Tiesinių lygčių sistemų sprendimas:
Gauso ir LU skaidos algoritmai

Temoje aiškinama:

- **Tiesinių lygčių sistemų (TLS) užrašymas matricomis, svarbiausi matricų algebros veiksmai;**
- **Gauso algoritmas TLS sprendimui;**
- **Kaip taikyti Gauso algoritmą, kai TLS matrica singuliari;**
- **Sprendinio tikslumo patikrinimas, panaudojant vektorių normas;**
- **Gauso algoritmo sudėtingumas;**
- **Kiti kintamųjų eliminavimu paremti algoritmai;**
- **LU skaidos algoritmas**

Tiesinių lygčių sistemų (TLS) užrašymas matricomis, svarbiausi matricų algebros veiksmai

Algebrinės lygtys

Viena lygtis

Tiesinės algebrinės lygtys
(vienas sprendinys)

$$ax + b = 0;$$

$$x = -\frac{b}{a}$$

Netiesinės algebrinės ir transcendentinės lygtys
(keli sprendiniai)

$$f(x) = 0$$

$$f(x) = ax^2 + bx + c = 0;$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$f(x) = ax^6 + \sin^2 x + \ln(x+2) = 0;$$

$$x = ???$$

Lygčių sistema

Tiesinių algebrinių lygčių sistemos
(vienas sprendinys, be galo daug sprendinių, nėra sprendinių)

$$[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}$$

$$[\mathbf{A}][\mathbf{X}] = [\mathbf{B}]$$

Netiesinių algebrinių ir transcendentinių lygčių sistemos (keli sprendiniai, be galo daug sprendinių, nėra sprendinių)

Tiesinė lygčių sistema 1

$$\begin{cases} 2x_1 + x_2 = 4; \\ x_1 - x_2 = -1 \end{cases}$$

Įprastinis 2 lygčių sistemos pavidalas

$$\begin{bmatrix} \boxed{2 \quad 1} \\ \boxed{1 \quad -1} \end{bmatrix} \left\{ \boxed{\begin{matrix} x_1 \\ x_2 \end{matrix}} \right\} = \left\{ \boxed{\begin{matrix} 4 \\ -1 \end{matrix}} \right\}$$

Lygčių sistemos pavidalas matricomis

$$\begin{cases} \boxed{2} * \boxed{x_1} + \boxed{1} * \boxed{x_2} = 4 \\ \boxed{1} * \boxed{x_1} \boxed{-1} * \boxed{x_2} = -1 \end{cases}$$

Ryšys tarp šių pavidalų nusakomas *matricų daugybos* veiksmu

Tiesinė lygčių sistema 2

Koeficientų matrica

Nežinomųjų vektorius

Laisvųjų narių vektorius

$$[\mathbf{A}]_{m\times n}\{\mathbf{x}\}_{n\times 1}=\{\mathbf{b}\}_{m\times 1}$$

Nežinomųjų skaičius

Lygčių skaičius

$$\{\mathbf{x}\}?$$

Dažniausiai sutinkamas atvejis m=n

Tiesinė lygčių sistema

$$[\mathbf{A}]_{m\times n}\{\mathbf{x}\}_{n\times 1}=\{\mathbf{b}\}_{m\times 1}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{Bmatrix}$$

Matricų daugybos veiksmas

- **Matricų daugyba:**

$$[\mathbf{C}]_{m\times p}=[\mathbf{A}]_{m\times n}[\mathbf{B}]_{n\times p}\quad\Rightarrow\quad c_{ik}=\sum_{j=1}^{n}a_{ij}b_{jk}$$

Pirmojo daugiklio 1-os eilutės ir antrojo daugiklio 3-io stulpelio ***skaliarinė sandauga***

$$\begin{array}{c}1\\ \\ \end{array}\begin{bmatrix}\boxed{1 & -2 & 4}\\5 & 2 & 3\end{bmatrix}_{2\times3}\begin{array}{c} \\ \end{array}\begin{array}{c}\\ \end{array}\begin{array}{cccc} & & 3 & \\ \end{array}$$

$$\begin{bmatrix}1 & -2 & 4\\5 & 2 & 3\end{bmatrix}_{2\times3}\begin{bmatrix}2 & 3 & \boxed{4} & -2\\1 & 5 & \boxed{-1} & 1\\2 & 4 & \boxed{2} & -3\end{bmatrix}_{3\times4}=$$

$$=\begin{bmatrix}1\times2-2\times1+4\times2 & 1\times3-2\times5+4\times4 & \boxed{1\times4+2\times1+4\times2} & -1\times2-2\times1-4\times3\\5\times2+2\times1+3\times2 & 5\times3+2\times5+3\times4 & 5\times4-2\times1+3\times2 & -5\times2+2\times1-3\times3\end{bmatrix}=$$

$$=\begin{bmatrix}8 & 9 & \boxed{14} & -14\\18 & 35 & 24 & -17\end{bmatrix}$$

$[\mathbf{B}]_{n\times p}[\mathbf{A}]_{m\times n}$ Veiksmas neapibrėžtas. Daugiklių sukeisti vietomis negalima net ir tuo atveju, kai matricos kvadratinės

$$\begin{bmatrix}2 & 3 & 4 & -2\\1 & 5 & -1 & 1\\2 & 4 & 2 & -3\end{bmatrix}_{3\times4}\begin{bmatrix}1 & -2 & 4\\5 & 2 & 3\end{bmatrix}_{2\times3}$$

$$4\neq2\ \ (!)$$

- **Matricų transponavimas:**

$$[\mathbf{A}]_{m\times n} = \begin{bmatrix} 1 & -2 & 4 \\ 5 & 2 & 3 \end{bmatrix}_{2\times 3}$$

$$\left([\mathbf{A}]^{T}\right)_{n\times m} = \begin{bmatrix} 1 & 5 \\ -2 & 2 \\ 4 & 3 \end{bmatrix}_{3\times 2}$$

- Išeities matricos eilutės įrašomos stulpeliais į rezultato matricą;
- Matricų matmenys susikeičia vietomis

- **Matricų sandaugos transponavimas:** $[\mathbf{A}][\mathbf{B}] \neq [\mathbf{B}][\mathbf{A}]$

$$[\mathbf{C}]_{m\times p} = [\mathbf{A}]_{m\times n}[\mathbf{B}]_{n\times p} \Longrightarrow \left([\mathbf{C}]^T\right)_{p\times m} = \left([\mathbf{B}]^T\right)_{p\times n}\left([\mathbf{A}]^T\right)_{n\times m}$$

$$[\mathbf{C}]_{2\times 4} = [\mathbf{A}]_{2\times 3}[\mathbf{B}]_{3\times 4} = \begin{bmatrix} 1 & -2 & 4 \\ 5 & 2 & 3 \end{bmatrix}_{2\times 3} \begin{bmatrix} 2 & 3 & 4 & -2 \\ 1 & 5 & -1 & 1 \\ 2 & 4 & 2 & -3 \end{bmatrix}_{3\times 4} =$$

$$= \begin{bmatrix} 1\times 2-2\times 1+4\times 2 & 1\times 3-2\times 5+4\times 4 & 1\times 4+2\times 1+4\times 2 & -1\times 2-2\times 1-4\times 3 \\ 5\times 2+2\times 1+3\times 2 & 5\times 3+2\times 5+3\times 4 & 5\times 4-2\times 1+3\times 2 & -5\times 2+2\times 1-3\times 3 \end{bmatrix} =$$

$$= \begin{bmatrix} 8 & 9 & 14 & -16 \\ 18 & 37 & 24 & -17 \end{bmatrix}_{2\times 4}$$

$$\left([\mathbf{C}]^T\right)_{p\times m} = \left([\mathbf{B}]^T\right)_{p\times n}\left([\mathbf{A}]^T\right)_{n\times m} = \begin{bmatrix} 2 & 1 & 2 \\ 3 & 5 & 4 \\ 4 & -1 & 2 \\ -2 & 1 & -3 \end{bmatrix}_{4\times 3} \begin{bmatrix} 1 & 5 \\ -2 & 2 \\ 4 & 3 \end{bmatrix}_{3\times 2} =$$

$$= \begin{bmatrix} 2\times 1-1\times 2+2\times 4 & 2\times 5+1\times 2+2\times 3 \\ 3\times 1-5\times 2+4\times 4 & 3\times 5+5\times 2+4\times 3 \\ 4\times 1+1\times 2+2\times 4 & 4\times 5-1\times 2+2\times 3 \\ -2\times 1-1\times 2-3\times 4 & -2\times 5+1\times 2-3\times 3 \end{bmatrix} = \begin{bmatrix} 8 & 18 \\ 9 & 37 \\ 14 & 24 \\ -16 & -17 \end{bmatrix}_{4\times 2}$$

- **Kai kurie matricų daugybos veiksmo taikymai:**

Vektorių skaliarinė sandauga:

$$\{\mathbf{a}\} = \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix}; \quad \{\mathbf{b}\} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}; \qquad \{\mathbf{a}\}^T \{\mathbf{b}\} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Vektoriaus “ilgis” (Euklido norma):

$$\{\mathbf{a}\} = \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix}; \qquad \sqrt{\{\mathbf{a}\}^T \{\mathbf{a}\}} = \sqrt{a_1^{\ 2} + a_2^{\ 2} + \cdots + a_n^{\ 2}}$$

Tiesinė lygčių sistema su daugeliu laisvųjų narių vektorių

$$[\mathbf{A}]_{m\times n}[\mathbf{X}]_{n\times p}=[\mathbf{B}]_{m\times p}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mm} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mp} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ b_{31} & b_{32} & \cdots & b_{3p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix}$$

Nežinomųjų vektoriai ir jiems atitinkantys laisvųjų narių vektoriai

Iš esmės, turime *p* lygčių sistemų, kurių koeficientų matricos vienodos

Matricos MATLAB terpėje 1

- Kiekvienas kintamasis MATLAB yra suvokiamas kaip *matrica arba vektorius*;
- Skaliarinis dydis yra *matricos atskiras atvejis*, kai jos išmatavimas 1x1;
- Kai tarpusavyje dauginami du kintamieji, MATLAB pagal nutylėjimą atlieka *matricų daugybos* veiksmą. Todėl matricų išmatavimai turi būti suderinti. Tuo turi pasirūpinti *programuotojas*

Matricos MATLAB terpėje 2

```
A= [1 -2  4 ; 5  2  3];
B=[2  3  4 -2 ; 1  5 -1  1 ; 2  4  2 -3];
C= A*B
D=B'*A'
```

```
C =

     8     9    14   -16
    18    37    24   -17

D =

     8    18
     9    37
    14    24
   -16   -17
```

Matricos Python terpėje 1

```
import numpy as np
A=np.array([[1, -2,  4],[5,  2,  3]])
B=np.array([[2,  3,  4, -2],[1,  5, -1,  1],[2,  4,  2, -3]])
print(A);print(B)
C=A.dot(B)
print(C)
D=(B.transpose()).dot(A.transpose())
print(D)
```

```
[[ 1 -2  4]
 [ 5  2  3]]
[[ 2  3  4 -2]
 [ 1  5 -1  1]
 [ 2  4  2 -3]]
[[  8   9  14 -16]
 [ 18  37  24 -17]]
[[  8  18]
 [  9  37]
 [ 14  24]
 [-16 -17]]
The thread 'MainThread' (0x1) has exited with code 0 (0x0).
```

```
import numpy as np
print('----------list----------------')

A=[[1,2,3], [4,6,8], [10,1,3] ];print(A)
a= [[3], [9], [8]]
b= [3,9,8]
print(np.dot(A,a))  # vektorius a yra stulpelis
print(np.dot(A,b)) # interpretuoja eilute b kaip stulpeli,
                    # kad butu galima dauginti, taciau
                    # rezultatas yra eilute (!!)
print(np.dot(b,A))
print(np.dot(np.transpose(a),A))
print(A[:][1:3])
#!!  interpretuoja antraja indeksu pozicija kaip eilutes
print('------------------------------')
```

```
----------list----------------
 [[1, 2, 3], [4, 6, 8], [10, 1, 3]]
 [[ 45]
  [130]
  [ 63]]
 [ 45 130 63]
 [119 68 105]
 [[119 68 105]]
 [[4, 6, 8], [10, 1, 3]]
 ------------------------------
```

Matricos Python terpėje 2

- Matricas ir vektorius Python galima sukurti kaip sąrašus (list), arba panaudojant np.matrix ir np.array metodus.
- A=[[1, -2, 4],[5, 2, 3]] sukuria sąrašus *list* , su kuriais matricų daugybos veiksmai atliekami funkcijos np.dot pagalba;
- Jeigu matricos kuriamos taikant np.matrix arba np.array , sukurti objektai turi metodą dot, pavyzdžiui A.dot(B);
- Matricų išmatavimai turi būti suderinti

Tiesinių algebrinių lygčių sistemų pavyzdžiai ir analitiniai sprendimo metodai

$$\begin{cases} 2x + y = 4; \\ x - y = -1 \end{cases}$$

Kramerio metodas:

$$\Delta = \begin{vmatrix} 2 & 1 \\ 1 & -1 \end{vmatrix} = -3; \quad \Delta_x = \begin{vmatrix} 4 & 1 \\ -1 & -1 \end{vmatrix} = -3; \quad \Delta_y = \begin{vmatrix} 2 & 4 \\ 1 & -1 \end{vmatrix} = -6;$$

$$x = \frac{\Delta_x}{\Delta} = 1; \; y = \frac{\Delta_y}{\Delta} = 2.$$

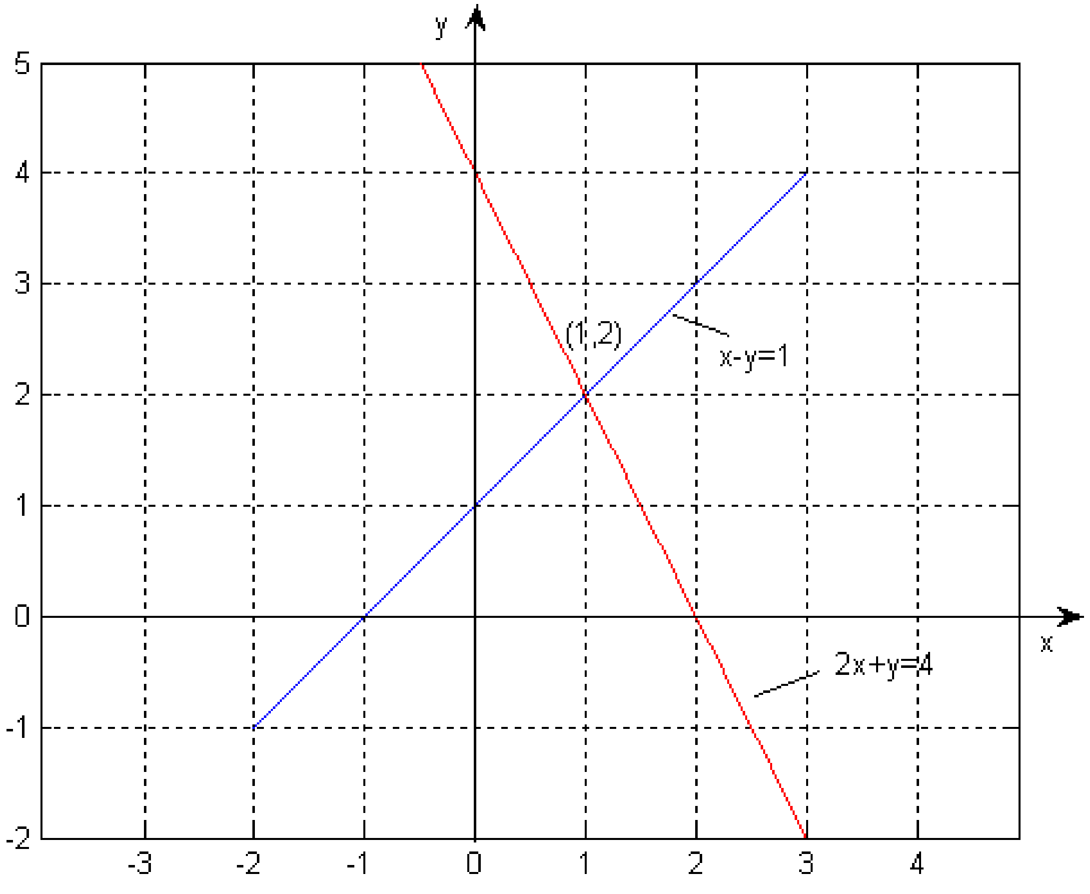
Kintamųjų eliminavimo metodas:

$$\begin{bmatrix} 2 & 1 & 4 \\ 1 & -1 & -1 \end{bmatrix} : 2 \Rightarrow \begin{bmatrix} 1 & 1/2 & 2 \\ 1 & -1 & -1 \end{bmatrix} \begin{matrix} \downarrow \\ \leftarrow - \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1/2 & 2 \\ 0 & -3/2 & -3 \end{bmatrix}$$

$$\begin{cases} x + & y/2 & = & 2 \\ & -3y/2 & = & -3 \end{cases}; \qquad y = 2; \; x = 1.$$

Grafinė interpretacija

$$\begin{cases} 2x + y = 4; \\ x - y = -1 \end{cases}$$



Lygčių sistema turi vienintelį sprendinį

$$\begin{cases} 4x + 6y = 10; \\ 2x + 3y = 6 \end{cases}$$

Kramerio metodas:

$\Delta = \begin{vmatrix} 4 & 6 \\ 2 & 3 \end{vmatrix} = 0; \quad \Delta_x = \begin{vmatrix} 10 & 6 \\ 6 & 3 \end{vmatrix} = -6; \quad \Delta_y = \begin{vmatrix} 4 & 10 \\ 2 & 6 \end{vmatrix} = 4;$ sprendinių nėra

Kintamųjų eliminavimo metodas:

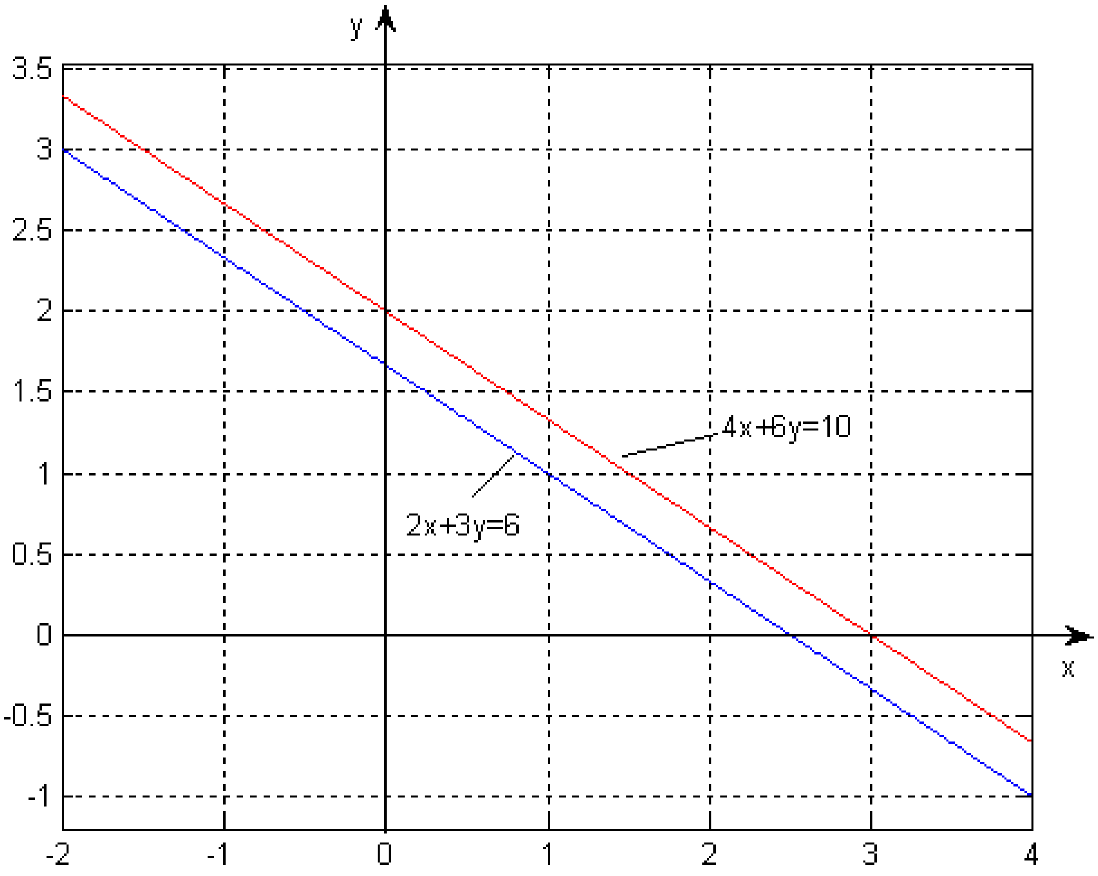
$$\begin{bmatrix} 4 & 6 & 10 \\ 2 & 3 & 6 \end{bmatrix} : 2 \rightarrow \begin{bmatrix} 2 & 3 & 5 \\ 2 & 3 & 6 \end{bmatrix} \begin{matrix} \downarrow \\ \leftarrow - \end{matrix} \rightarrow \begin{bmatrix} 2 & 3 & 5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{cases} 2x + 3y = 5 \\ 0 + 0 = 1 \end{cases};$$

antroji lygybė negali būti tenkinama, todėl sprendinių nėra

Grafinė interpretacija

$$\begin{cases} 4x + 6y = 10; \\ 2x + 3y = 6 \end{cases}$$



Lygčių sistema sprendinių neturi

$$\begin{cases} 4x + 6y = 10; \\ 2x + 3y = 5 \end{cases}$$

Kramerio metodas:

$$\Delta = \begin{vmatrix} 4 & 6 \\ 2 & 3 \end{vmatrix} = 0; \quad \Delta_x = \begin{vmatrix} 10 & 6 \\ 5 & 3 \end{vmatrix} = 0; \quad \Delta_y = \begin{vmatrix} 4 & 10 \\ 2 & 5 \end{vmatrix} = 0;$$

be galo daug sprendinių, kadangi visi determinantai lygūs nuliui

Kintamųjų eliminavimo metodas:

$$\begin{bmatrix} 4 & 6 & 10 \\ 2 & 3 & 5 \end{bmatrix} : 2 \rightarrow \begin{bmatrix} 2 & 3 & 5 \\ 2 & 3 & 5 \end{bmatrix} \begin{matrix} \downarrow \\ \leftarrow - \end{matrix} \rightarrow \begin{bmatrix} 2 & 3 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{cases} 2x + 3y = 5 \\ 0 + 0 = 0 \end{cases};$$

Lieka viena lygtis; be galo daug sprendinių

$$\begin{cases} 4x + 6y = 10; \\ 2x + 3y = 5 \end{cases}$$

$$\begin{cases} 2x \ + \ 3y = 5 \\ 0 \ + \ \ 0 \ = 0 \end{cases} ;$$

“Be galo daug sprendinių” nereiškia, kad sprendiniu gali būti bet kokia skaičių pora (!). Tokiu atveju galime vieno nežinomojo reikšmę pasirinkti laisvai, o kitus išreikšti per šią reikšmę:

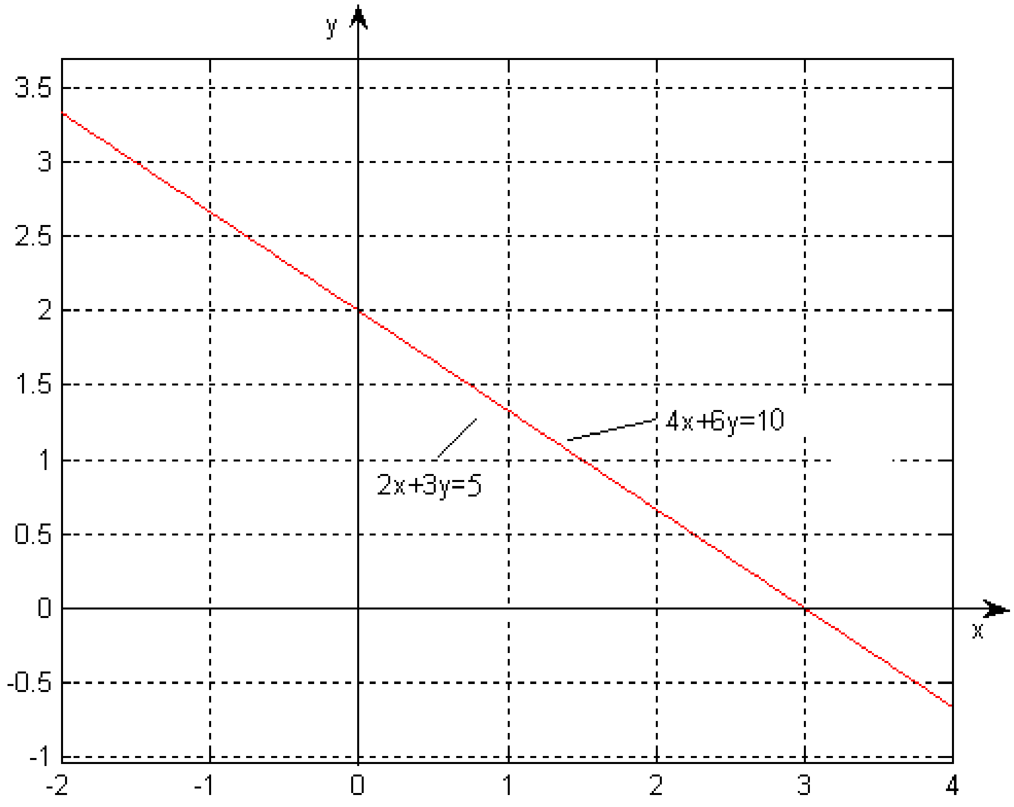
$$\begin{cases} 2x \ + \ 3y = 5 \\ 0 \ + \ \ 0 \ = 0 \end{cases} \Rightarrow \quad y = p \ (pasirenkame \ \ laisvai).$$

$$Tuomet \quad x = {}^{5}\!/_{2} - {}^{3}\!/_{2}\, p$$

Sprendinys yra (p, 5-3p), čia p – bet koks skaičius

Grafinė interpretacija

$$\begin{cases} 4x + 6y = 10; \\ 2x + 3y = 5 \end{cases}$$



be galo daug sprendinių

$$\begin{aligned} 5x+7y &= 12 \\ 7x+10y &= 17 \end{aligned};$$

Kramerio metodas:

$$\Delta = \begin{vmatrix} 5 & 7 \\ 7 & 10 \end{vmatrix} = 1;\ \Delta_x = \begin{vmatrix} 12 & 7 \\ 17 & 10 \end{vmatrix} = 1;\ \Delta_y = \begin{vmatrix} 5 & 12 \\ 7 & 17 \end{vmatrix} = 1;$$

$$x = \frac{\Delta_x}{\Delta} = 1; \quad y = \frac{\Delta_y}{\Delta} = 1.$$

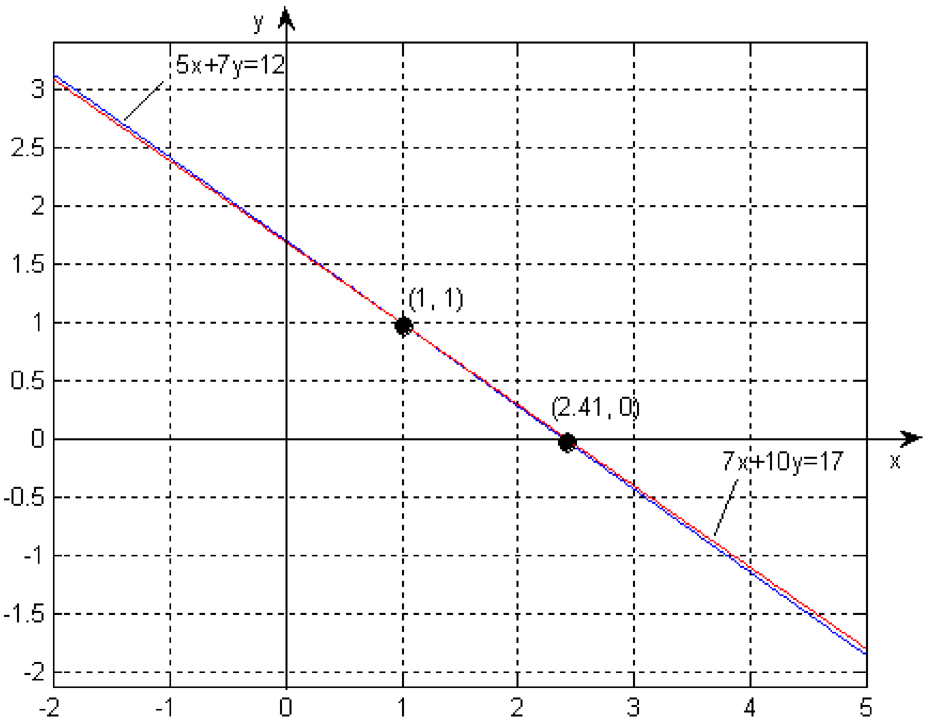
Kintamųjų eliminavimo metodas:

$$\begin{bmatrix} 5 & 7 & 12 \\ 7 & 10 & 17 \end{bmatrix} : \tfrac{5}{7} \Rightarrow \begin{bmatrix} 7 & \tfrac{49}{5} & \tfrac{84}{5} \\ 7 & 10 & 17 \end{bmatrix} \begin{matrix} \downarrow \\ \leftarrow - \end{matrix} \Rightarrow \begin{bmatrix} 7 & \tfrac{49}{5} & \tfrac{84}{5} \\ 0 & \tfrac{1}{5} & \tfrac{1}{5} \end{bmatrix}$$

$$\begin{cases} 7x + \tfrac{49y}{5} = \tfrac{84}{5} \\ \tfrac{y}{5} = \tfrac{1}{5} \end{cases}; \qquad y = 1;\ x = 1.$$

Grafinė interpretacija

$$\begin{aligned} 5x+7y &= 12; \\ 7x+10y &= 17 \end{aligned}$$



x=1; *y*=1 ⇨ $\begin{aligned} 5+7 &= 12 \\ 7+10 &= 17 \end{aligned}$

x=2,41; *y*=0 ⇨ $\begin{cases} 12,05+0 \cong 12 \\ 16,87+0 \cong 17 \end{cases}$

Lygčių sistema “silpnai apibrėžta”, ją apytiksliai tenkina ir kitos skaičių poros, gana tolimos nuo tikrojo sprendinio

Gauso algoritmas TLS sprendimui

Skaitiniai tiesinių algebrinių lygčių sistemų sprendimo metodai:

- *Tiesioginiai* – **sprendinys gaunamas algebriškai pertvarkant lygčių sistemą (t.y. koeficientų matrica skaičiuojant pertvarkoma)**

- *Iteraciniai* – **koeficientų matrica išlieka nepakitusi**

Tiesioginiai metodai, paremti kintamųjų eliminavimu : Gauso algoritmas (1)

Tiesioginis etapas:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 2 \\ x_1 - x_2 - x_3 + x_4 = 0 \\ 2x_1 + x_2 - x_3 + 2x_4 = 9 \\ 3x_1 + x_2 + 2x_3 - x_4 = 7 \end{cases}$$

Vedantieji elementai

$$\left[\begin{array}{cccc:c} 1^* & 1 & 1 & 1 & 2 \\ 1 & -1 & -1 & 1 & 0 \\ 2 & 1 & -1 & 2 & 9 \\ 3 & 1 & 2 & -1 & 7 \end{array}\right] \begin{array}{l} \downarrow \\ \leftarrow \times(-1) \\ \leftarrow \times(-2) \\ \leftarrow \times(-3) \end{array} \Rightarrow \left[\begin{array}{cccc:c} 1 & 1 & 1 & 1 & 2 \\ 0 & -2^* & -2 & 0 & -2 \\ 0 & -1 & -3 & 0 & 5 \\ 0 & -2 & -1 & -4 & 1 \end{array}\right] \begin{array}{l} \\ \\ \leftarrow \times\left(-\tfrac{1}{2}\right) \\ \leftarrow \times(-1) \end{array} \Rightarrow \left[\begin{array}{cccc:c} 1 & 1 & 1 & 1 & 2 \\ 0 & -2 & -2 & 0 & -2 \\ 0 & 0 & -2^* & 0 & 6 \\ 0 & 0 & 1 & -4 & 3 \end{array}\right] \begin{array}{l} \\ \\ \\ \leftarrow \times \tfrac{1}{2} \end{array} \Rightarrow$$

$$\Rightarrow \left[\begin{array}{cccc:c} 1 & 1 & 1 & 1 & 2 \\ 0 & -2 & -2 & 0 & -2 \\ 0 & 0 & -2 & 0 & 6 \\ 0 & 0 & 0 & -4 & 6 \end{array}\right]$$

Išplėstoji matrica

Gauso algoritmas (2)

Atvirkštinis etapas:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 2 \\ x_1 - x_2 - x_3 + x_4 = 0 \\ 2x_1 + x_2 - x_3 + 2x_4 = 9 \\ 3x_1 + x_2 + 2x_3 - x_4 = 7 \end{cases} \xrightarrow{\text{Atliktas tiesioginio etapo algoritmas}} \begin{cases} x_1 + x_2 + x_3 + x_4 = 2 \\ -2x_2 - 2x_3 = -2 \\ -2x_3 = 6 \\ -4x_4 = 6 \end{cases}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 2 \\ 0 & -2 & -2 & 0 & -2 \\ 0 & 0 & -2 & 0 & 6 \\ 0 & 0 & 0 & -4 & 6 \end{bmatrix} \begin{matrix} \to \\ \to \\ \to \\ \to \end{matrix} \begin{matrix} x_1 = (2 - x_2 - x_3 - x_4)/1 = 5/2 \\ x_2 = (-2 + 2x_3 - 0 * x_4)/(-2) = 4 \\ x_3 = (6 - 0 * x_4)/(-2) = -3 \\ x_4 = -3/2 \end{matrix}$$

Pvz_SMA_2_1_Gauso_algoritmas.m

Gauso algoritmas MATLAB

Pvz_SMA_2_1_Gauso_algoritmas.m

```
A=[1 1 1 1;
   1 -1 -1 1;
   2 1 -1 2;
   3 1 2 -1]
b=[2;0;9;7]
n=size(A,1)
A1=[A,b]

 %Tiesioginis etapas

for i=1:n-1
    for j=i+1:n
      A1(j,i:n+1)=A1(j,i:n+1)-A1(i,i:n+1)*A1(j,i)/A1(i,i);
    end
end

% Atgalinis etapas

x=zeros(n,1);     % reikia numatyti vieta x, kadangi skaiciuojame pradedami
                  % paskutiniu elementu
for i=n:-1:1
    x(i)=(A1(i,n+1) -A1(i,i+1:n)*x(i+1:n))/A1(i,i);
end
```

Gauso algoritmas (MATLAB kodavimo ypatybės)

```
A=[1 1 1 1;
   1 -1 -1 1;
   2 1 -1 2;
   3 1 2 -1]
b=[2;0;9;7]
n=size(A,1)
A1=[A,b]

 %Tiesioginis etapas

for i=1:n-1
    for j=i+1:n
      A1(j,i:n+1)=A1(j,i:n+1)-A1(i,i:n+1)*A1(j,i)/A1(i,i);
    end
end

% Atgalinis etapas

x=zeros(n,1);

for i=n:-1:1
    x(i)=(A1(i,n+1) -A1(i,i+1:n)*x(i+1:n))/A1(i,i);
end
```

Įrašę x(n+1) , gautume klaidos pranešimą "index exceeds matrix dimensions"

i=n

?

Aritmetikos veiksmo su [] rezultatas yra []: n+1:n=[], x([])=[], A1(i,i+1:n)*[]=[]…

Tačiau: A1(n,n+1:n)*x(n+1:n)=0

Gauso algoritmas Python

Pvz_SMA_2_01_Gauso_algoritmas.py

```
A=np.matrix([[1 , 1,  1,  1],
             [1, -1, -1,  1],
             [2,  1, -1,  2],
             [3,  1,  2, -1]]).astype(np.float)
b=np.matrix([[2],[0],[9],[7]]).astype(np.float)

n=(np.shape(A))[0]
nb=(np.shape(b))[1]
A1=np.hstack((A,b))


# tiesioginis etapas:

for i in range (0,n-1):
    for j in range (i+1,n):
        A1[j,i:n+nb]=A1[j,i:n+nb]-A1[i,i:n+nb]*A1[j,i]/A1[i,i];
        A1[j,i]=0;



#  atgalinis etapas:

x=np.zeros(shape=(n,nb))
for i in range (n-1,-1,-1):
    x[i,:]=(A1[i,n:n+nb]-A1[i,i-1:n-1]*x[i-1:n-1,:])/A1[i,i]
```

Gauso algoritmas (Python kodavimo ypatybės)

```
A=np.matrix([[1 , 1,  1,  1],
             [1, -1, -1,  1],
             [2,  1, -1,  2],
             [3,  1,  2, -1]]).astype(np.float)
b=np.matrix([[2],[0],[9],[7]]).astype(np.float)

n=(np.shape(A))[0]
A1=np.hstack((A,b))
```

Kadangi visi duomenys yra sveikieji skaičiai, pagal nutylėjimą būtų sukurta sveikaskaitinė matrica

```
# tiesioginis etapas:

for i in range (0,n-1):   # range pradeda 0 ir baigia n-2 (!)
    for j in range (i+1,n):  # range pradeda i+1 ir baigia n-1
        A1[j,i:n+1]=A1[j,i:n+1]-A1[i,i:n+1]*A1[j,i]/A1[i,i];
        A1[j,i]=0;
```

Nebylusis ciklas veikia analogiškai *range* (pradeda i ir baigia n)

```
#  atgalinis etapas:

x=np.zeros(shape=(n,1))
for i in range (n-1,-1,-1): # range pradeda n-1 ir baigia 0 (3-ias parametras yra zingsnis)
    x[i,:]=(A1[i,n:n+1]-A1[i,i-1:n-1]*x[i-1:n-1,:])/A1[i,i]
```

Gauso algoritmas (4)

Pvz_SMA_2_02_GA_su_vedancio_elemento_parinkimu.m

Vedančio elemento parinkimas:

- Jeigu vedantis elementas lygus 0, Gauso algoritmas neveiks;
- Lygtys sukeičiamos vietomis taip, kad vedančiu elementu taptų absoliutiniu dydžiu didžiausias koeficientas

$$\begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \Rightarrow \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \Rightarrow \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix}$$

Iš šių lygčių į vedančiosios poziciją perkeliama ta, kuri 2-ame stulpelyje turi didžiausią absoliutiniu dydžiu koeficientą

Kaip taikyti Gauso algoritmą, kai TLS matrica singuliari

Ką daryti, kai tam tikrame žingsnyje vedančio elemento(tarkime, k stulpelyje) parinkti negalime?:

$$\begin{array}{cccc|c} x & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{array}$$

Vedantis elementas =0

- **Tokia matrica yra singuliari (jos determinantas =0). Galėtume stabdyti programą ir išvesti klaidos pranešimą. Tai būtų paprasčiausia išeitis;**
- **Singuliari lygčių sistemos matrica gali reikšti, kad *sprendinių nėra*, arba kad *sprendinių yra be galo daug*;**
- ***Be galo daug* sprendinių – tai ne *bet koks* sprendinys. Parodysime, kaip galima apskaičiuoti tokius sprendinius**

$$\begin{array}{cccc|c} & k & & & \\ x & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{array}$$

- **Tiesioginį Gauso algoritmo etapą galime vykdyti toliau, (k stulpelio apatiniai elementai *jau* yra nuliniai)**

- **Nekeisdami k stulpelio, pereiname prie k+1 stulpelio ir parenkame vedantį elementą (k+1,k+1) pozicijoje**

1	1	1	1	2
1	1	-1	1	7
2	2	-2	2	14
-1	2	1	3	-7

2 lygtys vienodos. Tikėtina situacija 3 lygtys ir 4 nežinomieji, t.y. be galo daug sprendinių. Patikrinkime:

Tiesioginis Gauso algoritmo etapas

2	2	-2	2	14
0	3	-1	5	7
0	0	2	0	-5
0	0	0	0	0

$$0 \cdot x_4 = 0$$

Gavome tapatybę, todėl x4 gali būti bet koks skaičius.
Priimkime x4=1

Atgalinis Gauso algoritmo etapas

3.6667
-0.1667
-2.5000
1.0000

Pvz_SMA_2_3_GA_singular.m

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & -1 & 1 & 7 \\ 2 & 2 & -2 & 2 & 14 \\ -1 & 2 & 1 & 3 & -7 \end{array}\right)$$

Tiesioginis Gauso algoritmo etapas

$$\left(\begin{array}{cccc|c} 2 & 2 & -2 & 2 & 14 \\ 0 & 3 & -1 & 5 & 7 \\ 0 & 0 & 2 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \longrightarrow 0 \cdot x_4 = 0$$

Jeigu galime atlikti operacijas su simboliniais dydžiais, priimkime x4=p

Atgalinis Gauso algoritmo etapas

(14-2*p-5-2*(1.5-5/3*p))/2=**(2*p)/3 + 3**
(7-5*p-2.5)/3= **1.5-(5/3)*p**
-2.5
p

Pvz_SMA_2_4_GA_singular_symbolic.m

1	1	1	1	2
1	1	-1	1	7
2	2	-2	2	15
-2	1	1	3	-7

2 lygtys nesuderintos. Turėtų nebūti sprendinių. Patikrinkime:

Tiesioginis Gauso algoritmo etapas

2	2	-2	2	15
0	3	-1	5	8
0	0	2	0	-5.5
0	0	0	0	-0.5

$$\longrightarrow \quad 0 \cdot x_4 = 0.5$$

Lygybė negali būti tenkinama.
Stabdome programą su pranešimu
“Sprendinių nėra”

Pvz_SMA_2_3_GA_singular.m

Singuliari matrica: bendrasis atvejis, be galo daug sprendinių

1	1	1	1	2
1	1	-1	1	9.1429
1	1	-2	4	14
-1	-1	1	4	-7

Tiesioginis Gauso algoritmo etapas

1	1	1	1	2
0	$\boxed{0}$	-2	0	7.1429
0	0	-3	3	12
0	0	0	7	3

1	1	1	1	2
0	$\boxed{0}$	-2	0	7.1429
0	0	-3	3	12
0	0	0	7	3

$$x_1 = \frac{2 - x_2 - x_3 - x_4}{1} = 5.14286\text{-}p$$

$$0 \cdot x_2 - 2x_3 + 0 \cdot x_4 = 7.1429 \Rightarrow$$

$$\Rightarrow 0 \cdot x_2 = 2.85714\text{e-}006 \approx 0; \quad x_2 = p$$

$$x_3 = \frac{12 - 3x_4}{-3} = \text{-}3.57143$$

$$x_4 = {}^{3}\!/_{7} = 0.428571$$

Atgalinis Gauso algoritmo etapas

Laikome, kad patenkinamu tikslumu gauta *tapatybė*, t.y. x2 gali būti bet koks skaičius

Singuliari matrica: bendrasis atvejis, sprendinių nėra

1	1	1	1	2
1	1	-1	1	7
1	1	-2	4	14
-1	-1	1	4	-7

Tiesioginis Gauso algoritmo etapas

1	1	1	1	2
0	$\boxed{0}$	-2	0	5
0	0	-3	3	12
0	0	0	7	3

1	1	1	1	2
0	$\boxed{0}$	-2	0	5
0	0	-3	3	12
0	0	0	7	3

$$0 \cdot x_2 - 2x_3 + 0 \cdot x_4 = 5 \Rightarrow$$

$$\Rightarrow 0 \cdot x_2 = -2.14286;$$

$$x_3 = \frac{12 - 3x_4}{-3} = -3.57143$$

$$x_4 = 3/7 = 0.428571$$

Atgalinis Gauso algoritmo etapas

Ši lygybė negali būti tenkinama, todėl sprendinių nėra

Pvz_SMA_2_4_GA_singular_symbolic.m

Singuliarios koeficientų matricos: apibendrinimas

- **Parodėme, kad vykdant Gauso algoritmą, galima gauti sprendinį arba išvadą apie sprendinio nebuvimą, kai koeficientų matrica yra singuliari;**
- **Kai sprendinys egzistuoja, nuliniam vedančiajam elementui atitinkiantis kintamasis gali būti bet koks skaičius;**
- **"Bet kokios" kintamųjų reikšmės bendruoju atveju vaizduojamos simboliais, pvz. *pi, pj,* … . Kiti kintamieji išreiškiami skaičiais ir šiais simboliais;**
- **Jeigu pakanka rasti vieną sprendinį iš daugelio, "bet kokias" reikšmes galintiems priimti kintamiesiems skaitines reikšmes parenkame laisvai, pvz. =1**

Sprendinio tikslumo patikrinimas, panaudojant vektorių normas

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & -1 & 1 & 9.1429 \\ 1 & 1 & -2 & 4 & 14 \\ -1 & -1 & 1 & 4 & -7 \end{array}\right] \xrightarrow{\text{Taikome Gauso algoritmą}} \{\mathbf{x}\} = \begin{Bmatrix} 5.14286-p \\ p \\ -3.57143 \\ 0.428571 \end{Bmatrix}$$

$$p = 1$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & -2 & 4 \\ -1 & -1 & 1 & 4 \end{bmatrix} \begin{Bmatrix} 5.14286-1 \\ 1 \\ -3.57143 \\ 0.428571 \end{Bmatrix} - \begin{Bmatrix} 2 \\ 9.1429 \\ 14 \\ -7 \end{Bmatrix} = 1.0e\text{-}005 * \begin{Bmatrix} 0 \\ -0.2857 \\ 0 \\ 0 \end{Bmatrix}$$

MATLAB: bendra_santykine_paklaida= norm(liekana)/norm(x)

Python : **bendra_santykine_paklaida= numpy.linalg.norm(liekana)/ numpy.linalg.norm(x)**

bendra_santykine_paklaida=5.1232e-007

$$norm(x) = \sqrt{\sum_{i=1}^{n} x_i^{\ 2}}$$

$$
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & -2 & 4 \\ -1 & -1 & 1 & 4 \end{bmatrix}
\begin{Bmatrix} 5.14286\text{-}1 \\ 1 \\ -3.57143 \\ 0.428571 \end{Bmatrix}
-
\begin{Bmatrix} 2 \\ 9.1429 \\ 14 \\ -7 \end{Bmatrix}
= \quad 1.0\text{e-}005 * \begin{Bmatrix} 0 \\ -0.2857 \\ 0 \\ 0 \end{Bmatrix}
$$

Vektoriaus norma vienu teigiamu skaičiumi apibūdina vektoriaus dydį (ilgį). Dažniausiai taikoma Euklido norma, kurios formulė apibendrina geometrinio vektoriaus ilgio formulę

$$\vec{v} = (v_1, v_2, \ldots, v_n)$$

$$norm(\vec{v}) = \|\vec{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

MATLAB: bendra_santykine_paklaida= norm(liekana)/norm(x)

Python : bendra_santykine_paklaida= numpy.linalg.norm(liekana)/ numpy.linalg.norm(x)

Gauso algoritmo sudėtingumas

\+ n-i+2 kartų * * 1 kartą

```
%Tiesioginis etapas
for i=1:n-1
    for j=i+1:n
      A1(j,i:n+1)=A1(j,i:n+1)-A1(i,i:n+1)*A1(j,i)/A1(i,i);
    end
end
```

$$\sum_{i=1}^{n-1}(2(n-i+2)+1)(n-i)=\sum_{i=1}^{n-1}\left(2n^2+5n-\left(4n+5\right)i+2i^2\right)=$$

$$=2n^2(n-1)+5n(n-1)-(4n+5)\frac{1+n-1}{2}(n-1)+\frac{(n-1)(2n-1)(n)}{3}=$$

$$=\frac{4n^3+9n^2-13n}{6}$$

Tiesioginio etapo veiksmų skaičius

$$\sum_{i=1}^{n} i=\frac{1+n}{2}n$$

$$\sum_{i=1}^{n} i^2=\frac{n(2n+1)(n+1)}{6}$$

```
% Atgalinis etapas
x=zeros(n,1);
for i=n:-1:1
    x(i)=(A1(i,n+1) -A1(i,i+1:n)*x(i+1:n))/A1(i,i);
end
```

1 kartą n-1 kartų 1 kartą

\+ * *

$$\sum_{i=1}^{n}(n-i+2)=n^2-\frac{1+n}{2}n+2n=\frac{2n^2-n-n^2+4n}{2}=\frac{n^2+3n}{2}$$

Atgalinio etapo veiksmų skaičius

Jei n — didelis skaičius, Gauso metodo skaičiavimo apimtis $O\left(\frac{2}{3}n^3\right)$

- **Galima apskaičiuoti sudėties ir daugybos veiksmų skaičių atskirai**

sudėties veiksmų skaičius $s = \dfrac{n(n-1)(2n+5)}{6}$

daugybos veiksmų skaičius $d = \dfrac{n(n^2+3n-1)}{3}$

jei *n* — didelis skaičius $s \approx d \approx \dfrac{n^3}{3}$

- **Apytikslų sudėtingumo įvertį galime nustatyti vien pagal didžiausią įdėtinių ciklų skaičių;**
- **Kadangi kiekvieno ciklo didžiausias pakartojimų skaičius yra n-1 arba n-2, o įdėtinių ciklų skaičius 3, apytikslus sudėtingumo įvertis yra** $O(n^3)$

```
%Tiesioginis etapas
for i=1:n-1          % pagrindinis ciklas
    for j=i+1:n      % idetinis ciklas
      A1(j,i:n+1)=A1(j,i:n+1)-A1(i,i:n+1)*A1(j,i)/A1(i,i);
                                   %„nebylusis" idetinis ciklas
    end
end
```

- **Analogiškai, Gauso algoritmo atgalinio etapo sudėtingumas yra** $O(n^2)$

```
% Atgalinis etapas
x=zeros(n,1);
for i=n:-1:1 ; x(i)=(A1(i,n+1) -A1(i,i+1:n)*x(i+1:n))/A1(i,i); end
```

Kiti kintamųjų eliminavimu paremti algoritmai

Kiti kintamųjų eliminavimu paremti algoritmai 1

Gauso algoritmas

Įstrižainės elementų sandauga lygi matricos determinantui

$$
\begin{array}{ccccccc}
\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} &
\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & = &
\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \rightarrow &
\begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix}
\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \rightarrow
\begin{bmatrix} x_{1,1} & x_{2,1} & \cdots x_{p,1} \\ x_{2,1} & x_{2,2} & \cdots x_{2,p} \\ \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots x_{n,p} \end{bmatrix} \\
\mathbf{A} & \mathbf{X} & = & \mathbf{B} & & \mathbf{A} \quad \mathbf{B} &
\end{array}
$$

Gauso-Žordano algoritmas (netinka, jeigu matrica singuliari)

$$
\begin{array}{cccccc}
\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} &
\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & = &
\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \rightarrow &
\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & 0 & x \end{bmatrix}
\begin{bmatrix} x_{1,1} & x_{2,1} & \cdots x_{p,1} \\ x_{2,1} & x_{2,2} & \cdots x_{2,p} \\ \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots x_{n,p} \end{bmatrix} \\
\mathbf{A} & \mathbf{X} & = & \mathbf{B} & & \mathbf{A} \quad \mathbf{X}
\end{array}
$$

Atgalinis žingsnis nereikalingas

Kiti kintamųjų eliminavimu paremti algoritmai 2

Atvirkštinės matricos algoritmas

$$\mathbf{A\,x = b}\ ; \qquad \mathbf{A^{-1}A\,x = A^{-1}b}; \qquad \mathbf{x = A^{-1}b};$$

$$\underset{\mathbf{A}}{\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}} \underset{\mathbf{X}}{\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}} = \underset{\mathbf{E}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \xRightarrow{\text{Taikome Gauso-Žordano algoritmą}} \mathbf{A}^{-1} = \mathbf{X}$$

- **Vieną kartą apskaičiavę atvirkštinę matricą, galime rasti sprendinį esant bet kokiam dešinės pusės vektoriui;**
- **Tam pakanka padauginti atvirkštinę matricą iš laisvųjų narių vektoriaus**

LU skaidos algoritmas

Tarkime, reikia išspręsti TLS su ta pačia koeficientų matrica, tačiau daugeliu skirtingų dešinės pusės vektorių

- **Jeigu visi laisvųjų narių vektoriai žinomi iš anksto, galima juos visus įrašyti į išplėstąją matricą, tiesiogiai taikyti Gauso algoritmą ir taip iškart gauti visiems laisvųjų narių vektoriams atitinkančius sprendinius;**
- **Norint išspręsti lygčių sistemą su iš anksto nežinomomis laisvųjų narių reikšmėmis, reiktų <u>iš naujo įvykdyti visą Gauso algoritmą</u>;**
- **Iš kintamųjų eliminavimu pagrįstų algoritmų šio trūkumo neturi tik *atvirkštinės matricos algoritmas*, tačiau jis labai imlus skaičiavimams;**
- **šią problemą sprendžia *skaidos metodai* , kai matrica išskaidoma trikampiais daugikliais**

LU algoritmo esmė

Bet kuri ***kvadratinė*** ir ***nesinguliari*** lygčių sistemos matrica *A* gali būti išskaidyta į dviejų ***trikampių matricų L ir U*** sandaugą:

$$[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\} \Rightarrow [\mathbf{L}]\,[\mathbf{U}]\,\mathbf{x} = \mathbf{b}$$

$$[\mathbf{L}] = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \qquad \{\mathbf{U}\} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

Išskaidyti galima, jeigu $\Delta_1 = |a_{11}| \neq 0 \quad \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0 \quad \cdots \quad \Delta_n = \det(A) \neq 0$

Skaidinys nėra vienintelis, todėl galima priimti $l_{11}=\ldots=l_{nn}=1$

Taikant *LU algoritmą, lygčių sistema sprendžiama taip:*

$$[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}$$

Apskaičiuojami trikampiai daugikliai

$$[\mathbf{L}]\,[\mathbf{U}]\,\mathbf{x} = \mathbf{b} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ \otimes & 1 & 0 & 0 \\ \otimes & \otimes & 1 & 0 \\ \otimes & \otimes & \otimes & 1 \end{bmatrix} \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ 0 & \otimes & \otimes & \otimes \\ 0 & 0 & \otimes & \otimes \\ 0 & 0 & 0 & \otimes \end{bmatrix} \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix} = \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix}$$

Analogiškai, kaip ir Gauso algoritmo atgaliniame etape, kurio sudėtingumas tėra O(n^2)

Atliekamas pakeitimas $[\mathbf{U}]\,\{\mathbf{x}\} = \{\mathbf{y}\}$

$$[\mathbf{L}]\,\{\mathbf{y}\} = \{\mathbf{b}\} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ \otimes & 1 & 0 & 0 \\ \otimes & \otimes & 1 & 0 \\ \otimes & \otimes & \otimes & 1 \end{bmatrix} \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix} = \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix}$$

Pagal trikampę koeficientų matricą apskaičiuojamas **{y}**

$$[\mathbf{U}]\,\{\mathbf{x}\} = \{\mathbf{y}\} \qquad \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ 0 & \otimes & \otimes & \otimes \\ 0 & 0 & \otimes & \otimes \\ 0 & 0 & 0 & \otimes \end{bmatrix} \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix} = \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix}$$

Pagal trikampę koeficientų matricą apskaičiuojamas **{x}**

$$\{\mathbf{x}\}$$

LU skaidos apskaičiavimas (1)

U apskaičiuojamas atliekant Gauso algoritmo tiesioginį etapą:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 2 \\ x_1 - x_2 - x_3 + x_4 = 0 \\ 2x_1 + x_2 - x_3 + 2x_4 = 9 \\ 3x_1 + x_2 + 2x_3 - x_4 = 7 \end{cases}$$

$$\begin{bmatrix} 1^* & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & 2 \\ 3 & 1 & 2 & -1 \end{bmatrix} \begin{matrix} \downarrow \\ \leftarrow\times(-1) \\ \leftarrow\times(-2) \\ \leftarrow\times(-3) \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2^* & -2 & 0 \\ 0 & -1 & -3 & 0 \\ 0 & -2 & -1 & -4 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow\times\left(-\tfrac{1}{2}\right) \\ \leftarrow\times(-1) \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2^* & 0 \\ 0 & 0 & 1 & -4 \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow\times\tfrac{1}{2} \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 1^* & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & 2 \\ 3 & 1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2^* & -2 & 0 \\ 0 & -1 & -3 & 0 \\ 0 & -2 & -1 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2^* & -2 & 0 \\ 0 & -1 & -3 & 0 \\ 0 & -2 & -1 & -4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \tfrac{1}{2} & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2^* & 0 \\ 0 & 0 & 1 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2^* & 0 \\ 0 & 0 & 1 & -4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\tfrac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

LU skaidos apskaičiavimas (2)

$$\begin{bmatrix} 1^* & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & 2 \\ 3 & 1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1/2 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 1^* & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & 2 \\ 3 & 1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 1/2 & 1 & 0 \\ 3 & 1 & -1/2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

$$[A] = [L] \cdot [U]$$

Pradinė koeficientų matrica

Koeficientai (su minuso ženklu), kurie buvo panaudoti atliekant Gauso algoritmo tiesioginį etapą

Gauso algoritmo tiesioginio etapo rezultatas

LU skaidos apskaičiavimas (3)

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 2 \\ x_1 - x_2 - x_3 + x_4 = 0 \\ 2x_1 + x_2 - x_3 + 2x_4 = 9 \\ 3x_1 + x_2 + 2x_3 - x_4 = 7 \end{cases}$$

$$[\mathbf{A}]=\begin{bmatrix}1^* & 1 & 1 & 1\\ 1 & -1 & -1 & 1\\ 2 & 1 & -1 & 2\\ 3 & 1 & 2 & -1\end{bmatrix}\begin{matrix}\downarrow\\ \leftarrow\times(-1)\\ \leftarrow\times(-2)\\ \leftarrow\times(-3)\end{matrix} \Rightarrow \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & -2^* & -2 & 0\\ 0 & -1 & -3 & 0\\ 0 & -2 & -1 & -4\end{bmatrix}\begin{matrix}\\ \\ \leftarrow\times\left(-\tfrac{1}{2}\right)\\ \leftarrow\times(-1)\end{matrix} \Rightarrow \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & -2 & -2 & 0\\ 0 & 0 & -2^* & 0\\ 0 & 0 & 1 & -4\end{bmatrix}\begin{matrix}\\ \\ \\ \leftarrow\times\tfrac{1}{2}\end{matrix} \Rightarrow \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & -2 & -2 & 0\\ 0 & 0 & -2 & 0\\ 0 & 0 & 0 & -4\end{bmatrix} = [\mathbf{U}]$$

$$[\mathbf{L}]=\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 1 & 0 & 0\\ 2 & \tfrac{1}{2} & 1 & 0\\ 3 & 1 & -\tfrac{1}{2} & 1\end{bmatrix}$$

Įstrižainėje priimamos vienetinės reikšmės

LU skaidos apskaičiavimas (4)

Taupome kompiuterio atmintį:

$$
\begin{bmatrix} 1^* & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & 2 \\ 3 & 1 & 2 & -1 \end{bmatrix}
\begin{array}{l} \downarrow \\ \leftarrow\times(-1) \\ \leftarrow\times(-2) \\ \leftarrow\times(-3) \end{array}
\Rightarrow
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2^* & -2 & 0 \\ 2 & -1 & -3 & 0 \\ 3 & -2 & -1 & -4 \end{bmatrix}
\begin{array}{l} \\ \\ \leftarrow\times\left(-\tfrac{1}{2}\right) \\ \leftarrow\times(-1) \end{array}
\Rightarrow
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & 0 \\ 2 & \tfrac{1}{2} & -2^* & 0 \\ 3 & 1 & 1 & -4 \end{bmatrix}
\begin{array}{l} \\ \\ \\ \leftarrow\times\tfrac{1}{2} \end{array}
\Rightarrow
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & 0 \\ 2 & \tfrac{1}{2} & -2 & 0 \\ 3 & 1 & -\tfrac{1}{2} & -4 \end{bmatrix}
$$

Atlikus algoritmą, daugikliai [L] ir [U] užima pradinės matricos [A] vietą:

$$
[\mathbf{U}] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}
$$

$$
[\mathbf{L}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & \tfrac{1}{2} & 1 & 0 \\ 3 & 1 & -\tfrac{1}{2} & 1 \end{bmatrix}
$$

$$
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & 0 \\ 2 & \tfrac{1}{2} & -2 & 0 \\ 3 & 1 & -\tfrac{1}{2} & -4 \end{bmatrix} \Rightarrow [\mathbf{A}]
$$

Įstrižainės vienetus saugoti netikslinga. Ir taip žinome, kad [L] pagrindinėje įstrižainėje turi būti vienetai

LU skaidos algoritmas

Pvz_SMA_2_05, 2_06

Jeigu vedantysis elementas `A(i,i)=0`, programa sustos

```
A=[1  1  1  1;
   1 -1 -1  1;
   2  1 -1  2;
   3  1  2 -1]
b=[2;0;9;7]
n=size(A,1)

% LU skaida
for i=1:n-1
    for j=i+1:n
        r=A(j,i)/A(i,i);
        A(j,i+1:n)=A(j,i+1:n)-A(i,i+1:n)*r;
        A(j,i)=r;
    end
end
% 1-as atgalinis etapas, sprendziama  Ly=b, y->b
for i=2:n
    b(i,:)=b(i,:)-A(i,1:i-1)*b(1:i-1);
end
% 2-as atgalinis etapas , sprendziama Ux=b, x->b
for i=n:-1:1
    b(i)=(b(i)-A(i,i+1:n)*b(i+1:n))/A(i,i);
end
```

LU skaidos algoritmas su vedančiojo elemento parinkimu

Parenkant vedantįjį elementą, kai kurios lygtys sukeičiamos vietomis. Vėliau, sprendžiant lygčių sistemą, laisvųjų narių vektorius turės būti pateiktas tokia tvarka, kaip sutvarkytos lygtys po LU skaidos. Todėl reikia prisiminti, kaip pakito lygčių tvarka

```
A=[1  1  1  1;
   1 -1 -1  1;
   2  1 -1  2;
   3  1  2 -1]
b=[2;0;9;7]
n=size(A,1)
P=[1:n]
% LU skaida
for i=1:n-1
    [a,iii]=max(abs(A(i:n,i)));
    A([i,i+iii-1],:)=A([i+iii-1,i],:);
    P([i,i+iii-1])=P([i+iii-1,i]);
    for j=i+1:n
        r=A(j,i)/A(i,i);
        A(j,i+1:n)=A(j,i+1:n)-A(i,i+1:n)*r;
        A(j,i)=r;
    end
end

b=b(P)
% 1-as atgalinis etapas, sprendziama  Ly=b, y->b
for i=2:n,  b(i,:)=b(i,:)-A(i,1:i-1)*b(1:i-1); end
% 2-as atgalinis etapas , sprendziama Ux=b, x->b
for i=n:-1:1, b(i)=(b(i)-A(i,i+1:n)*b(i+1:n))/A(i,i); end
```

LU skaidos algoritmas su vedančiojo elemento parinkimu (Python)

```
A=np.matrix([[1 , 1,  1,  1],
             [1, -1, -1,  1],
             [2,  1, -1,  2],
             [3,  1,  2, -1]]).astype(np.float)

b=np.matrix([[2],[0],[9],[7]]).astype(np.float)
n=(np.shape(A))[0]
P=np.arange(0,n)

# tiesioginis etapas:
for i in range (0,n-1):
a=max(abs(A[i:n,i]));  iii=abs(A[i:n,i]).argmax()
    A[[i,i+iii],:]=A[[i+iii,i],:]  # sukeiciamos eilutes
    P[[i,i+iii]]=P[[i+iii,i]]      # sukeiciami eiluciu numeriai
    for j in range (i+1,n):
        r=A[j,i]/A[i,i]
        A[j,i:n+1]=A[j,i:n+1]-A[i,i:n+1]*r;
        A[j,i]=r;

b=b[P]
# 1-as atgalinis etapas, sprendziama  Ly=b, y->b
for i in range(1,n) :
    b[i]=b[i]-A[i,0:i]*b[0:i]
# 2-as atgalinis etapas , sprendziama Ux=b, x->b
for i in range (n-1,-1,-1) :
    b[i]=(b[i]-A[i,i+1:n]*b[i+1:n])/A[i,i]
```

Parenkant vedantįjį elementą, kai kurios lygtys sukeičiamos vietomis. Vėliau, sprendžiant lygčių sistemą, laisvųjų narių vektorius turės būti pateiktas tokia tvarka, kaip sutvarkytos lygtys po LU skaidos. Todėl reikia prisiminti, kaip pakito lygčių tvarka

SMA_02_Klausimai savikontrolei(1):

1. Paaiškinkite, kokia yra grafinė interpretacija atvejų, kai dviejų tiesinių lygčių sistema a)turi vienintelį sprendinį, b)neturi sprendinių, c)turi be galo daug sprendinių, d)silpnai apibrėžta;
2. Kokie yra Gauso algoritmo vykdymo etapai, apibūdinkite kiekvieną iš jų;
3. Kas yra išplėstoji lygčių sistemos matrica;
4. Kas yra vedantysis elementas;
5. Ką daryti, jeigu vedantysis elementas lygus nuliui. Ką pagal šią reikšmę galima pasakyti apie lygčių sistemos sprendinį;
6. Kaip vykdyti tiesioginį Gauso algoritmo etapą, kai nepavyksta parinkti nelygaus nuliui vedančiojo elemento;
7. Kaip vykdomas atgalinis Gauso algoritmo etapas, kai sutinkamas lygus nuliui įstrižainės elementas;
8. Kaip Gauso algoritmo atgalinio etapo metu atpažinti atvejus "be galo daug sprendinių" ir "sprendinių nėra";
9. Kaip skaičiuojama atveju "be galo daug sprendinių" ;

SMA_02_Klausimai savikontrolei(2):

10. Kaip patikrinti lygčių sistemos sprendinio tikslumą. Kas yra sprendinio ir liekanos Euklido normos;

11. Kiek apytiksliai aritmetinių operacijų reikia atlikti, taikant Gauso algoritmą;

12. Kaip Gauso algoritmas taikomas lygčių sistemos su daugeliu dešiniosios pusės vektorių sprendimui;

13. Apibūdinkite Gauso-Žordano algoritmą;

14. Kaip apskaičiuoti atvirkštinę matricą, taikant Gauso metodą

15. Kas yra LU skaida ir kaip ji pritaikoma lygčių sistemai spręsti. Kokią sąlygą turi tenkinti koeficientų matrica, kad jai būtų galima pritaikyti LU skaidą;

16. Paaiškinkite, kaip gaunami LU skaidos daugikliai L ir U. Kodėl, vykdant skaidą, reikia papildomai sugeneruoti perstatymų matricą P