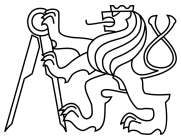
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Paralelní iterační řešič soustav lineárních rovnic pomocí řezů v grafu
Student:	Ondřej Brožek
Vedoucí:	Ing. Ivan Šimeček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	do konce zimního semestru 2016/17

Pokyny pro vypracování

1) Nastudujte metody pro iterační řešení lineárních rovnic (Jacobi, Gauss-Seidel a Conjugate Gradient).
2) Implementujte paralelní verze algoritmů pro tyto metody pomocí OpenMP technologie.
3) Nastudujte algoritmy pro konstrukci hranového a uzlového řezu.
4) Implementujte heuristické verze těchto algoritmů vhodné pro velké řídké grafy.
5) Implementujte řešič založený na metodě "nested dissection".
6) Porovnejte přesnost a výkonnost algoritmů z bodů 2) a 5).

Seznam odborné literatury

http://en.wikipedia.org/wiki/Nested_dissection

Zbytek dodá vedoucí práce

L.S.

doc.Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof.Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 25. února 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Paralelní iterační řešič soustav lineárních rovnic pomocí řezů v grafu

Ondřej Brožek

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

9. ledna 2017

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2017 .....

Odkaz na tuto práci

Brožek, Ondřej. *Paralelní iterační řešič soustav lineárních rovnic pomocí řezů v grafu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce popisuje různé způsoby, jak lze řešit soustavy lineárních rovnic, které vedou na řídké matice. V práci jsou popsány iterační metody a metoda hranového řezu. Jako iterační metody byly zvoleny Jacobi, Gauss-Seidel a Conjugate Gradient. V práci je bližší seznámení s metodou hranového řezu a je implementována její paralelní verze pomocí technologie OpenMP. Implementace algoritmů byly otestovány na vhodné sadě lineárních rovnic a byla porovnána jejich výkonost.

Klíčová slova řídká matice, hranový řez, iterační metody

Abstract

The thesis describes the different ways you can solve systems of linear equations that lead to the sparse matrices. The thesis describes numerical methods and method of nested dissection. As numerical methods were chosen Jacobi, Gauss-Seidel and the Conjugate Gradient. The nested dissection is described in more detail. The implementation of nested dissection method is parallelized with OpenMP technology. The implementation of algorithms were tested on a suitable set of linear equations and tested their performance.

Keywords sparse matrix, nested dissection, iterative methods

Obsah

Seznam obrázků

Úvod

Pro řešení soustav lineárních rovnic existuje mnoho různých algoritmů. Cílem práce je představit si některé z nich a porovnat jejich výkonnost.

Tématem práce je řešení velkých soustav lineárních rovnic vedoucích na řídké matice. Gaussova eliminační metoda je asi nejznámější způsob, jak soustavy lineárních rovnic řešit. Tato metoda vede vždy ke správnému řešení, ale pro tento typ soustav se bohužel nehodí. V takovém případě obvykle využíváme metody iterační.

Iterační metody nemění vstupní matici a obvykle nepotřebují mnoho paměti navíc. Nevýhodou těchto metod může být nepřesnost výsledku a fakt, že metody v některých případech ke správnému výsledku nekonvergují.

Další možnost, jak soustavu řešit, je metoda hranového řezu. Metoda využívá prohazování sloupců a řádků matice tak, aby se její části daly řešit současně. Nalezení nejvhodnějšího prohození sloupců a řádků není jednoduché a může být pomalejší, než kdybychom celou soustavu řešili pomocí Gaussovy eliminační metody.

V práci jsou popsány iterační metody a metoda hranového řezu. Je porovnána jejich výkonnost a přesnost výsledků na vhodné sadě soustav lineárních rovnic.

Kapitola 1

Teorie

1.1 Soustava lineárních rovnic

Uvažujme systém lineárních rovnic.

$$A\vec{x} = \vec{b} \tag{1.1}$$

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \vec{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix},$$

kde A je regulární čtvercová matice řádu n, $\vec{b}$ je vektor pravých stran lineárnách rovnic a $\vec{x}$ je vektor proměnných. Řešení soustavy lineárních rovnic dostaneme vynásobením rovnice inverzní maticí A

$$\vec{x} = A^{-1}\vec{b}. \tag{1.2}$$

V případě řešení rovnic vedoucích na řídké matice je nevhodné používat přímé metody. Přímé metody často potřebují další paměť, nebo jejich úpravy vedou na doplnění nenulových čísel na pozice, kde na počátku byla nula (např. Gaussova eliminační metoda). V takových případech je výhodnější využít metody iterační.

1.2 Spektrální poloměr

Spektrální poloměr pomáhá určit, jestli iterační metoda konverguje k výsledku. Obvykle se značí řeckým písmenem ρ. Spektrální poloměr je největší číslo z vlastních čísel matice.

$$\rho(A) = max(|\lambda_1|, ..., |\lambda_n|) \tag{1.3}$$

1.3 Konvergence iteračních metod

Iterační metody konvergují pro jakýkoliv počáteční tip právě tehdy, když

$$\rho(A) < 1 \tag{1.4}$$

Jednotlivé metody mohou konvergovat, přestože tato podmínka není splněna, ale je nutné vhodně zvolit počáteční tip.

Kapitola 2

Iterační metody

2.1 Úvod do iteračních metod

Upravou vzorce 1.1 dostaneme

$$\vec{x}^{i+1} = B^i \vec{x}^i + \vec{c}^i \tag{2.1}$$

kde $\vec{x}^{i+1}$ je vektor řešení následující iterace, B^i je matice specifická pro použitou metodu, $\vec{x}^i$ je současný vektor řešení (případně počáteční tip).

Pokud se matice B a vektor $\vec{c}$ nemění během iterací ($B^{i+1} = B^i, \vec{c}^{i+1} = \vec{c}^i$), pak se jedná o stacionární iterační metodu. V opačném případě se jedna o metodu nestacionární.

2.2 Jacobi

[1]Jacobiho metoda spočívá v postupném dosazování dílčího vektoru řešení. Matici A přepíšeme do tvaru $A = D + Z$, kde

$$D = \begin{pmatrix} a_{1,1} & & 0 \\ & \ddots & \\ 0 & & a_{n,n} \end{pmatrix} \text{ je diagonální matice}$$

$$Z = \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} \text{ je matice A s nulami na diagonále.}$$

Vyjádříme vektor $\vec{x}$

$$\begin{aligned} A\vec{x} &= \vec{b} \\ (D+Z)\vec{x} &= \vec{b} \\ D\vec{x}+Z\vec{x} &= \vec{b} \\ D\vec{x} &= -Z\vec{x}+\vec{b} \\ \vec{x} &= \underbrace{-D^{-1}Z}_{B}\vec{x}+\underbrace{D^{-1}\vec{b}}_{\vec{c}} \end{aligned}$$

Výsledný vektor je určen rovnicí

$$\vec{x}^{k+1} = D^{-1}(-Z\vec{x}^{k}+\vec{b})$$

kde $\vec{x}^{0}$ je vektor počátečního odhadu řešení soustavy. Po složkách výsledného vektoru má rovnice zápis

$$x_i^{k+1} = -\sum_{\substack{i\neq j \\ j=1}}^{n} \frac{a_{i,j}}{a_{i,i}} x_j^k + \frac{b_i}{a_{i,i}}, i = 1, ..., n; k \geq 0 \tag{2.2}$$

2.3 Gauss-Seidel

[1]Gauss-Seidel je velice podobná metodě Jacobi. Zatímco v Jacobi metodě držíme v paměti celý vektor řešení a počítáme z něj následující, v Gauss-Seidel ukládáme výsledky přímo do aktualního vektoru řešení. Tím se tato metoda dokáže rychleji přibližovat ke správnému výsledku.

$$x_i^{k+1} = -\sum_{j=1}^{i-1} \frac{a_{i,j}}{a_{i,i}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{i,j}}{a_{i,i}} x_j^k + \frac{b_i}{a_{i,i}}, i = 1, ..., n; k \geq 0 \tag{2.3}$$

2.4 Conjugate gradient (metoda sdružených gradientů)

[2]Metoda sdružených gradientů je složitější než predchozí dvě. Práce se metodou detailněji nezábývá. Níže je pouze popsaný použitý algoritmus na vyřešení soustavy lineárních rovnic.

$\vec{r}_0 = b - A\vec{x}_0$
$\vec{p}_0 = \vec{r}0$
$k = 0$
while true **do**
    $\alpha_k = \frac{\vec{r}_k^T \vec{r}_k}{\vec{p}_k A \vec{p}_k}$
    $\vec{x}_k + 1 = \vec{x}_k + \alpha_k \vec{p}_k$
    $\vec{r}_{k+1} = \vec{r}_k - \alpha_k A \vec{p}_k$
    if $\vec{r}_{k+1} jedost$ **then**
        Break While
    $\beta_k = \frac{\vec{r}_{k+1}^T \vec{r}_{k+1}}{\vec{r}_k^T \vec{r}_k}$
    $\vec{p}_{k+1} = \vec{r}_{k+1} + \beta_k \vec{p}_k$
    $k = k + 1$

Kapitola 3

Metoda hranového řezu

3.1 Úvod

[3]Metoda prohazuje sloupce a řádky matice tak, aby její časti byly řešitelné nezávisle na sobě. Cílem je dosáhnout následujícího tvaru matice

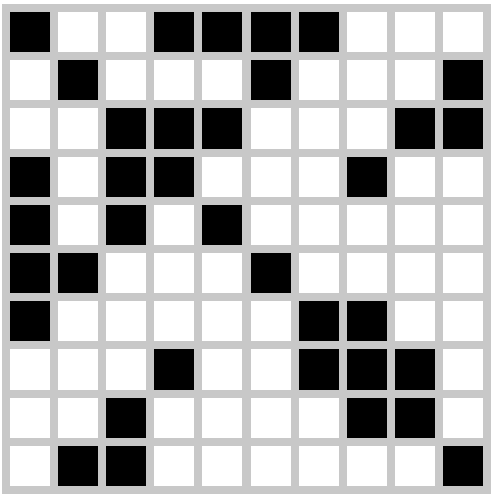
$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,k} & & & & s_{1,l+1} & \cdots & s_{1,n} \\ \vdots & \ddots & \vdots & & 0 & & \vdots & & \vdots \\ a_{k,1} & \cdots & a_{k,k} & & & & \vdots & & \vdots \\ & & & b_{k+1,k+1} & \cdots & b_{k+1,l} & \vdots & & \vdots \\ & 0 & & \vdots & \ddots & \vdots & \vdots & & \vdots \\ & & & b_{l,k+1} & \cdots & b_{l,l} & \vdots & & \vdots \\ s_{l+1,1} & \cdots & \cdots & \cdots & \cdots & \cdots & s_{l+1,l+1} & \cdots & s_{l+1,n} \\ \vdots & & & & & & \vdots & \ddots & \vdots \\ s_{n,1} & \cdots & \cdots & \cdots & \cdots & \cdots & s_{n,l+1} & \cdots & s_{n,n} \end{pmatrix}, \quad (3.1)$$

kde a a b jsou části matice, které je možné řešit současně a s jsou prvky množiny separátorů.

Pro soustavu rovnic o n neznámých, existuje $n!$ pertmutací řádků a sloupců. Metoda by měla najít takovou permutaci, kde části matice a a b mají přibližně stejně řádků a zároveň část separátorů má řádků co nejméně. Jakmile se podaří najít vhodné rozdělení matice, můžeme metodu znovu použít na její části a a b. Jedná se o algoritmus rozděl a panuj.

3.2 Vytvoření grafu

Předpokladem je, že matice je čtvercová a symetrická. Vygenerovní grafu se dá lépe představit díky matici sousednosti, která připomíná reprezentaci



Obrázek 3.1: Reprezentace nenulových čísel v matici

grafu. Za každý řádek a sloupec se stejným indexem vytvoříme uzel a přiřadíme mu stejný index. Pro každé nenulové číslo v matici vytvoříme hranu mezi uzly. Hrany začínají / končí v uzlech se stejným indexem, jako je index řádku / sloupce nenulového čísla. Vzhledem k faktu, že vstupní matice je symetrická, není nutné značit orientaci hran. Smyčky, které mohou vzniknout z nenulových čísel na diagonále, zanedbáme.

Obrázky 3.1 a 3.2 ilustrují transformaci jednoduché matice do grafu. Černá políčka reprezentují nenulová čísla v matici.

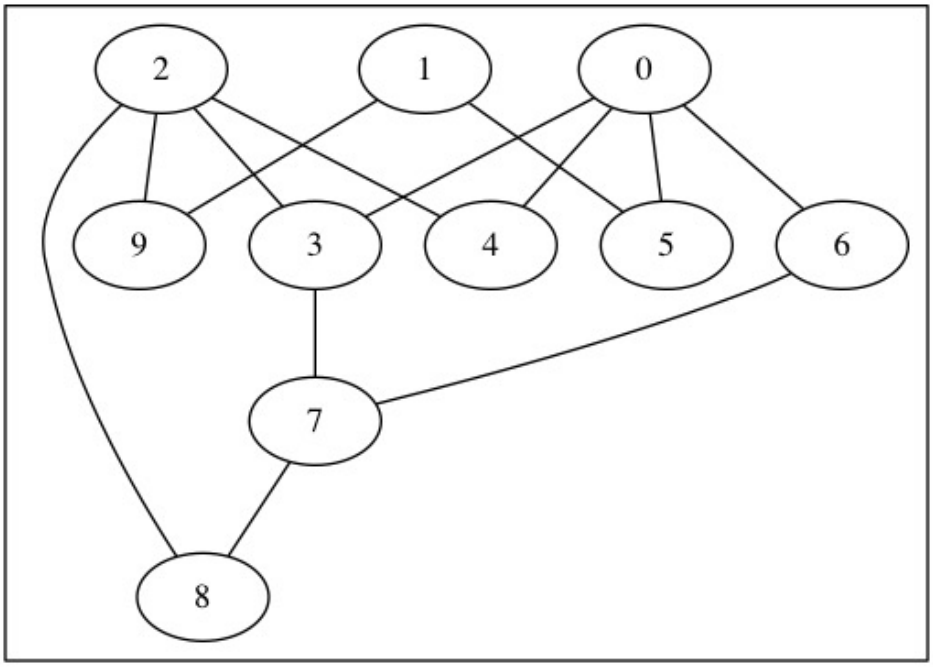
3.3 Nalezení množiny separátorů

[4][5]Hledání vhodného rozdělení grafu na jednotlivé části není jednoduchý problém. V ideálním případě by bylo nejlepší otestovat všechna možná rozdělení grafu a vybrat nejlepší. Takový postup je výpočetně příliš náročný není možné jej použít.

Mnoho prací se zabívá pouze vstupy, které vedou na planární graf. V této práci je použité obecné řešení, které se dá použít i v případě, že graf planární není. Nicméně tyto případy se často nedají dostatečně paralelizovat.

K vyřešení tohoto problému je použitý následující heuristický postup

1. Nelezení uzlu s nejmenším stupněm
2. Výpočet nejkratší cesty z uzlu do všech ostatních uzlů
3. Přiřazení uzlů do podgrafů podle hladin vzdáleností



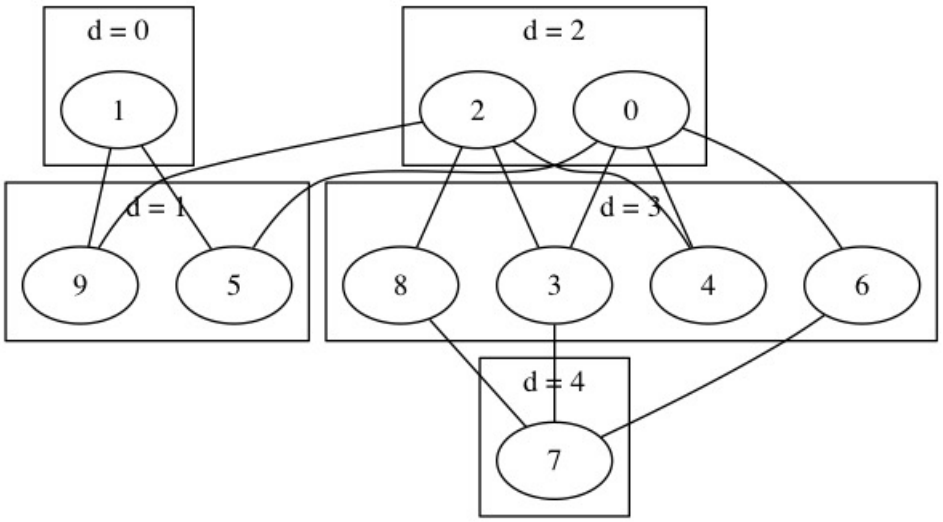
Obrázek 3.2: Graf vytvořený z matice

3.3.1 Nelezení uzlu s nejmenším stupněm

Stupeň uzlu je dán počtem hran, které do něj vedou. Stačí projít všechny uzly, a najít ten, do kterého vede nejméně hran. Význam tohoto kroku si lze představit jako hledání okrajového uzlu grafu. Například na obrázku 3.2 je více uzlů s nejmenším stupněm $\{1, 4, 5, 6, 8, 9\}$.

3.3.2 Výpočet nejkratší cesty z uzlu do všech ostatních uzlů

Jako startovní uzel je zvolen jeden z uzlů z minulého kroku. Pak je aplikován algoritmus na prohledávání do hloubky a všem uzlům je přiřazena vzdálenot. Uzly se stejnou vzdáleností tvoří potenciální adepty na množinu separátorů. Každá množina uzlů se stejnou vzdaleností dokáže rozdělit graf na tři monžiny uzlů A, S, B. Do množiny A patří uzly s menší vzdáleností než má množina separátorů, do množiny S patří samotná množina separátorů a do množiny B patří zbylé uzly. Je tak zajištěno, že mezi množinami A a B nevede žádná hrana. Tento krok je na obrázku 3.3.



Obrázek 3.3: Vzdálenost uzlů (hladiny)

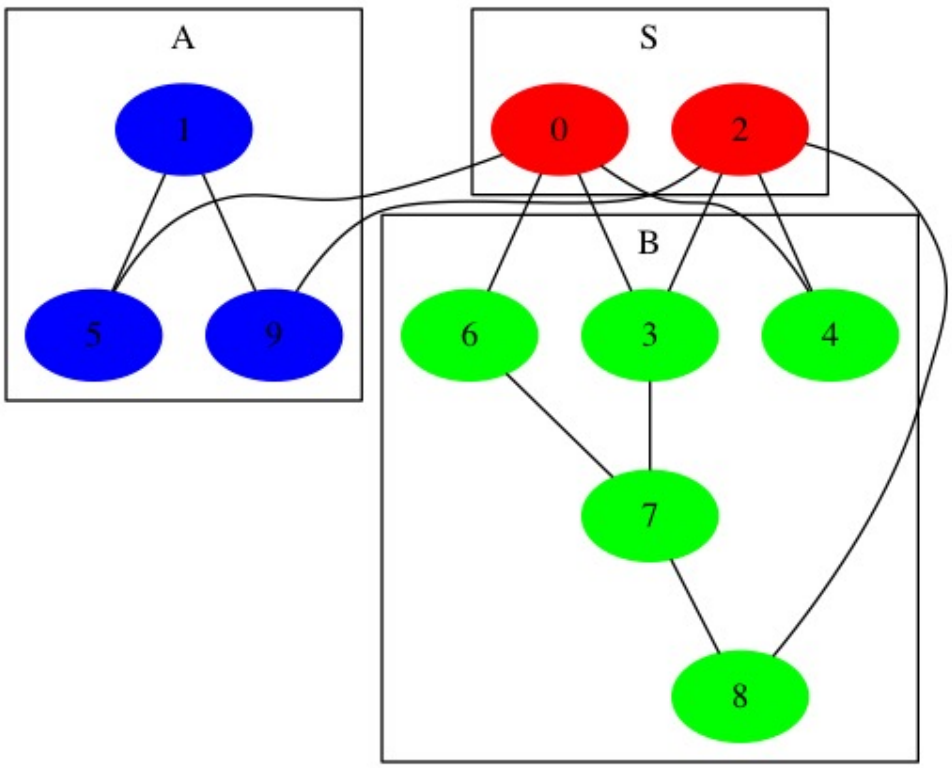
3.3.3 Přiřazení uzlů do podgrafů podle vzdáleností

Uzly je nutné přiřadit do množin A, S, B a vytvořit podgrafy. Jak bylo zmíněno dříve, množina separátorů musí být nejmenší, ale zároveň je výhodné aby množiny A a B byly podobně velké.

Zvolený algoritmus přidáva uzly vždy do menší množiny z množin A a B. Poslední množina uzlů je přiřazena do množiny separátorů. U zvoleného příkladu by postup byl následující:

1. $A = \{\}, B = \{\}, S = \{\}$
2. $A = \{1\}, B = \{\}, S = \{\}$
3. $A = \{1\}, B = \{7\}, S = \{\}$
4. $A = \{1, 5, 9\}, B = \{7\}, S = \{\}$
5. $A = \{1, 5, 9\}, B = \{7, 8, 3, 4, 6\}, S = \{\}$
6. $A = \{1, 5, 9\}, B = \{7, 8, 3, 4, 6\}, S = \{2, 0\}$

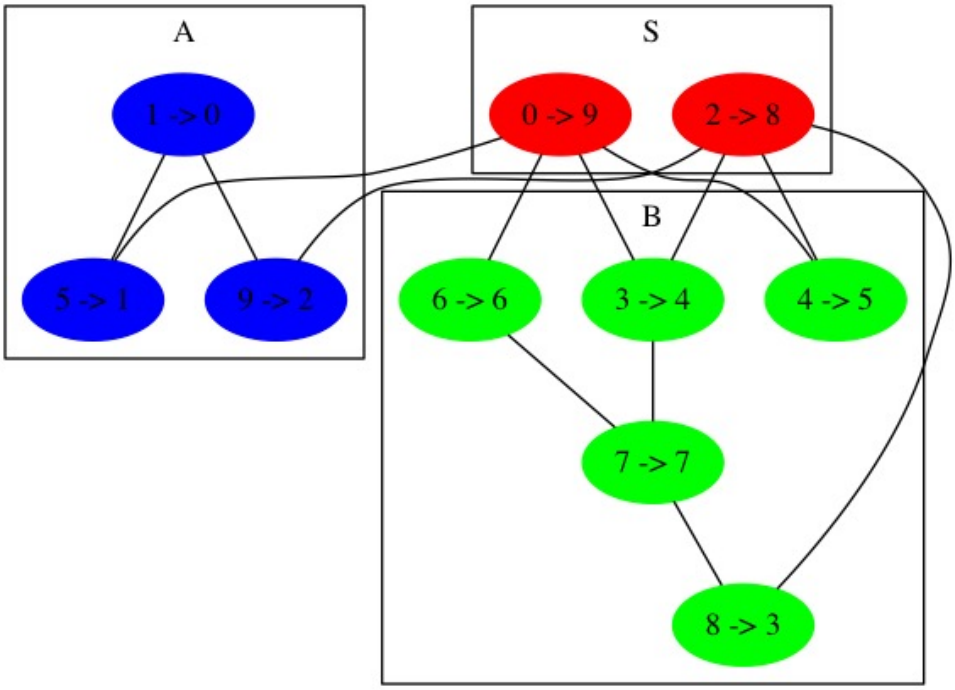
Odpovídající graf znázorňuje obrázek 3.4.



Obrázek 3.4: Přiřazené uzly do množin A, S, B

3.4 Změna indexů

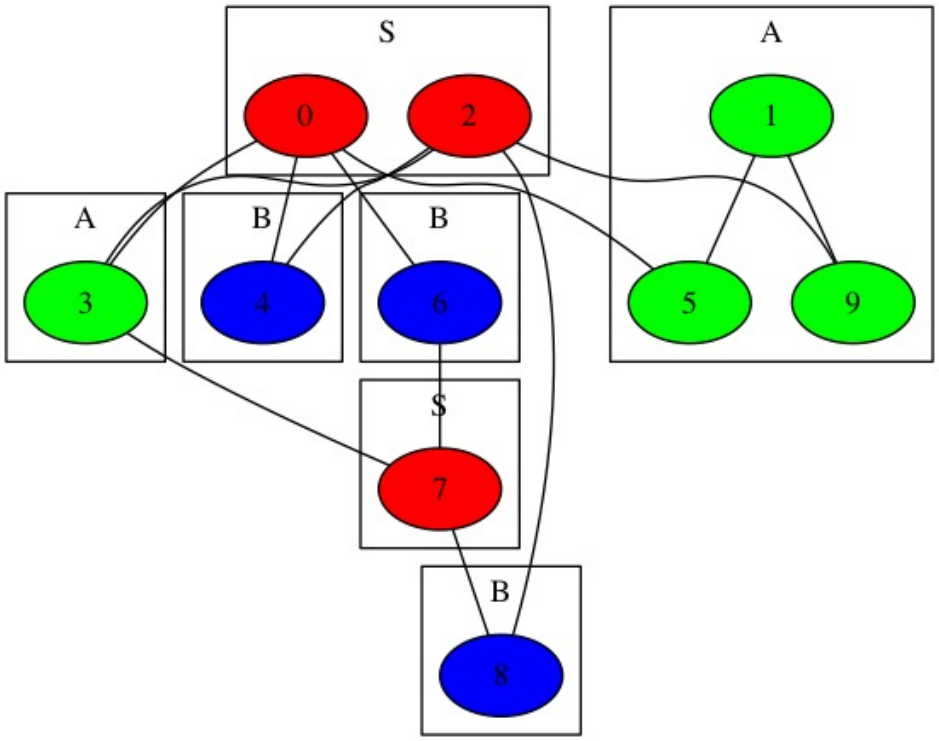
Změna indexu uzlu reprezentuje prohození řádku a sloupce v matici. Uzlům v množině separátoru jsou přiřazeny nejvyšší indexy. Řádky a sloupce matice, které jsou reprezentovány množinou separátorů, se tak dostanou dolů doprava. Řádky a sloupce reprezentované množinou uzlů A (nebo B) musí být vedle sebe. Indexy jsou přiřazeny vzestupně nejdříve množině A a pak množině B. Změna indexů je na obrázku 3.5.



Obrázek 3.5: Změna indexů uzlů

3.5 Rekurzivní zpracování

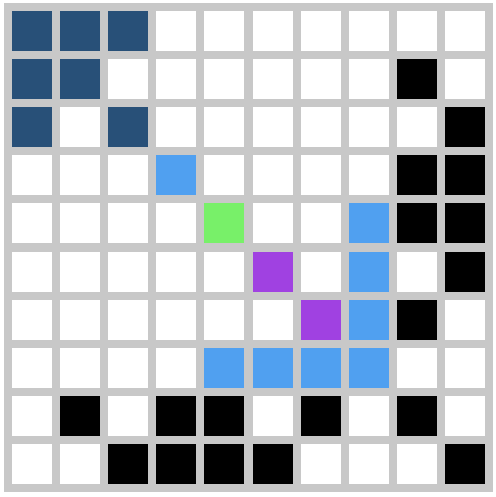
Množinu separátorů není možné dále zpracovat. Reprezentujve většinou řádky a sloupce matice, které mají mnoho nenulových čísel. Množiny A a B jsou rekurzivně rozdělovány, každá na 3 menši podmnožiny. Rekurzivní zpracování je ukončeno ve chvíli, kdy jsou množiny A a B příliš malé na další dělení (obsahují méně než 3 prvky). Do vznikajících množin je uložena hloubka rekurze, reprezentující hloubku ve vznikajícím grafu. Toto číslo je podstatné při paralelním zpracování matice. Výsledný graf je vidět na obrázku 3.6.



Obrázek 3.6: Výsledný graf po rekurzivním zpracování

3.6 Vyřešení soustavy rovnic na základě grafu

Řádky a sloupce matice jsou prohozeny na základě nových indexů uzlů v grafu, jak je vidět na obrázku 3.7. Matice je rozdělena na části odpovídající hloubce grafu. Části matice se stejnou hloubkou mohou být zpracovány paralelně. Tyto části mají většinou mnoho nenulových čisel, proto je výhodnější je dořešit gaussovou eliminační metodou. Iterační metody také lze použít, ale je možné, že by se zvyšovala nepřesnost, přičemž rychlost výpočtu by zůstala stejná.



Obrázek 3.7: Matice po prohození řádků a sloupců

Kapitola 4

Implementované řešení

4.1 Řešení na původní matici

Matice je implementována pomocí dvourozměrného pole. Řešení na původní matici používá následující metody:

1. Gauss-Seidel
2. Jacobi
3. Conjugate Gradient
4. Gassova eliminace

4.2 Řešení pomocí hranové řezu

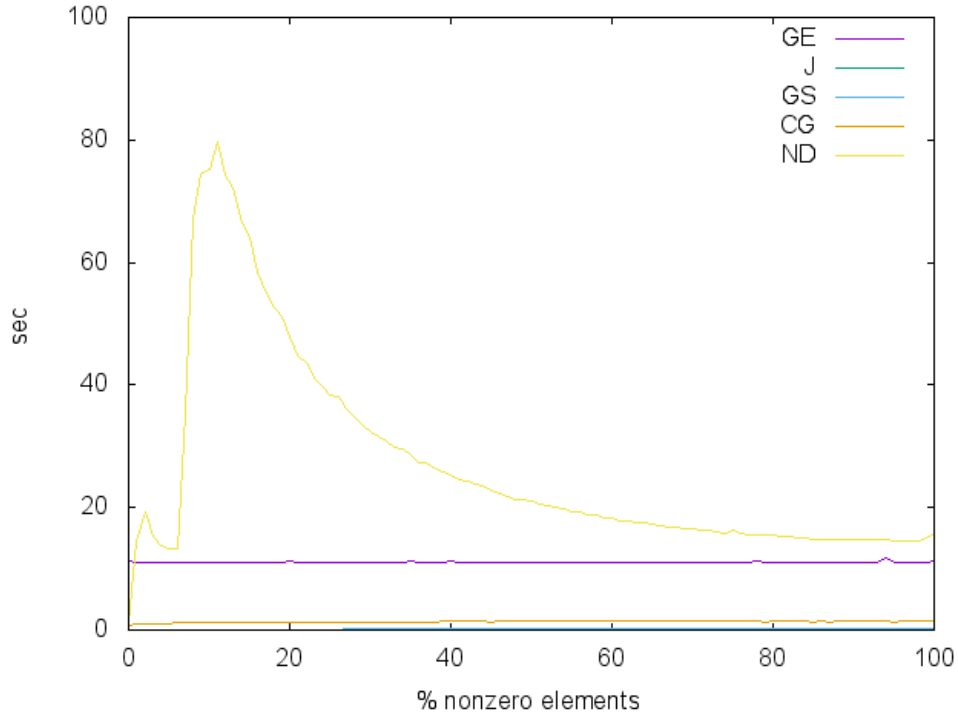
Matice je implementována pomocí 2D pole. Po použití metody hranového řezu je matice zpracována paralelně pomocí knihovy OpenMP. Ke zpracování jednotlivých částí matice je použita gaussova eliminační metoda.

4.3 Testovací sestava

1. Procesor: Intel(R) Core(TM) i5-4250U CPU @ 1.30GHz
2. RAM: 8 GB 1600 MHz DDR3
3. Operační system: OS X Yosemite, verze 10.10.5

4.4 Naměřené výsledky

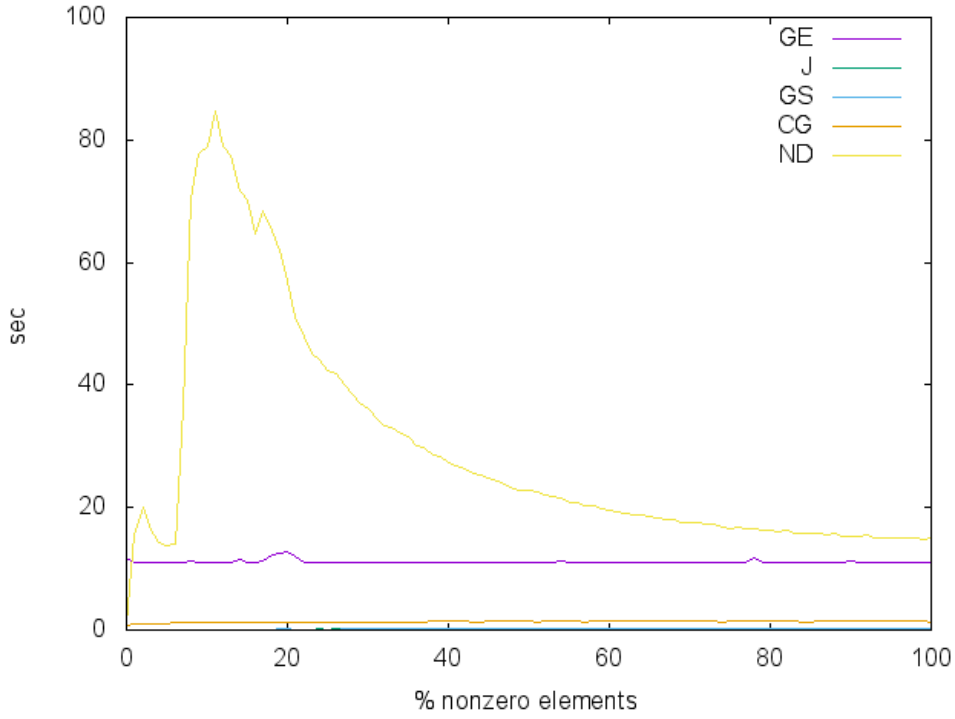
Všechna měření byla prováděna na matici o velikosti 2000 x 2000 prvků a všechny metody poskytly správné výsledky. Na obrázku 4.1 je vidět, že metoda hranového řezu s maticí plně obsazenou nenulovými čísly, je nejpomalejší.



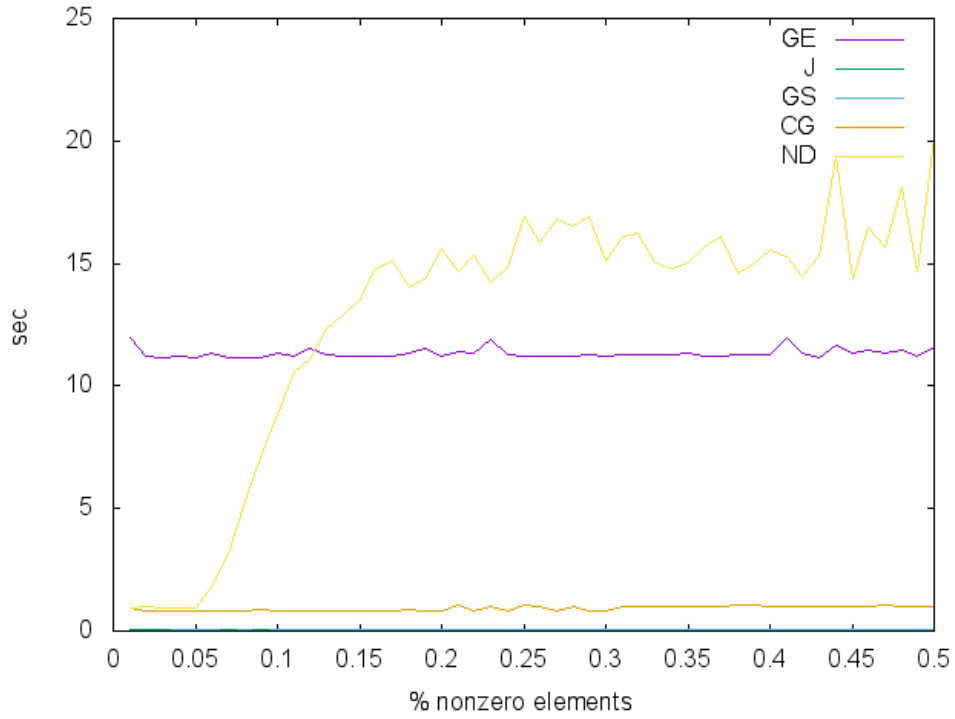
Obrázek 4.1: Závislost času na počtu nenulových prvků. (1 vlákno)

U plně obsazené matice je rozdíl mezi GE a ND způsoben hledáním hranového řezu. Nicméně u řidších maticí není rozdíl způsoben hledáním řezu, ale nevhodnou implementací. Gaussova eliminace, která je použita na jednotlivé části po rozdělení matice, nezohledňuje nulové prvky v matici. Tím se také ztrácí výhoda vícevláknového zpracování, jak je vidět na obrázku 4.2.

Obrázek 4.3 odpovídá řešením na řídkých maticích. U velice nízkého počtu nenulových prvků je vidět, že se metoda velice přiblížila k iteračním metodám. Lepší výsledek je způsoben především tím, že graf reprezentující matici není souvislý. Komponenty grafu se řeší samostatně, takže Gaussova eliminace nezohledňuje již nulové části matice.



Obrázek 4.2: Závislost času na počtu nenulových prvků. (2 vlákna)



Obrázek 4.3: Závislost času na počtu nenulových prvků.

Závěr

Práce detailněji popsala metodu hranového řezu. Ukázalo se, že dílčí části matice po rozdělení na samostatně řešitelné podmatice již nejsou řídké, tudíž nebylo nutné využít iterační metody. U metody hranového řezu byla využita Gaussova eliminační metoda. Iteřační metody byly implementovány a slouží jen jako porovnání s metodou hranového řezu.

Metoda hranového řezu byla úspěšně implementována, nicméně zpětná transformace do původní matice nevyužila celý potenciál této metody. Kvůli tomuto nedostatku bylo dosaženo nepříznivých výsledků při měření.

V budoucnu je možné implementovat Gaussovu eliminační metodu, která dokáže zohlednit nulové prvky matice na základě grafu hranového řezu. Zohlednění nulových částí matice z grafu přináší další možnosti paralelního zpracování a tím i zrychlení celé implementace.

Literatura

[1] Navara, M.: Numerické řešení soustav lineárních rovnic[online]. 2016. Dostupné z: http://cmp.felk.cvut.cz/~navara/nm/linear_print.pdf

[2] Conjugate Gradient[online]. Dostupné z: https://en.wikipedia.org/wiki/Conjugate_gradient_method

[3] Nested Dissection[online]. Dostupné z: https://en.wikipedia.org/wiki/Nested_dissection

[4] Khaira, M. S.: Nested Dissection: A survey and comparison of various nested dissection algorithms[online]. 1992. Dostupné z: https://www.cs.cmu.edu/~glmiller/Publications/CMU-CS-92-106R.pdf

[5] George, A.: Nested Dissection of a Regular Finite Element Mesh. *SIAM Journal on Numerical Analysis*, ročník 10, č. 2, 4 1973: s. 345–363. Dostupné z: http://web.math.ucsb.edu/~hdc/teaching/Math206D/Nested_Dissection.pdf

Příloha **A**

Seznam použitých zkratek

GE Gaussova eliminační metoda

GS Gauss-seidel

J Jacobiho metoda

CG Conjugate Gradient (metoda sdružených gradientů)

ND Nested dissection (metoda hranového řezu)

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `Bakalarka.run` ..... spustitelný soubor, řešič soustav
- `matrix` ..... ukázková vstupní data, matice A
- `matrixr` ..... ukázková vstupní data, matice B
- `scriptGraph.sh` ..... script na vytvoření grafů
- `MatrixGenerator.run` ..... spustitelný soubor, generator matic
- `Implementace` ..... Zdrojové kódy
- `LaTex` ..... Zdrojová forma práce ve formatu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF