Vysoká škola báňská – Technická univerzita Ostrava
Fakulta stavební

Přednáška z předmětu: Algoritmizace inženýrských výpočtů

Téma č.5: Soustavy lineárních rovnic

prof. Ing. Martin Krejsa, Ph.D.

Obsah

Kapitola 1

Soustavy lineárních rovnic

Cíle

Kapitola je zaměřena na problematiku:

- algoritmů řešení soustav lineárních rovnic přímými metodami,
- iteračních metod řešení soustav lineárních rovnic,
- trojných cyklů typu `for`,
- maticového počtu.

Velké množství úloh stavební mechaniky vede k řešení soustavy lineárních rovnic. Numerické metody pro jejich řešení jsou tedy velmi častým programátorským úkolem.

Obsah
3. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Obecně může být soustava n lineárních rovnic s n proměnnými zapsána jako:

$$\begin{array}{ccccccccc} a_{1,1}\cdot x_1 & + & a_{1,2}\cdot x_2 & + & \cdots & + & a_{1,n}\cdot x_n & = & b_1 \\ a_{2,1}\cdot x_1 & + & a_{2,2}\cdot x_2 & + & \cdots & + & a_{2,n}\cdot x_n & = & b_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{n,1}\cdot x_1 & + & a_{n,2}\cdot x_2 & + & \cdots & + & a_{n,n}\cdot x_n & = & b_n \end{array} , \tag{1.1}$$

kde proměnné $x_1, \ldots, x_n$, obecně x_i pro $i = 1, \ldots, n$, jsou neznámé, $a_{i,j}$ pro $i, j = 1, \ldots, n$ jsou koeficienty soustavy rovnic a čísla b_i pro $i = 1, \ldots, n$, jsou absolutní členy soustavy (nebo také tzv. pravá strana soustavy).

K řešení kořenů soustav lineárních rovnic se využívá maticového počtu. Koeficienty soustavy lze zapsat ve tvaru matice:

$$[A] = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} , \tag{1.2}$$

která bývá označována jako matice soustavy.

Neznámé a pravou stranu soustavy je možno vyjádřit jako vektory:

$$\{x\} = \begin{Bmatrix} x_1 & x_2 & \cdots & x_n \end{Bmatrix}^T , \tag{1.3}$$

$$\{b\} = \begin{Bmatrix} b_1 & b_2 & \cdots & b_n \end{Bmatrix}^T . \tag{1.4}$$

Obsah
4. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Celou soustavu lineárních rovnic pak lze vyjádřit maticově:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}, \tag{1.5}$$

nebo zkráceně v maticovém tvaru:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}\,, \tag{1.6}$$

kde **A** značí matici soustavy (levých stran rovnic), **x** sloupcový vektor neznámých kořenů soustavy a **b** sloupcový vektor pravých stran rovnic.

Jednou z podmínek jednoznačného řešení soustavy lineárních rovnic je skutečnost, že matice **A** musí být **regulární**.

Poznámka 1.1. Regulární matice je čtvercová matice, jejíž determinant je různý od nuly. Opakem regulární matice je tzv. **singulární matice** s nulovým determinantem. Důležitou vlastností regulární matice je možnost vypočítat jednoznačně inverzní matici. Toho lze využít např při řešení soustavy lineárních rovnic.

1.1. Přímé metody řešení soustav lineárních rovnic

Metody pro řešení soustav lineárních rovnic, které vedou k přesnému řešení (pokud se neberou v úvahu chyby numerického řešení) při konečném počtu výpočetních kroků, se označují jako *metody přímé*. Jejich základním rysem je eliminace neznámých. Pro plné matice bývají tyto metody nejefektivnější, při velkém počtu rovnic však může být výpočet omezen pamětí počítače.

1.1.1. Řešení trojúhelníkové soustavy lineárních rovnic

Obecnou horní trojúhelníkovou soustavu lineárních rovnic lze zapsat ve tvaru:

$$\begin{array}{ccccccccc} a_{1,1}\cdot x_1 & + & a_{1,2}\cdot x_2 & + & \cdots & + & a_{1,n}\cdot x_n & = & b_1 \\ & & a_{2,2}\cdot x_2 & + & \cdots & + & a_{2,n}\cdot x_n & = & b_2 \\ & & & & \ddots & & \vdots & & \vdots \\ & & & & & & a_{n,n}\cdot x_n & = & b_n \end{array}, \tag{1.7}$$

nebo maticově

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ & a_{2,2} & \cdots & a_{2,n} \\ & & \ddots & \vdots \\ & & & a_{n,n} \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}, \tag{1.8}$$

Řešení, označované jako zpětná substituce (zpětný chod), je možno popsat algoritmem 1.

Výpočet horní trojúhelníkové soustavy lineárních rovnic lze v programu MATLAB naprogramovat například takto:

```
n=input('Zadejte pocet neznamych v soustave rovnic:\n n=');
A=zeros(n,n);
fprintf('\n Zadejte prvky matice soustavy A:');
for i=1:n
  for j=i:n
    fprintf('\n A[%d,%d]=',i,j)
```

Obsah
6. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

for $i \leftarrow n, n-1, \ldots, 2, 1$ **do**

$$x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j}{a_{i,i}}$$

end

Algoritmus 1: Algoritmus zpětné substituce

```
    A(i,j)=input('');
  end
end
if det(A)==0
  error('Soustava rovnic je singulární! Det(A) je roven 0!')
end
fprintf('\n Zadejte prvky vektoru pravych stran b:');
for i=1:n
    fprintf('\n b[%d]=',i)
    b(i)=input('');
end
if n==1
  x(1)=b(1)/A(1,1);
else
  for i=n:-1:1
```

Obsah
7. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
    s=0;
    if i<n
      for j=i+1:n
        s=s+A(i,j)*x(j);
      end
    end
    x(i)=(b(i)-s)/A(i,i);
  end
end
fprintf('\n')
disp('Kořeny soustavy')
disp('--------------------------------')
for i=1:n
  fprintf('x[%d]=%16.8f\n',i,x(i))
end
```

Poznámka 1.2. V ukázce je pro zadání vstupních údajů použito příkazu `input`, který umožňuje výpis popisu zadávané veličiny na obrazovku a přiřazení hodnoty do proměnné zadáním přímo z klávesnice.

Příklad 1.3. Určete kořeny trojúhelníkové soustavy lineárních rovnic řádu 4:

$$\begin{array}{rcrcrcrcl} x_1 & + & 2\cdot x_2 & + & 3\cdot x_3 & + & 4\cdot x_4 & = & 2 \\ & & 2\cdot x_2 & + & 6\cdot x_3 & + & 12\cdot x_4 & = & 8 \\ & & & & 6\cdot x_3 & + & 24\cdot x_4 & = & 18 \\ & & & & & & 24\cdot x_4 & = & 24 \end{array} \tag{1.9}$$

Obsah
8. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Řešení. Výsledný vektor neznámých kořenů má tvar $x = \{-1 \quad 1 \quad -1 \quad 1\}^T$. ▲

Poznámka 1.4. Kontrolu správnosti řešení lze provést např. opětovným dosazením kořenů do jednotlivých rovnic soustavy nebo lépe odečtením levé strany soustavy od pravé, čímž je možno získat tzv. *reziduální vektor* řešení **r**, např. následujícím způsobem:

```
fprintf('\n')
disp('Reziduální vektor')
disp('---------------------------------')
for i=1:n
  r(i)=0;
  for j=i:n
    r(i)=r(i)+A(i,j)*x(j);
  end
  r(i)=r(i)-b(i);
  fprintf('r[%d]=%16.8f\n',i,r(i))
end
```

Jednotlivé prvky by se měly blížit k nule. Pro trojúhelníkovou soustavu vychází reziduální vektor:

```
r[1] = 0.00000000
r[2] = 0.00000000
r[3] = 0.00000000
r[4] = 0.00000000
```

Obsah
9. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Poznámka 1.5. Kontrolu přesnosti řešení lze provést rovněž pomocí *euklidovské normy reziduálního vektoru* $\mathbf{r}$, danou výrazem $\sqrt{\sum_i |r_i|^2}$, kterou lze v programu MATLAB vyvolat např. příkazem `norm(A*x-b)`.

Poznámka 1.6. Při realizaci algoritmu vzniknou potíže, pokud čísla na diagonále, tj. $a_{i,i}$, budou malá. Matice soustavy $\mathbf{A}$ pak může být téměř singulární ($\det \mathbf{A} \approx 0$). Řešením je přeuspořádání soustavy lineárních rovnic tak, aby na diagonále byly největší prvky matice soustavy $\mathbf{A}$.

Příklady k procvičení

1. Navrhněte algoritmus pro řešení obecné trojúhelníkové soustavy lineárních rovnic, která má dolní matici soustavy $\mathbf{A}$ (nuly nad diagonálou).

1.1.2. Gaussova eliminační metoda

Gaussova eliminace je jednou z nejstarších numerických metod, při které se matice soustavy $\mathbf{A}$ převádí na horní trojúhelníkovou matici. Řádkovými úpravami s využitím tzv. *multiplikátorů* se tato matice upraví do tvaru, kdy se pod hlavní diagonálou nachází pouze nuly. Upravená matice pak odpovídá soustavě rovnic, která je ekvivalentní s původní soustavou, a lze ji řešit podobně jako trojúhelníkovou soustavu lineárních rovnic s pomocí tzv. zpětné substituce (zpětného chodu). Celý výpočetní postup lze schématicky popsat algoritmem 2.

Samotný výpis m-funkce v programu MATLAB může nabývat tvaru:

Obsah
10. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

for $k \leftarrow 1, 2, \ldots, n-2, n-1$ **do**
  for $i \leftarrow k+1, k+2, \ldots, n-1, n$ **do**
    $m \leftarrow -\frac{a_{i,k}}{a_{k,k}}$
    for $j \leftarrow k, k+1, \ldots, n-1, n$ **do**
      $a_{i,j} \leftarrow a_{i,j} + m \cdot a_{k,j}$
    end
    $b_i \leftarrow b_i + m \cdot b_k$
  end
end
for $i \leftarrow n, n-1, \ldots, 2, 1$ **do**

$$x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j}{a_{i,i}}$$

end

Algoritmus 2: Algoritmus Gaussovy eliminace

```
function x=gauss(A,b)
if det(A)==0
  error('Soustava rovnic je singulární! Det(A) je roven 0!')
  return
end
n=length(A);
```

Obsah
11. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
if n==1
  x(1)=b(1)/A(1,1);
  return
end
for k=1:n-1
  if A(k,k)==0
    error('Na diagonále je nula!')
    return
  end
  for i=k+1:n
    m=-A(i,k)/A(k,k);
    for j=k:n
      A(i,j)=A(i,j)+m*A(k,j);
    end;
    b(i)=b(i)+m*b(k);
  end
end
for i=n:-1:1
  s=0;
  if i<n
    for j=i+1:n
      s=s+A(i,j)*x(j);
    end
  end
  x(i)=(b(i)-s)/A(i,i);
```

Obsah
12. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
end
```

Příklad 1.7. S využitím vytvořené m-funkce vyřešte kořeny soustavy 4 lineárních rovnic:

$$\begin{array}{rcrcrcrcr} 2\cdot x_1 & - & x_2 & + & 3\cdot x_3 & - & x_4 & = & 7 \\ x_1 & - & x_2 & + & 4\cdot x_3 & - & 2\cdot x_4 & = & 5 \\ 3\cdot x_1 & + & 2\cdot x_2 & + & x_3 & + & 4\cdot x_4 & = & 31 \\ 4\cdot x_1 & - & 3\cdot x_2 & + & 3\cdot x_3 & - & 3\cdot x_4 & = & -5 \end{array} \tag{1.10}$$

Řešení. Řešením je vektor neznámých kořenů $\{x\} = \{1 \quad 2 \quad 4 \quad 5\}^T$. V tomto příkladě lze demonstrovat chod algoritmu Gaussovy eliminace:

```
Původní matice A              Původní vektor b
----------------------------  ----------------------------
  2.000 -1.000  3.000 -1.000    7.000
  1.000 -1.000  4.000 -2.000    5.000
  3.000  2.000  1.000  4.000   31.000
  4.000 -3.000  3.000 -3.000   -5.000

Upravená matice A             Upravený vektor b
----------------------------  ----------------------------
  2.000 -1.000  3.000 -1.000    7.000
  0.000 -0.500  2.500 -1.500    1.500
  0.000  0.000 14.000 -5.000   31.000
  0.000  0.000  0.000 -0.857   -4.286
```

Obsah
13. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
Kořeny soustavy
-------------------
x[ 1] =    1.000
x[ 2] =    2.000
x[ 3] =    4.000
x[ 4] =    5.000

Reziduální vektor
-------------------
r[ 1] = 0.000e+000
r[ 2] = 0.000e+000
r[ 3] = -7.105e-015
r[ 4] = -1.776e-015
```

▲

Příklad 1.8. Určete kořeny soustavy 3 lineárních rovnic s maticí soustavy

$$[A] = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.11}$$

a vektorem pravých stran:

$$\{b\} = \begin{Bmatrix} 1 & 4 & 1 \end{Bmatrix}^T . \tag{1.12}$$

Řešení. Vektor neznámých kořenů je roven $\{x\} = \begin{Bmatrix} 3 & -5 & 3 \end{Bmatrix}^T$. ▲

Obsah
14. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.9. Vyřešte soustavu 4 lineárních rovnic, danou maticí soustavy

$$[A] = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \tag{1.13}$$

a vektorem pravých stran:

$$\{b\} = \begin{Bmatrix} 1 & 2 & 0 & 1 \end{Bmatrix}^T . \tag{1.14}$$

Řešení. Výsledný vektor neznámých kořenů je $\{x\} = \begin{Bmatrix} 0.5 & 0.75 & 0.25 & 0.5 \end{Bmatrix}^T$. ▲

Příklad 1.10. Stanovte strojový čas a přesnost řešení (normu reziduálního vektoru) náhodně vygenerované soustavy 600 lineárních rovnic.

Řešení. Příklad lze vyřešit např. s využitím následujícího sledu příkazů:

```
clc;
clear;
n=600;
m=1200;
A=randn(n,m);
A=A*A';
b=randn(n,1);
tic, x=gauss(A,b); toc
norm(A*x'-b)
```

▲

Obsah
15. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.11. Vyřešte reakce a vnitřní síly u příhradové konstrukce z obr. 1.1 pomocí obecné styčníkové metody. Vstupní parametry jsou $b = 3$ m, $h = 1,5$ m, $F_1 = 5$ kN a $F_2 = 12$ kN. Soustavu lineárních rovnic pak vyřešte Gaussovou eliminací.

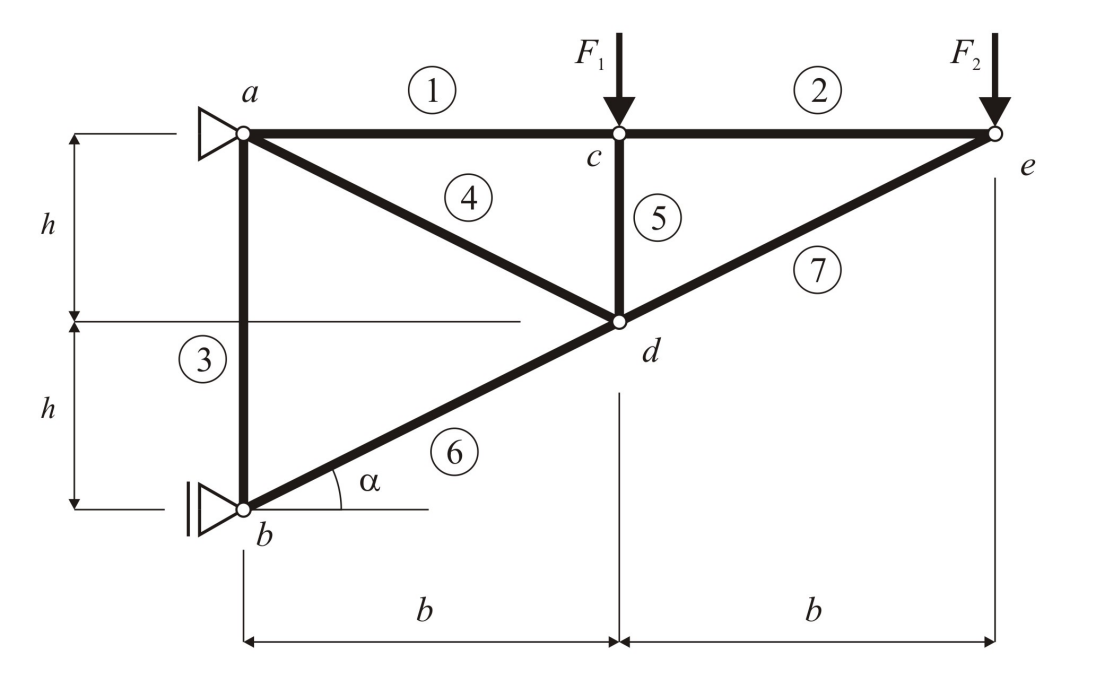
Řešení. Pokud se příhradová konstrukce z obr. 1.1 interpretuje jako soustava hmotných bodů, z kinematického hlediska obsahuje $2 \cdot s$ stupňů volnosti, kde s je počet hmotných bodů, tedy styčníků. Jelikož je konstrukce tvořena 5 styčníky ($s = 5$), vychází pak $n_v = 2 \cdot s = 10$ stupňů volnosti, které jsou odebrány 3 vnějšími vazbami ($v_\mathrm{e} = 3$) a 7 vnitřními ($v_i = 7$) (počet prutů). Vzhledem ke skutečnosti, že $n_v = v_\mathrm{e} + v_i$, jedná se o konstrukci staticky i kinematicky určitou.

Pokud se v každém ze styčníků stanoví 2 podmínky rovnováhy, lze získat celkem 10 podmínek rovnováhy, tvořících soustavu lineárních rovnic, ze kterých lze stanovit 10 neznámých - 3 reakce ($R_{ax}, R_{az}aR_{bx}$) a 7 vnitřních sil ($N_1, N_1, \ldots, N_7$).

Jednotlivé podmínky rovnováhy nabývají tvaru:

- Styčník a:
  1. $R_x = 0 : -R_{ax} + N_1 + N_4 \cdot \cos(\alpha) = 0$
  2. $R_z = 0 : -R_{az} + N_3 + N_4 \cdot \sin(\alpha) = 0$
- Styčník b:
  3. $R_x = 0 : +R_{bx} + N_6 \cdot \cos(\alpha) = 0$
  4. $R_z = 0 : -N_3 - N_6 \cdot \sin(\alpha) = 0$
- Styčník c:
  5. $R_x = 0 : -N_1 + -N_2 = 0$
  6. $R_z = 0 : +F_1 + N_5 = 0$

Obsah
16. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno



Obr. 1.1 Statické schéma řešené příhradové konstrukce

- Styčník d:

  7. $R_x = 0 : -N_4 \cdot \cos(\alpha) - N_6 \cdot \cos(\alpha) + N_7 \cdot \cos(\alpha) = 0$

Obsah
17. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

8. $R_z = 0 : -N_4 \cdot \sin(\alpha) - N_5 + N_6 \cdot \sin(\alpha) - N_7 \cdot \sin(\alpha) = 0$

- Styčník e:

9. $R_x = 0 : -N_2 - N_7 \cdot \cos(\alpha) = 0$
10. $R_z = 0 : +F_2 + N_7 \cdot \sin(\alpha) = 0$

Celou soustavu lineárních rovnic řádu 10 lze přehledně zapsat maticově:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}\,, \tag{1.15}$$

kde $\mathbf{A}$ značí matici levých stran rovnic, obsahující údaje týkající se geometrie konstrukce:

$$\begin{bmatrix}
-1 & 0 & 0 & +1 & 0 & 0 & +\cos(\alpha) & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & +1 & +\sin(\alpha) & 0 & 0 & 0 \\
0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & +\cos(\alpha) & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -\sin(\alpha) & 0 \\
0 & 0 & 0 & -1 & +1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\cos(\alpha) & 0 & -\cos(\alpha) & +\cos(\alpha) \\
0 & 0 & 0 & 0 & 0 & 0 & -\sin(\alpha) & -1 & +\sin(\alpha) & -\sin(\alpha) \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -\cos(\alpha) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +\sin(\alpha)
\end{bmatrix}, \tag{1.16}$$

$\mathbf{x}$ představuje sloupcový vektor neznámých kořenů, obsahující 10 neznámých reakcí a vnitřních sil:

$$\begin{Bmatrix} R_{ax} & R_{az} & R_{bz} & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 \end{Bmatrix}^T \tag{1.17}$$

Obsah
18. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

a **b** vyjadřuje sloupcový vektor pravých stran rovnic, obsahující uzlové zatížení příhradové konstrukce:

$$\{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -F_1 \quad 0 \quad 0 \quad 0 \quad -F_2\}^T \ . \tag{1.18}$$

Goniometrické funkce $\cos(\alpha)$ a $\sin(\alpha)$, obsažené v matici **A**, lze vyjádřit přímo z rozměru konstrukce:

$$\cos(\alpha) = \frac{b}{l} \ , \quad \sin(\alpha) = \frac{h}{l} \ , \tag{1.19}$$

kde l je délka prutů č.4, 6 a 7:

$$l = l_4 = l_6 = l_7 = \sqrt{b^2 + h^2} \ . \tag{1.20}$$

Vzhledem k podmínce řešení Gaussovy metody, že prvky na diagonále matice **A** se nesmí rovnat 0, matici **A** i vektor pravých stran **b** je nutno upravit vhodným přeuspořádáním pořadí styčníkových rovnic, a to:

- na 4.řádku bude původně 5. styčníková rovnice,
- na 5.řádku bude původně 9. styčníková rovnice,
- na 6.řádku bude původně 4. styčníková rovnice,
- na 8.řádku bude původně 6. styčníková rovnice,
- na 9.řádku bude původně 8. styčníková rovnice,

Obsah
19. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

takže pozměněná matice levých stran **A** bude mít výsledný tvar:

$$\begin{bmatrix}
-1 & 0 & 0 & +1 & 0 & 0 & +\cos(\alpha) & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & +1 & +\sin(\alpha) & 0 & 0 & 0 \\
0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & +\cos(\alpha) & 0 \\
0 & 0 & 0 & -1 & +1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -\cos(\alpha) \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -\sin(\alpha) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\cos(\alpha) & 0 & -\cos(\alpha) & +\cos(\alpha) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\sin(\alpha) & -1 & +\sin(\alpha) & -\sin(\alpha) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +\sin(\alpha)
\end{bmatrix} , \tag{1.21}$$

a sloupcový vektor pravých stran rovnic **b**:

$$\begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -F_1 & 0 & -F_2 \end{Bmatrix}^T . \tag{1.22}$$

Sloupcový vektor neznámých kořenů **x** zůstane nezměněn.

Takto sestavenou soustavu lineárních rovnic lze vyřešit pro konkrétně zadané vstupní veličiny s pomocí Gaussovy eliminační metody:

```
Kořeny soustavy
------------------
x[ 1] =   29.000 kN
x[ 2] =   17.000 kN
x[ 3] =   29.000 kN
x[ 4] =   24.000 kN
x[ 5] =   24.000 kN
```

Obsah
20. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
x[ 6] =   14.500 kN
x[ 7] =    5.590 kN
x[ 8] =   -5.000 kN
x[ 9] =  -32.423 kN
x[10] =  -26.833 kN

Norma reziduálního vektoru:
---------------------------
n =
     0
```

▲

1.1.3. Gauss-Jordanova metoda

Gaussovu eliminační metodu lze jednoduše modifikovat způsobem, který je označován jako Gauss-Jordanova metoda. Základní myšlenkou tohoto výpočetního postupu je úprava matice soustavy **A** na diagonální nebo dokonce jednotkovou matici. Výpočetní postup je schématicky popsán algoritmem 3 a lze jej implementovat do m-funkce programu MATLAB např. následujícím způsobem:

```
function x=gauss_jordan(A,b)
if det(A)==0
  error('Soustava rovnic je singulární! Det(A) je roven 0!')
  return
```

Obsah
21. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
end
n=length(A);
if n==1
  x(1)=b(1)/A(1,1);
  return
end
for k=1:n
  if A(k,k)==0
    error('Na diagonále je nula!')
    return
  end
  for i=1:n
    if ~(i==k)
      m=-A(i,k)/A(k,k);
      for j=k:n
        A(i,j)=A(i,j)+m*A(k,j);
      end;
      b(i)=b(i)+m*b(k);
    end
  end
end
for i=1:n
  x(i)=b(i)/A(i,i);
end
```

Obsah
22. strana ze 62

Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

for $k \leftarrow 1, 2, \ldots, n-2, n$ **do**
  for $i \leftarrow 1, 2, \ldots, k-1, k+1, \ldots, n$ **do**
    $m \leftarrow -\dfrac{a_{i,k}}{a_{k,k}}$
    for $j \leftarrow 1, 2, \ldots, n-1, n$ **do**
      $a_{i,j} \leftarrow a_{i,j} + m \cdot a_{k,j}$
    end
    $b_i \leftarrow b_i + m \cdot b_k$
  end
end
for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**
  $x_i \leftarrow \dfrac{b_i}{a_{i,i}}$
end

Algoritmus 3: Algoritmus Gauss-Jordanovy metody

Gauss-Jordanovu metodu je možno použít i k řešení tzv. maticových rovnic, jenž lze zkráceně vyjádřit:

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{B}\,, \tag{1.23}$$

kde $\mathbf{X}$ představuje matici kořenů soustavy a $\mathbf{B}$ matici levých stran, obě s obecným rozměrem $[n, r]$. Výpočetní postup je tedy nutno upravit tak, aby pracoval s více pravými stranami, jak je např. schématicky popsánu v upraveném algoritmu 4.

Obsah
23. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{B} = [b_{i,j}] = [b_{1,1}, \ldots, b_{n,r}]$
Výstup: $\mathbf{X} = [x_{i,j}] = [x_{1,1}, \ldots, x_{n,r}]$

for $k \leftarrow 1, 2, \ldots, n-2, n$ **do**
  for $i \leftarrow 1, 2, \ldots, k-1, k+1, \ldots, n$ **do**
    $m \leftarrow -\dfrac{a_{i,k}}{a_{k,k}}$
    for $j \leftarrow 1, 2, \ldots, n-1, n$ **do**
      $a_{i,j} \leftarrow a_{i,j} + m \cdot a_{k,j}$
    end
    for $j \leftarrow 1, 2, \ldots, r-1, r$ **do**
      $b_{i,j} \leftarrow b_{i,j} + m \cdot b_{k,j}$
    end
  end
end
for $j \leftarrow 1, 2, \ldots, r-1, r$ **do**
  for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**
    $x_{i,j} \leftarrow \dfrac{b_{i,j}}{a_{i,i}}$
  end
end

Algoritmus 4: Algoritmus Gauss-Jordanovy metody pro řešení tzv. maticových rovnic

Příklad 1.12. Pomocí Gauss-Jordanovy metody vypočtěte maticovou rovnici:

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 2 & 1 \\ 3 & -1 & 2 \end{bmatrix}^T . \tag{1.24}$$

Obsah
24. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Řešení. Řešením je matice:

$$[X] = \begin{bmatrix} 3 & -5 & 3 \\ 2 & -2 & 1 \\ -2 & 7 & -3 \end{bmatrix}^T . \tag{1.25}$$

▲

Řešení maticových rovnic lze využít např. k řešení **inverzní matice**, kdy je matice pravých stran tvořena jednotkovou diagonální maticí„ nebo pro výpočet příhradové konstrukce z příkladu 1.11 s více zatěžovacími stavy.

Příklad 1.13. Pomocí Gauss-Jordanovy metody vypočtěte inverzní matici k matici:

$$\begin{bmatrix} 1 & 2 & 6 \\ 2 & 5 & 15 \\ 6 & 15 & 46 \end{bmatrix} \tag{1.26}$$

Řešení. Řešením je inverzní matice:

$$[A]^{-1} = \begin{bmatrix} 5 & -2 & 0 \\ -2 & 10 & -3 \\ 0 & -3 & 1 \end{bmatrix} , \tag{1.27}$$

o čemž se lze přesvěčit např. výpisem:

```
A =                    B =                    A*B =
     1     2     6          5    -2     0          1     0     0
     2     5    15         -2    10    -3          0     1     0
     6    15    46          0    -3     1          0     0     1
```

▲

Obsah
25. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklady k procvičení

1. Určete reakce a vnitřní síly příhradové konstrukce z příkladu z příkladu 1.11 rovněž pro zatěžovací stav, tvořený dvojicí svislých uzlových zatížení $F = 10$ kN ve styčnících d a e.

1.1.4. LU rozklad

Princip metody LU rozkladu je založen na principu, kdy regulární matici $\mathbf{A}$ lze rozložit na součin dvou trojúhelníkových matic $\mathbf{L}$ a $\mathbf{U}$ tak, že platí:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \,. \tag{1.28}$$

Soustava lineárních rovnic se pak řeší ve dvou výpočetních krocích, kdy v prvním se vyřeší trojúhelníková soustava lineárních rovnic:

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b} \,, \tag{1.29}$$

obdobně pak

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y} \,. \tag{1.30}$$

V dolní trojúhelníkové matici $\mathbf{L}$ se přitom volí na diagonále hodnota 1. Matice $\mathbf{U}$ je horní trojúhelníková. Výhodou metody LU rozkladu je její použití u úloh s více soustavami lineárních rovnic se stejnou maticí soustavy $\mathbf{A}$, která se rozloží pouze jednou a dále se již opakovaně řeší pouze trojúhelníkové soustavy.

Výpočetní postup řešení soustavy lineárních rovnic LU rozkladem je schématicky vyjádřen v algoritmech 5 a 6.

Obsah
26. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$
Výstup: $\mathbf{U} = [u_{i,j}] = [u_{1,1}, \ldots, u_{n,n}]$, $\mathbf{L} = [l_{i,j}] = [l_{1,1}, \ldots, l_{n,n}]$

$\mathbf{U} \leftarrow 0$
$\mathbf{L} \leftarrow 0$

for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**
  $l_{i,i} \leftarrow 1$
end

for $j \leftarrow 1, 2, \ldots, n-1, n$ **do**
  for $i \leftarrow 1, 2, \ldots, j-1, j$ **do**

$$u_{i,j} \leftarrow a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} \cdot u_{k,j}$$

  end
  for $i \leftarrow j+1, j+2, \ldots, n-1, n$ **do**

$$l_{i,j} \leftarrow \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} \cdot u_{k,j}}{u_{j,j}}$$

  end
end

Algoritmus 5: Algoritmus metody LU rozkladu

Obsah
27. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$, $\mathbf{U} = [u_{i,j}] = [u_{1,1}, \ldots, u_{n,n}]$,
$\mathbf{L} = [l_{i,j}] = [l_{1,1}, \ldots, l_{n,n}]$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$y_i \leftarrow b_i - \sum_{j=1}^{i-1} l_{i,j} \cdot y_j$$

end

for $i \leftarrow n, n-1, \ldots, 2, 1$ **do**

$$x_i \leftarrow \frac{y_i - \sum_{j=i+1}^{n} u_{i,j} \cdot x_j}{u_{i,i}}$$

end

Algoritmus 6: Algoritmus zpětného chodu metody LU rozkladu

Výpočetní algoritmus metody LU rozkladu je možno implementovat do m-funkce programu MATLAB např. následujícím způsobem:

```
function x=lu_rozklad(A,b)
if det(A)==0
  error('Soustava rovnic je singulární! Det(A) je roven 0!')
  return
end
n=length(A);
if n==1
  x(1)=b(1)/A(1,1);
```

Obsah
28. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
  return
end
L=eye(n,n);
U=zeros(n,n);
for j=1:n
  for i=1:j
    s=0;
      if i>1
      for k=1:i-1
        s=s+L(i,k)*U(k,j);
      end
    end
    U(i,j)=A(i,j)-s;
  end
  if U(j,j)==0
    error('Na diagonále je nula!')
    return
  end
  for i=j+1:n
    s=0;
    if j>1
      for k=1:j-1
        s=s+L(i,k)*U(k,j);
      end
    end
```

Obsah

29. strana ze 62

Zavřít dokument

Konec

Celá obrazovka/Okno

```
    L(i,j)=(A(i,j)-s)/U(j,j);
  end
end
for i=1:n
  s=0;
  if i>1
    for j=1:i-1
      s=s+L(i,j)*y(j);
    end
  end
  y(i)=(b(i)-s);
end
for i=n:-1:1
  s=0;
  if i<n
    for j=i+1:n
      s=s+U(i,j)*x(j);
    end
  end
  if U(i,i)==0
    error('Na diagonále je nula!')
    return
  end
  x(i)=(y(i)-s)/U(i,i);
end
```

Obsah
30. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.14. Pomocí metody LU rozkladu vypočtěte kořeny soustavy lineárních rovnic z příkladu 1.7.

Řešení. Řešení lze demonstrovat podrobným výpisem průběžných i konečných výsledků:

```
Matice soustavy A              Vektor pravých stran b
---------------------------    ---------------------------
  2.000 -1.000  3.000 -1.000     7.000
  1.000 -1.000  4.000 -2.000     5.000
  3.000  2.000  1.000  4.000    31.000
  4.000 -3.000  3.000 -3.000    -5.000


Matice L                       Matice U
---------------------------    ---------------------------
  1.000  0.000  0.000  0.000     2.000 -1.000  3.000 -1.000
  0.500  1.000  0.000  0.000     0.000 -0.500  2.500 -1.500
  1.500 -7.000  1.000  0.000     0.000  0.000 14.000 -5.000
  2.000  2.000 -0.571  1.000     0.000  0.000  0.000 -0.857

Matice L*U
---------------------------
  2.000 -1.000  3.000 -1.000
  1.000 -1.000  4.000 -2.000
  3.000  2.000  1.000  4.000
  4.000 -3.000  3.000 -3.000
```

Obsah
31. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
Kořeny soustavy
---------------------------
x[ 1] = 1.000e+000
x[ 2] = 2.000e+000
x[ 3] = 4.000e+000
x[ 4] = 5.000e+000

Norma reziduálního vektoru:
---------------------------
n = 7.324106877635580e-015
```

▲

1.1.5. Choleského metoda (dekompozice)

Choleského metoda (také Choleského dekompozice nebo rozklad) je modifikace metody LU rozkladu pro řešení soustav lineárních rovnic se symetrickou regulární pozitivně definitní čtvercovou maticí soustavy $\mathbf{A}$, která se upraví na součin dolní a horní trojúhelníkové matice, přičemž jedna trojúhelníková matice je transpozicí matice druhé:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{U}^T \,. \tag{1.31}$$

Dolní trojúhelníková matice $\mathbf{U}$ z tohoto rozkladu se nazývá Choleského trojúhelník matice $\mathbf{A}$.

Poznámka 1.15. Pozitivně definitní matice je symetrická čtvercová matice, jejíž vlastní čísla jsou větší než nula.

Obsah
32. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Výpočetní postup řešení soustavy lineárních rovnic Choleského metodou obsahuje v porovnání s metodou LU rozkladu zhruba poloviční počet výpočetních operací a je schématicky popsán algoritmy 7 a 8.

Vstup : n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$
Výstup: $\mathbf{U} = [u_{i,j}] = [u_{1,1}, \ldots, u_{n,n}]$, $\mathbf{L} = [l_{i,j}] = [l_{1,1}, \ldots, l_{n,n}]$

$\mathbf{U} \leftarrow 0$

for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**
  $u_{i,i} \leftarrow 1$
end

for $j \leftarrow 1, 2, \ldots, n-1, n$ **do**
  for $i \leftarrow j, j+1, \ldots, n-1, n$ **do**

$$u_{j,j} \leftarrow \sqrt{a_{j,j} - \sum_{k=1}^{j-1} u_{j,k}^2}$$

$$u_{i,j} \leftarrow \frac{a_{i,j} - \sum_{k=1}^{j-1} u_{i,k} \cdot u_{j,k}}{u_{j,j}}$$

  end
end

Algoritmus 7: Algoritmus Choleského metody

Obsah
33. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$, $\mathbf{U} = [u_{i,j}] = [u_{1,1}, \ldots, u_{n,n}]$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$y_i \leftarrow \frac{b_i - \sum_{j=1}^{i-1} u_{i,j} \cdot y_j}{u_{i,i}}$$

end
for $i \leftarrow n, n-1, \ldots, 2, 1$ **do**

$$x_i \leftarrow \frac{y_i - \sum_{j=1}^{i-1} u_{j,i} \cdot x_j}{u_{i,i}}$$

end

Algoritmus 8: Algoritmus zpětného chodu Choleského metody

Sled příkazů m-funkce programu MATLAB, implementující postup Choleského metody, může vypadat následujícím způsobem:

```
function x=choleskeho_met(A,b)
n=length(A);
U=eye(n,n);
for j=1:n
  if U(j,j)==0
    error('Na diagonále je nula!')
    return
  end
```

Obsah
34. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
  for i=j:n
    s=0;
    if i>1
      for k=1:j-1
        s=s+U(j,k)^2;
      end
    end
    U(j,j)=sqrt(A(j,j)-s);
    s=0;
    if i>1
      for k=1:j-1
        s=s+U(i,k)*U(j,k);
      end
    end
    U(i,j)=(A(i,j)-s)/U(j,j);
  end
end
for i=1:n
  s=0;
  if i>1
    for j=1:i-1
      s=s+U(i,j)*y(j);
    end
  end
  y(i)=(b(i)-s)/U(i,i);
```

Obsah
35. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
end
for i=n:-1:1
  s=0;
  if i<n
    for j=i+1:n
      s=s+U(j,i)*x(j);
    end
  end
  x(i)=(y(i)-s)/U(i,i);
end
```

Příklad 1.16. Proveďte Choleského dekompozici matice:

$$[A] = \begin{bmatrix} 1 & -1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 3 & 1 & 1 \\ -1 & 0 & 1 & 4 & 2 \\ -1 & 0 & 1 & 2 & 5 \end{bmatrix} . \tag{1.32}$$

Řešení. Řešením je matice:

$$[U] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} , \tag{1.33}$$

Obsah
36. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

o čemž se lze přesvědčit:

```
A =                                    U =
     1    -1    -1    -1    -1              1     0     0     0     0
    -1     2     0     0     0             -1     1     0     0     0
    -1     0     3     1     1             -1    -1     1     0     0
    -1     0     1     4     2             -1    -1    -1     1     0
    -1     0     1     2     5             -1    -1    -1    -1     1


U*U' =
     1    -1    -1    -1    -1
    -1     2     0     0     0
    -1     0     3     1     1
    -1     0     1     4     2
    -1     0     1     2     5
```

▲

Příklad 1.17. Vypočtěte kořeny soustavy rovnic z příkladu 1.9 Choleského metodou.

Obsah
37. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2. Iterační metody řešení soustav lineárních rovnic

Řešení soustav lineárních rovnic iteračními metodami spočívá narozdíl od přímých způsobů výpočtů v postupném přibližování se k přesnému výsledku. Existuje množství způsobů tvorby takové posloupnosti aproximací, jejíž limitou je vektor $\mathbf{x}$ přesně určených kořenů soustavy lineárních rovnic.

Soustava lineárních rovnic s reálnou maticí soustavy, vyjádřena maticově, může být vyjádřena např. vztahem (1.6). Lze předpokládat, že existuje jediné přesné řešení:

$$\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b} \,. \tag{1.34}$$

Rovnici (1.6) pak lze upravit do tvaru, který je vhodný k iterování:

$$\mathbf{x} = \mathbf{H} \cdot \mathbf{x} + \mathbf{g} \,, \tag{1.35}$$

kde $\mathbf{H}$ představuje *iterační matici*. Na vztahu (1.35) je založeno sestrojení posloupnosti iterací $\mathbf{x}^{(k)} = \{x_1^k, x_2^k, \ldots, x_n^k\}^T$ podle rekurentní formule:

$$\mathbf{x}^{(k+1)} = \mathbf{H} \cdot \mathbf{x}^{(k)} + \mathbf{g} \,, \tag{1.36}$$

pro iterační kroky $k = 0, 1, 2, \ldots$.

Kromě iterační formule je nutno definovat také volbu nulté aproximace $\mathbf{x}^{(0)} = \{x_1^0, x_2^0$ až $x_n^0\}^T$ a způsob ukončení iteračního cyklu, který lze provést:

(a) stanovením přesného počtu iteračních cyklů, které se mají provést (např. s využitím cyklu typu `for`),

(b) zastavovací podmínkou se zadanou tolerancí nepřesnosti $\varepsilon > 0$, např. s využitím vektorové normy:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| < \varepsilon \,. \tag{1.37}$$

Obsah
38. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Iterační matice **H** i vektor **g** se může při každém iteračním kroku k měnit. Hovoří se pak o tzv. *nestacionárním iteračním procesu*, narozdíl od procesu *stacionárního*, kdy jsou matice **H** i vektor **g** nezávislými na iteračním cyklu k.

Iterační výpočet soustavy lineárních rovnic konverguje k řešení, pokud je matice soustavy **A** *diagonálně dominantní* (převažují hodnoty prvků na diagonále), což lze vyjádřit:

$$|a_{i,i}| > \sum_{j=1}^{i-1} |a_{i,j}| + \sum_{j=i+1}^{n} |a_{i,j}| \,, \quad i = 1, 2, \ldots, n \,. \tag{1.38}$$

Příklad 1.18. Sestrojte funkci pro posouzení, zda-li je matice **A** diagonálně dominantní. Vyzkoušejte podle (1.38) např. na matici:

$$[A] = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} . \tag{1.39}$$

1.2.1. Jacobiho iterace

V případě obecné soustavy lineárních rovnic $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ jsou všechny rovnice obecně vyjádřeny:

$$a_{i,1} \cdot x_1 \;+\; a_{i,2} \cdot x_2 \;+\; \cdots \;+\; a_{i,n} \cdot x_n \;=\; b_i \,, \quad i = 1, 2, \ldots, n \,. \tag{1.40}$$

Obsah
39. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Pokud je $a_{i,i} \neq 0$, lze každou z rovnic upravit:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j}{a_{i,i}}, \quad i = 1, 2, \ldots, n, \tag{1.41}$$

což znamená, že z i-té rovnice lze určit i-tý neznámý kořen soustavy.

Jacobiho iterační rekurentní formule pak má tvar:

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k-1)}}{a_{i,i}}, \quad i = 1, 2, \ldots, n, \tag{1.42}$$

kde k je číslo iteračního cyklu ($k = 1, 2, \ldots, m$).

Výpočetní postup Jacobiho iterační metody pro m iteračních cyklů je schématicky vyjádřen algoritmem 9. Pokud se pro ukončení iteračního výpočtu použije zakončovací podmínka 1.37, algoritmus se nepatrně změní (viz algoritmus 10).

Obsah
40. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : m, n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$,
$\mathbf{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\}^T$
Výstup: $\mathbf{x}^{(m)} = \{x_1^{(m)}, x_2^{(m)}, \ldots, x_n^{(m)}\}^T$

for $k \leftarrow 1, 2, \ldots, m-1, m$ **do**
  for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k-1)}}{a_{i,i}}$$

  end
end

Algoritmus 9: Algoritmus Jacobiho iterační metody

Obsah
41. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : n, ε, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$,
$\mathbf{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\}^T$
Výstup: $\mathbf{x}^{(m)} = \{x_1^{(m)}, x_2^{(m)}, \ldots, x_n^{(m)}\}^T$

$k \leftarrow 1$
for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$x_i^{(1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(0)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(0)}}{a_{i,i}}$$

end
while $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \geqq \varepsilon$ **do**
  $k \leftarrow k + 1$
  for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k-1)}}{a_{i,i}}$$

  end
end

Algoritmus 10: Algoritmus Jacobiho iterace, doplněné o zakončovací podmínku

Obsah
42. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Algoritmus Jacobiho iterace lze do programovacího jazyka programu MATLAB přepsat různými způsoby. První m-funkce obsahuje implementaci výpočetního postupu řešení systému lineárních rovnic, který pracuje Jacobiho metodou s konečným počtem cyklů, tedy s využitím cyklu typu `for`. K zápisu tohoto algoritmu se využívá možnosti maticových operací programového systému MATLAB:

```
function x=jacobi(A,b,x,m)
n=length(A);
d=diag(A);
r=A-diag(d);
for k=1:m
  x=(b-r*x)./d;
end
```

Druhá m-funkce je zapsána obecnějším způsobem. Výpočet je rovněž doplněn kontrolou, zda-li je matice soustavy **A** regulární a diagonálně dominantní:

```
function x=jacobi(A,b,x,m)
if det(A)==0
  error('Soustava rovnic je singulární! Det(A) je roven 0!')
  return
end
n=length(A);
for i=1:n
  if A(i,i)==0
    error('Na diagonále je nula!')
```

Obsah

43. strana ze 62

Zavřít dokument

Konec

Celá obrazovka/Okno

```
    return
  end
  s=0;
  for j=1:n
    if ~(i==j)
      s=s+abs(A(i,j));
    end
  end
  if abs(A(i,i))<s
    disp('Matice A není diagonálně dominantní!')
    return
  end
end
for k=1:m
  y=x;
  for i=1:n
    s1=0;
    if i>1
      for j=1:i-1
        s1=s1+A(i,j)*y(j);
      end
    end
    s2=0;
    if i<n
      for j=i+1:n
```

Obsah
44. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
        s2=s2+A(i,j)*y(j);
      end
    end
    x(i)=(b(i)-s1-s2)/A(i,i);
  end
end
```

Třetí m-funkce řeší soustavu lineárních rovnic Jacobiho iterací cyklem typu `while`, jenž je ukončen zakončovací podmínkou 1.37:

```
function x=jacobi(A,b,x,eps)
y=x;
k=1;
for i=1:n
  s1=0;
  if i>1
    for j=1:i-1
      s1=s1+A(i,j)*y(j);
    end
  end
  s2=0;
  if i<n
    for j=i+1:n
      s2=s2+A(i,j)*y(j);
    end
```

Obsah
45. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
  end
  if A(i,i)==0
    error('Na diagonále je nula!')
    return
  end
  x(i)=(b(i)-s1-s2)/A(i,i);
end
while ~(norm(y-x)<eps) & k<1000
  y=x;
  k=k+1;
  for i=1:n
    s1=0;
    if i>1
      for j=1:i-1
        s1=s1+A(i,j)*y(j);
      end
    end
    s2=0;
    if i<n
      for j=i+1:n
        s2=s2+A(i,j)*y(j);
      end
    end
    if A(i,i)==0
      error('Na diagonále je nula!')
      return
```

Obsah
46. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
    end
    x(i)=(b(i)-s1-s2)/A(i,i);
  end
end
```

Všechny tři způsoby zápisu předpokládají, že musí být zadána i tzv. nultá aproximace řešených kořenů soustavy $\mathbf{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\}^T$.

Příklad 1.19. Vyřešte kořeny soustavu lineárních rovnic Jacobiho iterací:

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{Bmatrix} . \tag{1.43}$$

Výpočet proveďte nejprve pro konečný počet cyklů $m = 5$, 10 a 20, přičemž sledujte vzrůstající přesnost dosaženého řešení. Nakonec výpočet proveďte m-funkcí se zakončovací podmínkou 1.37 se zadanou tolerancí nepřesnosti $\varepsilon = 1 \cdot 10^{-6}$.

Řešení. Správné řešení kořenů soustavy se rovná vektoru

$$\{x\} = \begin{Bmatrix} 0.5 & 0.75 & 0.25 & 0.5 \end{Bmatrix}^T . \tag{1.44}$$

Samotný průběh iteračního výpočtu může vypadat při zadání $\mathbf{x}^{(0)} = \{1, 1, 1, 1\}^T$ následujícím způsobem:

Obsah
47. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Průběh Jacobiho iterace

c.it.	x[1]	x[2]	x[3]	x[4]
0	1.000000	1.000000	1.000000	1.000000
1	0.750000	1.000000	0.500000	0.750000
2	0.625000	0.875000	0.375000	0.625000
3	0.562500	0.812500	0.312500	0.562500
4	0.531250	0.781250	0.281250	0.531250
5	0.515625	0.765625	0.265625	0.515625
6	0.507813	0.757813	0.257813	0.507813
7	0.503906	0.753906	0.253906	0.503906
8	0.501953	0.751953	0.251953	0.501953
9	0.500977	0.750977	0.250977	0.500977
10	0.500488	0.750488	0.250488	0.500488
11	0.500244	0.750244	0.250244	0.500244
12	0.500122	0.750122	0.250122	0.500122
13	0.500061	0.750061	0.250061	0.500061
14	0.500031	0.750031	0.250031	0.500031
15	0.500015	0.750015	0.250015	0.500015
16	0.500008	0.750008	0.250008	0.500008
17	0.500004	0.750004	0.250004	0.500004
18	0.500002	0.750002	0.250002	0.500002
19	0.500001	0.750001	0.250001	0.500001
20	0.500000	0.750000	0.250000	0.500000

Obsah
48. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

K dosažení výsledku s tolerancí nepřesností $\varepsilon = 1 \cdot 10^{-6}$ je tedy při výpočtu Jacobiho iterací celkem 20-ti iteračních cyklů.

▲

1.2.2. Gauss-Seidelova iterační metoda

Obecnou soustavu lineárních rovnic $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ lze opět vyjádřit vztahem (1.40), přičemž lze každou z rovnic definovat při splnění podmínky $a_{i,i} \neq 0$ jako:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j}{a_{i,i}}, \quad i = 1, 2, \ldots, n\,. \tag{1.45}$$

Z této rovnice lze odvodit Gauss-Seidelovu iterační rekurentní formuli:

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k-1)}}{a_{i,i}}, \quad i = 1, 2, \ldots, n\,, \tag{1.46}$$

která se oproti Jacobiho iterační formule liší ve skutečnosti, že vypočtené kořeny soustavy $x_j^{(k)}$ jsou k dalšímu výpočtu použity ihned a nikoliv až v dalším iteračním cyklu. Celý výpočet pak konverguje k přesnému řešení rychleji.

Výpočet Gauss-Seidelovou iterační metodou pro konečný počet m iteračních cyklů je schématicky vyjádřen algoritmem 11. Postup lze samozřejmě opět doplnit i zakončovací podmínkou 1.37.

Obsah
49. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : m, n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$,
$\mathbf{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\}^T$
Výstup: $\mathbf{x}^{(m)} = \{x_1^{(m)}, x_2^{(m)}, \ldots, x_n^{(m)}\}^T$

for $k \leftarrow 1, 2, \ldots, m-1, m$ **do**
  for $i \leftarrow 1, 2, \ldots, n-1, n$ **do**

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k-1)}}{a_{i,i}}$$

  end
end

Algoritmus 11: Algoritmus Gauss-Seidelovy iterační metody

Obsah
50. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.20. Vyřešte soustavu lineárních rovnic z příkladu 1.19 Gauss-Seidelovou iterační metodou. Při výpočtu uvažujte tolerancí nepřesnosti $\varepsilon = 1 \cdot 10^{-6}$ a nultou aproximaci kořenů řešené soustavy $\mathbf{x}^{(0)} = \{1, 1, 1, 1\}^T$.

Řešení. Samotný průběh iteračního výpočtu lze zobrazit např. takto:

Průběh Gauss-Seidelovy iterace

c.it.	x[1]	x[2]	x[3]	x[4]
0	1.000000	1.000000	1.000000	1.000000
1	0.750000	0.937500	0.437500	0.593750
2	0.593750	0.796875	0.296875	0.523438
3	0.523438	0.761719	0.261719	0.505859
4	0.505859	0.752930	0.252930	0.501465
5	0.501465	0.750732	0.250732	0.500366
6	0.500366	0.750183	0.250183	0.500092
7	0.500092	0.750046	0.250046	0.500023
8	0.500023	0.750011	0.250011	0.500006
9	0.500006	0.750003	0.250003	0.500001
10	0.500001	0.750001	0.250001	0.500000
11	0.500000	0.750000	0.250000	0.500000
12	0.500000	0.750000	0.250000	0.500000

K dosažení výsledku s tolerancí nepřesností $\varepsilon = 1 \cdot 10^{-6}$ je tedy při výpočtu Gauss-Seidelovou iterací celkem 12-ti iteračních cyklů.

▲

Obsah
51. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2.3. Řídké a pásové matice

V numerické matematice se při řešení soustav rovnic často vyskytují matice soustavy, které jsou řídké a pásové.

Řídkou maticí se rozumí matice, která obsahuje značný počet nulových prvků.

Pásová matice pak má nenulové prvky pouze v těsné blízkosti diagonály. Šířka pásu matice pak představuje maximální počet sloupců v blízkosti diagonály matice s nenulovými prvky. Rovněž lze rozlišit i tzv. šířky polopásu p, q, kterými se rozumí maximální počet sloupců s nenulovými prvky vlevo a vpravo od diagonály.

Příklad 1.21. Vyřešte kořeny soustavu lineárních rovnic s pásovou maticí soustavy:

$$\begin{bmatrix} 3 & -1 & & & \\ -1 & 3 & -1 & & \\ & & \ddots & & \\ & & -1 & 3 & -1 \\ & & & -1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} 2 \\ 1 \\ \vdots \\ 1 \\ 2 \end{Bmatrix} \tag{1.47}$$

pro $n = 100$ Gauss-Seidelovou metodou, upravenou pro řešení pásových matic. Uvažujte tolerancí nepřesnosti hodnotou $\varepsilon = 1 \cdot 10^{-6}$ a nultou aproximaci kořenů řešené soustavy $\mathbf{x}^{(0)} = \{0, 0, \ldots, 0, 0\}^T$.

Řešení. Matici soustavy **A** i vektor pravých stran lze vygenerovat příkazy:

```
clc;
clear;
n=1000;
E=ones(n,1);
```

Obsah
52. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
A=spdiags([-E 3*E -E],-1:1,n,n);
b=ones(n,1);
b(1)=2;
b(n)=2;
x=zeros(n,1);
```

Výpočet je pak možno provést m-funkcí, která implementuje výpočetní postup Gauss--Seidelovy iterační metody, upravené pro řešení pásových matic. Šířky polopásu p, q jsou v tomto případě rovny 1.

```
function x=gauss_seidel_pas(A,b,x,p,q,eps)
n=length(A);
maxkrok=1000;
y=x;
k=1;
for i=1:n
  s1=0;
  if i>1
    od=i-p;
    if od<1
      od=1;
    end
    for j=od:i-1
      s1=s1+A(i,j)*x(j);
    end
```

Obsah
53. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
  end
  s2=0;
  if i<n
    do=i+q;
    if do>n
      do=n;
    end
    for j=i+1:do
      s2=s2+A(i,j)*y(j);
    end
  end
  if A(i,i)==0
    error('Na diagonále je nula!')
    return
  end
  x(i)=(b(i)-s1-s2)/A(i,i);
end
while ~(norm(y-x)<eps) & k<maxkrok
  y=x;
  k=k+1;
  for i=1:n
    s1=0;
    if i>1
      od=i-p;
      if od<1
        od=1;
```

Obsah

54. strana ze 62

Zavřít dokument

Konec

Celá obrazovka/Okno

```
      end
      for j=od:i-1
        s1=s1+A(i,j)*x(j);
      end
    end
    s2=0;
    if i<n
      do=i+q;
      if do>n
        do=n;
      end
      for j=i+1:do
        s2=s2+A(i,j)*y(j);
      end
    end
    if A(i,i)==0
      error('Na diagonále je nula!')
      return
    end
    x(i)=(b(i)-s1-s2)/A(i,i);
  end
end
if k==maxkrok
  disp('Výpočet nebyl ukončen ukončovací podmínkou!')
end
```

Obsah

55. strana ze 62

Zavřít dokument

Konec

Celá obrazovka/Okno

Výpočetní utilita je rovněž doplněna zakončovací podmínkou, že maximální počet iteračních cyklů může být nejvýše 1000. Pokud bude výpočet ukončen splněním této podmínky, znamená to, že iterační výpočet nekonverguje dostatečně rychle nebo diverguje. ▲

Příklad 1.22. Proveďte výpočet soustavy lineárních rovnic s pásovou maticí soustavy z příkladu 1.21 Gauss-Seidelovou iterační metodou pro $n = 1000$. Uvažujte tolerancí nepřesnosti hodnotou $\varepsilon = 1 \cdot 10^{-6}$ a nultou aproximaci kořenů řešené soustavy $\mathbf{x}^{(0)} = \{0, 0, \ldots, 0, 0\}^T$.

Příklad 1.23. Vyřešte kořeny soustavu lineárních rovnic s pásovou maticí soustavy:

$$\begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{Bmatrix} \tag{1.48}$$

pro $n = 100$ Gauss-Seidelovou metodou, upravenou pro řešení pásových matic. Uvažujte tolerancí nepřesnosti hodnotou $\varepsilon = 1 \cdot 10^{-6}$ a nultou aproximaci kořenů řešené soustavy $\mathbf{x}^{(0)} = \{0, 0, \ldots, 0, 0\}^T$.

Řešení. Správné řešení kořenů soustavy se rovná vektoru

$$\{x\} = \begin{Bmatrix} 1 & -1 & 1 & -1 & \cdots & 1 & -1 \end{Bmatrix}^T . \tag{1.49}$$

▲

Příklad 1.24. Proveďte výpočet soustavy lineárních rovnic s pásovou maticí soustavy z příkladu 1.23 Gauss-Seidelovou iterační metodou pro $n = 1000$. Uvažujte tolerancí nepřesnosti hodnotou $\varepsilon = 1 \cdot 10^{-6}$ a nultou aproximaci kořenů řešené soustavy $\mathbf{x}^{(0)} = \{0, 0, \ldots, 0, 0\}^T$.

Obsah
56. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2.4. Metoda sdružených gradientů

Metoda sdružených gradientů (Conjugate Gradient Method) je vhodným výpočetním postupem pro řešení soustav lineárních rovnic, u nichž je matice soustavy $\mathbf{A}$ čtvercová, symetrická a pozitivně definitní.

Princip metody sdružených gradientů spočívá ve vhodně zvolené posloupnosti vektorů $\mathbf{d}^{(k)}$, které určují směr postupu od $\mathbf{x}^{(k)}$ k další aproximaci $\mathbf{x}^{(k+1)}$. Vektory $\mathbf{d}^{(k)}$ se postupně konstruují z posloupnosti reziduálních vektorů $\mathbf{r}^{(k)}$ tak, aby pro skalární součin $(\mathbf{d}^{(k)}, \mathbf{A}\cdot\mathbf{d}^{(k-1)})$ platilo $(\mathbf{d}^{(k)}, \mathbf{A}\cdot\mathbf{d}^{(k-1)}) = 0$. Posloupnost vektoru $\mathbf{d}^{(k)}$ je pak $\mathbf{A}$-ortogonální.

Postup výpočtu je schématicky popsán algoritmem 12, ve kterém se nejdříve vytvoří počáteční aproximace $\mathbf{x}^{(0)} = 0$ a reziduální vektor $\mathbf{r}^{(0)} = \mathbf{b}$. Za první směr $\mathbf{d}^{(0)}$ se zvolí $\mathbf{r}^{(0)}$. Určí se hodnota $\alpha^{(0)}$, pro kterou funkce (kvadratická forma) $F(\mathbf{x}^{(0)} + \alpha\cdot\mathbf{r}^{(0)}) = F(\mathbf{x}) = \frac{1}{2}\cdot(\mathbf{A}\cdot\mathbf{x},\mathbf{x}) - (\mathbf{b},\mathbf{x})$ nabývá svého minima, takže platí:

$$\alpha^{(0)} = \frac{(\mathbf{d}^{(0)}, \mathbf{r}^{(0)})}{(\mathbf{d}^{(0)}, \mathbf{A}\cdot\mathbf{d}^{(0)})} = \frac{(\mathbf{r}^{(0)})^T\cdot\mathbf{r}^{(0)}}{(\mathbf{d}^{(0)})^T\cdot\mathbf{A}\cdot\mathbf{d}^{(0)}}\,. \tag{1.50}$$

Nyní se opraví aproximace řešení soustavy $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha^{(0)}\cdot\mathbf{d}^{(0)}$, vypočte se nový reziduální vektor $\mathbf{r}^{(1)} = \mathbf{r}^{(0)} - \alpha^{(0)}\cdot\mathbf{A}\cdot\mathbf{d}^{(0)}$ a určí se nový směr postupu $\mathbf{d}^{(1)}$ ve tvaru $\mathbf{d}^{(1)} = \mathbf{r}^{(1)} + \beta^{(0)}\cdot\mathbf{d}^{(0)}$ tak, aby platilo $(\mathbf{d}^{(1)}, \mathbf{A}\cdot\mathbf{d}^{(0)}) = 0$. Hodnota $\beta^{(0)}$ se přitom určí ze vztahu:

$$\beta^{(0)} = -\frac{(\mathbf{d}^{(0)}, \mathbf{A}\cdot\mathbf{r}^{(1)})}{(\mathbf{d}^{(0)}, \mathbf{A}\cdot\mathbf{d}^{(0)})} = \frac{(\mathbf{r}^{(1)})^T\cdot\mathbf{r}^{(1)}}{(\mathbf{r}^{(0)})^T\cdot\mathbf{r}^{(0)}}\,. \tag{1.51}$$

Dále se určí hodnota $\alpha^{(1)}$ tak, aby funkce $F(\mathbf{x}^{(1)} + \alpha\cdot\mathbf{r}^{(1)})$ nabývala minima, a celý výpočet se opakuje tak dlouho, dokud není splněna zakončovací podmínka.

Obsah
57. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : m, n, $\mathbf{A} = [a_{i,j}] = [a_{1,1}, \ldots, a_{n,n}]$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$
Výstup: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$

$\mathbf{x}^{(0)} \leftarrow 0$
$\mathbf{d}^{(0)} \leftarrow \mathbf{r}^{(0)} \leftarrow \mathbf{b}$
for $k \leftarrow 1, 2, \ldots, m-1, m$ **do**
  if $\|\mathbf{r}^{(k-1)}\| < \varepsilon$ **then**
    stop
  else
    $$\alpha^{(k-1)} \leftarrow \frac{(\mathbf{r}^{(k-1)})^T \cdot \mathbf{r}^{(k-1)}}{(\mathbf{d}^{(k-1)})^T \cdot \mathbf{A} \cdot \mathbf{d}^{(k-1)}}$$
    $$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} + \alpha^{(k-1)} \cdot \mathbf{d}^{(k-1)}$$
    $$\mathbf{r}^{(k)} \leftarrow \mathbf{r}^{(k-1)} - \alpha^{(k-1)} \cdot \mathbf{A} \cdot \mathbf{d}^{(k-1)}$$
    $$\beta^{(k-1)} \leftarrow \frac{(\mathbf{r}^{(k)})^T \cdot \mathbf{r}^{(k)}}{(\mathbf{r}^{(k-1)})^T \cdot \mathbf{r}^{(k-1)}}$$
    $$\mathbf{d}^{(k)} \leftarrow \mathbf{r}^{(k)} + \beta^{(k-1)} \cdot \mathbf{d}^{(k-1)}$$
  end
end

Algoritmus 12: Algoritmus metody sdružených gradientů

Obsah
58. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

Výpočetní postup řešení soustavy rovnic metodou sdružených gradientů lze naprogramovat v programu MATLAB např. následujícím způsobem:

```
function x=sdruz_grad(A,b,eps)
n=length(A);
maxkrok=1000;
x=zeros(n,1);
d=b;
r=b;
for k=1:maxkrok
  if norm(r)<eps
    return
  end
  alfa=r'*r/(d'*A*r);
  x=x+alfa*d;
  rs=r;
  r=r-alfa*A*d;
  beta=r'*r/(rs'*rs);
  d=r+beta*d;
  if k==maxkrok
    disp('Výpočet nebyl ukončen ukončovací podmínkou!')
  end
end
```

Výpočet hodnot $\alpha^{(k-1)}$ a $\beta^{(k-1)}$ lze v m-funkci rovněž určit s využitím příkazu programu MATLAB pro skalární součin dvou vektorů s názvem `dot`, např.:

Obsah
59. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
alfa=dot(d,r)/dot(d,A*d);
```

resp.

```
beta=-dot(d,A*r)/dot(d,A*d);
```

Příklad 1.25. Proveďte výpočet soustavy lineárních rovnic s pásovou maticí soustavy z příkladu 1.21 metodou sdružených gradientů pro $n = 10000$ a tolerancemi nepřesnosti $\varepsilon = 1 \cdot 10^{-6}$ a $\varepsilon = 1 \cdot 10^{-12}$.

Příklad 1.26. Proveďte výpočet soustavy lineárních rovnic s pásovou maticí soustavy z příkladu 1.23 metodou sdružených gradientů pro $n = 1000$ a tolerancemi nepřesnosti $\varepsilon = 1 \cdot 10^{-6}$ a $\varepsilon = 1 \cdot 10^{-12}$.

Příklad 1.27. Vyřešte kořeny soustavu lineárních rovnic s řídkou maticí soustavy:

$$\begin{bmatrix} 3 & -1 & & & & & & & & 0.5 \\ -1 & 3 & -1 & & & & & & 0.5 & \\ & -1 & 3 & -1 & & & & 0.5 & & \\ & & & \ddots & & & \iddots & & & \\ & & & -1 & 3 & -1 & & & & \\ & & & & -1 & 3 & -1 & & & \\ & & & \iddots & & & \ddots & & & \\ & & 0.5 & & & & -1 & 3 & -1 & \\ & 0.5 & & & & & & -1 & 3 & -1 \\ 0.5 & & & & & & & & -1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ \\ \vdots \\ \\ \\ \\ \vdots \\ \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} 2.5 \\ 1.5 \\ \vdots \\ 1.5 \\ 1 \\ 1 \\ 1.5 \\ \vdots \\ 1.5 \\ 2.5 \end{Bmatrix} \tag{1.52}$$

pro $n = 10000$ metodou sdružených gradientů s požadovanou tolerancí nepřesnosti $1 \cdot 10^{-6}$.

Řešení. Matici soustavy **A** i vektor pravých stran lze vygenerovat příkazy:

```
clc;
clear;
n=100;
E=ones(n,1);
n2=n/2;
A=spdiags([-E 3*E -E],-1:1,n,n);
C=spdiags([E/2],0,n,n);
C=fliplr(C);
A=A+C;
```

Obsah
61. strana ze 62
Zavřít dokument
Konec
Celá obrazovka/Okno

```
A(n2+1,n2)=-1;
A(n2,n2+1)=-1;
b=zeros(n,1);
b(1)=2.5;
b(n)=2.5;
b(2:n-1)=1.5;
b(n2:n2+1)=1;
```

Správné řešení kořenů soustavy se rovná vektoru

$$\{x\} = \begin{Bmatrix} 1 & 1 & \cdots & 1 & 1 \end{Bmatrix}^T . \tag{1.53}$$

▲

Obsah

62. strana ze 62

Zavřít dokument

Konec

Celá obrazovka/Okno