UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

Zobecněná Moore-Penroseova inverze a její výpočet

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **RNDr. Jitka Machalová, Ph.D.**
Vypracoval: **Petr Tománek**
Studijní program: B1101 Matematika
Studijní obor: Matematika a její aplikace
Forma studia: prezenční
Rok odevzdání: 2018

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Petr Tománek

Název práce: Zobecněná Moore-Penroseova inverze a její výpočet

Typ práce: Bakalářská práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

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

Rok obhajoby práce: 2018

Abstrakt: Cílem této bakalářské práce je nastudovat možnosti výpočtu klasické a zobecněné Moore-Penroseovy inverze. Pro výpočet inverzí je v práci uvedeno celkem sedm algoritmů. Práce je navíc doplněna o vlastní kódy daných algoritmů vypracovaných v softwaru MATLAB.

Klíčová slova: Moore-Penroseova inverze, Chipmanova pseudoinverze, software MATLAB

Počet stran: 61

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Petr Tománek

Title: Generalized Moore-Penrose Inverse and Its Calculation

Type of thesis: Bachelor's

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: RNDr. Jitka Machalová, Ph.D.

The year of presentation: 2018

Abstract: The aim of this bachelor's thesis is to study the options of calculating classic and generalized Moore-Penrose inverse. The thesis features seven algorithms for the calculation of the inverses, as well as codes of said algorithms made in the computer software MATLAB.

Key words: Moore-Penrose inverse, Chipman pseudoinverse, software Matlab

Number of pages: 61

Number of appendices: 1

Language: Czech

Prohlášení

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

V Olomouci dne ....

podpis

Poděkování

Rád bych zde poděkoval vedoucí práce RNDr. Jitce Machalové, Ph.D. za spolupráci a čas, který mi věnovala při konzultacích. Dále děkuji mé rodině za nepřetržitou podporu během celého studia.

Obsah

Seznam použitých symbolů

$\mathcal{M}_{m,n}$	množina všech matic typu $m \times n$
$A \in \mathcal{M}_{m,n}$	A je matice typu $m \times n$
A^T	matice transponovaná k matici A
I	jednotková matice příslušného typu
$tr(A)$	stopa matice A,
	tj. součet prvků na hlavní diagonále čtvercové matice A
$h(A)$	hodnost matice A

Úvod

Soustava rovnic $Ax = b$, kde x a b jsou vektory řádu n a matice A je čtvercová regulární, má právě jedno řešení, které se vypočítá vcelku jednoduše. Stačí nám k tomu matice A^{-1}, tj. matice k A inverzní. V případě, kdy matice A je libovolná, již soustava řešení obecně nemá. Ke každé matici A však existuje právě jedna Moore-Penroseova inverze či její zobecnění, které můžeme využít k aproximaci teoretického řešení.

Cílem této bakalářské práce je seznámit čtenáře s klasickou a zobecněnou Moore-Penroseovou inverzí a uvést jejich metody výpočtu. Moore-Penroseova inverze má mnoho využití ve statistice a optimalizaci. Práce se však na tato využití nezaměřuje a zabývá se zejména metodami výpočtu této inverze.

Práce je rozdělena do tří kapitol.

V první kapitole si připomeneme a seznámíme s některými pojmy z lineární algebry, které budeme v následujících kapitolách potřebovat.

Ve druhé kapitole se seznámíme s klasickou Moore-Penroseovou inverzí a jejím zobecněním, podíváme se na jejich vlastnosti a ukážeme si, jak takové inverze vypočítat pomocí sedmi algoritmů. Kapitola je dále doplněna o kódy daných algoritmů vytvořené v softwaru MATLAB.

Třetí a poslední kapitola se zabývá účinností jednotlivých kódů algoritmů v závislosti na původně zadaných maticích a jiných vlastnostech.

Kódy algoritmů jsem vytvořil v softwaru MATLAB R2015b na přenosném počítači HP 250 G3 s procesorem Intel Pentium N3530 a operační pamětí 4GB. Jednotlivé kódy jsou uloženy na CD, které je součástí přílohy.

Rád bych se zde zmínil, že v softwaru MATLAB není definována nula. Proto při výpočtech budeme muset uvést přesnost na vstupu, což bude takové číslo, jehož absolutní hodnotu již budeme považovat za nulu. Více je o této problematice řečeno ve druhé kapitole.

V práci se kromě vět také vyskytují jejich důkazy. Konce těchto důkazů jsou označeny symbolem □.

1. Úvodní kapitola

Tato kapitola slouží k seznámení s pojmy, které budeme potřebovat v následujících kapitolách. Přitom se předpokládá, že je čtenář již obeznámen se základními poznatky z lineární algebry.

Nechť A je matice typu $m \times n$ (značíme $A \in \mathcal{M}_{m,n}$). Pokud je A regulární čtvercová matice, pak k ní existuje jediná inverzní matice A^{-1}. Pro takovou matici platí

$$AA^{-1} = A^{-1}A = I.$$

Nyní uvažujme soustavu lineárních rovnic

$$Ax = b, \tag{1.1}$$

kde $A \in \mathcal{M}_{n,n}$ je opět regulární čtvercová a b je vektor řádu n, tedy $b \in \mathcal{M}_{n,1}$. Řešením takové soustavy je vektor x, který dostaneme jako výsledek rovnice

$$x = A^{-1}b.$$

Uvažujme opět soustavu (1.1), přičemž $A \in \mathcal{M}_{m,n}$ není regulární. Pak soustava může mít nekonečně mnoho řešení nebo nemusí být řešitelná. Řešení poté hledáme ve smyslu metody nejmenších čtverců (MNČ). Tedy v případě, kdy existuje nekonečně mnoho řešení, hledáme to, které je v normě nejmenší. A v případě, kdy soustava nemá žádné řešení, hledáme vektor x, který minimalizuje normu rezidua

$$\|Ax - b\| \tag{1.2}$$

a je zároveň mezi všemi minimalizujícími vektory v normě nejmenší.
Jestliže uvažujeme euklidovskou normu, tedy

$$\|x\|_2 = \sqrt{x^T x}, \tag{1.3}$$

pak můžeme řešení nalézt ve smyslu MNČ s pomocí Moore-Penroseovy inverze. Její definice je uvedena v následující kapitole.

Jestliže si danou normu zadefinujeme jako

$$\|x\|_M = \sqrt{x^T M x} \tag{1.4}$$

a

$$\|y\|_N = \sqrt{y^T N y} \tag{1.5}$$

kde $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní matice, dále $x \in \mathcal{M}_{m,1}$ a $y \in \mathcal{M}_{n,1}$. Řešení pak můžeme nalézt ve smyslu MNČ s pomocí zobecněné Moore-Penroseovy inverze neboli Chipmanovy pseudoinverze. Její definice je uvedena v následující kapitole.

2. Moore-Penroseova inverze

V této kapitole si uvedeme definice klasické a zobecněné Moore-Penroseovy inverze matic, jejich vlastnosti a sedm algoritmů pro jejich výpočet.

2.1. Klasická Moore-Penroseova inverze

Moore-Penroseova inverze neboli pseudoinverze byla poprvé představena na začátku 20. století. Jednalo se o zobecnění inverzí matic. V této podkapitole si uvedeme její definici, vlastnosti a algoritmus pro její výpočet.

Definice 2.1. Nechť $A \in \mathcal{M}_{m,n}$. Matice $A^+ \in \mathcal{M}_{n,m}$ splňující axiomy

$$AA^+A = A \tag{2.1}$$

$$A^+AA^+ = A^+ \tag{2.2}$$

$$(AA^+)^T = AA^+ \tag{2.3}$$

$$(A^+A)^T = A^+A \tag{2.4}$$

se nazývá Moore-Penroseova inverze matice A.

Poznámka 2.1. Moore-Penroseova inverze se také někdy značí jako pseudoinverze.

Poznámka 2.2. Jestliže $A \in \mathcal{M}_{n,n}$ je čtvercová regulární, pak matice k ní inverzní A^{-1} splňuje všechny axiomy z Definice 2.1. A^{-1} je tedy zároveň pseudoinverze k A.

Věta 2.1. *Pro každou matici $A \in \mathcal{M}_{m,n}$ existuje právě jedna pseudoinverze.*

Důkaz: Viz [2].

Věta 2.2. *Nechť $A \in \mathcal{M}_{m,n}$, A^+ je její Moore-Penroseova inverze a $b \in \mathcal{M}_{n,1}$. Nechť $x_0 = A^+b$. Pak pro každý vektor $x \in \mathcal{M}_{n,1}$, $x \neq x_0$ platí:*

1. $\|Ax_0 - b\|_2 < \|Ax - b\|_2$

nebo

2. $\|Ax_0 - b\|_2 = \|Ax - b\|_2$ *a* $\|x_0\|_2 < \|x\|_2$,

kde jsme využili normu (1.3). *Vektor* x_0 *pak nazýváme řešením ve smyslu nejmenších čtverců.*

Důkaz: Viz [5].

K nalezení pseudoinverze existuje mnoho algoritmů. My si zde uvedeme jeden z nich, tzv. Grevillův algoritmus, který nám popisuje následující věta.

Věta 2.3. *Nechť* $A \in \mathcal{M}_{m,n}$. *Pro* $k = 1, 2, ..., n$ *označme* a_k *jako k-tý sloupec matice* A *a* A_k *označme jako matici tvořenou prvními* k *sloupci matice* A.

1. *Je-li* $a_1 = 0$, *pak* $A_1^+ = 0$.
   Je-li $a_1 \neq 0$, *pak* $A_1^+ = (a_1^T a_1)^{-1} a_1^T$.

2. *Pro* $k = 2, ..., n$ *postupně vypočítáme*

   (a) $d_k = A_{k-1}^+ a_k$
   $c_k = a_k - A_{k-1} d_k$

   (b) *Je-li* $c_k = 0$, *pak* $b_k = (1 + d_k^T d_k)^{-1} d_k^T A_{k-1}^+$.
   Je-li $c_k \neq 0$, *pak* $b_k = c_k^+$, *kde* $c_k^+ = (c_k^T c_k)^{-1} c_k^T$.

   (c) $A_k^+ = \begin{pmatrix} A_{k-1}^+ - d_k b_k \\ b_k \end{pmatrix}$

Potom A_n^+ *je pseudoinverzní matice k* A.

Důkaz: Důkaz provedeme matematickou indukcí a postupným ověřováním axiomů (2.1)–(2.4) z Definice 2.1.
Nechť $n = 1$, tj. $A \in \mathcal{M}_{m,1}$. Platí $A_1 = A = a_1$.
Jestliže $a_1 = 0$, pak $A_1^+ = 0$ a A_1^+ splňuje axiomy (2.1)–(2.4) triviálně.

Jestliže $a_1 \neq 0$, pak $A_1^+ = (a_1^T a_1)^{-1} a_1^T$. Postupně ověříme všechny axiomy.

$$A_1 A_1^+ A_1 = a_1 (a_1^T a_1)^{-1} a_1^T a_1 = a_1 = A_1$$
$$A_1^+ A_1 A_1^+ = (a_1^T a_1)^{-1} a_1^T a_1 (a_1^T a_1)^{-1} a_1^T = (a_1^T a_1)^{-1} a_1^T = A_1^+$$
$$(A_1 A_1^+)^T = (a_1 (a_1^T a_1)^{-1} a_1^T)^T = a_1 (a_1^T a_1)^{-1} a_1^T = A_1 A_1^+$$
$$(A_1^+ A_1)^T = ((a_1^T a_1)^{-1} a_1^T a_1)^T = (1)^T = 1 = (a_1^T a_1)^{-1} a_1^T a_1 = A_1^+ A_1$$

Tedy A_1^+ je pseudoinverzní matice k A_1. Dokázali jsme, že tvrzení platí pro $n = 1$. Nyní předpokládejme, že tvrzení platí pro $n - 1$. Máme dokázat, že platí i pro n, což dokážeme opět ověřením axiomů.
Nejprve ověříme axiom (2.3), tj. $(A_n A_n^+)^T = A_n A_n^+$. Stačí tedy dokázat, že matice $A_n A_n^+$ je symetrická. Tedy

$$A_n A_n^+ = \begin{pmatrix} A_{n-1} & a_n \end{pmatrix} \begin{pmatrix} A_{n-1}^+ - d_n b_n \\ b_n \end{pmatrix} = A_{n-1} A_{n-1}^+ - A_{n-1} d_n b_n + a_n b_n =$$
$$= A_{n-1} A_{n-1}^+ + (a_n - A_{n-1} d_n) b_n = A_{n-1} A_{n-1}^+ + c_n b_n,$$

kde jsme využili rovnosti $c_k = a_k - A_{k-1} d_k$. Platí tedy

$$A_n A_n^+ = A_{n-1} A_{n-1}^+ + c_n b_n. \tag{2.5}$$

Nyní budeme zkoumat dva případy v závislosti na tom, jestli $c_n = 0$ nebo $c_n \neq 0$.

1. Pro $c_n = 0$ je

   $$A_n A_n^+ = A_{n-1} A_{n-1}^+,$$

   kde $A_{n-1} A_{n-1}^+$ je již z indukčního předpokladu symetrická matice.

2. Pro $c_n \neq 0$ platí

   $$b_n = (c_n^T c_n)^{-1} c_n^T = c_n^+ \tag{2.6}$$

   a tak

   $$A_n A_n^+ = A_{n-1} A_{n-1}^+ + c_n c_n^+,$$

   kde $c_n c_n^+$ a $A_{n-1} A_{n-1}^+$ jsou již symetrické matice, tudíž jejich součet je též

symetrická matice.

Ověřili jsme tedy, že axiom (2.3) platí pro n.

Nyní budeme ověřovat platnost axiomu (2.1), tj. $A_n A_n^+ A_n = A_n$.

S využitím vztahu (2.5) lze psát

$$\begin{aligned} A_n A_n^+ A_n = (A_n A_n^+) A_n &= \left(A_{n-1} A_{n-1}^+ + c_n b_n\right) \begin{pmatrix} A_{n-1} & a_n \end{pmatrix} = \\ &= \begin{pmatrix} A_{n-1} A_{n-1}^+ A_{n-1} + c_n b_n A_{n-1} & A_{n-1} A_{n-1}^+ a_n + c_n b_n a_n \end{pmatrix} = \\ &= \begin{pmatrix} A_{n-1} + c_n b_n A_{n-1} & A_{n-1} A_{n-1}^+ a_n + c_n b_n a_n \end{pmatrix}, \end{aligned}$$

kde jsme využili indukčního předpokladu. Dále

1. pro $c_n = 0$ platí

   $$0 = c_n = a_n - A_{n-1} d_n = a_n - A_{n-1} A_{n-1}^+ a_n,$$

   tedy

   $$a_n = A_{n-1} A_{n-1}^+ a_n, \tag{2.7}$$

   čehož následně využijeme, tj.

   $$A_n A_n^+ A_n = \begin{pmatrix} A_{n-1} & A_{n-1} A_{n-1}^+ a_n \end{pmatrix} = \begin{pmatrix} A_{n-1} & a_n \end{pmatrix} = A_n.$$

2. Pro $c_n \neq 0$ platí

   $$A_n A_n^+ A_n = \begin{pmatrix} A_{n-1} + c_n c_n^+ A_{n-1} & A_{n-1} A_{n-1}^+ a_n + c_n c_n^+ a_n \end{pmatrix}. \tag{2.8}$$

   Přitom

   $$\begin{aligned} c_n c_n^+ A_{n-1} &= c_n (c_n^T c_n)^{-1} c_n^T A_{n-1} = c_n (c_n^T c_n)^{-1} (a_n - A_{n-1} d_n)^T A_{n-1} = \\ &= c_n (c_n^T c_n)^{-1} (a_n - A_{n-1} A_{n-1}^+ a_n)^T A_{n-1} = \\ &= c_n (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+)^T A_{n-1}, \end{aligned}$$

   Protože $(I - A_{n-1} A_{n-1}^+)^T = I - A_{n-1} A_{n-1}^+$, lze psát

   $$c_n c_n^+ A_{n-1} = c_n (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+) A_{n-1} = 0, \tag{2.9}$$

neboť

$$\begin{aligned}(I - A_{n-1}A_{n-1}^{+})A_{n-1} &= A_{n-1} - A_{n-1}A_{n-1}^{+}A_{n-1} = \\ &= A_{n-1} - A_{n-1} = 0,\end{aligned} \tag{2.10}$$

kde jsme opět využili indukčního předpokladu.

Dále platí

$$c_n c_n^{+} a_n = c_n (c_n^T c_n)^{-1} c_n^T a_n = c_n (c_n^T c_n)^{-1} c_n^T c_n = c_n, \tag{2.11}$$

neboť

$$c_n^T a_n = a_n^T (I - A_{n-1}A_{n-1}^{+}) a_n$$

a

$$\begin{aligned}c_n^T c_n &= (a_n - A_{n-1}A_{n-1}^{+}a_n)^T (a_n - A_{n-1}A_{n-1}^{+}a_n) = \\ &= a_n^T (I - A_{n-1}A_{n-1}^{+})(I - A_{n-1}A_{n-1}^{+}) a_n = \\ &= a_n^T (I - A_{n-1}A_{n-1}^{+} - A_{n-1}A_{n-1}^{+} + A_{n-1}A_{n-1}^{+}A_{n-1}A_{n-1}^{+}) a_n = \\ &= a_n^T (I - A_{n-1}A_{n-1}^{+}) a_n.\end{aligned}$$

Platí tedy

$$c_n^T a_n = c_n^T c_n$$

a rovnost (2.11) je dokázána. Následně ji využijeme.

$$\begin{aligned}A_{n-1}A_{n-1}^{+}a_n + c_n c_n^{+} a_n &= A_{n-1}A_{n-1}^{+}a_n + c_n = \\ &= A_{n-1}A_{n-1}^{+}a_n + a_n - A_{n-1}A_{n-1}^{+}a_n = a_n\end{aligned} \tag{2.12}$$

Celkem tedy s využitím vztahů (2.8), (2.9) a (2.12) platí

$$\begin{aligned}A_n A_n^{+} A_n &= \begin{pmatrix} A_{n-1} + c_n c_n^{+} A_{n-1} & A_{n-1}A_{n-1}^{+}a_n + c_n c_n^{+} a_n \end{pmatrix} = \\ &= \begin{pmatrix} A_{n-1} & a_n \end{pmatrix} = A_n.\end{aligned}$$

Ověřili jsme, že axiom (2.1) platí pro n.

Dále ověříme axiom (2.2), tj. $A_n^{+} A_n A_n^{+} = A_n^{+}$.

S využitím vztahu (2.5) lze psát

$$A_n^+ A_n A_n^+ = A_n^+ (A_n A_n^+) = \begin{pmatrix} A_{n-1}^+ - d_n b_n \\ b_n \end{pmatrix} \left(A_{n-1} A_{n-1}^+ + c_n b_n \right) =$$

$$= \begin{pmatrix} A_{n-1}^+ A_{n-1} A_{n-1}^+ + A_{n-1}^+ c_n b_n - d_n b_n A_{n-1} A_{n-1}^+ - d_n b_n c_n b_n \\ b_n A_{n-1} A_{n-1}^+ + b_n c_n b_n \end{pmatrix} =$$

$$= \begin{pmatrix} A_{n-1}^+ + A_{n-1}^+ c_n b_n - d_n b_n A_{n-1} A_{n-1}^+ - d_n b_n c_n b_n \\ b_n A_{n-1} A_{n-1}^+ + b_n c_n b_n \end{pmatrix}$$

1. Pro $c_n = 0$ platí

$$b_n = (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^+. \tag{2.13}$$

   Po dosazení

$$A_n^+ A_n A_n^+ = \begin{pmatrix} A_{n-1}^+ - d_n (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^+ A_{n-1} A_{n-1}^+ \\ (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^+ A_{n-1} A_{n-1}^+ \end{pmatrix} =$$

$$= \begin{pmatrix} A_{n-1}^+ - d_n (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^+ \\ (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^+ \end{pmatrix} =$$

$$= \begin{pmatrix} A_{n-1}^+ - d_n b_n \\ b_n \end{pmatrix} = A_n^+,$$

   kde jsme využili indukčního předpokladu.

2. Pro $c_n \neq 0$ platí

$$A_n^+ A_n A_n^+ = \begin{pmatrix} A_{n-1}^+ + A_{n-1}^+ c_n c_n^+ - d_n c_n^+ A_{n-1} A_{n-1}^+ - d_n c_n^+ c_n c_n^+ \\ c_n^+ A_{n-1} A_{n-1}^+ + c_n^+ c_n c_n^+ \end{pmatrix}. \tag{2.14}$$

   Přitom

$$\begin{aligned} A_{n-1}^+ c_n c_n^+ &= A_{n-1}^+ c_n (c_n^T c_n)^{-1} c_n^T = \\ &= A_{n-1}^+ (a_n - A_{n-1} A_{n-1}^+ a_n)(c_n^T c_n)^{-1} (a_n - A_{n-1} A_{n-1}^+ a_n)^T = \\ &= A_{n-1}^+ (I - A_{n-1} A_{n-1}^+) a_n (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+)^T = \\ &= 0, \end{aligned} \tag{2.15}$$

   kde jsme využili principu z rovnosti (2.10).

Dále platí

$$\begin{aligned} d_n c_n^+ A_{n-1} A_{n-1}^+ &= d_n (c_n^T c_n)^{-1} c_n^T A_{n-1} A_{n-1}^+ = \\ &= d_n (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+) A_{n-1} A_{n-1}^+ = 0 \end{aligned} \tag{2.16}$$

a ze vztahu (2.11) vyplývá

$$c_n^+ c_n c_n^+ = c_n^+. \tag{2.17}$$

Nakonec

$$\begin{aligned} c_n^+ A_{n-1} A_{n-1}^+ &= (c_n^T c_n)^{-1} c_n^T A_{n-1} A_{n-1}^+ = \\ &= (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+) A_{n-1} A_{n-1}^+ = 0, \end{aligned} \tag{2.18}$$

kde jsme po roznásobení využili indukčního předpokladu.

Celkem tedy s využitím vztahů (2.14), (2.15), (2.16), (2.17) a (2.18) platí

$$\begin{aligned} A_n^+ A_n A_n^+ &= \begin{pmatrix} A_{n-1}^+ + A_{n-1}^+ c_n c_n^+ - d_n c_n^+ A_{n-1} A_{n-1}^+ - d_n c_n^+ c_n c_n^+ \\ c_n^+ A_{n-1} A_{n-1}^+ + c_n^+ c_n c_n^+ \end{pmatrix} = \\ &= \begin{pmatrix} A_{n-1}^+ - d_n c_n^+ \\ c_n^+ \end{pmatrix} = A_n^+. \end{aligned}$$

Ověřili jsme, že platí axiom (2.2) pro n.

Nyní ověříme platnost posledního axiomu (2.4), tj. $(A_n^+ A_n)^T = A_n^+ A_n$.

$$\begin{aligned} A_n^+ A_n &= \begin{pmatrix} A_{n-1}^+ - d_n b_n \\ b_n \end{pmatrix} \begin{pmatrix} A_{n-1} & a_n \end{pmatrix} = \\ &= \begin{pmatrix} A_{n-1}^+ A_{n-1} - d_n b_n A_{n-1} & A_{n-1}^+ a_n - d_n b_n a_n \\ b_n A_{n-1} & b_n a_n \end{pmatrix} \end{aligned}$$

Stačí dokázat, že prvky na hlavní diagonále matice jsou symetrické a oba prvky na vedlejší diagonále jsou transpozicí toho druhého.

1. Pro $c_n = 0$ platí rovnost (2.13) a

$$d_n = A_{n-1}^+ a_n = A_{n-1}^+ A_{n-1} d_n. \tag{2.19}$$

Máme dokázat, že

(a) $A_{n-1}^{+}A_{n-1} - d_n b_n A_{n-1}$ je symetrická matice.

Platí

$$\begin{aligned} d_n b_n A_{n-1} &= d_n(1 + d_n^T d_n)^{-1} d_n^T (A_{n-1}^{+} A_{n-1})^T = \\ &= (1 + d_n^T d_n)^{-1} d_n d_n^T A_{n-1}^T A_{n-1}^{+T} = \\ &= (1 + d_n^T d_n)^{-1} d_n d_n^T, \end{aligned}$$

což už je symetrická matice. Celý součet

$$A_{n-1}^{+}A_{n-1} - d_n b_n A_{n-1}$$

je pak též symetrická matice.

(b) platí $A_{n-1}^{+}a_n - d_n b_n a_n = (b_n A_{n-1})^T$. Máme tedy dokázat, že

$$A_{n-1}^{+}a_n - d_n b_n a_n - A_{n-1}^T b_n^T = 0.$$

Platí

$$\begin{aligned} d_n b_n a_n + A_{n-1}^T b_n^T &= (1 + d_n^T d_n)^{-1}(d_n d_n^T A_{n-1}^{+} a_n + A_{n-1}^T A_{n-1}^{+T} d_n) = \\ &= (1 + d_n^T d_n)^{-1}(d_n d_n^T d_n + d_n) = \\ &= d_n(d_n^T d_n + 1)^{-1}(d_n^T d_n + 1) = d_n, \end{aligned}$$

kde jsme využili rovností (2.13) a (2.19). Celkem tedy

$$A_{n-1}^{+}a_n - d_n b_n a_n - A_{n-1}^T b_n^T = d_n - d_n = 0,$$

což jsme chtěli dokázat.

(c) $b_n a_n$ je symetrická matice.

Po využití (2.13) a (2.19) platí

$$b_n a_n = (1 + d_n^T d_n)^{-1} d_n^T A_{n-1}^{+} a_n = (1 + d_n^T d_n)^{-1} d_n^T d_n,$$

což už je symetrická matice.

2. Pro $c_n \neq 0$ máme dokázat, že

(a) $A_{n-1}^{+}A_{n-1} - d_n b_n A_{n-1}$ je symetrická matice.

Platí

$$d_n b_n A_{n-1} = d_n c_n^+ A_{n-1} = d_n (c_n^T c_n)^{-1} (a_n - A_{n-1} A_{n-1}^+ a_n)^T A_{n-1} =$$
$$= d_n (c_n^T c_n)^{-1} a_n^T (I - A_{n-1} A_{n-1}^+) A_{n-1} = 0,$$

kde jsme využili indukčního předpokladu. Platí tedy

$$A_{n-1}^+ A_{n-1} - d_n b_n A_{n-1} = A_{n-1}^+ A_{n-1},$$

což už je symetrická matice.

(b) $A_{n-1}^+ a_n - d_n b_n a_n = (b_n A_{n-1})^T$. Tedy

$$A_{n-1}^+ a_n - d_n c_n^+ a_n - A_{n-1}^T c_n^{+T} = 0.$$

Podle vztahu (2.11) platí

$$d_n c_n^+ a_n = d_n, \tag{2.20}$$

který funguje na stejném principu. Dále platí

$$\begin{aligned} A_{n-1}^T c_n^{+T} &= A_{n-1}^T (c_n^T c_n)^{-1} (a_n - A_{n-1} A_{n-1}^+ a_n) = \\ &= (c_n^T c_n)^{-1} A_{n-1}^T (I - (A_{n-1} A_{n-1}^+)^T) a_n = 0, \end{aligned} \tag{2.21}$$

kde jsme opět využili předpokladu. Celkem tedy s využitím vztahů (2.20) a (2.21) platí

$$A_{n-1}^+ a_n - d_n c_n^+ a_n - A_{n-1}^T c_n^{+T} = d_n - d_n = 0,$$

což jsme se snažili dokázat.

(c) $b_n a_n$ je symetrická matice.
Podle (2.11) platí

$$b_n a_n = c_n^+ a_n = 1,$$

což je symetrická matice.

Dokázali jsme, že matice

$$A_n^+ A_n = \begin{pmatrix} A_{n-1}^+ A_{n-1} - d_n b_n A_{n-1} & A_{n-1}^+ a_n - d_n b_n a_n \\ b_n A_{n-1} & b_n a_n \end{pmatrix}$$

je symetrická a tudíž axiom (2.4) platí pro n.

Ověřili jsme, že pro n platí všechny axiomy (2.1)–(2.4) z Definice 2.1. Tím jsme dokázali, že matice A_n^+ vytvořená pomocí algoritmu z Věty 2.1 je pseudoinverzní k matici A. □

Kód 2.1. Pro Grevillův algoritmus jsem vytvořil následující kód v MATLABu:

```
%na vstupu: A - daná matice
%           presnost - přesnost algoritmu
%na výstupu: vysledek - pseudoinverze matice A

function[vysledek]=grevil(A,presnost)
if nargin < 2
    presnost=1e-6;
end;
[r,s]=size(A);
A1=A(1:end,1);
Aplus=(1/(A1'*A1))*A1';
for i=2:s
    a=A(1:end,i);
    A1=A(1:end,1:i-1);
    d=Aplus*a;
    c=a-A1*d;
    for j=1:r
        if abs(c(j))<presnost
            c(j)=0;
        end;
    end;
    if c==0
        b=(1/(1+d'*d))*d'*Aplus;
    else
        b=(1/(c'*c))*c';
    end;
```

```
    Aplus=[Aplus-(d*b);b];
end;
vysledek=Aplus;
```

Jelikož v algoritmu postupně ověřujeme, zdali se vektor c_k rovná nule, musíme hodně malé hodnoty daného vektoru blízké nule zaokrouhlit. K tomu slouží přesnost na vstupu. Zanedbáním přesnosti by mohlo vést k astronomickým hodnotám výsledků, které by navíc ani nebyly správné (viz Příklad 2.2). Jestliže uživatel přesnost na vstup nezadá, přiřadí se jí automaticky hodnota 1e-6.

Pro výpočet pseudoinverzní matice lze použít v prostředí MATLABu i příkaz pinv(A,presnost), kde A je původní matice a presnost je největší možná absolutní hodnota, kterou již považujeme za nulu. Příkaz pinv využívá singulárního rozkladu matice.

Příklad 2.1. Pomocí Grevillova algoritmu vypočítejte pseudoinverzi matice

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}.$$

Jelikož $a_1 \neq 0$, pak

$$A_1^+ = (a_1^T a_1)^{-1} a_1^T = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}.$$

Dále

$$d_2 = A_1^+ a_2 = \frac{1}{3},$$

$$c_2 = a_2 - A_1 d_2 = \frac{2}{3}\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}.$$

Jelikož $c_2 \neq 0$, pak

$$b_2 = (c_2^T c_2)^{-1} c_2^T = \frac{1}{4}\begin{pmatrix} 1 & -2 & 1 \end{pmatrix},$$

$$A_2^+ = \begin{pmatrix} A_1^+ - d_2 b_2 \\ b_2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 1 & -2 & 1 \end{pmatrix},$$

$$d_3 = A_2^+ a_3 = \frac{1}{4}\begin{pmatrix} 3 \\ -1 \end{pmatrix},$$

$$c_3 = a_3 - A_2 d_3 = \frac{3}{2}\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

Jelikož $c_3 \neq 0$, pak

$$b_3 = (c_3^T c_3)^{-1} c_3^T = \frac{1}{3}\begin{pmatrix} 1 & 0 & -1 \end{pmatrix},$$

$$A_3^+ = \begin{pmatrix} A_2^+ - d_3 b_3 \\ b_3 \end{pmatrix} = \frac{1}{6}\begin{pmatrix} 0 & 3 & 0 \\ 2 & -3 & 1 \\ 2 & 0 & -2 \end{pmatrix}.$$

A_3^+ je hledaná pseudoinverzní matice k matici A, což můžeme snadno ověřit pomocí axiomů z Definice 2.1.

Zadáním příkazu grevil([1 1 2;1 -1 1;1 1 -1]) nebo pinv([1 1 2;1 -1 1;1 1 -1]) do prostředí MATLABu dostaneme stejný výsledek.

Příklad 2.2. Vypočítejte pomocí MATLABu pseudoinverzi matice

$$\mathbf{A} = \begin{pmatrix} 5 & 8 & 1 & 2 \\ 6 & 4 & 9 & 2 \\ 11 & 8 & 4 & 5 \end{pmatrix}$$

Do prostředí MATLABu zadáme příkaz grevil([5 8 1 2;6 4 9 2;11 8 4 5],1e-6), čímž dostaneme matici

$$\mathbf{A}^+ = \begin{pmatrix} -0.1240 & -0.0490 & 0.1529 \\ 0.2242 & 0.0240 & -0.1118 \\ 0.0026 & 0.1431 & -0.0750 \\ -0.0879 & -0.0451 & 0.1026 \end{pmatrix}.$$

A^+ je hledaná pseudoinverzní matice k matici A, což můžeme snadno ověřit pomocí axiomů z Definice 2.1.

Použitím příkazu pinv([5 8 1 2;6 4 9 2;11 8 4 5]) dostaneme stejný výsledek.

Vynecháním zaokrouhlování v kódu dostaneme po využití příkazu

grevil([5 8 1 2;6 4 9 2;11 8 4 5]) matici

$$\mathbf{A}^{+} = \begin{pmatrix} -6.6819 \times 10^{13} & -0.0741 & 1.1879 \times 10^{14} \\ 1.0023 \times 10^{13} & 0.0278 & -1.7818 \times 10^{13} \\ 1.3364 \times 10^{13} & 0.1481 & -2.3758 \times 10^{13} \\ 1.2027 \times 10^{14} & 0 & -2.1382 \times 10^{14} \end{pmatrix},$$

což rozhodně není správný výsledek. Přesnost na vstupu tedy není zanedbatelná veličina.

2.2. Zobecněná Moore-Penroseova inverze

Zobecnění Moore-Penroseovy inverze bylo poprvé zveřejněno v článku [1] v polovině 20. století. Toto zobecnění se také někdy nazývá Chipmanova Pseudoinverze podle autora daného článku. V této podkapitole si uvedeme její definici, vlastnosti a metody pro její výpočet.

Definice 2.2. Nechť $A \in \mathcal{M}_{m,n}$, $M \in \mathcal{M}_{m,m}$ je symetrická pozitivně definitní a $N \in \mathcal{M}_{n,n}$ je též symetrická pozitivně definitní. Pak se matice $A^{+}_{M,N}$ splňující axiomy

$$AA^{+}_{M,N}A = A \tag{2.22}$$

$$A^{+}_{M,N}AA^{+}_{M,N} = A^{+}_{M,N} \tag{2.23}$$

$$(MAA^{+}_{M,N})^{T} = MAA^{+}_{M,N} \tag{2.24}$$

$$(NA^{+}_{M,N}A)^{T} = NA^{+}_{M,N}A \tag{2.25}$$

nazývá Chipmanova pseudoinverze matice A.

Věta 2.4. *Pro Chipmanovu pseudoinverzi $A^{+}_{M,N}$ platí následující vlastnosti:*

1. *Jestliže $M = N = I$, pak $A^{+}_{M,N} = A^{+}$.*
2. *Jestliže A je čtvercová regulární matice, pak $A^{+}_{M,N} = A^{-1}$.*
3. $(A^{+}_{M,N})^{+}_{N,M} = A$.
4. $(A^{+}_{M,N})^{T} = (A^{T})^{+}_{N^{-1},M^{-1}}$

Důkaz: Plyne z Definice 2.2.

Jestliže je matice plné sloupcové nebo řádkové hodnosti, pak můžeme její Chipmanovu pseudoinverzi vypočítat použitím pouze jedné symetrické pozitivně definitní matice.

Věta 2.5. *Nechť $B \in \mathcal{M}_{m,r}$, $h(B) = r$ a $M \in \mathcal{M}_{m,m}$ je symetrická pozitivně definitní. Pak matice*

$$B^+_{M,.} = (B^T M B)^{-1} B^T M \tag{2.26}$$

je Chipmanova pseudoinverze matice B odpovídající matici M a libovolné symetrické matici $N \in \mathcal{M}_{r,r}$.

Důkaz: Aby byla matice $B^+_{M,.}$ Chipmanovou pseudoinverzí, pak musí splňovat axiomy (2.22)–(2.25) z Definice 2.2. Postupně je tedy všechny ověříme. Nejprve se však podíváme, čemu se rovná $B^+_{M,.}B$.

$$B^+_{M,.}B = (B^T M B)^{-1} B^T M B = I \tag{2.27}$$

Rovnost (2.27) nyní využijeme k ověření daných axiomů.

$$\begin{aligned}
BB^+_{M,.}B &= B(B^+_{M,.}B) = B \\
B^+_{M,.}BB^+_{M,.} &= (B^+_{M,.}B)B^+_{M,.} = B^+_{M,.} \\
(MBB^+_{M,.})^T &= (MB(B^TMB)^{-1}B^TM)^T = M^TB((B^TMB)^T)^{-1}B^TM^T = \\
&= MB(B^TMB)^{-1}B^TM = MBB^+_{M,.}
\end{aligned}$$

Protože platí (2.27), pak je axiom (2.25) splněn triviálně.
Matice $B^+_{M,.}$ splňuje všechny axiomy, čímž jsme dokázali, že je Chipmanovou pseudoinverzí matice B. □

Věta 2.6. *Nechť $C \in \mathcal{M}_{r,n}$, $h(C) = r$ a $N \in \mathcal{M}_{n,n}$ je symetrická pozitivně definitní. Pak matice*

$$C^+_{.,N} = N^{-1}C^T(CN^{-1}C^T)^{-1} \tag{2.28}$$

je Chipmanova pseudoinverze matice C odpovídající matici N a libovolné symetrické matici $M \in \mathcal{M}_{r,r}$.

Důkaz: Aby byla matice $C^+_{.,N}$ Chipmanovou pseudoinverzí, pak musí splňovat axiomy (2.22)–(2.25) z Definice 2.2. Postupně je tedy všechny ověříme. Nejprve se však podíváme, čemu se rovná $CC^+_{.,N}$.

$$CC^+_{.,N} = CN^{-1}C^T(CN^{-1}C^T)^{-1} = I \tag{2.29}$$

Rovnost (2.29) nyní využijeme k ověření daných axiomů.

$$\begin{aligned} CC^+_{.,N}C &= (CC^+_{.,N})C = C \\ C^+_{.,N}CC^+_{.,N} &= C^+_{.,N}(CC^+_{.,N}) = C^+_{.,N} \\ (NC^+_{.,N}C)^T &= (NN^{-1}C^T(CN^{-1}C^T)^{-1}C)^T = C^T(CN^{-1}C^T)^{-1}C = \\ &= NN^{-1}C^T(CN^{-1}C^T)^{-1}C = NC^+_{.,N}C \end{aligned}$$

Protože platí (2.29), pak je axiom (2.24) splněn triviálně.
Matice $C^+_{.,N}$ splňuje všechny axiomy, čímž jsme dokázali, že je Chipmanovou pseudoinverzí matice C. □

Věta 2.7. *Nechť $A \in \mathcal{M}_{m,n}$, $A^+_{M,N}$ je její Chipmanova pseudoinverze a $b \in \mathcal{M}_{n,1}$. Nechť $x_0 = A^+_{M,N}b$. Pak pro každý vektor $x \in \mathcal{M}_{n,1}$, $x \neq x_0$ platí:*

1. $\|Ax_0 - b\|_M < \|Ax - b\|_M$

nebo

2. $\|Ax_0 - b\|_M = \|Ax - b\|_M$ *a* $\|x_0\|_N < \|x\|_N$,

kde jsme využili norem (1.4) a (1.5). Vektor x_0 pak nazýváme řešením ve smyslu nejmenších čtverců.

Důkaz: Viz [5].

Některé z následujících iteračních algoritmů využívají skeletní rozklad, který si uvedeme v následující větě.

Věta 2.8. *(Skeletní rozklad) Nechť $A \in \mathcal{M}_{m,n}$, $h(A) = r$. Dále nechť $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní. Pak existují $B \in \mathcal{M}_{m,r}$ a $C \in \mathcal{M}_{r,n}$, pro které platí*

$$A = BC,$$

$r = h(B) = h(C)$. Pro jejich Chipmanovy pseudoinverze navíc platí

$$A^+_{M,N} = C^+_{M,N} B^+_{M,N}.$$

Důkaz: Viz [4].

Kód 2.2. Pro výpočet skeletního rozkladu jsem vytvořil následující kód v MATLABu:

```
function[C,D]=skelet(A)
%na vstupu: A - daná matice
%na výstupu: C - hledaná matice m x r
%            D - hledaná matice r x n
function[C,D]=skelet(A)
[m,n]=size(A);
B=A(:,1);
h1=rank(B);
j=2;
for i=2:n
    B(:,j)=A(:,i);
    h2=rank(B);
    if h2==h1+1
        j=j+1;
        h1=h2;
    else
        B(:,j)=[];
    end;
end;
C=B
D=B\A
```

Dle Vět 2.5, 2.6 a 2.8 můžeme pomocí skeletního rozkladu rovnou vypočítat Chipmanovu pseudoinverzi pro libovolnou matici $A \in \mathcal{M}_{m,n}$.

Kód 2.3. Pro výpočet Chipmanovy pseudoinverze pomocí skeletního rozkladu jsem vytvořil následující kód v MATLABu:

```
function[vysledek]=pseudoskelet(A,M,N)
%na vstupu: A - daná matice
%           M,N - symetrické pozitivně definitní matice
%na výstupu: vysledek - hledaná matice
[m,n]=size(A);
[m1,m2]=size(M);
if norm(M-M')~=0 | m1~=m2 | m1~=m | sum(eig(M)>0)~=m
   error(['M musi byt symetricka pozitivne ', ...
        'definitni matice typu m x m'])
end;
[n1,n2]=size(N);
if norm(N-N')~=0 | n1~=n2 | n1~=n | sum(eig(N)>0)~=n
   error(['N musi byt symetricka pozitivne ', ...
        'definitni matice typu n x n'])
end;
B=A(:,1);
h1=rank(B);
j=2;
plk=1
for i=2:n
    B(:,j)=A(:,i);
    h2=rank(B);
    if h2==h1+1
        j=j+1;
        h1=h2;
    else
        B(:,j)=[];
    end;
end;
C=B;
D=B\A;
BM=inv(C'*M*C)*C'*M;
CN=inv(N)*D'*inv(D*inv(N)*D');
AMN=CN*BM;
```

```
vysledek=AMN;
```

Výpočet Chipmanovy pseudoinverze pomocí skeletního rozkladu je pro menší matice postačující. U větších matic však dochází k nepřesnostem, což je způsobeno vysokým počtem inverzí matic ve výpočtu.

Poznámka 2.3. K ověření pozitivní definitnosti matice využíváme její vlastní čísla, tj. testujeme, zdali všechna vlastní čísla matice jsou kladná. Na výpočet vlastních čísel využíváme v MATLABu příkaz eig. Existuje mnoho způsobů, jak tato čísla vypočítat. V této bakalářské práci se však touto problematikou podrobněji zabývat nebudeme a na výpočet nám tento příkaz postačí.

Věta 2.9. *Pro každou matici $A \in \mathcal{M}_{m,n}$ a pozitivně definitní matice $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ existuje právě jedna Chipmanova pseudoinverzní matice $A^{+}_{M,N}$.*

Důkaz: Viz [4].

Pro výpočet Chipmanovy pseudoinverze opět existuje mnoho algoritmů. My si zde uvedeme několik z nich. Následující tři algoritmy volíme podle toho, zdali původní matice $A \in \mathcal{M}_{m,n}$ má plnou sloupcovou hodnost, plnou řádkovou hodnost nebo je-li obecná. Důkazy uvedených vět lze najít v [3].

Věta 2.10. *Nechť $B \in \mathcal{M}_{m,r}$ s plnou sloupcovou hodností, tj. platí $h(B) = r$, přičemž $r \geq 2$. Dále nechť $M \in \mathcal{M}_{m,m}$ je symetrická pozitivně definitní a q je libovolné celé číslo, pro které platí $q \geq 2$.*

1. *Vypočtěme*

   (a) $V_B = B^T M B$

   (b) $\alpha = \dfrac{2}{tr(V_B^T V_B)}$

   (c) $Y_0 = \alpha V_B^T$

2. *Dále pro k=0,1,2,... postupně vypočítáme*

   (a) $T_k = I - Y_k V_B$

(b) $Y_{k+1} = (I + T_k + T_k^2 + \cdots + T_k^{q-1})Y_k$

(c) $X_{k+1} = Y_{k+1}B^T M$

Pak $\lim_{k\to\infty} X_k = B_M^+$*, kde* $B_{M,\cdot}^+$ *je hledaná Chipmanova pseudoinverze.*

Nutno podotknout, že v následujícím kódu nepočítáme s limitou do nekonečna, jak je uvedeno ve Větě 2.10, ale pouze s limitou do konečného čísla k, pro které je již matice X_k dostatečně blízko přesnému výsledku.

Kód 2.4. Pro iterační algoritmus z uvedené Věty 2.10 jsem vytvořil následující kód v MATLABu:

```
function[vysledek,iterace]=iterative(B,M,Q,presnost,maxit)
%na vstupu: B - daná matice
%           M - symetrická pozitivně definitní matice
%           Q - celé číslo
%           presnost - přesnost algoritmu
%           maxit - maximální počet iterací
%na výstupu: vysledek - hledaná matice
%            iterace - konečný počet iterací
if nargin < 5
    maxit=5000;
    if nargin < 4
        presnost=1e-6;
    end;
end;
if Q<2 || rem(Q,1)~=0
   error('Celé číslo Q musí být větší nebo rovno 2.')
end;
[m,r]=size(B);
if rank(B)<2 || rank(B)~=r
   error(['Matice B typu m x r musi mit ', ...
        'plnou sloupcovou hodnost r>=2.'])
end;
[m1,m2]=size(M);
if norm(M-M')~=0 | m1~=m2 | m1~=m | sum(eig(M)>0)~=m
```

```
    error(['M musi byt symetricka pozitivne ', ...
        'definitni matice typu m x m'])
end;
k=0;
t=maxit;
Vb=B'*M*B;
alfa=2/trace(Vb'*Vb);
I=eye(r);
Yk=alfa*Vb';
while k<maxit
    Tk=I-Yk*Vb;
    q=0;
    soucet=0;
    while q<Q
        soucet=soucet+(Tk)^q;
        q=q+1;
    end;
    Yk=soucet*Yk;
    k=k+1;
    Xk=Yk*B'*M;
    n1=norm(B*Xk*B-B);
    n2=norm(Xk*B*Xk-Xk);
    n3=norm(M*B*Xk-(M*B*Xk)');
    n4=norm(Xk*B-(Xk*B)');
    if n1<=presnost && n2<=presnost ...
       && n3<=presnost && n4<=presnost
       t=k;
       k=maxit;
    end;
end;
iterace=t;
vysledek=Xk;
```

Jak již bylo zmíněno, jelikož počítáme jen s limitou do konečného čísla k, pak nám daný algoritmus v prostředí MATLABu udává pouze přibližný výsledek. Proto na vstupu musíme uvést přesnost, čímž je u daného Kódu 2.4 myšleno

takové číslo ε, pro jehož hodnotu platí

$$\begin{aligned} \|AXA - A\| &\leq \varepsilon \\ \|XAX - X\| &\leq \varepsilon \\ \|(MAX)^T - MAX\| &\leq \varepsilon \\ \|(XA)^T - XA\| &\leq \varepsilon \end{aligned}$$

kde A je původní matice, X je k ní Chipmanova pseudoinverzní matice, M je příslušná symetrická pozitivně definitní matice a $\|.\|$ je euklidovská norma. Tímto způsobem ověřujeme, zdali vypočítaná matice je skutečně Chipmanova pseudoinverzní k původní matici a splňuje axiomy z Definice 2.2.

Jak je vidět, pro ověření axiomu (2.25) ještě potřebujeme matici N. Dle Věty 2.5 je však $B^+_{M,\cdot}$ Chipmanovou pseudoinverzí odpovídající zadané matici M a libovolné symetrické matici N. Pro ověření daného axiomu nám tedy namísto matice N postačí jednotková matice příslušných rozměrů I.

Dále je třeba uvést maximální počet iterací pro případ, kdy je hodnota ε příliš malá a algoritmus se nebude chtít zastavit.

Jestliže uživatel maximální počet iterací na vstup nezadá, přiřadí se mu automaticky hodnota 5000. V případě, že uživatel nezadá ani přesnost, přiřadí se jí hodnota $1e-6$.

Věta 2.11. *Nechť $C \in \mathcal{M}_{r,n}$ s plnou řádkovou hodností, tj. platí $h(C) = r$, přičemž $r \geq 2$. Dále nechť $N \in \mathcal{M}_{n,n}$ je symetrická pozitivně definitní a q je libovolné celé číslo, pro které platí $q \geq 2$.*

1. *Vypočtěme*

   (a) $V_C = CN^{-1}C^T$

   (b) $\alpha = \dfrac{2}{tr(V_C^T V_C)}$

   (c) $Y_0 = \alpha V_C^T$

2. *Dále pro k=0,1,2,... postupně vypočítáme*

(a) $T_k = I - Y_k V_C$

(b) $Y_{k+1} = (I + T_k + T_k^2 + \cdots + T_k^{q-1})Y_k$

(c) $X_{k+1} = N^{-1}C^T Y_{k+1}$

Pak $\lim_{k\to\infty} X_k = C_N^+$, *kde* $C_{.,N}^+$ *je hledaná Chipmanova pseudoinverze.*

V následujícím kódu opět počítáme jen s limitou do konečného čísla k, pro které je již matice X_k dostatečně blízko přesnému výsledku.

Kód 2.5. Pro iterační algoritmus z uvedené Věty 2.11 jsem vytvořil následující kód v MATLABu:

```
function[vysledek,iterace]=iterative2(C,N,Q,presnost,maxit)
%na vstupu: C - daná matice
%           N - symetrická pozitivně definitní matice
%           Q - celé číslo
%           presnost - přesnost algoritmu
%           maxit - maximální počet iterací
%na výstupu: vysledek - hledaná matice
%            iterace - konečný počet iterací
if nargin < 5
    maxit=5000;
    if nargin < 4
        presnost=1e-6;
    end;
end;
if Q<2 || rem(Q,1)~=0
   error('Celé číslo Q musí být větší nebo rovno 2.')
end;
[r,n]=size(C);
if rank(C)<2 || rank(C)~=r
   error(['Matice C typu r x n musi mit ', ...
        'plnou sloupcovou hodnost r>=2.'])
end;
[n1,n2]=size(N);
if norm(N-N')~=0 | n1~=n2 | n1~=n | sum(eig(N)>0)~=n
```

```
   error(['N musi byt symetricka pozitivne ', ...
        'definitni matice typu n x n'])
end;
k=0;
t=maxit;
Vc=C*inv(N)*C';
alfa=2/trace(Vc'*Vc);
I=eye(r);
Yk=alfa*Vc';
while k<maxit
    Tk=I-Yk*Vc;
    q=0;
    soucet=0;
    while q<Q
        soucet=soucet+(Tk)^q;
        q=q+1;
    end;
    Yk=soucet*Yk;
    k=k+1;
    Xk=inv(N)*C'*Yk;
n1=norm(C*Xk*C-C);
    n2=norm(Xk*C*Xk-Xk);
    n3=norm(C*Xk-(C*Xk)');
    n4=norm(N*Xk*C-(N*Xk*C)');
    if n1<=presnost && n2<=presnost ...
       && n3<=presnost && n4<=presnost
        t=k;
        k=maxit;
    end;
end;
iterace=t;
vysledek=Xk;
```

Daný algoritmus nám díky limitě v prostředí MATLABu opět udává pouze přibližný výsledek. Přesností na vstupu je tentokrát myšleno takové číslo ε, pro

jehož hodnotu platí

$$\|AXA - A\| \leq \varepsilon$$
$$\|XAX - X\| \leq \varepsilon$$
$$\|(AX)^T - AX\| \leq \varepsilon$$
$$\|(NXA)^T - NXA\| \leq \varepsilon$$

kde A je původní matice, X je k ní Chipmanova pseudoinverzní matice, N je příslušná symetrická pozitivně definitní matice a $\|.\|$ je euklidovská norma.

Pro ověření axiomu (2.24) ještě potřebujeme matici M. Dle Věty 2.6 je však $C^+_{.,N}$ Chipmanovou pseudoinverzí odpovídající zadané matici N a libovolné symetrické matici M. Pro ověření daného axiomu nám tedy postačí příslušná jednotková matice I.

V případě nezadání maximálního počtu iterací a přesnosti na vstupu se jim přiřadí výchozí hodnoty analogicky jako u Kódu 2.4.

Věta 2.12. *Nechť $A \in \mathcal{M}_{m,n}$, $h(A) = r$, přičemž $r \geq 2$. Dále nechť $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní a q je libovolné celé číslo, pro které platí $q \geq 2$. Dále nechť pomocí skeletního rozkladu A vzniknou $B \in \mathcal{M}_{m,r}$ a $C \in \mathcal{M}_{r,n}$, tj. platí $A = BC$, přičemž $h(A) = h(B) = h(C) = r$.*

1. *Vypočtěme*
   (a) $V_A = B^T M A N^{-1} C^T$
   (b) $\alpha = \dfrac{2}{tr(V_A^T V_A)}$
   (c) $Y_0 = \alpha V_A^T$
2. *Dále pro k=0,1,2,... postupně vypočítáme*
   (a) $T_k = I - Y_k V_A$
   (b) $Y_{k+1} = (I + T_k + T_k^2 + \cdots + T_k^{q-1}) Y_k$
   (c) $X_{k+1} = N^{-1} C^T Y_{k+1} B^T M$

Pak $\lim_{k\to\infty} X_k = A^+_{M,N}$*, kde* $A^+_{M,N}$ *je hledaná Chipmanova pseudoinverze.*

Pro následující kód využijeme skeletního rozkladu z Kódu 2.2. Opět počítáme jen s limitou do konečného čísla k, pro které je již matice X_k dostatečně blízko přesnému výsledku.

Kód 2.6. Pro iterační algoritmus z uvedené Věty 2.12 jsem vytvořil následující kód v MATLABu:

```
function[vysledek,iterace]=iterative3(A,M,N,Q,presnost,maxit)
%na vstupu: A - daná matice
%           M,N - symetrické pozitivně definitní matice
%           Q - celé číslo
%           presnost - přesnost algoritmu
%           maxit - maximální počet iterací
%na výstupu: vysledek - hledaná matice
%            iterace - konečný počet iterací
if nargin < 6
    maxit=5000;
    if nargin < 5
        presnost=1e-6;
    end;
end;
if Q<2 || rem(Q,1)~=0
   error('Celé číslo Q musí být větší nebo rovno 2.')
end;
[m,n]=size(A);
r=rank(A);
if r<2
   error('Matice A musí mít hodnost alespoň 2.')
end;
[m1,m2]=size(M);
if norm(M-M')~=0 | m1~=m2 | m1~=m | sum(eig(M)>0)~=m
   error(['M musi byt symetricka pozitivne ', ...
        'definitni matice typu m x m'])
end;
[n1,n2]=size(N);
```

```
if norm(N-N')~=0 | n1~=n2 | n1~=n | sum(eig(N)>0)~=n
   error(['N musi byt symetricka pozitivne ', ...
        'definitni matice typu n x n'])
end;
[B,C]=skelet(A);
k=0;
t=maxit;
Va=B'*M*A*inv(N)*C';
alfa=2/trace(Va'*Va);
I=eye(r);
Yk=alfa*Va';
while k<maxit
    Tk=I-Yk*Va;
    q=0;
    soucet=0;
    while q<Q
        soucet=soucet+(Tk)^q;
        q=q+1;
    end
    Yk=soucet*Yk;
    k=k+1;
    Xk=inv(N)*C'*Yk*B'*M;
n1=norm(A*Xk*A-A);
    n2=norm(Xk*A*Xk-Xk);
    n3=norm(M*A*Xk-(M*A*Xk)');
    n4=norm(N*Xk*A-(N*Xk*A)');
    if n1<=presnost && n2<=presnost ...
       && n3<=presnost && n4<=presnost
        t=k;
        k=maxit;
    end;
end;
iterace=t;
vysledek=Xk;
```

Daný algoritmus nám díky limitě v prostředí MATLABu opět udává pouze přibližný výsledek. Proto na vstupu musíme uvést přesnost, čímž je u daného

iteračního algoritmu myšleno takové číslo ε, pro jehož hodnotu platí

$$\|AXA - A\| \leq \varepsilon$$
$$\|XAX - X\| \leq \varepsilon$$
$$\|(MAX)^T - AX\| \leq \varepsilon$$
$$\|(NXA)^T - NXA\| \leq \varepsilon$$

kde A je původní matice, X je k ní Chipmanova pseudoinverzní matice, M a N jsou příslušné symetrické pozitivně definitní matice a $\|.\|$ je euklidovská norma.

V případě nezadání maximálního počtu iterací a přesnosti na vstupu se jim přiřadí výchozí hodnoty analogicky jako u Kódu 2.4.

Jestliže má matice A hodnost pouze 1, tak se nám situace výrazně zjednodušuje, jak nám ukazuje následující Věta 2.13.

Věta 2.13. *Nechť $A \in \mathcal{M}_{m,n}$, $h(A) = 1$. Dále nechť $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní. Nechť pomocí skeletního rozkladu A vzniknou $B \in \mathcal{M}_{m,r}$ a $C \in \mathcal{M}_{r,n}$, tj. platí $A = BC$, přičemž $h(A) = h(B) = h(C) = 1$. Dále nechť $V_A = B^T M A N^{-1} C^T$. Vypočítáme*

$$A^+_{M,N} = \frac{1}{tr(V_A^T V_A)} N^{-1} C^T V_A^T B^T M,$$

kde $A^+_{M,N}$ je hledaná Chipmanova pseudoinverze.

Tento algoritmus se však v praxi moc neuplatní, protože požadavek na původní matici je příliš přísný. Je dobré si uvědomit, že v daném algoritmu již nepočítáme s limitou.

Kód 2.7. Pro výpočet Chipmanovy pseudoinverze použitím Věty 2.13 jsem vytvořil následující kód v MATLABu:

```
function[vysledek]=rank1(A,M,N)
%na vstupu: A - daná matice
%           M,N - symetrické pozitivně definitní matice
%na výstupu: vysledek - hledaná matice
```

```
[m,n]=size(A);
r=rank(A);
if r~=1
    error('Matice A musí mít hodnost 1.')
end;
[m1,m2]=size(M);
if norm(M-M')~=0 | m1~=m2 | m1~=m | sum(eig(M)>0)~=m
   error(['M musi byt symetricka pozitivne ', ...
        'definitni matice typu m x m'])
end;
[n1,n2]=size(N);
if norm(N-N')~=0 | n1~=n2 | n1~=n | sum(eig(N)>0)~=n
   error(['N musi byt symetricka pozitivne ', ...
        'definitni matice typu n x n'])
end;
[B,C]=skelet(A);
Va=B'*M*A*inv(N)*C';
Aplus=(inv(N)*C'*Va'*B'*M)/trace(Va'*Va);
vysledek=Aplus;
```

Nyní si uvedeme novou metodu pro výpočet Chipmanovy pseudoinverze, která byla zveřejněna v [6] v roce 2013.

Věta 2.14. *Nechť $A \in \mathcal{M}_{m,n}$. Dále nechť $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní a $I \in \mathcal{M}_{m,m}$ je jednotková matice. Nechť β je reálné kladné číslo, pro které platí*

$$0 < \beta < \frac{2}{\sigma_1^2},$$

kde σ_1 je největší vlastní číslo matice $N^{-1}A^TMA \in \mathcal{M}_{n,n}$.

1. *Vypočtěme*
   (a) $X_0 = \beta N^{-1}A^TM$
2. *Dále pro k=0,1,2,... postupně vypočítáme*
   (a) $\psi_k = AX_k$

(b) $\xi_k = -7I + \psi_k(9I + \psi_k(-5I + \psi_k))$

(c) $\omega_k = \psi_k \xi_k$

(d) $X_{k+1} = -\frac{1}{16} X_k \xi_k (4I + \omega_k)(8I + \omega_k(4I + \omega_k))$

Pak $\lim_{k\to\infty} X_k = A^+_{M,N}$*, kde* $A^+_{M,N}$ *je hledaná Chipmanova pseudoinverze.*

Důkaz: Viz [6].

Opět počítáme jen s limitou do konečného čísla k, pro které je již matice X_k dostatečně blízko přesnému výsledku.

Kód 2.8. Pro výpočet Chipmanovy pseudoinverze použitím Věty 2.14 jsem vytvořil následující kód v MATLABu:

```
function[vysledek,iterace]=new(A,M,N,beta,presnost,maxit)
%na vstupu: A - daná matice
%           M,N - symetrická pozitivně definitní matice
%           beta - konstanta
%           presnost - přesnost algoritmu
%           maxit - maximální počet iterací
%na výstupu: vysledek - hledaná matice
%            iterace - konečný počet iterací
if nargin < 6
    maxit=5000;
    if nargin < 5
        presnost=1e-6;
    end;
end;
[m,n]=size(A);
r=rank(A);
[m1,m2]=size(M);
if norm(M-M')~=0 | m1~=m2 | m1~=m | sum(eig(M)>0)~=m
   error(['M musi byt symetricka pozitivne ', ...
        'definitni matice typu m x m'])
end;
```

```
[n1,n2]=size(N);
if norm(N-N')~=0 | n1~=n2 | n1~=n | sum(eig(N)>0)~=n
   error(['N musi byt symetricka pozitivne ', ...
        'definitni matice typu n x n'])
end;
Akriz=inv(N)*A'*M;
vlcisla=eig(Akriz*A);
lambda=max(vlcisla);
betamax=2/(lambda.^2);
if beta <= 0 | beta >= betamax
    beta=1/(lambda.^2);
end;
X=beta*Akriz;
k=0;
t=maxit;
I=eye(m);
while k<maxit
    psi=A*X;
    ksi=-7*I+psi*(9*I+psi*(-5*I+psi));
    omega=psi*ksi;
    Xplus=-(1/16)*X*ksi*(4*I+omega)*(8*I+omega*(4*I+omega));
    X=Xplus;
    k=k+1;
n1=norm(A*X*A-A);
    n2=norm(X*A*X-X);
    n3=norm(M*A*X-(M*A*X)');
    n4=norm(N*X*A-(N*X*A)');
    if n1<=presnost && n2<=presnost ...
       && n3<=presnost && n4<=presnost
 t=k;
       k=maxit;
    end;
end;
iterace=t;
vysledek=X;
```

Pro přesnost a maximální počet iterací platí stejné vlastnosti jako u Kódu 2.6, včetně jejich případného vynechání na vstupu.

Pro daný algoritmus je nutné na vstup zadat i konstantu β, pro jejíž hodnotu potřebujeme znát vlastní čísla matice. Uživatel však tuhle informaci obecně nemá, a tak jestliže na vstup zadá hodnotu, které do intervalu $(0; \frac{2}{\sigma_1^2})$ nepatří, tak se konstantě β automaticky přiřadí hodnota $\frac{1}{\sigma_1^2}$, která do daného intervalu určitě zapadne.

Na výpočet vlastních čísel využíváme dle Poznámky 2.3 v MATLABu příkaz eig.

3. Numerické realizace

Obecně chceme počítat Moore-Penroseovy inverze a Chipmanovy pseudoinverze k maticím o různých rozměrech a hodnostech pomocí algoritmů s různou přesností, a proto je přirozené se ptát, jak velký vliv mají tyto a další vlastnosti na konečný počet iterací v případě iteračních algoritmů a zdali vůbec dospějeme ke správnému výsledku. V této kapitole se touto problematikou budeme zabývat podrobněji. Přitom nebudeme testovat výpočet Chipmanovy pseudoinverze pomocí Kódu 2.3 kvůli již zmíněným nepřesnostem ve výsledcích a Kódu 2.7 kvůli zbytečně přísnému požadavku na původní matici.

Kapitola je rozdělena na dvě podkapitoly – Grevillův algoritmus a Iterační algoritmy.

3.1. Grevillův algoritmus

Grevillův algoritmus je konečný algoritmus a měl by tedy teoreticky vždy dospět k přesnému výsledku. Algoritmus v MATLABu však nepočítá přesně a potřebuje zaokrouhlovat čísla blízká nule, aby se vyhnul astronomickým výsledkům (viz Příklad 2.2). K tomu nám v kódu slouží přesnost na vstupu, jejíž absolutní hodnota je nejvyšší možné číslo, které již považujeme za nulu. Otázkou však zůstává, zdali je odchylka výsledku vypočítaným algoritmem od přesného výsledku významná, nebo jestli ji naopak můžeme zanedbat.

V MATLABu jsem si nechal vygenerovat matice různých rozměrů, jejichž prvky byly náhodně vybrány z odlišných intervalů. Následně jsem k nim pomocí Grevillova algoritmu hledal pseudoinverzní matice a zkoumal jsem, s jakou přesností splňují axiomy z Definice 2.1. To jsem dělal tak, že jsem mezi jednotlivými

normami označené jako

$$n1 = \|AXA - A\|$$
$$n2 = \|XAX - X\|$$
$$n3 = \|(AX)^T - AX\|$$
$$n4 = \|(XA)^T - XA\|$$

hledal maximum, kde A byla původní matice, X matice k ní pseudoinverzní a $\|.\|$ byla euklidovská norma. Dané maximum jsem si označil jako η.

Přesnost	η		
1e-3	0.2035	1e-13	4e-13
1e-6	1e-12	1e-13	4e-13
1e-9	1e-12	1e-13	4e-13

Tabulka 3.1: Maximální odchylky pro matice velikosti 10×10 při vybírání prvků z intervalu [0;1]

Přesnost	η		
1e-3	0.86	6.8748	10.8572
1e-6	7e-11	4e-10	1e-9
1e-9	7e-11	4e-10	1e-9

Tabulka 3.2: Maximální odchylky pro matice velikosti 100×100 při vybírání prvků z intervalu [0;1]

Přesnost	η		
1e-3	161.284	234.1994	39.8991
1e-6	0.022	0.7698	0.0011
1e-9	9e-7	4e-5	1e-7

Tabulka 3.3: Maximální odchylky pro matice velikosti 1000×1000 při vybírání prvků z intervalu [0;1]

Jak lze vypozorovat z uvedených tabulek 3.1–3.3, vyšších odchylek dosáhneme pouze v případech, kdy přesnost není dostatečně vysoká. Je to snadno pozorovatelné zejména u velkých matic.

3.2. Iterační algoritmy

Iterační algoritmy uvedené v této práci se díky limitě obecně k přesnému výsledku pouze přibližují. Algoritmus se zastaví, jakmile přesáhne maximální počet iterací nebo už přesnost bude dostatečně velká. Přesnost je u iteračních algoritmů reprezentována takovým číslem ε, pro jehož hodnotu platí

$$\begin{aligned} \|AXA - A\| &\leq \varepsilon \\ \|XAX - X\| &\leq \varepsilon \\ \|(MAX)^T - MAX\| &\leq \varepsilon \\ \|(NXA)^T - NXA\| &\leq \varepsilon \end{aligned}$$

kde $X \in \mathcal{M}_{n,m}$ je příslušná Chipmanova pseudoinverze matice A, $M \in \mathcal{M}_{m,m}$ a $N \in \mathcal{M}_{n,n}$ jsou symetrické pozitivně definitní a $\|.\|$ je euklidovská norma. Ve speciálních případech můžeme za M nebo N dosadit jednotkovou matici příslušných rozměrů I.

V MATLABu jsem si nechal vygenerovat náhodné matice různých rozměrů a zkoumal jsem závislost rozdílných vstupů iteračních algoritmů na konečný počet iterací, které jsem poté zaznamenal do tabulek. Každá tabulka se věnuje právě třem různým maticím o shodných rozměrech, jejíž prvky byly náhodně vybrány ze shodných intervalů.

V případě Kódů 2.4–2.6 bylo zapotřebí na vstup zadat kromě původních matic a přesnosti i celé číslo $q \geq 2$. Vygeneroval jsem si proto matice se třemi různými rozměry, pro každý rozměr 3 matice, a nechal k nim pomocí algoritmů z Vět 2.10–2.12 hledat Chipmanovu pseudoinverzi, kde jsem na vstup zadával různá čísla q a rozdílné přesnosti. Výsledné počty iterací jsem pak zapsal do tabulek, kde ve sloupcích jsou zapsané konečné počty iterací pro algoritmy s různými čísly q na vstupu a v řádcích počty iterací pro algoritmus s různými přesnostmi na vstupu, přičemž každá tabulka reprezentuje právě jeden rozměr matice. Jestliže je číslo v tabulce červené, znamená to, že po daném počtu iterací se už přesnost přestala zlepšovat, tj. hodnota čísla ε přestala klesat.

U Kódu 2.4 bylo ještě zapotřebí na vstup zadat kromě matic a přesnosti i číslo β z intervalu $(0; \frac{2}{\sigma_1^2})$, kde σ_1^2 bylo největší vlastní číslo matice $N^{-1}A^TMA$. Za β jsem volil hodnotu $\frac{1}{\sigma_1^2}$.

Za matice M a N jsem u všech iteračních algoritmů volil čtvercové matice příslušných rozměrů $2I$, jejichž schéma vypadalo jako

$$2I = \begin{pmatrix} 2 & 0 & 0 & \dots & 0 \\ 0 & 2 & 0 & \dots & 0 \\ 0 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 2 \end{pmatrix}.$$

Volba jiných matic M a N (a u Kódu 2.4 i volba jiného čísla β) má samozřejmě vliv na konečný počet iterací. Tyto hodnoty jsem však již netestoval z důvodu časové náročnosti a rozsahu práce.

3.2.1. Tabulky pro algoritmus z Věty 2.10

Pro daný algoritmus budeme potřebovat, aby původní matice měly plnou sloupcovou hodnost. Vybereme například matice rozměrů 15×10, 150×100 a 1500×1000.

Prvky pro následující 3 matice jsou náhodně vybrány z intervalu $[0; 1]$, přičemž matice mají po vybrání prvků plnou sloupcovou hodnost.

Přesnost \ q	2			10			100		
1e-3	21	18	20	7	6	6	4	3	3
1e-6	21	19	21	7	6	7	4	3	4
1e-9	22	19	21	7	6	7	4	3	4

Tabulka 3.4: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	30	28	27	9	9	9	5	5	5
1e-6	30	29	28	10	9	9	5	5	5
1e-9	31	29	29	10	9	9	5	5	5

Tabulka 3.5: Počet iterací pro matice velikosti 150×100 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	35	35	35	11	11	11	6	6	6
1e-6	36	36	36	11	11	11	6	6	6
1e-9	37	37	37	11	11	11	6	6	6

Tabulka 3.6: Počet iterací pro matice velikosti 1500×1000 při vybírání prvků z intervalu [0;1]

Jak je vidět z tabulek 3.4–3.6, s velikostí matice roste i počet potřebných iterací k určení její Chipmanovy pseudoinverze. Přitom je zřejmé, že s rostoucí přesností opět roste počet iterací. Dále je možné vypozorovat, že s rostoucím číslem q klesá počet iterací. Rostoucí q má však negativní vliv na čas výpočtu. Je to způsobeno výpočtem mocnin matic, které je závislé na velikosti již zmíněného q. Tento trend budeme pozorovat i u tabulek věnovaným algoritmům z Vět 2.10–2.12.

Nyní se podíváme, jaký vliv má na konečný počet iterací interval, z něhož vybíráme prvky pro matice. Vezmeme si tedy například matice s plnou sloupcovou hodností, jejichž prvky jsou vybrané z intervalu $[-100; 100]$.

Přesnost \ q	2			10			100		
1e-3	14	15	14	4	5	5	2	3	3
1e-6	14	15	15	5	5	5	3	3	3
1e-9	15	16	15	5	5	5	3	3	3

Tabulka 3.7: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	20	20	19	6	6	6	3	3	3
1e-6	20	21	20	6	7	6	3	4	3
1e-9	21	21	21	7	7	7	4	4	4

Tabulka 3.8: Počet iterací pro matice velikosti 150×100 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	24	24	24	7	7	7	4	4	4
1e-6	24	24	24	7	7	7	4	4	4
1e-9	24	24	24	7	7	7	4	4	4

Tabulka 3.9: Počet iterací pro matice velikosti 1500×1000 při vybírání prvků z intervalu [-100;100]

K nalezení Chipmanovy pseudoinverze matic, jejichž prvky byly náhodně vybrány z intervalu [-100;100], potřebujeme menší počet iterací, než pro matice s prvky z intervalu [0;1]. Interval [0;1] je úzký, a tak rozdíl mezi čísly vybranými z tohoto intervalu je velmi malý, což způsobuje větší počet iterací.

Nyní se podíváme na případ, kdy prvky vybereme z velice širokého intervalu, přičemž matice mají po vybrání prvků plnou sloupcovou hodnost. Vezmeme si například interval [-1000000;1000000].

Přesnost \ q	2			10			100		
1e-3	13	17	15	4	6	5	2	3	3
1e-6	14	18	15	4	6	5	2	3	3
1e-9	16	21	17	6	7	6	4	5	5

Tabulka 3.10: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	21	20	21	7	6	7	4	3	4
1e-6	21	21	21	7	7	7	4	4	4
1e-9	23	24	23	9	9	9	5	5	5

Tabulka 3.11: Počet iterací pro matice velikosti 150 × 100 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	24	24	24	8	8	8	4	4	4
1e-6	25	25	25	8	8	8	4	4	4
1e-9	28	28	28	10	10	10	5	5	5

Tabulka 3.12: Počet iterací pro matice velikosti 1500 × 1000 při vybírání prvků z intervalu [-1000000;1000000]

Počet iterací algoritmu pro matice, jejichž prvky byly náhodně vybrány z intervalu [-1000000;1000000] se u vyšších přesností výrazně neliší od počtu iterací pro matice s prvky z intervalu [-100,100]. V případě, kdy zadáme přesnost 1e-9 či vyšší, však algoritmus nezkonverguje. Jelikož je interval velmi široký, rozdíl mezi čísly z tohoto intervalu je velmi vysoký, což způsobuje problémy s konvergencí.

3.2.2. Tabulky pro algoritmus z Věty 2.11

Pro daný algoritmus budeme potřebovat, aby původní matice měly plnou řádkovou hodnost. Vybereme například matice rozměrů 10 × 15, 100 × 150 a 1000 × 1500.

Prvky pro následující 3 matice jsou náhodně vybrány z intervalu $[0; 1]$, přičemž matice mají po vybrání prvků plnou řádkovou hodnost.

Přesnost \ q	2			10			100		
1e-3	19	18	18	6	6	6	3	3	3
1e-6	20	19	19	6	6	6	3	3	3
1e-9	21	20	19	7	6	6	4	3	3

Tabulka 3.13: Počet iterací pro matice velikosti 10×15 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	28	28	29	9	9	9	5	5	5
1e-6	29	29	30	9	9	9	5	5	5
1e-9	29	30	30	9	9	9	5	5	5

Tabulka 3.14: Počet iterací pro matice velikosti 100×150 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	35	35	35	11	11	11	6	6	6
1e-6	36	36	36	11	11	11	6	6	6
1e-9	36	36	37	11	11	11	6	6	6

Tabulka 3.15: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [0;1]

Pro tabulky 3.13–3.15 platí stejné poznatky jako u tabulek 3.4–3.6, přičemž nenastává žádná výrazná změna v konečném počtu iterací. Podíváme se tedy na širší intervaly.

Prvky pro následující 3 matice plné řádkové hodnosti jsou náhodně vybrány z intervalu $[-100; 100]$.

Přesnost \ q	2			10			100		
1e-3	13	15	18	4	5	6	2	3	3
1e-6	13	15	18	4	5	6	2	3	3
1e-9	14	16	19	5	5	6	3	3	3

Tabulka 3.16: Počet iterací pro matice velikosti 10×15 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	20	20	20	6	6	6	3	3	3
1e-6	21	21	21	7	7	7	4	4	4
1e-9	21	21	21	7	7	7	4	4	4

Tabulka 3.17: Počet iterací pro matice velikosti 100×150 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	24	24	24	7	7	7	4	4	4
1e-6	24	24	24	8	8	8	4	4	4
1e-9	25	25	25	8	8	8	4	4	4

Tabulka 3.18: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [-100;100]

Prvky pro následující 3 matice plné řádkové hodnosti jsou náhodně vybrány z intervalu $[-1000000; 1000000]$.

Přesnost \ q	2			10			100		
1e-3	16	16	15	5	5	5	3	3	3
1e-6	16	17	15	5	5	5	3	3	3
1e-9	18	19	17	8	7	11	4	5	8

Tabulka 3.19: Počet iterací pro matice velikosti 10×15 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	20	21	21	6	7	7	3	4	4
1e-6	20	21	22	7	7	7	4	4	4
1e-9	23	23	23	9	10	9	5	6	6

Tabulka 3.20: Počet iterací pro matice velikosti 100×150 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	24	24	24	8	8	8	4	4	4
1e-6	25	25	25	8	8	8	4	4	4
1e-9	27	28	26	9	9	12	5	5	5

Tabulka 3.21: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [-1000000;1000000]

Pro Tabulky 3.16–3.21 platí stejné poznatky jako u Tabulek 3.7–3.12, přičemž opět nedochází k žádným výrazným změnám.

3.2.3. Tabulky pro algoritmus z Věty 2.12

Daný algoritmus funguje pro obecné matice. Pro srovnání s algoritmem z Věty 2.10 si nejprve vybereme matice rozměrů 15×10, 150×100 a 1500×1000.

Prvky pro následující 3 matice jsou náhodně vybrány z intervalu $[0; 1]$.

Přesnost \ q	2			10			100		
1e-3	19	21	18	6	7	6	3	4	3
1e-6	20	22	19	6	7	6	3	4	3
1e-9	20	23	19	6	7	6	3	4	3

Tabulka 3.22: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	29	29	28	9	9	9	5	5	5
1e-6	30	30	29	9	9	9	5	5	5
1e-9	30	30	30	9	9	9	5	5	5

Tabulka 3.23: Počet iterací pro matice velikosti 150×100 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	35	35	35	11	11	11	6	6	6
1e-6	36	36	36	11	11	11	6	6	6
1e-9	37	37	37	11	11	11	6	6	6

Tabulka 3.24: Počet iterací pro matice velikosti 1500×1000 při vybírání prvků z intervalu [0;1]

Srovnáním s tabulkami 3.4–3.6 dostaneme podobný počet iterací. Nutno však podotknout, že čas výpočtu se výrazně zvýšil z důvodu výpočtu skeletního rozkladu v algoritmu.

Nyní se podíváme na matice, jejichž prvky jsou vybrané z intervalu $[-100; 100]$.

Přesnost \ q	2			10			100		
1e-3	13	15	14	4	5	5	2	3	3
1e-6	14	15	15	5	5	5	3	3	3
1e-9	15	16	15	5	5	5	3	3	3

Tabulka 3.25: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	20	20	20	6	6	6	3	3	3
1e-6	21	20	20	7	6	6	4	3	3
1e-9	21	21	21	7	7	7	4	4	4

Tabulka 3.26: Počet iterací pro matice velikosti 150×100 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	24	24	24	7	7	7	4	4	4
1e-6	24	24	24	7	7	7	4	4	4
1e-9	25	25	25	8	8	8	4	4	4

Tabulka 3.27: Počet iterací pro matice velikosti 1500×1000 při vybírání prvků z intervalu [-100;100]

Nyní se podíváme na případ, kdy prvky vybereme z velice širokého intervalu [-1000000;1000000].

Přesnost \ q	2			10			100		
1e-3	13	14	15	4	4	5	2	2	3
1e-6	14	14	15	4	5	5	2	3	3
1e-9	15	14	15	6	6	6	3	4	3

Tabulka 3.28: Počet iterací pro matice velikosti 15×10 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	20	21	21	6	7	7	3	4	4
1e-6	21	21	21	7	7	7	4	4	4
1e-9	23	23	23	8	7	8	5	5	5

Tabulka 3.29: Počet iterací pro matice velikosti 150×100 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	24	24	24	8	8	8	4	4	4
1e-6	25	25	25	8	8	8	4	4	4
1e-9	28	28	28	11	11	11	6	6	6

Tabulka 3.30: Počet iterací pro matice velikosti 1500 × 1000 při vybírání prvků z intervalu [-1000000;1000000]

Ve srovnáním s Tabulkami 3.7–3.12 není rozdíl mezi počty iterací algoritmů z Tabulek 3.25–3.30 velice významný. Vzhledem k tomu, že v Kódu 2.6 ještě počítáme se skeletním rozkladem, je Kód 2.4 méně časově náročný a tudíž je výhodnější ho využít pro matice plné sloupcové hodnosti.

Nyní se ještě zběžně podíváme na matice rozměrů 10 × 15, 100 × 150 a 1000 × 1500 pro srovnání s algoritmem z Věty 2.11.

Pro následující 3 tabulky jsou prvky matic vybrány z intervalu [0;1].

Přesnost \ q	2			10			100		
1e-3	31	24	28	10	8	9	5	4	5
1e-6	32	25	28	10	8	9	5	4	5
1e-9	32	26	29	10	8	9	5	4	5

Tabulka 3.31: Počet iterací pro matice velikosti 10 × 15 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	40	44	46	12	14	14	6	7	7
1e-6	41	45	47	13	14	15	7	7	8
1e-9	42	45	48	14	15	16	7	7	8

Tabulka 3.32: Počet iterací pro matice velikosti 100 × 150 při vybírání prvků z intervalu [0;1]

Přesnost \ q	2			10			100		
1e-3	64	66	57	19	20	17	10	10	9
1e-6	69	67	59	22	20	18	10	10	9
1e-9	69	69	57	22	20	20	10	10	9

Tabulka 3.33: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [0;1]

Nyní se podíváme na matice, jejichž prvky jsou vybrané z intervalu $[-100; 100]$.

Přesnost \ q	2			10			100		
1e-3	18	21	19	6	7	6	3	4	3
1e-6	19	22	19	6	7	6	3	4	3
1e-9	19	22	20	6	7	6	3	4	3

Tabulka 3.34: Počet iterací pro matice velikosti 10×15 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	30	34	31	9	10	10	5	5	5
1e-6	30	34	32	9	11	10	5	6	5
1e-9	31	36	32	10	12	11	5	6	5

Tabulka 3.35: Počet iterací pro matice velikosti 100×150 při vybírání prvků z intervalu [-100;100]

Přesnost \ q	2			10			100		
1e-3	48	43	44	15	14	14	8	7	7
1e-6	49	44	45	17	14	14	8	8	8
1e-9	49	49	48	17	16	16	8	7	7

Tabulka 3.36: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [-100;100]

Nyní se podíváme na případ, kdy prvky vybereme z velice širokého intervalu [-1000000;1000000].

Přesnost \ q	2			10			100		
1e-3	14	17	22	5	5	7	3	3	4
1e-6	15	17	22	5	6	7	3	3	4
1e-9	16	18	23	5	6	7	5	3	5

Tabulka 3.37: Počet iterací pro matice velikosti 10×15 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	36	67	48	11	32	15	6	12	8
1e-6	36	67	48	11	34	15	7	12	8
1e-9	36	67	48	11	34	15	7	12	8

Tabulka 3.38: Počet iterací pro matice velikosti 100×150 při vybírání prvků z intervalu [-1000000;1000000]

Přesnost \ q	2			10			100		
1e-3	48	45	43	18	17	16	7	7	8
1e-6	48	45	43	18	17	16	7	7	8
1e-9	48	45	43	18	17	16	7	7	8

Tabulka 3.39: Počet iterací pro matice velikosti 1000×1500 při vybírání prvků z intervalu [-1000000;1000000]

Oproti Tabulkám 3.13–3.21 mají příslušné Tabulky 3.31–3.39 průměrně vyšší počet iterací a u několika případů dokonce algoritmus popsaný v Kódu 2.6 nezkonverguje vůbec. Jestliže tedy chceme zjistit Chipmanovu pseudoinverzi k matici s plnou sloupcovou hodností, pak je výhodnější využít Kódu 2.5.

3.2.4. Tabulky pro algoritmus z Věty 2.14

Algoritmus funguje pro obecné matice. Pro daný algoritmus již nepotřebujeme zadávat q na vstup, a tak nyní sloupce jednotlivých tabulek budou reprezentovat velikost původní matice.

Nejprve vybereme matice rozměrů 15×10, 150×100 a 1500×1000.

Velikost / Přesnost	15×10			150×100			1500×1000		
1e-3	5	5	5	8	8	8	10	10	10
1e-6	5	5	5	8	8	8	11	11	11
1e-9	5	6	5	8	8	8	11	11	11

Tabulka 3.40: Počet iterací pro matice při vybírání prvků z intervalu [0;1]

Velikost / Přesnost	15×10			150×100			1500×1000		
1e-3	7	7	7	8	9	8	9	10	9
1e-6	7	7	8	9	9	9	10	10	10
1e-9	8	8	8	9	9	9	10	10	10

Tabulka 3.41: Počet iterací pro matice při vybírání prvků z intervalu [-100;100]

Velikost / Přesnost	15×10			150×100			1500×1000		
1e-3	14	14	14	16	16	16	17	17	17
1e-6	15	14	14	16	16	16	17	17	17
1e-9	18	17	17	19	19	19	20	20	20

Tabulka 3.42: Počet iterací pro matice při vybírání prvků z intervalu [-1000000;1000000]

Jak lze vyčíst z Tabulek 3.4–3.12, 3.22–3.30 a 3.40–3.42, využitím Kódu 2.8 pro matice, jejichž prvky byly vybírány z menších intervalů, dostaneme podobný počet iterací jako využitím Kódů 2.4 a 2.6 s $q = 10$ na vstupu. Časy výpočtu bohužel nemám zaznamenané, ale počítání Kódem 2.8 je mírně rychlejší a tudíž

je v tomto případě výhodnější. Jestliže je interval, z kterého prvky vybíráme, velice široký, pak se pro Kód 2.8 výsledný počet iterací zvyšuje zatímco pro Kódy 2.4 a 2.6 se počty iterací významně nemění. Kvůli nutnosti počítání skeletního rozkladu v Kódu 2.6 se nám však výrazně zvyšuje čas výpočtu. V daném případě je tedy výhodnější využít Kódu 2.4 s $q = 10$ na vstupu pro matice plné sloupcové hodnosti a Kódu 2.8 pro matice obecné. Kód 2.4 s $q = 100$ na vstupu ani neuvažujeme z důvodu velmi vysoké časové náročnosti.

Nyní se podíváme na matice rozměrů 10×15, 100×150 a 1000×1500.

Velikost / Přesnost	10×15			100×150			1000×1500		
1e-3	5	5	5	8	8	8	10	10	10
1e-6	5	5	5	8	8	8	11	11	11
1e-9	5	6	5	8	8	8	11	11	11

Tabulka 3.43: Počet iterací pro matice při vybírání prvků z intervalu [0;1]

Velikost / Přesnost	10×15			100×150			1000×1500		
1e-3	7	7	7	9	8	8	9	10	9
1e-6	7	7	8	9	9	9	10	10	10
1e-9	7	7	8	9	9	9	10	10	10

Tabulka 3.44: Počet iterací pro matice při vybírání prvků z intervalu [-100;100]

Velikost / Přesnost	10×15			100×150			1000×1500		
1e-3	15	15	14	16	16	16	17	17	17
1e-6	15	15	14	16	16	16	17	17	17
1e-9	16	20	17	21	21	20	23	22	22

Tabulka 3.45: Počet iterací pro matice při vybírání prvků z intervalu [-1000000;1000000]

Srovnáním Tabulek 3.13–3.21, 3.31–3.39 a 3.43–3.45 dojdeme k podobných poznatkům, jako u Tabulek 3.40–3.42.

3.2.5. Souhrn

Jestliže je původní matice plné řádkové či sloupcové hodnosti, jejíž prvky nejsou vybírány z velice širokého intervalu, pak je nejvýhodnější využít Kódu 2.8.

Jestliže je původní matice plné řádkové či sloupcové hodnosti, jejíž prvky jsou vybírány z velice širokého intervalu, pak je nejvýhodnější využít Kódů 2.4 či 2.5 s $q = 10$ na vstupu.

Jestliže původní matice není plné řádkové ani plné sloupcové hodnosti, pak je nejvýhodnější využít Kódu 2.8.

Závěr

Cílem této bakalářské práce bylo představit si klasické a zobecněné Moore-Penroseovy inverze. Pro obě inverze jsme si uvedli některé klíčové vlastnosti, přičemž nejdůležitější byly 4 axiomy, které musely inverze splňovat. Dále jsme se zaměřili na metody pro jejich výpočet, čemuž se věnovala většina práce, které byly realizovány pomocí algoritmů. Jednotlivé algoritmy jsme si popsali a uvedli si jejich kódy vytvořené v softwaru MATLAB. Nakonec jsme si otestovali, jak jsou jednotlivé algoritmy účinné a zdali vůbec dojdou ke správným výsledkům.

Psaní této práce byla velká výzva. Dorozumět se s programem LaTeX (program na sázení textu) bylo obzvláště obtížné, nicméně po hodinách neúnavného hledání na internetu jsem tomu začínal přicházet na kloub a průběžně se psaní v softwaru LaTeX stalo běžnou činností. Zároveň jsem si připomenul znalosti z hodin Software pro aplikovanou matematiku při vytváření jednotlivých kódů v MATLABu a seznámil se s pojmy a příklady, ke kterým bych se v hodinách běžně nedostal.

Celkem bych psaní této práce hodnotil jako dobrou zkušenost a doufám, že si budoucí čtenáři z práce odnesou alespoň tolik užitečných informací, jako jsem já získal studiem daného tématu.

Literatura

[1] Chipman, J. S.: *Specification problems in regression analysis.* T. L. Boullion, P. I. Odell, Proceedings of the Symposium on Theory and Applications of Generalized Inverses of Matrices, Texas, 1968, 114–176.

[2] Davis, P. J.: *Circulant Matrices.* J. Wiley, New York, 1979.

[3] Djordovič, D. S., Stanimirovič, P. S.: *Universal iterative methods for computing generalized inverses.* Acta Mathematica Hungarica **79** (1998), 253–268.

[4] Machalová, J.: *Chipman Pseudoinverse of Matrix, its Computation and Application in Spline Theory.* Acta Univ. Palacki. Olomuc., Fac. rer. nat., Mathematica **39** (2000), 143–157.

[5] Rao, C. R., Mitra, K. S.: *Generalized Inverse of Matrices and Its Application.* J. Wiley, New York, 1971.

[6] Soheili, A. R., Soleymani, F., Petković, M. D.: *On the computation of weighted Moore-Penrose inverse using a high-order matrix method.* Computers and Mathematics with Applications **66** (2013), 2344–2351.