VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra Matematiky

Nepřesná FETI metoda založená na rozšířených Lagrangiánech

2009 Martin Menšík

Prohlášení o autorství

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

V Ostravě dne 5. května 2009 Martin Menšík

Poděkování

Rád bych poděkoval panu prof. Dostálovi za mnoho podnětných hodin strávených nad tématem, za trpělivost při korekturách a za to, že vždy viděl o kus dál a já tak došel, kam jsem měl.

Musím poděkovat své rodině za důvěru a podporu, kterou mi dávala nejen během psaní této práce, ale během celého mého života.

A na závěr děkuji všem kamarádům, kteří brali vážně mé výmluvy na psaní a nikam mě nezvali, takže jsem neměl na výběr.

Abstrakt

Metoda rozložení oblastí - FETI je v současnosti hojně používanou metodou při řešení velkkých úloh. Jednou z nevýhod její současné podoby je "zbytečně přesné" řešení dílčích úloh ve chvíly, kdy to není nezbytné. V této diplomové práci je popsán odlišní přístup k FETI za využití rozšířených lagrangiánů. Takovýto postup umožňuje velice efektivně přesnost vnitřních úloh řídit a dramaticky tak snížit náročnost řešení.

Klíčová slova: FETI, rozložení oblasti ,lagrangián, optimalizace, metoda konečných prvků

Abstract

FETI method is currently widely used for large-scale problems solving. One of its main drawbacks is that it solves inner cycles with too high precision. This thesis is concerned with finding and describing different approach , using augmented lagrangians, that enables efective precision control of inner cycles.

Key words: FETI, domain decomposition, lagrangian, optimalization, finite element method

Obsah

Kapitola 1

Lineární algebra

1.1 Matice

Obecně je matice obdélník uspořádaných matematických prvků, pro které máme definovány operace sčítání a násobení, dále skalárů, o n sloupcích a m řádcích. O takové matici pak říkáme, že je řádu $m \times n$. Pokud je počet řádků stejný jako počet sloupců, mluvíme o matici čtvercové a říkáme o ní, že je řádu n.

Matice se poprvé více objevují v matematických textech 18. a 19. století hlavně při vyjádření geometrických struktur a soustav lineárních rovnic. Počátek podrobného zkoumání jejich vlastností a struktury ale spadá až do století dvacátého.

Pokud má matice jen jeden řádek, případně jeden sloupec, nazýváme ji **vektorem**.

Pro jednoduchost je dobré si zavést označení řádků, sloupců a prvků matice

$$\mathbf{A} = \begin{bmatrix} \mathbf{r}_1^{\mathbf{A}} \\ \vdots \\ \mathbf{r}_m^{\mathbf{A}} \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1^{\mathbf{A}} & \cdots & \mathbf{s}_n^{\mathbf{A}} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Příklad 1.1.1. Máme-li matici

$$\mathbf{A} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

potom platí následující vztahy:

$$\mathbf{s}_2^{\mathbf{A}} = \begin{bmatrix} 2 \\ 11 \\ 7 \\ 14 \end{bmatrix}$$

$$\mathbf{r}_3^{\mathbf{A}} = \begin{bmatrix} 9 & 7 & 6 & 12 \end{bmatrix}$$

$$a_{23} = 8$$

Definice Matici nazýváme **symetrickou**, pokud pro každý prvek matice a_{ik} platí

$$a_{ij} = a_{ji}$$

Je zřejmé, že pokud je matice symetrická, musí být také čtvercová.

Příklad 1.1.2. Matice **S** je symetrická

$$\mathbf{S} = \begin{bmatrix} 54 & 35 & 40 & 42 & 39 \\ 35 & 50 & 56 & 36 & 34 \\ 40 & 56 & 75 & 36 & 41 \\ 42 & 36 & 36 & 40 & 28 \\ 39 & 34 & 41 & 28 & 38 \end{bmatrix}$$

1.2 Lineární nezávislost vektorů

Definice Vektory $\mathbf{v}_1, \cdots, \mathbf{v}_n$ nazýváme lineárně nezávislé, pokud je jediným řešením rovnice

$$\alpha_1\mathbf{v}_1 + \cdots + \alpha_n\mathbf{v}_n = \mathbf{o}$$

triviální řešení

$$\alpha_1 = \cdots = \alpha_n = 0$$

Příklad 1.2.1. Například sloupce (ale i řádky) jednotkové matice

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

jsou nezávislé, protože neexistuje žádné jiné, než triviální, řešení rovnice

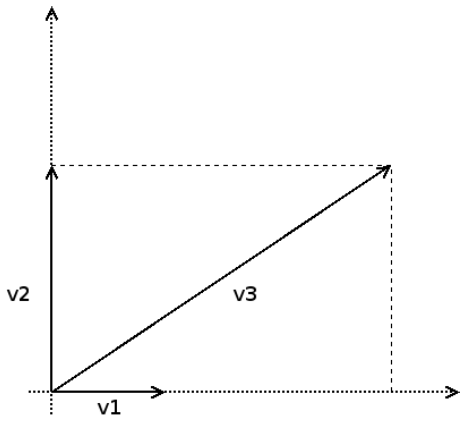
$$\mathbf{Ix} = \mathbf{o}$$

Pokud vektory nejsou nezávislé, jsou **závislé**.

Význam závislosti vektorů lze snadno chápat tak, že některý vektor lze vyjádřit součtem násobků ostatních. Takovému vyjádření vektoru říkáme **lineární kombinace**. V geometrii jsou například závislé směrnice rovnoběžných přímek. Naproti tomu vektory kolmé k rovině jsou nezávislé k směrnicím přímek této rovině náležících.

Příklad 1.2.2. Na obrázku 1.1 je vidět, že vektor $\mathbf{v_3}$ lze vyjádřit jako lineární kombinaci vektorů $\mathbf{v_1}, \mathbf{v_2}$. Konkrétně

$$\mathbf{v_3} = 3\mathbf{v_1} + \mathbf{v_2}$$



Obrázek 1.1: Lineární kombinace vektorů

1.3 Regulární matice a její inverze

Mějme čtvercovou matici řádu n $\mathbf{A} \in \mathbb{R}^{n \times n}$ s řádkovými vektory $\mathbf{r}_1^{\mathbf{A}}, \cdots, \mathbf{r}_n^{\mathbf{A}}$. Zkusme z nich nyní vybrat vektory $\mathbf{u}_1, \cdots, \mathbf{u}_m$ takové, které jsou nezávislé. Můžeme k tomu využít jednoduchý postup:

Pro i-tý krok, již jsme nalezli k nezávislých vektorů $\mathbf{u}$

1. Označ vektor $\mathbf{r}_i$ za $\mathbf{u}_{k+1}$

2. Zjisti zda jsou vektory $\mathbf{u}_1, \cdots, \mathbf{u}_{k+1}$ lineárně nezávislé

3. Pokud jsou nezávislé, zvyš k o 1. V opačném případě zahoď vektor $\mathbf{u}_{k+1}$

4. Znovu zopakuj pro $i+1$ dokud neprojdeš všechny vektory matice $\mathbf{A}$

Počtu m nezávislých vektorů říkáme **hodnost** matice.

Příklad 1.3.1. Matice

$$\mathbf{A} = \begin{bmatrix} 2 & 5 & 5 & 2 & 4 \\ 0 & 2 & 4 & 0 & 1 \\ 1 & 1 & 3 & 1 & 1 \\ 2 & 7 & 9 & 2 & 5 \\ 3 & 5 & 13 & 3 & 4 \end{bmatrix}$$

Je například hodnosti 3, protože řádky $\mathbf{r}_1^{\mathbf{A}}, \mathbf{r}_2^{\mathbf{A}}, \mathbf{r}_3^{\mathbf{A}}$ jsou nezávislé a jejich lineární kombinací lze zbylé dva řádky vyjádřit:

$$\mathbf{r}_1^{\mathbf{A}} + \mathbf{r}_2^{\mathbf{A}} = \begin{bmatrix} 2 & 5 & 5 & 2 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 4 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 9 & 2 & 5 \end{bmatrix} = \mathbf{r}_4^{\mathbf{A}}$$

$$\mathbf{r}_2^{\mathbf{A}} + 3\mathbf{r}_3^{\mathbf{A}} = \begin{bmatrix} 0 & 2 & 4 & 0 & 1 \end{bmatrix} + 3\begin{bmatrix} 1 & 1 & 3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 5 & 13 & 3 & 4 \end{bmatrix} = \mathbf{r}_5^{\mathbf{A}}$$

Definice Čtvercovou matici A řádu n nazýváme **regulární** pokud je její hodnost n.

Význam regulární matice je zřejmý například v soustavě lineárních rovnic $\mathbf{Ax} = \mathbf{b}$, kde regularita matice $\mathbf{A}$ zaručí existenci a jedinečnost řešení $\mathbf{x}$. A právě takto se dá dojít k pojmu **inverzní matice**.

Definice Matici $\mathbf{A}^{-1}$ nazveme **inverzní** k matici $\mathbf{A}$, pokud platí

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

kde $\mathbf{I}$ je jednotková matice.

Spojitost s výše zmíněnou soustavou je nasnadě:

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Inverzní matice může být velmi užitečná při teoretických rozborech a úvahách, ale v praxi se využívá jen velmi zřídka nebo vůbec, kvůli její příliš velké výpočetní složitosti.

Příklad 1.3.2. Regularitu matice

$$\mathbf{A} = \begin{bmatrix} 4 & 4 & 3 & 4 \\ 1 & 3 & 2 & 3 \\ 2 & 2 & 2 & 1 \\ 2 & 0 & 3 & 2 \end{bmatrix}$$

si můžeme snadno ověřit například úpravou na schodový tvar (viz. [1, str.13]).

$$\mathbf{A_{schod}} = \begin{bmatrix} 4 & 4 & 3 & 4 \\ 0 & 8 & 5 & 8 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Její inverze je

$$\mathbf{A}^{-1} = \frac{1}{30} \begin{bmatrix} 13 & -16 & -2 & -1 \\ -3 & 6 & 12 & -9 \\ -14 & 8 & 16 & 8 \\ 8 & 4 & -22 & 4 \end{bmatrix}$$

Příklad 1.3.3. Prakticky můžeme výsledek příkladu 1.3.2 využít například v rovnici

$$\mathbf{Ax} = \begin{bmatrix} 5 \\ 7 \\ -3 \\ 2 \end{bmatrix}$$

Když rovnici zleva vynásobíme inverzí $\mathbf{A}^{-1}$ získáme:

$$\mathbf{x} = \mathbf{A}^{-1} \begin{bmatrix} 5 \\ 7 \\ -3 \\ 2 \end{bmatrix}$$

a po vynásobení

$$\mathbf{x} = \frac{1}{30} \begin{bmatrix} -43 \\ -27 \\ -46 \\ 142 \end{bmatrix}$$

1.4 Vektorový prostor

S vektory se setkáváme již v jednoduché geometrii nebo ve fyzice. V těchto disciplínách je pro nás důležitý jejich význam. Vektorový prostor je naproti tomu velkým krokem k abstrakci. Vektor zde pozbývá konkrétních rysů a stává se pouze objektem s přesně definovaným chováním. Jedním modelem tak můžeme postihnou mnohem širší okruh problémů.

Definice Vektorovým prostorem nazveme množinu vektorů V, množinu skalárů R a operace násobení vektoru se skalárem $\times$ a sčítání vektorů $+$ takové, které splňují:

Pro každé $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ a $\alpha, \beta \in R$:

1. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{v} + \mathbf{u}) + \mathbf{w}$
2. Existuje $\mathbf{o} \in V$ takové že $\mathbf{u} + \mathbf{o} = \mathbf{o} + \mathbf{u} = \mathbf{u}$
3. Pro každé $\mathbf{u} \in V$ existuje $-\mathbf{u} \in V$ takové $\mathbf{u} + (-\mathbf{u}) = \mathbf{o}$
4. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
5. $\alpha(\beta\mathbf{u}) = (\alpha\beta)\mathbf{u}$
6. $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$
7. $\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$
8. $1\mathbf{u} = \mathbf{u}$

Definice Mějme množinu vektorů $U = \{\mathbf{u}_1, \cdots, \mathbf{u}_m\}$. Množinu všech vektorů $\mathbf{v}$, které lze vyjádřit lineární kombinací vektorů množinu U :

$$\mathbf{v} = \alpha_1\mathbf{u}_1 + \cdots + \alpha_m\mathbf{u}_m$$

budeme říkat **lineární obal** množiny U a budeme jej značit $< U >$.

Pokud jako množinu vektorů nějakého vektorového prostoru zvolíme lineární obal nějaké množiny, můžeme si být jisti, že je pro všechny operace úplná, protože již v definici lineárního obalu je řečeno, že obsahuje právě všechny vektory, které lze pomocí daných operací získat. Odsud je již jen malý krok k pojmu **báze**. Bází vektorového prostoru $\mathcal{V}$nazýváme množinu lineárně **nezávislých** vektorů, jejichž lineární obal tvoří množinu vektorů prostoru $\mathcal{V}$.

Každý vektorový prostor má bází pochopitelně nekonečně mnoho. Často je vhodné bázi, se kterou budeme pracovat nejprve upravit tak, aby splňovala podmínky, které nám práci usnadní. Může to být například její ortogonalita nebo ortonormalita, ale k tomu je třeba definovat pojmy skalární součin a norma a to není obsahem tohoto textu.

1.5 Nulový prostor

Definice Zobrazení $\mathbf{A} : \mathcal{V} \to \mathcal{U}$, kde $\mathcal{U}$ a $\mathcal{V}$ jsou vektorové prostory, nazýváme **lineárním zobrazením**, pokud pro libovolný vektor $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ a skalár α platí tyto podmínky:

1. $\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}$
2. $\mathbf{A}(\alpha\mathbf{x}) = \alpha\mathbf{A}\mathbf{x}$

Pro nás budou zajímavá pouze lineární zobrazení prostorů aritmetických vektorů, která můžeme reprezentovat jakou součin matice zobrazení s vektorem.

Definice Nulovým prostorem zobrazení $\mathbf{A} : \mathcal{V} \to \mathcal{U}$ nazýváme takový podprostor $\mathcal{N} \subset \mathcal{V}$, aby pro každý vektor $\mathbf{x} \in \mathcal{N}$ platilo

$$\mathbf{A}\mathbf{x} = \mathbf{o}$$

Význam nulového prostoru je snadno pochopitelný. Jde o prostor takových vektorů, které zobrazení $\mathbf{A}$ zobrazí na nulový vektor. V literatuře se také objevuje označení **defekt zobrazení**. Jelikož $\mathcal{N}$ je podprostorem $\mathcal{V}$, může být jeho dimenze nejvýš stejná jako dimenze prostoru $\mathcal{V}$, v tom případě by ale matice $\mathbf{A}$ musela být nulová.

1.6 Definitní matice

Definice Matice $\mathbf{A} \in \mathbb{R}^{n\times n}$ je pozitivně definitní (resp. semidefinitní), pokud $\forall x \in \mathbb{R}, x \neq \sigma : x\mathbf{A}x^T > 0$ (resp. $\forall x \in \mathbb{R} : x^T\mathbf{A}x \geq 0$).

Věta 1.6.1. *Pro každou matici $\mathbf{A} \in \mathbb{R}^{n\times n}$ platí, že $\mathbf{A}\mathbf{A}^T$ je positivně semidefinitní. Pokud je navíc $\mathbf{A}$ regulární, tak je $\mathbf{A}\mathbf{A}^T$ positivně definitní.*

Důkaz. $\forall x \in \mathbb{R} : x^T\mathbf{A}\mathbf{A}^T x = (Ax)^T Ax = ||Ax||_2^2 \geq 0$. Jestliže je $\mathbf{A}$ regulární, tak jsou jednotlivé sloupce matice $\mathbf{A}$ lineárně nezávislé a z definice vyplývá

$$||\mathbf{A}x|| = 0 \Rightarrow \mathbf{A}x = \sigma \Rightarrow x = \sigma$$

□

Věta 1.6.2. *Matice*

$$\mathbb{R}^{n\times n} \ni \mathbf{A} = \begin{bmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & \mathbf{A}^* \end{bmatrix}$$

je pozitivně definitní, právě když $\alpha > 0$ a $\mathbf{A}^* - \frac{1}{\alpha}\mathbf{a}\mathbf{a}^T$ je pozitivně definitní.

Důkaz. Podle [2]

Je-li $\mathbf{A}$ pozitivně definitní matice, potom

$$\alpha = \begin{bmatrix} 1 & 0 & \cdots \end{bmatrix} \mathbf{A} \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix} > 0$$

a pro každé $0 \neq \mathbf{x} \in \mathbb{R}^{n-1}$ platí

$$\mathbf{x}^T(\mathbf{A}^* - \frac{1}{\alpha}\mathbf{a}\mathbf{a}^T)\mathbf{x} = \mathbf{x}^T\mathbf{A}^*\mathbf{x} - \frac{1}{\alpha}(\mathbf{a}^T\mathbf{x})^2$$

. Výraz napravo můžeme dále upravit na součin

$$\begin{bmatrix} -\frac{1}{\alpha}\mathbf{a}^T\mathbf{x} \\ \mathbf{x} \end{bmatrix}^T \begin{bmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & \mathbf{A}^* \end{bmatrix} \begin{bmatrix} -\frac{1}{\alpha}\mathbf{a}^T\mathbf{x} \\ \mathbf{x} \end{bmatrix}$$

což je totéž jako

$$\begin{bmatrix} -\frac{1}{\alpha}\mathbf{a}^T\mathbf{x} \\ \mathbf{x} \end{bmatrix}^T \mathbf{A} \begin{bmatrix} -\frac{1}{\alpha}\mathbf{a}^T\mathbf{x} \\ \mathbf{x} \end{bmatrix}$$

a tomto součinu víme, že je vetší než nula. Matice $\mathbf{A}^*$ je tedy také pozitivně definitní.

Naopak, je-li $\alpha > 0$ a $\mathbf{A}^* - \frac{1}{\alpha}\mathbf{a}\mathbf{a}^T$ je pozitivně definitní, potom pro každé $\mathbf{x} \in \mathbb{R}^{n\times n}$, píšeme-li ho ve tvaru $\mathbf{x} = (\xi, \mathbf{x'}^T)^T$, kde $\mathbf{x'} \in \mathbf{R}^{n-1}$, platí

$$\begin{aligned}\mathbf{x}^T\mathbf{A}\mathbf{x} &= \begin{bmatrix} \xi \\ \mathbf{x'} \end{bmatrix}^T \begin{bmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & \mathbf{A}^* \end{bmatrix} \begin{bmatrix} \xi \\ \mathbf{x'} \end{bmatrix} = \alpha\xi^2 + 2\xi\mathbf{a}^T\mathbf{x'} + \mathbf{x'}^T\mathbf{A}^*\mathbf{x'} \\ &= (\sqrt{\alpha}\xi + \frac{1}{\alpha}\mathbf{a}^T\mathbf{x'})^2 + \mathbf{x'}^T(\mathbf{A}^* - \frac{1}{\alpha}\mathbf{a}\mathbf{a}^T)\mathbf{x'} \leq 0\end{aligned}$$

takže $\mathbf{A}$ je pozitivně semidefinitní a $\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$ implikuje $\mathbf{x'} = \mathbf{o}$ a $\xi = 0$, tedy $\mathbf{A}$ je pozitivně definitní. □

Díky této rekurzivní podmínce můžeme při určování definitnosti matice postupně snižovat její řád tak se dostat postupně až k matici o jednom prvku. Tento je velmi podobný tomu, kterým v další kapitole určíme rozklad takové matice.

Příklad 1.6.1. Zkusme výše uvedeným způsobem zjistit, zda je pozitivně definitní matice

$$\mathbf{A} = \begin{bmatrix} 16 & 4 & 8 & 8 \\ 4 & 10 & 8 & 2 \\ 8 & 8 & 12 & 10 \\ 8 & 2 & 10 & 17 \end{bmatrix}$$

Nyní budeme postupně snižovat řád matice a ověřovat její pozitivní definitnost:

1. $\alpha_1 = 57 > 0$. Vytvoříme matici

$$\mathbf{A_2} = \begin{bmatrix} 9 & 6 & 0 \\ 6 & 8 & 6 \\ 0 & 6 & 13 \end{bmatrix}$$

2. $\alpha_2 = 9 > 0$. Vytvoříme matici

$$\mathbf{A_3} = \begin{bmatrix} 4 & 6 \\ 6 & 13 \end{bmatrix}$$

3. $\alpha_3 = 4 > 0$. Vytvoříme poslední matici

$$\mathbf{A_4} = \begin{bmatrix} 4 \end{bmatrix}$$

4. $\alpha_4 = 4 > 0$.

Všechny vytvořené podmatice byly pozitivně definitní, tím jsem ukázali, že byla i původní matice $\mathbf{A}$.

Příklad 1.6.2. Zkusme nyní stejný postup, na velmi podobné matici

$$\mathbf{A} = \begin{bmatrix} 16 & 4 & 8 & 8 \\ 4 & 10 & 8 & 2 \\ 8 & 8 & 8 & 4 \\ 8 & 2 & 4 & 8 \end{bmatrix}$$

1. $\alpha_1 = 16 > 0$. Vytvořme

$$\mathbf{A_2} = \begin{bmatrix} 9 & 6 & 0 \\ 6 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

2. $\alpha_2 = 9 > 0$. Vytvořme

$$\mathbf{A_3} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$$

3. $\alpha_3 = 0 \not> 0$

V třetím kroku jsme narazili na nulový prvek. Tím pádem tato matice není pozitivně definitní.

1.6.1 Sylvestrovo kritérium

Další možností, jak si ověřit definitnost je **Sylvestrovo kritérium**. Princip této metody je založen na znalostech vlastních čísel a tím vybočuje z rozsahu této práce. Důkaz i s potřebnou teorií je v [1].

Věta 1.6.3. *Nechť* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *a nechť*

$$\mathbf{A_i} = \begin{bmatrix} a_{11} & \cdots & a_{1i} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ii} \end{bmatrix}, \quad i = 1, \cdots, n$$

potom matice $\mathbf{A}$ *je pozitivně definitní právě pokud*

$$\det \mathbf{A_i} > 0, \quad i = 1, \cdots, n$$

Důkaz. viz. [1, str.154] □

Příklad 1.6.3. Ověřme nyní znovu pozitivní definitnost matice z příkladu 1.6.1.

$$\mathbf{A} = \begin{bmatrix} 16 & 4 & 8 & 8 \\ 4 & 10 & 8 & 2 \\ 8 & 8 & 12 & 10 \\ 8 & 2 & 10 & 17 \end{bmatrix}$$

1. $$\det \mathbf{A_1} = \begin{vmatrix} 16 \end{vmatrix} = 16 > 0$$

2. $$\det \mathbf{A_2} = \begin{vmatrix} 16 & 4 \\ 4 & 10 \end{vmatrix} = 144 > 0$$

3. $$\det \mathbf{A_3} = \begin{vmatrix} 16 & 4 & 8 \\ 4 & 10 & 8 \\ 8 & 8 & 12 \end{vmatrix} = 720 > 0$$

4.

$$\det \mathbf{A_4} = \det \mathbf{A} = \begin{vmatrix} 16 & 4 & 8 & 8 \\ 4 & 10 & 8 & 2 \\ 8 & 8 & 12 & 10 \\ 8 & 2 & 10 & 17 \end{vmatrix} = 4176 > 0$$

Všechny determinanty podmatic na hlavní diagonále jsou kladné, takže i Sylvestrovo kritérium tuto matici potvrdilo jako pozitivně definitní.

1.6.2 Vlastnosti pozitivně semidefinitních matic

Zopakujme, že matice je **pozitivně semidefinitní**, pokud

$$\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$$

pro všechny vektory $\mathbf{x}$. Semidefinitní matice mají několik vlastností, které budou později užitečné

Věta 1.6.4. *Pokud je* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *pozitivně semidefinitní potom platí :*

$$|a_{ij}| \leq \frac{a_{ii} + a_{jj}}{2} \tag{1.1}$$

$$|a_{ij}| \leq \sqrt{a_{ii} a_{jj}} \quad (i \neq j) \tag{1.2}$$

$$\max_{i,j} |a_{ij}| = \max_{i} a_{ii} \tag{1.3}$$

$$a_{ii} = 0 \quad \Rightarrow \quad \mathbf{r}_i^{\mathbf{A}} = \mathbf{o}, \quad \mathbf{s}_i^{\mathbf{A}} = \mathbf{o} \tag{1.4}$$

Důkaz. Vezměme například vektor $\mathbf{x} = \mathbf{s}_i^{\mathbf{I}} + \mathbf{s}_j^{\mathbf{I}}$. Potom získáme rovnici

$$0 \leq \mathbf{x}^T \mathbf{A} \mathbf{x} = a_{jj} + a_{ii} + 2a_{ij}$$

Při volbě vektoru $\mathbf{x} = \mathbf{s}_i^{\mathbf{I}} - \mathbf{s}_j^{\mathbf{I}}$ rovnici

$$0 \leq \mathbf{x}^T \mathbf{A} \mathbf{x} = a_{jj} + a_{ii} - 2a_{ij}$$

Z těchto dvou rovnic vyplývá vztah 1.1. Vztah 1.3 z něj vyplývá. Kdyby totiž byl prvek mimo diagonálu největší, nemohl by potom být menší, než aritmetický průměr dvou jiných.

O něco složitější je situace se vztahem 1.2. Vezměme si vektor $\mathbf{x} = y\mathbf{s}_i^{\mathbf{I}} + \mathbf{s}_j^{\mathbf{I}}$. Získáváme tak rovnici

$$-0 \leq \mathbf{x}^T \mathbf{A}\mathbf{x} = y^2 a_{ii} + 2y a_{ij} + a_{jj}$$

To je kvadratická nerovnice, jejíž diskriminant tedy nesmí být kladný

$$0 \geq 4a_{ij}^2 - 4a_{ii}a_{jj}$$

odsud již snadno plyne vztah 1.2. A konečně implikace 1.4 je jeho důsledkem. □

Je zajímavé se zamyslet nad významem těchto vztahů. Je například zřejmé, že pozitivně semidefinitní matice mají nezáporné prvky na hlavní diagonále. Hlavní diagonála je navíc vždy dominantní, největší prvky se budou vyskytovat právě zde.

Příklad 1.6.4. Pro názornost je možné si všechny vztahy vyzkoušet například na matici

$$\mathbf{A} = \begin{bmatrix} 16 & 0 & 8 & 8 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 13 & 10 \\ 8 & 0 & 10 & 12 \end{bmatrix}$$

Kapitola 2

Metody optimalizace

2.1 Úvod

V této kapitole probereme nejvýznamnější metody optimalizace. Optimalizací zde myslíme minimalizaci funkcionálu na nějakém prostoru řešení. Konkrétně se budeme zabývat převážně minimalizací kvadratického funkcionálu:

$$q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{2.1}$$

respektive řešením soustavy

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.2}$$

kde matice $\mathbf{A} \in \mathbb{R}^{n\times n}$ je symetrická a pozitivně definitní a $\mathbf{b} \in \mathbb{R}^n$.

Věta 2.1.1. *Řešení minimalizační úlohy* $\bar{\mathbf{x}}$ *(tedy takové, pro které platí* $q(\bar{\mathbf{x}}) \leq q(\mathbf{x}) \,\forall \mathbf{x} \in \mathbb{R}^n$*) je ekvivalentní s řešením soustavy* $\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$.

Důkaz. "⇒" Funkcionál q je kvadratický a má proto derivaci v celém oboru $\mathbb{R}^n$. Má-li tedy v bodě $\bar{\mathbf{x}}$ minimum, musí zde být jeho derivace nulová.

$$q'(\bar{\mathbf{x}}) = \mathbf{A}\bar{\mathbf{x}} - \mathbf{b} = 0$$

Z toho již přímo vyplývá

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

"⇐" Nyní předpokládejme, že $\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$. Vezměme libovolné $\mathbf{x} \in \mathbb{R}^n$ a navíc si označme rozdíl $\mathbf{x} - \bar{\mathbf{x}} = \mathbf{h}$.

$$q(\mathbf{x}) - q(\bar{\mathbf{x}}) = \frac{1}{2}(\bar{\mathbf{x}} + \mathbf{h})^T\mathbf{A}(\bar{\mathbf{x}} + \mathbf{h}) - \mathbf{b}^T(\bar{\mathbf{x}} + \mathbf{h}) - \frac{1}{2}\bar{\mathbf{x}}^T\mathbf{A}\bar{\mathbf{x}} + \mathbf{b}^T\bar{\mathbf{x}} =$$

$$= \frac{1}{2}\mathbf{h}^T\mathbf{A}\mathbf{h} - \mathbf{b}^T\mathbf{h} + \bar{\mathbf{x}}^T\mathbf{A}\mathbf{h} = \underbrace{\frac{1}{2}\mathbf{h}^T\mathbf{A}\mathbf{h}}_{\geq 0(\mathbf{A}\text{jepos.def.})} + \underbrace{(\bar{\mathbf{x}}^T\mathbf{A} - \mathbf{b})}_{=0(\text{viz.předpoklad})}\ \mathbf{h}$$

Dokázali jsme tedy, že

$$q(\bar{\mathbf{x}}) \leq q(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathbb{R}^n$$

□

Jednotlivé algoritmy budeme testovat na soustavě lineárních rovnic s maticí $\mathbf{A}$ dimenze 50 , číslem podmíněnosti přibližně 1000 a dominantní hlavní diagonálou.

2.2 Jacobi a Gauss-Saidel

Jacobiho a Gauss-Saidelovy metody patří k vůbec nejzákladnějším iteračním metodám pro řešení soustav ve tvaru $\mathbf{Ax} = \mathbf{b}$. Vycházejí z jednoduchého principu, kdy se v každém kroku snaží co nejvíce eliminovat reziduální vektor $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$.

Pro jednodušší zápis si zaveďme rozložení matice $\mathbf{A}$:

$$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F} \tag{2.3}$$

kde $\mathbf{D}$ je diagonální, $\mathbf{E}$ striktně dolní trojúhelníková a $\mathbf{F}$ striktně horní trojúhelníková část matice $\mathbf{A}$.

2.2.1 Jacobi

Myšlenkou Jacobiho iterační metody je vynulování i-té složky přibližného řešení v $k+1$ kroku:

$$(\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1})_i = 0 \tag{2.4}$$

rozložíme-li si tento výraz po prvcích a kromě i-té složky řešení využijeme hodnoty z krok k, dostaneme výraz

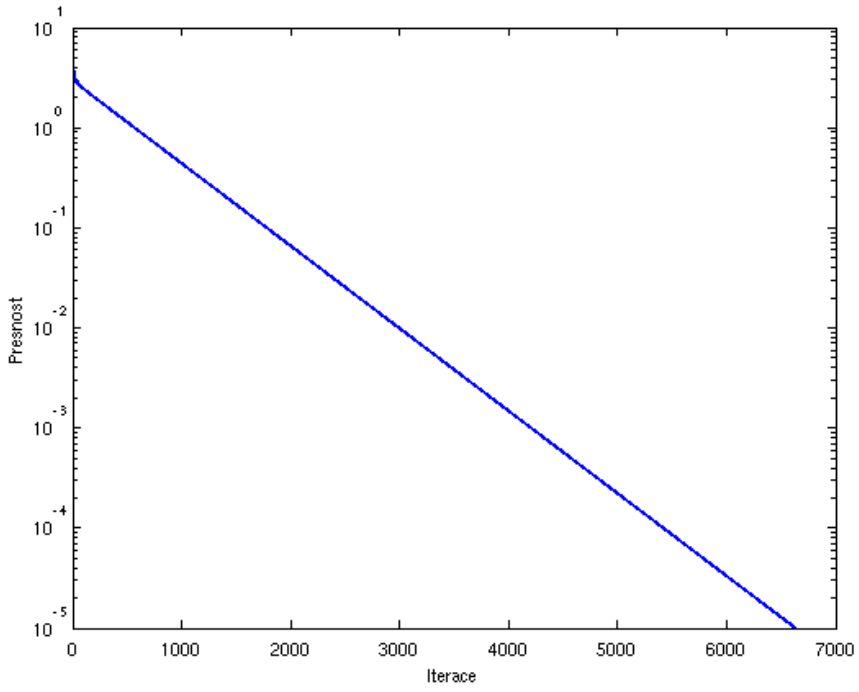
$$(\mathbf{b})_i - (\mathbf{A})_{ii}(\mathbf{x}^{k+1})_i - \sum_{j=1;j\neq i}^{n} (\mathbf{A})_{ij}(\mathbf{x}^k)_j = 0 \tag{2.5}$$

po úpravě tedy

$$(\mathbf{x}^{k+1})_i = \frac{1}{(\mathbf{A})_{ii}} \left((\mathbf{b})_i - \sum_{j=1;j\neq i}^{n} (\mathbf{A})_{ij}(\mathbf{x}^k)_j \right), \qquad i = 1, \ldots, n \tag{2.6}$$

Tento výraz se vztahuje postupně na všechny složky vektoru $\mathbf{x}^{k+1}$ a není problém si jej přepsat do podstatně užitečnější formy s využitím zavedeného rozkladu matice $\mathbf{A}$:

$$\mathbf{x}^{k+1} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x}^k + \mathbf{D}^{-1}\mathbf{b} \tag{2.7}$$



Obrázek 2.1: Vývoj chyby u Jacobiho metody - 6626 iterací

2.2.2 Gauss-Seidel

Gauss-Seidelova metoda je velice podobná metodě Jacobiho. Rozdíl spočívá v tom, že k nulování i-té složky aproximace řešení v $k+1$ kroku se využívá již spočítaných složek z $k+1$ kroku. Rozdíl je dobře patrný z porovnání rovnice 2.5 a rovnice 2.8.

$$(\mathbf{b})_i - \sum_{j=1}^{i-1}(\mathbf{A})_{ij}(\mathbf{x}^{k+1})_j - (\mathbf{A})_{ii}(\mathbf{x}^{k+1})_i - \sum_{j=i+1}^{n}(\mathbf{A})_{ij}(\mathbf{x}^{k})_j = 0 \tag{2.8}$$

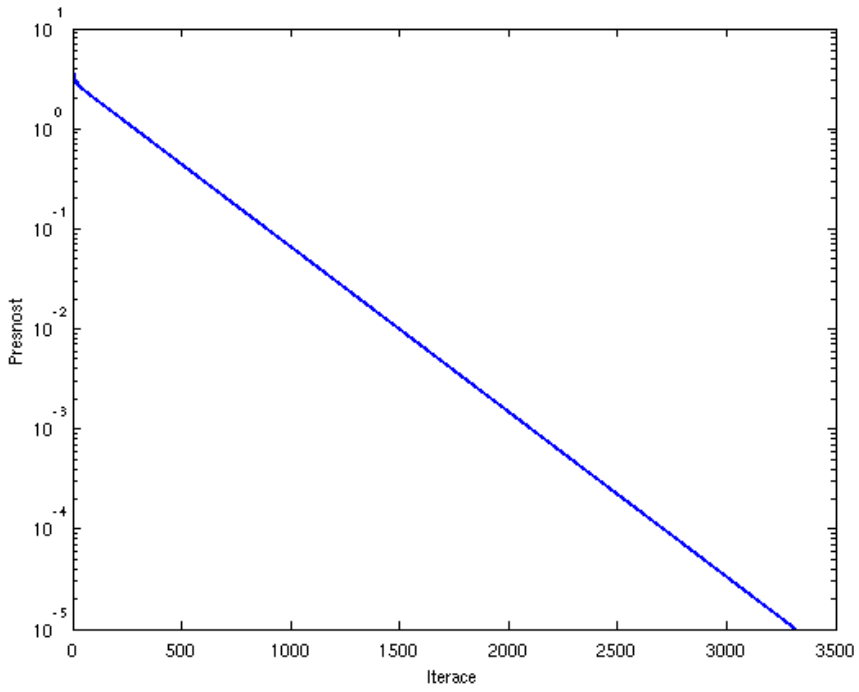
Rozdíl v metodách je dobře patrný v porovnání rovnic 2.5 a 2.8. Stejně jako u Jacobiho metody si vyjádříme i-tý prvek řešení:

$$(\mathbf{x}^{k+1})_i = \frac{1}{(\mathbf{A})_{ii}}\left((\mathbf{b})_i - \sum_{j=1}^{i-1}(\mathbf{A})_{ij}(\mathbf{x}^{k+1})_j - \sum_{j=i+1}^{n}(\mathbf{A})_{ij}(\mathbf{x}^{k})_j\right), \qquad i = 1,\ldots,n \tag{2.9}$$

a převedeme do přehlednějšího vektorového zápisu:

$$\mathbf{x}^{k+1} = (\mathbf{D}-\mathbf{E})^{-1}\mathbf{F}\mathbf{x}^{k} + (\mathbf{D}-\mathbf{E})^{-1}\mathbf{b} \tag{2.10}$$

Tato varianta se nazývá dopředná Gauss-Seidelova metoda, protože se využívají již spočtené předcházející prvky řešení. Obdobně se využívá zpětná Gauss-Seidolova metoda a varianta symetrická, kde se počítají postupně prvky od začátku i konce.



Obrázek 2.2: Vývoj chyby u GS metody - 3314 iterací

2.3 Metoda největšího spádu

Metod největšího spádu lze zařadit mezi takzvané **projekční metody**. To je taková třída metod, kde hledáme řešení ve stanoveném prostoru (zpravidla s o mnoho menší dimenzí, než původní prostor) a to tak, aby výsledné residuum bylo k tomuto prostoru kolmé.

Zabývejme se opět úlohou 2.2.V každém kroku projekční metody si stanovíme prostor $\mathcal{K} \subset \mathbb{R}^n$ a v tomto kroku tedy budeme hledat

$$\mathbf{x}^k \in \mathcal{K} \qquad \text{takové, aby} \qquad \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{r} \perp \mathcal{K} \tag{2.11}$$

Často pracujeme s nějakou počáteční aproximací $\mathbf{x}^0$, kterou chceme zahrnout a hledáme proto řešení ve tvaru

$$\mathbf{x}^k = \mathbf{x}^0 + \delta \quad \delta \in \mathcal{K} \tag{2.12}$$

a reziduum si můžeme vyjádřit jako

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{b} - \mathbf{A}(\mathbf{x}^0 + \delta) = \mathbf{r}^0 - \mathbf{A}\delta \tag{2.13}$$

kde $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ je počáteční reziduum. Úloha 2.11 se potom dá vyjádřit jako hledaní $\delta \in \mathcal{K}$ takového, že

$$(\mathbf{r}^0 - \mathbf{A}\delta, \mathbf{w}) = 0, \quad \forall \mathbf{w} \in \mathcal{K} \tag{2.14}$$

Konkrétně v **metodě největšího spádu** je za prostor $\mathcal{K}$ volen vektor předcházejícího residua $\mathbf{r}^k$. Následující aproximaci si můžeme vyjádřit jako

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{r}^k, \quad \alpha \in \mathbb{R} \tag{2.15}$$

hodnotu α si vyjádříme po dosazení do 2.11 :

$$\begin{aligned} (\mathbf{b} - \mathbf{A}\mathbf{x}^{k+1}, \mathbf{r}^k) &= 0 \\ (\mathbf{r}^k - \alpha \mathbf{A}\mathbf{r}^k, \mathbf{r}^k) &= 0 \\ (\mathbf{r}^k, \mathbf{r}^k) - \alpha(\mathbf{A}\mathbf{r}^k, \mathbf{r}^k) &= 0 \end{aligned}$$

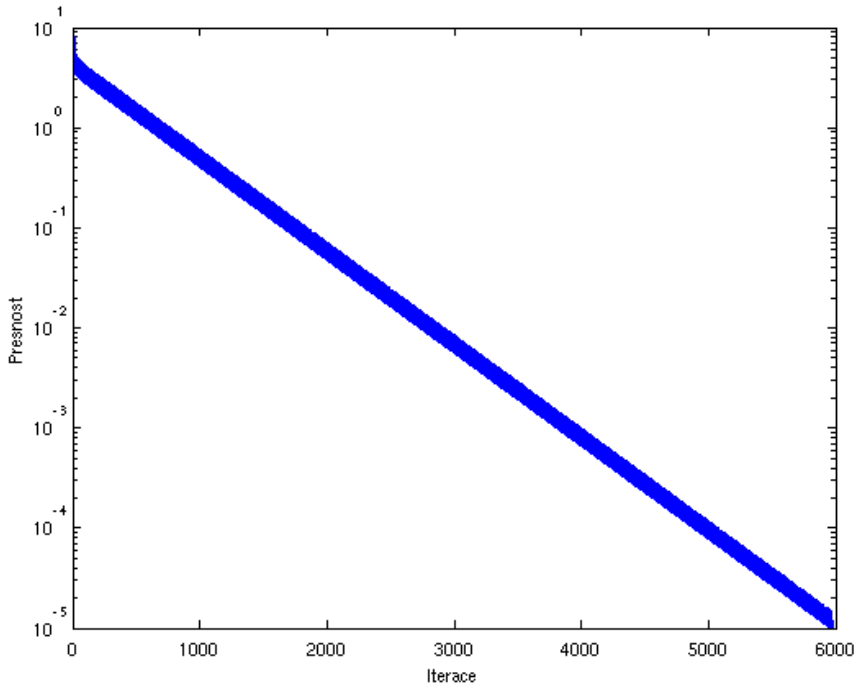
$$\alpha = \frac{(\mathbf{r}^k, \mathbf{r}^k)}{(\mathbf{A}\mathbf{r}^k, \mathbf{r}^k)} \tag{2.16}$$

2.4 Metoda sdružených gradientů

2.4.1 A-ortogonální báze

Definice Vektory $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{R}^n$ nazýváme **A-ortogonální**, pokud

$$(\mathbf{v_1}, \mathbf{v_2})_\mathbf{A} = \mathbf{v_1}^T \mathbf{A}\mathbf{v_2} = 0 \tag{2.17}$$



Obrázek 2.3: Vývoj chyby v metodě největšího spádu - 5974 iterací

Definice Mějme vektory $\mathbf{p_1}, \ldots, \mathbf{p_k} \in \mathcal{K} \subset \mathbb{R}^n$, které jsou bází prostoru $\mathcal{K}$. Tuto bázi nazýváme **A - ortogonální bází**, pokud

$$(\mathbf{p_i}, \mathbf{p_j})_\mathbf{A} = 0, \quad i, j = 1, \ldots, k \quad i \neq j \tag{2.18}$$

Odvození metody sdružených gradientů začneme ukázkou minimalizace funkcionálu 2.1 na podprostoru $\mathcal{K}$ za předpokladu, že známe jeho **A**-ortogonální bázi. Libovolný vektor $\mathbf{x}$ si tedy můžeme vyjádřit jako lineární kombinaci :

$$\mathbf{x} = \xi_1 \mathbf{p_1} + \cdots + \xi_k \mathbf{p_k}, \quad \xi_1, \ldots, \xi_k \in \mathbb{R} \tag{2.19}$$

což můžeme dosadit do q z 2.1 a s využitím **A**-ortogonality

$$\begin{aligned} q(\mathbf{x}) &= \left(\frac{1}{2}\xi_1^2 \mathbf{p_1}^T \mathbf{A}\mathbf{p_1} - \xi_1 \mathbf{b}^T \mathbf{p_1}\right) + \cdots + \left(\frac{1}{2}\xi_k^2 \mathbf{p_k}^T \mathbf{A}\mathbf{p_k} - \xi_k \mathbf{b}^T \mathbf{p_k}\right) = \\ &= q(\xi_1 \mathbf{p_1}) + \cdots + q(\xi_k \mathbf{p_k}) \end{aligned} \tag{2.20}$$

Funkcionál q má derivaci v celém prostoru $\mathbb{R}^n$, v jeho minimu tedy bude derivace nulová. Budeme jej postupně minimalizovat podle jednotlivých členů ξ_i :

$$\frac{\partial q(\mathbf{x})}{\partial \xi_i} = \xi_i \mathbf{p_i}^T \mathbf{A} \mathbf{p_i} - \mathbf{b}^T = 0 \tag{2.21}$$

odsud tedy pro každou složku

$$\xi_i = \frac{\mathbf{b}^T \mathbf{p_i}}{\mathbf{p_i}^T \mathbf{A} \mathbf{p_i}} \tag{2.22}$$

Situace se mírně komplikuje tím, že většinou potřebujeme hledat optimální řešení na prostoru, který je definován kolem počátečního odhadu $\mathbf{x_0}$, tedy:

$$\mathbf{x} = \mathbf{x_0} + \xi_1 \mathbf{p_1} + \cdots + \xi_k \mathbf{p_k} \tag{2.23}$$

po dosazení a využití **A**-ortogonality

$$\begin{aligned} q(\mathbf{x}) = q(\mathbf{x_0}) + \left(\frac{1}{2} \xi_1 \mathbf{p_1}^T \mathbf{A} \mathbf{p_1} + \xi_1 (\mathbf{A}\mathbf{x_0} - \mathbf{b})^T \mathbf{p_1} \right) + \dots \\ + \left(\frac{1}{2} \xi_k \mathbf{p_k}^T \mathbf{A} \mathbf{p_k} + \xi_1 (\mathbf{A}\mathbf{x_0} - \mathbf{b})^T \mathbf{p_k} \right) \end{aligned}$$

Výraz $\mathbf{A}\mathbf{x_0} - \mathbf{b}$ si můžeme vyjádřit jako první reziduum a obdobně jako v předchozím případě derivacemi získáváme hodnotu parametru

$$\xi_i = -\frac{\mathbf{g_0}^T \mathbf{p_i}}{\mathbf{p_i}^T \mathbf{A} \mathbf{p_i}}, \quad i = 1, \dots, k \tag{2.24}$$

Pokud bychom měli **A**-ortogonální bázi celého prostoru $\mathbb{R}^n$, dokázali bychom velice levně a přesně spočítat řešení rovnice. Samozřejmě takovouto bází v praxi nedisponujeme a postupně ji iteračně sestavujeme.

2.4.2 Krylovovy prostory

Nyní se budeme zabývat problémem tvoření **A**-ortogonální báze. Začneme přirozeně *Grand-Schmidtovým ortogonalizačním procesem.* Mějme bázi $\mathbf{p_1}, \dots, \mathbf{p_k}$ prostoru $\mathcal{K}^k \subset \mathbf{R}^n$ a vektor $\mathbf{h_k} \notin \mathcal{K}^k$ a budeme hledat další vektor $\mathbf{p_{k+1}}$ **A**-orotogonální báze prostoru $\mathcal{K}^{k+1} = \text{Span}\{\mathbf{p_1}, \dots, \mathbf{p_k}, \mathbf{h_k}\}$ ve tvaru

$$\mathbf{p_{k+1}} = \mathbf{h_k} + \beta_{k1} \mathbf{p_1} + \cdots + \beta_{kk} \mathbf{p_k} \tag{2.25}$$

z požadavku na **A**-otogonalitu vychází

$$0 = \mathbf{p_{k+1}}^T \mathbf{A} \mathbf{p_i} = \mathbf{h_k}^T \mathbf{A} \mathbf{p_i} + \beta_{ik} \mathbf{p_i}^T \mathbf{A} \mathbf{p_i}, \quad i = 1, \ldots, k \tag{2.26}$$

po úpravě

$$\beta_{ik} = -\frac{\mathbf{h_k}^T \mathbf{A} \mathbf{p_i}}{\mathbf{p_i}^T \mathbf{A} \mathbf{p_i}}, \quad i = 1, \ldots, k \tag{2.27}$$

Se stoupajícím indexem k stoupá i výpočetní náročnost ortogonalizace a algoritmus by se takto sta neúnosně náročným. Ukazuje se ale, že se tato procedura dá velice zefektivnit díky vlastnostem **Krylovových prostorů**

$$\mathcal{K}^k = \mathcal{K}^k(\mathbf{A}, \mathbf{g_0}) = \mathrm{Span}\{\mathbf{g_0}, \mathbf{A}\mathbf{g_0}, \ldots, \mathbf{A}^{k-1}\mathbf{g_0}\}, \quad 1, \ldots, n \tag{2.28}$$

a prostor $\mathcal{K}^0 = \{\mathbf{o}\}$.

Začněme s posloupností vektorů $\mathbf{p_1}, \ldots, \mathbf{p_i}$ tvořící **A**-orotgonální bázi prostoru $\mathcal{K}^i$, $i = 1, \ldots, k$. A mějme vektor $\mathbf{x_k}$ minimalizující funkci q na $\mathbf{x_0} + \mathcal{K}^k$. Potom musí být gradient $\mathbf{g_k}$ kolmý k prostoru $\mathcal{K}^k$

$$\mathbf{g_k}^T \mathbf{x} = 0, \quad \forall x \in \mathcal{K}^k \tag{2.29}$$

mimo jiné to znamená, že pokud $\mathbf{g_k} \neq \mathbf{o}$ pak $\mathbf{g_k} \notin \mathcal{K}^k$ (pokud $\mathbf{g_k} = \mathbf{o}$, našli jsme přesné řešení). Jelikož $\mathbf{g_k} \in \mathcal{K}^{k+1}$ využijeme postup z rovnice 2.25 s tím, že $\mathbf{g_k} = \mathbf{h_k}$. Dále si všimneme , že

$$\mathbf{A}\mathbf{x} \in \mathcal{K}^k, \quad \forall x \in \mathcal{K}^{k-1} \tag{2.30}$$

díky čemuž, společně s 2.29 získáváme

$$(\mathbf{A}\mathbf{p_i})^T \mathbf{g_k} = \mathbf{p_i}^T \mathbf{A} \mathbf{g_k} = \mathbf{g_k}^T \mathbf{A} \mathbf{p_i} 0, \quad i = 1, \ldots, k-1 \tag{2.31}$$

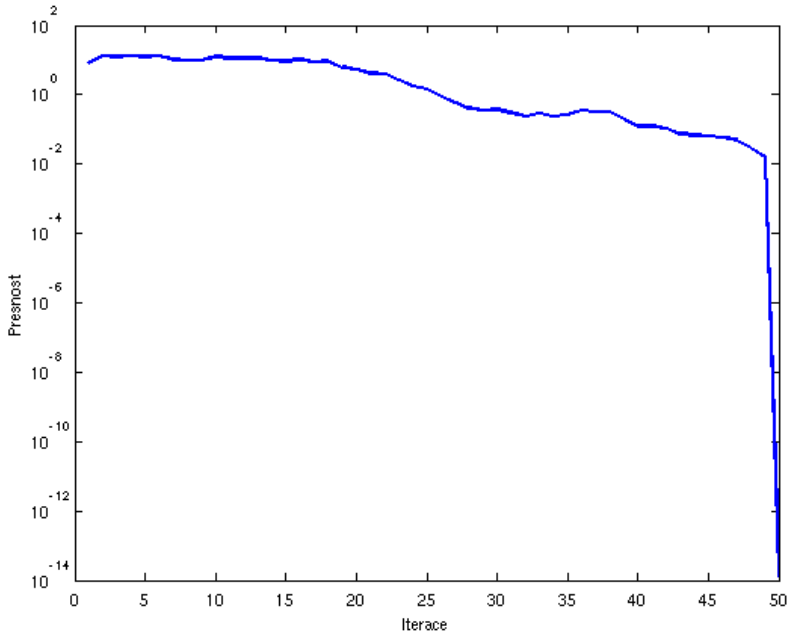
Z čehož vyplývá

$$\beta_{ik} = 0, \quad i = 1, \ldots, k-1 \tag{2.32}$$

a nový směr získáme ve tvaru

$$\mathbf{p_{k+1}} = \mathbf{g_k} + \beta_k \mathbf{p_k} \tag{2.33}$$

$$\beta_k = -\frac{\mathbf{g_k}^T \mathbf{A} \mathbf{p_k}}{\mathbf{p_k}^T \mathbf{A} \mathbf{p_k}} \tag{2.34}$$



Obrázek 2.4: Vývoj chyby v metodě sdružených gradientů - 50 iterací

2.4.3 Modifikace metody sdružených gradientů

Metoda sdružených gradientů je v současné době prakticky nejpoužívanější metodou minimalizace. Není tedy překvapivé, že vzniká veliké množství různých modifikací této metody, které ve speciálních případech zlepšují její vlastnosti.

Například proces **A**-ortogonalizace je u špatně podmíněných matic poměrně náchylný na numerické chyby. Tomu lze předcházet například restartováním procesu tvorby prostoru $\mathcal{K}$. Vždy po daném počtu kroků se využije aktuální reziduum. Extrémem této modifikace je, pokud tento restart provedeme v každém kroku - dostáváme tak **metodu největšího spádu**.

Dalších vylepšení jde dosáhnout podrobnějším zkoumáním možností počítání velikostí parametrů β a ξ. Podrobněji na toto téma například v [8].

2.5 Předpodmínění a stabilizace soustavy

2.5.1 Předpodmínění

Rychlost konvergence, a tedy i použitelnost, algoritmů, které jsme si představili, závisí na podmíněnosti matice $\mathbf{A}$ v řešeném funkcionálu. Jak už napovídá samotný název, předpodmínění je tedy proces, pomocí kterého budeme toto číslo zmenšovat.

Připomeňme si znovu roli, kterou matice $\mathbf{A}$ hraje :

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.35}$$

Základní myšlenka předpodmínění , je nalézt matici $\mathbf{B}$ takovou, že

$$\mathbf{A} \sim \mathbf{B} \tag{2.36}$$

respektive, že

$$\mathbf{B}^{-1} \sim \mathbf{A}^{-1} \tag{2.37}$$

Rovnici 2.35 potom můžeme inverzí matice $\mathbf{B}$ přenásobit:

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \mathbf{B}^{-1}\mathbf{b} \tag{2.38}$$

Pro snadný náhled do celé myšlenky si stačí představit, že $\mathbf{B} = \mathbf{A}$, potom by rovnice po přenásobení vypadala:

$$\mathbf{x} = \mathbf{B}^{-1}\mathbf{b} \tag{2.39}$$

2.5.2 Základní druhy předpodmiňovačů

Existuje mnoho způsobů, jakým se efektivně volí podoba matice $\mathbf{B}$, respektive její inverze. K těm úplně nejjednodušším na této škále se řadí například volba

$$\mathbf{B} = \mathrm{diag}(\mathbf{A}) \tag{2.40}$$

jak sestavení matice, tak hledání její inverze je v tomto případě naprosto primitivní. Toto předpodmínění má ovšem praktický význam jen u úzké škály matic, hlavně pak u těch s dominantní hlavní diagonálou.

Dalším významným způsobem předpodmínění je aplikace **neúplného LU rozkladu**. V případě běžného **LU rozkladu** se matice **A** rozkládá na součin dvou trojúhelníkových matic **L** a **U**. Tento rozklad by byl pro velké matice příliš nákladný a výsledné matice by se navíc nepříjemně zaplnily. Jeho modifikace na ne rozklad neúplný spočívá v zavedení omezujících pravidel pro sestavování těchto matic. Podle těchto pravidel se toto předpodmínění štěpí na mnoho dalších typů s různými vlastnostmi nejlépe vyhovujícími specifickým problémům. Z nejznámějších to jsou **ILU**, **MILU** případně **ILUT**.

2.5.3 Fixování pivotů

Fixování pivotů je metoda (popsána v [9]) vedoucí k zlepšení podmíněnosti matice soustavy řešené při využití **metody konečných prvků**. Vychází ze znalosti použité diskretizační sítě a úvahy, že eliminací vhodně zvolených bodů je možné výhodně předpodmínit řešení.

V následujících odstavcích popíšeme způsob, jak této myšlenky využít při tvorbě pseudoinverze singulární matice **A** (vzniklé například absencí alespoň části hranice s dirchletovou okrajovou podmínkou).

Vyjádření pseudoinverze

Při vyjadřování pseudoinverze vyjdeme z blokového schématu matice **A** :

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_{RR}} & \mathbf{A_{RS}} \\ \mathbf{A_{SR}} & \mathbf{A_{SS}} \end{bmatrix} \tag{2.41}$$

kde matice $\mathbf{A_{RR}}$ je regulární (a lépe podmíněná než matice **A**) a matice $\mathbf{A_{SS}}$ je velmi malá. V dalším kroku si blokově rozepíšeme řešení rovnice $\mathbf{Ax} = \mathbf{b}$.

$$\begin{bmatrix} \mathbf{A_{RR}} & \mathbf{A_{RS}} \\ \mathbf{A_{SR}} & \mathbf{A_{SS}} \end{bmatrix} \begin{bmatrix} \mathbf{x_R} \\ \mathbf{x_S} \end{bmatrix} = \begin{bmatrix} \mathbf{b_R} \\ \mathbf{b_S} \end{bmatrix}$$

po roznásobení tedy

$$\mathbf{A_{RR}x_R} + \mathbf{A_{RS}x_S} = \mathbf{b_R} \tag{2.42}$$

$$\mathbf{A_{SR}x_R} + \mathbf{A_{SS}x_S} = \mathbf{b_S} \tag{2.43}$$

Z (2.42) si vyjádříme $\mathbf{x_R}$

$$\mathbf{x_R} = \mathbf{A_{RR}}^{-1}(\mathbf{b_R} - \mathbf{A_{RS}}\mathbf{x_S})$$

a dosadíme do (2.43)

$$(\mathbf{A_{SS}} - \mathbf{A_{SR}}\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}})\mathbf{x_S} = \mathbf{b_S} - \mathbf{A_{SR}}\mathbf{A_{RR}}^{-1}\mathbf{b_R}$$

Z těchto rovnic si již můžeme vyjádřit pseudoinverzi jako součin matic, reprezentujících jednotlivé akce v řešení

$$\mathbf{A}^+ = \begin{bmatrix} \mathbf{A_{RR}}^{-1} & -\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{O} & (\mathbf{A_{SS}} - \mathbf{A_{SR}}\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}})^\dagger \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ -\mathbf{A_{SR}}\mathbf{A_{RR}}^{-1} & \mathbf{I} \end{bmatrix} \tag{2.44}$$

Operaci $\mathbf{A_{RR}}^{-1}$ budeme provádět iteračně (a mimo jiné tak nebudeme muset počítat vždy "přesně"). Všimněme si, že ačkoliv se zde matice $\mathbf{A_{RR}}^{-1}$ objevuje poměrně často, tak vždy jen v akci s vektorem pravé strany $\mathbf{b_R}$ nebo s maticí $\mathbf{A_{RS}}$. Vyčíslení této akce je v celém rozkladu nejdražší, ale provést ho je třeba pouze jednou, viz. kap. 2.5.3.

Věta 2.5.1. $\mathbf{A}^+$ *z (2.44) je pseudoinverze k matici* $\mathbf{A}$

Důkaz. Pro matici $\mathbf{A}^+$ musí platit:

1. $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$
2. $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$
3. $(\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+$
4. $(\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}$

Pro přehlednost zavedeme substituci $\mathbf{S} = \mathbf{A_{SS}} - \mathbf{A_{SR}}\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}$. Pseudoinverzi potom můžeme zapsat

$$\mathbf{A}^+ = \begin{bmatrix} \mathbf{A_{RR}}^{-1} + \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}\mathbf{S}^\dagger\mathbf{A_{SR}}\mathbf{A_{RR}}^{-1} & -\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}\mathbf{S}^\dagger \\ -\mathbf{S}^\dagger\mathbf{A_{SR}}\mathbf{A_{RR}}^{-1} & \mathbf{S}^\dagger \end{bmatrix} \tag{2.45}$$

Nyní si postupně odvodíme jednotlivé vztahy:

1.

$$\mathbf{A}\mathbf{A}^{+}\mathbf{A} = \mathbf{A}\begin{bmatrix} \mathbf{I} & \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}(\mathbf{I}-\mathbf{S}^{\dagger}\mathbf{S}) \\ \mathbf{O} & \mathbf{S}^{\dagger}\mathbf{S} \end{bmatrix} = \begin{bmatrix} \mathbf{A_{RR}} & \mathbf{A_{RS}} - \mathbf{A_{RS}}\mathbf{S}^{\dagger}\mathbf{S} + \mathbf{A_{RS}}\mathbf{S}^{\dagger}\mathbf{S} \\ \mathbf{A_{SR}} & \mathbf{A_{SR}}\mathbf{A_{RR}}^{-1}\mathbf{A_{RS}} - \mathbf{S}\mathbf{S}^{\dagger}\mathbf{S} \end{bmatrix} =$$

$$= \begin{bmatrix} \mathbf{A_{RR}} & \mathbf{A_{RS}} \\ \mathbf{A_{SR}} & \mathbf{A_{SS}} \end{bmatrix} = \mathbf{A}$$

2.

$$\mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} =$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}(\mathbf{I}-\mathbf{S}^{\dagger}\mathbf{S}) \\ \mathbf{O} & \mathbf{S}^{\dagger}\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{A_{RR}}^{-1} + \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}\mathbf{S}^{\dagger}\mathbf{A_{SR}}\mathbf{A_{RR}}^{-1} - \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}\mathbf{S}^{\dagger} \\ -\mathbf{S}^{\dagger}\mathbf{A_{SR}}\mathbf{A_{RR}}^{-1} \end{bmatrix} \begin{matrix} \\ \mathbf{S}^{\dagger} \end{bmatrix} =$$

$$= \mathbf{A}^{+}$$

Body 3 a 4 vplývají přímo z toho, že matice $\mathbf{A}^{+}$ i $\mathbf{A}$ jsou symetrické. □

Efektivní aplikace pseudoinverze

V našich aplikacích, se bude pro pevnou matici $\mathbf{A}$ často měnit pravá strana, je proto třeba z aplikace pseudoinverze nalézt co možná největší množství operací, které si lze připravit bez závislosti na pravé straně.

1. *Část bez pravé strany*

   $\mathbf{T_{RS}} \leftarrow \mathbf{A_{RR}}^{-1}\mathbf{A_{RS}}$

   $\mathbf{S}^{\dagger} \leftarrow (\mathbf{A_{SS}} - \mathbf{A_{SR}}\mathbf{T_{RS}})^{\dagger}$

2. *Iterovaná část*

   $\mathbf{T_{RR}} \leftarrow \mathbf{A_{RR}}^{-1}\mathbf{b_R}$

   $\mathbf{x_S} \leftarrow \mathbf{S}^{\dagger}(\mathbf{b_s} - \mathbf{A_{SR}}\mathbf{T_{RR}})$

   $\mathbf{x_R} \leftarrow \mathbf{T_{RR}} - \mathbf{T_{RS}}\mathbf{x_S}$

Algoritmus 1. ***Jacobiho metoda***

```
function [x] = jacobi(A,b,e)
n = size(A,1);
D = diag(diag(A));
E = -tril(A,-1);
F = -triu(A,1);
M = inv(D) * (E + F);
N = inv(D);
x = zeros(n,1);
k = 0;
while norm(A*x - b) > e
 x = M*x + N*b;
 k = k + 1;
end
```

Algoritmus 2. ***Gauss-Seidelova metoda***

```
function [x] = gaussSeidel(A,b,e)
n = size(A,1);
D = diag(diag(A));
E = -tril(A,-1);
F = -triu(A,1);
N = inv(D - E);
M = N*F;
x = zeros(n,1);
k = 0;
while norm(A*x - b) > e
 x = M*x + N*b;
 k = k + 1;
end
```

Algoritmus 3. ***Metoda největšího spádu***

```
function [x] = nejvSpad(A,b,e)
n = size(A,1);
x = zeros(n,1);
r = b - A*x;
k = 0;
while norm(r) > e
 a = r'*r / (r'*A*r);
 x = x + a*r;
 r = b - A*x;
 k = k + 1;
end
```

Algoritmus 4. ***Metoda sdružených gradientů***

```
function [x] = congrad(A,b, e)
n = size(A,1);
x = zeros(n,1);
g = A*x - b;
p = g;
while norm(g) > e
 al = norm(g)^2 / (p' * A * p);
 x = x - al*p;
 gn = g - al * A * p;
 be = norm(gn)^2 / norm(g)^2;
 p = gn + be*p;
 g = gn;
end
```

Kapitola 3

Metoda rozšířených lagrangiánů

3.1 Lagrangián

V minulé kapitole jsme se zabývali minimalizací funkcionálu. Minimalizace probíhala na daném prostoru bez omezení, ale v praktických aplikacích se často setkáváme s omezením, které má řešení splňovat. Ať už jsou to metody fiktivních oblastí, kdy potřebujeme vynutit složitou hranici řešeného problému, nebo právě v této práci diskutované společné hranice oblastí vzniklých diskretizací větší oblasti.

Velmi známou metodou, pro úpravu funkcionálu takovým způsobem, aby bylo stále možné jej minimalizovat na celém prostoru a přesto získat řešení splňující zadané podmínky je využití lagrangových multiplikátorů.

Pro naprostou jednoduchost předveďme základní myšlenku na minimalizaci funkcionálu

$$q(\mathbf{v}) = \frac{1}{2}(\mathbf{A}\mathbf{v}, \mathbf{v}) - (\mathbf{b}, \mathbf{v}) \tag{3.1}$$

kde $\mathbf{A} \in \mathbb{R}^{n\times n}$ je symetrická a pozitivně definitní matice, $\mathbf{b} \in \mathbb{R}^n$ a $(.,.)$ označuje Eukli-dovský skalární součin. Minimum funkcionálu q budeme hledat na množině

$$V = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{B}\mathbf{x} = \mathbf{c}\} \tag{3.2}$$

kde $\mathbf{B} \in \mathbb{R}^{m\times n}$ a $\mathbf{c} \in \mathbb{R}^m$. Množinu si lze zapsat také ve formě

$$V = \bar{\mathbf{x}} + \ker \mathbf{B}, \quad \mathbf{B}\bar{\mathbf{x}} = \mathbf{c} \tag{3.3}$$

Problém, kterým se budeme zabývat je tedy nalezení $\bar{\mathbf{v}}$:

$$\bar{\mathbf{v}} = \arg\min_{\mathbf{x}\in V}(q(\mathbf{x})) \tag{3.4}$$

Nyní zavedeme **lagrangův multiplikátor** $\lambda \in \mathbb{R}^m$ a díky němu přeformulujeme úlohu na minimalizaci bez omezení

$$\bar{\mathbf{v}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n}(q(\mathbf{x}) + (\lambda, \mathbf{B}\mathbf{x} - \mathbf{c})) \tag{3.5}$$

V této formulaci je λ nová neznámá, jejíž hodnotu získáme například řešením sedlobodové úlohy pro lagrangián

$$\mathscr{L}(\mathbf{x}, \lambda) = q(\mathbf{x}) + (\lambda, \mathbf{B}\mathbf{x} - \mathbf{c}) \tag{3.6}$$

přičemž sedlovým bodem nazýváme bod $\{\bar{\mathbf{x}}, \bar{\lambda}\}$, pokud

$$\mathscr{L}(\bar{\mathbf{x}}, \lambda) \leq \mathscr{L}(\bar{\mathbf{x}}, \bar{\lambda}) \leq \mathscr{L}(\mathbf{x}, \bar{\lambda}), \quad \forall \mathbf{x} \in \mathbb{R}^n, \forall \lambda \in \mathbb{R}^m \tag{3.7}$$

Z 3.7 také vyplývá klasický výsledek (známý jako *dualita*)

$$\min_{\mathbf{x}\in\mathbb{R}^n} \max_{\lambda\in\mathbb{R}^m} \mathscr{L}(\mathbf{x}, \lambda) = \max_{\lambda\in\mathbb{R}^m} \min_{\mathbf{x}\in\mathbb{R}^n} \mathscr{L}(\mathbf{x}, \lambda) = \mathscr{L}(\bar{\mathbf{x}}, \bar{\lambda}) \tag{3.8}$$

Klasicky formulovanou funkci $\mathscr{L}$ můžeme dále rozšířit o penalizační prvek, penalizující body vzdálené od přípustné množiny řešení

$$\mathscr{L}_r(\mathbf{x}, \lambda) = q(\mathbf{x}) + (\lambda, \mathbf{Bx} - \mathbf{c}) + \frac{r}{2}\|\mathbf{Bx} - \mathbf{c}\|^2 = \mathscr{L}(\mathbf{x}, \lambda) + \frac{r}{2}\|\mathbf{Bx} - \mathbf{c}\|^2 \tag{3.9}$$

Lze velice snadno ukázat, že jakýkoliv sedlový bod $\mathscr{L}_r$ je současně i sedlovým bodem $\mathscr{L}$ a obráceně (stačí si všimnout, že výraz $\frac{r}{2}\|\mathbf{Bx} - \mathbf{c}\|^2$ zmizí, pokud je splněna podmínka přípustnosti řešení $\mathbf{Bx} = \mathbf{c}$). Penalizační prvek výrazně zlepšuje konvergenci metod, které níže popíšeme.

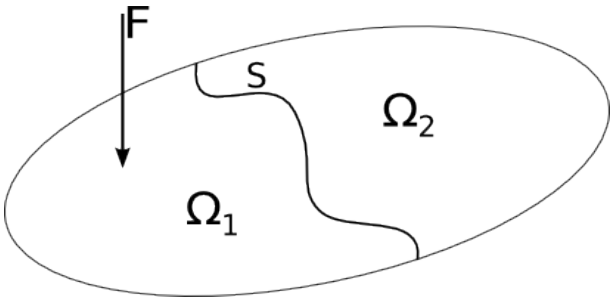
3.1.1 Minimalizace $\mathscr{L}_r$

Úplně základní algoritmus pro hledání sedlového bodu postupuje iteračně. Vždy najde pro dané λ minimalizující $\mathbf{x}$ (například metodou sdružených gradientů) a pro toto $\mathbf{x}$ potom maximalizující λ (většinou prostých iterací). V jednotlivých krocích tedy:

1. Zvolíme $\lambda_0 \in \mathbb{R}^m, \quad k = 0$
2. $\mathbf{x}_k = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathscr{L}_r(\mathbf{x}, \lambda_k)$ (vnitřní úloha)
3. $\lambda_{k+1} = \lambda_k + \rho_k(\mathbf{Bx} - \mathbf{c})$
4. $k = k + 1$

Na kostře tohoto postupu je pak založena celá řada algoritmů. Jedním z nejjednodušších je **Uzawův algoritmus** (alg. 5). Nevyužívá žádnou penaltu a vnitřní úlohy počítá s konstantní přesností. Právě nulová penalizace ovšem dává tomuto algoritmu vynikající paralerizační vlastnosti, vykoupeny jsou ale extrémně pomalou konvergencí.

O poznání komplikovanější je algoritmus **Smale** (alg. 6). Ten již plně využívá penalizačního parametru a podrobně řídí přesnost řešení vnitřních úloh. Nejprve začíná s poměrně velikou nepřesností, která v počátečních fázích není vůbec na škodu celkové konvergenci, a teprve s přibližováním úlohy předepsaným podmínkám zpřesňuje vnitřní úlohu.



Obrázek 3.1: Rozdělení oblastí a jejich společná hranice

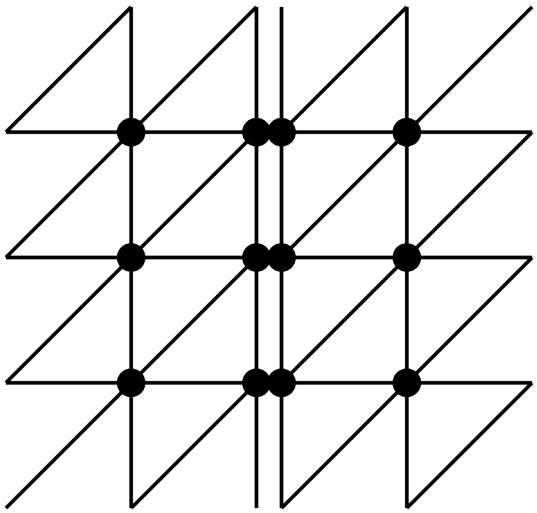
3.2 Klasické svázání funkcí

První představený způsob využití lagrangiánu pro diskretizaci oblastí bude přímo využívat představené algoritmy a bude vycházet z velice intuitivní a čitelné představy. Uzlové body, kterými vede hranice diskretizačních oblastí, zdvojíme (viz obr. 3.2), a s oblastmi od této chvíle budeme zacházet jako s nesouvisejícími. Matice soustavy $\mathbf{A}$ tedy získá podobu

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & & \\ & \ddots & \\ & & \mathbf{A_k} \end{bmatrix} \tag{3.10}$$

kde $\mathbf{A_1}, \ldots, \mathbf{A_k}$ jsou matice tuhosti jednotlivých podoblastí.

K zajištění podmínky, že hodnoty řešení v uzlech vzniklých zdvojením budou stejné jako v původních uzlech využijeme omezující podmínky, které budou vynucovaný lagrangovými multiplikátory. Matici $\mathbf{B}$, která tyto podmínky vynutí, složíme tak, že vždy jeden řádek bude zajišťovat rovnost hodnot v jedné dvojici uzlů. Například si vezměme k-tou dvojici



Obrázek 3.2: Duplikace bodů na společné hranici

uzlů s indexy i a j. k-tý řádek matice $\mathbf{B}$ bude vypadat

$$\mathbf{B} = \begin{bmatrix} & i & & j & \\ \ldots & 1 & \ldots & -1 & \ldots \end{bmatrix} \tag{3.11}$$

a příslušný řádek vektoru $\mathbf{c}$ bude 0. Je zřejmé, že po přenásobení vektorem řešení $\mathbf{x} = [x_1, \ldots, x_n]$ nám zůstane výraz

$$x_i = x_j \tag{3.12}$$

Budeme-li svazovat k dvojic uzlů, kterých z celkového počtu n uzlů, bude $\mathbf{B} \in \mathbb{R}^{k \times n}$ a $\mathbf{c}$ nulový vektor velikosti k.

3.3 Metoda projekce na společnou hranici

Zcela odlišný přístup k využití rozšířených lagrangiánů při řešení na podoblastech, prezentovaný například v [5], využívá navíc kromě lagrangova multiplikátoru další neznámou a to funkci q, jež udává požadovanou hodnotu na hranici. Zavedení této pomocné funkce usnadňuje paralelizaci výpočtů na jednotlivých oblastech.

V [5] je celý algoritmus popsán pouze v obecném, spojitém tvaru. I zde tedy uvedeme nejprve tento teoretický základ a poté odvodíme algoritmus v diskrétním tvaru.

Zabývat se budeme standardním variačním problémem, nad jednotlivými oblastmi i. Tedy hledáním takového $u_i \in V_i$, aby:

$$a_i(u_i, v) = L(v), \quad \forall v \in V_i \tag{3.13}$$

kde V_i je Sobolův prostor funkcí nad oblastí Ω_i.

Vzhledem k nové neznámé g budeme pracovat i s modifikovaným lagrangiánem, nyní již vyjádřeným jakou součet přes jednotlivé oblasti (pro jednoduchost prezentováno pro dvě) :

$$\mathscr{L}_r(v, q, \lambda_l) = \sum_{i=l}^{2} \left\{ \frac{1}{2} a_i(u_i, v) - L_i(v) + \frac{1}{2}\|v_i - q\|^2 + (\lambda_i, v_i - q) \right\} \tag{3.14}$$

přičemž označíme-li si společnou hranici těchto oblastí (Ω_1, Ω_2) S, označíme si ještě prostory

$$W = \mathrm{Tr}V_1|S = \mathrm{Tr}V_2|S \tag{3.15}$$

$$H = W^2 \tag{3.16}$$

$$V = V_1 \times V_2 \tag{3.17}$$

Problém jsme tak transformovali na hledání bodu $(u, q, \lambda) \in V \times W \times H$ takového, aby

$$\begin{cases} (i) & \frac{\partial \mathscr{L}_r}{\partial v}(u, q, \lambda).w = 0, & \forall w \in V \\ (ii) & \langle \frac{\partial \mathscr{L}_r}{\partial q}(u, q, \lambda), dq) \rangle = 0, & \forall dq \in W \\ (iii) & \langle \frac{\partial \mathscr{L}_r}{\partial \lambda}(u, q, \lambda), d\lambda \rangle = 0, & \forall d\lambda \in H \end{cases} \tag{3.18}$$

3.3.1 Postup řešení

$$a_l(u_l^n, w_i) - L_i(w_i) + r\langle \mathscr{S}(u_l^n - q^{n-1}), w_i \rangle + \langle \mathscr{S}\lambda_l^n, w_i \rangle = 0, \quad \forall w_i \in V_l,\ i = 1, 2 \tag{3.19}$$

$$\lambda_i^{n+\frac{1}{2}} = \lambda_i^n + r(u_i^n - q^{n-1}), \qquad i = 1, 2 \tag{3.20}$$

$$-r\langle \mathscr{S}(u_1^n + u_2^n - 2q^n), \hat{q} \rangle - \langle \mathscr{S}(\lambda_1^{n+\frac{1}{2}} + \lambda_2^{n+\frac{1}{2}}), \hat{q} \rangle = 0, \quad \forall \hat{q} \tag{3.21}$$

$$\lambda_i^{n+1} = \lambda_i^{n+\frac{1}{2}} + r(u_i^n - q^n), \qquad i = 1, 2 \tag{3.22}$$

3.4 Diskrétní přístup

Nyní si podrobně rozebereme situaci a přístup v prostorech konečné dimenze. Pro jednoduchost budeme stále pracovat se dvěma oblastmi: Ω_1 s prostorem funkcí nad ním V_1 , Ω_2 s prostorem funkcí V_2 a jejich společnou hranicí S s prostorem funkcí W. Báze prostorů funkcí nad těmito oblastmi si můžeme označit takto:

$$\{e_1^1, \ldots, e_{n_1}^1\} \in V_1$$

$$\{e_1^2, \ldots, e_{n_2}^2\} \in V_2$$

$$\{s_1, \ldots, s_k\} \in W$$

Stojí za povšimnutí, že prostory mají dimenze n_1, n_2 a k, přičemž pro ně musí platit $n_1 \geq k$ a $n_2 \geq k$.

3.4.1 Společná hranice

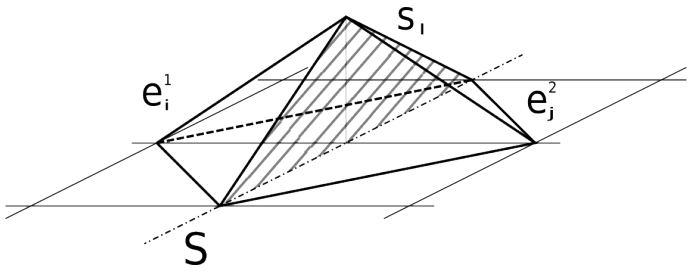
Definice Funkce $Tr_1 : V_1 \to W$ (resp. $Tr_2 : V_2 \to W$) je takové zobrazení, které funkci $u \in V_1$ (resp. $u \in V_2$) přiřadí takovou funkci $q \in W$, že pro všechny body hranice S budou mít stejnou funkční hodnotu ($\forall x \in S : u(x) = q(x)$).

Definice O diskretizačních sítích (respektive diskrétních bázích) dvou oblastí se společnou hranicí říkáme, že **souhlasné**, pokud existuje taková báze $\{s_1, \ldots, e_k\}$ funkčního prostoru nad hranicí S, že pro každý její člen s_l můžeme nalézt členy bází obou prostorů e_i^1 a e_i^2 takové, že $Tr_1 e_i^1 = s_l$ a $Tr_2 e_j^2 = s_l$. Viz. obr. 3.3.

V této práci předpokládáme, že diskretizace oblastí jsou souhlasné, vyhneme se tak komplikacím, které jdou nad rámec řešeného tématu. Velice se nám v bude hodit vyjádření funkce Tr_l jako matice. Takovouto matici snadno sestrojíme:

$$\{\mathbf{Tr_l}\}_{ij} = \begin{array}{ll} 1 & Tr_l e_j^l = s_i \\ 0 & Tr_l e_j^l \neq s_i \end{array} \qquad i = 1 \ldots k \quad j = 1 \ldots n_l \tag{3.23}$$

Získáváme tak 1 obdélníkovou matici $\mathbf{Tr_l} \in \mathbb{R}^{k \times n_l}$.



Obrázek 3.3: Vztah funkcí na diskrétní hranici

3.4.2 Skalární součin na hranici

V spojité verzi algoritmu 3.3.1 se setkáváme se skalárním součinem na společné hranici. Musíme proto navrhnout řešení této operace v diskrétní podobě a zvláště pak numerické vyjádření operátoru $\mathscr{S}$.

Skalární součin na hranici je bilineární zobrazení $W \times W \to \mathbb{R}$. Pokud se někdy jako argument objeví vektor jiného prostoru (z V_1 nebo V_2), je myšleno jeho zobrazení do prostoru W.

Definujme si nyní matice

$$\{\mathbf{M_W}\}_{ij} = \int_S s_i s_j dx \tag{3.24}$$

$$\{\mathbf{B_l}\}_{ij} = \int_S s_i Tr_l e_j^l dx \tag{3.25}$$

Jako nejjednodušší volba operátoru $\mathscr{S}$ je potom matice $\mathbf{M_w}^T\mathbf{M_w}$. Lze zvolit i podstatně rafinovanější operátory, jejich volba ovšem ovlivňuje pouze rychlost konvergence a na rozbor samotného algoritmu nemají žádný dopad. Skalární součin na hranici si tedy můžeme definovat

$$\mathbf{q_1}, \mathbf{q_2} \in W : \quad \langle \mathscr{S}\mathbf{q_1}, \mathbf{q_2}\rangle = \mathbf{q_1}^T\mathbf{M_W}^T\mathbf{M_W}\mathbf{q_2} \tag{3.26}$$

Je vidět, že $\mathbf{M_W}\mathbf{Tr_l}\mathbf{V_l} = \mathbf{B_l}\mathbf{V_l}$. Získáváme tak jednoduchý vztah pro funkce z oblastí:

$$\mathbf{u_l} \in V_l, \mathbf{u_k} \in V_k : \quad \langle \mathscr{S}\mathbf{u_l}, \mathbf{u_k}\rangle = \langle \mathscr{S}\mathbf{Tr_l}\mathbf{u_l}, \mathbf{Tr_k}\mathbf{u_k}\rangle = \mathbf{u_l}^T\mathbf{Tr_l}^T\mathbf{M_W}^T\mathbf{M_W}\mathbf{Tr_k}\mathbf{u_k} = \mathbf{u_l}^T\mathbf{B_l}^T\mathbf{B_k}\mathbf{u_k} \tag{3.27}$$

Pozorování Pro bázové vektory $\mathbf{e_i^l}, \mathbf{e_j^l}$ prostoru V_l platí:

$$\langle \mathscr{S}\mathbf{e_i^l}, \mathbf{e_j^l} \rangle = \{\mathbf{B_l}^T \mathbf{B_l}\}_{ij}$$

3.4.3 Diskrétní verze algoritmu z 3.3.1

Postupně si rozebereme všechny 4 body původního algoritmu a nalezneme jejich konkrétní podobu v diskrétním případě. Jednotlivé vektory budeme hledat ve tvarech:

$$\begin{aligned} \mathbf{u^l} &= \sum_{i=1}^{n^l} u_i^l \mathbf{e_l^i} \\ \mathbf{q} &= \sum_{i=1}^{k} q_i \mathbf{s_i} \\ \lambda^{\mathbf{l}} &= \sum_{i=1}^{k} \lambda_i^l \mathbf{s_i} \end{aligned}$$

Pro přehlednost jsou vynechány indexy jednotlivých kroků (v původním algoritmu index n).

Ad 1:

$$a(\mathbf{u^l}, \mathbf{e_j^l}) - L(\mathbf{e_j^l}) + r\langle \mathscr{S}(\mathbf{u^l} - \mathbf{q}), \mathbf{e_j^l} \rangle + \langle \mathscr{S}\lambda^{\mathbf{l}}, \mathbf{e_j^l} \rangle = 0 \qquad l = 1,2 \quad j = 1, \ldots, n^l$$

$$\sum_{i}^{n^l} u_i^l a(\mathbf{e_i^l}, \mathbf{e_j^l}) - L(e_j^l) + r \sum_{i=1}^{n^l} u_i^l \langle \mathscr{S}\mathbf{e_i^l}, \mathbf{e_j^l} \rangle - r \sum_{i=1}^{k} q_i \langle \mathscr{S}\mathbf{s_i}, \mathbf{e_j^l} \rangle + \sum_{i=1}^{k} \lambda_i^l \langle \mathscr{S}\mathbf{s_i}, \mathbf{e_j^l} \rangle = 0$$

$$\sum_{i=1}^{n^l} u_i^l [a(\mathbf{e_i^l}, \mathbf{e_j^l}) + r\langle \mathscr{S}\mathbf{e_i^l}, \mathbf{e_j^l} \rangle] = L(\mathbf{e_j^l}) + \sum_{i=1}^{k} [rq_i - \lambda_i^l] \langle \mathscr{S}\mathbf{s_i}, \mathbf{e_j^l} \rangle \qquad l = 1,2 \quad j = 1, \ldots, n^l$$

Což pokud spojíme s pozorováním 3.4.2 a definicí $\mathbf{A}$ a $\mathbf{b}$, získáváme vztah

$$(\mathbf{A} + r\mathbf{B_l}^T \mathbf{B_l})\mathbf{u^l} = \mathbf{b} + \mathbf{M}^T \mathbf{B_l}(r\mathbf{q} - \lambda^{\mathbf{l}}) \qquad l = 1,2 \tag{3.28}$$

Ad 2:

$$\lambda^{\mathbf{l}} \leftarrow \lambda^{\mathbf{l}} + r(\mathbf{Tr_l u^l} - \mathbf{q}) \qquad l = 1,2 \tag{3.29}$$

Ad 3:

$$-r\langle \mathscr{S}\mathbf{Tr_1 u^1} + \mathbf{Tr_2 u^2} - 2\mathbf{q}, \mathbf{s_j}\rangle - \langle \mathscr{S}(\lambda^{\mathbf{1}} + \lambda^{\mathbf{2}}), \mathbf{s_j}\rangle = 0 \qquad j = 1, \ldots, k$$

$$2r\sum_{i=1}^{k} q_i \langle \mathscr{S}\mathbf{s_i}, \mathbf{s_j} = r\sum_{i=1}^{n^1} u_i^1 \langle \mathscr{S}\mathbf{e_i^1}, \mathbf{s_j}\rangle + r\sum_{i=1}^{n^2} u_i^2 \langle \mathscr{S}\mathbf{e_i^2}, \mathbf{s_j}\rangle + \sum_{i=1}^{k} (\lambda_i^1 + \lambda_i^2)\langle \mathscr{S}\mathbf{s_i}, \mathbf{s_j}\rangle \qquad j = 1, \ldots, k$$

A opět podle definice skalárního součinu dostáváme:

$$2r\mathbf{M}^T\mathbf{M}\mathbf{q} = r\mathbf{u_1}^T\mathbf{B_1}^T\mathbf{M} + r\mathbf{u_2}^T\mathbf{B_2}^T\mathbf{M} + (\lambda^{\mathbf{1}} + \lambda^{\mathbf{2}})^T\mathbf{M}^T\mathbf{M}$$

$$\mathbf{M}^T\mathbf{M}\mathbf{q} = \frac{1}{2}\mathbf{M}^T\mathbf{M}\mathbf{Tr_1 u_1} + \mathbf{M}^T\mathbf{M}\mathbf{Tr_2 u_2}\mathbf{M}^T\mathbf{M}(\lambda^{\mathbf{1}} + \lambda^{\mathbf{2}})^T$$

$$q = \frac{1}{2}\mathbf{Tr_1 u_1} + \frac{1}{2}\mathbf{Tr_2 u_2} + (\lambda^{\mathbf{1}} + \lambda^{\mathbf{2}})^T \tag{3.30}$$

Ad 4:

$$\lambda^{\mathbf{l}} \leftarrow \lambda^{\mathbf{l}} + r(\mathbf{Tr_l u^l} - \mathbf{q}) \qquad l = 1, 2 \tag{3.31}$$

Od prvního pohledu je jasné, že algoritmus založený na tomto základě bude výborně paralelizovatelný. Kroky 1,2,4 mohou bez jakýchkoliv obtíží probíhat pro každou oblast na nezávislých strojích a pouze krok 3 je třeba provést společně.

3.4.4 Vliv velkého parametru r

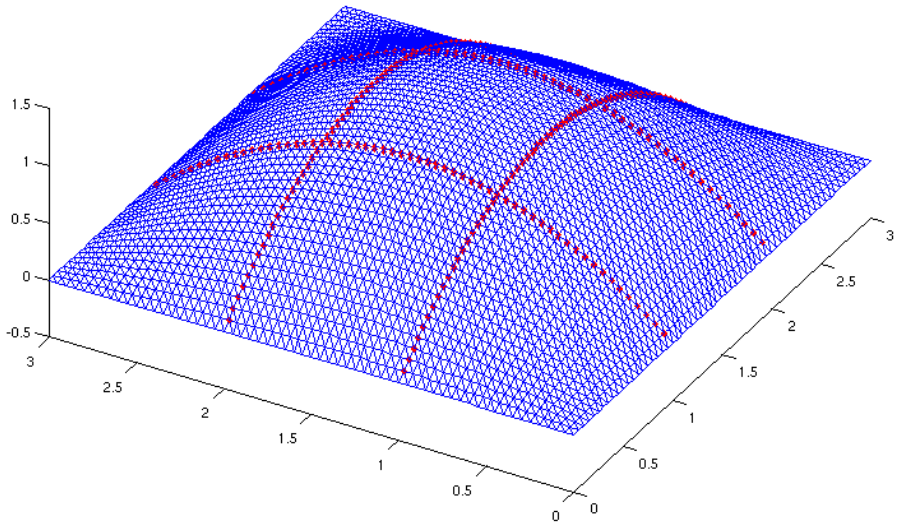
Při podrobném zkoumání rovnice 3.28 je dobře patrné, že čím bude parametr r větší, tím lépe se řešení přimkne k předepsané hranici $\mathbf{q}$. Na první pohled je to chování, které by mohlo zlepšit celkové vlastnosti algoritmu. Na toto téma jsem tedy provedl několik pokusů a chtěl jsem zaznamenat vliv rostoucího parametru r na počet vnějších iterací. Ale už při poměrně nízkém zvýšení začala úloha divergovat.

Důvod divergence vidím ve způsobu aktualizace Lagrangiánů λ popsaném v krocích 3.29 a 3.31. Díky vysokému r je odchylka okrajů oblastí od předepsané hranice velice malá a snadno se do ní promítne numerická chyba, která je poté mnohonásobně zvětšena. Minimalizuje se také vliv lagrangových multiplikátorů v řešení vnitřních úloh a tak přicházíme o důležitou informaci nutnou pro rychlou relaxaci předepsané hranice k optimální hodnotě.

Jako řešení se v tomto směru nabízí správná volba matice $\mathbf{M}$, respektive operátoru $\mathscr{S}$, který by zohledňoval chování úlohy v širším kontextu a umožnil tak velice rychlou adaptaci společné hranice.

3.5 Numerické experimenty

Všechny tři zkoumané algoritmy byly postaveny před řešení stejné úlohy - prohnutí dvojrozměrné membrány, rozložené na devět oblastí (obr.3.4). V následujících tabulkách jsou počty iterací pro různě husté sítě a vývoj podmíněnosti matice řešených ve vnitřních úlohách.



Obrázek 3.4: Řešení průhybu membrány - 9 oblastí

Náhled na chování počtu vnitřních iterací v závislosti na aktuální přesnosti vnější úlohy umožňují grafy 3.6, 3.5, 3.7. Červenou barvou je zobrazena přesnost vnější úlohy, modrou

počet iterací vnitřních úloh.

		Uzawa			
d.o.f. v Ω	**přesnost**	**vnější**	**vnitřní**	**cond min**	**cond max**
352	$1e-5$	30	85	114	114
$5e3$	$1e-5$	103	327	1653	1653
$3e4$	$1e-5$	366	1157	10110	10110

Tabulka 3.1: Uzawa - 9 oblastí

		Smale			
d.o.f. v Ω	**přesnost**	**vnější**	**vnitřní**	**cond min**	**cond max**
352	$1e-5$	10	54	110.4	110.9
$5e3$	$1e-5$	6	148	1591.3	1592.3
$3e4$	$1e-5$	8	351	9853.4	9855

Tabulka 3.2: Smale - 9 oblastí

		Alg3			
d.o.f. v Ω	**přesnost**	**vnější**	**vnitřní**	**cond min**	**cond max**
352	$1e-5$	46	1204	2198	156
$5e3$	$1e-5$	14	353		
$3e4$	$1e-5$	250	14493	10564	16570

Tabulka 3.3: Algoritmus projekce na hranici - 9 oblastí

3.6 Srovnání jednotlivých metod

Stojí za to porovnat tři odlišné metody spojení hranice. Z klasického přístupu je to metoda s lagrangiánem bez penalizace (alg. 5), tato metoda s penalizaci (rozšířený lagrangián - alg.

6) a naposledy diskutovaná metoda úplně oddělených funkčních prostorů s projekcí na společnou hranici (viz. 3.4.3).

Jak je vidět na grafech a výsledcích pokusů níže, naprosto bezkonkurenčně nejlepších výsledků bylo dosaženo s pomocí algoritmu *Smale*. Není to nijak překvapivý výsledek. Je to algoritmus využívající nejpodrobnějšího studia problému s velice propracovaným řízením přesnosti vnitřních úloh. Jeho velkou slabinou je ovšem velice omezená možnost paralerizace. Jeho výkon stojí na využití penalizačního prvku, který jinak diskrétní oblasti pevně spojuje a znemožňuje oddělené řešení.

Uzawův algoritmus je v zásadě algoritmus *Smale* ochuzený právě o penalizační prvek. U něj tak nestojí paralelizaci nic v cestě. Přináší to ovšem nové problémy. Konvergence vnější smyčky tím velice trpí, protože jednotlivé podmínky jsou již vynuceny pouze lagrangovými multiplikátory a ve vnitřních smyčkách se dokonce setkáváme se singulárními maticemi (v případě, že oblast sousedí všemi částmi své hranice s jinými oblastmi).

Algoritmus prezentovaný jako poslední se pomocí předepsané hranic pokouší kombinovat výhody obou předcházejících. V prvé řade je plně paralerizovatený, jelikož oblasti informaci o hranicích nečerpají ze sousedů, ale právě z předepsané hranice. Díky silné vazbě na tyto hranice jsou tedy i matice vnitřních úloh poměrně dobře podmíněné. Tento algoritmus ale ještě není podrobně prozkoumán a je mnoho faktorů, kterými by lze vlastnosti algoritmu ovlivnit.

3.7 Další práce

Jak již bylo výše naznačeno, skutečný potenciál metody s předepsanou společnou hranicí půjde využít teprve poté, co se naučíme efektivně volit operátor skalárního součinu $\mathscr{S}$. Nejúčinější bude využití **Steklov-Poincarého** operátoru.

Nabízí se několik možností jak efektivně získat jeho diskrétní formu pro konkrétní úlohu. Bylo by možné provést částečnou eliminaci neznámých na hranici a využít tedy přibližný **Schurův komplement**. V tomto směru by bylo snad bylo možné dosáhnout slušných výsledků adaptivní změnou přesnosti tohoto komplementu, který by se s blízkostí k hranici

mohl zpřesňovat.

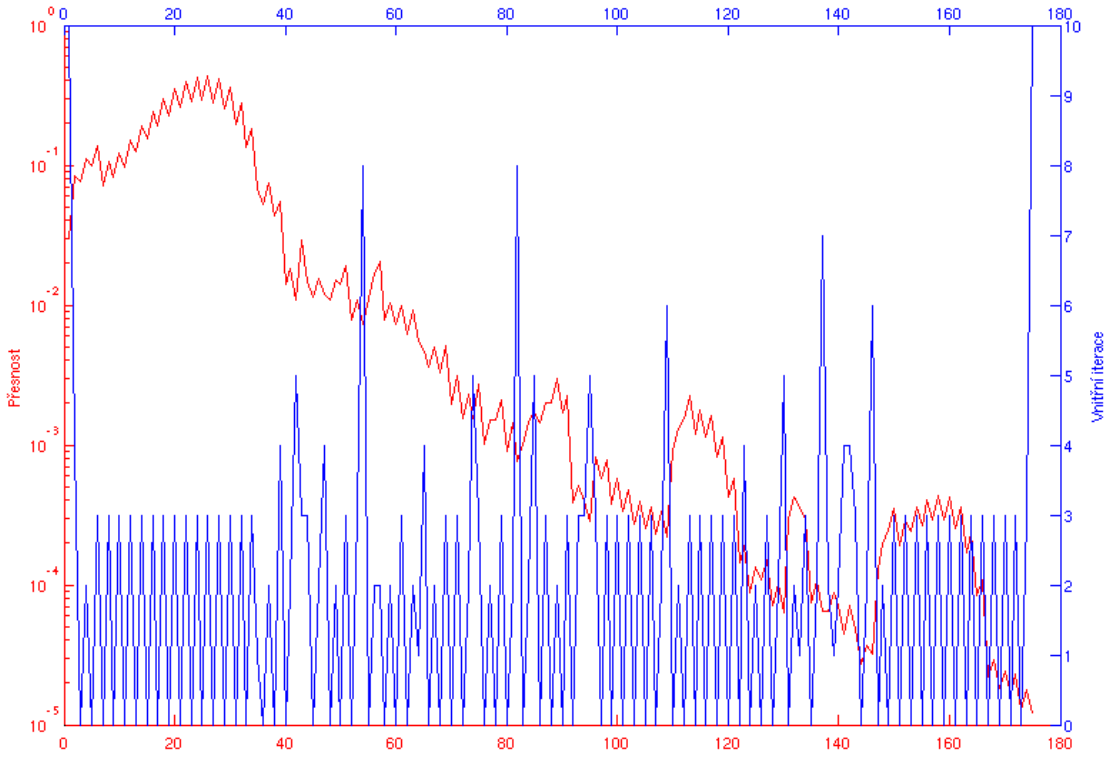
Zcela odlišným přístupem je potom využití **okrajových integrálních rovnic**, díky kterým je možné vyjádřit tento operátor velice efektivně a je možné, že právě tímto směrem vede cesta k skutečně velice silným výsledkům.

Algoritmus 5. ***Uzawa***

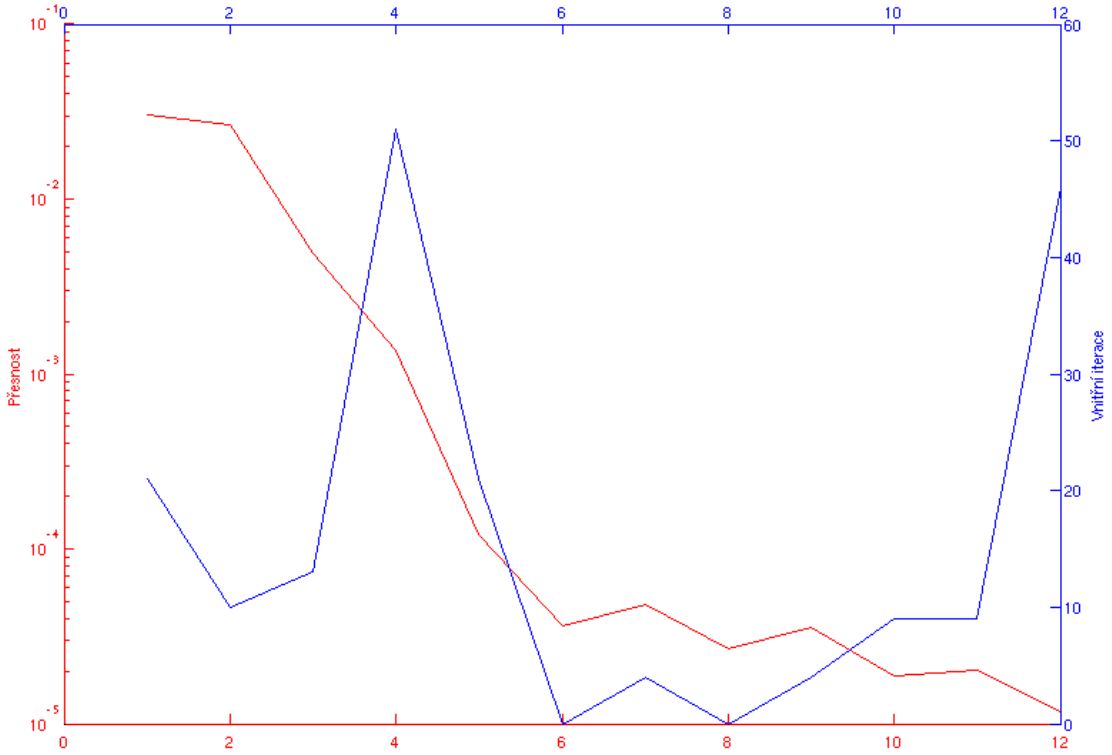
```
function [x,l=uzawa(A,b,B,c)
e = 1e-5;
n = size(A,1);
m = size(B,1);
k = 200; % pocet kroku
x = zeros(n,1);
l = zeros(m,1);
r = 0.5; % lagrangian step;
g = A*x - b + B'*l;
counter = 0;
while counter < k
 while norm(g) > e
  alfa = norm(g)^2/(g'*A*g);
  x = x - alfa*g;
  g = A*x - b + B'*l;
 end
 l = l + r*(B*x -c);
 g = A*x - b + B'*l;
 counter = counter + 1;
end
```

Algoritmus 6. ***Smale***

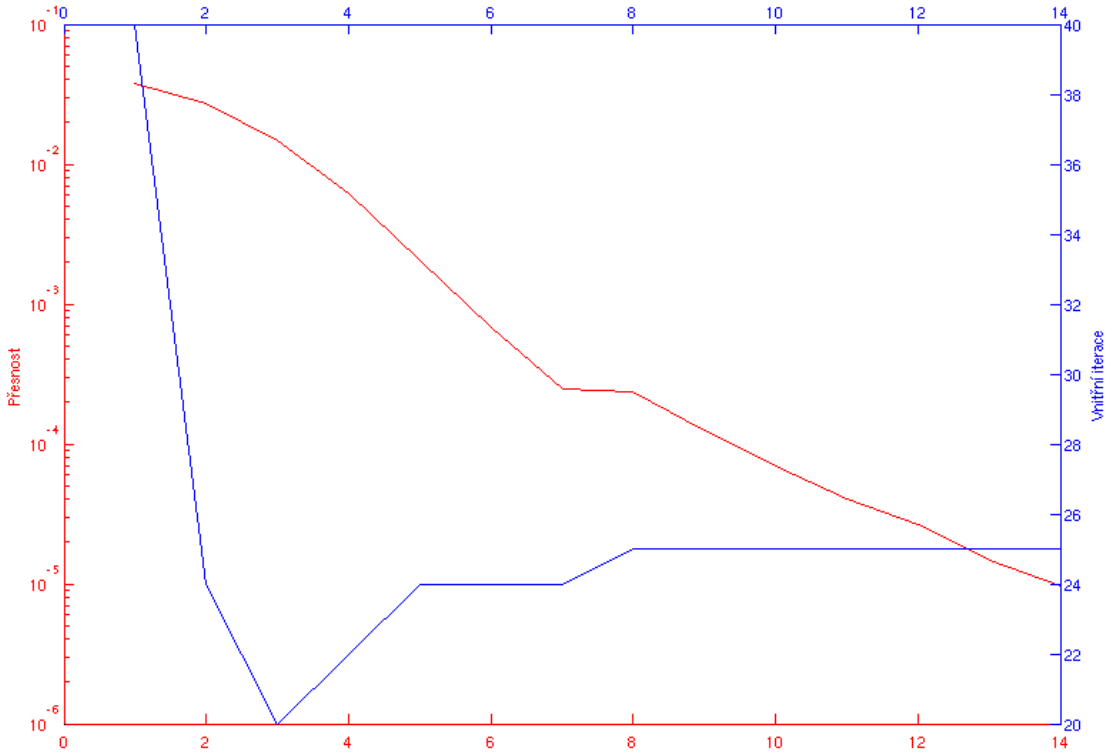
```
function [x,l] = smale(A,b,B,c,e,mi,ro,beta,M)
n =size(A,1);
m = size(B,1);
Ar = A + ro*B'*B;
br = b + ro*B'*c;
x = zeros(n,1);
l = zeros(m,1);
g = Ar*x - br + B'*l;
ng = norm(g);
k = 1;
while ng > e
 p = g;
 while norm(g) > min([M * norm(B*x - c), mi])
  alfa = norm(g)^2/(p'*Ar*p);
  x = x - alfa*p;
  gn = g - alfa * Ar * p;
  be = norm(gn)^2 / norm(g)^2;
  p = gn + be*p;
  g = gn;
 end
 l = l + ro*(B*x - c);
 lact = 0.5 * x' * A * x - b' * x + (B*x - c)'*l + %
  0.5*ro*norm(B*x - c)^2;
 if k > 1
  if lact < lprev + 0.5*ro*norm(B*x - c)^2
   ro=ro*beta;
   Ar = A + ro*B'*B;
   br = b + ro*B'*c;
  end
 end
 g = Ar*x - br + B'*l;
 ng = norm(g);
 lprev = lact;
 k = k + 1;
end
```



Obrázek 3.5: Uzawa



Obrázek 3.6: Smale



Obrázek 3.7: Projekce na hranici

Literatura

[1] *Zdeněk Dostál* **Lineární algebra**, VŠB-TUO, Ostrava 2004

[2] *Jiří Rohn* **Lineární algebra a optimalizace**, UK, Praha 2004

[3] *G. Golub, C. Van Loan* **Matrix Computations**, John Hopkins University Press, Baltimore 1996.

[4] *M. Menšík* **Bakalářská práce**, VŠB-TUO, Ostrava 2007

[5] *P. Le Tallec, T. Sassi* **Domain decomposition with nonmatching grids: Aumented Lagrangian approach**, Mathematics of Computation, vol. 64, no. 212, page 1367

[6] *R. Glowinski, P. Le Tallec* **Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics**, SIAM, Philadelphia 1989

[7] *Yousef Saad* **Iterative methods for Sparse Linear systems** Second Edition, 2000

[8] *Zdeněk Dostál* **Optimal Quadratic Programming Algorithms** Springer, New York 2009

[9] *T.Brzobohatý, Z.Dostál, T.Kozubek, A.Markopoulos* **Combining Cholesky decompostiion with SVD to stable evaluation of a generalized inverse of the stiffness matrix of a floating structure**, předloženo k publikaci

[10] *Z. Dostál, T. Kozubek, A. Markopoulos, M. Mensik* **Cholesky factorization of a positive semidefinite matrix with known kernel**, předloženo k publikaci