RTU Izdevniecība

RĪGAS TEHNISKĀ UNIVERSITĀTE

SKAITLISKĀS METODES AR *MATLAB*

Andrejs Koliškins,
Valentīna Koliškina

Mācību grāmata
2023

Andrejs Koliškins,
Valentīna Koliškina

SKAITLISKĀS METODES AR *MATLAB*

Mācību grāmata

RTU Izdevniecība
Rīga 2023

Andrejs Koliškins, Valentīna Koliškina. Skaitliskās metodes ar *MATLAB*. Mācību grāmata. Rīga: RTU Izdevniecība, 2023, 218 lpp.

Mācību grāmata paredzēta kursam "Skaitliskās metodes", kas tiek lasīts visiem Rīgas Tehniskās universitātes (RTU) Datorzinātnes un informācijas tehnoloģijas fakultātes bakalaura programmas 2. kursa studentiem. Grāmatā aplūkoti skaitliskās analīzes pamatuzdevumi: lineāru vienādojumu sistēmu risināšanas metodes, interpolācija, aproksimācija, skaitliskā integrēšana, nelineāru vienādojumu un vienādojumu sistēmu risināšana, Košī problēmas risināšana parastiem diferenciālvienādojumiem. Grāmatu var lietot arī citu fakultāšu un universitāšu studenti, kā arī industrijas pārstāvji, kurus interesē skaitlisko metožu datorrealizācija un to lietojumi inženierzinātnēs.

Literārā redaktore Inga Gulbe

Datorsalikums Baiba Puriņa

Vāka dizains Paula Lore

Grāmata sagatavota publicēšanai ar RTU Datorzinātnes un informācijas tehnoloģijas fakultātes finansiālu atbalstu.

https://doi.org/10.7250/9789934229398
ISBN 978-9934-22-939-8 (pdf)

Par grāmatu

Grāmatas pamatā ir *MATLAB* lietojums (tiek pieņemts, ka lasītājs zina *MATLAB* valodas pamatus, un grāmatā ir aplūkotas *MATLAB* komandas, kuras lieto kursā "Skaitliskās metodes"). Lai vienkāršotu darbu ar grāmatu, iepazīstināsim lasītāju ar grāmatas struktūru.

Katras nodaļas sākumā ir īss teorijas izklāsts (uz zila fona).

Pēc tam ir apskatīti aprēķini *MATLAB* vidē, izmantojot konkrēto skaitlisko metodi. Aplūkoti konkrēti piemēri ar *MATLAB* skriptiem. Studenti, kuri lieto RTU portālu Ortus, var izmantot arī *MATLAB* m-failus (kas satur visu grāmatā atrisināto uzdevumu skriptus). Grāmatā ir atrodami katra atrisinātā uzdevuma *MATLAB* skripti, turklāt ir saglabāts veids, kā daži koda elementi ir redzami *MATLAB* vidē.

MATLAB skripts ir parādīts rāmīšos.

```
Matlab skripts ir parādīts rāmīšos,
komentāri attēloti zaļā krāsā,
pats skripts un dažas komandas - melnā krāsā,
bet atsevišķas iebūvētās komandas ir parādītas zilā krāsā.
```

```
Skripta rezultāti redzami zilos rāmīšos.
```

```
Galarezultāti redzami sarkanos rāmīšos.
```

Katras nodaļas beigās ir virkne uzdevumu pastāvīgai risināšanai ar atbildēm.

Par autoriem

Valentīna Koliškina ir absolvējusi Rīgas Politehniskā institūta (vēlāk – Rīgas Tehniskās universitātes) Automātikas un skaitļošanas tehnikas fakultāti specialitātē "Lietišķā matemātika". Pašlaik strādā par docenti un vadošo pētnieci Rīgas Tehniskās universitātes Inženiermatemātikas katedrā, ir matemātikas zinātņu doktore. Zinātniskās darbības virzieni: virpuļstrāvas nesagraujošās kontroles metožu matemātiskie modeļi, šķidruma plūsmas stabilitātes pētījumi.

Andrejs Koliškins ir absolvējis Rīgas Politehniskā institūta (vēlāk – Rīgas Tehniskās universitātes) Automātikas un skaitļošanas tehnikas fakultāti specialitātē "Lietišķā matemātika". Ir Rīgas Tehniskās universitātes Inženiermatemātikas katedras profesors un vadošais pētnieks, Lietišķās matemātikas institūta direktors, valsts emeritētais zinātnieks. Zinātniskās darbības virzieni: matemātiskā modelēšana, virpuļstrāvas nesagraujošās kontroles metožu matemātiskie modeļi, šķidruma plūsmas lineārā un vāji nelineārā stabilitāte.

Saturs

1. nodaļa

LINEĀRU VIENĀDOJUMU SISTĒMAS. TIEŠĀS METODES

1.1. Lineāras vienādojumu sistēmas atrisinājums

Aplūkosim n vienādojumu sistēmu ar m nezināmajiem $x_1, x_2, \ldots, x_m$:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m &= b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m &= b_n. \end{aligned} \tag{1.1}$$

Atzīmēsim, ka vispārīgā gadījumā vienādojumu skaits n nav vienāds ar nezināmo skaitu m.

Skaitļus a_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$) sauc par sistēmas koeficientiem.

Sistēmu sauc par homogēnu, ja visi koeficienti b_j ($j = 1, 2, \ldots, n$) ir vienādi ar nulli. Pretējā gadījumā sistēmu sauc par nehomogēnu.

Pārrakstīsim sistēmu (1.1) matricu veidā

$$A\boldsymbol{x} = \boldsymbol{B}, \tag{1.2}$$

kur

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}$$

ir $n \times m$ koeficientu matrica, bet $m \times 1$ matrica x un $n \times 1$ matrica $\boldsymbol{B}$ ir

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ \cdots \\ x_m \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} b_1 \\ \cdots \\ b_n \end{pmatrix}$$

Dažos gadījumos ir ērti definēt $n \times (m + 1)$ paplašināto matricu A_{aug} (*augmented matrix*)

$$A_{aug} = \begin{pmatrix} a_{11}\, a_{12} \cdots a_{1m}\, b_1 \\ \cdots\ \cdots \cdots \cdots\ \cdots \\ a_{n1}\, a_{n2} \cdots a_{nm}\, b_n \end{pmatrix}$$

Par lineārās sistēmas (1.2) atrisinājumu sauc skaitļus $x_1, x_2, \ldots, x_m$, kas apmierina n vienādojumu sistēmā. Kas ir zināms par sistēmas (1.2) atrisinājumu skaitu?

Ir iespējami tikai trīs gadījumi:

- viens vienīgs atrisinājums;
- nav atrisinājuma;
- bezgalīgi daudz atrisinājumu.

Lai ilustrētu visus trīs gadījumus, aplūkosim piemērus.

1.1. piemērs. Atrisināt vienādojumu sistēmu $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 + x_2 = 3 \end{cases}$

Atrisinājums. $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 + x_2 = 3 \end{cases} \Big| \cdot(-2) \Rightarrow + \underline{\begin{cases} -2x_1 + 6x_2 = -10 \\ 2x_1 + x_2 = 3 \end{cases}} \Rightarrow x_1 = 2$

$7x_2 = -7 \Rightarrow x_2 = -1$

Sareizinot pirmo vienādojumu ar (–2) un saskaitot iegūto vienādojumu ar otro vienādojumu sistēmā, iegūstam $7x_2 = -7$, tādējādi $x_2 = -1$. Izmantojot pirmo vienādojumu, iegūstam $x_1 = 5 + 3 \cdot (-1) = 2$. Tas nozīmē, ka **lineāru vienādojumu sistēmai ir viens vienīgs atrisinājums** $x_1 = 2$, $x_2 = -1$.

1.2. piemērs. Atrisināt vienādojumu sistēmu $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 - 6x_2 = 7 \end{cases}$

Atrisinājums. $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 - 6x_2 = 7 \end{cases} \Big| \cdot(-2) \Rightarrow + \begin{cases} -2x_1 + 6x_2 = -10 \\ 2x_1 - 6x_2 = 7 \end{cases}$

$0 \neq -3 \Rightarrow$ **sistēmai nav atrisinājuma**

Sareizinot pirmo vienādojumu ar (–2) un saskaitot iegūto vienādojumu ar otro vienādojumu sistēmā, iegūstam $0 = -3$. Tā kā šī vienādība nav iespējama, sistēmai nav atrisinājuma.

1.3. piemērs. Atrisināt vienādojumu sistēmu $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 - 6x_2 = 10 \end{cases}$

Atrisinājums. $\begin{cases} x_1 - 3x_2 = 5 \\ 2x_1 - 6x_2 = 10 \end{cases} \Big| : 2 \Rightarrow \begin{cases} x_1 - 3x_2 = 5 \\ x_1 - 3x_2 = 5 \end{cases} \Rightarrow$ **sistēmai ir bezgalīgi daudz atrisinājumu**

Dalot otro vienādojumu ar 2, iegūstam $x_1 - 3x_2 = 5$. Šis vienādojums pilnībā sakrīt ar pirmo vienādojumu sistēmā. Tas nozīmē, ka sistēmā ir tikai viens neatkarīgais vienādojums. Svītrojot no sistēmas otro vienādojumu, iegūstam vienu vienādojumu ar diviem nezināmajiem: $x_1 - 3x_2 = 5$. Atrisinājumu var uzrakstīt šādā veidā

$$x_1 = 3x_2 + 5,$$

kur x_2 ir jebkurš reāls skaitlis.
Tādējādi sistēmai ir bezgalīgi daudz atrisinājumu.

Sistēmas (1.2) atrisinājumu metodes var sadalīt divās grupās:

- **tiešās metodes**
- **iterāciju metodes.**

Izmantojot **tiešo metodi**, sistēmas (1.2) atrisinājumu iegūstam, izmantojot galīgo aritmētisko un loģisko operāciju skaitu.

Iterāciju metodēm situācija atšķiras. Šajā gadījumā mēs konstruēsim tuvinājumus sistēmas (1.2) atrisinājumam. Iterāciju skaits, kas ir nepieciešams, lai iegūtu atrisinājumu ar konkrēto precizitāti, nav iepriekš zināms. Turklāt **iterāciju metode** var konverģēt vai diverģēt, tāpēc ir svarīgi zināt šīs metodes konverģences nosacījumus.

1.2. Gausa metode

Sāksim analīzi ar tiešajām metodēm. Viena no vispopulārākajām lineāru vienādojumu sistēmu risināšanas metodēm ir Gausa metode.

Gausa metode jeb nezināmo izslēgšanas metode sastāv no diviem posmiem. Pirmajā posmā sistēmas paplašinātās matricas struktūra tiek vienkāršota, izmantojot elementārus pārveidojumus ar rindām. Sistēmas atrisinājumu iegūst otrajā solī.

Tā kā Gausa metode ir detalizēti aplūkota matemātikas kursā, ilustrēsim risināšanas procesu ar piemēru gadījumam, kad vienādojumu sistēmai ir viens vienīgs atrisinājums.

1.4. piemērs. Atrisināt vienādojumu sistēmu, izmantojot **Gausa metodi**.

$$\begin{cases} x_1 - 2x_2 + 3x_3 = 2 \\ 2x_1 + x_2 - 2x_3 = 2 \\ 3x_1 + 4x_2 + x_3 = -6 \end{cases}$$

Atrisinājums. Sistēmas paplašinātā matrica ir:

$$A_{aug} = \begin{pmatrix} 1 & -2 & 3 & 2 \\ 2 & 1 & -2 & 2 \\ 3 & 4 & 1 & -6 \end{pmatrix}$$

Atzīmēsim, ka eksistē **viens pret vienu atbilstība** starp lineāro vienādojumu sistēmu un paplašinātās matricas struktūru. Risināšanas metodi parasti ilustrē, izmantojot elementārus pārveidojumus ar paplašinātās matricas rindām. **Pirmkārt**, atņemsim no paplašinātās matricas A_{aug} otrās rindas pirmo rindu, kas ir iepriekš sareizināta ar 2. **Analoģiski** atņemsim no matricas A_{aug} trešās rindas pirmo rindu, kas ir iepriekš sareizināta ar 3.

Rezultāts ir

$$\begin{pmatrix} 1 & -2 & 3 & 2 \\ 2 & 1 & -2 & 2 \\ 3 & 4 & 1 & -6 \end{pmatrix}_{\substack{R_2 - 2R_1 \\ R_3 - 3R_1}} \sim \begin{pmatrix} 1 & -2 & 3 & 2 \\ 0 & 5 & -8 & -2 \\ 0 & 10 & -8 & -12 \end{pmatrix}$$

Otrkārt, ir jādabū nulle otrajā kolonnā zem elementa $a_{22} = 5$. Atņemot no iegūtas matricas trešās rindas otro rindu, kas ir iepriekš sareizināta ar 2, iegūstam

$$\begin{pmatrix} 1 & -2 & 3 & 2 \\ 0 & 5 & -8 & -2 \\ 0 & 10 & -8 & -12 \end{pmatrix}_{R_3 - 2R_2} \sim \begin{pmatrix} 1 & -2 & 3 & 2 \\ 0 & 5 & -8 & -2 \\ 0 & 0 & 8 & -8 \end{pmatrix}$$

Tagad uzrakstīsim lineāro vienādojumu sistēmu, kas atbilst pārveidotajai paplašinātajai matricai:

$$\begin{cases} x_1 - 2x_2 + 3x_3 = 2 \\ 5x_2 - 8x_3 = -2 \\ 8x_3 = -8 \end{cases}$$

Tā kā iegūtās sistēmas koeficientu matrica ir augšējā trīsstūrveida matrica, atrisinājumu iegūsim, sākot ar pēdējo vienādojumu sistēmā: $8x_3 = -8$ ($\rightarrow x_3$) $= -1$. Atrisinot otro vienādojumu, iegūstam:

$$5x_2 = -2 + 8x_3 \rightarrow x_2 = \frac{-2-8}{5} = -2.$$

Nezināmo x_1 atrodam, izmantojot sistēmas pirmo vienādojumu:

$$x_1 = 2x_2 - 3x_3 + 2 \rightarrow x_1 = 2 \cdot (-2) - 3 \cdot (-1) + 2 = 1.$$

Tādējādi sistēmai ir viens vienīgs atrisinājums $x_1 = 1$, $x_2 = -2$, $x_3 = -1$.

1.3. Aprēķini *MATLAB* vidē. Gausa metode

Aplūkosim šīs procedūras realizāciju *MATLAB* vidē. Pirmkārt, *MATLAB* var izmantot, lai pārveidotu konkrēto paplašināto matricu par matricu speciālā formā (tā saucamajā *reduced row echelon* formā).

Matricu *reduced row echelon* formā var aprakstīt šādi:

- visas rindas, kas satur tikai nulles, atrodas matricas apakšējā daļā;
- pirmais nenulles elements katrā rindā ir 1 (**a *leading* 1**);
- visi pārējie elementi kolonnā, kas satur **a *leading* 1**, ir vienādi ar nulli;
- pieņemsim, ka kāda rinda satur **a *leading* 1**. Tad ***leading* 1** visās citās rindās zem konkrētās rindas atrodas pa labi no ***leading* 1** konkrētajā rindā.

Lai pārveidotu paplašināto matricu *reduced row echelon* formā, izmanto *MATLAB* komandu **`rref`**.

`rref(Aaug)`	Komanda **`rref`** pārveido paplašināto matricu A_{aug} *reduced row echelon* formā: **`rref`** → `reduced row echelon form`
`fprintf('formatSpec',A1,…,An)`	Komanda **`fprintf`** formatē datus saskaņā ar specifikāciju **`'formatSpec'`**. Formatēšana attiecas uz visiem elementiem $A_1,\ldots,A_n$.

1.5. piemērs. Atrisināt sistēmu, izmantojot **Gausa metodi**:

$$\begin{cases} 2x_1+2x_2-x_3+x_4=4 \\ 4x_1+3x_2-x_3+2x_4=6 \\ 8x_1+5x_2-3x_3+4x_4=12 \\ 3x_1+3x_2-2x_3+2x_4=6 \end{cases}$$

Atrisinājums.

Definēsim koeficientu matricu A un brīvo koeficientu matricu B (sistēmas labo pusi)

```
%% 1.5. piemērs. Gausa metode
clc, clearvars, format compact
A = [2,2,-1,1;4,3,-1,2;8,5,-3,4;3,3,-2,2];
B = [4;6;12;6];
[row,col] = size(A)
Aaug = [A B];           % divu matricu apvienojums
A_rank = rank(A)        % matricas rangs
Aaug_rank = rank(Aaug) % paplašinātās matricas rangs
sol = rref(Aaug)        % Ctrl+Enter
```

MATLAB komanda **`size(A)`** nosaka matricas A rindu skaitu (*row*) un kolonnu skaitu (*col*). Paplašinātā matrica A_{aug} ir izveidota kā divu matricu A un B apvienojums. Lineāro vienādojumu sistēmas atrisinājums ir iegūts, izmantojot iebūvēto komandu **`rref`**. Atzīmēsim, ka sistēmas paplašinātā matrica A_{aug} ir vienīgais arguments komandā **`rref`**.

Matricas ranga jēdzienu bieži izmanto, lai noskaidrotu atrisinājumu skaitu sistēmā. Atgādināsim, ka matricas rangs ir nenulles rindu skaits matricā. Izmantojot *MATLAB* skriptu 1.5. piemēram, apzīmēsim matricas A rangu ar **A_rank**, paplašinātās matricas A_{aug} rangu ar **Aaug_rank** un matricas A kolonnu skaitu ar **col**.

Jebkurai lineāro vienādojumu sistēmai ir iespējami tikai trīs gadījumi:

- viens atrisinājums → sistēma ir saderīga un noteikta
  A_rank = Aaug_rank = col;
- bezgalīgi daudz atrisinājumu → sistēma ir saderīga un nenoteikta
  A_rank = Aaug_rank < col;

- neviena atrisinājuma → sistēma ir nesaderīga
  A_rank ≠ Aaug_rank.

Tādējādi, ja koeficientu matricas un paplašinātās matricas rangi ir vienādi, tad sistēma ir saderīga. Ja rangi nav vienādi, tad sistēma ir nesaderīga. Saderīga sistēma ir noteikta, ja rangi sakrīt ar matricas A kolonnu skaitu, un nenoteikta, ja matricas A kolonnu skaits ir lielāks nekā matricas A (vai paplašinātās matricas A_{aug}) rangs. Rekomendējam lasītājam uzrakstīt *MATLAB* skriptu, kas ļauj lietotājam saprast, kāda ir konkrētā sistēma – saderīga un noteikta, saderīga un nenoteikta vai nesaderīga.

Pēc *MATLAB* skripta izpildes (sk. 1.5. piemēru) iegūstam tabulu:

```
row =
     4
col =
     4
A_rank =
     4
Aaug_rank =
     4
```

```
sol =
     1     0     0     0     1
     0     1     0     0     1
     0     0     1     0    -1
     0     0     0     1    -1
```

```
% 1.5. piemēra turpinājums
X_name = ['x1';'x2';'x3';'x4'];
X_value = sol(:,col+1);
disp('Atbilde:')
disp('LVS ir saderīga un noteikta,tai ir viens vienīgs atrisinājums')
solution = table(X_name,X_value)
```

Rezultāti ir parādīti tabulā.

```
Atbilde:
LVS ir saderīga un noteikta,tai ir viens vienīgs atrisinājums
solution =
    X_name    X_value
    ______    _______
    x1          1
    x2          1
    x3         -1
    x4         -1
```

1.6. piemērs. Atrisināt vienādojumu sistēmu, izmantojot Gausa metodi. Atbildi dot decimāldaļskaitļu formā un parasto daļu formā:

$$\begin{cases} 3x_1 - 5x_2 + 2x_3 + 4x_4 = 2 \\ 7x_1 - 4x_2 + x_3 + 3x_4 = 5 \\ 5x_1 + 7x_2 - 4x_3 - 6x_4 = 3 \end{cases}$$

Atrisinājums.

```
%% 1.6. piemērs. Gausa metode
clc, clearvars, format compact
A = [3,-5,2,4;7,-4,1,3;5,7,-4,-6]; B = [2;5;3]; [row,col] = size(A)
Aaug = [A B]; A_rank = rank(A), Aaug_rank = rank(Aaug)
sol_numval = rref(Aaug)       % decimāldaļskaitļu formā
sol_exval = sym(sol_numval)   % parastā daļskaitļa formā (simboliskā veidā)
% Ctrl+Enter
```

```
row =
     3
col =
     4
A_rank =
     2
Aaug_rank =
     3
```

```
sol_numval =
    1.0000         0   -0.1304   -0.0435         0
         0    1.0000   -0.4783   -0.8261         0
         0         0         0         0    1.0000
```

Izmantojot komandu **sym**, iegūstam paplašināto matricu simboliskā veidā.

```
sol_exval =
[ 1, 0,  -3/23,  -1/23, 0]
[ 0, 1, -11/23, -19/23, 0]
[ 0, 0,      0,      0, 1]
```

$0 \neq 1$

• Nav atrisinājuma

Atgādināsim, ka katrai paplašinātās matricas rindai atbilst vienādojums lineāro vienādojumu sistēmā. Tas nozīmē, ka pārveidotas paplašinātās matricas pēdējai rindai atbilst vienādojums $0 = 1$. Tā kā tas nav iespējams, sistēmai nav atrisinājuma. Konstatējot, ka koeficientu matricas un paplašinātās matricas rangi nav vienādi, var nonākt pie tā paša secinājuma.

```
% 1.6. piemēra turpinājums
disp('Atbilde:')
disp('LVS ir nesaderīga (nav atrisinājuma)')
```

```
Atbilde:
LVS ir nesaderīga (nav atrisinājuma)
```

1.7. piemērs. Atrast sistēmas vispārīgo atrisinājumu, izmantojot Gausa metodi:

$$\begin{cases} 9x_1 - 3x_2 + 5x_3 + 6x_4 = 4 \\ 6x_1 - 2x_2 + 3x_3 + x_4 = 5 \\ 3x_1 - x_2 + 3x_3 + 14x_4 = -8 \end{cases}$$

Atrisinājums.

```
%% 1.7. piemērs. Gausa metode
clc, clearvars, format compact
A = [9,-3,5,6;6,-2,3,1;3,-1,3,14];
B = [4;5;-8]; Aaug = [A B];
[row,col] = size(A)
A_rank = rank(A),Aaug_rank = rank(Aaug)
sol = sym(rref(Aaug)) % Ctrl+Enter
```

```
row =
     3
col =
     4
A_rank =
     2
Aaug_rank =
     2
sol =
[ 1, -1/3, 0, -13/3, 13/3]
[ 0,    0, 1,     9,   -7]
[ 0,    0, 0,     0,    0]
```

Sistēma ir saderīga (koeficientu matricas un paplašinātās matricas rangi ir vienādi), bet paplašinātās matricas rangs ir mazāks nekā vienādojumu skaits sistēmā. Tas nozīmē, ka sistēmai ir bezgalīgi daudz atrisinājumu. Lai konstruētu sistēmas vispārīgo atrisinājumu, katru vienādojumu sistēmā atrisināsim attiecībā pret nezināmo, kas satur **a *leading* 1**. Rezultāts ir:

```
% 1.7. piemēra turpinājums
syms x2 x4, X_gen = sol(:,5)-sol(:,4).*x4-sol(:,2).*x2  % Ctrl+Enter
```

Vispārīgais atrisinājums ir

```
X_gen =
 x2/3 + (13*x4)/3 + 13/3
              - 9*x4 - 7
                     0
```

```
% 1.7. piemēra turpinājums
disp('Atbilde:')
disp('LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)')
fprintf('x1 = %s, x3 = %s; \n', X_gen(1:2))
fprintf('x2 un x4 - jebkuri reāli skaitļi\n')
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x1 = x2/3 + (13*x4)/3 + 13/3, x3 = - 9*x4 - 7;
 x2 un x4 - jebkuri reāli skaitļi
```

1.8. piemērs. Atrisināt sistēmu, izmantojot Gausa metodi. Atrast koeficientu matricas determinantu un inverso matricu.

$$\begin{cases} 2x_1+3x_2+11x_3+5x_4=2 \\ x_1+x_2+5x_3+2x_4=1 \\ 2x_1+x_2+3x_3+2x_4=-3 \\ x_1+x_2+3x_3+4x_4=-3 \end{cases}$$

Atrisinājums.

```
%% 1.8. piemērs. Gausa metode
clc, clearvars, format compact
A = [2,3,11,5;1,1,5,2;2,1,3,2;1,1,3,4];
B = [2;1;-3;-3]; Aaug = [A B]; [row,col] = size(A)
A_rank = rank(A), Aaug_rank = rank(Aaug)
sol = rref(Aaug), A_det = det(A), A_inv = inv(A) % Ctrl+Enter
```

Lai aprēķinātu matricas determinantu un inverso matricu, izmanto *MATLAB* iebūvētās komandas **det** un **inv**.

```
row =
     4
col =
     4
A_rank =
     4
Aaug_rank =
     4
```

```
sol =
     1     0     0     0    -2
     0     1     0     0     0
     0     0     1     0     1
     0     0     0     1    -1
A_det =
    14
A_inv =
   -0.2857    0.2857    0.7143   -0.1429
    1.2857   -2.7857    0.2857   -0.3571
   -0.1429    0.6429   -0.1429   -0.0714
   -0.1429    0.1429   -0.1429    0.4286
```

```
% 1.8. piemēra turpinājums
disp('Atbilde:')
disp('LVS ir saderīga un noteikta (viens vienīgs atrisinājums)')
mas(:,1) = sym('x',[1 4]); mas(:,2) = sol(:,col+1);
for i = 1:col
   fprintf(' %s = %s \n',mas(i,1),mas(i,2))
end
fprintf('matricas determinants = %.f\n',A_det)
```

```
Atbilde:
 LVS ir saderīga un noteikta (viens vienīgs atrisinājums)
 x1 = -2
 x2 = 0
 x3 = 1
 x4 = -1
 matricas determinants = 14
```

1.9. piemērs. Atrisināt vienādojumu sistēmas:

a) $\begin{cases} 4x-z+2y=0 \\ x+2y+z-1=0 \\ y-z=-3 \end{cases}$ b) $\begin{cases} x_1+5x_2+4x_3=1 \\ 2x_1+10x_2+8x_3=3 \\ 3x_1+15x_2+12x_3=5 \end{cases}$ c) $\begin{cases} x-3y+2z=-1 \\ x+9y+6z=3 \\ x+3y+4z=1 \end{cases}$

Atrisinājums.

1.9. piemērs ilustrē trīs iespējamos gadījumus:

(a) sistēmai ir viens vienīgs atrisinājums;

(b) sistēmai nav atrisinājuma;

(c) sistēmai ir bezgalīgi daudz atrisinājumu.

Lai atrisinātu 1.9. piemēra uzdevumus, izmantosim arī komandu **linsolve**.

linsolve(A,B)	Komandu **linsolve** var izmantot, lai atrisinātu lineāru vienādojumu sistēmu $A\boldsymbol{x}=\boldsymbol{B}$.

```
%% 1.9. (a) piemērs
clc, clearvars, format compact
A = [4,2,-1;1,2,1;0,1,-1];
B = [0; 1; -3]; Aaug = [A B];
disp('1.9.(a) piemērs')
A_rank = rank(A), Aaug_rank = rank(Aaug)
sol = sym(rref(Aaug)), sol_1 = linsolve(A,B) % Ctrl+Enter
```

```
1.9. (a) piemērs
A_rank =
     3
Aaug_rank =
     3
sol =
[ 1, 0, 0,  1]
[ 0, 1, 0, -1]
[ 0, 0, 1,  2]
sol_1 =
     1
    -1
     2
```

```
1.9. (b) piemērs
A_rank =
     1
Aaug_rank =
     2
sol =
[ 1, 5, 4, 0]
[ 0, 0, 0, 1]
[ 0, 0, 0, 0]
Warning: Matrix is
singular to working
precision.
sol_1 =
   NaN
   Inf
  -Inf
```

```
1.9. (c) piemērs
A_rank =
     2
Aaug_rank =
     2
sol =
[ 1, 0,   3,   0]
[ 0, 1, 1/3, 1/3]
[ 0, 0,   0,   0]
Warning: Matrix is
singular to working
precision.
sol_1 =
   NaN
   NaN
   NaN
```

Visos gadījumos, lai atrisinātu lineāru vienādojumu sistēmu, izmantojām gan **rref**, gan arī **linsolve**. Abas komandas dod to pašu rezultātu 1.9. (a) piemērā (gadījumā, kad sistēmai ir viens vienīgs atrisinājums). Tomēr **linsolve** dod nepareizu atbildi 1.9. (b) piemērā (sistēmai nav atrisinājuma) un 1.9. (c) piemērā (sistēmai ir bezgalīgi daudz atrisinājumu).

Piezīme. Iesakām izmantot **linsolve** tikai tad, kad ir zināms, ka lineāru vienādojumu sistēmai ir viens vienīgs atrisinājums. Ja lineāru vienādojumu sistēmas atrisinājumu skaits nav iepriekš zināms, tad iesakām izmantot **rref**, nevis **linsolve**.

1.10. piemērs. Atrisināt vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu un atrast y, ja $z=3$:

$$\begin{cases} x+2z-3y=1 \\ y+2x-4z-5=0 \\ 5x+2z=8+8y \end{cases}$$

Atrisinājums.

```
%% 1.10. piemērs. Gausa metode
clc, clearvars, format compact
A = [1,-3,2;2,1,-4;5,-8,2]; B = [1;5;8]; Aaug = [A B];
[row,col] = size(A), A_rank = rank(A), Aaug_rank = rank(Aaug)
sol = sym(rref(Aaug)) % Ctrl+Enter
```

```
row =
     3
col =
     3
A_rank =
     2
Aaug_rank =
     2
```

```
sol =
[ 1, 0, -10/7, 16/7]
[ 0, 1,  -8/7,  3/7]
[ 0, 0,     0,    0]
```

```
% 1.10. piemēra turpinājums
syms z
X_gen = sol(:,4)-sol(:,3).*z    % simboliskā izteiksme
X_part = subs(X_gen,z,3)
y_ver = X_part(2)               % Ctrl+Enter
```

```
X_gen =
 (10*z)/7 + 16/7
   (8*z)/7 + 3/7
              0
```

```
X_part =
 46/7
 27/7
    0
```

```
y_ver =
27/7
```

```
% 1.10. piemēra turpinājums
disp('Atbilde:')
disp('LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)')
fprintf('x = %s, y = %s; \n', X_gen(1:2))
fprintf('kur z-jebkurš reāls skaitlis\n\n')
fprintf(' y = %s\n ',y_ver)
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x = (10*z)/7 + 16/7, y = (8*z)/7 + 3/7;
 kur z - jebkurš reāls skaitlis

 y = 27/7
```

1.4. LU metode

Ir zināms, ka lineāru vienādojumu sistēmu (1.2) var atrisināt, izmantojot Gausa metodi. Ja vienādojumu skaits sistēmā ir liels (piemēram, vairāki tūkstoši), Gausa metode kļūst neefektīva (īpaši gadījumos, kad ir jārisina vairākas vienādojumu sistēmas ar to pašu koeficientu matricu A, bet ar atšķirīgām brīvo koeficientu matricām $\boldsymbol{B}$). Lai pārvarētu šīs grūtības, izmanto metodes, kas balstās uz koeficientu matricas sadalījumu reizinātājos. Citiem vārdiem sakot, koeficientu matricu A uzraksta kā divu matricu reizinājumu, turklāt katra reizinātāja struktūra ir vienkāršāka nekā matricai A.

Teorēma. Pieņemsim, ka visi kvadrātiskās $m \times m$ matricas A galvenie minori nav vienādi ar nulli. Tad matricu A var vienā vienīgā veidā uzrakstīt kā reizinājumu

$$A = LU, \tag{1.3}$$

kur

L ir **apakšējā trīsstūrveida matrica** ar kārtu m ar vieniniekiem uz galvenās diagonāles;

U ir **augšējā trīsstūrveida matrica** ar kārtu m ar nenulles elementiem uz galvenās diagonāles.

1. piezīme. Matricas A galvenie minori ir minori ar kārtu 1, 2, ..., m, kas atrodas matricas kreisajā augšējā stūrī.

2. piezīme. Matricas L un U ir

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ l_{m1} & l_{m2} & l_{m3} & l_{m4} & \dots & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & \dots & u_{1m} \\ 0 & u_{22} & u_{23} & u_{24} & \dots & u_{2m} \\ 0 & 0 & u_{33} & u_{34} & \dots & u_{3m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & u_{mm} \end{pmatrix}$$

Var pierādīt, ka matricu L un U elementus aprēķina, izmantojot formulas:

$$u_{ji} = a_{ji} - \sum_{k=1}^{j-1} u_{ki} l_{jk} \text{ ja } i \geq j,$$

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} u_{kj} l_{ik} \right) / u_{jj} \text{ ja } i > j,$$

$$l_{ii} = 1 \text{ ja } i = 1, 2, \dots, m$$

1.11. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot LU metodi.

$$\begin{cases} 2x_1 - x_2 + x_3 = 4 \\ 4x_1 + 3x_2 - x_3 = 6 \\ 3x_1 + 2x_2 + 2x_3 = 15 \end{cases}$$

Atrisinājums. Pirmkārt, atradīsim koeficientu matricas A sadalījumu reizinātājos L un U. Koeficientu matrica ir

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 4 & 3 & -1 \\ 3 & 2 & 2 \end{pmatrix}$$

Izmantojot formulas matricu L un U elementu aprēķināšanai, iegūstam

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3/2 & 7/10 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 13/5 \end{pmatrix}$$

Vienādojumu sistēmu pārrakstīsim šādi:

$$A\boldsymbol{x} = LU\boldsymbol{x} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3/2 & 7/10 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 13/5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 15 \end{pmatrix} \tag{1.4}$$

Definēsim palīgvektoru $\boldsymbol{y}$, izmantojot formulu

$$\begin{pmatrix} 2 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 13/5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \tag{1.5}$$

Pārrakstīsim vienādojumu (1.4):

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3/2 & 7/10 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 15 \end{pmatrix}$$

Izmantojot matricu reizināšanas kārtulu, iegūstam

$$y_1 = 4, \quad 2y_1 + y_2 = 6, \quad \frac{3}{2}y_1 + \frac{7}{10}y_2 + y_3 = 15$$

Iegūto sistēmu var viegli atrisināt. Pirmkārt, aprēķināsim $y_2 = 6 - 2y_1 = -2$. Nezināmo y_3 atrodam, izmantojot trešo vienādojumu:

$$y_3 = 15 - \frac{3}{2}y_1 - \frac{7}{10}y_2 = \frac{52}{5}$$

Tādējādi sistēmu (1.5) var pārrakstīt šādi

$$\begin{pmatrix} 2 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 13/5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ -2 \\ 52/5 \end{pmatrix}$$

no kurienes izriet, ka

$$2x_1 - x_2 + x_3 = 4, \quad 5x_2 - 3x_3 = -2, \quad \frac{13}{5}x_3 = \frac{52}{5}$$

Atrisinot sistēmu, iegūstam $x_1 = 1$, $x_2 = 2$, $x_3 = 4$.

1. piezīme. Gan **Gausa metodi**, gan arī **LU** metodi lieto praksē, izmantojot permutācijas matricu. Permutācijas ir nepieciešamas, lai izvairītos no maziem (pēc moduļa) koeficientu matricas elementiem, kas atrodas uz galvenās diagonāles. Permutācijas matrica P ir kvadrātiska matrica, kuras viens elements katrā rindā un kolonnā ir 1, bet pārējie elementi ir vienādi ar nulli. Pieņemsim, ka daži vienādojumi sistēmā (1.2) ir samainīti vietām. Šo faktu var aprakstīt, izmantojot permutācijas matricu P. Pārveidotā sistēma ir

$$PA\boldsymbol{x} = P\boldsymbol{B} \tag{1.6}$$

2. piezīme. Ņemot vērā, ka $PA = LU$, vienādojumu (1.6) var pārrakstīt šādi:

$$LU\boldsymbol{x} = P\boldsymbol{B} \tag{1.7}$$

Definējot palīgvektoru $\boldsymbol{y}$, izmantojot formulu $U\boldsymbol{x} = \boldsymbol{y}$, vienādojuma (1.7) atrisinājumu iegūsim, atrisinot divas lineāru vienādojumu sistēmas ar trīsstūrveida matricām:

$$L\boldsymbol{y} = P\boldsymbol{B}, \quad U\boldsymbol{x} = \boldsymbol{y}.$$

Katru no divām sistēmām var efektīvi atrisināt (izmantojot 1.11. piemēra risinājumu kā paraugu).

1.5. Aprēķini *MATLAB* vidē, izmantojot LU metodi

MATLAB komandu **lu** izmanto, lai konstruētu matricas A LU dekompozīciju.

[L,U,P]=lu(A)	**L** – apakšējā trīsstūrveida matrica ar vieniniekiem uz galvenās diagonāles **U** – augšējā trīsstūrveida matrica, kurai $u_{ii} \neq 0$ **P** – permutācijas matrica

1.12. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot LU metodi.

$$\begin{cases} 2x_1 + x_2 + x_3 = 7 \\ x_1 - 4x_2 + 3x_3 = 2 \\ 3x_1 + 2x_2 + 2x_3 = 13 \end{cases}$$

Atrisinājums.

```
%% 1.12. piemērs. LU dekompozīcija
clc, clearvars, format compact
A = [2,1,1;1,-4,3;3,2,2]; B = [7;2;13];
ni = fun_prob12(A);  % pārbaude: galvenie minori nav vienādi ar nulli
if ni == 2
   disp('Vismaz viens no galvenajiem minoriem ir vienāds ar nulli')
   disp('Atbilde: LU metodi nedrīkst izmantot'), return
else
   disp('Visi galvenie minori nav vienādi ar nulli')
end
```

Formulu (1.3) var izmantot ar nosacījumu, ka matricas A galvenie minori nav vienādi ar nulli.

Lai pārbaudītu šo faktu, izmantosim ārējo funkciju **fun _ prob12**. Ja vismaz viens no galvenajiem minoriem ir vienāds ar nulli, parametram **ni** tiek piešķirta vērtība 2, pretējā gadījumā **ni=1**.

```
% ārēja funkcija (1.12. piemērs). LU metode
% pārbaude: galvenie minori nav vienādi ar nulli
function ni = fun_prob12(A_mat)
   ni = 1;
   [row,col] = size(A_mat);
   for i = 1:row
      if det(A_mat(1:i,1:i))~=0
      else ni = 2; break
      end
   end
end
```

Pēc pārbaudes lineāru vienādojumu sistēmu risināsim ar LU metodi.

```
% 1.12. piemēra turpinājums
[L,U,P] = lu(A)      % Ctrl+Enter
Amat = P*L*U;        % matrica A
Y = linsolve(L,P*B) % vai Y=L\(P*B)
X = U\Y              % Ctrl+Enter
```

```
Y =
   13.0000
   -2.3333
   -1.5000
```

```
X =
    1.0000
    2.0000
    3.0000
```

```
Visi galvenie minori nav vienādi ar nulli
L =
    1.0000         0         0
    0.3333    1.0000         0
    0.6667    0.0714    1.0000
U =
    3.0000    2.0000    2.0000
         0   -4.6667    2.3333
         0         0   -0.5000
P =
     0     0     1
     0     1     0
     1     0     0
```

```
% 1.12. piemēra turpinājums
disp('Atbilde:')
fprintf('x1 = %.0f; x2 = %.0f; x3 = %.0f\n',X)
```

```
Atbilde:
 x1 = 1; x2 = 2; x3 = 3
```

Piezīme. Sākot ar 2016. gada *MATLAB* versiju, nav jāsaglabā funkcija **fun _ prob12** atsevišķā m-failā. Funkciju **fun _ prob12** var uzrakstīt pamatprogrammas beigās.

1.6. Hoļecka metode

Pieņemsim, ka A ir kvadrātiska simetriska pozitīvi definēta matrica. Atgādināsim, ka simetriskai matricai $A=A^T$, bet pozitīvi definētai matricai skalārais reizinājums $\boldsymbol{x}^T A\boldsymbol{x}>0$ katram $\boldsymbol{x}\neq\boldsymbol{0}$.

Simetrisko pozitīvi definēto matricu A var sadalīt reizinātājos, izmantojot dekompozīciju

$$A=LL^T, \tag{1.8}$$

kur tagad nav ierobežojumu attiecībā pret elementiem, kas atrodas uz galvenās diagonāles. Metodi (1.8) sauc par **Hoļecka metodi.**

Tādējādi

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots & a_{2m} \\ a_{31} & a_{32} & a_{33} & a_{34} & \dots & a_{3m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & a_{m4} & \dots & a_{mm} \end{pmatrix}= \tag{1.9}$$

$$=\begin{pmatrix} l_{11} & 0 & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ l_{m1} & l_{m2} & l_{m3} & l_{m4} & \dots & l_{mm} \end{pmatrix}\begin{pmatrix} l_{11} & l_{21} & l_{31} & l_{41} & \dots & l_{m1} \\ 0 & l_{22} & l_{32} & l_{42} & \dots & l_{m2} \\ 0 & 0 & l_{33} & l_{43} & \dots & l_{m3} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & l_{mm} \end{pmatrix}$$

Izmantojot (1.9), iegūstam formulas matricas L elementu aprēķināšanai:

$$l_{11}=\sqrt{a_{11}},\quad l_{jj}=\sqrt{a_{jj}-\sum_{s=1}^{j-1}l_{js}^2},\quad j=2,3,\dots,m$$

$$l_{j1}=\frac{a_{j1}}{l_{11}},\quad j=2,3,\dots,m$$

$$l_{jk}=\frac{1}{l_{kk}}\left(a_{jk}-\sum_{s=1}^{k-1}l_{js}l_{ks}\right),\quad j=k+1,\dots,m,\quad k\geq 2$$

1.13. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot Hoļecka metodi.

$$\begin{cases} 4x_1+2x_2+14x_3=14 \\ 2x_1+17x_2-5x_3=-101 \\ 14x_1-5x_2+83x_3=155 \end{cases}$$

Atrisinājums. Nav grūti pārbaudīt, ka koeficientu matrica ir simetriska un pozitīvi definēta. Ja matrica ir simetriska, tad ir spēkā sakarības $a_{ij}=a_{ji}$ visiem $i=1,\ 2,\ 3$ un $j=1,\ 2,\ 3$. Koeficientu matrica ir

$$A=\begin{pmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{pmatrix}$$

Vizuālā pārbaude rāda, ka nosacījumi $a_{ij}=a_{ji}$ ir izpildīti.

Lai pārbaudītu, ka konkrētā kvadrātiskā matrica ir pozitīvi definēta, izmanto Silvestra kritēriju.

Silvestra kritērijs. Kvadrātiskā matrica A ir pozitīvi definēta tikai tad, kad visi matricas galvenie minori ir pozitīvi.

Atgādināsim, ka matricas A galvenais minors ar kārtu k ir tās $k\times k$ apakšmatricas determinants, kas atrodas matricas A kreisajā augšējā stūrī.

Mūsu piemērā pirmais galvenais minors ir $4>0$ (tas ir elements matricas A kreisajā augšējā stūrī), galvenais minors ar kārtu 2 ir

$$\begin{vmatrix} 4 & 2 \\ 2 & 17 \end{vmatrix} = 4\cdot 17 - 2\cdot 2 = 64 > 0,$$

un galvenais minors ar kārtu 3 ir matricas A determinants:

$$\begin{vmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{vmatrix} = 1600 > 0.$$

Tādējādi matrica A ir simetriska un pozitīvi definēta.

Tas nozīmē, ka

$$\begin{pmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}$$

Matricas L elementi ir:

$$l_{11} = \sqrt{a_{11}} = 2, \quad l_{21} = \frac{a_{21}}{l_{11}} = \frac{2}{2} = 1, \quad l_{31} = \frac{a_{31}}{l_{11}} = \frac{14}{2} = 7,$$

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{17-1} = 4, \quad l_{32} = \frac{1}{l_{22}}\left(a_{32} - l_{31}l_{21}\right) = \frac{1}{4}\left(-5 - 7\cdot 1\right) = -3,$$

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{83 - 7^2 - \left(-3\right)^2} = 5.$$

Konstruēsim sistēmu $L\boldsymbol{y} = \boldsymbol{b}$:

$$\begin{pmatrix} 2 & 0 & 0 \\ 1 & 4 & 0 \\ 7 & -3 & 5 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 14 \\ -101 \\ 155 \end{pmatrix}$$

Atrisinājums ir $\boldsymbol{y} = \begin{pmatrix} 7 \\ -27 \\ 5 \end{pmatrix}$.

Pēdējā solī atrisināsim sistēmu $U\boldsymbol{x} = L^T\boldsymbol{x} = \boldsymbol{y}$:

$$\begin{pmatrix} 2 & 1 & 7 \\ 0 & 4 & -3 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ -27 \\ 5 \end{pmatrix}$$

Atrisinājums ir $\boldsymbol{x} = \begin{pmatrix} 3 \\ -6 \\ 1 \end{pmatrix}$.

1.7. Aprēķini *MATLAB* vidē, izmantojot Hoļecka metodi

MATLAB komandu **chol** var izmantot, lai īstenotu Hoļecka metodi.

L=chol(A,'lower')	**L** – apakšējā trīsstūrveida matrica
isequal(A,A')	pārbaude, vai matrica A ir simetriska: $a_{ij}=a_{ji}$

1.14. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot Hoļecka metodi.

$$\begin{cases} x_1+2x_2+6x_3=7 \\ 2x_1+7x_2+3x_3=2 \\ 6x_1+3x_2+64x_3=13 \end{cases}$$

Atrisinājums.

```
%% 1.14. piemērs.Hoļecka metode
clc, clearvars, format compact
A = [1,2,6;2,7,3;6,3,64];  B = [7;2;13];
check = isequal(A,A');       % pārbaude: vai matrica ir simetriska
if check == 0                % check = 1(TRUE) vai check = 0(FALSE)
   disp('Koeficientu matrica nav simetriska')
   disp('Atbilde: Hoļecka metodi nedrīkst izmantot'), return
end
ni = fun_prob14(A);          % pārbaude: vai matrica ir pozitīvi definēta
if ni == 2
  disp('Koeficientu matrica nav pozitīvi definēta')
  disp('Atbilde: Hoļecka metodi nedrīkst izmantot'), return
end
disp('Koeficientu matrica ir simetriska un pozitīvi definēta')
```

Ārējo funkciju izmantosim, lai pārbaudītu, vai matrica A ir pozitīvi definēta.

```
% ārējā funkcija (1.14. piemērs). Hoļecka metode
% pārbaude: vai matrica ir pozitīvi definēta
function ni = fun_prob14(A_mat)
   ni = 1;
   [row,col] = size(A_mat);
   for i = 1:row
      if det(A_mat(1:i,1:i))>0
      else ni = 2; break
      end
   end
end
```

```
% 1.14. piemēra turpinājums
L = chol(A,'lower')
Amat = L*L'   % matrica A
Y = L\B       % vai Y=linsolve(L,B)
X = L'\Y      % vai X=linsolve(L',Y)
% Ctrl+Enter
```

```
Koeficientu matrica ir simetriska un pozitīvi definēta
L =
    1.0000         0         0
    2.0000    1.7321         0
    6.0000   -5.1962    1.0000
Amat =
    1.0000    2.0000    6.0000
    2.0000    7.0000    3.0000
    6.0000    3.0000   64.0000
```

```
Y =
    7.0000
   -6.9282
  -65.0000
X =
  795.0000
 -199.0000
  -65.0000
```

```
% 1.14. piemēra turpinājums
disp('Atbilde:')
fprintf('x1 = %.0f; x2 = %.0f; x3 = %.0f\n',X)
```

```
Atbilde:
 x1 = 795; x2 = -199; x3 = -65
```

1.8. *QR* dekompozīcija

QR dekompozīcija jeb atstarošanas metode arī balstās uz kvadrātiskās matricas sadalījumu reizinātājos. Pirmais reizinātājs ir ortogonālā matrica Q bet otrais reizinātājs ir augšējā trīsstūrveida matrica R (tas ir *QR* dekompozīcijas standarta apzīmējums, tāpēc nelietojam iepriekš izmantoto augšējās trīsstūrveida matricas apzīmējumu U).

Matricu Q sauc par ortogonālu, ja jebkuru divu atšķirīgu kolonnu skalārais reizinājums ir vienāds ar nulli, bet jebkuru divu vienādu kolonnu skalārais reizinājums ir vienāds ar 1.

Teorēma. Pieņemsim, ka A ir $m \times m$ matrica. Ja A nav singulāra, tad to var uzrakstīt kā reizinājumu vienā vienīgā veidā

$$A = QR, \tag{1.10}$$

kur

Q ir ortogonālā matrica;

R ir augšējā trīsstūrveida matrica.

Izmantojot ortogonālās matricas definīciju, iegūstam

$$QQ^T = Q^T Q = E, \tag{1.11}$$

kur E ir $m \times m$ vienības matrica.

No otras puses,

$$QQ^{-1} = Q^{-1} Q = E, \tag{1.12}$$

kur Q^{-1} ir matricas Q inversā matrica. Salīdzinot (1.11) un (1.12), iegūstam vienu no ortogonālās matricas īpašībām:

$$Q^{-1} = Q^T \tag{1.13}$$

Formulu (1.13) izmanto, lai atrisinātu lineāru vienādojumu sistēmu $A\boldsymbol{x} = \boldsymbol{b}$ ar *QR* dekompozīcijas metodi: $A\boldsymbol{x} = \boldsymbol{b} \to A = QR \to QR\boldsymbol{x} = \boldsymbol{b}$. Sareizinot iegūto vienādojumu ar Q^{-1} no kreisās puses, iegūstam $Q^{-1}QR\boldsymbol{x} = Q^{-1}\boldsymbol{b} \to R\boldsymbol{x} = Q^{-1}\boldsymbol{b} \to \left(\text{izmantojot } (1.13)\right) \to R\boldsymbol{x} = Q^T\boldsymbol{b}$.

1.9. Aprēķini *MATLAB* vidē, izmantojot *QR* dekompozīcijas metodi

MATLAB komandu **qr** izmanto, lai konstruētu konkrētās matricas A QR dekompozīciju.

[Q,R]=qr(A)	**R** – augšējā trīsstūrveida matrica **Q** – ortogonālā matrica

1.15. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot QR dekompozīciju.

$$\begin{cases} x_1 + 2x_2 + 6x_3 = 7 \\ 2x_1 + 7x_2 + 3x_3 = 2 \\ 6x_1 + 3x_2 + 64x_3 = 13 \end{cases}$$

Atrisinājums.

```
%% 1.15. piemērs.QR dekompozīcija
clc, clearvars, format compact
A = [1,2,6;2,7,3;6,3,64]; B = [7;2;13];
if det(A) == 0
  disp('Matrica A ir singulāra')
  disp(' Atbilde: atstarošanas metodi nedrīkst izmantot'), return
end
disp disp('Matrica A ir nesingulāra')
[Q,R] = qr(A)
Amat = ... ...
X = ... ...        % Ctrl+Enter
```

Skripts 1.15. piemērā nav uzrakstīts līdz galam. Iesakām lasītājam atrisināt šo uzdevumu, izmantojot formulas (sk. rindkopu pēc formulas (1.13)).

```
Matrica A ir nesingulāra
Q =
   -0.1562   -0.2014   -0.9670
   -0.3123   -0.9187    0.2417
   -0.9370    0.3398    0.0806
R =
   -6.4031   -5.3099  -61.8448
         0   -5.8142   17.7824
         0         0    0.0806
```

```
Amat =
    1.0000    2.0000    6.0000
    2.0000    7.0000    3.0000
    6.0000    3.0000   64.0000
X =
  795.0000
 -199.0000
  -65.0000
```

```
% 1.15. piemēra turpinājums
disp('Atbilde:')
fprintf(' x1 = %.0f; x2 = %.0f; x3 = %.0f\n',X)
```

```
Atbilde:
 x1 = 795; x2 = -199; x3 = -65
```

1.10. Faktorizācijas metode

Aplūkosim lineāru sistēmu (1.2). Pieņemsim, ka kvadrātiskā koeficientu matrica ir liela un satur daudz nuļļu, turklāt nulles elementi matricā atrodas noteiktā kārtībā. Konkrētāk aplūkosim gadījumu, kad nenulles elementi atrodas uz galvenās diagonāles un uz divām blakus diagonālēm (virs galvenās diagonāles un zem galvenās diagonāles). Šādu matricu sauc par trīsdiagonālu. Trīsdiagonālas matricas bieži rodas, risinot ar režģu metodi otrās kārtas parastu diferenciālvienādojumu robežproblēmas. Gausa izslēgšanas metodi trīsdiagonālām matricām sauc par **faktorizācijas metodi**. Šis metodes autors ir britu matemātiķis *L. H. Thomas*.

Lineāru vienādojumu sistēmu trīsdiagonālām matricām parasti pieraksta šādi:

$$a_i x_{i-1} - b_i x_i + c_i x_{i+1} = d_i, \quad i = 1,2,..,n, \quad a_1 = c_n = 0. \tag{1.14}$$

Gausa metodes pirmais solis (koeficientu matricas reducēšana uz trīsstūrveida matricu) vienādojumu sistēmai (1.14) atbilst mainīgo a_i izslēgšanai. Ja pirmais solis ir pabeigts, tad katrs vienādojums sistēmā satur ne vairāk kā divus nezināmos x_i un x_{i+1}.

Atrisinot katru vienādojumu attiecībā pret x_i, iegūstam

$$x_i = \xi_{i+1} x_{i+1} + \eta_{i+1}, i = n, n-1, \ldots, 1. \tag{1.15}$$

Vienādojumu (1.15) var pārrakstīt šādi (mainot indeksu i uz $i-1$):

$$x_{i-1} = \xi_i x_i + \eta_i \tag{1.16}$$

Ievietosim formulu (1.16) vienādojumā (1.14):

$$a_i \left(\xi_i x_i + \eta_i \right) - b_i x_i + c_i x_{i+1} = d_i \tag{1.17}$$

Vienādojumu (1.17) pārveidosim šādi

$$x_i = \frac{c_i}{b_i - a_i \xi_i} x_{i+1} + \frac{a_i \eta_i - d_i}{b_i - a_i \xi_i} \tag{1.18}$$

Salīdzinot (1.15) un (1.18), iegūstam divas rekurentas formulas koeficientu ξ_{i+1} un η_{i+1} noteikšanai:

$$\xi_{i+1} = \frac{c_i}{b_i - a_i \xi_i}, \quad \eta_{i+1} = \frac{a_i \eta_i - d_i}{b_i - a_i \xi_i}, \quad i = 1,2,\ldots,n \tag{1.19}$$

Lai sāktu aprēķinus ar formulu (1.19), ir jāpiešķir sākuma vērtības koeficientiem ξ_1 un η_1. Ievietojot $i = 1$ formulā (1.19), redzam, ka ξ_1 un η_1 parādās kā reizinātāji pie koeficienta a_1, kurš ir vienāds ar nulli (sk. (1.14)). Tādējādi vērtības ξ_1 un η_1 var būt patvaļīgas, piemēram, var pieņemt, ka $\xi_1 = \eta_1 = 0$. Analoģiski, lai sāktu aprēķinus ar formulu (1.15), ir jāzina parametra ξ_{n+1} sākuma vērtība. Ievietojot $i = n$ formulā (1.19), redzam, ka ξ_{n+1} ir proporcionāls koeficientam c_n, kurš ir vienāds ar nulli (sk. (1.14)). Tādējādi var pieņemt ka $\xi_{n+1} = 0$.

Turpmāk sniegts īss faktorizācijas metodes apraksts.

1. solis. Izmantojot sākuma vērtības $\xi_1 = \eta_1 = 0$, aprēķināt ξ_i un η_i ar formulu (1.19), kur $i = 1, 2, \ldots, n$.

2. solis. Izmantojot sākuma vērtību $\xi_{n+1} = 0$, aprēķināt x_i, $i = n, n-1, \ldots, 1$ ar formulu (1.15).

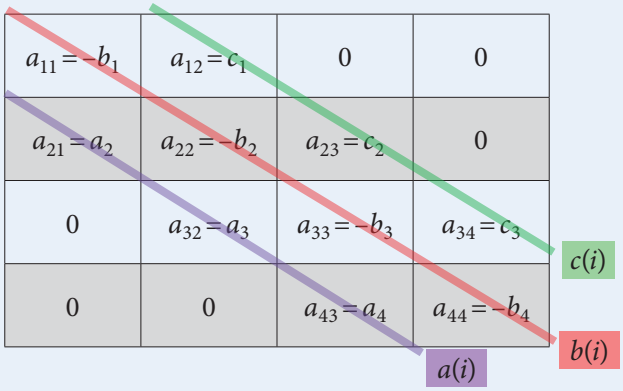
Var pierādīt, ka faktorizācijas metode* ir stabila attiecībā pret noapaļošanas kļūdām, ja ir spēkā nosacījumi

$$|b_i| \geq |a_i| + |c_i|$$

visiem $i = 1, 2, \ldots, n$ un vismaz vienam indeksam i ir pareiza nevienādība

$$|b_i| > |a_i| + |c_i|$$

Koeficientu matricas elementi shematiski parādīti tabulā:



*Thomas, L. H. Elliptic problems in linear differential equations over a network. Watson Sci. Comput. Lab Report, Columbia University, New York, 1949.

1.11. Aprēķini *MATLAB* vidē, izmantojot faktorizācijas metodi

1.16. piemērs. Atrisināt sistēmu, izmantojot faktorizācijas metodi.

$$\begin{cases} -5x_1 + x_2 = 1 \\ 2x_1 - 6x_2 + 3x_3 = 7 \\ x_2 - 9x_3 + 4x_4 = -2 \\ 4x_3 - 6x_4 = 9 \end{cases}$$

Atrisinājums.

Šajā piemērā vienādojumu sistēma satur četrus nezināmos. Būtu vēlams modificēt skriptu tā, lai to izmantotu vienādojumu sistēmām ar trīsdiagonālo matricu gadījumā, kad vienādojumu skaits sistēmā ir n. Iesakām lasītājam modificēt skriptu (ņemot vērā informāciju taisnstūrī). Parametri, kas jāmaina, ir iekrāsoti sarkanā krāsā.

```
%% 1.16. piemērs. Faktorizācijas metode
clc, clearvars, format compact
a = [0 2 1 4]; b = [-5 -6 -9 -6].*(-1);
c = [1 3 4 0]; d = [1;7;-2;9];
ksi =zeros(1,5); eta =zeros(1,5);
for i = 1:4
    ksi(i+1) =c(i)/(b(i)-a(i)*ksi(i));

    eta(i+1) =(a(i)*eta(i)-d(i))/(b(i)-a(i)*ksi(i));
end
X(4) =eta(5); % aprēķināt x
for i = 3:-1:1
    X(i) =ksi(i+1)*X(i+1)+eta(i+1);
end
X   % Ctrl+Enter
```

```
n = length(a); n_elem =n+1;
ksi =zeros(1,n_elem);...
```

$\xi_1 = 0, \xi_{n+1} = 0, \eta_1 = 0$

$\xi_{i+1} = \dfrac{c_i}{b_i - a_i\xi_i}$

$\eta_{i+1} = \dfrac{a_i\eta_i - d_i}{b_i - a_i\xi_i}$

$x_n = \eta_{n+1}$

$x_i = \xi_{i+1}x_{i+1} + \eta_{i+1}$

```
X =
   -0.5626   -1.8131   -0.9179   -2.1119
```

Ārējo funkciju izmantosim, lai pārbaudītu metodes stabilitāti.

```
% 1.16. piemēra turpinājums
disp('Atbilde:'), disp('Atrisinājums ='), disp(X)
fun_prob16(a,b,c)    % pārbaude: Faktorizācijas metode ir stabila
```

```
% ārēja funkcija (1.16. piemērs). Faktorizācijas metode
function fun_prob16(a_el,b_el,c_el)
   n = length(a_el);
   for i = 1:n
      if abs(b_el(i)) < ( abs(a_el(i))+abs(c_el(i)) )
         disp('Faktorizācijas metode nav stabila'), return
      end
    end
    disp('Faktorizācijas metode ir stabila')
end
```

```
Atbilde:
 Atrisinājums =
   -0.5626   -1.8131   -0.9179   -2.1119
 Faktorizācijas metode ir stabila
```

1.17. piemērs. Atrisināt sistēmu, izmantojot faktorizācijas metodi.

$$\begin{cases} -3x_1 + x_2 = 1 \\ x_1 - 3x_2 + x_3 = 1 \\ x_2 - 3x_3 + x_4 = 1 \\ x_3 - 3x_4 + x_5 = 1 \\ x_4 - 3x_5 = 1 \end{cases}$$

Atrisinājums.

```
%% 1.17. piemērs. Faktorizācijas metode
clc, clearvars, format compact
a =ones(1,5); % rindas matrica ar vieniniekiem
b(1,1:5) =-3.*(-1); c =ones(1,5);
d =ones(5,1); % kolonnas matrica ar vieniniekiem
a(1) =0; c(5) =0;
ksi =zeros(1,6); eta =zeros(1,6);
for i = 1:5
    ksi(i+1) =c(i)/(b(i)-a(i)*ksi(i));
    eta(i+1) =(a(i)*eta(i)-d(i))/(b(i)-a(i)*ksi(i));
end
X(5) =eta(6); % aprēķināt x
for i = 4:-1:1
    X(i) =ksi(i+1)*X(i+1)+eta(i+1);
end
X % Ctrl+Enter
```

n=5; a=ones(1,n);...

$x_n = \eta_{n+1}$

$x_i = \xi_{i+1} x_{i+1} + \eta_{i+1}$

Šajā piemērā arī ir vēlams pārrakstīt skriptu tā, lai būtu ērtāk risināt vienādojumu sistēmu ar patvaļīgu nezināmo skaitu n.

```
X =
   -0.6111   -0.8333   -0.8889   -0.8333   -0.6111
```

```
% 1.17. piemēra turpinājums
disp('Atbilde:'), disp('Atrisinājums ='), disp(X)
fun_prob16(a,b,c)   % pārbaude: Faktorizācijas metode ir stabila
```

```
Atbilde:
 Atrisinājums =
   -0.6111   -0.8333   -0.8889   -0.8333   -0.6111
 Faktorizācijas metode ir stabila
```

UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

1.1. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi.

$$\begin{cases} 3x - y + 4z = -3 \\ 2x + 3y + z = 5 \\ x - 5y - 3z = 3 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     3
sol =
     1     0     0     2
     0     1     0     1
     0     0     1    -2
```

```
Atbilde:
 LVS ir saderīga un noteikta (viens vienīgs atrisinājums)
 solution =
    X_main    X_value
    ______    _______
    x1          2
    x2          1
    x3         -2
```

1.2. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu.

$$\begin{cases} 3x + 4z - y = -1 \\ 2x + 3y + z = 1 \\ 7x + 6z + 5y - 1 = 0 \end{cases}$$

```
A_rank =
     2
Aaug_rank =
     2
```

```
sol =
[ 1, 0, 13/11, -2/11]
[ 0, 1, -5/11,  5/11]
[ 0, 0,     0,     0]
X_gen =
 - (13*z)/11 - 2/11
    (5*z)/11 + 5/11
                   0
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x = - (13*z)/11 - 2/11, y = (5*z)/11 + 5/11; z - jebkurš reāls skaitlis
```

1.3. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu un atrast x_1, ja $x_4 = -2$.

$$\begin{cases} 3x_1 - 3x_2 + 3x_3 + 6x_4 = 6 \\ 3x_1 + 6x_2 - 3x_3 + 7x_4 = 1 \\ 3x_1 - 9x_2 + 3x_3 + x_4 = 7 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     3
sol =
[ 1, 0, 0,   7/4,  5/4]
[ 0, 1, 0,   5/6, -1/6]
[ 0, 0, 1, 13/12, 7/12]
```

```
X_gen =
    5/4 - (7*x4)/4
  - (5*x4)/6 - 1/6
 7/12 - (13*x4)/12
```

```
X_part =
 19/4
  3/2
 11/4

x1_ver =
19/4
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x1 = 5/4 - (7*x4)/4, x2 = - (5*x4)/6 - 1/6; x3 = 7/12 - (13*x4)/12;
 kur x4 - jebkurš reāls skaitlis

 x1 = 19/4
```

1.4. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu un atrast x_2, ja $x_3 = -1$, $x_4 = 3$.

$$\begin{cases} x_1 + 2x_2 + x_3 - 3x_4 = 5 \\ 2x_1 + x_2 - 2x_3 + 9x_4 = -2 \\ 3x_1 + 3x_2 + 6x_4 - x_3 = 3 \\ 4x_1 + 5x_2 + 3x_4 = 8 \end{cases}$$

```
A_rank =
     2
Aaug_rank =
     2
```

```
sol =
[ 1, 0, -5/3,  7, -3]
[ 0, 1,  4/3, -5,  4]
[ 0, 0,    0,  0,  0]
[ 0, 0,    0,  0,  0]

X_gen =
 (5*x3)/3 - 7*x4 - 3
 5*x4 - (4*x3)/3 + 4
                   0
                   0
```

```
X_part =
 -77/3
  61/3
     0
     0

x2_ver =
61/3
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x1 = (5*x3)/3 - 7*x4 - 3, x2 = 5*x4 - (4*x3)/3 + 4;
 kur x3, x4 - jebkuri reāli skaitļi

 x2 = 61/3
```

1.5. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot

a) Gausa metodi;
b) LU dekompozīciju;
c) Hoļecka metodi;
d) *QR* dekompozīciju;
e) atrast koeficientu matricas determinantu un inverso matricu.

$$\begin{cases} 4x_1 + 4x_2 + 5x_3 + 5x_4 = 0 \\ 2x_1 + 3x_3 - x_4 = 10 \\ x_1 + x_2 - 5x_3 = -10 \\ 3x_2 + 2x_3 = 1 \end{cases}$$

a) Gausa metode.

```
A_rank =
     4
Aaug_rank =
     4
sol =
     1     0     0     0     1
     0     1     0     0    -1
     0     0     1     0     2
     0     0     0     1    -2
```

```
Atbilde:
 LVS ir saderīga un noteikta (viens vienīgs atrisinājums)
 x1 = 1, x2 = -1, x3 = 2, x4 = -2
```

b) LU dekompozīcija.

```
Y =
         0
    1.0000
  -10.0000
    7.7333
X =
    1.0000
   -1.0000
    2.0000
   -2.0000
```

```
Visi galvenie minori nav vienādi ar nulli
L =
    1.0000         0         0         0
         0    1.0000         0         0
    0.2500         0    1.0000         0
    0.5000   -0.6667   -0.2933    1.0000
U =
    4.0000    4.0000    5.0000    5.0000
         0    3.0000    2.0000         0
         0         0   -6.2500   -1.2500
         0         0         0   -3.8667
P =
     1     0     0     0
     0     0     0     1
     0     0     1     0
     0     1     0     0
```

```
Atbilde:
 x1 = 1, x2 = -1, x3 = 2, x4 = -2
```

c) Hoļecka metode.

```
Koeficientu matrica nav simetriska.
 Atbilde: Hoļecka metodi nedrīkst izmantot.
```

d) *QR* dekompozīcija.

```
Matrica A ir nesingulāra
Q =
   -0.8729    0.2178    0.1177   -0.4205
   -0.4364   -0.4628    0.2397    0.7334
   -0.2182    0.0544   -0.9503    0.2151
         0    0.8576    0.1598    0.4889
R =
   -4.5826   -3.7097   -4.5826   -3.9279
         0    3.4983    1.1434    1.5518
         0         0    6.3791    0.3489
         0         0         0   -2.8358
```

```
X =
    1.0000
   -1.0000
    2.0000
   -2.0000
```

```
Atbilde:
 x1 = 1, x2 = -1, x3 = 2, x4 = -2
```

e) Atrast koeficientu matricas determinantu un inverso matricu.

```
A_det =
  -290
A_inv =
    0.0586    0.2931    0.1793   -0.1379
   -0.0069   -0.0345    0.0966    0.3103
    0.0103    0.0517   -0.1448    0.0345
    0.1483   -0.2586   -0.0759   -0.1724
```

```
Atbilde:
 matricas determinants = -290
```

1.6. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot LU dekompozīciju.

$$\begin{cases} 3x_2 + x_3 = 3 \\ x_1 + 4x_2 + 5x_3 = 1 \\ 3x_1 + 2x_2 - 6x_3 = 2 \end{cases}$$

```
Vismaz viens no galvenajiem minoriem ir vienāds ar nulli.
 Atbilde: LU dekompozīcijas metodi nedrīkst izmantot.
```

1.7. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Atrast koeficientu matricas determinantu un inverso matricu.

$$\begin{cases} 2x_1 - x_2 = 0 \\ x_1 - x_3 + 2x_2 + 2 = 0 \\ x_3 + x_2 + 5 = 0 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     3
sol =
     1     0     0    -1
     0     1     0    -2
     0     0     1    -3
```

```
A_det =
     7
A_inv =
    0.4286    0.1429    0.1429
   -0.1429    0.2857    0.2857
    0.1429   -0.2857    0.7143
```

```
Atbilde:
 LVS ir saderīga un noteikta (viens vienīgs atrisinājums)
 x1 = -1, x2 = -2, x3 = -3
 matricas determinants = 7
```

1.8. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Atrast sistēmas vispārīgo atrisinājumu.

$$\begin{cases} x_1 + x_2 + 3x_3 - 2x_4 - 3x_5 = 0 \\ -x_2 + 4x_3 - 6x_5 = -3 \\ -14x_3 + 5x_4 + 19x_5 = 10 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     3
```

```
sol =
[ 1, 0, 0,   1/2,    1/2,    2]
[ 0, 1, 0, -10/7,    4/7,  1/7]
[ 0, 0, 1, -5/14, -19/14, -5/7]
```

```
X_gen =
                  2 - x5/2 - x4/2
    (10*x4)/7 - (4*x5)/7 + 1/7
   (5*x4)/14 + (19*x5)/14 - 5/7
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x1 = 2 - x5/2 - x4/2, x2 = (10*x4)/7 - (4*x5)/7 + 1/7,
 x3 = (5*x4)/14 + (19*x5)/14 - 5/7; kur x4, x5 - jebkuri reāli skaitļi
```

1.9. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi.

$$\begin{cases} x_2 + 2x_3 - x_4 = 2 \\ 2x_1 - 3x_2 + 2x_3 = -2 \\ 2x_1 - 4x_2 + x_4 = 3 \\ 5x_2 - 2x_1 - 3x_3 + 3x_4 = 1 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     4
```

```
sol =
[ 1, 0, 0, 5/2, 0]
[ 0, 1, 0,   1, 0]
[ 0, 0, 1,  -1, 0]
[ 0, 0, 0,   0, 1]
```

```
Atbilde:
 LVS ir nesaderīga (nav atrisinājuma)
```

1.10. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Gausa metodi. Pieņemt, ka x_4, x_5 ir brīvi parametri. Atrast x_3, ja $x_4 = -2$, $x_5 = -1$.

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 3 \\ x_1 + 2x_2 + 2x_3 + 4x_4 + 2x_5 = 5 \\ x_1 + 2x_2 + 3x_3 - x_4 + 3x_5 = 6 \end{cases}$$

```
A_rank =
     3
Aaug_rank =
     3
sol =
[ 1, 0, 0, -2, 0, 1]
[ 0, 1, 0,  8, 0, 1]
[ 0, 0, 1, -5, 1, 1]
```

```
X_gen =
      2*x4 + 1
      1 - 8*x4
 5*x4 - x5 + 1
X_part =
 -3
 17
 -8
```

```
x3_ver =
-8
```

```
Atbilde:
 LVS ir saderīga un nenoteikta (bezgalīgi daudz atrisinājumu)
 x1 = 2*x4 + 1, x2 = 1 - 8*x4; x3 = 5*x4 - x5 + 1;
 kur x4, x5 - jebkuri reāli skaitļi

 x3 = -8
```

1.11. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu un atrast x_1, ja $x_3 = 4.5$; $x_4 = -1.2$.

$$\begin{cases} x_1 - 4x_2 + 3x_3 - 2x_4 = 3 \\ 2x_3 - 3x_2 - x_1 + 5x_4 = 1 \\ 7x_2 - 5x_3 - 3x_4 = -4 \\ 2x_1 - x_2 + x_3 - 7x_4 = 2 \end{cases}$$

```
A_rank =
     2
Aaug_rank =
     2
sol =
[ 1, 0,  1/7, -26/7,  5/7]
[ 0, 1, -5/7,  -3/7, -4/7]
[ 0, 0,    0,     0,    0]
[ 0, 0,    0,     0,    0]
```

```
X_gen =
    (26*x4)/7 - x3/7 + 5/7
 (5*x3)/7 + (3*x4)/7 - 4/7
                        0
                        0
X_part =
 -307/70
  149/70
       0
       0
```

```
x1_ver =
-307/70
```

```
Atbilde:
  LVS ir saderīga un nenoteikta(bezgalīgi daudz atrisinājumu)
  x1 = (26*x4)/7 - x3/7 + 5/7, x2 = (5*x3)/7 + (3*x4)/7 - 4/7;
  kur x3, x4 - jebkuri reāli skaitļi

 x1 = -307/70
```

1.12. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot Gausa metodi. Noteikt vispārīgo atrisinājumu un atrast x_2, ja $x_3 = -2.6$, $x_4 = 3.2$.

$$\begin{cases} 2x_2 + x_1 - 2x_3 + 5x_4 = 2 \\ 3x_1 - x_2 - 5x_3 + 4x_4 = 4 \\ 4x_1 + x_2 - 7x_3 + 9x_4 = 6 \\ 2x_1 - 3x_2 - 3x_3 - x_4 = 2 \end{cases}$$

```
A_rank =
     2
Aaug_rank =
     2
sol =
[ 1, 0, -12/7, 13/7, 10/7]
[ 0, 1,  -1/7, 11/7,  2/7]
[ 0, 0,     0,    0,    0]
[ 0, 0,     0,    0,    0]
```

```
X_gen =
  12*x3/7 - 13*x4/7 + 10/7
       x3/7 - 11*x4/7 + 2/7
                        0
                        0
X_part =
 -314/35
 -179/35
       0
       0
```

```
x2_ver =
-179/35
```

```
Atbilde:
  LVS ir saderīga un nenoteikta(bezgalīgi daudz atrisinājumu)
  x1 = (12*x3)/7 - (13*x4)/7 + 10/7, x2 = x3/7 - (11*x4)/7 + 2/7;
  kur x3, x4 - jebkuri reāli skaitļi

 x2 = -179/35
```

1.13. uzdevums. Atrisināt sistēmu, izmantojot faktorizācijas metodi, ja a) $n = 20$; b) $n = 100$.

$$\begin{cases} -3x_1 + x_2 = 1 \\ x_1 - 3x_2 + x_3 = 1 \\ x_2 - 3x_3 + x_4 = 1 \\ \dots \\ x_{n-2} - 3x_{n-1} + x_n = 1 \\ x_{n-1} - 3x_n = 1 \end{cases}$$

a) $n = 20$

```
X_sol =
   -0.6180
   -0.8541
   -0.9443
   -0.9787
   -0.9919
   -0.9969
   -0.9988
   -0.9995
   -0.9998
   -0.9999
   -0.9999
   -0.9998
   -0.9995
   -0.9988
   -0.9969
   -0.9919
   -0.9787
   -0.9443
   -0.8541
   -0.6180
```

```
Atbilde: A(20X20)
 Atrisinājums = [-0.6180,-0.8541,... -0.8541,-0.6180]
 Faktorizācijas metode ir stabila.
```

b) $n = 100$:

```
X_sol =
   -0.6180
   -0.8541
   -0.9443
   -0.9787
   -0.9919
   -0.9969
   -0.9988
   -0.9995
   -0.9998
   -0.9999
   -1.0000
   -1.0000
    ...   ...
```

```
    ...   ...
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -0.9999
   -0.9998
   -0.9995
   -0.9988
   -0.9969
   -0.9919
   -0.9787
   -0.9443
   -0.8541
   -0.6180
```

```
Atbilde: A(100X100)
 Atrisinājums = [ -0.6180, -0.8541,... -0.8541, -0.6180]
 Faktorizācijas metode ir stabila.
```

1.14. uzdevums. Atrisināt sistēmu, izmantojot faktorizācijas metodi.

$$\begin{cases} -4x_1 + 2x_2 = 3 \\ 3x_1 - 4x_2 + 2x_3 = 3 \\ 3x_2 - 4x_3 + 2x_4 = 3 \\ 3x_3 - 4x_4 + 2x_5 = 3 \\ \dots \\ 3x_{19} - 4x_{20} = 3 \end{cases}$$

```
X_sol =
    5.5725
   12.6449
   18.4311
   19.3949
   12.6431
   -2.3061
  -22.0769
  -39.1947
  -43.7740
  -27.2559
   12.6492
   67.6822
  117.8906
  135.7579
   96.1799
   -9.7770
 -162.3239
 -308.4822
 -371.9787
 -279.7340
```

```
Atbilde: A(20X20)
 Atrisinājums = [5.5725,12.6449,... -371.9787,-279.7340]
 Faktorizācijas metode nav stabila.
```

2. nodaļa

LINEĀRU VIENĀDOJUMU SISTĒMAS. ITERĀCIJU METODES

2.1. Vektoru un matricu normas

Normas jēdzienu izmanto, lai raksturotu attālumu starp vektoriem vai matricām daudzdimensiju telpā.

Aplūkosim m dimensiju vektoru $\boldsymbol{x}=(x_1\ x_2\ldots x_m)^T$. Vektora normu apzīmēsim šādi: $\|\boldsymbol{x}\|$.

Normu definē kā nenegatīvu skaitli, kam ir spēkā īpašības:

1) $\|\boldsymbol{x}\|\geq 0$ visiem x;
2) $\|\boldsymbol{x}\|=0$ tad un tikai tad, kad $\boldsymbol{x}=\boldsymbol{0}$
3) $\|\alpha\boldsymbol{x}\|=|\alpha|\|\boldsymbol{x}\|$ jebkuram reālam skaitlim α
4) $\|\boldsymbol{x}+\boldsymbol{y}\|\leq\|\boldsymbol{x}\|+\|\boldsymbol{y}\|$

Vispārīgā gadījumā vektora L_p normu definē pēc formulas

$$\|\boldsymbol{x}\|_p=\left(|x_1|^p+|x_2|^p+\ldots+|x_m|^p\right)^{1/p}=\left(\sum_{i=1}^{m}|x_i|^p\right)^{1/p},$$

kur p ir jebkurš naturāls skaitlis.

Bieži izmanto vērtības $p=1$ un $p=2$.

Tādējādi

$$\|\boldsymbol{x}\|_1=|x_1|+|x_2|+\ldots+|x_m|,$$
$$\|\boldsymbol{x}\|_2=\sqrt{x_1^2+x_2^2+\ldots+x_m^2}.$$

Definē arī normu L_∞:

$$\|\boldsymbol{x}\|_\infty=\max_{1\leq j\leq m}|x_j|.$$

Pieņemsim, ka A ir $m\times m$ matrica un $\boldsymbol{x}$ ir jebkurš vektors ar m komponentēm. Aplūkosim patvaļīgu vektora $\boldsymbol{x}$ normu $\|\boldsymbol{x}\|$.

Var pierādīt, ka eksistē skaitlis c (kas ir atkarīgs no A), ka visiem $\boldsymbol{x}$ ir spēkā nevienādība $\|A\boldsymbol{x}\|\leq c\|\boldsymbol{x}\|$.

Ja $\boldsymbol{x}\neq\boldsymbol{0}$, tad šo nevienādību var pārrakstīt šādi

$$\frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leq c.$$

Aprēķinot maksimumu kreisajā pusē visiem $\boldsymbol{x}\neq\boldsymbol{0}$, iegūstam

$$\max_{\boldsymbol{x}\neq 0}\frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}=c=\|A\|.$$

Tādējādi matricas A norma ir

$$\|A\|=\max_{\boldsymbol{x}\neq 0}\frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$

No šīs formulas izriet, ka matricas normas definīcija ir atkarīga no izvēlētās vektora normas.

Var pierādīt, ka kvadrātiskās $m\times m$ matricas A normas L_1, L_2 un L_∞ definē pēc formulām:

$$\|A\|_1=\max_{1\leq j\leq m}\sum_{i=1}^{m}|a_{ij}|,$$
$$\|A\|_2=\sqrt{\lambda_{\max}\left(A^TA\right)},$$
$$\|A\|_\infty=\max_{1\leq i\leq m}\sum_{j=1}^{m}|a_{ij}|,$$

kur $\lambda_{\max}$ ir maksimāla matricas A^TA īpašvērtība (īpašvērtības un īpašvektori definēti nākamajā sadaļā).

2.2. Aprēķini *MATLAB* vidē, lai aprēķinātu vektoru un matricu normas

norm(v,p)	vektora **v** norma ($p = 1, 2, ...\text{inf}$) **norm(v,2)=norm(v)** – pēc noklusējuma
norm(A,p)	matricas A norma ($p = 1, 2, ...\text{inf}$)
num2str(skait)	pārveidot skaitli par simbolu

2.1. piemērs. Aprēķināt vektora $\boldsymbol{u} = \begin{pmatrix} 1 \\ 4 \\ 6 \\ 8 \end{pmatrix}$ normas $\|\boldsymbol{u}\|_1$, $\|\boldsymbol{u}\|_2$ un $\|\boldsymbol{u}\|_\infty$.

Atrast matricas A normu $\|A\|_2$.

$$A = \begin{pmatrix} 3 & 7 & -1 & 2 \\ 8 & 4 & 3 & 5 \\ -6 & 9 & 0 & 4 \\ 5 & 8 & 11 & 3 \end{pmatrix}$$

Atrisinājums.

```
%% 2.1. piemērs. Vektoru un matricu normas
clc, clearvars, format compact
u = [1;4;6;8];
A = [ 3,7,-1,2;8,4,3,5;-6,9,0,4;5,8,11,3];
norm_v1 = norm(u,1), norm_v2 = norm(u,2)
norm_vInf = norm(u,inf), norm_m2 = norm(A,2)% Ctrl+Enter
```

```
norm_v1 =
    19
norm_v2 =
   10.8167
norm_vInf =
     8
norm_m2 =
   18.6011
```

```
% 2.1. piemēra turpinājums
disp('Atbilde:')
fprintf('vektora norma (p=1) = %.1f\n' ,norm_v1)
fprintf('vektora norma (p=2) = %.4f\n' , norm_v2)
fprintf('vektora norma (inf) = %.1f\n' , norm_vInf)
disp(['matricas norma (p=2) = ' , num2str(norm_m2) ])
```

```
Atbilde:
  vektora norma (p=1) = 19.0
  vektora norma (p=2) = 10.8167
  vektora norma (inf) = 8.0
  matricas norma (p=2) = 18.6011
```

2.3. Matricas īpašvērtības un īpašvektori

Aplūkosim $n \times n$ matricu A:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

Skaitli λ sauc par matricas A īpašvērtību, ja eksistē vektors $\boldsymbol{x}$ ($\boldsymbol{x} \neq \boldsymbol{0}$), kas apmierina nosacījumu

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

Vektoru $\boldsymbol{x}$ sauc par matricas A īpašvektoru, kas atbilst īpašvērtībai λ.

Vienādojumu $A\boldsymbol{x} = \lambda \boldsymbol{x}$ var pārrakstīt šādi

$$\left(A - \lambda E\right)\boldsymbol{x} = 0, \tag{2.1}$$

kur E ir $n \times n$ vienības matrica.

Tā kā sistēma (2.1) ir homogēna, netriviāls atrisinājums eksistē tikai tad, kad

$$\det\left(A - \lambda E\right) = 0. \tag{2.2}$$

Vienādojumu (2.2) sauc par raksturīgo vienādojumu. Polinomu $p\left(\lambda\right) = \det\left(A - \lambda E\right)$ sauc par matricas A raksturīgo polinomu.

2.4. Aprēķini *MATLAB* vidē, lai aprēķinātu matricas īpašvērtības un īpašvektorus

Matricas A īpašvērtības ir raksturīgā vienādojuma saknes. Matricas īpašvērtības un īpašvektorus aprēķina, izmantojot *MATLAB* funkciju **`eig(A)`**.

`[V,D] = eig(A)`	**`V`** – matrica, kuras kolonnas veido īpašvektori **`D`** – diagonālā matrica (īpašvērtības atrodas uz galvenās diagonāles)
`lambda = eig(A)`	matricas **`A`** īpašvērtības
`eigs(A,1)`	vislielākā matricas **`A`** īpašvērtība

2.2. piemērs. Atrast matricas A īpašvērtības un īpašvektorus.

$$A = \begin{pmatrix} 1 & -2 & 3 & -4 \\ 3 & 7 & -8 & 2 \\ 13 & 5 & -9 & 0 \\ 6 & 4 & 10 & -3 \end{pmatrix}$$

Atrisinājums.

```
%% 2.2. piemērs. Matricas īpašvērtības un īpašvektori
clc, clearvars, format compact
A = [1,-2,3,-4;3,7,-8,2;13,5,-9,0;6,4,10,-3];
[V,D] = eig(A)
lambda = eig(A) % vai lambda=diag(D)
```

```
V =
 0.2820 + 0.0000i  0.0993 + 0.3293i   0.0993 - 0.3293i   0.4189 + 0.0000i
-0.3673 + 0.0000i -0.0864 + 0.1388i  -0.0864 - 0.1388i  -0.9044 + 0.0000i
-0.6058 + 0.0000i  0.2940 + 0.3026i   0.2940 - 0.3026i   0.0605 + 0.0000i
 0.6470 + 0.0000i  0.8228 + 0.0000i   0.8228 + 0.0000i  -0.0538 + 0.0000i
```

```
D =
 -12.0189 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i    0.0000 +
0.0000i
   0.0000 + 0.0000i   0.8770 + 6.7543i   0.0000 + 0.0000i    0.0000 +
0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.8770 - 6.7543i    0.0000 +
0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i    6.2649 +
0.0000i
```

```
lambda =
 -12.0189 + 0.0000i
   0.8770 + 6.7543i
   0.8770 - 6.7543i
   6.2649 + 0.0000i
```

2.5. Iterāciju metodes (atrisinājuma vispārīgā shēma)

Aplūkosim m lineāru vienādojumu sistēmu ar m nezināmajiem $x_1, x_2, \ldots, x_m$, kas ir uzrakstīta matricu veidā

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{2.3}$$

kur
A ir $m \times m$ nesingulārā matrica;
$\boldsymbol{b}$ ir $m \times 1$ matrica.

Formulēsim iterāciju metodes vispārīgo shēmu lineāru vienādojumu sistēmas (2.3) atrisināšanai.

1. solis. Izvēlēties sākuma tuvinājumu $\boldsymbol{x}^{(0)}$ sistēmas (2.3) atrisinājumam. Kļūdu (jeb nesaisti) aprēķina pēc formulas

$$\boldsymbol{r}^{(0)} = A\boldsymbol{x}^{(0)} - \boldsymbol{b}.$$

2. solis. Definēt vektoru virkni

$$\boldsymbol{x}^{(n+1)} = f\left(\boldsymbol{x}^{(n)}\right), n = 0,1,\ldots \tag{2.4}$$

kur funkcija f ir atkarīga no konkrētās metodes izvēles.

3. solis. Pārbaudīt, vai virkne (2.4) konverģē. Konkrētas iterāciju metodes konverģences nosacījumi parasti ir zināmi. Tos var pārbaudīt, analizējot matricas A elementus.

4. solis. Atrast atrisinājumu ar noteiktu precizitāti. 2. soli atkārto vairākas reizes, kamēr nav sasniegta aprēķinu precizitāte. Parasti pārbauda starpību starp $\boldsymbol{x}^{(n+1)}$ un $\boldsymbol{x}^{(n)}$ pēc normas:
$\left\|\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}\right\| < \varepsilon$, kur ε ir aprēķinu precizitāte.

2.6. Jakobi metode

Aplūkosim lineāru vienādojumu sistēmu:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m = b_2$$
$$\ldots\ldots\ldots\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mm}x_m = b_m$$

Pieņemsim, ka visi matricas A elementi, kas atrodas uz galvenās diagonāles, atšķiras no nulles:

$$a_{ii} \neq 0, \quad i = 1,2,\ldots,m.$$

Pārveidosim vienādojumu sistēmu

$$x_1 = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1m}x_m\right)$$
$$x_2 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2m}x_m\right) \qquad (2.5)$$
$$\ldots\ldots\ldots\ldots$$
$$x_m = \frac{1}{a_{mm}}\left(b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{m,m-1}x_{m-1}\right)$$

Izmantojot summēšanas simbolu, vienādojumu sistēmu (2.5) pārrakstīsim šādi:

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{m} a_{ij}x_j\right), \quad i = 1,2,\ldots,m.$$

Apzīmēsim ar $x_1^{(0)}, x_2^{(0)}, \ldots, x_m^{(0)}$ sākuma tuvinājumu sistēmas (2.3) atrisinājumam.

Bieži (ja nekas nav zināms par atrisinājuma struktūru) par sākuma tuvinājumu pieņem nulles vektoru: $x_i^{(0)} = 0, \quad i = 1,2,\ldots,m.$

Izmantojot formulas (2.5), aprēķināsim pirmo tuvinājumu atrisinājumam:

$$x_i^{(1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{m} a_{ij}x_j^{(0)}\right), \quad i = 1,2,\ldots,m.$$

Analoģiski atrodam otro tuvinājumu:

$$x_i^{(2)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{m} a_{ij}x_j^{(1)}\right), \quad i = 1,2,\ldots,m.$$

Iterācijas metodi definē pēc formulas

$$x_i^{(n+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{m} a_{ij}x_j^{(n)}\right), \quad i = 1,2,\ldots,m,\ n = 0,1,\ldots$$

Lai pabeigtu aprēķinus, izmanto šādu kritēriju:

$$\left\| \boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)} \right\| < \varepsilon$$

kur ε ir pietiekami mazs skaitlis.

Pietiekamais konverģences nosacījums (Jakobi metode).

Pieņemsim, ka matricas A elementi apmierina nosacījumu

$$|a_{ii}| > \sum_{j=1, j\neq i}^{m} |a_{ij}| \text{ visiem } i = 1, 2, \ldots, m. \qquad (2.6)$$

Tad Jakobi metode konverģē pie jebkura sākuma tuvinājuma. Šajā gadījumā saka, ka matrica A ir ar dominējošo galveno diagonāli.

Nosacījuma (2.6) jēga ir šāda: kvadrātiskajai matricai A ir dominējošā galvenā diagonāle, ja visiem $i = 1, 2, \ldots, m$ diagonālā elementa a_{ii} modulis ir lielāks nekā citu koeficientu moduļu summa tajā pašā rindā ar numuru i.

2.7. Aprēķini *MATLAB* vidē, izmantojot Jakobi metodi

2.3. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot Jakobi metodi.

$$\begin{cases} -4x_1 + x_2 + 2x_3 = 2 \\ 3x_1 - 7x_2 + 3x_3 = 29 \\ x_1 + 2x_2 + 5x_3 = 17 \end{cases}$$

Noskaidrot, vai izpildās konverģences pietiekamais nosacījums.

a) Ja konverģences pietiekamais nosacījums izpildās, atrast atrisinājumu ar precizitāti 10^{-3} un noteikt nepieciešamo iterāciju skaitu;
b) Ja konverģences pietiekamais nosacījums neizpildās, aprakstīt kļūdas izmaiņas atkarībā no iterāciju skaita.

Atrisinājums.

```
%% 2.3. piemērs.Jakobi metode
clc, clearvars, format compact
A = [-4,1,2;3,-7,3;1,2,5];  B = [2;29;17];
if det(A) == 0
    disp('Matrica A ir singulāra')
    disp('Atbilde: Jakobi metodi nedrīkst izmantot'), return
end
disp('Matrica A ir nesingulāra')
fun_prob3(A)  % pārbaude: Jakobi metode konverģē
```

```
Matrica A ir nesingulāra.
Izpildās konverģences pietiekamais nosacījums - Jakobi metode
konverģē.
```

Lai pārbaudītu, vai Jakobi metode konverģē, izmanto ārējo funkciju **fun _ prob3**.

```
% ārējā funkcija (2.3. piemērs). Jakobi metode
% pārbaude: vai Jakobi metode konverģē?
function fun_prob3(A_mat)
 [row,col] = size(A_mat);
 for i = 1:row
    sum =0;
    for j = 1:col
        if i~=j
            sum =sum+abs(A_mat(i,j));
        end
    end
    if abs(A_mat(i,i)) <= sum
      disp('Neizpildās konverģences pietiekamais nosacījums')
      fprintf('rindas numurs %.0f: --> %.0f < %.0f \n', i,A_mat(i,i),sum )
      return
    end
 end
 disp('Izpildās konver. pietiekamais nosacījums - Jakobi metode konverģē')
end
```

```
n = length(B);
x_app = zeros(n,1);
```

```
% 2.3. piemēra turpinājums
x_app = zeros(3,1);   % sākuma tuvinājums
iter = 0;             % iterāciju skaits
itermax = 20;         % maksimālais iterāciju skaits
epsi = 0.001;         % epsi=10^(-3) (aprēķinu precizitāte)
errnorm = 1           % kļūdas norma
prnorm = zeros(1,2); % datu masīvs rezultātu saglabāšanai
k = 1;
```

$$\text{x\_app} = \begin{pmatrix} x_1^{(0)} = 0 \\ x_2^{(0)} = 0 \\ x_3^{(0)} = 0 \end{pmatrix}$$

Šeit arī rekomendējam veikt izmaiņas skriptā, lai būtu vieglāk risināt vienādojumu sistēmas ar patvaļīgo nezināmo skaitu n.

```
% 2.3. piemēra turpinājums
while errnorm > epsi && iter < itermax
    k = k+1;  iter  = iter+1;
    for i = 1:3
      res_sum = 0;
      for j = 1:3
         if j~=i
           res_sum =res_sum+x_app(j,k-1)*A(i,j);
         end
      end
      x_app(i,k) =(B(i,1)-res_sum)/A(i,i);
    end
    errnorm  =norm(x_app(:,k)-x_app(:,k-1));
    prnorm(iter,:) =[iter,errnorm];
end
disp('Iter. numurs Kļūda')
disp(prnorm)
x_approx = x_app(:,k)
x_sol = linsolve(A,B) % Ctrl+Enter
```

```
for i = 1:n
```

$$\text{res\_sum} = \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)}$$

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

$$\left\| e^{(k)} \right\|_2 = \left\| X^{(k)} - X^{(k-1)} \right\|_2$$

```
Iter. numurs Kļūda
    1.0000    5.3827
    2.0000    2.2524
    3.0000    1.6995
    4.0000    0.6978
    5.0000    0.4128
    6.0000    0.2592
    7.0000    0.1005
    8.0000    0.0773
    9.0000    0.0350
   10.0000    0.0188
   11.0000    0.0125
   12.0000    0.0048
   13.0000    0.0036
   14.0000    0.0017
   15.0000    0.0008
```

```
x_approx =
    1.0002
   -1.9997
    3.9995
x_sol =
     1
    -2
     4
```

```
% 2.3. piemēra turpinājums
[row,col] = size(prnorm);
disp('Atbilde:')
fprintf('iterāciju skaits = %.0f -->',prnorm(row,1))
fprintf('kļūda = %.6f\n' ,prnorm(row,2))
fprintf('x_tuvinājumi: { %.4f , %.4f, %.4f }\n',x_approx(:)')
```

```
Atbilde:
  iterāciju skaits = 15 --> kļūda = 0.000849
  x_tuvinājumi: { 1.0002 , -1.9997, 3.9995 }
```

$$\mathbf{x}\_\text{app} = \begin{pmatrix} x_1^{(0)} & x_1^{(1)} & x_1^{(2)} \\ x_2^{(0)} & x_2^{(1)} & x_2^{(2)} \dots \\ x_3^{(0)} & x_3^{(1)} & x_3^{(2)} \end{pmatrix}$$

Ja ir izpildītas 15 iterācijas, tad atrisinājums pēc 15 iterācijām atrodas matricas **x _ app(3,16)** 16. kolonnā.

2.4. piemērs. Atrisināt vienādojumu sistēmu, izmantojot Jakobi metodi, ja $n=20$. Izpildīt astoņas iterācijas. Aprēķināt kļūdas normu pēc trīs iterācijām:

$$\left\|\boldsymbol{\varepsilon}^{(3)}\right\|_2, \text{ kur } \boldsymbol{\varepsilon}^{(3)} = \boldsymbol{x}^{(3)} - \boldsymbol{x}^{(2)}.$$

Aprēķināt atrisinājuma normu pēc astoņām iterācijām.

$$\begin{cases} -3x_1 + x_2 = 1 \\ x_1 - 3x_2 + x_3 = 1 \\ x_2 - 3x_3 + x_4 = 1 \\ \dots \\ x_{n-2} - 3x_{n-1} + x_n = 1 \\ x_{n-1} - 3x_n = 1 \end{cases}$$

Atrisinājums.

```
n =20; A =zeros(n,n);
```

```
%% 2.4. piemērs.Jakobi metode
clc, clearvars, format compact
A =zeros(20,20); B =ones(20,1);
for i = 1:19
   A(i,i) =-3; A(i,i+1) =1; A(i+1,i) =1;
end
A(20,20) =-3;
if det(A) == 0
    disp('Matrica A ir singulāra')
    disp('Atbilde: Jakobi metodi nedrīkst izmantot'), return
end
disp('Matrica A ir nesingulāra')
fun_prob3(A) % pārbaude: vai Jakobi metode konverģē?
```

```
Matrica A ir nesingulāra.
Izpildās konverģences pietiekamais nosacījums - Jakobi metode konverģē.
```

```
% 2.4. piemēra turpinājums
x_app = zeros(20,1); itermax = 8; k = 1;
errnorm =1; prnorm =zeros(1,2);
for iter = 1:itermax
    k = k+1;
    for i = 1:20
        res_sum = 0;
        for j =1 :20
            if j~=i
               res_sum = res_sum + x_app(j,k-1)*A(i,j);
            end
        end
        x_app(i,k) =(B(i,1)- res_sum)/A(i,i);
    end
    errnorm =norm((x_app(:,k)-x_app(:,k-1)),2);
    prnorm(iter,:) =[iter,errnorm];
end
disp('Iterācijas Kļūda')
disp(prnorm)
x_approx = x_app(:,k)  % Ctrl+Enter
```

```
Iterācijas     Kļūda
    1.0000     1.4907
    2.0000     0.9558
    3.0000     0.6220
    4.0000     0.4065
    5.0000     0.2663
    6.0000     0.1747
    7.0000     0.1147
    8.0000     0.0754
```

```
x_approx =
   -0.6084
   -0.8360
   -0.9188
   -0.9480
   -0.9573
   -0.9602
   -0.9608
   -0.9610
   -0.9610
   -0.9610
   -0.9610
   -0.9610
   -0.9608
   -0.9602
   -0.9573
   -0.9480
   -0.9188
   -0.8360
   -0.6084
```

```
% 2.4. piemēra turpinājums
epsi3_norm =norm(x_app(:,4)-x_app(:,3))
X_app8_norm =norm(x_app(:,9))% Ctrl+Enter
```

$$\varepsilon^{(3)} = \left\| \boldsymbol{x}^{(3)} - \boldsymbol{x}^{(2)} \right\|$$

```
epsi3_norm =
    0.6220
X_app8_norm =
    4.0851
```

$$\mathbf{x\_app} = \begin{pmatrix} x_1^{(0)} & x_1^{(1)} & x_1^{(2)} & \\ x_2^{(0)} & x_2^{(1)} & x_2^{(2)} & \dots \\ x_3^{(0)} & x_3^{(1)} & x_3^{(2)} & \\ \dots & \dots & \dots & \end{pmatrix}$$

Ja ir izpildītas astoņas iterācijas, tad atrisinājums pēc astoņām iterācijām atrodas matricas **x _ app(20,9)** 9. kolonnā.

```
% 2.4. piemēra turpinājums
disp('Atbilde:')
fprintf('iterāciju skaits = %.0f --> ',prnorm(3,1))
fprintf('kļūda = %.5f\n' ,prnorm(3,2))
disp(['X_app8_norm = ',num2str(X_app8_norm) ])
disp('x_tuvinājumi:')
fprintf(' %.5f  %.5f  %.5f  %.5f  %.5f\n ',x_approx(:))
```

```
Atbilde:
 iterāciju skaits = 3 --> kļūda = 0.62196
 X_app8_norm = 4.0851
 x_tuvinājumi:
  -0.60844  -0.83600  -0.91876  -0.94803  -0.95732
   -0.96022  -0.96083  -0.96098  -0.96098  -0.96098
   -0.96098  -0.96098  -0.96098  -0.96083  -0.96022
   -0.95732  -0.94803  -0.91876  -0.83600  -0.60844
```

2.8. Vienkāršā iterāciju metode

Aplūkosim lineāru vienādojumu sistēmu (2.3), kur A ir $m \times m$ nesingulāra matrica. Pieņemsim, ka D ir nesingulāra $m \times m$ matrica. Pārveidosim vienādojumu (2.3) šādi:

$$\boldsymbol{x} = \boldsymbol{x} - D\left(A\boldsymbol{x} - \boldsymbol{b}\right).$$

Iegūto vienādojumu pārrakstīsim šādi:

$$\boldsymbol{x} = F\boldsymbol{x} + \boldsymbol{c}, \tag{2.7}$$

kur $F = E - DA$ un $\boldsymbol{c} = D\boldsymbol{b}$.

Izmantojot sākuma tuvinājumu $\boldsymbol{x}^{(0)}$, konstruēsim iterāciju metodi pēc formulas

$$\boldsymbol{x}^{(n+1)} = F\boldsymbol{x}^{(n)} + \boldsymbol{c}, \quad n = 0,1,\ldots \tag{2.8}$$

Metodi (2.8) sauc par vienkāršo iterāciju metodi.

Ja tuvinājums $\boldsymbol{x}^{(n+1)}$ ir atkarīgs tikai no $\boldsymbol{x}^{(n)}$, iterāciju metodi sauc par divu slāņu jeb viena soļa metodi.

2.1. teorēma. Konverģences pietiekamais nosacījums. Ja $\|F\| < 1$, tad metode (2.8) konverģē.

Pierādījums. Izmantojot vektora un matricas normas definīciju, iegūstam

$$\|A\boldsymbol{x}\| \le \|A\|\|\boldsymbol{x}\|. \tag{2.9}$$

Ja $\boldsymbol{x}$ ir vienādojuma (2.7) precīzs atrisinājums, tad kļūdu iterācijas solī ar numuru n definē pēc formulas

$$\boldsymbol{r}^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{x}.$$

Kļūda nākamajā solī ir

$$\boldsymbol{r}^{(n+1)} = \boldsymbol{x}^{(n+1)} - \boldsymbol{x} = F\boldsymbol{x}^{(n)} + \boldsymbol{c} - \left(F\boldsymbol{x} + \boldsymbol{c}\right) = F\left[\boldsymbol{x}^{(n)} - \boldsymbol{x}\right] = F\boldsymbol{r}^{(n)}.$$

Tādējādi

$$\boldsymbol{r}^{(n+1)} = F\boldsymbol{r}^{(n)}. \tag{2.10}$$

Izmantojot formulu (2.10), iegūstam

$$\boldsymbol{r}^{(1)} = F\boldsymbol{r}^{(0)}, \; \boldsymbol{r}^{(2)} = F\boldsymbol{r}^{(1)} = F^2\boldsymbol{r}^{(0)} \text{ utt.}$$

Rezultātā iegūstam

$$\boldsymbol{r}^{(n)} = F^n\boldsymbol{r}^{(0)}. \tag{2.11}$$

Izmantojot (2.9) un (2.11), atrodam

$$\left\|\boldsymbol{r}^{(n)}\right\| \le \|F\|^n \left\|\boldsymbol{r}^{(0)}\right\| \to 0 \text{ tāpēc ka } \|F\| < 1. \tag{2.12}$$

Piezīme. Rezultāts nav atkarīgs no normas $\|F\|$ izvēles, bet konverģences ātrums ir atkarīgs no izvēlētās normas.

Aplūkosim iterācijas metodi

$$B_{n+1}\frac{\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}}{\tau_{n+1}} + A\boldsymbol{x}^{(n)} = \boldsymbol{b}, \tag{2.13}$$

kur B_{n+1} ir kvadrātiska matrica, kas raksturo konkrēto metodi, un τ_{n+1} ir parametrs, kas ir atkarīgs no iterācijas soļa n.

Pieņemsim, ka sākuma tuvinājums $\boldsymbol{x}^{(0)}$ ir izvēlēts un matrica B_n^{-1} eksistē visiem $n = 1, 2, \ldots, N-1$. Atrisinot vienādojumu (2.13), iegūstam $\boldsymbol{x}^{(n)}$, $n = 1, 2, \ldots, N$.

Iterācijas metodi (2.13) sauc par **atklātu**, ja $B_n = E$, kur E ir $m \times m$ vienības matrica, un **aizklātu** pretējā gadījumā (ja $B_n \neq E$).

Iterācijas metodi (2.13) sauc par **stacionāru**, ja $B_n = B$ un $\tau_n = \tau$ ir neatkarīgi no iterācijas soļa n, un par **nestacionāru** pretējā gadījumā.

Pārveidosim vienādojumu (2.13) šādi:

$$B_{n+1}\frac{\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}}{\tau_{n+1}}+A\boldsymbol{x}^{(n)}=\boldsymbol{b}\rightarrow B_{n+1}\left[\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}\right]=\tau_{n+1}\left[\boldsymbol{b}-A\boldsymbol{x}^{(n)}\right].$$

Atrisinot vienādojumu attiecībā pret $\boldsymbol{x}^{(n+1)}$, iegūstam

$$\boldsymbol{x}^{(n+1)}=\boldsymbol{x}^{(n)}+\tau_{n+1}B_{n+1}^{-1}\left[\boldsymbol{b}-A\boldsymbol{x}^{(n)}\right]. \tag{2.14}$$

Aplūkosim dažus vienādojuma (2.14) speciālos gadījumus.

1. Pieņemsim, ka $\tau_{n+1}=1$, $B_{n+1}=D$, kur D ir diagonālā matrica, kuras visi elementi, kas atrodas uz galvenās diagonāles, sakrīt ar matricas A galvenās diagonāles elementiem. Tā ir Jakobi metode.
2. Pieņemsim, ka $B_{n+1}=E$, $\tau_{n+1}=\tau>0$. Tā ir vienkāršā iterāciju metode (šajā gadījumā vienādojumu (2.14) var pārrakstīt veidā (2.8), kur $F=E-\tau A$, $\boldsymbol{c}=\tau\boldsymbol{b}$).

Definēsim vienkāršo iterācijas metodi

$$\boldsymbol{x}^{(n+1)}=\boldsymbol{x}^{(n)}+\tau\left[\boldsymbol{b}-A\boldsymbol{x}^{(n)}\right],\quad n=0,1,\ldots \tag{2.15}$$

pie nosacījuma, ka A ir simetriska un pozitīvi definēta matrica.

2.2. teorēma. Pieņemsim, ka A ir simetriska un pozitīvi definēta matrica. Iterāciju metode (2.15) konverģē pie nosacījuma

$$E-\frac{1}{2}\tau A>0. \tag{2.16}$$

Apzīmēsim ar λ_i, $i=1, 2, \ldots, m$ matricas A īpašvērtības, kas ir sakārtotas dilstošā secībā.

Nosacījums (2.16) ir ekvivalents nosacījumam, ka visas matricas $E-0.5\tau A$ īpašvērtības ir pozitīvi skaitļi. Tādējādi visām matricas $E-0.5\tau A$ īpašvērtībām ir spēkā nevienādība: $\tau<2/\lambda_i$, $i=1,2,\ldots,m$. Tas nozīmē, ka konverģences nosacījums ir

$$\tau<\frac{2}{\lambda_{\max}}, \tag{2.17}$$

kur $\lambda_{\max}$ ir vislielākā matricas A īpašvērtība.

Aplūkosim vēlreiz nosacījumu (2.12). Ja matricas F norma ir mazāka nekā 1, tad vienkāršā iterācijas metode konverģē. Konverģences ātrums ir atkarīgs no vērtības $\rho=\|F\|$ (koeficientu ρ sauc par kļūdas samazināšanas koeficientu vienā iterācijā). Ja skaitlis ρ ir ļoti tuvs 1, tad vienkāršā iterācijas metode konverģē lēni. Metodes (2.15) konverģences ātrums ir atkarīgs no τ. Metodes (2.15) kļūdas samazināšanas koeficientu aprēķina pēc formulas

$$\rho=\left\|E-\frac{1}{2}\tau A\right\|. \tag{2.18}$$

Pieņemsim, ka nosacījums (2.17) ir izpildīts. Vai ir iespējams atrast optimālo parametra τ vērtību tā, lai iterāciju skaits būtu minimāls? Atbilde uz šo jautājumu ir pozitīva: optimālā parametra τ vērtība ir

$$\tau_{\text{opt}}=\frac{2}{\lambda_{\max}+\lambda_{\min}}. \tag{2.19}$$

2.9. Aprēķini *MATLAB* vidē, izmantojot vienkāršo iterāciju metodi

2.5. piemērs. Atrisināt lineāro vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi ar precizitāti $\varepsilon = 10^{-3}$. Pieņemt $\tau = 10^{-2}$.

$$\begin{cases} 5x_1 + 3x_2 + 7x_3 = 32 \\ 3x_1 + 12x_2 - 4x_3 = 15 \\ 7x_1 - 4x_2 + 52x_3 = 155 \end{cases}$$

Atrisinājums.

```
%% 2.5. piemērs. Vienkāršā iterāciju metode
clc, clearvars, format compact
A =[5,3,7;3,12,-4;7,-4,52]; B =[32;15;155];
if det(A) == 0
    disp('Matrica A ir singulāra')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
disp('Matrica A ir nesingulāra')

ni = fun_prob5(A); % pārbaude, vai matrica ir pozitivi definēta
if ni == 2
    disp('Koeficientu matrica nav pozitīvi definēta')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
check=isequal(A,A');
if check==0
    disp('Koeficientu matrica nav simetriska')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
disp('Koeficientu matrica ir simetriska un pozitīvi definēta')
```

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

Lai pārbaudītu, vai matrica ir pozitīvi definēta, izmanto ārējo funkciju **fun _ prob5**.

```
%% ārējā funkcija(2.5. piemērs). Vienkāršā iterāciju metode
% pārbaude: vai matrica ir pozitīvi definēta
function ni = fun_prob5(A_mat)
   ni = 1;
   [row,col] = size(A_mat);
   for i = 1:row
      if det(A_mat(1:i,1:i))>0
      else ni = 2; break
      end
   end
end
```

```
% 2.5. piemēra turpinājums
tau = 0.01;
epsi = 10^(-3);        % aprēķinu precizitāte
itermax = 1000;        % max iterāciju skaits
x_app = zeros(3,1);    % sākuma tuvinājums
k_iter = 0; resid = B-A*x_app;
while norm(resid) > epsi && k_iter < itermax
    x_app = x_app + tau*resid;
    resid=b-A*x_app; k_iter = k_iter +1;
end
k_iter, x_app, x_sol=linsolve(A,b)    % Ctrl+Enter
```

```
n =length(B);
x_app =zeros(n,1)
```

$$x^{(n+1)} = x^{(n)} + \tau\left(b - Ax^{(n)}\right)$$

```
k_iter =
   251
x_app =
    1.0003
    1.9999
    2.9999
```

```
x_sol =
    1.0000
    2.0000
    3.0000
```

```
% 2.5. piemēra turpinājums
disp('Atbilde:')
disp(['iter. skaits = ' num2str(k_iter) ])
fprintf('x_tuvinājumi: { %.4f, %.4f, %.4f }\n',x_app(:))
```

```
Atbilde:
  iter. skaits = 251
  x_tuvinājumi: { 1.0003, 1.9999, 2.9999 }
```

Iterāciju metode konverģē pēc 251 iterācijas (ar precizitāti $\varepsilon = 10^{-3}$). Vienādojumu sistēmas precīzs atrisinājums: $x_1 = 1, x_2 = 2, x_3 = 3$.

2.6. piemērs. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi ar precizitāti $\varepsilon = 10^{-3}$.

$$\begin{cases} 5x_1 + 3x_2 + 7x_3 = 32 \\ 3x_1 + 12x_2 - 4x_3 = 15 \\ 7x_1 - 4x_2 + 52x_3 = 155 \end{cases}$$

Pieņemt:

a) $\tau = 0.01$ (2.5. piemērs);
b) $\tau = 0.02$;
c) $\tau = 0.03$;
d) $\tau = 0.04$.

```
tau =
    0.0100
k_iter =
   251
x_app =
    1.0003
    1.9999
    2.9999
```

```
tau =
    0.0200
k_iter =
   124
x_app =
    1.0003
    1.9999
    2.9999
```

```
tau =
    0.0300
k_iter =
    82
x_app =
    1.0003
    1.9999
    2.9999
```

```
tau =
    0.04
k_iter =
    1000
x_app =
     -6.38103603492399e+53
      3.96522189022496e+53
     -4.57495259388924e+54
```

```
format longG
```

```
Atbilde:
 tau = 0.01, 0.02, 0.03 - metode konverģē,
 bet tau = 0.04 - metode diverģē
```

2.7. piemērs. Atrast parametra vērtību τ_{max}, kas definē intervālu $(0, \tau_{max})$, kura vienkāršā iterāciju metode konverģē ($\tau_{max} = 2/\lambda_{max}$, sk. formulu (2.17)). Izmantot ieejas datus 2.5. piemēram.

Atrisinājums.

```
%% 2.7. piemērs. Atrast tau_max
% ieejas dati (2.5. piemērs).
clc, clearvars, format compact
A=[5,3,7;3,12,-4;7,-4,52];
eig_val = eig(A), tau_max = 2/max(eig_val) % Ctrl+Enter
```

```
eig_val =
    2.6054
   13.0716
   53.3230
tau_max =
    0.0375
```

```
% 2.7. piemēra turpinājums
disp('Atbilde:')
disp(['īpašvērtības = {' num2str(eig_val(:)') '}'])
disp(['tau_max = ' num2str(tau_max)])
```

```
Atbilde:
 īpašvērtības = {2.60539      13.0716      53.323}
 tau_max = 0.037507
```

2.8. piemērs. Noteikt kļūdas samazināšanas koeficientu vienā iterācijā (izmantot 2.5. piemēra ieejas datus).

Atrisinājums.

```
%% 2.8. piemērs. Atrast kļūdas samazināšanas koeficientu vienā iterācijā
% ieejas dati (2.5. piemērs).
clc, clearvars, format compact
A = [5,3,7;3,12,-4;7,-4,52];
E_mat = eye(3); tau = 0.01;
F = E_mat-tau*A; ro=norm(F,2)  % Ctrl+Enter
```

```
ro =
    0.9739
```

```
% 2.8. piemēra turpinājums
disp('Atbilde:')
disp(['ro = '  num2str(ro) ' ir ļoti tuvs 1'])
disp('tāpēc iterāciju skaits ir tik liels, n = 251')
```

```
Atbilde:
 ro = 0.9739 ir ļoti tuvs 1,
 tāpēc iterāciju skaits ir tik liels, n = 251
```

2.9. piemērs. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi ar precizitāti $\varepsilon = 10^{-3}$. Pieņemt optimālo parametra τ vērtību.

$$\begin{cases} 4x_1 - 2x_2 + 3x_3 = 5 \\ -2x_1 + 6x_2 - x_3 = 3 \\ 3x_1 - x_2 + 12x_3 = 14 \end{cases}$$

```
%% 2.9. piemērs. Vienkāršā iterāciju metode
% Pieņemt optimālo parametra tau vērtību
clc, clearvars, format compact
A =[4 -2 3;-2 6 -1;3 -1 12]; B =[5;3;14];
if det(A) == 0
    disp('Matrica A ir singulāra')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
disp('Matrica A ir nesingulāra')
ni = fun_prob5(A); % pārbaude, vai matrica ir pozitīvi definēta
if ni == 2
    disp('Koeficientu matrica nav pozitīvi definēta')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
check=isequal(A,A');
if check==0
    disp('Koeficientu matrica nav simetriska')
    disp('Atbilde: vienkāršo iterāciju metodi nedrīkst izmantot')
    return
end
disp('Koeficientu matrica ir simetriska un pozitīvi definēta')
% Ctrl+Enter
```

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
% 2.9. piemēra turpinājums
x_app = zeros(3,1); epsi = 10^(-3); itermax = 1000;
lambda = eig(A); tau_opt = 2/(max(lambda)+min(lambda));
k_iter = 0; resid = B-A*x_app;
while norm(resid) > epsi && k_iter < itermax
    x_app = x_app+tau_opt*resid;
    resid = B-A*x_app;
    k_iter = k_iter +1;
end
tau_opt, k_iter, x_app % Ctrl+Enter
```

$$\tau_{\text{opt}} = \frac{2}{\lambda_{\max} + \lambda_{\min}}$$

```
% 2.9. piemēra turpinājums
disp('Atbilde:')
disp([' iter. skaits = ' num2str(k_iter) ])
disp([' x_tuvinajumi = {' num2str(x_app(:)') '}'])
disp([' optimāla parametra tau vērtība = ' num2str(tau_opt)])
```

```
tau_opt =
    0.1279
k_iter =
    28
x_app =
    0.9999
    1.0000
    1.0000
```

```
Atbilde:
 iter. skaits = 28
 x_tuvinajumi = {0.99992     0.99999     0.99996}
 optimāla parametra tau vērtība = 0.12793
```

2.10. Minimālās nesaistes metode

Pieņemsim, ka A ir simetriska un pozitīvi definēta matrica. Aplūkosim iterācijas metodi.

$$\frac{\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}}{\tau_{n+1}}+A\boldsymbol{x}^{(n)}=\boldsymbol{b} \tag{2.19}$$

Parametru τ_{n+1} katrā solī izvēlas tā, lai kļūda (jeb nesaiste) būtu minimāla. Nesaisti iterācijas solī ar numuru n aprēķina pēc formulas (2.20).

$$\boldsymbol{r}^{(n)}=A\boldsymbol{x}^{(n)}-\boldsymbol{b} \tag{2.20}$$

Izmantojot (2.19) un (2.20), iegūstam formulu (2.21).

$$\boldsymbol{x}^{(n+1)}=\boldsymbol{x}^{(n)}-\tau_{n+1}\boldsymbol{r}^{(n)} \tag{2.21}$$

Lai iegūtu formulu koeficienta τ_{n+1} aprēķināšanai, sareizināsim (2.21) ar A no kreisās puses un atņemsim $\boldsymbol{b}$. Rezultāts ir

$$A\boldsymbol{x}^{(n+1)}-\boldsymbol{b}=A\boldsymbol{x}^{(n)}-\boldsymbol{b}-\tau_{n+1}A\boldsymbol{r}^{(n)}$$

Iegūto vienādojumu pārrakstīsim šādi:

$$\boldsymbol{r}^{(n+1)}=\boldsymbol{r}^{(n)}-\tau_{n+1}A\boldsymbol{r}^{(n)} \tag{2.22}$$

Aprēķināsim vektoru $\boldsymbol{r}^{(n+1)}$ un $\boldsymbol{r}^{(n+1)}$ skalāro reizinājumu.

$$\left(\boldsymbol{r}^{(n+1)},\boldsymbol{r}^{(n+1)}\right)=\left(\boldsymbol{r}^{(n)}-\tau_{n+1}A\boldsymbol{r}^{(n)},\boldsymbol{r}^{(n)}-\tau_{n+1}A\boldsymbol{r}^{(n)}\right)$$

Vienkāršojot iegūstam

$$\left\|\boldsymbol{r}^{(n+1)}\right\|^2=\left\|\boldsymbol{r}^{(n)}\right\|^2-2\tau_{n+1}\left(\boldsymbol{r}^{(n)},A\boldsymbol{r}^{(n)}\right)+\tau_{n+1}^2\left\|A\boldsymbol{r}^{(n)}\right\|^2 \tag{2.23}$$

Mūsu mērķis ir minimizēt $\left\|\boldsymbol{r}^{(n+1)}\right\|$. Vienīgais nezināmais vienādojuma (2.23) labajā pusē iterācijā ar numuru $n+1$ ir τ_{n+1}. Tā kā izteiksme vienādojuma (2.23) labajā pusē ir otrās pakāpes polinoms, minimums ir sasniegts punktā, kurā atvasinājums ir nulle. Diferencējot (2.23) pēc mainīgā τ_{n+1} un pielīdzinot atvasinājumu nullei, iegūstam

$$\tau_{n+1}=\frac{\left(\boldsymbol{r}^{(n)},A\boldsymbol{r}^{(n)}\right)}{\left\|A\boldsymbol{r}^{(n)}\right\|^2} \tag{2.24}$$

Tādējādi minimālās nesaistes metodi var aprakstīt šādi: izmantojot sākuma tuvinājumu $\boldsymbol{x}^{(0)}$, katrā iterācijas solī aprēķina τ_{n+1} pēc formulas (2.24). Nākamo tuvinājumu atrisinājumam iegūst, izmantojot formulu (2.21).

2.11. Aprēķini *MATLAB* vidē, izmantojot minimālās nesaistes metodi

2.10. piemērs. Atrisināt lineāru vienādojumu sistēmu, izmantojot minimālās nesaistes metodi.

$$\begin{cases} 4x_1 - 2x_2 + 3x_3 = 5 \\ -2x_1 + 6x_2 - x_3 = 3 \\ 3x_1 - x_2 + 12x_3 = 14 \end{cases}$$

Atrisinājums.

```
%% 2.10. piemērs. Minimālās nesaistes metode
clc, clearvars, format compact
A = [4,-2,3;-2,6,-1;3,-1,12]; B = [5;3;14];
if det(A) == 0
    disp('Matrica A ir singulāra')
    disp('Atbilde: minimālo nesaistes metodi nedrīkst izmantot')
    return
end
disp('Matrica A ir nesingulāra')
```

```
ni = fun_prob10(A); % pārbaude, vai matrica ir pozitīvi definēta
if ni == 2
    disp('Koeficientu matrica nav pozitīvi definēta')
    disp('Atbilde: minimālo nesaistes metodi nedrīkst izmantot')
    return
end
check=isequal(A,A');
if check==0
    disp('Koeficientu matrica nav simetriska')
    disp('Atbilde: minimālo nesaistes metodi nedrīkst izmantot')
    return
end
disp('Koeficientu matrica ir simetriska un pozitīvi definēta ')
```

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

Lai pārbaudītu, vai matrica ir pozitīvi definēta, izmanto ārējo funkciju **fun _ prob10**.

```
%% ārēja funkcija(2.10. piemērs.) Minimālās nesaistes metode
% pārbaude: vai matrica ir pozitīvi definēta
function ni = fun_prob10(A_mat)
   ni = 1;
   [row,col] = size(A_mat);
   for i = 1:row
      if det(A_mat(1:i,1:i))>0
      else ni = 2; break
      end
   end
end
```

```
% 2.10. piemēra turpinājums
k_iter = 0; epsi = 10^(-3);itermax =300;
x_app=zeros(3,1);
r = A*x_app-B; norm_r = norm(r);
while norm_r > epsi && k_iter < itermax
    k_iter  =k_iter+1;
    tau = ((A*r)'*r)/norm(A*r)^2;
    x_app = x_app-(tau*r')';r = A*x_app-B; norm_r = norm(r);
end
k_iter,tau, x_app, norm_r
x_sol = linsolve(A,B)
```

```
n = length(B);
```

```
k_iter =
    23
tau =
    0.1089
x_app =
    0.9997
    0.9999
    1.0001
norm_r =
7.3505e-04
```

```
x_sol =
     1
     1
     1
```

```
% 2.10. piemēra turpinājums
disp('Atbilde:')
fprintf('iter. skaits = %.f, nesaistes norma = %.8f\n', k_iter,norm_r)
disp(['x_tuvinājumi: {' num2str(x_app(:)') '}'])
```

```
Atbilde:
 iter. skaits = 23, nesaistes norma = 0.00073505
 x_tuvinājumi: {0.99974     0.99987      1.0001}
```

Iterāciju metode konverģē pēc 23 iterācijām. Vienādojumu sistēmas precīzs atrisinājums: $x_1 = 1, x_2 = 2, x_3 = 3$.

2.12. Matricas vislielākās (pēc moduļa) īpašvērtības aprēķināšana

Mēs zinām, kā aprēķināt matricas A īpašvērtības. *MATLAB* vidē var izmantot komandu eig(A). Ja matricas rindu un kolonnu skaits n ir liels (piemēram, $n=100$), aprēķini prasa laiku. Bieži nav jāzina visas matricas īpašvērtības, bet tikai vislielākā (pēc moduļa) īpašvērtība.

Pieņemsim, ka A ir kvadrātiska matrica ar kārtu m, kurai visas īpašvērtības $\lambda_1, \lambda_2, \ldots, \lambda_m$ ir atšķirīgas. Var pierādīt, ka šajā gadījumā atbilstoši īpašvektori ir lineāri neatkarīgi. Pieņemsim, ka īpašvērtības ir sakārtotas dilstošā secībā (pēc moduļa):

$$|\lambda_1| > |\lambda_2| \ldots > |\lambda_m| \tag{2.25}$$

Apzīmēsim ar $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m$ matricas A normalizētus īpašvektorus (īpašvektoram $\boldsymbol{e}_i$ atbilst īpašvērtība λ_i). Lineāri neatkarīgi īpašvektori $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m$ veido bāzi m – dimensiju telpā. Tas nozīmē, ka jebkuru vektoru (piemēram, sākuma tuvinājumu $\boldsymbol{x}^{(0)}$) var vienā vienīgā veidā uzrakstīt kā bāzes vektoru lineāru kombināciju:

$$\boldsymbol{x}^{(0)} = c_1\boldsymbol{e}_1 + c_2\boldsymbol{e}_2 + \ldots + c_m\boldsymbol{e}_m \tag{2.26}$$

Definēsim vektoru virkni, izmantojot formulu

$$\boldsymbol{x}^{(n+1)} = A\boldsymbol{x}^{(n)}, \quad n = 0,1,\ldots \tag{2.27}$$

Izmantojot īpašvērtības definīciju ($A\boldsymbol{x} = \lambda\boldsymbol{x}$), iegūstam

$$\boldsymbol{x}^{(1)} = A\boldsymbol{x}^{(0)} = A\left[c_1\boldsymbol{e}_1 + c_2\boldsymbol{e}_2 + \ldots + c_m\boldsymbol{e}_m\right] = c_1\lambda_1\boldsymbol{e}_1 + c_2\lambda_2\boldsymbol{e}_2 + \ldots + c_m\lambda_m\boldsymbol{e}_m$$

Analoģiski

$$\boldsymbol{x}^{(2)} = A\boldsymbol{x}^{(1)} = A\left[c_1\lambda_1\boldsymbol{e}_1 + c_2\lambda_2\boldsymbol{e}_2 + \ldots + c_m\lambda_m\boldsymbol{e}_m\right] = c_1\lambda_1^2\boldsymbol{e}_1 + c_2\lambda_2^2\boldsymbol{e}_2 + \ldots + c_m\lambda_{\mathrm{m}}^2\boldsymbol{e}_m$$

Atkārtojot šo procedūru, iegūstam

$$\boldsymbol{x}^{(n)} = A\boldsymbol{x}^{(n-1)} = A\left[c_1\lambda_1^{n-1}\boldsymbol{e}_1 + c_2\lambda_2^{n-1}\boldsymbol{e}_2 + \ldots + c_m\lambda_m^{n-1}\boldsymbol{e}_m\right] = c_1\lambda_1^n\boldsymbol{e}_1 + c_2\lambda_2^n\boldsymbol{e}_2 + \ldots + c_m\lambda_m^n\boldsymbol{e}_m =$$

$$= \lambda_1^n\left[c_1\boldsymbol{e}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^n\boldsymbol{e}_2 + \ldots + c_m\left(\frac{\lambda_m}{\lambda_1}\right)^n\boldsymbol{e}_m\right]$$

Tā kā λ_1 ir īpašvērtība ar vislielāko moduli,

$$\lim_{n\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^n = 0 \text{ visiem } i = 2, 3, \ldots, m$$

Tādējādi, ja n ir pietiekami liels, vektoru $\boldsymbol{x}^{(n)}$ var aproksimēt šādi:

$$\boldsymbol{x}^{(n)} \approx \lambda_1^n c_1\boldsymbol{e}_1 \tag{2.28}$$

Izmantojot formulu (2.28), aprēķināsim skalāros reizinājumus:

$$\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n-1)}\right) \approx \lambda_1^{2n-1}c_1^2\boldsymbol{e}_1^{\,2}$$

$$\left(\boldsymbol{x}^{(n-1)}, \boldsymbol{x}^{(n-1)}\right) \approx \lambda_1^{2n-2}c_1^2\boldsymbol{e}_1^{\,2}$$

Rezultātā iegūstam

$$\lambda_1 = \lim_{n\to\infty}\frac{\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n-1)}\right)}{\left(\boldsymbol{x}^{(n-1)}, \boldsymbol{x}^{(n-1)}\right)} \tag{2.29}$$

Aprakstīsim aprēķinu algoritmu.

1. solis. Izvēlēsimies patvaļīgu vektoru $\boldsymbol{x}^{(0)} \neq 0$ un aprēķināsim $\boldsymbol{e}^{(0)} = \dfrac{\boldsymbol{x}^{(0)}}{\left\|\boldsymbol{x}^{(0)}\right\|}$. Tādējādi $\left\|\boldsymbol{e}^{(0)}\right\| = 1$.

2. solis. Sākot ar $n = 1$ aprēķināsim $\boldsymbol{x}^{(n)} = A\boldsymbol{e}^{(n-1)}$, $\lambda_1^{(n)} = \left(\boldsymbol{x}^{(n)}, \boldsymbol{e}^{(n-1)}\right)$.

3. solis. Katrā solī ir jānormalizē $\boldsymbol{x}^{(n)}$: $\boldsymbol{e}^{(n)} = \dfrac{\boldsymbol{x}^{(n)}}{\left\|\boldsymbol{x}^{(n)}\right\|}$.

4. solis. Lai pabeigtu aprēķinus, izmanto vienu no konverģences kritērijiem: $\left\|\boldsymbol{e}^{(n)} - \boldsymbol{e}^{(n-1)}\right\| < \varepsilon$ vai $\left|\lambda_1^{(n)} - \lambda_1^{(n-1)}\right| < \varepsilon$.

2.13. Aprēķini *MATLAB* vidē, lai aprēķinātu matricas vislielāko īpašvērtību

2.11. piemērs. Atrast vislielāko (pēc moduļa) matricas A īpašvērtību, izmantojot iterācijas metodi (precizitāte 10^{-3}):

$$A = \begin{pmatrix} 4 & 1 & 3 & 2 \\ 2 & 5 & 6 & 1 \\ 7 & 4 & 8 & 3 \\ 2 & 1 & 5 & 9 \end{pmatrix}$$

Atrisinājums.

```
%% 2.11. piemērs. Vislielākās īpašvērtības aprēķināšana.
clc, clearvars, format compact
A = [ 4,1,3,2;2,5,6,1;7,4,8,3;2,1,5,9];
x_app = ones(4,1); e_mas(:,1) = x_app/norm(x_app);
x_app(:,2) = A*e_mas; e_mas(:,2) = x_app(:,2)/norm(x_app(:,2));
k = 2; epsi = 10^(-3); iter_max = 20; k_iter = 0;
while norm(e_mas(:,k)-e_mas(:,k-1)) > epsi && k <= iter_max
    x_app(:,k+1) = A*e_mas(:,k);
    e_mas(:,k+1) = x_app(:,k+1)/norm(x_app(:,k+1));
    k_iter = k_iter+1; lambda = dot( x_app(:,k+1)',e_mas(:,k));
    x_pr = x_app(:,k+1); e_pr = e_mas(:,k+1);
    k = k+1;
end
k_iter, lambda, x_pr, e_pr
eig_val_maxv = eigs(A,1) % pēc moduļa lielākā īpašvērtība
```

```
n = length(A(1,:));
```

```
k_iter =
     5
lambda =
   16.4290
x_pr =
    4.6145
    7.1580
   10.5393
    9.2901
```

```
e_pr =
    0.2809
    0.4357
    0.6415
    0.5655
eig_val_max =
   16.4286
```

```
% 11. piemēra turpinājums
disp('Atbilde:')
disp([' iterāciju skaits = ' num2str(k_iter) ])
disp([' lielākā īpašvērtība = ' num2str(lambda) '( ar prec. 10^(-3))'])
```

```
Atbilde:
 iterāciju skaits = 5
 lielākā īpašvērtība = 16.429(ar precizitāti 10^(-3 ))}
```

UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

2.1. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Jakobi metodi.

$$\begin{cases} 3x_1 - 7x_2 + 3x_3 = 29 \\ -4x_1 + x_2 + 2x_3 = 2 \\ x_1 + 2x_2 + 5x_3 = 17 \end{cases}$$

Noskaidrot, vai izpildās konverģences pietiekamais nosacījums.

a) Ja konverģences pietiekamais nosacījums izpildās, atrast atrisinājumu ar precizitāti 10^{-3} un noteikt nepieciešamo iterāciju skaitu.

b) Ja konverģences pietiekamais nosacījums neizpildās, aprakstīt kļūdas izmaiņu atkarībā no iterāciju skaita.

```
%% 2.1. uzdevums. Jakobi metode
clc, clearvars, format compact
format longG
… … …
```

```
Neizpildās konverģences pietiekamais nosacījums
 rindas numurs 1: --> 3 < 10
```

```
Iter.numurs      Kļūda
    1          10.4405193570265
    2          32.0087488040379
    3           78.883761961728
    4          337.032798755606
    5          834.003818587081
    6          3538.12916306385
    7          8946.33230616427
    8          37441.7419987472
    9          97015.6971483096
    10           399132.40334369
    11          1060945.33302116
    12          4282389.99885737
    13          11674541.6411063
    14          46204761.0548696
    15          129029686.004484
    16          500907316.441708
    17          1430295630.36982
    18          5452146995.93908
    19          15885121632.4982
    20          59542205680.3326
```

```
x_approx =
          41488236634.3623
          69692399420.1283
         -9704289951.64887
x_sol =
     1
    -2
     4
```

```
Atbilde:
 iterāciju skaits = 20
 kļūda (aug)= 59542205680.3326
 Jakobi metode diverģē
```

2.2. uzdevums. Atrisināt lineāru vienādojumu sistēmu, izmantojot Jakobi metodi, ja $n = 30$.

$$\begin{cases} -3x_1 + x_2 = 1 \\ x_1 - 3x_2 + x_3 = 1 \\ x_2 - 3x_3 + x_4 = 1 \\ \dots \\ x_{n-2} - 3x_{n-1} + x_n = 1 \\ x_{n-1} - 3x_n = 1 \end{cases}$$

Izpildīt 7 iterācijas.

- Aprēķināt kļūdas normu $\left\|\boldsymbol{\varepsilon}^{(5)}\right\|_2$ 5. iterācijā, kur $\boldsymbol{\varepsilon}^{(5)} = \boldsymbol{x}^{(5)} - \boldsymbol{x}^{(4)}$.
- Aprēķināt atrisinājuma normu $\left\|\boldsymbol{x}^{(6)}\right\|_2$ 6. iterācijā.
- Atrisināt šo sistēmu, arī izmantojot komandu **`linsolve`** (ar komandu **`linsolve`** iegūto rezultātu apzīmēsim ar ***x*_sol**).
- Cik liela ir norma $\left\|\boldsymbol{x}\_\mathbf{sol} - \boldsymbol{x}^{(4)}\right\|_2$?

```
Matrica A ir nesingulāra.
Izpildās konverģences
pietiekamais nosacījums -
Jakobi metode konverģē.
```

```
Iterācijas     Kļūda
    1.0000     1.8257
    2.0000     1.1863
    3.0000     0.7787
    4.0000     0.5126
    5.0000     0.3380
    6.0000     0.2231
    7.0000     0.1474
```

```
epsi5_norm =
    0.3380
X_app6_norm =
    4.8490
```

```
sol_X4_norm =
    0.9957
```

```
x_approx =
   -0.6031
   -0.8253
   -0.9049
   -0.9310
   -0.9392
   -0.9410
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9415
   -0.9410
   -0.9392
   -0.9310
   -0.9049
   -0.8253
   -0.6031
```

```
x_sol =
   -0.6180
   -0.8541
   -0.9443
   -0.9787
   -0.9919
   -0.9969
   -0.9988
   -0.9995
   -0.9998
   -0.9999
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -0.9999
   -0.9998
   -0.9995
   -0.9988
   -0.9969
   -0.9919
   -0.9787
   -0.9443
   -0.8541
   -0.6180
```

```
Atbilde:
 iterāciju skaits = 5 --> kļūda = 0.33800
 X_app6_norm = 4.849
 sol_X4_norm = 0.99565
 x_tuvinājumi:
   -0.60311  -0.82533  -0.90489  -0.93096  -0.93919
   -0.94102  -0.94147  -0.94147  -0.94147  -0.94147
   -0.94147  -0.94147  -0.94147  -0.94147  -0.94147
   -0.94147  -0.94147  -0.94147  -0.94147  -0.94147
   -0.94147  -0.94147  -0.94147  -0.94147  -0.94102
   -0.93919  -0.93096  -0.90489  -0.82533  -0.60311
```

2.3. uzdevums. Izmantojot Jakobi metodi, atrast sistēmas atrisinājumu pēc 25 iterācijām.

$$\begin{cases} 13x_1 + 10x_2 + 13x_3 = 2 \\ 10x_1 + 8x_2 + 10x_3 = 22 \\ 13x_1 + 10x_2 + 14x_3 = 3 \end{cases}$$

Sākuma tuvinājums ir $x_1^{(0)} = 0$, $x_2^{(0)} = 0$, $x_3^{(0)} = 0$. Interpretēt rezultātus.

```
Matrica A ir nesingulāra.
 Neizpildās konverģences pietiekamais nosacījums
 rindas numurs 1: --> 13 < 23
```

```
Iter. numurs      Kļūda
1            2.7626232110804
2           3.17477031308986
3           6.55936177941138
4           11.0917492491779
5           22.3586404418292
6           41.8490889014946
7           81.6584163595668
8           156.213571927604
9           301.896163275175
10           580.513664937117
11           1119.09927642208
12           2154.64064019475
13           4151.04302533206
14            7994.6830310911
15           15399.7872855811
16           29661.5155674254
17            57133.319744831
18           110046.639985103
19           211967.137349721
20           408279.956392045
21           786409.412416188
22           1514742.40411937
23           2917622.81986522
24           5619780.69627118
25           10824545.2725294
```

```
x_approx =
           3803008.35007559
           4817125.25480351
           3619732.92025536
x_sol =
          -52.0000000000001
           66.5000000000001
                          1
```

```
Atbilde:
 iterāciju skaits = 25
 kļūda (aug) = 10824545.2725
 Jakobi metode diverģē.
```

2.4. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x} = B$. Atrisināt vienādojumu sistēmu, izmantojot Jakobi metodi.

$$A = \begin{pmatrix} 2 & 14 & 10 \\ 14 & 2 & 8 \\ 10 & 8 & 2 \end{pmatrix}; B = \begin{pmatrix} 8 \\ 6 \\ 2 \end{pmatrix}$$

Sākuma tuvinājums ir $x_1^{(0)} = 0$, $x_2^{(0)} = 0$, $x_3^{(0)} = 0$. Izpildīt 13 iterācijas.

- Cik liela ir kļūdas norma $\left\|\varepsilon^{(11)}\right\|_2$ 11. iterācijā, kur $\varepsilon^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)}$?
- Kāda ir atrisinājuma norma $\left\|\boldsymbol{x}^{(10)}\right\|_2$ 10. iterācijā?
- Cik liela ir kļūdas norma $\left\|\varepsilon^{(13)}\right\|_2$ 13. iterācijā?
- Atrisināt šo sistēmu, arī izmantojot komandu **`linsolve`** (ar komandu **`linsolve`** iegūto rezultātu apzīmēsim ar $\boldsymbol{x}\_\mathbf{sol}$).
- Cik liela ir norma $\left\|\boldsymbol{x}\_\mathbf{sol} - \boldsymbol{x}^{(10)}\right\|_2$?

```
Matrica A ir nesingulāra.
 Neizpildās konverģences pietiekamais nosacījums
 rindas numurs 1: --> 2 < 24
```

```
Iter. numurs      Kļūda
1           5.09901951359278
2           52.1919534027996
3           556.884189037541
4           5982.94241991347
5           64380.1025162278
6           693132.466416052
7           7464005.98606405
8           80383439.2585935
9            865721208.55712
10           9323884512.12305
11           100419701310.978
12           1081539454844.26
13           11648403915251.2
```

```
epsi11_norm =
          100419701310.978
X_app10_norm =
          8531779084.76166
epsi13_norm =
          11648403915251.2
sol_X10_norm =
          8531779085.16876
```

```
x_approx =
              6649056973822
              6318847476021
              5428752106807
x_sol =
         0.0104712041884817
         0.0575916230366492
          0.717277486910995
```

```
Atbilde:
 iterāciju skaits = 13 --> kļūda = 11648403915251.20508 (kļūda aug)
 iterāciju skaits = 11 --> kļūda = 100419701310.97789 (kļūda aug)
 X_app10_norm = 8531779084.7617
 sol_X10_norm = 8531779085.1688
 Kļūda aug - Jakobi metode diverģē.
```

2.5. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x}=\boldsymbol{B}$. Atrisināt vienādojumu sistēmu, izmantojot Jakobi metodi.

$$A=\begin{pmatrix}14 & 3 & 8\\ 3 & 12 & 7\\ 8 & 7 & 17\end{pmatrix};\ \boldsymbol{B}=\begin{pmatrix}5\\ 2\\ 6\end{pmatrix}$$

Sākuma tuvinājums ir $x_1^{(0)}=0$, $x_2^{(0)}=0$, $x_3^{(0)}=0$. Izpildīt 9 iterācijas.

- Cik liela ir kļūdas norma $\left\|\boldsymbol{\varepsilon}^{(6)}\right\|_2$ 6. iterācijā, kur $\boldsymbol{\varepsilon}^{(n)}=\boldsymbol{x}^{(n)}-\boldsymbol{x}^{(n-1)}$?
- Kāda ir atrisinājuma norma $\left\|\boldsymbol{x}^{(5)}\right\|_2$ 5. iterācijā?
- Cik liela ir kļūdas norma $\left\|\boldsymbol{\varepsilon}^{(9)}\right\|_2$ 9. iterācijā?
- Atrisināt šo sistēmu, arī izmantojot komandu **`linsolve`** (ar komandu **`linsolve`** iegūto rezultātu apzīmēsim ar $\boldsymbol{x}$_**sol**).
- Cik liela ir norma $\left\|\boldsymbol{x}\_\mathbf{sol}-\boldsymbol{x}^{(7)}\right\|_2$?

```
Matrica A ir nesingulāra.
Izpildās konverģences pietiekamais nosacījums - Jakobi metode konverģē.
```

```
Iter. numurs  Kļūda
    1.0000    0.5291
    2.0000    0.4467
    3.0000    0.3644
    4.0000    0.3096
    5.0000    0.2570
    6.0000    0.2169
    7.0000    0.1810
    8.0000    0.1523
    9.0000    0.1273
```

```
x_approx =
    0.2444
   -0.0103
    0.3055
x_sol =
    0.2114
   -0.0448
    0.2719
```

```
epsi6_norm =
    0.2169
X_app5_norm =
    0.4382
epsi9_norm =
    0.1273
sol_X7_norm =
    0.0834
```

```
Atbilde:
 iterāciju skaits = 6 --> kļūda = 0.21689 (kļūda dilst)
 iterāciju skaits = 9 --> kļūda = 0.12734 (kļūda dilst)
 X_app5_norm = 0.43819
 sol_X7_norm = 0.08335
```

2.6. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x}=\boldsymbol{B}$. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi.

$$A=\begin{pmatrix}9 & 2 & 5\\ 2 & 18 & 6\\ 5 & 6 & 27\end{pmatrix};\ \boldsymbol{B}=\begin{pmatrix}1\\ 6\\ 2\end{pmatrix}$$

Sākuma nosacījums $\boldsymbol{x}^{(0)}=(0\ \ 0\ \ 0)^T$. Izmantot iterāciju parametru $\tau=0.01$. Izpildīt 12 iterācijas.

- Kāda ir tuvinātā atrisinājuma norma $\left\|\boldsymbol{x}^{(12)}\right\|_2$ 12. iterācijā?
- Kāda ir nesaistes norma $\left\|\boldsymbol{B}-A\boldsymbol{x}^{(12)}\right\|_2$ 12. iterācijā?

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
k_iter =
    12
x_app =
    0.0342
    0.2927
    0.0118
x_sol =
    0.0415
    0.3311
   -0.0072
```

```
x_app12_norm =
    0.2949
resid_norm =
    0.6437
```

```
Atbilde:
 iterāciju skaits = 12
 x_tuvinājumi = { 0.0342, 0.2927, 0.0118 }
 tuvinātā atr. norma 12. iter. = 0.2949
 nesaistes norma 12. iter. = 0.6437
```

2.7. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x}=\boldsymbol{B}$. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi.

$$A=\begin{pmatrix}9 & 2 & 5\\ 2 & 18 & 6\\ 5 & 6 & 27\end{pmatrix};\ \boldsymbol{B}=\begin{pmatrix}1\\ 6\\ 2\end{pmatrix}$$

Sākuma nosacījums $\boldsymbol{x}^{(0)}=(0\ \ 0\ \ 0)^T$. Izmantot iterāciju parametru $\tau_{\max}$. Izpildīt 15 iterācijas.

- Kāda ir tuvinātā atrisinājuma norma $\left\|\boldsymbol{x}^{(15)}\right\|_2$ 15. iterācijā?
- Kāda ir nesaistes norma $\left\|\boldsymbol{B}-A\boldsymbol{x}^{(15)}\right\|_2$ 15. iterācijā?

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
tau =
    0.0639
k_iter =
    15
x_app =
    0.0756
    0.3935
    0.1197
x_sol =
    0.0415
    0.3311
   -0.0072
```

```
x_app15_norm =
    0.4182
resid_norm =
    4.5531
```

```
Atbilde:
 iterāciju skaits = 15
 tau_max = 0.063913
 x_tuvinājumi = { 0.0756, 0.3935, 0.1197 }
 tuvinātā atr. norma 15. iter = 0.4182
 nesaistes norma 15. iter = 4.5531
```

2.8. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x}=\boldsymbol{B}$. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi.

$$A=\begin{pmatrix}9 & 2 & 5\\ 2 & 18 & 6\\ 5 & 6 & 27\end{pmatrix};\ \boldsymbol{B}=\begin{pmatrix}1\\ 6\\ 2\end{pmatrix}$$

Sākuma nosacījums $\boldsymbol{x}^{(0)}=(0\ \ 0\ \ 0)^T$. Izmantot iterāciju parametru τ_{opt}. Izpildīt 10 iterācijas.

- Kāda ir tuvinātā atrisinājuma norma $\left\|\boldsymbol{x}^{(10)}\right\|_2$ 10. iterācijā?
- Kāda ir nesaistes norma $\left\|\boldsymbol{B}-A\boldsymbol{x}^{(10)}\right\|_2$ 10. iterācijā?

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
tau =
    0.0513
k_iter =
    10
x_app =
    0.0411
    0.3307
   -0.0080
x_sol =
    0.0415
    0.3311
   -0.0072
```

```
x_app10_norm =
    0.3334
resid_norm =
    0.0303
```

```
Atbilde:
 iterāciju skaits = 10
 tau_opt =  0.051315
 x_tuvinājumi = { 0.0411, 0.3307, -0.0080 }
 tuvinātā atrisinājuma norma 10. iter. = 0.3334
 nesaistes norma 10. iter. = 0.0303
```

2.9. uzdevums. Noteikt matricas A pēc moduļa vislielāko īpašvērtību. Precizitāte $\varepsilon = 0.001$.

$$A = \begin{pmatrix} 1 & 4 & 7 & -1 & 5 \\ 8 & 2 & -3 & 4 & 10 \\ 6 & 12 & -2 & 7 & 4 \\ 3 & -4 & 9 & 0 & 11 \\ 5 & 6 & -4 & 2 & 8 \end{pmatrix}$$

```
k_iter =
     8
lambda =
   17.8345
eig_val_max =
   17.8338
```

```
Atbilde:
  iterāciju skaits = 8
  lielākā īpašvērtība = 17.8345(ar precizitāti 10^(-3))
```

2.10. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot minimālās nesaistes metodi. Precizitāte $\varepsilon = 0.001$.

$$\begin{cases} 16x_1 + 3x_2 + 4x_3 + 2x_4 = 25 \\ 3x_1 + 12x_2 + 2x_3 - x_4 = 16 \\ 4x_1 + 2x_2 + 8x_3 - x_4 = 13 \\ 2x_1 - x_2 - x_3 + 2x_4 = 2 \end{cases}$$

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
k_iter =
    57
tau =
    0.1071
```

```
x_app =
    1.0002
    0.9999
    0.9998
    0.9993
norm_r =
   9.4395e-04
x_sol =
    1.0000
    1.0000
    1.0000
    1.0000
```

```
Atbilde:
 iterāciju skaits = 57, nesaistes norma = 0.00094395
 tau = 0.10714
 x_tuvinājumi = {1.0002   0.99993   0.99983   0.99928}
```

2.11. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot minimālās nesaistes metodi.

Sākuma tuvinājums ir $x_1^{(0)} = 25$, $x_2^{(0)} = -10$, $x_3^{(0)} = 0$. Precizitāte $\varepsilon = 0.0001$.

$$\begin{cases} 7x_1 + 2.5x_2 + 0.5x_3 = 9 \\ 2.5x_1 + 14x_2 + 1.5x_3 = -4 \\ 0.5x_1 + 1.5x_2 + 21x_3 = 22 \end{cases}$$

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
k_iter =
    20
tau =
    0.0565
```

```
x_app =
    1.4447
   -0.6573
    1.0602
norm_r =
   6.3271e-05
```

```
x_sol =
    1.4447
   -0.6573
    1.0602
```

```
Atbilde:
 iterāciju skaits = 20, nesaistes norma = 0.00006327
 tau = 0.056464
 x_tuvinājumi: {1.4447     -0.6573      1.0602}
```

2.12. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot minimālās nesaistes metodi.

$$\begin{cases} 6x_1 + 3x_2 + 5x_3 = -12 \\ 3x_1 + 12x_2 + 6x_3 = 8 \\ 5x_1 + 6x_2 + 18x_3 = 35 \end{cases}$$

Sākuma tuvinājums ir $x_1^{(0)} = -15$, $x_2^{(0)} = 0$, $x_3^{(0)} = 22$. Precizitāte $\varepsilon = 0.001$.

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
k_iter =
    29
tau =
    0.0566
```

```
x_app =
   -4.7896
    0.2720
    3.1843
norm_r =
   7.1213e-04
```

```
x_sol =
   -4.7895
    0.2719
    3.1842
```

```
Atbilde:
 iterāciju skaits = 29, nesaistes norma = 0.00071213
 tau = 0.056601
 x_tuvinājumi: {-4.7896     0.27195      3.1843}
```

2.13. uzdevums. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi.

$$\begin{cases} 16x_1 + 3x_2 + 4x_3 + 2x_4 = 25 \\ 3x_1 + 12x_2 + 2x_3 - x_4 = 16 \\ 4x_1 + 2x_2 + 8x_3 - x_4 = 13 \\ 2x_1 - x_2 - x_3 + 2x_4 = 2 \end{cases}$$

Sākuma tuvinājums ir $\boldsymbol{x}^{(0)} = (0\ \ 0\ \ 0\ \ 0)^T$. Izmantot iterāciju parametru $\tau = 0.02$. Izpildīt 15 iterācijas.

- Kāda ir tuvinātā atrisinājuma norma $\left\|\boldsymbol{x}^{(15)}\right\|_2$ 15. iterācijā?
- Kāda ir nesaistes norma $\left\|\boldsymbol{B} - A\boldsymbol{x}^{(15)}\right\|_2$ 15. iterācijā?
- Par kādu skaitli ir jābūt mazākai iterāciju parametra vērtībai, lai metode konverģētu?
- Kādai ir jābūt iterāciju parametra vērtībai, lai metodes konverģence būtu visātrākā?

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
tau =
    0.0200
k_iter =
    15
```

```
x_app =
    1.1637
    0.9218
    0.8100
    0.3051
```

```
x_sol =
    1.0000
    1.0000
    1.0000
    1.0000
```

```
x_app15_norm =
    1.7184
resid_norm =
    0.9001
```

```
tau_max =
    0.1031
tau_opt =
    0.0971
```

```
Atbilde:
  x_tuvinājumi 15. iter. = {1.1637   0.9218   0.8100     0.3051}
  tuvinātā atrisinājuma norma 15. iter. = 1.7184
  nesaistes norma 15. iter. = 0.9001
  tau_max = 0.10306, optimālā parametra tau vērtība = 0.09707
```

2.14. uzdevums. Dota vienādojumu sistēma $A\boldsymbol{x}=\boldsymbol{B}$. Atrisināt vienādojumu sistēmu, izmantojot vienkāršo iterāciju metodi ar precizitāti $\varepsilon=10^{-3}$.

$$A=\begin{pmatrix} 8 & 2 & 4 & -2 \\ 2 & 16 & 12 & 10 \\ 4 & 12 & 24 & 2 \\ -2 & 10 & 2 & 32 \end{pmatrix};\ \boldsymbol{B}=\begin{pmatrix} 9 \\ 7 \\ 6 \\ 9 \end{pmatrix}$$

Sākuma tuvinājums ir $\boldsymbol{x}^{(0)}=(0\ 0\ 0\ 0)^T$. Izmantot iterāciju parametru $\tau=0.06$. Izpildīt 11 iterācijas.

- Kāda ir tuvinātā atrisinājuma norma $\left\|\boldsymbol{x}^{(11)}\right\|_2$ 11. iterācijā?
- Kāda ir nesaistes norma $\left\|\boldsymbol{B}-A\boldsymbol{x}^{(11)}\right\|_2$ 11. iterācijā?
- Par kādu skaitli ir jābūt mazākai iterāciju parametra vērtībai, lai metode konverģētu?
- Kādai ir jābūt iterāciju parametra vērtībai, lai metodes konverģence būtu visātrākā?

```
Matrica A ir nesingulāra.
Koeficientu matrica ir simetriska un pozitīvi definēta.
```

```
tau =
    0.0600
k_iter =
    11
```

```
x_app =
    2.0117
    9.2665
    8.2799
   12.7115
```

```
x_sol =
    1.1940
    0.1085
   -0.0302
    0.3238
```

```
x_app11_norm =
   17.8900
resid_norm =
  711.4217
```

```
tau_max =
    0.0493
tau_opt =
    0.0435
```

```
Atbilde:
  x_tuvinājumi 11. iter. = {2.0117   9.2665   8.2799   12.7115}
  tuvinātā atrisinājuma norma 11. iter. = 17.89
  nesaistes norma 11. iter. = 711.4217
  tau_max = 0.04926, optimālā parametra tau vērtība = 0.04351
```

3. nodaļa

INTERPOLĀCIJA

3.1. Interpolācijas mezgli. Polinomiālā interpolācija

Pieņemsim, ka dota funkcijas $y=y(x)$ tabula (funkcijas vērtības ir aprēķinātas punktos x_1, x_2, …, x_n.

x_i	y_i
x_1	y_1
x_2	y_2
x_2	y_2
...	...
x_n	y_n

Atzīmēsim, ka funkcija $y=y(x)$ nav noteikti jādod ar formulu. Mūsu mērķis ir tuvināti aprēķināt funkcijas vērtības citos punktos (kas atrodas starp punktiem x_1, x_2, …, x_n) intervālā $[a, b]$.

Šo procedūru (funkcijas y vērtību aprēķināšanu, pieņemot, ka funkcijas vērtības punktos x_1, x_2, …, x_n ir zināmas) sauc par **interpolāciju**.

Punktus x_1, x_2, …, x_n sauc par **interpolācijas mezgliem**.

Praktiski mēs funkcijas $y=y(x)$ vietā izmantosim citu funkciju $f(x)$, kuras vērtības ir (zināmā mērā) tuvas funkcijas $y(x)$ vērtībām. Tad var pieņemt, ka $y(x)\approx f(x)$ visiem $x\in[a,b]$.

Parasti funkcija $f(x)$ satur brīvos parametrus a_1, a_2, …, a_m, tā kā var rakstīt $f(x, a_1, a_2, …, a_m)$.

Interpolācijas uzdevumu formulē šādi: atrast funkciju $f(x, a_1, a_2, …, a_n)$, kas apmierina nosacījumu

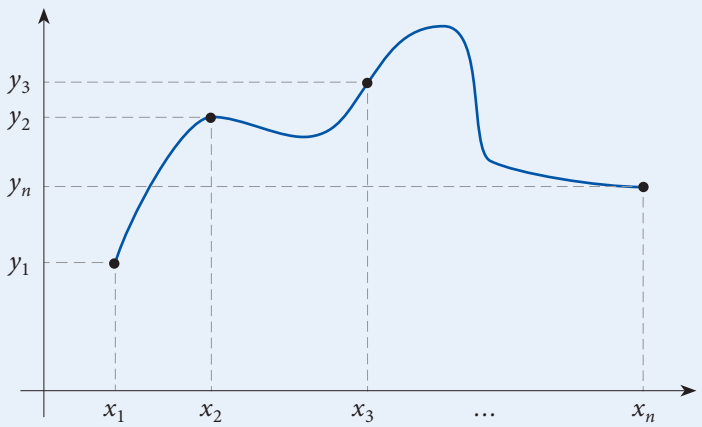
$$f(x_i,a_1,a_2,\ldots,a_n)=y_i,\quad i=1,2,\ldots,n \tag{3.1}$$

Atzīmēsim, ka interpolācijas uzdevumā nezināmo parametru a_1, a_2, …, a_n skaits ir vienāds ar interpolācijas mezglu skaitu x_1, x_2, …, x_n.

Interpolācijas uzdevuma (3.1) ģeometriskā interpretācija ir šāda: funkcijas $f(x, a_1, a_2, …, a_n)$ grafiks iet caur visiem punktiem (x_i, y_i), $i=1, 2, …, n$ (sk. 3.1. attēlu).

Vienādojumu sistēma (3.1) vispārīgā gadījumā ir nelineārā attiecībā pret parametriem a_1, a_2, …, a_n. Lai vienkāršotu aprēķinus, bieži pieņem, ka funkcija $f(x, a_1, a_2, …, a_n)$ ir lineāra attiecībā pret parametriem a_1, a_2, …, a_n.

Tādējādi lineārās interpolācijas uzdevumā



3.1. att. Interpolācijas uzdevums ($y=f(x)$; $y_i=f(x_i)$, $i=1, 2, …, n$).

$$f\left(x,a_1,a_2,\ldots,a_n\right)=a_1\varphi_1\left(x\right)+a_2\varphi_2\left(x\right)+\ldots+a_n\varphi_n\left(x\right) \tag{3.2}$$

kur $\varphi_1(x)$, $\varphi_2(x)$, …, $\varphi_n(x)$ ir lineāri neatkarīgas funkcijas intervālā $[a, b]$.

Izmantojot (3.1) un (3.2), iegūstam

$$a_1\varphi_1\left(x_i\right)+a_2\varphi_2\left(x_i\right)+\ldots+a_n\varphi_n\left(x_i\right)=y\left(x_i\right),\quad i=1,2,\ldots,n \tag{3.3}$$

Sistēma (3.3) ir lineāra vienādojumu sistēma ar nezināmajiem a_1, a_2, …, a_n. Sistēmai ir viens vienīgs atrisinājums tad un tikai tad, kad koeficientu matricas determinants nav vienāds ar nulli:

$$\det(A)=\begin{vmatrix}\varphi_1(x_1) & \varphi_2(x_1) & \varphi_3(x_1) & \varphi_4(x_1) & \ldots & \varphi_n(x_1)\\ \varphi_1(x_2) & \varphi_2(x_2) & \varphi_3(x_2) & \varphi_4(x_2) & \ldots & \varphi_n(x_2)\\ \varphi_1(x_3) & \varphi_2(x_3) & \varphi_3(x_3) & \varphi_4(x_3) & \ldots & \varphi_n(x_3)\\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\ \varphi_1(x_n) & \varphi_2(x_n) & \varphi_3(x_n) & \varphi_4(x_n) & & \varphi_n(x_n)\end{vmatrix}\neq 0 \tag{3.4}$$

Viena no populārākajām interpolācijas metodēm ir **polinomiālā interpolācija**. Šajā gadījumā funkcijas $\varphi_1(x)$, $\varphi_2(x)$, …, $\varphi_n(x)$ ir polinomi: $\varphi_1(x)=1$, $\varphi_2(x)=x$, …, $\varphi_n(x)=x^{n-1}$. Lineāro sistēmu (3.3) var pārrakstīt šādi:

$$a_1+a_2x_i+\ldots+a_nx_i^{n-1}=y\left(x_i\right),\quad i=1,2,\ldots,n \tag{3.5}$$

Nosacījumu (3.4) pieraksta šādi:

$$\det\left(A\right)=\begin{vmatrix}1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^{n-1}\\ 1 & x_2 & x_2^2 & x_2^3 & \ldots & x_2^{n-1}\\ 1 & x_3 & x_3^2 & x_3^3 & \ldots & x_3^{n-1}\\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\ 1 & x_n & x_n^2 & x_n^3 & \ldots & x_n^{n-1}\end{vmatrix}\neq 0 \tag{3.6}$$

Determinantu (3.6) sauc par **Vandermonda determinantu**.

Var pierādīt, ka Vandermonda determinants atšķiras no nulles, ja visi interpolācijas mezgli ir atšķirīgi.

Tādējādi interpolācijas shēmu var aprakstīt šādi:

1. solis. Definē interpolācijas mezglus x_1, x_2, …, x_n un funkcijas vērtības interpolācijas mezglos $y=y(x)$: y_1, y_2, …, y_n;

2. solis. Izvēlas interpolējošo funkciju pēc formulas (3.2);

3. solis. Atrisina lineāru vienādojumu sistēmu (3.3) un atrod a_1, a_2, …, a_n;

4. solis. Izmanto tuvināto formulu $y(x)\approx f(x, a_1, a_2, \ldots, a_n)$ visiem $x\in\left[a,b\right]$.

Ir viens svarīgs gadījums, kas sistēmu (3.3) nerisina, – tas ir polinomiālās interpolācijas gadījums.

3.2. Ņūtona interpolācijas polinoms

Pieņemsim, ka intervālā $a \le x \le b$ ir doti $n+1$ interpolācijas mezgli $x_0, x_1, x_2, \ldots, x_n$. Funkcijas vērtības interpolācijas mezglos arī ir zināmas: $y_0, y_1, y_2, \ldots, y_n$.

Formulēsim uzdevumu: atrast n-tās pakāpes polinomu $p_n(x)$, kas apmierina nosacījumus

$$p_n(x_0) = y_0,\ p_n(x_1) = y_1, \ldots,\ p_n(x_n) = y_n \tag{3.7}$$

Lai atrastu polinoma $p_n(x)$ koeficientus, sāksim ar visvienkāršāko gadījumu, kad tabulā ir tikai divi punkti: (x_0, y_0) un (x_1, y_1). Konstruēsim pirmās pakāpes polinomu $p_1(x)$, kas apmierina nosacījumus

$$p_1(x_0) = y_0,\ p_1(x_1) = y_1$$

Polinoma grafiks ir taisne, kas iet caur diviem punktiem (sk. 3.2. attēlu).

Polinomu $p_1(x)$ var uzrakstīt šādi

$$p_1(x) = y_0 + (x - x_0)\frac{(y_1 - y_0)}{(x_1 - x_0)}$$

Pārrakstīsim formulu šādi:

$$p_1(x) = y_0 + (x - x_0)\, y[x_0, x_1], \tag{3.8}$$

kur $y[x_0, x_1]$ ir pirmā izdalītā starpība:

$$y[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}$$

Aplūkosim gadījumu, kad ir doti trīs punkti: (x_0, y_0), (x_1, y_1) un (x_2, y_2). Otrās pakāpes interpolācijas polinomu var uzrakstīt šādi:

$$p_2(x) = y_0 + (x - x_0)\, y[x_0, x_1] + (x - x_0)(x - x_1)\, y[x_0, x_1, x_2] \tag{3.9}$$

kur $y[x_0, x_1, x_2]$ ir otrā izdalītā starpība

$$y[x_0, x_1, x_2] = \frac{y[x_1, x_2] - y[x_0, x_1]}{x_2 - x_0}$$

Kā var konstruēt interpolācijas polinomu $p_n(x)$ jebkuram naturālam skaitlim n?

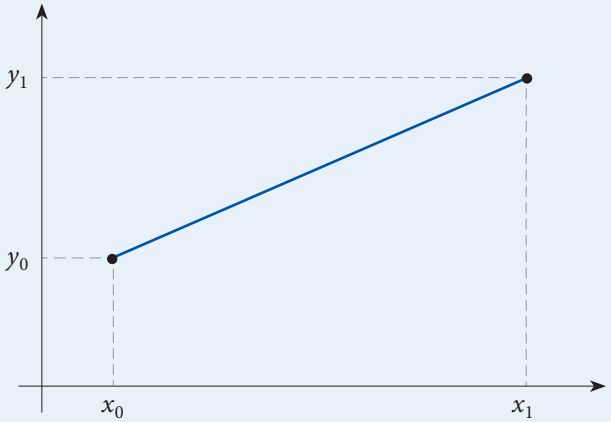
Izmantosim matemātiskās indukcijas principu. Pieņemsim, ka $p_{n-1}(x)$ ir $n-1$ kārtas interpolācijas polinoms.

Tādējādi

$$p_{n-1}(x_0) = y_0, \quad p_{n-1}(x_1) = y_1, \ldots, p_{n-1}(x_{n-1}) = y_{n-1}$$

Polinomu $p_n(x)$ uzrakstīsim veidā

$$p_n(x) = p_{n-1}(x) + g_n(x)$$



3.2. att. Lineārā interpolācija.

Tā kā $p_n(x)$ ir n-tās pakāpes interpolācijas polinoms, ir spēkā nosacījumi

$$p_n(x_0)=y_0,\ p_n(x_1)=y_1,\ \ldots,\ p_n(x_{n-1})=y_{n-1},\ p_n(x_n)=y_n$$

Polinomu $p_{n-1}(x)$ un $p_n(x)$ vērtības punktos $x_0, x_1, x_2, \ldots, x_{n-1}$ sakrīt.

Tas nozīmē, ka $g_n(x_i)=0$, $i=0, 1, \ldots, n-1$. Polinoms

$$g_n(x)=p_n(x)-p_{n-1}(x) \tag{3.10}$$

ir n-tās pakāpes polinoms.

Tādējādi

$$g_n(x)=a_n(x-x_0)(x-x_1)\ldots(x-x_{n-1}) \tag{3.11}$$

Lai aprēķinātu koeficientu a_n, izmantosim (3.10) un (3.11) gadījumā, kad $x=x_n$:

$$a_n=\frac{f_n-p_{n-1}(x_n)}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})}$$

kur $p_n(x_n)=f_n$.

Ja $n=1$, iegūstam

$$a_1=\frac{f_1-p_0(x_1)}{x_1-x_0}=\frac{f_1-f_0}{x_1-x_0}=f[x_0,x_1]$$

Ja $n=2$, iegūstam

$$a_2=\frac{f_2-p_1(x_2)}{(x_2-x_0)(x_1-x_0)}=\frac{f_2-f_0-(x_2-x_0)f[x_0,x_1]}{(x_2-x_0)(x_1-x_0)}=f[x_0,x_1,x_2]$$

Vispārīgā gadījumā

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

kur $f[x_0, x_1, \ldots, x_k]$ ir izdalītā starpība ar kārtu k.

Tādējādi

$$\begin{aligned}p_n(x)=f_0+(x-x_0)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_1,x_2]+\ldots\\ +(x-x_0)(x-x_1)\ldots(x-x_n)f[x_0,x_1,\ldots,x_n]\end{aligned} \tag{3.12}$$

Polinomu $p_n(x)$ formulā (3.12) sauc par **Ņūtona interpolācijas polinomu**.

Atzīmēsim, ka interpolācijas polinoma pakāpe ir saistīta ar interpolācijas mezglu skaitu. Ja interpolācijas mezglu skaits ir n, eksistē viens vienīgs Ņūtona interpolācijas polinoms ar kārtu $n-1$.

3.1. piemērs. Interpolēt $e^{0.5x}$ ar Ņūtona interpolācijas polinomu. Interpolācijas mezgli ir $x_0=0$, $x_1=2$, $x_2=4$ un $x_3=5$. Izmantot iegūto polinomu, lai tuvināti aprēķinātu $e^{0.5}$.

Atrisinājums.

Pirmkārt, noskaidrosim interpolācijas polinoma kārtu. Tā kā ir doti četri interpolācijas mezgli, uzdevumu interpolēsim ar trešās kārtas Ņūtona interpolācijas polinomu.

Otrkārt, aprēķināsim funkcijas $e^{0.5x}$ vērtības punktos x_0, x_1, x_2 un x_3:

$$f_0=e^0=1,\ f_1=e^1=2.718282,\ f_2=e^2=7.389056,\ f_3=e^{2.5}=12.182494$$

Treškārt, aprēķināsim izdalītās starpības, izmantojot 3.1. tabulu.

3.1. tabula. Izdalīto starpību aprēķināšana.

x_j	$f(x_j)$	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	$f[x_j, x_{j+1}, x_{j+2}, x_{j+3}]$
0.0	1.000000			
		0.859141		
2.0	2.718282		0.3690615	
		2.335387		0.0900577
4.0	7.389056		0.8193500	
		4.793438		
5.0	12.182494			

Skaitļi 3.1. tabulā ir aprēķināti pēc formulām:

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0} = \frac{2.718282 - 1.000000}{2.0 - 0.0} = 0.859141$$

$$f[x_1, x_2] = \frac{f_2 - f_1}{x_2 - x_1} = \frac{7.389056 - 2.718282}{4.0 - 2.0} = 2.335387$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{2.335387 - 0.859141}{4.0 - 0.0} = 0.3690615$$

Ņūtona interpolācijas polinoms ir

$$p_3(x) = 1.0 + 0.859141x + 0.3690615x(x-2) + 0.0900577x(x-2)(x-4)$$

Funkcijas $e^{0.5x}$ tuvinātā vērtība punktā $x=1$ ir

$$p_3(1) = 1.0 + 0.859141 - 0.3690615 + 0.2701731 = 1.7602526$$

Precīzā vērtība ir $e^{0.5} = 1.648721$. Relatīvā kļūda ir 6.76 %.

3.3. Lagranža interpolācijas polinoms

Interpolācijas polinomu var pierakstīt arī citā veidā. To sauc par Lagranža interpolācijas polinomu.

Lai saprastu Lagranža interpolācijas pamatideju, aplūkosim visvienkāršāko gadījumu – interpolāciju starp diviem dotajiem punktiem (x_0, f_0) un (x_1, f_1). Pirmās kārtas interpolācijas polinomu $p_1(x)$ pierakstīsim šādi:

$$p_1(x) = L_0(x) f_0 + L_1(x) f_1, \tag{3.13}$$

kur

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Tādējādi

$$p_1(x) = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1$$

Polinomus $L_0(x)$ un $L_1(x)$ sauc par **fundamentālajiem interpolācijas polinomiem**.

Polinomu $L_0(x)$ un $L_1(x)$ pakāpe ir tāda pati kā polinomam $p_1(x)$ (pirmās pakāpes polinomi), turklāt katrs polinoms vienā interpolācijas mezglā pieņem vērtību 0 un vērtību 1 – otrā interpolācijas mezglā.

Var viegli pārbaudīt, ka $L_0(x_0) = 1$ un $L_0(x_1) = 0$, bet $L_1(x_0) = 0$ un $L_1(x_1) = 1$.

Līdzīgi otrās kārtas interpolācijas polinoma grafiks iet cauri trim punktiem (x_0, f_0), (x_1, f_1) un (x_2, f_2).

To var pierakstīt šādi:

$$p_2(x) = L_0(x) f_0 + L_1(x) f_1 + L_2(x) f_2,$$

kur fundamentālie interpolācijas polinomi $L_0(x)$, $L_1(x)$ un $L_2(x)$ ir

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}; \quad L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_0 - x_2)}; \quad L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Ir redzams, ka

$$L_0(x_0) = 1; L_0(x_1) = 0; L_0(x_2) = 0$$
$$L_1(x_0) = 0; L_1(x_1) = 1; L_1(x_2) = 0$$
$$L_2(x_0) = 0; L_2(x_1) = 0; L_2(x_2) = 1$$

Tādējādi katrs no polinomiem $L_m(x)$, $m = 0, 1, 2$ ir vienāds ar nulli divos interpolācijas mezglos un ir vienāds ar 1 trešajā punktā.

Otrās kārtas Lagranža interpolācijas polinoms ir

$$p_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f_2$$

Analoģiski n-tās kārtas Lagranža interpolācijas polinomu var pierakstīt šādi:

$$p_n(x) = \sum_{i=0}^{n} f_i \prod_{j=0, j \neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

3.2. piemērs. Izmantojot vērtības $\ln 9.0 = 2.1972$, $\ln 9.5 = 2.2513$, konstruēt Lagranža interpolācijas polinomu ar kārtu 1 funkcijai $y = \ln x$. Tuvināti aprēķināt $\ln 9.2$ un atrast interpolācijas kļūdu (pieņemt, ka $\ln 9.2 = 2.2192$).

Atrisinājums.

Definēsim vērtības $x_0 = 9.0$, $x_1 = 9.5$, $f_0 = 2.1972$, $f_1 = 2.2513$ un $x = 9.2$.

Fundamentālo interpolācijas polinomu vērtības punktā x ir

$$L_0(x) = \frac{9.2 - 9.5}{9.0 - 9.5} = 0.6; \quad L_1(x) = \frac{9.2 - 9.0}{9.5 - 9.0} = 0.4$$

Izmantojot formulu (3.13), iegūstam

$$\ln 9.2 = p_1(9.2) = 0.6 \times 2.1972 + 0.4 \times 2.2513 = 2.2188$$

Relatīvā kļūda ir 1.8 %.

3.4. Interpolācijas aprēķini *MATLAB* vidē

`sym2poly(pol)`	Noteikt polinoma koeficientus (pol – polinoms) *double* formātā
`collect`	Sagrupēt saskaitāmos pēc argumenta pakāpēm
`expand(S)`	Uzrakstīt simbolisko izteiksmi **S** izvērstā veidā

3.3. piemērs. Interpolēt tabulu, izmantojot Ņūtona interpolācijas polinomu.

x_i	y_i
1	2.3
2	3.4
3	5.1
4	6.3
5	7.5
6	8.2
7	7.4

Atrisinājums.

```
%% 3.3. piemērs. Ņūtona interpolācijas polinoms
clc, clearvars, format compact
xnodes = (1:7);  ynodes = [2.3,3.4,5.1,6.3,7.5,8.2,7.4];
coef = ynodes; % koeficientu  noteikšana
for k = 2:7
   coef(k:7) = (coef(k:7) - coef(k-1:6))./…
               (xnodes (k:7) - xnodes (1:8-k));
end
```

```
m = length(xnodes);
```

Atzīmēsim, ka skaitlis 7 (iezīmēts sarkanā krāsā) nosaka interpolācijas mezglu skaitu. Būtu jāievieš parametrs (piemēram, m) pirms cikla, lai definētu interpolācijas mezglu skaitu. Jāmaina arī citi operatori skriptā, kur 7 ir jāaizstāj ar m, 6 – ar $m-1$ utt. Pēc labojumiem skriptu var izmantot uzdevumiem ar citu interpolācijas mezglu skaitu (vienīgais parametrs, kas būs jāmaina, ir m).

x_j	f_j	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	$f[x_j, x_{j+1}, x_{j+2}, x_{j+3}]$	…	$f[x_j, x_{j+1}, x_{j+2}, x_{j+3}, x_{j+4}, x_{j+5}, x_{j+6}]$
x_1	$a_1 = f_1$					
		$a_2 = f[x_1, x_2]$				
x_2	f_2		$a_3 = f[x_1, x_2, x_3]$			
		$f[x_2, x_3]$		$a_4 = f[x_1, x_2, x_3, x_4]$		
x_3	f_3		$f[x_2, x_3, x_4]$		…	
		$f[x_3, x_4]$		$f[x_2, x_3, x_4, x_5]$		
x_4	f_4		$f[x_3, x_4, x_5]$		…	$a_7 = f[x_1, x_2, x_3, x_4, x_5, x_6, x_7]$
		$f[x_4, x_5]$		$f[x_3, x_4, x_5, x_6]$		
x_5	f_5		$f[x_4, x_5, x_6]$		…	
		$f[x_5, x_6]$		$f[x_4, x_5, x_6, x_7]$		
x_6	f_6		$f[x_5, x_6, x_7]$			
		$f[x_6, x_7]$				
x_7	f_7					

$$f[x_1,x_2]=\frac{f_2-f_1}{x_2-x_1}$$
$$f[x_1,x_2,x_3]=\frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$$
...

Koeficientu a_1, a_2, a_3, ..., a_7 aprēķini tabulā ir parādīti ciklā (sk. ciklu **for**).

Tie ir izdalītās starpības (sk. informāciju sarkanajā taisntūrī).

```
% 3.3. piemēra turpinājums
syms x
pol = coef(7); % polinoma konstruēšana
for k = 6:-1:1
    pol = pol*(x-xnodes(k))+coef(k);
end
```

$$pol=a_7(x-x_6)(x-x_5)(x-x_4)\dots(x-x_1)+a_6(x-x_5)(x-x_4)\dots(x-x_1)+\dots$$
$$+a_3(x-x_2)(x-x_1)+a_2(x-x_1)+a_1 \qquad P_6(x)$$

Mainīgais **pol** ir sestās kārtas Ņūtona interpolācijas polinoms.

```
% 3.3. piemēra turpinājums
polyn(x) = collect(pol) % vai → polyn(x)= expand(pol)
coefpol = sym2poly(polyn) % polinoma koeficienti (formātā double)
```

```
polyn(x) =
(31*x^6)/7200 - (269*x^5)/2400 + (1649*x^4)/1440 -(937*x^3)/160 +
(28081*x^2)/1800 - (11209*x)/600 + 51/5
coefpol =
    0.0043   -0.1121    1.1451   -5.8563   15.6006  -18.6817   10.2000
```

Aprēķinu rezultāti rāda starpību starp polinoma pierakstu formātā *double* un simbolisko pierakstu. Tabulā ir septiņi punkti, un tas nozīmē, ka interpolācijas polinoms ir sestās pakāpes.

Pirmais koeficients (0.0043) ir koeficients pie x^6, skaitlis –0.1121 ir koeficients pie x^5 utt.

```
% 3.3. piemēra turpinājums
x_pr = 1:0.01:7;
plot(x_pr,polyn(x_pr),'r-',xnodes,ynodes,'ob','LineWidth',3)
title('Ņūtona interpolācijas polinoms')
legend('polinoms','mezgli'), grid on
fun_prob3(coefpol) % polinoma drukāšana
```

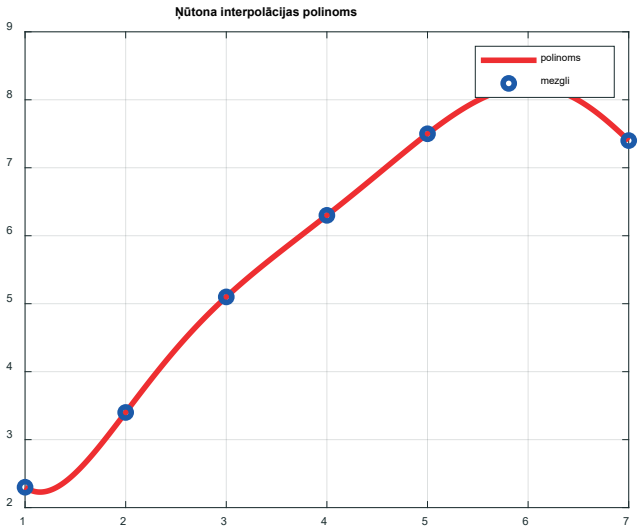
```
Atbilde.
  Ņutona 6. kārtas interpolācijas polinoms:
   +0.0043x^6 -0.1121x^5 +1.1451x^4 -5.8563x^3 +15.6006x^2 -18.6817x^1
   +10.2000x^0
```

Lai izdrukātu polinomu, izmanto ārējo funkciju **fun _ prob3**.

```
% ārēja funkcija (3.3. piemērs). Interpolācija
% polinoma drukāšana
function fun_prob3(koef)
   m = length(koef);
```

```
    fprintf('\n Atbilde. \n  Ņūtona %.0f. kārtas',m-1)
    fprintf(' interpolācijas polinoms: \n  ')
    n = m-1;
    for i = 1:m
       if koef(i) < 0
         fprintf(' %.4fx^%.0f',koef(i),n)
       else
          fprintf(' +')
          fprintf('%.4fx^%.0f',koef(i),n)
       end
       n = n-1;
    end
    fprintf('\n')
end
```

Atzīmēsim, ka interpolācijas polinoma grafiks iet caur visiem punktiem tabulā (interpolācijas kļūda visos interpolācijas mezglos ir nulle).



3.3. att. Interpolācijas polinoma grafiks kopā ar konkrētās funkcijas tabulveida grafiku.

3.4. piemērs. Interpolēt funkciju $\ln(1+x^2)$, izmantojot Ņūtona interpolācijas polinomu. Interpolācijas mezgli ir $x_0=0$, $x_1=1$, $x_2=2$, $x_3=3$. Uzzīmēt interpolācijas kļūdas grafiku.

Atrisinājums. Interpolācijas polinoms ir ar kārtu trīs (doti četri interpolācijas mezgli). Aprēķināsim dotās funkcijas vērtības interpolācijas mezglos.

```
%% 3.4. piemērs. Ņūtona interpolācijas polinoms
clc, clearvars, format compact, close all
syms x
y = @(x)log(1+x.^2);
xnodes = (0:3);  ynodes = y(xnodes);
…   …   …   …
```

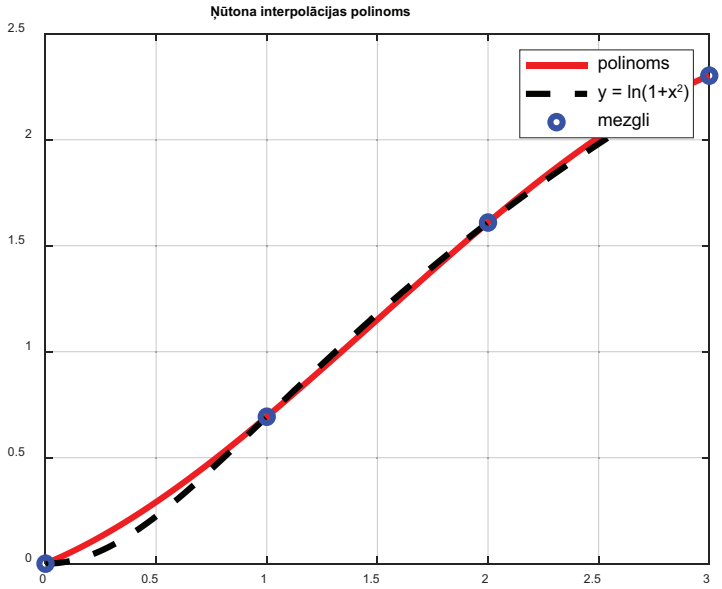
Punkti pēc operatora **`ynodes = y(xnodes)`** nosaka, ka aprēķinu modulis (Ņūtona interpolācijas polinoma konstruēšana) paliek tāds pats kā 3.3. piemērā (pie nosacījuma, ka ir definēts parametrs m, kas ir vienāds ar interpolācijas mezglu skaitu).

```
polyn(x) =
-    (1339932286065071*x^3)/18014398509481984+(6029695287292821
*x^2)/18014398509481984 + (974608316887871*x)/2251799813685248
coefpol =
   -0.0744    0.3347    0.4328         0
```

```
% 3.4. piemēra turpinājums
x_pr = xnodes(1):0.01:xnodes(m);
plot(x_pr,polyn(x_pr),'r-',x_pr,y(x_pr),'k--',xnodes,ynodes,'ob',...
    'LineWidth',3)
title('Ņūtona interpolācijas polinoms')
legend('polinoms','y = ln(1+x^2)','mezgli') , grid on
fun_prob3(coefpol)
```

```
Atbilde.
  Ņutona 3. kārtas interpolācijas polinoms:
   -0.0744x^3 +0.3347x^2 +0.4328x^1 +0.0000x^0
```

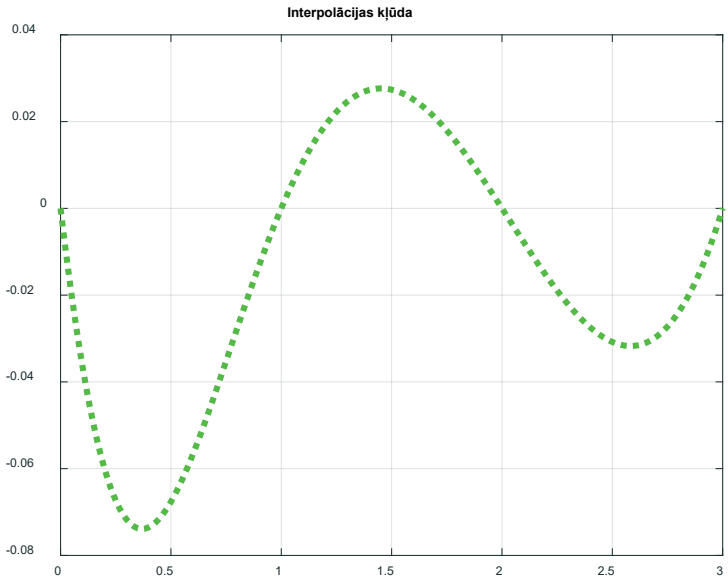
```
% 3.4. piemēra turpinājums
figure % interpolācijas kļūdas grafiks
plot(x_pr,y(x_pr)-polyn(x_pr),'g:','LineWidth',3)
title('Interpolācijas kļūda'), grid on
```



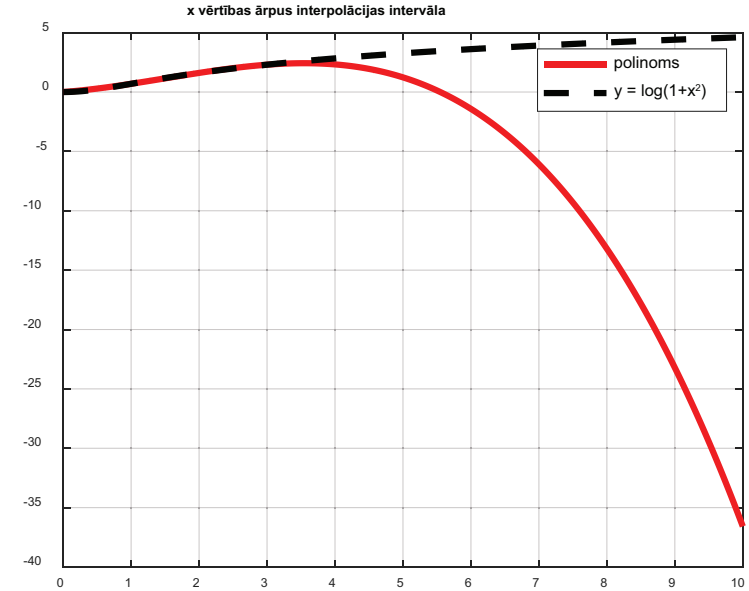
3.4. att. Interpolācijas polinoma grafiks un dotās funkcijas grafiks.

```
% 3.4. piemēra turpinājums
figure % x vērtības ārpus intervāla, kur ir doti interpolācijas mezgli
x1 = 0:0.01:10; plot(x1,polyn(x1),'r-',x1,y(x1),'k--','LineWidth',3)
title('x vērtības ārpus interpolācijas intervāla')
legend('polinoms','y = log(1+x^2)'), grid on
```

3.6. attēlā redzams, kas var notikt ekstrapolācijas rezultātā (interpolācijas polinoma grafiks ir parādīts intervālā [0,10], bet interpolācijas mezgli ir doti intervālā [0,3]). Interpolācijas kļūda var būtiski palielināties, ja interpolācijas polinomu izmanto ārpus intervāla, kur ir doti interpolācijas mezgli.



3.5. att. Interpolācijas kļūdas grafiks.



3.6. att. Interpolācijas polinoma un funkcijas grafiki intervālā [0,10].

3.5. piemērs. Konstruēt funkcijas $f(x)=\sqrt{2+x^7}$ tabulu intervālā [1,5] ar soli $\Delta x=1$. Interpolēt tabulu ar Ņūtona interpolācijas polinomu (apzīmēsim interpolācijas polinomu ar $\tilde{f}(x)$).

- Atrast interpolācijas polinoma koeficientu pie x^2.
- Aprēķināt interpolācijas polinoma vērtību punktā $x_0=2.5$.
- Aprēķināt interpolācijas kļūdu $\left|\tilde{f}(x_0)-f(x_0)\right|$.
- Atrast interpolācijas kļūdu punktā $x=3$.

Atrisinājums.

```
%% 3.5. piemērs. Ņūtona interpolācijas polinoms.
clc, clearvars, format compact, close all
syms x, f = @(x)sqrt(2+x.^7);
xnodes = (1:5); ynodes = f(xnodes);  coef = ynodes;
for k = 2:5
    coef(k:5) = (coef(k:5) - coef(k-1:4))./...
                (xnodes(k:5) - xnodes(1:6-k));
end
pol= coef(5); % polinoma konstruēšana
for k = 4:-1:1
   pol = pol*(x-xnodes(k))+coef(k);
end
```

```
m = length(xnodes);
```

```
polyn(x) = collect(pol);          % simboliskā funkcija
coefpol = sym2poly(polyn)         % polinoma koeficienti
int_res = double(polyn(2.5))      % polinoma vērtība punktā x0=2.5
f_res = f(2.5)                    % funkcijas vērtība punktā x0=2.5
int_error = abs(int_res-f_res) % interpolācijas kļūda punktā x0=2.5
```

```
coefpol =
    0.1803    1.5508   -0.9539   -1.0284    1.9832
int_res =
   24.7237
f_res =
   24.7457
int_error =
    0.0220
```

```
% 3.5. piemēra turpinājums
disp('Atbilde:')
disp(['1) koef_x^2 = ', num2str(coefpol(3))])
disp(['2) interpolācijas rezultāts punktā (2.5) = ', num2str(int_res)])
disp(['3) interpolācijas kļūda punktā (2.5) = ', num2str(int_error)])
disp('4) interpolācijas kļūda punktā (x=3) = 0')
disp('tā kā punkts x=3 ir interpolācijas mezgla punkts')
```

```
Atbilde:
1) koef_x^2 = -0.95385
2) interpolācijas rezultāts punktā (2.5) = 24.7237
3) interpolācijas kļūda punktā (2.5) = 0.022047
4) interpolācijas kļūda punktā (x=3) = 0
tā kā punkts x=3 ir interpolācijas mezgla punkts
```

Atzīmēsim, ka 3.5. piemērā redzamajā skriptā ir izmantotas konstanšu absolūtās vērtības (piemēram, skaitlis 5 nosaka interpolācijas punktu skaitu).

Ir ieteicams modificēt skriptu tā, lai būtu vieglāk mainīt interpolācijas mezglu skaitu (sk. komentārus pie 3.3. piemēra atrisinājuma).

MATLAB vidē interpolācijas uzdevumu var atrisināt, izmantojot komandu **`interp1`**.

`v = interp1(x,y,xs)`	**Lineārā interpolācija** • **`x`** (vektors) satur interpolācijas mezglus • **`y`** (vektors) satur funkcijas vērtības interpolācijas mezglos • **`xs`** (vektors) satur punktus, kuros konkrētā funkcija ir jāinterpolē
`v = interp1(x,y,xs,metode)`	**Interpolācijas metodes** • **`'spline'`** splaina interpolācija • **`'pchip'`** interpolācija ar trešās pakāpes Ermīta polinomiem katrā apakšintervālā • **`'cubic'`** interpolācija ar trešās kārtas polinomiem

3.6. piemērs. Interpolēt doto funkciju tabulveidā. Interpolācijas mezgli ir

$$x = \left(0, \frac{\pi}{8}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}\right).$$

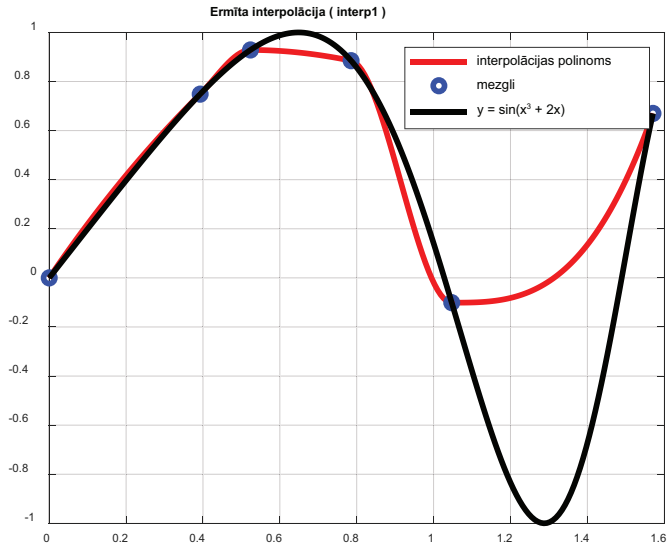
Funkcijas vērtības interpolācijas mezglos ir $y = (0, 0.7486, 0.9286, 0.8849, -0.1010, 0.67)$. Izmantot komandu **`interp1`** un Ermīta interpolāciju. Uzzīmēt interpolācijas kļūdas grafiku.

Atrisinājums.

Lai ilustrētu problēmas ar interpolācijas metodi, ir izvēlēta funkcija $y = \sin(x^3 + 2x)$. Funkcijas vērtības interpolācijas mezglos ($y = (0, 0.7486, 0.9286, 0.8849, -0.1010, 0.67)$) 3.6. piemērā ir funkcijas $y = \sin(x^3 + 2x)$ vērtības punktos

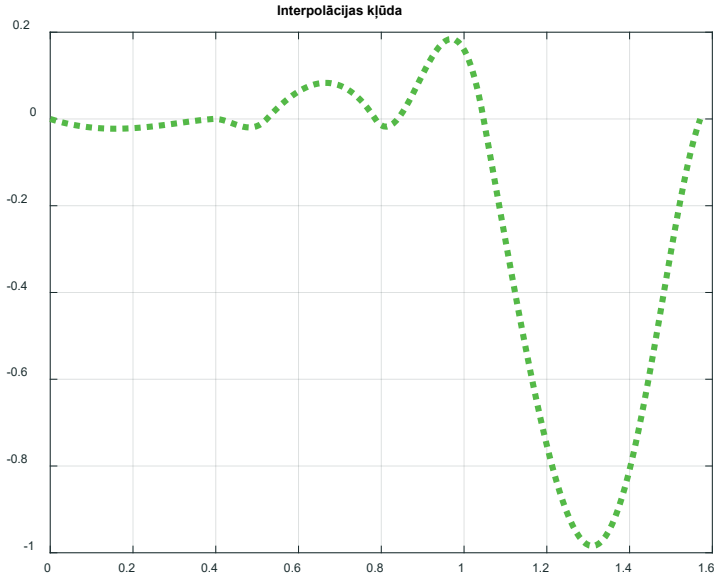
$$x = \left(0, \frac{\pi}{8}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}\right).$$

```
%% 3.6. piemērs. Ermīta interpolācija (interp1)
clc, clearvars, format compact, close all
xnodes = [0 pi/8 pi/6 pi/4 pi/3 pi/2] ; % 6 interpolācijas mezgli
ynodes = [0 0.7486 0.9286 0.8849 -0.1010 0.67];
f = @(x)sin(x.^3+2*x); % vektora ynodes komponentes ir funkcijas vērtības
% vektors x satur punktus, kuros ir jāaprēķina interpolācijas polinoma
% vērtības
x = linspace(0,pi/2,200); y = interp1(xnodes,ynodes,x,'pchip');
plot(x,y,'r-',xnodes,ynodes,'ob',x,f(x),'k-','LineWidth',3)
legend('interpolācijas polinoms','mezgli','y = sin(x^3+2x)')
title('Ermita interpolācija( interp1 )'), grid on
```



3.7. att. Interpolācijas polinoma un funkcijas $y = \sin(x^3 + 2x)$ grafiki.

```
% 3.6. piemēra turpinājums
figure  % Uzzīmēt interpolācijas kļūdas grafiku
plot(x,f(x)-y,'g:','LineWidth',3)
title('Interpolācijas kļūda'), grid on
```

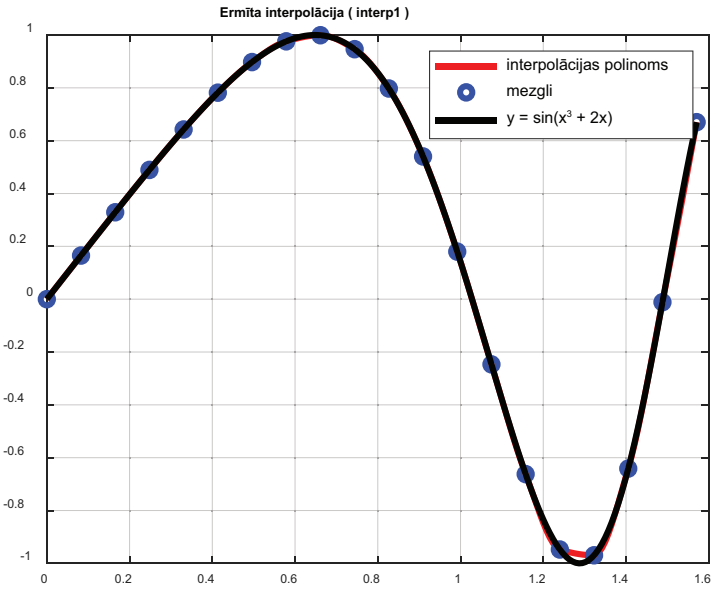


3.8. att. Interpolācijas kļūdas grafiks.

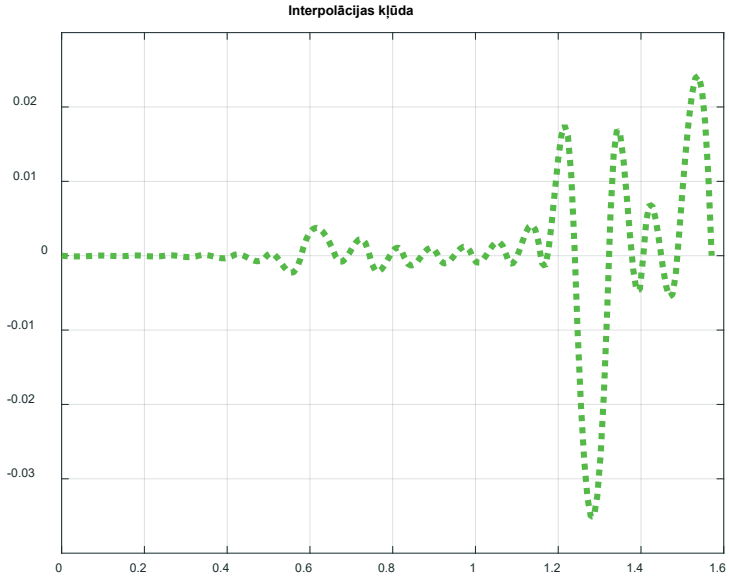
3.7. un 3.8. attēls ilustrē interpolācijās iespējamās problēmas. Interpolācijas kļūda ir pietiekami lielā intervālā (1, 1.6). Viens loģisks risinājums būtu palielināt interpolācijas mezglu skaitu.

Ja mainīgos **xnodes** un **ynodes** 3.6. piemērā aizstāj ar operatoriem, tad iegūst šādu rezultātu (sk. 3.9. attēlu).

```
%% 3.6. piemēra turpinājums
clc, clearvars, format compact, close all
xnodes = linspace(0,pi/2,20);    % 20 interpolācijas mezgli
f = @(x)sin(x.^3+2*x);
ynodes = f(xnodes);
... ... ...
```



3.9. att. Interpolācijas polinoma un funkcijas $y = \sin(x^3 + 2x)$ grafiki.



3.10. att. Interpolācijas kļūdas grafiks.

Tādējādi, palielinot interpolācijas mezglu skaitu 3.6. piemērā, iegūst precīzāku interpolācijas modeli.

Vai tā ir vienmēr?

Atbilde uz šo jautājumu ir negatīva.

Aplūkosim divus klasiskus piemērus, kas ir saistīti ar polinomiālo interpolāciju vienmērīgā režģī. Pirmais piemērs ir **Runges piemērs**.

3.7. piemērs. Runge 1901. gadā interpolēja funkciju $f(x)$ intervālā $[-1, 1]$ vienmērīgā režģī ar konstantu attālumu starp interpolācijas mezgliem.

$$f(x)=\frac{1}{1+25x^2}$$

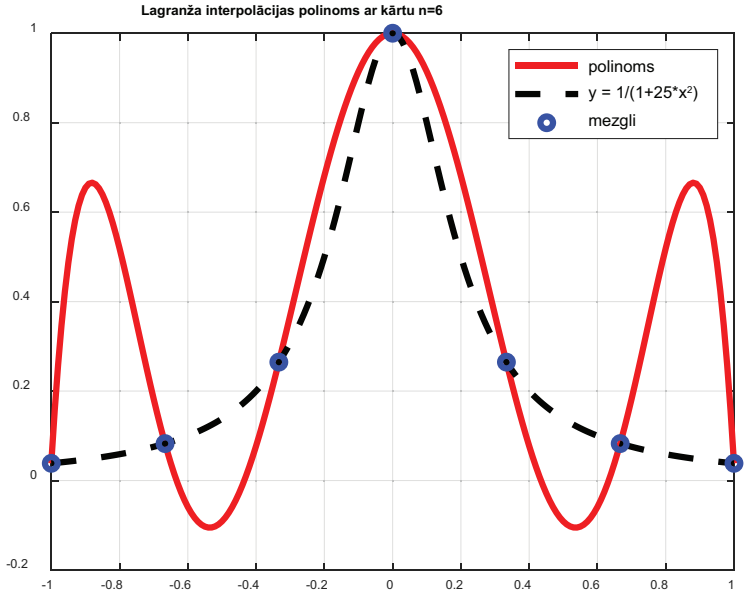
Atrisinājums.

Aplūkosim Lagranža interpolācijas polinomu ar kārtu 6 (skripts ir parādīts zemāk). Interpolācijas polinoma un funkcijas grafiki ir parādīti 3.11. attēlā.

```
%% 3.7. piemērs.(Runges piemērs)
clc, clearvars, format compact, close all
% vienmērīgais režģis ar 7 punktiem intervālā [-1,1]
xnodes = linspace(-1,1,7);
y = @(x)1./(1+25*x.^2); ynodes=y(xnodes);
coef = ynodes; m = length(xnodes);
syms x
pol = 0; % Lagranža interpolācijas polinoms
for k = 1:m
   w = 1;
     for j = [1:k-1 k+1:m]
         w = (x-xnodes(j))/(xnodes(k)-xnodes(j))*w;
     end
   pol = pol + w*ynodes(k);
end
polyn(x) = collect(pol);
coefpol = sym2poly(polyn) % polinoma koeficienti
```

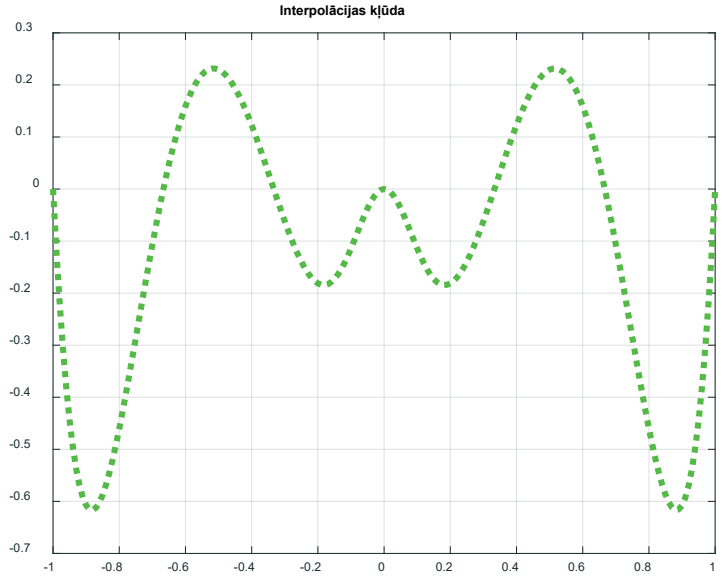
```
coefpol =
  -13.1349         0   20.9574         0   -8.7841         0    1.0000
```

```
% 3.7. piemēra turpinājums
x_pr = -1:0.01:1;
plot(x_pr,polyn(x_pr),'r-',x_pr,y(x_pr),'k--',xnodes,ynodes,'ob',...
     'LineWidth',3)
title('Lagranža interpolācijas polinoms ar kārtu n=6')
legend('polinoms','y=1/(1+25*x^2)','mezgli'), grid on
```



3.11. att. Interpolācijas polinoma un funkcijas grafiki (interpolācijas mezglu skaits ir 7).

```
% 3.7. piemēra turpinājums
figure  % Uzzīmēt interpolācijas kļūdas grafiku
plot(x_pr,y(x_pr)-polyn(x_pr),'g:','LineWidth',3)
title('Interpolācijas kļūda'), grid on
```

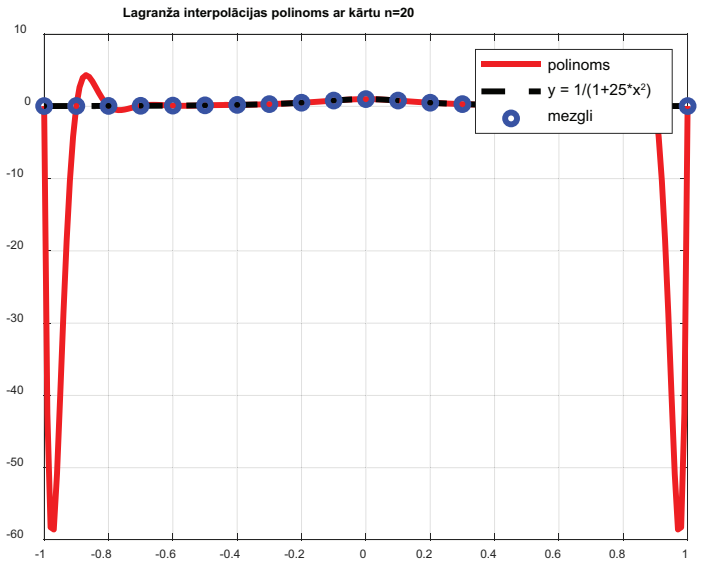


3.12. att. Interpolācijas kļūdas grafiks.

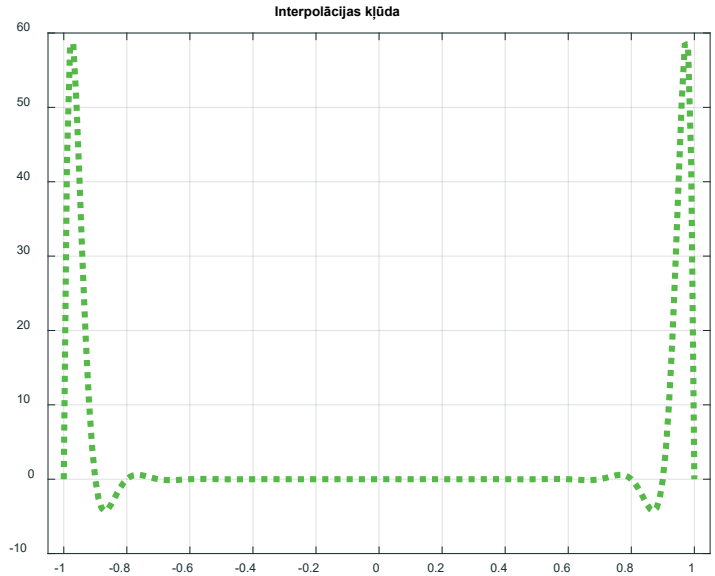
Kā var redzēt 3.12. attēlā, interpolācijas kļūda ir diezgan liela visā intervālā [–1, 1]. Lai uzlabotu situāciju, palielināsim interpolācijas mezglu skaitu (no 7 punktiem, kas atbilst polinomam ar kārtu 6, līdz 21 punktam, kas atbilst polinomam ar kārtu 20). Rezultāts ir parādīts 3.13. un 3.14. attēlā.

```
coefpol =
260178.630972     0.000000    -1012094.874480     0.000000    1639177.410848
0.000000    -1442944.963439     0.000000        757287.092775     0.000000
-245249.283925    0.000000      49317.542913      0.000000     -6119.219949
0.000000        470.846227      0.000000        -24.143480       0.000000
1.000000
```

Interpolācijas kļūda ir pietiekami maza intervāla vidusdaļā, bet tā ļoti strauji aug pie intervāla galapunktiem. Lielāka interpolācijas mezglu skaita gadījumā oscilācijas ar lielāko amplitūdu parādās pie intervāla galapunktiem.



3.13. att. Interpolācijas polinoma un funkcijas grafiki (interpolācijas mezglu skaits ir 21).



3.14. att. Interpolācijas kļūda.

Vēl viens piemērs, kas ilustrē problēmas ar polinomiālo interpolāciju vienmērīgā režģī, ir Bernšteina piemērs.

3.8. piemērs. Bernšteins 1916. gadā interpolēja funkciju $y = |x|$ intervālā [-1,1] ar Lagranža interpolācijas polinomiem vienmērīgā režģī.

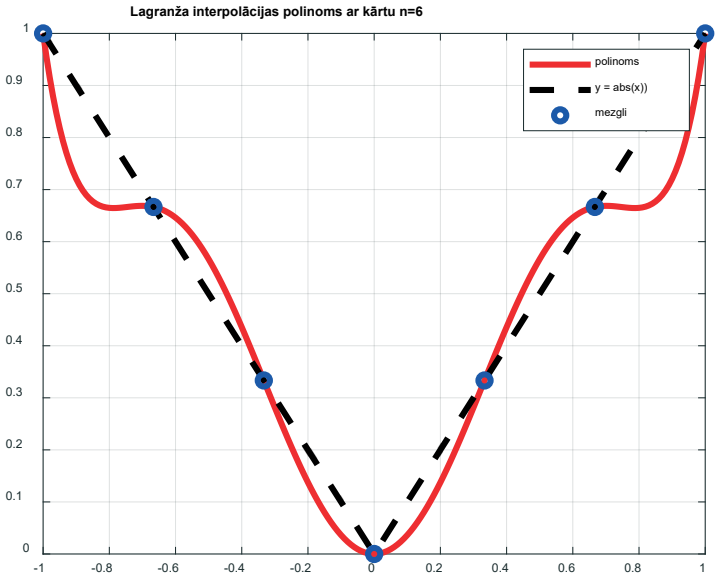
Atrisinājums.

Iegūtie rezultāti ir ļoti līdzīgi Runges piemēram (interpolācijas polinomu grafiki ir parādīti turpmāk).

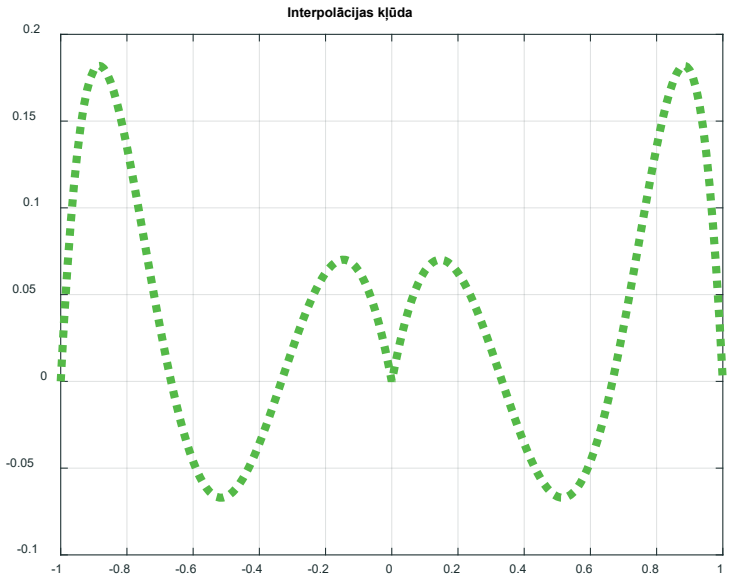
```
coefpol =
    4.0500         0   -6.7500         0    3.7000         0         0
```

```
coefpol =
   75.3520         0  -167.9274         0  128.3999         0  -41.2809
         0    6.4563         0         0
```

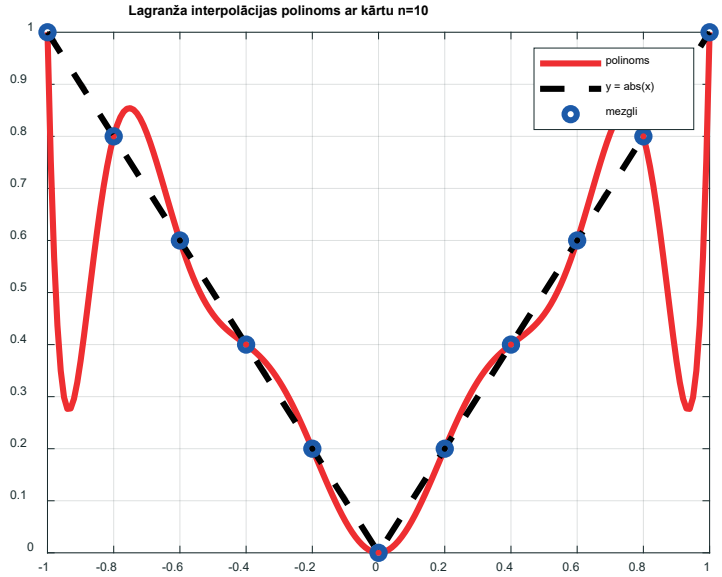
Tādējādi interpolācija ar augstāku kārtu polinomiem nav laba ideja. Ir zināms, ka interpolācijas mezglu skaits n ir tieši saistīts ar interpolācijas polinoma pakāpi $(n-1)$.



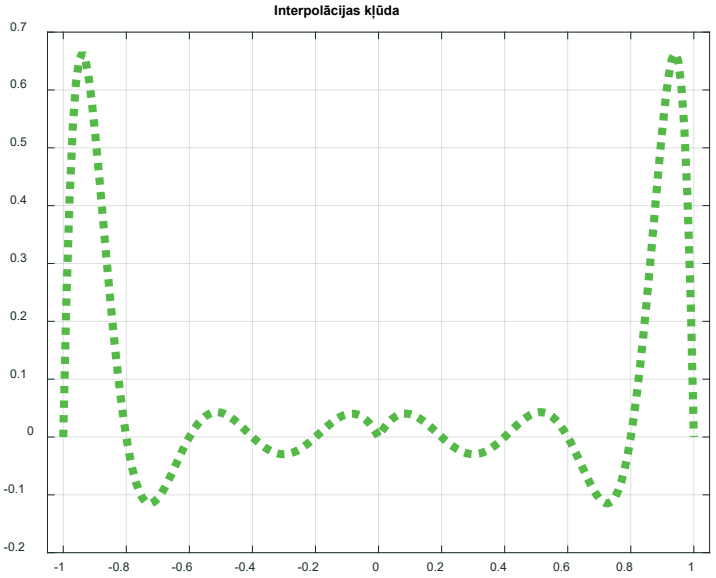
3.15. att. Interpolācijas polinoma un funkcijas grafiki (interpolācijas mezglu skaits ir 7).



3.16. att. Interpolācijas kļūda.



3.17. att. Interpolācijas polinoma un funkcijas grafiki (interpolācijas mezglu skaits ir 11).



3.18. att. Interpolācijas kļūda.

3.5. Splainu interpolācija

Jautājums: ko darīt, ja interpolācijas mezglu skaits ir liels?

Laba polinomiālās interpolācijas alternatīva ir splainu interpolācija.

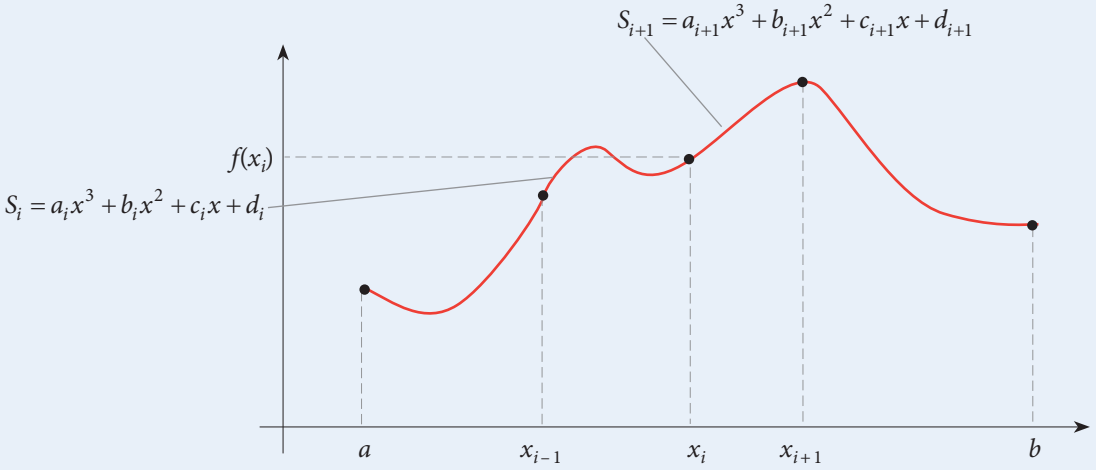
Pieņemsim, ka $y=f(x)$ ir nepārtraukta intervālā [a, b]. Par kubisko splainu (kas atbilst funkcijai $f(x)$ un interpolācijas mezgliem $a=x_0<x_1<x_2<\ldots<x_{(n-1)}<x_n=b$) sauc funkciju $s(x)$, kurai piemīt šādas īpašības:

(a) katrā apakšintervālā $(x_{(i-1)}, x_i)$, $i=1, 2, \ldots, n$ tā ir trešās pakāpes polinoms;

(b) funkcijas $s(x)$, $s'(x)$ un $s''(x)$ ir nepārtrauktas intervālā (a, b);

(c) $s(x_i)=f(x_i)$, $i=1, 2, \ldots, n-1$

Tādējādi, izmantojot kubiska splaina interpolācijas metodi, doto tabulu interpolē ar trešās kārtas interpolācijas polinomu katrā apakšintervālā (tas nozīmē, ka katrā apakšintervālā ir definēts polinoms ar koeficientiem a_i, b_i, c_i un d_i). Koeficientus izvēlās tā, lai polinomiem S_{i+1} un S_i (sk. 3.19. attēlu) punktos x_i sakristu ne tikai vērtības, bet arī pirmās un otrās kārtas atvasinājumi.



3.19. att. Splainu interpolācija.

3.6. Aprēķini *MATLAB* vidē. Splainu interpolācija

3.9. piemērs. Interpolēt uzdevumu 3.8. piemērā ar splainiem. Izmantot 11 interpolācijas mezglus.

Atrisinājums.

```
%% 3.9. piemērs. Splainu interpolācija
clc, clearvars, format compact, close all
f = @(x)abs(x);
xnodes = linspace(-1,1,11); ynodes = f(xnodes);
x = linspace(-1,1,200);
y = interp1(xnodes,ynodes,x,'spline');
plot(x,y,'r-',x,f(x),'k--',xnodes,ynodes,'ob','LineWidth',3)
legend('splains','y = abs(x)', 'mezgli')
title('Splainu interpolācija')
grid on
```

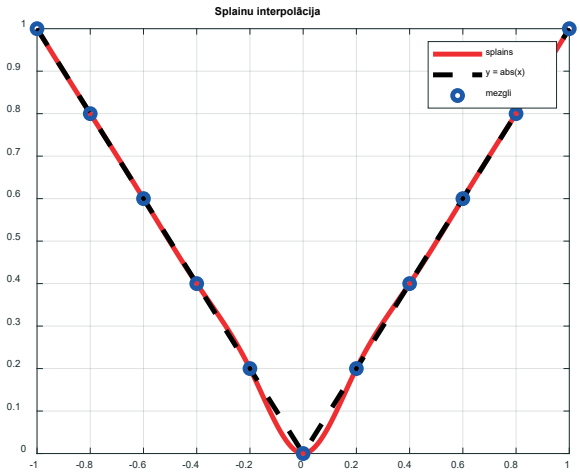
(x vektors) satur punktus, kuros dotā funkcija ir jāinterpolē

Sk. 3.20. attēlu.

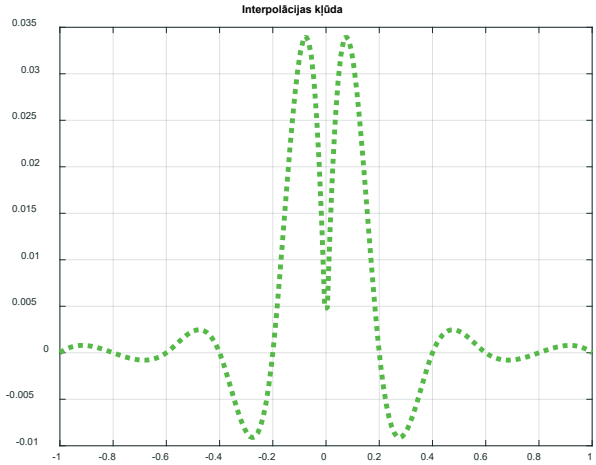
```
% 3.9. piemēra turpinājums
figure
plot(x,f(x)-y,'g:','LineWidth',3)
title('Interpolācijas kļūda')
grid on
```

Sk. 3.21. attēlu.

Kā redzams 3.20. un 3.21. attēlā, splainu interpolācijas kļūda ir diezgan maza visā intervālā.



3.20. att. Interpolācija ar kubiskajiem splainiem (Bernšteina piemērs).



3.21. att. Interpolācijas kļūda (Bernšteina piemērs).

3.10. piemērs. Izveidot funkcijas $f(x)=\sqrt{2+x^7}$ vienādi attālinātu vērtību tabulu intervālā $[1; 5]$ ar soli $\Delta x=1$. Interpolēt šo tabulu ar kubisko splainu.

- Atrast interpolācijas rezultātu punktā $x_0=2.5$.
- Kāda ir interpolācijas kļūda $\left|\tilde{f}(x_0)-f(x_0)\right|$, kur $\tilde{f}(x_0)$ ir interpolācijas rezultāts?
- Kāda ir interpolācijas kļūda punktā $x_1=3$?

```
%% 3.10. piemērs. Splainu interpolācija
clc, clearvars, format compact, close all
f = @(x)sqrt(2+x.^7);
xnodes = (1:5); ynodes=f(xnodes);
x0 = 2.5;
int_res=interp1(xnodes,ynodes,x0,'spline'),f_res=f(x0)
int_error=abs(int_res-f_res)
```

punkts, kurā jāinterpolē dotā funkcija

```
int_res =
   24.6899
f_res =
   24.7457
int_error =
    0.0558
```

```
% 3.10. piemēra turpinājums
disp('Atbilde:')
disp(['1) interpolācijas rezultāts punktā (2.5) = ', num2str(int_res)])
disp(['2) interpolācijas kļūda punktā (2.5) = ', num2str(int_error)])
disp('3) interpolācijas kļūda punktā (x=3) = 0, ')
disp('tā kā punkts x=3 ir interpolācijas mezgla punkts')
```

```
Atbilde:
1) interpolācijas rezultāts punktā (2.5) = 24.6899
2) interpolācijas kļūda punktā (2.5) = 0.055847
3) interpolācijas kļūda punktā (x=3) = 0,
tā kā punkts x=3 ir interpolācijas mezgla punkts
```

UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

3.1. uzdevums. Doti 11 punkti un funkcija $y(x)$:

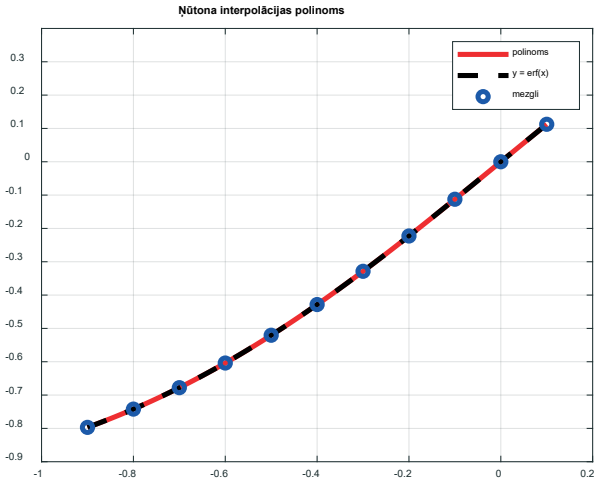
$$x_i = \frac{i-10}{10}, \quad i = 1,2,\ldots,11; \; y = \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t$$

Aprēķināt funkcijas $y(x)$ vērtības punktos x_i ar *MATLAB* (izmantot funkciju **`erf(x)`**). Konstruēt Ņūtona interpolācijas polinomu ar mezgliem punktos x_i. Uzzīmēt funkcijas $y(x)$ un interpolācijas polinoma grafikus intervālā $[x_1, x_{11}]$. Uzzīmēt interpolācijas kļūdas grafiku.

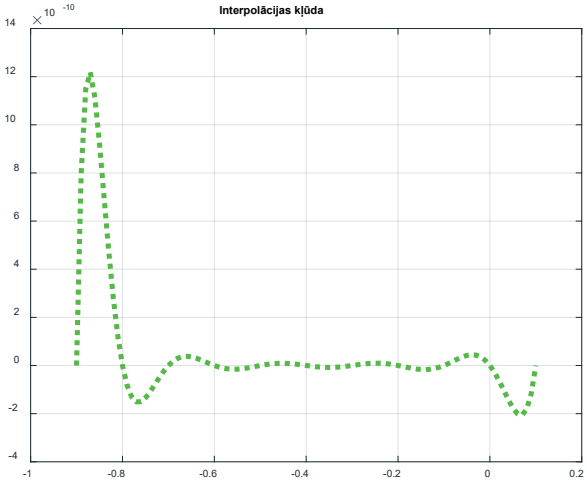
```
xnodes =
-0.9   -0.8   -0.7   -0.6   -0.5   -0.4   -0.3   -0.2   -0.1   0.0   0.1
```

```
coefpol =
   0.00181541515740166      0.0069918242700413      0.000906336611899692
  -0.0266590509872253      -1.4286111822919e-05     0.112819181917734
  -3.51775511226493e-06    -0.376126475297803       3.56816715189406e-08
   1.12837916960677        -4.14766951339907e-17
```

```
Atbilde. Ņūtona interpolācijas polinoms:
0.001815x^10+0.006992x^9+...+1.128379x-4.147669*10^(-17)
```



3.22. att. Interpolācijas polinoma un funkcijas grafiki.



3.23. att. Interpolācijas kļūdas grafiks.

3.2. uzdevums. Izveidot funkcijas $f(x)=x^3\sqrt{\cos x+x}$ vienādi attālinātu vērtību tabulu intervālā [3; 8] ar soli $\Delta x=1$. Interpolēt šo tabulu ar Ņūtona interpolācijas polinomu.

- Atrast interpolācijas rezultātu punktā $x_0=6.5$.
- Kāds ir interpolējošā polinoma koeficients pie mainīgā trešās pakāpes?
- Kāda ir interpolācijas kļūda $\left|\tilde{f}(x_0)-f(x_0)\right|$, kur $\tilde{f}(x_0)$ ir interpolācijas rezultāts?
- Kāda ir interpolācijas kļūda punktā $x_1=6$?

```
coefpol =
   1.0e+03 *
    0.0003   -0.0079    0.0910   -0.4571    1.0826   -0.9793
```

```
int_res =
  750.5564
f_res =
  750.9167
int_error =
    0.3603
```

```
Atbilde:
1) interpolācijas rezultāts punktā (6.5) = 750.5564
2) koef_x^3 = 91.0175
3) interpolācijas kļūda punktā (6.5) = 0.36029
4) interpolācijas kļūda punktā (x=6) = 0,
tā kā punkts x=6 ir interpolācijas mezgla punkts
```

3.3. uzdevums. Izveidot funkcijas $f(x)=x^3\sqrt{\cos x+x}$ vienādi attālinātu vērtību tabulu intervālā [3; 8] ar soli $\Delta x=1$. Interpolēt šo tabulu ar kubisko splainu.

- Atrast interpolācijas rezultātu punktā $x_0=6.5$.
- Kāds ir interpolējošā polinoma koeficients pie mainīgā trešās pakāpes?
- Kāda ir interpolācijas kļūda $\left|\tilde{f}(x_0)-f(x_0)\right|$, kur $\tilde{f}(x_0)$ ir interpolācijas rezultāts?
- Kāda ir interpolācijas kļūda punktā $x_1=6$?

```
int_res =
  750.3019
f_res =
  750.9167
int_error =
    0.6148
```

```
Atbilde:
1) interpolācijas rezultāts punktā (6.5) = 750.3019
2) interpolācijas kļūda punktā (6.5)  = 0.61485
3) interpolācijas kļūda punktā (6) = 0,
tā kā punkts x=6 ir interpolācijas mezgla punkts.
```

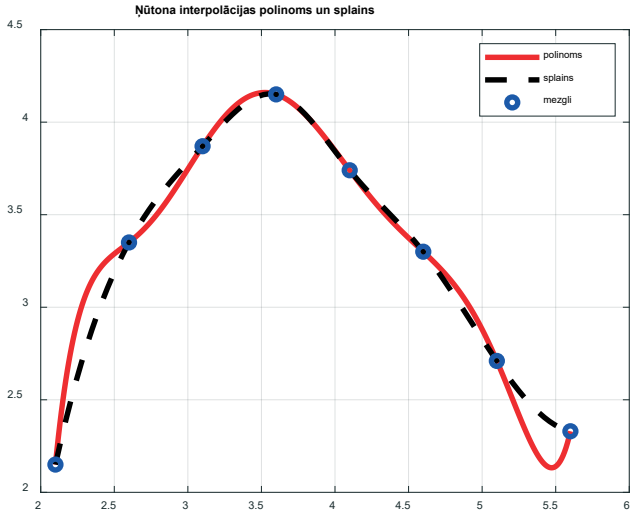
3.4. uzdevums. Funkcija ir dota ar tabulu:

x_i	2.1	2.6	3.1	3.6	4.1	4.6	5.1	5.6
y_i	2.15	3.35	3.87	4.15	3.74	3.30	2.71	2.33

Interpolēt doto funkciju, izmantojot interpolācijas polinomu un kubisko splainu. Uzzīmēt vienā attēlā dotās funkcijas, interpolācijas polinoma un kubiskā splaina grafikus.

```
coefpol =
   1.0e+03 *
   0.0002  -0.0049   0.0555  -0.3470   1.2783  -2.7741   3.2845  -1.6349
```

```
Atbilde. Ņūtona interpolācijas polinoms:
+0.1788x^7-4.8514x^6+55.5182x^5-347.0099x^4+1278.3078x^3
-2774.0712x^2+3284.5027x^1-1634.9137x^0
```



3.24. att. Interpolācijas polinoma un splaina grafiki.

3.5. uzdevums. Funkcija ir dota ar tabulu:

x_i	–1.000	–0.960	–0.860	–0.790	0.220	0.500	0.930
y_i	–1.000	–0.151	0.894	0.986	0.895	0.500	–0.306

Ir jānovērtē $y(x)$ intervālā [–1, 0.93], izmantojot interpolācijas metodes.

a) Atrast sestās pakāpes interpolācijas polinomu un uzzīmēt polinoma grafiku.

b) Atrast kubisko splainu, kas interpolē dotos punktus, un uzzīmēt splaina grafiku.

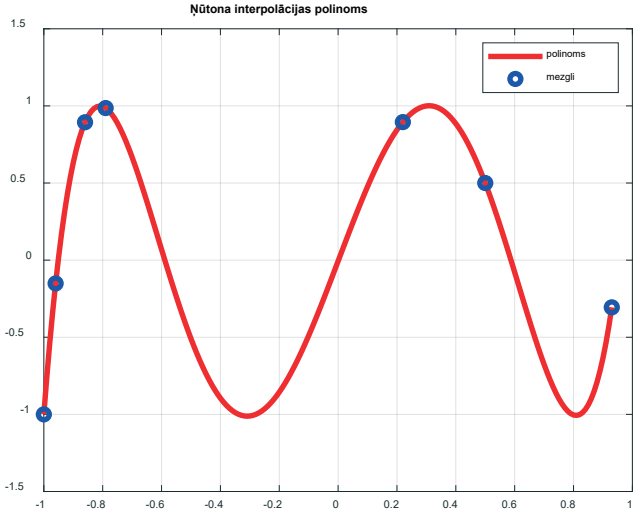
c) Kuru metodi labāk izmantot funkciju interpolācijai? Uzzīmēt polinoma un splaina grafikus vienā attēlā.

3.5. uzdevums vēlreiz ilustrē problēmas ar polinomiālo interpolāciju, ja funkciju interpolē ar augstāku kārtu polinomiem. Šajā gadījumā polinomam var būt vairāki ekstrēmi (sk. 3.27. attēlu), kas noved pie lielām interpolācijas kļūdām intervālos starp ekstrēma punktiem. Šeit ir rekomendējama interpolācija ar splainiem.

3.5. a)

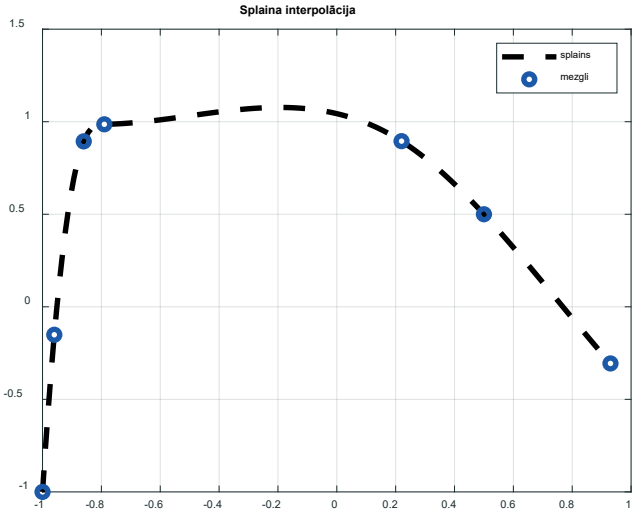
```
coefpol =
   0.0436   16.0767   -0.0484  -20.1005    0.0169    5.0295    -0.0064
```

```
Atbilde. Ņūtona interpolācijas polinoms:
+0.0436x^6+16.0767x^5-0.0484x^4-20.1005x^3+0.0169x^2+5.0295x^1-0.0064x^0
```



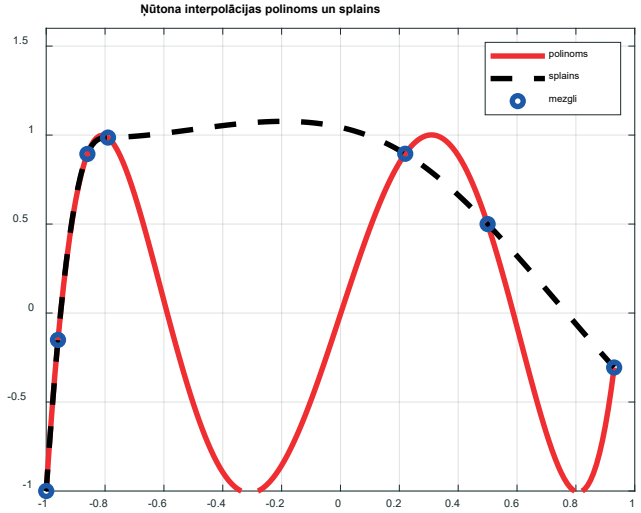
3.25. att. Interpolācijas polinoma grafiks.

3.5. b)



3.26. att. Splaina grafiks.

3.5. c)

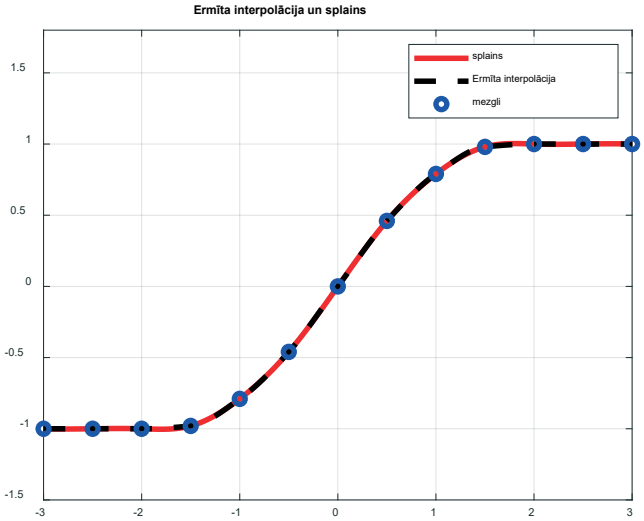


3.27. att. Interpolācijas polinoma un splaina grafiki.

3.6. uzdevums. Interpolēt tabulveidā doto funkciju, izmantojot komandu **`interp1`**.

Interpolācijas mezgli ir $x = -3{:}0.5{:}3$; funkcijas vērtības ir $y = (-1, -1, -1, -0.98, -0.79, -0.46, 0, 0.46, 0.79, 0.98, 1, 1, 1)$.

Izmantot Ermīta un splaina interpolāciju. Uzzīmēt Ermīta polinoma un splaina grafikus vienā attēlā.



3.28. att. Interpolācijas polinoma un splaina grafiki.

3.7. uzdevums. Izveidot funkcijas $f(x) = x^2 \cos\left(\sqrt[3]{x^2}\right)$ vienādi attālinātu vērtību tabulu intervālā $[2;6]$ ar soli $\Delta x = 1$.

a) Interpolēt šo tabulu ar Ņūtona interpolācijas polinomu. Atrast interpolācijas rezultātu punktā $x_0 = 2.5$.

b) Kāds ir interpolējošā polinoma koeficients, ja mainīgais ir trešajā pakapē?

c) Kāda ir interpolācijas kļūda $\left|\tilde{f}(x_0) - f(x_0)\right|$, kur $\tilde{f}(x_0)$ ir interpolācijas rezultāts?

d) Interpolēt šo tabulu ar kubisko splainu. Atrast interpolācijas rezultātu punktā $x_0 = 2.5$.

e) Kāda ir interpolācijas kļūda $\left|\tilde{f}(x_0) - f(x_0)\right|$, kur $\tilde{f}(x_0)$ ir interpolācijas rezultāts?

f) Kāda ir interpolācijas kļūda punktā $x_1 = 4$?

3.7. a), b), c)

```
coefpol =
    0.0648   -0.6556    0.1888    2.9796   -2.5726
```

```
pol_res =
   -1.6566
f_res =
   -1.6744
pol_error =
    0.0178
```

```
Atbilde:
 a) Ņūtona interpolācijas polinoms:
+0.0648x^4-0.6556x^3+0.1888x^2+2.9796x^1-2.5726x^0
 interpolācijas rezultāts punktā (2.5) = -1.6566
b) koef_x^3 = -0.65564
c) interpolācijas kļūda punktā (2.5) = 0.017804
```

3.7. d), e), f)

```
spl_res =
   -1.6202
f_res =
   -1.6744
spl_error =
    0.0543
```

```
Atbilde:
Kubiskais splains
d) interpolācijas rezultāts punktā (2.5) = -1.6202
e) interpolācijas kļūda punktā (2.5) = 0.054252
f) interpolācijas kļūda punktā (x=4) = 0,
tā kā punkts x=4 ir interpolācijas mezgla punkts
```

4. nodaļa

APROKSIMĀCIJA

4.1. Interpolācijas un aproksimācijas salīdzinājums

Aplūkosim punktu kopu (x_1, y_1), (x_2, y_2), …, (x_n, y_n), kur x ir neatkarīgais mainīgais un $y = y(x)$ ir funkcija no x.

x_i	y_i
x_1	y_1
x_2	y_2
x_2	y_2
…	…
x_n	y_n

Pieņemsim, ka funkcijas vērtības $y_i = y(x_i)$, $i = 1, 2, \ldots, n$ satur kļūdas (piemēram, tās ir iegūtas eksperimenta rezultātā). Pieņemsim arī, ka x nav vienīgais mainīgais, kas var ietekmēt y.

Formulēsim uzdevumu: atrast funkcijas $y(x)$ tuvinātās vērtības citos punktos, kas nav iekļauti tabulā.

Interpolācijas metode šeit neder!

Lai aproksimētu $y(x)$, aplūkosim funkciju $\varphi(x, a_1, a_2, \ldots, a_m)$, kur $a_1, a_2, \ldots, a_m$ ir nezināmie parametri.

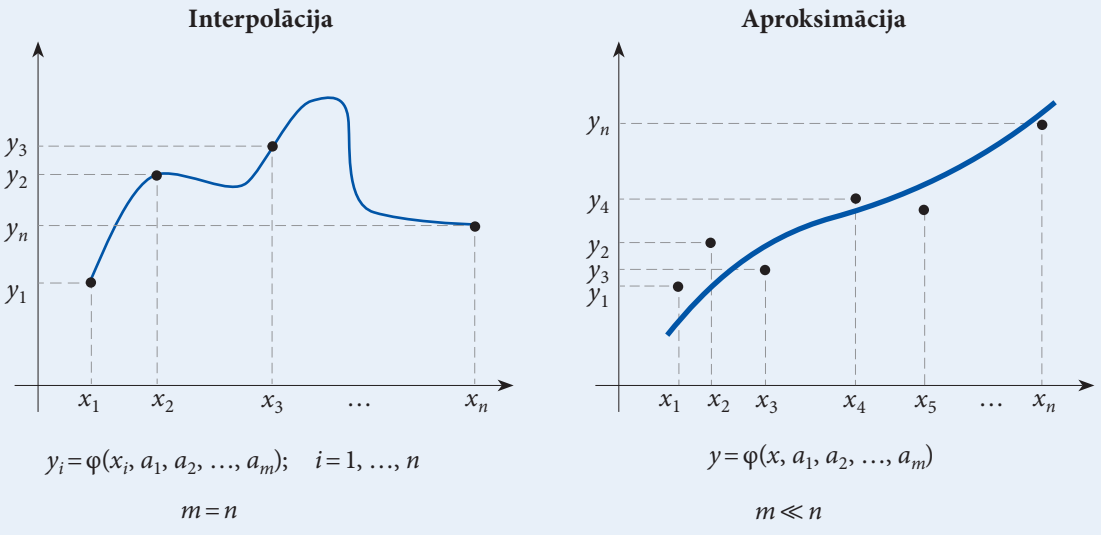
Starpība starp interpolācijas un aproksimācijas uzdevumiem ir redzama 4.1. attēlā.

Abos gadījumos ir dota funkcijas vērtību tabula un lietotājs izvēlas funkciju $\varphi(x, a_1, a_2, \ldots, a_m)$, kas ir atkarīga no nezināmajiem parametriem $a_1, a_2, \ldots, a_m$.

Interpolācijas metodē ir pieņemts, ka dotās funkcijas vērtības visos punktos $x = x_i$ sakrīt ar interpolējošās funkcijas $\varphi(x, a_1, a_2, \ldots, a_m)$ vērtībām. Nezināmo parametru skaits m interpolācijas uzdevumā sakrīt ar interpolācijas mezglu skaitu n. Interpolējošās funkcijas grafiks iet caur visiem punktiem tabulā.

Aproksimācijas uzdevumā nezināmo parametru skaits m parasti ir ievērojami mazāks nekā punktu skaits tabulā. Pavisam citu kritēriju izmanto, lai aprēķinātu parametrus $a_1, a_2, \ldots, a_m$. Konstruēsim funkcijas $\varphi(x, a_1, a_2, \ldots, a_m)$ grafiku tā, lai grafiks būtu "pēc iespējas tuvāk" punktiem tabulā. Protams, ir jāprecizē, ko tas nozīmē.

Uz brīdi pieņemsim, ka parametri $a_1, a_2, \ldots, a_m$ izteiksmē $\varphi(x, a_1, a_2, \ldots, a_m)$ ir zināmi. Tad varam aprēķināt aproksimācijas kļūdu ε_i katrā punktā $x = x_i$: $\varepsilon_i = y_i - \varphi(x_i, a_1, a_2, \ldots, a_m)$. Līdz ar to varam konstruēt vektoru $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)^T$.



4.1. att. Interpolācija un aproksimācija.

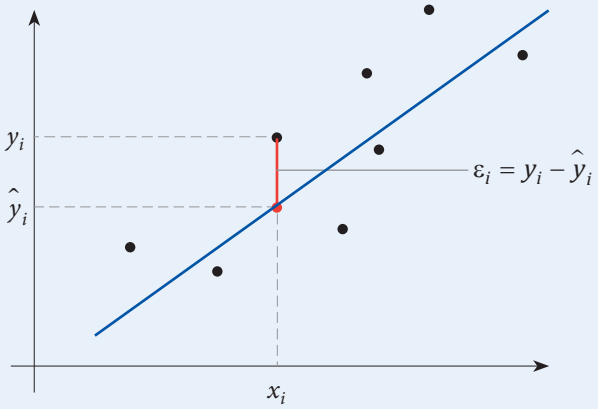
4.2. Aproksimācijas mazāko kvadrātu metode

Formulēsim uzdevumu: atrast tādas parametru a_1, a_2, …, a_m vērtības, kas minimizē normu $\|\varepsilon\|$ (vai normas kvadrātu). Ir zināms, ka var izmantot dažādas normas. Turpmāk aplūkosim vienu no populārākajām normām:

$$\|\varepsilon\|_2 = \sqrt{\varepsilon_1^2 + \varepsilon_2^2 + \ldots + \varepsilon_n^2}.$$

Šo aproksimācijas metodi sauc par **mazāko kvadrātu metodi**.

Aplūkosim visvienkāršāko aproksimāciju – aproksimāciju ar lineāru funkciju: $\hat{y} = a_0 + a_1 x + \varepsilon$, kur ε ir kļūda starp modeli un tabulā doto funkciju.



4.2. att. Aproksimācija ar mazāko kvadrātu metodi.

Kļūda parasti nav iekļauta modeļa formulējumā, bet vienmēr tiek pieņemts, ka kļūda ir.

Kļūda punktā $x = x_i$ ir $\varepsilon_i = y_i - \hat{y}_i = y_i - a_0 - a_1 x_i$.

Kļūdas normas kvadrāts ir

$$\|\varepsilon\|_2^2 = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i\right)^2. \tag{4.1}$$

Izteiksme formulas (4.1) labajā pusē ir divu argumentu a_0 un a_1 funkcija.

Tādējādi mazāko kvadrātu metodi var formulēt kā minimizācijas uzdevumu funkcijai (4.1).

Funkcijas $\|\varepsilon\|_2^2$ minimums var būt tikai stacionārajos punktos, kuros abi funkcijas (4.1) pirmās kārtas atvasinājumi pēc mainīgajiem a_0 un a_1 ir vienādi ar nulli. Pielīdzinot atvasinājumus nullei, iegūstam

$$\frac{\partial \|\varepsilon\|_2^2}{\partial a_0} = 0, \quad \frac{\partial \|\varepsilon\|_2^2}{\partial a_1} = 0. \tag{4.2}$$

Rezultāts ir

$$\frac{\partial \|\varepsilon\|_2^2}{\partial a_0} = -2\sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i\right) = 0 \tag{4.3}$$

$$\frac{\partial \|\varepsilon\|_2^2}{\partial a_1} = -2\sum_{i=1}^{n} x_i \left(y_i - a_0 - a_1 x_i\right) = 0 \tag{4.4}$$

Sistēmu (4.3), (4.4) var pārrakstīt šādi:

$$a_0 n + \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \tag{4.5}$$

$$a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \tag{4.6}$$

Sistēmas (4.5), (4.6) atrisinājums ir

$$a_0 = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{4.7}$$

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{4.8}$$

Lai novērtētu lineārās aproksimācijas precizitāti, izmantosim summas:

- $SST = \sum_{i=1}^{n} \left(y_i - \overline{y}\right)^2$ (kopējā noviržu kvadrātu summa);
- $SSR = \sum_{i=1}^{n} \left(\hat{y}_i - \overline{y}\right)^2$ (noviržu kvadrātu summa, ko izskaidro modelis);
- $SSE = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$ (noviržu kvadrātu summa, ko neizskaidro modelis).

Var pierādīt, ka

$$SST = SSR + SSE. \tag{4.9}$$

Dalot (4.9) ar *SST*, iegūstam

$$1 = \frac{SSR}{SST} + \frac{SSE}{SST} \tag{4.10}$$

Pirmo locekli formulas (4.10) labajā pusē sauc par determinācijas koeficientu:

$$R^2 = \frac{SSR}{SST}$$

Determinācijas koeficients rāda mainīgā *y* izkliedes īpatsvaru, kuru var izskaidrot ar aproksimācijas modeli.

Formula (4.10) rāda, ka divu nenegatīvu skaitļu summa ir vienāda ar 1. Tas nozīmē, ka katrs saskaitāmais nevar būt lielāks par 1.

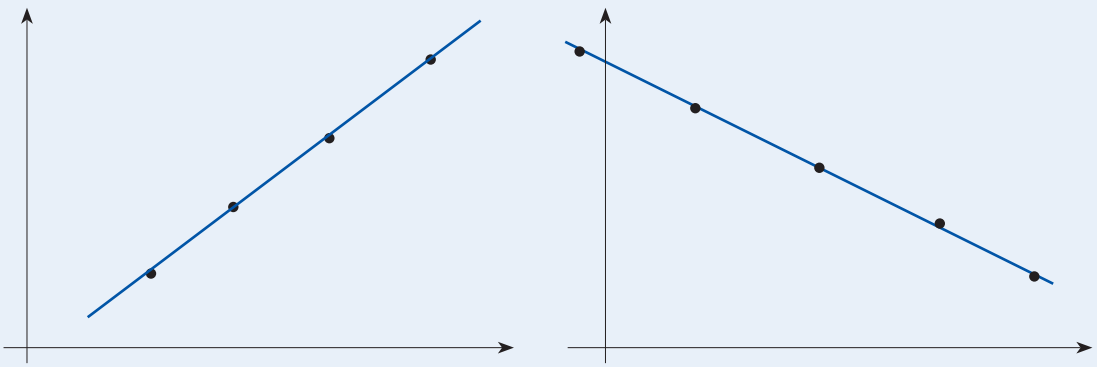
Tādējādi $0 \le R^2 \le 1$. Kas notiks gadījumā, kad $R^2 = 1$? No formulas (4.10) izriet, ka $SSE = 0$. Tas nozīmē, ka visas kļūdas ir vienādas ar nulli un visi punkti tabulā atrodas uz taisnes. Tādējādi ir perfekta lineārā sakarība starp *x* un *y*.

Determinācijas koeficienta R^2 vērtību var uzskatīt par rādītāju, kas raksturo lineārās sakarības ciešumu starp *x* un *y*.

Jo R^2 vērtība tuvāka 1, jo ciešāka ir lineārā sakarība starp *x* un *y*.

Datu punktiem, kas parāda nelineāro sakarību starp *x* un *y*, var izmantot polinomiālo aproksimāciju:

$$\hat{y} = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$$



4.3. att. Perfekta lineārā sakarība starp *x* un *y*.

4.3. Aproksimācijas aprēķini *MATLAB* vidē

MATLAB vidē lieto dažādas komandas, lai aproksimētu datu kopu.

`p = polyfit(x,y,n)`	**Aproksimācija ar polinomu** • `x` (rindas vektors) satur x vērtības tabulā • `y` (rindas vektors) satur atbilstošas funkcijas vērtības tabulā • `n` – polinoma kārta (`1` – pirmās kārtas polinoms, `2` – otrās kārtas polinoms, `3` – trešās kārtas polinoms utt.) • `p` (vektors) satur polinoma koeficientus
`y = polyval(p,x)`	aprēķina polinoma vērtības dotajos punktos
`[p,reg] = fit(x,y,modelis)` `// plot(p,x,y) //`	**Aproksimācija ar lineāru modeli** (modelis ir lineārs attiecībā pret parametriem) • `x` (kolonnas vektors) satur x vērtības tabulā • `y` (kolonnas vektors) satur atbilstošas funkcijas vērtības tabulā • `modelis` – izvēlētā modeļa veids (polinoms vai cita funkcija, kas ir lineāri atkarīga no koeficientiem) • `p` – lineārais modelis • `reg` – regresijas analīzes dati

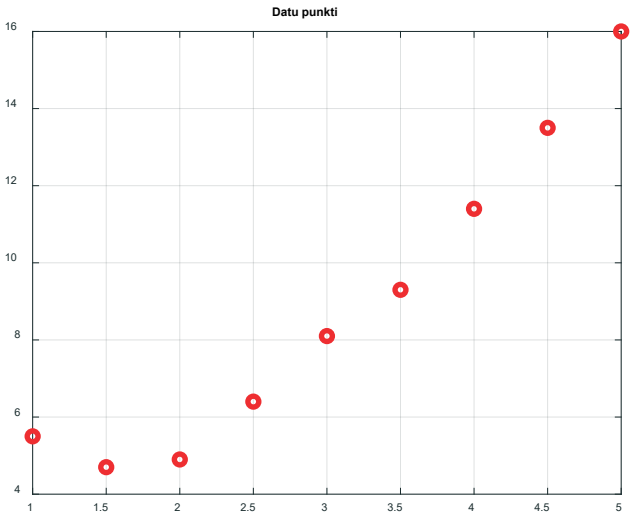
4.1. piemērs. Aproksimēt tabulu, izmantojot piemērotāko aproksimējošo funkciju

x_i	1	1.5	2	2.5	3	3.5	4	4.5	5
y_i	5.5	4.7	4.9	6.4	8.1	9.3	11.4	13.5	16.0

Atrisinājums.

Pirmais solis ir uzzīmēt datu kopu plaknē. Rezultāts ir parādīts 4.4. attēlā.

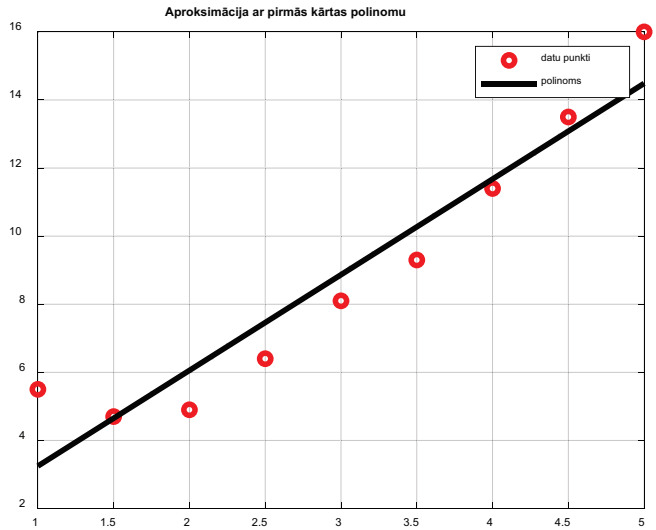
```
%% 4.1. piemērs. Aproksimācija (polyfit)
clc, clearvars,format compact, close all
xpoints=[1:0.5:5]; ypoints=[5.5,4.7,4.9,6.4,8.1,9.3,11.4,13.5,16.0];
plot(xpoints,ypoints,'or','LineWidth',3)
grid on
title('Datu punkti')
```



4.4. att. Tabulveidā dotās funkcijas grafiks.

```
% 4.1. piemēra turpinājums. Aproksimācija ar pirmās kārtas polinomu
p = polyfit(xpoints,ypoints,1)
xapprox = 1:0.01:5; yapprox = polyval(p,xapprox);
figure
plot(xpoints,ypoints,'or',xapprox,yapprox,'k','LineWidth',3)
legend('datu punkti','polinoms')
title('Aproksimācija ar pirmās kārtas polinomu'),grid on
fun_prob1(p) % polinoma drukāšana
```

```
p =
   2.8100    0.4367
```



4.5. att. Aproksimācija ar pirmās kārtas polinomu.

```
Atbilde. Pirmās kārtas polinoms:
   +2.8100x^1 +0.4367x^0
```

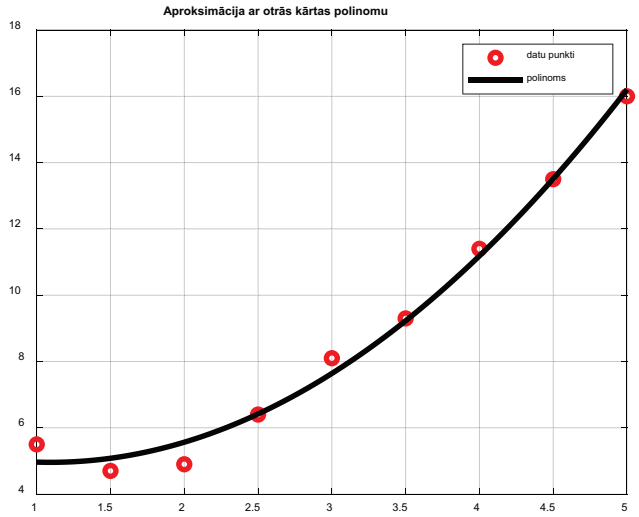
Lai izdrukātu polinomu, izmanto ārējo funkciju **fun _ prob1**.

```
% ārēja funkcija (4.1. piemērs). Aproksimācija
% polinoma drukāšana
function fun_prob1(koef)
   m = length(koef);
   fprintf('\n Atbilde.  %.0f. kārtas polinoms: \n  ',m-1)
   n = m-1;
   for i = 1:m
      if koef(i) < 0
        fprintf(' %.4fx^%.0f',koef(i),n)
      else
         fprintf(' +')
         fprintf('%.4fx^%.0f',koef(i),n)
      end
      n = n-1;
   end
   fprintf('\n')
end
```

Atkārtosim aprēķinus otrās un trešās kārtas polinomiem.

```
% 4.1. piemēra turpinājums - aproksimācija ar otrās kārtas polinomu
p = polyfit(xpoints,ypoints,2), yapprox = polyval(p,xapprox);
figure
plot(xpoints,ypoints,'or',xapprox,yapprox,'k','LineWidth',3)
legend('datu punkti','polinoms')
title('Aproksimācija ar otrās kārtas polinomu')
grid on
fun_prob1(p) % polinoma drukāšana
```

```
p =
    0.7364   -1.6082    5.8367
```

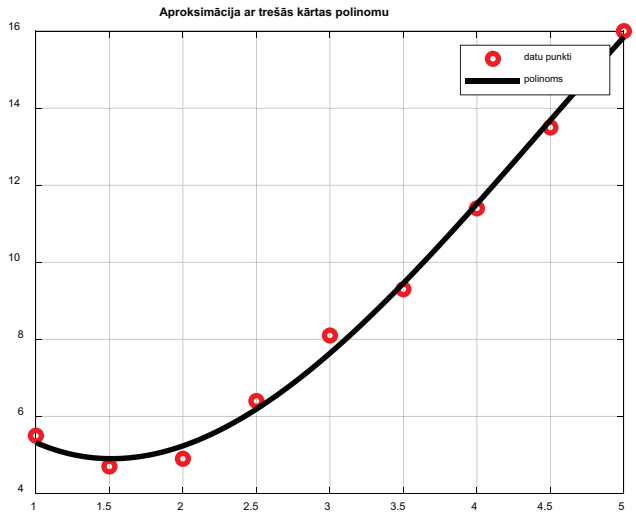


4.6. att. Aproksimācija ar otrās kārtas polinomu.

```
Atbilde. 2. kārtas polinoms:
   +0.7364x^2 -1.6082x^1 +5.8367x^0
```

```
% 4.1. piemēra turpinājums - aproksimācija ar trešās kārtas polinomu
p = polyfit(xpoints,ypoints,3), yapprox = polyval(p,xapprox);
figure
plot(xpoints,ypoints,'or',xapprox,yapprox,'k','LineWidth',3)
legend('datu punkti','polinoms')
title('Aproksimācija ar trešās kārtas polinomu')
grid on
fun_prob1(p) % polinoma drukāšana
```

```
p =
   -0.1697   2.2636   -5.6894   8.9167
```



4.7. att. Aproksimācija ar trešās kārtas polinomu.

```
Atbilde. Trešās kārtas polinoms:
   -0.1697x^3 +2.2636x^2 -5.6894x^1 +8.9167x^0
```

Vizuālā analīze (grafiku salīdzināšana 4.5., 4.6., un 4.7. attēlā) rāda, ka 4.1. piemērā visprecīzākā aproksimācija ir ar trešās kārtas polinomu.

Komandu **`polyfit`** var lietot tikai tad, kad aproksimācija ir ar polinomiem.

MATLAB vidē ir vēl viena iebūvēta funkcija **`fit`**, kuru var lietot, ja doto tabulu aproksimē ar funkciju, kas ir lineāri atkarīga no parametriem.

4.2. piemērs. Aproksimēt doto tabulu, izmantojot vispiemērotāko aproksimējošo funkciju.

x_i	0.0	1.0	1.5	2.0	3.0	4.0	5.0	6.0	6.5	7.0
y_i	8.2	7.0	6.3	5.0	4.7	7.0	8.0	8.5	8.8	9.0

Atrisinājums.

Komanda **`fit`** ļauj lietotājam iegūt ne tikai aproksimējošas funkcijas koeficientus, bet arī dažus aproksimācijas uzdevuma kvantitatīvos raksturotājus.

Aplūkosim trīs rādītājus: (a) determinācijas koeficientu R^2 (*rsquare*), (b) koriģēto determinācijas koeficientu R^2_{adj} (*adjrsquare*) un (c) vidējo kvadrātisko kļūdu *RMSE* (*rmse*).

Determinācijas koeficients rāda, kādu daļu no atkarīgā mainīgā variācijām izskaidro aproksimācijas modelis.

Koriģēto determinācijas koeficientu izmanto, lai salīdzinātu modeļus ar dažādu neatkarīgo mainīgo skaitu.

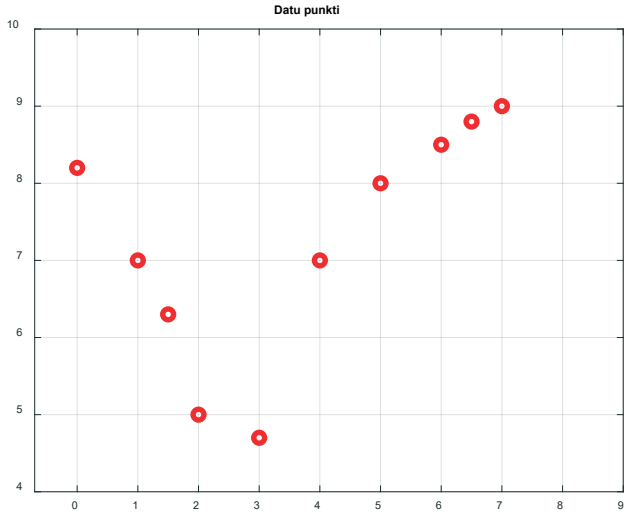
Vidējā kvadrātiskā kļūda raksturo, cik tuvu reālajām vērtībām tabulā atrodas prognozējamās vērtības.

Jāņem vērā:
- jo determinācijas koeficienta vērtība tuvāka 1, jo ciešāka ir sakarība starp atkarīgo mainīgo un neatkarīgajiem mainīgajiem;
- modeļi ar vislielāko koriģētā determinācijas koeficienta vērtību un vismazāko vidējo kvadrātisko kļūdu ir precīzāki.

Lai atrastu vispiemērotāko aproksimējošo funkciju, parasti rīkojas šādi:
- izvēlas dažas funkcijas;
- analizē R^2_{adj} un *RMSE* katrai funkcijai;
- starp aplūkotajiem modeļiem izvēlas to, kuram R^2_{adj} ir lielāka un *RMSE* ir mazāka.

Pirmajā solī (lai pieņemtu lēmumu par modeļa izvēli) uzzīmēsim dotās funkcijas grafiku.

```
%% 4.2. piemērs. Aproksimācija
clc, clearvars,format compact, close all
xpoints = [0,1,1.5,2:6,6.5,7]';
ypoints = [8.2,7.0,6.3,5.0,4.7,7,8,8.5,8.8,9]';
plot(xpoints,ypoints,'or','LineWidth',3)
grid on
title('Datu punkti')
```



4.8. att. Dotās funkcijas grafiks.

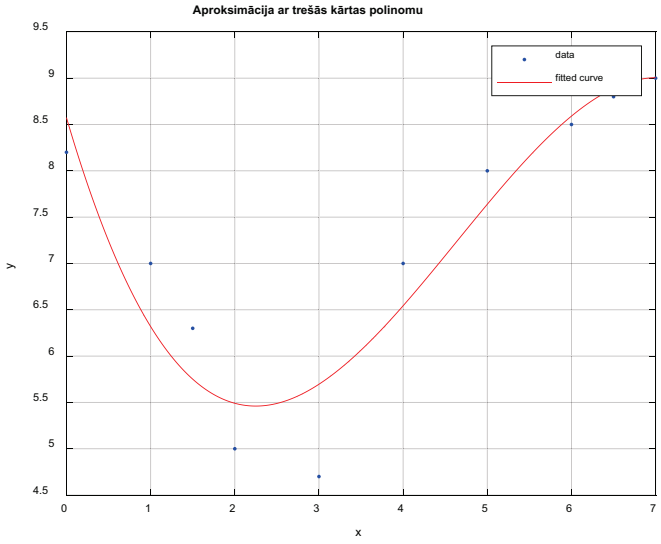
Risināsim aproksimācijas uzdevumu ar polinomiem. Aproksimēsim tabulu ar trešās kārtas polinomu.

```
% 4.2. piemēra turpinājums – aproksimācija ar trešās kārtas polinomu
[p,reg] = fit(xpoints,ypoints,'poly3')
figure
plot(p,xpoints,ypoints)
grid on
title('Aproksimācija ar trešās kārtas polinomu')
```

```
p =
     Linear model Poly3:
     p(x) = p1*x^3 + p2*x^2 + p3*x + p4
     Coefficients (with 95% confidence bounds):
       p1 =    -0.06544   (-0.1258, -0.005114)
       p2 =      0.9096   (0.2694, 1.55)
       p3 =      -3.099   (-4.973, -1.225)
       p4 =       8.574   (7.086, 10.06)
```

```
reg =
  struct with fields:

          sse: 2.4936
      rsquare: 0.8817
          dfe: 6
    adjrsquare: 0.8226
         rmse: 0.6447
```



4.9. att. Aproksimācija ar trešās kārtas polinomu.

Neskatoties uz to, ka $R^2 = 0.88$ aproksimācijas kļūda intervāla vidējā daļā (sk. 4.9. attēlu) ir pietiekami liela. Izvēlēsimies kādu citu modeli. Parasti praksē nelieto augstāko kārtu polinomus (piektās vai augstākas kārtas polinomus). Augstāku kārtu polinomiem var būt daudz ekstrēmu. Tas nozīmē, ka ekstrapolācijas rezultātā prognoze var ievērojami atšķirties no tendencēm, kuras apraksta datu kopa. Bieži izmanto aproksimāciju ar trigonometriskajām funkcijām.

Aproksimējošo funkciju izvēlēsimies šādu: $\hat{y} = a + b\sin x + c\cos x$. To var uzskatīt par divu vektoru skalāro reizinājumu, t. i., $\boldsymbol{d} = (a, b, c)$ un $\boldsymbol{z} = (1, \sin x, \cos x)$.

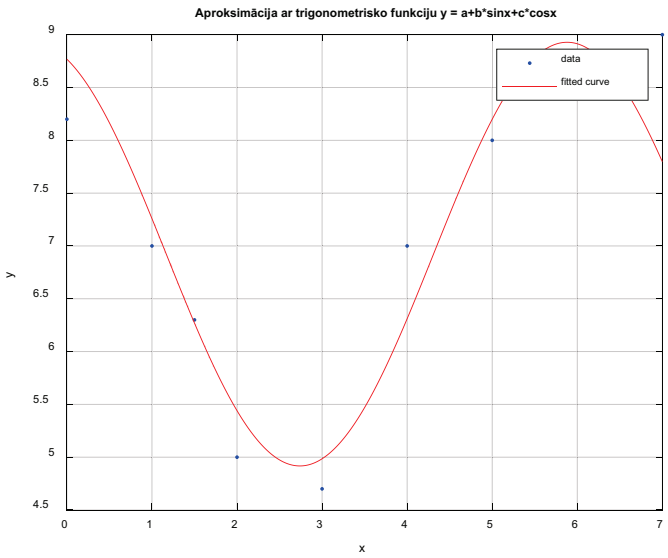
Skalārais reizinājums ir $\boldsymbol{d} \cdot \boldsymbol{z} = a \cdot 1 + b \cdot \sin x + c \cdot \cos x = a + b\sin x + c\cos x$.

Lai precizētu aproksimējošās funkcijas veidu, lietotājam ir jādefinē vektors $\boldsymbol{z}$. Šim nolūkam izmanto komandu **fittype**.

```
% 4.2. piemēra turpinājums – aproksimācija ar trigonometrisko
% funkciju: y = a+b*sinx+c*cosx
ft = fittype({'1','sin(x)','cos(x)'});
[p,reg] = fit(xpoints,ypoints,ft)
figure
plot(p,xpoints,ypoints), grid on
title('Aproksimācija ar trigonometrisko funkciju y = a+b*sinx+c*cosx')
```

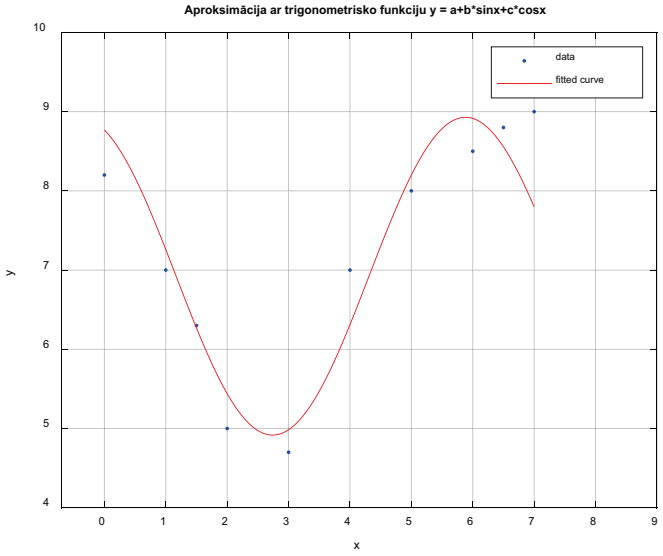
```
p =
     Linear model:
     p(x) = a + b*sin(x) + c*cos(x)
     Coefficients (with 95% confidence bounds):
       a =       6.923  (6.4, 7.445)
       b =     -0.7856  (-1.516, -0.05512)
       c =       1.845  (1.152, 2.538)
```

```
reg =
  struct with fields:
          sse: 2.8565
      rsquare: 0.8645
          dfe: 7
    adjrsquare: 0.8258
         rmse: 0.6388
```



4.10. att. Aproksimācija ar trigonometrisko funkciju.

```
% 4.2. piemēra turpinājums – aproksimācija ar trigonometrisko
% funkciju: y = a+b*sinx+c*cosx
ft = fittype({'1','sin(x)','cos(x)'});
[p,reg] = fit(xpoints,ypoints,ft)
figure
plot(p,xpoints,ypoints) ,grid on
title('Aproksimācija ar trigonometrisko funkciju y = a+b*sinx+c*cosx ')
xlim([-0.7 9]),ylim([4 10])
```



4.11. att. Aproksimācija ar trigonometrisko funkciju $\hat{y} = a + b\sin x + c\cos x$.

Salīdzināsim R^2_{adj} (*adjrsquare*) un *RMSE* (*rmse*) diviem modeļiem. Būtiskas starpības starp rādītājiem nav (koriģētais determinācijas koeficients ir nedaudz lielāks, un vidējā kvadrātiskā kļūda ir mazliet mazāka aproksimācijai ar trigonometriskajām funkcijām). Var arī redzēt (sk. 4.10. attēlu), ka intervāla vidējā daļā kļūda ir samazinājusies, bet pie galapunktiem tā joprojām ir pietiekami liela.

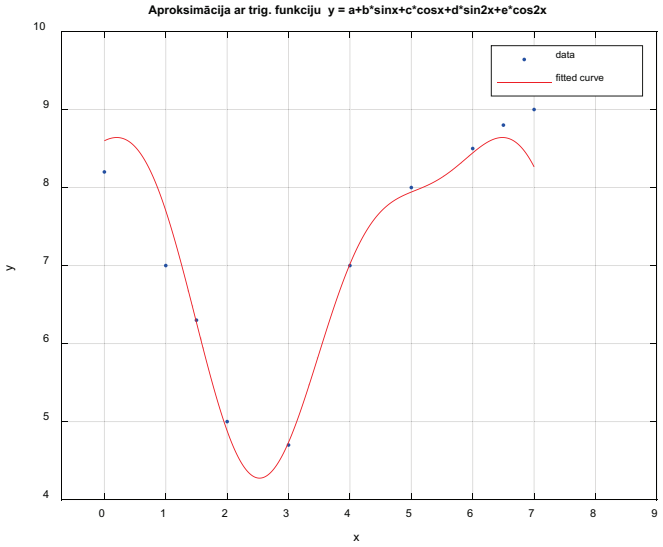
Izmantosim citu aproksimāciju ar trigonometriskajām funkcijām:

$$\hat{y} = a + b\sin x + c\cos x + d\sin 2x + e\cos 2x$$

```
% 4.2. piemēra turpinājums – aproksimējošā funkcija:
% y = a+b*sinx+c*cosx+d*sin2x+e*cos 2x
ft = fittype({'1','sin(x)','cos(x)','sin(2*x)','cos(2*x)'});
[p,reg] = fit(xpoints,ypoints,ft)
figure
plot(p,xpoints,ypoints), grid on
title(['Aproksimācija ar trigonometrisko funkciju',...
    'y = a+b*sinx+c*cosx+d*sin2x+e*cos2x'])
xlim([-0.7 9]), ylim([4 10])
```

```
p =
     Linear model:
     p(x) = a + b*sin(x) + c*cos(x) +
d*sin(2*x) + e*cos(2*x)
     Coefficients (with 95% confidence bounds):
       a =       6.875  (6.428, 7.322)
       b =     -0.8883  (-1.529, -0.2472)
       c =       1.791  (1.169, 2.412)
       d =      0.6389  (-0.01111, 1.289)
       e =    -0.06596  (-0.6389, 0.507)
```

```
reg =
  struct with fields:
           sse: 1.2424
       rsquare: 0.9411
           dfe: 5
    adjrsquare: 0.8939
          rmse: 0.4985
```



4.12. att. Aproksimācija ar trigonometrisko funkciju $\hat{y} = a + b\sin x + c\cos x + d\sin 2x + e\cos 2x$.

4.9.–4.12. attēla analīze rāda, ka vislabākā aproksimējošā funkcija (starp trim funkcijām) ir funkcija $\hat{y} = a + b\sin x + c\cos x + d\sin 2x + e\cos 2x$.

Par to liecina arī R^2_{adj} un *RMSE* salīdzinājums trim modeļiem:

N	1.	2.	3.
R^2_{adj}	0.8226	0.8258	0.8939
RMSE	0.6447	0.6388	0.4985

Aproksimācijai ar otro trigonometrisko funkciju ir vislielākā koriģētā determinācijas koeficienta vērtība un vismazākā vidējā kvadrātiskā kļūda.

```
% 4.2. piemēra turpinājums
disp('Atbilde:')
disp('Vislabākā aproksimācija ir otrā trigonometriskā funkcija')
disp([num2str(p.a),num2str(p.b),'sinx+',num2str(p.c),'cosx+',...
    num2str(p.d),'sin2x',num2str(p.e),'cos2x'])
```

```
Atbilde:
 Vislabākā aproksimācija ir otrā trigonometriskā funkcija
 6.8748-0.88825sinx+1.7905cosx+0.63891sin2x-0.06596cos2x
```

Rodas jautājums – kāpēc tieši funkcijas 1, $\sin x$, $\cos x$ vai funkcijas 1, $\sin x$, $\cos x$, $\sin 2x$, $\cos 2x$ ir izvēlētas aproksimācijai 4.2. piemērā? Atbilde uz šo jautājumu nav vienkārša. Bieži lietotājam jāizmanto dažādi varianti, kamēr izdosies atrast vispiemērotāko aproksimāciju.

Pirmais solis jebkurā aproksimācijas uzdevumā ir uzzīmēt tabulveidā dotās funkcijas grafiku. Aproksimācija ar polinomu ir loģiska 4.1. piemērā (sk. 4.4. attēlu).

Ja punktu sadalījums plaknē veido vissarežģītāko līkni (sk. 4.8. attēlu), tad lietotājs var mēģināt meklēt aproksimējošo funkciju šādi:

$$\varphi(x, a_1, a_2, \ldots, a_m) = a_1\varphi_1(x) + a_2\varphi_2(x) + \ldots + a_m\varphi_m(x), \tag{4.11}$$

kur funkciju $\varphi_1(x)$, $\varphi_2(x)$, …, $\varphi_m(x)$ izvēle nav tik acīmredzama.

Minimālā prasība funkcijām $\varphi_1(x)$, $\varphi_2(x)$, …, $\varphi_m(x)$ **ir lineārā neatkarība**, bet konkrēto izvēli dotajam piemēram formulēt grūti.

Parasti lietotājs izmanto dažādas funkcijas $\varphi_i(x)$, $i = 1, 2, \ldots, m$ formulā (4.11), turklāt parametru skaits m arī var būt atšķirīgs.

Pēc tam, analizējot dažas izvēlētās funkcijas un aproksimācijas skaitliskos rādītājus (R^2_{adj} un *RMSE*), pieņem lēmumu par to, kāda no funkcijām vislabāk aproksimē konkrēto uzdevumu.

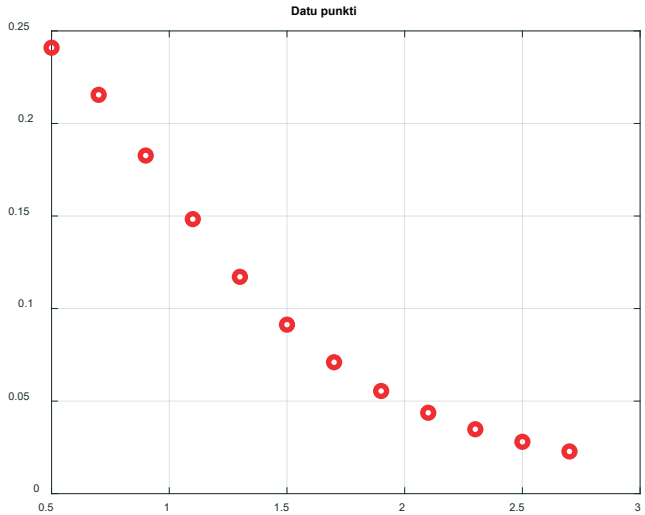
4.3. piemērs. Sastādīt funkcijas $f(x)$ tabulu intervālā [0.5; 2.7] ar soli 0.2. Aproksimēt tabulu, izmantojot formulu $y(x) = a + bx + c\sin x$. Aprēķināt aproksimācijas kļūdu punktā $x_0 = 1.33$ (kā starpību pēc moduļa starp funkciju $y(x)$ un $f(x)$ vērtībām punktā x_0).

$$f(x) = \frac{1}{3 + 2x^3 + \sqrt{\ln\left(x^2 + 2\right)}}$$

```
%% 4.3. piemērs. Aproksimācija
clc, clearvars,format compact, close all
f = @(x)1./(3+2*x.^3+sqrt(log(x.^2+2)));
xpoints = (0.5:0.2:2.7)'
ypoints = f(xpoints)
plot(xpoints,ypoints,'or','LineWidth',3), grid on
title('Datu punkti')
```

```
xpoints =
    0.5000
    0.7000
    0.9000
    1.1000
    1.3000
    1.5000
    1.7000
    1.9000
    2.1000
    2.3000
    2.5000
    2.7000
```

```
ypoints =
    0.2409
    0.2155
    0.1827
    0.1483
    0.1171
    0.0913
    0.0710
    0.0555
    0.0437
    0.0348
    0.0280
    0.0228
```

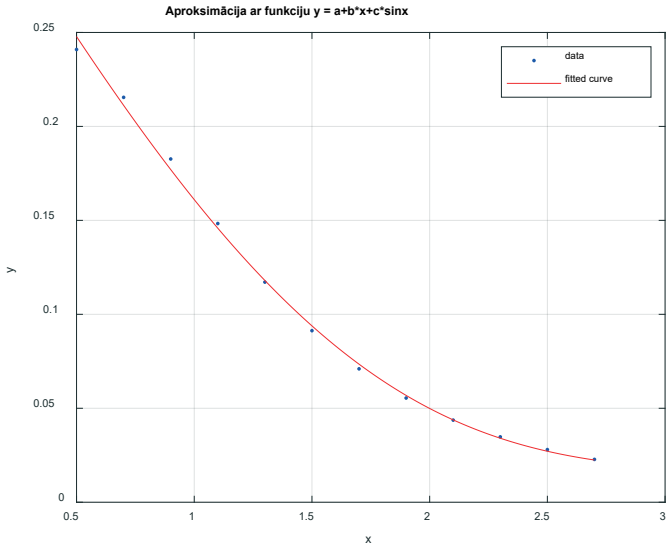


4.13. att. Dotās funkcijas grafiks.

```
% 4.3. piemēra turpinājums – aproksimācija ar trigonometrisko
% funkciju: y = a+b*x+c*sinx
ft = fittype({'1','x','sin(x)'});
[y,reg] = fit(xpoints,ypoints,ft)
figure
plot(y,xpoints,ypoints)
grid on
title('Aproksimācija ar trigonomterisko funkciju y = a+b*x+c*sinx')
```

```
y =
     Linear model:
     y(x) = a + b*x + c*sin(x)
     Coefficients (with 95% confidence bounds):
       a =       0.3458   (0.3339, 0.3576)
       b =      -0.1047   (-0.1082, -0.1012)
       c =     -0.09512   (-0.1076, -0.0826)
```

```
reg =
 struct with fields:
           sse: 1.2022e-04
       rsquare: 0.9981
           dfe: 9
     adjrsquare: 0.9977
          rmse: 0.0037
```



4.14. att. Aproksimācija ar funkciju $\hat{y} = a + bx + c\sin x$.

```
% 4.3. piemēra turpinājums. Aprēķināt aproksimācijas kļūdu punktā
% x_0 = 1.33
%(kā starpību pēc moduļa starp funkciju y(x) un f(x) vērtībām punktā x_0).
x0 = 1.33;
apr_error = abs(y(x0)-f(x0))
fprintf('Atbilde: \n ')
fprintf('Aproksimācijas kļūda punktā (1.33) = %.4f \n ',apr_error)
```

```
apr_error =
    0.0013
```

```
Atbilde:
 Aproksimācijas kļūda punktā (1.33) = 0.0013
```

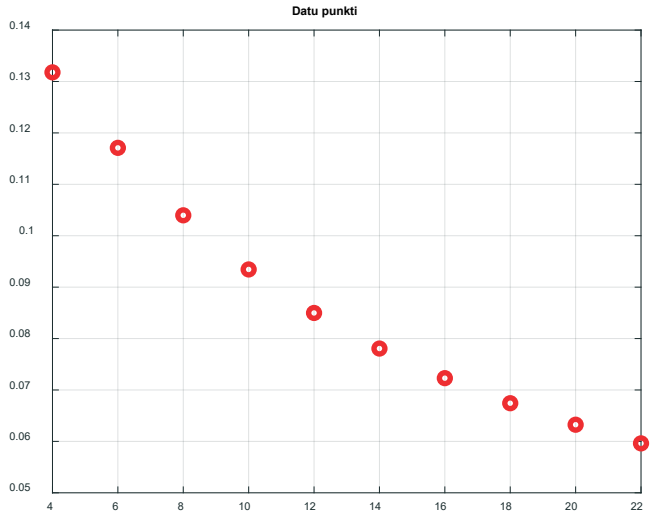
4.4. piemērs. Sastādīt funkcijas $f(x)$ vienādi attālinātu vērtību tabulu intervālā $[4, 22]$ ar soli $\Delta x = 2$. Interpolēt šo tabulu ar kubisko splainu (apzīmēsim šo funkciju ar $h(x)$). Aproksimēt šo tabulu ar trešās kārtas polinomu (apzīmēsim šo funkciju ar $g(x)$). Atrast $\left|h(x_1) - g(x_1)\right|$ punktā $x_1 = 8.25$

$$f(x) = \frac{\ln x}{2x + \sqrt[3]{x^2}}$$

```
%% 4.4. piemērs. Aproksimēt ar trešās kārtas polinomu un
% interpolēt ar kubisko splainu
clc, clearvars,format compact, close all
f = @(x)log(x)./(2*x+x.^(2/3));
xpoints = (4:2:22)'
ypoints = f(xpoints)
plot(xpoints,ypoints,'or','LineWidth',3)
grid on
title('Datu punkti')
```

```
xpoints =
     4
     6
     8
    10
    12
    14
    16
    18
    20
    22
```

```
ypoints =
    0.1318
    0.1171
    0.1040
    0.0934
    0.0850
    0.0781
    0.0723
    0.0674
    0.0632
    0.0596
```



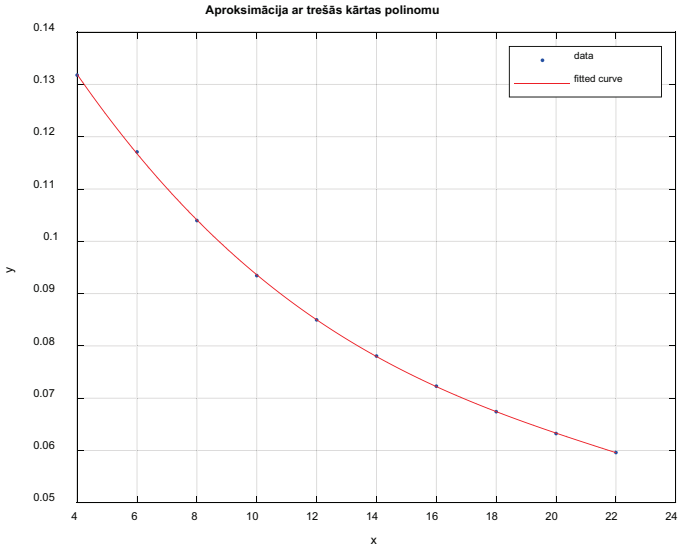
4.15. att. Dotās funkcijas grafiks.

```
% 4.4. piemēra turpinājums - aproksimācija ar trešās kārtas polinomu
[g,reg] = fit(xpoints,ypoints,'poly3')
figure
plot(g,xpoints,ypoints)
grid on
title('Aproksimācija ar trešās kārtas polinomu')
```

```
g =
     Linear model Poly3:
     g(x) = p1*x^3 + p2*x^2 + p3*x + p4
     Coefficients (with 95% confidence bounds):
       p1 =  -6.297e-06  (-7.283e-06, -5.311e-06)
       p2 =   0.0004239  (0.0003851, 0.0004626)
       p3 =    -0.01134  (-0.0118, -0.01088)
       p4 =      0.1709  (0.1693, 0.1725)
```

```
% 4.4. piemēra turpinājums - interpolācija ar kubisko splainu punktā x0
x0 = 8.25;
h_x0 = interp1(xpoints,ypoints,x0,'spline')
```

```
reg =
 struct with fields:
          sse: 1.9241e-07
      rsquare: 1.0000
          dfe: 6
   adjrsquare: 0.9999
         rmse: 1.7907e-04
```



4.16. att. Aproksimācija ar trešās kārtas polinomu.

```
h_x0 =
    0.1025
```

```
% 4. piemēra turpinājums. Aprēķināt kļūdu punktā x_0 = 8.25
% (aproksimācija un splains)
%(kā starpību pēc moduļa starp funkciju h_x0 un g(x)vērtībām punktā x_0).
kluda = abs(g(x0)-h_x0)
fprintf('Atbilde: \n ')
fprintf('Kļūda punktā (8.25) = %.4f \n ',kļūda)
```

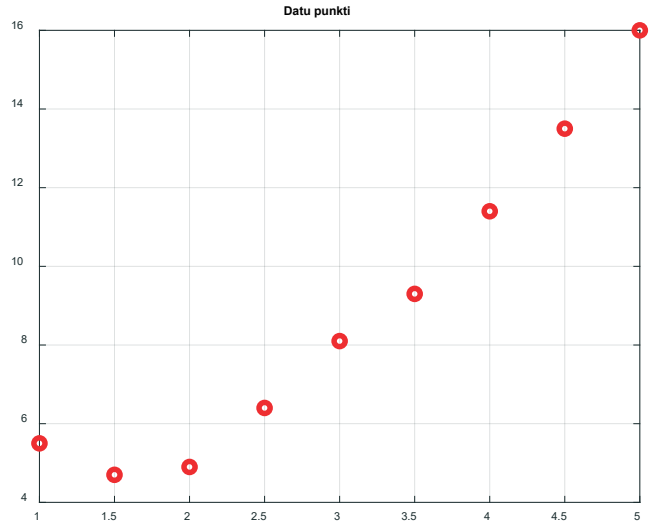
```
kļūda =
   1.5634e-04
```

```
Atbilde:
   Kļūda punktā (8.25) = 0.0002
```

UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

4.1. uzdevums. Aproksimēt tabulu, izmantojot vispiemērotāko aproksimējošo funkciju (izmantot komandu `fit`). Aproksimāciju veikt ar pirmās, otrās un trešās kārtas polinomiem.

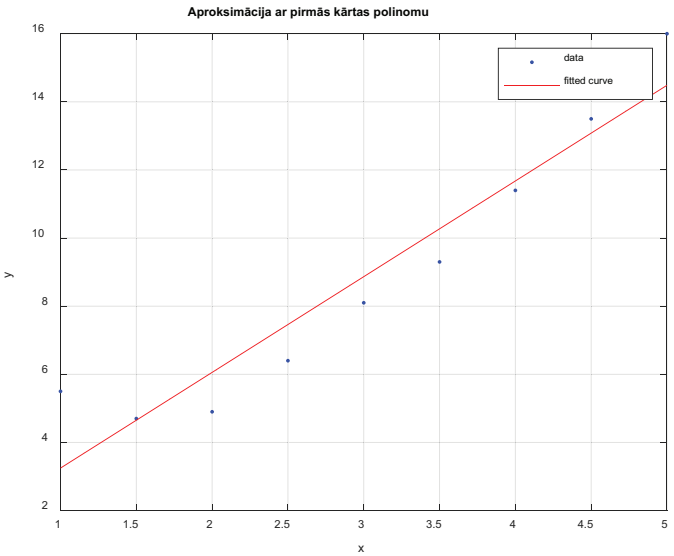
x_i	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
y_i	5.5	4.7	4.9	6.4	8.1	9.3	11.4	13.5	16.0



4.17. att. Dotās funkcijas grafiks.

```
p =
     Linear model Poly1:
       p(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
       p1 =  2.81     (2.023, 3.597)
       p2 =  0.4367   (-2.132, 3.006)
```

```
reg =
  struct with fields:
              sse: 11.6185
          rsquare: 0.9107
              dfe: 7
        adjrsquare: 0.8979
             rmse: 1.2883
```

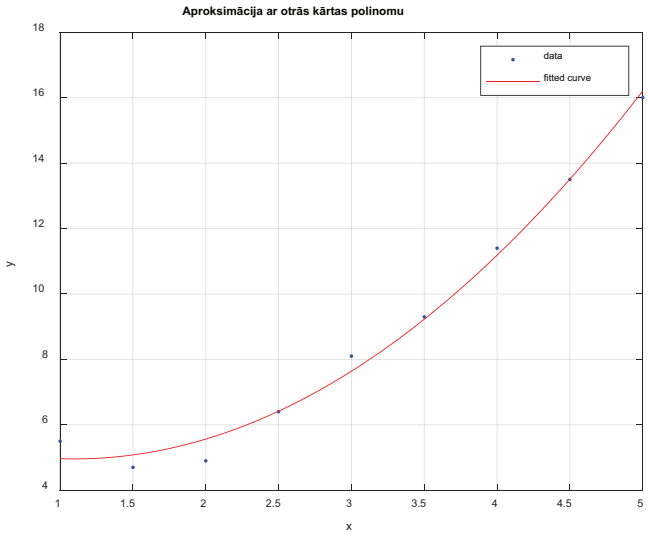


4.18. att. Aproksimācija ar pirmās kārtas polinomu.

```
Atbilde:
  Pirmās kārtas polinoms: 2.81x+0.43667
```

```
p =
 Linear model Poly2:
   p(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
 p1 =  0.7364  (0.489, 0.9837)
 p2 =  -1.608  (-3.119, -0.09766)
 p3 =   5.837  (3.805, 7.869)
```

```
reg =
  struct with fields:
      sse: 1.1805
      rsquare: 0.9909
          dfe: 6
    adjrsquare: 0.9879
         rmse: 0.44361
```

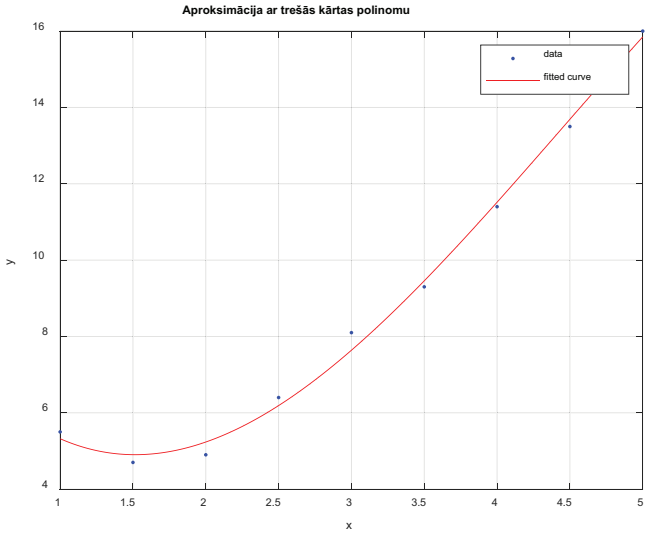


4.19. att. Aproksimācija ar otrās kārtas polinomu.

```
Otrās kārtas polinoms: 0.7364x^2-1.608x+5.837
```

```
p =
 Linear model Poly3:
p(x) = p1*x^3 + p2*x^2 + p3*x + p4
Coefficients (with 95% confidence bounds):
  p1 = -0.1697  (-0.3485, 0.009145)
  p2 =   2.264  (0.6426, 3.885)
  p3 =  -5.689  (-10.15, -1.231)
  p4 =   8.917  (5.306, 12.53)
```

```
reg =
  struct with fields:
      sse: 0.5391
      rsquare: 0.9959
          dfe: 5
    adjrsquare: 0.9934
         rmse: 0.3284
```



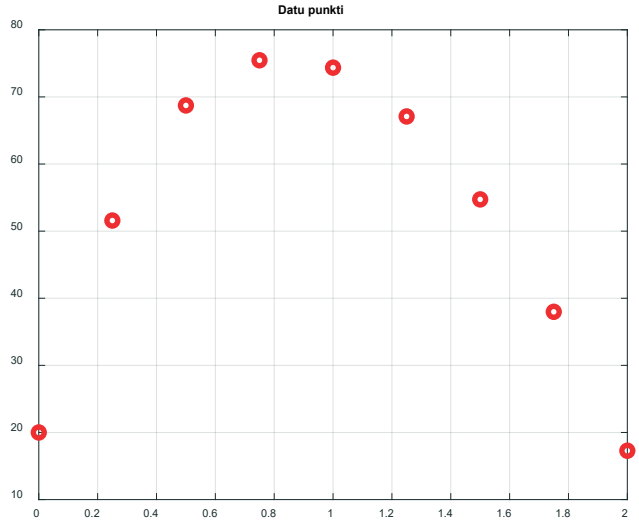
4.20. att. Aproksimācija ar trešās kārtas polinomu.

```
Trešās kārtas polinoms: -0.1697x^3+2.264x^2-5.689x+8.917
 Vislabākā aproksimācija ir trešās kārtas polinoms
```

Salīdzinot trīs aproksimējošo polinomu R^2_{adj} un $RMSE$ vērtības, secinām, ka visprecīzākā aproksimācija ir ar trešās kārtas polinomu.

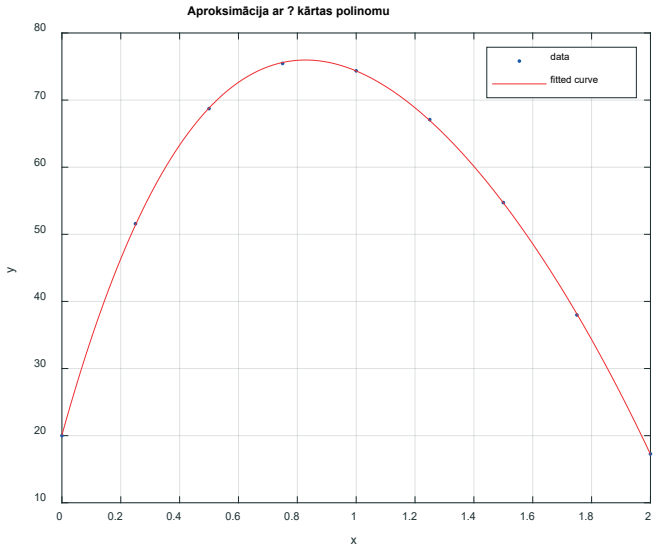
4.2. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar polinomiem (līdz ceturtās kārtas polinomiem ieskaitot).

x_i	0	0.25	0.5	0.75	1.0	1.25	1.5	1.75	2.0
y_i	20.00	51.58	68.73	75.46	74.36	67.09	54.73	37.98	17.28



4.21. zīm. Dotās funkcijas grafiks.

```
reg =
  struct with fields:
           sse: 0.1186
       rsquare: 1.0000
           dfe: 4
    adjrsquare: 0.9999
          rmse: 0.1722
```



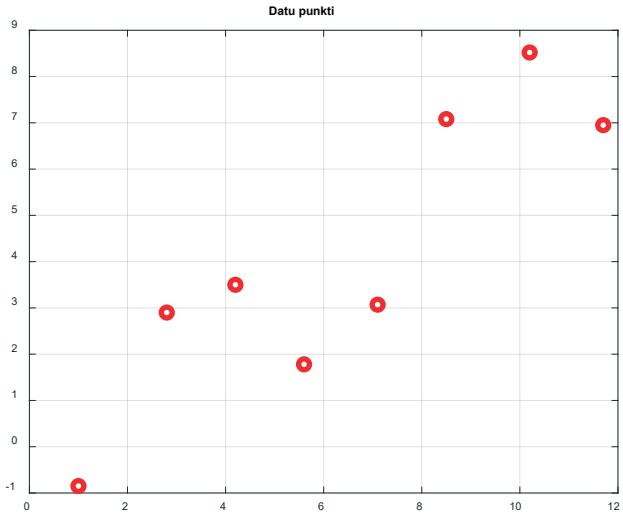
4.22. att. Aproksimācija ar polinomu.

```
Atbilde:
  Vislabākā aproksimācija ir ... ... ... ... ...
```

Atrisinājumā nav parādīta aproksimējošā polinoma pakāpe. Protams, $R^2 = 1.0000$ atbilst konkrētai polinoma pakāpei, bet līdzīgu bildi iegūst, izmantojot citas kārtas polinomu. Uzdevuma galvenais mērķis ir izmantot dažādus polinomus un izvēlēties vispiemērotāko.

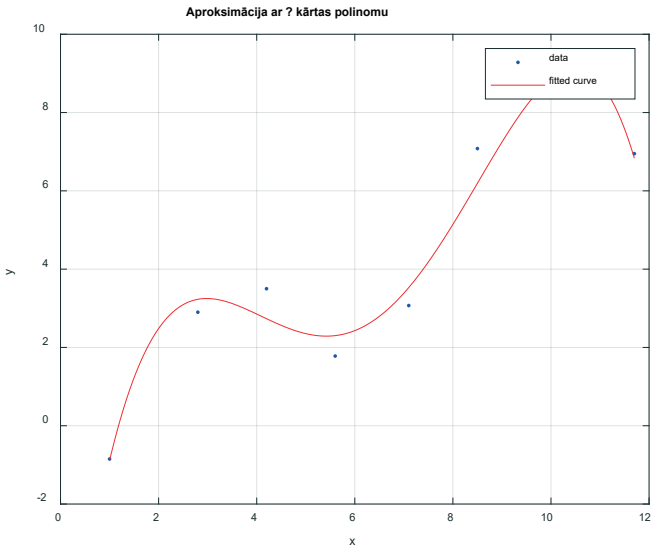
4.3. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar polinomiem (līdz ceturtās kārtas polinomiem ieskaitot).

x_i	1.0	2.8	4.2	5.6	7.1	8.5	10.2	11.7
y_i	−0.85	2.9	3.5	1.78	3.07	7.08	8.52	6.95



4.23. att. Dotās funkcijas grafiks.

```
reg =
  struct with fields:
       sse: 2.2263
       rsquare: 0.9679
           dfe: 3
     adjrsquare: 0.9250
          rmse: 0.8614
```



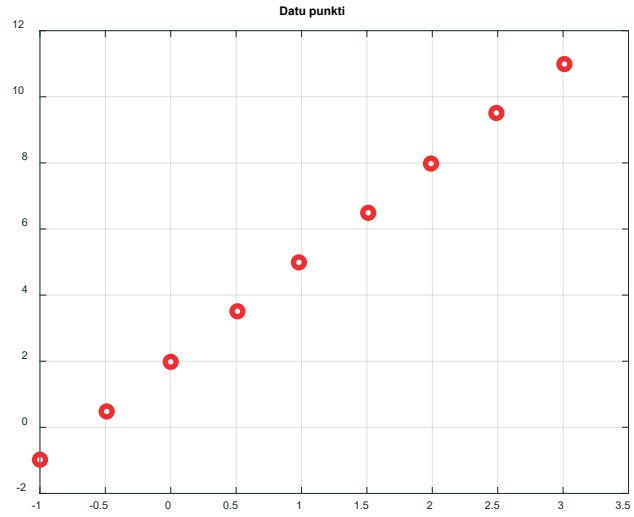
4.24. att. Aproksimācija ar polinomu.

```
Atbilde:
  Vislabākā aproksimācija ir ... ... ... ... ...
```

Šeit arī nav norādīta aproksimējošā polinoma pakāpe. Lietotājam pašam ir jāizvēlas vislabākā aproksimācija.

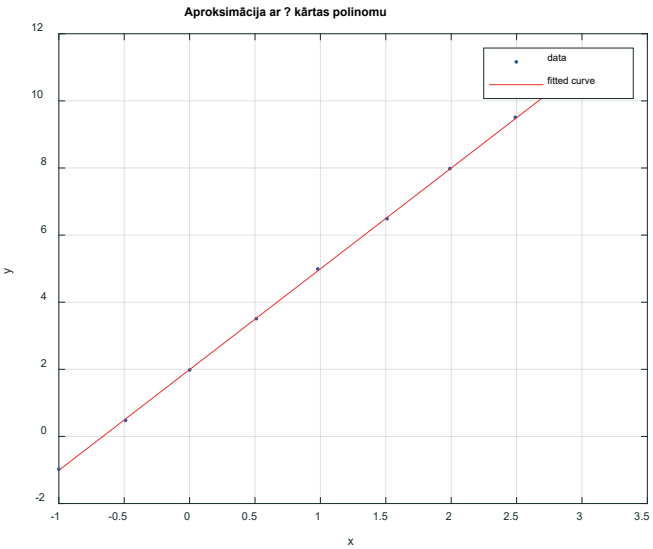
4.4. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar polinomiem (līdz ceturtās kārtas polinomiem ieskaitot).

x_i	−1.0	−0.49	0	0.51	0.98	1.51	1.99	2.49	3.01
y_i	−0.98	0.48	1.98	3.51	4.99	6.49	7.98	9.51	10.99



4.25. att. Dotās funkcijas grafiks.

```
reg =
  struct with fields:
          sse: 0.0108
      rsquare: 0.9999
          dfe: 7
    adjrsquare: 0.9999
         rmse: 0.0392
```

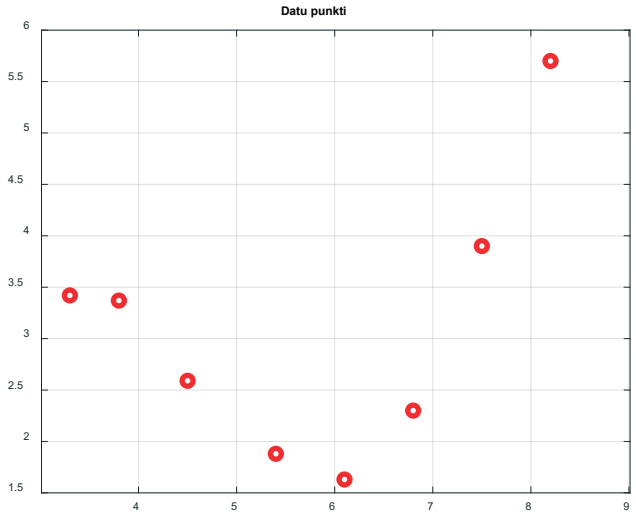


4.26. att. Aproksimācija ar polinomu.

```
Atbilde:
  Vislabākā aproksimācija ir … … … … …
```

4.5. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar funkcijām $y1 = a + b \cdot \sin x + c \cdot \cos x$, $y2 = a + b \cdot \sin x + c \cdot \cos x + d \cdot \sin 2x$, un salīdzināt rezultātus.

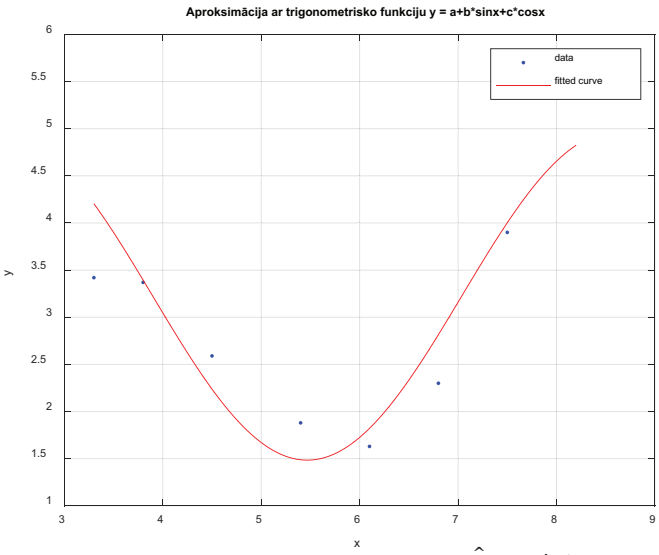
x_i	3.3	3.8	4.5	5.4	6.1	6.8	7.5	8.2
y_i	3.42	3.37	2.59	1.88	1.63	2.30	3.90	5.70



4.27. att. Dotās funkcijas grafiks.

```
p =
     Linear model:
     p(x) = a + b*sin(x) + c*cos(x)
     Coefficients (with 95% confidence bounds):
       a =       3.226   (2.652, 3.801)
       b =       1.268   (0.4486, 2.088)
       c =      -1.194   (-2.014, -0.3732)
```

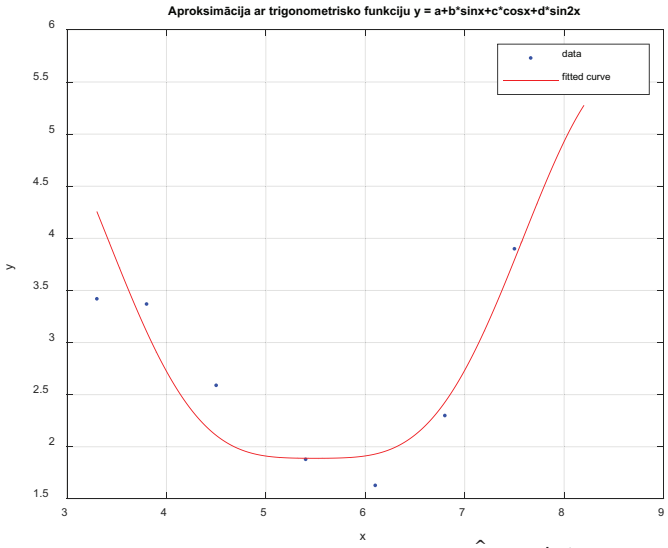
```
reg =
 struct with fields:
           sse: 1.9733
       rsquare: 0.8373
           dfe: 5
     adjrsquare: 0.7722
          rmse: 0.6282
```



4.28. att. Aproksimācija ar trigonometrisko funkciju $\hat{y} = a + b \sin x + c \cos x$.

```
p =
     Linear model:
     p(x) = a + b*sin(x) + c*cos(x) + d*sin(2*x)
     Coefficients (with 95% confidence bounds):
       a =       3.305   (2.722, 3.888)
       b =       1.318   (0.509, 2.126)
       c =      -1.317   (-2.155, -0.4788)
       d =     -0.4478   (-1.31, 0.4147)
```

```
reg =
  struct with fields:
           sse: 1.2987
       rsquare: 0.8929
           dfe: 4
     adjrsquare: 0.8126
          rmse: 0.5698
```



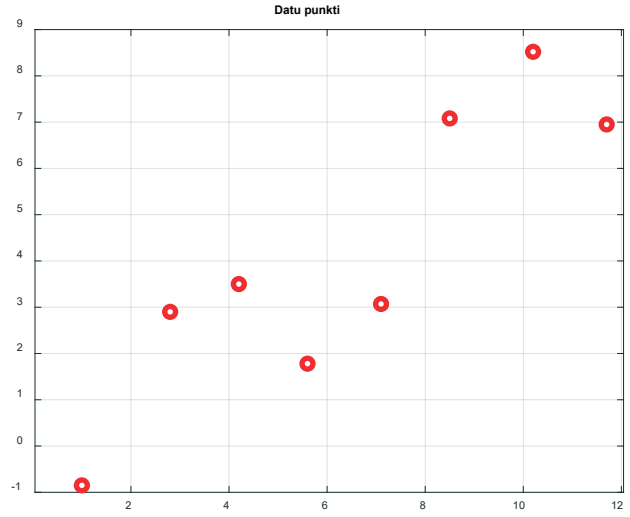
4.29. att. Aproksimācija ar trigonometrisko funkciju $\hat{y} = a + b\sin x + c\cos x + d\sin 2x$.

Salīdzinot R^2_{adj} un $RMSE$ vērtības, secinām, ka visprecīzākā aproksimācija ir ar trigonometrisko funkciju $\hat{y} = a + b\sin x + c\cos x + d\sin 2x$.

```
Atbilde:
  Vislabākā aproksimācija ir ar trigonometrisko funkciju
  ŷ=a+b*sinx+c*cosx+d*sin2x.
```

4.6. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar funkcijām $y1 = ax\ln x + b\cdot x\sin x + c\cdot x\cos x$, $y2 = ax + bx\ln x + c\cdot x\sin x + d\cdot x\cos x$, un salīdzināt rezultātus.

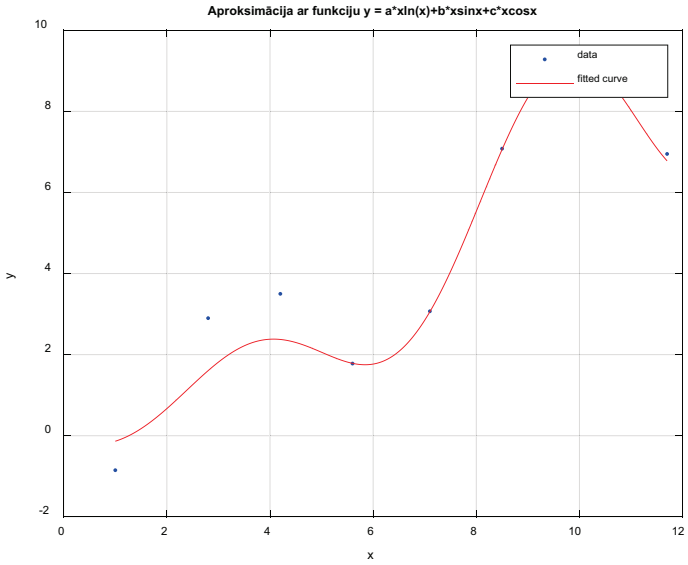
x_i	1.0	2.8	4.2	5.6	7.1	8.5	10.2	11.7
y_i	−0.85	2.9	3.5	1.78	3.07	7.08	8.52	6.95



4.30. att. Dotās funkcijas grafiks.

```
p =
     Linear model:
     p(x) = a*x*log(x) + b*x*sin(x) + c*x*cos(x)
     Coefficients (with 95% confidence bounds):
       a =       0.3091   (0.2538, 0.3643)
       b =      0.01038   (-0.1564, 0.1771)
       c =       -0.267   (-0.4345, -0.09944)
```

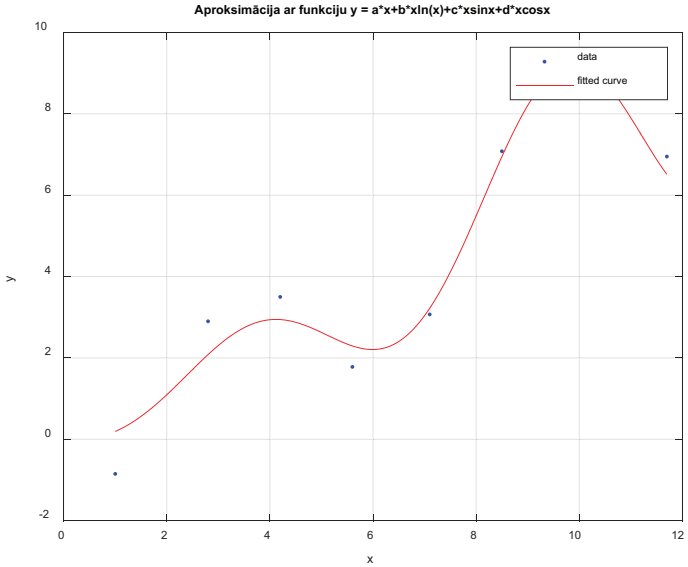
```
reg =
 struct with fields:
           sse: 3.9357
       rsquare: 0.9432
           dfe: 5
     adjrsquare: 0.9205
          rmse: 0.8872
```



4.31. att. Aproksimācija ar funkciju $\hat{y} = ax\ln x + bx\sin x + cx\cos x$.

```
p =
 Linear model:
 p(x) = a*x + b*x*log(x) + c*x*sin(x) + d*x*cos(x)
 Coefficients (with 95% confidence bounds):
   a =       0.3444   (-0.4549, 1.144)
   b =       0.1516   (-0.2183, 0.5215)
   c =     -0.01168   (-0.1919, 0.1685)
   d =       -0.262   (-0.436, -0.08802)
```

```
reg =
struct with fields:
        sse: 2.8986
    rsquare: 0.9582
        dfe: 4
 adjrsquare: 0.9268
       rmse: 0.8513
```



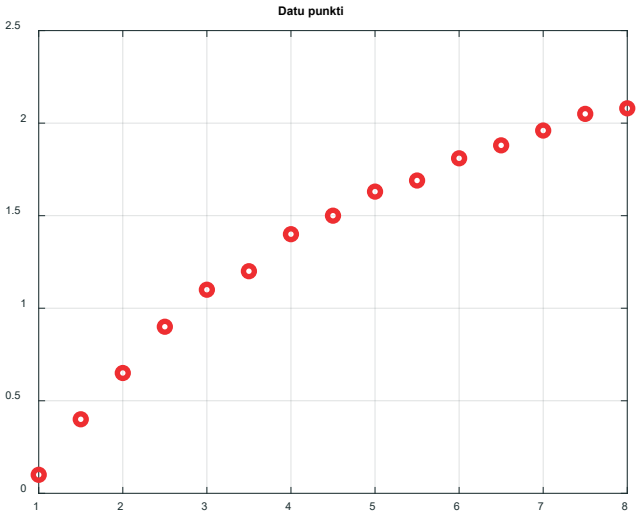
4.32. att. Aproksimācija ar funkciju $\hat{y} = ax + bx\ln x + cx\sin x + dx\cos x$.

Salīdzinot R^2_{adj} un *RMSE* vērtības, secinām, ka visprecīzākā aproksimācija ir ar funkciju $\hat{y} = ax + bx\ln x + cx\sin x + dx\cos x$.

```
Atbilde:
  Vislabākā aproksimācija ir ar funkciju
     y = a*x+b*xln(x)+c*xsinx+d*xcosx.
```

4.7. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar funkciju $y = a + b\ln x$.

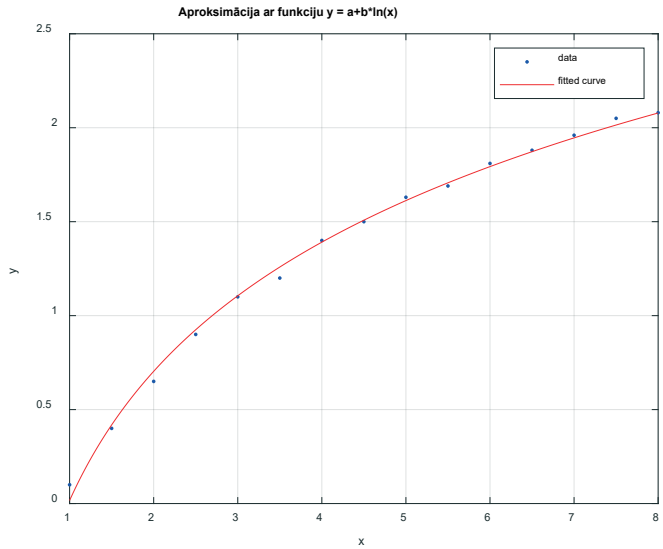
x_i	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0
y_i	0.1	0.4	0.65	0.9	1.1	1.2	1.4	1.5	1.63	1.69	1.81	1.88	1.96	2.05	2.08



4.33. att. Dotās funkcijas grafiks.

```
p =
     Linear model:
     p(x) = a + b*log(x)
     Coefficients (with 95% confidence bounds):
       a =     0.01632  (-0.03311, 0.06575)
       b =      0.9916  (0.9582, 1.025)
```

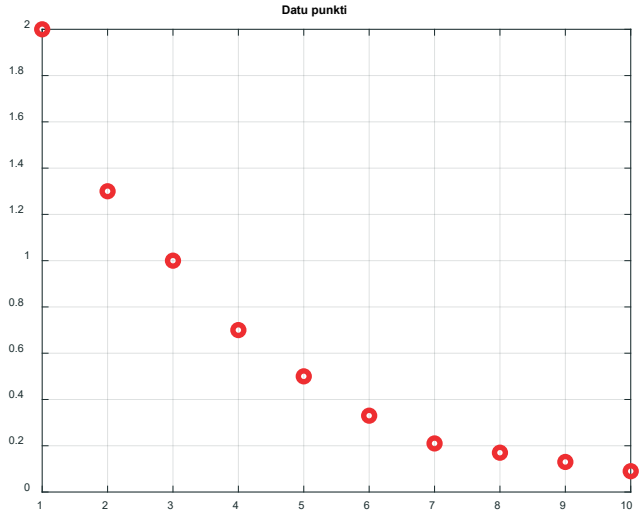
```
reg =
  struct with fields:
          sse: 0.0169
      rsquare: 0.9968
          dfe: 13
    adjrsquare: 0.9966
         rmse: 0.0360
```



4.34. att. Aproksimācija ar funkciju $\hat{y} = a + b\ln x$.

4.8. uzdevums. Aproksimēt tabulu, izmantojot aproksimāciju ar funkciju $y = a + \frac{b}{x} + \frac{c}{x^2}$.

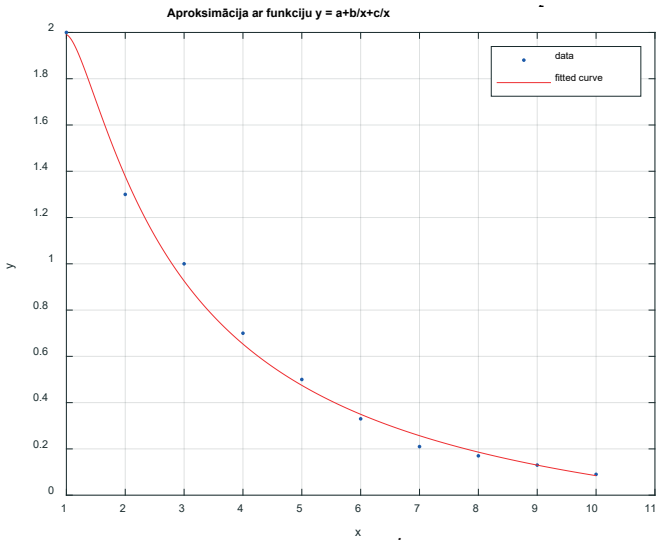
x_i	1	2	3	4	5	6	7	8	9	10
y_i	2.0	1.3	1.0	0.7	0.5	0.33	0.21	0.17	0.13	0.09



4.35. att. Dotās funkcijas grafiks.

```
p =
     Linear model:
     p(x) = a + b*1/x + c*1/x^2
     Coefficients (with 95% confidence bounds):
       a =     -0.3508  (-0.4577, -0.2438)
       b =       4.574  (3.944, 5.204)
       c =      -2.233  (-2.803, -1.662))
```

```
reg =
  struct with fields:
          sse: 0.0174
      rsquare: 0.9950
          dfe: 7
    adjrsquare: 0.9936
         rmse: 0.0499
```



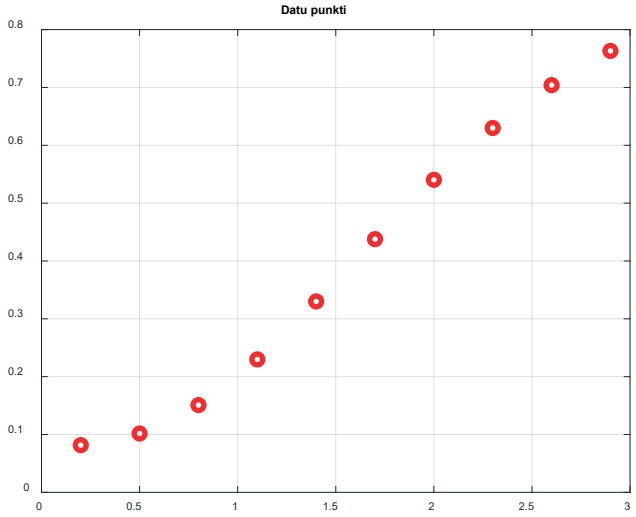
4.36. att. Aproksimācija ar funkciju $y = a + \frac{b}{x} + \frac{c}{x^2}$.

4.9. uzdevums. Sastādīt funkcijas $f(x)$ tabulu intervālā $[0.2; 2.9]$ ar soli 0.3. Aproksimēt tabulu ar formulu $y(x) = a + bx + c\cos x + d\sin 2x$. Aprēķināt aproksimācijas kļūdu punktā $x_0 = 2.22$ (kā starpību pēc moduļa starp funkciju $y(x)$ un $f(x)$ vērtībām punktā x_0).

$$f(x) = \frac{x^3 + \ln\left(x^2 + 2\right)}{x^3 + \sqrt{5x + 7} + 6}$$

```
xpoints =
    0.2000
    0.5000
    0.8000
    1.1000
    1.4000
    1.7000
    2.0000
    2.3000
    2.6000
    2.9000
```

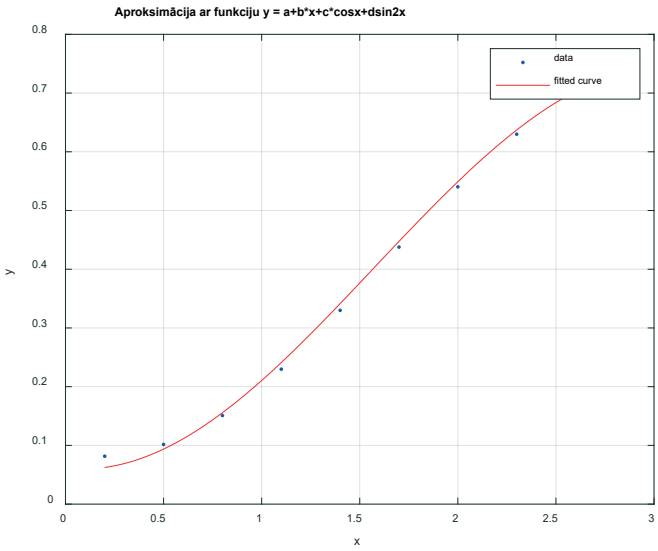
```
ypoints =
    0.0816
    0.1017
    0.1509
    0.2298
    0.3300
    0.4377
    0.5403
    0.6299
    0.7040
    0.7632
```



4.37. att. Dotās funkcijas grafiks.

```
y =
     Linear model:
     y(x) = a + b*x + c*cos(x) + d*sin(2*x)
     Coefficients (with 95% confidence bounds):
       a =      0.4757  (-1.666, 2.618)
       b =    -0.04713  (-1.411, 1.317)
       c =     -0.4147  (-2.452, 1.623)
       d =    0.006572  (-0.3608, 0.3739)
```

```
reg =
  struct with fields:
          sse: 0.0015
      rsquare: 0.9974
          dfe: 6
  adjrsquare: 0.9961
         rmse: 0.0160
```



4.38. att. Aproksimācija ar funkciju $y = a + bx + c\cos x + d\sin 2x$.

```
apr_error =
    0.0079
```

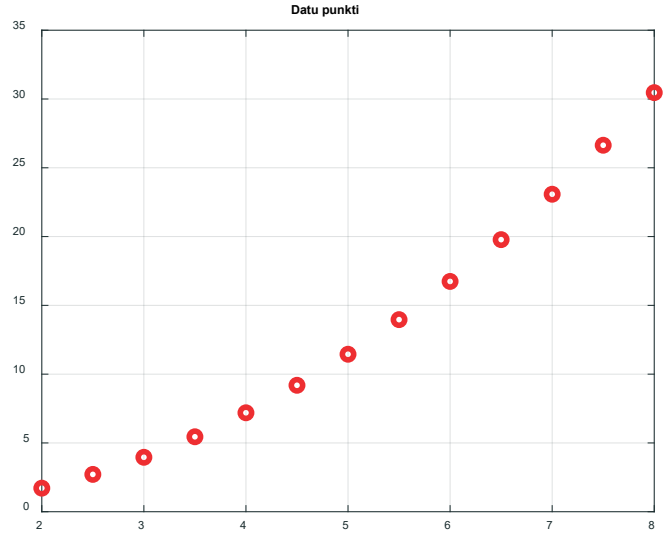
```
Atbilde:
 Aproksimācijas kļūda punktā (2.22) = 0.0079
```

4.10. uzdevums. Sastādīt funkcijas $f(x)$ vienādi attālinātu vērtību tabulu intervālā $[2; 8]$ ar soli $\Delta x = 0.5$. Interpolēt šo tabulu ar kubisko splainu (apzīmēsim šo funkciju ar $h(x)$). Aproksimēt šo tabulu ar otrās kārtas polinomu (apzīmēsim šo funkciju ar $g(x)$). Atrast $|h(x_1) - g(x_1)|$ punktā $x_1 = 6.1$

$$f(x) = \frac{2x^2}{4 + e^{-0.2x}}$$

```
xpoints =
    2.0000
    2.5000
    3.0000
    3.5000
    4.0000
    4.5000
    5.0000
    5.5000
    6.0000
    6.5000
    7.0000
    7.5000
    8.0000
```

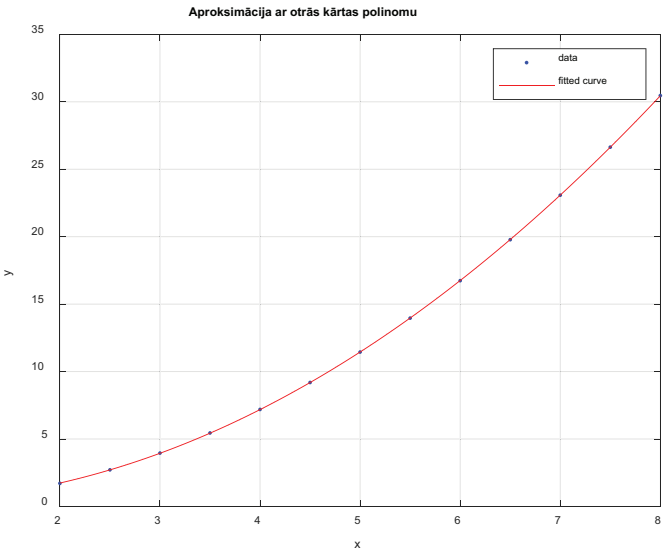
```
ypoints =
    1.7129
    2.7135
    3.9571
    5.4486
    7.1921
    9.1908
   11.4472
   13.9630
   16.7395
   19.7775
   23.0773
   26.6390
   30.4624
```



4.39. att. Dotās funkcijas grafiks.

```
g =
     Linear model Poly2:
     g(x) = p1*x^2 + p2*x + p3
     Coefficients (with 95% confidence bounds):
       p1 =       0.5157   (0.513, 0.5183)
       p2 =      -0.3715   (-0.3983, -0.3447)
       p3 =       0.4163   (0.3548, 0.4777)
```

```
reg =
  struct with fields:
            sse: 0.0018
        rsquare: 1.0000
            dfe: 10
     adjrsquare: 1.0000
           rmse: 0.0133
```



4.40. att. Aproksimācija ar otrās kārtas polinomu.

```
h_x0 =
   17.3262
```

```
Kļūda =
    0.0122
```

```
Atbilde:
   Kļūda punktā (6.1) = 0.0122
```

5. nodaļa

SKAITLISKĀ INTEGRĒŠANA

5.1. Integrēšanas jēdziens

Matemātikas kursā analizē dažādas noteiktā integrāļa aprēķināšanas metodes $\int_a^b f(x)\,\mathrm{d}x$:

- parciālo integrēšanu;
- substitūcijas metodi;
- sadalīšanu elementārās daļās.

Pieņemsim, ka $f(x)$ ir nepārtraukta funkcija intervālā $[a; b]$. Metodes izvēle ir atkarīga no funkcijas $f(x)$.

Ir arī daudz gadījumu, kad integrāli nevar novērtēt analītiski (šajos gadījumos integrālis ir jāaprēķina skaitliski).

Vienas no populārākajām metodēm pamatā ir intervāla $[a; b]$ sadale apakšintervālos un nepārtrauktas zemintegrāļa funkcijas aproksimēšana ar polinomu katrā apakšintervālā. Aplūkosim tikai gadījumu, kad visiem apakšintervāliem ir viens un tas pats garums (praksē lieto atšķirīga garuma intervālus).

Visvienkāršākā skaitliskās integrēšanas metode ir **taisnstūru metode**.

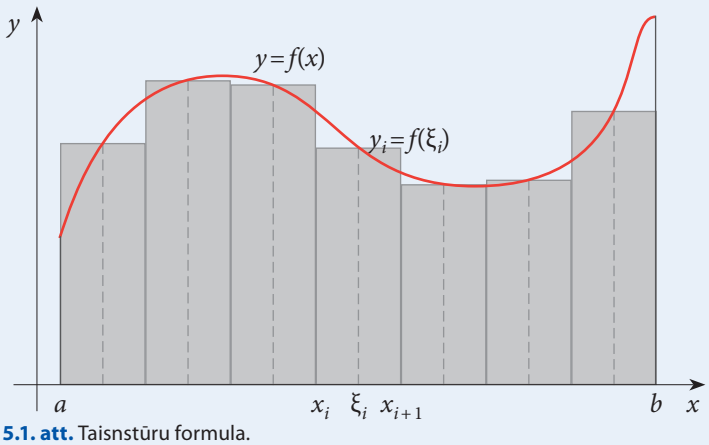
Pieņemsim, ka intervāls $[a; b]$ tiek sadalīts n apakšintervālos ar garumu h (sk. 5.1. attēlu).

Izvēlēsimies katrā intervālā (x_i, x_{i+1}) vidējo punktu ξ_i, aprēķināsim zemintegrāļa funkcijas vērtību punktā $x=\xi_i$ un reizinājumu $\Delta S_i = f(\xi_i)h$.

Ja $f(x)>0$ intervālā (a, b), tad ΔS_i ir taisnstūra laukums ar augstumu $f(\xi_i)$ un pamata garumu h. Saskaitot visas vērtības ΔS_i, iegūstam iesvītrotās figūras laukumu (sk. 5.1. attēlu).

Ja skaitlis h ir mazs, iesvītrotās figūras laukums ir tuvs līklīniju trapeces laukumam (laukums figūrai, kuru ierobežo sarkanā līnija 5.1. attēlā, Ox ass un divas taisnes $x=a$ un $x=b$).

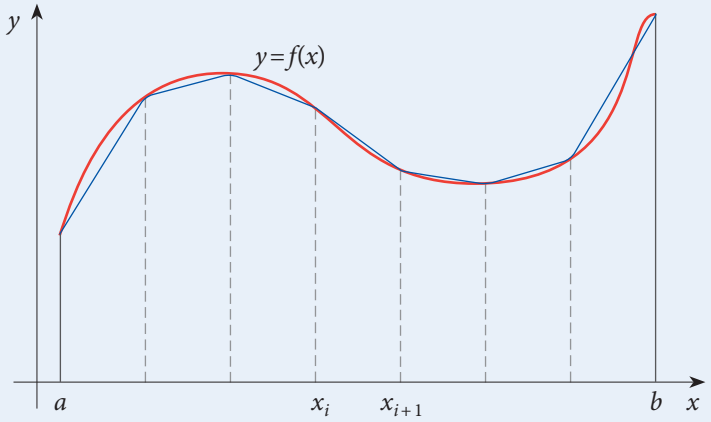
Atzīmēsim, ka taisnstūru formula ir balstīta uz zemintegrāļa funkcijas aproksimāciju katrā apakšintervālā ar konstanti (jeb ar nultās pakāpes polinomu).



5.1. att. Taisnstūru formula.

Citu skaitliskās integrēšanas formulu iegūst, aproksimējot zemintegrāļa funkciju katrā apakšintervālā ar pirmās kārtas polinomu. Tas nozīmē, ka mēs savienosim katra intervāla galapunktus ar taisnes nogriezni. Rezultātā līklīniju trapeces laukumu aproksimēsim ar trapeču laukumu summu.

Tā ir trapeču formula (sk. 5.2. attēlu). Šo ideju var vispārināt augstāku kārtu polinomiem.



5.2. att. Trapeču formula.

5.2. Noteiktā integrāļa aprēķini *MATLAB* vidē

MATLAB vidē ir divas iebūvētas komandas, kuras izmanto noteiktā integrāļa aprēķināšanai: **int** un **integral**. Pirmo komandu (**int**) parasti lieto, ja integrāli var aprēķināt analītiski. Otro komandu izmanto noteiktā integrāļa aprēķināšanai ar skaitliskām metodēm. Lietotājam ir jādefinē zemintegrāļa funkcija, integrēšanas robežas un (komandai **int**) integrēšanas mainīgais. Tas nozīmē, ka lietotājs var arī nezināt, kāda integrēšanas metode tiek izmantota integrāļa aprēķināšanai.

int(expr,var,a,b)	**expr** – simboliskā izteiksme, **var** – mainīgais, pēc kāda jāintegrē, **a** – apakšējā robeža, **b** – augšējā robeža.
integral(fun,a,b)	**fun** – arguments ir *function handle* (**jābūt vektorfunkcijai**) **a** – apakšējā robeža, b – augšējā robeža.

5.1. piemērs. Aprēķināt integrāli.

$$\int_0^2 \frac{\sin x}{x} \mathrm{e}^{-x^2}\, \mathrm{d}\, x$$

Atrisinājums.

Mēģināsim izmantot abas komandas: **int** un **integral**.

```
%% 5.1. piemērs. Komanda int un integral.
clc, clearvars,format compact
syms x
fun = @(x)sin(x)./x.*exp(-x.^2);
def_int = int(fun(x),0,2)
num_int = integral(fun,0,2)
```

```
def_int =
int((exp(-x^2)*sin(x))/x, x, 0, 2)
num_int =
    0.8161
```

Rezultāts rāda, ka integrālis nav aprēķināts, izmantojot komandu **int** – *MATLAB* vienkārši atkārto zemintegrāļa funkciju un nedod nekādas skaitliskās vērtības. Tas nozīmē, ka integrāli 5.1. piemērā nav iespējams novērtēt analītiski – integrālis nav elementārā funkcija.

Izmantojot otro komandu (**integral**), iegūstam integrāļa skaitlisko vērtību. Atzīmēsim, ka integrāļa aprēķināšanai ar komandu **int** jādefinē zemintegrāļa funkcija kā simboliska izteiksme, bet *function handle* jālieto komandā **integral**.

5.2. piemērs. Aprēķināt integrāli.

$$\int_0^5 \sin 1000x \, \mathrm{d}\, x$$

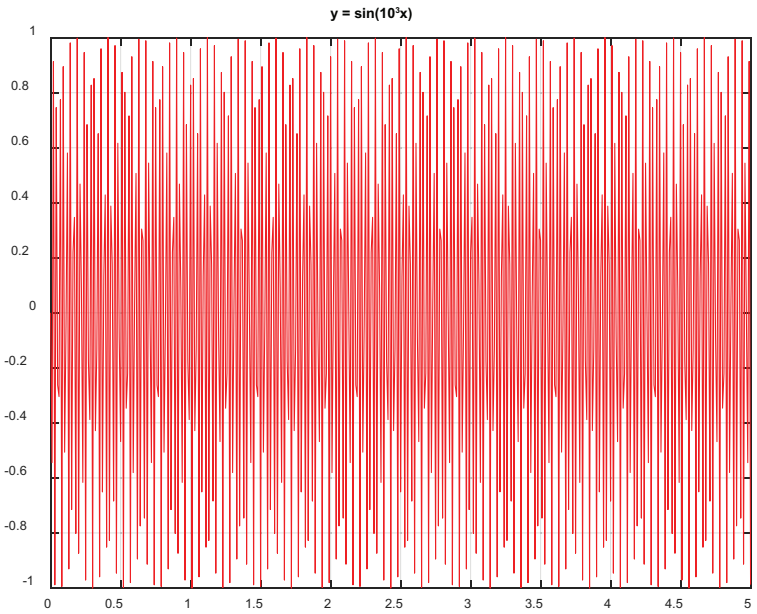
Atrisinājums.

Šis integrālis ir izvēlēts, lai ilustrētu iespējamās integrāļu skaitliskās aprēķināšanas problēmas. Integrāli 5.2. piemērā var aprēķināt analītiski. Tas dod iespēju salīdzināt integrāļa precīzo vērtību (kuru iegūst ar komandu **int**) ar skaitlisko vērtību (kuru iegūst ar komandu **integral**).

```
%% 5.2. piemērs. Noteiktā integrāļa aprēķināšana
clc, clearvars,format compact, close all, format longG
syms x, fun = @(x)sin(10^3.*x);
def_int = int(fun(x),0,5), skver_int = double(def_int)
num_int = integral(fun,0,5)
x_pr = 0:0.01:5;  plot(x_pr,fun(x_pr),'r')
grid on, title('y = sin(10^3x)')
format
```

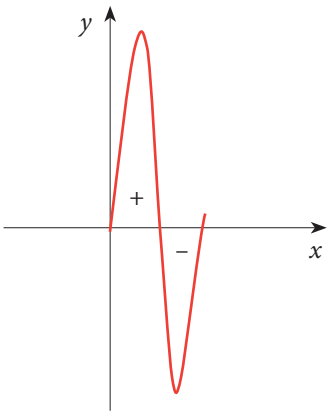
```
def_int =
sin(2500)^2/500
skver_int =
      0.000845331593819253
num_int =
      0.000845331593807798
```

Rezultāti rāda, ka *MATLAB* diezgan precīzi aprēķina šo integrāli ar skaitliskām metodēm. Divas atbildes nesakrīt (bet ir ļoti tuvas). Zemintegrāļa funkcijas grafiks ir parādīts 5.3. attēlā.



5.3. att. 5.2. piemēra zemintegrāļa funkcijas grafiks.

Funkcijas grafiks satur daudz oscilāciju. Tas nozīmē, ka ir jāaprēķina skaitliski apgabalu laukumi zem funkcijas $y = \sin 1000x$ grafika (ar plus zīmi, ja grafika daļa atrodas virs ass, un ar mīnus zīmi, ja grafika daļa atrodas zem Ox ass). Tas ir sarežģīts uzdevums.



5.4. att. Zemintegrāļa funkcijas grafiks vienā periodā.

5.4. attēlā ir palielinātā mērogā parādīts funkcijas $y=\sin 1000x$ grafiks vienā periodā. Problēmas rada tas, ka integrēšanas intervālā (0, 5) ir ļoti daudz oscilāciju, un ir jāaprēķina laukums figūrai virs Ox ass ar plus zīmi, un laukumu figūrai zem Ox ass ar mīnus zīmi (sk. 5.4. attēlu) katrā mazajā apakšintervālā (sk. 5.3. attēlu). Vienā periodā šo laukumu algebriskā summa ir nulle. Tādējādi nenulles ieguldījums integrālī ir saistīts ar pēdējo apakšintervālu (pie augšējās integrēšanas robežas $x=5$), kura garums nav precīzi vienāds ar vienu periodu funkcijai $y=\sin 1000x$.

5.3. piemērs. Aprēķināt integrāli.

$$\int_0^{\infty} \mathrm{e}^{-x}\,\frac{\sin(bx)}{\sinh x}\,\mathrm{d}\,x$$

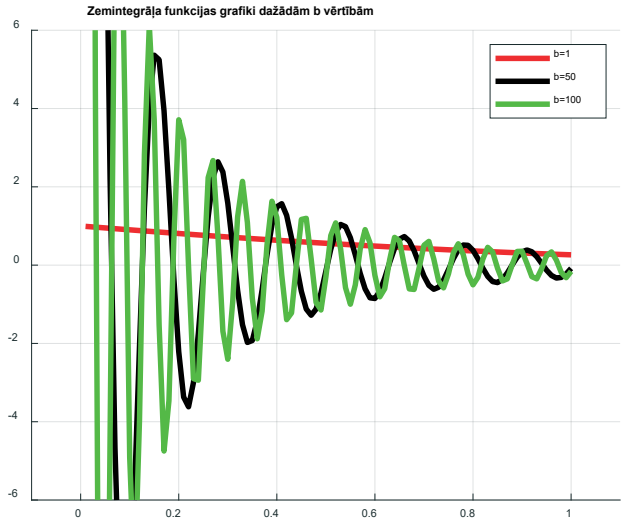
Salīdzināt rezultātu ar integrāļa precīzo vērtību:

$$\int_0^{\infty} \mathrm{e}^{-x}\,\frac{\sin(bx)}{\sinh x}\,\mathrm{d}\,x=\frac{\pi}{2}\coth\frac{b\pi}{2}-\frac{1}{b}$$

Integrāli aprēķināt, ja b = 1, b = 50 un b = 100.

Atrisinājums.

```
%% 5.3. piemērs. Noteiktā integrāļa aprēķināšana
clc, clearvars,format compact, close all, format longG
bmas = [1,50,100]; n = length(bmas); plot_col = ['r' 'k' 'g'];
x_pr = 0:0.01:1;
hold on
for i = 1:n
    b = bmas(i); fun = @(x)exp(-x).*sin(b*x)./sinh(x);
    num_int = integral(fun,0,inf);
    exact_val = pi/2*coth(b*pi/2)-1/b;
    difference(i) = exact_val - num_int;
    plot(x_pr,fun(x_pr),plot_col(i),'LineWidth',3)
end
hold off
title('Zemintegrāļa funkcijas grafiki dažādām b vērtībām')
legend('b=1','b=50','b=100'),ylim([-6,6]), xlim([-0.1 1.1])
grid on
```



5.5. att. Zemintegrāļa funkcijas grafiki dažādām b vērtībām.

5.5. attēlā redzams, ka zemintegrāļa funkcija ir oscilējoša ar dilstošu amplitūdu, turklāt oscilāciju frekvence pieaug lielākām b vērtībām.

```
% 5.3. piemēra turpinājums
disp('Rezultātu salīdzinājums')
disp(difference')
format
```

```
Rezultātu salīdzinājums
                          0
     -9.98918392447479e-09
      1.68858926841153e-09
```

Rezultāti rāda, ka integrāļa skaitlisko aprēķinu kļūda ir ļoti maza visām trim b vērtībām.

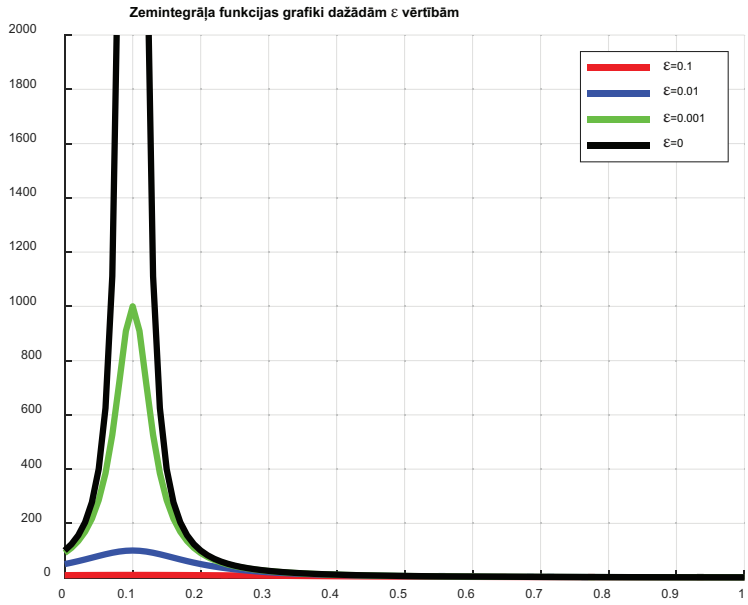
5.4. piemērs. Aprēķināt integrāli.

$$\int_0^1 \frac{1}{(x-0.1)^2+\varepsilon}\,\mathrm{d}x$$

ja $\varepsilon = 0.1;\ 0.01;\ 0.001$ un 0. Interpretēt rezultātus.

Atrisinājums.

```
%% 5.4. piemērs. Noteiktā integrāļa aprēķināšana
clc, clearvars,format compact, close all, format longG
emas = [0.1,0.01,0.001,0]; n = length(emas);
plot_col = ['r' 'b' 'g' 'k']; x_pr = 0:0.01:1;
hold on
for i = 1:n
    e = emas(i); fun = @(x)1./((x-0.1).^2+e);
    num_int(i) = integral(fun,0,1);
    plot(x_pr,fun(x_pr), plot_col(i),'LineWidth',3)
end
hold off
title('Zemintegrāļa funkcijas grafiki dažādām \epsilon vērtībām')
legend('\epsilon =0.1','\epsilon =0.01','\epsilon =0.001','\epsilon =0')
ylim([0,2000]), grid on
```



5.6. att. Zemintegrāļa funkcijas grafiki dažādām ε vērtībām.

Izmantojot skaitlisko integrēšanu (pirms aprēķiniem), ir ieteicams konstruēt zemintegrāļa funkcijas grafiku. Tas ir īpaši svarīgi gadījumos, kad zemintegrāļa funkcija satur pārtraukuma punktus. Atzīmēsim, ka 5.4. piemērā zemintegrāļa funkcijai ir otrā veida pārtraukuma punkts $x = 0.1$ pie parametra vērtības $\varepsilon = 0$ (pārbaudiet!). Var pierādīt, ka integrālis gadījumā $\varepsilon = 0$ diverģē.

```
% 5.4. piemēra turpinājums
fprintf(' Integrāļa vērtības pie dažādām ϵ vērtībām\n ')
disp(emas),disp(num_int)
format
```

```
          Integrāļa vērtības pie dažādām ϵ vērtībām
    0.1                0.01               0.001                0
4.86732744624566   22.4553726901845   88.5498876324137   709826458662326
```

Tā kā integrālis gadījumā $\varepsilon = 0$ diverģē, integrāļa vērtība iepriekšējā tabulā (iekrāsota ar sarkanu krāsu) nav pareiza (pareizā vērtība ir bezgalība).

5.5. piemērs. Konstruēt funkcijas $f(x)$ tabulu intervālā [0,5] ar soli 0.5.

$$f(x)=\int_0^x \sqrt{1+t^2}\cos t\,\mathrm{d}t$$

1. Aproksimēt tabulu ar ceturtās kārtas polinomu.
2. Uzzīmēt funkcijas $f(x)$ un aproksimējošā polinoma grafiku intervālā [0,5] ar soli 0.1.
3. Kāds ir aproksimējošā polinoma koeficients, ja mainīgais ir otrajā pakāpē?
4. Kāda ir šī polinoma vērtība punktā $x_0=1.2$?
5. Kāda ir funkcijas $f(x)$ vērtība punktā $x_0=1.2$?
6. Aprēķināt aproksimācijas kļūdu punktā kā starpību pēc moduļa starp aproksimējošā polinoma un funkcijas vērtībām punktā.

Atrisinājums.

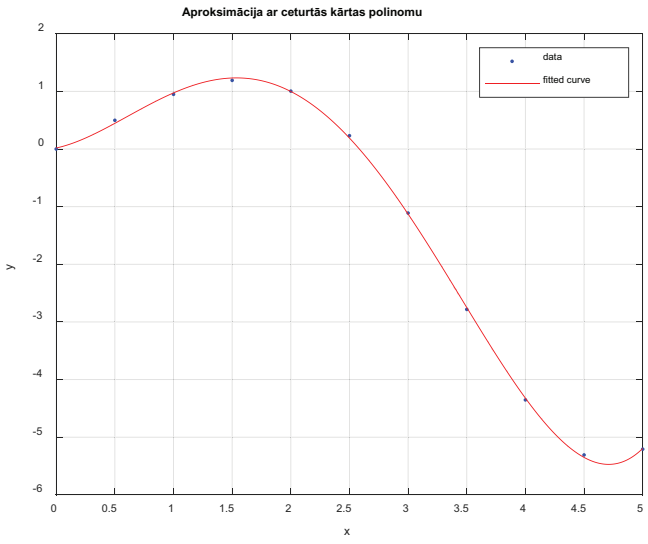
Vispirms sastādīsim integrāļa vērtību tabulu intervālā (0,5) ar soli 0.5 (sk. ciklu **for** skriptā). Tālāk izmantosim komandu **fit**.

```
%% 5.5. piemērs. Integrāļa aproksimācija ar ceturtās kārtas polinomiem
clc, clearvars,format compact, close all
syms t, y = @(t)sqrt(1+t.^2).*cos(t);
i = 0;
for x = 0:0.5:5
    i = i+1;
    f(i) = integral(y,0,x);
end
xpoints = (0:0.5:5)'; ypoints = f';
[p,reg] = fit(xpoints,ypoints,'poly4')
plot(p,xpoints,ypoints),grid on
title('Aproksimācija ar ceturtās kārtas polinomu')
```

```
p =
     Linear model Poly4:
     p(x) = p1*x^4 + p2*x^3 + p3*x^2 + p4*x + p5
     Coefficients (with 95% confidence bounds):
       p1 =       0.096  (0.08757, 0.1044)
       p2 =     -0.7794  (-0.8645, -0.6943)
       p3 =       1.201  (0.9243, 1.477)
       p4 =      0.4383  (0.1228, 0.7537)
       p5 =     0.01774  (-0.08468, 0.1202)
```

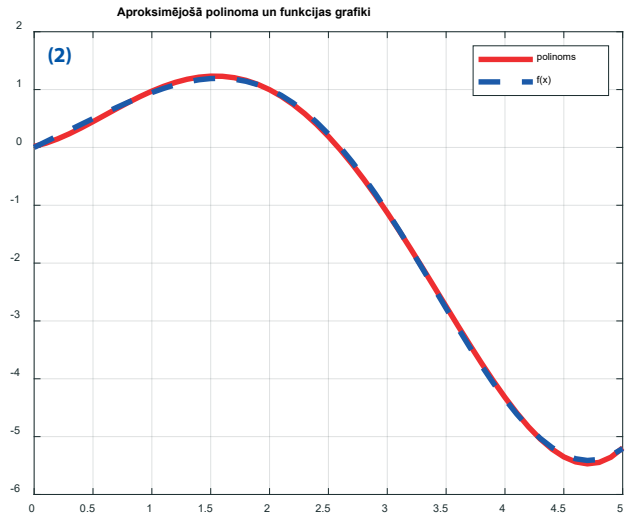
```
reg =
          sse: 0.0115
      rsquare: 0.9998
          dfe: 6
   adjrsquare: 0.9997
         rmse: 0.0437
```

Determinācijas koeficienta R^2 vērtība ir ļoti tuva 1. Tas nozīmē, ka aproksimācijas precizitāte ir diezgan augsta.



5.7. att. Aproksimācija ar ceturtās kārtas polinomu.

```
% 5.5. piemēra turpinājums. Uzzīmēt funkcijas f(x) un aproksimējošā
% polinoma grafikus intervālā [0,5] ar soli 0.1 (2)
i = 0;
for x = 0:0.1:5
    i = i+1;
    f(i) = integral(y,0,x);
end
figure
x_pr = 0:0.1:5; plot(x_pr,p(x_pr),'r-',x_pr,f,'b--','LineWidth',3)
legend('polinoms','f(x)'), grid on
title('Aproksimējošā polinoma un funkcijas grafiki')
```



5.8. att. Aproksimējošā polinoma un funkcijas grafiki.

```
% 5.5. piemēra turpinājums
% aproksimējošā polinoma koeficients, ja mainīgais ir otrajā pakāpē (3)
coef_x_2 = p.p3
% polinoma vērtība punktā x0=1.2  (4)
x0 = 1.2; pol_value = p(x0)
% funkcijas vērtība punktā x0=1.2  (5)
f_value = integral(y,0,x0)
% aproksimācijas kļūda punktā x1=3.2 (6)
app_error = abs(p(3.2)-integral(y,0,3.2))
```

```
coef_x_2 =
    1.2007
pol_value =
    1.1250
f_value =
    1.0826
app_error =
    0.0038
```

```
% 5.5. piemēra turpinājums
disp('Atbilde:')
disp([' 3) koeficients, ja mainīgais ir otrajā pakāpē = ',num2str(p.p3)])
disp([' 4) polinoma vērtība punktā (1.2) = ',num2str(pol_value)])
disp([' 5) funkcijas vērtība punktā (1.2) = ',num2str(f_value)])
disp([' 6) aproksimācijas kļūda punktā (3.2) = ',num2str(app_error)])
```

```
Atbilde:
 3) ja mainīgais ir otrajā pakāpē, koeficients = 1.2007
 4) polinoma vērtība punktā (1.2) = 1.125
 5) funkcijas vērtība punktā (1.2) = 1.0826
 6) aproksimācijas kļūda punktā (3.2) = 0.0037907
```

5.6. piemērs. Interpolēt doto integrāli ar trešās kārtas Ņūtona interpolācijas polinomu ar mezgliem punktos $x_0 = 0.2$; $x_1 = 0.6$; $x_2 = 1.2$; $x_3 = 2.0$ un uzzīmēt interpolācijas kļūdas grafiku.

$$f(x) = \int_0^x \frac{\sin t}{t}\,\mathrm{d}t$$

- Aprēķināt funkcijas $f(x)$ vērtības intervālā [0.2, 2] interpolācijas mezglos.
- Interpolēt funkciju $f(x)$ ar trešās kārtas interpolācijas polinomu, izmantojot interpolācijas mezglus $x_0 = 0.2$; $x_1 = 0.6$; $x_2 = 1.2$; $x_3 = 2.0$.
- Uzzīmēt funkcijas $f(x)$ un interpolācijas polinoma grafikus intervālā [0.2, 2] ar soli 0.1.
- Uzzīmēt interpolācijas kļūdas grafiku intervālā [0.2, 2].

```
%% 5.6. piemērs. Interpolēt integrāli ar trešās kārtas Ņūtona
% interpolācijas polinomu ar mezgliem punktos
clc, clearvars,format compact, close all
xnodes = [0.2,0.6,1.2,2.0];
syms t, y = @(t)sin(t)./t;
for i = 1:4                                   m = length(xnodes);
    ynodes(i) = integral(y,0,xnodes(i));
end
disp('Funkcijas vērtības interpolācijas mezglos:')
disp(ynodes)
```

```
Funkcijas vērtības interpolācijas mezglos:
    0.1996    0.5881    1.1080    1.6054
```

```
% 5.6. piemēra turpinājums. Interpolēt funkciju ar trešās kārtas
% interpolācijas polinomu
coef = ynodes;
for k = 2:4
    coef(k:4) = (coef(k:4) - coef(k-1:3))./...
                (xnodes(k:4) - xnodes(1:5-k));
end

syms x, pol = coef(4); % polinoma konstruēšana
for k = 3:-1:1
pol = pol*(x-xnodes(k))+coef(k);
end
polyn(x) = collect(pol);
coefpol = sym2poly(polyn) % polinoma koeficienti
fun_prob6(coefpol)        % polinoma drukāšana
```

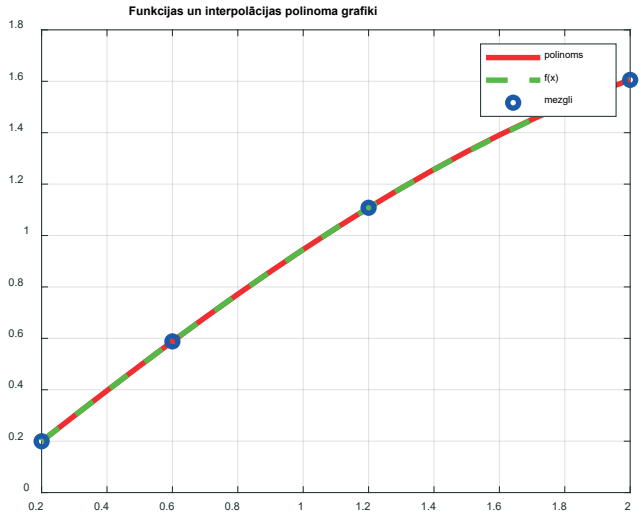
```
coefpol =
   -0.0389   -0.0272    1.0134   -0.0017
```

```
Atbilde.
  Ņutona trešās kārtas interpolācijas polinoms:
   -0.0389x^3 -0.0272x^2 +1.0134x^1 -0.0017x^0
```

Lai izdrukātu polinomu, izmanto ārējo funkciju **fun _ prob6**.

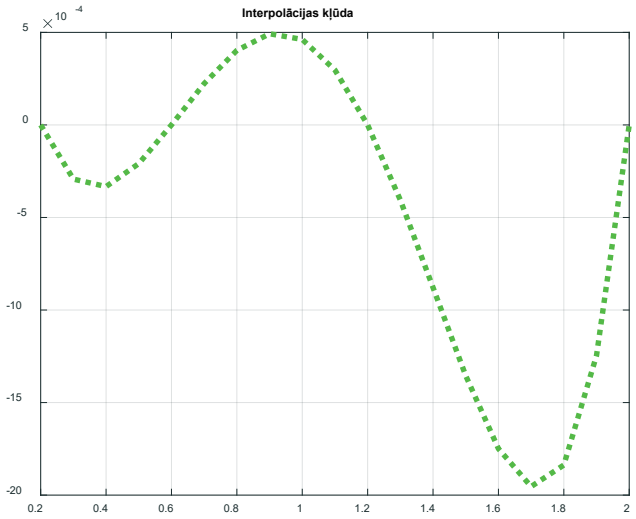
```
% ārējā funkcija (5.6. piemērs). Skaitliskā integrēšana polinoma drukāšana
function fun_prob6(koef)
   m = length(koef);
   fprintf('\n Atbilde. \n  Ņutona %.0f. kārtas',m-1)
   fprintf(' interpolācijas polinoms: \n  ')
   n = m-1;
   for i = 1:m
      if koef(i) < 0
        fprintf(' %.4fx^%.0f',koef(i),n)
      else
         fprintf(' +')
         fprintf('%.4fx^%.0f',koef(i),n)
      end
      n = n-1;
   end
   fprintf('\n')
end
```

```
% 5.6. piemēra turpinājums. Uzzīmēt funkcijas un interpolācijas
% polinoma grafikus intervālā [ 0.2,2 ] ar soli 0.1
i = 0;
for x = 0.2:0.1:2
    i = i+1; f(i) = integral(y,0,x);
end
x_pr = (0.2:0.1:2);
plot(x_pr,double(polyn(x_pr)),'r-',x_pr,f,'g--',xnodes,ynodes,'ob',...
     'LineWidth',3)
title('Funkcijas un interpolācijas polinoma grafiki')
legend('polinoms','f(x)','mezgli'), grid on
```



5.9. att. Funkcijas un interpolācijas polinoma grafiki.

```
% 5.6. piemēra turpinājums. Uzzīmēt interpolācijas kļūdas grafiku
figure
plot(x_pr,f-double(polyn(x_pr)),'g:','LineWidth',3)
title('Interpolācijas kļūda'), grid on
```



5.10. att. Interpolācijas kļūdas grafiks.

5.7. piemērs. Sastādīt funkcijas $f(x)$ tabulu intervālā [0.3, 1.8] ar soli 0.3. Aproksimēt šo tabulu ar otrās pakāpes polinomu. Kāda ir šī polinoma vērtība punktā $x_1 = 0.88$?

$$f(x) = \int_0^x \frac{1+\cos(2t^3)}{\sqrt{\sin(3t)+4t^2}+5}\,\mathrm{d}t$$

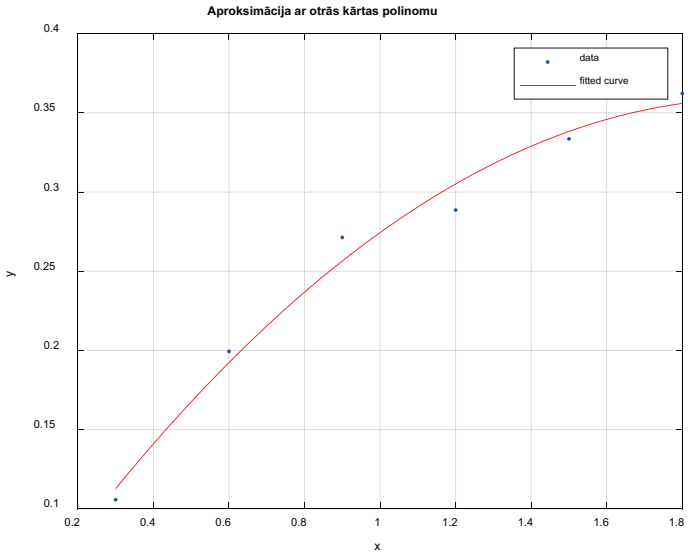
```
%% 5.7. piemērs. Integrāļa aproksimācija ar otrās kārtas polinomu
clc, clearvars,format compact, close all
syms t, y = @(t)(1+cos(2*t.^3))./(sqrt(sin(3*t)+4*t.^2)+5);
i = 0;
for x = 0.3:0.3:1.8
    i = i+1;
    f(i) = integral(y,0,x);
end
xpoints = (0.3:0.3:1.8)', ypoints = f'
[p,reg] = fit(xpoints,ypoints,'poly2')
plot(p,xpoints,ypoints),grid on
title('Aproksimācija ar otrās kārtas polinomu')
```

```
xpoints =
    0.3000
    0.6000
    0.9000
    1.2000
    1.5000
    1.8000
```

```
ypoints =
    0.1057
    0.1993
    0.2713
    0.2886
    0.3335
    0.3622
```

```
p =
   Linear model Poly2:
   p(x) = p1*x^2 + p2*x + p3
   Coefficients (with 95% confidence bounds):
     p1 =    -0.08599  (-0.1712, -0.0007744)
     p2 =      0.3427  (0.1599, 0.5255)
     p3 =     0.01767  (-0.06616, 0.1015)
```

```
reg =
struct with fields:
          sse: 6.5045e-04
      rsquare: 0.9853
          dfe: 3
  adjrsquare: 0.9755
         rmse: 0.0147
```



5.11. att. Aproksimācija ar otrās kārtas polinomu.

```
% 5.7. piemēra turpinājums
pol_ver = p(0.88)
fprintf('Polinoma vērtība punktā(0.88) = %.4f \n',pol_ver)
```

```
pol_ver =
    0.2526
```

```
Polinoma vērtība punktā(0.88) = 0.2526
```

5.8. piemērs. Interpolēt integrāli ar Ņūtona interpolācijas polinomu ar mezgliem punktos $x_0=0.3$; $x_1=0.7$; $x_2=1.2$; $x_3=1.6$; $x_4=2.1$. Kāda ir šī polinoma vērtība punktā $x=1.34$?

$$f(x)=\int_0^x \mathrm{e}^{-t^2}\sqrt{1+t^5}\,\mathrm{d}t$$

```
%% 5.8. piemērs. Interpolēt integrāli ar ceturtās kārtas Ņūtona
% interpolācijas polinomu ar mezgliem punktos
clc, clearvars,format compact, close all
xnodes = [0.3 0.7 1.2 1.6 2.1];
syms t, y = @(t)exp(-t.^2).*sqrt(1+t.^5);
for i = 1:5                                   m = length(xnodes);
    ynodes(i) = integral(y,0,xnodes(i));
end
disp('Funkcijas vērtības interpolācijas mezglos:')
disp(ynodes)
```

```
Funkcijas vērtības interpolācijas mezglos:
    0.2913    0.6074    0.8795    1.0213    1.1015
```

```
% 5.8. piemēra turpinājums. Interpolēt funkciju ar ceturtās kārtas
% interpolācijas polinomu
coef = ynodes;
for k = 2:5
    coef(k:5) = (coef(k:5) - coef(k-1:4))./...
                (xnodes(k:5) - xnodes(1:6-k));
end

syms x, pol = coef(5); % polinoma konstruēšana
for k = 4:-1:1
pol = pol*(x-xnodes(k))+coef(k);
end
polyn(x) = collect(pol);
coefpol = sym2poly(polyn) % polinoma koeficienti
fun_prob6(coefpol)        % polinoma drukāšana
```

```
coefpol =
   -0.0285    0.1564   -0.5194    1.2026   -0.0267
```

```
 Atbilde.
  Ņūtona ceturtās kārtas interpolācijas polinoms:
   -0.0285x^4 +0.1564x^3 -0.5194x^2 +1.2026x^1 -0.0267x^0
```

```
% 5.8. piemēra turpinājums
pol_ver = double(polyn(1.34))
fprintf('Polinoma vērtība punktā (1.34) = %.4f \n',pol_ver)
```

```
pol_ver =
    0.9364
```

```
Polinoma vērtība punktā (1.34) = 0.9364
```

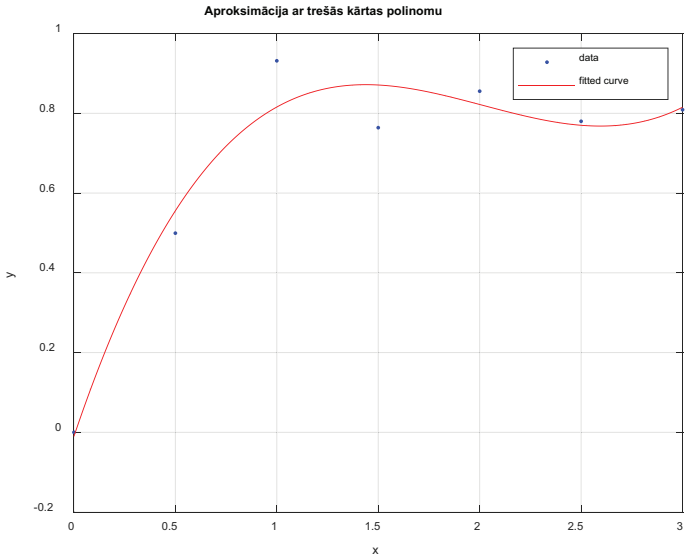
UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

5.1. uzdevums. Sastādīt funkcijas $f(x)$ tabulu intervālā [0, 3] ar soli 0.5. Aproksimēt šo tabulu ar trešās pakāpes polinomu.

$$f(x)=\int_0^x \cos t^3\,\mathrm{d}t$$

- Kāds ir aproksimējošā polinoma koeficients, ja mainīgais ir trešajā pakāpē?
- Kāda ir $f(x)$ precīza vērtība punktā $x_0=2.7$?
- Kāda ir aproksimējošā polinoma vērtība punktā x_0?

```
coef_x_3 =
    0.1344
pol_value =
    0.7708
f_value =
    0.8061
```



5.12. att. Aproksimācija ar trešās kārtas polinomu.

```
Atbilde:
 1) ja mainīgais ir trešajā pakāpē, koeficients = 0.13437
 2) funkcijas vērtība punktā (2.7) = 0.80607
 3) polinoma vērtība punktā (2.7) = 0.77077
```

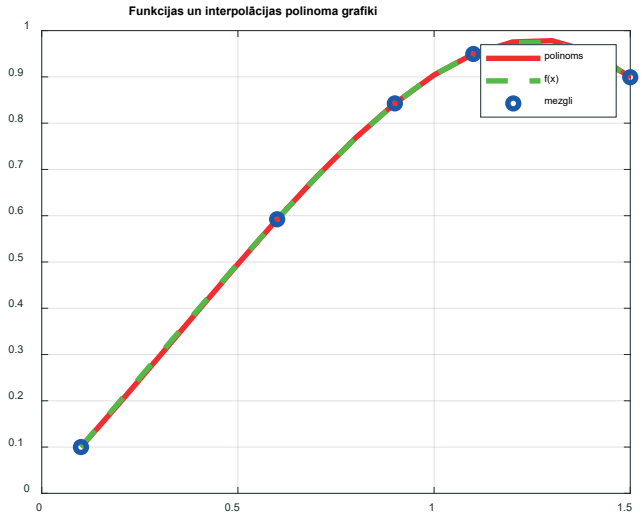
5.2. uzdevums. Interpolēt integrāli ar ceturtās kārtas Ņūtona interpolācijas polinomu ar mezgliem punktos $x_0=0.1$; $x_1=0.6$; $x_2=0.9$; $x_3=1.1$; $x_4=1.5$ un uzzīmēt interpolācijas kļūdas grafiku.

$$f(x)=\int_0^x \cos t^2\,\mathrm{d}t$$

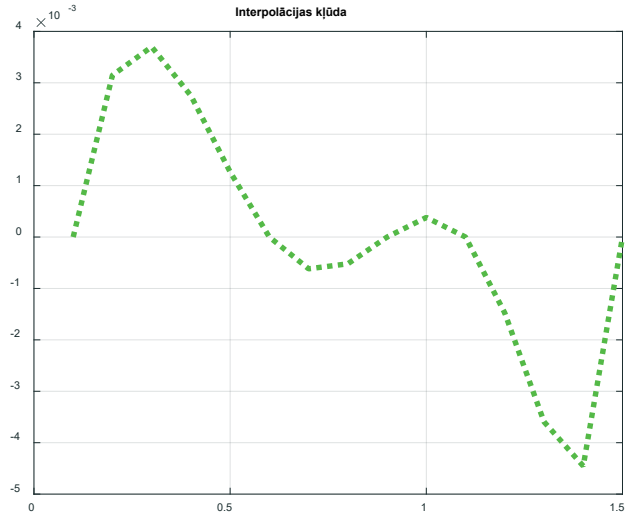
```
Funkcijas vērtības interpolācijas mezglos:
    0.1000    0.5923    0.8427    0.9495    0.8992
```

```
coefpol =
   -0.0989   -0.1477    0.2340    0.9098    0.0068
```

```
Atbilde.
  Ņūtona ceturtās kārtas interpolācijas polinoms:
   -0.0989x^4 -0.1477x^3 +0.2340x^2 +0.9098x^1 +0.0068x^0
```



5.13. att. Funkcijas un interpolācijas polinoma grafiki.



5.14. att. Interpolācijas kļūdas grafiks.

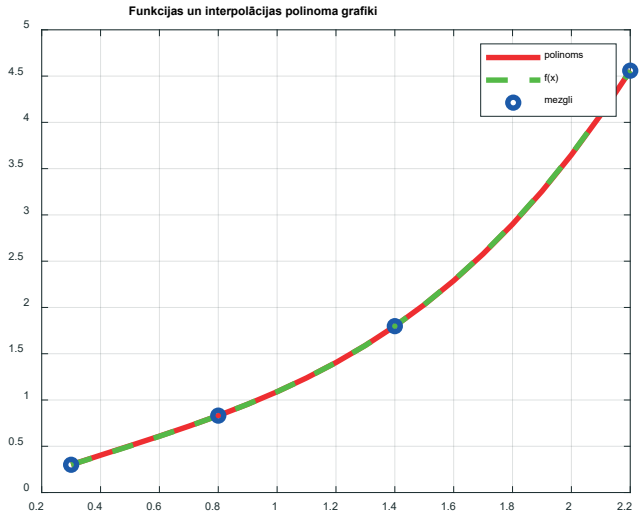
5.3. uzdevums. Interpolēt integrāli ar trešās kārtas Ņūtona interpolācijas polinomu ar mezgliem punktos $x_0 = 0.3$; $x_1 = 0.8$; $x_2 = 1.4$; $x_3 = 2.2$ un uzzīmēt interpolācijas kļūdas grafiku. Kāda ir šī polinoma vērtība punktā $x = 1.53$?

$$f(x) = \int_0^x \sqrt{1+t^4}\,\mathrm{d}t$$

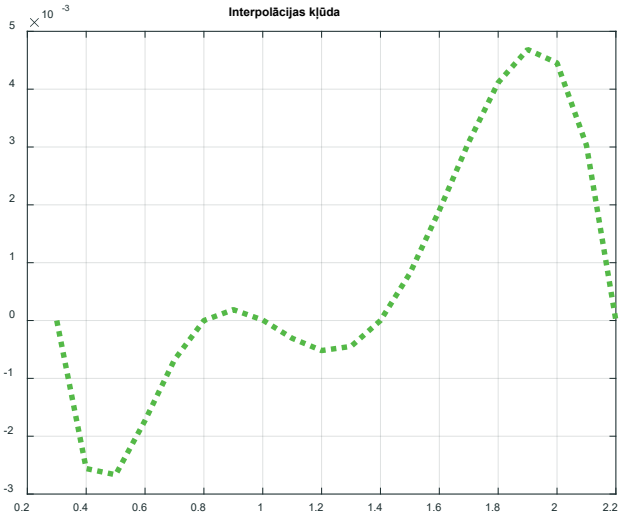
```
Funkcijas vērtības interpolācijas mezglos:
    0.3002    0.8311    1.7979    4.5586
```

```
coefpol =
    0.4286   -0.5719    1.2751   -0.0424
```

```
Atbilde.
  Ņūtona trešās kārtas interpolācijas polinoms:
   +0.4286x^3 -0.5719x^2 +1.2751x^1 -0.0424x^0
```



5.15. att. Funkcijas un interpolācijas polinoma grafiki.



5.16. att. Interpolācijas kļūdas grafiks.

```
pol_ver =
    2.1048
```

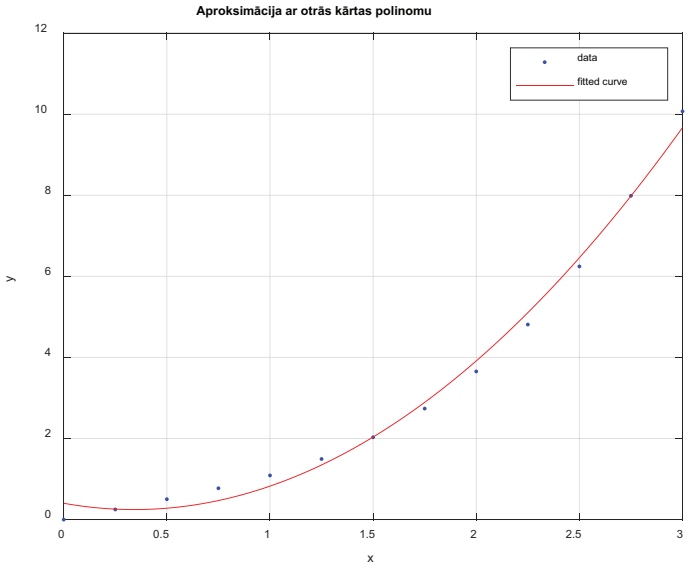
```
Polinoma vērtība punktā (1.53) = 2.1048
```

5.4. uzdevums. Sastādīt funkcijas tabulu intervālā [0, 3] ar soli 0.25. Aproksimēt šo tabulu ar otrās pakāpes polinomu.

$$f(x)=\int_0^x \sqrt{1+t^4}\,\mathrm{d}t$$

- Kāds ir aproksimējošā polinoma koeficients, ja mainīgais ir otrajā pakāpē?
- Aprēķināt aproksimācijas kļūdu punktā $x_0=1.65$ pēc formulas kā starpību pēc moduļa starp aproksimējošā polinoma un funkcijas vērtībām punktā x_0.

```
coef_x_2 =
    1.3332
aproks._kļūda =
    0.0978
```



5.17. att. Aproksimācija ar otrās kārtas polinomu.

```
Atbilde:
  1) ja mainīgais ir otrajā pakāpē, koeficients s = 1.3332
  2) aproksimācijas kļūda punktā (1.65) = 0.097822
```

5.5. uzdevums. Sastādīt funkcijas $f(x)$ tabulu intervālā $[0.2;\ 1.7]$ ar soli 0.15. Aproksimēt šo tabulu ar trešās pakāpes polinomu. Kāda ir šī polinoma vērtība punktā $x_1 = 1.32$?

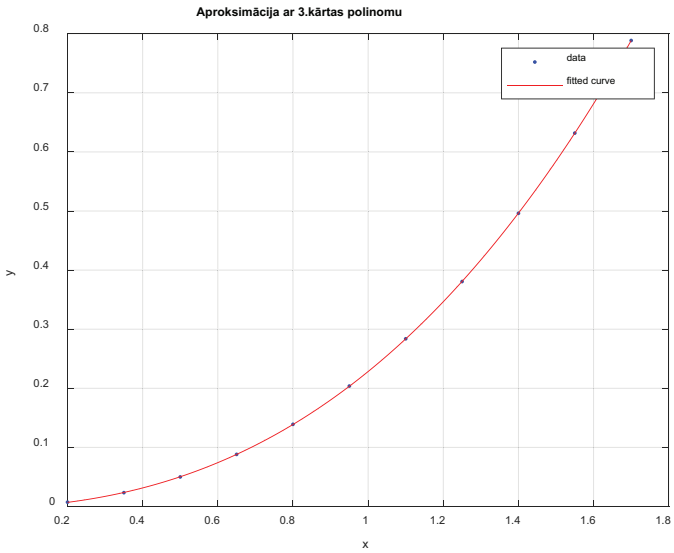
$$f(x) = \int_0^x \frac{t + t^2}{1 + t + \sqrt{\cos t^2 + 3}}\,\mathrm{d}t$$

```
xpoints =
    0.2000
    0.3500
    0.5000
    0.6500
    0.8000
    0.9500
    1.1000
    1.2500
    1.4000
    1.5500
    1.7000
```

```
ypoints =
    0.0072
    0.0233
    0.0499
    0.0881
    0.1390
    0.2038
    0.2838
    0.3806
    0.4961
    0.6319
    0.7886
```

```
p =
  Linear model Poly3:
  p(x) = p1*x^3 + p2*x^2 + p3*x + p4
  Coefficients (with 95% confidence bounds):
    p1 =     0.07019  (0.06739, 0.073)
    p2 =      0.1453  (0.1373, 0.1534)
    p3 =     0.01536  (0.008537, 0.02219)
    p4 =    -0.00252  (-0.004139, -0.0009016)
```

```
reg =
struct with fields:
          sse: 6.9149e-07
      rsquare: 1.0000
          dfe: 7
   adjrsquare: 1.0000
         rmse: 3.1430e-04
```



5.18. att. Aproksimācija ar trešāskārtas polinomu.

```
pol_ver =
    0.4324
```

```
Polinoma vērtība punktā (1.32) = 0.4324
```

5.6. uzdevums. Interpolēt integrāli ar Ņūtona interpolācijas polinomu ar mezgliem punktos $x_0 = 0.2$; $x_1 = 0.6$; $x_2 = 1.3$; $x_3 = 2.0$. Kāda ir šī polinoma vērtība punktā $x = 0.36$?

$$f(x) = \int_0^x \frac{3+\sqrt{t}}{2+\cos t + \sqrt{3+\sin 2t + t^2}}\,\mathrm{d}t$$

```
Funkcijas vērtības interpolācijas mezglos:
    0.1378    0.4349    1.0188    1.7183
```

```
coefpol =
    0.0193    0.0426    0.6985   -0.003
```

```
Atbilde.
  Ņūtona trešās kārtas interpolācijas polinoms:
   +0.0193x^3 +0.0426x^2 +0.6985x^1 -0.0037x^0
```

```
pol_ver =
    0.2542
```

```
Polinoma vērtība punktā (0.36) = 0.2542
```

6. nodaļa

NELINEĀRIE VIENĀDOJUMI UN TO SISTĒMAS

6.1. Nelineāra vienādojuma saknes

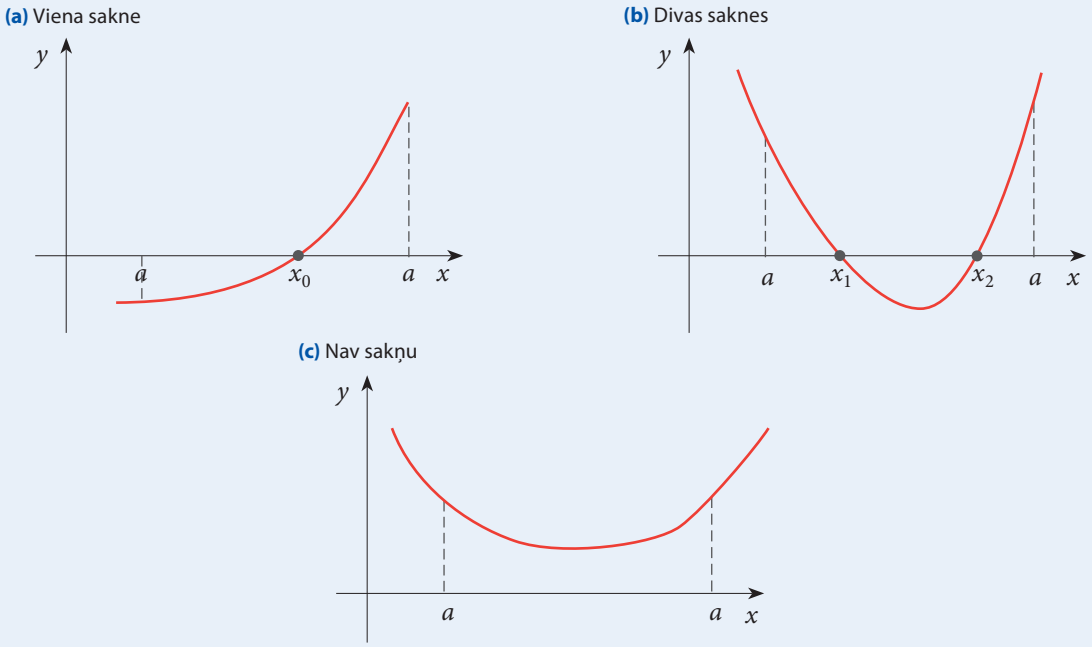
Dažos uzdevumos inženierzinātnēs ir jāaprēķina nelineārā vienādojuma saknes:

$$f(x) = 0. \tag{6.1}$$

Pieņemsim, ka funkcija $f(x)$ vienādojumā (6.1) ir nepārtraukta intervālā (a, b), kur atrodas sakne. Atkarībā no uzdevuma tipa vienādojumam (6.1) var būt galīgs vai bezgalīgs sakņu skaits. Saknes var būt reāli vai kompleksi skaitļi. Aprēķināt sakni ar formulu var tikai atsevišķos gadījumos (viens piemērs ir kvadrātiskā formula otrās pakāpes polinoma sakņu aprēķināšanai). Lai aprēķinātu sakni ar noteiktu precizitāti, praksē parasti lieto skaitliskās metodes.

Risinot vienādojumu (6.1), pirmais solis ir noskaidrot reālu sakņu skaitu un atrast sakņu tuvinātās vērtības. To var izdarīt, uzzīmējot funkcijas $f(x)$ grafiku pietiekami plašā intervālā. Grafiki 6.1. attēlā rāda dažus variantus (viena sakne, divas saknes vai nav sakņu intervālā (a, b)).

Ja vērtībām $f(a)$ un $f(b)$ ir atšķirīgas zīmes intervālā (a, b), tad šajā intervālā eksistē vismaz viena sakne. Mainot zīmēšanas diapazonu, var precīzāk atrast sakni.



6.1. att. Funkcijas $f(x)$ grafiks intervālā (a, b).

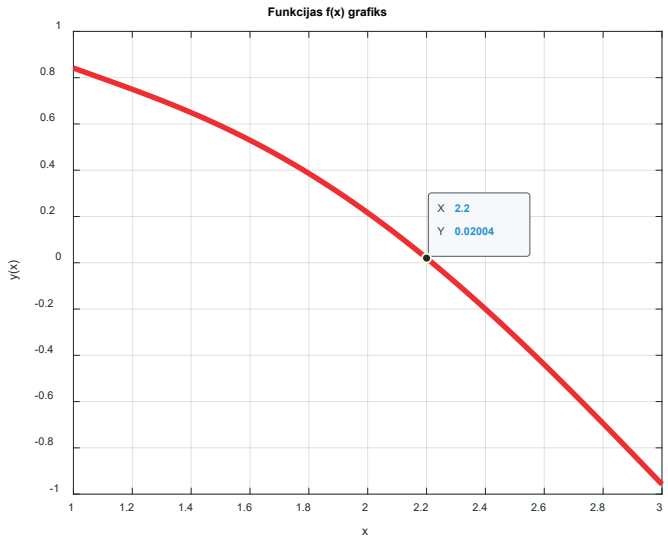
6.2. Aprēķini *MATLAB* vidē. Nelineāra vienādojuma risināšana

6.1. piemērs. Atrast sākuma tuvinājumu vienādojuma $\sin x - \ln x = 0$ saknei.

Atrisinājums.

Lai atrastu sākuma tuvinājumu vienādojuma saknei, konstruēsim funkcijas $f(x) = \sin x - \ln x$ grafiku un atradīsim grafika krustpunktus (ja tādi ir) ar Ox asi, ņemot vērā funkcijas $f(x)$ definīcijas apgabalu. Šajā piemērā funkcija ir definēta apgabalā $x > 0$.

```
%% 6.1. piemērs. Sākuma tuvinājums
clc, clearvars,format compact
y = @(x)sin(x)-log(x);
x_pr = 1:0.01:3; plot(x_pr,y(x_pr),'r','LineWidth',3)
xlabel('x'),ylabel('y(x)')
title('Funkcijas f(x) grafiks'),grid on
```



6.2. att. Funkcijas $f(x) = \sin x - \ln x$ grafiks.

Grafikā redzams, ka saknes tuvinātā vērtība ir apmēram 2.2. Atzīmēsim, ka grafiks ir konstruēts intervālā (1, 3). Sākumā ir ieteicams konstruēt grafiku pēc iespējas plašākā intervālā un samazināt intervālu, kad ir atrasts intervāls, kurā atrodas sakne.

Kas jādara, lai aprēķinātu saknes vērtību ar noteiktu precizitāti? Turpmāk ir apskatītas dažādas labāko saknes tuvinājumu atrašanas metodes.

6.3. Aprēķini *MATLAB* vidē. Bisekcijas metode

Pieņemsim, ka ir jau atrasts intervāls (x_l, x_u), kurā funkcija $f(x)$ maina zīmi.

Tādējādi $f(x_l)f(x_u) < 0$.

Visvienkāršākā metode saknes aprēķināšanai intervālā (x_l, x_u) ir **bisekcijas metode**.

Definēsim intervāla vidējo punktu $x_{mid} = (x_l + x_u)/2$ un aprēķināsim funkcijas vērtību iegūtajā punktā, t. i., $f(x_{mid})$. Ja vērtību $f(x_l)$ un $f(x_{mid})$ zīmes sakrīt, tad sakne atrodas intervālā (x_{mid}, x_u).

Ja vērtību $f(x_l)$ un $f(x_{mid})$ zīmes nesakrīt, tad sakne atrodas intervālā (x_l, x_{mid}). Tas nozīmē, ka nenoteiktības intervāls (kur atrodas sakne) ir samazināts uz pusi.

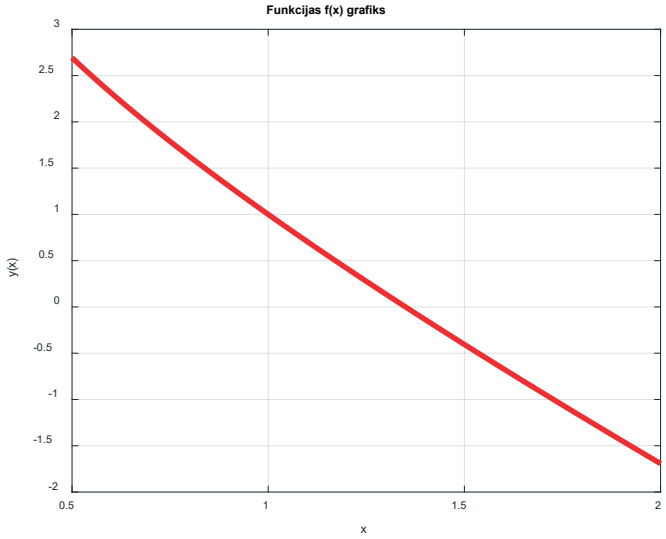
Atkārtojot šo procedūru vairākas reizes, var atrast visprecīzāko saknes aproksimāciju. Aprēķinus ilustrēsim 6.2. piemērā.

6.2. piemērs. Atrast vienādojuma $3 - 2x = \ln x$ sakni ar bisekcijas metodi.

Atrisinājums.

Vispirms uzzīmēsim funkcijas $f(x) = 3 - 2x - \ln x$ grafiku un atradīsim intervālu, kurā funkcija maina zīmi.

```
%% 6.2. piemērs. Bisekcijas metode
clc, clearvars,format compact
y = @(x)3-2*x-log(x);
x_pr = 0.5:0.01:2;  plot(x_pr,y(x_pr),'r','LineWidth',3)
xlabel('x'),ylabel('y(x)')
title('Funkcijas f(x) grafiks'),grid on
```



6.3. att. Funkcijas $f(x) = 3 - 2x - \ln x$ grafiks.

Grafikā ir redzams, ka funkcija maina zīmi intervālā (1, 1.5). Pieņemsim, ka $x_l = 1$; $x_u = 2$ (šeit būtu loģiski pieņemt $x_l = 1$; $x_u = 1.5$, bet plašāks intervāls izvēlēts, lai ilustrētu konverģences ātrumu).

Izmantojot bisekcijas metodi, iegūstam 6.1. tabulu.

6.1. tabula. Saknes aproksimācija 6.2. piemērā.

	x_l	x_u	x_{mid}	$f(x_{mid})$	$\lvert f(x_{mid})\rvert \le \varepsilon$?
1	1	2	1.5	−0.40547	Nē
2	1	1.5	1.25	0.27686	Nē
3	1.25	1.5	1.375	−0.06845	Nē
4	1.25	1.375	1.3125	0.10307	Nē
5	1.3125	1.375	1.34375	0.01704	Nē
6	1.34375	1.375	1.359375	−0.02578	Nē
7	1.34375	1.359375	1.351563	−0.00439	Nē
8	1.34375	1.351563	1.347657	0.006318	Nē
9	1.347657	1.351563	1.34961	0.000964	Jā

Katrā solī aprēķinājām vidējo punktu x_{mid} un funkcijas vērtību $f(x_{mid})$ intervālā (x_l, x_u) un pārbaudījām nosacījumu $|f(x_{mid})| < \varepsilon$.

Ir pieņemts, ka $\varepsilon = 0.001$. Tabulā redzams, ka pēc deviņām iterācijām sakne atrodas intervālā (1.347657, 1.351563). Tādējādi saknes tuvinātā vērtība ir 1.35. Atzīmēsim, ka iegūtajā atbildē tikai divi cipari aiz komata ir pareizi (pēc noapaļošanas).

Lai paaugstinātu aprēķinu precizitāti, var arī pārbaudīt nosacījumu $|b - a| < \varepsilon$ katrā iterācijas solī.

6.2. piemēra *MATLAB* skripts ir parādīts turpmāk. Ir pieņemts, ka $x_l = 1$; $x_u = 1.5$.

```
x =
    1.3496
```

```
% 6.2. piemēra turpinājums
epsi = 10^(-3); a = 1; b = 1.5;
while (b - a) > epsi
     m = (b+a)/2; ym = y(m);
     if ym > 0
        a = m;
     else
        b = m;
     end
end
x = a
```

```
% 6.2. piemēra turpinājums
disp('Atbilde:')
disp(['vienādojuma sakne x = ',num2str(x)])
```

```
Atbilde:
 vienādojuma sakne x = 1.3496
```

6.4. Ņūtona metode

Vislielākā bisekcijas metodes problēma ir saistīta ar zemo konverģences ātrumu. Iepriekšējā piemērā, lai aprēķinātu sakni ar precizitāti 10^{-2}, ir nepieciešamas deviņas iterācijas.

Kā uzlabot konverģences ātrumu?

Viena no populārākajām metodēm, kas ļauj uzlabot konverģences ātrumu, ir **Ņūtona metode**.

Pieņemsim, ka x_0 ir sākuma tuvinājums saknei. Apzīmēsim saknes precīzo vērtību ar x. Izvirzīsim funkciju $f(x)$ Teilora rindā punkta x_0 apkārtnē:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \dots \tag{6.2}$$

Ja punkts x_0 atrodas mazā punkta x apkārtnē, tad funkciju $f(x)$ var aproksimēt ar formulu:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \tag{6.3}$$

Aizstājot formulas (6.3) kreiso pusi ar nulli (atgādināsim, ka x ir vienādojuma (6.1) sakne) un risinot iegūto vienādojumu, iegūstam nākamo saknes aproksimāciju

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{6.4}$$

Turpinot šo procesu, iegūstam

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(n)}, n = 0,1,\dots \tag{6.5}$$

Formulu (6.5) sauc par **Ņūtona formulu**.

Ņūtona metodei ir vienkārša ģeometriskā interpretācija.

Tā kā (6.3) ir funkcijas $f(x)$ lineārā aproksimācija, formula (6.4) nosaka pieskares krustpunktu ar Ox asi.

Pieskare novilkta caur punktu $(x_0, f(x_0))$, turklāt pieskares virziena koeficients ir $f'(x_0)$.

Punkts x_1 ir pieskares krustpunkts ar Ox asi.

Piezīmes.

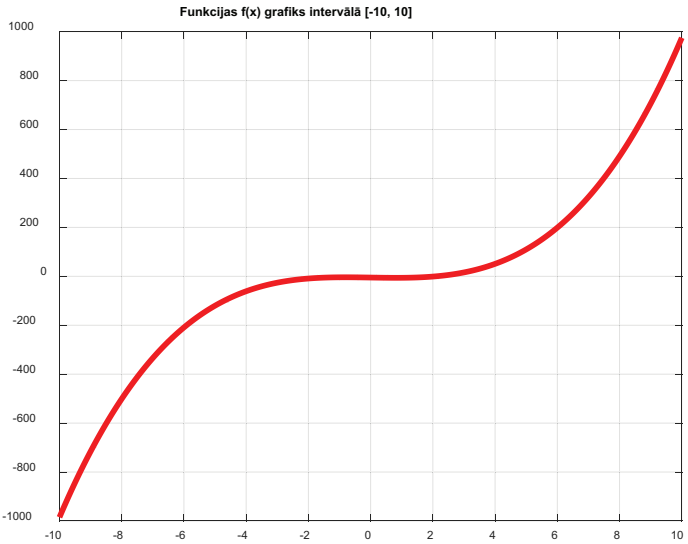
1. Ņūtona metodei ir nepieciešams atvasinājuma aprēķins. Ja funkcija $f(x)$ ir dota ar formulu, tad parasti nav problēmu aprēķināt funkcijas atvasinājumu. Ja atvasinājumu $f'(x)$ nav iespējams aprēķināt analītiski, tad jāizmanto formulas (6.5) modifikācijas.
2. Ņūtona metode konverģē ātri, ja sākuma tuvinājums atrodas tuvu saknei. No otras puses, metode var diverģēt, ja sākuma tuvinājums ir tālu no saknes.
3. Ņūtona metodi var izmantot komplekso sakņu aprēķināšanai.

6.5. Aprēķini *MATLAB* vidē. Ņūtona metode

6.3. piemērs. Atrast vienādojuma $x^3 - 2x - 5 = 0$ sakni ar precizitāti 0.00001, izmantojot Ņūtona metodi.

Atrisinājums.

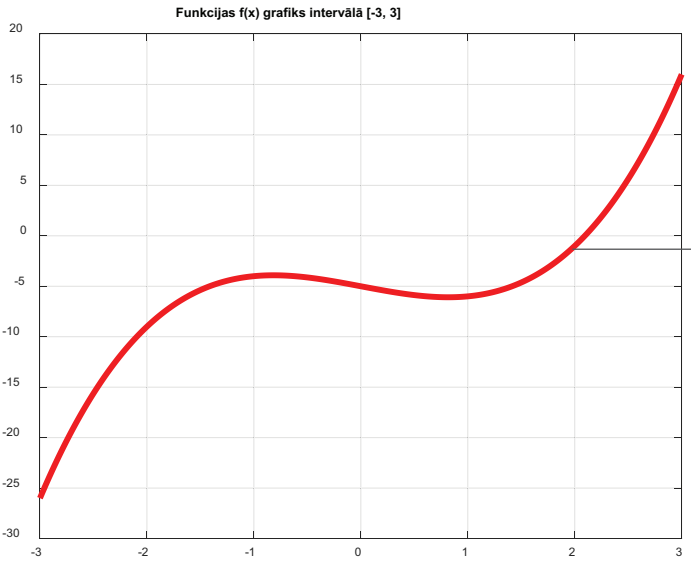
```
%% 6.3. piemērs. Ņūtona metode
clc, clearvars,format compact, close all, format longG
f = @(x)x.^3-2*x-5;
x_pr = -10:0.01:10; plot(x_pr,f(x_pr),'r','LineWidth',3)
grid on, title('Funkcijas f(x) grafiks intervālā [-10, 10]')
```



6.5. att. Funkcijas $f(x) = x^3 - 2x - 5$ grafiks intervālā (–10, 10).

Izmantosim šaurāko intervālu, lai atrastu precīzāku saknes aproksimāciju.

```
% 6.3 piemēra turpinājums
figure, x_pr = -3:0.01:3;  plot(x_pr,f(x_pr),'r','LineWidth',3)
grid on, title('Funkcijas f(x) grafiks intervālā [-3, 3]')
```



6.6. att. Funkcijas $f(x) = x^3 - 2x - 5$ grafiks intervālā (–3, 3).

```
% 6.3. piemēra turpinājums
x_app = 2;     % saknes tuvinājumi
iter = 6;      % maksimālais iterāciju skaits
syms x, fpr(x) = diff(f(x),x);
xn = x_app; M = zeros(iter,2);
for i = 1:iter
   xn = xn-f(xn)/double(fpr(xn));
   M(i,1) = xn; M(i,2) = f(xn);
end
disp(['saknes tuvinājums:  ',num2str(x_app(:))])
disp('              x                  f(x)')
disp(M)
```

```
Saknes tuvinājums:  2
              x                  f(x)
                    2.1        0.0610000000000008
         2.09456812110419      0.000185723173270702
          2.0945514816982      1.73976122397335e-09
         2.09455148154233     -8.88178419700125e-16
         2.09455148154233     -8.88178419700125e-16
         2.09455148154233     -8.88178419700125e-16
```

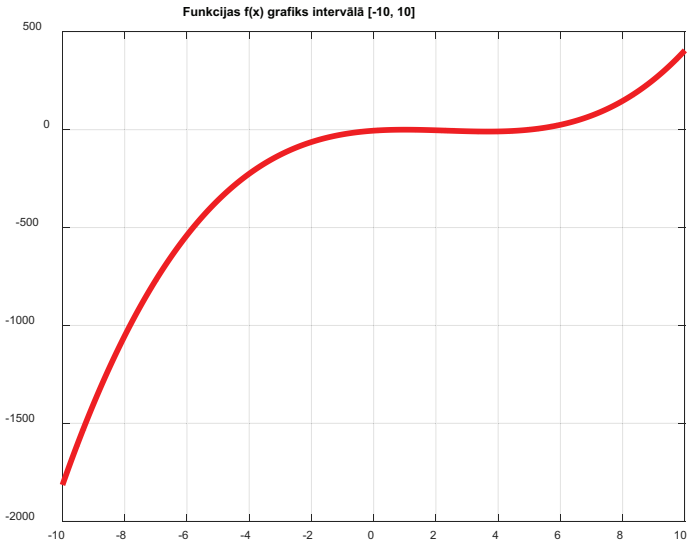
```
% 6.3. piemēra turpinājums
fprintf('Atbilde. Ņūtona metode: vienādojumam ir viena sakne \n' )
fprintf('x = %.5f ar precizitāti 10^(-5)\n',M(iter,1))
format
```

```
Atbilde. Ņūtona metode: vienādojumam ir viena sakne
x = 2.09455 ar precizitāti 10^(-5).
```

6.4. piemērs. Atrast visas vienādojuma $x^3-7x^2+11.01x-5=0$ saknes ar precizitāti 0.00001, izmantojot Ņūtona metodi.

Atrisinājums.

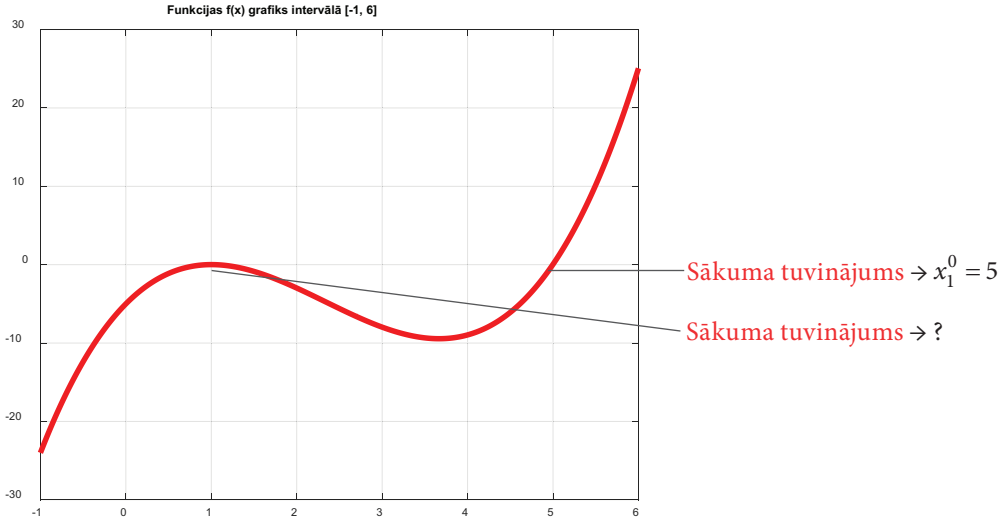
```
%% 6.4. piemērs. Ņūtona metode
clc, clearvars,format compact, close all, format longG
f = @(x)x.^3-7*x.^2+11.01*x-5;
x_pr = -10:0.01:10;  plot(x_pr,f(x_pr),'r','LineWidth',3)
grid on,  title('Funkcijas f(x) grafiks intervālā [-10, 10]')
```



6.7. att. Funkcijas $f(x)=x^3-7x^2+11.01x-5$ grafiks intervālā (–10, 10).

Uzzīmēsim grafiku mazākā intervālā, lai atrastu sakņu sākuma tuvinājumus.

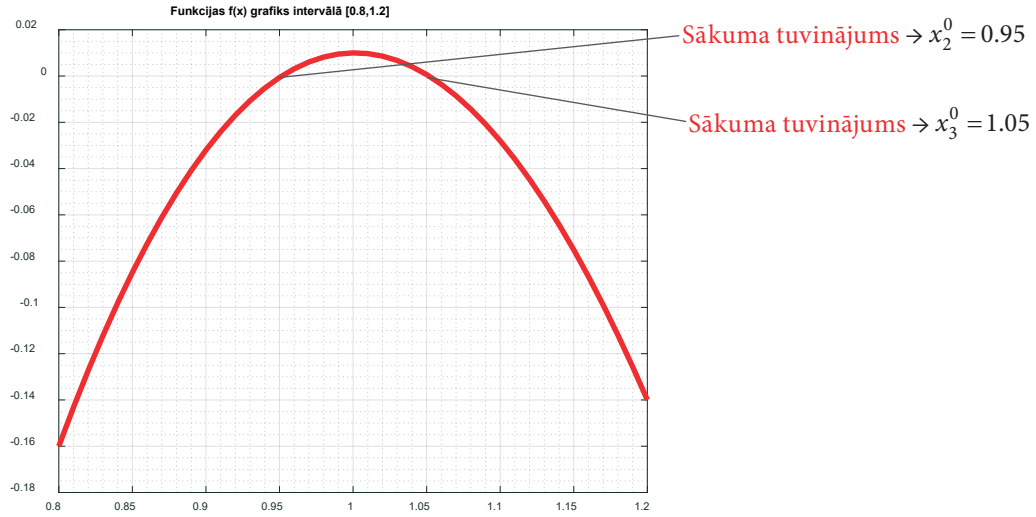
```
% 6.4. piemēra turpinājums
figure, x_pr = -1:0.01:6;  plot(x_pr,f(x_pr),'r','LineWidth',3)
grid on, title('Funkcijas f(x) grafiks intervālā [-1, 6]')
```



6.8. att. Funkcijas $f(x) = x^3 - 7x^2 + 11.01x - 5$ grafiks intervālā (–1, 6).

Viens sākuma tuvinājums (punkta $x = 5$ apkārtnē) ir skaidri redzams 6.8. attēlā. Situācija punkta $x = 1$ apkārtnē nav skaidra, tāpēc uzzīmēsim grafiku punkta $x = 1$ apkārtnē.

```
% 6.4 piemēra turpinājums
figure, x_pr = 0.8:0.01:1.2;  plot(x_pr,f(x_pr),'r','LineWidth',3)
grid on, grid minor, title('Funkcijas f(x) grafiks intervālā [0.8,1.2]')
```



6.9. att. Funkcijas $f(x) = x^3 - 7x^2 + 11.01x - 5$ grafiks intervālā (0.8, 1.2).

Tagad visi trīs vienādojuma $x^3 - 7x^2 + 11.01x - 5 = 0$ sakņu tuvinājumi ir zināmi:
$x_0 = [0.95, 1.05, 5]$

```
% 6.4 piemēra turpinājums
x_app = [0.95 1.05 5];          % saknes tuvinājumi
iter = 6;                       % maksimālais iterāciju skaits
syms x, fpr(x) = diff(f(x),x);  % simboliskā funkcija
for j = 1:3                     % šeit 3 ir sakņu skaits
    M = zeros(iter,2);
    xn = x_app(j);
    for i = 1:iter
       xn  = xn-f(xn)/double(fpr(xn));
       M(i,1) = xn; M(i,2) = f(xn);
    end
    disp(['saknes tuvinājums:  ',num2str(x_app(j))])
    disp('              x                  f(x)')
    disp(M)
    xapp_val(j) = M(iter,1);
end
```

```
Saknes tuvinājums:  0.95
              x                  f(x)
         0.951497005988027     -9.29690692075269e-06
         0.951519956691124     -2.18357154579962e-09
         0.951519962084106                         0
         0.951519962084106                         0
         0.951519962084106                         0
         0.951519962084106                         0
```

```
Saknes tuvinājums:  1.05
              x                  f(x)
          1.05163398692811     -1.02748035306632e-05
          1.05160797961909     -2.60076760127959e-09
          1.05160797303276      1.77635683940025e-15
          1.05160797303277                         0
          1.05160797303277                         0
          1.05160797303277                         0
```

```
Saknes tuvinājums:  5
              x                  f(x)
          4.99687695190506      7.79969747810583e-05
          4.99687206489509      1.90851778825163e-10
          4.99687206488313     -2.8421709430404e-14
          4.99687206488313       7.105427357601e-15
          4.99687206488313      1.4210854715202e-14
          4.99687206488313     -2.8421709430404e-14
```

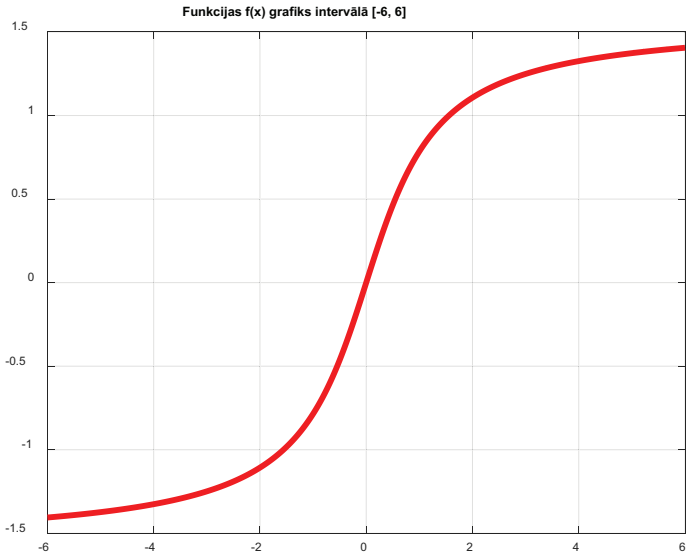
```
% 6.4 piemēra turpinājums
fprintf('Atbilde. Ņūtona metode: vienādojumam ir trīs saknes \n' )
fprintf('   x1 = %.5f,  x2 = %.5f,  x3 = %.5f ',xapp_val(:))
fprintf('ar precizitāti 10^(-5)\n'), format
```

```
Atbilde. Ņūtona metode: vienādojumam ir trīs saknes
   x1 = 0.95152,  x2 = 1.05161,  x3 = 4.99687 ar precizitāti 10^(-5)
```

6.5. piemērs. Noteikt vienādojuma $\operatorname{arctg} x = 0$ sakni ar Ņūtona metodi.

Atrisinājums.

Šis uzdevums ir izvēlēts, lai ilustrētu ar Ņūtona metodi saistītās problēmas. No elementārās matemātikas kursa ir zināms, ka vienādojumam $\operatorname{arctg} x = 0$ ir viena vienkāršā sakne $x = 0$ (sk. funkcijas $y = \operatorname{arctg} x$ grafiku 6.10. attēlā).



6.10. att. Funkcijas $f(x) = \operatorname{arctg} x$ grafiks intervālā (-6, 6).

Lai aprēķinātu sakni ar Ņūtona metodi, izvēlēsimes divus sākuma tuvinājumus: (a) $x_0 = 1.2$ un (b) $x_0 = 2$.

```
Saknes tuvinājums:  1.2 (a)
                x                       f(x)
        -0.937581643459592        -0.753194731844205
         0.477715950859446         0.445661974897971
        -0.0696516707235599       -0.0695393624520285
       0.000225051878509852       0.00022505187471035
      -7.59900383720188e-12     -7.59900383720188e-12
                          0                         0
```

```
Saknes tuvinājums:  2 (b)
                x                       f(x)
         -3.53574358897045         -1.29516905880261
          13.9509590869275          1.49923905274924
         -279.344066533617         -1.56721652737137
          122016.998917954          1.57078813121552
         -23386004197.9338         -1.57079632675214
       8.59076667195034e+20          1.5707963267949
```

Aprēķini rāda, ka gadījumā (a) Ņūtona metode konverģē, bet gadījumā (b) – diverģē. Vienīgā starpība starp aprēķiniem (a) un (b) ir saistīta ar sākuma tuvinājuma izvēli. Gadījumā (b) sākuma tuvinājums $x_0 = 2$ atrodas diezgan tālu no saknes $x = 0$.

Tas nozīmē, ka Ņūtona metodes sākuma tuvinājums ir jāizvēlas pēc iespējas tuvāk saknei.

6.6. Sekanšu metode

Ja funkcijas $f(x)$ atvasinājums nav viegli aprēķināms vai to nevar aprēķināt ar formulu (piemēram, ir dotas tikai skaitliskās funkcijas vērtības), tad Ņūtona metodi var aizstāt ar sekanšu metodi. Atvasinājuma $f'(x_n)$ tuvinātai aprēķināšanai var izmantot formulu:

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{6.6}$$

Atzīmēsim, ka (6.6) nav precīza formula, bet atvasinājuma aproksimācija. Ievietojot (6.6) formulā (6.5), iegūstam

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}, n = 1,2,\ldots \tag{6.7}$$

Formulu (6.7) sauc par **sekanšu metodi**.

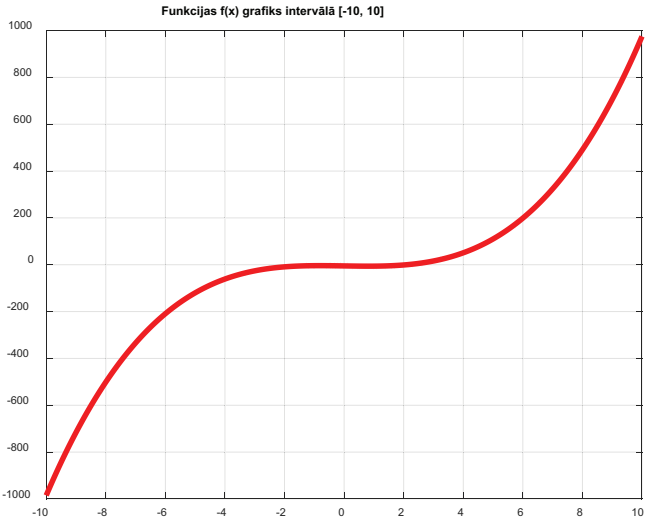
Atzīmēsim, ka, lai uzsāktu aprēķinus, ir nepieciešami divi sākuma tuvinājumi: x_0 un x_1. Formulai (6.7) ir skaidra ģeometriskā interpretācija: x_1 ir sekantes krustpunkts ar Ox asi, ja sekante iet caur punktiem $(x_{n-1}, f(x_{n-1}))$ un $(x_n, f(x_n))$. Var rasties būtiska noapaļošanas kļūda, ja skaitļi x_n un x_{n-1} ir ļoti tuvu viens otram.

6.7. Aprēķini *MATLAB* vidē. Sekanšu metode

6.6. piemērs. Atrast vienādojuma $x^3-2x-5=0$ sakni ar precizitāti 0.0001, izmantojot sekanšu metodi.

Atrisinājums.

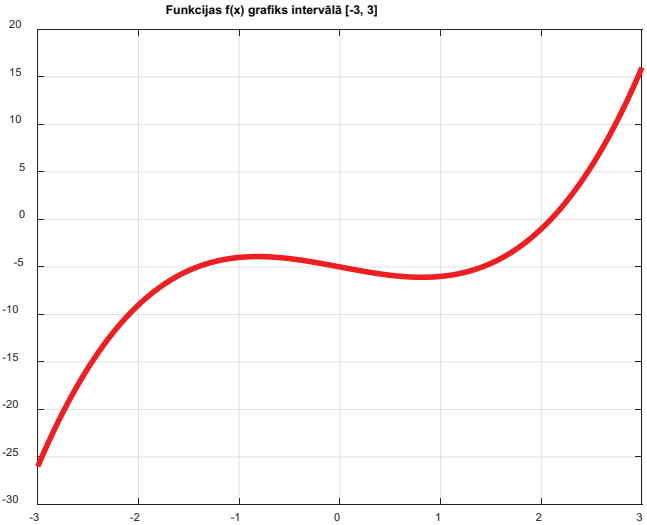
```
%% 6.6. piemērs. Sekanšu metode
clc, clearvars,format compact, close all, format longG
f = @(x)x.^3-2*x-5;
x_pr = -10:0.01:10;  plot(x_pr,f(x_pr),'r','LineWidth',3)
title('Funkcijas f(x) grafiks intervālā [-10, 10]'), grid on
```



6.11. att. Funkcijas $f(x)=x^3-2x-5$ grafiks intervālā (–10, 10).

Lai atrastu sakņu sākuma tuvinājumus, uzzīmēsim grafiku mazākā intervālā.

```
% 6.6. piemēra turpinājums
figure, x_pr =-3:0.01:3; plot(x_pr,f(x_pr),'r','LineWidth',3)
title('Funkcijas f(x) grafiks intervālā [-3, 3]'), grid on
```



6.12. att. Funkcijas $f(x)=x^3-2x-5$ grafiks intervālā (–3, 3).

Lai uzsāktu aprēķinus ar sekanšu metodi, ir nepieciešami divi sākuma tuvinājumi. Par sākuma tuvinājumiem var paņemt intervāla galapunktus (ja intervāls satur sakni). Pieņemsim, ka sākuma tuvinājumi 6.6. piemērā ir $x_0=1.5$ un $x_1=2.5$.

```
% 6.6. piemēra turpinājums
iter = 6;              % maksimālais iterāciju skaits
xn = zeros(1,iter+2);
xn = [1.5 2.5];        % sākuma tuvinājumi
for k = 1:iter
    k_2 = k+2; k_1 = k+1;

    xn(k_2) = xn(k_1)-f(xn(k_1))*(xn(k_1)-xn(k))/(f(xn(k_1))-f(xn(k)));
end
M_pr(:,1) = xn(3:iter+2); M_pr(:,2) = f(xn(3:iter+2));
disp(['saknes tuvinājumu intervāls: ', num2str(xn(1:2))])
disp('              x                  f(x)')
disp(M_pr)
```

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

```
Saknes tuvinājumu intervāls: 1.5         2.5
              x                  f(x)
          1.95121951219512         -1.47364373703225
          2.06514365866838        -0.322824785826985
          2.09710136856466         0.0285012776310722
           2.0945088038308      -0.000476333174634469
          2.09455142033905       -6.8311648337982e-07
           2.0945514815438       1.64099844823795e-11
```

```
% 6.6. piemēra turpinājums
fprintf('Atbilde. Sekanšu metode: vienādojumam ir viena sakne \n' )
fprintf('   x = %.5f ar precizitāti 10^(-5)\n',M_pr(iter,1))
format
```

```
Atbilde. Sekanšu metode: vienādojumam ir viena sakne
   x = 2.09455 ar precizitāti 10^(-5)
```

6.7. piemērs. Izmantojot sekanšu metodi, atrast vismazāko pozitīvo skaitli a, kas apmierina doto vienādojumu (precizitāte $\varepsilon = 0.00001$).

$$\int_2^3 \sqrt[3]{2x+3}\,\ln(ax)\,\mathrm{d}x = 5$$

Atrisinājums.

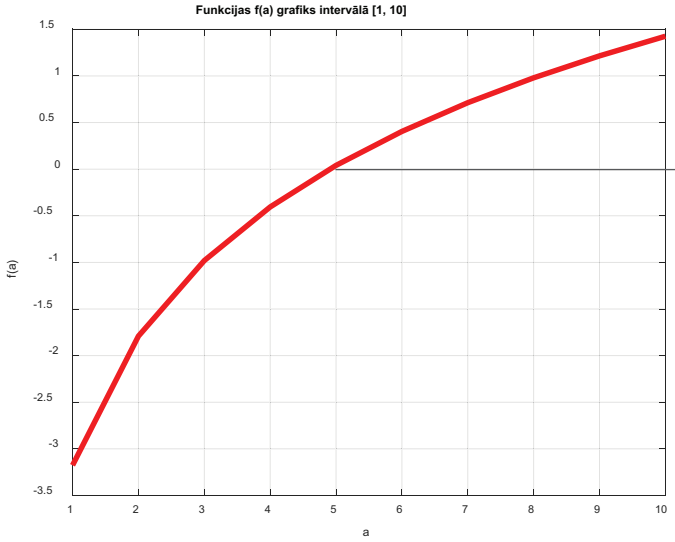
Pirmkārt, ir jāatrod intervāls, kurā atrodas sakne. Definēsim funkciju $f(a)$.

$$f(a) = \int_2^3 \sqrt[3]{2x+3}\,\ln(ax)\,\mathrm{d}x - 5$$

Uzzīmēsim funkcijas $f(a)$ grafiku dažādām a vērtībām (piemēram, intervālā (1, 10)).

Mērķis ir atrast intervālu, kurā $f(a)$ maina zīmi. Nav vispārīgu ieteikumu, kā izvēlēties intervālu. Bieži lietotājam ir jāizmēģina dažādi intervāli, lai atrastu saknes tuvinājumu.

```
%% 6.7. piemērs. Sekanšu metode (Atrast vismazāko pozitīvo skaitli)
clc, clearvars,format compact, close all, format longG
integ = @(a,x)(2*x+3).^(1/3).*log(a.*x);
for a = 1:10
    fun = @(x)integ(a,x);
    int_val(a) = integral(fun,2,3)-5;
end
a_pr = 1:10;  plot(a_pr,int_val,'r','LineWidth',3)
title(['Funkcijas f(a) grafiks intervālā [1, 10]'])
xlabel('a'),ylabel('f(a)'),grid on
```



6.13. att. Funkcijas *f*(*a*) grafiks intervālā (1, 10).

Grafikā redzams, ka par sākuma tuvinājumu var izvēlēties punktu $x = 5$. Ja tiek izmantota sekanšu metode, ir nepieciešami divi tuvinājumi.

Par sākuma tuvinājumiem paņemsim intervāla (4.7, 5.1) galapunktus (t. i., $x_0 = 4.7$ un $x_1 = 5.1$).

```
% 6.7. piemēra turpinājums
iter = 6; a_xn = zeros(1,iter+2);
a_xn = [4.7 5.1]; % sākuma tuvinājumi
for i = 1:2
    fun = @(x)integ(a_xn(i),x);
    f(i) = integral(fun,2,3)-5; % funkcijas f(a)vērtības punktos
end


for k = 1:iter
    k1 = k+1; i = k+2;
    a_xn(i) = a_xn(k1)-f(k1)*(a_xn(k1)-a_xn(k))/(f(k1)-f(k));
    fun = @(x)integ(a_xn(i),x);  f(i) = integral(fun,2,3)-5;
    M_pr(k,1) = a_xn(i);  M_pr(k,2) = f(i);
end
disp(['skaitļa a tuvinājumu intervāls: ', num2str(a_xn(1:2))])
disp('            a                    f(x)')
disp(M_pr)
```

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

```
Skaitļa a tuvinājumu intervāls: 4.7         5.1
            a                    f(x)
          4.90342102050635        0.00166513083466757
          4.89925488601511       -3.38809041933175e-05
          4.89933796513713        1.41160709787869e-08
          4.89933793053763        1.19015908239817e-13
          4.89933793053734        8.88178419700125e-16
          4.89933793053733                           0
```

```
% 6.7. piemēra turpinājums
fprintf('Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis  \n' )
fprintf('   a = %.5f ar precizitāti 10^(-5)\n',M_pr(iter,1))
format
```

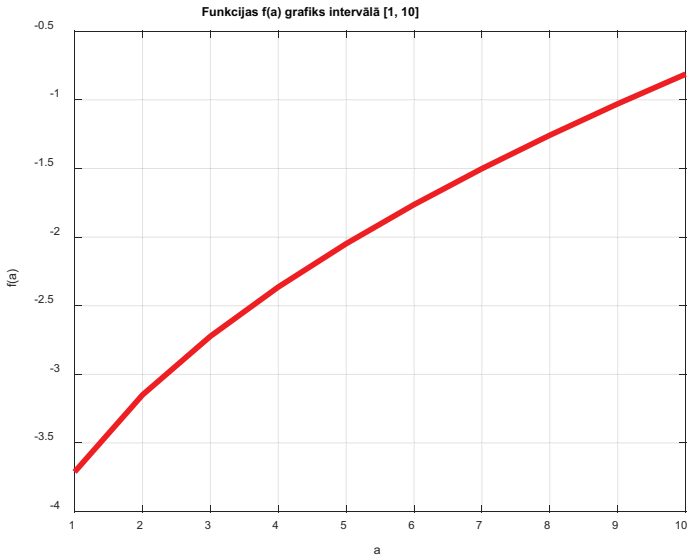
```
Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis
   a = 4.89934 ar precizitāti 10^(-5)
```

6.8. piemērs. Izmantojot sekanšu metodi, atrast minimālo pozitīvo skaitli a, kas apmierina doto vienādojumu (precizitāte $\varepsilon = 0.000001$).

$$\int_2^4 \sqrt{2+ax^4}\,\sin\left(x^2\right)\mathrm{d}\,x = 5$$

Atrisinājums.

```
%% 6.8. piemērs. Sekanšu metode (Atrast vismazāko pozitīvo skaitli)
clc, clearvars,format compact, close all, format longG
integ = @(a,x)(2+a.*x.^4).^(1/2).*sin(x.^2);
for a = 1:10
    fun = @(x)integ(a,x);
    int_val(a) = integral(fun,2,4)-5;
end
a_pr = 1:10;  plot(a_pr,int_val,'r','LineWidth',3)
title(['Funkcijas f(a) grafiks intervālā [1, 10]'])
xlabel('a'),ylabel('f(a)'),grid on
```



6.14. att. Funkcijas $f(a)$ grafiks intervālā (1, 10).

```
% 6.8. piemēra turpinājums
for a = 11:20
    fun = @(x)integ(a,x);
    int_val(a) = integral(fun,2,4)-5;
end
figure
a_pr = 11:20;  plot(a_pr,int_val(11:20),'r','LineWidth',3)
title(['Funkcijas f(a) grafiks intervālā [11, 20]'])
xlabel('a'),ylabel('f(a)'),grid on
```

6.15. att. Funkcijas *f*(*a*) grafiks intervālā (11, 20).

```
% 6.8. piemēra turpinājums
iter = 6;  a_xn = zeros(1,iter+2);
a_xn = [14 14.5];
for i = 1:2
    fun = @(x)integ(a_xn(i),x); f(i) = integral(fun,2,4)-5;
end
for k = 1:iter
    k1 = k+1; i = k+2;
    a_xn(i) = a_xn(k1)-f(k1)*(a_xn(k1)-a_xn(k))/(f(k1)-f(k));
    fun = @(x)integ(a_xn(i),x);  f(i) = integral(fun,2,4)-5;
    M_pr(k,1) = a_xn(i); M_pr(k,2) = f(i);
end
disp(['skaitļa a tuvinājumu intervāls: ', num2str(a_xn(1:2))])
disp('            a                  f(x)')
disp(M_pr)
```

```
Skaitļa a tuvinājumu intervāls: 14          14.5
            a                   f(x)
          14.2337097875253       0.000193422708948177
          14.2326081320141      -9.08012921740919e-07
          14.2326132795143       1.75619518927306e-11
          14.2326132794148      -8.88178419700125e-16
          14.2326132794148       5.32907051820075e-15
          14.2326132794148      -8.88178419700125e-16
```

```
% 6.8. piemēra turpinājums
fprintf('Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis  \n' )
fprintf('   a = %.6f ar precizitāti 10^(-6)\n',M_pr(iter,1))
format
```

```
Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis
   a = 14.232613 ar precizitāti 10^(-6)
```

Aplūkosim dažas iebūvētas *MATLAB* funkcijas, ko lieto vienādojuma (6.1) risināšanai.

KOMANDA `roots`

`roots(p)`	Komandu **`roots`** izmanto, lai atrastu polinoma saknes. Aprēķināt n-tās pakāpes polinoma saknes (**`p`** ir vektors ar $n+1$ komponentēm, kas satur polinoma koeficientus, sākot ar koeficientu pie x^n)

6.9. piemērs. Noteikt vienādojuma $x^3 - 5x + 3 = 0$ saknes, izmantojot komandu **`roots`**.

```
%% 6.9. piemērs. Komanda roots
clc, clearvars,format compact
coef_pol = [1 0 -5 3];  x_sol=roots(coef_pol)
fprintf('Atbilde: \n vienādojuma saknes: ')
fprintf('x1 = %.4f, x2 = %.4f, x3 = %.4f \n  ',x_sol(:))
```

```
x_sol =
   -2.4909
    1.8342
    0.6566
```

```
Atbilde:
 vienādojuma saknes: x1 = -2.4909, x2 = 1.8342, x3 = 0.6566
```

6.10. piemērs. Noteikt vienādojuma $x^3 - 3x^2 + 4 = 0$ saknes, izmantojot komandu **`roots`**.

```
%% 10. piemērs. Komanda roots
clc, clearvars,format compact
coef_pol = [1 0 -5 3];  x_sol=roots(coef_pol)
fprintf('Atbilde: \n vienādojuma saknes: ')
fprintf('x1 = %.0f, x2 = %.0f, x3 = %.0f \n  ',x_sol(:))
```

```
x_sol =
    2.0000
    2.0000
   -1.0000
```

```
Atbilde:
 vienādojuma saknes: x1 = 2, x2 = 2, x3 = -1
```

Kā redzams, ar komandas **`roots`** palīdzību var aprēķināt ne tikai vienkāršas saknes, bet arī saknes ar kārtu m. Vienādojumam 6.10. piemērā ir viena vienkārša sakne $x = -1$ un viena sakne $x = 2$ ar kārtu 2.

6.11. piemērs. Vilkinsons (*Wilkinson*) 1963. gadā ilustrēja, kā mazas kļūdas polinoma koeficientos var ietekmēt augstākās pakāpes polinoma saknes. Aplūkosim polinomu

$$p(x)=(x-1)(x-2)\ldots(x-20)$$

Skaidrs, kas tas ir divdesmitās pakāpes polinoms, kuram ir 20 reālās un vienkāršās saknes $x_1=1, x_2=2, \ldots, x_{20}=20$. Konstruēsim polinomu $p(x)$ pēc x pakāpēm.

```
%% 6.11. piemērs. Komanda roots
clc, clearvars,format compact, format longG
syms x, pol = 1;
for i = 1:20
   pol = pol*(x-i);
end
polyn(x) = collect(pol);
coef_pol = sym2poly(polyn)' % polinoma koeficienti
x_sol  =roots(coef_pol);
M_pr(:,1) = x_sol;
```

```
coef_pol =
                           1
                        -210
                       20615
                    -1256850
                    53327946
                 -1672280820
                 40171771630
              -756111184500
             11310276995381
          -135585182899530
        1.3075350105404e+15
      -1.01422998655115e+16
       6.30308120992949e+16
      -3.11333643161391e+17
       1.20664780378037e+18
      -3.59997951794761e+18
       8.03781182264505e+18
       -1.2870931245151e+19
       1.38037597536407e+19
       -8.7529480367616e+18
       2.43290200817664e+18
```

```
coef_pol =
                           1
          -209.999999880791
                       20615
                    -1256850
                    53327946
                 -1672280820
                 40171771630
              -756111184500
             11310276995381
          -135585182899530
        1.3075350105404e+15
      -1.01422998655115e+16
       6.30308120992949e+16
      -3.11333643161391e+17
       1.20664780378037e+18
      -3.59997951794761e+18
       8.03781182264505e+18
       -1.2870931245151e+19
       1.38037597536407e+19
       -8.7529480367616e+18
       2.43290200817664e+18
```

Koeficients pie x^{19} ir –210. Konstruēsim polinomu $\tilde{p}(x)$, kuram visi koeficienti sakrīt ar polinoma $p(x)$ koeficientiem, izņemot koeficientu pie x^{19}, kas ir vienāds ar $-210+2^{-23}$ (kur $2^{-23}\approx 1.19\cdot 10^{-7}$).

Aprēķināsim polinoma $\tilde{p}(x)$ saknes, izmantojot komandu **roots**.

```
% 6.11. piemēra turpinājums
coef_pol(2) = coef_pol(2)+2^(-23)
x_sol = roots(coef_pol);
M_pr(:,2) = x_sol;
disp('Atbilde.'), disp(' Polinomu saknes:')
disp('       pirmais polinoms        otrais polinoms')
disp(M_pr), format
```

```
Atbilde.
 Polinomu saknes:
       pirmais polinoms              otrais polinoms
 19.9998092912366 + 0i        20.4767768303906 + 1.0390203092122i
 19.0019098182994 + 0i        20.4767768303906 - 1.0390203092122i
 17.9909213527165 + 0i        18.1813329681283 + 2.5489513774804i
 17.0254271462374 + 0i        18.1813329681283 - 2.5489513774804i
 15.946286716608 +  0i        15.3059451708611 + 2.775398129872591
 15.0754937996995 + 0i        15.3059451708611 - 2.775398129872591
 13.9147555918021 + 0i        12.821765504111 +  2.123564919755091
 13.0743140324473 + 0i        12.821765504111 -  2.123564919755091
 11.9532832538469 + 0i        10.8929657643548 + 1.149555981494331
 11.0250229329093 + 0i        10.8929657643548 - 1.149555981494331
  9.99041304248173 + 0i       9.50162317560761 + 0i
  9.00291529436205 + 0i       9.14747678837822 + 0i
  7.99935582960776 + 0i       7.99303334275146 + 0i
  7.00010200279301 + 0i       7.000300993958 +   0i
  5.99998924582477 + 0i       5.99999294947909 + 0i
  5.00000066576979 + 0i       5.00000016640481 + 0i
  3.99999998373753 + 0i       3.99999998804744 + 0i
  2.9999999995921 +  0i       3.00000000047138 + 0i
  2.00000000002832 + 0i       2.00000000000147 + 0i
  0.999999999999699 + 0i      0.999999999999804 +0i
```

Kā redzams, ļoti maza kļūda (ap 10^{-7}) vienā koeficientā pie x^{19} rada lielas izmaiņas atrisinājumā (desmit sakņu ir kompleksi saistītie skaitļi ar imagināro daļu, kas pārsniedz pēc moduļa 1).

Tas ir klasisks nekorekta uzdevuma piemērs – mazas izmaiņas ieejas datu kopā rada lielas izmaiņas atrisinājumā. Šī iemesla dēļ augstākās pakāpes polinomus neiesaka izmantot skaitliskos aprēķinos (atgādināsim arī Runges un Bernšteina piemērus, kas ilustrē interpolācijas problēmas, izmantojot augstāku kārtu interpolācijas polinomus).

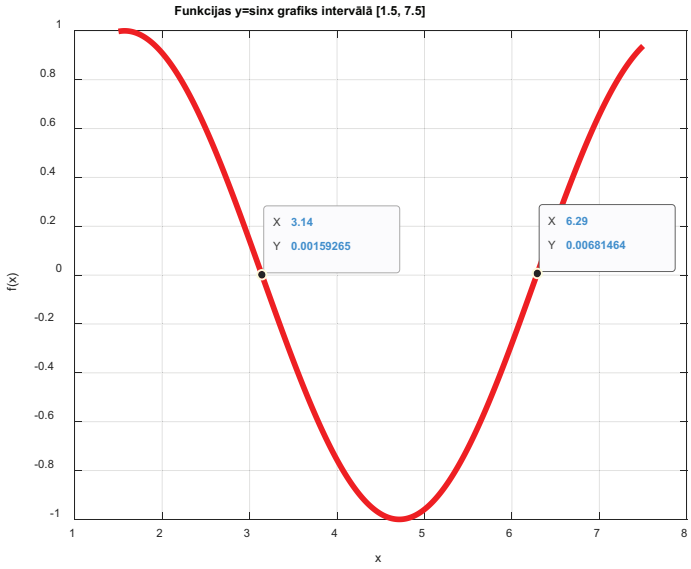
KOMANDA `fzero`

Komandu **`fzero`** izmanto, lai atrastu vienādojuma (6.1) sakni, ja ir zināms sākuma tuvinājums.

`fzero(f,x0)`	Aprēķināt vienādojuma $f(x)=0$ sakni (**`x0`** ir sākuma tuvinājums)

6.12. piemērs. Atrast vienādojuma $\sin x=0$ sakni, ja sākuma tuvinājumi ir (a) $x_0=2$, (b) $x_0=7$, (c) x_0 pieder intervālam [1.5, 2].

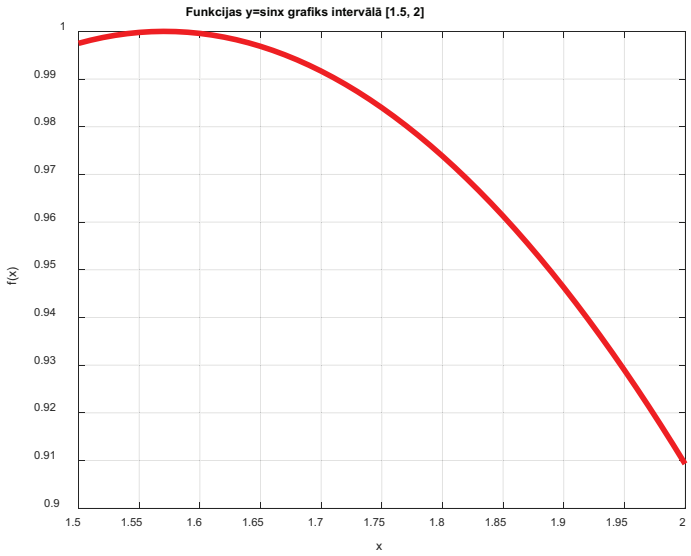
```
%% 6.12. piemērs. Komanda fzero
clc, clearvars,format compact, close all
f = @(x)sin(x);
x_pr = 1.5:0.01:7.5;  plot(x_pr,f(x_pr),'r','LineWidth',3)
title(['Funkcijas y=sinx grafiks intervālā [1.5, 7.5]'])
xlabel('x'),ylabel('f(x)'),grid on
```



6.16. att. Funkcijas $y=\sin x$ grafiks intervālā (1.5, 7.5).

Grafikā redzams, ka intervālā (1.5, 7.5) funkcijai $y=\sin x$ ir divas saknes (sakņu tuvinājumi ir 3.13 un 6.29).

```
% 6.12. piemēra turpinājums
figure
x_pr = 1.5:0.01:2;  plot(x_pr,f(x_pr),'r','LineWidth',3)
title(['Funkcijas y=sinx grafiks intervālā [1.5, 2]'])
xlabel('x'),ylabel('f(x)'),grid on
```



6.17. att. Funkcijas $y = \sin x$ grafiks intervālā (1.5, 2).

Funkcijas $y = \sin x$ grafiks nekrustojas ar Ox asi intervālā (1.5, 2). Tas nozīmē, ka šajā intervālā sakņu nav.

```
% 6.12. piemēra turpinājums
x1 = fzero(f,2)
x2 = fzero(f,7)
x3 = fzero(f,[1.5 2])
```

```
x1 =
    3.1416
x2 =
    6.2832
Error using fzero (line 290)
The function values at the interval endpoints must differ in sign.
Error in nelineāri_vienādojumi (line 267)
x3 = fzero(f,[1.5 2])
```

Rezultāti rāda, ka atkarībā no sākuma tuvinājuma iegūstam dažādas saknes. Piemēram, gadījumā (a) ir iegūta sakne $x = 3.14\ldots$, bet gadījumā (b) – sakne $x = 6.28\ldots$

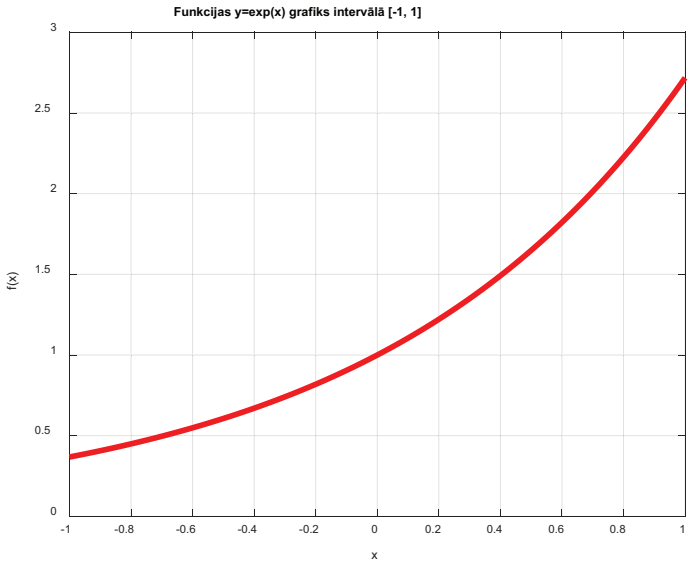
Gadījumā (c) sākuma tuvinājumu definē kā intervālu. Tā kā intervālā [1.5, 2] nav vienādojuma $\sin x = 0$ sakņu, gadījumā (c) atrisinājuma nav.

6.13. piemērs. Atrast vienādojuma $e^x = 0$ sakni.

Atrisinājums.

No matemātikas kursa ir zināms, ka vienādojumam $e^x = 0$ nav reālu sakņu (funkcijas $y = e^x$ grafiks nekrustojas ar Ox asi). Mēģināsim izmantot **fzero**, lai atrastu vienādojuma sakni (pieņemsim, ka sākuma tuvinājums ir $x_0 = 0$).

```
%% 6.13. piemērs. Komanda fzero
clc, clearvars,format compact, close all
f = @(x)exp(x);
x_pr = -1:0.01:1;  plot(x_pr,f(x_pr),'r','LineWidth',3)
title(['Funkcijas y=exp(x) grafiks intervālā [-1, 1]'])
xlabel('x'),ylabel('f(x)'),grid on
```



6.18. att. Funkcijas $y = e^x$ grafiks intervālā (–1, 1)

```
% 6.13. piemēra turpinājums
x = fzero(f,0)
```

```
x =
  -926.8190
```

Rezultātā it kā iegūstam "sakni": $x_1 = 926.8190$. *MATLAB* neziņo par kļūdu! Mēs taču zinām, ka vienādojumam $e^x = 0$ reālu sakņu nav.

Kāpēc rodas kļūdains aprēķins (bez paziņojuma par kļūdu)?

Mums, protams, nav precīzas informācijas, kā strādā **`fzero`**. Ļoti iespējams, ka katrā solī **`fzero`** aprēķina funkcijas $y = e^x$ vērtību un salīdzina to ar nulli. Protams, $e^{-926.819}$ ir ļoti tuva nullei – aprēķina *MATLAB* vidē rezultāts ir precīzi nulle.

Tomēr tas nenozīmē, ka $x_1 = 926.8190$ ir vienādojuma $e^x = 0$ sakne!

Mūsdienās izskan apgalvojumi, ka inženierim matemātika nav vajadzīga. Inženierim ir jāprot tikai lietot datorprogrammas. Ja inženieris zinās, kā formulēt problēmu datoram, atliks tikai analizēt rezultātus. Šis piemērs rāda, kur var novest šāda kļūdaina pieeja!

6.8. Nelineāro vienādojumu sistēmas

Aplūkosim vienādojumu sistēmu

$$\begin{aligned} &f_1(x_1,x_2,\ldots,x_n)=0\\ &f_2(x_1,x_2,\ldots,x_n)=0\\ &\ldots\\ &f_n(x_1,x_2,\ldots,x_n)=0 \end{aligned} \tag{6.8}$$

Apzīmēsim sistēmas (6.8) precīzo un tuvināto atrisinājumu attiecīgi ar $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ un $x_1, x_2, \ldots, x_n$.

Ja tuvinātais atrisinājums ir tuvu precīzam atrisinājumam, izvirzīsim funkcijas $f_1, f_2, \ldots, f_n$ pēc Teilora rindas punkta $M(x_1, x_2, \ldots, x_n)$ apkārtnē un ņemsim vērā tikai lineāros locekļus.

Rezultāts ir

$$f_1(\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n)=f_1(x_1,x_2,\ldots,x_n)+\left.\frac{\partial f_1}{\partial x_1}\right|_M(\bar{x}_1-x_1)+\ldots+\left.\frac{\partial f_1}{\partial x_n}\right|_M(\bar{x}_n-x_n)$$

$$\ldots$$

$$f_n(\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n)=f_n(x_1,x_2,\ldots,x_n)+\left.\frac{\partial f_n}{\partial x_1}\right|_M(\bar{x}_1-x_1)+\ldots+\left.\frac{\partial f_n}{\partial x_n}\right|_M(\bar{x}_n-x_n)$$

Izmantojot apzīmējumu $\Delta x_i=\bar{x}_i-x_i, i=1,2,\ldots,n$, pārrakstīsm vienādojumus matricu veidā:

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial x_1} & \ldots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}\begin{pmatrix}\Delta x_1\\ \ldots\\ \Delta x_n\end{pmatrix}=\begin{pmatrix}-f_1\\ \ldots\\ -f_n\end{pmatrix} \tag{6.9}$$

Matricu A sauc par **Jakobi matricu**.

$$A=\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial x_1} & \ldots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Atrisinot lineāro sistēmu (6.9), iegūstam korekcijas Δx_i, $i=1, 2, \ldots, n$. Nākamo sistēmas (6.8) atrisinājuma aproksimāciju iegūst pēc formulām

$$x_i^{(m+1)}=x_i^{(m)}+\Delta x_i^{(m)}, i=1,2,\ldots,n$$

Aplūkosim gadījumu $n=2$.

Dota sistēma

$$\begin{cases} f_1(x_1,x_2)=0\\ f_2(x_1,x_2)=0 \end{cases} \tag{6.10}$$

Apzīmēsim sākuma tuvinājumu atrisinājumam ar $\left(x_1^{(0)}, x_2^{(0)}\right)$.

Izvirzīsim funkcijas $f_1(x_1, x_2)$ un $f_2(x_1, x_2)$ pēc Teilora rindas punkta $M_0\left(x_1^{(0)}, x_2^{(0)}\right)$ apkārtnē un ņemsim vērā tikai lineāros locekļus attiecībā pret $\Delta x_1=x_1-x_1^{(0)}$ un $\Delta x_2=x_2-x_2^{(0)}$.

Ja punkts $\left(x_1^{(0)}, x_2^{(0)}\right)$ ir tuvu sistēmas precīzam atrisinājumam, tad Teilora rinda ir aptuveni vienāda ar nulli.

Tādējādi

$$f_1(x_1,x_2)=f_1\left(x_1^{(0)},x_2^{(0)}\right)+\left.\frac{\partial f_1}{\partial x_1}\right|_{M_0}\left(x_1-x_1^{(0)}\right)+\left.\frac{\partial f_1}{\partial x_2}\right|_{M_0}\left(x_2-x_2^{(0)}\right)\approx 0$$

$$f_2(x_1,x_2)=f_2\left(x_1^{(0)},x_2^{(0)}\right)+\left.\frac{\partial f_2}{\partial x_1}\right|_{M_0}\left(x_1-x_1^{(0)}\right)+\left.\frac{\partial f_2}{\partial x_2}\right|_{M_0}\left(x_2-x_2^{(0)}\right)\approx 0$$

Sistēmu var pārrakstīt šādā veidā

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{M_0}\left(x_1-x_1^{(0)}\right)+\left.\frac{\partial f_1}{\partial x_2}\right|_{M_0}\left(x_2-x_2^{(0)}\right)=-f_1\left(x_1^{(0)},x_2^{(0)}\right)$$

$$\left.\frac{\partial f_2}{\partial x_1}\right|_{M_0}\left(x_1-x_1^{(0)}\right)+\left.\frac{\partial f_2}{\partial x_2}\right|_{M_0}\left(x_2-x_2^{(0)}\right)=-f_2\left(x_1^{(0)},x_2^{(0)}\right)$$

vai arī matricu veidā:

$$A\Delta\boldsymbol{x}=\boldsymbol{b} \tag{6.11}$$

kur

$$A=\begin{pmatrix}\left.\frac{\partial f_1}{\partial x_1}\right|_{M_0} & \left.\frac{\partial f_1}{\partial x_2}\right|_{M_0}\\ \left.\frac{\partial f_2}{\partial x_1}\right|_{M_0} & \left.\frac{\partial f_2}{\partial x_2}\right|_{M_0}\end{pmatrix},\ \Delta\boldsymbol{x}=\begin{pmatrix}\Delta x_1\\ \Delta x_2\end{pmatrix}\text{ un }\boldsymbol{b}=\begin{pmatrix}-f_1\left(x_1^{(0)},x_2^{(0)}\right)\\ -f_2\left(x_1^{(0)},x_2^{(0)}\right)\end{pmatrix}.$$

Atrisinot sistēmu, iegūstam korekcijas $\Delta x_1=x_1-x_1^{(0)}$ un $\Delta x_2=x_2-x_2^{(0)}$.

Nākamo tuvinājumu atrisinājumam uzrakstīsim šādā veidā:

$$x_1^{(1)}=x_1^{(0)}+\Delta x_1,\ x_2^{(1)}=x_2^{(0)}+\Delta x_2$$

Atkārtojot šo procedūru, iegūstam

$$x_1^{(m)}=x_1^{(m-1)}+\Delta x_1^{(m)},\ x_2^{(m)}=x_2^{(m-1)}+\Delta x_2^{(m)}$$

Katrā solī pārbaudīsim, vai vektora $\boldsymbol{f}$ norma ir pietiekami maza.

$$\boldsymbol{f}=\begin{pmatrix}f_1\left(x_1^{(m)},x_2^{(m)}\right)\\ f_2\left(x_1^{(m)},x_2^{(m)}\right)\end{pmatrix}$$

Precīzam atrisinājumam $\|\boldsymbol{f}\|=0$.

$$\left\|\boldsymbol{f}\left(x_1^{(m)},x_2^{(m)}\right)\right\|<\varepsilon$$

6.9. Aprēķini *MATLAB* vidē. Nelineāro vienādojumu sistēmas

`fimplicit(f,interval)`	Var konstruēt apslēptā veidā ($f(x, y) = 0$) dotās funkcijas grafiku intervālā $[x_{min} x_{max} y_{min} y_{max}]$.

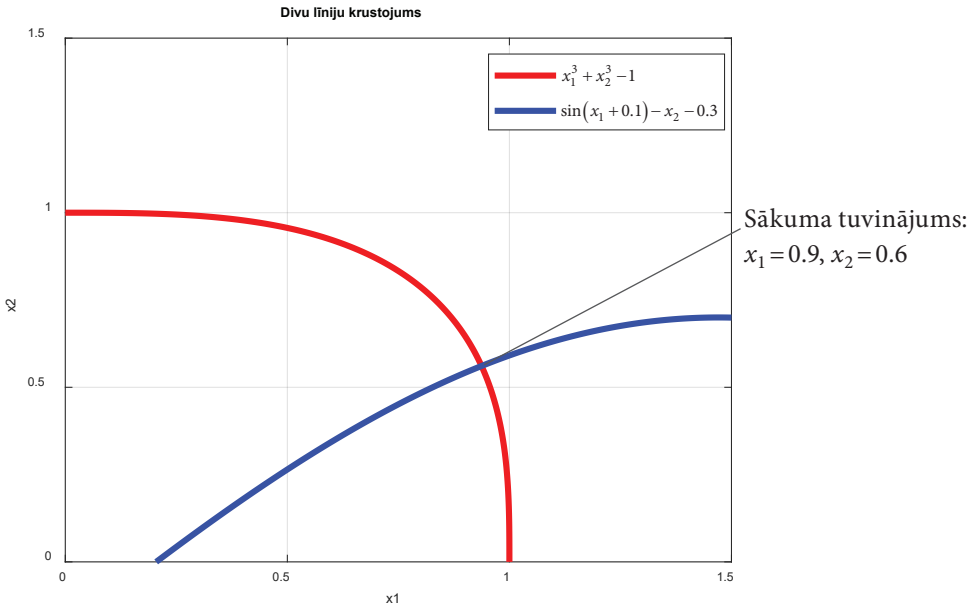
6.14. piemērs. Atrisināt sistēmu ar Ņūtona metodi apgabalā $D = \{0 \le x_1 \le 1.5, 0 \le x_2 \le 1.5\}$ ar precizitāti 0.00001:

$$\begin{cases} x_1^3 + x_2^3 = 1 \\ \sin(x_1 + 0.1) - x_2 = 0.3 \end{cases}$$

Atrisinājums.

Vispirms pārrakstīsim sistēmu standartā formā (6.10), pieņemot, ka $f_1(x_1, x_2) = x_1^3 + x_2^3 - 1$ un $f_2(x_1, x_2) = \sin(x_1 + 0.1) - x_2 - 0.3$. Uzzīmēsim funkciju $f_1(x_1, x_2) = 0$ un $f_2(x_1, x_2) = 0$ grafikus, izmantojot komandu `fimplicit`.

```
%% 6.14. piemērs. Ņūtona metode vienādojumu sistēmām
clc, clearvars,format compact, close all,format longG
f1 = @(x1,x2)x1.^3+x2.^3-1;
f2 = @(x1,x2)sin(x1+0.1)-x2-0.3;
fimplicit(f1,[0 1.5 0 1.5],'r','LineWidth',3), hold on
fimplicit(f2,[0 1.5 0 1.5],'b','LineWidth',3), hold off
grid on,xlabel('x1'),ylabel('x2')
legend, title('Divu līniju krustojums')
```



6.19. att. Funkciju $f_1(x_1, x_2) = 0$ un $f_2(x_1, x_2) = 0$ grafiki apgabalā D.

Vienādojumu sistēmas atrisinājuma ģeometriskā interpretācija ir divu līniju $f_1(x_1, x_2) = 0$ un $f_2(x_1, x_2) = 0$ krustpunkts. Tādējādi par sākuma tuvinājumu atrisinājumam pieņemsim krustpunkta koordinātas $x_1 = 0.9$, $x_2 = 0.6$, kurus nolasīsim no grafika (sk. 6.19. attēlu). Tālāk definēsim vektoru $\boldsymbol{fun} = (f_1, f_2)$ un aprēķināsim Jakobi matricu *fun_pr*.

```
% 6.14. piemēra turpinājums
syms x1 x2
xapp = [0.9 0.6]; xapp_pr = xapp; xpr = [x1 x2];
fun = [x1^3+x2^3-1,sin(x1+0.1)-x2-0.3];  % funkcijas
% funkcijas atvasinājumi
fun_pr = [diff(fun(1),xpr(1)) diff(fun(1),xpr(2))
          diff(fun(2),xpr(1)) diff(fun(2),xpr(2))]
```

```
fun_pr =
[          3*x1^2, 3*x2^2]
[ cos(x1 + 1/10),      -1]
```

Nākamajā blokā risina sistēmu (6.11), atrod korekciju $\Delta \boldsymbol{x}$ un konstruē nākamo tuvinājumu $\boldsymbol{x}^{(m+1)}$. Katrā iterācijas solī ciklā while pārbauda nosacījumu $\left\|\boldsymbol{x}^{(m+1)}\right\| < \varepsilon$ (mūsu piemērā $\varepsilon = 10^{-5}$). Aprēķinu turpina, ja nosacījums $\left\|\boldsymbol{x}^{(m+1)}\right\| < \varepsilon$ neizpildās. Pretējā gadījumā (ja metode konverģē) iegūstam atrisinājumu ar noteiktu precizitāti.

```
% 6.14. piemēra turpinājums
epsi = 10^(-5);  k = 0; % iterāciju skaits
sol_norm = 1;           % normas sākuma vērtība (jābūt lielākai par epsi)
while sol_norm > epsi
    for i = 1:2
        B(i,1) = -double(subs(fun(i),xpr,xapp));
        for j = 1:2
            A(i,j) = double(subs(fun_pr(i,j),xpr,xapp));
        end
    end
    xdelta = A\B; xapp = xapp+xdelta';
    c = double(subs(fun,xpr,xapp)); sol_norm =norm(c);
    k = k+1;
    M_pr(k,1:2) = xapp(1:2);
    M_pr(k,3) = sol_norm;
end
disp(['saknes tuvinājumi: ', num2str(xapp_pr(1:2))])
disp('          x1                    x2                  kļūdas norma')
disp(M_pr)
```

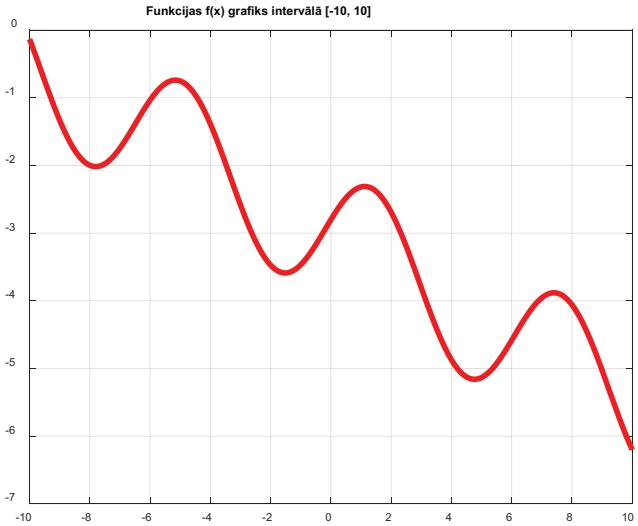
```
saknes tuvinājumi:      0.9      0.6
    x1                     x2                   kļūdas norma
0.939226911323494      0.562665375448065      0.0067037766103792
0.937292066425253      0.561031869039488      1.51271236740301e-05
0.937287727882533      0.56102804974393       7.79017108536289e-11
```

```
% 6.14. piemēra turpinājums
fprintf('Atbilde. Ņūtona metode: sistēmai ir divas saknes \n' )
fprintf('   x1 = %.5f,  x2 = %.5f ',M_pr(k,1:2))
fprintf('ar precizitāti 10^(-5)\n'), format
```

```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x1 = 0.93729,  x2 = 0.56103 ar precizitāti 10^(-5)
```

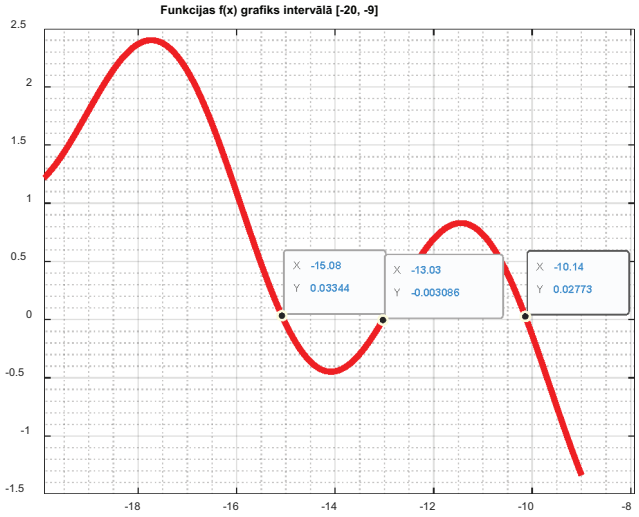
UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

6.1. uzdevums. Atrisināt vienādojumu $\sin(x+0.2)=\frac{x}{4}+3$ ar Ņūtona metodi (precizitāte 0.000001).



$$\sin(x+0.2)-\frac{x}{4}-3=0$$

6.20. att. Funkcijas grafiks intervālā (–20, 10).



6.21. att. Funkcijas grafiks intervālā (–20, –9).

```
saknes tuvinājums:  -15
                x                    f(x)
        -15.0442040302984     0.000761137820258995
        -15.0433579304803     2.72205580742479e-07
        -15.0433576276732     3.50830475781549e-14
        -15.0433576276731                        0
        -15.0433576276731                        0
        -15.0433576276731                        0
```

```
saknes tuvinājums:  -13
                x                    f(x)
        -13.0255801630639     7.84533178674884e-05
        -13.0256896440809     1.53632884192234e-09
        -13.0256896462249    -4.44089209850063e-16
        -13.0256896462249    -4.44089209850063e-16
        -13.0256896462249    -4.44089209850063e-16
        -13.0256896462249    -4.44089209850063e-16
```

```
saknes tuvinājums:  -10.1
                x                    f(x)
        -10.1153302684843     -5.42972683676979e-05
        -10.1153782311848      -5.4189541742744e-10
        -10.1153782316635      4.44089209850063e-16
        -10.1153782316635      4.44089209850063e-16
        -10.1153782316635      4.44089209850063e-16
        -10.1153782316635      4.44089209850063e-16
```

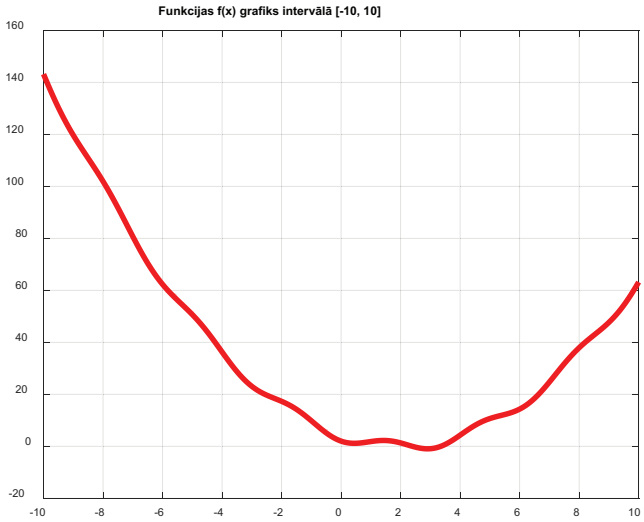
```
Atbilde. Ņūtona metode: vienādojumam ir trīs saknes
   x1 = -15.043358,  x2 = -13.025690,  x3 = -10.115378
   ar precizitāti 10^(-6)
```

6.2. uzdevums. Atrisināt vienādojumu $x^2 - 4x + 4 = 2\cos 2x$ ar Ņūtona metodi (precizitāte 0.00001).

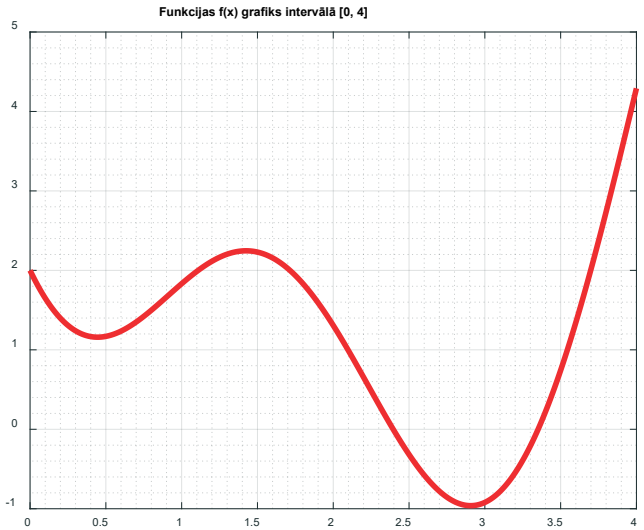
(a) Cik vienādojumam ir sakņu?

(b) Atrast vismazāko sakni.

(c) Atrast vislielāko sakni.



6.22. att. Funkcijas grafiks intervālā (–10, 10).



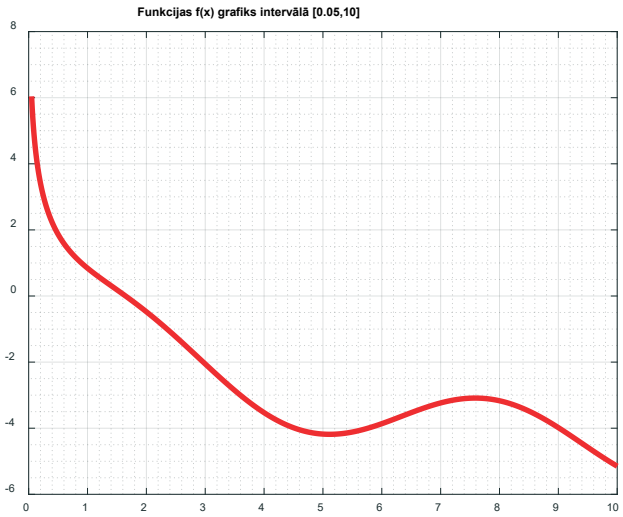
6.23. att. Funkcijas grafiks intervālā (0, 4).

```
saknes tuvinājums:   2.3
                x                   f(x)
          2.39313393544453      0.00693088606821768
          2.39529792970389      6.09242124502418e-06
          2.39529983526615      4.76552131090102e-12
          2.39529983526764                         0
          2.39529983526764                         0
          2.39529983526764                         0
```

```
saknes tuvinājums:   3.3
                x                   f(x)
          3.35472078266816      0.0142283332900697
          3.35145987263205      4.9399812501294e-05
          3.35144847177616      6.04769345713407e-10
          3.35144847163658                         0
          3.35144847163658                         0
          3.35144847163658                         0
```

```
Atbilde. Ņūtona metode: vienādojumam ir divas saknes
  x_vismaz = 2.39530,  x_visliel  = 3.35145 ar precizitāti 10^(-5)
```

6.3. uzdevums. Atrisināt vienādojumu $\sin x - 2\ln x = 0$ ar sekanšu metodi (precizitāte 0.00001).



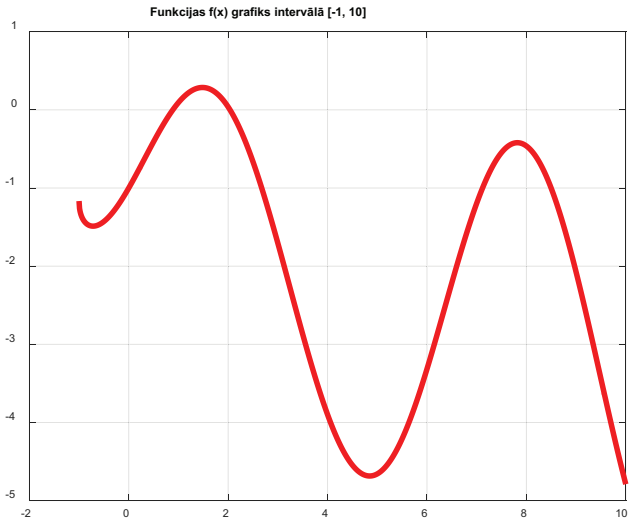
6.24. att. Funkcijas grafiks intervālā (0.05, 10).

Lai konstruētu funkcijas grafiku, jāņem vērā funkcijas $f(x) = \sin x - 2\ln x$ definīcijas apgabals ($x > 0$), tāpēc grafiks ir uzzīmēts tikai pozitīvām x vērtībām.

```
saknes tuvinājumu intervāls: 1.3          2
                x                   f(x)
          1.63541367179252      0.0141214604409474
          1.64589686986917      0.000610388825919239
          1.64637046864417     -6.64542730954842e-07
          1.6463699535883       3.15732995304074e-11
          1.64636995361277      1.11022302462516e-16
          1.64636995361277      1.11022302462516e-16
```

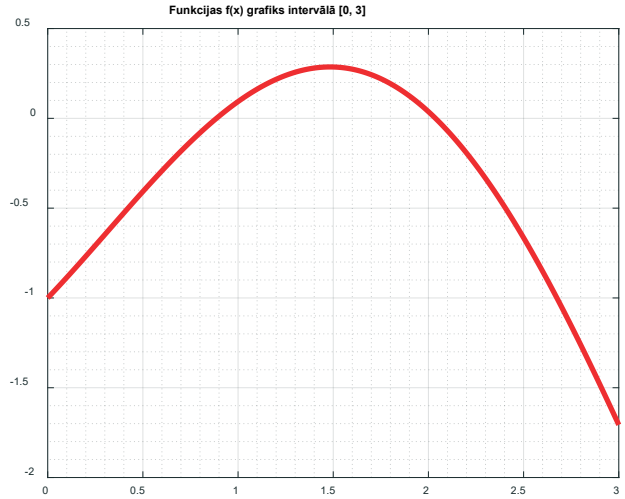
```
Atbilde. Sekanšu metode: vienādojumam ir viena sakne
   x = 1.64637 ar precizitāti 10^(-5)
```

6.4. uzdevums. Atrisināt vienādojumu $\sin x \ln(x+5) = \sqrt{x+1}$ ar Ņūtona metodi (precizitāte 0.00001).



6.25. att. Funkcijas grafiks intervālā (–1, 10).

Grafiks uzzīmēts intervālā, kurā funkcija ir definēta: $x \geq 1$.



6.26. att. Funkcijas grafiks intervālā (0, 3).

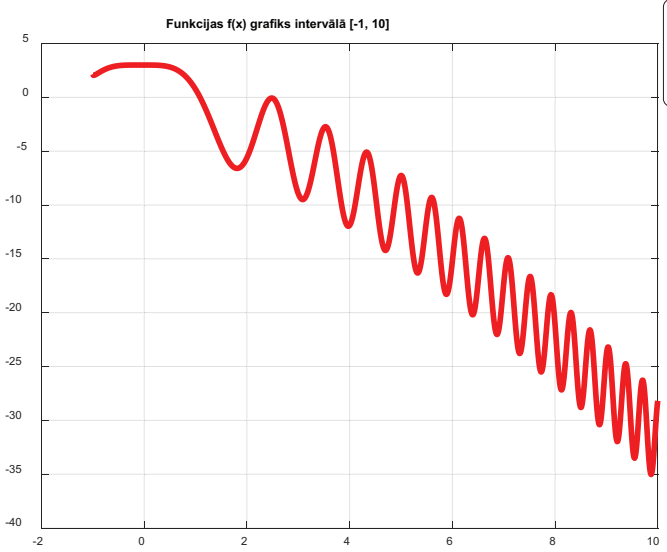
```
saknes tuvinājums:  0.9
              x                    f(x)
    0.886302209151927     -0.000103127628316324
    0.886418296313774     -7.30692617523232e-09
    0.886418304540091      2.22044604925031e-16
    0.88641830454009      -2.22044604925031e-16
    0.886418304540091      2.22044604925031e-16
    0.88641830454009      -2.22044604925031e-16
```

```
saknes tuvinājums:  2
              x                    f(x)
    2.0385730253129       -0.00137852191552335
    2.0372474212691       -1.61747124116651e-06
    2.0372458622296       -2.23798757303939e-12
    2.03724586222744      -2.22044604925031e-16
    2.03724586222744      -2.22044604925031e-16
    2.03724586222744      -2.22044604925031e-16
```

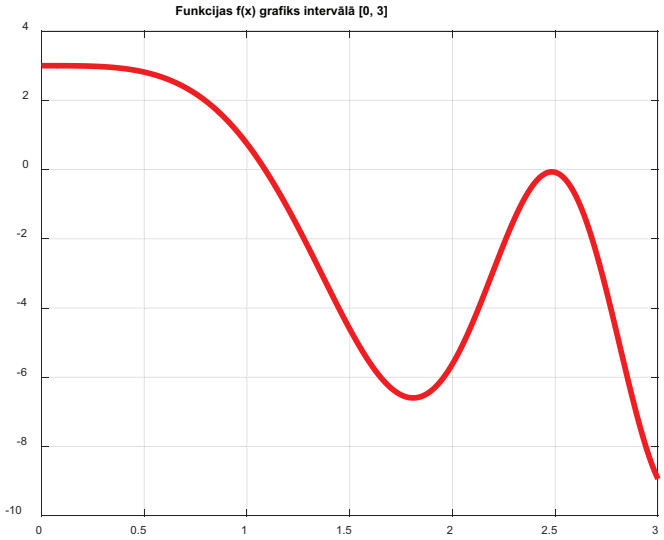
```
Atbilde. Ņūtona metode: vienādojumam ir divas saknes
  x1 = 0.88642,   x2  = 2.03725 ar precizitāti 10^(-5)
```

6.5. uzdevums. Atrisināt vienādojumu $4\cos\left(x^2\right)-\sqrt{1+x^3}=0$ ar sekanšu metodi (precizitāte 0.0001).

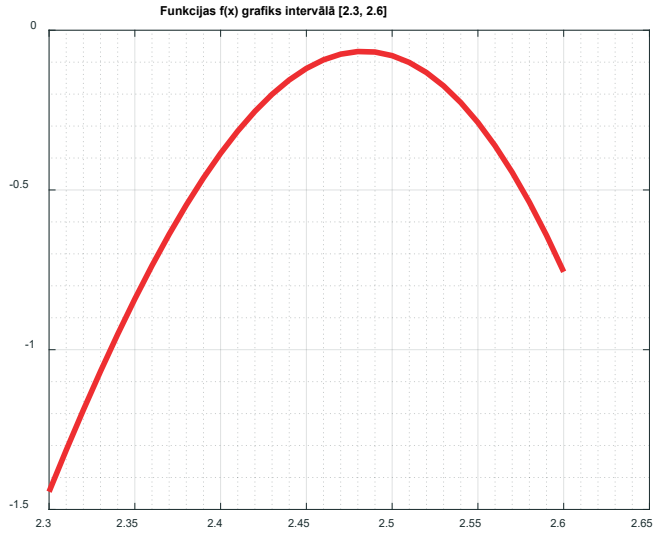
Grafiks uzzīmēts intervālā, kurā funkcija ir definēta: $x \geq -1$.



6.27. att. Funkcijas grafiks intervālā (–1, 10).



6.28. att. Funkcijas grafiks intervālā (0, 3).



6.29. att. Funkcijas grafiks intervālā (2.3, 2.6).

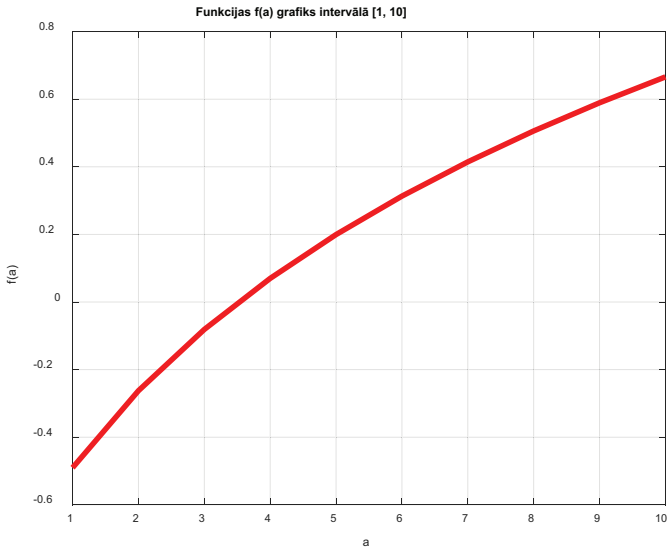
```
saknes tuvinājumu intervāls: 1          1.3
                x                   f(x)
          1.07443414392261       0.121062166496323
          1.08588495835891       0.016751684521028
          1.08772389560964      -0.000194326980957227
          1.08770280774929       3.03252583488955e-07
           1.0877028406062       5.4678483962789e-12
          1.08770284060679       4.44089209850063e-16
```

```
Atbilde. Sekanšu metode: vienādojumam ir viena sakne
   x = 1.0877 ar precizitāti 10^(-4)
```

6.6. uzdevums. Atrast vismazāko pozitīvo skaitli, kas apmierina vienādojumu

$$\int_0^1 \ln\left(1+ax+x^3\right)\mathrm{d}\,x = 1.$$

Izmantot sekanšu metodi (precizitāte 0.000001).



6.30. att. Funkcijas grafiks intervālā [1, 10].

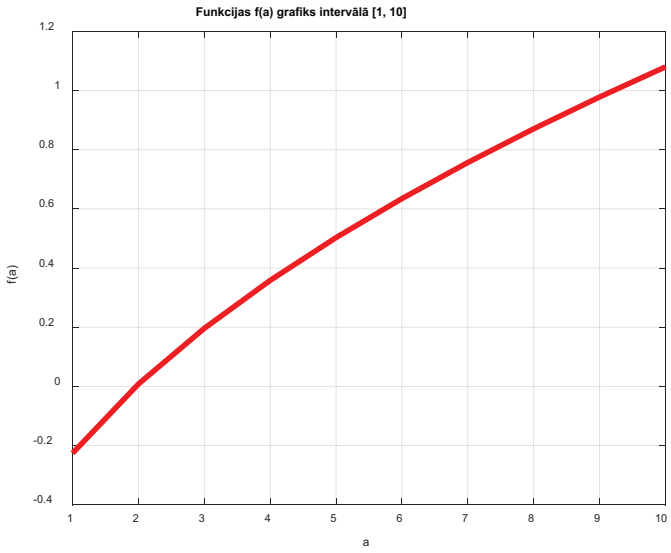
```
skaitļa a tuvinājumu intervāls: 3  4
      a                   f(x)
    3.5373634867467        0.0030883028746953
    3.5159691097849       -0.000121097552774785
    3.51677636545516       2.04920117541363e-07
    3.51677500173253       1.35831346170789e-11
    3.51677500164213      -1.11022302462516e-16
    3.51677500164213       2.22044604925031e-16
```

```
Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis
   a = 3.516775 ar precizitāti 10^(-6)
```

6.7. uzdevums. Atrast vismazāko pozitīvo skaitli, kas apmierina vienādojumu

$$\int_0^1 \sqrt{a\sin x + x^3}\,\mathrm{d}\,x = 1.$$

Izmantot sekanšu metodi (precizitāte 0.0001).



6.31. att. Funkcijas grafiks intervālā [1, 10].

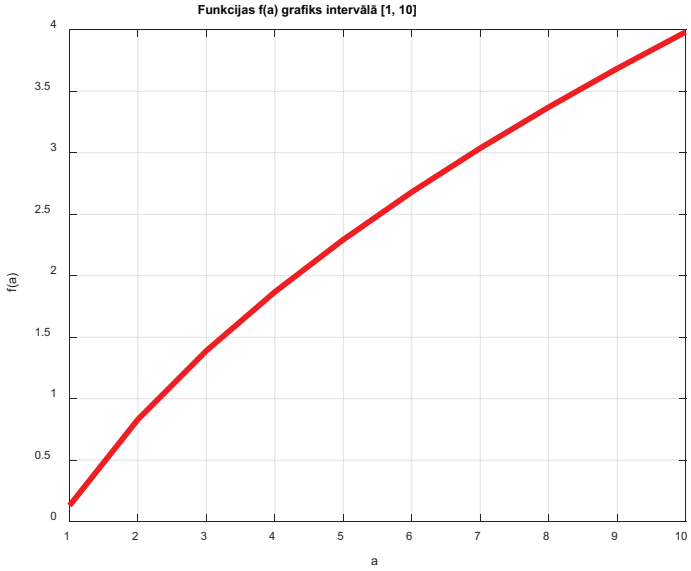
```
skaitļa a  tuvinājumu intervāls: 1.5         2.5
             a                    f(x)
          1.98815511746172       0.00545753227786272
          1.96039343301781      -0.000294669050501239
          1.96181558581262       8.19872176149872e-07
          1.96181163986603       1.23179910715976e-10
          1.96181163927309       2.22044604925031e-16
          1.96181163927309                          0
```

```
Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis
   a = 1.9618 ar precizitāti 10^(-4)
```

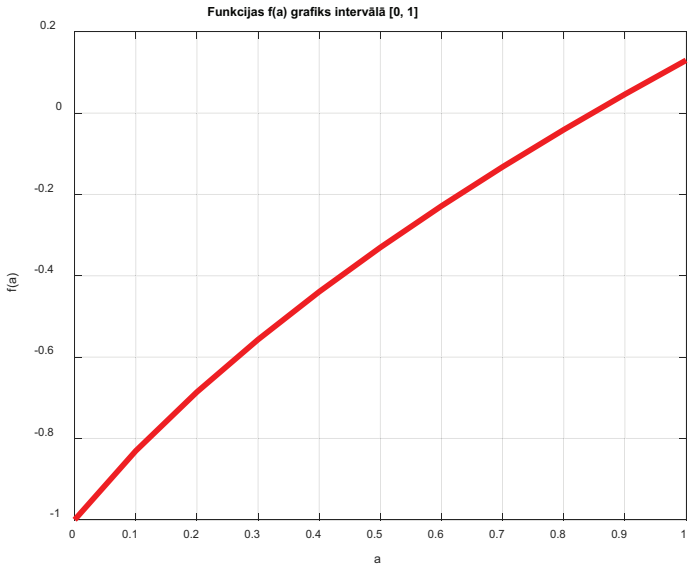
6.8. uzdevums. Izmantojot sekanšu metodi, atrast minimālo pozitīvo skaitli a, kas apmierina vienādojumu

$$\int_1^2 \sqrt{ax^3+1}\,\mathrm{d}\,x = 2$$

(precizitāte $\varepsilon = 0.0001$).



6.32. att. Funkcijas $f(a)$ grafiks intervālā [1, 10].



6.33. att. Funkcijas $f(a)$ grafiks intervālā [0, 1].

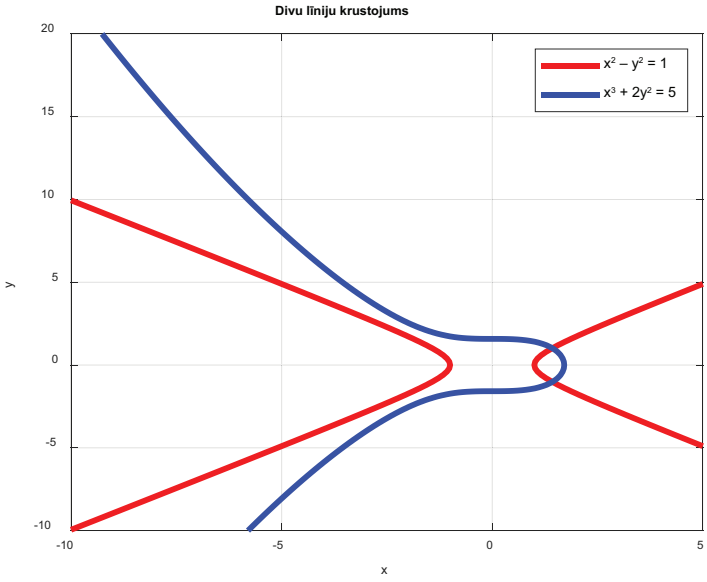
```
skaitļa a tuvinājumu intervāls: 0.8         0.9
            a                    f(x)
         0.846854891858317       0.000483961878640748
         0.846293849842368      -5.73286370864068e-06
         0.846300417969681       7.09730052506075e-10
         0.846300417156645       1.33226762955019e-15
         0.846300417156644                          0
         0.846300417156644                          0
```

```
Atbilde. Sekanšu metode - vismazākais pozitīvais skaitlis
   a = 0.8463 ar precizitāti 10^(-4)
```

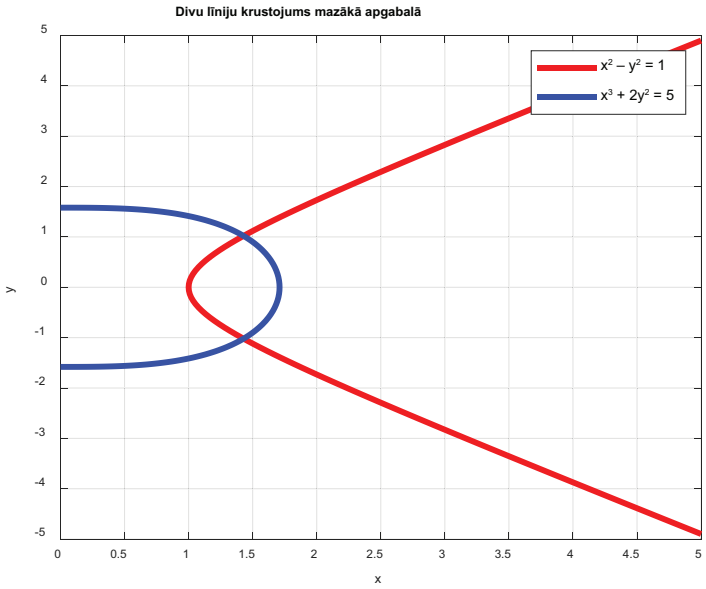
6.9. uzdevums. Atrisināt doto sistēmu ar Ņūtona metodi apgabalā D (precizitāte 0.00001).

$D = \{-10 \le x \le 5, -10 \le y \le 20\}$

$$\begin{cases} x^2 - y^2 = 1 \\ x^3 + 2y^2 = 5 \end{cases}$$



6.34. att. Funkciju grafiki apgabalā $-10 \le x \le 5$, $-10 \le y \le 20$.



6.35. att. Funkciju grafiki apgabalā $0 \le x \le 5$, $-5 \le y \le 5$.

```
saknes tuvinājumi: 1.3          1
              x                          y                    kļūdas norma
      1.43855890944499      1.02512658227848      0.0809552270158599
      1.42886772249466      1.02058301635578      0.000451685479427335
      1.42881770310983      1.02054888534701      1.31228342936034e-08
```

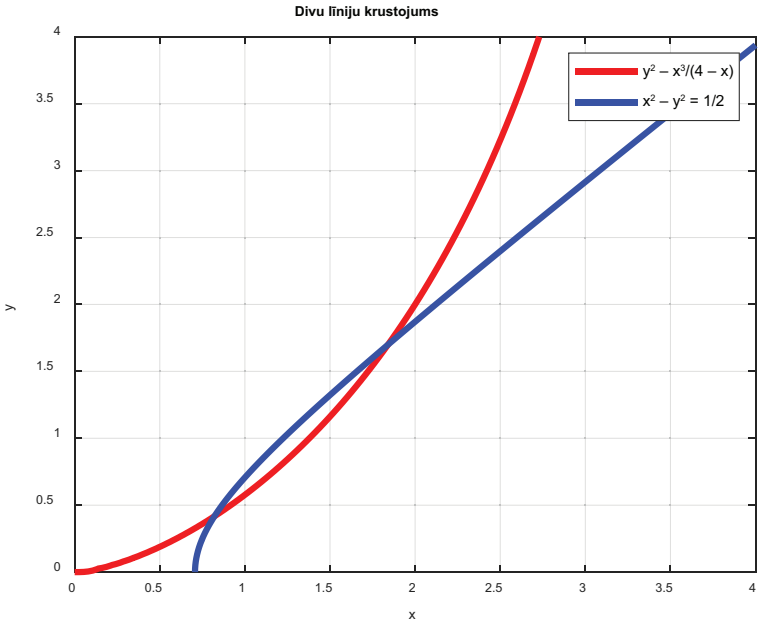
```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = 1.42882,  y = 1.02055 ar precizitāti 10^(-5)
```

```
saknes tuvinājumi:    1.3        -1
      x                        y                    kļūdas norma
1.43855890944499       -1.02512658227848       0.0809552270158599
1.42886772249466       -1.02058301635578       0.000451685479427335
1.42881770310983       -1.02054888534701       1.31228342936034e-08
```

```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = 1.42882,  y = -1.02055 ar precizitāti 10^(-5)
```

6.10. uzdevums. Atrisināt vienādojumu sistēmu ar Ņūtona metodi, apgabalā $x, y \in [0 \quad 4; 0 \quad 4]$ (precizitāte $\varepsilon = 0.00001$).

$$\begin{cases} y^2 - \dfrac{x^3}{4-x} = 0 \\ x^2 - y^2 = \dfrac{1}{2} \end{cases}$$



6.36. att. Funkciju grafiki apgabalā $0 \le x \le 4$, $0 \le y \le 4$.

```
saknes tuvinājumi: 0.7         0.4
              x                        y                    kļūdas norma
   0.823438121766813       0.403516713091922       0.0199811520696918
   0.821036845523193       0.417481435579144       0.00026765585928733
   0.82103681624075        0.417254724383934       7.268769948726e-08
```

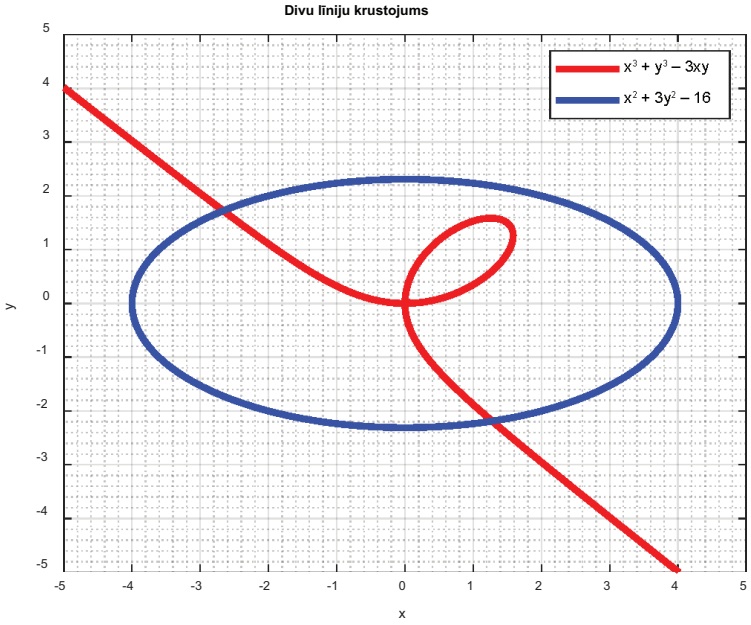
```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = 0.82104,  y = 0.41725 ar precizitāti 10^(-5)
```

```
saknes tuvinājumi: 1.8         1.7
              x                        y                    kļūdas norma
    1.84403594771242       1.70250865051903       0.0101340270292948
    1.84068610225503       1.69944799073363       5.09980734948579e-05
    1.84066532259851       1.69942602953679       1.83137380527074e-09
```

```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = 1.84067,  y = 1.69943 ar precizitāti 10^(-5)
```

6.11. uzdevums. Atrisināt sistēmu ar Ņūtona metodi apgabalā $D=\{-5\le x\le 5,\ -5\le y\le 5\}$ (precizitāte 0.000001).

$$\begin{cases} x^3+y^3-3xy=0 \\ x^2+3y^2=16 \end{cases}$$



6.37. att. Funkciju grafiki apgabalā $-5\le x\le 5$, $-5\le y\le 5$

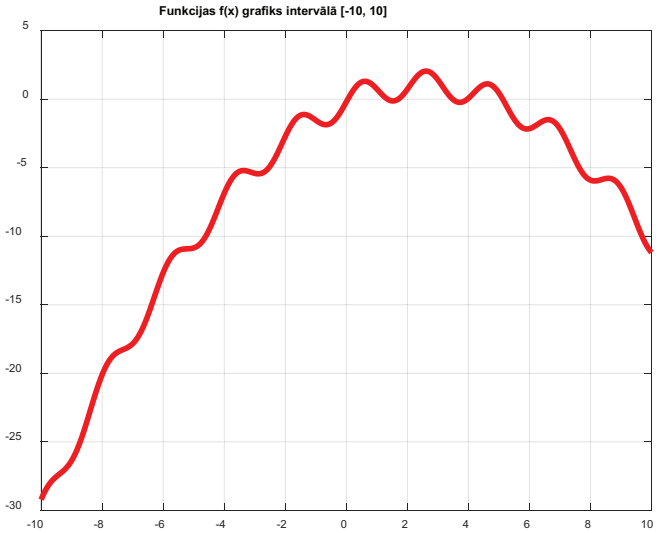
```
saknes tuvinājumi: -2.7         1.6
           x                      y                    kļūdas norma
-2.66400491400491      1.7275389025389       0.0750852078019693
-2.66295881017065      1.72324372577783      0.000115060865888745
-2.66295913379418      1.72323810038323      1.83780647571782e-10
```

```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = -2.662959,  y = 1.723238 ar precizitāti 10^(-6)
```

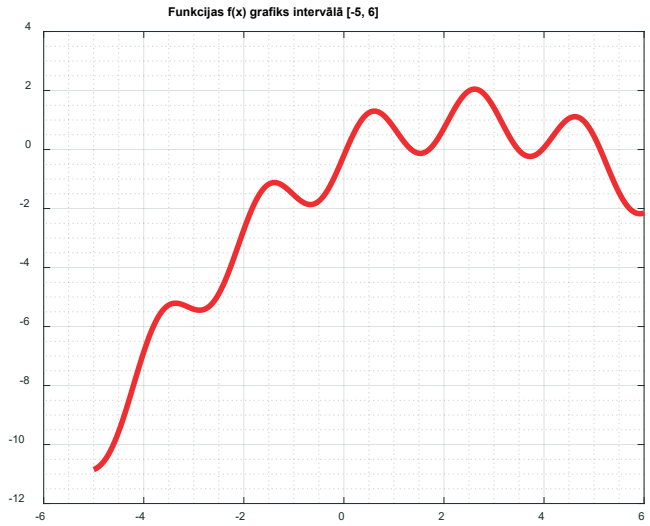
```
saknes tuvinājumi: 1.2         -2.1
           x                      y                    kļūdas norma
1.27945056205552       -2.19042211516403      0.0317328012164267
1.27817267675274       -2.18832427854048      2.08163459168694e-05
1.27817287665888       -2.18832310970278      1.03772995407762e-11
```

```
Atbilde. Ņūtona metode: sistēmai ir divas saknes
   x = 1.278173,  y = -2.188323 ar precizitāti 10^(-6)
```

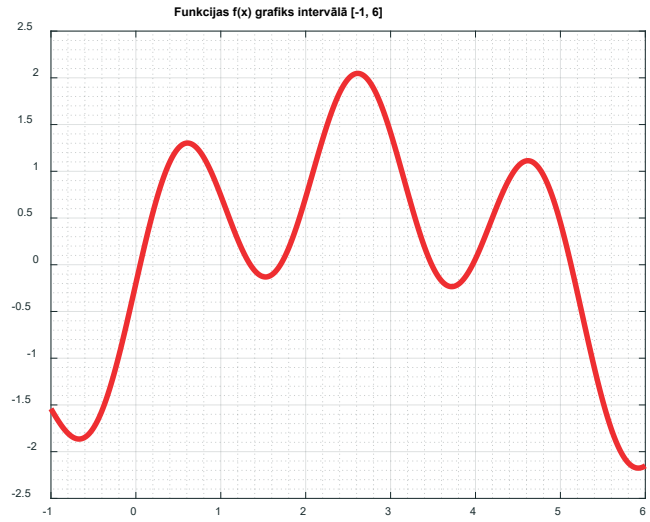
6.12. uzdevums. Atrisināt vienādojumu $\sin 3x - \frac{x^2}{5} = -x + 0.2$ ar Ņūtona metodi (precizitāte $\varepsilon = 10^{-6}$).



6.38. att. Funkcijas grafiks intervālā [–10, 10].



6.39. att. Funkcijas grafiks intervālā [–5, 6].



6.40. att. Funkcijas grafiks intervālā [–1, 6].

```
saknes tuvinājums:  0
                x                   f(x)
                     0.05     -0.00106186752640075
        0.0502690783684639     -6.32563904034988e-08
        0.0502690943996123     -2.22044604925031e-16
        0.0502690943996124                         0
        0.0502690943996124                         0
        0.0502690943996124                         0
```

```
saknes tuvinājums:  1.3
                x                   f(x)
          1.34372362811187      0.00579877858534344
          1.34778807308253      5.4632383868769e-05
          1.34782710237312      5.0725869615853e-09
          1.34782710599764     -5.55111512312578e-17
          1.34782710599764     -5.55111512312578e-17
          1.34782710599764     -5.55111512312578e-17
```

```
saknes tuvinājums:  1.8
                x                   f(x)
          1.71793541214115      0.0235279065889817
          1.70317965297968      0.0008483645002903
          1.7026060862114       1.29971290147024e-06
          1.70260520479357      3.07082137496195e-12
          1.70260520479149      1.66533453693773e-16
          1.70260520479149     -5.55111512312578e-17
```

```
saknes tuvinājums:  3.4
                  x                   f(x)
          3.4751660305281       0.0179503876552577
          3.48410032760139      0.000288143428724907
          3.48424851427796      8.02049912596026e-08
          3.48424855554884      6.38378239159465e-15
          3.48424855554884     -7.21644966006352e-16
          3.48424855554884      6.10622663543836e-16
```

```
saknes tuvinājums:  4
                x                   f(x)
          3.96716280084983      0.00252022421680981
          3.96574223854897      5.20920890928478e-06
          3.96573929009946      2.25283680599375e-11
          3.96573929008671     -2.77555756156289e-16
          3.96573929008671     -2.77555756156289e-16
          3.96573929008671     -2.77555756156289e-16
```

```
saknes tuvinājums:  5.1
                x                   f(x)
          5.12497252474804      -0.00117327493697578
          5.12467053452101     -1.52523081109468e-07
          5.12467049525275     -4.27435864480685e-15
          5.12467049525275      1.66533453693773e-16
          5.12467049525275      1.66533453693773e-16
          5.12467049525275      1.66533453693773e-16
```

```
Atbilde. Ņūtona metode: vienādojumam ir sešas saknes
  x1 = 0.050269,  x2  = 1.347827,  x3  = 1.702605
  x4 = 3.484249,  x5  = 3.965739,  x6  = 5.124670
  ar precizitāti 10^(-6)
```

7. nodaļa

PARASTO DIFERENCIĀL-VIENĀDOJUMU RISINĀŠANA

7.1. Košī problēma

Aplūkosim parasto diferenciālvienādojumu

$$y' = f(x, y) \tag{7.1}$$

ar sākuma nosacījumu

$$y(x_0) = y_0 \tag{7.2}$$

Uzdevumu (7.1), (7.2) sauc par Košī problēmu. Pieņemsim, ka uzdevums (7.1), (7.2) ir jāatrisina intervālā (x_0, b).

Ir zināms, ka pietiekami maza punkta (x_0, y_0) apkārtnē eksistē Košī problēmas (7.1), (7.2) viens vienīgs atrisinājums, ja funkcija $f(x, y)$ un parciāls atvasinājums $f_y(x, y)$ ir nepārtrauktas funkcijas šajā apkārtnē. Turpmāk pieņemsim, ka šis nosacījums ir izpildīts.

Dažos gadījumos var atrast uzdevuma (7.1), (7.2) analītisko atrisinājumu, bet ir ļoti daudz piemēru, kad Košī problēmu (7.1), (7.2) var atrisināt tikai ar skaitliskām metodēm.

Pārveidosim nepārtraukto problēmu (7.1), (7.2) par diskrēto.

Sadalīsim intervālu (x_0, b) apakšintervālos (x_i, x_{i+1}), kur $x_i = x_0 + ih$, $i = 0, 1, \ldots, n$.

Skaitlis h ir diskretizācijas solis. Punktu kopu $\omega_h = \{x_i = x_0 + ih,\ i = 0, 1, \ldots, n\}$ sauc par režģi, bet punktus x_i – par režģa mezgliem.

Apzīmēsim ar $y(x)$ Košī problēmas (7.1), (7.2) precīzo atrisinājumu. Uzdevuma (7.1), (7.2) tuvināto atrisinājumu punktā $x = x_i$ apzīmēsim šādi: $y_i = y(x_i)$.

Tādējādi Košī problēmas tuvināto atrisinājumu definēsim tikai režģa mezglos. Citiem vārdiem, uzdevums (7.1), (7.2) ir uzskatāms par atrisinātu, ja ir zināmas funkcijas $y(x)$ tuvinātās vērtības y_i režģa mezglos x_i, $i = 0, 1, \ldots, n$.

Pieņemsim, ka x ir viens no režģa mezgliem. Ja atrisinājums punktā x ir zināms, tad funkcijas $y(x)$ vērtību punktā $x + h$ var aprēķināt, izmantojot Teilora formulu:

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + \ldots \tag{7.3}$$

Aprēķinus pēc formulas (7.3) var veikt, ja funkcijas $y(x)$ atvasinājumi punktā x ir zināmi.

Pirmās kārtas atvasinājumu aprēķināsim pēc formulas (1): $y'(x) = f(x, y)$.

Aprēķināsim otrās kārtas atvasinājumu:

$$y''(x) = \frac{\mathrm{d}}{\mathrm{d}x}(y)' = \frac{\mathrm{d}}{\mathrm{d}x}\big(f(x, y)\big) = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}x} + \frac{\partial f}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}x} = f_x + f_y y' = f_x + ff_y \tag{7.4}$$

Aprēķini veikti, izmantojot saliktas funkcijas atvasināšanas kārtulu (ņemot vērā, ka $y = y(x)$).

Var pierādīt, ka

$$y'''(x) = f_{xx} + 2f_{xy}f + f_y f_x + f_{yy}f^2 + f_y^2 f$$

Augstāku kārtu atvasinājumi izskatās vēl sarežģītāk, tāpēc aplūkosim alternatīvu risinājumu.

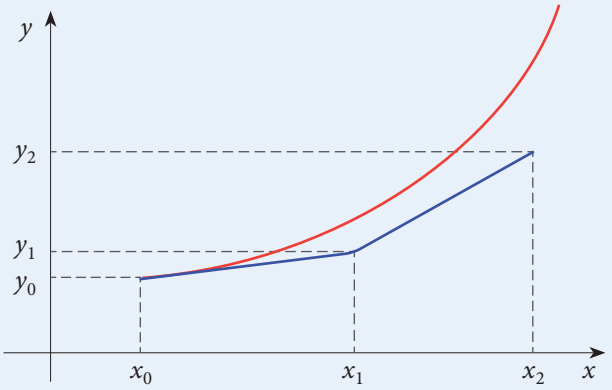
Sāksim ar visvienkāršāko aproksimāciju. Ņemot vērā tikai pirmos divus locekļus formulas (7.3) labajā pusē (pārējie locekļi ir svītroti ar nosacījumu, ka režģa solis h ir pietiekami mazs), iegūstam:

$$y(x+h) \approx y(x) + hf(x, y)$$

Izmantojot iegūto tuvināto vienādojumu režģa ω_h mezglos, iegūstam

$$y_{i+1} = y_i + hf\left(x_i, y_i\right), \quad i = 0,1,\ldots,n \tag{7.5}$$

Formulu (7.5) sauc par **Eilera metodi**. Funkcijas $y(x)$ tuvinātās vērtības punktos x_i aprēķina, izmantojot sākuma nosacījumu (7.2) un formulu (7.5).



7.1. att. Eilera metodes ģeometriskā interpretācija.

Eilera metodi sauc arī par **lauzto līniju metodi**.

Saskaņā ar formulu (7.5) funkcijas $y(x)$ grafiku intervālā (x_i, x_{i+1}) aproksimē ar pieskari, kas ir novilkta funkcijas grafikam punktā (x_i, y_i). Turpinot šo aproksimāciju, aizstāsim nepārtrauktas funkcijas $y(x)$ grafiku ar lauzto līniju, kurai katrs posms ir taisnes nogrieznis ar virziena koeficientu $f(x_i, y_i)$.

Gadījumā, kad funkcija $y(x)$ ir **strauji augoša** (sk. 7.1. attēlu), tuvinātās vērtības y_i (lauztas līnijas virsotņu y koordinātas 7.1. attēlā) var ievērojami atšķirties no precīzām vērtībām $y(x)$.

7.1. piemērs. Izmantojot Eilera metodi, atrisināt Košī problēmu $y' = x + y$, $y(0) = 0$ intervālā (0, 1) ar soli $h = 0.2$.

Atrisinājums.

Izmantojot formulu (7.5), iegūstam

$$y_{i+1} = y_i + 0.2(x_i + y_i), \quad i = 0,1,2,3,4,5$$

Uzdevumu var atrisināt arī analītiski. Precīzs atrisinājums ir $y(x) = e^x - x - 1$.

Aprēķinu rezultāti ir parādīti tabulā.

7.1. tabula. Aprēķini pēc Eilera metodes.

i	x_i	y_i	$0.2 \times (x_i + y_i)$	Precīzs atrisinājums	Absolūtā kļūda
0	0.0	0.0	0.0	0.0	0.0
1	0.2	0.0	0.04	0.021	0.021
2	0.4	0.04	0.088	0.092	0.052
3	0.6	0.128	0.146	0.222	0.094
4	0.8	0.274	0.215	0.426	0.152
5	1.0	0.489	0.298	0.718	0.229

MATLAB skripts un rezultāti ir parādīti turpmāk.

```
%% 7.1. piemērs.Eilera metode.
clc, clearvars, format compact, syms  y(x)
x_a = 0; x_b = 1;  % intervāls (0,1)
ya = 0;            % sākuma nosacījums y(0)=0
h = 0.2;           % soli
f = @(x,y)x+y;
f_exact = @(x)exp(x)-x-1;                    % precīzs atrisinājums
egn = diff(y,x)==x+y;
y(x) = dsolve(egn,y(x_a)==ya)                % y(x) simboliskā funkcija
x_set = x_a:h:x_b; n = length(x_set);
sol_tab = zeros(n,4);  sol_tab(1,2) = ya;
sol_tab(:,1) = x_set;
sol_tab(:,3) = double(f_exact(x_set));       % precīzs atrisinājums
for i = 2:n
    ik = i-1;

    sol_tab(i,2) = sol_tab(ik,2)+h*f(sol_tab(ik,1),sol_tab(ik,2));
end
sol_tab(:,4) = abs(sol_tab(:,2)-sol_tab(:,3)); % absolūta kļūda
disp('      x_i       y_i      Precīzs    Absolūtā ')
disp('                         atrisinājums    kļūda')
disp(sol_tab)
```

$$y_{i+1} = y_i + h \cdot f(x_i, y_i)$$

```
   x_i        y_i       Precīzs     Absolūtā
                        atrisinājums    kļūda
         0          0         0          0
    0.2000          0    0.0214     0.0214
    0.4000     0.0400    0.0918     0.0518
    0.6000     0.1280    0.2221     0.0941
    0.8000     0.2736    0.4255     0.1519
    1.0000     0.4883    0.7183     0.2300
```

Kā redzams, aprēķinu kļūda pakāpeniski aug (punktā $x = 1$ relatīvā kļūda ir 31.9 %) divu iemeslu dēļ:

a) funkcija $y(x) = e^x - x - 1$ (uzdevuma precīzs atrisinājums) ir strauji augoša;
b) režģa solis $h = 0.2$ ir pietiekami liels.

Kā var paaugstināt precizitāti?

Acīmredzams risinājums ir samazināt soli, bet var arī izmantot citu formulu.

Problēma ar skaitlisko aprēķinu pēc Eilera formulas 7.1. piemērā ir saistīta ar to, ka funkcijas atvasinājums paliek konstants intervālā (x_i, x_{i+1}), bet precīzam atrisinājumam atvasinājumu rēķina pēc formulas $y'(x) = e^x - 1$.

Pārbaudīsim, vai ir vērts koriģēt atvasinājuma vērtību intervālā (x_i, x_{i+1}), lai paaugstinātu precizitāti.

Aplūkosim **uzlaboto Eilera metodi**:

$$y_{i+1} = y_i + \frac{s_1 + s_2}{2} h, \quad i = 0,1,\ldots,n, \tag{7.6}$$

kur

$$s_1 = f(x_i, y_i), \quad s_2 = f(x_{i+1}, y_i + hs_1) \tag{7.7}$$

Formulas (7.6), (7.7) var pārrakstīt šādi:

$$y^*_{i+1} = y_i + hf(x_i, y_i) \tag{7.8}$$

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, y^*_{i+1}) \right], \quad i = 0,1,\ldots,n \tag{7.9}$$

Metodi (7.8), (7.9) sauc par **prognozes un korekcijas metodi**.

Formulu (7.8) sauc par **prognozes daļu**, bet formulu (7.9) **par korekcijas daļu**.

Tādējādi, izmantojot parasto Eilera metodi (7.8), sākumā veic prognozi (aprēķina y^*_{i+1}), pēc tam izmanto korekcijas daļu (7.9).

Atrisināsim 7.1. piemērā doto uzdevumu ar uzlaboto Eilera metodi.

7.2. piemērs. Izmantojot uzlaboto Eilera metodi, atrisināt Košī problēmu $y' = x + y$, $y(0) = 0$ intervālā (0, 1) ar soli $h = 0.2$.

```
%% 7.2. piemērs.Uzlabotā Eilera metode.
clc, clearvars, format compact, syms  y(x)
x_a = 0; x_b = 1;              % intervāls (0,1)
ya = 0;                        % sākuma nosacījums y(0)=0
h = 0.2;
f = @(x,y)x+y;
f_exact = @(x)exp(x)-x-1;                    % precīzs atrisinājums
egn = diff(y,x)==x+y;
y(x) = dsolve(egn,y(x_a)==ya)                % y(x) simboliskā funkcija
x_set = x_a:h:x_b; n = length(x_set);
sol_tab = zeros(n,4);  sol_tab(1,2) = ya;
sol_tab(:,1) = x_set;
sol_tab(:,3) = double(f_exact(x_set));       % precīzs atrisinājums
for i = 2:n
    ik = i-1;
    y_star = sol_tab(ik,2)+h*f(sol_tab(ik,1),sol_tab(ik,2))
    sol_tab(i,2) = sol_tab(ik,2)+...
       h/2*( f(sol_tab(ik,1),sol_tab(ik,2))+f(sol_tab(ik+1,1),y_star));
 end
sol_tab(:,4)=abs(sol_tab(:,2)-sol_tab(:,3));  % absolūtā kļūda
disp('      x_i       y_i       Precīzs    Absolūtā ')
disp('                        atrisinājums    kļūda')
disp(sol_tab)
```

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, y^*_{i+1}) \right]$$

```
     x_i        y_i       Precīzs     Absolūtā
                       atrisinājums    kļūda
       0          0          0          0
  0.2000     0.0200     0.0214     0.0014
  0.4000     0.0884     0.0918     0.0034
  0.6000     0.2158     0.2221     0.0063
  0.8000     0.4153     0.4255     0.0102
  1.0000     0.7027     0.7183     0.0156
```

Salīdzinot rezultātus ar 7.1. piemēra risinājumu (izmantojot parasto Eilera metodi), redzams, ka kļūda ir būtiski mazāka.

Formulas (7.5) un (7.9) var pierakstīt vispārīgā veidā

$$y_{i+1} = y_i + hF\left(x_i, y_i\right), \tag{7.10}$$

kur Eilera metodei (7.5) atbilst funkcija $F(x_i, y_i) = f(x_i, y_i)$, bet uzlabotajai Eilera metodei funkcija

$$F\left(x_i, y_i\right) = \frac{1}{2}\left[f\left(x_i, y_i\right) + f\left(x_i + h, y_i + hf\left(x_i, y_i\right)\right)\right]$$

Izmantojot formulu (7.10), uzdevuma (7.1), (7.2) precīzu atrisinājumu punktā x_{i+1} var uzrakstīt šādi:

$$y\left(x_{i+1}\right) = y\left(x_i\right) + hF\left(x_i, y_i\right) + h\tau_i, \tag{7.11}$$

kur τ_i ir aproksimācijas kļūda.

Definīcija. Metode (7.11) ir metode ar kārtu p, ja visiem x_i, $a \le x_i \le b$ un pietiekami maziem h var atrast tādus skaitļus C un p, ka būs pareiza nevienādība

$$\left|\tau_i\right| \le Ch^p$$

Citiem vārdiem, $\tau_i = O\left(h^p\right)$.

Eilera metode (7.5) ir metode ar kārtu 1. Šo faktu var viegli pārbaudīt, salīdzinot (7.3) un (7.11).

Lai noteiktu uzlabotās Eilera metodes kārtu, salīdzināsim (7.3) un (7.11), turklāt funkciju $F(x_i, y_i)$ izvirzīsim Teilora rindā punkta (x_i, y_i) apkārtnē. Rezultātā iegūstam, ka uzlabotajā Eilera metodē pirmie trīs locekļi Teilora rindā (7.3) ir pareizi aprēķināti. Tas nozīmē, ka uzlabotā Eilera metode ir metode ar kārtu 2.

Aproksimācijas kļūdu vēl var samazināt, palielinot locekļu skaitu Teilora rindā (7.3). Rezultātā iegūst Runges-Kuta metodes. Pamatideja ir iekļaut formulā pēc iespējas vairāk locekļu Teilora rindā (7.3), nerēķinot atvasinājumus $y''\left(x\right)$, $y'''\left(x\right)$,…

Ilustrēsim šo Runges-Kuta metodes procedūru ar kārtu 2.

Pieņemsim, ka funkcijas y tuvināto vērtību punktā x_{i+1} var aprēķināt pēc formulas

$$y_{i+1} = y_i + h\left[a_1 f\left(x_i, y_i\right) + a_2 f\left(\hat{x}_i, \hat{y}_i\right)\right] \tag{7.12}$$

kur

$$\hat{x}_i = x_i + b_1 h, \quad \hat{y}_i = y_i + hb_2 f\left(x_i, y_i\right) \tag{7.13}$$

un a_1, a_2, b_1, b_2 ir nezināmie koeficienti.

Izmantosim Teilora formulu (7.3), kur y' ir aizstāts ar $f(x, y)$ un y'' ir aprēķināts pēc formulas (7.4).

Pieņemsim arī, ka $x = x_i$ un $y = y_i$:

$$y_{i+1} = y_i + hf\left(x_i, y_i\right) + \frac{h^2}{2!}\left[f_x\left(x_i, y_i\right) + f\left(x_i, y_i\right) f_y\left(x_i, y_i\right)\right] + O\left(h^3\right) \tag{7.14}$$

Izvirzīsim (7.12) pēc Teilora formulas punkta (x_i, y_i) apkārtnē:

$$y_{i+1} = y_i + h\left\{(a_1 + a_2) f(x_i, y_i) + a_2 h\left[b_1 f_x(x_i, y_i) + b_2 f(x_i, y_i) f_y(x_i, y_i)\right] + O\left(h^2\right)\right\} \quad (7.15)$$

Salīdzinot (7.14) un (7,15), iegūstam vienādojumu sistēmu:

$$\begin{cases} a_1 + a_2 = 1 \\ a_2 b_1 = \dfrac{1}{2} \\ a_2 b_2 = \dfrac{1}{2} \end{cases}$$

Vienādojumu sistēmai ir bezgalīgi daudz atrisinājumu (sistēmā ir trīs vienādojumi, bet četri nezināmie). Vienu nezināmo (piemēram, a_2) var pieņemt par brīvo parametru ($a_2 = c$).

Tādējādi

$$a_1 = 1 - c, b_1 = b_2 = \frac{1}{2c}, a_2 = c \quad (7.16)$$

Izmantojot (7.12), (7.13) un (7.16), iegūstam **Runges-Kuta metodes saimi ar kārtu 2**.

$$y_{i+1} = y_i + h\left[(1-c) f(x_i, y_i) + cf\left(x_i + \frac{h}{2c}, y_i + \frac{h}{2c} f(x_i, y_i)\right)\right] \quad (7.17)$$

Ja $c = 1/2$, formula (7.17) sakrīt ar uzlaboto Eilera metodi (7.8), (7.9).

Līdzīgā veidā var konstruēt augstāku kārtu Runges-Kuta metodes.

Viena no populārākajām metodēm ir Runges-Kuta metode ar kārtu 4.

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (7.18)$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1\right)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_2\right)$$

$$k_4 = f(x_i + h, y_i + h k_3)$$

Atrisināsim 7.1. piemērā doto uzdevumu ar formulu (7.18).

7.3. piemērs. Izmantojot Runges-Kuta metodi ar ceturto kārtu, atrisināt Košī problēmu intervālā (0, 1) ar soli $h = 0.2$.

```
%% 7.3. piemērs. Runges-Kuta metode ar ceturto kārtu.
clc, clearvars, format compact, syms  y(x)
x_a = 0; x_b = 1;     % intervāls (0,1)
ya = 0;               % sākuma nosacījums y(0)=0
h = 0.2;
f = @(x,y)x+y;
f_exact = @(x)exp(x)-x-1;                    % precīzs atrisinājums
egn = diff(y,x)==x+y;
y(x) = dsolve(egn,y(x_a)==ya)                % y(x) simboliskā funkcija
x_set = x_a:h:x_b; n = length(x_set);
sol_tab = zeros(n,4);  sol_tab(1,2) = ya;
sol_tab(:,1) = x_set;
sol_tab(:,3) = double(f_exact(x_set));       % precīzs atrisinājums
for i = 2:n
   ik = i-1;
   k1 = f(sol_tab(ik,1),sol_tab(ik,2));
   k2 = f(sol_tab(ik,1)+h/2,sol_tab(ik,2)+h/2*k1);
   k3 = f(sol_tab(ik,1)+h/2,sol_tab(ik,2)+h/2*k2);

   k4 = f(sol_tab(ik,1)+h,sol_tab(ik,2)+h*k3)
   sol_tab(i,2) = sol_tab(ik,2)+h/6*(k1+2*k2+2*k3+k4);

 end

sol_tab(:,4)=abs(sol_tab(:,2)-sol_tab(:,3)); % absolūta kļūda
disp('      x_i       y_i      Precīzs    Absolūtā ')
disp('                         atrisinājums    kļūda')
disp(sol_tab)
```

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)$$

$$k_4 = f(x_i + h, y_i + hk_3)$$

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

x_i	y_i	Precīzs atrisinājums	Absolūtā kļūda
0	0	0	0
0.2000	0.0214	0.0214	0.0000
0.4000	0.0918	0.0918	0.0000
0.6000	0.2221	0.2221	0.0000
0.8000	0.4255	0.4255	0.0000
1.0000	0.7183	0.7183	0.0000

Tabulā ir redzams, ka aprēķinu precizitāte ir diezgan augsta.

7.2. Aprēķini *MATLAB* vidē. Košī problēma

Aplūkosim Košī problēmas (7.1), (7.2) risinājumu *MATLAB* vidē. Aprēķinus veic, izmantojot shēmu:

sol = solver(@fun,x,y0,options)

solver – funkcijas nosaukums, kas raksturo konkrēto aprēķināšanas metodi (piemēram, **ode45** atbilst Runges-Kuta metodei ar kārtu 4 un 5)

Citi varianti ir **ode23**, **ode113** utt.

@fun – **function handle**, kas apraksta diferenciālvienādojuma labo pusi. Funkcija **fun** ir ir jāsaglabā kā atsevišķs m-fails ar nosaukumu **fun.m**

x – (vektors), kas nosaka skaitļus **a** un **b** (**a** un **b** ir intervāla apakšējā un augšējā robeža)

y0 – (vektors) sākuma nosacījums

options – opcijas, kas nosaka aprēķināšanas algoritma vadības struktūru

Piezīme. Ir vēl viens variants, kā var aprakstīt funkciju **fun** (sk. 7.5. piemēru).

7.4. piemērs. Atrisināt Košī problēmu intervālā (0, 2). Uzzīmēt funkcijas $y(x)$ grafiku.

$$y' = y\sin x + 2\sin 2x,\ y(0) = 1$$

Atrisinājums.

```
%% 7.4. piemērs. Košī problēma
% Pirmās kārtas diferenciālvienādojums
clc, clearvars, format compact
x_int = [0,2];    % intervāls
y0 = 1;           % sākuma nosacījums
% uzdevuma skaitliskā risināšana ar komandu ode45
sol = ode45(@fun_prob4,x_int,y0)
```

```
sol =
     solver: 'ode45'
    extdata: [1x1 struct]
          x: [1x11 double]
          y: [1x11 double]
      stats: [1x1 struct]
      idata: [1x1 struct]
```

Lai definētu diferenciālvienādojuma labo pusi, izmantosim ārējo funkciju **fun _ prob4**

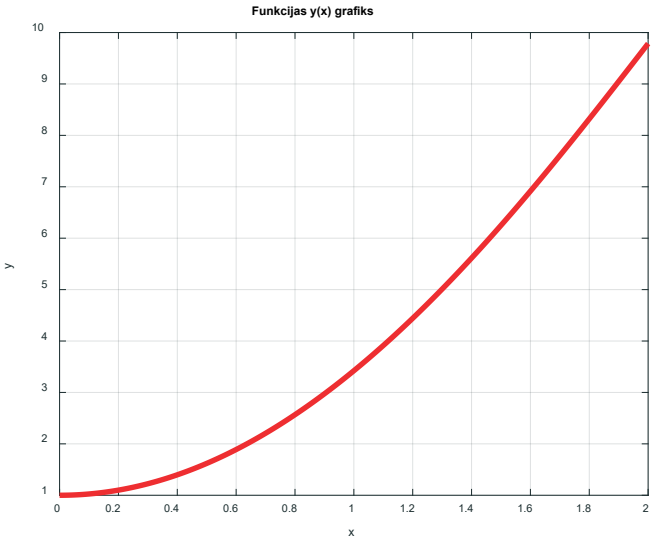
```
% ārējā funkcija (7.4. piemērs).
% definēsim diferenciālvienādojuma labo pusi.
function dydx = fun_prob4(x,y)
      dydx = y.*sin(x)+2*sin(2*x);
end
```

```
% 7.4. piemēra turpinājums
sol_x =sol.x'
sol_y =sol.y'
```

```
sol_x =
         0
    0.2000
    0.4000
    0.6000
    0.8000
    1.0000
    1.2000
    1.4000
    1.6000
    1.8000
    2.0000
```

```
sol_y =
    1.0000
    1.0999
    1.3979
    1.8895
    2.5675
    3.4224
    4.4426
    5.6135
    6.9156
    8.3205
    9.7858
```

```
% 7.4. piemēra turpinājums
x = (0:0.01:2);    % x vērtību vektors
y = deval(sol,x); % y vērtību vektors
plot(x,y,'r','LineWidth',3), xlabel('x'), ylabel('y')
title('Funkcijas y(x) grafiks'),grid on
```



7.2. att. 7.4. piemēra atrisinājuma grafiks.

7.5. piemērs. Atrisināt Košī problēmu intervālā (0, 1). Uzzīmēt funkcijas $y(x)$ grafiku un atrast vērtību $y(0.2)$.

$$y' = x + y,\ y(0) = 0$$

```
%% 7.5. piemērs. Košī problēma.
% Pirmās kārtas diferenciālvienādojums
clc, clearvars, format compact
% definēsim diferenciālvienādojuma labo pusi:
fun_prob5 = @(x,y)x+y;   % function handle
x_int = [0 1];           % intervāls
y0 = 0;                  % sākuma nosacījums
% uzdevuma skaitliskā risināšana ar komandu ode45
sol = ode45(fun_prob5,x_int,y0)
```

```
sol =
  struct with fields:
     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×11 double]
          y: [1×11 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

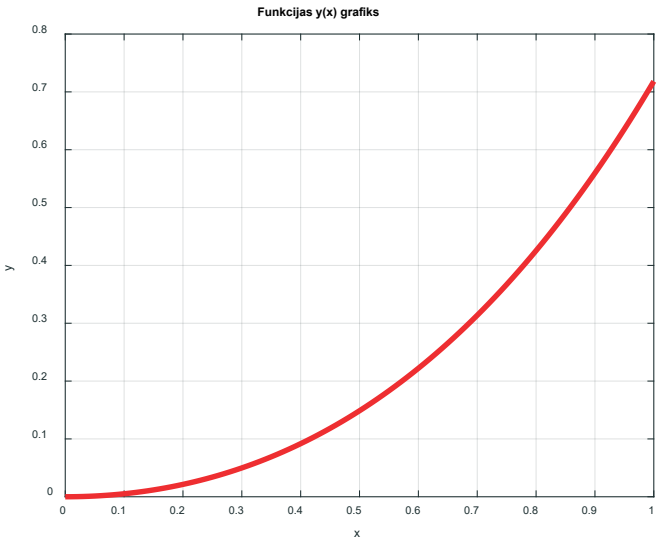
```
% 7.5. piemēra turpinājums
sol_x =sol.x'
sol_y =sol.y'
```

```
sol_x =
         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000
```

```
sol_y =
         0
    0.0052
    0.0214
    0.0499
    0.0918
    0.1487
    0.2221
    0.3138
    0.4255
    0.5596
    0.7183
```

```
% 7.5. piemēra turpinājums
y_02 = deval(sol,0.2)    % y vērtība punktā 0.2
x = (0:0.01:1);          % x vērtību vektors
y = deval(sol,x);        % y vērtību vektors
plot(x,y,'r','LineWidth',3), xlabel('x'), ylabel('y')
title('Funkcijas y(x) grafiks'),grid on
```

```
y_02 =
    0.0214
```



7.3. att. 7.5. piemēra atrisinājuma grafiks.

```
% 7.5. piemēra turpinājums
disp('Atbilde:')
fprintf(' funkcijas vērtība punktā(0.2) = %.4f \n ',y_02)
```

```
Atbilde:
 funkcijas vērtība punktā(0.2) = 0.0214
```

7.3. Diferenciālvienādojumu sistēmas

Aplūkosim pirmās kārtas parasto diferenciālvienādojumu sistēmu:

$$\begin{aligned}
\frac{\mathrm{d}\,y_1}{\mathrm{d}\,x} &= f_1\left(x, y_1, y_2, \ldots, y_n\right)\\
\frac{\mathrm{d}\,y_2}{\mathrm{d}\,x} &= f_2\left(x, y_1, y_2, \ldots, y_n\right)\\
&\cdots\\
\frac{\mathrm{d}\,y_n}{\mathrm{d}\,x} &= f_n\left(x, y_1, y_2, \ldots, y_n\right)
\end{aligned} \tag{7.19}$$

ar sākuma nosacījumiem:

$$\begin{aligned}
&y_1\big|_{x=x_0} = y_{10}\\
&y_2\big|_{x=x_0} = y_{20}\\
&\cdots\\
&y_n\big|_{x=x_0} = y_{n0}
\end{aligned} \tag{7.20}$$

Problēmu (7.19), (7.20) arī sauc par Košī problēmu. To var pārrakstīt šādi:

$$\frac{\mathrm{d}\,\boldsymbol{y}}{\mathrm{d}\,x} = \boldsymbol{f}\left(x, \boldsymbol{y}\right),\ \boldsymbol{y}\left(x_0\right) = \boldsymbol{y}_0, \tag{7.21}$$

kur

$$\boldsymbol{y} = \left(y_1 y_2 \ldots y_n\right)^T,\ \boldsymbol{f} = \left(f_1 f_2 \ldots f_n\right)^T,\ \boldsymbol{y}_0 = \left(y_{10} y_{20} \ldots y_{n0}\right)^T.$$

Košī problēma (7.21) ir līdzīga (pēc pieraksta) problēmai (7.1), (7.2).

Vienīgā starpība ir saistīta ar to, ka formulā (7.21) $\boldsymbol{y}$, $\boldsymbol{f}$ un $\boldsymbol{y}_0$ ir vektori, bet formulās (7.1) un (7.2) y, f un y_0 ir skalāri.

Tas nozīmē, ka jebkuru skaitlisko metodi (piemēram, Runges-Kuta metodi (7.18)) Košī problēmas (7.19), (7.20) risināšanai var pārrakstīt vektoru veidā. Piemēram, formulā (7.18) (ja to izmanto sistēmai (7.21)) $\boldsymbol{y}_{i+1}$, $\boldsymbol{y}_i$, $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, $\boldsymbol{k}_4$ un $\boldsymbol{f}$ būs vektori ar n komponentēm.

7.4. Aprēķini *MATLAB* vidē. Diferenciālvienādojumu sistēmas

MATLAB vidē Košī problēmu (7.21) pirmās kārtas parasto diferenciālvienādojumu sistēmai (7.21) risina līdzīgi problēmai (7.1), (7.2). Aprēķinus veic šādi:

sol = solver(@funsyst,x,y0,options)
solver – funkcijas nosaukums, kas raksturo konkrēto aprēķināšanas metodi (piemēram, **ode45** atbilst Runges-Kutta metodei ar kārtu 4 un 5) Citi varianti ir **ode23**, **ode113** utt.
@funsyst – **function handle**, kas apraksta diferenciālvienādojuma labo pusi. Funkcija **funsyst** ir jāsaglabā kā atsevišķs m-fails ar nosaukumu **funsyst.m**
x – (vektors), kas nosaka skaitļus **a** un **b** (**a** un **b** ir intervāla apakšējā un augšējā robeža)
y0 – sākuma nosacījums (vektors, kura komponentes ir sākuma nosacījumi katrai funkcijai sistēmā)
options – opcijas, kas nosaka aprēķināšanas algoritma vadības struktūru

Piezīme. Ir vēl viens variants, kā var aprakstīt funkciju **funsyst** (sk. 7.7. piemēru).

7.6. piemērs. Atrisināt Košī problēmu intervālā (0, 12). Uzzīmēt funkciju $y_1(x)$, $y_2(x)$ un $y_3(x)$ grafikus.

$$\begin{cases} \dfrac{\mathrm{d}\,y_1}{\mathrm{d}\,x} = y_2 y_3 \\ \dfrac{\mathrm{d}\,y_2}{\mathrm{d}\,x} = -y_1 y_3 \\ \dfrac{\mathrm{d}\,y_3}{\mathrm{d}\,x} = -0{,}5 y_1 y_2 \end{cases}, \; y_1(0) = 0,\; y_2(0) = 1,\; y_3(0) = 1$$

Atrisinājums.

```
%% 7.6. piemērs. Košī problēma.
% Pirmās kārtas diferenciālvienādojumu sistēma.
clc, clearvars, format compact
x_int = [0,12];          % intervāls
y0 = [0;1;1];            % sākuma nosacījums
% uzdevuma skaitliskā atrisināšana ar komandu ode45
sol = ode45(@fun_prob6,x_int,y0)
```

```
sol =
  struct with fields:
      solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×20 double]
          y: [3×20 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

Lai definētu diferenciālvienādojumu sistēmas labo pusi, izmantosim ārējo funkciju **fun _ prob6**

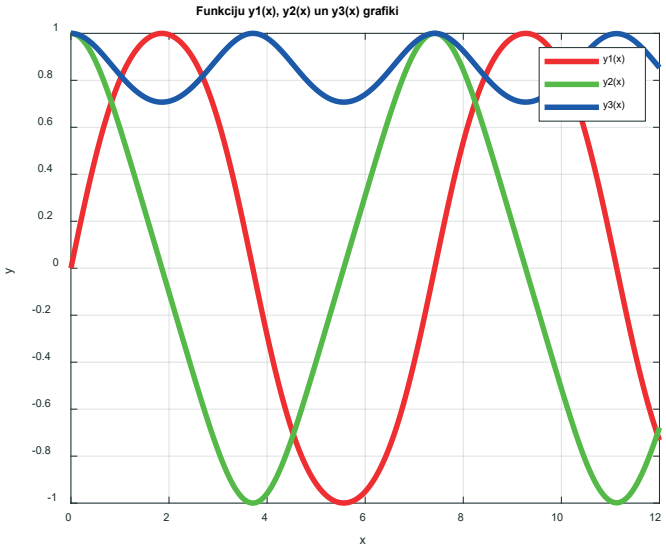
```
% ārējā funkcija (7.6. piemērs).
% definēsim diferenciālvienādojumu sistēmas labo pusi.
function dy_dx = fun_prob6(x,y)
   dy_dx = zeros(3,1);
   dy_dx(1) = y(2).*y(3);
   dy_dx(2) = -y(1).*y(3);
   dy_dx(3) = -0.5*y(1).*y(2);
end
```

```
% 7.6. piemēra
%turpinājums
sol_x =sol.x'
sol_y =sol.y'
```

```
sol_x =
         0
    0.0002
    0.0012
    0.0062
    0.0313
    0.1569
    0.6347
    1.2929
    2.2292
    3.3377
    4.0689
    4.8000
    5.5847
    6.7056
    7.5307
    8.3558
    9.0933
   10.1594
   11.3296
   12.0000
```

```
sol_y =
         0    1.0000    1.0000
    0.0002    1.0000    1.0000
    0.0012    1.0000    1.0000
    0.0062    1.0000    1.0000
    0.0313    0.9995    0.9998
    0.1560    0.9878    0.9939
    0.5771    0.8167    0.9129
    0.9181    0.3957    0.7604
    0.9638   -0.2650    0.7315
    0.3582   -0.9333    0.9672
   -0.3489   -0.9373    0.9693
   -0.8449   -0.5344    0.8018
   -0.9994    0.0158    0.7072
   -0.6317    0.7747    0.8945
    0.1131    0.9936    0.9969
    0.7716    0.6359    0.8382
    0.9918    0.1255    0.7128
    0.7854   -0.6186    0.8316
   -0.1981   -0.9786    0.9887
   -0.7321   -0.6787    0.8538
```

```
% 7.6. piemēra turpinājums
x = (0:0.01:12);        % x vērtību vektors
y = deval(sol,x);       % y1, y2 un y3 vērtību vektori
plot(x,y(1,:),'r', x,y(2,:),'g', x,y(3,:),'b','LineWidth',3)
legend('y1(x)','y2(x)','y3(x)'),xlabel('x'),ylabel('y')
title('Funkciju y1(x),y2(x) un y3(x) grafiki'),grid on
```



7.4. att. 7.6. piemēra atrisinājuma grafiki.

7.7. piemērs. Atrisināt Koši problēmu intervālā (0, 10). Uzzīmēt funkciju $x(t)$ un $y(t)$ grafikus. Atrast $x(2.7)$ un $y(2.7)$.

$$\begin{cases} x'(t) = -y(t) - x^2(t) \\ y'(t) = 2x(t) - y(t) \end{cases}, \; x(0) = 1, \; y(0) = 1$$

```
%% 7.7. piemērs. Koši problēma.
% Pirmās kārtas diferenciālvienādojumu sistēma.
clc, clearvars, format compact
% definēsim diferenciālvienādojumu sistēmas labo pusi:
dxy_dt = @(t,y) [-y(2)-y(1).^2; 2*y(1)-y(2)];  % function handle
t_int = [0,10];                                % intervāls
y0 = [1;1];                                    % sākuma nosacījums
% uzdevuma skaitliskā atrisināšana ar komandu ode45
sol = ode45(dxy_dt,t_int,y0)
```

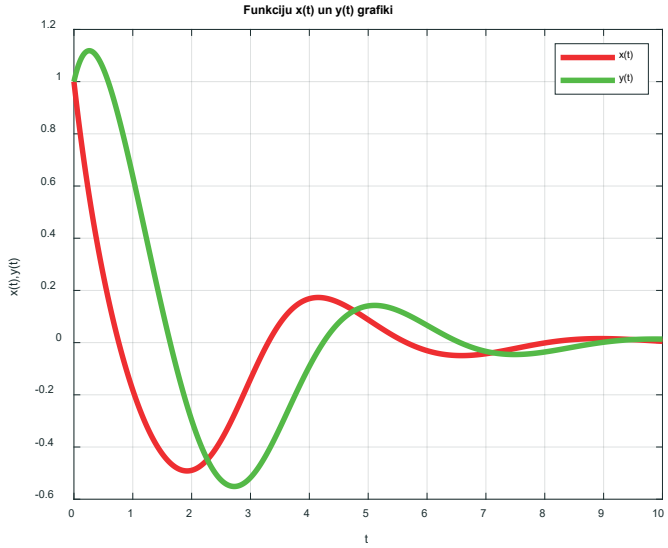
```
sol =
  struct with fields:
     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×20 double]
          y: [2×20 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

```
% 7.7. piemēra turpinājums
sol_t = sol.x'
sol_xy = sol.y'
```

```
sol_t =
         0
    0.1005
    0.5310
    1.0573
    1.5651
    2.2588
    2.9363
    3.5121
    4.0878
    4.6363
    5.1500
    5.6928
    6.2498
    6.8323
    7.3869
    7.9798
    8.5552
    9.1261
    9.6922
   10.0000
```

```
sol_xy =
    1.0000    1.0000
    0.8128    1.0770
    0.2237    1.0321
   -0.2187    0.5799
   -0.4444    0.0652
   -0.4503   -0.4445
   -0.1715   -0.5308
    0.0779   -0.3208
    0.1722   -0.0548
    0.1385    0.1038
    0.0656    0.1423
   -0.0044    0.1040
   -0.0438    0.0348
   -0.0470   -0.0233
   -0.0273   -0.0451
   -0.0020   -0.0366
    0.0128   -0.0143
    0.0149    0.0049
    0.0092    0.0133
    0.0050    0.0134
```

```
% 7.7. piemēra turpinājums
t = (0:0.01:10);       % t vērtību vektors
xy = deval(sol,t);     % x un y vērtību vektors
plot(t,xy(1,:),'r', t,xy(2,:),'g','LineWidth',3)
title('Funkciju x(t)un y(t) grafiki'),grid on
legend('x(t)','y(t)'),xlabel('t'),ylabel('x(t),y(t)')
xy_2_7 = deval(sol,2.7)
```



7.5. att. 7.7. piemēra atrisinājuma grafiki.

```
xy_2_7 =
   -0.2875
   -0.5509
```

```
% 7.7. piemēra turpinājums
fprintf('Atbilde. Funkciju vērtības punktā(2.7)  \n ')
fprintf('   x(2.7) = %.5f,   y(2.7) = %.5f \n',xy_2_7(:))
```

```
Atbilde. Funkciju vērtības punktā(2.7)
    x(2.7) = -0.28751,   y(2.7) = -0.55091
```

7.5. Augstāku kārtu diferenciālvienādojumi

Aplūkosim Košī problēmas risināšanu augstāku kārtu diferenciālvienādojumiem ar *MATLAB*.

Ir zināms, ka n-tās kārtas parasto diferenciālvienādojumu var pārveidot par pirmās kārtas diferenciālvienādojumu sistēmu (kas satur n vienādojumus).

Pēc pārveidojumiem iegūto sistēmu risina, izmantojot *MATLAB* rīkus Košī problēmas risināšanai parastiem diferenciālvienādojumiem.

7.6. Aprēķini *MATLAB* vidē. Augstāku kārtu diferenciālvienādojumi

7.8. piemērs. Atrisināt Košī problēmu intervālā (0, 10). Uzzīmēt funkcijas $y(x)$ grafiku un funkciju $y'(x)$, $y''(x)$ otrajā logā.

$$y''' + y'' + y' = -y^4,\ y(0) = 1,\ y'(0) = 0,\ y''(0) = 0$$

Atrisinājums.

Vispirms pārveidosim trešās kārtas diferenciālvienādojumu par pirmās kārtas vienādojumu sistēmu. Definēsim trīs nezināmās funkcijas: y_1, y_2 un y_3 (atzīmēsim, ka nezināmo funkciju skaits sakrīt ar dotā vienādojuma kārtu) pēc formulām:

$$\begin{cases} y_1 = y \\ y_2 = y' \\ y_3 = y'' \end{cases} \tag{7.22}$$

Pirmos divus vienādojumus sistēmā iegūstam, salīdzinot pirmā vienādojuma atvasinājumu formulā (7.22) ar otro vienādojumu:

$$y_1' = y',\ y_2 = y'$$

Tas nozīmē, ka

$$y_1' = y_2 \tag{7.23}$$

Analoģiski

$$y_2' = y'',\ y_3 = y''$$

Rezultātā iegūstam

$$y_2' = y_3 \tag{7.24}$$

Trešo vienādojumu sistēmā iegūst, izmantojot doto trešās kārtas diferenciālvienādojumu un funkcijas y_1, y_2 un y_3.

Pēdējā vienādojuma formulā (7.22) atvasinājums ir $y_3' = y'''$. Aizvietosim dotajā trešās kārtas vienādojumā y''' ar y_3', y'' ar y_3, y' ar y_2 un y ar y_1.

Rezultātā iegūstam sistēmas trešo vienādojumu šādā veidā:

$$y_3' = -y_3 - y_2 - y_1^4 \tag{7.25}$$

Izmantojot (7.23), (7.24) un (7.25), atrodam

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = -y_3 - y_2 - y_1^4 \end{cases} \tag{7.26}$$

Sākuma nosacījumus funkcijām y_1, y_2 un y_3 iegūst, izmantojot sākuma nosacījumus funkcijai y un funkciju y_1, y_2 un y_3 definīcijas.

Rezultāts ir

$$y_1(0) = 1,\ y_2(0) = 0,\ y_3(0) = 0 \tag{7.27}$$

Uzdevumu (7.26), (7.27) var risināt kā pirmās kārtas vienādojumu sistēmu.

```
%% 7.8. piemērs.Augstāku kārtu diferenciālvienādojumi
clc, clearvars, format compact, close all
x_int = [0 10];            % intervāls
y0 = [1;0;0];              % sākuma nosacījumi
% uzdevuma skaitliskā atrisināšana ar komandu ode45
sol = ode45(@fun_prob8,x_int,y0)
```

```
sol =
  struct with fields:
     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×21 double]
          y: [3×21 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

Lai definētu diferenciālvienādojumu sistēmas labo pusi, izmantosim ārējo funkciju **fun _ prob8**.

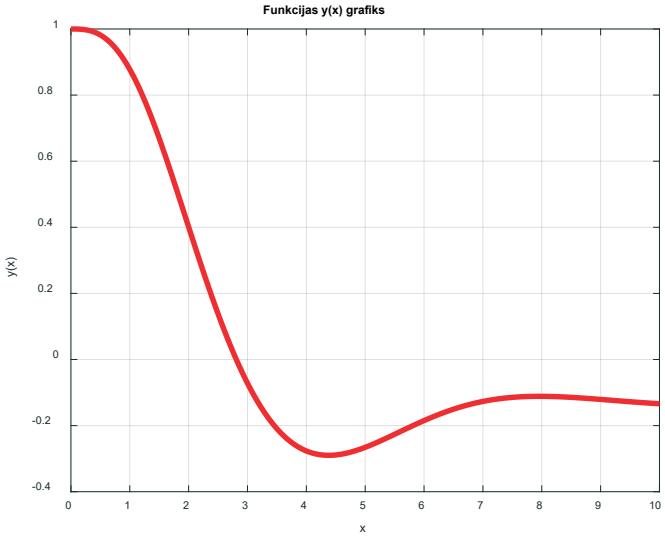
```
% ārēja funkcija (7.8. piemērs).
% definēsim diferenciālvienādojumu sistēmas labo pusi.
function dy_dx = fun_prob8(x,y)
   dy_dx = zeros(3,1);
   dy_dx(1) = y(2);
   dy_dx(2) = y(3);
   dy_dx(3) = -y(3)-y(2)-y(1).^4;
end
```

```
% 7.8. piemēra
% turpinājums
sol_x = sol.x'
sol_y = sol.y'
```

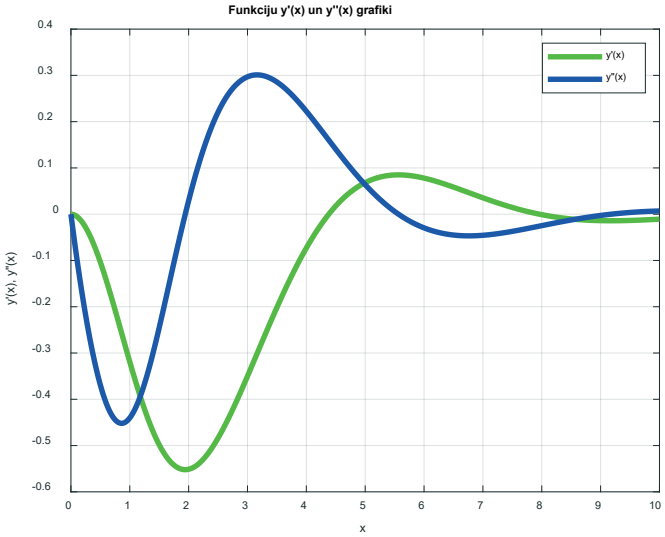
```
sol_x =
         0
    0.0002
    0.0012
    0.0062
    0.0313
    0.1569
    0.4300
    0.8507
    1.4401
    2.1027
    2.9046
    3.6638
    4.4069
    5.2198
    5.9145
    6.6335
    7.3632
    8.1372
    9.0027
    9.7385
   10.0000
```

```
sol_y =
    1.0000         0         0
    1.0000   -0.0000   -0.0002
    1.0000   -0.0000   -0.0012
    1.0000   -0.0000   -0.0062
    1.0000   -0.0005   -0.0309
    0.9994   -0.0117   -0.1446
    0.9882   -0.0789   -0.3341
    0.9203   -0.2521   -0.4519
    0.6986   -0.4826   -0.2771
    0.3447   -0.5457    0.0779
   -0.0370   -0.3781    0.2902
   -0.2386   -0.1561    0.2691
   -0.2899    0.0040    0.1558
   -0.2494    0.0791    0.0357
   -0.1916    0.0803   -0.0250
   -0.1430    0.0525   -0.0464
   -0.1169    0.0198   -0.0400
   -0.1118   -0.0042   -0.0213
   -0.1208   -0.0139   -0.0027
   -0.1308   -0.0124    0.0057
   -0.1338   -0.0107    0.0069
```

```
% 7.8. piemēra turpinājums
x = (0:0.01:10);          % x vērtību vektors
y = deval(sol,x);         % y,y' un y'' vērtību vektors
plot(x,y(1,:),'r', 'LineWidth',3)
xlabel('x'),ylabel('y(x)')
title('Funkcijas y(x) grafiks'),grid on
figure
plot(x,y(2,:),'g',x,y(3,:),'b','LineWidth',3)
legend('y''(x)','y''''(x)'),xlabel('x'),ylabel('y''(x), y''''(x)')
title('Funkciju y''(x) un y''''(x) grafiki'),grid on
```



7.6. att. 7.8. piemēra atrisinājuma grafiks.



7.7. att. 7.8. piemēra atvasinājumu grafiki.

7.9. piemērs. Atrisināt Košī problēmu intervālā (1, 4). Uzzīmēt funkciju $y(x)$, $y'(x)$, $y''(x)$ grafikus vienā logā. Atrast $y(1.87)$, $y'(1.87)$ un $y''(1.87)$.

$$x^3y''' - x^2y'' + 2xy' - 2y = x^3$$

$$y(1) = \frac{5}{4},\ y'(1) = \frac{3}{4},\ y''(1) = \frac{11}{2}$$

$$x^3y''' - x^2y'' + 2xy' - 2y = x^3 \longrightarrow y''' = \frac{y''}{x} - \frac{2y'}{x^2} + \frac{2y}{x^3} + 1$$

Rezultātā iegūstam pirmās kārtas vienādojumu sistēmu:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = \dfrac{y_3}{x} - \dfrac{2y_2}{x^2} + \dfrac{2y_1}{x^3} + 1 \end{cases}$$

Sākuma nosacījumi ir $y_1(1) = \frac{5}{4},\ y_2(1) = \frac{3}{4},\ y_3(1) = \frac{11}{2}$.

Definēsim diferenciālvienādojumu sistēmas labo pusi.

```
%% 7.9. piemērs. Augstāku kārtu diferenciālvienādojumi
clc, clearvars, format compact, close all
x_int = [1 4];            % intervāls
y0 = [5/4;3/4;11/2];      % sākuma nosacījumi
% uzdevuma skaitliskā atrisināšana ar komandu ode45
sol = ode45(@fun_prob9,x_int,y0)
```

```
sol =
  struct with fields:
     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×13 double]
          y: [3×13 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

Lai definētu diferenciālvienādojumu sistēmas labo pusi, izmantosim ārējo funkciju **fun _ prob9**.

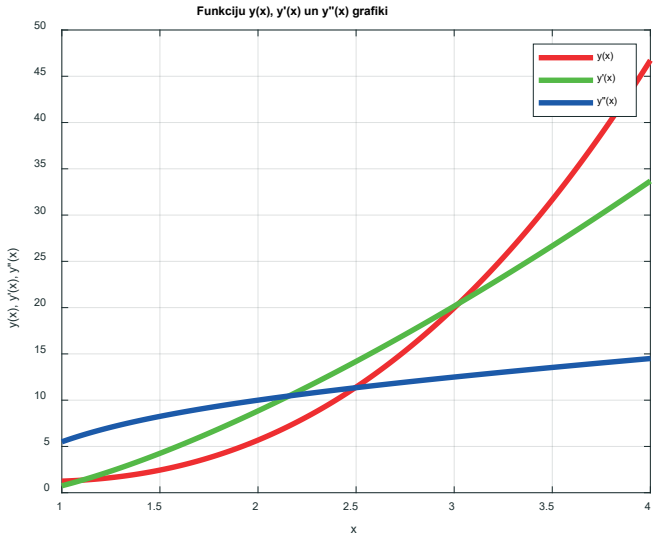
```
% ārējā funkcija (7.9. piemērs).
% definēsim diferenciālvienādojumu sistēmas labo pusi.
function dy_dx = fun_prob9(x,y)
   dy_dx = zeros(3,1);
   dy_dx(1) = y(2);
   dy_dx(2) = y(3);
   dy_dx(3) = y(3)./x-2*y(2)./x.^2+2*y(1)./x.^3+1;
end
```

```
% 7.9. piemēra
% turpinājums
sol_x = sol.x'
sol_y = sol.y'
```

```
sol_x =
    1.0000
    1.0274
    1.1644
    1.4644
    1.7644
    2.0644
    2.3644
    2.6644
    2.9644
    3.2644
    3.5644
    3.8644
    4.0000
```

```
sol_y =
    1.2500    0.7500    5.5000
    1.2726    0.9035    5.7011
    1.4529    1.7477    6.5938
    2.2984    3.9638    8.0994
    3.8702    6.5721    9.2461
    6.2726    9.4914   10.1902
    9.5913   12.6738   11.0090
   13.9001   16.0886   11.7447
   19.2655   19.7149   12.4226
   25.7487   23.5380   13.0586
   33.4069   27.5469   13.6633
   42.2946   31.7336   14.2440
   46.7289   33.6822   14.5000
```

```
% 7.9. piemēra turpinājums
x = (1:0.01:4);           % x vērtību vektors
y = deval(sol,x);         % y vērtību vektors
plot(x,y(1,:),'r',x,y(2,:),'g',x,y(3,:),'b','LineWidth',3)
legend('y(x)','y''(x)','y''''(x)')
xlabel('x'),ylabel('y(x), y''(x), y''''(x)')
title('Funkciju y(x), y''(x) un y''''(x) grafiki'),grid on
y_value = deval(sol,1.87)
```




```
y_value =
    4.6163
    7.5670
    9.5964
```

7.8. att. 7.9. piemēra atrisinājuma grafiki.

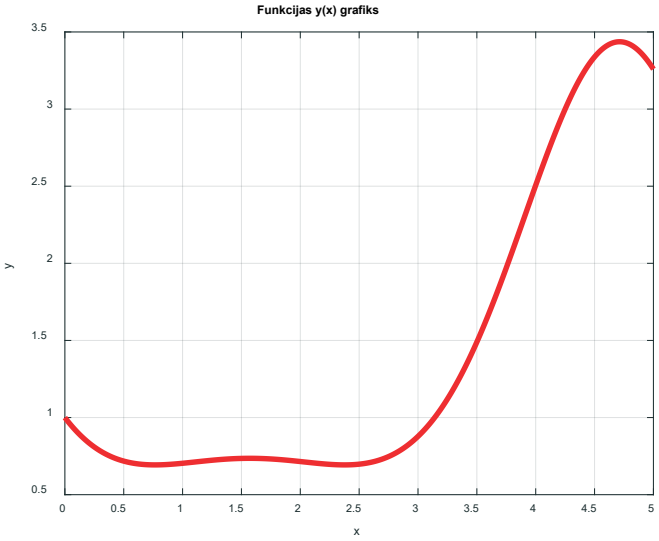
```
% 7.9. piemēra turpinājums
fprintf('Atbilde. Funkciju vērtības punktā(1.87)\n ')
fprintf('    y(1.87) = %.4f\n    y''(1.87) = %.4f\n',y_value(1:2))
fprintf('   y''''(1.87) = %.4f \n',y_value(3))
```

```
Atbilde. Funkciju vērtības punktā(1.87)
    y(1.87) = 4.6163
   y'(1.87) = 7.5670
  y''(1.87) = 9.5964
```

UZDEVUMI PATSTĀVĪGAI RISINĀŠANAI

7.1. uzdevums. Atrisināt Košī problēmu intervālā (0, 5). Uzzīmēt funkcijas $y(x)$ grafiku un atrast $y(2)$.

$$y' + y\cos x = \sin x \cos x,\ y(0) = 1$$



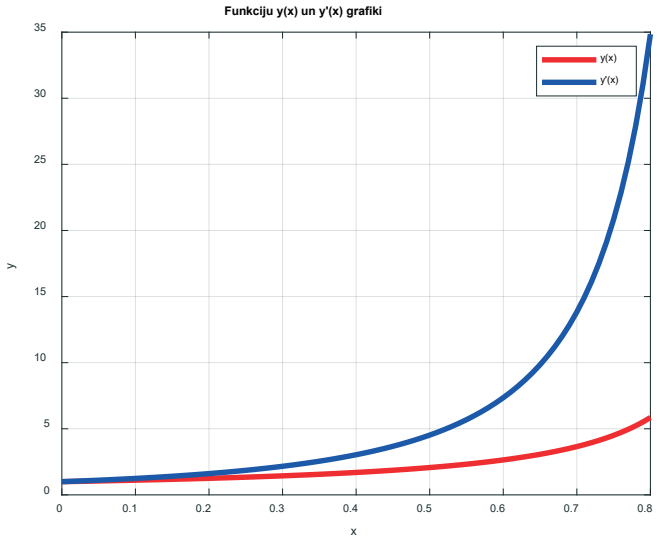

```
y_2 =
    0.7149
```

7.9. att. Atrisinājuma grafiks.

```
Atbilde:
 funkcijas vērtība punktā(2) = 0.7149
```

7.2. uzdevums. Atrisināt Košī problēmu intervālā (0, 0.8). Uzzīmēt funkciju $y(x)$, $y'(x)$ grafikus un atrast $y(0.5)$.

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



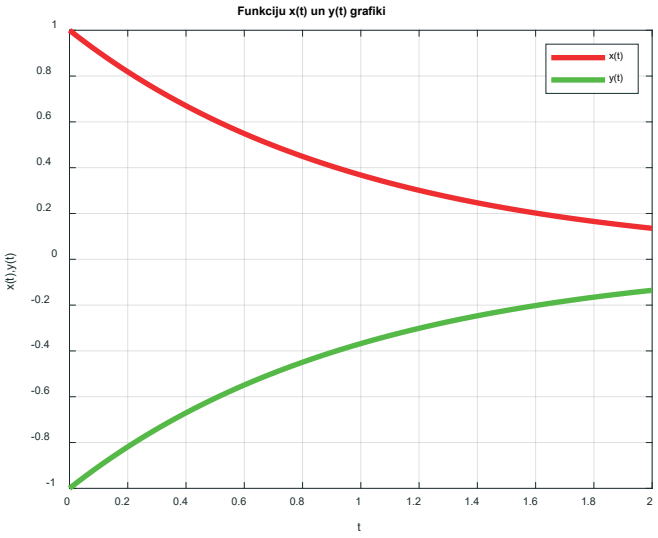

```
y_05 =
    2.0670
```

7.10. att. Funkcijas un pirmās kārtas atvasinājuma grafiki.

```
Atbilde:
  funkcijas vērtība punktā (0.5) = 2.0670
```

7.3. uzdevums. Atrisināt Košī problēmu intervālā (0, 2). Uzzīmēt funkciju $x(t)$ un $y(t)$ grafikus. Atrast $x(0.7)$ un $y(0.7)$.

$$\begin{cases} x'(t) = y(t) \\ y'(t) = 10y(t) + 11x(t) \end{cases}, \; x(0) = 1, \; y(0) = -1$$



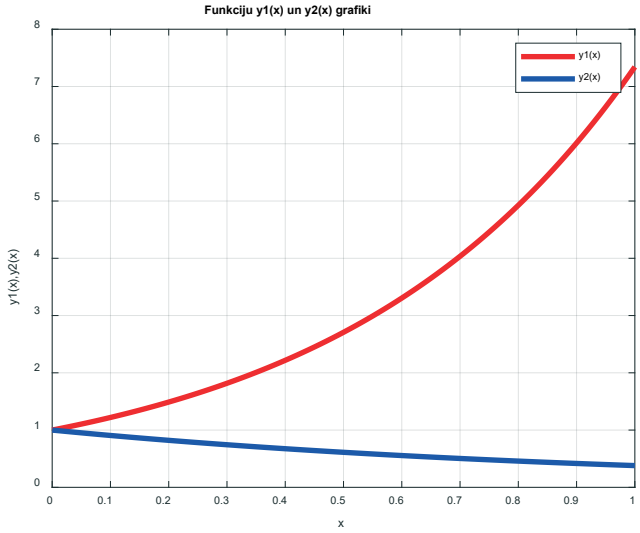
7.11. att. Funkciju grafiki.

```
xy_07 =
    0.4966
   -0.4966
```

```
Atbilde. Funkciju vērtības punktā (0.7)
    x(0.7) = 0.49659
    y(0.7) = -0.49659
```

7.4. uzdevums. Atrisināt Košī problēmu intervālā (0, 1). Uzzīmēt funkciju y_1(x) un y_2(x) grafikus. Atrast $y_1(0.3)$ un $y_2(0.3)$. Atrast maksimālo y_1 un y_2 vērtību intervālā (0,1).

$$\begin{cases} \dfrac{\mathrm{d}\, y_1}{\mathrm{d}\, x} = 2y_1 - 0.01y_1y_2 \\ \dfrac{\mathrm{d}\, y_2}{\mathrm{d}\, x} = -y_2 + 0.01y_1y_2 \end{cases}, \; y_1(0) = 1, \; y_2(0) = 1$$



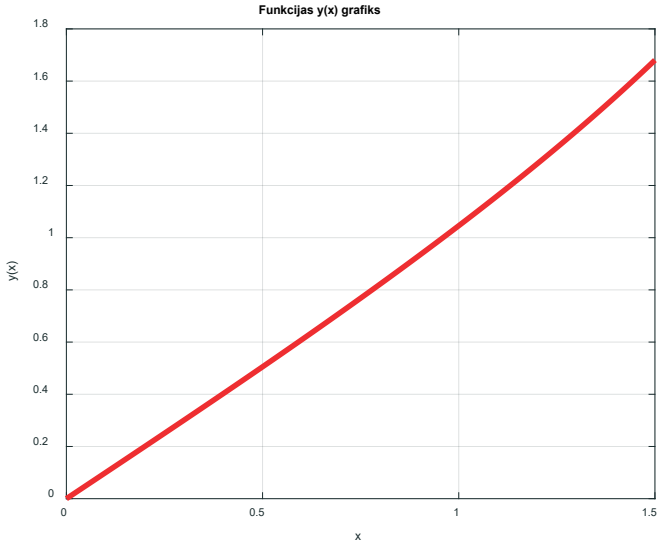
7.12. att. Funkciju grafiki.

```
y12_03 =
    1.8174
    0.7439
y1_max =
    7.3421
y2_max =
     1
```

```
Atbilde. Funkciju vērtības punktā (0.3)
    y1(0.3) = 1.81739
    y2(0.3) = 0.74387
    max:   y1 = 7.3421,   y2 = 1
```

7.5. uzdevums. Atrisināt Košī problēmu intervālā (0, 1.5). Uzzīmēt funkcijas $y(x)$ grafiku. Atrast $y(0.7)$ un $y'(0.7)$.

$$y'' + \frac{y}{x^2 - 4} = 0,\ y(0) = 0,\ y'(0) = 1$$



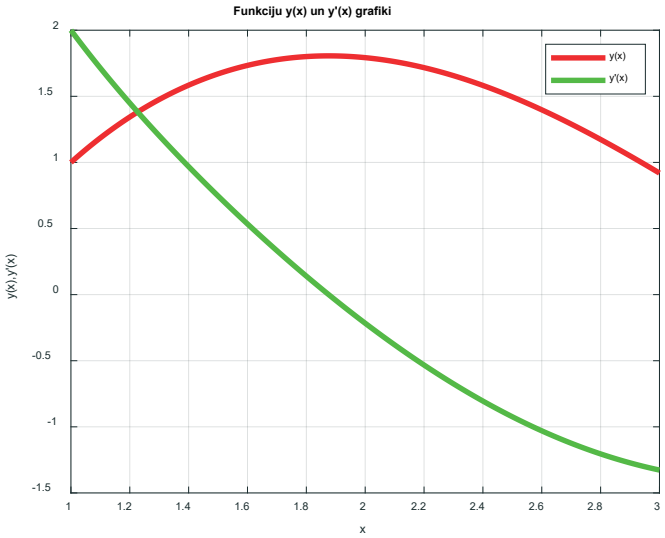

```
y_value =
    0.7149
    1.0660
```

7.13. att. Atrisinājuma grafiks.

```
Atbilde. Funkciju vērtības punktā (3.7)
    y(0.7) = 0.7149
   y'(0.7) = 1.0660
```

7.6. uzdevums. Atrisināt Košī problēmu intervālā (1, 3). Uzzīmēt funkciju $y(x)$ un $y'(x)$ grafikus. Atrast $y(2.3)$ un $y'(2.3)$.

$$xy'' + y' + xy = 0,\ y(1) = 1,\ y'(1) = 2$$



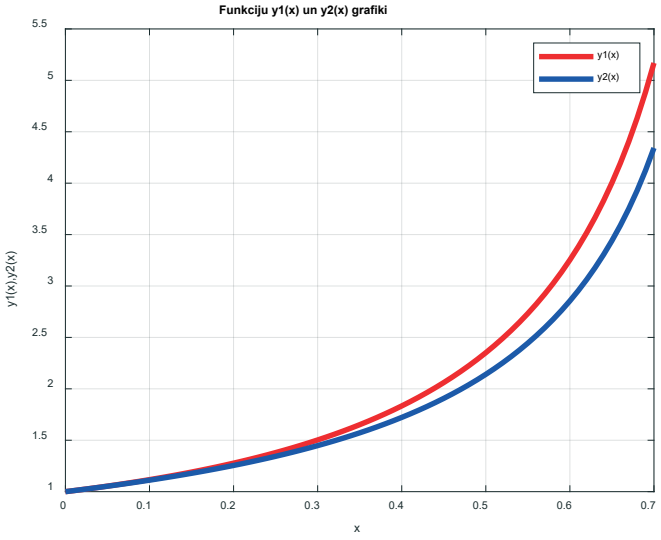

```
y_value =
    1.6563
   -0.6746
```

7.14. att. Funkcijas un atvasinājuma grafiki.

```
Atbilde. Funkciju vērtības punktā (2.3)
    y(2.3) = 1.6563
   y'(2.3) = -0.6746
```

7.7. uzdevums. Atrisināt Košī problēmu intervālā (0, 0.7). Uzzīmēt funkciju $y_1(x)$ un $y_2(x)$ grafikus. Atrast $y_1(0.25)$ un $y_2(0.25)$. Atrast minimālo y_1 un maksimālo y_2 vērtību intervālā (0, 0.7).

$$\begin{cases} \dfrac{\mathrm{d}\,y_1}{\mathrm{d}\,x} = y_1^{\,2} + xy_2 \\ \dfrac{\mathrm{d}\,y_2}{\mathrm{d}\,x} = x^2 + y_1 y_2 \end{cases}, \; y_1(0) = 1,\; y_2(0) = 1$$



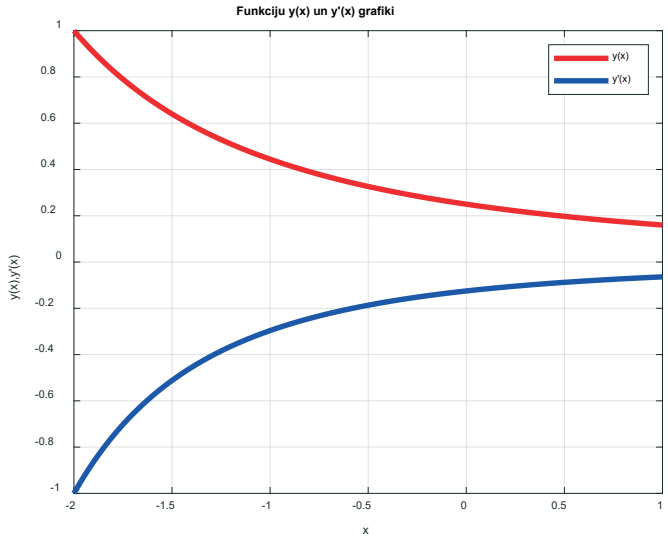

```
y12_03 =
    1.3799
    1.3436
y1_min =
     1
y2_max =
    4.3424
```

7.15. att. Funkciju grafiki.

```
Atbilde. Funkciju vērtības punktā (0.25)
    y1(0.25) = 1.37987
    y2(0.25) = 1.34364
   y1_min = 1.0000,   y2_max = 4.3424
```

7.8. uzdevums. Atrisināt Košī problēmu intervālā (–2, 1). Atrast y un y' vērtības punktā $x = -1.25$. Uzzīmēt funkciju (x) un (x) grafikus.

$$2y'' - 3y^2 = 0,\; y(-2) = 1,\; y'(-2) = -1$$



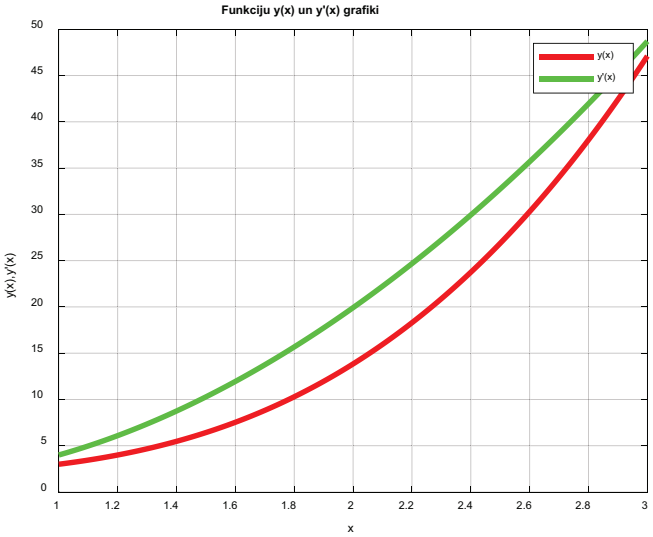

```
y_value =
    0.5289
   -0.3847
```

7.16. att. Funkcijas un atvasinājuma grafiki.

```
Atbilde. Funkciju vērtības punktā (-1.25)
    y(-1.25) = 0.52892
   y'(-1.25) = -0.38469
```

7.9. uzdevums. Atrisināt Košī problēmu intervālā (1, 3). Atrast $y(1.86)$ un $y'(2.68)$ vērtības. Uzzīmēt funkciju $y(x)$ un $y'(x)$ grafikus.

$$x^2 y'' - xy' + y = 8x^3,\ y(1) = 3,\ y'(1) = 4$$




```
y_value1 =
   11.2669
   16.8959
y_value2 =
   33.2517
   38.1369
```

7.17. att. Funkcijas un atvasinājuma grafiki.

```
Atbilde. Funkciju vērtības punktos
    y(1.86) = 11.26689
   y'(2.68) = 38.13695
```

2023