Univerza v Ljubljani

Fakulteta za elektrotehniko

Andrej Perne

GRADIVO ZA VAJE IZ NUMERIČNIH METOD

Ljubljana, 2018

Predgovor

Gradivo za vaje iz Numeričnih metod pokriva vsebine, ki jih obdelujemo na laboratorijskih vajah pri predmetu Numerične metode na prvostopenjskem univerzitetnem študijskem programu Elektrotehnika na Fakulteti za elektrotehniko Univerze v Ljubljani. Kot tako je v prvi vrsti namenjeno kot pomoč študentom pri izvajanju laboratorijskih vaj pri omenjenem predmetu, pa tudi kot učno gradivo vsem ostalim, ki bi želeli samostojno obdelati obravnavane vsebine. V gradivu so zajete vsebine iz spoznavanja programskega paketa Octave, aritmetike v premični piki, vektorskih in matričnih norm, reševanja (pre)določenih sistemov linearnih enačb (LU razcep, razcep Choleskega, QR razcep), reševanja nelinearnih enačb in sistemov nelinearnih enačb, aproksimacije po metodi najmanjših kvadratov, polinomske interpolacije, zlepkov, numeričnega odvajanja in integriranja ter reševanja začetnih in robnih problemov pri diferencialnih enačbah. Avtor se zahvaljuje prof. dr. Bojanu Orlu za strokovni pregled in nasvete, ki so pomagali izboljšati gradivo.

A. Perne

Kazalo

1 Uvod

1.1 Osnove programskega paketa Octave

Osnovni ukazi v operacijskem sistemu UNIX:

- **mkdir**: nova mapa
  Z ukazom `mkdir` MojaMapa ustvarimo novo mapo z imenom MojaMapa.
- **cd**: prehajanje med mapami
  Z ukazom `cd MojaMapa` se iz mape, ki vsebuje mapo MojaMapa, preselimo v mapo MojaMapa, z ukazom `cd ..` se iz mape MojaMapa preselimo v mapo, ki vsebuje mapo MojaMapa.
- **rmdir**: izbris mape
  Z ukazom `rmdir MojaMapa` izbrišemo mapo z imenom MojaMapa.
- **ls** ali **dir**: izpis vseh map in datotek, ki se nahajajo v trenutni mapi
- **man**: pomoč
- **cp**: kopiranje datoteke
  Z ukazom `cp vaja.txt Vaja` skopiramo datoteko vaja.txt, ki se nahaja v trenutni mapi, v mapo Vaja, ki se nahaja v trenutni mapi.
- **rm**: brisanje datoteke
  Z ukazom `rm vaja.txt` izbrišemo datoteko vaja.txt, ki se nahaja v trenutni mapi.
- **mv**: preimenovanje datoteke
  Z ukazom `mv vaja.txt naloga.txt` preimenujemo datoteko vaja.txt v naloga.txt.
- **octave**: zagon Octave

Osnovni ukazi v programskem paketu Octave:

- **help**, **doc**: pomoč, dokumentacija
  Z ukazom `help exp` izpišemo uporabo in primere za ukaz `exp`.
- **clear**: izbris spremenljivk
  Z ukazom `clear all` izbrišemo vse spremenljivke, z ukazom `clear x` izbrišemo spremenljivko `x`.
- **clc**: izbris ukaznega okna
- **exit** ali **quit**: izhod
- **who**, **whos**: informacija o trenutno uporabljenih spremenljivkah
- **what**: izpis vseh m-datotek v trenutni mapi
- **format**: oblikovanje izpisa na zaslonu
  Z ukazom `format long` spremenimo izpis števil v daljši izpis na 14 decimalk, z ukazom `format short` spremenimo izpis števil v krajši izpis na 4 decimalke. Ta izpis je privzet.
- **diary**: zapis v datoteko
  Z ukazom `diary vaja.txt` ustvarimo tekstovno datoteko vaja.txt, kamor se zapisuje vse kar se izpiše na zaslon. Z ukazom `diary off` izklopimo zapis na datoteko, z ukazom `diary on` ponovno vklopimo zapis na datoteko.
- **load**, **save**: naloži datoteko, shrani v datoteko

Aritmetika IEEE:

- **isieee**: ali je aritmetika IEEE?
- **computer**: tip računalnika
- **eps**: razdalja od 1.0 do naslednjega predstavljivega števila v aritmetiki IEEE (`2.2204e-016`)
- **realmin**: najmanjše predstavljivo število (`2.2251e-308`)
- **realmax**: največje predstavljivo število (`1.7977e+308`)
- **inf**: neskončno
- **NaN**: ni število
- **tic**, **toc**: vklop in izklop štoparice

Za računanje z realnimi (in kompleksnimi) števili imamo naslednje algebrske operacije:

- **+, -** : seštevanje, odštevanje
- ***, /** : množenje, deljenje
- ^ : potenciranje

Decimalno število zapišemo z decimalno piko. Najpogostejše konstante:

- **pi** : število π
- **exp(1)** : osnova naravnega logaritma e
- **i** ali **j** : imaginarna enota i

Relacijski operatorji:

- == : enako
- ~= : različno
- < : manjše
- <= : manjše ali enako
- > : večje
- >= : večje ali enako

Logični (Boolovi) operatorji:

- & : logični in (in hkrati)
- | : logični ali
- ~ : logični ne (negacija)

Argumente funkcij pišemo v okroglih oklepajih ( ). Elementarne funkcije:

- **sqrt**: kvadratni koren ($\sqrt{x}$): `sqrt(9)` → `3`
- **abs**: absolutna vrednost ($|x|$): `abs(-1)` → `1`
- **exp**: eksponentna funkcija (e^x): `exp(0)` → `1`
- **log**: naravni logaritem ($\ln x$): `log(1)` → `0`
- **log10**: desetiški logaritem ($\log x$): `log10(10)` → `1`
- **log2**: logaritem z osnovo 2 ($\log_2 x$): `log2(4)` → `2`
- **pow2**: eksponentna funkcija z osnovo 2 (2^x): `pow2(3)` → `8`
- **sin**: sinus ($\sin x$): `sin(pi)` → `0`
- **cos**: kosinus ($\cos x$): `cos(0)` → `1`
- **tan**: tangens ($\mathrm{tg}\, x$): `tan(pi/4)` → `1`
- **cot**: kotangens ($\mathrm{ctg}\, x$): `cot(pi/2)` → `0`
- **asin**: arcus sinus ($\arcsin x$): `asin(1)` → `1.5708`
- **acos**: arcus kosinus ($\arccos x$): `acos(-1)` → `3.1416`
- **atan**: arcus tangens ($\mathrm{arctg}\, x$): `atan(1)` → `0.7854`
- **acot**: arcus kotangens ($\mathrm{arcctg}\, x$): `acot(-1)` → `-0.7854`
- **sign**: predznak števila x: `sign(-5)` → `-1`, `sign(5)` → `1`

Funkcije nad kompleksnimi števili:

- **real**: realna komponenta: `real(4+3*i)` → `4`
- **imag**: imaginarna komponenta: `imag(4+3*i)` → `3`
- **conj**: konjugirana vrednost kompleksnega števila: `conj(4+3*i)` → `4-3*i`
- **abs**: absolutna vrednost: `abs(4+3*i)` → `5`
- **angle**: kot: `angle(4+3*i)` → `0.6435`
- **isreal**: ali je število realno?: `isreal(4+3*i)` → `0`
- **complex**: pretvorba v kompleksne podatke

Zaokrožitvene funkcije in ostanek pri deljenju:

- **fix**: zaokrožanje na najbližje celo število proti 0: `fix(1.8)` → `1`, `fix(-1.8)` → `-1`
- **floor**: zaokrožanje navzdol: `floor(1.8)` → `1`
- **ceil**: zaokrožanje navzgor: `ceil(1.8)` → `2`
- **round**: zaokrožanje k najbližjemu celemu številu: `round(-1.8)` → `-2`
- **mod(x,y)**: ostanek pri deljenju x z y: `mod(5,2)` → `1`, `mod(-5,2)` → `1`
- **rem(x,y)**: ostanek pri deljenju x z y: `rem(5,2)` → `1`, `rem(-5,2)` → `-1`

Vektorji in polja:

- Vektor pišemo v oglatih oklepajih [ ]. Decimalno število zapišemo z decimalno piko. Komponente vektorja ločimo z vejicami ali presledki:
  `x = [1.65, 2.23, 1.99], y = [1 2 3 4 5]`
- Do komponent vektorja dostopamo z okroglimi oklepaji ( ). Več komponent izpišemo z operatorjem **:**. Kot primer izpišimo prvo komponento vektorja x ter drugo do četrto komponento vektorja y: `x(1)` $\rightarrow$ `1.65, y(2:4)` $\rightarrow$ `[2 3 4]`
- Z ukazom **linspace(a,b,n)** zgradimo vektor z n ekvidistantnimi komponentami med a in b. Kot primer definirajmo vektor s sedmimi ekvidistantnimi komponentami med 0 in 3: `z = linspace(0,3,7)` $\rightarrow$ `z=[0 0.5 1 1.5 2 2.5 3]`
- Z operatorjem **:** zgradimo vektor z ekvidistantnimi komponentami. Privzeti korak je 1. Kot primer definirajmo vektorja med 0 in 6 s korakoma 1 in 2:
  `z = 0:6` $\rightarrow$ `z = [0 1 2 3 4 5 6]`, `z = 0:2:6` $\rightarrow$ `z = [0 2 4 6]`

Pri poljih je potrebno pri nekaterih algebrskih operacijah pred operatorjem uporabljati piko:

- **+, -** : seštevanje, odštevanje
- **.*, .^** : množenje, potenciranje
- **./, .** : desno deljenje, levo deljenje
- **'** : transponiranje

Skalarni in vektorski produkt vektorjev:

- **dot(x,y)** : skalarni produkt vektorjev x in y
- **cross(x,y)** : vektorski produkt vektorjev x in y

Osnovne funkcije za delo s polji (vektorji):

- **length**: število komponent: `length(x)` $\rightarrow$ `3`
- **min**: najmanjša komponenta: `min(y)` $\rightarrow$ `1`
- **max**: največja komponenta: `max(y)` $\rightarrow$ `5`
- **sum**: vsota komponent: `sum(x)` $\rightarrow$ `5.87`
- **prod**: produkt komponent: `prod(y)` $\rightarrow$ `120`
- **cumsum**: kumulativna vsota komponent: `cumsum(x)` $\rightarrow$ `[1.65 3.88 5.87]`
- **cumprod**: kumulativen produkt komponent: `cumprod(y)` $\rightarrow$ `[1 2 6 24 120]`

Izračun norme vektorja. Z ukazom **norm** izračunamo normo vektorja. Privzeta je evklidska ali druga norma:

- `norm(x)` ali `norm(x,2)`: evklidska norma: `norm(y)` $\rightarrow$ `7.4162`
- `norm(x,1)`: prva norma: `norm(y,1)` $\rightarrow$ `15`
- `norm(x,inf)`: neskončna norma: `norm(y,inf)` $\rightarrow$ `5`

Urejanje in iskanje v poljih:

- **diff(x)**: vektor razlik med sosednjima komponentama vektorja x:
  `diff(y)` → `[1 1 1 1]`
- **sort(x)**: urejen vektor x
- **issort(x)**: ali je vektor x urejen?
- **unique(x)**: urejen vektor x, kjer se vsako število pojavi samo enkrat
- **find(x)**: iskanje neničelnih elementov v vektorju x, kjer kot rezultat dobimo vektor indeksov, kjer se nahajajo neničelni elementi: `z = [1 0 2]; find(z)` → `[1 3]`

Matrike:

- Matriko pišemo v oglatih oklepajih [ ]. Decimalno število napišemo z decimalno piko. Elemente v vrstici ločimo s presledkom ali vejico, vrstice pa ločimo s podpičjem:
  `A = [1 2 3; 4 5 6; 7 8 9]`
- Do elementov matrike dostopamo z okroglimi oklepaji ( ). Več elementov izpišemo z operatorjem **:**. Kot primer izpišimo element matrike A, ki leži v drugi vrstici in tretjem stolpcu, ter podmatriko matrike A od prve do druge vrstice in od drugega do tretjega stolpca: `A(2,3)` → `6`, `A(1:2,2:3)` → $\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$

Algebrske operacije z matrikami:

- **+**, **-** : seštevanje, odštevanje
- *****, ^ : množenje, potenciranje
- **/**, \ : desno deljenje, levo deljenje
- **'** ali **transpose** : transponiranje

Osnovne funkcije za delo z matrikami:

- **size(A)**: dimenzije matrike A (število vrstic in stolpcev): `size(A)` → `3 3`
- **rows(A)**, **columns(A)**: število vrstic, število stolpcev
- **max(A)**: vektor največjih elementov v stolpcih matrike A:
  `max(A)` → `[7 8 9]`, `max(max(A))` → `9`
- **min(A)**: vektor najmanjših elementov v stolpcih matrike A:
  `min(A)` → `[1 2 3]`, `min(min(A))` → `1`
- **sum(A,1)**: vektor vsot elementov po stolpcih (privzeto): `sum(A,1)` → `[12 15 18]`
- **sum(A,2)**: vektor vsot elementov po vrsticah: `sum(A,2)` → `[6 15 24]`
- **prod(A,1)**: vektor produktov elementov po stolpcih (privzeto):
  `prod(A,1)` → `[28 80 162]`
- **prod(A,2)**: vektor produktov elementov po vrsticah: `prod(A,2)` → `[6 120 504]`
- **fliplr(A)**: zrcaljenje matrike A levo desno
- **flipud(A)**: zrcaljenje matrike A gor dol

Izračun norme matrike. Z ukazom **norm** izračunamo normo matrike. Privzeta je spektralna ali druga norma:

- `norm(x)` ali `norm(x,2)`: spektralna ali druga norma: `norm(A)` → `16.8481`
- `norm(A,1)`: prva norma (največja stolpična vsota): `norm(A,1)` → `18`
- `norm(A,inf)`: neskončna norma (največja vrstična vsota): `norm(A,inf)` → `24`
- `norm(A,'fro')`: Frobeniusova norma: `norm(A,'fro')` → `16.8819`

Matrične funkcije za konstrukcijo matrik:

- **ones(n,m)**: matrika dimenzije $n \times m$ iz samih enic: `ones(2,3)` → $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- **zeros(n,m)**: matrika dimenzije $n \times m$ iz samih ničel: `zeros(2,2)` → $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
- **eye(n)**: identična matrika dimenzije $n \times n$: `eye(2)` → $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- **rand(n,m)**: naključna matrika dimenzije $n \times m$
- **diag(A)**: diagonala matrike A (privzeto $n = 0$): `diag(A)` → `[1 5 9]`
- **diag(A,n)**: n-ta naddiagonala (poddiagonala) matrike A
- **diag(A,1)**: prva naddiagonala matrike A, **diag(A,-1)**: prva poddiagonala matrike A
- **diag(x)**: sestavimo diagonalno matriko A, ki ima na diagonali elemente vektorja x: `diag([1 2 3])` → $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$
- **diag(diag(A))**: diagonalna matrika, ki ima na diagonali diagonalne elemente matrike A: `diag(diag(A))` → $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$
- **triu(A)**: zgornji trikotnik matrike A (privzeto $n = 0$): `triu(A)` → $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix}$
- **triu(A,n)**: zgornji trikotnik od n-te naddiagonale naprej
- **triu(A,1)**: strogi zgornji trikotnik matrike A: `triu(A,1)` → $\begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix}$
- **tril(A)**: spodnji trikotnik matrike A (privzeto $n = 0$): `tril(A)` → $\begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 7 & 8 & 9 \end{bmatrix}$
- **tril(A,-n)**: spodnji trikotnik od n-te poddiagonale naprej
- **tril(A,-1)**: strogi spodnji trikotnik matrike A: `tril(A,-1)` → $\begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}$

Preoblikovanje matrik in razpršene matrike:

- Z ukazom **reshape(x,n,n)** iz vektorja dolžine n^2 konstruiramo matriko dimenzije $n \times n$ tako, da elemente zložimo po stolpcih.
  `x = [1 2 3 4 5 6 7 8 9];`
  `reshape(x,3,3)` $\rightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
- Z ukazom **sparse(i,j,a,m,n)** sestavimo razpršeno matriko dimenzije $m \times n$, kjer so v vektorju i shranjeni indeksi vrstic, v vektorju j indeksi stolpcev in v vektorju a elementi matrike.
  `i = [1 2]; j =[3 1]; a=[9 5];`
  `sparse(i,j,a,3,3)` $\rightarrow \begin{bmatrix} 0 & 0 & 9 \\ 5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- Ukaz **full** redko matriko pretvori v polno.

Linearna algebra:

- **det(A)**: determinanta matrike A
- **eig(A)**: lastne vrednosti in lastni vektorji matrike A
- **poly(A)**: karakteristični polinom matrike A
- **norm(A)**: norma matrike A
- **rank(A)**: rang matrike A
- **inv(A)**: inverz matrike A
- **cond(A)**: pogojenostno število matrike A
- **trace(A)**: sled matrike A (vsota diagonalnih elementov)
- **rref(A)**: reducirana oblika matrike A
- **null(A)**: ničelni prostor matrike A
- **orth(A)**: ortonormalna baza matrike A
- **lu(A)**: LU razcep matrike A
- **chol(A)**: razcep Choleskega matrike A
- **qr(A)**: QR razcep matrike A
- **svd(A)**: singularni razcep matrike A

Sistem linearnih enačb $Ax = b$ rešimo z ukazom $\mathbf{x = A \setminus b}$ (levo deljenje):

```
A = [1 2 1; 2 1 2; 1 1 2];
b = [4 5 4]';
x = A \ b → [1 1 1]'
```

Posebne matrike:

- **hilb**: Hilbertova matrika
- **pascal**: Pascalova matrika
- **toeplitz**: Toeplitzova matrika
- **vandre**: Vandermondova matrika
- **hankel**: Hankelova matrika
- **hadamard**: Hadamardova matrika
- **compan**: matrika spremljevalka
- **magic**: magični kvadrat

Ukaz **gallery** skupaj z ustrezno izbiro parametrov uporabimo za konstrukcijo testnih matrik:

- **circul**: cirkulantska (krožna) matrika:

  `A = gallery('circul',[1 2 3 4])` $\rightarrow A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}$

- **tridiag**: tridiagonalna matrika:

  ```
  A = gallery('tridiag',[1 1 1],[1 2 3 4],[2 2 2]);
  ```

  `full(A)` $\rightarrow A = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{bmatrix}$

Zanke in pogojni stavki:

- V Octave lahko tudi programiramo.
- Na voljo imamo zanki **for** in **while**. For zanka izvede določeno število ponovitev, while zanka pa se izvaja dokler je pogoj izpolnjen. Iz zanke lahko izstopimo z ukazom **break**.
- Na voljo imamo tudi pogojna stavka **if** in **switch**, ki izvajata ukaze glede na pogoj.

m-datoteke:

- Poznamo dva tipa m-datotek: skripte in funkcije.
- Skripte so zgolj daljše kode, ki jih ne želimo pisati v ukazno vrstico.
- Funkcije pa se vedno začnejo z ukazom **function**. Funkcijam podamo vhodne in izhodne argumente. Tako napisano funkcijo kličemo na enak način kot vgrajene funkcije. Ime funkcijske m-datoteke naj bo enako imenu funkcije. Kot primer napišimo funkcijo, ki izračuna obseg in ploščino kroga z danim polmerom r.

  ```
  function [o,p] = krog(r)
  o = 2*pi*r;
  p = pi*r ^ 2;
  ```

  `[o,p] = krog(3)` $\rightarrow$ `o = 18.8496, p = 28.2743`

Ukazi za vrednotenje nizov:

- Z ukazom **eval** ovrednotimo nize. Funkcijo $f(x) = x^2$ zapišimo kot niz in izračunajmo njeno vrednost v točki 2.

  ```
  f = 'x ^ 2';
  x = 2; eval(f) → 4
  ```

- Z ukazom **feval** izračunamo funkcijske vrednosti: `feval('sin',3)` → `0.1411`

- Z ukazom **vectorize** operacije v izrazu nadomestimo z ustreznimi operacijami nad polji: `vectorize('x*y ^ 2/z')` → `x.*y.^ 2./z`

Notranje in anonimne funkcije:

- Z ukazom **inline** definiramo notranje funkcije. Definirajmo funkcijo $f(x) = x^2$ in izračunajmo njeno vrednost v točki 2.

  ```
  f = inline('x ^ 2','x');
  f(2) → 4
  ```

- Z operatorjem **@** definiramo anonimne funkcije. Definirajmo funkcijo $f(x) = \frac{\sin x}{x}$ in izračunajmo njeno vrednost v točki 2.

  ```
  f = @(x) sin(x)/x;
  f(2) → 0.4546
  ```

Ukazi za delo s polinomi:

- Z ukazom **polyval(p,x)** izračunamo vrednost polinoma p v točki x. Polinom podamo z vektorjem koeficientov, ki so urejeni po padajočih potencah x-a.

- Z ukazom **roots(p)** izračunamo ničle polinoma p. Kot primer izračunajmo vrednost polinoma $p(x) = x^3 - 3x + 2$ v točki $x = 2$ ter njegove ničle.

  ```
  polyval([1 0 -3 2],2) → 4
  roots([1 0 -3 2]) → [-2 1 1]
  ```

- Z ukazom **polyfit(x, y, n)** določimo koeficiente aproksimacijskega oz. interpolacijskega polinoma stopnje n, kjer za interpolacijske točke vzamemo pare komponent iz vektorjev x in y.

- Z ukazom **spline(x,y)** določimo kubični zlepek, kjer za interpolacijske točke vzamemo pare komponent iz vektorjev x in y.

Risanje grafov funkcij:

- Z ukazom **plot(x,y)** narišemo graf funkcije, ki je podana z vektorjema `x` (x koordinate) in `y` (y koordinate). Kot primer narišimo funkcijo $y = \sin x$ na intervalu $[0, 2\pi]$.

  ```
  x = 0:0.01:2*pi;
  y = sin(x);
  plot(x,y)
  ```

- Sliki lahko dodamo naslov z opcijo **titel**, oznake na x- in y-osi z opcijama **xlabel** in **ylabel** ter legendo z opcijo **legend**.

- Poleg osnovnega ukaza **plot** imamo za izris slik na voljo še druge ukaze, npr. **semilogy** (logaritemska skala na y-osi) in **plot3** (grafi v 3 dimenzijah).

Ukazi za delo s slikami:

- **axis**: osi
  Z ukazom **axis equal** določimo, da sta enoti na obeh koordinatnih oseh enaki, z ukazom **axis([-5,5,-3,3])** pa določimo, da se graf funkcije izriše na intervalu $[-5, 5]$ na x-osi in $[-3, 3]$ na y-osi.
- **hold**: zadržanje slike
  Z ukazom **hold on** zadržimo sliko, da lahko na isto sliko narišemo nov graf, z ukazom **hold off** izklopimo zadržanje slike.
- **grid**: mreža
  Z ukazom `grid on` vklopimo mrežo, z ukazom `grid off` izklopimo mrežo.

Opcije za izris grafov funkcij (zapišemo jih v enojnih navednicah **' '**):

- Barve: **b** modra, **g** zelena, **r** rdeča, **y** rumena, **c** zelenomodra, **m** vijolična, **k** črna, **w** bela.
- Črte: **-** polna, **:** pikčasta, − črtkasta, **-.** pikčasto-črtkasta.
- Debelina črt: **'LineWidth',2** dvojna debelina.
- Markerji: **.** točka, **o** krogec, **x** x-znak, + plusek, $*$ zvezdica, **s** kvadratek, **d** diamant, **v** trikotnik navzdol, ^ trikotnik navzgor, $<$ trikotnik levo, $>$ trikotnik desno, **p** petkotnik, **h** šestkotnik.

1.2 Uvod v Octave

1. Izračunajte vrednost izrazov $f(x, y, z) = xe^y - \frac{2\sqrt{x}}{z^3}$ in $g(x, y, z) = \frac{x^3 \ln y + 3y}{\sqrt{z}}$, kjer je $x = 3.18$, $y = 1.79$ in $z = 1.68$.

   Koda v Octave:

   ```
   x = 3.18; y = 1.79; z = 1.68;
   f = @(x,y,z) x*exp(y)-2*sqrt(x)/z^3;
   g = @(x,y,z) (x^3*log(y)+3*y)/sqrt(z);
   f(x,y,z)
   g(x,y,z)
   ```

   Rezultati: a) 18.2942893579874, b) 18.5878129722647.

2. Dan je vektor

   ```
   x = fix(36*sin(sqrt(1:1000)+1));
   ```

   Koliko komponent vektorja x je

   a) manjših od -5 ali večjih od 3,

   b) različnih od 0 in različnih od 1?

   Koda v Octave:

   ```
   x = fix(36*sin(sqrt(1:1000)+1));
   sum(x<-5 | x>3)
   sum(x~=0 & x~=1)
   ```

   Rezultati: a) 909, b) 973.

3. Dan je vektor

```
x = fix(500*cos(sqrt(1:62)+1).^2);
```

Izračunajte število komponent, največjo in najmanjšo komponento ter vsoto komponent vektorja.
Koda v Octave:

```
x = fix(500*cos(sqrt(1:62)+1).^2);
length(x), min(x), max(x), sum(x)
```

Rezultati: 62, 0, 499, 14001.

4. Sestavite tridiagonalno matriko $A = \begin{bmatrix} 1 & 1 & & & \\ -2 & 2 & 1 & & \\ & -2 & 3 & 1 & \\ & & -2 & 4 & 1 \\ & & & -2 & 5 \end{bmatrix}$.

Koda v Octave:

```
A = diag(1:5)+diag(ones(1,4),1)-2*diag(ones(1,4),-1)
```

5. Dana je matrika $A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$. Konstruirajte matrike L, U in T, kjer je L spodnji trikotnik matrike A, U zgornji trikotnik, T pa tridiagonalni del matrike A.
Koda v Octave:

```
A = gallery('circul',1:6);
A = [A(1,:); flipud(A(2:6,:))]
L = tril(A), U = triu(A), T = tril(A,1)-tril(A,-2)
```

6. Dan je vektor $x = (1, -2, 4, -1, -5, 3, 1, 2, -3, -1)^T$. Izračunajte prvo, drugo (evklidsko) in neskončno normo vektorja x.
Koda v Octave:

```
x = [1 -2 4 -1 -5 3 1 2 -3 -1];
norm(x,1), norm(x), norm(x,inf)
```

Rezultati: 23, 8.42614977317636, 5.

7. Dana je matrika $A = \begin{bmatrix} 1 & -2 & 3 & -4 \\ 2 & 3 & -1 & -2 \\ 1 & -3 & 4 & 3 \\ -4 & 2 & -2 & 1 \end{bmatrix}$. Izračunajte prvo, drugo (spektralno), neskončno in Frobeniusovo normo matrike A.
Koda v Octave:

```
A = [1 -2 3 -4; 2 3 -1 -2; 1 -3 4 3; -4 2 -2 1];
norm(A,1), norm(A), norm(A,inf), norm(A,'fro')
```

Rezultati: 10, 7.66108637940076, 11, 10.3923048454133.

8. Rešite sistem linearnih enačb

$$\begin{aligned} x + y + z &= 6, \\ 2x - y + z &= 3, \\ 3x + 2y + 2z &= 13. \end{aligned}$$

Uporabite operator levega deljenja matrik. Izračunajte še pogojenostno število matrike koeficientov danega sistema.

Koda v Octave:

```
A = [1 1 1; 2 -1 1; 3 2 2]; b = [6; 3; 13];
x = A\b, cond(A)
```

Rezultati: $x = 1$, $y = 2$, $z = 3$, 21.2303172459758.

9. Sestavite m-datoteko, ki kot vhodni podatek dobi polmer in višino valja ter vrne njegovo površino in prostornino. Izračunajte površino in prostornino valja s podatki $r = 2$, $v = 3$.

m-datoteka `valj.m`:

```
function [P,V] = valj(r,v)
  P = 2*pi*r*(r+v);
  V = pi*r^2*v;
```

Koda v Octave:

```
[P,V] = valj(2,3)
```

Rezultati: $P = 62.8318530717959$, $V = 37.6991118430775$.

1.3 Aritmetika v premični piki in numerične napake

Število x v premični piki predstavimo kot

$$x = \pm m \cdot b^e,$$

kjer je $m = 0.c_1c_2 \ldots c_t$ mantisa, t dolžina mantise, b baza (npr. 2, 10 ali 16), e eksponent ($L \leq e \leq U$), c_i pa so števke v mejah od 0 do $b - 1$. Če je $c_1 \neq 0$, je mantisa normalizirana, sicer ($c_1 = 0$) pa je denormalizirana. Zapis označimo z $P(b, t, L, U)$. Številom, ki jih dobimo na ta način, pravimo predstavljiva števila. Ostala števila zaokrožimo na najbližje predstavljivo število.

Aritmetika, ki jo predpisuje standard IEEE, ima bazo $b = 2$ in normalizirano mantiso s $c_1 = 1$.

- V enojni natančnosti, kjer za predstavitev števila porabimo 32 bitov, gre 1 bit za predznak, 8 bitov za eksponent ter 23 bitov za mantiso ($c_2, \ldots, c_{24}$). Zapis $P(2, 24, -125, 128)$.
- V dvojni natančnosti, kjer za predstavitev števila porabimo 64 bitov, gre 1 bit za predznak, 11 bitov za eksponent ter 52 bitov za mantiso ($c_2, \ldots, c_{53}$). Zapis $P(2, 53, -1021, 1024)$.

Standard IEEE pozna še vrednosti 0, ∞, $-\infty$ ter NaN.

Osnovna zaokrožitvena napaka je enaka $u = \frac{1}{2}b^{1-t}$. V enojni natančnosti je $u \approx 6 \cdot 10^{-8}$, v dvojni natančnosti pa $u \approx 10^{-16}$. Če z $\mathrm{fl}(x)$ označimo najbližje predstavljivo število k številu x, potem je relativna napaka $\frac{|\mathrm{fl}(x)-x|}{|x|} \leq u$.

1. V formatu $P(2, 6, -10, 10)$ zapišite število $x = 7.712_{(10)}$. Poiščite prvo večje in prvo manjše predstavljivo število od števila x ter izračunajte relativni napaki za oba primera. Izračunajte osnovno zaokrožitveno napako ter največje in najmanjše predstavljivo število v danem formatu.

Rešitev: Najprej za celi del:

$$\begin{aligned} 7:2 &= 3 \text{ ost. } 1,\\ 3:2 &= 1 \text{ ost. } 1,\\ 1:2 &= 0 \text{ ost. } 1. \end{aligned}$$

Sledi $7_{(10)} = 111_{(2)}$. Nato za decimalni del:

$$\begin{aligned} 0.712 * 2 &= 0.424 + 1,\\ 0.424 * 2 &= 0.848 + 0,\\ 0.848 * 2 &= 0.696 + 1,\\ 0.696 * 2 &= 0.392 + 1,\\ 0.392 * 2 &= 0.784 + 0,\\ 0.784 * 2 &= 0.568 + 1,\\ 0.568 * 2 &= 0.136 + 1,\\ &\vdots \end{aligned}$$

Sledi $0.712_{(10)} = 0.1011011\ldots_{(2)}$. Skupaj

$$7.712_{(10)} = 111.1011011\ldots_{(2)} = 0.1111011011\ldots_{(2)} \cdot 2^3.$$

Prvo manjše predstavljivo število od števila x je $x_1 = 0.111101_{(2)} \cdot 2^3 = 7.625$, prvo večje pa $x_2 = 0.111110_{(2)} \cdot 2^3 = 7.750$. Relativni napaki sta

$$\frac{|x - x_1|}{|x|} = 0.0112811, \quad \frac{|x - x_2|}{|x|} = 0.00492739,$$

torej je predstavljivo število v tem formatu $\mathrm{fl}(x) = 7.75$, osnovna zaokrožitvena napaka pa je $u = 2^{-6} = 0.015625$. Največje predstavljivo število v tem formatu je $0.111111 \cdot 2^{10} = 1008$, najmanjše (normalizirano) pa $0.100000 \cdot 2^{-10} = 0.000488281$.

2. V formatu $P(10, 5, -100, 100)$ izračunajte vrednost polinoma

$$p(x) = 5.5555x^2 - 3.2111x + 9.8765$$

za $x = 12.345$ po Hornerjevem algoritmu. Izračunajte relativno napako

$$\frac{|\widehat{p}(x) - p(x)|}{|p(x)|}$$

ter jo primerjajte z osnovno zaokrožitveno napako v dani aritmetiki.

Rešitev: Hornerjev algoritem za izračun vrednosti polinoma $p(x) = a_2x^2 + a_1x + a_0$ izvedemo z zaporedjem operacij

$$\begin{aligned} p_1 &= a_2x,\\ v_1 &= p_1 + a_1,\\ p_2 &= v_1x,\\ v_2 &= p_2 + a_0, \end{aligned}$$

kjer je $p(x) = v_2$. V našem primeru dobimo

$$\begin{aligned} \widehat{p}_1 &= \mathrm{fl}(68.5826475) = 0.68583 \cdot 10^2,\\ \widehat{v}_1 &= \mathrm{fl}(65.3718999) = 0.65372 \cdot 10^2,\\ \widehat{p}_2 &= \mathrm{fl}(807.01734) = 0.80702 \cdot 10^3,\\ \widehat{v}_2 &= \mathrm{fl}(816.8965) = 0.81690 \cdot 10^3. \end{aligned}$$

Rešitev v dani aritmetiki je torej $\widehat{p}(x) = 816.9$, točna rešitev pa je $p(x) = 816.8882538875$. Relativna napaka $\frac{|\widehat{p}(x) - p(x)|}{|p(x)|} = 1.438 \cdot 10^{-5}$ je manjša od osnovne zaokrožitvene napake $u = 5 \cdot 10^{-5}$.

3. Integral

$$I_n = \int_0^1 \frac{x^n}{x+10}\,dx$$

izračunajte s pomočjo rekurzivne formule na dva načina za $n = 1, 2, \ldots, 17$.

Rešitev: Velja:

$$I_n + 10I_{n-1} = \int_0^1 \frac{x^n + 10x^{n-1}}{x+10}\,dx = \int_0^1 x^{n-1}\,dx = \frac{1}{n}.$$

1. rekurzivna formula:

$$I_0 = \int_0^1 \frac{dx}{x+10} = \log\frac{11}{10},\ I_n = \frac{1}{n} - 10I_{n-1},\ n = 1, 2, \ldots, 17.$$

2. rekurzivna formula:

$$I_{30} = 0,\ I_{n-1} = \frac{1}{10}\left(\frac{1}{n} - I_n\right),\ n = 30, 29, \ldots, 1.$$

Točna vrednost za $I_{17} = 0.00507489273026384324$ je na 20 decimalnih mest izračunana s programom `Mathematica`®.

Koda v Octave:

```
format long;
I1 = zeros(1,18); % prva rekurzija
I1(1) = log(11/10);
for i = 2:18 I1(i) = 1/(i-1) - 10*I1(i-1); end;
I2 = zeros(1,31); % druga rekurzija
for i = 30:-1:1 I2(i) = (1/i - I2(i+1))/10; end;
fprintf('prvi = %0.16f, drugi = %0.16f\n',I1(18),I2(18));
```

Rezultati: $I_{17}^1 = -7.4783887813187206$, $I_{17}^2 = 0.0050748927302641$.

4. Dana je linearen sistem $Ax = b$, kjer je matrika A Hilbertova matrika dimenzije 12×12 z elementi

$$a_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, 2, \ldots, 12,$$

vektor b pa je podan z

$$b = A \cdot (1, 1, \ldots, 1)^T.$$

Točna rešitev je seveda vektor $x = (1, 1, \ldots, 1)^T$.

a) Konstruiraje matriko A s `for` zanko ter z vgrajenim ukazom `hilb`.

b) Rešite sistem z operatorjem levega deljenja.

c) Primerjajte dobljeno rešitev s točno rešitvijo, tj. izračunajte drugo normo razlike med točno in izračunano vrednostjo. Kaj opazite?

d) Izračunajte pogojenostno število matrike A.

Koda v Octave:

```
A = zeros(12);
for i=1:12 for j=1:12 A(i,j)=1/(i+j-1); end; end;
A = hilb(12)
b = A*ones(12,1);
x = A\b
norm(x-ones(12,1))
cond(A)
```

Rezultati: c) 0.00086562490595, d) 17408259866660362.

1.4 Vektroske in matrične norme

1. Za vektor $x = (-1, 2, 5, 3, -2)^T$ izračunajte $\|x\|_1$, $\|x\|_2$ in $\|x\|_\infty$.
   Rešitev:

$$\begin{aligned} \|x\|_1 &= \sum_{i=1}^{n} |x_i| = 13, \\ \|x\|_2 &= \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{43}, \\ \|x\|_\infty &= \max_{1 \le i \le n} |x_i| = 5. \end{aligned}$$

2. Za matriko $A = \begin{bmatrix} 4 & 3 & 5 & -1 \\ 1 & -2 & 1 & 1 \\ 2 & -1 & -3 & 2 \\ -1 & 0 & 2 & 0 \end{bmatrix}$ izračunajte $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$ in $\|A\|_F$.
   Rešitev:

$$\begin{aligned} \|A\|_1 &= \max_{1 \le j \le n} \left(\sum_{i=1}^{m} |a_{ij}| \right) = 11, \\ &\quad \text{največja stolpična vsota} \\ \|A\|_2 &= \max_{1 \le i \le n} \sqrt{\lambda_i(A^T A)} = 7.5000651665, \\ \|A\|_\infty &= \max_{1 \le i \le m} \left(\sum_{j=1}^{n} |a_{ij}| \right) = 13, \\ &\quad \text{največja vrstična vsota} \\ \|A\|_F &= \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} = 9. \end{aligned}$$

2 Sistemi linearnih enačb

2.1 LU razcep

1. Dan je linearen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 2 & 1 & 3 & -4 \\ -4 & -1 & -4 & 7 \\ 2 & 3 & 5 & -3 \\ -2 & -2 & -7 & 9 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 8 \\ -14 \\ 7 \\ -16 \end{bmatrix}.$$

a) Izračunajte LU razcep (brez pivotiranja) za matriko A.

b) Z direktnim in obratnim vstavljanjem rešite sistem $Ax = b$.

c) Z uporabo LU razcepa izračunajte $\det A$.

LU razcep:

Sistem linearnih enačb z LU razcepom brez pivotiranja rešimo s sledečim postopkom.

- Matriko A razbijemo na produkt $A = LU$, kjer je L spodnja trikotna matrika z enkami na diagonali, U pa nesingularna zgornja trikotna matrika. Algoritem za LU razcep brez pivotiranja je

$$\begin{aligned} &\text{for } i = 1 : n-1 \\ &\quad \text{for } j = i+1 : n \\ &\qquad \ell_{ji} = \frac{a_{ji}}{a_{ii}} \ ; \\ &\qquad \text{for } k = i+1 : n \\ &\qquad\quad a_{jk} = a_{jk} - \ell_{ji} a_{ik} \ ; \\ &\qquad \text{end} \\ &\quad \text{end} \\ &\text{end} \end{aligned}$$

- Rešimo sistem $Ly = b$ z direktnim vstavljanjem. Algoritem za direktno vstavljanje je

$$\begin{aligned} &\text{for } i = 1 : n \\ &\qquad y_i = b_i - \sum_{k=1}^{i-1} \ell_{ik} y_k \ ; \\ &\text{end} \end{aligned}$$

- Rešimo sistem $Ux = y$ z obratnim vstavljanjem. Algoritem za obratno vstavljanje je

$$\begin{aligned} &x_n = \frac{y_n}{u_{nn}} \ ; \\ &\text{for } i = n-1 : -1 : 1 \\ &\qquad x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^{n} u_{ik} x_k \right) \ ; \\ &\text{end} \end{aligned}$$

Rešitev:

a) LU razcep (brez pivotiranja) za matriko A dobimo z redukcijo matrike A

$$A \sim \begin{bmatrix} 2 & 1 & 3 & -4 \\ 0 & 1 & 2 & -1 \\ 0 & 2 & 2 & 1 \\ 0 & -1 & -4 & 5 \end{bmatrix} \sim \begin{bmatrix} 2 & 1 & 3 & -4 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & -2 & 3 \\ 0 & 0 & -2 & 4 \end{bmatrix} \sim \begin{bmatrix} 2 & 1 & 3 & -4 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & -2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = U,$$

kjer je

$$L = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 1 & 2 & 1 & \\ -1 & -1 & 1 & 1 \end{bmatrix}.$$

b) Z direktnim vstavljanjem najprej rešimo sistem $Ly = b$:

$$\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 1 & 2 & 1 & \\ -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 8 \\ -14 \\ 7 \\ -16 \end{bmatrix},$$

ki ima rešitev $y = \begin{bmatrix} 8 & 2 & -5 & -1 \end{bmatrix}^T$, nato pa z obratnim vstavljanjem rešimo še sistem $Ux = y$:

$$\begin{bmatrix} 2 & 1 & 3 & -4 \\ & 1 & 2 & -1 \\ & & -2 & 3 \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ -5 \\ -1 \end{bmatrix},$$

ki ima rešitev $x = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^T$.

c) Determinanto spodnje ali zgornje trikotne matrike izračunamo kot produkt diagonalnih elementov:

$$\det A = \det LU = \underbrace{\det L}_{=1} \cdot \det U = -4.$$

2. V Octave sprogramirajte tri m-datoteke za reševanje linearnih sistemov $Ax = b$:

   a) LU razcep (brez pivotiranja) $A = LU$,

   b) direktno vstavljanje $Ly = b$,

   c) obratno vstavljanje $Ux = y$.

   Programe preizkusite na prejšnjem primeru.
   Rešitev:

   a) Koda v Octave (`lu_razcep.m`):

```
function [L,U] = lu_razcep(A)
  n = size(A,1);
  L = eye(n);
  for i = 1:n-1
    for j = i+1:n
      L(j,i) = A(j,i)/A(i,i);
      for k = i+1:n
        A(j,k) = A(j,k)-L(j,i)*A(i,k);
      end;
    end;
  end;
  U = triu(A);
```

   b) Koda v Octave (`direktno.m`):

```
function y = direktno(L,b)
  n = size(L,1);
  y = zeros(n,1);
  for i = 1:n
    y(i) = b(i)-L(i,1:i-1)*y(1:i-1);
  end;
```

c) Koda v Octave (`obratno.m`):

```
function x = obratno(U,y)
  n = size(U,1);
  x = zeros(n,1);
  for i = n:-1:1
    x(i) = (y(i)-U(i,i+1:n)*x(i+1:n))/U(i,i);
  end;
```

Koda v Octave:

```
A = [2 1 3 -4; -4 -1 -4 7; 2 3 5 -3; -2 -2 -7 9];
b = [8; -14; 7; -16];
[L,U] = lu_razcep(A)
y = direktno(L,b)
x = obratno(U,y)
```

3. Dan je linearen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 2 & 1 & -2 & 1 \\ 2 & 1 & -4 & 2 \\ 3 & -2 & 3 & -1 \\ -1 & 3 & -1 & 1 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 10 \\ 15 \\ -2 \\ 5 \end{bmatrix}.$$

a) Izračunajte LU razcep z delnim pivotiranjem za matriko A.

b) Z direktnim in obratnim vstavljanjem rešite sistem $Ax = b$.

c) Z uporabo LU razcepa izračunajte $\det A$.

Rešitev:

a) LU razcep z delnim pivotiranjem za matriko A dobimo z redukcijo matrike A

$$\begin{aligned} A &= \begin{bmatrix} 2 & 1 & -2 & 1 \\ 2 & 1 & -4 & 2 \\ 3 & -2 & 3 & -1 \\ -1 & 3 & -1 & 1 \end{bmatrix} \overset{*}{\sim} \begin{bmatrix} 3 & -2 & 3 & -1 \\ 2 & 1 & -4 & 2 \\ 2 & 1 & -2 & 1 \\ -1 & 3 & -1 & 1 \end{bmatrix} \sim \begin{bmatrix} 3 & -2 & 3 & -1 \\ 0 & \frac{7}{3} & -6 & \frac{8}{3} \\ 0 & \frac{7}{3} & -4 & \frac{5}{3} \\ 0 & \frac{7}{3} & 0 & \frac{2}{3} \end{bmatrix} \\ &\overset{**}{\sim} \begin{bmatrix} 3 & -2 & 3 & -1 \\ 0 & \frac{7}{3} & 0 & \frac{2}{3} \\ 0 & \frac{7}{3} & -4 & \frac{5}{3} \\ 0 & \frac{7}{3} & -6 & \frac{8}{3} \end{bmatrix} \sim \begin{bmatrix} 3 & -2 & 3 & -1 \\ 0 & \frac{7}{3} & 0 & \frac{2}{3} \\ 0 & 0 & -4 & 1 \\ 0 & 0 & -6 & 2 \end{bmatrix} \\ &\overset{***}{\sim} \begin{bmatrix} 3 & -2 & 3 & -1 \\ 0 & \frac{7}{3} & 0 & \frac{2}{3} \\ 0 & 0 & -6 & 2 \\ 0 & 0 & -4 & 1 \end{bmatrix} \sim \begin{bmatrix} 3 & -2 & 3 & -1 \\ 0 & \frac{7}{3} & 0 & \frac{2}{3} \\ 0 & 0 & -6 & 2 \\ 0 & 0 & 0 & -\frac{1}{3} \end{bmatrix} = U, \end{aligned}$$

kjer ustrezno popravljamo matriko L

$$\begin{bmatrix} 1 & & & \\ \frac{2}{3} & & & \\ \frac{2}{3} & & & \\ -\frac{1}{3} & & & \end{bmatrix} \overset{**}{\to} \begin{bmatrix} 1 & & & \\ -\frac{1}{3} & 1 & & \\ \frac{2}{3} & 1 & & \\ \frac{2}{3} & 1 & & \end{bmatrix} \overset{***}{\to} \begin{bmatrix} 1 & & & \\ -\frac{1}{3} & 1 & & \\ \frac{2}{3} & 1 & 1 & \\ \frac{2}{3} & 1 & \frac{2}{3} & 1 \end{bmatrix} = L$$

in permutacijsko matriko P

$$I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \overset{*}{\to} \begin{bmatrix} & 1 & & \\ 1 & & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \overset{**}{\to} \begin{bmatrix} & 1 & & \\ & & & 1 \\ 1 & & & \\ & & 1 & \end{bmatrix} \overset{***}{\to} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = P.$$

b) Z direktnim vstavljanjem najprej rešimo sistem

$$Ly = Pb = \begin{bmatrix} -2 & 5 & 15 & 10 \end{bmatrix}^T,$$

ki ima rešitev $y = \begin{bmatrix} -2 & \frac{13}{3} & 12 & -1 \end{bmatrix}^T$, nato pa z obratnim vstavljanjem še sistem

$$Ux = y,$$

ki ima rešitev $x = \begin{bmatrix} 2 & 1 & -1 & 3 \end{bmatrix}^T$.

c) Ker je

$$P\,A = L\,U,$$

je

$$\det A = \frac{\det L \cdot \det U}{\det P} = \frac{14 \cdot 1}{-1} = -14.$$

Za permutacijsko matriko P velja, da je $P^{-1} = P^T$ in $|\det P| = 1$.

4. V Octave sprogramirajte tri m-datoteke za reševanje linearnih sistemov $Ax = b$:

   a) LU razcep z delnim pivotiranjem $PA = LU$,

   b) direktno vstavljanje $P^T Ly = b$, oziroma $Ly = Pb$,

   c) obratno vstavljanje $Ux = y$.

   Programe preizkusite na prejšnjem primeru.
   Rešitev:

   a) Koda v Octave (`lu_razcep_delno.m`):

```
function [L,U,P] = lu_razcep_delno(A)
  n = size(A,1);
  P = eye(n);
  for i = 1:n-1
    [M,imax] = max(abs(A(i:n,i)));
    imax = imax+i-1;
    A([i,imax],:) = A([imax,i],:);
    P([i,imax],:) = P([imax,i],:);
    A(i+1:n,i) = A(i+1:n,i)/A(i,i);
    A(i+1:n,i+1:n) = A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
  end;
  L = eye(n)+tril(A,-1);
  U = triu(A);
```

   b) Glejte zgornjo nalogo (`direktno.m`).

   c) Glejte zgornjo nalogo (`obratno.m`).

   Koda v Octave:

```
A = [2 1 -2 1; 2 1 -4 2; 3 -2 3 -1; -1 3 -1 1];
b = [10; 15; -2; 5];
[L,U,P] = lu_razcep_delno(A)
y = direktno(L,P*b)
x = obratno(U,y)
```

5. Dana je tridiagonalna matrika A reda $n \times n$.

   a) Izpeljite algoritem za LU razcep, ki upošteva pasovno obliko matrike.
   b) Določite število potrebnih operacij (časovno zahtevnost algoritma).
   c) Algoritem implementirajte v Octave.

   Rešitev:

   a) Če je matrika (šibko) diagonalno dominantna, pivotiranje ni potrebno.
   Matrika A je tridiagonalna z elementi

$$A = \begin{bmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_2 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \gamma_{n-1} \\ & & & \beta_n & \alpha_n \end{bmatrix},$$

   matriki L in U v LU razcepu matrike A pa sta bidiagonalni z elementi

$$L = \begin{bmatrix} 1 & & & & \\ \ell_2 & 1 & & & \\ & \ell_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ell_n & 1 \end{bmatrix} \quad \text{in} \quad U = \begin{bmatrix} u_1 & d_1 & & & \\ & u_2 & d_2 & & \\ & & \ddots & \ddots & \\ & & & u_{n-1} & d_{n-1} \\ & & & & u_n \end{bmatrix}.$$

   Iz primerjave med elementi produkta

$$L \cdot U = \begin{bmatrix} u_1 & d_1 & & & & \\ \ell_2 u_1 & \ell_2 d_1 + u_2 & d_2 & & & \\ & \ell_3 u_2 & \ell_3 d_2 + u_3 & d_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ell_{n-1} u_{n-2} & \ell_{n-1} d_{n-2} + u_{n-1} & d_{n-1} \\ & & & & \ell_n u_{n-1} & \ell_n d_{n-1} + u_n \end{bmatrix}$$

   in matrike A dobimo enačbe

$$\begin{aligned} d_i &= \gamma_i, \quad i = 1, 2, \ldots, n-1, \\ u_1 &= \alpha_1, \\ \ell_i u_{i-1} &= \beta_i \quad \Rightarrow \quad \ell_i = \frac{\beta_i}{u_{i-1}}, \quad i = 2, 3, \ldots, n, \\ \ell_i d_{i-1} + u_i &= \alpha_i \quad \Rightarrow \quad u_i = \alpha_i - \ell_i \gamma_{i-1}, \quad i = 2, 3, \ldots, n \end{aligned}$$

   od koder sledi algoritem za tridiagonalni LU razcep:

$$\begin{aligned} & u_1 = \alpha_1 \ ; \\ & \text{for } i = 2 : n \\ & \quad \ell_i = \frac{\beta_i}{u_{i-1}} \ ; \\ & \quad u_i = \alpha_i - \ell_i \gamma_{i-1} \ ; \\ & \text{end} \end{aligned}$$

   b) Število operacij za tridiagonalni LU razcep je $0 + 3(n-1) = 3n - 3 = \mathcal{O}(n)$.

c) Koda v Octave (`tridiag_lu.m`):

```
function [L,U] = tridiag_lu(A)
  n = size(A,1);
  alfa = diag(A);
  beta = diag(A,-1);
  gama = diag(A,1);
  u = zeros(n,1);
  l = zeros(n-1,1);
  u(1) = alfa(1);
  for i = 2:n
    l(i-1) = beta(i-1)/u(i-1);
    u(i) = alfa(i)-l(i-1)*gama(i-1);
  end;
  L = eye(n)+diag(l,-1);
  U = diag(u)+diag(gama,1);
```

6. Dan je tridiagonalen sistem $Ax = b$, kjer je A matrika reda $n \times n$ na kateri napravimo tridiagonalni LU razcep.

   a) Izpeljite algoritem za direktno ($Ly = b$) in obratno ($Ux = y$) vstavljanje.
   b) Določite število potrebnih operacij (časovno zahtevnost algoritma).
   c) Algoritem implementirajte v Octave.

   Rešitev:

   a) Algoritem za direktno vstavljanje je:

$$
\begin{aligned}
&y_1 = b_1 \ ; \\
&\text{for } i = 2 : n \\
&\quad y_i = b_i - \ell_i y_{i-1} \ ; \\
&\text{end}
\end{aligned}
$$

   Algoritem za obratno vstavljanje je:

$$
\begin{aligned}
&x_n = \frac{y_n}{u_n} \ ; \\
&\text{for } i = n-1 : -1 : 1 \\
&\quad x_i = \frac{y_i - d_i x_{i+1}}{u_i} \ ; \\
&\text{end}
\end{aligned}
$$

   b) Število operacij za direktno vstavljanje je $0 + 2(n-1) = 2n - 2 = \mathcal{O}(n)$, za obratno vstavljanje $1 + 3(n-1) = 3n - 2 = \mathcal{O}(n)$, za reševanje sistema pa skupaj

$$3n - 3 + 2n - 2 + 3n - 2 = 8n - 7 = \mathcal{O}(n)$$

   operacij.

   c) Koda v Octave (`tridiag_direktno.m`):

```
function y = tridiag_direktno(L,b)
  n = size(L,1);
  l = diag(L,-1);
  y = zeros(n,1);
  y(1) = b(1);
  for i = 2:n
    y(i) = b(i)-l(i-1)*y(i-1);
  end;
```

   Koda v Octave (`tridiag_obratno.m`):

```
function x = tridiag_obratno(U,y)
  n = size(U,1);
  u = diag(U); d = diag(U,1);
  x = zeros(n,1);
  x(n) = y(n)/u(n);
  for i = n-1:-1:1
    x(i) = (y(i)-d(i)*x(i+1))/u(i);
  end;
```

7. Dan je linearen tridiagonalen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 2 & -1 & & & \\ 1 & 2 & -1 & & \\ & 1 & 2 & -1 & \\ & & 1 & 2 & -1 \\ & & & 1 & 2 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}.$$

a) Ali je matrika A (šibko) diagonalno dominantna?

b) Z uporabo algoritmov iz prejšnjih dveh nalog rešite dani sistem.

c) Izračunajte evklidsko normo rešitve sistema.

Koda v Octave:

```
A = [2 -1 0 0 0; 1 2 -1 0 0; 0 1 2 -1 0; 0 0 1 2 -1; 0 0 0 1 2];
b = [1; 2; 3; 4; 5];
[L,U] = tridiag_lu(A);
y = tridiag_direktno(L,b);
x = tridiag_obratno(U,y)
norm(x)
```

Rezultati:

a) Da. b) $x = \begin{bmatrix} 1.1571 & 1.3143 & 1.7857 & 1.8857 & 1.5571 \end{bmatrix}^T$. c) $\|x\| = 3.4980$.

8. Zgenerirajte tridiagonalno matriko A s spodnjim zaporedjem ukazov.

```
n = 1000;
A = 2*diag(ones(n,1))-diag(ones(n-1,1),1)+diag(ones(n-1,1),-1);
```

Z ukazoma `tic` in `toc` preverite časovno zahtevnost LU razcepa za tridiagonalne matrike ter jo primerjajte s časovno zahtevnostjo običajnega LU razcepa s pivotiranjem.

Koda v Octave:

```
n = 1000; A = 2*diag(ones(n,1))-diag(ones(n-1,1),1)+diag(ones(n-1,1),-1);
tic; [L,U] = tridiag_lu(A); toc;
tic; [L,U,P] = lu_razcep_delno(A); toc;
```

2.2 Razcep Choleskega

1. Dana je simetrična pozitivno definitna matrika

$$A = \begin{bmatrix} 4 & 6 & -4 & 4 & -2 \\ 6 & 10 & -7 & 4 & -2 \\ -4 & -7 & 14 & 7 & -8 \\ 4 & 4 & 7 & 18 & -11 \\ -2 & -2 & -8 & -11 & 19 \end{bmatrix}.$$

a) Izračunajte razcep Choleskega (brez pivotiranja) za matriko A.

b) Z uporabo razcepa Choleskega izračunajte $\det A$.

Razcep Choleskega:

Matrika A je simetrična pozitivno definitna natanko tedaj, ko je $A^T = A$ in so vse lastne vrednosti pozitivne.

Če je matrika A s.p.d., potem jo lahko razcepimo na $A = L\,L^T$, kjer je matrika L nesingularna spodnja trikotna matrika. Algoritem za razcep Choleskega je:

for $j = 1 : n$

$$\ell_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2}\ ;$$

for $i = j + 1 : n$

$$\ell_{ij} = \frac{1}{\ell_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk} \right),\ i > j\ ;$$

end

end

Rešitev:

a) Matriko L (faktor Choleskega) v razcepu Choleskega matrike A dobimo z algoritmom, kjer napravimo redukcijo matrike A

$$A = \begin{bmatrix} 4 & 6 & -4 & 4 & -2 \\ 6 & 10 & -7 & 4 & -2 \\ -4 & -7 & 14 & 7 & -8 \\ 4 & 4 & 7 & 18 & -11 \\ -2 & -2 & -8 & -11 & 19 \end{bmatrix} \sim \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 10 & 0 & 0 & 0 \\ -2 & -7 & 14 & 0 & 0 \\ 2 & 4 & 7 & 18 & 0 \\ -1 & -2 & -8 & -11 & 19 \end{bmatrix} \sim \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ -2 & -1 & 14 & 0 & 0 \\ 2 & -2 & 7 & 18 & 0 \\ -1 & 1 & -8 & -11 & 19 \end{bmatrix}$$

$$\sim \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ -2 & -1 & 3 & 0 & 0 \\ 2 & -2 & 3 & 18 & 0 \\ -1 & 1 & -3 & -11 & 19 \end{bmatrix} \sim \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ -2 & -1 & 3 & 0 & 0 \\ 2 & -2 & 3 & 1 & 0 \\ -1 & 1 & -3 & 2 & 19 \end{bmatrix} \sim \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ -2 & -1 & 3 & 0 & 0 \\ 2 & -2 & 3 & 1 & 0 \\ -1 & 1 & -3 & 2 & 2 \end{bmatrix} = L.$$

b) Determinanto izračunamo takole:

$$\det A = \det L \, L^T = (\det L)^2 = 12^2 = 144,$$

kjer je $\det L = 2 \cdot 1 \cdot 3 \cdot 1 \cdot 2 = 12$ produkt diagonalnih elementov.

Koda v Octave (`cholesky.m`):

```
function L = cholesky(A)
  n = size(A,1);
  L = zeros(n);
  for j = 1:n
    L(j,j) = sqrt(A(j,j)-L(j,1:j-1)*L(j,1:j-1)');
    for i = j+1:n
      L(i,j) = (A(i,j)-L(i,1:j-1)*L(j,1:j-1)')/L(j,j);
    end;
  end;
```

Koda v Octave:

```
A = [4 6 -4 4 -2; 6 10 -7 4 -2; -4 -7 14 7 -8;...
   4 4 7 18 -11; -2 -2 -8 -11 19];
L = chol(A)'
L = cholesky(A)
det(A)
```

3 Reševanje nelinearnih enačb

3.1 Nelinearne enačbe

1. Na intervalu $[0, 1]$ ima enačba $f(x) = x$, kjer je

$$f(x) = 3\sqrt{x}e^{-x} - \tfrac{1}{3},$$

dve rešitvi. Katero izmed teh dveh rešitev je mogoče poiskati z navadno iteracijo

$$x_{n+1} = f(x_n), \quad f(x_n) = 3\sqrt{x_n}e^{-x_n} - \tfrac{1}{3}$$

s primerno izbranim začetnim približkom x_0?

Rešitev: Najprej z Newtonovo iteracijo

$$x_{n+1} = x_n - \frac{f(x_n) - x_n}{f'(x_n) - 1}$$

poiščemo rešitvi enačbe $f(x) = x$. Izberemo en začetni približek $x_0 = 0.02$ ter drugi začetni približek $x_0 = 1.00$ in dobimo zaporedje približkov

x_0	x_1	x_2	x_3
0.02	0.0130373	0.013748	0.0137594
1.00	0.851983	0.84922	0.849218

.

Absolutna vrednost odvoda $f'(x) = \frac{3e^{-x}(1-2x)}{2\sqrt{x}}$ v izračunanih točkah je

$$|f'(0.0137594)| = 12.2658 > 1$$

za prvo negibno točko ter

$$|f'(0.849218)| = 0.486293 < 1$$

za drugo negibno točko. To pomeni, da je točka 0.013759 odbojna, točka 0.849218 pa privlačna negibna točka. Torej lahko z navadno iteracijo poiščemo le drugo negibno točko.

2. Definirajte funkcijo

$$f(x) = x^3 - 2x - \tfrac{1}{2}$$

v Octave z uporabo ukaza `inline` ter anonimne funkcije (operator `@`) in izračunajte vrednost funkcije v točki $x = 1$.

Koda v Octave:

```
f = inline('x.^3-2*x-1/2','x'); f(1)
f = @(x) x.^3-2*x-1/2; f(1)
```

3. Z uporabo ukaza `fzero` poiščite vse ničle funkcije

$$f(x) = x^3 - x - \sin(x).$$

Koda v Octave:

```
f = inline('x.^3-x-sin(x)','x');
fzero(f,-1)
fzero(f,1)
fzero(f,2)
```

Rezultati: $x_1 = -1.31716296$, $x_2 = 0$, $x_3 = 1.31716296$. Za različne ničle potrebujemo različne začetne približke.

4. Z uporabo ukaza `roots` poiščite vse ničle polinoma

$$p(x) = 2x^3 - 3x^2 - 4x + 1.$$

Koda v Octave:

```
roots([2 -3 -4 1])
```

Rezultati: $x_1 = -1$, $x_2 = 0.21922359$, $x_3 = 2.28077641$. Koeficiente polinoma podamo od vodilnega proti prostemu.

5. Z uporabo pridružene matrike izračunajte vse ničle polinoma

$$p(x) = 3x^4 + 6x^3 - 39x^2 - 42x + 72.$$

Uporabite ukaza `compan` in `eig`.

Pridružena matrika:

Za polinom

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots a_1 x + a_0$$

je pridružena matrika oblike

$$A_n = \begin{bmatrix} 0 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & \cdots & \cdots & -\frac{a_{n-1}}{a_n} \end{bmatrix}.$$

Ničle danega polinoma so tedaj lastne vrednosti pridružene matrike.

Za naš primer ($n = 4$) je pridružena matrika

$$A_4 = \begin{bmatrix} 0 & 1 & & \\ & 0 & 1 & \\ & & 0 & 1 \\ -24 & 14 & 13 & -2 \end{bmatrix}.$$

Koda v Octave:

```
p = [3 6 -39 -42 72];
A = compan(p);
eig(A)
roots(p)
```

Rezultati: $x_1 = -4$, $x_2 = -2$, $x_3 = 1$, $x_4 = 3$.

6. V Octave implementirajte algoritme za metodo bisekcije, sekantno metodo ter tangentno (Newtonovo) metodo.

Metode za reševanje nelinearnih enačb $f(x) = 0$:

- Metoda bisekcije — algoritem:

dokler $|b - a| \geq \varepsilon$
$c = a + \dfrac{b-a}{2}$;
če $f(a)f(b) < 0$
  $b = c$;
sicer
  $a = c$;

Pri tem mora za začetni interval $[a, b]$ veljati pogoj $f(a)\, f(b) < 0$, ε je predpisana toleranca.

Koda v Octave (`bisekcija.m`):

```
function [x,i] = bisekcija(f,a,b,eps)
  i = 0;
  while abs(b-a) >= eps
    c = a+(b-a)/2;
    if f(a)*f(c) < 0 b = c; else a = c; end;
    i = i+1;
  end;
  x = a+(b-a)/2;
```

- Sekantna metoda — algoritem:

$$\begin{aligned}&\text{dokler } |b-a| \geq \varepsilon\\ &c = b - f(b)\frac{b-a}{f(b)-f(a)};\\ &a = b; b = c;\end{aligned}$$

Pri tem sta a in b začetna približka, ε pa je predpisana toleranca.

Koda v Octave (`sekantna.m`):

```
function [x,i] = sekantna(f,a,b,eps)
  for i = 1:100
    c = b-f(b)*(b-a)/(f(b)-f(a));
    a = b; b = c;
    if abs(b-a) < eps break; end;
  end;
  x = a+(b-a)/2;
```

- Tangentna (Newtonova) metoda — algoritem:

$$\begin{aligned}&\text{dokler } |x_{n+1}-x_n| \geq \varepsilon\\ &x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)};\end{aligned}$$

Pri tem je x_0 začetni približek, ε pa je predpisana toleranca.

Koda v Octave (`tangentna.m`):

```
function [x,i] = tangentna(f,df,x0,eps)
  for i = 1:100
    x = x0-f(x0)/df(x0);
    if abs(x-x0) < eps break; end;
    x0 = x;
  end;
```

7. Z metodo bisekcije, s sekantno metodo ter s tangentno (Newtonovo) metodo poiščite približno rešitev enačbe $f(x) = 0$, kjer je

$$f(x) = x^3 - 2x - \tfrac{1}{2}.$$

Pri prvih dveh metodah izberite za začetni interval $[-1, 1]$, pri tangentni pa za začetni približek $x_0 = 0.5$. Toleranca naj bo $\varepsilon = 10^{-12}$. Za primerjavo: točna rešitev na 20 decimalnih mest je $\hat{x} = -0.25865202250415276284$.

Koda v Octave:

```
f = @(x) x.^3-2*x-1/2;
df = @(x) 3*x.^2-2;
[xB,iB] = bisekcija(f,-1,1,1e-12)
[xS,iS] = sekantna(f,-1,1,1e-12)
[xT,iT] = tangentna(f,df,0.5,1e-12)
```

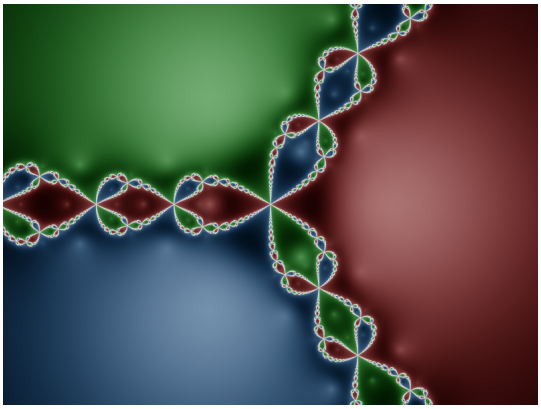
Rezultati: $x_B = -0.258652022505$, $x_S = -0.258652022504$ in $x_T = -0.258652022504$.

8. Izračunajte $\sqrt[3]{106}$ na 8 decimalnih mest natančno z uporabo metode bisekcije.
   Koda v Octave:

```
f = @(x) x.^3-106;
[xB,iB] = bisekcija(f,4,5,1e-9)
```

   Rezultat: $\sqrt[3]{106} \approx 4.73262349$.

9. Območja privlačnosti za enačbo $z^3 - 1 = 0$ v kompleksni ravnini:



10. Dokažite, da ima tangentna metoda kvadratično konvergenco.
    Rešitev: Privzemimo, da veljajo sledeči pogoji:

    - $f'(x) \neq 0$ za vsak $x \in I = [\alpha - r, \alpha + r]$ za nek $r \geq |\alpha - x_0|$,
    - $f''(x)$ je končen za vsak $x \in I$,
    - x_0 je dovolj blizu α.

    Naj α označuje rešitev enačbe $f(x) = 0$. Enakost

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \underbrace{\frac{1}{2!} f''(\xi_n)(\alpha - x_n)^2}_{\text{ostanek}} = 0,$$

    kjer je $\xi_n \in (x_n, \alpha)$, delimo z $f'(x_n)$, da dobimo

$$\frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) = -\frac{f''(\xi_n)}{2f(x_n)}(\alpha - x_n)^2.$$

    Z upoštevanjem iteracije $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ dobimo

$$(x_n - x_{n+1}) + (\alpha - x_n) = \underbrace{\alpha - x_{n+1}}_{=\varepsilon_{n+1}} = -\frac{f''(\xi_n)}{2f(x_n)} \underbrace{(\alpha - x_n)}_{=\varepsilon_n}{}^2.$$

    Ocena z absolutno vrednostjo nam da

$$|\varepsilon_{n+1}| = \frac{|f''(\xi_n)|}{2|f'(x_n)|} \varepsilon_n^2.$$

    Konvergenca je kvadratična zaradi potence 2.

11. Na dveh vrvicah dolžine $\ell = 12$cm visita kroglici enakih mas $m = 1$g. Kroglici sta prevodni in naelektreni s $Q = 2.5 \cdot 10^{-8}$As naboja. Kolikšen je kot (α) med vrvicama? Influenčna konstanta je $\epsilon_0 = 8.854184817 \cdot 10^{-12}$ F/m, težni pospešek pa $g = 9.80665$ m/s^2. Sila teže in električna sila sta

$$F_e = \frac{Q^2}{4\pi\epsilon_0 d^2}, \quad F_g = mg, \quad d = 2\ell \sin\frac{\alpha}{2}.$$

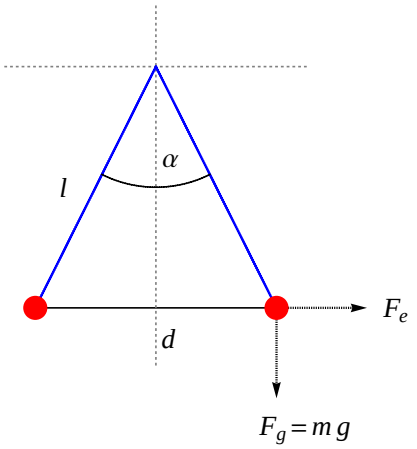
Rešitev: V ravnovesnem stanju je

$$\sum F_x = 0 \to \frac{F_e}{F_g} = \tan\frac{\alpha}{2}.$$

Označimo $u = \tan\frac{\alpha}{2}$ in $k = \frac{Q^2}{16\pi\epsilon_0 mg\ell^2}$.
Sledi enačba za u

$$u^3 - ku^2 - k = 0.$$

Slika:



Koda v Octave (uporabimo ukaz `eval`):

```
% definicija konstant
epsilon0 = 8.854184817*1e-12; g = 9.80665;
m = 1e-3; l = 0.12; Q = 2.5*1e-8;
k = Q^2/(16*pi*epsilon0*m*g*l^2);
% definicija funkcije za ukaz eval
fs = 'u.^3-k*u.^2-k';
dfs = '3*u.^2-2*k*u';
% resitev z uporabo tangentne metode
alpha = 20*pi/180; eps = 1e-8; u = tan(alpha/2);
for i=1:100
  u1 = u-eval(fs)/eval(dfs);
  if abs(u1-u)<eps, break, end;
  u = u1;
end;
fprintf('iter = %d, alfa = %0.8f stopinj\n',i,2*atan(u)/pi*180);
```

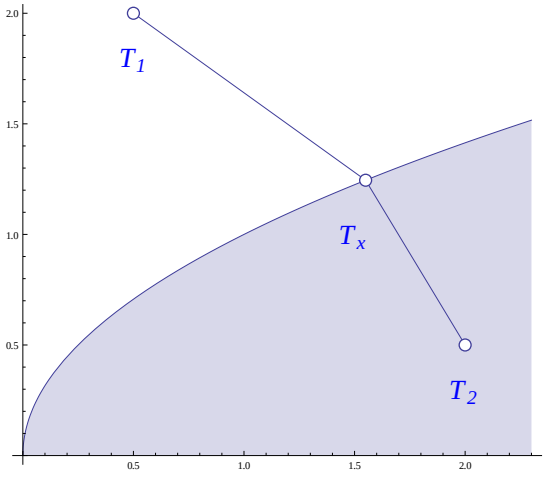
Rezultati: število iteracij je 5, kot med vrvicama je $\alpha = 24.64097133$ stopinj.

12. Svetlobni žarek potuje od točke $T_1(1/2, 2)$ do točke $T_2(2, 1/2)$. Svetlobna hitrost na svetlejšem delu območja je $c_1 = 1$, na temnejšem delu pa $c_2 = 0.3$. Mejo območja določa funkcija $f(x) = \sqrt{x}$. Poiščite koordinate točke $T_x(x, f(x))$, kjer žarek prečka mejo $y = f(x)$ (glejte spodnjo sliko). Čas potovanja žarka od točke $T_1(x_1, y_1)$ do točke $T_2(x_2, y_2)$ je

$$t(x) = \frac{\sqrt{(x_1 - x)^2 + (y_1 - f(x))^2}}{c_1} + \frac{\sqrt{(x - x_2)^2 + (f(x) - y_2)^2}}{c_2}.$$

Fermatov princip: Žarek izbere med dvema točkama takšno pot, da je čas potovanja najkrajši:

$$t'(x) = 0.$$



Rešitev nelinearne enačbe z uporabo sekantne metode.

```
t = inline('norm([0.5-x,2-sqrt(x)])+norm([x-2,sqrt(x)-0.5])/0.3','x');
eps = 1e-8; x1 = 1; x2 = 2;
for i = 1:100
   dtx1 = (t(x1+eps)-t(x1-eps))/2/eps;
   dtx2 = (t(x2+eps)-t(x2-eps))/2/eps;
   xi = x2-dtx2*(x2-x1)/(dtx2-dtx1);
   x1 = x2; x2 = xi;
   if abs(x1-x2) < eps, break, end;
end;
fprintf('Tx(%0.6f,%0.6f)\n',xi,sqrt(xi));
```

Rezultat: $T_x(1.550027, 1.245001)$.

3.2 Sistemi nelinearnih enačb

1. Poiščite presečišči krožnice $x^2 + y^2 = 4$ in parabole $y = \frac{1}{3}(x-1)^2 - \frac{1}{2}$ z uporabo tangentne (Newtonove) metode za reševanje sistema nelinearnih enačb $f(x,y) = 0$ in $g(x,y) = 0$:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \begin{bmatrix} \frac{\partial f}{\partial x}(x_n, y_n) & \frac{\partial f}{\partial y}(x_n, y_n) \\ \frac{\partial g}{\partial x}(x_n, y_n) & \frac{\partial g}{\partial y}(x_n, y_n) \end{bmatrix}^{-1} \begin{bmatrix} f(x_n, y_n) \\ g(x_n, y_n) \end{bmatrix}.$$

Za začetna približka izberite $(-1, 0)$ za levo in $(2, 0)$ za desno točko. Krožnico in parabolo tudi narišite. Sistem rešite še z uporabo vgrajenih ukazov `fsolve` in `fminsearch`.

Tangentna (Newtonova) metoda za sisteme:

Pri danem začetnem približku $X_0 = \begin{bmatrix} x_1^0 \\ \vdots \\ x_n^0 \end{bmatrix}$ imamo shemo

$$X_{k+1} = X_k - J^{-1}(X_k)\, F(X_k),$$

kjer je $X_k = \begin{bmatrix} x_1^k \\ \vdots \\ x_n^k \end{bmatrix}$, $X_{k+1} = \begin{bmatrix} x_1^{k+1} \\ \vdots \\ x_n^{k+1} \end{bmatrix}$, $F(X_k) = \begin{bmatrix} f_1(x_1^k, \ldots, x_n^k) \\ \vdots \\ f_2(x_1^k, \ldots, x_n^k) \end{bmatrix}$ in $J(X_k)$ Jacobijeva matrika odvodov

$$J(X_k) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x_1, \ldots, x_n) & \cdots & \frac{\partial f_1}{\partial x_n}(x_1, \ldots, x_n) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x_1, \ldots, x_n) & \cdots & \frac{\partial f_n}{\partial x_n}(x_1, \ldots, x_n) \end{bmatrix}.$$

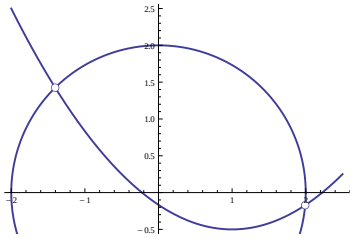
Koda v Octave:

```
F = @(x,y) [x.^2+y.^2-4;1/3*(x-1).^2-1/2-y];
J = @(x,y) [2*x,2*y;2/3*(x-1),-1];
x0 = [-1;0]; eps = 1e-8;
for i=1:100
  x1 = x0-J(x0(1),x0(2))\F(x0(1),x0(2));
  if abs(x1-x0)<eps, break, end;
  x0 = x1;
  fprintf('x%d = %0.8f, y%d = %0.8f\n',i,x1(1),i,x1(2));
end;
```

Prvih 6 približkov za levo točko $x_0 = -1$, $y_0 = 0$:

```
x1 = -2.50000000, y1 = 2.83333333
x2 = -1.70274390, y2 = 1.72306911
x3 = -1.43249027, y3 = 1.44799062
x4 = -1.40349916, y4 = 1.42532257
x5 = -1.40320722, y5 = 1.42513495
x6 = -1.40320719, y6 = 1.42513493
```

Grafični prikaz:



Prvi 3 približki za desno točko $x_0 = 2$, $y_0 = 0$:

```
x1 = 2.00000000, y1 = -0.16666667
x2 = 1.99264706, y2 = -0.17156863
x3 = 1.99262792, y3 = -0.17156327
```

Koda v Octave (slika):

```
[x,y] = meshgrid(-4:0.01:4,-4:0.01:4);
f = @(x,y) x.^2+y.^2-4;
g = @(x,y) y-(x-1).^2/3+1/2;
z = f(x,y);
w = g(x,y);
contour(x,y,z,[0,0],'r'); axis equal; hold on;
contour(x,y,w,[0,0],'b');
mesh(x,y,z);
surf(x,y,z);
```

Koda v Octave (vgrajeni ukazi):

Z ukazom `fminsearch` iščemo stacionarno točko funkcije $F(f,g) = f^2 + g^2$, kar je ekvivalentno iskanju rešitve sistema $f = 0$ in $g = 0$.

```
fsolve('[x(1).^2+x(2).^2-4; x(2)-(x(1)-1).^2/3+1/2]',[-1;0])
fsolve('[x(1).^2+x(2).^2-4; x(2)-(x(1)-1).^2/3+1/2]',[2;0])
fminsearch('(x(1).^2+x(2).^2-4).^2+(x(2)-(x(1)-1).^2/3+1/2).^2',[-1;0])
fminsearch('(x(1).^2+x(2).^2-4).^2+(x(2)-(x(1)-1).^2/3+1/2).^2',[2;0])
```

4 Predoločeni sistemi in aproksimacija

4.1 Metoda najmanjših kvadratov

1. Poiščite premico, ki v smislu najmanjših kvadratov, najbolje aproksimira točke (x_i, y_i), kjer je $\mathbf{x} = (0, 1, 2, 3, 4)$ in $\mathbf{y} = (1.5, 2.5, 3.0, 3.5, 3.7)$.

   Metoda najmanjših kvadratov:

   Enačba premice: $y = \alpha\, x + \beta$. Minimiziramo funkcijo

   $$S(\alpha, \beta) = \sum_{i=1}^{m} (y_i - \alpha x_i - \beta)^2.$$

   Rešitev $\mathbf{x}$ predoločenega sistema $A\mathbf{x} = \mathbf{b}$, $A \in \mathbb{R}^{m\times n}$, $m > n$, kjer je matrika A polnega ranga, je tisti vektor $\mathbf{x} \in \mathbb{R}^n$, za katerega velja

   $$\|A\mathbf{x} - \mathbf{b}\|_2^2 = \min_{\mathbf{u}} \|A\,\mathbf{u} - \mathbf{b}\|_2^2, \quad \mathbf{u} \in \mathbb{R}^n.$$

   Poiščemo ga tako, da bodisi rešimo normalni sistem

   $$A^T A\,\mathbf{x} = A^T \mathbf{b}$$

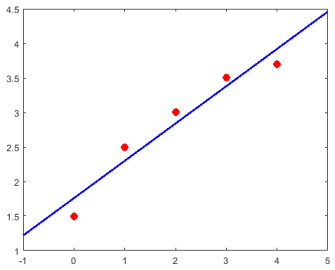
   bodisi uporabimo QR razcep matrike A

   $$A = QR, \quad R\,\mathbf{x} = Q^T \mathbf{b},$$

   kjer je Q matrika z ortonormiranimi stolpci, za katero velja $Q^T Q = I$, in R zgornja trikotna matrika s pozitivnimi diagonalnimi elementi.

   Koda v Octave:

```
x = (0:4)'; y = [1.5;2.5;3.0;3.5;3.7];
A = [x ones(size(x))];
a = (A'*A)\(A'*y); fprintf('a = (%f,%f)\n',a);
[Q,R] = qr(A);
a = R\(Q'*y); fprintf('a = (%f,%f)\n',a);
xx = -1:0.01:5; yy = a(1)*xx+a(2);
plot(xx,yy,'b',x,y,'ro','LineWidth',2,'MarkerFaceColor','r')
```



   Rezultat: $y = 0.54x + 1.76$.

2. Določite konstanti α in β tako, da bo linearna kombinacija $\alpha e^{-x} + \beta x$ aproksimirala točke (x_i, y_i) v smislu najmanjših kvadratov, kjer je $\mathbf{x} = (0, 1, 2, 3, 4, 5)$ in $\mathbf{y} = (5, 1, -1.5, -2.5, -4, -5)$.

   Koda v Octave:

```
x = (0:5)'; y = [5;1;-1.5;-2.5;-4;-5];
A = [exp(-x) x];
a = (A'*A)\(A'*y); fprintf('a = (%f,%f)\n',a);
[Q,R] = qr(A);
a = R\(Q'*y); fprintf('a = (%f,%f)\n',a);
```

Rezultati: $\alpha = 5.03971231513792$, $\beta = -1.00017917487166$.

3. Pri merjenju višine `h` izstrelka v odvisnosti od časa `t` smo dobili naslednje podatke o višini izstrelka nad tlemi.

```
t = [1.00 1.20 1.40 1.60 1.80 2.00];
h = [10.090 10.973 11.343 11.440 11.017 10.360];
```

Aproksimirajte meritve po metodi najmanjših kvadratov s polinomom druge stopnje in določite največjo višino izstrelka.

Koda v Octave ($h(t) = \alpha t^2 + \beta t + \gamma$):

```
t = [1.00 1.20 1.40 1.60 1.80 2.00]';
h = [10.090 10.973 11.343 11.440 11.017 10.360]';
A = [t.^2 t ones(length(t),1)];
k = (A'*A)\(A'*h); fprintf('k = (%f,%f,%f)\n',k);
[Q,R] = qr(A);
k = R\(Q'*h); fprintf('k = (%f,%f,%f)\n',k);
r = roots(k);
H = polyval(k,mean(r))
```

Rezultati: $\alpha = -4.853571$, $\beta = 14.786286$, $\gamma = 0.177857$, $H = 11.4394$.

4. Določite konstanti α in β tako, da bo funkcija

$$f(x) = \frac{\alpha}{1 + \beta e^{-x}}$$

aproksimirala točke (x_i, y_i), kjer je $\mathbf{x} = (-1, -0.6, -0.2, 0.2, 0.6, 1, 1.4, 1.8, 2.2, 2.6, 3)$ in $\mathbf{y} = (0.4, 0.53, 0.72, 0.89, 1.11, 1.23, 1.53, 1.6, 1.7, 1.85, 1.87)$. Nelinearni problem aproksimacije linearizirajte in ga nato rešite po metodi najmanjših kvadratov.

Rešitev: Linearizacijo funkcije

$$y = \frac{\alpha}{1 + \beta e^{-x}}$$

naredimo tako, da izraz obrnemo in dobimo

$$\frac{1}{y} = \frac{1}{\alpha} + \frac{\beta}{\alpha} e^{-x},$$

oziroma

$$Y = K + LX,$$

kjer je $X = e^{-x}$, $Y = \frac{1}{y}$, $K = \frac{1}{\alpha}$ in $L = \frac{\beta}{\alpha}$. Sledi: $\alpha = \frac{1}{K}$ in $\beta = \frac{L}{K}$.

Koda v Octave:

```
x = (-1:0.4:3)';
y = [0.4;0.53;0.72;0.89;1.11;1.23;1.53;1.6;1.7;1.85;1.87];
X = exp(-x); Y = 1./y;
A = [ones(size(X)) X];
k = (A'*A)\(A'*Y); K = k(1); L = k(2);
alfa = 1/K, beta = L/K
xx = -2:0.01:4;
yy = alfa./(1+beta*exp(-xx));
plot(xx,yy,'b',x,y,'ro');
```

Rezultati: $\alpha = 1.99700545312280$, $\beta = 1.48281571444400$.

4.2 QR in SVD razcep

1. Dan je predoločen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -3 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 4 \\ 6 \\ -1 \\ 2 \end{bmatrix}.$$

   a) Sistem rešite z uporabo normalnega sistema.

   b) Sistem rešite z uporabo QR razcepa.

   Rešitev:

   a) Normalni sistem je $A^TAx = A^Tb$. Najprej množimo

$$\begin{aligned} A^TA &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & -3 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -3 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & 0 \\ -4 & 0 & 12 \end{bmatrix}, \\ A^Tb &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & -3 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 6 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ -3 \\ -21 \end{bmatrix}. \end{aligned}$$

   Rešitev tako dobljenega sistema enačb je $x = \begin{bmatrix} \frac{9}{2} & -\frac{3}{2} & -\frac{1}{4} \end{bmatrix}^T$.

   b) QR razcep napravimo z modificiranim Gram-Schmidtovim postopkom.
   1. korak:

$$\begin{aligned} r_{1,1} &= \|a_1\| = \| \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T \| = \sqrt{2}, \\ q_1 &= \frac{a_1}{r_{1,1}} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \\ r_{1,2} &= q_1^Ta_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} = 0, \\ r_{1,3} &= q_1^Ta_3 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T \cdot \begin{bmatrix} -1 \\ -3 \\ 1 \\ 1 \end{bmatrix} = -2\sqrt{2}, \\ q_2 &= a_2 - r_{1,2}q_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}, \\ q_3 &= a_3 - r_{1,3}q_1 = \begin{bmatrix} -1 \\ -3 \\ 1 \\ 1 \end{bmatrix} + 2\sqrt{2} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned}$$

2. korak:

$$\begin{aligned} r_{2,2} &= \|q_2\| = \| \begin{bmatrix} 0 & 0 & 1 & -1 \end{bmatrix}^T \| = \sqrt{2}, \\ q_2 &= \frac{q_2}{r_{2,2}} = \frac{\sqrt{2}}{2} \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}, \\ r_{2,3} &= q_2^T q_3 = \frac{\sqrt{2}}{2} \begin{bmatrix} 0 & 0 & 1 & -1 \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} = 0, \\ q_3 &= q_3 - r_{2,3} q_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned}$$

3. korak:

$$\begin{aligned} r_{3,3} &= \|q_3\| = \| \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix}^T \| = 2, \\ q_3 &= \frac{q_3}{r_{3,3}} = \frac{1}{2} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned}$$

Sledi $A = QR$, kjer je

$$Q = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{1}{2} \\ 0 & \frac{\sqrt{2}}{2} & \frac{1}{2} \\ 0 & -\frac{\sqrt{2}}{2} & \frac{1}{2} \end{bmatrix} \quad \text{in} \quad R = \begin{bmatrix} \sqrt{2} & 0 & -2\sqrt{2} \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

Rešitev sistema $Ax = QRx = b$, oziroma $Rx = Q^T b$ je $x = \begin{bmatrix} \frac{9}{2} & -\frac{3}{2} & -\frac{1}{4} \end{bmatrix}^T$.

2. S pomočjo Householderjevih zrcaljenj in QR razcepa rešite sistem enačb

$$\begin{aligned} 2\alpha + 2\beta + 6\gamma &= 6, \\ 2\alpha + \beta - 2\gamma &= -1, \\ \alpha + 6\beta - 2\gamma &= -7. \end{aligned}$$

Householderjevo zrcaljenje:

Householderjeva zrcaljenja uporabimo za izračun QR razcepa matrike koeficientov A. Vektor $\mathbf{x}$ označuje stolpec (pod)matrike A. Najprej izračunamo vektor

$$\mathbf{w} = \begin{bmatrix} x_1 + \operatorname{sign}(x_1)\|\mathbf{x}\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Nato določimo matriko $P = I - \frac{2}{\mathbf{w}^T\mathbf{w}} \mathbf{w}\mathbf{w}^T$, za katero velja: $P = P^T$ in $PP^T = I$. Pri tem je $P\mathbf{x}$ zrcalna slika vektorja $\mathbf{x}$ čez (hiper)ravnino skozi izhodišče, ki ima za normalo vektor $\mathbf{w}$.

Rešitev: Matrika koeficientov danega sistema je $A = \begin{bmatrix} 2 & 2 & 6 \\ 2 & 1 & -2 \\ 1 & 6 & -2 \end{bmatrix}$.

Prvi korak: iz $\mathbf{x} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$ (prvi stolpec matrike A) in $\|\mathbf{x}\|_2 = 3$ dobimo $\mathbf{w} = \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}$ in

$$P_1 = I - \frac{2}{30}\begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} 5 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} \frac{5}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{4}{15} & \frac{2}{15} \\ \frac{1}{3} & \frac{2}{15} & \frac{1}{15} \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \\ -\frac{2}{3} & \frac{11}{15} & -\frac{2}{15} \\ -\frac{1}{3} & -\frac{2}{15} & \frac{14}{15} \end{bmatrix}.$$

Sledi $P_1 A = \begin{bmatrix} -3 & -4 & -2 \\ 0 & -\frac{7}{5} & -\frac{26}{5} \\ 0 & \frac{24}{5} & -\frac{18}{5} \end{bmatrix}$.

Drugi korak: iz $\mathbf{x} = \begin{bmatrix} -\frac{7}{5} \\ \frac{24}{5} \end{bmatrix}$ (drugi stolpec matrike $P_1 A$ od diagonale navzdol) in $\|\mathbf{x}\|_2 = 5$ dobimo $\mathbf{w} = \begin{bmatrix} -\frac{32}{5} \\ \frac{24}{5} \end{bmatrix}$ in

$$\tilde{P}_2 = I - \frac{2}{64}\begin{bmatrix} -\frac{32}{5} \\ \frac{24}{5} \end{bmatrix}\begin{bmatrix} -\frac{32}{5} & \frac{24}{5} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} \frac{32}{25} & -\frac{24}{25} \\ -\frac{24}{25} & \frac{18}{25} \end{bmatrix} = \begin{bmatrix} -\frac{7}{25} & \frac{24}{25} \\ \frac{24}{25} & \frac{7}{25} \end{bmatrix},$$

oziroma $P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{7}{25} & \frac{24}{25} \\ 0 & \frac{24}{25} & \frac{7}{25} \end{bmatrix}$.

Matriki Q in R v QR razcepu matrike A sta

$$Q = P_1 P_2 = \begin{bmatrix} -\frac{2}{3} & -\frac{2}{15} & -\frac{11}{15} \\ -\frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \\ -\frac{1}{3} & \frac{14}{15} & \frac{2}{15} \end{bmatrix} \quad \text{in} \quad R = P_2 P_1 A = \begin{bmatrix} -3 & -4 & -2 \\ 0 & 5 & -2 \\ 0 & 0 & -6 \end{bmatrix}.$$

Sistem $A\mathbf{x} = b$, kjer je $\mathbf{b} = \begin{bmatrix} 6 \\ -1 \\ -7 \end{bmatrix}$ pretvorimo v sistem $R\mathbf{x} = Q^T\mathbf{b}$, kjer je $Q^T\mathbf{b} = \begin{bmatrix} -1 \\ -7 \\ -6 \end{bmatrix}$.

Le-tega rešimo z obratnim vstavljanjem:

$$\begin{bmatrix} -3 & -4 & -2 \\ 0 & 5 & -2 \\ 0 & 0 & -6 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} -1 \\ -7 \\ -6 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Rezultati: $\alpha = 1$, $\beta = -1$, $\gamma = 1$.

Opomba: Z ustrezno implementacijo programa, tako da ne tvorimo matrik, lahko izračunamo QR razcep z bistveno manj računskimi operacijami.

3. Z uporabo QR iteracije izračunajte (realne) lastne vrednosti matrike $A = \begin{bmatrix} 8 & 0 & 1 & 2 \\ 8 & 3 & 1 & 1 \\ 3 & 7 & 1 & 0 \\ 8 & 0 & 5 & 6 \end{bmatrix}$.

   Rezultat preverite z uporabo vgrajenega ukaza `eig`.

   QR iteracija:

   S QR itarcijo matrike A dobimo bločno zgornjo trikotno matriko, kjer so na diagonali bloki dimenzije 1×1 za realne lastne vrednosti in dimenzije 2×2 za kompleksne lastne vrednosti.

Algoritem za QR iteracijo:

$A_0 = A;$
for n $= 0 : N$
  $A_n = Q_n R_n;$
  $A_{n+1} = R_n Q_n;$
end

Koda v Octave:

```
A = [8 0 1 2; 8 3 1 1; 3 7 1 0; 8 0 5 6]; B = A;
for i=1:20
  [Q,R] = qr(B);
  B = R*Q;
end; B
eig(A)
```

Rezultati: $\lambda_1 = 12.6943$, $\lambda_2 = -0.4469$, $\lambda_{3,4} = 2.8763 \pm 2.8187i$.

Opomba: S QR iteracijo smo dobili obe realni lastni vrednosti. Kletka dimenzije 2×2 pa v tem primeru ne konvergira k podmatriki, ki bi imela ustrezne kompleksne lastne vrednosti.

4. Dan je predoločen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -3 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 4 \\ 6 \\ -1 \\ 2 \end{bmatrix}.$$

   a) Sistem rešite v Octave z uporabo normalnega sistema.

   b) Sistem rešite v Octave z uporabo QR razcepa.

   c) Sistem rešite v Octave z uporabo singularnega (SVD) razcepa.

   Rešitev:

   a) Normalni sistem je $A^T Ax = A^T b$.

      Koda v Octave:

      ```
      A = [1 0 -1; 1 0 -3; 0 1 1; 0 -1 1]; b = [4; 6; -1; 2];
      x = (A'*A)\(A'*b)
      ```

   b) S QR razcepom $A = QR$ dobimo sistem $Rx = Q^T b$.

      Koda v Octave:

      ```
      A = [1 0 -1; 1 0 -3; 0 1 1; 0 -1 1]; b = [4; 6; -1; 2];
          [Q,R] = qr(A);
      x = R\(Q'*b)
      ```

   c) rang$\Sigma = 3$, torej $x = V_3 \Sigma_3^{-1} U_3^T b$.

      Koda v Octave:

      ```
      A = [1 0 -1; 1 0 -3; 0 1 1; 0 -1 1]; b = [4; 6; -1; 2];
      [U,S,V] = svd(A)
      rank(S)
      x = V(:,1:3)*inv(S(1:3,1:3))*U(:,1:3)'*b
      ```

   Rezultat: $x = \begin{bmatrix} \frac{9}{2} & -\frac{3}{2} & -\frac{1}{4} \end{bmatrix}^T$.

5. Z uporabo singularnega razcepa rešite predoločen sistem $Ax = b$, kjer je

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -3 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \quad \text{in} \quad b = \begin{bmatrix} 4 \\ 6 \\ -1 \\ 2 \end{bmatrix}.$$

Rešitev: rang$\Sigma = 3$, torej $x = V_3 \Sigma_3^{-1} U_3^T b$.

Koda v Octave:

```
A = [1 0 -1; 1 0 -3; 0 1 1; 0 -1 1]; b = [4; 6; -1; 2];
[U,S,V] = svd(A)
rank(S)
x = V(:,1:3)*inv(S(1:3,1:3))*U(:,1:3)'*b
```

Rezultat: $x = \begin{bmatrix} \frac{9}{2} & -\frac{3}{2} & -\frac{1}{4} \end{bmatrix}^T$.

6. V Octave implementirajte program `stisni.m` za kompresijo slik. Uporabite singularni razcep. Program preizkusite na konkretnem primeru.

Koda v Octave:

```
function stisni(slika)
A = imread(slika,'jpg');
AA = double(A); % slika v formatu double
A1 = AA(:,:,1); % nivo R (red, rdeca barva)
A2 = AA(:,:,2); % nivo G (green, zelena barva)
A3 = AA(:,:,3); % nivo B (blue, modra barva)
[U1,S1,V1] = svd(A1); % singularni razcep nivoja R
[U2,S2,V2] = svd(A2); % singularni razcep nivoja G
[U3,S3,V3] = svd(A3); % singularni razcep nivoja B
[m,n] = size(A1);
clc;
fprintf('Dimenzija slike je %d x %d.\n',m,n);
k = input('Vnesite rang aproksimacije (0 za konec):  ');
while k > 0
    A1k = U1(:,1:k)*S1(1:k,1:k)*V1(:,1:k)';
    A2k = U2(:,1:k)*S2(1:k,1:k)*V2(:,1:k)';
    A3k = U3(:,1:k)*S3(1:k,1:k)*V3(:,1:k)';
    AAk = zeros(size(AA));
    AAk(:,:,1) = A1k;
    AAk(:,:,2) = A2k;
    AAk(:,:,3) = A3k;
    % pretvorba iz formata double nazaj v 'slikovni' format
    Ak = uint8(round(AAk-1));
    figure(1);
    subplot(2,1,1); zoom(2);
    image(A);
    axis equal; axis off;
    subplot(2,1,2); zoom(2);
    image(Ak);
    axis equal; axis off;
    clc;
    fprintf('Dimenzija slike je %d x %d.\n',m,n);
    k = input('Vnesite rang aproksimacije (0 za konec):  ');
end;
```

5 Interpolacija

5.1 Polinomska interpolacija

1. Izračunajte približno vrednost funkcije v točki $x_0 = 1.45$ tako, da skozi dane točke $(0, 1.70)$, $(1, 0.92)$, $(2, 0.69)$ in $(3, 1.67)$ interpolirate kubičen polinom

$$y = Ax^3 + Bx^2 + Cx + D.$$

Polinomska interpolacija — klasična oblika:

Interpolacijski polinom je oblike

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Ko vstavimo točke (x_i, y_i), $i = 0, 1, \ldots, n$, dobimo sistem linearnih enačb v matrični obliki $Va = y$, kjer je

$$V = \begin{bmatrix} x_0^n & \cdots & x_0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^n & \cdots & x_n & 1 \end{bmatrix}$$

Vandermondova matrika, ki je slabo pogojena, čeprav je $\det V = \prod_{i<j} (x_i - x_j) \neq 0$.

Koda v Octave:

```
x = (0:3)'; y = [1.70;0.92;0.69;1.67];
A = [x.^3 x.^2 x ones(size(x))];
k = A\y;
x0 = 1.45; y0 = dot(k,[x0^3;x0^2;x0;1])
```

Rezultat: 0.70896125.

2. S pomočjo kubične interpolacije izračunajte približno vrednost funkcije za $x_0 = 3.271$, če poznamo le naslednje vrednosti funkcije.

```
x = [2.40 2.80 3.20 3.60 4.00 4.40];
y = [0.875 0.454 -0.033 -0.509 -0.898 -1.139];
```

Interpolacijske točke izberemo tako, da je iskana vrednost med drugo in tretjo interpolacijsko točko.

Koda v Octave:

```
x = [2.40 2.80 3.20 3.60 4.00 4.40];
y = [0.875 0.454 -0.033 -0.509 -0.898 -1.139];
x = x(2:5)'; y = y(2:5)';
A = [x.^3 x.^2 x ones(size(x))];
k = A\y;
x0 = 3.271; y0 = dot(k,[x0^3;x0^2;x0;1])
```

Rezultat: -0.120470462406249.

3. Prvo nalogo rešite z uporabo vgrajenih funkcij `polyfit` in `polyval` za delo s polinomi. Z vgrajenim ukazom `roots` poiščite še vse ničle interpolacijskega polinoma.

Koda v Octave:

```
x = (0:3)'; y = [1.70;0.92;0.69;1.67];
k = polyfit(x,y,3);
x0 = 1.45; y0 = polyval(k,x0)
roots(k)
```

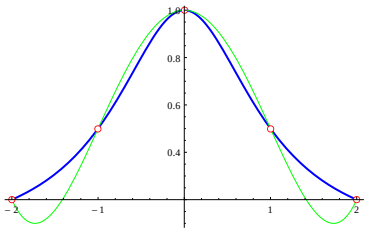
Rezultat: $y_0 = 0.70896125$, $x_1 = -3.26872354$, $x_{2,3} = 1.88436177 \pm 1.08498249i$.

Opomba: Polinomski interpolaciji v klasični obliki se v praksi izogibamo. To velja tako za reševanje sistema linearnih enačb kot za uporabo vgrajenega ukaza `polyfit`.

4. Poiščite interpolacijski polinom skozi točke (x, y), kjer je $x \in \{-2, -1, 0, 1, 2\}$, $y = f(x)$ in $f(x) = \dfrac{1}{1+x^2}$. (Rungejev primer neustrezne interpolacije.)

   Koda v Octave:

```
x = linspace(-2,2,5);
y = 1./(1+x.^2);
d = polyfit(x,y,4);
xx = linspace(-2,2);
yp = polyval(d,xx);
yy = 1./(1+xx.^2);
figure; plot(xx,yp,xx,yy,x,y,'o');
```



5. Določite Lagrangeov interpolacijski polinom p stopnje 4, ki interpolira točke $(1, 2)$, $(2, 0)$, $(4, -1)$, $(6, 3)$ in $(7, 5)$.

   Polinomska interpolacija — Lagrangeova oblika:

   Lagrangeov interpolacijski polinom je oblike

$$p_n(x) = \sum_{i=0}^{n} y_i \ell_{n,i}(x),$$

   kjer je

$$\ell_{n,i}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, 1, \ldots, n$$

   Lagrangeov bazni polinom. Pri tej obliki je potrebno poznati stopnjo polinoma, ni ekonomična.

   Rešitev: Najprej zapišemo vse Lagrangeove bazne polinome:

$$\begin{aligned}
\ell_{4,0}(x) &= \frac{1}{90}(x-2)(x-4)(x-6)(x-7), \\
\ell_{4,1}(x) &= -\frac{1}{40}(x-1)(x-4)(x-6)(x-7), \\
\ell_{4,2}(x) &= \frac{1}{36}(x-1)(x-2)(x-6)(x-7), \\
\ell_{4,3}(x) &= -\frac{1}{40}(x-1)(x-2)(x-4)(x-7), \\
\ell_{4,4}(x) &= \frac{1}{90}(x-1)(x-2)(x-4)(x-6).
\end{aligned}$$

   Lagrangeov interpolacijski polinom $p \in \mathbb{P}_4$ je

$$\begin{aligned}
p(x) &= \frac{1}{45}(x-2)(x-4)(x-6)(x-7) - \frac{1}{36}(x-1)(x-2)(x-6)(x-7) \\
&\quad + \frac{3}{40}(x-1)(x-2)(x-4)(x-7) + \frac{1}{18}(x-1)(x-2)(x-4)(x-6).
\end{aligned}$$

6. Poiščite Newtonov interpolacijski polinom p stopnje 5, za katerega velja $p(0) = 4$, $p'(0) = -4$, $p''(0) = 8$, $p(1) = 2$, $p'(1) = -1$ in $p''(1) = 1$. Izračunajte še vrednost interpolacijskega polinoma v točki $x = 2$. Uporabite prilagojeni Hornerjev algoritem.

Polinomska interpolacija — Newtonova oblika:

Newtonov interpolacijski polinom je oblike

$$\begin{aligned} p_n(x) &= [x_0]f + (x - x_0)[x_0, x_1]f + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})[x_0, x_1, \ldots, x_n]f \\ &= \sum_{i=0}^{n} [x_0, x_1, \ldots, x_i]f \prod_{j=0}^{i-1}(x - x_j), \end{aligned}$$

kjer je deljena razlika (diferenca)

$$[x_0, x_1, \ldots, x_k]f = \begin{cases} \frac{f^{(k)}(x_0)}{k!}, & x_0 = \cdots = x_k, \\ \frac{[x_1, x_2, \ldots, x_k]f - [x_0, x_1, \ldots, x_{k-1}]f}{x_k - x_0}, & \text{sicer.} \end{cases}$$

Rešitev: Z uporabo formule za deljene razlike dobimo shemo

x_i	$[.]f = f(x_i)$	$[.,.]f$	$[.,.,.]f$	$[.,.,.,.]f$	$[.,.,.,.,.]f$	$[.,.,.,.,.,.]f$
0	4					
		-4				
0	4		4			
		-4		-2		
0	4		2		1	
		-2		-1		$-\frac{1}{2}$
1	2		1		$\frac{1}{2}$	
		-1		$-\frac{1}{2}$		
1	2		$\frac{1}{2}$			
		-1				
1	2					

.

Newtonov interpolacijski polinom $p \in \mathbb{P}_5$ je

$$p(x) = 4 - 4x + 4x^2 - 2x^3 + x^3(x - 1) - \frac{1}{2}x^3(x - 1)^2.$$

Z uporabo prilagojenega Hornerjevega algoritma dobimo shemo

	$-\frac{1}{2}$	1	-2	4	-4	4
2		$-\frac{1}{2}$	$\frac{1}{2}$	-3	2	-4
	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{3}{2}$	1	-2	0

.

Sledi $p(2) = 0$.

7. Z uporabo Newtonovega interpolacijskega polinoma ter vgrajenih funkcij izračunajte približek za $\cos(0.15)$ tako, da interpolirate polinom skozi točke $(x_i, \cos(x_i))$, kjer je $x_i = i/10$, $i = 0, 1, 2, 3$. Na isto sliko narišite graf funkcije in njenega interpolacijskega polinoma.

Koda v Octave:

```
x = linspace(0,0.3,4); y = cos(x); x0 = 0.15; n = length(x);
d = newtoninterp(x,y);
dx = x0-x; y0 = d(end);
for i=n-1:-1:1
  y0 = y0*dx(i)+d(i);
end;
fprintf('y0 = %f, cos(%0.2f) = %f\n',y0, x0, cos(x0));
```

Rezultati: $y0 = 0.988769$, $\cos(0.15) = 0.988771$.

Koda v Octave (`newtoninterp.m`):

```
function D = newtoninterp(x,y)
  x = x(:); y = y(:); n = length(x);
  D = zeros(n);
  D(:,1) = y;
  for i=1:n-1
    D(1:n-i,1+i) = diff(D(1:n-i+1,i))./(x(i+1:n)-x(1:n-i));
  end;
  D = D(1,:);
```

Koda v Octave (slika):

```
xx = (-0.1:0.001:0.4)';
yy = cos(xx);
pp = zeros(size(xx));
for j=1:length(xx)
  dx = xx(j)-x; pp(j) = d(end);
  for i=n-1:-1:1
    pp(j) = pp(j)*dx(i)+d(i);
  end;
end;
plot(xx,yy,'b',xx,pp,'k',x,y,'ro');
```

8. Funkcijo $f(x) = \sin x$ interpolirajte z Newtonovim interpolacijskim polinomom stopnje 2 v točkah 0, $\frac{\pi}{6}$ in $\frac{\pi}{3}$. Ocenite napako na intervalu $[0, \frac{\pi}{3}]$.

   Rešitev: Interpolacijske točke so ekvidistantne $x_0 = 0$, $x_1 = x_0 + h = \frac{\pi}{6}$ in $x_2 = x_0 + 2h = \frac{\pi}{3}$, kjer je $h = \frac{\pi}{6}$. Dobimo shemo

x_i	$[.]f$	$[.,.]f$	$[.,.,.]f$
0	0		
		$\frac{3}{\pi}$	
$\frac{\pi}{6}$	$\frac{1}{2}$		$\frac{9\sqrt{3}-18}{\pi^2}$
		$\frac{3\sqrt{3}-3}{\pi}$	
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$		

   Newtonov interpolacijski polinom je tedaj

$$p_2(x) = \frac{3}{\pi}x + \frac{9\sqrt{3}-18}{\pi^2}x\left(x - \frac{\pi}{6}\right).$$

   Za oceno napake velja

$$\begin{aligned} \left|f(x) - p_2(x)\right| &= \left|(x-x_0)(x-x_1)(x-x_2)[x_0,x_1,x_2,x]f\right| \\ &= \left|(x-x_0)(x-x_1)(x-x_2)\right| \cdot \frac{1}{6} \cdot \left|f^{(3)}(\xi)\right| \\ &\leq \frac{2\pi^3\sqrt{3}}{9\cdot 6^3} \cdot \frac{1}{6} \cdot 1 = \frac{\pi^3\sqrt{3}}{27\cdot 6^3} \approx 9.21 \cdot 10^{-3}. \end{aligned}$$

   Pri tem upoštevamo, da je $[x_0, x_1, x_2, x]f = \frac{f^{(3)}(\xi)}{3!}$ za nek $\xi \in [0, \frac{\pi}{3}]$. Ocenimo najprej

$$\max_{x\in[0,\frac{\pi}{3}]} \left|f^{(3)}(x)\right| = \max_{x\in[0,\frac{\pi}{3}]} \left|-\cos x\right| = 1,$$

nato pa še

$$\max_{x\in[0,\frac{\pi}{3}]} \left|\omega(x)\right| = \max\left\{|\omega(0)|, |\omega(\alpha_1)|, |\omega(\alpha_2)|, |\omega(\tfrac{\pi}{3})|\right\} = \frac{2h^3\sqrt{3}}{9} = \frac{2\pi^3\sqrt{3}}{9\cdot 6^3},$$

kjer je $\omega(x) = (x-x_0)(x-x_1)(x-x_2)$ in kjer sta $\alpha_{1,2} = h\left(1 \pm \frac{\sqrt{3}}{3}\right)$ lokalna ekstrema funkcije $\omega(x)$. Le-te dobimo kot ničle odvoda funkcije $\omega(x)$. Odvajamo funkcijo

$$\omega(x) = x(x-h)(x-2h) = x^3 - 3hx^2 + 2h^2x = x^3 - \frac{\pi x^2}{2} + \frac{\pi^2 x}{18}$$

in dobimo

$$\omega'(x) = 3x^2 - 6hx + 2h^2 = 3x^2 - \pi x + \frac{\pi^2}{18}.$$

Rešitvi enačbe $\omega'(x) = 0$ sta $\alpha_{1,2} = h\left(1 \pm \frac{\sqrt{3}}{3}\right) = \frac{\pi}{6}\left(1 \pm \frac{\sqrt{3}}{3}\right)$. Velja še $\omega(0) = \omega(\frac{\pi}{3}) = 0$ in $\omega(\alpha_{1,2}) = \pm\frac{2h^3\sqrt{3}}{9}$.

9. Zapišite Newtonov interpolacijski polinom skozi točke

x_i	-2	-1	0	1	2	3
f_i	1	2	-1	-2	1	3

.

Izračunajte še vrednost interpolacijskega polinoma v točki $x = \frac{1}{2}$. Uporabite prilagojeni Hornerjev algoritem.

Rešitev: Z uporabo formule

$$[x_0,\ldots,x_k]f = \frac{[x_1,\ldots,x_k]f - [x_0,\ldots,x_{k-1}]f}{x_k - x_0}$$

dobimo shemo

x_i	$[.]f$	$[.,.]f$	$[.,.,.]f$	$[.,.,.,.]f$	$[.,.,.,.,.]f$	$[.,.,.,.,.,.]f$
-2	1					
		1				
-1	2		-2			
		-3		1		
0	-1		1		$-\frac{1}{6}$	
		-1		$\frac{1}{3}$		$-\frac{1}{40}$
1	-2		2		$-\frac{7}{24}$	
		3		$-\frac{5}{6}$		
2	1		$-\frac{1}{2}$			
		2				
3	3					

Newtonov interpolacijski polinom je tedaj

$$\begin{aligned} p(x) &= 1 + (x+2) - 2(x+2)(x+1) + (x+2)(x+1)x \\ &\quad -\frac{1}{6}(x+2)(x+1)x(x-1) - \frac{1}{40}(x+2)(x+1)x(x-1)(x-2). \end{aligned}$$

Z uporabo prilagojenega Hornerjevega algoritma dobimo shemo

	$-\frac{1}{40}$	$-\frac{1}{6}$	1	-2	1	1
$\frac{1}{2}$		$\frac{3}{80}$	$\frac{31}{480}$	$\frac{511}{960}$	$-\frac{1409}{640}$	$-\frac{769}{256}$
	$-\frac{1}{40}$	$-\frac{31}{240}$	$\frac{511}{480}$	$-\frac{1409}{960}$	$-\frac{769}{640}$	$-\frac{513}{256}$
	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	
	$\cdot(-\frac{3}{2})$	$\cdot(-\frac{1}{2})$	$\cdot\frac{1}{2}$	$\cdot\frac{3}{2}$	$\cdot\frac{5}{2}$	

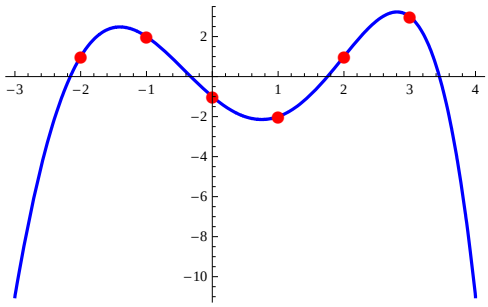
.

Sledi $p(\frac{1}{2}) = -\frac{513}{256}$.

Koda v Octave:

```
x = (-2:3)'; y = [1 2 -1 -2 1 3]'; x0 = 0.5; n = length(x);
d = newtoninterp(x,y);
dx = x0-x; y0 = d(end);
for i=n-1:-1:1
  y0 = y0*dx(i)+d(i);
end;
y0
```

Rezultat: $y0 = -2.0039$. Interpolacijski polinom je na spodnji sliki.



5.2 Zlepki

1. Točke (x, y), kjer je $x = (0, 1, 2, 3, 4)$ in $y = (1, 3, 2, 0, 3)$ interpolirajte s kubičnim zlepkom (p, q), kjer je $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, $x \in [0, 2]$ in $q(x) = b_0 + b_1x + b_2x^2 + b_3x^3$, $x \in [2, 4]$. V točki $x = 2$ naj se ujemata še prvi in drugi odvod polinomov $p(x)$ in $q(x)$. Skozi dane točke interpolirajte še polinom stopnje 4.

   Rešitev: Dobimo sistem 8 linearnih enačb za 8 neznank:

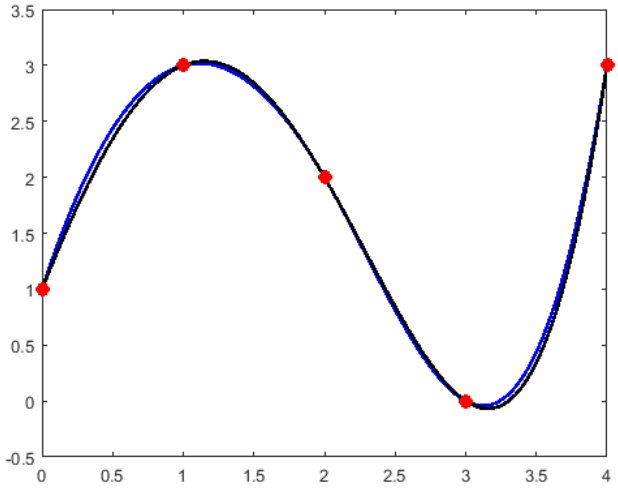
$$\begin{aligned} y_i = p(x_i) &= a_0 + a_1x_i + a_2x_i^2 + a_3x_i^3, \quad i = 1, 2, 3, \\ y_j = q(x_j) &= b_0 + b_1x_j + b_2x_j^2 + b_3x_j^3, \quad j = 3, 4, 5, \\ p'(x_3) &= q'(x_3), \\ p''(x_3) &= q''(x_3), \end{aligned}$$

   ki ga rešimo z ukazom levega deljenja v Octave.

   Koda v Octave:

```
X = [0 1 2 3 4]'; Y = [1 3 2 0 3]';
A = zeros(8); B = [Y(1:3); Y(3:5); 0; 0];
A(1:3,1:4) = [ones(3,1) X(1:3) X(1:3).^2 X(1:3).^3];
A(4:6,5:8) = [ones(3,1) X(3:5) X(3:5).^2 X(3:5).^3];
A(7,:) = [0 1 2*X(3) 3*X(3).^2 0 -1 -2*X(3) -3*X(3).^2];
A(8,:) = [0 0 2 6*X(3) 0 0 -2 -6*X(3)];
a = A\B;
x = linspace(0,4,100);
y = [polyval(a(4:-1:1),x(1:50)),polyval(a(8:-1:5),x(51:100))];
b = polyfit(X,Y,4);
z = polyval(b,x);
plot(x,y,'b',x,z,'k',X,Y,'ro','LineWidth',2,'MarkerFaceColor','r');
```

   Slika kubičnega zlepka je podana spodaj.



Rezultati: $p(x) = 1 + \frac{23}{6}x - 2x^2 + \frac{1}{6}x^3$, $q(x) = -7 + \frac{95}{6}x - 8x^2 + \frac{7}{6}x^3$.

5.3 Numerično odvajanje

1. Z metodo nedoločenih koeficientov določite uteži formule za numerično odvajanje oblike

$$f'(x) = w_1 f(x-2h) + w_2 f(x-h) + w_3 f(x)$$

tako, da bo točna za polinome stopnje ≤ 2.
Rešitev: Zapišimo sistem enačb, kjer vstavimo $f(x) \in \{1,\ x,\ x^2\}$:

$$\begin{aligned} f(x) = 1 \quad &: \quad 0 = \omega_1 + \omega_2 + \omega_3, \\ f(x) = x \quad &: \quad 1 = \omega_1(x-2h) + \omega_2(x-h) + \omega_3 x, \\ f(x) = x^2 \quad &: \quad 2x = \omega_1(x-2h)^2 + \omega_2(x-h)^2 + \omega_3 x^2. \end{aligned}$$

Rešitev mora biti neodvisna od x, kar hitro ugotovimo z upoštevanjem prve enačbe v drugi in tretji, ter dobljene druge enačbe v tretji. Po krajšanju z x dobimo sistem enačb

$$\begin{aligned} 0 &= \omega_1 + \omega_2 + \omega_3, \\ 1 &= -2\omega_1 h - \omega_2 h, \\ 0 &= 4\omega_1 h^2 + \omega_2 h^2, \end{aligned}$$

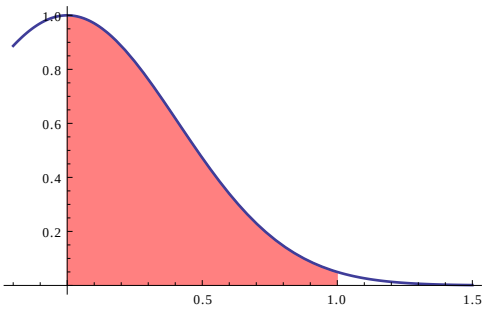
ki ima rešitev $\omega_1 = \frac{1}{2h}$, $\omega_2 = -\frac{2}{h}$ in $\omega_3 = \frac{3}{2h}$, kjer je $h > 0$. Sledi formula za numerično odvajanje oblike

$$f'(x) = \frac{1}{2h}\left(f(x-2h) - 4f(x-h) + 3f(x)\right).$$

6 Numerično integriranje

6.1 Newton–Cotesove kvadraturne formule

1. Z uporabo pravokotniškega (srednja vrednost), trapeznega, Simpsonovega tretjinskega in Simpsonovega triosminskega pravila izračunajte približno vrednost integrala $\int_0^1 e^{-3x^2}\,dx$, kjer interval $[0,1]$ razdelite na $n = 12$ enako dolgih podintervalov. Za primerjavo: točna vrednost integrala na 16 decimalnih mest je 0.5043435602314388.



Newton-Cotesove formule:

- Pravokotniško pravilo:

$$\int_a^b f(x)\,dx = h\sum_{i=1}^{n} w_i f(x_i) + \frac{(b-a)h^2}{24} f''(\xi),$$

kjer je $w_i = 1$, $h = \frac{b-a}{n}$, $x_i = a + i\,h - \frac{h}{2}$, $i = 1, \ldots, n$ in $\xi \in [a,b]$.
Koda v Octave (`pravokotnik.m`):

```
function I = pravokotnik(f,a,b,n)
  h = (b-a)/n;
  x = a+h/2:h:b-h/2;
  y = f(x);
  w = ones(1,n);
  I = h*dot(w,y);
```

- Trapezno pravilo:

$$\int_a^b f(x)\,dx = \frac{h}{2}\sum_{i=0}^{n} w_i f(x_i) - \frac{(b-a)h^2}{12} f''(\xi),$$

kjer je $w_0 = w_n = 1$, $w_i = 2$, $i = 1, \ldots, n-1$, $h = \frac{b-a}{n}$, $x_i = a + i\,h$, $i = 0, \ldots n$ in $\xi \in [a,b]$.
Koda v Octave (`trapez.m`):

```
function I = trapez(f,a,b,n)
  h = (b-a)/n;
  x = a:h:b;
  y = f(x);
  w = [1 2*ones(1,n-1) 1];
  I = h/2*dot(w,y);
```

- Simpsonovo tretjinsko pravilo:

$$\int_a^b f(x)\,dx = \frac{h}{3}\sum_{i=0}^{n} w_i f(x_i) - \frac{(b-a)h^4}{180} f^{(4)}(\xi),$$

kjer je $w_0 = w_n = 1$, $w_i = 4 - \big(1 + (-1)^i\big)$, $i = 1, \ldots, n-1$, $h = \frac{b-a}{n}$, $x_i = a + i\,h$, $i = 0, \ldots n$, $n = 2k+1$, $k \in \mathbb{N}$ in $\xi \in [a,b]$.

Koda v Octave (`simpson.m`):

```
function I = simpson(f,a,b,n)
  h = (b-a)/n;
  x = a:h:b;
  y = f(x);
  w = [1 2*ones(1,n-1) 1];
  w(2:2:end-1) = 4;
  I = h/3*dot(w,y);
```

- Simpsonovo triosminsko pravilo:

$$\int_a^b f(x)\,dx = \frac{3h}{8}\sum_{i=0}^{n} w_i f(x_i) - \frac{(b-a)h^4}{80} f^{(4)}(\xi),$$

kjer je $w_0 = w_n = 1$, $w_i = 3$, če je $\mod(i,3) \in \{2,0\}$ in $w_i = 2$, če je $\mod(i,3) = 1$, $h = \frac{b-a}{n}$, $x_i = a + i\,h$, $i = 0, \ldots n$, $n = 3k+1$, $k \in \mathbb{N}$ in $\xi \in [a,b]$.

Koda v Octave (`triosminska.m`):

```
function I = triosminska(f,a,b,n)
  h = (b-a)/n;
  x = a:h:b;
  y = f(x);
  w = [1 3*ones(1,n-1) 1];
  w(4:3:end-3) = 2;
  I = 3*h/8*dot(w,y);
```

Koda v Octave:

```
f = @(x) exp(-3*x.^2);
I = 0.5043435602314388;
Ip = pravokotnik(f,0,1,12), errp = abs(I-Ip)
It = trapez(f,0,1,12), errt = abs(I-It)
Is = simpson(f,0,1,12), errs = abs(I-Is)
I38 = triosminska(f,0,1,12), err38 = abs(I-I38)
```

Rezultati: $I_p = 0.504429679946736$, $I_t = 0.504171049685728$, $I_s = 0.504342098376087$, $I_{3/8} = 0.504340224489799$. Absolutne napake: $\mathrm{err}_p = 8.6120e{-}05$, $\mathrm{err}_t = 1.7251e{-}04$, $\mathrm{err}_s = 1.4619e{-}06$, $\mathrm{err}_{3/8} = 3.3357e-06$.

2. Z uporabo pravokotniškega (srednja vrednost), trapeznega, Simpsonovega tretjinskega in Simpsonovega triosminskega pravila izračunajte približno vrednost integrala $\int_0^2 \frac{\sin x}{x}\,dx$, kjer interval $[0,2]$ razdelite na $n = 18$ enako dolgih podintervalov. Ocenite napako pri računanju s temi pravili. Za primerjavo: točna vrednost integrala na 16 decimalnih mest je $I = 1.6054129768026968$.

Koda v Octave:

```
f = @(x) sin(x)./x;
eps = 1e-14;
I = 1.6054129768026968;
Ip = pravokotnik(f,0,2,18), errp = abs(I-Ip)
It = trapez(f,eps,2,18), errt = abs(I-It)
Is = simpson(f,eps,2,18), errs = abs(I-Is)
I38 = triosminska(f,eps,2,18), err38 = abs(I-I38)
```

Rezultati: $I_p = 1.60563699075083$, $I_t = 1.60496498653544$, $I_s = 1.60541317764024$, $I_{3/8} = 1.60541342913650$. Absolutne napake: $\mathrm{err}_p = 2.2401e-04$, $\mathrm{err}_t = 4.4799e-04$, $\mathrm{err}_s = 2.0084e-07$, $\mathrm{err}_{3/8} = 4.5233e-07$.

3. Z uporabo vgrajene funkcije `quad` izračunajte vrednost integrala $\int_0^1 e^{-2x^2}\,dx$ na 8 decimalnih mest natančno.

   Koda v Octave:

   ```
   f = @(x) exp(-2*x.^2);
   quad(f,0,1,1e-8)
   ```

   Rezultat: 0.59814401.

4. Z Rombergovo metodo izračunajte približno vrednost integrala $\int_0^1 e^{-3x^2}\,dx$. Za primerjavo: točna vrednost integrala na 16 decimalnih mest je 0.5043435602314388.

   Rombergova metoda (ekstrapolacija):

   Integral $I = \int_a^b f(x)\,dx$ izračunamo s trapeznim pravilom s korakoma h in $\frac{h}{2}$:

$$\begin{aligned} I &\approx T_h + c_1 h^2 + c_2 h^4 + \cdots \\ I &\approx T_{\frac{h}{2}} + c_1 \left(\frac{h}{2}\right)^2 + c_2 \left(\frac{h}{2}\right)^4 + \cdots \end{aligned}$$

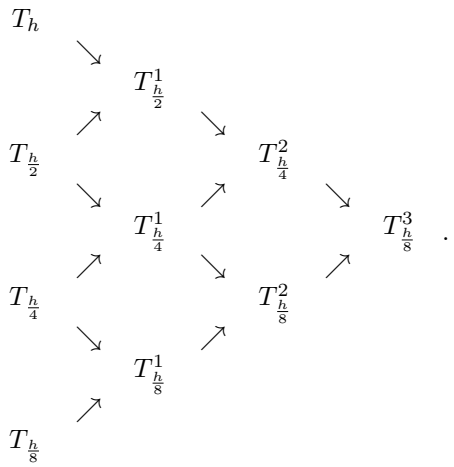
   Prvo enačbo odštejemo od štirikratnika druge ter delimo s 3, da dobimo

$$I \approx \underbrace{\frac{4T_{\frac{h}{2}} - T_h}{3}}_{T^1_{\frac{h}{2}}} + \tilde{c}_2 h^4 + \cdots$$

   Sledi splošna formula

$$T^n_{\frac{h}{2^k}} = \frac{4^n \cdot T^{n-1}_{\frac{h}{2^k}} - T^{n-1}_{\frac{h}{2^{k-1}}}}{4^n - 1}.$$

   Shematično:



   Za osnovno metodo (prvi stolpec) uporabimo trapezno pravilo ($I_{j,1} = T_{h_j}$ in $h_j = (b-a)/2^{j-1}$). V programu vrednosti shranimo v spodnjo trikotno matriko $I \in \mathbb{R}^{n\times n}$,

$$\begin{matrix} I_{1,1} & & & & \\ I_{2,1} & I_{2,2} & & & \\ I_{3,1} & I_{3,2} & I_{3,3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ I_{n,1} & I_{n,2} & I_{n,3} & \cdots & I_{n,n} \end{matrix},$$

   kjer je

$$I_{j,k} = \frac{4^{k-1} I_{j,k-1} - I_{j-1,k-1}}{4^{k-1} - 1}.$$

   Koda v Octave:

   ```
   f = @(x) exp(-3*x.^2);
   romberg(f,0,1,6)
   ```

   Rezultat: 0.504343560178457.

Koda v Octave (`romberg.m`):

```
function I = romberg(f,a,b,n)
  I = zeros(n);
  for i=1:n I(i,1) = trapez(f,a,b,2^(i-1)); end;
  for j=2:n
   for i=j:n
     I(i,j) = (4^(j-1)*I(i,j-1)-I(i-1,j-1))/(4^(j-1)-1);
    end;
  end;
  I = I(end,end);
```

6.2 Gaussove kvadraturne formule

1. Določite uteži w_1 in w_2 ter vozla x_1 in x_2 tako, da bo Gaussova kvadraturna formula

$$\int_0^1 f(x)\,dx \approx w_1 f(x_1) + w_2 f(x_2)$$

točna za funkcije $f(x) = x^n$, $n = 0, 1, 2, 3$. Posplošite formulo na intervala $[-1, 1]$ in $[a, a + h]$.

Rešitev: Iz pogojev

$$\int_0^1 x^n\,dx = w_1 x_1^n + w_2 x_2^n$$

z upoštevanjem $\int_0^1 x^n\,dx = \frac{1}{n+1}$ dobimo sistem nelinearnih enačb

$$\begin{aligned} 1 - w_1 - w_2 &= 0, \\ \tfrac{1}{2} - w_1 x_1 - w_2 x_2 &= 0, \\ \tfrac{1}{3} - w_1 x_1^2 - w_2 x_2^2 &= 0, \\ \tfrac{1}{4} - w_1 x_1^3 - w_2 x_2^3 &= 0. \end{aligned}$$

Sistem rešimo v Octave z uporabo Newtonove metode za sisteme.

Koda v Octave:

```
% Podatki:
f = @(x1,w1,x2,w2) [1-w1-w2;1/2-w1*x1-w2*x2;1/3-w1*x1^2-w2*x2^2;...
  1/4-w1*x1^3-w2*x2^3];
J = @(x1,w1,x2,w2) [0,-1,0,-1;-w1,-x1,-w2,-x2;...
  -2*w1*x1,-x1^2,-2*w2*x2,-x2^2;-3*w1*x1^2,-x1^3,-3*w2*x2^2,-x2^3];
% Newtonova metoda:
u = [0;1;1;1]; eps = 1e-8;
for i=1:100
  x1 = u(1); w1 = u(2); x2 = u(3); w2 = u(4);
  u = [x1;w1;x2;w2]-J(x1,w1,x2,w2)\f(x1,w1,x2,w2);
  if abs(u-[x1;w1;x2;w2]) < eps break; end;
end;
```

Rezultati: $w_1 = w_2 = 0.5000$, $x_1 = 0.2113$, $x_2 = 0.7887$. Z Mathematico dobimo točne rešitve zgornjega sistema: $w_1 = w_2 = \frac{1}{2}$, $x_{1,2} = \frac{1}{2} \mp \frac{\sqrt{3}}{6}$.

Posplošeni formuli:

$$\begin{aligned} \int_{-1}^{1} f(x)\,dx &\approx f\left(-\tfrac{\sqrt{3}}{3}\right) + f\left(\tfrac{\sqrt{3}}{3}\right), \\ \int_a^{a+h} f(x)\,dx &\approx \tfrac{h}{2}\left(f\left(a + \tfrac{h}{2} - \tfrac{h\sqrt{3}}{6}\right) + f\left(a + \tfrac{h}{2} + \tfrac{h\sqrt{3}}{6}\right)\right). \end{aligned}$$

2. Z Gaussovo kvadraturno formulo iz prejšnje naloge

$$\int_0^1 f(x)\,dx \approx \tfrac{1}{2}\left(f(x_1)+f(x_2)\right),$$

kjer je $x_{1,2} = \frac{1}{2} \mp \frac{\sqrt{3}}{6}$, izračunajte približno vrednost integrala $\int_0^1 e^{-x^2}\,dx$. Za primerjavo: točna vrednost integrala na 16 decimalnih mest je 0.7468241328124270.

Koda v Octave:

```
w1 = 1/2; w2 = 1/2;
x1 = 1/2-sqrt(3)/6; x2 = 1/2+sqrt(3)/6;
f = @(x) exp(-x.^2);
I = w1*f(x1) + w2*f(x2)
```

Rezultat: 0.746594688282860.

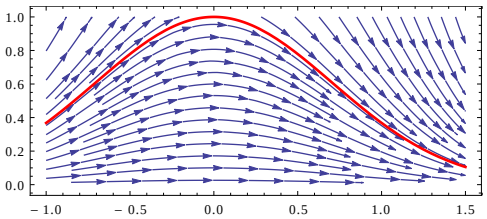
7 Numerično reševanje diferencialnih enačb

7.1 Začetni problemi

1. Z uporabo Eulerjeve, modificirane Eulerjeve, trapezne in Leapfrog metode rešite začetni problem oblike

$$y' = f(x, y), \quad y(x_0) = y_0,$$

kjer je $f(x, y) = -2xy$ in $y(0) = 1$. Interval $[0, 1]$ razdelite na $n = 10$ enako dolgih podintervalov. Točna rešitev je $y = e^{-x^2}$, točna vrednost pri $x = 1$ na 14 decimalnih mest pa je 0.36787944117144. Polje smeri:



Metode za reševanje začetnih problemov:

Rešujemo začetne probleme oblike

$$y' = f(x, y), \qquad y(x_0) = y_0.$$

- Eulerjeva metoda:

$$y_{i+1} = y_i + hf(x_i, y_i),$$

kjer je $h = \frac{b-a}{n}$, $x_0 = a$, $x_n = b$, $x_{i+1} = x_i + h$, $i = 0, 1, \ldots, n-1$.

Koda v Octave (`euler.m`):

```
function y = euler(f,a,b,n,y0)
  h = (b-a)/n; x = linspace(a,b,n+1);
  y = zeros(length(y0),n+1); y(:,1) = y0(:);
  for i = 2:n+1
    y(:,i) = y(:,i-1)+h*f(x(i-1),y(:,i-1));
  end;
  y = y(1,end);
```

- Modificirana Eulerjeva metoda:

$$y_{i+1} = y_i + hf\left(x_i + \tfrac{h}{2}, y_i + \tfrac{h}{2} f(x_i, y_i)\right),$$

kjer je $h = \frac{b-a}{n}$, $x_0 = a$, $x_n = b$, $x_{i+1} = x_i + h$, $i = 0, 1, \ldots, n-1$.

Koda v Octave (`modeuler.m`):

```
function y = modeuler(f,a,b,n,y0)
  h = (b-a)/n; x = linspace(a,b,n+1);
  y = zeros(length(y0),n+1); y(:,1) = y0(:);
  for i = 2:n+1
    k1 = f(x(i-1),y(:,i-1));
    k2 = f(x(i-1)+h/2,y(:,i-1)+h*k1(:)/2);
    y(:,i) = y(:,i-1)+h*k2(:);
  end;
  y = y(1,end);
```

- Trapezna metoda (implicitna):

$$y_{i+1} = y_i + \tfrac{h}{2}\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1})\right),$$

kjer je $h = \frac{b-a}{n}$, $x_0 = a$, $x_n = b$, $x_{i+1} = x_i + h$, $i = 0, 1, \ldots, n-1$.

Iteracijo:

$$\begin{aligned} y_{i+1}^{(0)} &= y_i, \\ y_{i+1}^{(k+1)} &= y_i + \tfrac{h}{2}\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k)})\right) \end{aligned}$$

prekinemo, ko je $\left|\frac{y_{i+1}^{(k+1)} - y_{i+1}^{(k)}}{y_{i+1}^{(k+1)}}\right| < \varepsilon$.

Koda v Octave (`trapezna.m`):

```
function y = trapezna(f,a,b,n,y0)
  h = (b-a)/n; x = linspace(a,b,n+1);
  y = zeros(length(y0),n+1); y(:,1) = y0(:);
  for i = 2:n+1
    z = y(:,i-1);
    for k=1:100
      y(:,i) = y(:,i-1)+h/2*(f(x(i-1),y(:,i-1))+f(x(i),z));
      if abs((y(:,i)-z)/y(:,i)) < 1e-10 break; end;
      z = y(:,i);
    end;
  end;
  y = y(1,end);
```

- Leapfrog metoda (večkoračna, nestabilna!):

$$y_{i+1} = y_{i-1} + 2hf(x_i, y_i),$$

kjer je $h = \frac{b-a}{n}$, $x_0 = a$, $x_n = b$, $x_{i+1} = x_i + h$, $i = 1, \ldots, n-1$. Prvi korak y_1 izračunamo z modificirano Eulerjevo metodo.

Koda v Octave (`leapfrog.m`):

```
function y = leapfrog(f,a,b,n,y0)
  h = (b-a)/n; x = linspace(a,b,n+1);
  y = zeros(length(y0),n+1); y(:,1) = y0(:);
  y(:,2) = modeuler(f,a,a+h,1,y0);
  for i = 3:n+1
    y(:,i) = y(:,i-2)+2*h*f(x(i-1),y(:,i-1));
  end;
  y = y(1,end);
```

Koda v Octave:

```
f = @(x,y) -2*x.*y;
ye = euler(f,0,1,10,1)
ym = modeuler(f,0,1,10,1)
yh = trapezna(f,0,1,10,1)
yl = leapfrog(f,0,1,10,1)
```

Rezultati: $y_e = 0.3817066806$, $y_m = 0.3671529103$, $y_h = 0.3691083539$, $y_l = 0.3653551738$.

2. Z uporabo Eulerjeve, modificirane Eulerjeve, trapezne in Leapfrog metode rešite začetni problem

$$y' = -y, \quad y(0) = 2,$$

kjer interval $[0, 2]$ razdelite na $n = 36$ enako dolgih podintervalov. Numerične rešitve primerjajte s točno rešitvijo $y = 2e^{-x}$ v točki $x = 2$: $y(2) = 0.270670566473225$. Narišite grafe točne ter numeričnih rešitev z različnimi barvami.

Koda v Octave:

```
f = @(x,y) -y;
ye = euler(f,0,2,36,2)
ym = modeuler(f,0,2,36,2)
yh = trapezna(f,0,2,36,2)
yl = leapfrog(f,0,2,36,2)
```

Rezultati: $y_e = 0.2554941623$, $y_m = 0.2709610502$, $y_h = 0.2705313040$, $y_l = 0.2709401893$.

3. Z uporabo vgrajenih ukazov `ode23` in `ode45` v Matlabu rešite začetni problem $y' = -y$, $y(0) = 2$.
   Koda v Matlabu:

```
f = @(x,y) -y;
ode23(f,[0 2],2), [x,y] = ode23(f,[0 2],2); y(end)
ode45(f,[0 2],2), [x,y] = ode45(f,[0 2],2); y(end)
```

   Rezultati: 0.270476879667586, 0.270670633436974.

4. Z uporabo Runge-Kutta metode četrtega reda

$$\begin{aligned} k_1 &= h\, f(x_i, y_i), \\ k_2 &= h\, f\left(x_i + \tfrac{h}{2}, y_i + \tfrac{k_1}{2}\right), \\ k_3 &= h\, f\left(x_i + \tfrac{h}{2}, y_i + \tfrac{k_2}{2}\right), \\ k_4 &= h\, f(x_i + h, y_i + k_3), \\ y_{i+1} &= y_i + \tfrac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad i = 0, 1, \ldots, \end{aligned}$$

   rešite začetni problem

$$y' = -3x^2 y, \quad y(0) = 2.$$

   Interval $[0, 1]$ razdelite na $n = 24$ enako dolgih podintervalov. Numerično rešitev s to metodo v točki $x = 1$ primerjajte s točno vrednostjo ter z vrednostima, dobljenima z Eulerjevo in trapezno metodo. Točna rešitev je $y = 2e^{-x^3}$: $y(1) = 0.73575888234288$.
   Koda v Octave:

```
f = @(x,y) -3*x.^2.*y;
yrk4 = rk4(f,0,1,24,2)
ye = euler(f,0,1,24,2)
yh = trapezna(f,0,1,24,2)
```

   Rezultati: $y_{rk4} = 0.735759281355605$, $y_e = 0.755036888643877$, $y_h = 0.736146451372801$.
   Koda v Octave za Runge-Kutta metodo četrtega reda (eksplicitna) (`rk4.m`):

```
function y = rk4(f,a,b,n,y0)
  h = (b-a)/n; x = linspace(a,b,n+1);
  y = zeros(length(y0),n+1); y(:,1) = y0(:);
  for i = 2:n+1
    k1 = h*f(x(i-1),y(:,i-1));
    k2 = h*f(x(i-1)+h/2,y(:,i-1)+k1(:)/2);
    k3 = h*f(x(i-1)+h/2,y(:,i-1)+k2(:)/2);
    k4 = h*f(x(i-1)+h,y(:,i-1)+k3(:));
    y(:,i) = y(:,i-1)+(k1(:)+2*k2(:)+2*k3(:)+k4(:))/6;
  end;
  y = y(1,end);
```

5. Numerično rešite začetni problem drugega reda

$$y'' + 2xy' + 2xy = (3x-2)e^{-x}, \quad y(0) = 0, \quad y'(0) = 1.$$

a) Prevedite zgornji problem na sistem dveh diferencialnih enačb prvega reda.

b) Izračunajte numerični približek za rešitev sistema v točkah $x_2 = 0.2$ in $x_{10} = 1.0$, kjer vzamete korak $h = 0.1$ in rešujete z Eulerjevo metodo.

c) Preverite, da je $y(x) = xe^{-x}$ točna rešitev, in izračunajte absolutno napako numerične rešitve v primerjavi s točno v točkah x_2 in x_{10}.

Rešitev:

a) Vzamemo novi spremenljivki $y_1 = y$ in $y_2 = y'$ za kateri dobimo sistem diferencialnih enačb

$$\begin{aligned} y_1' &= y_2 \quad \text{in} \\ y_2' &= -2xy_2 - 2xy_1 + (3x-2)e^{-x}. \end{aligned}$$

b) Eulerjeva metoda $Y_{n+1} = Y_n + h\,F(x_n, Y_n)$, $n = 0, 1, \ldots$ z začetnim pogojem Y_0 za naš primer je

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^n \\ y_2^n \end{bmatrix} + h \begin{bmatrix} f_1(x_n, y_1^n, y_2^n) \\ f_2(x_n, y_1^n, y_2^n) \end{bmatrix} = \begin{bmatrix} y_1^n \\ y_2^n \end{bmatrix} + h \begin{bmatrix} y_2^n \\ (3x_n - 2)e^{-x_n} - 2x_n y_2^n - 2x_n y_1^n \end{bmatrix},$$

kjer je $h = 0.1$ in $\begin{bmatrix} y_1^0 \\ y_2^0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Koda v Octave:

```
f = @(x,y) [y(2);(3*x-2)*exp(-x)-2*x*y(2)-2*x*y(1)];
h = 0.1; x = linspace(0,1,11);
y = zeros(2,11); y(2,1) = 1;
for i=2:11
  y(:,i)=y(:,i-1)+h*f(x(i-1),y(:,i-1));
end;
y2 = y(1,3), y10 = y(1,11)
```

Rezultati: $y_2 \approx 0.18$, $y_{10} \approx 0.37841137183645$.

c) Funkcijo $y(x) = xe^{-x}$ dvakrat odvajamo in vstavimo v diferencialno enačbo, da se prepričamo, da je to točna rešitev. Dobimo, da sta točni vrednosti na 14 decimalnih mest: $y(x_2) = 0.16374615061560$ in $y(x_{10}) = 0.36787944117144$, absolutni napaki pa: $|y_2 - y(x_2)| = 0.01625384938440$ in $|y_{10} - y(x_{10})| = 0.01053193066501$.

6. Rešite diferencialno enačbo

$$y' = -y, \quad y(0) = 1,$$

z navadno Eulerjevo in trapezno metodo. Izberite korak $h = \frac{1}{2}$ in izračunajte približno vrednost $y(1)$ z obema metodama. Funkcija y je padajoča in v limiti, ko x narašča čez vse meje, gre proti 0. Največ kolikšen je lahko korak h po eni in po drugi metodi, da numerična rešitev ohrani ti dve lastnosti?

Rešitev: Zapišimo formulo za izračun funkcijskih vrednosti y_n po obeh metodah in označimo $y_0 = y(0)$.

- Eulerjeva metoda

$$y_n = y_{n-1} - hy_{n-1} = y_{n-1}(1-h) = y_0(1-h)^n.$$

Približna vrednost za $y(1)$ je tedaj $y_2 = 0.25$.

- Trapezna metoda

$$\begin{aligned} y_n &= y_{n-1} + \tfrac{h}{2}(-y_{n-1} - y_n) \\ y_n\left(1+\tfrac{h}{2}\right) &= y_{n-1}\left(1-\tfrac{h}{2}\right) \\ y_n &= y_{n-1}\,\frac{1-\frac{h}{2}}{1+\frac{h}{2}} = y_0\left(\frac{2-h}{2+h}\right)^n \end{aligned}$$

  Približna vrednost za $y(1)$ je tedaj $y_2 = 0.36$.

Točna rešitev te diferencialne enačbe je $y(x) = e^{-x}$, vrednost $y(1)$ na 6 decimalnih mest pa je enaka $y(1) = e^{-1} \approx 0.367879$. Opazimo, da je trapezna metoda bistveno natančnejša.

Da numerična rešitev ohrani iskani lastnosti, mora biti v prvem primeru izpolnjeno $|1-h| < 1$, to pomeni, da mora biti $0 < h < 1$. V drugem primeru pa za vsak pozitiven $h > 0$ velja

$$\left|\frac{1-\frac{h}{2}}{1+\frac{h}{2}}\right| < 1.$$

Ker mora dodatno veljeti še, da je $1 - \frac{h}{2} > 0$, mora biti v drugem primeru $0 < h < 2$.

7.2 Robni problemi

1. Rešite robni problem

$$y''(x) = -2x, \quad y(0) = 0, \quad y(1) = 2,$$

kjer interval $[0, 1]$ razdelite na $n = 4$ enako dolge podintervale.

Rešitev: Z uporabo končne centralne razlike (diference) za drugi odvod

$$y''(x_i) \approx \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2},$$

kjer je $y_i \approx y(x_i)$, dobimo sistem linearnih enačb

$$y_{i-1} - 2y_i + y_{i+1} = -2h^2 x_i, \quad i = 1, 2, 3,$$

kjer je $h = \frac{1}{4}$, $y_0 = y(0) = 0$ in $y_4 = y(1) = 2$. Delilne točke so $x_0 = 0$, $x_1 = \frac{1}{4}$, $x_2 = \frac{1}{2}$, $x_3 = \frac{3}{4}$ in $x_4 = 1$. Sistem linearnih enačb

$$\begin{aligned} i = 1 \;&:\; y_0 - 2y_1 + y_2 = -\frac{1}{32}, \\ i = 2 \;&:\; y_1 - 2y_2 + y_3 = -\frac{1}{16}, \\ i = 3 \;&:\; y_2 - 2y_3 + y_4 = -\frac{3}{32}, \end{aligned}$$

zapisan v matrični obliki

$$\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{32} \\ -\frac{1}{16} \\ -\frac{67}{32} \end{bmatrix}$$

ima rešitev: $y_1 = \frac{37}{64}$, $y_2 = \frac{9}{8}$ in $y_3 = \frac{103}{64}$. Za primerjavo: točna rešitev je $y(x) = -\frac{x^3}{3} + \frac{7x}{3}$.

Nekatere končne razlike (diference):

- Centralna razlika za prvi odvod: $f'(x) \approx \dfrac{f(x+h) - f(x-h)}{2h}$ je reda $\mathcal{O}(h^2)$.
- Desna enostranska razlika za prvi odvod: $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$ je reda $\mathcal{O}(h)$.
- Leva enostranska razlika za prvi odvod: $f'(x) \approx \dfrac{f(x) - f(x-h)}{h}$ je reda $\mathcal{O}(h)$.
- Centralna razlika za drugi odvod: $f''(x) \approx \dfrac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ je reda $\mathcal{O}(h^2)$.

2. Poiščite rešitev robnega problema (Airyjeva diferencialna enačba)

$$y''(x) + x\,y(x) = 0, \quad y(a) = A, \quad y(b) = B,$$

kjer je $a = 0$, $b = 2$, $A = 0$ in $B = 1$ in interval $[a, b]$ razdelite na $n = 20$ enakih delov.
Rešitev: Približne vrednosti funkcije $y_i \approx y(x_i)$ v vozlih

$$x_i = a + i\,h, \quad i = 1, \ldots, n-1, \quad h = \frac{b-a}{n},$$

dobimo kot rešitev sistema linearnih enačb

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + x_i y_i = 0,$$

kjer je $i = 1, \ldots, n-1$, $y_0 = A$ in $y_n = B$. Sistem linearnih enačb lahko zapišemo v matrični obliki $My = v$, kjer je matrika M tridiagonalna. Razširjena matrika sistema $[M|v]$ je

$$\left[\begin{array}{ccccc|c} -2 + x_1h^2 & 1 & 0 & \cdots & 0 & -y_0 \\ 1 & -2 + x_2h^2 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 1 & -2 + x_{n-2}h^2 & 1 & 0 \\ 0 & \cdots & 0 & 1 & -2 + x_{n-1}h^2 & -y_n \end{array}\right]$$

Glavna diagonala matrike $M \in \mathbb{R}^{(n-1)\times(n-1)}$ je podana z vektorjem $\mathbf{d}$, ki ima elemente

$$d_i = -2 + x_ih^2, \quad i = 1, \ldots, n-1.$$

Prva nad- in poddiagonala sta podani z vektorjema $\mathbf{u}$ in $\mathbf{l}$, ki imata za elemente $n-2$ enojk. Desna stran sistema je podana z vektorjem $\mathbf{v}$.
Koda v Octave:

```
% glavni program
x0 = 0; xend = 2; y0 = 0; yend = 1;
n = 20; h = (xend-x0)/n;
d = -2+(1:n-1)*h^3;
l = ones(1,n-2); u = l;
v = zeros(size(d)); v(1) = -y0; v(end) = -yend;
y = trisys(u,d,l,v);
y = [y0,y,yend];
```

Program `trisys.m` reši tridiagonalni sistem linearnih enačb z uporabo Gaussove eliminacije ne da bi zapisali polno obliko matrike M.

Koda v Octave za program `trisys.m`:

```
function y = trisys(u,d,l,v)
  n = length(v);
  for k = 2:n
    mult = l(k-1)/d(k-1);
    d(k) = d(k)-mult*u(k-1);
    v(k) = v(k)-mult*v(k-1);
  end;
  y(n) = v(n)/d(n);
  for k = (n-1):-1:1
    y(k) = (v(k)-u(k)*y(k+1))/d(k);
  end;
```

3. Prejšnjo nalogo rešite še z ukazom levega deljenja, tako da z ukazom `M = sparse(I,J,V)` sestavite redko (razpršeno) matriko sistema M.

   Koda v Octave:

```
% konstrukcija redke matrike
I = [1:n-1,2:n-1,1:n-2];
J = [1:n-1,1:n-2,2:n-1];
V = [-2+(1:n-1)*h^3,ones(1,2*n-4)];
M = sparse(I,J,V);
% resevanje sistema z ukazom levega deljenja
v = zeros(n-1,1);
y0 = 0; yend = 1;
v(1) = -y0;
v(end) = -yend;
Y = M\v;
Y = [y0,Y',yend];
```

4. S strelsko metodo rešite robni problem porazdelitve temperature v cevi z notranjem radijem $\rho_1 = 1$ in zunanjim radijem $\rho_2 = 3$. Temperatura v votlini naj bo 80, zunaj pa 40. Kakšen je radialni potek temperature?

   Rešitev: Enačba za prevajanje toplote v valjnih koordinatah je

$$\frac{d^2T}{d\rho^2} + \frac{1}{\rho}\frac{dT}{d\rho} = 0,$$

   robna pogoja sta $T(\rho_1) = 80$ in $T(\rho_2) = 40$. Za začetna približka za odvod vzamemo $k_0 = -1$ in $k_1 = -2$. Za reševanje začetnega problema uporabimo modificirano Eulerjevo metodo.

   Strelska metoda:

   Rešujemo splošen (nelinearen) problem drugega reda

$$y'' = f(x, y, y'), \quad y(a) = \alpha, \quad y(b) = \beta.$$

   Vzamemo poljubno začetno vrednost $y'(a) = k_0$ in rešimo začetni problem, da dobimo rešitev $y(\cdot, k_0)$. Ker v prvem koraku najverjetneje velja $y(b, k_0) \neq \beta$, ponavljamo. Iščemo tak $k = y'(a)$, da je $y(b, k) = \beta$. Sestavimo nelinearno enačbo

$$F(k) = y(b, k) - \beta = 0$$

   in iščemo rešitev te enačbe s sekantno metodo, kjer si poljubno izberemo dva začetna približka k_0 in k_1:

$$k_{r+2} = k_{r+1} - \frac{F(k_{r+1})}{\frac{F(k_{r+1})-F(k_r)}{k_{r+1}-k_r}}, \quad r = 0, 1, 2, \ldots.$$

   Na vsakem koraku moramo rešiti začetni problem z izbrano vrednostjo $y'(a) = k_r$.

   Koda v Octave:

```
F = @(x,y) [y(2);-1/x*y(2)];
[y,x] = strelska(F,[1;3],[80;40],[-1;-2],100);
plot(x,y(1,:));
```

Koda v Octave za strelsko metodo (`strelska.m`):

```
function [y,x] = strelska(F,a,ya,u,n)
  e = 1e-10;
  h = (a(2)-a(1))/n;
  x = linspace(a(1),a(2),n+1);
  y1 = [ya(1),ya(1);u(1),u(2)];
  y2 = zeros(2,2);
  % izracun prvih dveh zadetkov za zacetne vrednosti sekantne metode
  for i = 1:2
    y = modeuler(F,a(1),a(2),n,y1(:,i)); % modificirana Eulerjeva metoda
    y2(:,i) = y(:,end);
    f(i) = y2(1,i)-ya(2);
  end;
  % sekantna metoda
  for i = 1:32
    df = diff(f);
    if abs(df) < e, return, end;
    uu = u(2)-f(2)*diff(u)/df;
    y1(:,1) = y1(:,2); y1(2,2) = uu; u = y1(2,:);
    y = modeuler(F,a(1),a(2),n,y1(:,2)); % modificirana Eulerjeva metoda
    y2(:,1) = y2(:,2); y2(:,2) = y(:,end);
    f(1) = f(2); f(2) = y2(1,2)-ya(2);
    if (abs(f(2)) < e) | abs(diff(u)) < e return; end;
  end;
```