Sem vložte zadanie Vašej práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalárska práca

# Řešič řídkých soustav lineárních rovnic

***Lukáš Turčan***

Vedúci práce: Ing. Ivan Šimeček, Ph.D.

12. mája 2015

# Poďakovanie

Na tomto mieste by som sa chcel poďakovať všetkým, ktorí pri mne stáli počas písania mojej bakalárskej práce. Ďakujem vedúcemu práce Ing. Ivanovi Šimečkovi, Ph.D. za vedenie tejto práce a cenné rady.

# Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov. V súlade s ustanovením § 46 odst. 6 tohoto zákona týmto udeľujem bezvýhradné oprávnenie (licenciu) k užívaniu tejto mojej práce, a to vrátane všetkých počítačových programov ktoré sú jej súčasťou alebo prílohou a tiež všetkej ich dokumentácie (ďalej len „Dielo“), a to všetkým osobám, ktoré si prajú Dielo užívať.

Tieto osoby sú oprávnené Dielo používať akýmkoľvek spôsobom, ktorý nezhoršuje hodnotu Diela, a za akýmkoľvek účelom (vrátane komerčného využitia). Toto oprávnenie je časovo, územne a množstevne neobmedzené. Každá osoba, ktorá využije vyššie uvedenú licenciu, sa však zaväzuje priradiť každému dielu, ktoré vznikne (čo i len čiastočne) na základe Diela, úpravou Diela, spojením Diela s iným dielom, zaradením Diela do diela súborného či zpracovaním Diela (vrátane prekladu), licenciu aspoň vo vyššie uvedenom rozsahu a zároveň sa zaväzuje sprístupniť zdrojový kód takého diela aspoň zrovnateľným spôsobom a v zrovnateľnom rozsahu ako je zprístupnený zdrojový kód Diela.

V Prahe 12. mája 2015 .....

### Odkaz na túto prácu

Turčan, Lukáš. *Řešič řídkých soustav lineárních rovnic*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Táto práca sa zaoberá problematikou riedkych sústav lineárnych rovníc a ich riešením s využitím priamych metód. Popisuje vybrané metódy riešenia, heuristiky určené na zefektívnenie výpočtu a obsahuje prehľad vybraných riešení. Výsledkom tejto práce je kompaktná aplikácia s grafickým užívateľským rozhraním, ktorá ponúka niekoľko metód a heuristík s cieľom umožniť voľbu efektívneho riešenia pre čo najväčšiu množinu riedkych sústav.

**Klíčová slova** riedka sústava, minimalizácia dodatočného zaplnenia, Gaussova eliminačná metóda, LU rozklad, Choleského rozklad, Heuristika AMD, Heuristika COLAMD, Heuristika RCM, Markowitzová stratégia, CLI, GUI

# Abstract

This thesis deals with sparse linear systems and their solutions using direct methods. It describes a selected set of methods for solving these systems and heuristics for performance optimization. Furthermore it contains a survey of a selected solvers. The result of this thesis is a compact application with a graphical user interface that offers several methods and heuristics and aims to allow the user to choose an efficient solution for the largest set of sparse linear systems.

**Keywords** sparse systems, fill-in reduction, Gaussian elimination method, LU decomposition, Cholesky decomposition, Approximate Minimum Degree Ordering, Column Approximate Minimum Degree Ordering, Reverse Cuthill-McKee, Markowitz strategy, CLI, GUI

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Úvod

Lineárna algebra je matematický odbor, ktorý nám dovoľuje určovať a chápať množstvo objektov, situácii a priestorov v nejakom kontexte. Esenciálnou súčasťou lineárnej algebry sú lineárne rovnice, ktorých riešenie je jednoduché a z toho dôvodu sa nimi popisujú rôzne problémy, prípadne sú využité na ich aproximáciu. Lineárne rovnice majú široké využitie napríklad v informatike, elektrotechnike, jadrovom inžinierstve a v mnohých ďalších odvetviach.

Táto práca sa zaoberá špecifickou podmnožinou lineárnych rovníc, rozoberá jej problematiku, metódy a heuristiky na jej riešenie, analyzuje existujúce riešenia a navrhuje vlastné riešenie s možnosťou užívateľského výberu vhodnej metódy pre daný problém. Hlavným cieľom tejto práce je práve určenie požiadaviek, návrh a implementácia tohto riešenia.

V prvej časti je uvedená problematika a teoretický základ dôležitý pre popísanie daných metód. Druhá časť vysvetľuje princípy použitých metód a popisuje vybrané heuristiky a je zakončená stručnou rešeršou existujúcich riešení ako základ pre návrh vlastného riešenia. Posledná časť obsahuje návrh, realizáciu a následné testovanie vlastného riešenia.

Práca je zakončená záverom, ktorý sumarizuje obsah práce, dosiahnuté ciele a navrhuje možnosti na pokračovanie tejto práce.

Kapitola 1

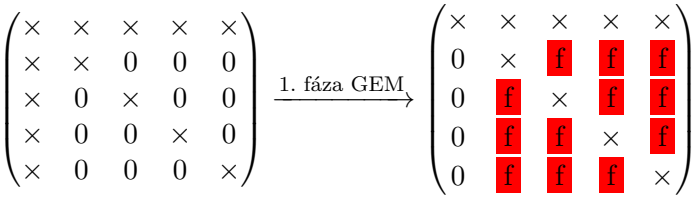
# Popis problému a cieľ práce

## 1.1 Riedke sústavy lineárnych rovníc

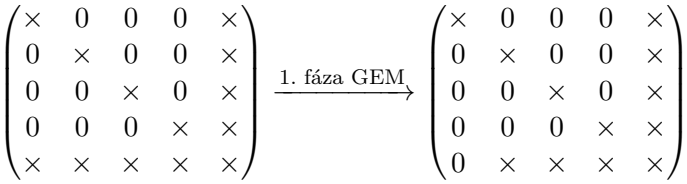
Množstvo problémov vyjadrených vzťahom malého množstva elementov s veľkým množstvom elementov je možné popísať lineárnymi rovnicami so zanedbateľným množstvom členov, teda tzv. riedkymi sústavami lineárnych rovníc. Tento fakt sa využíva a matice týchto sústav sa reprezentujú v špecializovaných formátoch s cieľom znížiť časovú a pamäťovú náročnosť výpočetných operácii. Pre skutočne rozsiahle riedke sústavy lineárnych rovníc môže byť štandardný spôsob reprezentácie extrémne náročný, ba až nerealizovateľný, a to dokonca aj na modernom hardvéry.

Medzi praktické uplatnenia týchto rovníc v oblasti informatiky je možné zahrnúť rôzne doporučovacie systémy, reprezentáciu mapy webovej stránky, výpočet hodnotenia webových stránok na základe spätných odkazov používaný vyhľadávacími službami a podobne.

Na výpočet riedkych sústav sa využívajú priame a nepriame metódy. Daňou za relatívnu presnosť u priamych metód, ktorými sa táto práca zaoberá, je vznik nových prvkov s výpočetne a pamäťovo náročnou reprezentáciou. Počet nových prvkov (dodatočné zaplnenie) je možné minimalizovať preskupením matice sústavy. Optimálne preskupenie nie je možné jednoznačne určiť v polynomiálnom čase, preto je tento problém pokrytý množstvom heuristík. Obrázky 1.1 a 1.2 ilustrujú vznik dodatočného zaplnenia a jeho minimalizáciu preskupením matice.




Obr. 1.1: Ilustrácia vzniku dodatočného zaplnenia matice




Obr. 1.2: Po preskupení matice už dodatočné zaplnenie nevniká

## 1.2 Špecifikácia cieľa

Cieľom tejto práce je návrh a implementácia aplikácie na riešenie riedkych sústav lineárnych rovníc, ktorá dovolí užívateľovi vybrať čo optimálny spôsob riešenia danéj sústavy s nasledujúcimi parametrami:

- rozhranie v príkazovom riadku a grafické užívateľské rozhranie,
- možnosť výberu Gaussovej eliminačnej metódy, LU rozkladu a Choleského rozkladu,
- výber z minimálne dvoch heuristík pre minimalizáciu dodatočného zaplnenia matice,
- dostupné rozhranie pre dodatočnú implementáciu ďalších metód a heuristík,
- zobrazenie informácii o danej sústave,
- podpora vonkajšej reprezentácie vo formáte „Matrix Market“,
- vnútorná reprezentácia vo formáte „Compressed Column Storage“,
- podpora platforiem GNU/Linux a OS X.

Kapitola 2

# Teoretický základ

Táto kapitola obsahuje teoretický základ čerpaný z [2, 3], ktorý je základnou esenciou potrebnou na vysvetlenie jednotlivých metód používaných v aplikácii.

## 2.1 Matica

Matica typu $(m, n)$ je usporiadaná $m$-tica prvkov z $\mathbf{R}^n$. Jednotlivé zložky tejto $m$-tice nazývame *riadky matice*. Nech $\vec{a}_r = (a_{r,1}, a_{r,2}, \ldots, a_{r,n})$ je $r$-tý riadok matice typu $(m, n)$. $s$-tá zložka tohto riadku $a_{r,s} \in \mathbf{R}$ sa nazýva $(r, s)$-*tý prvok* matice. Riadky matice $\mathbf{A}$ zapisujeme ako skutočné riadky pod seba takto:

$$\mathbf{A} = \begin{pmatrix} a_{1,1}, & a_{1,2}, & \ldots, & a_{1,n} \\ a_{2,1}, & a_{2,2}, & \ldots, & a_{2,n} \\ & & \vdots & \\ a_{m,1}, & a_{m,2}, & \ldots, & a_{m,n} \end{pmatrix}.$$

Nech $\mathbf{A} = (a_{r,s}), r \in \{1, \ldots, m\}, s \in \{1, \ldots, n\}$. Usporiadanú $m$-ticu reálnych čísel $(a_{1,s}, a_{2,s}, \ldots, a_{m,s})$ nazývame *s-tým stĺpcom matice* $\mathbf{A}$.

Maticu nazývame *štvorcová matica*, pokiaľ $m = n$. Maticu, ktorá má všetky prvky nulové, nazývame *nulová matica*.

Nech matica neobsahuje nulové riadky. Táto matica je v *hornom trojuholníkovom tvare*, ak pre ľubovolný riadok platí, že prvý nenulový prvok je napravo oproti prvému nenulovému prvku predchádzajúceho riadku. Takúto maticu budeme značiť $\mathbf{U}$. Naopak táto matica je v *dolnom trojuholníkovom tvare*, ak pre ľubovolný riadok platí, že prvý nenulový prvok je napravo oproti prvému nenulovému prvku predchádzajúceho riadku. Takúto maticu budeme značiť $\mathbf{L}$.

**Definícia.** Nech $\mathbf{A} = (a_{i,j})$ je matica typu $(n, n)$. *Hlavná diagonála matice* $\mathbf{A}$ (resp. len *diagonála*) je skupina jej prvkov $a_{1,1}, a_{2,2}, \ldots, a_{n,n}$. *Vedľajšia*

*diagonála matice* $\mathbf{A}$ zahŕňa prvky $a_{1,n}, a_{2,n-1}, \ldots, a_{n,1}$. *Prvok pod hlavnou diagonálou* je každý prvok $a_{i,j}$, pre ktorý platí $i > j$. *Prvok nad hlavnou diagonálou* je každý prvok $a_{i,j}$, pre ktorý platí $i < j$.

## 2.2 Násobenie matíc

**Definícia.** Nech $\mathbf{A} = (a_{i,j})$ je matica typu $(m, n)$ a $\mathbf{B} = (b_{j,k})$ je matica typu $(n, p)$. Potom je definovaný *súčin matíc* $\mathbf{A} \cdot \mathbf{B}$ (v tomto poradí) ako matica typu $(m, p)$ takto: každý prvok $c_{i,k}$ matice $\mathbf{A} \cdot \mathbf{B}$ je daný vzorcom

$$c_{i,k} = a_{i,1}\, b_{1,k} + a_{i,2}\, b_{2,k} + \cdots + a_{i,n}\, b_{n,k} = \sum_{j=1}^{n} a_{i,j}\, b_{j,k},$$

$$i \in \{1, \ldots, m\}, k \in \{1, \ldots, p\}.$$

**Pozn.** Všimnime si, že násobenie je definované len vtedy, pokiaľ počet stĺpcov prvej matice je rovný počtu riadkov druhej matice. Výsledná matica má rovnaký počet riadkov, ako prvá matica a rovnaký počet stĺpcov, ako druhá matica.

## 2.3 Determinant

**Definícia.** Nech $M$ je konečná množina o $n$ prvkoch. *Permutácia prvkov množiny* $M$ je usporiadaná $n$-tica prvkov množiny $M$ taká, že žiadny prvok z množiny $M$ sa v nej neopakuje. Permutáciou prvkov množiny $M = \{1, 2, \ldots, n\}$ nazývame stručne *permutáciu n prvkov*.

**Definícia.** Nech $(i_1, i_2, \ldots, i_n)$ je permutácia $n$ prvkov. *Inverzia* tejto permutácie je taká dvojica $(i_k, i_l)$, pre ktorú platí $i_k > i_l$, a pritom $k < l$.

**Definícia.** Pre každú permutáciu $\pi = (i_1, \ldots, i_n)$ definujeme *znamienko permutácie* $\operatorname{sgn}(\pi)$ takto:

$$\operatorname{sgn}(\pi) = \begin{cases} +1 & \text{ak } \pi \text{ má párny počet inverzii} \\ -1 & \text{ak } \pi \text{ má nepárny počet inverzii} \end{cases}$$

**Definícia.** Nech $\mathbf{A} = (a_{i,j})$ je štvorcová matica typu $(n, n)$. Číslo

$$\sum_{\pi=(i_1,i_2,\ldots,i_n)} \operatorname{sgn}\pi \cdot a_{1,i_1}\, a_{2,i_2} \cdots a_{n,i_n}$$

nazývame *determinantom matice* $\mathbf{A}$ a značíme ho $\det \mathbf{A}$.

## 2.4 Špeciálne typy matíc

**Definícia.** Štvorcová matica $\mathbf{A}$ typu $(n, n)$ sa nazýva *regulárna*, pokiaľ $\det(\mathbf{A}) \neq 0$. Štvorcová matica $\mathbf{A}$ sa nazýva *singulárna*, pokiaľ nie je regulárna.

**Definícia.** Štvorcovú maticu $\mathbf{E}$ typu $(n, n)$ nazývame *jednotkovou maticou*, pokiaľ pre jej prvky $e_{i,j}$ platí: $e_{i,j} = 0$ pre $i \neq j$ a $e_{i,j} = 1$ pre $i = j$.

**Definícia.** Nech $\mathbf{A} = (a_{i,j})$ je matica typu $(m, n)$. Maticu $\mathbf{A}^T = (a_{j,i})$, ktorá je typu $(n, m)$, nazývame *transponovanou maticou* k matici $\mathbf{A}$. Matica $\mathbf{A}^T$ vznikne z matice $\mathbf{A}$ prepísaním riadkov matice $\mathbf{A}$ do stĺpcov matice $\mathbf{A}^T$, respektíve prepísaním stĺpcov matice $\mathbf{A}$ do riadkov matice $\mathbf{A}^T$.

**Definícia.** Štvorcová matica $\mathbf{A}$ typu $(n, n)$ je *symetrická*, pokiaľ platí $\mathbf{A}^T = \mathbf{A}$.

**Definícia.** Nech $\mathbf{A}$ je štvorcová matica typu $(n, n)$. Označme $\mathbf{A}_i$ štvorcovú maticu typu $(n-i, n-i)$, ktorá vzniká z matice $\mathbf{A}$ vynechaním posledných $i$ riadkov a posledných $i$ stĺpcov. Matica $\mathbf{A}$ sa nazýva *pozitívne definitná*, pokiaľ všetky determinanty $\det \mathbf{A}_i$, $i \in \{0, 1, 2, \ldots, n-1\}$ sú kladné.

**Definícia.** Nech $\mathbf{A}$ je štvorcová matica typu $(n, n)$ a $\mathbf{E}$ je jednotková matica rovnakého typu. Maticu $\mathbf{B}$ typu $(n, n)$, ktorá splňuje vlastnosť $\mathbf{A} \cdot \mathbf{B} = \mathbf{E} = \mathbf{B} \cdot \mathbf{A}$ nazývame *inverzná matica* k matici $\mathbf{A}$. Inverznú maticu k matici $\mathbf{A}$ označujeme symbolom $\mathbf{A}^{-1}$.

**Definícia.** Matica typu (n,n), ktorá má v každom riadku a v každom stĺpci práve jeden prvok 1, zvyšné prvky majú nulovú hodnotu sa nazýva *permutačná matica*. Permutačnú maticu budeme označovať $\mathbf{P}$. Ďalej platí: $\mathbf{P}^{-1} = \mathbf{P}^T$. [4]

**Pozn.** Maticu, ktorá je symetrická a pozitívne definitná budeme označovať skratkou *SPD*.

## 2.5 Sústava lineárnych rovníc

Nech $\mathbf{A}$ je matica reálnych čísel typu $(m, n)$, nech ďalej $\vec{x}$ je jednostĺpcová matica symbolov

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

typu $(n, 1)$ a $\vec{b}$ je matica reálnych čísel

$$\begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

typu $(m, 1)$. Potom maticovú rovnosť

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

nazývame *sústavou m lineárnych rovníc o n neznámych*. Maticu $\mathbf{A}$ nazývame *maticou sústavy* a vektor $\vec{b}^T = (b_1, \ldots, b_m)$ nazývame *vektorom pravých strán*.

Sústavu $\mathbf{A}\,\vec{x} = \vec{b}$ môžme ekvivalentne zapísať

1. ako maticu s rozšírenou pravou stranou:

$$(\mathbf{A}|\vec{b}),$$

2. vypísaním jednotlivých lineárnych rovníc:

$$\begin{aligned} a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \cdots \quad & + a_{1,n} \cdot x_n = b_1 \\ a_{2,1} \cdot x_1 + a_{2,2} \cdot x_2 + \cdots \quad & + a_{2,n} \cdot x_n = b_2 \\ & \vdots \\ a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \cdots \quad & + a_{m,n} \cdot x_n = b_m, \end{aligned}$$

kde $a_{r,s}$ je prvok matice $\mathbf{A}$, $b_r$ je prvok vektora $\vec{b}$ a $x_s$ je prvok vektora $\vec{x}$, $r \in \{1, 2, \ldots, m\}, s \in \{1, 2, \ldots, n\}$.

**Riešením sústavy** $\mathbf{A}\,\vec{x} = \vec{b}$ je taký vektor $\vec{a} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbf{R}^n$, pre ktorý platí: ak dosadíme hodnoty $\alpha_i$ za symboly $x_i$, potom je splnená požadovaná maticová rovnosť, tj.

$$\mathbf{A} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

*Riešiť sústavu* $\mathbf{A}\,\vec{x} = \vec{b}$ znamená nájsť všetky jej riešenia, tj. nájsť podmnožinu $\mathbf{R}^n$ všetkých riešení tejto sústavy.

## 2.6 Riedke sústavy lineárnych rovníc

V tejto sekcii sú stručne uvedené riedke sústavy [5] a formáty na reprezentáciu riedkych matíc.

### 2.6.1 Počet nenulových prvkov matice

Nech $\mathbf{A}$ je matica typu $(m, n)$, potom mohutnosť množiny

$$\{a_{r,s} \in \mathbf{A}, a_{r,s} \neq 0, r \in \{1, \ldots, m\}, s \in \{1, \ldots, n\}\}$$

budeme nazývať počet nenulových prvkov matice $\mathbf{A}$ a označovať $nnz(A)$.

### 2.6.2 Riedka matica

Nech $\mathbf{A}$ je matica typu $(m, n)$. Ak platí

$$nnz(\mathbf{A}) < \frac{mn}{3},$$

potom túto maticu nazývame riedkou maticou. V opačnom prípade ide o hustú maticu. Pri reprezentácii riedkych matíc využívame úsporné formáty.

### 2.6.3 Riedka sústava lineárnych rovníc

Nech $\mathbf{A}$ je riedka matica. Potom rovnosť

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

nazývame riedkou sústavou lineárnych rovníc.

### 2.6.4 Dodatočné zaplnenie matice

Pri riešení riedkych sústav lineárnych rovníc môžu vznikať v matici sústavy nové prvky (napr. pričítaním riadkov matice), ktoré budeme nazývať „fill-in“ [6], respektívne dodatočné zaplnenie matice.

### 2.6.5 Reprezentácia riedkych matíc

Táto sekcia popisuje súradnicový a súborový formát „Matrix Market“ na vonkajšiu reprezentáciu matíc a vylepšenú verziu formátu „Column Sparse Row“ na vnútornú reprezentáciu riedkych matíc.

#### 2.6.5.1 Súradnicový formát „Matrix Market“

Súradnicový formát „Matrix Market“ [7] je určený na reprezentáciu riedkych matíc/vektorov. Tento formát obsahuje len nenulové prvky spolu s ich koordináciami.

Matice v tomto formáte sú reprezentované vo forme textového súboru, ktorý obsahuje nasledujúce časti:

(1) **Hlavička** obsahuje identifikátor („%%MatrixMarket“) a štyri textové polia:

- typ reprezentovaného objektu - matica („matrix“), alebo vektor („vector“),
- formát súboru - súradnicový formát („coordinates“),
- formát prvkov matice - reálne čísla („real“), reálne čísla s dvojitou presnosťou („double“), komplexné čísla („complex“) a ďalšie,
- symetria - obecná matica („general“), symetrická matica („symmetric“).

(2) **Komentár** časť nie je povinná a môže byť ľubovolne dlhá. Každý riadok musí začínať s „%“.

(3) **Veľkosť matice** špecifikuje počet riadkov, stĺpcov a počet reprezentovaných prvkov v matici.

(4) **Dátová časť** popisuje každý reprezentovaný prvok v matici, pre reálne čísla obsahuje 3 údaje:

- riadok v ktorom sa nachádza element (číslované od 1),
- stĺpec v ktorom sa nachádza element (číslované od 1),
- hodnota elementu v predtým špecifikovanom formáte.

**Pozn.** Pri symetrických maticiach sú uvedené len prvky pod hlavnou diagonálou (vrátane).

**Príklad.** Uvažujme maticu typu (5,5):

$$\begin{pmatrix} 1 & 0 & 0 & 6 & 21 \\ 0 & 19.5 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -270 & 31.82 \\ 0 & 3 & 0 & 0 & 12 \end{pmatrix}.$$

Táto matica môže byť v súradnicovej variante formátu „Matrix Market“ reprezentovaná nasledovne:

```
%%MatrixMarket matrix coordinate real general
% Matica 3x3 s deviatimi nenulovymi prvkami.
5 5 9
1 1    1
1 4    6
1 5    21
2 2    19.5
3 1    14
4 4    -270
4 5    31.82
5 2    3
5 5    12
```

#### 2.6.5.2 Súborový formát „Matrix Market“

Súborový formát „Matrix Market“ [7] je vhodný na reprezentáciu hustých matíc a vektorov.

Matice v tomto formáte sú reprezentované vo forme textového súboru, ktorý obsahuje nasledujúce časti:

(1) **Hlavička** obsahuje identifikátor („%%MatrixMarket“) a štyri textové polia:

- typ reprezentovaného objektu - matica („matrix“), alebo vektor („vector“),
- formát súboru - súborový formát („array“),
- formát jednotlivých elementov - reálne čísla („real“), reálne čísla s dvojitou presnosťou („double“), komplexné čísla („complex“) a ďalšie,
- symetria - obecná matica („general“), symetrická matica („symmetric“).

(2) **Komentár** časť nie je povinná a môže byť ľubovolne dlhá. Každý riadok musí začínať s „%“.

(3) **Veľkosť matice** špecifikuje počet riadkov a stĺpcov reprezentovanej matice.

(4) **Dátová časť** obsahuje hodnoty matice v predtým špecifikovanom formáte, zoradené podľa stĺpcov matice.

**Príklad.** Uvažujme maticu typu (3,3):

$$\begin{pmatrix} 1 & 15 \\ 12 & 3 \end{pmatrix}.$$

Táto matica môže byť v súborovej variante formátu „Matrix Market“ reprezentovaná nasledovne:

```
%%MatrixMarket matrix array real general
% Matica 2x2 so styrmi elementmi.
2 2
1
12
15
3
```

#### 2.6.5.3 „Compressed Column Storage“

Vylepšená verzia [8] formátu „Compressed Column Storage“ (CCS) je určená na vnútornú reprezentáciu riedkych matíc. CCS pozostáva z nasledujúcich 4 polí:

(1) **„Values“** - obsahuje hodnoty nenulových prvkov,

(2) **„InnerIndicies“** - obsahuje riadkové indexy (1),

(3) **„OuterStarts“** - pre jednotlivé stĺpce matice udáva index prvého nenulového prvku v predchádzajúcich dvoch poliach[1],

(4) **„InnerNNZs“** - uchováva počty nenulových prvkov v jednotlivých stĺpcoch.

**Príklad.** Uvažujme maticu typu (5,5):

$$\begin{pmatrix} 0 & 3 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 17 \\ 7 & 5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 0 & 8 \end{pmatrix},$$

jedná z možných reprezentácii vo vylepšenom „Compressed Column Storage“ formáte vyzerá nasledovne:

| Values | 22 | 7 | - | 3 | 5 | 14 | - | - | 1 | - | 17 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| InnerIndices | 1 | 2 | - | 0 | 2 | 4 | - | - | 2 | - | 1 | 4 |

| OuterStarts | 0 | 3 | 5 | 8 | 10 | 12 |
|---|---|---|---|---|---|---|

| InnerNNZs | 2 | 2 | 1 | 1 | 2 |
|---|---|---|---|---|---|

Je garantované, že prvky v poli „Values“ sú vždy zoradené podľa rastúcich riadkových indexov. Hodnota „-“ je vylepšením pôvodného formátu a značí voľné miesto na uloženie nového prvku. Takáto úprava dovolí oveľa efektívnejšie vkladanie nových prvkov.

Reprezentácia uvedenej matice v pôvodnom CCS formáte, ktorú získame odstránením „-“ a poľa „InnerNNZs“, vyzerá nasledovne:

| Values | 22 | 7 | 3 | 5 | 14 | 1 | 17 | 8 |
|---|---|---|---|---|---|---|---|---|
| InnerIndices | 1 | 2 | 0 | 2 | 4 | 2 | 1 | 4 |

| OuterStarts | 0 | 2 | 4 | 5 | 6 | 8 |
|---|---|---|---|---|---|---|

[1] posledný element tohto poľa ukazuje na koniec (1) a (2)

Kapitola 3

# Metódy na riešenie sústav lineárnych rovníc

Táto kapitola popisuje vybrané metódy riešenia sústav lineárnych rovníc spolu s príkladmi. V kapitole bolo čerpané z [2], [3], [9] a [10].

## 3.1 Gaussova eliminačná metóda

Gaussova eliminačná metóda pozostáva z dvoch častí, doprednej eliminácie a následnej spätnej substitúcie.

V doprednej eliminácii prevedieme maticu $(\mathbf{A}|\vec{b})$ na ekvivalentnú $(\mathbf{U}|\vec{g})$ pomocou nasledujúcich krokov:

1. prehodenie dvoch riadkov,
2. vynásobenie riadku nenulovou hodnotou,
3. pripočítame k nejakému riadku násobok iného riadku,
4. odstránenie nulového riadku,

kde pre $\mathbf{U}$ je horná trojuholníková matica. Matice $(\mathbf{A}|\vec{b})$ a $(\mathbf{U}|\vec{g})$ sú navzájom prevoditeľné a majú rovnakú množinu riešení.

**Pozn.** Vytváranie núl v ľubovolnom stĺpci prebieha spôsobom že si zvolíme prvok v $k$ stĺpci, ktorý preusporiadaním riadkov umiestnime na pozíciu $k, k$. Následne pripočítavame také násobky $k$-tého riadku k zvyšným riadkom pod pivotom, aby mal $k$-ty stĺpec pod pozíciou $k, k$ iba nulové hodnoty. Tento prvok sa nazýva pivot. Voľba pivota je veľmi dôležitá a pivot nesmie byť 0.

Následne pokračujeme spätnou substitúciou. Ignorujúc nulové riadky, zapíšeme sústavu $(\mathbf{U}|\vec{g})$ v tvare lineárnych rovníc. Sústava $(\mathbf{U}|\vec{g})$

1. nemá riešenie, ak obsahuje rovnicu v tvare $0 = c, c \neq 0$,

2. má práve jedno riešenie, ak počet neznámych je rovný počtu rovníc,
3. má $\infty$ mnoho riešení, v opačnom prípade.

Začíname rovnicou, ktorú sme získali z posledného nenulového riadku matice. Z tejto rovnice vyjadríme neznáme a dosadíme ich do rovnice získanej z predposledného nenulového riadku matice. Takto pokračujeme až po prvý riadok matice. Na konci by sme mali mať vyjadrenú celú množinu riešenia.

**Príklad.**

Máme nasledujúcu sústavu rovníc:

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 8 \\ -3x_1 - x_2 + 2x_3 &= -11 \\ -2x_1 + x_2 + 2x_3 &= -3, \end{aligned}$$

ktorá v maticovom zápise vyzerá nasledovne:

$$(\mathbf{A}|\vec{b}) = \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right).$$

Začíname doprednou eliminációu.

$$(\mathbf{A}|\vec{b}) = \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right) \overset{(3)}{\sim}$$

$$\overset{(3)}{\sim} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ -2 & 1 & 2 & -3 \end{array}\right) \overset{(3)}{\sim} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{array}\right) \overset{(2)}{\sim} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1 & 1 & 2 \\ 0 & 2 & 1 & 5 \end{array}\right) \overset{(3)}{\sim}$$

$$\overset{(3)}{\sim} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -1 & 1 \end{array}\right) = (\mathbf{U}|\vec{g})$$

.

Pôvodnú maticu sme upravili na hornú trojuholníkovú maticu s rozšírenou pravou stranou. Pokračujeme spätnou substitúciou, prepíšeme si sústavu $(\mathbf{U}|\vec{b})$ na jednotlivé rovnice:

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 8 \\ x_2 + x_3 &= 2 \\ -x_3 &= 1 \end{aligned}$$

Máme 3 rovnice o 3 neznámych, sústava má jedno riešenie. Prvú neznámu máme priamo v poslednej rovnici, postupným dosadením získame zvyšné a výsledné riešenie vyzerá nasledovne: $\vec{x} = (2, 3, -1)$.

## 3.2 LU rozklad

LU rozklad (tiež nazývaný ako LU dekompozícia, prípadne LU faktorizácia) je metóda rozkladu matice **A** typu n na súčin hornej trojuholníkovej matice **U** a dolnej trojuholníkovej matice **L**. Táto metóda bola uvedená anglickým matematikom Alanom Turingom v roku 1948 [11].

Postup pri riešení $\mathbf{A}\vec{x} = \vec{b}$ pomocou LU rozkladu:

(1) Nájdeme rozklad **A** (tj. nájdeme **L**,**U** také, aby platilo $\mathbf{A} = \mathbf{LU}$).

(2) Dosadíme do maticovej rovnice sústavy **LU** namiesto **A**, dostávame $(\mathbf{LU})\vec{x} = \vec{b}$.

(3) Nech $\mathbf{U}\vec{x} = \vec{y}$, potom potrebujeme vyriešiť $\mathbf{L}\vec{y} = \vec{b}$.

(4) Získaný $\vec{y}$ dosadíme do $\mathbf{U}\vec{x} = \vec{y}$ a získame $\vec{x}$, ktorý je riešením $\mathbf{A}\vec{x} = \vec{b}$.

**Pozn.** LU rozklad nie je unikátny.

### 3.2.1 Doolittlov algoritmus

Tento algoritmus je najčastejšie používaným algoritmom na výpočet LU rozkladu.

Výpočet matice **U** je takmer totožný s doprednou elimináciou v Gaussovej eliminačnej metóde s výnimkou nasledujúcich rozdielov:

(1) Výmena riadkov matice nie je dovolená. Ak vymeníme riadky, potom LU rozklad nebude existovať!

(2) Je vhodné, no nie nevyhnutné, podeliť riadky matice tak, aby prvý nenulový prvok každého riadku mal hodnotu 1. Avšak je nutné dbať na zachovanie numerickej stability. Tento krok sa vykonáva ako posledná operácia meniaca hodnoty v danom riadku.

(3) Všetky operácie je nutné zaznamenávať, históriu operácii využijeme na vytvorenie **U**.

Pre výpočet matice **L** začneme s jednotkovou maticou typu n, ktorú upravíme s využitím krokov pri výpočte **U** podľa nasledujúcich pravidiel:

(1) Riadkové operácie, ktoré zahŕňajú pričítanie násobku jedného riadku k druhému:
Pre každú operáciu v tvare $\mathbf{A}[i] \leftarrow \mathbf{A}[i] + k \cdot \mathbf{A}[j]$ vložíme hodnotu $-k$ na pozíciu $\mathbf{L}[i][j]$.

(2) Delenie riadku nenulovou hodnotou:
Pre každú operáciu v tvare $\mathbf{A}[i] \leftarrow k \cdot \mathbf{A}[i]$ vložíme hodnotu $\frac{1}{k}$ na pozíciu zhodnú s prvým prvkom zľava v riadku $\mathbf{U}[i]$.

**Príklad:**

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & -4 & 6 \\ 3 & -9 & -3 \end{pmatrix}$$

S použitím doprednej eliminácie získame $\mathbf{U}$:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -4 & 6 \\ 3 & -9 & -3 \end{pmatrix} \overset{[2]=[2]-2[1]}{\sim} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -8 & 0 \\ 3 & -9 & -3 \end{pmatrix} \overset{[3]=[3]-3[1]}{\sim} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -8 & 0 \\ 0 & -15 & -12 \end{pmatrix} \overset{[2]=-\frac{1}{8}[2]}{\sim}$$

$$\overset{[2]=-\frac{1}{8}[2]}{\sim} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & -15 & -12 \end{pmatrix} \overset{[3]=[3]+15[2]}{\sim} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & -12 \end{pmatrix} \overset{[3]=-\frac{1}{12}[3]}{\sim} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & -12 \end{pmatrix} = \mathbf{U}$$

S využitím histórie úprav a pravidiel získame maticu $\mathbf{L}$:

(1) Začíname s jednotkovou maticou typu n:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(2) 1. úprava matice $\mathbf{A}$ (pravidlo (1)), $\mathbf{A}[2] \leftarrow \mathbf{A}[2]$ **-2** $\cdot \mathbf{A}[1]$.

$$\begin{pmatrix} 1 & 0 & 0 \\ \mathbf{2} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(3) 2. úprava matice $\mathbf{A}$ (pravidlo (1)), $\mathbf{A}[3] \leftarrow \mathbf{A}[3]$ **-3** $\cdot \mathbf{A}[1]$.

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ \mathbf{3} & 0 & 1 \end{pmatrix}$$

(4) 3. úprava matice $\mathbf{A}$ (pravidlo (2)), $\mathbf{A}[2] \leftarrow -\mathbf{\frac{1}{8}} \cdot \mathbf{A}[2]$.

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & \mathbf{-8} & 0 \\ 3 & 0 & 1 \end{pmatrix}$$

(5) 4. úprava matice $\mathbf{A}$ (pravidlo (1)), $\mathbf{A}[3] \leftarrow \mathbf{A}[3] + \mathbf{15} \cdot \mathbf{A}[2]$.

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & -8 & 0 \\ 3 & \mathbf{2} & 1 \end{pmatrix}$$

(6) Posledná úprava matice $\mathbf{A}$ (pravidlo (2)), $\mathbf{A}[3] \leftarrow -\frac{\mathbf{1}}{\mathbf{12}} \cdot \mathbf{A}[3]$.

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & -8 & 0 \\ 3 & 2 & \mathbf{-12} \end{pmatrix} = \mathbf{L}$$

LU rozklad vyzerá nasledovne:

$$\underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 2 & -4 & 6 \\ 3 & -9 & -3 \end{pmatrix}}_{\mathbf{A}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 2 & -8 & 0 \\ 3 & 2 & -12 \end{pmatrix}}_{\mathbf{L}} \cdot \underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & -12 \end{pmatrix}}_{\mathbf{U}}$$

### 3.2.2 Pivotácia

Ide o preusporiadanie riadkov/stĺpcov pôvodnej matice $\mathbf{A}$ za účelom zvýšenia numerickej stability, zníženia počtu „fill-inov“, prípadne umožnenia vykonania LU rozkladu vôbec. Napríklad pre maticu, ktorá má v ľavom hornom rohu nulový prvok nie je možné vykonať LU rozklad.

Pivotácia prebieha prenásobením pôvodnej matice $\mathbf{A}$ permutačnou maticou $\mathbf{P}$. Ak chceme preusporiadať:

(1) riadky, musíme vynásobiť maticu $\mathbf{A}$ permutačnou maticou $\mathbf{P}$ zľava,

(2) stĺpce, musíme vynásobiť maticu $\mathbf{A}$ permutačnou maticou $\mathbf{P}$ sprava.

**Pozn.** Násobenie permutačnou maticou $\mathbf{P}$ v ľubovoľnom smere je pri vhodnej implementácii zanedbateľné vzhľadom k ostatným operáciám.

**Čiastočná pivotácia** znamená preusporiadanie riadkov matice $\mathbf{A}$:

$$\mathbf{PA} = \mathbf{LU} \iff \mathbf{A} = \mathbf{P}^T\mathbf{LU}$$

.
Stručný postup LU rozkladu s čiastočnou pivotáciou:

(1) $\mathbf{A}\vec{x} = b$.

(2) Dosadíme $\mathbf{P}^T\mathbf{LU}$ za $\mathbf{A}$, dostávame $\mathbf{P}^T\mathbf{LU}\vec{x} = \vec{b} \iff \mathbf{LU}\vec{x} = \mathbf{P}^T\vec{b}$.

(3) Nech $\mathbf{U}\vec{x} = \vec{y}$, potom potrebujeme vyriešiť $\mathbf{L}\vec{y} = \mathbf{P}^T\vec{b}$.

(4) Získaný $\vec{y}$ dosadíme do $\mathbf{U}\vec{x} = \vec{y}$ a získame $\vec{x}$, ktorý je riešením $\mathbf{A}\vec{x} = \vec{b}$.

**Úplná pivotácia** znamená preusporiadanie riadkov a stĺpcov matice $\mathbf{A}$:

$$\mathbf{P_r A P_c} = \mathbf{LU} \iff \mathbf{A} = \mathbf{P_r^T L U P_c^T}$$

.
Stručný postup LU rozkladu s úplnou pivotáciou:

(1) $\mathbf{A}\vec{x} = b$.

(2) Dosadíme $\mathbf{P_r^T L U P_c^T}$ za $\mathbf{A}$, dostávame $\mathbf{P_r^T L U P_c^T}\vec{x} = \vec{b} \iff \mathbf{L U P_c^T}\vec{x} = \mathbf{P_r^T}\vec{b}$.

(3) Rozložíme problém na tri podproblémy: $\mathbf{L}\vec{y} = \mathbf{P_r^T}\vec{b}$, $\mathbf{U}\vec{z} = \vec{y}$, $\mathbf{P_c^T}\vec{z} = \vec{x}$.

(4) Z $\mathbf{L}\vec{y} = \mathbf{P_r^T}\vec{b}$ vypočítame $\vec{y}$, ktoré dosadíme do $\mathbf{U}\vec{z} = \vec{y}$ a z $\mathbf{P_c^T}\vec{z} = \vec{x}$ získame $\vec{x}$, ktorý je riešením $\mathbf{A}\vec{x} = b$.

## 3.3 Choleského rozklad

Choleského rozklad (tiež označovaný ako Choleského dekompozícia) je rozklad symetrickej pozitívne definitnej matice na súčin dolnej trojuholníkovej matice a hornej trojuholníkovej matice, pričom horná trojuholníková matica je transponovaná matica k dolnej trojuholníkovej matici:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T.$$

Metóda je pomenovaná po jej autorovi, francúzskom matematikovi, Andréovi-Louisovi Choleskom.

Postup pri riešení $\mathbf{A}\vec{x} = \vec{b}$ pomocou Choleského rozkladu:

(1) Nájdeme rozklad $\mathbf{A}$ (tj. nájdeme $\mathbf{L}$ také, aby platilo $\mathbf{A} = \mathbf{L}\mathbf{L}^T$).

(2) Dosadíme do maticovej rovnice $\mathbf{L}\mathbf{L}^T$ namiesto $\mathbf{A}$, dostávame $(\mathbf{L}\mathbf{L}^T)\vec{x} = \vec{b}$.

(3) Nech $\mathbf{L}^T\vec{x} = \vec{y}$, potom $\mathbf{L}\vec{y} = \vec{b}$.

(4) $\mathbf{L}\vec{y} = \vec{b}$ vyriešime spätnou substitúciou, $\vec{y}$ dosadíme do $\mathbf{L}^T\vec{x} = \vec{y}$ a získame $\vec{x}$, ktorý je riešením $\mathbf{A}\vec{x} = \vec{b}$.

### 3.3.1 Choleského-Croutov a Choleského-Banachiewiczov algoritmus

Tento algoritmus sa používa na výpočet Choleského rozkladu. Nech $\mathbf{A}$ je SPD matica:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{1,2} & a_{2,2} & \dots & a_{2,n} \\ \vdots & & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{n,n} \end{pmatrix} = \mathbf{L}\mathbf{L}^T =$$

$$\begin{pmatrix} l_{1,1} & 0 & 0 & \dots & 0 \\ l_{2,1} & l_{2,2} & 0 & \ddots & \vdots \\ l_{3,1} & l_{3,2} & l_{3,3} & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{n,n} \end{pmatrix} \cdot \begin{pmatrix} l_{1,1} & l_{2,1} & l_{3,1} & \dots & l_{n,1} \\ 0 & l_{2,2} & l_{3,2} & \dots & l_{n,2} \\ 0 & 0 & l_{3,3} & \dots & l_{n,3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & l_{n,n} \end{pmatrix} =$$

$$= \underbrace{\begin{pmatrix} l_{1,1}^2 & l_{1,1}l_{2,1} & l_{3,1}l_{1,1} & \dots & l_{n,1}l_{1,1} \\ l_{2,1}l_{1,1} & l_{2,2}^2 l_{2,1}^2 & l_{2,1}l_{3,1} + l_{3,2}l_{2,2} & \dots & l_{n,1}l_{2,1} + l_{n,2}l_{2,2} \\ l_{3,1}l_{1,1} & l_{3,1}l_{2,1} + l_{3,2}l_{2,2} & l_{3,1}^2 l_{3,2}^2 l_{3,3}^2 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n,1}l_{1,1} & l_{n,1}l_{2,1} + l_{n,2}l_{2,2} & \dots & \dots & \sum_{i=1}^{n} l_{n,i}^2 \end{pmatrix}}_{\mathbf{L}\mathbf{L}^T}$$

Z uvedeného si vieme odvodiť jednotlivé prvky matice $\mathbf{L}$.
Prvý stĺpec $\mathbf{L}$:

$$\begin{aligned} a_{1,1} = l_{1,1}^2 \to l_{1,1} &= \sqrt{a_{1,1}} \\ a_{2,1} = l_{2,1} \cdot l_{1,1} \to l_{2,1} &= a_{2,1}/l_{1,1} \\ a_{3,1} = l_{3,1} \cdot l_{1,1} \to l_{3,1} &= a_{3,1}/l_{1,1} \\ &\vdots \\ a_{n,1} = l_{n,1} \cdot l_{1,1} \to l_{n,1} &= a_{n,1}/l_{1,1} \end{aligned}$$

Druhý stĺpec $\mathbf{L}$:

$$\begin{aligned} a_{2,2} = l_{2,1}^2 + l_{2,2}^2 \to l_{2,2} &= \sqrt{a_{2,2} - l_{2,1}^2} \\ a_{3,2} = l_{3,1} \cdot l_{2,1} + l_{3,2} \cdot l_{2,2} \to l_{3,2} &= \frac{a_{3,2} - l_{3,1} \cdot l_{2,1}}{l_{2,2}} \\ &\vdots \\ a_{n,2} = l_{n,1} \cdot l_{2,1} + l_{n,2} \cdot l_{2,2} \to l_{n,2} &= \frac{a_{n,2} - l_{n,1} \cdot l_{2,1}}{l_{2,2}} \end{aligned}$$

Tretí stĺpec **L**:

$$a_{3,3} = l_{3,1}^2 + l_{3,2}^2 + l_{3,3}^2 \to l_{3,3} = \sqrt{a_{3,3} - l_{3,1}^2 - l_{3,2}^2}$$

$$\vdots$$

$$a_{n,3} = l_{n,1} \cdot l_{3,1} + l_{n,2} \cdot l_{3,2} + l_{n,3} \cdot l_{3,3} \to l_{n,3} = \frac{a_{n,3} - l_{n,1} \cdot l_{3,1} \cdot + l_{n,2} \cdot l_{3,2}}{l_{3,3}}$$

Takýmto postupom je možné postupne odvodiť nasledujúci vzťah pre ľubovolný prvok matice **L**:

$$l_{j,j} = \sqrt{a_{j,j} - \sum_{k=1}^{i-1} l_{i,k}^2},$$
$$l_{i,j} = \frac{1}{l_{j,j}}(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} \cdot l_{j,k}) \text{ pre } i > j.$$

**Pozn.** V prípade, že začíname v ľavom hornom rohu a vykonávame tento výpočet

(1) po riadkoch, jedná sa o Choleského-Banachiewiczov algoritmus,

(2) po stĺpcoch, ide o Choleského-Croutov algoritmus.

**Príklad.**

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{2,1} & a_{2,2} & a_{3,2} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix} = \mathbf{L}\mathbf{L}^T =$$

$$\begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix} \cdot \begin{pmatrix} l_{1,1} & l_{2,1} & l_{3,1} \\ 0 & l_{2,2} & l_{3,2} \\ 0 & 0 & l_{3,3} \end{pmatrix}$$

Postupne získame prvky **L**:

$$l_{1,1} = \sqrt{a_{1,1} - \sum_{k=1}^{0} l_{i,k}^2} = \sqrt{a_{1,1}} = 5$$

$$l_{2,1} = \frac{1}{l_{1,1}}(a_{2,1} - \sum_{k=1}^{0} l_{2,k} \cdot l_{1,k}) = \frac{1}{5}(15) = 3$$

$$l_{2,2} = \sqrt{a_{2,2} - \sum_{k=1}^{1} l_{2,k}^2} = \sqrt{18 - 3^2} = 3$$

$$l_{3,1} = \frac{1}{l_{1,1}}(a_{3,1} - \sum_{k=1}^{0} l_{3,k} \cdot l_{3,k}) = \frac{1}{5}(-5) = -1$$

$$l_{3,2} = \frac{1}{l_{2,2}}(a_{3,2} - \sum_{k=1}^{1} l_{3,k} \cdot l_{2,k}) = \frac{1}{3}(0 - (-1 \cdot 3) = 1$$

$$l_{3,3} = \sqrt{a_{3,3} - \sum_{k=1}^{2} l_{3,k}^2} = \sqrt{11 - ((-1)^2 + 1^2)} = 3$$

Matica **L**:

$$\mathbf{L} = \begin{pmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{pmatrix}$$

$$\mathbf{L}\mathbf{L}^T = \begin{pmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 5 & 3 & 1 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix} = \mathbf{A}$$

### 3.3.2 Pivotácia

Choleského metóda je numericky stabilná a je ju možné vykonať na ľubovoľnej SPD matici aj bez pivotácie. Avšak pivotácia je stále vhodná pre minimalizáciu fill-inov v prípade riedkej matice **A**.

Pivotácia prebieha preusporiadaním riadkov a stĺpcov permutačnou maticou a transponovanou maticou k tej istej permutačnej matici:

$$\mathbf{P}^T\mathbf{A}\mathbf{P} = \mathbf{L}\mathbf{L}^T \iff \mathbf{A} = \mathbf{P}\mathbf{L}\mathbf{L}^T\mathbf{P}^T$$

Stručný postup výpočtu $\mathbf{A}\vec{x} = b$ pomocou Choleského rozkladu s pivotáciou:

(1) Dosadíme $\mathbf{P}\mathbf{L}\mathbf{L}^T\mathbf{P}^T$ za **A**, dostávame $\mathbf{L}\mathbf{L}^T\mathbf{P}^T\vec{x} = \vec{b} \iff \mathbf{L}\mathbf{L}^T\mathbf{P}^T\vec{x} = \mathbf{P}^T\vec{b}$.

(2) Rozdelíme problém na podproblémy: $\mathbf{L}\vec{z} = \mathbf{P}^T\vec{b}$, $\mathbf{L}^T\vec{y} = \vec{z}$, $\mathbf{P}\vec{y} = \vec{x}$.

(3) $\mathbf{L}\vec{z} = \mathbf{P}^T\vec{b}$ vyriešime doprednou substitúciou, $\mathbf{L}^T\vec{y} = \vec{z}$ vyriešime spätnou substitúciou.

(4) Dosadíme $\vec{y}$ do $\mathbf{P}\vec{y} = \vec{x}$ a získame $\vec{x}$, ktoré je riešením $\mathbf{A}\vec{x} = \vec{b}$.

Kapitola 4

# Heuristiky pre minimalizáciu dodatočného zaplnenia matice

V tejto kapitole sú stručne popísané vybrané heuristiky. Väčšina zmienených heuristík je založená na grafových algoritmoch, ktoré sú bližšie popísané hneď v prvej časti tejto kapitoly. Následne sú zmienené jednotlivé heuristiky, kategorizované podľa symetrie matice. Výsledkom heuristík (s výnimkou Markowitzovej stratégie (viď kap. 4.3.1)) je permutačný vektor, ktorým sa v prípade symetrickej matice preskupia riadky a potom stĺpce matice. V prípade nesymetrickej matice sa preskupia len stĺpce matice.

## 4.1 Zavedenie pojmov z teórie grafov

Táto časť stručne zavádza základné pojmy z teórie grafov [12].

### 4.1.1 Neorientovaný graf

Usporiadaná dvojica $G = (V, E)$ pozostávajúca z neprázdnej množiny uzlov $V$ a množiny neorientovaných hrán $E$, pričom každá hrana z $E$ spája dva uzly z $U$ sa nazýva neorientovaným grafom. Slučkou označujeme jav, kedy hrana spája uzol so sebou samým.

### 4.1.2 Susedné uzly

Nech $G = (V, E)$ je neorientovaný graf. Ak hrana $e \in E$ spája dva uzly $u, v \in V$ v grafe $G$, potom uzly $u, v$ sa nazývajú susedné podľa hrany $e$. A výraz

$$adj_G(v)$$

označuje množinu susedných uzlov uzlu $v$ v grafe $G$.

### 4.1.3 Stupeň uzlu

Nech $G = (V, E)$ je neorientovaný graf s uzlom $v \in V$. Potom

$$deg_G(v)$$

nazývame stupňom uzlu $v$ v grafe $G$ a označuje počet hrán grafu $G$, ktoré spájajú uzol $v \in V$ s iným uzlom (slučky sú počítané dvakrát).

### 4.1.4 Matica susednosti

Nech $G = (V, E)$ je neorientovaný graf s množinou uzlov $V = \{v_1, v_2, \ldots, v_n\}$ a množinou hrán $E = \{e_1, e_2, \ldots, e_n\}$. A nech $\mathbf{A}$ je symetrická matica, pre ktorej každý prvok platí:

$$a_{i,j} = \text{počet hrán spájajúcich } v_i \text{ a } v_j.$$

Potom túto maticu nazývame maticou susednosti grafu $G$.

### 4.1.5 Graf reprezentujúci pozície nenulových elementov matice

Nech $\mathbf{A}$ je symetrická matica a $\mathbf{B}$ je matica, ktorej elementy majú hodnotu 1 na pozíciach zhodných s pozíciami nenulových elementov v matici $\mathbf{A}$ a hodnotu 0 na ostatných pozíciach. Potom neorientovaný graf reprezentovaný maticou susednosti $\mathbf{B}$ je grafom reprezentujúcim pozície nenulových elementov matice $\mathbf{A}$ [13].

## 4.2 Heuristiky určené pre symetrické matice

Nasledujúce heuristiky sú primárne určené pre symetrické matice, avšak možno ich použiť aj na nesymetrické matice. V tomto prípade sa namiesto pôvodnej matice $\mathbf{A}$ spočíta heuristika pre maticu $\mathbf{A^T} + \mathbf{A}$.

### 4.2.1 Minimum Degree Ordering

„Minimum Degree Ordering“ (MD) je heuristika založená na neorientovanom grafe, ktorý popisuje miesta matice s nenulovými prvkami. Dva uzly $i, j$ sú spojené hranou, ak element $(i, j)$ má nenulovú hodnotu. Číslo uzlu teda vyjadruje maticové súradnice. Táto heuristika priradí uzlom nové číslovanie, ktoré značí preskupenie riadkov a stĺpcov pôvodnej matice.

**Základný algoritmus**:

(1) vytvorí sa graf reprezentujúci pozície nenulových elementov matice,

(2) zvolí sa uzol s najmenším stupňom,

(3) zvolený uzol sa odstráni a jeho susedné uzly sa vzájomne prepoja,

(4) ak je počet zvolených uzlov menší ako počet stĺpcov matice, tak algoritmus pokračuje bodom (2).

V prípade existencie viacerých uzlov s rovnakým stupňom je v základnom algoritme výber stanovený arbitrárne. Poradie zvolených uzlov určuje preskupenie matice.

**Pozn.** V praxi sa používajú rôzne varianty tohto algoritmu, ktoré sa snažia urýchliť výpočet, či ešte viac znížiť počet „fill-inov“. Vylepšenia sú napríklad zlučovanie uzlov s rovnakým stupňom a množinou susedných uzlov, nahradenie odstraňovania uzlov efektívnejším spôsobom a ďalšie.

### 4.2.2 Multiple Minimum Degree Ordering

„Multiple Minimum Degree Ordering“ (MMD) je vylepšením základnej MD heuristiky. Pomerne často sa stáva, že mnoho uzlov má rovnaký stupeň. Táto heuristika tohto faktu využíva a snaží sa tieto uzly eliminovať súčasne. V [14] sú obidve heuristiky popísané bližšie spolu s princípmi a aj ich vylepšeniami.

### 4.2.3 Approximate Minimum Degree Ordering

Heuristiky MD a MMD stále nie je možné považovať za optimálne, nakoľko počítanie stupňov u jednotlivých uzlov je výpočetne príliš náročné. Heuristika „Approximate Minimum Degree Ordering“ (AMD) na tento problém reaguje [15].

Táto heuristika namiesto exaktného výpočtu stupňov jednotlivých uzlov počíta iba ich hornú hranicu. Tento výpočet je značne jednoduchší a v špecifikácii algoritmu je ukázané, že horná hranica pre stupeň je často rovná stupňu daného uzlu.

Heuristika AMD je oproti predchádzajúcim heuristikám z rodiny „Minimum Degree“ značne rýchlejšia, pričom dosahuje obdobné výsledky.

### 4.2.4 Cuthill–McKee

Jednou z najjednoduchších heuristík je „Cuthill-McKee“ (CM), ktorá je pomenovaná podľa jej autorov Elizabeth Cuthill a J. McKee [16]. Cieľom tejto heuristiky je zníženie pásu matice, čo vedie k redukcii počtu „fill-inov“.

**Postup**:

(1) zo zadanej matice sa vytvorí neorientovaný graf (zadaná matica je uvažovaná ako matica incidencie tohto grafu), ďalej sa vytvorí množina výsledkov $R$,

(2) v grafe sa zvolí uzol s najmenším stupňom, ktorý sa nenachádza v $R$ a vloží sa do $R$,

(3) z množiny výsledkov sa vyberie prvý uzol, ktorý doteraz nebol spracovaný,

(4) do množiny výsledkov sa vo vzostupnom poradí podľa stupňa uzlu, vložia susedné uzly vybraného uzlu s výnimkou uzlov, ktoré sa už v tejto množine nachádzajú,

(5) ak množina $R$ obsahuje nespracované uzly, tak algoritmus pokračuje bodom (3),

(6) ak množina $R$ obsahuje menej uzlov ako je pôvodný počet uzlov grafu, tak algoritmus pokračuje bodom (2).

Výsledná množina $R$ obsahuje nové poradie uzlov a určuje preskupenie matice. Voľba uzlu v prípade rovnakého stupňa je opäť stanovená arbitrárne.

Postup je veľmi podobný „Breadth-first search“ algoritmu [17] s rozdielom, že „Cuthill–McKee“ určuje poradie nasledujúcich uzlov v závislosti od ich stupňa.

### 4.2.5 Reverse Cuthill–McKee

„Reverse Cuthill–McKee“ je modifikáciou CM s jediným rozdielom, a to takým, že výsledné poradie uzlov sa invertuje. Podľa [18] vedie táto úprava k zníženiu počtu „fill-inov“ oproti CM.

Ďalším vylepšením tejto heuristiky je heuristika GPS, ktorá je pomenovaná po jej autoroch: Gibbsovi, Poolovi a Stockmeyerovi. Podľa [19] majú obidve heuristiky porovnateľné výsledky, avšak GPS je niekoľko násobne rýchlejšia. Bližšie je táto heuristika popísaná v [19].

## 4.3 Heuristiky vhodné pre nesymetrické matice

Nasledujúce heuristiky sú primárne určené pre nesymetrické matice, avšak možno ich použiť aj na symetrické matice.

### 4.3.1 Markowitzová stratégia

Markowitzová stratégia [20] popisuje jeden z možných spôsobov výberu pivota pri doprednej eliminácii. Nech $\mathbf{A}^{(k)}$ je matica $\mathbf{A}$ po vykonaní $k-1$ fáz doprednej eliminácie.
Nech

(1) $r(i,k)$ je počet nenulových prvkov v $i$-tom riadku matice $\mathbf{A}^{(k)}$, pričom uvažujeme prvky od $k$-tého stĺpca (vrátane),

(2) $c(i,k)$ je počet nenulových prvkov v $j$-tom stĺpci matice $\mathbf{A}^{(k)}$, pričom uvažujeme prvky od $k$-tého riadku (vrátane),

potom:

$$M_{i,j,k} = (r(i,k)-1)\cdot(c(j,k)-1)$$

nazývame Markowizovou cenou elementu $a_{i,j}^{(k)}$.
**Pozn.** $M_{i,j,k}$ udáva počet prvkov matice $\mathbf{A}^{(k)}$, ktoré zmenia hodnotu po vykonaní $k$-tej fáze doprednej eliminácie s pivotom $a_{i,j}^{(k)}$. Taktiež je to horná hranica pre počet „fill-inov“, ktoré môžu vzniknúť. Metóda nezaručuje minimalizáciu „fill-inov“, avšak je značne jednoduchšia na výpočet a v praxi sa ukázala ako rovnako účinná.

### 4.3.2 Zobecnená Markowitzova stratégia

Markowitzova stratégia má predovšetkým nasledujúce nedostatky:

(1) je nutné prehľadávať príliš veľa prvkov,

(2) nezohľadňuje numerickú stabilitu,

a odpoveďou na ne je Zobecnená Markowitzova stratégia.

**Zobecnená Markowitzova stratégia (GMS) v $k$-tej fáze**

(1) Nech $p \in \mathrm{N}$ sú konštanty, $p$ určuje počet prehľadávaných riadkov a $u$ silu požiadavky na numerickú stabilitu,

(2) $I_k$ je množina čísel obsahujúca riadkové indexy $min(p, n-k+1)$ prvkov od $k$-tého riadku (vrátane) v $k$-tom stĺpci, matice $\mathbf{A}^{(k)}$ s minimálnou hodnotou $r(i_k,k), i_k \in I_k$,

(3) množina $B_k$ obsahuje všetky prvky matice $\mathbf{A}^{(k)}$ nachádzajúce sa na riadku $i_k \in I_k$ a spĺňajúce podmienku stability:

$$|\mathbf{A}^{(k)}[I_k][j]|\cdot u \geq max(|\mathbf{A}^{(k)}[I_k][j]|...|\mathbf{A}^{(k)}[I_k][n]|),$$

(4) potom pre každý prvok v $B_k$ spočítame Markowitzovu cenu a prvky s minimálnou Markowitzovou cenou sú ideálnymi kandidátmi na pivota.

### 4.3.3 Column Approximate Minimum Degree Ordering

Najprv si je nutné uviesť symbolickú LU faktorizáciu, ktorá podobne ako LU faktorizácia vychádza z Gaussovej Eliminačnej metódy.

Cieľom symbolickej LU faktorizácie je zistiť, kde sa budú nachádzať nulové prvky po doprednej eliminácii. Pri tejto metóde nie sú podstatné hodnoty nenulových prvkov matice, ale iba ich umiestnenie.

Stručný popis metódy:

(1) fáza tejto metódy je podobná fáze doprednej eliminácii, avšak namiesto celého výpočtu sa len určia stĺpce, ktoré budú obsahovať nulové prvky bez ohľadu výberu pivota,

(2) najprv sa pre každý riadok určí $A_i$, teda množina stĺpcových indexov nenulových prvkov v $i$-tom riadku matice,

(3)
$$R_k = \left( \left( \bigcup_{k=min(R_i)} R_i \right) \cup \left( \bigcup_{k=min(A_i)} A_i \right) \right) \setminus \{k\},$$
obsahuje pre $k$-tú fázu stĺpcové indexy určujúce výskyty nenulových prvkov v $k$-tom riadku bez ohľadu na výber pivota.

**Pozn.** Uvedená metóda je symbolická LU faktorizácia bez stĺpcových úprav. Táto metóda je vhodná na vysvetlenie jednotlivých operácii, ale je neefektívna a v praxi sa nahradzuje efektívnejšou metódou so stĺpcovými úpravami. Metóda so stĺpcovými úpravami nie je v tejto práci vysvetlená, nakoľko pre potreby vysvetlenia nasledujúcej heuristiky postačuje pôvodná metóda.

„Column Approximate Minimum Degree Ordering“ je heuristika určená na preskupenie riadkov matice pred doprednou eliminációu. Táto heuristika vychádza zo symbolickej LU faktorizácie a snaží sa nájsť vhodný permutačný vektor, ktorý minimalizuje počet „fill-inov“.

Heuristika v $k$-tej fáze vyberie stĺpec s najmenšou metrikou a následne sa $k$-ty a $s$-ty stĺpec vzájomne vymenia.

Metriky môžu byť nasledovné:

(1) odhad hornej hranice použitý v heuristike „Column Minimum Degree Ordering“ [21],

(2) presný výpočet, mohutnosť množiny $R_k$,

(3) odhad hornej hranice použitý v heuristike AMD

V [22] je táto heuristika popísaná bližšie spolu s metrikami.

Kapitola 5

# Prehľad existujúcich riešení

V tejto kapitole je uvedený stručný prehľad existujúcich riešení určených na riešenie riedkych sústav lineárnych rovníc. Kapitola je zakončená tabuľkou, ktorá sumarizuje dôležité črty týchto riešení.

## 5.1 SuperLU

„SuperLU“ je riešič riedkych sústav lineárnych rovníc ($\mathbf{A}\vec{x} = \vec{b}$) pomocou LU rozkladu. Matica sústavy $\mathbf{A}$ môže byť ľubovoľná štvorcová regulárna matica, nemusí byť symetrická, či definitná. Dokonca „SuperLU“ je obzvlášť vhodný v prípade nesymetrických matíc.

Na preskupenie stĺpcov je možné zvoliť jednu z nasledujúcich heuristík:

(1) ponechá sa pôvodné usporiadanie,

(2) Multiple Minimum Degree (MMD) aplikovaná na $\mathbf{A}^T * \mathbf{A}$,

(3) Multiple Minimum Degree (MMD) aplikovaná na $\mathbf{A}^T + \mathbf{A}$,

(4) Column Minimum Degree Ordering (COLAMD),

(5) permutačná matica zadaná užívateľom.

Numerická stabilita je zaistená preskupením riadkov matice v priebehu výpočtu. V $i$-tej fáze rozkladu sa v $i$-tom stĺpci pod a na hlavnej diagonále vyberie prvok $m_i$ s maximálnou absolútnou hodnotou, vynásobí sa s užívateľsky definovanou konštantou $u \in \langle 0, 1\rangle$ a výsledná hodnota je minimálna hodnota pivota pre túto fázu. Ak prvok na diagonále v $i$-tom stĺpci splní túto podmienku, tak sa stane pivotom. V opačnom prípade dôjde k výmene riadkov a pivotom sa stáva prvok $m_i$.

SuperLU je napísaný v jazyku C, k dispozícii je rozhranie pre Fortran a Matlab. Celú dokumentáciu je možné nájsť v [23].

## 5.2 CHOLMOD

„CHOLMOD“ je skupina súvisiacich procedúr určených na riešenie SLR, $\mathbf{A}\vec{x} = \vec{b}$, kde $\mathbf{A}$ je riedka a symetrická pozitívne definitná matica a vektory $\vec{x}$ a $\vec{b}$ môžu byť riedke, alebo husté [24]. Riešenie prebieha pomocou Choleského rozkladu s možnosťou výberu z nasledujúcich heuristík:

(1) ponechá sa pôvodné usporiadanie,

(2) Approximate Minimum Degree (AMD),

(3) Column Approximate Minimum Degree (COLAMD),

(4) METIS [25]

(5) NESTIS (heuristika, ktorá je priamo súčasťou CHOLMOD-u)

(6) permutačná matica zadaná užívateľom.

Matica sústavy môže byť zadaná vo formáte „Matrix Market“, povolené sú reálne a komplexné čísla. Implementácia je v jazyku C s rozhraním pre Matlab.

## 5.3 UMFpack

„UMFpack“ je množina súvisiacich procedúr, ktorá je určená na riešenie riedkych SLR, $\mathbf{A}\vec{x} = \vec{b}$, kde $\mathbf{A}$ je riedka, štvorcová a nesymetrická matica. Podľa [26] používa metódu LU rozkladu, povolené sú reálne a komplexné čísla.

„UMFpack“ preskupí riadky s cieľom redukcie „fill-inov“ bez ohľadu na numerické hodnoty. Najprv sa z matice $\mathbf{A}$ odstránia pivoty s nulovou Markowitzovou cenou a vložia sa do $\mathbf{LU}$ matíc. Podľa zvyšnej časti matice $\mathbf{A}$ sa použije jedna z nasledujúcich stratégii:

(1) nesymetrická stratégia: preskupenie stĺpcov je vypočítané upravenou verziou COLAMD, počas faktorizácie sa preskupenie môže ešte zmeniť,

(2) symetrická stratégia: preskupenie je vypočítané AMD,

následne sa v prípade potreby preskupia riadky pre zachovanie numerickej stability.

„UMFpack“ je napísaný v jazyku C, poskytuje rozhranie pre Fortran a Matlab. Do verzie 2.2.1 a nižšej je „UMFpack“ napísaný v jazyku Fortran pod názvom „MA38“.

## 5.4 Sparse 1.4

„Sparse 1.4“ je jednoduchý balík procedúr, ktoré sú schopné rýchlo a presne riešiť rozsiahle riedke sústavy lineárnych rovníc. Je schopný zvládnuť reálne a komplexné maticové rovnice.

Sústavy rieši pomocou LU rozkladu a na redukciu „fill-in“ používa upravenú formu Markowitzovej stratégie.

Je napísaný v jazyku C a poskytuje rozhranie pre Fortran. Dokumentáciu je možné nájsť v [27] a [28].

## 5.5 Zhrnutie

V tabuľke 5.1 sú zhrnuté podstatné črty existujúcich riešení uvedených v tejto kapitole.

Tabuľka 5.1: Zhrnutie vybraných riešení

| Názov | Metóda | Matica | Heuristiky | Jazyk |
|---|---|---|---|---|
| SuperLU | LU rozklad | regulárna | MMD, COLAMD | C |
| CHOLMOD | Choleského metóda | SPD | AMD, COLAMD, METIS, NETIS | C |
| UMFpack | LU rozklad | regulárna | AMD, COLAMD[2], Markowitzová stratégia | C |
| Sparse 1.4 | LU rozklad | regulárna | Markowitzová stratégia[2] | C |

[2]heuristika nie je použitá v exaktnej podobe

Kapitola 6

# Analýza a návrh

## 6.1 Analýza požiadavkov

V tejto časti sú analyzované požiadavky na aplikáciu, ktoré vychádzajú z požiadavkov uvedených v úvode tejto práce (viď kap. 1.2). Prvá časť obsahuje prípady použitia z ktorých je možné identifikovať funkčné požiadavky popisujúce interakciu užívateľa s aplikáciou a očakávanú odozvu aplikácie. Nefunkčné požiadavky kladú nároky na dostupné rozhrania, spôsob implementácie, rozšíriteľnosť a podporu platforiem.

Predpoklad na splnenie funkčných a nefunkčných požiadaviek dáva návrh architektúry (viď kap. 6.2) a zvolené technológie (viď kap. 7.1). Overenie funkčných požiadaviek je súčasťou testovania implementovaného riešenia.

### 6.1.1 Prípady a scenáre použitia

Zoznam prípadov použitia, ktoré bude možné na aplikácii vykonať. Ku každému prípadu je zároveň aj ilustrovaný scenár použitia. Popisované scenáre sa týkajú grafického rozhrania, obdobné scenáre majú byť taktiež vykonateľné pomocou rozhrania v príkazovom riadku.

(UC1) Výpočet sústavy lineárnych rovníc

- Užívateľ zvolí maticu, príslušnú pravú stranu a súbor do ktorého sa uloží výsledok.
- Vyberie si vhodnú metódu na riešenie sústavy, heuristiku na minimalizáciu „fill-inov“ a následne si môže nechať túto sústavu vyriešiť.
- Voliteľne si môže zobraziť štatistiky o priebehu výpočtu.

(UC2) Zobrazenie informácii o sústave

- Užívateľ zvolí maticu, príslušnú pravú stranu a zadá požiadavku na zobrazenie informácii o sústave.

- Aplikácia následne zobrazí užívateľovi veľkosť sústavy, tvar matice a možné metódy riešenia tejto sústavy.

### 6.1.2 Funkčné požiadavky

(F1) Zobrazenie informácii o sústave

- Aplikácia zobrazí informácie o užívateľom zadanej sústave.

(F2) Povinná voľba metódy na riešenie riedkej sústavy lineárnych rovníc

- Užívateľ si musí zvoliť jednu z troch metód (viď kap. 1.2), ktorá bude použitá pre výpočet.

(F3) Možnosť voľby heuristiky na minimalizáciu dodatočného zaplnenia matice

- Užívateľ si môže zvoliť jednu z heuristík zvolených v kap. 6.3.

(F4) Kontrola vstupných dát

- Aplikácia overí integritu vstupných dát a zamedzí prípadnému výpočtu s nesprávnym vstupom.

(F5) Overenie kompatibility zvolenej metódy s riešenou sústavou

- Aplikácia zabráni výpočtu v prípade nekompatibility vstupnej matice so zvolenou metódou.

(F6) Výpočet riedkej sústavy lineárnych rovníc

- Aplikácia vyrieši užívateľom zadanú sústavu a výsledok uloží do súboru.

(F7) Možnosť zobrazenia štatistík o priebehu výpočtu

- Užívateľ si môže zobraziť štatistiku sumarizujúcu priebeh výpočtu.

### 6.1.3 Nefunkčné požiadavky

(NF1) Dostupnosť aplikácie cez príkazovú riadku

- Aplikáciu bude možné využívať cez príkazovú riadku štandardným spôsobom.

(NF2) Dostupnosť aplikácie cez GUI

- Aplikáciu bude možné využívať cez jej grafické rozhranie, ktoré bude z funkčného hľadiska obdobou rozhrania v príkazovom riadku.

(NF3) Rozšíriteľnosť

- V budúcnosti bude možné aplikáciu rozšíriť o ďalšie metódy a heuristiky.

(NF4) Podporované platformy

- Aplikáciu bude možné používať na platformách GNU/Linux a OS X.

(NF5) Vhodná technológia pre vykonávanie výpočtov

- Výpočty vykonávané aplikáciou musia byť založené na technológii dávajúcej predpoklad na ich nenáročnosť na systémový čas.

## 6.2 Návrh architektúry

Na aplikáciu sú kladené rozdielne nároky. Pri výpočte zvyčajne očakávame vysoký výkon a minimálne možné pamäťové nároky, avšak pri grafickom rozhraní tieto parametre nie sú príliš podstatné. Na grafické rozhranie sú skôr kladené požiadavky na použiteľnosť a užívateľskú prívetivosť. Z tohto dôvodu je vhodné rozdeliť aplikáciu na:

- výpočetnú časť s rozhraním v príkazovom riadku a
- grafické užívateľské rozhranie (GUI).

Takéto rozdelenie dovolí, aby výpočetná časť aplikácie mohla byť implementovaná pomocou technológie zaisťujúcej vysoký výkon a GUI mohlo byť implementované pomocou implementačne nenáročnej technológie. Čas ušetrený na implementácii GUI môže byť investovaný do prevedenia samotného rozhrania. Taktiež implementácia s použitím implementačne nenáročnej technológie umožní pohodlnejšie úpravy grafického rozhrania.

## 6.3 Voľba heuristík

Cieľom tejto práce je aplikácia, ktorá umožní užívateľovi zvoliť optimálny spôsob riešenia riedkej sústavy. Z tohto dôvodu boli zvolené nasledovné heuristiky:

(1) „Approximate Minimum Degree Ordering“ (4.2.3)

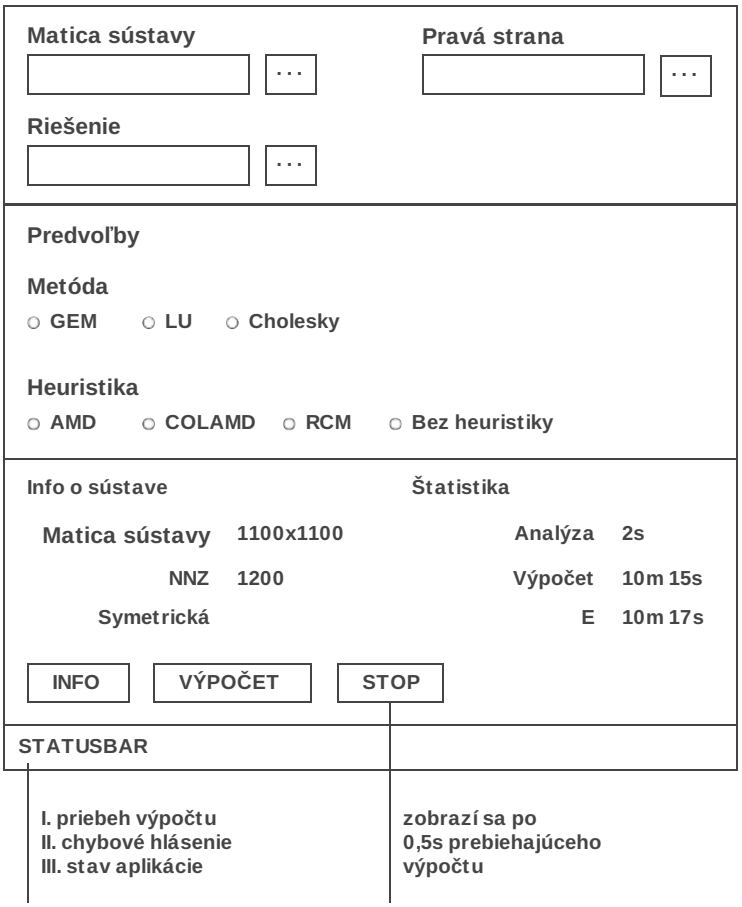
(2) „Column Approximate Minimum Degree Ordering“ (4.3.3)

(3) „Reverse Cuthill–McKee“ (4.2.5)

Pri vyberaní heuristík boli uvažované iba heuristiky popísané v (kap. 4). Heuristika AMD je vylepšená varianta heuristík MD a MMD, z toho dôvodu neboli tieto heuristiky uvažované. COLAMD bola uprednostnená pred Markowitzovou stratégiou nakoľko sa zdala byť vhodnejšia z implementačného hľadiska, zároveň COLAMD je značne komplexnejšia heuristika a mala by dosahovať lepšie výsledky.

## 6.4 Návrh grafického rozhrania

Grafické rozhranie bolo navrhnuté s dôrazom na jednoduchosť, návrh tohto rozhrania je reprezentovaný v podobe wireframu obr. 6.1.




Obr. 6.1: Wireframe grafického užívateľského rozhrania

Kapitola 7

# Realizácia

V kapitole sú diskutované zvolené technológie, reprezentácia sústavy, implementácia výpočetnej časti aplikácie a tvorba grafického rozhrania. Kapitolu uzatvára zhrnutie, ktoré popisuje jednotlivé požiadavky na aplikáciu a spôsob ich splnenia.

## 7.1 Zvolené technológie

V tejto časti sú diskutované jednotlivé časti aplikácie (viď kap. 6.2), výber technológii a ich vzájomná komunikácia.

### 7.1.1 Výpočetná časť aplikácie

Vzhľadom na výkonnostné nároky a nároky na podporu platforiem v nefunkčných požiadavkách bol ako jazyk implementácie pre aplikáciu v príkazovom riadku zvolený jazyk C++, konkrétne vo verzii C++11.

### 7.1.2 Grafické rozhranie

Grafická časť aplikácie nekladie žiadne požiadavky na výkon, podstatná je rýchla a pohodlná implementácia a taktiež priateľný vzhľad aplikácie na všetkých platformách. Preto bol na implementáciu grafického rozhrania zvolený jazyk Python vo verzii 2.7 spolu s grafickým frameworkom wxWidgets [29], ktorý je v jazyku Python k dispozícii vo forme knižnice wxPython [30].

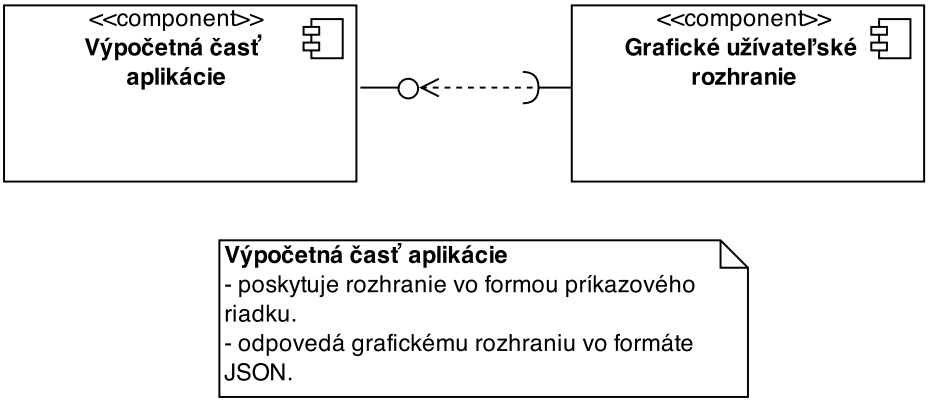
### 7.1.3 Komunikácia rozhraní aplikácie

Počas implementácie boli uvažované nasledovné možnosti pre komunikáciu výpočetnej časti a grafického rozhrania:

(1) dve samostatne stojace časti, použitie výpočetnej časti ako knižnice v grafickej časti,

(2) dve samostatne stojace časti, komunikácia jednotlivých častí pomocou príkazového riadku vo formáte JSON.

Nakoniec bola zvolená 2. možnosť (viď obr. 7.1). Tento spôsob riešenia nie je obvyklý, avšak vzhľadom na predpokladanú nízku frekvenciu komunikácie medzi výpočetnou častou a grafickým rozhraním je možné toto riešenie považovať za akceptovateľné.




Obr. 7.1: Diagram komponent ilustrujúci komunikáciu výpočetnej časti aplikácie a grafického rozhrania

## 7.2 Použité formáty pre reprezentáciu sústavy

### 7.2.1 Vonkajšia reprezentácia

Formát „Matrix Market“ je jeden z najznámejších súborových formátov na reprezentáciu riedkych matíc, čo dokazuje aj existujúci repozitár [31] obsahujúci matice v tomto formáte. Jednoduchosť a dostatok testovacích matíc sú dôvody pre voľbu práve tohto formátu. Matica sústavy bude reprezentovaná v súradnicovom formáte „Matrix Market“ (viď kap. 2.6.5.1) a vektor pravej strany v súborovom formáte „Matrix Market“ (viď kap. 2.6.5.2).

### 7.2.2 Vnútorná reprezentácia

Pre vnútornú reprezentáciu matice bola zvolená vylepšená verzia formátu „Compressed Column Storage“ (2.6.5.3). Tento formát bol zvolený z dôvodu

jeho jednoduchosti. Vektor reprezentujúci pravú stranu sústavy a riešenie, ktorý je vlastne hustá matica typu $(n, 1)$ je reprezentovaný poľom.

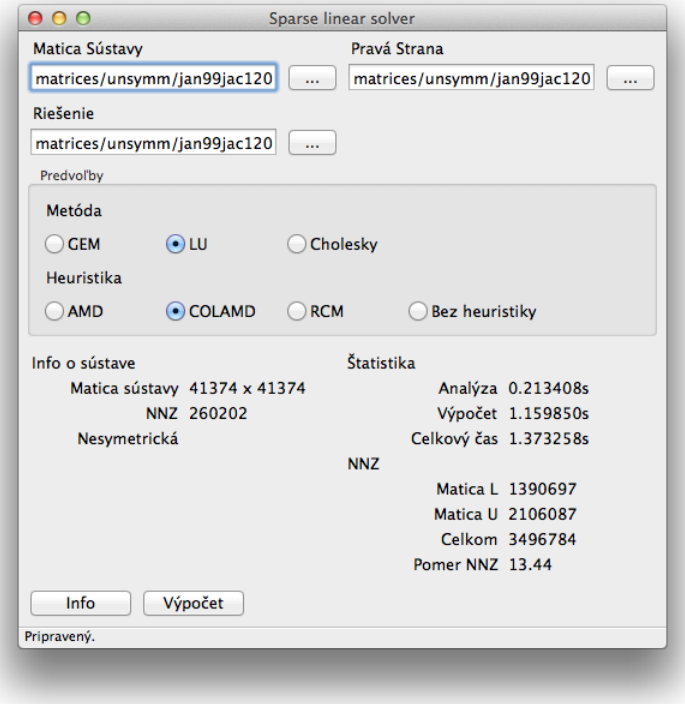
## 7.3 Tvorba aplikácie

### 7.3.1 Implementácia výpočetnej časti

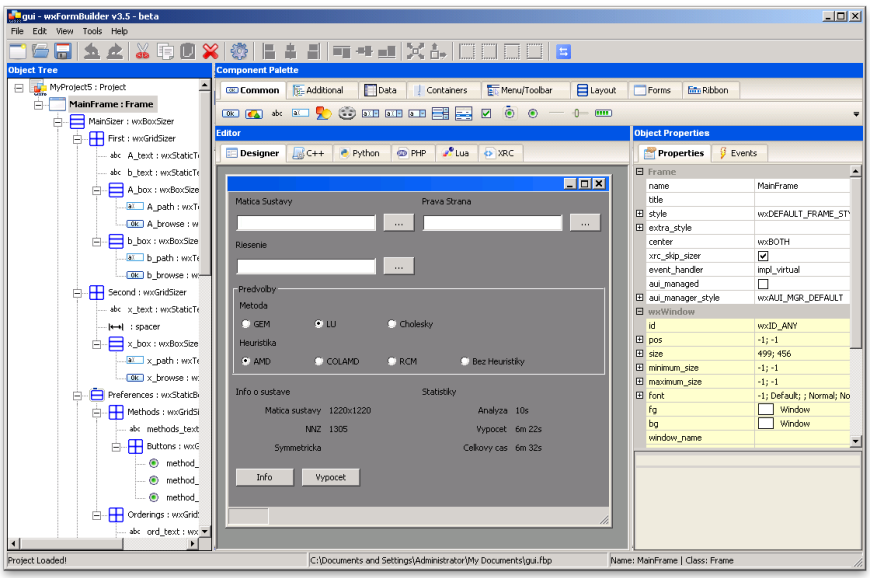
Základom implementácie výpočetnej časti sa stala knižnica Eigen [32], ktorá poskytuje široké možnosti pre prácu s riedkymi sústavami. Knižnica je dostupná na zvolenej implementačnej platforme a natívne podporuje zvolené formáty „Matrix Market“ a „Compressed Column Storage“.

### 7.3.2 Realizácia grafického rozhrania

Na základe wireframu (viď. obr. 6.4) bola vytvorená finálna podoba grafického rozhrania. Rozhranie bolo najprv navrhnuté a následne doplnené o funkčnú logiku a komunikáciu s výpočetnou časťou. GUI je ilustrované obrázkom 7.2.




Obr. 7.2: Grafické rozhranie vytvorenej aplikácie




Obr. 7.3: wxFormBuilder v3.5 beta [1]

## 7.4 Implementačné prostredie

Autor tejto práce obľubuje jednoduchosť a snaží sa vyhýbať zložitým vývojárskym prostrediam. Na písanie kódu bol použitý textový editor Sublime Text 2, na návrh užívateľského rozhrania bola použitá aplikácia wxFormBuilder v3.5 beta (viď obr. 7.3), ktorá dokáže z navrhnutého rozhrania vytvoriť kód podporovaný frameworkom wxWidgets v rôznych jazykoch (Python, C++, PHP, atď.). Pre odladovanie chýb vo výpočetnej časti aplikácie bol použitý memory debugger Valgrind. Na verzovanie a zálohu bol používaný systém Git.

## 7.5 Zhodnotenie realizácie

V tejto časti sú zhodnotené jednotivé požiadavky na aplikáciu a spôsob ich splnenia.

- **Rozhranie v príkazovom riadku a grafické užívateľské rozhranie**

  Aplikácia bola rozdelená na dva samostatné celky (viď kap. 6.2), výpočetnú časť s rozhraním v príkazovom riadku a GUI. Spôsob implementácie výpočetnej časti je popísaný v kap. 7.3.1. Grafická časť bola najprv navrhnutá v podobe wireframu (viď obr. 6.4) a následne prevedená do reálnej podoby a doplnená o funkčnú logiku (viď obr. 7.3.2).

- **Možnosť výberu Gaussovej eliminačnej metódy, LU rozkladu a Choleského rozkladu**

  Možnosť výberu jednotlivých metód bola prevedená do podoby funkčného požiadavku (viď kap. 6.1.2), ktorý je testovaný v poslednej kapitole (viď 8.3).

- **Výber z minimálne dvoch heuristík pre minimalizáciu dodatočného zaplnenia matice**

  Heuristiky boli po teoretickej stránke popísané v kap. 4, výber konkrétnych heuristík použitých v aplikácii je diskutovaný v kap. 6.3. Táto možnosť bola taktiež zaistená v podobe funkčného požiadavku (viď. kap. 6.1.2), ktorý je testovaný v poslednej kapitole (viď 8.3).

- **Dostupné rozhranie pre dodatočnú implementáciu ďalších metód a heuristík**

  Knižnica Eigen (viď kap. 7.3.1), ktorá bola zvolená ako základ výpočetnej časti aplikácie poskytuje pohodlné rozhranie pre implementáciu dodatočných metód a heuristík.

- **Zobrazenie informácii o danej sústave**

  Táto možnosť je zaistená funkčným požiadavkom (viď kap. 6.1.2) a testovaná v poslednej kapitole (viď 8.3).

- **Podpora vonkajšej reprezentácie vo formáte „Matrix Market“**

  Formát vonkajšej reprezentácie je popísaný v kap. 2.6.5.1, jeho podpora bola diskutovaná v kap. 7.2.1.

- **Vnútorná reprezentácia vo formáte „Compressed Column Storage“**

  Formát vnútornej reprezentácie je popísaný v kap. 2.6.5.3, použitie „Compressed Column Storage“ formátu bolo diskutované v kap. 7.2.2.

- **Podpora platforiem GNU/Linux a OS X**

  Podpora týchto platforiem bola jedným z argumentov pri výbere technológii (viď kap. 7.1). Testovací hardvér (viď kap. 8.2) bol zvolený tak, aby testovanie overilo podporu týchto platforiem.

- **Vhodná technológia pre vykonávanie výpočtov**

  Výpočetná časť bola implementovaná v jazyku C++ (viď. kap. 7.1), ktorý dáva dostatočný predpoklad na efektívne riešenie.

Kapitola 8

# Testovanie

## 8.1 Testovacie dáta

Ako zdroj testovacích dát boli zvolené repozitáre „Matrix Market“ [31] a kolekcia matíc „University of Florida“ [33]. Tieto repozitáre majú zvyčajne dostupné matice aj v iných formátoch ako je „Matrix Market“, čo bude výhoda pri porovnávaní aplikácie s existujúcimi riešeniami. V prípade, že matica neobsahovala vektor s pravou stranou, bol tento vektor náhodne vygenerovaný.

Na testovanie boli zvolené nasledujúce matice:

**Reálne, SPD**

(1) Dubcova2

- **Disciplína:** 2D/3D problém
- **Rozmery:** 65 025 × 65 025
- **nnz:** 1 030 225

(2) bcsstk17

- **Disciplína:** Inžinierske staviteľstvo
- **Rozmery:** 10 974 × 10 974
- **nnz:** 219 812

(3) bundle1

- **Disciplína:** Počítačová grafika
- **Rozmery:** 10 581 × 10 581
- **nnz:** 770 811

(4) finan512

- **Disciplína:** Ekonomický problém
- **Rozmery:** 74 752 × 74 752
- **nnz:** 596 992

(5) G2_circuit

- **Disciplína:** Simulácia obvodov
- **Rozmery:** 150 102 × 150 102
- **nnz:** 726 674

**Reálne, Nesymetrické**

(1) Baumann

- **Disciplína:** 2D/3D problém
- **Rozmery:** 112 211 × 112 211
- **nnz:** 748 331

(2) ecl32

- **Disciplína:** Polovodičové zariadenia
- **Rozmery:** 51 993 × 51 993
- **nnz:** 380 415

(3) fidapm11

- **Disciplína:** Modelovanie konečných prvkov
- **Rozmery:** 22 294 × 22 294
- **nnz:** 623 554

(4) g7jac200

- **Disciplína:** Ekonomický problém
- **Rozmery:** 59 310 × 59 310
- **nnz:** 717 620

(5) rajat23

- **Disciplína:** Simulácia obvodov
- **Rozmery:** 110 355 × 110 355
- **nnz:** 555 441

**Pozn.** Na ilustráciu fungovania niektorých funkčných požiadavkov boli vytvorené malé sústavy. Tieto sústavy sú popísané pri ich použití v jednotlivých testoch.

## 8.2 Testovací hardvér

Testovanie prebiehalo na nasledujúcom hardvéry:

(1) Laptop

- **CPU:** Intel(R) Core(TM) i5-3427U @ 1.80GHz
- **RAM:** 4 GiB
- **HDD:** 1x SSD 128 GB
- **OS:** OS X Mavericks

(2) Server

- **CPU:** 2x Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
- **RAM:** 48 GiB
- **HDD:** 2x SSD 120 GB v RAID 1
- **OS:** Debian GNU/Linux 8 (jessie)

## 8.3 Testovanie funkčných požiadavkov

Testovanie funkčných požiadavkov (viď kap. 6.1.2) prebiehalo s využitím testovacích matíc. Uvedené vstupy a výstupy nemajú presnú formu používania aplikácie, avšak význam zostáva bez zmeny. Chovanie rozhraní je v zásade totožné a je možné mať za to, že nasledujúce testy a k ním uvádzaný výstup je rovnaký pri obidvoch rozhraniach. Prípadné odlišnosti týchto rozhraní sú explicitne uvedené.

### 8.3.1 (F1) Zobrazenie informácii o sústave

**Test 1**

**Vstupné parametre:**

- Vstupná matica + pravá strana: Baumann
- Voľba: Zobrazenie informácii o sústave

**Výstup:** Matica sústavy je typu...

**Záver:** Funkčný požiadavok je splnený.

### 8.3.2 (F2) Povinná voľba metódy na riešenie riedkej sústavy lineárnych rovníc

**Test 1**

**Vstupné parametre:**

- Vstupná matica + pravá strana: fidapm11
- Voľba: Riešenie sústavy

**Výstup:** Nebola zvolená metóda na riešenie sústavy.

**Test 2**

**Vstupné parametre:**

- Vstupná matica + pravá strana: g7jac200
- Metóda: GEM
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 3**

**Vstupné parametre:**

- Vstupná matica + pravá strana: rajat23
- Metóda: LU rozklad
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 4**

**Vstupné parametre:**

- Vstupná matica + pravá strana: Dubcova2
- Metóda: Choleského rozklad
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Záver:** Funkčný požiadavok je splnený.
**Pozn.** V grafickom rozhraní je predvolená možnosť LU rozkladu a nie je možné nevybrať žiadnu metódu.

### 8.3.3 (F3) Možnosť voľby heuristiky na minimalizáciu dodatočného zaplnenia matice

**Test 1**

**Vstupné parametre:**

- Vstupná matica + pravá strana: ecl32
- Predvoľby: LU rozklad + AMD
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 2**

**Vstupné parametre:**

- Vstupná matica + pravá strana: rajat23
- Predvoľby: Gaussova eliminačná metóda + COLAMD
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 3**

**Vstupné parametre:**

- Vstupná matica + pravá strana: bundle1
- Predvoľby: LU rozklad + RCM
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 4**

**Vstupné parametre:**

- Vstupná matica + pravá strana: G2_circuit
- Predvoľby: Choleského rozklad + Bez heuristiky
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Záver:** Funkčný požiadavok je splnený.

### 8.3.4 (F4) Kontrola vstupných dát

**Test 1**

**Vstupné parametre:**

- Matica s nevalidným formátom (v súbore chýba 1 reprezentovaný element)
- Metóda: LU rozklad
- Voľba: Riešenie sústavy

**Výstup:** Matica nemá validný formát.

**Test 2**

**Vstupné parametre:**

- Matica $2 \times 3$ + pravá strana
- Metóda: LU rozklad
- Voľba: Riešenie sústavy

**Výstup:** Matica sústavy musí byť štvorcová.

**Test 3**

**Vstupné parametre:**

- Matica $2 \times 2$ + pravá strana $5 \times 1$
- Metóda: LU rozklad
- Voľba: Riešenie sústavy

**Výstup:** Vektor s pravou stranou nemá zhodný počet prvkov so šírkou matice sústavy.

**Záver:** Funkčný požiadavok je splnený.

### 8.3.5 (F5) Overenie kompatibility zvolenej metódy s riešenou sústavou

**Test 1**

**Vstupné parametre:**

- Nesymetrická matica $5 \times 5$ + pravá strana
- Metóda: Choleského rozklad
- Voľba: Riešenie sústavy

**Výstup:** Nie je možné použiť Choleského rozklad, matica nie je symetrická.

**Test 2**

**Vstupné parametre:**

- SPD matica $3 \times 3$ + pravá strana
- Metóda: Choleského rozklad
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Záver:** Funkčný požiadavok je splnený.

### 8.3.6 (F6) Výpočet riedkej sústavy lineárnych rovníc

Tento požiadavok už bol otestovaný pri testovaní požiadavku F3 a ďalších.

### 8.3.7 (F7) Možnosť zobrazenia štatistík o priebehu výpočtu

**Test 1**

**Vstupné parametre:**

- Matica ecl32 + pravá strana
- Metóda: LU rozklad
- Voľba: Riešenie sústavy

**Výstup:** Výpočet bol úspešný.

**Test 2**

**Vstupné parametre:**

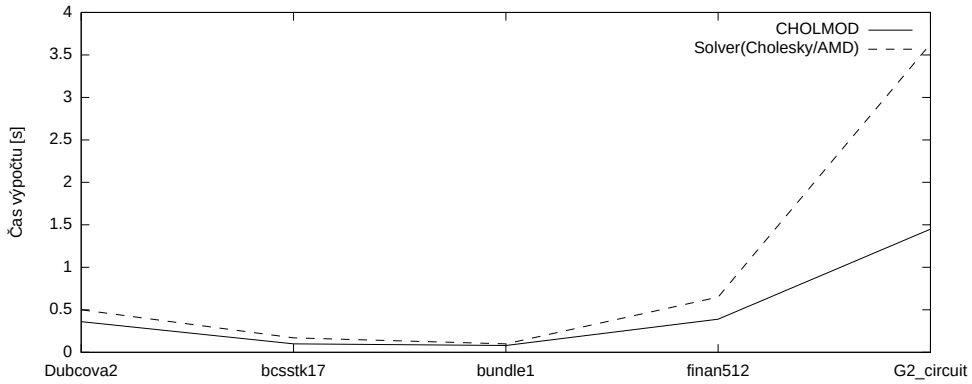
- Matica ecl32 + pravá strana
- Metóda: LU rozklad
- Voľba: Riešenie sústavy + zobrazenie štatistik

**Výstup:** Výpočet bol úspešný + štatistika.

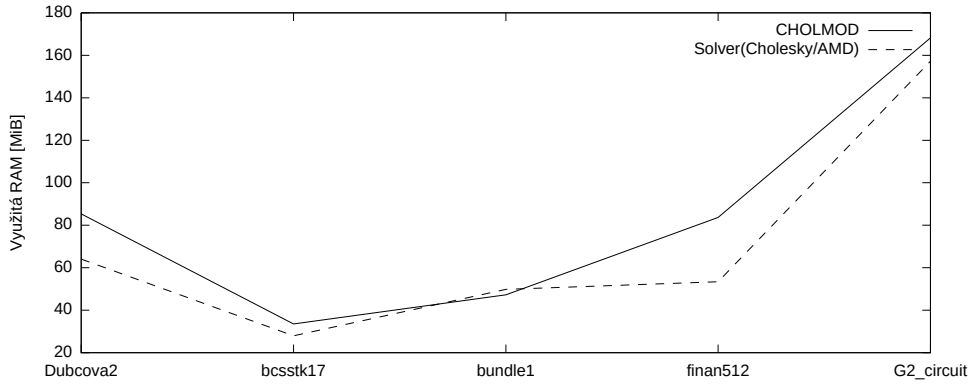
**Záver:** Funkčný požiadavok je splnený.
**Pozn.** V grafickom rozhraní sú štatistiky zobrazené vždy.

## 8.4 Porovnanie s existujúcimi riešeniami

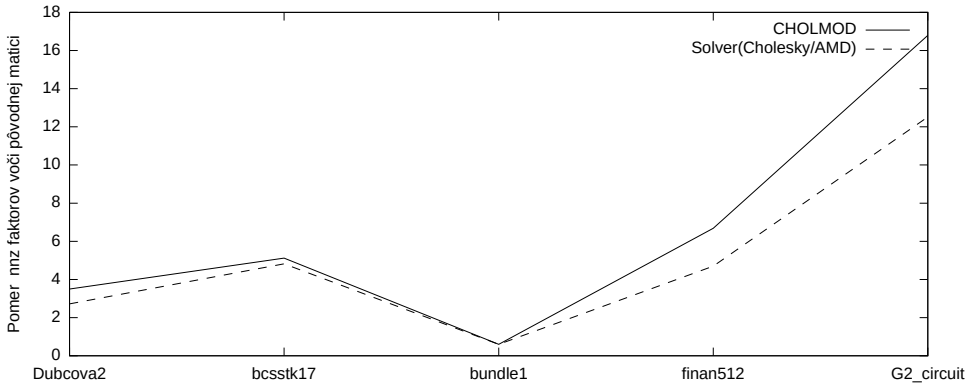
Nasledujúci text porovnáva vyvinutú aplikáciu s existujúcimi riešeniami. Pre porovnanie boli použité matice zvolené v kap. 8.1. Toto porovnanie je rozdelené podľa symetrie matice a vyvinuté riešenie je v nich označené pod pojmom „Solver(metóda/heuristika)“.




Obr. 8.1: Porovnanie podľa dĺžky trvania výpočtu




Obr. 8.2: Porovnanie podľa spotreby operačnej pamäte



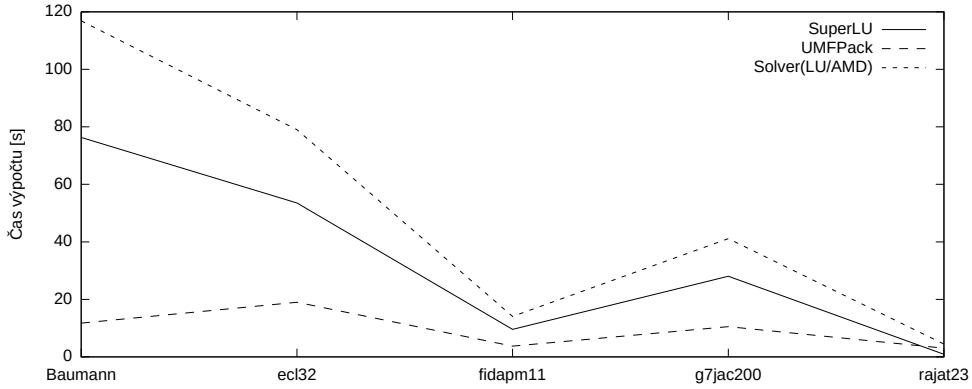

Obr. 8.3: Porovnanie podľa pomeru nnz faktorov (**L** a **U**) voči pôvodnej matici.

### 8.4.1 Symetrické matice

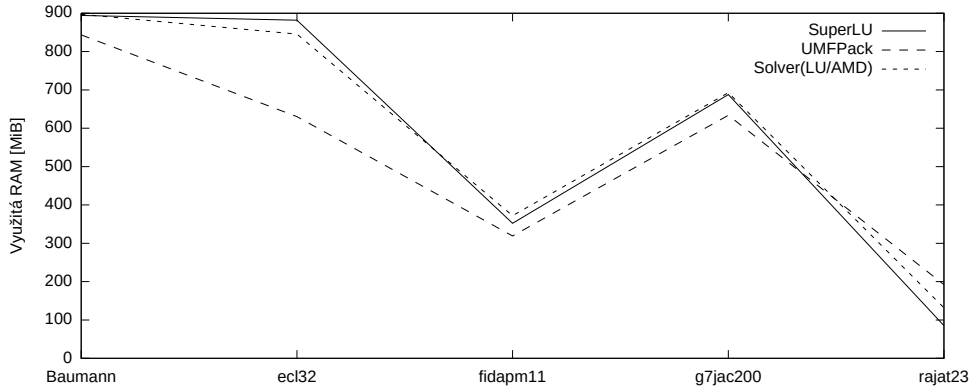
Grafy obr. 8.1, obr .8.2 a obr. 8.3 porovnávajú efektívnosť vyvinutej aplikácie s riešením „CHOLMOD“, ktoré bolo popísané v 5.2.

### 8.4.2 Nesymetrické matice

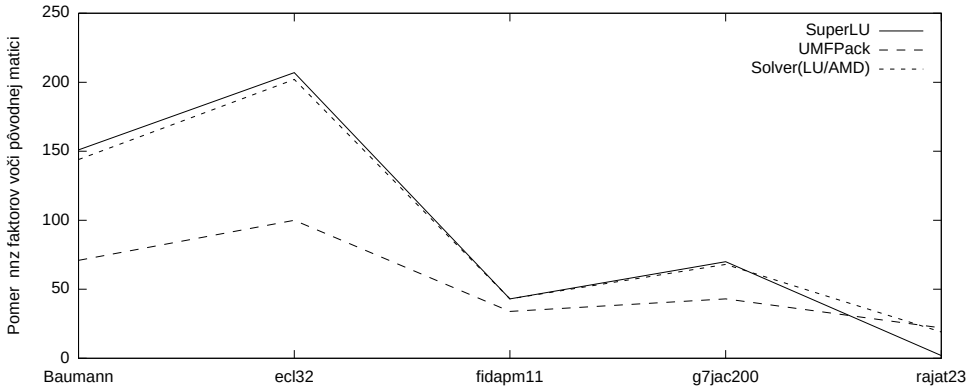
Grafy obr. 8.4, obr. 8.5 a obr. 8.6 porovnávajú efektívnosť vyvinutej aplikácie pri počítaní sústav s nesymetrickými maticami s riešeniami SuperLU (popísané v kap. 5.1) a UMFPack (popísané v kap. 5.3).




Obr. 8.4: Porovnanie podľa dĺžky trvania výpočtu




Obr. 8.5: Porovnanie podľa spotreby operačnej pamäte




Obr. 8.6: Porovnanie podľa pomeru nnz faktoru ($\mathbf{L}$) voči pôvodnej matici.

# Záver

Táto práca sa zaoberala problematikou riedkych sústav lineárnych rovníc a ich riešením pomocou priamych metód. V práci boli popísané vybrané metódy, heuristiky a existujúce riešenia s cieľom návrhu vlastnej aplikácie s grafickým rozhraním a rozhraním v príkazovom riadku. Hlavným cieľom práce bolo navrhnúť kompaktné riešenie, ktoré sa bude snažiť ponúknuť možnosť optimálnej voľby pre ľubovolný typ riedkej sústavy. Vytvorená aplikácia ponúka na výber Gaussovu eliminačnú metódu, metódu LU rozkladu a Choleského metódu a heuristiky AMD, COLAMD a RCM určené na minimalizáciu dodatočného zaplnenia matice. Testovanie (viď kap. 8.4) ukázalo, že aplikácia by mala byť minimálne pre niektoré sústavy schopná konkurovať existujúcim riešeniam. Napriek snahe o široký záber vyvinutej aplikácie nie je možné tvrdiť, že ponúka efektívne riešenie pre všetky typy riedkych sústav. Avšak ďalšími vylepšeniami vyvinutej aplikácie by bolo možné k takémuto cieľu postupne dospieť.

Pokračovanie tejto práce by bolo možné hneď niekoľkými smermi. Vytvorená implementácia a použitá knižnica Eigen [32] zaručene poskytujú priestor na zlepšenie. Vylepšením knižnice Eigen by sa nepomohlo len tejto aplikácii, ale aj ďalším aplikáciám a tým aj ich užívateľom, ktorí používajú riešenia postavené na tejto knižnici a taktiež komunite, ktorá spomínanú knižnicu vyvíja. Táto aplikácia má v predpokladoch možnosť dodatočnej implementácie ďalších metód a heuristík, čo ponúka možnosť pokračovania tejto práce ich implementovaním, prípadne možnosť doplnenia aplikácie o ďalšie formy vnútornej a vonkajšej reprezentácie stojí tiež za zmienku. Za najzaujímavejšiu možnosť osobne považujem rozšírenie tejto aplikácie o iteračné metódy, ktorými sa táto práca nezaoberala.

# Literatúra


[1] wxFormBuilder - a RAD tool for wxWidgets GUI design. online, [cit. 2015-05-07]. Dostupné z: http://sourceforge.net/projects/wxformbuilder/

[2] Olšák, P.: *Úvod do algebry, zejména lineární*. Fakulta elektrotechnická ČVUT v Praze, první vydání, 2007, ISBN 978-80-01-03775-1.

[3] František Štampach, K. K.: Materiály k predmetu BI-LIN. online, 2014. Dostupné z: http://users.fit.cvut.cz/~stampfra/lectures/lin/lin-prednaska-7-handout.v1.pdf

[4] Weisstein, E. W.: MathWorld–A Wolfram Web Resource: Permutation Matrix. online, [cit. 2015-04-02]. Dostupné z: http://mathworld.wolfram.com/PermutationMatrix.html

[5] Weisstein, E. W.: Scientic Computing: Numerical Linear Algebra. online, [cit. 2015-04-28]. Dostupné z: http://web.stanford.edu/~mzahr/extras/cme292/autumn2014/lec/lec03.pdf

[6] Tůma, M.: Direct Methods for Sparse Matrices. online, [cit. 2015-05-05]. Dostupné z: http://www2.cs.cas.cz/~tuma/ps/direct.pdf

[7] Boisvert, R. F.; Pozo, R.; Remington, K. A.: The Matrix Market Formats: Initial Design. online, 1996. Dostupné z: http://math.nist.gov/MatrixMarket/reports/MMformat.ps

[8] Eigen: Sparse matrix manipulations. online, [cit. 2015-04-25]. Dostupné z: http://eigen.tuxfamily.org/dox/group__TutorialSparse.html

[9] Lecture 12: LU Decomposition. online, [cit. 2015-04-27]. Dostupné z: https://www.math.ohiou.edu/courses/math3600/lecture12.pdf

[10] 6. Cholesky factorization. online, [cit. 2015-04-27]. Dostupné z: http://www.dam.brown.edu/people/mchb/la/lu.pdf

[11] Alan Turing and the origins of modern Gaussian elimination. online, [cit. 2015-04-27]. Dostupné z: http://gauss.uc3m.es/web/personal_web/fdopico/papers/arbor-2011-turing.pdf

[12] Graph Theory. online, [cit. 2015-04-14]. Dostupné z: https://www.math.ust.hk/~mabfchen/Math2343/Graph-General.pdf

[13] Sparse Matrices. online, [cit. 2015-04-27]. Dostupné z: http://www.di.unipi.it/~lgemignani/sparse.pdf

[14] George, A.; Liu, J. W. H.: The Evolution of the Minimum Degree Ordering Algorithm. online, 2011. Dostupné z: http://web.ornl.gov/info/reports/1987/3445602624257.pdf

[15] Amestoy, P. R.; Davis, T. A.; Duff, I. S.: An Approximate Minimum Degree Ordering Algorithm. online, 1996. Dostupné z: http://faculty.cse.tamu.edu/davis/publications_files/An_Approximate_Minimum_Degree_Ordering_Algorithm.pdf

[16] Cuthill, E.; McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In *ACM '69 Proceedings of the 1969 24th national conference*, 1969, s. 157–172. Dostupné z: http://dl.acm.org/citation.cfm?id=805928

[17] Lecture 4: Breadth-First Search. 2011, [cit. 2015-04-18]. Dostupné z: http://www.fas.harvard.edu/~libcs124/CS/lec4.pdf

[18] George, J. A.; Liu, J. W.-H.: *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981, ISBN 01-3165-274-5.

[19] Gibbs, N. E.; Poole, W. G.; Jr.; aj.: An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. In *SIAM Journal on Numerical Analysis*, 1976, s. 236–250. Dostupné z: http://www.jstor.org/stable/2156090

[20] Østerby, O.; Zlatev, Z.: *Lecture Notes in Computer Science: Direct Methods for Sparse Matrices*. 1983, ISBN 3-540-12676-7.

[21] Gilbert, J. R.; Moler, C.; Schreiber, R.: Sparse Matrices in MATLAB: Design and Implementation. In *SIAM Journal on Matrix Analysis and Applications 13*, 1992, ISSN 1095-7162, s. 333–356.

[22] Davis, T. A.; Gilbert, J. R.; Larimore, S. I.; aj.: Column Approximate Minimum Degree Ordering Algorithm. online, 2000, [cit. 2015-04-18]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.9285&rep=rep1&type=pdf

[23] Li, X. S.; Demmel, J. W.; Gilbert, J. R.; aj.: SuperLU Users' Guide. online, 2011. Dostupné z: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/superlu_ug.pdf

[24] Davis, T. A.: User Guide for CHOLMOD: a sparse Cholesky factorization and modification package. online, 2014. Dostupné z: https://www.cise.ufl.edu/research/sparse/cholmod/CHOLMOD/Doc/UserGuide.pdf

[25] METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering. online, 2003, [cit. 2015-05-09]. Dostupné z: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[26] Davis, T. A.: UMFPACK User Guide. online, 2013. Dostupné z: https://www.cise.ufl.edu/research/sparse/umfpack/UMFPACK/Doc/UserGuide.pdf

[27] Kundert, K.: Sparse User's Guide. online, 2003. Dostupné z: http://sparse.sourceforge.net/spDoc.pdf

[28] Sparse Reference Manual. online, 2003. Dostupné z: http://sparse.sourceforge.net/refman.pdf

[29] wxWidgets: Cross-Platform GUI Library. online, [cit. 2015-04-25]. Dostupné z: http://www.wxwidgets.org

[30] wxPython. online, [cit. 2015-04-25]. Dostupné z: http://www.wxpython.org/what.php

[31] Matrix Market. online, [cit. 2015-04-25]. Dostupné z: http://math.nist.gov/MatrixMarket/index.html

[32] Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. online, [cit. 2015-05-09]. Dostupné z: http://eigen.tuxfamily.org/

[33] Davis, T. A.: The University of Florida Sparse Matrix Collection. online, [cit. 2015-04-26]. Dostupné z: http://www.cise.ufl.edu/research/sparse/matrices/

[34] Doxygen: Generate documentation from source code. online, [cit. 2015-05-10]. Dostupné z: http://www.stack.nl/~dimitri/doxygen/

Dodatok **A**

# Zoznam použitých skratiek

**AMD** Approximate Minimum Degree Ordering

**CLI** Command Line Interface

**COLAMD** Column Approximate Minimum Degree Ordering

**GEM** Gaussian Elimination Method

**GUI** Graphical User Interface

**JSON** JavaScript Object Notation

**RCM** Reverse Cuthill-McKee

**SLR** Sústava lineárnych rovníc

**SPD** Symmetric Positive Definite

Dodatok B

# Inštalačná a užívateľská príručka

Nasledujúci text platí pre systémy OS X Mavericks a Debian GNU/Linux 8 (jessie) na ktorých bola aplikácia testovaná. V prípade iných systémov sa môže postup mierne líšiť.

## B.1 Preklad výpočetnej časti aplikácie

Program preložíme zadaním príkazu *make* v koreňovom adresári programu. Po preložení by mal vzniknúť súbor *solver* predstavujúci výpočetnú časť aplikácie.

Správne sprevádzkovanie je možné otestovať príkazom *make test*, ktorý spustí výpočet s testovacou sústavou. Výsledkom by malo byť riešenie sústavy uložené do súboru a zobrazené štatistiky o priebehu výpočtu. Po skončení práce je možné vrátiť adresár do pôvodného stavu príkazom *make clean*.

## B.2 Práca s výpočetnou časťou

Výpočetnú časť spúšťame príkazom *./solver* a môžme ju použiť nasledujúcimi spôsobmi:

(1) Zobrazenie informácii o sústave

*./solver -d -A <matrix.mtx> -b <vektor.mtx>*

(2) Vyriešenie sústavy lineárnych rovníc

*./solver -s -i -m <metóda> -o <heuristika> -A <matica.mtx> -b <vektor.mtx> -x <riešenie.mtx>*

**Pozn.** Prepínače *-i* a *-o* nie sú povinné.

Význam uvedených prepínačov je nasledovný:

- ***-s*** riešiť sústavu (predvolené)
- ***-i*** zobraziť informácie o sústave
- ***-m*** výber metódy: gem (3.1), lu (3.2), chol (3.3)
- ***-o*** výber heuristiky: no (bez heuristiky - predvolené), amd (4.2.3), colamd (4.3.3), rcm (4.2.5)
- ***-A*** cesta k matici sústavy v súradnicovom formáte Matrix Market (2.6.5.1)
- ***-b*** cesta k pravej strane sústavy v súborovom formáte Matrix Market (2.6.5.2)
- ***-x*** súbor do ktorého sa uloží riešenie sústavy
- ***-i*** zobrazenie štatistík o priebehu výpočtu (voliteľné)
- ***-h*** zobrazí spôsob používania programu

## B.3 Sprevádzkovanie grafického rozhrania

Na začiatok je potrebné nainštalovať

- grafický framework wxWidgets [29] a
- knižnicu wxPython [30] (zaisťuje komunikáciu s wxWidgets v Pythone).

Po ich nainštalovaní spustíme grafické rozhranie príkazom *python solver.py*.

## B.4 Vygenerovanie dokumentácie

Pre vygenerovanie dokumentácie je potrebná aplikácia doxygen [34]. Po jej nainštalovaní vygenerujeme dokumentáciu pomocou príkazu *make doc*.

V adresári s aplikáciou sa vytvorí adresár *doc*, ktorý obsahuje adresáre *cli* a *gui* s dokumentáciou výpočetnej časti a grafického rozhrania.

Dodatok C

# Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `app` ..... vyvinutá aplikácia
  - `matrices` ..... zbierka sústav používaných pri testovaní
  - `solver.py` ..... grafické rozhranie vytvorenej aplikácie
  - `src` ..... zdrojový kód výpočetnej časti
- `text` ..... text práce
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
  - `thesis.pdf` ..... text práce vo formáte PDF