Fakulta aplikovaných věd
Katedra matematiky
Obor: Matematické inženýrství

Diplomová práce

Chování metody sdružených gradientů v konečné aritmetice

Veronika Kalusová
Vedoucí práce: RNDr. Petr Tichý, Ph.D.
Plzeň 2012

Čestné prohlášení:

Prohlašuji, že jsem svou diplomovou práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne ....

Veronika Kalusová

Poděkování:

Chtěla bych poděkovat vedoucímu práce panu RNDr. Petru Tichému, Ph.D. za odbornou pomoc, všechny rady a hlavně za čas, který mi věnoval.

Název práce:
Chování metody sdružených gradientů v konečné aritmetice

Abstrakt:

Výsledky spočtené metodou sdružených gradientů (CG) v konečné aritmetice počítače se často výrazně liší od ideálních (přesných) hodnot. Vlivem konečné aritmetiky dochází ke ztrátě ortogonality, zpoždění konvergence, případně k zastavení metody na hladině maximální dosažitelné přesnosti řešení. Analýza chování metody sdružených gradientů v konečné aritmetice byla předmětem mnoha prací. Např. Greenbaum a Strakoš ve svých článcích vysvětlují, že na výpočty CG aplikované na systém $\mathbf{Ax} = \mathbf{b}$ v konečné aritmetice lze hledět jako na výpočty přesné CG aplikované na jinou soustavu $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, která závisí na iteračním kroku. Také ukazují, že výpočty v konečné aritmetice jsou podobné přesným výpočtům aplikovaným na soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$, která na iteračním kroku nezávisí. Jinak řečeno, některé teoretické vlastnosti CG zůstavají zachovány i v konečné aritmetice.

Hlavním cílem diplomové práce bude shrnutí poznatků o chování CG v konečné aritmetice počítače. Součástí práce budou numerické experimenty ukazující výše zmíněné jevy (ztráta ortogonality, zpoždění konvergence, maximální dosažitelná přesnost, podobnost FP výpočtů aplikovaných na $\mathbf{Ax} = \mathbf{b}$ a přesných výpočtů aplikovaných na $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ nebo na $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$) a další. Budou zkoumány vlastnosti simulace FP výpočtů (úlohy $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$) a problémy spojené s matematickým modelem FP výpočtů (s úlohou $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$).

Klíčová slova:
Metoda sdružených gradientů, Gaussova kvadratura, Lanczosův algoritmus, konečná aritmetika počítače, ztráta ortogonality, zpoždění konvergence, maximální dosažitelná přesnost

Title:
The conjugate gradient method and its behavior in finite precision arithmetic

Abstract:

Results of finite precision conjugate gradient computations can dramatically differ from their ideally (exact) counterparts. When computing in finite precision aritjmetic, orthogonality is usually lost, convergence is delayed and there is a limitation on the accuracy of the computed solution (ultimate attainable accuracy). Behavior of finite precision conjugate gradient computations was object in several papers. In their papers Greenbaum and Strakoš explain, that finite precision conjugate gradient computations applied to $\mathbf{Ax} = \mathbf{b}$ correspond to exact computations applied to another system $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, which depends on number of iterations. They also explain, that finite precision computations are similar to exact computations applied to system $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$, which doesn't depend on number of iterations. In other words, some theoretical properties CG hold in finite precison computations.

The main aim of this diploma thesis will be summary of results on behavior of conjugate gradient method in finite precision arithemtic. We will present numerical experiments showing above-mentioned phenomena, thus loss of orthogonality, delay of convergence, ultimate attainable accuracy, resemblance between FP computations $\mathbf{Ax} = \mathbf{b}$ and exact computations $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ ($\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$) and others. Properties of the system that simulates FP computations ($\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$) and properties of the system associated with the mathematical model of FP computations ($\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$) will be investigated too.

Key words:
Conjugate Gradient method, Gauss-quadrature, Lanczos algorithm, finite precision arithmetic, loss of orthogonality, delay of convergence, ultimate attainable accuracy

Obsah

Seznam použitých symbolů a zkratek

symbol	význam
$\{a_1, a_2, \ldots, a_N\}$	množina prvků $a_1, a_2, \ldots, a_N$
$\mathbf{A} = [A_{ij}]_{i,j=1}^{N}$	matice daná výčtem prvků A_{ij}
$\mathbf{A} = \mathrm{diag}[a_1, a_2, \ldots, a_N]$	diagonální matice, jejíž nenulové prvky jsou dány čísly a_i
$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N]$	matice, jejíž sloupce jsou vektory $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$
$\mathbf{e}_{(i)}$	i-tý sloupec jednotkové matice
$f(x)$	funkce f závislá na proměnné x
$F(x)$	funkcionál F závislý na proměnné x
$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$	k-tý Krylovův prostor generovaný maticí $\mathbf{A}$ a vektorem $\mathbf{r}_0$
$\mathbf{o}$	nulový vektor
$O(k)$	číslo blízké hodnotě k
p_k	polynom stupně k
rand	náhodné reálné číslo
$\mathrm{rank}(\mathbf{A})$	hodnost matice $\mathbf{A}$
$\mathbb{R}^{N\times N}$	$N \times N$ rozměrný prostor reálných čísel
$\mathrm{span}\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N\}$	lineární obal vektorů $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$
$\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$	sloupcový vektor daný výčtem prvků $x_1, x_2, \ldots, x_N$
$\mathbf{x}^T$	transponovaný vektor k vektoru $\mathbf{x}$
$\mathbf{x}_k$	vektor generovaný v k-tém kroce algoritmu
ε	strojová přesnost
$\kappa(\mathbf{A})$	číslo podmíněnosti matice $\mathbf{A}$

zkratka	význam
CG	Metoda sdružených gradientů (Conjugate Gradient method)
FP výpočty	výpočty provedené v konečné aritmetice (finite precision)
GQ	Gaussova kvadratura (Gauss quadrature)
HS verze	Hestenesova-Stiefelova verze
HY verze	Hagemanova-Youngova verze
R verze	Rutishauserova verze

Úvod

Řešením soustavy lineárních algebraických rovnic

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

kde $\mathbf{A} \in \mathbb{R}^{N\times N}$ je čtvercová matice, $\mathbf{x} \in \mathbb{R}^N$ označuje hledaný vektor a $\mathbf{b} \in \mathbb{R}^N$ je vektor pravé strany, se zabývají metody, které lze obecně rozdělit na metody přímé a iterační.

Pro přímé metody je charakteristické, že naleznou řešení soustavy po konečném počtu kroků. Jejich principem je eliminace neznámých a jsou vhodné, pokud matice $\mathbf{A}$ je malá a plná. Jejich nevýhodou je, že jsou paměťově náročné a jejich algoritmus musí být spuštěn po stejný počet kroků i v případě, že chceme pouze přibližné řešení.

Iterační metody jsou efektivní především pro velké řídké matice. Základní myšlenkou je využití násobení matice vektorem ke konstrukci posloupnosti aproximací řešení. Jednotlivé iterační metody se od sebe liší tím, jakým způsobem se určují nové aproximace řešení.

Je-li matice $\mathbf{A}$ symetrická a pozitivně definitní je vhodné k řešení soustavy použít třídu metod nazvaných gradientní metody. Těmito metodami se zabývá část 1.2, kde je ukázána jejich souvislost s minimalizací kvadratického funkcionálu. Mezi tyto metody patří i metoda sdružených gradientů, která je zde označena zkratkou CG (CG = Conjugate Gradient).

Kapitola 1 je věnovaná metodě sdružených gradientů a různým způsobům její algoritmické realizace. Je rozdělena na několik částí. První z nich představuje metodu stejně jako byla představena v roce 1952 jejími autory, tedy algoritmem A1. Další části se věnují matematické podstatě CG. Nejprve její souvislostí s minimalizací funkcionálu, jak bylo uvedeno výše, poté jejím vztahem s Krylovovými metodami a nakonec souvislostí s Gaussovou kvadraturou.

Vztah mezi CG a Gaussovou kvadraturou je velmi důležitý, protože na základě této souvislosti vznikla myšlenka matematického modelu výpočtů CG v konečné aritmetice, viz. [4, 6]. Teorií založenou na této myšlence se zabývá podstatná část práce. Dále je zde představeno několik způsobů realizace CG a důkazy jejich ekvivalence. Na závěr kapitoly jsou zde uvedeny a dokázány nejdůležitější algebraické vlastnosti algoritmů.

Kapitola 2 popisuje chování CG v konečné aritmetice. Je zde prezentován vliv zaokrouhlovacích chyb na některé vlastnosti CG, například na zachování ortogonality reziduových vektorů nebo na konvergenci rezidua k nulovému vektoru. Je zde ukázán i vliv na Gaussovu

kvadraturu a také vliv na CG z pohledu Gaussovy kvadratury. Druhá část této kapitoly porovnává různé algoritmické realizace metody. Je zde vybráno několik vlastností a ukázáno jejich zachování v konečné aritmetice pro tři různé verze algoritmu CG.

Kapitola 3 se zabývá simulací výpočtů provedených v konečné aritmetice (FP výpočty) pomocí přesných výpočtů aplikovaných na pomocný systém. Je založena na článku [4], kde je dokázáno, že výpočty v konečné aritmetice odpovídají přesným výpočtům aplikovaným na větší systém rovnic $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, kde $\hat{\mathbf{A}}$ má všechna vlastní čísla umístěna ve shlucích okolo vlastních čísel matice $\mathbf{A}$ a je sestrojena způsobem uvedeným v článku [4]. Jsou zde experimenty potvrzující výsledky tohoto článku a také návrh jiného postupu sestrojení matice $\hat{\mathbf{A}}$, který je zjednodušením postupu původního. Přestože tato matice $\hat{\mathbf{A}}$ nesplňuje předpokládané vlastnosti, přesné výpočty aplikované na ní simulují FP výpočty déle.

Matice $\hat{\mathbf{A}}$ je sestrojena na základě dat získaných pomocí výpočtů provedených v konečné aritmetice. Jde tedy o aposteriorní konstrukci systému. Kapitola 4 se zabývá apriorní konstrukcí systému, pomocí něhož lze modelovat FP výpočty. Narozdíl od simulace se zde tedy výpočty modelují pouze pomocí znalosti spektrálních vlastností matice $\mathbf{A}$ a znalosti projekcí pravé strany do ortonormální báze vlastních vektorů matice $\mathbf{A}$. Kapitola je založena na myšlence uvedené v článku [6]. Jsou zde prezentovány experimenty inspirované tímto článkem a navíc je zde uveden jiný model, který může, ale i nemusí lépe aproximovat FP výpočty, viz. sekce 4.2. V těchto experimentech používáme speciální matice, které jsou označené jako Strakošovy matice a jejichž důležité vlastnosti jsou popsány v kapitole 2.

Kromě výše zmíněného jsou v práci uvedeny i jiné grafy, které jsou inspirované různými články. V těchto případech je vždy v popisu obrázku napsáno, jakým článkem je experiment inspirován. Všechny výpočty byly uskutečněny pomocí programu Matlab 7.11.0 a jejich zdrojové kódy jsou na přiloženém CD spolu s návodem, jak se orientovat v jeho adresářové struktuře.

Kapitola 1

Metoda sdružených gradientů (CG)

1.1 Klasický algoritmus

Jak bylo uvedeno v úvodu, metoda sdružených gradientů je iterační metoda řešící soustavy lineárních algebraických rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.1}$$

kde vektor $\mathbf{x}$ je neznámá, $\mathbf{A}$ je symetrická pozitivně definitní matice typu $N \times N$ a $\mathbf{b}$ je vektor pravé strany. Takové soustavy vznikají například při numerickém řešení eliptických parciálních diferenciálních rovnic metodou konečných prvků či konečných diferencí.

Metoda sdružených gradientů byla představena Hestenesem a Stiefelem v roce 1952 v článku [9]. Byla zde vysvětlena bez odvození pomocí algoritmu A1, a proto i tato práce bude začínat tímto algoritmem. Princip odvození metody pomocí minimalizace funkcionálu bude popsán až v následující podkapitole a je založen na pracech [3, 24].

Algoritmus A1 Hestenes-Stiefel

vstup $\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{p}_0 := \mathbf{r}_0$
for $k = 1, 2 \ldots$
  $\gamma_{k-1} := \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{p}_{k-1}^T \mathbf{A}\mathbf{p}_{k-1}}$
  $\mathbf{x}_k := \mathbf{x}_{k-1} + \gamma_{k-1}\mathbf{p}_{k-1}$
  $\mathbf{r}_k := \mathbf{r}_{k-1} - \gamma_{k-1}\mathbf{A}\mathbf{p}_{k-1}$
  $\delta_k := \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}$
  $\mathbf{p}_k := \mathbf{r}_k + \delta_k \mathbf{p}_{k-1}$
end

Algoritmus A1 je nejpoužívanější realizací metody sdružených gradientů, ale existují i jiné algoritmické realizace, kterým bude věnována část 1.5.1.

Z matematického hlediska existuje několik přístupů, kterými lze popsat tuto metodu. V

následujících sekcích budou uvedeny některé z nich, konkrétně:

- CG jako minimalizace funkcionálu,
- CG jako projektivní metoda,
- CG jako Gaussova kvadratura.

1.2 Souvislost s minimalizací funkcionálu

Na gradientní metody lze nahlížet jako na optimalizační metody, které hledají minimum funkcionálu

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}. \tag{1.2}$$

Ekvivalence hledání řešení rovnice (1.1) a hledání minima tohoto funkcionálu je zřejmá. Gradient tohoto funkcionálu je totiž tvaru $\nabla J(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$, a protože tento funkcionál je kvadratický, musí pro jeho globální minimum $\mathbf{x}_m$ platit $\nabla J(\mathbf{x}_m) = 0$.

Díky tomuto přístupu ke gradientním metodám víme, jak nejlépe měřit vzdálenost mezi přesným a přibližným řešením soustavy. Z vyjádření funkcionálu pro aproximaci řešení $\mathbf{x}_k$

$$\begin{aligned} J(\mathbf{x}_k) &= \frac{1}{2}\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k - \mathbf{x}_k^T\mathbf{A}\mathbf{x} = \\ &= \frac{1}{2}\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k - \mathbf{x}_k^T\mathbf{A}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} = \\ &= \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T\mathbf{A}(\mathbf{x} - \mathbf{x}_k) - \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \end{aligned}$$

totiž plyne, že ke globálnímu minimu funkcionálu se můžeme přiblížit pouze pokud minimalizujeme člen $(\mathbf{x} - \mathbf{x}_k)^T\mathbf{A}(\mathbf{x} - \mathbf{x}_k)$, jehož velikost je dána energetickou normou chyby:

$$\|\mathbf{e}_k\|_{\mathbf{A}} = \|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}} = \sqrt{(\mathbf{x} - \mathbf{x}_k)^T\mathbf{A}(\mathbf{x} - \mathbf{x}_k)}. \tag{1.3}$$

Minimalizací této chyby se tedy minimalizuje i funkcionál (1.2).

Jak bylo řečeno v úvodu, všechny iterační metody se po zvolení počátečního odhadu řešení postupně přibližují přesnému řešení. V iteraci k tedy metody předpokládají znalost $\mathbf{x}_{k-1}$ a liší se tím, jakým způsobem volí $\mathbf{x}_k$. Gradientní metody hledají $\mathbf{x}_k$ pomocí předpisu

$$\mathbf{x}_k := \mathbf{x}_{k-1} + \gamma_{k-1}\mathbf{p}_{k-1}.$$

Výše uvedený tvar aproximace řešení $\mathbf{x}_k$ se tedy objevuje i v algoritmu A1. Princip gradientních metod spočívá v nalezení nové aproximace řešení $\mathbf{x}_k$ na přímce dané bodem $\mathbf{x}_{k-1}$ a směrovým vektorem $\mathbf{p}_{k-1}$. Tato aproximace je poté vybrána tak, aby $J(\mathbf{x}_k)$ bylo co nejmenší,

to znamená

$$J(\mathbf{x}_{k-1} + \gamma_{k-1}\mathbf{p}_{k-1}) = \min_{\gamma} J(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}).$$

Konkrétní tvar parametru γ je tedy určen tak, aby derivace $J(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1})$ podle γ byla nulová

$$\begin{aligned}\frac{\partial J}{\partial \gamma}\left(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}\right) &= \frac{\partial}{\partial \gamma}\left[\frac{1}{2}\left(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}\right)^T \mathbf{A}\left(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}\right) - \left(\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}\right)^T \mathbf{b}\right] = \\ &= \frac{1}{2}\mathbf{x}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1} + \frac{1}{2}\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{x}_{k-1} + \gamma\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1} - \mathbf{p}_{k-1}^T\mathbf{b} = \\ &= \mathbf{p}_{k-1}^T\left(\mathbf{A}\mathbf{x}_{k-1} - \mathbf{b}\right) + \gamma\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1} = 0.\end{aligned}$$

Odvodili jsme tedy parametr tvaru

$$\gamma_{k-1} := \frac{\mathbf{p}_{k-1}^T\mathbf{r}_{k-1}}{\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}} \tag{1.4}$$

a dalšími algebraickými úpravami lze získat tvar A1:(5).

Jednotlivé gradientní metody se liší tím, jak volí směrové vektory $\mathbf{p}_k$. Pokud jsou zvoleny tak, aby byly sdružené ($\mathbf{A}$-ortogonální)

$$\mathbf{p}_i^T\mathbf{A}\mathbf{p}_j = 0, \quad i, j = 0, 1, ..., k, \quad i \neq j \tag{1.5}$$

vznikne metoda sdružených směrů. Tato vlastnost směrových vektorů je důležitá, protože z ní vyplývá $\mathbf{A}$-ortogonalita chyby $\mathbf{e}_k$ ke směrovým vektorům $\mathbf{p}_i$, tj.

$$\langle \mathbf{e}_k, \mathbf{p}_i \rangle_{\mathbf{A}} = 0, \quad i = 0, 1, \ldots, k-1,$$

což je podmínkou minimality $\mathbf{A}$-normy chyby $\|\mathbf{e}_k\|_{\mathbf{A}}$. Princip odvození spočívá ve vyjádření chyby ve tvaru

$$\mathbf{e}_k = \mathbf{e}_0 - \sum_{i=1}^{k} \gamma_i \mathbf{p}_i$$

a lze ho nalézt v [24]. Dalším důležitým důsledkem (1.5) je, že algoritmy těchto metod naleznou řešení soustavy rovnic (minimum funkcionálu) nejvýše po N krocích, kde N je velikost matice $\mathbf{A}$.

Uchovávat všechny směrové vektory ale může být paměťově náročné. Proto byla metoda sdružených gradientů navržena tak, aby $\mathbf{A}$-ortogonalitu $\mathbf{p}_k$ ke všem předchozím směrovým vektorům zajistila pouze pomocí $\mathbf{p}_{k-1}$ a $\mathbf{r}_k$. CG nový směrový vektor počítá pomocí předpisu

$$\mathbf{p}_k := \mathbf{r}_k + \delta_k\mathbf{p}_{k-1},$$

kde tvar koeficientu δ_k zajišťuje lokální **A**-ortogonalitu. Musí tedy platit

$$\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_k = \mathbf{p}_{k-1}^T\mathbf{A}(\mathbf{r}_k + \delta_k\mathbf{p}_{k-1}) = \mathbf{p}_{k-1}^T\mathbf{A}\mathbf{r}_k + \delta_k\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1} = 0, \tag{1.6}$$

$$\delta_k := -\frac{\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{r}_k}{\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}}. \tag{1.7}$$

Globální **A**-ortogonalitu směrových vektorů lze pak dokázat pomocí indukce, viz. podkapitola 1.6. Jak získat rovnice určující parametry γ_{k-1} a δ_k, které se objevují v algoritmu A1, lze najít např. v [22].

1.3 CG jako projektivní metoda

CG lze chápat i jako projektivní metodu. Algoritmus totiž ve svém průběhu vytváří posloupnost Krylovových podprostorů, ve kterých hledá nejlepší aproximace řešení. To znamená, že v těchto prostorech najde vektory, pro něž je **A**-norma chyby nejmenší. Aproximace řešení je projekcí přesného řešení do prostoru, který algoritmus vytváří. Výše zmíněný prostor je k-tý Krylovův podprostor generovaný maticí **A** a vektorem $\mathbf{r}_0$

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) \equiv \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}. \tag{1.8}$$

Tento prostor se během výpočtů CG zvětšuje, dokud algoritmus nenalezne přesné řešení. Největší možná dimenze podprostorů (1.8) úzce souvisí s počtem nenulových složek rezidua $\mathbf{r}_0$, který je vyjádřený v ortogonální bázi vlastních vektorů matice **A**. Platí, že s rostoucím počtem nulových složek $\mathbf{r}_0$ klesá počet iterací algoritmu potřebných k dosažení přesného řešení a tedy i největší dimenze, které může posloupnost Krylovových podprostorů dosáhnout. Nejvýše pak ale hodnoty N.

Metoda sdružených gradientů je z pohledu projektivních metod popsána následujícím způsobem

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k-1}\gamma_i\mathbf{p}_i \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0),$$

$$\|\mathbf{x} - \mathbf{x}_k\|_\mathbf{A} = \min_{\mathbf{y}\in\mathbf{x}_0+\mathcal{K}_k(\mathbf{A},\mathbf{r}_0)}\|\mathbf{x} - \mathbf{y}\|_\mathbf{A}. \tag{1.9}$$

Tyto vztahy jsou v podstatě matematická interpretace výše zmíněné myšlenky o minimalizaci **A**-normy chyby přes vektory Krylovova prostoru. Podmínka (1.9) je ekvivalentní podmínce

$$\mathbf{x} - \mathbf{x}_k \perp_\mathbf{A} \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0),$$

která říká, že vektor $\mathbf{x} - \mathbf{x}_k$ je **A**-ortogonální k prostoru $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, tzn.

$$(\mathbf{x} - \mathbf{x}_k)^T\mathbf{A}\mathbf{d}_i = 0, \quad i = 1, 2, \ldots, k,$$

kde $\mathbf{d}_i$ tvoří bázi tohoto prostoru.

Jednou z důležitých vlastností prostoru (1.8) je, že jeho ortogonální bázi tvoří vektory $\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1}\}$ a jeho $\mathbf{A}$-ortogonální bázi tvoří $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{k-1}\}$. Princip důkazu tohoto tvrzení lze nalézt např. v [22]. Platí proto důležitý vztah

$$\operatorname{span}\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1}\} = \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = \operatorname{span}\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{k-1}\}.$$

Metodami Krylovových podprostorů se podrobně zabývá [23].

1.4 CG a Gaussova kvadratura

1.4.1 Lanczosův algoritmus

Lanczosova metoda slouží k výpočtu vlastních čísel symetrické matice $\mathbf{A} \in \mathbb{R}^{N\times N}$ a je založena na Lanczosově algoritmu. Vstupem algoritmu je právě tato matice a počáteční vektor $\mathbf{v} \in \mathbb{R}^N$. Lanczosův algoritmus je zde označen A2.

Algoritmus A2 Lanczosův algoritmus

1: **vstup** $\mathbf{A}$, $\mathbf{v}$
2: $\mathbf{v}_0 := \mathbf{o}$
3: $\mathbf{v}_1 := \frac{\mathbf{v}}{\|\mathbf{v}\|}$
4: $\beta_1 := 0$
5: **for** $k = 1, 2 \ldots$
6: $\quad \mathbf{w} := \mathbf{A}\mathbf{v}_k - \beta_k \mathbf{v}_{k-1}$
7: $\quad \alpha_k := \mathbf{v}_k^T \mathbf{w}$
8: $\quad \mathbf{w} := \mathbf{w} - \alpha_k \mathbf{v}_k$
9: $\quad \beta_{k+1} := \|\mathbf{w}\|$
10: $\quad \mathbf{v}_{k+1} := \frac{\mathbf{w}}{\beta_{k+1}}$
11: **end**

Vektory generované tímto algoritmem v k-té iteraci splňují vztah

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k \mathbf{T}_k + \beta_{k+1} \mathbf{v}_{k+1} \mathbf{e}_{(k)}^T, \tag{1.10}$$

kde matice $\mathbf{T}_k$ je složena z koeficientů α_i a β_i

$$\mathbf{T}_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix}, \tag{1.11}$$

matice $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ je složena z vektorů $\mathbf{v}_k$, resp. sloupce této matice se rovnají jednotlivým vektorům, a $\mathbf{e}_{(k)} = [0, 0, \ldots, 0, 1]^T \in \mathbb{R}^k$.

Po rozepsání (1.10) vyplývá, že nový směrový vektor $\mathbf{v}_{k+1}$ se počítá pomocí tříčlenné rekurence

$$\beta_{k+1}\mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k - \alpha_k\mathbf{v}_k - \beta_k\mathbf{v}_{k-1}. \tag{1.12}$$

Během chodu tohoto algoritmu vzniká ortonormální báze Krylovových podprostorů

$$\mathcal{K}_k(\mathbf{A}, \mathbf{v}) \equiv \text{span}\{\mathbf{v}, \mathbf{A}\mathbf{v}, \mathbf{A}^2\mathbf{v}, \ldots, \mathbf{A}^{k-1}\mathbf{v}\}. \tag{1.13}$$

Tyto podprostory se budou zvětšovat až do iterace $m \leq N$, pro kterou bude člen $\beta_{m+1}\mathbf{v}_{m+1}\mathbf{e}_{(m)}^T$ rovnice (1.10) roven nule, tzn. bude platit

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_m\mathbf{T}_m. \tag{1.14}$$

Vlastní čísla $\mathbf{T}_m$ se pak budou rovnat vlastním číslům matice $\mathbf{A}$ a vlastní vektory $\mathbf{T}_m$ budou určovat vlastní vektory $\mathbf{A}$ pomocí součinu $\mathbf{V}_m\mathbf{Q}$, kde $\mathbf{Q}$ je matice vlastních vektorů $\mathbf{T}_m$. Toto tvrzení plyne z následující úpravy výrazu (1.14). Nechť $\mathbf{T}_m = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ je spektrální rozklad matice $\mathbf{T}_m$. Pak platí

$$\begin{aligned}\mathbf{A}\mathbf{V}_m &= \mathbf{V}_m\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T,\\ \mathbf{A}\mathbf{V}_m\mathbf{Q} &= \mathbf{V}_m\mathbf{Q}\mathbf{\Lambda}.\end{aligned}$$

V [22] je ukázáno, že pokud nebude splněna rovnice (1.14), pro vlastní čísla $\mu_1, \mu_2, \ldots, \mu_k$ a vlastní vektory $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$ matice $\mathbf{T}_k$ platí

$$\|\mathbf{A}\mathbf{z}_i - \mu_i\mathbf{z}_i\| = \beta_{k+1}|\mathbf{e}_{(k)}^T\mathbf{q}_i|, \quad i = 1, 2, \ldots, k, \tag{1.15}$$

kde $\mathbf{z}_i \equiv \mathbf{V}_k\mathbf{q}_i$. Na tomto vztahu je založen princip Lanczosovy metody, který spočívá v určení aproximací vlastních čísel matice $\mathbf{A}$ jako vlastních čísel matice $\mathbf{T}_k$ (*Ritzova čísla*). A za aproximace vlastních vektorů $\mathbf{A}$ považuje vektory $\mathbf{z}_i$ (*Ritzovy vektory*).

Navíc pomocí pravé strany (1.15) lze kontrolovat, jak moc jsou Ritzova čísla vzdálena od skutečných vlastních čísel $\lambda_1, \lambda_2, \ldots, \lambda_N$ matice $\mathbf{A}$. Platí totiž

$$\min_{j=1,2,\ldots,N} |\lambda_j - \mu_i| \leq \beta_{k+1}|\mathbf{e}_{(k)}^T\mathbf{q}_i|, \quad i = 1, 2, \ldots, k.$$

Důkaz tohoto tvrzení lze opět nalézt v [22].

1.4.2 Vztah Lanczosova algoritmu s CG

Lanczosova metoda souvisí s metodou sdružených gradientů velmi úzce. Platí totiž, že pokud jako počáteční vektor Lanczosova algoritmu zvolíme

$$\mathbf{v} = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|}, \tag{1.16}$$

oba algoritmy budou ve svém průběhu vytvářet stejné Krylovovy podprostory, to znamená

$$\mathcal{K}_k(\mathbf{A}, \mathbf{v}) = \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0), \quad k = 1, 2, ..., m,$$

kde $\mathcal{K}_k(\mathbf{A}, \mathbf{v})$ je dáno předpisem (1.13) a $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ (1.8). Parametr m je stejně jako v předchozí části určen iterací, ve které Lanczosova metoda nalezne všechna vlastní čísla nebo CG nalezne přesné řešení.

Krylovův podprostor (1.8) lze kromě již zmíněných bází (viz. podkapitola 1.3) určit i ortonormální bází $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$. Proto aproximace řešení $\mathbf{x}_k$ musí jít vyjádřit pomocí

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k. \tag{1.17}$$

Protože reziduum $\mathbf{r}_k$ je ortogonální k prostoru (1.8), viz. sekce 1.3, platí

$$\begin{aligned} 0 = \mathbf{V}_k^T \mathbf{r}_k = \mathbf{V}_k^T(\mathbf{b} - \mathbf{A}\mathbf{x}_k) = \mathbf{V}_k^T[\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k)] = \mathbf{V}_k^T\mathbf{r}_0 - \mathbf{V}_k^T\mathbf{A}\mathbf{V}_k\mathbf{y}_k = \\ = \|\mathbf{r}_0\|\mathbf{e}_{(1)} - \mathbf{T}_k\mathbf{y}_k. \end{aligned} \tag{1.18}$$

Z toho plyne, že aproximaci řešení $\mathbf{x}_k$ úlohy (1.1) s počátečním odhadem $\mathbf{x}_0$ a příslušným reziduem $\mathbf{r}_0$ můžeme získat pomocí Lanczosova algoritmu s počátečním vektorem (1.16) pomocí vztahu (1.17), kde $\mathbf{y}_k$ je dáno rovnicí (1.18).

A naopak, matici $\mathbf{T}_k$ lze získat z algoritmu metody sdružených gradientů porovnáním rekurenčních koeficientů z A1 a A2:

$$\alpha_{k+1} = \frac{1}{\gamma_k} + \frac{\delta_{k-1}}{\gamma_{k-1}}, \tag{1.19}$$

$$\beta_{k+2} = \frac{\sqrt{\delta_k}}{\gamma_k} \tag{1.20}$$

s počátečními koeficienty $\delta_{-1} \equiv 0$ a $\gamma_{-1} \equiv 1$. Matici $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ pak určují normalizovaná rezidua

$$\mathbf{v}_{j+1} = (-1)^j \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}, \quad j = 0, 1, \ldots, k-1. \tag{1.21}$$

Odvození těchto vztahů lze nalézt např. v [13, 22] a je založeno na vyjádření rezidua $\mathbf{r}_k$ pomocí tříčlenné rekurence (eliminací směrového vektoru $\mathbf{p}_k$) a jejím srovnáním s tříčlennou rekurencí (1.12). Z tohoto odvození také vyplývá, že metoda sdružených gradientů pomocí koeficientů γ_i a δ_i počítá Choleského rozklad matice $\mathbf{T}_k$, tedy

$$\mathbf{T}_k = \mathbf{L}_k\mathbf{L}_k^T,$$

kde

$$\mathbf{L}_k = \begin{bmatrix} \frac{1}{\sqrt{\gamma_0}} & & & \\ \sqrt{\frac{\delta_1}{\gamma_0}} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\frac{\delta_{k-1}}{\gamma_{k-2}}} & \frac{1}{\sqrt{\gamma_{k-1}}} \end{bmatrix}.$$

1.4.3 Gaussova kvadratura (GQ)

Nejprve je potřeba naznačit, co to Gaussova kvadratura (GQ) vlastně je (GQ = Gauss Quadrature). Jde o matematický nástroj sloužící k numerické aproximaci hodnoty integrálu. V této práci budeme aproximovat Riemannův-Stieltjesův integrál spojité funkce $f(\xi)$.

Definice 1. *Riemannův-Stieltjesův integrál reálné funkce $f(\xi)$ na intervalu $\langle a, b\rangle$ vzhledem k reálné distribuční funkci $\omega(\xi)$ je označen*

$$\int_a^b f(\xi)\mathrm{d}\omega(\xi) \tag{1.22}$$

a definován následující limitou (pokud existuje)

$$\lim_{\|\mathbf{D}\|\to 0} \sum_{j=1}^{n} f(c_j)[\omega(\xi_{j+1}) - \omega(\xi_j)], \tag{1.23}$$

kde $\mathbf{D} \equiv \{\xi_0, \xi_1, \ldots, \xi_n\}$ je dělení intervalu $\langle a, b\rangle$, kde $a = \xi_0 \leq \xi_1 \leq \ldots \leq \xi_n = b$, $c_j \in \langle \xi_j, \xi_{j+1}\rangle$ a norma $\mathbf{D}$ je definována předpisem

$$\|\mathbf{D}\| \equiv \max_{j=1,2,\ldots,n} (\xi_j - \xi_{j-1}).$$

Tento integrál existuje, pokud $f(\xi)$ je spojitá funkce a pro funkci $\omega(\xi)$ existuje konstanta M taková, že pro libovolné dělení $\mathbf{D}$ intervalu $\langle a, b\rangle$ platí:

$$\sum_{k=1}^{n} |\omega(\xi_k) - \omega(\xi_{k-1})| \leq M.$$

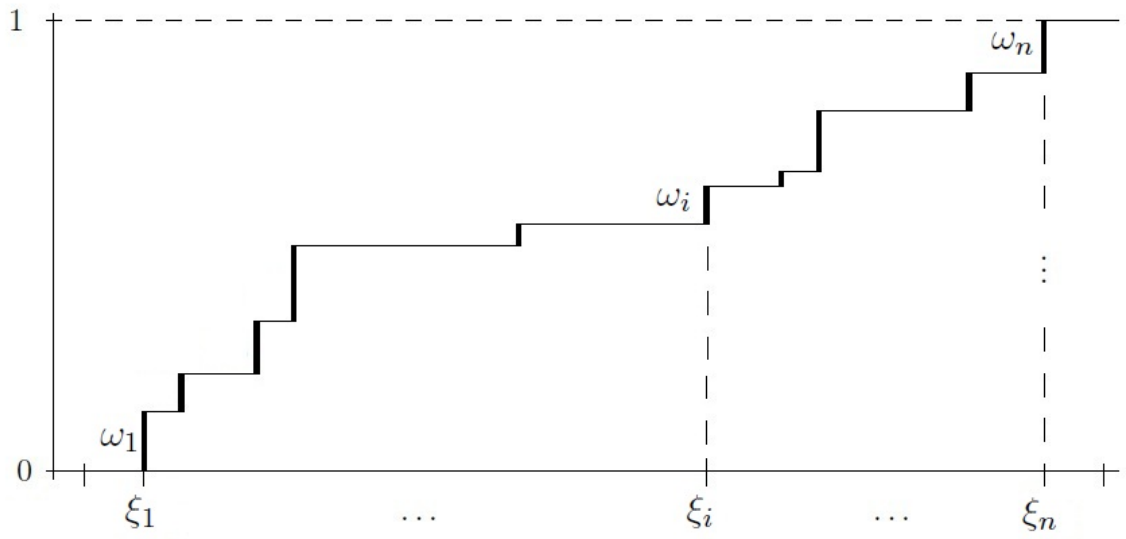
Funkce splňující tuto vlastnost se označuje jako funkce s *omezenou variací*. Pro existenci integrálu ale stačí i opačná podmínka, tzn. funkce $f(\xi)$ má omezenou variaci a funkce $\omega(\xi)$ je spojitá. Je tedy patrné, že integrál (1.22) je obecnější než klasický Riemannův integrál. Navíc za podmínky hladkosti funkce $\omega(\xi)$ platí

$$\int_a^b f(\xi)\mathrm{d}\omega(\xi) = \int_a^b f(\xi)\omega'(\xi)\mathrm{d}\xi.$$

Nás ale budou zajímat případy, kdy funkce $\omega(\xi)$ je po částech konstantní, viz. obr. 1.1. Výraz (1.23) definující integrál se díky tomu zredukuje na

$$\sum_{i=1}^{n} \omega_i f(\xi_i),$$

kde ξ_i je hodnota, ve které má funkce $\omega(\xi)$ skok a ω_i je výška tohoto skoku.



Obrázek 1.1: Náčrt distribuční funkce $\omega(\xi)$.

Gaussova kvadratura aproximuje integrál (1.22) pomocí sumy

$$\sum_{i=1}^{k} \omega_i^k f(\xi_i^k),$$

kde $a < \xi_1^k < \xi_2^k < \ldots < \xi_k^k < b$ jsou tzv. uzly a ω_i^k jsou tzv. váhy Gaussovy kvadratury. Podrobné informace o Gaussově kvadratuře lze nalézt např. v [1, 19].

1.4.4 Souvislost mezi GQ a CG

Vztah mezi GQ a CG je důležitý, protože v dalších kapitolách budeme na problém chování CG v konečné aritmetice nahlížet z pohledu Gaussovy kvadratury. Tento vztah byl popsán už v původní práci [9] z roku 1952. Odvození souvislosti lze nalézt v [13, 22] a jeho náznak je uveden níže.

Nechť je dán skalární součin

$$\langle f(\lambda), g(\lambda)\rangle_\omega = \int_a^b f(\lambda)g(\lambda)\mathrm{d}\omega(\lambda), \tag{1.24}$$

kde $\omega(\lambda)$ je neklesající a nezáporná funkce. Z teorie ortogonálních polynomů (viz. [2]) a GQ vyplývá, že pomocí tříčlenné rekurence lze sestrojit posloupnost polynomů, které jsou vůči tomuto skalárnímu součinu ortogonální. Z koeficientů této rekurence lze sestavit tridiagonální matici Θ, jejíž vlastní čísla a první komponenty vlastních vektorů určují uzly a váhy Gaussovy kvadratury aproximující integrál, pomocí něhož byl skálární součin určen.

Nechť je dána symetrická pozitivně definitní matice $\mathbf{A}$. Tuto matici můžeme vyjádřit ve tvaru

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \quad \mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I},$$

kde $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ je unitární matice se sloupci odpovídajícími vlastním vektorům $\mathbf{A}$ a $\mathbf{\Lambda}$ je diagonální matice, jejíž prvky $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ jsou vlastní čísla $\mathbf{A}$.

Při výpočtu metodou sdružených gradientů získáme reziduum $\mathbf{r}_k$, které lze díky (1.8) napsat ve tvaru polynomu s proměnnou $\mathbf{A}$

$$\mathbf{r}_k = p_k(\mathbf{A})\mathbf{r}_0,$$

kde $p_k(\mathbf{A})$ je polynom stupně k. Ze vzájemné ortogonality reziduí vyplývá ortogonalita polynomů p_k vůči skalárnímu součinu

$$\langle f(\lambda), g(\lambda)\rangle_\omega = \sum_{i=1}^{N} \omega_i f(\lambda_i)g(\lambda_i). \tag{1.25}$$

Váhy ω_i jsou určeny

$$\omega_i = (\mathbf{v}, \mathbf{u}_i)^2, \tag{1.26}$$

kde $\mathbf{u}_i$ jsou sloupce výše zmíněné matice $\mathbf{U}$ a $\mathbf{v} = \mathbf{r}_0.\|\mathbf{r}_0\|^{-1}$.

Posloupnost polynomů ortogonálních vůči tomuto skalárnímu součinu lze získat i Lanczosovým algoritmem aplikovaným na matici $\mathbf{A}$ a vektor $\mathbf{v}$.

Skalární součin (1.25) lze chápat jako skalární součin ve tvaru (1.24), pokud jako distribuční funkci $\omega(\lambda)$ uvažujeme funkci definovanou

$$\omega(\lambda) = \begin{cases} 0 & \lambda < \lambda_1 \\ \sum_{i=1}^{j} \omega_i & \lambda_j \le \lambda < \lambda_{j+1} \\ 1 & \lambda_N \le \lambda \end{cases}. \tag{1.27}$$

Tato funkce je navíc nezáporná a neklesající, a proto víme, že polynomy $p_k(\mathbf{A})$ vzniklé

během chodu algoritmu CG (Lanczosova algoritmu) určují uzly a váhy Gaussovy kvadratury aproximující integrál (1.22), kde $\omega(\lambda)$ je dána (1.27). Nyní zbývá doplnit, jak se tyto parametry určují.

Nechť $\mathbf{T}_k$ je tridiagonální matice (1.11) získaná Lanczosovým algoritmem v iteraci k. Tuto matici lze získat i algoritmem CG pomocí vztahů (1.19) a (1.20). $\mathbf{T}_k$ odpovídá matici Θ zmíněné na začátku této sekce.

Nyní je potřeba provést spektrální rozklad matice

$$\mathbf{T}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{U}_k^T, \quad \mathbf{U}_k \mathbf{U}_k^T = \mathbf{U}_k^T \mathbf{U}_k = \mathbf{I} \tag{1.28}$$

a díky němu spočítat uzly a váhy GQ. Uzly λ_i^k se budou rovnat vlastním číslům matice $\mathbf{\Lambda}_k$ a pro koeficienty ω_i^k bude platit

$$\omega_i^k = (\mathbf{e}_{(1)}, \mathbf{u}_i^k)^2, \tag{1.29}$$

kde $\mathbf{u}_i^k$ jsou sloupce $\mathbf{U}_k$ a $\mathbf{e}_{(1)} = [1, 0, \dots 0] \in \mathbb{R}^k$.

Gaussovu kvadraturu aproximující Riemannův-Stieltjesův integrál spojité funkce $f(\lambda)$ vůči funkci $\omega(\lambda)$ lze tedy zapsat ve tvaru

$$\int_a^b f(\lambda) \mathrm{d}\omega(\lambda) \approx \sum_{i=1}^{k} \omega_i^k f(\lambda_i^k), \tag{1.30}$$

kde navíc pravá strana rovnice definuje integrál spojité funkce $f(\lambda)$

$$\int_a^b f(\lambda) \mathrm{d}\omega^k(\lambda)$$

vzhledem k váhové funkci

$$\omega^k(\lambda) = \begin{cases} 0 & \lambda < \lambda_1^k \\ \sum_{i=1}^{j} \omega_i^k & \lambda_j^k \leq \lambda < \lambda_{j+1}^k \\ 1 & \lambda_k^k \leq \lambda \end{cases} . \tag{1.31}$$

Platí tedy

$$\int_a^b f(\lambda) \mathrm{d}\omega(\lambda) \approx \int_a^b f(\lambda) \mathrm{d}\omega^k(\lambda) = \sum_{i=1}^{k} \omega_i^k f(\lambda_i^k).$$

Z toho plyne, že metoda sdružených gradientů jednoznačně určuje Gaussovu kvadraturu integrálu (1.22). Další informace o souvislosti CG a GQ lze nalézt v [12, 14, 21, 23].

1.5 Další algoritmické realizace CG

1.5.1 Rutishauserova a Hagemanova-Youngova verze

Algoritmus A1 je nejznámější algoritmická verze metody sdružených gradientů. Není ale jediná možná. Už v části 1.4.1 je příklad jiné realizace metody sdružených gradientů. K výpočtu aproximace řešení se totiž může použít Lanczosův algoritmus A2 aplikovaný na matici $\mathbf{A}$ a startovací vektor (1.16) společně se vztahy (1.17) a (1.18).

Algoritmus A3 Rutishauser

vstup $\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\Delta\mathbf{r}_{-1} := \mathbf{o}$
$\Delta\mathbf{x}_{-1} := \mathbf{o}$
$\eta_{-1} := 0$
for $k = 1, 2 \ldots$
  $\tau_{k-1} := \frac{\mathbf{r}_{k-1}^T \mathbf{A}\mathbf{r}_{k-1}}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}} - \eta_{k-2}$
  $\Delta\mathbf{r}_{k-1} := \frac{-\mathbf{A}\mathbf{r}_{k-1} + \Delta\mathbf{r}_{k-2}\eta_{k-2}}{\tau_{k-1}}$
  $\Delta\mathbf{x}_{k-1} := \frac{\mathbf{r}_{k-1} + \Delta\mathbf{x}_{k-2}\eta_{k-2}}{\tau_{k-1}}$
  $\mathbf{r}_k := \mathbf{r}_{k-1} + \Delta\mathbf{r}_{k-1}$
  $\mathbf{x}_k := \mathbf{x}_{k-1} + \Delta\mathbf{x}_{k-1}$
  $\eta_{k-1} := \tau_{k-1} \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}$
end

Algoritmus A4 Hageman - Young

vstup $\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{r}_{-1} := \mathbf{o}$
$\mathbf{x}_{-1} := \mathbf{o}$
$\lambda_0 := \frac{\mathbf{r}_0^T \mathbf{r}_0}{\mathbf{r}_0^T \mathbf{A}\mathbf{r}_0}$
$\rho_0 := 1$
for $k = 1, 2 \ldots$
  $\mathbf{r}_k := \rho_{k-1}(-\lambda_{k-1}\mathbf{A}\mathbf{r}_{k-1} + \mathbf{r}_{k-1}) + (1 - \rho_{k-1})\mathbf{r}_{k-2}$
  $\mathbf{x}_k := \rho_{k-1}(\lambda_{k-1}\mathbf{r}_{k-1} + \mathbf{x}_{k-1}) + (1 - \rho_{k-1})\mathbf{x}_{k-2}$
  $\lambda_k := \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{A}\mathbf{r}_k}$
  $\rho_k := \left(1 - \frac{\lambda_k}{\lambda_{k-1}} \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}} \frac{1}{\rho_{k-1}}\right)^{-1}$
end

Dalšími možnostmi jsou např. verze založená na tříčlenných rekurencích nebo tzv. Rutishauserova (R) verze. Algoritmus založený na tříčlenných rekurencích lze nalézt např. v knize [8], a proto zde tato verze bude označena podle jmen autorů této knihy, tedy Hagemanova-

Youngova (HY). Její algoritmus je zde označen A4. Rutishauserova varianta daná algoritmem A3 byla publikována už v roce 1959 v [18].

Důkazy, že tyto verze jsou ekvivalentní s původní verzí CG jsou těžko k nalezení, a proto jsou v následující části práce uvedeny. Otázkou různé realizace metody sdružených gradientů se zabývá i článek [17].

1.5.2 Důkazy ekvivalence algoritmů

Než začneme s důkazy ekvivalence jednotlivých algoritmů, upravíme výraz $\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k$ do vhodného tvaru. Úprava je založena na rovnicích A1:(9) a (1.7):

$$\begin{aligned}\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k &= \left(\mathbf{r}_k+\delta_k\mathbf{p}_{k-1}\right)^T\mathbf{A}\left(\mathbf{r}_k+\delta_k\mathbf{p}_{k-1}\right)= \\ &= \mathbf{r}_k^T\mathbf{A}\mathbf{r}_k+2\delta_k\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{r}_k+\delta_k^2\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}= \\ &= \mathbf{r}_k^T\mathbf{A}\mathbf{r}_k-2\delta_k^2\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}+\delta_k^2\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}= \\ &= \mathbf{r}_k^T\mathbf{A}\mathbf{r}_k-\delta_k^2\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}. \end{aligned} \tag{1.32}$$

Vztah (1.32) bude použit v obou následujících důkazech.

Věta 1. *Rutishauserova verze metody sdružených gradientů daná algoritmem A3 je ekvivalentní Hestenesově-Stiefelově verzi dané algoritmem A1.*

Důkaz: Důkaz spočívá nejprve v odvození algoritmu A3 z algoritmu A1 a to vyřazením směrových vektorů.

i) Odvození reziduových vektorů $\mathbf{r}_k$ vychází z rovnice A1:(9):

$$\mathbf{A}\mathbf{p}_k=\mathbf{A}\mathbf{r}_k+\delta_k\mathbf{A}\mathbf{p}_{k-1},$$

$$\frac{\mathbf{r}_k-\mathbf{r}_{k+1}}{\gamma_k}=\mathbf{A}\mathbf{r}_k+\frac{\delta_k}{\gamma_{k-1}}\left(\mathbf{r}_{k-1}-\mathbf{r}_k\right),$$

$$\mathbf{r}_{k+1}=\mathbf{r}_k+\gamma_k\left[-\mathbf{A}\mathbf{r}_k+\frac{\delta_k}{\gamma_{k-1}}\left(\mathbf{r}_k-\mathbf{r}_{k-1}\right)\right]= \tag{1.33}$$

$$=\mathbf{r}_k+\Delta\mathbf{r}_k, \tag{1.34}$$

$$\tau_k:=\gamma_k^{-1},$$

$$\eta_{k-1}:=\delta_k\gamma_{k-1}^{-1},$$

$$\Delta\mathbf{r}_k = \frac{1}{\tau_k}\left[-\mathbf{A}\mathbf{r}_k + \eta_{k-1}\left(\mathbf{r}_k - \mathbf{r}_{k-1}\right)\right] = \\ = \frac{1}{\tau_k}\left(-\mathbf{A}\mathbf{r}_k + \eta_{k-1}\Delta\mathbf{r}_{k-1}\right). \tag{1.35}$$

Nyní již rovnice (1.34) a (1.35) odpovídají těm uvedeným v algoritmu. K odvození bylo zatím potřeba jen algebraických úprav založených na rovnicích z A1 a definování nových koeficientů τ_k a η_{k-1}.

ii) Odvození aproximace řešení $\mathbf{x_k}$ je velmi podobné předchozímu odvození $\mathbf{r_k}$. Dosazením A1:(6) do A1:(9) a algebraickou úpravou vzniknou vztahy, které po dosazení nově definovaných koeficientů získají tvar A3:(11) a A3:(9).

iii) Nyní je potřeba dokázat, že pomocí vztahů z A1 je možné koeficienty τ_k a η_{k-1} vyjádřit ve tvaru A3:(7) a A3:(12):

$$\tau_k = \gamma_k^{-1} = \frac{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}{\mathbf{r}_k^T\mathbf{r}_k} = \frac{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{r}_k} - \delta_k^2\frac{\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}}{\mathbf{r}_k^T\mathbf{r}_k} = \frac{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{r}_k} - \delta_k\frac{\mathbf{p}_{k-1}^T\mathbf{A}\mathbf{p}_{k-1}}{\mathbf{r}_{k-1}^T\mathbf{r}_{k-1}} = \\ = \frac{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{r}_k} - \delta_k\gamma_{k-1}^{-1} = \frac{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{r}_k} - \eta_{k-1},$$

$$\eta_{k-1} = \delta_k\gamma_{k-1}^{-1} = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_{k-1}^T\mathbf{r}_{k-1}}\tau_{k-1}.$$

Tato úprava koeficientů byla kromě vztahů z obou algoritmů založena ještě na rovnici (1.32). Je tedy ukázáno, že Rutishauserovu verzi lze odvodit z klasické verze. HS verze lze z R verze odvodit přesně opačným postupem. Obě verze jsou tedy ekvivalentní. □

Věta 2. *Hagemanova-Youngova verze metody sdružených gradientů daná algoritmem A4 je ekvivalentní Hestenesově-Stiefelově verzi dané algoritmem A1.*

Důkaz: Odvození HY varianty metody sdružených gradientů z HS verze spočívá v podobném postupu použitým v předchozím důkazu, a proto jeho první část bude z tohoto důkazu vycházet.

i) Lehce upravená rovnice (1.33)

$$\mathbf{r}_{k+1} = -\gamma_k\mathbf{A}\mathbf{r}_k + \left(1 + \frac{\gamma_k\delta_k}{\gamma_{k-1}}\right)\mathbf{r}_k - \frac{\gamma_k\delta_k}{\gamma_{k-1}}\mathbf{r}_{k-1}$$

po definování koeficientů

$$\rho_k := 1 + \frac{\gamma_k\delta_k}{\gamma_{k-1}},$$

$$\lambda_k := \gamma_k\left(1 + \frac{\gamma_k\delta_k}{\gamma_{k-1}}\right)^{-1}$$

získá výsledný tvar A4:(8).

ii) Postupem popsaným v druhé části předchozího důkazu lze získat rovnici

$$\mathbf{x_{k+1}} = \gamma_k \mathbf{r}_k + \left(1 + \frac{\gamma_k \delta_k}{\gamma_{k-1}}\right) \mathbf{x}_k - \frac{\gamma_k \delta_k}{\gamma_{k-1}} \mathbf{x}_{k-1},$$

která po dosazení nových koeficientů odpovídá A4:(9).

iii) Nyní je potřeba dokázat, že koeficienty λ_k a ρ_k jdou vyjádřit ve tvaru A4:(10) a A4:(11):

$$\begin{aligned}
\rho_k &= 1 + \frac{\gamma_k \delta_k}{\gamma_{k-1}} = 1 + \frac{\delta_k}{\gamma_{k-1}} \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} = \frac{\gamma_{k-1} \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k + \delta_k \mathbf{r}_k^T \mathbf{r}_k}{\gamma_{k-1} \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} = \\
&= \frac{\gamma_{k-1} \mathbf{r}_k^T \mathbf{A} \mathbf{r}_k - \gamma_{k-1} \delta_k^2 \mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1} + \delta_k \mathbf{r}_k^T \mathbf{r}_k}{\gamma_{k-1} \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} = \frac{\gamma_{k-1} \mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}{\gamma_{k-1} \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} = \left(\frac{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}\right)^{-1} = \\
&= \left(1 - \delta_k^2 \frac{\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}\right)^{-1} = \left(1 - \delta_k \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}} \frac{\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}\right)^{-1} = \\
&= \left(1 - \frac{\delta_k}{\gamma_{k-1}} \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}\right)^{-1} = \left(1 - \frac{\lambda_k}{\lambda_{k-1}} \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}} \frac{1}{\rho_{k-1}}\right)^{-1}, \\
\lambda_k &= \gamma_k \left(1 + \frac{\gamma_k \delta_k}{\gamma_{k-1}}\right)^{-1} = \gamma_k \frac{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k} = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k}.
\end{aligned}$$

Dokázání opačné implikace, tj. odvození HS verze z HY verze, je zřejmé. □

Pokud tedy počítáme přesně, algoritmy jsou ekvivalentí. Při počítání v konečné aritmetice ale tato vlastnost neplatí. Experimentům porovnávajícím chování těchto variant v konečné aritmetice je věnována sekce 2.7.

1.6 Vybrané vlastnosti algoritmů

Vlastnosti, které jsou zde uvedeny a dokázány, jsou výběrem nejdůležitějších vlastností algoritmů CG. Na následující věty se budou další kapitoly práce odkazovat a jejich důkazy jsou zde uvedeny právě z důvodu jejich důležitosti pro zkoumání vlastností CG v konečné aritmetice. V předchozí části je dokázáno, že algoritmy HS, R a HY jsou ekvivalentní, a proto následující vlastnosti CG budou dokázány jen pro původní HS variantu. V sekci 1.2 bylo napsáno, že metoda byla odvozena tak, aby směrové vektory byly **A**-ortogonální. Je tedy potřeba ukázat, že algoritmus tuto vlastnost splňuje. Spolu s tím zde bude dokázána i ortogonalita reziduových vektorů, monotonie **A**-normy chyby a monotonie euklidovské normy chyby.

Věta 3. *Předpokládejme, že* $\mathbf{A} \in \mathbb{R}^{N\times N}$ *je symetrická pozitivně definitní matice a vektory* $\mathbf{r}_k$ *a* $\mathbf{p}_k$ *jsou generované algoritmem A1. Pak vektory* $\mathbf{r}_k$ *jsou ortogonální a vektory* $\mathbf{p}_k$ *jsou* $\mathbf{A}$*-ortogonální, tj.*

$$\mathbf{r}_i^T\mathbf{r}_j = 0, \quad \mathbf{p}_i^T\mathbf{A}\mathbf{p}_j = 0, \quad i,j = 0,1,\ldots,k, \quad i \neq j.$$

Důkaz: Jde o důkaz indukcí.
i) Pro k = 1:

$$\mathbf{r}_1^T\mathbf{r}_0 = (\mathbf{r}_0 - \gamma_0\mathbf{A}\mathbf{p}_0)^T\mathbf{r}_0 = \mathbf{r}_0^T\mathbf{r}_0 - \gamma_0\mathbf{p}_0^T\mathbf{A}\mathbf{r}_0 = \mathbf{r}_0^T\mathbf{r}_0 - \frac{\mathbf{r}_0^T\mathbf{r}_0}{\mathbf{p}_0^T\mathbf{A}\mathbf{p}_0}\mathbf{p}_0^T\mathbf{A}\mathbf{r}_0 = 0.$$

K důkazu ortogonality nultého a prvního rezidua bylo potřeba pouze vztahů z algoritmu, konkrétně (3), (5) a (7) a dále symetrie matice $\mathbf{A}$. $\mathbf{A}$-ortogonalita vektorů

$$\mathbf{p}_1^T\mathbf{A}\mathbf{p}_0 = 0$$

platí, protože tvar δ byl odvozen tak, aby dva po sobě jdoucí směrové vektory byly $\mathbf{A}$-ortogonální, viz. (1.6) a (1.7).
ii) Předpokládejme, že směrové vektory jsou $\mathbf{A}$-ortogonální a reziduové vektory jsou ortogonální až do kroku k (pokud $\mathbf{r}_{k+1} \neq o$), tzn.

$$\mathbf{r}_i^T\mathbf{r}_j = 0, \quad \mathbf{p}_i^T\mathbf{A}\mathbf{p}_j = 0, \quad i,j = 1,2,\ldots,k, \quad i \neq j.$$

iii) Chceme ověřit, že

$$\mathbf{r}_{k+1}^T\mathbf{r}_j = 0, \quad \mathbf{p}_{k+1}^T\mathbf{A}\mathbf{p}_j = 0, \quad j = 0,1,\ldots,k.$$

K tomu bude potřeba vztahů A1:(7), A1:(9), symetrie matice $\mathbf{A}$ a samozřejmě indukčních předpokladů. Pro $j = 1,2,\ldots,k-1$ platí

$$\mathbf{r}_{k+1}^T\mathbf{r}_j = (\mathbf{r}_k - \gamma_k\mathbf{A}\mathbf{p}_k)^T\mathbf{r}_j = \mathbf{r}_k^T\mathbf{r}_j - \gamma_k(\mathbf{A}\mathbf{p}_k)^T\mathbf{r}_j = -\gamma_k\mathbf{p}_k^T\mathbf{A}(\mathbf{p}_j - \delta_j\mathbf{p}_{j-1}) = 0,$$

$$\mathbf{p}_{k+1}^T\mathbf{A}\mathbf{p}_j = (\mathbf{r}_{k+1} + \delta_{k+1}\mathbf{p}_k)^T\mathbf{A}\mathbf{p}_j = \mathbf{r}_{k+1}^T\mathbf{A}\mathbf{p}_j = \mathbf{r}_{k+1}^T\gamma_j^{-1}(\mathbf{r}_j - \mathbf{r}_{j+1}) = 0$$

a pro $j = k$ platí

$$\mathbf{r}_{k+1}^T\mathbf{r}_k = (\mathbf{r}_k - \gamma_k\mathbf{A}\mathbf{p}_k)^T\mathbf{r}_k = \mathbf{r}_k^T\mathbf{r}_k - \gamma_k\mathbf{p}_k^T\mathbf{A}(\mathbf{p}_k - \delta_k\mathbf{p}_{k-1}) = \mathbf{r}_k^T\mathbf{r}_k - \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k = 0.$$

Proč $\mathbf{p}_{k+1}^T\mathbf{A}\mathbf{p}_k = 0$ je popsáno v podkapitole 1.2 v (1.6) a (1.7).

□

Věta 4. *Předpokládejme, že* $\mathbf{A} \in \mathbb{R}^{N\times N}$ *je symetrická pozitivně definitní matice, vektory* $\mathbf{x}_k$ *jsou generovány algoritmem A1 a* $\mathbf{x}$ *je přesné řešení rovnice* $\mathbf{Ax} = \mathbf{b}$*. Pak platí*

$$\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}^2 - ||\mathbf{x} - \mathbf{x}_{k+1}||_{\mathbf{A}}^2 = \gamma_k \|\mathbf{r}_k\|^2 > 0,$$

tj. $\mathbf{A}$*-norma chyby je klesající.*

Důkaz: Důkaz vychází z rovnice A1:(6), ze které vyplývá vztah

$$\|\mathbf{x} - \mathbf{x}_{k+1}\|_{\mathbf{A}}^2 = \|\mathbf{x} - \mathbf{x}_k - \gamma_k \mathbf{p}_k\|_{\mathbf{A}}^2.$$

Nyní, protože vektor $\mathbf{x} - \mathbf{x}_{k+1}$ je $\mathbf{A}$-ortogonální k vektoru $\mathbf{p}_k$ (viz. podkapitola 1.3) lze použít Pythagorovu větu

$$\|\mathbf{x} - \mathbf{x}_k - \gamma_k \mathbf{p}_k\|_{\mathbf{A}}^2 = \|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}^2 + \gamma_k^2 \|\mathbf{p}_k\|_{\mathbf{A}}^2.$$

Odsud již vyplývá dokazovaný vztah

$$\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}^2 - \|\mathbf{x} - \mathbf{x}_{k+1}\|_{\mathbf{A}}^2 = \gamma_k^2 \|\mathbf{p}_k\|_{\mathbf{A}}^2 = \gamma_k \frac{\|\mathbf{r}_k\|^2}{\|\mathbf{p}_k\|_{\mathbf{A}}^2} \|\mathbf{p}_k\|_{\mathbf{A}}^2 > 0.$$

□

Nejen $\mathbf{A}$-norma chyby je klesající. I klasická euklidovská norma chyby je klesající. K důkazu monotonie této normy je potřeba uvést a dokázat následující lemma:

Lemma 1. *Předpokládejme, že* $\mathbf{A} \in \mathbb{R}^{N\times N}$ *je symetrická pozitivně definitní matice a vektory* $\mathbf{p}_k$ *jsou generovány algoritmem A1. Pak platí:*

$$\mathbf{p}_i^T \mathbf{p}_j \geq 0, \quad i, j = 0, 1, \ldots, N.$$

Důkaz: Důkaz je založen na indukci.
i) Pro k = 1:

$$\mathbf{p}_1^T \mathbf{p}_0 = (\mathbf{r}_1 + \delta_1 \mathbf{p}_0)^T \mathbf{p}_0 = \mathbf{r}_1^T \mathbf{p}_0 + \delta_1 \mathbf{p}_0^T \mathbf{p}_0 = \mathbf{r}_1^T \mathbf{r}_0 + \delta_1 \mathbf{p}_0^T \mathbf{p}_0 = \delta_1 \mathbf{p}_0^T \mathbf{p}_0 \geq 0.$$

ii) Indukční předpoklad: $\mathbf{p}_i^T \mathbf{p}_j \geq 0, \quad i, j = 1, 2, \ldots, k.$
iii) K dokázání: $\mathbf{p}_{k+1}^T \mathbf{p}_j \geq 0, \quad j = 1, 2, \ldots, k+1$:

$$\mathbf{p}_{k+1}^T \mathbf{p}_j = (\mathbf{r}_{k+1} + \delta_{k+1} \mathbf{p}_k)^T \mathbf{p}_j = \mathbf{r}_{k+1}^T \mathbf{p}_j + \delta_{k+1} \mathbf{p}_k^T \mathbf{p}_j = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \mathbf{p}_k^T \mathbf{p}_j \geq 0.$$

□

Věta 5. *Předpokládejme, že* $\mathbf{A} \in \mathbb{R}^{N\times N}$ *je symetrická pozitivně definitní matice, vektory* $\mathbf{x}_k$ *jsou generované algoritmem A1 a* $\mathbf{x}$ *je přesné řešení rovnice* $\mathbf{Ax} = \mathbf{b}$*. Pak platí:*

$$\|\mathbf{x} - \mathbf{x}_k\|^2 - \|\mathbf{x} - \mathbf{x}_{k+1}\|^2 > 0.$$

Důkaz. Důkaz je postaven na stejném principu jako v předchozí větě:

$$\begin{aligned}\|\mathbf{x} - \mathbf{x}_k\|^2 - \|\mathbf{x} - \mathbf{x}_{k+1}\|^2 &= 2\gamma_k(\mathbf{x} - \mathbf{x}_k)^T\mathbf{p}_k - \gamma_k^2\|\mathbf{p}_k\|^2 = \\ &= 2\gamma_k\sum_{j=k}^{n}\gamma_j\mathbf{p}_j^T - \gamma_k^2\|\mathbf{p}_k\|^2 = \\ &= 2\gamma_k\sum_{j=k+1}^{n}\gamma_j\mathbf{p}_j^T\mathbf{p}_k > 0.\end{aligned}$$

Protože koeficienty γ_j a výrazy $\mathbf{p}_j^T\mathbf{p}_k$ jsou kladné, celá pravá strana tohoto výrazu je kladná. □

Kapitola 2

Chování CG v konečné aritmetice

2.1 Ztráta ortogonality

Už v původní práci Hestense a Stiefela [9] je uvedeno, že algoritmus CG v konečné aritmetice ztrácí vlastnosti, díky nimž najde řešení nejvýše po N krocích. Zaokrouhlovací chyby totiž v algoritmu A1 hrají významnou roli. Díky nim se velmi rychle ztrácí ortogonalita mezi reziduovými vektory a **A**-ortogonalita mezi směrovými vektory.

Z předchozí kapitoly vyplývá, že podmínka **A**-ortogonality směrových vektorů je velmi důležitá. Algoritmus si vynucuje tuto podmínku v rovnici A1:(9). Je zde patrné, že nový směrový vektor se počítá pouze pomocí vektorů získaných z minulé iterace. Zajistí tak přímo **A**-ortogonalitu k minulému směrovému vektoru a pak pomocí indukce lze dokázat, že je **A**-ortogonální i ke všem minulým směrovým vektorům (viz. podkapitola 1.6, věta 3).

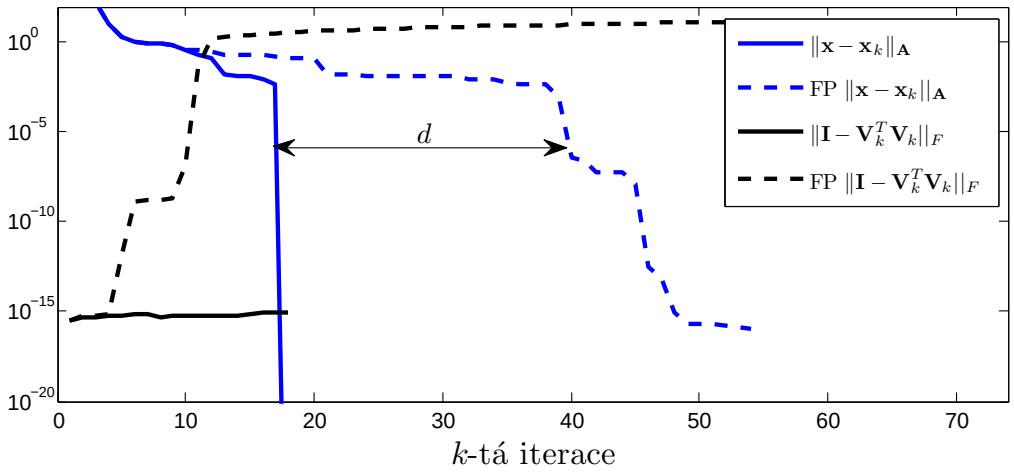
I tato vlastnost, tzn. získávání směrového vektoru pouze pomocí vektorů vypočtených v minulé iteraci, přispívá k rychlému šíření zaokrouhlovacích chyb. Ztrátu ortogonality reziduových vektorů lze vysvětlit podobným způsobem.

Na obr. 2.1 demonstrujeme ztrátu ortogonality i další jevy způsobené zaokrouhlovacími chybami. Porovnávají se zde dvě veličiny získané buď algoritmem A1 počítajícím v konečné aritmetice nebo algoritmem, který simuluje počítání v přesné aritmetice, viz. [6]. Tento algoritmus se od A1 liší tím, že v každém kroku se reziduum dvakrát reortogonalizuje proti všem předchozím reziduím pomocí

$$\mathbf{r}_k = \mathbf{r}_k - \mathbf{V}_k(\mathbf{V}_k^T\mathbf{r}_k),$$
$$\mathbf{r}_k = \mathbf{r}_k - \mathbf{V}_k(\mathbf{V}_k^T\mathbf{r}_k),$$

kde $\mathbf{V}_k$ je matice Lanczosových vektorů (1.21).

FP výpočty tohoto pozměněného algoritmu odpovídají přesným výpočtům algoritmu A1 s relativní chybou řádu 10^{-15}. Proto zde bude označován jako přesný. Algoritmu A1 počítajícím v konečné aritmetice budeme říkat FP algoritmus (FP = Finite Precision).



Obrázek 2.1: **A**-norma chyby $\|\mathbf{x}-\mathbf{x}_k\|_\mathbf{A}$ vypočtená přesným algoritmem (modře) a FP algoritmem (modře přerušovaně), ztráta ortogonality $\|\mathbf{I}-\mathbf{V}_k^T\mathbf{V}_k\|_F$ vypočtená přesným algoritmem (černě) a FP algoritmem (černě přerušovaně). Hodnota d představuje zpoždění konvergence. Inspirováno [22].

Vstupní hodnoty algoritmu pro tento výpočet jsou dány rovnicemi

$$\mathbf{x}_0 = [0, 0, \ldots, 0]^T, \quad \mathbf{b} = \mathbf{A}\mathbf{x}_e, \quad \mathbf{x}_e = [1, 1, \ldots, 1]^T \tag{2.1}$$

a matici **A** (LF10.mtx) lze najít na CD přiloženém k práci a také na webové adrese [10].

Křivka $\|\mathbf{I} - \mathbf{V}_k^T\mathbf{V}_k\|_F$ reprezentuje ztrátu ortogonality reziduových vektorů a je vykreslena černě. Jde o Frobeniovu normu rozdílu jednotkové matice a matice $\mathbf{V}_k^T\mathbf{V}_k$, která by při přesném výpočtu byla jednotková. Modře je zobrazena **A**-norma chyby (1.3). Přerušovaně jsou značeny výsledky FP výpočtů a plnou čarou jsou vykresleny výsledky přesných výpočtů.

2.2 Limitní přesnost

V sekci 1.2 bylo ukázáno, že vhodný způsob jak měřit rozdíl mezi přesným řešením a aproximací řešení je pomocí **A**-normy chyby (1.3). Ale to není jediný způsob. K pozorování konvergence metody lze použít i normu *skutečného* rezidua $\mathbf{r}_k^{(t)}$ (t = true):

$$\|\mathbf{r}_k^{(t)}\| = \|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|, \tag{2.2}$$

přestože narozdíl od **A**-normy a euklidovské normy chyby pro normu rezidua neplatí, že klesá v každém kroku (viz. sekce 1.6, věty 4 a 5).

Zajímavé je, že normu rezidua lze pomocí **A**-normy chyby odhadnout shora i zdola (a také naopak). Platí

$$\sqrt{\lambda_1}\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{A}} \leq \|\mathbf{r}_k^{(t)}\| \leq \sqrt{\lambda_N}\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{A}},$$
$$\frac{1}{\sqrt{\lambda_N}}\|\mathbf{r}_k^{(t)}\| \leq \|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{A}} \leq \frac{1}{\sqrt{\lambda_1}}\|\mathbf{r}_k^{(t)}\|,$$

kde λ_1 (λ_N) je nejmenší (největší) vlastní číslo matice $\mathbf{A}$. Princip odvození jednotlivých nerovností je velmi podobný, a proto zde bude uvedeno pouze jedno odvození:

$$\begin{aligned}\|\mathbf{r}_k^{(t)}\| &= \|\mathbf{A}\mathbf{x}-\mathbf{A}\mathbf{x}_k\| = \|\mathbf{A}^{1/2}\mathbf{A}^{1/2}(\mathbf{x}-\mathbf{x}_k)\| \leq \\ &\leq \|\mathbf{A}^{1/2}\|\|\mathbf{A}^{1/2}(\mathbf{x}-\mathbf{x}_k)\| = \sqrt{\lambda_N}\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{A}}.\end{aligned}$$

Analýzou konvergence metody pomocí reziduí se zabývala A. Greenbaum ve svém článku [5]. Ukázala zde, že nelze očekávat, že norma skutečného rezidua klesne pod určitou hodnotu, tzv. *hladinu limitní přesnosti*.

Tato hladina byla odvozena na základě rozdílu mezi skutečnými a *rekurzivními* rezidui

$$\mathbf{f}_k = \mathbf{b}-\mathbf{A}\mathbf{x}_k-\mathbf{r}_k, \tag{2.3}$$

kde rekurzivní reziduum $\mathbf{r}_k$ je to objevující se v algoritmu, tedy A1:(7). Bylo zde dokázáno, že platí:

$$\|\mathbf{f}_k\| \leq \varepsilon\|\mathbf{A}\|\|\mathbf{x}\|O(k)\left(1+\max_{j\leq k}\frac{\|\mathbf{x}_j\|}{\|\mathbf{x}\|}\right), \tag{2.4}$$

kde ε je strojová přesnost a $O(k)$ je číslo blízké hodnotě k.

Také zde bylo numericky ukázáno, že v průběhu algoritmu rekurzivní rezidua CG konvergují k nulovému vektoru, nebo alespoň jejich norma bude mnohem menší než je strojová přesnost ε. Za určitých podmínek (např. malé číslo podmíněnosti) a pro některé varianty CG lze tato vlastnost i dokázat.

V těchto případech pravá strana (2.4) dává rozumný odhad nejmenších možných hodnot skutečných reziduí, tzn. hladinu limitní přesnosti. Tento jev lze pozorovat i při výpočtu $\mathbf{A}$-normy chyby v konečné aritmetice, jak ukazuje obr. 2.1.

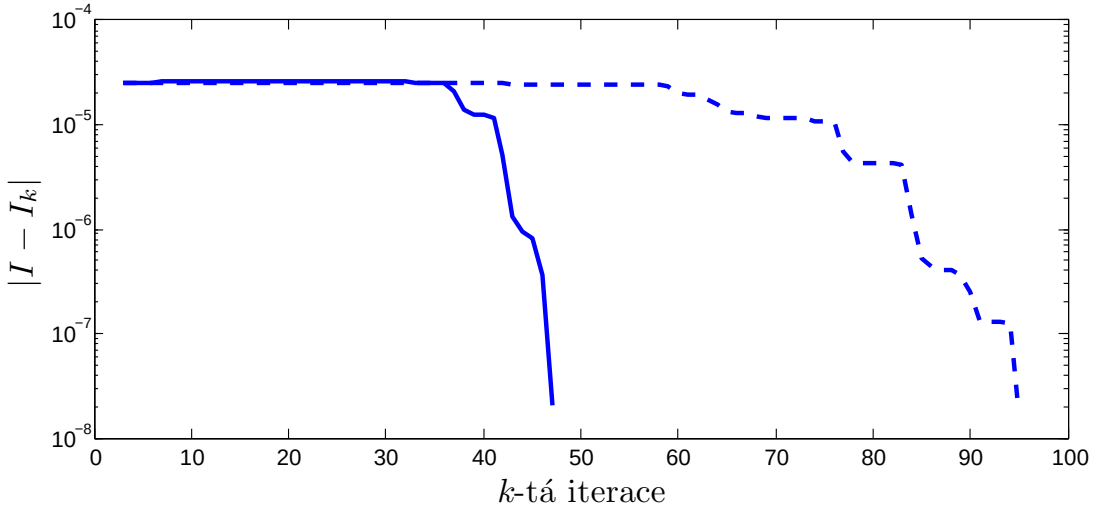
2.3 Vliv konečné aritmetiky na Gaussovu kvadraturu

Z části 1.4.3 vyplývá, že metoda sdružených gradientů a Gaussova kvadratura spolu úzce souvisí. Je tedy očekávatelné, že vliv konečné aritmetiky bude patrný i při počítání Gaussovy kvadratury.

Následující experiment je inspirovaný článkem [15]. Obrázek 2.2 byl vytvořen pomocí přesného i FP algoritmu metody sdružených gradientů aplikovaného na matici $\mathbf{A}$, která se jmenuje bcsstk01.mtx a lze ji najít na přiloženém CD nebo v [11] a vstupní vektory byly

zvoleny následujícím způsobem:

$$\mathbf{b} = [1, 1, \ldots, 1]^T, \quad \mathbf{x}_0 = [0, 0, \ldots, 0]^T. \tag{2.5}$$



Obrázek 2.2: Chyba $E = |I - I_k|$ pro přesný algoritmus (modře) a FP algoritmus (modře přerušovaně). Inspirováno [15].

V grafu je vykreslena chyba Gaussovy kvadratury integrálu (1.22), tedy rozdíl levé a pravé strany rovnice (1.30), pro funkci $f(\lambda) = \lambda^{-1}$ v závislosti na počtu iterací k. Tato chyba je zde označená E a počítá se pomocí vztahu

$$E = |I - I_k|, \quad I = \sum_{i=1}^{N} \frac{\omega_i}{\lambda_i}, \quad I_k = \sum_{i=1}^{k} \frac{\omega_i^k}{\lambda_i^k},$$

kde v případě přesných výpočtů jsou λ_i^k a ω_i^k dány vztahy (1.28) a (1.29) pro $k = 1, 2, \ldots, N-1$ a λ_i a ω_i jsou dány stejnými vztahy pro $k = N$, kde N je velikost matice $\mathbf{A}$. V případě FP výpočtů se λ_i^k a ω_i^k počítá pro $k = 1, 2, \ldots, 2N-1$ a λ_i a ω_i jsou dány koeficientem $k = 2N$.

Na obrázku je patrné, že konečná aritmetika ovlivnila chybu Gaussovy kvadratury integrálu (1.22) funkce $f(\lambda) = \lambda^{-1}$ podobně jako $\mathbf{A}$-normu chyby metody sdružených gradientů.

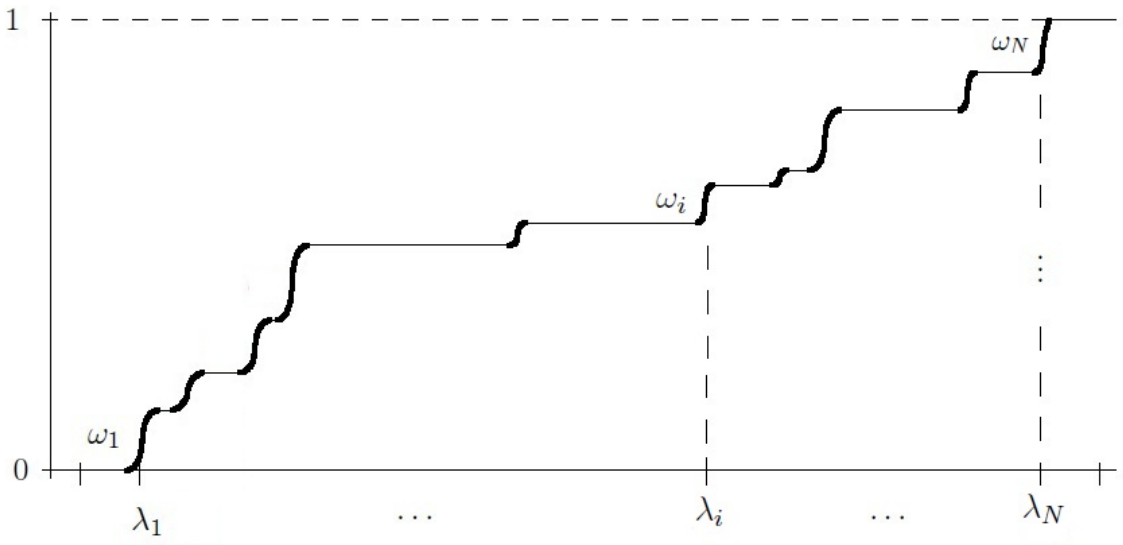
2.4 Vliv na CG z pohledu Gaussovy kvadratury

Bylo pozorováno, že Lanczosův algoritmus provedený v konečné aritmetice často nenalezne všechna vlastní čísla matice $\mathbf{A} \in \mathbb{R}^{N \times N}$ do kroku N. Lanczosovy FP rekurence totiž v některých iteracích místo nové aproximace vlastního čísla naleznou aproximaci velmi blízkou již nalezenému Ritzovu číslu, tzv. *kopii* Ritzova čísla. Vlivem konečné aritmetiky na výpočty

Lanczosova algoritmu se zabývá článek [6]. Jsou zde přehledné grafy ukazující například kolik Ritzových čísel FP algoritmus našel v kroku k a jaké kopie do tohoto kroku vygeneroval.

V části 1.4.4 bylo naznačeno, že při přesném počítání Lanczosův algoritmus generuje posloupnost polynomů, které jsou ortogonální vzhledem ke skalárnímu součinu, který je dán vztahem (1.25) s vahami (1.26) (nebo vztahem (1.24) s distribuční funkcí (1.27)). Platí, že kořeny těchto polynomů odpovídají Ritzovým číslům získaným Lancosovým algoritmem, viz. [2].

Pokud by se tedy aproximace vlastních čísel $\mathbf{A}$ určovaly jako kořeny polynomů získaných pomocí přesného výpočtu CG, kopie Ritzových čísel by se neobjevovaly. Na druhou stranu výskyt kopií je očekávatelný, pokud bychom Ritzova čísla získávali jako kořeny polynomů, které jsou ortogonální vůči skalárnímu součinu, který by byl dán jiným systémem vah. Tento systém by byl větší a jeho váhy by vždy byly rozprostřeny okolo vlastních čísel $\mathbf{A}$. Velikost těchto vah by byla dána tak, aby součet vah blízkých jednomu vlastnímu číslu odpovídal původní váze v tomto vlastním čísle, tedy ω_i. Příklad distribuční funkce sestavené z těchto vah je ukázán na obr. 2.3. Dochází k rozmazání funkce.



Obrázek 2.3: Náčrt distribuční funkce odpovídající výpočtům CG v konečné aritmetice

Posloupnost polynomů ortogonálních vůči skalárnímu součinu daném novým systémem vah $\hat{\omega}_i$ nebo novou distribuční funkcí $\hat{\omega}(\lambda)$ lze získat aplikováním přesného Lanczosova algoritmu na matici $\hat{\mathbf{A}}$, která je větší než původní matice a jejíž vlastní čísla leží uvnitř malých intervalů okolo vlastních čísel původní matice $\mathbf{A}$.

Polynom p_k této posloupnosti pak může mít několik kořenů blízkých jednomu vlastnímu číslu $\mathbf{A}$ a přitom nemít kořeny odpovídající ostatním vlastním číslům $\mathbf{A}$. Tato vlastnost tedy odpovídá vzniku kopií Ritzových čísel u Lanczosova FP algoritmu, a protože těchto polynomů je více než N, nelze od algoritmu CG očekávat, že nalezne řešení po N krocích.

Z těchto úvah vyplývá, že na výpočty FP algoritmu metody sdružených gradientů aplikovaného na soustavu rovnic $\mathbf{Ax} = \mathbf{b}$ lze hledět jako na výpočty přesného algoritmu aplikovaného na větší systém rovnic $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ a také jako na Gaussovu kvadraturu Riemannova-Stieltjesova integrálu

$$\int_a^b f(\lambda)\mathrm{d}\hat{\omega}(\lambda). \tag{2.6}$$

2.5 Konvergence

2.5.1 Zpoždění

Důležitým důsledkem ztráty lineární nezávislosti reziduí je zpoždění konvergence metody sdružených gradientů, které je patrné na obr. 2.1. Toto zpoždění reprezentuje hodnota d představující počet iterací, které FP algoritmus potřebuje navíc, aby dosáhl stejné úrovně poklesu $\mathbf{A}$-normy chyby získané přesným výpočtem. Platí:

$$d \approx m - \mathrm{rank}(\mathbf{V}_m), \tag{2.7}$$

kde m je počet kroků, který potřebuje FP algoritmus, aby dosáhl dané $\mathbf{A}$-normy chyby a $\mathrm{rank}(\mathbf{V}_m)$ je hodnost (počet lineárně nezávislých sloupců) matice $\mathbf{V}_m$ vystupující v (1.10). Odvození tohoto vztahu lze nalézt v [13, 23] a jeho princip je následující.

Rychlost konvergence v přesné aritmetice můžeme napsat ve tvaru

$$\|\mathbf{e}_0\|_{\mathbf{A}}^2 - \|\mathbf{e}_k\|_{\mathbf{A}}^2 = \|\mathbf{r}_0\|^2 \sum_{i=1}^{k} \frac{\omega_i^k}{\lambda_i^k},$$

kde suma na pravé straně odpovídá k-té Gaussově kvadratuře integrálu (1.22) pro $f(\lambda) = \lambda^{-1}$. V konečné aritmetice lze tento vztah vyjádřit pomocí

$$\|\mathbf{e}_0\|_{\mathbf{A}}^2 - \|\mathbf{e}_k\|_{\mathbf{A}}^2 = \|\mathbf{r}_0\|^2 \sum_{i=1}^{k} \frac{\hat{\omega}_i^k}{\hat{\lambda}_i^k} + O(\varepsilon),$$

kde suma představuje k-tou Gaussovou kvadraturu integrálu (2.6) a $O(\varepsilon)$ je číslo úměrné strojové přesnosti.

Hodnoty těchto k-tých kvadratur se ale mohou podstatně lišit, protože Gaussova kvadratura integrálu (2.6) může určit několik uzlů, které jsou velmi blízké již nalezeným uzlům (kopie uzlů). Pokud se tak stane v kroku j, GQ nedává lepší aproximaci integrálu než tu určenou v kroku $j-1$. Z toho vyplývá, že aby si Gaussovy kvadratury integrálů (1.22) a (2.6) odpovídaly, musí GQ integrálu (2.6) udělat d kroků navíc. Hodnota d je určena počtem kopií uzlů.

Nechť $m = d + k$ a matice $\mathbf{V}_m$ je matice Lanczosových vektorů získaná algoritmem provedeným v konečné aritmetice. Pak d můžeme určit jako rozdíl počtu sloupců matice a hodnosti matice, viz. (2.7).

2.5.2 Odhad

Existuje několik prací zabývajících se odhadováním konvergence metody v konečné aritmetice. Např. v [21] je dokázáno, že pokles chyby v konečné aritmetice splňuje vztah

$$\|\mathbf{e}_k\|_{\mathbf{A}}^2 - \|\mathbf{e}_{k+1}\|_{\mathbf{A}}^2 = \gamma_k \|\mathbf{r}_k\|^2 + \varsigma_k, \tag{2.8}$$

kde velikost ς_k závisí na kvalitě zachování ortogonality mezi $\mathbf{r}_{k+1}$ a $\mathbf{p}_k$. Vztah odpovídající (2.8) pro přesné výpočty je dokázán větou 4 v části 1.6.

Díky tomuto vztahu můžeme vyjádřit dolní odhad $\|\mathbf{e}_k\|_{\mathbf{A}}^2$ ve tvaru

$$\sum_{l=k}^{k+r-1} \gamma_l \|\mathbf{r}_l\|^2.$$

Půjde o dobrý odhad, pokud $\|\mathbf{e}_{k+r}\|_{\mathbf{A}}^2 \ll \|\mathbf{e}_k\|_{\mathbf{A}}^2$. Tato hranice je málo ovlivněna zaokrouhlovacími chybami pro $\|\mathbf{e}_k\|_{\mathbf{A}} / \|\mathbf{e}_0\|_{\mathbf{A}} > \varepsilon$, viz. [13].

Pokud tedy v kroku k chceme určit odhad $\mathbf{A}$-normy chyby, musíme udělat r iterací algoritmu navíc. Počet kroků r potřebných k získání dobrého odhadu závisí na tom, jak moc $\mathbf{A}$-norma chyby kolem kroku k klesá. Volba hodnoty r tedy zůstává otázkou.

2.5.3 Strakošovy matice

Nyní se budeme věnovat určitým typům matic, na kterých lze dobře demonstrovat možný vliv konečné aritmetiky na chování CG. Tyto matice jsou závislé na parametru ζ, který určuje rozložení vlastních čísel v matici. Zajímavé je, že malou změnou tohoto parametru získáme velké změny v zaokrouhlovacích chybách algoritmu CG v konečné aritmetice.

V následujících pokusech bude matice $\mathbf{A}$ sestavená takto:

$$A_{ii} = \lambda_i = \lambda_1 + \frac{i-1}{N-1}(\lambda_N - \lambda_1)\zeta^{N-1}, \quad i = 2, 3, \ldots, N-1. \tag{2.9}$$

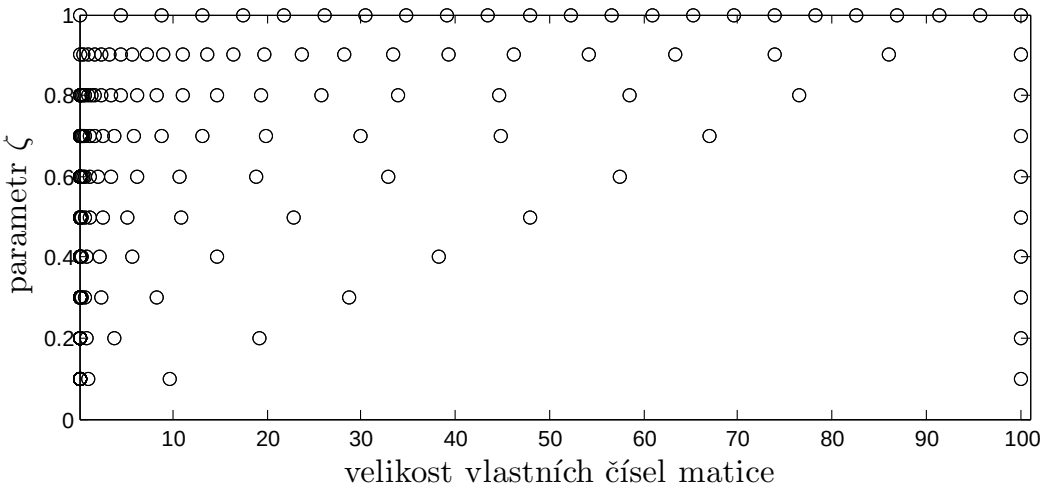
Nejprve se musí zvolit parametry N, λ_1, λ_N a ζ, kde N určuje velikost matice, λ_1 a λ_N určují nejmenší a největší vlastní číslo matice a parametr ζ reprezentuje rozložení vlastních čísel v požadované matici. Z těchto vstupních hodnot a předpisu (2.9) lze získat celou matici $\mathbf{A}$.

Parametr ζ se volí z intervalu $(0, 1\rangle$. Platí, že pro $\zeta = 1$ je rozložení vlastních čísel uniformní, tedy vlastní čísla jsou rovnoměrně rozložena v intervalu $\langle \lambda_1, \lambda_N \rangle$. S klesajícím ζ se tato vlastnost ztrácí a vlastní čísla se přesouvají směrem k λ_1, jak ukazuje obr. 2.4. Vstupní

parametry tohoto grafu jsou dány rovnicemi: $N = 24$,

$$\lambda_1 = 0.1, \quad \lambda_N = 100 \tag{2.10}$$

a je zde vykresleno rozložení vlastních čísel pro různé hodnoty parametru ζ.



Obrázek 2.4: Rozložení vlastních čísel matic sestavených pomocí (2.9) pro různé hodnoty paramteru ζ. Inspirováno článkem [6].

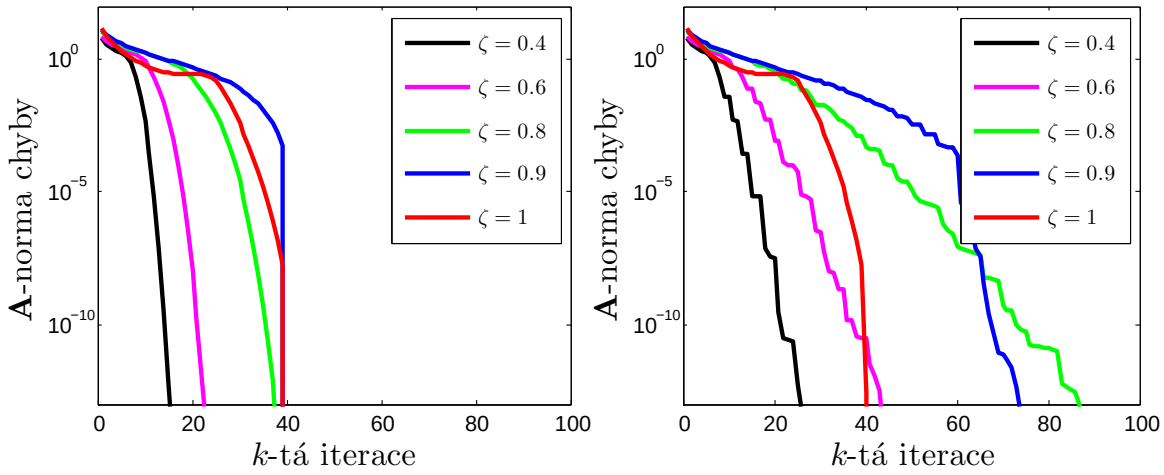
2.5.4 Vliv rozložení vlastních čísel

Na obr. 2.5 je naznačeno, jak rozložení vlastních čísel ovlivňuje konvergenci CG. Matice $\mathbf{A}$, pro kterou je výpočet realizován, je sestrojena pomocí (2.9, 2.10) a $N = 40$. Vstupní vektory pro algoritmus CG jsou dány (2.1). Je zde vykreslena $\mathbf{A}$-norma chyby v závislosti na počtu iterací pro různé hodnoty ζ. Obrázek vlevo je vytvořen pomocí přesného algoritmu CG a vpravo pomocí FP algoritmu.

Je patrné, že zatímco pro některé hodnoty ζ se odpovídající křivky velmi liší, pro $\zeta = 1$ je křivka vpravo a vlevo obrázku 2.5 stejná. To znamená, že parametr ζ ovlivňuje výskyt zaokrouhlovacích chyb ve výpočtech algoritmu CG.

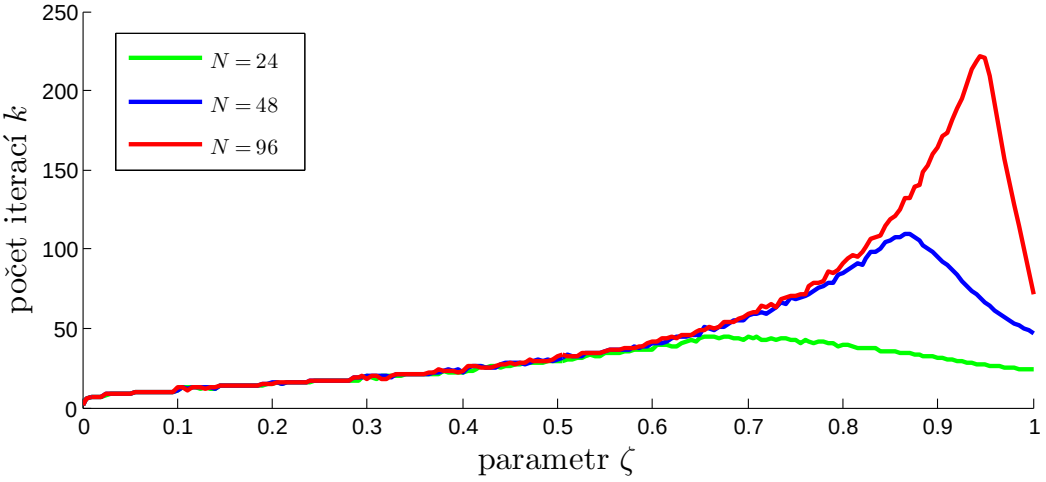
V článku [20] bylo ukázáno, že existuje kritická hodnota ζ (označená ζ^*), ve které jsou zaokrouhlovací chyby maximální. Jev patrný na obr. 2.5 ukazuje z trochu jiného úhlu pohledu i obr. 2.6, kde je v závislosti na parametru ζ vykreslen počet iterací algoritmu CG, kterých bylo potřeba k dosáhnutí určité blízkosti aproximace řešení a skutečného řešení. Ta je dána zastavovací podmínkou

$$\frac{\|\mathbf{e}_k\|_{\mathbf{A}}}{\|\mathbf{e}_0\|_{\mathbf{A}}} < 10^{-12}. \tag{2.11}$$



Obrázek 2.5: **A**-norma chyby $\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}$ vypočtená přesným algoritmem CG (vlevo) a FP algoritmem (vpravo) v závislosti na kroku k a pro různé rozložení vlastních čísel ζ. Inspirováno článkem [6].

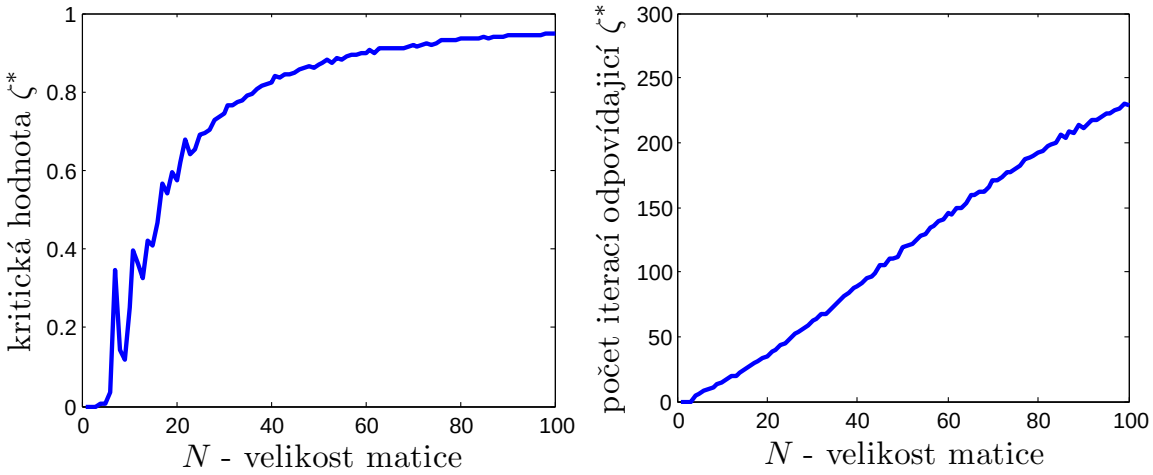
Výpočty algoritmu byly provedeny v konečné aritmetice se vstupními hodnotami (2.5) a matice $\mathbf{A}$ byla získaná z (2.9) pomocí (2.10) a $N = 24, 48, 96$.



Obrázek 2.6: Počet iterací k algoritmu CG potřebných k získání zastavovací podmínky (2.11) v závislosti na ζ a pro $N = 24, 48, 96$. Inspirováno článkem [20].

Nyní již je patrné, že malá změna ζ v okolí jeho kritické hodnoty vyvolá velkou změnu v zaokrouhlovacích chybách algoritmu CG, jak bylo napsáno v začátku podkapitoly 2.5.3.

K dalšímu zkoumání tohoto jevu poslouží dva grafy v obr. 2.7. Oba grafy byly sestrojeny pomocí vstupních hodnot použitých již v předchozím případě. Tentokrát zde ale bylo počítáno s $N \in \langle 0, 100 \rangle$. Graf vlevo ukazuje, jak se vyvíjí kritická hodnota ζ^* v závislosti na počtu vlastních čísel N a druhý (vpravo) ukazuje, kolik iterací je potřeba k dosažení podmínky



Obrázek 2.7: Kritická hodnota ζ^* v závislosti na N (vlevo) a počet iterací potřebných k dosažení podmínky (2.11) pro kritické ζ^* odpovídající hodnotě N.

(2.11) pro kritické ζ^* dané počtem vlastních čísel N.

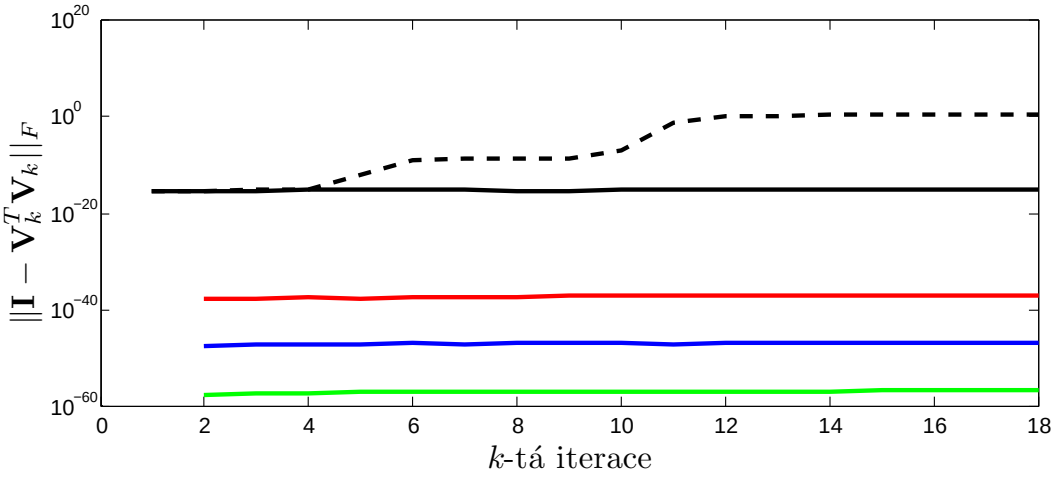
Je patrné, že ζ^* globálně roste, ale může i lokálně klesat. Stejně tak i počet iterací, ale lokální pokles je zde méně znatelný.

2.6 Použití násobné aritmetiky

V sekci 2.1 bylo popsáno, jak lze v konečné aritmetice až na malou nepřesnost simulovat přesné výpočty. Tomuto způsobu simulace budeme říkat dvojitá reortogonalizace. Z obr. 2.1 je patrné, že ztráta ortogonality $\|\mathbf{I} - \mathbf{V}_k^T \mathbf{V}_k\|_F$ vypočtená pomocí dvojité reortogonalizace je řádu 10^{-15}.

Přesnější výpočty můžeme získat pomocí násobné aritmetiky (Matlab: Symbolic Math Toolbox), kde můžeme zvolit, na kolik platných desetinných míst bude program zaokrouhlovat. Samotná implementace algoritmu A1 pomocí násobné aritmetiky ale nedává dobré výsledky ani pro velký počet platných desetinných míst. Pokud tedy chceme zpřesnit výpočet, musíme v násobné aritmetice aplikovat algoritmus, který v každém kroku dvakrát reortogonalizuje reziduový vektor proti předchozím reziduím. Výsledky vypočtené tímto způsobem budeme označovat jako výsledky vypočtené násobnou aritmetikou.

Obr. 2.8 slouží k demonstrování těchto simulací. Pro vstupní hodnoty stejné jako v případě obr. 2.1 je zde vykreslena ztráta ortogonality v závislosti na iteraci k. Černě přerušovaně jsou zobrazeny hodnoty vypočtené FP algoritmem. Simulace pomocí dvojité reortogonalizace je zobrazena černě a zbylé křivky jsou vypočteny násobnou aritmetikou. V průběhu výpočtu program zaokrouhloval na 30, 40 nebo 50 platných desetinných míst (zobrazeno červeně, modře nebo zeleně).



Obrázek 2.8: Ztráta ortogonality vypočtená pomocí FP algoritmu (černě přerušovaně), pomocí dvojité reortogonalizace (černě) a pomocí násobné aritmetiky s 30 platnými desetinnými čísly (červeně), 40 platnými desetinnými čísly (modře) a 50 platnými desetinnými čísly (zeleně).

2.7 Různé varianty CG

2.7.1 Zaokrouhlovací chyby při výpočtu rozdílu reziduí

Lokálními zaokrouhlovacími chybami dvoučlenných rekurencí a tříčlenných rekurencí se zabývali např. Martin Gutknecht a Zdeněk Strakoš ve své práci [7], kteří se zde často odkazovali na práci Anne Greenbaum [5], kde analyzovala lokální chyby dvoučlenných rekurencí. V článku [7] lze tedy najít odvození lokálních zaokrouhlovacích chyb pro tříčlenné rekurence a v článku [5] odvození téhož pro dvoučlenné rekurence. Oba články používají k této analýze rozdíl mezi skutečnými a rekurzivními rezidui.

Výsledky jednotlivých analýz pro dvoučlenné a tříčlenné rekurence se velmi liší. Rozdíl mezi rezidui spočtenými dvoučlennými rekurencemi je dán:

$$\mathbf{f}_{k+1} = \mathbf{f}_0 - \sum_{j=0}^{k} \mathbf{l}_j, \quad \mathbf{l}_j := \mathbf{A}\mathbf{m}_j + \mathbf{n}_j, \tag{2.12}$$

kde $\mathbf{m}_j$ představuje lokální zaokrouhlovací chybu vzniklou implementací A1:(6) v konečné aritmetice a $\mathbf{n}_j$ je chyba vzniklá implementací A1:(7), tzn.

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \gamma_k \mathbf{p}_k + \mathbf{m}_k,$$

$$\mathbf{r}_{k+1} := \mathbf{r}_k - \gamma_k \mathbf{A}\mathbf{p}_k + \mathbf{n}_k.$$

Tento rozdíl $\mathbf{f}_{k+1}$ je ve skutečnosti pouze součet lokálních chyb.

Oproti tomu rozdíl mezi rezidui spočtenými tříčlennými rekurencemi je složitější. Pro jeho

odvození v [7] autoři předpokládali, že tříčlenné rekurence objevující se v algoritmu vypadají v konečné aritmetice následovně

$$\mathbf{r}_{k+1} = (\mathbf{A}\mathbf{r}_k - \mathbf{r}_k\alpha_k - \mathbf{r}_{k-1}\beta_{k-1} + \mathbf{g}_k)/\varphi_k,$$
$$\mathbf{x}_{k+1} = -(\mathbf{r}_k + \mathbf{x}_k\alpha_k + \mathbf{x}_{k-1}\beta_{k-1} - \mathbf{h}_k)/\varphi_k.$$

Koeficient φ_k je záměrně vybrán tak, aby $\varphi_k = -(\alpha_k + \beta_{k-1})$. Výsledkem jejich analýzy je tedy tvar rozdílu $\mathbf{f}_{k+1}$ mezi skutečnými a rekurzivními rezidui pro algoritmy založené na tříčlenných rekurencích:

$$\begin{aligned}\mathbf{f}_{k+1} = \mathbf{f}_0 &- \mathbf{l}_0 \\ &- \mathbf{l}_0\frac{\beta_0}{\alpha_1} - \mathbf{l}_1 \\ &- \mathbf{l}_0\frac{\beta_0\beta_1}{\alpha_1\alpha_2} - \mathbf{l}_1\frac{\beta_1}{\alpha_2} - \mathbf{l}_2 \\ &\vdots \\ &- \mathbf{l}_0\frac{\beta_0\beta_1\cdots\beta_{k-1}}{\alpha_1\alpha_2\cdots\alpha_k} - \ldots - \mathbf{l}_{k-1}\frac{\beta_{k-1}}{\alpha_k} - \mathbf{l}_k,\end{aligned}$$

kde $\mathbf{l}_k = (-\mathbf{b}\varepsilon_k + \mathbf{A}\mathbf{h}_k + \mathbf{g}_k)\frac{1}{\varphi_k}$ a ε_k je zaokrouhlovací chyba vzniklá počítáním koeficientu φ_k v konečné aritmetice, tj. $\varphi_k = -(\alpha_k + \beta_{k-1}) + \varepsilon_k$.

Na první pohled je patrné, že tvar této mezery je mnohem složitější. Velikost $\mathbf{f}_{k+1}$ záleží na více proměnných než u (2.12), a proto je možné, že bude mnohem větší. Jak bylo napsáno v části 2.2, větší rozdíl může znamenat, že je algoritmus více poznamenán zaokrouhlovacími chybami a maximální dosažitelná přesnost řešení je horší. Tento jev je patrný na obr. 2.9.

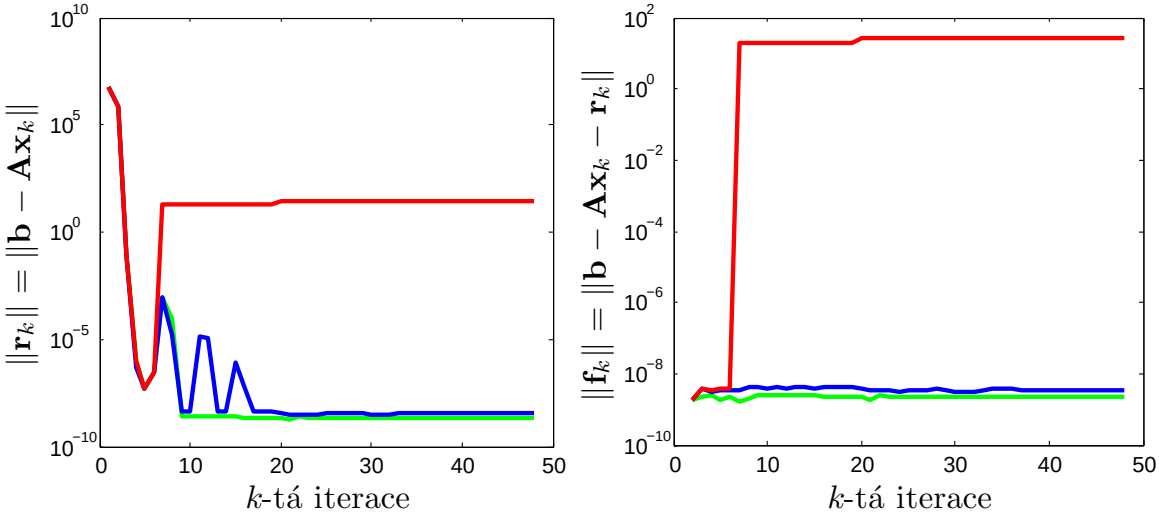
Oba grafy obr. 2.9 jsou vytvořeny pomocí FP algoritmů A1, A3 a A4. Je zde tedy porovnána Hestenesova-Stiefelova, Rutishauserova a Hagemanova-Youngova verze metody sdružených gradientů. Vstupní parametry byly nastaveny následujícím způsobem:

$$\mathbf{A} = \mathbf{V}\mathbf{T}\mathbf{V}^T, \quad \mathbf{x}_0 = [0, 0, \ldots, 0]^T, \quad \mathbf{b} = \mathbf{A}\mathbf{x}_e,$$

kde $\mathbf{x}_e = [1, 1, \ldots, 1]^T$, $\mathbf{V}$ je unitární matice získaná pomocí QR rozkladu matice náhodné a $\mathbf{T}$ je tridiagonální matice s koeficienty

$$T_{11} = 10^{-7}, \quad T_{NN} = 10^7,$$
$$T_{ii} = 10^5, \quad T_{i-1,i} = T_{i,i-1} = 10^{-2}, \quad i = 2, 3, \ldots N-1.$$

První z grafů vykresluje euklidovskou normu skutečného rezidua $\|\mathbf{r}_k^{(t)}\|$ danou (2.2) v závislosti na iteraci k a druhý euklidovskou normu rozdílu (2.3) také v závislosti na počtu iterací k.



Obrázek 2.9: Norma skutečného rezidua $\|\mathbf{r}_k^{(t)}\|$ (vlevo) a norma rozdílu reziduí $\|\mathbf{f}_k\|$ (vpravo). Oba grafy byly vypočteny třemi různými algoritmy: Hestenesovým-Stiefelovým algoritmem (zeleně), Rutishauserovým algoritmem (modře) a Hagemanovým-Youngovým algoritmem (červeně). Inspirováno článkem [7].

Z obrázku je patrné, že algoritmy CG založené na dvoučlenných rekurencích (tedy HS a R verze) mají oproti algoritmu založeném na tříčlenných rekurencích (HY verze) mnohem menší normu rozdílu $\|\mathbf{f}_k\|$ i normu skutečných reziduí $\|\mathbf{r}_k^{(t)}\|$. To tedy znamená, že tyto varianty jsou méně poznamenány zaokrouhlovacími chybami a také, že jejich hladiny limitní přesnosti jsou menší.

2.7.2 Vliv zaokrouhlovacích chyb při analýze zachování ortogonality

Další jev, díky kterému lze porovnávat různé varianty metody sdružených gradientů, je zachování ortogonality. Pro HS variantu byla totiž v [23] dokázána vlastnost

$$|\mathbf{p}_j^T \mathbf{r}_{j+1}| \leq \varepsilon \|\mathbf{r}_j\|^2 \sqrt{\kappa(\mathbf{A})} O(jn + \frac{j^2}{2}) + O(\varepsilon^2), \quad j = 1, 2, \ldots, N-1, \tag{2.13}$$

kde ε je strojová přesnost a $\kappa(\mathbf{A})$ je číslo podmíněnosti matice $\mathbf{A}$.

Obr. 2.10 ale ukazuje, že tato vlastnost pro ostatní verze CG neplatí. Graf je vytvořen opět pomocí FP algoritmů A1, A3 a A4. Tentokrát se vstupními parametry

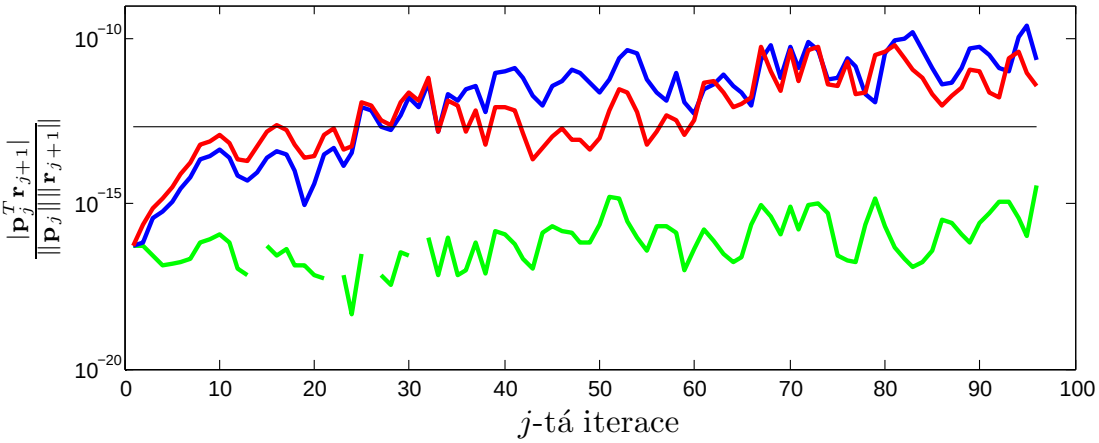
$$\mathbf{x}_0 = [0, 0, \ldots, 0]^T, \quad \mathbf{b} = \frac{\mathbf{b}}{\|\mathbf{b}\|}, \quad \mathbf{b} = [1, 1, \ldots, 1]^T \tag{2.14}$$

a matice $\mathbf{A}$ (bcsstk01.mtx) je na přiloženém CD a v [11]. V grafu je vykreslena lokální ortogonalita mezi normalizovanými vektory $\mathbf{p}_j$ a $\mathbf{r}_{j+1}$, kde pro algoritmy R a HY je vektor $\mathbf{p}_j$ nahrazen vektorem $\mathbf{x}_{j+1} - \mathbf{x}_j$. Platí totiž, že normalizované vektory odpovídající těmto

vektorům jsou si rovny, viz. A1:(6). V grafu jsou tedy vykresleny hodnoty výrazu

$$\frac{|\mathbf{p}_j^T \mathbf{r}_{j+1}|}{\|\mathbf{p}_j\|\|\mathbf{r}_{j+1}\|}, \quad \text{resp.} \quad \frac{|(\mathbf{x}_{j+1} - \mathbf{x}_j)^T \mathbf{r}_{j+1}|}{\|\mathbf{x}_{j+1} - \mathbf{x}_j\|\|\mathbf{r}_{j+1}\|} \tag{2.15}$$

v závislosti na hodnotě iterace j.



Obrázek 2.10: Lokální ortogonalita (2.15) vypočtená Hestenesovým-Stiefelovým algoritmem (zeleně), Rutishauserovým algoritmem (modře) a Hagemanovým-Youngovým algoritmem (červeně).

V grafu je kromě samotných skalárních součinů (2.15) vykreslena i hranice

$$\varepsilon\sqrt{\kappa(\mathbf{A})},$$

která je odvozená z hranice (2.13). Je patrné, že ani jedna z křivek odpovídajících verzi R nebo HY neleží celá pod touto hranicí a také, že jejich lokální ortogonalita se ztrácí velmi rychle.

Křivka odpovídající HS algoritmu není souvislá. Platí, že v místech přerušení se hodnota (2.15) rovná nule. V přesné aritmetice by se vždy tato hodnota rovnala nule. Důkaz se opírá o vyjádření parametru γ_j ve tvaru (1.4), tedy

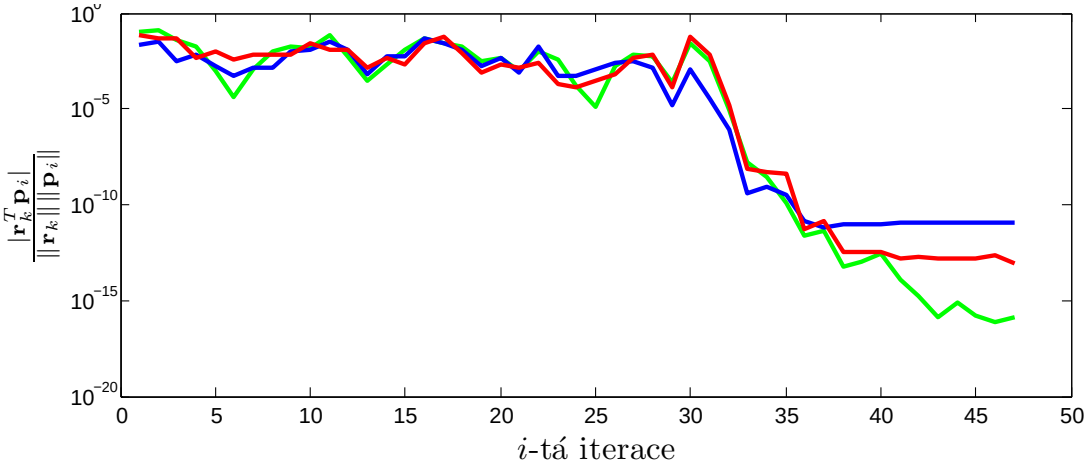
$$\mathbf{p}_j^T \mathbf{r}_{j+1} = \mathbf{p}_j^T(\mathbf{r}_j - \gamma_j \mathbf{A}\mathbf{p}_j) = \mathbf{p}_j^T \mathbf{r}_j - \frac{\mathbf{p}_j^T \mathbf{r}_j}{\mathbf{p}_j^T \mathbf{A}\mathbf{p}_j} \mathbf{p}_j^T \mathbf{A}\mathbf{p}_j = 0.$$

Obr. 2.11 se opět zabývá ztrátou ortogonality mezi reziduovými a směrovými vektory. Tentokrát ale vývojem ztráty ortogonality, tedy počítá se výraz $|\mathbf{r}_k^T \mathbf{p}_i|$, $i = 1, 2, \ldots, k-1$ pro předem daný počet kroků k a normalizované vektory $\mathbf{r}_k$ a $\mathbf{p}_i$. Vstupní hodnoty jsou zvoleny

stejně jako v předchozím případě. Vykresleny jsou tedy hodnoty výrazů

$$\frac{|\mathbf{r}_k^T \mathbf{p}_i|}{\|\mathbf{r}_k\|\|\mathbf{p}_i\|}, \quad \text{resp.} \quad \frac{|\mathbf{r}_k^T (\mathbf{x}_{i+1} - \mathbf{x}_i)|}{\|\mathbf{r}_k\|\|\mathbf{x}_{i+1} - \mathbf{x}_i\|} \tag{2.16}$$

pro $k = N$, kde N je velikost matice $\mathbf{A}$, v závislosti na parametru i.



Obrázek 2.11: Vývoj ztráty ortogonality (2.16) vypočtený Hestenesovým-Stiefelovým algoritmem (zeleně), Rutishauserovým algoritmem (modře) a Hagemanovým-Youngovým algoritmem (červeně).

Zachování ortogonality vektorů $\mathbf{r}_k$ a $\mathbf{p}_i$ je pro algoritmy důležité, protože pokud tato vlastnost je zachována, metoda sdružených gradientů konverguje rychleji. Výraz $\mathbf{r}_k^T \mathbf{p}_i$ totiž měří kvalitu zachování $\mathbf{A}$-ortogonality chyby $\mathbf{e}_k$ k předchozím směrovým vektorům (tj. ke korespondujícímu Krylovovu podprostoru):

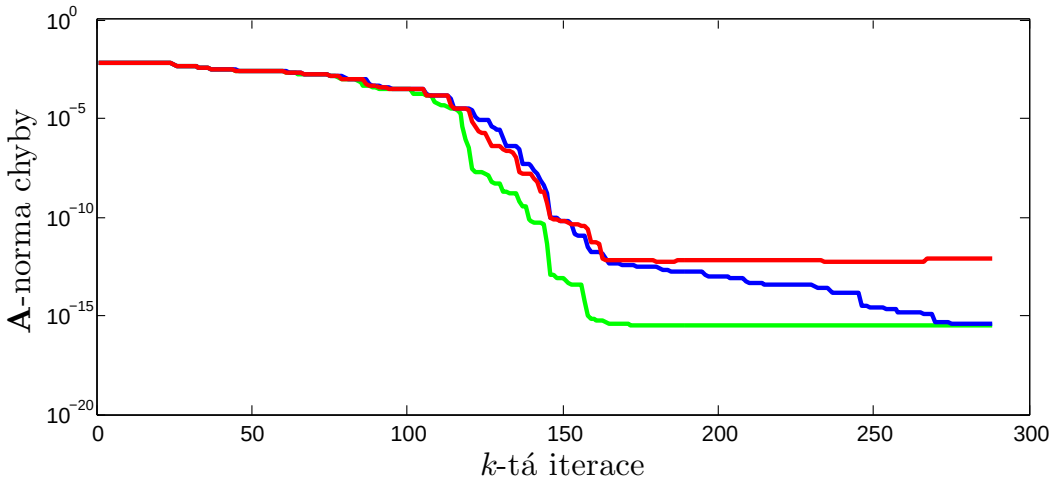
$$\mathbf{r}_k^T \mathbf{p}_i = (\mathbf{b} - \mathbf{A}\mathbf{x}_k)^T \mathbf{p}_i = (\mathbf{x} - \mathbf{x}_k)^T \mathbf{A}\mathbf{p}_i.$$

A pokud je tato vlastnost ortogonality zachována, znamená to, že $\mathbf{A}$-norma chyby je minimální, viz. sekce 1.2.

Další obrázek (obr. 2.12) byl vykreslen opět se stejnými vstupními hodnotami, tedy matice $\mathbf{A}$ je dána [11] a vektory jsou dány (2.14). Ukazuje $\mathbf{A}$-normu chyby v závislosti na kroku k pro všechny tři algoritmy.

Lze zde pozorovat, že aproximace řešení získané HS algoritmem konvergují rychleji než ty získané ostatními algoritmy, což odpovídá tomu, že algoritmus lépe zachovává podmínku $\mathbf{A}$-ortogonality chyby k prostoru danému směrovými vektory $\mathbf{p}_i$, tj. ke Krylovovým podprostorům (1.8). Tento jev tedy potvrzuje výsledek našeho předchozího experimentu.

Na obrázku je patrný i jev popsaný v podkapitole 2.2, tzn. $\mathbf{A}$-norma chyby neklesne pod určitou hladinu přesnosti, která je pro R a HS algoritmus mnohem menší než pro HY algoritmus.



Obrázek 2.12: **A**-norma chyby $\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}$ vypočtená Hestenesovým-Stiefelovým algoritmem (zeleně), Rutishauserovým algoritmem (modře) a Hagemanovým-Youngovým algoritmem (červeně).

2.7.3 Vliv na skalární součin směrových vektorů

Náš další experiment se zabývá zachováním vlastnosti

$$\mathbf{p}_i^T \mathbf{p}_j \geq 0, \quad i, j = 0, 1, \ldots, N \tag{2.17}$$

při výpočtech v konečné aritmetice. Tato vlastnost je pro přesné výpočty dokázána v lemmatu 1 uvedeném v části 1.6. Nejprve se budeme věnovat případu se dvěma po sobě jdoucími směrovými vektory, tj.

$$\mathbf{p}_j^T \mathbf{p}_{j+1} \geq 0, \quad j = 0, 1, \ldots, N-1. \tag{2.18}$$

Myšlenka důkazu zachování (2.18) v konečné aritmetice by mohla být následující. Rovnice A1:(9) počítaná v konečné aritmetice je dána předpisem

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \delta_{j+1}\mathbf{p}_j + \mathbf{q}_{j+1},$$

kde $\mathbf{q}_{j+1}$ je vektor obsahující zaokrouhlovací chyby. Přenásobením této rovnice zleva směrovým vektorem $\mathbf{p}_j^T$ získáme

$$\mathbf{p}_j^T \mathbf{p}_{j+1} = \mathbf{p}_j^T \mathbf{r}_{j+1} + \delta_{j+1}\mathbf{p}_j^T \mathbf{p}_j + \mathbf{p}_j^T \mathbf{q}_{j+1}.$$

Výrazy $\mathbf{p}_j^T \mathbf{r}_{j+1}$ a $\mathbf{p}_j^T \mathbf{q}_{j+1}$ jsou velmi malé (úměrné strojové přesnosti). Vektor $\mathbf{q}_{j+1}$ totiž odpovídá zaokrouhlovací chybě a absolutní hodnota $\mathbf{p}_j^T \mathbf{r}_{j+1}$ je shora omezená pravou stranou

nerovnice (2.13). Zanedbáme-li pro jednoduchost obě čísla úměrné ε, dostaneme

$$\delta_{j+1}\mathbf{p}_j^T\mathbf{p}_j = \frac{\mathbf{r}_{j+1}^T\mathbf{r}_{j+1}}{\mathbf{r}_j^T\mathbf{r}_j}\mathbf{p}_j^T\mathbf{p}_j > 0.$$

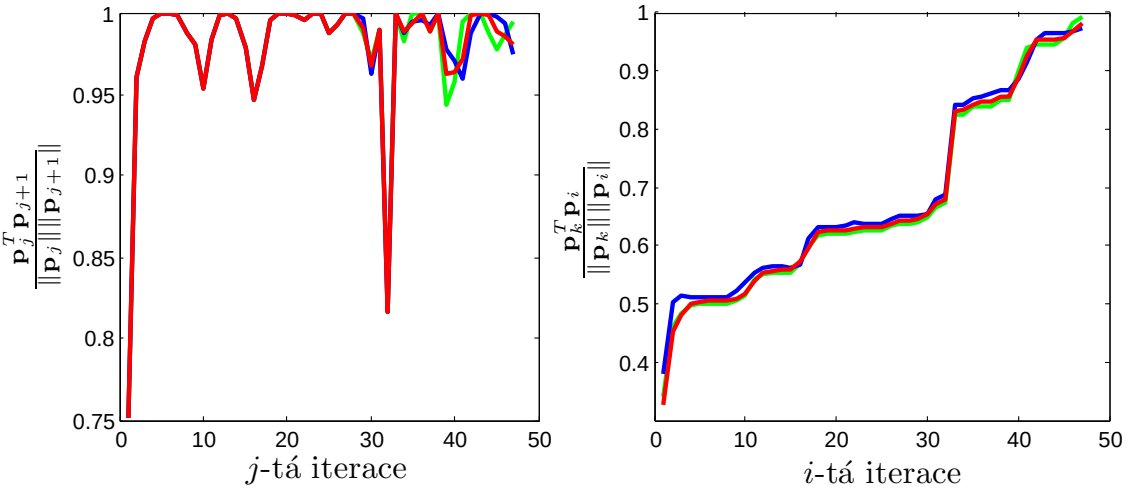
Obrázek 2.13 byl vypočten pro vstupní hodnoty opět danými [11] a (2.14). Vlevo je graf vykreslující hodnotu skalárního součinu po sobě jdoucích normalizovaných směrových vektorů. V případě R a HY algoritmu se normalizované směrové vektory určují stejně jako v předchozím případě. Počítáme tedy

$$\frac{\mathbf{p}_j^T\mathbf{p}_{j+1}}{\|\mathbf{p}_j^T\|\|\mathbf{p}_{j+1}\|} \quad \text{resp.} \quad \frac{(\mathbf{x}_{j+1}-\mathbf{x}_j)^T(\mathbf{x}_{j+2}-\mathbf{x}_{j+1})}{\|\mathbf{x}_{j+1}-\mathbf{x}_j\|\|\mathbf{x}_{j+2}-\mathbf{x}_{j+1}\|}, \quad j = 1, 2, \ldots, N-1. \tag{2.19}$$

Vpravo je vykreslen skalární součin normalizovaného k-tého a i-tého směrového vektoru (analogicky k obrázku 2.11), tedy

$$\frac{\mathbf{p}_k^T\mathbf{p}_i}{\|\mathbf{p}_k\|\|\mathbf{p}_i\|}, \quad \text{resp.} \quad \frac{(\mathbf{x}_{k+1}-\mathbf{x}_k)^T(\mathbf{x}_{i+1}-\mathbf{x}_i)}{\|\mathbf{x}_{k+1}-\mathbf{x}_k\|\|\mathbf{x}_{i+1}-\mathbf{x}_i\|}, \quad i = 1, 2, \ldots, k-1, \tag{2.20}$$

kde k je určeno jako velikost matice $\mathbf{A}$.



Obrázek 2.13: Skalární součin daný vztahem (2.19) (vlevo) v závislosti na iteraci j a skalární součin daný (2.20) (vpravo) v závislosti na iteraci i pro $k = N$. Křivky byly vypočteny Hestenesovým-Stiefelovým algoritmem (zeleně), Rutishauserovým algoritmem (modře) a Hagemanovým-Youngovým algoritmem (červeně).

Důkaz zachování vlastnosti $\mathbf{p}_k^T\mathbf{p}_i \geq 0$ v konečné aritmetice vyžaduje řádnou analýzu zaokrouhlovacích chyb, která je nad rámec této práce. Přesto obrázek naznačuje, že vlastnost (2.17) může být pro všechny tři verze CG zachována.

Kapitola 3

Simulace výpočtů CG v konečné aritmetice

3.1 Úvod do problému

V sekci 2.4 bylo naznačeno, proč lze na FP výpočty Lanczosova algoritmu hledět jako na výpočty přesného algoritmu aplikovaného na větší matici $\hat{\mathbf{A}}$. Touto myšlenkou se podrobně zabývala Anne Greenbaum ve své práci [4].

Jedním z cílů její práce bylo dokázat, že výpočty pomocí jakýchkoliv trochu perturbovaných rekurencí Lanczosova algoritmu jsou ekvivalentní (až na malou nepřesnost) přesným výpočtům s vhodnou maticí $\hat{\mathbf{A}}$ a startovacím vektorem $\hat{\mathbf{v}}$. Matice $\hat{\mathbf{A}}$ je určena tak, aby všechna její vlastní čísla ležela v malých intervalech okolo vlastních čísel původní matice, tzv. *shlucích* vlastních čísel, a vektor $\hat{\mathbf{v}}$ je zvolen tak, aby jeho první komponenta byla rovna normě původního vektoru $\mathbf{v}$ a ostatní komponenty byly nulové.

K tomu bylo potřeba dokázat, že pro každou iteraci k se matice $\mathbf{T}_k$ generovaná Lanczosovým FP algoritmem aplikovaným na $\mathbf{A}$ rovná matici, která je generovaná přesnými rekurencemi aplikovanými na $\hat{\mathbf{A}}$. Nejprve bylo nutno ukázat, že každá $\mathbf{T}_k$ může být rozšířena na matici $\mathbf{T}_{k+m}$, jejíž vlastní čísla jsou rozprostřena okolo vlastních čísel $\mathbf{A}$.

Z toho důvodu navrhla metodu, jak z matice $\mathbf{T}_k$ získat $\mathbf{T}_{k+m}$. Tato metoda spočívá v rozšíření $\mathbf{T}_k$ o koeficienty vypočtené Lanczosovým algoritmem, který bude nové vektory reortogonalizovat navzájem a také vůči nezkonvergovaným Ritzovým vektorům. Dokázala, že pro tuto matici do kroku $N + m$, kde N je velikost matice $\mathbf{A}$ a m je počet zkonvergovaných vektorů, bude β rovna nule. Za předpokladu, že tato skutečnost nastane v nejhorším případě, tedy v kroku $N + m$, bude β_{N+m+1} rovna nule a matice $\mathbf{T}_{k+m}$ bude odpovídat matici $\hat{\mathbf{A}}$.

Postup sestrojení matice $\mathbf{T}_{k+m}$ a tedy i postup sestrojení systému, jehož přesné výpočty odpovídají perturbovaným výpočtům původního systému, byl v [4] popsán pečlivě. V dalším budeme o přesných výpočtech tohoto systému uvažovat jako o simulaci FP výpočtů.

3.2 Sestrojení simulační soustavy rovnic

Příklad simulace FP výpočtů, který vychází z postupu uvedeného v [4] a který byl použit pro experimenty zde uvedené, je následující.

Nejprve se pomocí porušených Lanczosových rekurencí (FP algoritmus) aplikovaných na matici $\mathbf{A}$ a vektor $\mathbf{v}$ vytvoří matice $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ a $\mathbf{T}_k$, viz. (1.10). Z těchto matic a také z hodnoty β_{k+1} se zjistí, jaké vektory nejsou zkonvergované následujícím způsobem.

- Vypočte se spektrální rozklad matice $\mathbf{T}_k$:

  $$\mathbf{T}_k = \mathbf{S}^T \mathbf{\Theta} \mathbf{S},$$

  kde matice $\mathbf{\Theta}$ má na své diagonále uložena Ritzova čísla a matice $\mathbf{S}$ má ve svých sloupcích uloženy ortogonální vlastní vektory matice $\mathbf{T}_k$.

- Provede se taková normalizace, aby poslední komponenty vlastních vektorů byly kladné a pomocí této upravené matice $\mathbf{S}$ se vypočítají Ritzovy vektory

  $$\mathbf{Z} = \mathbf{V}_k \mathbf{S}.$$

- Následuje posouzení, zda Ritzova čísla jsou dobře oddělená nebo zda se nachází v shluku. Platí, že Ritzovo číslo θ_i je součástí shluku, pokud

  $$\min_{i \neq j} \frac{|\theta_i - \theta_j|}{\|\mathbf{A}\|} \leq \mu,$$

  kde μ je vhodná tolerance. Vlastní čísla, která tuto vlastnost nesplňují, jsou označena jako dobře oddělená.

- Ritzovy vektory $\mathbf{z}_i$ odpovídající dobře oddělenému Ritzovu číslu se nazývají zkonvergované, pokud

  $$\beta_{k+1} s_k^i \leq \vartheta,$$

  kde s_k^i je i-tá komponenta k-tého vlastního vektoru matice $\mathbf{T}_k$ a ϑ je opět vhodná tolerance, ovšem obecně jiná než μ. Ostatní vektory odpovídající odděleným Ritzovým číslům jsou označeny jako nezkonvergované.

- Pro vlastní čísla, která jsou součástí jednoho shluku, se určí tzv. *vektor shluku*, který je lineární kombinací všech Ritzových vektorů odpovídajících těmto Ritzovým číslům, tedy

  $$\tilde{\mathbf{z}} = \frac{1}{\upsilon} \sum_l s_k^l \mathbf{z}_l, \quad \upsilon = \sqrt{\sum_l \left(s_k^l\right)^2},$$

kde l je koeficient Ritzových čísel patřících do shluku. Zda je vektor shluku zkonvergovaný, se kontroluje pomocí podmínky

$$\beta_{k+1}\upsilon \leq \vartheta.$$

- Hodnoty parametrů μ a ϑ mohou být zvoleny různě. Zde jsou určeny pomocí strojové přesnosti a v případě ϑ také pomocí normy matice $\mathbf{A}$:

$$\mu := \sqrt{\varepsilon}, \quad \vartheta := \sqrt{\varepsilon}\|\mathbf{A}\|.$$

Od k-tého kroku spustíme pozměněný Lanczosův algoritmus, tj. aplikujeme pozměněný algoritmus na matici $\mathbf{A}$ a vektor $\mathbf{v} = \mathbf{v}_k$. Tento algoritmus v každé iteraci reortogonalizuje nově vzniklý vektor proti všem vektorům již spočteným tímto algoritmem a také proti nezkonvergovaným vektorům $\mathbf{z}$ a $\tilde{\mathbf{z}}$. Koeficienty β_{k+j} a α_{k+j} vznikající během toho výpočtu se postupně ukládají do matice

$$\mathbf{T}_{k+j} = \begin{bmatrix} \mathbf{T}_k & \beta_{k+1} & & & \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} & & \\ & \beta_{k+2} & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{k+j} \\ & & & \beta_{k+j} & \alpha_{k+j} \end{bmatrix}$$

od indexu k. Pokud $\beta_{k+j+1} = 0$, tato matice bude odpovídat matici $\hat{\mathbf{A}}$ uvažované v předchozí podkapitole. V [4] je dokázáno, že vlastní čísla $\mathbf{T}_{k+j}$ jsou všechna umístěna ve shlucích okolo vlastních čísel původní matice $\mathbf{A}$. Zde je tedy ukázáno, jak matici $\mathbf{T}_k$ rozšířit na matici $\hat{\mathbf{A}}$.

V [4] je také dokázáno, že výpočty Lanczosovým algoritmem aplikovaným na matici $\mathbf{A}$ a vektor $\mathbf{v}$ v konečné aritmetice je možné simulovat pomocí přesných výpočtů aplikovaných na matici $\hat{\mathbf{A}} = \mathbf{T}_{k+j}$ (za předpokladu $\beta_{k+j+1} = 0$) a startovací vektor $\hat{\mathbf{v}} = \|\mathbf{v}\|\mathbf{e}_{(1)}$.

Podobná vlastnost lze dokázat i pro výpočty provedené algoritmem metody sdružených gradientů. Pomocí FP algoritmu a rovnic (1.19) a (1.20) se sestrojí matice $\mathbf{T}_k$, najdou se nezkonvergované vektory a pak se pokračuje v sestavování matice $\mathbf{T}_{k+j}$ pozměněným algoritmem (reortogonalizace reziduí proti nezkonvergovaným Ritzovým vektorům a vzájemná reortogonalizace), dokud β_{k+j+1} nebude rovno nule.

Platí, že výpočty FP algoritmu aplikovaného na soustavu $\mathbf{Ax} = \mathbf{b}$ odpovídají (až na malou nepřesnost) výpočtům přesného algoritmu aplikovaného na soustavu

$$\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}, \quad \text{kde} \quad \hat{\mathbf{A}} = \mathbf{T}_{k+j} \quad a \quad \hat{\mathbf{b}} = \|\mathbf{b}\|\mathbf{e}_{(1)}. \tag{3.1}$$

3.3 Srovnání simulace se skutečností

3.3.1 Experimenty

Tato sekce se bude zabývat experimenty srovnávajícími FP výpočty algoritmu metody sdružených gradientů s jejich simulacemi.

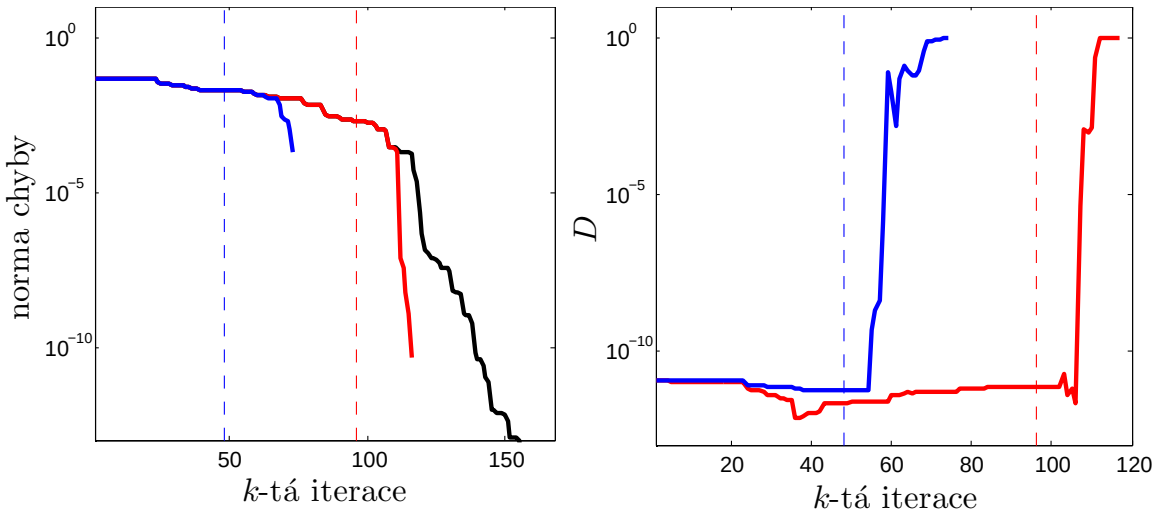
Následující experiment ukazuje, že pokud přesný algoritmus je určen algoritmem popsaným v sekci 2.1, simulace odpovídá skutečnosti až na relativní chybu řádu 10^{-10}. Jako matice $\mathbf{A}$ je zde vybrána opět matice bcsstk01.mtx (viz. [11]), jejíž velikost označíme N a vstupní vektory jsou dány (2.5).

V grafu vlevo obrázku 3.1 je černou barvou vykreslena $\mathbf{A}$-norma chyby vypočtená pomocí algoritmu CG v konečné aritmetice a červeně a modře jsou vykresleny $\hat{\mathbf{A}}$-normy chyby vypočtené pomocí přesného algoritmu aplikovaného na soustavu rovnic (3.1). Matice $\hat{\mathbf{A}}$ vznikla postupem popsaným v podkapitole 3.2 pro hodnotu parametru $k = N$ (zobrazeno modře) a pro $k = 2N$ (zobrazeno červeně). Přerušované křivky určují iteraci k pro obě simulace FP algoritmu. Modrá určuje hodnotu N a červená $2N$.

Graf vpravo obrázku 3.1 vykresluje relativní rozdíl $\mathbf{A}$-normy chyby vypočtené FP algoritmem s $\hat{\mathbf{A}}$-normou chyby vypočtenou simulací, tedy

$$D = \frac{\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}} - \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_k\|_{\hat{\mathbf{A}}}}{\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}}. \tag{3.2}$$

Modře je vykreslen relativní rozdíl odpovídající modré a černé křivce grafu vlevo a červeně relativní rozdíl odpovídající červené a černé křivce grafu vlevo.



Obrázek 3.1: Vlevo je vykreslena $\mathbf{A}$-norma chyby $\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{A}}$ vypočtená FP algoritmem (černě) a $\hat{\mathbf{A}}$-norma chyby $\|\hat{\mathbf{x}}-\hat{\mathbf{x}}_k\|_{\hat{\mathbf{A}}}$ vypočtená přesným algoritmem pro matici $\hat{\mathbf{A}}$ danou parametrem $k = N$ (modře) a $k = 2N$ (červeně). Vpravo je relativní rozdíl chyb určený vztahem (3.2) odpovídající stejným simulacím.

Z předchozích úvah vyplývá, že do kroku označeném přerušovanou čarou by křivka určená přesným algoritmem aplikovaným na (3.1) měla odpovídat křivce určené FP algoritmem pro (1.1), což tento obrázek potvrzuje. Dále by mělo platit, že β_{k+j+1} odpovídající matici $\hat{\mathbf{A}} = \mathbf{T}_{k+j}$ se rovná nule. Pro matici $\hat{\mathbf{A}}$ získanou přechozím experimentem a danou parametrem $k = N$ platí $\beta_{k+j+1} = 3.9 * 10^{-33}$ a pro k = 2N platí $\beta_{k+j+1} = 3.0 * 10^{-29}$, takže i tato vlastnost simulace platí.

3.3.2 Přesnější výpočty pomocí násobné aritmetiky

Pomocí násobné aritmetiky lze matici $\hat{\mathbf{A}}$ určit přesněji. Pokud výpočty algoritmu, pomocí něhož sestavujeme druhou část matice $\hat{\mathbf{A}}$ provedeme násobnou aritmetikou, získáme mnohem menší hodnotu β_{k+j+1}. Například pokud program zaokrouhluje na 50 platných desetinných míst, v předchozím experimentu pro k = N vyjde $\beta_{k+j+1} = 3.1 * 10^{-60}$.

Lepší aproximace FP řešení ale pomocí násobné aritmetiky nedostaneme. Relativní rozdíl chyb (3.2) v tomto případě totiž neklesne pod hranici 10^{-12}, přestože i výpočet $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ provádíme násobnou aritmetikou.

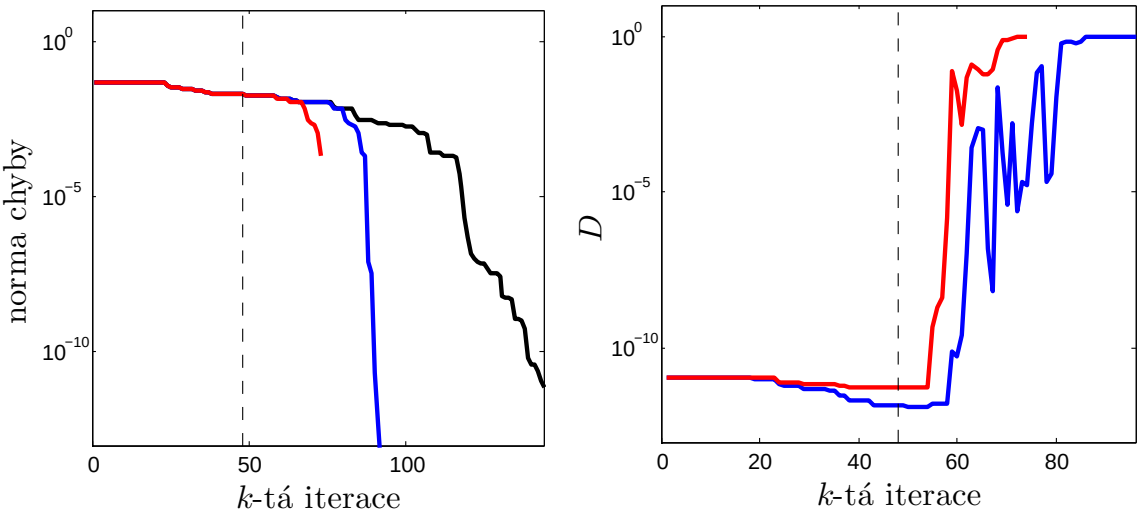
3.4 Jiné varianty simulace

Podobné výsledky při výpočtu $\mathbf{A}$-normy chyby FP algoritmem dává i přesný algoritmus aplikovaný na soustavu (3.1), kde matice $\hat{\mathbf{A}}$ vznikla následujícím postupem. Nejprve se vytvoří matice $\mathbf{T}_k$ pomocí FP algoritmu a poté se pokračuje v sestavení $\mathbf{T}_{k+j}$ pomocí přesného algoritmu. Tento algoritmus bude reortogonalizovat nově vzniklá rezidua vůči předchozím reziduím (vzniklým až při počítání přesného algoritmu), ale nebude reortogonalizovat vůči nezkonvergovaným Ritzovým vektorům $\mathbf{T}_k$. Za matici $\hat{\mathbf{A}}$ pak bude považovaná matice $\mathbf{T}_{k+j}$, pro kterou platí $\beta_{k+j+1} = 0$, kde parametr j bude obecně jiný než v předchozím případě. Vektor $\hat{\mathbf{b}}$ bude opět dán vztahem $\hat{\mathbf{b}} = \|\mathbf{b}\|\mathbf{e}_{(1)}$, kde velikost matice $\mathbf{e}_{(1)}$ je dána velikostí matice $\hat{\mathbf{A}}$.

Matici $\hat{\mathbf{A}}$, která vznikla postupem popsaným v části 3.2, budeme značit $\hat{\mathbf{A}}_G$ (G = Greenbaum) a matici, která vznikla tímto postupem, budeme značit $\hat{\mathbf{A}}_{FE}$ (FE = Finite-Exact).

Obrázek č. 3.2 byl vykreslen pro stejné vstupní hodnoty jako v předchozím případě. Graf vlevo ukazuje $\mathbf{A}$-normu chyby vypočtenou pomocí FP algoritmu aplikovaného na soustavu (1.1) (černě), $\hat{\mathbf{A}}_{FE}$-normu chyby vypočtenou přesným algoritmem (modře) a $\hat{\mathbf{A}}_G$-normu chyby vypočtenou přesným algoritmem (červeně). Parametr k byl tentokrát pro obě simulace zvolen stejně a jeho hodnota je zobrazena černě přerušovaně.

Graf vpravo je závislostí relativního rozdílu norem chyb spočteného stejným způsobem jako v předchozím případě, tj. pomocí (3.2) na počtu iterací. Červeně je vykreslen relativní rozdíl $\mathbf{A}$-normy chyby a $\hat{\mathbf{A}}_G$-normy chyby a modře rozdíl $\mathbf{A}$-normy chyby s $\hat{\mathbf{A}}_{FE}$-normou chyby.



Obrázek 3.2: Vlevo je vykreslena $\mathbf{A}$-norma chyby vypočtená FP algoritmem (černě), $\hat{\mathbf{A}}_{FE}$-norma chyby vypočtená přesným algoritmem (modře) a $\hat{\mathbf{A}}_G$-norma chyby vypočtená přesným algoritmem (červeně). Vpravo je relativní rozdíl chyb určený vztahem (3.2) pro odpovídající křivky z grafu vlevo.

Je patrné, že $\hat{\mathbf{A}}_{FE}$-norma chyby odpovídá $\mathbf{A}$-normě chyby déle než $\hat{\mathbf{A}}_G$-norma chyby. Matice $\hat{\mathbf{A}}_{FE}$ ale nesplňuje vlastnost, která byla pro matici $\hat{\mathbf{A}}$ předpokládaná. V matici $\hat{\mathbf{A}}_{FE}$ totiž existují vlastní čísla, která nejsou blízká žádnému vlastnímu číslu matice $\mathbf{A}$. V článku [4] je dokázáno, že velikost shluků matice $\hat{\mathbf{A}}_G$ nepřesáhne určitou hranici a následující obrázek ukazuje, že pro $\hat{\mathbf{A}}_{FE}$ je tato hranice překročena.

Obr. 3.3 je vypočten pro stejné vstupní hodnoty jako v předchozích případech. Je zde pro každé číslo matice $\hat{\mathbf{A}}_{FE}$ (modře) a $\hat{\mathbf{A}}_G$ (červeně) vykreslena jeho vzdálenost k nejbližšímu vlastnímu číslu matice $\mathbf{A}$, tedy

$$\min_{j=1,2,\ldots,N} |\theta_i - \lambda_j|, \quad i = 1, 2, \ldots, k+j,$$

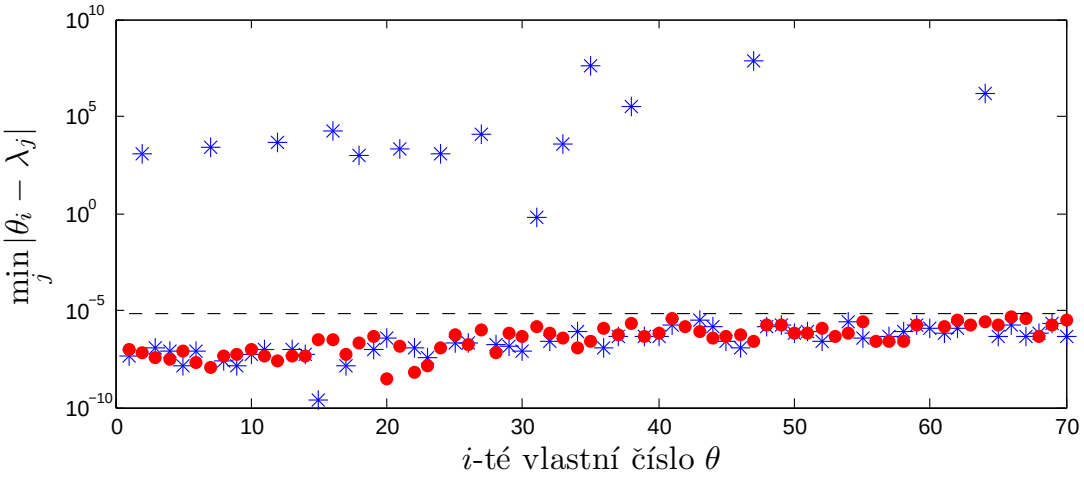
kde θ_i jsou vlastní čísla matice $\hat{\mathbf{A}}_G$ (resp. $\hat{\mathbf{A}}_{FE}$) a λ_j jsou vlastní čísla matice $\mathbf{A}$. Černě je vykreslena hranice

$$10\varepsilon\|\mathbf{A}\|,$$

která byla odvozena z článku [4] a která udává odhad velikosti shluku matice $\hat{\mathbf{A}}_G$.

Je patrné, že vzdálenost vlastních čísel matice $\hat{\mathbf{A}}_{FE}$ od vlastních čísel $\mathbf{A}$ převyšuje tuto hranici o 15 řádů. Jinak řečeno, vlastní čísla této matice neleží ve shlucích okolo vlastních čísel $\mathbf{A}$.

Hlavním přínosem článku [4] bylo ukázání, že rekurence CG implementované v konečné aritmetice se stále chovají jako rekurence CG (aplikované na jiný systém rovnic). Z toho plyne, že některé vlastnosti metody sdružených gradientů v konečné aritmetice můžeme určit jako



Obrázek 3.3: Vzdálenost θ_i od nejbližšího vlastního čísla λ_j vykreslená pro hodnoty parametru $i = 1, 2, \ldots, k + j$. Červeně jsou vykresleny vzdálenosti pro vlastní čísla matice $\hat{\mathbf{A}}_G$ a modré značky odpovídají vzdálenostem vlastních čísel $\hat{\mathbf{A}}_{FE}$ od λ_j.

vlastnosti metody počítající $\hat{\mathbf{A}}_G\hat{\mathbf{x}} = \hat{\mathbf{b}}$.

Tuto myšlenku ukazuje i náš experiment, tedy některé vlastnosti výpočtů aplikovaných na $\hat{\mathbf{A}}_{FE}$ odpovídají vlastnostem FP výpočtů aplikovaných na $\mathbf{A}$. Otázkou ale je, zda matice $\hat{\mathbf{A}}_{FE}$ je užitečná pro další experimenty či teoretické účely. Matice $\hat{\mathbf{A}}_G$ totiž díky svým vlastnostem umožňuje další zamyšlení nad problémem.

Této otázce se věnuje i článek [16], jehož výsledky připomínají zpětnou analýzu zaokrouhlovacích chyb, přestože se o standardní analýzu nejedná. Je zde ukázáno, že aplikováním přesného Lanczosova algoritmu na

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{T}_k & \mathbf{o} \\ \mathbf{o} & \mathbf{A} \end{bmatrix} + \mathbf{H}_k, \quad \hat{\mathbf{v}} = \begin{bmatrix} \mathbf{o} \\ \mathbf{v} \end{bmatrix},$$

kde $\mathbf{H}_k$ je matice určená perturbacemi, získáme po k krocích matici $\mathbf{T}_{k,k+1}$, která odpovídá matici získané FP algoritmem aplikovaným na $\mathbf{A}$ a vektor $\mathbf{v}$. Sami autoři ovšem naznačují, že není zřejmé, jak tento výsledek chápat a interpretovat. Jeho další použití bude vyžadovat hlubší zamyšlení.

Kapitola 4

Model CG v konečné aritmetice

4.1 Model FP výpočtů

4.1.1 Konstrukce modelu

V kapitole 3 bylo ukázáno, že FP výpočty algoritmu metody sdružených gradientů až na malou nepřesnost odpovídají přesným výpočtům aplikovaným na soustavu $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ danou (3.1). Bylo ukázáno, že matice $\hat{\mathbf{A}}$ má všechna vlastní čísla umístěna ve shlucích okolo vlastních čísel $\mathbf{A}$. Šlo o simulaci FP výpočtů, která vznikla aposteriorní konstrukcí systému, tedy na základě dat spočtených při FP výpočtu se sestrojila matice $\hat{\mathbf{A}}$, pomocí níž simulujeme výpočty CG v konečné aritmetice.

Tato část práce se zabývá modelem výpočtů provedených v konečné aritmetice. Tento model je vytvořen na základě myšlenky uvedené v kapitole 3, ale jde o apriorní konstrukci systému, tzn. systém je sestrojen pouze na základě znalosti spektrálních vlastností matice $\mathbf{A}$ a projekcí pravé strany do ortonormální báze vlastních vektorů $\mathbf{A}$. Platí, že výpočty přesného algoritmu aplikovaného na tento systém ($\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$) jsou podobné výpočtům FP algoritmu aplikovaného na systém $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Model je sestrojen na základě článku [6]. Autoři zde chtěli ukázat, že přesný výpočet aplikovaný na jakoukoliv matici $\bar{\mathbf{A}}$, která má vlastní čísla rozprostřena okolo vlastních čísel $\mathbf{A}$, je podobný FP výpočtu aplikovaného na matici $\mathbf{A}$. To je tedy dalším rozdílem mezi modelem a simulací, protože při simulaci jde o konkrétní matici.

V následujících experimentech je matice $\bar{\mathbf{A}}$ dána předpisem

$$\bar{\mathbf{A}} = \mathrm{diag}[\lambda_{11}, \lambda_{12}, \ldots, \lambda_{1m}, \lambda_{21}, \ldots, \lambda_{2m}, \ldots, \lambda_{N1}, \ldots, \lambda_{Nm}], \tag{4.1}$$

kde λ_{ij} jsou vlastní čísla uvnitř shluku odpovídajícímu i-tému vlastnímu číslu matice $\mathbf{A}$. Tyto

vlastní čísla jsou rovnoměrně rozložena na intervalu velikosti χ a jsou dána vztahem

$$\lambda_{ij} = \lambda_i + \frac{j - \frac{m+1}{2}}{m-1}\chi, \quad j = 1, 2, \ldots, m, \quad m = 7, \quad \chi = 10\varepsilon\|\mathbf{A}\|, \tag{4.2}$$

kde m je počet vlastních čísel v shluku.

Vektor $\bar{\mathbf{b}}$ je určen pomocí vektoru $\mathbf{b}$ následujícím způsobem

$$\bar{\mathbf{b}} = [b_{11}, b_{12}, \ldots, b_{1m}, b_{21}, \ldots, b_{2m}, \ldots, b_{N1}, \ldots, b_{Nm}]^T. \tag{4.3}$$

Hodnoty $b_{i1}, \ldots, b_{im}$ odpovídající číslu b_i se rovnají a platí, že

$$\sum_{j=1}^{m} b_{ij}^2 = (U^T b)_i^2, \tag{4.4}$$

kde sloupce U odpovídají jednotkovým vlastním vektorům matice $\mathbf{A}$.

4.1.2 Experimenty

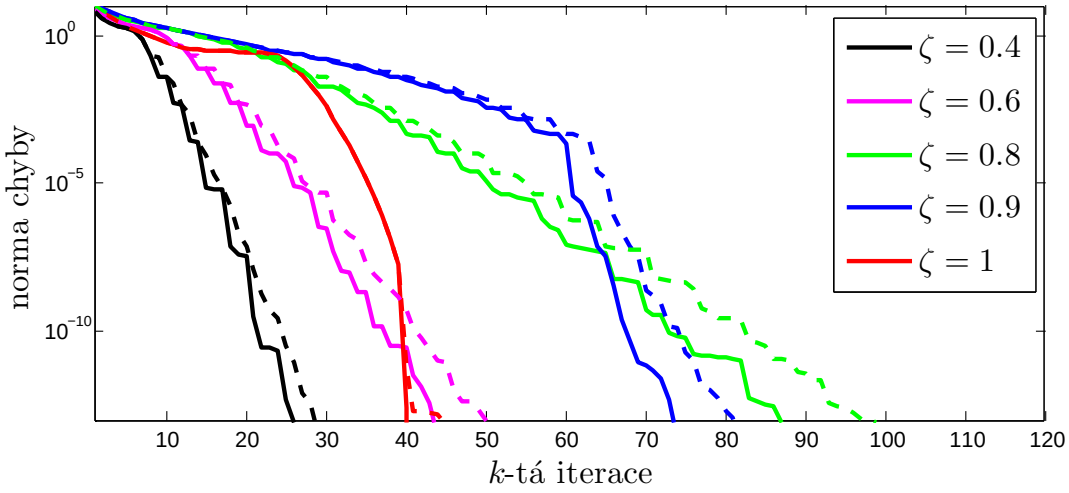
Následující experiment vykresluje $\mathbf{A}$-normu chyby vypočtenou algoritmem provedeným v konečné aritmetice a aplikovaným na matici $\mathbf{A}$ popsanou v části 2.5.3 a danou vztahy (2.9) a (2.10) pro velikost matice $N = 40$ a vhodný parametr ζ. Vektory $\mathbf{b}$ a $\mathbf{x}_0$ jsou zvoleny podle (2.1). Dále je zde vykreslena $\bar{\mathbf{A}}$-norma chyby získaná přesným algoritmem aplikovaným na matici $\bar{\mathbf{A}}$ (4.1, 4.2) a vektor $\bar{\mathbf{b}}$ (4.3, 4.4), které jsou vypočtené z $\mathbf{A}$ a $\mathbf{b}$. Vektor $\bar{\mathbf{x}}_0$ byl i zde zvolen jako nulový.

V grafu na obr. 4.1 jsou vykresleny křivky vypočtené pro různé hodnoty parametru ζ. Plnou čarou jsou označeny $\mathbf{A}$-normy chyby vypočtené v konečné aritmetice. Tyto $\mathbf{A}$-normy chyby jsou vykreslené již v grafu 2.5. Přerušovaně jsou označeny $\bar{\mathbf{A}}$-normy chyby vypočtené přesným algoritmem.

Z obrázku je patrná určitá podobnost konvergenčního chování FP výpočtů a přesných výpočtů aplikovaných na modelový systém. Například obě křivky odpovídající $\zeta = 0.8$ mají etapy, kdy nedochází k přibližování aproximace řešení k přesnému řešení vystřídané prudkým poklesem chyby.

4.2 Konstrukce dalšího modelu

Tato část práce se zabývá hledáním modelu, který by odpovídal lépe. V předchozím modelu byly složky vektoru $\bar{\mathbf{b}} = [b_{i1}, b_{i2}, \ldots, b_{im}]$ stejné, což odpovídá stejným vahám v i-tém shluku distribuční funkce $\omega_G(\lambda)$, dané (1.27). Platí, že polynomy sestrojené přesnou metodou sdružených gradientů jsou ortogonální vůči skalárnímu součinu, který je dán distribuční funkcí $\omega(\lambda)$, (1.27).



Obrázek 4.1: $\bar{\mathbf{A}}$-norma chyby vypočtená přesným algoritmem CG (přerušovaně) a $\mathbf{A}$-norma chyby vypočtená algoritmem v konečné aritmetice (plnou čarou) v závislosti na kroku k a pro různé rozložení vlastních čísel ζ. Inspirováno článkem [6].

Pokud tedy počítáme simulaci aplikováním přesných sdružených gradientů na soustavu $\hat{\mathbf{A}}_G\hat{\mathbf{x}} = \hat{\mathbf{b}}$, získáme polynomy, které jsou ortogonální vůči skalárnímu součinu, který je dán jinou (porušenou) distribuční funkcí $\omega_G(\lambda)$. Tato distribuční funkce bude odpovídat funkci $\hat{\omega}(\lambda)$ sestrojené FP algoritmem a ukázané v sekci 2.4 na obrázku 2.3. Pomocí souvislosti CG a Gaussovy kvadratury víme, že tuto $\omega_G(\lambda)$ lze aproximovat pomocí funkce $\omega_G^k(\lambda)$, která je dána předpisem (1.31).

Následující experiment vykresluje v obr. 4.2 aproximace vah získané FP algoritmem aplikovaným na (1.1). Tyto aproximace budou určeny jako váhy (1.29) dané přesným algoritmem aplikovaným na větší soustavu $\hat{\mathbf{A}}_G\hat{\mathbf{x}} = \hat{\mathbf{b}}$.

Matice $\hat{\mathbf{A}}_G$ zde byla sestrojena způsobem uvedeným v sekci 3.2 pro $\mathbf{A}$ danou (2.9, 2.10) s $N = 24$ a $\zeta = 0.9$. Vektory $\mathbf{b}$ a $\mathbf{x}_0$ byly zvoleny stejně jako v předchozím případě, tedy (2.1). Pro iteraci k, která udává velikost matice $\mathbf{T}_k$ vypočtené FP algoritmem, platí $k = 12N$, kde N je velikost matice $\mathbf{A}$.

Vykreslené váhy jsou spočteny pomocí

$$\omega_i^G = (\mathbf{e}_{(1)}^T \hat{\mathbf{u}}_i^{k+j})^2, \tag{4.5}$$

kde $\hat{\mathbf{u}}_i^{k+j}$ značí i-tý sloupec matice $\hat{\mathbf{U}}_{k+j}$ získané pomocí spektrálního rozkladu matice $\hat{\mathbf{T}}_{k+j}$, tedy $\hat{\mathbf{T}}_{k+j} = \hat{\mathbf{U}}_{k+j}\hat{\mathbf{\Lambda}}_{k+j}\hat{\mathbf{U}}_{k+j}^T$. Matice $\hat{\mathbf{T}}_{k+j}$ je tridiagonální matice, kterou vypočítal přesný algoritmus metody sdružených gradientů aplikovaný na matici $\hat{\mathbf{A}}_G$ a vektor $\hat{\mathbf{b}}$ po $k+j$ krocích, kde $k+j$ je velikost matice $\hat{\mathbf{A}}_G$. Tyto váhy jsou vykresleny modrými značkami.

Červenou značkou je vykreslena váha

$$\omega_i = (\mathbf{e}_{(1)}^T \mathbf{u}_i^N)^2, \tag{4.6}$$

kde $\mathbf{u}_i^N$ je sloupec matice $\mathbf{U}_N$ dané spektrálním rozkladem $\mathbf{T}_N = \mathbf{U}_N \mathbf{\Lambda}_N \mathbf{U}_N^T$, kde $\mathbf{T}_N$ vznikla přesným algoritmem aplikovaným na původní soustavu (1.1) po N krocích.

Protože tyto váhy jsou spočteny přesným algoritmem a dostatečným množstvím kroků, měly by odpovídat vahám daným (1.26), tedy

$$\omega_i^G = \left(\frac{\hat{\mathbf{b}}^T \hat{\mathbf{u}}_i}{\|\hat{\mathbf{b}}\|} \right)^2, \quad \omega_i = \left(\frac{\mathbf{b}^T \mathbf{u}_i}{\|\mathbf{b}\|} \right)^2,$$

kde $\hat{\mathbf{u}}_i$ odpovídá i-tému sloupci matice $\hat{\mathbf{U}}$ získané rozkladem $\hat{\mathbf{A}}_G = \hat{\mathbf{U}} \hat{\mathbf{\Lambda}} \hat{\mathbf{U}}^T$ a $\mathbf{u}_i$ odpovídá sloupci matice $\mathbf{U}$ získané z rozkladu $\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$.

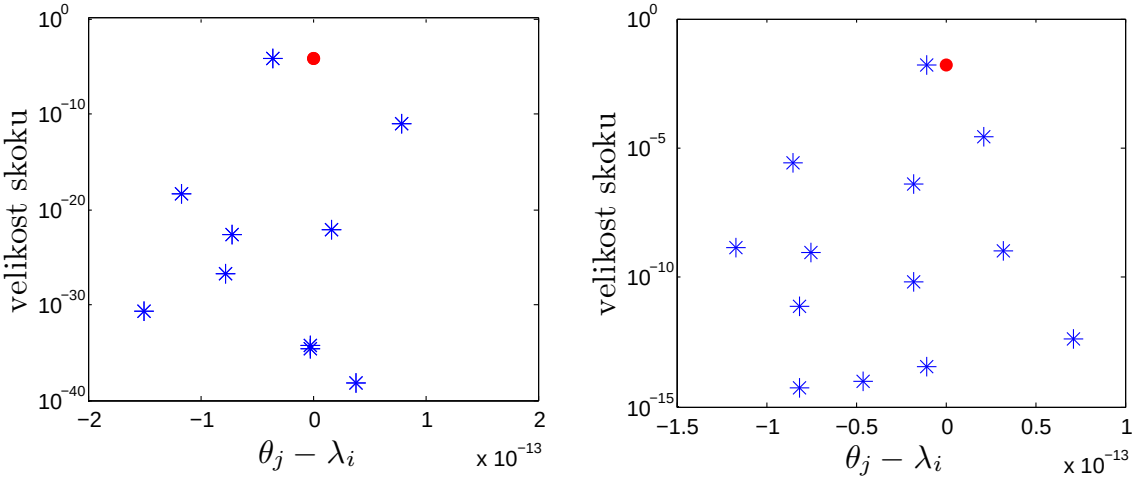
Graf vlevo ukazuje vlastní čísla θ_j (λ_i) a jim příslušné váhy ω_j^G (ω_i) patřící do shluku kolem čtvrtého vlastního čísla matice $\mathbf{A}$. Souřadnice (x, y) jednotlivých bodů jsou dány následovně:

$$x = \theta_j - \lambda_i, \quad (x = 0), \quad j = 1, 2, \ldots,$$

$$y = \omega_j^G, \quad (y = \omega_i), \quad j = 1, 2, \ldots,$$

kde $i = 4$.

V pravém grafu jsou vykreslena vlastní čísla θ_j a λ_i patřící do 15. shluku. Grafy odpovídající ostatním shlukům byly velmi podobné.



Obrázek 4.2: Modré body představují jednotlivá vlastní čísla θ_j matice $\hat{\mathbf{T}}_{k+j}$ patřící do shluku kolem i-tého vlastního čísla λ_i matice $\mathbf{A}$. Toto číslo je označeno červeně. Na ose x je vykreslen rozdíl $\theta_j - \lambda_i$ a na ose y váhy dané (4.5), popř. (4.6).

Z obrázku 4.2 je patrné, že váhy jednotlivých čísel ve shluku se velmi liší. Předchozí model byl ale sestaven tak, aby se rovnaly. Proto jsme se rozhodli sestavit model, který by vytvořil vektor $\bar{\mathbf{b}}$ tak, aby výsledné váhy lépe odpovídaly těm na obrázku.

Tento model sestaví matici $\bar{\mathbf{A}}$ stejně jako předchozí model, tedy pomocí (4.1, 4.2), ale vektor $\bar{\mathbf{b}}$ je jiný. Tento nový vektor pravých stran budeme značit $\bar{\bar{\mathbf{b}}}$ a je dán vztahy

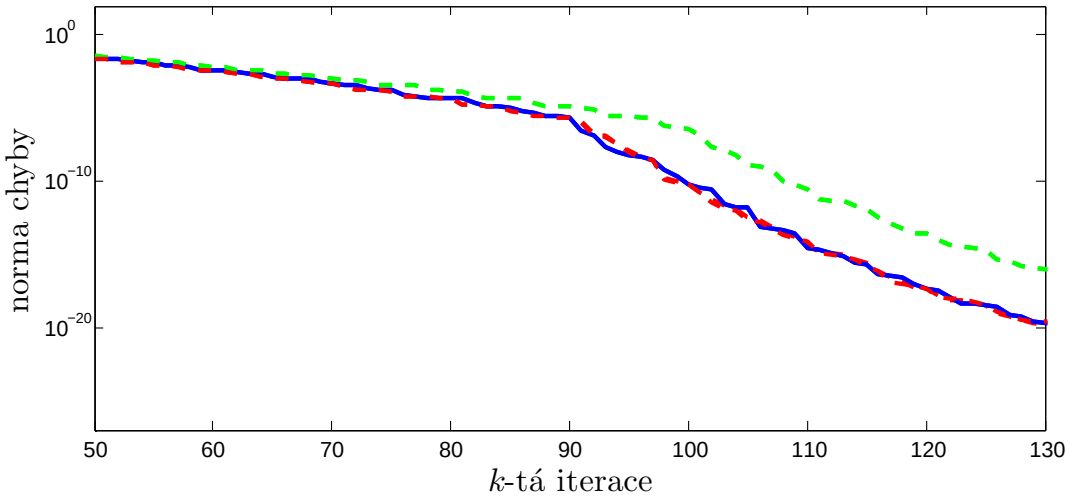
$$\bar{\bar{b}}_{ij} = \text{rand}.\vartheta, \quad j = 2, 3, \ldots, m, \tag{4.7}$$

$$\bar{\bar{b}}_{i1} = (\mathbf{U}^T\mathbf{b})_i^2 - \sum_{j=2}^{m} \bar{\bar{b}}_{ij} \tag{4.8}$$

pro $i = 1, 2, \ldots, N$. Z tohoto předpisu vyplývá, že kromě $\bar{\bar{b}}_{i1}$ jsou hodnoty $\bar{\bar{b}}_{ij}$ dány násobkem náhodného čísla s parametrem ϑ, kde ϑ je vhodné malé číslo.

K vytvoření grafu 4.3 bylo použito matice $\mathbf{A}$ dané vztahy (2.9, 2.10) a parametry $N = 40$ a $\zeta = 0.9$, vektorů $\mathbf{b}$ a $\mathbf{x}_0$ danými (2.5), matice $\bar{\mathbf{A}}$ sestavené pomocí (4.1, 4.2), vektoru $\bar{\mathbf{b}}$ sestaveného pomocí (4.3, 4.4) a také vektoru $\bar{\bar{\mathbf{b}}}$ daného (4.7, 4.8) s $\vartheta = 10^{-4}$.

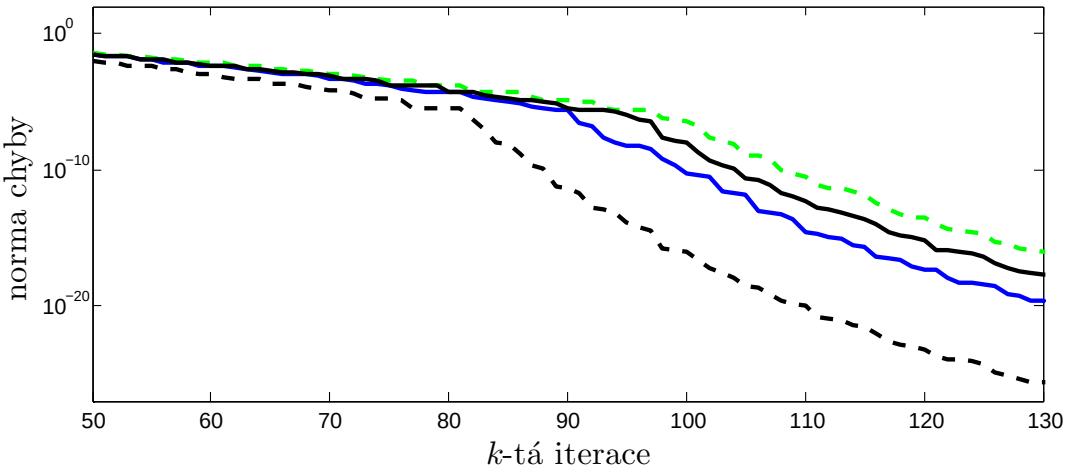
V grafu je vykreslena $\mathbf{A}$-norma chyby vypočtená algoritmem provedeným v konečné aritmetice (modře), $\bar{\mathbf{A}}$-norma chyby vypočtená přesným algoritmem aplikovaným na soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ (zeleně) a $\bar{\mathbf{A}}$-norma chyby vypočtená přesným algoritmem aplikovaným na soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\bar{\mathbf{b}}}$ (červeně).



Obrázek 4.3: $\mathbf{A}$-norma chyby vypočtená FP algoritmem (modře), $\bar{\mathbf{A}}$-norma chyby vypočtená přesným algoritmem aplikovaným na soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ (zeleně) nebo soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\bar{\mathbf{b}}}$ (červeně).

Přestože obrázek naznačuje, že přesné výpočty systému $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\bar{\mathbf{b}}}$ modelují výpočty v konečné aritmetice lépe, není tomu tak vždy. Záleží hlavně na volbě parametru ϑ, jak na-

značuje obr. 4.4. Vstupní hodnoty tohoto obrázku byly zvoleny stejně jako v předchozím případě, ale s volbou $\vartheta = 10^{-2}$ (zobrazeno černě) a $\vartheta = 10^{-10}$ (zobrazeno černě přerušovaně). FP výpočty (resp. výpočty předchozího modelu) jsou opět zobrazeny modře (resp. zeleně).



Obrázek 4.4: **A**-norma chyby vypočtená FP algoritmem (modře), $\bar{\mathbf{A}}$-norma chyby vypočtená přesným algoritmem aplikovaným na soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ (zeleně), soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \tilde{\bar{\mathbf{b}}}$ s $\vartheta = 10^{-2}$ (černě) a soustavu $\bar{\mathbf{A}}\bar{\mathbf{x}} = \tilde{\bar{\mathbf{b}}}$ s $\vartheta = 10^{-10}$ (černě přerušovaně).

Otázkou tedy zůstává, zda lze pomocí spektrálních vlastností matice **A** předem určit hodnotu ϑ a zda vůbec lze apriorní model FP výpočtů vytvořit.

Závěr

Diplomová práce byla zaměřena na analýzu chování metody sdružených gradientů v konečné aritmetice. Samotné analýze předchází popis metody při přesném počítání zaměřený na vlastnosti, které jsou důležité pro analýzu v konečné aritmetice, viz. kapitola 1. Je zde nastíněna jak matematická podstata metody, tak její algoritmické realizace. První kapitola je shrnutím informací získaných z článků a knih uvedených na konci práce s výjimkou důkazů ekvivalence jednotlivých algoritmů, viz. věty 1 a 2.

V kapitole 2 jsou popsány a demonstrovány základní jevy, k nimž dochází vlivem konečné aritmetiky. Při popisu těchto jevů hraje důležitou roli vztah mezi CG a Gaussovou kvadraturou, který je pro celou tuto práci velmi důležitý a který je v této i předchozí kapitole uveden. Navíc jsou některé z těchto jevů demonstrovány pro různé algoritmické realizace. Jde tedy o jakési porovnávání těchto realizací CG.

Za zmínku stojí náš experiment uvedený v části 2.7.2, který ukazuje, že lokální ortogonalita směrového a reziduového vektoru je lépe zachovaná pro HS verzi metody, což může být důvod rychlejší konvergence této verze metody.

Kapitola 3 je založena na článku [4], kde je dokázáno, že výpočty CG provedené v konečné aritmetice je možné simulovat pomocí přesných výpočtů CG aplikovaných na jiný systém rovnic. Je zde uveden experiment, který potvrzuje správnost závěru tohoto článku. Tento systém se však sestrojuje poměrně obtížným postupem, proto jsme uvedli i jiný systém, jehož sestrojení je výrazně lehčí a experimentem ukázali, že i pro náš systém platí závěr článku [4]. Otázkou ale zůstává, zda je tento nový systém vhodný pro další zamyšlení nad problémem.

Poslední kapitola se věnuje vytvoření apriorního modelu FP výpočtů CG. V kapitole jsou experimenty zabývající se modelem uvedeném v článku [6]. Tzn. přesné výpočty jsou aplikovány na matici $\bar{\mathbf{A}}$, jejíž vlastní čísla jsou rovnoměrně rozložena kolem vlastních čísel $\mathbf{A}$, a vektor $\bar{\mathbf{b}}$, jehož komponenty odpovídající jedné složce $\mathbf{b}$ se rovnají.

Uvedli jsme zde i naší obdobu tohoto modelu, tedy model s vektorem $\bar{\bar{\mathbf{b}}}$, který má jednu komponentu mnohem větší než ostatní komponenty odpovídající stejnému vlastnímu číslu. Tento model by mohl odpovídat FP výpočtům lépe. Závisí ovšem na parametrech, které zatím neumíme předem odhadnout. Otázkou zůstává, zda tyto parametry předem odhadnout lze.

Literatura

[1] G. Dahlquist and Å. Björck. *Numerical methods in scientific computing. Vol. I.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.

[2] W. Gautschi. *Ortogonal polynomials: computation and approximation.* Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2004.

[3] G. H. Golub and C. F. Van Loan. *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[4] A. Greenbaum. Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.*, 113:7–63, 1989.

[5] A. Greenbaum. Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Matrix Anal. Appl.*, 18(3):535–551, 1997.

[6] A. Greenbaum and Z. Strakoš. Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.*, 13(1):121–137, 1992.

[7] M. H. Gutknecht and Z. Strakoš. Accuracy of two three-term and three two-term recurrences for Krylov space solvers. *SIAM J. Matrix Anal. Appl.*, 22(1):213–229 (electronic), 2000.

[8] L. A. Hageman and D. M. Young. *Applied iterative methods.* Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1981. Computer Science and Applied Mathematics.

[9] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436, 1952.

[10] Y. Hu and T. Davis. Matlab function. `http://www.cise.ufl.edu/research/sparse/matrices/Oberwolfach/LF10.html`, 2011.

[11] Y. Hu and T. Davis. Matlab function. `http://www.cise.ufl.edu/research/sparse/matrices/HB/bcsstk01.html`, 2011.

[12] G. Meurant. *The Lanczos and conjugate gradient algorithms*, volume 19 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[13] G. Meurant and Z. Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numer.*, 15:471–542, 2006.

[14] G. Meurant and P. Tichý. On computing quadrature-based bounds for the a-norm of the error in conjugate gradients. *submitted to Numerical Algorithms*, in December, 2011.

[15] D. P. O'Leary, Z. Strakoš, and P. Tichý. On sensitivity of Gauss-Christoffel quadrature. *Numer. Math.*, 107(1):147–174, 2007.

[16] C. C. Paige. An augmented stability result for the Lanczos Hermitian matrix tridiagonalization process. *SIAM J. Matrix Anal. Appl.*, 31(5):2347–2359, 2010.

[17] J. K. Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. In *Large sparse sets of linear equations (Proc. Conf., St. Catherine's Coll., Oxford, 1970)*, pages 231–254. Academic Press, London, 1971.

[18] H. Rutishauser. *Theory of gradient methods*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Basel-Stuttgart, 1959.

[19] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2002.

[20] Z. Strakoš. On the real convergence rate of the conjugate gradient method. *Linear Algebra Appl.*, 154/156:535–549, 1991.

[21] Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.*, 13:56–80 (electronic), 2002.

[22] J. Tebbens, I. Hnětynková, M. Plešinger, Z. Strakoš, and P. Tichý. *Analýza metod pro maticové výpočty: Zakladní metody*. submitted to Matfyzpress, 2011.

[23] P. Tichý. *O některých otevřených problémech v Krylovovských metodách*. PhD thesis, Faculty of Mathematics and Physics, Charles University, 2002.

[24] D. S. Watkins. *Fundamentals of matrix computations*. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, 2002. Second editon.