

Tiesinių lygčių sistemų sprendimas:

Atspindžio ir QR skaidos algoritmai

Temoje aiškinama:

- Motyvuojamas TLS sprendimo algoritmų patobulinimas, kai tiesioginio etapo pertvarkymuose panaudojamos visos lygtys
- Kaip apskaičiuoti 3D geometrinio vektoriaus atspindžio plokštumos atžvilgiu koordinatės ir kaip parinkti atspindžio plokštumą, kad gauti tam tikras pasirinktas atspindžio vektoriaus koordinatės;
- Atspindžio pakeitimo išplėtimas į n -matę erdvę;
- Atspindžio pakeitimo panaudojimas QR algoritmui sukurti;
- QR skaidos algoritmas;
- Kokios yra algebrinės lygties sprendimui taikomos programinės funkcijos MATLAB ir Python terpėse

Prisiminkime: kintamųjų eliminavimu paremti algoritmai

Gauso algoritmas

$$\begin{array}{|c|c|} \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline x & x & x & x \\ \hline 0 & x & x & x \\ \hline 0 & 0 & x & x \\ \hline 0 & 0 & 0 & x \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|} \hline x_{1,1} & x_{2,1} & \cdots x_{p,1} \\ \hline x_{2,1} & x_{2,2} & \cdots x_{2,p} \\ \hline \vdots & \vdots & \vdots \\ \hline x_{n,1} & x_{n,2} & \cdots x_{n,p} \\ \hline \end{array}$$

A **X** = **B** **A** **B**

Gauso-Žordano algoritmas

$$\begin{array}{|c|c|} \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline x & x & x \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline x & 0 & 0 & 0 \\ \hline 0 & x & 0 & 0 \\ \hline 0 & 0 & x & 0 \\ \hline 0 & 0 & 0 & x \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline x_{1,1} & x_{2,1} & \cdots x_{p,1} \\ \hline x_{2,1} & x_{2,2} & \cdots x_{2,p} \\ \hline \vdots & \vdots & \vdots \\ \hline x_{n,1} & x_{n,2} & \cdots x_{n,p} \\ \hline \end{array}$$

A **X** = **B** **A** **X**

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

Atvirkštinės matricos algoritmas

$$\begin{array}{|c|c|} \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|} \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline x & x & x & x \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}$$

A **X** = **E**

Taikome Gauso-Žordano algoritmą



$$\mathbf{A}^{-1} = \mathbf{X}$$

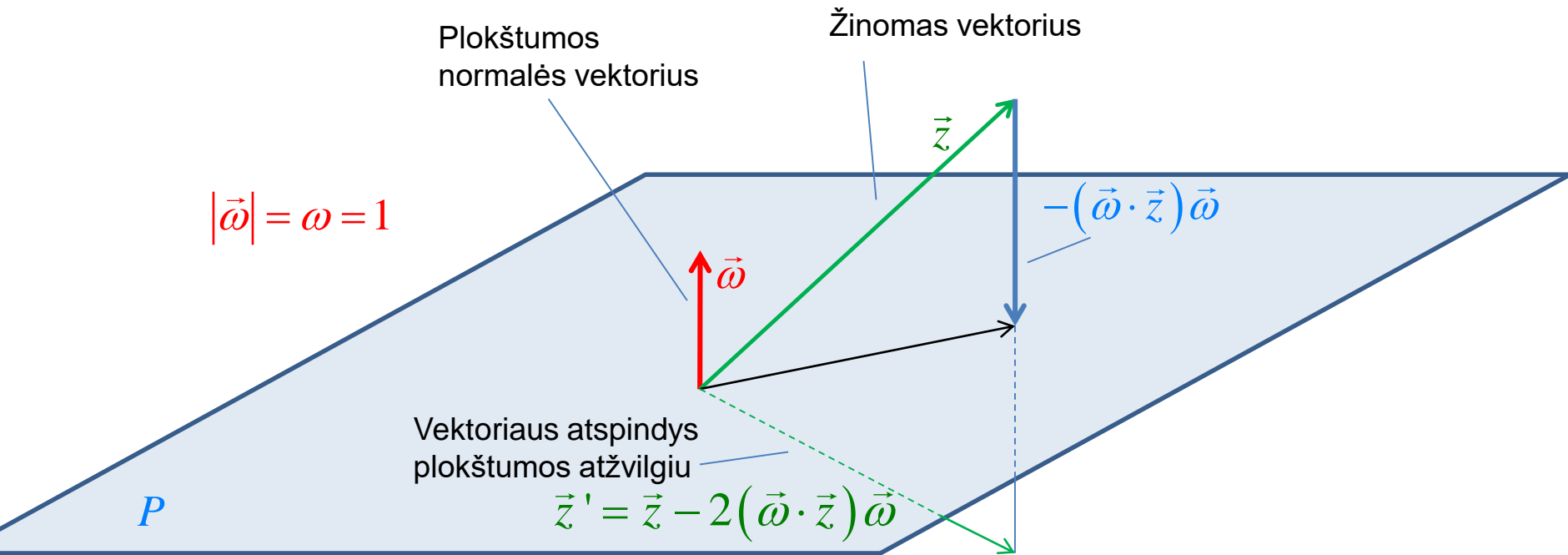
LU skaidos algoritmas

$$\mathbf{A} \mathbf{x} = \mathbf{b} \Rightarrow \mathbf{L} \mathbf{U} \mathbf{x} = \mathbf{b} \Rightarrow \mathbf{L} \mathbf{y} = \mathbf{b} \Rightarrow \mathbf{y} \Rightarrow \mathbf{U} \mathbf{x} = \mathbf{y} \Rightarrow \mathbf{x}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \otimes & 1 & 0 & 0 \\ \otimes & \otimes & 1 & 0 \\ \otimes & \otimes & \otimes & 1 \end{bmatrix} \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ 0 & \otimes & \otimes & \otimes \\ 0 & 0 & \otimes & \otimes \\ 0 & 0 & 0 & \otimes \end{bmatrix} \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix} = \begin{Bmatrix} \times \\ \times \\ \times \\ \times \end{Bmatrix}$$

- Jeigu viename iš Gauso algoritmo vykdymo žingsnių vedančio elemento skaitinė reikšmė maža, galimos ženklios apvalinimo paklaidos;
- Apvalinimo paklaidos mažesnės, taikant *atspindžio algoritmą (QR algoritmas, Hausholderio algoritmas)*
- Atspindžio algoritmo idėją galima paaiškinti pagal du vektoriais aprašomus geometrinius veiksmus:
 - ✓ Kaip apskaičiuoti 3D geometrinio vektoriaus atspindžio duotos plokštumos atžvilgiu koordinates;
 - ✓ Kaip parinkti atspindžio plokštumą, kad gauti tam tikras pasirinktas atspindžio vektoriaus koordinates

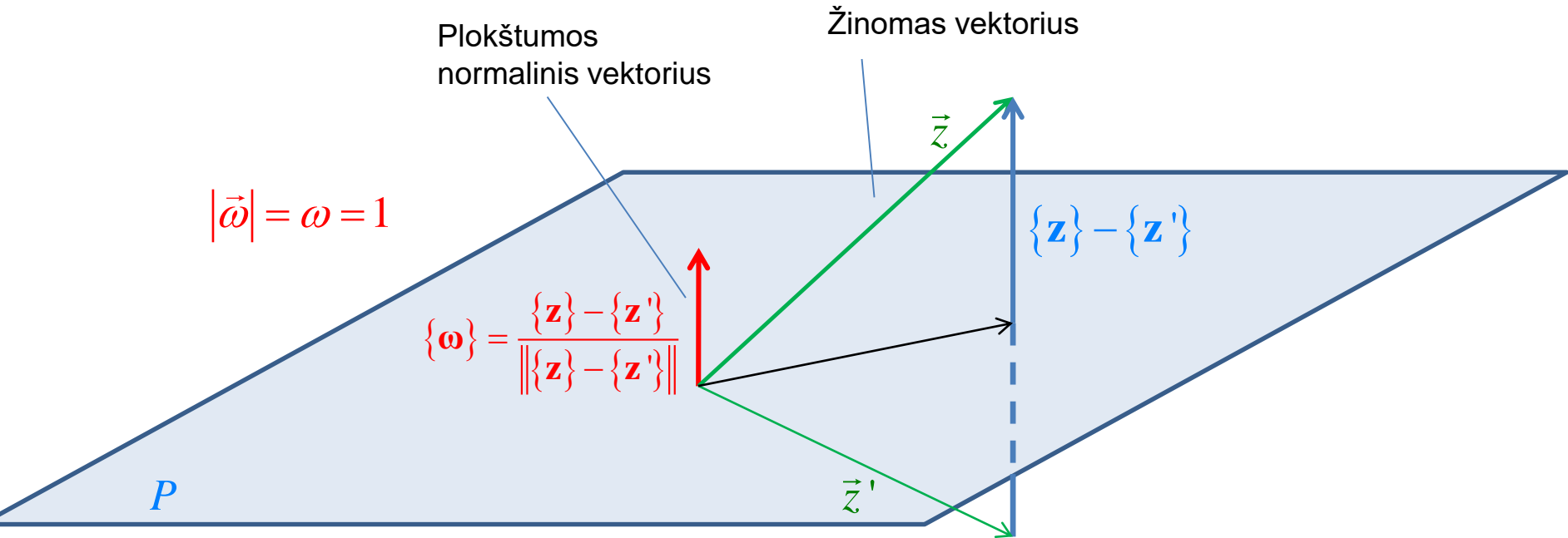
Atspindžio algoritmo idėja: vektoriaus atspindys plokštumos atžvilgiu 3D erdvėje



$$\vec{z}' = \vec{z} - 2\vec{\omega}(\vec{\omega} \cdot \vec{z}) = \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} - 2 \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \left(\begin{Bmatrix} \omega_x & \omega_y & \omega_z \end{Bmatrix} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} \right) =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} - 2 \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \left\{ \omega_x \quad \omega_y \quad \omega_z \right\} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} = [\mathbf{E}]\{\mathbf{z}\} - 2\{\boldsymbol{\omega}\}\{\boldsymbol{\omega}\}^T\{\mathbf{z}\} = \left([\mathbf{E}] - 2\{\boldsymbol{\omega}\}\{\boldsymbol{\omega}\}^T \right) \{\mathbf{z}\}$$

Atspindžio algoritmo idėja: veidrodis, atspindintis duotą vektorių į duotą atspindį



Jeigu žinome $\{z\}$ ir jo atspindį $\{z'\}$, plokštumos normalę galime apskaičiuoti taip:

$$\{\omega\} = \frac{\{z\} - \{z'\}}{\|\{z\} - \{z'\}\|}$$

$$\|\{v\}\| = \sqrt{\sum_{i=1}^n v_i^2}$$

- vektoriaus Euklido norma ("ilgis"), n- erdvės matavimų skaičius

Atspindžio algoritmo idėja: apibendrinimas

1. Pagal formulę $\{\mathbf{z}'\} = [\mathbf{Q}]\{\mathbf{z}\}$, $[\mathbf{Q}] = [\mathbf{E}] - 2\{\boldsymbol{\omega}\}\{\boldsymbol{\omega}\}^T$

vektorių $\{\mathbf{z}\}$ pertvarkomas į jo atspindį atžvilgiu plokštumos, kurios normalės vektorius yra $\{\boldsymbol{\omega}\}$

2. Pagal formulę $\{\boldsymbol{\omega}\} = \frac{\{\mathbf{z}\} - \{\mathbf{z}'\}}{\|\{\mathbf{z}\} - \{\mathbf{z}'\}\|}$ nustatoma normalė veidrodžio

plokštumos, kuris atspindi duotą vektorių į duotą jo atspindį. Duotųjų atspindimo $\{\mathbf{z}\}$ vektoriaus ir atspindžio vektorių $\{\mathbf{z}'\}$ Euklido normos turi būti lygios, t.y. $\|\{\mathbf{z}\}\| = \|\{\mathbf{z}'\}\|$

Atspindžio pakeitimo išplėtimas į n -matę erdvę

Uždavinys: koks turėtų būti $\{\omega\}$, kad atspindėto vektoriaus $\{z'\}$ visos koordinatės būtų =0, išskyrus pirmąją?

$$\{z'\} = \text{sign}(z_1) \|\{z\}\| \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad - \text{Atspindžio norma turi būti lygi vektoriaus } \{z\} \text{ normai}$$

$$\{\omega\} = \frac{\{z\} - \{z'\}}{\|\{z\} - \{z'\}\|}$$

$$[\mathbf{Q}] = [\mathbf{E}] - 2\{\omega\}\{\omega\}^T$$



Taikydami atspindžio matricą, galime apskaičiuoti bet kurio vektoriaus atspindį atžvilgiu plokštumos, kurios normalė yra $\{\omega\}$

3D atveju suformuluotos ir geometriškai paaiškintos priklausomybės pritaikomos **n-matėje tiesinėje erdvėje**:

$$\{\mathbf{z}'\} = \text{sign}(z_1) \|\{\mathbf{z}\}\| \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \{\boldsymbol{\omega}\} = \frac{\{\mathbf{z}\} - \{\mathbf{z}'\}}{\|\{\mathbf{z}\} - \{\mathbf{z}'\}\|} \quad [\mathbf{Q}] = [\mathbf{E}] - 2\{\boldsymbol{\omega}\}\{\boldsymbol{\omega}\}^T$$


- Taikant formulę $\{\mathbf{z}'\} = [\mathbf{Q}]\{\mathbf{z}\}$ bet koks n-matis vektorius $\{\mathbf{z}\}$ gali būti pertvarkytas į tos pačios normos (“ilgio”) vektorių $\{\mathbf{z}'\}$, turinčio tik pirmąją nenulinę koordinatę;
- Kiekviena $\{\mathbf{z}'\}$ koordinatė yra vektoriaus $\{\mathbf{z}\}$ koordinačių tiesinė kombinacija. Tai seka iš matricų daugybos veiksmo apibrėžimo;

Kiekviena atspindžio matrica yra **simetrinė** ir **ortogonalioji**:

$$[\mathbf{Q}] = [\mathbf{E}] - 2\{\omega\}\{\omega\}^T;$$

$$[\mathbf{Q}]^T = ([\mathbf{E}] - 2\{\omega\}\{\omega\}^T)^T = [\mathbf{E}]^T - 2(\{\omega\}^T)^T \{\omega\}^T = [\mathbf{E}] - 2\{\omega\}\{\omega\}^T;$$

$$\begin{aligned} [\mathbf{Q}]^T [\mathbf{Q}] &= ([\mathbf{E}] - 2\{\omega\}\{\omega\}^T)([\mathbf{E}] - 2\{\omega\}\{\omega\}^T) = \\ &= [\mathbf{E}][\mathbf{E}] - 2\{\omega\}\{\omega\}^T [\mathbf{E}] - 2[\mathbf{E}]\{\omega\}\{\omega\}^T + 4\{\omega\}\{\omega\}^T \{\omega\}\{\omega\}^T = [\mathbf{E}]; \end{aligned}$$


$$[\mathbf{Q}]^{-1} = [\mathbf{Q}]^T \quad \{\omega\}^T \{\omega\} = 1$$

- Kad gauti ortogonaliosios matricos inversiją (t.y. atvirkštinę matricą), pakanka atlikti transponavimo veiksmą. Jo sudėtingumas žymiai mažesnis, nei atvirkštinės matricos apskaičiavimas pagal bendrąjį algoritmą. Jeigu skaičiuosime aritmetikos veiksmus, transponavimo veiksmo sudėtingumas = 0;
- Jeigu ortogonalioji matrica simetrinė, jos atvirkštinė matrica yra ji pati, t.y. $[\mathbf{Q}] = [\mathbf{Q}]^{-1}$
- **Svarbiausia: duotą vektorių visuomet galime pertvarkyti bet kurį kitą tos pačios normos vektorių, taikydami ortogonalųjį koordinačių pakeitimą**

**Atspindžio pakeitimo panaudojimas QR
algoritmui sukurti**

Atspindžio algoritmo taikymas lygčių sistemos pertvarkymui 1

- Duota lygčių sistemos koeficientų ir laisvųjų narių matrica:

$$[A] = \left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 2 \\ 1 & -1 & -1 & 1 & 0 \\ 2 & 1 & -1 & 2 & 9 \\ 3 & 1 & 2 & -1 & 7 \end{array} \right]$$

- Kiekvienas stulpelis laikomas vektoriumi, kuriam gali būti pritaikytas *atspindžio pakeitimas*;

- Taikant formulę $[A'] = [Q][A]$, vienas ir tas pats atspindžio pakeitimas pritaikomas kiekvienam stulpeliui. T.y. matricos $[A']$ eilutės gaunamos, tiesiškai kombinuojant matricos $[A]$ eilutes (t.y. sprendžiamos sistemos lygtis)

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\ \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \end{bmatrix}$$

$[A'] = [Q][A]$

- Matricomis $[A']$ ir $[A]$ aprašomų lygčių sistemų sprendiniai yra tokie patys

Atspindžio algoritmo taikymas lygčių sistemos pertvarkymui 3

vektorius $\{\mathbf{z}\} = \begin{Bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{Bmatrix}$

$$[\mathbf{A}] = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 \\ 1 & -1 & -1 & 1 & 0 \\ 2 & 1 & -1 & 2 & 9 \\ 3 & 1 & 2 & -1 & 7 \end{bmatrix}$$

Pertvarkomi visi stulpeliai,
tačiau nuliai gaunami tik
tame, pagal kuri buvo
apskaiciuota matrica Q.
Toks stulpelių pertvarkymas
reiškia, kad lygtys buvo
tiesiškai kombinuojamos

atspindetas vektorius $\{\mathbf{z}'\} = \text{sign}(1)\sqrt{1^2 + 1^2 + 2^2 + 3^2} \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} 3.873 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$

$$\{\boldsymbol{\omega}\} = \frac{\{\mathbf{z}\} - \{\mathbf{z}'\}}{\|\{\mathbf{z}\} - \{\mathbf{z}'\}\|}$$

$$[\mathbf{Q}^{(1)}] = [\mathbf{E}] - 2\{\boldsymbol{\omega}\}\{\boldsymbol{\omega}\}^T =$$

$$[\mathbf{A}^{(1)}] = [\mathbf{Q}^{(1)}] \begin{bmatrix} 1 & 1 & 1 & 1 & 2 \\ 1 & -1 & -1 & 1 & 0 \\ 2 & 1 & -1 & 2 & 9 \\ 3 & 1 & 2 & -1 & 7 \end{bmatrix} =$$

$$\begin{bmatrix} 0.2582 & 0.2582 & 0.5164 & 0.7746 \\ 0.2582 & 0.9101 & -0.1797 & -0.2696 \\ 0.5164 & -0.1797 & 0.6405 & -0.5392 \\ 0.7746 & -0.2696 & -0.5392 & 0.1912 \end{bmatrix}$$

$$\begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.1013 & -1.0114 & 1.0785 & -2.9886 \\ 0.0000 & 0.7974 & -1.0228 & 2.1569 & 3.0228 \\ -0.0000 & 0.6961 & 1.9658 & -0.7646 & -1.9658 \end{bmatrix}$$

Atspindžio algoritmo taikymas lygčių sistemos pertvarkymui 4

vektorius $\{z\}$

$$[A^{(1)}] = \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.1013 & -1.0114 & 1.0785 & -2.9886 \\ 0.0000 & 0.7974 & -1.0228 & 2.1569 & 3.0228 \\ -0.0000 & 0.6961 & 1.9658 & -0.7646 & -1.9658 \end{bmatrix}$$

atspindetas vektorius $\{z'\} = \text{sign}(-1.103) \sqrt{1.103^2 + 0.7974^2 + 0.6961^2} \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} -1.5275 \\ 0 \\ 0 \end{Bmatrix}$

$$\{\omega\} = \frac{\{z\} - \{z'\}}{\|\{z\} - \{z'\}\|}$$

$$[Q^{(2)}] = [E] - 2\{\omega\}\{\omega\}^T = \begin{bmatrix} 0.7210 & -0.5220 & -0.4557 \\ -0.5220 & 0.0233 & -0.8526 \\ -0.4557 & -0.8526 & 0.2557 \end{bmatrix}$$

Pertvarkomas tik matricos blokas (2:n,2:n+1) (t.y. pirmoji lygtis nebekinta). Todėl erdvės, kurioje atliekamas atspindys, išmatavimas sumažėja.

$$[Q^{(2)}] \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.1013 & -1.0114 & 1.0785 & -2.9886 \\ 0.0000 & 0.7974 & -1.0228 & 2.1569 & 3.0228 \\ -0.0000 & 0.6961 & 1.9658 & -0.7646 & -1.9658 \end{bmatrix} = \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0 & -1.1719 & 0.1393 & 3.3067 \\ -0.0000 & -0.0000 & 1.8356 & -2.5260 & -1.7179 \end{bmatrix} = [A^{(2)}]$$

Atspindžio algoritmo taikymas lygčių sistemos pertvarkymui 5

vektorius $\{z\}$

$$[A^{(2)}] = \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0 & -1.1719 & 0.1393 & 3.3067 \\ -0.0000 & -0.0000 & 1.8356 & -2.5260 & -1.7179 \end{bmatrix}$$

atspindetas vektorius $\{z'\} = \text{sign}(-1.1719) \sqrt{1.1719^2 + 1.8356^2} \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} = \begin{Bmatrix} -2.1778 \\ 0 \end{Bmatrix}$

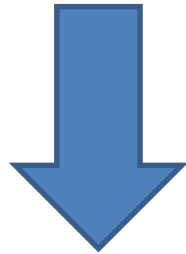
$$\{\omega\} = \frac{\{z\} - \{z'\}}{\|\{z\} - \{z'\}\|}$$

$$[Q^{(3)}] = [E] - 2\{\omega\}\{\omega\}^T = \begin{bmatrix} 0.5381 & -0.8429 \\ -0.8429 & -0.5381 \end{bmatrix}$$

$$\begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0 & -1.1719 & 0.1393 & 3.3067 \\ -0.0000 & -0.0000 & 1.8356 & -2.5260 & -1.7179 \end{bmatrix} [Q^{(3)}] = \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0.0000 & -2.1778 & 2.2040 & 3.2274 \\ -0.0000 & 0.0000 & -0.0000 & 1.2418 & -1.8628 \end{bmatrix} = [A^{(3)}]$$

Atspindžio algoritmo taikymas lygčių sistemos pertvarkymui 6

$$\begin{bmatrix} \mathbf{A}^{(3)} \end{bmatrix} = \begin{array}{cccc|c} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0.0000 & -2.1778 & 2.2040 & 3.2274 \\ -0.0000 & 0.0000 & -0.0000 & 1.2418 & -1.8628 \end{array}$$



Atliekamas Gauso algoritmo
atgalinis etapas

$\mathbf{x} =$

2.5000
4.0000
-3.0000
-1.5000

Atspindžio algoritmas esant *singuliariai matricai*

1.0000	1.0000	1.0000	1.0000	2.0000
1.0000	1.0000	-1.0000	1.0000	9.1429
1.0000	1.0000	-2.0000	4.0000	14.0000
-1.0000	-1.0000	1.0000	4.0000	-7.0000



Jeigu gauname **nulinį stulpelį** $z=0$,
atspindžio pertvarkymo netaikome, t.y.
vieną ciklo žingsnį praleidžiame

2.0000	2.0000	-1.5000	1.0000	16.0714
0	0	1.5000	1.0000	-4.9286
0	0	0.5000	4.0000	-0.0714
0	0	-1.5000	4.0000	7.0714



2.0000	2.0000	-1.5000	1.0000	16.0714
0	0	1.5000	1.0000	-4.9286
0	0	1.5811	-2.5298	-6.7311
0	0	-0.0000	-5.0596	-2.1684

Tiesioginis žingsnis

Atgalinis žingsnis toks pats, kaip ir Gauso algoritmo, kai
matrica singuliari

[Pvz_SMA_3_3_atspindzio_QR_algoritmas_singular.m](#)

Atspindžio algoritmas ir Gauso algoritmas – tikslumo sulyginimas

Atspindžio algoritmas

$$[A^{(3)}] = \begin{bmatrix} 3.8730 & 1.2910 & 1.0328 & 0.7746 & 10.5862 \\ 0.0000 & -1.5275 & -1.0911 & -0.0000 & -2.8368 \\ 0.0000 & 0.0000 & -2.1778 & 2.2040 & 3.2274 \\ -0.0000 & 0.0000 & -0.0000 & 1.2418 & -1.8628 \end{bmatrix}$$



x =
2.5000
4.0000
-3.0000
-1.5000

liekana = 1e-14 *
0.1332
0.0444
0
-0.1776

Gauso algoritmas, parenkant didžiausią vedantįjį elementą

$$A1 = \begin{bmatrix} 3.0000 & 1.0000 & 2.0000 & -1.0000 & 7.0000 \\ 0 & -1.3333 & -1.6667 & 1.3333 & -2.3333 \\ 0 & 0 & -2.7500 & 3.0000 & 3.7500 \\ 0 & 0 & 0 & 1.4545 & -2.1818 \end{bmatrix}$$

x =
2.5000
4.0000
-3.0000
-1.5000

liekana = 1e-14 *
0
-0.0888
0.3553
0

Taikydami **atspindžio algoritmą**, didžiausią
liekaną gavome mažesnę, o liekanas –
tolygiau pasiskirsčiusias tarp sistemos lygčių,
negu taikydami **Gauso algoritmą**

Nesuklyskime, eksperimentiškai vertindami algoritmų savybes:

Gauso algoritmas, parenkant didžiausią vedantįjį elementą

$$A1 = \begin{pmatrix} 3.0000 & 1.0000 & 2.0000 & -1.0000 & 7.0000 \\ 0 & -1.3333 & -1.6667 & 1.3333 & -2.3333 \\ 0 & 0 & -2.7500 & 3.0000 & 3.7500 \\ 0 & 0 & 0 & 1.4545 & -2.1818 \end{pmatrix}$$

x =
2.5000
4.0000
-3.0000
-1.5000

liekana = 1e-14 *
0
-0.0888
0.3553
0

Gauso algoritmas, kai vedantis elementas imamas toks, kokį gauname įstrižainėje

$$A1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 2 \\ 0 & -2 & -2 & 0 & -2 \\ 0 & 0 & -2 & 0 & 6 \\ 0 & 0 & 0 & -4 & 6 \end{pmatrix}$$

x =
2.5000
4.0000
-3.0000
-1.5000

liekana =
0
0
0
0

Tiesioginį algoritmo žingsnį pavyko atlikti sveikaisiais skaičiais. Bendruoju atveju taip nepavyktų, todėl paklaidos gautųsi didesnės. Paeksperimentuokite.

Atspindžio algoritmas MATLAB

Pvz_SMA_3_2_atspindzio_QR_algoritmas.m

```
A1=[A,b]
```

```
% Tiesioginis etapas (atspindziai):
```

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

```
    z=A1(i:n,i);
```

```
    zp=zeros(size(z)); zp(1)=norm(z);
```

```
    omega=(z-zp); omega=omega/norm(omega);
```

```
    Q=eye(n-i+1)-2*omega*omega';
```

```
    A1(i:n,:)=Q*A1(i:n,:);
```

```
end
```

```
% Atgalinis etapas toks pats, kaip ir Gauso algoritmo:
```

```
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
```

Atspindžio algoritmas Python

Pvz_SMA_3_02_atspindzio_QR_algoritmas.m

```
A1=np.hstack((A,b))
```

```
# tiesioginis etapas(atspindziai):
```

```
for i in range (0,n-1):
```

```
    z=A1[i:n,i]
```

```
    zp=np.zeros(np.shape(z)); zp[0]=np.linalg.norm(z)
```

```
    omega=z-zp; omega=omega/np.linalg.norm(omega)
```

```
    Q=np.identity(n-i)-2*omega*omega.transpose()
```

```
    A1[i:n,:]=Q.dot(A1[i:n,:])
```

```
# atgalinis etapas:
```

```
x=np.zeros(shape=(n,1))
```

```
for i in range (n-1,-1,-1):
```

```
    x[i,:]=(A1[i,n:n+nb]-A1[i,i+1:n]*x[i+1:n,:])/A1[i,i]
```

Atspindžio algoritmas: apibendrinimai (1)

- Atspindžio algoritmas paremtas lygčių tiesiniu kombinavimu, siekiant gauti nulius žemiau matricos pagrindinės įstrižainės. Todėl *atspindžio algoritmas yra Gauso algoritmo apibendrinimas*;
- *Gauso algoritme* tiesinės kombinacijos gaunamos, pridedant ar atimant vedančią lygtį iš žemiau esančių. *Atspindžio algoritme* tiesinėse kombinacijose dalyvauja visos lygtys;

Atspindžio algoritmas: apibendrinimai (2)

- *Atspindžio algoritme* po kiekvieno atspindžio matricos stulpelių normos nepakinta. Todėl tikėtina, kad lygčių tiesinėmis kombinacijomis nesukursime stulpelių, kurių visi koeficientai absoliutiniu dydžiu yra labai maži ;
- Nereiktų galvoti, kad *atspindžio algoritmu* gauti sprendiniai visuomet tikslesni, nei gautieji *Gauso algoritmu*. Galima pateikti tam prieštaraujančių pavyzdžių, ypač kai lygčių skaičius nedidelis, arba kai lygtis išsprendžiama sveikaisiais skaičiais;
- Svarbiausia tai, kad *atspindžio algoritmu* pertvarkant lygčių sistemą, galima jai suteikti tam tikrą reikalingą pavidalą. Vystant algoritmą toliau, galima atlikti matricos QR skaidą, sukurti tikrinių reikšmių apskaičiavimo algoritmus ir pan.

QR skaidos algoritmas

QR skaida

Atspindžio algoritmas išskaido lygčių sistemos matricą į **ortogonalųjį** ir **trikampį** daugiklius:

$$\underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \\ \vdots & \vdots \\ \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \end{bmatrix}}_{[Q]^T} \dots \begin{bmatrix} 1 & [0 & \dots & 0] \\ \vdots & \vdots \\ 0 & [Q^{(2)}] \end{bmatrix} [Q^{(1)}][A] = [R]$$

Ortogonalųjų matricų sandauga yra ortogonalioji matrica:

$$[Q]^T [A] = [R]$$

$$[A] = [Q][R]$$

← QR skaida

Ortogonalųjų matricų sandauga yra ortogonalioji matrica - **įrodytas**:

$$\begin{aligned}
 & ([\mathbf{C}_1][\mathbf{C}_2]\cdots[\mathbf{C}_k])^T ([\mathbf{C}_1][\mathbf{C}_2]\cdots[\mathbf{C}_k]) = \\
 & = [\mathbf{C}_k]^T \cdots [\mathbf{C}_2]^T \underbrace{[\mathbf{C}_1]^T [\mathbf{C}_1]}_{[\mathbf{E}]} \underbrace{[\mathbf{C}_2] \cdots [\mathbf{C}_2]}_{[\mathbf{E}]} \cdots [\mathbf{C}_k] = [\mathbf{E}]
 \end{aligned}$$

Ortogonalūs daugikliai

Todėl matrica $[\mathbf{Q}]$ yra **ortogonalioji**, tačiau **nebe simetrinė(!)** matrica:

$$\underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} & \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} \mathbf{Q}^{(n-1)} \end{bmatrix} \end{bmatrix}}_{[\mathbf{Q}]^T} \cdots \begin{bmatrix} \begin{bmatrix} 1 & [0 \cdots 0] \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} \mathbf{Q}^{(2)} \end{bmatrix} \end{bmatrix} [\mathbf{Q}^{(1)}][\mathbf{A}] = [\mathbf{R}]$$

Skirtingų simetrinių matricų sandauga **nėra** simetrinė matrica

Matricos [Q] apskaičiavimas

$$[\mathbf{Q}]^T = \begin{bmatrix} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} & \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & \dots & 0 \end{bmatrix} & [\mathbf{Q}^{(n-1)}] \end{bmatrix} \dots \begin{bmatrix} 1 & [0 & \dots & 0] \\ \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} & [\mathbf{Q}^{(2)}] \end{bmatrix} [\mathbf{Q}^{(1)}]$$

$$[\mathbf{Q}] = [\mathbf{Q}^{(1)}] \begin{bmatrix} 1 & [0 & \dots & 0] \\ \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} & [\mathbf{Q}^{(2)}] \end{bmatrix} \dots \begin{bmatrix} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} & \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \\ [0 & \dots & 0] & [\mathbf{Q}^{(n-1)}] \end{bmatrix}$$

Visi daugikliai yra simetrinės matricos:

$$[\mathbf{Q}^{(i)}] = [\mathbf{Q}^{(i)}]^T$$

QR skaidos taikymas tiesinių lygčių sistemų sprendimui

$$[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}; \quad [\mathbf{A}] = [\mathbf{Q}][\mathbf{R}]$$



$$[\mathbf{Q}][\mathbf{R}]\{\mathbf{x}\} = \{\mathbf{b}\}$$

Trikampė matrica $\rightarrow [\mathbf{R}]\{\mathbf{x}\} = [\mathbf{Q}]^T \{\mathbf{b}\}$

- **Privalumas:** Kartą apskaičiavę skaidos daugiklius, juos galime naudoti pakartotinai, esant vis kitokiam laisvųjų narių vektoriui;
- **Privalumas:** Trikampėje matricoje stulpelių normos lieka tokios pačios, kaip ir išeities matricoje
- **Privalumas:** Algoritmas veikia ir tuo atveju, kai matricos R įstrižainėje yra nulių. T. y. skaida galioja ir singularioms lygčių sistemoms;
- **Trūkumas:** Daugikliai užima daugiau atminties, nei pradinė matrica

QR skaidos algoritmas MATLAB

Pvz_SMA_3_4_sistemas_sprendimas_taikant_QR_skaida.m

% Tiesioginis etapas - QR skaida

```
Q=eye(n);
```

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

```
    z=A(i:n,i);
```

```
    zp=zeros(size(z));
```

```
    zp(1)=sign(z(1))*norm(z);
```

```
    omega=(z-zp); omega=omega/norm(omega);
```

```
    Qi=eye(n-i+1)-2*omega*omega';
```

```
    A(i:n,:)=Qi*A(i:n,:);
```

```
    % Q=Q*[eye(i-1),zeros(i-1,n-i+1);zeros(n-i+1,i-1),Qi];
```

```
    Q(:,i:n)= Q(:,i:n)*Qi;
```

```
end
```

% Atgalinis etapas


```
b1=Q'*b;
```

```
x=zeros(n,nb);
```

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

```
    x(i,:)=(b1(i,:)-A(i,i+1:n)*x(i+1:n,:))/A(i,i);
```

```
end
```

$$\begin{bmatrix} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \end{bmatrix} \quad [Q^{(i)}]$$


QR skaidos algoritmas Python

Pvz_SMA_3_04_sistemas_sprendimas_taikant_QR_skaida.py

```
Q=np.identity(n)
for i in range (0,n-1):
    z=A[i:n,i]
    zp=np.zeros(np.shape(z)); zp[0]=np.linalg.norm(z)

    omega=z-zp; omega=omega/np.linalg.norm(omega)
    Qi=np.identity(n-i)-2*omega*omega.transpose()
    A[i:n,:]=Qi.dot(A[i:n,:])
```

$$\left[\begin{array}{ccc|ccc} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ \hline 0 & \dots & 0 & & & \\ \vdots & \ddots & \vdots & & & \\ 0 & \dots & 0 & & & \end{array} \right] [Q^{(i)}]$$

```
Q=Q.dot(
    np.vstack(
        (
            np.hstack((np.identity(i),np.zeros(shape=(i,n-i)))),
            np.hstack((np.zeros(shape=(n-i,i)),Qi))
        )
    )
)
```

atgalinis etapas:

```
b1=Q.transpose().dot(b);
x=zeros(shape=(n,nb));
for i in range (n-1,-1,-1):
    x[i,:]=(b1[i,:]-A[i,i+1:n]*x[i+1:n,:])/A[i,i];
```


QR skaidos algoritmas Python (ver.2)

Pvz_SMA_3_04_sistemas_sprendimas_taikant_QR_skaida.py


```
Q=np.identity(n)
for i in range (0,n-1):
    z=A[i:n,i]
    zp=np.zeros(np.shape(z)); zp[0]=np.linalg.norm(z)

    omega=z-zp; omega=omega/np.linalg.norm(omega)
    Qi=np.identity(n-i)-2*omega*omega.transpose()
    A[i:n,:]=Qi.dot(A[i:n,:])
```

```
Q[:,i:n]= Q[:,i:n].dot(Qi)
```

atgalinis etapas:

```
b1=Q.transpose().dot(b);
x=zeros(shape=(n,nb));
for i in range (n-1,-1,-1):
    x[i,:]=(b1[i,:]-A[i,i+1:n]*x[i+1:n,:])/A[i,i];
```

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^{i+1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_1^i \\ \mathbf{Q}_2^i \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_1^i \\ \mathbf{Q}^{i+1} \mathbf{Q}_2^i \end{bmatrix}$$


**Kokios yra algebrinės lygties sprendimui
taikomos programinės funkcijos MATLAB ir
Python terpėse**

MATLAB funkcijos algoritmams, paremtiems koeficientų matricos pertvarkymu

$x=A\backslash b$ % “atvirkštinė dalyba”. Algoritmas paremtas LU skaida

$[L,U,P]=lu(A,'vector')$ % LU skaida. Gaunami trikampiai
% daugikliai ir lygciu eiles tvarkos
% pakeitimu vektorius P

$X=U\backslash(L\backslash b)$ % savaime nustato, kad matrica trikampė

$L = chol(A)$ % Choleckio skaida

$[Q,R] = qr(A)$ % QR skaida

$x = linsolve(A,B)$ % kombinuotas metodas, taiko LU skaida ir
% atspindzius

$Y = inv(X)$ % atvirkstines matricos apskaiciavimas.

% Neekonomiska naudoti $x=inv(A)*b$, geriau $x=A\backslash b$

Python funkcijos algoritmams, paremtiems koeficientų matricos pertvarkymu

`X=np.linalg.solve(A,b)` # TLS sprendimas

`P,L,U=scipy.linalg.lu(A)` # LU skaida. Gaunamas lygciu eiles
tvarkos pakeitimu vektorius P ir
trikampiai daugikliai

`LU,P=scipy.linalg.lu_factor(A)` # LU daugikliai vienoje matricoje
`X=scipy.linalg.lu_solve((LU,P),b)` #sprendimas panaudojant LU

`L = np.linalg.cholesky(A)` # Choleckio skaida

`Q,R = np.linalg.qr(A)` # QR skaida

`Y = np.linalg.inv(X)` # atvirkstines matricos apskaiciavimas

SMA_03_Klausimai savikontrolei(1):

1. Kas yra atspindžio matrica;
2. Kaip rasti atspindinčios plokštumos normalės vektorių, kai žinomas vektorius ir jo atspindys;
3. Kokią sąlygą turi tenkinti vektorius, kad jį būtų galima laikyti atspindžio vektoriumi duotajam vektoriui;
4. Ką reiškia matricos ortogonalumas. Įrodykite, kad atspindžio matrica visuomet ortogonalioji ;
5. Paaiškinkite, kodėl išplėstą lygčių sistemos koeficientų matricą padauginus iš atspindžio matricos kairėje pusėje, jos sprendinys nepasikeičia;
6. Paaiškinkite, kaip sprendžiama lygčių sistema, taikant atspindžio metodą;
7. Ar galima taikyti atspindžio algoritmą, kai lygčių sistemos koeficientų matrica yra singuliari;

SMA_03_Klausimai savikontrolei(2):

8. Kodėl taikant atspindžio algoritmą, pertvarkytos koeficientų matricos stulpelių normos nepakinta;
9. Kas yra QR skaida;
10. Ką galima pasakyti apie QR skaidos daugiklio Q ortogonalumą ir simetriškumą;
11. Kaip QR skaida pritaikoma lygčių sistemai spręsti. Kuo toks būdas geresnis už tiesiogiai pritaikytą atspindžio metodą;