ČESKÉ
VYSOKÉ
UČENÍ
TECHNICKÉ
V PRAZE

Ústav technické matematiky

Soustavy lineárních algebraických rovnic v metodě konečných prvků

Systems of Linear Equations in the Finite Element Method

BAKALÁŘSKÁ PRÁCE

2019 rok

Karel Vacek

Studijní program: TZSI
Studijní obor: bez oboru
Vedoucí práce: doc. RNDr. Petr Sváček Ph.D.

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem: „Soustavy lineárních algebraických rovnic v metodě konečných prvků “ vypracoval samostatně pod vedením doc. RNDr. Petra Sváčka Ph.D. a s použitím literatury uvedené na konci mé bakalářské práce v seznamu použité literatury.

V Praze …….. .....

Jméno Příjmení

PODĚKOVÁNÍ

Na tomto místě bych velmi rád poděkoval svému vedoucímu mé bakalářské práce doc. RNDr. Petru Sváčkovi Ph.D. za odborný dohled, konzultace, vedení a čas, který mi věnoval, a které mi značně pomohly k vypracování této bakalářské práce.

ANOTAČNÍ LIST

Jméno autora:	Karel Vacek
Název BP:	Soustavy lineárních algebraických rovnic v metodě konečných prvků
Anglický název:	Systems of Linear Equations in the Finite Element Method
Rok:	2019
Studijní program:	TZSI
Obor studia:	bez oboru
Ústav:	Ústav Technické matematiky
Vedoucí BP:	doc. RNDr. Petr Sváček Ph.D.
Bibliografické údaje:	Počet stran 39
	Počet obrázků 8
	Počet tabulek 7
	Počet příloh 1
Klíčová slova:	soustavy lineárních rovnic, iterační metody, metoda sdružených gradientů, předpodmínění
Keywords:	systems of linear equations, iterative methods, conjugate gradient method, preconditioning
Anotace:	Bakalářská práce se zabývá užitím iteračních metod speciálně i pro soustavy vzniklé užitím metody konečných prvků. Obsahuje popis klasických iteračních ale i gradientních metod. Mimo to v práci byly tyto metody realizovány v Jazyce C. Pro tyto účely bylo užito formátu Compressed sparse row. Metody byly otestovány na několika případech.

Annotation: The bachelor thesis deals with the use of iterative methods especially for systems created using the finite element method. It contains description of classical iterative as well as gradient methods. In addition, these methods were implemented in C language. Compressed sparse row format was used for this purpose. Methods were tested in several cases.

OBSAH

Prohlášení 2
Poděkování 3
Anotační list 4
Obsah 6
1. ÚVOD 1
2. Matematické Základy 2
3. Metoda konečných prvků 5
4. Přímé metody řešení lineárních soustav 8
4.1. Gaussova eliminační metoda, pivotace 8
4.2. LU rozklad 9
4.3. QR rozklad 10
4.4. Choleského rozklad 10
5. Klasické iterační metody 12
5.1. Prostá iterační metoda 12
5.2. Jacobiova iterační metoda 13
5.3. Gaussova-Seidelova iterační metoda 14
5.4. SOR- (super relaxační metoda) 16
6. Moderní iterační metody 19
6.1. Metoda největšího spádu 20
6.2. Metoda sdružených gradientů 21
6.3. Možnosti výpočtu pro nesymetrické matice 24
7. Numerické výsledky 26
7.1. Uložení řídkých matic 26
7.2. Realizace Jacobiovy metody 27
7.3. Realizace Gaussovy-Seidelovy metody 28
7.4. Realizace Super relaxační metody 29
7.5. Realizace metody největšího spádu 30
7.6. Realizace metody sdružených gradientů 31
7.7. Použití a srovnání metod, jejich konvergence 32
8. Závěr 39
1 Seznam zkratek a symbolů 40
2 Seznam použité literatury 41
3 Seznam obrázků 42
4 Seznam tabulek 42
5 Seznam příloh 42

1. ÚVOD

Tato bakalářská práce se zabývá řešením n soustav lineárních algebraických rovnic v metodě konečných prvků, zejména se zaměřuje na praktické využití iteračních metod a jejich srovnání. Iterační metody jsou voleny tak, aby je bylo možné používat pro soustavy rovnic vzniklé diskretizací pomocí metod konečných prvků. Pomocí metody konečných prvků lze aproximovat parciální diferenciální rovnice, které popisují mnoho technických problémů a setkáváme se s nimi v podstatě ve všech inženýrských odvětvích. Touto aproximací dostaneme soustavy lineárních rovnic typu $\boldsymbol{Ax} = \boldsymbol{b}$ o n neznámých, kde $\boldsymbol{A}$ je matice typu $n \times n$, $\boldsymbol{x}$ hledaný vektor řešící soustavu a $\boldsymbol{b}$ je vektor pravých stran.

V této práci se budu primárně zabývat, jakými metodami lze danou rovnici řešit a jaké jsou mezi nimi rozdíly. Základní rozdělení těchto metod je na přímé a iterační. Přímé metody jsou metody, které vedou k přesnému řešení v konečném počtu kroků ale pro velké rozměry matic n se nepoužívají. Iterační metody se používají pro veké n, ale většinou nedosáhnou přesného řešení, což je zapříčiněno, že můžeme udělat pouze konečný počet iterací. Vzniká nám tedy posloupnost vektorů blížící se k správnému výsledku a řešení pouze aproximujeme. Iterační metody konvergují k správnému řešení pouze za předpokladu, že matice $\boldsymbol{A}$ splní podmínky pro použitou iterační metodu.

Cílem práce byla realizace vybraných iteračních metod v jazyce C. Pro uložení řídké matice $\boldsymbol{A}$ byl zvolen formát Compressed sparse row. Dále se práce zabývá testováním rychlosti konvergence různých iteračních metod a částečně vlivem předpodmínění. Jednotlivé chyby byly porovnány z hlediska velikosti chyby(rezidua) a počtu iterací.

Tato práce se skládá ze sedmi kapitol. První kapitola je úvod a motivace proč řešíme soustavu lineárních rovnic. V druhé kapitole nalezneme potřebné matematické pojmy k výkladu přímých a iteračních metod. Ve třetí kapitole najdeme, jak se dostaneme k matici, na které budeme metody porovnávat. Ve čtvrté kapitole jsou popsány přímé metody, a to Gaussova eliminace, LU rozklad a Choleského rozklad. V páté kapitole se budeme věnovat klasickým iteračním metodám a to Jacobiově, Gaussově-Seidelově a SOR (Successive Over Relaxation). V šesté si ukážeme moderní iterační metody největšího spádu, sdružených gradientů a metody které lze použít pro nesymetrické matice. V sedmé kapitole je popsána realizace a srovnány numerické výsledky jednotlivých metod. Poslední osmá kapitola obsahuje závěr s porovnáním metod.

2. Matematické Základy

V této kapitole jsou popsány základní matematické pojmy, které budeme dále používat. S řešením soustav lineárních rovnic jsou v první řadě spojeny otázky existence a jednoznačnosti řešení soustavy lineárních rovnic, k tomu budeme užívat pojmy lineární závislost a nezávislost vektoru.

Definice 1 (*Lineární závislost a nezávislost vektoru*) *Skupinu vektorů* $\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_n}$ *nazýváme lineárně závislou, jestliže existují reálná čísla* $\alpha_1, \alpha_2, \dots, \alpha_n$ *(z nichž alespoň jedno je různé od nuly) tak, že*

$$\alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u_2} + \cdots + \alpha_n \boldsymbol{u_n} = 0. \tag{1}$$

Skupinu vektorů, která není lineárně závislá, nazýváme lineárně nezávislá.

S pojmem lineární závislosti souvisí pojem hodnost matice.

Definice 2 *(Hodnost matice*) *Maximální počet lineárně nezávislých řádků matice* $\boldsymbol{A}$ *nazýváme hodností matice A. Značíme* $h(\boldsymbol{A})$.

Speciálně pro čtvercové matice užíváme pojem regulární matice.

Definice 3 (*Regulární matice*) *Čtvercovou matici typu* $n \times n$, *která má hodnost* n *nazýváme regulární maticí. Pokud matice není regulární, je singulární*.

Na základě předchozích pojmů je již možné uvést větu o existenci a jednoznačnosti řešení, to je Frobeniova veta.

Věta 1 (*Frobeniova věta*) *Soustava lineárních algebraických rovnic* $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ *má řešení právě tehdy, je-li*

$$h(\boldsymbol{A}) = h(\boldsymbol{A}|\boldsymbol{b}) .$$

Je-li $h(\boldsymbol{A}) = h(\boldsymbol{A}|\boldsymbol{b})$ *= n má soustava jediné řešení.*

Je-li $h(\boldsymbol{A}) = h(\boldsymbol{A}|\boldsymbol{b})$ *<n má soustava nekonečně mnoho řešení*.

Speciálně si uvedeme, že pro regulární matici $\boldsymbol{A}$ tato věta zaručuje existenci a jednoznačnost řešení pro soustavu $\boldsymbol{Ax} = \boldsymbol{b}$. Teď přejdeme k definicím, které nám pomohou v řešení iteračních metod. V první řadě jde o skalární součin dvou vektorů.

Definice 4 *(Skalární součin) Skalárním součinem vektorů* $\boldsymbol{u}= [u_1, u_2, \dots, u_n]$ *a* $\boldsymbol{v}= [v_1, v_2, \dots v_n]$ *nazýváme číslo* $(\boldsymbol{u}, \boldsymbol{v})$ *definované*

$$(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \cdot \boldsymbol{v} = u_1 \cdot v_1 + u_2 \cdot v_2 \dots + u_n \cdot v_n.$$

Dále definujeme pojem vlastní číslo matice, které nám určí spektrální poloměr matice a díky němuž zjistíme konvergenci klasických iteračních metod.

Definice 5 *(Vlastní číslo a vlastní vektor) Komplexní číslo* λ *nazýváme vlastním číslem čtvercové matice* $\boldsymbol{A}$, *existuje-li nenulový vektor* $\boldsymbol{x}$ *takový, že*

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}.$$

Takový vektor $\boldsymbol{x}$ *se nazývá vlastním vektorem matice* $\boldsymbol{A}$ *odpovídajícím vlastnímu číslu* λ.

Definice 6 *(Spektrální poloměr matice) Spektrum čtvercové matice* $\boldsymbol{A}$ *typu* $n \times n$ *nazýváme množinu* $\{\lambda_1, \lambda_{2,\dots}, \lambda_n\}$ *všech vlastních čísel matice* $\boldsymbol{A}$*. Potom číslo*

$$\rho_a = max|\lambda_i|,$$

nazýváme spektrálním poloměrem matice $\boldsymbol{A}$*.*

Rychlost konvergence budeme vyšetřovat užitím normy vektoru.

Definice 7 (*Norma vektoru*) *Normou vektoru rozumíme takové zobrazení* $||\cdot|| : \mathbb{R}^n \rightarrow <0, +\infty)$, *které pro libovolné vektory* $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ *a číslo* $\alpha \epsilon \mathbb{R}$ *splňuje:*

a) $\|\boldsymbol{u}\| = 0 \Leftrightarrow \boldsymbol{u} = 0$

b) $\|\alpha \boldsymbol{u}\| = |\alpha||\boldsymbol{u}|$

c) $\|\boldsymbol{u} + \boldsymbol{v}\| < |\boldsymbol{u}| + |v|$

Dále definujeme pojem normy matic a souvislost normy matic a vektoru.

Definice 8 (*Norma matice*) *Nechť každé komplexní matici* $\boldsymbol{A}$ *typu* $n \times n$ *je jednoznačně přiřazeno reálné číslo, přičemž platí:*

a) $\|\boldsymbol{A}\| \geq 0$; $\|\boldsymbol{A}\| = 0$ *právě tehdy když A je nulová matice*

b) $\|\alpha \boldsymbol{A}\| = |\alpha|\|\boldsymbol{A}\|$ *pro každé* $\alpha \in \mathbb{C}$

c) $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$ *pro každé dvě matice stejného typu*

d) $\|\boldsymbol{A} \cdot \boldsymbol{B}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{B}\|$ *pro každé dvě matice pro které je definován součin*

Pak číslo $\|\boldsymbol{A}\|$ *nazýváme normou matice* $\boldsymbol{A}$*.*

Nyní si můžeme ukázat základní typy norem. Základní typy norem lze nalézt např v [7]. Zde budeme užívat řádkovou normu $\big||\boldsymbol{x}|\big|_{\infty}$, sloupcovou normu $\big||\boldsymbol{x}|\big|_{\boldsymbol{1}}$ a eukleidovskou normu $\big||\boldsymbol{x}|\big|_{\boldsymbol{2}}$.

Konvergence iteračních metod je často spojena s vlastnostmi matice $\boldsymbol{A}$, například matice je symetrická, ostře diagonálně dominantní a pozitivně definitní. Tyto definice nám později pomůžou určit, kdy iterační metody konvergují.

Definice 9 *(Ostře diagonálně dominantní matice) Nechť* $\boldsymbol{A}$ *je reálná čtvercová matice typu* $n \times n$*. Matici* $\boldsymbol{A}$ *nazveme ostře diagonálně dominantní, platí-li*

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ij}| \text{ pro všechna i = 1,2,…,n}$$

nebo

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ji}| \text{ pro všechna i=1,2,…,n.}$$

Definice 10 (*Pozitivně definitní matice*) *Nechť* $\boldsymbol{A}$ *je čtvercová matice typu* $n \times n$*, pak řekneme, že je pozitivně definitní, jestli platí*

$$(\mathbf{Aw}, \mathbf{w}) = \boldsymbol{w}^T \boldsymbol{Aw} > 0 \; pro \; všechna \; nenulová \; \boldsymbol{w} \epsilon \mathbb{R}^n.$$

Definice 11 (*Symetrická matice*) *Nechť* $\boldsymbol{A}$ *je čtvercová matice typu* $n \times n$*. Pak řekneme, že je symetrická, jestliže platí*

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

pro všechna i, j. Tedy matice $\boldsymbol{A}$ *je symetrická, pokud platí*

$$\boldsymbol{A}^T = \boldsymbol{A},$$

kde $\boldsymbol{A}^T$ *je matice trasponovaná k matici* $\boldsymbol{A}$*.*

Při použití iteračních metod často minimalizujeme reziduum, tedy řešíme velikost vektoru $\boldsymbol{b} - \boldsymbol{Ax}^{(k)}$. Pro nevhodné matice (špatně podmíněné) je ale možné, že toto číslo bude malé, ale přiblížení $\boldsymbol{x}^{(k)}$bude od přesného řešení ještě velmi vzdáleno. Uvažujeme, že neřešíme rovnici $\boldsymbol{Ax} = \boldsymbol{b}$, ale rovnici

$$\boldsymbol{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}. \qquad (1)$$

Užitím $\boldsymbol{Ax} = \boldsymbol{b}$ a vztahem (1) určíme

$$\Delta\mathbf{x} = \mathbf{A}\Delta\mathbf{b}, \qquad \mathbf{x} = \mathbf{A}^{-1}\boldsymbol{b}\,.$$

Pomocí norem odhadneme pak

$$||\Delta\mathbf{x}|| \leq \left||\mathbf{A}^{-1}|\right| \cdot ||\Delta\mathbf{b}||\,, \qquad ||\mathbf{x}|| \geq \frac{||\mathbf{b}||}{||\mathbf{A}||}\,.$$

Užitím obou nerovnic pak dostaneme

$$\frac{||\Delta\mathbf{x}||}{||\mathbf{x}||} \leq ||\mathbf{A}|| \cdot \left||\boldsymbol{A}^{-1}|\right| \cdot \frac{||\Delta\mathbf{b}||}{||\mathbf{b}||}\,,$$

kde vidíme, že relativní chyba řešení $\frac{||\Delta\mathbf{x}||}{||\mathbf{x}||}$ lze odhadnout pomocí relativní chyby pravé strany $\frac{||\Delta\mathbf{b}||}{||\mathbf{b}||}$ vynásobené faktorem $||\mathbf{A}|| \cdot \left||\boldsymbol{A}^{-1}|\right|$. Číslo $\varkappa(\boldsymbol{A}) = ||\mathbf{A}|| \cdot \left||\boldsymbol{A}^{-1}|\right|$ nazýváme číslem podmíněnosti.

3. Metoda konečných prvků

V této kapitole ukážeme, jak se dostane speciální soustava rovnic, užitím metody konečných prvků v případě 1D problému, viz [1]. Tuto soustavu budeme později řešit různými iteračními metodami. Máme úlohu, kdy řešíme deformaci nosníku délky 1 uchyceného na obou stranách. Deformaci vyjádříme za předpokladu malých posunutí a lineární pružnosti pomocí, viz [2]

$$\begin{aligned} \sigma &= \mathrm{Eu}', \\ -\sigma' &= \mathrm{f}, \end{aligned} \tag{2}$$

kde E je elastický modul, u posunutí, σ napětí a f je intenzita sil. Rovnice (2) je doplněna okrajovými podmínkami $u(0) = u(1) = 0$. Složením rovnic (2) a uvažujeme-li $E = 1$, nám vznikne rovnice

$$-\mathrm{u}'' = f,$$

se stejnými okrajovými podmínkami. Tuto úlohu lze formulovat obecněji, v tzv. slabém smyslu. Hledáme funkci $u \epsilon C^2(I)$, kde $I =< 0{,}1 >$, takovou že platí

$$\begin{aligned} -\mathrm{u}''(x) &= f(x) \quad \forall x \in (0{,}1), \\ u(0) &= u(1) = 0. \end{aligned} \tag{3}$$

Jestliže vezmeme rovnici (3) a přenásobíme ji testovací funkcí $v \epsilon D$, kde prostor D je definován jako

$$D = \{\varphi \in C^{\infty}, \varphi(0) = \varphi(1) = 0\}$$

a zintegrujme přes interval I. Tím dostaneme

$$\int_0^1 -\mathrm{u}''(x)v(x)dx = \int_0^1 f(x)v(x)dx.$$

Užijeme per-partes na levou stranu rovnice, využijeme definice D a dostaneme

$$\int_0^1 \mathrm{u}'(x)v'(x)dx = \int_0^1 f(x)v(x)dx. \tag{4}$$

Rovnice (4) je řešení u problému (3) pro libovolné $v \epsilon D$. Zde přejdeme ke Galerkinově problému. Hledáme funkci $u \epsilon V$, tak aby platilo (4) pro všechna $v \epsilon V$. Zde prostor V je definován jako

$$V = \{\varphi \in C(I), \varphi' - \text{po částech spojitá}, \varphi(0) = \varphi(1) = 0\}.$$

Navážeme metodou konečných prvků, která aproximuje řešení. Volíme podprostor $V_h \subset V$ s konečnou dimenzí n. Předchozí problém budeme řešit v tomto podprostoru, tedy hledáme $u_h \in V_h$ splňující rovnici (4) pro $\forall v_h \in V_h$. Zvolíme bázi tohoto podprostoru $\varphi_1, \dots, \varphi_n$ a funkci $u_h(x)$ zapíšeme jako lineární kombinaci této báze

$$u_h(x) = \sum_{i=1}^{n} \alpha_i \varphi_i(x),$$

kde $\alpha \epsilon \mathbb{R}$. Dosazením do rovnice (4) a nahrazením $v_h = \varphi_1, \dots, \varphi_n$ nám vznikne

$$\int_0^1 \sum_{i=1}^{n} (\alpha_i \varphi'_i(x)) \varphi'_j(x) dx = \int_0^1 f(x) \varphi_i(x) dx\,.$$

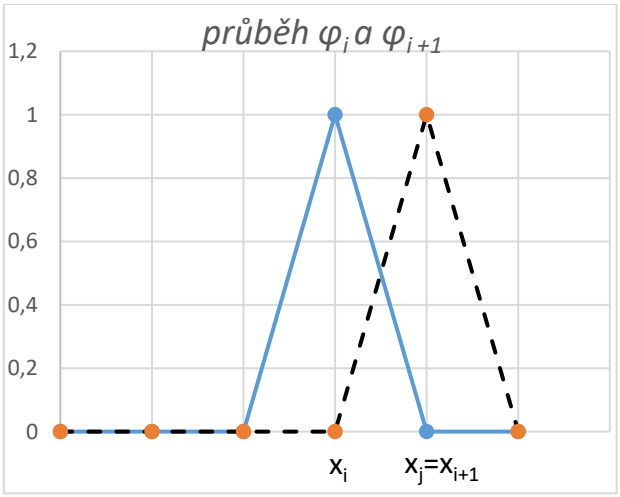
Toto lze ještě upravit na

$$\sum_{i=1}^{n} \alpha_i \int_0^1 \varphi'_i(x) \varphi'_j(x) dx = \int_0^1 f(x) \varphi_i(x) dx,$$

kde integrál na levé straně se značí k_{ij} a reprezentuje prvek v matici tuhosti a integrál na pravé straně se značí b_i a je to prvek vektoru zatížení. Lze tedy rovnici zapsat ve tvaru $\boldsymbol{K\alpha = b}$.

Je zřejmé, že volba báze je nejednoznačná. Metodu konečných prvků dostaneme, když volíme V_h jako prostor po částech polynomiálních funkcí. Uvažujme dělení intervalu $I =< 0{,}1 >$.

Obrázek 1 -Průběh $\varphi_i(x)$ a $\varphi_{i+1}(x)$



Interval I dělíme krokem $h > 0$, tedy $x_j = j \cdot h$. Prostor V_h volíme jako spojité po částech lineární funkce, tedy

$$V_h = \left\{ \varphi \in C(I): \varphi|_{<x_j, x_{j+1}>} \in P_1\left(< x_j, x_{j+1} >\right) \ \forall j \in 0,1, \dots, n-1, \varphi(0) = \varphi(1) = 0 \right\}.$$

Bázi prostoru V_h volíme tak aby $\varphi_i(x_j) = \delta_{ij}$ pro všechna $i, j \in \{1 \dots n-1\}$. Graf funkce φ_i je zobrazen na Obrázku 1 modrou barvou a na tomtéž obrázku je i φ_{i+1} černou přerušovanou barvou. Jelikož funkce $\varphi_i(x)$ je mimo interval $< x_{i-1}, x_{i+1} >$ nulová, tak pro prvky mimo hlavní a vedlejší diagonály matice $\boldsymbol{K}$ platí

$$k_{ij} = \int_0^1 \varphi'_j(x)\varphi'_i(x)dx = 0.$$

Pro prvky na diagonále dostaneme

$$k_{ii} = \int_0^1 \varphi'_i(x)\varphi'_i(x)dx \ = \int_{x_{i-1}}^{x_{i+1}} \varphi'_i(x)\varphi'_i(x)dx.$$

Zde funkce φ_i je lineární na každém z podintervalů, derivace φ_i' je tedy konstanta a lze snadno spočítat

$$k_{ii} = 2\int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_i(x)dx = \frac{2}{h}.$$

Obdobně spočteme prvek k_{ii+1}(viz Obrázek 1- společný nosič je $< x_i, x_{i+1} >$)

$$k_{ii+1} = \int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_{i+1}(x)dx = -\frac{1}{h},$$

a prvek $k_{ii-1} = -\frac{1}{h}$. Matice $\boldsymbol{K}$ po vytknutí $\frac{1}{h}$ bude tedy vypadat

$$\boldsymbol{K} = \frac{1}{h}\begin{pmatrix} 2 & -1 & \cdots & 0 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ 0 & \cdots & -1 & 2 \end{pmatrix},$$

Řešíme tedy soustavu lineárních rovnic s maticí $\boldsymbol{K}$ a pravou stranou $\boldsymbol{b}$, jejíž prvky jsou dány při užití numerické integrace

$$b_i = \int_0^1 f(x)\varphi_i(x)dx \cong f(x_i) \cdot h.$$

4. Přímé metody řešení lineárních soustav

Uvažujme dále soustavu lineárních rovnic

$$\boldsymbol{Ax} = \boldsymbol{b},$$

kde $\boldsymbol{A}$ je regulární matice typu $n \times n$. Tuto soustavu můžeme řešit tzv. přímými metodami. Přímé metody vedou konečným počtem kroků k přesnému výsledku. V praktické realizaci ale vznikají zaokrouhlovací chyby během výpočtu. Tyto chyby se mohou akumulovat a výsledek se může velice lišit od správného řešení. Chyby se dají minimalizovat správným výběrem pivota, tzv. hlavního prvku.

4.1. Gaussova eliminační metoda, pivotace

Gaussova eliminační metoda je bezpochyby nejznámější metoda pro řešení soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. Princip spočívá v postupné eliminaci proměnných užitím ekvivalentních úprav jako je přičtení lineární kombinace jiných řádků k vybranému řádku. Matici $\boldsymbol{A}$ převedeme na matici $\boldsymbol{U}$ a matice $\boldsymbol{L}$ reprezentuje provedené úpravy. Matice $\boldsymbol{L}$ je dolní trojúhelníková matice, to znamená s jedničkami na hlavní diagonále a nulovými prvky nad diagonálou. Matice $\boldsymbol{U}$ je horní trojúhelníková matice, která má nenulové prvky na hlavní diagonále a nuly pod hlavní diagonálou. Tento rozklad provedeme pomocí n kroků, viz [3].

$$\boldsymbol{A} = \boldsymbol{A^{(1)}} \to \boldsymbol{A^{(a)}} \to \cdots \to \boldsymbol{A^{(n)}} = \boldsymbol{U} = \boldsymbol{L^{-1}A}. \tag{5}$$

V obecném kroku $\boldsymbol{A}^{(k)} \to \boldsymbol{A}^{(k+1)}$ vynulováním všech prvků pod diagonálou ve sloupci k vznikne z matice $\boldsymbol{A}^{(k)}$ nová matice $\boldsymbol{A}^{(k+1)}$. V kroku k tedy vypočítáme prvky nové matice $\boldsymbol{A}^{(k+1)}$pomocí vzorce

$$\begin{gathered} m_{ik} \coloneqq \frac{a_{ik}}{a_{kk}}, \\ a_{ij} \coloneqq a_{ij} - m_{ik}a_{kj}, \\ b_i \coloneqq b_i - m_{ik}b_k, \\ i = k+1, \dots, n, \;\; j = k+1, \dots, n. \end{gathered} \tag{6}$$

Jelikož při eliminaci násobíme rovnici číslem m_{ik}, nazýváme tento člen multiplikátor. Řešením původní soustavy $\boldsymbol{Ax} = \boldsymbol{b}$ pak získáme pomocí zpětné substituce pro řešení soustavy $\boldsymbol{Ux} = \boldsymbol{b}$. Tedy postupně počítáme, $pro\ i = n, \dots, 1$

$$\mathrm{x_i} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n}\left(x_j a_{ij}\right)\right). \tag{7}$$

Tento algoritmus funguje se složitostí $O(n^3)$ operací pro eliminaci a $O(n^2)$ při zpětném kroku. Problém může nastat například v případě, že daná matice $\boldsymbol{A}$ je singulární. Další problém může nastat v případě vynulování pivotu (tj. prvku a_{kk}) v některém z předchozích kroků. Toto vyřešíme volbou nenulového pivotu z jiného řádku, viz [4].

V každém kroku k vybereme z k-tého sloupce pod diagonálou největší prvek v absolutní hodnotě (částečný výběr ze sloupce). Obdobně lze provést i výběr z řádku (částečný výběr

z řádku), nebo z řádku i sloupce (úplný výběr hlavního prvku) z matice $\boldsymbol{A}^{(k)}$. Samozřejmě spolu s pivotem prohodíme celý řádek nebo sloupec, viz [3].
Gaussova eliminace lze využít bez pivotace, například pro matice $\boldsymbol{A}$, která je ostře diagonálně dominantní nebo matice $\boldsymbol{A}$, která je M-matice, viz [4]. V řadě případů matice vzniklé Metodou konečných prvků diskretizací jsou M matice.
Zavedeme pojem reziduum. Reziduum značené $\boldsymbol{r}$ se vypočítá jako

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}},$$

kde $\tilde{\boldsymbol{x}}$ je přibližné řešení. Pokud je $\boldsymbol{A}$ regulární matice, chyba by měla být $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = 0$, ale vlivem zaokrouhlovací chyby může být nenulová. Když rovnici $\boldsymbol{Ax} = \boldsymbol{b}$ vynásobíme libovolnou nenulovou konstantou, řešení se nezmění, ale reziduum ano. Z toho důvodu zavádíme pojem velikost relativního rezidua viz [3]

$$R(\tilde{\boldsymbol{x}}) = \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\| \cdot \|\tilde{\boldsymbol{x}}\|}.$$

4.2. LU rozklad

Pokud použijeme Gaussovu eliminační metodu pro vyřešení jedné soustavy rovnic s pravou stranou $\boldsymbol{b}$ a následně potřebujeme řešit soustavu se stejnou maticí $\boldsymbol{A}$, ale s odlišnou pravou stranou, musíme celý postup opakovat. Toto je velice nežádoucí, protože v reálných problémech je často potřeba porovnávat výsledky pro určitý problém a počáteční podmínky se mohou měnit. Tento nedostatek lze vyřešit převedením rovnice $\boldsymbol{Ax} = \boldsymbol{b}$ na $\boldsymbol{LUx} = \boldsymbol{b}$, kde $\boldsymbol{A} = \boldsymbol{LU}$, $\boldsymbol{L}$ je dolní trojúhelníková matice s jedničkami na diagonále a $\boldsymbol{U}$ je horní trojúhelníková matice s nenulovými prvky na diagonále. Nejprve vypočítáme $\boldsymbol{Ly} = \boldsymbol{b}$ a v dalším kroku provedeme $\boldsymbol{Ux} = \boldsymbol{y}$. V případě znalosti matic $\boldsymbol{L}$ a $\boldsymbol{U}$ lze soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ vyřešit velmi efektivně v $O(n^2)$ krocích. Samostatný LU rozklad realizujeme pomocí vzorců viz [5]

$$u_{11} = a_{11},$$
$$l_{i1} = \frac{a_{i1}}{u_{11}},$$
$$u_{1r} = a_{1r},$$

a dále pro $r = 2, \dots, n$ počítáme nejprve pro $i = 2, \dots, r$ (8)

$$u_{ir} = a_{ir} - \sum_{j=1}^{i-1} l_{ij} u_{jr},$$

poté pro $i = r + 1, \dots, n$

$$l_{ir} = \frac{1}{u_{rr}} (a_{ir} - \sum_{j=1}^{r-1} l_{ij} u_{jr}).$$

Při programování daného problému se prvky matice $\boldsymbol{A}$ přepisují maticemi $\boldsymbol{L}$ a $\boldsymbol{U}$. Jelikož matice $\boldsymbol{L}$ má jedničky na hlavní diagonále, prvky matic $\boldsymbol{L}$ a $\boldsymbol{U}$ se v paměti uchovávají na místě původní matice $\boldsymbol{A}$. Pří částečné pivotaci nemusíme řádky prohazovat a stačí nám pouze pomocný vektor indexů. Výsledné umístění řádků je pouze nějaká permutace 1,…, n, viz [3].

4.3. QR rozklad

Jestliže je matice $\boldsymbol{A}$ regulární, lze LU rozklad zapsat pomocí diagonálních matic $\boldsymbol{L}$ a $\boldsymbol{M}$, které mají na diagonále 1 a pomocí diagonální matici $\boldsymbol{D}$, která má prvky pouze na diagonále. Tedy

$$\boldsymbol{A} = \boldsymbol{LDM^T}, \tag{9}$$

kde $\boldsymbol{M^T}$ značí transponovanou matici z matice $\boldsymbol{M}$. Užitím tohoto rozkladu lze soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ vyřešit postupným řešením

$$\boldsymbol{Ly} = \boldsymbol{b}, \qquad \boldsymbol{Dz} = \boldsymbol{y}, \qquad \boldsymbol{M^T x} = \boldsymbol{z}. \tag{10}$$

Za předpokladu, že $\boldsymbol{A}$ je symetrická matice, pak matice $\boldsymbol{M}$ je rovna matici $\boldsymbol{L}$ a tedy vztah $\boldsymbol{A} = \boldsymbol{LDM^T}$ přejde do $\boldsymbol{A} = \boldsymbol{LDL^T}$.

Tento rozklad předpokládá, že matice $\boldsymbol{A}$ je čtvercová. Zobecnění soustavy na soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ kde matice $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ (n sloupců a m řádků), vektor $\boldsymbol{x} \in \mathbb{R}^n$ a vektor $\mathbf{b} \in \mathbb{R}^m$. Potom matici $\boldsymbol{A}$ je možné rozložit ve tvaru

$$\boldsymbol{A} = \boldsymbol{QR}, \tag{11}$$

kde $\boldsymbol{R}$ je horní trojúhelníková čtvercová matice typu $n \times \mathrm{n}$ s prvky rovné 1 na diagonále a pro matici $\boldsymbol{Q}$ platí

$$\boldsymbol{D} = \boldsymbol{QQ^T}, \boldsymbol{D} = diag(d_1, \dots, d_n)\ , d_k > 0\ pro\ k = 1, \dots, n.$$

Je-li $\boldsymbol{A}$ zapsaná ve tvaru (11) pak soustavu řešíme postupně jako

$$\boldsymbol{Dy} = \boldsymbol{Q^T b} \text{ a } \boldsymbol{Rx} = \boldsymbol{y}. \tag{12}$$

Tato metoda je stabilnější než LU rozklad, na druhou stranu je více náročná. Více viz [4].

4.4. Choleského rozklad

Choleského rozklad je speciálním případem LU rozkladu, použitým pro symetrické a pozitivně definitní matice. Matici $\boldsymbol{A}$ je možné rozložit pomocí Choleského rozkladu viz [6]

$$\boldsymbol{A} = \boldsymbol{LL^T}, \tag{13}$$

kde $\boldsymbol{L}$ je dolní trojúhelníková matice s kladnými prvky na diagonále. Toto rozložení je dáno jednoznačně. Výpočet je možné provést bez pivotace. Získání matice $\boldsymbol{L}$ je možné následovně pro prvky na diagonále viz [6]

$$l_{11} \coloneqq \sqrt{a_{11}}\,,$$

$$l_{kk} \coloneqq \sqrt{a_{kk} - \sum_{p=1}^{k-1} l_{kp}^2} \quad pro\ k = 2, \dots, n,$$

$$l_{1k} \coloneqq \frac{a_{1k}}{l11}, \tag{14}$$

$$l_{kk} \coloneqq \frac{1}{l_{kk}}\left(a_{ik} - \sum_{p=1}^{k-1} l_{ip} l_{kp}\right),$$

$$pro\ k = 2, \dots, n\,, \quad i = k+1, \dots, n \ \ a \quad k \neq n.$$

Učiněním tohoto rozkladu převedeme řešení soustavy rovnic $\boldsymbol{Ax} = \boldsymbol{b}$ na rovnici $\boldsymbol{L^T y} = \boldsymbol{b}$ a $\boldsymbol{Lx} = \boldsymbol{y}$**.** Jestliže matice $\boldsymbol{A}$ je pouze symetrická, můžeme ji rozložit pomocí $\boldsymbol{A} = \boldsymbol{LDL^T}$. V tomto případě však musíme použít pivotace, jinak je metoda nestabilní. Pivotaci můžeme efektivně provést pomocí Bunchova a Kaufmanova algoritmu, viz [4].

5. Klasické iterační metody

Iterační metody používáme pro řešení rovnic, které mají příliš velký počet neznámých a přímé metody jsou zde velice časově, paměťově a výpočtově náročné. V těchto případech můžou být iterační metody daleko efektivnější. Nejčastější příklad jsou řídké matice soustavy. Jestliže máme soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ kde $\boldsymbol{A}$ je regulární matice typu $n \times n$, $\boldsymbol{b} \in \mathbb{R}^n$ vektor pravé strany a $\boldsymbol{x}^* \in \mathbb{R}^n$ přesné řešení soustavy. V každém iteračním kroku se postupně blížíme k přesnému výsledku. Tedy vytváříme posloupnost vektorů $\{\boldsymbol{x}^{(k)}\} \in \mathbb{R}^n$, která při splnění daných podmínek soustavy konverguje k správnému řešení. Vybereme počáteční vektor $\boldsymbol{x}^{(0)}$ a ostatní vektory posloupnosti vyjádříme rekurzivně. Výsledný tvar tedy vypadá například jako viz [8]

$$\boldsymbol{x^k} = \boldsymbol{F}\left(\boldsymbol{x}^{(\boldsymbol{k-1})}\right), \tag{15}$$

kde F je nějaké lineární zobrazení. Jelikož je proveditelné množství iteračních kroků konečné, přesný výsledek pouze aproximujeme. V posledním kroku je nějaké dosažené řešení $\boldsymbol{x}^{(k)}$ a tedy chyba definovaná jako $\left|\left|\boldsymbol{x}^{(\boldsymbol{k})} - \boldsymbol{x}^*\right|\right|$ nějaká norma rozdílu přesného a aktuálního řešení. Poněvadž nezáleží na volbě počátečních podmínek, je možné předpokládat, že každá iterace je první, tedy se nám neakumuluje zaokrouhlovací chyba jako u přímých metod, pokud daná metoda konverguje. V této kapitole si nejdřív ukážeme prostou iterační metodu a v dalších kapitolách klasické iterační metody, které jsou speciálními případy této metody, viz [8].

5.1. Prostá iterační metoda

Pro účel analýzy konvergence se užívá prostá iterační metoda. Prostá iterační metoda uvažuje místo původní soustavy rovnic $\boldsymbol{Ax} = \boldsymbol{b}$ ekvivalentní tvar

$$\boldsymbol{x} = \boldsymbol{Ux} + \boldsymbol{v}, \tag{16}$$

kde $\boldsymbol{U}$ je čtvercová regulární matice typu $n \times n$ a $\boldsymbol{v} \in \mathbb{R}^n$ vektor. Pak posloupnost aproximací $\left\{\boldsymbol{x}^{(\boldsymbol{k})}\right\}$ je definována vztahem

$$\boldsymbol{x}^{(\boldsymbol{k+1})} = \boldsymbol{Ux}^{(\boldsymbol{k})} + \boldsymbol{v}, \quad k = 0,1, \ldots . \tag{17}$$

Vektor $\boldsymbol{x}^{(\boldsymbol{0})}$ zde bude počáteční aproximace. Většinou je volen nulový vektor. Následně si uvedeme větu, která nám řekne, kdy posloupnost vektor konverguje.

Věta 2 *Nechť* $\left\{\boldsymbol{x}^{(k)}\right\}$ *je posloupnost aproximací. Pak jestliže daná posloupnost konverguje, pak její limita je řešením soustavy* $\boldsymbol{x} = \boldsymbol{Ux} + \boldsymbol{v}$ *a tedy i soustavy* $\boldsymbol{Ax} = \boldsymbol{b}$**.** *Posloupnost konverguje nezávisle na volbě počátečních podmínek právě tehdy když* $\rho(\boldsymbol{U}) < 1$.

Věta je uvedená viz [8].

Jestliže pro některou z norem platí $\left|\left|\boldsymbol{U}\right|\right| < 1$ pak prostá iterační metoda konverguje nezávisle na volbě počátečních podmínek, a navíc chybu je možné odhadnout viz [8].
Zatímco nerovnost $\rho(\boldsymbol{U}) < 1$ nám ukazuje nutnou a postačující podmínku konvergence, pak nerovnost $\left|\left|\boldsymbol{U}\right|\right| < 1$ je postačující podmínka. Stačí nám ověřit pouze některou normu,

viz [7]. Dále si uvedeme Jacobiovu iterační metodu, která lze zapsat ve tvaru prosté iterační metody.

5.2. Jacobiova iterační metoda

Soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ nejprve zapíšeme ve složkovém zápise pro $i = 1,2, \dots n$, tedy

$$\sum_{k=1}^{n} a_{ik}x_k = b_i.$$

Z rovnice $\boldsymbol{Ax} = \boldsymbol{b}$ vyjádříme x_i. Tím dostaneme

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum\nolimits_{j<i} a_{ij}x_j^k - \sum\nolimits_{j>i} a_{ij}x_j^k),$$

kde zápisem suma $j < i$ rozumíme součet od $j = 1$ do $j = i - 1$ a obdobně pro zápis suma $j > i$ rozumíme součet od $j = i + 1$ do $i = n$. Rovnici lze ještě upravit na viz [8]

$$x_i^{k+1} = x_i^k + \frac{1}{a_{ii}}(b_i - \sum\nolimits_{j} a_{ij}x_j^k). \tag{18}$$

Tato metoda lze zapsat ve tvaru prosté iterační metody

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{U}_J\boldsymbol{x}^k + \boldsymbol{v}_J, \tag{19}$$

kde $\boldsymbol{U}_J$ je iterační matice Jacobiovy metody.

Indexy J zde značí, že jde o Jacobiovu metodu. Rozklad provedeme v maticovém tvaru, nejprve rozdělíme matici $\boldsymbol{A}$ s prvky a_{ij} na matice

$$\boldsymbol{P} = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix},$$

$$\boldsymbol{L} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \tag{20}$$

$$\boldsymbol{D} = \begin{pmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{pmatrix}.$$

Tedy matice $\boldsymbol{P}$ je horní trojúhelníková s nulami na diagonále, matice $\boldsymbol{L}$ je dolní trojúhelníková s nulami na diagonále a matice $\boldsymbol{D}$ je diagonální matice. Matici $\boldsymbol{A}$ lze tedy napsat ve tvaru

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{L} + \boldsymbol{P}, \tag{21}$$

a rovnici $\boldsymbol{Ax} = \boldsymbol{b}$ přepsat na

$$(\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{P})\boldsymbol{x} = \boldsymbol{b}. \tag{22}$$

Převedením matic $\boldsymbol{L}$ a $\boldsymbol{P}$ na pravou stranu a vynásobením rovnice zleva maticí $\boldsymbol{D}^{-1}$ získáme

$$\boldsymbol{x}^{k+1} = -\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{P})\boldsymbol{x}^k + \boldsymbol{D}^{-1}\boldsymbol{b}. \tag{23}$$

Matice $\boldsymbol{U_J} = -\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{P})$ a vektor $\boldsymbol{v_J} = \boldsymbol{D}^{-1}\boldsymbol{b}$ **.** Jelikož zápis odpovídá prosté iterační metodě, lze konvergenci vyšetřit její pomocí, viz [7].

Věta 3 *Jacobiova metoda je konvergentní právě tehdy když* $\rho(\boldsymbol{U_J}) < 1.$

Věta je uvedena viz [7].

Abychom zjistili spektrální poloměr, využijeme tvrzení, které nám ukáže, že číslo $\lambda \in \mathbb{C}$ je vlastní číslo matice $\boldsymbol{U_J}$ právě když je kořenem

$$\det(\boldsymbol{L} + \lambda\boldsymbol{D} + \boldsymbol{P}) = 0. \tag{24}$$

Důkaz: Toto tvrzení dokážeme díky tomu, jelikož vlastní číslo λ matice $\boldsymbol{U_J}$splňuje

$$\begin{aligned} 0 = \det(\boldsymbol{U_J} - \lambda\boldsymbol{E}) &= \det(-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{P}) - \lambda\boldsymbol{E}) \\ &= \det(-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{P}) - \lambda\boldsymbol{D}^{-1}\boldsymbol{D}) \\ &= \det\left(-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{P}+\lambda\boldsymbol{D})\right). \end{aligned} \tag{25}$$

Jelikož determinant součinu matic je součin jednotlivých matic a determinant $\boldsymbol{D}^{-1}$je nenulový, je tvrzení dokázané.■

Věta 4 *Je-li matice A ostře diagonálně dominantní, pak je Jacobiova iterační metoda konvergentní.*

Důkaz: Jelikož je matice ODD, platí: $|a_{ii}| > \sum_{i\neq j}|a_{ij}|\ \ pro\ všechna\ i$ platí tedy

$$\left|\left|\boldsymbol{U_J}\right|\right|_\infty = \max\left(\frac{1}{a_{ii}}\sum_{i\neq j}|a_{ij}|\right) < 1. \tag{26}$$

Tedy dle věty 4 metoda konverguje, viz [7]. ■

Chyba se při této metodě počítá pomocí rezidua, které v každém kroku vypočítáme jako

$$\boldsymbol{r}^{(k)} = \left|\left|\boldsymbol{x}^k - \boldsymbol{x}^{(k-1)}\right|\right|. \tag{27}$$

Budeme tedy sledovat, jak moc se iterační kroky blíží. Výpočet ukončíme v případě, že reziduum bude menší než požadovaná velikost. Chybu je možné odhadnout pomocí přímé iterační metody.

5.3. Gaussova-Seidelova iterační metoda

Obdobně jako u Jacobiovy metody zapíšeme soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ ve složkovém zápise

$$\sum_{k=1}^{n} a_{ik}x_k = b_i.$$

Vyjádřením x_i a nahrazením v sumě x_k^k ,x_k^{k+1}pro $k < i$ nám vznikne rekurentní vyjádření Gaussovy-Seidelovy metody, kdy pomocí iterace $\boldsymbol{x^k}$ vyjádříme vektor $\boldsymbol{x}^{k+1}$

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum\nolimits_{j<i} a_{ij}x_j^{(k+1)} - \sum\nolimits_{j>i} a_{ij}x_j^{(k)}). \tag{28}$$

Zase si zde ukážeme, jak tuto metodu zapsat jako prostou iterační metodu

$$\boldsymbol{x}^{k+1} = \boldsymbol{U}_{GS}\boldsymbol{x}^{k} + \boldsymbol{v}_{GS}. \tag{29}$$

Tato metoda pouze modifikuje Jacobiovu metodu. Předpokládáme, že rovnice počítáme popořadě a v každém následujícím výpočtu již dosazujeme vypočtené hodnoty. Ve své podstatě se nám vektor $\boldsymbol{x}^{k}$ aktualizuje po každé vyčíslené rovnici. Metoda je někdy nazývána metodou postupných oprav. Jacobiova metoda vektor $\boldsymbol{x}^{k}$ přepočítává až na konci každé iterace. Z toho důvodu Gaussova-Seidelova metoda spotřebovává méně paměti programu, nemusíme v paměti mít vektor $\boldsymbol{x}^{k}$ i $\boldsymbol{x}^{k+1}$. Z výše uvedeného plyne, že záleží, v jakém pořadí rovnice vyhodnocujeme. Tedy u vektoru $\boldsymbol{x}^{k+1}$se proházením rovnic změní nejen pořadí prvků, ale i jejich hodnota. Toto pořadí nám ovlivní zejména konvergenci. Můžeme ji značně zrychlit ale i snížit, viz [4].

Dále zde uvedeme rozklad pomocí matic, aby bylo lépe ukázáno, jak tato metoda souvisí s přímou iterační metodou. Jak z rovnice $\boldsymbol{Ax} = \boldsymbol{b}$ přejdeme ke Gaussově-Seidelově iterační rovnici. Matici $\boldsymbol{A}$ rozdělíme na matice $\boldsymbol{D}, \boldsymbol{L}$ a $\boldsymbol{P}$. Tyto matice jsou zase diagonální, horní trojúhelníková s nulami na diagonále a dolní trojúhelníková s nulami na diagonále dány vztahem (*20*). Rovnici $\boldsymbol{Ax} = \boldsymbol{b}$ přepíšeme na

$$(\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{P})\boldsymbol{x} = \boldsymbol{b},$$

poté převedeme do tvaru

$$(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x}^{k+1} = -\boldsymbol{P}\boldsymbol{x}^{k} + \boldsymbol{b}.$$

Pak ji vynásobíme zleva $(\boldsymbol{D} + \boldsymbol{L})^{-1}$ a vznikne nám výsledný tvar

$$\boldsymbol{x}^{k+1} = -(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{P}\boldsymbol{x}^{k} + (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{b}. \tag{30}$$

Matice $\boldsymbol{U}_{GS} = -(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{P}$ a vektor $\boldsymbol{v}_{GS} = (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{b}$. Matice $(\boldsymbol{D} + \boldsymbol{L})^{-1}$ bude mít tvar viz [7]

$$(\boldsymbol{D} + \boldsymbol{L}) = \begin{pmatrix} a_{11} & \cdots & \cdots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & a_{nn} \end{pmatrix}. \tag{31}$$

I zde si uvedeme, jak vypočítat vlastní čísla a díky tomu určit spektrální poloměr matice $\boldsymbol{U}_{GS}$.

V tomto případě nám pomůže tvrzení, že číslo $\lambda \in \mathbb{C}$ je vlastní číslo matice $\boldsymbol{U}_{GS}$ právě tehdy, platí-li

$$\det(\lambda\boldsymbol{L} + \lambda\boldsymbol{D} + \boldsymbol{P}) = 0. \tag{32}$$

Důkaz: Toto tvrzení dokážeme díky tomu, že vlastní číslo matice $\boldsymbol{U}_{GS}$ splňuje

$$\det(\boldsymbol{U}_{GS} - \lambda\boldsymbol{E}) = 0.$$

Matici $\boldsymbol{U}_{GS}$ je možné nahradit $-(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{P}$ a jednotkovou matici nahradíme součinem $(\boldsymbol{D} + \boldsymbol{L})^{-1}(\boldsymbol{D} + \boldsymbol{L})$ a postupně upravíme

$$\det\big(-(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{P} - \lambda(\boldsymbol{D} + \boldsymbol{L})^{-1}(\boldsymbol{D} + \boldsymbol{L})\ \big) = 0,$$
$$\det(-(\boldsymbol{D} + \boldsymbol{L})^{-1}(\lambda\boldsymbol{D} + \lambda\boldsymbol{L} + \boldsymbol{P}) = 0.$$

Vzhledem k tomu, že výsledný determinant je součin jednotlivých determinantů a determinant matice $(\boldsymbol{D}+\boldsymbol{L})^{-1}$ je nenulový, je tvrzení dokázáno, viz [7].

Následují dvě věty, které ukážou kdy Gaussova-Seidelova metoda konverguje.

Věta 5 (*Postačující podmínka konvergence) Jestli je matice* $\boldsymbol{A}$ *ostře diagonálně dominantní, pak Gaussova-Seidelova metoda konverguje nebo je-li daná matice symetrická a pozitivně definitní.*

Důkaz věty viz [8].

Věta 6 (N*utná a postačující podmínka konvergence*) *Gaussova-Seidelova metoda je konvergentní, jestliže* $\rho(\boldsymbol{U}_{GS})<1$.

Důkaz věty viz [8].

Abychom porovnali Gaussovu-Seidelovu a Jacobiovu metodu, uvedeme si zde Stein-Rosenbergovu větu. Ta ukáže vztahy mezi konvergencí a rychlostí konvergence jednotlivých metod.

Věta 7 *(Stein-Rosenberg) Nechť* $\boldsymbol{A}$ *je matice typu* $n\times n$ *s prvky* a_{ij}. *Jestliže platí* $a_{ij}\leq 0$ *pro všechny* ij *a* $a_{ii}\geq 0$ *pro všechny* $i=1,\dots,n$ *pak platí právě jedno z následujících tvrzení:*

$$(i)\quad 0\leq\rho(\boldsymbol{U}_{GS})<\rho(\boldsymbol{U}_{J})<1$$
$$(ii)\quad 1<\rho(\boldsymbol{U}_{J})<\rho(\boldsymbol{U}_{GS})$$
$$(iii)\quad \rho(\boldsymbol{U}_{J})=\rho(\boldsymbol{U}_{GS})=0$$
$$(iv)\quad \rho(\boldsymbol{U}_{j})=\rho(\boldsymbol{U}_{GS})=1$$

Důkaz věty viz [4].

Z této věty plyne, že pro speciální matice (např. M matice) obě metody buď konvergují nebo divergují a Gaussova-Seidelova metoda v případě konvergence, konverguje dřív. V případě divergence, Gaussova-Seidelova metoda diverguje pomaleji. Tyto speciální matice se ale často objevují při diskretizaci pomocí metody konečných prvků. Následující věta uvažuje další speciální případ.

Věta *8* *Nechť* $\boldsymbol{A}$ *je tři diagonální matice. Jestliže* λ *je vlastní číslo matice* $\boldsymbol{U}_{J}$ *pak je* λ^2 *vlastní číslo matice* $\boldsymbol{U}_{GS}$. *Analogicky platí, jestliže je* μ *vlastní číslo matice* $\boldsymbol{U}_{GS}$ *pak je* $\sqrt{\mu}$ *vlastní číslo matice* $\boldsymbol{U}_{J}$**.** *Gaussova-Seidelova metoda konverguje tehdy, a pouze tehdy, jestliže konverguje Jacobiova metoda a navíc platí*

$$\rho(\boldsymbol{U}_{GS})=\left[\rho(\boldsymbol{U}_{J})\right]^2.$$

Důkaz věty viz [4].

5.4. SOR- (super relaxační metoda)

Jak je vidět v přechozí kapitole, rychlost konvergence závisí na spektrálním poloměru. Z toho důvodu se nabízí, jestli nejde nějak vhodně změnit spektrální poloměr tak aby iterační metoda konvergovala rychleji. Proto vznikla relaxační metoda, která se dá

použít v podstatě na kteroukoliv iterační metodu s předpisem $\boldsymbol{x}^{(k+1)} = f(\boldsymbol{x}^k)$, ale zde relaxační parametr ω přidáme ke Gaussově-Seidelově iterační metodě, který při vhodném zvolení zrychluje konvergenci. Rekurentní vyjádření vektoru $\mathbf{x}^{\mathrm{k+1}}$ pomocí vektoru $\mathbf{x}^{\mathrm{k}}$ je viz [4], $pro\ i = 1, \dots, n$

$$\mathrm{x_i^{k+1}} = (1-\omega)\mathrm{x_i^k} + \frac{\omega}{\mathrm{a_{ii}}}(\mathrm{b_i} - \sum_{\mathrm{j<i}} \mathrm{a_{ij}x_j^{(k+1)}} - \sum_{\mathrm{j>i}} \mathrm{a_{ij}x_j^{(k)}}). \qquad (33)$$

Ze vzorce plyne, že v případě zvolení $\omega = 1$, se výsledná metoda shoduje s Gaussovou-Seidelovou.

Teď si uvedeme, jak vypadá maticový zápis této metody. Nahrazením matice $\boldsymbol{A}$ v rovnici $\boldsymbol{Ax} = \boldsymbol{b}$ maticemi $\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{U}$ a doplněním relaxačního parametru vznikne, viz [4]

$$(\boldsymbol{D} + \omega\boldsymbol{L})\boldsymbol{x}^{k+1} = \omega\boldsymbol{b} - [\boldsymbol{U} + (\omega - 1)\boldsymbol{D}]\boldsymbol{x}^k.$$

Jestliže přenásobíme rovnici $(\boldsymbol{D} + \omega\boldsymbol{L})^{-\mathbf{1}}$ zleva, vznikne nám tvar

$$\boldsymbol{x}^{k+1} = (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\omega\boldsymbol{b} - (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}[\omega\boldsymbol{U} + (\omega - 1)\boldsymbol{D}]\boldsymbol{x}^k.$$

Z toho lze vidět, že matice $\boldsymbol{U}_\omega$ se bude rovnat

$$\boldsymbol{U}_\omega = (\boldsymbol{D} + \omega\boldsymbol{L})^{-\mathbf{1}}[\omega\boldsymbol{U} + (\omega - 1)\boldsymbol{D}]. \qquad (34)$$

Zase je zde vidět, že pokud volíme $\omega = 1$ pak $\boldsymbol{U}_{GS} = \boldsymbol{U}_\omega$.

Zde si uvedeme věty, které nám ukážou, v jakém intervalu vybrat relaxační parametr ω a jaká je jeho optimální velikost.

Věta 9 (*Kahan*) *Iterační metoda SOR konverguje v případě, že*

$$\rho(\boldsymbol{U}_\omega) \geq |\omega - 1|.$$

Vidíme, že pro konvergenci je nutné, aby relaxační parametr omega splňoval

$$0 < \omega < 2,$$

Důkaz věty viz [4].

Za předpokladu že Jacobiova metoda konverguje pro danou matici $\boldsymbol{A}$, je možné tvrdit, že relaxační metoda konverguje pro všechny $0 < \omega < 1$.

Pokud je $\omega\epsilon(0;1)$ metody se nazývají dolní relaxace. Jestli je $\omega\epsilon(1;2)$ řekneme, že nastává horní relaxace tedy SOR. Následují dvě věty, které nám říkají pro jaké matice $\boldsymbol{A}$ metoda konverguje a jak vybrat parametr ω.

Věta 10 (*Ostrowski-Reich*) *Nechť matice $\boldsymbol{A}$ je symetrická a pozitivně definitní, pak relaxační metoda konverguje pouze tehdy, konverguje-li Jacobiova metoda.*

Věta 11 *Optimální relaxační parametr nám minimalizuje $\rho(\boldsymbol{U}_\omega)$ a tedy zrychluje konvergenci. V případě, že ho chceme vyjádřit, musíme předpokládat:*

(i) $0 < \omega < 2$

(ii) $Pro\ matici\ \boldsymbol{U}_J\ platí$: $\forall\lambda \in \mathbb{R}$

(iii) $Jacobiova\ metoda\ konverguje : \rho(\boldsymbol{U}_J) < 1$

(iv) $soustava\ má\ právě\ jedno$ řešení $det(\boldsymbol{A}) \neq 0$

Pak může být optimální relaxační parametr vyjádřen jako

$$\omega_{opt} = 1 + \sqrt{\frac{\mu}{1+\sqrt{1-\mu^2}}}. \qquad (35)$$

A výsledný spektrální poloměr bude

$$\rho\left(G_{\omega_{opt}}\right) = \omega_{opt} - 1.$$

Důkaz věty viz [4].

Pokud matice $\mathbf{A}$ nesplňuje výše uvedené podmínky, zjištění parametru by bylo daleko složitější.

Zde si uvedeme speciální případ Relaxační metody. Věty říkající, kdy metoda konverguje jsou pouhou modifikací vět již uvedených, proto je zde nebudeme znovu opakovat. Symetrickou relaxační metodu je možné použít, pokud matice $\boldsymbol{A}$ je symetrická a splňuje všechny předpoklady jako pro relaxační metodu. Algoritmus vypadá, že zkombinujeme dvě relaxační metody, aby výsledná iterační matice byla podobná matici $\boldsymbol{A}$**.** V prvním kroku použijeme klasickou relaxační metodu, ale ve druhém budeme počítat rovnice v opačném pořadí zase relaxační metodou. Díky podobě iterační matice se symetrickou maticí $\boldsymbol{A}$ je hlavní využití pro předpodmínění při jiných iteračních metodách. Rychlost konvergence je pro ω_{opt} pomalejší než pro relaxační metodu.

Nejprve tedy vypočítáme pomocí relaxační metody z vektoru $\boldsymbol{x}^k$ vektor $\boldsymbol{y}^{k+1}$ a poté pomocí rekurentního vzorce

$$\mathrm{x_i^{k+1}} = (1-\omega)y_\mathrm{i}^{\mathrm{k+1}} + \frac{\omega}{\mathrm{a_{ii}}}(\mathrm{b_i} - \sum\nolimits_{\mathrm{j<i}} \mathrm{a_{ij}} y_\mathrm{j}^{(\mathrm{k+1})} - \sum\nolimits_{\mathrm{j>i}} \mathrm{a_{ij}} \mathrm{x_j^{(k+1)}}). \qquad (36)$$

Pro $i = 1, \dots, n$. Vyjádříme vektor $\boldsymbol{x}^{(k+1)}$. Maticově vypadá iterační matice $\boldsymbol{U}_\omega^S$ následovně

$$\begin{aligned} \boldsymbol{U}_\omega^S = (\boldsymbol{D} + \omega\boldsymbol{U})^{-1}\boldsymbol{D}(\boldsymbol{D} + \omega\boldsymbol{L})^{-1} \times \\ \times [(1-\omega)\boldsymbol{D} - \omega\boldsymbol{L}]\boldsymbol{D}^{-1}[(1-\omega)\boldsymbol{D} - \omega\boldsymbol{U}]. \end{aligned} \qquad (37)$$

Podmínky konvergence jsou stejné jako pro klasickou super relaxační metodu, viz [9].

6. Moderní iterační metody

Nejprve si rozebereme základní a nejstarší metodu, největšího spádu (Gradientní metodu), poté metodu sdružených gradientů a v poslední podkapitole si uvedeme některé další metody, které dokáží řešit soustavy rovnic. Zejména takové, ve kterých matice soustavy $\boldsymbol{A}$ nesplňuje podmínku, že je symetrická a pozitivně definitní. Metody sdružených gradientů a největšího spádu jsou velice podobné, liší se pouze ve výběru směru. Metoda sdružených gradientů rychleji konverguje. Také si ukážeme, že konvergence jde zrychlit takzvaným předpodmíněním, kdy rovnici $\boldsymbol{Ax} = \boldsymbol{b}$ vynásobíme z leva maticí $\boldsymbol{P}$, která je regulární typu $n \times n$ a navíc pro ni platí, že je blízká matici $\boldsymbol{A}^{-1}$.
Zde si uvedeme některé moderní iterační metody. Nejprve uvažujme matici $\boldsymbol{A}$, která je symetrická a pozitivně definitní. Pro takovou matici lze řešení $\boldsymbol{Ax} = \boldsymbol{b}$ ekvivalentně zapsat jako hledání minima funkce funkcionálu F zobrazení $\mathbb{R}^n \to \mathbb{R}$, která bude o n proměnných dané předpisem

$$F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{x}^T\boldsymbol{b}. \tag{38}$$

Taková funkce se také nazývá kvadratickým funkcionálem. Následující věta nám ukáže, že se opravdu dá řešit rovnice $\boldsymbol{Ax} = \boldsymbol{b}$ jako minimalizace funkcionálu F, viz [7].

Věta 12 (*Řešení soustavy lineárních rovnic jako minimalizace funkce) Nechť matice* $\boldsymbol{A}$ *je symetrická a pozitivně definitní pak*

$$\boldsymbol{Ax}^* = \boldsymbol{b} \quad \Leftrightarrow F(\boldsymbol{x}^*) \leq F(\boldsymbol{x}) \quad \forall \boldsymbol{x} \epsilon \mathbb{R}^{\boldsymbol{n}}.$$

Důkaz povedeme dle [7]. Dosazením do vztahu (36) $\boldsymbol{x} + \alpha\boldsymbol{\beta}$ za $\boldsymbol{x}$ a upravením a užitím symetrie matice $\boldsymbol{A}$ dostáváme

$$F(\boldsymbol{x} + \alpha\boldsymbol{\beta}) - F(\boldsymbol{x}) = \alpha\boldsymbol{\beta}^{\boldsymbol{T}}(\boldsymbol{Ax} - \boldsymbol{b}) + \frac{1}{2}\alpha^2\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{\beta}.$$

Jestliže položíme $\boldsymbol{x} = \boldsymbol{x}^*$ je vidět, že pokud je $\boldsymbol{x}^*$ řešení soustavy, pak $F(\boldsymbol{x}) \geq F(\boldsymbol{x}^*)$ jelikož je matice $\boldsymbol{A}$ pozitivně definitní, tedy $\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{\beta} > 0$.
Je-li $\boldsymbol{x}^*$ minimem funkcionálu, pak funkce $F(\boldsymbol{x}^* + \alpha\boldsymbol{\beta}) - F(\boldsymbol{x}^*)$ má minimum pro $\alpha = 0$. Jestliže je derivace této funkce v bodě 0 rovna 0 a $\boldsymbol{\beta}^{\boldsymbol{T}}(\boldsymbol{Ax}^* - \boldsymbol{b}) = 0$ pro libovolné $\boldsymbol{\beta}^{\boldsymbol{T}}$, pak vektor $(\boldsymbol{Ax}^* - \boldsymbol{b})$ musí být nulový a máme řešení soustavy. ■

6.1. Metoda největšího spádu

Uvažujeme hledání minima F pomocí iterační metody. Obecně iterační krok zapíšeme ve tvaru

$$\boldsymbol{x^{k+1}} = \boldsymbol{x^k} + \alpha\boldsymbol{\beta}, \tag{39}$$

kde značí $\alpha\epsilon\mathbb{R}$ délku kroku ve směru $\boldsymbol{\beta}$ a vektor $\boldsymbol{\beta}\epsilon\mathbb{R}^n$ směr. Metoda největšího spádu je základní minimalizační metoda. Problém řešení rovnice $\boldsymbol{Ax} = \boldsymbol{b}$ převedeme na problém minimalizace funkce F dané předpisem $F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{x}^T\boldsymbol{b}$. V každém iteračním kroku hledáme takové $\boldsymbol{x}$, aby se velikost funkce zmenšovala a konstruujeme posloupnost vektorů $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^k\}$ pro které platí

$$F(\boldsymbol{x}^0) < F(\boldsymbol{x}^1) < \cdots < F(\boldsymbol{x}^k).$$

Posloupnost je daná vztahem $\boldsymbol{x^{k+1}} = \boldsymbol{x^k} + \alpha\boldsymbol{\beta}$. Otázka je, jakým způsobem najdeme v každém kroku nový směr a jaká bude jeho velikost.

Volba zde je velice jednoduchá, najdeme směr, kde funkce v daném bodě nejstrměji klesá a jdeme tak dlouho dokud je tato podmínka splněna. Jelikož předpokládáme pouze lineární rovnice, je funkce diferencovatelná. Největší nárůst funkce je ve směru gradientu dané funkce. V případě, že vezmeme záporný gradient, najdeme směr, ve kterém funkce klesá nejvíce. Tato metoda se někdy nazývá zig-zag, viz [10].

Nyní si uvedeme dvě věty, které nám ukážou, jakým směrem a jak velký krok je možné udělat v každé iteraci.

Věta 13 (*Směr iterace) Nechť $\boldsymbol{A}$ je SPD matice a funkce F dána vztahem (38). Pak směr nejprudšího klesání kvadratického funkcionálu je v bodě $\boldsymbol{x}$ dán směrem*

$$\boldsymbol{\beta} = \boldsymbol{b} - \boldsymbol{Ax}. \tag{40}$$

Důkaz provedeme viz [11]. Jestliže chceme provést krok $\boldsymbol{x^{k+1}} = \boldsymbol{x^k} + \alpha\boldsymbol{\beta}$ pak budeme minimalizovat funkcionál ve tvaru $F = (\boldsymbol{x} + \alpha\boldsymbol{\beta})$ tedy dosazením do funkcionálu nám vznikne

$$F = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} + \alpha\boldsymbol{\beta}^T\boldsymbol{Ax} + \alpha^2\boldsymbol{\beta}^T\boldsymbol{A\beta} - \boldsymbol{x}^T\boldsymbol{b} - \alpha^2\boldsymbol{\beta}^T\boldsymbol{b}.$$

Pro pevně zvolené $\boldsymbol{\beta}$ je rychlost změny funkcionálu při pohybu ve směru $\boldsymbol{\beta}$ rovná derivaci podle α v bodě $\alpha = 0$. Derivováním a dosazením vznikne

$$\left.\frac{\partial F(\alpha)}{\partial\alpha}\right|_{\alpha=0} = \boldsymbol{\beta}^T\boldsymbol{Ax} - \boldsymbol{\beta}^T\boldsymbol{b} = -\boldsymbol{\beta}^T(\boldsymbol{b} - \boldsymbol{Ax}).$$

Jestliže chceme tuto derivaci mít co nejmenší, musí součin $-\boldsymbol{\beta}^T(\boldsymbol{b} - \boldsymbol{Ax})$ být co největší, a to je v případě, že volíme $\boldsymbol{\beta} = \boldsymbol{b} - \boldsymbol{Ax}$.■

Věta 14 (*Velikost kroku) Nechť $\boldsymbol{A}$ je SPD matice, funkce F dána vztahem (38) a známe směr $\boldsymbol{\beta}\epsilon\mathbb{R}^n$ dán vztahem (40). Pak definujeme $\alpha\epsilon\mathbb{R}$ takové, že*

$$\alpha = \frac{\boldsymbol{\beta}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})}{\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta}}. \tag{41}$$

Důkaz povedeme viz [11]. Jestliže jsme v nějakém bodě a známe směr největšího spádu, zajímá nás jak velký krok α musíme udělat, aby byl funkcionál co nejmenší. Zase do funkcionálu F dosadíme obecný krok $F = (\boldsymbol{x} + \alpha\boldsymbol{\beta})$ a určíme hodnotu funkionálu

$$F = \frac{1}{2}(\boldsymbol{x} + \alpha\boldsymbol{\beta})^T\boldsymbol{A}(\boldsymbol{x} + \alpha\boldsymbol{\beta}) - (\boldsymbol{x} + \alpha\boldsymbol{\beta})^T\boldsymbol{b}.$$

Pro pevně zvolené $\boldsymbol{x}$ a pevně zvolené $\boldsymbol{\beta}$ minimalizace funkcionálu závisí pouze na α. Funkcionál tedy zderivujeme a položíme rovný nule. Vyjádřením pak získáme minimum pro

$$\alpha = \frac{\boldsymbol{\beta}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})}{\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta}}. \blacksquare$$

U metody největšího spádu volíme směr největšího spádu vztahem (40) a optimální délku kroku dle vztahu (41). Metodu největšího spádu můžeme nejlépe ukázat pomocí jejího algoritmu:

Nejprve zvolíme nějaké $\boldsymbol{x}^0$, a poté k-tý krok iterace pro lze zapsat $k = 1,2, \ldots$ následovně

$$\begin{aligned} \boldsymbol{r}^k &:= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k, \\ \alpha_k &:= \frac{((\boldsymbol{r}^k)^T, \boldsymbol{r}^k)}{((\boldsymbol{r}^k)^T, \boldsymbol{A}\boldsymbol{r}^k)}, \\ \boldsymbol{x}^{\boldsymbol{k+1}} &:= \boldsymbol{x}^k + \alpha_k\boldsymbol{r}^k. \end{aligned} \tag{42}$$

Tento postup opakujeme, dokud neplatí že

$$||\boldsymbol{r}^k|| < \varepsilon$$

pro nějaké zvolené $\varepsilon > 0$. Pokud je $\boldsymbol{r}^k = 0$ je zřejmé, že jsme dosáhli přesného výsledku. Pokud je matice $\boldsymbol{A}$ symetrická a pozitivně definitní, daná metoda vždy konverguje. Konvergence ale může být velmi pomalá pro špatně podmíněnou matici $\boldsymbol{A}$, viz [10].

6.2. Metoda sdružených gradientů

Volba směru největšího spádu není nejvýhodnější. Vraťme se k minimalizaci kvadratického funkcionálu a uvažujme předpis $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha\boldsymbol{\beta}$. V každém kroku nebudeme volit směr $\boldsymbol{r}^k$, ale směry, které jsou určitým způsobem ortogonální. Například lze volit posloupnost $\{\boldsymbol{p}^0, \boldsymbol{p}^1, \ldots, \boldsymbol{p}^k\}$ tak aby platilo

$$(\boldsymbol{p}^i, \boldsymbol{p}^j) = \boldsymbol{0}\ pro\ všechna\ i \neq j.$$

Toto lze zaručit například volbou $\boldsymbol{p}^i = \boldsymbol{e}^i$, tím ale dostáváme Gaussovu-Seidelovu iterační metodu. Jelikož matice $\boldsymbol{A}$ je symetrická a pozitivně definitní, lze uvažovat směry tzv. A-ortogonální, tj. viz [13]

$$(\boldsymbol{p}^i)^T\boldsymbol{A}\boldsymbol{p}^j = \boldsymbol{0}\ pro\ všechna\ i \neq j.$$

Neboli v každém kroku hledáme směr $\boldsymbol{p}^k$ tak, aby byl kolmý na všechny předchozí směry. Tím dostáváme tzv. Metodu sdružených gradientů. Tato metoda je velmi dobře analyzována v řadě publikací, viz např [4]. Bylo dokázáno, že stačí volit směr $\boldsymbol{p}$ A-ortogonální jen na směr přechozí, tím pro symetrické a pozitivně definitní matice je zaručena A-ortogonalita všech předchozích směrů.

Metodu sdružených gradientů je možné formulovat pomocí algoritmu:

V prvním kroku zvolíme $\boldsymbol{x}^0, \boldsymbol{r}^0 := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0$ a položíme $\boldsymbol{p}^0 = \boldsymbol{r}^0$. Pak počítáme k-tou iteraci jako

$$\begin{aligned} \alpha_k &:= \frac{(\boldsymbol{r}^k, \boldsymbol{p}^k)}{(\boldsymbol{p}^k, \boldsymbol{A}\boldsymbol{p}^k)}, \\ \boldsymbol{x}^{k+1} &:= \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k, \\ \boldsymbol{r}^{k+1} &:= \boldsymbol{r}^k - \alpha_k \boldsymbol{A}\boldsymbol{p}^k, \\ \boldsymbol{p}^{k+1} &:= \boldsymbol{r}^{k+1} + \frac{(\boldsymbol{r}^{k+1}, \boldsymbol{A}\boldsymbol{p}^k)}{(\boldsymbol{p}^k, \boldsymbol{A}\boldsymbol{p}^k)} \boldsymbol{p}^k. \end{aligned} \tag{43}$$

První krok je shodný s metodou největšího spádu. V dalších krocích ale již počítáme s jiným směrem. Výpočet ukončíme, je-li pro nějaké $||\boldsymbol{r}^k|| < \varepsilon$, kde $\varepsilon > 0$ je námi zvolená přesnost řešení. Problém může nastat v případě, že směr $\boldsymbol{p}^k = 0$ a my v k+1 kroku budeme dělit nulou. To se může stát pouze, je-li

$$\boldsymbol{x}^k = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

A tedy máme přesné řešení a výpočet je ukončen. Více viz [4].

O metodě je dokázáno, že dojde k přesnému řešení nejpozději v n krocích. Vlivem zaokrouhlovacích chyb však přesného výsledku většinou nedosáhneme, chyba je ale velmi malá. Jestliže matice $\boldsymbol{A}$ je symetrická a pozitivně definitní, její vlastní čísla jsou tedy všechny reálná kladná, je možné její podmíněnost vypočítat jako

$$\varkappa(\boldsymbol{A}) = \frac{\max(\lambda)}{\min(\lambda)}. \tag{44}$$

Rychlost konvergence pak lze odhadnout vztahem viz [12]

$$|e^k| \leq 2\left(\frac{\sqrt{\varkappa(\boldsymbol{A})} - 1}{\sqrt{\varkappa(\boldsymbol{A})} + 1}\right)^k |e^0|, \tag{45}$$

kde $\boldsymbol{e}^k = \boldsymbol{x}^k - \boldsymbol{x}^*$ je chyba a $\boldsymbol{x}^*$ je přesné řešení.

Nabízí se zde tedy otázka, jestli nelze podmíněnost matice $\boldsymbol{A}$ „vylepšit“, a tím konvergence zrychlit, to je primární předpodmínění soustavy. Toto lze zapsat následovně, místo rovnice $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ řešíme rovnici

$$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}, \tag{46}$$

kde $\widetilde{\boldsymbol{A}} = \boldsymbol{P}^{-1}\boldsymbol{A}$ a $\widetilde{\boldsymbol{b}} = \boldsymbol{P}^{-1}\boldsymbol{b}$. Pokud matice $\boldsymbol{P}$ je blízká matici $\boldsymbol{A}$ a zároveň dobře invertovatelná nebo lze snadno vyřešit soustavu $\boldsymbol{P}\boldsymbol{z} = \boldsymbol{g}$ pak

$$\varkappa(\widetilde{\boldsymbol{A}}) < \varkappa(\boldsymbol{A})$$

a tedy metoda sdružených gradientů bude konvergovat pro (46) rychleji. Matice $\widetilde{\boldsymbol{A}}$ musí být také symetrická. Lze ukázat, že je možné předpodmínění použit vynásobením zprava i zleva symetrickou maticí $\boldsymbol{P}^{\frac{1}{2}}$ a upravit na vztah, který je zaručeně symetrický a ve tvaru (46). Ideální případ by nastal při $\boldsymbol{P^{-1}} = \boldsymbol{A^{-1}}$ a z rovnice by se stala rovnice $\boldsymbol{x} = \boldsymbol{A^{-1}b}$. Matice $\widetilde{\boldsymbol{A}} = \boldsymbol{I}$, kde $\boldsymbol{I}$ je jednotková matice, by měla číslo podmíněnosti 1 a zároveň bychom řešení získali 1 iterací. Invertování matice $\boldsymbol{A}$ ale odpovídá přímému řešení soustavy. V případě zvolení $\boldsymbol{P} = \boldsymbol{I}$, se metoda shoduje s metodou bez předpodmínění, tedy s metodou Sdružených gradientů, viz [12].

Cílem předpodmínění je volit matici $\boldsymbol{P}$ tak, aby byla snadno invertovatelná a aby co nejlépe odpovídala matici $\boldsymbol{A}$. Jedna z možností je užít $\boldsymbol{A} = \boldsymbol{D}$, kde $\boldsymbol{D}$ je diagonální matice. Další obvyklou možností je použit tzv. neúplného LU rozkladu matice $\boldsymbol{A}$. Tedy matici $\boldsymbol{P} = \widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{U}}$, kde matice $\widetilde{\boldsymbol{L}}$ a $\widetilde{\boldsymbol{U}}$ počítáme postupem uvedeným v kapitole 4.2 ale pouze pro prvky matice $\boldsymbol{A}$, které jsou nenulové.

Algoritmus předpodmínění je následujíc viz [4]:

V prvním kroku zvolíme $\boldsymbol{x}^0$, $\boldsymbol{r}^0 \coloneqq \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0$ a položíme $\boldsymbol{p}^0 \coloneqq \boldsymbol{z^0} \coloneqq \boldsymbol{P^{-1}}\boldsymbol{r}^0$ a pak vypočítáme k tou iteraci jako

$$\begin{aligned} \alpha_k &\coloneqq \frac{(\boldsymbol{z}^k, \boldsymbol{p}^k)}{(\boldsymbol{p}^k, \boldsymbol{A}\boldsymbol{p}^k)}, \\ \boldsymbol{x}^{k+1} &\coloneqq \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k, \\ \boldsymbol{r}^{k+1} &\coloneqq \boldsymbol{r}^k - \alpha_k \boldsymbol{A}\boldsymbol{p}^k, \\ \boldsymbol{z}^{k+1} &\coloneqq \boldsymbol{P^{-1}}\boldsymbol{r}^{k+1}, \\ \boldsymbol{p}^{k+1} &\coloneqq \boldsymbol{z}^{k+1} + \frac{(\boldsymbol{z}^{k+1}, \boldsymbol{r}^k)}{(\boldsymbol{z}^k, \boldsymbol{r}^k)} \boldsymbol{p}^k. \end{aligned} \tag{47}$$

Je zde vidět, že oproti metodě sdružených gradientů počítáme zde navíc $\boldsymbol{z}^{k+1}$. Výpočet se ukončí, pokud bude platit, pro nějaké $||\boldsymbol{z}^k|| < \varepsilon$, kdy $\varepsilon > 0$ bude předem zvolená přesnost. Obdobně jako u předchozí metody platí viz [12]

$$|e^k| \leq 2\left(\frac{\sqrt{\varkappa(\widetilde{\boldsymbol{A}})} - 1}{\sqrt{\varkappa(\widetilde{\boldsymbol{A}})} + 1}\right)^k |e^0|, \tag{48}$$

kde $\widetilde{\boldsymbol{A}} = \boldsymbol{P}^{-1}\boldsymbol{A}$. První je předpodmínění diagonálou, nebo též Jacobiovo předpodmínění, kdy volíme $\boldsymbol{P}$ jako viz [4]

$$\boldsymbol{P} \coloneqq (a_{11}, a_{22}, \dots, a_{nn})^T.$$

Je snadno vidět, že výpočet $\boldsymbol{P^{-1}}$ snadný. Další možnost předpomínění je volba diagonální matice viz [4]

$$\boldsymbol{P} \coloneqq (c_1, c_2, \dots, c_n)^T \; kdy \; c_i \coloneqq \sqrt{\left(\sum_{j=1}^{n} \boldsymbol{a}_{ij}^2\right)}.$$

6.3. Možnosti výpočtu pro nesymetrické matice

Budeme předpokládat, že máme rovnici $\boldsymbol{Ax} = \boldsymbol{b}$, kde $\boldsymbol{x}, \boldsymbol{b} \epsilon \mathbb{R}^{\boldsymbol{n}}$. Navíc matice $\boldsymbol{A}$ je regulární typu $n \times n$ a obecně nesymetrická. Minimalizace reziduí $\left|\left|\boldsymbol{Ax} - \boldsymbol{b}\right|\right|^2$ vede na soustavu normálních rovnic $\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b}$. Tato soustava rovnic je shodná s metodou předpodmínění, v případě, že volíme

$$\boldsymbol{P}^{-1} \coloneqq \boldsymbol{A}^T.$$

Proto použijeme metodu sdružených gradientů a tím dostaneme metodu minimálních reziduí. V prvním kroku si zvolíme $\boldsymbol{x}^0$ a položíme $\boldsymbol{r}^0 \coloneqq \boldsymbol{b} - \boldsymbol{Ax}^0, \boldsymbol{p}^0 \coloneqq \boldsymbol{A}^T\boldsymbol{r}^0$ a pak vyjádříme k-tou iteraci jako

$$\begin{aligned} \alpha_k &\coloneqq \frac{|\boldsymbol{A}^T\boldsymbol{r}^k|^2}{|\boldsymbol{Ap}^k|^2}, \\ \boldsymbol{x}^{k+1} &\coloneqq \boldsymbol{x}^k + \alpha_k\boldsymbol{p}^k, \\ \boldsymbol{r}^{k+1} &\coloneqq \boldsymbol{r}^k - \alpha_k\boldsymbol{Ap}^k, \\ \boldsymbol{p}^{k+1} &\coloneqq \boldsymbol{A}^T\boldsymbol{r}^{k+1} + \frac{|\boldsymbol{A}^T\boldsymbol{r}^{k+1}|^2}{|\boldsymbol{A}^T\boldsymbol{r}^k|^2}\boldsymbol{p}^k. \end{aligned} \tag{49}$$

Tato metoda zkonverguje stejně jako metoda sdružených gradientů nejpozději v n krocích. Číslo podmíněnosti matice $\boldsymbol{AA}^T$ je ale druhou mocninou čísla podmíněnosti matice $\boldsymbol{A}$. Konvergence této metody je tedy daleko pomalejší a nevyplatí se jí používat na symetrické matice, viz [4].

Kromě metody minimálních reziduí se pro nesymetrické matice dají používat také metody GMRES a BICG. Zde si uvedeme metodu Bi-konjungovaných gradientů. Protože matice $\boldsymbol{A}$ není symetrická, nelze bez uložení předchozích směrů najít nový který splňuje podmínku ortogonality a zároveň minimalizuje funkcionál. Konstruujeme tedy dvě posloupnosti $\{\boldsymbol{r}^0, \boldsymbol{r}^1, \dots, \boldsymbol{r}^k\}$ a $\{\tilde{\boldsymbol{r}}^0, \tilde{\boldsymbol{r}}^1, \dots, \tilde{\boldsymbol{r}}^k\}$ tak že platí

$$(\boldsymbol{r}^i, \tilde{\boldsymbol{r}}^j) = 0 \; pro \; všechna \; i \neq j.$$

O takových posloupnostech řekneme, že jsou biortogonální. V této metodě ale neminimalizujeme funkcinál. V případě použití této metody na matici $\boldsymbol{A}$, která je symetrická konverguje ve stejném počtu iterací jako metoda sdružených gradientů, ale je dvakrát více časově náročná kvůli počítání dvou reziduí, viz [14].

Algoritmus metody Bi-conjugovaných gradientů vypadá následovně:

V prvním kroku volíme $\boldsymbol{x}^0$, položíme $\boldsymbol{p}^0 \coloneqq \boldsymbol{r}^0 \coloneqq \boldsymbol{b} - \boldsymbol{Ax}^0$ a spočteme

$$\rho_{\boldsymbol{0}} = (\boldsymbol{r}^k, \tilde{\boldsymbol{r}}^k) \neq 0.$$

Dále zvolíme $\tilde{\boldsymbol{r}}^0$,libovolně tak aby neplatilo $(\boldsymbol{r}^0, \tilde{\boldsymbol{r}}^0) = 0$. Je možné zvolit $\tilde{\boldsymbol{r}}^0 = \boldsymbol{r}^0$. Potom položíme $\tilde{\boldsymbol{p}}^0 \coloneqq \tilde{\boldsymbol{r}}^0$ a pro $k = 0,1,2, \dots$ počítáme viz [4]

$$
\begin{aligned}
\boldsymbol{v}^k &:= \boldsymbol{A}\boldsymbol{p}^k, \\
\alpha_k &:= \frac{\rho_k}{(\boldsymbol{v}^k, \tilde{\boldsymbol{p}}^k)}, \\
\boldsymbol{x}^{k+1} &:= \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k, \\
\boldsymbol{r}^{k+1} := \boldsymbol{r}^k - \alpha_k \boldsymbol{v}^k, \quad & \boldsymbol{r}^{k+1} := \boldsymbol{r}^k - \alpha_k \boldsymbol{A}^T \tilde{\boldsymbol{p}}^k, \\
\rho_{k+1} &:= (\boldsymbol{r}^{k+1}, \tilde{\boldsymbol{r}}^{k+1}), \\
\beta_{k+1} &:= \frac{\rho_{k+1}}{\rho_k}, \\
\boldsymbol{p}^{k+1} := \boldsymbol{r}^{k+1} + \beta_{k+1}\boldsymbol{p}^k, \quad & \tilde{\boldsymbol{p}}^{k+1} := \tilde{\boldsymbol{r}}^{k+1} + \beta_{k+1}\tilde{\boldsymbol{p}}^k.
\end{aligned}
\tag{50}
$$

Tato metoda se dá také dále vylepšovat a modifikovat za účelem zrychlení konvergence.

7. Numerické výsledky

V této kapitole si ukážeme možnosti, jakým způsobem jsme realizovali vybrané iterační metody v jazyce C. V druhé části si je srovnáme z hlediska výpočetního času a konvergence. Iterační metody budeme nejprve používat na soustavy lineárních rovnic, které vzniknou diskretizací metodou konečných prvků. Nejprve je použijeme pro tří diagonální matici vzniklou z 1D problému. Zadruhé je budeme testovat na blokové matici, kterou také dostaneme metodou konečných prvků. V posledním případě je budeme zkoušet na obecnější matici. Dále uvedeme realizaci 5 vybraných iteračních metod a ukážeme výsledky jejich použití pro zmíněné soustavy.

7.1. Uložení řídkých matic

Jelikož použitím metod konečných prvků dostaneme soustavu lineárních rovnic, s řídkou maticí $\boldsymbol{A}$, není žádoucí uchovávat v paměti všechny prvky matice (velké množství nulových prvků). V této práci jsme pracovali se dvěma druhy uložení řídké matice.

První způsob je takzvaný triplet (COO-Coordinate format). Tento způsob je uživatelsky přijatelnější a nevyžaduje setříděné hodnoty prvků. Je vhodné ho používat např. při sestavování matice $\boldsymbol{A}$. Uložení matice se řeší strukturou, ve které uchováváme rozměr matice n, počet alokovaných členů a počet nenulových členů. Struktura vypadá následovně.

```
typedef struct {
        int n,nz,nalloc;
        int *I, *J;
        double *AIJ;
                } triplet;
```

Je možné, že je triplet nesetříděný a že se některé prvky opakují, tedy po setřídění tripletu a sečtení stejných prvků může být číslo značící nenulové prvky menší, než je počet alokovaných. Toto nám zrychluje program, který nemusí pracovat s nulovými prvky. Dále ve struktuře nalezneme tři pole o velikosti počtu alokovaných. V jednom poli uchováváme hodnotu daného prvku a_{ij} v druhých dvou jeho pozici, řádek i a sloupec j. Prvek a_{ij} v matici alokovaný jako k-tý bude uložen v tripletu jako

$$A.AIJ[k] = a_{ij},$$
$$A.I[k] = i,$$
$$A.J[k] = j.$$

Formát triplet ale není výhodný pro realizaci iteračních metod, například není jasné, kde jsou uloženy prvky i-tého řádku.

Pro iterační metody zvolíme formát CSR (Compressed sparse row) označovaný též jako CRF. Je reprezentován strukturou CRF, která obsahuje velikost matice, počet alokovaných a počet nenulových prvků. Dále obsahuje tři pole, ale pouze dvě budou o velikosti počtu alokovaných. První značené VAL uchovává hodnotu prvku a_{ij} a druhé značené J pozici

prvku ve sloupci. Prvky ve formátu CRF jsou setříděny po řádcích. Pozice na řádku jednotlivých řádků jsou pak ukládány v poli PI, které je alokováno dle velikosti matice n Na velikost $n+1$. Zde $P[i]$ obsahuje index kde začíná i-tý řádek a $PI[i+1]$ kde začíná i+1 řádek (tj. a tedy kde končí i-tý řádek). V souladu s konvencí jazyka C formát RCF používá indexaci řádků i sloupců $0, \ldots n-1$. Struktura vypadá následovně.

```
typedef struct {
    int       n,   nz, nalloc;
    int      *PI, *J;
    double   *VAL;
} CRF;
```

7.2. Realizace Jacobiovy metody

První realizovaná metoda je Jacobiova metoda viz kapitola 5.2. Tato metoda je náročnější na paměť, než Gaussova-Seidelova. Je potřeba uchovávat vektor $\boldsymbol{x}^k$ i vektor $\boldsymbol{x}^{k+1}$. Nejprve si zde uvedeme vzorec, ze kterého jsme vycházeli

$$x_i^{k+1} = x_i^k + \frac{1}{a_{ii}}(b_i - \sum_j a_{ij}x_j^k).$$

Metoda je realizována ve funkci jejíž hlavička vypadá

```
double crf_Jacobi(CRF *this, double *x, double *b, int niter, double eps);
```

Kde vlastní algoritmus je sepsán následovně

```
for(iter = 0; iter < niter; iter++)
{
    err = 0;
    for(i = 0; i < this->n; i++)
    {
        rezi = b[i];
        aii  = 0;
        for(k = this->PI[i]; k < this->PI[i+1]; k++)
        {
            aij    = this->VAL[k];
            j      = this->J[k];
            if(j == i) aii = aij;
            rezi -= aij * x[j] ;
        }
        err   = max(err, fabs(rezi));
        xn[i] += rezi / aii ;
    }
    printf("Chyba v iteraci k = %d je %e\n", iter, err);
    for(i = 0; i < this->n; i++) x[i] = xn[i];
    if (err < eps)
        break;
}

free(xn) ;
return err;
```

Zde uvažujeme pomocné pole xn, které obsahuje novou iteraci $\boldsymbol{x}^{k+1}$. Této funkci musíme dát matici $\boldsymbol{A}$, vektor pravé strany $\boldsymbol{b}$, maximální počet iterací niter a přesnost eps, při které bude výpočet ukončen. První cyklus odpovídá iteracím k=1,…(iter=k). V každé iteraci

nejprve vynulujeme chybu err. Poté postupně procházíme jednotlivé řádky 1,…,n. Do rezidua přepíšeme prvek pravé strany $\boldsymbol{b}$. Dalším cyklem projdeme všechny prvky matice $\boldsymbol{A}$ v řádku i a odčítáme a_{ij}a x_j^k. Během tohoto cyklu najdeme prvek a_{ii} na diagonále. Dělením rezidua prvkem a_{ii} dostáváme hodnotu nové aproximace $\boldsymbol{x}^{k+1}$.

7.3. Realizace Gaussovy-Seidelovy metody

Jelikož Gaussova-Seidelova metoda během výpočtu postupně aktualizuje vektor $\boldsymbol{x}$ není potřeba ukládat v paměti dvě pole, ale pole rovnou během výpočtu přepisujeme. O této metodě pojednáváme v kapitole 5.3. Vyjdeme z následujícího vzorce

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}).$$

Tento vzorec je také možné upravit na

$$x_i^{k+1} = x_j^k - \frac{1}{a_{ii}}(b_i - \sum_j a_{ij}x_j),$$

kde na pravé straně značíme

$$x_j \coloneqq x_j^k \; pro \; j < i \; a \; x_j \coloneqq x_j^{k+1} pro \; j > i \; .$$

Zde jsme záměrně nenapsali, zdali je prvek x již aktualizovaný nebo ne. V algoritmu nás to nezajímá, jelikož vektor se rovnou aktualizuje. Proto je možné použít stejný algoritmus jako u Jacobiovy metody s tím rozdílem, že vektor $\boldsymbol{x}$ rovnou přepisujeme po prvcích a neukládáme do jiného pole, než s kterým počítáme. Realizujeme ve funkci jejíž hlavička vypadá

```
double crf_Gauss-Seidl(CRF *this, double *x, double *b, int niter, double eps);
```

Kde vlastní algoritmus je sepsán následovně

```
for(iter = 0; iter < niter; iter++)
{
    err = 0;
    for(i = 0; i < this->n; i++)
    {
        rezi = b[i];
        aii  = 0;
        for(k = this->PI[i]; k < this->PI[i+1]; k++)
        {
            aij    = this->VAL[k];
            j      = this->J[k];
            if(j == i) aii = aij;
            rezi -= aij * x[j] ;

        }
        err   = max(err, fabs(rezi));
        x[i] += rezi / aii;

    }
    printf("Chyba v iteraci k = %d je %e\n", iter, err);
    if (err < eps)
        break;
}
return err;
```

Algoritmus funkce je tedy v podstatě analogický předešlému, proto ho zde již nebudeme komentovat. Se stejnými vstupními parametry.

7.4. Realizace Super relaxační metody

Navážeme Super relaxační metodou, která zrychluje konvergenci Gaussovy-Seidelovy metody. Metoda byla uvedena v kapitole 5.4. Pro připomenutí si zde uvedeme vzorec, z kterého postupnou úpravou je možné udělat lépe programovatelný

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}). \qquad (51)$$

Jako u předchozích metod je možné tento vzorec upravit na

$$x_i^{k+1} = (1-\omega)x_i^k + \omega x_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_j a_{ij}x_j\right),$$

kde obdobně jako u Gaussovy-Seidelovy metody značíme

$$x_j := x_j^k \; pro \; j < i \; a \; x_j := x_j^{k+1} pro \; j > i \; .$$

Dále upravíme na tvar, který budeme dále používat

$$x_i^{k+1} = x_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_j a_{ij}x_j\right).$$

Nyní si můžeme ukázat realizaci funkce. Její hlavička bude vypadat

```
double crf_SOR(CRF *this, double *x, double *b, int niter, double eps, double omega);
```

A algoritmus vypadá následovně

```
for(iter = 0; iter < niter; iter++)
{
    err = 0;
    for(i = 0; i < this->n; i++)
    {
        rezi = b[i];
        aii  = 0;
        for(k = this->PI[i]; k < this->PI[i+1]; k++)
        {
            aij    = this->VAL[k];
            j      = this->J[k];
            if(j == i) aii = aij;
            rezi -= aij * x[j] ;
        }
        err   = max(err, fabs(rezi));
        x[i] +=  omega * rezi / aii;
    }
    printf("Chyba v iteraci k = %d je %e\n", iter, err);
    if (err < eps)
        break;
}
return err;
```

Algoritmus je v podstatě stejný, jako u Gaussovy-Seidelovy metody. V podstatě jediný rozdíl je v přidání relaxačního parametru ω v místě počítání nové iterace. Tento postup nám při správně zvoleném relaxačním parametru dokáže zrychlit konvergenci.

7.5. Realizace metody největšího spádu

Nejprve si ukážeme metodu největšího spádu. Její algoritmus se skládá z několika kroků. Jak už bylo uvedeno v kapitole 6.1, tato metoda vychází z rovnice $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha\boldsymbol{\beta}$. Připomeneme si zde algoritmus

$$\boldsymbol{r}^k := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k,$$
$$\alpha_k := \frac{((\boldsymbol{r}^k)^T, \boldsymbol{r}^k)}{((\boldsymbol{r}^k)^T, \boldsymbol{A}\boldsymbol{r}^k)},$$
$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \alpha_k \boldsymbol{r}^k.$$

Jelikož časově nejnáročnější je násobení matice a vektoru, snažíme se těchto operací dělat v každém iteračním kroku co nejméně. Abychom zde nemuseli počítat součin $\boldsymbol{A}\boldsymbol{x}^k$ a $\boldsymbol{A}\boldsymbol{r}^k$, je možné vyjádřit $\boldsymbol{r}^{k+1}$ jako

$$\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \alpha_k \boldsymbol{A}\boldsymbol{r}^k,$$

kde byl výpočet $\boldsymbol{A}\boldsymbol{x}^k$ nahrazen $\boldsymbol{A}\boldsymbol{r}^k$, které využíváme při výpočtu α_k. Díky této úpravě budeme v každém kroku násobit matici a vektor pouze jednou. Teď si ukážeme, jak bude vypadat realizace této metody pomocí funkce. Hlavička funkce vypadá následovně

```
double crf_nejvetsispad(CRF *this, double *x,double *b, int niter, double eps);
```

Samotná funkce je realizovaná kódem

```
 crf_multiply(this,x,Ap);
 for(i=0;i < n ; i++)
              {
              rezi[i] = b[i] - Ap[i];
              }
 for(iter = 0 ; iter < niter ; iter ++ )
              {
               chyba = 0;
                cit   = 0;
                jmen  = 0;
                crf_multiply(this,rezi,Ap);
                for(i = 0 ; i < n ; i++)
                     {
                     cit   += rezi[i] * rezi[i];
                     jmen  += rezi[i] * Ap[i];

                     }
                alfa = cit/jmen;

                for ( i = 0; i < n ; i++)
                    {
                    x[i]    += alfa * rezi[i];
                    rezi[i] += -alfa * Ap[i] ;
                    chyba += rezi[i] * rezi[i];
                    }
                chyba = sqrt(chyba);
              printf("v iterraci %d je chyba %18.9e \n",iter,chyba);
              if (chyba < eps) break;
              }
return chyba;
```

Funkci předává matici $\boldsymbol{A}$ uloženou pomocí CRF formátu, pole reprezentující vektor $\boldsymbol{x}$, počet iterací niter a požadovanou přesnost eps, při které se výpočet ukončí. V prvním kroku pomocí cyklu vypočítáme reziduum, což je směr minimalizace F. Dále vypočítáme velikost kroku alfa, tak že si nejprve určíme čitatel a jmenovatel potřebný pro výpočet α_k dle vzorce. Nakonec najdeme hledaný vektor $\boldsymbol{x}^k$ a ve stejném cyklu vypočítáme chybu a reziduum, s kterým budeme počítat v dalším kroku. V posledním kroku zkontrolujeme, jestli není chyba menší než požadovaná přesnost.

7.6. Realizace metody sdružených gradientů

Metoda sdružených gradientů zlepšuje předchozí metodu, tak aby rychleji konvergovala, jak bylo uvedeno v kapitole 6.2. Připomeneme si zde vzorce pro tuto metodu. Pro $k = 0,1,2,\ldots$ počítáme dle

$$\alpha_k := \frac{(\boldsymbol{r}^k, \boldsymbol{p}^k)}{(\boldsymbol{p}^k, \boldsymbol{A}\boldsymbol{p}^k)},$$
$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k,$$
$$\boldsymbol{r}^{k+1} := \boldsymbol{r}^k - \alpha_k \boldsymbol{A}\boldsymbol{p}^k,$$
$$\boldsymbol{p}^{k+1} := \boldsymbol{r}^{k+1} + \frac{(\boldsymbol{r}^{k+1}, \boldsymbol{A}\boldsymbol{p}^k)}{(\boldsymbol{p}^k, \boldsymbol{A}\boldsymbol{p}^k)} \boldsymbol{p}^k.$$

Z těchto vzorců vychází následná realizace metody sdružených gradientů jejíž hlavička vypadá

```
double crf_gradient(CRF *this, double *x,double *b, int niter, double eps);
```

A její algoritmus je následující

```
for(iter = 0 ; iter < niter ; iter ++ )
 {
 chyba = 0;cit   = 0; jmen  = 0;
 crf_multiply( this , p , Ap );
 for(i = 0 ; i < n ; i++)
   {
   cit   += p[i] * rezi[i];
   jmen  += p[i] * Ap[i];
   }
 alfa = cit/jmen;
 cit   = 0;
 for ( i = 0; i < n ; i++)
   {
   x[i]    += alfa * p[i];
   chyba   += rezi[i] * rezi[i];
   rezi[i] += -alfa* Ap[i];
   cit     += rezi[i] * Ap[i];
   }
 chyba = sqrt(chyba);
 Beta= -cit/ jmen ;
 for(i = 0 ; i < n ; i++) p[i] = rezi[i] + Beta * p[i];
 printf("v iterraci %d je chyba %18.9e \n",iter,chyba);
 if (chyba < eps)break;
 }
```

V podstatě postupujeme stejně jako u metody největšího spádu, pouze přibyl krok s A-ortogonalizací směru $\boldsymbol{p}$. V každé iteraci nejdříve vynulujeme čitatel, jmenovatel a chybu. Vypočítáme si potřebný součin matice $\boldsymbol{A}$ a vektoru $\boldsymbol{p}^k$**.** Následně vypočítáme čitatel a jmenovatel potřebný k určení α_k. Dále vypočítáme vektor $\boldsymbol{x}^{k+1}$ a velikost rezidua, který později porovnáme s požadovanou přesností eps. Poté vypočítáme $\boldsymbol{r}^{k+1}$ vypočítáme nový čitatel (jmenovatel zůstává v celé iteraci stejný) a určíme $\boldsymbol{p}^{k+1}$, které budeme potřebovat v dalším kroku.

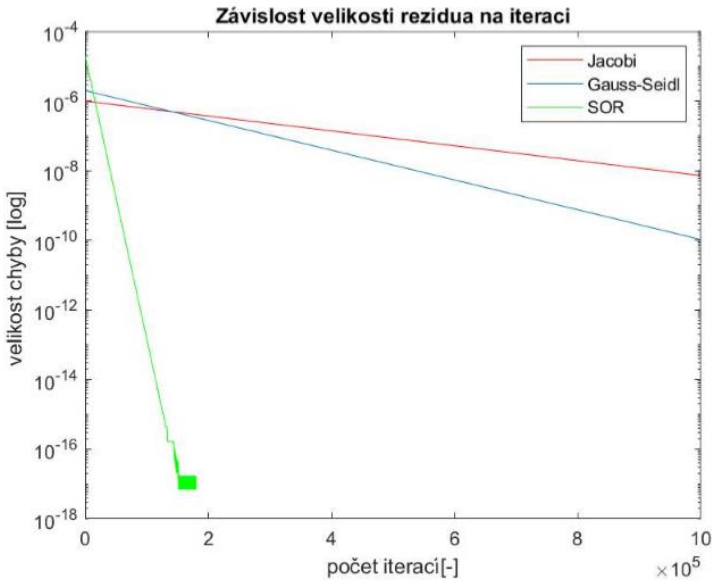
7.7. Použití a srovnání metod, jejich konvergence

Zde si ukážeme použití metod. Nejprve budeme uvažovat matici, která vznikne pří diskretizaci metodou konečných prvků 1D úlohy, viz kapitola 3. Danou soustavu budeme řešit užitím iteračních metod Jacobiovy, Gaussovy-Seidelovy a super relaxační, dále pak metodou největšího spádu a metodou sdružených gradientů. Matice vzniklá z diskretizace metodou konečných prvků, je tří diagonální s prvky 2 na hlavní diagonále a -1 na diagonálách vedlejších. Matice bude vypadat následovně

$$\boldsymbol{A} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

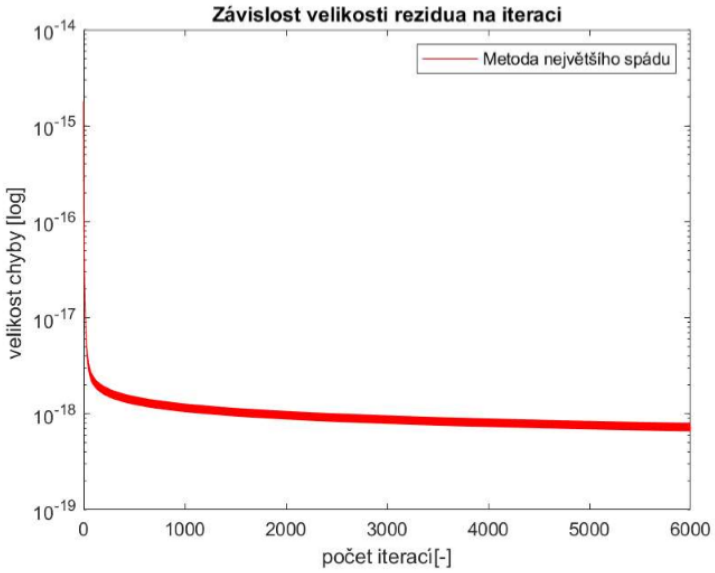
Velikost matice $\boldsymbol{A}$ zvolíme $n = 5000$. V Obrázku 2 najdeme průběhy konvergencí tří klasických iteračních metod a to Jacobiovy, Gaussovy-Seidelovy a super relaxační.

Obrázek 2- Graf průběhu rezidua v závislosti na iteraci pro Jacobiovu, Gaussovu-Seidelovu a SOR metodu a tří diagonální matici.



Na Obrázku 2 je vidět, že nejrychleji konverguje super relaxační metoda. Dále si ukážeme výsledky pro moderní iterační metody.

Obrázek 3- Graf průběhu rezidua v závislosti na iteraci pro metodu největšího spádu a tří diagonální matici.



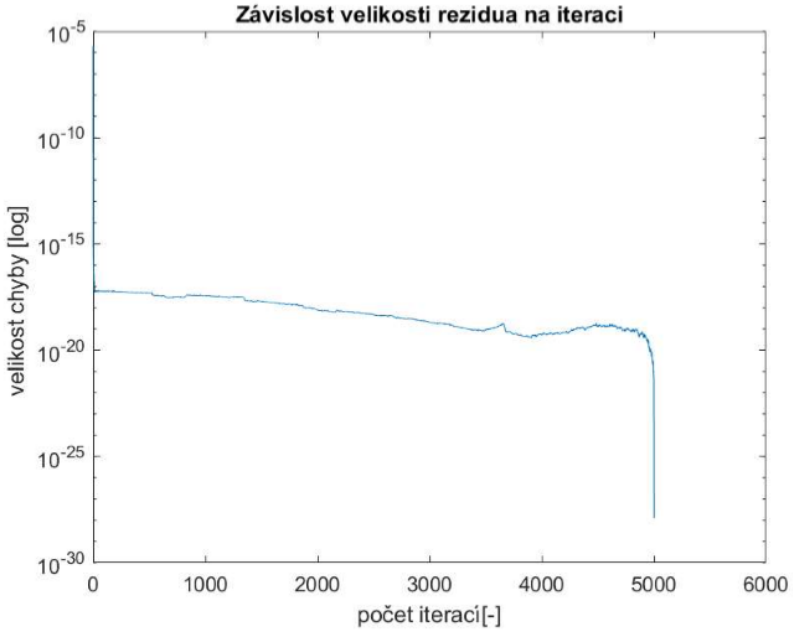
Konvergenci metody největšího spádu najdeme na Obrázku 3. Navážeme metodou sdružených gradientů, která měla z realizovaných metod nejrychlejší konvergenci.

Tabulka 1-Tři diagonální matice n=5000

Počet iterací	Reziduum Sdružené gradienty
100	5.918436895e-18
1000	3.623245233e-18
2000	7.952872399e-19
3000	2.114029169e-19
4000	5.298238954e-20
5000	2.878081457e-22
5001	2.079833328e-25
5002	1.245853479e-28

V Tabulce 1 je vidět, že jakmile dosáhneme 5000 iterace, přesnost se velice zvýší, a výsledek je možné považovat za přesný, což je v souladu s teorií uvedenou v kapitole 6.2. Pro názornější ukázku si zde uvedeme graf průběhu rezidua, který najdeme na Obrázku 4. Je zde vidět průběh závislosti rezidua na počtu iterace pro metodu sdružených gradientů a v průběhu nastává zlom po dosažení 5000 iterace.

Obrázek 4- Graf průběhu rezidua v závislosti na iteraci pro metodu sdružených gradientů a tří diagonální matici.



Dále budeme porovnávat metody na blokové matici $n \times n$, která má bloky velikosti $n \times n$ a vznikne metodou konečných prvků. Velikost matice tedy bude n^2. Budeme porovnávat metody jako v předchozím případě. Matice $\boldsymbol{A}$ vypadá

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{B} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{B} & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \ddots & \ddots & \ddots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \ddots & \boldsymbol{B} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{B} & \boldsymbol{C} \end{pmatrix},$$

kde matice $\boldsymbol{C}$ značí tři diagonální matici s 4 na diagonále, -1 pod a nad hlavní diagonálou a vypadá

$$\boldsymbol{C} = \begin{pmatrix} 4 & -1 & 0 & 0 \\ -1 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}.$$

Matice $\boldsymbol{B}$ je s -1 na diagonále. Nejprve budeme volit rozměr bloku matice $n = 200$, výsledná matice bude mít tedy rozměr $n = 40\,000$. V Tabulce 2 najdeme výsledky pro Jacobiovu metodu a Gaussovu-Seidelovu metodu.

Tabulka 2

počet iterací	Reziduum Jacobi	Reziduum Gauss-Seidl
1000	2.499931439e-04	4.970348063e-04
2000	2.485151114e-04	3.758436909e-04
4000	1.879023777e-04	1.465217355e-04
6000	9.340297024e-05	3.385619006e-05
8000	3.521865467e-05	4.795890985e-06
10000	1.038236199e-05	4.168780366e-07

Dále si porovnáme metodu horní relaxace, Gaussovu-Seidelovu metodu, metodu největšího spádu a metodu sdružených gradientů. Zde bylo zvoleno $n = 500$, pro zvýšení rozdílů mezi metodami. Jelikož metoda sdružených gradientů má daleko lepší konvergenci než zbylé tři, pro přehlednost výsledky rozdělíme do dvou tabulek.

Tabulka 3

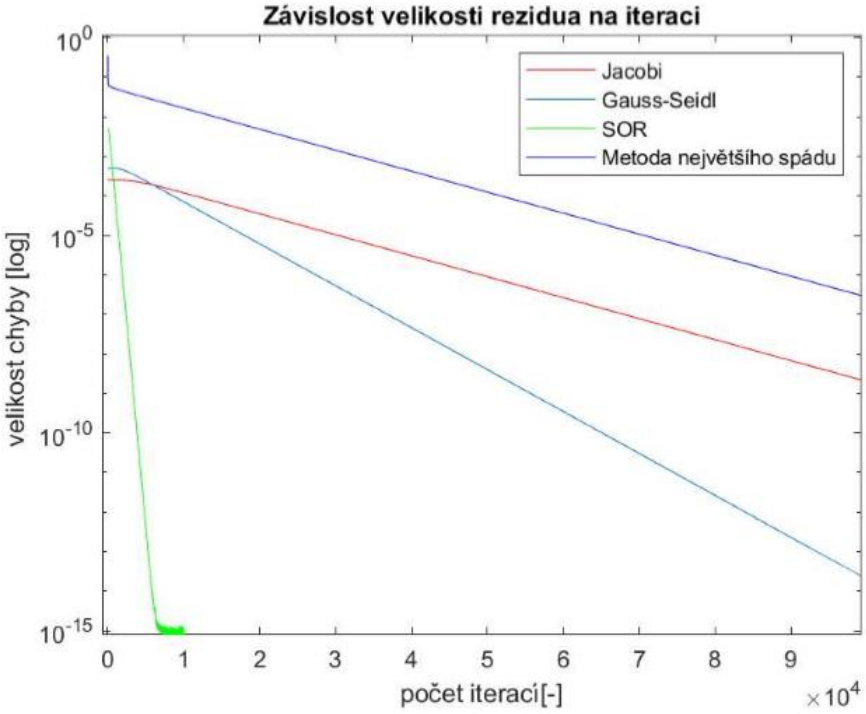
počet iterací	Reziduum Gauss-Seidel	Reziduum SOR	Reziduum Největší spád
1000	8.000000000e-05	4.863162162e-04	2.465628903e-02
2000	7.999845930e-05	9.462812682e-06	2.366275348e-02
4000	7.912145139e-05	3.510134849e-09	2.213958550e-02
6000	7.128615422e-05	8.393286066e-14	2.092773865e-02
8000	5.541140273e-05	2.442490654e-15	1.989085029e-02
10000	3.813520290e-05	1.221245327e-15	1.896987945e-02

Tabulka 4

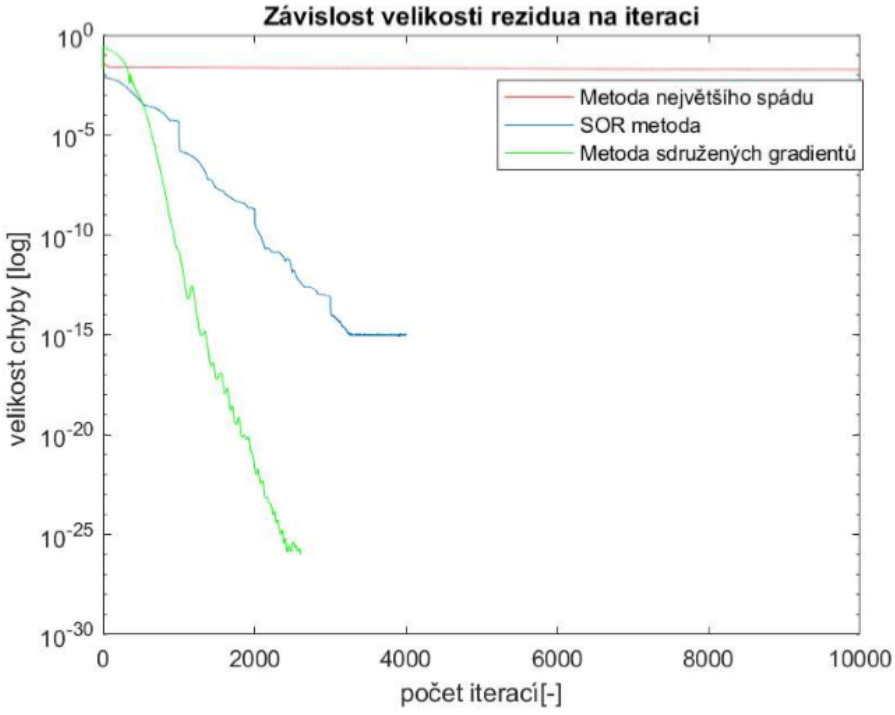
Počet iterací	Reziduum Sdružené gradienty
600	4.947208986e-05
800	2.892852385e-08
1000	1.578409574e-11
6000	9.797589837e-27

Porovnání konvergencí lépe uvidíme na Obrázku 5 a Obrázku 6. Na Obrázku 5 má super relaxační metoda nejrychlejší konvergenci mezi klasickými metodami, proto již na dalším obrázku porovnáváme pouze metodu sdružených gradientů a super relaxační metodu. Metoda největšího spádu je zde uvedena jen okrajově pro úplnost realizovaných metod, jelikož pro velké n je její konvergence velmi špatná, jak je vidět na Obrázku 6. Pro $n = 500$ metoda sdružených gradientů konverguje daleko rychleji než zbylé metody, a navíc dosahuje vyšší přesnosti řešení. Je zde také vidět, že čím větší volíme n rozměr matic, tím více potřebujeme iterací pro stejnou přesnost výsledku.

Obrázek 5- Graf průběhu rezidua v závislosti na iteraci pro Gaussovu-Seidelovu, Jacobiovu, SOR metodu a metodu největšího spádu a blokovou matici s blokem $n = 200$



Obrázek 6- Graf průběhu rezidua v závislosti na iteraci pro metodu sdružených gradientů, SOR a metodu největšího spádu a blokovou matici s blokem $n = 500$



Metody ještě porovnáme na obecnější rovnici. Jelikož metoda sdružených gradientů konverguje daleko rychleji než Gaussova-Seidelova a super relaxační, rozdělíme výsledky do dvou tabulek. První Tabulka 5, kde najdeme reziduum v daném počtu iterací pro Gaussovu-Seidelovu metodu a super relaxační metodu.

Tabulka 5

Počet iterací	Reziduum Gauss-Seidel	Reziduum SOR
10000	2.544300773e-06	1.178522350e-08
20000	1.404157371e-08	1.866151005e-13
30000	7.749312173e-11	1.110223025e-15
50000	2.335588320e-15	7.077671782e-16

A druhá Tabulka 6 kde najdeme metodu sdružených gradientů.

Tabulka 6

Počet iterací	Reziduum Sdružené gradienty
210	2.909109520e-15
300	6.672794090e-20

Srovnání z časového hlediska, které jsme testovali na blokové matici, kde jsme zvolili $n = 200$, nalezneme v Tabulce 7. Zde jsme porovnávali metodu sdružených gradientů, Gaussovu-Seidelovu metodu a SOR metodu.

Tabulka 7

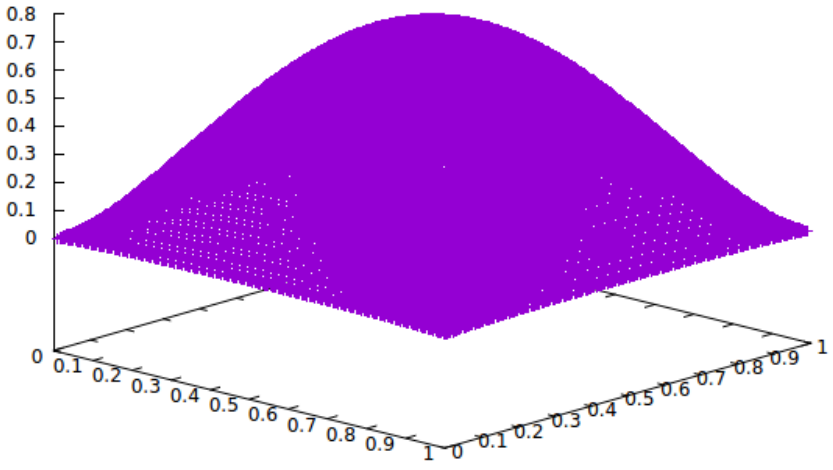
Počet iterací	Sdružené gradienty	Gauss-Seidl	SOR
1000	[0:02.20s]	[0:01.69s]	[0:01.94s]
100	[0:00.20s]	[0:00.21s]	[0:00.21s]
50	[0:00.12s]	[0:00.16s]	[0:00.15s]

Je vidět, že metoda sdružených gradientů, je nejpomalejší. Je to zapříčiněno násobením vektoru a matice v každém kroku iterace. Přesto je tato metoda nejvýhodnější.

Zde si uvedeme graficky, jak vypadali výsledky pro použité matice. Na Obrázku 7 najdeme výsledek pro blokovou matici, která má pravou stranu danou předpisem

$$b_i = \frac{10}{n^2}.$$

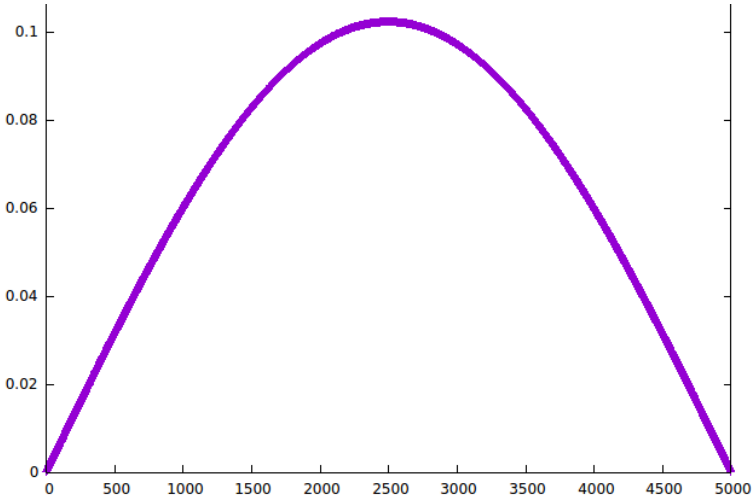
Obrázek 7-Bloková matice



Na Obrázku 8 je vidět, jak vypadá výsledek pro tři diagonální matici, kde vektor $\boldsymbol{b}$ pravé strany je zvolen dle předpisu

$$b_i = \frac{\sin\left(\pi \cdot \frac{i}{n+1}\right)}{(n+1)^2}.$$

Obrázek 8- Tři diagonální matice $n = 5000$



8. Závěr

Cílem práce bylo seznámení s metodami řešení soustav lineárních rovnic, které vznikají diskretizací metodou konečných prvků. Nejprve jsme si ukázali nutný teoretický základ. Pro jaké matice $\boldsymbol{A}$ se dají metody použít, aby konvergovali. Poté jsme se seznámili s přímými metodami, které se ale v praxi moc nevyužívají. Pro velké n matic jsou značně náročnější než iterační metody. Ty jsme si ukázali jak klasické, tak moderní.

Některé vybrané iterační metody jsme realizovali. Uvedli jsme, v jakém formátu je výhodné uložit řídké matice a poté jsme s nimi dále pracovali. Porovnávali jsme vybrané metody zejména z hlediska velikosti rezidua. Nejlepší konvergenci má z vybraných metod jednoznačně metoda sdružených gradientů, a navíc bylo ukázáno, že opravdu nejdéle po n krocích dosáhne téměř přesného výsledku. I přes výpočetní čas, který má metoda sdružených gradientů nejdelší, je tato metoda nejvýhodnější.

1 SEZNAM ZKRATEK A SYMBOLŮ

ISO 9001	*Certifikát prokazující, že společnost má zavedený systém řízení ve shodě s požadavky této normy*
ISO 14001	*Certifikát prokazující, že systém řízení společnosti v oblasti životního prostředí je ve shodě s požadavky této normy*
OHSAS 18001	*Certifikát potvrzující, že systém řízení společnosti v oblasti bezpečnosti práce je ve shodě s požadavky této normy*

2 Seznam použité literatury

[1] SVÁČEK, Petr a Miloslav FEISTAUER. *Metoda konečných prvků*. Praha: Nakladatelství ČVUT, 2006. ISBN 80-01-03522-0.

[2] BRDIČKA, Miroslav, Ladislav SAMEK a Bruno SOPKO. *Mechanika kontinua*. Vyd. 4., rev. a upr. Praha: Academia, 2011. Gerstner. ISBN 978-80-200-2039-0.

[3] *Řešení soustav Lineárních rovnic* [online]. [cit. 2019-08-02]. Dostupné z: https://zolotarev.fd.cvut.cz/static/mag/mag-2013-11-handouts.pdf

[4] QUARTERONI, Alfio a Alberto VALLI. *Numerical Approximation of Partial Differential Equations*. Berlin: Springer, 1997. ISBN 3-540-57111-6.

[5] *Vybrané partie z Numerické matematiky* [online]. [cit. 2019-08-02]. Dostupné z: 3 http://physics.ujep.cz/~mlisal/nm_1-chomutov/jkrejci/texty/Tema2.pdf

[6] [online]. [cit. 2019-08-02]. Dostupné z: https://www.math.muni.cz/~kolacek/vyuka/nummet/prime_metody.pdf

[7] *Numerická matematika,poznámky k přednáškám* [online]. [cit. 2019-08-02]. Dostupné z: http://marian.fsik.cvut.cz/~svacek/numericka_matematika/pdf/lecture_notes.pdf

[8] BENDA, Josef a Růžena ČERNÁ. *Numerická matematika: doplňkové skriptum*. Vyd. 3. V Praze: České vysoké učení technické, 2008. ISBN 978-80-01-04037-9.

[9] *Netlib* [online]. [cit. 2019-08-02]. Dostupné z: http://www.netlib.org/linalg/html_templates/node17.html

[10] Metoda největšího spádu. [online]. [cit. 2019-08-02]. Dostupné z: http://marta.certik.cz/NM/mns.pdf

[11] KUČERA, Luděk. *Metoda sdružených gradientů* [online]. [cit. 2019-08-02]. Dostupné z: https://kam.mff.cuni.cz/~ludek/LA/CG.pdf?fbclid=IwAR26YzcUeWNvy7rWtPX13TvMNAx5ry1tDTrpq4Kw0ENfUpTzi3wGtcZkQjQ

[12] KŘÍŽEK, Michal a Jan BRANDTS. Pokroky matematiky,fyziky a astronomie[online]. [cit. 2019-08-02]. Dostupné z: https://dml.cz/bitstream/handle/10338.dmlcz/141120/PokrokyMFA_47-2002-2_2.pdf

[13] HEGGELIEN, Runar. *A brief introduction to the conjugate gradient method* [online]. [cit. 2019-08-02]. Dostupné z: https://folk.idi.ntnu.no/elster/tdt24/tdt24-f09/cg.pdf

[14] NEUSTUPA, J. *Matematika I.Skriptum FS CVUT*. Praha: ČVUT, 2005. ISBN 978-80-01-05328-7.

3 Seznam obrázků

Obrázek 1 -Průběh $\varphi_i(x)$ a $\varphi_{i+1}(x)$ 6
Obrázek 2- Graf průběhu rezidua v závislosti na iteraci pro Jacobiovu, Gaussovu-Seidelovu a SOR metodu a tří diagonální matici. 32
Obrázek 3- Graf průběhu rezidua v závislosti na iteraci pro metodu největšího spádu a tří diagonální matici. 33
Obrázek 4- Graf průběhu rezidua v závislosti na iteraci pro metodu sdružených gradientů a tří diagonální matici. 34
Obrázek 5- Graf průběhu rezidua v závislosti na iteraci pro Gaussovu-Seidelovu, Jacobiovu, SOR metodu a metodu největšího spádu a blokovou matici s blokem $n = 200$ 36
Obrázek 6- Graf průběhu rezidua v závislosti na iteraci pro metodu sdružených gradientů, SOR a metodu největšího spádu a blokovou matici s blokem $n = 500$ 36
Obrázek 7-Bloková matice 38
Obrázek 8- Tři diagonální matice $n = 5000$ 38

4 Seznam tabulek

Tabulka 1 33
Tabulka 2 35
Tabulka 3 35
Tabulka 4 35
Tabulka 5 37
Tabulka 6 37
Tabulka 7 37

5 Seznam příloh

Příloha I. CD s zip souborem obsahující program