Sem vložte zadání Vaší práce.

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

Bakalářská práce

Použití intervalové aritmetiky v řešičích soustav lineárních rovnic

Martin Kulle

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

11. května 2015

Poděkování

Děkuji vedoucímu mé práce Ing. Ivanu Šimečkovi, Ph.D. za cenné rady a ochotu při vedení této práce. Dále bych rád poděkoval svým rodičům za podporu během studia.

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

Kulle, Martin. *Použití intervalové aritmetiky v řešičích soustav lineárních rovnic*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá řešením soustavy lineárních rovnic v intervalové aritmetice. Pro řešení soustav rovnic jsou použity metody Gaussovy eliminace, Jacobiova metoda, Gauss-Seidelova metoda a metoda sdružených gradientů. Práce porovnává vliv těchto metod na výslednou šířku intervalů. Dále porovnává vliv pivotace při použití různých heuristik a také analyzuje výslednou šířku intervalů při použití metody předpodmínění. Metody jsou implementovány pomocí knihoven PROFIL/BIAS a Boost, které jsou mezi sebou porovnány z hlediska výkonnosti.

Klíčová slova intervalová aritmetika, soustava lineárních intervalových rovnic, PROFIL, Boost, předpodmínění, pivotace

Abstract

The thesis describes solution of system of linear equation in interval arithmetic. For solving systems of linear equation are used Gauss elimination, Jacobi method, Gauss-Seidel method and Conjugate gradient method. The thesis compares influence of these methods on resulting width of intervals. Also, it compares influence of pivotization with different heuristic and it analyses resulting width for preconditioning. Methods are implemented by using PROFIL/BIAS and Boost library, which are compare in terms of performance.

Keywords interval arithmetic, linear interval system, PROFIL, Boost, preconditioning, pivotization

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Úvod

Intervalová aritmetika byla představena v roce 1957 Ramonem Moorem, který také následně napsal knihu Interval Analysis. Intervalová aritmetika má široké množství použití jako počítání s hodnotami, které jsou měřeny s určitou nepřesností, nebo počítání zaokrouhlovací chyby při numerický výpočtech na počítačích.
Při použití intervalové aritmetiky v soustavě lineárních rovnic kde bereme v úvahu možnou nepřesnost v koeficientech rovnic, je hledané řešení vektor intervalů, ve kterém se výsledný vektor nachází. Důležitou vlastností výsledného vektoru je jeho správnost ale zároveň šířka jednotlivých intervalů. Pokud by výsledný interval byl $[-\infty, \infty]$, bylo by to sice správné řešení, ale pro nás nic neříkající.
Úkolem této bakalářské práce je nastudovat už existující knihovny pro práci s intervalovou aritmetikou, implementovat pomocí nich metody pro řešení soustav lineárních intervalových rovnic a porovnat výkon těchto knihoven a výslednou šířku intervalů u jednotlivých metod. Pro řešení soustavy rovnic jsou použity metody jak přímé jako Gaussova eliminace, tak metody iterační jako například Gauss-Seidelova metoda.
Obsah této práce je následující:

1. kapitola seznamuje čtenáře s intervalovou aritmetikou a definuje některé vlastnosti intervalové aritmetiky.
2. kapitola popisuje metody, které můžeme použít při řešení soustavy lineárních intervalových rovnic.
3. kapitola porovnává existující knihovny pro intervalovou aritmetiku na architektuře x86.
4. kapitola popisuje realizaci této bakalářské práce a implementaci jednotlivých metod.
5. kapitola se zabývá měřením výkonnosti knihoven a výslednou šířkou jednotlivých metod.

Kapitola 1

Intervalová aritmetika

Na rozdíl od běžné aritmetiky kde počítáme s reálnými čísly, v intervalové aritmetice počítáme s intervaly, které jsou dány dvojicí reálných čísel, které ohraničují uzavřený interval. Tento interval reprezentuje rozsah, ze kterého může být daná hodnota, ale neříká nic o pravděpodobnosti, jakou hodnota nabývá.
Intervalovou aritmetiku používáme zejména v reálných aplikacích při odhadu chyb a jejich vlivu na výsledek. Také nám zaručuje meze, ve kterých se řešení nachází a které nepřesáhne.
Intervalová aritmetika má i další rozšíření, například aritmetika která umožňuje dělit intervalem, který obsahuje nulu [1], nebo intervalová aritmetika nad komplexními čísly [2]. V této kapitole čerpáme z [3] a z [4].

1.1 Interval

Definice 1.1. Uzavřený reálný interval je uzavřená množina reálných čísel

$$\mathbf{IR} = \{x \in \mathbf{R} \mid \underline{x} \leq \tilde{x} \leq \overline{x}\}.$$

Dále v textu budeme dolní mez intervalu označovat $\underline{x}$ a horní mez $\overline{x}$. Prvek, který leží v intervalu, budeme značit symbolem $\tilde{x}$.
Interval můžeme popsat některými jeho vlastnostmi. Mezi základní vlastnosti patří šířka intervalu

$$wid(x) = \overline{x} - \underline{x},$$

kde $x \in \mathbf{IR}$. Průměr intervalu $rad(x)$ je definován jako polovina šířky intervalu

$$rad(x) = \frac{1}{2}wid(x) = \frac{1}{2}(\overline{x} - \underline{x}).$$

Protože interval chápeme jako rovnoměrné rozdělení, je střední hodnota definována jako střed intervalu

$$mid(x) = \frac{1}{2}(\overline{x} + \underline{x}).$$

Absolutní hodnota je v intervalové aritmetice definována

$$abs(x) = max\{|\tilde{x}| \mid \tilde{x} \in x\},$$

v jistých případech jsme ale nuceni použít minimální absolutní hodnotu, proto definujeme mignitude jako

$$mig(x) = min\{|\tilde{x}| \mid \tilde{x} \in x\}.$$

Samozřejmě v intervalové aritmetice je definováno velké množství dalších vlastností, které zde nejsou zmíněny, protože v této práci nebudou použity.

1.2 Operace v intervalové aritmetice

Definice 1.2. Binární operace v intervalové aritmetice nad intervaly x a y je definována jako

$$x \circ y = \{\tilde{x} \circ \tilde{y} \mid \tilde{x} \in x,\ \tilde{y} \in y\}.$$

Výsledný interval základních binárních operací jako sčítání, odčítání, násobení a dělení lze určit jen pomocí mezí operátorů

$$x \circ y = [min(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}), max(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y})].$$

Pro operace sčítání a odčítání to můžeme dále zjednodušit.

$$\begin{aligned}
&[a,b] + [c,d] = [a+c, b+d] \\
&[a,b] - [c,d] = [a-d, b-c] \\
&[a,b] \cdot [c,d] = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)] \\
&\frac{[a,b]}{[c,d]} = \left[min\left(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d}\right), max\left(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d}\right)\right] \text{pokud } 0 \notin [c,d]
\end{aligned}$$

Je důležité si uvědomit, že každá z výše uvedených operací interval rozšiřuje. Pokud máme intervaly a, b a $wid(b) > 0$, pak $(a/b) \cdot b = c \neq a$, ale interval a je podmnožina c.

Příklad 1.3.

$$\frac{[2,3]}{[5,6]} = \left[\frac{2}{6}, \frac{3}{5}\right] \cdot [5,6] = [1.5, 3.6] \supset [2,3]$$

Proto při různých výpočtech jako je například řešení soustavy lineárních rovnic, je důležité minimalizovat počet provedených operací. Tím zamezíme rozšiřování výsledného intervalu.

1.3 Intervalový vektor

Definice 1.4. Reálný intervalový vektor je uspořádaná n-tice reálných intervalů.

$$\mathbf{IR}^n = \{x \in \mathbf{R}^n \mid \quad \underline{x} \leq \tilde{x} \leq \overline{x}\}.$$

Intervalový vektor budeme značit znakem $\boldsymbol{x}$. Při práci s vektory používáme stejné vlastnosti jako u intervalů. Dolní mez vektoru $\boldsymbol{x}$ je definována jako vektor dolních mezí vektoru $\boldsymbol{x}$

$$inf(\boldsymbol{x}) = \underline{x}.$$

Podobně je definována horní mez vektoru

$$sup(\boldsymbol{x}) = \overline{x}.$$

Dále jsou podobným způsobem definovány další vlastnosti vektorů

$$\begin{aligned}
wid(\boldsymbol{x}) &= \overline{x} - \underline{x} \\
rad(\boldsymbol{x}) &= \frac{1}{2} wid(x) = \frac{1}{2}(\overline{x} - \underline{x}) \\
mid(\boldsymbol{x}) &= \frac{1}{2}(\overline{x} + \underline{x}) \\
abs(\boldsymbol{x}) &= max\{|\tilde{x}| \mid \tilde{x} \in x\} \\
mig(\boldsymbol{x}) &= min\{|\tilde{x}| \mid \tilde{x} \in x\}.
\end{aligned}$$

Pro vektor dále definujeme vzdálenost mezi vektory $\boldsymbol{x}$ a $\boldsymbol{y}$.

$$dist(\boldsymbol{x}, \boldsymbol{y}) = sup\{|\underline{x} - \underline{y}|, |\overline{x} - \overline{y}|\}$$

Definice 1.5. Reálná intervalová matice je matice, jejíž prvky tvoří intervaly

$$\mathbf{IR}^{n \times m} = \{x \in \mathbf{R}^{n \times m} \mid \quad \underline{x} \leq \tilde{x} \leq \overline{x}\}.$$

Pro matice můžeme použít stejné vlastnosti, jako jsou definovány pro intervalové vektory.

Definice 1.6. Nechť $\mathbf{A}$ je matice typu (n, n). Matice $\mathbf{A}$ je diagonálně dominantní, právě když absolutní hodnota prvku na diagonále je větší nebo rovna součtu absolutních hodnot ostatních prvků, které jsou ve stejném řádku

$$\forall i \in \{1, \ldots, n\} \quad |a_{i,i}| \geq \sum_{j \neq i} |a_{i,j}|.$$

Definice 1.7. Matice $\mathbf{A}$ je pozitivně definitní, pokud platí

$$\boldsymbol{x} \neq 0 \implies \boldsymbol{x}^T \mathbf{A} \boldsymbol{x} > 0.$$

1.4 Soustava lineárních intervalových rovnic

Soustavou lineárních intervalových rovnic rozumíme

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,3}x_3 &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,3}x_3 &= b_2 \\ &\vdots \\ a_{4,1}x_1 + a_{4,2}x_2 + \cdots + a_{4,3}x_3 &= b_4, \end{aligned}$$

kde $a_{i,j}$ jsou koeficienty, x_i jsou neznámé a b_i jsou absolutní členy rovnice. Soustava lineárních rovnic může mít žádné, jedno nebo více řešení. V každém případě ale řešení soustavy lineárních rovnic tvoří lineární prostor.
V lineární algebře se soustava lineárních rovnic zapisuje pomocí matice $\mathbf{A}$ a vektorů $\boldsymbol{x}$ a $\boldsymbol{b}$

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Soustavu rovnic tedy můžeme vyjádřit jako

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Pro soustavu lineárních rovnic budeme používat zkrácený zápis

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}.$$

Vektor $\boldsymbol{x}$ hledáme tak, aby šířky jeho intervalů byly co nejmenší, snažíme se tedy snížit počet operací použitých při výpočtu zejména odčítání a dělení. Pro výsledný vektor tedy nebude platit $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ ale $\boldsymbol{b} \in \mathbf{A}\boldsymbol{x}$.
Dále se v textu omezíme na soustavy rovnic, kde $\mathbf{A}$ je čtvercová matice a je regulární. V takovém případě soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ má pro každé $\boldsymbol{b}$ právě jedno řešení. To můžeme ověřit pomocí Cramerova pravidla, které říká, že pro i-tou složku řešení platí

$$x_i = \frac{det(\mathbf{B}_i)}{det(\mathbf{A})},$$

kde matice $\mathbf{B}_i$ je shodná s maticí $\mathbf{A}$ mimo i-tého sloupce, který je nahrazen vektorem $\boldsymbol{b}$.

Kapitola 2

Metody řešení soustavy rovnic v intervalové aritmetice

Pro řešení soustav lineárních intervalových rovnic můžeme použít stejné metody jako při řešení soustav lineárních rovnic s reálnými koeficienty. Při použití těchto metod ale nemusíme vždy nalézt řešení soustavy rovnic, i když existuje. Je to způsobeno tím, že postupnými úpravami matice rozšíříme intervaly tak, že obsahují nulu. Pokud algoritmus je nucen tímto intervalem dělit, nemůže pokračovat, protože dělení intervalem, který obsahuje nulu, není definováno. Zvětšení úspěšnosti řešení soustav lineárních intervalových rovnic dosáhneme pomocí metod jako pivotace nebo přepodmínění, které jsou popsány níže.
V této kapitole jsou popsány metody přímé a některé iterativní metody. Tyto metody jsou také popsány v [5], [6] nebo v [7].

2.1 Trojúhelníkové soustavy

Řešení soustavy lineárních rovnic s trojúhelníkovou maticí je velmi efektivní, proto se většina přímých metod snaží obecnou soustavu rovnic převádět na trojúhelníkovou soustavu. Dopředný chod Gaussovy eliminační metody je v podstatě převedení řešení dané soustavy na trojúhelníkovou soustavy.

Definice 2.1. Nechť $\mathbf{L} \in \mathbf{IR}^{n\times n}$ je dolní trojúhelníková matice

$$L = \begin{pmatrix} a_{1,1} & 0 & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & 0 & \cdots & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{pmatrix}.$$

Budeme předpokládat, že $\mathbf{L}$ je regulární. To znamená, že prvky na diagonále musí být rozdílné od nuly, protože

$$det(\mathbf{L}) = \prod_{i=1}^{n} a_{i,i}.$$

Pokud soustavu $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ rozepíšeme do rovnic

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

a je-li $a_{1,1} \neq 0$ a $a_{2,2} \neq 0$, pak dostáváme

$$x_1 = \frac{b_1}{a_{1,1}}, \quad x_2 = \frac{b_2 - a_{2,1}x_1}{a_{2,2}}.$$

Obecně tedy pokud platí $\forall i \; a_{i,i} \neq 0$ a známe hodnoty $x_1, x_2, \cdots, x_{i-1}$, pak

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j}x_j}{a_{i,i}}. \tag{2.1}$$

Z toho je také vidět, že pro každé $\boldsymbol{b}$ má soustava s horní trojúhelníkovou maticí právě jedno řešení.
Nevýhodou tohoto řešení je, že pro výpočet i-tého prvku řešení musíme znát předchozí prvky řešení. Nemůžeme tedy tuto metodu paralelizovat a musíme výpočet provádět sekvenčně.

2.2 Gaussova eliminační metoda

Gaussova eliminační metoda je základní metoda pro řešení soustav lineárních rovnic. Nejprve soustavu rovnic přepíšeme do maticového tvaru, kde hodnoty v matici reprezentují koeficienty soustavy rovnic a poslední sloupec je tvořen vektorem absolutních členů rovnice

$$\left(\begin{array}{cccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & b_n \end{array}\right).$$

Následně se snažíme dostat matici do tvaru horní trojúhelníkové. Postupujeme postupně po sloupcích. Pro první sloupec začínáme na druhém řádku a

přičítáme k i-tému řádku první řádek vynásobený koeficientem $(-a_{i,1}/a_{1,1})$ a dostáváme matici

$$\left(\begin{array}{cccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a'_{n,2} & \cdots & a'_{n,n} & b'_n \end{array}\right).$$

Pro druhý sloupec začínáme na třetím řádku a přičítáme k i-tému řádku druhý řádek vynásobený koeficientem $(-a_{i,2}/a_{2,2})$. Takto pokračujeme postupně v dalších sloupcích a tím dostáváme horní trojúhelníkovou matici ve tvaru

$$\left(\begin{array}{cccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a''_{n,n} & b''_n \end{array}\right).$$

Tuto matici můžeme vyřešit pomocí metody trojúhelníkové soustavy, která je uvedena výše. Jenom si musíme dát pozor na to, že nemáme dolní trojúhelníkovou matici ale horní trojúhelníkovou matici. Pro výpočet i-tého prvku tedy musíme znát $x_{i+1}, x_{i+2}, \cdots, x_n$ a vzorec 2.1 upravíme

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{i,j}x_j}{a_{i,i}}. \tag{2.2}$$

Pro zmenšení šířky výsledných intervalů můžeme použít metody řádkové, sloupcové nebo celkové pivotace, která před vytvářením nul v i-tém sloupci hledá takzvaný pivot.
Hledání pivotu spočívá v nalezení prvku matice v řádku, respektive sloupci, který má největší absolutní hodnotu. V intervalové aritmetice se někdy také pivot hledá pomocí mignitude nebo střední hodnoty. Při sloupcové pivotaci je následně řádek, ve kterém se nachází pivot, zaměněn s i-tým řádkem. V případě řádkové pivotace je zaměněn sloupec s pivotem s i-tým sloupcem, ale je také potřeba vyměnit prvky ve vektoru $\boldsymbol{b}$ a to b_p, kde p je index sloupce, ve kterém se nachází pivot, za b_i.

2.3 LU rozklad

LU rozklad je metoda, ve které je regulární matice soustavy $\mathbf{A}$ rozložena na matice $\mathbf{L}$ (dolní trojúhelníková matice) a $\mathbf{U}$ (horní trojúhelníková matice), a platí rovnost

$$\mathbf{PA} = \mathbf{LU},$$

kde $\mathbf{P}$ je permutační matice, která říká, jak byly prohozeny řádky matice $\mathbf{A}$. Postup pro vytvoření matic $\mathbf{L}$ a $\mathbf{U}$ ukážeme na jednoduchém příkladu matic

3 x 3. Nejdříve vytvoříme matici $\mathbf{L_1}$, tak aby platilo

$$\mathbf{L_1 A} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{a_{2,1}}{a_{1,1}} & 1 & 0 \\ -\frac{a_{3,1}}{a_{1,1}} & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a'_{2,2} & a'_{2,3} \\ 0 & a'_{3,2} & a'_{3,3} \end{pmatrix}.$$

Podobným způsobem vytvoříme matici $\mathbf{L_2}$

$$\mathbf{L_2 L_1 A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a_{3,2}}{a_{2,2}} & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a'_{2,2} & a'_{2,3} \\ 0 & a'_{3,2} & a'_{3,3} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a'_{2,2} & a'_{2,3} \\ 0 & 0 & a''_{3,3} \end{pmatrix}$$

a tím získáme horní trojúhelníkovou matici, tedy matici $\mathbf{U}$, kterou hledáme. Protože matice $\mathbf{L_1}$ a $\mathbf{L_2}$ jsou dolní trojúhelníkové a platí, že inverzní matice k dolní trojúhelníkové matici je dolní trojúhelníková a násobení dolních trojúhelníkových matic je také dolní trojúhelníková matice, pak platí

$$\begin{aligned} \mathbf{L_2 L_1 A} &= \mathbf{U} \\ \mathbf{A} &= (\mathbf{L_2 L_1})^{-1}\mathbf{U} \\ \mathbf{A} &= \mathbf{L_1}^{-1}\mathbf{L_2}^{-1}\mathbf{U} \\ \mathbf{A} &= \mathbf{LU} \end{aligned}$$

kde $\mathbf{L} = \mathbf{L_1}^{-1}\mathbf{L_2}^{-1}$. Sestrojení matice $\mathbf{L}$ tímto způsobem je obtížné. Pro soustavu s n proměnnými musíme provést násobení n matic o rozměrech n x n a následně spočítat inverzní matici. S výhodou ale můžeme využít vlastnosti matic $\mathbf{L_i}$. Matici inverzní k $\mathbf{L_i}$ vytvoříme tak, že prvky pod diagonálou v i-tém sloupci vynásobíme -1. Součin matic $\mathbf{L_1}^{-1}, \mathbf{L_2}^{-1}, \ldots, \mathbf{L_N}^{-1}$ je také dolní trojúhelníková matice, jejíž i-tý sloupec je tvořen i-tým sloupcem z matice $\mathbf{L_i}$. Díky tomu můžeme pro výpočet použít algoritmus 1.

Algoritmus 1 Výpočet matice $\mathbf{L}$ a $\mathbf{U}$

```
1: U ← A
2: L ← I
3: for k ← 1, ..., n − 1 do
4:     for j ← k + 1, ..., n do
5:         L_{j,k} ← U_{j,k}/U_{k,k}
6:         for r ← k, ..., n do
7:             U_{j,r} ← U_{j,r} − L_{j,k}U_{k,r}
8:         end for
9:     end for
10: end for
```

Pro některé regulární matice ale tento algoritmus nebude fungovat. Například pro matici $\mathbf{A}$ jejíž prvek $A_{0,0} = 0$ algoritmus skončí na řádku pět, protože

bude dělit nulou. Tomu se dá zamezit pomocí nalezení nenulového prvku a následné prohození sloupců nebo řádků matice **A**. Také se může hledat nenulový prvek v celé matici, pak je ale potřeba zaměnit jak řádky tak sloupce. Této metodě se také říká výběr hlavního prvku nebo také pivotace. V LU rozkladu se nejčastěji používá sloupcová pivotace a jako prvek, kterým se bude dělit, se hledá největší číslo v absolutní hodnotě.
Obecně tedy platí, že každou regulární matici **A** lze rozložit na dolní trojúhelníkovou matici **L** a horní trojúhelníkovou matici **U** tak, že

$$\mathbf{PA} = \mathbf{LU},$$

kde **P** je permutační matice, která říká, které sloupce se mezi sebou vyměňují. Zatím jsme popsali jen, jak rozložit matici **A** na **LU**. K řešení soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ použijeme vztah

$$\mathbf{PA}\boldsymbol{x} = \mathbf{P}\boldsymbol{b}.$$

Tedy nejprve nalezneme vektor

$$\boldsymbol{b}' = \mathbf{P}\boldsymbol{b},$$

to znamená, že jsme prohodili prvky ve vektoru $\boldsymbol{b}$ stejně jako v matici **A** a následně řešíme rovnici

$$\mathbf{LU}\boldsymbol{x} = \boldsymbol{b}'.$$

Matice **L** a **U** jsou dolní trojúhelníková, respektive horní trojúhelníková. Pokud řešení rovnice rozdělíme na dva kroky

$$\mathbf{L}\boldsymbol{y} = \boldsymbol{b}'$$

$$\mathbf{U}\boldsymbol{x} = \boldsymbol{y},$$

můžeme soustavu řešit pomocí trojúhelníkové metody, která je uvedená v kapitole 2.1.

2.4 Iterační metody

Iterační metody řešení soustavy lineárních rovnic využívají posloupnosti vektorů, která postupně konverguje k řešení. Výsledný vektor tedy není úplně přesný, ale zaručuje maximální možnou velikost chyby.
Pro soustavu $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ pomocí iterační metody nacházíme vektory $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\infty$ přičemž pro tuto posloupnost vektorů platí

$$\boldsymbol{x} = \lim_{k \to \infty} \boldsymbol{x}_k$$

a $\boldsymbol{x}$ je přesné řešení soustavy.
Protože nemůžeme hledat nekonečnou posloupnost, tak se většinou spokojíme s výsledkem který nám zaručuje maximální chybu ϵ. Výpočet zastavíme jakmile $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \epsilon$, kde $\|.\|$ je norma vektoru. Pro intervalovou aritmetiku se používá několik norem, dále v textu budeme používat normu maximovou

$$\|\boldsymbol{x}\|_\infty = max(|x_1|, x_2|, \ldots, x_n|).$$

2.4.1 Jacobiova metoda

Jacobiova metoda patří mezi základní iterační metody. Ty převádí soustavu

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

na tvar

$$\boldsymbol{x}_{k+1} = \mathbf{B}\boldsymbol{x}_k + \boldsymbol{c},$$

kde $\boldsymbol{x}_0 \in \mathbf{IR}^n$. Dostáváme nekonečnou posloupnost $\{x_k\}_{k=0}^{\infty}$, která v určitých případech konverguje k

$$\boldsymbol{x} = \lim_{k\to\infty} \boldsymbol{x}_k.$$

Postačující podmínkou pro konvergenci je, pokud $\|\mathbf{B}\| < 1$. Pro Jacobiovu metodu platí, že posloupnost konverguje, pokud $\mathbf{A}$ je diagonálně dominantní. Zbývá určit, jak sestrojit matici $\mathbf{B}$ a vektor $\boldsymbol{c}$. Jacobiova metoda rozkládá matici $\mathbf{A}$ na součet

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U},$$

kde $\mathbf{D}$ je diagonální matice, $\mathbf{L}$ je dolní trojúhelníková a $\mathbf{U}$ je horní trojúhelníková matice. Soustavu $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ můžeme upravit do tvaru

$$(\mathbf{D} - \mathbf{L} - \mathbf{U})\boldsymbol{x} = \boldsymbol{b},$$

to můžeme postupně upravovat do tvaru

$$\mathbf{D}\boldsymbol{x} = (\mathbf{L} + \mathbf{U}) + \boldsymbol{b}, \quad \boldsymbol{x} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\boldsymbol{x} + \mathbf{D}^{-1}\boldsymbol{b}.$$

Pokud tedy položíme $\mathbf{B} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ a $\boldsymbol{c} = \mathbf{D}^{-1}\boldsymbol{b}$, pak dostáváme tvar, který byl uveden výše

$$\boldsymbol{x} = \mathbf{B}\boldsymbol{x} + \boldsymbol{c}.$$

Tuto rovnici můžeme také zapsat ve tvaru pro i-tý prvek k-tého kroku

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{i,j}x_j^{k-1}\right) = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j}x_j^{k-1}\right) \quad (2.3)$$

Počáteční vektor $\boldsymbol{x}_0$ můžeme volit libovolně, často se používá nulový nebo jedničkový vektor.

2.4.2 Gaussova-Seidelova metoda

Gaussova-Seidelova metoda vychází z Jacobiovy metody. Pro výpočet x_i^{k+1} se nepoužívá pouze vektor $\boldsymbol{x}^k$, ale kombinuje prvky z vektorů $\boldsymbol{x}^k$ a už vypočítané hodnoty z $\boldsymbol{x}^{k+1}$.

Konkrétně pro i-tý prvek x_i^{k+1} se používají hodnoty $x_{1,\dots,i-1}^{k+1}$ a $x_{i+1,\dots,n}^{k}$. Hodnoty vektoru tedy budeme počítat ve tvaru

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^k - \sum_{j=i+1}^{n} a_{i,j}x_j^{k-1}\right).$$

Gauss-Seidelova metoda konverguje v případě, že matice $\mathbf{A}$ je diagonálně dominantní 1.6 nebo je symetrická a zároveň pozitivně definitní 1.7. Výhodou této metody je, že hodnoty vektoru x^{k+1} a x^k můžeme uložit jenom do jednoho vektoru. Nevýhodou je, že složky vektoru musíme počítat sekvenčně v pořadí od 1 do n, tedy nemůžeme použít paralelizaci.

2.4.3 Metoda sdružených gradientů

Metoda sdružených gradientů je iterační metoda, která konverguje v případě že pro rovnici

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

platí, že matice $\mathbf{A}$ je symetrická a pozitivně definitní definice v kapitole 1.7, a tedy k matici $\mathbf{A}$ existuje inverzní matice $\mathbf{A}^{-1}$ a soustava má právě jedno řešení, které označíme $\boldsymbol{x}_\infty$

$$\boldsymbol{x}_\infty = \mathbf{A}^{-1}\boldsymbol{b}.$$

Vezmeme-li $\boldsymbol{x}$ jako řešení v i-té iteraci, pak chybu řešení označíme symbolem $\boldsymbol{e}$ a platí

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_\infty$$

Dále budeme pomocí symbolu $\boldsymbol{r}$ značit reziduum

$$\boldsymbol{r} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x}.$$

Protože platí

$$\mathbf{A}\boldsymbol{e} = \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}_\infty) = \mathbf{A}\boldsymbol{x} - \mathbf{A}\boldsymbol{x}_\infty = \mathbf{A}\boldsymbol{x} - \boldsymbol{b} = -\boldsymbol{r},$$

pak můžeme také psát

$$\boldsymbol{e} = -\mathbf{A}^{-1}\boldsymbol{r}, \quad \boldsymbol{r} = -\mathbf{A}\boldsymbol{e}.$$

Definujme funkci

$$\varphi(x) = \frac{1}{2}\boldsymbol{e}^T\mathbf{A}\boldsymbol{e}.$$

Pokud $\boldsymbol{x} = \boldsymbol{x}_\infty$ pak $\boldsymbol{e} = 0$, proto $\varphi(x_\infty) = 0$. Funkci $\varphi(x)$ můžeme dále upravit

$$\varphi(x) = \frac{1}{2}\boldsymbol{e}^T\mathbf{A}\boldsymbol{e} = \frac{1}{2}\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b} + \frac{1}{2}\boldsymbol{x}_\infty^T\mathbf{A}\boldsymbol{x}_\infty.$$

Protože výraz $\frac{1}{2}\boldsymbol{x}_\infty^T \mathbf{A}\boldsymbol{x}_\infty$ je roven nějaké konstantě, můžeme definovat funkci $J(\boldsymbol{x})$, která se bude lišit právě o tu konstantu, ale bude platit $\forall \boldsymbol{x}\ J(\boldsymbol{x}_\infty) < J(\boldsymbol{x})$

$$J(x) = \frac{1}{2}\boldsymbol{x}^T \mathbf{A}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b}.$$

Metoda sdružených gradientů spočívá v hledání posloupnosti vektorů $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ tak, aby platilo

$$J(\boldsymbol{x}_{k+1}) < J(\boldsymbol{x}_k).$$

Vektor $\boldsymbol{x}_{k+1}$ získáme přičtením jiného vektoru tedy

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k.$$

Dále si definujeme funkci

$$g(\alpha) = J(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k),$$

to můžeme upravit na tvar

$$g(\alpha) = J(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) = J(\boldsymbol{x}_k) - \alpha \boldsymbol{p}_k^T \boldsymbol{r}_k + \frac{1}{2}\alpha^2 \boldsymbol{p}_k^T \mathbf{A}\boldsymbol{p}_k.$$

Pro řešení soustavy hledáme bod, pro který platí, že $g(\alpha)$ je minimální. Protože funkce $J(\boldsymbol{x})$ je parabola, tak i funkce $g(\alpha)$ je parabola. Proto pro minimum platí, že je v bodě kde $g'(\alpha) = 0$ a

$$g'(\alpha) = -\boldsymbol{p}_k^T \boldsymbol{r}_k + \alpha \boldsymbol{p}_k^T \mathbf{A}\boldsymbol{p}_k.$$

2.5 Předpodmínění

Předpodmínění matice se používá v intervalové aritmetice ke zmenšení šířky výsledných intervalů. Při použití předpodmínění pomocí matice středních hodnot pro matici $\mathbf{P}$ platí

$$\mathbf{P} = mid(\mathbf{A})^{-1}\mathbf{A}.$$

Při řešení soustavu vynásobíme maticí $mid(\mathbf{A})^{-1}$

$$mid(\mathbf{A})^{-1}\mathbf{A}\boldsymbol{x} = mid(\mathbf{A})^{-1}\boldsymbol{b}$$

a následně řešíme

$$\mathbf{P}\boldsymbol{x} = \boldsymbol{b}'.$$

Matice $\mathbf{P}$ se blíží jednotkové matici. Předpodmínění můžeme použít v případě, že $\mathbf{A}$ je silně regulární, to znamená $mid(\mathbf{A})^{-1}\mathbf{A}$ je regulární.

Kapitola 3

Realizace

Pro implementaci řešičů lineárních rovnic jsem použil programovací jazyk C++ z důvodů vysoké výkonnosti a možnosti kontroly využití paměti. Není tedy nutné se spoléhat na různé garbage collectory nebo další služby, které nemáme pod kontrolou a zpomalují běh programu.
Existuje několik knihoven pro architekturu x86, které implementují intervalovou aritmetiku [8]. Mezi ně patří knihovna MPFI, která rozšiřuje knihovnu MPFR. Tyto knihovny umožňují pracovat s volitelnou přesností čísel s plovoucí čárkou. Kromě základních operací jako sčítání, odčítání, násobení a dělení také implementují funkce pro intervalovou aritmetiku jako goniometrické funkce, logaritmus a další viz [9]. Další rozšířenou knihovnou je knihovna Boost Interval Arithmetic Library [10], která implementuje třídu Interval pomocí šablony, tedy můžeme pro interval zvolit námi požadovaný datový typ. Mezi další patří knihovna PROFIL/BIAS [11], která je dále popsána, a knihovna Gaol [12], která implementuje základní operace pro několik datových typů s plovoucí čárkou a různou přesností.

3.1 PROFIL/BIAS

PROFIL/BIAS (Programmer's Runtime Optimized Fast Interval Library / Basic Interval Arithmetic Subroutines) je knihovna určená pro jazyk C++, která umožňuje provádění operací s intervaly, ale zároveň i operace s intervalovými maticemi a intervalovými vektory.
Vývoj knihovny byl inspirován již používanou knihovnou BLAS (Basic Linear Algebra Subprograms) k poskytnutí rozhraní pro počítání s maticemi a vektory. Hlavním cílem bylo vytvořit knihovnu, která bude výkonná na různých platformách.
Knihovna implementuje intervaly, intervalové vektory a intervalové matice jako samostatné třídy. PROFIL používá pro intervaly datové typy INT (celé číslo) a REAL (číslo s plovoucí řádovou čárkou). Třídy v této knihovně mají přetížené operátory sčítání, násobení, přiřazení a další operátory používané

při práci s intervaly, ale nemají žádné metody. Pro získání vlastností intervalových vektorů, respektive intervalových matic slouží definované funkce. Například pro nastavení počátečních hodnot matice, je nutno zavolat funkci $Initialize(\mathbf{IA}, v)$, kde **IA** je matice, která má být nastavena, a v je interval. Všechny metody jsou podrobně popsány v [11].

3.2 Boost Interval Arithmetic Library

Boost je sada knihoven určených pro programovací jazyk C++. Obsahuje knihovnu pro práci s vlákny, knihovnu pro práci s datovými strukturami a také knihovnu pro intervalovou aritmetiku. Ta byla inspirována předchozí prací Jense Maurera, ve které pokračovali Jeremy Sick a Maarten Keijzer, a postupně se k nim přidali další.
Knihovna definuje třídu *Interval*, která pro určení používaného datového typu používá šablonu. Tato třída zaručuje správné chování pro datové typy *float*, *double* a *long double*, tak jak jsou definovány v standardu IEEE-754. Dále je možno pomocí takzvaných *policies* nastavit chování intervalu při různých operacích jako zaokrouhlování, porovnávání intervalů a dalších.
Při práci s knihovnou můžeme použít přetížené operátory, které definuje třída *Interval*, na rozdíl od knihovny PROFIL/BIAS třída definuje veřejné metody, které můžeme použít, jako například výpočet šířky intervalu hodnoty nebo absolutní hodnoty. Práce s knihovnou je podrobně popsána v [10].

3.3 Implementace metod pomocí knihoven

Protože tato práce má za cíl implementovat jednotlivé metody pro řešení lineárních soustav rovnic v intervalové aritmetice, je použití knihovny jako MPFI zbytečné, většinu jejích funkcí bychom nevyužili.
Pro následnou realizaci jsem vybral knihovnu PROFIL/BIAS, která je zaměřená na počítání s intervalovými maticemi a vektory a měla by být výkonná [8].Jako druhou knihovnu jsem vybral Boost Interval Arithmetic Library, která naopak přímo neposkytuje rozhraní pro práci s maticemi nebo vektory.
Při implementaci metod pro řešení soustav intervalových rovnic pomocí knihovny PROFIL/BIAS využíváme tříd, které umožňují práci s intervalovými maticemi (IntervalMatrix) a intervalovými vektory (IntervalVector). Také využíváme všechny dostupné metody, které tato knihovna poskytuje například, metody pro výměnu sloupců, řádků. Pro sčítání, odčítání, násobení a dělení využíváme přetížené operátory.
Knihovna Boost Interval Arithmetic library nabízí pouze třídu pro práci s intervaly. Implementoval jsem tedy vlastní třídy pro práci s intervalovými maticemi a vektory, při počítání s intervaly využíváme stejně jako u knihovny PROFIL/BIAS přetížené operátory. Pro přístup k dolní a horní mezi inter-

valu používáme metody lower, respektive upper. Knihovna nedefinuje metodu mignitude, tu jsme nuceni implementovat sami.

3.4 Struktura řešení

Řešení obsahuje tři programy. První generuje náhodné soustavy rovnic požadovaných vlastností za použití knihovny PROFIL/BIAS, které poskytuje metody pro generování náhodných čísel a matic, které mají námi požadované vlastnosti. Další dva programy mají totožnou funkčnost, ale zatímco první používá knihovnu PROFIL/BIAS, druhý používá knihovnu Boost.
Spustitelné soubory jsou umístěny v adresáři bin, programy určené pro řešení soustavy intervalových rovnic se spouštějí se třemi argumenty. První odpovídá použité metodě, druhý použité heuristice při výběru pivotu a třetí argument určuje výpis programu. Argumenty jsou popsány v tabulce 3.1 a 3.2.
Řešiče nejdříve na standardním vstupu přijmou matici $\mathbf{A}$ v požadovaném tvaru, poté vektor $\boldsymbol{x}$ a vektor $\boldsymbol{b}$, který odpovídá $\boldsymbol{b} = \mathbf{A}\boldsymbol{x}$. Vektor $\boldsymbol{x}$ je použit k vyhodnocení správnosti řešení. Následně provedou jednotlivé metody popsané výše a na standardní výstup vypíšou požadované hodnoty jako například průměrná šířka intervalů.

Tabulka 3.1: První argument pro spuštění řešiče

1. argument	Metoda
A	Gaussova eliminace
B	Gaussova eliminace s řádkovou pivotací
C	Gaussova eliminace se sloupcovou pivotací
D	Gaussova eliminace s celkovou pivotací
E	LU rozklad
F	Jacobiova metoda
G	Gauss-Seidelova metoda
H	Metoda sdružených gradientů
-	Další metody jsou použity po předpodmínění
I	Gaussova eliminace
J	Gaussova eliminace s řádkovou pivotací
K	Gaussova eliminace se sloupcovou pivotací
L	Gaussova eliminace s celkovou pivotací
M	LU rozklad
M	Jacobiova metoda
O	Gauss-Seidelova metoda
P	Metoda sdružených gradientů

Tabulka 3.2: Druhý a třetí argument pro spuštění řešiče

1. argument	Heuristika	2. argument	Vvýstup
A	Abs	A	Řešení
B	Mig	B	Průměrná šířka
C	Mid	C	Doba běhu

3.5 Implementace metod

3.5.1 Gaussova eliminační metoda

Gaussova eliminační metoda je implementována pomocí dvojrozměrného pole intervalů, které reprezentuje matici **A**, a pole intervalů, které reprezentuje vektor ***b***. Postupujeme stejně, jak je popsáno v kapitole 2.2. Pro zlepšení výkonnosti nejdříve vypočteme násobek, kterým budeme odečítat i-tý řádek, a uložíme ho do proměnné *ratio*. Následně tímto poměrem násobíme jednotlivé prvky v řádku a odečítáme je od řádku, kde vytváříme nulu.
Zpětný chod provádíme podle vzorce (2.2).

Algoritmus 2 Gaussova eliminační metoda

```
1:  procedure GAUSS(A, b)
2:      for i ← 1, ..., n − 1 do
3:          for j ← i + 1, ..., n do
4:              ratio ← A_{j,i}/A_{i,i}
5:              for k ← i + 1, ..., n do
6:                  A_{j,k} ← A_{j,k} − ratio ∗ A_{i,k}
7:              end for
8:              b_j ← b_j − ratio ∗ b_i
9:          end for
10:     end for
11:     for i ← n, ..., 1 do
12:         for j ← n, ..., i + 1 do
13:             b_i ← b_i − b_j ∗ A_{i,j}
14:         end for
15:         b_i ← b_i/A_{i,i}
16:     end for
17: end procedure
```

Proběhnutí vnitřního cyklu odpovídá $6(n-i-1)$ operací s číslem s plovoucí čárkou. Výpočet *ratio* k tomu připočteme a dostáváme $6(n-i-1)+16$ operací. Pro zpětný chod připočteme $n(3n+1)$ operací, celkový počet tedy bude

$$\sum_{i=1}^{n-1} \sum_{j=k+1}^{n} (6(n-i-1)+16) + n(3n+1) = 2n^3 + 5n^2 - 3n. \tag{3.1}$$

Pro zlepšení výsledků v intervalové aritmetice můžeme použít řádkovou, sloupcovou nebo celkovou pivotaci. Pivot vybíráme na základě zvolené heuristiky. Používáme absolutní hodnotu, střední hodnotu nebo mignitude. Hledání pivotu provádíme před výpočtem hodnoty *ratio*, tedy před čtvrtý řádek vložíme pro řádkovou pivotaci

Algoritmus 3 Řádková Pivotace

```
max ← A_{i,i}
index ← i
for j ← i, ..., n do
    if mid(A_{j,i} > max) then
        max ← mid(A_{j,i})
        index ← i
    end if
end for
```

nebo pro sloupcovou pivotaci

Algoritmus 4 Sloupcová Pivotace

```
max ← A_{i,i}
index ← i
for j ← i, ..., n do
    if mid(A_{i,j} > max) then
        max ← mid(A_{i,j})
        index ← i
    end if
end for
swap(A_{i,i}, A_{i,j})
swap(b_i, b_j)
```

U sloupcové pivotace také musíme zaměnit hodnoty ve vektoru $\boldsymbol{b}$.
Spojením řádkové a sloupcové pivotace dostáváme pivotaci celkovou. Při pivotaci dostáváme sice lepší výsledky, ale také dochází k zvětšení počtu operací. Při vytvářením nul v i-tém sloupci je potřeba porovnat u sloupcové a řádkové pivotace $n - i$ prvků, u celkové pivotace to je porovnání $(n - i)^2$ prvků.

3.5.2 LU rozklad

Metodu LU pro lepší výkonnost rozkládáme do dvou funkcí. První funkce rozloží matici **A** na dolní trojúhelníkovou a horní trojúhelníkovou matici a vytvoří permutační vektor. Druhá funkce na základě **LU** matice a permutačního vektoru vypočte řešení pro vektor $\boldsymbol{b}$.
To má výhodu při řešení lineárních problémů, které jsou ve tvaru

$\mathbf{A}\boldsymbol{x}_1 = \boldsymbol{b}_1, \mathbf{A}\boldsymbol{x}_2 = \boldsymbol{b}_2 \ldots \mathbf{A}\boldsymbol{x}_n = \boldsymbol{b}_n$. Při těchto úlohách nám stačí jednou rozložit matici $\mathbf{A}$ na $\mathbf{LU}$ a následně řešíme $\mathbf{LU}\boldsymbol{x}_1 = \boldsymbol{b}_1, \mathbf{LU}\boldsymbol{x}_2 = \boldsymbol{b}_2 \ldots \mathbf{LU}\boldsymbol{x}_n = \boldsymbol{b}_n$. Například tímto způsobem lze vypočítat inverzní matici, pro kterou platí $\mathbf{A}\boldsymbol{x}_i = \boldsymbol{e}_i$, kde $\boldsymbol{e}_i$ je jednotkový vektor s 1 na i-tém místě a $\boldsymbol{x}_i$ je sloupec inverzní matice na i-tém místě.
Při implementaci využíváme toho, že matice $\mathbf{L}$ má na diagonále všechny prvky rovny jedné. Díky tomu pro uložení matic $\mathbf{L}$ a $\mathbf{U}$ stačí jedno dvojrozměrné pole o velikosti matice $\mathbf{A}$, kde v dolní polovině jsou uloženy prvky matice $\mathbf{L}$ a v horní polovině včetně diagonály jsou uloženy prvky matice $\mathbf{U}$. Také neukládáme celou permutační matici, protože obsahuje jenom informaci o tom, kam se který sloupec posunul. Tuto informaci ukládáme do permutačního vektoru.

Algoritmus 5 LU rozklad

```
1:  function DECOMPOSITIONLU(A,p)
2:      LU ← A
3:      p                                          ▷ permutační vektor
4:      for j ← 1, ..., n − 1 do
5:          ma ← 0.0
6:          mi ← −1

7:          for i ← j, ..., n do
8:              if mid(LU_{i,j}) > ma then          ▷ Pivotace
9:                  ma ← abs(LU_{i,j})
10:                 mi ← i
11:             end if
12:             p_j ← mi

13:             if mi ≠ j then                      ▷ Prohození řádků
14:                 swap(LU_j, LU_mi)
15:             end if

16:             for i ← j + 1, ..., n do
17:                 LU_{i,j} ← −1 ∗ LU_{i,j}/LU_{j,j}
18:             end for
19:             for i ← j + 1, ..., n do
20:                 for k ← j + 1, ..., n do
21:                     LU_{i,k} ← LU_{i,k} + LU_{i,j} ∗ LU_{j,k}
22:                 end for
23:             end for
24:         end for
25:     end for
26:     return LU, p
27: end function
```

Algoritmus 6 LU řešení

```
function SOLVELU(LU,b,p)
    for i ← 1, ..., n − 1 do        ▷ Prohození vektoru b podle permutačního
vektoru p
        temp ← b_i
        b_i ← b_{p[i]}
        b_{p[i]} ← temp
    end for
    for i ← 1, ..., n − 1 do
        for j ← 1, ..., n do
            b_j ← b_j + LU_{j,i} * b_i
        end for
    end for
    for i ← n, ..., 1 do
        b_i ← b_i/LU_{i,i}
        for j ← 1, ..., i do
            b_j ← b_j − LU_{j,i} * b_i
        end for
    end for
    return b
end function
```

Řešení LU metody implementujeme pomocí algoritmu 6. Metoda LU rozkladu je stejně složitá jako Gaussova eliminace s pivotací, tedy potřebný počet operací je $2n^3 + 5n^2 - 3n$ pro pivotaci přičteme $1/2(n^2 + n - 2)$ operací.

3.5.3 Jacobiova metoda

Jacobiova metoda je implementována jako cyklus, který počítá vektor x^{k+1} pomocí vzorce 2.3. Pro lepší výkonnost je cyklus, který počítá i-tý prvek vektoru x^{k+1}, rozdělen do dvou cyklů a tím odpadá nutnost použít podmínku v tomto cyklu. Postup řešení pomocí Jacobiovy metody je popsán v algoritmu 7.
Pro výpočet nového prvku vektoru je potřeba $4 * (n-1) + 10$ operací, výpočet celého vektoru zabere $2n(2n + 3)$ operací. Celková náročnost algoritmu závisí na počtu provedených iterací

$$Počet\ iterací * (4n^2 + 6n).$$

Algoritmus 7 Jacobiova metoda

```
1: function jacobiMethod(A,b,max)
2:     xOld
3:     xNew
4:     e ← [1.0, 1.0]
5:     for i ← 1, ..., max do
6:         for j ← 1, ..., n do
7:             sum ← [0.0, 0.0]
8:             for k ← 1, ..., j − 1 do
9:                 sum ← A_{j,k} * xOld_k
10:            end for
11:            for k ← j + 1, ..., n do
12:                sum ← A_{j,k} * xOld_k
13:            end for
14:            xNew_j ← (e/A_{j,j}) * (b_j − sum)
15:        end for
16:    end for
17: end function
```

3.5.4 Gaussova-Seidelova metoda

Implementace Gauss-Seidelovy metody je velmi podobná Jacobiově metodě. Na rozdíl od ní ale nemusíme v paměti uchovávat vektor $xOld$ a stačí nám používat pouze jeden výsledný vektor, jehož první prvky budou tvořeny $x_1^{k+1}, x_2^{k+1}, \ldots, x_{i-1}^{k+1}$ a druhá část bude obsahovat $x_{i+1}^k, x_{i+2}^k, \ldots, x_n^k$. Paměťová náročnost tedy bude dvakrát menší. Pseudokód popisující Gauss-Seidelovu metodu je v 8.

Počet provedených operací je stejný jako u Jacobiovy metody tedy *Počet iterací*$*$ $(4n^2 + 6n)$.

3.5.5 Metoda sdružených gradientů

Metoda sdružených gradientů je implementována pomocí algoritmu 9, kde p je vektor posunu, α je velikost vektoru posunu. Směr vektoru posunu volíme jako největší spád funkce J v daném bodě, tedy $p = r$. Proměnná q nám umožňuje zmenšit počet výsledných operací. Při výpočtu α a r nemusíme násobit matici $\mathbf{A}$ s vektorem $\mathbf{p}$, tedy ušetříme jedno násobení matice krát vektor, což činí $8n^2$ operací s čísly s plovoucí čárkou. Pro jednu iteraci je potřeba $2(2n^2 + 5n + 4)$ operací s plovoucí čárkou, tedy celkový počet operací *Počet iterací* $* (4n^2 + 10n + 8)$.

Při implementaci pomocí pseudokódu 9 se interval $\boldsymbol{p}^T\boldsymbol{q}$, kterým se následně dělí, rozšiřuje takovým způsobem, že obsahuje nulu. Protože dělení intervalem, který obsahuje nulu, není definováno, algoritmus nemůže pokračovat a

Algoritmus 8 Gaussova-Seidelova metoda

```
1: function gaussSeidelMethod(A,b,max)
2:     x
3:     e ← [1.0, 1.0]
4:     for i ← 1, ..., max do
5:         for j ← 1, ..., n do
6:             sum ← [0.0, 0.0]
7:             for k ← 1, ..., j − 1 do
8:                 sum ← A_{j,k} * x_k
9:             end for
10:            for k ← j + 1, ..., n do
11:                sum ← A_{j,k} * x_k
12:            end for
13:            x_j ← (e/A_{j,j}) * (b_j − sum)
14:        end for
15:    end for
16: end function
```

Algoritmus 9 Metoda sdružených gradientů

```
1: function cgMethod(A,b,max)
2:     r ← b − Ax
3:     p ← r
4:     for i ← 1, ..., max do
5:         q ← Ap
6:         α ← p^T r / p^T q
7:         x ← x + αp
8:         r ← r − αq
9:         p ← r
10:    end for
11: end function
```

soustavu rovnic nevyřeší. Při použití tohoto algoritmu na soustavy uvedené v kapitole 4 se ani jednou nepodařilo soustavu vyřešit.
Pro řešení soustavy lineárních intervalových rovnic můžeme použít metody, které jsou založeny na metodě sdružených gradientů například [13].

Kapitola 4

Testování

Při testování porovnáváme výslednou šířku jednotlivých metod, vliv heuristik na výslednou šířku a výkonnost knihoven PROFIL/BIAS a Boost.
Pro testování používáme generátor, který generuje matici $\mathbf{A}$, která odpovídá našim požadavkům. Následně vytvoří vektor $\boldsymbol{x}$, jehož šířky intervalů jsou rovny nule. Dále vynásobením matice $\mathbf{A}$ a vektoru $\boldsymbol{x}$ získáme vektor $\boldsymbol{b}$, jehož šířky intervalů jsou větší než nula. S vektorem $\boldsymbol{x}$ se při řešení soustavy vůbec nepočítá, slouží pouze ke kontrole nalezeného řešení.
Výsledky provedeného měření jsou uvedeny v tabulkách, kde v první sloupci je uveden počet rovnic soustavy, v dalších jsou výsledky jednotlivých metod.

4.1 Porovnání šířky výsledných intervalů

První měření porovnává metody bez použití předpodmínění. Pro měření byla použita soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\mathbf{A}$ je pozitivně definitní matice a diagonálně dominantní. V tomto případě všechny iterační metody konvergují ke správnému výsledku. Šířka intervalů v matici $\mathbf{A}$ je z intervalu $[1, 10]$ a střední hodnota prvků matice $\mathbf{A}$ je z intervalu$[10, 200]$. Výsledky tohoto měření jsou uvedeny v tabulce 4.1.
Jak z tabulky vyplývá, nejlepší výsledky dostáváme při použití Gaussovy eliminace, pivot vybíráme podle absolutní hodnoty. Pivotace při deseti měřeních osmkrát výslednou šířku zmenšila, ale také dvakrát zvětšila. Výsledné šířky při použití Jacobiovy a Gauss-Seidelovy metody jsou 200 krát až 500 krát horší než u Gaussovy eliminace, což tyto metody činí bez použití předpodmínění nepoužitelné. Navíc šířka intervalů se zvětšuje s rostoucím počtem provedených iterací. Při výpočtu pomocí metody sdružených gradientů se při výpočtu velikosti vektoru posunu dělilo intervalem, který obsahoval nulu, takže se pomocí této metody nepovedlo soustavu vyřešit.
Při druhém porovnání testujeme metody s použitím předpodmínění. Při použití přímých metod je zlepšení oproti použití bez předpodmínění zhruba dvoj-

Tabulka 4.1: Porovnání výsledné šířky u metod bez použití předpodmínění

Řád	GEM	GEM s pivotací	Jacobi	Gauss-Seidel
10	1.701	1.68	368.5	326.7
20	1.419	1.437	2543	2102
30	2.11	2.107	6124	5145
40	1.649	1.612	106.1	97.29
50	1.919	1.891	516	454.6
60	1.803	1.768	1024	880.9
70	1.709	1.718	653.3	568.8
80	1.767	1.722	1436	1220
90	1.901	1.859	961.3	829.2
100	1.845	1.833	1008	862.7

násobné, vliv pivotace je minimální. To je způsobeno tím, že po předpodmínění je matice **A** podobná jednotkové matici. Co se týče iteračních metod, jejich výsledné šířky intervalů jsou totožné, což je způsobeno tím, že Jacobiova a Gauss-Seidelova metoda jsou velice podobné. V porovnání iteračních metod s Gaussovou eliminací jsou výsledné šířky prakticky shodné. Metoda sdružených gradientů ani po použití předpodmínění neřeší soustavu, protože se snaží dělit intervalem, který obsahuje nulu. Výsledky měření jsou uvedené v tabulce 4.2.

Tabulka 4.2: Porovnání výsledné šířky u metod s použití předpodmínění

Řád	GEM	GEM s pivotací	Jacobi	Gauss-Seidel
10	1.005 41	1.0054	1.005 51	1.005 51
20	0.945 008	0.945 004	0.945 053	0.945 053
30	1.006 88	1.006 88	1.0069	1.0069
40	0.928 163	0.928 164	0.928 181	0.928 181
50	0.920 19	0.920 191	0.920 205	0.920 205
60	1.041 21	1.041 21	1.041 23	1.041 23
70	0.889 401	0.889 401	0.889 411	0.889 411
80	0.785 404	0.785 404	0.785 411	0.785 411
90	0.892 276	0.892 276	0.892 284	0.892 284
100	0.830 688	0.830 688	0.830 694	0.830 694

U dalšího porovnání budeme využívat předpodmínění a dále budeme používat matici **A**, která je pozitivně definitní a diagonálně dominantní. Šířka intervalů v matici **A** je z intervalu $[1, 5]$ a budeme měnit střední hodnotu matice. Výsledky jsou postupně uvedeny v tabulkách 4.3, 4.4 a 4.5.

Tabulka 4.3: Porovnání výsledné šířky střední hodnota $\mathbf{A} \in [1, 10]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	10.9096	10.8982	10.9486	10.9479
20	12.7477	12.7418	12.7893	12.7864
30	15.4567	15.4589	15.4928	15.4882
40	13.7821	13.7834	13.8023	13.7979
50	15.6079	15.6066	15.624	15.6175
60	15.8147	15.8147	15.8265	15.8201
70	15.8487	15.8468	15.8558	15.8492
80	15.8866	15.8866	15.8924	15.8862
90	16.0548	16.0554	16.0582	16.0517
100	17.7549	17.7545	17.7559	17.7481

Tabulka 4.4: Porovnání výsledné šířky střední hodnota $\mathbf{A} \in [1, 100]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	1.145 73	1.145 72	1.145 96	1.145 96
20	1.323 27	1.323 27	1.323 37	1.323 37
30	1.065 58	1.065 58	1.065 63	1.065 63
40	1.212 03	1.212 03	1.212 07	1.212 07
50	1.033 59	1.033 59	1.033 62	1.033 62
60	1.232 01	1.232 01	1.232 03	1.232 03
70	1.129 55	1.129 55	1.129 57	1.129 57
80	1.209 67	1.209 67	1.209 69	1.209 69
90	1.127 87	1.127 87	1.127 88	1.127 88
100	1.180 62	1.180 62	1.180 64	1.180 64

Tabulka 4.5: Porovnání výsledné šířky střední hodnota $\mathbf{A} \in [1, 200]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	0.638 454	0.638 457	0.638 481	0.638 481
20	0.594 436	0.594 436	0.594 445	0.594 445
30	0.593 609	0.593 609	0.593 615	0.593 615
40	0.574 868	0.574 868	0.574 872	0.574 872
50	0.532 565	0.532 565	0.532 568	0.532 568
60	0.565 829	0.565 829	0.565 831	0.565 831
70	0.546 316	0.546 316	0.546 318	0.546 318
80	0.584 713	0.584 713	0.584 715	0.584 715
90	0.545 499	0.545 499	0.5455	0.5455
100	0.544 831	0.544 831	0.544 832	0.544 832

Z výsledků měření je vidět, že šířka výsledných intervalů při stejné šířce intervalů matice **A** závisí na středních hodnotách matice **A**. Čím větší je maximální možná střední hodnota, tím menší je šířka výsledného intervalu.
Při další sérii měření ponecháme stejný interval, ze kterého bude střední hodnota $mid(\mathbf{A}) \in [1, 200]$, a budeme postupně měnit možnou šířku intervalu matice **A**. Výsledky měření jsou uvedeny v tabulkách 4.6, 4.7 a 4.8

Tabulka 4.6: Porovnání výsledné šířky $wid(\mathbf{A}) \in [1, 5]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	0.638 454	0.638 457	0.638 481	0.638 481
20	0.594 436	0.594 436	0.594 445	0.594 445
30	0.593 609	0.593 609	0.593 615	0.593 615
40	0.574 868	0.574 868	0.574 872	0.574 872
50	0.532 565	0.532 565	0.532 568	0.532 568
60	0.565 829	0.565 829	0.565 831	0.565 831
70	0.546 316	0.546 316	0.546 318	0.546 318
80	0.584 713	0.584 713	0.584 715	0.584 715
90	0.545 499	0.545 499	0.5455	0.5455
100	0.544 831	0.544 831	0.544 832	0.544 832

Tabulka 4.7: Porovnání výsledné šířky $wid(\mathbf{A}) \in [1, 50]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	8.389 82	8.389 03	8.421 34	8.421 16
20	7.471 04	7.470 91	7.481 22	7.481 03
30	7.130 12	7.129 72	7.1358	7.135 73
40	5.451 47	5.451 25	5.454 83	5.454 75
50	5.703 33	5.703 22	5.705 95	5.705 88
60	5.683 59	5.683 58	5.685 84	5.685 78
70	5.413 66	5.413 63	5.4154	5.415 34
80	6.178 97	6.179	6.180 84	6.180 77
90	5.637 68	5.637 65	5.639 18	5.639 11
100	6.017 24	6.017 28	6.018 78	6.018 71

Tabulka 4.8: Porovnání výsledné šířky $wid(\mathbf{A}) \in [1, 100]$

Řád	**GEM**	**GEM s pivotací**	**Jacobi**	**Gauss-Seidel**
10	26.0038	26.0258	26.0805	26.0512
20	19.553	19.5349	19.6161	19.6067
30	16.7252	16.7237	16.7537	16.7476
40	16.518	16.5157	16.5341	16.5277
50	18.0193	18.0187	18.037	18.0293
60	16.749	16.7496	16.7577	16.7502
70	15.5918	15.5913	15.5965	15.5892
80	18.4009	18.3999	18.403	18.3949
90	15.652	15.6508	15.6538	15.6474
100	15.2124	15.213	15.212	15.2055

Z těchto měření je vidět, že výsledná šířka intervalů ani tak nezáleží na počtu rovnic, ale na poměru středních hodnot matice **A** a šířkách jejích intervalů. Při použití předpodmínění jsou rozdíly ve výsledných šířkách intervalů mezi přímými metodami a iteračními metodami minimální.

4.2 Porovnání vlivu heuristiky pro výběr pivotu

Pivot můžeme volit několika způsoby. Jako největší absolutní hodnotu, největší mignitude nebo největší střední hodnotu. Pro porovnání jednotlivých výběrů používáme matici **A**, jejíž prvky mají střední hodnotu z intervalu $[10, 200]$ a šířka je z intervalu $[1, 50]$. Na matici soustavy nepoužijeme předpodmínění.

Tabulka 4.9: Porovnání výsledné šířky u jednotlivých heuristik

Řád	**Abs**	**Mig**	**Mid**
10	12.4585	12.6176	12.4585
20	11.5296	11.5299	11.4848
30	11.6093	11.596	11.6048
40	10.7541	10.747	10.7318
50	10.8122	10.7449	10.7804
60	10.9577	10.9505	10.968
70	10.8202	10.8347	10.8244
80	11.7316	11.7295	11.7243
90	10.7017	10.6996	10.715
100	11.1727	11.1668	11.17
Průměr	11.254	11.261	11.246

Z měření nejlépe vychází použití střední hodnoty pro výběr pivotu jako nejlepší, druhý nejlepší výsledek má výběr pomocí absolutní hodnoty a nejhůře

dopadl výběr pomocí mignitude. Nicméně rozdíly jsou velmi malé a pokud použijeme předpodmínění rozdíly, budou ještě menší.

4.3 Porovnání jednotlivých metod

4.3.1 Gaussova eliminace

Při použití Gaussovy eliminace s předpodmíněním dostáváme výsledky zhruba dvakrát lepší než bez předpodmíněním. Pokud zachováme šířku intervalů matice **A** a měníme interval, ze kterého jsou prvky matice **A**, pak se zvětšující se střední hodnotou se zmenšuje výsledná šířka intervalu.
V případě, že střední hodnota matice **A** zůstává stejná a měníme šířku intervalů, se zvětšující se šířkou intervalů matice **A** se také zvětšuje šířka výsledného vektoru. Vliv pivotace při použití na zvolených soustavách je minimální.

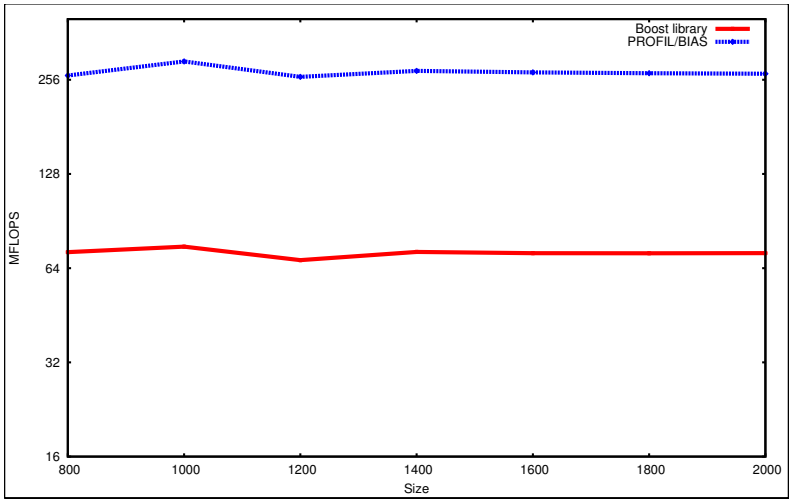
4.3.2 Jacobiova a Gauss-Seidelova metoda

Při použití Jacobiovy a Gauss-Seidelovy metody bez předpodmínění je výsledná šířka vektoru 200 krát až 500 krát horší než u Gaussovy eliminace bez předpodmínění. Tyto metody jsou tedy prakticky nepoužitelné bez předpodmínění. Navíc s větším počtem iterací se také zvětšuje výsledná šířka intervalů.
Pokud před samotným řešením pomocí těchto metod aplikujeme na matici **A** metodu předpodmínění, dostáváme poté podobné výsledky jako s použitím Gaussovy eliminace s předpodmíněním.
Pro soustavy, které jsou příliš velké pro řešení pomocí přímých metod, nebo pro řídké soustavy můžeme tyto metody s předpodmíněním použít.

4.4 Porovnání výkonnosti knihoven

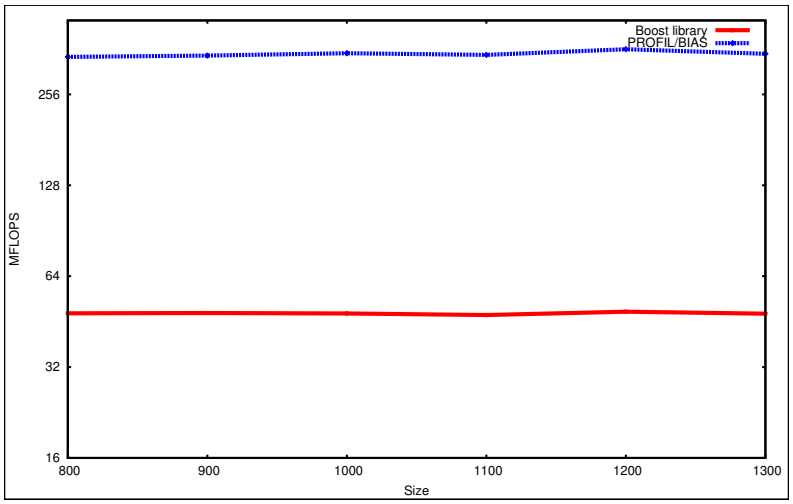
Pro měření výkonnosti knihoven byl použit výpočetní svazek STAR2 a úloha byla spouštěna na uzlu gpu-02, jehož konfigurace je dva procesory Intel Xeon 2620 o taktovací frekvenci 2.1 Ghz a 32 GB kapacita operační paměti. Program byl zkompilován pomocí kompilátoru g++ s použitým parametrem -O2 pro optimalizaci. Díky tomuto způsobu měření máme zajištěno, že úloha není přerušována operačním systémem nebo jiným programem a můžeme využít po celou dobu běhu maximální kapacitu operační paměti.
Pro měření byla použita metoda Gaussovy eliminace jako zástupce přímé metody a Gauss-Seidelova metoda jako zástupce iterační metody. Protože ani jedna metoda není závislá na vstupních datech, známe počet provedených operací a výkonnost knihovny měříme v počtu provedených operací v plovoucí řádce za vteřinu (FLOPS). Jako vstupní soubor byly použity soustavy vygenerované připraveným generátorem.
Při měření výkonnosti Gaussovy eliminace byly použity soustavy rovnic o

velikost $800, 1000, \ldots, 2000$ rovnic. Nejdelší výpočet při 2000 generátorech s knihovnou Boost trval jedenáct minut a deset vteřin.



Obrázek 4.1: Porovnání výkonnosti knihoven na Gaussově eliminační metodě

Jak je z grafu 4.1 vidět, při použití knihovny PROFIL/BIAS je výpočet zhruba čtyřikrát rychlejší. To je způsobeno tím, že některé operace knihovna PROFIL/BIAS vektorizuje.
Stejným způsobem se postupovalo při měření výkonnosti iterační metody. Pro měření byly použity soustavy rovnic o velikost $800, 900, \ldots, 1300$ rovnic. Nejdelší výpočet při 1300 rovnicích s knihovnou Boost trval sedm minut.



Obrázek 4.2: Porovnání výkonnosti knihoven na Gauss-Seidelově metodě

Z grafu 4.2 je vidět, že je opět PROFIL/BIAS výkonnější než Boost knihovna. Porovnání doby běhu pro Gaussovu eliminaci a Gauss-Seidelovu metodu s předpodmíněním vychází Gaussova eliminace čtyřikrát rychlejší.

Závěr

Cílem této práce bylo implementovat metody pro řešení soustavy lineárních intervalových rovnic a následně porovnat výsledné šířky intervalů při použití jednotlivých metod, navrhnout možné řešení jejich minimalizace a porovnání výkonnosti použitých knihoven.
Pro tato porovnání byly implementovány metody Gaussovy eliminace, LU rozkladu, Jacobiovy, Gauss-Seidelovy a metoda Sdružených gradientů pomocí knihoven PROFIL/BIAS a Boost interval arithmetic library. Pro minimalizaci výsledných šířek vektorů byly implementovány metody pivotace a předpodmínění.
Při použití metod na soustavu se zvolenými vlastnostmi pro šířku intervalů výsledného vektoru platí, že Gaussova eliminace bez předpodmínění má zhruba dvakrát větší šířku výsledného vektoru než s předpodmíněním. Šířky výsledných intervalů u Jacobiovy a Gauss-Seidelovy metody jsou bez použití předpodmínění tak velké, že jsou téměř nepoužitelné. Navíc při zvyšování počtu iterací se zvětšují. Při použití těchto metod s předpodmíněním dosahujeme podobných výsledků jako u Gaussovy eliminace s předpodmíněním.
U výpočtu metodou sdružených gradientů docházelo k rozšíření intervalu směrového vektoru tak, že obsahoval nulu a při následném pokusu dělením tímto intervalem byla metoda ukončena. Soustavu intervalových rovnic se tedy touto metodou nepovedlo ani jednou vyřešit.
Jako nejlepší metoda pro výběr pivotu vychází střední hodnota intervalu, nicméně rozdíly mezi touto metodou a výběru pomocí absolutní hodnoty nebo mignitude je velmi malý. Při použití zároveň s předpodmínění je rozdíl nepodstatný.
Pro implementaci těchto metod vychází jako výkonnější knihovna PROFIL, která implementuje třídy intervalových matic a intervalových vektorů, jejichž operace optimalizuje.

Literatura

[1] Pryce, J. D.; Corliss, G. F.: Interval arithmetic with containment sets. [cit. 2015-20-04]. Dostupné z: `http://www.cas.mcmaster.ca/~isl/Publications/IntvlArithCsets.pdf`

[2] Ramdani, N.; Raissi, T.; Candau, Y.: Complex interval arithmetic using polar form. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.9298&rep=rep1&type=pdf`, [cit. 2015-20-04].

[3] Moore, R. E.; Kearfott, R. B.; Cloud, M. J.: *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2004.

[4] Kearfott, R. B.; Kreinovich, V.: *Applications of Interval Computations*. Springer US, 1996.

[5] Dont, M.: *Elementy numerické lineární algebry*. Vydavatelství ČVUT, 2004.

[6] Strang, G.: *Linear Algebra and Its Applications*. Brooks/Cole Publishing, 2005.

[7] Olšák, P.: *Úvod do algebry, zejména lineární*. FEL ČVUT, 2004.

[8] Brönnimann, H.; Melquiond, G.; Pion, S.: *The design of the Boost interval arithmetic library*. 2004, [cit. 2015-20-04]. Dostupné z: `https://www.lri.fr/~melquion/doc/06-tcs-rnc5.pdf`

[9] Ens de Lyon: *MPFI příručka*. [cit. 2015-20-04]. Dostupné z: `http://perso.ens-lyon.fr/nathalie.revol/software.html`

[10] Boost Software: *Boost příručka*. [cit. 2015-28-03]. Dostupné z: `http://www.boost.org/doc/libs/1_57_0/libs/numeric/interval/doc/interval.htm`

[11] Knüppel, O.: *PROFIL/BIAS příručka*. 2009.

[12] Goualard, F.: *Gaol příručka.* [cit. 2015-20-04].

[13] Neumaier, A.: On Shary's algebraic approach for linear interval equations. [cit. 2015-20-04]. Dostupné z: `https://www.mat.univie.ac.at/~neum/ms/shary.pdf`

Příloha **A**

Seznam použitých zkratek

GEM Gaussova eliminační metoda

GEMC Gaussova eliminační metoda se sloupcovou pivotací

GEMR Gaussova eliminační metoda s řádkovou pivotací

CG Metoda sdružených gradientů

IEEE 754 Standard pro dvojkovou aritmetiku v pohyblivé řádové čárce

PROFIL/BIAS Programmer's Runtime Optimized Fast Interval Library / Basic Interval Arithmetic Subroutines, knihovna pro práci s intervalovou aritmetikou

Příloha **B**

Obsah přiloženého DVD

- `readme.pdf` ..... stručný popis obsahu DVD
- `readme.txt` ..... stručný popis obsahu DVD
- `bp`
  - `bin` ..... adresář se spustitelnou formou implementace
    - `BoostSolver.out` ..... řešič implem. pomocí knihovny Boost
    - `ProfilSover.out` ..... řešič implem. pomocí knihovny PROFIL
    - `generator.out` ..... generátor soustav
  - `BoostSolverSource` ..... zdrojové kódy řešiče s knihovnou Boost
  - `ProfilSolverSource` ..... zdrojové kódy řešiče s knihovnou PROFIL
  - `GeneratorSource` ..... zdrojové kódy generátoru
  - `lib` ..... knihovny pro řešiče
  - `data` ..... připravená vstupní data pro řešič
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS