VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Metody rozložení oblasti s předpodmíněním

Preconditioned Domain Decomposition Methods

2011

Lukáš Malý

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2011 .....

Rád bych na tomto místě poděkoval doc. Ing. Daliboru Lukášovi, Ph.D. za příkladné vedení, mnohé rady, přípomínky a ochotu, a své rodině za podporu a trpělivost, jež projevili při psaní této bakalářské práce.

Abstrakt

Práce se zabývá primárními metodami rozložení oblasti při řešení okrajových úloh s Dirichletovými okrajovými podmínkami a studiem předpodmínění gradientních metod. Problém řešení okrajových úloh se metodou konečných prvků převede na soustavu lineárních rovnic, které se řeší numerickými iteračními metodami s předpodmíněním. Snahou je nalézt co nejrychlejší metody řešení takových soustav, k čemuž napomáhají právě metody rozložení oblasti, které umožňují dobře paralelizovat výpočet. Práce ukazuje jak ze zadaných okrajových úloh v 1D a 2D sestavit soustavy lineárních rovnic a jak aplikovat metody rozložení oblasti. Dále analyzuje možnosti předpodmínění s cílem nalézt co nejrozumnější variantu z hlediska časové a paměťové náročnosti výpočtu a přesnosti řešení.

Klíčová slova: metoda konečných prvků, primární metody rozložení oblasti, tříkroková metoda

Abstract

This work deals with primary methods of domain decomposition for solving boundary problems with the Dirichlet's boundary conditions and study of preconditioning gradient methods. The problem of solving boundary problems is converted into the system of linear equations by finite element method and this system is solved by numerical iterative methods with preconditioning. The aim is to find the fastest methods of solving such systems and right methods of domain decomposition, which allows good parallelization of computation, helps to that. This work demonstrates how to build systems of linear equations from given boundary problems in 1D and 2D and how to apply the domain decomposition method. It also analyzes the possibilities of preconditioning with aim to find the most sensible option in terms of time and memory consumption of computation and precision of the solution.

Keywords: finite element method, primary domain decomposition, threestep method

Seznam použitých zkratek a symbolů

$\partial\Omega$	–	Hranice množiny Ω
$\bar{\Omega}$	–	Uzávěr množiny Ω
Δ	–	Laplaceův operátor
∇	–	Gradient
$C(\Omega)$	–	Funkce spojitá na Ω
$C^1(\Omega)$	–	Funkce se spojitými prvními derivacemi na Ω
$C^2(\Omega)$	–	Funkce se spojitými druhými derivacemi na Ω
$\mathbb{R}$	–	Množina všech reálných čísel
S	–	Schurův doplněk
ř1	–	První řádek matice, označení při maticových úpravách
$\emptyset$	–	Prázdná množina
I	–	Jednotková matice
$\lambda_{\max}$	–	Největší vlastní číslo matice
$\|\|M\|\|$	–	Norma matice M
$\hat{A}$	–	Aproximace matice A
DDM	–	Domain Decomposition Methods
MKP	–	Metoda konečných prvků
BPS	–	Bramble, Pasciak, Schatze
PDR	–	Parciální diferenciální rovnice
PCG	–	Metoda sdružených gradientů s předpodmíněním
CG	–	Metoda sdružených gradientů

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Rozděl a panuj, vyslovil na přelomu 15. a 16. století Niccolò Machiavelli. Původně se touto zásadou řídili zejména účastníci politického boje, při snaze dostat se k moci. Díky poznání, že menší problémy se zvládají řešit snáze než velké, se dnes toto heslo hojně uplatňuje při řešení velkých matematických a programovacích úloh. Metody rozložení oblasti jsou založeny právě na principu rozděl a panuj, kdy zadanou úlohu rozdělí na menší části.

V mé bakalářské práci se zabývám řešením okrajových úloh s Dirichletovými okrajovými podmínkami v jedné a dvou dimenzích, tyto úlohy popisují například lineární pružnost, vedení tepla či elektromagnetismus. Uvedené úlohy spadají do oblasti matematického modelování, což je popisování jevů z reálného světa pomocí matematických modelů. V dnešní době výkonných počítačů je řešení těchto úloh velmi rozšířené. Navíc je snaha výpočty paralelizovat abychom dosáhli řešení v co nejkratším čase.

Cílem mé práce je ukázat, jak se řeší okrajové úlohy s Dirichletovými okrajovými podmínkami metodami rozložení oblasti autorů Brambla, Pasciaka a Schatze [1] a provézt studii nejrůznější předpodmiňovačů použitých při řešení.

V kapitole 2 předvedu přechod ze zadané spojité formulace úlohy na diskrétní, kterou potom můžeme řešit iteračními numerickými metodami. Diskretizaci úlohy provádím metodou konečných prvků, tato metoda má na rozdíl od metody konečných diferencí několik výhod. První z nich je fakt, že metoda vychází z fyzikálního modelu a je tedy v jistém smyslu přirozenější, kdežto metoda konečných diferencí má základ až v matematické formulaci úlohy. Další výhodou je univerzálnost a nezávislost MKP na tvaru diskretizované oblasti, není proto problém diskretizovat i geometricky složité tvary. Po diskretizaci dostaneme soustavu lineárních rovnic, které budeme dále řešit.

Metodám řešení lineárních rovnic se věnuji v kapitole 3, zde pouze nastíním princip řešení a odkážu na metody, které při řešení v praktické části používám.

Hlavní kapitolou je kapitola 4, primární metody rozložení oblasti. V ní ukážu jak oblast, na které řešíme úlohu, rozdělit do více podoblastí a jak vhodně přeuspořádat matici soustavy na diagonální část, odpovídající vnitřním uzlům, a části odpovídající uzlům na hranicích. Na rozdíl od populárnějších FETI (finite element tearing and interconnect) metod, metody autorů Brambla, Pasciaka a Schatze nezvyšují počet neznámých a zachovávají pozitivní definitnost úlohy, tudíž je lze použít jako předpodmínění v metodě sdružených gradientů, což v této části popisuji. Navíc ukážu paralelní způsob řešení implementovaný v Matlabu.

V poslední části (kapitola 5) představím výsledky praktické části své práce, kterou byla studie předpodmínění. Testoval jsem nejrůznější varianty předpodmínění a sledoval počet iterací a čísla podmíněnosti matice soustavy. Na závěr zhodnotím efektivnost paralelního programu a dosaženého zrychlení.

2 Diskretizace metodou konečných prvků

Pro numerické řešení úlohy je třeba provést diskretizaci a rozdělit tak interval $\langle a, b\rangle$ v 1D, popřípadě oblast Ω ve 2D, na síť konečně mnoha uzlů, ve kterých potom budeme počítat výsledné posunutí. Řešení tak bude pouze interpolací v těchto uzlech (nejčastěji lineárními splajn funkcemi, obrázek 1).

Pro diskretizaci okrajových úloh použiji metodu konečných prvků, která na rozdíl od metody konečných diferencí vychází z fyzikálního modelu a je v tomto směru více intuitivní. Přechodem z klasické formulace na slabou a variační, a následnou volbou vhodných bázových funkcí získáme soustavu s řídkou maticí, která je vhodná pro řešení velkých soustav.

2.1 MKP pro úlohy v 1D

Mějme okrajovou úlohu pro PDR druhého řádu v 1 dimenzi, tzv. klasická formulace:

$$\begin{aligned} -u''(x) &= f(x) \quad \text{pro } \forall x \in (a;b)\,, \\ u(a) = u(b) &= 0, \end{aligned} \tag{1}$$

$$f \in C((a;b)), u \in C^2((a;b)) \cap C(\langle a;b\rangle).$$

Prozatím definujme V jako množinu testovacích funkcí

$$V = \left\{v \in C^1((a;b)) \cap C(\langle a;b\rangle) | v(a) = v(b) = 0\right\}$$

a vezměme z ní libovolnou testovací funkci $v(x) \in V$. Pro funkce z V platí, že pokud u je řešením okrajové úlohy, potom $u + v$ také řeší okrajou úlohu Dirichletova typu.

$$\begin{aligned} -u''(x) &= f(x) \Big/ \cdot v(x) \\ -u''(x)v(x) &= f(x)v(x) \Big/ \int_a^b \mathrm{d}x \\ \int_a^b -u''(x)v(x)\,\mathrm{d}x &= \int_a^b f(x)v(x)\,\mathrm{d}x \end{aligned}$$

Integrací per partes dostaneme

$$\begin{aligned} \underbrace{\Big[-u'(x)v(x)\Big]_a^b}_{=0} - \int_a^b -u'(x)v'(x)\,\mathrm{d}x &= \int_a^b f(x)v(x)\,\mathrm{d}x \\ \int_a^b u'(x)v'(x)\,\mathrm{d}x &= \int_a^b f(x)v(x)\,\mathrm{d}x \end{aligned} \tag{2}$$

Rovnicni (2), kdy hledáme $u(x) \in V$ říkáme slabá formulace.

Pro funkci u, která je řešením okrajové úlohy, uvažujme prostor U, po částech hladkých funkcí definovaných na $\langle a;b\rangle$, přesněji tedy množinu funkcí splňující následujíci vlastnosti:

1. jsou spojité na $\langle a;b\rangle$, tj. $u \in C\,\langle a;b\rangle$,
2. na intervalu $\langle a;b\rangle$ je lze rozdělit do konečně mnoha podintervalů[1], na kterých jsou spojité i jejich první derivace[2],
3. na každém podintevalu platí, že $u'(x)$ je omezená

Množina U je lineárním prostorem, protože je uzavřená vzhledem k lineárním operacím sčítání funkcí a násobení funkce konstantou.

Prostorem testovacích funkcí pak rorumíme množinu

$$V = \{v \in U, v(a) = v(b) = 0\}\,.$$

Slabou formulaci pak lze zapsat v tomto tvaru

$$\begin{gathered} a(u,v) = b(v) \qquad \forall v \in V, \\ a(u,v) = \int_a^b u'(x)v'(x)\,\mathrm{d}x, \quad b(v) = \int_a^b f(x)v(x)\,\mathrm{d}x \end{gathered} \tag{3}$$

Pokud $f(x)$ bude omezená funkce, potom vlastnosti U zaručí integrovatelnost a definici zobrazení na celém U,

$$a : U \times U \to \mathbb{R}^1, \quad b : U \to \mathbb{R}^1$$

Poznámka 2.1 U je lineární prostor, o a a b říkáme že:
a je bilineární forma na U,
b je lineární funkcionál na U.

Poznámka 2.2 Bilineární forma a z (3) je symetrická, to znamená platnost vztahu

$$a(u,v) = a(v,u) \qquad \forall u, v \in U.$$

Pro symetrickou bilineární formu je ekvivalentní úlohou k (2) tzv. variační (energetická) formulace

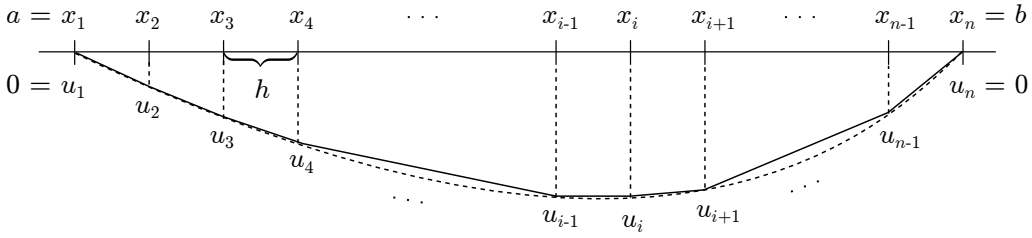
$$u(x) := \underset{w(x)\in V}{\arg\min} \Bigg\{ \underbrace{\frac{1}{2}\int_a^b \left[w'(x)\right]^2\,\mathrm{d}x - \int_a^b f(x)w(x)\,\mathrm{d}x}_{J(w)} \Bigg\},$$

kde tedy $u(x)$ hledáme jako argument minima funkcionálu $J(w)$. Podmínkou minima je rovnost

$$J'\left(u(x)\right)\left(v(x)\right) = 0 \quad \forall v(x) \in V,$$

což znamená, že derivace funkcionálu J v bodě u a směru v musí být rovna nule.

[1] pro každou funkci $u \in U$ lze najít body $\xi_i(u), i = 1, \ldots, m$ tak, že $a = \xi_0(u) < \xi_1(u) < \ldots < \xi_m(u) = b$.
[2] $u \in C^1\left(\xi_{i-1}(u), \xi_i(u)\right)$ pro všechna $i = 1, \ldots, m < \infty$.



Obrázek 1: Diskretizace

Poznámka 2.3 Tuto směrovou derivaci funkcionálu rozumějme jako tzv. Fréchetovu derivaci definovanou takto,

$$J'(u(x))(v(x)) := \lim_{\tau \to 0_+} \frac{J(u(x) + \tau \cdot v(x)) - J(u(x))}{\tau}.$$

Poznámka 2.4 Pro důkaz existence řešení úlohy (2) předpokládejme úplnost prostoru V v normě $||v(x)||_V := \int_a^b v^2(x)\,\mathrm{d}x + \int_a^b [v'(x)]^2\,\mathrm{d}x$, kde navíc integrály jsou lebegueovy a derivace je zobecněná. Korektní prostor testovacích funkcí je pak

$$V := \Big\{ v(x) : (a;b) \mapsto \mathbb{R} \,\Big|\, \int_a^b v^2(x)\,\mathrm{d}x < +\infty, \int_a^b [v(x)]^2\,\mathrm{d}x < +\infty, \\ v(a) = v(b) = 0 \text{ ve smyslu stop} \Big\}.$$

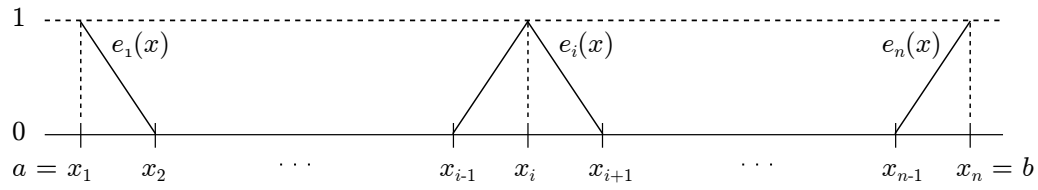
2.1.1 Síťová diskretizace

Interval $\langle a;b\rangle$ rozdělme na ekvidistantní síť s krokem $h = \frac{b-a}{n}$, kde n je počet dílků. Dostaneme tedy síť $S := \{x_1, x_2, \ldots, x_n\}$ s n uzly, jak je vidět na obrázku 1. Úlohu budeme nadále řešit pouze v těchto uzlech. Na síti S definujme spojité po částech lineární bázové funkce

$$e_1 = \begin{cases} \frac{1}{h}(x_2 - x), & x \in \langle x_1, x_2\rangle, \\ 0 \quad \text{na zbývající části intervalu } \langle a, b\rangle \end{cases}$$

$$e_i = \begin{cases} \frac{1}{h}(x - x_{i-1}), & x \in \langle x_{i-1}, x_i\rangle, \\ -\frac{1}{h}(x - x_{i+1}), & x \in \langle x_i, x_{i+1}\rangle, \\ 0 \quad \text{na zbývající části intervalu } \langle a, b\rangle \end{cases} \qquad i = 2, 3, \ldots, n-1$$

$$e_n = \begin{cases} \frac{1}{h}(x - x_{n-1}), & x \in \langle x_{n-1}, x_n\rangle, \\ 0 \quad \text{na zbývající části intervalu } \langle a, b\rangle \end{cases}$$



Obrázek 2: Bázové funkce e_i

Množina všech lineárních kombinací těchto bázových funkcí tvoří lineární prostor dimenze n, který označujeme V_h,

$$V_h := \langle e_1(x), e_2(x), \ldots, e_n(x) \rangle .$$

Libovolnou funkci u_h tohoto prostoru vyjádříme ve tvaru

$$u(x) \approx u_h(x) = \sum_{i=1}^{n} u_i \cdot e_i(x) = u_1 \cdot e_1(x) + u_2 \cdot e_2(x) + \ldots + u_n \cdot e_n(x).$$

Koeficienty $u_1, u_2, \ldots, u_n$ jsou hledané hodnoty $u_h(x_1), u_h(x_2), \ldots, u_h(x_n)$. Místo testovací funkce vezměme tyto bázové funkce:

$$v(x) := e_i(x) \text{ pro } i = 1, 2, \ldots, n \Rightarrow n \text{ lineárních rovnic.}$$

2.1.2 Sestavení soustavy

Dostáváme tedy soustavu

$$A \cdot \bar{u} = \bar{b},$$

kde

$$\begin{aligned} (A)_{ij} &= \int_a^b e_i'(x) \cdot e_j'(x)\,\mathrm{d}x, \\ (\bar{b})_i &= \int_a^b f(x) \cdot e_i(x)\,\mathrm{d}x, \end{aligned} \qquad \bar{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}.$$

V maticovém zápise:

$$\begin{bmatrix} A_{11} & \ldots & A_{n1} \\ \vdots & \ddots & \vdots \\ A_{1n} & \ldots & A_{nn} \end{bmatrix} \cdot \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$b_j\colon\quad 1\neq j\neq n:\quad \int_a^b f(x)e_j(x)\,\mathrm{d}x=\int_{x_{j-1}}^{x_{j+1}} f(x)e_j(x)\,\mathrm{d}x=$$

$$=\frac{1}{h}\left(\int_{x_{j-1}}^{x_j} f(x)(x-x_{j-1})\,\mathrm{d}x+\int_{x_j}^{x_{j+1}} f(x)(x_{j+1}-x)\,\mathrm{d}x\right)$$

$$j=1:\quad \int_a^b f(x)e_j(x)\,\mathrm{d}x=\frac{1}{h}\int_{x_1}^{x_2} f(x)(x_2-x)\,\mathrm{d}x$$

$$j=n:\quad \int_a^b f(x)e_j(x)\,\mathrm{d}x=\frac{1}{h}\int_{x_{n-1}}^{x_n} f(x)(x-x_{n-1})\,\mathrm{d}x$$

$$A_{ij}\colon\quad 1\neq i=j\neq n:\quad \int_a^b e_i'(x)e_i'(x)\,\mathrm{d}x=\int_{x_{i-1}}^{x_i}\frac{1}{h^2}\,\mathrm{d}x+\int_{x_i}^{x_{i+1}}-\frac{1}{h^2}\,\mathrm{d}x=\frac{2}{h}$$

$$i=j=1:\quad \int_a^b e_1'(x)e_1'(x)\,\mathrm{d}x=\int_{x_1}^{x_2}\frac{1}{h^2}\,\mathrm{d}x=\frac{1}{h}$$

$$i=j=n:\quad \int_a^b e_n'(x)e_n'(x)\,\mathrm{d}x=\int_{x_{n-1}}^{x_n}\frac{1}{h^2}\,\mathrm{d}x=\frac{1}{h}$$

$$|i-j|=1:\quad \int_a^b e_i'(x)e_{i+1}'(x)\,\mathrm{d}x=\int_{x_i}^{x_{i+1}}\left(-\frac{1}{h}\right)\cdot\frac{1}{h}\,\mathrm{d}x=-\frac{1}{h}$$

$$|i-j|>1:\quad \int_a^b e_i'(x)e_j'(x)\,\mathrm{d}x=0$$

Výhoda takto zvolených bázových funkcí je v tom, že matice A je třídiagonální,

$$\frac{1}{h}\begin{bmatrix}1&-1&0&\dots&0&0\\-1&2&-1&\dots&0&0\\0&-1&2&\dots&0&0\\\vdots&\vdots&\ddots&\ddots&\vdots&\vdots\\0&0&\dots&-1&2&-1\\0&0&\dots&0&-1&1\end{bmatrix}\cdot\begin{pmatrix}u_1\\u_2\\u_3\\\vdots\\u_{n-1}\\u_n\end{pmatrix}=\frac{1}{h}\begin{pmatrix}b_1\\b_2\\b_3\\\vdots\\b_{n-1}\\b_n\end{pmatrix}.$$

Nyní zakomponujeme okrajové podmínky, výsledná soustava je tato:

$$\frac{1}{h}\begin{bmatrix}1&0&0&\dots&0&0\\0&2&-1&\dots&0&0\\0&-1&2&\dots&0&0\\\vdots&\vdots&\ddots&\ddots&\vdots&\vdots\\0&0&\dots&-1&2&0\\0&0&\dots&0&0&1\end{bmatrix}\cdot\begin{pmatrix}u_1\\u_2\\u_3\\\vdots\\u_{n-1}\\u_n\end{pmatrix}=\frac{1}{h}\begin{pmatrix}0\\b_2\\b_3\\\vdots\\b_{n-1}\\0\end{pmatrix}.$$

2.2 MKP pro úlohy ve 2D

Okrajová úloha pro PDR druhého řádu ve 2D:

$$\begin{aligned} -\Delta u(x_1,x_2) &= f(x_1,x_2) && \text{v } \Omega, \text{ kde } \Omega := (a;b)\times(a;b), \\ u(x_1,x_2) &= 0 && \text{na } \partial\Omega, \end{aligned}$$

$f \in C(\Omega), u \in C^2(\Omega) \cap C(\bar{\Omega})$.

Definujme testovací množinu funkcí V jako

$$V := \Big\{ v(x_1,x_2) : \Omega \mapsto \mathbb{R} \,\Big|\, \int_\Omega v^2(x_1,x_2)\,\mathrm{d}\Omega < +\infty, \int_\Omega ||\nabla v(x_1,x_2)||^2\,\mathrm{d}\Omega < +\infty, v(x_1,x_2) = 0 \text{ na } \partial\Omega \text{ ve smyslu stop} \Big\}.$$

Poznámka 2.5 Funkce f, v a u jsou funkce $\mathbb{R}^2 \mapsto \mathbb{R}$. Pro přehlednost zápisu budu dále psát pouze $f(x)$, $u(x)$ a $v(x)$, čímž bude myšleno $f(x_1,x_2)$, $u(x_1,x_2)$ a $v(x_1,x_2)$. Stejně tak $\int \mathrm{d}x$ je rozuměno $\iint \mathrm{d}x_1\mathrm{d}x_2$.

Vezměme libovolnou funkci $v(x)$ z V a přejděme k variační formulaci:

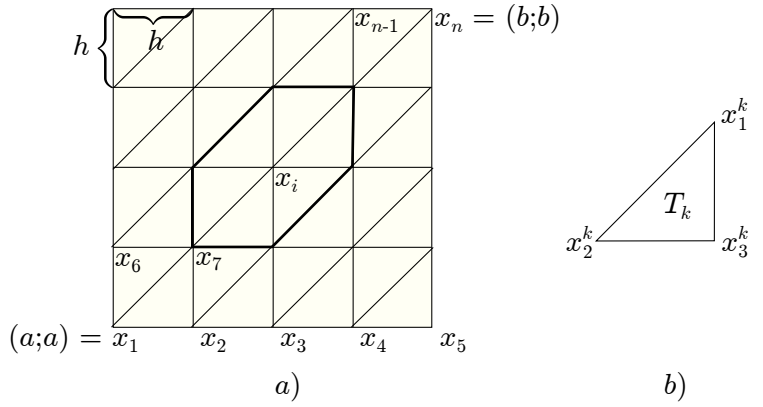
$$\begin{aligned} -\Delta u(x) &= f(x) \,\Big/ \cdot v(x) \\ -\Delta u(x)v(x) &= f(x)v(x) \,\Big/ \int_\Omega \mathrm{d}\Omega \\ -\Delta u(x)v(x)\,\mathrm{d}\Omega &= \int_\Omega f(x)v(x)\,\mathrm{d}\Omega \end{aligned}$$

Užitím 1. Greenovy formule[3] dostaneme

$$\begin{aligned} \int_\Omega \nabla u(x)\nabla v(x)\,\mathrm{d}\Omega - \underbrace{\int_{\partial\Omega} \frac{\partial u(x)}{\partial u} v(x)\,\mathrm{d}s}_{=0} &= \int_\Omega f(x)v(x)\,\mathrm{d}\Omega \\ \int_\Omega \nabla u(x)\nabla v(x)\,\mathrm{d}\Omega &= \int_\Omega f(x)v(x)\,\mathrm{d}\Omega \end{aligned} \tag{4}$$

Hledáme tedy $u(x) \in V$ pro které platí rovnice (4).

$$\begin{aligned} a(u,v) = b(v) \qquad \forall v \in V, \\ a(u,v) = \int_\Omega \nabla u(x)\nabla v(x)\,\mathrm{d}\Omega, \quad b(v) = \int_\Omega f(x)v(x)\,\mathrm{d}\Omega \end{aligned} \tag{5}$$



Obrázek 3: a) Triangulace oblasti Ω, b) jeden trojúhelník T_k z T_h

2.2.1 Triangulace

Oblast $\Omega \subset \mathbb{R}^2$ na níž okrajovou úlohu řešíme, rozdělíme na čtvercovou sít s krokem h a poté celou sít rozdělíme na konečný počet trojúhelníků, obrázek 3. Uzly očíslujeme od 1 do n, dostaneme tak síť uzlů S, kde n je počet uzlů.

$$
\begin{aligned}
&S := \{x_1, x_2, \ldots, x_n\}, \\
&T_h := \{T_1, T_2, \ldots, T_{n_T}\}, \quad T_k = \triangle_{x_1^k, x_2^k, x_3^k}, \quad n_T = 2\left(\frac{b-a}{h}\right)^2 \\
&T_h \approx \overline{\Omega}, \quad u_i \approx u(x_i), i = 1, \ldots, n
\end{aligned}
$$

Triangulace T_h je množina všech trojúhelníků T_k, počet všech trojúhelníků je n_T.

Po částech lineární bázovou funkci $e_i = e_i(x)$ příslušnou uzlu x_i (obrázek 4) definujme následovně:

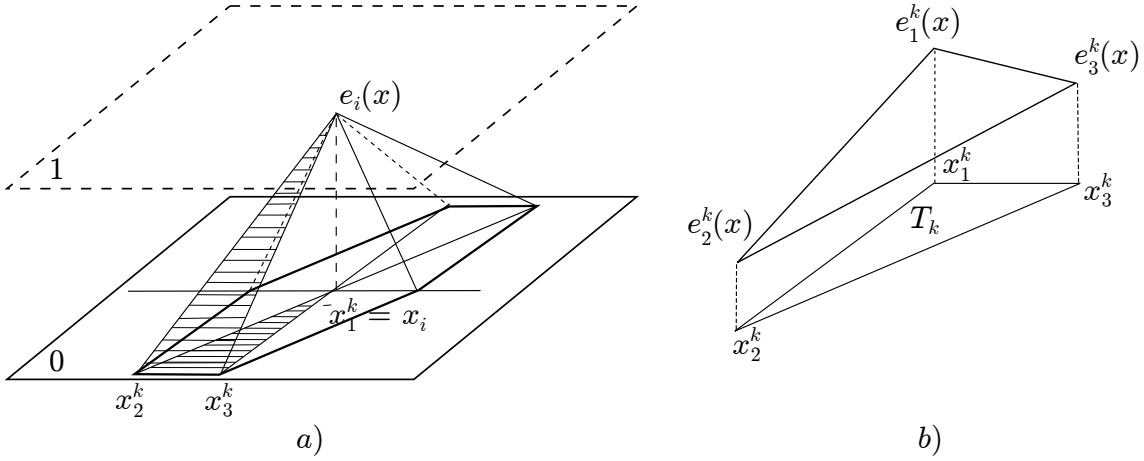
1. $e_i(x_1, x_2) = a^k + b^k x_1 + c^k x_2, \quad \forall (x_1, x_2) \in T_k$
2. $e_i(x_i) = 1, \qquad e_i(x_j) = 0 \quad \forall j \neq i,$
3. je nenulová pouze na trojúhelnících, jejichž společným vrcholem je uzel x_i.

$$(A)_{ij} = \int_\Omega \nabla e_i(x) \nabla e_j(x)\,\mathrm{d}x = \sum_{T_k} \int_{T_k} \nabla e_i(x) \nabla e_j(x)\,\mathrm{d}x$$

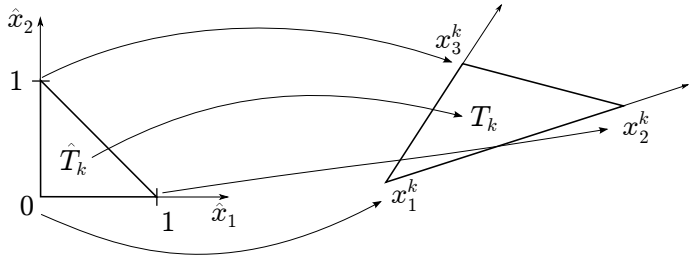
Lokální matice tuhosti pro každý trojúhelník T_k bdue

$$A^k \in \mathbb{R}^{3\times 3}, A^k = \begin{bmatrix} a_{11}^k & a_{12}^k & a_{13}^k \\ a_{21}^k & a_{22}^k & a_{23}^k \\ a_{31}^k & a_{32}^k & a_{33}^k \end{bmatrix}, A = \sum_{k=1}^{n_k} A^k$$

[3] Odvozeno z Gaussovy – Ostrogradského věty



Obrázek 4: a) Bázová funkce $e_i(x)$ v x_i, b) řešení nad jedním trojúhelníkem T_k



Obrázek 5: Zobrazení referenčního trojúhelníka $\hat{T}_k$ na trojúhelník T_k

2.2.2 Sestavení soustavy

Zaveďme si referenční trojúhelník $\hat{T}_k$ a transformaci na původní trojúhelník T_k (obrázek 5), která je dána předpisem

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1^k + \underbrace{\begin{pmatrix} x_2^k - x_1^k, x_3^k - x_1^k \end{pmatrix}}_{R^k} \cdot \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix}.$$

Pro transformované testovací funkce $\hat{e}_i$ platí

$$\begin{aligned} \hat{e}_1 &= 1 - \hat{x}_1 - \hat{x}_2 \\ \hat{e}_2 &= \hat{x}_1 \\ \hat{e}_3 &= \hat{x}_2. \end{aligned}$$

Jednotlivé funkce e_i a $\nabla e_i(x)$ pak dostaneme následovně:

$$\begin{aligned} e_i(x) &= \hat{e}_i(\hat{x}) \\ \nabla e_i(x) &= \left(R^k\right)^{-T} \cdot \nabla \hat{e}_i(\hat{x}). \end{aligned}$$

Tento vztah dosadíme do rovnice pro matici soustavy

$$\begin{aligned} (A^k)_{ij} &= \int_{T_k} \nabla e_i(x) \nabla e_j(x) \,\mathrm{d}x = \nabla e_i(x) \nabla e_j(x) \cdot \text{obsah}\,(T_k) \\ &= \left[\left(R^k\right)^{-T} \cdot \nabla \hat{e}_i(\hat{x})\right]^T \cdot \left[\left(R^k\right)^{-T} \cdot \nabla \hat{e}_j(\hat{x})\right] \cdot |T_k| \end{aligned}$$

Pomocí transformace z referenčního trojúhelníku dostáváme vzorec pro lokální matici tuhosti

$$A^k = (B^k)^T \cdot B^k \cdot \frac{|\det R^k|}{2},$$

kde

$$B^k = \left(R^k\right)^{-T} \cdot [\nabla \hat{e}_1, \nabla \hat{e}_2, \nabla \hat{e}_3] = \left(R^k\right)^{-T} \cdot \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

Lokální vektor pravých stran pro trojúhelník T_k dostaneme z

$$(b^k)_i = \int_{T_k} f(x) e_i(x) \,\mathrm{d}x.$$

Sestavení globální matice tuhosti a globálního vektoru pravých stran provedeme následujícím algoritmem, I^k jsou indexy uzlů x_1^k, x_2^k, x_3^k tojúhelníku T_k.

1. $A = 0$; $b = 0$;
2. **for** $k = 1, \ldots, n_T$ **do**
3. $\quad A(I^k, I^k)$ += A^k;
4. $\quad b(I^k)$ += b^k;
5. **end for**

Nakonec vynulujeme mimodiagonální prvky ve sloupcích a řádcích matice A odpovídající Dirichletovým indexům a vynulujeme příslušné prvky vektoru b odpovídající nulovým Dirichletovým podmínkám.

3 Metody řešení

Soustavy lineárních rovnic, které dostáváme nejen při diskretizaci okrajových úloh, se dají řešit mnoha způsoby. V numerické matematice nejčastěji používáme iterační metody, které jsou na rozdíl od přesných řešičů rychlejší, avšak jsou zatíženy chybou v závislosti na relativní přesnosti iterační metody. Vyspělé iterační metody, jako jsou Richardsonova metoda nebo metoda sdružených gradientů konvergují k řešení poměrně rychle, proto se v praxi velmi často využívají. Tyto metody lze navíc ještě více zrychlit vhodným předpodmíněním.

Zde nastíním princip předpodmínění a ukážu algoritmus metody sdružených gradientů, který jsem použil v praktické části své práce pro numerické experimenty.

3.1 Předpodmínění

Rychlost konvergence iteračních metod závisí na čísle podmíněnosti matice A. Procesem kdy nahradíme soustavu $Au = b$ ekvivalentní soustavou $\bar{A}u = \bar{b}$, kde číslo podmíněnosti matice $\bar{A}$ je menší než číslo podmíněnosti matice A, nazýváme předpodmínění. Volíme matici C, která je „blízká“ k matici A v tom smyslu, že $C^{-1}A$ má malé číslo podmíněnosti, pak

$$C^{-1}Au = C^{-1}b$$

je ekvivalentní soustava.

3.2 Metoda sdružených gradientů s předpodmíněním

Metoda sdružených gradientů s předpodmíněním je popsána v [2], zde ukážu jen algoritmus metody napsán v Matlabu, který jsem použil při testování. Vstupními parametry funkce PCG z výpisu kódu jsou matice soustavy A, vektor pravých stran b, počáteční aproximace řešení u_0, relativní přesnost, neboli chyba metody err a předpodmiňující matice C. Výstupními parametry jsou řešení soustavy vektor u a počet iterací k.

```
function [ u,k ] = PCG( A,b,u0,err,C)
n = length(A);
u = u0;
r = A*u0 - b;
v = inv(C)*r;                          % Aplikace predpodmineni
p = v;
rho0 = v'*r;
rho = rho0;
for k=0:n-2
    s = A*p;
    alpha = rho/(s'*p);
    u = u - alpha*p;
    r = r - alpha*s;
    v = inv(C)*r;                      % Aplikace predpodmineni
    rhom = rho;
    rho = v'*r;
    if rho <= err^2 * rho0
        break;
```

```
    else
        beta = rho/rhom;
        p = v + beta*p;
    end
end
k = k+1;
end
```

Výpis 1: Metoda sdružených gradientů s předpodmíněním

V další části této práce popíši metody rozložení oblasti a řešení tříkrokovou metodou, která nahradí aplikací předpodmínění v uvedeném algoritmu PCG

4 Primární metody rozložení oblasti

Primární metody rozložení oblasti (angl. Primary domain decomposition methods, DDM) jsou založeny na principu rozděl a panuj. Tyto metody byly původně vyvinuty pro řešení PDR nad oblastí ve dvou nebo třech dimenzích. Já se jimi budu zabývat v jedné a dvou dimenzích.

V dnešní době mají tyto metody široké využití především pro svou snadnou paralelizovatelnost. Navíc metody rozložení oblasti podle autorů Brambla, Pasciaka a Schatze [1], na rozdíl od populárnějších FETI metod, nezvyšují počet neznámých a zachovávají pozitivní definitnost úlohy, tudíž je lze použít jako předpodmínění v Richardsonově metodě nebo metodě sdružených gradientů.

V této kapitole ukáži dekompozici úlohy na podoblasti a tříkrokovou metodu, kterou se úloha následně řeší. Také předvedu její paralelizaci.

4.1 Dekompozice úlohy na podoblasti

Jednotlivé 1D a 2D úlohy rozložíme na podoblasti, následně vytvoříme indexové množiny vnitřních uzlů a uzlů na kostře (na hranicích podoblastí) a přeuspořádáme celou soustavu.

4.1.1 Úlohy v 1D

Množinu Ω (interval $\langle a; b\rangle$) rozložíme na N podoblastí s krokem H (podle obrázku 6),

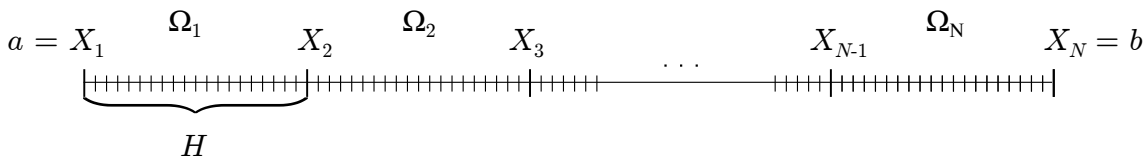
$$\bar{\Omega} = \bigcup_{i=1}^{N} \bar{\Omega}_i, \quad \Omega_i \cap \Omega_j = \emptyset \text{ pro } i \neq j.$$

Z kapitoly 2.1 máme soustavu n rovnic o n neznámých na síti S. Jednotlivé uzly sítě a jejich indexy nám tvoří indexovou množinu $I := \{1, \ldots, n\}$. Z této množiny si vytvoříme indexové množiny (vektory) pro uzly jednotlivých podoblastí $I_1, \ldots, I_N$ a uzly na kostře I_S (obrázek 7),

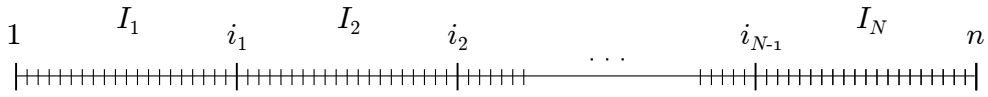
$$\begin{aligned} I_1 &= \{2, \ldots, i_1 - 1\} \\ I_2 &= \{i_1 + 1, \ldots, i_2 - 1\} \\ &\vdots \\ I_N &= \{i_{N-1} + 1, \ldots, n - 1\} \\ I_S &= \{1, i_1, i_2, \ldots, i_{N-1}, n\}. \end{aligned}$$

4.1.2 Úlohy ve 2D

Čtvercovou oblast Ω rozložíme na $\sqrt{N}$ podoblastí s krokem H v každém směru, tedy dostaneme N podoblastí. Z kapitoly 2.2 máme soustavu n rovnic o n neznámých na síti



Obrázek 6: Rozložení intervalu $\langle a; b\rangle$ do N podoblastí



Obrázek 7: Indexové množiny $I_1, \ldots, I_N$

S. Jednotlivé uzly sítě a jejich indexy nám tvoří indexovou množinu $I := \{1, \ldots, n\}$. Z této množiny indexů opět, podobně jako v 1D, vytvoříme indexové množiny pro uzly jednotlivých podoblastí $I_1, \ldots, I_N$ a uzly na kostře I_S. Konkrétní takovéto dělení můžeme vidět na obrázku 8, kde množinu Ω dělíme na 2 části v každém směru, dostaneme tedy 4 podoblasti.

Pomocí indexových množin přeuspořádáme soustavu následujícím způsobem,

$$\begin{array}{rcll} A_i & = & A_{I_iI_i} & \ldots \text{ diagonální bloky,} \\ A_{iS} & = & A_{I_iI_S}, & \\ A_{Si} & = & A_{I_SI_i}, & \\ A_{SS} & = & A_{I_SI_S} & \ldots \text{ blok hraničních uzlů,} \end{array}$$

$$i \in \langle 1, \ldots, N\rangle\,.$$

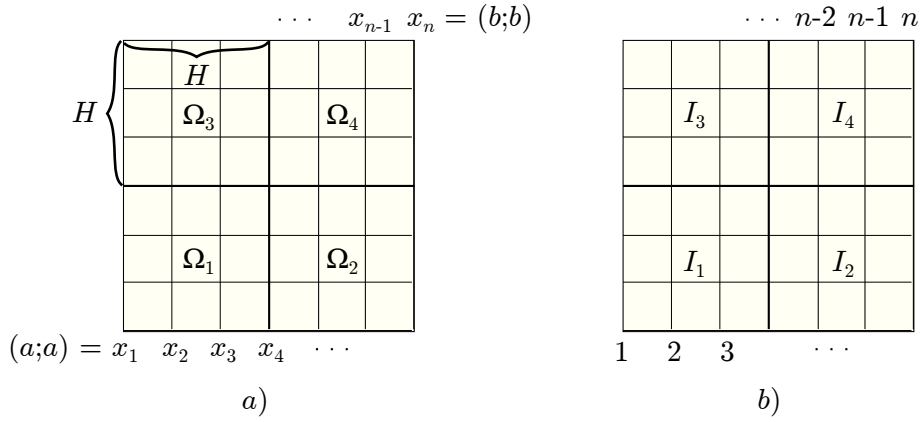
Stejně uspořádáme i vektor pravých stran a dostaneme soustavu ve tvaru

$$\begin{pmatrix} A_1 & & & & A_{1S} \\ & A_2 & & & A_{2S} \\ & & \ddots & & \vdots \\ & & & A_N & A_{NS} \\ A_{S1} & A_{S2} & \cdots & A_{SN} & A_{SS} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \\ u_S \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \\ b_S \end{pmatrix},$$

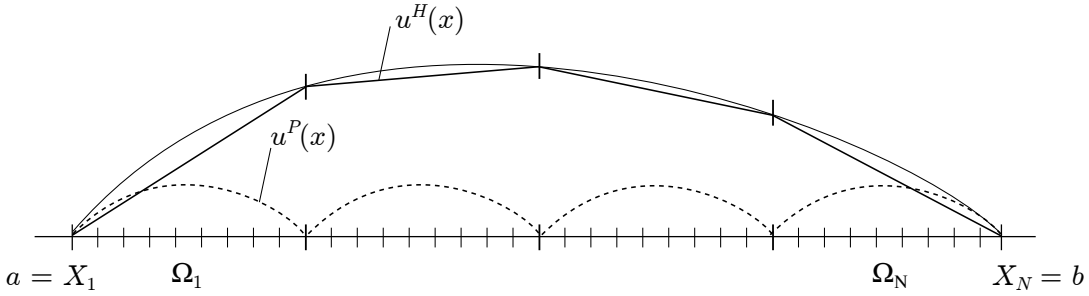
kde u_i představuje vektory neznámých na podoblasti Ω_i a u_S představuje vektor všech neznámých na hranicích. Pro zjednodušení si můžeme soustavu vyjádřit jako

$$\begin{pmatrix} A_{II} & A_{IS} \\ A_{SI} & A_{SS} \end{pmatrix} \cdot \begin{pmatrix} u_I \\ u_S \end{pmatrix} = \begin{pmatrix} b_I \\ b_S \end{pmatrix},$$

kde A_{II} představují uzly vně jednotlivých podoblastí a A_{SS} uzly na hranicích. A_{IS} představuje spojení uzlů podoblastí s hranicí a A_{SI} představuje spojení hraničních uzlů s podoblastmi. Platí, že $A_{IS} = A_{SI}^T$, tedy matice soustavy je symetrická.



Obrázek 8: a) Rozložení oblasti Ω na Ω_i, b) Indexové množiny $I_1, \ldots, I_4$



Obrázek 9: Partikulární a homogenní řešení soustavy

4.2 Řešení tříkrokovou metodou

Na obrázku 9 vidíme ukázku řešení v 1D po dekompozici. Úloha (1) přejde z původního tvaru na úlohu (6) ve tvaru

$$\begin{array}{rcll} -u_i''(x) & = & f(x) \quad \text{v } \Omega_i, & i = 1, \ldots, N \\ u(x) & = & 0 \qquad \text{na } \partial\Omega & \\ u_i(X_i) & = & u_{i+1}(X_i) & \left.\begin{array}{l}\\\end{array}\right\} \text{podmínky přechodu.} \\ u_i'(X_{i-}) & = & u_{i+1}'(X_{i+}) & \end{array} \tag{6}$$

Řešení u rozdělíme na partikulární a homogenní řešení, a provedeme restrikci na jednotlivé podoblasti.

$$\begin{array}{rcl} u(x)|_{\Omega_i} & = & u_i(x) \\ u(x) & := & u^H(x) + u^P(x), \end{array}$$

kde

$$\begin{array}{rcl} u^P(x)|_{\Omega_i} =: u_i^P(x), \quad u_i^P : \left\{ \begin{array}{rcl} -\left[u_i^P(x)\right]'' & = & f(x) \quad \text{v } \Omega_i \\ u_i^P(x) & = & 0 \quad \text{na } \partial\Omega_i \end{array} \right. \\ u^H(x)|_{\Omega_i} =: u_i^H(x), \end{array}$$

$$u_i^H : \begin{cases} -\left[u_i^H(x)\right]'' &= 0 \quad \text{v } \Omega_i \\ u^H(x) &= 0 \quad \text{na } \partial\Omega_i \\ u_i^H(X_i) &= u_{i+1}^H(X_i) \\ \left[u_i^H(X_{i-})\right]' - \left[u_{i+1}^H(X_{i+})\right]' &= -\left(\left[u_i^P(X_{i-})\right]' - \left[u_i^P(X_{i+})\right]'\right) \end{cases}$$

Poznámka 4.1 Pro 2D úlohu provedeme stejnou úvahu. Postup je analogický, akorát derivace zde uvažujeme jako parciální derivace.

V obou dimenzích, 1D i 2D, dostáváme soustavu s vektorem řešení u ve tvaru

$$\begin{pmatrix} A_{II} & A_{IS} \\ A_{SI} & A_{SS} \end{pmatrix} \cdot \begin{pmatrix} u_I^P + u_I^H \\ u_S^H \end{pmatrix} = \begin{pmatrix} b_I \\ b_S \end{pmatrix}.$$

Jednotlivé části řešení spočítáme následovně:

u_I^P: partikulární řešení u_I^P je řešením Dirichletovy okrajové úlohy na podoblasti Ω_i s nulovými okrajovými podmínkami, je tedy přímo řešením soustavy

$$A_{II} \cdot u_I^P = b_I. \tag{7}$$

Pro vyjádření zbylých dvou vektorů převedeme vektor u^P na druhou stranu a provedeme následující úpravy.

$$\begin{aligned} \begin{pmatrix} A_{II} & A_{IS} \\ A_{SI} & A_{SS} \end{pmatrix} \cdot \begin{pmatrix} u_I^H \\ u_S^H \end{pmatrix} &= \begin{pmatrix} \underbrace{b_I - A_{II} \cdot u_I^P}_{0} \\ b_S - A_{SI} \cdot u_I^P \end{pmatrix} \quad -A_{SI} \cdot A_{II}^{-1} \cdot \text{ř1} \\ \begin{pmatrix} A_{II} & A_{IS} \\ 0 & S \end{pmatrix} \cdot \begin{pmatrix} u_I^H \\ u_S^H \end{pmatrix} &= \begin{pmatrix} 0 \\ b_S - A_{SI} \cdot u_I^P \end{pmatrix} \end{aligned} \tag{8}$$

$$S := A_{SS} - A_{SI} A_{II}^{-1} A_{IS}, \quad S \text{ je tzv. Schurův doplněk}$$

u_S^H: z druhého řádku přechozí rovnice nám u_S^H přímo vychází jako řešení soustavy

$$S \cdot u_S^H = b_S - A_{SI} \cdot u_I^P.$$

u_I^H: z prvního řádku předchozí rovncice dostáváme u_I^H jako řešení soustavy

$$-A_{II} \cdot u_I^H = A_{IS} \cdot u_S^H.$$

Tímto dostáváme řešení ve třech krocích,

1. spočítat u_I^P,
2. spočítat u_S^H,

3. spočítat u_I^H.

Výsledné u dostaneme jako

$$u = \begin{pmatrix} u_I^P + u_I^H \\ u_S^H \end{pmatrix}.$$

Tříkrokovou metodu použijeme v metodě sdružených gradientů (kapitola 3.2) jako aplikaci předpodmínění.

```
function [v] = aplikujC(r)

    vip = Aii \ r(1:length(Aii));                 %partikulární řešení na podoblastech
    vsh = S \ (r(length(Aii)+1:end) - Asi*vip);   %homogenní řešení na skeletu
    vih = -Aii \ (Ais*vsh);                       %homogenní řešení na podoblastech
    v = [vip + vih; vsh];
end
```

Výpis 2: Tříkroková metoda jako aplikace předpodmínění

Pro lepší pochopení chování metody při různých formách předpodmínění si ukážeme jak vypadá inverze matice soustavy A, a to jak přímo, tak i její blokově-diagonální UDU^T rozklad.

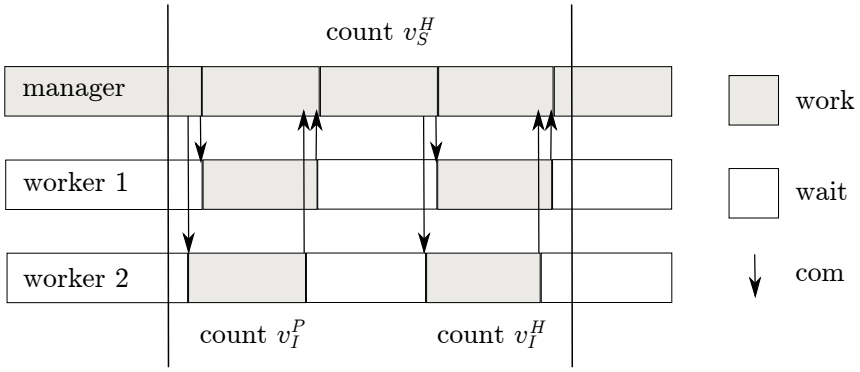
$$\begin{aligned} A^{-1} &= \begin{pmatrix} A_{II}^{-1} + A_{II}^{-1} A_{IS} S^{-1} A_{SI} A_{II}^{-1} & -A_{II}^{-1} A_{IS} S^{-1} \\ -S^{-1} A_{SI} A_{II}^{-1} & S^{-1} \end{pmatrix} = \\ &= \begin{pmatrix} I & -A_{II}^{-1} A_{IS} \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \cdot \begin{pmatrix} I & -A_{II}^{-1} A_{IS} \\ 0 & I \end{pmatrix}^T \end{aligned} \tag{9}$$

V kapitole numerické experimenty budeme jako předpomiňovače používat aproximaci matice A^{-1}.

4.3 Paralelizace řešení tříkrokovou metodou

V předchozí části jsme si ukázali jak při předpodmínění tříkrokovou metodou spočítat řešení ve třech krocích. Přitom kroky 1 a 3 lze počítat nezávisle pro jednotlivé podoblasti. Toho se dá využít k paralelizaci výpočtu. Paralelizace algoritmů je v dnešní době velmi rozšířená záležitost, která nám přináší urychlení výpočtů, jelikož se výpočty rozdělí mezi několik počítačů, které pak pracují souběžně. Při řešení rozsáhlých úloh se dnes bez paralelního přístupu téměř neobejdeme. Proto jsem paralelizoval tříkrokovou metodu, která je k tomu navíc velmi vhodná.

Všechny programy k této práci jsem realizoval v Matlabu, proto jsem se rozhodl použít Matlab i k paralelizaci. Využil jsem k tomu parallel computing toolbox, Matlabovské paralelní prostředí pmode, které umožňuje uživateli s více jádrovými procesory spouštět programy na jednotlivých procesech, a MPI posílání zpráv. MPI je protokol pro podporu paralelního řešení výpočetních problémů[6].



Obrázek 10: Graf komunikace a vytížení procesů během jedné iterace v tříkrokové metodě

4.3.1 Popis paralelizace

Pro paralelní algoritmus jsem využil modelu manager - worker, kdy hlavní proces, tzv. root, řídí běh celého programu a během něj zasílá na další procesy (workery) jednotlivé úlohy. Zaměřil jsem se pouze na paralelizaci tříkrokové metody, proto v tomto případě root sestaví sám celou soustavu rovnic a rozdělí ji do podoblastí. Přitom rozešle ostatním procesům jejich příslušné maticové bloky. Dále root-proces řídí celý běh programu a především samotné řešení soustavy. Další procesy se zapojují až při aplikaci předpodmínění, tedy ve tříkrokové metodě. Kdy každý proces obdrží příslušná data, spočítá partikulární řešení pro svou danou podoblast a vrátí výsledek hlavnímu procesu. Ten pak spočítá homogenní řešení na hranicích a to opět rozešle na jednotlivé procesy. Ty spočítají homogenní řešení pro své oblasti a vrátí výsledky. Nakonec root sestaví výsledný vektor. Root-procesu je také přidělena příslušná oblast, ten se tak zapojuje do výpočtu spolu s workry při výpočtu lokálních řešení na jednotlivých podoblastech.

Na obrázku 10 vidíme graf vytíženosti jednotlivých porcesů v každé iteraci. Šipky označují komunikaci mezi procesy, kdy si navzájem přeposílají data. Bílá místa znázorňují stav, kdy procesy čekají a nic nedělají. Šedé pole znázorňuje práci procesu, tedy nějaký výpočet.

4.3.2 Vlastní paralelizace v Matlabu

Pro paralelní programování v Matlabu je k dispozici paralelní mód pmode. Při zavolání příkazu `pmode start local n` se aktivuje lokální paralelní mód pro `n` procesů (maximálně 8) a otevře se speciální okno Matlabu, ve kterém pak můžeme spouštět paralelní algoritmy.

Popsaný paralelní algoritmus v Matlabu vypadá následovně, výpis 3 ukazuje tříkrokovou metodu z pohledu root-procesu, která je implementována ve funkci `aplikujC_par`. Výpis 4 představuje tříkrokovou metodu z pohledu worker-procesů. Při paralelním programu v Matlabu od sebe rozlišujeme jednotlivé procesy proměnnou `labindex`, procesy jsou číslovány od jedničky. Pomocí jednoduché if podmínky tak můžeme určit který proces má vykonávat jakou činnost.

```
function [v] = aplikujC_par(r)
    st = 1;
    dd = length(myAi);
    myri = r(st:st+dd-1);              % Root lokalni vektor r
    st = st + dd;
    labBarrier;                        % R-bariera2
    for i=2:numlabs                    % Odeslani r pro vypocet lokalnich vip
        dd = length(cell2mat(AAp(i)));
        data = r(st:st+dd-1);
        labSend(data, i);
        st = st + dd;
    end
    vip = myAi \ myri;                 % Root pocita lokalni vip
    labBarrier;                        % R-bariera3
    for i=2:numlabs                    % Prijem vip od workru
        vip = [vip;labReceive(i)];
    end
    vsh = S \ (r(st:end) - Asi*vip);   % r pro uzly na skeletu
    labBarrier;                        % R-bariera4
    for i=2:numlabs                    % Odeslani vsh pro vypocet vih
        labSend(vsh, i);
    end
    vih = -myAi \ (myAis*vsh);         % Root pocita lokalni vih
    labBarrier;                        % R-bariera5
    for i=2:numlabs                    % Prijem vih od workru
        vih = [vih;labReceive(i)];
    end
    v = [vip + vih; vsh];
end
```

Výpis 3: Tříkroková metoda paralelně - kód řídícího root-procesu

```
labBarrier;                        % W-bariera2
myri = labReceive(1);              % Prijem lokalniho r od root
vip = myAi \ myri;                 % Vypocet lokalniho reseni vip
labBarrier;                        % W-bariera3
labSend(vip, 1);
labBarrier;                        % W-bariera4
vsh = labReceive(1);
vih = -myAi \ (myAis*vsh);         % Vypocet lokalniho reseni vih
labBarrier;                        % W-bariera5
labSend(vih, 1);
```

Výpis 4: Tříkroková metoda paralelně - kód workrů

Ve výpisech programů si můžeme všimnout příkazů labBarrier, labSend a labReceive, to jsou právě funkce Matlabovského MPI obsaženého v parallel computing toolboxu.

labBarrier	zastaví daný proces do doby než tento příkaz zavolají všechny procesy programu, toho se využívá k sjednocení procesů.
labSend(data, i)	posílá data, což může být jakýkoli Matlabovský objekt, procesu i.
data = labReceive(i)	přijímá data od procesu i.

Poznámka 4.2 Vypsané zdrojové kódy v této kapitole jsou jen pro ukázku a demonstrují použití paralelizace v Matlabu. Není zde uveden celý kód programu, ani popis jednotlivých proměnných. Důležité je však použití funkcí pro komunikaci mezi jednotlivými procesy a způsob paralelního řešení výpočtu.

5 Numerické experimenty

V této kapitole představím výsledky praktické části své bakalářské práce. V ní jsem se zabýval především studií různých druhů předpodmínění. Testoval jsem zvlášť předpodmínění pro diagonální bloky A_{II} a pro Schurův doplněk S.

Přesným předpodmíněním a řešením tříkrokovou metodou lze dosáhnout výsledku v jedné iteraci. Nevýhodou však je náročnost výpočtu. K sestrojení Schurova doplňku je třeba napočítat několik inverzí, což je z hlediska času výpočtu dost náročná operace. Inverze bloků A_{II} jsou v tomto případě plné matice, které zabírají spoustu místa a je problém s jejich uložením do paměti. To značně limituje využití těchto metod k řešení velkých soustav. Snažíme se tak najít způsoby jak se vyhnout přímým výpočtům Schurova doplňku buď to úplně, nějakým jiným předpodmíněním, a nebo jej řešit nepřesně některou s numerických metod. V první části jsem se pokusil srovnat výsledky různých předpodmínění a počtu iterací. V druhé části demonstruji výsledky paralelizace výpočtu a dosaženého zrychlení.

Všechny výpočty a měření uvedené dále v textu byly prováděny na těchto úlohách:

1D:

$$\begin{aligned} f(x) &= -1 \\ \Omega &= (-2;6) \end{aligned}$$

2D:

$$\begin{aligned} f(x_1,x_2) &= 2\pi^2 \cdot \sin(\pi x_1^2) \cdot \sin(\pi x_2^2) \\ \Omega &= (-2;4) \times (-2;4) \end{aligned}$$

Relativní přesnost použité PCG metody a CG metody v tabulkách je 10^{-8}.

5.1 Předpodmínění tříkrokovou metodou

Při testování předpodmínění tříkrokovou metodou budu používat různé typy předpodmiňovačů. Předpodmiňovači tříkrokové metody budeme aproximovat inverzi matice A

$$A^{-1} \approx \begin{pmatrix} I & -\hat{A}_{II}^{-1}A_{IS} \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} \hat{A}_{II}^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{pmatrix} \cdot \begin{pmatrix} I & -\hat{A}_{II}^{-1}A_{IS} \\ 0 & I \end{pmatrix}^T,$$

tento rozložený tvar je pro studii dopadu jednotlivých předpodmiňovačů na chování soustavy názornější. Pokud by $\hat{A}_{II} = A_{II}$ a $\hat{S} = S$, pak máme řešení v 1 iteraci. V druhé části této podkapitoly se věnuji testování předpodmiňovači s narušením spektra.

5.1.1 Předpodmínění tříkrokovou metodou s různými předpodmiňovači

V této části testuji předpodmínění diagonálních bloků $\hat{A}_{II}$ a Schurova doplňku $\hat{S}$. Jako první variantu v tabulkách uvádím řešení metodou sdružených gradientů bez předpodmínění, tuto variantu značím A). Dále používám kombinace předpodmínění diagonálních bloků a Schurova doplňku:

B) $\hat{A}_{II} = A_{II}, \quad \hat{S} = I$ (jednotková matice),

C) $\hat{A}_{II} = A_{II}, \quad \hat{S} = A_{SS}$,

D) $\hat{A}_{II} = A_{II}, \quad \hat{S} = \text{diag}(S)$,

E) $\hat{A}_{II} = \text{diag}(A_{II}), \quad \hat{S} = S$,

F) $\hat{A}_{II} = A_{II}, \quad \hat{S} = S$.

		počet iterací					
N	H/h	A)	B)	C)	D)	E)	F)
2	2	4	2	2	1	1	1
	4	8	2	2	1	6	1
	8	16	2	2	1	14	1
	16	32	2	2	1	32	1
	32	64	2	2	1	64	1
4	2	8	4	4	3	1	1
	4	16	4	4	3	13	1
	8	32	4	4	3	32	1
	16	64	4	4	3	64	1
	32	128	4	4	3	128	1
8	2	16	7	8	7	1	1
	4	32	8	8	7	28	1
	8	64	8	8	7	64	1
	16	128	8	8	7	128	1
	32	256	9	9	7	256	1
16	2	32	15	16	15	1	1
	4	64	16	17	15	64	1
	8	128	17	17	15	128	1
	16	256	17	17	15	256	1
	32	512	17	18	15	512	1

Tabulka 1: Úloha 1D - počty iterací při různých metodách předpodmínění, řešením PCG s tříkrokovou metodou.

V 1D (tabulka 1) vidíme, že výsledky varianty B) a C) jsou téměř totožné. Varianta D) se od nich také příliš neliší. Je to dáno tím, že v jedné dimenzi je matice A_{SS} diagonální, ve všech třech případech je tedy matice $\hat{S}$ diagonální.

		počet iterací					
$N \times N$	H/h	A)	B)	C)	D)	E)	F)
2×2	2	3	2	2	1	1	1
	4	7	2	2	3	6	1
	8	15	2	2	5	14	1
	16	28	2	2	9	29	1
	32	72	2	2	16	89	1
4×4	2	7	11	5	10	1	1
	4	15	21	11	19	21	1
	8	28	27	15	26	62	1
	16	72	43	24	42	182	1
	32	156	62	35	59	495	1
8×8	2	15	16	7	15	1	1
	4	28	21	11	20	26	1
	8	72	36	20	35	117	1
	16	156	51	28	49	355	1
16×16	2	28	25	12	24	1	1
	4	72	45	24	44	77	1
	8	156	66	35	62	358	1

Tabulka 2: Úloha 2D - počty iterací při různých metodách předpodmínění, řešením PCG s tříkrokovou metodou.

Z tabulky vyplývá, že počet iterací při předpodmínění B), C) a D) závisí pouze na počtu podoblastí, nikoliv pak na celkovém počtu neznámých. Diagonální prvky jsou v těchto případech předpodmíněny přesně, nepřesnost je pouze u prvků odpovídající hranicím, jejich počet samozřejmě závisí pouze na počtu podoblastí. Dále můžeme pozorovat, jak je soustava citlivá na předpodmínění diagonálních bloků. Nejhorší výsledky počtu iterací má varianta E). Při řešení této soustavy je proto vhodné diagonální prvky vždy předpodmínit přesně nebo téměř přesně narušením, tomu se věnuji v další části. Při přesném předpodmínění Schurova doplňku i diagonálních bloků je počet iterací vždy 1 nezávisle na počtu neznámých nebo počtu podoblastí, jedná se o přesné řešení tříkrokovou metodou.

Ve 2D úloze (tabulka 2) je situace poněkud odlišná. Matice A_{SS} už není diagonální, variantu, u které by počet iterací silně závisel pouze na počtu podoblastí, zde již nevidíme. Můžeme si všimnout varianty E), u níž jsou počty iterací nejvyšší. Z toho je patrné, že předpodmínění diagonálních bloků bychom měli dělat přesně. Všechny ostatní varianty,

kde jsou bloky A_{II} předpodmíněny přesně mají lepší výsledky, dokonce i řešením CG bez předpodmínění dosáhneme přívětivějšího počtu iterací.

Nejlepší variantou je opět F), kde se jedná o přesné předpodmínění tříkrokovou metodou. Druhá nejlepší varianta se jeví D), tedy varianta kdy je S předpodmíněno maticí hraničních prvků A_{SS}.

5.1.2 Předpodmínění tříkrokovou metodou s narušením spektra

V následujících testech budeme uvažovat předpodmínění $\hat{A}_{II}$ a $\hat{S}$ narušené o ε relativně ke spektrální normě matice, $||M|| = \lambda_{\max}(M)$. Mějme rozklad matice M

$$M = V \cdot \Lambda \cdot V^T,$$

potom definujme narušení spektra matice M o ε následovně

$M = V \cdot \Lambda \cdot V^T$;
$\Lambda_\varepsilon = \Lambda + \varepsilon \cdot \lambda_{\max} \cdot I$;
naruš$(M, \varepsilon) := V \cdot \Lambda_\varepsilon \cdot V^T$;

kde $\lambda_{\max}$ je největší vlastní číslo matice M a I je jednotková matice.

Pro testování jsem použil tyto 3 varianty předpodmínění:

G) $\hat{A}_{II} = A_{II}, \quad \hat{S} = \text{naruš}(S, \varepsilon)$

H) $\hat{A}_{II} = \text{naruš}(A_{II}, \varepsilon), \quad \hat{S} = \text{naruš}(S, \varepsilon)$

I) $\hat{A}_{II} = \text{naruš}(A_{II}, \varepsilon^2), \quad \hat{S} = \text{naruš}(S, \varepsilon)$

Číslo narušení ε je pro všechny testy stejné a to 10^{-2}, relativní přesnost PCG je opět 10^{-8}. V tabulkách měřím počty iterací a číslo podmíněnosti při zvedání počtu neznámých, ale zachování poměru počtu neznámých ku počtu podoblastí $n/N = H/h$. Číslem podmíněnosti v tabulkách je číslo podmíněnosti matice $\hat{A}^{-1}A$.

V tabulce 3 pro 1D úlohy vidíme, že předpodmínění H) není příliš dobré. Při poměru H/h větším než 2 je vidět jak číslo podmíněnosti rychle roste, s tím se samozřejmě zvyšují i počty iterací. Kdežto předpodmínění I) je na tom o poznání lépe. V číslech podmíněnosti i počtech iterací se vyrovná prvnímu případu, kdy bereme diagonální bloky přesné. Z toho vyplývá, že narušení ε^2 by mohlo být dostačující i při řešení velkých úloh.

Ve 2D úloze, tabulka 4, je již varianta předpodmínění H) vyrovnanější, avšak při větším poměru H/h je vidět nárůst čísla podmíněnosti, který se projevuje i v počtech iterací. Varianta I) je opět velmi dobrá a má srovnatelné výsledky s variantou G). V této tabulce jsem pro $n \times n$ nad 40×40 uvedl pouze počty iterací z důvodu náročnosti přímého výpočtu inverze matice A, kterou je potřeba spočítat pro určení čísla podmíněnosti.

U obou případů, úloh v 1D i ve 2D je z tabulek vidět, že je potřeba předpodmiňovat diagonální bloky $\hat{A}_{II}$ přesněji než Schurův doplněk $\hat{S}$. Toto chování je patrné i z inverze

H/h	Var							
		n	10	20	100	200	5000	1000
2	G)	Iter	5	6	12	18	36	63
		Cond	1,31298	1,59911	20,3725	81,0697	506,589	2026,4
	H)	Iter	5	6	12	19	36	63
		Cond	1,14715	1,61385	20,2275	80,3327	501,681	2006,59
	I)	Iter	5	6	12	18	36	63
		Cond	1,1315	1,59166	20,0049	79,4958	496,549	1986,12
4	G)	Iter	3	5	8	12	21	35
		Cond	1,05782	1,2128	9,14934	38,8525	251,35	1011,22
	H)	Iter	5	7	14	19	33	54
		Cond	1,4338	5,35022	758,001	3854,68	26074.3	105538
	I)	Iter	4	5	8	12	21	35
		Cond	1,05444	1,19107	7,74518	31,7996	203,672	818,237
8	G)	Iter	3	4	7	10	17	27
		Cond	1,04658	1,16168	7,61973	35,7836	247,557	1007,22
	H)	Iter	6	8	18	30	62	113
		Cond	1,59221	7,77024	2359,7	14931,4	107558	445637
	I)	Iter	4	4	7	10	17	27
		Cond	1,04213	1,13116	5,06432	20,3766	136,493	548,757
16	G)	Iter	2	3	6	8	13	21
		Cond	1,01	1,13453	6,00643	31,2384	240,402	999,381
	H)	Iter	6	8	21	39	96	204
		Cond	1,38479	7,22358	2811,18	20544,9	169123	709656
	I)	Iter	3	4	6	8	14	21
		Cond	1,0099	1,10309	2,74025	8,17056	50,4689	196,403

Tabulka 3: Úloha 1D - narušení přesného řešení číslem ε ($\varepsilon = 10^{-2}$). Řešení PCG s tříkrokovou metodou.

H/h	Var							
		$n \times n$	10×10	20×20	30×30	40×40	50×50	60×60
2	G)	Iter	4	5	6	6	7	8
		Cond	1,26654	2,10296	3,55552	5,63639		
	H)	Iter	4	4	5	6	7	8
		Cond	1,2283	1,92194	3,08519	4,71966		
	I)	Iter	4	5	6	6	7	8
		Cond	1,26581	2,09949	3,54637	5,61841		
4	G)	Iter	5	8	8	11	10	12
		Cond	1,22912	3,05465	3,2308	44,6856		
	H)	Iter	5	8	9	12	11	12
		Cond	1,24699	3,76789	4,38512	59,5431		
	I)	Iter	5	8	8	11	10	12
		Cond	1,22689	3,0526	3,18725	44,2594		
8	G)	Iter	3	6	7	8	9	10
		Cond	2,4807	1,69883	2,69266	7,63525		
	H)	Iter	4	8	11	10	13	15
		Cond	2,4829	3,72414	11,4131	38,7226		
	I)	Iter	3	6	7	9	9	10
		Cond	2,48071	1,67236	2,60373	7,47828		
16	G)	Iter	1	3	3	7	8	15
		Cond	1,01	11,8336	2,54817	3,62927		
	H)	Iter	3	5	6	14	17	22
		Cond	1,38479	16,115	13,7639	36,8003		
	I)	Iter	2	4	4	7	8	15
		Cond	1,0099	11,8336	2,55184	3,35322		

Tabulka 4: Úloha 2D - narušení přesného řešení číslem ε ($\varepsilon = 10^{-2}$). Řešení PCG s tříkrokovou metodou.

matice $\hat{A}$, kterou jsem uvedl na začátku této části, diagonální bloky $\hat{A}_{II}$ (jejich inverze) se v matici vyskytují častěji a mají tak větší vliv na řešení. Jak je vidět, ukázalo se, že při předpodmínění diagonálních bloků stačí brát narušení ε^2 abychom dosáhli postačujících výsledků.

5.2 Paralelní výpočet

Zde porovnávám doby běhu programu při sekvenčním algoritmu a při paralelním algoritmu. Pro testování paralalelizace jsem měl k dispozici pouze osobní počítač se třemi jádry. Pro paralelizaci tříkrokové metody je ideální, aby každý proces řešil úlohu na jedné podoblasti, proto jsem paralelizaci testoval pouze při dělení do 2 nebo 3 podoblastí.

n	N	ε	iter	t-sekvenčně	t-paralelně
1500	2	10^{-3}	67	$50\,s$	$38\,s$
1500	2	10^{-2}	185	$160\,s$	$140\,s$
1500	3	10^{-3}	71	$25\,s$	$13\,s$
1500	3	10^{-2}	200	$96\,s$	$62\,s$
3000	3	10^{-3}	135	$560\,s$	$250\,s$
3000	3	10^{-2}	387	$1484\,s$	$570\,s$

Tabulka 5: Paralelizace - srovnání času běhu paralelního a sekvenčního algoritmu pro 1D úlohu

V tabulce 5 vidíme srovnání sekvenčního a paralelního programu pro 1D úlohy z hlediska výpočetního času. Měřený čas, je čas pouze samotného řešení PCG s předpodmíněním tříkrokovou metodou. Do tohoto času se tedy nezapočítává čas potřebný k sestavení matice a jednotlivých bloků. Číslo N v tabulce značí počet podoblastí a zároveň počet procesů, na kterých byl testován paralelní program. Číslo ε je číslo narušení spektra matic $\hat{A}_{II}$ a $\hat{S}$ z části 5.1. Při tomto měření jsem použil předpodmínění H) z předchozí části (kap. 5.1.2). Číslo n je počet neznámých a iter značí počet iterací programu.

Z tabulky je patrné zrychlení programu při jeho paralelizaci. V první části, pro 2 procesy a dělení na 2 podoblasti je naměřené zrychlení jen něco kolem 15 až 20%. V druhé části je měření pro 3 procesy a dělení úlohy do 3 podoblastí. Vidíme, že zrychlení je už o něco více patrné, zejména pro větší úlohy. Při takto malých úlohách se zrychlení tolik neprojeví, čas výpočtu se odvíjí od momentálního zatížení procesoru, při větších úlohách by rozdíly mezi paralelním a sériovým programem byly znatelnější. Avšak i takovéto měření nám stačí pro to, abychom si ověřili účinnost provedené paralelizace.

Při obrovských úlohách a velkém počtu neznámých by mohlo dojít k problémům při přenášení velkých dat, to by mohlo značně zpomalit, nebo úplně zabrzdit komunikaci, což by použití paralelního programu značně limitovalo. Pro důkladnější testování a studii by bylo vhodné vyzkoušet paralelizaci na strojích s možností využití většího počtu procesů a otestovat algoritmus na větších úlohách pro tisíce až desítky tisíc neznámých.

6 Závěr

V této práci jsem se věnoval řešení okrajových úloh s Dirichletovými okrajovými podmínkami pomocí primárních metod rozložení oblasti, ty vedly na řešení PCG s předpodmíněním tříkrokovou metodou. Použitím této metody jsme schopni dosáhnout řešení v jedné iteraci. Tyto úlohy lze řešit i nepřesně, což může být v praxi užitečné z hlediska úspory výpočetního času. Při nepřesném předpodmínění už nedostáváme výsledek v 1 iteraci, počet iterací roste v závislosti na číslu podmíněnosti soustavy. Náplní práce bylo studovat chování soustavy (počet iterací a číslo podmíněnosti) při nepřesném předpodmínění ještě v závislosti na poměru počtu neznámých ku počtu podoblastí.

Z výsledků jsou zřejmé nároky na předpodmínění diagonálních bloků matice soustavy, které musejí být předpodmíněny přesně, nebo alespoň téměř přesně s narušením spektra o 10^{-4}, jinak výsledky nebyly uspokojivé. Závislost počtu iterací na poměru počtu neznámých ku počtu podoblastí se při testování nepodařila prokázat. Při řešení těmito metodami v praxi se však ukazuje, že počet iterací by měl být $O(\log \frac{n}{N})$, kde n je počet neznámých a N počet podoblastí. Proto by bylo vhodné v práci pokračovat a vyzkoušet použité metody řešení na větších úlohách z praxe.

Dále jsem prokázal efektivnosti paralelního řešení, kdy bylo zrychlení patrné už při výpočtu na dvou a třech procesech. Opět ani zde nebyla vidět žádná časová závislost mezi paralelním a sekvenčním přístupem, která by se v praxi při řešení větších úloh měla projevit.

Lukáš Malý

7 Reference

[1] J. H. Bramble, J. E. Pasciak, A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring I. Mathematics of computation*, [online]. 1986, 175, [cit. 2011-03-21]. Dostupný z WWW: http://www.ams.org/home/page

[2] Saad Yousef, *Iterative Methods for Sparse Linear Systems*, [online]. Second edition with corections. [s.l.] : [s.n.], 3.1.2000 [cit. 2011-03-21]. Dostupné z WWW: http://www-users.cs.umn.edu/~ saad/books.html

[3] Blaheta Radim, *Matematické modelování a metoda konečných prvků*, VŠB-TU Ostrava, 2011.

[4] Míka S., Přikryl P., Brandner M., *Speciální numerické metody: Numerické metody řešení okrajových úloh pro diferenciální rovnice*, [online]. 1. vydání. [s.l.] : [s.n.], 2006 [cit. 2011-04-14]. Dostupné z WWW: http://home.zcu.cz/ mika/SNM2/SNM2.pdf. ISBN 80-86843-13-0.

[5] Starý Jiří, *Využití PVM pro paralelní implementaci metody sdružených gradientů s předpodmíněním*, [s.l.], 1996. 58 s. Diplomová práce. VŠB-Technická universita Ostrava

[6] Foster Ian, *Designing and building parallel programs :concepts and tools for parallel software engineering*. Reading : Addison-Wesley Publishing Company, 1995. xiii, 381. ISBN 0-201-57594-9.