije.

4.44 *Odrediti polinom prvog stepena najbolje ravnomerne aproksimacije funkcije* $f(x) = |x|$ *na intervalu* $[-1, 5]$.

Rešenje: Pošto je funkcija konveksna, tačke Čebišev-ljeve alternanse su $x_0 = -1$, $x_1 = 0$ i $x_2 = 5$. Sistem jednačina kojim su određeni koeficijenti najbolje aproksimacije i greška je

$$\begin{array}{rcl} 1 - c_0 + c_1 &=& \alpha L \\ 0 - c_0 &=& -\alpha L \\ 5 - c_0 - 5c_1 &=& \alpha L \end{array} \quad \Longrightarrow \quad \begin{array}{rcl} c_0 &=& 5/6 \\ c_1 &=& 2/3 \\ L &=& 5/6 \end{array}$$

Polinom najbolje ravnomerne aproksimacije i greška su

$$p_1(x) = \frac{2}{3}x + \frac{5}{6} \qquad L = \frac{5}{6}.$$

4.45 *Naći polinom najbolje ravnomerne aproksimacije drugog stepena* $p_2(x)$ *za funkciju* $f(x) = 144/(x + 2)$ *na intervalu* $[0, 6]$, *pod pretpostavkom da su tačke Čebišev-ljeve alternanse* 0, 1, 4 *i* 6. *Oceniti grešku aproksimacije.*

Rešenje: Prema teoremi Čebiševa je

$$f(x_i) - c_0 - c_1 x_i - c_2 x_i^2 = \alpha(-1)^i L, \qquad i = 0, 1, 2, 3,$$

gde su x_i date tačke Čebišev-ljeve alternanse, a $\alpha = 1$ ili $\alpha = -1$ za svako i. Uvrstimo zadate vrednosti za x_i, i dobijamo sistem linearnih jednačina

$$\begin{array}{rcl} 72 - c_0 = \alpha L \\ 48 - c_0 - c_1 - c_2 = -\alpha L \\ 24 - c_0 - 4c_1 - 16c_2 = \alpha L \\ 18 - c_0 - 6c_1 - 36c_2 = -\alpha L \end{array} \quad \Longrightarrow \quad \begin{array}{l} c_0 = 69 \\ c_1 = -20 \\ c_2 = 2 \\ L = 3, \quad \alpha = 1 \end{array} .$$

Stoga je

$$p_2(x) = 2x^2 - 20x + 69, \qquad L = \max_{x \in [0,6]} |f(x) - p_2(x)| = 3.$$

4.46 *Odrediti na intervalu* $[0,1]$ *polinom najbolje ravnomerne aproksimacije prvog stepena za funkciju* $f(x) = e^x(x^2+2)$ *i odrediti grešku aproksimacije. Koeficijente polinoma izračunati sa tačnošću* 10^{-4}.

Rešenje: Kako je $f''(x) = e^x(x+2)^2 > 0$, funkcija $f(x)$ je konveksna na intervalu $[0,1]$. Stoga su tačke Čebišev-ljeve alternanse 0, d i 1, gde je d tačka ekstrema razlike funkcija $f(x)$ i traženog polinoma prvog stepena $p_1(x) = c_0 + c_1 x$. Rešenja sistema jednačina dobijenog primenom Čebišev-ljeve teoreme su

$$c_0 = 1.3014, \quad c_1 = 6.1548, \quad d = 0.5724, \quad L = 0.6986, \quad \alpha = 1,$$

pa je traženi polinom

$$p_1(x) = 6.1548x + 1.3014, \qquad \|f(x) - p_1(x)\| = 0.6986.$$

4.47 *Naći polinom najbolje aproksimacije oblika* $p_1(x) = c_0 + c_1 x$ *za funkciju* $f(x) = \sqrt{1+x^2}$ *na odsečku* $[0,1]$. *Pomoću tog polinoma pokazati da, ako su* a *i* b *katete pravouglog trougla* ($a \geq b$), *tada je hipotenuza trougla* $c = 0.955a + 0.414b$, *sa greškom do* 4.5% *od veličine* a.

Rešenje: Funkcija $f(x) = \sqrt{1+x^2}$ je na intervalu $[0,1]$ konveksna, te su tačke Čebišev-ljeve alternanse pri ravnomernoj aproksimaciji ove funkcije linearnom funkcijom krajevi intervala i tačka d u kojoj razlika $f(x) - p_1(x)$ dostiže maksimum. Primenom Čebišev-ljeve teoreme se dobija da je

$$d = 0.4551, \quad c_0 = 0.9551, \quad c_1 = 0.4142, \quad L = 0.0449, \quad \alpha = 1.$$

Dakle,

$$p_1(x) = 0.414x + 0.955, \qquad \max_{x\in[0,1]} |f(x) - p_1(x)| = 0.045.$$

Prema Pitagorinoj teoremi hipotenuza trougla je $c = \sqrt{a^2+b^2}$, gde su a i b dužine kateta, te je $c = a\, f(b/a)$. Kako je $a \geq b$, tj. $b/a \in [0,1]$, prema prethodnom je $c \approx a\, p_1(b/a) = 0.955a + 0.414b$ sa greškom $|c - a\, p_1(b/a)| = a\, |f(b/a) - p_1(b/a)| = 0.045a = 4.5\%a$, što je i trebalo dokazati.

4.48 *Neka je* $f(x)$ *neprekidna funkcija, neparna u odnosu na sredinu intervala* $[a,b]$. *Dokazati da je i njena ravnomerna aproksimacija na datom intervalu neparna funkcija.*

Rešenje: Funkcija se može linearnom smenom preslikati na interval $[-1,1]$, te se, ne umanjujući opštost, može pretpostaviti neparnost u odnosu na tačku 0. Neka je $Q^0(x)$ polinom najbolje ravnomerne aproksimacije, i

$$E_n(f) = \|f - Q^0\| = \sup_{x\in[-1,1]} |f(x) - Q^0(x)|$$

odstupanje najbolje aproksimacije. Kako je za proizvoljno $x \in [-1,1]$ i $-x \in [-1,1]$, sledi da je

$$\begin{aligned} E_n(f) &\geq |f(-x) - Q^0(-x)| = |-f(x) - Q^0(-x)| \\ &= |f(x) + Q^0(-x)| = |f(x) - (-Q^0(-x))|. \end{aligned}$$

To znači da je i $-Q^0(-x)$ polinom najbolje ravnomerne aproksimacije funkcije $f(x)$ na intervalu $[-1,1]$. Pošto je polinom najbolje ravnomerne aproksimacije jedinstveno određen, mora biti $Q^0(x) \equiv -Q^0(-x)$, odnosno polinom najbolje ravnomerne aproksimacije je neparna funkcija. Analogno tvrđenje se izvodi za parne funkcije.

4.49 *Za funkciju $f(x) = |x|$ odrediti polinom najbolje ravnomerne aproksimacije drugog stepena na intervalu $[-1,1]$ i oceniti grešku aproksimacije.*

Rešenje: Funkcija je parna, pa i njena aproksimacija mora biti parna funkcija

$$p_2(x) = c_0 + c_2 x^2.$$

Zbog parnosti, problem možemo rešavati samo na intervalu $[0,1]$. Smenom $x = \sqrt{t}$, $t \in [0,1]$ problem svodimo na određivanje linearne aproksimacije $q_1(t) = c_0 + c_2 t$ funkcije $\sqrt{t}$ na intervalu $[0,1]$. Algoritmom opisanim u prethodnim zadacima dobijamo da je $c_0 = 1/8$, $c_2 = 1$, a greška aproksimacije $L = 1/8$. Stoga je

$$p_2(x) = \frac{1}{8} + x^2 \qquad \text{i} \qquad \max_{[-1,1]} |f(x) - p_2(x)| = 0.125.$$

4.50 *Odrediti polinom najbolje ravnomerne aproksimacije trećeg stepena funkcije $f(x) = \int_0^x \frac{\sin\sqrt{|t|}}{t}\,dt$ na intervalu $[-1,1]$ i oceniti grešku aproksimacije.*

Rešenje: Funkcija je parna, jer, korišćenjem smene $t = -s$, dobijamo da je

$$f(-x) = \int_0^{-x} \frac{\sin\sqrt{|t|}}{t}\,dt = \int_0^x \frac{\sin\sqrt{|s|}}{s}\,ds = f(x).$$

Na osnovu prethodnog zadatka sledi da je i polinom najbolje ravnomerne aproksimacije takodje parna funkcija, tj. on je oblika $p_3(x) = c_0 + c_2 x^2$. Zbog parnosti je dovoljno zadatak rešavati na intervalu $[0,1]$. Ako se uvede smena $y = x^2$, problem se svodi na aproksimaciju funkcije

$$f(x) = F(y) = \int_0^{\sqrt{y}} \frac{\sin\sqrt{t}}{t}\,dt, \qquad y \in [0,1]$$

polinomom prvog stepena $q_1(y) = c_0 + c_2 y$. Prvi i drugi izvod funkcije $F(y)$ su

$$F'(y) = \frac{\sin(y^{1/4})}{2y} \qquad F''(y) = \frac{y^{1/4}\cos(y^{1/4}) - 4\sin(y^{1/4})}{8y^2}$$

Kako je za $x \in [0,1]$ i $y = x^4$

$$\tan x \geq \frac{x}{4} \quad \Rightarrow \quad \sin x - \frac{x}{4}\cos x \geq 0 \quad \Rightarrow \quad 4\sin(y^{1/4}) - y^{1/4}\cos(y^{1/4}) \geq 0,$$

sledi da je $F''(y) \leq 0$, tj. funkcija $F(y)$ je konkavna funkcija na $[0,1]$. Problem je tako sveden na aproksimaciju konkavne funkcije $F(y)$ pravom $q_1(y)$. Tačke Čebiševljeve alternanse su $y_0 = 0$, $y_1 = d$ i $y_2 = 1$. Koeficijenti najbolje aproksimacije su rešenja sistema jednačina

$$\begin{aligned} &F(0) - c_0 = \alpha L \\ &F(d) - c_0 - c_2 d = -\alpha L \\ &F(1) - c_0 - c_2 = \alpha L \\ &\big(F(y) - q_1(y)\big)'\big|_{x=d} = 0 \end{aligned}$$

Poslednja jednačina je nelinearna i rešava se po d nekom iterativnom metodom, na primer Newton-ovom (§7.3). Integrali $F(d)$ i $F(1)$ se računaju nekom kvardaturnom formulom, na primer Simpson-ovom (§3.1). Rešavanjem dobijenog sistema linearnih jednačina i vraćanjem na staru promenljivu x, nalazimo da su traženi polinom najbolje ravnomerne aproksimacije funkcije $f(x)$ i greška aproksimacije

$$p_3(x) = 1.892x^2 + 0.467, \qquad \max_{x\in[-1,1]} |f - p_3| = 0.47.$$

4.51 *Naći na intervalu* $[-1,1]$ *polinom najbolje ravnomerne apoksimacije trećeg stepena za funkciju*

$$f(x) = x^2 \ln|x|.$$

U tački $x = 0$ *funkcija je definisana svojom graničnom vrednošću* $f(0) = 0$.

Rešenje: Funkcija $f(x)$ je parna na intervalu $[-1,1]$ te i traženi polinom treba da bude parna funkcija, tj. da bude oblika $p_3(x) = c_0 + c_2x^2$. Posle uvođenja smene $t = x^2$ dati zadatak se svodi na problem nalaženja polinoma najbolje ravnomerne aproksimacije prvog stepena $q_1(t) = c_0 + c_2t$ za funkciju $F(t) = t\ln t/2$ na intervalu $[0,1]$. Funkcija $F(t)$ je na tom intervalu konveksna ($F''(t) = 1/(2t) > 0$), te je prava $q_1(t)$ paralelna sečici kroz tačke $(0, F(0))$ i $(1, F(1))$, i nalazi se na sredini rastojanja između te sečice i njoj paralelne tangente funkcije $F(t)$. Sečica pripada x-osi, a kako je $F'(t) = 0$ za $t = e^{-1}$ i $F(e^{-1}) = -1/(2e)$, to je

$$p_3(x) \equiv q_1(t) = \frac{-1}{4e}, \qquad \text{i} \qquad \max_{[-1,1]} |f(x) - p_3(x)| = 0.092.$$

4.52 *Za funkciju* $f(x) = (1+|x|)^{-1}$ *na intervalu* $[-1,1]$ *naći polinom najbolje ravnomerne aproksimacije trećeg stepena i oceniti grešku.*

Rešenje: Funkcija $f(x)$ je parna, pa se smenom $t = x^2$ problem svodi na ravnomernu aproksimaciju pravom funkcije $1/(1+\sqrt{t})$, $t \in [0,1]$. Tako se dobija da su traženi polinom najbolje ravnomerne aproksimacije i greška

$$p_3(x) = -0.5x^2 + 0.8954, \qquad \max_{[-1,1]} |f(x) - p_3(x)| = 0.1046.$$

4.53 *Naći polinom najbolje ravnomerne aproksimacije trećeg stepena za funkciju*

$$f(x) = e^{x^2-2x}$$

na intervalu $[0,2]$.

Rešenje: Smenom $x = y + 1$, interval se preslikava u interval $[-1,1]$ a funkcija $f(x)$ u $f(y+1) = F(y) = e^{y^2-1}$. Funkcija $F(y)$ je parna pa je i polinom najbolje aproksimacije parna funkcija, $p(y) = c_0 + c_2y^2$, a aproksimaciju je dovoljno odrediti na intervalu $[0,1]$. Uvođenjem nove smene $t = y^2$, zadatak svodimo na problem aproksimacije konveksne funkcije $F(\sqrt{t}) = g(t) = e^{t-1}$ na intervalu $[0,1]$ polinomom prvog stepena $q_1(t) = c_0 + c_2t$. Rešenje ovog problema je $q_1(t) = 0.3289 + 0.6321t$, $L = 0.039$, pa je, uzimajući u obzir uvedene smene, traženi polinom najbolje ravnomerne aproksimacije trećeg stepena za funkciju $f(x)$ na intervalu $[0,2]$

$$p_3(x) = 0.3289 + 0.6321(x-1)^2, \qquad \max_{x\in[0,2]} |f(x) - p_3(x)| = 0.039.$$

4.54 *Naći polinom najbolje ravnomerne aproksimacije drugog stepena za funkciju*

$$f(x) = e^{x^2-2x+7} + 2x^2 - 4x + 6$$

na intervalu $[0,2]$. *Koeficijente polinoma izračunati sa tačnošću* 10^{-5}.

Rešenje: Smenom $x = y + 1$, interval se preslikava u interval $[-1,1]$ a funkcija $f(x)$ u $f(y+1) = F(y) = e^{y^2+6} + 2y^2 + 4$. Funkcija $F(y)$ je parna pa je i polinom najbolje aproksimacije parna funkcija, $p(y) = c_0 + c_2y^2$, a aproksimaciju je dovoljno odrediti na intervalu $[0,1]$. Uvođenjem nove smene $t = y^2$, zadatak svodimo na problem aproksimacije konveksne funkcije $F(\sqrt{t}) = g(t) = e^{t+6} + 2t + 4$ na intervalu $[0,1]$ polinomom prvog stepena $q_1(t) = c_0 + c_2t$. Rešenje ovog problema je $q_1(t) = 364.69220 + 695.20436t$, $L = 42.74$, pa je, uzimajući u obzir uvedene smene, traženi polinom najbolje ravnomerne aproksimacije drugog stepena za funkciju $f(x)$ na intervalu $[0,2]$

$$p_2(x) = 1059.89657 - 1390.40873x + 695.20436x^2, \qquad \max_{x\in[0,2]} |f(x) - p_2(x)| = 42.74.$$

C Implementacija metode najbolje ravnomerne aproksimacije biće demonstrirana na slučaju kada su date tačke Čebišev-ljeve alternanse. Implementacija metode je relativno jednostavna i sastoji se u postavljanju i rešavanju sistema jednačina po koeficijentima polinoma najbolje aproksimacije i najvećem odstupanju. Rešavanje sistema jednačina je dovoljno generalan problem da bude izdvojeno u posebnu funkciju, pa tako biblioteka `libnumerics` implementira Gauss-ovu metodu za rešavanje sistema jednačina preko procedure `gauss()`. Ova procedura je, prikazana u odeljku §5.1. Na nju treba obratiti posebnu pažnju, jer je opet u pitanju metoda koja se često primenjuje u praksi.

Metodu najbolje ravnomerne aproksimacije implementira procedura `uniform()`. Ova procedura prosto postavlja sistem jednačina po koeficijentima polinoma najbolje aproksimacije i odstupanju i zatim poziva proceduru `gauss()` da reši sistem. S obzirom da je funkcija koja se aproksimira preneta proceduri u analitičkom obliku, procedura demonstrira i korišćenje biblioteke `libmatheval`: za kreiranje evaluatora poziva se procedura `evaluator_create()`, zatim se za izračunavanje funkcije u datim tačkama poziva procedura `evaluator_evaluate_x()`, i na kraju se evaluator uništava pozivom procedure `evaluator_destroy()`. Sledi kod koji sačinjava proceduru `uniform()`:

```
#include <assert.h>
#include <stdlib.h>
#include <matheval.h>

extern void     gauss(int n, double **A, double *b, double *x);

/*
 * Funkcija uniform() izracunava koeficijente polinoma najbolje ravnomerne
 * aproksimacije za datu funkciju. Argumenti funkcije su:
 *   function - string koji predstavlja funkciju koja se aproksimira
 *   n - red polinoma kojim se vrsi apoksimacija (tj. broj zadatih
 *       tacaka Cebisevljeve alternanse je n+2)
 *   x - polje sa tackama Cebisevljeve alternanse
 *   c - polje u koje ce biti smesteni izracunati koeficijenti polinoma
 *       najbolje ravnomjerne aproksimacije
 *   alpha, L - vrednosti koje odredjuju odstupanje najbolje
 *              aproksimacije u skladu sa Cebisevljevom teoremom
 * Argumenti funkcije moraju da zadovolje uslov da je funkcija sintaksno
 * ispravno zadata (za nacin zadavanja funkcije videti dokumentaciju
 * libmatheval biblioteke), te da je red trazenog interpolacionog
 * polinoma veci ili jednak 0, kao i da su tacke Cebisevljeve alternanse
 * date u rastucem redosledu.
 */
void
uniform(char *function, int n, double *x, double *c, int *alpha, double *L)
{
        void           *evaluator;      /* Evaluator koji sluzi za
                                         * izracunavanje funkcije. */
        double        **A;      /* Matrica sistema jednacina koji
                                 * odredjuje koeficijente aproksimacionog
                                 * polinoma i vjednost najveceg
                                 * odstupanja. */
        double         *b;      /* Vektor desnih strana gornjeg sistema. */
        double         *X;      /* Vektor sa resenjima gornjeg sistema. */
```

```
        int             i,
                        j;      /* Brojaci u petljama. */

        /* Kreira se evaluator za izracunavanje vrednosti funkcije. */
        evaluator = evaluator_create(function);

        /* Proverava se validnost ulaznih podataka. */
        assert(evaluator != NULL);
        assert(n >= 0);
        for (i = 0; i <= n; i++)
                assert(x[i] < x[i + 1]);

        /* Alocira se memorija za matricu sistema i vektor desnih strana
         * sistema jednacina koji odredjuje koeficijente aproksimacionog
         * polinoma i najvece odstupanje i postavljaju se te jednacine. */
        A = (double **) malloc((n + 2) * sizeof(double *));
        for (i = 0; i <= n + 1; i++)
                A[i] = (double *) malloc((n + 2) * sizeof(double));
        b = (double *) malloc((n + 2) * sizeof(double));
        for (i = 0; i <= n + 1; i++) {
                A[i][0] = 1;
                for (j = 1; j <= n; j++)
                        A[i][j] = A[i][j - 1] * x[i];
                A[i][n + 1] = (i % 2) ? -1 : 1;
                b[i] = evaluator_evaluate_x(evaluator, x[i]);
        }

        /* Brise se evaluator. */
        evaluator_destroy(evaluator);

        /* Poziva se Gauss-ova metoda radi resavanja sistema. */
        X = (double *) malloc((n + 2) * sizeof(double));
        gauss(n + 2, A, b, X);

        /* Oslobadjaju se matrica i vektor gornje strane gornjeg sistema. */
        for (i = 0; i <= n + 1; i++)
                free(A[i]);
        free(A);
        free(b);

        /* Ocitavaju se koeficijenti aproksimacionog polinoma i vrednost
         * najveceg odstupanja. */
        for (i = 0; i <= n; i++)
                c[i] = X[i];
        if (X[n + 1] >= 0)
                *alpha = 1, *L = X[n + 1];
        else
                *alpha = -1, *L = -X[n + 1];

        /* Oslobadja se polje sa resenjima sistema. */
        free(X);
}
```

4.55 *Za funkciju* $f(x) = (5 - x)^{-1}$ *odrediti polinom* $p(x)$ *što je moguće nižeg stepena tako da je*

$$\|f-p\| = \max_{x\in[-1,1]} |f(x)-p(x)| \le 5\cdot 10^{-3}.$$

Rešenje: Pokušajmo najpre ravnomernu aproksimaciju funkcije $f(x)$ polinomom nultog stepena. Kako je $f'(x) = (5-x)^{-2} > 0$ funkcija je rastuća pa je $m = \min_{x\in[-1,1]} f(x) = 1/6$ i $M = \max_{x\in[-1,1]} f(x) = 1/4$. Traženi polinom nultog stepena je

$$p_0(x) = \frac{m+M}{2} = \frac{5}{24}, \qquad \|f-p_0\| = 0.04.$$

Greška aproksimacije je veća od dozvoljene, pa ćemo konstruisati polinom najbolje ravnomerne aproksimacije prvog stepena $p_1(x) = c_0+c_1x$. Funkcija je konveksna, te su tačke Čebišev-ljeve alternanse krajevi intervala $x=-1$ i $x=1$, i tačka ekstrema $x=d$ funkcije $f(x)-p_1(x)$. Rešavanjem sistema jednačina po nepoznatima c_0, c_1, d i $L = \|f-p_1\|$ dobija se da je traženi polinom

$$p_1(x) = \frac{1}{24}x + \frac{\sqrt{6}}{12}, \qquad \|f-p_1\| = 4.2\cdot 10^{-3}.$$

4.56 *a) Ispitati da li je moguće aproksimirati funkciju $u(x) = \cos x$ polinomom drugog stepena tako da važi*

$$(*) \qquad \max_{x\in[-0.5,0.5]} |\cos x - p_2(x)| \le 0.00005.$$

b) U slučaju potvrdnog odgovora pod a) naći jedan polinom koji ispunjava uslov $(*)$. U slučaju negativnog odgovora pod a), odrediti najniži stepen polinoma za koji će važiti $(*)$ i bar jedan polinom koji ispunjava taj uslov.

Rešenje: a) Odredićemo polinom najbolje ravnomerne aproksimacije $p_2(x)$ funkcije $f(x) = \cos x$, jer je za proizvoljan polinom $p(x)$ drugog stepena

$$\max_{x\in[-0.5,0.5]} |\cos x - p(x)| \ge \max_{x\in[-0.5,0.5]} |\cos x - p_2(x)|.$$

Pošto je $\cos x$ parna funkcija, polinom je oblika $p_2(x) = c_0+c_2x^2$. Uvođenjem smene $t = x^2$, problem se svodi na nalaženje polinoma najbolje ravnomerne aproksimacije prvog stepena funkcije $\cos\sqrt{t}$ na intervalu $[0, 0.25]$. Zamenom stare promenljive dobija se da su traženi polinom i greška aproksimacije

$$p_2(x) = 0.9997 - 0.4897x^2, \qquad \max_{x\in[-0.5,0.5]} |\cos x - p_2(x)| = 0.0003.$$

Stoga ni jedan polinom drugog stepena ne ispunjava uslov $(*)$.

b) Zbog parnosti funkcije i simetričnosti intervala sledeći stepen aproksimacionog polinoma je četiri. Jedan takav polinom je onaj koji se dobija kao parcijalna suma razvoja funkcije $\cos x$ u red oko tačke 0,

$$p_4(x) = 1 - \frac{x^2}{2} + \frac{x^4}{24}.$$

Pošto je red alternativni, greška odsecanja je manja po apsolutnoj vrednosti od modula prvog izostavljenog člana

$$\max_{x\in[-0.5,0.5]} |\cos x - p_4(x)| \leq \frac{0.5^6}{720} = 0.000022,$$

te polinom $p_4(x)$ zadovoljava uslov $(*)$.

4.57 *Neka je $\mathcal{P}$ skup polinoma stepena n i $p_n(x) \in \mathcal{P}$ polinom najbolje ravnomerne aproksimacije za funkciju $f(x)$. Naći u skupu $\mathcal{P}$ polinom najbolje ravnomerne aproksimacije za funkciju $h(x) = f(x) + q_n(x)$, gde je $q_n(x)$ proizvoljan polinom stepena n.*

Rešenje: Neka je $E_n(f) = \|f - p_n\|$. Tada je

$$\|h - (p_n + q_n)\| = \|f + q_n - (p_n + q_n)\| = \|f - p_n\| = E_n(f).$$

Za proizvoljan polinom $p \in \mathcal{P}$ je, međutim,

$$\|h - p\| = \|f - (p - q_n)\| \geq E_n(f),$$

jer je $(p - q_n) \in \mathcal{P}$, a $E_n(f)$ je veličina najbolje aproksimacije za f u $\mathcal{P}$. Dakle, ne postoji polinom stepena n koji bolje ravnomerno aproksimira h od polinoma $p_n + q_n$. Stoga je $p_n + q_n$ polinom stepena n najbolje ravnomerne aproksimacije za funkciju $h(x) = f(x) + q_n(x)$.

4.58 *U klasi $\mathcal{C}[-1, 1]$ definisane su parna funkcija $f(x)$ i neparna funkcija $g(x)$. Neka je*

$$E_n(f) = \max_{[-1,1]} |f(x) - p_n(f; x)|,$$

gde je $p_n(f; x)$ polinom najbolje ravnomerne aproksimacije stepena n za funkciju $f(x)$. Dokazati da je

$$E_n(f + g) \geq \max\{E_n(f), E_n(g)\}.$$

Rešenje: Neka je $p_n(f + g; x)$ polinom najbolje ravnomerne aproksimacije za funkciju $f(x) + g(x)$, $p_n^p(f + g; x)$ polinom koga čine parni stepeni ovog polinoma, a $p_n^n(f + g; x)$ polinom koga čine neparni stepeni ovog polinoma. Funkcija $F(x) = f(x) - p_n^p(f + g; x)$ je parna funkcija, a $G(x) = g(x) - p_n^n(f + g; x)$ neparna funkcija. Ako je x_p tačka takva da je $|F(x_p)| = \max_{x\in[-1,1]} |F(x)| > 0$, onda je

$$|F(x_p) + G(x_p)| = |F(x_p)| + |G(x_p)|$$

ili

$$|F(-x_p) + G(-x_p)| = |F(-x_p)| + |G(-x_p)| = |F(x_p)| + |G(x_p)|,$$

jer je $F(x_p) = F(-x_p)$ i $G(x_p) = -G(-x_p)$ pa u jednoj od tačaka x_p ili $-x_p$ funkcije F i G imaju isti znak. Stoga je

$$E_n(f+g) \geq |F(x_p)| + |G(x_p)| \geq |F(x_p)| = \max_{x\in[-1,1]} |f(x) - p_n^p(f+g;x)| \geq E_n(f).$$

Analogno dokazujemo, ako je x_n tačka ekstrema funkcije $G(x)$, da je

$$E_n(f+g) \geq E_n(g),$$

odakle sledi tvrđenje iskazano u zadatku.

4.59 *Polinomom prvog stepena ravnomerno aproksimirati funkciju* $f(x) = x^2$ *na intervalu* $[0,1]$.

Rešenje: Traži se polinom $p_1(x) = c_0 + c_1 x$ takav da je $\max_{[0,1]} |x^2 - c_1 x - c_0|$ minimalan. To znači da se traži polinom drugog stepena najmanjeg odstupanja od nule na intervalu $[0,1]$. Na intervalu $[-1,1]$ to je polinom $\overline{T}_2(t) = t^2 - \frac{1}{2}$. Proizvoljni interval $[a,b]$ se može preslikati u interval $[-1,1]$ linearnom transformacijom $x = \frac{b-a}{2}t + \frac{b+a}{2}$, odnosno $t = \frac{2x-(b+a)}{b-a}$. Tako je Čebišev-ljev polinom reda n na intervalu $[a,b]$:

$$T_n^{[a,b]}(x) = T_n^{[-1,1]}\left(\frac{2x-(b+a)}{b-a}\right) = 2^{n-1}\left(\frac{2x-(b+a)}{b-a}\right)^n + \ldots = \frac{2^{2n-1}}{(b-a)^n}x^n + \ldots$$

a polinom najmanjeg odstupanja od nule na istom intervalu

$$\overline{T}_n^{[a,b]}(x) = \frac{(b-a)^n}{2^{2n-1}} T_n^{[a,b]}(x)$$

U datom primeru je $n = 2$, $a = 0$ i $b = 1$, pa je:

$$T_2^{[0,1]}(x) = 2\,(2x-1)^2 - 1 = 8x^2 - 8x + 1$$

i

$$\overline{T}_2^{[0,1]}(x) = \frac{1}{8}\,T_2^{[0,1]}(x) = x^2 - x + \frac{1}{8}$$

Odavde sledi da je $c_0 = -\frac{1}{8}$ i $c_1 = 1$.

4.60 *Između svih polinoma* $p_n(x)$ *stepena* n *čije su vrednosti u tački* ξ *van odsečka* $[-1,1]$ *jednake* η, *odrediti polinom sa najmanjim odstupanjem od nule na intervalu* $[-1,1]$.

Rešenje: Polinom

$$p_n(x) = c_0 + c_1 x + \cdots + c_n x^n = c_n\left(\frac{c_0}{c_n} + \frac{c_1}{c_n}x + \cdots + x^n\right), \quad c_n \neq 0,$$

će imati najmanje odstupanje od nule na intervalu $[-1,1]$ ako je

$$p_n(x) = c_n \overline{T}_n(x), \qquad \overline{T}_n(x) = 2^{1-n} T_n(x),$$

gde je $T_n(x) = \cos(n \arccos x)$ Čebišev-ljev polinom stepena n. Iz uslova $p_n(\xi) = \eta$ sledi da je $c_n = \eta/\overline{T}_n(\xi)$ ($\overline{T}_n(\xi) \neq 0$ jer su sve nule polinoma $T_n(x)$ u intervalu $[-1,1]$). Stoga je traženi polinom

$$p_n(x) = \frac{\eta}{\overline{T}_n(\xi)}\, \overline{T}_n(x) \qquad \text{i} \qquad \max_{[-1,1]} |p_n(x)| = \left| \frac{\eta}{\overline{T}_n(\xi)} \right| 2^{1-n}.$$

4.61 *Neka je $\mathcal{P}_2$ skup svih polinoma stepena ne većeg od dva i $\mathcal{X} = \{p \in \mathcal{P}_2,\, p(0) = 1\}$, $\mathcal{Y} = \{p \in \mathcal{P}_2,\, p'(0) = 1\}$. Naći sve polinome a) $p \in \mathcal{X}$, b) $p \in \mathcal{Y}$, koji ravnomerno najmanje odstupaju od nule na intervalu [-1,1].*

Rešenje: a) Da bi polinom $p(x) = ax^2 + bx + c$ zadovoljavao uslov $p(0) = 1$ mora biti $c = 1$. Preostale koeficijente odredimo tako da tačka $x = 0$ bude tačka maksimuma polinoma. Da bi to bila tačka ekstrema treba da je $b = 0$ ($p'(0) = 0$), a da bi bilo $|p(x)| = |ax^2 + 1| \leq 1$ kada $x \in [-1,1]$ treba da je $-2 \leq a \leq 0$. Dakle, polinomi najmanjeg odstupanja od nule koji pripadaju skupu $\mathcal{X}$ su

$$p(x) = ax^2 + 1, \qquad -2 \leq a \leq 0.$$

b) Polinom $p(x) = ax^2 + bx + c$ zadovoljava uslov $p'(0) = (2ax + b)_{x=0} = 1$ ako je $b = 1$, tj. ako je oblika $p(x) = ax^2 + x + c$. a i c mogu biti ma kakvi realni brojevi, pa i nula. Stoga je jedan od mogućih polinoma $p(x) = x$, a $\max_{x\in[-1,1]} |p(x)| = 1$. Dakle, najveće odstupanje traženih polinoma od nule ne može biti manje od jedan. Uslov $|p(\pm 1)| = |\pm 1 + (a + c)| \leq 1$ daje $c = -a$ što sužava klasu dopustivih polinoma na polinome oblika

$$p(x) = ax^2 + x - a = a[x + 1/(2a)]^2 - (1 + 4a^2)/(4a), \quad a \neq 0.$$

Ekstremna vrednost ovih polinoma je $p(x_e) = -(1 + 4a^2)/(4a)$ i postiže se u tački $x_e = -1/(2a)$.

(i) Ako je $|a| \leq 1/2$ tačka $x_e \notin [-1,1]$. Polinom $p(x)$ je monotona funkcija u intervalu $[-1,1]$ i ekstremne vrednosti, koje su jednake 1 ili -1, dostiže na svojim krajevima.

(ii) Ako je $|a| > 1/2$ tačka $x_e \in [-1,1]$. Ali tada je $\max_{x\in[-1,1]} |p(x)| = |p(x_e)| > 1$, jer zbog $(1-2|a|)^2 = 1+4a^2-4|a| > 0$ za $|a| > 1/2$ sledi da je $(1+4a^2)/(4|a|) > 1$.

Objedinjujući analizu (i) i (ii), zaključujemo da su traženi polinomi najmanjeg odstupanja od nule koji pripadaju skupu $\mathcal{Y}$

$$p(x) = ax^2 + x - a, \qquad |a| \leq \frac{1}{2}.$$

4.62 *Napisati Čebišev-ljev polinom $T_n^{[0,1]}(x)$ definisan na intervalu $[0,1]$ i rekurentnu formulu koju on zadovoljava.*

Rešenje: Čebišev-ljev polinom definisan na intervalu $[-1,1]$ je

$$T_n(t) = \cos(n \arccos t).$$

Smenom $x = (1+t)/2$ interval $[-1,1]$ se preslikava u interval $[0,1]$, pa će traženi polinom biti

$$T_n(t) = T_n(2x-1) = T_n^{[0,1]}(x) = \cos\big(n \arccos(2x-1)\big) = \cos n\theta, \qquad x = \cos^2\frac{\theta}{2}.$$

Rekurentna formula koju ovaj polinom zadovoljava izvodi se uvođenjem naznačene smene u rekurentnu formulu za Čebišev-ljeve polinome definisane na intervalu $[-1,1]$

$$T_0(t) = 1, \quad T_1(t) = t, \qquad T_{n+1}(t) = 2t\,T_n(t) - T_{n-1}(t) \quad n = 1, 2, \ldots.$$

Tako se dobija da je

$$T_0^{[0,1]}(x) = 1, \qquad T_1^{[0,1]}(x) = 2x - 1,$$
$$T_{n+1}^{[0,1]}(x) = 2(2x-1)T_n^{[0,1]}(x) - T_{n-1}^{[0,1]}(x), \quad n = 1, 2, \ldots.$$

4.63 *Za Čebišev-ljev polinom stepena n napisan u obliku*

$$T_n(x) = \sum_{k=0}^{n} c_k^{(n)} x^k,$$

pokazati da su nenula koeficijenti $c_k^{(n)}$ dati izrazom

$$c_k^{(n)} = 2^{k-1}\left(2\binom{(n+k)/2}{(n-k)/2} - \binom{(n+k)/2-1}{(n-k)/2}\right)(-1)^{(n-k)/2}.$$

Rešenje: Čebišev-ljev polinom parnog stepena je parna funkcija, a neparnog stepena neparna funkcija. Stoga su nenula koeficijenti $c_k^{(n)}$ oni kod kojih je k parno ako je n parno, odnosno k neparno kada je n neparno. To znači da je $(n \pm k)/2$ uvek ceo broj i da je $c_{n-1}^{(n)} = 0$. Osim toga je $c_n^{(n)} = 2^{n-1}$ i za n parno $c_0^{(n)} = (-1)^{n/2}$ (za n neparno je taj koeficijent nula), što je saglasno tvrđenju.

Za preostale koeficijente $c_k^{(n)}$, $k = 1, \ldots, n-2$, tvrđenje ćemo dokazati indukcijom. Za $n = 2$ izraz se neposredno proverava. Pretpostavimo da tvrđenje važi za koeficijente svih polinoma stepena manjeg od n i dokažimo da važi za koeficijente Čebišev-ljevog polinoma stepena n. Iz rekurentne veze koju zadovoljavaju Čebišev-ljevi polinomi (videti prethodni zadatak) sledi veza koeficijenata

$$c_k^{(n)} = 2\,c_{k-1}^{(n-1)} - c_k^{(n-2)}, \qquad k = 1, \ldots, n-2.$$

Na osnovu indukcijske hipoteze je onda

$$c_k^{(n)} = 2\cdot 2^{k-2}\left(2\binom{(n-1+k-1)/2}{(n-1-k+1)/2} - \binom{(n-1+k-1)/2-1}{(n-1-k+1)/2}\right)(-1)^{(n-k)/2}$$
$$- 2^{k-1}\left(2\binom{(n-2+k)/2}{(n-2-k)/2} - \binom{(n-2+k)/2-1}{(n-2-k)/2}\right)(-1)^{(n-2-k)/2}$$

tj.

$$c_k^{(n)} = 2^{k-1}\Bigg(2\left(\binom{(n+k-2)/2}{(n-k)/2} + \binom{(n+k-2)/2}{(n-k-2)/2}\right)$$
$$-\left(\binom{(n+k-2)/2-1}{(n-k)/2} + \binom{(n+k-2)/2-1}{(n-k-2)/2}\right)\Bigg)(-1)^{(n-k)/2}$$

Tvrđenje neposredno sledi jer je

$$\binom{(n+k)/2-1}{(n-k)/2} + \binom{(n+k)/2-1}{(n-k)/2-1} = \binom{(n+k)/2}{(n-k)/2}$$
$$\binom{(n+k)/2-2}{(n-k)/2} + \binom{(n+k)/2-2}{(n-k)/2-1} = \binom{(n+k)/2-1}{(n-k)/2}$$

4.64 *Da li postoji polinom drugog stepena $p_2(x)$ takav da je*

$$\max_{x\in[-1,1]} |x^4 - p_2(x)| < 0.1\,?$$

Obrazložiti odgovor.

Rešenje: Polinom koji najmanje odstupa od date funkcije u uniformnoj normi je njen polinom najbolje ravnomerne aproksimacije, te ćemo odrediti takav polinom drugog stepena i oceniti grešku aproksimacije. Kako je $f(x) = x^4$ parna funkcija, to će i ovaj polinom biti parna funkcija, tj. oblika $p_2(x) = c_0 + c_2x^2$. Dakle, c_0 i c_2 treba odrediti tako da je

$$\max_{x\in[-1,1]} |x^4 - c_2x^2 - c_0| = \min,$$

tj. odrediti polinom četvrtog stepena koji najmanje odstupa od nule na intervalu $[-1,1]$. Polinom sa koeficijentom jedan uz x^4 koji ima ovo svojstvo je $\overline{T}_4(x) = 2^{-3}T_4(x)$, gde je $T_4(x) = 8x^4 - 8x^2 + 1$ Čebišev-ljev polinom četvrtog stepena. Stoga treba da je

$$x^4 - c_2x^2 - c_0 \equiv x^4 - x^2 + \frac{1}{8}, \qquad x\in[-1,1],$$

pa je $c_2 = 1$ i $c_0 = -1/8$. Greška aproksimacije je

$$\max_{x\in[-1,1]} |\overline{T}_4(x)| = 2^{-3}\max_{x\in[-1,1]} |T_4(x)| = 0.125 > 0.1,$$

te ovaj polinom ne zadovoljava zadatu tačnost. Dakle, ne postoji polinom drugog stepena koji ravnomerno aproksimira funkciju x^4 sa greškom manjom od 0.1.

4.65 *Odrediti linearnu funkciju $p_1(x) = c_0 + c_1 x$ koje najbolje aproksimira funkciju $f(x) = x^2$ na intervalu $[0,1]$ tako da je*

$$a) \qquad E_1(f) = \int_0^1 \big(f(x) - p_1(x)\big)^2\, dx,$$

$$b) \qquad E_1(f) = \int_0^1 x(1-x)\big(f(x) - p_1(x)\big)^2\, dx,$$

$$c) \qquad E_1(f) = \big(f(0) - p_1(0)\big)^2 + \big(f(0.5) - p_1(0.5)\big)^2 + \big(f(1) - p_1(1)\big)^2,$$

$$d) \qquad E_1(f) = \max_{x\in[0,1]} |f(x) - p_1(x)|,$$

minimalno.

Rešenje: Element najbolje aproksimacije se traži u prostoru funkcija čiji je bazis $g_i(x) = x^i$, $i = 0, 1$. U tačkama (a), (b) i (c) tražimo element najbolje aproksimacije u Hilbertovom prostoru, jer je norma definisana skalarnim proizvodom. Stoga se koeficijenti aproksimacije c_0 i c_1 određuju kao rešenja sistema linearnih jednačina

$$\sum_{i=0}^{1} c_i(g_i, g_j) = (f, g_j), \qquad j = 0, 1,$$

gde je a) $(f,g) = \int_0^1 f(x)g(x)\,dx$, b) $(f,g) = \int_0^1 x(1-x)f(x)g(x)\,dx$ i c) $(f,g) = f(0)g(0) + f(0.5)g(0.5) + f(1)g(1)$. Tako se dobijaju linearne aproksimacije i greške

a)	$p_1(x) = x - 1/6,$	$E_1(f) = 0.006$
b)	$p_1(x) = x - 1/5,$	$E_1(f) = 0.13$
c)	$p_1(x) = x - 1/12$	$E_1(f) = 0.042$

U tački (d) određujemo element najbolje ravnomerne aproksimacije. Polinom $x^2 - c_1 x - c_0$, prema uslovu (d) treba da bude polinom najmanjeg odstupanja od nule na intervalu $[0,1]$ sa koeficijentom 1 uz najviši stepen. Stoga je

$$x^2 - c_1 x - c_0 = \overline{T}_2^{[0,1]}(x) = 2^{-3} T_2(2x-1) = x^2 - x + \frac{1}{8},$$

odakle sledi da je $c_0 = -1/8$ i $c_1 = 1$. Dakle, linearna aproksimacija i greška su

$$d) \qquad p_1(x) = x - \frac{1}{8}, \qquad E_1(f) = 0.13.$$

5

Sistemi linearnih jednačina

Rešava se sistem jednačina oblika

$$A\mathbf{x} = \mathbf{b} \qquad A = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mm} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

gde je A regularna matrica.

5.1 Gauss-ove metode

Gauss-ova metoda eliminacije bez ili sa izborom glavnog elementa se sastoji u transformaciji polaznog sistema linearnih jednačina u ekvivalentan sistem sa gornje trougaonom matricom

$$U\mathbf{x} = \mathbf{c}, \qquad U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ 0 & u_{22} & \dots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{mm} \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}$$

Transformacije se sastoje u dodavanju izabrane jednačine tekućeg sistema pomnožene odgovarajućim koeficijentima ostalim jednačinama radi anuliranja koeficijenata uz jednu promenljivu u njima. Dobijeni sistem $U\mathbf{x} = \mathbf{c}$ se direktno rešava

$$x_m = \frac{c_m}{u_{mm}}, \qquad x_k = \frac{1}{u_{kk}} \left(c_k - \sum_{j=k+1}^{m} u_{kj} x_j \right), \quad k = m-1, \dots, 1.$$

Gauss–Jordan-ovom metodom se na način opisan u Gauss-ovoj metodi eliminacije sistem svodi na sistem sa dijagonalnom matricom tako što se u svakom koraku transformišu sve jednačine.

5.1 *Dat je sistem*

$$\begin{aligned} 6x_1 + 2x_2 + 2x_3 &= -2 \\ 2x_1 + 2/3x_2 + 1/3x_3 &= 1 \\ x_1 + 2x_2 - x_3 &= 0 \end{aligned}$$

Proveriti da li je njegovo tačno rešenje $x_1 = 2.6$, $x_2 = -3.8$, $x_3 = -5.0$. *Računajući sa četiri značajne cifre rešiti dati sistem Gauss-ovom metodom eliminacije: a) bez biranja glavnog elementa, b) sa izborom glavnog elementa.*

Rešenje: U tabelama koje slede prikazani su rezultati izračunavanja

a) Gauss-ova metoda eliminacije bez izbora glavnog elementa

m_i	a_{i1}	a_{i2}	a_{i3}	b_i
	6	2	2	−2
0.3333	2	0.6667	0.3333	1
0.1667	1	2	−1	0
*	1	0.3333	0.3333	−0.3333
		0.0001	−0.3333	1.667
$1667 \cdot 10$		1.667	−1.333	0.3333
*		1	−3333	1667
			5555	−2779
*			1	−0.5002
		1		−0.1666
	1			−0.1111

b) Gauss-ova metoda eliminacije sa izborom glavnog elementa

m_i	a_{i1}	a_{i2}	a_{i3}	b_i
*	<u>6</u>	2	2	−2
0.3333	2	0.6667	0.3333	1
0.1667	1	2	−1	0
0.0001		0.0001	−0.3333	1.667
*		<u>1.667</u>	−1.333	0.3334
*			<u>−0.3332</u>	1.667
			1	−5.003
		1		−3.801
	1			2.601

($*$ u koloni m_i označava jednačine trougaonog sistema.) Očigledno je da rezultat određen Gaussovom metodom eliminacije bez izbora glavnog elementa (a) nije tačan. Primenom iste metode sa izborom glavnog elementa (b) dobija se tačan rezultat na dve decimale. Razlog tome je gubitak sigurnih cifara zbog oduzimanja bliskih brojeva, što je posledica loše uslovljenosti matrice sistema ($\text{cond}(A) = 40$).

5.2 *Dat je sistem*

$$0.50x_1 + 1.1x_2 + 3.1x_3 = 6.0$$
$$2.0x_1 + 4.5x_2 + 0.36x_3 = 0.020$$
$$5.0x_1 + 0.96x_2 + 6.5x_3 = 0.96$$

Računajući sa tri značajne cifre rešiti dati sistem Gauss-ovom metodom eliminacije: a) bez biranja glavnog elementa, b) sa izborom glavnog elementa.

Rešenje: U tabelama koje slede prikazani su rezultati izračunavanja
a) Gauss-ova metoda eliminacije bez izbora glavnog elementa

m_i	a_{i1}	a_{i2}	a_{i3}	b_i
	0.50	1.1	3.1	6.0
4	2.0	4.5	0.36	0.020
10	5.0	0.96	6.5	0.96
$*$	1	2.20	6.20	12.0
		0.100	-12.0	-24.0
-100		-10.0	-24.5	-59.0
$*$		1	-120	-240
			$-122 \cdot 10$	$-246 \cdot 10$
$*$			1	2.02
		1		2.40
	1			-5.80

b) Gauss-ova metoda eliminacije sa izborom glavnog elementa

m_i	a_{i1}	a_{i2}	a_{i3}	b_i
0.477	0.50	1.1	3.1	6.0
0.055	2.0	4.5	0.36	0.020
$*$	5.0	0.96	<u>6.5</u>	0.96
0.144	-1.88	0.642		5.54
$*$	1.72	<u>4.45</u>		-0.0328
$*$	<u>-2.13</u>			5.54
	1			-2.60
		1		1.00
			1	2.00

($*$ u koloni m_i označava jednačine trougaonog sistema.) Rezultat određen Gaussovom metodom eliminacije sa izborom glavnog elementa (b) je tačan. Ista metoda bez izbora glavnog elementa, primenjena na sistem čija je matrica loše uslovljena, daje pogrešan rezultat zbog računanja sa malim brojem sigurnih cifara.

5.3 *Sa tačnošću* 10^{-3} *Gauss-ovom metodom sa izborom glavnog elementa rešiti sistem*

$$\begin{aligned} 8.23x_1 + 3.67x_2 - 2.08x_3 &= 7.34 \\ 2.55x_1 - 5.17x_2 - 1.62x_3 &= -1.93 \\ 3.02x_1 + 2.18x_2 + 7.81x_3 &= 9.23 \end{aligned}$$

i izračunati determinantu matrice sistema.

Rešenje: Rezultati izračunavanja prikazani su sledećom tabelom

m_i	a_{i1}	a_{i2}	a_{i3}	b_i
	$\underline{8.23}$	3.67	-2.08	7.34
0.30984	2.55	-5.17	-1.62	-1.93
0.36695	3.02	2.18	7.81	9.23
-0.11379		-6.30712	-0.97553	-4.20424
		0.83329	$\underline{8.57326}$	6.53659
		$\underline{-6.21230}$		-3.46046
		1		0.55703
			1	0.70830
	1			0.82247

(podvučeni brojevi predstavljaju glavne elemente u svakom koraku). Rešenja sistema su

$$x_1 = 0.822, \qquad x_2 = 0.557, \qquad x_3 = 0.708,$$

a determinanta

$$\det = 8.23 \cdot (-6.21230) \cdot 8.57326 = -438.327.$$

5.4 *Gauss-Jordan-ovom metodom odrediti inverznu matricu matrice*

$$A = \begin{pmatrix} 1 & 3 & 9 & 1 \\ 1 & 1 & 1 & 5 \\ 2 & 7 & 3 & 2 \\ 4 & 2 & 1 & 1 \end{pmatrix}$$

Računati na pet decimala.

Rešenje: Inverzna matrica X date matrice A dimenzije m se određuje istovremenim rešavanjem m sistema linearnih jednačina

$$A \cdot X = I \qquad \longrightarrow \qquad A\mathbf{x}_k = \mathbf{e}_k, \quad k = 1, \ldots, m,$$

gde su $\mathbf{x}_k = (x_{1k}, \ldots, x_{mk})^\top$ vektori kolona matrice X i $\mathbf{e}_k = (0, \ldots, 1, \ldots, 0)^\top$ vektori kolona jedinične matrice. Transformacija matrice A Gauss–Jordan-ovom metodom prikazana je tabelom koja sledi

m_i	a_{i1}	a_{i2}	a_{i3}	a_{i4}
	1	3	<u>9</u>	1
0.11111	1	1	1	5
0.33333	2	7	3	2
0.11111	4	2	1	1
0.50000	1	3	9	1
0.11111	0.88889	0.66667		4.88889
	1.66667	<u>6.00000</u>		1.66667
0.27778	3.88889	1.66667		0.88889
0.03543	0.16667		9	0.16667
	0.70370			<u>4.70370</u>
0.35433	1.66667	6.00000		1.66667
0.09055	3.42593			0.42593
0.04216	0.14174		9	
0.20930	0.70370			4.70370
0.42155	1.41733	6.00000		
	<u>3.36221</u>			
			9	
				4.70370
D		6.00000		
	3.36221			

Iste transformacije se vrše i na vrstama jedinične matrice i kao rezultat se dobija matrica

$$B = \begin{pmatrix} 1.16979 & -0.03161 & -0.48477 & -0.04216 \\ -0.07160 & 1.01895 & -0.05508 & -0.20930 \\ -0.30210 & -0.31616 & 1.15223 & -0.42155 \\ -0.01181 & -0.09055 & -0.26772 & 1 \end{pmatrix}$$

čiji vektori kolona su desne strane 4 sistema sa dijagonalnom matricom $D\,X = B$. Rešenja ovih sistema su vektori kolona matrice $X = A^{-1}$,

$$A^{-1} = \begin{pmatrix} -0.00351 & -0.02693 & -0.07963 & 0.29742 \\ -0.05035 & -0.05269 & 0.19204 & -0.07026 \\ 0.12998 & -0.00351 & -0.05386 & -0.00468 \\ -0.01522 & 0.21663 & -0.01171 & -0.04450 \end{pmatrix}$$

5.5 *Gauss-ovom metodom sa izborom glavnog elementa odrediti vrednost determinante i inverznu matricu matrice*

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

Računati na pet decimala.

Rešenje: Algoritmom opisanim u prethodnom zadatku, sa tom razlikom što se jednačina u kojoj je jednom izabran glavni element više ne transformiše, dobija se inverzna matrica

$$A^{-1} = \begin{pmatrix} 24.99303 & -40.98853 & 9.99713 & -5.99836 \\ -40.98844 & 67.98097 & -16.99524 & 9.99728 \\ 9.99711 & -16.99524 & 4.99882 & -2.99933 \\ -5.99830 & 9.99720 & -2.99931 & 1.99961 \end{pmatrix}$$

Determinanta je

$$\det(A) = 10 \cdot 5.1 \cdot 1.33332 \cdot 0.01471 = 1.00027.$$

C Implementirana je Gauss-ova metoda eliminacije sa delimičnim pivotiranjem. Na početku procedure vrši se normalizacija matrice sistema. Potom se u svakom koraku bira pivotirajući element među elementima tekuće kolone na i ispod tekućeg reda. Ovaj element se, ako je potrebno, zamenom vrsta matrice sistema dovodi na glavnu dijagonalu i potom se anuliraju svi elementi ispod glavne dijagonale u tekućoj koloni. Sve transformacije matrice sistema prate odgovarajuće transformacije vektora desne strane sistema. Na kraju se rešenja sistema direktno izračunavaju iz dobijenog sistema sa gornje trougaonom matricom.

Napomena: Algoritam Gauss-ove metode eliminacije, prilagođen za rešavanje sistema jednačina sa trodijagonalnom matricom, dat je u uvodu kao ilustracija načina izlaganja materijala.

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <matheval.h>

/* Definicija makroa za swap-ovanje dve velicine zadatog tipa. */
#define swap(type,a,b) {type tmp; tmp=a; a=b; b=tmp;}

/*
 * Funkcija gauss() implementira Gauss-ovu metodu eliminacije sa
 * delimicnim pivotiranjem za resavanje sistema od n linearnih jednacina
 * sa n nepoznatih.  Argumenti funkcije su:
 *   n - red sistema
 *   A - matrica sistema
 *   b - vektor sa desnim stranama jednacina
```

```
 *   x - vektor u koji ce biti smestena resenja sistema
 * Red sistema mora biti veci od 0.
 */
void
gauss(int n, double **A, double *b, double *x)
{
        int             i,
                        j;      /* Brojaci u petljama. */

        /* Provjerava se da li je red sistema veci od 0. */
        assert(n > 0);

        /* Normalizuju se jednacine sistema. */
        for (i = 0; i < n; i++) {
                double          max;    /* Maksimalni po modulu
                                         * koeficijent za tekucu
                                         * jednacinu. */
                double          mul;    /* Broj kojim se mnozi tekuca
                                         * jednacina. */

                /* Odredjuje se maksimalni po modulu koeficijent u tekucoj
                 * jednacini. */
                max = 0;
                for (j = 0; j < n; j++)
                        if (max < fabs(A[i][j]))
                                max = fabs(A[i][j]);

                /* Deli se tekuca jednacina sa ovako izracunatom vrednoscu
                 */
                mul = 1 / max;
                for (j = 0; j < n; j++)
                        A[i][j] *= mul;
                b[i] *= mul;
        }

        /* Kolonu po kolonu, svodi se matrica sistema na gornje trougaonu.
         */
        for (j = 0; j < n - 1; j++) {
                double          pivot;  /* Vrednost elementa koji je
                                         * izabran kao pivotirajuci. */
                int             pivot_index;    /* Indeks pivotirajuceg
                                                 * elementa. */

                /* Odredjuje se vrednost i indeks pivotirajuceg elementa. */
                pivot = 0;
                for (i = j; i < n; i++)
                        if (fabs(pivot) < fabs(A[i][j]))
                                pivot = A[i][j], pivot_index = i;

                /* Dovodi se pivotirajuci elemenat na glavnu dijagonalu
                 * matrice sistema tako sto se po potrebi swap-uju redovi
                 * te matrice. */
                if (pivot_index != j) {
                        i = j;
                        for (j = 0; j < n; j++)
                                swap(double, A[i][j], A[pivot_index][j]);
                        swap(double, b[i], b[pivot_index]);
```

```
                j = i;
        }

        /* Anuliraju se elementi ispod glavne dijagonale u tekucoj
         * koloni. */
        for (i = j + 1; i < n; i++)
                if (A[i][j] != 0) {
                        double          mul;    /* Mnozilac za
                                                 * kombinovanje
                                                 * osnovnog i
                                                 * tekuceg reda. */
                        int             base;   /* Indeks osnovnog
                                                 * reda. */

                        /* Pamti se indeks osnovnog reda. */
                        base = j;

                        /* Izracunava se mnozilac kojim se mnozi
                         * osnovni red da bi se dodao tekucem
                         * redu. */
                        mul = A[i][j] / pivot;

                        /* Anulira se element ispod glavne
                         * dijagonale u tekucem redu. */
                        A[i][j] = 0;

                        /* Kombinuju se osnovni i tekuci red. */
                        for (j++; j < n; j++)
                                A[i][j] -= mul * A[base][j];
                        b[i] -= mul * b[base];

                        /* Vraca se originalna vrednost
                         * promenljive j. */
                        j = base;
                }
    }

    /* Resenja sistema se direktno izracunavaju iz gornje trougane
     * matrice sistema i vektora desnih strana jednacina. */
    x[n - 1] = b[n - 1] / A[n - 1][n - 1];
    for (i = n - 2; i >= 0; i--) {
            x[i] = b[i];
            for (j = n - 1; j > i; j--)
                    x[i] -= A[i][j] * x[j];
            x[i] /= A[i][i];
    }
}
```

5.2 Trougaona dekompozicija

LU dekompozicijom se matrica predstavlja proizvodom dve trougaone matrice,

$$A = L \cdot U, \qquad L = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & \dots & 1 \end{pmatrix} \qquad U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ 0 & u_{22} & \dots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{mm} \end{pmatrix}$$

gde je

$$u_{1k} = a_{1k}, \qquad u_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk}, \quad k = i, \dots, m, \qquad i = 2, \dots, m.$$

$$l_{k1} = \frac{a_{k1}}{u_{11}}, \qquad l_{ki} = \frac{1}{u_{ii}} \left(a_{ki} - \sum_{j=1}^{i-1} l_{kj} u_{ji} \right), \quad k = i+1, \dots, m,$$

Sada se rešenje sistema linearnih jednačina $A\mathbf{x} = \mathbf{b}$ nalazi rešavanjem dva sistema sa trougaonim matricama

$$L\mathbf{y} = \mathbf{b}, \qquad U\mathbf{x} = \mathbf{y}$$

Determinanta matrice A je

$$\det(A) = u_{11} \dots u_{mm}.$$

Cholesky dekompozicija je trougaona dekompozicija pozitivno definisane matrice,

$$A = L \cdot L^*, \qquad L = \begin{pmatrix} l_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ l_{m1} & \dots & l_{mm} \end{pmatrix}$$

Elementi matrice L su određeni formulama

$$l_{11} = \sqrt{a_{11}}, \qquad l_{i1} = \frac{a_{i1}}{l_{11}},$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} |l_{ik}|^2}, \qquad l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} \overline{l}_{jk} \right) \qquad 1 < j < i \leq m.$$

Rešenje sistema linearnih jednačina $A\mathbf{x} = \mathbf{b}$ nalazi se rešavanjem dva sistema sa trougaonim matricama

$$L\mathbf{y} = \mathbf{b}, \qquad L^*\mathbf{x} = \mathbf{y},$$

a determinanta matrice A je

$$\det(A) = (l_{11} \dots l_{mm})^2$$

5.6 *LU dekompozicijom rešiti sistem*

$$\begin{aligned} 2x_1 - 4x_2 - 3.25x_3 + x_4 &= 4.84 \\ 3x_1 - 3x_2 - 4.3x_3 + 8x_4 &= 8.89 \\ x_1 - 5x_2 + 3.3x_3 - 20x_4 &= -14.01 \\ 2.5x_1 - 4x_2 + 2x_3 - 3x_4 &= -20.29 \end{aligned}$$

i izračunati determinantu matrice sistema. Računati sa pet decimala.

Rešenje: Rezultati su dati u tabeli koja sledi

a_{i1}	a_{i2}	a_{i3}	a_{i4}	b_i
2	−4	−3.25	1	4.84
3	−3	−4.3	8	8.89
1	−5	3.3	−20	−14.01
2.5	−4	2	−3	−20.29
1 | 2	−4	−3.25	1.	4.84
1.5	1 | 3	0.575	6.5	1.63
0.5	−1	1 | 5.5	−14	−14.8
1.25	0.33333	1.06742	1 | 8.52730	−11.08551
			1	−1.30000
		1		−6.00001
	1			4.51001
1				2.34000

Rešenje sistema je

$$x_1 = 2.34, \qquad x_2 = 4.51, \qquad x_3 = -6, \qquad x_4 = -1.3,$$

a determinanta matrice sistema je

$$\det = 2 \cdot 3 \cdot 5.5 \cdot 8.52730 = 281.4009.$$

Može se računati i na drugi način:

a_{i1}	a_{i2}	a_{i3}	a_{i4}	b_i
2	−4	−3.25	1	4.84
3	−3	−4.3	8	8.89
1	−5	3.3	−20	−14.01
2.5	−4	2	−3	−20.29
2 | 1	−2	−1.625	0.5	2.42
3	3 | 1	0.19167	2.16667	0.54333
1	−3	5.50000 | 1	−2.54545	−2.69091
2.5	1	5.87083	8.52723 | 1	−1.30001
			1	−1.30001
		1		−6.00002
	1			4.51005
1				2.34006

LU dekompozicijom je razlaganje izvršeno tako da su jedinice na dijagonali gornje trougaone matrice U. Za izračunavanje elemenata matrica L i U korišćene su sledeće formule:

$$l_{k1} = a_{k1}, \qquad l_{ki} = a_{ki} - \sum_{j=1}^{i-1} l_{kj} u_{ji}, \quad 1 < i \le k \le m,$$

$$u_{1k} = \frac{a_{1k}}{l_{11}}, \qquad u_{ik} = \frac{1}{l_{ii}} \left(a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk} \right), \quad 1 < i < k \le m.$$

5.7 *Sa tačnošću* 10^{-4} *Cholesky dekompozicijom rešiti sistem*

$$\begin{aligned} 3.1x_1 + 1.5x_2 + \quad x_3 &= 10.83 \\ 1.5x_1 + 2.5x_2 + 0.5x_3 &= 9.20 \\ 1.0x_1 + 0.5x_2 + 4.2x_3 &= 17.10 \end{aligned}$$

i izračunati determinantu matrice sistema.

Rešenje: Ovom metodom se vrši dekompozicija matrice sistema

$$A = L\,L^*, \qquad \text{gde je} \quad L = \begin{pmatrix} 1.76068 & 0 & 0 \\ 0.85194 & 1.33199 & 0 \\ 0.56796 & 0.01211 & 1.96908 \end{pmatrix}$$

Problem se razlaže na rešavanje dva sistema sa trougaonim matricama

$$L\mathbf{y} = \mathbf{b}, \qquad L^*\mathbf{x} = \mathbf{y},$$

gde je $\mathbf{b}$ vektor desne strane datog sistema. Rešenje prvog je

$$\mathbf{y} = \begin{pmatrix} 6.15103 & 2.97276 & 6.89178 \end{pmatrix}^\top,$$

a rešenje drugog trougaonog sistema, što je ujedno i rešenje polaznog sistema, je

$$\mathbf{x} = \begin{pmatrix} 1.3000 & 2.2000 & 3.5000 \end{pmatrix}^\top.$$

Determinanta matrice sistema je

$$\det = (1.76068 \cdot 1.33199 \cdot 1.96908)^2 = 21.3250.$$

5.8 *Cholesky dekompozicijom rešiti sistem*

$$\begin{aligned} 1.00x_1 + 0.42x_2 + 0.54x_3 + 0.66x_4 &= 0.3 \\ 0.42x_1 + 1.00x_2 + 0.32x_3 + 0.44x_4 &= 0.5 \\ 0.54x_1 + 0.32x_2 + 1.00x_3 + 0.22x_4 &= 0.7 \\ 0.66x_1 + 0.44x_2 + 0.22x_3 + 1.00x_4 &= 0.9 \end{aligned}$$

Računati sa pet decimala.

Rešenje: Matrica sistema se predstavlja proizvodom $A = L\,L^*$, pa se problem svodi na rešavanje dva trougaona sistema $L\mathbf{y} = \mathbf{b}$ i $L^*\mathbf{x} = \mathbf{y}$. Vektor $\mathbf{b}$ je vektor desne strane sistema, a

$$L = \begin{pmatrix} 1.00 & 0 & 0 & 0 \\ 0.42 & 0.90752 & 0 & 0 \\ 0.54 & 0.10270 & 0.83537 & 0 \\ 0.66 & 0.17939 & -0.18533 & 0.70560 \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} 0.30 \\ 0.41211 \\ 0.59336 \\ 1.04597 \end{pmatrix}$$

Rešenje sistema je

$$x_1 = -1.25778, \qquad x_2 = 0.04348, \qquad x_3 = 1.03917, \qquad x_4 = 1.48238.$$

MATLAB

LU dekompozicija	`[L, U, P] = lu(A)`
Cholesky dekompozicija	`R = chol(A)`

MATLAB raspolaže i velikim brojem funkcija za različite oblike dekompozicija matrica.

LU dekompozicija matrice se izvodi korišćenjem funkcije `lu`. Najčešći oblik korišćenja ove funkcije je `[L, U, P] = lu(A)`, pri čemu je `A` proizvoljna kvadratna matrica, `L` je donje-trougaona sa jedinicama na dijagonali, `U` je gornje trougaona matrica, dok je `P` permutaciona matrica, pri čemu važi $P \cdot A = L \cdot U$.

Cholesky dekompozicija matrice se izvodi korišćenjem funkcije `chol`. Ukoliko je `A` simetrična, pozitivno-definitna matrica, poziv `R = chol(A)` vraća gornje-trougaonu matricu `R` tako da je $A = R^t \cdot R$.

5.3 Numerička stabilnost

Norme vektora i njima saglasne norme matrica koje se najčešće koriste su:

$\|\mathbf{x}\|_\infty = \max\limits_{1\le k\le m} \|x_k\|,$	$\|A\|_\infty = \max\limits_{1\le j\le m} \sum\limits_{k=1}^{m} \|a_{jk}\|$	uniformna
$\|\mathbf{x}\|_1 = \sum\limits_{k=1}^{m} \|x_k\|,$	$\|A\|_1 = \max\limits_{1\le k\le m} \sum\limits_{j=1}^{m} \|a_{jk}\|$	apsolutna
$\|\mathbf{x}\|_2 = \sum\limits_{k=1}^{m} \|x_k\|^2,$	$\|A\|_2 = \sum\limits_{j=1}^{m}\sum\limits_{k=1}^{m} \|a_{jk}\|^2$	euklidska

Uslovljenost regularne matrice je

$$\mathrm{cond}(A) = \|A\|\,\|A^{-1}\|, \qquad 1 \leq \mathrm{cond(A)} < \infty,$$

gde je $\|\cdot\|$ proizvoljna norma matrice. Uslovljenost singularne matrice je

$$\mathrm{cond}(A) \overset{\text{def}}{=} \infty \qquad \big(\det(A) = 0\big),$$

MATLAB

Norme vektora	`n = norm(x, p);`
	`p = 1, 2, 3, ..., Inf`
Norme matrice	`n = norm(A, p);`
	`p = 1, 2, Inf, 'Fro'`
Uslovljenost matrice	`c = cond(A, p)`
	`p = 1, 2, Inf, 'Fro'`

MATLAB funkcija `norm` izračunava normu vektora odnosno matrice. Od normi vektora, na raspolaganju su uobičajene p-norme, dok su od matričnih normi na raspolaganju 1-norma, 2-norma, ∞-norma i tzv. Frobenijusova norma jednaka sumi kvadrata svih elemenata matrice.

5.9 *Izračunati tačno rešenje sistema linearnih jednačina*

$$A\mathbf{x} = \mathbf{b}, \qquad \textit{gde je} \quad A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix}$$

Zatim, izračunati rešenje sa istom matricom sistema A i vektorom desne strane $\mathbf{b} + \Delta\mathbf{b}$, *gde je* $\Delta\mathbf{b} = \begin{pmatrix} 0.01 & 0 \end{pmatrix}^\top$.

Korišćenjem MATLAB*-a dati grafičku interpretaciju datih sistema linearnih jednačina i objasniti razliku u rešenjima.*

Rešenje: Rešavanje datih sistema linearnih jednačina odgovara pronalaženju preseka dve prave. Prave se mogu skicirati korišćenjem narednog skripta.

```
A =  [4.1    2.8; ...
      9.7    6.6];
b  = [ 4.1;  9.7];
db = [ 0.01; 0];

% Resavamo sisteme
x  = A\b; x1 = A\(b+db); dx = x1 - x;

disp('Uslovljenost matrice A');
cond(A, Inf)
disp('Relativna greska vektora b');
norm(db, Inf)/norm(b, Inf)
disp('Relativna greska resenja x');
```

```
norm(dx, Inf)/norm(x, Inf)

% Iscrtavamo prave
X   = linspace(0.2, 1.2);              % x in [0.2, 1.2]
Y1  = (b(1)  - A(1,1)*X)/A(1, 2);      % 4.1*x + 2.8*y = 4.1
Y2  = (b(2)  - A(2,1)*X)/A(2, 2);      % 9.7*x + 6.6*y = 9.7
Y1p = (b(1)+db(1) - A(1,1)*X)/A(1,2);  % 4.1*x + 2.8*y = 4.11
hold on, plot(X, Y1, 'b', X, Y2, 'r', X, Y1p, 'k')
plot(x(1), x(2), 'o', x1(1), x1(2), 'o')
```

Rešenja sistema $A\mathbf{x} = \mathbf{b}$ i $A(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$ su

$$\mathbf{x} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top, \qquad \mathbf{x} + \Delta\mathbf{x} = \begin{pmatrix} 0.34 & 0.97 \end{pmatrix}^\top.$$

Relativne greške vektora desne strane $\mathbf{b}$ i rešenja $\mathbf{x}$ izražene u uniformnoj normi su

$$\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = 10^{-3}, \qquad \frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} = 0.97.$$

Dakle, sistem je nestabilan jer maloj promeni vektora desne strane odgovara velika promena rešenja. Nestabilnost je posledica loše uslovljenosti matrice sistema

$$\mathrm{cond}(A) = \|A\|\,\|A^{-1}\| = 2249.4$$

i poznate ocene relativne greške rešenja

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathrm{cond}(A)\,\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

Ovako velika uslovljenost matrice sistema je posledica činjenice da su prave čiji se presek pronalazi skoro paralelne.

5.10 *Dati su sistemi linearnih jednačina*

$$\begin{aligned} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 &= 1 & 1.00x_1 + 0.50x_2 + 0.33x_3 &= 1 \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 &= 0 & 0.50x_1 + 0.33x_2 + 0.25x_3 &= 0 \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 &= 0 & 0.33x_1 + 0.25x_2 + 0.20x_3 &= 0 \end{aligned}$$

Izračunati rešenja oba sistema, pri čemu kod rešavanja drugog sistema zaokrugliti brojeve na dve decimale. Objasniti razliku među rešenjima.

Rešenje: Rešenje prvog sistema je

$$x_1 = 9, \qquad x_2 = -36, \qquad x_3 = 30,$$

a rešenje drugog sistema je

$$x_1 = 7.02, \qquad x_2 = -23.25, \qquad x_3 = 17.$$

Koeficijenti drugog sistema su približne vrednosti koeficijenata prvog sistema uzete sa tačnošću $5 \cdot 10^{-3}$. S obzirom da je matrica sistema (Hilbertova matrica) loše uslovljena, $\text{cond}(A) = 748$, relativna greška rešenja drugog sistema je velika, približno 37% (vidi prethodni zadatak).

5.11 *Fibonacci-jev niz se generiše diferencnom jednačinom*

$$f_0 = 0, \qquad f_1 = 1, \qquad f_n = f_{n-1} + f_{n-2}, \quad n = 2, 3, \ldots.$$

a) Dokazati da je

$$f_n f_{n+2} - f_{n+1}^2 = (-1)^{n+1}, \quad n = 0, 1, \ldots.$$

b) Dokazati da sistem

$$f_n x_1 + f_{n+1} x_2 = f_{n+2}, \qquad f_{n+1} x_1 + f_{n+2} x_2 = f_{n+3}$$

postaje to lošije uslovljen što je n veće.
c) Odrediti rešenje sistema

$$f_n x_1 + f_{n+1} x_2 = f_{n+2}, \qquad f_{n+1} x_1 + (f_{n+2} + \varepsilon) x_2 = f_{n+3}.$$

Zatim, za $n = 10$ odrediti vrednost ε za koju rešenje ne postoji.

Rešenje: a) Dokaz se izvodi matematičkom indukcijom. Za $n = 0$ je $f_0 f_2 - f_1^2 = 0 \cdot f_2 - 1 = -1$, što je tačno. Pretpostavimo da važi za $n = k$ i dokažimo da važi za $n = k + 1$,

$$\begin{aligned} f_{k+1} f_{k+3} - f_{k+2}^2 &= f_{k+1}(f_{k+1} + f_{k+2}) - f_{k+2}^2 = f_{k+1}^2 - f_{k+2}(f_{k+2} - f_{k+1}) \\ &= f_{k+1}^2 - f_{k+2} f_k = -\big(f_k f_{k+2} - f_{k+1}^2\big) = (-1)^{k+2}. \end{aligned}$$

b) S obzirom na dokazano u tački (a) matrica sistema A_n i njena inverzna matrica su

$$A_n = \begin{pmatrix} f_n & f_{n+1} \\ f_{n+1} & f_{n+2} \end{pmatrix} \qquad A_n^{-1} = (-1)^{n+1} \begin{pmatrix} f_{n+2} & -f_{n+1} \\ -f_{n+1} & f_n \end{pmatrix}.$$

Uniformne norme ovih matrica su

$$\|A_n\|_\infty = \|A_n^{-1}\|_\infty = f_{n+1} + f_{n+2} = f_{n+3},$$

te je

$$\lim_{n\to\infty} \text{cond}(A_n) = \lim_{n\to\infty} \|A_n\|_\infty \|A_n^{-1}\|_\infty = \lim_{n\to\infty} f_{n+3}^2 = \infty.$$

c) Rešenja datog sistema su

$$x_1 = 1 + \frac{\varepsilon f_{n+1}}{(-1)^{n+1} + \varepsilon f_n}, \qquad x_2 = 1 - \frac{\varepsilon f_n}{(-1)^{n+1} + \varepsilon f_n}, \qquad \text{za } \varepsilon \neq \frac{(-1)^n}{f_n}.$$

Na osnovu diferencne jednačine kojom je definisan Fibonacci-jev niz nalazimo da je

$$f_n = \frac{1}{2^n\sqrt{5}}\left((1+\sqrt{5})^n - (1-\sqrt{5})^n\right),$$

pa je $f_{10} = 55$. Dakle, sistem nema rešenje ako je $\varepsilon = 1/55 = 0.02$. Ovo je upravo posledica loše uslovljenosti matrice sistema - mala promena jednog koeficijenta sistema dovodi do toga da je matrica sistema singularna.

5.12 *Ispitati uslovljenost matrice A_n reda n čiji su elementi*

$$a_{ij} = \begin{cases} 1, & i = j \neq n \\ (-1)^{i+j-1}, & i > j \text{ ili } j = n, \\ 0, & i < j \text{ i } j \neq n \end{cases}$$

Rešenje: Matrica A_n je oblika

$$A_n = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & (-1)^n \\ 1 & 1 & 0 & \dots & 0 & (-1)^{n-1} \\ -1 & 1 & 1 & \dots & 0 & (-1)^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ (-1)^n & (-1)^{n-1} & (-1)^{n-2} & \dots & 1 & -1 \end{pmatrix}$$

pa je $\|A_n\|_\infty = n$. Da bi izračunali determinantu ove matrice, oduzmimo od i-te vrste $(i = 2, \dots, n)$ prvu pomnoženu sa $a_{i1} = (-1)^i$ i razvijmo je po prvoj koloni

$$\det A_n = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 & (-1)^n \\ 0 & 1 & 0 & \dots & 0 & 2(-1)^{n-1} \\ 0 & 1 & 1 & \dots & 0 & 2(-1)^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & (-1)^{n-1} & (-1)^{n-2} & \dots & 1 & -2 \end{vmatrix}$$

$$= 2\begin{vmatrix} 1 & 0 & \dots & 0 & (-1)^{n-1} \\ 1 & 1 & \dots & 0 & (-1)^{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (-1)^{n-1} & (-1)^{n-2} & \dots & 1 & -1 \end{vmatrix} = 2\det A_{n-1}$$

Primenjujući isti algoritam na $\det A_{n-1}, \dots$, dobijamo da je

$$\det A_n = 2\det A_{n-1} = 2^2 \det A_{n-2} = \dots = 2^{n-2}\det A_2 = -2^{n-1}.$$

Koristeći opisani algoritam i izračunatu vrednost za $\det A_n$ nalazimo da je inverzna matrica A_n^{-1}

$$\begin{pmatrix} 2^{-1} & 2^{-2} & -2^{-3} & \dots & (-1)^{n-1}2^{-(n-1)} & (-1)^{n}2^{-(n-1)} \\ 0 & 2^{-1} & 2^{-2} & \dots & (-1)^{n-2}2^{-(n-2)} & (-1)^{n-1}2^{-(n-2)} \\ 0 & 0 & 2^{-1} & \dots & (-1)^{n-3}2^{-(n-3)} & (-1)^{n-2}2^{-(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2^{-1} & 2^{-1} \\ (-1)^{n}2^{-1} & (-1)^{n-1}2^{-2} & (-1)^{n-2}2^{-3} & \dots & 2^{-(n-1)} & 2^{-(n-1)} \end{pmatrix}$$

te je stoga

$$\|A^{-1}\|_\infty = \sum_{k=1}^{n-1} \frac{1}{2^k} + \frac{1}{2^{n-1}} = 1.$$

Uslovljenost matrice A_n je jednaka njenoj dimenziji,

$$\mathrm{cond}(A_n) = \|A_n\|_\infty \|A_n^{-1}\|_\infty = n.$$

Matrica je to lošije uslovljena što joj je dimenzija veća.

5.4 Iterativne metode

Jacobi-jeva metoda iteracije se definiše iterativnim algoritmom

$$\mathbf{x}^{(n+1)} = B\mathbf{x}^{(n)} + \mathbf{c}, \qquad n = 0, 1, \dots,$$

i izborom početne aproksimacije $\mathbf{x}^{(0)}$. Matrica B i vektor $\mathbf{c}$ su definisani transformacijom polaznog sistema u ekvivalentan sistem

$$A\mathbf{x} = \mathbf{b} \qquad \Longleftrightarrow \qquad \mathbf{x} = B\mathbf{x} + \mathbf{c}$$

Dovoljan uslov konvergencije iterativnog algoritma je $q = \|B\| < 1$ za bilo koju normu matrice. Apriorna i aposteriorna ocena greške su

$$\|\mathbf{x}^* - \mathbf{x}^{(n)}\| \le \frac{q^n}{1-q}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|, \qquad \|\mathbf{x}^* - \mathbf{x}^{(n)}\| \le \frac{q}{1-q}\|\mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}\|,$$

gde je norma vektora saglasna normi matrice kojom je određen parametar q.

Metoda Gauss–Seidel-a je varijanta metode iteracije u kojoj se aproksimacija koordinate vektora rešenja izračunata na novom nivou iteracije odmah koristi za računanje aproksimacije sledeće koordinate.

Metoda najstrmijeg spuštanja je iterativna metoda definisana formulama

$$\begin{aligned} \mathbf{x}^{(n+1)} &= \mathbf{x}^{(n)} - \tau_n \mathbf{r}^{(n)}, \\ \mathbf{r}^{(n)} = A\mathbf{x}^{(n)} - \mathbf{b}, \qquad \tau_n &= \frac{(\mathbf{r}^{(n)}, \mathbf{r}^{(n)})}{(A\mathbf{r}^{(n)}, \mathbf{r}^{(n)})}, \end{aligned} \qquad n = 0, 1, \dots,$$

5.13 *Sa tačnošću* 10^{-3} *metodom iteracije rešiti sistem*

$$A\mathbf{x} = \mathbf{b}, \qquad \textit{gde je} \quad A = \begin{pmatrix} 1.02 & -0.25 & -0.30 \\ -0.41 & 1.13 & -0.15 \\ -0.25 & -0.14 & 1.21 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 0.515 \\ 1.555 \\ 2.780 \end{pmatrix}$$

Rešenje: Matrica A je dijagonalno dominantna $\left(|a_{ii}| > \sum_{i \neq j} |a_{ij}|\right)$. Ako se predstavi u obliku $A = I - C$, iterativni algoritam se može definisati formulom

$$\mathbf{x}^{(n+1)} = C\mathbf{x}^{(n)} + \mathbf{b} \qquad \text{tj.} \qquad \begin{aligned} x_1^{(n+1)} &= 0.515 - 0.02x_1^{(n)} + 0.25x_2^{(n)} + 0.30x_3^{(n)} \\ x_2^{(n+1)} &= 1.555 + 0.41x_1^{(n)} - 0.13x_2^{(n)} + 0.15x_3^{(n)} \\ x_3^{(n+1)} &= 2.780 + 0.25x_1^{(n)} + 0.14x_2^{(n)} - 0.21x_3^{(n)} \end{aligned}$$

Koeficijent kontrakcije je $q = \|C\|_\infty = 0.69$, te je tačnost postignuta kada je

$$\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\|_\infty = \max_{1 \leq i \leq 3} |x_i^{(n+1)} - x_i^{(n)}| < \frac{1-q}{q} 10^{-3} = 0.45 \cdot 10^{-3}.$$

Polazeći od vektora $\mathbf{b}$ kao početne aproksimacije, u jedanaestoj iteraciji se dobija rešenje sa traženom tačnošću

$$x_1 = 2.000, \qquad x_2 = 2.500, \qquad x_3 = 3.000.$$

5.14 *Sa tačnošću* $5 \cdot 10^{-4}$ *metodom iteracije rešiti sistem*

$$\begin{aligned} 2x_1 + 3x_2 + 11x_3 + 5x_4 &= 2 \\ x_1 + x_2 + 5x_3 + 2x_4 &= 1 \\ 2x_1 + x_2 + 3x_3 + 2x_4 &= -3 \\ x_1 + x_2 + 3x_3 + 4x_4 &= -3 \end{aligned}$$

Rešenje: Ako jednačine polaznog sistema redom označimo sa $(i), (ii), (iii)$ i (iv), onda sistem ekvivalentan polaznom, koji definiše konvergentan iterativni postupak, dobijamo na sledeći način:

$$\begin{aligned} -3x_1 - x_4 &= 7 && (i) - (ii) - 2(iii) \\ -x_1 + 3x_2 + x_3 &= 3 && 4(i) - 8(ii) - (iv) \\ 2x_2 + 8x_3 + 3x_4 &= 5 && (i) - (iii) \\ -x_1 + 2x_4 &= 0 && (iv) - (iii) \end{aligned}$$

Jacobi-jev iterativni postupak definisan je formulama

$$x_1^{(n+1)} = -\frac{1}{3}(7 + x_4^{(n)})$$
$$x_2^{(n+1)} = \frac{1}{3}(3 + x_1^{(n)} - x_3^{(n)})$$
$$x_3^{(n+1)} = \frac{1}{8}(5 - 2x_2^{(n)} - 3x_4^{(n)})$$
$$x_4^{(n+1)} = \frac{1}{2}x_1^{(n)}$$

i konvergira, jer je uniformna norma matrice sistema $q = 0.67$. Polazeći od aproksimacije rešenja definisane slobodnim članom $\mathbf{x}^{(0)} = (-2.3333,\ 1,\ 0.625,\ 0)^\top$ tačnost

$$\max_{1\le i\le 4} |x_i^{(n+1)} - x_i^{(n)}| < \frac{1-q}{q} 5 \cdot 10^{-4} = 0.25 \cdot 10^{-3}$$

je postignuta u jedanaestoj iteraciji, i dobijeno je rešenje

$$x_1 = -2.000, \qquad x_2 = 0.000, \qquad x_3 = 1.000, \qquad x_4 = -1.000.$$

5.15 *Dat je sistem*

$$A\mathbf{x} = \mathbf{b} \qquad \textit{gde je} \quad A = \begin{pmatrix} 1 & 0 & -0.25 & -0.25 \\ 0 & 1 & -0.25 & -0.25 \\ -0.25 & -0.25 & 1 & 0 \\ -0.25 & -0.25 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

a) Dokazati da za dati sistem Gauss–Seidel-ov iterativni algoritam konvergira.

b) Izračunati četiri aproksimacije rešenja sistema Gauss–Seidel-ovim algoritmom polazeći od aproksimacije rešenja $\mathbf{x}^{(0)} = (0, 0, 0, 0)^\top$.

c) Oceniti grešku približnog rešenja određenog pod (b).

Rešenje: a) Ako matricu sistema predstavimo u obliku $A = L + D + U$, gde je L donje trougaona matrica određena elementima matrice A ispod glavne dijagonale, D dijagonalna matrica određena elementima matrice A na glavnoj dijagonali i U gornje trougaona matrica određena elementima matrice A iznad glavne dijagonale, Gauss–Seidel-ova metoda je određena algoritmom

$$(L+D)\mathbf{x}^{(n+1)} = -U\mathbf{x}^{(n)} + \mathbf{b}. \qquad \text{tj.} \qquad \mathbf{x}^{(n+1)} = -(L+D)^{-1}U\mathbf{x}^{(n)} + (L+D)^{-1}\mathbf{b}$$

Uniformna norma matrice

$$C = -(L+D)^{-1}U = \begin{pmatrix} 0 & 0 & 0.25 & 0.25 \\ 0 & 0 & 0.25 & 0.25 \\ 0 & 0 & 0.125 & 0.125 \\ 0 & 0 & 0.125 & 0.125 \end{pmatrix}$$

je $q = \|C\|_\infty = 0.5$, te Gauss–Seidel-ov algoritam konvergira.
b) Polazeći od aproksimacije $\mathbf{x}^{(0)}$ i računajući po formulama

$$x_1^{(n+1)} = 0.25x_3^{(n)} + 0.25x_4^{(n)} + 0.5$$
$$x_2^{(n+1)} = 0.25x_3^{(n)} + 0.25x_4^{(n)} + 0.5$$
$$x_3^{(n+1)} = 0.25x_1^{(n+1)} + 0.25x_2^{(n+1)} + 0.5$$
$$x_4^{(n+1)} = 0.25x_1^{(n+1)} + 0.25x_2^{(n+1)} + 0.5$$

dobijamo aproksimacije rešenja

n	0	1	2	3	4
x_1	0	0.5	0.875	0.9688	0.9922
x_2	0	0.5	0.875	0.9688	0.9922
x_3	0	0.75	0.9375	0.9844	0.9961
x_4	0	0.75	0.9375	0.9844	0.9961

c) $\mathbf{x}^{(4)}$ je približno rešenje sa tačnošću

$$\|\mathbf{x}^* - \mathbf{x}^{(4)}\|_\infty = \max_{1\le i\le 4} |x_i^* - x_i^{(4)}| \le \frac{q}{1-q} \max_{1\le i\le 4} |x_i^{(4)} - x_i^{(3)}| = 0.024.$$

Zapisano samo sigurnim ciframa rešenje sistema je

$$x_1 = 1.0, \qquad x_2 = 1.0, \qquad x_3 = 1.0, \qquad x_4 = 1.0.$$

5.16 *Sa tačnošću* 10^{-3} *Gauss–Seidel-ovom metodom rešiti sistem*

$$2.7x_1 + 3.3x_2 + 1.3x_3 = 2.1$$
$$3.5x_1 - 1.7x_2 + 2.8x_3 = 1.7$$
$$4.1x_1 + 5.8x_2 - 1.7x_3 = 0.8$$

Rešenje: Ako jednačine polaznog sistema redom označimo sa (i), (ii) i (iii), onda je sistem ekvivalentan polaznom, koji definiše konvergentan iterativni postupak

$$7.6x_1 + 4.1x_2 + 1.1x_3 = 2.5 \qquad (ii) + (iii)$$
$$3.3x_1 + 10.8x_2 - 3.2x_3 = 1.2 \qquad (i) - (ii) + (iii)$$
$$1.3x_1 + 0.8x_2 + 4.3x_3 = 3.4 \qquad 2(i) - (iii)$$

Gauss–Seidel-ov iterativni postupak definisan je formulama

$$x_1^{(n+1)} = \frac{5}{38}(2.5 - 4.1x_2^{(n)} - 1.1x_3^{(n)})$$
$$x_2^{(n+1)} = \frac{5}{54}(1.2 - 3.3x_1^{(n+1)} + 3.2x_3^{(n)})$$
$$x_3^{(n+1)} = \frac{10}{43}(3.4 - 1.3x_1^{(n+1)} - 0.8x_2^{(n+1)})$$

U osmoj iteraciji, polazeći od aproksimacije rešenja definisane slobodnim članom

$$\mathbf{x}^{(0)} = (0.3289,\, 0.1111,\, 0.7907)^\top$$

dobijeno je rešenje

$$x_1 = 0.061, \qquad x_2 = 0.304, \qquad x_3 = 0.716.$$

5.17 *Metodom najstrmijeg spuštanja rešiti sistem linearnih jednačina*

$$A\mathbf{x} = \mathbf{b}, \qquad A = \begin{pmatrix} 4.00 & 0.32 & 0.15 \\ 0.32 & 4.00 & 0.64 \\ 0.15 & 0.64 & 4.00 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4.190 \\ 2.448 \\ 1.270 \end{pmatrix}.$$

Za početnu aproksimaciju uzeti $\mathbf{x} = (1,\, 0.4,\, 0.1)^\top$. *Računati sa pet značajnih cifara.*

Rešenje: U petoj iteraciji dobija se traženo rešenje (ujedno i tačno rešenje)

$$\mathbf{x}^* = (1,\, 0.5,\, 0.2)^\top.$$

5.18 *Metodom najstrmijeg spuštanja rešiti sistem*

$$\begin{aligned} 8.467x_1 + 5.137x_2 + 3.141x_3 + 2.063x_4 &= 29.912 \\ 5.137x_1 + 6.421x_2 + 2.617x_3 + 2.003x_4 &= 25.058 \\ 3.141x_1 + 2.617x_2 + 4.128x_3 + 1.628x_4 &= 16.557 \\ 2.063x_1 + 2.003x_2 + 1.628x_3 + 3.446x_4 &= 12.690 \end{aligned}$$

Rešenje: Polazeći od početne aproksimacije $\mathbf{x}^{(0)} = (2,\, 1,\, 1,\, 2)^\top$ u trinaestoj iteraciji dobijamo sa tačnošću 10^{-3} rešenje

$$\mathbf{x}^* = (1.876,\, 1.595,\, 1.140,\, 1.095)^\top$$

5.19 *Dokazati da metoda iteracije za sistem linearnih jednačina*

$$A\mathbf{x} = \mathbf{b}, \qquad A = \begin{pmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{pmatrix}$$

konvergira za $-1/2 < a < 1/2$.

Rešenje: Metoda iteracije definisana je iterativnim algoritmom

$$\mathbf{x}^{(n+1)} = C\mathbf{x}^{(n)} + \mathbf{b}, \qquad C = I - A,$$

gde je I jedinična matrica. Karakteristični polinom matrice C je

$$\det(C-\lambda I)=\begin{vmatrix}-\lambda & -a & -a\\ -a & -\lambda & -a\\ -a & -a & -\lambda\end{vmatrix}=-(\lambda-a)^2(\lambda+2a)$$

pa su $\lambda_{1/2}=a$, $\lambda_3=-2a$ sopstvene vrednosti matrice C. Iterativni algoritam konvergira ako je $\max|\lambda_i|<1$ odakle sledi tvrđenje.

5.20 *Dat je sistem linearnih jednačina*

$$A\mathbf{x}=\mathbf{b}, \qquad A=I-C, \quad c_{ij}\geq 0, \quad b_i\geq 0,$$

gde je I jedinična matrica, $C=\{c_{ij}\}$ i $\mathbf{b}=(b_1,\dots,b_m)^\top$. Pokazati da, ako je rešenje sistema $\mathbf{x}^*$ nenegativno, onda iterativni algoritam

$$(*) \qquad \mathbf{x}^{(0)}=\mathbf{0}, \qquad \mathbf{x}^{(n+1)}=C\mathbf{x}^{(n)}+\mathbf{b}, \quad n=0,1,\dots,$$

konvergira ka rešenju sistema.

Rešenje: Ako uvedemo oznake $\mathbf{x}^*=(x_1^*,\dots,x_m^*)^\top$ i $\mathbf{x}^{(n)}=(x_1^{(n)},\dots,x_m^{(n)})^\top$, matematičkom indukcijom ćemo dokazati da je

$$0\leq x_i^{(0)}\leq x_i^{(1)}\leq\cdots\leq x_i^{(n)}\leq\cdots\leq x_i^*, \qquad i=1,\dots,m.$$

Kako je $\mathbf{x}^{(0)}=\mathbf{0}$ to je $\mathbf{x}^{(1)}=\mathbf{b}$ pa je prema pretpostavci zadatka za $n=1$ tvrđenje tačno,

$$0=x_i^{(0)}\leq x_i^{(1)}=b_i, \qquad i=1,\dots,m.$$

Pretpostavimo da tvrđenje važi za proizvoljno n

$$x_i^{(n-2)}\leq x_i^{(n-1)}, \qquad i=1,\dots,m.$$

Tada je na osnovu iterativnog algoritma $(*)$ i pretpostavke o matrici C

$$x_i^{(n-1)}=\sum_{k=1}^m c_{ik}x_k^{(n-2)}+b_i\leq\sum_{k=1}^m c_{ik}x_k^{(n-1)}+b_i=x_i^{(n)}, \qquad i=1,\dots,m.$$

Još treba dokazati da je $x_i^{(n)}\leq x_i^*$ za svako n i $i=1,\dots,m$. Rešenje sistema $\mathbf{x}^*$ predstavlja nepokretnu tačku preslikavanja koje definiše iterativni algoritam $(*)$, jer iz

$$A\mathbf{x}^*=\mathbf{b}, \quad \text{tj.} \quad (I-C)\mathbf{x}^*=\mathbf{b} \qquad \text{sledi} \qquad \mathbf{x}^*=C\mathbf{x}^*+\mathbf{b}.$$

Kako je $c_{ij}\geq 0$ i $x_i^*\geq 0$ sledi da je $x_i^*\geq b_i=x_i^{(1)}$, $i=1,\dots,m$. Dakle, za $n=1$ tvrđenje je dokazano. Dalje, pretpostavimo da je $x_i^{(n-1)}\leq x_i^*$ za proizvoljno n. Tada je

$$x_i^{(n)}=\sum_{k=1}^m c_{ik}x_k^{(n-1)}+b_i\leq\sum_{k=1}^m c_{ik}x_k^*+b_i=x_i^*, \qquad i=1,\dots,m.$$

Ovim je dokazano da su nizovi $\{x_i^{(n)}\}_0^\infty$, $i = 1, \ldots, m$, monotono neopadajući i ograničeni sa gornje strane, pa stoga konvergiraju

$$\lim_{n\to\infty} x_i^{(n)} = \tilde{x}_i, \qquad i = 1, \ldots, m.$$

Kada u formuli $(*)$ pustimo da $n \to \infty$ zaključujemo da vektor $\tilde{\mathbf{x}} = (\tilde{x}_1, \ldots, \tilde{x}_m)^\top$ zadovoljava sistem $A\tilde{\mathbf{x}} = \mathbf{b}$. Kako je matrica sistema regularna, rešenje sistema je jedinstveno pa je $\tilde{\mathbf{x}} = \mathbf{x}^*$, tj. iterativni algoritam $(*)$ konvergira ka rešenju sistema $A\mathbf{x} = \mathbf{b}$.

5.21 *a) Pokazati da za sistem linearnih jednačina* $A\mathbf{x} = \mathbf{b}$, *gde je*

$$A = \begin{pmatrix} 2 & 0.3 & 0.5 \\ 0.1 & 3 & 0.4 \\ 0.1 & 0.1 & 4.8 \end{pmatrix}$$

metoda iteracije opisana formulom

$$\mathbf{x}^{(n+1)} = (I - \tau A)\mathbf{x}^{(n)} + \tau\mathbf{b}$$

(I jedinična matrica) konvergira za $0 < \tau < 0.4$.

b) Sa tačnošću $5 \cdot 10^{-4}$ *izračunati rešenje sistema navedenom formulom ako je* $\tau = 0.3$ *i* $\mathbf{b} = (4.1,\ 7.3,\ 14.7)^\top$.

Rešenje: a) Metoda iteracije konvergira ako je preslikavanje koje definiše iterativni algoritam kontrakcija. U ovom slučaju to se svodi na uslov da je $\|I - \tau A\| < 1$. Ako koristimo uniformnu matričnu normu, uslov je ispunjen ako je

$$|1 - 2\tau| + |0.3\tau| + |0.5\tau| < 1$$
$$|0.1\tau| + |1 - 3\tau| + |0.4\tau| < 1$$
$$|0.1\tau| + |0.1\tau| + |1 - 4.8\tau| < 1$$

Da bi bilo $|1 - k\tau| < 1$ mora pre svega biti $\tau > 0$ jer je $k > 0$. Tada su prethodni uslovi ispunjeni ako je

$$|1 - 2\tau| < 1 - 0.8\tau, \qquad 1 - 2\tau > -1 + 0.8\tau, \qquad \tau < 0.71$$
$$|1 - 3\tau| < 1 - 0.5\tau, \qquad 1 - 3\tau > -1 + 0.5\tau, \qquad \tau < 0.57$$
$$|1 - 4.8\tau| < 1 - 0.2\tau, \qquad 1 - 4.8\tau > -1 + 0.2\tau, \qquad \tau < 0.4$$

Dakle, vrednost parametra τ koja zadovoljava sva tri uslova, tj. koja garantuje konvergenciju iterativnog algoritma je $0 < \tau < 0.4$.

b) Za $\tau = 0.3$ koeficijent kontrakcije je $q = 0.64$. Iterativni postupak definisan je formulama

$$x_1^{(n+1)} = 0.4x_1^{(n)} - 0.09x_2^{(n)} - 0.15x_3^{(n)} + 1.23$$
$$x_2^{(n+1)} = -0.03x_1^{(n)} + 0.1x_2^{(n)} - 0.12x_3^{(n)} + 2.19$$
$$x_3^{(n+1)} = -0.03x_1^{(n)} - 0.03x_2^{(n)} - 0.44x_3^{(n)} + 4.41$$

Polazeći od aproksimacije rešenja definisane slobodnim članom

$$\mathbf{x}^{(0)} = (1.23,\, 2.19,\, 4.41)^\top$$

tačnost

$$\max_{1\le i\le 4} |x_i^{n+1} - x_i^n| < \frac{1-q}{q} 5 \cdot 10^{-4} = 2.8 \cdot 10^{-4}$$

je postignuta u dvanaestoj iteraciji, i dobijeno je rešenje

$$x_1 = 1.000, \qquad x_2 = 2.000, \qquad x_3 = 3.000.$$

5.22 *Dokazati da Jacobi-jeva i Gauss–Seidel-ova metoda iteracije za sistem linearnih jednačina*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned} \qquad a_{11}a_{22} \neq 0.$$

pod istim uslovom konvergiraju.

Rešenje: Jacobi-jeva metoda je definisana iterativnim algoritmom

$$x_1^{(n+1)} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}} x_2^{(n)}$$

$$x_2^{(n+1)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1^{(n)}$$

tj.

$$\mathbf{x}^{(n+1)} = C_j \mathbf{x}^{(n)} + \mathbf{d}_j$$

a Gauss–Seidel-ova iterativnim algoritmom

$$x_1^{(n+1)} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}} x_2^{(n)}$$

$$x_2^{(n+1)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1^{(n+1)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} \frac{b_1}{a_{11}} + \frac{a_{21}a_{12}}{a_{22}a_{11}} x_2^{(n)}$$

tj.

$$\mathbf{x}^{(n+1)} = C_s \mathbf{x}^{(n)} + \mathbf{d}_s.$$

Sopstvene vrednosti matrice C_j su $\lambda_{1/2} = \pm\sqrt{a_{21}a_{12}/a_{22}a_{11}}$, a sopstvene vrednosti matrice C_s su $\lambda_1 = 0$, $\lambda_2 = a_{21}a_{12}/a_{22}a_{11}$. Kako je potreban i dovoljan uslov da iterativni algoritam $\mathbf{x}^{(n+1)} = C\mathbf{x}^{(n)} + \mathbf{d}$ konvergira je da sopstvene vrednosti matrice C zadovoljavaju uslov $\max|\lambda_i| < 1$, u oba slučaja će taj uslov biti ispunjen ako je

$$\left| \frac{a_{21}\, a_{12}}{a_{22}\, a_{11}} \right| < 1.$$

5.23 *Dat je iterativni algoritam*

$$\begin{aligned} x^{(n+1)} &= -x^{(n)} + y^{(n)} \\ y^{(n+1)} &= x^{(n)} - y^{(n)} + 3 \end{aligned} \qquad n = 0, 1, \ldots .$$

Kako zavisi konvergencija od izbora početne aproksimacije?

Rešenje: Jedinstvena nepokretna tačka preslikavanja kojim je definisan iterativni algoritam je očigledno $x^* = 1$, $y^* = 2$. Sabiranjem datih jednačina dobijamo da je

$$(*) \qquad x^{(n+1)} + y^{(n+1)} = 3, \qquad n \geq 0$$

što znači da nezavisno od izbora početne aproksimacije sve aproksimacije $(x^{(n)}, y^{(n)})$ pripadaju pravoj $x + y = 3$. Ako sa $s^{(n)}$ i $t^{(n)}$ označimo odstupanja n-te iteracije od tačnog rešenja,

$$s^{(n)} = x^{(n)} - x^*, \qquad t^{(n)} = y^{(n)} - y^*,$$

onda je, na osnovu $(*)$,

$$s^{(n)} + x^* + t^{(n)} + y^* = 3, \qquad \text{tj.} \qquad t^{(n)} = -s^{(n)}, \qquad n \geq 1$$

Stoga je

$$\begin{aligned} s^{(n)} &= -x^{(n-1)} + y^{(n-1)} - x^* = -s^{(n-1)} + t^{(n-1)} - 2x^* + y^* = -2s^{(n-1)} \\ &= \cdots = (-2)^{n-1} s^{(1)} \underset{n\to\infty}{\longrightarrow} \infty, \end{aligned}$$

$$\begin{aligned} t^{(n)} &= x^{(n-1)} - y^{(n-1)} + 3 - y^* = s^{(n-1)} - t^{(n-1)} + 3 + x^* - 2y^* = 2s^{(n-1)} \\ &= \cdots = 2^{n-1} s^{(1)} \underset{n\to\infty}{\longrightarrow} \infty, \end{aligned}$$

izuzev kada je $s^{(1)} = t^{(1)} = 0$. Stoga iterativni algoritam konvergira ako i samo ako je

$$y^{(0)} = 1 + x^{(0)}.$$

5.24 *Dat je iterativni algoritam*

$$\begin{aligned} x^{(n+1)} &= -x^{(n)} + y^{(n)} \\ y^{(n+1)} &= x^{(n)} - 2y^{(n)} + 3 \end{aligned} \qquad n = 0, 1, \ldots .$$

Kako zavisi konvergencija od izbora početne aproksimacije?

Rešenje: Jedinstvena nepokretna tačka preslikavanja kojim je definisan iterativni algoritam je očigledno $x^* = 0.6$, $y^* = 1.2$. Ako sa $s^{(n)}$ i $t^{(n)}$ označimo odstupanja n-te iteracije od tačnog rešenja,

$$s^{(n)} = x^{(n)} - x^*, \qquad t^{(n)} = y^{(n)} - y^*,$$

onda je

$$\begin{aligned} s^{(n+1)} &= x^{(n+1)} - x^* = -x^{(n)} + y^{(n)} - x^* = -s^{(n)} + t^{(n)} - 2x^* + y^* \\ &= -s^{(n)} + t^{(n)} \end{aligned}$$

$$\begin{aligned} t^{(n+1)} &= y^{(n+1)} - y^* = x^{(n)} - 2y^{(n)} + 3 - y^* = s^{(n)} - 2t^{(n)} + 3 + x^* - 3y^* \\ &= s^{(n)} - 2t^{(n)} \end{aligned}$$

Kvadratna forma

$$\Phi_n = -\left(s^{(n)}\right)^2 + s^{(n)}t^{(n)} + \left(t^{(n)}\right)^2$$

ne zavisi od n, jer je

$$\begin{aligned} \Phi_n = -\left(-s^{(n-1)} + t^{(n-1)}\right)^2 &+ \left(-s^{(n-1)} + t^{(n-1)}\right)\left(s^{(n-1)} - 2t^{(n-1)}\right) \\ &+ \left(s^{(n-1)} - 2t^{(n-1)}\right)^2 \\ = -\left(s^{(n-1)}\right)^2 + s^{(n-1)}t^{(n-1)} + \left(t^{(n-1)}\right)^2 &= \Phi_{n-1} = \cdots = \Phi_0. \end{aligned}$$

Ako je $\Phi_0 \neq 0$, tačke $(s^{(n)}, t^{(n)})$, $n = 0, 1, \ldots$, pripadaju hiperboli $-s^2+st+t^2 = \Phi_0$ koja ne sadrži tačku $(0, 0)$, te iterativni algoritam ne konvergira. Ako je $\Phi_0 = 0$ ove tačke pripadaju jednoj od pravih

$$t = s(\sqrt{5} - 1)/2, \qquad t = -s(\sqrt{5} + 1)/2.$$

Ako pripadaju prvoj pravoj iterativni proces konvergira, jer je

$$\begin{aligned} s^{(n)} &= -s^{(n-1)} + s^{(n-1)}(\sqrt{5} - 1)/2 = s^{(n-1)}(\sqrt{5} - 3)/2 = \ldots \\ &= s^{(0)}\left((\sqrt{5} - 3)/2\right)^n \underset{n\to\infty}{\longrightarrow} 0, \end{aligned}$$

$$\begin{aligned} t^{(n)} &= s^{(n-1)} - s^{(n-1)}(\sqrt{5} - 1) = s^{(n-1)}(2 - \sqrt{5}) = \ldots \\ &= s^{(0)}(2 - \sqrt{5})^n \underset{n\to\infty}{\longrightarrow} 0, \end{aligned}$$

Ako pripadaju drugoj pravoj iterativni proces ne konvergira jer je

$$\begin{aligned} s^{(n)} &= -s^{(n-1)} - s^{(n-1)}(\sqrt{5} + 1)/2 = -s^{(n-1)}(\sqrt{5} + 3)/2 = \ldots \\ &= -s^{(0)}\left((\sqrt{5} + 3)/2\right)^n \underset{n\to\infty}{\longrightarrow} \infty, \end{aligned}$$

$$\begin{aligned} t^{(n)} &= s^{(n-1)} + s^{(n-1)}(\sqrt{5} + 1) = s^{(n-1)}(\sqrt{5} + 2) = \ldots \\ &= s^{(0)}(\sqrt{5} + 2)^n \underset{n\to\infty}{\longrightarrow} \infty, \end{aligned}$$

osim ako je $s^{(0)} = t^{(0)} = 0$, tj. ako je za početnu aproksimaciju izabrana baš nepokretna tačka.

Dakle, dati iterativni algoritam će konvergirati samo ako je početna aproksimacija izabrana tako da je

$$2(y^{(0)} - y^*) = (x^{(0)} - x^*)(\sqrt{5} - 1).$$

6

Sopstvene vrednosti i sopstveni vektori matrica

Sopstvene vrednosti kvadratne matrice A dimenzije m su vrednosti skalara λ za koje homogeni sistem $A\mathbf{x} = \lambda\mathbf{x}$ ima netrivijalna rešenja, a odgovarajuća rešenja su sopstveni vektori. Sopstvene vrednosti su nule karakterističnog polinoma

$$D(\lambda) = (-1)^m(\lambda^m + p_1\lambda^{m-1} + \cdots + p_m).$$

6.1 *Date su matrice dimenzije* $m \times m$

$$A = \begin{pmatrix} a & 1 & 0 & \dots & 0 \\ 0 & a & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & a \end{pmatrix} \qquad B = \begin{pmatrix} a & 1 & 0 & \dots & 0 \\ 0 & a & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \varepsilon & 0 & 0 & \dots & a \end{pmatrix}$$

a) Pokazati da sopstvene vrednosti λ_k *matrice* B *zadovoljavaju relacije*

$$|\lambda_k - a| = |\varepsilon|^{1/m}, \qquad k = 1, \dots, m.$$

b) Za $\varepsilon = 10^{-m}$ *analizirati zavisnost procentualne greške sopstvenih vrednosti od* m *i* a.

Rešenje: a) Karakteristični polinom matrice B je

$$\det(B-\lambda I)=\begin{vmatrix} a-\lambda & 1 & \dots & 0 \\ 0 & a-\lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon & 0 & \dots & a-\lambda \end{vmatrix}$$

$$=(a-\lambda)\begin{vmatrix} a-\lambda & 1 & \dots & 0 \\ 0 & a-\lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a-\lambda \end{vmatrix}+(-1)^{m-1}\varepsilon\begin{vmatrix} 1 & 0 & \dots & 0 \\ a-\lambda & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{vmatrix}$$

$$=(a-\lambda)^m-(-1)^m\varepsilon.$$

Kako su sopstvene vrednosti nule karakterističnog polinoma, tvrđenje neposredno sledi.

b) Stavljajući da je $\varepsilon=10^{-m}=0.1^m$, na osnovu prethodnog imamo da je

$$|\lambda_k-a|=0.1,$$

tj. apsolutna greška sopstvene vrednosti je $|\varepsilon|^{1/m}=0.1$. Procentualna greška je $100|\varepsilon|^{1/m}/|a|$, i za datu vrednost za ε je $10/|a|$.

6.1 Givens-ova metoda rotacije

Kompleksna matrica rotacije U_{kl} je unitarna matrica ($U_{kl}^{-1}=U_{kl}^*$) oblika

$$U_{kl}=\begin{pmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & \alpha & & -\overline{\beta} & & \\ & & & \ddots & & & \\ & & \beta & & \alpha & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{pmatrix},\quad \alpha=\cos(\phi),\ \beta=e^{\imath\psi}\sin(\phi)$$

Givens-ovom metodom se, pomoću Givens-ove matrice rotacije U_{kl}, matrica $A=(a_{ij})$ transformiše u matricu $B=(b_{ij})=U_{kl}^*AU_{kl}$ čiji je element $b_{l,k-1}=0$. Kako bi se ovo postiglo, parametri matrice U_{kl} su određeni izrazima

$$\alpha = \frac{|a_{k,k-1}|}{\sqrt{|a_{k,k-1}|^2+|a_{l,k-1}|^2}},\quad \beta = \frac{a_{l,k-1}}{a_{k,k-1}}\,\alpha,\quad \text{za}\quad a_{k,k-1}\neq 0$$

$$\alpha = 0,\qquad \beta = 1,\qquad \text{za}\quad a_{k,k-1}=0$$

Izračunavanje matrice B se može realizovati u dva koraka. U prvom koraku se računa matrica $C = (c_{i,j}) = A\,U_{k,l}$, koja se razlikuje od matrice A samo u k-toj i l-toj koloni,

$$c_{i,k} = a_{i,k}\alpha + a_{i,l}\beta, \qquad c_{i,l} = -a_{i,k}\overline{\beta} + a_{i,l}\alpha.$$

U drugom koraku se računa matrica $B = U_{k,l}^* C$, koja se razlikuje od matrice C samo u k-toj i l-toj vrsti,

$$b_{k,j} = c_{k,j}\alpha + c_{l,j}\overline{\beta}, \qquad b_{l,j} = -c_{k,j}\overline{\beta} + c_{l,j}\alpha.$$

6.2 *Givens-ovom metodom rotacije svesti matricu*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

na trodijagonalnu.

Rešenje: Pošto je matrica A simetrična, ona će Givens-ovom metodom rotacije biti svedena na trodijagonalnu. Stoga se svodi se na nulu samo element $a_{3,1}$. Parametri rotacije su

$$\alpha = \frac{|a_{2,1}|}{\sqrt{|a_{2,1}|^2 + |a_{3,1}|^2}} = \frac{1}{\sqrt{2}} \qquad \beta = \frac{a_{3,1}}{a_{2,1}}\,\alpha = \frac{1}{\sqrt{2}}$$

pa je

$$U_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \qquad B = U_{23}^* A U_{23} = \begin{pmatrix} 2 & \sqrt{2} & 0 \\ \sqrt{2} & 7/2 & 1/2 \\ 0 & 1/2 & 3/2 \end{pmatrix}$$

C Pri implementaciji Givens-ove metode rotacije treba imati na umu da se prilikom množenja sa matricama rotacije menjaju samo dve kolone, odn. dve vrste polazne matrice. Stoga matrično množenje date matrice sa matricama rotacije nije optimalno rešenje, već umesto toga treba samo izračunati nove vrednosti elemenata polazne matrice u kolonama odn. vrstama koje bivaju promenjene. Radi dalje uštede, sve operacije se direktno vrše nad odgovarajućim kolonama odn. vrstama polazne matrice, tako da nije potrebno alocirati nikakav dodatni prostor za međurezultate. Na taj način, procedura koja implementira metodu ima sledeći oblik (sve metode u ovom poglavlju su radi jednostavnosti implementirane samo za slučaj matrica sa svim realnim elementima):

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
```

```
/*
 * Funkcija givens() implementira Givens-ovu metodu rotacije kojom se
 * data matrica svodi na gornju Hessenberg-ovu matricu. Argumenti
 * funkcije su:
 *   n - dimenzija matrice
 *   A - data matrica, cije ce vrednosti nakon poziva funkcije biti
 *       izmenjene
 * Pretpostavlja se da je dimenzija matrice veca od 0.
 */
void
givens(int n, double **A)
{
        double          alpha,
                        beta;   /* Elementi matrice rotacije razliciti od 0. */
        int             i, j, k, l;     /* Brojaci u petljama. */

        /* Proverava se da li je dimenzija matrice veca od 0. */
        assert(n > 0);

        /* Anuliraju se jedan po jedan elementi matrice A dok se ista ne
         * svede na gornju Hessenberg-ovu formu. */
        for (k = 1; k < n - 1; k++)
                for (l = k + 1; l < n; l++) {
                        /* Izracunavaju se elementi matrice rotacije
                         * razliciti od 0. */
                        alpha =
                            (A[k][k - 1] == 0) ?
                                        0 :
                                        fabs(A[k][k - 1]) /
                                                sqrt(A[k][k - 1]*A[k][k - 1] +
                                                        A[l][k - 1]*A[l][k - 1]);
                        beta =
                            (A[k][k - 1] == 0) ?
                                        0 :
                                        A[l][k - 1] / A[k][k - 1] * alpha;

                        /* Mnozi se matrica A matricom rotacije sa zadnje
                         * strane (pritom se odredjuju samo nove vrednosti
                         * elemenata koji bivaju izmenjeni prilikom
                         * mnozenja, a to su elementi u k-toj i l-toj
                         * koloni). */
                        for (i = 0; i < n; i++) {
                                double          a_ik,
                                                a_il;   /* Prethodne vrednosti
                                                         * elemenata A[i][k] i A[i][l]. */

                                a_ik = A[i][k], a_il = A[i][l];
                                A[i][k] = alpha * a_ik + beta * a_il;
                                A[i][l] = -beta * a_ik + alpha * a_il;
                        }

                        /* Mnozi se matrica A matricom rotacije sa prednje
                         * strane (pritom se odredjuju samo nove vrednosti
                         * elemenata koji bivaju izmenjeni prilikom
                         * mnozenja, a to su elementi u k-tom i l-tom
                         * redu). */
```

```
                        for (j = 0; j < n; j++) {
                                double          a_kj,
                                                a_lj;   /* Prethodne vrednosti
                                                           elemenata A[k][j] i A[l][j]. */

                                a_kj = A[k][j], a_lj = A[l][j];
                                A[k][j] = alpha * a_kj + beta * a_lj;
                                A[l][j] = -beta * a_kj + alpha * a_lj;
                        }
                }
}
```

MATLAB

Hessenberg-ova forma: `H = hess(A)`

MATLAB poseduje funkciju `hess` koja služi za svođenje matrice u gornje-Hessenbergovu formu. Poziv `[P, H] = hess(A)` pronalazi unitarnu matricu P i Hessenbergovu H tako da je $P^*AP = H$.

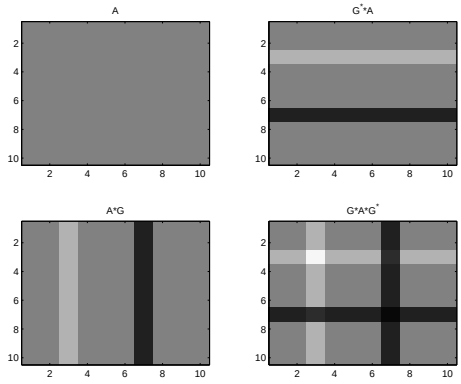
Ipak, kroz naredni zadatak ilustrujemo korišćenje matrica rotacije.

6.3 *Koristeći* MATLAB *grafički prikazati dejstvo množenja matrice matricama rotacije sa leve i desne strane.*

Za iscrtavanje matrica koristimo funkciju `imagesc` kojoj navodimo matricu kao prvi argument, dok kao drugi argument navodimo vrednosti koje će biti prikazivane najtamnijom i najsvetlijom bojom. Pošto je ovaj skript samo ilustracija nije vođeno računa o optimizaciji množenja matricama rotacije.

Kako bi se metode numeričke linearne algebre bolje razumele, savetujemo čitaocima iscrtavanje matrica međurezultata prilikom njihovog izvršavanja.

```
% Dimenzije matrica
n = 10; k = 3; l = 7;

% Gradimo proizvoljnu Givensovu matricu Gkl(phi)
phi = rand(1);
s = sin(phi); c = cos(phi);
G = eye(n);
G(k,k) = c; G(k,l) = -s;
G(l,k) = s; G(l,l) = c;

% Kako bi se efekat mnozenja bolje primecivao, za
% A uzimamo konstantnu matricu
A = ones(n);

% Iscrtavamo odgovarajuce proizvode
colormap gray, colorbar
subplot(2,2,1), imagesc(A,      [0 2]), title('A')
subplot(2,2,2), imagesc(G'*A,   [0 2]), title('G^**A')
subplot(2,2,3), imagesc(A*G,    [0 2]), title('A*G')
subplot(2,2,4), imagesc(G'*A*G, [0 2]), title('G*A*G^*')
```



Slika 6.1: Efekat množenja matricama rotacije

6.2 Jacobi-jeva metoda

Jacobi-jevom metodom se određuju sopstvene vrednosti simetrične (u opštem slučaju Hermite-ove) matrice $A = (a_{ij})$. Nizom ortogonalnih transformacija pomoću realnih matrica rotacije

$$U_{kl} = \begin{pmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & \cos\phi & & -\sin\phi & \\ & & & \ddots & & \\ & & \sin\phi & & \cos\phi & \\ & & & & & \ddots \\ 0 & & & & & 1 \end{pmatrix}$$

matrica A se transformiše u njoj sličnu dijagonalnu matricu

$$D = \operatorname{diag}(\lambda_1, \ldots, \lambda_m) = \lim_{m\to\infty} U_m^\top \ldots U_1^\top A\, U_1 \ldots U_m$$

Ugao rotacije ϕ na svakom koraku se računa po formuli

$$\tan 2\phi = \frac{-2a_{kl}}{a_{ll} - a_{kk}}$$

tako da su elementi sa indeksima kl i lk u transformisanoj matrici jednaki nuli.

Sopstveni vektori su približno vektori kolona matrice

$$U = U_1 \dots U_m.$$

Stabilan algoritam za računanje elemenata matrica U_{kl} dat je formulama

$$\sin\phi = \frac{\omega}{\sqrt{2(1+\sqrt{1-\omega^2})}}, \qquad \cos\phi = \sqrt{1-\sin^2\phi},$$

$$\omega = sign(\mu)\frac{\lambda}{\sqrt{\lambda^2+\mu^2}}, \qquad \lambda = -a_{kl}, \qquad \mu = \tfrac{1}{2}(a_{ll}-a_{kk}).$$

6.4 *Jacobi-jevom metodom odrediti sopstvene vrednosti i sopstvene vektore matrice:*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Rešenje: U prvom koraku se anulira element a_{21}, pa je:

$$\tan(2\phi) = -\frac{2a_{21}}{a_{11}-a_{22}} = -\frac{1}{2-2} \quad \Rightarrow \quad 2\phi = -\frac{\pi}{2} \quad \Rightarrow \quad \phi = -\frac{\pi}{4}$$

Matrica rotacije i transformisana matrica su

$$U_{21} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad A_1 = U_{21}^{\top} A\, U_{21} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & \sqrt{2} \\ 0 & \sqrt{2} & 3 \end{pmatrix}$$

S obzirom da je u ovom koraku anuliran i element a_{31}, u sledećem koraku se anulira element a_{32}. Ugao rotacije je

$$\tan(2\phi) = -\frac{2a_{32}}{a_{22}-a_{33}} = -\frac{2\sqrt{2}}{3-3} \quad \Rightarrow \quad 2\phi = -\frac{\pi}{2} \quad \Rightarrow \quad \phi = -\frac{\pi}{4}$$

pa su matrica rotacije i transformisana matrica

$$U_{32} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \qquad A_2 = U_{32}^{\top} A_1 U_{32} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3-\sqrt{2} & 0 \\ 0 & 0 & 3+\sqrt{2} \end{pmatrix}$$

Dakle, sopstvene vrednosti matrice A su

$$\lambda_1 = 1, \qquad \lambda_2 = 3-\sqrt{2}, \qquad \lambda_3 = 3+\sqrt{2},$$

a sopstveni vektori su redom vektori kolona matrice:

$$U = U_{21}\, U_{32} = \begin{pmatrix} 1/\sqrt{2} & 1/2 & 1/2 \\ -1/\sqrt{2} & 1/2 & 1/2 \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

Treba napomenuti da je ovaj zadatak specijalan slučaj, u opštem slučaju metoda ne daje tačan rezultat u konačnom broju koraka.

6.5 *Jacobi-jevom metodom odrediti sopstvene vrednosti i sopstvene vektore matrice:*

$$A = \begin{pmatrix} 4.2 & 0.4 & 0.2 \\ 0.4 & 5.6 & 0.8 \\ 0.2 & 0.8 & 6.4 \end{pmatrix}$$

sa tačnošću 10^{-2}.

Rešenje: Nedijagonalna norma polazne matrice iznosi $\nu = 0.916515$. Najveći po modulu nedijagonalni element je $a_{32} = 0.8$, pa je $\sin\phi = -0.525731$ i $\cos\phi = 0.850651$. Transformisana matrica je

$$A_1 = U_{32}^{\top} A\, U_{32} = \begin{pmatrix} 4.200000 & 0.235114 & 0.380423 \\ 0.235114 & 5.105573 & 0 \\ 0.380423 & 0 & 6.894427 \end{pmatrix}$$

Nedijagonalna norma ove matrice je $\nu_1 = 0.447214$. Njen najveći po modulu nedijagonalni element je $a_{31} = 0.380423$, pa je $\sin\phi = -0.137172$ i $\cos\phi = 0.990547$. U drugom koraku dobija se transformisana matrica

$$A_2 = U_{31}^{\top} A_1\, U_{31} = \begin{pmatrix} 4.147319 & 0.232892 & 0 \\ 0.232892 & 5.105573 & 0.032251 \\ 0 & 0.032251 & 6.947109 \end{pmatrix}$$

Nedijagonalna norma ove matrice iznosi $\nu_2 = 0.235114$. Najveći po modulu nedijagonalni element je $a_{21} = 0.232892$, pa je $\sin\phi = -0.224298$ i $\cos\phi = 0.974521$. Transformisana matrica je

$$A_3 = U_{21}^{\top} A_2\, U_{21} = \begin{pmatrix} 4.093716 & 0 & -0.007234 \\ 0 & 5.159176 & 0.031429 \\ -0.007234 & 0.031429 & 6.947109 \end{pmatrix}$$

Nedijagonalna norma ove matrice iznosi $\nu_3 = 0.032251$. Najveći po modulu nedijagonalni element je $a_{32} = 0.031429$, pa je $\sin\phi = -0.017570$ i $\cos\phi = 0.999846$. Transformisana matrica je

$$A_4 = \overline{U}_{32}^{\top} A_3\, \overline{U}_{32} = \begin{pmatrix} 4.093716 & 0.000127 & -0.007233 \\ 0.000127 & 5.158623 & 0 \\ -0.007233 & 0 & 6.947661 \end{pmatrix}$$

Nedijagonalna norma ove matrice iznosi $\nu_4 = 0.007234$ i zadovoljava uslov $\nu_4 < 10^{-2}\nu$, što znači da je tačnost postignuta. Sopstvene vrednosti polazne matrice su

$$\lambda_1 = 4.093716, \qquad \lambda_2 = 5.158623 \qquad \lambda_3 = 6.947661.$$

Sopstveni vektori su vektori kolona matrice koja je jednaka proizvodu matrica rotacija

$$U = U_{32}\,U_{31}\,U_{21}\,\overline{U}_{32} = \begin{pmatrix} 0.966 & 0.220 & 0.139 \\ -0.261 & 0.804 & 0.536 \\ 0.004 & -0.553 & 0.833 \end{pmatrix}$$

6.6 *Jacobi-jevom metodom sa tačnošću* 10^{-3} *odrediti sopstvene vrednosti i vektore matrice*

$$A = \begin{pmatrix} 8 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & 2 & 5 \end{pmatrix}.$$

Rešenje: Matricama transformacije

$$U_{12}^{(1)} = \begin{pmatrix} 0.85065 & -0.52573 & 0 \\ 0.52573 & 0.85065 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad U_{13}^{(1)} = \begin{pmatrix} 0.93381 & 0 & -0.35776 \\ 0 & 1 & 0 \\ 0.35776 & 0 & 0.93381 \end{pmatrix}$$

$$U_{23}^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.78069 & -0.62492 \\ 0 & 0.62492 & 0.78069 \end{pmatrix} \qquad U_{12}^{(2)} = \begin{pmatrix} 0.99716 & -0.07532 & 0 \\ 0.07532 & 0.99716 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$U_{13}^{(2)} = \begin{pmatrix} 0.99921 & 0 & 0.03963 \\ 0 & 1 & 0 \\ -0.03963 & 0 & 0.99921 \end{pmatrix} \qquad U_{23}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.99996 & -0.00885 \\ 0 & 0.00885 & 0.99996 \end{pmatrix}$$

matrica A se transformiše u njoj sličnu skoro dijagonalnu matricu

$$\tilde{A} = (U_{23}^{(2)})^{\top}(U_{13}^{(2)})^{\top}(U_{12}^{(2)})^{\top}(U_{23}^{(1)})^{\top}(U_{13}^{(1)})^{\top}(U_{12}^{(1)})^{\top}\,A\,U_{12}^{(1)}U_{13}^{(1)}U_{23}^{(1)}U_{12}^{(2)}U_{13}^{(2)}U_{23}^{(2)}$$

$$= \begin{pmatrix} 10.00000 & -0.00079 & 0 \\ -0.00079 & 5.61803 & 0 \\ 0 & 0 & 3.38197 \end{pmatrix}$$

Zbog sličnosti ovih matrica sopstvene vrednosti su dijagonalni elementi matrice $\tilde{A}$, a sopstveni vektori su vektori kolona matrice $U = U_{12}^{(1)}U_{13}^{(1)}U_{23}^{(1)}U_{12}^{(2)}U_{13}^{(2)}U_{23}^{(2)}$,

$$\lambda_1 = 10.000 \qquad \lambda_2 = 5.618 \qquad \lambda_3 = 3.38297$$

$$\mathbf{x}_1 = \begin{pmatrix} 0.743 \\ 0.557 \\ 0.371 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} -0.658 \\ 0.502 \\ 0.562 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 0.126 \\ -0.661 \\ 0.739 \end{pmatrix}$$

6.7 *Jacobi-jevom metodom sa tačnošću* 10^{-2} *odrediti sopstvene vrednosti i vektore matrice*

$$A = \begin{pmatrix} 6.00 & 1.34 & 1.45 \\ 1.34 & 6.00 & 1.46 \\ 1.45 & 1.46 & 6.00 \end{pmatrix}.$$

Rešenje:

$$\lambda_1 = 8.83 \qquad \lambda_2 = 4.66 \qquad \lambda_3 = 4.51$$

$$\mathbf{x}_1 = \begin{pmatrix} 0.57 \\ 0.58 \\ 0.58 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 0.72 \\ -0.68 \\ -0.03 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} -0.38 \\ -0.44 \\ 0.81 \end{pmatrix}$$

C Implementacija Jacobi-jeve metode ima dosta sličnosti sa implementacijom Givens-ove metode. I ovde se umesto množenja polazne matrice sa matricom rotacije samo izračunavaju nove vrednosti onih elemenata polazne matrice koji bivaju izmenjeni. Za računanje koeficijenata matrice rotacije koriste se stabilne formule. Postupak anuliranja nedijagonalnih elemenata date matrice se nastavlja u petlji sve dok odnos nedijagonalnih normi tekuće i polazne matrice ne postane manji od željene tačnosti. Treba uočiti kako je u uslovu petlje iskorišćeno množenje umesto deljenja koje je skuplja operacija. Procedura koja implementira metodu ima oblik:

```
#include <assert.h>
#include <float.h>
#include <math.h>
#include <stdlib.h>

/* Makro za odredjivanje znaka. */
#define sign(x) ((x)<0 ? -1 : 1)

/*
 * Funkcija jacobi() racuna sopstvene vrednosti simetricne matrice
 * Jacobi-jevom metodom. Argumenti funkcije su:
 *   n - dimenzija matrice
 *   A - matrica cije se sopstvene vrednosti izracunavaju (ova matrica
 *       biva izmenjena tokom izracunavanja i sopstvene vrednosti se
 *       nakon izracunavanja ocitavaju sa njene glavne dijagonale)
 *   epsilon - zeljena tacnost u vidu odnosa nedijagonalnih normi
 *             poslednje i polazne matrice
 * Dimenzija matrice i tacnost moraju biti brojevi veci od 0. Matrica A
 * mora biti simetricna matrica.
 */
void
jacobi(int n, double **A, double epsilon)
{
        double          nu,
                        nu_curr;        /* Nedijagonalne norme polazne i
                                         * tekuce matrice. */
        int             i,
                        j;      /* Indeksi u petljama. */
```

```
int             k,
                l;      /* Indeksi elementa koji se anulira u
                         * tekucem koraku. */
double          max;    /* Promenljiva koja cuva tekuci maksimalan
                         * element. */
double          lambda,
                mu,
                omega;  /* Promenljive koje se koriste pri
                         * izracunavanju ugla rotacije. */
double          sin_fi,
                cos_fi; /* Sinus i kosinus ugla rotacije. */

/* Proverava se da li su uslovi zadatka ispunjeni. */
assert(n > 0);
for (j = 0; j < n - 1; j++)
        for (i = j + 1; i < n; i++)
                assert(A[i][j] == A[j][i]);
assert(epsilon > 0);

/* Obradjuje se specijalni slucaj matrice sa dimenzijom 1. */
if (n == 1)
        return;

/* Izracunava se nedijagonalna norma polazne matrice. */
nu = 0;
for (j = 0; j < n - 1; j++)
        for (i = j + 1; i < n; i++)
                nu += A[i][j] * A[i][j];
nu = sqrt(nu);

/* Obradjuje se specijalni slucaj dijagonalne matrice. */
if (nu == 0)
        return;

for (;;) {
        /* Odredjuje se maksimalni nedijagonalan element tekuce
         * matrice. */
        max = -DBL_MAX;
        for (j = 0; j < n - 1; j++)
                for (i = j + 1; i < n; i++)
                        if (max < fabs(A[i][j]))
                                k = j, l = i, max =
                                    (double) fabs(A[i][j]);

        /* Odredjuju se koeficijenti rotacije. */
        lambda = -A[k][l];
        mu = (A[l][l] - A[k][k]) / 2;
        omega =
            sign(mu) * lambda /
            ((double) sqrt(lambda * lambda + mu * mu));
        sin_fi =
            omega /
            ((double) sqrt(2 * (1 + sqrt(1 - omega * omega))));
        cos_fi = (double) sqrt(1 - sin_fi * sin_fi);

        /* Izracunavaju se nove vrednosti elemenata u k-toj i
         * l-toj koloni. */
```

```
                for (i = 0; i < n; i++) {
                        double          a_ik,
                                        a_il;   /* Prethodne vrednosti
                                                 * elemenata A[i][k] i
                                                 * A[i][l]. */

                        a_ik = A[i][k], a_il = A[i][l];
                        A[i][k] = cos_fi * a_ik + sin_fi * a_il;
                        A[i][l] = -sin_fi * a_ik + cos_fi * a_il;
                }

                /* Izracunavaju se nove vrednosti elemenata u k-toj i
                 * l-toj vrsti. */
                for (j = 0; j < n; j++) {
                        double          a_kj,
                                        a_lj;   /* Prethodne vrednosti
                                                 * elemenata A[k][j] i
                                                 * A[l][j]. */

                        a_kj = A[k][j], a_lj = A[l][j];
                        A[k][j] = cos_fi * a_kj + sin_fi * a_lj;
                        A[l][j] = -sin_fi * a_kj + cos_fi * a_lj;
                }

                /* Izracunava se nedijagonalna norma tekuce matrice. */
                nu_curr = 0;
                for (j = 0; j < n - 1; j++)
                        for (i = j + 1; i < n; i++)
                                nu_curr += A[i][j] * A[i][j];
                nu_curr = sqrt(nu_curr);

                /* Ako je odnos nedijagonalne norme tekuce i polazne
                 * matrice postao manji od zadate vrednosti, iterativni
                 * postupak se prekida. */
                if (nu_curr <= epsilon * nu)
                        break;
        }
}
```

6.3 Householder-ova metoda

Ovom metodom se matrica transformiše u njoj sličnu Hessenberg-ovu ili trodijagonalnu matricu pomoću matrica oblika

$$T = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & H & \\ 0 & 0 & & & \end{pmatrix}$$

Householder-ova matrica H kojom se proizvoljan vektor $\mathbf{x} = (x_1, \dots, x_m)^\top$ preslikava u vektor kolinearan koordinatnom vektoru $\mathbf{e}_1 = (1, 0, \dots, 0)^\top$ određena je formulama

$$H = I - \beta \mathbf{u}\,\mathbf{u}^*, \qquad \mathbf{u} = \mathbf{x} - k\mathbf{e}_1, \qquad \beta = \big(\sigma(\sigma + |x_1|)\big)^{-1}$$

$$\sigma = \sqrt{\sum_{i=1}^{m} |x_i|^2}, \qquad x_1 = |x_1| \exp(\imath\alpha), \qquad k = -\sigma \exp(\imath\alpha)$$

Proizvoljni vektor kolona x se množi matricom H, bez njenog eksplicitnog konstruisanja, na osnovu veze

$$H\mathbf{x} = \mathbf{x} - \beta(\mathbf{u}^*\mathbf{x})\mathbf{u}$$

6.8 *Householder-ovom metodom redukovati matricu:*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

na trodijagonalnu.

Rešenje: Parametri koji određuju matricu transformacije su

$$\sigma = \sqrt{2}, \quad k = -\sqrt{2}, \quad \beta = \big(\sqrt{2}(\sqrt{2}+1)\big)^{-1}, \quad \mathbf{u} = \begin{pmatrix} 1+\sqrt{2} & 1 \end{pmatrix}^\top,$$

Householder-ova matrica je

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{1}{\sqrt{2}(\sqrt{2}+1)} \begin{pmatrix} 1+\sqrt{2} \\ 1 \end{pmatrix} \begin{pmatrix} 1+\sqrt{2} & 1 \end{pmatrix} = \begin{pmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

a matrica transformacije i transformisana matrica su

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1/\sqrt{2} & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \qquad A_1 = T\,A\,T = \begin{pmatrix} 2 & -\sqrt{2} & 0 \\ -\sqrt{2} & 7/2 & -1/2 \\ 0 & -1/2 & 3/2 \end{pmatrix}$$

6.9 *Householder-ovom redukcijom svesti na jednostavniju formu matricu*

$$A = \begin{pmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{pmatrix}.$$

Računati na šest decimala.

Rešenje: Matricama transformacije

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.94281 & -0.23570 & -0.23570 \\ 0 & -0.23570 & 0.97140 & -0.02860 \\ 0 & -0.23570 & -0.02860 & 0.97140 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.70711 & -0.70711 \\ 0 & 0 & -0.70711 & 0.70711 \end{pmatrix}$$

polazna matrica se svodi na trodijagonalnu matricu njoj sličnu

$$\tilde{A} = T_2\,T_1\,A\,T_1\,T_2 = \begin{pmatrix} 5 & -4.24264 & 0 & 0 \\ -4.24264 & 6.00000 & 1.41421 & 0 \\ 0 & 1.41421 & 5.00000 & 0 \\ 0 & 0 & 0 & 2.00000 \end{pmatrix}$$

6.10 *Householder-ovom metodom odrediti trodijagonalnu matricu sličnu matrici*

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix}.$$

Računati na šest decimala.

Rešenje:

$$\tilde{A} = \begin{pmatrix} 1.000000 & -5.385165 & 0 & 0 \\ -5.385165 & 5.137932 & -1.995238 & 0 \\ 0 & -1.995238 & -1.374490 & 0.289525 \\ 0 & 0 & 0.289525 & -0.763441 \end{pmatrix}$$

6.11 *a) Ako je* $H = I - 2\mathbf{w} \cdot \mathbf{w}^\top$ *(I je jedinična matrica), pokazati da je vektor* $\mathbf{w}$*, čija je euklidska norma* 1*, sopstveni vektor matrice* H*. Takođe pokazati da je i vektor* $\mathbf{x}$ *koji je ortogonalan na vektor* $\mathbf{w}$ *takođe sopstveni vektor matrice* H*. Koje sopstvene vrednosti odgovaraju ovim sopstvenim vektorima?*

b) Na osnovu osobina Householder-ove matrice, dokazati da su joj sve sopstvene vrednosti po modulu jednake jedan.

c) Householder-ovom redukcijom svesti na jednostavniju formu matricu

$$A = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 6 & 4 & 2 \\ 2 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

Rešenje: a) Kako je je $\|\mathbf{w}\|_2^2 = (\mathbf{w}, \mathbf{w}) = \mathbf{w}^\top \mathbf{w} = 1$, to je

$$H\,\mathbf{w} = (I - 2\mathbf{w} \cdot \mathbf{w}^\top)\mathbf{w} = \mathbf{w} - 2\mathbf{w}(\mathbf{w}, \mathbf{w}) = -\mathbf{w},$$

što znači da je $\mathbf{w}$ sopstveni vektor matrice H kome odgovara sopstvena vrednost $\lambda = -1$.

Takođe, iz ortogonalnosti $(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{x} = 0$ sledi da je

$$H\,\mathbf{x} = (I - 2\mathbf{w} \cdot \mathbf{w}^\top)\mathbf{x} = \mathbf{x} - 2\mathbf{w}(\mathbf{x}, \mathbf{w}) = \mathbf{x},$$

što znači da je i $\mathbf{x}$ sopstveni vektor matrice H sa sopstvenom vrednošću $\lambda = 1$.

b) Householder-ova matrica je Hermite-ova i unitarna, te je $H = H^* = H^{-1}$. Kako je sopstvena vrednost inverzne matrice jednaka recipročnoj vrednosti sopstvene vrednosti same matrice, sledi da je

$$\lambda(H) = \frac{1}{\lambda(H^{-1})} = \frac{1}{\lambda(H)}, \quad \text{te je} \quad \big(\lambda(H)\big)^2 = 1, \quad \text{tj.} \quad \lambda(H) = \pm 1.$$

c) Householder-ovom redukcijom se pomoću matrica transformacije

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.80178 & -0.53452 & -0.26726 \\ 0 & -0.53452 & 0.84143 & -0.07929 \\ 0 & -0.26726 & -0.07929 & 0.96036 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.74863 & -0.66299 \\ 0 & 0 & -0.66299 & 0.74863 \end{pmatrix}$$

dobija trodijagonalna matrica slična polaznoj

$$\tilde{A} = T_2\,T_1\,A\,T_1\,T_2 = \begin{pmatrix} 4 & -3.74166 & 0 & 0 \\ -3.74166 & 11.00000 & 2.31455 & 0 \\ 0 & 2.31455 & 3.33333 & -0.44544 \\ 0 & 0 & -0.44544 & 1.66667 \end{pmatrix}$$

C Pri implementaciji Householder-ove redukcije se nije išlo na izbegavanje matričnog množenja (ova tehnika će biti prikazana u poglavlju o MATLAB-u), tako da se na početku procedure alocira memorija za dve pomoćne matrice koje će zbog matričnog množenja biti potrebne. Zatim se u jednoj petlji anuliraju elementi ispod dve glavne dijagonale po kolonama polazne matrice. Treba uočiti kako su ovoj petlji donekle svedene formule koje služe za izračunavanje matrice transformacije, a u cilju upotrebe minimalnog broja promenljivih. Procedura `householder()` ima sledeći oblik:

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>

/*
 * Funkcija householder() svodi datu matricu na gornju Hessenberg-ovu
 * matricu Householder-ovom metodom. Argumenti funkcije su:
 *   n - dimenzija matrice
 *   A - data matrica, cije ce vrednosti nakon poziva funkcije biti
 *       izmenjene
 * Dimenzija ulazne matrice mora biti veca od 0.
 */
void
householder(int n, double **A)
{
        double        **T;      /* Matrica transformacije u datom koraku
                                 * algoritma. */
        double        **TA;     /* Matrica za cuvanje medjurezultata
                                 * prilikom mnozenja. */
        int             col;    /* Indeks kolone matrice A u kojoj se vrsi
                                 * svodjenje. */
        int             i,
                        j,
                        k;      /* Brojaci u petljama. */

        /* Proverava se validnost ulaznih parametara. */
        assert(n > 0);

        /* Alocira se memorija za matrice T i TA. */
        T = (double **) malloc(n * sizeof(double *));
        for (i = 0; i < n; i++)
                T[i] = (double *) malloc(n * sizeof(double));
        TA = (double **) malloc(n * sizeof(double *));
        for (i = 0; i < n; i++)
                TA[i] = (double *) malloc(n * sizeof(double));

        /* Anuliraju se svi elementi ispod glavne dijagonale (osim prvog)
         * u jednoj po jednoj koloni polazne matrice. */
        for (col = 0; col < n - 2; col++) {
                int             length; /* Velicina vektora koji se
                                         * anulira. */
                double         *x;      /* Vektor koji se svodi na prvi
                                         * koordinatni vektor u datom
                                         * koraku algoritma. */
                double          sigma,
                                beta;   /* Pomocne promenljive za
                                         * racunanje Householder-ove
```

```
                                * matrice. */

        /* Odredjuje se velicina i alocira memorija za vektor koji
         * se svodi na prvi koordinatni vektor. */
        length = n - (col + 1);
        x = (double *) malloc(length * sizeof(double));

        /* Popunjavaju se elementi ovog vektora. */
        for (i = 0; i < length; i++)
                x[i] = A[i + (col + 1)][col];

        /* Izracunava se norma ovog vektora. Ukoliko se eventualno
         * ustanovi da je vektor vec oblika prvog koordinatnog
         * vektora, odmah se prelazi na narednu iteraciju. */
        sigma = 0;
        for (i = 1; i < length; i++)
                sigma += x[i] * x[i];
        if (sigma == 0) {
                free(x);
                continue;
        }
        sigma += x[0] * x[0];
        sigma = sqrt(sigma);

        /* Izracunava se vrednost pomocne promenljiva beta. */
        beta = 1 / (sigma * (sigma + fabs(x[0])));

        /* Izracunava se Householder-ova matrica. */
        x[0] -= (x[0] > 0) ? -sigma : sigma;
        for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                        T[i][j] = (i == j) ? 1 : 0;
        for (i = 0; i < length; i++)
                for (j = 0; j < length; j++)
                        T[i + (col + 1)][j + (col + 1)] -=
                            beta * x[i] * x[j];

        /* Mnozi se polazna matrica sa leve i desne strane
         * Householder-ovom matricom cime se anuliraju svi
         * elementi ispod glavne dijagonale (osim prvog) u j-toj
         * koloni polazne matrice. */
        for (i = 0; i < n; i++)
                for (j = 0; j < n; j++) {
                        TA[i][j] = 0;
                        for (k = 0; k < n; k++)
                                TA[i][j] += T[i][k] * A[k][j];
                }
        for (i = 0; i < n; i++)
                for (j = 0; j < n; j++) {
                        A[i][j] = 0;
                        for (k = 0; k < n; k++)
                                A[i][j] += TA[i][k] * T[k][j];
                }

        /* Oslobadja se vektor koji je svodjen. */
        free(x);
}
```

```
        /* Oslobadja se zauzeta memorija. */
        for (i = 0; i < n; i++)
                free(T[i]);
        free(T);
        for (i = 0; i < n; i++)
                free(TA[i]);
        free(TA);
}
```

MATLAB

Householder-ova matrica `[H, y] = qr(x)`

Householder-ovu matricu koja anulira sve osim prvog elementa datog vektora `x` moguće je odrediti pozivom `[H, y] = qr(x)`. Matrica H je unitarna i Hermitska i važi da je $H\mathbf{x} = \mathbf{y}$, pri čemu je $y_i = 0$, za $i > 1$.

6.12 *Napisati* MATLAB *skript koji za datu realnu kvadratnu matricu A određuje matricu $B = H \cdot A$ kojoj su anulirani svi elementi prve kolone osim njenog prvog elementa. Matricu B odrediti bez eksplicitnog konstruisanja Householder-ove matrice H.*

Rešenje: Matrica A se sa leve strane množi matricom H tako što se matricom H pomnoži posebno svaka njena kolona.

Važna osobina Householder-ovih matrica je da se one ne moraju eksplicitno konstruisati kako bi se njima pomnožio dati vektor. Naime, Householder-ove matrice $H = I - \beta\mathbf{u}\mathbf{u}^*$ su u potpunosti određene vrednošću vektora $\mathbf{u}$ i skalara β i važi da je

$$H\mathbf{x} = (I - \beta\mathbf{u}\mathbf{u}^*)\mathbf{x} = \mathbf{x} - \beta(\mathbf{u}^*\mathbf{x})\mathbf{u}$$

Ovo znači da je proizvod matrice i vektora moguće naći u linearnom vremenu, tačnije vremenu potrebnom da se skalarno pomnože vektori $\mathbf{u}^*$ i $\mathbf{x}$ i zatim da se od vektora $\mathbf{x}$ oduzme vektor $\mathbf{u}$ skaliran proizvodom dobijenog skalara i skalara β.

Matrica A se sa desne strane množi matricom H tako što se matricom H pojedinačno množi svaka njena vrsta. Vektori vrste se množe Householder-ovom matricom H sa desna na način veoma sličan množenju vektora kolona matricom H sa leva.

```
% A je slucajna matrica dimenzije n = 5
n = 5; A = rand(n);

% Odredjujemo parametre Householder-ove matrice H
sigma = norm(A(:,1));
beta = 1/(sigma*(sigma+A(1,1)));
u = A(:,1);  u(1) = u(1) + sigma;

% Mnozenje B1 = H*A bez konstruisanja matrice H
B1 = zeros(n); % prealokacija
```

```
for j = 1:n
    B1(:, j) = A(:, j) - beta*u'*A(:,j)*u;
end

% Mnozenje B2 = H*A sa konstruisanjem matrice H
H = eye(n) - beta*u*u';
B2 = H * A;

% Rezultati
B1, B2
```

6.4 QR metoda

QR metoda je definisana iterativnim algoritmom

$$A^{(0)} = A, \qquad A^{(n)} = Q^{(n)}R^{(n)}, \qquad A^{(n+1)} = R^{(n)}Q^{(n)}, \quad n = 0, 1, \ldots,$$

gde je $Q^{(n)}$ unitarna matrica (na primer, proizvod matrica rotacije ili Householder-ovih matrica) i $R^{(n)}$ gornje trougaona matrica.
Za dovoljno veliko n je $A^{(n)}$ skoro gornje trougaona matrica čije se sopstvene vrednosti očitavaju sa dijagonale. Zbog sličnosti $A \sim A^{(n)}$ to su približno i sopstvene vrednosti matrice A.

LR metoda je definisana iterativnim algoritmom

$$A^{(0)} = A, \qquad A^{(n)} = L^{(n)}R^{(n)}, \qquad A^{(n+1)} = R^{(n)}L^{(n)}, \quad n = 0, 1, \ldots,$$

gde su matrice $L^{(n)}$ i $R^{(n)}$ određene LU dekompozicijom. Za dovoljno veliko n je $A^{(n)}$ skoro gornje trougaona matrica čije se sopstvene vrednosti očitavaju sa dijagonale. Zbog sličnosti $A \sim A^{(n)}$ to su približno i sopstvene vrednosti matrice A.

6.13 *Uraditi jedan korak QR metode za matricu*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Rešenje: Matrica A se prvo Householder-ovom metodom transformiše u sebi sličnu trodijagonalnu matricu (zad.7)

$$A^{(0)} = \begin{pmatrix} 2 & -\sqrt{2} & 0 \\ -\sqrt{2} & 7/2 & -1/2 \\ 0 & -1/2 & 3/2 \end{pmatrix}$$

Sada se matrica $A^{(0)}$ dekomponuje u proizvod unitarne i gornje trougaone matrice $A^{(0)} = Q^{(0)}R^{(0)}$. Da bi se odredila ova dekompozicija, matrica $A^{(0)}$ se svodi Givens-

ovom metodom rotacije na gornje trougaonu matricu $R^{(0)}$.

Prvo se anulira element a_{21},

$$U_{21} = \begin{pmatrix} 0.81650 & -0.57735 & 0 \\ 0.57735 & 0.81650 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad U_{21}A^{(0)} = \begin{pmatrix} 2.44950 & -3.17543 & 0.28868 \\ 0 & 2.04125 & -0.40825 \\ 0 & -0.5 & 1.5 \end{pmatrix}$$

a zatim se anulira element a_{32},

$$U_{32} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.97129 & -0.23792 \\ 0 & 0.23792 & 0.97129 \end{pmatrix} \quad R^{(0)} = \begin{pmatrix} 2.44950 & -3.17543 & 0.28868 \\ 0 & 2.10160 & -0.75339 \\ 0 & 0 & 1.35981 \end{pmatrix}$$

gde je

$$R^{(0)} = U_{32}\, U_{21}\, A^{(0)}$$

Matrica $Q^{(0)}$ se određuje iz relacije

$$A^{(0)} = (U_{32}\, U_{21})^{-1} R^{(0)} = (U_{32}\, U_{21})^{\top} R^{(0)} = U_{21}^{\top} U_{32}^{\top} R^{(0)} \qquad \Rightarrow \qquad Q^{(0)} = U_{21}^{\top} U_{32}^{\top}$$

i, u ovom zadatku je

$$Q^{(0)} = \begin{pmatrix} 0.81650 & 0.56077 & 0.13736 \\ 0.57735 & 0.79306 & 0.19425 \\ 0 & -0.23791 & 0.97129 \end{pmatrix}$$

Na kraju prvog koraka QR algoritma dobija se matrica

$$A^{(1)} = R^{(0)}\, Q^{(0)} \begin{pmatrix} 3.83335 & 1.21336 & 0 \\ 1.21336 & 1.84593 & -0.32351 \\ 0 & -0.32351 & 1.32077 \end{pmatrix}$$

6.14 *a) Ako je* $H = I - 2\mathbf{w}\mathbf{w}^{\top}$, *gde je* I *jedinična matrica, a* $\mathbf{w} \in \mathcal{R}^n$ *i* $\|\mathbf{w}\|_2 = 1$, *dokazati da je* H *ortogonalna matrica, tj. da je*

$$H\, H^{\top} = H^{\top} H = I.$$

b) Householder-ovom transformacijom svesti matricu

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 3 & 4 & 2 \\ 4 & 2 & 1 \end{pmatrix}$$

na trodijagonalnu matricu $A^{(0)}$.
c) Uraditi jedan korak QR algoritma za matricu $A^{(0)}$.

Rešenje: a) Kako je

$$H^\top = (I - 2\mathbf{w}\mathbf{w}^\top)^\top = I - 2(\mathbf{w}^\top)^\top \mathbf{w}^\top = I - 2\mathbf{w}\mathbf{w}^\top = H,$$

to je

$$H\,H^\top = H^\top H = (I - 2\mathbf{w}\mathbf{w}^\top)\,(I - 2\mathbf{w}\mathbf{w}^\top) = I - 4\mathbf{w}\mathbf{w}^\top + 4\mathbf{w}(\mathbf{w}^\top\mathbf{w})\mathbf{w}^\top = I,$$

s obzirom da je $\mathbf{w}^\top\mathbf{w} = (\mathbf{w},\mathbf{w}) = \|\mathbf{w}\|_2^2 = 1$.
b) Parametri za konstrukciju Householder-ove matrice su

$$\sigma = 5, \quad k = -5, \quad \beta = 1/40, \quad \mathbf{u} = (8,\,4)^\top,$$

te su ova matrica i odgovarajuća matrica transformacije

$$H = I - \beta\mathbf{u}\mathbf{u}^\top = \begin{pmatrix} -0.6 & -0.8 \\ -0.8 & 0.6 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.6 & -0.8 \\ 0 & -0.8 & 0.6 \end{pmatrix}.$$

Householder-ovom transformacijom matrice A matricom T dobija se trodijagonalna matrica

$$A^{(0)} = T\,A\,T = \begin{pmatrix} 2 & -5 & 0 \\ -5 & 4 & 2 \\ 0 & 2 & 1 \end{pmatrix}.$$

c) Jedan korak QR algoritma se realizuje matricama rotacije

$$U_{12} = \begin{pmatrix} 0.3714 & -0.9285 & 0 \\ 0.9285 & 0.3714 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad U_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8447 & -0.5352 \\ 0 & 0.5352 & 0.8447 \end{pmatrix},$$

kojima se matrica $A^{(0)}$ transformiše u gornjetrougaonu matricu $R^{(0)} = U_{23}U_{12}A^{(0)}$. Tako se matrica $A^{(0)}$ može predstaviti proizvodom ortogonalne matrice $Q^{(0)} = U_{12}^\top U_{23}^\top$ i gornjetrougaone matrice $R^{(0)}$,

$$A^{(0)} = Q^{(0)}\,R^{(0)} \qquad Q^{(0)} = \begin{pmatrix} 0.3714 & 0.7843 & 0.4969 \\ -0.9285 & 0.3137 & 0.1988 \\ 0 & -0.5352 & 0.8447 \end{pmatrix}$$

$$R^{(0)} = \begin{pmatrix} 5.3852 & -5.5709 & -1.8570 \\ 0 & -3.7370 & 0.0923 \\ 0 & 0 & 1.2422 \end{pmatrix}$$

Trodijagonalna matrica na koju se primenjuje sledeći korak QR algoritma je

$$A^{(1)} = R^{(0)}\,Q^{(0)} = \begin{pmatrix} 7.1726 & 3.4699 & 0 \\ 3.4699 & -1.2217 & -0.6648 \\ 0 & -0.6648 & 1.0493 \end{pmatrix}$$

6.15 *Householder-ovom redukcijom transformisati matricu*

$$A = \begin{pmatrix} 7 & 0 & 1 \\ 2 & 5 & 3 \\ 1 & 2 & 4 \end{pmatrix}$$

u gornju Hessenberg-ovu matricu, a zatim uraditi jedan korak QR algoritma. Računati sa četiri decimale.

Rešenje: Householder-ovom matricom T se data matrica transformiše u njoj sličnu gornju Hessenberg-ovu matricu $A^{(0)}$

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.8949 & -0.4473 \\ 0 & -0.4473 & 0.8944 \end{pmatrix}$$

$$A^{(0)} = T\,A\,T = \begin{pmatrix} 7 & -0.4473 & 0.8944 \\ -2.2371 & 6.8060 & -1.5999 \\ 0 & -0.5994 & 2.1999 \end{pmatrix}$$

Givens-ovim matricama rotacije

$$U_{12} = \begin{pmatrix} 0.9525 & -0.3044 & 0 \\ 0.3044 & 0.9525 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad U_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.9956 & -0.0940 \\ 0 & 0.0940 & 0.9956 \end{pmatrix}$$

matrica $A^{(0)}$ se transformiše u gornje trougaonu matricu $R^{(0)}$. Stoga je nova matrica QR algoritma slična matrici A

$$A^{(1)} = R^{(0)}\,Q^{(0)} = \begin{pmatrix} 7.7600 & -0.2671 & 1.3195 \\ -1.9406 & 6.1820 & -0.8754 \\ 0 & -0.1948 & 2.0634 \end{pmatrix}$$

gde je

$$R^{(0)} = \begin{pmatrix} 7.3487 & -2.4978 & 1.3390 \\ 0 & 6.3750 & -1.4530 \\ 0 & 0 & 2.0725 \end{pmatrix}$$

$$Q^{(0)} = (U_{23}U_{12})^{\top} = \begin{pmatrix} 0.9525 & 0.3031 & 0.0286 \\ -0.3044 & 0.9483 & 0.0896 \\ 0 & -0.0940 & 0.9956 \end{pmatrix}$$

6.16 *Householder-ovom redukcijom transformisati matricu*

$$A = \begin{pmatrix} 1 & 3 & 4 \\ 3 & 2 & 1 \\ 4 & 1 & 0 \end{pmatrix}$$

u trodijagonalnu matricu, a zatim uraditi jedan korak QR algoritma. Računati sa četiri decimale.

Rešenje: Householder-ova i dobijena trodijagonalna matrica su

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.6 & -0.8 \\ 0 & -0.8 & 0.6 \end{pmatrix} \qquad A^{(0)} = T\,A\,T = \begin{pmatrix} 1 & -5 & 0 \\ -5 & 1.68 & 1.24 \\ 0 & 1.24 & 0.32 \end{pmatrix}$$

Givens-ovim matricama rotacije anulira se poddijagonala

$$U_{12} = \begin{pmatrix} 0.1961 & -0.9806 & 0 \\ 0.9806 & 0.1961 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad U_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.9652 & -0.2617 \\ 0 & 0.2617 & 0.9652 \end{pmatrix}$$

Nova matrica QR algoritma slična matrici A

$$A^{(1)} = R^{(0)}\,Q^{(0)} = \begin{pmatrix} 3.5718 & 4.6464 & 0 \\ 4.6464 & -0.9365 & -0.0975 \\ 0 & -0.0975 & 0.3595 \end{pmatrix}$$

gde je

$$R^{(0)} = \begin{pmatrix} 5.0990 & -2.6280 & -1.2159 \\ 0 & -4.7385 & 0.1510 \\ 0 & 0 & 0.3725 \end{pmatrix}$$

$$Q^{(0)} = (U_{23}U_{12})^{\top} = \begin{pmatrix} 0.1961 & 0.9464 & 0.2566 \\ -0.9806 & 0.1893 & 0.0513 \\ 0 & -0.2617 & 0.9652 \end{pmatrix}$$

6.17 *Transformacijama pomoću Householder-ovih matrica realizovati jedan korak QR algoritma za matricu*

$$A = \begin{pmatrix} 2.22 & 0.46 & 0.50 \\ 0.34 & 2.22 & 0.48 \\ 0.46 & 0.52 & 2.22 \end{pmatrix}$$

Rešenje: Množenjem Householder-ovim matricama

$$T_1 = \begin{pmatrix} -0.9685 & -0.1483 & -0.2007 \\ -0.1483 & 0.9888 & -0.0151 \\ -0.2007 & -0.0151 & 0.9795 \end{pmatrix} \qquad T_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.9840 & -0.1781 \\ 0 & -0.1781 & 0.9840 \end{pmatrix}$$

određujemo dekompoziciju matrice A na unitarnu matricu $Q^{(0)}$

$$Q^{(0)} = (T_2\,T_1)^{\top} = \begin{pmatrix} -0.9685 & 0.1817 & -0.1711 \\ -0.1483 & -0.9703 & -0.1910 \\ -0,2007 & -0.1596 & 0.9665 \end{pmatrix}$$

i gornje trougaonu matricu $R^{(0)}$

$$R^{(0)} = T_2\, T_1\, A = \begin{pmatrix} -2.2928 & -0.8791 & -1.0010 \\ 9 & -2.1535 & -0.7292 \\ 0 & 0 & 1.9685 \end{pmatrix}$$

Sledeća matrica u nizu sličnih matrica određenih QR algoritmom je

$$A^{(1)} = R^{(0)}\, Q^{(0)} = \begin{pmatrix} 2.5518 & 0.5961 & -0.4073 \\ 0.4657 & 2.2059 & -0.2935 \\ -0.3951 & -0.3142 & 1.9026 \end{pmatrix}$$

6.18 *Householder-ovom metodom svesti matricu*

$$A = \begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & 1.6 & 2 \end{pmatrix}$$

na trodijagonalnu i uraditi jedan korak QR metode.

Rešenje: U prvom koraku parametri Householder-ove metode su

$$\sigma = 2.2913, \quad k = -2.2913, \quad \beta = 0.1326, \quad \mathbf{u} = \begin{pmatrix} 3.2913 & 0.5 & 2 \end{pmatrix}^\top$$

pa je matrica transformacije je

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.4364 & -0.2182 & -0.8729 \\ 0 & -0.2182 & 0.9668 & -0.1326 \\ 0 & -0.8729 & -0.1326 & 0.4696 \end{pmatrix}$$

Transformisana matrica slična polaznoj je

$$A_1 = T_1\, A\, T_1 = \begin{pmatrix} 2.2 & -2.2913 & 0 & 0 \\ -2.2913 & 3.5476 & -1.5546 & 0.7649 \\ 0 & -1.5546 & -0.6319 & -0.8032 \\ 0 & 0.7649 & -0.8032 & 0.8842 \end{pmatrix}$$

U drugom koraku parametri Householder-ove metode su

$$\sigma = 1.7325, \quad k = 1.7325, \quad \beta = 0.1756, \quad \mathbf{u} = \begin{pmatrix} -3.2871 & 0.7649 \end{pmatrix}^\top$$

i matrica transformacije je

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.8972 & 0.4415 \\ 0 & 0 & 0.4415 & 0.8973 \end{pmatrix}$$

Konačno, trodijagonalna matrica slična polaznoj je

$$A^{(0)} = T_2 A_1 T_2 = \begin{pmatrix} 2.2 & -2.2913 & 0 & 0 \\ -2.2913 & 3.5476 & 1.7325 & 0 \\ 0 & 1.7325 & 0.3000 & 1.0906 \\ 0 & 0 & 1.0906 & -0.0476 \end{pmatrix}$$

Matricu $A^{(0)}$ razlažemo na proizvod unitarne i gornje trougaone matrice $A^{(0)} = Q^{(0)} R^{(0)}$ primenom Givens-ove metode rotacije.

Prvo se svodi na nulu element a_{21} matricom rotacije:

$$U_{12} = \begin{pmatrix} 0.6916 & -0.7213 & 0 & 0 \\ 0.7213 & 0.6916 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

što daje matricu

$$U_{12}\, A^{(0)} = \begin{pmatrix} 3.1765 & -4.1459 & -1.2497 & 0 \\ 0 & 0.8043 & 1.1999 & 0 \\ 0 & 1.7325 & 0.3000 & 1.0906 \\ 0 & 0 & 1.0906 & -0.0476 \end{pmatrix}$$

Zatim se svodi na nulu element a_{32} matricom rotacije

$$U_{23} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.4211 & 0.9070 & 0 \\ 0 & -0.9070 & 0.4211 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

što daje matricu

$$U_{23}\, U_{12}\, A^{(0)} = \begin{pmatrix} 3.1765 & -4.1459 & -1.2497 & 0 \\ 0 & 1.9100 & 0.7773 & 0.9893 \\ 0 & 0 & -0.9621 & 0.4592 \\ 0 & 0 & 1.0907 & -0.0475 \end{pmatrix}$$

Konačno, na nulu se svodi element a_{43} matricom rotacije

$$U_{34} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.6615 & -0.7499 \\ 0 & 0 & 0.7499 & 0.6615 \end{pmatrix}$$

tako da se dobija gornje trougaona matrica

$$R^{(0)} = U_{34}\, U_{23}\, U_{12}\, A^{(0)} = \begin{pmatrix} 3.1765 & -4.1459 & -1.2497 & 0 \\ 0 & 1.9101 & 0.7773 & 0.9893 \\ 0 & 0 & -1.4544 & 0.3394 \\ 0 & 0 & 0 & 0.3130 \end{pmatrix}$$

Unitarna matrica $Q^{(0)}$ je

$$Q^{(0)} = U_{12}^{\top} U_{23}^{\top} U_{34}^{\top} = \begin{pmatrix} 0.6926 & 0.3037 & -0.4328 & -0.4907 \\ -0.7213 & 0.2916 & -0.4156 & -0.4711 \\ 0 & 0.9070 & 0.2786 & 0.3158 \\ 0 & 0 & -0.7499 & 0.6615 \end{pmatrix}$$

Posle prvog koraka QR algoritma dobija se trodijagonalna matrica

$$A^{(1)} = R^{(0)} Q^{(0)} = \begin{pmatrix} 5.1906 & -1.3778 & 0 & 0 \\ -1.3778 & 1.2620 & -1.3191 & 0 \\ 0 & -1.3191 & -0.6597 & -0.2347 \\ 0 & 0 & -0.2347 & 0.2070 \end{pmatrix}$$

Ponavljajući ovaj postupak dobijamo da su sopstvene vrednosti date matrice

$$\lambda_1 = 5.6520, \quad \lambda_2 = 1.5454, \quad \lambda_3 = -1.4201, \quad \lambda_4 = 0.2226.$$

C Pri implementaciji QR metode, prvo se poziva procedura `householder()` koja vrši svođenje polazne matrice. Zatim se u petlji vrši QR dekompozicija i izračunava nova vrednost date matrice; ovaj postupak se ponavlja sve dok se ne postigne željena tačnost. Kriterijum za prekid iterativnog postupka je isti kao kod Jacobi-jeve metode.

QR dekompozicija se vrši matricama rotacije. Parametar φ se određuje tako da matrica rotacije $G_{k,l}(\varphi)$ anulira element na poziciji (l, k) matrice A. Ukoliko je $B \equiv G_{k,l}(\varphi) \cdot A$ važi da je

$$\begin{aligned} b_{k,i} &= a_{k,i}\cos(\varphi) - a_{l,i}\sin(\varphi) \\ b_{l,i} &= a_{k,i}\sin(\varphi) + a_{l,i}\cos(\varphi), \quad i = \overline{1,n} \\ b_{i,j} &= a_{i,j}, \quad j \neq k, l \end{aligned}$$

Iz uslova da je $b_{l,k}$ nula, sledi da je

$$b_{l,k} = a_{k,k}\sin(\varphi) + a_{l,k}\cos(\varphi) = 0$$

Može se pretpostaviti da je $a_{l,k} \neq 0$ jer je u suprotnom $G_{k,l}(\varphi) = I$. Ukoliko podelimo prethodnu jednačinu sa $\sqrt{a_{k,k}^2 + a_{l,k}^2}$ sledi da je

$$\sin(\varphi) = -\frac{a_{l,k}}{\sqrt{a_{k,k}^2 + a_{l,k}^2}}, \quad \cos(\varphi) = \frac{a_{k,k}}{\sqrt{a_{k,k}^2 + a_{l,k}^2}}$$

Procedura `qr()` na taj način ima oblik:

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
```

```
extern void     householder(int n, double **A);

/*
 * Funkcija qr() racuna sopstvene vrednosti zadate matrice QR metodom.
 * Argumenti funkcije su:
 *   n - dimenzija matrice
 *   A - matrica cije se sopstvene vrednosti izracunavaju, a koja ce
 *       biti izmenjena tokom iterativnog postupka (sopstvene vrednosti
 *       polazne matrice bice vrednosti duz glavne dijagonale na kraju
 *       postupka)
 *   epsilon - trazena tacnost u vidu odnosa nedijagonalnih normi
 *             poslednje i polazne matrice
 * Dimenzija matrice i trazena tacnost moraju biti brojevi veci od 0.
 */
void
qr(int n, double **A, double epsilon)
{
        double          nu,
                        nu_curr;        /* Donja dijagonalna norma polazne
                                         * i tekuce matrice. */
        double        **Q;      /* Unitarna matrica. */
        double        **R;      /* Gornja trougaona matrica. */
        int             i,
                        j,
                        k;      /* Brojaci u petljama. */

        /* Proveravaju se uslovi zadatka. */
        assert(n > 0);
        assert(epsilon > 0);

        /* Funkcijom householder() se transformise polazna matrica u
         * gornju Hessenberg-ovu matricu. */
        householder(n, A);

        /* Racuna se donja dijagonalna norma polazne matrice. */
        nu = 0;
        for (j = 0; j < n - 1; j++)
                for (i = j + 1; i < n; i++)
                        nu += A[i][j] * A[i][j];
        nu = sqrt(nu);

        /* Alocira se memorija za matrice Q i R. */
        Q = (double **) malloc(n * sizeof(double *));
        for (i = 0; i < n; i++)
                Q[i] = (double *) malloc(n * sizeof(double));
        R = (double **) malloc(n * sizeof(double *));
        for (i = 0; i < n; i++)
                R[i] = (double *) malloc(n * sizeof(double));

        /* U svakom koraku algoritma tekuca matrica A se dekomponuje na
         * proizvod unitarne matrice Q i gornje trougaone matrice R. Ova
         * dekompozicija se vrsi svodjenjem matrice A na matricu R nizom
         * transformacija matricama rotacije. */
        for (;;) {
                /* Matrica Q se inicijalizuje na jedinicnu matricu, a
                 * matrica R na tekucu matricu A. */
                for (i = 0; i < n; i++)
```

```
        for (j = 0; j < n; j++)
                Q[i][j] = (i == j) ? 1 : 0;
for (i = 0; i < n; i++)
        memcpy(R[i], A[i], n * sizeof(double));

/* Za svaki elemenat ispod glavne dijagonale tekuce
 * matrice R... */
for (k = 1; k < n; k++) {
        double          sin_fi,
                        cos_fi; /* Sinus i kosinus ugla
                                 * rotacije. */
        double          mul;    /* Pomocna vrednost za
                                 * izracunavanje gornjih
                                 * trigonometrijskih
                                 * velicina. */

        /* ...proverava se da li je element vec jednak
         * 0... */
        if (R[k][k - 1] == 0)
                continue;

        /* ...i ako nije, izracunavaju se sinus i kosinus
         * ugla rotacije za trasnformaciju matricom
         * rotacije kojom ce elemenat u (k-1)-voj vrsti
         * ispod glavne dijagonale biti anuliran,... */
        mul =
            1 / sqrt(R[k][k - 1] * R[k][k - 1] +
                     R[k - 1][k - 1] * R[k - 1][k - 1]);
        sin_fi = -R[k][k - 1] * mul;
        cos_fi = R[k - 1][k - 1] * mul;

        /* ...mnozi se tekuca matrica R sa matricom
         * rotacije da bi se anulirao zeljeni elemenat,...
         */
        for (j = 0; j < n; j++) {
                double          r0,
                                r1;
                /* Prethodne vrednosti elemenata
                        R[k-1][j] i R[k][j]
                   (mnozenjem koje implementira ova petlja
                        menjanju se  samo redovi matrice R sa
                        indeksima k-1 i k). */

                r0 = R[k - 1][j], r1 = R[k][j];
                R[k - 1][j] = r0 * cos_fi - r1 * sin_fi;
                R[k][j] = r0 * sin_fi + r1 * cos_fi;
        }

        /* ...a istovremeno se akumulira matrica rotacije
         * U u unitarnu matricu Q. */
        for (i = 0; i < n; i++) {
                double          q0,
                                q1;
                /* Prethodne vrednosti elemenata
                        Q[i][k-1] i R[i][k]
                   (mnozenjem koje implementira ova petlja
                        menjanju se  samo kolone matrice Q sa
```

```
                                        indeksima k-1 i k). */

                                q0 = Q[i][k - 1], q1 = Q[i][k];
                                Q[i][k - 1] = q0 * cos_fi - q1 * sin_fi;
                                Q[i][k] = q0 * sin_fi + q1 * cos_fi;
                        }
                }

                /* Nova vrednost za matricu A se izracunava tako sto se
                 * permutuju unitarna i gornja trougaona matrica u
                 * dekompoziciji. */
                for (i = 0; i < n; i++)
                        for (j = 0; j < n; j++) {
                                A[i][j] = 0;
                                for (k = i; k < n; k++)
                                        A[i][j] += R[i][k] * Q[k][j];
                        }

                /* Izracunava se donja dijagonalna norma tekuce matrice, i
                 * ako je odnos normi tekuce i polazne matrice manji od
                 * zadatog broja, iterativni postupak se prekida. */
                nu_curr = 0;
                for (j = 0; j < n - 1; j++)
                        for (i = j + 1; i < n; i++)
                                nu_curr += A[i][j] * A[i][j];
                nu_curr = sqrt(nu_curr);
                if (nu_curr / nu < epsilon)
                        break;
        }

        /* Oslobadja se memorija zauzeta matricama Q i R. */
        for (i = 0; i < n; i++)
                free(Q[i]);
        free(Q);
        for (i = 0; i < n; i++)
                free(R[i]);
        free(R);
}
```

6.19 *Neka je A realna simetrična matrica dimenzije $m \times m$ i k realan broj. Dalje, neka je Q ortogonalna matrica takva da je $Q^\top(A - kI) = R$, gde je R gornje trougaona a I jedinična matrica.*

Ako je $B = RQ+kI$, dokazati da je B simetrična matrica koja ima iste sopstvene vrednosti kao i matrica A.

Ako je A trodijagonalna, dokazati da se $Q^\top$ može odrediti kao proizvod $m-1$ ortogonalnih matrica tako da je B takođe trodijagonalna matrica.

Rešenje: Kako je $Q\,Q^\top(A-kI) = QR$, to je $A = QR+kI$, pa zbog pretpostavke o simetričnosti matrice A i ortogonalnosti matrice Q

$$QR + kI = (QR + kI)^\top = R^\top Q^\top + kI \qquad \text{sledi} \qquad R^\top = QRQ.$$

Simetričnost matrice B sledi iz

$$B^\top = (RQ + kI)^\top = Q^\top R^\top + kI = Q^\top (QRQ) + kI = RQ + kI = B.$$

Identičnost sopstvenih vrednosti matrica A i B sledi iz njihove sličnosti

$$B = RQ + kI = (R + kQ^\top)Q = (Q^\top QR + kQ^\top)Q = Q^\top (QR + kI)Q = Q^\top AQ.$$

Ako je matrica A trodijagonalna, takva je i matrica $A - kI$. $(m-1)$ nenula element ispod glavne dijagonale se može svesti na nulu Givensovim matricama rotacije – element $a_{i,i-1}$ se anulira pomoću ortogonalne matrice $U_{i,i-1}$ koja se razlikuje od jedinične matrice samo u elementima $u_{i-1,i-1}$, $u_{i-1,i}, u_{i,i-1}$ i $u_{i,i}$. Posle $(m-1)$ ovakvih transformacija dobija se trougaona matrica R. Proizvod matrica $U_{i,i-1}$, $i = m, \ldots, 2$, je gore pomenuta ortogonalna matrica $Q^\top$. Matrica $B - kI$ se dobija množenjem matrice R matricom Q sa desne strane, tj. množenjem ove matrice redom matricama $U_{i,i-1}^\top, i = 2, \ldots, m$, koje su iste strukture kao i matrice $U_{i,i-1}$. Stoga je proizvod matrica koja ima samo jednu poddijagonalu nenula elemenata, tj. zbog dokazane simetričnosti ove matrice, trodijagonalna matrica. Takva je onda i matrica B.

6.20 *LR metodom sa tačnošću* 10^{-4} *odrediti sve sopstvene vrednosti i vektore matrice*

$$A = \begin{pmatrix} 1 & 1.5 & 2.5 & 3.5 \\ 1.5 & 1 & 2 & 1.6 \\ 2.5 & 2 & 1 & 1.7 \\ 3.5 & 1.6 & 1.7 & 1 \end{pmatrix}$$

Rešenje: U osmom koraku dobija se da je sa traženom tačnošću

$$A^{(8)} = \begin{pmatrix} 7.50359 & 18.32529 & 5.11433 & 3.50000 \\ 0 & -2.67265 & -0.50672 & -0.93336 \\ 0 & 0 & -0.96673 & 0.78531 \\ 0 & 0 & 0 & 0.13579 \end{pmatrix}$$

Dakle, sopstvene vrednosti i vektori matrice A su

$$\lambda_1 = 7.5036 \qquad \lambda_2 = -2.6727 \qquad \lambda_3 = -0.9667 \qquad \lambda_4 = 0.1358$$

$$\mathbf{x}_1 = \begin{pmatrix} 0.5651 \\ 0.4090 \\ 0.4816 \\ 0.5306 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} -0.7262 \\ -0.1224 \\ 0.2748 \\ 0.6182 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 0.0785 \\ -0.5438 \\ 0.7576 \\ -0.3522 \end{pmatrix} \quad \mathbf{x}_4 = \begin{pmatrix} -0.3837 \\ 0.7225 \\ 0.3442 \\ -0.4607 \end{pmatrix}$$

MATLAB

QR dekompozicija matrice:	`[Q, R] = qr(A)`
Sopstvene vrednosti:	`[V, D] = eig(A)`

Funkcija `[Q, R] = qr(A)` vrši QR dekompoziciju matrice `A`, vraćajući unitarnu matricu `Q` i gornje-trougaonu matricu `R`.

Sopstvene vrednosti i sopstvene vektore matrice je moguće pronaći korišćenjem funkcije `eig`. Poziv `x = eig(A)` pronalazi sve sopstvene vrednosti matrice A, dok poziv `[P, D] = eig(A)` pronalazi dijagonalnu matricu sopstvenih vrednosti D i matricu sopstvenih vektora P tako da je $P^{-1}AP = D$.

6.21 *Korišćenjem* MATLAB-a *pronaći sopstvene vrednosti matrice dimenzije 10 sa elementima*

$$a_{ij} = \begin{cases} 1 & i \neq j \\ 2 & i = j \end{cases}$$

i to korišćenjem ugrađene funkcije `eig`*, odnosno traženjem nula karakterističnog polinoma.*

Rešenje: Napravimo opisanu matricu i izračunajmo njene sopstvene vrednosti korišćenjem ugrađene funkcije `eig`

```
A=ones(10)+eye(10)
eig(A)
```

Matrica ima devetostruku sopstvenu vrednost 1 i jednostruku sopstvenu vrednost 11. Konstruišimo njen karakteristični polinom korišćenjem `p = poly(A)`.

```
p =
  1.0e+003 *
  Columns 1 through 4
   0.00100000000  -0.02000000000   0.13500000000  -0.48000000000
  Columns 5 through 8
   1.05000000000  -1.51200000000   1.47000000000  -0.96000000000
  Columns 9 through 11
   0.40500000000  -0.10000000000   0.01100000000

roots(p)

10.99999999999999
  1.03673166116602 + 0.01359121147890i
  1.03673166116602 - 0.01359121147890i
  1.01899325911164 + 0.03394583940530i
  1.01899325911164 - 0.03394583940530i
  0.99273598028206 + 0.03780092716992i
  0.99273598028206 - 0.03780092716992i
  0.97053973583786 + 0.02422249368624i
  0.97053973583786 - 0.02422249368624i
  0.96199872720484
```

Primetimo da bi koeficijenti karakterističnog polinoma morali da budu celi. Moglo bi se pomisliti da je ovo uzrok nastale greške. Međutim, čak i kada se koeficijenti karakterističnog polinoma odrede tačno, dobija se da su nule polinoma

```
roots(round(p))
```

```
10.99999999999999
  1.03525664668637
  1.02658185497208 + 0.02290758087942i
  1.02658185497208 - 0.02290758087942i
  1.00532458022745 + 0.03425776020518i
  1.00532458022745 - 0.03425776020518i
  0.98237400833111 + 0.02929227029256i
  0.98237400833111 - 0.02929227029256i
  0.96809123312619 + 0.01135421626058i
  0.96809123312619 - 0.01135421626058i
```

Primetimo da zbog loše uslovljenosti ovog polinoma jako mala promena njegovih koeficijenata uzrokuje prilično veliku promenu njegovih nula.

6.22 *Korišćenjem* MATLAB-*a uraditi 25 koraka QR metode za matricu*

$$\begin{pmatrix} 4.1 & -3.4 & 3.2 & -1.3 & 4.6 \\ 1.0 & 3.3 & 1.1 & -3.4 & -3.6 \\ -1.7 & 4.6 & 2.0 & 3.4 & 3.7 \\ -0.2 & 1.0 & -4.1 & 3.4 & 2.7 \\ 1.0 & -4.8 & -0.8 & -0.5 & -0.6 \end{pmatrix}$$

i na osnovu ovoga proceniti njene sopstvene vrednosti.

Rešenje:

```
A= [4.1   -3.4    3.2   -1.3    4.6; ...
    1.0    3.3    1.1   -3.4   -3.6; ...
   -1.7    4.6    2.0    3.4    3.7; ...
   -0.2    1.0   -4.1    3.4    2.7; ...
    1.0   -4.8   -0.8   -0.5   -0.6];
A = hess(A);
for i = 1 : 25
    [Q, R] = qr(A);
    A = R*Q;
end
```

Nakon izvršenja QR metode matrica A ima oblik

```
A =

    7.2696   -4.1578    0.0993    1.7827   -1.0945
    0.0001    0.4517    3.1315   -1.8901    2.6343
         0   -5.5063    2.3191   -4.2988    3.9244
         0         0   -0.0068    3.3936    6.0589
         0         0         0   -0.0000   -1.2340
```

Tri sopstvene vrednosti se direktno mogu očitati sa dijagonale, dok se preostale dve pronalaze kao nule karakterističnog polinoma blok matrice

$$\begin{pmatrix} 0.4517 & 3.1315 \\ -5.5063 & 2.3191 \end{pmatrix}$$

```
>> eigs = [A(1,1) ...
          (A(2,2)+A(3,3)+sqrt((A(2,2)-A(3,3))^2+4*A(2,3)*A(3,2)))/2 ...
          (A(2,2)+A(3,3)-sqrt((A(2,2)-A(3,3))^2+4*A(2,3)*A(3,2)))/2 ...
          A(4,4) ...
          A(5,5)]'
eigs =
   7.2696
   1.3854 - 4.0461i
   1.3854 + 4.0461i
   3.3936
  -1.2340
```

Matrice rotacije kao i Householder-ove matrice se mogu koristiti za određivanje QR dekompozicije date matrice. QR dekomozicija matrice ima široku oblast primene, od QR metode za nalaženja sopstvenih vrednosti matrice, pa sve do rešavanja preodređenih sistema linearnih jednačina, što će biti ilustrovano kroz naredni zadatkom. Za dekompoziciju matrica u Hessenberg-ovoj formi efikasnije je koristiti matrice rotacije (što je i prikazano u okviru C implementacije QR metode) dok je za dekompoziciju punih matrica efikasnije koristiti Householder-ove matrice.

6.23 *a) Napisati* MATLAB *skript koji vrši QR dekompoziciju realne, kvadratne, matrice A koristeći Householder-ove matrice.*
b) Napisati MATLAB *skript koji korišćenjem QR dekompozicije matrice sistema, rešava sistem jednačina* $A\mathbf{x} = \mathbf{b}$.

Rešenje: a) Iskoristimo Householder-ove matrice H kako bismo anulirali elemente ispod glavne dijagonale jedne po jedne kolone matrice R. Householder-ove matrice se određuju tako da anuliraju sve, osim prvog, elementa datog vektora. Prilikom anuliranja elemenata kolona matrice R, potrebno je prilagoditi Householder-ove matrice tako da anuliraju sve osim prvih nekoliko elemenata datih vektora. Ovo se radi korišćenjem matrica T koje se dobijaju umetanjem Householder-ovih blokova H u jedinične matrice.

Matrica R se sa leve strane množi matricom T tako što se matricom T pomnoži posebno svaka njena kolona. Množenje vektora kolone $\mathbf{r}$ blokom T se vrši na sledeći način:

$$T\mathbf{r} = \begin{pmatrix} I & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \mathbf{r}^{(1)} \\ \mathbf{r}^{(2)} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^{(1)} \\ H\mathbf{r}^{(2)} \end{pmatrix}$$

Ovo znači da početni elementi svake kolone ostaju nepromenjeni, dok se ostali dobijaju množenjem odgovarajućeg dela vektora originalnom Householder-ovom matricom H. Podsetimo se da nije potrebno eksplicitno konstruisati matrice H kako bi se njima množili dati vektori.

Prilikom množenja matrice Q matricom T sa desne strane, vrši se množenje svake njene vrste pojedinačno. Ovo množenje se izvodi analogno prethodnom.

```
% Inicijalizacija
R = A;
Q = eye(n);
% Anuliraju se elementi i-te vrste matrice A
for i = 1:n-1
```

```
    % Odredjuju se parametri Householderove matrice
    sigma = norm(R(i:n,i));
    beta = 1/(sigma*(sigma+R(i,i)));
    u = R(i:n,i);  u(1) = u(1) + sigma;

    % Matrica R se sa leve strane mnozi Householderovim blokom
    % umetnutim u jedinicnu matricu
    % Mnozenje se vrsi kolonu po kolonu matrice R
    % Prvih i-1 elementa svake kolone ostaju nepromenjeni
    for j = 1:n
        R(i:n, j) = R(i:n, j) - beta*u'*R(i:n,j)*u;
    end

    % Matrica Q se sa desne strane mnozi pomenutim blokom
    % Mnozenje se vrsi vrstu po vrstu matrice Q
    % Prvih i-1 elementa svake vrste ostaju nepromenjeni
    for j = 1:n
            Q(j, i:n) = Q(j, i:n) - beta*Q(j,i:n)*u*u';
    end
end
```

b) Ukoliko je poznata QR dekompozicija matrice A, sistem $A\mathbf{x} = \mathbf{b}$ se svodi na $QR\mathbf{x} = \mathbf{b}$, odnosno na trougaoni sistem $R\mathbf{x} = Q^{-1}\mathbf{b}$. Zbog ortogonalnosti matrice Q, $R\mathbf{x} = Q^T\mathbf{b}$, tako da njeno invertovanje nije potrebno. Konačno rešenje sistema se dobija rešavanjem trougaonog sistema, što se jednostavno radi metodom povratne zamene.

Prilikom implementacije rešavanja sistema, matrice QR dekompozicije nije neophodno eksplicitno konstruisati. Ukoliko se QR dekompozicija vrši Householder-ovim matricama, važi da je $H_{n-1} \dots H_1 A = R$, tj. $H_{n-1} \cdot \ldots \cdot H_1 A\mathbf{x} = R\mathbf{b}$. Odavde je $(H_{n-1} \cdot \ldots H_1)^{-1} R\mathbf{x} = \mathbf{b}$, tako da je $R\mathbf{x} = H_{n-1} \cdot \ldots \cdot H_1\mathbf{b}$. Funkcija koja vrši rešavanje sistema se implementira tako da u svakom koraku određuje odgovarajuću Householder-ovu matricu i njome, sa leve strane, množi i tekuću matricu sistema i vektor slobodnih članova. Na posletku, matrica sistema postaje trougaona i poslednji korak je rešavanje dobijenog trougaonog sistema povratnom zamenom.

```
% Inicijalizacija
R = A;
% Anuliraju se elementi pojedinacnih kolona matrice R
for i = 1:n-1
    % Odredjuju se parametri Householderove matrice
    sigma = norm(R(i:n,i));
    beta = 1/(sigma*(sigma+R(i,i)));
    u = R(i:n,i);  u(1) = u(1) + sigma;

    % Matrica A se sa leve strane mnozi Householderovim blokom
    for j = 1:n
        R(i:n, j) = R(i:n, j) - beta*u'*R(i:n,j)*u;
    end

    % Vektor slobodnih clanova se sa leve strane mnozi
    % Householderovim blokom
    b(i:n) = b(i:n) - beta*u'*b(i:n)*u;
end
```

```
% Resenja sistema se odredjuju povratnom zamenom
x(n) = b(n)/R(n,n);
for i = n-1:-1:1
   x(i) = (b(i)-R(i,i+1:n)*x(i+1:n))/R(i,i);
end
```

Napomenimo da se ova tehnika može koristiti i za rešavanje preodređenih sistema.

Naredni zadatak pokazuje značaj gornje-Hessenberg-ove forme matrice pri implementaciji QR metode.

6.24 *a) Dokazati da se u QR dekompoziciji matrice koja je u gornje-Hessenberg-ovoj formi dobija unitarna matrica Q koja je takođe u gornje-Hessenberg-ovoj formi.*
b) Dokazati da se množenjem gornje trougaone matrice R i gornje-Hessenberg-ove matrice Q, dobija matrica koja je, takođe, gornje-Hessenberg-ova.
c) Imajući prethodne rezultate u vidu, napisati MATLAB *skript koji uz minimalni broj množenja brojeva množi trougaonu i gornje-Hessenberg-ovu matricu.*

Rešenje: a) Neka je kvadratna matrica A dimenzije n u gornje-Hessenberg-ovoj formi. Množenjem matrice A sa leve strane matricama rotacije $G_{i+1,i}(\varphi_i, \psi_i)$, $i = 1, \ldots, n-1$, za odgovarajuće određene uglove φ_i i ψ_i dobija se gornje trougaona matrica R tj.

$$G_{n,n-1}(\varphi_{n-1}, \psi_{n-1}) \cdot \ldots \cdot G_{2,1}(\varphi_1, \psi_1)A = R$$

tada je

$$A = (G_{n,n-1}(\varphi_{n-1}, \psi_{n-1}) \cdot \ldots \cdot G_{2,1}(\varphi_1, \psi_1))^{-1}R$$

odnosno, imajući u vidu unitarnost matrica rotacije,

$$\begin{aligned} Q &= G^*_{2,1}(\varphi_1, \psi_1) \cdot \ldots \cdot G^*_{n,n-1}(\varphi_{n-1}, \psi_{n-1}) = \\ &= G_{2,1}(-\varphi_1, \psi_1) \cdot \ldots \cdot G_{n,n-1}(-\varphi_{n-1}, \psi_{n-1}) \end{aligned}$$

Svaka matrica u ovom proizvodu dodaje tačno jedan element na glavnu subdijagonalu matrice Q. Preciznije, matrice

$$M_k = G_{2,1}(-\varphi_1, \psi_1) \cdot \ldots \cdot G_{k+1,k}(-\varphi_k, \psi_k), \quad k = 1, \ldots, n-1$$

imaju svojstvo da su matrice koje su u gornje-Hessenberg-ovoj formi, pri čemu su, dodatno, pojedini elementi na glavnoj subdijagonali nula tj. $m^k_{i+1,i} = 0, \quad i = k+1, \ldots, n$.

Zaista, M_1 je matrica rotacije i ona zadovoljava pomenuto svojstvo.

Ukoliko pretpostavimo da matrica M_k ima pomenuto svojstvo, tada to svojstvo ima i matrica $M_{k+1} = M_k G_{k+2,k+1}(-\varphi_{k+1}, \psi_{k+1})$. Množenje matrice M_k matricom rotacije $G_{k+2,k+1}(-\varphi_{k+1}, \psi_{k+1})$ sa desna menja isključivo kolone $k+1$ i $k+2$ tj.

$$m^{k+1}_{i,j} = m^k_{i,j}, \quad j \neq k+1, k+2$$

Dalje je:

$$m^{k+1}_{i,k+1} = m^k_{i,k+1} \cos\varphi_{k+1} + m^k_{i,k+2} e^{i\psi_{k+1}} \sin\varphi_{k+1}$$

dok je

$$m^{k+1}_{i,k+2} = -m^k_{i,k+1} e^{i\psi_{k+1}} \sin\varphi_{k+1} + m^k_{i,k+2} \cos\varphi_{k+1}.$$

Za $i > k+2$, $m^k_{i,k+1} = m^k_{i,k+2} = 0$ pa je $m^{k+1}_{i,k+1} = 0$ i $m^{k+1}_{i,k+2} = 0$, što pokazuje da i matrica M_{k+1} ima pomenuto svojstvo.

Pošto je $Q = M_{n-1}$, Q je gornje-Hessenberg-ova matrica.

b) Neka je $A = RQ$, pri čemu je $r_{m,n} = 0$ za $m > n$ i $r_{m,n} = 0$ za $m > n+1$, i

$$a_{i,j} = \sum_{k=1}^{n} r_{i,k} q_{k,j}, \quad i,j = 1,\ldots,n$$

.

Ukoliko je $i > 1$ grupišimo sabirke na sledeći način:

$$a_{i,j} = \sum_{k=1}^{i-1} r_{i,k} q_{k,j} + \sum_{k=i}^{n} r_{i,k} q_{k,j}$$

Pošto je u prvoj sumi $i > k$, $r_{i,k} = 0$ pa je i cela suma 0, odnosno

$$a_{i,j} = \sum_{k=i}^{n} r_{i,k} q_{k,j}$$

I u slučaju $i = 1$, ova relacija trivijalno važi.

Ukoliko je $i > j+1$ tada je $q_{k,j} = 0$ za $k = i,\ldots,n$ pa je i cela suma 0 što dokazuje da je A gornje-Hessenberg-ova.

Razmatrajmo slučaj u kome je $i \leq j+1$ i, naravno, $1 \leq j \leq n$. U ovom slučaju je $j \geq \max(i-1, 1)$.

Ukoliko je $j < n$, grupišimo sabirke na sledeći način:

$$a_{i,j} = \sum_{k=i}^{n} r_{i,k} q_{k,j} = \sum_{k=i}^{j+1} r_{i,k} q_{k,j} + \sum_{k=j+2}^{n} r_{i,k} q_{k,j}$$

U drugoj sumi je $k > j+1$ je pa je $q_{k,j} = 0$ tako da je i cela suma 0, odnosno važi

$$a_{i,j} = \sum_{k=i}^{j+1} r_{i,k} q_{k,j}$$

Ukoliko je $j = n$, tada je

$$a_{i,j} = \sum_{k=i}^{n} r_{i,k} q_{k,n}.$$

Oba slučaja se mogu objediniti kroz

$$a_{i,j} = \sum_{k=i}^{\min(j+1,n)} r_{i,k} q_{k,n}$$

c)

```
%Mnozenje gornje trougaone matrice R i Hessenbergove matrice Q

A = zeros(n);
for i = 1:n
    for j = max(i-1,1) : n
        for k = i : min(j+1, n)
            A(i,j) = A(i,j) + R(i,k)*Q(k,j);
        end
    end
end
```

Naglasimo da zbog interpreterske prirode MATLAB-a i postojanja petlji, prethodni kod može da radi čak i neefikasnije od ugrađenog operatora množenja matrica `A=R*Q;`. Ukoliko bi se, pak, implementacija vršila u nekom od uobičajenih programskih jezika (C, FORTRAN, ...), množenje bi trebalo realizovati na opisani način.

Naglasimo još i da prilikom implementacije QR metode, nakon prevođenja polazne matrice A u Hessenberg-ovu formu, nije potrebno eksplicitno konstruisati matrice Q u dekompoziciji. Dovoljno je pamtiti samo niz odgovarajućih matrica rotacije (odnosno samo uglova φ_i i ψ_i) kojima se množenje matrica A_i vrši. Zaista, ukoliko je

$$G_{n-1,n}(\varphi_{n-1}, \psi_{n-1}) \cdot \ldots \cdot G_{2,1}(\varphi_1, \psi_1) A = R$$

tada je

$$RQ = RG^*_{1,2}(\varphi_1, \psi_1) \cdot \ldots \cdot G^*_{n-1,n}(\varphi_{n-1}, \psi_{n-1})$$

tako da se sva matrična množenja potrebna za implementaciju QR metode svode na množenja matricama rotacije koja se efikasno implementiraju.

6.25 *Imajući na raspolaganju funkciju* `eig` *koja računa sopstvene vrednosti date matrice, implementirati funkciju koja pronalazi nule datog polinoma.*

Rešenje: Ukoliko su poznati koeficijenti moničnog polinoma $p = x^n + a_1 x^{n-1} + \ldots + a_n$, moguće je pronaći matricu kojoj je on karakteristični polinom. Jedna od takvih matrica je i Frobenius-ova matrica

$$A = \begin{pmatrix} -a_1 & -a_2 & \ldots & -a_{n-1} & -a_n \\ 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ \ldots & & & & \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}$$

Sopstvene vrednosti matrice A su nule polinoma p.

```
% Funkcija pronalazi nule datog polinoma p
function x = roots(p)

% Normalizacija polinoma na monicni oblik
p = p/p(1);

% Kreiranje matrice koja za sopstvene vrednosti ima nule polinoma p
A = diag(repmat(1,1,length(p)-2),-1);
A(1, :) = -p(2:end)
```

```
% Pronalazenje sopstvenih vrednosti
x = eig(A);
```

6.5 Delimičan problem

Metoda proizvoljnog vektora – Aproksimacija najveće po modulu sopstvene vrednosti matrice A je za dovoljno veliki prirodan broj n

$$\lambda_1 \approx v_k^{(n)}/v_k^{(n-1)}, \qquad \text{za neko } \ 0 \le k \le m, \quad n = 1, 2, \ldots,$$

$$\mathbf{v}^{(n)} = (v_1^{(n)}, \ldots, v_m^{(n)})^\top = A\mathbf{v}^{(n-1)} = A^n\mathbf{v}^{(0)},$$

gde je $\mathbf{v}^{(0)}$ proizvoljan vektor. Sopstveni vektor je $\mathbf{x}_1 \approx c\mathbf{v}^{(n)}$, gde je c proizvoljna konstanta.

Metoda skalarnog proizvoda – Aproksimacija najveće po veličini modula sopstvene vrednosti za dovoljno veliki prirodan broj n je

$$\lambda_1 \approx \frac{\left(\mathbf{v}^{(n)}, \mathbf{w}^{(n)}\right)}{\left(\mathbf{v}^{(n-1)}, \mathbf{w}^{(n)}\right)},$$

gde je

$$\mathbf{v}^{(n)} = A^n\mathbf{v}^{(0)} = A\mathbf{v}^{(n-1)}, \quad \mathbf{w}^{(n)} = (A^*)^n\mathbf{w}^{(0)} = A^*\mathbf{w}^{(n-1)},$$

a $\mathbf{v}^{(0)}$ i $\mathbf{w}^{(0)}$ su proizvoljni vektori. Sopstveni vektor je $\mathbf{x}_1 \approx c\mathbf{v}^{(n)}$, gde je c proizvoljna konstanta.

Metoda tragova – Trag matrice $\mathrm{tr}(A)$ je zbir njenih dijagonalnih elemenata. Najveća po veličini modula sopstvena vrednost λ_1 matrice A se metodom tragova može odrediti kao granična vrednost

$$|\lambda_1| = \lim_{n\to\infty} \sqrt[n]{|tr(A^n)|} \qquad \text{ili} \qquad \lambda_1 = \lim_{n\to\infty} \frac{\mathrm{tr}(A^{n+1})}{\mathrm{tr}(A^n)}$$

Aproksimacija sopstvenog vektora $\mathbf{x}_1$ je za dovoljno veliko n vektor $A^n\mathbf{v}$, gde je $\mathbf{v}$ proizvoljan vektor.

Metodom iscrpljivanja se nalazi druga po veličini modula sopstvena vrednost λ_2 matrice A ako su poznati najveća po veličini modula sopstvena vrednost λ_1 i njoj odgovarajući sopstveni vektor $\mathbf{x}_1$ matrice A i sopstveni vektor $\mathbf{y}_1$ matrice A^* koji odgovara sopstvenoj vrednosti $\overline{\lambda}_1$. Ako je $(\mathbf{x}_1, \mathbf{y}_1) = 1$, sopstvena vrednost λ_2 i njoj odgovarajući sopstveni vektor $\mathbf{x}_2$ su najveća po veličini modula sopstvena vrednost i njoj odgovarajući sopstveni vektor matrice

$$A_1 = A - \lambda_1\mathbf{x}_1\,\mathbf{y}_1^*$$

6.26 *Metodom proizvoljnog vektora sa tačnošću* 10^{-3} *odrediti najveću po modulu sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Rešenje: Ako se pođe od vektora $\mathbf{v}^{(0)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$, izračunavanje se može predstaviti sledećom tabelom:

k	$v_1^{(k)}$	$v_2^{(k)}$	$v_3^{(k)}$	$v_1^{(k)}/v_1^{(k-1)}$	$v_2^{(k)}/v_2^{(k-1)}$	$v_3^{(k)}/v_3^{(k-1)}$
0	1	0	0			
1	2	1	1			
2	6	5	6	3	5	6
3	23	22	29	3.833	4.400	4.833
4	97	96	132	4.217	4.364	4.552
5	422	421	589	4.351	4.385	4.462
6	1854	1853	2610	4.393	4.401	4.431
7	8171	8170	11537	4.407	4.409	4.420
8	36049	36048	50952	4.412	4.412	4.416
9	159098	159097	224953	4.413	4.413	4.415
10	702246	702245	993054	4.414	4.414	4.414

Najveća po modulu sopstvena vrednost date matrice i njoj odgovarajuí sopstveni vektor su

$$\lambda_1 = 4.414, \qquad \mathbf{x}_1 = \begin{pmatrix} 0.5 & 0.5 & 0.707 \end{pmatrix}^\top.$$

Sopstveni vektor je određen normiranjem vektora $vkv^{(10)}$. Treba napomenuti da je kod ove i srodnih metoda dobro s vremena na vreme normirati tekući vektor $\mathbf{v}^{(k)}$ da ne bi došlo do prekoračenja.

6.27 *Metodom proizvoljnog vektora sa tačnošću* $5 \cdot 10^{-5}$ *odrediti najveću po modulu sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

ako je $\mathbf{v}^{(0)} = \begin{pmatrix} 0.4 & -0.6 & 0.6 & -0.4 \end{pmatrix}^\top$.

Rešenje: Kako je matrica simetrična i $v_1^{(0)} = -v_4^{(0)}$ i $v_2^{(0)} = -v_3^{(0)}$, biće $v_1^{(n)} = -v_4^{(n)}$ i $v_2^{(n)} = -v_3^{(n)}$ i stoga $v_1^{(n)}/v_1^{(n-1)} = v_4^{(n)}/v_4^{(n-1)}$ i $v_2^{(n)}/v_2^{(n-1)} = v_3^{(n)}/v_3^{(n-1)}$. Zato je dovoljno računati samo prve dve koordinate tekućih vektora. Polazeći od

vektora $\mathbf{v}^{(0)} = \begin{pmatrix} 0.4 & -0.6 & 0.6 & -0.4 \end{pmatrix}^\top$ u dvanaestoj iteraciji dobijamo da je sa traženom tačnošću

$$\lambda_1 = 3.6180, \qquad \mathbf{x}_1 = \begin{pmatrix} 0.3718 & -0.6015 & 0.6015 & -0.3718 \end{pmatrix}^\top .$$

6.28 *Metodom proizvoljnog vektora sa tačnošću* $5 \cdot 10^{-5}$ *odrediti najveću po modulu sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A = \begin{pmatrix} 4 & 0 & 3 \\ 2 & 2 & 3 \\ 1 & 3 & 4 \end{pmatrix}$$

Rešenje: Polazeći od vektora $\mathbf{v}^{(0)} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\top$ u šestoj iteraciji se dobija

$$\lambda_1 = 7.4641, \qquad \mathbf{x}_1 = \begin{pmatrix} 0.5477 & 0.5477 & 0.6325 \end{pmatrix}^\top .$$

6.29 *a) Ako je* λ *sopstvena vrednost regularne matrice* A*, dokazati da je* $1/\lambda$ *sopstvena vrednost matrice* A^{-1}*.*
b) Metodom proizvoljnog vektora odrediti sa tačnošću $5 \cdot 10^{-4}$ *po modulu najveću i najmanju sopstvenu vrednost i njima odgovarajuće sopstvene vektore matrice*

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

Rešenje: a) Iz definicije sopstvene vrednosti λ i sopstvenog vektora $\mathbf{x}$ regularne matrice A

$$\frac{1}{\lambda} A^{-1} \,/ A\mathbf{x} = \lambda \mathbf{x} \qquad \text{sledi} \qquad \frac{1}{\lambda}\mathbf{x} = A^{-1}\mathbf{x},$$

što znači da je $1/\lambda$ sopstvena vrednost matrice A^{-1}, a njoj odgovarajući sopstveni vektor je $\mathbf{x}$.
b) S obzirom da je najmanja po modulu sopstvena vrednost matrice A recipročna vrednost najveće po modulu sopstvene vrednosti matrice A^{-1}, odredićemo je upravo nalaženjem najveće po modulu sopstvene vrednosti matrice A^{-1},

$$A^{-1} = \begin{pmatrix} 0.75 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.75 \end{pmatrix}$$

metodom proizvoljnog vektora. Polazeći od vektora $\mathbf{v} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\top$ u devetoj iteraciji se dobija da su najveća po modulu sopstvena vrednost i odgovarajući sopstveni vektor matrice A

$$\lambda_1 = 3.414, \qquad \mathbf{x}_1 \approx c\, A^9 \mathbf{v} = \begin{pmatrix} 0.5 & -0.707 & 0.5 \end{pmatrix}^\top ,$$

gde je c faktor normiranja. Na isti način, primenom metode proizvoljnog vektora na matricu A^{-1} sa istim početnim vektorom dobija se da je njena najveća po modulu sopstvena vrednost 1.707. Stoga je najmanja po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor matrice A

$$\lambda_3 = \frac{1}{1.707} = 0.586, \qquad \mathbf{x}_3 \approx c\left(A^{-1}\right)^5 \mathbf{v} = \begin{pmatrix} 0.5 & 0.707 & 0.5 \end{pmatrix}^\top.$$

6.30 *Metodom skalarnog proizvoda sa* 4 *sigurne cifre odrediti najveću po modulu sopstvenu vrednost i njoj odgovarajući sopstveni vektor matrice:*

$$A = \begin{pmatrix} 2.4 & 0.8 & 3.3 \\ 0.8 & 1.4 & 1.7 \\ 3.3 & 1.7 & 0.6 \end{pmatrix}$$

Rešenje: U slučaju kada je $A = A^*$, kao u ovom zadatku, radi lakšeg računanja pogodno je uzeti da je $\mathbf{v}^{(0)} = \mathbf{w}^{(0)}$. Polazeí od vektora $\mathbf{v}^{(0)} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\top$, dobijamo sledeće aproksimacije

k	$v_1^{(k)}$	$v_2^{(k)}$	$v_3^{(k)}$	$(\mathbf{v}^{(k)}, \mathbf{v}^{(k)})$	$(\mathbf{v}^{(k-1)}, \mathbf{v}^{(k)})$	λ_1
0	1	1	1			
1	6.5	3.9	5.6	88.82	16	5.5513
2	37.2	19.16	31.44	2739.4	492.59	5.5612
3	209.176	111.460	175.930	87129.3	15561.8	5.5989
4	1171.759	622.466	985.321	2731340	487831	5.5989

Najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor su

$$\lambda_1 = 5.599, \qquad \mathbf{x}_1 = \begin{pmatrix} 0.709 & 0.377 & 0.596 \end{pmatrix}^\top.$$

6.31 *Metodom skalarnog proizvoda naći sa tačnošću* 10^{-3} *najveću po modulu sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A = \begin{pmatrix} 3.56 & 2.75 & 3.04 \\ 2.86 & 3.45 & 3.04 \\ 3.26 & 2.75 & 3.34 \end{pmatrix}$$

Rešenje: Polazeći od vektora $\mathbf{v}^{(0)} = \mathbf{w}^{(0)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$ u trećoj iteraciji dobijamo da je sa traženom tačnošću

$$\lambda_1 = 9.35, \qquad \mathbf{x}_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\top.$$

6.32 *Metodom skalarnog proizvoda odrediti sa tačnošću* $5\cdot10^{-4}$ *prve dve sopstvene vrednosti i njima odgovarajuće sopstvene vektore matrice*

$$A=\begin{pmatrix}4&3&2&1\\3&6&4&2\\2&4&6&3\\1&2&3&4\end{pmatrix}$$

Rešenje: Kako je matrica A realna i simetrična, metodom skalarnog proizvoda je aproksimacija tražene sopstvene vrednosti za dovoljno veliki prirodan broj n određena izrazom

$$\lambda_1\approx\frac{\left(\mathbf{v}^{(n)},\mathbf{v}^{(n)}\right)}{\left(\mathbf{v}^{(n-1)},\mathbf{v}^{(n)}\right)},\qquad\text{gde je}\quad\mathbf{v}^{(n)}=A^n\mathbf{v}^{(0)}=A\mathbf{v}^{(n-1)},$$

U četvrtoj iteraciji polazeći od vektora $\mathbf{v}^{(0)}=\begin{pmatrix}1&1&1&1\end{pmatrix}^\top$ nalazimo da je sa traženom tačnošću najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor

$$\lambda_1=13.0902,\qquad\mathbf{x}_1=\begin{pmatrix}0.3718&0.6015&0.6015&0.3718\end{pmatrix}^\top$$

(rezultat je zapisan sa većim brojem sigurnih cifara, da bi u daljem računu mogla biti postignuta tražena tačnost).

Sledeća po veličini modula sopstvena vrednost i njoj odgovarajući sopstveni vektor matrice A će biti određeni istom metodom, ali kao najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor matrice

$$\tilde{A}=A-\lambda_1\mathbf{x}_1\mathbf{x}_1^\top=\begin{pmatrix}2.1904&0.0727&-0.9273&-0.8096\\0.0727&1.2646&-0.7354&-0.9273\\-0.9273&-0.7354&1.2646&0.0727\\-0.8096&-0.9273&0.0727&2.1904\end{pmatrix}$$

Polazeći od vektora $\mathbf{v}=\begin{pmatrix}1&1&1&1\end{pmatrix}^\top$ u trećoj iteraciji dobija se sledeća po veličini modula sopstvena vrednost matrice A i njoj odgovarajući sopstveni vektor,

$$\lambda_2=1.910,\qquad\mathbf{x}_1=\begin{pmatrix}0.601&-0.372&-0.372&0.601\end{pmatrix}^\top.$$

6.33 *Metodom tragova sa tačnošću* 10^{-3} *odrediti najveću po modulu sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A=\begin{pmatrix}1.12&4.45&0.38\\4.45&1.31&0.56\\0.38&0.56&0.52\end{pmatrix}.$$

Rešenje: Izabrana je varijanta metode tragova u kojoj se moduo najveće po modulu sopstvene vrednosti računa za dovoljno veliki prirodan broj n pomoću izraza

$$|\lambda_1|\approx\sqrt[n]{|\mathrm{tr}(A^n)|},$$

gde je $\mathrm{tr}(A^n)$ zbir dijagonalnih elemenata matrice A^n. Radi ubrzanja konvergencije aproksimacija se racuna uvek na osnovu kvadrata matrice iz prethodnog koraka, tj. za $n = 2^k$, $k = 0, 1, \ldots$. Aproksimacija sopstvenog vektora je određena vektorom $c\,A^n\mathbf{v}$, gde je $\mathbf{v}$ proizvoljni vektor i c proizvoljna konstanta. Znak sopstvene vrednosti se određuje prema tome koja od vrednosti $-|\lambda_1|$ ili $|\lambda_1|$ zamenom u jednačini $A\mathbf{x}_1 = \lambda_1\mathbf{x}_1$ daje manju grešku.

U ovom zadatku tražena tačnost je postignuta za $n = 32$,

$$\lambda_1 = 5.751, \qquad \mathbf{x}_1 = cA^{32}\begin{pmatrix}1 & 0 & 0\end{pmatrix}^\top = \begin{pmatrix}0.693 & 0.710 & 0.126\end{pmatrix}^\top.$$

6.34 *Metodom iscrpljivanja odrediti sa tačnošću 10^{-3} prve dve po veličini modula sopstvene vrednosti i odgovarajuće sopstvene vektore matrice*

$$\begin{pmatrix}8 & 2 & 1\\ 2 & 6 & 2\\ 1 & 2 & 5\end{pmatrix}$$

Rešenje: Metodom skalarnog proizvoda sa početnim vektorom $\mathbf{v} = (1,\,1,\,1)^\top$ određena je najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor

$$\lambda_1 = 10.0000, \qquad \mathbf{x}_1 = \begin{pmatrix}0.7399 & 0.5588 & 0.3745\end{pmatrix}^\top.$$

Kao najveća po modulu sopstvena vrednost i odgovarajući sopstveni vektor matrice

$$\tilde{A} = A - \lambda_1\mathbf{x}_1\mathbf{x}_1^\top = \begin{pmatrix}2.5254 & -2.1348 & -1.7709\\ -2.1348 & 2.8771 & -0.0928\\ -1.7709 & -0.0928 & 3.5975\end{pmatrix}$$

određena je druga po veličini modula sopstvena vrednost sa odgovarajućim sopstvenim vektorom matrice A. Primenjena je ponovo metoda skalarnog proizvoda sa istim početnim vektorom $\mathbf{v}$ i dobijeno

$$\lambda_2 = 5.616, \qquad \mathbf{x}_2 = \begin{pmatrix}-0.660 & 0.482 & 0.576\end{pmatrix}^\top.$$

6.35 *Metodom tragova sa tačnošću 10^{-4} odrediti drugu po veličini modula sopstvenu vrednost i odgovarajući sopstveni vektor matrice*

$$A = \begin{pmatrix}6.00 & 1.34 & 1.45\\ 1.34 & 6.00 & 1.46\\ 1.45 & 1.46 & 6.00\end{pmatrix},$$

ako su najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor

$$\lambda_1 = 8.834015, \qquad \mathbf{x}_1 = \begin{pmatrix}0.571486 & 0.572894 & 0.587534\end{pmatrix}^\top.$$

Rešenje:

$$\lambda_2 = 4.6602, \qquad \mathbf{x}_1 = \begin{pmatrix}0.7226 & -0.6906 & -0.0294\end{pmatrix}^\top.$$

6.36 *Primenom metode iscrpljivanja i metode tragova odrediti drugu po veličini modula sopstvenu vrednost i njoj odgovarajući sopstveni vektor matrice:*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Rešenje: U zadatku 19. su određeni najveća po modulu sopstvena vrednost i njoj odgovarajući sopstveni vektor date matrice

$$\lambda_1 = 4.414 \qquad \mathbf{x}_1 = \begin{pmatrix} 0.5 & 0.5 & 0.707 \end{pmatrix}^\top .$$

Kako je $A = A^*$ i $\|\mathbf{x}_1\|_2 = 1$, to je odgovarajući sopstveni vektor matrice A^* jednak $\mathbf{y}_1 = \mathbf{x}_1$. Matrica kojoj je tražena sopstvena vrednost najveća po modulu je

$$\tilde{A} = A - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^* = \begin{pmatrix} 0.8965 & -0.1035 & -0.5606 \\ -0.1035 & 0.8965 & -0.5606 \\ -0.5606 & -0.5606 & 0.7929 \end{pmatrix}$$

Koristeći varijantu metode tragova izraženu preko korena, dobijaju se sledeće vrednosti

$$\operatorname{tr}(\tilde{A}) = 2.5869 \qquad \sqrt[2]{\operatorname{tr}(\tilde{A}^2)} = 1.8748 \qquad \sqrt[4]{\operatorname{tr}(\tilde{A}^4)} = 1.6451$$
$$\sqrt[8]{\operatorname{tr}(\tilde{A}^8)} = 1.5907 \qquad \sqrt[16]{\operatorname{tr}(\tilde{A}^{16})} = 1.5858 \qquad \sqrt[32]{\operatorname{tr}(\tilde{A}^{32})} = 1.5858$$

Dakle, aproksimacija najveće po veličini modula sopstvene vrednosti matrice $\tilde{A}$, odnosno druge po veličini modula sopstvene vrednosti matrice A, je $|\lambda_2| = 1.5858$. Apoksimacija odgovarajućeg sopstvenog vektora se dobija normiranjem vektora $\tilde{A}^{32}\mathbf{v}$, gde je $\mathbf{v}$ proizvoljni vektor. Ako se uzme da je $\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$, dobija se sopstveni vektor $\mathbf{x}_2 = \begin{pmatrix} 0.5 & 0.5 & -0.707 \end{pmatrix}^\top$. Znak broja λ_2 se uzima onaj koji daje manju vrednost norme vektora $A\mathbf{x}_2 - |\lambda_2|\mathbf{x}_2$ i $A\mathbf{x}_2 + |\lambda_2|\mathbf{x}_2$. U ovom slučaju treba uzeti pozitivnu vrednost, tj.

$$\lambda_2 = 1.5858, \qquad \mathbf{x}_2 = \begin{pmatrix} 0.5 & 0.5 & -0.707 \end{pmatrix}^\top .$$

C Za demonstraciju implementacije metoda za rešavanje delimičnog problema sopstvenih vrednosti odabrana je metoda tragova. Slično većini metoda iz ovog poglavlja, implementacija se sastoji u prostom prevođenju u kod formula koje opisuju metodu. U svakoj iteraciji, posmatrana matrica se kvadrira, čime se ubrzava konvergencija; postupak se ponavlja dok se ne postigne željena tačnost. Treba uočiti kako je na kraju određen znak sopstvene vrednosti - dve moguće vrednosti treba uvrsiti u izraz $\|Ax - \lambda x\|$, gde je x sopstveni vektor. Ona vrednost koja daje manji izraz određuje znak sopstvene vrednosti. Procedura koja implementira metodu tragova ima oblik:

```
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

/*
 * Funkcija track() racuna najvecu po modulu sopstvenu vrednost date
 * matrice metodom tragova. Argumenti funkcije su:
 *   n - dimenzija matrice
 *   A - data matrica
 *   epsilon - zeljena tacnost
 * Funkcija vraca najvecu po modulu sopstvenu vrednost matrice
 * A. Dimenzija matrice i tacnost moraju biti brojevi veci od 0.
 */
double
track(int n, double **A, double epsilon)
{
        double        **Ak,
                      **A2k;    /* Matrice koje predstavljaju tekuci i
                                 * dvostruko veci od tekuceg stepen ulazne
                                 * matrice prilikom izracunavanja. */
        double          prev,
                        curr;   /* Apsolutna vrednost prethodne i tekuce
                                 * iteracije za najvecu po modulu
                                 * sopstvenu vrednost. */
        double          lambda; /* Najveca po modulu sopstvena vrednost
                                 * matrice. */
        int             i,
                        j,
                        k;      /* Brojaci u petljama. */

        /* Proverava se da li su uslovi zadatka ispunjeni. */
        assert(n > 0);
        assert(epsilon > 0);

        /* Alocira se memorija za matrice tekuceg i dvostruko veceg od
         * tekuceg stepena polazne matrice i kopira se u matricu tekuceg
         * stepena polazna matrica. */
        Ak = (double **) malloc(n * sizeof(double *));
        A2k = (double **) malloc(n * sizeof(double *));
        for (i = 0; i < n; i++) {
                Ak[i] = (double *) malloc(n * sizeof(double));
                memcpy(Ak[i], A[i], n * sizeof(double));
                A2k[i] = (double *) malloc(n * sizeof(double));
        }

        /* Pocetna aproksimacija za najvecu po modulu sopstvenu vrednost
         * date matrice jednaka je tragu matrice. */
        curr = 0;
        for (i = 0; i < n; i++)
                curr += Ak[i][i];

        for (k = 2;; k *= 2) {
                double        **B;      /* Pokazivac koji sluzi za
                                         * swap-ovanje matrica. */
                int             l;      /* Indeks koriscen za mnozenje
                                         * matrica. */
```

```
                /* Izracunava se kvadrat tekuce matrice. */
                for (i = 0; i < n; i++)
                        for (j = 0; j < n; j++) {
                                A2k[i][j] = 0;
                                for (l = 0; l < n; l++)
                                        A2k[i][j] += Ak[i][l] * Ak[l][j];
                        }

                /* Kvadrat tekuce matrice postaje tekuca matrica. */
                B = Ak, Ak = A2k, A2k = B;

                /* Izracunava se aproksimacija apsolutne vrednosti najvece
                 * sopstvene vrednosti tekuce matrice. */
                prev = curr;
                curr = 0;
                for (i = 0; i < n; i++)
                        curr += Ak[i][i];
                curr = pow(fabs(curr), 1.0 / k);

                /* Ukoliko je razlika tekuce i prethodne aproksimacije
                 * apsolutne vrednosti najvece sopstvene vrednosti
                 * dostigla zadatu tacnost, iterativni postupak se
                 * prekida. */
                if (fabs(curr - prev) < epsilon)
                        break;
        }

        /* Odredjuje se znak najvece po modulu sopstvene vrednosti. */
        lambda =
            fabs(A[0][0] - curr) < fabs(A[0][0] + curr) ? curr : -curr;

        /* Oslobadja se zauzeta memorija. */
        for (i = 0; i < n; i++) {
                free(Ak[i]);
                free(A2k[i]);
        }
        free(Ak);
        free(A2k);

        return lambda;
}
```

7

Nelinearne jednačine i sistemi

Nalazi se rešenje $\mathbf{x}^*$ nelinearne jednačine (m=1) ili sistema

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{f} : \mathcal{R}^m \to \mathcal{R}^m,$$

gde je $\mathbf{f} = (f_1, \ldots, f_m)^\top$ i $\mathbf{x} = (x_1, \ldots, x_m)^\top$.

Za rešavanje jednodimenzionog problema (u $\mathcal{R}^1$), pored metode iteracije i Newton-ove metode (biće prikazane u narednim odeljcima), koriste se i sledeće metode:

Metoda polovljenja – Polazeći od intervala $[a_0, b_0] = [a, b]$ na kome je lokalizovano rešenje, obrazuje se niz intervala $[a_n, b_n]$, $n = 0, 1, \ldots$, tako što je svaki dobijen polovljenjem prethodnog,

$$[a_0, b_0] \supset [a_1, b_1] \supset \cdots \supset [a_n, b_n] \supset \cdots,$$

$$f(a_n)\, f(b_n) < 0, \qquad b_n - a_n = (b-a)/2^n, \qquad x^* \in [a_n, b_n], \qquad n = 0, 1, \ldots.$$

Aproksimacija rešenja x^* je

$$x^{(n)} = \frac{1}{2}(a_n + b_n), \qquad |x^* - x^{(n)}| \le \frac{1}{2}(b_n - a_n) = \frac{1}{2^{n+1}}\,(b-a).$$

Potreban broj iteracija da se postigne tačnost ε je

$$n \ge \frac{1}{\ln 2}\, \ln \frac{b-a}{\varepsilon}.$$

Metoda regula-falsi (metoda lažnog položaja)

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f(x_F) - f(x^{(n)})}\,(x_F - x^{(n)}), \qquad n = 0, 1, \ldots,$$

$x^{(0)}$ i x_F su obično krajevi intervala na kome je lokalizovano rešenje.

Metoda sečice

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})}\,(x^{(n-1)} - x^{(n)}), \qquad n = 1, 2, \ldots,$$

$x^{(0)}$ i $x^{(1)}$ su obično krajevi intervala na kome je lokalizovano rešenje.
Greška i metode regula-falsi i metode sečice može se oceniti izrazom

$$|x^* - x^{(n+1)}| \le \frac{M_1 - m_1}{m_1}\,|x^{(n+1)} - x^{(n)}|,$$

gde je

$$M_1 = \max_{[a,b]} |f'(x)|, \qquad m_1 = \min_{[a,b]} |f'(x)|.$$

Ocena greške približnog rešenja $x^{(n)}$ koja ne zavisi od metode kojom je rešenje određeno je

$$|x^* - x^{(n)}| \le \frac{1}{m_1}\,|f(x^{(n)})|.$$

7.1 *Sa tačnošću* $5 \cdot 10^{-8}$ *naći sva pozitivna rešenja jednačine*

$$\exp(-x)\,|2 + x - x^2|^{-1/4} = 10.$$

Rešenje: Singulariteti funkcije $f(x) = \exp(-x)\,|2 + x - x^2|^{-1/4} - 10$ su $x = 2$ i $x = -1$. Pošto se traže pozitivna rešenja jednačine, analiziraćemo ponašanje funkcije f(x) u intervalima $(0, 2)$ i $(2, \infty)$. U intervalu $(0, 2)$ je

$$f'(x) = \exp(-x)\left(x - \frac{1 + \sqrt{37}}{4}\right)\left(x - \frac{1 - \sqrt{37}}{4}\right)\big((2 - x)(x + 1)\big)^{-5/4},$$

te $f(x)$ dostiže minimum za $x = 1.77$. Za $0 < x < 1.77$ je $f' < 0$, tj. f opada, a za $1.77 < x < 2$ je $f' > 0$ i f raste. U intervalu $(2, \infty)$ je $f' < 0$ za svako x, što znači da f stalno opada. Tako su intervali monotonosti funkcije $f(x)$ $(0, 1.77)$, $(1.77, 2)$, $(2, \infty)$. Na krajevima ovih intervala je

$$f(0) = -9.16, \qquad f(1.77) = -9,81, \qquad \lim_{x \to 2} f(x) = \infty, \qquad \lim_{x \to \infty} f(x) = -10,$$

što znači da $f(x)$ ima pozitivne korene, tj. data jednačina ima pozitivna rešenja u intervalima $(1.77, 2)$ i $(2, \infty)$. Preciznije, rešenja jednačine pripadaju intervalima

$$x_1^* \in (2 - 10^{-7}, 2 - 10^{-8}), \qquad x_2^* \in (2 + 10^{-8}, 2 + 10^{-7}).$$

Metodom polovljenja dobijamo da su rešenja sa traženom tačnošću

$$x_1^* = 1.99999997, \qquad x_2^* = 2.00000003.$$

7.2 *Metodom regula falsi odrediti sa tačnošću* $5 \cdot 10^{-5}$ *sva rešenja jednačine*

$$\exp x - 2(x-1)^2 = 0.$$

Rešenje: Analizirajmo funkciju $f(x) = \exp x - 2(x-1)^2$. Njeni izvodi su $f'(x) = \exp x - 4(x-1)$, $f''(x) = \exp x - 4$. Pri tome je $f''(\ln 4) = 0$, i $f''(x) < 0$ za $x < \ln 4$ i $f''(x) > 0$ za $x > \ln 4$. Stoga za $x \in (-\infty, \ln 4)$ $f'(x)$ opada, a za $x \in (\ln 4, \infty)$ $f'(x)$ raste. Kako je još $f'(\ln 4) = 8(1 - \ln 2) > 0$ i $f'(x)$ neprekidna funkcija, to je $f'(x) > 0$ za svako x. S obzirom da je $f(-\infty) < 0$ a $f(\infty) > 0$, sledi da funkcija $f(x)$ ima jednu realnu nulu, tj. da data jednačina ima jedno rešenje. Sa grafika zaključujemo da je ono u intervalu $(0, 0.5)$. Aproksimacije rešenja određujemo primenom iterativnog algoritma za izbor početne aproksimacije $x^{(0)} = 0.2$, i fiksne tačke $x_F = 0.3$. Još ocenjujemo dozvoljenu razliku vrednosti dveju uzastopnih iteracija, s obzirom na zadatu tačnost. Kako je

$$M_1 = \max_{(0,0.5)} |f'(x)| = 5, \qquad m_1 = \min_{(0,0.5)} |f'(x)| = 3.65,$$

to je dovoljno da bude ispunjen uslov

$$|x^{(n)} - x^{(n-1)}| \leq \frac{m_1}{M_1 - m_1} \frac{1}{2} 10^{-4} = 1.4 \cdot 10^{-4}.$$

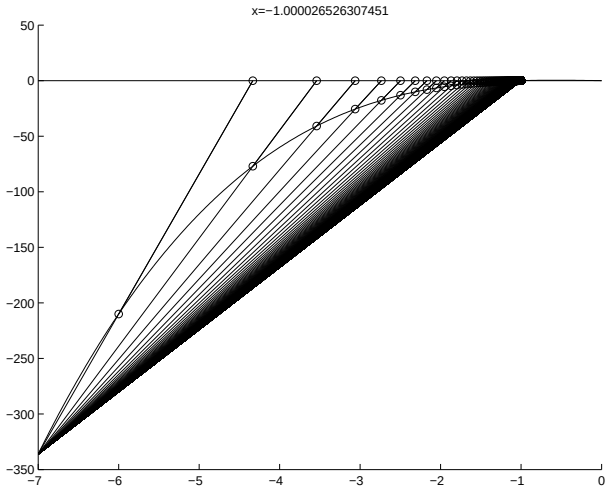
Ovaj uslov je ispunjen u trećoj iteraciji. Rešenje jednačine sa traženom tačnošću je

$$x^* = 0.2133.$$

MATLAB

7.3 *Koristeći* MATLAB, *grafički prikazati rešavanje jednačine* $x^3 = x$ *metodom regula-falsi na intervalu* $[-7, 0]$ *za fiksirani levi kraj intervala i početnu iteraciju* $x_0 = -6$.

Rešenje: Ovaj primer pokazuje moguću sporost konvergencije metode regula-falsi.



Slika 7.1: Metoda Regula-falsi primenjena na funkciju $f(x) = x^3 - x$.

```
% Konstruisemo inline objekat koji odgovara funkciji f
f = inline('x.^3-x');
% Postavljamo krajeve intervala i fiksnu tacku
a = -7; b = 0; xF = -7;
% Postavljamo pocetnu iteraciju i tacnost
x0 = -6; epsilon = 1e-6;

% Crtamo funkciju f na intervalu [a,b]
t  = linspace(a, b, 250);
ft = feval(f, t);
clf, hold on, plot(t, ft, 'g');
% Crtamo x osu
plot([a b], [0 0], 'k');

% x i y=f(x) sadrze koordinate tacke tekuce iteracije
% Inicijalizujemo ih na pocetnu iteraciju
x = x0; y = feval(f,x0);
% yF = f(xF)
yF = feval(f,xF);

while(1)
    % Pamtimo vrednosti prethodne iteracije
    xp = x; yp = y;

    % Proveravamo da li postoji presek secice sa x osom
    if (yF == yp)
        error('Secica nema presek sa x osom!');
    end

    % Izracunavamo tacku sledece iteraciju
    x = xp - yp/(yF-yp)*(xF-xp);
    y = feval(f, x);
```

```
    %Iscrtavamo karakteristicne tacke na secici
    plot(xp, yp, 'o'); plot(x, 0, 'o');

    % Secicu iscrtavamo iz dva dela
    plot([xF x], [yF 0],'r'); plot([xp x], [yp 0],'r');

    % Postavljamo naslov grafika i cekamo dok korisnik ne pritisne taster
    title(['x=' num2str(x,16)]); pause

    % Ukoliko je postignuta zeljena tacnost, zavrsavamo
    if abs(xp-x)<epsilon
        break;
    end
end
```

7.1 Metoda iteracije

Nepokretna tačka operatora G je tačka za koju važi da je

$$x = G(x)$$

Operator kontrakcije $G : \mathcal{B} \to \mathcal{B}$ zadovoljava uslov

$$\|G(x) - G(y)\| \leq q\|x - y\|, \quad \forall x, y \in \mathcal{B}, \qquad q < 1.$$

Metoda iteracije se definiše iterativnim algoritmom

$$\mathbf{x}^{(n+1)} = \mathbf{g}(\mathbf{x}^{(n)}), \qquad n = 0, 1, \ldots,$$

pri čemu je operator $\mathbf{g}$ operator kontrakcije definisan transformacijom polaznog sistema u ekvivalentan sistem

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \qquad \Longleftrightarrow \qquad \mathbf{x} = \mathbf{g}(\mathbf{x})$$

Dovoljan uslov konvergencije iterativnog algoritma je

$$q = \|J\| < 1, \qquad J(\mathbf{x}) = \begin{pmatrix} \frac{\partial g_1(\mathbf{x})}{\partial x_1} & \frac{\partial g_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial g_1(\mathbf{x})}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m(\mathbf{x})}{\partial x_1} & \frac{\partial g_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial g_m(\mathbf{x})}{\partial x_m} \end{pmatrix}$$

gde je $\|J\|$ neka norma Jakobijeve matrice (jakobijana) vektora $\mathbf{g} = (g_1, \ldots, g_m)^\top$. Kada je $m = 1$ taj uslov se za neprekidno diferencijabilnu funkciju svodi na uslov

$$q = \max_{[a,b]} |g'(x)| < 1.$$

Apriorna i aposteriorna ocena greške su

$$\|\mathbf{x}^* - \mathbf{x}^{(n)}\| \le \frac{q^n}{1-q}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|, \qquad \|\mathbf{x}^* - \mathbf{x}^{(n)}\| \le \frac{q}{1-q}\|\mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}\|.$$

Iz apriorne ocene sledi da je potreban broj iteracija da bi se postigla tačnost ε

$$n \ge \frac{1}{\ln q}\,\ln\frac{\varepsilon(1-q)}{\|\mathbf{g}(\mathbf{x}^{(0)}) - \mathbf{x}^{(0)}\|}$$

7.4 *Neka je* $f(x) = \pi/2 + x - \arctan x$, $x \in \mathcal{R}$. *Dokazati da za svaki par tačaka* x, y *postoji konstanta* $q < 1$ *tako da je*

$$|f(x) - f(y)| \le q|x - y|$$

a preslikavanje f *nema nepokretnu tačku.*

Rešenje: Prema teoremi o srednjoj vrednosti je

$$|f(x) - f(y)| = \left(1 - \frac{1}{1+\xi^2}\right)|x - y|, \qquad \xi \in (x, y).$$

Dakle, $q = q(x, y) = 1 - 1/(1 + \xi^2) < 1$, ali ne postoji majoranta ove funkcije koja ne zavisi od x i y a manja je od 1. Preslikavanje f nema nepokretnu tačku jer bi u toj tački bilo

$$x^* = \pi/2 + x^* - \arctan x^*, \qquad \text{tj.} \qquad \arctan x^* = \pi/2,$$

što je nemoguće.

7.5 *Neka su* α *i* β *realni koreni kvadratne jednačine* $x^2 + a_1 x + a_2 = 0$. *Dokazati da sledeći iterativni algoritmi konvergiraju ka korenu* α

$$x^{(n+1)} = -\frac{1}{x^{(n)}}(a_1 x^{(n)} + a_2) \qquad za \qquad |\alpha| > |\beta|, \tag{1}$$

$$x^{(n+1)} = -\frac{a_2}{x^{(n)} + a_1} \qquad za \qquad |\alpha| < |\beta|, \tag{2}$$

$$x^{(n+1)} = -\frac{\left(x^{(n)}\right)^2 + a_2}{a_1} \qquad za \qquad |2\alpha| < |\alpha + \beta|, \tag{3}$$

Rešenje: Ako je iterativni algoritam definisan jednačinom $x = g(x)$ koja je ekvivalentna polaznoj jednačini, uslov konvergencije se svodi na uslov $|g'(x)| < 1$ u nekoj okolini rešenja. Odavde, uzimajući u obzir Viete-ove formule $\alpha + \beta = -a_1$ i

$\alpha\beta = a_2$, za date iterativne algoritme neposredno slede navedeni uslovi

$$g(x) = -a_1 - \frac{a_2}{x}, \qquad |g'(\alpha)| = \Big|\frac{a_2}{\alpha^2}\Big| = \Big|\frac{\alpha\beta}{\alpha^2}\Big| = \Big|\frac{\beta}{\alpha}\Big| < 1, \tag{1}$$

$$g(x) = -\frac{a_2}{x + a_1}, \qquad |g'(\alpha)| = \Big|\frac{a_2}{(\alpha + a_1)^2}\Big| = \Big|\frac{\alpha\beta}{\beta^2}\Big| = \Big|\frac{\alpha}{\beta}\Big| < 1, \tag{2}$$

$$g(x) = -\frac{x^2 + a_2}{a_1}, \qquad |g'(\alpha)| = \Big|\frac{2\alpha}{a_1}\Big| = \Big|\frac{2\alpha}{\alpha + \beta}\Big| < 1, \tag{3}$$

7.6 *Postoji li za svako t neprekidna funkcija $x(t)$ takva da je*

$$x(t) - \frac{1}{2}\sin x(t) + a(t) = 0,$$

gde je $a(t)$ zadata neprekidna funkcija. Ako postoji, odrediti broj iteracija koje je potrebno uraditi da se $x(t)$ odredi sa tačnošću 10^{-2}.

Rešenje: Za svako t nepokretna tačka $x(t)$ funkcije $g(x) = \frac{1}{2}\sin x - a(t)$ postoji i jedinstveno je određena jer je g kontrakcija sa koeficijentom kontrakcije $q = 0.5$,

$$\begin{aligned}\|g(x(t)) - g(y(t))\| &= \max|\frac{1}{2}\sin x(t) - a(t) - \frac{1}{2}\sin y(t) + a(t)| \\ &= \max|\cos\frac{x(t) + y(t)}{2}\,\sin\frac{x(t) - y(t)}{2}| \le \frac{1}{2}\,\|x(t) - y(t)\|,\end{aligned}$$

pošto je $|\cos z| \le 1$, $|\sin z| \le |z|$ za svako z.

Broj iteracija iterativnog algoritma $x^{(n+1)} = g(x^{(n)})$ koje treba uraditi da bi se postigla tačnost ε je

$$n \ge \frac{1}{\ln q}\ln\frac{\varepsilon(1-q)}{\|x^{(1)} - x^{)0)}\|}.$$

Ako izaberemo da je $x^{(0)}(t) = -a(t)$, onda je

$$x^{(1)} = -\frac{1}{2}\sin a(t) - a(t), \qquad \|x^{(1)}(t) - x^{(0)}(t)\| = \max\Big|-\frac{1}{2}\sin a(t)\Big| \le \frac{1}{2},$$

pa je za postizanje tačnosti $\varepsilon = 10^{-2}$ potrebno $n \ge 6.64$ iteracija. Kako n treba da bude prirodan broj, potreban broj iteracija je $n = 7$.

7.7 *Metodom iteracije na tri sigurne cifre odrediti pozitivno rešenje sistema:*

$$\begin{aligned}f_1(x, y) &\equiv x^3 + y^3 - 6x + 3 = 0 \\ f_2(x, y) &\equiv x^3 - y^3 - 6y + 2 = 0\end{aligned}$$

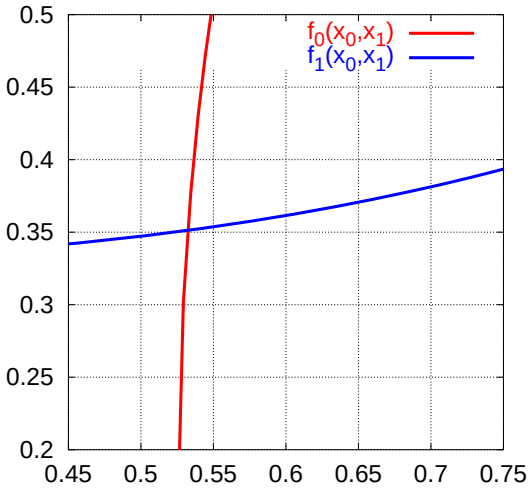
Rešenje: Sistem se prvo transformiše u oblik $\mathbf{x} = \mathbf{g}(\mathbf{x})$, recimo, na sledeći način:

$$\begin{aligned} x &= \tfrac{1}{6}\left(x^3 + y^3 + 3\right) \equiv g_1(x,y) \\ y &= \tfrac{1}{6}\left(x^3 - y^3 + 2\right) \equiv g_2(x,y) \end{aligned}$$

Jakobijan vektora $\mathbf{g}$ je

$$J = \begin{pmatrix} x^2/2 & y^2/2 \\ x^2/2 & -y^2/2 \end{pmatrix}$$

Treba odrediti oblast u kojoj će se tražiti rešenje i početne vrednosti za iterativni postupak. Ukoliko ovi parametri nisu zadati, mogu se odrediti grafički. Na slici 7.2 su prikazani grafici funkcija $f_1(x,y) = 0$ i $f_2(x,y) = 0$. Približne vrednosti koordinata tačke preseka ova dva grafika predstavljaju početne vrednosti $x^{(0)} = 1/2$ i $y^{(0)} = 1/3$, a oblast u kojoj se traži rešenje je $D = \{(x,y) \mid \frac{1}{2} \le x \le \frac{5}{6},\ \frac{1}{6} \le y \le \frac{1}{2}\}$.



Slika 7.2: Grafici funkcija $f_1(x,y) = 0, \quad f_2(x,y) = 0.$

Uniformna norma jakobijana u oblasti D je

$$q = \|J\|_\infty = \max_i \sum_{j=1}^{2} |a_{ij}| = \frac{1}{2}\left(\frac{5}{6}\right)^2 + \frac{1}{2}\left(\frac{1}{2}\right)^2 = \frac{17}{36}$$

(sa a_{ij} su označeni elementi matrice J). Ova norma predstavlja koeficijent kontrakcije q, i kako je $q < 1$ uslov konvergencije iterativnog postupka je ispunjen. S obzirom da rešenje treba odrediti na tri sigurne cifre, sledi da je tražena tačnost $\varepsilon = 0.5 \cdot 10^{-3}$. Korišćenjem aposteriorne ocene greške, zaključuje se da će iterativni postupak biti završen kada bude zadovoljen uslov

$$\max\left|x^{(n)} - x^{(n-1)},\, y^{(n)} - y^{(n-1)}\right| < \frac{1-q}{q}\,\varepsilon = 0.5 \cdot 10^{-3}$$

Iterativni postupak je određen formulama

$$\begin{aligned} x^{(n+1)} &= \tfrac{1}{6}\big((x^{(n)})^3 + (y^{(n)})^3 + 3\big) \\ y^{(n+1)} &= \tfrac{1}{6}\big((x^{(n)})^3 + (y^{(n)})^3 + 2\big) \end{aligned}$$

po kojima su izračunate vrednosti predstavljene sledećom tabelom:

$x^{(n)}$	0.5	0.5270	0.5314	0.5322	0.5324
$y^{(n)}$	0.3333	0.3480	0.3507	0.3511	0.3513

U četvrtoj iteraciji dobija se rešenje na tri sigurne cifre

$$(x^*\,,\, y^*) = (0.532\,,\, 0.351).$$

Treba uočiti da se na osnovu apriorne ocene dobija da je za postizanje tačnosti potrebno uraditi bar 7 iteracija.

7.8 *Odrediti sa tačnošću* 10^{-2} *rešenje sistema jednačina:*

$$\begin{aligned} x + 3\log(x) - y^2 &= 0 \\ 2x^2 - xy - 5x + 1 &= 0 \end{aligned}$$

u okolini tačke $(3.4, 2.2)$.

Rešenje: Sistem se može napisati u obliku $\mathbf{x} = \mathbf{g}(\mathbf{x})$ recimo na sledeći način

$$x = -3\log(x) + y^2 \equiv g_1(x,y) \qquad y = 2x - 5 + \frac{1}{x} \equiv g_2(x,y).$$

Tada je jakobijan preslikavanja

$$J(x,y) = \begin{pmatrix} -3/(x\ln 10) & 2y \\ 2 - 1/x^2 & 0 \end{pmatrix} \qquad q = \|J(3.4, 2.2)\|_\infty = 1.913 > 1$$

Uslov konvergencije metode iteracije nije ispunjen. Stoga je potrebno naći drugu tranformaciju sistema, na primer

$$\begin{aligned} x &= \sqrt{\tfrac{1}{2}(xy + 5x - 1)} \equiv g_1(x,y) \\ y &= \sqrt{x + 3\log(x)} \equiv g_2(x,y) \end{aligned}$$

U ovom slučaju jakobijan vektora $\mathbf{g}$ je

$$J = \begin{pmatrix} \tfrac{1}{2}(y+5)/\sqrt{2(xy+5x-1)} & \tfrac{1}{2}\,x/\sqrt{2(xy+5x-1)} \\ \tfrac{1}{2}\big(1 + \tfrac{3}{x\ln 10}\big)/\sqrt{x + 3\log(x)} & 0 \end{pmatrix}$$

pa je u okolini početne aproksimacije $D = \{(x,y)|\ |x-3.4| \le 0.1,\ |y-2.2| \le 0.1\}$, uniformna norma jakobijana

$$q = \|J\|_\infty = \max\Big(\big|\frac{2.3+5}{2\sqrt{2(3.3\cdot 2.1+5\cdot 3.3-1)}}\big| + \big|\frac{3.5}{2\sqrt{2(3.3\cdot 2.1+5\cdot 3.3-1)}}\big|\,,$$
$$\big|\frac{1+\frac{3}{3.3ln10}}{2\sqrt{3.3+3\log(3.3)}}\big|\Big) = \max(0.54+0.26\,,\,0.32) = 0.8$$

Dakle, metoda iteracije definisana formulama

$$\begin{aligned} x^{(n+1)} &= \sqrt{\tfrac{1}{2}(x^{(n)}y^{(n)} + 5x^{(n)} - 1)} \\ y^{(n+1)} &= \sqrt{x^{(n)} + 3\log(x^{(n)})} \end{aligned}$$

konvergira, a tačnost je postignuta kada je

$$\max\big|x^{(n)} - x^{(n-1)}\,,\, y^{(n)} - y^{(n-1)}\big| \le \frac{1-q}{q}\,\varepsilon = 0.25\cdot 10^{-2}$$

U osmoj iteraciji postiže se tražena tačnost, i rešenje je

$$(x^*, y^*) = (3.48\,,\,2.26).$$

7.9 *a)Korišćenjem teoreme o nepokretnoj tački dokazati egzistenciju jedinstvenog rešenja sistema nelinearnih jednačina*

$$x^3y^2 + 17y = a, \qquad 9x - x^4\sin y^2 = b,$$

za $|x| \le 1, |y| \le 2$*; a i b su dati brojevi koji zadovoljavaju uslov* $|a| \le 30$, $|b| \le 8$. *b) Metodom iteracije odrediti sa tačnošću* $5\cdot 10^{-5}$ *rešenje zadatka definisanog pod tačkom a) za* $a = 8.5$ *i* $b = 4.5$.

Rešenje: a)Transformišimo dati sistem u sistem oblika

$$(*) \qquad x = \frac{1}{9}\big(b + x^4\sin y^2\big), \qquad y = \frac{1}{17}(a - x^3y^2).$$

Egzistenciju jedinstvenog rešenja datog sistema, tj. nepokretne tačke preslikavanja $\mathbf{g}$,

$$\mathbf{x} = \mathbf{g}(\mathbf{x}) \qquad \text{gde je} \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \mathbf{g}(\mathbf{x}) = \begin{pmatrix} \big(b + x^4\sin y^2\big)/9 \\ (a - x^3y^2)/17 \end{pmatrix},$$

dokazaćemo tako što ćemo pokazati da je preslikavanje $\mathbf{g}$ kontrakcija. Imamo da je

$$\begin{aligned} \mathbf{g}(\mathbf{u}) - \mathbf{g}(\mathbf{x}) &= \begin{pmatrix} \big(u^4\sin v^2 - x^4\sin y^2\big)/9 \\ (x^3y^2 - u^3v^2)/17 \end{pmatrix} \\ &= \begin{pmatrix} \big(u^4(\sin v^2 - \sin y^2) + (u^4 - x^4)\sin y^2\big)/9 \\ (x^3(y^2 - v^2) + (x^3 - u^3)v^2)/17 \end{pmatrix} \end{aligned}$$

gde je $\mathbf{u} = (u, v)^\top$. Na osnovu pretpostavki zadatka je

$$\begin{aligned}\frac{1}{9}\big|u^4(\sin v^2 - \sin y^2) + (u^4 - x^4)\sin y^2\big| &\\ \le \frac{1}{9}\left(|u^4|\,\big|2\cos\frac{v^2+y^2}{2}\sin\frac{v^2-y^2}{2}\big| + |u^4 - x^4|\,\big|\sin y^2\big|\right)&\\ \le \frac{1}{9}\left(2\big|\sin\frac{v^2-y^2}{2}\big| + |u-x|\,|u+x|\,(u^2+x^2)\right)&\\ \le \frac{1}{9}\left(|v-y|\,|v+y| + 4|u-x|\right) \le \frac{8}{9}\max\left(|u-x|, |v-y|\right)&\end{aligned}$$

jer je $|\cos x| \le 1$, $|\sin x| \le |x|$, i

$$\frac{1}{17}\big|x^3(y^2 - v^2) + (x^3 - u^3)v^2\big| \le \frac{12}{17}|u-x| + \frac{4}{17}|v-y| \le \frac{16}{17}\max\left(|u-x|, |v-y|\right).$$

Stoga je

$$\|\mathbf{g}(\mathbf{u}) - \mathbf{g}(\mathbf{x})\|_\infty \le \frac{16}{17}\max\left(|u-x|, |v-y|\right) \le \frac{16}{17}\|\mathbf{u} - \mathbf{v}\|_\infty.$$

Dakle, $\mathbf{g}$ je kontrakcija sa konstantom kontrakcije $q = 16/17$.

Uslov $|a| \le 30$, $|b| \le 8$ je dovoljan da iterativni postupak bude dobro definisan, tj. da je $|x| \le 1$, $|y| \le 2$. Naime, ako iteracije (x_k, y_k), $k = 0, \ldots, n$ za proizvoljno n zadovoljavaju ovaj uslov, onda je i

$$\begin{aligned}|x_{n+1}| &\le \frac{1}{9}\left(|b| + x_n^4|\sin y_n^2|\right) \le 1\\ |y_{n+1}| &\le \frac{1}{17}\left(|a| + |x_n|^3 y_n^2\right) \le 2\end{aligned}$$

b) S obzirom na zaključak pod a), sistem možemo rešiti metodom proste iteracije definisanom sistemom $(*)$. Da bismo dobili rezultat sa traženom tačnošću, potrebno je računati dok ne bude zadovoljen sledeći kriterijum

$$\max\left(|x^{(n)} - x^{(n-1)}|, |y^{(n)} - y^{(n-1)}|\right) \le 3.125 \cdot 10^{-6}.$$

Polazeći od početne aproksimacije $x^{(0)} = b/9 = 0.5$, $y^{(0)} = a/17 = 0.5$ dobijamo da je

$$x^{(2)} = 0.501729, \quad y^{(2)} = 0.498156, \qquad x^{(3)} = 0.501729, \quad y^{(2)} = 0.498156,$$

pa je rešenje datog sistema sa traženom tačnošću

$$(x^*, y^*) = (0.5017,\, 0.4982).$$

7.10 *Metodom iteracije odrediti sa tačnošću* 10^{-3} *sva rešenja sistema jednačina*

$$\begin{aligned} 5x - 6y + 20\ln x &= -16 \qquad (1)\\ 2x + y - 10\ln x &= 4 \qquad (2) \end{aligned}$$

Rešenje: Uvedimo oznaku

$$Y(x) \equiv f_1(x) - f_2(x) = \frac{1}{6}(17x - 40\ln x - 8),$$

gde je $f_1(x) = (5x + 20\ln x + 16)/6$ i $f_2(x) = -2x + 10\ln x + 4$. Očigledno je da su nule funkcije $Y(x)$ apscise rešenja traženog sistema, pa pomoću nje možemo oceniti gde se rešenja nalaze i koliko ih ima. Kako je $Y'(x) = (17 - 40/x)/6 = 0$ za $x = 40/17 \approx 2.35$, a oblast definisanosti funkcije $Y(x)$ je $x > 0$, to su intervali monotonosti ove funkcije $(0, 2.35)$ i $(2.35, \infty)$. Još je $\lim_{x\to 0} Y(x) = \infty$, $Y(2.35) = -0.37$ i $\lim_{x\to\infty} Y(x) = \infty$, pa u svakom od intervala monotonosti funkcija $Y(x)$ ima po jednu nulu. To znači da u svakom od ovih intervala po x polazni sistem ima po jedno rešenje. Preciznije, $x_1 \in (1.6, 1.7)$ jer je $Y(1.6) = 0.07$ i $Y(1.7) = -0.05$, a $x_2 \in (3.2, 3.3)$ jer je $Y(3.2) = -0.02$ i $Y(3.3) = 0.06$. Stoga polazni sistem ima dva rešenja i za njihove početne aproksimacije su uzete vrednosti $x_1^{(0)} = 1.6$, $y_1^{(0)} = 5.5$, $x_2^{(0)} = 3.2$, $y_2^{(0)} = 9.2$.

Definišimo iterativni postupak za nalaženje prvog rešenja sa traženom tačnošću. Linearnim kombinacijama jednačina sistema ($(1) + 2(2)$ i $(1) + 4(2)$) dobijamo konvergentan iterativni postupak definisan izrazima

$$\begin{aligned} x_1^{(n+1)} &= \frac{4}{9}(y_1^{(n)} - 2) \equiv g_1(x_1^{(n)}, y_1^{(n)}),\\ y_1^{(n+1)} &= 6.5x_1^{(n)} - 10\ln x_1^{(n)} \equiv g_2(x_1^{(n)}, y_1^{(n)}) \end{aligned} \qquad n = 0, 1, \ldots.$$

Kada $x \in (1.5, 1.7)$ preslikavanje definisano funkcijama $g_1(x, y)$ i $g_2(x, y)$ je kontrakcija jer je

$$\left|\frac{\partial g_1}{\partial x}\right| = 0, \qquad \left|\frac{\partial g_1}{\partial y}\right| = \frac{4}{9}, \qquad \left|\frac{\partial g_2}{\partial x}\right| \le 0.62, \qquad \left|\frac{\partial g_2}{\partial y}\right| = 0,$$

pa je uniformna norma jakobijana preslikavanja manja ili jednaka od $q = 0.62$. Tačnost je postignuta kada je

$$(*) \qquad \max(|x_1^{(n)} - x_1^{(n-1)}|, |y_1^{(n)} - y_1^{(n-1)}|) \le \frac{1-q}{q}10^{-3} \le 6 \cdot 10^{-4}.$$

Polazeći od $x_1^{0)} = 1.6$, $y_1^{(0)} = 5.5$ dobija se primenom iterativnog algoritma

$$x_1^{(7)} = 1.6522, \quad y_1^{(7)} = 5.7183, \qquad x_1^{(8)} = 1.6526, \quad y_1^{(8)} = 5.7182.$$

Kriterijum $(*)$ je zadovoljen, te je sa traženom tačnošću prvo rešenje sistema

$$(x_1^*, y_1^*) = (1.653, 5.718).$$

Definišemo iterativni postupak za nalaženje drugog rešenja sa traženom tačnošću. Linearnim kombinacijama jednačina sistema ((1)+6(2) i (1)+18(2)) definišemo konvergentan iterativni postupak formulama

$$x_2^{(n+1)} = \frac{1}{17}(40\ln x_2^{(n)} + 8) \equiv h_1(x_2^{(n)}, y_2^{(n)}),$$
$$y_2^{(n+1)} = \frac{1}{12}(160\ln x_2^{(n)} - 41x_2^{(n)} + 56) \equiv h_2(x_2^{(n)}, y_2^{(n)}) \qquad n = 0, 1, \ldots.$$

Iterativni proces konvergira kada $x \in (3.2, 3.3)$ jer je

$$\left|\frac{\partial h_1}{\partial x}\right| \le 0.74, \qquad \left|\frac{\partial h_1}{\partial y}\right| = 0, \qquad \left|\frac{\partial h_2}{\partial x}\right| \le 0.75, \qquad \left|\frac{\partial h_2}{\partial y}\right| = 0,$$

pa je uniformna norma jakobijana preslikavanja manja ili jednaka od konstante $q = 0.75$. Tačnost je postignuta kada je

$$(**) \qquad \max(|x_2^{(n)} - x_2^{(n-1)}|, |y_2^{(n)} - y_2^{(n-1)}|) \le \frac{1-q}{q}10^{-3} \le 3 \cdot 10^{-4}.$$

Polazeći od $x_2^{(0)} = 3.2$, $y_2^{(0)} = 9.2$ dobija se primenom iterativnog algoritma

$$x_2^{(11)} = 3.2268, \quad y_2^{(11)} = 9.2614, \qquad x_2^{(12)} = 3.2270, \quad y_2^{(12)} = 9.2616.$$

Kriterijum $(**)$ je zadovoljen, te je sa traženom tačnošću drugo rešenje sistema

$$(x_2^*, y_2^*) = (3.227, 9.262).$$

7.11 *Metodom iteracije izračunati na pet sigurnih cifara rešenje sistema jednačina*

$$4x^3 - 27xy^2 = -25 \qquad 4x^2y - 3y^3 = 1,$$

u prvom kvadrantu.

Rešenje: Transformišimo dati sistem u sistem jednačina oblika

$$(*) \qquad x = \sqrt{\frac{1}{4y}(3y^3 + 1)} \equiv g_1(x, y), \qquad y = \sqrt{\frac{1}{27x}(4x^3 + 25)} \equiv g_2(x, y).$$

U pravougaoniku $D = \{(x, y) \,|\, 0.9 \le x, y \le 1.1\}$ je, zato što je

$$\left|\frac{\partial g_1}{\partial x}\right| = 0, \qquad \left|\frac{\partial g_1}{\partial y}\right| \le 0.79, \qquad \left|\frac{\partial g_2}{\partial x}\right| \le 0.42, \qquad \left|\frac{\partial g_2}{\partial y}\right| = 0,$$

uniformna norma jakobijana preslikavanja definisanog funkcijama g_1 i g_2 manja od konstante $q = 0.79$. Stoga iterativni postupak definisan ovim preslikavanjem konvergira, a tačnost je postignuta kada je

$$\max(|x^{(n)} - x^{(n-1)}|, |y^{(n)} - y^{(n-1)}|) \le \frac{1-q}{q}\,\frac{1}{2}10^{-4} \le 1.3 \cdot 10^{-5}.$$

Polazeći od $x^{(0)} = 1$, $y^{(0)} = 1$ dobija se primenom iterativnog algoritma

$$x^{(6)} = 1.01944, \quad y^{(6)} = 1.03065, \qquad x^{(7)} = 1.01943, \quad y^{(7)} = 1.03065.$$

Kriterijum tačnosti je zadovoljen, te je u prvom kvadrantu rešenje sistema na pet sigurnih cifara

$$(x^*, y^*) = (1.0194,\ 1.0306).$$

7.12 *Metodom iteracije sa tačnošću* $5 \cdot 10^{-4}$ *odrediti rešenje sistema jednačina*

$$x + 3\log x - y^2 = 0 \qquad 2x^2 - x\,y - 5x + 1 = 0,$$

koje ima veću apscisu.

Rešenje: Sistem ima dva rešenja, jedno u okolini tačke $(1.4, -1.4)$ i drugo u okolini tačke $(3.4, 2.2)$. Drugo rešenje (sa većom apscisom) izračunato je metodom iteracije određenom rekurentnim formulama

$$\begin{aligned} x^{(n+1)} &= \sqrt{\frac{1}{2}(5x^{(n)} + x^{(n)}y^{(n)} - 1)}, \\ y^{(n+1)} &= \sqrt{x^{(n)} + 3\log x^{(n)}} \end{aligned} \qquad n = 0, 1, \ldots.$$

Koeficijent kontrakcije $q = 0.92$, te je tačnost postignuta kada je

$$\max(|x^{(n)} - x^{(n-1)}|, |y^{(n)} - y^{(n-1)}|) \leq 4 \cdot 10^{-5}.$$

Polazeći od $x^{(0)} = 3.4$, $y^{(0)} = 2.2$ dobija se primenom iterativnog algoritma u 16-oj iteraciji rešenje sa traženom tačnošću

$$(x^*, y^*) = (3.487,\ 2.262).$$

FORTRAN Implementacija metode iteracije je jednostavna. Procedura koja implementira metodu štiti se od slučajeva kada metoda ne konvergira pomoću argumenta koji predstavlja maksimalni dozvoljeni broj iteracija. Ostatak procedure se sastoji u prostom izračunavanju sledećih aproksimacija rešenja tako što se vektor tekućih vrednosti uvrsti u funkciju $\mathbf{g}(\mathbf{x})$. Procedura intenzivno koristi biblioteku `libmatheval`. Takodje, treba napomenuti da će počev od ovog poglavlja pa do kraja zbirke implementacije numeričkih metoda biti pisane programskim jezikom *Fortran 90*. Treba obratiti pažnju kako se koriste ugrađene operacije nad vektorima koje *Fortran* podržava. Tako procedura `iteration()` ima sledeći oblik:

```
module iteration_module
  use matheval_module
  implicit none
  public iteration

  ! Deklaracija tipa za stringove fiksne duzine.
```

```
  type string
     character(len=256) :: chars
  end type string

contains
  ! Funkcija iteration() resava sistem nelinearnih jednacina oblika
  ! x=g(x) metodom iteracije. Argumenti funkcije su:
  !   n - dimenzija sistema
  !   variables - string sa imenima promenljivih
  !   g - polje sa desnim stranama jednacina sistema
  !   x - polje koje sadrzi pocetnu aproksimaciju resenja, a u koje ce biti
  !       smesteno resenje
  !   epsilon - zadata tacnost
  !   max_iterations - maksimalni dozvoljeni broj iteracija
  ! Pretpostavlja se da su sva polja alocirana izvan funkcije. Dimenzija
  ! sistema, tacnost i maksimalni broj iteracija moraju biti veci od
  ! 0. Jednacine moraju biti sintaksno ispravno zadate (za detalje
  ! sintakse videti dokumentaciju biblioteke libmatheval). Rezultat
  ! koji vraca funkcija pokazuje da li je dostignuta zadata tacnost.
  logical function iteration (n, variables, g, x, epsilon, max_iterations)
    integer, intent(in) :: n
    type(string), intent(in) :: variables
    type(string), dimension(0:n-1), intent(in) :: g
    real(kind=8), dimension(0:n-1), intent(inout) :: x
    real(kind=8), intent(in) :: epsilon
    integer, intent(in) :: max_iterations
    integer(kind=8), dimension(0:n-1) :: evaluator ! Evaluatori za desne
        ! strane jednacina sistema.
    real(kind=8), dimension(0:n-1) :: x_curr, x_next ! Aproksimacija
        ! resenja u tekucoj i narednoj iteraciji.
    real(kind=8) :: error ! Ocena greske u tekucoj iteraciji.
    logical :: flag ! Promenljiva koja pokazuje da li je postignuta
        ! zadata tacnost.
    integer :: i, k ! Brojaci u petljama.

    ! Proverava se validnost ulaznih podataka. Pritom se kreiraju
    ! evaluatori za desne strane jednacina sistema.
    if (n<=0) stop
    if (epsilon<=0) stop
    if (max_iterations<=0) stop
    do k=0, n-1
       evaluator(k)=evaluator_create(g(k)%chars)
       if (evaluator(k)==0) stop
    end do

    ! Inicijalizuju se promenljive bitne za izracunavanje.
    flag=.false.
    x_curr=x

    ! Sve dok se ne premasi maksimalni dozvoljeni broj iteracija...
    do i=0, max_iterations-1
       ! ...izracunava se nova aproksimacija resenja sistema,...
       do k=0, n-1
          x_next(k)=evaluator_evaluate(evaluator(k),n,variables%chars,x_curr)
       end do

       ! ...odredjuje se greska aproksimacije,...
```

```
      error=0
      do k=0, n-1
         if (error<abs(x_next(k)-x_curr(k))) then
            error=abs(x_next(k)-x_curr(k))
         end if
      end do

      ! ...postavlja se tekuca aproksimacija na novu aproksimaciju i...
      x_curr=x_next

      ! ...ukoliko je greska aproksimacije manja od zadate tacnosti,
      ! prekida se sa daljim izracunavanjem.
      if (error<epsilon) then
         flag=.true.
         exit
      end if
   end do

   ! Upisuje se rezultat u odgovarajuce polje.
   x=x_curr

   ! Unistavaju se korisceni evaluatori.
   do k=0, n-1
      call evaluator_destroy(evaluator(k))
   end do

   ! Vraca se odgovarajuci rezultat iz funkcije.
   iteration = flag
 end function iteration
end module iteration_module
```

7.13 *Neka je $\mathcal{X}$ normirani prostor i $S(x^{(0)},r)=\{x\in\mathcal{X}\,|\,\|x-x^{(0)}\|\le r\}$. Dalje, neka je $G:\mathcal{X}\to\mathcal{X}$ operator kontrakcije na $S(x^{(0)},r)$ sa koeficijentom kontrakcije q, $0\le q<1$, takvim da je*

$$\|G(x^{(0)})-x^{(0)}\|\le(1-q)r. \tag{1}$$

Neka je $\tilde{G}:\mathcal{X}\to\mathcal{X}$ operator koji za zadato $\varepsilon>0$ zadovoljava uslove

$$\|G(x)-\tilde{G}(x)\|\le\varepsilon\qquad\forall x\in S(x^{(0)},r) \tag{2}$$

$$2\varepsilon+\|\tilde{G}(x^{(0)})-x^{(0)}\|\le(1-q)r. \tag{3}$$

Dokazati da je za svaki prirodan broj n

$$(*)\qquad \|x^{(n)}-\tilde{x}^{(n)}\|\le\varepsilon\frac{1-q^n}{1-q}\quad i\quad \tilde{x}^{(n)}\in S(x^{(0)},r)$$

gde je

$$x^{(n)}=G(x^{(n-1)}),\qquad \tilde{x}^{(n)}=\tilde{G}(\tilde{x}^{(n-1)}),\qquad n=1,2,\ldots\quad(\tilde{x}^{(0)}=x^{(0)}) \tag{4}$$

Ako je x^* nepokretna tačka operatora G ($G(x^*)=x^*$), dokazati da je

$$(**)\qquad \|x^*-\tilde{x}^{(n)}\|\le\frac{1}{1-q}\left(\varepsilon+q^n\|G(x^{(0)})-x^{(0)}\|\right).$$

Rešenje: Ocenu $(*)$ ćemo dokazati indukcijom. Za $n = 1$ tvrđenje sledi iz uslova (2),

$$\|x^{(1)} - \tilde{x}^{(1)}\| = \|G(x^{(0)}) - \tilde{G}(x^{(0)})\| \leq \varepsilon.$$

Na osnovu (4) i (3) važi ocena

$$\|\tilde{x}^{(1)} - x^{(0)}\| = \|\tilde{G}(x^{(0)}) - x^{(0)}\| \leq (1-q)r - 2\varepsilon \leq (1-q)r,$$

odakle, s obzirom da je $q < 1$, sledi da $\tilde{x}^{(1)} \in S(x^{(0)}, r)$.

Pretpostavimo da tvrđenje važi za svako $k \leq n-1$:

$$\|x^{(k)} - \tilde{x}^{(k)}\| \leq \varepsilon \frac{1-q^k}{1-q}, \qquad \tilde{x}^{(k)} \in S(x^{(0)}, r), \quad k = 1, \ldots, n-1.$$

Tada je za $k = n$, s obzirom da je G kontrakcija i da za $\tilde{x}^{(k)} \in S(x^{(0)}, r)$, $k \leq n-1$, važi pretpostavka (2),

$$\begin{aligned}
\|x^{(n)} - \tilde{x}^{(n)}\| &= \|G(x^{(n-1)}) - \tilde{G}(\tilde{x}^{(n-1)})\| \\
&\leq \|G(x^{(n-1)}) - G(\tilde{x}^{(n-1)})\| + \|G(\tilde{x}^{(n-1)}) - \tilde{G}(\tilde{x}^{(n-1)})\| \\
&\leq q\|x^{(n-1)} - \tilde{x}^{(n-1)}\| + \varepsilon \leq q\varepsilon \frac{1-q^{n-1}}{1-q} + \varepsilon = \varepsilon \frac{1-q^n}{1-q}
\end{aligned}$$

Da bi dokazali da $\tilde{x}^{(n)} \in S(x^{(0)}, r)$ uočimo da je

$$\begin{aligned}
\|\tilde{x}^{(n)} - x^{(0)}\| &\leq \|\tilde{x}^{(n)} - x^{(n)}\| + \|x^{(n)} - x^{(0)}\| \\
&\leq \varepsilon \frac{1-q^n}{1-q} + \frac{1-q^n}{1-q}\|x^{(1)} - x^{(0)}\| \\
&\leq \frac{1-q^n}{1-q}(\varepsilon + \|x^{(1)} - \tilde{x}^{(1)}\| + \|\tilde{x}^{(1)} - x^{(0)}\|) \\
&\leq \frac{1-q^n}{1-q}(2\varepsilon + \|\tilde{G}(x^{(0)}) - x^{(0)}\|)
\end{aligned},$$

pa je na osnovu pretpostavke (3)

$$\|\tilde{x}^{(n)} - x^{(0)}\| \leq (1-q^n)r \leq r.$$

Time je tvrđenje $(*)$ upotpunosti dokazano.

Tvrđenje $(**)$ sledi neposredno iz prethodno dokazanog i teoreme o nepokretnoj tački preslikavanja G,

$$\begin{aligned}
\|x^* - \tilde{x}^{(n)}\| &\leq \|x^* - x^{(n)}\| + \|x^{(n)} - \tilde{x}^{(n)}\| \\
&\leq \frac{q^n}{1-q}\|G(x^{(0)}) - x^{(0)}\| + \varepsilon \frac{1-q^n}{1-q} \\
&\leq \frac{1}{1-q}\big(\varepsilon + q^n\|G(x^{(0)}) - x^{(0)}\|\big)
\end{aligned}$$

Napomena: Zaključak je da se metodom iteracije u ovom slučaju ne može dobiti rešenje sa proizvoljnom tačnošću, tj. greška rešenja ne može biti manja od greške aproksimacije operatora G.

MATLAB

7.14 *Korišćenjem* MATLAB-a, *sa tačnošću* $5 \cdot 10^{-5}$ *odrediti sva rešenja jednačine*

$$x + \ln x = 0.$$

metodom iteracije i grafički prikazati postupak rešavanja jednačine.

Rešenje: Funkcija $f(x) = x + \ln x$ je definisana za $x > 0$, pa data jednačina može imati samo pozitivne korene. Kako je $f'(x) = 1 + 1/x$, to je $f'(x) > 0$ za $x > 0$, što znači da će koren, ako postoji, biti jedinstven. On postoji jer je $f(x)$ neprekidna funkcija i $\lim_{x\to 0_+} f(x) = -\infty$, $\lim_{x\to\infty} f(x) = \infty$. Sa grafika nalazimo da je približna vrednost korena $x^{(0)} = 0.55$. Sa traženom tačnošću odredićemo ga metodom iteracije.

Kako je $|(\ln x)'| > 1$ kada $x \in [0.5, 0.6]$, iterativni postupak definisan jednačinom $x = -\ln x$ ne konvergira. Iterativni postupak ćemo definisati formulom

$$x^{(n+1)} = \exp(-x^{(n)}) \equiv g(x^{(n)}), \qquad n = 0, 1, \ldots.$$

Kako je $|g'(x)| \leq 0.6 = q$ u okolini tačke $x^{(0)}$, iterativni algoritam konvergira, a tačnost će biti zadovoljena ako je

$$|x^{(n)} - x^{(n-1)}| \leq \frac{q}{1-q}\,\frac{1}{2}10^{-4} = 3 \cdot 10^{-5}.$$

```
% Tacnost
epsilon = 3e-5;
% x = f(x)
f = inline('exp(-x)');
% Interval i pocetna iteracija
a = 0.5; b = 0.6; x0 = 0.55;

% Brisemo trenutni prozor i iscrtavamo funkciju na intervalu [a,b]
clf; hold on
t  = linspace(a, b); ft = feval(f, t); plot(t, ft, 'r');
% Iscrtavamo pravu y=x na intervalu [a,b]
plot([a b], [a b]);
% Promenjiva x sadrzi tekucu iteraciju.
% Inicijalizujemo je na pocetnu iteraciju
x = x0;
% Visina na kojoj ce se iscrtavati tacke iteracije.
d = min([a min(ft)]);
% Iscrtavamo pocetnu iteraciju
plot(x, d, 'o');

while(1)
    % Ocekujemo da korisnik pritisne taster
```

```
    pause
    % Nalazimo vrednost funkcije u tacki tekuce iteracije
    fx = feval(f,x);
    % "Dizemo" visinu do grafika funkcije f
    plot([x x], [d fx]);
    % Zatim se "spustamo" do prave y=x
    plot([x fx], [fx fx]);

    % Pamtimo prethodnu iteraciju
    xp = x;
    % Postavljamo koordinate sledece iteracije
    x = fx
    % "Spustamo" se sa prave y=x do x ose
    plot([x x], [x d]);
    % Iscrtavamo vrednost sledece iteracije
    plot(x, d, 'o');

    % Ukoliko je postignuta zeljena tacnost, zavrsavamo
    if (abs(xp-x)<epsilon)
        plot(x, d, 'ro'); break;
    end
end
```

U dvanaestoj iteraciji ovaj uslov je ispunjen i dobijena je približna vrednost rešenja sa traženom tačnošću

$$x^* = 0.5672.$$

7.2 Metoda Newton-a

Definisana je iterativnim algoritmom

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - [\mathbf{f}'(\mathbf{x}^{(n)})]^{-1}\mathbf{f}(\mathbf{x}^{(n)}), \qquad n = 0, 1, \ldots$$

gde je $[\mathbf{f}'(\mathbf{x})]^{-1}$ inverzna matrica jakobijanu vektora $\mathbf{f}(\mathbf{x})$

$$\mathbf{f}'(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \frac{\partial f_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_m} \end{pmatrix}$$

Ova metoda je reda 2, te se pokazuje u praksi da je $\mathbf{x}^{(n)}$ aproksimacija rešenja sa tačnošću ε kada je kada je

$$\|\mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}\| \leq \varepsilon$$

U slučaju jedne dimenzije ($m = 1$) Newtonova metoda je zadata formulom

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}, \qquad n = 0, 1, \ldots,$$

a aposteriorna ocena greške je:

$$|x^* - x^{(n)}| \leq \frac{M_2}{2m_1} |x^{(n)} - x^{(n-1)}|^2, \qquad n = 1, 2, \ldots,$$

gde je

$$M_2 = \max_{[a,b]} |f''(x)|, \qquad m_1 = \min_{[a,b]} |f'(x)|.$$

7.15 *Izračunati na četiri sigurne cifre sva rešenja jednačine*

$$6 \sin x = x^3.$$

Rešenje: Rešenja jednačine ćemo odrediti kao nule funkcije $f(x) = x^3 - 6 \sin x$. Očigledno je da je jedno rešenje $x_1^* = 0$. Kako je funkcija $f(x)$ neparna, možemo se ograničiti samo na nalaženje pozitivnih nula, tj. za $x > 0$. S obzirom da je $f'(x) = 3x^2 - 6 \cos x$ i $f''(x) = 6x + 6 \sin x$, to je $f''(x) > 0$ za svako $x > 0$, pa je $f'(x)$ rastuća funkcija. Dalje, približno je $f'(1.02) = 0$, odakle sledi da je $f'(x) > 0$ za $x \in (1.02, \infty)$ i $f'(x) < 0$ za $x \in (0, 1.02)$. Još je $f(0.1) < 0$, $f(1.02) < 0$ i $f(2) > 0$, te se jedino pozitivno rešenje jednačine nalazi u intervalu $(1.02, 2)$. Sa grafika se tačnije određuje da ono pripada intervalu $(1.8, 1.85)$. Rešenje se traži na četiri sigurne cifre, pa ga treba odrediti sa tačnošću $5 \cdot 10^{-4}$. Odredićemo ga Newtonovom metodom za $x^{(0)} = 1.85$. Kako je

$$M_2 = \max_{(1.8,1.85)} |f''(x)| = f''(1.85) = 16.87,$$

$$m_1 = \min_{(1.8,1.85)} |f'(x)| = f'(1.8) = 11,$$

tačnost će biti zadovoljena kada bude ispunjen uslov

$$|x^{(n)} - x^{(n-1)}| \leq \sqrt{\frac{2m_1}{M_2} \frac{1}{2} 10^{-3}} = 0.026.$$

U drugoj iteraciji taj uslov je zadovoljen. Dakle, sa traženom tačnošću sva realna rešenja jednačine su

$$x_1^* = 0, \qquad x_2^* = 1.801, \qquad x_3^* = -1.801.$$

7.16 *Odrediti na pet sigurnih cifara najveće po modulu rešenje jednačine*

$$15x - 10 \sinh x = 1.$$

Rešenje: Funkcija $f(x) = 15x - 10\sinh x - 1$ ima izvod $f'(x) = 15 - 10\cosh x$ i $f'(x) = 0$ za $\cosh x = 1.5$, tj. za $x = \pm 0.96$. U intervalima $(-\infty, -0.96)$ i $(0.96, \infty)$ funkcija $f(x)$ opada, a u intervalu $(-0.96, 0.96)$ raste. Kako je još

$$\lim_{x\to-\infty} f(x) > 0, \qquad f(-0.96) < 0, \qquad f(0.96) > 0, \qquad \lim_{x\to\infty} f(x) < 0,$$

jednačina ima tri realna rešenja, u svakom od intervala u kome $f'(x)$ ne menja znak po jedno. Unutar ovih intervala preciznije odredimo intervale u kojima se nalaze rešenja,

$$x_1^* \in (-1.8, -1.7), \qquad x_2^* \in (0, 0.5), \qquad x_1^* \in (1.5, 1.6).$$

Očigledno je da je najveće po modulu rešenje x_1^*. Izračunaćemo ga Newtonovom metodom polazeći od aproksimacije $x_1^{(0)} = -1.8$ (jer je $f(-1.8)f''(-1.8) > 0$). Kako je $f''(x) = -10\sinh x$ i $f'''(x) = -10\cosh x$, to je u intervalu $(-1.8, -1.7)$ $f'(x)$ negativna i rastuća, a $f''(x)$ pozitivna i opadajuća funkcija. Stoga je

$$M_2 = \max_{(-1.8,-1.7)} |f''(x)| = 29.4, \qquad m_1 = \min_{(-1.8,-1.7)} |f'(x)| = 13.3,$$

i računamo dok ne bude zadovoljen uslov

$$|x^{(n)} - x^{(n-1)}| \le \sqrt{\frac{2m_1}{M_2}\,\frac{1}{2}10^{-4}} = 0.67 \cdot 10^{-2}.$$

Primenom Newtonove metode u trećoj iteraciji dobijamo da je traženo rešenje

$$x_1^* = -1.7033.$$

7.17 *Odrediti sa tačnošću* $5 \cdot 10^{-6}$ *čisto imaginarne korene jednačine*

$$z\sin z + 1 = 0.$$

Rešenje: Imaginarni koren tražimo obliku $z = \imath\, y$, gde je y realan broj. Zamenom u jednačini dobijamo da je y realno rešenje jednačine

$$y\sinh y = 1,$$

jer je

$$\sin z = \frac{1}{2\imath}(\exp(\imath\, z) - \exp(-\imath\, z)) = \frac{1}{2\imath}(\exp(-y) - \exp(y)) = -\frac{1}{\imath}\sinh y.$$

Funkcija $f(y) = y\sinh y - 1$ je parna funkcija, pa je dovoljno naći njenu nulu za $y \ge 0$. Kako je ova funkcija monotono rastuća i $f(0) = -1$ a $f(1) = 0.1752$, tražena nula pripada intervalu $(0, 1)$. Newtonovom metodom, polazeći od aproksimacije $y^{(0)} = 1$, dobijamo u petoj iteraciji da je sa traženom tačnošću nula funkcije $f(y)$ $y^* = 0.93202$. Zbog parnosti funkcije $f(y)$ njena nula je i $-y^*$. Stoga su imaginarni koreni polazne jednačine

$$z^* = \pm 0.93202\imath.$$

7.18 *Funkcija y je data u implicitnom obliku*

$$y^3 + y + x = 0.$$

Sastaviti tabelu funkcije $y(x)$ na intervalu a) $[1, 2]$, b) $[2, 3]$, koja dozvoljava linearnu interpolaciju sa tačnošću 10^{-2}. Vrednosti funkcije izračunati sa tačnošću 10^{-4}.

Rešenje: Greška linearne interpolacije na intervalu $[a, b]$ se ocenjuje izrazom

$$|R_1(x)| \le \frac{1}{2}\,\frac{(b-a)^2}{4}\,M_2, \qquad M_2 = \max_{[a,b]} |y''(x)|.$$

Ako je dužina intervala $h = b - a$, onda je

$$(*) \qquad |R_1(x)| \le \frac{h^2}{8}\,M_2 \le 10^{-2} \qquad \text{za} \qquad h \le 10^{-1}\sqrt{\frac{8}{M_2}}.$$

Dakle, ostaje nam još da ocenimo konstantu M_2 u oba intervala.

Iz jednačine je $y(y^2 + 1) = -x$, pa je $y < 0$ za $x > 0$. Stoga diferenciranjem implicitne jednačine dobijamo da za $x > 0$

$$3y^2y' + y' + 1 = 0 \qquad \text{sledi} \qquad y' = -\frac{1}{3y^2 + 1} < 0,$$

i, ponovnim diferenciranjem

$$y'' = \frac{6y\,y'}{(3y^2 + 1)^2} = -\frac{6y}{(3y^2 + 1)^3} > 0.$$

Da bismo ocenili promenu funkcije y'' analizirajmo znak funkcije y''',

$$y''' = -6\frac{1 - 15y^2}{(3y^2 + 1)^4}\,y' = 6\frac{1 - 15y^2}{(3y^2 + 1)^5} = 6\frac{(1 - y\sqrt{15})(1 + y\sqrt{15})}{(3y^2 + 1)^5}.$$

Newtonovom metodom određujemo nulu funkcije $f(y) = y^3 + y + x$ za vrednosti parametra $x = 1$ i $x = 3$,

$$\begin{aligned} x = 1, &\qquad y : -0.5,\ -0.71,\ -0.6832,\ -0.6823,\ -0.6823, \\ x = 3, &\qquad y : -1.2,\ -1.2135,\ -1.2134,\ -1.2134, \end{aligned}$$

a očigledno je da je za $x = 2$ rešenje jednačine $y = -1$. Kako je funkcija $y(x)$ monotono opadajuća za $x > 0$, kada $x \in [1, 3]$ funkcija $y \in [-1.22, -0.68] \subset (-\infty, -1/\sqrt{15})$, a za te vrednosti y funkcija $y''' < 0$. Dakle, u intervalima u kojima rešavamo problem funkcija y'' je pozitivna i opadajuća, pa maksimalne vrednosti dostiže u levim krajevima intervala. Stoga je

$$\max_{[1,2]} |y''(x)| = y''(1) = 0.30, \qquad \max_{[2,3]} |y''(x)| = y''(2) = 0.094,$$

i, prema oceni $(*)$, maksimalni dopustivi korak za linearnu interpolaciju u ovim intervalima je

$$x \in [1,2] \quad h_1 \leq 0.51, \qquad\qquad x \in [2,3] \quad h_2 \leq 0.92.$$

U oba slučaja $h = 0.5$ je zadovoljavajuća vrednost koraka. Vrednosti $y(1.5)$ i $y(2.5)$ takođe određujemo Newtonovom metodom kao rešenje jednačine $f(y) \equiv y^3 + y + x = 0$

$$\begin{aligned} x &= 1.5, \qquad y : -1,\ -0.875,\ -0.8614,\ -0.8612,\ -0.8612, \\ x &= 2.5, \qquad y : -1.1,\ -1.1149,\ -1.1148,\ -1.1148. \end{aligned}$$

Tražena tabela je

x	1	1.5	2	2.5	3
$y(x)$	-0.6823	-0.8612	-1	-1.1148	-1.2134

7.19 *Za izračunavanje* $\sqrt[k]{a}$, $a > 0$, *definisana su dva iterativna algoritma*

$$\overline{x}_{n+1} = \frac{1}{k}\left(\frac{a}{\overline{x}_n^{k-1}} + (k-1)\overline{x}_n\right), \quad n = 0,1,\ldots, \qquad \overline{x}_0 > \sqrt[k]{a}, \quad (1)$$

$$\underline{x}_{n+1} = \frac{k\,a\,\underline{x}_n}{\underline{x}_n^k + (k-1)a}, \quad n = 0,1,\ldots, \qquad \underline{x}_0 < \sqrt[k]{a}, \qquad (2)$$

Dokazati da a) algoritmi konvergiraju na nekom odsečku koji sadrži tačku $\sqrt[k]{a}$; *b) važi da je* $\underline{x}_n \leq \sqrt[k]{a} \leq \overline{x}_n$, $n = 1,2,\ldots$.

Rešenje: Algoritam (1) dobijamo primenom Newtonove metode za rešavanje jednačine $f(x) \equiv x^k - a = 0$,

$$\overline{x}_{n+1} = \overline{x}_n - \frac{\overline{x}_n^k - a}{k\overline{x}_n^{k-1}}.$$

Kako je $f(\overline{x}_0) = \overline{x}_0^k - a > 0$ i $f''(\overline{x}_0) = k(k-1)\overline{x}_0^{k-2} > 0$ na osnovu pretpostavke zadatka, izbor početne aproksimacije zadovoljava uslov konvergencije algoritma. Indukcijom dokažimo da su i sve aproksimacije veće od traženog rešenja. Pretpostavimo da to važi za aproksimacije $\overline{x}_1,\ldots,\overline{x}_n$, pa je $f(\overline{x}_n) > 0$ i $f'(\overline{x}_n) = k\overline{x}_n^{k-1} > 0$ i $f''(\xi) > 0$ za $\xi \in [a^{1/k}, \overline{x}_n]$. Stoga iz razvoja

$$0 = f(\sqrt[k]{a}) = f(\overline{x}_n) + f'(\overline{x}_n)\,(\sqrt[k]{a} - \overline{x}_n) + \frac{1}{2}f''(\xi)(\sqrt[k]{a} - \overline{x}_n)^2$$

sledi da je

$$f(\overline{x}_n) + f'(\overline{x}_n)\,(\sqrt[k]{a} - \overline{x}_n) < 0,$$

tj.

$$\overline{x}_{n+1} = \overline{x}_n - \frac{f(\overline{x}_n)}{f'(\overline{x}_n)} > \sqrt[k]{a}.$$

Analogno, polazeći od funkcije $f(x) = a/x^{k-1} - x$, izvodi se algoritam (2) i dokazuje da su svi članovi niza aproksimacija određeni ovim algoritmom manji od $\sqrt[k]{a}$.

7.20 *a) Za koje početne vrednosti Newtonov iterativni algoritam za jednačinu* $f(x) \equiv x^3 - x = 0$ *nije definisan.*
b) Odrediti granične vrednosti niza aproksimacija određenih Newtonovim algoritmom za sledeće početne aproksimacije: 1) $x_0 = 0.447220$, *2)* $x_0 = 0.447210$, *3)* $x_0 = 0.447215$.

Rešenje: a) Pocetna aproksimacija u Newtonovom algoritmu treba da zadovoljava uslov

$$f(x_0)\, f''(x_0) = 6x_0^2(x_0^2 - 1) > 0, \qquad \text{tj.} \qquad |x_0| > 1.$$

b) Rešenja jednačine $f(x) = 0$ su -1, 0 i 1. Aproksimacije izračunate Newtonovim algoritmom

$$x_{n+1} = x_n - (x_n^3 - x_n)/(3x_n^2 - 1), \qquad n = 0, 1, \ldots,$$

za date pocetne vrednosti su

$$x_n^{(1)} = 0.447220, -0.447252, 0.447444, -0.448600, 0.455631, -0.501528, 1.028082, \\ 1.001110, 1.000002, \underline{1.000000},$$

$$x_n^{(2)} = 0.447210, -0.447192, 0.447084, -0.446438, 0.442590, -0.420510, 0.316746, \\ -0.090924, 0.001542, \underline{0.000000},$$

$$x_n^{(3)} = 0.447215, -0.447222, 0.447264, -0.447517, 0.449039, -0.458338, 0.520772, \\ -1.515492, -1.181856, -1.034869, -1.001687, -1.000004, \underline{-1.000000},$$

Ovakvo ponašanje nizova je uzrokovano tačkama ekstrema $x = \pm 1/\sqrt{3}$. Naime, kada je aproksimacija u okolini jedne od ovih tačaka, očigledno je da dolazi do velike promene u narednoj aproksimaciji.

7.21 *Newton-ovom modifikovanom metodom naći rešenje sistema:*

$$x = \log \frac{y}{z} + 1 \qquad y = 0.4 + z^2 - 2x^2 \qquad z = 2 + \frac{xy}{20}$$

u okolini tačke (1, 2.2, 2).

Rešenje: Newton-ova modifikovana metoda je definisana formulom

$$\begin{pmatrix} x^{(n+1)} \\ y^{(n+1)} \\ z^{(n+1)} \end{pmatrix} = \begin{pmatrix} x^{(n)} \\ y^{(n)} \\ z^{(n)} \end{pmatrix} - [J(x^{(0)}, y^{(0)}, z^{(0)})]^{-1} \begin{pmatrix} f(x^{(n)}, y^{(n)}, z^{(n)}) \\ g(x^{(n)}, y^{(n)}, z^{(n)}) \\ h(x^{(n)}, y^{(n)}, z^{(n)}) \end{pmatrix}, \\ n = 0, 1, \ldots,$$

gde je

$$f(x,y,z) = \log\frac{y}{z} + 1 - x$$
$$g(x,y,z) = 0.4 + z^2 - 2x^2 - y \qquad J(x,y,z) = \begin{pmatrix} -1 & \frac{1}{y\ln 10} & -\frac{1}{z\ln 10} \\ -4x & -1 & 2z \\ \frac{y}{20} & \frac{x}{20} & -1 \end{pmatrix}$$
$$h(x,y,z) = 2 + \frac{xy}{20} - z$$

Primenom navedene formule polazeći od date početne aproksimacije dobijaju se vrednosti

$$\left(x^{(1)}, y^{(1)}, z^{(1)}\right) = (1.0900,\ 2.5998,\ 2.1399),$$
$$\left(x^{(2)}, y^{(2)}, z^{(2)}\right) = (1.0882,\ 2.6209,\ 2.1426),\ \ldots$$

7.22 *Sa tačnošću* $5 \cdot 10^{-4}$ *naći u oblasti* $D = \{(x,y,z) | x > 0,\ y > 0,\ z > 0\}$ *sva rešenja sistema jednačina*

$$x + y + z = 1, \qquad x^2 + y^2 + z^2 = 0.5, \qquad z - y^2 = 0.$$

Rešenje: Odredimo prvo broj rešenja datog sistema u prvom oktantu. Eliminisanjem promenljivih x i z iz druge jednačine pomoću prve i treće, dobijamo polinom četvrtog stepena po promenljivoj y,

$$2y^4 + 2y^3 - 2y + 0.5 = 0,$$

koji ima dva realna pozitivna korena i par kompleksnih korena. Realnim korenima odgovaraju dva rešenja sistema u prvom oktantu. Odredimo ih Newtonovom metodom definisanom formulom

$$\begin{pmatrix} x^{(n+1)} \\ y^{(n+1)} \\ z^{(n+1)} \end{pmatrix} = \begin{pmatrix} x^{(n)} \\ y^{(n)} \\ z^{(n)} \end{pmatrix} - [J(x^{(n)}, y^{(n)}, z^{(n)})]^{-1} \begin{pmatrix} f(x^{(n)}, y^{(n)}, z^{(n)}) \\ g(x^{(n)}, y^{(n)}, z^{(n)}) \\ h(x^{(n)}, y^{(n)}, z^{(n)}) \end{pmatrix}, \qquad n = 0, 1, \ldots,$$

gde je

$$f(x,y,z) = x + y + z - 1,$$
$$g(x,y,z) = x^2 + y^2 + z^2 - 0.5, \qquad J(x,y,z) = \begin{pmatrix} 1 & 1 & 1 \\ 2x & 2y & 2z \\ 0 & -2y & 1 \end{pmatrix}.$$
$$h(x,y,z) = z - y^2,$$

Sa traženom tačnošću, polazeći od aproksimacije $(0,\ 0.5,\ 0.3)^\top$, dobijamo u trećoj iteraciji prvo rešenje

$$x_1^* = 0.030, \qquad y_1^* = 0.605, \qquad z_1^* = 0.366,$$

a polazeći od aproksimacije $(0.5,\ 0.3,\ 0)^\top$ dobijamo u četvrtoj iteraciji drugo rešenje

$$x_2^* = 0.646, \qquad y_2^* = 0.277, \qquad z_2^* = 0.077.$$

Napomena: Broj rešenja sistema u prvom oktantu se može dobiti analizom grafika $f = 0$, $g = 0$ i $h = 0$. Presek ravni $f = 0$ i lopte $g = 0$ je krug u prvom kvadrantu. Presek parabole $h = 0$ i ravni $f = 0$ je prava, koja seče pomenuti krug u dve tačke. Te tačke predstavljaju tražena rešenja.

7.23 *Sa tačnošću* $5 \cdot 10^{-6}$ *odrediti sva rešenja sistema*

$$(x-2)^2 + (y-2)^2 = 1, \qquad \tan(xy) = \tan 1.$$

Rešenje: Zbog periodičnosti funkcije $\tan x$ imamo da

$$\tan(xy) = \tan 1 \qquad \Longrightarrow \qquad xy = 1 + k\pi, \quad k = 0, \pm 1, \pm 2, \ldots.$$

Iz prve jednačine sistema je

$$x^2 - 4x + 4 + y^2 - 4y + 4 = 1, \qquad \text{tj.} \qquad (x+y)^2 - 4(x+y) + 7 - 2xy = 0,$$

pa na osnovu druge jednačine za rešenje sistema važi

$$(x+y)^2 - 4(x+y) + 5 - 2k\pi = 0 \qquad \text{tj.} \qquad (x+y)_{1/2} = 2 \pm \sqrt{2k\pi - 1}.$$

Kako rešenja pripadaju krugu definisanom prvom jednačinom, to je $2 < x + y < 6$, što je moguće samo za $k = 1$ i $k = 2$. Stoga sistem ima četiri rešenja koja su simetrična u odnosu na pravu $y = x$ jer su obe funkcije sistema simetrične u odnosu na tu pravu. Dovoljno je odrediti samo dva od njih.

Odredićemo ih Newtonovom metodom. Rezultati izračunavanja su dati sledećom tabelom

n	0	1	2	3
$x_1^{(n)}$	1.4	1.457557	1.458083	1.458085
$y_1^{(n)}$	2.8	2.843167	2.840436	2.840433
$x_2^{(n)}$	2.6	2.603747	2.603804	2.603804
$y_2^{(n)}$	2.8	2.797189	2.797133	2.797133

gde je n indeks iteracije, (x_1, y_1) prvo a (x_2, y_2) drugo rešenje. Dakle, sa traženom tačnošću rešenja sistema su

$$(x_1^*, y_1^*) = (1.45808, 2.84043), \qquad (x_2^*, y_2^*) = (2.60380, 2.79713),$$
$$(x_3^*, y_3^*) = (2.84043, 1.45808), \qquad (x_4^*, y_4^*) = (2.79713, 2.60380).$$

7.24 *Newtonovom metodom sa tačnošću* $5 \cdot 10^{-5}$ *naći sva rešenja sistema*

$$\sin(x+y) = 1.5x, \qquad x^2 + y^2 = 1.$$

Rešenje: Newtonovom metodom rešavamo sistem

$$f(x,y) \equiv \sin(x+y) - 1.5x = 0, \qquad g(x,y) \equiv x^2 + y^2 - 1 = 0.$$

S obzirom da je $f(-x,-y) = -f(x,y)$ i $g(-x,-y) = g(x,y)$, to ako je tačka (x,y) rešenje sistema, onda je i $(-x,-y)$ takođe rešenje. Nula funkcije $g(x,y)$ pripada oblasti $R = \{(x,y) \,|\, 0 \le |x|, |y| \le 1\}$ (jediničnom kvadratu sa centrom u koordinatnom početku). U okolini nule je $\sin(x+y) \approx x+y$, te je $f(x,y) \approx x+y-1.5x$. Stoga je $f \approx 0$ za $y = 0.5x$, te za početnu aproksimaciju možemo uzeti da je $y = 0.5x$. Zamenom u drugoj jednačini sistema dobijamo da je $x = 0.8944$ i $y = 0.4479$, tj. približno $x_1^{(0)} = 0.8$ i $y_1^{(0)} = 0.5$. Popravke rešenja određujemo kao rešenja sistema linearnih jednačina

$$J(x_1^{(n)}, y_1^{(n)}) \begin{pmatrix} \Delta x_1^{(n)} \\ \Delta y_1^{(n)} \end{pmatrix} = \begin{pmatrix} f(x_1^{(n)}, y_1^{(n)}) \\ g(x_1^{(n)}, y_1^{(n)}) \end{pmatrix}, \qquad n = 0, 1, \ldots,$$

gde je

$$\Delta x_1^{(n)} = x_1^{(n+1)} - x_1^{(n)}$$
$$\Delta y_1^{(n)} = y_1^{(n+1)} - y_1^{(n)}$$

i Jakobijan

$$J(x,y) = \begin{pmatrix} \cos(x+y) - 1.5 & \cos(x+y) \\ 2x & 2y \end{pmatrix}$$

Tako dobijamo aproksimacije rešenja

n	0	1	2	3	4
$x_1^{(n)}$	0.8	0.67533	0.66012	0.65818	0.65818
$y_1^{(n)}$	0.5	0.80947	0.75340	0.75286	0.75286

S obzirom na već pomenutu simetriju jednačina sistema, rešenja su sa traženom tačnošću

$$(x_1^*, y_1^*) = (0.6582, 0.7529), \qquad (x_2^*, y_2^*) = (-0.6582, -0.7529).$$

7.25 *Odrediti sa tačnošću* 10^{-4} *u prvom kvadrantu sva rešenja sistema jednačina*

$$x^2 + 20x + y^2 = 1 \qquad\qquad y = 0.5x + \sin xy.$$

Rešenje: U prvom kvadrantu sistem ima samo jedno rešenje. Rezultati izračunavanja Newtonovom metodom dati su u tabeli

n	0	1	2	3
$x^{(n)}$	0	0.05	0.04984	0.04984
$y^{(n)}$	0	0.025	0.02623	0.02623

Dakle, rešenje sa traženom tačnošću je

$$(x^*, y^*) = (0.0498, 0.0262).$$

7.26 *Odrediti na četiri sigurne cifre u prvom kvadrantu sva rešenja sistema jednačina*

$$x^3 + y^3 = 1 \qquad x^2 + y^2 = 2x.$$

Rešenje: U prvom kvadrantu postoji samo jedno rešenje sistema. Rezultati izračunavanja Newtonovom metodom dati su u tabeli

n	0	1	2	3
$x^{(n)}$	0.6	0.60671	0.60646	0.60646
$y^{(n)}$	0.9	0.91965	0.91931	0.91931

Dakle, rešenje sa traženom tačnošću je

$$(x^*, y^*) = (0.6065, 0.9193).$$

7.27 *Odrediti sa tačnošću* $5 \cdot 10^{-6}$ *u prvom kvadrantu rešenje sistema jednačina*

$$x + y + 0.1 \sin xy = 1 \qquad y - 0.1 \exp(x + y) = 0.7.$$

Rešenje: Rezultati izračunavanja Newtonovom metodom dati su u tabeli

n	0	1	2	3
$x^{(n)}$	0	0.026260	0.026311	0.026311
$y^{(n)}$	1	0.971114	0.971134	0.971134

Dakle, rešenje sa traženom tačnošću je

$$(x^*, y^*) = (0.02631, 0.97113).$$

7.28 *Odrediti na pet sigurnih cifara rešenje sistema nelinearnih jednačina*

$$\sin x \sinh y = 0.2, \qquad \cos x \cosh y = 1.2$$

najbliže koordinatnom početku.

Rešenje: Rezultati izračunavanja Newtonovom metodom dati su u tabeli

n	0	1	2	3	4
$x^{(n)}$	0.2	0.290097	0.273169	0.272450	0.272449
$y^{(n)}$	0.5	0.725246	0.688767	0.687734	0.687733

Dakle, rešenje sa traženom tačnošću je

$$(x^*, y^*) = (0.27245, 0.68773).$$

7.29 *Odrediti sa tačnošću* $5 \cdot 10^{-6}$ *sva rešenja sistema jednačina*

$$\tan xy = x^2 \qquad \frac{1}{2}x^2 + 2y^2 = 1.$$

Rešenje: Dva rešenja su očigledna. Za $x = 0$ prva jednačina je zadovoljena za svako y, a druga za $y = \pm 1/\sqrt{2}$. Treće rešenje je izračunato Newtonovom metodom

n	0	1	2	3	4
$x_3^{(n)}$	0.5	0.675180	0.662791	0.662932	0.662932
$y_3^{(n)}$	0.5	0.643705	0.624954	0.624604	0.624604

a četvrto je očigledno $(-x_3, -y_3)$. Dakle, dati sistem ima četiri rešenja i ona su sa traženom tačnošću

$$(x_1^*, y_1^*) = (0, 0.70711), \qquad (x_2^*, y_2^*) = (0, -0.70711),$$
$$(x_3^*, y_3^*) = (0.66293, 0.62460), \qquad (x_4^*, y_4^*) = (-0.66293, -0.62460).$$

7.30 *U oblasti* $D = \{(x, y) | \, x \geq 0, \, y \geq 0\}$ *odrediti sa tačnošću* $5 \cdot 10^{-4}$ *sva rešenja sistema jednačina*

$$|y^2 - 4| = 5 - x^2 \qquad x^{2/3} + y^{2/3} = 9^{1/3}.$$

Rešenje: Jedno rešenje je očigledno, $x = 0$, $y = 3$. Drugo rešenje je izračunato Newtonovom metodom

n	0	1	2	3
$x_2^{(n)}$	1.3	1.3027	1.3031	1.3031
$y_2^{(n)}$	0.8	0.8356	0.8355	0.8355

Dakle, sa traženom tačnošću rešenja datog sistema u prvom kvadrantu su

$$(x_1^*, y_1^*) = (0, 3), \qquad (x_2^*, y_2^*) = (1.303, \, 0.836).$$

7.31 *Sa tačnošću* $5 \cdot 10^{-5}$ *naći koren jednačine*

$$\exp z = z, \qquad z = x + iy,$$

sa pozitivnim imaginarnim delom.

Rešenje: Izjednačavanjem realnih i imaginarnih delova leve i desne strane date jednačine, ova se svodi na sistem jednačina po realnim promenljivim x i y

$$\exp x \cos y = x, \qquad \exp x \sin y = y.$$

Transformišimo ga u pogodniji oblik. Kvadriranjem obe jednačine i sabiranjem dobijenih izraza dobijamo prvu jednačinu novog sistema, a drugu eliminisanjem $\exp x$ iz prethodne dve jednačine:

$$f(x,y) = x^2 + y^2 - \exp(2x) = 0, \qquad g(x,y) = x\sin y - y\cos y = 0.$$

Jakobijan preslikavanja definisanog funkcijama f i g je

$$J(x,y) = \begin{pmatrix} 2(x - \exp(2x)) & 2y \\ \sin y & (x-1)\cos y + y\sin y \end{pmatrix}.$$

Sistem rešavamo metodom Newtona sa početnom aproksimacijom rešenja $(0.3, 1.3)$ (određenom sa grafika), i dobijamo sledeće aproksimacije

n	0	1	2	3
$x^{(n)}$	0.3	0.31873	0.31813	0.31813
$y^{(n)}$	1.3	1.33814	1.33723	1.33723

Stoga je sa tačnošću $5 \cdot 10^{-5}$ traženi koren date jednačine

$$z = 0.3181 + 1.3372\imath.$$

7.32 *Sa tačnošću $5 \cdot 10^{-5}$ naći sva rešenja sistema*

$$y(x-1) = 1, \qquad x^2 = y^2 + 1.$$

Rešenje: Sistem ćemo rešavati Newtonovom modifikovanom metodom (optimalnom iterativnom metodom). Grafik prve funkcije je hiperbola sa osama $x = 1$ i $y = 0$, a druge hiperbola sa osama $y = \pm x$. Stoga sistem ima dva rešenja, prvo u okolini tačke $(1.7, 1.5)$ i drugo u okolini tačke $(-1, -0.5)$. Računajući po rekurentnoj formuli

$$\begin{pmatrix} x^{(n+1)} \\ y^{(n+1)} \end{pmatrix} = \begin{pmatrix} x^{(n)} \\ y^{(n)} \end{pmatrix} - J(x^{(0)}, y^{(0)}) \begin{pmatrix} f(x^{(n)}, y^{(n)}) \\ g(x^{(n)}, y^{(n)}) \end{pmatrix} \quad n = 0, 1, \ldots,$$

gde je

$$f(x,y) = y(x-1) - 1, \quad g(x,y) = x^2 - y^2 - 1, \qquad J(x,y) = \begin{pmatrix} y & x-1 \\ 2x & -2y \end{pmatrix}$$

dobijamo aproksimacije rešenja

n	0	1	2	3	4
$x_1^{(n)}$	1.7	1.71483	1.71655	1.71667	1.71667
$y_1^{(n)}$	1.5	1.39680	1.39528	1.39533	1.39534
$x_2^{(n)}$	-1.0	-1.11111	-1.10665	-1.10693	-1.10692
$y_2^{(n)}$	-0.5	-0.47222	-0.47488	-0.47461	-0.47463

gde je n indeks iteracije, (x_1, y_1) prvo a (x_2, y_2) drugo rešenje. Dakle, sa traženom tačnošću rešenja sistema su

$$(x_1^*, y_1^*) = (1.7167, 1.3953) \qquad (x_2^*, y_2^*) = (-1.1069, -0.4746).$$

7.33 *a) Pokazati da je iterativni algoritam*

$$B_{i+1} = B_i\,(2I - A\,B_i), \quad i = 1, 2, \ldots, \qquad B_1 \text{ proizvoljno}$$

gde je I jedinična matrica, definisan za nalaženje matrice A^{-1}, matrični analog Newton-Raphsonove metode za nalaženje recipročne vrednosti broja.
b) Ako je $C_i = I - A\,B_i$, dokazati da je $C_i = {C_1}^{2^{i-1}}$.
c) Na osnovu prethodnog dokazati da je potreban i dovoljan uslov za konvergenciju metode definisane u tački (a) da su sve sopstvene vrednosti matrice C_1 unutar jediničnog kruga .

Rešenje: a) Ako je $b = a^{-1}$, gde je a realan broj, broj b možemo odrediti kao rešenje jednačine $f(x) \equiv (ax)^{-1} - 1 = 0$. Newtonova metoda primenjena na ovu funkciju daje iterativni algoritam

$$b_{i+1} = b_i - f(b_i)/f'(b_i) = 2b_i - a\,b_i^2, \qquad i = 1, 2, \ldots$$

što je analog datom matričnom iterativnom algoritmu.

b) Prema datom algoritmu je $B_{i+1} = B_i(I + C_i)$ pa je

$$\begin{aligned} C_i &= I - A\,B_i = I - A\,B_{i-1}(I + C_{i-1}) = C_{i-1} - A\,B_{i-1}C_{i-1} = C_{i-1}^2 \\ &= {C_{i-2}}^{2^2} = \cdots = {C_1}^{2^{i-1}} \end{aligned}$$

c) Prema dokazanom u tački (b) je

$$\begin{aligned} B_{i+1} &= B_i(I + C_i) = B_{i-1}(I + C_{i-1})\,(I + C_i) = \ldots \\ &= B_1(I + C_1)\,(I + C_2)\cdots(I + C_i) \\ &= B_1(I + C_1)\,(I + C_1^2)\cdots(I + {C_1}^{2^{i-1}}) \\ &= B_1(I - C_1)^{-1}(I - C_1)(I + C_1)(I + C_1^2)\cdots(I + {C_1}^{2^{i-1}}) \\ &= B_1(A\,B_1)^{-1}(I - {C_1}^{2^2})(I + {C_1}^{2^2})\cdots(I + {C_1}^{2^{i-1}}) = A^{-1}(I - {C_1}^{2^i}) \end{aligned}$$

tj.

$$(*) \qquad A^{-1} - B_{i+1} = A^{-1}C_1^{2^i} = A^{-1}C_{i+1}.$$

Ako sopstvene vrednosti matrice C_1 zadovoljavaju uslov $|\lambda(C_1)| < 1$, onda to važi i za sopstvene vrednosti matrice C_i jer je $\lambda(C_i) = \big(\lambda(C_1)\big)^{2^{i-1}}$. Odatle sledi da je $\lim_{i\to\infty} C_i = 0$, pa je na osnovu $(*)$ $\lim_{i\to\infty} B_i = A^{-1}$.

MATLAB

7.34 *Korišćenjem* MATLAB-*a grafički ilustrovati primenu Newton-ove metode tangente za nalaženje nule funkcije*

$$f(x) = \begin{cases} \sqrt{x} & x \geq 0 \\ -\sqrt{-x} & x < 0 \end{cases}$$

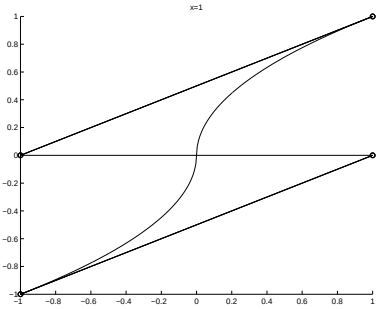
u intervalu $[-1, 1]$, *pri čemu se za početnu iteraciju uzima tačka* $x_0 = 1$.

Rešenje: Skript koji grafički ilustruje primenu Newton-ove metode je veoma sličan skriptu koji ilustruje metodu iteracije i dat je u nastavku.

```
% Konstruisemo inline objekat koji odgovara funkciji f
f = inline('sign(x).*sqrt(abs(x))');
% Konstruisemo inline objekat koji odgovara izvodu funkcije f
fp = inline('1/(2*sqrt(abs(x)))');

% Postavljamo krajeve intervala i pocetnu iteraciju
a = -1; b = 1; x0 = 1;

% Crtamo funkciju f na intervalu [a,b]
t  = linspace(a, b, 250); ft = feval(f, t);
clf, hold on, plot(t, ft, 'g');
% Crtamo x osu
plot([a b], [0 0], 'k');

% x i y=f(x) sadrze koordinate tacke tekuce iteracije
% Inicijalizujemo ih na pocetnu iteraciju
x = x0; y = feval(f,x0);

while(1)
    % Pamtimo vrednosti prethodne iteracije
    xp = x; yp = y;
    % Racunamo vrednost izvoda f' u prethodnoj iteraciji
    dyp = feval(fp, xp);
    if (dyp == 0)
        error('Tangenta nema presek sa x osom!');
    end

    % Izracunavamo tacku sledece iteraciju
    x = xp - yp/dyp;  y = feval(f, x);

    % Iscrtavamo tangentu i karakteristicne tacke na njoj
    plot([xp x], [yp 0],'r'); plot(xp, yp, 'o'); plot(x, 0, 'o')
    title(['x=' num2str(x,16)]); pause

    % Ukoliko je postignuta zeljena tacnost, zavrsavamo
    if abs(xp-x)<epsilon
        break;
    end
end
```

Primetimo kako se funkcija $f(x)$ može zapisati u obliku $f(x) = \text{sign}(x)\sqrt{|x|}$ i njen izvod je $\frac{1}{2\sqrt{|x|}}$.

Prilikom pokretanja skripta, primećuje se da metoda ne konvergira, što je i ilustrovano na slici 7.2. Razlog tome je promena znaka drugog izvoda funkcije u njenoj nuli, što dovodi do oscilovanja iteracija između tačaka x_0 i $-x_0$.



Slika 7.3: Newton-ova metoda primenjena na funkciju $f(x) = \text{sign}(x)\sqrt{|x|}$.

Funkcija očigledno ima nulu $x = 0$ i svi uslovi teoreme o konvergenciji Newton-ove metode su ispunjeni, osim uslova da drugi izvod ne menja znak na intervalu u kome je nula lokalizovana. Drugi izvod funkcije f je $f''(x) = \frac{\text{-sign}(x)}{4\sqrt{|x|^3}}$, i on menja znak na intervalu $[-1, 1]$.

7.35 *Napisati* MATLAB *funkciju*

```
function x = r2_newton(f, x0, epsilon)
```

koja Newton-ovom metodom pronalazi nulu funkcije $f : \mathbb{R}^2 \to \mathbb{R}$.

Korišćenjem ove funkcije, sa tačnošću 10^{-5} *naći Newton-ovom metodom sva rešenja sistema jednačina*

$$x^3 - y^3 - x = 0, \qquad x^3 + y^3 - 3xy = 0.$$

u oblasti $D = \{(x, y) | \, x < 0, \, y > 0\}$.

Rešenje: Pošto se kao argument funkciji `r2_newton` zadaje isključivo funkcija f, a ne i njeni izvodi, javlja se problem pronalaženja jakobijana funkcije f. Jedan od načina da se ovo uradi je korišćenje numeričkog diferenciranja. Prilikom implementacije, odlučeno je da umesto numeričkog diferenciranja iskoristimo mogućnosti MATLAB-ovog modula za simbolička izračunavanja (*Symoblic toolbox*-a). U tom slučaju diferenciranje vršimo funkcijom `diff`, dok izračunavanje vrednosti funkcije dve promenljive $f(x, y)$ u tački (x_t, y_t) vršimo pozivom `subs(f, {'x', 'y'}, {xt, yt})`.

Korišćenje modula za simbolička izračunavanja smanjuje brzinu izvršavanja programa, ali smanjuje numeričku grešku izbegavanjem korišćenja numeričkog diferenciranja. Sa druge strane, korisnik je oslobađen potrebe za ručnim izračunavanjem jakobijana čime je automatizovan veći deo metode.

```
% Funkcija vrsi Newton-ov iterativni algoritam i pokusava da
% pronadje nulu funkcije, f:R^2->R sa datom tacnoscu.
% Argumenti funkcije su :
%       f  - simbolicki 2x1 vektor koji predstavlja funkciju f(x, y)
%       x0 - 2x1 vektor koji sadrzi pocetnu iteraciju
%       epsilon - tacnost
%       maxiter - maksimalni dozvoljeni broj iteracija
% Funkciji nedostaju provere korektnosti
function x = r2_newton(f, x0, epsilon, maxiter)

% Odredjujemo inverznu matricu Jakobijana funkcije f
iJ = inv([diff(f, 'x') diff(f, 'y')]);

% Postavljamo tekucu iteraciju na pocetnu
x = x0; iter = 0;
while(1)
    % Proveravamo da li smo presli dozvoljen broj iteracija
    iter = iter + 1;
    if (iter >= maxiter) error('Metoda ne konvergira'); end

    % Pamtimo prethodnu iteraciju
    xp = x;

    % Odredjujemo sledecu iteraciju po Newton-ovoj formuli
    x = xp - subs(iJ, {'x', 'y'}, {xp(1), xp(2)})* ...
             subs(f,  {'x', 'y'}, {xp(1), xp(2)})

    % Ukoliko je zadovoljen kriterijum tacnosti, prekidamo iteriranje
    if (norm(x-xp)<epsilon)
        break;
    end
end
```

Primetimo kako je jakobijan simbolički invertovan pre početka izvođenja iterativnog postupka, čime je izbegnuta potreba za invertovanjem njegove vrednosti u svakoj iteraciji.

Transformacijom polaznog sistema (sabiranjem i oduzimanjem jednačina) dobijamo funkcije

$$y = \frac{1}{3}(2x^2 - 1), \qquad x = \frac{2y^3}{3y - 1},$$

čiji grafici u oblasti D imaju samo jednu presečnu tačku, koja je u okolini tačke $(-1, 0.3)$.

```
% Funkcija ciju nulu trazimo
f = [sym('1/3*(2*x^2-1)-y');
     sym('2*y^3/(3*y-1)-x')];

% Pocetna iteracija, tacnost i maksimalni broj iteracija
x0 = [-1; 0.3]; epsilon = 1e-5; maxiter = 10;

% Poziv iterativnog algoritma
r2_newton(f, x0, epsilon, maxiter);
```

Pokretanjem skripta, dobijamo da je sa traženom tačnošću rešenje sistema

$$(x^*, y^*) = (-0.98436, 0.31264).$$

7.3 Metoda Bairstow-a

Metodom Bairstow-a se nalazi kvadratni činilac polinoma

$$\begin{aligned} P_m(x) &= x^m + a_1 x^{m-1} + \cdots + a_{m-1}x + a_m \\ &= (x^2 + px + q)\,(x^{m-2} + b_1 x^{m-3} + \cdots + b_{m-3}x + b_{m-2}) \end{aligned}$$

Koeficijenti p i q kvadratnog faktora i polinoma količnika b_k, $k = 1, \ldots, m-2$ se određuju iterativnim algoritmom

$$b_{-1}^{(n)} = 0, \quad b_0^{(n)} = 1, \qquad b_k^{(n)} = a_k - p^{(n)} b_{k-1}^{(n)} - q^{(n)} b_{k-2}^{(n)}, \quad k = 1, \ldots, m,$$

$$c_{-1}^{(n)} = 0, \quad c_0^{(n)} = 1, \qquad c_k^{(n)} = b_k^{(n)} - p^{(n)} c_{k-1}^{(n)} - q^{(n)} c_{k-2}^{(n)}, \quad k = 1, \ldots, m-2,$$
$$n = 0, 1, \ldots,$$

Popravke vrednosti koeficijenata

$$\Delta p^{(n)} = p^{(n+1)} - p^{(n)}, \qquad \Delta q^{(n+1)} = q^{(n+1)} - q^{(n)}$$

na svakom koraku se određuju kao rešenja sistema jednačina

$$\begin{aligned} c_{m-2}^{(n)} \Delta p^{(n)} + c_{m-3}^{(n)} \Delta q^{(n)} &= b_{m-1}^{(n)} \\ -(p^{(n)} c_{m-2}^{(n)} + q^{(n)} c_{m-3}^{(n)}) \Delta p^{(n)} + c_{m-2}^{(n)} \Delta q^{(n)} &= b_m^{(n)}, \end{aligned}$$

Za početne aproksimacije koeficijenata se obično uzimaju vrednosti

$$p^{(0)} = q^{(0)} = 0, \qquad \text{ili} \qquad p^{(0)} = -(\alpha_1 + \alpha_2), \quad q^{(0)} = \alpha_1 \alpha_2,$$

gde su α_1 i α_2 približne vrednosti korena polinoma, ako su poznate.

7.36 *Metodom Bairstow-a odrediti korene polinoma*

$$P_4(x) = x^4 - 10x^3 + 34x^2 - 50x + 25.$$

Za početnu aproksimaciju koeficijenata kvadratnog faktora uzeti $p_0 = -5.8$, $q_0 = 5$.

Rešenje: Rezultati izračunavanja dati su u sledećoj tabeli

p		-5.8		-6.0220		-5.9996		-6.0000
q		5		4.9324		4.9942		5.0000
k	a	b	c	b	c	b	c	b
0	1	1	1	1	1	1	1	1
1	−10	−4.2	1.6	−3.9780	2.0440	−4.0004	1.9992	−4.0000
2	34	4.64	8.92	5.1120	12.4882	5.0050	12.0052	5.0000
3	−50	−2.088		0.4055		0.0068		0
4	25	−10.3104		2.2275		0.0449		0
Δp		−0.2220		0.0224		−0.0004		
Δq		−0.0676		0.0617		0.0058		

Polinom se stoga može faktorizovati u obliku

$$P_4(x) = (x^2 - 6x + 5)\,(x^2 - 4x + 5),$$

pa su koreni polinoma

$$x_1^* = 5, \qquad x_2^* = 1, \qquad x_{3/4}^* = 2 \pm \imath.$$

7.37 *Metodom Bairstow-a sa tačnošću* $5 \cdot 10^{-4}$ *odrediti korene polinoma*

$$P_3(x) = x^3 - x - 1.$$

Za početne aproksimacije korena kvadratnog faktora uzeti $p_0 = q_0 = 1$.

Rešenje:

$$P_3(x) = (x^2 + 1.3248x + 0.7549)\,(x - 1.3247)$$
$$x_1 = 1.3247, \qquad x_{2/3} = 0.6624 \pm 0.5623\imath.$$

7.38 *Metodom Bairstow-a sa tačnošću* $5 \cdot 10^{-5}$ *odrediti korene polinoma*

$$P_4(x) = x^4 - 8x^3 + 39x^2 - 62x + 50.$$

Za početne aproksimacije korena kvadratnog faktora uzeti $p_0 = q_0 = 0$.

Rešenje:

$$P_4(x) = (x^2 - 2x + 2)\,(x^2 - 6x + 25) \qquad x_{1/2}^* = 1 \pm \imath, \quad x_{3/4}^* = 3 \pm 4\imath.$$

7.39 *Metodom Bairstow-a sa tačnošću* $5 \cdot 10^{-5}$ *odrediti korene polinoma*

$$P_4(x) = x^4 - 3x^3 - 54x^2 - 150x - 100.$$

Za početne aproksimacije korena kvadratnog faktora uzeti $p_0 = q_0 = -9$.

Rešenje:

$$P_4(x) = (x^2 - 9x - 10)\,(x^2 + 6x + 10) \qquad x_1^* = -1, \quad x_2^* = 10, \quad x_{3/4}^* = -3 \pm \imath.$$

7.40 *Metodom Bairstow-a sa tačnošću* $5 \cdot 10^{-4}$ *odrediti korene polinoma*

$$P_4(x) = 4x^4 + 14x^3 + 14x^2 + 13x - 3.$$

Za početne aproksimacije korena kvadratnog faktora uzeti $p_0 = q_0 = 1.25$.

Rešenje: Kako koeficijent polinoma uz najviši stepen nije jednak 1, polinom se prvo podeli tim koeficijentom (ovde 4), i zatim primeni opisani algoritam. Tako se dobija da je

$$P_4(x) = 4(x^2 + x + 1.5)\,(x^2 + 2.5x - 0.5),$$

$$x_1^* = -2.6861, \qquad x_2^* = 0.1861, \qquad x_{3/4}^* = -0.5 \pm 1.1180\imath.$$

7.41 *Metodom Bairstow-a sa tačnošću* 10^{-3} *odrediti korene polinoma*

$$P_4(x) = 2x^4 - 10x^3 + 19x^2 - 14x + 5.$$

Rešenje:

$$P_4(x) = 2\,(x^2 - 4x + 5)\,(x^2 - x + 0.5) \qquad x_{1/2}^* = 2 \pm \imath, \quad x_{3/4}^* = \frac{1}{2}(1 \pm \imath).$$

FORTRAN Procedura koja implementira metodu Bairstow-a se sastoji od prostog prevodjenja u kod formula koje opisuju metodu. Treba medjutim uočiti kako je, u cilju unapredjivanja prostorne efikasnosti rešenja, tokom izračunavanja koeficijenata b_k i c_k dovoljno održavati samo njihove dve poslednje vrednosti koje su u kodu označene sa `b_prev` i `b_curr`, odnosno `c_prev` i `c_curr`. Procedura tako ima sledeći oblik.

```
module bairstow_module
  implicit none
  public bairstow

contains
  ! Procedura bairstow() izdvaja kvadratni faktor iz datog polinoma i
  ! odredjuje nule polinoma. Argumenti procedure su:
  !   n - stepen polinoma
  !   a - polje sa koeficijentima polinoma (pocev od koeficijenta uz
  !       najvisi stepen)
  !  p, q - pocetne aproksimacije za koeficijente kvadratnog faktora
  !  epsilon - trazena tacnost
  !  x0, x1 - promenljive u koje treba smestiti nule kvadratnog faktora
  ! Po zavrsetku funkcije u polje a ce biti smesten kolicnik polaznog
```

```
! polinoma i kvadratnog faktora. Stepen polinoma mora biti najmanje 3,
! koeficijent uz najvisi stepen polinoma mora biti jednak 1 i tacnost
! mora biti broj veci od 0. Pretpostavlja se da su sva polja alocirana
! izvan procedure.
subroutine bairstow (n, a, p, q, epsilon, x0, x1)
  integer, intent(in) :: n
  real(kind=8), dimension(0:n), intent(inout) :: a
  real(kind=8), intent(inout)  :: p, q
  real(kind=8), intent(in) :: epsilon
  complex, intent(out) :: x0, x1
  real(kind=8):: b_prev, b_curr, b_next ! Vrednosti koeficijenata
      ! polinoma kolicnika.
  real(kind=8) :: c_prev, c_curr, c_next ! Pomocne vrednosti za
      ! izracunavanje koeficijenata polinoma kolicnika.
  real(kind=8) :: c_prim, D, Dp, Dq ! Pomocne promenljive za
      ! izracunavanje inkremenata koeficijenata kvadratnog faktora.
  real(kind=8) :: delta_p, delta_q ! Inkrementi koeficijenata
      ! kvadratnog faktora.
  complex :: discr ! Diskriminanta kvadratnog faktora.
  integer :: k ! Brojac u petljama

  ! Proverava se da li su argumenti procedure ispravni.
  if (n<3) stop
  if (a(0)/=1) stop
  if (epsilon<=0) stop

  do
     ! Izracunavaju se nove vrednosti koeficijenata polinoma
     ! kolicnika, kao i pomocnih vrednosti za izracunavanje istih.
     b_prev=0; b_curr=1
     c_prev=0; c_curr=1
     do k=1, n-2
        b_next=a(k)-p*b_curr-q*b_prev
        b_prev=b_curr; b_curr=b_next
        c_next=b_curr-p*c_curr-q*c_prev
        c_prev=c_curr; c_curr=c_next
     end do
     do k=n-1, n
        b_next=a(k)-p*b_curr-q*b_prev
        b_prev=b_curr; b_curr=b_next
     end do

     ! Resava se sistem jednacina po inkrementima koeficijenata
     ! kvadratnog faktora.
     c_prim=-(p*c_curr+q*c_prev)
     D=c_curr*c_curr-c_prim*c_prev
     if (D==0) stop
     Dp=b_prev*c_curr-b_curr*c_prev
     Dq=-b_prev*c_prim+b_curr*c_curr
     delta_p=Dp/D
     delta_q=Dq/D

     ! Inkrementiraju se vrednosti koeficijenata kvadratnog faktora.
     p=p+delta_p
     q=q+delta_q

     ! Proverava se da li je dostignuta zeljena tacnost i ako jeste
```

```
      ! prekida se iterativni postupak.
      if (abs(delta_p)<epsilon .and. abs(delta_q)<epsilon) then
         exit
      end if
   end do

   ! Odredjuju se nule kvadratnog faktora.
   discr=sqrt(cmplx(p*p-4*q))
   x0=(-p+discr)/2
   x1=(-p-discr)/2

   ! Izracunava se kolicinik polaznog polinoma i kvadratnog faktora i
   ! smesta u polazni polinom.
   a(1)=a(1)-p
   do k=2, n
      a(k)=a(k)-p*a(k-1)-q*a(k-2)
   end do
  end subroutine bairstow
end module bairstow_module
```

7.4 Gradijentne metode

Iterativne metode za nalaženje tačke minimuma funkcionala $F(\mathbf{x})$, definisane su formulom

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \lambda^{(n)}\,\mathbf{d}^{(n)}, \qquad n = 0, 1, \ldots.$$

Popravka $\lambda^{(n)}$ se računa po formuli

$$\lambda^{(n)} = -\frac{\nabla F(\mathbf{x}^{(n)}) \cdot \mathbf{d}^{(n)}}{(\mathbf{d}^{(n)})^{\top} H(\mathbf{x}^{(n)})\,\mathbf{d}^{(n)}} = -\frac{\sum_{i=1}^{m} \frac{\partial F(\mathbf{x}^{(n)})}{\partial x_i}\, d_i^{(n)}}{\sum_{i=1}^{m}\sum_{j=1}^{m} \frac{\partial^2 F(\mathbf{x}^{(n)})}{\partial x_i \partial x_j}\, d_i^{(n)} d_j^{(n)}}.$$

Vektor pravca popravke $\mathbf{d}^{(n)}$ definiše metodu.

Metoda pokoordinatnog spusta

$$\mathbf{d}^{(n)} = \mathbf{e}_k^{(n)} = (0, \ldots, 1, \ldots, 0)^{\top}, \quad n = 0, 1, \ldots$$

(jedinica je na k-tom mestu).

Metoda najbržeg spusta

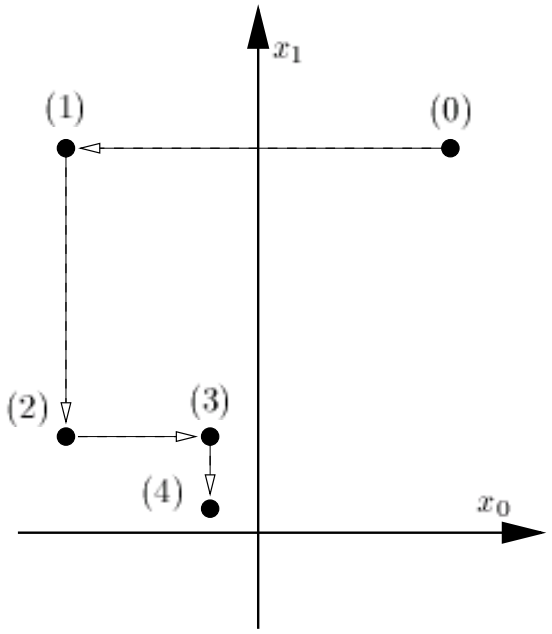
$$\mathbf{d}^{(n)} = -\nabla F(\mathbf{x}^{(n)}) = -\left(\frac{\partial F(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial F(\mathbf{x})}{\partial x_m}\right)^{\top}$$

7.42 *Metodom pokoordinatnog spusta odrediti minimum funkcije*

$$F(x, y) = x^2 + y^2 + xy,$$

polazeći od tačke $x^{(0)} = 1$, $y^{(0)} = 2$.

Rešenje: Izračunavanje je grafički prikazano na slici 7.4.



Slika 7.4: Izračunavanje metodom pokoordinatnog spusta.

Iterativni postupak se sastoji u tome da se naizmenično u svakom koraku funkcija $F(x, y)$ tretira kao funkcija jedne promenljive, tako što se za drugu promenljivu stavi da je jednaka vrednosti u poslednjoj iteraciji. Zatim se računa tačka ekstrema dobijene funkcije.
Prva iteracija:

$$f_0(x) = F(x, y^{(0)}) = x^2 + 2x + 4 \qquad \frac{df_0}{dx} = 2x + 2$$
$$\frac{df_0}{dx} = 0 \quad \text{za} \quad x = -1 \qquad \Rightarrow \qquad x^{(1)} = -1 \quad y^{(1)} = 2$$

Druga iteracija:

$$f_1(y) = F(x^{(1)}, y) = y^2 - y + 1 \qquad \frac{df_1}{dy} = 2y - 1$$
$$\frac{df_1}{dy} = 0 \quad \text{za} \quad y = \frac{1}{2} \qquad \Rightarrow \qquad x^{(2)} = -1 \quad y^{(2)} = \frac{1}{2}$$

Treća iteracija:

$$f_2(x) = F(x, y^{(2)}) = x^2 + \tfrac{1}{2}\,x + \tfrac{1}{4} \qquad \tfrac{df_2}{dx} = 2x + \tfrac{1}{2}$$
$$\tfrac{df_2}{dx} = 0 \quad \text{za} \quad x = -\tfrac{1}{4} \qquad \Rightarrow \qquad x^{(3)} = -\tfrac{1}{4} \quad y^{(3)} = \tfrac{1}{2}$$

Četvrta iteracija:

$$f_3(y) = F(x^{(3)}, y) = y^2 - \tfrac{1}{4}\,y + \tfrac{1}{16} \qquad \tfrac{df_3}{dy} = 2y - \tfrac{1}{4}$$
$$\tfrac{df_3}{dy} = 0 \quad \text{za} \quad y = \tfrac{1}{8} \qquad \Rightarrow \qquad x^{(4)} = -\tfrac{1}{4} \quad y^{(4)} = \tfrac{1}{8}$$
$$\vdots$$

S obzirom da je $x^2 + y^2 \geq 2|xy|$, može se i intuitivno zaključiti da je traženi minimum tačka $(0, 0)$. Proces sporo konvergira ka rešenju.

7.43 *Naći minimum funkcije $F(x, y) = x^2 + y^2 + xy$ metodom najstrmijeg spusta, polazeći od tačke $x^{(0)} = 1$, $y^{(0)} = 2$.*

Rešenje: Gradijent funkcije $F(x, y)$ je vektor

$$\nabla F(x, y) = \begin{pmatrix} 2x + y \\ 2y + x \end{pmatrix} \qquad \Rightarrow \qquad \nabla F(x^{(0)}, y^{(0)}) = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

Aproksimacija rešenja u sledećem koraku je

$$\begin{pmatrix} x^{(1)} \\ y^{(1)} \end{pmatrix} = \begin{pmatrix} x^{(0)} \\ y^{(0)} \end{pmatrix} - \lambda \nabla F(x^{(0)}, y^{(0)}) = \begin{pmatrix} 1 - 4\lambda \\ 2 - 5\lambda \end{pmatrix}$$

Funkcija $F(x^{(1)}, y^{(1)})$ je funkcija promenljive λ,

$$\Phi_1(\lambda) \equiv F(x^{(1)}, y^{(1)}) = (1 - 4\lambda)^2 + (2 - 5\lambda)^2 + (1 - 4\lambda)(2 - 5\lambda).$$

Parametar λ se određuje tako da ova funkcija postiže minimalnu vrednost,

$$\frac{d\Phi_1(\lambda)}{d\lambda} = 0 \quad \text{za} \quad \lambda^{(0)} = 0.3361, \quad \text{tj.} \quad x^{(1)} = -0.3443, \quad y^{(1)} = 0.3197.$$

U drugom koraku je $\nabla F(x^{(1)}, y^{(1)}) = \begin{pmatrix} -0.3689 & 0.2951 \end{pmatrix}^\top$, pa je

$$\begin{pmatrix} x^{(2)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} x^{(1)} \\ y^{(1)} \end{pmatrix} - \lambda \nabla F(x^{(1)}, y^{(1)}) = \begin{pmatrix} -0.3443 + 0.3689\lambda \\ 0.3917 - 0.2951\lambda \end{pmatrix}$$

Funkcija $F(x^{(2)}, y^{(2)})$,

$$\begin{aligned} \Phi_2(\lambda) \equiv F(x^{(2)}, y^{(2)}) =& (-0.3443 + 0.3689\lambda)^2 + (0.3917 - 0.2951\lambda)^2 \\ &+ (-0.3443 + 0.3689\lambda)(0.3917 - 0.2951\lambda), \end{aligned}$$

ima minimalnu vrednost

$$\frac{d\Phi_2(\lambda)}{d\lambda} = 0 \quad \text{za} \quad \lambda^{(1)} = 0.9764, \quad \text{tj.} \quad x^{(2)} = 0.0159 \quad y^{(2)} = 0.0316.$$

Poređenjem ovog i prethodnog zadatka se može zaključiti da metoda najstrmijeg spusta znatno brže konvergira od metode pokoordinatnog spusta.

7.44 *Primeniti metodu najstrmijeg spusta za rešavanje sistema linearnih jednačina* $A\mathbf{x} = \mathbf{b}$*, gde je* A *simetrična* ($A = A^*$) *i pozitivno definisana* ($(A\mathbf{x}, \mathbf{x}) \geq 0$) *matrica.*

Rešenje: Neka je $F(\mathbf{x}) = (A\mathbf{x}, \mathbf{x}) - 2(\mathbf{b}, \mathbf{x})$ i neka je vektor $\mathbf{x}^*$ rešenje polaznog sistema $A\mathbf{x} = \mathbf{b}$. Tada je:

$$\begin{aligned} F(\mathbf{x}) - F(\mathbf{x}^*) &= (A\mathbf{x}, \mathbf{x}) - 2(\mathbf{b}, \mathbf{x}) - (A\mathbf{x}^*, \mathbf{x}^*) + 2(\mathbf{b}, \mathbf{x}^*) = \\ &= (A\mathbf{x}, \mathbf{x}) - 2(A\mathbf{x}^*, \mathbf{x}) - (A\mathbf{x}^*, \mathbf{x}^*) + 2(A\mathbf{x}^*, \mathbf{x}^*) = \\ &= (A\mathbf{x}, \mathbf{x}) - (A\mathbf{x}^*, \mathbf{x}) - (\mathbf{x}^*, A\mathbf{x}) + (A\mathbf{x}^*, \mathbf{x}^*) = \\ &= (A\mathbf{x}, \mathbf{x}) - (A\mathbf{x}^*, \mathbf{x}) - (A\mathbf{x}, \mathbf{x}^*) + (A\mathbf{x}^*, \mathbf{x}^*) = \\ &= (A(\mathbf{x} - \mathbf{x}^*), \mathbf{x}) - (A(\mathbf{x} - \mathbf{x}^*), \mathbf{x}^*) = \\ &= (A(\mathbf{x} - \mathbf{x}^*), \mathbf{x} - \mathbf{x}^*) \geq 0 \end{aligned}$$

Dakle, rešenje datog sistema je tačka minimuma funkcije $F(\mathbf{x}) = (A\mathbf{x}, \mathbf{x}) - 2(\mathbf{b}, \mathbf{x})$, koji se metodom najstrmijeg spusta određuje iterativnim postupkom opisanim sledećim izrazom:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \lambda^{(n)} \nabla F(\mathbf{x}^{(n)}) = \mathbf{x}^{(n)} - 2\lambda^{(n)}(A\mathbf{x}^{(n)} - \mathbf{b}) = \mathbf{x}^{(n)} + \delta^{(n)}(A\mathbf{x}^{(n)} - \mathbf{b})$$

Sada je $F(\mathbf{x}^{(n+1)}) = \Phi(\delta)$ i $\frac{d\mathbf{x}^{(n+1)}}{d\delta} = A\mathbf{x}^{(n)} - \mathbf{b}$, pa je

$$\begin{aligned} \tfrac{d\Phi}{d\delta} &= \tfrac{d}{d\delta}[\big(A\mathbf{x}^{(n+1)}\,,\, \mathbf{x}^{(n+1)}\big) - 2\big(\mathbf{b}, \mathbf{x}^{(n+1)}\big)] = \\ &= \big(A\tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\,,\, \mathbf{x}^{(n+1)}\big) + \big(A\mathbf{x}^{(n+1)}\,,\, \tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\big) - 2\big(\mathbf{b}\,,\, \tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\big) = \\ &= 2\big(A\mathbf{x}^{(n+1)}\,,\, \tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\big) - 2\big(\mathbf{b}\,,\, \tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\big) = \\ &= 2\big(A\mathbf{x}^{(n+1)} - \mathbf{b}\,,\, \tfrac{d\mathbf{x}^{(n+1)}}{d\delta}\big) = 2\big(A\mathbf{x}^{(n+1)} - \mathbf{b}\,,\, A\mathbf{x}^{(n)} - \mathbf{b}\big) = \\ &= 2\big(A(\mathbf{x}^{(n)} + \delta(A\mathbf{x}^{(n)} - \mathbf{b})) - \mathbf{b}\,,\, A\mathbf{x}^{(n)} - \mathbf{b}\big) \end{aligned}$$

Iz uslova da je $\frac{d\Phi}{d\delta} = 0$ dolazi se do rešenja:

$$\delta^{(n)} = \frac{\big(A\mathbf{x}^{(n)} - \mathbf{b}\,,\, A\mathbf{x}^{(n)} - \mathbf{b}\big)}{\big(A(A\mathbf{x}^{(n)} - \mathbf{b})\,,\, A\mathbf{x}^{(n)} - \mathbf{b}\big)}$$

Napomena: Ova metoda je korišćena za iterativno rešavanje sistema linearnih jednačina, §5.4.

MATLAB

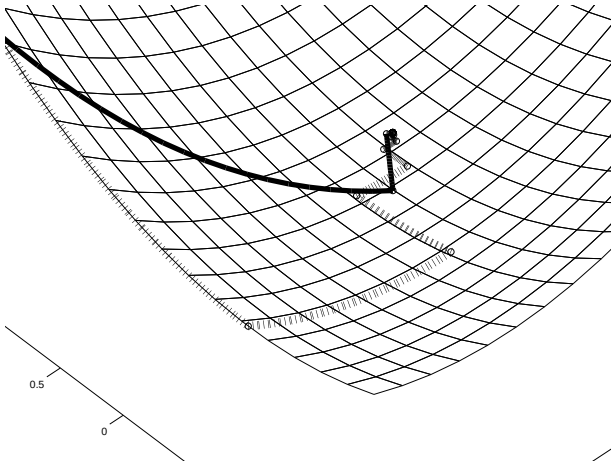
7.45 *Korišćenjem* MATLAB-*a grafički prikazati i uporediti metodu pokoordinatnog i najbržeg spusta na nalaženju minimuma funkcionala* $f(x, y) = x^2+y^2+xy$ *polazeći od tačke* $(x_0, y_0) = (1, 2)$

Rešenje: Za razliku od prikazane implementacije Newton-ove metode, parcijalne izvode funkcije f ćemo unapred izračunati i time ćemo zaobići potrebu za korišćenjem modula za simbolička izračunavanja tokom implementacije spusta. Ove preporuke bi se trebalo pridržavati kad god se želi postići maksimalna brzina rada programa. Modul za simbolička izračunavanja je i dalje moguće interaktivno koristiti u fazi izrade programa čime se značajno olakšava posao i smanjuje mogućnost nastanka greške prilikom diferenciranja složenijih izraza.

```
% Funkcional
f   = inline('x.^2 + y.^2 + x.*y', 'x', 'y');
% Prvi izvodi
fx  = inline('2*x+y', 'x', 'y'); fy  = inline('2*y+x','x', 'y');
% Drugi izvodi
fxx = inline('2', 'x', 'y'); fxy = inline('1', 'x', 'y');
fyx = inline('1', 'x', 'y'); fyy = inline('2', 'x', 'y');

% Pocetna iteracija i tacnost
x0 = 1; y0 = 2; epsilon = 1e-6;

% Iscrtavamo funkcional na [-2, 2]x[-2, 2]
[xg, yg] = meshgrid(linspace(-2, 2, 20), linspace(-2, 2, 20));
zg = feval(f, xg, yg);
hold on; mesh(xg, yg, zg);

% Izracunavamo vrednost funkcionala u pocetnoj tacki
z0 = feval(f, x0, y0);
% Iscrtavamo pocetnu tacku
plot3(x0, y0, z0, 'o');

% Postavljamo vrednost tekuce iteracije na pocetnu
x = x0;  y = y0;

while(1)
   % Pamtimo vrednosti prethodne iteracije
   xp = x; yp = y;

   % Izracunavamo vektor gradijenta u tekucoj tacki
   d = -[feval(fx, x, y) feval(fy, x, y)];

    % Izracunavamo vrednost Hessiana u tekucoj tacki
    H = [feval(fxx, x, y) feval(fxy, x, y);...
         feval(fyx, x, y) feval(fyy, x, y)];

    % Ukoliko tekuca tacka nije stacionarna izracunavamo koeficijenat lambda
    % koji odredjuje "koliko dugo" je potrebno spustati se u pravcu d
    if (~isequal(d,[0 0]))
        lambda = d*d'/(d*H*d');
    else
```

```
        % Ukoliko je tacka stacionarna, nasli smo lokalni minimum
        break;
    end

    % Izracunavamo vrednost nove tacke
    x = x + lambda*d(1);
    y = y + lambda*d(2);

    % Izracunavamo vrednost funkcionala u novoj tacki
    z = feval(f, x, y);

    % Iscrtavamo novu tacku na funkcionalu
    plot3(x, y, z,'o');
    % Krivom po funkcionalu spajamo staru i novu tacku
    xk = linspace(xp, x, 20); yk = linspace(yp, y, 20);
    zk = feval(f, xk, yk);
    plot3(xk, yk, zk, 'k', 'LineWidth', 5);

    % Ispisujemo naslov grafika
    title(['x=' num2str(x) ', y=' num2str(y)]);

    % Ukoliko je postignuta zeljena tacnost prekidamo iterativni postupak
    if (abs(x-xp)<epsilon & abs(y-yp)<epsilon)
        break;
    end
end
```

Primetimo navođenje imena promenljivih iza simboličkih izraza prilikom kreiranja `inline` objekata. Ovo je neophodno, kako bi korišćenjem funkcije `feval` bilo moguće izračunavanje njihovih vrednosti, kao vrednosti funkcija dve promenljive. Implementacija metode pokoordinatnog spusta je veoma slična. Poređenje dato na slici 7.5 ukazuje na mnogo bržu konvergenciju metode najbržeg spusta.



Slika 7.5: Metode pokoordinatnog i najbržeg spusta

Ostale metode

Pronalaženje nule funkcije :	`x = fzero(f, x0)`
Simboličko rešavanje jednačina:	`[x, y] = solve(eq1, ..., eqn)`

Funkcija `fzero` za pronalaženje nula realnih funkcija se može upotrebiti i za rešavanje proizvoljnih nelinearnih jednačina u $\mathbb{R}^1$. Prvi argument funkcije je funkcija f čija se nula traži. Ukoliko je drugi argument dvočlani vektor `[a, b]`, pretpostavlja se da je f različitog znaka u tačkama a i b i pronađena aproksimacija nule se nalazi u intervalu $[a, b]$. Ukoliko je drugi argument funkcije `fzero` skalar x_0, pretpostavlja se da nula funkcije f leži u blizini tačke x_0 i tačka x_0 se uzima za početnu iteraciju iterativnog algoritma.

7.46 *Korišćenjem* MATLAB-*a pronaći pozitivno rešenje jednačine*

$$6\sin(x) = x^3$$

Rešenje: Crtanjem grafika `fplot('6*sin(x)-x^3', [-3 3])`, uočava se da pored trivijalne nule $x = 0$, jednačina ima pozitivno rešenje u blizini tačke $x = 1.8$. Nulu određujemo preko `fzero('6*sin(x)-x^3', 1.8)` i dobijamo $x \approx 1.8010$.

Modul za simbolička izračunavanja (*Symbolic toolbox*) raspolaže funkcijom `solve` koja se koristi za rešavanje nelinearnih jednačina i sistema jednačina. Funkcija pokušava da rešenja sistema pronađe analitički, a u slučaju da analitičko rešavanje ne uspe, rešenja se pronalaze numerički. Npr. moguće je analitički rešiti kvadratne i kubne jednačine u najopštijem obliku

```
>> solve('a*x^2+b*x+c')

ans =

 1/2/a*(-b+(b^2-4*a*c)^(1/2))
 1/2/a*(-b-(b^2-4*a*c)^(1/2))
```

7.47 *Korišćenjem* MATLAB-*a rešiti sistem*

$$\begin{aligned} x &= sin(x+y) \\ y &= cos(x-y) \end{aligned}$$

Rešenje:

```
>> [x, y] = solve('sin(x+y)=x', 'y=cos(x-y)')
x =
    .935082064123103935072592917787 03
y =
    .998020058160098979663061073142 26
```

8

Obične diferencijalne jednačine – Cauchy-jevi problemi

Rešava se Cauchy-jev problem

$$\mathbf{u}'(x) = \mathbf{f}(x, \mathbf{u}(x)), \qquad \mathbf{u}(x_0) = \mathbf{u}_0,$$

gde je

$$\mathbf{u}(x) = \begin{pmatrix} u_1(x) \\ \vdots \\ u_m(x) \end{pmatrix}, \qquad \mathbf{f}(x, \mathbf{u}(x)) = \begin{pmatrix} f_1(x, u_1, \ldots, u_m) \\ \vdots \\ f_m(x, u_1, \ldots, u_m) \end{pmatrix}, \qquad \mathbf{u}_0 = \begin{pmatrix} u_{01} \\ \vdots \\ u_{0m} \end{pmatrix}$$

8.1 Aproksimativne metode

Metoda uzastopnih aproksimacija je određena rekurentnom formulom

$$\mathbf{v}^{(0)}(x) = \mathbf{u}_0,$$
$$\mathbf{v}^{(n+1)}(x) = \mathbf{u}_0 + \int_{x_0}^{x} \mathbf{f}(t, \mathbf{v}^{(n)}(t))\, dt, \quad n = 0, 1, \ldots.$$

Greška se približno ocenjuje veličinom

$$\|\mathbf{u}(x) - \mathbf{v}^{(n)}(x)\| \approx \|\mathbf{v}^{(n)}(x) - \mathbf{v}^{(n-1)}(x)\|.$$

Metodom Taylor-ovog razvoja je aproksimacija rešenja određena parcijalnom sumom Taylor-ovog reda rešenja napisanog u okolini tačke x_0

$$\mathbf{v}^{(n)}(x) = \sum_{j=0}^{n} \frac{\mathbf{u}^{(j)}(x_0)}{j!} (x - x_0)^j$$

Koeficijenti razvoja se računaju uzastopnim diferenciranjem diferencijalne jednačine i računanjem tako dobijenih izvoda rešenja višeg reda u tački x_0. Greška aproksimacije se ocenjuje greškom odsecanja Taylor-ovog razvoja.

Metoda stepenih redova za rešavanje Cauchy-jevog problema definisanog linearnom diferencijalnom jednačinom drugog reda

$$u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \qquad u(0) = u_0, \quad u'(0) = u_0',$$

koristi se onda kada se koeficijenti jednačine mogu razviti u Macloren-ov red

$$p(x) = \sum_{j=0}^{\infty} p_j x^j, \qquad q(x) = \sum_{j=0}^{\infty} q_j x^j, \qquad f(x) = \sum_{j=0}^{\infty} f_j x^j.$$

Rešenje (formalno) se takođe traži u obliku reda

$$u(x) = \sum_{j=0}^{\infty} c_j x^j,$$

pri čemu se rekurentne veze za nalaženje koeficijenata c_k dobijaju zamenom navedenih redova u diferencijalnu jednačinu i izjednačavanjem koeficijenata uz jednake stepene promenljive x,

$$\begin{aligned} c_0 = u_0, \qquad & c_1 = u_0' \\ 2c_2 + p_0 c_1 + q_0 c_0 &= f_0 \\ 6c_3 + 2p_0 c_2 + (p_1 + q_0)c_1 + q_1 c_0 &= f_1 \\ \vdots & \end{aligned}$$

Početne vrednosti su određene početnim uslovima Cauchy-jevog problema. U oblasti u kojoj taj red uniformno konvergira, aproksimacija rešenja je parcijalna suma razvoja rešenja u red

$$v^{(n)}(x) = \sum_{j=0}^{n} c_j x^j,$$

8.1 *Metodom uzastopnih aproksimacija sa tačnošću* $\varepsilon = 5\cdot 10^{-3}$ *na odsečku* $[0, 0.3]$ *odrediti rešenje Cauchy-jevog problema:*

$$\begin{aligned} u_1' &= x + u_1 u_2, \qquad u_1(0) = 1, \\ u_2' &= x^2 - u_1^2, \qquad u_2(0) = \tfrac{1}{2}. \end{aligned}$$

Rešenje: Početna aproksimacija $\mathbf{v}^{(0)}$ biće izabrana tako da su njena vrednost i vrednost njenog prvog izvoda u tački $x = 0$ jednake sa odgovarajućim vrednostima funkcije $\mathbf{u}(\mathbf{x})$ (tangenta na funkciju u tački 0),

$$\begin{aligned} u_1'(0) &= 0 + u_1(0)\, u_2(0) = \frac{1}{2} \\ u_2'(0) &= 0 - u_1^2(0) = -1 \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} v_1^{(0)}(x) &= \frac{1}{2}\, x + 1 \\ v_2^{(0)}(x) &= -x + \frac{1}{2}. \end{aligned}$$

Dalje je, prema iterativnoj formuli,

$$\begin{aligned} v_1^{(1)} &= 1 + \textstyle\int_0^x \big(t + (\frac{1}{2}t + 1)(-t + \frac{1}{2})\big)\, dt &&= -\tfrac{x^3}{6} + \tfrac{x^2}{8} + \tfrac{x}{2} + 1 \\ v_2^{(1)} &= \tfrac{1}{2} + \textstyle\int_0^x \big(t^2 - (\frac{1}{2}t + 1)^2\big)\, dt &&= \tfrac{x^3}{4} - \tfrac{x^2}{2} - x + \tfrac{1}{2} \end{aligned}$$

Već aproksimacija $v_1^{(1)}$ ne zadovoljava traženu tačnost,

$$\max_{[0,0.3]} |v_1^{(1)} - v_2^{(0)}| = \max_{[0,0.3]} \Big| \frac{-x^3}{6} + \frac{x^2}{8} \Big| = 0.012 > \varepsilon.$$

U drugoj iteraciji je

$$\begin{aligned} v_1^{(2)} &= 1 + \textstyle\int_0^x \big(t + (-\frac{t^3}{6} + \frac{t^2}{8} + \frac{t}{2} + 1)(\frac{t^3}{4} - \frac{t^2}{2} - t + \frac{1}{2})\big)\, dt = \\ &= -\tfrac{x^7}{168} + \tfrac{11x^6}{168} + \tfrac{11x^5}{240} - \tfrac{5x^4}{96} - \tfrac{5x^3}{16} + \tfrac{x^2}{8} + \tfrac{x}{2} + 1 \\ v_1^{(2)} &= \tfrac{1}{2} + \textstyle\int_0^x \big(t^2 - (-\frac{t^3}{6} + \frac{t^2}{8} + \frac{t}{2} + 1)^2\big)\, dt = \\ &= -\tfrac{x^7}{252} + \tfrac{x^6}{144} + \tfrac{29x^5}{960} + \tfrac{5x^4}{96} + \tfrac{x^3}{6} - \tfrac{x^2}{2} - x + \tfrac{1}{2} \end{aligned}$$

Tačnost je postignuta, jer je ocena greške ove aproksimacije

$$\max_{[0.3]} \big|v_1^{(2)} - v_1^{(1)}\big| \le 0.00436 < \varepsilon \qquad \max_{[0,0.3]} \big|v_2^{(2)} - v_2^{(1)}\big| \le 0.00225 < \varepsilon.$$

8.2 *Metodom uzastopnih aproksimacija odrediti približno rešenje Cauchyevog problema*

$$u''(x) + 4u'(x) + 4u(x) = 8, \qquad u(0) = 0, \quad u'(0) = 0.$$

Rešenje: Integraljenjem jednačine u intervalu $[0, t]$, uzimajući u obzir date početne uslove, dobijamo jednačinu

$$u'(t) + 4u(t) + 4\int_0^t u(t_1)\, dt_1 = 8t.$$

Ponovnim integraljenjem u intervalu $[0, x]$ dobijamo da je

$$u(x) + 4\int_0^x u(t)\,dt + 4\int_0^x \int_0^t u(t_1)\,dt_1\,dt = 4x^2,$$

i, s obzirom da je

$$\int_0^x \int_0^t u(t_1)\,dt_1\,dt = \int_0^x (x-t)u(t)\,dt,$$

Volterraova jednačina ekvivalentna polaznom problemu je

$$u(x) + 4\int_0^x (1+x-t)u(t)\,dt = 4x^2.$$

Metodom uzastopnih aproksimacija definisanom rekurentnom formulom

$$v_0(x) = 4x^2, \qquad v_{n+1}(x) = 4x^2 - 4\int_0^x (1+x-t)v_n(t)\,dt, \quad n = 0, 1, \ldots,$$

dobijamo aproksimacije

$$\begin{aligned} v_1(x) &= \frac{4}{3}(-x + 3x^2 - 4x^3), \\ v_2(x) &= \frac{4}{45}(45x^2 - 60x^3 + 45x^4 + 24x^5 + 2x^6), \\ &\vdots \end{aligned}$$

8.3 *Odrediti prva četiri člana razvoja u Taylor-ov red rešenja sistema jednačina*

$$\begin{aligned} u_1' &= u_1\cos(x) - u_2\sin(x), \qquad u_1(0) = 1, \\ u_2' &= u_1\sin(x) + u_2\cos(x), \qquad u_2(0) = 0. \end{aligned}$$

Rešenje: Iz datih jednačina je $u_1'(0) = 1$, $u_2'(0) = 0$. Uzastopnim diferenciranjem jednačina sistema dobija se

$$\begin{aligned} u_1''(x) &= (u_1' - u_2)\cos x - (u_1 + u_2')\sin x, \qquad u_1''(0) = 1, \\ u_2''(x) &= (u_1' - u_2)\sin x + (u_1 + u_2')\cos x, \qquad u_2''(0) = 1, \end{aligned}$$

i

$$\begin{aligned} u_1'''(x) &= (u_1'' - 2u_2' - u_1)\cos x - (u_2'' + 2u_1' - u_2)\sin x, \qquad u_1'''(0) = 0, \\ u_2'''(x) &= (u_1'' - 2u_2' - u_1)\sin x + (u_2'' + u_2)\cos x, \qquad u_2'''(0) = 3. \end{aligned}$$

Zamenom dobijenih koeficijenata u Taylor-ov razvoj dobijamo tražene aproksimacije

$$u_1(x) = 1 + x + \tfrac{1}{2}\,x^2, \qquad u_2(x) = \tfrac{1}{2}\,x^2 + \tfrac{1}{2}\,x^3.$$

8.4 *Metodom stepenih redova odrediti rešenje Cauchy-jevog problema:*

$$u''(x) - xu'(x) + u(x) = 1 - \cos x, \qquad u(0) = 0, \quad u'(0) = 1.$$

Rešenje: Razvojem desne strane jednačine u stepeni red

$$1 - \cos x = 1 - \sum_{j=0}^{\infty} (-1)^j \frac{x^{2j}}{(2j)!} = \sum_{j=1}^{\infty} (-1)^{j-1} \frac{x^{2j}}{(2j)!},$$

i uvrštavanjem ovog izraza u polaznu diferencijalnu jednačinu, dobija se:

$$\sum_{j=2}^{\infty} j(j-1)c_j x^{j-2} - x\sum_{j=1}^{\infty} jc_j x^{j-1} + \sum_{j=0}^{\infty} c_j x^j = \sum_{j=1}^{\infty} (-1)^{j-1} \tfrac{x^{2j}}{(2j)!}$$

$$\sum_{j=0}^{\infty} (j+2)(j+1)c_{j+2} x^j - \sum_{j=1}^{\infty} jc_j x^j + \sum_{j=0}^{\infty} c_j x^j = \sum_{j=1}^{\infty} (-1)^{j-1} \tfrac{x^{2j}}{(2j)!}$$

$$2c_2 + c_0 + \sum_{j=1}^{\infty} \big((j+2)(j+1)c_{j+2} - (j-1)c_j\big) x^j = \sum_{j=1}^{\infty} (-1)^{j-1} \tfrac{x^{2j}}{(2j)!}$$

Izjednačavanjem koeficijenata uz odgovarajuće stepene x na levoj i desnoj strani dobija se:

$$2c_2 + c_0 = 0, \quad 6c_3 = 0, \quad 12c_4 - c_2 = \frac{1}{2}, \quad 20c_5 - 2c_3 = 0, \quad 30c_6 - 3c_4 = -\frac{1}{24}, \ldots$$

Kako je na osnovu početnih uslova $c_0 = u(0) = 0$ i $c_1 = u'(0) = 1$, to je dalje $c_2 = 0$, $c_3 = 0$, $c_4 = 1/24$, $c_5 = 0$ i $c_6 = 1/360, \ldots$. Aproksimacija rešenja polinomom šestog stepena je

$$v(x) = x + \frac{1}{24}\, x^4 + \frac{1}{360} x^6$$

8.5 *Metodom stepenih redova naći rešenje Cauchyevog problema*

$$(1 - \alpha x)u'' + \alpha^2 xu' - \alpha^2 u = 0, \qquad u(0) = 1, \quad u'(0) = \alpha,$$

α je data konstanta.

Rešenje: Rešenje tražimo u obliku stepenog reda $\sum_{k=0}^{\infty} c_k x^k$. Uvrstimo ga u jednačinu i posle sređivanja dobijamo

$$(2c_2 - \alpha^2 c_0) + \sum_{k=1}^{\infty} \big((k+2)(k+1)c_{k+2} - \alpha(k+1)k\, c_{k+1} + \alpha^2(k-1)c_k\big) x^k = 0.$$

Jednačina je zadovoljena za svako x ako je

$$2c_2 - \alpha^2 c_0 = 0,$$
$$(k+2)(k+1)c_{k+2} - \alpha(k+1)k\, c_{k+1} + \alpha^2(k-1)c_k = 0, \quad k = 1, 2, \ldots$$

Iz početnih uslova je $c_0 = 1$ i $c_1 = \alpha$, te je $c_k = \alpha^k/k!$, $k = 0, 1, \ldots,$, a tačno rešenje je

$$u(x) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!} x^k = e^{(\alpha x)}.$$

8.6 *Metodom stepenih redova naći opšte rešenje diferencijalne jednačine*

$$xu'' - (x+n)u' + nu = 0,$$

gde je n dati prirodan broj.

Rešenje: Rešenje tražimo u obliku stepenog reda $\sum_{k=0}^{\infty} c_k x^k$. Uvrstimo ga u jednačinu i posle sređivanja dobijamo

$$n(c_0 - c_1) + \sum_{k=1}^{\infty} (k-n)\big((k+1)c_{k+1} - c_k\big)x^k = 0.$$

Jednačina je zadovoljena za svako x ako je

$$c_0 - c_1 = 0, \qquad \text{i} \qquad (k-n)\big((k+1)c_{k+1} - c_k\big) = 0, \quad k = 1, 2, \ldots,$$

tj.

$$c_1 = c_0, \qquad c_{k+1} = \frac{1}{k+1} c_k, \quad k = 1, 2, \ldots \ \text{ i } \ k \neq n, \qquad c_{n+1} \ \text{ proizvoljno.}$$

Kako početni uslov nije zadat, koeficijent c_0 je takođe proizvoljan. Stoga, na osnovu prethodnih veza imamo da je

$$\begin{aligned} &c_0 \ \text{ proizvoljno}, && c_k = \frac{c_0}{k!}, \quad k = 1, \ldots, n, \\ &c_{n+1} \ \text{ proizvoljno}, && c_{n+k} = \frac{c_{n+1}}{(n+2)\cdots(n+k)} = \frac{(n+1)!}{(n+k)!} c_{n+1}, \quad k = 2, \ldots. \end{aligned}$$

pa je opšte rešenje diferencijalne jednačine

$$\begin{aligned} u(x) &= \sum_{k=0}^{n} \frac{c_0}{k!} x^k + \sum_{k=1}^{\infty} \frac{(n+1)!}{(n+k)!} c_{n+1} x^{n+k} \\ &= \big(c_0 - c_{n+1}(n+1)!\big) \sum_{k=0}^{n} \frac{x^k}{k!} + c_{n+1}(n+1)! \sum_{k=0}^{\infty} \frac{x^k}{k!}. \end{aligned}$$

c_0 i c_{n+1} su proizvoljne konstante pa su i $A = c_0 - c_{n+1}(n+1)!$ i $B = c_{n+1}(n+1)!$ proizvoljne konstante. Koristeći uvedene oznake i Macloren-ov red funkcije e^x dobijamo opšte rešenje jednačine

$$u(x) = A \sum_{k=0}^{n} \frac{1}{k!} x^k + B\, e^x.$$

8.2 Metoda Runge–Kutt-a

Metodom tipa Runge–Kutt-a se na osnovu poznate vrednosti rešenja u tački x određuje vrednost rešenja u tački $x+h$, gde je h dati parametar.

Euler-ova metoda (red tačnosti $p=1$)

$$\mathbf{v}(x+h) = \mathbf{u}(x) + h\,\mathbf{f}(x, \mathbf{u}(x))$$

Modifikovana Euler-ova metoda (red tačnosti $p=2$)

$$\mathbf{v}(x+h) = \mathbf{u}(x) + h\,\mathbf{f}(x+\frac{h}{2}, \mathbf{v}^*), \qquad \mathbf{v}^* = \mathbf{u}(x) + \frac{h}{2}\,\mathbf{f}(x, \mathbf{u}(x))$$

ili

$$\mathbf{v}(x+h) = \mathbf{u}(x) + \frac{h}{2}\left(\mathbf{f}(x, \mathbf{u}(x)) + \mathbf{f}(x+h, \mathbf{v}^*)\right), \qquad \mathbf{v}^* = \mathbf{u}(x) + \frac{h}{2}\,\mathbf{f}(x, \mathbf{u}(x))$$

Metoda Runge–Kutt-a (red tačnosti $p=4$)

$$\mathbf{v}(x+h) = \mathbf{u}(x) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Vektori $\mathbf{k}_j = (k_{j1}, \ldots, k_{jm})^\top$ računaju se prema formulama

$$\mathbf{k}_1 = h\mathbf{f}(x, \mathbf{u}(x)), \qquad \mathbf{k}_2 = h\mathbf{f}(x+h/2, \mathbf{u}(x) + \mathbf{k}_1/2),$$
$$\mathbf{k}_3 = h\mathbf{f}(x+h/2, \mathbf{u}(x) + \mathbf{k}_2/2), \qquad \mathbf{k}_4 = h\mathbf{f}(x+h, \mathbf{u}(x) + \mathbf{k}_3)$$

Greška se može oceniti Runge-ovim kriterijumom

$$\|\mathbf{u}(x+2h) - \mathbf{v}^{(1)}\| \approx \frac{1}{2^p - 1}\,\|\mathbf{v}^{(1)} - \mathbf{v}^{(2)}\|,$$

gde je $\mathbf{v}^{(1)}$ približno rešenje u tački $x+2h$ određeno sa korakom h, $\mathbf{v}^{(2)}$ približno rešenje u istoj tački određeno sa korakom $2h$, a p je red tačnosti metode. Vrednost rešenja popravljena na osnovu procenjene greške je

$$\tilde{\mathbf{v}}(x+h) = \mathbf{v}^{(1)} + \frac{\mathbf{v}^{(1)} - \mathbf{v}^{(2)}}{2^p - 1}$$

8.7 *a) Primenom teoreme o nepokretnoj tački na odgovarajuću integralnu jednačinu dokazati egzistenciju jedinstvenog rešenja Cauchy-jevog problema*

$$u' = \cos(x^2 u + 3x) + 2, \qquad u(0) = 1.$$

na intervalu $[0, L]$, *gde je* $0 < L < 1$.

b) Naći Metodom Runge–Kutt-a sa tačnošću 10^{-4} *rešenje* $u(0.4)$ *zadatka definisanog u tački a).*

Rešenje: a) Integraljenjem date jednačine u granicama od 0 do x, dobijamo integralnu jednačinu

$$u(x) = 1 + \int_0^x \big(\cos(t^2u + 3t) + 2\big)\, dt, \qquad 0 < x < L < 1.$$

Ako označimo integralni operator sa

$$\mathcal{T}(u) = 1 + \int_0^x \big(\cos(t^2u + 3t) + 2\big)\, dt,$$

operatorski oblik integralne jednačine je

$$u = \mathcal{T}(u).$$

Dakle, rešenje datog Cauchy-jevog problema je nepokretna tačka operatora $\mathcal{T}$; ona postoji i jedinstvena je ako je $\mathcal{T}$ operator kontrakcije. Kako je

$$\begin{aligned}\mathcal{T}(u) - \mathcal{T}(w) &= \int_0^x \big(\cos(t^2u + 3t) - \cos(t^2w + 3t)\big)\, dt \\ &= 2\int_0^x \sin\frac{(u+w)t^2 + 6t}{2} \sin\frac{(u-w)t^2}{2}\, dt\end{aligned}$$

to se, majoriranjem prvog činioca podintegralne funkcije sa 1 a drugog sa njegovim argumentom, dobija ocena

$$|\mathcal{T}(u) - \mathcal{T}(w)| \le 2\int_0^x \frac{t^2}{2}|u - w|\, dt$$

iz koje sledi

$$\max_{x\in(0,L)} |\mathcal{T}(u) - \mathcal{T}(w)| \le \frac{L^3}{3} \max_{x\in(0,L)} |u - w|,$$

tj., s obzirom da je $0 < L < 1$,

$$\|\mathcal{T}(u) - \mathcal{T}(w)\| \le \frac{1}{3}\|u - w\|,$$

čime je dokazano da je $\mathcal{T}$ kontrakcija.
b) Metodom Runge–Kutt-a dobija se $v^{(1)}(0.4) = 2.07838$ sa korakom $h = 0.2$ i $v^{(2)}(0.4) = 2.07897$ sa korakom $h = 0.4$. Kako je $|v^{(1)} - v^{(2)}|/15 = 4 \cdot 10^{-5}$, prema Runge-ovom kriterijumu je postignuta tačnost, i rešenje je sa traženom tačnošću $u(0.4) = 2.0784$.

8.8 *Neka je* $u(x)$ *kriva koja prolazi kroz tačku* $(0, 1)$ *i ima osobinu da je koeficijent nagiba tangente u svakoj njenoj tački jednak dužini radius vektora te tačke. Naći* $u(0.1)$ *i* $u(0.2)$ *sa tačnošću* 10^{-4}.

Rešenje: Koeficijent nagiba tangente u tački x je $u'(x)$, a dužina radius vektora tačke (x, u) je $\sqrt{x^2+u^2}$. Stoga je funkcija $u(x)$ rešenje Cauchy-jevog problema

$$u'(x) = \sqrt{x^2+u^2}, \qquad u(0) = 1.$$

Traženo rešenje je određeno metodom Runge–Kutt-a sa korakom $h = 0.1$,

$$u(0.1) \approx v(0.1) = 1.1053, \qquad u(0.2) \approx v(0.2) = 1.2226.$$

Tačnost je zadovoljena prema Runge-ovom kriterijumu, jer je sa korakom $h = 0.2$ takođe dobijena vrednost $u(0.2) = 1.2226$.

8.9 *Neka je $u(x)$ rešenje diferencijalne jednačine koje zadovoljava zadate uslove*

$$u'(x) = e^{(x+u)} - 1, \qquad u(x_0) = 0.1, \quad u'(x_0) = 0.$$

Sa tačnošću 10^{-4} naći $u(0)$ i $u(0.1)$.

Rešenje: Zamenom u jednačini datih vrednosti u tački x_0 za u i u', dobijamo vrednost $x_0 = -0.1$. Dalje se metodom Runge–Kutt-a sa korakom $h = 0.1$ dobija $v^{(1)}(0) = 0.10536$ i $v^{(1)}(0.1) = 0.12314$. Istom metodom sa korakom $h = 0.2$ dobija se da je $v^{(2)}(0.1) = 0.12314$. Kako je $|v^{(1)}(0.1) - v^{(2)}(0.1)|/15 < 10^{-4}$, prema Runge-ovom kriterijumu tačnost je postignuta, pa je na četiri sigurne cifre

$$u(0) = 0.1054, \qquad u(0.1) = 0.1231.$$

8.10 *Data je diferencijalna jednačina*

$$u' = (e^u - x - 1)\sqrt{x^2+2} + \frac{1}{x+1}.$$

Neka je $x_0 > 0$ tačka u kojoj je $u(x_0) = 0.530628$ i $u'(x_0) = 0.588235$. Odrediti $u(0.5)$ sa tačnošću 10^{-4}.

Rešenje: Zadatak se rešava kao prethodni, sa tom razlikom što se tačka x_0 dobija kao rešenje nelinearne jednačine

$$(e^{0.530628} - x_0 - 1)\sqrt{x_0^2+2} + \frac{1}{x_0+1} = 0.588235.$$

Newtonovom metodom se određuje da je sa traženom tačnošću $x_0 = 0.700000$. Metodom Runge–Kutt-a dobija se da je u tački $x = 0.5$ približno rešenje $v^{(1)} = 0.405464$ izračunato sa korakom $h = -0.1$ i $v^{(2)} = 0.405451$ izračunato sa korakom $h = -0.2$. Prema Runge-ovom kriterijumu tačnost je postignuta, pa je rešenje sa traženom tačnošću $u(0.5) = 0.4055$.

8.11 *Dat je Cauchy-jev problem*

$$u' - u\int_0^x \frac{1}{2}\, e^{t^2}\, dt - 2x = 0, \qquad u(0) = 1.2345.$$

Izračunati $u(0.4)$ *sa tačnošću* 10^{-4}.

Rešenje: Jednačina se može zapisati u obliku

$$u' = u\int_0^x \frac{1}{2}\, e^{t^2}\, dt + 2x = f(x,u)$$

Ako metodom Runge–Kutt-a sa korakom $h = 0.2$ računamo približno rešenje jednačine, potrebne su nam vrednosti funkcije $f(x,u)$ u tačkama $x = 0.1,\ 0.2,\ 0.3,\ 0.4$ izračunate sa tačnošću 10^{-4}. Da bi integral u izrazu za funkciju $f(x,u)$ računali Simpsonovom kvadraturnom formulom sa traženom tačnošću, korak k za izračunavanje integrala treba da bude takav da je greška $(b-a)k^4M_4/180 \le 10^{-4}$, gde je $a = 0$, $b \le 0.4$, a $M_4 = \max_{[0,0.4]} |(e^{x^2}/2)^{(4)}| = 11.8$. Stoga je $k \le 0.2$ i

x	0	0.1	0.2	0.3	0.4
$\int_0^x e^{t^2}/2\, dt$	0	0.05017	0.10135	0.15463	0.21123 .

Sada, računajući metodom Runge–Kutt-a, dobijamo da je $v^{(1)}(0.4) = 1.44967$ za $h = 0.2$ i $v^{(2)}(0.4) = 1.44969$ za $h = 0.4$. Kako je petnaestina apsolutne vrednosti razlike ova dva rešenja jednaka 10^{-6}, prema Runge-ovom kriterijumu tačnost je postignuta. Sa traženom tačnošću je

$$u(0.4) = 1.4497.$$

8.12 *Izračunati vrednosti funkcija* $u_1(x)$, $u_2(x)$ *i* $u_3(x)$ *na odsečku* $[0, 0.3]$ *sa korakom* $h = 0.1$ *ako su ove funkcije rešenja Cauchy-jevog problema*

$$\begin{aligned} u_1' &= -2u_1 + 5u_3, && u_1(0) = 2,\\ u_2' &= (\sin(x) - 1)u_1 - u_2 + 3u_3, && u_2(0) = 1,\\ u_3' &= -u_1 + 2u_3, && u_3(0) = 1. \end{aligned}$$

Rešenje: Izračunavanje za tačku $x = 0.1$ se može predstaviti sledećom tabelom

$\mathbf{u}_0$	$\mathbf{k}_1$	$\mathbf{k}_2$	$\mathbf{k}_3$	$\mathbf{k}_4$	$\mathbf{u}(0.1)$
2	0.1000	0.0900	0.0897	0.0795	2.0898
1	0.0000	0.0052	0.0047	0.0099	1.0050
1	0.0000	−0.0050	−0.0050	−0.0100	0.9950

Popravke su zanemarljive u odnosu na tačnost sa kojom se računa. Koristeći ovako dobijene vrednosti funkcija za $x = 0.1$ kao početni uslov, na isti način se se dobija

$$u_1(0.2) = 2.1588, \qquad u_2(0.2) = 1.0195, \qquad u_3(0.2) = 0.9801.$$

Konačno, polazeći od ovih vrednosti dobija se da je

$$u_1(0.3) = 2.2062, \qquad u_2(0.2) = 1.0427, \qquad u_3(0.2) = 0.9553.$$

8.13 *Sa tačnošću* 10^{-4} *naći u tački* $x = 1.2$ *rešenje Cauchy-jevog problema*

$$u''(x) = u'(x) + e^{x^2}, \qquad u(1) = 1.23425, \quad u'(1) = 0.78502.$$

Rešenje: Smenom $u_1 = u$, $u_2 = u'$, zadatak se svodi na Cauchy-jev problem

$$\begin{aligned} u_1'(x) &= u_2(x), & u_1(1) &= 1.23425 \\ u_2'(x) &= u_2(x) + e^{x^2}, & u_2(1) &= 0.78502. \end{aligned}$$

Rešavan je metodom Runge–Kutt-a sa koracima $h = 0.1$ i $h = 0.2$

x	1	1.1	1.2	korak
$v_1^{(1)}$	1.23425	1.33187	1.47516	$h = 0.1$
$v_2^{(1)}$	0.78502	1.18462	1.70433	
$v_1^{(2)}$	1.23425		1.47512	$h = 0.2$
$v_2^{(2)}$	0.78502		1.70432	

gde su sa v_1 označene približne vrednosti funkcije u_1 a sa v_2 približne vrednosti funkcije u_2. Kako je $|v_1^{(1)}(1.2) - v_1^{(2)}(1.2)|/15 = 3 \cdot 10^{-6}$ i $|v_2^{(1)}(1.2) - v_2^{(2)}(1.2)|/15 = 7 \cdot 10^{-7}$, prema Runge-ovom kriterijumu je sa traženom tačnošću

$$u(1.2) = v_1^{(1)}(1.2) = 1.4752.$$

FORTRAN Implementacija metode Runge–Kutt-a sastoji se od dve procedure. Procedura `runge_kutta_step()` računa jednu aproksimaciju rešenja Cauchy-jevog problema metodom Runge–Kutt-a. Ova procedura se potom poziva iz glavne procedure `runge_kutta()` da bi se izračunalo rešenje u traženoj tački sa koracima h odn. $\frac{h}{2}$. Na osnovu ove dve vrednosti izračunava se korekcija i na taj način se dolazi do konačne vrednosti rešenja. Treba uočiti kako se u proceduri `runge_kutta_step()` intenzivno koriste operacije nad vektorima. Kompletan kod koji implementira metodu ima sledeći oblik:

```
module runge_kutta_module
  use matheval_module
  implicit none
  public runge_kutta

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string
```

```
contains
  ! Funkcija runge_kutta() odredjuje resenje Cauchy-jevog problema u
  ! tacki na zadatom rastojanju od tacke u kojoj je resenje problema
  ! poznato. Argumenti funkcije su:
  !   n - dimenzija sistema
  !   variables - string sa imenom nezavisne promenljive i imenima
  !               funkcija cija se vrednost izracunava
  !   f - vektor sa desnim stranama sistema
  !   x0 - tacka u kojoj je resenje problema dato
  !   h - rastojanje tacke u kojoj se trazi resenje sistema od tacke x0
  !   u0 - vektor sa vrednostima funkcija u datoj tacki
  !   epsilon - zeljena tacnost
  !   v - vektor u koji ce biti smestene trazene vrednosti funkcija
  ! Pretpostavlja se da su sva polja alocirana izvan funkcije. Dimenzija
  ! sistema i tacnost moraju biti veci od 0, dok velicina h mora biti
  ! razlicita od 0. Rezultat funkcije je indikator da li je postignuta
  ! zeljena tacnost.
  logical function runge_kutta (n, variables, f, x0, h, u0, epsilon, v)
    integer, intent(in) :: n
    type(string), intent(in) :: variables
    type(string), dimension(0:n-1), intent(in) :: f
    real(kind=8), intent(in) :: x0
    real(kind=8), intent(in) :: h
    real(kind=8), dimension(0:n-1), intent(in) :: u0
    real(kind=8), intent(in) :: epsilon
    real(kind=8), dimension(0:n-1), intent(out) :: v
    integer(kind=8), dimension(0:n-1) :: evaluator ! Evaluatori za desne
        ! strane jednacina sistema.
    real(kind=8), dimension(0:n-1) :: vh ! Vrednosti funkcija u tacki
        ! x0+h izracunate sa korakom h.
    real(kind=8), dimension(0:n-1) :: vhh ! Vrednosti funkcija u tacki
        ! x0+h/2 izracunate sa korakom h/2.
    real(kind=8) :: correction ! Popravka rezultata.
    integer :: k ! Brojac u petljama.

    ! Proverava se validnost ulaznih podataka. Pritom se kreiraju
    ! evaluatori za desne strane jednacina sistema.
    if (n<=0) stop
    do k=0, n-1
       evaluator(k)=evaluator_create(f(k)%chars)
       if (evaluator(k)==0) stop
    end do
    if (h==0) stop
    if (epsilon<=0) stop

    ! Izracunavaju se vrednosti funkcija u tacki x0+h sa korakom h.
    call runge_kutta_step (n, variables, evaluator, x0, h, u0, vh)

    ! Izracunavaju se vrednosti funkcija u tackama x0+h/2 i x0+h sa
    ! korakom h/2.
    call runge_kutta_step (n, variables, evaluator, x0, h/2, u0, vhh)
    call runge_kutta_step (n, variables, evaluator, x0+h/2, h/2, vhh, v)

    ! Unistavaju se korisceni evaluatori.
    do k=0, n-1
       call evaluator_destroy(evaluator(k))
    end do
```

```
  ! Odredjuju se Runge-ova ocene greske i vrsi se korekcija
  ! izracunatih vrednosti. Istovremeno se porede ocjene greske sa
  ! zeljenom tacnoscu.
  runge_kutta=.true.
  do k=0, n-1
     correction=(v(k)-vh(k))/15
     if (abs(correction)>epsilon) then
        runge_kutta=.false.
     end if
     v(k)=v(k)+correction
  end do
end function runge_kutta

! Procedura runge_kutta_step() izracunava aproksimaciju resenja
! Cauchy-jevog problema u zadatoj tacki na osnovu zadatih pocetnih
! uslova. Argumenti funkcije su:
!   n - dimenzija sistema
!   variables - string sa imenom nezavisne promenljive i imenima
!               funkcija cija se vrednost u datoj tacki izracunava
!   evaluator - polje evaluatora za izracunavanje funkcija na desnoj
!               strani sistema
!   x0 - tacka u kojoj je resenje problema dato
!   h - rastojanje od tacke x0 tacke u kojoj se trazi resenje sistema
!   u0 - vektor sa pocetnim uslovima
!   v - vektor u koji ce biti smestene trazene vrednosti funkcija
subroutine runge_kutta_step (n, variables, evaluator, x0, h, u0, v)
  integer, intent(in) :: n
  type(string), intent(in) :: variables
  integer(kind=8), dimension(0:n-1), intent(in) :: evaluator
  real(kind=8), intent(in) :: x0
  real(kind=8), intent(in) :: h
  real(kind=8), dimension(0:n-1), intent(in) :: u0
  real(kind=8), dimension(0:n-1), intent(out) :: v
  real(kind=8), dimension(0:n) :: values ! Vektor sa vrednostima
      ! nezavisne promenljive i pocetnim uslovima.
  real(kind=8), dimension(0:n-1) :: k0, k1, k2, k3 ! Pomocni vektori
      ! za nalazenje aproksimacije.
  integer :: k ! Brojac u petljama

  ! Izracunavaju se vrednosti pomocnog vektora k0.
  values=(/x0,u0/)
  k0=(/(h*evaluator_evaluate(evaluator(k),n+1,variables%chars,values), k=0, n-1)/)

  ! Izracunavaju se vrednosti pomocnog vektora k1.
  values=(/x0+h/2,(u0(k)+k0(k)/2, k=0, n-1)/)
  k1=(/(h*evaluator_evaluate(evaluator(k),n+1,variables%chars,values), k=0, n-1)/)

  ! Izracunavaju se vrednosti pomocnog vektora k2.
  values=(/x0+h/2,(u0(k)+k1(k)/2, k=0, n-1)/)
  k2=(/(h*evaluator_evaluate(evaluator(k),n+1,variables%chars,values), k=0, n-1)/)

  ! Izracunavaju se vrednosti pomocnog vektora k3.
  values=(/x0+h,(u0(k)+k2(k), k=0, n-1)/)
  k3=(/(h*evaluator_evaluate(evaluator(k),n+1,variables%chars,values), k=0, n-1)/)

  ! Izracunava se trazena aproksimacija resenja.
```

```
    v=(/(u0(k)+(k0(k)+2*k1(k)+2*k2(k)+k3(k))/6, k=0, n-1)/)
  end subroutine runge_kutta_step
end module runge_kutta_module
```

8.14 *a) Pokazati da se greška $\epsilon_j = u(x_j) - v_j$ približne vrednosti rešenja Cauchy-jevog problema $u' = f(x,u), \quad u(x_0) = u_0$ izračunatog formulom*

$$(*) \qquad v_{j+1} = v_j + h(\alpha_0 v'_{j+1} + \alpha_1 v'_j), \qquad \alpha_0, \alpha_1 \geq 0, \quad \alpha_0 + \alpha_1 = 1,$$

($v'_j \approx u'(x_j)$), može oceniti, za $Kh\alpha_0 < 1$, izrazom

$$|\epsilon_j| \leq \frac{E}{Kh}\left(\left(\frac{1+Kh\alpha_1}{1-Kh\alpha_0}\right)^j - 1\right),$$

gde je $|\partial f/\partial u| \leq K$ i E je najveća greška formule $(*)$ učinjena na jednom koraku. b) Oceniti veličinu E i napisati ocenu $|\epsilon_j|$ ako je $\alpha_0 = \alpha_1 = 0.5$.

Rešenje: (a) Formula $(*)$, koristeći oznake iz Cauchy-jevog problema, može da se napiše u obliku

$$(1) \qquad v_{j+1} = v_j + h\big(\alpha_0 f(x_{j+1}, v_{j+1}) + \alpha_1 f(x_j, v_j)\big).$$

Ako sa $u(x_j)$ označimo tačnu vrednost rešenja Cauchy-jevog problema u tački x_j, imamo da je

$$(2) \qquad u(x_{j+1}) = u(x_j) + h\big(\alpha_0 f(x_{j+1}, u(x_{j+1})) + \alpha_1 f(x_j, u(x_j))\big) + E_j,$$

gde je E_j greška formule $(*)$ na jednom koraku. Oduzimajući form. (1) od form. (2) dobijamo da je

$$\epsilon_{j+1} = \epsilon_j + h\Big(\alpha_0\big(f(x_{j+1}, u(x_{j+1})) - f(x_{j+1}, v_{j+1})\big) + \alpha_1\big(f(x_j, u(x_j)) - f(x_j, v_j)\big)\Big) + E_j,$$

što, koristeći teoremu o srednjoj vrednosti, postaje

$$\epsilon_{j+1} = \epsilon_j + h\left(\alpha_0\epsilon_{j+1}\frac{\partial f}{\partial u}(x_{j+1}, \eta_{j+1}) + \alpha_1\epsilon_j\frac{\partial f}{\partial u}(x_j, \eta_j)\right) + E_j$$

tj.

$$\big(1 - \alpha_0 h\frac{\partial f}{\partial u}(x_{j+1}, \eta_{j+1})\big)\epsilon_{j+1} = \big(1 + \alpha_1 h\frac{\partial f}{\partial u}(x_j, \eta_j)\big)\epsilon_j + E_j.$$

Uzimajući u obzir da je $\left|\frac{\partial f}{\partial u}(x,u)\right| \leq K$ and $\max_j E_j = E$ imamo da je za svako j

$$(1 - h\alpha_0 K)|\epsilon_{j+1}| \le (1 + h\alpha_1 K)|\epsilon_j| + E,$$

tj. za $1 - h\alpha_0 K > 0$

$$|\epsilon_{j+1}| \le \frac{1}{1 - h\alpha_0 K}\big((1 + h\alpha_1 K)|\epsilon_j| + E\big).$$

Radi kraćeg zapisa u poslednjoj oceni uvedimo oznake $a = 1 - h\alpha_0 K$ i $b = 1 + h\alpha_1 K$, i primenimo je rekurentno. Tako dobijamo da je

$$\begin{aligned} |\epsilon_{j+1}| &\le \frac{b}{a}|\epsilon_j| + \frac{E}{a} \le \cdots \\ &\le \frac{b^j}{a^j}|\epsilon_1| + \frac{1}{a}\big(1 + \frac{b}{a} + \cdots + \frac{b^{j-1}}{a^{j-1}}\big)E = \frac{b^j}{a^j}|\epsilon_1| + \frac{1}{a}\frac{1 - (b/a)^j}{1 - b/a}E. \end{aligned} \tag{3}$$

Kako je

$$\epsilon_1 = u(x_1) - v^{(1)} = h\alpha_0\big(f(x_1, u(x_1)) - f(x_1, v^{(1)})\big) + E_0 = h\alpha_0\epsilon_1\frac{\partial f}{\partial u}(x_1, \eta_1) + E_0,$$

to je

$$(1 - h\alpha_0 K)|\epsilon_1| \le E, \qquad \text{tj.} \qquad |\epsilon_1| \le \frac{E}{1 - h\alpha_0 K} = \frac{E}{a}.$$

Zamenom poslednje procene u (3) dobijamo da je

$$|\epsilon_{j+1}| \le \frac{b^j}{a^j}\,\frac{E}{a} + \frac{1}{a - b}\big(1 - \frac{b^j}{a^j}\big)E = \frac{E}{b - a}\big(\frac{b^{j+1}}{a^{j+1}} - 1\big),$$

što nam, s obzirom na uvedenu smenu, daje tvrđenje zadatka.
b) Za $\alpha_0 = \alpha_1 = 0.5$ formula $(*)$ postaje

$$v_{j+1} = v_j + \frac{h}{2}(v'_{j+1} + v'_j)$$

i dobija se primenom trapezne kvadraturne formule na izračunavanje integrala

$$u(x_{j+1}) - u(x_j) = \int_{x_j}^{x_{j+1}} u'(x)\,dx = \frac{h}{2}(u'(x_{j+1}) + u'(x_j)) + E, \qquad E \le \frac{h^3}{12}M_3,$$

gde je $M_3 = \max|u'''|$. Stoga je na osnovu tvrđenja dokazanog pod (a)

$$|\epsilon_j| \le \frac{h^2}{12}\,\frac{M_3}{K}\left(\big(\frac{2 + hk}{2 - hk}\big)^j - 1\right).$$

MATLAB

Cauchy-jevi problemi: `ode45(@sistem, interval, initcond)`
Sistem j-na prvog reda se opisuje posebnom funkcijom:

```
function du = system(t, u)
 du = [f1(t,u);...;fn(t,u)];
```

Za rešavanje Cauchy-jevih problema, MATLAB poseduje nekoliko ugrađenih funkcija. Izdvojimo funkciju `ode45`. Sistem jednačina se opisuje posebnom funkcijom koja se snima u m-datoteku i zatim prosleđuje funkciji `ode45`, zajedno sa intervalom na kome se traži rešenje i početnim uslovima sistema. Npr. sistem

$$\begin{array}{rclrcl} u_1' &=& u_2u_3, & u_1(0) &=& 0, \\ u_2' &=& -u_1u_3, & u_2(0) &=& 1, \\ u_3' &=& -0.51u_1u_2, & u_3(0) &=& 1 \end{array}$$

se na intervalu $[0, 12]$ rešava kreiranjem funkcije

```
function du = sistem(t, u)
du = [u(2) * u(3); ...
     -u(1) * u(3); ...
     -0.51 * u(1) * u(2)];
```

Funkcijom se kodira sistem $\mathbf{u}' = f(t, \mathbf{u})$. Prvi argument funkcije je skalar t, dok je drugi argument vektor kolona $\mathbf{u}$, a povratna vrednost predstavlja vektor kolonu $\mathbf{u}'$. Sistem se rešava pozivom funkcije

```
[t,u] = ode45(@sistem, [0 12], [0; 1; 1]);
```

pri čemu vrsta `[0, 12]` predstavlja interval na kome se rešenje traži dok kolona `[0; 1; 1]` predstavlja početne uslove Cauchy-jevog problema. Funkcija vraća vektor `t` koji predstavlja mrežu na kojoj je rešenje izračunato kao i matricu `u`, čija svaka pojedinačna kolona sadrži vrednosti jedne od nepoznatih funkcija u tačkama mreže. Dodatni opcioni parametar `options` omogućuje zadavanje dodatnih opcija za rešavanje jednačina. Više o ovome pogledati u dokumentaciji MATLAB-a.

8.15 *Korišćenjem* MATLAB-*a rešiti jednačinu*

$$u'' + xu = u', \qquad u(0) = 0, \quad u'(0) = 3$$

Rešenje: Prvi korak je transformisanje jednačine u sistem jednačina prvog reda uvođenjem smene $u_2 = u_1'$, pri čemu je $u_1 \equiv u$:

$$\begin{array}{rclrcl} u_1' &=& u_2, & u_1(0) &=& 0 \\ u_2' &=& -xu_1 + u_2, & u_2(0) &=& 3 \end{array}$$

Kreiramo m-datoteku `sistem.m` sledećeg sadržaja:

```
function up = sistem(x, u)
    up = [u(2); ...
          -x*u(1)+u(2);];
```

Zatim sistem rešavamo putem

```
[t, u] = ode45(@sistem, [0 5], [0 3])
```

i iscrtavamo rešenje polazne jednačine

```
plot(t, u(:, 1))
```

8.16 *Korišćenjem* MATLAB-*a simulirati ponašanje loptice koja odskače odbijajući se od tvrde podloge. Na lopticu koja je izbačena iz tačke* (x_0, y_0) *brzinom* $\vec{v} = (v_{x0}, v_{y0})$ *u svakom trenutku deluju dve sile.*

i) Sila otpora vazduha koja ima suprotan smer od vektora brzine loptice i intenzitet proporcionalan kvadratu intenziteta vektora brzine tj. $F_o = cv^2$.

ii) Gravitaciona sila koja je uvek usmerena u pravcu podloge i intenzitet joj je $F_g = mg$

Rešenje: Vektor brzine se može razložiti na svoju x i y komponentu,

$$\vec{v} = (v_x, v_y) = (|v|\cos(\phi),\ |v|\sin(\phi)), \quad \phi = \arctan\frac{v_y}{v_x}$$

Tada je

$$\begin{aligned} \vec{F_o} &= (-cv^2\cos(\phi),\ -cv^2\sin(\phi)), \quad v^2 = v_x^2 + v_y^2 \\ \vec{F_g} &= (0,\ -mg) \end{aligned}$$

Na osnovu drugog Newtonovog zakona je $\Sigma\vec{F} = m\vec{a}$, odnosno

$$\vec{F_o} + \vec{F_g} = m\vec{v}\,'$$

Odavde je

$$\begin{aligned} v_x' &= -\frac{cv^2}{m}\cos(\phi) \\ v_y' &= -\frac{cv^2}{m}\sin(\phi) - g \end{aligned}$$

Pošto nas zanima položaj loptice u svakom trenutku, posmatrajmo još jednačine

$$\begin{aligned} x' &= v_x \\ y' &= v_y \end{aligned}$$

Kako bi kretanje loptice u potpunosti bilo opisano, potrebno je još opisati odbijanje loptice u trenutku udara o podlogu. „Trik" koji je ovde moguće iskoristiti je dopuštanje da vrednost koordinate y bude i negativna, a da se u trenutku prolaska prave $y = 0$ promeni znak gravitacione konstante umesto menjanja znaka v_y komponente vektora v.

Dobijeni sistem od četiri diferencijalne jednačine sa četiri nepoznate funkcije rešavamo korišćenjem funkcije `ode45`. Prilikom poziva funkcije `ode45`, prosleđujemo

i dodatne parametre `c`, `g` i `m` koji određuju gravitacionu konstantu, gravitaciono ubrzanje i masu loptice. Funkcija `ode45`, zatim ove parametre prosleđuje funkciji `bouncing_ball` pri svakom njenom pozivu.

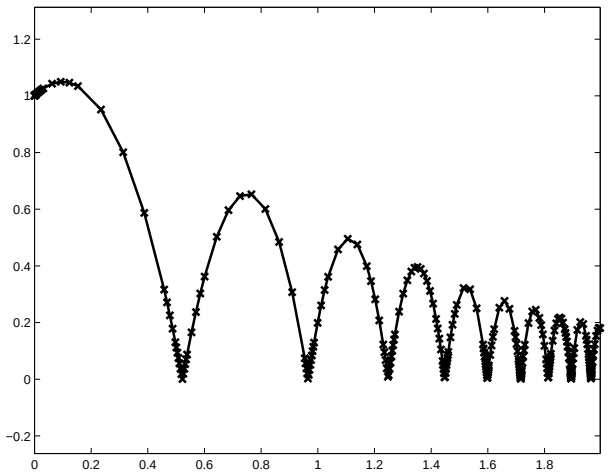
Datoteka `bouncing_ball.m`:

```
function Xp = bouncing_ball(t, X, c, m, g)
x = X(1); y = X(2); vx = X(3); vy = X(4);

phi = atan2(vy, vx);
v = vx^2 + vy^2;

Xp = [ vx; ...
       vy; ...
      -c*v/m * cos(phi); ...
      -c*v/m * sin(phi) - sign(y)*g; ];
```

Simulaciju vršimo preko

```
c = 5; m = 20; g = 9.81;
X0 = [0; 1; 1; 1];

[t, X] = ode45(@bouncing_ball, [0, 5], X0, [], c, m, g);
plot(X(:,1), abs(X(:, 2)), '-x'); axis equal
```

Rezultat je prikazan na slici 8.1.



Slika 8.1: Simulacija loptice koja odskače

8.17 *Koristeći* MATLAB *prikazati putanju Zemlje koje se kreće u gravitacionom polju Sunca. Masa Sunca je* $M \approx 2 \cdot 10^{30}$ *kg, udaljenost Zemlje od Sunca u trenutku kada mu je najbliža je* $d_0 \approx 147 \cdot 10^6$ *km, i u tom trenutku brzina joj je* $v_0 \approx 30.4$ $\frac{km}{s}$.

Rešenje: Postavimo koordinatni sistem tako da mu je Sunce u centru, dok se Zemlja nalazi u tački $(x,\ y)$. U trenutku kada je zemlja najbliža suncu ona se nalazi u tački $(d_0,\ 0)$ i vektor brzine joj je $(0,\ v_0)$. Na zemlju deluje gravitaciona sila sunca koja je uvek usmerena ka suncu i intenzitet joj je $F_g = \frac{\gamma m M}{r^2}$, pri čemu je r tekuća udaljenost zemlje od sunca tj. $r = \sqrt{x^2 + y^2}$. Na osnovu drugog Newton-ovog zakona je $-\vec{F_g} = m\vec{a}$ tj., ukoliko razložimo ovu jednakost na komponente,

$$\begin{aligned} m\,x'' &= -\frac{\gamma m M}{r^2}\cos(\phi) \\ m\,y'' &= -\frac{\gamma m M}{r^2}\sin(\phi) \end{aligned}$$

Pošto je $\cos(\phi) = \frac{x}{r}$ i $\sin(\phi) = \frac{y}{r}$, dobijamo sistem jednačina drugog reda

$$\begin{aligned} x'' &= -\frac{\gamma M x}{r^3} \\ y'' &= -\frac{\gamma M y}{r^3}, \quad r = \sqrt{x^2 + y^2} \end{aligned}$$

odnosno uvođenjem smena $x' = v_x,\ y' = v_y$,

$$\begin{aligned} x' &= v_x \\ y' &= v_y \\ v_x' &= -\frac{\gamma M x}{r^3} \\ v_y' &= -\frac{\gamma M y}{r^3}, \quad r = \sqrt{x^2 + y^2} \end{aligned}$$

Rešavanjem ovog sistema za početne uslove $x_0 = d_0$, $y_0 = 0$, $v_{x0} = 0$, $v_{y0} = v_0$ na intervalu dužine godinu dana, dobija se tražena putanja.
Datoteka `kepler.m`:

```
function Xp = kepler(t, X)
global gamma; global M;
x = X(1); y = X(2); vx = X(3); vy = X(4);

r = sqrt(x^2+y^2);
Xp = [vx; ...
      vy; ...
      -gamma*M * x / r^3; ...
      -gamma*M * y / r^3];
```

Sistem rešavamo preko

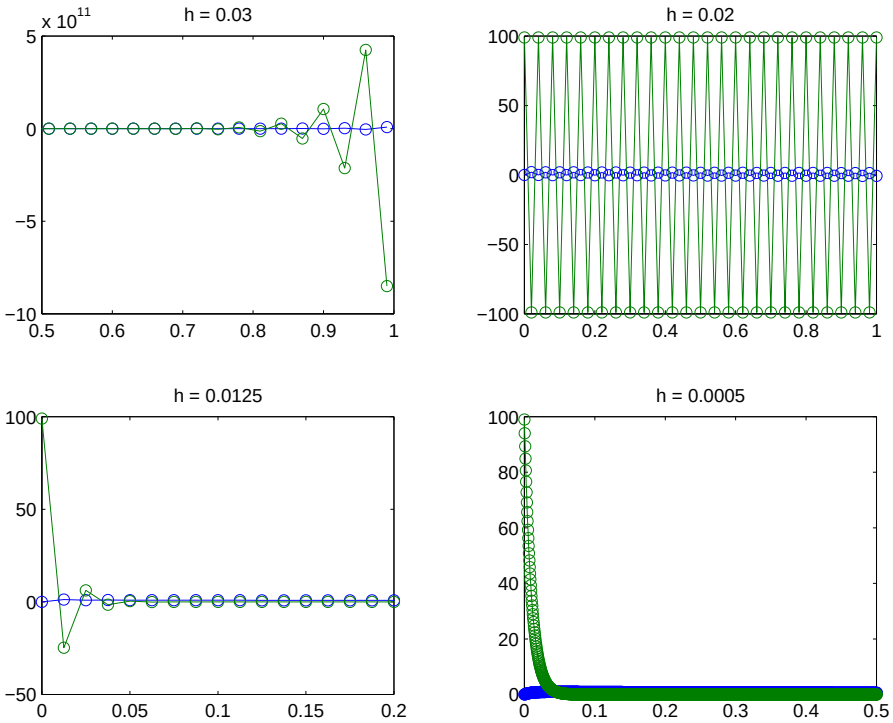
```
global gamma; global M;
gamma = 6.673e-11; M = 2e30; d0 = 147e9; v0 = 30.4e3;
year = 365.25 * 24 * 60 * 60;

[t, X] = ode45('kepler', [0 year], [d0; 0; 0; v0]);
plot(X(:,1), X(:,2)); axis equal
```

Dobijena putanja je elipsa veoma malog ekscentriciteta tako da prilično liči na kružnicu.

8.18 *Korišćenjem* MATLAB-*a, Euler-ovom metodom rešiti sistem jednačina*

$$\begin{array}{rclrcl} u_1'(x) & = & -u_1(x) + u_2(x), & u_0(0) & = & 1 \\ u_2'(x) & = & -100u_2(x), & u_2(0) & = & 99 \end{array}$$

za različite korake h *iz intervala* $[5 \cdot 10^{-1}, 5 \cdot 10^{-4}]$.



Slika 8.2: Nestabilnost Euler-ove metode

Rešenje:

```
% Pocetni uslovi
u0 = [0; 99];
% Interval na kome se jednacina resava
t = [0 1];

% Razliciti koraci
for h = [0.5 0.1 0.05 0.03 0.02 0.0125 0.01 0.005 0.0005]
    % Vrednst resenja u levom kraju postavimo na pocetne uslove
```

```
    u(:,1) = u0;

    % Konstruisemo mrezu odredjenu novim korakom
    x = [t(1):h:t(2)];

    % Eulerovom metodom izracunamo vrednosti resenja na mrezi
    for i = 2:length(x)
        u(:,i) = u(:,i-1) + h*[-u(1,i-1)+u(2,i-1); -100*u(2,i-1)];
    end

    %Iscrtavamo resenje
    plot(x, u, '-o'); title(['h = ' num2str(h)]);
    pause
end
```

Neki rezultati su dati na slici 8.2. Primetimo kako ponašanje pronađenog rešenja za veće korake ne odgovara prirodi stvarnog rešenja sistema. Ovi problemi su inherentni i svojstveni kako za Eulerovu metodu tako i za većinu metoda tipa Runge-Kutta.

Čitaocu se preporučuje da isti problem reši modifikacijama Euler-ove metode (navedenim u uvodnom tekstu) za različiti izbor koraka h.

8.3 Prediktor-korektor metode

Milne-ova metoda definisana je formulama

$$\mathbf{v}_j^* = \mathbf{v}_{j-4} + \frac{4h}{3}(2\mathbf{f}_{j-1} - \mathbf{f}_{j-2} + 2\mathbf{f}_{j-3})$$
$$\mathbf{v}_j = \mathbf{v}_{j-2} + \frac{h}{3}(\mathbf{f}_j^* + 4\mathbf{f}_{j-1} + \mathbf{f}_{j-2})$$

gde je $x_j = x_0 + jh$, $\mathbf{v}_j \approx \mathbf{u}(x_j)$, $\mathbf{f}_j = \mathbf{f}(x_j, \mathbf{v}_j)$ i $\mathbf{f}_j^* = \mathbf{f}(x_j, \mathbf{v}_j^*)$.
Greška se ocenjuje izrazom

$$\|\mathbf{u} - \mathbf{v}\| \approx \frac{1}{29}\|\mathbf{v} - \mathbf{v}^*\|.$$

Adams-ova metoda je definisana formulama

$$\mathbf{v}_j^* = \mathbf{v}_{j-1} + \frac{h}{24}(55\mathbf{f}_{j-1} - 59\mathbf{f}_{j-2} + 37\mathbf{f}_{j-3} - 9\mathbf{f}_{j-4})$$
$$\mathbf{v}_j = \mathbf{v}_{j-1} + \frac{h}{24}(9\mathbf{f}_j^* + 19\mathbf{f}_{j-1} - 5\mathbf{f}_{j-2} + \mathbf{f}_{j-3})$$

gde je $x_j = x_0 + jh$, $\mathbf{v}_j \approx \mathbf{u}(x_j)$, $\mathbf{f}_j = \mathbf{f}(x_j, \mathbf{v}_j)$ i $\mathbf{f}_j^* = \mathbf{f}(x_j, \mathbf{v}_j^*)$.
Greška se ocenjuje izrazom

$$\|\mathbf{u} - \mathbf{v}\| \approx \frac{1}{14}\|\mathbf{v} - \mathbf{v}^*\|.$$

8.19 *Taylor-ovim polinomom šestog stepena odrediti u tačkama* 0.1, 0.2 *i* 0.3 *vrednosti rešenja Cauchy-jevog problema*

$$u' + xu^2 = 0, \qquad u(0) = 1,$$

a zatim Milne-ovom metodom odrediti rešenje i u tačkama 0.4 *i* 0.5. *Računati sa pet decimala.*

Rešenje: Aproksimacija rešenja Taylor-ovim polinomom šestog stepena je

$$w(x) = \sum_{k=0}^{6} \frac{u^{(k)}(0)}{k!} x^k,$$

pri čemu se prvi koeficijent dobija iz početnog uslova, drugi iz diferencijalne jednačine, a ostali uzastopnim diferenciranjem jednačine do izvoda rešenja $u^{(6)}$ i računanjem tih izvoda za $x = 0$. Tako se dobija aproksimacija

$$w(x) = 1 - x^2/2 + x^4/4 - x^6/8.$$

kojom je određen početni odsečak za Milne-ovu metodu. Rezultati izračunavanja su dati sledećom tabelom:

x	0	0.1	0.2	0.3	0.4	0.5
v^*					0.92597	0.88891
v	1	0.99502	0.98039	0.95693	0.92593	0.88885
v'^*					−0.34297	−0.39508
v'	0	−0.09900	−0.19224	−0.27472	−0.34294	

8.20 *Milne-ovom metodom sa tačnošću* $5 \cdot 10^{-4}$ *odrediti* $u(0.25)$ *ako je* $u(x)$ *rešenje Cauchy-jevog problema*

$$u' + u + xu^2 = 0, \qquad u(0) = 1.$$

Početni odsečak odrediti pomoću Taylor-ovog razvoja.

Rešenje: Postupkom opisanim u prethodnom zadatku dobija se Taylor-ov polinom

$$w(x) = 1 - x + 2x^3/3 - 5x^4/12,$$

kojim se određuje početni odsečak. Pošto se pomoću ovog polinoma računaju vrednosti rešenja u intervalu $[0, 0.15]$, nije potrebno koristiti polinom višeg stepena jer je maksimalna vrednost prvog sledećeg člana reda veličine 10^{-5}. Rezultati izračunavanja prikazani su sledećom tabelom

x	0	0.05	0.10	0.15	0.20	0.25
v^*					0.80463	0.75869
v	1	0.95008	0.90063	0.85204	0.80464	0.75871
v'^*					−0.93412	−0.90260
v'	−1	−0.99521	−0.98174	−0.96094	−0.93413	

Kako je $|v - v^*|/29 = 0.6 \cdot 10^{-6}$, gde je v^* procenjena a v korigovana vrednost rešenja u tači $x = 0.25$, tačnost je postignuta i traženo rešenje je $u(0.25) = 0.759$.

8.21 *Adams-ovom metodom na intervalu* $[0, 0.5]$ *sa korakom* $h = 0.1$ *odrediti približno rešenje Cauchy-jevog problema*

$$u'(x) = x\,u^3(x), \qquad u(0) = 1.$$

Početni odsečak odrediti metodom Runge–Kutt-a.

Rešenje: Rezultati računanja dati su sledećom tabelom

x	0.0	0.1	0.2	0.3	0.4	0.5
v^*					1.09086	1.15418
v	1	1.00504	1.02062	1.04828	1.09110	1.15461
v'^*					0.05192	0.07688
v'	0	0.01015	0.02126	0.03456	0.05196	

Vrednosti u tačkama $x = 0.1, 0.2, 0.3$ određene su metodom Runge–Kutt-a, a preostale dve Adams-ovom metodom.

8.22 *Adams-ovom metodom sa tačnošću* $5 \cdot 10^{-5}$ *odrediti* $u(0.5)$*, ako je* $u(x)$ *rešenje Cauchy-jevog problema*

$$xu' = 1 - u + x^2u^2, \qquad u(0) = 1.$$

Početni odsečak odrediti pomoću Taylor-ovog razvoja.

Rešenje: Smenom $w = xu$ polazni problem se svodi na Cauchy-jev problem

$$w' = 1 + w^2, \qquad w(0) = 0.$$

Iz poslednje jednačine je $w'(0) = 1$, a njenim diferenciranjem dobijamo ostale potrebne koeficijente $w^{(k)}(0)$, $k = 2, \ldots, 7$, Taylor-ovog razvoja funkcije $w(x)$

$$w(x) \approx x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \frac{17}{315}x^7$$

(maksimalna vrednost u posmatranom intervalu poslednjeg sabirka je 10^{-5}). Koristeći nađenu aproksimaciju za $w(x)$ i uvedenu smenu, dobijamo aproksimaciju funkcije $u(x)$,

$$u(x) \approx 1 + \frac{1}{3}x^2 + \frac{2}{15}x^4 + \frac{17}{315}x^6 \equiv v(x),$$

pomoću koje nalazimo početni odsečak neophodan za primenu Adams-ove metode. Sa korakom $h = 0.1$ u tački $x = 0.5$ dobijena je procena vrednosti rešenja $v^* = 1.09255$ i korekcija $v = 1.09262$. Kako je $|v - v^*|/14 = 5 \cdot 10^{-6}$ sa traženom tačnošću rešenje je $u(0.5) = 1.0926$.

8.23 *Adams-ovom metodom sa korakom $h = 0.1$ odrediti na intervalu $[1, 1.5]$ približno rešenje Cauchy-jevog problema*

$$u' = \sqrt{xu - 1}, \qquad u(1) = 2.$$

Početni odsečak odrediti metodom Runge–Kutt-a.

Rešenje: Rezultati su prikazani sledećom tabelom

x	1.0	1.1	1.2	1.3	1.4	1.5
v^*					2.518218	2.684604
v	2	2.107434	2.229617	2.366684	2.518180	2.684603
v'^*					1.589184	1.739800
v'	1	1.148119	1.294427	1.441072	1.589167	

8.24 *Adams-ovom metodom sa korakom $h = 0.1$ odrediti na intervalu $[0, 0.5]$ približno rešenje Cauchy-jevog problema*

$$2u''(x) + u'(x) + 4u(x) = 2\cos(2x), \qquad u(0) = 0, \quad u'(0) = 2.$$

Početni odsečak odrediti pomoću Taylor-ovog polinoma funkcije $u(x)$ šestog stepena.

Rešenje: Problem je definisan jednačinom drugog reda, te da bi se primenila Adams-ova metoda moramo ga svesti na sistem jednačina prvog reda. Smenom $u_1 = u$, $u_2 = u'$ dati problem se definiše sistemom

$$\begin{aligned} u_1'(x) &= u_2(x) && u_1(0) = 0, \\ u_2'(x) &= \cos(2x) - 2u_1(x) - \frac{1}{2}u_2(x), && u_2(0) = 2. \end{aligned}$$

Početni odsečak za funkciju $u_1(x) \equiv u(x)$ ćemo izračunati pomoću Taylor-ovog polinoma šestog stepena, a početni odsečak za funkciju $u_2(x) \equiv u'(x)$ pomoću izvoda Taylor-ovog polinoma sedmog stepena. Prva dva koeficijenta Taylor-ovog polinoma su određena zadatim početnim uslovima. Ostale koeficijente nalazimo uzastopnim diferenciranjem polazne jednačine i računanjem tako dobijenih izvoda funkcije $u(x)$ u tački 0,

$$u(0) = 0, \quad u'(0) = 2, \quad u''(0) = 0, \quad u'''(0) = -4, \quad u^{(4)}(0) = -2,$$
$$u^{(5)}(0) = 9, \quad u^{(6)}(0) = 31/2, \quad u^{(7)}(0) = -103/4.$$

Tražene aproksimacije su

$$w_1(x) = 2x - \frac{2}{3}x^3 - \frac{1}{12}x^4 + \frac{3}{40}x^5 + \frac{31}{1440}x^6, \qquad w_2(x) = w_1'(x) - 103x^6/2880.$$

Ovim polinomima izračunamo približne vrednosti funkcija $u_1(x)$ i $u_2(x)$ u tačkama 0.1, 0.2, 0.3 a zatim Adams-ovom formulom približne vrednosti u preostale dve tačke. Rezultati su prikazani sledećom tabelom

x	0.0	0.1	0.2	0.3	0.4	0.5
v_1^*					0.756009	0.914054
v_1	0	0.199326	0.394559	0.581523	0.756050	0.914098
$v_1^{*\prime}$					1.669389	1.485122
v_1'	2	1.979705	1.917972	1.814325	1.669426	
v_2^*					1.669389	1.485122
v_2	2	1.979705	1.917972	1.814325	1.669426	1.485143
$v_2^{*\prime}$					−1.650006	−2.030367
v_2'	0	−0.408438	−0.827043	−1.244873	−1.650106	

FORTRAN Implementacija prediktor-korektor metoda biće prikazana na primeru metode Adams-a. Implementacija se sastoji od prostog prevođenja formula koje opisuju metodu u odgovarajući *Fortran* kod, pri čemu su opet intenzivno korišćene vektorske operacije. Na primeru ove metode, kao i metode Runge–Kutt-a može se dobro videti kako je obično na *Fortran*-u moguće znatno kompaktnije izraziti rešenja numeričkih problema nego što je to slučaj na *C*-u. Procedura `adams()` ima sledeći oblik:

```
module adams_module
  use matheval_module
  implicit none
  public adams

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string

contains
  ! Funkcija adams() odredjuje resenje Cauchy-jevog problema u datoj
  ! tacki ukoliko su poznata resenja u cetiri tacke koje prethode datoj
  ! tacki na istom odstojanju. Argumenti funkcije su:
  !   n - dimenzija sistema
  !   variables - string sa imenom nezavisne promenljive i imenima
  !               funkcija cija se vrednost izracunava
  !   f - vektor sa desnim stranama sistema
  !   x - tacka u kojoj treba naci resenje problema
  !   h - rastojanje medju posmatranim tackama
  !   u - vektor sa vrednostima funkcija u tackama koje prethode datoj
  !       tacki
  !   epsilon - zeljena tacnost
  !   v - vektor u koji ce biti smestene vrednosti funkcija u datoj
  !       tacki
  ! Pretpostavlja se da su sva polja alocirana izvan funkcije. Dimenzija
  ! sistema i tacnost moraju biti veci od 0, dok velicina h mora biti
```

```
! razlicita od 0. Rezultat funkcije je indikator da li je postignuta
! zeljena tacnost. Za racunanje greske je koriscena uniformna norma
! vektora.
logical function adams (n, variables, f, x, h, u, epsilon, v)
  integer, intent(in) :: n
  type(string), intent(in) :: variables
  type(string), dimension(0:n-1), intent(in) :: f
  real(kind=8), intent(in) :: x
  real(kind=8), intent(in) :: h
  real(kind=8), dimension(-4:-1,0:n-1), intent(in) :: u
  real(kind=8), intent(in) :: epsilon
  real(kind=8), dimension(0:n-1), intent(out) :: v
  integer(kind=8), dimension(0:n-1) :: evaluator ! Evaluatori za desne
      ! strane jednacina sistema.
  real(kind=8), dimension(0:n) :: values ! Vektor sa vrednostima
      ! nezavisne promenljive i pocetnim uslovima u jednoj zadatoj
      ! tacki.
  real(kind=8), dimension(-4:-1,0:n-1) :: f_u ! Vrednosti desnih
      ! strana jednacine za zadate tacke.
  real(kind=8), dimension(0:n-1) :: v_asterisk ! Trazene vrednosti
      ! izracunate prediktor formulom.
  real(kind=8), dimension(0:n-1) :: f_asterisk ! Vrednosti desnih
      ! strana jednacina u datoj tacki izracunate na osnovu prediktor
      ! vrednosti.
  integer :: j, k ! Brojaci u petljama.

  ! Proverava se validnost ulaznih podataka. Pritom se kreiraju
  ! evaluatori za desne strane jednacina sistema.
  if (n<=0) stop
  do k=0, n-1
     evaluator(k)=evaluator_create(f(k)%chars)
     if (evaluator(k)==0) stop
  end do
  if (h==0) stop
  if (epsilon<=0) stop

  ! Izracunavaju se vrednosti desnih strana sistema za poznate tacke.
  do j=-4, -1
     values=(/x+j*h,(u(j,k), k=0, n-1)/)
     f_u(j,:)=(/(evaluator_evaluate(evaluator(k),n+1,variables%chars,values),k=0, n-1)/)
  end do

  ! Primenjuje se prediktor formula.
  v_asterisk=(/(u(-1,k)+h/24*(55*f_u(-1,k)-59*f_u(-2,k)+37*f_u(-3,k)-9*f_u(-4,k)), k=0, n-1)/)

  ! Izracunavaju se vrednosti desnih strana sistema za datu tacku na
  ! osnovu vrednosti funkcija dobijenih prediktor formulom.
  values=(/x,v_asterisk/)
  f_asterisk=(/(evaluator_evaluate(evaluator(k),n+1,variables%chars,values), k=0, n-1)/)

  ! Primenjuje se korektor formula.
  v=(/(u(-1,k)+h/24*(9*f_asterisk(k)+19*f_u(-1,k)-5*f_u(-2,k)+f_u(-3,k)), k=0, n-1)/)

  ! Unistavaju se korisceni evaluatori.
  do k=0, n-1
     call evaluator_destroy(evaluator(k))
  end do
```

```
    ! Izracunava se ocena greske i ispituje da li je zadovoljena
    ! tacnost.
    adams=.true.
    do k=0, n-1
       if (abs(v(k)-v_asterisk(k))/14>epsilon) then
          adams=.false.
       end if
    end do
  end function adams
end module adams_module
```

8.25 *a) Integraljenjem jednačine*

$$u' + 2u\cos x = \cos x + \sin 2x, \qquad x \in [0,1]$$

i korišćenjem kvadraturnih formula izvesti diferencijsku šemu

$$\begin{aligned}\frac{1}{h}(v_{i+1}-v_i) + \frac{1}{2}(\cos x_i + \cos x_{i+1})(v_{i+1}+v_i) \qquad & i = 0,\dots,n-1,\\ = \frac{1}{2}(\cos x_i + \cos x_{i+1}) + \frac{1}{2}(\sin 2x_i + \sin 2x_{i+1})&\end{aligned}$$

gde je $h = 1/n$. *Oceniti red tačnosti šeme.*
b) Rešiti jednačinu datu u tački a) za početni uslov $u(0) = 0$ *i korak* $h = 0.2$. *Računati sa četiri decimale.*

Rešenje: a) Definišimo na intervalu $[0,1]$ ravnomernu mrežu

$$\omega = \{x_i \,|\, x_i = ih,\ i = 0,\dots,n,\ h = \frac{1}{n}\}.$$

Integralimo jednačinu na svakom od podintervala definisanih čvorovima mreže ω

$$\int_{x_i}^{x_{i+1}} u'\,dx + 2\int_{x_i}^{x_{i+1}} u\cos x, dx = \int_{x_i}^{x_{i+1}} (\cos x + \sin 2x)\,dx, \quad i = 0,\dots n-1.$$

Integral na desnoj strani jednakosti odredimo sa greškom na jednom koraku $O(h^3)$ trapeznom formulom, a korišćenjem formule pravougaonika imamo da je takođe sa greškom $O(h^3)$

$$2\int_{x_i}^{x_{i+1}} u\cos x\,dx = 2hu(x_{i+1/2})\cos x_{i+1/2} + O(h^3),$$

gde je $x_{i+1/2} = (x_i + x_{i+1})/2$. S obzirom da je

$$\begin{aligned}u(x_i) + u(x_{i+1}) &= (u(x_{i+1/2}) - \frac{h}{2}u'(x_{i+1/2})) + (u(x_{i+1/2}) + \frac{h}{2}u'(x_{i+1/2})) + O(h^2)\\ &= 2u(x_{i+1/2}) + O(h^2)\end{aligned}$$

i slično za funkciju $\cos x$, to je

$$2hu(x_{i+1/2})\cos x_{i+1/2} + O(h^3) = \frac{h}{2}\big(u(x_i) + u(x_{i+1})\big)\big(\cos x_i + \cos x_{i+1}\big) + O(h^3)$$

Kada uvrstimo navedene aproksimacije u integraljenu jednačinu, dobijamo

$$\begin{aligned} &u(x_{i+1}) - u(x_i) + \frac{h}{2}\big(u(x_i) + u(x_{i+1})\big)\big(\cos x_i + \cos x_{i+1}\big) \\ &= \frac{h}{2}(\cos x_i + \cos x_{i+1}) + \frac{h}{2}(\sin 2x_i + \sin 2x_{i+1}) + O(h^3) \end{aligned}$$

što deljenjem sa h i zanemarivanjem člana $O(h^2)$ daje diferencijsku šemu po približnom rešenju v formulisanu zadatkom. Zanemareni član određuje grešku šeme, dakle ona je $O(h^2)$.
b) Na mreži ω za $h = 0.2$, tj. $n = 5$, po formuli

$$\begin{aligned} \big(5 + \frac{1}{2}(\cos x_i + \cos x_{i+1})\big)v_{i+1} &= \big(5 - \frac{1}{2}(\cos x_i + \cos x_{i+1})\big)v_i \\ &= \frac{1}{2}(\cos x_i + \cos x_{i+1} + \sin 2x_i + \sin 2x_{i+1}) \end{aligned}$$

dobijene su sledeće približne vrednosti rešenja

x_i	0	0.2	0.4	0.6	0.8	1.0
v_i	0	0.1978	0.3873	0.5613	0.7127	0.8357
$u(x_i)$	0	0.1987	0.3894	0.5646	0.7174	0.8415

U poslednjoj vrsti tabele date su, poređenja radi, tačne vrednosti rešenja $u(x) = \sin x$ u čvorovima mreže ω.

8.26 *a) Da li diferencijska šema*

$$v_0 = 0, \quad v_1 = 0, \qquad \frac{1}{2h}(v_{i+1} - v_{i-1}) + v_i = i\,h + 1, \quad i = 1, \ldots, n-1,\ h = \frac{1}{n},$$

aproksimira na intervalu $[0, 1]$ *Cauchy-jev problem*

$$u' + u = x + 1, \qquad u(0) = 0,$$

sa greškom reda h^2*? Ako ne, izmeniti diferencijsku šemu tako da je ona drugog reda tačnosti.*
b) Rešiti dati zadatak šemom tačnosti $O(h^2)$ *za* $h = 0.1$.

Rešenje: a) Koristeći Taylorov razvoj funkcije $u(x)$, pod pretpostavkom da je ona dovoljno glatka na intervalu $(0, 1)$, imamo da je

$$\frac{1}{2h}\big(u(x_{i+1}) - u(x_{i-1})\big) + u(x_i) - x_i - 1 = (u' + u - x - 1)_{/x=x_i} + O(h^2)$$

tj. šema je tačnosti $O(h^2)$. Početni uslov $v_0 = 0$ ne unosi grešku u aproksimaciju, jer je početni uslov zadatka $u(0) = 0$. Što se tiče aproksimacije u tački $x = h$, imamo da je s obzirom na zadati problem i zadate početne uslove diskretnog zadatka

$$\begin{aligned} u(x_1) - v_1 &= u(0) + hu'(0) + O(h^2) - v_1 \\ &= u(0) + h(x + 1 - u)_{x=0} + O(h^2) - v_1 = h - v_1 + O(h^2). \end{aligned}$$

tj. greška aproksimacije je reda $O(h)$. Ona će biti reda $O(h^2)$ ako je $v_1 = h$, tj. šema kojom se dati zadatak aproksimira sa tačnošću $O(h^2)$ je

$$(*)\ \ v_0 = 0, \quad v_1 = h, \qquad \frac{1}{2h}(v_{i+1} - v_{i-1}) + v_i = i\,h + 1, \quad i = 1, \ldots, n-1,\ h = \frac{1}{n},$$

b) Modifikovana šema (*) može da se napiše i u obliku

$$v_0 = 0, \quad v_1 = h, \qquad v_{i+1} = 2h(ih + 1 - v_i) + v_{i-1}, \quad i = 1, \ldots n - 1.$$

i njeno rešenje je $v_i = ih$. Zaista, za $i = 1$ tvrđenje je tačno; ako je ono tačno i za $i = k$, onda je

$$v_{k+1} = 2h(kh + 1 - v_k) + v_{k-1} = 2h(kh + 1 - kh) + (k - 1)h = (k + 1)h,$$

te na osnovu matematičke indukcije sledi da je tvrđenje tačno za svako i. Dakle, rešenje zadatka za $h = 0.1$ je $v_i = 0.1i$, $i = 0, 1, ..., 10$. (Ove vrednosti su tačne, jer je tačno rešenje zadatka $u(x) = x$.)

8.27 *Za koji izbor početnog odsečka rešenje određeno troslojnom šemom*

$$v_{j+2} = 4v_{j+1} - 3v_j - 2h\,v'_j, \qquad j = 0, 1, \ldots,$$

konvergira ka rešenju Cauchy-jevog problema

$$u'(x) = u(x), \qquad u(0) = 1,$$

kada $j \to \infty$, $h \to 0$, $jh \to const.$

Rešenje: Kako je šema troslojna, početni odsečak čine vrednosti v_0 i v_1. Iz zadatog početnog uslova je $v_0 = 1$. Još treba odrediti vrednost aproksimacije u tački $x = h$, tj. v_1. S obzirom da je na osnovu diferencijalne jednačine $v'_j = v_j$, diferencna jednačina je

$$v_{j+2} - 4v_{j+1} + (3 + 2h)v_j = 0.$$

Njeno rešenje je

$$v_j = c_1(2 + \sqrt{1 - 2h})^j + c_2(2 - \sqrt{1 - 2h})^j.$$

Partikularno rešenje koje zadovoljava uslov $v_0 = 1$ ima uslov $c_1 + c_2 = 1$. Drugi uslov za konstante c_1 i c_2 se određuje zadavanjem uslova v_1. Kako je rešenje Cauchy-jevog problema funkcija $u(x) = \exp x$, v_1 treba zadati tako da $v_j \to \exp(jh)$ kada $j \to \infty$, $h \to 0$, $jh \to \text{const}$. Kako je

$$v_1 = 2(c_1 + c_2) + (c_1 - c_2)\sqrt{1-2h} = 2 + (2c_1 - 1)\sqrt{1-2h},$$

tj.

$$v_1 = 2 + (2c_1 - 1)(1 - h - \frac{1}{2}h^2 - \frac{1}{2}h^3 + O(h^4)),$$

to za $c_1 = 0$

$$v_1 = 1 + h + \frac{1}{2}h^2 + O(h^3) \underset{h\to 0}{\to} \exp h.$$

Sledi da je tada $c_2 = 1$ i traženi početni uslov je

$$v_1 = 2 - \sqrt{1-2h}.$$

Rešenje diferencne jednačine

$$v_0 = 1, \quad v_1 = 2 - \sqrt{1-2h}, \qquad v_{j+2} = 4v_{j+1} - 3v_j - 2h\, v'_j, \quad j = 0, 1, \ldots,$$

je prema prethodnom

$$v_j = (2 - \sqrt{1-2h})^j = 1 + jh + \frac{1}{2!}(jh)^2 + \frac{1}{3!}(jh)^3 + \cdots + o(h),$$

što teži e^x kada $h \to 0$.

8.28 *Za približno rešavanje Cauchy-jevog problema*

$$u'(x) = f(x,u), \qquad u(x_0) = u_0,$$

izvesti formulu što je moguće višeg reda tačnosti oblika

$$(a) \qquad v_{j+1} = a\, v_j + b\, v_{j-1} + h\, c\, v'_j,$$

$$(b) \qquad v_{j+1} = a\, v_j + h(b\, v'_j + c\, v'_{j-1}).$$

Rešenje: Kako formule sadrže tri slobodna parametra, njih treba odrediti tako da je greška na jednom koraku

$$R(x_{j+1}) = u(x_{j+1}) - v_{j+1} = O(h^3).$$

Pretpostavimo da su u prethodnim čvorovima vrednosti funkcije i izvoda izračunati tačno, tj. da je $v_k = u(x_k)$ i $v'_k = u'(x_k)$ za $k \le j$, i da je funkcija $u(x)$ dovoljno glatka. Korišćenjem Taylorovih razvoja funkcija $u(x)$ i $u'(x)$ u okolini tačke x_j dobijamo da je greška formule (a)

$$R(x_{j+1}) = (1 - a - b)u(x_j) + h(1 + b - c)u'(x_j) + \frac{h^2}{2}(1 - b)u''(x_j) + O(h^3).$$

Formula će biti drugog reda tačnosti, tj. greška će biti $O(h^3)$, ako su koeficijenti u grešci uz h^k, $k = 0, 1, 2$, jednaki nuli, a to je za $a = 0$, $b = 1$ i $c = 2$. Na isti način, za formulu (b) dobijamo da je $a = 1$, $b = 3/2$ i $c = -1/2$. Tražene formule su stoga

$$\text{(a)} \qquad v_{j+1} = v_{j-1} + 2hv'_j,$$
$$\text{(b)} \qquad v_{j+1} = v_j + \frac{h}{2}(3v'_j - v'_{j-1}).$$

8.29 *Za približno rešavanje Cauchy-jevog problema*

$$u'(x) = f(x, u), \qquad u(x_0) = u_0,$$

izvesti formulu petog reda tačnosti oblika

$$(*) \qquad v_{j+1} = a\,v_j + b\,v_{j-1} + c\,v_{j-2} + d\,v_{j-3} + h(e\,v'_{j+1} + f\,v'_j + g\,v'_{j-1})$$

gde je b slobodni parametar. Odrediti vrednosti ostalih parametara ako je $b = 1$.

Rešenje: Parametre treba odrediti tako da je greška na jednom koraku

$$R(x_{j+1}) = u(x_{j+1}) - v_{j+1} = O(h^6).$$

Pretpostavimo da su u prethodnim čvorovima vrednosti funkcije i izvoda izračunati tačno, tj. da je $v_k = u(x_k)$ i $v'_k = u'(x_k)$ za $k \le j$, i da je funkcija $u(x)$ dovoljno glatka. Korišćenjem Taylorovih razvoja funkcija $u(x)$ i $u'(x)$ u okolini tačke x_j dobijamo da je greška formule $(*)$

$$\begin{aligned} R(x_{j+1}) = {} & (1 - a - b - c - d)u(x_j) + h(1 + b + 2c + 3d - e - f - g)u'(x_j) \\ & + \frac{h^2}{2}(1 - b - 4c - 9d - 2e + 2g)u''(x_j) \\ & + \frac{h^3}{6}(1 + b + 8c + 27d - 3e - 3g)u'''(x_j) \\ & + \frac{h^4}{24}(1 - b - 16c - 81d - 4e + 4g)u^{(4)}(x_j) \\ & + \frac{h^5}{120}(1 + b + 32c + 243d - 5e - 5g)u^{(5)}(x_j) + O(h^6). \end{aligned}$$

Formula će biti petog reda tačnosti, tj. greška će biti $O(h^6)$, ako su koeficijenti u grešci uz h^k, $k = 0, \ldots, 5$, jednaki nuli, a to je za

$$a = \frac{3276 - 2891b}{2142}, \qquad c = \frac{13b - 20}{34}, \qquad d = \frac{63 - 35b}{1071},$$
$$e = \frac{12 - b}{34}, \qquad f = \frac{252 + 217b}{357}, \qquad g = \frac{37b - 36}{34}.$$

Za $b = 1$ formula $(*)$ je

$$v_{j+1} = 0.18v_j + v_{j-1} - 0.21v_{j-2} + 0.03v_{j-3} + h(0.32v'_{j+1} + 1.31v'_j + 0.03v'_{j-1}).$$

8.30 *Za približno rešavanje Cauchy-jevog problema*

$$u'(x) = f(x,u), \qquad u(x_0) = u_0,$$

odrediti formule četvrtog reda tačnosti oblika

$$v_{j+1} = a_k v_{j-k} + h(b_k v'_j + c_k v'_{j-1} + d_k v'_{j-2} + e_k v'_{j-3}), \quad k = 0,1,2,3.$$

Rešenje: Postupkom opisanim u prethodnom zadatku dobijamo da će metode biti četvrtog reda tačnosti ako je

$$a_k = 1, \qquad b_k = \frac{1}{24}\left(55+24k-22k^2+8k^3-k^4\right), \qquad c_k = \frac{1}{24}\left(-59+36k^2-20k^3+3k^4\right)$$

$$d_k = \frac{1}{24}\left(37-18k^2+16k^3-3k^4\right) \qquad e_k = \frac{1}{24}\left(-9+4k^2-4k^3+k^4\right).$$

Zamenjujući u izrazima za koeficijente redom vrednosti $k = 0,1,2$ i 3 dobijamo formule

$$k=0 \qquad v_{j+1} = v_j + \frac{h}{24}\left(55v'_j - 59v'_{j-1} + 37v'_{j-2} - 9v'_{j-3}\right),$$

$$k=1 \qquad v_{j+1} = v_{j-1} + \frac{h}{3}\left(8v'_j - 5v'_{j-1} + 4v'_{j-2} - v'_{j-3}\right),$$

$$k=2 \qquad v_{j+1} = v_{j-2} + \frac{h}{8}\left(21v'_j - 9v'_{j-1} + 15v'_{j-2} - 3v'_{j-3}\right),$$

$$k=3 \qquad v_{j+1} = v_{j-3} + \frac{h}{3}\left(2v'_j - v'_{j-1} + 2v'_{j-2}\right),$$

Napomena: Formula dobijena za $k=0$ je poznata Adams-ova, a za $k=3$ Milne-ova prediktor formula.

8.31 *Korišćenjem Newtonove interpolacione formule za interpolaciju unazad izvesti a) prediktor formulu oblika*

$$v_{j+1} = v_{j-1} + \sum_{k=0}^{p} c_k v'_{j-k},$$

b) korektor formulu oblika

$$v_{j+1} = v_j + \sum_{k=-1}^{p} c_k v'_{j-k},$$

za $p = 0,1,2$. Oceniti greške izvedenih formula.

Rešenje: a) Kako je

$$u(x_{j+1}) = u(x_{j-1}) + \int_{x_{j-1}}^{x_{j+1}} u'(x)\,dx,$$

aproksimacijom izvoda Newtonovim interpolacionim polinomom za interpolaciju unazad (II Newtonov polinom) napisanim u odnosu na čvor x_j dobijamo da je $u(x_{j+1})$ približno jednako

$$v_{j+1} = v_{j-1} + \int_{x_{j-1}}^{x_{j+1}} \Big(v'_j + q\Delta v'_{j-1} + \frac{1}{2!}q(q+1)\Delta^2 v'_{j-2} + \cdots + \frac{1}{p!}q(q+1)\dots(q+p-1)\Delta^p v'_{j-p} \Big)\, dx,$$

gde je $x = x_j + qh$. Greška se određuje integraljenjem greške interpolacione formule (napisane za u')

$$|R_p| = \left| \int_{x_{j-1}}^{x_{j+1}} \frac{1}{(p+1)!} u^{(p+2)}(\xi(x)) h^{p+1} q(q+1)\dots(q+p)\, dx \right|$$
$$\le \frac{h^{p+1}}{(p+1)!} \max_{[x_{j-p}, x_{j+1}]} |u^{(p+2)}| \int_{x_{j-1}}^{x_{j+1}} |q(q+1)\dots(q+p)|\, dx$$

Tako se dobijaju formule

$$\underline{p=0,1} \qquad v_{j+1} = v_{j-1} + 2h\, v'_j, \qquad |R_{0/1}| \le \frac{h^3}{3} \max |u'''|,$$
$$\underline{p=2} \qquad v_{j+1} = v_{j-1} + \frac{h}{3}(7v'_j - 2v'_{j-1} + v'_{j-2}), \qquad |R_2| \le \frac{h^4}{3} \max |u^{(4)}|,$$

b) Da bismo dobili korektor formule traženog oblika integralimo izvod u intervalu $[x_j, x_{j+1}]$,

$$u(x_{j+1}) = u(x_j) + \int_{x_j}^{x_{j+1}} u'(x)\, dx,$$

i zamenimo podintegralnu funkciju II Newtonovim interpolacionim polinomom stepena $p+1$ napisanim u odnosu na čvor x_{j+1}

$$v_{j+1} = v_j + \int_{x_j}^{x_{j+1}} \Big(v'_{j+1} + q\Delta v'_j + \frac{1}{2!}q(q+1)\Delta^2 v'_{j-1} + \cdots + \frac{1}{(p+1)!}q(q+1)\dots(q+p)\Delta^{p+1} v'_{j-p} \Big)\, dx,$$

gde je $x = x_{j+1} + qh$. Greška se određuje integraljenjem greške interpolacione formule

$$|R_{p+1}| = \left| \int_{x_j}^{x_{j+1}} \frac{1}{(p+2)!} u^{(p+3)}(\xi(x)) h^{p+2} q(q+1)\dots(q+p+1)\, dx \right|$$
$$\le \frac{h^{p+2}}{(p+2)!} \max_{[x_{j-p}, x_{j+1}]} |u^{(p+3)}| \int_{x_j}^{x_{j+1}} |q(q+1)\dots(q+p+1)|\, dx$$

Tako se dobijaju formule

$$\underline{p=0} \qquad v_{j+1} = v_j + \frac{h}{2}(v'_{j+1} + v'_j), \qquad |R_0| \le \frac{h^3}{12} \max|u'''|,$$

$$\underline{p=1} \qquad v_{j+1} = v_j + \frac{h}{12}(5v'_{j+1} + 8v'_j - v'_{j-1}), \qquad |R_1| \le \frac{h^4}{24} \max|u^{(4)}|,$$

$$\underline{p=2} \qquad v_{j+1} = v_j + \frac{h}{24}(9v'_{j+1} + 19v'_j - 5v'_{j-1} + v'_{j-2}), \quad |R_2| \le \frac{19h^5}{720} \max|u^{(5)}|,$$

9

Obične diferencijalne jednačine – granični problemi

Rešavaćemo granični problem oblika

$$-u''(x) + q(x)u(x) = f(x)$$

$$\alpha_1 u'(a) + \beta_1 u(a) = 0, \qquad \alpha_2 u'(b) + \beta_2 u(b) = 0$$

gde je $q(x) \geq 0$ i $\alpha_i^2 + \beta_i^2 \neq 0$, $i = 1, 2$.

9.1 Metoda gađanja

Metodom gađanja se rešenje graničnog problema traži u obliku linearne kombinacije rešenja dva Cauchy-jeva problema

$$u(x) = u_1(x) + c\, u_2(x).$$

Funkcija $u_1(x)$ zadovoljava datu jednačinu, a $u_2(x)$ odgovarajuću homogenu jednačinu. Početne uslove treba izabrati tako da navedena linearna kombinacija zadovoljava levi granični uslov polaznog zadatka za svaku vrednost konstante c.

9.1 *Metodom gađanja rešiti granični problem*

$$u''(x) + u(x) = x, \qquad u(0) = 2, \quad u(0.5) = 0.$$

Rešenje: Ako se $u(x)$ predstavi linearnom kombinacijom $u(x) = u_1(x) + cu_2(x)$, gde je c proizvoljna konstanta, jednačina postaje

$$\big(u_1''(x) + u_1(x)\big) + c\big(u_2''(x) + u_2(x)\big) = x.$$

Ona će biti identički zadovoljena za svako c ako je

$$u_1''(x) + u_1(x) = x, \qquad u_2''(x) + u_2(x) = 0.$$

Slično, granični uslov će biti zadovoljen za svako c u levom kraju intervala

$$u_1(0) + cu_2(0) = 2, \qquad \text{ako je} \qquad u_1(0) = 2, \quad u_2(0) = 0.$$

Drugi početni uslovi se određuju proizvoljno, uz ograničenje da je $u_2'(0) \neq 0$ (da ne bi bilo $u_2(x) \equiv 0$). Ako se zada da je $u_1'(0) = 0$ i $u_2'(0) = 1$ i reše odgovarajući Cauchy-jevi problemi (recimo metodom Runge–Kutt-a) dobija se $u_1(0.5) = 1.77574$ i $u_2(0.5) = 0.47943$. Konstanta c se određuje iz desnog graničnog uslova

$$u(0.5) = u_1(0.5) + cu_2(0.5) = 0 \qquad \Longrightarrow \qquad c = -\frac{u_1(0.5)}{u_2(0.5)} = -3.7039$$

Rešenje polaznog zadatka je $u(x) = u_1(x) - 3.7039u_2(x)$.

9.2 *Metodom gađanja odrediti rešenje graničnog problema*

$$u'' + 3u' + 2u = 12\,e^{2x}$$
$$u(0) - u'(0) = 4, \qquad u(1) + u'(1) = 22.0318.$$

Rešenje: Rešenje problema $u(x)$ se traži u obliku linearne kombinacije

$$u(x) = u_1(x) + c\,u_2(x),$$

rešenja $u_k(x)$, $k = 1, 2$, dva Cauchy-jeva problema. Cauchy-jevi problemi su definisani zahtevom da navedena linearna kombinacija zadovoljava jednačinu i granicni uslov u levom kraju intervala za svaki izbor konstante c. Tako iz jednačine

$$(u_1'' + cu_2'') + 3(u_1' + cu_2') + 2(u_1 + cu_2) = 12\,e^{2x}$$

dobijamo jednačine koje treba da zadovoljavaju funkcije u_1 i u_2,

$$u_1'' + 3u_1' + 2u_1 = 12\,e^{2x}, \qquad u_2'' + 3u_2' + 2u_2 = 0.$$

Takođe, iz zadatog graničnog uslova u levom kraju intervala je

$$(u_1 + cu_2) - (u_1' + cu_2') = 4 \quad \longrightarrow \quad u_1(0) - u_1'(0) = 4, \quad u_2(0) - u_2'(0) = 0.$$

Početne uslove u tački 0 treba izabrati tako da zadovoljavaju dobijene veze, uz uslov da je $u_2(0) \neq 0$, jer bi u protivnom bila funkcija $u_2(x) \equiv 0$ kao rešenje homogenog problema.

Na osnovu prethodnih zaključaka, definišimo Cauchy-jeve probleme

$$u_1'' + 3u_1' + 2u_1 = 12\,e^{2x}, \qquad u_1(0) = 4, \quad u_1'(0) = 0,$$
$$u_2'' + 3u_2' + 2u_2 = 0, \qquad u_2(0) = u_2'(0) = 1.$$

Metodom Runge–Kutt-a dobijamo da je $u_1(1) = 8.7252$, $u_1'(1) = 13.5773$ i $u_2(1) = 0.8330$, $u_2'(1) = -0.5623$. Konstantu c nalazimo tako da funkcija $u(x)$ zadovoljava i drugi granični uslov

$$\big(u_1(1) + u_1'(1)\big) + c\big(u_2(1) + u_2'(1)\big) = 22.0318.$$

Tako se dobija da je $c = -1.0001$ pa je traženo rešenje

$$u(x) = u_1(x) - u_2(x).$$

Njegove numeričke vrednosti se mogu izračunati u tačkama u kojima su određena rešenja navedenih Cauchy-jevih problema.

MATLAB

Prikažimo implementaciju metoda gađanja u MATLAB-u kroz sledeći zadatak.

9.3 *Korišćenjem* MATLAB-a, *metodom gađanja rešiti sledeće granične probleme:*

i) $y'' + x^2 = y + x + 1, \qquad y'(0) = -y(0), \quad y'(1) = y(1)$

ii) $y'' + y^2 = 1, \qquad y(0) = 0, \quad y(\frac{\pi}{2}) = 1$

Rešenje:

i) Pridružimo polaznoj jednačini dva Cauchy-jeva problema:

$$\begin{array}{rclcrcl} y_1' &=& y_2, & & y_1(0) &=& A_1 \\ y_2' &=& y_1 - x^2 + x - 1, & & y_2(0) &=& -A_1 \end{array}$$

i drugi, homogeni

$$\begin{array}{rclcl} y_1' &=& y_2, & & y_1(0) = A_2 \\ y_2' &=& y_1, & & y_2(0) = -A_2 \end{array}$$

Kombinovanjem rešenja ova dva sistema se dobija rešenje polazne jednačine.

Opisane sisteme kodiramo kroz dve pomoćne funkcije i rešavamo ih korišćenjem `ode45`.

```
function shooting_linear
% Pocetni uslovi su oblika :
%     y'(0) - sigma1 y(0) = 0,     y'(1) + sigma2 y(1) = 0
sigma1 = -1; sigma2 = -1;

% Prvi Kosijev problem resavamo adaptivno na intervalu [0, 1]
[x, y1] = ode45(@nonhomogenous, [0 1], [-1 -sigma1])

% Resenja drugog problema trazimo u tackama vektora x u kojim znamo
% resenje prvog problema
[x, y2] = ode45(@homogenous, x, [ 1  sigma1])

% Kombinujemo dobijena resenja
```

```
c = - y1(end,2)+sigma2*y1(end,1) /...
      y2(end,2)+sigma2*y2(end,1);
y = y1(:,1) + c*y2(:,1);

% Iscrtavamo dobijeno resenje
plot(x, y, '-o');

% Funkcija zadaje jednacinu y''-y  = 0
function yp = homogenous(x,y)
yp = [y(2); ...
      y(1)];

% Funkcija odredjuje jednacinu y''-y = -x^2+x+1
function yp = nonhomogenous(x,y)
yp = [y(2); ...
      y(1)-x^2+x+1];
```

ii) Za rešavanje nelinearnog problema koristimo iterativnu metodu gađanja uz tehniku binarne pretrage tj. polovljenja intervala. Sistem jednačina opisujemo kroz pomoćnu funkciju `system` i za njegovo rešavanje koristimo `ode45`.

```
function shooting_nonlinear
% Interval na kome se jednacina resava
a = 0; b = pi/2;
% Granicni uslovi: y(a) = A, y(b) = B
A = 0; B = 1;
% Tacnost
epsilon = 1e-3;
% Funkcija koja opisuje sistem
f = @system;

% Pocetni Kosijevi problemi
% s0 i s1 su odredjeni proizvoljno uz uslov da
% je za prvi sistem y(b) > B, a za drugi y(b) < B
s0 = -2;
[x,y] = ode45(f, [a b], [A s0]);
hold on; plot(x, y(:,1), 'r');

s1 = 2;
[x,y] = ode45(f, [a b], [A s1]);
plot(x, y(:,1), 'r'); pause;


% Iteracija se vrsi dok se drugi granicni uslov
% ne ispuni sa dovoljnom tacnoscu
while (abs(y(end, 1)-B) > epsilon)
    % Tehnika binarne pretrage
    s = (s0+s1)/2;

    % Resavamo Kosijev problem
    [x,y]=ode45(f, [a b], [A s]);
    plot(x, y(:,1)); pause;

    if (y(end,1) < B)
        s0 = s;
    else
```

```
        s1 = s;
    end
end

% Iscrtavamo konacno resenje
plot(x, y(:, 1), 'r');


% Pomocna funkcija koja predstavlja sistem diferencijalnih jednacina
function yp = system (x, y)
yp = [ y(2); ...
      -y(1)^2+1 ];
```

9.2 Metoda konačnih razlika

Metodom mreže polazni granični problem aproksimira se na intervalu $[a, b]$ na mreži

$$\omega = \{x_i \,|\, x_i = a + ih,\ i = 0, \dots n,\ h = \frac{b-a}{n}\}$$

diferencijskom šemom

$$\begin{aligned} &\alpha_1 v_{x,0} + \beta_1 v_0 = 0 \\ &-v_{\overline{x}x,i} + q_i v_i = f_i, \qquad i = 1, \dots, n-1, \\ &\alpha_2 v_{\overline{x},n} + \beta_2 v_n = 0 \end{aligned}$$

Moguće aproksimacije izvoda pomoću približnih vrednosti rešenja u čvorovima mreže $v_i \approx u(x_i)$, sa navedenim redom greške, su

Prvi izvod $u'(x_i)$

$$v_{x,i} = \frac{1}{h}\big(v_{i+1} - v_i\big) \quad \text{ili} \quad v_{\overline{x},i} = \frac{1}{h}\big(v_i - v_{i-1}\big) \qquad O(h)$$

$$v_{\dot{x},i} = \frac{1}{2h}\big(v_{i+1} - v_{i-1}\big) = \frac{1}{2}(v_{\overline{x},i} + v_{x,i}) \qquad O(h^2)$$

$$\frac{1}{2h}(-3v_i + 4v_{i+1} - v_{i+2}) \quad \text{ili} \quad \frac{1}{2h}(v_{i-2} - 4v_{i-1} + 3v_i) \quad O(h^2)$$

Drugi izvod $u''(x_i)$

$$v_{\overline{x}x,i} = \frac{1}{h^2}\big(v_{i+1} - 2v_i + v_{i-1}\big) \quad O(h^2)$$

$$v_{\overline{x}x,i} - \frac{h^2}{12} v_{\overline{x}x\overline{x}x,i} \qquad O(h^4)$$

Mešoviti granični problem ($p(x) > 0$, $q(x) \geq 0$, $\sigma_1 > 0$, $\sigma_2 > 0$)

$$-\big(p(x)u'(x)\big)' + q(x)u(x) = f(x)$$
$$p(a)u'(a) - \sigma_0 u(a) = 0, \qquad p(b)u'(b) + \sigma_1 u(b) = 0$$

aproksimira se sa tačnošću $O(h^2)$ diferencijskom šemom

$$\frac{2}{h}\big(-\frac{p_0+p_1}{2}v_{x,0} + \sigma_0 v_0\big) + q_0 v_0 = f_0$$
$$-\frac{1}{2}\big((p\,v_x)_{\overline{x},i} + (p\,v_{\overline{x}})_{x,i}\big) + q_i v_i = f_i, \qquad i = 1, \ldots, n-1,$$
$$\frac{2}{h}\big(\frac{p_n+p_{n-1}}{2}v_{\overline{x},n} + \sigma_1 v_n\big) + q_n v_n = f_n$$

9.4 *Metodom konačnih razlika sa korakom $h = 0.2$ rešiti granični problem*

$$u'' + u + x = 0, \qquad u(0) = u(1) = 0.$$

Računati na četiri sigurne cifre.

Rešenje: Diferencijska šema predstavlja trodijagonalni sistem jednačina

$$\begin{array}{rcl} v_0 & = & 0 \\ 25\,(v_2 - 2v_1 + v_0) + v_1 + 0.2 & = & 0 \\ 25\,(v_3 - 2v_2 + v_1) + v_2 + 0.4 & = & 0 \\ 25\,(v_4 - 2v_3 + v_2) + v_3 + 0.6 & = & 0 \\ 25\,(v_5 - 2v_4 + v_3) + v_4 + 0.8 & = & 0 \\ v_5 & = & 0 \end{array}$$

Njegovim rešavanjem dobija se da su približne vrednosti rešenja

i	0	1	2	3	4	5
x_i	0	0.2	0.4	0.6	0.8	1
v_i	0	0.03623	0.06302	0.07128	0.05269	0

(Poređenja radi, tacno rešenje zadatka je $u(x) = \sin x / \sin 1 - x$.)

9.5 *Metodom konačnih razlika sa korakom $h = 0.2$ rešiti granični problem*

$$u'' + x^2 u + 2 = 0, \qquad u(-1) = u(1) = 0.$$

Računati na četiri decimale.

Rešenje: Koeficijenti jednačine, oblast definisanosti problema i granični uslovi su simetrični u odnosu na koordinatni početak, te je rešenje parna funkcija. Stoga

se problem može rešavati samo na intervalu $[0,1]$. Diskretizacijom zadatka, dobija se sistem linearnih jednačina

$$-25v_{i-1} + (50 - x_i^2)v_i - 25v_{i+1} = 2, \qquad i = 1,2,3,4,$$

gde je $v_i \approx u(x_i)$ i $x_i = 0.2i$. Rešenje ovog sistema je

$$v_0 = 1.0533, \quad v_1 = 1.0133, \quad v_2 = 0.8916, \quad v_3 = 0.6843, \quad v_4 = 0.3871, \quad v_5 = 0.$$

9.6 *a)Dokazati da je, ako* $u(x) \in C^6(0,1)$,

$$u_{\overline{x}x} = u'' + \frac{h^2}{12}u^{(4)} + O(h^4).$$

b) Konstruisati diferencijsku šemu tačnosti $O(h^4)$ *za granični problem*

$$\begin{aligned} a\,u''(x) + b\,u(x) &= f(x), \qquad\qquad (a, b = const.\ , a \neq 0) \\ u(0) = A, &\qquad u(1) = B. \end{aligned}$$

c) Koristeći šemu dobijenu pod b), rešiti granični problem

$$u'' + u + x = 0, \qquad u(0) = u(1) = 0$$

ako je korak mreže $h = 0.2$.

Rešenje: a) Razvojem u Taylor-ov red do šestog izvoda oko tačke x veličina $u(x+h)$ i $u(x-h)$ u izrazu

$$u_{\overline{x}x} = \frac{1}{h^2}(u(x+h) - 2u(x) + u(x-h))$$

dokazujemo da je

$$u_{\overline{x}x} = u'' + \frac{h^2}{12}u^{(4)} + O(h^4). \qquad (*)$$

b) Iz jednačine je

$$u'' = \frac{1}{a}(-b\,u + f) \qquad \text{i} \qquad u^{(4)} = -\frac{b}{a^2}(-b\,u + f) + \frac{1}{a}f''.$$

Zamenom dobijenih izraza u $(*)$, zanemarivanjem sabirka $O(h^4)$ i uzimanjem u obzir zadatih graničnih uslova dobijamo traženu diferencijsku šemu

$$v_{\overline{x}x} + \frac{b}{a}\left(1 - \frac{b}{a}\frac{h^2}{12}\right)v = \frac{1}{a}\left(1 - \frac{b}{a}\frac{h^2}{12}\right)f + \frac{h^2}{12a}f''$$
$$v_0 = A, \qquad v_n = B.$$

c) Korišćenjem ove šeme za $a = b = 1$, $f(x) = -x$, $A = B = 0$ i $h = 0.2$ dobijamo sledeći sistem linearnih jednačina

$$-v_{i-1} + 1.9601v_i - v_{i+1} = 0.0399x_i, \qquad i = 1,2,3,4.$$

Njegovim rešavanjem Gauss-ovom metodom eliminacije dobijamo približno rešenje

$$v_1 = 0.03615, \quad v_2 = 0.06286, \quad v_3 = 0.07106, \quad v_4 = 0.05253.$$

Uporediti dobijeni rezultat sa rezultatom prethodnog zadatka, kao i sa tačnim rešenjem $u(x) = \sin x / \sin 1 - x$. Greška dobijenih vrednosti nije veća od 10^{-5}.

9.7 *Šemom povišene tačnosti sa korakom $h = 0.2$ rešiti granični problem*

$$u''(x) = \sin x, \qquad u(0) = 1, \quad u(1) = 0.$$

Računati sa pet sigurnih cifara.

Rešenje: Stavljajući u prethodnom zadatku $a = 1$, $b = 0$ i $f(x) = \sin x$, dobijamo diferencijsku šemu

$$v_0 = 1, \qquad v_{\overline{x}x,i} = \big(1 - \frac{h^2}{12}\big) \sin x_i, \quad i = 1, 2, 3, 4, \qquad v_5 = 0,$$

čije rešenje je

$$v_0 = 1, \quad v_1 = 0.76962, \quad v_2 = 0.54717, \quad v_3 = 0.34024, \quad v_4 = 0.15582, \quad v_5 = 0.$$

9.8 *Šemom povišene tačnosti sa korakom $h = 0.25$ rešiti granični problem*

$$u''(x) = u + e^{-x^2/2}, \qquad u(-1) = u(1) = 0.$$

Računati sa četiri decimale.

Rešenje: Stavljajući u zadatku 5. da je $a = 1$, $b = -1$ i $f(x) = e^{-x^2/2}$, dobijamo približno rešenje datog graničnog problema

$$\begin{aligned} u(0) &\approx -0.3269, \\ u(-0.25) = u(0.25) &\approx -0.3058, \\ u(-0.50) = u(0.50) &\approx -0.2433, \\ u(-0.75) = u(0.75) &\approx -0.1408, \\ u(-1) = u(1) &= 0. \end{aligned}$$

9.9 a) *Za koje α, β i γ diferencijski problem*

$$v_0 = v_n = 0,$$

$$\Lambda v \equiv \frac{1}{h^2}(-v_{i+1} + 2v_i - v_{i-1}) + (\alpha v_{i+1} + \beta v_i + \gamma v_{i-1}) = f(x_i) + \frac{h^2}{12} f''(x_i), \qquad i = 1, \ldots, n-1,$$

gde je $x_i = i\,h$, $h = 1/n$, $i = 0, \ldots, n$, *aproksimira granični problem*

$$-u''(x) + u(x) = f(x), \qquad u(0) = u(1) = 0$$

sa greškom $O(h^4)$*?*
b) Ovom šemom rešiti granični problem

$$-u''(x) + u(x) = -e^x, \qquad u(0) = u(1) = 0,$$

ako je $h = 0.2$. *Računati sa pet decimala.*

Rešenje: a) Neka je $z = u - v$ funkcija greške koja je definisana u čvorovima mreže. Ona zadovoljava isti diferencijski zadatak kao i funkcija v, samo sa drugačijim slobodnim članom. U graničnim čvorovima mreže je $z_0 = z_n = 0$, a u unutrašnjim

$$\Lambda z = \Lambda(u - v) = \Lambda u - f(x) - \frac{h^2}{12} f''(x).$$

Ako $u \in C^6(0,1)$, razvojem u Taylorov red oko tačke x veličina $u(x \pm h)$ dobijamo da je

$$\Lambda z = \big(-u'' + (\alpha+\beta+\gamma)u - f\big) + \frac{h^2}{12}\big(-u'' + 6(\alpha+\gamma)u - f\big)'' + h(\alpha-\gamma)\big(u - \frac{h^2}{6}u''\big)' + O(h^4),$$

Koristeći datu jednačinu imamo da je $\Lambda z = O(h^4)$ ako je

$$\alpha + \beta + \gamma = 1, \qquad 6(\alpha+\gamma) = 1, \qquad \alpha - \gamma = 0,$$

tj. $\alpha = \gamma = 1/12$ i $\beta = 5/6$.

b) Kada uvrstimo vrednosti parametara određene u tački (a), dobijamo diferencijsku šemu

$$v_0 = v_n = 0,$$
$$-\frac{1}{h^2}(v_{i+1} - 2v_i + v_{i-1}) + \frac{1}{12}(v_{i+1} + 10v_i + v_{i-1}) = f(x_i) + \frac{h^2}{12} f''(x_i), \qquad i = 1, \ldots, n-1,$$

gde je $n = 1/h = 5$ i $f(x) = -e^x$. Rešenja trodijagonalnog sistema jednačina koji definiše ova šema data su u tabeli koja sledi. S obzirom da je data jednačina jednačina drugog reda sa konstantnim koeficijentima, lako se rešava analitički. Tačno rešenje je u svim čvorovima mreže identično približnom na pet decimala.

x_i	0.0	0.2	0.4	0.6	0.8	1.0
v_i	0	−0.11071	−0.17668	−0.18967	−0.13690	0

9.10 *Dat je granični problem*

$$-u'' + (x+1)u' + u = 3x^2, \qquad u(0) = 3, \quad u'(1) = 1.$$

Diskretizovati granični problem na ravnomernoj mreži koraka $h = 0.25$ diferencijskom šemom tačnosti $O(h^2)$ i rešiti diskretni problem. Računati sa četiri decimale.

Rešenje: Aproksimacija jednačine diferencijskom šemom tačnosti $O(h^2)$ u unutrašnjim čvorovima je

$$-\frac{1}{h^2}(v_{i-1} - 2v_i + v_{i+1}) + (x_i + 1)\,\frac{1}{2h}(v_{i+1} - v_{i-1}) + v_i = 3x_i^2, \qquad i = 1, 2, 3.$$

Na osnovu graničnog uslova u levom kraju intervala sledi da je $v_0 = 3$. Za aproksimaciju prvog izvoda u desnom kraju intervala koristimo razvoj

$$u'(x_4) = \frac{1}{2h}\big(u(x_2) - 4u(x_3) + 3u(x_4)\big) + O(h^2),$$

koji određuje nesimetričnu aproksimaciju ovog graničnog uslova tačnosti $O(h^2)$

$$\frac{1}{2h}(v_2 - 4v_3 + 3v_4) = 1.$$

Sređivanjem ovih jednačina dobija se sistem

$$\begin{aligned} v_0 &= 3 \\ -18.5v_0 + 33v_1 - 13.5v_2 &= 0.1875 \\ -19v_1 + 33v_2 - 13v_3 &= 0.75 \\ -19.5v_2 + 33v_3 - 12.5v_4 &= 1.6875 \\ 2v_2 - 8v_3 + 6v_4 &= 1 \end{aligned}$$

čije rešenje je

$$v_0 = 3, \qquad v_1 = 2.8125, \qquad v_2 = 2.7499, \qquad v_3 = 2.8123, \qquad v_4 = 2.9998.$$

9.11 *Dat je granični problem*

$$\begin{aligned} u'' + 2u' - xu &= x^2 \\ u'(0.6) = 0.7, \qquad & u(0.9) - 0.5u'(0.9) = 1. \end{aligned}$$

Diskretizovati granični problem na ravnomernoj mreži koraka $h = 0.05$ diferencijskom šemom tačnosti $O(h^2)$ i rešiti diskretni problem. Računati sa četiri decimale.

Rešenje: Aproksimacija jednačine diferencijskom šemom tačnosti $O(h^2)$ je

$$\frac{1}{h^2}(v_{i-1} - 2v_i + v_{i+1}) + \frac{1}{h}(v_{i+1} - v_{i-1}) - x_i v_i = x_i^2, \quad i = 1, \ldots, 5.$$

Za aproksimaciju prvog izvoda u graničnim uslovima koristimo razvoje

$$u'(x_0) = \frac{1}{2h}\big(-3u(x_0) + 4u(x_1) - u(x_2)\big) + O(h^2),$$

$$u'(x_6) = \frac{1}{2h}\big(u(x_4) - 4u(x_5) + 3u(x_6)\big) + O(h^2),$$

pomoću kojih dobijamo nesimetrične aproksimacije datih graničnih uslova tačnosti $O(h^2)$

$$-3v_0 + 4v_1 - v_2 = 0.07, \qquad v_6 - \frac{1}{4h}(v_4 - 4v_5 + 3v_6) = 1.$$

Sređivanjem ovih jednačina dobija se sistem

$$44v_0 - 43.9675v_1 = -1.4911$$

$$-19v_{i-1} + (40 + \frac{x_i}{20})v_i - 21v_{i+1} = -\frac{x_i^2}{20}, \quad i = 1, \ldots, 5,$$

$$-35.9575v_5 + 32.2v_6 = -3.8361$$

čije rešenje je

$$v_0 = 1.1568, \quad v_1 = 1.1899, \quad v_2 = 1.2227, \quad v_3 = 1.2555,$$
$$v_4 = 1.2888, \quad v_5 = 1.3229, \quad v_6 = 1.3581.$$

9.12 *Diskretizovati granični problem*

$$-(x\,u')' + u = 1 - xe^x, \qquad x \in (0.5, 0.9)$$

$$u'(0.5) - u(0.5) = -1, \qquad u(0.9) = 3.4596$$

diferencijskom šemom tačnosti $O(h^2)$ na ravnomernoj mreži koraka $h = 0.1$ i rešiti diskretni zadatak računajući na četiri decimale.

Rešenje: Prvo se navedeni zadatak svodi smenom

$$u(x) = w(x) + m\,x + n, \qquad m = 1.75686, \quad n = 1.87843,$$

na granični zadatak sa homogenim graničnim uslovima

$$-(x\,w')' + w = 1 + m - n - x(m + e^x),$$

$$0.5w'(0.5) - 0.5w(0.5) = 0, \qquad u(0.9) = 0.$$

Diferencijska šema koja sa tačnošću $O(h^2)$ aproksimira ovaj granični problem je

$$\left(\frac{x_0 + x_1}{h^2} + \frac{1}{h} + 1\right) v_0 - \frac{x_0 + x_1}{h^2} v_1 = f_0$$

$$-\frac{x_{i-1} + x_i}{2h^2} v_{i-1} + \left(\frac{x_{i-1} + 2x_i + x_{i+1}}{2h^2} + 1\right) v_i - \frac{x_i + x_{i+1}}{2h^2} v_{i+1} = f_i, \quad i = 1, 2, 3,$$

$$v_4 = 0$$

Rešenje diferencijskog problema je

$$v_0 = -0.10545, \quad v_1 = -0.10850, \quad v_2 = -0.09323, \quad v_3 = -0.05777, \quad v_5 = 0,$$

pa je približno rešenje polaznog problema

$$u(0.5) \approx 2.6514, \qquad u(0.6) \approx 2.8240, \qquad u(0.7) \approx 3.0150,$$
$$u(0.8) \approx 3.2262, \qquad u(0.9) = 3.4596.$$

9.13 *Metodom mreže sa korakom $h = 0.1$ rešiti granični problem*

$$-(x^2u'(x))' + 13u(x) = x^3$$
$$0.01u'(0.1) - u(0.1) = -0.0007, \qquad u'(0.5) + u(0.5) = 0.875.$$

Rešenje: Potrebno je transformisati polazni zadatak u oblik koji omogućava aproksimaciju diferencijskom šemom tačnosti $O(h^2)$. Prvo se množi desni granični uslov sa $p(0.5) = 0.5^2$ i dobija se

$$0.25u'(0.5) + 0.25u(0.5) = 0.2188.$$

Zatim se linearnom smenom $u(x) = w(x) + mx + n$ svodi dati problem na problem sa homogenim graničnim uslovima,

$$\begin{array}{rcl} 0.01\big(w'(0.1) + m\big) - \big(w(0.1) + 0.1m + n\big) = -0.0007 \\ 0.25\big(w'(0.5) + m\big) + 0.25\big(w(0.5) + 0.5m + n\big) = \quad 0.2188 \end{array} \quad \Longrightarrow \quad \begin{array}{r} m = \quad 0.6201 \\ n = -0.0551 \end{array}$$

Dakle, smenom $u(x) = w(x) + 0.6201x - 0.0551$ dati mešoviti problem se svodi na

$$-(x^2w'(x))' + 13w(x) = x^3 - 6.8211x + 0.7163,$$
$$0.01w'(0.1) - w(0.1) = 0, \qquad 0.25w'(0.5) + 0.25w(0.5) = 0$$

Odgovarajuća diferencijska šema je

$$\frac{2}{h}\Big(-\frac{x_0^2 + x_1^2}{2}\,\frac{v_1 - v_0}{h} + v_0\Big) + 13v_0 = \quad 0.0352$$

$$-\frac{x_i^2 + x_{i-1}^2}{2h^2}\,v_{i-1} + \Big(\frac{x_{i+1}^2 + 2x_i^2 + x_{i-1}^2}{2h^2} + 13\Big)v_i - \frac{x_{i+1}^2 + x_i^2}{2h^2}\,v_{i+1}$$
$$= x_i^3 - 6.8211x_i + 0.7136, \qquad i = 1, 2, 3,$$

$$\frac{2}{h}\Big(\frac{x_3^2 + x_4^2}{2}\,\frac{v_4 - v_3}{h} + 0.25v_4\Big) + 13v_4 = -2.5692$$

gde je $x_i = 0.1 + ih$, $i = 0, \ldots, 4$. Posle sređivanja dobija se trodijagonalni sistem jednačina

$$\begin{array}{rcl} 38v_0 - 5v_1 & = & 0.0352 \\ -2.5v_0 + 22v_1 - 6.5v_2 & = & -0.6399 \\ -6.5v_1 + 32v_2 - 12.5v_3 & = & -1.3030 \\ -12.5v_2 + 46v_3 - 20.5v_4 & = & -1.9481 \\ -41v_3 + 59v_4 & = & -2.5692 \end{array}$$

Kada se odrede rešenja v_i, $i = 0, \ldots, 4$, ovog sistema, rešenja polaznog problema se računaju iz smene $u(x_i) \approx v_i + 0.6201x_i - 0.0551$,

i	0	1	2	3	4
x_i	0.1	0.2	0.3	0.4	0.5
v_i	-0.0071	-0.0606	-0.1039	-0.1304	-0.1342
$u(x_i)$	-0.0002	0.0083	0.0270	0.0625	0.1208

Tačno rešenje je $u(x) = x^3$. Greška se povećava na krajevima intervala.

9.14 *a) Korišćenjem trapezne formule konstruisati diferencijsku šemu za rešavanje graničnog problema*

$$\mathbf{u}'(x) = \mathbf{f}(x, \mathbf{u}), \qquad g(\mathbf{u}(a)) = 0, \quad h(\mathbf{u}(b)) = 0,$$

gde su $\mathbf{u} = (u_1, u_2)^\top$ *i* $\mathbf{f} = (f_1, f_2)^\top$ *vektori.*
b) Izvedenom šemom rešiti granični problem

$$w'' = 6w/(x+1)^2, \qquad w(0) = 1, \quad w(1) = 8$$

za $h = 0.5$. *Računati sa tri decimale.*

Rešenje: a) Na intervalu $[a, b]$ definisaćemo mrežu

$$\omega_h = \{x_i \,|\, x_i = a + ih,\ i = 0, \ldots n,\ h = (b-a)/n\}.$$

Integraljenjem jednačine od a do x_i dobijamo da je

$$\mathbf{u}(x_i) - \mathbf{u}(a) = \int_a^{x_i} \mathbf{f}(x, \mathbf{u}(x))\, dx, \quad i = 1, \ldots n.$$

Aproksimacijom integrala trapeznom kvadraturnom formulom, uključujući i zadate granične uslove, dobijamo traženu diferencijsku šemu predstavljenu sledećim sistemom nelinearnih jednačina

$$g(v_{1,0}, v_{2,0}) = 0$$

$$v_{k,i} - v_{k,0} - \frac{h}{2}\big(f_k(a, v_{1,0}, v_{2,0}) + 2\sum_{j=1}^{i-1} f_k(x_j, v_{1,j}, v_{2,j}) + f_k(x_i, v_{1,i}, v_{2,i})\big) = 0,$$

$$k = 1, 2, \quad i = 1, \ldots n,$$

$$h(v_{1,n}, v_{2,n}) = 0,$$

pri čemu je suma jednaka nuli za $i = 1$. $v_{k,i}$, $k = 1, 2$, označava vrednost k-te komponente vektora približnog rešenja $\mathbf{v} = (v_1, v_2)^\top$ u čvoru x_i mreže ω_h. Sistem

dimenzije $2n+2$ ima skoro trougaonu matricu

$$\begin{pmatrix} * & * & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ * & * & * & * & 0 & 0 & \dots & 0 & 0 \\ * & * & * & * & * & * & \dots & 0 & 0 \\ \vdots & & & & & & \ddots & & \vdots \\ * & * & * & * & * & * & \dots & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & * & * \end{pmatrix}$$

te se efikasno može rešiti Newtonovom metodom.
b) Dati graniči problem definisan jednačinom drugog reda se smenom $u_1 = w$ i $u_2 = w'$ svodi na problem dat pod tačkom a), pri čemu je

$$f_1(x,u_1,u_2) \equiv u_2, \qquad f_1(x,u_1,u_2) \equiv 6u_1/(x+1)^2,$$
$$g(u_1(0),u_2(0)) \equiv u_1(0)-1, \qquad h(u_1(1),u_2(1)) \equiv u_1(1)-8.$$

U ovom slučaju diferencijska šema je predstavljena sistemom linearnim jednačina

$$\begin{aligned} v_{1,0} - 1 &= 0 \\ v_{1,1} - v_{1,0} - 0.25(v_{2,0} + v_{2,1}) &= 0 \\ v_{2,1} - v_{2,0} - 0.25(6v_{1,0} + \frac{8}{3}v_{1,1}) &= 0 \\ v_{1,2} - v_{1,0} - 0.25(v_{2,0} + 2v_{2,1} + v_{2,2}) &= 0 \\ v_{2,2} - v_{2,0} - 0.25(6v_{1,0} + \frac{16}{3}v_{1,1} + 3v_{1,2}/2) &= 0 \\ v_{1,2} - 8 &= 0 \end{aligned}$$

čije rešenje je

$$v_{1,0} = 1, \qquad v_{1,1} = 3.375, \qquad v_{1,2} = 8,$$
$$v_{2,0} = 2.875, \qquad v_{2,1} = 6.625, \qquad v_{2,2} = 11.875.$$

Prva komponenta vektora $\mathbf{v}$ sadrži približne vrednosti rešenja datog zadatka, pa je

$$u(0) = 1, \qquad u(0.5) \approx 3.375, \qquad u(1) = 8.$$

9.15 *Odrediti sopstvene vrednosti λ graničnog problema*

$$u''(x) + \lambda u(x) = 0, \qquad u(0) = u(1) = 0.$$

Koristiti metodu mreže sa $n = 2,3,4$ podintervala.

Rešenje: Diskretizacijom datog graničnog problema na mreži $\omega = \{x_i \mid x_i = ih,\ i = 0,\dots,n,\ h = \frac{1}{n}\}$ dobija se sistem jednačina

$$v_0 = v_n = 0, \qquad \frac{1}{h^2}\left(-v_{i+1} + 2v_i - v_{i-1}\right) = \lambda_i v_i, \quad i = 1,\dots,n-1.$$

Očigledno je da je problem sveden na nalaženje sopstvenih vrednosti matrice A dimenzije $(n-1)\times(n-1)$,

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 2 \end{pmatrix}$$

$\underline{n=2}$

$$A = \frac{1}{(\frac{1}{2})^2}(2), \qquad \det(A-\lambda I) = 8-\lambda, \qquad \lambda_1 = 8.$$

$\underline{n=3}$

$$A = \frac{1}{(\frac{1}{3})^2}\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \qquad \det(A-\lambda I) = (9-\lambda)(27-\lambda),$$

$$\lambda_1 = 9, \quad \lambda_2 = 27.$$

$\underline{n=4}$

$$A = \frac{1}{(\frac{1}{4})^2}\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \qquad \det(A-\lambda I) = (32-\lambda)(1024-64\lambda+\lambda^2-512),$$

$$\lambda_1 = 9.37, \quad \lambda_2 = 32, \quad \lambda_3 = 54.63.$$

FORTRAN Implementacija metode mreže sastoji se od postavljanja trodijagonalnog sistema jednačina po rešenjima graničnog problema u čvorovima mreže i od rešavanja toga sistema. Treba uočiti kako se nastojalo da broj računskih operacija pri izračunavanju koeficijenata trodijagonalnog sistema bude minimizovan. Za rešavanje sistema, koristi se *Fortran*-ska implementacija procedure `tridiag()` koja se može naći u biblioteci `libnumerics`, a takođe i u dodatku A ove zbirke. Procedura `mesh()` ima oblik:

```
module mesh_module
  use matheval_module
  use tridiag_module
  implicit none
  public mesh

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string

contains
  ! Procedura mesh() odredjuje resenja datog granicnog problema metodom
  ! mreze semom tacnosti O(h^2). Parametri procedure su:
  !   n - broj podintervala mreze
  !   lo, hi - granice intervala na kome se resava problem
  !   q, f - funkcije koje odredjuju problem
```

```
!   alpha0, beta0, alpha1, beta1 - koeficijenti granicnih uslova
!   v - polje u koje ce biti smestena resenja
! Broj podintervala mreze ne sme biti manji od 2. Granice intervala
! moraju biti tako postavljene da je druga granica veca od
! prve. Funkcije koje figurisu u problemu moraju biti sintaksno
! ispravno zadate. Pretpostavlja se da je polje u koje ce biti
! smesteni rezultati alocirano izvan procedure.
subroutine mesh (n, lo, hi, q, f, alpha0, beta0, alpha1, beta1, v)
  integer, intent(in) :: n
  real(kind=8), intent(in) :: lo, hi
  type(string), intent(in) :: q, f
  real(kind=8), intent(in) :: alpha0, beta0, alpha1, beta1
  real(kind=8), dimension(0:n), intent(out) :: v
  integer(kind=8) :: q_evaluator, f_evaluator ! Evaluatori za funkcije
      ! koje se javljaju u jednacini.
  real(kind=8) :: h ! Korak mreze.
  real(kind=8) :: a1 ! Vrednost -1/h**2.
  real(kind=8), dimension(0:n) :: a, b, c, d ! Koeficijenti
      ! trodijagonalnog sistema jednacina koji odredjuje vrednosti
      ! resenja problema.
  integer :: i ! Brojac u petljama.

  ! Proveravaju se argumenti procedure i zaustavlja se program ako
  ! neki od njih nije validan.
  if (n<2) stop
  if (hi<=lo) stop
  q_evaluator=evaluator_create(q%chars)
  if (q_evaluator==0) stop
  f_evaluator=evaluator_create(f%chars)
  if (f_evaluator==0) stop

  ! Odredjuju se korak mreze i promenljiva a1.
  h=(hi-lo)/n
  a1=-1/h**2

  ! Odredjuju se koeficijenti unutrasnjih jednacina trodijagonalnog
  ! sistema.
  a(1:n-1)=(/(a1, i=1, n-1)/)
  b(1:n-1)=a(1:n-1)
  c(1:n-1)=(/(evaluator_evaluate_x(q_evaluator,lo+i*h)-2*a1, i=1, n-1)/)
  d(1:n-1)=(/(evaluator_evaluate_x(f_evaluator,lo+i*h), i=1, n-1)/)

  ! Unistavaju se korisceni evaluatori.
  call evaluator_destroy(q_evaluator)
  call evaluator_destroy(f_evaluator)

  ! Odredjuju se koeficijenti prve jednacine trodijagonalnog sistema.
  a(0)=0
  b(0)=alpha0/2*c(1)+2*alpha0*a1
  c(0)=-alpha0*a1-beta0/h
  d(0)=alpha0/2*d(1)

  ! Odredjuju se koeficijenti poslednje jednacine trodijagonalnog sistema.
  a(n)=alpha1/2*c(n-1)+2*alpha1*a1
  b(n)=0
  c(n)=-alpha1*a1+beta1/h
  d(n)=alpha1/2*d(n-1)
```

```
    ! Resava se trodijagonalni sistem jednacina cija su resenja
    ! istovremeno i resenja datog granicnog problema.
    call tridiag (n+1, a, b, c, d, v)
  end subroutine mesh
end module mesh_module
```

9.3 Varijacione metode

Približno rešenje se određuje u obliku

$$v(x) = \phi_0(x) + \sum_{k=1}^{n} c_k \phi_k(x)$$

Funkcije $\phi_k(x)$, $k = 1, \ldots, n$ su poznate linearno nezavisne funkcije koje zadovoljavaju homogen granični uslov, a funkcija $\phi_0(x)$ se bira tako da zadovoljava zadate granične uslove. Konstante c_k, $k = 1, \ldots, n$, se određuju minimizacijom funkcije greške koja je za diferencijalnu jednačinu zapisanu u obliku

$$L\,u = f$$

gde je L diferencijalni operator, data izrazom

$$R(x; c_1, \ldots, c_n) = L\,v - f$$

Metoda kolokacije. Konstante c_k se određuju tako da funkcija greške u izabranim tačkama x_k, $k = 1, \ldots, n$, intervala $[a, b]$ bude jednaka nuli

$$R(x_k; c_1, \ldots, c_n) = 0, \qquad k = 1, \ldots, n.$$

Ritz-ova metoda. Koristi se ako je L samokonjugovani operator. Konstante c_k se određuju tako da je funkcija greške ortogonalna na funkcije ϕ_k,

$$(R, \phi_k) = \int_a^b R(x; c_1, \ldots, c_n)\phi_k(x)\,dx = 0, \qquad k = 1, \ldots, n.$$

Galerkin-ova metoda. Konstante c_k se određuju tako da je funkcija greške ortogonalna na izabrani sistem linearno nezavisnih funkcija ψ_k, $k = 1, \ldots, n$,

$$(R, \psi_k) = \int_a^b R(x; c_1, \ldots, c_n)\psi_k(x)\,dx = 0, \qquad k = 1, \ldots, n.$$

Može se uzeti i da je $\psi_k(x) \equiv \phi_k(x)$, $k = 1, \ldots, n$

Metoda najmanjih kvadrata. Minimizira se srednjekvadratna norma funkcije greške

$$\|R\|^2 = \int_a^b R^2(x; c_1, \ldots, c_n)\, dx = \min$$

Ovaj integral je funkcija koeficijenata $I(c_1, \ldots, c_n)$. Tačku minimuma određujemo iz uslova

$$\frac{\partial I}{\partial c_k} = 0, \qquad k = 1, \ldots, n,$$

tj.

$$\int_a^b R \frac{\partial R}{\partial c_k}\, dx = 0, \qquad k = 1, \ldots, n.$$

9.16 *Metodom kolokacije rešiti granični problem*

$$u'' + u = \sin x, \qquad u(0) = 1, \quad u(\pi/2) = 1,$$

ako su tačke kolokacije $1/2$ *i* 1*, a bazisne funkcije*

$$\phi_0(x) = \sin x + \cos x, \quad \phi_1(x) = x\, \cos x, \quad \phi_2(x) = x^2 \cos x.$$

Računati na četiri decimale.

Rešenje: Približno rešenje tražimo u obliku

$$v(x) = \sin x + \cos x + c_1 x \cos x + c_2 x^2 \cos x.$$

Zamenom u jednačini ono daje funkciju greške

$$R(x; c_1, c_2) = -\sin x - 2c_1 \sin x + 2c_2(\cos x - 2x \sin x).$$

Uslovi kolokacije $R(1/2; c_1, c_2) = 0$ i $R(1; c_1, c_2) = 0$ daju sistem

$$\begin{aligned} -0.9589\, c_1 + 0.7963\, c_2 &= 0.4794 \\ -1.6829\, c_1 - 2.2853\, c_2 &= 0.8415 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} c_1 &= -0.5000 \\ c_2 &= 0.0000 \end{aligned}$$

Stoga je traženo približno rešenje (pokazuje se da je to i tačno rešenje)

$$v(x) = \sin x + (1 - 0.5x) \cos x = u(x).$$

9.17 *Metodom kolokacije rešiti granični problem*

$$u'' + u = \sin(2x), \qquad u(0) = 1, \quad u(\pi/2) = 1,$$

ako su tačke kolokacije $1/2$*,* 1 *i* $3/2$*, a bazisne funkcije* $\phi_0(x) = \sin x + \cos x$ *i* $\phi_k(x) = \sin(2kx)$*,* $k = 1, 2, 3$*. Računati na četiri decimale.*

Rešenje: Približno rešenje tražimo u obliku

$$v(x) = \sin x + \cos x + \sum_{k=1}^{3} c_k \sin(2kx).$$

Zamenom u jednačini dobijamo funkciju greške

$$R(x; c_1, c_2, c_3) = -(1 + 3c_1)\sin(2x) - 15c_2 \sin(4x) - 35c_3 \sin(6x).$$

Uslovi kolokacije $R(1/2; c_1, c_2, c_3) = 0$, $R(1; c_1, c_2, c_3) = 0$ i $R(3/2; c_1, c_2, c_3) = 0$ daju sistem

$$\begin{array}{rcl} 2.5245c_1 + 13.6395c_2 + 4.9385c_3 = -0.8415 & & c_1 = -0.3333 \\ -2.7279c_1 + 11.3520c_2 + 9.7790c_3 = \ \ 0.9093 & \Longrightarrow & c_2 = \ \ 0 \\ 0.4233c_1 - 4.1910c_2 + 14.4235c_3 = -0.1411 & & c_3 = \ \ 0 \end{array}$$

Stoga je traženo približno (i tačno) rešenje

$$v(x) = \sin x + \cos x - 0.3333 \sin(2x).$$

9.18 *Ritz-ovom metodom naći približno rešenje graničnog problema*

$$(xu')' + u = x, \qquad u(0) = 0, \quad u(1) = 1$$

u obliku $v(x) = x + x(1-x)(c_1 + c_2 x)$.

Rešenje: Dati oblik približnog rešenja zadovoljava granične uslove za svaki izbor parametara c_1 i c_2, a zamenom u jednačini daje grešku

$$R(x; c_1, c_2) = (xv')' + v - x = (1 - 3x - x^2)c_1 + (4x - 8x^2 - x^3)c_2 + 1.$$

Ritz-ovom metodom parametri c_1 i c_2 se određuju tako da greška bude ortogonalna na bazisne funkcije, tj. tako da je

$$\int_0^1 R(x; c_1, c_2)\phi_i(x)\,dx = 0, \quad i = 1, 2$$

gde je $\phi_1(x) = x(1-x)$ i $\phi_2(x) = x^2(1-x)$ jer je $v(x) = c_1\phi_1(x) + c_2\phi_2(x) + x$. Izračunavanjem navedenih integrala dobija se sistem linearnih jednačina

$$\frac{2}{15}c_1 + \frac{1}{10}c_2 = \frac{1}{6}, \quad \frac{1}{10}c_1 + \frac{19}{210}c_2 = \frac{1}{12}, \qquad \Longrightarrow \qquad c_1 = \frac{85}{26}, \quad c_2 = -\frac{35}{13}.$$

Stoga je tražena aproksimacija

$$v(x) = x + \frac{5}{26}x(1-x)(17 - 14x).$$

9.19 *Korišćenjem Ritz-ove metode sa bazisnim funkcijama*

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/h, & x \in (x_{i-1}, x_i) \\ (x_{i+1} - x)/h, & x \in (x_i, x_{i+1}) \\ 0, & x \notin (x_{i-1}, x_{i+1}) \end{cases} \qquad i = 1, \ldots, n-1,$$

konstruisati diferencijsku šemu za granični problem

$$\big(p(x)\, u'(x)\big)' = 1, \qquad u(0) = u(1) = 0,$$

gde je

$$p(x) = \begin{cases} 3/2, & 0 \le x \le \pi/4, \\ 2, & \pi/4 < x \le 1 \end{cases}$$

Rešenje: Aproksimaciju rešenja $u(x)$ tražimo u obliku

$$v(x) = \sum_{i=1}^{n-1} v_i \phi_i(x),$$

gde je $v_i = v(x_i)$ jer je $\phi_i(x_j) = \delta_{ij}$. Koeficijente v_i određujemo kao rešenja sistema jednačina

$$\int_0^1 \left(1 + \frac{d}{dx}\big(p(x) \sum_{i=1}^{n-1} v_i \phi_i'(x)\big)\right) \phi_j(x)\, dx = 0, \qquad j = 1, \ldots n-1,$$

koji je posle parcijalne integracije

$$\int_0^1 p(x) \big(\sum_{i=1}^{n-1} v_i \phi_i'(x)\big) \phi_j'(x)\, dx = \int_0^1 \phi_j(x)\, dx \qquad j = 1, \ldots n-1.$$

Ako se mreža izabere tako da je $x_k = \pi/4$ čvor mreže, prethodni sistem se može napisati u obliku

$$\frac{3}{2} \sum_{i=1}^{n-1} v_i \int_0^{x_k} \phi_i'(x)\phi_j'(x)\, dx + 2 \sum_{i=1}^{n-1} v_i \int_{x_k}^1 \phi_i'(x)\phi_j'(x)\, dx = \int_0^1 \phi_j(x)\, dx$$
$$j = 1, \ldots n-1.$$

S obzirom da je funkcija $\phi_j(x)$ funkcija sa kompaktnim nosačem, tj. da je ta funkcija jednaka nuli na celom intervalu $(0, 1)$ osim na podintervalu (x_{j-1}, x_{j+1}), prethodni sistem se svodi na sistem

$$\frac{3}{2}\Bigg(v_{j-1} \int_{x_{j-1}}^{x_j} \phi_{j-1}'(x)\phi_j'(x)\, dx + v_j \int_{x_{j-1}}^{x_{j+1}} \big(\phi_j'(x)\big)^2 dx$$
$$+ v_{j+1} \int_{x_j}^{x_{j+1}} \phi_{j+1}'(x)\phi_j'(x)\, dx \Bigg) = \int_{x_{j-1}}^{x_{j+1}} \phi_j(x)\, dx \quad j = 1, \ldots k-1,$$

$$\frac{3}{2}\left(v_{k-1}\int_{x_{k-1}}^{x_k}\phi'_{k-1}(x)\phi'_k(x)\,dx + v_k\int_{x_{k-1}}^{x_k}(\phi'_k(x))^2dx\right) + 2\left(v_k\int_{x_k}^{x_{k+1}}(\phi'_k(x))^2dx + v_{k+1}\int_{x_k}^{x_{k+1}}\phi'_{k+1}(x)\phi'_k(x)\,dx\right) = \int_{x_{k-1}}^{x_{k+1}}\phi_k(x)\,dx$$

$$2\left(v_{j-1}\int_{x_{j-1}}^{x_j}\phi'_{j-1}(x)\phi'_j(x)\,dx + v_j\int_{x_{j-1}}^{x_{j+1}}(\phi'_j(x))^2dx + v_{j+1}\int_{x_j}^{x_{j+1}}\phi'_{j+1}(x)\phi'_j(x)\,dx\right) = \int_{x_{j-1}}^{x_{j+1}}\phi_j(x)\,dx \quad j = k+1,\dots n-1,$$

Izračunavanjem integrala dobijamo diferencijsku šemu

$$\begin{aligned} v_0 &= 0 \\ v_{j-1} - 2v_j + v_{j+1} &= -2h^2/3, \qquad 1 \le j < k \\ 3v_{k-1} - 7v_k + 4v_{k+1} &= -2h^2, \\ v_{j-1} - 2v_j + v_{j+1} &= -h^2/2, \qquad k < j \le n-1 \\ v_n &= 0 \end{aligned}$$

9.20 *a) Dat je granični problem*

$$-\frac{d^2u}{dx^2} + a\frac{du}{dx} + bu = 1, \quad x \in [0,1], \qquad u(0) = u(1) = 0,$$

gde je $a, b = const$ i $b \ge 0$. Dokazati da se rešavanjem ovog problema Galerkinovom metodom na mreži $\omega = \{x_i \,|\, x_i = ih,\ i = 0,\dots,n,\ h = 1/n\}$ sa bazisnim funkcijama

$$\phi_i(x) = \begin{cases} 1 - |x_i - x|/h, & x \in (x_{i-1}, x_{i+1}) \\ 0, & x \notin (x_{i-1}, x_{i+1}) \end{cases} \qquad i = 1,\dots,n-1,$$

dobija diferencijska šema

$$\begin{gathered} v_0 = 0 \\ -\frac{1}{h^2}(v_{j-1}-2v_j+v_{j+1}) + \frac{a}{2h}(v_{j+1}-v_{j-1}) + \frac{b}{6}(v_{j-1}+4v_j+v_{j+1}) = 1, \quad j = 1,\dots,n-1 \\ v_n = 0. \end{gathered}$$

b) Rešiti diferencijski zadatak ako je $a = 2$, $b = 6$ i $h = 0.1$. Računati sa pet decimala.

Rešenje: a) Zadatak se rešava na isti način kao prethodni, s obzirom da se aproksimacija određuje nad istim skupom bazisnih funkcija.

b) Za $a = 2$, $b = 6$ i $h = 0.1$ diferencijska šema je

$$\begin{aligned} v_0 &= 0 \\ -1.09v_{j-1} + 2.04v_j - 0.89v_{j+1} &= 0.01, \qquad j = 1, \ldots, 9, \\ v_{10} &= 0. \end{aligned}$$

Rešavananjem sistema linearnih jednačina sa trodijagonalnom matricom Gaussovom metodom eliminacije dobijeni su sledeći rezultati

$$v_0 = 0, \quad v_1 = 0.0233, \quad v_2 = 0.0421, \quad v_3 = 0.0568, \quad v_4 = 0.0673, \quad v_5 = 0.0735,$$
$$v_6 = 0.0748, \quad v_7 = 0.0703, \quad v_8 = 0.0581, \quad v_9 = 0.0359, \quad v_{10} = 0.$$

Tražena aproksimacija je deo po deo linearna funkcija

$$v(x) = \sum_{i=1}^{9} v_i \phi_i(x).$$

9.21 *Galerkin-ovom metodom rešiti granični problem:*

$$u''(x) + u(x) + x = 0, \quad u(0) = u(1) = 0.$$

Rešenje: Za bazisne funkcije se mogu uzeti $\phi_1(x) = x(1-x)$ i $\phi_2(x) = x^2(1-x)$, pa je približno rešenje oblika

$$v(x) = c_1 x(1-x) + c_2 x^2(1-x)$$

Funkcija greške

$$R(x; c_1, c_2) = v'' + v + x = -2c_1 + c_2(2 - 6x) + x(1-x)(c_1 + c_2 x) + x$$

je ortogonalna na bazisne funkcije ako je

$$\begin{aligned} \frac{3}{10}\, c_1 + \frac{3}{20}\, c_2 &= \frac{1}{12} \\ \frac{3}{20}\, c_1 + \frac{13}{105}\, c_2 &= \frac{1}{20} \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} c_1 &= \frac{71}{369} \\ c_2 &= \frac{7}{41} \end{aligned}$$

Približno rešenje problema je

$$v(x) = x(1-x)\left(\frac{71}{369} + \frac{7}{41}\, x\right)$$

9.22 *Metodom Galerkin-a naći približno rešenje graničnog problema*

$$u''(x) + u(x) = \cos x, \qquad u(0) = 1, \quad u(\pi/2) = 0.$$

Za bazisne funkcije uzeti $\phi_0(x) = \cos x$, $\phi_k(x) = \sin(2kx)$, $k = 1, \ldots, 4$.

Rešenje: Približno rešenje tražimo u obliku

$$v(x) = \cos x + \sum_{k=1}^{4} c_k \sin(2kx).$$

Ono očigledno zadovoljava granične uslove, a zamenom u jednačinu daje funkciju greške

$$R(x; c_1, c_2, c_3, c_4) = \sum_{k=1}^{4} (1 - 4k^2) c_k \sin(2kx) - \cos x.$$

Galerkinovom metodom konstante c_k se određuju iz uslova

$$\int_0^{\pi/2} R(x; c_1, c_2, c_3, c_4) \sin(2jx)\, dx = 0, \qquad j = 1, 2, 3, 4,$$

tj.

$$\int_0^{\pi/2} \sum_{k=1}^{4} (1 - 4k^2) c_k \sin(2kx) \sin(2jx)\, dx = \int_0^{\pi/2} \cos x \sin(2jx)\, dx, \quad j = 1, 2, 3, 4.$$

Kako je

$$\int_0^{\pi/2} \sin(2kx) \sin(2jx)\, dx = \begin{cases} 0, & k \neq j \\ \pi/4, & k = j \end{cases}$$

$$\int_0^{\pi/2} \cos x \sin(2jx)\, dx = \frac{2j}{4j^2 - 1},$$

to je

$$c_j = \frac{8}{\pi} \frac{-j}{(4j^2 - 1)^2}.$$

Dakle, približno rešenje graničnog problema je

$$v(x) = \cos x - \frac{8}{\pi} \left(\frac{1}{9} \sin 2x + \frac{2}{225} \sin 4x + \frac{3}{1225} \sin 6x + \frac{4}{3969} \sin 8x \right).$$

MATLAB

Kroz sledeći zadatak ilustrovaćemo korišćenje modula za simbolička izračunavanja *(symbolic math toolbox-a).*

9.23 *Korišćenjem* MATLAB-*a Galerkin-ovom metodom, metodom kolokacije sa tačkama kolokacije* 0 *i* 1 *i metodom najmanjih kvadrata rešiti granični problem*

$$u''(x) + xu'(x) + u(x) = 2x, \quad u(0) = 1, \quad u(1) = 0$$

Rešenje: Prilikom implementacije pomenutih varijacionih metoda, javlja se potreba za diferenciranjem i integracijom analitički zadatih funkcija. Iako je jedan od uobičajenih načina da se ovo izvrši korišćenje numeričkih algoritama, u ovom primeru smo se odlučili da iskoristimo mogućnosti MATLAB-ovog modula za simbolička izračunavanja i navedene probleme rešimo analitički. Prednost ovog pristupa je izbegavanje nastajanja greške do koje neizbežno dolazi prilikom numeričkog diferenciranja i integracije. Mana je, naravno, smanjena efikasnost rešenja kao i ograničen domen primene. Naime, poznato je da veliki broj elementarnih funkcija nemaju elementarne primitivne funkcije i nije moguće analitički pronaći njihove integrale. Kod koji sledi bi trebalo da posluži kao inspiracija čitaocima i da ih uputi na korišćenje MATLAB-a kao alata koji može da uradi veliki deo rutinskih izračunavanja prilikom analitičkog rešavanja matematičkih problema.

Približno rešenje zadatka tražimo u obliku polinoma

$$v(x) = \phi_0(x) + c_1\phi_1(x) + c_2\phi_2(x),$$

gde je bazisna funkcija $\phi_0(x) = 1 - x$ izabrana tako da zadovoljava date granične uslove, a funkcije $\phi_1(x) = x(1 - x)$ i $\phi_2(x) = x^2(1 - x)$ zadovoljavaju homogene granične uslove.

```
syms c1 c2 c3 x

% Bazisne funkcije
phi0 = 1-x; phi1 = x*(1-x); phi2 = x^2*(1-x);

% Priblizno resenje
v = phi0 + c1*phi1 + c2*phi2;

% Greska
R = diff(v, 2) + x*diff(v, 1) + v - 2*x;

% Galerkinova metoda
[c1g, c2g] = solve(int(R*phi1, 0, 1), int(R*phi2, 0, 1))

% Kolokacija
[c1k, c2k] = solve(subs(R, x, 0), subs(R, x, 1))

% Najmanji kvadrati
[c1ls, c2ls] = solve(int(R*diff(R, c1), 0, 1), int(R*diff(R, c2), 0, 1))

% Zamenjujemo simbolicke promenljive c1 i c2 pronadjenim
% resenjima sistema
vg = subs(v, {c1, c2}, {c1s, c2s});
vk = subs(v, {c1, c2}, {c1k, c2k});
vls = subs(v, {c1, c2}, {c1ls, c2ls});

% Uproscavamo i ispisujemo rezultate
pretty(simplify(vg))
pretty(simplify(vk))
pretty(simplify(vls))
```

Granični problemi: `sol = bvp4c(@system, @boundary_conditions, initial_guess)`
Sistem j-na prvog reda se opisuje posebnom funkcijom:
`function dy = system(x, y)`
`dy = [`$f_1(t,u)$`;...;`$f_n(t,u)$`];`
Granični uslovi oblika $g(y(a), y(b)) = 0$ se opisuju funkcijom:
`function res = boundary_conditions(ya, yb)`
`res =` g`(ya, yb);`
Pretpostavljeno rešenje se tabelira korišćenjem:
`initial_guess = bvpinit(x, guess_function)`

MATLAB raspolaže funkcijom `bvp4c` za rešavanje graničnih problema koja se zasniva na određenoj varijanti metode kolokacije. Sistemi diferencijalnih jednačina koji mogu da se rešavaju korišćenjem ove funkcije imaju oblik

$$\mathbf{y}'(x) = \mathbf{f}(\mathbf{x}, \mathbf{y}(\mathbf{x})), \quad \mathbf{g}(\mathbf{y}(\mathbf{a}), \mathbf{y}(\mathbf{b})) = \mathbf{0}$$

Rešenje se pronalazi na intervalu $[a, b]$, a granični uslovi su zadati funkcijom g. Pošto granični problemi ovog tipa mogu da imaju više od jednog rešenja, funkciji `bvp4c` je, uz jednačine, proslediti i pretpostavljenu funkciju rešenja. Imajući ovo u vidu, opšti oblik poziva funkcije je:

```
bvp4c(system, boundary_conditions, initial_guess)
```

Funkcija `dy = system(x, y)` je funkcija koja predstavlja funkciju $\mathbf{y}'(x) = \mathbf{f}(\mathbf{x}, \mathbf{y}(\mathbf{x}))$ i za dati broj `x` i vektor `y` izračunava vektor kolonu prvih izvoda `dy`.

Funkcija `res = boundary_conditions(ya, yb)` izračunava vrednost funkcije $g(\mathbf{y}(\mathbf{a}), \mathbf{y}(\mathbf{b}))$ za date vektore `ya` i `yb`.

`initial_guess` je struktura od koje se zahteva da ima polja `x` i `y` koja predstavljaju vrednosti pretpostavljenog rešenja tabelirane na intervalu $[a, b]$. Za kreiranje ove strukture, moguće je koristiti pomoćnu funkciju `initial_guess = bvpinit(x, f)`, koja vrši tabeliranje funkcije `f` na mreži zadatoj vektorom `x`.

Funkcija `bvp4c` vraća strukturu `sol` čija polja `x` i `y` sadrže tabelirane vrednosti rešenja jednačine.

9.24 *Poprečni presek kapljice vode postavljene na ravnu površinu zadovoljava diferencijalnu jednačinu*

$$h''(x) + (1 - h(x)) \left(\sqrt{1 + h'(x)^2}\right)^3 = 0, \quad h(-1) = 0, \quad h(1) = 0,$$

pri čemu $h(x)$ označava visinu kapljice u tački x. Koristeći MATLAB *rešiti datu diferencijalnu jednačinu i grafički prikazati rešenje.*

Rešenje: Prevedimo polaznu jednačinu i sistem jednačina uvođenjem smena $y_1(x) = h(x)$, $y_2(x) = h'(x)$.

$$\begin{aligned} y_1'(x) &= y_2(x) \\ y_2'(x) &= (y_1(x) - 1) \left(\sqrt{1 + y_2(x)^2}\right)^3 \end{aligned}$$

Ovaj sistem predstavljamo na standardni način, funkcijom `drop_system`. Granični uslovi postaju $y_1(-1) = 0$, $y_1(1) = 0$ i predstavljeni su funkcijom `drop_bc`. Za pretpostavljeno rešenje, uzmimo funkcije $y_1(x) = \sqrt{1-x^2}$, $y_2(x) = \frac{-x}{0.1+\sqrt{1-x^2}}$. Ono je predstavljeno funkcijom `drop_init`, a zatim je tabelirano na ekvidistantnoj mreži intervala $[-1, 1]$ koristeći funkciju `bvpinit`.

```
function drop
% Resavamo jednacinu
sol = bvp4c(@drop_system, @drop_bc, bvpinit(linspace(-1, 1, 20), @drop_init));
%Iscrtavamo resenje
fill(sol.x, sol.y(1, :), 'r'), axis equal

% Funkcija kojom se zadaje sistem jednacina
function dy = drop_system(x, y)
    dy = [y(2); ...
          (y(1)-1)*sqrt(1+y(2)^2)^3];

% Funkcija kojom se zadaju granicni uslovi
function res = drop_bc(ya, yb)
    res = [ya(1);...
           yb(1)];

% Funkcija kojom se zadaje pretpostavljeno resenje sistema
function y = drop_init(x)
    y = [sqrt(1-x.^2); ...
         -x./(0.1 + sqrt(1-x.^2))];
```

9.25 *Uže koje se okreće zadovoljava diferencijalnu jednačinu*

$$y''(x) + \mu y(x) = 0$$

sa graničnim uslovima

$$y(0) = 0, \qquad y'(0) = 1, \qquad y(1) + y'(1) = 0$$

Korišćenjem MATLAB-a, *pronaći vrednost parametra μ za koje jednačina ima rešenje.*

Rešenje: Uvođenjem smena $y_1(x) \equiv y(x)$, $y_2(x) = y'(x)$, sistem se svodi na

$$\begin{aligned} y_1'(x) &= y_2(x) \\ y_2'(x) &= -\mu y_1(x) \end{aligned}$$

i predstavljen je funkcijom `skip_system`, dok se granični uslovi prevode u

$$y_1(0) = 0, \qquad y_2(0) = 0, \qquad y_1(1) + y_2(1) = 0$$

i predstavljeni su funkcijom `skip_bc`. Pretpostavljeno rešenje sistema je $y = sin(x)$ i ono se zadaje funkcijom `skip_init`.

```
function skip_rope
% Resavamo sistem diferencijalnih jednacina
sol = bvp4c(@skip_system, @skip_bc, bvpinit(linspace(0,1,20), @skip_init, 5))
% Iscrtavamo resenje
plot(sol.x, sol.y(1,:),'-o');
% Ispisujemo pronadjenu sopstvenu vrednost
sol.parameters

% Funkcija kojom se zadaje sistem jednacina
function dy = skip_system(x, y, mi)
    dy = [y(2); -mi * y(1)];

% Funkcija kojom se zadaju granicni uslovi
function res = skip_bc(ya, yb, mi)
    res = [ya(1); ya(2)-1; yb(1)+yb(2)];

% Funkcija kojom se zadaje pretpostavljeno resenje sistema
function y = skip_init(x)
    y = [sin(x); cos(x)];
```

Primetimo kako se nepoznati parametar `mi` prosleđuje funkcijama `bvpinit`, `skip_system` i `skip_bc` i na taj način se funkciji `bvp4c` stavlja do znanja da se rešava problem sopstvenih vrednosti. Po završetku rada ove funkcije, polje `parameters` strukture `sol` sadrži izračunatu sopstvenu vrednost.

10

Integralne jednačine

Fredholm-ova integralna jednačina druge vrste

$$u(x) = f(x) + \lambda \int_a^b K(x,t)u(t)\,dt, \qquad x \in [a,b]$$

Volterra-ova integralna jednačina druge vrste

$$u(x) = f(x) + \lambda \int_a^x K(x,t)u(t)\,dt, \qquad x \in [a,b]$$

10.1 *Pokazati da je integralna jednačina*

$$u(x) = -\frac{x-a}{b-a}\int_x^b (b-t)f(t,u(t))\,dt - \frac{b-x}{b-a}\int_a^x (t-a)f(t,u(t))\,dt + \frac{\beta-\alpha}{b-a}x + \frac{b\alpha - a\beta}{b-a}$$

ekvivalentna graničnom problemu

$$u''(x) = f(x,u(x)), \qquad u(a) = \alpha, \quad u(b) = \beta.$$

Rešenje: Diferenciranjem integralne jednačine po x dobijamo da je

$$(b-a)u'(x) = -\int_x^b (b-t)f(t,u(t))\,dt + \int_a^x (t-a)f(t,u(t))\,dt + (\beta-\alpha).$$

Ponovnim diferenciranjem po x dobijamo diferencijalnu jednačinu

$$u''(x) = f(x,u(x)).$$

Stavljajući u integralnoj jednačini $x = a$ dobijamo granični uslov $u(a) = \alpha$ i za $x = b$ granični uslov $u(b) = \beta$.

10.1 Metoda uzastopnih aproksimacija

Metoda uzastopnih aproksimacija za rešavanje Fredholm-ove integralne jednačine određena je rekurentnom formulom

$$v_0(x) = f(x),$$
$$v_{n+1}(x) = f(x) + \lambda \int_a^b K(x,t)v_n(t)\,dt, \quad n = 0, 1, \ldots.$$

Greška se približno ocenjuje veličinom

$$\max_{[a,b]} |u(x) - v_n(x)| \approx \max_{[a,b]} |v_n(x) - v_{n-1}(x)|$$

Metoda konvergira ako je

$$|\lambda|(b-a)M < 1,$$

gde je $M = \max_{a\le x,t,\le b} |K(x,t)|$.
Ako se rešava Volterra-ova jednačina, granicu b treba zameniti sa x. U ovom slucaju metoda konvergira za svako λ.

10.2 *Metodom uzastopnih aproksimacija odrediti rešenje integralne jednačine*

$$u(x) = \lambda \int_0^1 xt^2 u(t)\,dt + 1$$

Odrediti vrednost parametra λ za koje proces konvergira.

Rešenje: Rekurentnom formulom

$$v_{n+1}(x) = \lambda \int_0^1 xt^2 v_n(t)\,dt + 1, \quad n = 0, 1, \ldots, \qquad v_0(x) = 1,$$

dobijamo niz aproksimacija

$$v_1(x) = 1 + \frac{x}{3}\lambda, \quad , \ldots, \quad v_n(x) = 1 + \frac{x}{3}\sum_{k=1}^{n} \frac{\lambda^k}{4^{k-1}}, \quad \ldots.$$

Tačno rešenje zadatka je

$$u(x) = \lim_{n\to\infty} v_n(x) = 1 + \frac{4x}{3}\sum_{k=1}^{\infty} \left(\frac{\lambda}{4}\right)^k = 1 + \frac{4\lambda}{3(4-\lambda)}\,x, \quad \text{za} \quad |\lambda| < 4.$$

10.3 *a)Dokazati da je rešenje graničnog problema*

$$-u''(x) = u(x), \quad 0 < x < 1, \qquad\qquad u(0) = 0, \quad u'(1) = 1,$$

i rešenje integralne jednačine

$$u(x) = x + \int_0^1 K(x,t)u(t)dt, \qquad K(x,t) = \begin{cases} t, & 0 \le t \le x \le 1 \\ x, & 0 \le x \le t \le 1 \end{cases}$$

b) Metodom uzastopnih aproksimacija odrediti dve aproksimacije rešenja integralne jednačine.

Rešenje: a) Integraljenjem diferencijalne jednačine od x do 1, uzimajući u obzir drugi granični uslov, dobijamo da je

$$u'(x) = 1 + \int_x^1 u(t)dt,$$

a integraljenjem dobijenog izraza od 0 do x, uzimajući u obzir prvi granični uslov, i parcijalnom integracijom dvostrukog integrala, konačno dobijamo

$$u(x) = x + \int_0^x \left(\int_s^1 u(t)\, dt \right) ds = x + \int_0^1 K(x,t)u(t)dt,$$

gde je $K(x,t)$ funkcija definisana zadatkom.
b) Računajući po rekurentnoj formuli

$$v_{n+1}(x) = x + \int_0^1 K(x,t)v_n(t)\, dt, \qquad n = 0, 1, \ldots,$$

dobijamo aproksimacije

$$v_0(x) = x, \qquad v_1(x) = \frac{3}{2}\, x - \frac{1}{6}\, x^3, \qquad v_2(x) = \frac{41}{24}\, x - \frac{1}{4}\, x^3 + \frac{1}{120}\, x^5.$$

10.4 *Metodom uzastopnih aproksimacija odrediti približno rešenje jednačine*

$$u(x) = x + \lambda \int_0^1 \frac{u(t)}{10 + x + t}\, dt, \quad \lambda \in R$$

Diskutovati konvergenciju metode.

Rešenje: Polazeći od početne aproksimacije $v_0(x) = x$ dobija se u prvom koraku

$$v_1(x) = x + \lambda \int_0^1 \frac{t}{10 + x + t}\, dt = x + \lambda \left(1 - (10 + x) \ln \frac{11 + x}{10 + x} \right)$$

Analitičko izračunavanje integrala je problem već u narednom koraku, što ukazuje na ograničenu primenljivost analitičkih metoda. Uniformna norma jezgra na datom intervalu je

$$M = \max_{0 \le x,t \le 1} \Big| \frac{1}{10 + x + t} \Big| = 0.1$$

te će metoda konvergirati ako je

$$|\lambda| < \frac{1}{M(b - a)} = 10$$

10.2 Metoda degenerisanih jezgara

Ako je jezgro Fredholm-ove integralne jednačine degenerisano,

$$K(x,t) = \sum_{k=1}^{n} \alpha_k(x)\beta_k(t),$$

rešenje se može napisati u obliku

$$u(x) = f(x) + \lambda \sum_{k=1}^{n} c_k \alpha_k(x), \qquad c_k = \int_a^b u(t)\beta_k(t)\,dt.$$

Konstante c_k su rešenja sistema linearnih jednačina

$$c_k = \int_a^b \left(f(t) + \lambda \sum_{j=1}^{n} c_j \alpha_j(t) \right) \beta_k(t)\,dt, \qquad k = 1, \ldots, n.$$

Kada se ovom metodom rešava Volttera-ova jednačina, problem se svodi na Cauchy-jev problem za sistem diferencijalnih jednačina prvog reda po funkcijama $c_k(x)$, $k = 1, \ldots, n$.

10.5 *Metodom degenerisanih jezgara rešiti jednačinu:*

$$u(x) = x^2 + \lambda \int_{-1}^{1} (x+t)u(t)\,dt$$

Rešenje: Jezgro je degenerisano, te se jednačina može napisati u obliku

$$u(x) = x^2 + \lambda \left(x \int_{-1}^{1} u(t)\,dt + \int_{-1}^{1} t\,u(t) \right)\,dt,$$

tj.

$$u(x) = x^2 + \lambda(c_1 x + c_0)$$

gde je:

$$c_1 = \int_{-1}^{1} u(t)\,dt = \int_{-1}^{1} \big(t^2 + \lambda(c_1 t + c_0)\big)\,dt = \frac{2}{3} + 2\lambda c_0$$

$$c_0 = \int_{-1}^{1} t\,u(t)\,dt = \int_{-1}^{1} t\big(t^2 + \lambda(c_1 t + c_0)\big)\,dt = \frac{2}{3}\lambda c_1$$

Iz dobijenog sistema linearnih jednačina računamo koeficijente rešenja,

$$\begin{aligned} -2\lambda c_0 + c_1 &= \frac{2}{3} \\ c_0 - \frac{2}{3}\lambda c_1 &= 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} c_0 &= \frac{4\lambda}{3(3-4\lambda^2)} \\ c_1 &= \frac{2}{3-4\lambda^2} \end{aligned}$$

uz uslov da je determinanta sistema $\frac{4}{3}\lambda^2 - 1 \neq 0$.

Stoga, rešenje date integralne jednačine je

$$u(x) = x^2 + \frac{2\lambda}{3-4\lambda^2}\left(x + \frac{2}{3}\lambda\right), \quad \text{za} \quad \lambda \neq \pm\frac{\sqrt{3}}{2}.$$

10.6 *Integralnu jednačinu*

$$u(x) - \lambda \int_0^\infty e^{-(x+t)} u(t)\, dt = f(x)$$

rešiti za $f(x) = x$: a) metodom uzastopnih aproksimacija, b) metodom degenerisanih jezgara.
c) Naći funkciju $f(x)$ tako da za $\lambda = 2$ postoji rešenje integralne jednačine.

Rešenje: a) Koristeći rekurentnu formulu

$$v_0(x) = x, \qquad v_{n+1}(x) = x + \lambda e^{-x} \int_0^\infty e^{-t} v_n(t)\, dt, \quad n = 0, 1, \ldots,$$

dobijamo niz aproksimacija

$$v_1(x) = x + \lambda e^{-x}, \qquad v_2(x) = x + \lambda\left(1 + \frac{\lambda}{2}\right)e^{-x}, \qquad \ldots$$

$$v_{n+1}(x) = x + \lambda e^{-x} \sum_{k=0}^{n} \left(\frac{\lambda}{2}\right)^k$$

koji konvergira ka tačnom rešenju jednačine

$$\lim_{n\to\infty} v_n(x) = u(x) = x + \frac{2\lambda}{2-\lambda}\, e^{-x} \qquad \text{za} \qquad |\lambda| < 2.$$

b) Jednačina ima degenerisano jezgro jer se može napisati u obliku

$$u(x) = x + \lambda e^{-x} \int_0^\infty e^{-t} u(t)\, dt = x + \lambda\, c\, e^{-x},$$

gde je $c = \int_0^\infty e^{-t} u(t)\, dt$. Zamenom dobijenog izraza za funkciju $u(x)$ u integralu koji predstavlja konstantu c dobijamo jednačinu po c,

$$c = \int_0^\infty e^{-t}(t + \lambda c e^{-t})\, dt = 1 + \frac{1}{2}\lambda c.$$

Za $\lambda \neq 2$ jednačina ima rešenje $c = 2/(2-\lambda)$. Stoga za $\lambda \neq 2$ postoji jedinstveno rešenje integralne jednačine i jednako je

$$u(x) = x + \frac{2\lambda}{2-\lambda}\, e^{-x}.$$

c) Za $\lambda = 2$ postoji beskonačno mnogo rešenja integralne jednačine

$$u(x) = f(x) + ce^{-x}, \qquad \text{ako je} \qquad \int_0^\infty f(x)e^{-x}\,dx = 0,$$

gde je c proizvoljna konstanta.

Poseban slučaj je za $f(x) = 0$, tj. kada je jednačina homogena. Rešenje je

$$u(x) = ce^{-x},$$

i predstavlja sopstvenu funkciju integralnog operatora kojim je definisana data jednačina. $\lambda = 2$ je karakteristična vrednost integralnog operatora.

10.7 *Zamenom jezgra degenerisanim jezgrom oblika* $\overline{K}(x,t) = a + b\,xt + c(xt)^2$, $a, b, c \in R$, *rešiti integralnu jednačinu*

$$u(x) + \int_0^1 \ln(1 + xt)u(t)\,dt = 1.$$

Računati sa pet decimala.

Rešenje: Asimptotskim razvojem jezgra integralne jednačine

$$K(x,t) = \ln(1 + xt) \approx xt - \frac{1}{2}(xt)^2 \equiv \overline{K}(x,t) \quad \text{za} \quad |xt| < 1$$

nalazimo aproksimaciju jezgra traženog oblika. Zamenom dobijamo integralnu jednačinu po približnom rešenju

$$v(x) + x\int_0^1 tv(t)\,dt - \frac{1}{2}x^2\int_0^1 t^2 v(t)\,dt = 1,$$

iz koje sledi da je

$$v(x) = 1 - c_1 x + c_2\frac{x^2}{2}.$$

Pri tome je

$$c_1 = \int_0^1 tv(t)dt = \int_0^1 t(1 - c_1 t + c_2\frac{t^2}{2})\,dt = \frac{1}{2} - \frac{1}{3}c_1 + \frac{1}{8}c_2,$$

$$c_2 = \int_0^1 t^2 v(t)\,dt = \int_0^1 t^2(1 - c_1 t + c_2\frac{t^2}{2})\,dt = \frac{1}{3} - \frac{1}{4}c_1 + \frac{1}{10}c_2.$$

Rešavanjem dobijenog sistema linearnih jednačina po c_1 i c_2 dobijamo približno rešenje

$$v(x) = 1 - 0.39932x + 0.12972x^2.$$

10.8 *Zamenom jezgra integralne jednačine odgovarajućim Taylor-ovim polinomom, metodom degenerisanih jezgara odrediti približno rešenje integralne jednačine*

$$u(x) = 1 + \int_0^{1/2} e^{-x^2t^2} u(t)\, dt$$

Rešenje: Jezgro aproksimiramo sa prva tri člana Taylor-ovog razvoja,

$$K(x,t) = e^{-x^2t^2} \approx \overline{K(x,t)} = 1 - x^2t^2 + \frac{1}{2}x^4t^4.$$

Zamenom u polaznoj jednačini, dobija se integralna jednačina po približnom rešenju $v(x)$ sa degenerisanim jezgrom,

$$v(x) = 1 + \int_0^{1/2} \big(1 - x^2t^2 + \frac{1}{2}x^4t^4\big)v(t)\, dt = 1 + c_0 + c_1x^2 + c_2x^4,$$

gde je

$$\begin{aligned} c_0 &= \textstyle\int_0^{1/2} v(t)\, dt = \frac{1}{2} + \frac{1}{2}\, c_0 + \frac{1}{24}\, c_1 + \frac{1}{160}\, c_2 \\ c_1 &= -\textstyle\int_0^{1/2} t^2 v(t)\, dt = -\frac{1}{24} - \frac{1}{24}\, c_0 - \frac{1}{160}\, c_1 - \frac{1}{896}\, c_2 \\ c_2 &= \textstyle\frac{1}{2}\int_0^{1/2} t^4 v(t)\, dt = \frac{1}{320} + \frac{1}{320}\, c_0 + \frac{1}{1792}\, c_1 + \frac{1}{9216}\, c_2 \end{aligned}$$

Rešenja sistema su $c_0 = 0.9930$, $c_1 = -0.0833$ i $c_2 = 0.0007$, pa je približno rešenje date integralne jednačine

$$v(x) = 1.9930 - 0.0833x^2 + 0.0007x^4.$$

10.9 *Aproksimacijom jezgra Taylor-ovim polinomom prvog stepena odrediti približno rešenje integralne jednačine*

$$u(x) = \cos x + \frac{2}{5}\int_0^{\pi/2} \sin\,(t\cos x)u(t)\, dt.$$

Rešenje: Aproksimacijom $\sin\,(t\cos x) \approx t\cos x$ dobijamo integralnu jednačinu sa degenerisanim jezgrom po približnom rešenju

$$v(x) = \cos x + \frac{2}{5}\int_0^{\pi/2} t\cos x v(t)\, dt.$$

Ako uvedemo oznaku $c = \frac{2}{5}\int_0^{\pi/2} tv(t)\, dt$, približno rešenje je $v(x) = (1+c)\cos x$. Zamenom ove reprezentacije u izrazu za konstantu c, dobijamo da je

$$c = \frac{\pi - 2}{7 - \pi}, \qquad \text{tj.} \qquad v(x) = \frac{5}{7-\pi}\cos x.$$

10.10 *Odrediti približno rešenje integralne jednačine*

$$u(x) - \int_0^1 \frac{xt}{\sqrt{1+0.1xt}} u(t)\,dt = e^{-x}$$

zamenom jezgra zbirom prva tri člana Taylor-ovog razvoja. Računati sa četiri decimale.

Rešenje: Aproksimacijom jezgra polinomom

$$\frac{xt}{\sqrt{1+0.1xt}} \approx xt - \frac{0.1}{2}(xt)^2 + \frac{0.03}{8}(xt)^3$$

problem svodimo na rešavanje integralne jednačine sa degenerisanim jezgrom

$$v(x) - \int_0^1 \big(xt - \frac{0.1}{2}(xt)^2 + \frac{0.03}{8}(xt)^3\big) v(t)\,dt = e^{-x}.$$

Algoritmom opisanim u prethodnom zadatku dobijamo približno rešenje

$$v(x) = e^{-x} + 0.3917x - 0.0128x^2 + 0.0007x^3.$$

10.11 *Zamenom jezgra zbirom prva tri člana Taylor-ovog razvoja odrediti približno rešenje integralne jednačine*

$$u(x) - \int_0^1 \sinh(xt) u(t)\,dt = 1 - x^2.$$

Rešenje: Kao i u prethodnim zadacima, zamenom jezgra polinomom

$$\sinh(xt) \approx xt + \frac{(xt)^3}{3!} + \frac{(xt)^5}{5!}$$

dobijamo integralnu jednačinu sa degenerisanim jezgrom po približnom rešenju $v(x)$. Primenom već opisanog algoritma dobijamo da je

$$v(x) = 1 + 0.3833x - x^2 + 0.0273x^3 + 0.0008x^5.$$

10.12 *a) Rešiti integralnu jednačinu*

$$u(x) = \lambda \int_{-1}^1 (1+xt) u(t)\,dt + ax^2 + bx + c.$$

b)Naći sve vrednosti parametara a, b i c za koje ova integralna jednačina ima rešenje za svako λ, ako je $a^2 + b^2 + c^2 = 1$.

Rešenje: a)Pošto je jezgro jednačine degenerisano, rešićemo je metodom degenerisanih jezgara:

$$u(x) = \lambda \int_{-1}^{1} u(t)\,dt + \lambda x \int_{-1}^{1} t\,u(t)\,dt + ax^2 + bx + c.$$

Ako označimo sa

$$d_1 = \int_{-1}^{1} u(t)\,dt \qquad\qquad d_2 = \int_{-1}^{1} t\,u(t)\,dt,$$

onda je

$$u(x) = (\lambda d_1 + c) + (\lambda d_2 + b)x + ax^2,$$

te je

$$d_1 = \int_{-1}^{1} \big((\lambda d_1 + c) + (\lambda d_2 + b)t + at^2\big)\,dt = 2\lambda d_1 + 2c + \frac{2}{3}a,$$

$$d_2 = \int_{-1}^{1} t\big((\lambda d_1 + c) + (\lambda d_2 + b)t + at^2\big)\,dt = \frac{2}{3}\lambda d_2 + \frac{2}{3}b.$$

Konstante d_1 i d_2 su stoga rešenja sistema linearnih jednačina

$$d_1(1 - 2\lambda) = \frac{2}{3}a + 2c, \qquad\qquad d_2(1 - \frac{2}{3}\lambda) = \frac{2}{3}b.$$

Determinanta ovog sistema je različita od nule ako je $\lambda \neq 1/2$ i $\lambda \neq 3/2$. Tada je

$$d_1 = \frac{2}{3}\,\frac{a + 3c}{1 - 2\lambda}, \qquad d_2 = \frac{2b}{3 - 2\lambda},$$

pa jednačina ima za svako a, b i c rešenje

$$u(x) = \frac{2\lambda a + 3c}{3(1 - 2\lambda)} + \frac{3b}{3 - 2\lambda}x + ax^2.$$

Ako je $\lambda = 1/2$, da bi sistem imao rešenje, treba da bude $a = -3c$. Tada je d_1 proizvoljno, $d_2 = b$, a rešenje integralne jednačine je

$$u(x) = d_1 + \frac{3}{2}bx + ax^2.$$

U slučaju da je $\lambda = 3/2$, sistem ima rešenje ako je $b = 0$. To rešenje je $d_1 = -(a + 3c)/3$ i d_2 proizvoljno, pa je rešenje jednačine

$$u(x) = -\frac{1}{2}(a + c) + d_2 x + ax^2.$$

b) Uzimajući u obzir sva ograničenja koja su u toku rešavanja integralne jednačine postavljena za konstante a, b i c, kao i ograničenje dato u zadatku, sledi da će jednačina imati rešenje za svako λ ukoliko je

$$a + 3c = 0, \qquad b = 0, \qquad a^2 + b^2 + c^2 = 1,$$

tj. ako je

$$a = -3/\sqrt{10}, \quad b = 0, \quad c = 1/\sqrt{10},$$

ili

$$a = 3/\sqrt{10}, \quad b = 0, \quad c = -1/\sqrt{10}.$$

10.13 *Metodom degenerisanih jezgara rešiti integralnu jednačinu*

$$u(x) = 2x + \int_0^x xt\, u(t)\, dt$$

Rešenje: Data jednačina je Volterra-ova jednačina druge vrste. Može se zapisati u sledećem obliku

$$u(x) = x\big(2 + v(x)\big),$$

gde je

$$v(x) = \int_0^x tu(t)\, dt, \qquad \text{tj.} \qquad v(x) = \int_0^x t^2\big(2 + v(t)\big)\, dt.$$

Diferenciranjem ove funkcije po x, dobijamo Cauchy-jev problem za funkciju $v(x)$,

$$v'(x) = x^2\big(2 + v(x)\big), \qquad v(0) = 0.$$

U opštem slučaju, Cauchy-jev problem se numerički rešava. U konkretnom slučaju moguće je naći analitičko rešenje,

$$v(x) = 2e^{x^3/3} - 2, \quad \Longrightarrow \quad u(x) = 2xe^{x^3/3}.$$

10.3 Metoda kvadraturnih formula

Metoda kvadraturnih formula se zasniva na aproksimaciji integrala nekom kvadraturnom formulom

$$v_k = f(x_k) + \lambda \sum_{j=1}^{n} c_j K(x_k, x_j) v_j, \qquad k = 1, \ldots, n,$$

gde su x_k čvorovi a c_k koeficijenti izabrane kvadraturne formule, i $v_k \approx u(x_k)$.

10.14 *Korišćenjem Simpson-ove kvadraturne formule sa korakom $h = 0.5$ naći približno rešenje integralne jednačine:*

$$u(x) + \int_0^1 xe^{xt} u(t) dt = e^x$$

Rešenje: Čvorovi kvadraturne formule su $x_0 = 0$, $x_1 = 0.5$ i $x_2 = 1$, a sama formula glasi

$$\int_0^1 f(x)dx \approx \frac{1}{6}\Big(f(x_0) + 4f(x_1) + f(x_2)\Big).$$

Kada se primeni za aproksimaciju integrala u jednačini, dobija se jednačina približnog rešenja

$$v(x) + \frac{1}{6}\Big(xv_0 + 4xe^{x/2}v_1 + xe^x v_2\Big) = e^x.$$

Da bismo odredili nepoznate koeficijente v_k, $k = 0, 1, 2$, napišimo prethodni izraz u čvorovima kvadraturne formule,

$$\begin{aligned}
x = x_0: &\quad v_0 + \tfrac{x_0}{6}(v_0 + 4e^{x_0/2}v_1 + e^{x_0}v_2) = e^{x_0}\\
x = x_1: &\quad v_1 + \tfrac{x_1}{6}(v_0 + 4e^{x_1/2}v_1 + e^{x_1}v_2) = e^{x_1}\\
x = x_2: &\quad v_2 + \tfrac{x_2}{6}(v_0 + 4e^{x_2/2}v_1 + e^{x_2}v_2) = e^{x_2}
\end{aligned}$$

Rešenjem dobijenog sistema linearnih jednačina

$$\begin{array}{lcll}
v_0 & = & 1 & \\
1.4280v_1 + 0.1374v_2 & = & 1.5654 & \Longrightarrow \quad v_1 = 1.0002\\
1.0991v_1 + 1.4530v_2 & = & 2.5516 & \quad\ \ v_2 = 0.9995
\end{array}$$

određeno je približno rešenje date integralne jednačine

$$v(x) = e^x - \frac{x}{6}\big(1 + 4.0008e^{x/2} + 0.9995e^x\big).$$

10.15 *Koristeći trapeznu formulu sa korakom $h = 0.25$ izračunati na intervalu $(0, 1)$ približno rešenje integralne jednačine*

$$u(x) + \int_0^1 \frac{u(t)}{1 + x^2 + t^2}\, dt = 1.5 - x^2.$$

Rešenje: Aproksimacijom integrala trapeznom formulom, dobija se približno rešenje

$$v(x) + \frac{1}{8}\left(\frac{v_0}{1 + x^2} + 2\sum_{i=1}^{3}\frac{v_i}{1 + x^2 + t_i^2} + \frac{v_4}{2 + x^2}\right) = 1.5 - x^2,$$

Da bismo odredili nepoznate koeficijente $v_i = v(x_i)$, $i = 0, \ldots, 4$, napišimo prethodni izraz u čvorovima kvadraturne formule,

$$v_i + \frac{1}{8}\left(\frac{v_0}{1 + x_i^2} + \frac{2v_1}{1.0625 + x_i^2} + \frac{2v_2}{1.25 + x_i^2} + \frac{2v_3}{1.5625 + x_i^2} + \frac{v_4}{2 + x_i^2}\right) = 1.5 - x_i^2,$$
$$i = 0, \ldots, 4.$$

Rešenja sistema su približne vrednosti rešenja integralne jednačine u čvorovima

x_i	0	0.25	0.50	0.75	1.0
v_i	0.92059	0.88763	0.78215	0.54605	0.18690

10.16 *Korišćenjem osnovne Simpson-ove kvadraturne formule odrediti približno rešenje integralne jednačine*

$$u(x) + \int_0^{\pi/2} (x + \sin t)u(t)\,dt = x + \cos x, \qquad x \in [0, \pi/2].$$

Računati sa pet decimala.

Rešenje: Aproksimacijom integrala osnovnom Simpson-ovom kvadraturnom formulom dobijamo približno rešenje jednačine

$$(*) \qquad v(x) = -\frac{\pi}{12}\left(xv_0 + 4(x + \frac{\sqrt{2}}{2})v_1 + (x+1)v_2\right) + x + \cos x,$$

gde je $v_i = v(x_i)$ i $x_0 = 0$, $x_1 = \pi/4$, $x_2 = \pi/2$. Pišući približno rešenje $(*)$ u čvorovima kvadraturne formule x_i, $i = 0, 1, 2$, dobijamo sledeći sistem linearnih jednačina

$$v_0 + \frac{\pi}{12}2\sqrt{2}v_1 + \frac{\pi}{12}v_2 = 1$$

$$\frac{\pi^2}{48}v_0 + (1 + \frac{\pi}{3\sqrt{2}} + \frac{\pi^2}{12})v_1 + \frac{\pi}{12}(\frac{\pi}{4} + 1)v_2 = \frac{\pi}{4} + \frac{\sqrt{2}}{2}$$

$$\frac{\pi^2}{24}v_0 + \frac{\pi}{3}(\frac{\pi}{2} + \frac{\sqrt{2}}{2})v_1 + (1 + \frac{\pi}{12} + \frac{\pi^2}{24})v_2 = \frac{\pi}{2}$$

Njegovo rešenje je

$$v_0 = 0.59928, \qquad v_1 = 0.52690, \qquad v_2 = 0.04034.$$

Zamenom ovih vrednosti u izrazu $(*)$ dobijamo analitički izraz za približno rešenje

$$v(x) = \cos x + 0.28078x - 0.40072.$$

10.17 *Primenom Gauss-ove kvadraturne formule sa tri čvora, naći približno rešenje integralne jednačine*

$$u(x) - \frac{1}{2}\int_0^1 e^{xt}u(t)\,dt = 1 - \frac{1}{2x}(e^x - 1).$$

Računati sa pet decimala.

Rešenje: Gauss-ova kvadraturna formula je

$$\int_{-1}^{1} f(x)\,dx \approx \frac{1}{9}\left(5f\big(-\sqrt{\frac{3}{5}}\big) + 8f(0) + 5f\big(\sqrt{\frac{3}{5}}\big)\right),$$

te ćemo je, s obzirom da je interval integracije u integralnoj jednačini $[0,1]$, smenom $x = \frac{1}{2}(1+t)$ prilagoditi zadatku,

$$\int_0^1 f(x)\,dx = \frac{1}{2}\int_{-1}^{1} f\big(\frac{1+t}{2}\big)\,dt \approx \frac{1}{18}\big(5f(x_0) + 8f(x_1) + 5f(x_2)\big),$$

gde su slike čvorova Gauss-ove kvadraturne formule na intervalu $[0,1]$

$$x_0 = \frac{1}{2}\big(1 - \sqrt{\frac{3}{5}}\big) = 0.11270, \qquad x_1 = 0.5, \qquad x_2 = \frac{1}{2}\big(1 + \sqrt{\frac{3}{5}}\big) = 0.88730,$$

Pisanjem integralne jednačine u čvorovima x_i, $i = 0,1,2$, i primenom izvedene formule, dobija se sistem linearnih jednačina po $v_i \approx u(x_i)$

$$v_i - \frac{1}{36}\left(5e^{x_0x_i}v_0 + 8e^{x_1x_i}v_1 + 5e^{x_2x_i}v_2\right) = 1 - \frac{1}{2x_i}(e^{x_i} - 1), \qquad i = 0,1,2.$$

Njegovo rešenje je sa tačnošću 10^{-5}

$$v_0 = v_1 = v_2 = 1,$$

(što je i tačno rešenje, $u(x) \equiv 1$), pa se rešenje integralne jednačine može aproksimirati funkcijom

$$v(x) = \frac{1}{36}\left(5e^{0.11270x} + 8e^{0.5x} + 5e^{0.88730x}\right) + 1 - \frac{1}{2x}(e^x - 1).$$

10.18 *Korišćenjem trapezne kvadraturne formule sa korakom $h = 0.2$ na intervalu $[0,1]$ naći približno rešenje integralne jednačine:*

$$u(x) = \int_0^x \cos(t-x)u(t)dt - 1 + x + \cos(x)$$

Rešenje: Integralna jednačina u čvorovima kvadraturne formule $x_i = 0.2i$ $(i = 0,\ldots,5)$ glasi:

$$u(x_i) = \int_0^{x_i} \cos(t-x_i)u(t)dt - 1 + x_i + \cos(x_i)$$

Kada se integrali u gornjim jednačinama aproksimiraju trapeznom kvadraturnom formulom dobija se trougaoni sistem linearnih jednačina po približnim vrednostima rešenja $v_i \approx u(x_i)$:

$$\begin{aligned} v_0 =& f(x_0) = 0 \\ v_i =& 0.1\Big(v_0\cos(0 - 0.2i) + 2\sum_{j=1}^{i-1} v_j\cos(0.2(j-i)) + v_i\cos(0.2(i-i))\Big) \\ & - 1 + 0.2i + \cos(0.2i), \qquad\qquad i = 1,\ldots,5, \end{aligned}$$

odnosno, u razvijenom obliku:

$$\begin{aligned}
v_0 &= 0\\
v_1 &= 0.0980067v_0 + 0.1v_1 + 0.1800666\\
v_2 &= 0.0921061v_0 + 0.1960133v_1 + 0.1v_2 + 0.3210610\\
v_3 &= 0.0825336v_0 + 0.1842122v_1 + 0.1960133v_2 + 0.1v_3 + 0.4253356\\
v_4 &= 0.0696707v_0 + 0.1650671v_1 + 0.1842122v_2 + 0.1960133v_3 + 0.1v_4\\
&+ 0.4967067\\
v_5 &= 0.0540302v_0 + 0.1393413v_1 + 0.1650671v_2 + 0.1842122v_3 + 0.1960133v_4\\
&+ 0.1v_5 + 0.5403023
\end{aligned}$$

čije rešenje je

x_i	0.0	0.2	0.4	0.6	0.8	1.0
v_i	0.0	0.2000740	0.4003091	0.6007307	0.8013615	1.0022202

10.19 *Naći približno rešenje integralne jednačine*

$$u(x) - \int_0^x e^{-x-t}u(t)\,dt = \frac{1}{2}(e^{-x} + e^{-3x})$$

primenom trapezne formule sa čvorovima 0, 0.2, 0.4, 0.6, 0.8 *i* 1. *Računati na četiri decimale.*

Rešenje: Integralna jednačina napisana u čvorovima $x_i = 0.2i$, $i = 0, \ldots, 5$, ima oblik

$$u(x_i) - \int_0^{x_i} e^{-x_i-t}u(t)\,dt = \frac{1}{2}(e^{-x_i} + e^{-3x_i}).$$

Aproksimiranjem integrala trapeznom formulom sa zadatim čvorovima, dobijamo sistem linearnih jednačina sa trougaonom matricom sistema po nepoznatim vrednostima $v_i \approx u(x_i)$

$$\begin{aligned}
v_0 &= 1\\
v_1 - 0.1\big(e^{-x_1-x_0}v_0 + e^{-2x_1}v_1\big) &= \frac{1}{2}(e^{-x_1} + e^{-3x_1}),\\
v_i - 0.1\big(e^{-x_i-x_0}v_0 + 2\sum_{k=1}^{i-1} e^{-x_i-x_k}v_k + e^{-2x_i}v_i\big) &= \frac{1}{2}(e^{-x_i} + e^{-3x_i}),\\
& \qquad i = 2, 3, 4, 5,
\end{aligned}$$

čije rešenje je

x_i	0	0.2	0.4	0.6	0.8	1.0
v_i	1	0.8207	0.6731	0.5518	0.4522	0.3705

FORTRAN Implementacija metode kvadraturnih formula biće demonstrirana na primeru trapezne formule. Implementacija zahteva da se formula za aproksimaciju vrednosti rešenja u datoj tački napiše u razvijenom obliku, tako da se na levu stranu ove jednačine prebace svi izrazi koji sadrže vrednost rešenja u tekućoj tački, a na desnu stranu svi izrazi koji sadrže već poznate vrednosti rešenja u prethodnim tačkama. Polazeći od vrednosti rešenja u početnoj tački, koje je jednako $f(a)$, na ovaj način se mogu redom izračunati vrednosti rešenja u svim ostalim tačkama. Na taj način, procedura `quadrature()` ima sledeći oblik:

```
module quadrature_module
  use matheval_module
  implicit none
  public quadrature

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string

contains
  ! Procedura quadrature() odredjuje resenja date Volterra-ove
  ! jednacine metodom trapezne kvadraturne formule. Parametri procedure
  ! su:
  !   n - broj tacaka u kojima se trazi resenje jednacine
  !   a, h - donja granica intervala integracije i korak integracije
  !   f - funkcija koja predstavlja prvi sabirak na desnoj strani
  !       jednacine
  !   lambda - mnozilac integrala na desnoj strani jednacine
  !   K - kernel problema
  !   v - vektor u koji ce biti smestena resenja jednacine
  ! Broj tacaka u kojima se trazi resenje ne sme biti manji od 2. Korak
  ! integracije mora biti veci od 0. Funkcije koje figurisu u problemu
  ! moraju biti sintaksno ispravno zadate. Pretpostavlja se da je polje
  ! u koje ce biti smesteni rezultati alocirano izvan procedure.
  subroutine quadrature (n, a, h, f, lambda, K, v)
    integer, intent(in) :: n
    real(kind=8), intent(in) :: a, h
    type(string), intent(in) :: f
    real(kind=8), intent(in) :: lambda
    type(string), intent(in) :: K
    real(kind=8), dimension(0:n-1), intent(out) :: v
    integer(kind=8) :: f_evaluator, K_evaluator ! Evaluatori za funkcije
        ! na desnoj strani jednacine.
    real(kind=8) :: nom, den ! Brojilac i imenilac izraza za
        ! odredjivanje resenja u datoj tacki.
    integer :: i, j ! Brojaci u petljama.

    ! Proverava se validnost argumenata procedure i zaustavlja program
    ! ako se uoce problemi u vezi sa tim.
    if (n<2) stop
    if (h<=0) stop
    f_evaluator=evaluator_create(f%chars)
    if (f_evaluator==0) stop
    K_evaluator=evaluator_create(K%chars)
    if (K_evaluator==0) stop
```

```
    ! Izracunava se resenje jednacine u pocetnoj tacki.
    v(0)=evaluator_evaluate_x(f_evaluator,a)

    ! Izracunavaju se resenja u ostalim tackama.
    do i=1, n-1
       nom=evaluator_evaluate_x(f_evaluator,a+i*h)
       nom=nom+lambda*h/2*v(0)*evaluator_evaluate(K_evaluator,2,"t x",(/a,a+i*h/))
       do j=1, i-1
          nom=nom+lambda*h*v(j)*evaluator_evaluate(K_evaluator,2,"t x",(/a+j*h,a+i*h/))
       end do
       den=(1-lambda*h/2*evaluator_evaluate(K_evaluator,2,"t x",(/a+i*h,a+i*h/)))
       v(i)=nom/den
    end do

    ! Unistavaju se korisceni evaluatori.
    call evaluator_destroy(f_evaluator)
    call evaluator_destroy(K_evaluator)
  end subroutine quadrature
end module quadrature_module
```

10.4 Varijacione metode

Približno rešenje se određuje u obliku

$$v(x) = \sum_{k=1}^{n} c_k \phi_k(x)$$

Funkcije $\phi_k(x)$, $k = 1, \ldots, n$ su poznate linearno nezavisne funkcije, a konstante c_k, $k = 1, \ldots, n$, se određuju minimizacijom funkcije greške

$$R(x; c_1, \ldots, c_n) = v(x) - f(x) - \lambda \int_a^b K(x,t) v(t)\, dt$$

Metoda kolokacije – U n izabranih tačaka intervala $[a, b]$ je

$$R(x_k; c_1, \ldots, c_n) = 0, \qquad k = 1, \ldots, n.$$

Ritz–Galerkin-ova metoda

$$(R, \phi_k) = \int_a^b R(x; c_1, \ldots, c_n)\phi_k(x)\, dx = 0, \qquad k = 1, \ldots, n.$$

Metoda najmanjih kvadrata

$$\|R\|^2 = \int_a^b R^2(x; c_1, \ldots, c_n)\, dx = \min \quad \longrightarrow \quad \int_a^b R \frac{\partial R}{\partial c_k}\, dx = 0, \qquad k = 1, \ldots, n.$$

10.20 *Metodom kolokacije rešiti integralnu jednačinu*

$$u(x) = \int_0^\pi \sin(x+t)u(t)\,dt + 1 + \cos x,$$

ako su tačke kolokacije $x_0 = 0$, $x_1 = \pi/2$, $x_2 = \pi$, *a bazisne funkcije* $\phi_1(x) = 1$, $\phi_2(x) = \cos x$, $\phi_3(x) = \sin x$.

Rešenje: Rešenje tražimo u obliku funkcije

$$v(x) = c_1 + c_2\cos x + c_3 \sin x.$$

Zamenom u integralnoj jednačini dobijamo funkciju greške

$$\begin{aligned} R(x; c_1, c_2, c_3) &= v(x) - \int_0^\pi \sin(x+t)v(t)\,dt - 1 - \cos x \\ &= c_1(1 - 2\cos x) + c_2(\cos x - \frac{\pi}{2}\sin x) + c_3(\sin x - \frac{\pi}{2}\cos x) - 1 - \cos x \end{aligned}$$

Metodom kolokacije se konstante c_k, $k = 1, 2, 3$, određuju iz uslova

$$R(x_k; c_1, c_2, c_3) = 0, \qquad k = 0, 1, 2,$$

gde su x_k date tačke kolokacije. Ovi uslovi se svode na sistem linearnih jednačina

$$-c_1 + c_2 - \frac{\pi}{2}c_3 = 2, \qquad c_1 - \frac{\pi}{2}c_2 + c_3 = 1, \qquad 3c_1 - c_2 + \frac{\pi}{2}c_3 = 0.$$

Rešenje sistema određuje traženo približno rešenje integralne jednačine

$$v(x) = 1 + \frac{12}{4-\pi^2}\cos x + \frac{6\pi}{4-\pi^2}\sin x.$$

Zamenom u jednačinu dokazuje se da je to tačno rešenje.

10.21 *Metodom kolokacije naći rešenje integralne jednačine*

$$u(x) - \int_0^1 \frac{t^2 u(t)}{x^2+t^2}dt = x \arctan\frac{1}{x}$$

Rešenje: Aproksimirajmo rešenje linearnom funkcijom $v(x) = c_0 + c_1 x$. Funkcija greške je

$$\begin{aligned} R(x; c_0, c_1) &= v(x) - \left(x\arctan\tfrac{1}{x} + \textstyle\int_0^1 \frac{t^2 v(t)}{x^2+t^2}dt\right) = \\ &= x\arctan\tfrac{1}{x}(c_0 - 1) + (x - \tfrac{1}{2} + \tfrac{x^2}{2}\ln\tfrac{x^2+1}{x^2})c_1 \end{aligned}$$

Uslovi kolokacije $R(0;c_0,c_1)=0$ i $R(1;c_0,c_1)=0$ daju sistem jednačina

$$\begin{aligned} -\frac{1}{2}\,c_1 &= 0 \\ \frac{\pi}{4}\,c_0+\frac{1}{2}(1+\ln 2)\,c_1 &= \frac{\pi}{4} \end{aligned} \qquad\Longrightarrow\qquad \begin{aligned} c_0 &= 1 \\ c_1 &= 0 \end{aligned}$$

pa je aproksimacija rešenja $v(x)=1$ (to je tačno rešenje).

10.22 *Metodom Galerkin-a naći približno rešenje jednačine*

$$u(x)=2+\int_0^{\pi/2}\sin x\sin t u(t)\,dt$$

ako su bazisne funkcije $\phi_1(x)=\sin x$, $\phi_2(x)=\cos x$ *i* $\phi_3(x)=\sin 2x$. *Računati sa pet decimala.*

Rešenje: Aproksimaciju tražimo u obliku

$$v(x)=c_1\sin x+c_2\cos x+c_3\sin 2x+2.$$

Zamenom aproksimacije u integralnu jednačinu definišemo funkciju greške

$$\begin{aligned} R(x;c_1,c_2,c_3) &= c_1\sin x+c_2\cos x+c_3\sin 2x \\ &\quad -\sin x\int_0^{\pi/2}\sin t(c_1\sin t+c_2\cos t+c_3\sin 2t+2)\,dt \\ &= c_1(1-\frac{\pi}{4})\sin x+c_2(\cos x-\frac{1}{2}\sin x)+c_3(\sin 2x-\frac{2}{3}\sin x)-2\sin x \end{aligned}$$

Uslovi ortogonalnosti greške na bazisne funkcije

$$\int_0^{\pi/2}R(x;c_1,c_2,c_3)\phi_j(x)\,dx=0,\qquad j=1,2,3,$$

daju sledeći sistem linearnih jednačina za izračunavanje nepoznatih koeficijenata c_i

$$\begin{aligned} c_1\big(1-\frac{\pi}{4}\big)\frac{\pi}{4}+c_2\big(\frac{1}{2}-\frac{\pi}{8}\big)+c_3\big(\frac{2}{3}-\frac{\pi}{6}\big) &= \frac{\pi}{2} \\ c_1\big(1-\frac{\pi}{4}\big)\frac{1}{2}+c_2\big(\frac{\pi}{4}-\frac{1}{4}\big)+c_3\frac{1}{3} &= 1 \\ c_1\big(1-\frac{\pi}{4}\big)\frac{2}{3}+c_2\frac{1}{3}+c_3\big(\frac{\pi}{4}-\frac{4}{9}\big) &= \frac{4}{3} \end{aligned}$$

Rešavanjem sistema dobijamo približno rešenje

$$v(x)=\frac{8}{4-\pi}\sin x+2\qquad\qquad(c_2=c_3=0),$$

što je i tačno rešenje.

10.23 *Metodom Galerkin-a rešiti integralnu jednačinu*

$$u(x) = 1 + \int_0^1 (x^2t^2 + 3x)u(t)\,dt,$$

ako su bazisne funkcije $\phi_i(x)$, $i = 1, \ldots, n-1$,

$$\phi_i(x) = \begin{cases} n\,x - i + 1, & x \in [(i-1)/n, i/n] \\ -n\,x + i + 1, & x \in [i/n, (i+1)/n] \\ 0, & x \notin [(i-1)/n, (i+1)/n] \end{cases}$$

Uzeti da je $n = 5$.

Rešenje: Aproksimaciju rešenja tražimo u obliku

$$v(x) = \sum_{i=1}^{n-1} c_i\phi_i(x).$$

Greška aproksimacije je

$$\begin{aligned} R(x; c_1, \ldots, c_{n-1}) &= \sum_{i=1}^{n-1} c_i\phi_i(x) - 1 - \int_0^1 (x^2t^2 + 3x) \sum_{i=1}^{n-1} c_i\phi_i(t)\,dt \\ &= \sum_{i=1}^{n-1} c_i \left(\phi_i(x) - \int_0^1 (x^2t^2 + 3x)\phi_i(t)\,dt \right) - 1. \end{aligned}$$

Koeficijenti c_i se Galerkin-ovom metodom određuju iz uslova

$$\int_0^1 R(x; c_1, \ldots, c_{n-1})\phi_j(x)\,dx = 0, \qquad j = 1, \ldots, n-1,$$

tj. kao rešenja sistema linearnih jednačina

$$\sum_{i=1}^{n-1} c_i \int_0^1 \left(\phi_i(x) - \int_0^1 (x^2t^2 + 3x)\phi_i(t)\,dt \right) \phi_j(x)\,dx = \int_0^1 \phi_j(x)\,dx, \qquad j = 1, \ldots, n-1.$$

Radi lakšeg izračunavanja integrala, u funkcije $\phi_i(x)$, $i = 1, \ldots, n-1$, uvedimo smenu $y = nx - i$,

$$\phi_i(y) = \begin{cases} y + 1, & y \in [-1, 0] \\ -y + 1, & y \in [0, 1] \\ 0, & y \notin [-1, 1] \end{cases}$$

Tada je

$$\int_0^1 \phi_i(x)\,dx = \frac{1}{n} \int_{-1}^1 \phi_i(y)\,dy = \frac{1}{n} \left(\int_{-1}^0 (y+1)\,dy + \int_0^1 (-y+1)\,dy \right) = \frac{1}{n}$$

i

$$\int_0^1 \phi_i(x)\phi_j(x)\,dx = \begin{cases} \frac{1}{n}\int_0^1 y(1-y)\,dy = \frac{1}{6n}, & |i-j|=1, \\ \frac{1}{n}\int_{-1}^0 (1-y)^2\,dy + \frac{1}{n}\int_0^1 (1+y)^2\,dy = \frac{14}{3n}, & i=j, \\ 0 & |i-j|>1. \end{cases}$$

Koristeći prethodne izraze, za $n = 5$ dobijamo trodijagonalni sistem linearnih jednačina po c_i,

$$\begin{aligned} 4.54623c_1 + 0.04511c_2 &= 1 \\ -0.07489c_1 + 4.42111c_2 - 0.08556c_3 &= 1 \\ -0.20556c_2 + 4.27978c_3 - 0.24076c_4 &= 1 \\ -0.36076c_3 + 4.10303c_4 &= 1 \end{aligned}$$

Rešenje ovog sistema određuje traženu aproksimaciju

$$v(x) = 0.2176\phi_1(x) + 0.2349\phi_2(x) + 0.2599\phi_3(x) + 0.2666\phi_4(x).$$

10.24 *Odrediti prve dve sopstvene vrednosti integralne jednačine*

$$u(x) = \lambda \int_0^1 K(x,t)u(t)\,dt,$$

gde je:

$$K(x,t) = \begin{cases} t, & t \le x \\ x, & t > x \end{cases}$$

Sopstvene vrednosti su vrednosti skalara λ za koje homogena jednačina ima netrivijalno rešenje.

Rešenje: Neka je približno rešenje oblika $v(x) = c_1x + c_2x^2$. Funkcija greške je

$$\begin{aligned} R(x;c_1,c_2) &= v(x) - \lambda\Big(\int_0^x tv(t)dt + x\int_x^1 v(t)dt\Big) = \\ &= c_1\Big((1-\tfrac{\lambda}{2})x + \tfrac{\lambda}{6}x^3\Big) + c_2\Big(-\tfrac{\lambda}{3}x + x^2 + \tfrac{\lambda}{12}x^4\Big) \end{aligned}$$

Konstante c_1 i c_2 se odredjuju iz uslova Galerkin-ove metode $\int_0^1 R(x;c_1,c_2)x^k dx = 0,\ \ k = 1,2$, što daje sistem linearnih jednačina

$$\begin{aligned} c_1(120-48\lambda) + c_2(90-35\lambda) &= 0 \\ c_1(630-245\lambda) + c_2(504-180\lambda) &= 0 \end{aligned}$$

Sistem će imati netrivijalno rešenje ako je determinanta sistema jednaka 0, tj. ako je

$$\begin{vmatrix} 120-48\lambda & 90-35\lambda \\ 630-245\lambda & 504-180\lambda \end{vmatrix} = 0$$

odakle se izračunavaju tražene sopstvene vrednosti $\lambda_0 = 2.4680$ i $\lambda_1 = 23.5267$.

10.25 *Metodom najmanjih kvadrata odrediti aproksimaciju rešenja integralne jednačine*

$$\int_0^1 K(x,t)u(t)\,dt = x - 2x^3 + x^4, \qquad K(x,t) = \begin{cases} x(1-t), & 0 \le x \le t \le 1, \\ t(1-x), & 0 \le t \le x \le 1 \end{cases}.$$

polinomom drugog stepena. Računati na četiri decimale.

Rešenje: Zamenom u integralnoj jednačini približnog rešenja $v(x) = c_1 + c_2x + c_3x^2$ dobijamo funkciju greške

$$\begin{aligned} R(x;c_1,c_2,c_3) &= \int_0^1 K(x,t)v(t)\,dt - x + 2x^3 - x^4 \\ &= \frac{1}{2}(x-x^2)c_1 + \frac{1}{6}(x-x^3)c_2 + \frac{1}{12}(x-x^4)c_3 - x + 2x^3 - x^4. \end{aligned}$$

Koeficijenti c_i, $i = 1,2,3$, se određuju minimizacijom integrala

$$I(c_1,c_2,c_3) = \int_0^1 R(x;c_1,c_2,c_3)^2\,dx = \min \qquad \text{tj.} \qquad \frac{\partial I}{\partial c_k} = 0, \quad k = 1,2,3.$$

Rešenje dobijenog sistema linearnih jednačina

$$\int_0^1 \frac{1}{2}\left((x-x^2)c_1 + \frac{1}{6}(x-x^3)c_2 + \frac{1}{12}(x-x^4)c_3 - x + 2x^3 - x^4\right)(x-x^2)\,dx = 0$$

$$\int_0^1 \frac{1}{2}\left((x-x^2)c_1 + \frac{1}{6}(x-x^3)c_2 + \frac{1}{12}(x-x^4)c_3 - x + 2x^3 - x^4\right)(x-x^3)\,dx = 0$$

$$\int_0^1 \frac{1}{2}\left((x-x^2)c_1 + \frac{1}{6}(x-x^3)c_2 + \frac{1}{12}(x-x^4)c_3 - x + 2x^3 - x^4\right)(x-x^4)\,dx = 0$$

određuje aproksimaciju

$$v(x) = 0.1920 + 11.0600x - 11.0658x^2.$$

10.26 *Ako se prilikom rešavanja integralne jednačine*

$$u(x) = x^2 + \frac{12}{13}\int_0^1 u(t)\,dt$$

metodom najmanjih kvadrata sa bazisnim funkcijama $\phi_k(x) = x^{k-1}$, $k = 1,2,3$, ne vrše zaokruživanja, dobiće se tačno rešenje integralne jednačine. Dokazati.

Rešenje: Aproksimaciju rešenja tražimo u obliku

$$v(x) = c_1 + c_2x + c_3x^2.$$

Zamenom približnog rešenja u integralnu jednačinu dobijamo funkciju greške

$$R(x; c_1, c_2, c_3) = \frac{1}{13}(c_1 - 6c_2 - 4c_3) + c_2 x + (c_3 - 1)x^2.$$

Minimizacijom $\mathcal{L}_2$ norme funkcije greške (videti prethodni zadatak) dobijamo sistem linearnih jednačina po c_i, čije je rešenje $c_1 = 4$, $c_2 = 0$, $c_3 = 1$. Kako je $R(x; 4, 0, 1) \equiv 0$ za svako x, to je ovim određeno tačno rešenje integralne jednačine

$$u(x) = 4 + x^2.$$

Napomena: S obzirom da je integral u jednačini jednak konstanti, rešenje je očigledno oblika $u(x) = x^2 + c$, pa smo mogli odmah da zaključimo da je $c_2 = 0$ i $c_3 = 1$.

10.27 *Metodom najmanjih kvadrata naći aproksimaciju polinomom drugog stepena rešenja integralne jednačine*

$$u(x) = \int_0^1 \frac{x\,t}{1+t^2} u(t)\,dt + 1 - x^2.$$

Rešenje: Aproksimacija se traži u obliku $v(x) = c_0 + c_1 x + c_2 x^2$. Funkcija greške je

$$R(x; c_0, c_1, c_2) = c_0 + \left(-\frac{\ln 2}{2}\,c_0 + \frac{\pi}{4}\,c_1 + \frac{\ln 2 - 1}{2}\,c_2\right) x + c_2 x^2.$$

Minimimum $\mathcal{L}_2$ norme funkcije greške se postiže za $c_0 = 0.99972$, $c_1 = 0.24763$ i $c_2 = -1.00170$. Stoga je traženo približno rešenje

$$v(x) = 1.00 + 0.25x - 1.00x^2.$$

10.28 *Odrediti približno rešenje u obliku polinoma drugog stepena integralne jednačine*

$$u(x) - \frac{1}{2}\int_0^1 x u(t)\,dt = x^2$$

a) metodom Galerkina,
b) metodom najmanjih kvadrata,
c) metodom kolokacije, ako su tačke kolokacije $x_0 = 0$, $x_1 = 0.5$, $x_2 = 1$.
Računati sa pet decimala.

Rešenje: Aproksimacija je oblika $v(x) = c_1 + c_2 x + c_3 x^2$, tako da zamenom u integralnu jednačinu daje funkciju greške

$$R(x; c_1, c_2, c_3) = (1 - \frac{x}{2})c_1 + \frac{3x}{4}c_2 + (x^2 - \frac{x}{6})c_3 - x^2.$$

a) Uslov ortogonalnosti funkcije greške na bazisne funkcije

$$\int_0^1 R(x; c_1, c_2, c_3) x^k dx = 0, \qquad k = 0, 1, 2,$$

daje sistem linearnih jednačina po koeficijentima c_i, $i = 1, 2, 3$, čijim rešavanjem se dobija aproksimacija

$$v(x) = x^2 + \frac{2}{9}x.$$

b) Koeficijenti c_i, $i = 1, 2, 3$, se računaju minimizacijom integrala

$$I(c_1, c_2, c_3) = \int_0^1 R(x; c_1, c_2, c_3)^2 \, dx = \min \qquad \text{tj.} \qquad \frac{\partial I}{\partial c_k} = 0, \quad k = 1, 2, 3.$$

Rešenje dobijenog sistema linearnih jednačina određuje aproksimaciju

$$v(x) = 0.99636x^2 + 0.22605x - 0.00070.$$

c) Interpolacijom funkcije greške sa tačkama kolokacije kao čvorovima interpolacije,

$$R(x_k; c_1, c_2, c_3) = 0, \quad k = 0, 1, 2,$$

dobijamo sistem linearnih jednačina koji određuje aproksimaciju

$$v(x) = x^2 + \frac{2}{9}x.$$

Ovo je i tačno rešenje zadatka.

11

Parcijalne diferencijalne jednačine

Metodom mreže rešavaju se u oblasti G sa granicom ∂G linearne parcijalne diferencijalne jenačine oblika

$$L\,u \equiv p_x \frac{\partial^2 u}{\partial x^2} + p_y \frac{\partial^2 u}{\partial y^2} + q_x \frac{\partial u}{\partial x} + q_y \frac{\partial u}{\partial y} + ru = f,$$

gde su p_x, p_y, q_x, q_y, r i f zadate funkcije promenljivih x i y.

Mreža u oblasti $G \subseteq [a,b] \times [c,d]$ definiše se familijama paralelnih pravih

$$\omega = \{(x_i, y_j) \,|\, x_i = a + ih_x,\ y_j = c + jh_y,\ i = 0, \dots n, j = 0, \dots m,\\ h_x = \frac{b-a}{n},\ h_y = \frac{d-c}{m}\}$$

Mreža je kvadratna ako je $h_x = h_y$. Kada promenljiva y označava vreme, biće označena sa t, a korak h_y sa τ.

Aproksimacije izvoda pomoću približnih vrednosti rešenja u čvorovima mreže $v_{i,j} \approx u(x_i, y_j)$ sa navedenim redom greške (data je aproksimacija parcijalnih izvoda rešenja po promenljivoj x, analogno se definiše po ostalim promenljivim):

Prvi izvod $\frac{\partial u}{\partial x}(x_i, y_j)$

$$v_{x,ij} = \frac{1}{h_x}\big(v_{i+1,j} - v_{i,j}\big) \quad \text{ili} \quad v_{\overline{x},ij} = \frac{1}{h_x}\big(v_{i,j} - v_{i-1,j}\big) \quad O(h_x)$$

$$v_{\dot{x},ij} = \frac{1}{2h_x}\big(v_{i+1,j} - v_{i-1,j}\big) = \frac{1}{2}(v_{\overline{x},ij} + v_{x,ij}) \qquad O(h_x^2)$$

Drugi izvod $\frac{\partial^2 u}{\partial x^2}(x_i, y_j)$

$$v_{\overline{x}x,ij} = \frac{1}{h_x^2}\big(v_{i+1,j} - 2v_{i,j} + v_{i-1,j}\big) \qquad O(h_x^2)$$

11.1 *Oceniti grešku aproksimacije diferencijalnog operatora*

$$L(u) \equiv \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x}, \qquad a = const,$$

diferencijskim operatorom

$$\Lambda(v) \equiv \frac{1}{\tau}\big(v_i^{j+1} - \frac{1}{2}(v_{i+1}^j + v_{i-1}^j)\big) + \frac{a}{2h}(v_{i+1}^j - v_{i-1}^j).$$

Rešenje: Greška aproksimacije je $O(h^2 + \tau)$ jer je

$$\begin{aligned} L(u) - \Lambda(u) &= \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - \frac{1}{\tau}\big(u(x, t+\tau) - \frac{1}{2}(u(x+h,\tau) + u(x-h,\tau))\big) \\ &\qquad - \frac{a}{2h}(u(x+h,\tau) - u(x-h,\tau)) \\ &= \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - \frac{1}{\tau}\big(\tau\frac{\partial u}{\partial t} + O(\tau^2) - \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2} + O(h^4)\big) - a\frac{\partial u}{\partial x} + O(h^2) \\ &= O(h^2 + \tau) \end{aligned}$$

11.1 Jednačina eliptičkog tipa

Za eliptičku jednačinu definisanu Laplaceovim operatorom (Poissonova jednačina)

$$\Delta u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$$

"krst" šema tačnosti $O(h_x^2 + h_y^2)$ je

$$\Delta_h v_{i,j} \equiv v_{\overline{x}x,ij} + v_{\overline{y}y,ij} = f_{i,j}$$

11.2 *Dat je granični problem:*

$$\begin{aligned} \Delta u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= f(x,y), \qquad (x,y) \in G \\ u(x,y) &= g(x,y), \qquad (x,y) \in \partial G \end{aligned}$$

Neka je:

$$(x,y) \in G \iff (y,x) \in G$$
$$f(x,y) = f(y,x), \qquad (x,y) \in G$$
$$g(x,y) = g(y,x), \qquad (x,y) \in \partial G$$

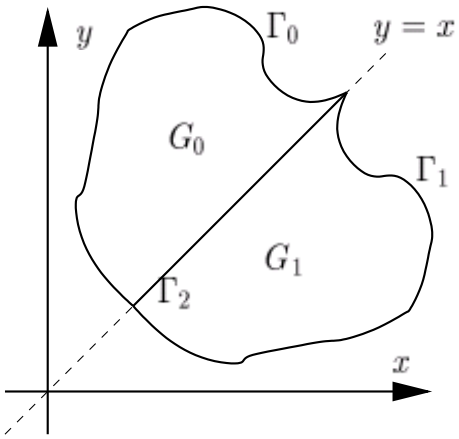
Dokazati da je tada i $u(x,y) = u(y,x)$ *(simetrija u odnosu na pravu* $y = x$*).*

Rešenje: Ako se oblast G podeli na podoblasti G_0 i G_1 sa granicama $\Gamma_0 + \Gamma_2$, odnosno $\Gamma_1 + \Gamma_2$ kao na slici 11.1, problem se može razbiti na dva potproblema:

$$(1) \qquad \begin{array}{ll} \Delta u_0(x,y) = f(x,y) & (x,y) \in G_0 \\ u_0(x,y) = g(x,y) & (x,y) \in \Gamma_0 \\ u_0(x,y) = u(x,x) & (x,y) \in \Gamma_2 \end{array}$$

i:

$$(2) \qquad \begin{array}{ll} \Delta u_1(x,y) = f(x,y) & (x,y) \in G_1 \\ u_1(x,y) = g(x,y) & (x,y) \in \Gamma_1 \\ u_1(x,y) = u(x,x) & (x,y) \in \Gamma_2 \end{array}$$



Slika 11.1: Podela oblasti G na podoblasti.

S obzirom na pretpostavke zadatka, prvi potproblem je ekvivalentan sa:

$$\begin{array}{ll} \Delta u_0(x,y) = f(y,x) & (y,x) \in G_1 \\ u_0(x,y) = g(y,x) & (y,x) \in \Gamma_1 \\ u_0(x,y) = u(x,x) & (y,x) \in \Gamma_2 \end{array}$$

Laplace-ov operator Δ je invarijantan na smenu $y = x$, $x = y$, pa je na osnovu prethodnog

$$\begin{array}{ll} \Delta u_0(y,x) = f(x,y) & (x,y) \in G_1 \\ u_0(y,x) = g(x,y) & (x,y) \in \Gamma_1 \\ u_0(y,x) = u(x,x) & (x,y) \in \Gamma_2 \end{array}$$

Kako problem (2) ima jedinstveno rešenje, sledi da je

$$u_1(x,y) = u_0(y,x) \qquad (x,y) \in G_1 \cup \Gamma_1 \cup \Gamma_2 = \overline{G}_1$$

odakle je

$$u(x,y) = u_1(x,y) = u_0(y,x) = u(y,x) \quad (x,y) \in \overline{G}_1$$

Analogno se izvodi dokaz za oblast $\overline{G}_0$, pa je onda;

$$u(x,y) = u(y,x) \quad (x,y) \in G \cup \partial G$$

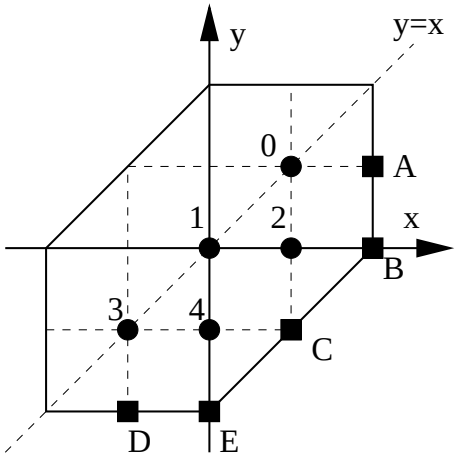
Analogno se može dokazati simetrija rešenja u odnosu na pravu $y = -x$, ili u odnosu na koordinatne ose, ukoliko su parametri problema simetrični u odnosu na odgovarajuću pravu.

11.3 *Metodom mreže rešiti granični problem*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = x + y, \qquad (x,y) \in G$$
$$u(x,y) = xy, \qquad (x,y) \in \partial G$$

gde je $G = \{(x,y) \big| \, |x| \le 1, |y| \le 1, |x-y| \le 1\}$. *Uzeti da je korak* $h = 0.5$ *i računati sa 5 decimala.*

Rešenje: Problem je simetričan u odnosu na pravu $y = x$, pa je prema prethodnom zadatku $u(x,y) = u(y,x)$, tj. $v_{ij} = v_{ji}$, i dovoljno ga je rešavati u polovini oblasti. Radi jednostavnijeg zapisa, koristićemo oznake čvorova koje su date na slici 11.2.



Slika 11.2: Oblast i mreža u zadatku 3.

U graničnim čvorovima vrednosti su određene graničnim uslovom

$$v_A = \frac{1}{2}, \quad v_B = 0, \quad v_C = -\frac{1}{4}, \quad v_D = \frac{1}{2}, \quad v_E = 0.$$

U unutrašnjim čvorovima "krst" šema se svodi na sistem jednačina

$$\begin{aligned}
(v_2 - 2v_0 + v_A) + (v_A - 2v_0 + v_2) &= 1 \cdot h^2 \\
(v_4 - 2v_1 + v_3) + (v_2 - 2v_1 + v_4) &= 0 \cdot h^2 \\
(v_1 - 2v_2 + v_B) + (v_0 - 2v_2 + v_C) &= \tfrac{1}{2} \cdot h^2 \\
(v_D - 2v_3 + v_4) + (v_4 - 2v_3 + v_D) &= -1 \cdot h^2 \\
(v_3 - 2v_4 + v_C) + (v_1 - 2v_4 + v_E) &= -\tfrac{1}{2} \cdot h^2
\end{aligned}$$

čije rešenje je

$$v_0 = 0.16072, \quad v_1 = 0.00000, \quad v_2 = -0.05357, \quad v_3 = 0.33928, \quad v_4 = 0.05357.$$

11.4 *Diferencijskom šemom tačnosti $O(h^2)$ sa korakom $h = 0.25$ rešiti granični problem*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - u = -1, \qquad (x, y) \in G,$$

$$u = 2 \cos x \cos y, \quad (x, y) \in \partial G,$$

gde je $G = \{(x, y) \,|\, |x| < 1,\ |x| - 1 < y < 1 - |x|, \}$. *Računati sa pet decimala.*

Rešenje: Problem je simetričan u odnosu na koordinatne ose i prave $y = x$ i $y = -x$ te se može rešavati samo u osmini kvadratne oblasti. Ako je u koordinatnom početku čvor $(0, 0)$, onda "krst" šemu pišemo u unutrašnjim čvorovima $(0, 0)$, (0.1), $(0, 2)$, $(0, 3)$, $(1, 1)$, i $(2, 1)$, a u graničnim čvorovima $(0, 4)$, $(1, 3)$ i $(2, 2)$ vrednosti rešenja su zadate graničnim uslovom. Rešenje zadatka u prvom kvadrantu je

1.00	1.08060				
0.75	1.32045	1.41788			
0.50	1.38544	1.42960	1.54030		
0.25	1.38621	1.40163	1.42960	1.41788	
0.00	1.38026	1.38621	1.38544	1.32045	1.08060
$y_j \setminus x_i$	0.00	0.25	0.50	0.75	1.00

Dobijene vrednosti rešenja treba simetrično preslikati u ostatak oblasti saglasno uočenoj simetriji.

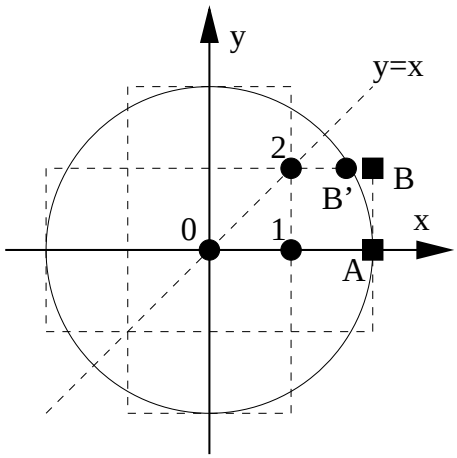
11.5 *Metodom mreže rešiti granični problem:*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = x^2 + y^2, \qquad (x, y) \in G$$

$$u(x, y) = |x| + |y|, \qquad (x, y) \in \partial G$$

gde je $G = \{(x,y) \big| \, x^2+y^2 < 1\}$. *Uzeti da je korak* $h = 0.5$ *i računati sa 5 decimala.*

Rešenje: Problem je simetričan u odnosu na prave $x = 0$, $y = 0$ i $y = x$ pa ga je dovoljno rešavati samo u osmini kruga. Za čvorove mreže koristićemo oznake kao na slici 11.3.



Slika 11.3: Oblast i mreža u zadatku 4.

"Krst" šema u unutrašnjim čvorovima se svodi na sistem jednačina

$$\begin{aligned} v_1 - 4v_0 &= h^2 \cdot 0 \\ (v_0 - 2v_1 + v_A) + (v_2 - 2v_1 + v_2) &= h^2 \cdot 0.5^2 \\ (v_1 - 2v_2 + v_B) + (v_B - 2v_2 + v_1) &= h^2 \cdot (0.5^2 + 0.5^2) \end{aligned}$$

Granični čvor A pripada granici oblasti, pa je $v_A = 1$. Vrednost u čvoru B će biti ekstrapolirana pomoću vrednosti v_2 i $v_{B'}$ na sledeći način:

$$v_B = -\frac{\delta}{h-\delta} v_2 + \frac{h}{h-\delta} v_{B'},$$

gde je sa δ označeno rastojanje izmedju tačaka B' i B. Tačka B' je presek kruga i prave $y = \frac{1}{2}$, pa je $\delta = 1 - \frac{\sqrt{3}}{2}$ i $v_B = -0.36603 v_2 + 1.86603$. Rešavanjem prethodnog sistema linearnih jednačina dopunjenog poslednjim vezama, dobijamo da je približno rešenje u čvorovima mreže

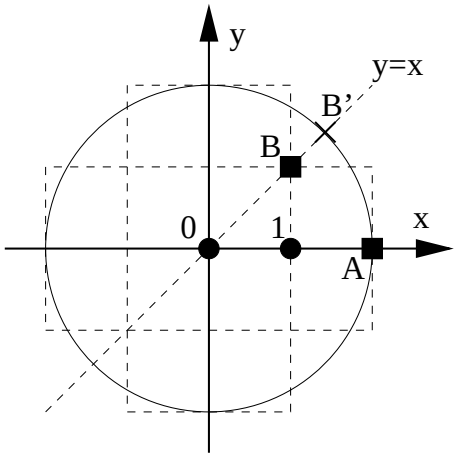
$$v_0 = v_1 = 1.14263, \quad v_2 = 1.24519, \quad v_A = 1, \quad v_B = 1.41025.$$

11.6 *Metodom mreže rešiti granični problem:*

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= e^{(xy)^2}, \qquad (x,y) \in G \\ u(x,y) &= \cos(xy), \qquad (x,y) \in \partial G \end{aligned}$$

gde je $G = \{(x,y) \big| \, x^2+y^2 < 1\}$. *Uzeti da je korak* $h = 0.5$ *i računati sa 5 decimala.*

Rešenje: Kao i u prethodnom zadatku, zbog simetrije je dovoljno tražiti rešenje samo u osmini zadate oblasti. Neka oznake čvorova mreže budu kao na slici 11.4.



Slika 11.4: Oblast i mreža u zadatku 5.

Ilustrujmo u ovom primeru drugi način aproksimacije vrednosti rešenja u graničnom čvoru koji ne pripada granici oblasti. Neka granični čvor bude onaj čiji bar jedan kraj "krsta" ne pripada oblasti G. Ovde je to čvor B. Uzmimo da je

$$v_B = v_{B'} = \cos(\frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}}) = 0.87758,$$

pošto je B' tačka na kružnici koja je najbliža tački B. Čvor A pripada granici, pa je $v_A = \cos(0) = 1$. U preostala dva čvora (unutrašnji) "krst" šema je predstavljena jednačinama

$$16v_1 - 16v_0 = 1, \qquad 4v_0 - 16v_1 + 4v_A + 8v_B = 1$$

Rešenja dobijenog sistema linearnih jednačina su

$$v_0 = 0.75172, \quad v_1 = 0.81422, \quad v_A = 1, \quad v_B = 0.87758.$$

11.7 *Metodom mreže sa korakom* $h = 0.5$ *rešiti granični zadatak*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = (x-3)^2 + (y-3)^2, \qquad (x,y) \in G,$$

$$u = 1, \qquad (x,y) \in \partial G,$$

gde je $G = \{(x,y) \,|\, (x-3)^2 + (y-3)^2 < 1\}$. *Sve čvorove unutar oblasti* G *tretirati kao unutrašnje, a u spoljašnjim čvorovima rešenje aproksimirati interpolacijom. Računati na pet decimala.*

Rešenje: Pošto je Laplaceov operator invarijantan u odnosu na smenu $\xi = x - 3$, $\eta = y - 3$, zadatak ćemo zbog jednostavnosti rešavati u $\xi\eta$ koordinatnom sistemu

$$\frac{\partial^2 u}{\partial \xi^2} + \frac{\partial^2 u}{\partial \eta^2} = \xi^2 + \eta^2, \qquad (\xi, \eta) \in \overline{G},$$
$$u = 1, \qquad (\xi, \eta) \in \partial\overline{G},$$

gde je $\overline{G} = \{(\xi, \eta) \,|\, \xi^2 + \eta^2 < 1\}$. Zadatak je simetričan u odnosu na prave $\xi = 0$, $\eta = 0$ i $\eta = \xi$, te ćemo ga rešavati u jednoj osmini kruga $\overline{G}$. Približnu vrednost rešenja u graničnom čvoru $(2, 1)$ (centar kruga je čvor $(0, 0)$) nalazimo linearnom interpolacijom pomoću vrednosti u čvoru $(1, 1)$ i tački na granici $A(\sqrt{3}/2, 1/2)$

$$\frac{1}{h}(v_{2,1} - v_{1,1}) = \frac{1}{\delta}(v_A - v_{1,1}), \qquad \delta = \sqrt{3}/2 - 1/2$$

Tačka A pripada granici pa je $v_A = 1$. Stoga je

$$v_{2,1} = -0.36603 v_{1,1} + 1.36603.$$

Granični čvor $(2, 0)$ pripada granici, pa je $v_{20} = 1$. U unutrašnjim tačkama mreže jednačinu aproksimiramo "krst" šemom

$$\begin{aligned} 4v_{1,0} - 4v_{0,0} &= 0, \\ v_{0,0} + v_{2,0} + 2v_{1,1} - 4v_{1,0} &= 0.0625, \\ 2v_{1,0} + 2v_{2,1} - 4v_{1,1} &= 0.125. \end{aligned}$$

Rešenje sistema linearnih jednačina je približno rešenje zadatka

4.0		1.01795	1	1.01795	
3.5	1.01795	0.95097	0.94648	0.95097	1.01795
3.0	1	0.94648	0.94648	0.94648	1
2.5	1.01795	0.95097	0.94648	0.95097	1.01795
2.0		1.01795	1	1.01795	
$y_j \setminus x_i$	2.0	2.5	3.0	3.5	4.0

11.8 *Rešiti granični problem*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad (x, y) \in G,$$
$$u = |x| + |y|, \qquad (x, y) \in \partial G,$$

gde je $G = \{(x, y) | (|x| + 2)(|y| + 2) < 12\}$, *"krst" šemom sa korakom* $h = 1$. *Za aproksimaciju vrednosti rešenja u graničnim čvorovima koristiti linearnu interpolaciju. Računati sa pet decimala.*

Rešenje: Oblast G je ograničena hiperbolama i simetrična je u odnosu na koordinatne ose i pravu $y = x$. Kako je i funkcija kojom je zadat početni uslov takođe simetrična u odnosu na ove prave, zadatak se može rešavati u jednoj osmini oblasti i rešenje simetrično preslikati u ostatak oblasti.

Ako centar oblasti (koordinatni početak) označimo kao čvor $(0,0)$, unutrašnji čvorovi u kojima se za aproksimaciju jednačine koristi "krst" šema su $(i,0)$, $i = 0,1,2,3$ i $(1,1)$

$$\begin{aligned} 4v_{1,0} - 4v_{0,0} &= 0 \\ v_{0,0} + v_{2,0} + 2v_{1,1} - 4v_{1,0} &= 0 \\ v_{1,0} + v_{3,0} + 2v_{2,1} - 4v_{2,0} &= 0 \\ v_{2,0} + v_{4,0} + 2v_{3,1} - 4v_{3,0} &= 0 \\ 2v_{1,0} + 2v_{2,1} - 4v_{1,1} &= 0 \end{aligned}$$

Granični čvorovi $(4,0)$ i $(2,1)$ pripadaju granici te je u njima vrednost rešenja zadata tačno. U graničnom čvoru $(3,1)$ rešenje se aproksimira linearnom interpolacijom pomoću vrednosti u čvoru $(3,0)$ i tački na granici $A' = (3, 0.4)$,

$$v_A + 1.5v_{3,0} = 8.5.$$

Rešavanjem sistema linearnih jednačina dobijaju se približne vrednosti rešenja zadatka. U tabeli koja sledi prikazana je četvrtina oblasti, zbog simetrije zadatka,

4	4				
3	3.44681				
2	3.12766	3			
1	3.06383	3.03191	3		
0	3.06383	3.06383	3.12766	3.44681	4
$y_j \setminus x_i$	0	1	2	3	4

11.9 *Rešiti granični problem*

$$2\frac{\partial^2 u}{\partial x^2} + 2\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad (x,y) \in G,$$

$$u = 1, \qquad (x,y) \in \partial G,$$

gde je $G = \{(x,y) | \, \frac{5}{4}x^2 - xy + y^2 < 1\}$, *"krst" šemom sa korakom* $h = 0.5$. *Za vrednosti rešenja u graničnim čvorovima uzeti vrednosti rešenja u najbližim tačkama na granici. Računati sa pet decimala.*

Rešenje: Smenom $\xi = x$ i $\eta = x - 2y$ dati problem se svodi na granični problem

$$\frac{\partial^2 u}{\partial \xi^2} + \frac{\partial^2 u}{\partial \eta^2} = 0, \quad (\xi,\eta) \in \overline{G}, \qquad\qquad u = 1, \quad (\xi,\eta) \in \partial\overline{G},$$

definisan u elipsi $\overline{G} = \{(\xi,\eta) | \, \left(\frac{\xi}{2}\right)^2 + \eta^2 < 1\}$ sa granicom $\partial\overline{G}$. Simetrija oblasti i parametara problema u odnosu na ose $\xi = 0$ i $\eta = 0$ omogućava rešavanje problema

samo u prvom kvadrantu. Ako je centar elipse čvor $(0,0)$, onda su granični čvorovi $(4,0)$, $(3,1)$, $(2,2)$, $(1,2)$ i $(0,2)$ i vrednost rešenja u njima je prema postavci zadatka jednaka 1. Diskretizacija jednačine u unutrašnjim čvorovima "krst" šemom daje sledeći sistem linearnih jednačina

$$\begin{aligned}
2v_{1,0} + 2v_{0,1} - 4v_{0,0} &= 0\\
v_{0,0} + v_{2,0} + 2v_{1,1} - 4v_{1,0} &= 0\\
v_{1,0} + v_{3,0} + 2v_{2,1} - 4v_{2,0} &= 0\\
v_{2,0} + v_{4,0} + 2v_{3,1} - 4v_{3,0} &= 0\\
2v_{1,1} + v_{0,2} + v_{0,0} - 4v_{0,1} &= 0\\
v_{0,1} + v_{2,1} + v_{1,2} + v_{1,0} - 4v_{1,1} &= 0\\
v_{1,1} + v_{3,1} + v_{2,2} + v_{2,0} - 4v_{2,1} &= 0
\end{aligned}$$

čije rešenje je

$$v_{0,0} = v_{1,0} = v_{2,0} = v_{3,0} = v_{4,0} = v_{0,1} = v_{1,1} = v_{2,1} = v_{3,1} = v_{0,2} = v_{1,2} = v_{2,2} = 1.$$

Dakle, rešenje zadatka je identički jednako jedan u celoj oblasti.

11.10 *Rešiti granični problem u jediničnom kvadratu* $(0,1)\times(0,1)$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -1$$

$$u(1,y) = u(x,0) = u(x,1) = 1, \qquad \left(\frac{\partial u}{\partial x} + u\right)_{/(0,y)} = 0.$$

diferencijskom šemom tačnosti $O(h_x^2 + h_y^2)$, *ako je* $h_x = 1/2$ *i* $h_y = 1/3$. *Računati sa četiri decimale.*

Rešenje: Diferencijska šema koja aproksimira datu jednačinu sa tačnošću $O(h_x^2 + h_y^2)$ u unutrašnjim tačkama mreže je "krst" šema

$$v_{\overline{x}x,ij} + v_{\overline{y}y,ij} = -1.$$

Konstruišemo aproksimaciju istog reda tačnosti graničnog uslova zadatog na pravoj $x = 0$. Kako je

$$u_x(0,y) = \frac{1}{h_x}\big(u(h_x,y) - u(0,y)\big) = \left(\frac{\partial u}{\partial x} + \frac{h_x}{2}\frac{\partial^2 u}{\partial x^2}\right)_{/(0,y)} + O(h_x^2),$$

a iz graničnog uslova i jednačine je

$$\frac{\partial u}{\partial x}(0,y) = -u(0,y), \qquad \frac{\partial^2 u}{\partial x^2}(0,y) = -1 - \frac{\partial^2 u}{\partial y^2}(0,y),$$

to je

$$u_x(0,y) = -u(0,y) + \frac{h_x}{2}\big(-1 - \frac{\partial^2 u}{\partial y^2}(0,y)\big) + O(h_x^2).$$

Konačno, korišćenjem aproksimacije

$$\frac{\partial^2 u}{\partial y^2}(0,y) = u_{\overline{y}y}(0,y) + O(h_y^2),$$

dobijamo da je

$$u_x(0,y) = -u(0,y) + \frac{h_x}{2}\big(-1 - u_{\overline{y}y}(0,y) + O(h_y^2)\big) + O(h_x^2)$$

Tražena aproksimacija tačnosti $O(h^2)$ u čvorovima mreže na granici $x = 0$ je

$$v_{0,j} + v_x 0, j + \frac{h_x}{2} v_{\overline{y}y,0,j} = -\frac{h_x}{2}.$$

U čvorovima mreže na ostalom delu granice vrednosti rešenja su zadate tačno.

Za zadate vrednosti koraka h_x i h_y aproksimacija jednačine u unutrašnjim tačkama je

$$26v_{i,j} - 4(v_{i+1,j} + v_{i-1,j}) - 9(v_{i,j+1} + v_{i,j-1}) = 1, \quad i = 1, \quad j = 1, 2,$$

na levoj granici

$$22v_{0,j} - 8v_{1,j} - 9(v_{0,j+1} + v_{0,j-1}) = 1, \quad j = 1, 2,$$

i u ostalim čvorovima na granici

$$v_{0,0} = v_{1,0} = v_{2,0} = v_{0,3} = v_{1,3} = v_{2,3} = v_{2,1} = v_{2,2} = 1.$$

Šema se, stoga, svodi na sistem linearnih jednačina

$$\begin{aligned} 22v_{0,1} - 8v_{1,1} - 9v_{0,2} &= 10 \\ -4v_{0,1} + 26v_{1,1} - 9v_{1,2} &= 14 \\ -9v_{0,1} + 22v_{0,2} - 8v_{1,2} &= 10 \\ -9v_{1,1} - 4v_{0,2} + 26v_{1,2} &= 14 \end{aligned}$$

čije rešenje predstavlja tražene približne vrednosti rešenja u čvorovima mreže

1.00	1	1	1
0.67	1.49206	1.17460	1
0.33	1.49206	1.17460	1
0.00	1	1	1
$y_j \setminus x_i$	0.0	0.5	1.0

11.11 *Funkcija $u(x,y) \in \mathcal{C}^1(G)$ koja minimizira integral*

$$I = \int\int_G \left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \right) dx\, dy,$$

je rešenje graničnog problema

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad u(x,y)\big/_{\partial G} = f(x,y),$$

gde je ∂G granica oblasti G. Pokazati da se minimizacijom integrala I kada se ovaj prethodno aproksimira sumom a parcijalni izvodi u integralu aproksimiraju količnicima konačnih razlika unapred dobija "krst" šema za Laplaceovu jednačinu.

Rešenje: Ako koristimo aproksimacije

$$\int\int_G F(x,y)\,dx\,dy \approx h_x h_y \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} F(x_i,y_j),$$

$$\frac{\partial u}{\partial x}(x_i,y_j) \approx v_{x,i,j}, \qquad \frac{\partial u}{\partial y}(x_i,y_j) \approx v_{y,i,j},$$

gde su h_x i h_y koraci mreže u pravcu x i y ose, imamo da je

$$\begin{aligned} I \approx S &= h_x h_y \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} (v_{x,i,j}^2 + v_{y,i,j}^2) \\ &= h_x h_y \big(\cdots + \frac{1}{h_x^2}(v_{i,j} - v_{i-1,j})^2 + \frac{1}{h_x^2}(v_{i+1,j} - v_{i,j})^2 \\ &\qquad + \frac{1}{h_y^2}(v_{i,j+1} - v_{i,j})^2 + \frac{1}{h_y^2}(v_{i,j} - v_{i,j-1})^2 + \cdots \big) \end{aligned}$$

Potreban uslov minimuma sume S kao funkcije parametara $v_{i,j}$ je

$$\partial S/\partial v_{i,j} = 0, \qquad i = 1,\ldots,n-1, \quad j = 1,\ldots m-1,$$

što daje sistem linearnih jednačina

$$\frac{1}{h_x^2}(v_{i,j} - v_{i-1,j}) - \frac{1}{h_x^2}(v_{i+1,j} - v_{i,j}) - \frac{1}{h_y^2}(v_{i,j+1} - v_{i,j}) + \frac{1}{h_y^2}(v_{i,j} - v_{i,j-1}) = 0,$$
$$i = 1,\ldots,n-1, \quad j = 1,\ldots,m-1.$$

Dobijeni sistem predstavlja "krst" šemu za Laplaceovu jednačinu

$$v_{\overline{x}x,i,j} + v_{\overline{y}y,i,j} = 0, \qquad i = 1,\ldots,n-1, \quad j = 1,\ldots,m-1$$
$$v_{i,0} = f_{i0}, \quad v_{i,m} = f_{im}, \quad v_{0,j} = f_{0j}, \quad v_{n,j} = f_{nj}.$$

FORTRAN Implementacija "krst" šeme se, slično nekim prethodnim metodama, svodi na postavljanje sistema jednačina i pozivanje procedure koja rešava taj sistem. U ovom slučaju, matrica sistema je retka ali nije trodijagonalna, tako da će za rešavanje sistema biti korišćena ranije pominjana procedura `gauss()`. Koeficijenti

sistema se jednostavno računaju, treba samo napraviti razliku između jednačina za čvorove na obodu mreže i za unutrašnje čvorove. U prvom slučaju rešenja su zapravo poznata, tako da treba razmisliti i o alternativnoj implementaciji u kojoj bi se jednačine postavljale samo u unutrašnjim čvorovima mreže; takva implementacija bi bila nešto komplikovanije, ali svakako i efikasnija. Procedura koja implementira šemu ima oblik:

```
module cross_module
  use matheval_module
  use numerics_module
  implicit none
  public cross

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string

contains
  ! Funkcija cross() odredjuje resenje Poisson-ove jednacine koristeci
  ! metodu mreze sa semom krsta. Argumenti funkcije su:
  !   f - desna strana Poisson-ove jednacine
  !   x0, y0 - koordinate donjeg levog ugla mreze
  !   nx, ny - broj tacaka u mrezi po x- odn. y-osi
  !   hx, hy - korak mreze po x, odn. y-osi
  ! x u0 - funkcija koja odreduje granicne uslove, tj. vrednosti
  !        resenja jednacine na obodima mreze
  !   v - matrica u koju treba smestiti resenja jednacine na mrezi
  ! Pretpostavlja se da su sva polja alocirana izvan funkcije. Broj
  ! cvorova duz svake od osa mora biti veci ili jednak 3. Korak duz
  ! svake od osa mora biti veci od 0. Desna strana jednacine i pocetni
  ! uslov moraju biti sintaksno ispravno zadati.
  subroutine cross (f, x0, y0, nx, ny, hx, hy, u0, v)
    type(string), intent(in) :: f
    real(kind=8), intent(in) :: x0, y0
    integer, intent(in) :: nx, ny
    real(kind=8), intent(in) :: hx, hy
    type(string), intent(in) :: u0
    real(kind=8), dimension(0:nx-1, 0:ny-1), intent(out) :: v
    integer(kind=8) :: f_evaluator ! Evaluator za desnu stranu jednacine.
    integer(kind=8) :: u0_evaluator ! Evaluator za funkciju granicnog uslova.
    real(kind=8), dimension(0:nx*ny-1, 0:nx*ny-1) :: A ! Matrica sistema
        ! jednacina za odredjivanje resenja polazne jednacine na mrezi.
    real(kind=8), dimension(0:nx*ny-1) :: b ! Vektor sa desnim stranama
        ! sistema.
    real(kind=8), dimension(0:nx*ny-1) :: x ! Vektor sa resenjima sistema.
    real(kind=8) :: mul_x, mul_y ! Reciprocne vrednosti kvadrata koraka
        ! mreze duz pojedinih koordinatnih osa.
    real(kind=8), dimension(0:(nx*ny)**2-1) :: A_vector ! Vektor koji
        ! odgovara matrici sistema.
    integer :: i, j, k ! Brojaci u petljama.

    ! Proverava se validnost argumenata procedure.
    if (nx<3 .or. ny<3) stop
    if (hx<=0 .or. hy<=0) stop
    f_evaluator=evaluator_create(f%chars)
```

```
if (f_evaluator==0) stop
u0_evaluator=evaluator_create(u0%chars)
if (u0_evaluator==0) stop

! Incijalizuje se matrica sistema jednacina po vrednostima resenja
! na mrezi.
A(:,:)=0

! Postavljaju se jednacine za cvorove na donjoj ivici mreze.
k=0
j=0
do i=0, nx-1
   A(k,k)=1
   b(k)=evaluator_evaluate_x_y(u0_evaluator,x0+i*hx,y0+j*hy)
   k=k+1
end do

! Postavljaju se jednacine za cvorove izmedju donje i gornje ivice
! mreze.
mul_x=1/(hx*hx)
mul_y=1/(hy*hy)
do j=1, ny-2
   ! Postavlja se jednacina za cvor na levoj ivici mreze.
   i=0
   A(k,k)=1
   b(k)=evaluator_evaluate_x_y(u0_evaluator,x0+i*hx,y0+j*hy)
   k=k+1

   ! Postavljaju se jednacine za unutrasnje cvorove mreze.
   do i=1, nx-2
      A(k,k)=-2*(mul_x+mul_y)
      A(k,k-1)=mul_x
      A(k,k+1)=mul_x
      A(k,k-nx)=mul_y
      A(k,k+nx)=mul_y
      b(k)=evaluator_evaluate_x_y(f_evaluator,x0+i*hx,y0+j*hy)
      k=k+1
   end do

   ! Postavlja se jednacina za cvor na desnoj ivici mreze.
   i=nx-1
   A(k,k)=1
   b(k)=evaluator_evaluate_x_y(u0_evaluator,x0+i*hx,y0+j*hy)
   k=k+1
end do

! Postavljaju se jednacine za cvorove na gornjoj ivici mreze.
j=ny-1
do i=0, nx-1
   A(k,k)=1
   b(k)=evaluator_evaluate_x_y(u0_evaluator,x0+i*hx,y0+j*hy)
   k=k+1
end do

! Brisu se korisceni evaluatori.
call evaluator_destroy(f_evaluator)
call evaluator_destroy(u0_evaluator)
```

```
    ! Kopira se matrica sistema u vektor radi poziva gauss() procedure.
    A_vector=(/((A(i,j), j=0, nx*ny-1), i=0, nx*ny-1)/)

    ! Odredjuju se resenja sistema.
    call gauss(nx*ny, A_vector, b, x)

    ! Kopiraju se resenja sistema u matricu koja odgovara cvorovima
    ! mreze.
    k=0
    do i=0, nx-1
       do j=0, ny-1
          v(i,j)=x(k)
          k=k+1
       end do
    end do
  end subroutine cross
end module cross_module
```

11.2 Jednačina paraboličkog tipa

Za paraboličku jednačinu

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f$$

šema sa težinom $0 \le \sigma \le 1$ je

$$v_{t,i}^j = \sigma(v_{\overline{x}x,i}^{j+1} + f_i^{j+1}) + (1-\sigma)(v_{\overline{x}x,i}^j + f_i^j) \qquad \begin{cases} O(h_x^2 + \tau), & \sigma \neq 1/2 \\ O(h_x^2 + \tau^2), & \sigma = 1/2 \end{cases}$$

gde je τ korak po t. Za $\sigma = 1/2$ šema se naziva Crank-Nicholson-ova šema.

11.12 *Eksplicitnom dvoslojnom šemom odrediti rešenje mešovitog problema*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + 2x + t, \qquad 0 < x < 1, \quad 0 < t \le 0.02$$

$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = (1.1x^2 + 1.5)\sin(\pi x)$$

Uzeti za korake mreže $h = 0.1$ *i* $\tau = 0.01$*, i računati sa pet decimala.*

Rešenje: Interval po promenljivoj x je podeljen na $n_x = 1/h = 10$ intervala mreže, a po promenljivoj t na $n_t = 0.02/\tau = 2$ intervala mreže. Diskretizacija datog problema eksplicitnom dvoslojnom diferencijskom šemom u unutrašnjim tačkama mreže daje

$$\frac{1}{\tau}(v_i^{j+1} - v_i^j) = \frac{1}{h^2}(v_{i+1}^j - 2v_i^j + v_{i-1}^j) + 2x_i + t_j, \qquad i = 1, \ldots, 9, \quad j = 0, 1.$$

Kada se uvrste date vrednosti za h i τ i dodaju početni i granični uslovi, dobijaju se eksplicitne formule

$$v_i^0 = (1.1x_i^2 + 1.5)\sin(\pi x_i)$$
$$v_0^{j+1} = 0, \quad v_i^{j+1} = v_{i+1}^j - v_i^j + v_{i-1}^j + 0.01(2x_i + t_j), \quad v_{10}^{j+1} = 0,$$
$$i = 1, \ldots, 9, \quad j = 0, 1,$$

koje daju sledeće vrednosti približnog rešenja problema

$t_j \backslash x_i$	0.0	0.1	0.2	0.3	0.4	0.5
0.00	0	0.46692	0.90754	1.29362	1.59397	1.77500
0.01	0	0.44262	0.85700	1.21389	1.48265	1.63217
0.02	0	0.41648	0.80361	1.13186	1.37151	1.49397

0.6	0.7	0.8	0.9	1.0
1.80320	1.64959	1.29548	0.73886	0
1.63339	1.46309	1.10897	0.57462	0
1.47397	1.29337	0.94484	0.55245	0

11.13 *Rešiti prethodni zadatak,*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + 2x + t, \qquad 0 < x < 1, \quad 0 < t \leq 0.02$$
$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = (1.1x^2 + 1.5)\sin(\pi x)$$

Crank-Nicholson-ovom šemom ($\sigma = 1/2$). Uzeti za korake mreže $h = 0.1$ i $\tau = 0.01$, i računati sa pet decimala.

Rešenje: Diskretizacijom jednačine u unutrašnjem čvoru (x_i, t_j) Crank-Nicholson-ovom šemom dobija se formula

$$\frac{1}{\tau}(v_i^{j+1} - v_i^j) = \frac{1}{2}\Big(\frac{1}{h^2}(v_{i+1}^{j+1} - 2v_i^{j+1} + v_{i-1}^{j+1}) + \frac{1}{h^2}(v_{i+1}^j - 2v_i^j + v_{i-1}^j)\Big)$$
$$+ \frac{1}{2}\Big((2x_i + t_{j+1}) + (2x_i + t_j)\Big)$$

Kada se uvrste date vrednosti za h i τ i dodaju početni i granični uslovi, dobijaju se sistemi sa trodijagonalnom matricom

$$v_i^0 = (1.1x_i^2 + 1.5)\sin(\pi x_i)$$
$$-50v_{i-1}^{j+1} + 200v_i^{j+1} - 50v_{i+1}^{j+1} = 50(v_{i-1}^j + v_{i+1}^j) + 2x_i + \frac{1}{2}(t_j + t_{j+1})$$
$$i = 1, \ldots, 9, \quad j = 0, 1,$$

gde je $v_0^j = v_{10}^j = 0$ za svako j. Rešenja ovih sistema, tj. rešenja polaznog problema, predstavljena su sledećom tabelom

$t_j \backslash x_i$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.00	0	0.46692	0.90754	1.29362	1.59397	1.77500	1.80320	1.64959	1.29548	0.73886	0
0.01	0	0.44189	0.85593	1.21319	1.48320	1.63493	1.63923	1.47328	1.12713	0.61468	0
0.02	0	0.41577	0.80284	1.13219	1.37449	1.50139	1.48834	1.31945	0.94482	0.53956	0

11.14 *Crank-Nicholson-ovom šemom (šema sa težinom za $\sigma = 1/2$), naći približno rešenje mešovitog problema*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < 1, \quad 0 < t \le 0.02,$$

$$u(0,t) = u(1,t) = 0, \quad t \ge 0, \qquad u(x,0) = \begin{cases} 2x, & 0 \le x \le 1/2, \\ 2(1-x), & 1/2 \le x \le 1. \end{cases}$$

Uzeti korake mreže $h = 0.1$ i $\tau = 0.01$, i računati sa četiri decimale.

Rešenje: Kako su početni i granični uslovi simetrični u odnosu na pravu $x = 1/2$, i rešenje zadatka je simetrično u odnosu na tu pravu. Stoga ćemo problem rešavati u oblasti $[0, 0.5] \times [0, 0.02]$. Diskretizacija jednačine u unutrašnjim čvorovima mreže šemom sa težinama daje

$$\frac{1}{\tau}(v_i^{j+1} - v_i^j) = \frac{1}{2h^2}(v_{i+1}^{j+1} - 2v_i^{j+1} + v_{i-1}^{j+1} + v_{i+1}^j - 2v_i^j + v_{i-1}^j),$$
$$i = 1, \ldots, 5, \quad j = 0, 1.$$

Uzimajući u obzir zadate početne i granične uslove, diferencijska šema kojom se aproksimira dati zadatak je

$$v_i^0 = 2ih, \quad i = 1, \ldots, 5, \qquad v_0^j = 0, \quad j = 1, 2,$$

$$-v_{i-1}^{j+1} + 4v_i^{j+1} - v_{i+1}^{j+1} = v_{i+1}^j + v_{i-1}^j, \qquad i = 1, \ldots, 5, \quad j = 0, 1.$$

Pri tome treba imati u vidu da je zbog simetrije zadatka $v_6^j = v_4^j$, $j = 0, 1, 2$. Rešavanjem na svakom vremenskom koraku (za svako j) odgovarajućeg trodijagonalnog sistema, dobijamo sledeće vrednosti rešenja

$t_j \setminus x_i$	0.0	0.1	0.2	0.3	0.4	0.5
0.00	0.0	0.2	0.4	0.6	0.8	1.0
0.01	0	0.1989	0.3956	0.5834	0.7381	0.7691
0.02	0	0.1936	0.3789	0.5400	0.6461	0.6921

Rešenje treba simetrično preslikati u ostatak oblasti.

11.15 *Implicitnom šemom odrediti približno rešenje mešovitog problema*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial t^2} + \sin x, \qquad 0 < x < 2, \quad 0 < t \le 0.2,$$

$$u(0,t) = \frac{t}{4} \quad u(2,t) = 0, \qquad u(x,0) = 0.$$

Uzeti korake mreže $h = 0.25$ *i* $\tau = 0.1$. *Računati sa četiri decimale.*

Rešenje: Implicitna šema je šema sa težinama za vrednost težinskog parametra $\sigma = 1$,

$$\frac{1}{\tau}(v_i^{j+1} - v_i^j) = \frac{1}{h^2}(v_{i+1}^{j+1} - 2v_i^{j+1} + v_{i-1}^{j+1}) + \sin x_i, \qquad i = 1, \ldots, 7, \quad j = 0, 1.$$

Uzimajući u obzir zadate početne i granične uslove, diferencijska šema kojom se aproksimira dati zadatak je

$$v_i^0 = 0, \quad i = 0, \ldots, 8, \qquad v_0^j = j/40, \quad v_8^j = 0, \quad j = 1, 2,$$

$$-v_{i-1}^{j+1} + 2.625 v_i^{j+1} - v_{i+1}^{j+1} = 0.0625(10 v_i^j + \sin x_i), \qquad i = 1, \ldots, 7, \quad j = 0, 1.$$

Rešavanjem za $j = 0$ i $j = 1$ odgovarajućih trodijagonalnih sistema dobijamo rešenje

$t_j \setminus x_i$	0.0	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2
0.1	0.0250	0.0338	0.0482	0.0627	0.0739	0.0787	0.0733	0.0513	0
0.2	0.0500	0.0691	0.0948	0.1196	0.1374	0.1422	0.1274	0.0842	0

Napomena: Algoritam za rešavanje trodijagonalnog sistema linearnih jednačina dat je u prvom poglavlju.

11.16 a) *Data je parcijalna diferencijalna jednačina*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2},$$

i diferencijska šema

$$\frac{1}{\tau}(v_i^{j+1} - v_i^j) = \frac{\sigma}{h^2}\big(v_{i+1}^{j+1} - 2v_i^{j+1} + v_{i-1}^{j+1}\big) + \frac{1-\sigma}{h^2}\big(v_{i+1}^j - 2v_i^j + v_{i-1}^j\big),$$

Odrediti parametar σ *tako da red aproksimacije bude četiri po* h *i dva po* τ.
b) *Koristeći šemu dobijenu u tački* a), *odrediti približno rešenje mešovitog problema*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < 3, \quad 0 < t \le 0.3,$$

$$u(x,0) = \sin(\pi x/3), \qquad u(0,t) = t, \quad u(3,t) = 0.$$

za $t = 0.3$. *Uzeti da je* $h = 0.6$, *a* τ *izabrati tako da korišćena šema bude eksplicitna. Računati sa pet decimala.*

Rešenje: Korišćenjem Taylorovog razvoja funkcije $u(x,t)$ u okolini tačke $(x, t+\tau/2)$, pod pretpostavkom da je funkcija u dovoljno glatka, dobijamo da je greška aproksimacije

$$\begin{aligned} R(u) \equiv & \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{1}{\tau}\big(u(x,t+\tau) - u(x,t)\big) \\ & + \frac{\sigma}{h^2}\big(u(x+h,t+\tau) - 2u(x,t+\tau) + u(x-h,t+\tau)\big) \\ & + \frac{1-\sigma}{h^2}\big(u(x+h,t) - 2u(x,t) + u(x-h,t)\big) \\ = & \left(\frac{h^2}{12} - (1-2\sigma)\frac{\tau}{2}\right)\frac{\partial^4 u}{\partial x^4}(x, t+\frac{\tau}{2}) + O(h^4+\tau^2) \end{aligned}$$

jer je iz jednačine

$$\frac{\partial^3 u}{\partial x^2 \partial t} = \frac{\partial^2}{\partial x^2}\left(\frac{\partial u}{\partial t}\right) = \frac{\partial^2}{\partial x^2}\left(\frac{\partial^2 u}{\partial x^2}\right) = \frac{\partial^4 u}{\partial x^4}.$$

Šema će biti traženog reda tačnosti ako je

$$\sigma = \frac{1}{2}\big(1 - \frac{h^2}{6\tau}\big).$$

b) Šema je eksplicitna ako je $\sigma = 0$ a to je za $\tau = h^2/6$. Za zadatu vrednost koraka h eksplicitna diferencijska šema povišene tačnosti je

$$v_i^{j+1} = \frac{1}{6}(v_{i+1}^j + v_{i-1}^j) + \frac{2}{3}v_i^j.$$

Približne vrednosti rešenja datog mešovitog problema određene primenom ove šeme date su sledećom tabelom

$t_j \setminus x_i$	0.0	0.6	1.2	1.8	2.4	3.0
0.00	0	0.58779	0.95106	0.95106	0.58779	0
0.06	0.06	0.55037	0.89052	0.89052	0.55037	0
0.12	0.12	0.52533	0.83383	0.83383	0.51533	0
0.18	0.18	0.50919	0.78241	0.78075	0.48253	0
0.24	0.24	0.49986	0.73660	0.73132	0.45181	0
0.30	0.30	0.49601	0.69626	0.68562	0.42309	0

FORTRAN Implementacija Crank-Nicholson-ove šeme se sastoji u uzastopnom rešavanju sistema trodijagonalnih jednačina po slojevima šeme po t osi. U svakom koraku postupka se postavlja i rešava sistem u jednom sloju šeme, čime postaje moguće postavljanje sistema u narednom sloju. Treba uočiti kako se oni koeficijenti sistema koji su zajednički za svaki sloj računaju samo jednom, čime se dobija na efikasnosti. Implementacija šeme ima sledeći oblik:

```
module crank_nicholson_module
  use matheval_module
  use tridiag_module
  implicit none
  public crank_nicholson

  ! Deklaracija tipa za stringove fiksne duzine.
  type string
     character(len=256) :: chars
  end type string

contains
  ! Procedura crank_nicholson() odredjuje resenja parabolicke parcijalne
  ! diferencijalne jednacine po promenljivim t i x. Parametri procedure
  ! su:
  !   f - funkcija koja figurise na desnoj strani jednacine
  !   x0, t0 - koordinate pocetne tacke mreze
  !   h, tau - korak mreze duz x- odn. t-ose
  !   nx, nt - broj podintervala mreze duz x- odn. t-ose
  !   u0_t0 - funkcija koja odredjuje granicni uslov na donjoj ivici
  !           mreze po t-osi
  !   u0_x0, u0_x1 - funkcije koje odredjuju granicne uslove na donjoj
  !                  odn. gornjoj ivici mreze po x-osi
  !   v - matrica u koju ce biti smjestena resenja
  ! Korak mreze duz bilo koje od koordinatnih osa mora biti veci od
  ! 0. Broj podintervala mreze duz bilo koje ose ne sme biti manji od
  ! 1. Funkcije koje figurisu u problemu moraju biti sintaksno ispravno
  ! zadate. Pretpostavlja se da je matrica u koju ce biti upisani
  ! rezultati alocirana izvan procedure.
  subroutine crank_nicholson (f, x0, t0, h, tau, nx, nt, u0_t0, u0_x0, u0_x1, v)
    type(string), intent(in) :: f
    real(kind=8), intent(in) :: x0, t0
    real(kind=8), intent(in) :: h, tau
    integer, intent(in) :: nx, nt
    type(string), intent(in) :: u0_t0
    type(string), intent(in) :: u0_x0, u0_x1
    real(kind=8), dimension(0:nt,0:nx), intent(out) :: v
    integer(kind=8) :: f_evaluator ! Evaluator za funkciju koja stoji na
        ! desnoj strani jednacine.
    integer(kind=8) :: u0_t0_evaluator, u0_x0_evaluator, u0_x1_evaluator
        ! Evaluatori za funkcije koje odredjuju granicne uslove.
    real(kind=8), dimension(0:nx) :: x ! Cvorovi mreze duz x-ose.
    real(kind=8), dimension(0:nt) :: t ! Cvorovi mreze duz t-ose.
    real(kind=8), dimension(1:nx-1) :: a, b, c, d ! Koeficijenti
        ! trodijagonalnog sistema jednacina po vrednostima resenja za
        ! fiksirano t.
    integer :: i, j ! Brojaci u petljama.

    ! Proveravaju se argumenti procedure i zaustavlja se program ako
    ! neki od njih nije validan.
    if (h<=0 .or. tau<=0) stop
    if (nx<1 .or. nt<1) stop
    f_evaluator=evaluator_create(f%chars)
    if (f_evaluator==0) stop
    u0_t0_evaluator=evaluator_create(u0_t0%chars)
    if (u0_t0_evaluator==0) stop
    u0_x0_evaluator=evaluator_create(u0_x0%chars)
```

```
    if (u0_x0_evaluator==0) stop
    u0_x1_evaluator=evaluator_create(u0_x1%chars)
    if (u0_x1_evaluator==0) stop

    ! Izracunavaju se cvorovi mreze duz x- i t-ose.
    x=(/(x0+j*h, j=0, nx)/)
    t=(/(t0+i*tau, i=0, nt)/)

    ! Izracunavanju se vrednosti resenja na donjoj ivici mreze po t-osi.
    v(0,:)=(/(evaluator_evaluate(u0_t0_evaluator,1,"x",x(j)), j=0, nx)/)

    ! Izracunavanju se vrednosti resenja na donjoj i gornoj ivici
    ! mreze po x-osi.
    v(1:,0)=(/(evaluator_evaluate(u0_x0_evaluator,1,"t",t(i)), i=1, nt)/)
    v(1:,nx)=(/(evaluator_evaluate(u0_x1_evaluator,1,"t",t(i)), i=1, nt)/)

    ! Unistavaju se evaluatori koji vise nisu potrebni.
    call evaluator_destroy(u0_t0_evaluator)
    call evaluator_destroy(u0_x0_evaluator)
    call evaluator_destroy(u0_x1_evaluator)

    ! Odredjuju se koeficijenti na levoj strani jednacina
    ! trodijagonalnog sistema.
    a(1)=0
    a(2:nx-1)=-tau
    b(nx-1)=0
    b(1:nx-2)=-tau
    c=2*(h*h+tau)

    ! Izracunavaju se vrednosti resenja u ostalim slojevima mreze duz
    ! t-ose.
    do j=1, nt
       ! Odredjuju se koeficijenti na desnoj strani jednacina
       ! trodijagonalnog sistema.
       do i=1, nx-1
          d(i)=tau*(v(j-1,i+1)-2*v(j-1,i)+v(j-1,i-1))+2*h*h*v(j-1,i)+&
               h*h*tau*(evaluator_evaluate(f_evaluator,2,"x t",(/x(i),t(j)/))+&
               evaluator_evaluate(f_evaluator,2,"x t",(/x(i),t(j-1)/)))
       end do

       ! Vrsi se korekcija koeficijenata u prvoj i poslednjoj jednacini.
       d(1)=d(1)+tau*v(j,0)
       d(nx-1)=d(nx-1)+tau*v(j,nx)

       ! Resava se trodijagonalni sistem, odn. odredjuju se vrednosti
       ! resenja parabolicke jednacine u tekucem sloju mreze po t-osi.
       call tridiag(nx-1,a,b,c,d,v(j,1:nx-1))
    end do

    ! Unistava se preostali evaluator.
    call evaluator_destroy(f_evaluator)
  end subroutine crank_nicholson
end module crank_nicholson_module
```

11.3 Jednačina hiperboličkog tipa

Za hiperboličku jednačinu

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + f$$

šema sa težinama tačnosti $O(h^2+\tau^2)$ je

$$v^j_{\overline{t}t,i} = \sigma_1 v^{j+1}_{\overline{x}x,i} + (1-\sigma_1-\sigma_2) v^j_{\overline{x}x,i} + \sigma_2 v^{j-1}_{\overline{x}x,i} + f^j_i.$$

11.17 *Eksplicitnom šemom ($\sigma_1 = \sigma_2 = 0$) sa koracima $h = 0.25$ i $\tau = 0.05$ rešiti mešoviti problem*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < 1, \quad 0 < \tau \le 0.15,$$

$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = x^2(1-x)^2, \quad \frac{\partial u}{\partial t}(x,0) = \sin(\pi x).$$

Rešenje: U unutrašnjim čvorovima koristi se aproksimacija

$$v^j_{\overline{t}t,i} = v^j_{\overline{x}x,i}, \quad \text{tj.} \quad \frac{1}{\tau^2}(v_i^{j+1} - 2v_i^j + v_i^{j-1}) = \frac{1}{h^2}(v^j_{i-1} - 2v_i^j + v^j_{i-1})$$

Uzimajući u obzir date vrednosti koraka i zadate početne i granične uslove, rešenje računamo za $j = 0,1,2,3$, i $i = 0,1,2,3,4$, po formulama

$$v_i^0 = x_i^2(1-x_i)^2, \qquad v_i^1 = x_i^2(1-x_i)^2 + 0.05\sin(\pi x_i),$$
$$v_0^j = 0, \qquad v_i^j = 0.04(v^j_{i+1} + v^j_{i-1}) + 1.92v_i^j - v_i^{j-1}, \qquad v_4^j = 0,$$

Ovde je za aproksimaciju izvoda u drugom početnom uslovu korišćena formula $\frac{\partial u}{\partial t} \approx v^0_{t,i} = \frac{1}{\tau}(v_i^1 - v_i^0)$). Dobijeni rezultati prikazani su tabelom

$t_j \backslash x_i$	0.00	0.25	0.50	0.75	1.00
0.00	0	0.03516	0.06250	0.03516	0
0.05	0	0.07052	0.11250	0.07052	0
0.10	0	0.10474	0.15914	0.10474	0
0.15	0	0.13693	0.20143	0.13693	0

11.18 *Rešiti mešoviti problem*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + x^2 + t^2, \qquad 0 < x < 1, \quad 0 < t \leq 0.15$$

$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = \sin(\pi x), \quad \frac{\partial u}{\partial t}(x,0) = 0,$$

eksplicitnom diferencijskom šemom tačnosti $O(h^2 + \tau^2)$. *Uzeti za* $h = 0.25$ *i* $\tau = 0.05$. *Računati sa četiri decimale.*

Rešenje: Da bi se konstruisala šema tražene tačnosti, potrebno je drugi početni uslov na odgovarajući način aproksimirati. Polazeći od količnika konačnih razlika

$$u_t(x,0) = \frac{1}{\tau}\big(u(x,\tau) - u(x,0)\big) = \left(\frac{\partial u}{\partial t} + \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}\right)_{/(x,0)} + O(\tau^2),$$

i uzimajući u obzir jednačinu i oba zadata početna uslova, dobijamo da je

$$u_t(x,0) = \frac{\tau}{2}(-\pi^2 \sin(\pi x) + x^2) + O(\tau^2).$$

Zanemarivanjem člana $O(\tau^2)$ nalazimo da je tražena aproksimacija

$$v_i^1 = \sin(\pi x_i) + \frac{\tau^2}{2}(x_i^2 - \pi^2 \sin(\pi x_i)).$$

Koristeći standardne količnike konačnih razlika za aproksimaciju izvoda drugog reda u jednačini, konačno dobijamo traženu šemu

$$v_i^0 = \sin(\pi x_i), \qquad v_i^1 = \sin(\pi x_i) + 0.00125(x_i^2 - \pi^2 \sin(\pi x_i)) \quad i = 0, \ldots 4,$$

$$v_i^{j+1} = 1.92 v_i^j + 0.04(v_{i+1}^j + v_{i-1}^j) - v_i^{j-1} + 0.0025(x_i^2 + t_j^2), \qquad i = 1,2,3, \quad j = 1,2,$$

$$v_0^j = v_4^j = 0, \qquad j = 2,3.$$

Rezultati izračunavanja ovom šemom prikazani su tabelom

$t_j \setminus x_i$	0.00	0.25	0.50	0.75	1.00
0.00	0	0.7071	1.0000	0.7071	0
0.05	0	0.6985	0.9880	0.6991	0
0.10	0	0.6737	0.9535	0.6771	0
0.15	0	0.6333	0.8974	0.6386	0

11.19 *Šemom sa težinama* $\sigma_1 = 1/4$, $\sigma_2 = 0$ *rešiti mešoviti problem*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(e^{-x}\frac{\partial u}{\partial x}\right) - u + x\sin(\pi x) \qquad 0 < x < 1, \quad 0 \leq t \leq 1.05$$

$$u(0,t) = 1, \qquad u(1,t) = t + 2,$$

$$u(x,0) = 1 + x + x^2 - 2x^3 + x^4, \qquad \frac{\partial u}{\partial t}(x,0) = 1 - \cos(2\pi x) + x.$$

Uzeti za $h = 0.25$ *i* $\tau = 0.35$. *Računati sa pet decimala.*

Rešenje: Problem se rešava na mreži

$$\omega = \{(x_i, t_j) \,|\, x_i = ih,\ t_j = j\tau,\ i = 0, \ldots, 4,\ j = 0, \ldots, 3,\ h = 0.25,\ \tau = 0.35\}$$

diferencijskom šemom

$$v_i^0 = 1 + x_i + x_i^2 - 2x_i^3 + x_i^4, \qquad v_i^1 = v_i^0 + \tau(1 - \cos(2\pi x_i) + x_i)$$

$$v_{\overline{t}t,i}^j = \frac{1}{4}\left(\frac{1}{2}\big((e^{-x}v_{\overline{x}})_x + (e^{-x}v_x)_{\overline{x}}\big) - v\right)_i^{j+1} \qquad i = 1,2,3, \quad j = 1,2,$$

$$+ \frac{3}{4}\left(\frac{1}{2}\big((e^{-x}v_{\overline{x}})_x + (e^{-x}v_x)_{\overline{x}}\big) - v\right)_i^{j} + x_i \sin(\pi x_i)$$

$$v_0^j = 1, \qquad v_4^j = 2 + t_j$$

Kada se aproksimacije izvoda napišu u razvijenom obliku, ova šema za $j = 1, 2$ predstavlja sistem linearnih jednačina sa trodijagonalnom matricom

$$v_0^{j+1} = 1$$

$$-\frac{1}{8h^2}\big(e^{-x_{i+1}} + e^{-x_i}\big)v_{i+1}^{j+1} + \left(\frac{1}{8h^2}\big(e^{-x_{i+1}} + 2e^{-x_i} + e^{-x_{i-1}}\big) + \frac{1}{\tau^2} + \frac{1}{4}\right)v_i^{j+1}$$

$$- \frac{1}{8h^2}\big(e^{-x_i} + e^{-x_{i-1}}\big)v_{i-1}^{j+1}$$

$$= \frac{3}{8h^2}\big(e^{-x_{i+1}} + e^{-x_i}\big)v_{i+1}^{j} - 3\left(\frac{1}{8h^2}\big(e^{-x_{i+1}} + 2e^{-x_i} + e^{-x_{i-1}}\big) - \frac{2}{3\tau^2} + \frac{1}{4}\right)v_i^{j}$$

$$+ \frac{3}{8h^2}\big(e^{-x_i} + e^{-x_{i-1}}\big)v_{i-1}^{j} - \frac{1}{\tau^2}v_i^{j-1} + x_i \sin(\pi x_i) \qquad i = 1,2,3$$

$$v_4^{j+1} = 2 + t_{j+1}$$

Vrednosti rešenja za $j = 0$ i $j = 1$ određene su početnim uslovima

$$v_i^0 = 1 + x_i + x_i^2 - 2x_i^3 + x_i^4, \qquad v_i^1 = v_i^0 + \tau(1 - \cos(2\pi x_i) + x_i)$$

Rezultati izračunavanja dati su sledećom tabelom

$t_j \setminus x_i$	0.00	0.25	0.50	0.75	1.00
0.00	1	1.28516	1.56250	1.78516	2.00
0.35	1	1.72266	2.43750	2.39766	2.35
0.70	1	1.66265	2.23940	2.66839	2.70
1.05	1	1.20944	1.63763	2.33141	3.05

A

Fortran 90

Fortran 90 predstavlja modernu verziju *Fortran*-a u kojoj je jeziku dodata većina funkcionalnosti koja karakteriše ostale imperativne programske jezike, a koja je nedostajala *Fortran*-u *77*. Na taj način je ovaj jezik, čiji su primarni domen upotrebe oduvek bila matematička izračuvanja, ponovo postao najbolji izbor za rešavanje numeričkih problema. U ovom dodatku će ukratko biti predstavljena sintaksa *Fortran*-a *90*, u onoj meri koliko je dovoljno da se razumeju *Fortran* programi predstavljeni u ovoj zbirci. Nastojalo se da način izlaganja bude prilagođen čitaocu koji već poznaje programiranje na višim programskim jezicima (prvenstveno na *C* jeziku). S obzirom da nije bila namera da se u potpunosti predstavlja sintaksa *Fortran*-a (a to bi i bilo nemoguće na ovako kratkom prostoru), za sve eventualne nejasnoće treba konsultovati drugu dostupnu literaturu, a pre svega dokumentaciju svog prevodioca.

A.1 Osnovni elementi jezika

Program na *Fortran*-u se sastoji od određenog broja linija koda koje sadrže naredbe. Svaka linija može biti duga najviše 132 karaktera; ukoliko ima potrebe za dužim linijama može se naznačiti da se tekuća linija nastavlja u sledećoj liniji tako što se za njen poslednji karakter stavi karakter `&`. Ukoliko dve ili više naredbi stoji u istoj liniji, onda se iste razdvajaju karakterom `;`.

Fortran ne pravi razliku između malih i velikih slova u ključnim rečima ili imenima promenljivih i procedura. Blanko karakteri u *Fortran*-u imaju određeno leksičko značenje, jer bar jedan blanko karakter mora da stoji između dve ključne reči jezika ili između ključne reči i nekog imena (svakako, blanko karakteri mogu po volji biti korišćeni i na drugim mestima u cilju poboljšanja čitljivosti, npr. oko operatora i sl.).

Komentari se u *Fortran*-u naznačavaju karakterom `!`; sadržaj linije počev od ovog karaktera pa do kraja linije se ignoriše od strane prevodioca.

Imena (promenljivih ili procedura) u *Fortran*-u mogu imati do 31 karakter i mogu se sastojati od slova, brojeva i donje crte (`_`). Prvi karakter u imenu mora

biti slovo.

Struktura koda na *Fortran*-u je dosta čvrsto definisana; precizna defincija ove struktura će biti data kasnije. Sledi primer kratkog *Fortran 90* programa koji štampa pozdravnu poruku na standardni izlaz i na kome će biti pokazani osnovni elementi koji sačinjavaju neki program na *Fortran*-u:

```
program hello
  implicit none

  write (*, *) 'Hello, world!'
end program hello
```

Početak glavnog programa naznačava ključna reč `program` koju sledi ime programa. Zatim dolazi blok deklaracija, a potom naredbe koje sačinjavaju telo programa. U ovom slučaju blok delaracija se sastoji samo od `implicit none` naredbe. Ovom naredbom treba da počinje svaki program i procedura na *Fortran*-u *90*; radi se o tome da ukoliko se ova naredba ne navede *Fortran 90* zbog kompatibilnosti sa *Fortran*-om *77* dozvoljava da promenljive budu nedeklarisane, u kom slučaju prevodilac njihov tip određuje prema prvom slovu imena. Ovo svakako treba izbegavati i zato obavezno treba ovom naredbom počinjati svaki program odn. proceduru. Telo gornjeg programa sastoji se od naredbe koja štampa poruku na standardnom izlazu. Više o naredbama za ulaz odn. izlaz kasnije, a ovde treba uočiti da se stringovi na *Fortran*-u mogu navesti unutar jednostrukih navodnika. Na kraju programa dolazi `end program` naredba koju opet prati ime programa. Za *Fortran* je karakteristično da svaku početnu naredbu neke konstrukcije uvek prati odgovarajuća završna naredba koja počinje sa `end` i onda ponavlja početnu naredbu konstrukcije, pa na to treba obratiti pažnju.

Svaka naredba *Fortran*-a može biti označena labelom koja je predstavljena brojem od jedne do pet cifara. Labele se u *Fortran*-u *90* retko koriste, ali ipak ima još uvek nekih naredbi, kao naravno `goto` naredba ili pak `format` naredba, koje se bez labela ne mogu koristiti.

A.2 Tipovi podataka

Fortran podržava sledeće osnovne tipove podataka: `character` za stringove od jednog ili više karaktera, `logical` za istinitosne vrednosti, `integer` za cele brojeve, `real` (i `double precision`) za realne brojeve jednostruke odn. dvostruke tačnosti i `complex` za kompleksne brojeve.

Deklaracija promenljivih u *Fortran*-u se sastoji od tipa, opcionih atributa, zatim sledi `::` i onda imena promenljivih koje se deklarišu, razdvojena zarezima. Imena promenljivih mogu biti praćena znakom `=` koga onda sledi inicijalizator odn. vrednost na koju se postavlja data promenljiva. Tako bi npr. prosta deklaracija realnih promenljivih `x` i `y` bila oblika:

```
real :: x, y
```

Za *Fortran* je karakteristično da dužina stringova mora biti navedena prilikom deklaracije i da se ne može menjati (ovo je jedan od najvećih nedostataka *Fortran*-a,

koji je doduše donekle prevaziđen time što standard predviđa postojanje podrške za stringove promenljive dužine preko posebnog modula `iso_varying_string`; problem sa ovim modulom je što ga trenutno mali broj prevodilaca zaista i isporučuje, tako da su u biblioteci `libnumerics` korišćeni isključivo stringovi fiksne dužine). Za specificiranje dužine stringova, nakon što se navede `character` tip, između zagrada se stavlja `len=` i broj koji određuje dužinu stringa. Ako se ovo izostavi, smatra se da je string dužine 1. Tako bi npr. deklaracije stringova `c` dužine 1 i `hello` dužine 5 bile oblika[1]:

```
character :: c
character(len=5) :: hello
```

Ostali tipovi mogu na isti način imati specificiranu širinu u bajtovima, s tim što se onda umesto `len` specifikatora koristi `kind` specifikator, koji dalje može imati samo vrednosti 1, 2, 4 ili 8. Tako bi npr. deklaracije celog 16-to bitnog broja `i` odn. realnog broja `d` širine 8 bajtova (tj. realnog broja dvostruke tačnosti, za čije specificiranje inače alternativno može da se koristi već pomenuti `double precision` tip) bile oblika:

```
integer(kind=2) :: i
real(kind=8) :: d
```

Podrazumevana širina svih tipova koji podržavaju ovaj specifikator iznosi 4.

Kao logičke konstante se koriste `.true.` odn. `.false.` za odgovarajuće vrednosti. Celobrojne i realne konstante se pišu u uobičajenoj notaciji. Kompleksne konstante se pišu kao dva realna broja u zagradama, razdvojena zarezom. Tako bi npr. deklaracije nekih promenljivih sa inicijalizatorima mogle imati oblik:

```
logical :: flag=.true.
integer :: n=10
real :: x=-1.67, y=2.34e-3
complex :: i=(0,1), root=(1.09868,-0.45509)
```

Konstante svih pomenutih tipova mogu na kraju imati karakter `_`, koga onda sledi broj koji određuje širinu.

Konstante koje su stringovi se pišu između jednostrukih ili dvostrukih znakova navoda. Ukoliko string sadrži jednostruke ili dvostruke znakove navoda, onda se može string terminirati drugih tipom navodnika ili se alternativno može karakter koji predstavlja navodnike ponoviti. Slede deklaracije sa inicijalizatorima nekih promenljivih koje su stringovi:

```
character(len=5) :: hello='hello', hello2="hello"
character(len=8) :: reply="I won't!", reply2='I won''t!'
```

Ukoliko je promenljiva koja je string incijalizovana, može se u njenoj deklaraciji izostaviti specifikacija dužine, što se radi tako što se umesto broja koji predstavlja dužinu string upiše karakter `*`. Tako bi deklaracije prve od promenljivih iz prethodnog primera mogla biti i oblika:

[1] za *C* programere treba naglasiti da na *Fortran*-u stringovi nisu terminirani nulom

```
character(len=*) :: hello='hello'
```

Specifikaciju tipa u deklaraciji mogu da prate brojni atributi, koji se navode zarezima razdvojeni od specifikacije tipa i jedan od drugog, a ispred `::` unutar deklaracije. Jedan od tih atributa je i `parameter`, kojim se specificira da je data promenljiva odn. promenljive ustvari konstanta. S obzirom da se konstante ne mogu menjati u programu, deklaracija svake konstante treba obavezno da sadrzi inicijalizator. Primer deklaracije konstante bi bio:

```
real(kind=8), parameter :: pi=3.1416
```

Fortran omogućava definisanje izvedenih, korisničkih tipova. Ovakvi tipovi se navode ključnom rečju `type`, koju sledi ime tipa i onda specifikacija polja koja sačinjavaju taj tip. Definicija se završava sa `end type` i ponovljenim imenom tipa. Tako bi definicija izvedenog tipa koji predstavlja tačku u 2D prostoru mogla biti sledećeg oblika:

```
type coord
  real :: x, y
end type coord
```

Jednom kada se definiše izvedeni tip, onda se isti može navoditi u deklaracijama umesto ugrađenih tipova tako što se navede ključna reč `type` i onda u zagradama ime izvedenog tipa. Deklaracije promenljivih tipa `coord` bi tako imale oblik:

```
type(coord) :: O, P
```

Pojedine komponente izvedenog tipa se referenciraju tako što se iza imena promenljive doda karakter `%`, a onda ime komponente. Tako bi se se npr. dodela vrednosti promenljivoj `P` iz gornje deklaracije mogla izvršiti na sledeći način:

```
P%x=5; P%y=3
```

Inicijalizacija promenljive izvedenog tipa se može izvšiti tako što se iza znaka `=` navede ime tipa, a onda u zagradama vrednosti za komponente tipa, onim redom kako su komponente nabrojane u definiciji tipa. Inicijalizacija tačaka iz pretposlednjeg primera bi se tako mogla izvršiti na sledeći način:

```
type(coord) :: O=coor(0,0), P=coord(5,3)
```

Kako je već pomenuto ranije, *Fortran* zbog kompatibilnosti unazad omogućava i da se koriste nedeklarisane promenljive, u kom slučaju određuje njihov tip prema prvom slovu imena. S obzirom da ovakav način programiranja rezultuje u nečitkim i teškim za održavanje programima, preporučuje se da svaka programska celina u *Fortran*-u počinje `implicit none` naredbom kojom se isključuje ovakvo interpretiranje nedeklarisanih promenljivih odnosno kojom se praktično specificira da pojava svake nedeklarisane promenljive bude prijavljena kao greška.

A.3 Polja

Fortran podržava jedno- i višedimenzionalna polja za svaki od osnovnih i izvedenih tipova podataka. Da bi se deklarisalo polje, treba datom tipu dodati atribut `dimension` koga slede zagrade unutar kojih se navode početni i krajnji indeksi polja po svakoj dimenziji. Početni i krajnji indeksi po datoj dimenziji su razdvojeni dvotačkom, a specifikacije indeksa za pojedine dimenzije su razdvojene zarezom. Na taj način, deklaracije vektora celih brojeva od 11 elementa sa indeksima od -5 do 5 odn. matrice realnih brojeva dimenzija 4×4 sa indeksima od 0 do 3 duž svake dimenzije bi imale oblik:

```
integer, dimension(-5:5) :: v
real, dimension(0:3, 0:3) :: A
```

Referenciranje elemenata polja se vrši tako što se iza naziva odgovarajuće promenljive u zagradama navedu zarezima razdvojeni indeksi. Tako bi npr. dodela nekim elementima gore deklarisanih polja bila oblika:

```
v(-3)=6
a(0,0)=0; a(1,3)=13.7
```

Jednodimenzionalno polje se može pri deklaraciji inicijalizovati tako što se u inicijalizatoru navedu zarezima razdvojeni elementi polja i to između oznaka `(/` i `/)`. Tako bi deklaracije polja sa inicijalizatorom mogla biti oblika:

```
real, dimension(1:3) :: origin=(/0,0,0/)
```

Višedimenzionalna polja se moraju incijalizovati korišćenjem `reshape()` ugrađene procedure. Ova procedura menja oblik polja koje joj je preneseno kao prvi argument na onaj oblik koji je zadat drugim argumentom. Tako bi npr. deklaracija sa inicijalizacijom jedinične 3×3 matrice bila oblika:

```
real, dimension(0:2,0:2) :: one=reshape((/1,0,0,0,1,0,0,0,1/),(/3,3/))
```

Procedura `reshape()` spada, kako je već rečeno, u tzv. ugrađene procedure. *Fortran* ima veliki broj ovih ugrađenih procedura koje se mogu direktno, bez ikakvih predradnji, koristiti u programima. Ove procedure otprilike odgovaraju *C* standardnoj biblioteci. U vezi sa poljima, pored `reshape()` procedure na raspolaganju je i niz drugih procedura kao npr. `lbound()` i `ubound()` koje vraćaju najmanji odn. najveći indeks (eventualno po datoj dimenziji), `size()` koja vraća dužinu polja po datoj dimenziji ili eventualno ukupan broj elemenata polja, zatim `sum()` koja računa sumu elemenata polja duž određene dimenzije ili sumu svih elemenata polja itd. Za listu svih ugrađenih procedura treba konsultovati dokumentaciju prevodioca, a takođe se treba potruditi da se što više ovih procedura nauči, jer je poznavanje istih jedan od ključnih uslova za efikasno programiranje na *Fortran*-u.

Fortran podržava i polja promenljive dužine. Kod deklaracije ovakvih polja treba izostaviti indekse u atributu `dimension`, a dodati atribut `allocatable`. Nakon ovoga se u programu može dinamički alocirati memorija za elemente polja ugrađenom procedurom `allocate()`. Kada se završi sa korišćenjem polja, memorija se oslobađa ugrađenom procedurom `deallocate()`. Sledi isečak koda koji demonstrira ovaj tip polja:

```
integer :: n
real, dimension(:,:), allocatable :: A

! Recimo da se u prvom delu koda ucita vrednost promenljive n.

allocate(A(1:n,1:n))

! U ovom delu koda se koristi matrica.

deallocate(A)
```

Jedna od ključnih karakteristika *Fortran*-a koja ga čini pogodnim za numerička izračunavanja jeste mogućnost direktnog rada sa poljima kao sa skalarnim tipovima. Veliki broj operatora podržava polja kao operande; tako je moguće npr. kod operatora dodele (`=`) da se na levoj strani nađe promenljiva koja predstavlja polje, a na desnoj strani sve ono što je pomenuto u vezi sa inicijalizatorima polja. Na ovaj način operacije sa poljima se mogu u *Fortran*-u veoma kompaktno zapisati, a takođe prevodilac ima mogućnost da takve operacije prevede direktno u paralelne instrukcije ako ih data arhitektura za koju se prevodi program podržava. *Fortran* takođe podržava rad sa sekcijama polja, odn. skupom elemenata polja. Notacija za referenciranje sekcije polja je ista kao za referenciranje pojedinačnih elemenata, samo što se umesto jednog indeksa ovde specificiraju opsezi indeksa duž jedne ili više dimenzija. Jedini zahtev koji *Fortran* postavlja pri korišćenju operacija nad poljima ili sekcijama polja jeste da operandi budu istih dimenzija.

A.4 Izrazi

Fortran podržava standardan set operatora koji se inače sreću u višim programskim jezicima.

Od aritmetičkih operatora podržani su naravno operatori `+`, `-`, `*` i `/`. *Fortran* direktno podržava i operator stepenovanja, čija je oznaka `**`. Operatori `+` i `-` se mogu javiti i kao unarni operatori; za *Fortran* je karakterisitčno da dva operatora ne mogu stajati jedan pored drugog, pa tako npr. izraz `4+-3` nije sintaksno ispravan, već se mora napisati kao `4+(-3)`.

Relacioni operatori u *Fortran*-u su `==` za jednakost, `/=` za nejednakost i `<`, `<=`, `>` i `>=` za odgovarajuće relacije. Rezultat primene ovih operatora je `logical` tipa. Pored numeričkih operanada, kao operandi za relacione operatore mogu stajati i stringovi, u kom slučaju se vrši leksikografsko poređenje.

Od logičkih operatora podržani su negacija (`.nor.`), konjukcija (`.and.`) i disjunkcija (`.or.`), a takođe i ekvivalencija (`.eqv.`) koja praktično odgovara operatoru "ekskluzivno nili", kao i negacija ekvivalencije (`.neqv.`) koja odgovara operatoru "ekskluzivno ili".

Za stringove je definisan i operator konkatenacije (nadovezivanja). Oznaka za ovaj operator je `//`.

Redosled prioriteta operatora u *Fortran*-u je uobičajen i vrlo sličan redosledu prioriteta operatora na *C*-u. Stepenovanje naravno ima veći prioritet od svih drugih aritmetičkih operatora, dok konkatenacija stringova po prioritetu dolazi iza arit-

metičkih, a ispred relacionih operatora. Prioritet pojedinačnih elemenata izraza može naravno biti eksplicitno određen korišćenjem zagrada.

Ukoliko su operandi različitog tipa, primenjuju se uobičajena pravila o implicitnoj konverziji. Tako npr. ukoliko je prvi operand neke aritmetičke operacije cio broj, a drugi operand realan broj, prevodi se prvi operand u realan broj pre nego što se izvrši operacija; ili recimo ako je prvi operand realan, a drugi kompleksan broj, onda se pre izračunavanja prevodi prvi operand u kompleksan broj. Isto važi za operator dodele: kada se izračuna izraz na desnoj strani ovog operatora, on se po potrebi prevodi u tip promenljive koja stoji na levoj strani operatora.

U sekciji o poljima već je pomenuto da se jedan broj operatora u *Fortran*-u može primeniti i nad poljima. U takvim slučajevima se paralelno primenjuje operator na odgovarajuće elemente polja; operandi moraju biti istih dimenzija. Druga varijanta je da je jedan operand polje, a drugi operand skalar i tada se primenjuje dati operator nad svakim elementom polja i skalarom, dajući kao rezultat polje istih dimenzija kao dato polje.

Određen broj operatora je definisan i za izvedene tipove podataka. *Fortran* takođe pruža mogućnost da se redefinišu ostali operatori za izvedene tipove, tj. moguće je uraditi ono što se označava kao *operator overloading* u objektno-orijentisanim programskim jezicima.

U vezi sa operatorima treba reći nešto i o već pomenutim ugrađenim funkcijama u *Fortran*-u. Veliki broj ovih funkcija je na raspolaganju programeru i one se mogu podeliti u nekoliko grupa, od kojih su najvažnije:

- funkcije za eksplicitnu konverziju tipova (`int()`, `real()`, `cmplx()` itd.)
- matematičke funkcije (`sqrt()`, `exp()`, `log()`, `log10()`, `sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`, `sinh()`, `cosh()`, `tanh()` itd.)
- numeričke funkcije (`abs()`, `min()`, `max()`, `floor()`, `ceiling()` itd.)
- funkcije za rad sa stringovima (`len()`, `scan()`, `index()`, `repeat()`, `trim()` itd.)

Takođe postoje i ranije pominjane ugrađene funkcije za rad sa poljima, zatim funkcije koje se tiču generisanja slučajnih brojeva, funkcije koje barataju sa vektorima i matricama, funkcije koje rade sa kompleksnim brojevima, funkcije za manipulaciju bitovima itd. Za kompletan spisak treba konsultovati dokumentaciju svog prevodioca.

A.5 Kontrolne strukture

Kao i za operatore, tako se i za kontrolne strukture može reći da *Fortran* podržava većinu kontrolnih struktura koji se susreću u ostalim programskim jezicima, uz dodatak nekih korisnih strutkura koje su karakteristične samo za *Fortran*.

Naredba grananja ima standardnu strukturu: počinje ključnom rečju `if` koju sledi neki logički izraz u zagradama. Zatim dolazi ključna reč `then` i blok naredbi koji se izvršava ako je uslov ispunjen. Naredba se završava sa `end if`. Ukoliko se

pomenuti blok naredbi sastoji samo od jedne naredbe onda se `then` i `end if` mogu izostaviti i naredba se piše u istom redu u kome su ključna reč `if` i logički izraz. Ukoliko postoji blok naredbi koji treba izvršiti za slučaj kada uslov nije ispunjen onda se ispred `end if` stavlja ključna reč `else` i taj blok naredbi. Sledi primer `if` naredbe:

```
if (x/=0 .and. y/=0) then
  z = 1/(x*y)
else
  z = 1
end if
```

Narebda grananja može sadržati i ispitivanja drugih uslova, u kom slučaju se koristi ključna reč `elseif` praćena novim uslovom. Sledi primer ovog oblika naredbe grananja:

```
if (x<-5) then
  y=abs(x)-5
elseif (x<3) then
  y=10*x
else
  y=log(x)
endif
```

U slučajevima kada se grananje vrši na osnovu izbora jedne vrednosti iz skupa vrednosti koristi se `select case` konstrukcija. Ova konstrukcija počine sa `select case`, a zatim u zagradama sledi izraz na osnovu čije vrednosti se vrši grananje. Potom sledi niz `case` stavki praćenih blokovima naredbi koje se izvršavaju u slučaju da izračunati izraz ima vrednost koja je navedena u stavci. Konstrukcija se završava sa `end select`. Vrednosti za pojedinačne stavke se navode unutar zagrada iza ključne reči `case` i mogu biti ili pojedinačni brojevi ili nizovi brojeva razdvojeni zarezima ili opsezi sa eventualno izostavljenom donjom ili gornjom granicom. Podrazumevana stavka se opciono navodi sa `case default` i ista sadrži blok naredbi koji se izvršava u slučaju kada izraz ima vrednost različitu od svih vrednosti navedenih u ostalim stavkama. Može se dakle videti da *Fortran* ima znatno fleksibilniju `case` konstrukciju od *C*-a; za *C* programere je važno naglasiti i da se *Fortran*-u izvršavanje prelazi na naredbu iza `select case` konstrukcije čim se završi blok naredbi koji odgovara datoj stavci, tj. nema potrebe za nekim ekvivalentom `break` naredbe iz *C*-a. Sledi primer koji demonstrira upotrebu `select case` konstrukcije:

```
select case (x)
case (:-2)
  y=-1
case (-1,1,2)
  y=0
case (5:10)
  y=2
case default
  y=3
end select
```

Najjednostavniji oblik petlje na *Fortran*-u je `do` konstrukcija. Ova konstrukcija počinje ključnom rečju `do`, zatim sledi blok naredbi koji sačinjava telo petlje i na kraju dolazi `end do`. Iz ovakve petlje se može izaći koristeći `exit` naredbu. Takođe, unutar petlje se može nastaviti sa narednom iteracijom pre nego što se tekuća iteracija u potpunosti završi pomoću `cycle` naredbe. Naredbe `cycle` i `exit` za trenutni prekid tekuće iteracije odn. čitave petlje se zapravo mogu primeniti u bilo kojoj konstrukciji koja predstavlja petlju u *Fortran*-u. Sledi jedan primer proste `do` konstrukcije:

```
i=1
do
  if (i>100) exit
  if (i>=50 .and. i<60) cycle
  A(i)=i
  i=i+1
end do
```

Fortran podržava i tzv. brojačku `do` petlju, kod koje se iza ključne reči `do` na vrhu petlje navodi brojačka promenljiva, a onda znak jednakosti i zarezima razdvojene početna i krajnja vrednost koju ta promenljiva uzima. Opcino se iza još jednog zareza može navesti korak, tj. vrednost za koju će brojačka promenljiva biti uvećana nakon svake iteracije. Prethodni primer bi korišćenjem brojačke petlje mogao biti kompaktnije zapisan kao:

```
do i=1,100
  if (i>=50 .and. i<60) cycle
  A(i)=i
end do
```

Standardna petlja sa uslovom je takođe podržana preko `do while` konstrukcije. Ova konstrukcija počinje sa `do while` nakon čega dolazi uslov, onda telo petlje i na kraju `end do`. Petlja se izvršava sve dok je uslov ispunjen. Evo opet prethodnog primera, izmenjenog tako da koristi `do while` konstrukciju:

```
i=0
do while (i<=100)
  if (i>=50 .and. i<60) cycle
  A(i)=i
  i=i+1
end do
```

Jedan važan oblik petlje koji se često koristi u *Fortran*-u je tzv. implicitna `do` petlja. Ova konstrukcija je oivičena zagradama, unutar kojih se nalaze zarezima razdvojeni neki izraz i specifikacija brojačke promenljive koja je ista kao kod brojačke `do` petlje (dakle ime promenljive, pa znak jednakosti, pa zarezima razdvojene početna i krajnja vrednost i eventualno korak). Efekat ove petlje jeste da se kreira zarezima razdvojena lista vrednosti izraza kada se u njega uvrste sve vrednosti brojačke promenljive. Ovaj tip petlje je pogodan za inicijalizaciju ili dodelu polja, kao i za ulazno-izlazne operacije. Tako bi se npr. vektor `v` čiji su elementi 0, 3, 4, 5 i 6 mogao kompaktno inicijalizovati naredbom:

```
v=(/0, (3+i, i=0,3)/)
```

U vezi sa naredbama za kontrolu toka obično se pominje i `goto` naredba koja je podržana i na *Fortran*-u; ovom naredbom se skače na specificiranu labelu, pri čemu je labela predstavljena brojem kako je već ranije pojašnjeno. Kontrolne strukture na *Fortran*-u mogu imati i imena, pri čemu se ime piše ispred strukture i iza njega se stavlja dvotačka. Za davanje imena kontrolnim strukturama važi sve što je rečeno za ostale identifikatore; ukoliko kontrolna struktura ima ime, onda to ime mora biti ponovljeno na kraju `end` naredbe kojom se završava struktura. Imena osim povećanja čitljivosti mogu biti korisna i jer se mogu navesti iza `cycle` odn. `exit` naredbi u kom slučaju se ove naredbe odnose na imenovanu kontrolnu strukturu, čime se mogu postići efekti preskakanja na spoljašnju petlju iz unutrašnje odn. izlaska iz ugnježdene petlje.

A.6 Procedure

Kao i svi ostali viši programski jezici, *Fortran* omogućava dekompoziciju programa na određene celine. Primarno sredstvo dekompozicije su procedure; *Fortran* razlikuje dva tipa procedura i to potprograme, koji se navode ključnom rečju `subroutine`, i funkcije, koje se navode ključnom rečju `function`. Jedina razlika između ova dva tipa procedura je što potprogrami ne vraćaju nikakav rezultat, dok funkcije vraćaju rezultat; takođe, pri pozivanju potprograma mora se ispred imena potprograma navesti ključna reč `call` dok se funkcije pozivaju samo po imenu.

Sintaksa potprograma i funkcija je vrlo slična. Kako je već rečeno, potprogrami počinju ključnom rečju `subroutine`, dok funkcije počinju ključnom rečju `function` kojoj prethodi specifikacija tipa podataka koga vraća funkcija. Nakon toga u oba slučaja sledi lista sa zarezima razdvojenim imenima argumenata u zagradama, onda lista deklaracija u kojoj su na početku deklarisani argumenti, a potom lokalne promenljive. Zatim dolazi telo procedure; za funkcije je karakteristično da se negde unutar ovog dela koda mora naći dodela povratne vrednosti, koja se vrši prosto tako što se ime funkcije stavi na levi stranu naredbe dodele. Na kraju procedure se stavlja `end subroutine` odn. `end function` praćeno ponovljenim imenom procedure.

Na deklaracije argumenata se odnosi sve što je rečeno za deklaracije običnih promenljivih. Argumenti procedura treba međutim da imaju jedan dodatni atribut čije je ime `intent` i koga sledi u zagradama jedna od ključnih reči `in`, `out` ili `inout`. Na ovaj način se specificira svrha argumenta, tj. da li je u pitanju ulazna vrednost, izlazna vrednost ili ulazno-izlazna vrednost. Ako je neki argument deklarisan kao isključivo ulazni, njegova vrednost se ne može menjati u telu procedure (ovo je ekvivalento pridruživanju kvalifikatora `const` nekom argumentu funkcije u *C*-u); ako je pak argument deklarisan kao isključivo ulazni, onda se on može naći samo na levoj strani naredbe dodele u telu procedure. Sledi primer procedure koja računa dužinu hipotenuze pravouglog trougla pri čemu su date dužine kateta i koji demonstrira ovo što je dosad rečeno o procedurama:

```
real function hypot(cat0,cat1)
  real, intent(in) :: cat0, cat1
```

```
  hypot=sqrt(cat0**2+cat1**2)
end function hypot
```

Pozivanje procedure se vrši na uobičajen način - navodi se ime procedure, a onda u zagradama zarezima razdvojena lista stvarnih argumenata (takođe, kako je već rečeno, kada se poziva potprogram onda ovome mora da prethodni ključna reč `call`). Tako bi gornja procedure mogla biti pozvana na sledeći način:

```
real :: a, b, c

! Ovde se ucitavaju duzine kateta a i b.

c=hypot(a,b)
```

Fortran omogućava i da se stvarni argumenti ne navode istim redom kao i fiktivni argumenti, u kom slučaju se u listi argumenata za svaki argument navodi ime fiktivnog argumenta, zatim znak jednakosti i onda vrednost stvarnog argumenta. Tako bi gornja procedura sa obrnutim redosledom argumenata mogla biti pozvana na sledeći način:

```
real :: a, b, c

! Ovde se ucitavaju duzine kateta a i b.

c=hypot(cat1=b,cat0=a)
```

Za promenljive u *Fortran*-u važe pravila dosega koja su identična pravilima dosega na *C*-u, što za procedure znači da su njihovi fiktivni argumenti i lokalne promenljive vidljive samo unutar procedura, kao i da lokalne promenljive imaju prioritet u slučajevima kada lokalna i globalna promenljiva imaju isto ime (tj. tada u proceduri data globalna promenljiva nije vidljiva).

Za lokalne promenljive procedura se može naznačiti da se čuvaju između poziva procedura, što se radi atributom `save`, lokalne promenljive sa ovakvim atributom su ekvivalentne lokalnim promenljivim sa `static` kvalifikatorom u *C*-u. Sve lokalne promenljive koje su inicijalizovane imaju implicitno ovaj atribut. *Fortran* takođe ima ugrađenu podršku za opcione argumente, tj. za argumente kojima se ne mora zadati vrednost, u kom slučaju se uzima da oni imaju neke podrazumevane vrednosti određene samom procedurom. Opcioni argumenti se naznačavaju atributom `optional`.

Kada se stringovi ili polja prenose kao argumenti, moguće je izostaviti broj karaktera ili dužinu polja (tačnije, tada se za string piše `len=*`, a za polje se stavlja samo `:` za dimenziju). Na taj način, moguće je napraviti procedure koje rade sa stringovima ili poljima proizvoljne dužine. Sledi primer procedure koja koristi ovu mogućnost da implementira računanje euklidske norme za vektor proizvoljne dužine:

```
real function euclid_norm (v)
  real, dimension(:), intent(in) :: v
  real :: sum
```

```
  integer :: i

  sum=0
  do i=lbound(v),ubound(v)
    sum=sum+v(i)**2
  end do
  euclid_norm=sqrt(sum)
end function euclid_norm
```

Alternativno, dužina stringa odn. dimenzija polja za argument procedure ili lokalnu promenljivu se može specificirati koristeći u odgovarajućim izrazima neki od argumenata. Tako bi se npr. procedura koja treba da izračuna euklidsku normu sume dva data vektora $v0$ i $v1$ čija je dužina n koristeći pritom prethodnu proceduru mogla napisati u sledećem obliku:

```
real function euclid_norm_of_sum (n, v0, v1)
  integer, intent(in) :: n
  real, dimension(1:n), intent(in) :: v0, v1
  real, dimension(1:n) :: v
  integer :: i

  v=0
  do i=1,n
    v(i)=v0(i)+v1(i)
  end do

  euclid_norm_of_sum=euclid_norm(v)
end function euclid_norm_of_sum
```

Fortran naravno podržava i rekurzivne procedure, pri čemu je karakteristično da ovakve procedure moraju biti deklarisane dodavanjem ključne reči `recursive` ispred ključne reči `subroutine` ond. `function` u zaglavlju procedure.

A.7 Programi i moduli

Osnovne programske jedinice u *Fortran*-u su programi i moduli. Programi su naravno namenjeni izvršavanju, dok moduli služe za grupisanje deklaracija i procedura koji se koriste u većem broju programa. Struktura programa i modula je relativno slična. Programi počinju ključnom rečju `program`, a moduli ključnom rečju `module` nakon čega sledi ime programa odn. modula. Potom može da sledi jedna ili više `use` naredbi sa imenima modula koje dati program odn. modul koristi. Onda obično u oba slučaja dolazi `implicit none` naredba, čija je svrha ranije pojašnjena. U slučaju modula zatim može da dođe niz `public` naredbi kojima se deklarišu simboli koji su dostupni kodu koji koristi modul. Potom slede definicije i deklaracije, s tim što su to u slučaju programa najčešće deklaracije lokalnih promenljivih za program, a u slučaju modula su to obično definicije tipova i konstanti koje eksportuje modul. U slučaju programa ovde sledi telo programa. Zatim u oba slučaja može da dođe ključna reč `contains` koju slede definicije nekih procedura. U slučaju programa to su interne procedure, odn. one procedure koje nisu dovoljno opšte da bi bile izdvojene u neki modul u nadi da će biti korisne i nekom drugom programu; u slučaju

modula ovde su obično u pitanju procedure koje eksportuje modul tj. procedure čija su imena navedena na početku modula u `public` naredbama. Program odn. modul se završavaju sa `end program` odn. `end module` nakon čega se ponavlja ime programa odn. modula.

Za korišćenje nekog modula dovoljno je na početku programske jedinice koja koristi modul staviti `use` naredbu koju sledi ime modula koji se želi koristiti. Na ovaj način, svi simboli koji su u modulu eksportovani preko `public` naredbe postaju dostupni kodu u datoj programskoj jedinici. Naravno, potrebno je pri prevođenju obezbediti i da se data programska jedinica linkuje sa kodom modula.

Moduli su veoma korisni za dekompoziciju programa na više celina, a takođe bitno doprinose mogućnosti višestruke upotrebe jednom napisanog koda; zato se treba truditi da se što više koriste u *Fortran* programiranju. Logička organizacija u kojoj se kod sastoji od jednog programa i većeg broja modula se obično prenosi i na fizičku organizaciju, pa se tako obično svaki modul stavlja u poseban fajl, a program opet u odgovarajući fajl (ovakva organizacija je primenjena i na `libnumerics` biblioteku, gde je implementacija svake metode koja je implementirana na *Fortran*-u unutar posebnog modula i u zasebnom fajlu). Sledi primer jednostavnog modula koji sadrži proceduru za računanje korena kvadratne jednačine i programa koji koristi ovaj modul. Modul bi se mogao nalaziti u jednom fajlu sa sledećim sadržajem:

```
module square_roots_module
  implicit none
  public square_roots

contains
  subroutine square_roots(a, b, c, x0, x1)
    real, intent(in) :: a, b, c
    complex, intent(out) :: x0, x1
    complex :: discr

    discr=sqrt(b**2-4*a*c)
    x0=(-b+discr)/(2*a)
    x1=(-b-discr)/(2*a)
  end subroutine square_roots
end module square_roots_module
```

Program bi se nalazio u drugom fajlu i bio bi oblika:

```
program test_square_roots
  use square_roots_module
  implicit none

  real :: a, b, c
  complex :: x0, x1

  read (*,*) a, b, c
  call square_roots(a,b,c,x0,x1)
  write (*,*) x0, x1
end program test_square_roots
```

U gornjem programu su korišćene naredbe za čitanje sa standardnog ulaza odn. ispis na standardni ulaz koje su diskutovane u narednoj sekciji. Takođe, na kraju

poglavlja biće dat jedan veći primer iz biblioteke `libnumerics` koji takođe demonstrira upotrebu modula.

A.8 Ulaz i izlaz

Primarne komande za ulazno/izlazne operacije na *Fortran*-u su `read` i `write`. Sintaksa ovih naredbi je slična - nakon ključne reči u zagradama slede zarezima razdvojene specifikacije ulaznog odn. izlaznog fajla i formata, a potom dolazi zarezima razdvojena lista promenljivih u koje treba učitati vrednosti odn. lista vrednosti koje treba ispisati.

Narebom `open` moguće je pridružiti deskriptor nekom fajlu, kji onda može da se koristi u specifikaciju ulaznog odn. izlaznog fajla u `read` odn. `write` naredbama. Ukoliko se međutim čita sa standardnog ulaza, a piše na standardni izlaz, onda je dovoljno za specifikaciju fajla staviti `*`.

Specifikacija formata može biti navedena direktno ili pak može biti predstavljena labelom (za koje je već rečeno da su na *Fortran*-u brojevi) na kojoj se nalazi odgovarajuća `format` naredba. Pomoću `format` naredbe može se do u detalje (mogućnosti su slične onome što pružaju funkcije `scanf()` odn. `printf()` na *C*-u) kontrolisati oblik ulazno/izlaznih podataka, ali u većini slučajeva podrazumevani način rada `read` i `write` naredbi je sasvim dovoljan, tako da je najčešće i za format koristi specifikator `*`. Jedini izuzetak je kada treba učitati neki string, u kom slučaju za format treba navesti odgovarajući specifikator `'(A)'`; ovo stoga da bi se `read` naredbi naznačilo da ne treba blanko karaktere da smatra krajem tokena, što je inače način na koji `read` naredba funkcioniše.

Treba naglasiti da ispis nakon svake `write` naredbe prelazi u novi red. Da bi se ovo sprečilo, može se iza specifikacije formata staviti zarez, a onda kvalifikator `advance='no'`.

Ukoliko se radi sa fajlovima, na raspolaganju su i naredbe `rewind` za vraćanje na početak fajla, `close` za zatvaranje fajla, kao i još neke ređe korišćene naredbe.

Primer

Na kraju ovog dodatka biće prezentiran jedan kompletan primer iz `libnumerics` biblioteke. Radi se o metodi čija je implementacija na *C*-u već prezentirana u uvodnom poglavlju, dakle o metodi rešavanja sistema trodijagonalnih jednačina. Na osnovu ovoga što je prezentirano u ovom dodatku, kao i na osnovu komentara (a i imajući u vidu da je *Fotran* kod koji sledi potpuno ekvivalentan *C* kodu prezentiranom u prvom poglavlju ove zbirke), trebalo bi da je kod potpuno razumljiv; ako nešto ipak nije jasno, treba se vratiti naviše i ponovo proučiti odgovarajući segment sintakse *Fortran*-a. Fajl koji sadrži implementaciju metode (tj. proceduru `tridiag()` bi imao sledeći oblik:

```
module tridiag_module
  implicit none
  public tridiag
```

```
contains
  ! Procedura tridiag() resava trodijagonalni sistem jednacina oblika:
  !   a[i]*x[i-1]+c[i]*x[i]+b[i]*x[i+1]=d[i]
  ! gde je n>0, i=0,...,n-1, a[0]==0, b[n-1]==0 i c[i]!=0. Argumenti
  ! procedure su:
  !   n - dimenzija sistema
  !   a, b, c, d - polja sa koeficijentima sistema jednacina
  !   x - polje u koje ce biti smesteno resenje sistema
  ! Pretpostavlja se da su sva polja alocirana izvan procedure.
  subroutine tridiag (n, a, b, c, d, x)
    integer, intent(in) :: n
    double precision, dimension(0:n-1), intent(in) :: a
    double precision, dimension(0:n-1), intent(in) :: b
    double precision, dimension(0:n-1), intent(in) :: c
    double precision, dimension(0:n-1), intent(in) :: d
    double precision, dimension(0:n-1), intent(out) :: x
    double precision, dimension(0:n-1) :: alpha, beta ! Polja pomocnih
    ! koeficijenata.
    integer :: i ! Brojac u petljama

    ! Proverava se da li su ispunjeni uslovi sistema.
    if (n<=0) stop
    if (a(0)/=0 .or. b(n-1)/=0) stop
    do i=0, n-1
       if (c(i)==0) stop
    end do

    ! Resava se specijalni slucaj za n jednako 1.
    if (n==1) then
       x(0)=d(0)/c(0)
       return
    end if

    ! Racunaju se vrednosti pomocnih koeficijenata.
    alpha(1)=-b(0)/c(0)
    beta(1)=d(0)/c(0)
    do i=2, n-1
       alpha(i)=-b(i-1)/(a(i-1)*alpha(i-1)+c(i-1))
       beta(i)=(d(i-1)-a(i-1)*beta(i-1))/(a(i-1)*alpha(i-1)+c(i-1))
    end do

    ! Racunaju se resenja sistema.
    x(n-1)=(d(n-1)-a(n-1)*beta(n-1))/(a(n-1)*alpha(n-1)+c(n-1))
    do i=n-2, 0, -1
       x(i)=alpha(i+1)*x(i+1)+beta(i+1)
    end do
  end subroutine tridiag
end module tridiag_module
```

Fajl koji sadrži test program bi bio oblika:

```
! Program testira resavanje trodijagonalnog sistema jednacina.
Program cita ! postavku problema sa standardnog ulaza i ispisuje
rezultate na standardni ! izlaz. Na ulazu se prvo unosi dimenzija
sistema, a zatim red po red ! koeficijenti jednacina sistema a[i],
b[i], c[i] i d[i]. Na izlazu se u prvi ! red upisuje dimenzija
sistema, a zatim red po red resenja sistema. Program ! nema
ugra\dj enu obradu sintaksnih gresaka pri unosu problema. program
```

```
test
  use tridiag_module
  implicit none

  integer :: n ! Dimenzija sistema.
  double precision, dimension(:), allocatable :: a, b, c, d ! Polja sa
  ! koeficijentima sistema.
  double precision, dimension(:), allocatable :: x ! Polje sa resenjem
  ! sistema.
  integer :: i ! Brojac u petljama.

  ! Ucitava se dimenzija sistema.
  read (*,*) n

  ! Alociraju se polja sa koeficijentima sistema.
  allocate (a(0:n-1))
  allocate (b(0:n-1))
  allocate (c(0:n-1))
  allocate (d(0:n-1))

  ! Ucitavaju se koeficijenti sistema.
  do i=0, n-1
     read (*,*) a(i), b(i), c(i), d(i)
  end do

  ! Alocira se polje za resenje sistema.
  allocate (x(0:n-1))

  ! Resava se sistem.
  call tridiag (n, a, b, c, d, x)

  ! Osloba\dj aju se polja sa koeficijentima sistema.
  deallocate (a)
  deallocate (b)
  deallocate (c)
  deallocate (d)

  ! Ispisuje se resenje sistema.
  write (*,*) n
  do i=0, n-1
     write (*,*) x(i)
  end do

  ! Osloba\dj a se polje sa resenjem sistema.
  deallocate (x)
end program test
```

B

Matlab

MATLAB je profesionalni alat za tehnička izračunavanja. Ime MATLAB dolazi od termina *matrix laboratory* što znači matrična laboratorija. Sistem omogućava izvršavanje numeričkih proračuna, vizuelizaciju i daje mogućnost programiranja kroz okruženje koje se lako savladava i koristi. Najčešće primene uključuju matematičke proračune, razvoj algoritama, modeliranje, simuliranje i izradu prototipova, prihvatanje i analizu podataka, vizuelizaciju, razvoj aplikacija uključujući i izradu grafičkih korisničkih interfejsa.

MATLAB je interaktivni sistem koji omogućava da se problemi rešavaju mnogo brže i jednostavnije nego pomoću klasičnih programskih jezika kao što su npr. C ili FORTRAN. Neki od osnovnih razloga za ovo su sledeći:

- MATLAB uključuje programski jezik prilično visokog nivoa koji omogućava brzo i lako kodiranje
- Strukture podataka se jednostavno koriste. Osnovnu strukturu podataka čine višedimenzionalni dinamički nizovi koji ne zahtevaju deklarisanje i dimenzionisanje pre upotrebe. Ovo je naročito pogodno za izvođenje proračuna koji imaju vektorske, odnosno matrične formulacije.
- Interaktivni interfejs omogućava izuzetno jednostavno eksperimentisanje. Pronalaženje grešaka u programima, takođe, teče prilično jednostavno.
- MATLAB raspolaže mogućnostima prikazivanja veoma napredne grafike na jednostavan način.
- MATLAB datoteke (`m`-datoteke) su portabilne i mogu da se koriste na širokom spektru računarskih platformi.
- Veliki broj besplatnih `m`-datoteka je dostupan preko interneta
- Osnovno jezgro sistema nadograđuju brojni moduli koji sadrže gotove funkcije specifične za pojedinačne oblasti primene. Ovi moduli se obično nazivaju *toolboxes* tj. tulboksovi. Oblasti za koje postoje moduli su npr.

obrada signala, obrada slika, neuronske mreže, fazi logika, talasići, simulacija itd.

MATLAB je prvobitno napisan na jeziku FORTRAN kasnih sedamdesetih godina kao nastavno sredstvo i alat koji bi omogućio jednostavnije korišćenje softvera za manipulaciju matricama koji je razvijen u okviru LINPACK i EISPACK projekata. Danas, MATLAB obuhvata LAPACK i BLAS biblioteke, koje sadrže najsavremenije tehnike i algoritme za manipulaciju sa matricama. MATLAB je godinama menjan i unapređivan. 1984. počinje da se prodaje kao komercijalni proizvod koji je napisan na jeziku C. Verzija koja će u ovom tekstu biti opisana je verzija 6.5 (R13), mada je većinu teksta moguće primeniti i na starije verzije.

U akademskom okruženju, MATLAB se koristi kao standardno nastavno sredstvo u okviru kako uvodnih, tako i naprednih kurseva matematike i tehničkih disciplina. U industriji, MATLAB predstvlja odličan izbor za istraživanja, eksperimentisanja i brz razvoj efikasnih prototipskih aplikacija.

MATLAB sistem

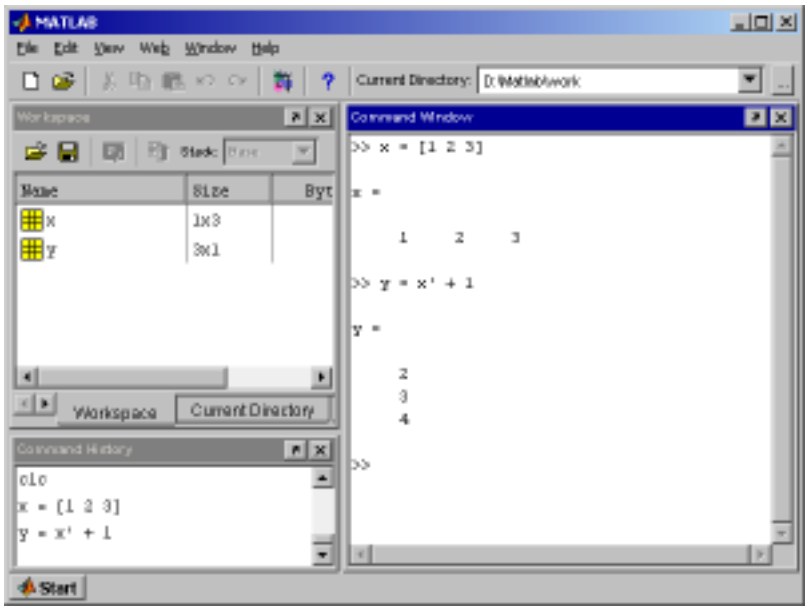
MATLAB sistem se sastoji od 5 glavnih delova:

- *Razvojno okruženje* - razvojno okruženje sačinjava skup alata koji omogućavaju jednostavno kreiranje, korišćenje i organizovanje MATLAB funkcija i datoteka. Većina ovih alata se jednostavno koristi kroz intuitivne grafičke interfejse. Među njima se ističu MATLAB komandni prozor, istorija korišćenih komandi, editor, dibager, kao i brauzer za čitanje helpa, prikaz korišćenih promenljivih, pretraga kroz datoteke i slično. Izgled MATLAB okruženja je prikazan na slici B.1.

- *Biblioteka matematičkih funkcija* - MATLAB raspolaže bogatom kolekcijom ugrađenih funkcija. Među njima se nalaze kako elementarne funkcije (npr. sumiranje, trigonometrijske funkcije, kompleksnu aritmetiku) tako i naprednije funkcije (npr. invertovanje matrica, pronalaženje sopstvene vrednosti, brza Fourier-ova transformacija, ...).

- *Programski jezik* - MATLAB poseduje programski jezik prilično visokog nivoa koji sadrži upravljačke iskaze, funkcije, strukture podataka, ulaz/izlaz, kao i neke odlike objektno-orijentisanih jezika. U ovom jeziku, moguće je formirati programe koji sežu od jednostavnih programa za jednokratnu upotrebu, do složenih, samostalnih, aplikacija namenjenih za intenzivnu i učestalu upotrebu.

- *Grafika* - MATLAB raspolaže mogućnostima grafičkog prikaza matematičkih objekata (funkcija, vektora, matrica, ...). Ovo obuhvata dvodimenizionalno i trodimenzionalno prikazivanje podataka, kreiranje, prikaz i obradu slika, kreiranje animacija itd.



B.1: MATLAB - razvojno okruženje

- MATLAB *aplikacije* - MATLAB poseduje tzv. MATLAB Application Program Interface (API). Ovaj interfejs omogućuje korisnicima da pišu C i FORTRAN rutine koje se mogu pozivati i koristiti iz MATLAB-a.

B.1 Matlab kao digitron

Format ispisa:	`format`, `;`
Operatori:	`+`, `-`, `*`, `/`, `^`
Elementarne funkcije:	`sqrt`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `exp`, `log`, `log10`, ...
	Detaljnije: `help elfun`
Predefinisane konstante:	`pi`, `Inf`, `NaN`, `eps`, `realmin`, `realmax`
Kompleksni brojevi:	`i` (ili `j`), `real`, `imag`, `abs`, `angle`, `conj`

MATLAB je moguće koristiti kao običan digitron. Da bi se izračunala vrednost aritmetičkog izraza dovoljno ga je jednostavno otkucati u komandnom prozoru,

```
>> 1+2*3                    >> 2^5/3
ans =                       ans =
      7                          10.6667
```

MATLAB direktno podržava sve aritmetičke operacije. Operatori se zapisuju uobičajeno (`+`, `-`, `*`, `/`, `^`) i imaju uobičajeni prioritet i asocijativnost.

Rezultat izraza se automatski beleži u specijalnu promenjivu `ans` koju je moguće koristiti u okviru daljih izračunavanja. Rezultat se automatski ispisuje na ekran. Ovo je moguće sprečiti navođenjem znaka `;` posle izraza, odnosno naredbe, čime se sprečava ispis rezultata u komandni prozor.

Promenljive

MATLAB dozvoljava dodelu vrednosti izraza promenljivim, za koje se ne zahteva prethodno deklarisanje. Promenljive, kojima je dodeljena vrednost, se mogu koristiti u okviru izraza. Npr.

```
>> x = 3 + 2;      % znak ; sprecava ispis rezultata
>> 2*x-1
ans =
     9
```

Spisak svih definisanih promenljivih se može dobiti komandom `who` (odnosno detaljnije komandom `whos`). Promenljiva koja više nije potrebna se može ukloniti komandom `clear`.

Zapis brojeva - IEEE754 standard

Primetimo da je rezultat izračunavanja iz prethodnog primera prikazan zaokružen na 4 decimale. Interno, svi rezultati se čuvaju i sva izračunavanja se vrše u dvostrukoj tačnosti prema *IEEE754 standardu*. Rezultat se zaokružuje samo prilikom prikazivanja, dok se interno čuva u tačnosti koju *IEEE754* dopušta (to je obično oko 16 značajnih cifara). Format prikaza rezultata je moguće izmeniti koristeći komandu `format`. Uobičajeni parametri ove komande su :

`short` - nepokretni zarez sa 5 cifara
`long` - nepokretni zarez sa 15 cifara
`rat` - aproksimacija u obliku razlomka

Pregled ostalih formata prikaza se može dobiti kucanjem `help format` u komandnom prozoru. [1]

Apsolutne vrednosti brojeva kojima MATLAB direktno može da manipuliše se nalaze u intevalu `[realmin, realmax]` koji je, otprilike, $[10^{-308}, 10^{308}]$. Relativna preciznost sa kojom MATLAB manipuliše se izražava konstantom `eps`, koja predstavlja rastojanje između broja 1.0 i prvog većeg broja u pokretnom zarezu koji može da se zapiše. Ova konstanta obično iznosi oko 10^{-16}.

MATLAB, kao i sam *IEEE754*, omogućava rad sa specijalnim vrednostima. Konstanta `Inf` označava ∞ (npr. rezultat deljenja 1/0 je `Inf`), dok konstanta `NaN` predstavlja *not-a-number*, odnosno vrednost koja matematički nije definisana (npr. rezultat deljenja 0/0 je `NaN`).

[1] Pomoć sa većinu komandi je moguće dobiti direktno u komandnom prozoru navođenjem `help` *komanda*

Celi brojevi

MATLAB ima nekoliko funkcija za rad sa celim brojevima. Osnovne su funkcije zokruživanja realnih brojeva i to

`round` - zaokruživanje na najbliži ceo broj
`floor` - zaokruživanje na najveći ceo broj manji od datog broja (ka $-\infty$)
`ceil` - zaokruživanje na najmanji ceo broj veći od datog broja (ka $+\infty$)
`fix` - zaokruživanje ka 0

Funkcija `mod` vrši celobrojno deljenje dva broja, dok funkcija `rem` određuje ostatak pri deljenju.

Kompleksni brojevi

MATLAB direktno podržava rad sa kompleksnim brojevima. Imaginarna jedinica i se označava sa `i` (ili alternativno sa `j`, ukoliko se `i` koristi kao promenljiva).

```
>> (2+i)/(2+4*i)
ans =
   0.4000 - 0.3000i
```

Osnovne funkcije za rad sa kompleksnim brojevima su

`real` - izdvaja realni deo kompleksnog broja
`imag` - izdvaja imaginarni deo kompleksnog broja
`abs` - izračunava moduo kompleksnog broja
`angle` - izračunava argument - ugao φ za koji je $z = |z|e^{i\varphi}$, $\varphi \in (-\pi, \pi]$
`conj` - izračunava kompleksno-konjugovanu vrednost

Elementarne funkcije i konstante

MATLAB direktno podržava široki spektar elementarnih funkcija, kao i nekoliko često korišćenih matematičkih konstanti. Npr. vrednost konstante π se dobija kao `pi`. Uzevši ovo u obzir, i imajući u vidu da se trigonometrijska funkcija sinus dobija kao `sin`, vrednost $\sin(\frac{\pi}{4})$ se može izračunati kao

```
>> sin(pi/4)
ans =
    0.7071
```

Spisak svih elementarnih funkcija koje MATLAB podržava je moguće dobiti navođenjem komande `help elfun` u komandnom prozoru.

Većina funkcija je definisana i za kompleksne brojeve. Npr.

```
>> sqrt(-1)
ans =
        0 + 1.0000i
```

```
>> exp(pi*i)
ans =
  -1.0000 + 0.0000i
```

Unos matrica:	Npr. [a_{11}, a_{12}, a_{13}; a_{21}, a_{22}, a_{23}]
Specijalne matrice:	`zeros`, `ones`, `eye`, `diag`, `rand`, `randn`, `linspace`, `logspace`, ...
Indeksiranje:	Npr. `A(v, k)` - a_{vk}, `A(i, :)` - i-ta vrsta, `A(:, i)` - i-ta kolona
Operatori:	`+`, `-`, `*`, `'`, `.*`, `./`, `.^`
Relacije:	`isequal`, `==`, `~=`, `<`, `<=`, `>`, `>=`
	`&`, `|`, `all`, `any`, `find`,...
Funkcije:	`length`, `size`, `min`, `max`, `sum`, `prod`, `sort`, `diff` ...
	Detaljnije: `help elmat`

B.2 Vektori i matrice

Osnovnu strukturu podataka u MATLAB-u čine matrice i vektori. Matrice u MATLAB-u predstavljaju elementarni tip podataka, i sa njima se može direktno manipulisati. Moguće je direktno dodeljivati matrice promenljivim, koristiti ih kao argumente i rezultate funkcija i slično. Većina ugrađenih funkcija operiše sa matricama, bilo direktno sa celim matricama, bilo nad njihovim kolonoma, ili pojedinačnim elementima.

Pored uobičajenih dvodimenzionalnih matrica, MATLAB odlikuje i mogućnost korišćenja višedimenzionalnih matričnih struktura.

Vektori

MATLAB ne poseduje poseban tip za zapis vektora, već se vektori predstavljaju matricama čija je jedna dimenzija 1. Uostalom, i skalari odnosno brojevi, o kojima je bilo ranije reči, su predstavljeni matricama dimenzija 1×1.

Vektori se u MATLAB-u navode tako što se njihovi elementi navedu unutar uglastih zagrada (eventualno odvojeni zapetama). Npr.

```
>> a = [1 2 3]                          >> b = [4, 5, 6]
a =                                     b =
     1     2     3                            4     5     6
```

Veoma važan operator kojim mogu da se kreiraju vektori je operator `:`. Operator se najčešće koristi u binarnoj varijanti `a:b`, koja kreira vektor `[a, a+1, ..., b]`. Iako `a` i `b` mogu biti proizvoljni realni skalari, najčešći slučaj je da su celi brojevi.

```
>> 1:5
ans =
     1     2     3     4     5

>> 1.5:7
ans =
    1.5000    2.5000    3.5000    4.5000    5.5000    6.5000
```

Ternarna varijanta operatora `:` ima oblik `a:c:b` i kreira vektor brojeva između `a` i `b` sa korakom `c`. Npr.

```
>> 1:2:20
ans =
     1     3     5     7     9    11    13    15    17    19
```

Umesto ovog operatora, često se koristi i funkcija `linspace(a, b, n)` koja gradi ekvidistantnu mrežu intervala $[a, b]$ sa n čvorova. Npr.

```
>> linspace(0, 1, 5)
ans =
         0    0.2500    0.5000    0.7500    1.0000
```

Vektori se mogu sabirati i oduzimati korišćenjem operatora `+` i `-`. Operatori `.*`, `./` i `.^` služe za izvođenje aritmetičkih operacija "pokoordinatno". Npr.

```
>> [1 2 3 4].*[5 6 7 8]
ans =
     5    12    21    32
```

```
>> [1 2 3 4].^2
ans =
     1     4     9    16
```

Operatori `+`, `-`, `*` i `/` se mogu koristiti i u obliku u kome je jedan operand vektor, a drugi operand skalar. U tom slučaju se operacija primenjuje na svaki element vektora pojedinačno.

```
>> [1 2 3]*3
ans =
     3     6     9
```

MATLAB poseduje mnoštvo ugrađenih funkcija za rad sa vektorima. Samo neke od najviše korišćenih su:

`length` - izračunava dužinu vektora
`sum` - izračunava sumu elemenata vektora
`prod` - izračunava proizvod elemenata vektora
`min` - izračunava najmanji element vektora
`max` - izračunava najveći element vektora
`sort` - sortira elemente vektora u rastući niz
`diff` - izračunava konačne razlike vektora (razlike susednih elemenata)

```
>> v = [1 3 8 9 2 1 0];
>> diff(v)
ans =
     2     5     1    -7    -1    -1

>> sort(v)
ans =
     0     1     1     2     3     8     9
```

Indeksiranje vektora

Pojedinačnim elementima vektora se može pristupiti korišćenjem indeksiranja. Indeksi se navode u okviru zagrada `( )`. Važno je naglasiti da indeksi vektora u MATLAB-u kreću od 1, za razliku od npr. jezika C gde indeksi kreću od 0. Da bi se pristupilo elementu vektora, potrebno je da indeks bude manji ili jednak dimenziji vektora.

```
>> v = [1 3 8 9 2 1 0];
>> v(3)
ans =
     8

>> v(8)
??? Index exceeds matrix dimensions.
```

Sa druge strane, moguće je menjati odgovarajuće elemente vektora korišćenjem indeksnog pristupa. U ovom slučaju, moguće je da indeks premašuje dužinu vektora. Tada se vrži redimenzionisanje vektora, a nedefinisani elementi se popunjavaju nulama.

Naglasimo da je redimenzionisanje relativno "skupa" operacija i poželjno ju je izbegavati ukoliko je to moguće kako bi se dobilo na efikasnosti. Ukoliko je unapred poznata konačna dimenzija vektora `v` npr. n, poželjno je odmah dimenzionisati vektor `v` na dimenziju n npr. dodeljivanjem vektora koji se sastoji od n nula, a zatim postupno menjati jedan po jedan njegov element. Ova tehnika se zove *prealokacija* vektora i uz vektorizaciju o kojoj će kasnije biti reči predstavlja osnovnu tehniku poboljšanja efikasnosti koda.

```
>> v = [1 2 3 4 5];
>> v(3) = 1
v =
     1     2     1     4     5

>> v(10) = 2
v =
     1     2     1     4     5     0     0     0     0     2
```

Vektor je moguće indeksirati i drugim vektorom i rezultat je tada vektor elemenata. Ovakvo indeksiranje se izvodi na osnovu pravila

$$\texttt{v([a}_1\texttt{,...,a}_k\texttt{]) = [v(a}_1\texttt{),...,v(a}_k\texttt{)]}$$

Najčešći oblik upotrebe ovoga je izdvajanje pojedinačnih elemenata vektora, odnosno izdvajanje pojedinih opsega vektora. Ovakvo indeksiranje je moguće koristiti i za promenu određenih delova vektora. Npr.

```
>> v = [5 6 7 8 9];
>> v([2 4])
ans =
     6     8

>> v(2:4) = [1 2 3]
v =
     5     1     2     3     9
```

Ukoliko je potrebno izdvojiti sve elemente vektora, počevši od neke pozicije pa sve do kraja, moguće je koristiti ugrađenu promenljivu `end` koja uvek sadrži indeks poslednjeg elementa vektora. Npr.

```
>> v = [9 8 7 6 5 4 3 2 1 0];
>> v(3:end)             % svi sem prva dva elementa vektora
ans =
     7     6     5     4     3     2     1     0
>> v(end-2:end)         % poslednja tri elementa vektora
ans =
     2     1     0
```

Ukoliko se izostave oba operanda operatora `:`, podrazumeva se da su u pitanju svi elementi vektora tj. vrednost `1:end`.

Logički i relacijski operatori

MATLAB poseduje većinu standardnih relacijskih operatora, međutim, osnovna razlika u odnosu na većinu programskih jezika je to što se ovi operatori primenjuju pokoordinatno na vektore, odnosno matrice istih dimenizija. Rezultati primena ovih relacija su vektori logičkih vrednosti, koje se u MATLAB-u predstavljaju preko brojevnih vrednosti i to 0 za netačno i 1 za tačno. Za poređenje da li su dve matrice odnosno dva vektora međusobno jednaki, koristi se funkcija `isequal`.

```
>> [1 3 5 7] == [0 3 6 9]          >>isequal([1 3 5 7], [0 3 6 9])
ans =                              ans =
     0     1     0     0                 0
```

Od logičkih operatora, MATLAB razlikuje konjunkciju `&`, disjunkciju `|` i negaciju `~`. Funkcije `all` i `any` imaju ulogu univerzalnog, odnosno egzistencijalnog kvantifikatora i određuju da li su svi elementi vektora različiti od nule, odnosno da li postoji element vektora koji je različit od nule. U ovu grupu funkcija, ubrojaćemo i funkciju `find` koja vraća sve indekse vektora koji su različiti od nule.

U kombinaciji sa relacijskim operatorima, dobija se prilično izražajan mehanizam. Npr.

```
>> v = [3 4 5 6 7];
>> any(v>7)           % da li v sadrzi vrednost vecu od 7
ans =
     0

>> all(0<=v & v<=10)  % da li svi elementi vektora leze u intervalu [0, 10]
ans =
     1

>> find(v<6)          % Na kojim pozicijama se nalaze elementi manji od 6
ans =
     1     2     3
```

Više o relacijskim i logičkim operacijama se može saznati komandom `help relop` u komandnom prozoru MATLAB-a.

Korišćenjem funkcije `find`, u kombinaciji sa indeksiranjem vektora drugim vektorom, postiže se izdvajanje elemenata vektora koji imaju određeno svojstvo. Kako bi se ova operacija optimizovala, MATLAB uvodi tzv. *logičko indeksiranje* koje omogućuje da se preskoči poziv funkcije `find`, i da se vektor direktno indeksira vektorom logičkih vrednosti iste dimenzije. Kao rezultat, dobija se novi vektor

koji sadrži elemente polaznog vektora za koje je indeksni vektor na odgovarajućim pozicijama sadržao logičku vrednost tačno tj. 1. Npr.

```
>> v = [3 4 5 6 7 1 2];
>> v(find(v<6))     % izdvajamo elemente vektora strogo manje od 6
ans =
     3     4     5     1     2

>>v(v<6)            % ekvivalentno prethodnom, ali optimalnije
ans =
     3     4     5     1     2
```

Matrice

Matrice se u MATLAB-u navode tako što se njihove vrste navedu unutar uglastih zagrada, međusobno odvojene simbolom ;. Elementi svake vrste mogu eventualno biti odvojeni zapetama. Neophodno je da sve vrste imaju jednak broj elemenata. Npr.

```
>> a = [1 2 3; 4 5 6]
a =
     1     2     3
     4     5     6
```

MATLAB na neki način omogućava i rad sa blok matricama. Ovo je podržano činjenicom da se matrice mogu kreirati od drugih matrica, a ne isključivo od skalara. Ukoliko su dimenzije gradivnih matrica odgovarajuće, vrši se njihovo stapanje u jednu veću matricu. Npr.

```
>> A = [1 2; 3 4];
>> [A A; A A]
ans =
        1  2  1  2
        3  4  3  4
        1  2  1  2
        3  4  3  4
```

Postoji nekoliko funkcija za generisanje specifičnih matrica i ovde ih navodimo:

`zeros`	(`m`, `n`)	gradi nula matricu dimenzije `m`×`n`
`ones`	(`m`, `n`)	gradi matricu jedinica dimenzije `m`×`n`
`eye`	(`n`)	gradi jediničnu matricu dimenzije `n`×`n`
`rand`	(`m`, `n`)	gradi matricu slučajnih brojeva dimenzije `m`×`n` brojevi imaju uniformnu raspodelu na $[0, 1]$.
`randn`	(`m`, `n`)	gradi matricu slučajnih brojeva dimenzije `m`×`n` brojevi imaju normalnu raspodelu $\mathcal{N}(0, 1)$.
`diag`	(`v`)	gradi dijagonalnu matricu sa vektorom `v` na dijagonali

Neke od matrica koje su sa matematičkog stanovišta zanimljive iz nekog razloga se mogu dobiti korišćenjem funkcije `gallery`. Za detalje pogledati dokumentaciju ove funkcije. Pomenimo još i funkcije `hilb`, `pascal`, `toeplitz` i mnoge slične koje služe za dobijanje čuvenih tipova matrica.

Većina funkcija i operatora koji su bili opisani u poglavlju o vektorima, mogu da se primene i na matrice.

Operatori `+`, `-`, `.*`, `./` i `.^` predstavljaju osnovne aritmetičke operacije i primenjuju se pokoordinatno. Operator `*` predstavlja uobičajeno množenje dve matrice. Da bi moglo da se primeni množenje matrica, one moraju biti odgovarajućih dimenzija.

```
>> [1 2 3; 4 5 6]*[1 2; 3 4; 5 6]
ans =
    22    28
    49    64
```

Operator `'` vrši transponovanje realnih matrica. Ukoliko matrica sadrži i kompleksne brojeve treba biti obazriv, pošto se u tom slučaju vrši konjugovanje, a ne samo transponovanje. Da bi se kompleksna matrica transponovala, potrebno je koristiti operator `.'`.

```
>> R = [1 2; 3 4; 5 6];
>> R'
ans =
     1     3     5
     2     4     6

>> C = [1+i 2+2*i; 3+3*i 4+4*i; 5+5*i 6+6*i];
>> C'
ans =
   1.0000 - 1.0000i   3.0000 - 3.0000i   5.0000 - 5.0000i
   2.0000 - 2.0000i   4.0000 - 4.0000i   6.0000 - 6.0000i
```

Sve elementarne funkcije (`sin`, `exp`, ...) se na matrice primenjuju na svaki element matrice pojedinačno (pokoordinatno). Npr.

```
>> sin([0 pi/2; pi/4 pi])
ans =
         0    1.0000
    0.7071    0.0000
```

Funkcije poput `max`, `sum`, `sort`, `diff`, itd. se primenjuju na svaku kolonu matrice pojedinačno i rezultati njihove primene su vektori, a ne skalari. Oko ovoga treba biti obazriv, i obično je potrebno dva puta pozvati neku funkciju kako bi se dobio krajnji rezultat. Npr.

```
>> a = [1 2 3; 9 8 7; 5 6 4]
a =
     1     2     3
     9     8     7
     5     6     4

>> sum(a)                          >> sum(sum(a))
ans =                              ans =
    15    16    14                     45
```

Indeksiranje matrica

Najčešći oblik indeksiranja matrica je indeksiranje putem dvostrukih indeksa navedenih između malih zagrada `( )`. Prvi indeks predstavlja indeks vrste, dok drugi

indeks predstavlja indeks kolone. Kao i u slučaju vektora, indeksi mogu biti bilo skalari, bilo vektori. U slučaju da su indeksi vektori, rezultat je nova matrica. Elementi vektorskih indeksa se kombinuju "svaki sa svakim", odnosno važi

```
A([i1 ... in],[j1 ... jm])  =  [A(i1,j1) ... A(i1,jm)  ;
                                          ...           ;
                               A(in,j1) ... A(in,jm)]
```

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(2,3)
ans =
     6

>> A([1 2], [1 3])
ans =
     1     3
     4     6
```

Ukoliko se želi izdvojiti pojedina vrsta, odnosno kolona matrice, moguće je koristiti posebnu varijantu operatora `:` u kojoj su izostavljeni operandi. U ovom slučaju podrazumeva se vrednost `1:end` koja izdvaja sve elemente. Tako bi se prva vrsta izdvojila korišćenjem `A(1,:)`, dok bi se treća kolona izdvajala korišćenjem `A(:, 3)`.

Kao i u slučaju vektora, da bi se pročitale vrednosti određenih elemenata matrice, indeksi moraju biti celi brojevi koji su između 1 i odgovarajuće dimenzije matrice. Sa druge strane, izdvojenom opsegu je moguće dodeliti vrednost i tada se prethodno ograničenje ne poštuje već se matrica automatski proširuje uz dopunjavanje nulama. Kao što je već rečeno, realokacija je neefikasna operacija koju bi trebalo izbegavati.

Pored ovoga, moguće je indeksirati matrice i samo jednim indeksom. U ovom slučaju, elementi se nabrajaju po kolonama. Npr.

$$\begin{bmatrix} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \end{bmatrix}$$

I za ovaj jedan indeks važi da može da bude bilo skalar bilo vektor. Prilikom ovakvog indeksiranja, matrična struktura se ne zadržava, već se kao rezultat dobija vektor traženih elemenata. Čest slučaj upotrebe je oblik `A(:)` koja od matrice `A` gradi vektor njenih elemenata.

B.3 Ostale strukture podataka

Višedimenzioni nizovi

MATLAB nudi i mogućnost kreiranja višedimenzionih nizova (matrica). Npr. trodimenzionu matricu koja u jednom sloju ima magični kvadrat, a u drugom Pascalovu matricu je moguće kreirati preko:

```
>> A(:, :, 1) = magic(3);
>> A(:, :, 2) = pascal(3)
A(:,:,1) =
     8     1     6
     3     5     7
     4     9     2
A(:,:,2) =
     1     1     1
     1     2     3
     1     3     6
```

Ćelijski nizovi

Ćelijski nizovi (Cell arrays) su strukture podataka koje poseduju nizovske karakteristike i koje kao svoje elemente mogu da sadrže proizvoljne strukture podataka uključujući brojeve, vektore, matrice, pa čak i druge ćelijske nizove. Ćelijski nizovi se kreiraju na način sličan kreiranju matrica, pri čemu se pojedinačni elementi navode između vitičastih `{}` umesto uglastih zagrada `[]`. Pravila indeksiranja su veoma slična indeksiranju običnih vektora i matrica, pri čemu se za indeksiranje opet koriste vitičaste zagrade. Npr.

```
>> A = {1, 1:3; eye(2), 'Zdravo'}
A =
    [1]             [1x3 double]
    [2x2 double]    'Zdravo'

>> A{2, 1}
ans =
    1   0
    0   1
```

Strukture

Strukture odgovaraju slogovima koji se javljaju u većini viših programskih jezika. Slično ćelijskim nizovima i strukture omogućuju grupisanje heterogenih podataka u jednu celinu. Osnovna razlika je u tome što strukture ne poseduju nizovske osobine već se pojedinačnim elementima pristupa preko odabranog imena. Npr.

```
>> A.name = 'Identity';
>> A.dimension = 3;
>> A.matrix = eye(3);

>> A.matrix
ans =
    1   0   0
    0   1   0
    0   0   1
```

B.4 Polinomi

Polinomi se u MATLAB-u predstavljaju preko vektora svojih koeficijenata, pri čemu se koeficijent uz najveći stepen nalazi na prvoj poziciji vektora, a slobodni član na

poslednjoj. Npr. polinom $3x^2 + 2x + 1$ se predstavlja vektorom `[3, 2, 1]`.

Polinomi se mogu direktno sabirati, odnosno oduzimati, pod pretpostavkom da su vektori kojima su predstavljeni iste dimenzije (u slučaju da nisu, uvek je moguće kraći vektor proširiti vodećim nulama do dimenzije većeg). Za množenje polinoma može se koristiti funkcija `conv`. Slično, za deljenje dva polinoma, moguće je koristiti funkciju `deconv`, koja vraća količnik i ostatak pri deljenju. Naglasimo, da se za dodavanje broja `a` polinomu `p` ne sme koristiti konstrukcija `p = p + a`, jer podsetimo se, ovo dodaje broj `a` svakom koeficijentu polinoma `p`. Umesto ovoga, potrebno je koristiti `p(end) = p(end) + a`.

Za rad sa polinomima, izgrađen je skup efikasnih funkcija.

`polyval`	izračunava vrednost polinoma u datoj tački.
`roots`	pronalazi sve nule datog polinoma.
`poly`	ukoliko joj je predat vektor `x`, konstruiše moničan polinom čije su nule elementi vektora `x`, odnosno polinom $\prod_{i=1}^{n}(x - x_i) = (x - x_1) \cdot \ldots \cdot (x - x_n)$
`polyder`	pronalazi izvod polinoma
`polyint`	pronalazi neodređeni integral polinoma

B.5 Grafika

Funkcije:	`plot`, `fplot`, `stem`, `mesh`, `surf`, `contour`, `meshgrid`, ...
Simboličke funkcije:	`ezplot`, `ezsurf`, ...
Kontrola prozora iscrtavanja:	`hold on`, `hold of`, `figure`, `gcf`, `subplot`, `axis equal`, ...

MATLAB raspolaže prilično bogatom bibliotekom za vizuelizaciju i grafičko predstavljanje matematičkih objekata. Osnovna podela ovih rutina je na dvodimenizonalne (vizuelizacija u ravni) i trodimenzionalne (vizuelizacija u prostoru).

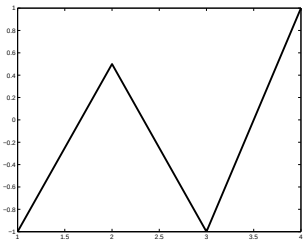
2D vizuelizacija

Najjednostavnija funkcija za dvodimenzionalnu vizuelizaciju je funkcija `plot`. Osnovni oblik korišćenja ove funkcije je `plot([x1,...,xn],[y1,...,yn])`. Ovim se iscrtava izlomljena linija $(x_1, y_1), \ldots, (x_n, y_n)$. Primer je dat na slici B.2.

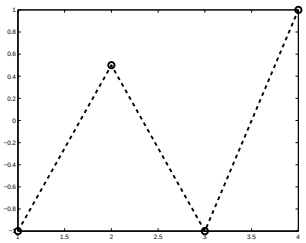
Funkcija se odlikuje mnoštvom opcija koje kontrolišu izgled crteža. Ove opcije se navode preko trećeg parametra koji je niska karaktera navedenih između jednostrukih navodnika `'`. U okviru ove niske moguće je istovremeno navesti i nekoliko različitih opcija. Neke od najviše korišćenih su:

`r`, `g`, `b`, ...	boja kojom se crta (crvena, zelena, plava, ...)
`.`, `o`, `x`, `+`, ...	tačke se obeležavaju posebnim markerima
`-`, `:`, `--`, ...	tačke se spajaju punom, tačkastom, odnosno isprekidanom linijom

Ukoliko se želi iscrtavanje dela realne funkcije $f : \mathbb{R} \mapsto \mathbb{R}$ na određenom intervalu $[a, b]$, potrebno je tabelirati na dovoljno gustoj podeli domena i zatim iscrtati njenu aproksimaciju izlomljenom linijom. Npr. za crtanje dela funkcije $\sin(x)$ na intervalu $[0, \pi]$ se koristi



plot([1 2 3 4], [-1 0.5 -1 1])



plot([1 2 3 4], [-1 0.5 -1 1], 'ro--')

B.2: Upotreba funkcije `plot`

```
>> x = linspace(0, 2*pi);
>> plot(x, sin(x))
```

Pored ovoga, postoji i funkcija `fplot` koja direktno iscrtava datu funkciju na datom intervalu.

```
>> fplot('sin(x)', [0 2*pi])
```

Toolbox za simbolička izračunavanja (*Symbolic toolbox*) sadrži funkciju `ezplot` koja se može koristiti za iscrtavanje funkcija (često i implicitno zadatih). Npr.

```
>> ezplot('x^2+y^2=1',[-1 1],[-1 1])
```

Kontrola prozora za iscrtavanje

Prilikom prvog poziva neke funkcije za iscrtavanje, otvara se novi prozor i slika se pojavljuje u njemu. Prilikom svakog sledećeg iscrtavanja, nova slika zamenjuje staru. Ukoliko se u komandnoj liniji unese komanda `hold on`, stare slike bivaju zadržane, i nove se iscrtavaju preko njih. Na prvobitno ponašanje se može vratiti komandom `hold off`. Da bi se slika pojavila u novom prozoru, a da stari prozor zadrži svoj sadržaj, potrebno je uneti komandu `figure`. Bez parametara, ova komanda prouzrokuje otvaranje novog prozora koji postaje tekući prozor za iscrtavanje. Ukoliko se kao parametar zada broj prozora, `figure` proglašava prozor sa datim brojem za tekući i svako naredno iscrtavanje će se dešavati u njemu. Broj tekućeg prozora je moguće saznati korišćenjem komande `gcf`.

Moguće je, takođe, postići da jedan prozor istovremeno sadrži više koordinatnih osa. Za detalje, pogledati dokumentaciju funkcije `subplot`.

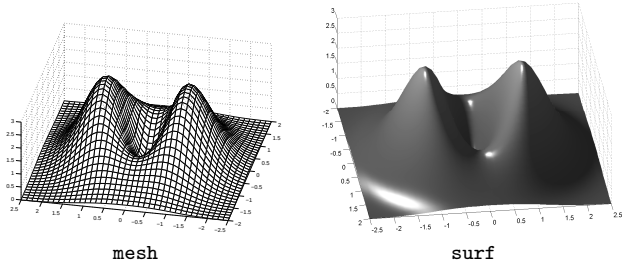
Svaki prozor za crtanje sadrži niz kontrola korisničkog interfejsa koje omogućuju izmenu detalja prikaza. Za većinu od ovih izmena postoje i odgovarajuće komande jezika kojima se postiže isti efekat. Kao primer navedimo komandu `axis equal` kojom se postiže proprocionalnost koordinatnih osa, čime se dobija da npr. krug liči na krug umesto na elipsu.

3D vizuelizacija

Iako MATLAB poseduje jako veliki broj različitih funkcija za 3d vizuelizaciju, u ovom tekstu će biti prikazane samo neke od najosnovnijih.

Ukoliko želimo vizuelizovati deo realne funkcije $f : \mathbb{R}^2 \mapsto \mathbb{R}$ na intervalu $[a_x, b_x] \times [a_y, b_y]$, potrebno ju je tabelirati na diskretnoj mreži koja predstavlja podelu tog intervala. Ova podela se predstavlja sa dve matrice `X` i `Y`, dok se vrednosti funkcije predstavljaju trećom matricom `z`. Funkcija `meshgrid` kreira matrice `X` i `Y` na osnovu datih vektora koji predstavljaju podele intervala $[a_x, b_x]$ i $[a_y, b_y]$. Na osnovu konstruisanih matrica `X` i `Y`, matrica `Z` se konstruiše koristeći pokoordinatne matrične operatore i elementarne funkcije koje, podsetimo se, rade pokoordinatno. Npr. funkciju $(x^2+3y^2)e^{1-x^2-y^2}$ na intervalu $[-2, 2] \times [-2.5, 2.5]$ je moguće iscrtati korišćenjem

```
>> x = linspace(-2, 2, 40);
>> y = linspace(-2.5, 2.5, 40);
>> [X, Y] = meshgrid(x, y);
>> Z = (X.^2+3*Y.^2).*exp(1-X.^2-Y.^2);
>> mesh(X, Y, Z)
```



B.3: Upotreba funkcija `mesh` i `surf`

Ukoliko se umesto mreže želi iscrtavanje površi, potrebno je upotrebiti funkciju `surf` umesto funkcije `mesh`. U sledećem primeru je pored samog iscrtavanja prilagođeno nekoliko osnovnih grafičkih parametara kako bi slika izgledala plastičnije. Ove parametre je, takođe, moguće menjati i preko kontrola grafičkog korisničkog interfejsa prozora koji sadrži sliku. Za detalje pogledati dokumentaciju.

```
>> surf(X, Y, Z)
>> camlight right, lighting phong, shading interp
```

Rezultati ovih iscrtavanja su dati na slici B.3.

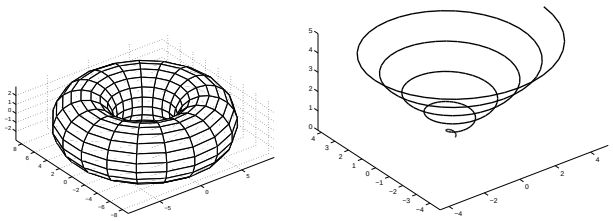
Trodimenzionalne površi koje se često vizuelizuju se obično zadaju putem parametrizacije. Npr. paramtarske jednačine

$$\begin{aligned} x &= (a + b\cos(\theta))\cos(\varphi) \\ y &= (a + b\cos(\theta))\sin(\varphi) \\ z &= b\sin(\theta), \qquad\qquad \theta \in [0, 2\pi), \varphi \in [-\pi, \pi) \end{aligned}$$

određuju torus. Ovaj torus je moguće vizuelizovati preko

```
>> phi = linspace(0, 2*pi, 20);
>> theta = linspace(-pi, pi, 20);
>> [PHI, THETA] = meshgrid(phi, theta);
>> X = (6+3*cos(THETA)).*cos(PHI);
>> Y = (6+3*cos(THETA)).*sin(PHI);
>> Z = 3*sin(THETA);
>> mesh(X, Y, Z), axis equal
```

Rezultat je prikazan na slici B.4.



B.4: Parametarski zadata kriva i površ

Pored ovoga, moguće je vizuelizovati i krive u prostoru koristeći funkciju `plot3`. Ova funkcija je veoma slična funkciji `plot`, osim što joj se prosleđuju tri vektora koji predstavljaju x, y i z koordinate tačaka u prostoru. Npr. zavojnica na slici B.4 koja ima parametrizaciju

$$\begin{aligned} x &= h\cos(\varphi) \\ y &= h\sin(\varphi) \\ z &= h \qquad\qquad \varphi \in [0, 5\pi], h \in [0, 5] \end{aligned}$$

je iscrtana putem

```
>> phi = linspace(0, 10*pi, 200);
>> h = linspace(0, 5, 200);
>> x = h.*cos(phi);
>> y = h.*sin(phi);
>> z = h;
>> plot3(x, y, z), axis equal
```

B.6 Kontrola toka izvršavanja

MATLAB programski jezik poseduje naredbe kontrole toka uobičajene za većinu programskih jezika. Uslovno grananje se postiže korišćenjem naredbe `if` koja u svom najjednostavnijem obliku glasi

```
if <logicki uslov>
      <naredbe>
end
```

Kontrola toka:	`if`, `elseif`, `else`, `end`
	`switch`, `case`, `otherwise`, `end`
	`for`, `end`
	`while`, `end`
	`break`

Ukoliko je logički uslov ispunjen, izvršavaju se sve naredbe do naredbe `end`.

Pored ovoga, uslovno grananje je moguće koristiti u svom punom obliku

```
if <logicki uslov>
   <naredbe>
elseif <logicki uslov>
   <naredbe>
else
    <naredbe>
end
```

Takođe, postoji naredba višestrukog grananja `switch`. Njen pun oblik je

```
switch <izraz>
    case <vrednost1>
        <naredbe>
    case <vrednost2>
        <naredbe>
    ...
    otherwise
        <naredbe>
end
```

Iteracija se postiže korišćenjem naredbe `for` i to u obliku

```
for promenljiva = vektor
    <naredbe>
end
```

Prilikom iteracije, promenljiva uzima redom jedan po jedan element navedenog vektora. Ovo se veoma često koristi u kombinaciji sa izgradnjom vektora korišćenjem operatora `:`. Npr. faktorijel broja 10 se može izračunati putem

```
>> p = 1;
>> for i = 1:10
      p = p*i;
   end
>> p
p =
     3628800
```

Naglasimo da je, s obzirom da je MATLAB interpreterski jezik, izvršavanje petlji prilično neefikasno. Umesto njih, poželjno je koristiti ugrađene funkcije, što je češće

moguće. Tako se npr. 10 faktorijel mnogo brže izračunava preko `prod(1:10)`. Iako i funkcija za izračunavanje proizvoda takođe vrši iteraciju, ta iteracija je ostvarena u kompajliranom segmentu C koda i samim tim je mnogo brža. Postupak eliminacije petlji je poznat pod imenom *vektorizacija*.

Iteraciju je moguće izvršiti i korišćenjem `while` petlje u obliku

```
while <logicki uslov>
    <naredbe>
end
```

Tako npr. sledeći kod pronalazi presek krivih e^{-x} i $\sin(x)$, korišćenjem Newton-ove metode tangente sa tačnošću 10^{-5} i maksimalno 100 iteracija.

```
>> x0 = 0.5; x = x0;
>> n = 0;
>> while (abs(x - x0) > 1e-5 | n==0) & n < 100
       n = n + 1;
       x0 = x;
       x = x - (exp(-x)-sin(x))/(-exp(-x)-cos(x));
   end
```

Ispitivanjem vrednosti promenljive `n` po završetku petlje, vidi se da je tačnost postignuta već u 3 iteracije.

Naredba `break` eksplicitno prekida iteraciju.

B.7 m-datoteke - skriptovi i funkcije

m-datoteke:	`edit`, `cd`, `pathtool`
Funkcije:	`function [o`$_1$`,...,o`$_n$`] = name (i`$_1$`,...,i`$_m$`)`

Iako se puno stvari može postići direktnim radom iz MATLAB komandnog prozora, ozbiljniji zadaci obično zahtevaju izdvajanje zasebnih funkcionalnosti u potprograme. MATLAB razlikuje dve vrste potprograma i to *skriptove* i *funkcije*. Osnovne razlike između njih bi mogle da se rezmiraju kroz tabelu B.5

Svaki MATLAB skript odnosno funkcija se smešta u zasebnu datoteku koja ima ekstenziju `.m` (ovo je razlog zbog čega se ove datoteke obično nazivaju `m`-datotekama, odnosno `m`-fajlovima).

`m`-datoteke su obične tekstualne datoteke koje mogu biti kreirane bilo kojim tekst editorom. Sistem MATLAB poseduje specijalizovani editor koji je moguće pokrenuti komandom `edit`. Pošto ovaj editor poseduje integrisane elemente okruženja (npr. direktno pokretanje datoteke, integraciju sa dibagerom), uobičajeno je njegovo korišćenje.

`m`-datoteke mogu biti snimljene u proizvoljnom direktorijumu računara, međutim, da bi MATLAB mogao da pronađe odgovarajuću datoteku, potrebno je da dirketorijum bude uvršten u tzv. stazu pretrage (*search path*). Ovo se radi korišćenjem

Skriptovi	*Funkcije*
Ne prihvataju ulazne parametre i ne vraćaju rezultat	Mogu da prihvate ulazne argumente i da vrate rezultate kroz izlazne argumente
Ne poseduju poseban memorijski prostor već rade nad podacima zajedničkim za rad iz komandne linije i sve skriptove	Koriste poseban memorijski prostor i imaju lokalne promenljive
Koriste se za automatizovanje niza koraka koji se često izvršavaju zajedno	Koriste se za proširivanje funkcionalnosti jezika i koriste se u aplikacijama

B.5: Razlike između skriptova i funkcija

alata koji se pokreće komandom `pathtool`. Pogledati dokumentaciju za detaljniji opis korišćenja ovog alata.

Svaka `m`-datoteka bi trebalo da bude dokumentovana po MATLAB standardima. Komentari se navode iza znaka `%` i komentarom se smatra celokupni tekst do kraja reda. Komentari u početnim linijama `m`-datoteke se smatra njenom dokumentacijom. MATLAB korisniku automatski ispisuje tekst ovih početnih komentara kada korisnik zatraži pomoć komandom: `help` *ime m-datoteke*. Prva linija (H1 linija) je od posebnog značaja jer predstavlja kraći oblik dokumentacije i trebalo bi da sadrži veoma kratak i jasan opis funkcionalnosti `m`-datoteke. Linije standardne dokumentacije prestaju posle prve linije koja ne počinje znakom `%`.

Skriptovi

Skriptovi predstavljaju kolekcije nekoliko komandi. Komande se zadaju u potpuno identičnoj sintaksi kao i prilikom rada iz komandne linije. Po unošenju imena skripta, MATLAB pokreće taj niz komandi i rezultat njihovog rada se ispisuje u komandnom prozoru. Skriptovi nemaju zaseban, već koriste zajednički (globalni) memorijski prostor. Po završetku rada skripta, sve u njemu kreirane promenljive zadržavaju svoju vrednost i dostupne su i drugim skriptovima i prilikom nastavka rada iz komandnog prozora. Takođe, svi skriptovi mogu da pristupaju svim promenjivim koje su ranije bile definisane u okviru komandnog prozora ili drugih skriptova (sve ovo, naravno, pod uslovom da promenljive nisu eksplicitno obrisane komandom `clear`).

Funkcije

Veliki broj standardnih MATLAB funkcija se isporučuje u obliku `m`-datoteka. Ipak, mali deo najznačajnijih i najviše korišćenih standardnih funkcija je distribuiran u vidu binarnih rutina kreiranih u C-u. Korisnici su u mogućnosti da kreiraju nove funkcije putem kreiranja `m`-datoteka i ove funkcije postaju potpuno ravnopravne ugrađenim. Pogledajmo sadržaj `m`-datoteke koja sadrži definiciju ugrađene funkcije

`factorial` koja izračunava $n!$.

```
1:  function p = factorial(n)
2:  %FACTORIAL Factorial function.
3:  %   FACTORIAL(N) is the product of all the integers from 1 to N,
4:  %   i.e. prod(1:N).
5:  %
6:  %   See also PROD.
7:
8:  %   Copyright 1984-2002 The MathWorks, Inc.
9:  %   $Revision: 1.7 $
10:
11: if (length(n)~=1) | (fix(n) ~= n) | (n < 0)
12:   error('MATLAB:factorial:NNotPositiveInteger', ...
13:         'N must be a positive integer.');
14: end
15:
16: p = prod(1:n);
```

Prva linija sadrži definiciju funkcije, koja je oblika

`function [`o_1`,...,`o_n`] = name (`i_1`,...,`i_m`)`

Nakon ovoga, slede linije dokumentacije, a zatim telo same funkcije.

Svaka funkcija prihvata određeni broj argumenata i takođe, može da vrati određeni broj vrednosti. Pošto MATLAB nije striktno tipiziran jezik, uz ove vrednosti se ne navode njihovi tipovi. Funkcija za računanje faktorijela je definisana samo za prirodne brojeve. Pošto jezik ne vrši proveru da je prosleđeni argument funkcije `n` celobrojnog tipa, dužni smo da u okviru implementacije same funkcije izvršimo takvu proveru. Ona je izvršena u linijama `11-14`. Ukoliko uslov nije ispunjen, funkcijom `error`, prijavljuje se greška. Nedostatak statičke provere tipova daje, sa jedne strane fleksibilnost prilikom korišćenja funkcija, ali, sa druge strane, programerima nameće veliku odgovornost provere korektnosti argumenata.

Linija `16` vrši samo izračunavanje, koje je u ovom slučaju trivijalno. Tehnika vraćanja vrednosti iz funkcije je njihovo dodeljivanje izlaznim argumentima.

Funkcije kao argumenti funkcija

Često je u radu potrebno da se jednoj funkciji prenese druga funkcija kao argument. Funkcija `feval` izračunava vrednost funkcije koja joj se navodi kao prvi argument u tačkama koje se navode kao drugi argument. Funkciji `feval`, prvi argument je moguće preneti na četiri načina:

1. Navodi se ime funkcije odnosno m-datoteke u kojoj je funkcija definisana u obliku stringa. Npr. `feval('sin', 1)`
2. Navodi se analitički izraz koji opisuje funkciju u obliku stringa. Npr. `feval('sin(x)', 1)`
3. Navodi se tzv. inline-objekat kojim je opisana funkcija. Npr. `feval(inline('sin(x)'), 1)`.

4. Navodi se tzv. function-handle funkcije. Npr. `feval(@sin, 1)`.

Alternativa broj 2 je ekvivalentna alternativi broj 3 jer se prilikom zadavanja analitičkog izraza funkcije u obliku stringa implicitno kreira inline-objekat koji opisuje funkciju. Sa stanovišta brzine izvršavanja, najpoželjnije je korišćenje alternative 4. Opciju 2 ili 3 bi trebalo koristiti u slučajevima kada se želi izbeći kreiranje posebnih m-datoteka za svaku funkciju koja se prosleđuje funkciji `feval`.

Mnoge ugrađene funkcije (npr. `quad` koja vrši numeričku integraciju) kao neke od svojih argumenata primaju druge funkcije. Većina ovakvih funkcija u svojoj implementaciji pozivaju funkciju `feval` tako da se pravila za prenos funkcije funkciji `feval` direktno prenose i na njih.

Kao primer pisanja funkcije koja prima drugu funkciju kao argument, napišimo funkciju za numeričko diferenciranje. Funkcija `fdiff` kao argumente ima funkciju f koja se diferencira, tačku x u kojoj se izvod traži i korak h koji se koristi pri aproksimaciji.

```
function fp = fdiff(f, x, h)
fp = (feval(f, x+h) - feval(f, x)) / h;
```

B.8 Simbolička izračunavanja

Kreiranje simboličkih izraza:	`sym`, `syms`
Uprošćavanja izraza:	`simplify`, `simple`, `expand`, `factor`, `collect`
Ispisivanje izraza:	`pretty`, `latex`, `ccode`, `fortran`
Zamena promenljive:	`subs`
Rešavanje jednačina:	`solve`
Analiza:	`diff`, `int`, `taylor`, `dsolve`, `limit`, `symsum`, ...
Veliki brojevi:	`vpa`
	Detaljnije: `help symbolic`

Iako postoji mnogo softverskih paketa koji su specijalizovani za vršenje simboličkih izračunavanja (npr. MATHEMATICA, MATH-CAD, MAPLE, ...), MATLAB poseduje određene mogućnosti simboličkih izračunavanja koje su date kroz *symbolic math toolbox*. Ovaj modul je zasnovan na biblioteci koda koja predstavlja jezgro pomenutog sistema MAPLE.

Simbolička izračunavanja se vrše nad simboličkim izrazima. Simbolički izraz se kreira na osnovu niske karaktera koja predstavlja njegov tekstualni zapis u uobičajenoj MATLAB sintaksi. Ovakve niske se predaju funkciji `sym` koja vrši njihovo parsiranje. Kraći način da se kreiraju simbolički izrazi koji predstavljaju promenljive (tzv. simboličke promenljive) je korišćenje funkcije `syms` sa listom simboličkih promenljivih. Npr.

```
>> e = sym('sin(x)^2 + cos(x)^2')       >> syms x y z
e =
sin(x)^2 + cos(x)^2
```

Većina funkcija za rad sa simboličkim izrazima primaju kao argumente i niske karaktera i automatski ih konvertuju u simboličke izraze korišćenjem funkcije `sym`.

Izraze je moguće prikazati u formatu koji je čitiljiv za čoveka, formatu koji je spreman za umetanje u LaTeX ili koji je spreman za direktno umetanje u programski kod jezika C, odnosno FORTRAN. Ovo se radi funkcijama `pretty`, `latex`, `ccode`, `fortran`.

```
e = sym('(x+1)/(x^2+3)');
>> pretty(e)

                                    x + 1
                                    ------
                                     2
                                    x  + 3
>> latex(e)
ans =
{\frac {x+1}{{x}^{2}+3}}

>> ccode(e)
ans =
      t0 = (x+1.0)/(x*x+3.0);
```

Jednom kreirani izrazi se mogu transformisati i uprošćavati korišćenjem niza funkcija poput `factor`, `expand`, `collect`, `simple`, `simplify`, ... Za detalje ovih funkcija pogledati dokumentaciju. Naglasimo da `simplify` kombinuje većinu ostalih transformacija kako bi dobila najmanji rezultujući izraz.

```
>> e = sym('sin(x)^2 + cos(x)^2');
>> simplify(e)
ans =
1
```

```
>> factor('x^9 - 1')
   ans =
   (x-1)*(x^2+x+1)*(x^6+x^3+1)
```

Promenljive u izrazima se mogu zameniti drugim izrazima koristeći funkciju `subs`. Ovo je moguće iskoristiti i za izračunavanje vrednosti izraza u datoj tački. Npr.

```
>> e = sym('sin(x)+sin(y)');
>> subs(e, {x, y}, {pi/4, pi/2})
ans =
    1.7071
```

MATLAB ima mogućnost izvođenja osnovnih operacija matematičke analize kao što su:

- `diff` - pronalazi izvod funkcije. Kao opcioni argumenti se mogu zadati promenljiva po kojoj se diferencira i red izvoda
- `int` vrši integraciju. Kao opcioni argumenti se mogu zadati promenljiva integracije kao i interval integracije, u kom slučaju se izračunava određeni integral
- `limit` - pronalazi limes funkcije. Kao opcioni argumenti se zadaju promenjiva po kojoj se limes pronalazi kao i tačka kojoj promenljiva teži
- `taylor` - pronalazi Tejlorov polinom. Kao opcioni argumenti se zadaju tačka u čijoj se okolini funkcija razvija, kao i stepen polinoma
- `symsum` - vrši sumaciju niza odnosno reda

Izračunajmo npr.

$$\lim_{x\to\infty}\left(1+2\frac{t}{x}\right)^{3x}$$
$$\int_0^x \sin^2 t\ \mathrm{d}t$$

```
>> syms x t

>> limit((1+2*t/x)^(3*x), x, inf)
ans =
exp(6*t)

>> int('sin(t)^2', 0, x)
ans =
-1/2*cos(x)*sin(x)+1/2*x
```

Toolbox redefiniše i većinu osnovnih funkcija za rad sa matricama tako da mogu da rade i na simboličkom nivou. Tako je korišćenjem funkcije `inv` moguće pronaći inverz simboličke matrice

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
>> syms a b c d
>> inv([a b; c d])
ans =
[  d/(a*d-b*c), -b/(a*d-b*c)]
[ -c/(a*d-b*c),  a/(a*d-b*c)]
```

Toolbox sadrži još i funkcije za rad sa brojevima proizvoljne preciznosti (*variable precision arithmetic*). Ovo se postiže komandom `vpa`. Npr. π se može izračunati na 50 decimala korišćenjem

```
>> vpa(pi, 50)
ans =
3.1415926535897932384626433832795028841971693993751
```

Literatura

[1] Aljančić S., *Uvod u realnu i funkcionalnu analizu*, Građevinska knjiga, Beograd, 1968.

[2] Bahvalov N.S., *Numerical Methods*, Mir Publishers, Moscow, 1977.

[3] Berezin I.S., Židkov N., *Numerička analiza*, Naučna knjiga, Beograd, 1963.

[4] Davis P., *Interpolation and Approximation*, Dover Publications, New York, 1975.

[5] Demidovich B.P., Maron I.A., *Computational Mathematics*, Mir Publishers, Moscow, 1987.

[6] Fox L., Parker J.B., *Chebyshev Polynomials in Numerical Analysis*, Oxford University Press, London, 1972.

[7] Gustafsson F.,Bergman N., *MATLAB for Engineers Explained*, Springer, London, 2003.

[8] Hageman L.,Young D., *Applied Iterative Methods*, Academic Press, New York, 1981.

[9] Hall G., Watt J.M. (editors), *Modern Numerical Methods for Ordinary Differential Equations*, Clarendon Press, Oxford, 1976.

[10] Higham D.J.,Higham N.J., *MATLAB Guide*, SIAM, Philadelphia, 2000.

[11] Hildebrand F.B., *Introduction to Numerical Analysis*, McGraw–Hill, New York, 1974.

[12] Hildebrand F.B., *Advanced Calculus for Applications*, Prentice–Hall, Englewood Cliffs, N.J., 1976.

[13] Jovanović B., Radunović D., *Numerička analiza*, Naučna knjiga, 1993.

[14] Kernigan B., Riči D., *Programski jezik C*, Savremena administracija, 1992.

[15] Moler C.B., *Numerical Computing with MATLAB*, SIAM, Philadelphia, 2004.

[16] Oden J.T., Reddy J.N., *An Introduction to the Mathematical Theory of Finite Elements*, Wiley, New York, 1976.

[17] Ortega J., *Numerical Analysis – A Second Course*, SIAM, Philadelphia, 1990.

[18] Ortega J., Rheinboldt W., *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[19] Parlett B., *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, N.J., 1980.

[20] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P., *Numerical Recipes in C*, Cambridge University Press, 1992

[21] Radunović D., *Numeričke metode*, Univerzitet u Beogradu, 1998.

[22] Ralston A., *A First Course in Numerical Analysis*, McGraw–Hill, Tokyo, 1965.

[23] Samardžić A., *GNU programerski alati*, Matematički fakultet Univerziteta u Beogradu, 2001.

[24] Stoer J., Bulirsch R., *Introduction to Numerical Analysis*, Springer–Verlag, New York 1980.

[25] Strang G., *Introduction to Applied Mathematics*, Willesley–Cambridge Press, 1986.

[26] `http://www.gnu.org/software/libmatheval/`

[27] `http://www.matf.bg.ac.yu/r3nm/`