ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Katedra matematiky

Numerická analýza

Užití Matlabu k řešení numerických úloh

Ondřej Tuček
ondrej_tucek@seznam.cz

PLZEŇ 2006

Obsah

Úvod

V této práci využijeme získaných poznatků z předmětu NA či jiných podobných předmětů (např. LA), ale především také z různých skript či knih zabývajících se numerickou matematikou. Tyto zdroje jsou uvedeny na konci této práce. Značení, definice, věty atd. použijeme ze skript pana prof. S. Míky [1]. Tyto věty či tvrzení budou uvedena bez důkazů, neboť podstatou této práce je užití těchto vět v "numerické" praxi. Numerickou praxí rozumějme aplikování těchto znalostí při řešení různých numerických úloh. Jedním z mnoha nástrojů k nalezení tohoto řešení je program Matlab.

1 Obor hodnot a nulový prostor lineárního zobrazení

Mějme lineární zobrazení $\mathbf{A} : \mathcal{R}^N \to \mathcal{R}^M$ a $\mathbf{A}^T : \mathcal{R}^M \to \mathcal{R}^N$. Zvolme $M = 3$, $N = 5$ a matici $\mathbf{A}$ takto:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad \text{pak matice k ní transponovaná je} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Nyní hledejme nulové prostory a obory hodnot zobrazení $\mathbf{A}$ a $\mathbf{A}^T$, tedy zajímá nás jak vypadá množina nezávislých sloupců a řádků matice $\mathbf{A}$ a $\mathbf{A}^T$. Použitím tutorial3.pdf z [4] a nápovědy Matlabu vytvoříme m-skript dimenze.m

Algoritmus 1 dimenze.m

```
function [LN_sm, LN_rm, np] = dimenze(A)
% LN_sm - lin. nezavisle sloupce matice A,
%         tj. obor hodnot matice A
% LN_rm - lin. nezavisle radky matice A
% np - nulovy prostor matice A

[R, jb] = rref(A);
r = length(jb);
LN_sm = double(colspace(sym(A)))
LN_rm = R(1:r,:)'
np = null(A,'r')
```

Pozastavme se na chvíli u třetího řádku zdola v algoritmu 1. V nápovědě k funkci `colspace` se dozvíme, že se dá tato funkce použít i bez symbolického zadání matice. Bohužel v mé verzi Matlabu (6.5 R13) to nejde. Proto převod symbolické matice na matici numerickou realizuji pomocí funkce `double`.

Nyní napišme do Command Window v Matlabu námi zadanou matici $\mathbf{A}$:

```
>> A = [1 1 1 1 1; 1 0 1 0 0; 1 1 1 0 1];
```

a aplikujme algoritmus 1:

```
>> dimenze(A)
```

dostaneme

$$\mathrm{LN\_sm} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{LN\_rm} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathrm{np} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Naskýtá se otázka, zda je výsledek Matlabu správný. O tom se přesvědčíme následovně. V matici $\mathbf{A}$ provedeme sloupcové úpravy, tj. k třetímu sloupci přičteme -1 násobek prvního sloupce, k pátému sloupci přičteme -1 násobek druhého, atd.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \to \mathrm{LN\_sm} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Odtud vidíme, že lineárně nezávislé sloupce matice $\mathbf{A}$ tvoří matici LN_sm. Tedy sloupce matice LN_sm tvoří obor hodnot zobrazení $\mathbf{A}$. Z definice nulového prostoru ověříme zda platí $\mathbf{A}\mathbf{v}_i = \mathbf{0}_3, i = 1, 2$, kde $\mathbf{v}_i$ tvoří sloupce matice np, tj. $\mathrm{np} = [\mathbf{v}_1, \mathbf{v}_2]$. Tedy ukážeme, že platí $\mathbf{A}\mathbf{v}_1 = \mathbf{0}_3$ a $\mathbf{A}\mathbf{v}_2 = \mathbf{0}_3$.

$$\mathbf{A}\mathbf{v}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{A}\mathbf{v}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Zbývá jenom ukázat, že lineárně nezávislé sloupce matice $\mathbf{A}$ (resp. $\mathbf{A}^T$) tvoří lineárně nezávislé řádky matice $\mathbf{A}^T$ (resp. $\mathbf{A}$). Stejným postupem použijeme algoritmus 1 na transponovanou matici, tj. na $\mathbf{A}^T$:

```
>> dimenze(A')
```

a dostaneme

$$\mathrm{LN\_sm} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathrm{LN\_rm} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{np} = \texttt{Empty matrix: 3-by-0}$$

Porovnáme-li sloupce matic LN_sm u $\mathbf{A}$ a LN_rm u $\mathbf{A}^T$ (resp. sloupce matic LN_rm u $\mathbf{A}$ a LN_sm u $\mathbf{A}^T$) zjistíme, že se liší až na pořadí sloupců. Tedy lze odtud usuzovat na správnost výsledků získaných Matlabem. Dále vidíme, že nulový prostor zobrazení $\mathbf{A}^T$ je prázdná množina. Tyto výsledky byly též ověřeny v programu Maple verze 9.

2 Levostranná zobecněná inverzní matice

Uvažujme nekonzistentní úlohu "$\mathbf{Ax} = \mathbf{b}$" za těchto předpokladů:

a) $\mathbf{b} \notin \mathcal{H}(\mathbf{A}), \mathbf{b} \in \mathcal{R}^M, \mathcal{H}(\mathbf{A}) \subset \mathcal{R}^M$;

b) $\mathbf{A} \in \mathcal{R}^{M,N}$ (matice typu $M \times N$), $\mathbf{A} : \mathcal{R}^N \rightarrow \mathcal{R}^M, \mathbf{x} \in \mathcal{R}^N$;

c) $\dim \mathcal{H}(\mathbf{A}) = h = N, M > N$.

Na základě těchto předpokladů sestrojíme algoritmus 2 takto

Algoritmus 2 levostranna.m

```
function levostranna(A,b)
% Levostranna zobecnena inverzni matice
% Uvazujeme nekonzistentni ulohu "Ax = b"

format long g                          % pocitame na 15 des. mist
[M, N] = size(A);                      % rozmery matice A
LN_sm = double(colspace(sym(A)));     % obor hodnot zobrazeni A
h = rank(LN_sm);                      % sloupcova hodnost matice A
b = b';
Mb = length(b);
i = 1;
lze_resit = 0;

% Pozadovane vlastnosti a), b), c) dle [1]
if (M > N) & (h == N) & (Mb == M)
    while i <= h
        if b(:) == LN_sm(:,i)
           disp('Vektor pravych stran nesmi byt prvkem H(A)')
           lze_resit = 1;
           break
        end % of if
        i = i + 1;
    end % of while
    if lze_resit ~= 1
       x_p = inv(A'*A)*A'*b
    end
else
    disp('Zvolte jinou matici A (pocet linearne nezavislych sloupcu
          matice A neni roven N nebo M <= N) nebo vektor b nema
          M slozek.')
end % of if M > N ...

x = A\b
```

Předpoklad a) je dán podmínkou `if b(:) == LN\_sm(:,i)`. Cyklem `while` procházíme jednotlivé sloupce matice `LN_sm` a testujeme zda vektor pravých stran není obsažen v této matici. Ovšem můžeme zadat libovolně velkou matici $\mathbf{A}$, která bude mít třeba 100 000 lineárně nezávislých sloupců (tj. tyto sloupce tvoří matici `LN_sm`). Dejme tomu, že vektor $\mathbf{b}$ je roven sloupci až na pozici 3275. Potom z hlediska časové náročnosti je zbytečné testovat všechny sloupce od 3276 do konce, když už jsme jeden sloupec nalezli. Proto příkazem `break` ukončujeme toto hledaní. Předpoklad c) je dán podmínkou `if (M > N) && (h == N)`.

Zvolme $M = 4$ a $N = 3$, matici $\mathbf{A}$ a vektor pravých stran $\mathbf{b}$ následovně:

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & 4 \\ 6 & 8 & 1 \\ 0 & 7 & 9 \\ 3 & 4 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$, vektoru $\mathbf{b}$ do Matlabu a použitím algoritmu 2

```
>> A = [1 5 4; 6 8 1; 0 7 9; 3 4 0];
>> b = [2 3 0 0];
>> levostranna(A,b)
```

dostáváme

$$\mathrm{x\_p} = \begin{bmatrix} -0.600649350649339 \\ 0.823747680890518 \\ -0.591372912801473 \end{bmatrix}, \quad \mathrm{x} = \begin{bmatrix} -0.600649350649351 \\ 0.823747680890538 \\ -0.591372912801484 \end{bmatrix}$$

kde x_p je řešení spočtené z `inv(A'*A)*A'*b` a x je řešení spočtené standardní procedurou `A\b`.

Porovnáme-li tyto dvě řešení zjistíme, že se začínají lišit (jednotlivé složky) až od předposledního místa. Rozdílnost může být dána nepřesným provedením aritmetických operací, neboť řešení x_p je realizováno operacemi inverze, transponování a součin matic oproti "jedné" operaci `A\b`. Ovšem musíme se ptát, jak Matlab tuto operaci realizuje v závislosti na typu matice. V nápovědě Matlabu zjistíme toto:

Jestliže $\mathbf{A}$ je matice typu $M \times N$, $M \neq N$ a $\mathbf{b}$ je sloupcový vektor s M složkami, pak hodnost k matice $\mathbf{A}$ je určena QR rozkladu s pivotací. Řešení $\mathbf{x}$ má maximálně k nenulových složek v sloupci (vektoru) a je spočteno pomocí ortogonálního rozkladu Housholderovou metodou. Nejdříve se stanoví $\mathbf{AP} = \mathbf{QU}$, kde $\mathbf{P}$ je permutační matice, $\mathbf{Q}$ je ortogonální matice, $\mathbf{U}$ je horní trojúhelníková matice a pak $\mathbf{x} = \mathbf{P}(\mathbf{U}\backslash(\mathbf{Q}'\mathbf{b}))$.

Tedy z jedne operace `A\b` hned máme operací několik plus jednu či dvě metody na rozklad. Princip ortogonálního rozkladu Housholderovou metodou bude ukázán v odstavci 8.

Nalezená řešení nejsou správná, neboť provedeme-li zkoušku, tj. x_p a x dosadíme do $\mathbf{Ax} = \mathbf{b}$ dostáváme

$$\mathrm{Ax\_p} = \begin{bmatrix} 1.15259740259736 \\ 2.39471243042664 \\ 0.44387755102037 \\ 1.49304267161406 \end{bmatrix} \neq \mathrm{b}, \quad \mathrm{Ax} = \begin{bmatrix} 1.15259740259740 \\ 2.39471243042672 \\ 0.44387755102041 \\ 1.49304267161410 \end{bmatrix} \neq \mathrm{b}$$

3 Pravostranná zobecněná inverzní matice

Uvažujme konzistentní úlohu $\mathbf{Ax} = \mathbf{b}$ za těchto předpokladů:

a) $\mathbf{b} \in \mathcal{H}(\mathbf{A}) \subset \mathcal{R}^M$;

b) $\mathbf{A} \in \mathcal{R}^{M,N}$ (matice typu $M \times N$), $\mathbf{A} : \mathcal{R}^N \to \mathcal{R}^M, \mathbf{x} \in \mathcal{R}^N$;

c) $\dim \mathcal{H}(\mathbf{A}) = h = M, M \leq N$.

Na základě těchto předpokladů sestrojíme algoritmus 3, který bude pracovat obdobně jako algoritmus 2.

Algoritmus 3 pravostranna.m

```
function pravostranna(A,b)
% Pravostranna zobecnena inverzni matice
% Uvazujeme konzistentni ulohu Ax = b

format long g                          % pocitame na 15 des. mist
[M, N] = size(A);                      % rozmery matice A
LN_sm = double(colspace(sym(A)))      % obor hodnot zobrazeni A
h = rank(LN_sm);                      % sloupcova hodnost matice A
[R, jb] = rref(A); r = length(jb);
np = null(A,'r')                       % np - nulovy prostor matice A
b = b'; Mb = length(b);
i = 1; lze_resit = 0;

% Pozadovane vlastnosti a), b), c) dle [1]
if (M <= N) & (h == M) & (Mb == M)
    while i <= h
        if b(:) ~= LN_sm(:,i)
           disp('Vektor pravych stran neni zvolen z H(A)')
           lze_resit = 1;
           break
        end % of if
        i = i + 1;
    end % of while
    if lze_resit == 1
       x_p = A'*inv(A*A')*b
    end
else
    disp('Zvolte jinou matici A (pocet linearne nezavislych sloupcu
          matice A neni roven M nebo M > N) nebo vektor b nema
          M slozek.')
end % of if M <= N ...

x = A\b
```

Zvolme $M = 3$ a $N = 4$, matici $\mathbf{A}$ a vektor pravých stran $\mathbf{b}$ následovně:

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & -4 & 0 \\ 6 & 8 & 0 & 1 \\ -3 & 0 & 7 & 9 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 17 \\ 0 \\ 0 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$, vektoru $\mathbf{b}$ do Matlabu a použitím algoritmu 3

```
>> A = [1 5 -4 0; 6 8 0 1; -3 0 7 9];
>> b = [17 0 0];
>> pravostranna(A,b)
```

dostáváme

$$\mathrm{np} = \begin{bmatrix} 1.012 \\ -0.884 \\ -0.852 \\ 1 \end{bmatrix}, \quad \mathrm{x\_p} = \begin{bmatrix} -2.37863980898790 \\ 1.60742844974832 \\ -2.83537439006157 \\ 1.41241125594081 \end{bmatrix}, \quad \mathrm{x} = \begin{bmatrix} 0 \\ -0.47035573122529 \\ -4.83794466403162 \\ 3.76284584980237 \end{bmatrix}$$

kde x_p je řešení spočtené z `A'*inv(A*A')*b` a x je řešení spočtené standardní procedurou `A\b`.

Ukážeme, že obě tato vypočtená řešení jsou správná. Poznamenejme, že hodnost matice $\mathbf{A}$ je 3 a tedy řešení x má právě tři nenulové složky jak bylo vysvětleno v závěru předchozího odstavce. Z [1] víme, že soustava $\mathbf{Ax} = \mathbf{b}$ pro $M < N$ má nekonečně mnoho řešení ve tvaru

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{u} = \bar{\mathbf{x}} + \sum_{i=1}^{N-M} \alpha_i \mathbf{s}_i,$$

kde $\alpha_i \in \mathcal{R}$ a $\mathbf{s}_i \in \mathcal{N}(\mathbf{A}) \subset \mathcal{R}^N$ jsou lineárně nezávislá řešení homogenní soustavy $\mathbf{Au} = \mathbf{0}$, tj. $\mathbf{s}_i$ jsou bázové vektory nulového prostoru $\mathcal{N}(\mathbf{A}) = \mathrm{np} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_i]$ a $\bar{\mathbf{x}}$ je nějaké partikulární řešení dané soustavy, tj. $\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$.

Označíme-li $\mathbf{x} = \mathrm{x\_p}$ a $\bar{\mathbf{x}} = \mathrm{x}$ pak dle výše uvedeného platí $\mathbf{u} = \mathrm{x\_p} - \mathrm{x} = \alpha_1 \mathbf{s}_1$. Odtud stanovíme $\alpha_1 = -2.35043459386156$. Vektor $\mathbf{u}$ a součin $\mathbf{Au}$ vypadají takto

$$\mathrm{u} = \begin{bmatrix} -2.37863980898790 \\ 2.07778418097362 \\ 2.00257027397005 \\ -2.35043459386156 \end{bmatrix}, \quad \mathrm{Au} = \begin{bmatrix} 5.32907051820075 \cdot 10^{-15} \\ -8.88178419700125 \cdot 10^{-16} \\ -3.55271367880050 \cdot 10^{-15} \end{bmatrix}$$

O součinu $\mathbf{Au}$ můžeme říct, že je roven "nule", neboť vypočtené hodnoty jsou hodně malé. Pro úplnost proveďme ještě zkoušku: x_p a x dosadíme do $\mathbf{Ax} = \mathbf{b}$ a dostáváme

$$\mathrm{Ax\_p} = \begin{bmatrix} 17.0 \\ -1.99840144432528 \cdot 10^{-15} \\ 0 \end{bmatrix} \approx \mathrm{b}, \quad \mathrm{Ax} = \begin{bmatrix} 17.0 \\ -2.22044604925031 \cdot 10^{-15} \\ 7.10542735760100 \cdot 10^{-15} \end{bmatrix} \approx \mathrm{b}$$

Lze tedy považovat řešení x_p a x za správná.

4 Gaussova eliminace bez výběru pivota

Nechť matice $\mathbf{A}$ je regulární a $\mathbf{A} \in \mathcal{C}^{N,N}$, $\mathbf{b} \in \mathcal{C}^{N}$ a nechť $\bar{\mathbf{x}} \in \mathcal{C}^{N}$ je jediné řešení soustavy $\mathbf{Ax} = \mathbf{b}$. Hledejme toto řešení $\bar{\mathbf{x}}$ pomocí algoritmu 4.

Algoritmus 4 GEM.m

```
function GEM(A,b)
% Gaussova eliminace bez vyberu pivota
b = b';
[M, N] = size(A);
Nb = length(b);

if (M == N) & (Nb == N)
    if (diag(A) ~= 0) & (det(A) ~= 0)
        for k = 1:N-1          % primy chod
            T = eye(rank(A));
            for i = k+1:N
                T(i,k) = - A(i,k)/A(k,k)
                A(i,:) = A(i,:) + T(i,k)*A(k,:)
                b(i,1) = b(i,1) + T(i,k)*b(k,1)
            end
        end
        x(N)=0;
        x(N) = b(N)/A(N,N)
        for i = N-1:-1:1     % zpetny chod
            pom = 0;
            for j=i+1:N
                pom = pom + A(i,j) * x(j);
            end
            x(i) = (b(i) - pom)/A(i,i)
        end
        x = x'                 % reseni
    else
        disp('Nelze pouzit GEM. Zadana matice obsahuje v
              diagonalnim prvku nulu nebo je singularni.')
    end % of diag ...
else
    disp('Musite zadat ctvercovou matici a vektor pravych stran
          stejneho radu jako je rad matice.')
end
```

Zvolme $N = 4$, matici $\mathbf{A}$ a vektor pravých stran $\mathbf{b}$ takto:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 1 & 3 & 1 & 2 \\ 2 & 3 & -3 & 9 \\ 1 & 4 & 4 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 2 \\ 20 \\ 9 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$, vektoru $\mathbf{b}$ do Matlabu a použitím algoritmu 4

```
>> A = [1 2 -1 3; 1 3 1 2; 2 3 -3 9; 1 4 4 4];
>> b = [5 2 20 9];
>> GEM(A,b)
```

dostáváme:

Přímý chod $\qquad \mathbf{A}^{(0)} = \mathbf{A} \qquad \mathbf{b}^{(0)} = \mathbf{b}$

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{A}^{(1)} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 1 & 2 & -1 \\ 0 & -1 & -1 & 3 \\ 0 & 2 & 5 & 1 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} 5 \\ -3 \\ 10 \\ 4 \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{bmatrix}, \quad \mathbf{A}^{(2)} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{bmatrix} 5 \\ -3 \\ 7 \\ 10 \end{bmatrix}$$

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \quad \mathbf{A}^{(3)} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b}^{(3)} = \begin{bmatrix} 5 \\ -3 \\ 7 \\ 3 \end{bmatrix}$$

Zpětný chod:

$$\mathbf{x}^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \end{bmatrix}, \quad \mathbf{x}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{bmatrix}, \quad \mathbf{x}^{(2)} = \begin{bmatrix} 0 \\ -2 \\ 1 \\ 3 \end{bmatrix}, \quad \mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ -2 \\ 1 \\ 3 \end{bmatrix} = \bar{\mathbf{x}}$$

Dále je vidět, že součin matic $\mathbf{T}_{N-1} \cdot \ldots \cdot \mathbf{T}_1$ je dolní trojúhelníková matice:

$$\mathbf{T} = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -2 & 1 & 1 & 0 \\ -1 & -2 & -1 & 1 \end{bmatrix}$$

5 LU rozklad

K dané čtvercové matici $\mathbf{A}$ určeme dolní trojúhelníkovou matici $\mathbf{L}$ a horní trojúhelníkovou matici $\mathbf{U}$ tak, aby platilo $\mathbf{A} = \mathbf{LU}$. K nalezení matic $\mathbf{L}, \mathbf{U}$ použijeme algoritmus 5 [2].

Algoritmus 5 lu_rozklad.m

```
function lu_rozklad(A);
% LU rozklad pro ctvercovou diagonalne dominantni matici

[M, N] = size(A);
U = zeros(N);
L = eye(N);
B = A - diag(diag(A));
a_ik = (sum(abs(B')))';
a_ii = abs(diag(A));

if (M == N) & (det(A) ~= 0) & (a_ii >= a_ik)
    for j=1:N
        for i = 1:j
            pom = 0;
            for k = 1:i-1
                pom = pom + L(i,k)*U(k,j);
            end
            U(i,j) = A(i,j)-pom
        end
        for i = j+1:N
            pom = 0;
            for s = 1:j-1
                pom = pom + L(i,s)*U(s,j);
            end
            L(i,j) = (A(i,j) - pom)/U(j,j)
        end
    end
    L       % dolni trojuhelnikova matice
    U       % horni trojuhelnikova matice
    L*U     % A = L*U
else
    disp('Zadana matice musi byt ctvercova, regularni a diagonalne
          dominantni.')
end
```

Zvolme $N = 3$ a matici $\mathbf{A}$ takto:

$$\mathbf{A} = \begin{bmatrix} 11 & 1 & 2 \\ 1 & 7 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$ do Matlabu a použitím algoritmu 5

```
>> A = [11 1 2; 1 7 2; 3 2 5];
>> lu_rozklad(A)
```

dostáváme:

j = 1

$$\mathbf{U} = \begin{bmatrix} 11.0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.0909 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.0909 & 1.0 & 0 \\ 0.2727 & 0 & 1.0 \end{bmatrix}$$

j = 2

$$\mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 0 \\ 0 & 6.909 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.0909 & 1.0 & 0 \\ 0.2727 & 0.2500 & 1.0 \end{bmatrix}$$

j = 3

$$\mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0 & 6.909 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0 & 6.909 & 1.818 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0 & 6.909 & 1.818 \\ 0 & 0 & 4.0 \end{bmatrix}$$

Hledané matice $\mathbf{L}$ a $\mathbf{U}$ jsou

$$\mathbf{L} = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.0909 & 1.0 & 0 \\ 0.2727 & 0.2500 & 1.0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0 & 6.909 & 1.818 \\ 0 & 0 & 4.0 \end{bmatrix}$$

Nyní ověříme, zda tyto nalezené matice $\mathbf{L}$ a $\mathbf{U}$ vyhovují vztahu $\mathbf{A} = \mathbf{LU}$.

$$\begin{bmatrix} 1.0 & 0 & 0 \\ 0.0909 & 1.0 & 0 \\ 0.2727 & 0.2500 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0 & 6.909 & 1.818 \\ 0 & 0 & 4.0 \end{bmatrix} = \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0.9999 & 6.9999 & 1.9998 \\ 2.9997 & 1.9999 & 4.9999 \end{bmatrix}$$

Tedy

$$\mathbf{A} = \begin{bmatrix} 11 & 1 & 2 \\ 1 & 7 & 2 \\ 3 & 2 & 5 \end{bmatrix} \approx \begin{bmatrix} 11.0 & 1.0 & 2.0 \\ 0.9999 & 6.9999 & 1.9998 \\ 2.9997 & 1.9999 & 4.9999 \end{bmatrix}$$

Takže $\mathbf{A} \approx \mathbf{LU}$.

6 QR rozklad

K dané matici $\mathbf{A}$ sestrojme ortogonální matici $\mathbf{Q}$ a horní trojúhelníkovou matici $\mathbf{U}$ tak, aby platilo $\mathbf{A} = \mathbf{QU}$. K nalezení matic $\mathbf{Q}$, $\mathbf{U}$ použijeme algoritmus 6 [2].

Algoritmus 6 qr_rozklad.m

```
function qr_rozklad(A);
%  QR rozklad pro matici A typu M x N, M >= N

[M, N] = size(A);
Q = eye(M);

if (M >= N) & (isreal(A) == 1)
    for q = 1:N
        for p = q+1:M
            Qpq = eye(M);
            if (A(q,q) ~= 0) && (A(p,q) ~= 0)
                r = sqrt(A(q,q)^2 + A(p,q)^2);
                c = A(q,q)/r;
                s = A(p,q)/r;
                Qpq(p,p) = c;
                Qpq(q,q) = c;
                Qpq(p,q) = -s;
                Qpq(q,p) = s;
            end
            Q = Qpq*Q
            A = Qpq*A
        end
    end
    Q            % ortogonalni matice
    U = A        % horni trojuhelnikova matice
    I_M = Q*Q'   % jednotkova matice
    Q'*U         % A = Q'*U
else
    disp('Zadejte matici A typu M x N, M => N a vsechny jeji prvky
         musi byt realna cisla.')
end
```

Zvolme $M = 4$ a $N = 3$ a matici $\mathbf{A}$ následovně:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 8 \\ 4 & 5 & 7 \\ 1 & 0 & 3 \\ 9 & 5 & 1 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$ a použitím algoritmu 6

```
>> A = [1 2 8; 4 5 7; 1 0 3; 9 5 1];
>> qr_rozklad(A)
```

dostáváme

$$\mathbf{Q}_1 = \begin{bmatrix} 0.1005 & 0.4020 & 0.1005 & 0.9045 \\ -0.9701 & 0.2425 & 0 & 0 \\ -0.05717 & -0.2287 & 0.9720 & 0 \\ -0.2132 & -0.8526 & -0.2132 & 0.4264 \end{bmatrix}, \quad \mathbf{A}^{(1)} = \begin{bmatrix} 9.951 & 6.734 & 4.826 \\ 0 & -0.7277 & -6.065 \\ 0 & -1.258 & 0.8577 \\ 0 & -2.558 & -7.889 \end{bmatrix}$$

$$\mathbf{Q}_2 = \begin{bmatrix} 0.1005 & 0.4020 & 0.1005 & 0.9045 \\ 0.4497 & 0.7793 & -0.2300 & -0.3708 \\ -0.8112 & 0.3244 & -0.4866 & 0 \\ 0.3603 & -0.3547 & -0.8368 & 0.2106 \end{bmatrix}, \quad \mathbf{A}^{(2)} = \begin{bmatrix} 9.951 & 6.734 & 4.826 \\ 0 & 2.942 & 7.992 \\ 0 & 0 & -5.678 \\ 0 & 0 & -1.901 \end{bmatrix}$$

$$\mathbf{Q}_3 = \begin{bmatrix} 0.1005 & 0.4020 & 0.1005 & 0.9045 \\ 0.4497 & 0.7793 & -0.2300 & -0.3708 \\ 0.6548 & -0.1951 & 0.7271 & -0.06685 \\ -0.5990 & 0.4392 & 0.6390 & -0.1997 \end{bmatrix}, \quad \mathbf{A}^{(3)} = \begin{bmatrix} 9.951 & 6.734 & 4.826 \\ 0 & 2.942 & 7.992 \\ 0 & 0 & 5.987 \\ 0 & 0 & 2.220 \cdot 10^{-16} \end{bmatrix}$$

Tedy $\mathbf{U} = \mathbf{A}^{(3)}$, $\mathbf{Q} = \mathbf{Q}_3$ a matice $\mathbf{I}_M = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_4$ vypadá takto:

$$\mathbf{I}_4 = \begin{bmatrix} 1.0 & 5.551 \cdot 10^{-17} & 2.776 \cdot 10^{-17} & 0 \\ 5.551 \cdot 10^{-17} & 1.0 & 4.163 \cdot 10^{-17} & 1.388 \cdot 10^{-16} \\ 2.776 \cdot 10^{-17} & 4.163 \cdot 10^{-17} & 1.0 & 8.153 \cdot 10^{-17} \\ 0 & 1.388 \cdot 10^{-16} & 8.153 \cdot 10^{-17} & 1.0 \end{bmatrix}$$

Pro kontrolu ještě stanovme součin $\mathbf{QU}$ a porovnejmeho s maticí $\mathbf{A}$

$$\mathbf{QU} = \begin{bmatrix} 1.0 & 2.0 & 8.0 \\ 4.0 & 5.0 & 7.0 \\ 1.0 & 4.441 \cdot 10^{-16} & 3.0 \\ 9.0 & 5.0 & 1.0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 2 & 8 \\ 4 & 5 & 7 \\ 1 & 0 & 3 \\ 9 & 5 & 1 \end{bmatrix}$$

Takže $\mathbf{A} \approx \mathbf{QU}$.

7 Choleského rozklad

K dané reálné symetrické matici $\mathbf{A}$ sestrojme horní trojúhelníkovou matici $\mathbf{U}$ tak, aby platilo $\mathbf{A} = \mathbf{U}^T\mathbf{U}$. K nalezení matice $\mathbf{U}$ použijeme algoritmus 7 [1],[2].

Algoritmus 7 cholesky.m

```
function cholesky(A)
% Choleskeho rozklad A = U'*U

[M, N] = size(A);
U = zeros(N);
vl_cisla = eig(A);
k = length(find(vl_cisla > 0));
z = length(find(vl_cisla < 0));
d = N -(k+z);

if (k>0) & (z==0) & (d==0) & (norm(A-A')==0) & (M==N) & (isreal(A)==1)
    for j = 1:N
        for i = 1:j-1
            pom = 0;
            for r = 1:i-1
                pom = pom + U(r,i)*U(r,j);
            end
            U(i,j) = (A(i,j) - pom)/U(i,i);
        end
        pom = 0;
        for s = 1:j-1
            pom = pom + U(s,j)^2;
        end
        U(j,j) = sqrt(A(j,j) - pom)
    end
    U        % horni trojuhelnikova matice
    U'*U     % A = U'*U
else
    disp('Zadana matice A musi byt ctvercova, symetricka, pozitivne
          definitni a vsechny jeji prvky musi byt realna cisla.')
end
```

Poznamenejme, že u tohoto algoritmu využíváme poznatků z lineární algebry [3]. Jde zejména o pojmy jako je inercie matice, tj. $\text{in}(\mathbf{A}) = (k, z, d)$, kvadratická forma $\kappa(\mathbf{x})$ jež se dá psát ve tvaru $\kappa(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ či pozitivní definitnost, tj. $\kappa(\mathbf{x}) > 0$ pro $\mathbf{x} \neq 0$. Z [3, str. 222] víme, že kvadratická forma na reálném lineárním (vektorovém) prostoru konečné dimenze N je pozitivně definitní právě tehdy, když $\text{in}(\mathbf{A}) = (N, 0, 0)$. To zajišťuje podmínka `if (k>0) & (z==0) & (d==0)`, pak následuje ověření zda je matice symetrická, čtvercová a reálná.

Zvolme $N = 5$ a matici $\mathbf{A}$ takto:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$ do Matlabu a použitím algoritmu 7

```
>> A = pascal(5);
>> cholesky(A)
```

dostáváme pro:

j = 1

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

j = 2

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

j = 3

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

j = 4

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

j = 5

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Zbývá ukázat, že platí $\mathbf{A} = \mathbf{U}^T\mathbf{U}$:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

8 QR rozklad Housholderovou metodou

K dané čtvercové matici $\mathbf{A}$ sestrojme unitární matici $\mathbf{Q}$ a horní trojúhelníkovou matici $\mathbf{U}$ tak, aby platilo $\mathbf{A} = \mathbf{QU}$. K nalezení matic $\mathbf{Q}$, $\mathbf{U}$ použijeme algoritmus 8 [1].

Algoritmus 8 qr_housholder.m

```
function qr_housholder(A)
% QR rozklad Housholderovou metodou

[M, N] = size(A);
H = eye(N);

if M == N
    for k = 1:N
        cit = A(k:N,k) + norm(A(k:N,k))*eye(N-k+1,1)*sign(A(k,k));
        jme = sqrt(2*(norm(A(k:N,k)) + norm(A(k,k)))*norm(A(k:N,k)));
        vk = cit/jme;
        Hdef = eye(N-k+1,N-k+1) - 2*vk*vk';
        Hk = [eye(k-1), zeros(k-1,N-k+1); zeros(N-k+1,k-1), Hdef];
        H = Hk*H;
        A = Hk*A;
    end
    U = A   % horni trojuhelnikova matice
    Q = H'  % unitarni matice
    Q*U     % A = Q*U
else
    disp('Zadana matice musi byt ctvercova.')
end
```

Zvolme $N = 4$ a matici $\mathbf{A}$ takto

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & -9 & 2 \\ 0 & 2 & -1 & 4 \\ -3 & 5 & 1 & 7 \\ 7 & 2 & 3 & 1 \end{bmatrix}$$

Zadáním matice $\mathbf{A}$ do Matlabu a použitím algoritmu 8

```
>> A = [1 4 -9 2; 0 2 -1 4; -3 5 1 7; 7 2 3 1];
>> qr_housholder(A)
```

dostáváme ($\mathbf{A}^{(0)} = \mathbf{A}$)

$$\mathbf{H}_1 = \begin{bmatrix} -0.1302 & 0 & 0.3906 & -0.9110 \\ 0 & 1.0 & 0 & 0 \\ 0.3906 & 0 & 0.8650 & 0.3149 \\ -0.9110 & 0 & 0.3149 & 0.2652 \end{bmatrix}, \qquad \mathbf{v}^{(1)} = \begin{bmatrix} 0.7517 \\ 0 \\ -0.2598 \\ 0.6062 \end{bmatrix},$$

$$\mathbf{A}^{(1)} = \begin{bmatrix} -7.681 & -0.3906 & -1.171 & 1.562 \\ 0 & 2.0 & -1.0 & 4.0 \\ -8.882 \cdot 10^{-16} & 6.517 & -1.705 & 7.151 \\ 6.661 \cdot 10^{-16} & -1.540 & 9.312 & 0.6470 \end{bmatrix},$$

$$\mathbf{H}_2 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & -0.2861 & -0.9325 & 0.2204 \\ 0 & -0.9325 & 0.3239 & 0.1598 \\ 0 & 0.2204 & 0.1598 & 0.9622 \end{bmatrix}, \qquad \mathbf{v}^{(2)} = \begin{bmatrix} 0.8019 \\ 0.5814 \\ -0.1374 \end{bmatrix},$$

$$\mathbf{A}^{(2)} = \begin{bmatrix} -7.681 & -0.3906 & -1.171 & 1.562 \\ 9.750 \cdot 10^{-16} & -6.988 & 3.929 & -7.671 \\ -1.813 \cdot 10^{-16} & 3.53 \cdot 10^{-16} & 1.868 & -1.310 \\ 4.991 \cdot 10^{-16} & -4.441 \cdot 10^{-16} & 8.468 & 2.647 \end{bmatrix},$$

$$\mathbf{H}_2 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & -0.2154 & -0.9765 \\ 0 & 0 & -0.9765 & 0.2154 \end{bmatrix}, \qquad \mathbf{v}^{(3)} = \begin{bmatrix} 0.7796 \\ 0.6263 \end{bmatrix},$$

$$\mathbf{A}^{(3)} = \begin{bmatrix} -7.681 & -0.3906 & -1.171 & 1.562 \\ 9.750 \cdot 10^{-16} & -6.988 & 3.929 & -7.671 \\ -4.483 \cdot 10^{-16} & 3.679 \cdot 10^{-16} & -8.671 & -2.302 \\ 2.845 \cdot 10^{-16} & -3.938 \cdot 10^{-16} & -1.554 \cdot 10^{-15} & 1.850 \end{bmatrix},$$

$$\mathbf{H}_4 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & -1.0 \end{bmatrix}, \qquad \mathbf{v}^{(4)} = 1.0,$$

$$\mathbf{A}^{(4)} = \begin{bmatrix} -7.681 & -0.3906 & -1.171 & 1.562 \\ 9.750 \cdot 10^{-16} & -6.988 & 3.929 & -7.671 \\ -4.483 \cdot 10^{-16} & 3.679 \cdot 10^{-16} & -8.671 & -2.302 \\ -2.845 \cdot 10^{-16} & 3.938 \cdot 10^{-16} & 1.554 \cdot 10^{-15} & -1.850 \end{bmatrix}$$

Horní trojúhelníková matice je tedy $\mathbf{U} = \mathbf{A}^{(4)}$, tj. :

$$\mathbf{U} = \begin{bmatrix} -7.681 & -0.3906 & -1.171 & 1.562 \\ 9.750 \cdot 10^{-16} & -6.988 & 3.929 & -7.671 \\ -4.483 \cdot 10^{-16} & 3.679 \cdot 10^{-16} & -8.671 & -2.302 \\ -2.845 \cdot 10^{-16} & 3.938 \cdot 10^{-16} & 1.554 \cdot 10^{-15} & -1.850 \end{bmatrix}$$

a ortogonální (unitární) matice $\mathbf{Q} = \mathbf{H}_1\mathbf{H}_2\mathbf{H}_3\mathbf{H}_4$ vypadá takto

$$\mathbf{Q} = \begin{bmatrix} -0.1302 & -0.5650 & 0.7995 & 0.1568 \\ 0 & -0.2861 & -0.01433 & -0.9581 \\ 0.3906 & -0.7372 & -0.5021 & 0.2277 \\ -0.9110 & -0.2352 & -0.3294 & 0.07518 \end{bmatrix}$$

Porovnáním součinu matic $\mathbf{Q}$ a $\mathbf{U}$ s maticí $\mathbf{A}$

$$\mathbf{QU} = \begin{bmatrix} 1.0 & 4.0 & -9.0 & 2.0 \\ 4.930 \cdot 10^{-32} & 2.0 & -1.0 & 4.0 \\ -3.0 & 5.0 & 1.0 & 7.0 \\ 7.0 & 2.0 & 3.0 & 1.0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 4 & -9 & 2 \\ 0 & 2 & -1 & 4 \\ -3 & 5 & 1 & 7 \\ 7 & 2 & 3 & 1 \end{bmatrix}$$

vidíme, že $\mathbf{A} \approx \mathbf{QU}$.

V následujícím odstavci ukážeme, jak vypadají jednotlivé prvky horní trojúhelníkove matice $\mathbf{U}$ při různých rozkladech. Zvolíme dvě čtvercové matice $\mathbf{A}$ a $\tilde{\mathbf{A}}$ obě řádu $N = 4$, kde matice $\tilde{\mathbf{A}}$ se liší od matice $\mathbf{A}$ pouze v prvku a_{11}. Aplikací výše uvedených rozkladů na matice $\mathbf{A}$, $\tilde{\mathbf{A}}$ dostaneme matice $\mathbf{U}$ a $\tilde{\mathbf{U}}$. Matice $\mathbf{U}$ odpovídá rozkladu matice $\mathbf{A}$ a matice $\tilde{\mathbf{U}}$ odpovídá rozkladu matice $\tilde{\mathbf{A}}$. Na další stránce můžeme porovnávat jak výsledky jednotlivých metod rozkladů mezi sebou, tak i změny prvků matice $\mathbf{U}$, změníme-li nepatrně původní prvek matice $\mathbf{A}$.

9 Porovnání matice U v závislosti na jednotlivých rozkladech

$$\mathbf{A} = \begin{bmatrix} 1 & 0.1 & 0.2 & 0.3 \\ 0.1 & 10 & 0.3 & 0.4 \\ 0.2 & 0.3 & 100 & 0.5 \\ 0.3 & 0.4 & 0.5 & 1000 \end{bmatrix} \qquad \tilde{\mathbf{A}} = \begin{bmatrix} 0.9 & 0.1 & 0.2 & 0.3 \\ 0.1 & 10 & 0.3 & 0.4 \\ 0.2 & 0.3 & 100 & 0.5 \\ 0.3 & 0.4 & 0.5 & 1000 \end{bmatrix}$$

LU rozklad

$$\mathbf{U} = \begin{bmatrix} 1.0 & 0.1000 & 0.2000 & 0.3000 \\ 0 & 9.990 & 0.2800 & 0.3700 \\ 0 & 0 & 99.95 & 0.4296 \\ 0 & 0 & 0 & 999.9 \end{bmatrix} \qquad \tilde{\mathbf{U}} = \begin{bmatrix} 0.9000 & 0.1000 & 0.2000 & 0.3000 \\ 0 & 9.989 & 0.2778 & 0.3667 \\ 0 & 0 & 99.95 & 0.4231 \\ 0 & 0 & 0 & 999.9 \end{bmatrix}$$

QR rozklad

$$\mathbf{U} = \begin{bmatrix} 1.068 & 1.199 & 19.09 & 281.4 \\ 0 & 9.940 & 1.040 & 6.724 \\ 0 & 0 & 98.16 & -49.18 \\ 0 & 0 & 0 & 958.3 \end{bmatrix} \qquad \tilde{\mathbf{U}} = \begin{bmatrix} 0.9747 & 1.303 & 20.89 & 308.2 \\ 0 & 9.928 & 0.6046 & 0.2598 \\ 0 & 0 & 97.79 & -60.21 \\ 0 & 0 & -8.882 \cdot 10^{-16} & 949.4 \end{bmatrix}$$

Choleského rozklad

$$\mathbf{U} = \begin{bmatrix} 1.0 & 0.1000 & 0.2000 & 0.3000 \\ 0 & 3.162 & 0.08862 & 0.1171 \\ 0 & 0 & 9.998 & 0.04297 \\ 0 & 0 & 0 & 31.62 \end{bmatrix} \qquad \tilde{\mathbf{U}} = \begin{bmatrix} 0.9486 & 0.1054 & 0.2108 & 0.3162 \\ 0 & 3.160 & 0.08790 & 0.1161 \\ 0 & 0 & 9.997 & 0.04232 \\ 0 & 0 & 0 & 31.62 \end{bmatrix}$$

QR rozklad pomocí Housholderovy metody

$$\mathbf{U} = \begin{bmatrix} -1.068 & -1.199 & -19.09 & -281.4 \\ -2.628 \cdot 10^{-17} & -9.940 & -1.040 & -6.724 \\ -4.509 \cdot 10^{-17} & 1.632 \cdot 10^{-17} & -98.16 & 49.18 \\ -5.605 \cdot 10^{-17} & 3.830 \cdot 10^{-19} & 4.441 \cdot 10^{-16} & -958.3 \end{bmatrix}$$

$$\tilde{\mathbf{U}} = \begin{bmatrix} -0.9747 & -1.303 & -20.89 & -308.2 \\ -3.489 \cdot 10^{-17} & -9.928 & -0.6046 & -0.2598 \\ -7.59 \cdot 10^{-17} & 3.675 \cdot 10^{-17} & -97.79 & 60.21 \\ -1.467 \cdot 10^{-18} & -2.669 \cdot 10^{-17} & -4.441 \cdot 10^{-16} & -949.4 \end{bmatrix}$$

10 Literatura

[1] MÍKA, S. *Numerická analýza.* Učební text ZČU. Plzeň, 2005.

[2] MÍKA, S. - BRANDNER, M. *Numerické metody I.* Skripta ZČU. Plzeň, 2000.

[3] TESKOVÁ, L. *Lineární algebra.* Skripta ZČU. Plzeň, 2001.

Internet:

[4] `www.math.siu.edu/matlab`