Sem vložte zadání Vaší práce.

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

Bakalářská práce

Efektivní LU rozklad pro řídké matice

Stanislav Kusý

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

11. května 2015

Poděkování

Rád bych poděkoval panu Ing. Ivanu Šimečkovi, Ph.D. za odbornou pomoc při psaní této práce.

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 11. května 2015 .....

Odkaz na tuto práci

Kusý, Stanislav. *Efektivní LU rozklad pro řídké matice*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce řeší LU rozklad řídkých matic a možnost jeho paralelizace. K rozkladu je využita Croutova, Choleského a QR metoda. Tyto metody implementuje pro několik formátů ukládání řídkých matic.

Implementované metody jsou testovány a porovnávány z hlediska časové a paměťové náročnosti.

Klíčová slova Paralelní LU rozklad, QR rozklad, řídké matice, Croutova metoda, Choleského metoda.

Abstract

This paper describes parallel LU decomposition algorithms of sparse matrices. It uses Crout's, Cholesky and QR method. These methods are implemented for serveral types of sparse matrices.

Implemented methods are tested for their time and memory performance.

Keywords Parallel LU decomposition, QR decomposition, sparse matrices, Crout's method, Cholesky method.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tato práce popisuje LU rozklad matic. LU rozklad je metoda, která se využívá pro řešení soustav lineárních rovnic. Práce upravuje LU rozklad pro řídké matice.

Řídké matice jsou takové matice, které mají většinu prvků nulových. Často jsou velmi rozsáhlé a náročné na paměť, proto se pro jejich ukládání do paměti a na disk používají různé řídké formáty. Práce popisuje dva vybrané řídké formáty uložení matic.

Cílem této práce je implementace vybraných metod pro LU rozklad řídkých matic. Práce se zabývá Croutovou, Choleského a QR metodou. Tyto metody je potřeba nejprve implementovat pro husté matice a poté upravit pro řídké matice. Dále pro tyto metody diskutuje a implementuje možnosti paralelizace nad sdílenou pamětí.

Výsledkem jsou implementované metody, které jsou testovány nad vhodnými daty a porovnávány z hlediska časové a paměťové náročnosti. Porovnávány jsou také různé formáty uložení řídkých matic pro jednotlivé metody a zrychlení metod pro různé počty vláken.

Práce obsahuje dvě hlavní části. V první části jsou popsány teoretické základy metod provádějící LU rozklad a formáty uložení matic v paměti. V druhé části jsou výsledky měření a porovnání metod.

Kapitola 1

Popis problému a cíl práce

1.1 Popis problému

Jedním z možných řešení soustavy lineárních rovnic je LU rozklad. Ten lze provést několika metodami. Vybrané z nich tato práce popisuje. Problém nastává u řídkých matic, které mohou mít veliké rozměry a nemusí se vejít do paměti, přitom většinu jejich prvků tvoří nuly. Proto je dobré ukládat řídké matice v jiném formátu než klasické, husté matice.

1.2 Cíl práce

Cílem práce je implementace vybraných metod pro LU rozklad. Konkrétně to jsou Croutova, Choleského a QR metoda. Implementované metody následně upravit pro dva formáty uložení řídkých matic a navrhnout jejich paralelizaci nad sdílenou pamětí pomocí OpenMP API.

Implementované metody je potřeba otestovat z hlediska časové a paměťové náročnosti. Naměřené hodnoty je dále potřeba vyhodnotit a porovnat metody podle formátů uložení matic, počtu vláken, porovnat metody mezi sebou a srovnat s teoretickým předpokladem.

Kapitola 2

Teoretický základ

V této kapitole jsou popsány teoretické základy jednotlivých metod, možnosti jejich paralelizace a různé formáty uložení matic v paměti.

2.1 LU rozklad

LU rozklad je operace, při které se matice **A** rozloží na součin dvou matic.

$$A = LU$$

Kde matice **L** je dolní trojúhelníková matice a matice **U** je horní trojúhelníková matice.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

2.1.1 Řešení soustavy lineárních rovnic pomocí LU rozkladu

Tento rozklad se využívá při řešení soustavy lineárních rovnic.

$$Ax = b$$

Protože se rozkládá pouze matice **A**, využívá se tato metoda především u soustav, kde se mění pouze výsledky pravé strany **b** a matice **A** zůstává stejná.

Dosazením LU rozkladu za matici **A** dostáváme rovnici

$$LUx = b.$$

Zavedeme substituci

$$y = Ux.$$

Tím dostaneme soustavu dvou rovnic

$$Ly = b$$

$$Ux = y.$$

Protože matice **L** a **U** jsou trojúhelníkové, je jejich vyřešení lehké.

2.1.2 Croutova metoda

2.1.2.1 Popis metody

Při použití Croutovy metody má matice **L** na diagonále samé jedničky.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Pro výpočet matice **L** po sloupcích a matice **U** po řádcích z matice **A** o velikosti $n \times n$ platí následující pravidla, pro $i = 1, 2, .., n$:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1}(l_{ik}u_{kj}) \qquad j = i, .., n$$

$$l_{ji} = \frac{1}{u_{ii}}(a_{ji} - \sum_{k=1}^{j-1}(l_{jk}u_{ki})) \qquad j = i+1, .., n$$

2.1.2.2 Časová složitost

Pro matici o velikosti $n \times n$ Croutův algoritmus vyžaduje $2n^3/3$ operací.[5]

2.1.2.3 Datová závislost během LU rozkladu

Pro výpočet prvku na pozici u_{ij} nebo l_{ij} je z matice **A** potřeba znát pouze prvek a_{ij} a protože na diagonále matice **L** jsou pouze jedničky je možné prvky matic **L** a **U** zapisovat přímo do matice **A** (tzv. *in-place* implementace). Dále je pro výpočet prvku l_{ij} potřeba znát ještě všechny prvky matice **L**, které se nachází v řádce před prvkem l_{ij} a všechny prvky matice **U**, které se nachází ve sloupci nad prvkem u_{jj} i samotný prvek u_{jj}.

Znázornění datové závislosti pro prvek l_{43} (prvky, které jsou potřeba k jeho výpočtu jsou tučně zvýrazněny):

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & \mathbf{a_{43}} & a_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ \mathbf{l_{41}} & \mathbf{l_{42}} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \mathbf{u_{13}} & u_{14} \\ 0 & u_{22} & \mathbf{u_{23}} & u_{24} \\ 0 & 0 & \mathbf{u_{33}} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

Pro výpočet prvku u_{ij} je potřeba znát kromě prvku a_{ij} ještě všechny prvky matice **U**, které se nachází ve sloupci nad prvkem u_{ij} a všechny prvky matice **L**, které se nachází v řádce před prvkem l_{ii}.

Znázornění datové závislosti pro prvek u_{33} (prvky, které jsou potřeba k jeho výpočtu jsou tučně zvýrazněny):

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & \mathbf{a_{33}} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ \mathbf{l_{31}} & \mathbf{l_{32}} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \mathbf{u_{13}} & u_{14} \\ 0 & u_{22} & \mathbf{u_{23}} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

2.1.2.4 Paralelizace

Z datové závislosti vyplývá, že se paralelně může v k-tém kroku nejprve vypočítat k-tý řádek matice **U**. Tím se vypočítá i prvek u_{ii}, který je potřeba pro paralelní výpočet k-tého sloupce matice **L**.

2.1.3 Choleského metoda

2.1.3.1 Popis metody

Choleského metoda se dá použít pouze pro symetrické matice. Z toho vyplývá, že pro tento LU rozklad platí

$$A = LL^T$$

$$A = U^T U.$$

Pro výpočet prvků matice **U** po řádcích z matice **A** o velikosti $n \times n$ platí následují pravidla, pro i=1, 2,..,n:

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2}$$

$$u_{ij} = \frac{1}{u_{ii}}(a_{ij} - \sum_{k=1}^{i-1}(u_{ki}u_{kj})) \qquad j = i+1, .., n$$

Matice **L** je rovna transponové matici **U**.

2.1.3.2 Časová složitost

Pro matici o velikosti $n \times n$ Choleského algoritmus vyžaduje $n^3/3$ operací.[2]

2.1.3.3 Datová závislost během LU rozkladu

Stejně jako u Croutovy metody, tato metoda potřebuje pro výpočet prvku na pozici u_{ij} znát z matice **A** pouze prvek a_{ij} a protože prvky na diagonálách obou matic **L** a **U** se rovnají, je zde možnost *in-place* implementace. Dále pro výpočet prvku u_{ij} je potřeba znát všechny prvky matice **U**, které se nachází ve sloupci nad prvkem u_{ij} a všechny prvky, které se nachází ve sloupci nad prvkem u_{ii}, včetně prvku u_{ii}.

Znázornění datové závislosti pro prvek u_{34} (prvky, které jsou potřeba k jeho výpočtu jsou tučně zvýrazněny):

$$A = U^T U.$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & \mathbf{a_{34}} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} u_{11} & 0 & 0 & 0 \\ u_{12} & u_{22} & 0 & 0 \\ \mathbf{u_{13}} & \mathbf{u_{23}} & \mathbf{u_{33}} & 0 \\ \mathbf{u_{14}} & \mathbf{u_{24}} & u_{34} & u_{44} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \mathbf{u_{13}} & \mathbf{u_{14}} \\ 0 & u_{22} & \mathbf{u_{23}} & \mathbf{u_{24}} \\ 0 & 0 & \mathbf{u_{33}} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

2.1.3.4 Paralelizace

Z datové závislosti vyplývá, že v k-tém kroku se může paralelně vypočítat k-tý řádek matice **U**.

2.2 QR rozklad

Mezi další rozklady matic patří QR rozklad, který rozloží matici **A** na matice **Q** a **R**.

$$A = QR$$

Kde matice **R** je horní trojúhelníková matice a matice **Q** má vzájemně ortogonální sloupce, tedy pro ní platí

$$Q^T Q = E.$$

Kde $\mathbf{Q^T}$ je transponovaná matice **Q** a **E** je jednotková matice.[6]

Tato metoda se také používají při řešení soustav lineárních rovnic.

$$Ax = b.$$

Dosazením předchozích vztahů vznikne rovnice.

$$Rx = Q^T b.$$

2.2.1 Gramův-Schmidtův proces

Pro nalezení QR rozkladu existuje několik postupů. Jedním z nich je Gramův-Schmidtův proces. Gramův-Schmidtův proces nalezne pro danou konečnou množinu vektorů jejich ortogonální bázi.

QR rozklad pomocí Gramova-Schmidtova procesu se vypočítá pro matice o velikosti $n \times n$ podle následujících pravidel, pro i=1, 2,..n:

$$r_{ji} = Q_j * A_i \qquad j = 1, .., i-1$$

$$v = A_i - \sum_{j=1}^{i-1} (r_{j,i} * Q_j)$$

$$r_{ii} = ||v||$$

$$Q_i = \frac{v}{||v||}$$

Kde Q_j znamená j-tý sloupec matice **Q** a $Q_j * A_i$ znamená skalární součin j-tého sloupce matice **Q** a i-tého sloupce matice **A** a ||v|| znamená normalizace vektoru.

2.2.1.1 Pseudo kód

for $i = 1 : n$ **do**
  $v = A(:, i)$
  for $j = 1 : i - 1$ **do**
    $R(j, i) = Q(:, j) * A(:, i)$
  end for
  for $j = 1 : i - 1$ **do**
    $v = v - R(j, i) * Q(:, j)$
  end for
  R(i,i)=||v||
  Q(:,i)=v/R(i,i)
end for

Kde Q(:,i) znamená i-tý sloupec matice **Q**, $Q(:, j) * A(:, i)$ znamená skalární součin vektorů a ||v|| znamená normalizace vektoru.

2.2.1.2 Časová složitost

Pro matici o velikosti $n \times n$ Gramův-Schmidtův proces vyžaduje $2n^3$ operací.[3]

2.2.1.3 Datová závislost během QR rozkladu

Pro výpočet k-tého sloupce matice **R** je potřeba znát k-tý sloupec matice **A** a sloupce 1 až $k - 1$ matice **Q**. Pro prvky na diagonále (r_{kk}) je navíc potřeba znát i všechny prvky, které se nachází ve sloupci nad ním. Pro výpočet k-tého

sloupce matice **Q** je potřeba znát k-tý sloupec matice **A** i matice **R** a sloupce 1 až $k-1$ matice **Q**.

Znázornění datové závislosti pro třetí sloupec matice **R** (kromě prvku r_{33}) (prvky, které jsou potřeba k jeho výpočtu jsou tučně zvýrazněny):

$$\begin{pmatrix} a_{11} & a_{12} & \mathbf{a_{13}} & a_{14} \\ a_{21} & a_{22} & \mathbf{a_{23}} & a_{24} \\ a_{31} & a_{32} & \mathbf{a_{33}} & a_{34} \\ a_{41} & a_{42} & \mathbf{a_{43}} & a_{44} \end{pmatrix} = \begin{pmatrix} \mathbf{q_{11}} & \mathbf{q_{12}} & q_{13} & q_{14} \\ \mathbf{q_{21}} & \mathbf{q_{22}} & q_{23} & q_{24} \\ \mathbf{q_{31}} & \mathbf{q_{32}} & q_{33} & q_{34} \\ \mathbf{q_{41}} & \mathbf{q_{42}} & q_{43} & q_{44} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{pmatrix}$$

Znázornění datové závislosti pro třetí sloupec matice **Q** (prvky, které jsou potřeba k jeho výpočtu jsou tučně zvýrazněny):

$$\begin{pmatrix} a_{11} & a_{12} & \mathbf{a_{13}} & a_{14} \\ a_{21} & a_{22} & \mathbf{a_{23}} & a_{24} \\ a_{31} & a_{32} & \mathbf{a_{33}} & a_{34} \\ a_{41} & a_{42} & \mathbf{a_{43}} & a_{44} \end{pmatrix} = \begin{pmatrix} \mathbf{q_{11}} & \mathbf{q_{12}} & q_{13} & q_{14} \\ \mathbf{q_{21}} & \mathbf{q_{22}} & q_{23} & q_{24} \\ \mathbf{q_{31}} & \mathbf{q_{32}} & q_{33} & q_{34} \\ \mathbf{q_{41}} & \mathbf{q_{42}} & q_{43} & q_{44} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \mathbf{r_{13}} & r_{14} \\ 0 & r_{22} & \mathbf{r_{23}} & r_{24} \\ 0 & 0 & \mathbf{r_{33}} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{pmatrix}$$

2.2.1.4 Paralelizace

Gramův-Schmidtův proces se dá počítat pouze po sloupcích. Jednotlivé prvky ve sloupci na sobě nejsou závislé ale sdílejí spolu vektor **v** (viz pseudo kód 2.2.1.1). Z tohoto důvodu se paralelně počítají pouze prvky ve sloupci matice **Q**.

2.3 Formáty pro ukládání řídkých matice

Řídké matice mají většinu prvků nulových. Takové matice mívají často velké rozměry a pro snížení jejich paměťové náročnosti se pro uložení do paměti využívají různé formáty.

2.3.1 Hustý formát

Hustý formát ukládá do paměti všechny prvky matice. Jeho nevýhodou je, že pokud má matice velké množství nulových prvků, potřebuje více paměti než jiné formáty. Výhodou je, že přístup k jednotlivým prvkům má konstantní složitost.

$$\begin{pmatrix} 2 & 0 & 6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix}$$

hodnoty	2	0	6	0	0	1	0	0	0	5	3	0	0	0	2	4

2.3.2 COO formát

COO (Coordinate) formát ukládá pro každý nenulový prvek jeho hodnotu, index řádku a index sloupce. Pro uložení matice jsou potřeba 3 pole, které mají stejnou velikost shodnou s počtem nenulových prvků v matici.

Pořadí, ve kterém se prvky ukládají, je libovolný. Nejčastěji se však ukládají po řádcích nebo po sloupcích (použito v příkladu uložení matice).

Uložení řídkých matic v COO formátu je pro řídké matice oproti hustému formátu méně náročné na paměť, ale nalezení konkrétního prvku nemá konstantní složitost.

$$\begin{pmatrix} 2 & 0 & 6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix}$$

hodnoty	2	1	5	6	3	2	4
indexy řádků	1	2	3	1	3	4	4
indexy sloupců	1	2	2	3	3	3	4

2.3.3 CRS formát

CRS (Compressed Row Storage) formát ukládá nenulové prvky matice po řádcích, proto se hodí pro algoritmy, které pracují s maticí po řádcích. K uložení matice do paměti potřebuje 3 pole. V prvním poli se ukládají hodnoty nenulových prvků, v druhém jsou uloženy jejich sloupcové indexy a ve třetím poli jsou ukazatelé na první nenulový prvek v daném řádku.

Z uvedených formátů je tento nejméně náročný na paměť. Nevýhodou je, že nalezení daného prvku má lineární složitost. Pokud však program prochází matici po řádcích, může mít nalezení jednoho prvku konstantní složitost.

$$\begin{pmatrix} 2 & 0 & 6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix}$$

hodnoty	2	6	1	5	3	2	4
indexy sloupců	1	3	2	2	3	3	4

ukazatelé na nový řádek	1	3	4	6

2.3.4 CCS formát

CCS (Compressed Column Storage) formát je podobný CRS formátu. Na rozdíl od něho se nenulové prvky ukládají po sloupcích, proto je vhodný pro algoritmy, které pracují s maticí po sloupcích. K uložení matice do paměti potřebuje 3 pole. V prvním, stejně jako u CRS formátu, jsou uloženy hodnoty

nenulových prvků. V druhém poli jsou indexy řádků a ve třetím poli jsou ukazatelé na první nenulový prvek v daném sloupci.

$$\begin{pmatrix} 2 & 0 & 6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix}$$

hodnoty	2	1	5	6	3	2	4
indexy řádků	1	2	3	1	3	4	4

ukazatelé na nový sloupec	1	2	4	7

2.4 Redukce zaplnění pro Croutovu a Choleského metodu

Při LU rozkladu dochází u výsledných matic k jejich zaplnění nenulovými prvky (tzv. *fill-in*). Zaplnění lze snížit prohozením řádků a sloupců v matici.

2.4.1 Postup

Nechť matice **A** je řídká matice, kde pro k-tý krok se na řádcích a na sloupcích $1, 2, .., k-1$ nachází prvky po provedení LU rozkladu a na zbylých místech se nachází prvky před provedením LU rozkladu.

$$A = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ l_{3,1} & l_{3,2} & a_{k,k} & a_{k,k+1} \\ l_{4,1} & l_{4,2} & a_{k+1,k} & a_{k+1,k+1} \end{pmatrix}$$

Pro k-tý krok LU rozkladu je potřeba najít prvek, který se přesune na pozici prvku a_{kk} a tím se minimalizuje zaplnění.

Nejprve se vytvoří matice **B** tak, že se nenulové prvky v matici **A** nahradí jedničkami a nulové prvky nulami. Matice **S**, **T** a **N** jsou submatice matice **B**, pro které platí:

$$S = \{b_{ij}\} \qquad i \geq k, j < k$$
$$T = \{b_{ij}\} \qquad i < k, j \geq k$$
$$N = \{b_{ij}\} \qquad i \geq k, j \geq k$$

Matice **D** je definována jako

$$D = S * T.$$

Kde $*$ je Booleovské násobení matic. Matice $\bar{\mathbf{D}}$ pak vznikne z matice **D** nahrazením nenulových prvků nulou a nulových prvků jedničkou.

Matice **W** je definována jako

$$W = \bar{D} + N.$$

Kde + je Booleovské sčítání matic.

Nechť **V** je sloupcový vektor vektor délky $n-k+1$ obsahující samé jedničky. Pak **r** a **c** jsou celočíselná pole o délce $n+k-1$ a jsou definována jako

$$c = V'W$$

$$r = WV.$$

Nechť $r_i + c_j = max(r_p + c_q)$. Pak přesun prvku $a_{i+k-1,j+k-1}$ na pozici prvku a_{kk} na začátku k-tého kroku Croutovy metody sníží zaplnění výsledných matic.

Pro Choleského metodu pak musí platit $r_i + c_i = max(r_p + c_p)$, aby zůstala zachována symetričnost matice.

2.4.2 Důkaz

Důkaz je uveden v [4].

Kapitola 3

Realizace

V této kapitole je popsána implementace metod. Zaměřuje se především na matice, jejich formáty uložení a práci s nimi.

3.1 Programovací jazyk

Pro implementaci byl zvolen programovací jazyk C++ z důvodů jeho výkonnosti. Pro paralelizaci programu je použito OpenMP API. Jeho výhodou je platformní přenositelnost a jednodušší implementace než POSIXových vláken.

3.2 Implementace matic

Hustý formát (viz kap. 2.3.1) je implementován pomocí dvourozměrného pole. K jednotlivým prvkům přistupuje s konstantní složitostí.

CRS/CCS formát (viz kap. 2.3.3/2.3.4) je implementován pomocí pole a spojového seznamu. Pole má velikost shodnout s velikostí matice a nachází se v něm začátky spojových seznamů. Umístění v poli pak značí v jaké řádce (CRS formát) nebo sloupci (CCS formát) se prvek v matici nachází. V jednom prvku spojového seznamu je uložena hodnota prvku v matici a index sloupce (pro CRS formát) nebo řádku (pro CCS formát) ve kterém se prvek nachází. Prvky ve spojovém seznamu jsou řazeny podle indexu sloupce/řádky. Zapisování prvků do matic tak má lineární složitost. Protože metody přistupují k prvkům po sloupcích nebo po řádcích je jejich nalezení implementováno s konstantní složitostí.

COO formát (viz kap. 2.3.2) je implementován pomocí pole struktur. Pole má velikost shodnou s počtem nenulových prvků v matici. Struktura (jeden prvek v poli) se skládá z hodnoty prvku matice, indexu řádku a indexu sloupce. Přístup k prvku má logaritmickou složitost.

3.3 Výběr mezi CRS a CCS formátem uložení matic

Mezi CRS (viz kap. 2.3.3) a CCS (viz kap. 2.3.4) formátem se rozhoduje na základě toho, jakým způsobem přistupují jednotlivé metody k prvkům v maticích. LU rozklad (viz kap. 2.1) přistupuje k prvkům matice **L** po řádcích, proto je pro ni vhodné při použití CRS/CCS formátu použít formát CRS, který má prvky uloženy po řádcích. K prvkům matice **U** přistupuje po sloupcích, proto je pro ni vhodné použít formát CCS, který má prvky uloženy po sloupcích.

QR rozklad (viz kap. 2.2) přistupuje ke všem maticím po sloupcích, proto je pro tyto matice při použití CRS/CCS formátu vhodné použít formát CCS.

3.4 Ukládání výsledných matic ve vnitřní paměti

Croutova metoda (viz kap. 2.1.2) pro hustý formát uložení matic používá *in-place* implementaci. Pro uložení vstupní a výstupní matice stačí pouze jedna matice, do které se během rozkladu zapisují výsledky. Pro CRS/CCS a COO formát ukládá výsledky do jiných matic než je původní, vstupní matice a proto pro tyto formáty potřebuje během výpočtu tři různé matice.

Choleského metoda (viz kap. 2.1.3), stejně jako Croutova metoda, pro hustý formát uložení matice používá *in-place* implementaci. Potřebuje tedy pouze jednu matici. Pro CRS/CCS a COO formát ukládá výsledek do jiných matic než je původní, vstupní matice. Jelikož jsou výsledné matice symetrické, stačí uložit pouze jednu matici. Pro tyto formáty potřebuje během výpočtu dvě různé matice.

QR rozklad (viz kap. 2.2) pro hustý formát uložení matic používá *in-place* implementaci pro matici **Q**. Matici **R** ukládá do jiné matice, protože matice **Q** není trojúhelníková matice. Pro CRS/CCS a COO formát ukládá matici **Q** a **R** do samostatných matic. Pro tyto formáty potřebuje během výpočtu tři různé matice.

Kapitola 4

Testování

V této kapitole jsou popsány měřící metodiky, výsledky měření a porovnání jednotlivých metod s různým počtem vláken, formátem uložení matic a porovnání metod mezi sebou.

4.1 Testovací soustava

Testování probíhalo na školním serveru star.fit.cvut.cz s následujícími parametry:

Procesor:	2ks Xeon 2620 @ 2.1Ghz
RAM:	32 GB

4.2 Testované matice

Program je testován na vhodných čtvercových maticích. Matice byly pro danou velikost náhodně vygenerovány. Pro testování redukce zaplnění byly na vygenerovaných maticích znát jen malé rozdíly, proto byly použity matice s rozdílnými vlastnostmi z veřejně dostupných databází. Konkrétně z databáze *The University of Florida Sparse Matrix Collection*.

Při testování byl měřen čas potřebný k rozkladu matice a počet nenulových prvků ve výsledných maticích.

4.3 Stanovení časové náročnosti

Časová náročnost byla měřena v sekundách pomocí OpenMP API. Čas je měřen od počátku výpočtu LU rozkladu až do jeho dokončení. Neobsahuje tedy čas potřebný pro načtení vstupní matice.

Časová složitost byla měřena pro různé formáty uložení matic a dále pro paralelní rozklad při použití CRS/CCS formátu (viz kap. 2.3.3/2.3.4) uložení

matic. Paralelní verze byly testovány pro 2, 4, 6 a 12 vláken a bylo vypočítáno jejich zrychlení oproti sekvenční verzi jako

$$S = \frac{T_{sek}}{T_{par}}.$$

Kde S je zrychlení, T_{par} je čas paralelní verze a T_{sek} je čas sekvenční verze.

Výsledný čas je průměrný čas ze čtyř měření.

4.4 Stanovení paměťové náročnosti

Paměťová náročnost je odvozena od velikosti matice, počtu nenulových prvků, způsobu uložení matic v paměti a počtu matic, které daná metoda potřebuje. Číselné hodnoty prvků matic jsou uloženy ve formátu *double*, který má velikost 8 B. Celočíselné hodnoty jsou uloženy ve formátu *integer*, který má velikost 4 B.

Hustý formát (viz kap. 2.3.1) potřebuje k uložení matice dvourozměrné pole prvků. Pro uložení matice o velikosti n potřebuje $n * n * 8$ B paměti. Na počtu nenulových prvků v matici nezáleží.

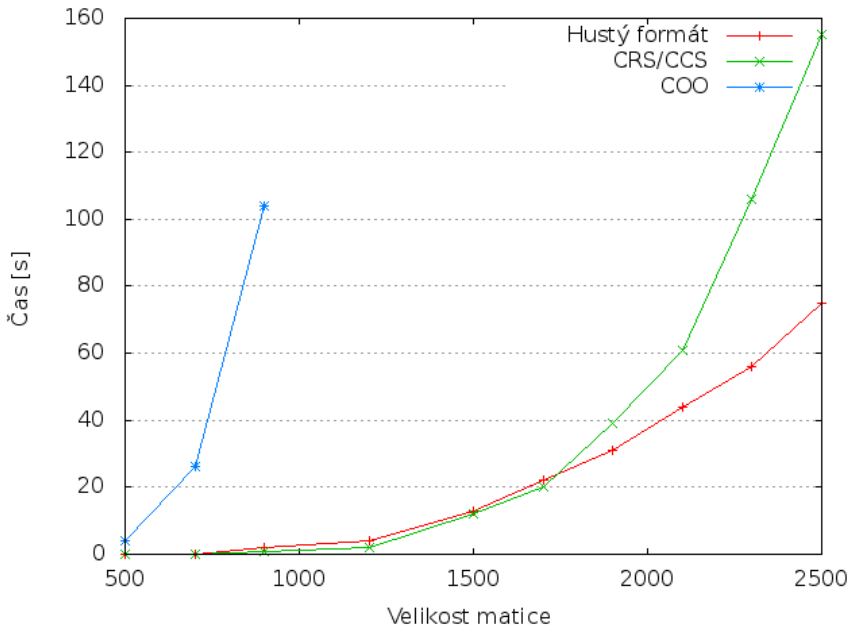
CSR/CCS formát (viz kap. 2.3.3/2.3.4) ukládá pouze nenulové prvky, jejich indexy řádky/sloupce a ukazatele na začátek daného sloupce/řádku. Pro uložení matice o velikosti n s m nenulovými prvky potřebuje $m*(4+8)+n*4$ B.

COO formát (viz kap. 2.3.2) ukládá nenulové prvky a jejich indexy řádků a sloupců. Pro uložení matice o m nenulových prvcích potřebuje $m*(8+4+4)$ B. Na velikosti matice nezáleží.

4.5 Časová náročnost

Tabulka 4.1: Časová náročnost Croutovy metody podle způsobu uložení matic

Velikost matice	Čas [s]		
	Hustý formát	CRS/CCS	COO
500	0,228	0,182	4,607
700	0,936	0,500	26,747
900	2,153	1,491	104,639
1200	4,924	2,906	
1500	13,149	12,08	
1700	22,019	20,579	
1900	31,487	39,788	
2100	44,504	61,427	
2300	56,681	106,321	
2500	75,972	155,902	



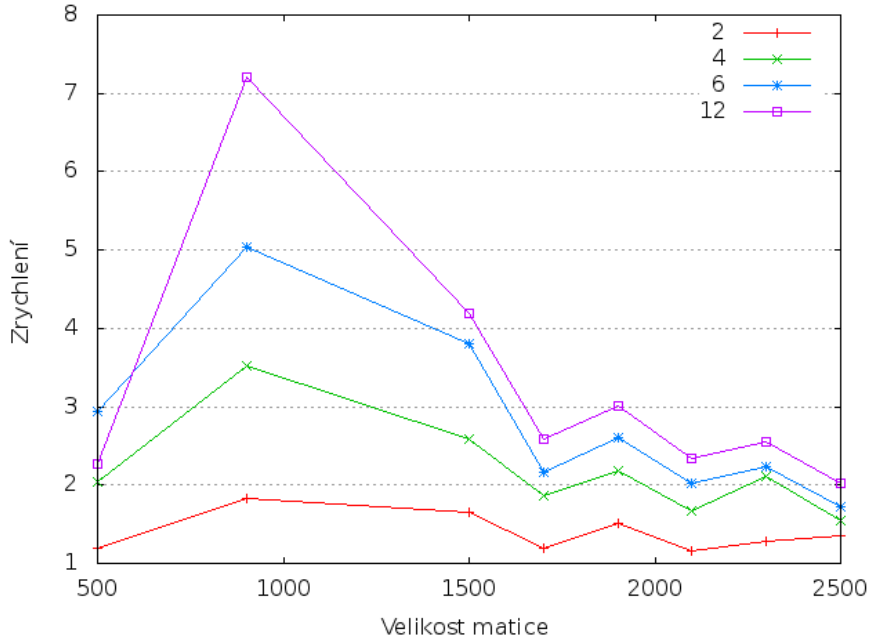
Obrázek 4.1: Časová náročnost Croutovy metody podle způsobu uložení matic

Z měření je jasně patrné, že nejhorší časovou náročnost má Croutova metoda (viz kap. 2.1.2) pro COO formát matic, protože nalezení jednoho prvku matice má logaritmickou složitost. Se zvětšující se velikostí matice roste i zaplnění výsledné matice a tím i časová náročnost Croutovy metody pro CRS/CCS

formát uložení matic, který byl nejrychlejší pro matice do velikossti 1 700. Hustý formát byl nejrychlejší pro matice větší než 1 700.

Tabulka 4.2: Časová náročnost Croutovy metody podle počtu vláken pro CSR/CCS formát uložení matic

Velikost matice	Čas [s]				
	1 vlákno	2 vlákna	4 vlákna	6 vlákna	12 vlákna
500	0,182	0,153	0,089	0,062	0,08
900	1,491	0,818	0,424	0,296	0,207
1500	12,08	7,33	4,67	3,17	2,878
1700	16,58	13,974	8,903	7,637	6,408
1900	39,788	26,23	18,297	15,233	13,233
2100	56,427	48,344	33,619	27,777	24,063
2300	106,321	82,642	50,546	47,442	41,606
2500	155,902	115,079	100,63	90,657	76,878

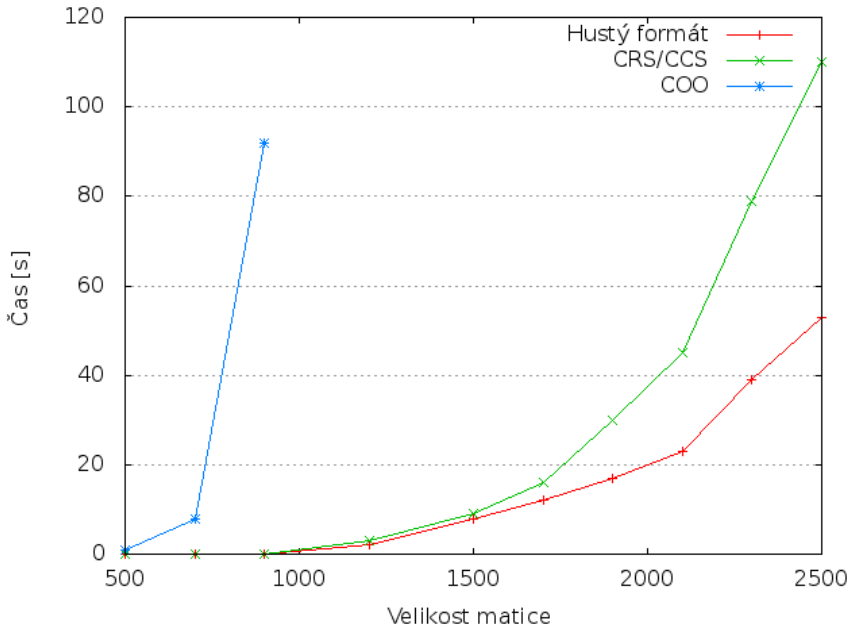


Obrázek 4.2: Zrychlení Croutovy metody podle počtu vláken oproti sekvenční verzi s CSR/CCS formátem uložení matic

S rostoucím počtem vláken se snižuje časová náročnost. Při použití 12 vláken došlo průměrně k 2, 5-krát zrychlení oproti sekvenční verzi. S rostoucí velikostí matice dochází ke snižování rozdílů zrychlení mezi různým počtem vláken.

Tabulka 4.3: Časová náročnost Choleského metody podle způsobu uložení matic

Velikost matice	Čas [s]		
	Hustý formát	CRS/CCS	COO
500	0,105	0,126	1,039
700	0,442	0,345	8,923
900	0,711	0,737	92,610
1200	2,695	3,735	
1500	8,192	9,960	
1700	12,765	16,738	
1900	17,124	30,950	
2100	23,722	45,652	
2300	39,746	79,499	
2500	53,875	110,617	

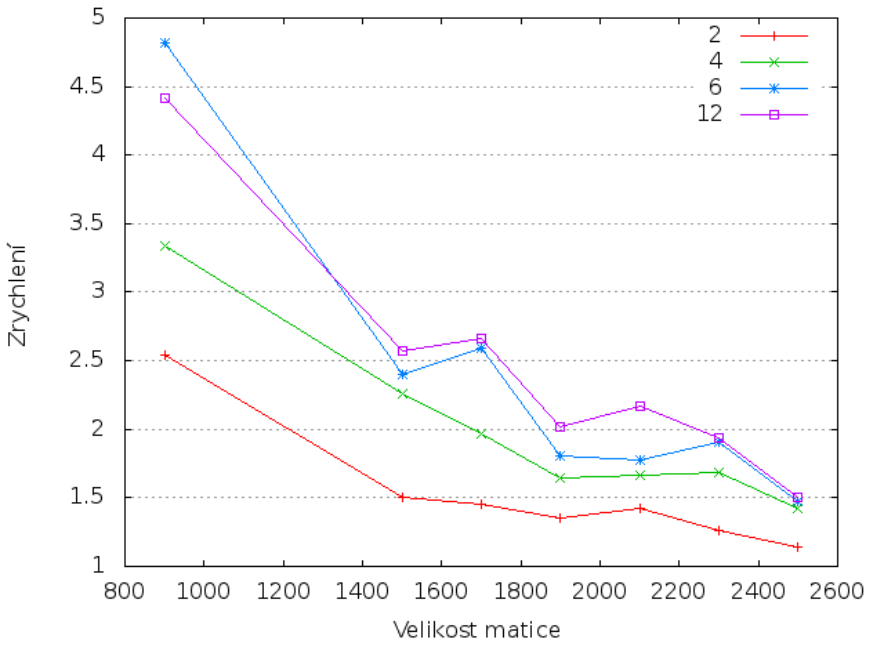


Obrázek 4.3: Časová náročnost Choleského metody podle způsobu uložení matic

Stejně jako u Croutovy metody i pro Choleského metodu (viz kap.2.1.3) je časově nejnáročnější COO formát uložení matic a nejméně časově náročný je CRS/CCS formát a to pro všechny velikosti matice. Se zvětšující velikostí matice roste rozdíl mezi hustým a CRS/CCS formátem.

Tabulka 4.4: Časová náročnost Choleského metody podle počtu vláken pro CSR/CCS formát uložení matic

Velikost matice	Čas [s]				
	1 vlákno	2 vlákna	4 vlákna	6 vláken	12 vláken
900	0,737	0,29	0,221	0,153	0,167
1500	9,96	6,643	3,578	4,416	3,875
1700	16,738	11,521	8,504	6,468	6,283
1900	30,95	22,893	18,772	17,091	15,316
2100	45,652	32,161	27,357	25,761	21,091
2300	79,499	62,813	47,301	41,782	41,123
2500	93,062	81,267	65,402	63,108	61,863

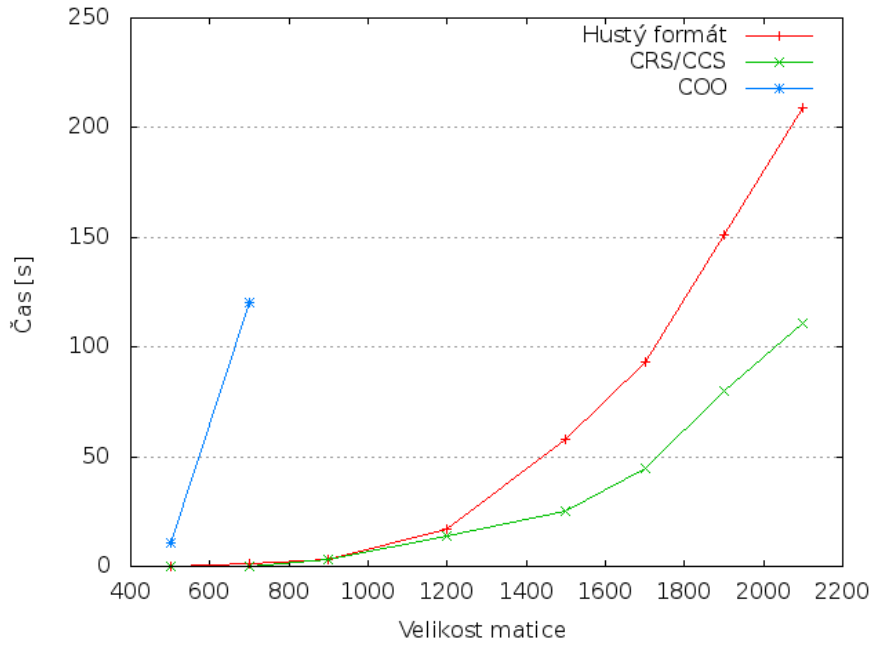


Obrázek 4.4: Zrychlení Choleského metody podle počtu vláken oproti sekvenční verzi s CSR/CCS formátem uložení matic

Se zvyšujícím se počtem vláken roste i zrychlení oproti sekvenční verzi. S rostoucí velikostí matic se ale zrychlení snižuje u všech počtů vláken.

Tabulka 4.5: Časová náročnost QR rozkladu podle způsobu uložení matic

Velikost matice	Čas [s]		
	Hustý formát	CRS/CCS	COO
500	0,426	0,499	11,789
700	1,885	0,898	120,824
900	3,823	3,947	
1200	17,959	14,793	
1500	58,137	25,098	
1700	93,934	50,027	
1900	151,441	80,15	
2100	209,186	111,331	

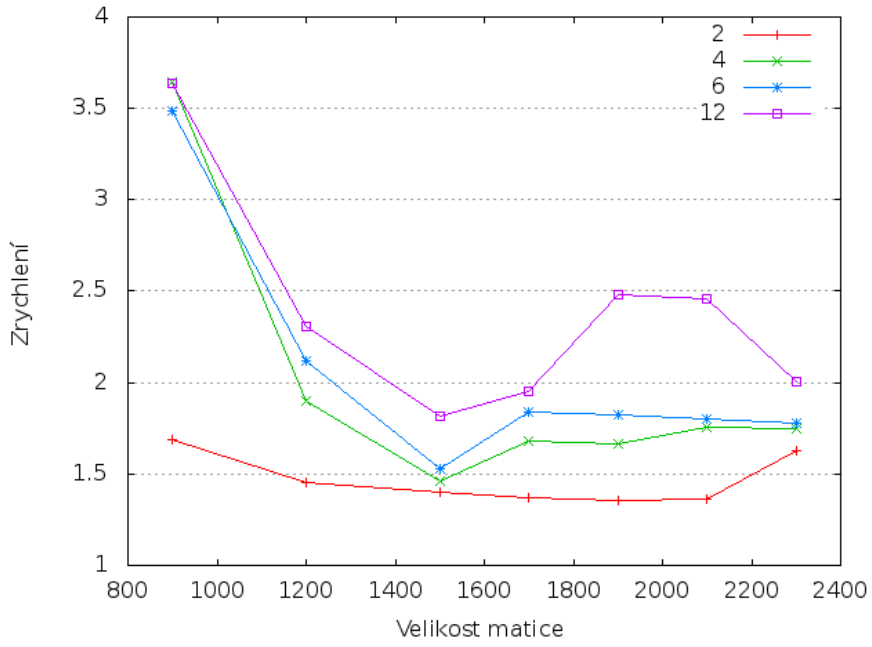


Obrázek 4.5: Časová náročnost QR rozkladu podle způsobu uložení matic

QR rozklad (viz. kap. 2.2) je časově nejnáročnější pro COO formát matic. Jako druhý je časově nejnáročnější hustý formát. Na rozdíl od předchozích metod nejlépe dopadl CRS/CCS formát i pro velké matice matice s vysokým počtem nenulových prvků.

Tabulka 4.6: Časová náročnost QR rozkladu podle počtu vláken pro CSR/CCS formát uložení matic

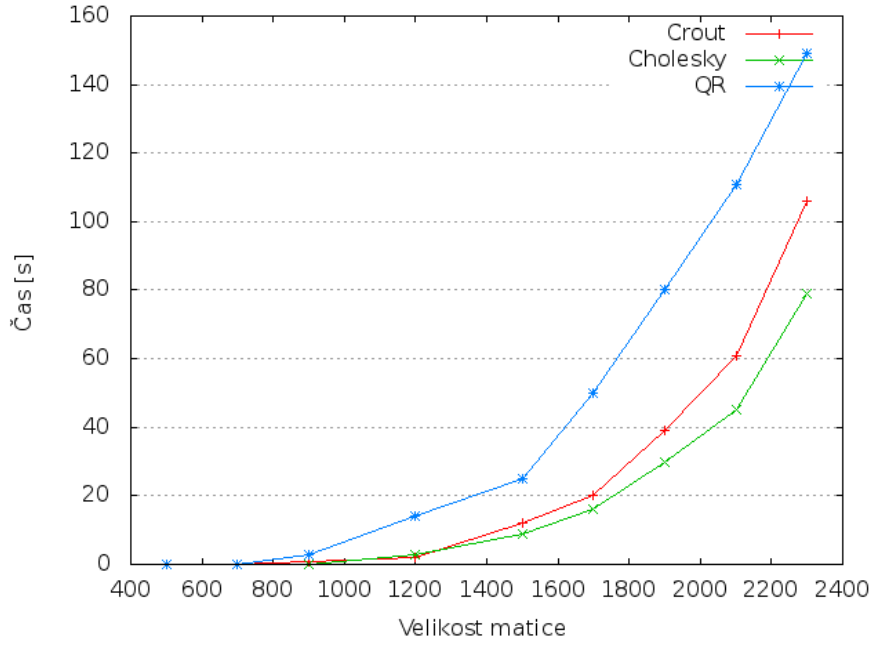
Velikost matice	Čas [s]				
	1 vlákno	2 vlákna	4 vlákna	6 vláken	12 vláken
900	3,947	2,341	1,082	1,131	1,086
1200	14,793	10,182	7,799	6,977	6,419
1500	25,098	17,934	17,182	16,431	13,83
1700	56,027	40,841	33,294	30,453	28,666
1900	80,15	59,093	48,158	44,001	32,353
2100	111,331	81,696	63,467	61,804	43,771
2300	149,307	91,834	85,4476	83,809	74,557



Obrázek 4.6: Zrychlení QR rozkladu podle počtu vláken oproti sekvenční verzi s CSR/CCS formátem uložení matic

Se zvyšujícím se počtem vláken roste i zrychlení QR rozkladu. Zrychlení zůstává podobné pro všechny velikosti matic. QR rozklad pro 12 vláken se oproti sekvenční verzi zrychlil průměrně 2 až 2,5-krát.

4.5.1 Porovnání metod



Obrázek 4.7: Časová náročnost metod pro CSR/CCS formát uložení matic

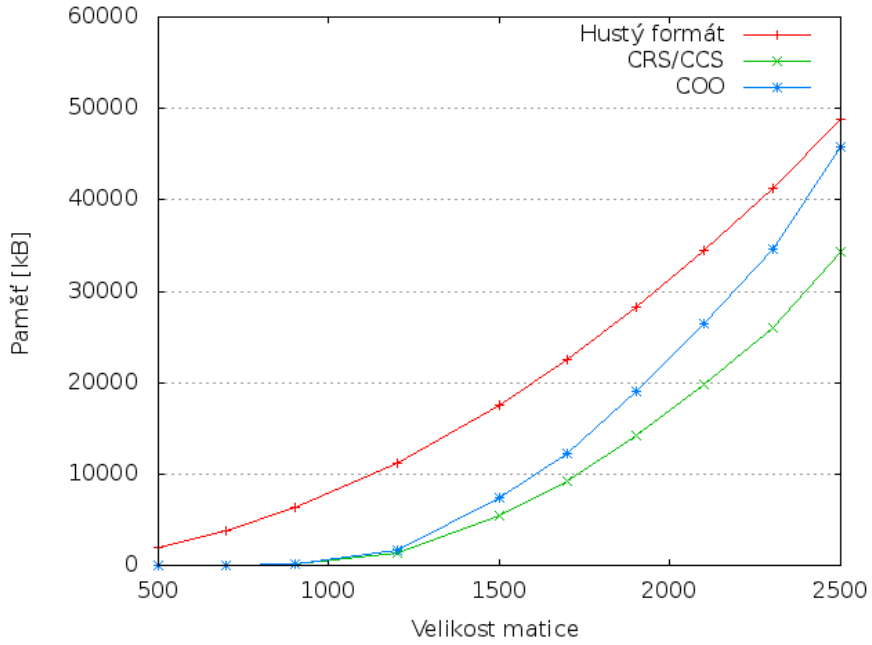
Z implementovaných metod je nejméně časově náročná Choleského metoda. Má ovšem i největší omezení a to, že matice musí být symetrické. Druhou nejméně časově náročnou metodou je Croutova metoda a nejvíce časově náročnou je QR rozklad.

Naměřené výsledky se shodují s teoretickou složitostí jednotlivých metod, kde pro matici o rozměrech $n \times n$ musí Choleského metoda vykonat $n^3/3$ operací, Croutova metoda $2n^3/3$ operací a QR rozklad $2n^3$ operací.

4.6 Paměťová náročnost

Tabulka 4.7: Paměťová náročnost Croutovy metody podle formátu uložení matic

Velikost matice	Počet nenulových prvků		Paměť [kB]		
	před	po rozkladu	Hustý formát	CRS/CCS	COO
500	625	645	1953,12	18,78	19,84
700	1225	1497	3828,12	37,36	42,53
900	2025	8439	6328,12	129,65	163,50
1200	3600	105541	11250,00	1288,37	1705,32
1500	5625	463672	17578,12	5511,29	7332,76
1700	7225	776365	22578,12	9195,97	12243,59
1900	9025	1208006	28203,12	14276,92	19016,10
2100	11025	1681699	34453,12	19853,01	26448,81
2300	13225	2203786	41328,12	25998,56	34640,79
2500	15625	2913243	48828,12	34342,20	45763,56



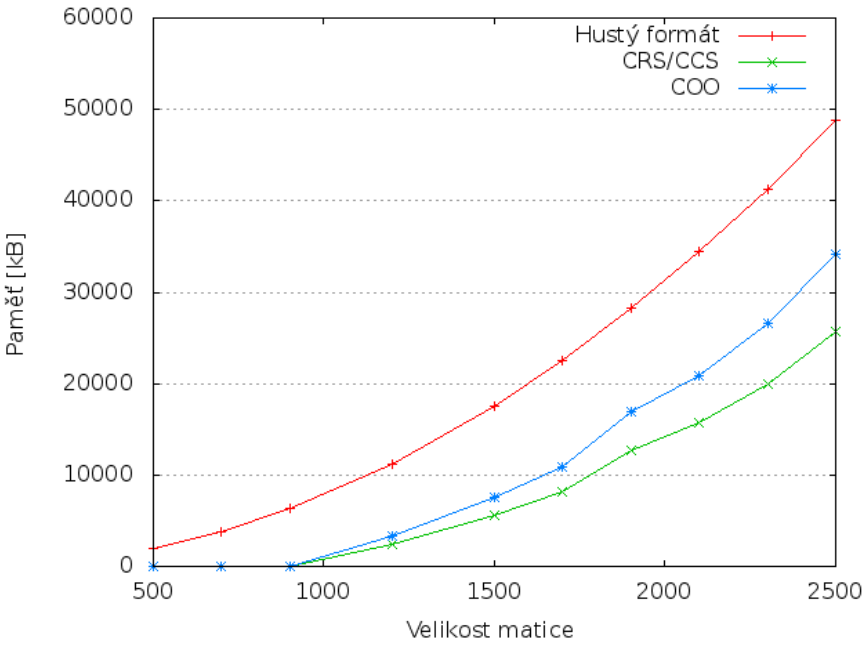
Obrázek 4.8: Paměťová náročnost Croutovy metody podle formátu uložení matic

S rostoucí velikostí matice roste i paměťová náročnost hustého formátu uložení matic. U CRS/CCS formátu záleží kromě velikosti matice i na počtu

nenulových prvků. Jeho paměťová náročnost je menší než u hustého formátu. COO formát má o něco větší paměťovou náročnost než CRS/CCS formát a pro matice s počtem nenulových prvků větším než 50% je paměťově náročnější než hustý formát.

Tabulka 4.8: Paměťová náročnost Choleského metody podle formátu uložení matic

Velikost matice	Počet nenulových prvků		Pamět [kB]		
	před	po rozkladu	Hustý formát	CRS/CCS	COO
500	625	672	1953,12	17,36	14,28
700	1225	1840	3828,12	31,63	29,75
900	2025	4636	6328,12	56,60	59,50
1200	3600	414092	11250,00	2470,85	3273,17
1500	5625	956406	17578,12	5666,19	7528,31
1700	7225	1381390	22578,12	8169,61	10862,66
1900	9025	2161662	28203,12	12755,97	16974,26
2100	11025	2654470	34453,12	15659,15	20841,61
2300	13225	3389640	41328,12	19983,58	26603,97
2500	15625	4348992	48828,12	25622,75	34119,32

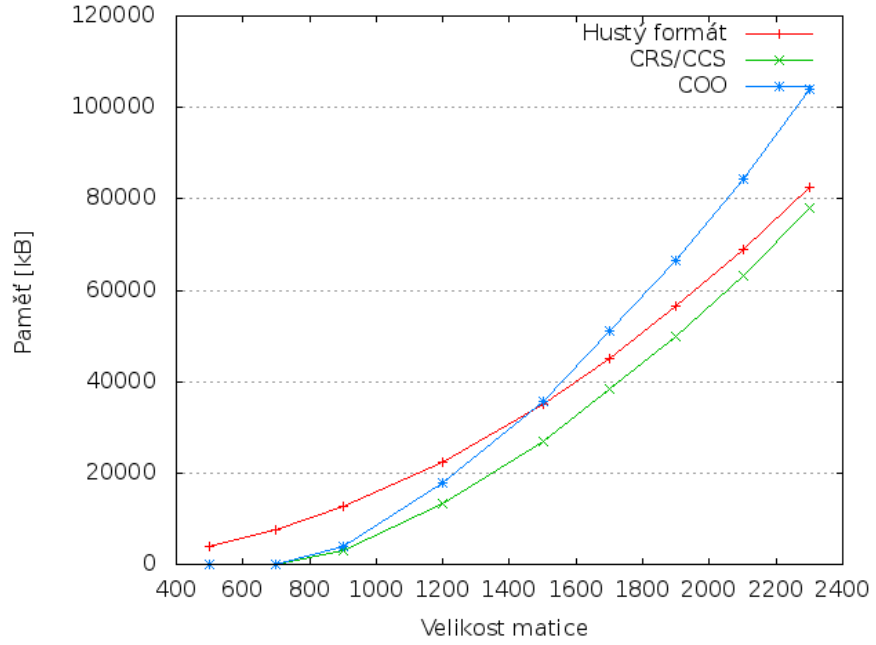


Obrázek 4.9: Paměťová náročnost Choleského metody podle formátu uložení matic

Podobně jako u Croutovy metody i u Choleského metody je paměťově nejnáročnější hustý formát. CRS/CCS formát je paměťově nejméně náročný a COO formát je druhý nejméně náročný. CRS/CCS a COO jsou paměťově méně nároční především díky tomu, že všechny matice jsou symetrické a pro jejich uložení stačí uložit pouze polovinu nenulových prvků.

Tabulka 4.9: Paměťová náročnost QR rozkladu podle formátu uložení matic

Velikost matice	Počet nenulových prvků		Pamět [kB]		
	před	po rozkladu	Hustý formát	CRS/CCS	COO
500	625	1229	3906,25	27,58	28,96
700	1225	4364	7656,25	73,69	87,32
900	2025	253293	12656,25	3002,55	3989,34
1200	3600	1134018	22500,00	13345,52	17775,28
1500	5625	2278621	35156,25	26786,08	35691,34
1700	7225	3260095	45156,25	38308,82	51051,87
1900	9025	4243774	56406,25	49859,75	66449,98
2100	11025	5381491	68906,25	63218,15	84258,06
2300	13225	6660327	82656,25	78077,66	104067,60



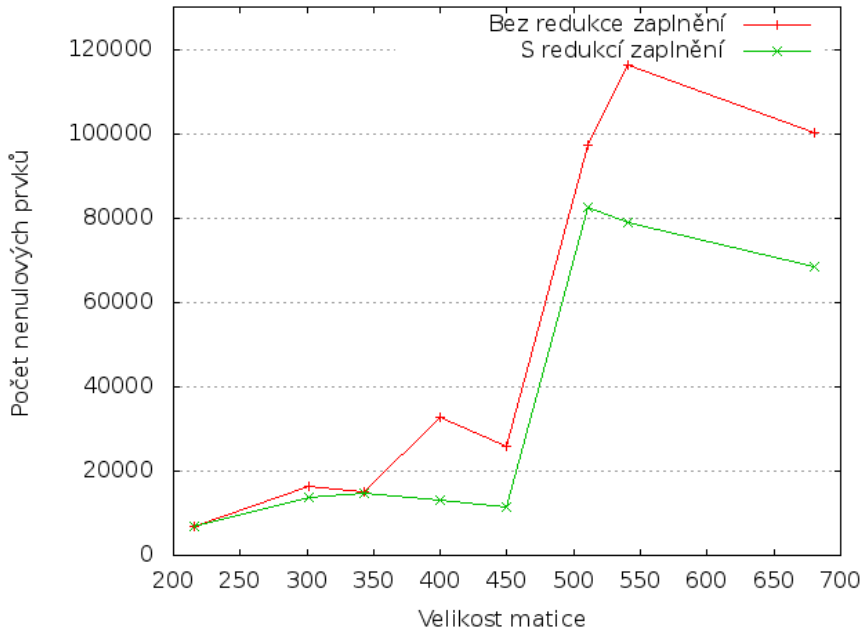
Obrázek 4.10: Paměťová náročnost QR rozkladu podle formátu uložení matic

QR rozklad je paměťově přibližně dvakrát náročnější než Croutova a Choleského metoda, protože matice **Q** není trojúhelníková matice a pro uložení výsledků nestačí jen jedna matice. Nejméně paměťově náročný je CRS/CCS formát uložení matic, přestože se blíží k paměťové náročnosti hustého formátu, protože výsledné matice obsahují větší množství nenulových prvků.

4.6.1 Redukce zaplnění matic

Tabulka 4.10: Redukce zaplnění výsledných matic při použití Croutovy metody

Velikost matice	Počet nenulových prvků	
	bez redukce zaplnění	s redukcí zaplnění
216	6945	6856
301	16529	13607
343	15179	14821
400	32774	12968
450	25764	11486
511	97284	82387
541	116156	78895
680	100293	68533



Obrázek 4.11: Redukce zaplnění výsledných matic při použití Croutovy metody

Naměřené hodnoty potvrdily, že použitím metody pro redukci zaplnění matic (viz kap. 2.4) lze dosáhnout výrazného snížení počtu nenulových prvků. V nejlepším případě se jejich počet snížil o 55%. V nejhorším případě se jejich počet snížil velmi málo.

Závěr

V práci byly implementovány metody pro LU rozklad matic a to konkrétně Croutova, Choleského a QR metoda. Metody byly implementovány pro husté matice a následně upraveny pro různé formáty ukládání řídkých matic. Dále byla provedena jejich paralelizace nad sdílenou pamětí. Metody byly testovány a porovnávány z hlediska časové a paměťové náročnosti.

Na základě naměřených hodnot bylo zjištěno, že nejméně časově náročná je Choleského metoda. Tato metoda má ale také největší omezení a to, že funguje pouze na symetrických maticích. Druhá nejméně časově náročná je Croutova metoda. Nejvíce časově náročný je QR rozklad. Naměřené hodnoty se tak shodují s teoretickým předpokladem, kde podle velikosti matice musí nejméně operací vykonat Choleského metoda a nejvíce QR rozklad.

Z hlediska způsobu uložení matic dopadl nejhůře COO formát, který hledá prvky s logaritmickou složitostí. Pro Croutovu a Choleského metodu dopadl nejlépe hustý formát. CRS/CCS formát si vedl lépe pouze u matic s malými rozměry a malým počtem nenulových prvků ve výsledné matici. Pro QR rozklad dopadl nejlépe CRS/CCS formát pro všechny velikosti matic.

Při srovnání paměťové náročnosti dopadl nejlépe CRS/CCS formát uložení matic. Jeho paměťová náročnost je ovšem závislá kromě velikosti matice i na počtu nenulových prvků a proto přestává být paměťově výhodný oproti hustému formátu u Croutovy metody přibližně při 33% a QR rozkladu přibližně při 66% zaplnění výsledné matice, u Choleského metody se ukládá pouze polovina prvků, proto je pro ni CRS/CCS formát výhodný i při 100% zaplnění výsledné matice.

Pro paralelní verze došlo u všech metod ke zrychlení. Pro Croutovu a Choleského metodu se však se zvětšující se velikostí matic zrychlení klesá. Pro QR rozklad zůstává zrychlení podobné pro všechny velikosti matic.

Literatura

[1] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. : *Numerical Recipes in C.* CAMBRIDGE UNIVERSITY PRESS, 2002, ISBN 0-521-43108-5.

[2] Vandenberghe Lieven, *Cholesky factorization.* UCLA engineering. [Citace: 20. 3. 2015].
`http://www.seas.ucla.edu/~vandenbe/103/lectures/chol.pdf`

[3] Sudipto Banerjee, Anindya Roy: *Linear Algebra and Matrix Analysis for Statistics.* CRC Press, 2015, ISBN 978-1420095388.

[4] Tewarson R. P.: *Sparse Matices.*
Hong Kong: Academic Press, 1973, ISBN 01-268-5650-8.

[5] Vandenberghe Lieven, *LU factorization.* UCLA engineering. [Citace: 20. 3. 2015].
`http://www.seas.ucla.edu/~vandenbe/103/lectures/lu.pdf`

[6] Dongarra J.: *Survey of Sparse Matrix Storage Formats.* [Citace: 11. 4. 2015].
`http://netlib.org/linalg/html_templates/node90.html`

[6] Olšák Petr: *Lineární algebra.* [Citace: 11. 4. 2015].
`ftp://math.feld.cvut.cz/pub/olsak/linal/linal.pdf`

[7] *Gram-Schmidt in 9 lines of MATLAB.* [Citace: 13. 4. 2015].
`http://web.mit.edu/18.06/www/Essays/gramschmidtmat.pdf`

Příloha A

Uživatelská příručka

A.1 Spuštění programu

Program se zkompiluje zadáním příkazu make do příkazového řádku ve složce s programem. Zkompilovaný program se spouští z příkazového řádku. Při spuštění se zadávají parametry, které ovlivňují jeho chování.

1. První parametr určuje jaká metoda bude použita. Může nabývat hodnot 0, 1, 2 a 3.
   a) 0 - Croutova metoda
   b) 1 - Choleského metoda
   c) 2 - QR rozklad
   d) 3 - Croutova medota s redukcí zaplnění
2. Druhý parametr určuje v jakém formátu bude matice uložena v paměti. Může nabývat hodnot 0, 1 a 2.
   a) 0 - Hustý formát
   b) 1 - CRS/CCS formát
   c) 2 - COO formát
3. Třetí parametr určuje s kolika vlákny bude program spuštěn.
4. Čtvrtý parametr je cesta a název souboru, ve kterém je uložena vstupní matice.
5. Pátý parametr (nepovinný) určuje zda se má do konzole vypsat výsledek.
   a) 0 nebo bez pátého parametru - Výsledek se nevypíše
   b) 1 - Výsledek se vypíše do konzole

Například následující výraz:

$$lu\ 0\ 1\ 4\ ./data/matice.mtx\ 1$$

spustí program provádějící Croutovu metodu, s maticemi uloženými v CRS/CCS formátu, běžícím se 4 vlákny, vstupním souborem bude matice.mtx a výsledek bude vypsán do konzole.

A.2 Výstup programu programu

```
Croutova metoda
CRS/CCS matice.
-----------------------
Cas:
4.1196e-05
Celkem nenulovych prvku:
7
-----------------------
Vysledek:
L:
.   .   .   .
.   .   .   .
1.5 .   .   .
.   .   3   .
U:
2   .   .   .
.   5   .   .
.   .   2   2
.   .   .   1
```

Na výstupu programu je uvedena použitá metoda, formát uložení matic, časová náročnost potřebná k provedení rozkladu, počet nenulových prvků ve výsledných maticích a pokud je při spuštění programu uvedeno, že se má výsledek vypsat do konzole, vypíšou se i výsledné matice. Ve výsledných maticích jsou nuly nahrazeny tečkami a u Croutovy metody nevypisuje na diagonále matice **L** jedničky.

Příloha B

Seznam použitých zkratek

COO Coordinate

CRS Compressed Row Storage

CCS Compressed Column Storage

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `ZZP.pdf` ..... zadání bakalářské práce
- `program`
  - `src` ..... zdrojové kódy implementace
  - `data` ..... testované matice
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `img` ..... grafy k práci
- `text`
  - `BP_Kusý_Stanislav_2015.pdf` ..... text práce ve formátu PDF