

Tiesinių lygčių sistemų sprendimas:

- *Iteraciniai metodai*
- *Sprendinio tikslumas*

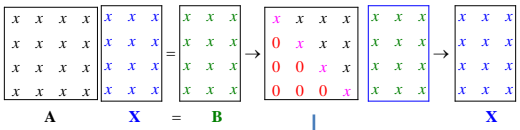
Temoje aiškinama:

- **TLS tiesioginių sprendimo metodų santrauka. Iteracinių sprendimo metodų poreikis;**
- Paprastųjų iteracijų algoritmas TLS sprendimui;
- **Gauso-Zeidelio algoritmas;**
- TLS laisvųjų narių vektoriaus paklaidos įtaka sprendinio paklaidai. Matricos sąlygotumo skaičius;

**TLS tiesioginių sprendimo metodų santrauka.
Iteracinių sprendimo metodų poreikis**

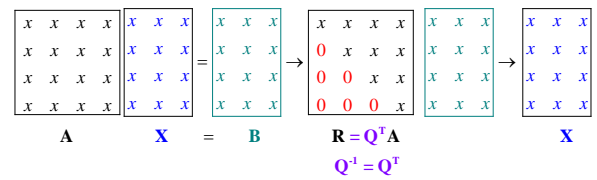
Prisiminkime: tiesioginiai (kintamųjų eliminavimu paremti) algoritmai

Gauso



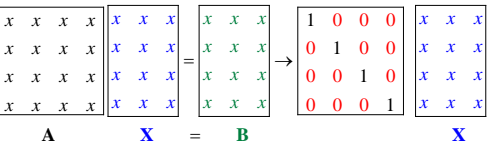
det(A)

Atspindžio

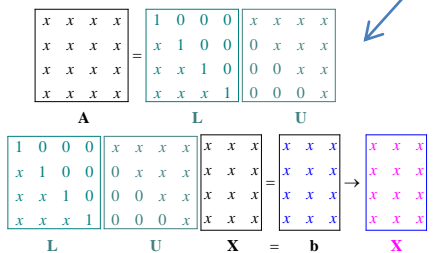


Choleckio skaida
 $A = L^T L$, kai $A = A^T$

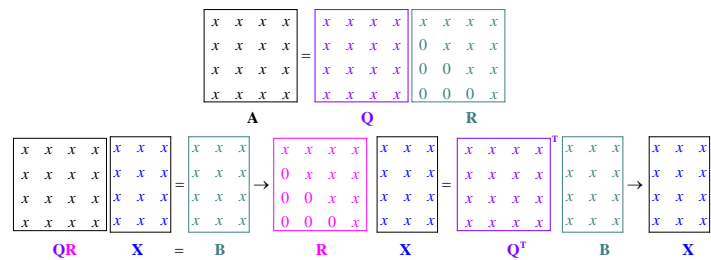
Gauso-Žordano



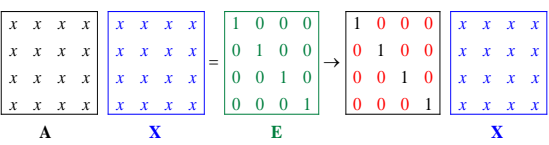
LU skaida



QR skaida



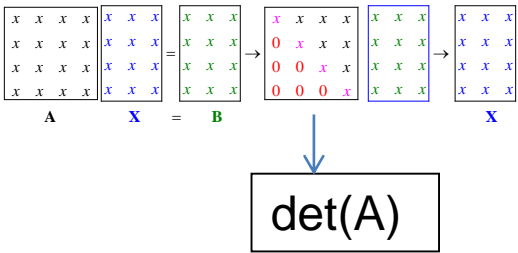
Atvirkštinė matrica



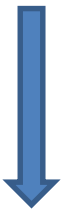
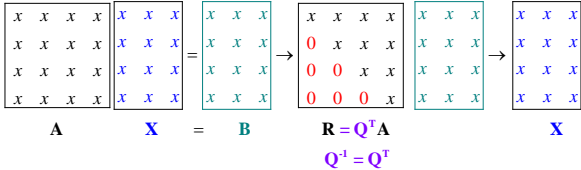
Prisiminkime:

tik šie algoritmai veikia, esant **singuliariai koeficientų matricai**

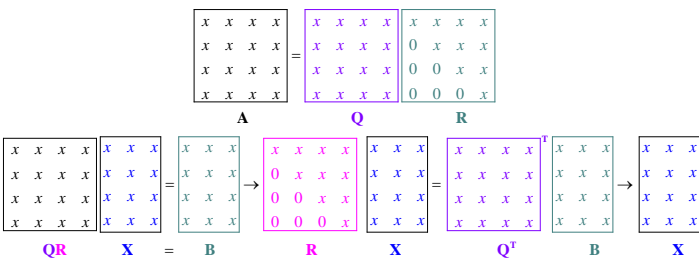
Gauso



Atspindžio



QR skaida

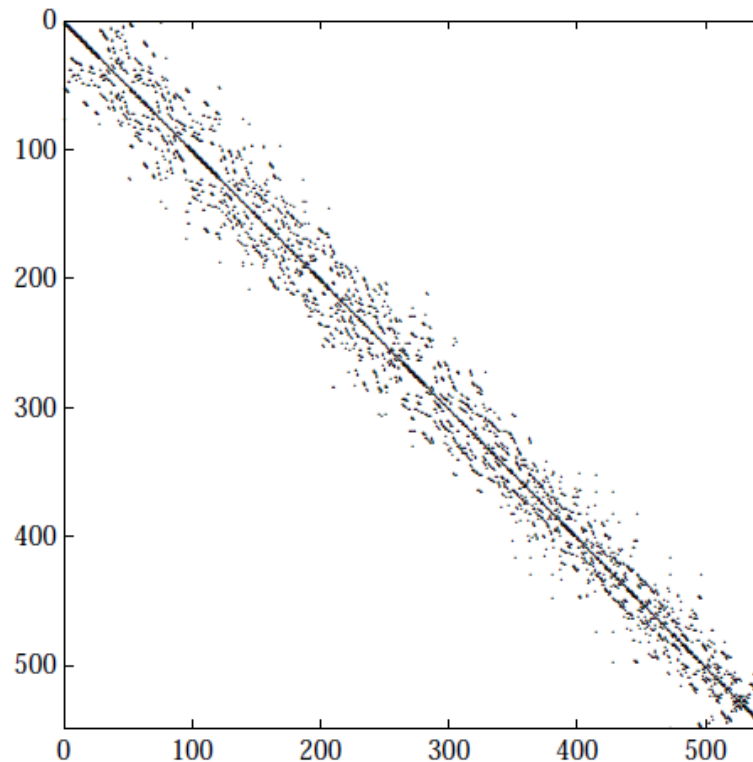


Prisiminkime: skaitiniai tiesinių algebrinių lygčių sistemų sprendimo algoritmai

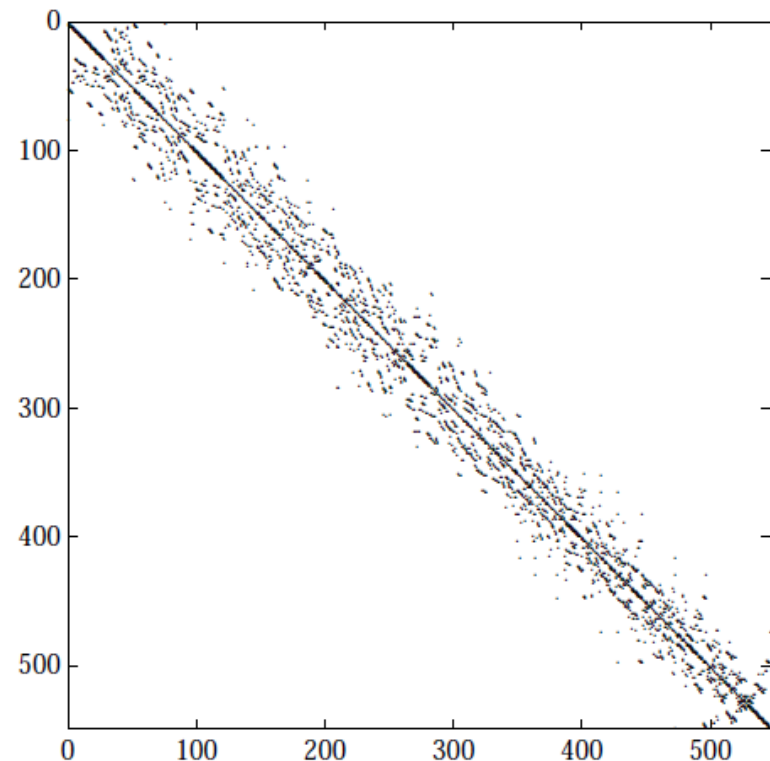
- *Tiesioginiai* – sprendinys gaunamas algeбриškai pertvarkant lygčių sistemą (t.y. koeficientų matrica skaičiuojant pertvarkoma)
- ***Iteraciniai* – koeficientų matrica išlieka nepakitusi. Sprendinį apskaičiuojame nuosekliais artiniais**

Iteraciniai TLS sprendimo metodai ypač svarbūs, kai TLS matricos yra retosios (*sparse*)

- Pasitaiko, kad koeficientų matricoje yra daug nulinių koeficientų. Tokias matricas vadiname **retosiomis**. Jas saugant stačiakampiame pavidale, kompiuterio atmintis būtų panaudojama labai neekonomiškai. Joms saugoti naudojami specialūs kompiuterio atmintį taupantys duomenų formatai;



- Taikant tiesioginius TLS sprendimo algoritmus, sprendimo metu koeficientų matricos retoji struktūra gali pasikeisti ir užimti žymiai daugiau atminties, nei pradinė matrica;
- Srendžiant TLS iteraciniais metodais, koeficientų matrica nekinta. Pakanka turėti našų algoritmą, kuris padaugina retąją matricą iš vektoriaus



Paprastųjų iteracijų algoritmas TLS sprendimui

***Paprastųjų iteracijų* algoritmo taikymas tiesinių lygčių sistemai (1)**

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

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix};$$

Jeigu įstrižainėje yra “0” reikšmė, reikia sukeisti vietomis lygtis arba kintamuosius. Pavyzdžiui, sukeitus vietomis kintamuosius x_2 ir x_3 , susikeičia vietomis 2-as ir 3-ias matricos A stulpeliai

$$\begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1 & \frac{a_{23}}{a_{22}} \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \end{Bmatrix};$$

Metodas konverguoja greičiau, kai įstrižainėje yra absoliutiniu dydžiu didesni koeficientai

Paprastųjų iteracijų algoritmo taikymas tiesinių lygčių sistemai (2)

Laisvai parinkti skaičiai, nuo kurių galėtų priklausyti konvergavimo sparta. Pradžioje dažniausiai imama $\alpha=1$

$$\begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{bmatrix} 1-\alpha_1 & \frac{a_{12}}{a_{11}} & \frac{a_{23}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1-\alpha_2 & \frac{a_{23}}{a_{22}} \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 1-\alpha_3 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_1}{a_{22}} \\ \frac{b_1}{a_{33}} \end{bmatrix};$$

$$[\alpha] \{ \mathbf{x} \} + [\tilde{\mathbf{A}}] \{ \mathbf{x} \} = \{ \tilde{\mathbf{b}} \}$$

$$\{ \mathbf{x} \}^{(k+1)} = [\alpha]^{-1} \left(\{ \tilde{\mathbf{b}} \} - [\tilde{\mathbf{A}}] \{ \mathbf{x} \}^{(k)} \right)$$

Paprastųjų iteracijų algoritmas

$\{ \mathbf{x} \}^{(0)}$ - pradinis artinys

Iteracijų pabaigos sąlyga:

$$\frac{\left\| \{\mathbf{X}\}^{(k+1)} - \{\mathbf{X}\}^{(k)} \right\|}{\left\| \{\mathbf{X}\}^{(k+1)} \right\| + \left\| \{\mathbf{X}\}^{(k)} \right\|} < \varepsilon$$

Suprastintas vektorių ir matricų žymėjimas

$$\{\mathbf{x}\}^{(k+1)} = [\mathbf{a}]^{-1} \left(\{\tilde{\mathbf{b}}\} - [\tilde{\mathbf{A}}] \{\mathbf{x}\}^{(k)} \right)$$



$$\mathbf{x}^{(k+1)} = [\mathbf{a}]^{-1} \left(\tilde{\mathbf{b}} - \tilde{\mathbf{A}} \mathbf{x}^{(k)} \right)$$

$$[\mathbf{a}] = \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_n \end{bmatrix}$$

- Dažniausiai vektorius žymimas riestiniais { }, o matrica laužtiniais [] skliausteliais;
- Siekiant mažesnio simbolių skaičiaus formulėse, vektoriai ir matricos gali būti žymimi be skliaustelių, t.y. vien tik storesnių linijų (Bold) šriftu. Jų elementai žymimi įprastiniu šriftu;
- Vektorius dažniausiai žymimas mažąja, o matrica – didžiąja raide. Esant suprastintam žymėjimui, tik taip juos galime atskirti vieną nuo kito;
- Pagal reikalą, papildomai galima panaudoti ir skliaustelius. Taip pabrėžiame, kad dydis yra matrica arba vektorius

Paprastųjų iteracijų algoritmas MATLAB

Pvz_SMA_4_1_Paprastuju_iteraciju_ir_Gauso_Zeidelio_algoritmai.m

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

$$\begin{bmatrix} \frac{1}{a_{11}} & 0 & 0 \\ 0 & \frac{1}{a_{22}} & 0 \\ 0 & 0 & \frac{1}{a_{33}} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{23} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} 1-\alpha_1 & \frac{a_{12}}{a_{11}} & \frac{a_{23}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1-\alpha_2 & \frac{a_{23}}{a_{22}} \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 1-\alpha_3 \end{bmatrix}$$

```
alpha=[1,1,1,1]; % metodo parametrai
Atld=diag(1./diag(A))*A-diag(alpha);
btld=diag(1./diag(A))*b;
```

```
nitmax=1000; eps=1e-12;
```

$$\begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix}$$

```
x=zeros(n,1); %pradinis artinys
```

```
for it=1:nitmax
```

```
    x1=(btld-Atld*x)./alpha;
```

$$\{\mathbf{x}\}^{(k+1)} = [\mathbf{a}]^{-1} \left(\{\tilde{\mathbf{b}}\} - [\tilde{\mathbf{A}}] \{\mathbf{x}\}^{(k)} \right)$$

```
    tikslumas=norm(x1-x)/(norm(x)+norm(x1));
```

```
    if tikslumas < eps, break, end
```

```
    x=x1;
```

$$\frac{\|\{\mathbf{x}\}^{(k+1)} - \{\mathbf{x}\}^{(k)}\|}{\|\{\mathbf{x}\}^{(k+1)}\| + \|\{\mathbf{x}\}^{(k)}\|} < \varepsilon$$

```
end
```

Paprastųjų iteracijų algoritmas Python

Pvz_SMA_4_01_Paprastuju_iteraciju_ir_Gauso_Zeidelio_algoritmai.py

```

A=np.matrix([[ 1, -1,  0,  0],
              [-1,  2, -1,  0],
              [ 0, -1,  2, -1],
              [ 0,  0, -1,  2]]).astype(np.float)
b=np.array([[2],[0],[0],[0]]).astype(np.float)
n=np.shape(A)[0]

alpha=np.array([1, 1, 1, 1]) # laisvai parinkti metodo parametrai
Atld=np.diag(1./np.diag(A)).dot(A)-np.diag(alpha)
btld=np.diag(1./np.diag(A)).dot(b)

nitmax=1000; eps=1e-12;

x=np.zeros(shape=(n,1));x1=np.zeros(shape=(n,1));

for it in range (0,nitmax):
    x1=((btld-Atld.dot(x)).transpose()/alpha).transpose();
    prec=np.linalg.norm(x1-x)/(np.linalg.norm(x)+np.linalg.norm(x1))
    if prec < eps : break
    x[:]=x1[:]
```

A =

1	1	1	1
1	-5	-1	1
2	1	-10	2
3	1	2	-10

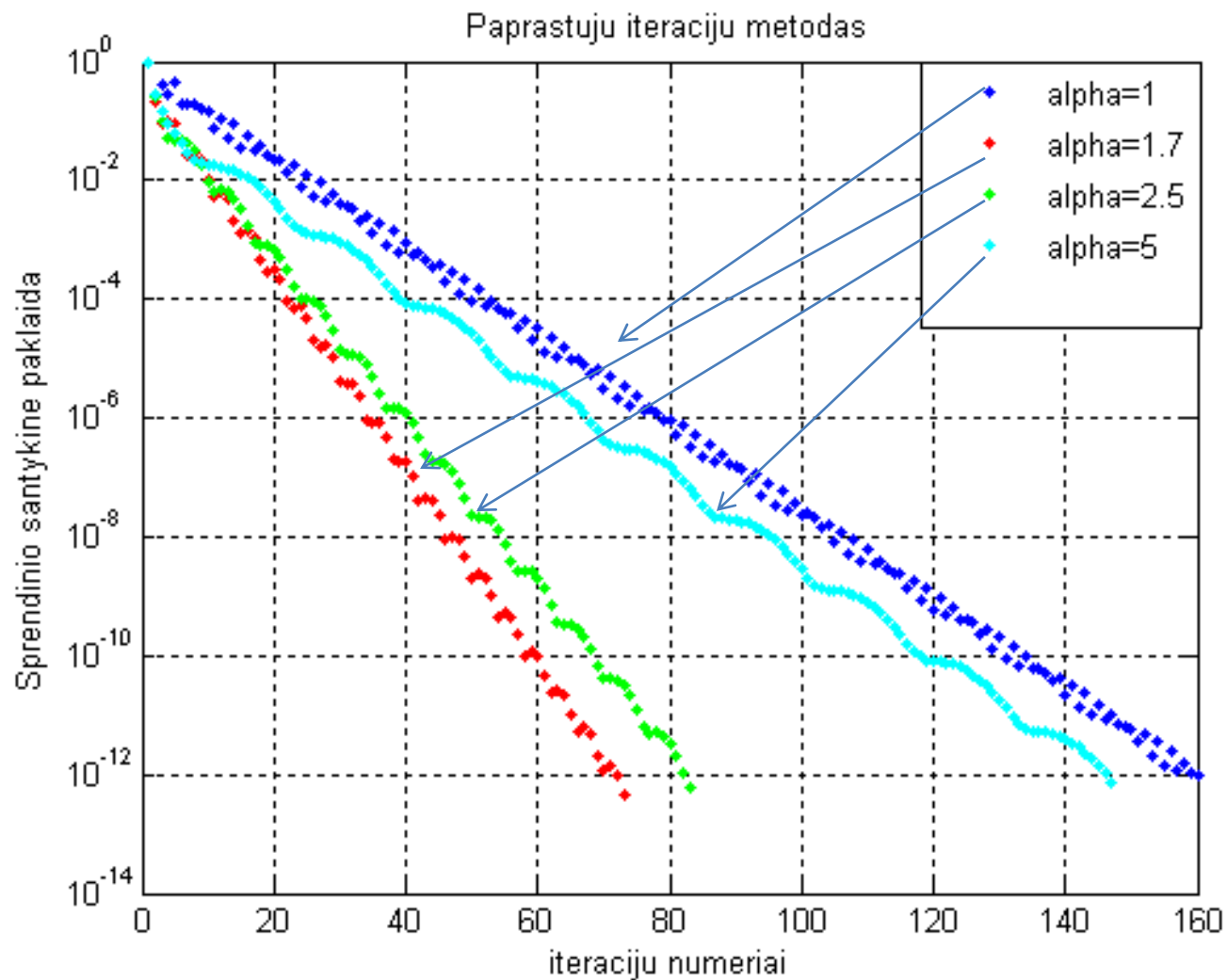
b =

2
0
9
-7

x =

1.0110
0.4857
-0.4571
0.9604

Paprastųjų iteracijų metodo **konvergavimo sparta** esant skirtingoms α reikšmėms



Gauso-Zeidelio algoritmas

Gauso-Zeidelio iteracijų algoritmo taikymas tiesinių lygčių sistemai

$$\mathbf{x}^{(k+1)} = \boldsymbol{\alpha}^{-1} \left(\tilde{\mathbf{b}} - \tilde{\mathbf{A}} \mathbf{x}^{(k)} \right)$$

Paprastųjų iteracijų algoritmas



$$x_i^{(k+1)} = \frac{1}{\alpha_i} \left(\tilde{b}_i - \sum_{j=1}^n \tilde{a}_{ij} x_j^{(k)} \right), \quad i = 1:n$$



$$x_i^{(k+1)} = \frac{1}{\alpha_i} \left(\tilde{b}_i - \sum_{j=1}^{i-1} \tilde{a}_{ij} x_j^{(k+1)} - \sum_{j=i}^n \tilde{a}_{ij} x_j^{(k)} \right), \quad i = 1:n$$

Gauso-Zeidelio iteracijų algoritmas

Gauso-Zeidelio iteracijų algoritmas MATLAB

Pvz_SMA_4_1_Paprastuju_iteraciju_ir_Gauso_Zeidelio_algoritmai.m

```
A=[ 1   -1   0   0;  
    -1   2  -1   0;  
     0  -1   2  -1;  
     0   0  -1   2]
```

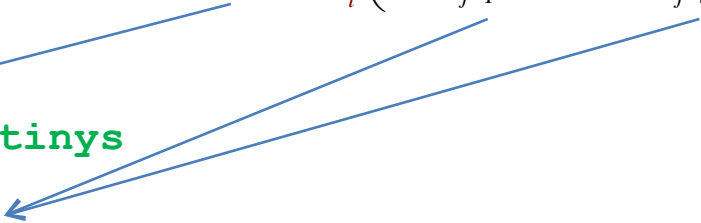
```
b=[2;0;0;0]
```

```
n=size(A,1)
```

```
alpha=[1,1,1,1]; % metodo parametrai
```

```
Atld=diag(1./diag(A))*A- diag(alpha);
```

```
btld=diag(1./diag(A))*b;
```

$$x_i^{(k+1)} = \frac{1}{\alpha_i} \left(\tilde{b}_i - \sum_{j=1}^{i-1} \tilde{a}_{ij} x_j^{(k+1)} - \sum_{j=i}^n \tilde{a}_{ij} x_j^{(k)} \right), \quad i=1:n$$


```
nitmax=1000; eps=1e-12;
```

```
x=zeros(n,1); x1=x; % pradinis artinys
```

```
for it=1:nitmax
```

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

```
        x1(i)=(btld(i)- Atld(i,:)*x1)/alpha(i);
```

```
    end
```

```
    tikslumas=norm(x1-x)/(norm(x)+norm(x1));
```

```
    if tikslumas < eps, break, end
```

```
    x=x1;
```

```
end
```

Gauso-Zeidelio iteracijų algoritmas Python

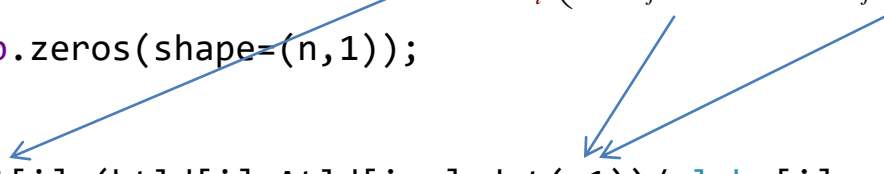
Pvz_SMA_4_01_Paprastuju_iteraciju_ir_Gauso_Zeidelio_algoritmai

```
A=np.matrix([[ 1, -1,  0,  0],
             [-1,  2, -1,  0],
             [ 0, -1,  2, -1],
             [ 0,  0, -1,  2]]).astype(np.float)
b=np.array([[2],[0],[0],[0]]).astype(np.float)
n=np.shape(A)[0]
```

```
alpha=np.array([1, 1, 1, 1]) # metodo parametrai
Atld=np.diag(1./np.diag(A)).dot(A)-np.diag(alpha)
btld=np.diag(1./np.diag(A)).dot(b)
```

```
nitmax=1000; eps=1e-12
x=np.zeros(shape=(n,1));x1=np.zeros(shape=(n,1));
```

```
for it in range (0,nitmax):
    for i in range (0,n) : x1[i]=(btld[i]-Atld[i,:].dot(x1))/alpha[i];
```

$$x_i^{(k+1)} = \frac{1}{\alpha_i} \left(\tilde{b}_i - \sum_{j=1}^{i-1} \tilde{a}_{ij} x_j^{(k+1)} - \sum_{j=i}^n \tilde{a}_{ij} x_j^{(k)} \right), \quad i = 1:n$$


```
prec=(np.linalg.norm(x1-x)/(np.linalg.norm(x)+np.linalg.norm(x1)))
if prec < eps : break
x[:]=x1[:]
```

A =

1	1	1	1
1	-5	-1	1
2	1	-10	2
3	1	2	-10

b =

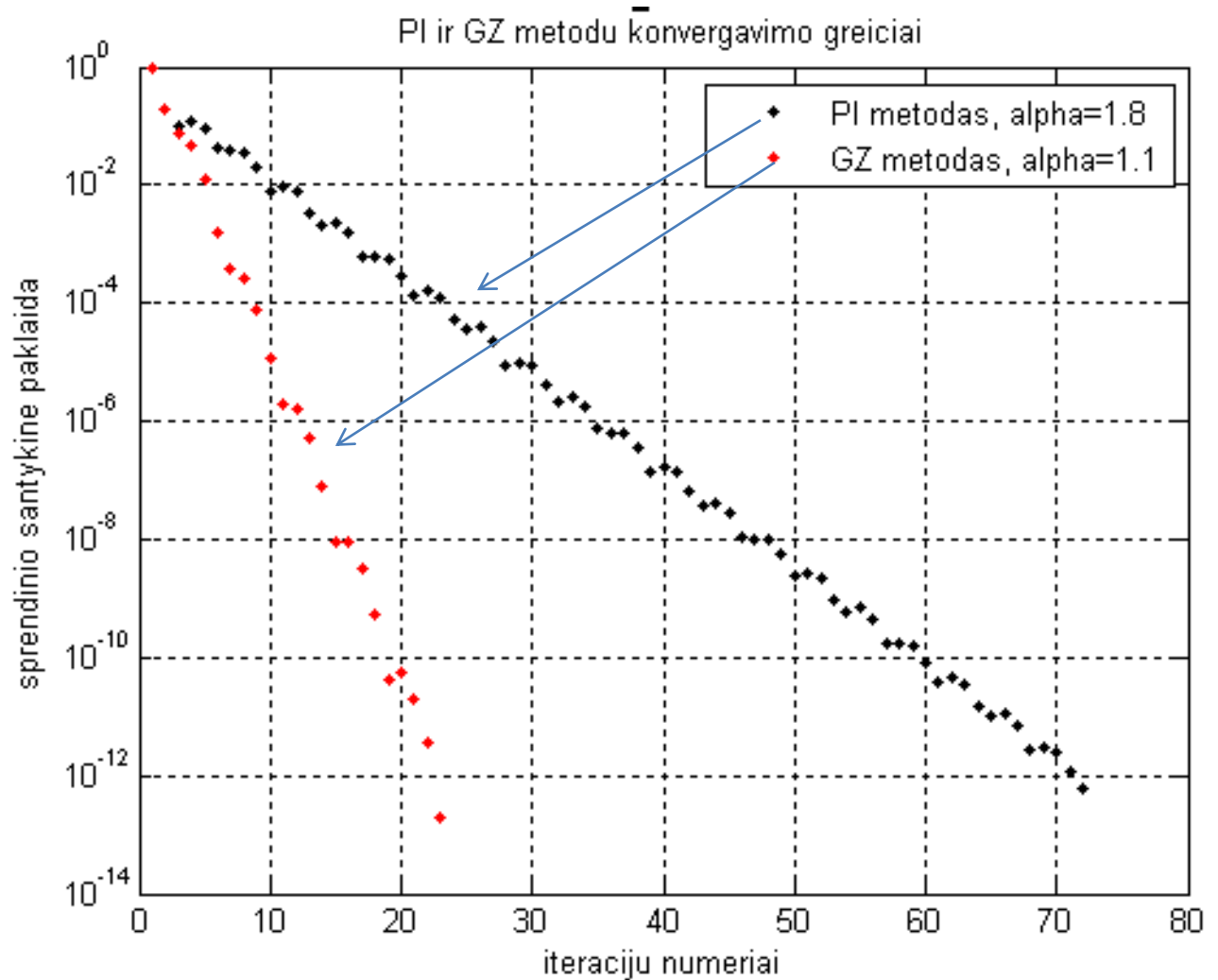
2
0
9
-7

x =

1.0110
0.4857
-0.4571
0.9604

Paprastųjų iteracijų ir Gauso-Zeidelio metodai: konvergavimo sparta

Pvz_SMA_4_2_Paprastuju_iteraciju_ir_Gauso_Zeidelio_algoritmai_su_alpha_ciklu.m



Iteraciniai tiesinių lygčių sistemų sprendimo metodai – apibendrinimas (1)

- Svarbiausias iteracinių metodų privalumas, kad vykdant algoritmą koeficientų matrica nekinta. Tai patogiu, kai matrica saugoma retuoju pavidalu;
- Iteracinių metodų TLS spręsti yra ir daugiau: *jungtinių gradientų (funkcija cgs), mažiausių kvadratų (funkcija lsqr)* ir kt. Jie paremti funkcijos minimizavimo algoritmais. Dėl ribotos kurso apimties jų nenagrinėsime

Iteraciniai tiesinių lygčių sistemų sprendimo metodai – apibendrinimas (2)

- Iteraciniais metodais sprendžiant *singuliarią TLS*, kartais sprendinys gali konverguoti į vieną iš be galo daugelio sprendinių. Tačiau negalėsime padaryti išvados, ar sprendinių be galo daug, ar šis sprendinys vienintelis;
- Jeigu sprendinys nekonverguoja, gali būti, kad sistema neturi sprendinių. Tačiau taip pat gali būti, kad nekonverguoja tik dėl nesėkmingai parinktų *alfa* koeficientų, nors TLS turi vienintelį sprendinį;
- Dažniausiai iteraciniais metodais sprendžiame nesusinguliaras sistemas, kurios turi vienintelį sprendinį

**TLS laisvųjų narių vektoriaus paklaidos įtaka
sprendinio paklaidai. Matricos sąlygotumo
skaičius**

Laisvųjų narių vektoriaus paklaidos įtaka sprendinio paklaidai

$$\mathbf{A}(\mathbf{x} + \Delta \mathbf{x}) = \mathbf{b} + \Delta \mathbf{b}$$

$$\mathbf{A} \cdot \Delta \mathbf{x} = \Delta \mathbf{b}$$

- Nepalanku, jeigu mažos laisvųjų narių vektoriaus paklaidos galėtų sukurti ženklias sprendinio paklaidas;
- Nustatysime skaliarinę įvertį, kaip laisvųjų narių vektoriaus paklaidos įtakoja sprendinio paklaidas;
- Panaudosime *matricos ir vektoriaus normų sąvokas*

Vektorių ir matricų normos

Pagal apibrėžimą, *norma yra matricą ar vektorių apibūdinantis neneigiamas skaičius*, tenkinantis tokias sąlygas:

$$\|\mathbf{A}\| \geq 0;$$

$$\|\alpha \mathbf{A}\| = \alpha \|\mathbf{A}\|;$$

$$\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|;$$

$$\|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|;$$

Naudosime Euklido normas:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2}; \quad \|\{\mathbf{x}\}\| = \sqrt{\sum_{i=1}^n x_i^2}$$

MATLAB-e: `norm(A)`

`norm(x)`

Python-e: `np.linalg.norm(A)`

`np.linalg.norm(x)`

Laisvųjų narių ir sprendinio santykinių paklaidų tarpusavio priklausomybė

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

$$\mathbf{A} (\mathbf{x} + \Delta \mathbf{x}) = \mathbf{b} + \Delta \mathbf{b}$$

$$\mathbf{A} \cdot \Delta \mathbf{x} = \Delta \mathbf{b}$$

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

$$\|\Delta \mathbf{x}\| = \|\mathbf{A}^{-1} \Delta \mathbf{b}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\Delta \mathbf{b}\|$$

$$\|\mathbf{b}\| = \|\mathbf{A} \mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$$

$$\|\Delta \mathbf{x}\| \cdot \|\mathbf{b}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \|\Delta \mathbf{b}\| \cdot \|\mathbf{x}\|$$

$$\frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \frac{\|\Delta \mathbf{b}\|}{\|\mathbf{b}\|}$$

← Santykinės paklaidos

Matricos sąlygotumo skaičius. Gerai, jeigu artimas 1; blogai, jeigu labai didelis

Matricos sąlygotumo skaičiaus apskaičiavimas

$\|[\mathbf{A}]\| \cdot \|[\mathbf{A}]^{-1}\|$ formulę taikyti nėra racionalu dėl didelio veiksmų skaičiaus;

Greičiau skaičiuojama, matricos sąlygotumo skaičių išreiškiant per matricos tikrines reikšmes:

$$\det([\mathbf{A}] - \sigma[\mathbf{E}]) = 0 \quad \Rightarrow \quad \text{Matricos tikrinės reikšmės} \quad \sigma_1, \sigma_2, \dots, \sigma_n$$

$$\|[\mathbf{A}]\| \cdot \|[\mathbf{A}]^{-1}\| = \frac{\max_{1 \leq i \leq n} \sigma_i}{\min_{1 \leq i \leq n} \sigma_i}$$

MATLAB: $\text{norm}(\mathbf{A}) * \text{norm}(\text{inv}(\mathbf{A}))$ $\text{cond}(\mathbf{A})$

Python: $\text{Ainv} = \text{np.linalg.inv}(\mathbf{A})$
 $\text{np.linalg.norm}(\mathbf{A}) * \text{np.linalg.norm}(\text{Ainv})$ $\text{np.linalg.cond}(\mathbf{A})$

Matricų sąlygotumo skaičių pavyzdžiai

Pvz_SMA_4_3_Jautrumas_paklaidoms.m

CondNumb=cond(A)

$$\frac{\|\Delta \{\mathbf{x}\}\|}{\|\{\mathbf{x}\}\|} \leq \|[\mathbf{A}]\| \cdot \|[\mathbf{A}]^{-1}\| \frac{\|\Delta \{\mathbf{b}\}\|}{\|\{\mathbf{b}\}\|}$$

A =

5	0	0
0	5	0
0	0	5

CondNumb = 1

A =

1	2	-3
-4	5	6
7	-8	9

CondNumb = 6.9511

A =

1.0000	2.0000	3.0100
1.0000	2.0000	3.0000
1.0100	2.0000	3.0000

CondNumb = 1.9890e+003

A =

1	0	0
0	5	0
0	0	9

CondNumb = 9

A =

11	-5	12
6	-2	10
0	13	29

CondNumb = 70.7940

A =

1.0000	2.0000	3.0001
1.0000	2.0000	3.0000
1.0001	2.0000	3.0000

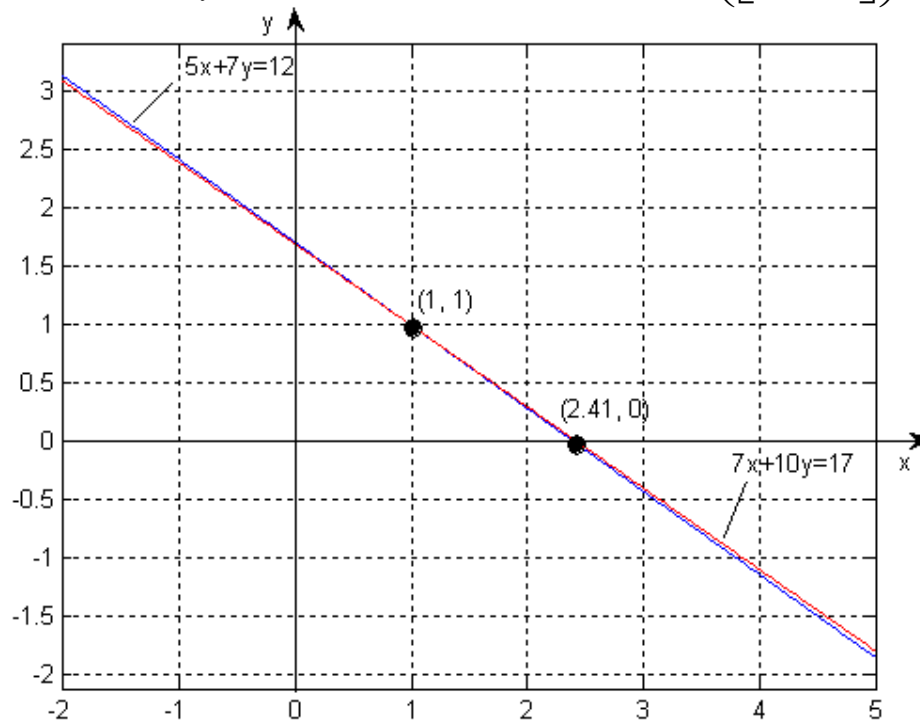
CondNumb = 1.9850e+005

$$5x + 7y = 12;$$

$$7x + 10y = 17$$



$$\text{cond}\left(\begin{bmatrix} 5 & 7 \\ 7 & 10 \end{bmatrix}\right) \approx 223$$



$$x=1; y=1 \Rightarrow \begin{cases} 5 + 7 = 12 \\ 7 + 10 = 17 \end{cases}$$

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

- Lygčių sistemą apytiksliai tenkina ir kitos skaičių poros, gana skirtingos nuo tikrojo sprendinio;
- Priežastis yra didelė matricos sąlygotumo skaičiaus reikšmė. Ši situacija žodžiais įvardinama įvairiai : *matrix close to singular*; *ill-conditioned*; *poorly(badly) defined*, ir pan.

SMA_04_Klausimai savikontrolei:

1. Koks yra esminis tiesioginių ir iteracinių tiesinėms algebrinių lygčių sistemoms taikomų metodų skirtumas;
Kaip reikia pertvarkyti lygčių sistemos koeficientų matricą, prieš pradedant taikyti Paprastųjų iteracijų algoritmą;
2. Kokį vaidmenį atlieka koeficientai "alpha" Paprastųjų iteracijų algoritme. Ką jie įtakoja?
3. Kas yra sprendimo proceso konvergavimas arba divergavimas. Kokia jų sąsaja su koeficientų "alpha" reikšmėmis;
4. Pagal kokias sąlygas stabdomas sprendimo Paprastųjų iteracijų metodu procesas;
5. Paaiškinkite, kuo Gauso-Zeidelio algoritmas skiriasi nuo Paprastųjų iteracijų algoritmo;
6. Kaip nustatoma lygčių sistemos dešinėsios pusės vektoriaus santykinės paklaidos įtaka gauto sprendinio reikšmei;
7. Kas yra matricos sąlygotumo skaičius. Kaip jį apskaičiuoti, taikant MATLAB arba Python funkcijas