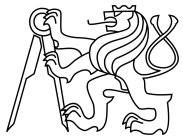
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Pokročilé algoritmy pro řešení soustav lineárních intervalových rovnic
Student:	Bc. Martin Kulle
Vedoucí:	doc. Ing. Ivan Šimeček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Systémové programování
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

1. Seznamte se a implementujte speciální algoritmy pro řešení soustav lineárních intervalových rovnic (zejména Gaussovu eliminaci, HBR metodu [1], Modifikovanou intervalovou aritmetiku pro GE [2]).
2. Porovnejte šířky výsledných vektorů u jednotlivých metod.
3. Diskutujte možnosti paralelizace a možnosti použití knihoven pro intervalovou aritmetiku ve vícevláknovém prostředí.
4. Implementujte paralelní (vícevláknovou) verzi těchto algoritmů pomocí knihovny OpenMP.
5. Porovnejte výkonnost a efektivnost jednotlivých metod na fakultním svazku Star.

Seznam odborné literatury

[1] Rohn, J.: A Manual of Results on Interval Linear Problems, http://uivtx.cs.cas.cz/ rohn/publist/!zdmanual.pdf
[2] Zohreh, K.; Allahdadi, M.: Solving Interval Linear Equations with Modified Interval Arithmetic, http://www.journalrepository.org/media/journals/BJMCS_6/2015/Jul/Allahdadi1022015BJMCS18825.pdf

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

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 1. prosince 2016

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

Diplomová práce

Pokročilé algoritmy pro řešení soustav lineárních intervalových rovnic

Bc. Martin Kulle

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

8. května 2017

Poděkování

Děkuji vedoucímu mé práce doc. Ing. Ivan Šimeček, 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 8. května 2017 .....

Odkaz na tuto práci

Kulle, Martin. *Pokročilé algoritmy pro řešení soustav lineárních intervalových rovnic.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato diplomová práce se zabývá pokročilými algoritmy pro řešení soustav lineárních intervalových rovnic. Metody použité pro řešení těchto soustav jsou Gaussova eliminace, Modifikovaná Gaussova eliminace a metoda HBR. Porovnává tyto algoritmy z hlediska přesnosti výsledného řešení a rychlosti řešení. Také se zabývá možností paralelizace těchto algoritmů pomocí knihovny OpenMP.

Klíčová slova Intervalová aritmetika, Soustava lineárních intervalových rovnic, Gaussova eliminace, Modifikovaná intervalová aritmetika, Metoda HBR

Abstract

The diploma thesis describes advanced algorithms of system of linear interval equation. Methods used to solve these systems are Gaussian elimination, Modified Gaussian elimination and HBR method. The thesis compares these algorithms in terms of accuracy and in terms of efectivity. It also deals with possibility of parallelization these algorithms with the OpenMP library.

Keywords Interval arithmetic, System of linear interval equations, Gaussian elimination, Modified interval arithmetic, HBR method

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Úvod

Intervalová aritmetika byla představena v padesátých letech dvacátého století, jako nástroj pro měření chyb při využití numerických výpočtů. S postupem času se rozšiřují i oblasti, kde je možnost využití intervalové aritmetiky. Jednou z oblastí kde intervalová aritmetika je v nynější době často používána je ekonomie, některé metody jsou představeny v [1] a [2].
Spolu s tímto rozšířením intervalové aritmetiky vznikla nutnost dokázat řešit soustavy lineárních intervalových rovnic. Zatímco v klasické aritmetice existují metody, které dokáží soustavy lineárních rovnic v polynomiálním čase, tak řešení soustavy lineárních intervalových rovnic je NP-těžký problém [3]. Proto při řešení těchto intervalových rovnic hledáme pouze obal a snaha je najít takovýto obal co nejmenší.
Ukazuje se, že použití klasických metod, ve kterých pouze běžné operace nahradíme intervalovými operacemi, nedává vždy uspokojivé výsledky. Z tohoto důvodu vznikají speciální metody pro řešení soustav lineárních intervalových rovnic.
Těchto speciálních metod je celá řada. V této diplomové práci jsou prezentovány metody Modifikované intervalové aritmetiky a metody HBR, které jsou porovnány s klasickou Gaussovou eliminací z hlediska kvality výsledků. Dále je rozebrána možnost paralelizace těchto metod a změřena jejich efektivita.

Kapitola 1

Úvod do problematiky

Tato kapitola popisuje základy lineární algebry [4], [5], intervalovou aritmetiku a přenesení některých pojmů do intervalové aritmetiky [6]. Na konci kapitoly je popsána Modifikovaná intervalová aritmetika [7].

Definice 1.1. Reálný vektor je uspořádaná n-tice reálných čísel

$$\mathbf{R}^n = \{(x_1, x_2, \cdots, x_n) \mid x_i \in \mathbf{R}\}.$$

Definice 1.2. Lineární kombinace vektorů je vektor

$$\boldsymbol{x} = \alpha_1 \cdot \boldsymbol{x}_1 + \alpha_2 \cdot \boldsymbol{x}_2 + \cdots + \alpha_n \cdot \boldsymbol{x}_n.$$

Definice 1.3. Triviální lineární kombinace vektorů je lineární kombinace, která má všechny koeficienty α nulové.

$$0\boldsymbol{x}_1 + 0\boldsymbol{x}_2 + \cdots + 0\boldsymbol{x}_n.$$

Výsledkem triviální lineární kombinace je nulový vektor.

Definice 1.4. Lineárně závislá množina vektorů $\mathbf{Z}$ je taková množina vektorů, pro kterou existuje netriviální lineární kombinace vektorů z množiny $\mathbf{Z}$, která odpovídá nulovému vektoru.

Definice 1.5. Reálná matice (m, n) je uspořádaná m-tice reálných vektorů z množiny $\mathbf{R}^n$

$$\mathbf{R}^{m,n} = \{(\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}) \mid \boldsymbol{x_i} \in \mathbf{R}^n\}.$$

Matici zapisujeme ve tvaru

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{1,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix}.$$

Definice 1.6. Hodnost matice $\mathbf{A}$ je počet lineárně nezávislých řádků matice $\mathbf{A}$

Věta 1.7. Prohození řádků matice $\mathbf{A}$, vynásobení řádku matice $\mathbf{A}$ nenulovou konstantou a přičtení násobku řádku k jinému v matici $\mathbf{A}$ nemění hodnost matice $\mathbf{A}$. (Důkaz viz [4, s. 46])

Definice 1.8. Regulární matice $\mathbf{A}^{n,n}$ je taková matice, pro kterou platí

$$\mathrm{hod}(\mathbf{A}) = n$$

Definice 1.9. Singulární matice $\mathbf{A}^{n,n}$ je taková matice, která není regulární

A zřejmě tedy platí, že pokud je matice $\mathbf{A}$ singulární, pak

$$\mathrm{hod}(\mathbf{A}) < n$$

1.1 Soustava lineárních rovnic

Soustavou lineárních intervalový 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

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

Věta 1.10. Soustava rovnic $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, má jedno řešení právě tehdy, když $\mathbf{A}$ je regulární. (Důkaz viz [4, s. 76])

Dále v textu se omezíme na čtvercové soustavy, kde matice $\mathbf{A}$ je regulární.

1.2 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.

Definice 1.11. Uzavřený reálný interval je uzavřená množina reálných čísel

$$[\underline{x}, \overline{x}] = \{x \in \mathbf{R} \mid \underline{x} \leq \tilde{x} \leq \overline{x}\}.$$

Dále v textu budeme množinu všech intervalů označovat symbolem

$$\mathbf{IR} = \{\underline{x}, \overline{x} \in \mathbf{R} \mid [\underline{x}, \overline{x}]\}.$$

Dolní mez intervalu budeme označovat symbolem $\underline{x}$ a horní mez $\overline{x}$ prvek, který leží v intervalu, budeme značit symbolem $\tilde{x}$.

Definice 1.12. Degenerovaný interval je interval, pro který platí

$$\underline{x} = \overline{x}.$$

Pro práci s intervaly používáme určité vlastnosti. Mezi základní 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

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

1.3 Binární operace v intervalové aritmetice

Definice 1.13. 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.

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

Pro výsledný interval $z = x \circ y$ platí, že

$$wid(z) \geq max(wid(x), wid(y)).$$

Po každé binární operaci tedy dojde k rozšíření výsledného intervalu. Například pokud máme intervaly a, b a b není degenerovaný interval, pak $(a/b) \cdot b = c \neq a$, ale interval c obsahuje interval a.

Příklad 1.14.

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

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

1.4 Množinové operace v intervalové aritmetice

Definice 1.15. Průnik dvojice intervalů x a y je množina

$$x \cap y = \{\tilde{z} \in \mathbf{R} \mid \tilde{z} \in x \wedge \tilde{z} \in y\}.$$

Průnik dvou intervalů je tedy klasický průnik množin. Pro výpočet průniku můžeme použít vztah

$$\begin{aligned}
&x \cap y = [\max(\underline{x}, \underline{y}), \min(\overline{x}, \overline{y})] \text{ pokud } \max(\underline{x}, \underline{y}) \leq \min(\overline{x}, \overline{y}) \\
&x \cap y = \emptyset \text{ pokud } \max(\underline{x}, \underline{y}) > \min(\overline{x}, \overline{y})
\end{aligned}$$

Průnik dvou intervalů je také symetrický, takže platí

$$x \cap y = y \cap x.$$

V případě sjednocení dvojice intervalů x a y je situace složitější. Pokud bychom použili klasické sjednocení množin, tak v případě že $x \cap y = \emptyset$ dostaneme $z = x \cup y$. Ale z pak není interval, protože existuje prvek e, pro který platí

$$\underline{z} < e < \overline{z} \ \wedge \ e \notin x \cup y.$$

Příklad 1.16.

$$[2,3] \cup [5,6] = \{x \in \mathbf{IR} |\ x \in [2,3] \vee x \in [5,6]\}$$

Platí že $2 < 4 < 6$ ale $4 \notin [2,3] \cup [5,6]$.
Tento problém můžeme vyřešit tak, že při sjednocení intervalů x a y, pro které platí $x \cap y = \emptyset$, do výsledné množiny přidáme prvky, které jsou mezi těmito intervaly.

Definice 1.17. Sjednocení dvojice intervalů x a y je množina

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

1.5 Intervalové obaly

Z výše uvedeného sjednocení intervalů je vidět, že někdy je potřeba popsat určitou množinu pomocí intervalů. Mějme tedy množinu

$$A = \{a \in \mathbf{R}^n\}.$$

Pro popis množiny pomocí intervalů budeme používat výraz intervalový obal.

Definice 1.18. Intervalová obálka(Interval hull) v množiny A je intervalový vektor, který obsahuje všechny prvky množiny A

$$A \subseteq v.$$

Intervalových obálek je nekonečné mnoho. V extrémním případě obálka v může být

$$v = ([-\infty, \infty], \cdots, [-\infty, \infty]).$$

Proto se zaměřujeme na hledání co nejmenší obálky.

Definice 1.19. Přesná intervalová obálka(Exact interval hull) w množiny A je nejmenší intervalová obálka a platí tedy

$$A \subseteq w \subseteq v,$$

kde v jsou všechny intervalové obálky množiny A.

1.6 Intervalový vektor

Definice 1.20. 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 můžeme použít vlastnosti intervalů pro vytvoření reálných vektorů, tedy jedná se o zobrazení $\mathbf{IR}^n \to \mathbf{R}^n$.
Vektor dolních a horních mezí vektoru $\boldsymbol{x}$ značíme

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

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

Vektor poloměrů

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

Vektor středních hodnot

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

Vektor šířek

$$wid(\boldsymbol{x}) = \overline{x_i} - \underline{x_i}.$$

Vektor absolutních hodnot

$$abs(\boldsymbol{x}) = max\{|\tilde{x_i}| \mid \tilde{x_i} \in x_i\}.$$

Vektor mignitude

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

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

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

1.7 Intervalová matice

Definice 1.21. 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, které převádí $\mathbf{IR}^{n\times m} \to \mathbf{R}^{n\times m}$.
Matice poloměrů

$$\boldsymbol{\Delta} = rad(\mathbf{A}) = \frac{1}{2}wid(A_{i,j}) = \frac{1}{2}(\overline{A_{i,j}} - \underline{A_{i,j}}).$$

Matice středních hodnot

$$\mathbf{A_c} = mid(\mathbf{A}) = \frac{1}{2}(\overline{A_{i,j}} + \underline{A_{i,j}}).$$

Matice šířek

$$\mathbf{A_w} = wid(\mathbf{A}) = \overline{A_{i,j}} - \underline{A_{i,j}}.$$

Matice absolutních hodnot

$$\mathbf{A_{abs}} = abs(\mathbf{A}) = max\{|\tilde{A}_{i,j}| \mid \tilde{A}_{i,j} \in A_{i,j}\}.$$

Matice mignitude

$$\mathbf{A_{mig}} = mig(\mathbf{A}) = min\{|\tilde{A_{i,j}}| \mid \tilde{A_{i,j}} \in A_{i,j}\}.$$

Definice 1.22. Reálná intervalová matice $[\mathbf{A}]$ je regulární právě tehdy, když $\forall A \in [\mathbf{A}]$ je A regulární. V opačném případě je singulární.

Zjistit, jestli intervalová matice je regulární je *co-NP-úplný problém*, tedy nejsme schopni zjistit regularitu v polynomiálním čase. (viz [3])

Věta 1.23. Reálná intervalová matice $\mathbf{A}$ je regulární, pokud platí

$$\varrho(|A_c^{-1}|\Delta) < 1.$$

Pokud tedy intervalová matice $\mathbf{A}$ splňuje nerovnost $\varrho(|A_c^{-1}|\Delta) < 1$, můžeme o ní říct, že je regulární. Navíc tuto podmínku jsme schopni ověřit v polynomiálním čase, protože tato nerovnost je ekvivaletní s

$$(I - |A_c^{-1}|\Delta)^{-1} \geq 0.$$

Dokázáno v [8].

1.8 Soustava lineárních intervalových rovnic

Soustavou lineárních intervalový rovnic je soustava

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

kde $\mathbf{A} \in \mathbf{IR}^{n,n}$ a $\boldsymbol{b} \in \mathbf{IR}^n$. Řešením soustavy lineárních intervalových rovnic $[\mathbf{A}]\boldsymbol{x} = [\boldsymbol{b}]$ je množina všech řešení $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\mathbf{A} \in [\mathbf{A}]$ a $\boldsymbol{b} \in [\boldsymbol{b}]$

$$\sum = \{\boldsymbol{x} \in \mathbf{R}^n \mid \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \wedge \mathbf{A} \in [\mathbf{A}] \wedge \boldsymbol{b} \in [\boldsymbol{b}]\}.$$

Řešením soustavy lineárních intervalových rovnic nemusí být vektor intervalů, proto budeme za řešení považovat intervalovou obálku $\sum$. Nalezení přesné intervalové obálky $\sum$ je ale NP-těžký problém, kvůli tomu si při hledání řešení

spokojíme pouze s hledáním co nejmenší obálky $\sum$.
Na druhou stranu pokud máme soustavu $[\mathbf{A}][\boldsymbol{x}] = [\boldsymbol{b}]$ pak vektor $[\boldsymbol{x}] = ([-\infty, \infty], \cdots, [-\infty, \infty])$ je určitě obálkou $\sum$. Takové řešení nám ale vůbec nic neřekne, proto hledáme vektor $\boldsymbol{x}$, tak aby šířky jeho intervalů byly co nejmenší. Hledáme tedy co nejtěsnější obálku $\sum$.

Příklad 1.24. Mějme soustavu dvou rovnic, která je uvedená také v [6].

$$[2,3]x_1 + [-1,2]x_2 = [3,4]$$
$$[1,3]x_1 + [4,6]x_2 = [2,4]$$

První rovnici můžeme upravit na tvar

$$[2,3]x_1 = [3,4] - [-1,2]x_2.$$

Z toho dostaneme řešení

$$x_1 = \begin{cases} [\frac{3}{2} - x_2, 2 + \frac{1}{2}x_2] & \text{pro } x_2 \geq \frac{3}{2} \\ [1 - \frac{2}{3}x_2, 2 + \frac{1}{2}x_2] & \text{pro } 0 \leq x_2 < \frac{3}{2} \\ [1 + \frac{1}{3}x_2, 2 - x_2] & \text{pro } -3 \leq x_2 < 0 \\ [\frac{3}{2} + \frac{1}{2}x_2, 2 - x_2] & \text{pro } x_2 < -3 \end{cases}$$

Intervaly x_1 jsou na obrázku 1.1 zobrazeny horizontálním šrafováním. Podobně druhou rovnici upravíme na

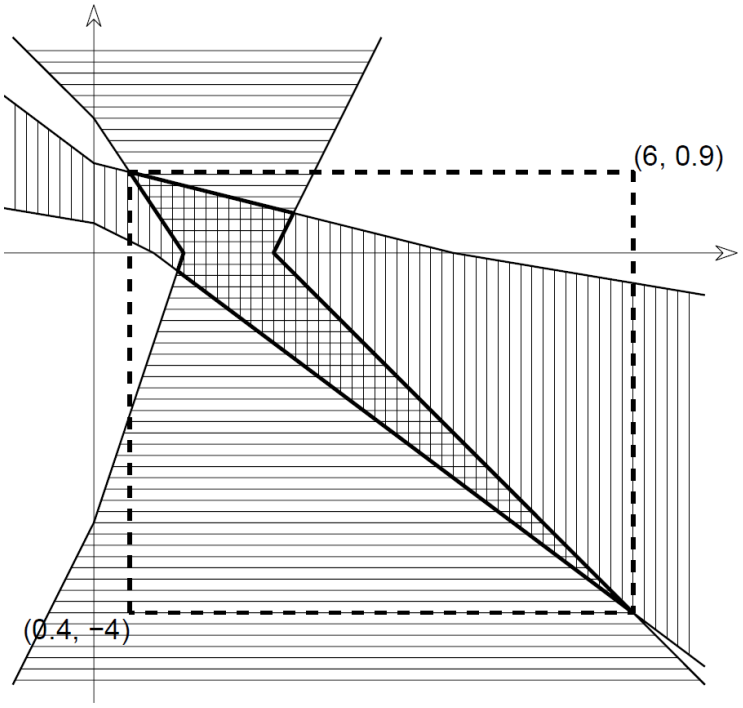
$$[4,6]x_2 = [2,4] - [1,3]x_1.$$

a dostaneme řešení

$$x_2 = \begin{cases} [\frac{1}{2} - \frac{3}{4}x_1, \frac{2}{3} - \frac{1}{6}x_1] & \text{pro } x_1 \geq 4 \\ [\frac{1}{2} - \frac{3}{4}x_1, 1 - \frac{1}{4}x_1] & \text{pro } \frac{2}{3} \leq x_1 < 4 \\ [\frac{1}{3} - \frac{1}{2}x_1, 1 - \frac{1}{4}x_1] & \text{pro } 0 \leq x_1 < \frac{2}{3} \\ [\frac{1}{3} - \frac{1}{6}x_1, 1 - \frac{3}{4}x_1] & \text{pro } x_1 < 0 \end{cases}$$

Intervaly x_1 jsou na obrázku 1.1 zobrazeny vertikálním šrafováním.
Výsledek soustavy rovnic je průnik těchto dvou oblastí. Jak je z obrázku 1.1 vidět, výslednou oblast nejsme schopni popsat dvojicí intervalů, proto určíme obal této oblasti, která je no obrázku značena přerušovanou čarou. Výsledný vektor x je tedy

$$x = \begin{pmatrix} [0.4, 6] \\ [-4, 0.9] \end{pmatrix}.$$



Obrázek 1.1: Řešení lineární soustavy intervalových rovnic

1.9 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

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

Matice $\mathbf{P}$ se blíží jednotkové matici. 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

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

Předpodmínění můžeme použít v případě, že $\mathbf{A}$ je regulární.

1.10 Modifikovaná intervalová aritmetika

Motivací k zavedení rozšířené intervalové aritmetiky je fakt, že při provádění operací v intervalové aritmetice roste šířka intervalů velmi rychle [7]. Při řešení složitějších problémů, které vyžadují velký počet operací, tak dostáváme sice správné výsledky, ale šířka těchto výsledných intervalů je pro praktické účely příliš velká.

Jedním z možných rozšířeních je intervalová aritmetika rozšířena nulou [7]. Předpokládejme jednoduchou rovnici

$$ax - b = [0, 0], \tag{1.1}$$

kde a a b je nedegenerovaný interval. Takováto rovnice v intervalové aritmetice nemá příliš velký smysl, protože výsledkem levé strany rovnice bude vždy interval, jehož šířka je větší než nula.
V klasické intervalové aritmetice se předpokládá, že nulový prvek je interval, který obsahuje nulu. Pokud bychom se ale na nulový prvek dívali jako na rozdíl dvou různých intervalů, dostáváme

$$[\underline{a}, \overline{a}] - [\underline{a}, \overline{a}] = [\underline{a} - \overline{a}, \overline{a} - \underline{a}] = [-(\overline{a} - \underline{a}), \overline{a} - \underline{a}].$$

To znamená, že nulový prvek je interval, jehož střední hodnota je rovna nule. V modifikované intervalové aritmetice bereme, že nulový interval je interval, jehož střední hodnota je rovna nule. Také můžeme říkat, že je symetrický od nuly. Původní rovnici $ax - b = [0, 0]$ můžeme tedy upravit na

$$[\underline{a}, \overline{a}][\underline{x}, \overline{x}] - [\underline{b}, \overline{b}] = [-y, y], \tag{1.2}$$

kde hodnoty y jsou neznámé, protože neznáme ani $[\underline{x}, \overline{x}]$.
Pokud budem předpokládat, že intervaly $[\underline{a}, \overline{a}]$ a $[\underline{b}, \overline{b}]$ jsou kladné, tak dostáváme dvě rovnice

$$\underline{a} \cdot \underline{x} - \overline{b} = -y,$$
$$\overline{a} \cdot \overline{x} - \underline{b} = y.$$

Z toho dostáváme jednu lineární rovnici o dvou neznámých $\underline{x}$ a $\overline{x}$.

$$\underline{a} \cdot \underline{x} - \overline{b} + \overline{a} \cdot \overline{x} - \underline{b} = 0.$$

Pro řešení této rovnice budeme používat určité omezení. Nejprve předpokládejme, že $\underline{x} = \overline{x}$. V tomto případě dostáváme z výše uvedené rovnice degenerovaný interval

$$x_c = \frac{\underline{b} + \overline{b}}{\underline{a} + \overline{a}}$$

Hodnotu x_c můžeme brát jako horní mez pro $\underline{x}$ a dolní mez pro $\overline{x}$. Opačné meze můžeme definovat pomocí klasické intervalové aritmetiky. Z toho můžeme říct, že hodnoty $\underline{x}$ a $\overline{x}$ se nacházejí v intervalu

$$\underline{x} = [\frac{\underline{b}}{\overline{a}}, x_c] \ , \ \underline{x} = [x_c \frac{\overline{b}}{\underline{a}}]$$

Při dosazení do výše uvedené rovnice $\underline{a} \cdot \underline{x} - \overline{b} + \overline{a} \cdot \overline{x} - \underline{b} = 0$ dostáváme

$$\underline{x} = \frac{\underline{b} + \overline{b} - \overline{ax}}{\underline{a}}, \overline{x} \in \left[x_c, \frac{\overline{b}}{\underline{a}}\right]$$

$$\overline{x} = \frac{\underline{b} + \overline{b} - \underline{a}\underline{x}}{\overline{a}}, \underline{x} \in \left[x_c, \frac{\underline{b}}{\overline{a}}\right]$$

Když za $\overline{x}$ dosadíme maximální možnou hodnotu, což je $\overline{x} = \overline{b}/\underline{a}$, dostáváme minimální hodnotu $\underline{x}$

$$\underline{x}_{min} = \frac{\underline{b} + \overline{b}}{\underline{a}} - \frac{\overline{a}\,\overline{b}}{\underline{a}^2}$$

Když naopak za $\underline{x}$ dosadíme minimální hodnotu $\underline{x} = \underline{b}/\overline{a}$, dostáváme maximální hodnotu $\overline{x}$

$$\overline{x}_{min} = \frac{\underline{b} + \overline{b}}{\overline{a}} - \frac{\underline{a}\,\underline{b}}{\overline{a}^2}$$

Protože je možné, že hodnota $\underline{x}_{min} < \underline{b}/\overline{a}$, což je minimální hodnota $\underline{x}_{min}$ s využitím klasické intervalové matematiky, tak budeme používat

$$\underline{x}_{max} = \max\left(\frac{\underline{b}}{\overline{a}}, \frac{\underline{b} + \overline{b}}{\underline{a}} - \frac{\overline{a}\,\overline{b}}{\underline{a}^2}\right)$$

Podobně je možné, že hodnota $\overline{x}_{max} > \overline{b}/\underline{a}$, proto budeme používat

$$\overline{x}_{min} = \min\left(\frac{\overline{b}}{\underline{a}}, \frac{\underline{b} + \overline{b}}{\overline{a}} - \frac{\underline{a}\,\underline{b}}{\overline{a}^2}\right)$$

Tedy dostaneme intervaly

$$[\underline{x}] = [\underline{x}_{max}, x_c]\ ,\ [\overline{x}] = [x_c, \overline{x}_{min}]$$

Intervaly $[\underline{x}]$ a $[\overline{x}]$ nám určují všechna možná řešení rovnice

$$[\underline{a}, \overline{a}][\underline{x}, \overline{x}] - [\underline{b}, \overline{b}] = [-y, y].$$

Hodnoty $\overline{x}_{min}$ a $\underline{x}_{max}$ nám tvoří nejširší možný interval výše uvedené rovnice. Tedy maximální šířka $w_{max} = \overline{x}_{min} - \underline{x}_{max}$, což odpovídá hodnotě

$$y : y_{max} = \frac{\overline{a}\,\overline{b}}{\underline{a}} - \underline{b}.$$

A naopak při dosazení do rovnice degenrovaného řešení x_c dostáváme

$$y : y_{min} = \frac{\overline{a}\,\overline{b} - \underline{a}\underline{b}}{\underline{a} + \overline{a}}.$$

Tudíž intervaly $\underline{x}$ a $\overline{x}$ představují nepřetržitou množinu vnořených řešení. Takováto množina intervalů může být reprezentována jako fuzzy číslo, kde nám hodnota y reprezentuje šířku pravé strany rovnice. Hodnoty y mohou být v určitém smyslu použity jako míra nejistoty řešení odvozené od původní nejistoty z rovnice 1.2.

$$\alpha = 1 - \frac{y - y_{min}}{y_{max} - y_{min}} \tag{1.3}$$

Hodnotu α můžeme použít jako stupeň jistoty intervalového řešení rovnice 1.2. Hodnota α bude vždy od 0 do 1, přičemž $\alpha = 0$ v případě maximální šířky a $\alpha = 1$ v případě minimální šířky. Hodnotu α můžeme použít jako α - řez pro triangulární *fuzzy číslo* x', které je řešením rovnice 1.2.

$$x' = \left\{ \underline{x}_{max}, \frac{\underline{b} + \overline{b}}{\underline{a} + \overline{a}}, \overline{x}_{min} \right\} \tag{1.4}$$

Z x' jsme schopni získat řešení rovnice 1.2 klasickou operací *defuzzifikace*. V našem případě

$$\underline{x}_{def} = \frac{\int_0^1 \underline{x}(\alpha) d\alpha}{\int_0^1 d\alpha}, \overline{x}_{def} = \frac{\int_0^1 \overline{x}(\alpha) d\alpha}{\int_0^1 d\alpha} \tag{1.5}$$

Pokud budeme předpokládat, že intervaly $a, b > 0$, dostaneme následující výrazy

$$\underline{x}_{def} = \frac{\overline{b}}{\underline{a}} - \frac{y_{max} - y_{min}}{2\underline{a}}, \overline{x}_{def} = \frac{\underline{b}}{\overline{a}} - \frac{y_{max} - y_{min}}{2\overline{a}} \tag{1.6}$$

Hodnoty $\underline{x}_{def}$ a $\overline{x}_{def}$ můžeme použít jako přibližné řešení původní rovnice 1.1, kterou pokud upravíme, dostaneme $[x] = [b]/[a]$. Tudíž se můžeme na hodnoty $\underline{x}_{def}$ a $\overline{x}_{def}$ dívat jako na výsledky modifikovaného intervalového dělení.

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 většinou dostaneme výsledky, které jsou příliš pesimistické (šířka výsledného vektoru bude příliš veliká). Navíc někdy nemusíme nalézt vůbec žádné řešení soustavy rovnic, i když existuje. Je to způsobeno tím, že postupnými úpravami matice rozšíříme intervaly tak, že obsahují nulu. Pokud algoritmus potřebuje tímto intervalem dělit, nemůže pokračovat, protože dělení intervalem, který obsahuje nulu, není definováno.
Proto vznikají metody speciálně určené pro řešení soustav lineárních intervalových rovnic, které řeší problém s možným nevyřešením soustavy rovnic a zároveň se snaží o co nejužší výsledný interval. Tyto metody jsou také například popsány v [4] nebo v [5]. Metoda HBR je také popsána v [9].

2.1 Trojúhelníkové soustavy

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

Definice 2.1. Nechť $\mathbf{L} \in \mathbf{R}^{nxn}$ 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.

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

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

Výsledný algoritmus lze rozdělit na dvě části. Dopředný průchod vytvoří z matice A horní trojúhelníkovou.
A zpětný průchod, který odpovídá rovnici 2.2

Algoritmus 1 Dopředný průchod Gaussovy eliminace

```
1:  procedure ForwardGem(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 ← j, ..., 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: end procedure
```

2.3 Gaussova eliminační metoda v intervalové aritmetice

Nejjednoduší možnost jak vyřešit soustavu lineárních intervalových rovnic je použití Gaussovy eliminační metody, kde matici **A** nahradíme intervalovou maticí [**A**], vektor $\boldsymbol{b}$ nahradím intervalový vektorem $[\boldsymbol{b}]$ a při výpočtu použijeme operace definované intervalovou aritmetikou. Toto řešení ale přináší určité problémy. Za prvé nalezené řešení bude pravděpodobně příliš pesimestické (šířka výsledného vektoru bude příliš veliká).

Algoritmus 2 Zpětný průchod Gaussovy eliminace

```
1: procedure BackwardGem(A',b')
2:     for i ← n, ..., 1 do
3:         for j ← n, ..., i + 1 do
4:             b_i ← b_i − b_j ∗ A_{i,j}
5:         end for
6:         b_i ← b_i / A_{i,i}
7:     end for
8: end procedure
```

Příklad 2.2. Mějme rovnici

$$\begin{pmatrix} [2,3] & [-1,2] \\ [1,3] & [4,6] \end{pmatrix} x = \begin{pmatrix} [3,4] \\ [2,4] \end{pmatrix}$$

V prvním kroku spočítáme poměr

$$r = A_{1,0}/A_{0,0} = \left[\frac{1}{3}, \frac{3}{2}\right]$$

V druhém kroku od 2. řádku odečteme 1. řádek matice A krát r. Stejně tak vektor b upravíme na b' a dostavame

$$A' = \begin{pmatrix} [2,3] & [-1,2] \\ [0,0] & [1,7.5] \end{pmatrix} x = \begin{pmatrix} [3,4] \\ [-4,3] \end{pmatrix}$$

Po zpětné substituci jako v 2.2 dostáváme výsledný vektor x

$$x = \begin{pmatrix} [-1.5,6] \\ [-4,3] \end{pmatrix}$$

Původní rovnici ale splňuje i vektor

$$x^* = \begin{pmatrix} [0.4,6] \\ [-4,0.9] \end{pmatrix}. \tag{2.3}$$

Z výsledků dostáváme $wid(x*) = (5.6, 4.9), wid(x) = (7.5, 7)$. Tedy průměrná šířka vektoru x je zhruba o 38% větší než průměrná šířka intervalu $x*$.

Druhým možným problémem je, že nenalezneme řešení ani v případě, když matice A je regulární. To je způsobeno tím, že v průběhu upravování matice A na trojúhelníkový tvar dojde k rozšíření intervalů na diagonále takovým způsobem, že bude obsahovat nulu.

Příklad 2.3. Mějme rovnici

$$\begin{pmatrix} [2,4] & [0,2] & [-1,1] \\ [1,1] & [3,4] & [-0.5,1.5] \\ [4,6] & [3,5] & [8,10] \end{pmatrix} x = \begin{pmatrix} [8,10] \\ [10,12] \\ [8,10] \end{pmatrix}$$

Postupným upravováním matice **A** dostaneme

$$A' = \begin{pmatrix} [2,4] & [0,2] & [-1,1] \\ [0,0] & [2,4] & [-1,2] \\ [0,0] & [0,0] & [0,16] \end{pmatrix}, b' = \begin{pmatrix} [5,10] \\ [-47,17] \\ [-64.5,27.5] \end{pmatrix}.$$

V prvním kroku zpětné substituce dostáváme výsledný vektor

$$x_3 = [-64.5, 27.5]/[0, 16].$$

Protože dělení nulovým intervalem není definováno, tak v tento okamžik metoda Gaussovy eliminace není schopna nalézt řešení. Přitom víme, že matice **A** je regulární a při použití například metody HBR dostáváme obálku řešení

$$\begin{pmatrix} [-1.88, 9.4] \\ [0.2, 5.27] \\ [-8.78, -0.06] \end{pmatrix}.$$

2.3.1 Pivotace

Jedním z možných řešení těchto problému je použití pivotace. Pivotace se dělí na řádkovou, sloupcovou a celkovou, přičemž celková pivotace se u velkých soustav příliš nepoužívá, protože hledání pivotu u této metody je časově náročné.

Pivotace spočívá v tom, že se před vytvářením nul v i-tém sloupci hledá takzvaný pivot. V případě řádkové pivotace hledáme prvek v i-tém sloupci, který splňuje nejlépe naše předem dané kritérium. Pokud takovýto prvek nalezneme na k-tém řádku, tak v matici A prohodíme i-tý s k-tým řádkem

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,i} & \cdots & a_{1,n} \\ [0,0] & \ddots & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{i,i} & \cdots & a_{i,n} \\ [0,0] & [0,0] & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{k,i} & \cdots & a_{k,n} \\ [0,0] & [0,0] & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{n,i} & \cdots & a_{n,n} \end{pmatrix} \sim \begin{pmatrix} a_{1,1} & \cdots & a_{1,i} & \cdots & a_{1,n} \\ [0,0] & \ddots & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{k,i} & \cdots & a_{k,n} \\ [0,0] & [0,0] & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{i,i} & \cdots & a_{i,n} \\ [0,0] & [0,0] & \vdots & \cdots & \vdots \\ [0,0] & [0,0] & a_{n,i} & \cdots & a_{n,n} \end{pmatrix}.$$

V případě sloupcové pivotace postupujeme stejně, jen hledáme pivot v určitém řádku a prohazujeme sloupce matice **A** a vektoru $\boldsymbol{b}$.

Otázkou je jaké použít kritérium pro hledání pivota. V případě klasické aritmetiky se nejčastěji používá nalezení prvku, který má největší absolutní hodnotu. To můžeme do intervalové aritmetiky převést na hledání největší absolutní hodnoty, mignitude nebo střední hodnoty.
Občas se někdy také používá pro výběr pivotu kritérium nejmenší šířky intervalu, který neobsahuje nulu. To má za cíl zamezit přílišnému rozšiřování výsledného intervalu.

2.3.2 Předpodmínění

Další možnou variantou pro zlepšení výsledků je předpodmínění zadané soustavy. To se provádí pomocí vynásobení matice $\mathbf{A}$ a vektoru $\boldsymbol{b}$ maticí středních $\mathbf{A_c}$ hodnot a dostáváme tedy soustavu

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

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

Příklad 2.4. Mějme stejnou soustavu rovnic jako v příkladu 2.3

$$\begin{pmatrix} [2,4] & [0,2] & [-1,1] \\ [1,1] & [3,4] & [-0.5,1.5] \\ [4,6] & [3,5] & [8,10] \end{pmatrix} x = \begin{pmatrix} [8,10] \\ [10,12] \\ [8,10] \end{pmatrix}$$

Matice středních hodnot a její inverzní matice jsou

$$\mathbf{A_c} = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 3.5 & 0.5 \\ 5 & 4 & 9 \end{pmatrix} \quad (\mathbf{A_c})^{-1} = \begin{pmatrix} 0.36 & -0.11 & 0.01 \\ -0.08 & 0.33 & -0.02 \\ -0.16 & -0.09 & 0.12 \end{pmatrix}$$

Po vynásobení původní matice $\mathbf{A}$ maticí $(\mathbf{A_c})^{-1}$ dostáváme

$$\mathbf{P} = \begin{pmatrix} [0.63,1.37] & [-0.42,0.42] & [-0.48,0.48] \\ [-0.10,0.10] & [0.74,1.26] & [-0.43,0.43] \\ [-0.28,0.28] & [-0.32,0.32] & [0.63,1.37] \end{pmatrix} \quad b' = \begin{pmatrix} [1.61,2.56] \\ [2.32,3.17] \\ [-1.74,-1.01] \end{pmatrix}.$$

Při řešení této soustavy Gaussovou eliminační metodou se už nestane, že by došlo k dělení intervalem, který obsahuje nulu jako v případě 2.3 .

2.4 Gaussova eliminace s modifikovanou intervalovou aritmetikou

Gaussova eliminace s modifikovanou intervalovou aritmetikou odpovídá klasické Gaussově eliminaci, kde je místo klasického intervalového dělení použito modifikované dělení uvedené v kapitole 1.10.
Výhodou tohoto řešení je, že výsledné šířky intervalů by měly být výrazně

menší než u klasické Gaussovy eliminace. Na druhou stranu nemáme zaručeno, že výsledný vektor intervalů bude obsahovat všechna řešení. To ale ve skutečnosti nemusí být až tak velká potíž, protože k tomuto problému dojde jen v extrémních případech a i když k tomu dojde, s velkou pravděpodobností bude většinu řešení výsledný vektor intervalů obsahovat viz [7]. Navíc řešení, která nebude obsahovat, budou hraniční řešení, která nejsou při většině aplikací soustavy intervalových rovnic tak důležitá.
Pokud se vrátíme k příkladu 2.2

$$\begin{pmatrix} [2,3] & [-1,2] \\ [1,3] & [4,6] \end{pmatrix} x = \begin{pmatrix} [3,4] \\ [2,4] \end{pmatrix}.$$

Pomocí klasické Gaussovy eliminace jsme dostali řešení

$$x_{\mathrm{GEM}} = \begin{pmatrix} [-1.5,6] \\ [-4,3] \end{pmatrix}.$$

Při použití modifikované Gaussovy eliminace dostáváme

$$x_{\mathrm{MGEM}} = \begin{pmatrix} [-0.5,6] \\ [-4,2.3] \end{pmatrix}.$$

Výsledek x_{MGEM} při tom obsahuje všechna řešení 2.3 a průměrná šířka x_{GEM} je větší o 12 %.

2.5 HBR metoda

Metoda Hansen-Bliek-Rohn [9] je přímá metoda, která nalezne řešení soustavy lineárních intervalových rovnic za podmínky

$$\varrho(|A_c^{-1}|\Delta) < 1, \tag{2.4}$$

kde $\varrho(A)$ je spektrální poloměr matice A. Tato podmínka je ekvivalentní s nerovností

$$(I - |A_c^{-1}|\Delta)^{-1} \geq 0. \tag{2.5}$$

Metoda Hansen-Bliek-Rohn na rozdíl od předešlých metod pro výpočet nepoužívá intervalové matice a vektory, ale používá klasické matice a vektory.
Pro výpočet budeme potřebovat matici středních hodnot, matici poloměrů, vektor středních hodnot a vektor poloměrů

$$\begin{aligned} \mathbf{A_c} &= mid(\mathbf{A}) \\ \boldsymbol{\Delta} &= rad(\mathbf{A}) \\ \boldsymbol{x_c} &= mid(\boldsymbol{x}) \\ \boldsymbol{\delta} &= rad(\boldsymbol{x}). \end{aligned}$$

Dále spočítáme matici M, kterou využijeme i pro ověření nerovnosti (2.5)

$$M = (I - |A_c^{-1}|\Delta)^{-1}. \tag{2.6}$$

Z matice M určíme vektor μ

$$\mu = (M_{1,1}, M_{2,2}, \cdots, M_{n,n}) \tag{2.7}$$

a matici T_μ, což je diagonální matice s prvky μ na diagonále. Pomocí T_μ určíme diagonální matici

$$T_v = (2T_\mu - I)^{-1}. \tag{2.8}$$

Dále vyřešíme soustavu lineárních rovnic pro matici a vektor středních hodnot

$$x_c = A_c^{-1} b_c. \tag{2.9}$$

Tento vektor použijeme pro určení středu výsledného vektoru soustavy intervalových rovnic. Dále spočítáme vektor určující poloměry výsledného vektoru

$$x^* = M(|x_c| + |A_c^{-1}\delta|). \tag{2.10}$$

Pomocí těchto vektorů spočítáme možnou horní a dolní mez výsledného vektoru

$$\underline{x_o} = -x^* + T_\mu(x_c + |x_c|)$$
$$\overline{x_o} = x^* + T_\mu(x_c - |x_c|).$$

Výsledný vektor je minimum, respektive maximum těchto vektorů a jejich násobků s maticí T_v

$$\underline{x} = \min\{\underline{x_o}, \underline{x_o} T_v\}$$
$$\overline{x} = \max\{\overline{x_o}, \overline{x_o} T_v\}.$$

Kapitola 3

Realizace

V této kapitole jsou popsána existující řešení pro lineární algebru a intervalovou aritmetiku. Dále jsou rozebrány možnosti optimalizace a na konci kapitoly je popsána vlastní realizace.

3.1 Existující řešení

Pro práci s lineární algebrou a intervalovou aritmetikou už existují knihovny, které při vlastní implementaci můžeme použít.

3.1.1 Knihovny pro lineární algebru

Nutnost použití matic a vektorů se objevuje ve velkém množství nejrůznějších problémů. Díky tomu postupně vzniklo velké množství knihoven pro lineární algebru, proto je dále popsán jenom malý výčet z těchto knihoven.

BLAS, OpenBLAS

Knihovna BLAS vznikla na počátku sedmdesátých a obsahuje základní operace lineární algebry. Knihovna je napsána v jazyku *Fortran*, ale lze použít i v programech psaných v jazyku C. Pro snadnější použití v jazyku C je možnost použít rozhraní CBLAS. Knihovna OpenBLAS je jedna z vylepšených implementací knihovny BLAS.

LAPACK

LAPACK [10] je knihovna napsaná v jazyku Fortran pro řešení problémů lineární algebry. Knihovna vznikla v roce 1977 a byla navržena tak, aby byla výkonná na co největším počtu počítačů. Na rozdíl od knihovny BLAS obsahuje funkce pro složitější problémy jako například hledání vlastních čísel

matice. Knihovna LAPACK pro základní operace s maticemi a vektory používá knihovnu BLAS.

Armadillo

Armadillo je knihovna zaměřená na lineární algebru pro programovací jazyk C++. Knihovna umožňuje práci s reálnými vektory a maticemi. Pro práci s maticí je používána třída *Mat*, pro sloupcový vektor třída *Col* a pro řádkový vektor *Row*. Jako hodnoty těchto matic je možné použít celá čísla, čísla s plovoucí čárkou (*float*, *double*) a komplexní čísla. Knihovna pro některé operace používá knihovnu LAPACK, ale místo ní mohou být použity Intel MKL [11], OpenBLAS nebo NVBLAS [12], která používá pro některé operace grafickou kartu.

3.1.2 Knihovny pro intervalovou aritmetiku

Pro práci s intervalovou aritmetikou je také možno použít už některé existující knihovny. Mezi nejpoužívanější patří knihovna PROFIL/BIAS [13], nebo Boost Interval Arithmetic Library [14].

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.
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. Pro získání vlastností intervalových vektorů, respektive intervalových matic slouží definované funkce. Všechny funkce jsou podrobně popsány v [13].

Boost Interval Arithmetic Library

Boost Interval Arithmetic Library je sada knihoven určených pro programovací jazyk C++. 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 [14].
Žádná z těchto knihoven ale neimplementuje metodu pro řešení soustavy lineárních intervalových rovnic. Jedním z důvodů je, že neexistuje obecné řešení tohoto problému. Při implementaci metod pro řešení soustav lineárních intervalových rovnic za použití knihoven pro intervalovou aritmetiku tak musíme při každé operaci volat knihovní funkce. Z čehož vyplývá určitá neefektivita.

3.2 Možnosti optimalizace

3.2.1 Vektorizace

Vektorové počítače vznikly na začátku sedmdesátých let dvacátého století, jejich hlavní využití bylo v superpočítačích. Princip vektorizace spočívá v tom, že procesor v jeden okamžik provádí opearce nad různými daty. Vektorové procesory se rozdělují na *SIMD* (Single Instruction, Multiple Data), kdy je možno provádět pouze jednu instrukci nad větším počtem dat, a na *MIMD* (Multiple Instruction, Multiple Data) procesory, které můžou provádět více instrukcí pro různá data v jednom okmažiku.
V roce 1997 přišla firma *Intel* s procesory, které obsahovaly technologii *MMX*. *MMX* umožňovala použít *SIMD* instrukce pro operace s celými čísly. Používala proto osm 64bitových registrů, díky kterým mohlo být v jeden okmažik zpracováno osm 8-bitových čísel, čtyři 16-bitová čísla nebo dvě 32-bitová čísla. Možnost využití těchto *SIMD* instrukcí pouze pro celá čísla bylo velké omezení, proto další vylepšení *MMX* označované jako *SSE* (Streaming SIMD Extensions) už umí pracovat s čísly s plovoucí čárkou. V současné době je nejnovější verze *SSE 4.2*.
Většina kompilátorů dokáže automaticky generovat kód s vektorovými instrukcemi. Nejdříve ale musí zdrojový kód analyzovat a zjistit, jestli jsou splněny určité podmínky. Pokud jsou podmínky splněny, pak použije vektorové instrukce. Kompilátor *GCC* používá automatickou vektorizaci pouze pro cykly. Hlavní podmínkou je, aby neexistovala datová závislost mezi různými iteracemi, protože pak by nebylo možné zaručit správnost výpočtu. Kompilátor je obecně při generování kódu velmi konzervativní, když si není jist, že výsledný program se bude chovat správně, radši žádnou optimalizaci neudělá. Proto při použití automatické vektorizace je potřeba upravit kód takovým způsobem, aby kompilátor mohl použít vektorizaci.

3.2.2 Paralelizace

Motivací pro paralelizaci je fakt, že velké množství problémů lze rozdělit na menší problémy, které jsou na sobě nezávislé. Po vyřešení těchto nezávislých

částí dostaneme jejich částečné výsledky a z nich vytvoříme konečný výsledek. Dále v textu se omezíme na paralelní zpracování se sdílenou pamětí, které dnes podporují prakticky všechny procesory určené do běžný počítačů. Jednotka provádějící určitou část kódu sekvenčně se označuje jako *vlákno*. Při paralelním zpracování se sdílenou pamětí mají všechna vlákna přístup do stejné paměti. Výhodou tohoto přístupu je, že v průběhu výpočtu mohou mezi sebou vlákna jednoduše sdílet mezivýpočty. Naopak nevýhodou je, že pokud program nepracuje korektně, pak si vlákna mohou tyto hodnoty navzájem nezáměrně přepisovat. Zajištění, že vícevláknový program bude pracovat korektně, a odhalování případných chyb je poměrně složité, proto existují knihovny, které vytváření vícevláknových programů usnadňují.

3.2.3 OpenMP

OpenMP je knihovna pro vytváření vícevláknových programů se sdílenou pamětí v jazycích C,C++ a Fortran. OpenMP umožnuje vytvářet vícevláknové programy pro různé platformy a různé operační systémy například Linux, Windows, macOS a další.
Pro určení části kódu, které mají být paralelizovány, se používají direktivy, jak je vidět v algoritmu 3 na 2. řádku.
Překladač podporující OpenMP tento cyklus upraví tak, že na začátku vy-

Algoritmus 3 Jednoduchý cyklus v OpenMP

```
1: ⋮
2: #pragma omp parallel for
3: for (int i=0; i<N; i++) {
4:     A[i] = B[i] + C[i]
5: }
6: ⋮
```

tvoří určitý počet vláken a každému vláknu přiřadí jen určitou část cyklu. Na konci cyklu se počká na dokončení všech vláken a další část programu se dále provádí sekvenčně. Pro práci s proměnnými OpenMP rozděluje data na několik tříd

shared data jsou mezi vlákny sdílena, pokud není uvedeno jinak, tak proměnná je shared

private každé vlákno má svojí proměnnou, která není na počátku cyklu inicializována

default umožnuje pro různé programovací jazyky určit různou defaultní třídu

firstprivate jako private s tím rozdílem, že na začátku je proměnná inicializována na hodnotu, kterou měla daná proměnná na začátku cyklu

lastprivate jako private s tím rozdílem, že po ukončení všech vláken se do původní proměnné v sekvenční části zapíše hodnota té samé privátní proměnné z posledního vlákna

reduction pro redukci všech privátních proměnných na konci cyklu

Před použitím OpenMP je pro správné chování programu potřeba určit, do jaké třídy proměnné v paralelním bloku spadají, a následně je specifikovat. Pokud není určeno, jakého typu daná proměnná je, pak je braná jako sdílená proměnná.
OpenMP dokáže rozdělovat části cyklů různými způsoby. To je možné nastavit pomocí plánování, kde se ve zdrojovém souboru do direktivy přidá část schedule(type, chunk). OpenMP používá tři různé typy plánování.

static Všechna vlákna dostávají stejný počet chunků

dynamic Na začátku všechna vlákna dostanou jeden chunk a ve chvíli, kdy dané vlákno skončí, dostane další chunk

guided Na začátku každé vlákno dostane velký chunk a další jsou přidělovány dynamicky, ale velikost chunků se exponencionálně zmenšuje

Použití různého plánování ovlivňuje výslednou efektivitu programu. Pokud každá iterace trvá zhruba stejný čas, je možné použít statické plánování, při použití například dynamického plánování by mohlo naopak dojít ke zpomalení kvůli vyšší režii. Naopak pokud různé iterace trvají různou dobu, je výhodnější použít plánování dynamic nebo schedule.
Pro paralelizaci bloku pomocí direktivy *parallel* je potřeba splnit několik podmínek. Není možné, aby existovala možnost vyskočit nebo vskočit do paralelního bloku. Paralelní blok musí být v rámci jedné procedury. Před paralelním blokem může být pouze jedna podmínka, jeden výraz *num_threads*.

3.3 Vlastní realizace

Tato část popisuje implementaci metod pro řešení soustav lineárních intervalových rovnic, pro implementaci je použit programovací jazyk C++.

3.3.1 Implementace Gaussovy eliminace

Gaussova eliminace se skládá ze dvou částí. První část dopředný průchod upraví matici na tvar horní trojúhelníkové matice. Tato část se skládá ze tří do sebe vnořených cyklů, kde dva vnitřní cykly nezačínají od prvního prvku, ale jsou určeny prvním cyklem. Ve dvou vnitřních cyklech není žádná datová závislost, ale mezi iteracemi vnějšího cyklu už existuje datová závislost.
Algoritmus 4 popisuje intervalovou Gaussovu eliminaci. Všechny binární operace jsou intervalové binární operace, které jsou popsány v kapitole 1.3.

Algoritmus 4 Gaussova eliminace

```
1:  procedure GEM(A,b)
2:      for i ← 0, ..., n − 2 do
3:          for j ← i + 1, ..., n − 1 do
4:              ratio ← A[j, i]/A[i, i]
5:              for k ← j, ..., n − 1 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:
12:     for i ← n − 1, ..., 0 do
13:         for j ← n − 1, ..., i + 1 do
14:             b[i] ← b[i] − b[j] ∗ A[i, j]
15:         end for
16:         b[i] ← b[i]/A[i, i]
17:     end for
18: end procedure
```

3.3.2 Implementace Modifikované Gaussovy eliminace

Modifikovaná Gaussova eliminace používá stejný algoritmus jako běžná Gaussova elimace uvedená v algoritmu 4, kde je ale dělení nahrazeno dělením z modifikované intervalové aritmetiky uvedené v algoritmu 5. Pro výpočet modifikovaného dělení je potřeba 18 operací v plovoucí řádce.

Algoritmus 5 Modifikované dělení

```
1: procedure MODDIVISION(Interval a, Interval b)
2:     yMin ← 0.0
3:     yMax ← 0.0
4:     yMax ← ((a.up ∗ b.up)/a.low) − b.low
5:     yMin ← ((a.up ∗ b.up) − (a.low ∗ b.low))/(a.low + a.up)
6:     ret.low ← (b.up/a.low) − ((yMax − yMin)/(2 ∗ a.low))
7:     ret.up ← (b.low/a.up) + ((yMax + yMin)/(2 ∗ a.up))
8:     return ret
9: end procedure
```

3.3.3 Implementace metody HBR

Metodu HBR lze rozdělit na tři části. V první části se ze zadané intervalové matice **A** a intervalového vektoru ***b*** vypočítají reálné matice a vektory středních hodnot a průměru. Následně se nad těmito reálnými maticemi a vektory provádí výpočet. Ve třetí části je následně z vektorů $xLow, xHigh$ a matice Tm sestaven výsledný intervalový vektor.

Algoritmus 6 HBR

```
1:  procedure HBR(A, b)
2:      E ← {Jednotková matice}
3:      Tm ← {Nulová matice}
4:
5:      Amid ← mid(A)
6:      Arad ← rad(A)
7:      bmid ← mid(b)
8:      brad ← rad(b)
9:
10:     AmidInverse ← (Amid)^-1
11:     AmidInverseAbs ← abs(AmidInverse)
12:     M ← (E − AmidInverseAbs ∗ Arad)^-1
13:
14:     for j ← 0, . . . , n − 1 do
15:         m[i] ← M[i, i]
16:         Tm[i] ← M[i, i]
17:     end for
18:
19:     Tv ← (2 ∗ Tm − E)^-1
20:     xc ← AmidInverse ∗ bmid
21:     xcAbs ← abs(xc)
22:     ttt ← abs(AmidInverse ∗ brad)
23:     xStar ← M ∗ (xcAbs + ttt)
24:     xLow ← −1.0 ∗ xStar + Tm ∗ (xc + xcAbs)
25:     xHigh ← xStar + Tm ∗ (xc − xcAbs)
26:
27:     xLow ← min(xLow, Tv ∗ xLow)
28:     xHigh ← max(xHigh, Tv ∗ xHigh)
29:
30:     return IntervalVector(xLow, xHigh)
31: end procedure
```

Jak je z algoritmu 6 vidět, metoda HBR je pamětově velmi náročná. V průběhu algoritmu je potřeba mít v paměti osm reálných matic o velikosti dané

soustavy a devět vektorů.

3.3.4 Vektorizace Gaussovy eliminace

Při Gaussově eliminaci pro reálné soustavy lineárních rovnic je možno použít vektorizaci v nejvnitřnějším cyklu, kde se odečítá od j-tého řádku i-tý řádek. U soustav intervalových rovnic je potřeba pro vektorizaci udělat několik úprav. Pokud ukládáme intervalovou matici jako dvourozměrné pole, kde ve sloupcích $0, 2, \cdots, n \cdot 2$ jsou dolní meze intervalů a ve sloupcích $1, 3, \cdots, (n \cdot 2) + 1$. Pak odčítání i-tého řádku od j-tého znamená tedy

$$A_j = (\underline{a}_{(j,0)}, \overline{a}_{(j,1)}, \underline{a}_{(j,2)}, \overline{a}_{(j,2)}, \cdots, \underline{a}_{(j,n\cdot 2)}, \overline{a}_{(j,(n\cdot 2)+1)})$$

$$A_i = (\underline{a}_{(i,0)}, \overline{a}_{(i,1)}, \underline{a}_{(i,2)}, \overline{a}_{(i,2)}, \cdots, \underline{a}_{(i,n\cdot 2)}, \overline{a}_{(i,(n\cdot 2)+1)})$$

$$A_j - A_i = (\underline{a}_{(j,0)} - \overline{a}_{(i,1)}, \overline{a}_{(j,1)} - \underline{a}_{(i,0)}, \cdots, \underline{a}_{(j,n\cdot 2)} - \overline{a}_{(i,(n\cdot 2)+1)}, \overline{a}_{(j,(n\cdot 2)+1)} - \underline{a}_{(i,n\cdot 2)}).$$

Jinými slovy neodečítají se hodnoty v jednorozměrném poli, které mají stejný index, ale odečítají se křížem.
Tento problém lze vyřešit, když intervalovou matici ukládáme do dvou dvourozměrných polí $\underline{A}$ a $\overline{A}$, kde v $\underline{A_{i,j}}$ je uložena dolní mez intervalu $A_{i,j}$ a v v $\overline{A_{i,j}}$ je uložena horní mez intervalu $\overline{A_{i,j}}$. Pak při odečítání dvou řádků dostaneme

$$\underline{A_j} = (\underline{a}_{(j,0)}, \underline{a}_{(j,1)}, \cdots, \underline{a}_{(j,n)})$$

$$\overline{A_j} = (\overline{a}_{(j,0)}, \overline{a}_{(j,1)}, \cdots, \overline{a}_{(j,n)})$$

$$\underline{A_i} = (\underline{a}_{(i,0)}, \underline{a}_{(i,1)}, \cdots, \underline{a}_{(i,n)})$$

$$\overline{A_i} = (\overline{a}_{(i,0)}, \overline{a}_{(i,1)}, \cdots, \overline{a}_{(i,n)})$$

$$\underline{A_j - A_i} = (\underline{a}_{(j,0)} - \overline{a}_{(i,0)}, \underline{a}_{(j,1)} - \overline{a}_{(i,1)}, \cdots, \underline{a}_{(j,n)} - \overline{a}_{(i,n)})$$

$$\overline{A_j - A_i} = (\overline{a}_{(j,0)} - \underline{a}_{(i,0)}, \overline{a}_{(j,1)} - \underline{a}_{(i,1)}, \cdots, \overline{a}_{(j,n)} - \underline{a}_{(i,n)})$$

V tomto případě už lze použít vektorizace, protože při odečítání se používají hodnoty se stejným indexem.
Druhý problém je, že při Gaussově eliminaci se odečítá násobek i-tého řádku. Protože u soustav intervalových rovnic je tento násobek také interval, pak

$$[\underline{\alpha}, \overline{\alpha}] * [A_{i,k}] = [\min(\underline{\alpha}\cdot\underline{a_{i,k}}, \underline{\alpha}\cdot\overline{a_{i,k}}, \overline{\alpha}\cdot\underline{a_{i,k}}, \overline{\alpha}\cdot\overline{a_{i,k}}), \max(\underline{\alpha}\cdot\underline{a_{i,k}}, \underline{\alpha}\cdot\overline{a_{i,k}}, \overline{\alpha}\cdot\underline{a_{i,k}}, \overline{\alpha}\cdot\overline{a_{i,k}})].$$

Což nelze nijak zjednodušit. Jedním z možných řešení je vytvořit si čtyři pomocná pole, do kterých nejprve vypočteme všechny možné výsledky dolní a horní meze. Pak použijeme další dvě pomocná pole, do kterých uložíme minimální, respektive maximální hodnoty z předem vypočítaných možných hodnot násobení.
V případě modifikované Gaussovy eliminace nelze vektorizace použít, protože výpočet modifikovaného dělení se různě větví podle vstupních intervalů. Takovýto výpočet je tedy pro použití vektorizace příliš složitý.

Algoritmus 7 Odečtení násobku intervalového vektoru

```
1:  AA,AB,BA,BB,min,max
2:  for k ← i + 1, ..., n do
3:      AA[k] ← lowAlpha · lowAi[k]
4:      AB[k] ← lowAlpha · highAi[k]
5:      BA[k] ← highAlpha · lowAi[k]
6:      BB[k] ← highAlpha · highAi[k]
7:
8:      min[k] ← (AA[k] < AB[k]) ? AA[k] : AB[k]
9:      min[k] ← (BA[k] < min[k]) ? BA[k] : min[k]
10:     min[k] ← (AA[k] < min[k]) ? AA[k] : min[k]
11:
12:     max[k] ← (AA[k] > AB[k]) ? AA[k] : AB[k]
13:     max[k] ← (BA[k] > max[k]) ? BA[k] : max[k]
14:     max[k] ← (AA[k] > max[k]) ? AA[k] : max[k]
15:
16:     lowAj[k] ← lowAj[k] − max[k]
17:     highAj[k] ← highAj[k] − min[k]
18: end for
```

3.3.5 Paralelizace Gaussovy eliminace

V Gaussově eliminaci nelze paralelně zpracovávat nejvnější cyklus, kde se vytvářejí nuly v i-tém sloupci, protože vytváření nul v i-tém sloupci modifikuje celou čtvercovou podmatici, která má levý horní roh v $A_{i,i}$ a dolní v $A_{n,n}$. Proto než začneme vytvářen nuly v $i + 1$ sloupci musíme dokončit i-tou iteraci.
Můžeme ale paralelizovat část, kde se odčítá násobek i-tého řádku od j-tého řádku, protože mezi těmito operacemi není žádná datová závislost. Po odečtení od všech řádku je potřeba počkat na všechna vlákna a až následně začít další iteraci. Dopředná část intervalové Gaussovy eliminace s použitím knihovny OpenMP je popsána v algoritmu 8. Direktiva „#pragma omp parallel“ určuje, že se před vstoupení do cyklu vytvoří vlákna, ale tento cyklus se neparalelizuje. Až direktiva „#pragma omp for“ určuje cyklus, jehož iterace se rozdělí pro různá vlákna. Díky tomu zlepšíme efektivitu programu, protože vytváření vláken zabere určitý čas. Pokud bychom vlákna vytvářeli až před cyklem začínajícím na pátém řádku, museli bychom tato vlákna v průběhu běhu programu vytvořit celkově n-krát. Výše uvedeným způsobem ale dojde pouze k jednomu vytvoření všech vláken a před cyklem na pátém řádku dojde pouze k rozdělení práce pro už vytvořená vlákna.
Protože každý řádek je stejně velký, je možné použít plánování static, kde dojde k rovnoměrnému rozdělení počtu řádků ke zpracování. Složitější plánování by v tompto případě bylo zbytečné. Proměnné deklarované uvnitř paralelního

Algoritmus 8 Dopředná část Gaussovy eliminace v OpenMP

```
1: #pragma omp parallel
2: for (int i=0; i<N; i++) {
3:    Pivotace
4:    #pragma omp for schedule (static)
5:    for (int j=i+1; i<N; i++) {
6:       Interval ratio = A[j][i] / A[i][i]
7:       for (int k=i+1; i<N; i++) {
8:          Odečtení násobku i-tého řádku od j-tého jako v 7 agloritmu.
9:       }
10:      b[j] = b[j] - ratio * b[i]
11:      A[j][i] = 0
12:   }
13: }
```

bloku, matice **A** a vektor ***b*** jsou sdílená data. Protože v cyklu začínajícím na pátém řádku nejsou žádné datové závislosti, nemůže se tak stát, že by se dvě vlákna snažila zapsat na jedno místo ve sdílených proměnných. To znamená, že není potřeba při výpočtu použít nějaké synchronizační funkce, které by paralelní výpočet zpomalovaly.

3.3.6 Pivotace v OpenMP

Při pivotaci se každému vláknu přidělí část sloupce, respektive řádku, ve kterém každé vlákno najde pivot pro danou část. Po tom co každé vlákno najde kandidáta na pivot, se ve druhém cyklu už sekvenčně vybere pivot z těchto kandidátů. Bohužel v intervalové aritmetice nelze použít vestavěná funkce OpenMP pro redukci maximální hodnoty, protože kromě hodnoty pivotu potřebujeme znát i její index.

3.3.7 Paralelizce metody HBR

Metodu HBR lze rozdělit na tři části. V první části se z intervalové matice **A** musí vypočítat matice středních hodnot, matice poloměrů a z intervalového vektoru ***b*** se musí vytvořit vektor středních hodnot a vektor poloměrů. V druhé části se provádí operace nad reálnými maticemi a vektory. Ve třetí části se z těchto reálných matic a reálných vektorů vytvoří výsledný intervalový vektor. Pro druhou část, která pracuje s reálnými vektory a maticemi, je použita knihovna Armadillo. První a třetí část je paralelizována pomocí knihovny OpenMP.

Algoritmus 9 Pivotace v OpenMP

```
1: #pragma omp for schedule (static) firstprivate(pivotRow)
2: for (int j = i; j < n; j++) {
3: int tid = omp _get _thread _num()
4:    float curAbs = Abs(A[j][i])
5:    if (curAbs > pivot) {
6:       pivots[tid] = curAbs
7:       pivotRows[tid] = j
8:    }
9: }
10: for(int j=0;j<omp _get _num _threads();j++) {
11:    if (pivots[j] > pivot) {
12:       pivot = pivots[j]
13:       pivotRow = pivotsRows[j]
14:    }
15: }
```

3.3.8 Program pro řešení soustavy lineárních intervalových rovnic

Řešení obsahuje dva programy. První generuje soustavy lineárních intervalových rovnic s požadovanými vlastnostmi. Druhý program řeší soustavu lineárních intervalových rovnice.
Program pro řešení soustavy lineárních intervalových rovnic nejdříve ze souboru načte matici $\mathbf{A}$ v požadovaném tvaru a poté vektor $\boldsymbol{b}$, který odpovídá $\boldsymbol{b} = \mathbf{A}\boldsymbol{x}$. Následně provede jednotlivé metody popsané v tabulce 3.1 a na standardní výstup vypíše požadované hodnoty.
Programy určený pro řešení soustavy intervalových rovnic se spouští s pěti argumenty. První odpovídá souboru, ze kterého je soustava čtena, druhý použité metodě, třetí určuje, jestli bude použito předpodmínění, čtvrtý určuje výpis programu a patý je počet vláken použitých pro výpočet. Argumenty jsou popsány v tabulce 3.1 a 3.2.

Tabulka 3.1: První argument pro spuštění řešiče

2. argument	Metoda
A	Gaussova eliminace
B	Gaussova eliminace s řádkovou pivotací
C	Gaussova eliminace se sloupcovou pivotací
D	Modifikovaná Gaussova eliminace
E	Modifikovaná Gaussova eliminace s řádkovou pivotací
F	Modifikovaná Gaussova eliminace se sloupcovou pivotací
G	Metoda HBR
H	Paralelní Gaussova eliminace
I	Paralelní GEM s vektorizací
J	Paralelní metoda HBR

Tabulka 3.2: Druhý argument pro spuštění řešiče

3. Argument	Předpodmínění	4. Argument	Výstup
F	Bez předpodmínění	A	Řešení
T	S předpodmínění	B	Průměrná šířka
-	-	C	Doba běhu

Kapitola 4

Měření šířek výsledných intervalů a výkonosti

Tato kapitola se věnuje měření šířek výsledných intervalů pro různé lineární intervalové soustavy, porovnává výkonnost jednotlivých metod a porovnává výkonost různých optimalizací.

4.1 Měření výsledných šířek

Šířky výsledných intervalů budeme měřit na náhodně generované soustavě rovnic $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ s určitými parametry. V první části této kapitoly jsou uvedeny příklady malých soustav spolu s výslednými vektory pro dané metody a analýzou výsledných vektorů. Při analýze výsledných vektorů jsou uvedeny hodnoty minimální šířky, maximální šířky, průměrné šířky a index relativní určitosti intervalu (RUI). RUI je spočítána jako průměrná šířka výsledných intervalů vydělena průměrnou absolutní hodnotou koeficientů v matici $\mathbf{A}$ pro danou proměnnou.

$$RUI(x) = \sum_{i=1}^{n} \frac{wid(x)}{q}, \text{ kde } q = \sum_{j=1}^{n} \frac{A_{j,i}}{n}$$

V druhé části této kapitoly jsou uvedeny výsledky šířek pro matice o velikosti tisíců proměnných. Soustavy rovnic $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ jsou opět náhodně generovány s určitými parametry.
Při generování intervalové matice $\mathbf{A}$ nejdříve vygenerujem náhodnou horní trojúhelníkovou matici $\mathbf{U}$, jejíž hodnoty jsou předem určeny minimální a maximální možnou hodnotou. Tím je zajištěno, že daná matice je regulární. Následně ke každému řádku přičteme náhodný násobek vyššího řádku, tak aby matice neobsahovala nulovou hodnotu v dolní trojúhelníkové matici a dostaneme matici $\mathbf{A_c}$. Následně vygenerujeme poměr (ratio), který je určený pře-

dem danými parametry a spočtáme danou deltu

$$\Delta_{i,j} = \text{ratio}_{i,j} \cdot abs(A_{c_{i,j}}).$$

Na základě této delty vytvoříme intervalovou matici

$$A_{i,j} = [A_{c_{i,j}} - \Delta_{i,j}, A_{c_{i,j}} + \Delta_{i,j}]$$

4.1.1 Měření výsledných šířek pro malé matice

Příklad 4.1. V prvním příkladu máme soustavu s pěti proměnnými. Minimální možná střední hodnota je 5 maximální 250. Poloměr je od 0 % do 3 % vzhledem k dané střední hodnotě.

$$A = \begin{pmatrix} [14.71, 14.74] & [27.42, 27.58] & [17.89, 17.94] & [12.06, 12.07] & [30.49, 30.70] \\ [9.21, 9.25] & [40.24, 40.64] & [50.64, 50.83] & [28.65, 28.66] & [29.78, 30.06] \\ [7.00, 7.03] & [36.51, 36.54] & [75.51, 75.96] & [44.50, 44.91] & [44.43, 44.65] \\ [10.16, 10.25] & [40.83, 41.17] & [74.62, 74.93] & [72.02, 72.05] & [94.88, 95.72] \\ [12.21, 12.22] & [28.19, 28.36] & [73.07, 73.34] & [87.97, 88.71] & [242.37, 243.23] \end{pmatrix}$$

$$b = \begin{pmatrix} [259.24, 260.62] \\ [209.53, 210.30] \\ [207.09, 207.84] \\ [321.29, 323.68] \\ [295.95, 297.90] \end{pmatrix}$$

Tabulka 4.1: Výsledky pro soustavu z příkladu 4.1

Bez předpodmínění		
Gaussova eliminace	**GEM s pivotací**	**Modifikovaná GEM**
$\begin{pmatrix} [0.67, 33.93] \\ [-5.90, 5.85] \\ [-2.93, 1.25] \\ [2.32, 6.41] \\ [-1.67, -0.39] \end{pmatrix}$	$\begin{pmatrix} [-104.03, 153.20] \\ [-59.08, 50.36] \\ [-15.86, 15.30] \\ [2.14, 11.15] \\ [-3.42, -0.28] \end{pmatrix}$	$\begin{pmatrix} [13.68, 20.77] \\ [-1.38, 1.13] \\ [-0.99, -0.08] \\ [3.40, 4.21] \\ [-0.98, -0.74] \end{pmatrix}$
Modifikovaná GEM	**MGEM s pivotací**	**HBR**
$\begin{pmatrix} [13.68, 20.77] \\ [-1.38, 1.13] \\ [-0.99, -0.08] \\ [3.40, 4.21] \\ [-0.98, -0.74] \end{pmatrix}$	$\begin{pmatrix} [8.99, 25.89] \\ [-3.87, 3.31] \\ [-1.49, 0.53] \\ [3.50, 4.09] \\ [-0.95, -0.77] \end{pmatrix}$	$\begin{pmatrix} [17.04, 17.71] \\ [-0.32, 0.068] \\ [-0.70, -0.39] \\ [3.66, 4.021] \\ [-0.91, -0.80] \end{pmatrix}$

Z výsledků bez přepodmínění je vidět při použití Gaussovy eliminace, že rozdíly ve výsledných šířkách x_{GEM} jsou poměrně velké i když v původní matici **A** a vektoru $\boldsymbol{b}$ je poměrně konzistentní. Maximální šířka je více než 3.5 krát větší než průměrná šířka.
V případě $\boldsymbol{x}_{\text{GEMP}}$ došlo k velkému zhoršení. Výslednou šířku lze sice zlepšit používáním pivotace, ale také záleží na šířce intervalu, kterým dělíme. Také záleží na tom, jaké šířky má řádek, který odečítáme.

Tabulka 4.2: Vlastnosti výsledných vektorů z 4.1

Bez předpodmínění					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
Minimální sířka	1.27	3.14	0.23	0.17	0.11
Maximální sířka	33.25	257.23	7.09	16.89	0.67
Pruměrná sířka	10.09	81.10	2.31	5.37	0.30
RUI	36.14%	279.35%	7.69%	18.33%	0.87%

Při použití Modifikované Gaussovy eliminace dojde poměrně k velkému zmenšení výsledných šířek. Průměrná šířka je čtyřikrát menší, stejně tak index RUI je zhruba pětkrát menší. Výrazně nejlepší výsledky jsme dosáhli při použití metody HBR.

Tabulka 4.3: Výsledky pro soustavu z příkladu 4.1 po předpodmínění

S předpodmíněním		
Gaussova eliminace	**GEM s pivotací**	**Modifikovaná GEM**
$\begin{pmatrix} [16.59, 17.76] \\ [-0.42, 0.17] \\ [-0.75, -0.30] \\ [3.50, 4.04] \\ [-0.93, -0.75] \end{pmatrix}$	$\begin{pmatrix} [16.59, 17.76] \\ [-0.42, 0.17] \\ [-0.75, -0.30] \\ [3.50, 4.04] \\ [-0.93, -0.75] \end{pmatrix}$	$\begin{pmatrix} [16.59, 17.76] \\ [-0.39, 0.15] \\ [-0.73, -0.32] \\ [3.52, 4.02] \\ [-0.92, -0.76] \end{pmatrix}$
Modifikovaná GEM	**MGEM s pivotací**	**HBR**
$\begin{pmatrix} [16.59, 17.76] \\ [-0.39, 0.15] \\ [-0.73, -0.32] \\ [3.52, 4.02] \\ [-0.92, -0.76] \end{pmatrix}$	$\begin{pmatrix} [16.59, 17.76] \\ [-0.39, 0.15] \\ [-0.73, -0.32] \\ [3.52, 4.02] \\ [-0.92, -0.76] \end{pmatrix}$	$\begin{pmatrix} [16.86, 17.89] \\ [-0.42, 0.17] \\ [-0.76, -0.32] \\ [3.59, 4.10] \\ [-0.94, -0.77] \end{pmatrix}$

Tabulka 4.4: Vlastnosti výsledných vektorů z 4.1 po předpodmínění

S předpodmíněním					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
Minimální šířka	0.18	0.18	0.16	0.16	0.17
Maximální šířka	1.16	1.16	1.16	1.16	1.04
Průměrná šířka	0.58	0.58	0.56	0.56	0.55
RUI	1.46%	1.46%	1.43%	1.43%	1.35%

Při použití předpodmínění jsme dostali výrazně lepší výsledky. Pivotace nemá na výsledky vliv, protože po pivotaci jsou na diagonále největší hodnoty. Výsledné šířky jsou velmi podobné, přesto nejhorší výsledky dostáváme

při použití Gaussovy eliminace. Výsledky modifikované Gaussovy eliminace a HBR jsou velmi podobné.

Příklad 4.2. V druhém příkladu jsme upravili původní matici **A** z příkladu 4.1. Šířka intervalu $A_{2,2}$ je šestkrát větší než v původní matici **A**. Vektor $\boldsymbol{b}$ zůstává stejný jako v příkladě 4.1.

$$A = \begin{pmatrix} [14.71, 14.74] & [27.42, 27.58] & [17.89, 17.94] & [12.06, 12.07] & [30.49, 30.70] \\ [9.21, 9.25] & [38.24, 40.64] & [50.64, 50.83] & [28.65, 28.66] & [29.78, 30.06] \\ [7.00, 7.03] & [36.51, 36.54] & [75.51, 75.96] & [44.50, 44.91] & [44.43, 44.65] \\ [10.16, 10.25] & [40.83, 41.17] & [74.62, 74.93] & [72.02, 72.05] & [94.88, 95.72] \\ [12.21, 12.22] & [28.19, 28.36] & [73.07, 73.34] & [87.97, 88.71] & [242.37, 243.23] \end{pmatrix}$$

Tabulka 4.5: Vlastnosti výsledných vektorů z 4.2

Bez předpodmínění					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
Minimální šířka	6.01	0.00	0.26	0.18	0.13
Maximální šířka	176.06	0.00	8.63	17.08	0.89
Průměrná šířka	57.56	0.00	2.80	5.43	0.47
RUI	191.41%	0.00%	9.37%	18.54%	1.15%

Tabulka 4.6: Vlastnosti výsledných vektorů z 4.2 po předpodmínění

S předpodmíněním					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
Minimální šířka	0.21	0.21	0.17	0.17	0.20
Maximální šířka	1.22	1.22	1.50	1.50	1.37
Průměrná šířka	0.71	0.71	0.67	0.67	0.70
RUI	1.87%	1.87%	1.81%	1.81%	1.76%

Touto změnou dojde při vytváření nul ve druhé sloupci u Gaussovy eliminace s pivotací k prohození druhého řádku se čtvrtým řádkem. V tomto případě je při vytváření nul ve druhé sloupci rozdíl v šířce intervalu násobku druhého řádku. Při Gaussově eliminaci bez pivotace je druhý řádek násoben intervalem $[36.51, 36.54]/[38.24, 40.64] = [0.89, 0.96]$. Při Gaussově eliminaci s pivotací je druhý řádek násoben intervalem $[36.51, 36.54]/[40.83, 41.17] = [0.89, 0.90]$.
Přesto, při použití Gaussovy eliminace s pivotací, dojde k takovému rozšíření intervalů, že se soustavu nepodaří vyřešit. Modifikovaná Gaussova eliminace dává výrazně lepší výsledky. Průměrná šířka u Modifikované Gaussovy eliminace je zhruba dvacetkrát lepší než u běžné Gaussovy eliminace. Výrazně nejlepší výsledky dává použití metody HBR, kde průměrná šířka je zhruba šestkrát menší než u druhého nejlepšího výsledku.

Při použití předpodmínění se výsledné šířky u všech metod vyrovnají. Pivotace nemá znovu vliv z důvodů uvedených v prvním příkladě. Nejlepší výsledky znovu dává metoda HBR i když oproti výsledku bez předpodmínění dojde ke zhoršení. Je to jediná metoda, u které došlo ke zhoršení u použití přepodmínění.

Příklad 4.3. V dalším příkladu jsme znovu upravili původní matici **A** z příkladu 4.1. Nyní jsme prohodili druhý řádek se čtvrtým a následně intervaly $A_{2,2}$ a $A_{4,2}$. Vektor $\boldsymbol{b}$ zůstává stejný

$$A = \begin{pmatrix} [14.71, 14.74] & [27.42, 27.58] & [17.89, 17.94] & [12.06, 12.07] & [30.49, 30.70] \\ [10.16, 10.25] & [40.24, 40.64] & [74.62, 74.93] & [72.02, 72.05] & [94.88, 95.72] \\ [7.00, 7.03] & [36.51, 36.54] & [75.51, 75.96] & [44.50, 44.91] & [44.43, 44.65] \\ [9.21, 9.25] & [40.83, 41.17] & [50.64, 50.83] & [28.65, 28.66] & [29.78, 30.06] \\ [12.21, 12.22] & [28.19, 28.36] & [73.07, 73.34] & [87.97, 88.71] & [242.37, 243.23] \end{pmatrix}$$

Tabulka 4.7: Vlastnosti výsledných vektorů z 4.3

Bez předpodmínění					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
Minimální šířka	0.00	2.72	0.00	0.18	0.28
Maximální šířka	0.00	210.19	0.00	18.23	1.96
Průměrná šířka	0.00	68.77	0.00	5.93	0.91
RUI	0.00%	228.72%	0.00%	19.81%	2.38%

Tabulka 4.8: Vlastnosti výsledných vektorů z 4.3 po předpodmínění

S předpodmíněním					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
Minimální šířka	0.34	0.34	0.28	0.28	0.32
Maximální šířka	2.51	2.51	2.50	2.50	2.52
Průměrná šířka	1.18	1.18	1.12	1.12	1.12
RUI	3.09%	3.09%	3.02%	3.02%	3.01%

V tomto případě znovu dojde při použití Gaussovy eliminace s pivotací k prohození druhého a čtvrtého řádku. Rozdíl oproti minulému příkladu je v tom, že čtvrtý řádek obsahuje výrazně menší hodnoty, než druhý řádek. Při odečítání násobku druhého řádku od ostatních se při použití pivotace odečítá vektor s menšími šířkami.
Bez použití předpodmínění nám Gaussova eliminace a Modifikovaná Gaussova eliminace nedává žádné výsledky, protože postupně dojde k takovému rozšíření intervalů v matici **A**, že dojde k dělení intervalem, který obsahuje nulu. Gaussova eliminace s pivotací nám sice dává nějaké výsledky, ale vzhledem k šířkám výsledného vektoru je takové řešení téměř nepoužitelné. Zatímco Modifikonvaná Gaussova eliminace nám dává výsledky výrazně lepší (zhruba

desetkrát lepší oproti Gaussově eliminaci s pivotací). Nejlépe znovu vychází metoda HBR. Matice **A** obsahuje intervaly, které mají šířku mezi 0 % a 3 %. Průměrná šířka výsledného vektoru $\boldsymbol{x}_{\mathrm{HBR}}$, vzhledem k středním hodnotám matice A je 2.38 %. To znamená, že po vyřešení metodou HBR nedošlo k nějakému zvětšení šířek intervalů.
Při použití předpodmínění došlo jak u Gaussovy eliminace, tak u Modifikované Gaussovy eliminace k výraznému zlepšení. Naopak u metody HBR se výsledné šířky zhoršily, ale nijak výrazně a pořád je tato metoda nejlepší.

4.1.2 Měření výsledných šířek pro velké matice

V této části jsou porovnány výsledky měření pro různě velké matice s různými parametry střední hodnoty a poloměrů matice.
V prvním případě jsou testovány metody na regulární matici **A**, jejíž střední hodnoty jsou z intervalu $[100, 1000]$ a poloměr intervalů v matici je mezi 0 % až 1 % ze střední hodnoty. Výsledky jsou uvedeny v tabulkách 4.9 a 4.10.
Pokud není použito předpodmínění pak Gaussova eliminace, Gaussova eli-

Tabulka 4.9: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [100, 1000]$ a $\mathrm{rad}([\mathbf{A}]) = [0, \mathbf{A_c}/100]$

Bez předpodmínění					
N	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
10	0.00%	0.00%	0.00%	22.80%	0.50%
20	0.00%	0.00%	0.00%	0.00%	0.27%
30	0.00%	0.00%	0.00%	0.00%	0.81%
40	0.00%	0.00%	0.00%	0.00%	3.66%
50	0.00%	0.00%	0.00%	0.00%	3.78%
60	0.00%	0.00%	0.00%	0.00%	5.30%
70	0.00%	0.00%	0.00%	0.00%	7.92%
80	0.00%	0.00%	0.00%	0.00%	19.04%
90	0.00%	0.00%	0.00%	0.00%	21.04%
100	0.00%	0.00%	0.00%	0.00%	38.40%

mace s pivotací a Modifikovaná Gaussova eliminace nedává žádně výsledy. Modifikovaná Gaussova eliminace s pivotací je schopna spočítat výsledek pouze pro matici o deseti proměnných. Metoda HBR dává výsledky pro všechny soustavy rovnic. Z výsledků pro různě velké soustavy je vidět, že šířka výsledného vektoru roste spolu s velikostí soustav. Relativní určitost intervalu pro soustavu o stu proměnných je zhruba 77 krát větší než pro soustavu o deseti proměnných.
Při použití předpodmínění dávají všechny metody nějaké řešení. Pro všechny metody platí, že s rostoucí velikostí soustav je také větší relativní určitost výsledných intervalů a to poměrně výrazně. Ve všech případech měření je nejlepší Modifikovaná Gaussova eliminace. Klasická Gaussova eliminace dává podobné

Tabulka 4.10: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [100, 1000]$ a $\mathrm{rad}([\mathbf{A}]) = [0, \mathbf{A_c}/100]$

S předpodmíněním					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
10	0.84%	0.84%	0.78%	0.78%	0.84%
20	0.45%	0.45%	0.42%	0.42%	0.45%
30	1.31%	1.31%	1.21%	1.21%	1.30%
40	6.98%	6.98%	6.39%	6.39%	7.00%
50	6.77%	6.77%	6.21%	6.21%	6.78%
60	9.83%	9.83%	8.97%	8.97%	9.84%
70	13.46%	13.46%	12.19%	12.19%	13.49%
80	32.77%	32.77%	28.45%	28.45%	33.10%
90	35.63%	35.63%	31.25%	31.25%	35.83%
100	65.56%	65.56%	55.28%	55.28%	66.20%

výsledky jako metoda HBR.
Pokud porovnáme Modifikovanou Gaussovu eliminaci s předpodmíněním s metodou HBR bez předpodmínění, pak lepší výsledky dává metoda HBR. V průměru je Modifikovaná Gaussova eliminace o 1.66 % horší než metoda HBR. Ve druhém měření zůstavají parametry soustavy rovnic stejné jako v předchozím případě kromě možného poloměru intervalů, který teď může být třikrát větší oproti předchozímu příkladu. Naměřené hodnoty jsou uvedeny v tabulkách 4.11 a 4.12.

Tabulka 4.11: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [100, 1000]$ a $\mathrm{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.03]$

Bez předpodmínění					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
10	0.00%	0.00%	0.00%	86.58%	1.84%
20	0.00%	0.00%	0.00%	0.00%	0.85%
30	0.00%	0.00%	0.00%	0.00%	2.69%
40	0.00%	0.00%	0.00%	0.00%	17.47%
50	0.00%	0.00%	0.00%	0.00%	16.74%
60	0.00%	0.00%	0.00%	0.00%	25.73%
70	0.00%	0.00%	0.00%	0.00%	42.09%
80	0.00%	0.00%	0.00%	0.00%	45.37%
90	0.00%	0.00%	0.00%	0.00%	93.25%
100	0.00%	0.00%	0.00%	0.00%	229.54%

V případě bez předpodmínění Gaussova eliminace, Gaussova elimace s pivotací a Modifikovaná Gaussova eliminace opět nedává žádně výsledky. Modifikovaná Gaussova eliminace dává znovu výsledky pouze pro matici o deseti

Tabulka 4.12: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [100, 1000]$ a $\text{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.03]$

S předpodmíněním					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
10	3.01%	3.01%	2.69%	2.69%	3.09%
20	1.42%	1.42%	1.31%	1.31%	1.41%
30	4.35%	4.35%	3.96%	3.96%	4.35%
40	32.70%	32.70%	27.01%	27.01%	33.51%
50	29.68%	29.68%	25.14%	25.14%	30.12%
60	47.21%	47.21%	38.70%	38.70%	47.93%
70	70.78%	70.78%	55.91%	55.91%	72.10%
80	0.00%	0.00%	327.83%	327.83%	80.80%
90	1138.95%	1138.95%	288.30%	205.30%	1366.42%
100	0.00%	0.00%	5772.46%	5772.46%	389.03%

proměnných. Metoda HBR dává výsledky pro všechny soustavy rovnic. Šířka výsledného vektoru opět roste spolu s velikostí soustav. V porovnání s minulým měřením jsou výsledné šířky zhruba třikrát větší.

S metodami, které používají předpodmínění, dostáváme téměř od všech metod výsledky. Gaussova elimice bez i s pivotací pro soustavy o velikosti 80 a 100 proměnných nedává žádné výsledky. V některých případech má větší soustava menší míru relativní určitosti než nějaká menší soustava. Například soustava o velikosti 80 v případě Modifikované Gaussovy eliminace je RUI 327.83 %, zatímco pro soustavu o 90 proměnných je RUI 205.30 %. Zároveň u soustavy o 90 proměnných dává lepší výsledky Modifikovaná Gaussova eliminace s pivotací. Ve většině případů je nejlepší Modifikovaná Gaussova eliminace, v některých případech je lepší metoda HBR.

Celkově nejlepší výsledky dává ve všech případech metoda HBR. Při porovnání s Modifikovanou Gaussovou eliminací se dá říci, že výsledné šířky nejsou tak rozdílné, kromě případů soustav o 80 a 100 proměnných, kde se výsledky MGEM výrazně zhorší.

V porovnání s předchozím příkladem je vidět, že nejlepší výsledné šířky jsou zhruba třikrát horší než v předchozím příkladu, což odpovídá tomu, že i poloměr soustavy je třikrát větší. Z toho tedy vyplývá, že šířky intervalů soustavy mají vliv na výsledné šířky. Navíc toto zvětšení poloměrů má za následek to, že v některých případech se soustava rovnic nepovede vůbec vyřešit a v některých případech se výsledná šířka nepřiměřeně zvětší.

V dalších dvou příkladech budeme používat soustavy, jejichž střední hodnoty jsou větší než v předchozích dvou případech. Následující měření je tedy provedeno pro soustavy, jejichž střední hodnoty jsou z intervalu $[500, 1000]$ a poloměr intervalů v matici je mezi 0 % až 1 % ze střední hodnoty. Výsledky jsou uvedeny v tabulkách 4.13 a 4.14.

U metod bez předpodmínění jako jediná metoda HBR dává výsledky pro

Tabulka 4.13: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [500, 1000]$ a $\mathrm{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.01]$

Bez předpodmínění					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
10	0.00%	66.88%	0.00%	4.46%	0.06%
20	0.00%	0.00%	0.00%	3554.77%	0.40%
30	0.00%	0.00%	0.00%	0.00%	0.48%
40	0.00%	0.00%	0.00%	0.00%	1.44%
50	0.00%	0.00%	0.00%	0.00%	3.33%
60	0.00%	0.00%	0.00%	0.00%	4.80%
70	0.00%	0.00%	0.00%	0.00%	35.70%
80	0.00%	0.00%	0.00%	0.00%	10.02%
90	0.00%	0.00%	0.00%	0.00%	24.80%
100	0.00%	0.00%	0.00%	0.00%	29.82%

Tabulka 4.14: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [500, 1000]$ a $\mathrm{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.01]$

S předpodmíněním					
	$\boldsymbol{x}_{\mathrm{GEM}}$	$\boldsymbol{x}_{\mathrm{GEMP}}$	$\boldsymbol{x}_{\mathrm{MGEM}}$	$\boldsymbol{x}_{\mathrm{MGEMP}}$	$\boldsymbol{x}_{\mathrm{HBR}}$
10	0.09%	0.09%	0.09%	0.09%	0.09%
20	0.64%	0.64%	0.60%	0.60%	0.64%
30	0.80%	0.80%	0.74%	0.74%	0.79%
40	2.64%	2.64%	2.46%	2.46%	2.63%
50	6.07%	6.07%	5.56%	5.56%	6.07%
60	8.50%	8.50%	7.80%	7.80%	8.51%
70	68.79%	68.79%	54.81%	54.81%	70.39%
80	17.27%	17.27%	15.57%	15.57%	17.33%
90	45.12%	45.12%	38.90%	38.90%	45.51%
100	53.43%	53.43%	45.90%	45.90%	53.83%

všechny soustavy rovnic. Metody používající předpodmínění dávají výsledky pro všechny soustavy. Ve všech případech nejlépe vychází použití Modifikované Gaussovy eliminace.

Celkově ale nejlepší výsledky znovu dává ve všech případech metoda HBR. V porovnání s prvním měřením nelze říci, že by výsledné RUI bylo obecně horší nebo lepší. V tomto případě se tedy dá říct, že výsledné šířky nezávisí na velikosti střední hodnoty.

V dalším měření ponecháme parametry soustavy rovnic stejné jako v předchozím případě kromě možného poloměru intervalů, který teď může být třikrát větší oproti předchozímu příkladu. Naměřené hodnoty jsou uvedeny v tabulkách 4.15 a 4.16.

Výsledky jsou podobné předchozím měřením. Celkově nejlepší výsledky dává metoda HBR bez předpodmínění. Ostatní metody až na jednu výjmku nejsou

Tabulka 4.15: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [500, 1000]$ a $\text{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.03]$

Bez předpodmínění					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
10	0.00%	0.00%	0.00%	13.58%	0.17%
20	0.00%	0.00%	0.00%	0.00%	1.28%
30	0.00%	0.00%	0.00%	0.00%	1.54%
40	0.00%	0.00%	0.00%	0.00%	8.08%
50	0.00%	0.00%	0.00%	0.00%	21.19%
60	0.00%	0.00%	0.00%	0.00%	10.85%
70	0.00%	0.00%	0.00%	0.00%	40.99%
80	0.00%	0.00%	0.00%	0.00%	106.05%
90	0.00%	0.00%	0.00%	0.00%	83.30%
100	0.00%	0.00%	0.00%	0.00%	80.19%

Tabulka 4.16: RUI pro $[\mathbf{A}]$, kde $\mathbf{A_c} \in [500, 1000]$ a $\text{rad}([\mathbf{A}]) = [0, \mathbf{A_c} \cdot 0.03]$

S předpodmíněním					
	$\boldsymbol{x}_{\text{GEM}}$	$\boldsymbol{x}_{\text{GEMP}}$	$\boldsymbol{x}_{\text{MGEM}}$	$\boldsymbol{x}_{\text{MGEMP}}$	$\boldsymbol{x}_{\text{HBR}}$
10	0.29%	0.29%	0.27%	0.27%	0.28%
20	2.06%	2.06%	1.92%	1.92%	2.06%
30	2.55%	2.55%	2.35%	2.35%	2.55%
40	14.94%	14.94%	13.18%	13.18%	15.11%
50	39.83%	39.83%	31.17%	31.17%	41.91%
60	19.18%	19.18%	17.20%	17.20%	19.28%
70	0.00%	0.00%	0.00%	0.00%	81.00%
80	192.11%	192.11%	116.19%	116.19%	205.50%
90	405.13%	405.13%	201.39%	201.39%	168.07%
100	153.08%	153.08%	110.61%	110.61%	156.47%

schopny soustavy bez předpodmínění vyřešit. V případě předpodmínění dávají všechny metody nějaké výsledky mimo soustavy o velikosti $n = 70$.

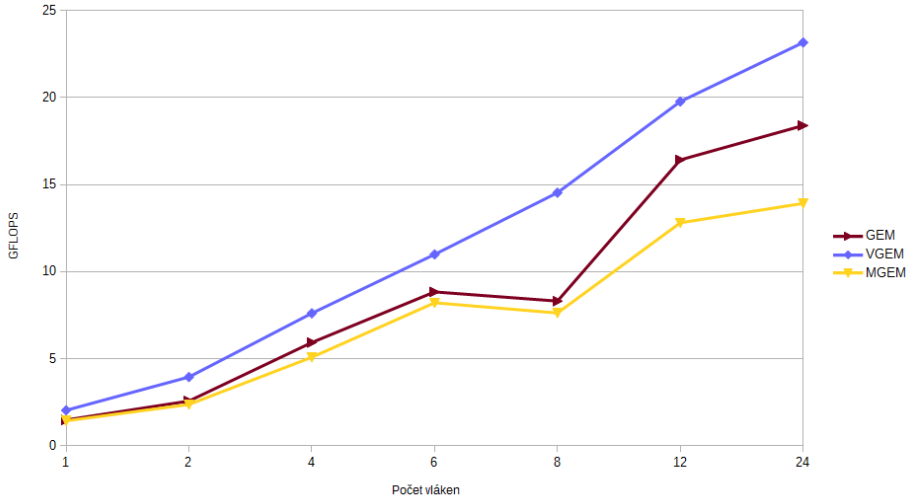
4.2 Shrnutí měření výsledných šířek

Z výše uvedeného měření plyne, že nejlepší výsledky dostáváme při použití metody HBR bez předpodmínění. V některých případech s pomocí Modifikované Gaussovy eliminace s pivotací dostáváme v některých případech podobné výsledky, pokud použijeme předpodmínění.
Co se týče vlivu velikosti poloměrů intervalů a velikosti středních hodnot soustavy rovnic, tak s rostoucí velikostí poloměrů roste také velikost poloměrů výsledných šířek. Dále nelze určit, že by velikost středních hodnot měla vliv na šířku výsledných vektorů

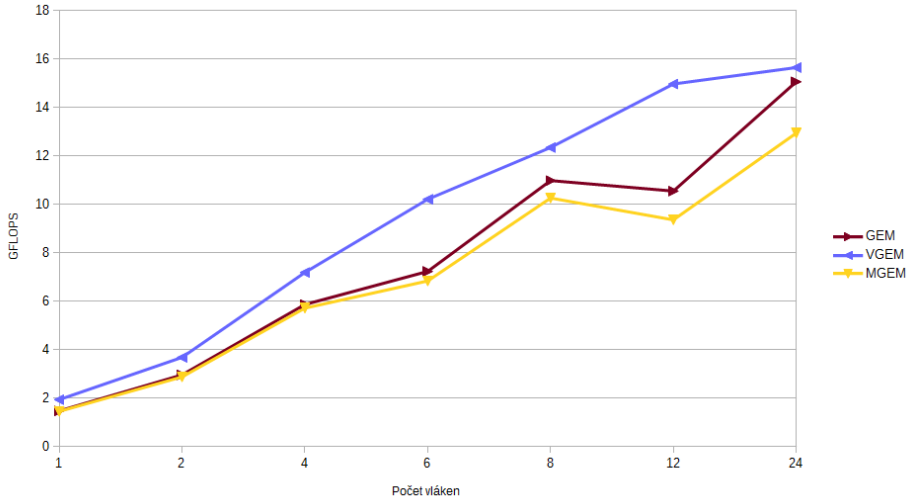
4.3 Měření výkonnosti

Měření výkonnosti probíhalo na svazku Star na uzlu gpu-01. Procesor Intel Xeon 6core 2620 v2 @ 2.1Ghz a 32 GB operační paměti.
V první části měření porovnáváme výkonnost měřenou v GFLOPS (počet operací v plovoucí řádce provedených za jednu sekundu) pro různý počet procesorů.

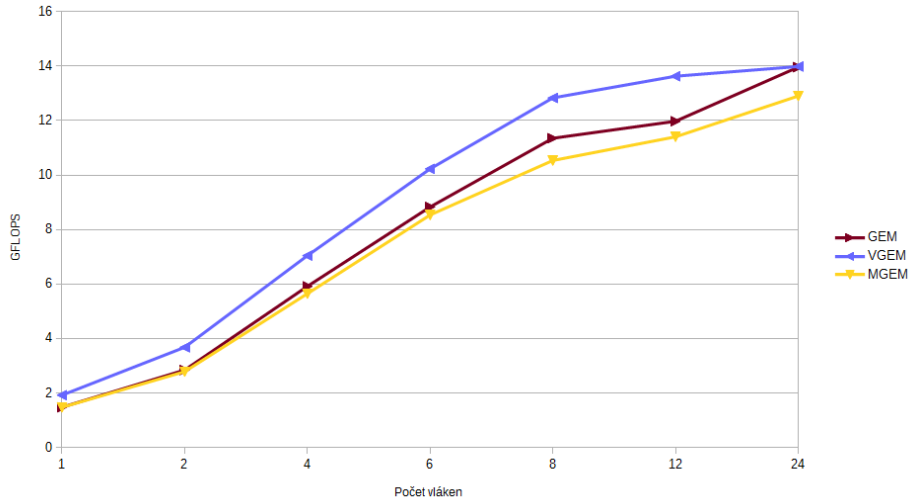
Obrázek 4.1, 4.2 a 4.3 porovnává výkonnost Gaussovy eliminace s vek-



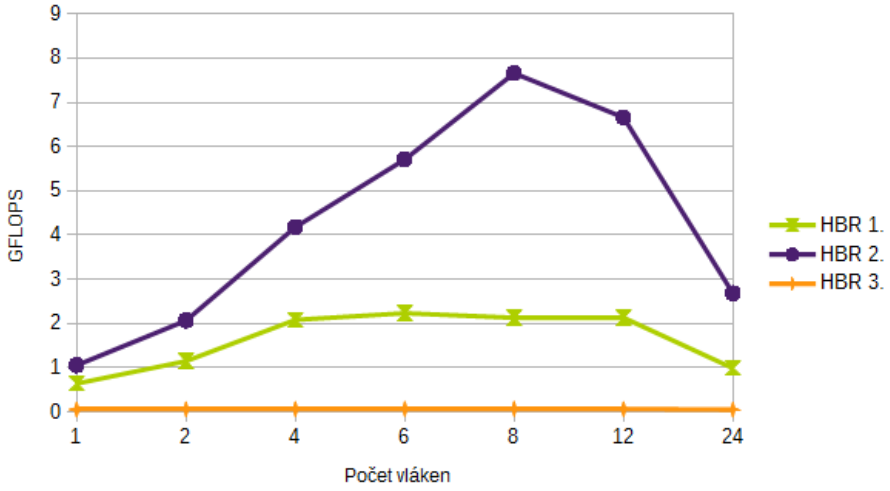
Obrázek 4.1: Měření výkonosti pro matici $n = 2000$



Obrázek 4.2: Měření výkonosti pro matici $n = 3000$



Obrázek 4.3: Měření výkonosti pro matici $n = 4000$



Obrázek 4.4: Měření výkonosti HBR pro matici $n = 4000$

torizací a bez vektorizace pro různě velké soustavy. Z grafu je vidět, že díky paralelizaci s rostoucím počtem vláken roste i výkonnost řešení soustavy intervalových rovnic. Dále je vidět, že při použití vektorizace dosahujeme o něco lepších výsledků zhruba o 10 %.

V případě metody HBR rozdělujeme výpočet na tři části. V první dojde k výpočtu matice středních hodnot a matice poloměrů. V této části díky paralelizaci dojde s vyšším počtem vláken také ke zlepšení výkonnosti, ale už od počtu šesti vláken dojde ke stagnaci a v případě čtyřiadvaceti vláken dojde ke zmenšení výkonnosti. To je způsobeno tím, že je velké množství vláken na

málé množství práce. Výsledný čas tedy negativně ovlivní režie nutná pro vytváření a plánování vláken.
Druhá část, ve které se provádí výpočty v reálné aritmetice, dochází s rostoucím počtem vláken také k vyšší výkonnosti až do počtu osmi vláken. Ve třetí části dochází k výpočtu výsledného intervalového vektoru. Protože je tento výpočet tak krátký, použitím paralelizace nedosáhneme v této části žádného zlepšení.
Ve druhé části měříme celkový čas, který trvá výpočet řešení soustavy intervalových rovnic pro metody Gaussovy eliminace(I_GEM), Gaussovy eliminace s použitím knihovny Boost(I_BGEM) a pomocí metody HBR(I_HBR). Zároveň jsou tyto naměřené časy porovnány s řešením soustavy lineárních rovnic s reálnými koeficienty pomocí knihovny Armadillo(GEM). Uvedené časy jsou v sekundách. V tabulce 4.17 je zobrazen vliv počtu vláken při použití soustavy o

Tabulka 4.17: Porovnání času pro matici $n = 4000$. Čas uveden v sekundách

# Vláken	**GEM**	**I_GEM**	**I_HBR**	**I_BGEM**
1	6.13	133.04	51.54	537.20
2	3.43	69.56	26.46	292.53
4	2.06	36.35	13.62	143.29
6	1.62	25.036	9.51	95.93
8	1.82	19.96	7.38	73.74
12	1.67	18.79	8.28	58.20
24	3.15	18.31	21.83	44.55

velikosti 4000 proměnných. Z celkového času nejlépe vychází metoda HBR a to i vzhledem k tomu, že je tato metoda nejsložitější. Důvodem je, že velká část této metody počítá s reálnými maticemi a vektory, pro jejichž implementaci je použita knihovna Armadillo. Nejhůře vychází použití Gaussovy eliminace za použití knihovny Boost Interval Arithmetic Library. To je způsobeno tím, že při každé operaci je potřeba volat metodu této knihovny, která zahrnuje určitou režii.

Při použití optimálního počtu vláken, které bylo změřeno výše, jsou výsledky zobrazené v tabulce 4.18 podobné. Nejlépe vychází znovu metoda HBR, která pro soustavu $n = 9000$ trvá zhruba minutu a půl. Nejhůře znovu dopadá Gaussova eliminace za použití knihovny Boost Interval Arithmetic Library, která pro soustavu o velikosti devíti tisíc trvá zhruba osmkrát déle než metoda HBR.
V porovnání s řešením soustavy lineárních rovnic s reálnými koeficienty pomocí knihovny Armadillo, má použití intervalové aritmetiky velký vliv na čas výsledného řešení. Nejrychlejší metoda HBR je zhruba šestkrát pomalejší oproti Gaussově eliminaci v knihovně Armadillo.

Tabulka 4.18: Měření času pro optimálního početu vláken. Čas uveden v sekundách

N	**GEM**	**I_GEM**	**I_HBR**	**I_BGEM**
1000	0.05	0.95	0.23	1.25
2000	0.34	5.40	1.08	9.08
3000	0.79	15.51	4.61	33.75
4000	1.72	30.15	7.51	68.83
5000	2.86	54.50	14.27	138.01
6000	4.64	74.62	24.04	234.20
7000	7.59	110.00	37.70	365.73
8000	10.09	154.50	55.76	532.17
9000	19.39	210.76	79.32	768.32

Závěr

Řešení soustavy lineárních intervalových rovnic je NP-těžký problém, který se snažíme řešit hledáním co nejmenšího obalu tohoto řešení. V rámci této práce byla implementována metoda Gaussovy eliminace, Modifikované intervalové aritmetiky a metody HBR. Pro možné zlepšení výsledných intervalů byla použita metoda předpodmínění, kde se daná intervalová soustava násobí maticí středních hodnot této soustavy.
Z hlediska výsledných šířek je možné říci, že nejlepší z těchto metod je metoda HBR bez použití předpodmínění. Metoda s využitím Modifikované intervalové aritmetiky po předpodmínění dává ve většině případů horší výsledné šířky, ale nejsou zase o tolik horší. Zato použití Gaussovy eliminace bez předpodmínění se ukazuje jako prakticky nepoužitelné, protože u větších soustav nedává tato metoda žádné řešení. Při použití předpodmínění většinou Gaussova eliminace už poskytuje nějaké výsledky, ale oproti metodě HBR a Modifikované intervalové aritmetice je výsledná šířka intervalů příliš velká.
Šířka výsledných intervalů se s rostoucí velikostí soustav také u všech metod zvyšuje. Také při zvyšováním šířky intervalů řešené soustavy se zvětšuje šířka výsledných intervalů zhruba lineárně. Na rozdíl od velikosti středních hodnot řešené intervalové soustavy, která nemá na šířku výsledných intervalů vliv.
U všech metod je možné díky paralelizaci pomocí knihovny OpenMP zvýšit efektivitu těchto metod. Co se týká rychlosti, nejlépe vychází metoda HBR navzdory tomu, že je tato metoda nejsložitější. To je způsobeno zejména tím, že velká část této metody počítá s reálnými maticemi a vektory. Pro implementaci metody HBR byla pro práci s reálnými maticemi a vektory využita knihovna Armadillo, která má na dané výsledky také velký vliv. Použití knihovny Boost Interval Arithmetic Library není tak efektivní, protože neobsahuje žádnou metodu pro řešení soustav intervalových rovnic a tak jedinou moužností je implementace klasických metod za použití tříd pro práci s intervalovou aritmetikou a jejich metod.

Literatura

[1] Moore, R. E.: *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. Stanford University, 1962.

[2] Buckley, J. J.: *Solving fuzzy equations in economics and finance,*. Fuzzy Sets & Systems, 1992.

[3] Kolář, J.: *Teoretická informatika*. České vysoké učení technické, 2009.

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

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

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

[7] Pilarek, M.: Solving systems of linear interval equations using the interval extended zero method and multimedia extensions. *Scientific Research of the Institute of Mathematics and Computer Science*, ročník 9, 2010: s. 203–212.

[8] Rohn, J.: An algorithm for computing the hull of the solution set of interval linear equations. *Linear Algebra and Its Applications*, ročník 435, 2011: s. 193–201. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0024379511001418`

[9] Rohn, J.: *A Handbook of Results on Interval Linear Problems*. Czech Academy of Sciences, 2005.

[10] Anderson, E.; Bai, Z.; Bischof, C.; aj.: *LAPACK Users' Guide Third Edition*. [cit. 2017-25-04].

[11] Intel: *Math Kernel Library (MKL)*. [cit. 2017-25-04].

[12] NVIDIA: *NVBLAS Library*. [cit. 2017-25-04].

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

[14] Boost Software: *Boost příručka*. [cit. 2017-10-04]. Dostupné z: http://www.boost.org/doc/libs/1_57_0/libs/numeric/interval/doc/interval.htm

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

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

[17] Brönnimann, H.; Melquiond, G.; Pion, S.: *The design of the Boost interval arithmetic library*. 2004, [cit. 2017-10-04]. Dostupné z: https://www.lri.fr/~melquion/doc/06-tcs-rnc5.pdf

[18] Ramdani, N.; Raissi, T.; Candau, Y.: Complex interval arithmetic using polar form.

[19] Pryce, J. D.; Corliss, G. F.: Interval arithmetic with containment sets. Dostupné z: http://www.cas.mcmaster.ca/~isl/Publications/IntvlArithCsets.pdf

[20] Goualard, F.: *Gaol příručka*. [cit. 2017-10-04].

[21] Neumaier, A.: On Shary's algebraic approach for linear interval equations. Dostupné z: https://www.mat.univie.ac.at/~neum/ms/shary.pdf

[22] Hijazi, Y.; Hagen, H.; Hansen, C.; aj.: Why interval arithmetic is so useful. 2010. Dostupné z: https://subs.emis.de/LNI/Seminar/Seminar07/148.pdf

[23] Breuel, T. M.: On the Use of Interval Arithmetic in Geometric Branch-and-Bound Algorithms. *Pattern Recognition Letters*, ročník 24, 2003: s. 1375–1384.

[24] Markov, S.: An iterative method for algebraic solution to interval equations. *Applied Numerical Mathematics*, ročník 30, 1999: s. 225–239.

[25] Sanderson, C.; Curtin, R.: Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, ročník 1, 2016: str. 26.

Příloha **A**

Seznam použitých zkratek

BGEM Gaussova eliminační metoda s použitím knihovny Boost Interval Arithmetic Library

GCC GNU Compiler Collection

GEM Gaussova eliminační metoda

HBR Metoda Hansen-Bliek-Rohn

LAPACK Linear Algebra Package

MGEM Modifikovaná Gaussova eliminační metoda

MMX MultiMedia eXtension

OMP Open Multi-Processing

SSE Streaming SIMD Extensions

VGEM Gaussova eliminační metoda s použitím vektorizace

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `readme.pdf` ..... stručný popis obsahu DVD
- `dp`
  - `bin` ..... zdrojové kódy implementace
    - `Solver.out` ..... program pro řešení soustav int. rovnic
    - `Generator.out` ..... program pro generování soustav int rovnic
  - `lib` ..... knihovny pro potřebné pro kompilaci
  - `data` ..... příklady soustav intervalových rovnic
  - `source` ..... zdrojové soubory pro řešič
  - `generatorSource` ..... zdrojové soubory pro generátor
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS