UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

Zobecněná metoda minimálních reziduí

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **doc. RNDr. Jitka Machalová, Ph.D.**
Vypracovala: **Kristina Pimerová**
Studijní program: B1103 Aplikovaná matematika
Studijní obor: Matematika–ekonomie se zaměřením na bankovnictví/pojišťovnictví
Forma studia: prezenční
Rok odevzdání: 2023

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Kristina Pimerová

Název práce: Zobecněná metoda minimálních reziduí

Typ práce: Bakalářská práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: doc. RNDr. Jitka Machalová, Ph.D.

Rok obhajoby práce: 2023

Abstrakt: Zobecněná metoda minimálních reziduí je iterační metoda pro výpočet řešení rozsáhlých soustav lineárních rovnic. Cílem práce je tuto metodu představit a seznámit se s jejími dílčími částmi, algoritmem a hlavními variacemi. Práce je doplněna vlastními zdrojovými kódy sepsanými v matematickém softwaru Matlab a praktickými příklady, kterým je věnována poslední kapitola.

Klíčová slova: Zobecněná metoda minimálních reziduí, GMRES, Arnoldiho algoritmus, QR rozklad, soustavy lineárních rovnic, software Matlab

Počet stran: 70

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Kristina Pimerová

Title: Minimal residual methods

Type of thesis: Bachelor's

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: doc. RNDr. Jitka Machalová, Ph.D.

The year of presentation: 2023

Abstract: The generalized minimal residual method is an iterative method for solving large systems of linear equations. The aim of the thesis is to present this method and to get acquainted with its sub-parts, algorithm and main variations. The work is supplemented by codes written in the mathematical software Matlab and practical examples, which the last chapter is devoted to.

Key words: generalized minimal residual method, GMRES, Arnoldi algorithm, QR decomposition, systems of linear equations, software Matlab

Number of pages: 70

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením paní doc. RNDr. Jitky Machalové, Ph.D. a všechny použité zdroje jsem uvedla v seznamu literatury.

V Olomouci dne ....

podpis

Poděkování

Ráda bych poděkovala vedoucí mé bakalářské práce, paní doc. RNDr. Jitce Machalové, Ph.D., za cenné rady, trpělivost a čas, který mi na konzultacích věnovala.

Obsah

Použité symboly

$\mathbb{N}$	obor přirozených čísel
$\mathbb{R}$	obor reálných čísel
$\mathbb{R}^n$	vektorový prostor dimenze n nad $\mathbb{R}$
$\mathbb{M}^n$	třída všech reálných čtvercových matic řádu n
$\mathbb{M}^{m\times n}$	třída všech reálných matic typu $m \times n$
$\mathcal{A}$	vektorový prostor
$\mathbf{x} = (x_1, \ldots, x_n)^T$	uspořádaná n-tice reálných čísel – vektor
$\mathbf{A} = (a_{ij})_{i,j=1}^n$	reálná čtvercová matice řádu n
$\mathbf{o}$	nulový vektor
$\mathbf{O}$	nulová matice
$\mathbf{e}$	první sloupec jednotkové matice
$\mathbf{I}$	jednotková matice
$\mathbf{A}^{-1}$	inverzní matice k matici $\mathbf{A}$
$\mathbf{x}^T$	transponovaný vektor
$\mathbf{A}^T$	transponovaná matice
$\|\mathbf{x}\|$	norma vektoru
$\|\mathbf{A}\|$	norma matice
$\langle \mathbf{x}, \mathbf{y} \rangle$	skalární součin vektorů
$\{\mathbf{x}^j\}$	posloupnost vektorů
$\|\mathbf{A}\|$	determinant matice

Úvod

Iterační metody představují obsáhlou kapitolu numerické matematiky. Se základními metodami pro řešení systémů lineárních rovnic, konkrétně s Jacobiovou a Gauss-Seidlovou iterační metodou, jsem se seznámila při svém dosavadním vysokoškolském studiu. Řešení rovnic a soustav rovnic mě okouzlilo již na střední škole, a toto nadšení přetrvává dodnes. Z tohoto důvodu jsem si jako téma své bakalářské práce vybrala Zobecněnou metodu minimálních reziduí.

Podstata využití metody spočívá v možnosti nalezení přibližného řešení rozsáhlých soustav lineárních rovnic. Tímto se v roce 1986, kdy byla metoda vytvořena Yousefem Saadem a Martinem H. Schultzem, rozšířily možnosti pro řešení soustav rovnic, které do té doby byly dosavadními metodami neřešitelné.

První kapitola této bakalářské práce je věnována nadefinování základních pojmů, především z oblasti lineární algebry, bez kterých bychom se v dalších částech práce neobešli. Samotná metoda je představena v následující kapitole, kde jsou popsány její jednotlivé části. Kapitola je doplněna příslušnými naprogramovanými kódy a názornými příklady jejich využití. Další kapitola je věnována několika nejznámějším variacím Zobecněné metody minimálních reziduí. Poslední kapitola obsahuje příklady řešené Zobecněnou metodou minimálních reziduí pomocí vlastních zdrojových kódů vytvořených v matematickém softwaru Matlab. V této části práce jsou také srovnány naše kódy s matlabovskou funkcí Zobecněné metody minimálních reziduí z hlediska výsledků a výpočetní rychlosti.

Cílem této bakalářské práce je tedy seznámit čtenáře se Zobecněnou metodou minimálních reziduí, jejími variacemi a ukázat použití metody na příkladech řešených v matematickém softwaru Matlab.

1. Přípravná kapitola

Abychom se mohli zabývat Zobecněnou metodou minimálních reziduí jakožto iterační metodou pro výpočet řešení soustavy lineárních rovnic, budeme si muset nejprve uvést některé základní pojmy, bez jejichž předchozí znalosti bychom se neobešli. Při formulaci následujících definic a vět budeme vycházet z [4] a [5]. Definici semi-ortogonální matice lze dohledat v [1]. Poslední definice je převzata z [6].

Definice 1.1. Vektory $\mathbf{x}_1, \ldots, \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, n$, se nazývají lineárně závislé, existuje-li alespoň jedna jejich netriviální nulová kombinace. V opačném případě se vektory $\mathbf{x}_1, \ldots, \mathbf{x}_n$ nazývají lineárně nezávislé.

Definice 1.2. Nechť je dána množina vektorů $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, n$. Pak lineárním obalem množiny $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ rozumíme množinu všech lineárních kombinací vektorů $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Značíme $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$.

Definice 1.3. Nechť prostor $\mathcal{A}$ je lineárním obalem množiny vektorů $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, n$, tedy $\mathcal{A} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Pak říkáme, že množina $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ konečně generuje prostor $\mathcal{A}$, a prvky této množiny jsou generátory prostoru $\mathcal{A}$.

Definice 1.4. Skalární součin je operace $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ s následujícími vlastnostmi

1. $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n$, přičemž $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{o}$,
2. $\langle c\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, c\mathbf{y} \rangle = c\langle \mathbf{x}, \mathbf{y} \rangle \quad \forall c \in \mathbb{R}, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,
3. $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,
4. $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$, $\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$.

Poznámka 1.1. V prostoru $\mathbb{R}^n$ je skalární součin $\langle \mathbf{x}, \mathbf{y} \rangle$ dán následovně

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Definice 1.5. Vektorová norma na $\mathbb{R}^n$ je funkce $\|.\| : \mathbb{R}^n \to \mathbb{R}$ s následujícími vlastnostmi

1. $\|\mathbf{x}\| \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n$, přičemž $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{o}$ *(pozitivně definitní)*,
2. $\|c\,\mathbf{x}\| = |c|\|\mathbf{x}\| \quad \forall c \in \mathbb{R}, \forall \mathbf{x} \in \mathbb{R}^n$ *(pozitivně homogenní)*,
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *(trojúhelníková nerovnost)*.

Poznámka 1.2. V následujících kapitolách budeme uvažovat Euklidovskou normu vektoru $\mathbf{x}$ danou vztahem

$$\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2},$$

která udává vzdálenost bodu $\mathbf{x}$ od počátku soustavy souřadnic (důsledek Pythagorovy věty). Dále ji budeme označovat $\|\mathbf{x}\|$.

Příklad 1.1. Vypočtěte Euklidovskou normu vektoru $\mathbf{x} = (-3, 1, 2)^T$.
Řešení:

$$\|\mathbf{x}\| = \sqrt{(-3)^2 + (1)^2 + (2)^2} = \sqrt{9 + 1 + 4} = \sqrt{14}.$$

Definice 1.6. Vektory $\mathbf{x}$, $\mathbf{y}$ nazýváme ortogonální, jestliže $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Mají-li $\mathbf{x}$ a $\mathbf{y}$ navíc jednotkovou normu, jsou ortonormální.

Definice 1.7. Regulární matice je taková matice $\mathbf{A} \in \mathbb{M}^n$, pro kterou platí $|\mathbf{A}| \neq 0$. V opačném případě nazýváme matici $\mathbf{A}$ singulární.

Definice 1.8. Hodnost matice $\mathbf{A} \in \mathbb{M}^{m \times n}$ je číslo $h(\mathbf{A}) \in \mathbb{R}$, které je rovno maximálnímu počtu lineárně nezávislých řádků nebo sloupců matice $\mathbf{A}$.

Definice 1.9. Nechť $\mathbf{A} \in \mathbb{M}^n$. Řekneme, že matice $\mathbf{A}$ je symetrická, jestliže $\mathbf{A}^T = \mathbf{A}$.

Definice 1.10. Řekneme, že symetrická matice $\mathbf{A} \in \mathbb{M}^n$ je pozitivně definitní, jestliže pro každý vektor $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \neq \mathbf{o}$, je $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$.

Věta 1.1 (Sylvestrovo kritérium)**.** *Nechť je dána matice* $\mathbf{A} \in \mathbb{M}^n$ *a její submatice jsou tvaru*

$$\mathbf{A}_k = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix}, \quad \forall k = 1, \dots, n.$$

Pak matice $\mathbf{A}$ *je pozitivně definitní právě tehdy, když*

$$|\mathbf{A}_k| > 0, \quad \forall k = 1, \dots, n.$$

Důkaz: viz [4, str. 168-169]. □

Definice 1.11. Nechť $\mathbf{A} \in \mathbb{M}^n$. Řekneme, že matice $\mathbf{A}$ je ortogonální, jestliže $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$.

Věta 1.2. *Je-li* $\mathbf{A} \in \mathbb{M}^n$ *ortogonální, pak k ní existuje inverzní matice a platí* $\mathbf{A}^{-1} = \mathbf{A}^T$.

Důkaz: viz [4, str. 123]. □

Věta 1.3. *Jsou-li* $\mathbf{A}, \mathbf{B} \in \mathbb{M}^n$ *ortogonální, pak i* $\mathbf{AB}$ *je ortogonální.*

Důkaz: viz [4, str. 123]. □

Definice 1.12. Nechť $\mathbf{A} \in \mathbb{M}^{m\times n}$. Řekneme, že matice $\mathbf{A}$ je semi-ortogonální, jestliže $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, $\mathbf{I} \in \mathbb{M}^m$ nebo $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, $\mathbf{I} \in \mathbb{M}^n$, přičemž obě rovnosti nutně platit nemusí.

Poznámka 1.3. Každá ortogonální matice $\mathbf{A} \in \mathbb{M}^n$ je i semi-ortogonální.

Věta 1.4. *Nechť* $\mathbf{A} \in \mathbb{M}^{m\times n}$ *je semi-ortogonální matice. Platí-li* $\mathbf{A}^T\mathbf{A} = \mathbf{I}$*, pak*

1. $\langle \mathbf{A}\mathbf{x}, \mathbf{A}\mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,
2. $\|\mathbf{A}\mathbf{x}\| = \|\mathbf{x}\| \quad \forall \mathbf{x} \in \mathbb{R}^n$.

Platí-li $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, *pak*

3. $\langle \mathbf{A}^T\mathbf{x}, \mathbf{A}^T\mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m$,

4. $\|\mathbf{A}^T\mathbf{x}\| = \|\mathbf{x}\| \quad \forall \mathbf{x} \in \mathbb{R}^m$.

Důkaz: Vlastnost 1. pro skalární součin $\mathbf{Ax}$ a $\mathbf{Ay}$ dokážeme s využitím předpokladu $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ a poznámky 1.1 následovně

$$\langle \mathbf{Ax}, \mathbf{Ay} \rangle = (\mathbf{Ax})^T\mathbf{Ay} = \mathbf{x}^T\mathbf{A}^T\mathbf{Ay} = \mathbf{x}^T\mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle.$$

Důkaz vlastnosti 2. pro normu $\mathbf{Ax}$ vychází z vlastnosti 1. a poznámky 1.2. Zřejmě platí

$$\|\mathbf{Ax}\| = \sqrt{\langle \mathbf{Ax}, \mathbf{Ax} \rangle} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \|\mathbf{x}\|.$$

Důkazy vlastností 3. a 4. bychom provedli obdobně. □

Poznámka 1.4. Pro ortogonální matici $\mathbf{A} \in \mathbb{M}^n$ platí všechny čtyři vlastnosti uvedené ve větě 1.4 pro $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Definice 1.13. Maticová norma na $\mathbb{M}^n$ je funkce $\|.\| : \mathbb{M}^n \to \mathbb{R}$ s následujícími vlastnostmi

1. $\|\mathbf{A}\| \geq 0 \quad \forall \mathbf{A} \in \mathbb{M}^n$, přičemž $\|\mathbf{A}\| = 0 \Leftrightarrow \mathbf{A} = \mathbf{O}$,

2. $\|c\,\mathbf{A}\| = |c|\|\mathbf{A}\| \quad \forall c \in \mathbb{R}, \forall \mathbf{A} \in \mathbb{M}^n$,

3. $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\| \quad \forall \mathbf{A}, \mathbf{B} \in \mathbb{M}^n$,

4. $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\| \quad \forall \mathbf{A}, \mathbf{B} \in \mathbb{M}^n$.

Definice 1.14. Nechť $\mathbf{A} \in \mathbb{M}^n$ je regulární. Pak číslo $k(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, $k(\mathbf{A}) \geq 1$, nazýváme číslem podmíněnosti matice $\mathbf{A}$.

Poznámka 1.5. Je-li $k(\mathbf{A}) \approx 1$ je matice, příp. úloha, která obsahuje výpočet s maticí $\mathbf{A}$, dobře podmíněna. Je li $k(\mathbf{A}) \gg 1$ je matice, příp. úloha, špatně podmíněna.

Definice 1.15. Soustavou n lineárních rovnic o n neznámých rozumíme systém lineárních rovnic ve tvaru

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n, \end{aligned} \tag{1.1}$$

přičemž reálná čísla a_{ij}, $i, j \in \{1, 2, \ldots, n\}$, nazýváme koeficienty levé strany soustavy rovnic, reálná čísla b_i, $i \in \{1, 2, \ldots, n\}$ nazýváme koeficienty pravé strany soustavy rovnic a proměnné $x_1, x_2, \ldots, x_n$ nazýváme neznámé.

Poznámka 1.6. Soustavu (1.1) lze ekvivalentně přepsat do maticového tvaru

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

neboli zkráceně $\mathbf{Ax} = \mathbf{b}$. Je-li matice $\mathbf{A} = (a_{ij})_{i,j=1}^{n}$ regulární, pak má soustava (1.1) jediné řešení $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.

Věta 1.5 (Frobeniova)**.** *Soustava* $\mathbf{Ax} = \mathbf{b}$ *má (alespoň jedno) řešení právě tehdy, když* $h(\mathbf{A}|\mathbf{b}) = h(\mathbf{A})$.

Důkaz: viz [8, str. 51]. □

Definice 1.16. Matice $\mathbf{A} = (a_{ij})$, $\mathbf{A} \in \mathbb{M}^{m\times n}$ je horní trojúhelníková, pokud $a_{ij} = 0$ pro všechna $i > j$, kde $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Poznámka 1.7. Horní trojúhelníkové matice mohou vypadat například následovně

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \end{pmatrix}, \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}, \begin{pmatrix} a_{11} \end{pmatrix}.$$

Věta 1.6. *Inverzní maticí k horní trojúhelníkové matici* $\mathbf{A} \in \mathbb{M}^n$ *je také horní trojúhelníková matice.*

Důkaz: viz [1, str. 186-187]. □

Definice 1.17. Řekneme, že matice $\mathbf{A} = (a_{ij})$, $\mathbf{A} \in \mathbb{M}^{m\times n}$ je v horním Hessenbergově tvaru, jestliže platí $a_{ij} = 0$ pro $i > j+1$, kde $i = 1, \ldots, m$, $j = 1, \ldots, n$.

2. Zobecněná metoda minimálních reziduí

2.1. Krylovovy metody

Zobecněná metoda minimálních reziduí je Krylovovou iterační metodou pro řešení rozsáhlých soustav lineárních rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.1}$$

kde matice $\mathbf{A} \in \mathbb{M}^n$ je regulární.

Krylovovy metody spadají do iteračních metod, které na rozdíl od metod přímých fungují na principu nalezení přibližného řešení s danou přesností nebo případně za maximální počet kroků k, tzv. iterací. Nejprve je nutné zvolit počáteční aproximaci, vektor $\mathbf{x}^0$. Opakovaným průběhem iteračního algoritmu je pak postupně konstruována posloupnost vektorů $\mathbf{x}^0 \to \mathbf{x}^1 \to \mathbf{x}^2 \dots$ (zkráceně $\{\mathbf{x}^j\}$), která v ideálním případě konverguje k přesnému řešení $\mathbf{x}^*$. Zastavovacím kritériem iteračního algoritmu tak může být např. $\|\mathbf{r}^j\| < \epsilon$, $\mathbf{r}^j = \mathbf{b} - \mathbf{A}\mathbf{x}^j$, $j = 1, 2, \dots$, kde velikost přípustné chyby ϵ je předem dána.

Při formulaci následujících definic budeme vycházet z [6] a [9].

Definice 2.1. Nechť je dána regulární matice $\mathbf{A} \in \mathbb{M}^n$ a vektor $\mathbf{r}^0 \in \mathbb{R}^n$. Krylovův prostor (někdy také podprostor) dimenze k je lineárním obalem posloupnosti vektorů $\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \mathbf{A}^2\mathbf{r}^0, \dots, \mathbf{A}^{k-1}\mathbf{r}^0$, tj.

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0) = [\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \mathbf{A}^2\mathbf{r}^0, \dots, \mathbf{A}^{k-1}\mathbf{r}^0].$$

Značíme jej zkráceně $\mathcal{K}_k$.

V další části této práce položíme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{r}^0 \neq \mathbf{o}$, což je tzv. reziduum počáteční iterace. Vektor $\mathbf{x}^0$ je zvolená počáteční aproximace řešení $\mathbf{x}^*$. Při $\mathbf{r}^0 = \mathbf{o}$ by bylo $\mathbf{x}^0 = \mathbf{x}^*$ a nebylo by dále nutné řešení soustavy hledat Zobecněnou metodou minimálních reziduí.

Poznámka 2.1. Horní index u matice $\mathbf{A}$ je mocnina matice, zatímco indexy u vektorů $\mathbf{r}$ a $\mathbf{x}$ značí číslo iterace.

Podstatou Krylovových metod pro řešení soustav lineárních rovnic je využití vektorů z Krylovova prostoru při výpočtu řešení úlohy (2.1). Vektory $\mathbf{x}^j$, $j = 1, \ldots, k$, získané v jednotlivých iteracích, leží tedy v prostoru, který je generovaný vektorem $\mathbf{x}^0$ a vektory z prostoru $\mathcal{K}_j$. Postupně je tak generována posloupnost vektorů $\{\mathbf{x}^j\}$, kde

$$\mathbf{x}^j \in \mathbf{x}^0 + \mathcal{K}_j, \tag{2.2}$$

které aproximují řešení $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$. Dle definice 1.2 platí následující rovnost

$$\mathcal{K}_k = [\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \mathbf{A}^2\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0] = \tag{2.3}$$

$$= \{c_1\mathbf{r}^0 + c_2\mathbf{A}\mathbf{r}^0 + \cdots + c_k\mathbf{A}^{k-1}\mathbf{r}^0 | c_1, \ldots, c_k \in \mathbb{R}\}.$$

Prostor $\mathcal{K}_k$ je dle definice 1.3 generovaný konečnou množinou vektorů $\{\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \mathbf{A}^2\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0\}$. Libovolný vektor $\mathbf{x}^j$, $j = 1, \ldots, k$ z (2.2) můžeme s využitím vztahu (2.3) zapsat ekvivalentně ve tvaru

$$\mathbf{x}^j = \mathbf{x}^0 + \mathbf{K}_j\mathbf{c}, \quad \mathbf{c} \in \mathbb{R}^j, \tag{2.4}$$

kde matice $\mathbf{K}_j$ je tzv. Krylovova matice.

Definice 2.2. Nechť je dána regulární matice $\mathbf{A} \in \mathbb{M}^n$ a vektor $\mathbf{r}^0 \in \mathbb{R}^n$. Krylovova matice generovaná vektorem $\mathbf{r}^0$ je matice $\mathbf{K}_k(\mathbf{r}^0, \mathbf{A}) \in \mathbb{M}^{n\times k}$ jejíž sloupce tvoří vektory $\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \mathbf{A}^2\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0$. Značíme ji zkráceně $\mathbf{K}_k$.

Prostor $\mathcal{K}_k$ tak při výpočtech můžeme nahradit součinem matice $\mathbf{K}_k$ a libovolného vektoru $\mathbf{c} = (c_1, \ldots, c_k)^T$, $\mathbf{c} \in \mathbb{R}^k$.

Z definice 2.2 ovšem není zaručena lineární nezávislost jednotlivých sloupců matice $\mathbf{K}_k$. Mohlo by se tedy stát, že matice $\mathbf{K}_k$ bude singulární. S takovou maticí bychom si při dalších výpočtech Zobecněnou metodou minimálních reziduí neporadili.

Nabízí se ortogonalizace vektorů tvořících sloupce matice $\mathbf{K}_k$. Ovšem tím, že by sloupce matice $\mathbf{K}_k$ mohly být lineárně závislé, nejsme již schopni jimi popsat prostor $\mathcal{K}_k$ a pouhou ortogonalizací bychom ztracené informace zpět nezískali. Namísto toho ortonormální bázi prostoru $\mathcal{K}_k$ obdržíme tzv. Arnoldiho algoritmem. Ortonormální báze bude tvořena sloupcovými vektory matice $\mathbf{V}_k \in \mathbb{M}^{n\times k}$. První sloupec matice $\mathbf{V}_k$ položíme roven $\mathbf{r}^0$. Algoritmus postupně vypočte sloupcové vektory matice $\mathbf{V}_k$ tak, že předchozí sloupcový vektor vynásobí zleva maticí $\mathbf{A}$ (podle definice 2.2) a následně jej ortogonalizuje vůči všem již vypočteným sloupcovým vektorům [11, str. 298].

Vektory $\mathbf{x}^j$ z (2.4) tak budeme za využití matice $\mathbf{V}_j$ hledat ve tvaru

$$\mathbf{x}^j = \mathbf{x}^0 + \mathbf{V}_j\mathbf{y}, \quad \mathbf{y} \in \mathbb{R}^j, \tag{2.5}$$

ze kterého budeme při dalších výpočtech vycházet.

Výhodou Krylovových metod oproti stacionárním iteračním metodám, jako je např. Jacobiova či Gauss-Seidlova metoda, je jednoduchost početních operací. Při výpočtech Krylovovými metodami totiž dochází buď k násobení matice vektorem, případně k násobení řídkých matic (tj. takových, které mají převážně nulové prvky) mezi sebou, a není tak nutné násobit velké husté matice mezi sebou.

Jejich další výhodou je zajištěná konvergence k přesnému řešení $\mathbf{x}^*$ soustavy $\mathbf{Ax} = \mathbf{b}$ s nějakou výpočetní chybou ϵ po nejvýše n iteracích, kde n je řád matice $\mathbf{A}$. V přesné aritmetice algoritmus výpočtu řešení soustavy zaručeně skončí nejvýše po n krocích nalezením $\mathbf{x}^*$. Takovou metodu nazýváme metodou finitní.

Kromě Zobecněné metody minimálních reziduí do Krylovových metod spadají mimo jiné Arnoldiho algoritmus, Lanczosova metoda či Metoda konjugovaných gradientů. V současné době jsou Krylovovy metody široce využívány jak při řešení soustav lineárních rovnic, tak při výpočtu vlastních čísel velkých řídkých matic.

2.2. Arnoldiho algoritmus

Arnoldiho metoda je jednou z dílčích částí algoritmu Zobecněné metody minimálních reziduí. Používá se k sestavení matice $\mathbf{V}_k$, jejíž sloupce $\mathbf{v}^j \in \mathbb{R}^n$, $j = 1, \ldots, k$, tvoří ortonormální bázi Krylovova podprostoru $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$, $1 \leq k \leq n$, kde $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{r}^0 \neq \mathbf{o}$. V této části práce budeme vycházet z poznatků z [2] a [10].

Na vstupu je dána matice $\mathbf{A} \in \mathbb{M}^n$ a vektor $\mathbf{b} \in \mathbb{R}^n$. Na výstupu je semi-ortogonální matice $\mathbf{V}_{k+1} \in \mathbb{M}^{n\times(k+1)}$ a matice $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$ v horním Hessenbergově tvaru, případně matice $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ a $\overline{\mathbf{H}}_j = (h_{ij})$, $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1)\times j}$, $j = 1, \ldots, k$ při předčasném ukončení výpočtu z důvodu rovnosti $h_{j+1,j} = 0$, tzv. breakdown, kterému se budeme věnovat v následující kapitole této práce.

Dále je možné na vstupu algoritmu zvolit maximální počet iterací $k \in \mathbb{N}$, tedy požadovanou dimenzi prostoru $\mathcal{K}_k$, kde $1 \leq k \leq n$. Jedná se ovšem o nepovinný parametr, implicitně nastavíme $k = n$. Dalším nepovinným parametrem je počáteční aproximace $\mathbf{x}^0 \in \mathbb{R}^n$. Implicitně zvolíme $\mathbf{x}^0 = \mathbf{o}$. Teoreticky vypadá Arnoldiho algoritmus následovně.

Algoritmus 2.1 (Arnoldiho)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$ a k
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{w}^j = \mathbf{A}\mathbf{v}^j$
  - Pro $i = 1, \ldots, j$ položme
    - $h_{ij} = \langle \mathbf{v}^i, \mathbf{w}^j \rangle$
  - $\mathbf{w}^{j+1} = \mathbf{w}^j - \sum_{i=1}^{j} h_{ij}\mathbf{v}^i$

- $h_{j+1,j} = \|\mathbf{w}^{j+1}\|$
- Je-li $h_{j+1,j} = 0$
  - Konec výpočtu (breakdown)
  - Matice $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^j$
  - Matice $\overline{\mathbf{H}}_j = (h_{ij})$, $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1)\times j}$
- Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^{j+1}}{h_{j+1,j}}$.

- Matice $\mathbf{V}_{k+1} \in \mathbb{M}^{n\times(k+1)}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^{k+1}$
- Matice $\overline{\mathbf{H}}_k = (h_{ij})$, $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$

Náš naprogramovaný algoritmus sestavený v prostředí Matlab musíme kvůli automatickému zaokrouhlování upravit tak, že místo podmínky $h_{j+1,j} = 0$ v kódu použijeme $h_{j+1,j} < \epsilon$, kde $\epsilon \in \mathbb{R}$ je libovolně zvolené malé kladné číslo, které představuje přípustnou chybu a určuje tak přesnost výpočtu. Implicitně nastavíme $\epsilon = 10^{-6}$.

V kódu dále budeme nastavený maximální počet iterací k značit $k_{\max}$. Samotné $k \in \mathbb{N}$ pak bude na výstupu vyjadřovat skutečně provedený počet iterací. Na výstupu dále bude možné nechat si vypsat vektor počátečního rezidua $\mathbf{r}^0 \in \mathbb{R}^n$ a pomocnou proměnnou $p \in \{0, 1, 2\}$, kterou využijeme později v m-souboru 2.3. Jestliže $p = 0$, pak byl výpočet zastaven po dosažení maximálního počtu iterací, tedy $k = k_{\max}$, $k_{\max} < n$. Pro $p = 1$ platí, že zadaný vektor $\mathbf{x}^0$ řeší soustavu $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$ se zadanou přesností ϵ, tedy $\mathbf{x}^* = \mathbf{x}^0$. Poslední možná situace $p = 2$ nastane, pokud je výpočet ukončen z důvodu splnění podmínky $h_{j+1,j} < \epsilon$, což pro $k_{\max} \geq n$ v přesné aritmetice platí vždy. To je dáno tím, že se jedná právě o finitní metodu.

Poznámka 2.2. U všech naprogramovaných kódů budeme předpokládat správné zadání rozměrů vstupních údajů uživatelem podle poznámek uvedených v úvodu jednotlivých kódů.

M-soubor 2.1.

```
function [V,H,r0,k,p] = arnoldiK(A,b,kmax,x0,e)

  % A je ctvercova matice radu n
  % b je n-slozkovy sloupcovy vektor
  % kmax je maximalni pocet iteraci
  % x0 je n-slozkovy sloupcovy vektor, pocatecni aproximace
  % e je libovolne male kladne cislo epsilon

  % V je matice s ortonormalnimi sloupcovymi vektory
  % H je matice v hornim Hessenbergove tvaru
  % r0 je pocatecni reziduum
  % k je pocet iteraci
  % p je pomocna promenna vyuzita v kodu gmresK

  p = 0;
  n = size(A,1);

  if (nargin<3)||(isempty(kmax))||(kmax <= 0)
    kmax = n; %implicitni hodnota kmax
  end

  if (nargin<4)||(isempty(x0))
    x0 = zeros(n,1); %implicitni vektor x0
  end

  if (nargin<5)||(isempty(e))
    e = 1e-6; %implicitni hodnota e
  end
```

```
kmax = min(kmax,n);

V = zeros(n,kmax);
H = zeros(kmax+1,kmax);

r0 = b - (A*x0);
if (norm(r0)<e)
  p = 1;
  k = 0;
  disp('Zadany vektor x0 resi soustavu A*x0 = b s presnosti e.')
  V = [];
  H = [];
  return
end

V(:,1) = r0/norm(r0);
for j = 1:kmax
  w = A*V(:,j);
  for i = 1:j
    H(i,j) = V(:,i)'*w;
    w = w - H(i,j)*V(:,i);
  end
  H(j+1,j) = norm(w);
  if (H(j+1,j)<e)
    H = H(1:j+1,1:j);
    V = V(:,1:j);
    p = 2;
    k = j;
    return
  end
```

```
    V(:,j+1) = w/H(j+1,j);
  end
  k = kmax;
end
```

Věta 2.1. *Nechť* $\mathbf{A} \in \mathbb{M}^n$. *Dále nechť* $\mathbf{V}_{k+1} \in \mathbb{M}^{n\times(k+1)}$ *a* $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$, *kde* $1 \leq k < n$, *jsou matice získané algoritmem 2.1. Tedy matice* $\mathbf{V}_{k+1}$ *je semi-ortogonální a matice* $\overline{\mathbf{H}}_k$ *je v horním Hessenbergově tvaru. Pak symbolem* $\mathbf{V}_k \in \mathbb{M}^{n\times k}$ *označíme matici* $\mathbf{V}_{k+1}$ *bez posledního sloupce a symbolem* $\mathbf{H}_k \in \mathbb{M}^k$ *matici* $\overline{\mathbf{H}}_k$ *bez posledního řádku. Platí*

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\overline{\mathbf{H}}_k, \tag{2.6}$$

$$\mathbf{V}_k^T\mathbf{A}\mathbf{V}_k = \mathbf{H}_k. \tag{2.7}$$

Při předčasném ukončení výpočtu z důvodu rovnosti $h_{j+1,j} = 0$, $j = 1,\ldots,k$, *kde na výstupu algoritmu 2.1 je matice* $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ *a matice* $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1)\times j}$, *platí*

$$\mathbf{V}_j^T\mathbf{A}\mathbf{V}_j = \mathbf{H}_j, \tag{2.8}$$

přičemž $\mathbf{H}_j \in \mathbb{M}^j$ *je matice* $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1)\times j}$ *bez posledního řádku. Pro* $k = n$ *je na výstupu algoritmu 2.1 matice* $\mathbf{V}_k$ *a platí pouze rovnost (2.7).*

Důkaz: Vztah (2.6) vyplývá z bodů 6-15 Arnoldiho algoritmu 2.1. Pro $j = 1,\ldots,k$ platí

$$\mathbf{A}\mathbf{v}^j = \mathbf{w}^j = \mathbf{w}^{j+1} + \sum_{i=1}^{j} h_{ij}\mathbf{v}^i = h_{j+1,j}\mathbf{v}^{j+1} + \sum_{i=1}^{j} h_{ij}\mathbf{v}^i = \sum_{i=1}^{j+1} h_{ij}\mathbf{v}^i,$$

kde $\mathbf{v}^i$, $i = 1,\ldots,j+1$ značí i-tý sloupec matice $\mathbf{V}_{j+1}$ a h_{ij} jsou prvky matice $\overline{\mathbf{H}}_j$. Vynásobením matic $\mathbf{V}_j^T$ a $\mathbf{V}_{j+1}$ vznikne jednotková matice $\mathbf{I} \in \mathbb{M}^{j\times(j+1)}$. To je dáno tím, že matice $\mathbf{V}_{j+1}$ a $\mathbf{V}_j$ jsou semi-ortogonální. Pro $j = 1,\ldots,k$ platí tedy rovnost

$$\mathbf{V}_j^T\mathbf{V}_{j+1}\overline{\mathbf{H}}_j = \mathbf{H}_j. \tag{2.9}$$

Dosazením (2.6) do (2.9) získáme vztahy (2.7) a (2.8).

Pro $k = n$ platí, že na výstupu algoritmu 2.1 je matice $\mathbf{V}_k$ místo $\mathbf{V}_{k+1}$. V přesné aritmetice se totiž jedná o finitní metodu, a výpočet tak skončí z důvodu splnění podmínky $h_{k+1,k} = 0$. Z toho vyplývá poslední tvrzení věty. □

Poznámka 2.3. Důkaz k větě 2.1 je zde uveden z důvodu jeho převzetí z [10, str. 147] a následného rozvedení pro hlubší pochopení vztahů mezi jednotlivými maticemi.

Příklad 2.1. Pomocí kódu 2.1 vypočtěte matice $\mathbf{V}_4$ a $\overline{\mathbf{H}}_3$, je-li dáno

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 4 & -1 & 2 \\ 0 & -2 & -3 & 0 & 3 \\ 3 & 1 & 4 & -3 & 3 \\ -2 & 3 & 2 & 1 & -1 \\ 3 & -3 & 4 & -2 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 2 \\ 4 \\ -1 \\ 1 \\ -3 \end{pmatrix}.$$

Řešení: Ze zadání plyne, že $k = 3 < 5 = n$. V kódu tak nastavíme $k_{\max} = 3$. Vektor $\mathbf{x}^0$ a hodnota ϵ v zadání nejsou stanoveny. Zadáme příkazy

```
>> A = [2,0,4,-1,2;0,-2,-3,0,3;3,1,4,-3,3;-2,3,2,1,-1;3,-3,4,-2,1];
>> b = [2;4;-1;1;-3];
>> [V,H] = arnoldiK(A,b,3)
```

Výsledné matice $\mathbf{V}_4$ (v kódu `V`) a $\overline{\mathbf{H}}_3$ (v kódu `H`) jsou tvaru

```
V =

 0.3592  -0.2613  -0.4518  -0.6496
 0.7184  -0.5227   0.1997   0.2618
-0.1796  -0.2561  -0.7929   0.5201
 0.1796   0.4189  -0.3370  -0.3246
-0.5388  -0.6461   0.1170  -0.3656
```

```
H =


-0.2903    0.5731    3.1873
 4.4118    3.2395    3.0330
      0    5.7932    4.0739
      0         0    3.1034
```

Snadno lze také ověřit, že sloupce matice $\mathbf{V}_4$ jsou dle definice 1.6 ortonormální:

```
>> I = V'*V;
I =


 1.0000e+00   3.6461e-17  -4.6546e-18  -1.5394e-16
 3.6461e-17   1.0000e+00  -5.4030e-17  -1.1068e-16
-4.6546e-18  -5.4030e-17   1.0000e+00   3.4284e-17
-1.5394e-16  -1.1068e-16   3.4284e-17   1.0000e+00
```

Během výpočtu pomocí kódu vznikají zaokrouhlovací chyby. Uvažujeme-li maximální přípustnou chybu $\epsilon > 10^{-15}$, pak můžeme říct, že matice `I` je jednotková. Tím jsme ověřili, že jsou sloupce matice $\mathbf{V}_4$ ortonormální.

Poznámka 2.4. V další části této práce budeme maticemi $\overline{\mathbf{H}}_k$, $\mathbf{H}_k$, $\mathbf{V}_{k+1}$ a $\mathbf{V}_k$ (příp. maticemi $\overline{\mathbf{H}}_j$, $\mathbf{H}_j$ a $\mathbf{V}_j$) rozumět matice uvedené ve větě 2.1.

Věta 2.2. *Označme symbolem j^* index j, $1 \leq j \leq k \leq n$, takový, pro který Arnldiho algoritmus 2.1 skončí z důvodu rovnosti $h_{j+1,j} = 0$. Pak matice $\mathbf{H}_j$, $j \leq j^*$ je regulární a hodnost matice $\overline{\mathbf{H}}_j$ je rovna j.*

Důkaz: viz [2, str. 670-671]. □

Případu ukončení Arnoldiho algoritmu 2.1 z důvodu rovnosti $h_{j+1,j} = 0$, kde $1 \leq j \leq k$, se budeme blíže věnovat v následující kapitole v kontextu se samotnou Zobecněnou metodou minimálních reziduí.

2.3. Princip GMRES

V této kapitole se budeme opírat o poznatky z literatury [2].

Zobecněná metoda minimálních reziduí (zkráceně GMRES, z anglického *generalized minimal residual method*), jak již název napovídá, minimalizuje v j-té iteraci normu rezidua soustavy (2.1). Postupně generujeme posloupnost vektorů $\{\mathbf{x}^j\}$, které aproximují hledané řešení $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ a splňují vlastnosti

$$\mathbf{x}^j \in \mathbf{x}^0 + \mathcal{K}_j,$$

$$\|\mathbf{r}^j\| = \|\mathbf{b} - \mathbf{A}\mathbf{x}^j\| = \min_{\mathbf{x} \in \mathbf{x}^0 + \mathcal{K}_j} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|, \quad j = 1, \ldots, k. \tag{2.10}$$

Vektor $\mathbf{x}^j$ tedy leží v prostoru generovaném vektorem počáteční aproximace $\mathbf{x}^0$ a vektory z Krylovova podprostoru $\mathcal{K}_j$ řádu j.

Z definice 2.1 vyplývá vlastnost Krylovových prostorů

$$\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \cdots \subseteq \mathcal{K}_k \subseteq \cdots \subseteq \mathcal{K}_n, \tag{2.11}$$

ze které plyne, že norma $\|\mathbf{r}^j\|$ daná vztahem (2.10) je pro $j = 1, \ldots, k$ nerostoucí.

Jak již bylo rozebráno v předchozích kapitolách, je žádoucí při výpočtech pracovat s maticí $\mathbf{V}_k$, jejíž sloupce tvoří ortonormální bázi prostoru $\mathcal{K}_k$. Do algoritmu GMRES tedy zakomponujeme Arnoldiho algoritmus 2.1 a namísto řešení úlohy (2.10) hledáme pro $j = k$ vektor

$$\mathbf{x}^k = \mathbf{x}^0 + \mathbf{V}_k\mathbf{y}, \quad \mathbf{y} \in \mathbb{R}^k, \tag{2.12}$$

takový, aby

$$\|\mathbf{r}^k\| = \min_{\mathbf{y}} \|\mathbf{b} - \mathbf{A}(\mathbf{x}^0 + \mathbf{V}_k\mathbf{y})\|. \tag{2.13}$$

Využitím vztahu (2.6) a rovnosti $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$ upravíme výraz z úlohy (2.13)

$$\|\mathbf{b} - \mathbf{A}(\mathbf{x}^0 + \mathbf{V}_k\mathbf{y})\| = \|\mathbf{r}^0 - \mathbf{A}\mathbf{V}_k\mathbf{y}\| = \|\|\mathbf{r}^0\|\mathbf{v}^1 - \mathbf{V}_{k+1}\overline{\mathbf{H}}_k\mathbf{y}\|. \tag{2.14}$$

Snadno lze dokázat, že je matice $\mathbf{V}_{k+1}$ dle definice 1.12 semi-ortogonální. Víme, že jednotlivé sloupce matice $\mathbf{V}_{k+1}$ jsou ortonormální a platí tak

$\mathbf{V}_{k+1}^T \mathbf{V}_{k+1} = \mathbf{I}$, $\mathbf{I} \in \mathbb{M}^{k+1}$. Lze tedy využít 2. bodu věty 1.4 o vlastnostech semiortogonálních matic k úpravě výrazu (2.14) do tvaru

$$\|\mathbf{V}_{k+1}\mathbf{z}\| = \|\mathbf{z}\|,$$

kde $\mathbf{z} = \|\mathbf{r}^0\|\mathbf{e} - \overline{\mathbf{H}}_k\mathbf{y}$, $\mathbf{e} \in \mathbb{R}^{k+1}$. Výslednou úlohou je minimalizace normy vektoru

$$\begin{pmatrix} \|\mathbf{r}^0\| \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1k} \\ h_{21} & h_{22} & \dots & h_{2k} \\ 0 & h_{32} & \dots & h_{3k} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & h_{k+1,k} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}$$

nalezením vhodného vektoru $\mathbf{y}$, tedy

$$\min_{\mathbf{y}} \|\|\mathbf{r}^0\|\mathbf{e} - \overline{\mathbf{H}}_k\mathbf{y}\|. \tag{2.15}$$

Na tomto místě je vhodné se zmínit o situaci, kdy je Arnoldiho algoritmus 2.1 v rámci řešení soustavy lineárních rovnic Zobecněnou metodou minimálních reziduí ukončen při rovnosti $h_{j+1,j} = 0$, $1 \leq j \leq k$. Platí následnující věta.

Věta 2.3 (Breakdown)**.** *Arnoldiho algoritmus 2.1 je při řešení soustavy lineárních rovnic* $\mathbf{Ax} = \mathbf{b}$ *ukončen v j-té iteraci z důvodu rovnosti* $h_{j+1,j} = 0$, $1 \leq j \leq k \leq n$, *právě tehdy, je-li nalezeno přesné řešení této soustavy* $\mathbf{x}^0 + \mathbf{V}_j\mathbf{y} = \mathbf{x}^j = \mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.

Důkaz: viz [2, str. 671-672] a [10, str. 164]. □

Je-li algoritmus 2.1 ukončen z důvodu rovnosti $h_{j+1,j} = 0$, $1 \leq j \leq k \leq n$, bude na jeho výstupu matice $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ a $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1)\times j}$ v horním Hessenbergově tvaru, jejíž poslední řádek bude z důvodu $h_{j+1,j} = 0$ nulový. Má tedy smysl uvažovat místo $\overline{\mathbf{H}}_j$ matici $\mathbf{H}_j$.

Úloha, kterou budeme v tomto případě řešit, bude mít po obdobné úpravě, jako byla úprava úlohy (2.13) v předchozí části do tvaru (2.15), podobu

$$\min_{\mathbf{y}} \|\|\mathbf{r}^0\|\mathbf{e} - \mathbf{H}_j\mathbf{y}\|,$$

kde $\mathbf{e} \in \mathbb{R}^j$. Matice $\mathbf{H}_j \in \mathbb{M}^j$ je na základě věty 2.2 regulární. Dle definice 1.15 tak řešíme soustavu j lineárních rovnic o j neznámých, která má podobu $\mathbf{H}_j\mathbf{y} = \|\mathbf{r}^0\|\mathbf{e}$. Její řešení je dle poznámky 1.6 ve tvaru

$$\mathbf{y} = \|\mathbf{r}^0\|\mathbf{H}_j^{-1}\mathbf{e}.$$

Řešením soustavy (2.1) je v tomto případě podle věty 2.3 vektor

$$\mathbf{x}^* = \mathbf{x}^j = \mathbf{x}^0 + \mathbf{V}_j\mathbf{y}.$$

2.4. Úloha nejmenších čtverců

Dílčí částí GMRES je i úloha (2.15). Hodnost rozšířené matice $(\overline{\mathbf{H}}_k|\|\mathbf{r}^0\|\mathbf{e}) \in \mathbb{M}^{k+1}$ je větší než hodnost matice $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$, která je dle věty 2.2 rovna k, tj.

$$h(\overline{\mathbf{H}}_k|\|\mathbf{r}^0\|\mathbf{e}) = k+1 > k = h(\overline{\mathbf{H}}_k).$$

Dle Frobeniovy věty 1.5 tak soustava $\overline{\mathbf{H}}_k\mathbf{y} = \|\mathbf{r}^0\|\mathbf{e}$ nemá řešení. Úlohu (2.15) řešíme tedy jako úlohu nejmenších čtverců.

Definice 2.3. Buď $\mathbf{A} = (a_{ij})$, $\mathbf{A} \in \mathbb{M}^{m\times n}$, kde $m > n$, a $\mathbf{b} = (b_1, \ldots, b_m)^T$, $\mathbf{b} \in \mathbb{R}^m$. Nechť dále platí $h(\mathbf{A}) \neq h(\mathbf{A}|\mathbf{b})$. Pak soustava $\mathbf{Ax} = \mathbf{b}$ nemá řešení. Aproximaci řešení získáme tzv. metodou nejmenších čtverců, při které řešíme úlohu

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\| = \min_{\mathbf{x}} \sqrt{\sum_{i=1}^{m}(\sum_{j=1}^{n} a_{ij}x_j - b_i)^2}. \tag{2.16}$$

Takovou úlohu nazýváme úlohou nejmenších čtverců.

Věta 2.4. *Řešení úlohy (2.16) existuje a je rovno řešení soustavy* $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$.

Důkaz: viz [4, str. 121-122]. □

Důsledek 2.1. *Mějme* $\mathbf{A} \in \mathbb{M}^{m\times n}$ *a* $\mathbf{b} \in \mathbb{R}^m$ *určené v definici 2.3. Pak přibližné řešení soustavy* $\mathbf{Ax} = \mathbf{b}$ *metodou nejmenších čtverců je* $\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$, *a je jednoznačné.*

Z důsledku 2.1 vyplývá, že úloha (2.15) má jediné řešení ve tvaru

$$\mathbf{y} = (\overline{\mathbf{H}}_k^T \overline{\mathbf{H}}_k)^{-1} \overline{\mathbf{H}}_k^T \|\mathbf{r}^0\| \mathbf{e}. \tag{2.17}$$

Výpočet řešení pomocí vzorce (2.17) je vzhledem k tvaru matice $\overline{\mathbf{H}}_k$ velmi náročný co do výpočetních operací. Nabízí se rozklad matice $\overline{\mathbf{H}}_k$ pomocí QR rozkladu, přičemž využijeme horního Hessenbergova tvaru matice $\overline{\mathbf{H}}_k$.

Věta 2.5 (QR rozklad)**.** *Pro každou matici* $\mathbf{A} \in \mathbb{M}^{m\times n}$ *existují ortogonální matice* $\mathbf{Q} \in \mathbb{M}^m$ *a horní trojúhelníková matice* $\mathbf{R} \in \mathbb{M}^{m\times n}$, $\mathbf{R} = (r_{ij}) \geq 0$, *takové, že* $\mathbf{A} = \mathbf{QR}$.

Důkaz: viz [4, str. 185]. □

Obecně tedy budeme chtít matici $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$ v horním Hessenbergově tvaru rozložit na součin ortogonální matice $\mathbf{Q} \in \mathbb{M}^{k+1}$ a horní trojúhelníkové matice $\mathbf{R} \in \mathbb{M}^{(k+1)\times k}$, jejíž poslední řádek je dle definice 1.16 nulový. Je zřejmé, že budeme převádět matici $\overline{\mathbf{H}}_k$ na matici $\mathbf{R}$ vynulováním k prvků nacházejících se pod hlavní diagonálou $\overline{\mathbf{H}}_k$. K tomu využijeme tzv. Givensovy matice.

Definice 2.4. Givensova matice rovinné rotace je ortogonální matice $\mathbf{G}_{ij} = (g_{ij})_{i,j=1}^n$ tvaru

$$\mathbf{G}_{ij} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & \sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix},$$

která se od jednotkové matice $\mathbf{I} \in \mathbb{M}^n$ liší v nejvýše čtyřech prvcích

$$g_{ii} = g_{jj} = \cos\theta \quad \text{a} \quad g_{ij} = -g_{ji} = \sin\theta, \quad i \neq j,\ \theta \in \langle -\pi, \pi \rangle.$$

Poznámka 2.5. V další části této práce budeme uvažovat Givensovy matice $\mathbf{G}_{j,j+1} \in \mathbb{M}^{k+1}$. Zkráceně je budeme značit $\mathbf{G}_j$.

Postupným násobením matice $\overline{\mathbf{H}}_k$ maticemi $\mathbf{G}_j$, $j = 1, \ldots, k$ zleva se dopočítáme na matici v horním trojúhelníkovém tvaru, kterou si účelně označíme $\overline{\mathbf{R}}$. Mějme nejprve matici $\overline{\mathbf{H}}_k$, $1 \leq k < n$, tvaru

$$\overline{\mathbf{H}}_k = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1k} \\ h_{21} & h_{22} & \ldots & h_{2k} \\ 0 & h_{32} & \ldots & h_{3k} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & h_{k+1,k} \end{pmatrix}.$$

Naším úkolem je vynulovat prvky na pozicích pod hlavní diagonálou, tj. $h_{21}, h_{32}, \ldots, h_{k+1,k}$. Všimněme si, že k tomu budeme potřebovat přesně k matic $\mathbf{G}_j$, $j = 1, \ldots, k$, které budou tvaru

$$\mathbf{G}_j = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \frac{h_{jj}}{\sqrt{h_{jj}^2+h_{j+1,j}^2}} & \frac{h_{j+1,j}}{\sqrt{h_{jj}^2+h_{j+1,j}^2}} & \cdots & 0 \\ 0 & \cdots & -\frac{h_{j+1,j}}{\sqrt{h_{jj}^2+h_{j+1,j}^2}} & \frac{h_{jj}}{\sqrt{h_{jj}^2+h_{j+1,j}^2}} & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \Bigg\} \text{řádky } j \text{ a } j+1.$$

$$\overbrace{\qquad\qquad\qquad\qquad}^{\text{sloupce } j \text{ a } j+1}$$

Při dalších výpočtech nebude nutné přímo vyčíslit úhel θ. Postačí, aby platila rovnost $\sin^2\theta + \cos^2\theta = 1$, tedy

$$\left(\frac{h_{j+1,j}}{\sqrt{h_{jj}^2 + h_{j+1,j}^2}}\right)^2 + \left(\frac{h_{jj}}{\sqrt{h_{jj}^2 + h_{j+1,j}^2}}\right)^2 = 1, \tag{2.18}$$

což je zřejmě splněno.

Poznámka 2.6. V další části práce budeme horními indexy v závorce u vektorů, příp. matic, značit iteraci, ve které se vektor, příp. matice, v daném tvaru vyskytují.

Vynásobením matice $\overline{\mathbf{H}}_k$ zleva maticí $\mathbf{G}_1$ vznikne matice $\overline{\mathbf{H}}_k^{(2)}$, která se od matice $\overline{\mathbf{H}}_k$ liší v prvních dvou řádcích. Následující Givensovu matici $\mathbf{G}_2$ je tedy nutné vypočítat pomocí prvků nově vzniklé matice $\overline{\mathbf{H}}_k^{(2)}$. Obecně tedy počítáme další Givensovy matice $\mathbf{G}_{i+1}$, $i = 1, \ldots, k-1$ pomocí prvků matic $\overline{\mathbf{H}}_k^{(i+1)}$, které mají již vynulovaných celkem i prvků pod hlavní diagonálou oproti matici $\overline{\mathbf{H}}_k$.

Platí tedy vztah

$$\mathbf{G}_k \mathbf{G}_{k-1} \cdots \mathbf{G}_1 \overline{\mathbf{H}}_k = \overline{\mathbf{H}}_k^{(k+1)} = \overline{\mathbf{R}}, \tag{2.19}$$

kde $\overline{\mathbf{R}} \in \mathbb{M}^{(k+1)\times k}$ je již dříve zmíněná matice v horním trojúhelníkovém tvaru.

Dále víme, že jsou matice $\mathbf{G}_j$ ortogonální. S využitím vět 1.2 a 1.3 získáme ze vztahu (2.19) QR rozklad matice $\overline{\mathbf{H}}_k$

$$\overline{\mathbf{H}}_k = \mathbf{Q}^{-1}\overline{\mathbf{R}} = \mathbf{Q}^T\overline{\mathbf{R}}, \tag{2.20}$$

kde $\mathbf{Q} = \mathbf{G}_k \mathbf{G}_{k-1} \cdots \mathbf{G}_1$, $\mathbf{Q} \in \mathbb{M}^{k+1}$ a $\overline{\mathbf{R}} \in \mathbb{M}^{(k+1)\times k}$.

QR rozklad matice $\overline{\mathbf{H}}_k$ je tedy součin matic $\mathbf{Q}^T$ a $\overline{\mathbf{R}}$, přičemž toto označení ponecháme, a to právě kvůli způsobu výpočtu těchto matic. Tak bude teoretický zápis shodný s výpočtem v našem kódu pro získání QR rozkladu matice $\overline{\mathbf{H}}_k$ pomocí Givensových matic rovinné rotace.

Na vstupu kódu 2.2 tedy zadáme matici $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1)\times k}$, a případně i nepovinný parametr $\epsilon \in \mathbb{R}$, tedy přípustnou chybu výpočtu, která je implicitně nastavena na $\epsilon = 10^{-6}$. Na výstupu budou matice $\mathbf{Q}$ a $\overline{\mathbf{R}}$, přičemž nadále platí, že $\overline{\mathbf{H}}_k = \mathbf{Q}^T\overline{\mathbf{R}}$.

M-soubor 2.2.

```
function [Q,R] = givensK(H,e)

  % H je matice v hornim Hessenbergove tvaru
  % e je libovolne male kladne cislo epsilon

  % Q je ortogonalni matice
  % R je horni trojuhelnikova matice
  % Q' a R tvori QR rozklad matice H, tzn. H = Q'*R

  k = length(H(1,:));
  Q = eye(k+1);
  H0 = H;

  if (nargin<2)||(isempty(e))
    e = 1e-6; %implicitni hodnota e
  end

  for j = 1:k
    G = eye(k+1);
    G(j,j) = H0(j,j)/sqrt(H0(j,j)^2+H0(j+1,j)^2);
    G(j+1,j+1) = G(j,j);
    G(j,j+1) = H0(j+1,j)/sqrt(H0(j,j)^2+H0(j+1,j)^2);
    G(j+1,j) = -G(j,j+1);
    Q = G*Q;
    H0 = G*H0;
  end
  R = Q*H;
  R(abs(R)<e) = 0;
end
```

Příklad 2.2. Pomocí kódu 2.2 vypočtěte QR rozklad matice $\overline{\mathbf{H}}_3$ z příkladu 2.1.

Řešení: Mějme matici $\overline{\mathbf{H}}_3$ (v kódu `H`) z příkladu 2.1.

```
>> H
H =

-0.2903   0.5731   3.1873
 4.4118   3.2395   3.0330
      0   5.7932   4.0739
      0        0   3.1034
```

Její QR rozklad získáme zadáním příkazu

```
>> [Q,R] = givensK(H)
Q =

-0.0657   0.9978        0        0
 0.1339   0.0088   0.9910        0
 0.6627   0.0436  -0.0899   0.7422
-0.7339  -0.0483   0.0996   0.6702

R =

4.4214   3.1949   2.8172
     0   5.8461   4.4907
     0        0   4.1813
     0        0        0
```

Na první pohled je vidět, že matice $\overline{\mathbf{R}}$ (v kódu `R`) je maticí v horním trojúhelníkovém tvaru, jejíž poslední řádek je nulový. Matice $\mathbf{Q}^T$ (v kódu `Q`) a $\overline{\mathbf{R}}$ tvoří QR rozklad matice $\overline{\mathbf{H}}_3$, což můžeme ověřit zadáním příkazu

```
>> QR = Q'*R
QR =

-0.2903    0.5731    3.1873
 4.4118    3.2395    3.0330
      0    5.7932    4.0739
      0         0    3.1034
```

Normu z úlohy minimálních čtverců (2.15) lze tedy nyní využitím rovnosti (2.20) a poznámky 1.4 upravit následovně

$$\|\|\mathbf{r}^0\|\mathbf{e} - \overline{\mathbf{H}}_k\mathbf{y}\| = \|\|\mathbf{r}^0\|\mathbf{e} - \mathbf{Q}^T\overline{\mathbf{R}}\mathbf{y}\| = \|\|\mathbf{r}^0\|\mathbf{e} - \mathbf{Q}^{-1}\overline{\mathbf{R}}\mathbf{y}\| =$$

$$= \|\mathbf{Q}\|\mathbf{r}^0\|\mathbf{e} - \mathbf{Q}\mathbf{Q}^{-1}\overline{\mathbf{R}}\mathbf{y}\| = \|\overline{\mathbf{g}} - \overline{\mathbf{R}}\mathbf{y}\|,$$

kde $\overline{\mathbf{g}} = \mathbf{Q}\|\mathbf{r}^0\|\mathbf{e}$, $\mathbf{e}, \overline{\mathbf{g}} \in \mathbb{R}^{k+1}$. Úlohu nejmenších čtverců (2.15) lze tedy přepsat do tvaru

$$\min_{\mathbf{y}} \|\overline{\mathbf{g}} - \overline{\mathbf{R}}\mathbf{y}\|. \tag{2.21}$$

Věta 2.6. *Nechť jsou dány matice* $\mathbf{Q}^T \in \mathbb{M}^{k+1}$ *a* $\overline{\mathbf{R}} \in \mathbb{M}^{(k+1)\times k}$*, které tvoří QR rozklad matice* $\overline{\mathbf{H}}_k$ *a vektor* $\overline{\mathbf{g}} = \mathbf{Q}\|\mathbf{r}^0\|\mathbf{e}$, $\overline{\mathbf{g}} \in \mathbb{R}^{k+1}$*. Dále nechť* $\mathbf{R} \in \mathbb{M}^k$ *je regulární matice získaná odstraněním posledního řádku matice* $\overline{\mathbf{R}}$*, který je nulový, a* $\mathbf{g} \in \mathbb{R}^k$ *je vektor získaný odstaněním poslední složky* g_{k+1} *vektoru* $\overline{\mathbf{g}}$*. Pak řešením úlohy* (2.21) *je vektor*

$$\mathbf{y} = \mathbf{R}^{-1}\mathbf{g} \tag{2.22}$$

a pro reziduum původní úlohy (2.10) *platí*

$$\|\mathbf{r}^k\| = \|\mathbf{b} - \mathbf{A}\mathbf{x}^k\| = |g_{k+1}|. \tag{2.23}$$

Důkaz: viz [10, str. 162]. □

Hledané přibližné řešení soustavy $\mathbf{Ax} = \mathbf{b}$ získané Zobecněnou metodou minimálních reziduí dopočteme dosazením nalezeného vektoru $\mathbf{y}$ z (2.22) do vztahu (2.12).

Na tomto místě již máme popsané všechny části Zobecněné metody minimálních reziduí. V následující kapitole těchto znalostí využijeme při konstrukci uceleného algoritmu.

2.5. Algoritmus GMRES

Ke konstrukci algoritmu Zobecněné metody minimálních reziduí pro řešení rozsáhlých soustav lineárních rovnic využijeme poznatků ze všech předchozích kapitol. Výchozím algoritmem je Arnoldiho algoritmus 2.1, který rozšíříme o úlohu nejmenších čtverců zahrnující i QR rozklad matice $\overline{\mathbf{H}}_k$, která je na výstupu algoritmu 2.1, pomocí Givensových matic.

Na vstupu algoritmu GMRES je dána matice $\mathbf{A} \in \mathbb{M}^n$ a vektor $\mathbf{b} \in \mathbb{R}^n$. Na výstupu je vektor $\mathbf{x}^j \in \mathbb{R}^n$, $1 \leq j \leq k \leq n$, který je aproximací řešení soustavy $\mathbf{Ax} = \mathbf{b}$, tedy $\mathbf{x}^j \approx \mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.

Dále na vstupu můžeme zvolit maximální počet iterací $k \in \mathbb{N}$, $1 \leq k \leq n$, přičemž implicitně je $k = n$, a počáteční aproximaci $\mathbf{x}^0 \in \mathbb{R}^n$, která je implicitně nastavena na $\mathbf{x}^0 = \mathbf{o}$. Na výstupu by nás mohla zajímat i norma rezidua v k-té iteraci $r^k = \|\mathbf{r}^k\|$, $r^k \in \mathbb{R}$ a relativní norma rezidua $r^k_{\text{rel}} \in \mathbb{R}$, kterou volíme ve tvaru $r^k_{\text{rel}} = \frac{r^k}{\|\mathbf{b}\|}$.

Algoritmus 2.2 (GMRES)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$ a k
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{w}^j = \mathbf{Av}^j$
  - Pro $i = 1, \ldots, j$ položme

        - $h_{ij} = \langle \mathbf{v}^i, \mathbf{w}^j \rangle$
    - $\mathbf{w}^{j+1} = \mathbf{w}^j - \sum_{i=1}^{j} h_{ij}\mathbf{v}^i$
    - $h_{j+1,j} = \|\mathbf{w}^{j+1}\|$
    - Je-li $h_{j+1,j} = 0$
        - Konec výpočtu (breakdown)
        - Matice $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^j$
        - Matice $\mathbf{H}_j = (h_{ij})$, $\mathbf{H}_j \in \mathbb{M}^j$
        - Vypočítejme $\mathbf{y} = \|\mathbf{r}^0\|\mathbf{H}_j^{-1}\mathbf{e}$
        - Vypočítejme $\mathbf{x}^* = \mathbf{x}^0 + \mathbf{V}_j\mathbf{y}$
    - Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^j}{h_{j+1,j}}$.
- Položme $\overline{\mathbf{H}}_k^{(1)} = \overline{\mathbf{H}}_k = (h_{ij})$, $\overline{\mathbf{H}}_k^{(1)} \in \mathbb{M}^{(k+1)\times k}$
- Pro $j = 1, \ldots, k$ položme
    - $\mathbf{G}_j = \mathbf{I}$
    - Vypočítejme $g_{jj} = g_{j+1,j+1} = \frac{h_{jj}^{(j)}}{\sqrt{(h_{jj}^{(j)})^2 + (h_{j+1,j}^{(j)})^2}}$
    - Vypočítejme $g_{j,j+1} = -g_{j+1,j} = \frac{h_{j+1,j}^{(j)}}{\sqrt{(h_{jj}^{(j)})^2 + (h_{j+1,j}^{(j)})^2}}$
    - Položme $\overline{\mathbf{H}}_k^{(j+1)} = \mathbf{G}_j\overline{\mathbf{H}}_k^{(j)}$
- Vypočítejme $\mathbf{Q} = \mathbf{G}_{k+1}\mathbf{G}_k \cdots \mathbf{G}_1$
- Matice $\mathbf{V}_k \in \mathbb{M}^{n\times k}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^k$
- Vypočítejme $\overline{\mathbf{R}} = \mathbf{Q}\overline{\mathbf{H}}_k = (r_{ij}) \in \mathbb{M}^{(k+1)\times k}$
- Matice $\mathbf{R} \in \mathbb{M}^k$ vznikne odstraněním posledního řádku $\overline{\mathbf{R}}$
- Vypočítejme $\overline{\mathbf{g}} = \mathbf{Q}\|\mathbf{r}^0\|\mathbf{e} = (g_1, g_2, \ldots, g_{k+1})^T$
- Vektor $\mathbf{g} = (g_1, g_2, \ldots, g_k)^T$ vznikne z $\overline{\mathbf{g}}$ odstraněním poslední složky g_{k+1}

- Vypočítejme $\mathbf{y} = \mathbf{R}^{-1}\mathbf{g}$
- Vypočítejme $\mathbf{x}^k = \mathbf{x}^0 + \mathbf{V}_k\mathbf{y}$
- Položme $r^k = |g_{k+1}|$
- Položme $r^k_{\text{rel}} = \frac{r^k}{\|\mathbf{b}\|}$

Náš kód algoritmu GMRES vytvořený v prostředí Matlab taktéž vychází z kódu Arnoldiho algoritmu a algoritmu QR rozkladu pomocí Givensových matic. Do algoritmu GMRES tedy zakomponujeme oba jmenované algoritmy 2.1 a 2.2. Navíc je nutné na začátku otestovat regularitu matice $\mathbf{A}$, která je hlavním předpokladem pro využití metody GMRES.

Na vstupu je možné kromě výše zmíněných vstupů zvolit navíc velikost přípustné chyby $\epsilon \in \mathbb{R}$, která určuje přesnost výpočtu algoritmu. V algoritmu nahrazuje ϵ kvůli automatickému zaokrouhlování nulu. Implicitně je $\epsilon = 10^{-6}$. Na výstupu je možné nechat si navíc vypsat počet skutečně provedených iterací $k \in \mathbb{N}$.

M-soubor 2.3.

```
function [x,k,r,rrel] = gmresK(A,b,kmax,x0,e)

  % A je regularni ctvercova matice radu n
  % b je n-slozkovy sloupcovy vektor
  % kmax je maximalni pocet iteraci
  % x0 je n-slozkovy sloupcovy vektor, pocatecni aproximace
  % e je libovolne male kladne cislo epsilon

  % x je odhad reseni soustavy Ax = b
  % k je pocet iteraci
  % r je norma rezidua
  % rrel je relativni norma rezidua

  n = size(A,1);
```

```
if (det(A) == 0)
  error('Matice A neni regularni!')
end

if (nargin<3)||(isempty(kmax))||(kmax <= 0)
  kmax = n; %implicitni hodnota kmax
end

if (nargin<4)||(isempty(x0))
  x0 = zeros(n,1); %implicitni vektor x0
end

if (nargin<5)||(isempty(e))
  e = 1e-6; %implicitni hodnota e
end

kmax = min(kmax,n);

[V,H,r0,k,p] = arnoldiK(A,b,kmax,x0,e);

if (p == 1)
  x = x0;
  r = norm(r0);
  rrel = r/norm(b);
  return
end

if (p == 2)
  H = H(1:k,1:k);
```

```
    y = norm(r0)*inv(H)*eye(k,1);
    x = x0 + V*y;
    r = norm(b - A*x);
    rrel = r/norm(b);
    return
  end

  [Q,R] = givensK(H,e);

  R = R(1:kmax,:);
  V = V(:,1:kmax);
  g = Q*norm(r0)*eye(kmax+1,1);
  y = inv(R)*g(1:kmax);
  x = x0 + V*y;
  r = abs(g(kmax+1));
  rrel = r/norm(b);
end
```

2.6. Konvergence GMRES

V této části práce budeme čerpat z poznatků z [2], [3] a [10].

Jak již bylo zmíněno v předchozích kapitolách, výhodou Zobecněné metody minimálních reziduí pro řešení rozsáhlých soustav lineárních rovnic tvaru (2.1) je zajištěná konvergence k přesnému řešení soustavy $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ po nejvýše n iteracích, kde n je dáno řádem matice $\mathbf{A}$. Konvergence metody GMRES vyplývá především z tvrzení, že norma rezidua v j-té iteraci $\|\mathbf{r}^j\|$ definovaná vztahem (2.10) je pro $j = 1, \ldots, k$ na základě vlastnosti (2.11) nerostoucí. Tuto vlastnost bychom mohli vypozorovat i sledováním $|g_{j+1}|$, $j = 1, \ldots, k$, které je v každé iteraci rovno normě rezidua $\|\mathbf{r}^j\|$, viz (2.23). Protože pomocí GMRES obvykle řešíme soustavy, kde $n \gg 1$, je ovšem žádoucí, aby metoda zkonvergovala po méně než n iteracích.

Rychlost konvergence Krylovových metod, tedy i GMRES, je závislá na číslu podmíněnosti matice $\mathbf{A}$ ze soustavy (2.1). Čím menší tedy bude číslo podmíněnosti matice vystupující v řešené soustavě, tím bude metoda konvergovat rychleji. Číslo podmíněnosti lze snížit např. pomocí tzv. levého předpodmínění, které funguje na principu transformace původní soustavy $\mathbf{Ax} = \mathbf{b}$ na soustavu tvaru

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b},$$

kde matice $\mathbf{M} \in \mathbb{M}^n$ je regulární symetrická pozitivně definitní matice, která aproximuje matici $\mathbf{A}$. Dále požadujeme, aby číslo podmíněnosti matice $\mathbf{M}^{-1}\mathbf{A}$ bylo menší než číslo podmíněnosti matice $\mathbf{A}$, a aby byla soustava $\mathbf{Mx} = \mathbf{b}$ snadno řešitelná.

Původní úlohu (2.10) převedeme zakomponováním předpodmiňovací matice $\mathbf{M}$ do tvaru

$$\mathbf{x}^j \in \mathbf{x}^0 + \mathcal{K}_j, \quad \mathcal{K}_j(\mathbf{A}, \mathbf{z}^0) = [\mathbf{z}^0, \mathbf{Az}^0, \mathbf{A}^2\mathbf{z}^0, \ldots, \mathbf{A}^{j-1}\mathbf{z}^0],$$

$$\|\mathbf{z}^j\| = \|\mathbf{M}^{-1}(\mathbf{b} - \mathbf{Ax}^j)\| = \|\mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{Ax}^j\| =$$

$$= \min_{\mathbf{x} \in \mathbf{x}^0 + \mathcal{K}_j} \|\mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{Ax}\|, \quad j = 1, \ldots, k.$$

V algoritmu 2.3 se použití předpodmiňovací matice projeví ihned v úvodu. Dále je patrné, že algoritmus 2.3 pracuje s normou $\|\mathbf{z}^0\|$, která nahradí normu $\|\mathbf{r}^0\|$ použitou v algoritmu 2.2. Výsledný algoritmus vypadá následovně.

Algoritmus 2.3 (GMRES s levým předpodmíněním)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$, k a $\mathbf{M}$
- Vypočítejme $\mathbf{z}^0 = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{Ax}^0)$
- Položme $\mathbf{v}^1 = \frac{\mathbf{z}^0}{\|\mathbf{z}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme

- $\mathbf{w}^j = \mathbf{M}^{-1}\mathbf{A}\mathbf{v}^j$
- Pro $i = 1, \ldots, j$ položme
  - $h_{ij} = \langle \mathbf{v}^i, \mathbf{w}^j \rangle$
- $\mathbf{w}^{j+1} = \mathbf{w}^j - \sum_{i=1}^{j} h_{ij}\mathbf{v}^i$
- $h_{j+1,j} = \|\mathbf{w}^{j+1}\|$
- Je-li $h_{j+1,j} = 0$
  - Konec výpočtu (breakdown)
  - Matice $\mathbf{V}_j \in \mathbb{M}^{n\times j}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^j$
  - Matice $\mathbf{H}_j = (h_{ij})$, $\mathbf{H}_j \in \mathbb{M}^j$
  - Vypočítejme $\mathbf{y} = \|\mathbf{z}^0\|\mathbf{H}_j^{-1}\mathbf{e}$
  - Vypočítejme $\mathbf{x}^* = \mathbf{x}^0 + \mathbf{V}_j\mathbf{y}$
- Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^j}{h_{j+1,j}}$.

- Položme $\overline{\mathbf{H}}_k^{(1)} = \overline{\mathbf{H}}_k = (h_{ij})$, $\overline{\mathbf{H}}_k^{(1)} \in \mathbb{M}^{(k+1)\times k}$
- Pro $j = 1, \ldots, k$ položme
  - $\mathbf{G}_j = \mathbf{I}$
  - Vypočítejme $g_{jj} = g_{j+1,j+1} = \frac{h_{jj}^{(j)}}{\sqrt{(h_{jj}^{(j)})^2 + (h_{j+1,j}^{(j)})^2}}$
  - Vypočítejme $g_{j,j+1} = -g_{j+1,j} = \frac{h_{j+1,j}^{(j)}}{\sqrt{(h_{jj}^{(j)})^2 + (h_{j+1,j}^{(j)})^2}}$
  - Položme $\overline{\mathbf{H}}_k^{(j+1)} = \mathbf{G}_j\overline{\mathbf{H}}_k^{(j)}$
- Vypočítejme $\mathbf{Q} = \mathbf{G}_{k+1}\mathbf{G}_k \cdots \mathbf{G}_1$
- Matice $\mathbf{V}_k \in \mathbb{M}^{n\times k}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^k$
- Vypočítejme $\overline{\mathbf{R}} = \mathbf{Q}\overline{\mathbf{H}}_k = (r_{ij}) \in \mathbb{M}^{(k+1)\times k}$
- Matice $\mathbf{R} \in \mathbb{M}^k$ vznikne odstraněním posledního řádku $\overline{\mathbf{R}}$

- Vypočítejme $\overline{\mathbf{g}} = \mathbf{Q}\|\mathbf{z}^0\|\mathbf{e} = (g_1, g_2, \ldots, g_{k+1})^T$
- Vektor $\mathbf{g} = (g_1, g_2, \ldots, g_k)^T$ vznikne z $\overline{\mathbf{g}}$ odstraněním poslední složky g_{k+1}
- Vypočítejme $\mathbf{y} = \mathbf{R}^{-1}\mathbf{g}$
- Vypočítejme $\mathbf{x}^k = \mathbf{x}^0 + \mathbf{V}_k\mathbf{y}$
- Položme $r^k = |g_{k+1}|$
- Položme $r^k_{\text{rel}} = \frac{r^k}{\|\mathbf{b}\|}$

Pravé předpodmínění vychází z úpravy původní soustavy (2.1) do tvaru

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{u} = \mathbf{b},$$

kde

$$\mathbf{u} = \mathbf{M}\mathbf{x}.$$

Oproti algoritmu 2.2 se algoritmus GMRES s pravým předpodmíněním liší jen v některých bodech, zejména v úvodu a závěru. Postačí zde tedy uvést pouze tyto části algoritmu.

Algoritmus 2.4 (GMRES s pravým předpodmíněním)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$, k a $\mathbf{M}$
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{w}^j = \mathbf{A}\mathbf{M}^{-1}\mathbf{v}^j$
  - Pro $i = 1, \ldots, j$ položme

    ...

- Vypočítejme $\mathbf{y} = \mathbf{R}^{-1}\mathbf{g}$
- Vypočítejme $\mathbf{x}^k = \mathbf{x}^0 + \mathbf{M}^{-1}\mathbf{V}_k\mathbf{y}$
- Položme $r^k = |g_{k+1}|$
- Položme $r^k_{\text{rel}} = \frac{r^k}{\|\mathbf{b}\|}$

Všimněme si, že v algoritmu 2.4 navíc není nutné explicitně vyčíslit zmiňovaný vektor $\mathbf{u} \in \mathbb{R}^n$.

Je možné použít i předpodmínění štěpením (z anglického *split preconditioning*), kterému se ovšem v této práci z důvodu rozsahu dále věnovat nebudeme.

Při využití metody GMRES je také nutné si uvědomit, že v rámci Arnoldiho algoritmu dochází k ortogonalizaci vektoru $\mathbf{A}\mathbf{v}^j$, $j = 1, \ldots, k$, vůči všem již vypočteným vektorům $\mathbf{v}^j$, díky čemuž obdržíme vektor $\mathbf{v}^{j+1}$. Výpočetní složitost tak roste úměrně ke zvolenému počtu iterací k. Pro $n \gg 0$ budeme požadovat, aby taktéž $0 \ll k < n$, což výpočet zřejmě velmi zpomalí. Řešením pro tento problém je tzv. restartovaná metoda GMRES, která bude blíže rozebrána v následující kapitole.

3. Variace GMRES

Následující podkapitoly se opírají o [3], [6], [7] a [10].

3.1. Restarted GMRES

Restartovaná metoda GMRES (z anglického *restarted generalized minimal residual method*) je variací klasické Zobecněné metody minimálních reziduí, která se, jak již název napovídá, restartuje vždy po $N \leq n$ iteracích, kde n je řád matice $\mathbf{A}$. Při dalším průběhu algoritmu 2.2 je na vstupu za počáteční aproximaci nastaven vektor, který jsme získali na výstupu předchozího průběhu, tedy $\mathbf{x}^0 = \mathbf{x}^N$. Výpočet je opět zastaven při dosažení maximálního počtu iterací k (v kódu označeno $k_{\max}$), případně při nalezení přesného řešení $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ soustavy $\mathbf{Ax} = \mathbf{b}$ (breakdown).

Následující kód je námi naprogramovaným kódem algoritmu restartované metody GMRES s možností předpodmínění maticí $\mathbf{M}$ popsaném v algoritmu 2.3. Tento kód funguje, až na drobné odchylky, stejně, jako vestavěný kód `gmres` softwaru Matlab. Od kódu 2.3 se liší na vstupu, kde je možné zadat parametry $\mathbf{M}$ a N, přičemž $\mathbf{M} \in \mathbb{M}^n$ je již zmíněná předpodmíňovací matice a $N \in \mathbb{N}$, $N \leq n$, je počet provedených iterací před restartem algoritmu. Na výstupu je možné nechat si oproti kódu 2.3 vypsat provedený počet restartů, který je označen *res*.

M-soubor 3.1.

```
function [x,k,r,rrel,res] = gmres2K(A,b,kmax,x0,e,M,N)

  % A je regularni ctvercova matice radu n
```

```
% b je n-slozkovy sloupcovy vektor
% kmax je maximalni pocet iteraci
% x0 je n-slozkovy sloupcovy vektor, pocatecni aproximace
% e je libovolne male kladne cislo epsilon
% M je regularni predpodminovaci matice radu n
% N je pocet iteraci provedenych pred restartem algoritmu

% x je odhad reseni soustavy Ax = b
% k je pocet iteraci
% r je norma rezidua
% rrel je relativni norma rezidua
% res je pocet restartu

n = size(A,1);

if (det(A) == 0)
  error('Matice A neni regularni!')
end

if (nargin<3)||(isempty(kmax))||(kmax <= 0)
  kmax = n; %implicitni hodnota kmax
end

if (nargin<4)||(isempty(x0))
  x0 = zeros(n,1); %implicitni vektor x0
end

if (nargin<5)||(isempty(e))
  e = 1e-6; %implicitni hodnota e
end
```

```
if (nargin<6)||(isempty(M))
  M = eye(n); %implicitni matice M
end

if (nargin<7)||(isempty(N))||(N <= 0)||(N > kmax)
  N = kmax; %implicitni hodnota N
end

if (det(M) == 0)
  error('Matice M neni regularni!')
end

s1 = floor(kmax/N);
n = kmax - s1*N;
rrelvek(1) = 0;

for s = 1:s1
  [V,H,r0,z0,k,p,j] = arnoldi2K(A,b,N,x0,e,M,s);

  if (p == 1)
    x = x0;
    r = norm(r0);
    rrel = r/norm(b);
    res = s - 1;
    disp('Algoritmus zkonvergoval k presnemu reseni!')
    return
  end

  if (p == 2)
```

```
  H = H(1:j,1:j);
  y = norm(z0)*inv(H)*eye(j,1);
  x = x0 + V*y;
  r = norm(b - A*x);
  rrel = r/norm(b);
  res = s - 1;
  disp('Algoritmus zkonvergoval k presnemu reseni!')
  return
end

[Q,R] = givensK(H,e);

R = R(1:N,:);
V = V(:,1:N);
g = Q*norm(z0)*eye(N+1,1);
y = inv(R)*g(1:N);
x = x0 + V*y;
r = abs(g(N+1));
rrel = r/norm(b);
res = s - 1;
if (rrel < e)
  disp('Algoritmus zkonvergoval k presnemu reseni!')
  return
end
rrelvek(s+1)=rrel;
if (abs(rrelvek(s) - rrelvek(s+1)) < e)
  disp('Algoritmus stagnuje!')
  return
end
x0 = x;
```

```
end

if (n>0)
  [V,H,r0,z0,k,p,j] = arnoldi2K(A,b,n,x0,e,M);

  if (p == 1)
    k = N*s1;
    x = x0;
    r = norm(r0);
    rrel = r/norm(b);
    res = s1;
    disp('Algoritmus zkonvergoval k presnemu reseni!')
    return
  end

  if (p == 2)
    H = H(1:j,1:j);
    y = norm(z0)*inv(H)*eye(j,1);
    x = x0 + V*y;
    r = norm(b - A*x);
    rrel = r/norm(b);
    k = N*s1 + j
    res = s1;
    disp('Algoritmus zkonvergoval k presnemu reseni!')
    return
  end

  Q = eye(kmax+1);

  [Q,R] = givensK(H,e);
```

```
    R = R(1:n,:);
    V = V(:,1:n);
    g = Q*norm(z0)*eye(n+1,1);
    y = inv(R)*g(1:n);
    x = x0 + V*y;
    r = abs(g(n+1));
    rrel = r/norm(b);
    res = s1;
    k = kmax;
    disp('Dosazen maximalni pocet iteraci!')
  else
    disp('Dosazen maximalni pocet iteraci!')
  end
  return
end
```

Z kódu je zřejmé, že pro zakomponování předpodmiňovací matice $\mathbf{M}$ bylo nutné změnit i kód Arnoldiho algoritmu 2.1. Liší se jednak přímo na vstupu, kde přibyly parametry $\mathbf{M}$ a s, kde $\mathbf{M}$ je předpodmiňovací matice a s je pomocná proměnná získaná z kódu 3.1 pro výpočet počtu provedených iterací. Dále nastaly změny i na výstupu, kde se nově objevil vektor počátečního rezidua při využití předpodmínění $\mathbf{z}^0 \in \mathbb{R}^n$ a pomocná proměnná j, kterou opět využijeme v kódu 3.1. Nový kód Arnoldiho algoritmu vypadá následovně.

M-soubor 3.2.

```
function [V,H,r0,z0,k,p,j] = arnoldi2K(A,b,kmax,x0,e,M,s)

  % A je ctvercova matice radu n
  % b je n-slozkovy sloupcovy vektor
  % kmax je maximalni pocet iteraci
```

```
% x0 je n-slozkovy sloupcovy vektor, pocatecni aproximace
% e je libovolne male kladne cislo epsilon
% M je regularni predpodminovaci matice radu n
% s je pomocna promenna vyuzita v kodu gmres2K

% V je matice s ortonormalnimi sloupcovymi vektory
% H je matice v hornim Hessenbergove tvaru
% r0 je pocatecni reziduum
% z0 je pocatecni reziduum za pouziti predpodminovaci matice
% k je pocet iteraci
% p je pomocna promenna vyuzita v kodu gmres2K
% j je pomocna promenna vyuzita v kodu gmres2K

p = 0;
n = size(A,1);

if (nargin<3)||(isempty(kmax))
  kmax = n; %implicitni hodnota kmax
end

if (nargin<4)||(isempty(x0))
  x0 = zeros(n,1); %implicitni vektor x0
end

if (nargin<5)||(isempty(e))
  e = 1e-6; %implicitni hodnota e
end

if (nargin<6)||(isempty(M))
  M = eye(n); %implicitni matice M
```

```
end

if (nargin<7)||(isempty(s))
  s = 1; %implicitni hodnota s
end

if (det(M) == 0)
  error('Matice M neni regularni!')
end

kmax = min(kmax,n);
if (kmax <= 0)
  kmax = n;
end

V = zeros(n,kmax);
H = zeros(kmax+1,kmax);

r0 = b - (A*x0);
z0 = inv(M)*(b - (A*x0));

if (norm(r0)<e)
  p = 1;
  k = kmax*(s - 1);
  disp('Zadany vektor x0 resi soustavu A*x0 = b s presnosti e.')
  V = [];
  H = [];
  return
end
V(:,1) = z0/norm(z0);
```

```
  for j = 1:kmax
    w = inv(M)*A*V(:,j);
    for i = 1:j
      H(i,j) = V(:,i)'*w;
      w = w - H(i,j)*V(:,i);
    end
    H(j+1,j) = norm(w);
    if (H(j+1,j)<e)
      H = H(1:j+1,1:j);
      V = V(:,1:j);
      p = 2;
      k = kmax*(s - 1)+j;
      return
    end
    V(:,j+1) = w/H(j+1,j);
  end
  k = s*kmax;
end
```

Nevýhodou restartované metody GMRES je, že již není zaručena konvergence po nejvýše n iteracích. Při každém restartu se totiž ztratí informace o semiortogonální matici získané v iteraci předešlé, tj. matici $\mathbf{V}_{N+1}$, která je při použití této metody na výstupu zakomponovaného Arnoldiho algoritmu 3.2. Může se tedy stát, že algoritmus začne takzvaně stagnovat, pokud matice $\mathbf{A}$ na vstupu není pozitivně definitní. Jinými slovy to znamená, že posloupnost $\{\mathbf{x}^j\}$ bude konvergovat k vektoru, který si označíme $\mathbf{x}^F$, $\mathbf{x}^F \neq \mathbf{x}^0$. Tím se ustálí i reziduum r^j, $j = 1, \ldots, k$, a relativní reziduum r^j_{rel} po určitém počtu iterací. V kódu 3.1 je právě toto reziduum použito jako indikátor stagnace. Je-li rozdíl mezi relativními rezidui ve dvou po sobě jdoucích iteracích menší než je přípustná odchylka ϵ, došlo ke stagnaci algoritmu. Mimo jiné to znamená, že celkový počet provedených iterací může přesáhnout řád matice $\mathbf{A}$ na vstupu, tedy může nastat $k > n$.

3.2. MINRES a SYMMLQ

Je-li matice $\mathbf{A} \in \mathbb{M}^n$ soustavy $\mathbf{Ax} = \mathbf{b}$ symetrická, nabízí se použití metody MINRES (z anglického *minimal residual method*), příp. metody SYMMLQ (z anglického *symmetric LQ*). U obou těchto metod nedochází ke stagnaci, předpoklad, že matice $\mathbf{A}$ je pozitivně definitní, zde tedy splněn být nemusí.

Metoda MINRES se od klasické GMRES liší především v zakomponovaném Arnoldiho algoritmu 2.1, který se zde zjednodušší na tzv. Lanczosův algoritmus.

Algoritmus 3.1 (Lanczosův)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$ a k
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Položme $h_{0,1}\mathbf{v}^0 = 0$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{w}^j = \mathbf{A}\mathbf{v}^j - h_{j-1,j}\mathbf{v}^{j-1}$
  - $h_{jj} = \langle \mathbf{v}^j, \mathbf{w}^j \rangle$
  - $\mathbf{w}^{j+1} = \mathbf{w}^j - h_{jj}\mathbf{v}^j$
  - $h_{j,j+1} = \|\mathbf{w}^{j+1}\|$
  - $h_{j+1,j} = h_{j,j+1}$
  - Je-li $h_{j+1,j} = 0$
    - Konec výpočtu (breakdown)
    - Matice $\mathbf{V}_j \in \mathbb{M}^{n \times j}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^j$
    - Matice $\mathbf{H}_j = (h_{ij})$, $\mathbf{H}_j \in \mathbb{M}^j$
  - Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^{j+1}}{h_{j+1,j}}$.

- Matice $\mathbf{V}_k \in \mathbb{M}^{n\times k}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^k$
- Matice $\mathbf{H}_k = (h_{ij})$, $\mathbf{H}_k \in \mathbb{M}^k$

Nedojde-li k dřívějšímu ukončení výpočtu, bude na výstupu algoritmu 3.1 regulární symetrická matice $\mathbf{H}_k \in \mathbb{M}^k$,

$$\mathbf{H}_k = \begin{pmatrix} h_{11} & h_{12} & 0 & 0 & \ldots & 0 \\ h_{21} & h_{22} & h_{23} & 0 & \ldots & 0 \\ 0 & h_{32} & h_{33} & h_{34} & \ddots & \vdots \\ 0 & 0 & h_{43} & h_{44} & h_{4,5} & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \ldots & 0 & h_{k,k-1} & h_{k,k} \end{pmatrix},$$

v Hessenbergově tvaru a ortogonální matice $\mathbf{V}_k \in \mathbb{M}^k$. Řešení soustavy $\mathbf{Ax} = \mathbf{b}$ pak získáme vypočtením

$$\mathbf{y} = \|\mathbf{r}^0\|\mathbf{H}_k^{-1}\mathbf{e},$$
$$\mathbf{x}^k = \mathbf{x}^0 + \mathbf{V}_k\mathbf{y}.$$

Zatímco metoda MINRES, stejně jako GMRES, minimalizuje normu rezidua soustavy $\mathbf{Ax} = \mathbf{b}$, metoda SYMMLQ využívá LQ rozkladu, tedy rozkladu matice $\mathbf{A} \in \mathbb{M}^n$ na dolní trojúhelníkovou matici $\mathbf{L} \in \mathbb{M}^n$ a ortogonální matici $\mathbf{Q} \in \mathbb{M}^n$. V této metodě je v každé iteraci vypočten vektor rezidua, který je ortogonální vůči všem vektorům rezidua získaným v předchozích iteracích.

V softwaru Matlab můžeme najít vestavěné funkce k oběma metodám pod názvy `minres` a `symmlq`.

3.3. QGMRES a DQGMRES

Další variace GMRES, tzv. kvazi-GMRES (zkráceně QGMRES, z anglického *quasi generalized minimal residual method*), vznikne nahrazením Arnoldiho algoritmu 2.1 algoritmem neúplné ortogonalizace. V rámci tohoto algoritmu dochází

k ortogonalizaci vektoru $\mathbf{A}\mathbf{v}^j$, $j = 1, \ldots, k$ vůči již vypočteným vektorům $\mathbf{v}^j$, kterých je celkem q, $q < k$. Na rozdíl od Arnoldiho algoritmu tak nově získaný vektor $\mathbf{v}^{j+1}$ nebude ortogonální vůči všem již vypočteným vektorům $\mathbf{v}^j$.

Algoritmus 3.2 (Neúplná ortogonalizace)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$, k a q
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{w}^j = \mathbf{A}\mathbf{v}^j$
  - Pro $i = \max\{1, j - q + 1\}, \ldots, j$ položme
    - $h_{ij} = \langle \mathbf{v}^i, \mathbf{w}^j \rangle$
  - $\mathbf{w}^{j+1} = \mathbf{w}^j - \sum_{i=1}^{j} h_{ij}\mathbf{v}^i$
  - $h_{j+1,j} = \|\mathbf{w}^{j+1}\|$
  - Je-li $h_{j+1,j} = 0$
    - Konec výpočtu (breakdown)
    - Matice $\mathbf{V}_j \in \mathbb{M}^{n \times j}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^j$
    - Matice $\overline{\mathbf{H}}_j = (h_{ij})$, $\overline{\mathbf{H}}_j \in \mathbb{M}^{(j+1) \times j}$
  - Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^{j+1}}{h_{j+1,j}}$.
- Matice $\mathbf{V}_{k+1} \in \mathbb{M}^{n \times (k+1)}$ je tvořena sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^{k+1}$
- Matice $\overline{\mathbf{H}}_k = (h_{ij})$, $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1) \times k}$

Nedojde-li k dřívějšímu ukončení výpočtu, budou na výstupu algoritmu 3.2 matice $\overline{\mathbf{H}}_k \in \mathbb{M}^{(k+1) \times k}$ a $\mathbf{V}_{k+1} \in \mathbb{M}^{n \times (k+1)}$. Všimněme si, že matice $\overline{\mathbf{H}}_k$ bude

v horním Hessenbergově tvaru s nejvýše $q+1$ prvky v každém řádku. Například pro $k = 6$ a $q = 3$ bude mít matice $\overline{\mathbf{H}}_6$ tento tvar

$$\overline{\mathbf{H}}_6 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & 0 & 0 & 0 \\ h_{21} & h_{22} & h_{23} & h_{24} & 0 & 0 \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} & 0 \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \\ 0 & 0 & 0 & 0 & 0 & h_{76} \end{pmatrix}.$$

Kromě zabudovaného algoritmu neúplné ortogonalizace se metoda QGMRES od metody GMRES nijak neliší. Tato metoda je tak zřejmě výpočetně méně náročná než metoda GMRES, ale z hlediska paměti jsou obě metody totožné. Pro výpočet (2.12) je totiž i při využití QGMRES nutné uschovat v paměti všech k vektorů $\mathbf{v}^j$, $j = 1, \ldots, k$, které tvoří sloupce matice $\mathbf{V}_k$.

Pro účel optimalizace velikosti využité paměti byla vytvořena další variace GMRES, tzv. přímá kvazi-GMRES (zkráceně DQGMRES, z anglického *direct quasi generalized minimal residual method*). Namísto výpočtu aproximace řešení na samém konci algoritmu vztahem (2.12) je $\mathbf{x}^k$ v každé iteraci aktualizováno, a tak mizí nutnost uschovat všech k vektorů $\mathbf{v}^j$, $j = 1, \ldots, k$ v paměti.

Po dosazení (2.22) do (2.12) získáme vztah pro přímý výpočet $\mathbf{x}^k$ ve tvaru

$$\mathbf{x}^k = \mathbf{x}^0 + \mathbf{V}_k \mathbf{R}^{-1} \mathbf{g}.$$

Položme $\mathbf{P}_k = \mathbf{V}_k \mathbf{R}^{-1}$. Pak

$$\mathbf{x}^k = \mathbf{x}^0 + \mathbf{P}_k \mathbf{g}.$$

V každé iteraci j, $j = 1, \ldots, k$, přibude matici $\mathbf{P}_k$ jeden sloupec, který si označíme $\mathbf{p}^j$. Tato vlastnost plyne z tvaru matic $\mathbf{V}_k$ a $\mathbf{R}^{-1}$.

Dále je nutné si uvědomit, že v j-té iteraci přibude vektoru $\mathbf{g}^{(j)}$ jedna složka, a to g_j, přičemž předchozí zůstanou nezměněné. To je dáno tím, že vektor $\mathbf{g}^{(j)}$ vznikne

z $\overline{\mathbf{g}}^{(j)} = \mathbf{Q}\|\mathbf{r}^0\|\mathbf{e}$ odebráním poslední složky. V každé iteraci j, $j = 1, \ldots, k-1$ má na výpočet $\overline{\mathbf{g}}^{(j)}$ vliv pouze matice $\mathbf{G}_j$, poněvadž platí $\mathbf{Q}^{(j)} = \mathbf{G}_j\mathbf{Q}^{(j-1)}$. Z tvaru matice $\mathbf{G}_j$ je zřejmé, že se jejím vynásobením vektoru $\overline{\mathbf{g}}^{(j)}$ zleva změní pouze složky g_j a g_{j+1}. V iteraci $j+1$ pak g_{j+1} a g_{j+2}. Složky vektoru $\mathbf{g}^{(j)} = (g_1, g_2, \ldots, g_j)^T$ tak zůstanou v každé další iteraci nezměněny. Dále také platí

$$g_1 = \cos\theta^{(1)}\|\mathbf{r}^0\|,$$
$$g_j = \cos\theta^{(j)} g_{j-1}, \quad j = 2, \ldots, k,$$

kde

$$\cos\theta^{(j)} = \frac{h_{jj}}{\sqrt{h_{jj}^2 + h_{j+1,j}^2}}, \quad j = 1, \ldots, k,$$

viz (2.18). Proto je možné $\mathbf{x}^k$ aktualizovat v každé iteraci $j = 1, \ldots, k$ vztahem

$$\mathbf{x}^k = \mathbf{x}^{k-1} + g_k\mathbf{p}^k.$$

Kompletní algoritmus metody DQGMRES lze dohledat v [10, str. 169].

3.4. QMR a TFQMR

Další variací GMRES je metoda kvazi-minimálních reziduí (zkráceně QMR, z anglického *quasi-minimal residual method*), která je založena na principu metod MINRES a DQGMRES. Na matici $\mathbf{A} \in \mathbb{M}^n$ zde není kladen požadavek symetrie, stačí tedy, aby byla matice regulární.

Na rozdíl od Arnoldiho algoritmu 2.1 metoda QMR opět využívá Lanczosův algoritmus 3.1. Tím, že na jeho vstupu již není symetrická matice, tak se na výstupu objeví matice $\mathbf{V}_{k+1} \in \mathbb{M}^{n\times(k+1)}$ tvořená sloupcovými vektory $\mathbf{v}^1, \ldots, \mathbf{v}^{k+1}$, které již netvoří zcela ortonormální bázi, což platí i pro matici $\mathbf{V}_{k+1}$ na výstupu algoritmu 3.2. Dále je tedy algoritmus QMR shodný s algoritmem DQGMRES, jehož princip byl vysvětlen v předchozí podkapitole. Na výsledek je pak nahlíženo jako na řešení s kvazi-minimálním reziduem, odtud i název metody.

Algoritmus metody QMR lze dohledat v [10, str. 212]. V softwaru Matlab existuje vestavěná funkce k této metodě pod názvem `qmr`.

Modifikací metody, která se vyhýbá výpočtu s použitím matice $\mathbf{A}^T$, je metoda kvazi-minimálních reziduí bez transpozice (zkráceně TFQMR, z anglického *transpose-free quasi-minimal residual method*). Princip a algoritmus metody lze dohledat v [10, str. 219-226] a [6, str. 51-54]. I k této metodě existuje v softwaru Matlab zabudovaná funkce s názvem `tfqmr`.

3.5. FGMRES

Flexibilní zobecněná metoda minimálních reziduí (zkráceně FGMRES, z anglického *flexible generalized minimal residual method*) je restartovanou verzí algoritmu GMRES s pravým předpodmíněním 2.4 s možností využití jiné předpodmiňovací matice $\mathbf{M}$ v každé iteraci. Algoritmus je zformulován tak, aby se dynamicky přizpůsoboval měnící se předpodmiňovací matici, odtud název metody.

Algoritmus metody FGMRES tedy vychází z algoritmu GMRES s pravým předpodmíněním 2.4. Hlavním rozdílem je, že namísto jedné konstantní zvolené předpodmiňovácí matice $\mathbf{M}$ můžeme počítat v každé iteraci j s jinou maticí $\mathbf{M}_j$, $j = 1, \ldots, k$.

Algoritmus 3.3 (FGMRES)**.**

- Mějme $\mathbf{A}$ a $\mathbf{b}$
- Zvolme $\mathbf{x}^0$, k, ϵ a $\mathbf{M}_1, \ldots, \mathbf{M}_k$
- Vypočítejme $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
- Položme $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|}$
- Pro $j = 1, \ldots, k$ vypočítejme
  - $\mathbf{z}^j = \mathbf{M}_j^{-1}\mathbf{v}^j$
  - $\mathbf{w}^j = \mathbf{A}\mathbf{z}^j$
  - Pro $i = 1, \ldots, j$ položme
    - $h_{ij} = \langle \mathbf{v}^i, \mathbf{w}^j \rangle$

- $\mathbf{w}^{j+1} = \mathbf{w}^j - \sum_{i=1}^{j} h_{ij}\mathbf{v}^i$
- $h_{j+1,j} = \|\mathbf{w}^{j+1}\|$
- Je-li $h_{j+1,j} = 0$
  - Konec výpočtu (breakdown)
  - Matice $\mathbf{Z}_j \in \mathbb{M}^{n\times j}$ je tvořena sloupcovými vektory $\mathbf{z}^1, \ldots, \mathbf{z}^j$
  - Matice $\mathbf{H}_j = (h_{ij})$, $\mathbf{H}_j \in \mathbb{M}^j$
  - Vypočítejme $\mathbf{y} = \|\mathbf{r}^0\|\mathbf{H}_j^{-1}\mathbf{e}$
  - Vypočítejme $\mathbf{x}^* = \mathbf{x}^0 + \mathbf{Z}_j\mathbf{y}$
- Jinak položme $\mathbf{v}^{j+1} = \frac{\mathbf{w}^j}{h_{j+1,j}}$.

- Položme $\overline{\mathbf{H}}_k^{(1)} = \overline{\mathbf{H}}_k = (h_{ij})$, $\overline{\mathbf{H}}_k^{(1)} \in \mathbb{M}^{(k+1)\times k}$
- Pro $j = 1, \ldots, k$ položme

  ...
- Vypočítejme $\mathbf{y} = \mathbf{R}^{-1}\mathbf{g}$
- Matice $\mathbf{Z}_k \in \mathbb{M}^{n\times k}$ je tvořena sloupcovými vektory $\mathbf{z}^1, \ldots, \mathbf{z}^k$
- Vypočítejme $\mathbf{x}^k = \mathbf{x}^0 + \mathbf{Z}_k\mathbf{y}$
- Položme $r^k = |g_{k+1}|$
- Položme $r_{\text{rel}}^k = \frac{r^k}{\|\mathbf{b}\|}$
- Je-li $r_{\text{rel}}^k < \epsilon$
  - Konec výpočtu
- Jinak položme $\mathbf{x}^0 = \mathbf{x}^k$ a vraťme se k bodu 3

Dle [10] je při předčasném ukončení výpočtu algoritmem FGMRES (breakdown) nutno ověřit, zda je matice $\mathbf{H}_j$ v dané iteraci j regulární, aby platilo $\mathbf{x}^* = \mathbf{x}^j$.

FGMRES je vhodné využít, pokud řešená soustava vyžaduje použití alespoň dvou různých předpodmiňovacích matic pro co nejrychlejší konvergenci k přesnému řešení $\mathbf{x}^*$. Oproti restartované metodě GMRES je nutno v paměti uschovat navíc vektory $\mathbf{z}^1, \ldots, \mathbf{z}^k$, což se vzhledem k rychlosti konvergence v některých případech může vyplatit.

3.6. Další variace

Nyní jsme si představili již všechny hlavní variace Zobecněné metody minimálních reziduí. Je nutno zmínit, že se metoda GMRES neustále zdokonaluje a přizpůsobuje různým typům úloh, a tak vznikají nové variace lišící se mimo jiné typem předpodmínění, způsobem určení počtu iterací před restartem nebo třeba v zakomponovaném algoritmu na místě Arnoldiho algoritmu 2.1 v základním algoritmu GMRES 2.2. Volba vhodné variace pak závisí často především na tvaru matice $\mathbf{A}$ na vstupu.

Mezi další variace, které nebudeme podrobně rozebírat, patří například následující

- Newton-GMRES, NGB, NGQCGB (viz [12])
- GGMRES, MGMRES, LAN/MGMRES (viz [13])
- Polynomial preconditioned GMRES, GMRES-DR (viz [14])
- Weighted GMRES, deflated GMRES (viz [15])
- a další

Bližší informace ke jmenovaným variacím lze dohledat v publikacích uvedených v závorkách.

4. Aplikace GMRES

V poslední části této práce se budeme věnovat samotné aplikaci našich naprogramovaných kódů 2.3 a 3.1. Ukážeme a vyzkoušíme si jejich funkčnost na praktických příkladech a následně srovnáme dosažené výsledky s výsledky získanými matlabovskou funkcí `gmres`. U rozsáhlejších úloh srovnáme i výpočetní rychlost kódů.

Příklad 4.1. Pomocí kódu 2.3 vypočtěte aproximaci řešení soustavy $\mathbf{Ax} = \mathbf{b}$, pro $\mathbf{A} \in \mathbb{M}^5$ a $\mathbf{b} \in \mathbb{R}^5$ z příkladu 2.1 pro $k_{\max} = 3$ a $k_{\max} = 4$.
Řešení: Pro $k_{\max} = 3$ zadáme příkaz

```
>> [x,k,r,rrel] = gmresK(A,b,3)
x =


-0.3437
 0.2861
-0.5144
-0.5723
 0.5920


k = 3
r = 4.0862
rrel = 0.7339
```

Obdobně pro $k_{\max} = 4$

```
>> [x,k,r,rrel] = gmresK(A,b,4)
```

```
x =

-2.166016
-0.298893
-0.039192
-1.539964
 0.929019

k = 4
r = 3.6728
rrel = 0.6597
```

Zřejmě na vstupu nemáme řídkou matici $\mathbf{A}$, čímž se vysvětluje stále velmi vysoká relativní norma rezidua. U řídkých matic metoda GMRES totiž konverguje rychleji. V tomto příkladu bychom při implicitně nastavené výpočetní chybě $\epsilon = 10^{-6}$ dostali pro $k_{\max} = 5 = n$ přesné řešení soustavy. Při použití funkce `gmres` Matlabu, zadáme příkaz pro $k_{\max} = 3$ následovně

```
>> [x,flag,relres,iter,resvec] = gmres(A,b,[],[],3)
x =

-0.3437
 0.2861
-0.5144
-0.5723
 0.5920

flag = 1
relres = 0.7339
```

```
iter =

1   3
resvec =

5.5678
5.5557
5.5055
4.0862
```

Obdobně bychom dospěli i pro $k_{\max} = 4$ ke stejnému výsledku, jako naším naprogramovaným kódem. Narozdíl od našeho kódu si můžeme na výstupu `gmres` navíc nechat vypsat parametr $flag$, který značí, z jakého důvodu byl výpočet ukončen. Pro $flag = 0$ metoda zkonvergovala před dosažením maximálního počtu iterací, pro $flag = 1$ byl dosažen maximální počet iterací, pro $flag = 2$ jsme zadali singulární předpodmiňovací matici na vstupu a pro $flag = 3$ došlo ke stagnaci algoritmu. V našem kódu 3.1 je na výstupu automaticky slovně popsáno, z jakého důvodu byl výpočet ukončen. Dále se na výstupu `gmres` objeví parametr $resvec$, což je vektor norem reziduí v jednotlivých iteracích, včetně nulté. V našem algoritmu je vypsána pouze jeho poslední složka, tedy norma rezidua v poslední provedené iteraci. Posledním rozdílem je, že `gmres` nám vypíše jak počet vnějších, tak počet vnitřních iterací. Počet vnějších iterací značí, v kolikátém průběhu algoritmu byl výpočet ukončen. V našem kódu 3.1 tento údaj vypočteme přičtením 1 k počtu provedených restartů res. V tomto příkladu byl ovšem použit kód 2.3, tedy GMRES možnosti restartu. Počet vnějších iterací tak při použití tohoto kódu bude vždy roven jedné. Počet vnitřních iterací pak značí, kolik iterací bylo provedeno v rámci posledního průběhu algoritmu.

V dalším příkladu si ukážeme výpočet pomocí kódu 3.1 na řídké matici.

Příklad 4.2. Pomocí kódu 3.1 vypočtěte aproximaci řešení soustavy $\mathbf{Ax} = \mathbf{b}$, pro $k_{\max} = 6$, je-li dáno

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 1 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -5 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 0 \\ -5 \\ 3 \\ 1 \\ 3 \\ 8 \\ 9 \end{pmatrix}.$$

Řešení: Zadáme příkazy

```
>> A = eye(8); A(2,3) = 2; A(3,2) = -3; A(3,5) = -2;
>> A(5,5) = -1; A(6,3) = -5; A(8,1) = 2;
>> b = [3;0;-5;3;1;3;8;9];
>> [x,k,r,rrel] = gmres2K(A,b,6)
Algoritmus zkonvergoval k presnemu reseni!
x =

 3
 2
-1
 3
-1
-2
 8
 3

k = 5
```

```
r = 2.3076e-15
rrel = 1.6399e-16
```

Opět výpočet srovnáme s výpočtem funkce `gmres` v Matlabu. Dále už nebude nutné nechat výsledný vektor $\mathbf{x}^*$ znovu vypsat.

```
>> [~,flag,relres,iter] = gmres(A,b,[],[],6)
flag = 0
relres = 2.6613e-15
iter =


1   5
```

Příklad 4.3. Mějme stejné zadání, jako v příkladu 4.2. Tentokrát ovšem nastavíme maximální počet provedených iterací před restartem na $N = 4$ a zvýšíme maximální počet iterací na $k_{\max} = 100$.
Řešení: Zadáme příkazy

```
>> [~,k,r,rrel,res] = gmres2K(A,b,100,[],[],[],4)
Algoritmus zkonvergoval k presnemu reseni!
k = 48
r = 1.1227e-05
rrel = 7.9789e-07
res = 11
>> [~,flag,relres,iter]=gmres(A,b,4,[],25)
flag = 0
relres = 7.9789e-07
iter =


12    4
```

Celkový počet iterací k u funkce `gmres` lze vypočítat vzorcem

$$k = (\text{počet vnějších iteracích} - 1) \times N + \text{počet vnitřních iteracích}),$$

tedy v našem případě $k = (12 - 1) \times 4 + 4 = 48$. Počet provedených iterací při použití `gmres` je tudíž shodný s počtem provedených iterací při použití našeho naprogramovaného kódu 3.1.

V následujícím příkladu si předvedeme použití předpodmiňovací matice. Tuto matici lze získat například již zmiňovaným neúplným LU rozkladem, který nalezneme v Matlabu pod názvem `ilu`. Dále se podíváme na výpočetní rychlosti použitých kódů.

Příklad 4.4. Pomocí kódu 3.1 vypočtěte aproximaci řešení soustavy $\mathbf{Ax} = \mathbf{b}$, pro $k_{\max} = 100$ bez i za použití předpodmiňovací matice $\mathbf{M}$ získanou neúplným LU rozkladem, je-li dáno

```
>> rng(1)
>> A = randn(1000);
>> b = randn(1000,1);
```

Dále mezi sebou srovnejte jejich výpočetní rychlosti.

Řešení: Nejprve zadáme příkaz

```
>> [~,k,r,rrel] = gmres2K(A,b,100)
Dosazen maximalni pocet iteraci!
k = 100
r = 30.354
rrel = 0.9468
```

Zřejmě je i po 100 iteracích relativní norma rezidua stále velmi vysoká. Nyní vyzkoušíme použít předpodmiňovací matici, kterou zkonstruujeme užitím LU rozkladu a aplikujeme zadáním příkazů

```
>> B = sparse(A); [L,U] = ilu(B); M = L*U;
>> [~,k,r,rrel] = gmres2K(A,b,100,[],[],M)
Algoritmus zkonvergoval k presnemu reseni!
k = 1
r = 1.9797e-09
rrel = 6.1750e-11
```

Při použití matice **M** stačila ke konvergenci jediná iterace. Z hlediska rychlosti výpočet ovšem trval značně déle. To ověříme zadáním příkazů

```
>> tic;[~,k,r,rrel] = gmres2K(A,b,100);toc;
Dosazen maximalni pocet iteraci!
Elapsed time is 0.898525 seconds.
>> tic;B = sparse(A);[L,U] = ilu(B);M = L*U;
   [~,k,r,rrel] = gmres2K(A,b,100,[],[],M);toc
Algoritmus zkonvergoval k presnemu reseni!
Elapsed time is 10.9656 seconds.
```

Nyní vše srovnáme s funkcí `gmres` softwaru Matlab.

```
>> [~,flag,relres,iter] = gmres(A,b,[],[],100)
flag = 1
relres = 0.9468
iter =

1   100
>> [~,flag,relres,iter] = gmres(A,b,[],[],100,M)
flag = 0
relres = 2.9440e-11
iter =

1   1
```

```
>> tic;[~,flag,relres,iter] = gmres(A,b,[],[],100);toc;
Elapsed time is 0.318865 seconds.
>> tic;B = sparse(A);[L,U] = ilu(B);M = L*U;
   [~,flag,relres,iter] = gmres(A,b,[],[],100,M);toc
Elapsed time is 1.87884 seconds.
```

Zatímco dosažené výsledky jsou srovnatelné, výpočetní rychlost funkce `gmres` je značně rychlejší, než rychlost našeho naprogramovaného kódu 3.1, a to především při použití matice $\mathbf{M}$. Příčinu bychom mohli vypátrat srovnáním našeho kódu se zdrojovým kódem `gmres`, který ovšem nemáme k dispozici.

Na závěr můžeme vyzkoušet, zda algoritmus zkonverguje po $k_{\max} = 1000$ iteracích k přesnému řešení soustavy s danou výpočetní přesností ϵ. Zadáme příkaz

```
>> [~,k,r,rrel] = gmres2K(A,b,1000)
Algoritmus zkonvergoval k presnemu reseni!
k = 1000
r = 1.4614e-12
rrel = 4.5584e-14
```

S implicitně zvolenou přesností $\epsilon = 10^{-6}$ algoritmus zkonvergoval k přesnému řešení zadané soustavy. Zastavovacím kritériem v našem kódu 3.1 je $r_{\text{rel}}^k < \epsilon$. Kdybychom tedy na vstupu zvolili například $\epsilon = 10^{-14}$, již bychom se k přesnému řešení soustavy nepřiblížili s požadovanou přesností. To můžeme ověřit zadáním příkazu

```
>> [~,k,r,rrel] = gmres2K(A,b,1000,[],1e-14)
Dosazen maximalni pocet iteraci!
k = 1000
r = 7.6129e-13
rrel = 2.3746e-14
```

Závěr

Cílem této bakalářské práce bylo nastudovat Zobecněnou metodu minimálních reziduí pro řešení rozsáhlých soustav lineárních rovnic a následně s ní srozumitelným způsobem seznámit čtenáře, popsat některé její variace, vytvořit zdrojové kódy k této metodě v softwaru Matlab a předvést je na příkladech.

Při psaní této práce jsem se nejprve věnovala úvodu do problematiky ve smyslu zavedení definic a vět nezbytných pro její pochopení. Dále jsem se zabývala představením Krylovových metod, mezi které se Zobecněná metoda minimálních reziduí řadí. Popsala jsem všechny dílčí části metody, které jsem poté využila při sepsání teoretického uceleného algoritmu metody a poté i jejího kódu v Matlabu. Nedílnou součástí této kapitoly byla i analýza konvergence metody.

Při studiu literatury jsem narazila na nespočet variací Zobecněné metody minimálních reziduí, z nichž jsem se v kapitole o variacích snažila vyfiltrovat jen ty nejčastěji zmiňované a nejpoužívanější. Tyto jsem jednak teoreticky popsala a jednak jsem srovnala, v čem se od sebe liší.

V poslední části práce jsem své naprogramované kódy aplikovala na vlastní příklady a následně srovnala dosažené výsledky s výsledky získanými funkcí `gmres` softwaru Matlab. Dospěla jsem k závěru, že můj kód, i přes stejný výsledek, zaostává ve výpočetní rychlosti, a to především při použití předpodmiňovací matice.

Všechny vytvořené kódy byly sepsány a vyzkoušeny v matematickém softwaru GNU Octave 7.2.0. Jejich použitelnost v softwaru Matlab jsem následně prověřila.

Největším přínosem při psaní této bakalářské práce pro mě bylo, že jsem si mohla vyzkoušet napsat detailní studijní oporu ke Zobecněné metodě minimálních reziduí v českém jazyce, která v této podobě zatím sepsána nebyla. Dále jsem se naučila pracovat s typografickým programem LaTeX, ve kterém je tato práce vysázena. Věřím, že práce v budoucnu poslouží ještě mnoha dalším studentům.

Literatura

[1] Abadir, K., Magnus, J.: *Matrix Algebra. Econometric Exercises*. Cambridge: Cambridge University Press, 2005.

[2] Bulirsch, R., Stoer J.: *Introduction to Numerical Analysis*. Third Edition. New York: Springer, 2002.

[3] Demmel, J.: *Applied Numerical Linear Algebra*. Philadelphia: SIAM, 1997.

[4] Hladík, M.: *Lineární algebra (nejen) pro informatiky*. Praha: Matfyzpress, 2019.

[5] Horová, I., Zelinka, J.: *Numerické metody*. Brno: Masarykova univerzita, 2004.

[6] Kelley, C. T.: *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia: SIAM, 1995.

[7] Liesen, J., Strakoš, Z.: *Krylov Subspace Methods: Principles and Analysis*. Oxford: Oxford University Press, 2013.

[8] Olšák, P.: *Lineární algebra*. Praha: RNDr. Petr Olšák, 2007.

[9] Parlett, B. N.: *The Symmetric Eigenvalue Problem*. Philadelphia: SIAM, 1998.

[10] Saad, Y.: *Iterative Methods for Sparse Linear Systems*. Second Edition. Philadelphia: SIAM, 2003.

[11] Stewart, G. J.: *Matrix Algorithms. Volume II: Eigensystems*. Philadelphia: SIAM, 2001.

[12] An, H., Bai, Z.: A globally convergent Newton-GMRES method for large sparse systems of nonlinear equations. *Applied Numerical Mathematics* [online]. 57(3). Elsevier, 2007 [cit. 2023-03-22]. DOI: https://doi.org/10.1016/j.apnum.2006.02.007.

[13] Chen, J., Kincaid, D. R., Young, R. B.: Variations of the GMRES iterative method. *Applied Numerical Mathematics* [online]. 45(1). Elsevier, 2003 [cit. 2023-03-22]. DOI: https://doi.org/10.1016/S0168-9274(02)00231-3.

[14] Liu, Q., Morgan, R. B., Wilcox, W.: Polynomial Preconditioned GMRES and GMRES-DR. *SIAM Journal on Scientific Computing* [online]. 37(5). Siam, 2015 [cit. 2023-03-22]. DOI: https://doi.org/10.1137/140968276.

[15] Tajaddini, A., Wu, G., Zadeh, N. A.: Weighted and deflated global GMRES algorithms for solving large Sylvester matrix equations. *Numerical Algorithms* [online]. 82(1). Springer, 2018 [cit. 2023-03-22]. DOI: https://doi.org/10.1007/s11075-018-0597-9.