MRMMI

Matěj Popďakunik

21. prosince 2023

Obsah

1 Reprezentace řídkých matic v počítači **2**
1.1 Souřadnicový formát . . . 2
1.2 CSR formát (Compressed Sparse Rows) . . . 3
1.3 MSR formát (Modified Sparse Rows) . . . 3
1.4 Ellpack - Hpack . . . 3
1.5 Diagonální uložení . . . 4

2 Přímé metody pro řešení soustav lineárních algebraických rovnic **4**
2.1 Odvození Choleského rozkladu . . . 4
2.2 Submaticová metoda . . . 4
2.3 Sloupcová metoda . . . 5
2.4 Popis struktury řídkých matic . . . 5
2.5 Vznik zaplnění . . . 6
2.6 Eliminační strom . . . 6
2.6.1 Vlastnosti eliminačních stromů . . . 7
2.6.2 Konstrukce eliminačního stromu . . . 8

3 Řešení rovnic s řídkou SPD maticí **8**
3.1 Algoritmy pro vytvoření pásu, respektive profilu . . . 9
3.1.1 Algoritmus CMK (Cuthill, McKee) . . . 9
3.1.2 Algoritmus GPS . . . 10
3.1.3 Algoritmus RCM (Reversed Cuthill McKee) . . . 10
3.2 Uspořádání minimalizující zaplnění . . . 10
3.2.1 Algoritmus minimálního stupně (Minimum degree ordering) . . . 10
3.3 Uspořádání přizpůsobující A počítačové architektuře . . . 11
3.3.1 Frontální metoda . . . 11

4 Poznámky k obecnějším systémům **12**
4.1 Symetrické indefinitní matice . . . 12
4.2 Předpodmiňování metody složených gradientů . . . 14
4.3 Rychlost konvergence PCG . . . 14
4.4 Testovací případ . . . 15
4.4.1 Metoda prosté iterace . . . 15
4.4.2 Richardson . . . 15
4.4.3 Jacobi . . . 15
4.4.4 Gauss-Seidel a SOR(w) . . . 15

5 Neúplné LU rozklady (ILU - incomplete LU decomposition) **16**
5.1 ILU(k) . . . 16
5.1.1 Grafová interpretace ILU(k) . . . 17
5.2 Další varianty . . . 17

Budeme řešit $\mathbf{A}x = b$, kde $\mathbf{A}$ je řídká.

Definice 0.1. Matice $\mathbf{A} \in \mathbb{R}^{n,n}$ je řídká $\Leftrightarrow$ obsahuje málo nenulových prvků, např $O(n)$

Definice 0.2. Matice $\mathbf{A}$ je řídká $\Leftrightarrow$ jsme schopni využít znalosti pozic nenulových prvků k účinnému řešení soustavy.

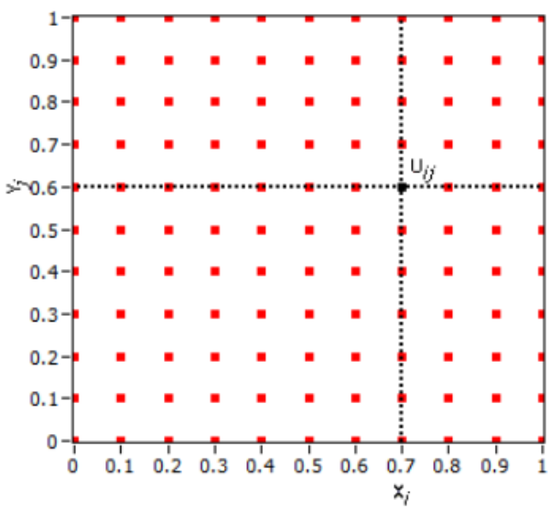
Definice 0.3. Matici která není řídká se nazývá hustá.

Poznámka. Řídké matice přirozeným způsobem vznikají při diskretizace PDR metodou sítí, konečných objemů, konečných prvků, atd.

Příklad. Řešme metodou konečných diferencí Poissonovu rovnici ve 2D

$$-\Delta u = f, \text{ na } \Omega = (0,1) \times (0,1) \tag{1}$$
$$u|_{\partial\Omega} = 0$$

Diskretizací dostaneme obrázek



Diskrerizace

Počet uzlů na ose x označíme n_x, na ose y n_y.
Pak dostaneme $h_x = \frac{1}{n_x+2}, h_y = \frac{1}{n_y+2}$

Diskretizací (1) pak dostáváme

$$-\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} - \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = f_{i,j} \tag{2}$$

Předpokládáme $n_x = n_y \implies h_x = h_y := h$ a dostaneme

$$\frac{1}{h^2}\begin{pmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 \end{pmatrix} = \dots$$

Ale pokud například $n_x = n_y = 1000 \implies n = 10^6$ vrcholů.
Při hustém uložení n^2 prvků 10^{12}, pokud 8 bajtů na prvek $\implies$ 8TB dat.

Vkládáme-li pouze nenulové prvky (tj. jen 5 diagonál kterých vzniká) $\approx 5 * n$ prvků po 8B $\implies$ 40MB dat

1 Reprezentace řídkých matic v počítači

1.1 Souřadnicový formát

- 1 pole reálných čísel délky n_z (počet nenulových prvků v matici), obsahující hodnoty nenulových prvků matice $\mathbf{A}$ v libovolném pořadí.
- 2 pole celých čísel obsahující řádkové, resp. sloupcové indexy nenulových prvků.

Příklad.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix} \tag{3}$$

Pak dostáváme následující 3 pole

$$A\mathbf{A} = (1,2,3,4,5,6,7,8,9,10,11,12)$$
$$I\mathbf{A} = (1,1,2,2,2,3,3,3,3,4,4,5)$$
$$J\mathbf{A} = (1,4,1,2,4,1,3,4,5,3,4,5)$$

Lze vylepšit pokud ukládáme prvky systematicky (například po řádkách), pak $I\mathbf{A}$ obsahuje redundantní informaci.

1.2 CSR formát (Compressed Sparse Rows)

- $A\mathbf{A}$ pole reálných čísel délky n_z obsahující nenulové prvky matice $\mathbf{A}$ zapsané po řádcích.
- $J\mathbf{A}$ pole celých čísel délky n_z, obsahující sloupcové indexy prvků v $A\mathbf{A}$
- $I\mathbf{A}$ pole délky $n+1$ (počet řádků +1), kde i-tý prvek udává index začátku i-tého řádku matice $\mathbf{A}$ v polích $A\mathbf{A}$ a $J\mathbf{A}$

Příklad. Vezměme matici z (3), pak dostaneme

$$A\mathbf{A} = (1,2,3,4,5,6,7,8,9,10,11,12)$$
$$J\mathbf{A} = (1,4,1,2,4,1,3,4,5,3,4,5)$$
$$I\mathbf{A} = (1,3,6,10,12,13)$$

Poslední člen v $I\mathbf{A}$ (zde 13) je umělý ukazatel za konec $A\mathbf{A}$ a $J\mathbf{A}$

Výhody:

- Šetří paměť
- Cache hits

Nevýhody:

- Špatně se přidávají nenulové prvky

Existuje i alternativa, kdy matici ukládáme po sloupcích, tzv. CSC formát.

1.3 MSR formát (Modified Sparse Rows)

Často jsou diagonální prvky $\mathbf{A}$nenulové a jsou potřeba častěji než ostatní prvky, proto je uložíme zvlášť.

- $A\mathbf{A}$ pole reálných čísel délky n_z+1 obsahující nenulové prvky matice $\mathbf{A}$ zapsané po řádcích.
- $J\mathbf{A}$ pole celých čísel délky n_z+1, obsahující sloupcové indexy prvků v $A\mathbf{A}$

Příklad. Vezměme opět matici z (3), pak dostaneme

$$A\mathbf{A} = (1,4,7,11,12,\star,2,3,5,6,8,9,10)$$
$$J\mathbf{A} = (7,8,10,13,14,\mathbf{14},4,1,4,1,4,5,3)$$

V $A\mathbf{A}$ jsou tedy nejprve zapsané všechny diagonální prvky, jedno volné místo ($\star$)
a ostatní nenulové prvky zapsané po řádcích

V $J\mathbf{A}$ máme před **14** indexy začátků řádků v poli $A\mathbf{A}$, **14** je umělý ukazatel
za konec pole $A\mathbf{A}$, za máme sloupcové indexy nediagonálních prvků

1.4 Ellpack - Hpack

Pro matice, které mají omezený počet nenulových prvků na 1 řádek, maximálně nějaké N_d, uvažujeme malé.

- $COEF$ pole reálných čísel délky $n * N_d$ obsahuje na i-tém řádku (pokud máme 2D pole) hodnoty nenulových prvků i-tého řádku matice $\mathbf{A}$.
- $ICOEF$ obsahuje sloupcové indexy odpovídajících prvků v poli $COEF$

Příklad.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 2 & 0 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 0 & 6 & 7 & 0 & 8 \\ 0 & 0 & 9 & 10 & 0 \\ 0 & 0 & 0 & 11 & 12 \end{pmatrix} \tag{4}$$

Vezměme $N_d = 3$, pak máme

$$COEF = \begin{pmatrix} 1 & 2 & \cdot \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & \cdot \\ 11 & 12 & \cdot \end{pmatrix}, ICOEF = \begin{pmatrix} 1 & 3 & \cdot \\ 1 & 2 & 4 \\ 2 & 3 & 5 \\ 3 & 4 & \cdot \\ 4 & 5 & \cdot \end{pmatrix}$$

1.5 Diagonální uložení

Jedná se tedy o matice v nichž se nenulové prvky nachází jen na několika málo diagonálách.

Nenulové diagonály matice **A**jsou uloženy v poli $DIAG$ velikosti $n * N_diag$.
Offsety každé diagonály jsou v poli $IOFF$, což je celočíselné pole délky N_diag
$DIAG[i,j]$ obsahuje $\mathbf{A}_{i,i+IOFF[j]}$

Příklad. Příklad pro matici (4)

$$DIAG = \begin{pmatrix} 1 & \star & 2 \\ 4 & 3 & 5 \\ 7 & 6 & 8 \\ 10 & 9 & \star \\ 12 & 11 & \star \end{pmatrix}, IOFF = (0, -1, 2)$$

2 Přímé metody pro řešení soustav lineárních algebraických rovnic

Řešíme $\mathbb{A}x = b$, kde **A**je SPD matice.

Obecně: Řešení $\mathbb{A}x = b$ GEM $\Leftrightarrow$ přenásobení rozšířené matice soustavy maticemi elementárních úprav.

$$\mathbb{L}_{n-1} \dots \mathbb{L}_2 \mathbb{L}_1 \mathbb{A} = \tilde{\mathbb{L}}\mathbb{A} = \mathbb{U}$$

$\mathbb{U}$ je matice v horním trojúhelníkovém tvaru, $\mathbb{L}$ je v dolním trojúhelníkovém tvaru.

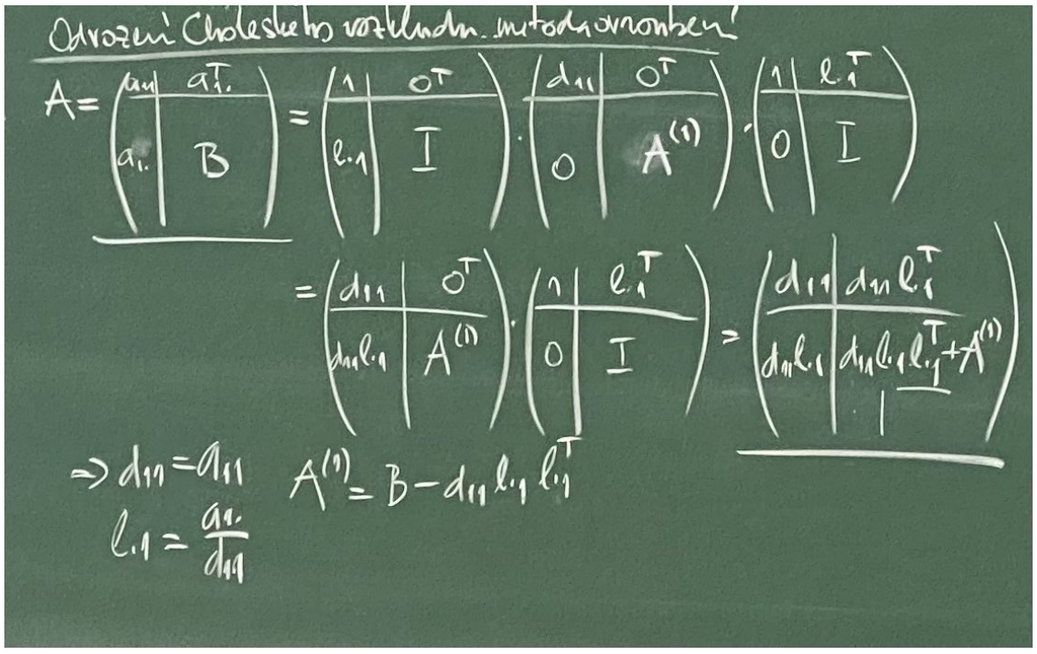
$$\tilde{\mathbb{L}}\mathbb{A} = \mathbb{U} \Leftrightarrow \mathbb{A} = \tilde{\mathbb{L}}^{-1}\mathbb{U}$$

GEM je ekvivalentní konstrukci LU rozkladu matice **A**.
Algoritmus:

1. Kostrukce $\mathbb{L}, \mathbb{U}$ tak, že $\mathbb{A} = \mathbb{L}y$, $O(n^3)$
2. $\mathbb{L}y = b$, $O(n^2)$ - řeší zpětný chod
3. $\mathbb{U}x = y$, $O(n^2)$ - řeší zpětný chod

LU rozklad existuje $\Leftrightarrow$ **A**je silně negativní, pokud není je třeba přehazovat řádky a sloupce matice **A**
SPD matice je silně negativní $\implies$ přehazovat řádky a sloupce není třeba.
Pro **A**SPD, $\mathbb{U} = \mathbb{L}^T$
$\mathbb{A} = \mathbb{L}\mathbb{L}^T$, kde $\mathbb{L}$ je v dolním trojúhelníkovém tvaru provedeme Choleského rozklad
$\mathbb{A} = \mathbb{L}\mathbb{D}\mathbb{L}^T$ - bezodmocninová metoda

2.1 Odvození Choleského rozkladu



2.2 Submaticová metoda

Right looking approach (k,i,j varianta)

```
for k  = 1 ... n
    d_{kk} = a_{kk}
    for i = k+1 ... n
        l_{ik}  = a_{ik}/d_{kk}
    end
    for i = k+1 ... n
        for j = k+1 ... n
            a_{ij} = a_{ij} - l_{ik} a_{kj}
        end j
    end i
end k
```

2.3 Sloupcová metoda

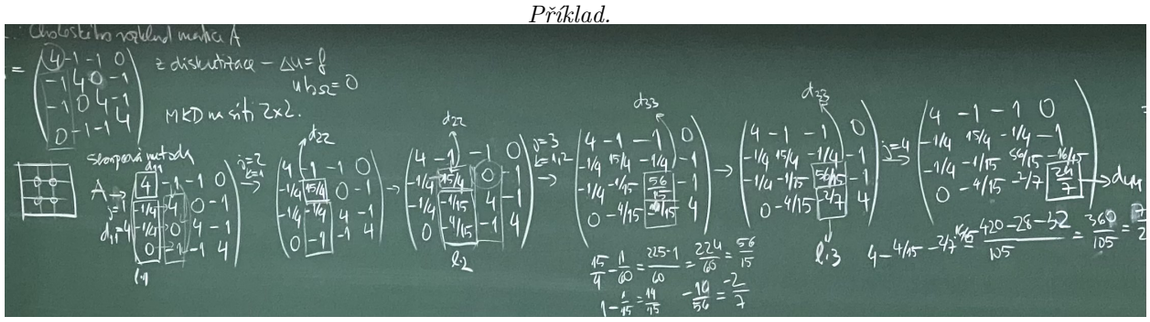
Left looking approach (j,k,i varianta)

```
for j = 1 ... n
    for k = 1 ... j-1
        for i = k+1 ... n
            a_{ij} = a_{ij} - l_{ik}a_{kj}
        end i
    end k
    d_{jj} = a_{jj}
    for i = j+1 ... n
        l_{ij} = a_{ij} / d_{jj}
    end i
end j
```

Poznámka. Obě metody lze vylepšit, stačí provádět cykly pouze přes nenulové prvky

Poznámka. Pro hustou matici lze výpočet provádět přímo v matici $\mathbf{A}$

Příklad.



Z toho dostáváme

$$\mathbb{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 & 0 \\ -\frac{1}{4} & -\frac{1}{15} & 1 & 0 \\ 0 & -\frac{4}{15} & -\frac{2}{7} & 1 \end{pmatrix}, \mathbb{D} = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & \frac{15}{4} & 0 & 0 \\ 0 & 0 & \frac{56}{15} & 0 \\ 0 & 0 & 0 & \frac{24}{7} \end{pmatrix}$$

a platí $\mathbb{L}\mathbb{D}\mathbb{L}^T = \mathbb{A}$.
Pro řešení $\mathbb{A}x = b$ řešíme $\mathbb{L}y = b, \mathbb{D}z = y, \mathbb{L}^T x = z$

Poznámka. $l_{32} \neq 0$ ačkoliv $a_{32} = 0$, na pozici 3,2 došlo k zaplnění.

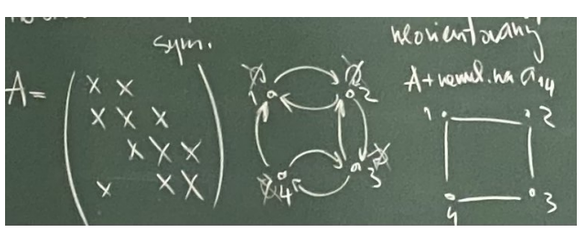
Faktor $\mathbb{L}$ má obecně více nenulových prvků než $\mathbf{A}$: $\mathbf{A}$je řídká $\not\Longrightarrow$ $\mathbb{L}$ je řídká, $\mathbb{L}$,může být i kompletně zaplněná.
Datové struktury neumožňují snadno přidat nový nenulový prvek, dynamické datové struktury jsou drahé $\Longrightarrow$ je třeba umět předpovědět množství a polohy nenulových prvků, tj. určit strukturu $\mathbb{L}$ na základě struktury $\mathbf{A}$.

2.4 Popis struktury řídkých matic

Matici $\mathbb{A} \in \mathbb{R}^{n,n}$ přiřadíme graf $G(\mathbb{A}) = (V, E)$, $V = \hat{n}$.

$$E \subset V \times V \ldots (i,j) \in E \Leftrightarrow a_{ij} \neq 0$$

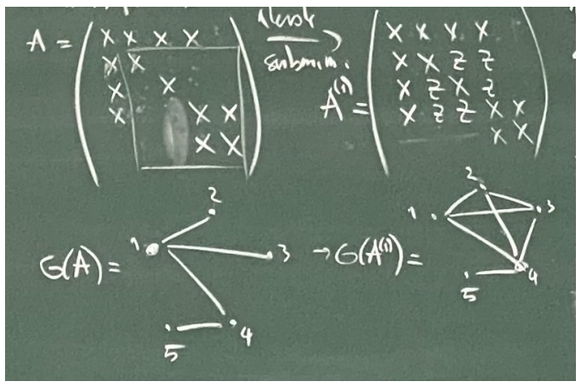
Pro strukturně nesymetrické matice je $G(\mathbb{A})$ orientovaný.



Poznámka. Grafy matic pocházející z diskretizací PDR (MKD, MKO, MKP) často jsou totožné s grafem sítě.

2.5 Vznik zaplnění

Nechť **A**je SPD



1 sloupec "otiskne"svoji strukturu do všech sloupců j, pro které $a_{1j} \neq 0$.
Po provedení 1 kroku submaticového algoritmu se incidentní vrcholy pivota propojí do kliky (každý s každým), tedy vzniká zaplnění.

Lemma 2.1 (O zaplnění)**.** Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$je SPD, $i, j \in \hat{n}, i > j, k \in \hat{n}$.
Potom $a_{ij}^{(k)} \neq 0 \Leftrightarrow a_{ij}^{(k-1)} \neq 0$ nebo $a_{ik}^{(k-1)} \neq 0 \wedge a_{kj}^{(k-1)} \neq 0$

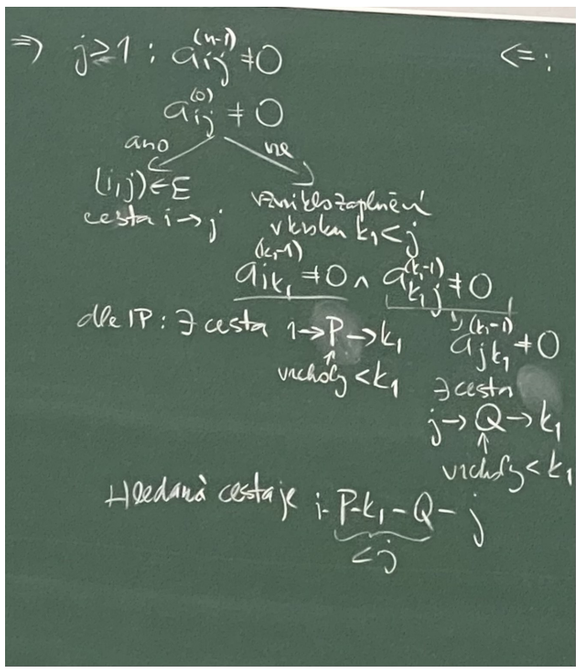
Důkaz. Aby prvek v kroku k byl nenulový, tak musel buď být nenulový v minulém kroku nebo vznikne násobením právě těchto dvou prvků. □

Věta 2.2 (O zaplnění)**.** Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$je SPD,$i, j \in \hat{n}, i > j$.
Potom $a_{ij}^{(n-1)} \neq 0 \Leftrightarrow \exists$ cesta v $G(\mathbf{A})$, $i, p_1, p_2 ... p_t, j$ kde $\forall k \in \hat{t}, p_k < j$.

Důkaz. ($\Longrightarrow$)

Dokazujeme indukcí podle j:
$j = 1$:$a_{ij}^{(n-1)} \neq 0 \Leftrightarrow a_{ij}^{(0)} \neq 0 \Leftrightarrow (i, 1) \in E$.

Předpokládáme, že platí pro $j - 1$.



($\Longleftarrow$)

Při eliminaci například p_1 z $G(\mathbf{A})$ vznikne hrana $i \to p_2$. Při eliminaci vrcholu se cesta z $i \to j$ vedoucí přes $p_1, ..., p_t$ zkrátí alespoň o 1.

$p_1 ... p_t$ se eliminují před j a i. Po eliminaci posledního z $p_1, ..., p_t$ se i a j spojí hranou

Z toho víme, že na pozici $a_i j$ vzniká zaplnění $\implies a_{ij}^{(n-1)} \neq 0$. □

2.6 Eliminační strom

Eliminační strom matice **L** (pocházející z Choleského faktorizace SPD matice $\mathbf{A} \in \mathbb{R}^{n \times n}$) je $T = (V, E_T)$, kde $V = \hat{n}$ a předek $[j] = \min\{i \in \hat{n} \mid e_{ij} \neq 0, i > j\}$, tj. $(i, j) \in E_T \Leftrightarrow e_{ij} \neq 0 \wedge e_{kj} = 0$

$$\mathbf{A} = \begin{bmatrix} \times & & & \times & \\ & \times & & & \times \\ & & \times & \times & \times \\ \times & & \times & \times & \\ & \times & \times & & \times \end{bmatrix} \to \mathbf{L} + \hat{\mathbf{L}} = \begin{bmatrix} \times & & & \times & \\ & \times & & & \times \\ & & \times & \times & \times \\ \times & & \times & \times & z \\ & \times & \times & z & \times \end{bmatrix}$$

Eliminační stromy se definují na základě struktury matice **L**, ne **A**.

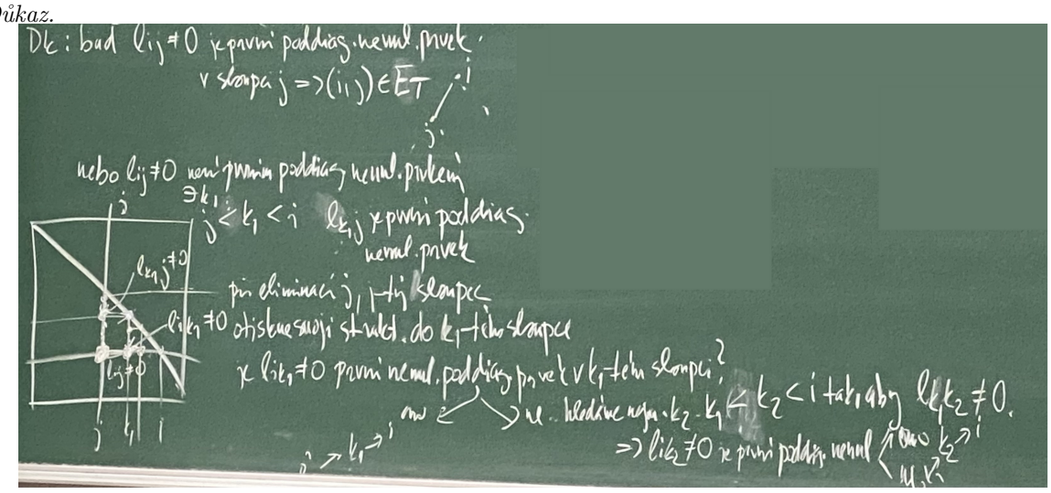
2.6.1 Vlastnosti eliminačních stromů

Tvrzení 2.1. Kořenem eliminačního strom je vrchol s největším indexem

Tvrzení 2.2. i je předkem j v $T \implies i > j$

Tvrzení 2.3. $e_{ij} \neq 0$ pro $i > j \implies i$ je předkem j v T

Důkaz.



□

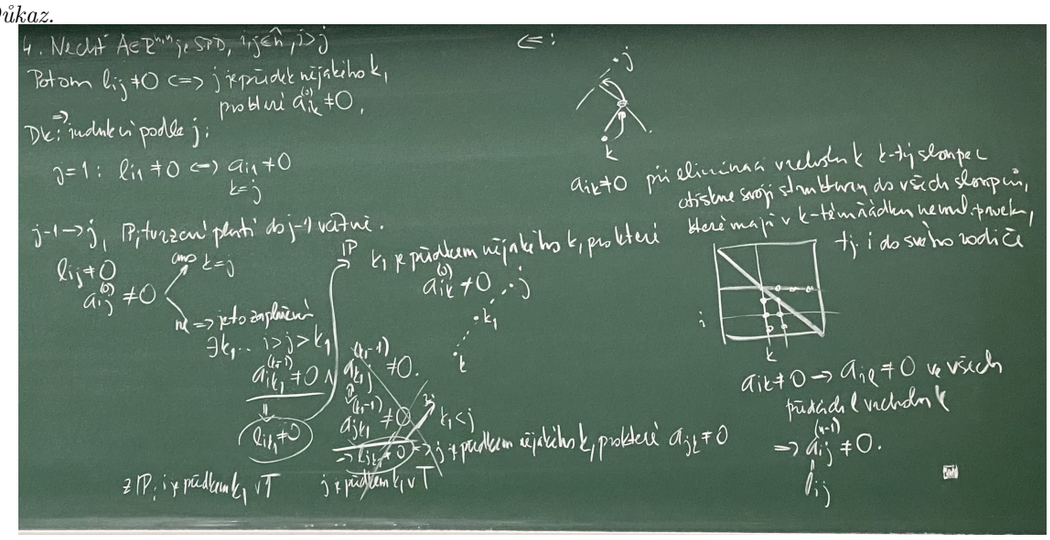
Poznámka. Tuto implikaci nelze obrátit.

Tvrzení 2.4. Jsou-li T_i a T_j disjunktní podstromy eliminačního stromu s kořeny ve vrcholech i, respektive j, pak $\forall r \in T_i$ a $s \in T_j$ platí $e_{rs} = 0$

Důkaz. Negace předchozího tvrzení. □

Tvrzení 2.5. Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$je SPD, $i, j \in \hat{n}, i > j$.
Potom $e_{ij} \neq 0 \Leftrightarrow j$ je předek nějakého k, pro dané $a_{ik}^{(0)} \neq 0$.

Důkaz.



□

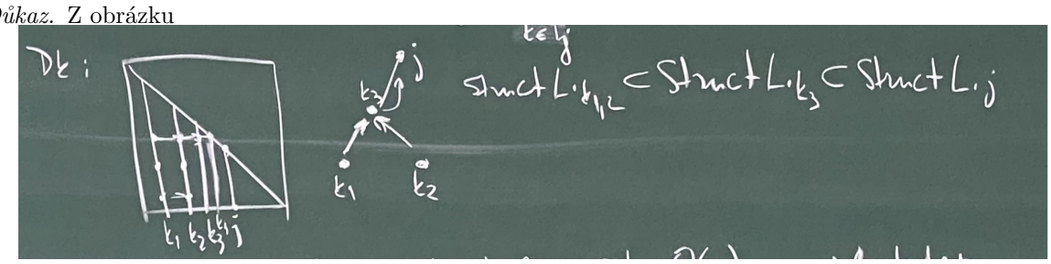
Důsledek 2.2.1. Strukturu matice $\mathbf{L}$lze odvodit z eliminačního stromu a struktury matice $\mathbf{A}$.

Nenulové prvky v i-tém řádku matice $\mathbf{L}$(struktura i-tého řádku) jsou právě ty prvky v T_i ležící na cestě mezi i a nějakým k, pro které $a_{ik} \neq 0$.

Definice 2.1. $Struct(\mathbf{L}, j) - \{i \in \hat{n} \mid e_{ij} \neq 0, i > j\}$

Věta 2.3. $Struct(L, j) = Struct(A, j) \cup \bigcup_{k \in T_j} Struct(L, k) \setminus \widehat{j-1}$

Důkaz. Z obrázku



□

$\implies$ strukturu sloupců matice $\mathbf{L}$lze určit v $O(m)$ operacích, kde m je počet nenulových prvků matice $\mathbf{L}$.

2.6.2 Konstrukce eliminačního stromu

Z prvků matice **A**.
Pseudokód:

```
for i = 1 ... n
    předek[i] = 0
    for k in {Soused i| index k< index i}
        r=k
        while(předek[r]!= 0 and předek[r]!= i)
            r = předek[r]
        end while
        if(předek[r]==0) předek[r] = 1
    end k
end i
```

Příklad.

$$\mathbb{A} = \begin{pmatrix} \times & & & \times & & \\ & \times & & & & \times \\ & & & \times & \times & \times \\ \times & & & \times & \times & \\ & \times & \times & & & \times \end{pmatrix}$$

3 Řešení rovnic s řídkou SPD maticí

Řešení $\mathbb{A}x = b$ s řídkou (SPD) maticí.

1. Konstrukce eliminačního stromu
2. Výpočet struktur řádků (sloupců) matice $\mathbb{L}$
3. Alokace paměti, vytvoření datových struktur $\mathbb{L}$
4. Výpočet Choleského faktorizace $\mathbb{A} = \mathbb{L} \cdot \mathbb{L}^T$, respektive $\mathbb{A} = \mathbb{L} \cdot \mathbb{D} \cdot \mathbb{L}^T$
5. 2 zpětné chody: $\mathbb{L}y = b, \mathbb{L}^T x = y$.

V kroku 1-3 provádíme symbolickou faktorizaci, pracujeme se strukturou matice **A**. V kroku 4 a 5 provádíme numerickou faktorizaci.

Příklad. Zkoumejme vznik zaplnění u matice

$$\mathbb{A} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix}$$

Po provedení jednoho kroku submaticového algoritmu dostaneme

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix}$$

tj. kompletně zaplněnou matici.

Pokud ale prohodíme první a poslední řádek a sloupec matice **A**, tj.

$$\begin{pmatrix} \times & & & & & \times \\ & \times & & & & \times \\ & & \times & & & \times \\ & & & \times & & \times \\ & & & & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix}$$

tak v této matici zaplnění nevznikne.

Pokud chceme minimalizovat zaplnění, je možné permutovat řádky a sloupce matice **A**.

Vezměme permutační matici $\mathbb{P}$, pak místo $\mathbb{A}x = b$ řešíme

$$\mathbb{P}\mathbb{A}\mathbb{P}^T y = \mathbb{P}b$$

Omezujeme se na symetrické permutace $\mathbb{A} = \mathbb{A}^T \implies \tilde{\mathbb{A}} = \mathbb{P}\mathbb{A}\mathbb{P}^T$ je také symetrická. Diagonální prvky matice **A**zůstávají na diagonále.

Ukažme souvislost s grafy matic. Nechť $\tilde{\mathbb{A}} = \mathbb{P}\mathbb{A}\mathbb{P}^T$. Pak platí

$\tilde{a_{ij}} = a_{\varpi(i)\varpi(j)}$

$$\implies (i,j) \in G(\tilde{\mathbb{A}}) \Leftrightarrow (\varpi(i), \varpi(j)) \in G(\mathbb{A})$$

$$\implies \text{Grafy } G(\mathbb{A}) \text{ a } G(\tilde{\mathbb{A}}) \text{ se liší jen číslováním vrcholů.}$$

Číslování vrcholů určuje pořadí v eliminaci, tím vznikají rozdílné zaplnění.
Jak zvolit $\mathbb{P}$, aby se s maticí $\tilde{\mathbb{A}} = \mathbb{P}\mathbb{A}\mathbb{P}^T$ lépe pracovalo? Například aby

1. Matice $\tilde{\mathbb{A}}$ mělo lepší strukturu než $\mathbb{A}$
2. $\mathbb{P}$ minimalizovala zaplnění
3. $\tilde{\mathbb{A}}$ byla přizpůsobena počítačové architektuře (například možnost paralelizace)

3.1 Algoritmy pro vytvoření pásu, respektive profilu

Označme $\forall i \in \hat{n} : f_i = \min\{j \in \hat{n} | a_{ij} \neq 0\}$ toto označuje pozici 1. nenulového prvku v i-tém řádku.
Označme $\delta_i = i - f_i$, to nazvěme šířkou profilu v i-tém řádku
Profilem **A**pak nazveme

$$Profil(\mathbb{A}) = \{(i,j) \in \hat{n} \times \hat{n} | i \geq j \geq i - \delta_i\}$$

Dále označme šířku pásu jako $\delta = \max_i \delta_i$.
Nakonec označme jako pás

$$Pas(\mathbb{A}) = \{(i,j) \in \hat{n} \times \hat{n} | i \geq j \geq i - \delta\}$$

Pro ilustraci
Zaplnění vzniká jen uvnitř profilu.

Příklad. Sousedi mají mít co nejbližší indexy, problém je u nepravidelných oblastí

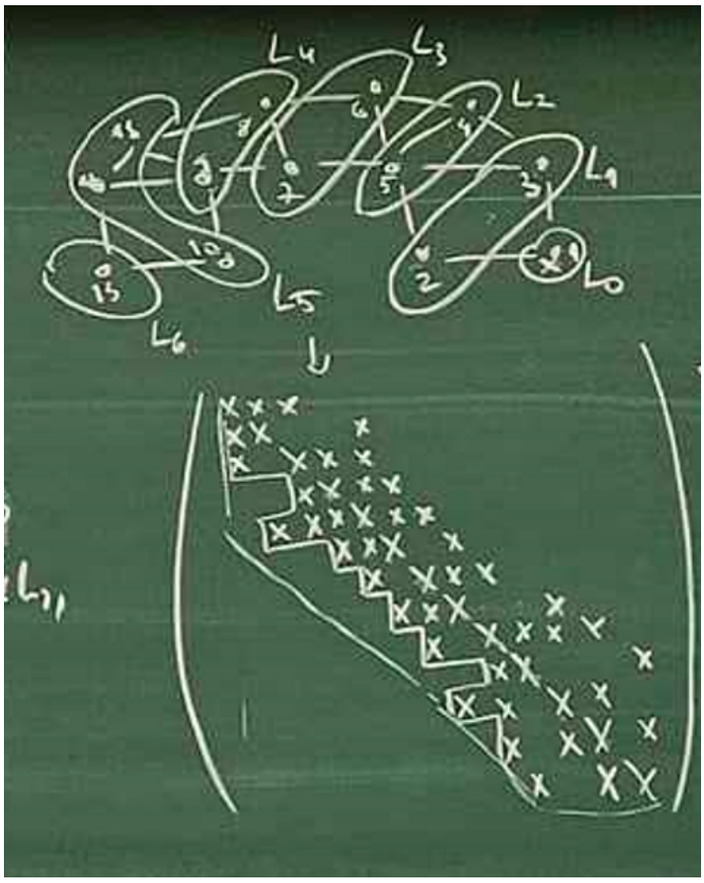
3.1.1 Algoritmus CMK (Cuthill, McKee)

Číslování podle úrovní

1. Zvol počáteční vrchol x, $L_0 = \{x\}$
2. $L_1 = Adj(x)$
3. $L_2 = Adj(L_1) \setminus (L_1 \cup \{x\})$

k. $L_{k-1} = Adj(L_{k-2}) \setminus \bigcup_{j=0}^{k-1} L_j$

End Každý vrchol $G(\mathbb{A})$ je v některé z množin $L_1, L_2, \ldots, L_p$, pak vrcholy čísluji nejprve v L_0, pak L_1, atd.



Příklad.

Sousedící prvky se liší maximálně o 1 úroveň, dostávají čísla po sobě, proto se vyváří pás.

Vzniká otázka jak zvolit počáteční vrchol.

Aby byl pás úzk, je třeba volit x tak, aby vzniklo co nejvíce úrovní.

Průměr grafu označme $d(G)$, označujeme tak největší vzdálenost mezi 2 vrcholy v $G(A)$ měřenou délkou nejkratší cesty.

Vrcholy, mezi kterými existuje nejkratší cesta délky $d(G)$ se nazývají periferní, ty lze najít v $O(n^3)$ operací, to je složité proto hledáme pseudoperiferní vrcholy.

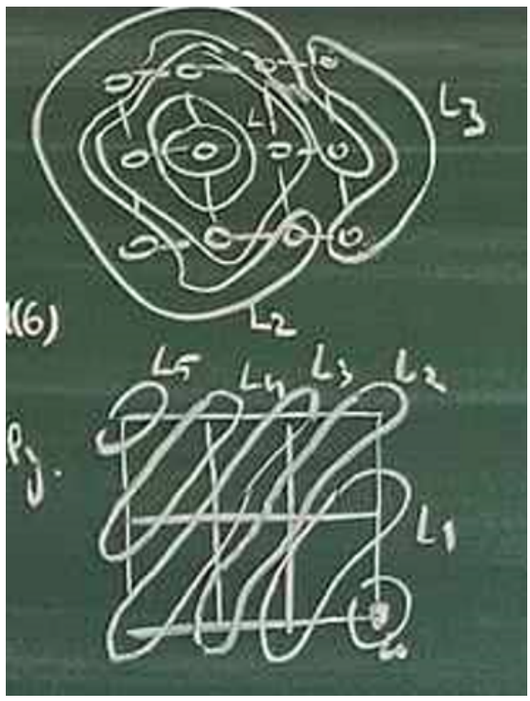
3.1.2 Algoritmus GPS

Nechť x_0 je startovní vrchol. Z CMK dostaneme $x_0 \to L_0, L_1, \ldots, L_{K_1}$.

Zvolíme $x_1 \in L_{K_1}$, z CMK opět $x_1 \to \tilde{L_0}, \tilde{L_1}, \ldots, \tilde{L_{K_2}}$.

Pokud $K_2 > K_1$ (tedy máme více úrovní) tak zvolme $x_2 \in \tilde{L_{K_2}}$ a proces opakujeme, jinak algoritmus skončí a x_1 je pseudoperiferní vrchol.

Ukažme si ilustrační obrázek



Příklad.

Poznámka. Bylo zjištěno, že se někdy vyplatí po použití algoritmu CKM číslovat vrcholy opačně. Z toho plyne následující algoritmus

3.1.3 Algoritmus RCM (Reversed Cuthill McKee)

1. Najdi startovací vrchol x_1 pomocí GPS
2. Pro $i = \hat{n}$, najdi všechny neočíslované sousedy x_i a očísluj je (vzestupně podle stupně)
3. Otoč číslování $y_i = x_{n+1-i}$

Otočením číslování se někdy vylepší profil ($\sum_{i=1}^{n} \delta_i$).
Šířka pásma se nezmění, dá se ukázat, že $\sum_{i=1}^{n} \delta_i$ se nezhorší.

3.2 Uspořádání minimalizující zaplnění

Optimální by bylo najít permutovanou matici

$$\tilde{\mathbb{P}}, |Struct(L_{\tilde{\mathbb{P}}\mathbb{A}\tilde{\mathbb{P}}^T}) = \min_{Permutace} |Struct(L_{\mathbb{P}\mathbb{A}\mathbb{P}^T})|$$

Na toto není znám účinný algoritmus, pro nesymetrické matice je to NP-úplný problém.

Díky tomu hledáme "dostatečně dobrá"uspořádání, ne nutně optimální-

3.2.1 Algoritmus minimálního stupně (Minimum degree ordering)

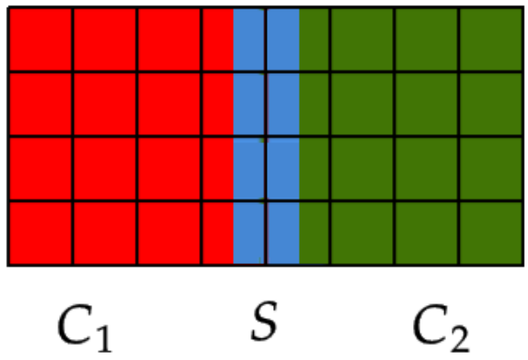
1. $G_0 = G(A), A_0 = A, i = 1$
2. V G_i najdi vrchol s nejnižším stupněm a očísluj ho $\to i$
3. Sestav G_i eliminací i z grafu G_{i-1}
4. Dokud $i < n$, pak jdi na 2) jinak konec

Hlavní myšlenka algoritmu je, že vrcholy s malým stupněm způsobí malé zaplnění.

Poznámka. Může se stát, že vrcholů s minimálním stupněm je více → různé verze algoritmu

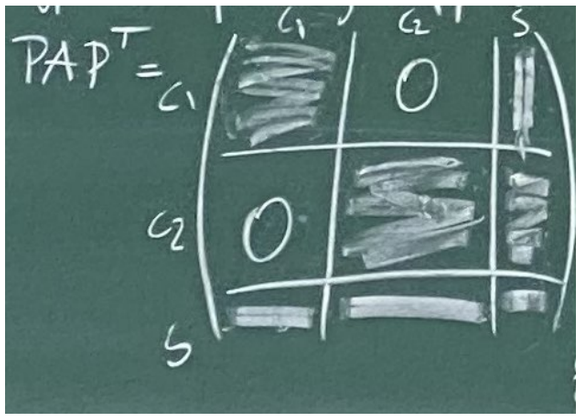
Poznámka. Algoritmus AMD (aproximate minimum degree) - stupně vrcholů se nepočítají přesně pouze se odhadují

3.3 Uspořádání přizpůsobující A počítačové architektuře

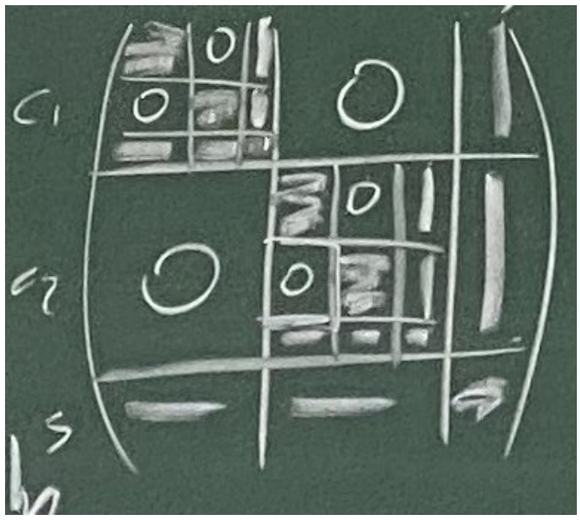


S označuje takzvaný separátor

Rozdělím vrcholy v $G(A)$ na 2 komponenty (přibližně stejně velké) a separátor (malý).
Nejprve očíslujeme vrcholy v C_1, pak v C_2 a nakonec v S
Matice pak vypadá následovně:



Obroubený pásový tvar.
Nulové bloky zůstanou při Choleského faktorizaci zachovány
Diagonální bloky lze faktorizovat paralelně .
Výhodné pokud komponenty jsou velké a separátor malý.
Postup lze rekurentně použít na C_1 a C_2 → Metoda vnořených řezů (nested disection), viz obrázek.



3.3.1 Frontální metoda

Uspořádání A lze hledat za běhu Choleského faktorizace
V každém kroku permutujeme řádky a sloupce matice **A**tak, aby se nenulové prvky nahrnuly k diagonále.

U pásových matic se eliminace provádí v malých hustých maticích, nic se nemusí přehazovat. Tato metoda má nevýhodu, permutace za běhu je drahá. Její hlavní výhoda je možnost konstruovat Choleského rozklad matice **A**i když **A**není celá složená.

Toto má aplikaci v MKP, **A**se skládá z příspěvků jednotlivých elementů, nemusí se skládat **A**, ale rovnou její Choleského rozklad.

4 Poznámky k obecnějším systémům

4.1 Symetrické indefinitní matice

Příklad.

$$\mathbb{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$a_{11} = 0$, nelze dělit nulou a Choleského rozklad neexistuje.

Je třeba permutovat řádky a sloupce matice **A**, ale žádnou symetrickou permutací nedostaneme $a_{11} \neq 0$.

Při nesymetrické permutaci ztratíme symetrii - nebudeme diskutovat obecný indefinitní případ

Symetrické indefinitní soustavy vznikají například ve smíšené formulaci MKP nebo při řešené vázaných extrémů funkcí.

V tomto případě má matice **A**následující tvar:

$$\mathbb{R}^{n,m} \to \mathbb{A} = \begin{pmatrix} \mathbb{B} & \mathbb{C} \\ \mathbb{C}^T & 0 \end{pmatrix}$$

A dále pro $x \in \mathbb{R}^m, y \in \mathbb{R}^{n-m}$

$$\mathbb{B}x + \mathbb{C}y = b_1 \tag{5}$$

$$\mathbb{C}^T x = b_2 \tag{6}$$

Obvykle navíc platí, že $\mathbb{B} \in \mathbb{R}^{m,m}$ je SPD a řídká, $\mathbb{C} \in \mathbb{R}^{m,m-n}, m \geq n-m$ a navíc $h(C) = n-m$.

Postup řešení je pak následující:

$\mathbb{B}$ je SPD $\implies \exists$ Choleského rozklad, tj. $\mathbb{B} = \mathbb{L}\mathbb{L}^T \implies$ máme $\mathbb{B}^{-1}$

Přenásobme (5) $\mathbb{B}^{-1}$

$$x = \mathbb{B}^{-1}(b_1 - \mathbb{C}y)$$

dosadíme do (6)

$$\mathbb{C}^T\mathbb{B}^{-1}(b_1 - \mathbb{C}y) = b_2$$

Upravíme na

$$\mathbb{C}^T\mathbb{B}^{-1}\mathbb{C}y = \mathbb{C}^T\mathbb{B}^{-1}b_1 - b_2$$

Označme $\mathbb{C}^T\mathbb{B}^{-1}\mathbb{C} =: \mathbb{S} \in \mathbb{R}^{n-m,n-m}$, takzvaný Schurův doplněk.

Soustavu:

$$\mathbb{S}y = \mathbb{C}^T\mathbb{B}^{-1}b_1 - b_2$$

lze řešit přímo nebo iterativně.

$\mathbb{B}$ je SPD a $h(C) = n-m \implies \mathbb{S}$ je SPD

$$< \mathbb{C}^T\mathbb{B}^{-1}\mathbb{C}y, y > = < \mathbb{B}^{-1}\mathbb{C}y, \mathbb{C}y > \geq \alpha||\mathbb{C}y||^2 > 0$$

, kde α je konstanta PD $\mathbb{B}^{-1}$.

$\mathbb{B}$ je SPD $\implies \mathbb{B}^{-1}$ je SPD $\Leftrightarrow y \neq 0$.

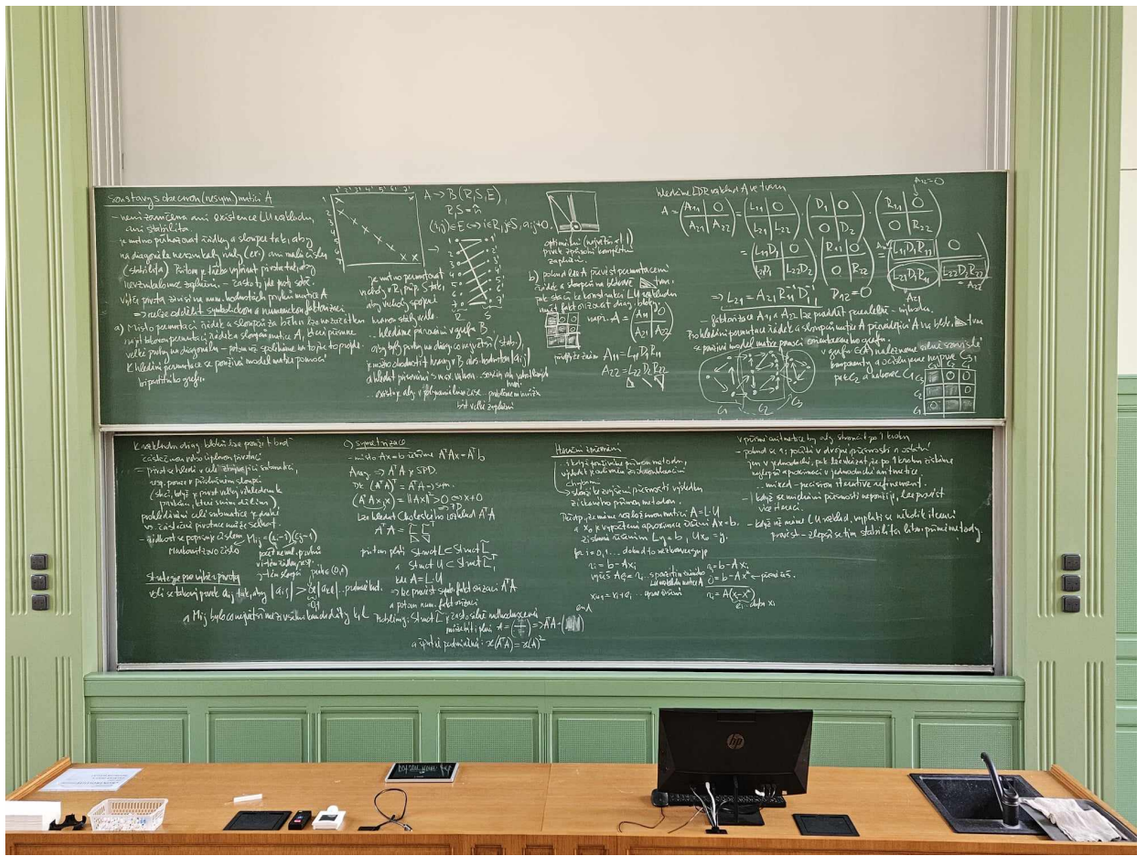
Nejprve budeme řešit $\mathbb{S}y = \mathbb{C}^T\mathbb{B}^{-1}b_1 - b_2 \to y$. Pak řešíme $\mathbb{B}x = b_1 - \mathbb{C}_y$.

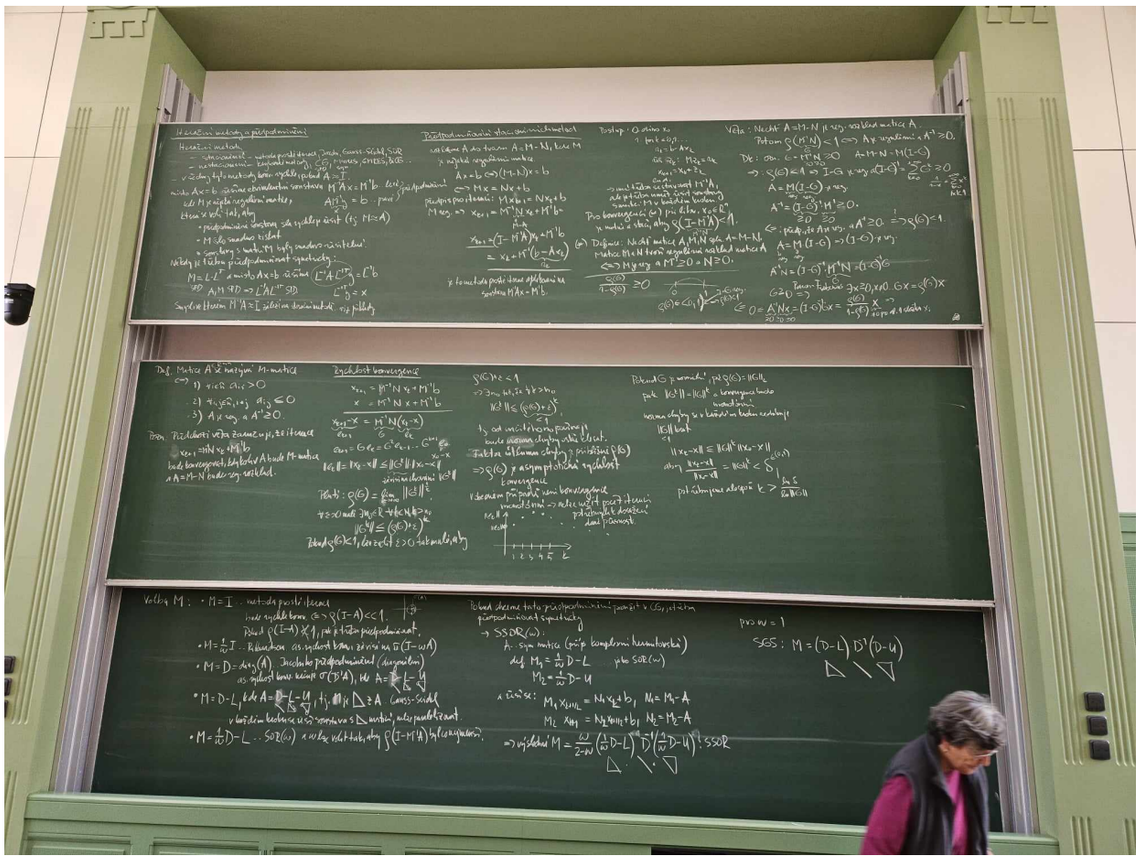
Obě soustavy mají SPD matice. $\mathbb{S}$ je hustá (!), i když je $\mathbb{B}$ řídká.

Pokud je $n-m$ malá, pak hustota nevadí. Alternativně lze soustavu s maticí $\mathbb{S}$ řešit iteračně. Pak není třeba $\mathbb{S}$ explicitně sestavovat, stačí s nimi umět přenásobovat vektory

$$\mathbb{S}y = \mathbb{C}^T\mathbb{B}^{-1}\mathbb{C}y = \mathbb{C}^T{\mathbb{L}^{-1}}^T\mathbb{L}^{-1}\mathbb{C}y$$

1. Spočteme $\mathbb{C}y$
2. Řešme $\mathbb{L}z = \mathbb{C}y \to z$
3. $\mathbb{L}^T u = z$
4. $\mathbb{S}y = \mathbb{C}^T u$





4.2 Předpodmiňování metody složených gradientů

Aplikací CG (konjugovaných gradientů) na předpodmíněnou soustavu

$$\mathbb{L}^{-1}\mathbb{A}\mathbb{L}^{-1^T} y = \mathbb{L}^{-1}b, \mathbb{L}^{-1^T} y = x, \mathbb{M} = \mathbb{L}\mathbb{L}^T$$

Lze odvodit algoritmus PCG — dopln jmeno

1. vstup: $\mathbb{A} \in \mathbb{R}^{n,n}$ SPD,$x_0 \in \mathbb{R}^n$ libovolný.
2. $r_0 = b - \mathbb{A}x_0$
3. vyřeš: $\mathbb{M}z_0 = r_0$, z_0 je předpodmíněné reziduum
4. $d_0 = z_0$
5. for $i = 0, 1 \ldots$ až je aproximace dost přesná
   (a) $\alpha_i = \frac{<r_i, z_i>}{<d_i, \mathbb{A}d_i>}$
   (b) $x_{i+1} = x_i + \alpha_i d_i$
   (c) $r_{i+1} = r_i - \alpha_i \mathbb{A} d_i$
   (d) vyřeš $\mathbb{M}z_{i+1} = r_{i+1}$
   (e) $\beta_i = \frac{<r_{i+1}, z_{i+1}>}{<r_i, z_i>}$
   (f) $d_{i+1} = z_{i+1} + \beta_i d_i$
6. end for

Poznámka. Při porovnání s normálními konjugovanými gradienty si všimněme rozdílu pouze v pár krocích

4.3 Rychlost konvergence PCG

O konjugovaných gradientech pro konvergenci víme následující

$$||x_n - x||_{\mathbb{A}} \leq 2 \left(\frac{\sqrt{\varkappa(\mathbb{A})} - 1}{\sqrt{\varkappa(\mathbb{A})} + 1} \right)^n ||x_0 - x||_{\mathbb{A}}$$

předpodmínění $\mathbb{M} = \mathbb{L}\mathbb{L}^T$ bude účinné, pokud $\varkappa(\mathbb{L}^{-1}\mathbb{A}\mathbb{L}^{-1^T}) << \varkappa(\mathbb{A})$

Aje SPD s vlastními čísly: $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Poté $\varkappa(\mathbb{A}) = \frac{\lambda_n}{\lambda_1}$. Toto je dobré, pokud vlastní čísla $\mathbb{L}^{-1}\mathbb{A}\mathbb{L}^{-1^T}$ tvoří malý shluk daleko od 0.

4.4 Testovací případ

Vezměme matici **A**která pochází z diskretizace Poissonovy rovnice $u''(x) = f(x), u(0) = u(1) = 0$ metodou sítí, $h = \frac{1}{n+1}$.

Například $A_3 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$

Platí následující:

Aje SPD

$$k \in \hat{n} : \lambda_k = 4\sin^2\frac{k\pi}{2(n+1)}, v_k = \left(\sin\frac{ik\pi}{n+1}\right)_{i+1}^{n}$$

4.4.1 Metoda prosté iterace

$\mathbb{M} = \mathbb{I}$

$$\varrho(\mathbb{I} - \mathbb{A}) = \max_{k\in\hat{n}} |1 - \lambda_k| = \max_{k\in\hat{n}} |1 - 4\sin^2\frac{k\pi}{2(n+1)}| = \max\{|1-\lambda_1|, 1-\lambda_n\} = \max\{< 1, \approx 3\} \approx 3$$

Metoda prosté iterace NEkonverguje!, je nutno předpodmiňovat.

4.4.2 Richardson

$\mathbb{M} = \frac{1}{\omega}\mathbb{I}, \omega \in \mathbb{R}$

$$\varrho(\mathbb{I} - \omega\mathbb{A}) = \max_{k\in\hat{n}} |1 - \omega\lambda_k| = \max |1 - 4\omega\sin^2\frac{k\pi}{2(n+1)}|$$

Z toho plyne, že ω je třeba volit tak aby $1 - 4\omega\sin^2\frac{k\pi}{2(n+1)} \in (-1, 1)$, proto

$$1 - 4\omega\sin^2\frac{k\pi}{2(n+1)} > 1 - 4\omega\sin^2\frac{\pi}{2} > -1$$

Z toho plyne, že stačí volit $\omega \leq \frac{1}{2}$

Pro konvergenci Richardsonovy metody je nutná a postačující podmínka, že $\omega \in (0; \frac{1}{2}\rangle$

$$\varrho(\mathbb{I} - \omega\mathbb{A}) = \max\{|1 - \omega\lambda_1|, 1 - \omega\lambda_n\} = \max\left\{|1 - 4\omega\sin^2\frac{\pi}{2(n+1)}|, 1 - 4\omega\sin^2\frac{n\pi}{2(n+1)}\right\} = \max\left\{1 - O(h^2), 1 - 4\omega\cos^2\frac{\pi}{2(n+1)}\right\}$$

Poslední rovnost plyne z

$$\sin\frac{n\pi}{2(n+1)} = \sin\left(\frac{\pi}{2} - \frac{\pi}{2(n+1)}\right) = \cos\frac{\pi}{2(n+1)}$$

Pro $\omega = \frac{1}{2}$ je maximum ze 2 stejných čísel. Z toho

$$\varrho(\mathbb{I} - \omega\mathbb{A}) = 1 - O(h^2)$$

Pro $h \to 0+$ se konvergence zastavuje.

4.4.3 Jacobi

$\mathbb{M} = \mathbb{D} = diag(\mathbb{A}) \implies$ Richardson s $\omega = \frac{1}{2}$

$$\varrho(\mathbb{I} - \mathbb{D}^{-1}\mathbb{A}) = \max\left\{|1 - \frac{1}{2}\lambda_1|, 1 - \frac{1}{2}\lambda_n\right\} = \max\left\{|1 - 2\sin^2\frac{\pi}{2(n+1)}|, 1 - 2\sin^2\frac{n\pi}{2(n+1)}\right\} = 1 - 2\sin^2\frac{\pi}{2(n+1)} = 1 - 2\sin^2\frac{\pi h}{2} = 1 - O(h^2)$$

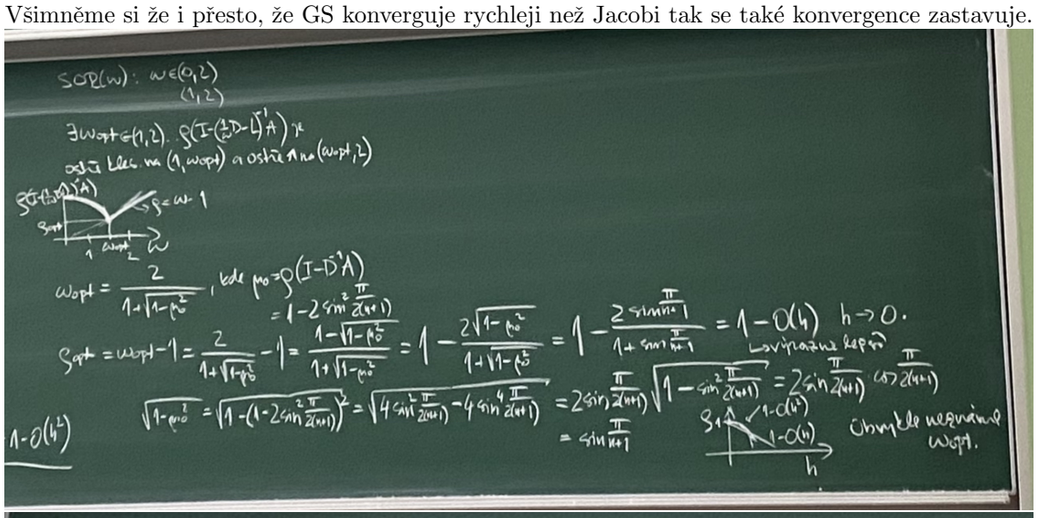
4.4.4 Gauss-Seidel a SOR(w)

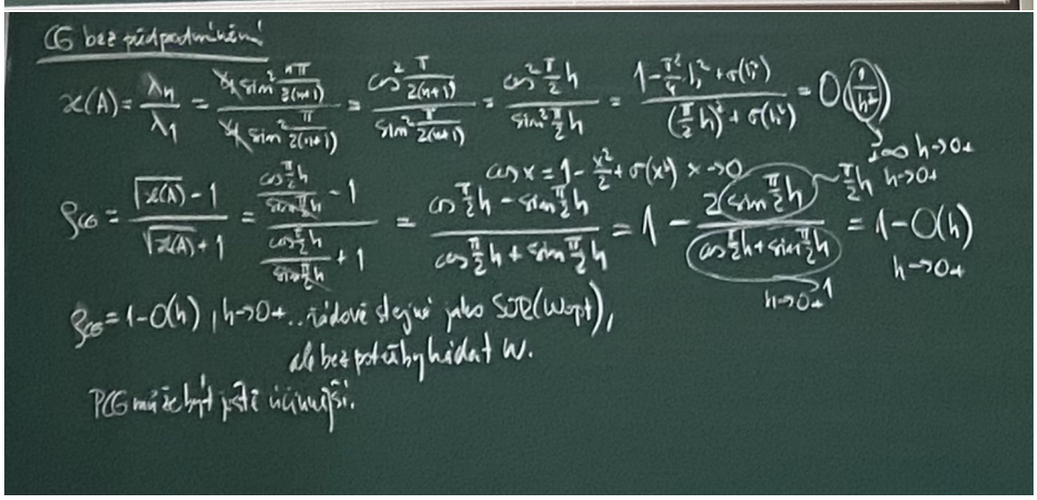
Aje dvoucyklická a shodně uspořádaná. Proto lze použít teorii z numerické matematiky. O rychlosti konvergence rozhoduje $\varrho(\mathbb{I} - (\frac{1}{\omega}\mathbb{D} - \mathbb{L})^{-}1)\mathbb{A}$, kde označme $G_{SOR} \coloneqq \mathbb{I} - (\frac{1}{\omega}\mathbb{D} - \mathbb{L})^{-}1)\mathbb{A}$. Vlastní čísla $\lambda \in \sigma(G_{SOR})$ a $\mu \in \sigma(G_{Jacobi})$ jsou svázaná rovnicí $(\lambda + \omega - 1)^2 = \omega^2\mu^2\lambda$.

Gauss-Seidel: $\omega = 1 : \lambda = \mu^2$

$$\varrho(\mathbb{I} - (\mathbb{D} - \mathbb{L})^{-1}\mathbb{A}) = \varrho(\mathbb{I} - \mathbb{D}^{-1}\mathbb{A})^2 = (1 - 2\sin^2\frac{\pi}{2(n+1)}) = 1 - 4\sin^2\frac{\pi}{2(n+1)} + 4\sin^4\frac{\pi}{2(n+1)} = 1 - O(h^2)$$

Všimněme si že i přesto, že GS konverguje rychleji než Jacobi tak se také konvergence zastavuje.





5 Neúplné LU rozklady (ILU - incomplete LU decomposition)

Při LU rozkladu vzniká zaplnění, LU rozklad se počítá "přibližně"a ignorují se některá (nebo všechna) zaplnění. Takováto aproximace $\tilde{L}\tilde{U} \approx \mathbb{A}$ může sloužit jako účinné předpodmínění.

5.1 ILU(k)

Pro SPD matice můžeme použít neúplný Choleského rozklad (IC - Incomplete Cholesky). Základní varianta ILU(0), resp. IC(0).
LU rozklad po řádkách:

```
for i = 2 ... n
    for k = 1 ... i-1
        a_{ik} = a_{ik}/a_{kk}
        for j = k+1 ... n
            a_{ij} = a_{ij} - a_{ik}a_{kj}
        end j
    end k
end i
```

Z toho dostáváme ILU(0):

```
for i = 2 ... n
    for k = 1 ... i-1
        if a_{ik} != 0
            a_{ik} = a_{ik}/a_{kk}
            for j = k+1 ... n
                if a_{ij} != 0
                    a_{ij} = a_{ij} - a_{ik}a_{kj}
                end if
            end j
        en dif
    end k
end i
```

Výsledkem je matice se stejnou strukturou jako původní matice, vzniká nulové zaplnění.
Obecněji lze předepsat $P \subset \hat{n} \times \hat{n}$ ve kterých zanedbáváme zaplnění

$$P = \{(i,j) \in \hat{n} \times \hat{n} | l_{ij} = 0 \text{ pro i¿j, respektive } u_{ij} = 0 \text{ pro } i > j\}$$

Nesmí se zanedbávat diagonální prvky $\implies P \subset \{(i,j) \in \hat{n} \times \hat{n}, i \neq j\}$

Podmínky if v algoritmu se pak nahradí na $if(i,k) \notin P$ a $if(i,j) \notin P$, jde o neúplnost podle masky.

Platí věta:

Věta 5.1. Pokud **A**je M-matice a $P \subset \{(i,j) \in \hat{n} \times \hat{n} | i \neq j\}$, pak algoritmus ILU s maskou P neselže a sestrojí neúplný LU rozklad matice **A**tvaru $\mathbb{A} = \mathbb{LU} - \mathbb{R}$, který je regulární.

Metoda prosté iterace s tímto předpodmíněním bude konvergovat.

Matice $\mathbb{R}$ odpovídá zanedbaným prvkům, platí že pro $(i,j) \notin P, R_{ij} = 0$.

ILU(0) odpovídá volbě $P = \{(i,j) \in \hat{n} \times \hat{n} | a_{ij} = 0\}$.

Pokud chceme přesnější LU rozklad, lze připustit i některé zaplnění, například takto:

Definujeme pro každé $(i,j) \in \hat{n} \times \hat{n}$ takzvaný $level_{ij}, level_{ij} = 0$ pokud $a_{ij} \neq 0$. Pokud při konstrukci LU rozkladu vzniká na pozici (i,j) zaplnění a vzniká součinem $a_{ik}a_{kj} \neq 0$, pak definujeme $level_{ij} = level_{ik} + level_{kj} + 1$.

Definujeme $P_k = \{(i,j) \in \hat{n} \times \hat{n} | i \neq j \wedge level_{ij} > k\}$.

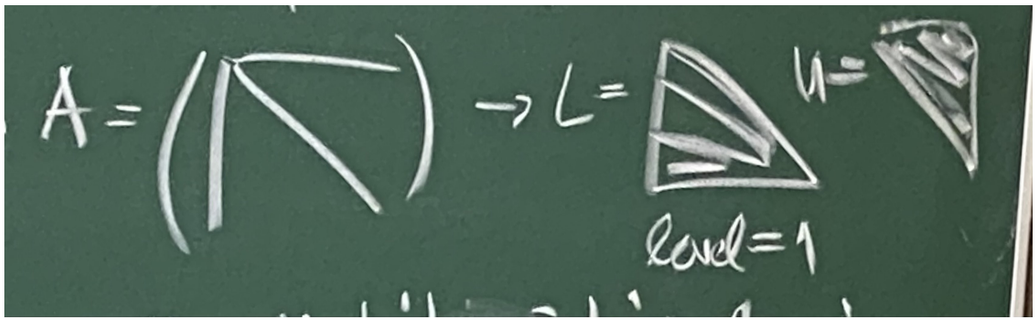
ILU(k) pak označuje neúplnost podle úrovně zaplnění.

5.1.1 Grafová interpretace ILU(k)

Zaplnění na pozici (i,j) má $level_{ij} = k \Leftrightarrow$ v $G(\mathbb{A})\exists$ cesta z i do j vedoucí přes vrcholy s indexy $< \min\{i,j\}$ délky $k+1$.

Z toho jde předpovědět pozice zaplnění v ILU(k) (symbolická faktorizace).

Poznámka. I pro ILU(1) může vznikat problém s množstvím zaplnění například



5.2 Další varianty

Maska P vzniká za běhu v závislosti na velikosti zanedbaných prvků.

Z toho máme neúplnost podle hodnoty, malé hodnoty zanedbáme, velké ponecháme.

Říkáme že $a_{ij}^{(k)}$ je malý pokud

$$|a_{ij}^{(}k)| < \alpha ||a_{i,\cdot}^{(k)}, \alpha = 0,01$$

Výhoda je, že se nemůže stát abychom zanedbali velký prvek.

Nevýhoda je, že struktura L,U závisí na numerických hodnotách $a_{ij}^{(k)}$, může vzniknout úplně nová struktura, mnoho zaplnění. Proto je potřeba doplnit omezení počtu nenulových prvků na řádek.

Z toho dostáváme ILUT(α, n) - ILUT with threshold. Obvykle se volí $\alpha = 0.01, n = 10$

V algoritmu se pak vybírá n největších prvků v $|\cdot|$, které splní prahové kritérium.

Poznámka. Existují i varianty ILU, které se snaží kompenzovat zanedbané prvky. MILU, respektive MIC (modified ILU resp. IC). V nich se zanedbaný prvek přičte k diagonále.

Poznámka. Při použití IC(0) + CG je potřeba daleko méně iterací než při použití diagonálního předpodmínění.

Přesto $\varkappa(\mathbb{M}^{-1}\mathbb{A}) = O(h^{-2})$, počet iterací $O(h^{-1})$.

Pro MIC + CG $\varkappa(\mathbb{M}^{-1}\mathbb{A}) = O(h^{-1})$. Pak je počet iterací k dosažení zadané přesnosti $O(h^{-\frac{1}{2}})$.

Poznámka. V každé iteraci je třeba řešit soustavy $\mathbb{PAP}^T$ s maticí $M = LU$, děláme 2 zpětné chody.

To je silně sekvenční operace, špatně se paralelizuje, lze přeuspořádat rovnice: